Help Center

Help Center

1. Get started

A step-by-step guide to building your first automation:

Learn the basics

Get to know Make, how automation works, and the meaning of APIs

Create your first scenario

Build your first automation with this guided walkthrough

Expand your scenario

Take your automation further by adding essential Make tools

1.1. Learn the basics

Welcome to Make! This guide will introduce you to automation and show you how to create your own workflows that save time and reduce manual work.

What is automation?

Automation is about using technology to perform tasks automatically, reducing the need for manual work. It frees up your valuable time and resources by handling routine work for you. From simple day-to-day tasks to complex business processes, automation helps you reduce errors and focus on more important activities.

Benefits of automation

Automation brings many advantages to your work:

- **Save time**: Let technology handle repetitive tasks while you focus on what matters
- **Reduce errors**: Automated processes run consistently without human mistakes
- **Improve efficiency**: Complete more work with the same resources
- **Scale easily**: Handle growing workloads without adding more manual steps

With Make, you can connect your favorite apps and create custom automations that work for you - all without writing a single line of code.

1.1.1. What is Make?

Make is a powerful automation platform that allows you to connect various applications and services to automate workflows and tasks. With Make, you can automate processes just like a developer, all without any coding expertise.

What can I do using Make?

The possibilities with Make are nearly endless. You can improve how your business works by automating many different tasks.

Here are a few examples of common manual tasks you can automate:

- Notify the sales team when a customer fills out a form expressing interest in your products and services.
- Automatically send a welcome email to new employees one week before they join the company.
- Add new prospects to your CRM once the marketing team has qualified them as leads.
- Generate quarterly PDF reports of won sales opportunities.

Make is also great at handling complex end-to-end processes that make your business run more smoothly:

- Set up complete employee onboarding that automatically creates IT accounts, requests laptops, orders access cards, and sends welcome emails.
- Handle customer questions by automatically creating tickets and using AI to sort them based on what they're asking about, their location, and which language they're using.
- Process invoices automatically after orders are fulfilled and products are shipped.

While you focus on growing your business, Make takes care of repetitive tasks in the background.

How does it work?

To automate tasks or build complex processes, you need to communicate with different applications and systems. For instance, extracting information from a Google or Microsoft spreadsheet requires connecting to their services. These services offer what we call APIs.

In the following pages, you'll learn how Make handles these connections for you, opening up automation to everyone.

1.1.2. What's an API?

An API, or Application Programming Interface, is a set of rules and protocols that allows different software applications to communicate with each other. It specifies how requests and data should be structured so that applications are able to interact smoothly.

> Think of an API like a universal remote control. Just as a remote lets you control different devices without knowing how they work inside, an API lets applications work together without knowing what's happening behind the scenes.
>
> In this analogy:
>
> - The remote control is like the API interface.
> - The buttons on the remote are like API endpoints.
> - Users, pressing buttons, are like a second application making requests.
> - The TV or device responding is like an application fulfilling those requests.
>
> Just as you press a specific button to change the channel without knowing how the TV processes that signal, one application can use another application's API to request data or perform actions without understanding its internal workings.

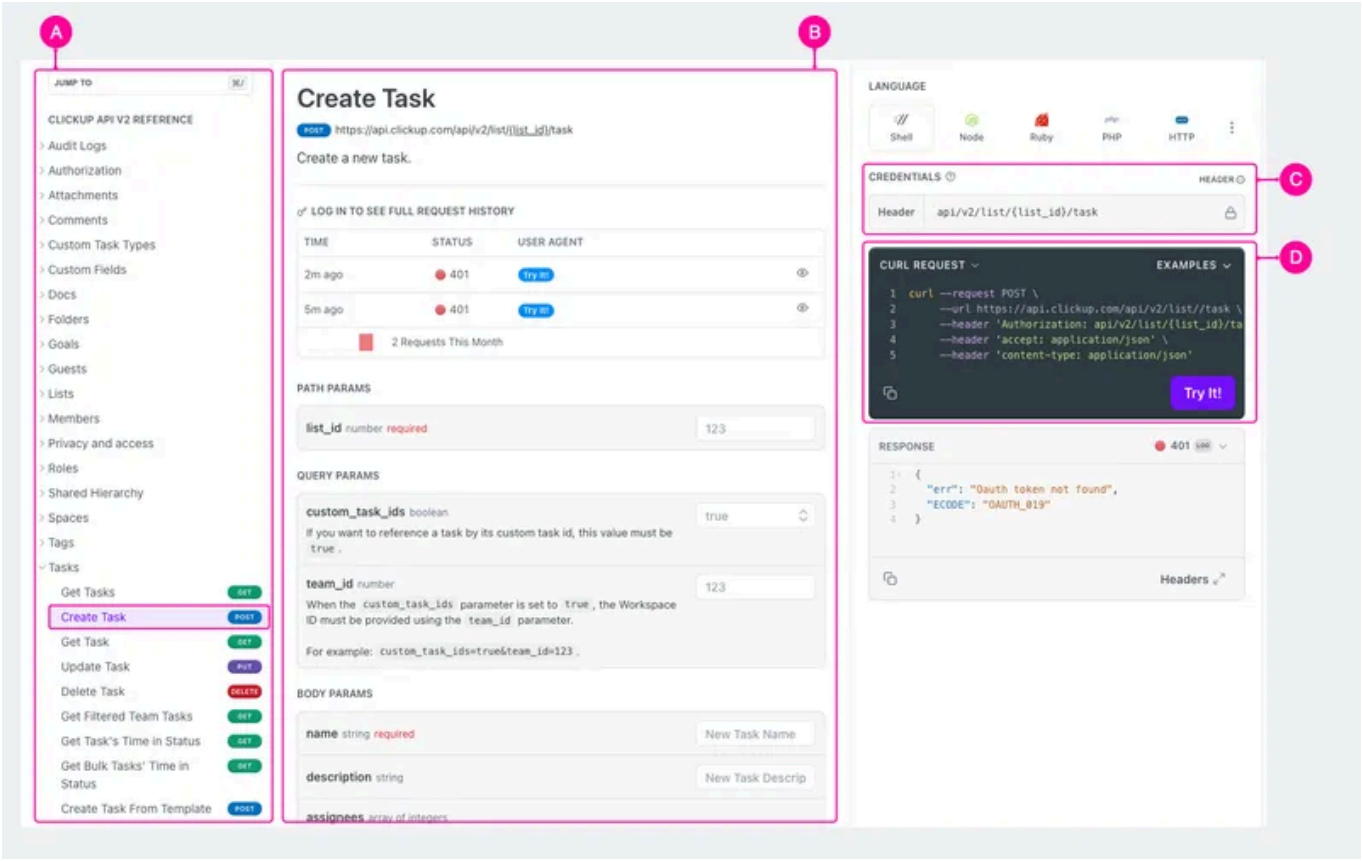
Many SaaS (Software as a Service) applications provide APIs. For example:

- ClickUp API provides numerous endpoints for creating tasks, searching tasks, adding comments, attaching files, and more.
- Gmail API lets you send emails, mark them as read, or move them to a different folder in your inbox.
- Notion API lets you update pages, create databases, retrieve information for database items, and more.

An API consists of a list of services known as **API endpoints**. Each endpoint performs a specific action. Make is a solution that **orchestrates calls to any API**, allowing you to define any type of process. It's important to know that an API endpoint requires **input** (to understand what action to perform) and provides **output** (to deliver the result back to the calling application).

Just like your remote needs you to press the correct button (input) before the TV can change to the channel you want (output), an API needs specific information before it can return the desired result.

The **ClickUp API documentation** gives a clear example of why APIs are essential for Make.



In the ClickUp API documentation, you can find:

- List of available endpoints: A comprehensive catalog of endpoints available through ClickUp's API, including options like "Get Tasks," "Create Task," and more. (**A**)
- Instructions for the "Create a Task" endpoint: Detailed guidelines on how to use the "Create a Task" endpoint. Required inputs: name of the task, status, and other necessary parameters. Expected output: comprehensive information about the created task. (**B**)
- Endpoint access location: Information on where and how to access this specific endpoint. (**C**)
- Practical examples of endpoint calls: Real examples demonstrating how to call the "Create a Task" endpoint effectively. (**D**)

APIs are a convenient way to help applications exchange information and services. Make leverages these APIs from various SaaS applications, acting as a solution that orchestrates

calls to any API. Make already integrates with over 2,000 APIs—meaning 2,000 different applications—and this number is constantly increasing.

1.2. Create your first scenario

The best way to understand Make is to try it yourself! This guide will walk you through creating your first Make scenario—an automated workflow that will send notifications to a sales team. By the end, you'll understand Make's core concepts and be ready to build your own scenarios.

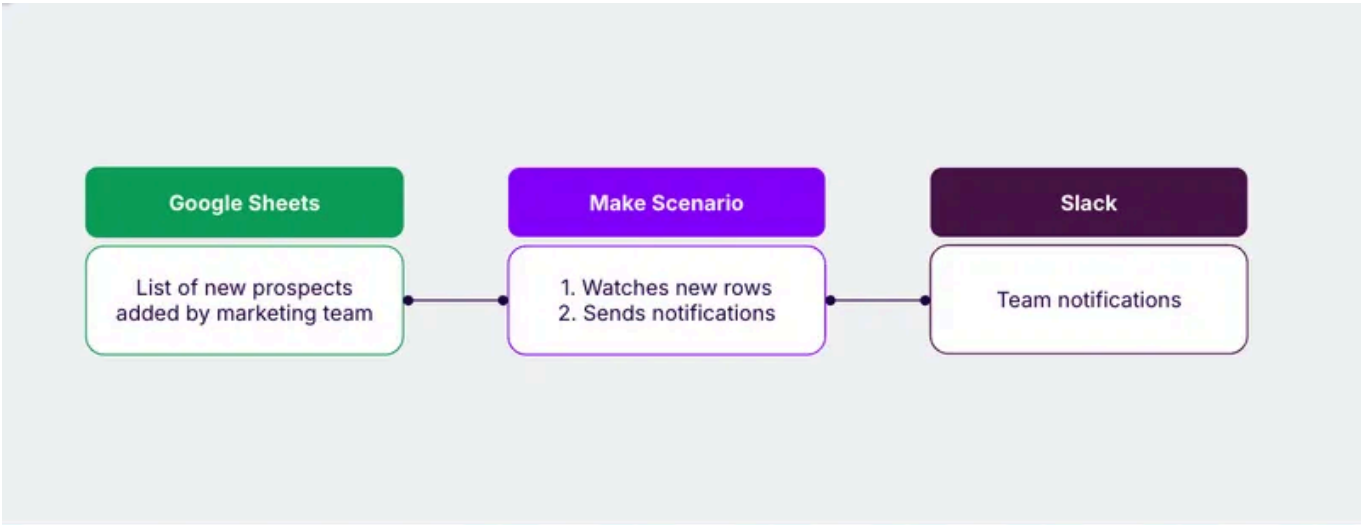
> A scenario is a series of modules that indicate how data should be transferred and transformed between apps/services. In Make, each **team** can create up to 100 scenarios per day.

What you'll build

You'll create a notification system that bridges the gap between marketing and sales teams. Imagine this business challenge: your marketing team adds new prospects to a spreadsheet, and your sales team needs immediate notifications.

Your Make scenario will:

- Monitor a Google Sheet for new prospect entries by the marketing team.
- Send Slack notifications to the sales team when new prospects are added.



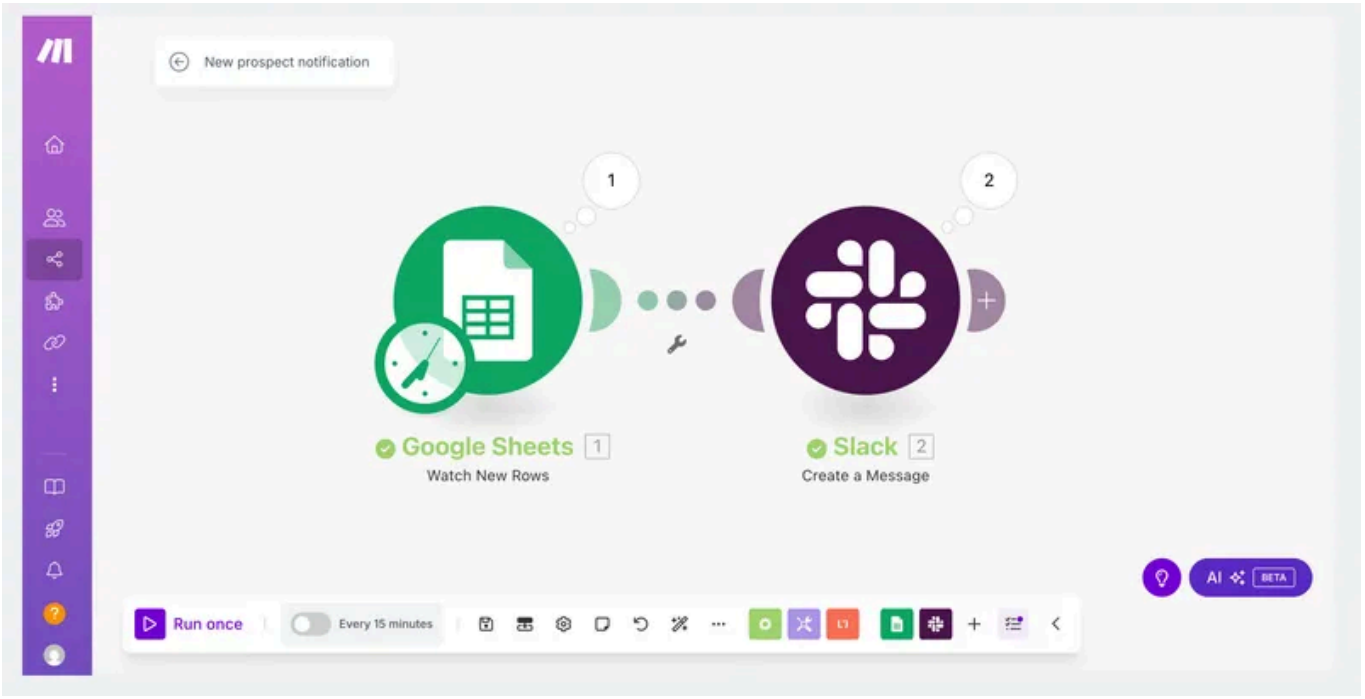
What you'll need

- A Make account (**Create one here**)

- A Slack account and workspace (**Sign up for Slack**)
- A Google account (**Create an account**)

All these accounts are free to create, including a Make account. While this guide uses Slack and Google Sheets, you can adapt these steps to work with your preferred applications.

Here's a preview of what your finished scenario will look like:



1.2.1. Step 1. Plan your scenario

Successful automation begins with a clear understanding of what needs to be automated. For any Make scenario, the first thing you do is identify what applications to connect and what actions to perform.

Define the applications you'll connect

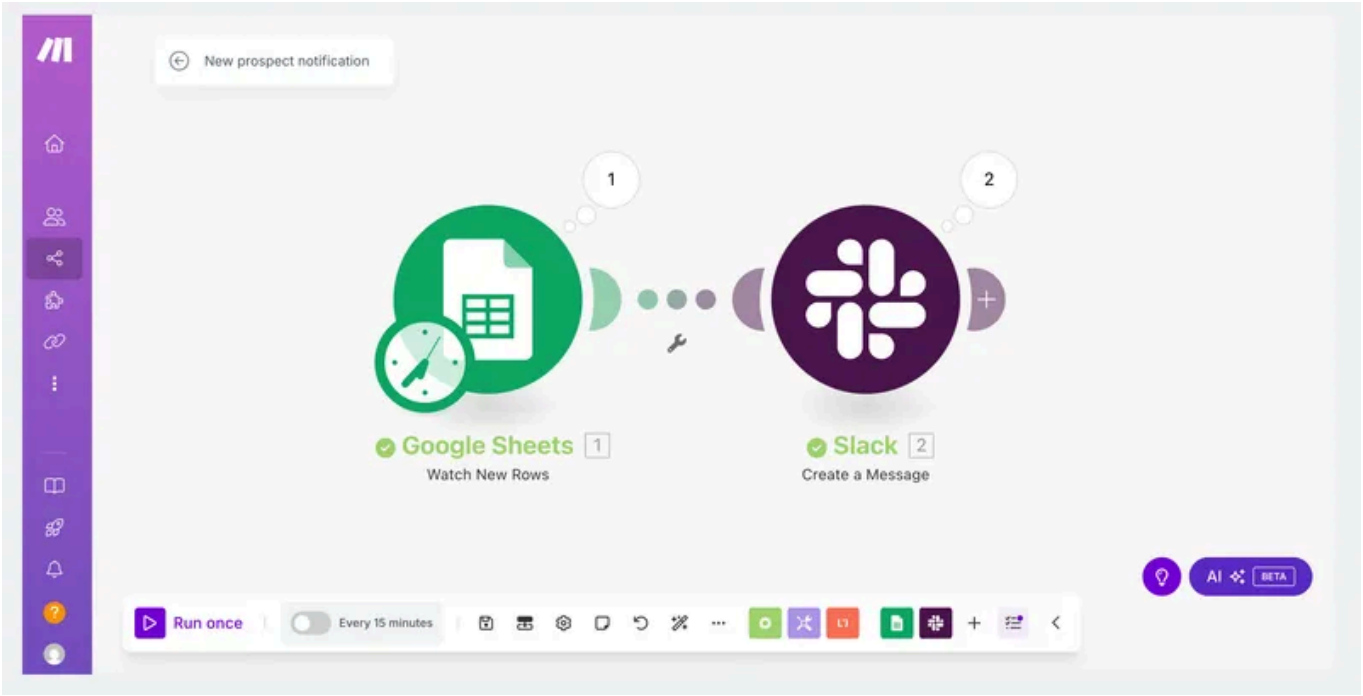
- Check if the applications are available in Make (Gmail, Slack, Google Sheets, and thousands more are supported).
- Verify you have access to these applications (such as Slack workspace admin rights or a Google Workspace account).
- Ensure you have the necessary permissions (access to specific Google Sheets or Slack channels).

Outline the specific actions needed

- What triggers the scenario? (a new email, form submission, or updates to a spreadsheet row)
- What information needs processing? (contact details, order information, or customer data)
- What should happen with this information? (create a new contact, send notifications, update databases)

For our sales notification scenario, here's the breakdown:

- **Applications**: Google Sheets, Slack
- **Trigger**: new row added in Google Sheets
- **Actions**: send Slack notification to the team



In the next sections, you'll prepare these apps to be used in your scenario.

1.2.2. Step 2. Get your apps ready

Before building your first scenario, you need to get your apps and accounts ready.

Required accounts

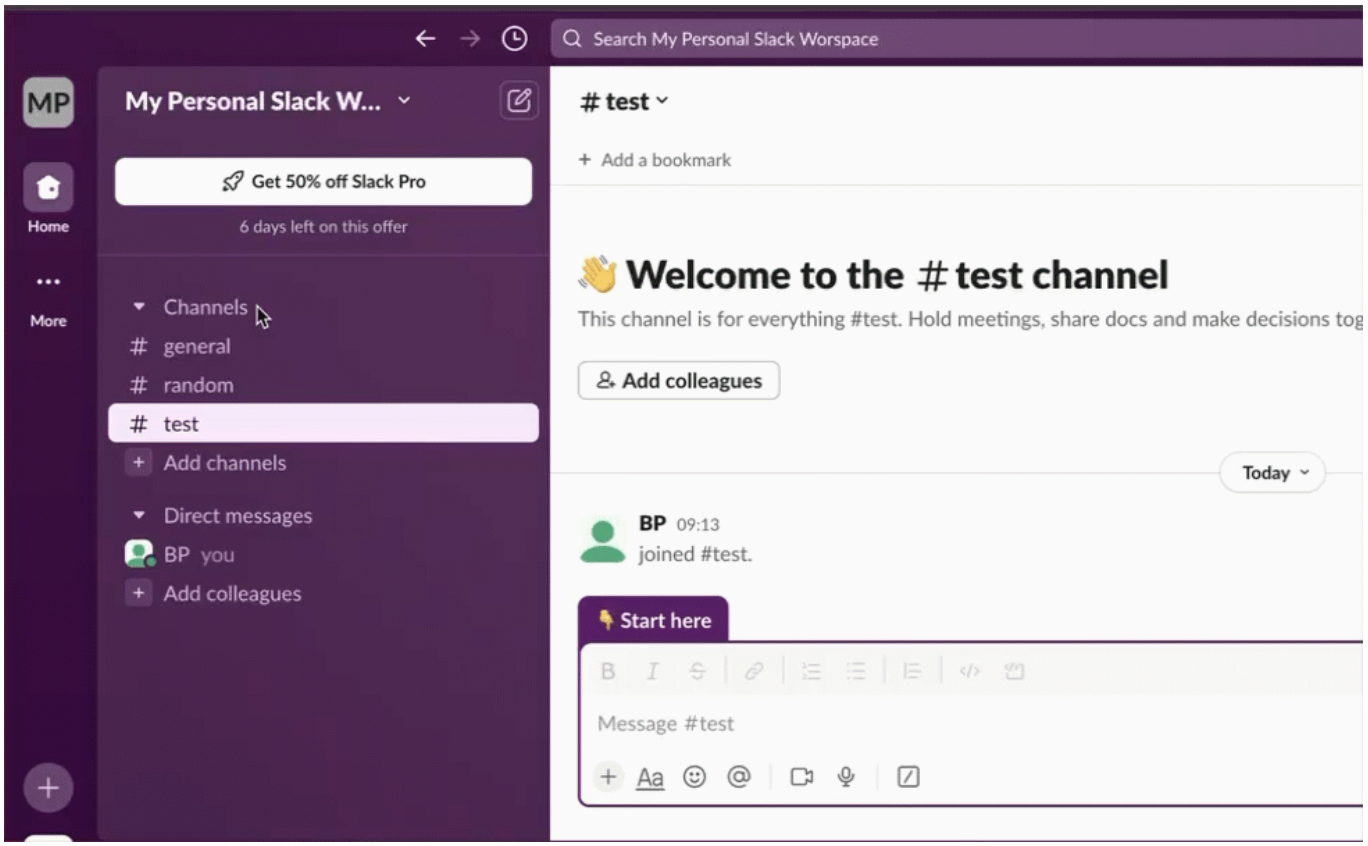
- A Make account (**Create one here**)
- A Slack account and workspace (**Sign up for Slack**)
- A Google account (**Create an account**)

With your accounts created, the next step is to set up Slack and Google Sheets that you will use in your scenario.

Set up Slack

To create a channel where your sales team will receive notifications:

1. Open your **Slack workspace**.
2. Click the dropdown arrow next to **Channels** in the sidebar.
3. Select **Create channel** from the dropdown menu.
4. Name it "sales-team".
5. Set visibility to "Public".
6. Add a description (optional).
7. Click **Create**.



This channel will be used to send prospect notifications to the sales team.

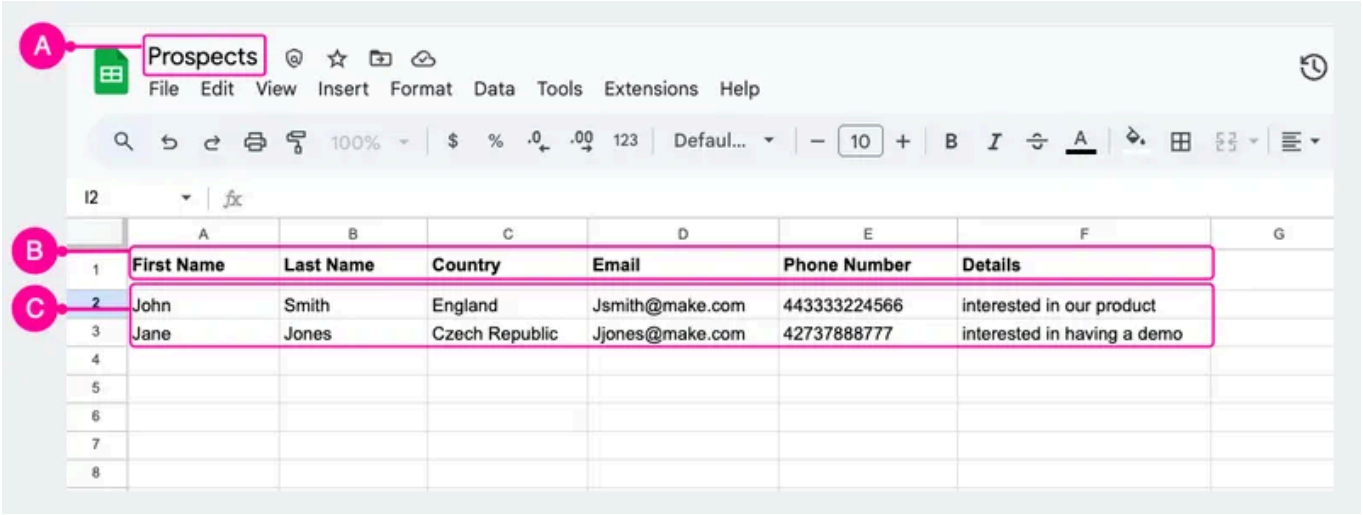
Set up Google Sheets

To create a spreadsheet to store prospect information:

1. Go to **Google Sheets.**
2. Start a new spreadsheet.
3. Rename it to "Prospects".
4. Add these columns:
    - First Name
    - Last Name
    - Country
    - Email

- Phone Number
- Details

5 Add one row of sample data.



Google Sheet with proper column setup and sample data showing:

- The name of your file (**A**)
- Columns you added (**B**)
- Sample data (**C**)

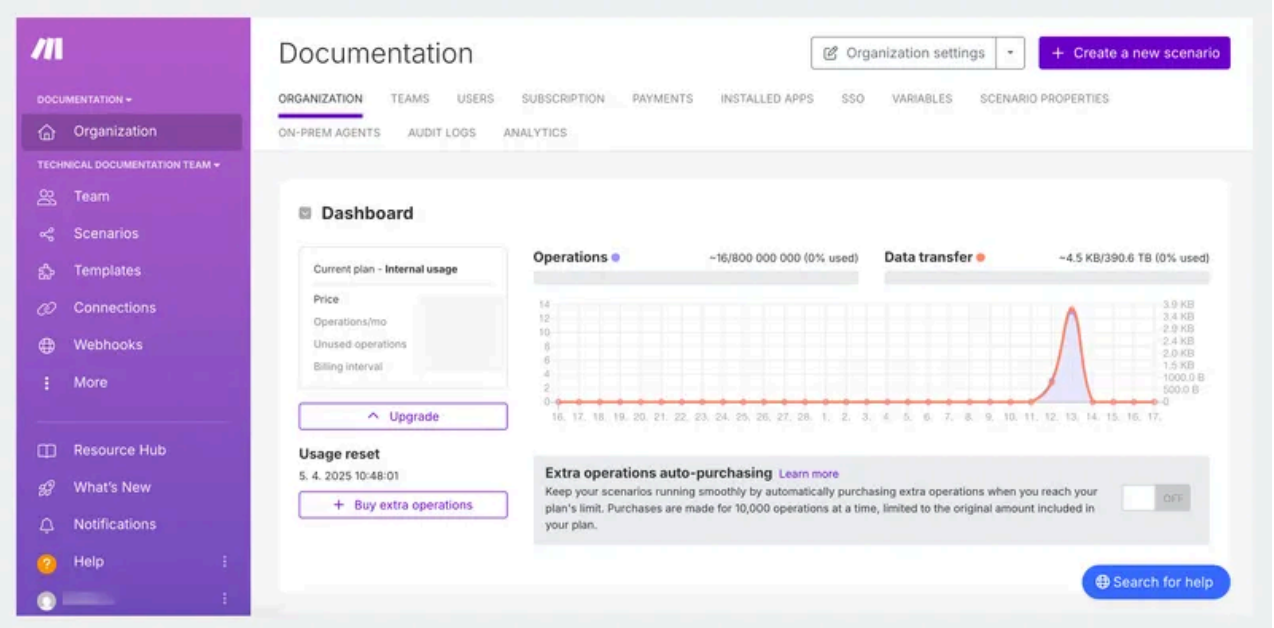
> Keep this spreadsheet open in another browser tab for easy access while you build your scenario.

1.2.3. Step 3. Add your first app

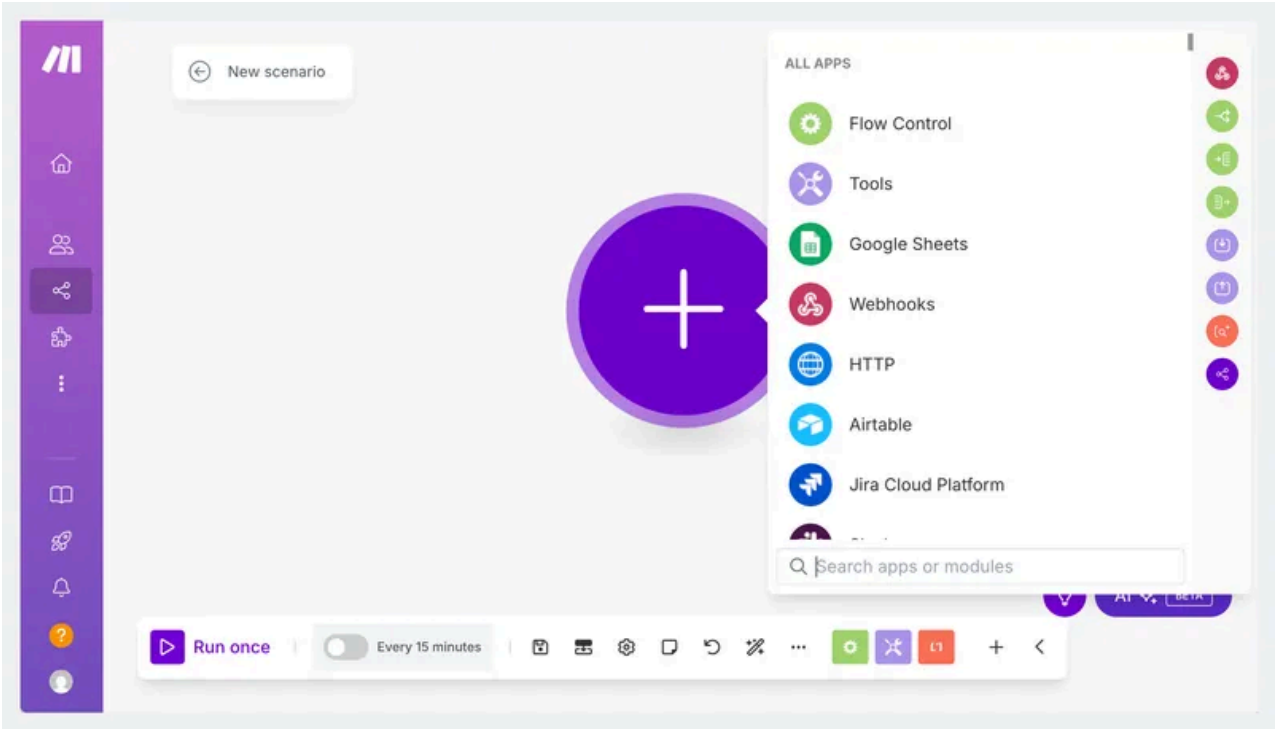
At this step, you'll add the first application and module that will trigger your scenario.

> A trigger module is what kicks off your scenario—for example, when somebody fills in a form, when a new email arrives in your inbox, or when a row is added to a spreadsheet.
>
> In this guide, we're using "Watch New Rows" for Google Sheets as our trigger.

1. Sign in to Make. You'll see the main dashboard where you can access all platform features.



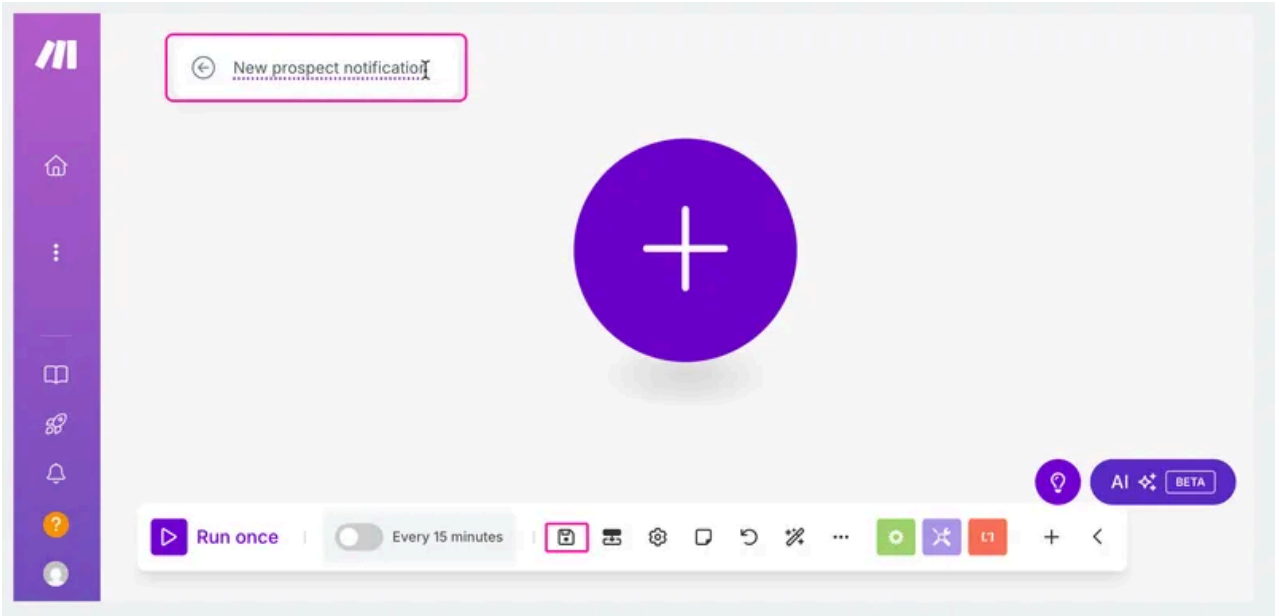
2. Click **+ Create a new scenario** in the top-right corner to open the Scenario Builder.



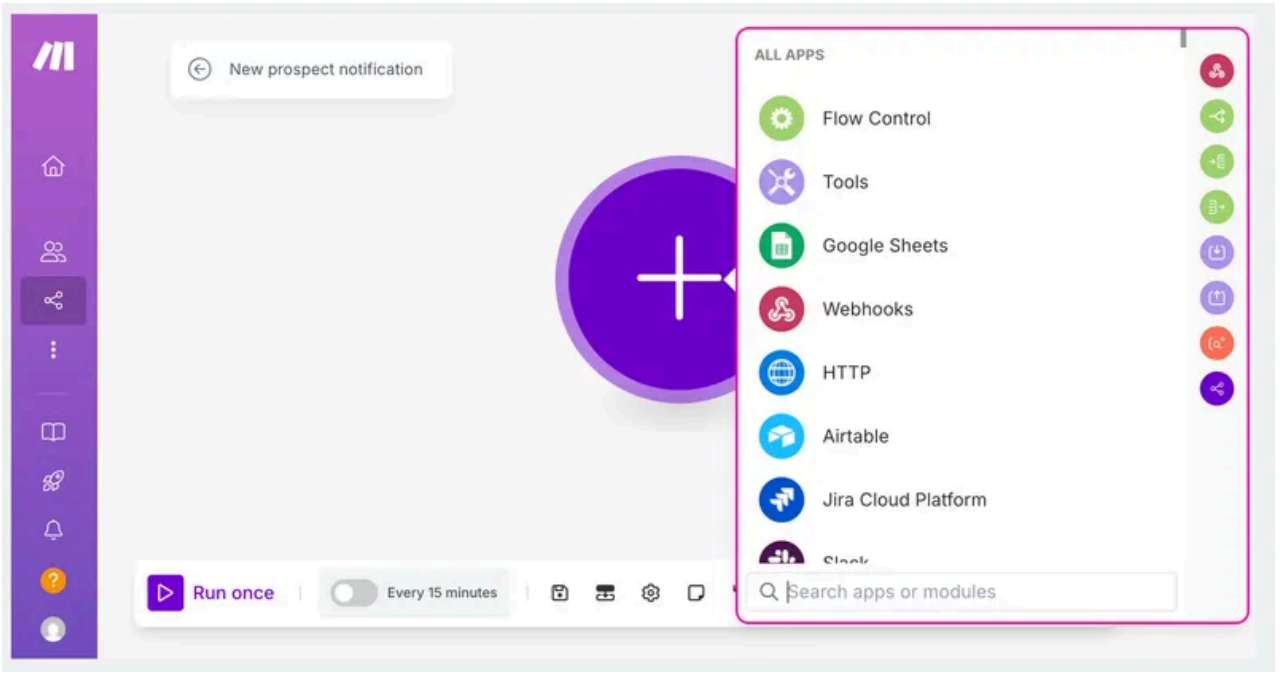
> You can also find the **+ Create a new scenario** button on the Organization, Team, or Scenarios pages found in the left menu.

3 In the top-left corner, click **New scenario** to update your scenario's name. Enter "New prospect notification" (or any name you prefer).

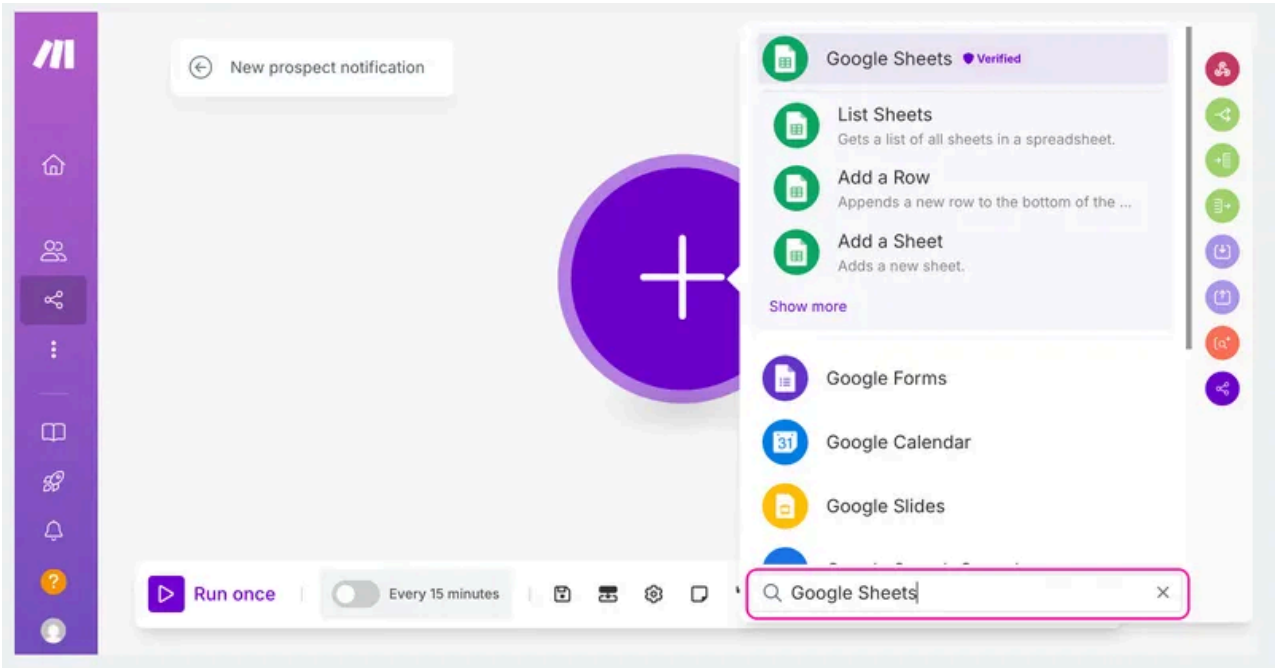
4 Click the **Save** icon on the Scenario Builder toolbar to save your scenario.



5 Click the giant plus in the center of your canvas. This will open a window with the list of apps that you can add to your scenario.

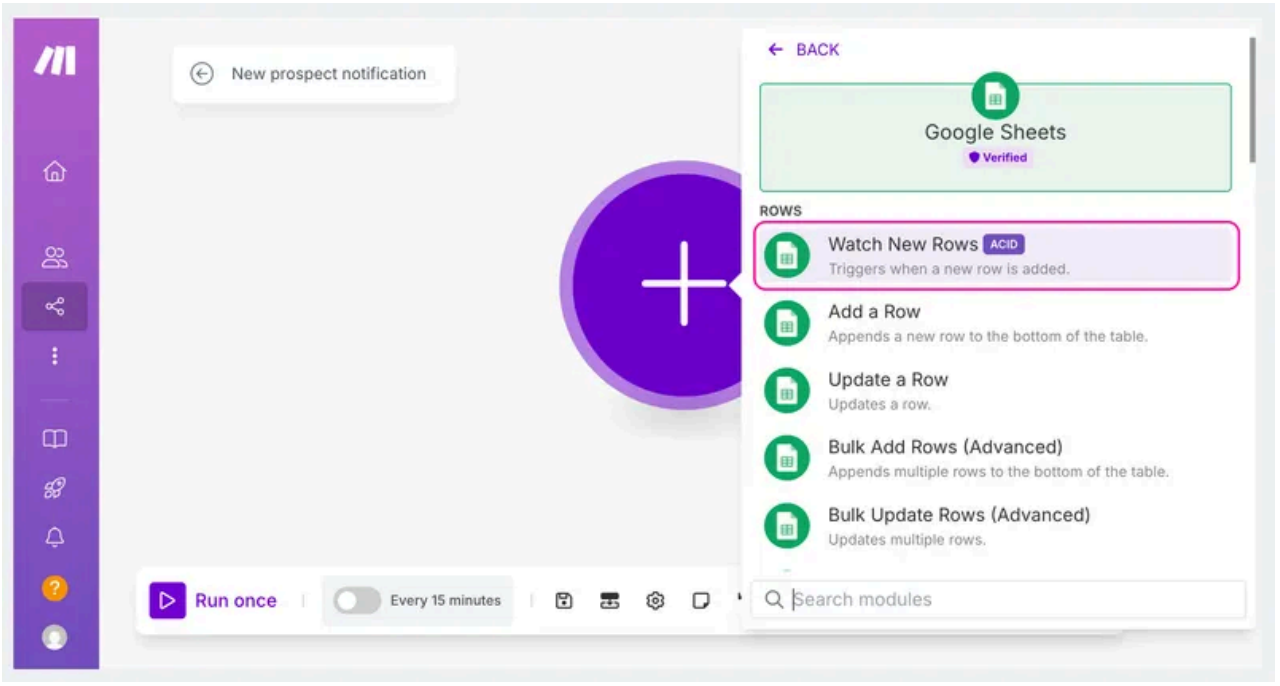


6 Search for "Google Sheets" in the search bar or browse the available apps.

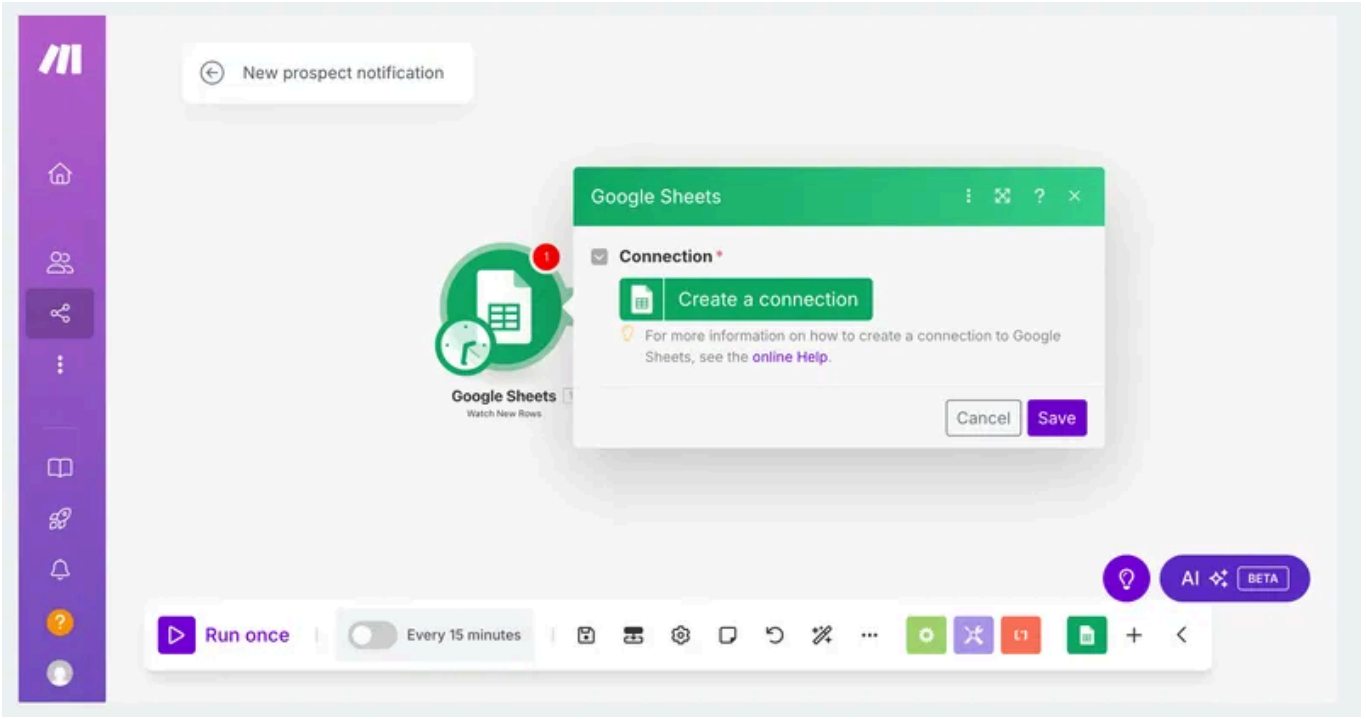


> If you select the wrong app, click **Back** to return to the list.

7 Select "Google Sheets" and then choose "Watch New Rows" from the available modules.



Now you've added your trigger module that will start your scenario whenever new prospects are added to your spreadsheet. Before you can configure it, you need to create a connection to Google Sheets.

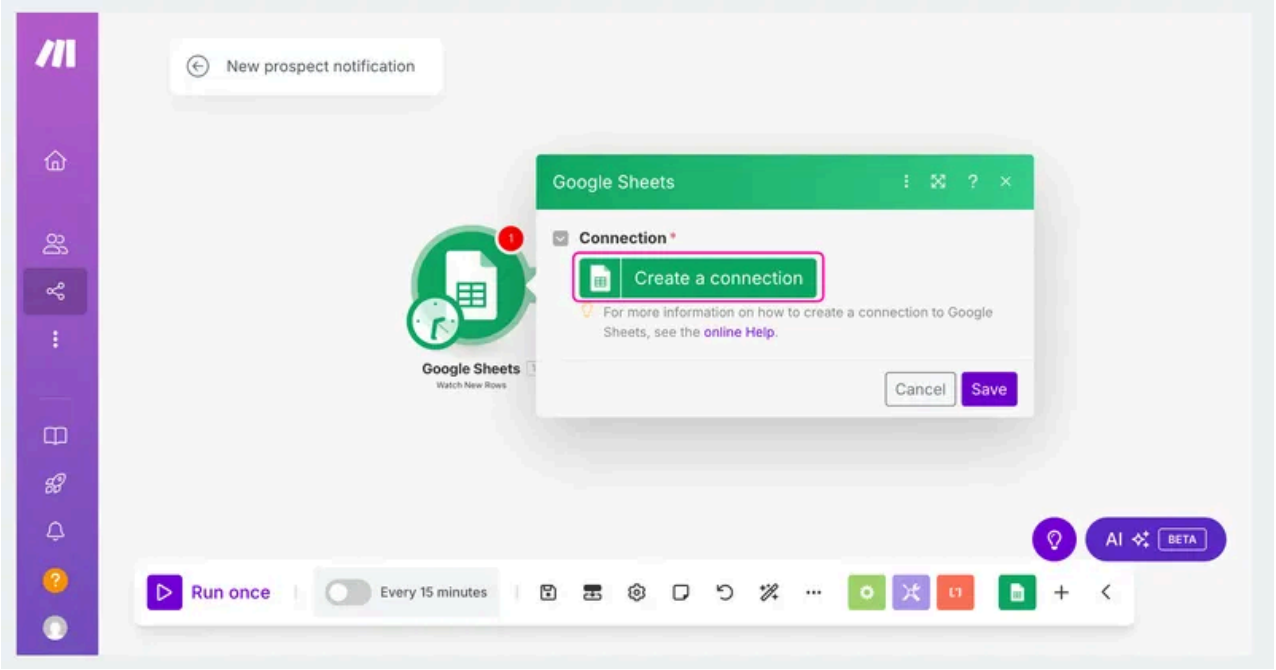


1.2.4. Step 4. Create a connection

To use most apps in Make, you first need to create a connection. Connections securely link Make to your external applications, allowing data to flow between them.

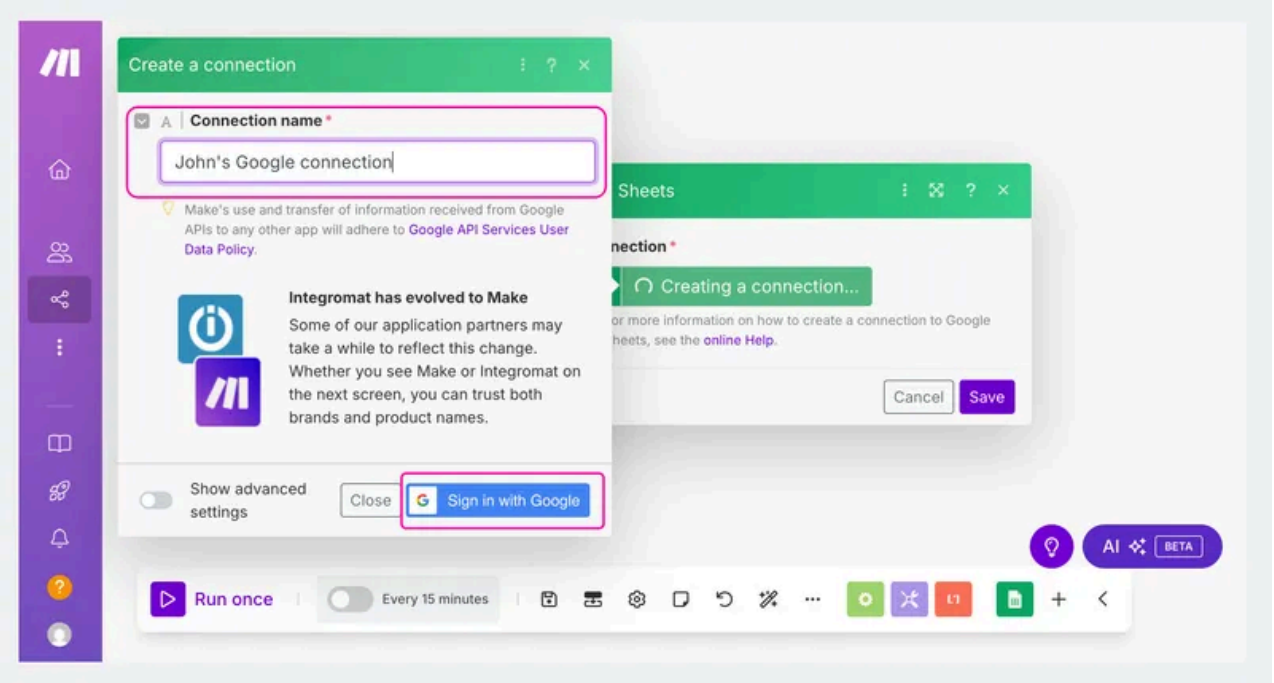
To create a connection to Google Sheets:

1. Click **Create a connection.**

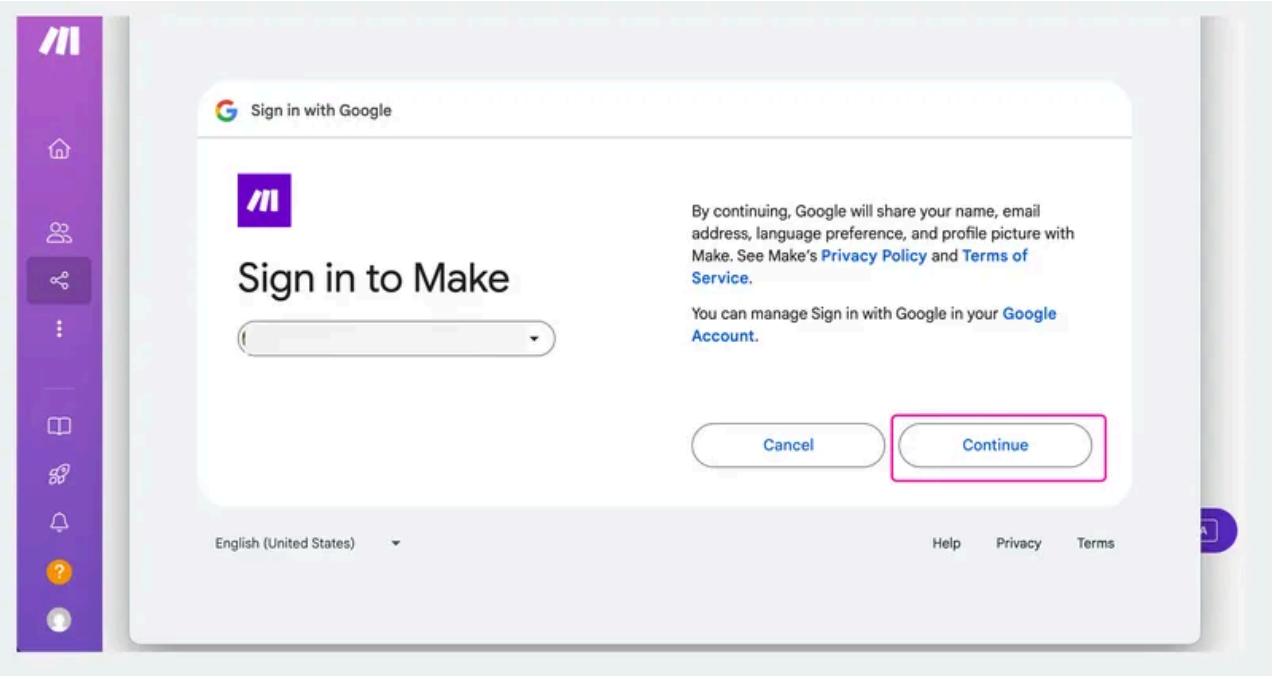


2. Name your connection and click **Sign in with Google.**

> Use your name or the Google account name you're connecting with to be able to recognize it later.

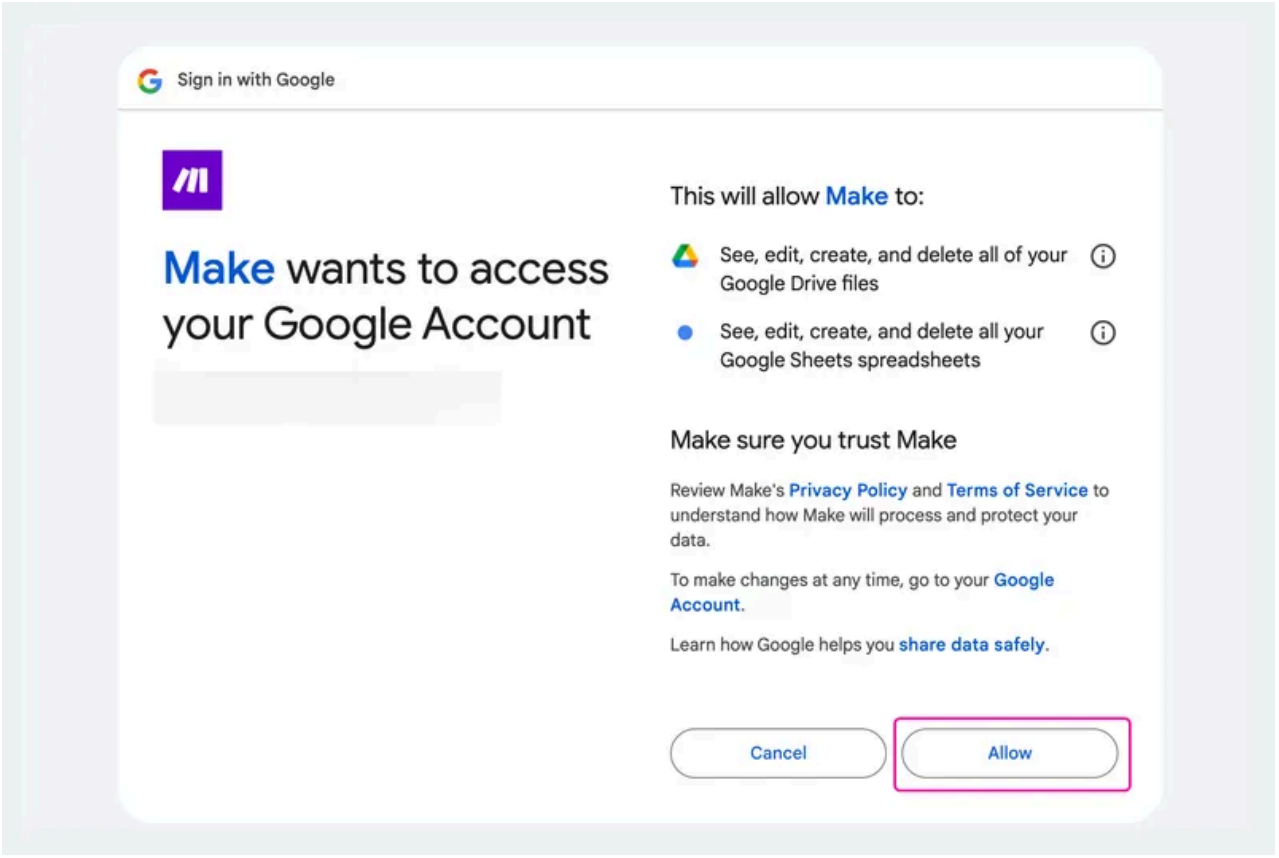


3 A Google sign-in window will open. Click **Continue**.

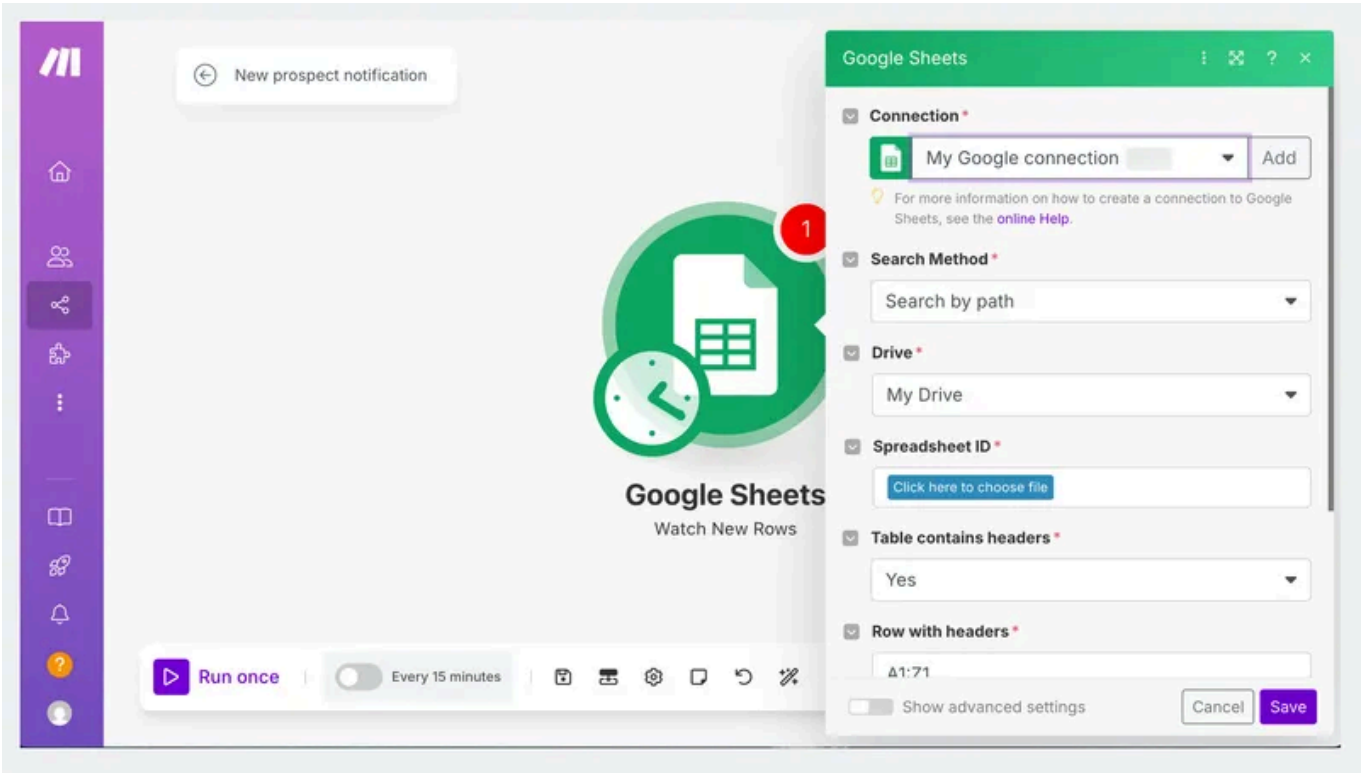


4 Follow the steps to authorize Make to access your Google account.

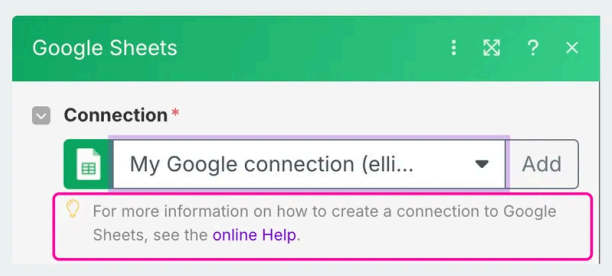
> This authorization step is required so Make can access your spreadsheet data.



Once you've completed the authorization, it means your connection is created and you're now ready to configure your Google Sheets trigger module so it can watch for changes in the spreadsheet.



Connections are an essential part of Make scenarios. They allow your scenarios to get information from and send it to third-party applications. Different apps require different types of authentication — from simple username and password combinations to API keys or OAuth. When you create a connection to a new app, you can click the **Online Help** link to learn more about how to connect the selected app.



Once created, a connection is saved for future use. You can reuse the same connection in other scenarios that work with Google Sheets.

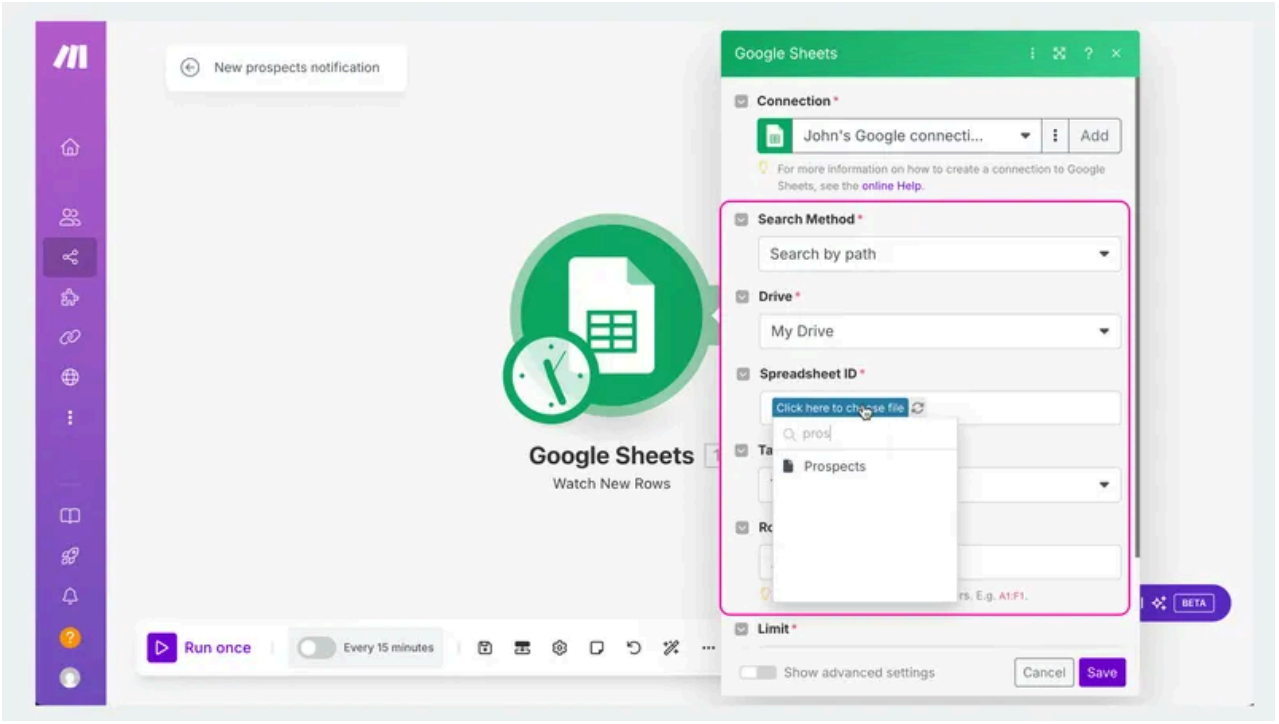
1.2.5. Step 5. Set up the trigger

At this step, you'll set up your Google Sheets trigger to watch for new rows in your spreadsheet. You'll specify which spreadsheet to monitor and how many rows to process at once.

To set up your Google Sheets trigger:

1. Select your **Drive** from the dropdown menu. You can leave the **Search Method** field at its default setting ("Search by path").

2. In the **Spreadsheet ID** field, click "Click here to choose file". Search for and select the "Prospects" spreadsheet you created during Step 2.

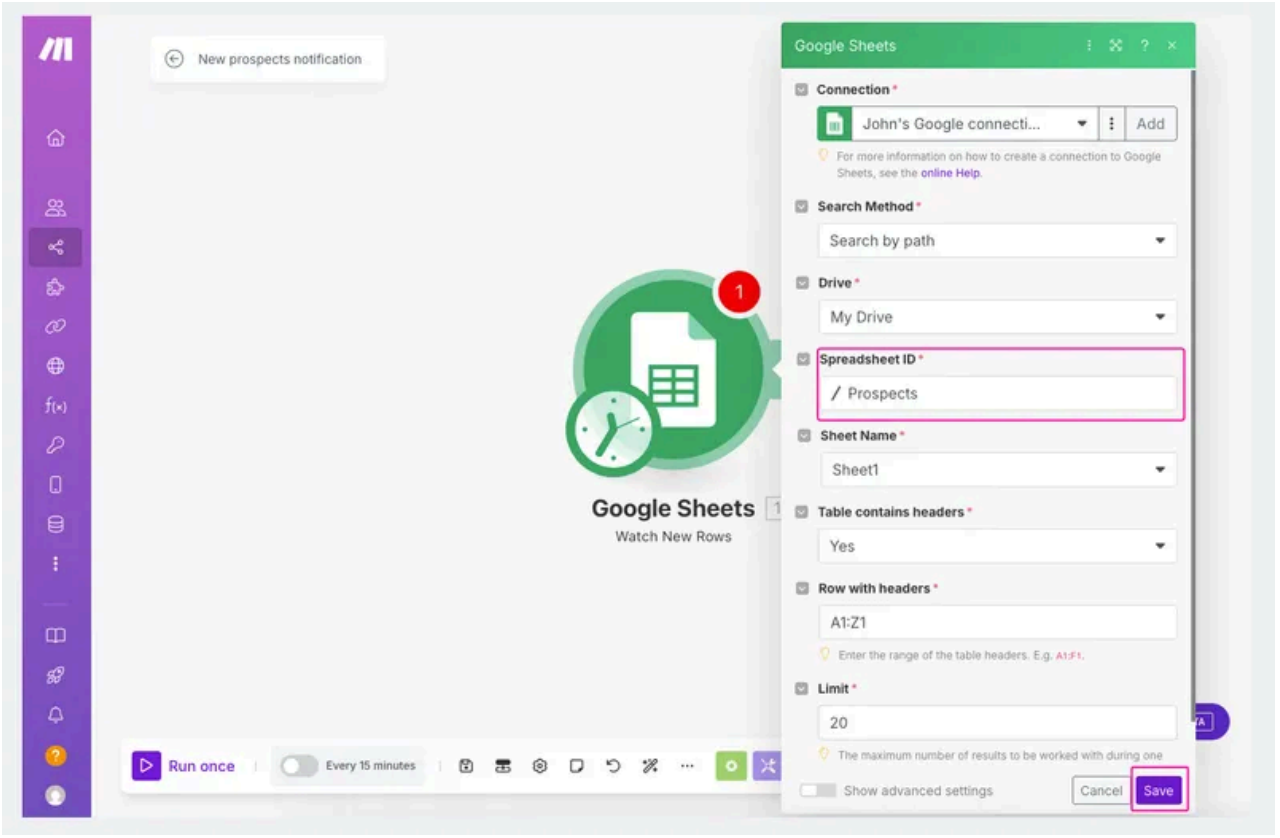


3. Select "Sheet1" in the **Sheet Name** field. This specifies which sheet contains the rows you want to monitor.

4. For **Table contains headers**, select "Yes". This tells Make that the first row contains column names, which you'll use for mapping data later.

5 For **Row with headers**, leave the default "A1:Z1" value. This indicates where your column headers are located.

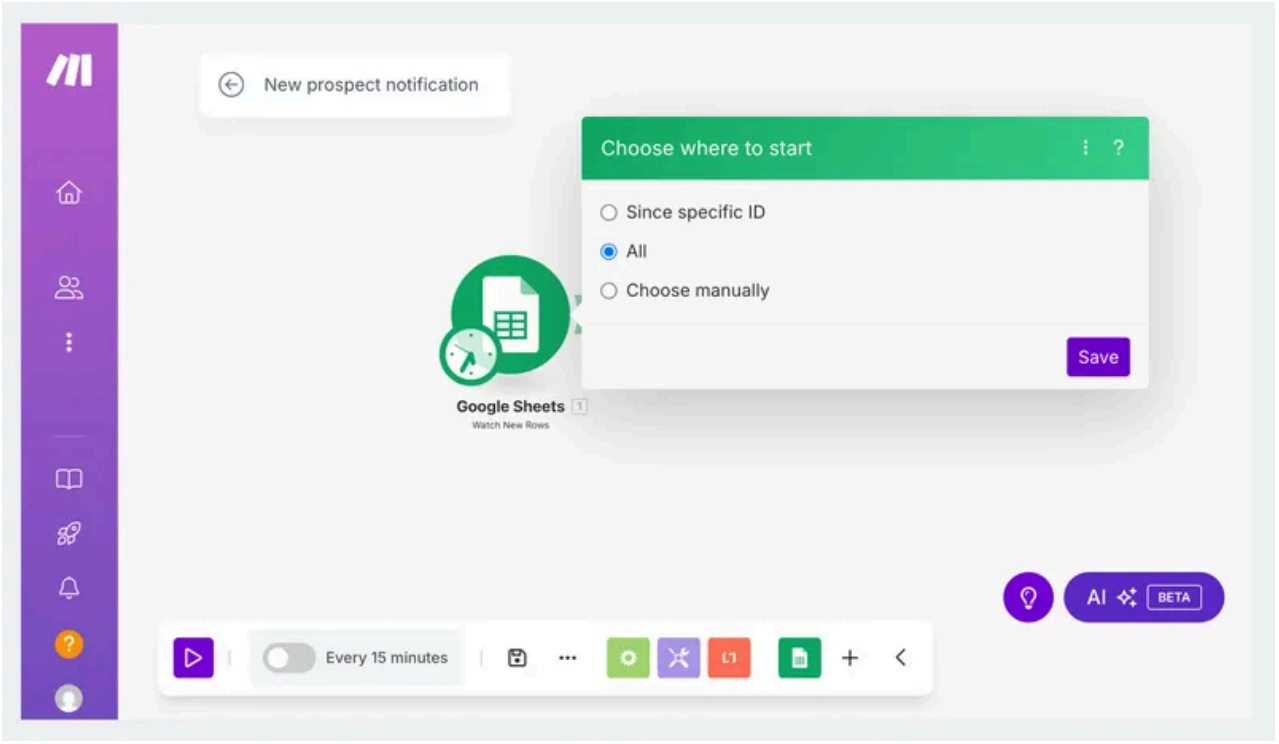
6 Set the **Limit** to 20. This tells the module to process up to 20 new rows in a single run. You can adjust this based on your needs.



7 Click **Save**. This saves your module settings, but not the entire scenario.

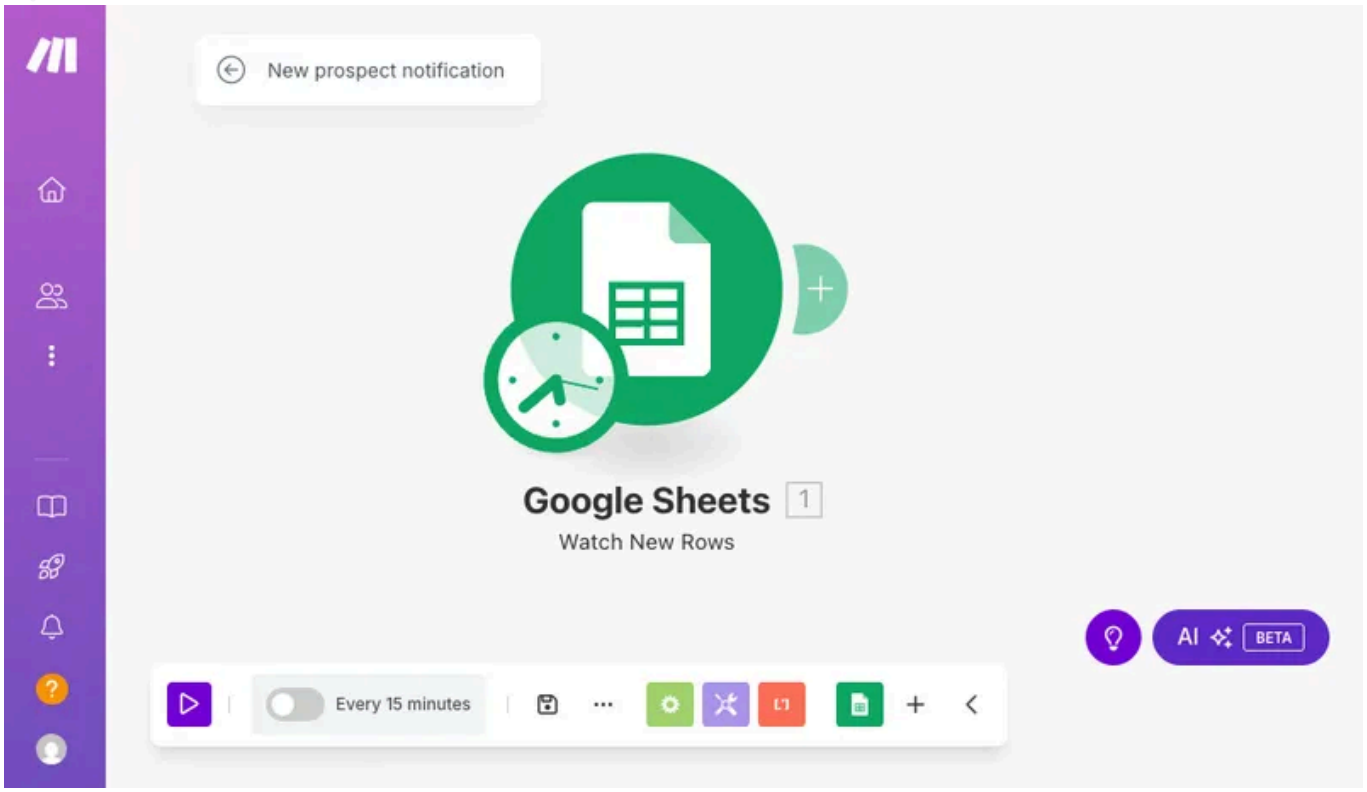
> Remember to click the **Save** icon in the Scenario Builder toolbar to save your entire scenario. It's good practice to save your scenario regularly as you build it.

8 In the **Choose where to start** window that appears, select "All".



> If you accidentally close that window, right-click the Google Sheets module and select **Choose where to start** from the menu.

The "Choose where to start" option determines which rows the module will process. By selecting "All," the module will first process the existing rows in the sheet, and then any new rows added afterwards. Also, the module will track which rows it has already processed and only fetch the ones it hasn't processed yet.



Your Google Sheets trigger is now configured to detect new rows in your Prospects spreadsheet. Next, you'll test the module to ensure it works properly.

Understanding trigger modules

Trigger modules like the one you've just configured can only be used at the beginning of scenarios. They process data when specific events occur in connected applications. There are two main types:

- **Instant triggers** start your scenario immediately when an event occurs in the connected application (marked with an "instant" tag in Make). These are ideal when you need real-time responses.
- **Polling triggers** (like your Google Sheets module) check for new data at regular intervals when your scenario runs. The module remembers what it has already processed and only retrieves new items each time.

The Google Sheets "Watch New Rows" module you're using is a polling trigger, which will check for new rows each time your scenario runs.

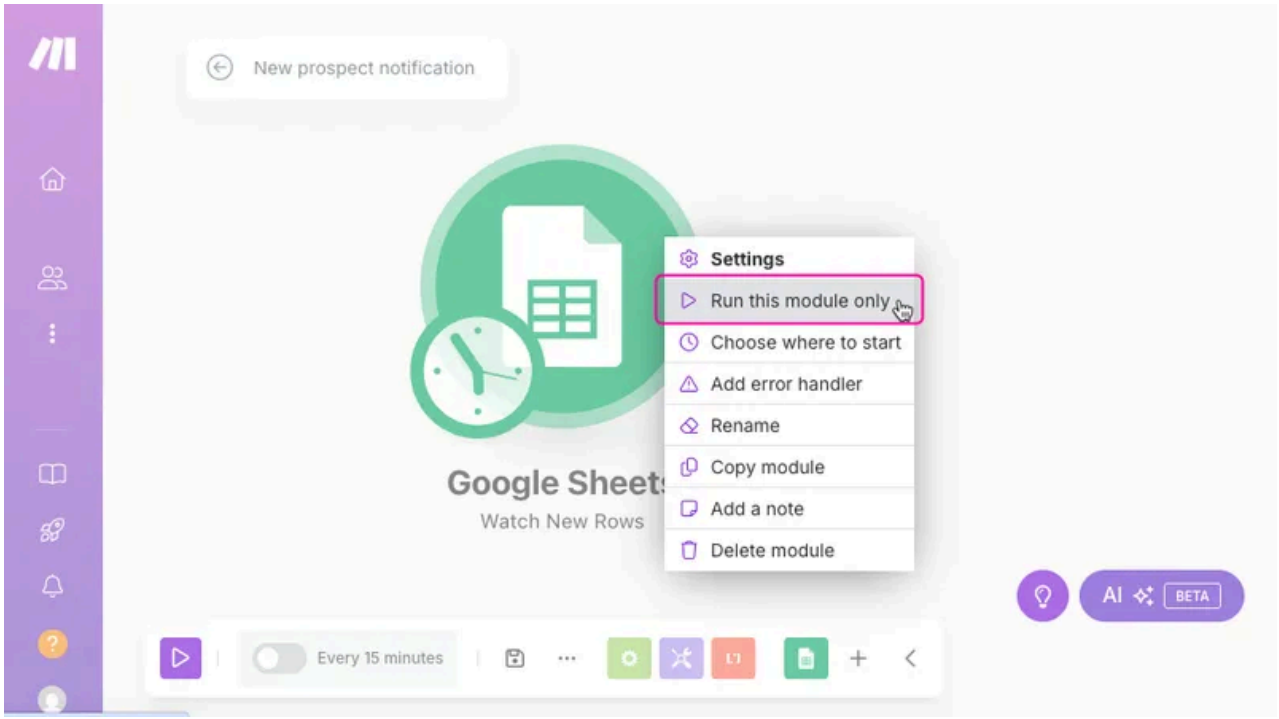
1.2.6. Step 6. Test the module

Testing makes sure your module is working correctly before you continue building your scenario.

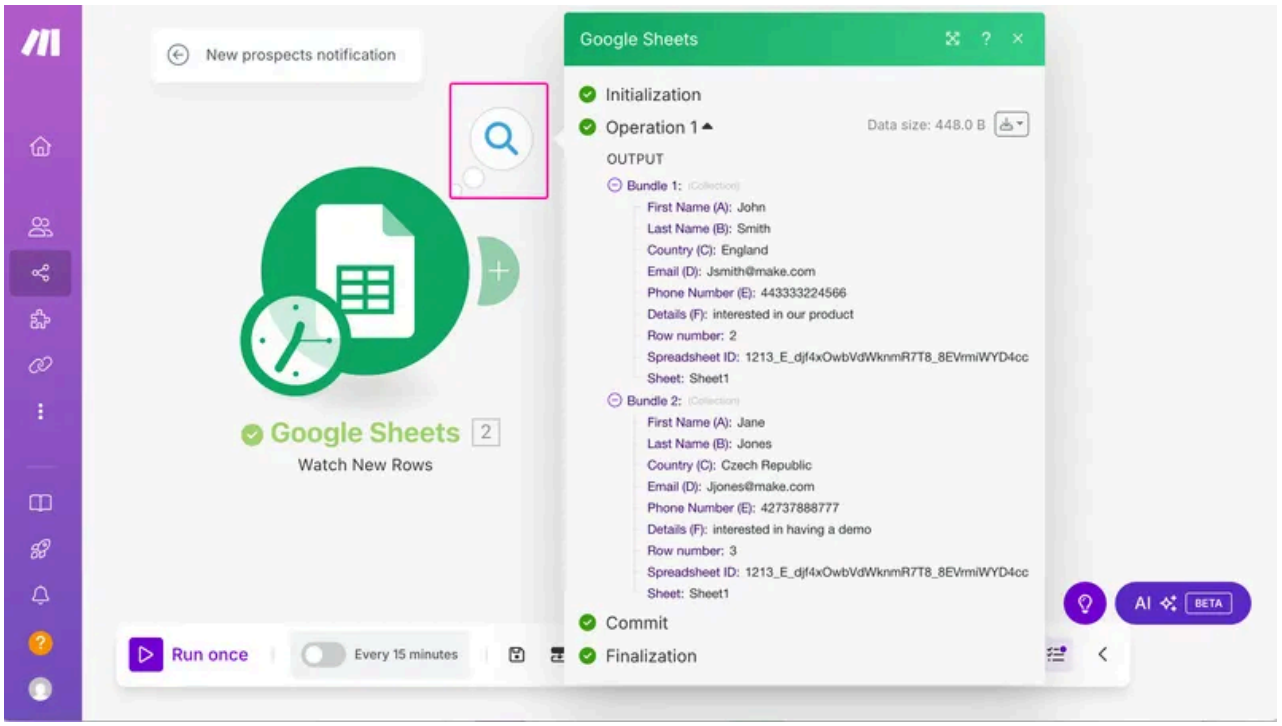
To test the Google Sheets "Watch New Rows" module:

1. Right-click on the Google Sheets module.

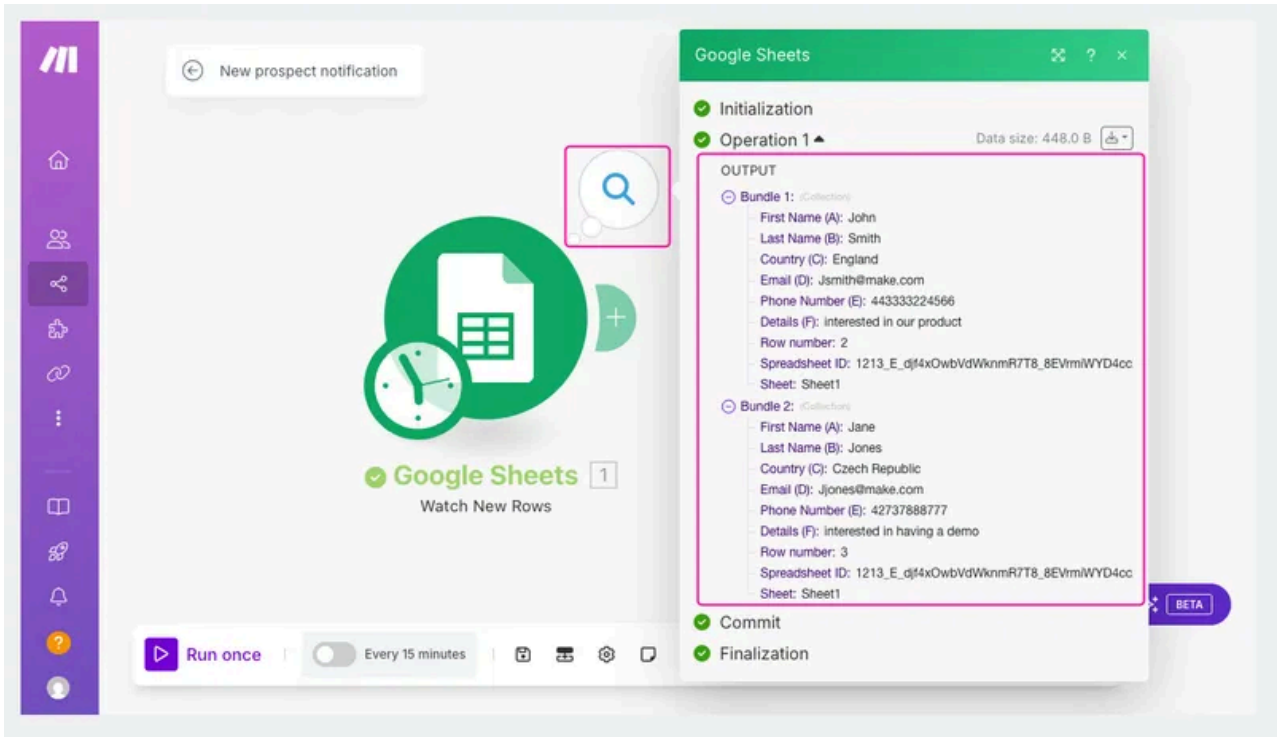
2. Select **Run this module only.**



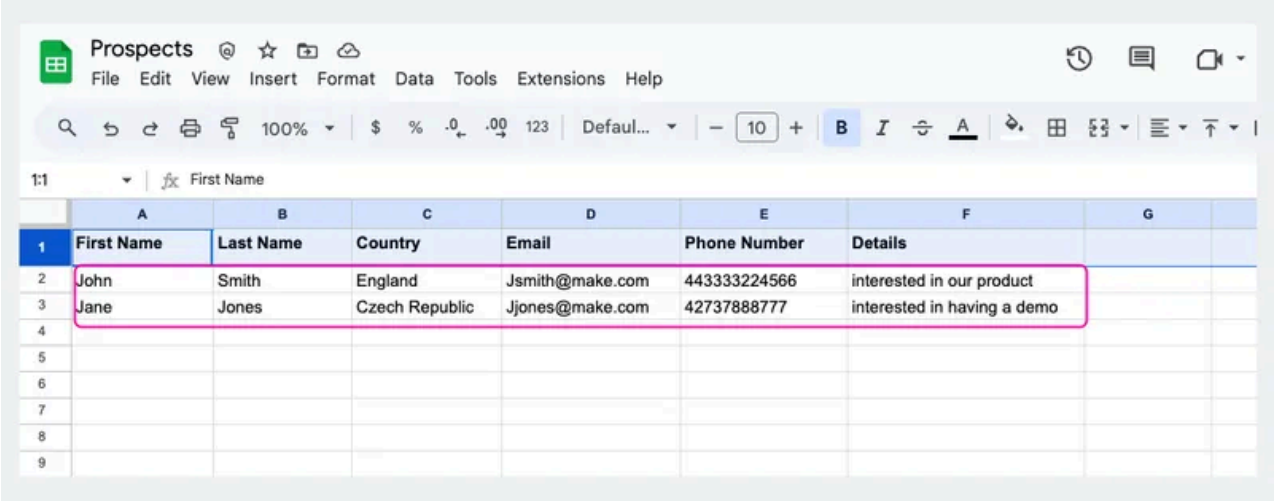
When the run completes, an operation bubble will appear above the module.



3 Click on this operation bubble to view the results. Since this module reads rows from your Prospects spreadsheet according to the limit you set in **Step 5**, you'll see all your spreadsheet data when you click on the operation bubble.



4 Compare the data in the operation output with your original spreadsheet to verify that all information appears correctly.



When creating a scenario, regularly test each step to identify potential issues early.

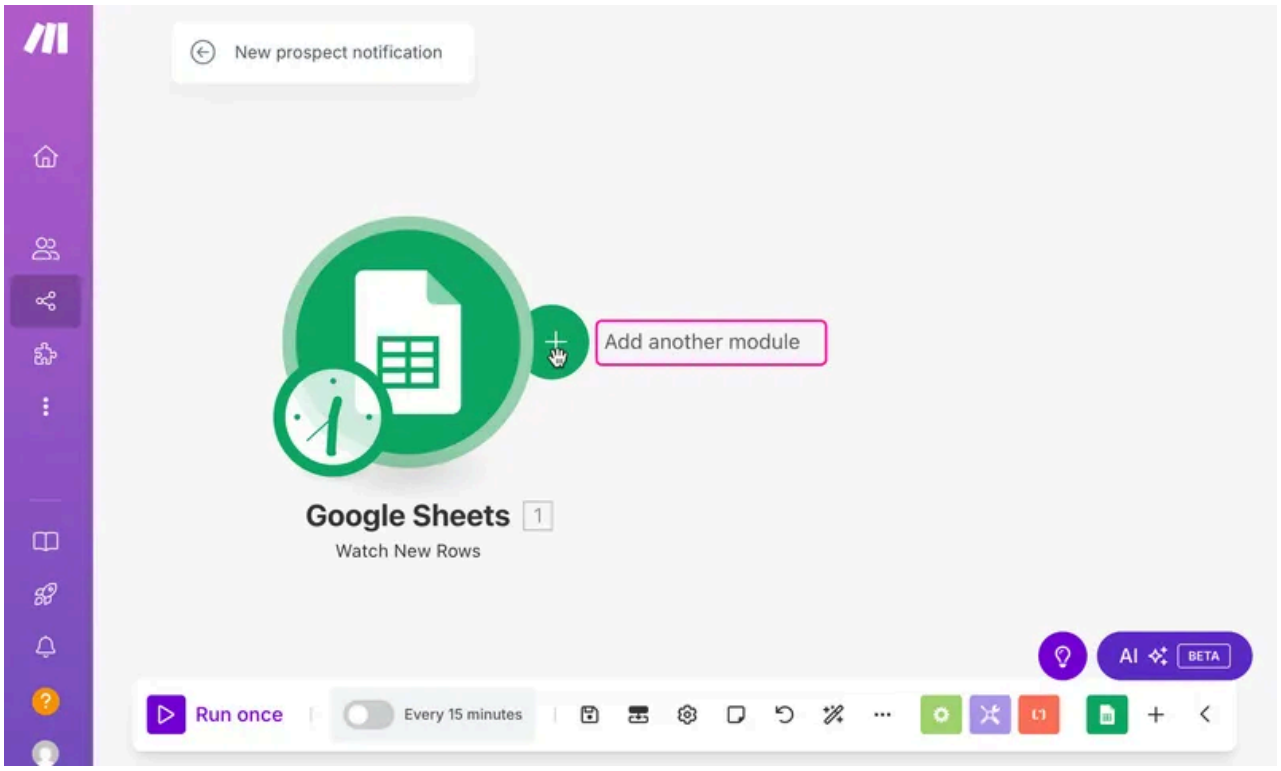
Every test run for this specific module consumes one **operation** from your account.

1.2.7. Step 7. Add another module

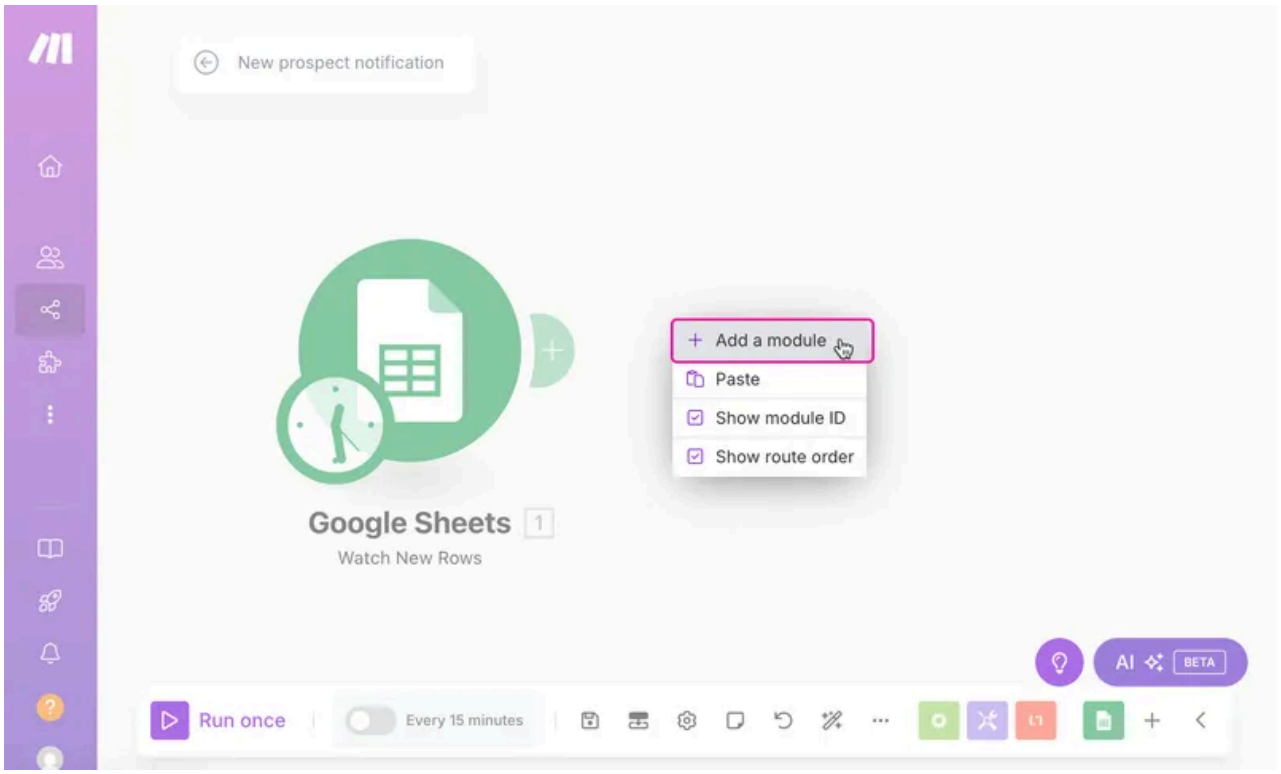
Now that your Google Sheets trigger is working correctly, you need to add a second module to send notifications and connect it to your Slack workspace.

To add another module to your scenario:

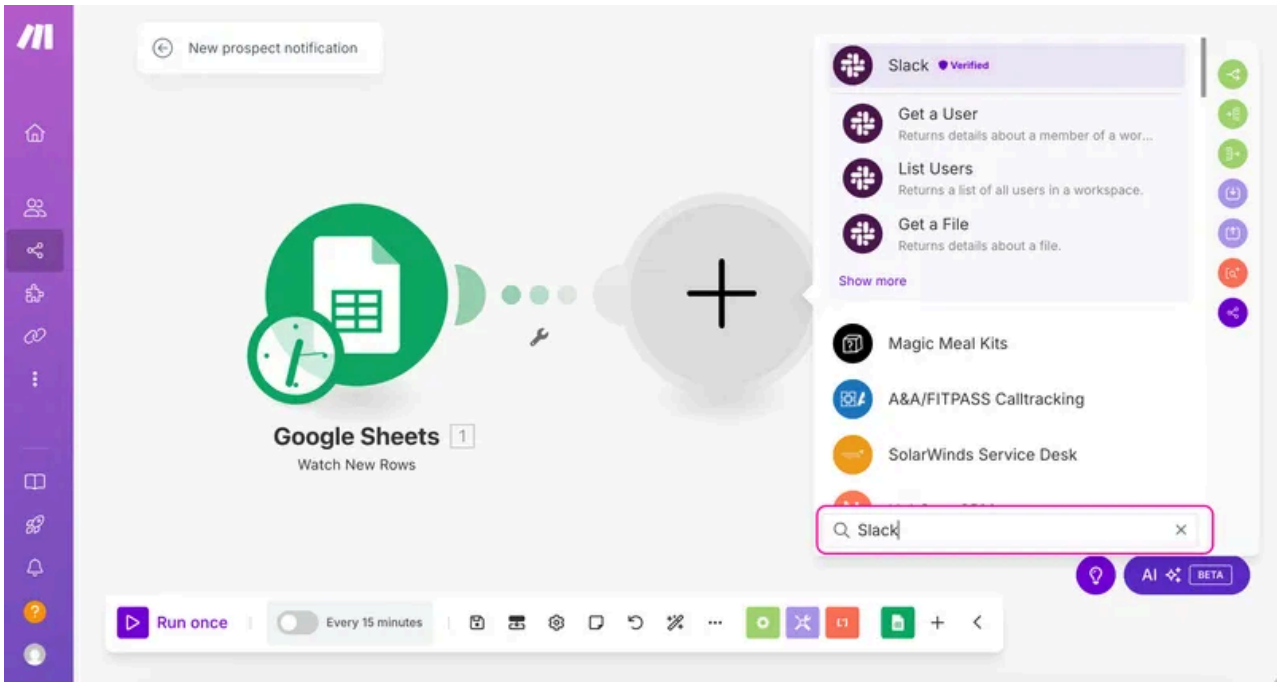
1. Click the **+** icon on the right side of the Google Sheets module.



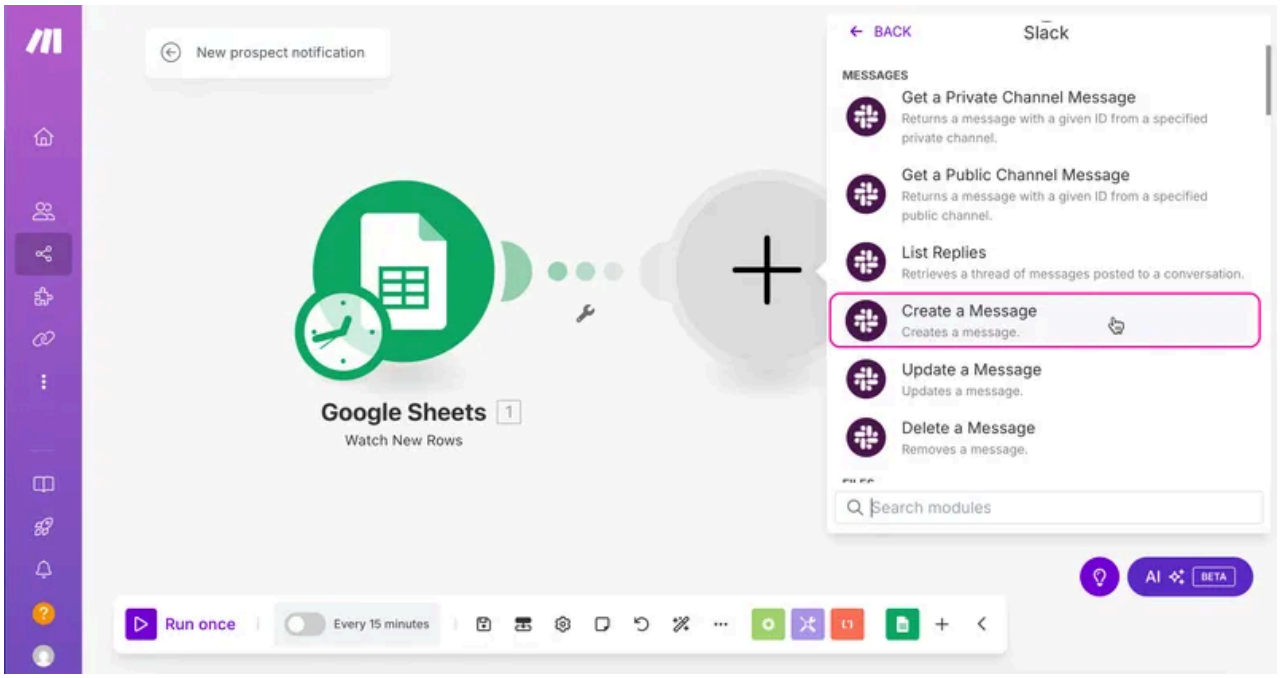
> Alternatively, to add a new module, you can double click anywhere in your canvas or right click anywhere and select **Add a module** from the menu.



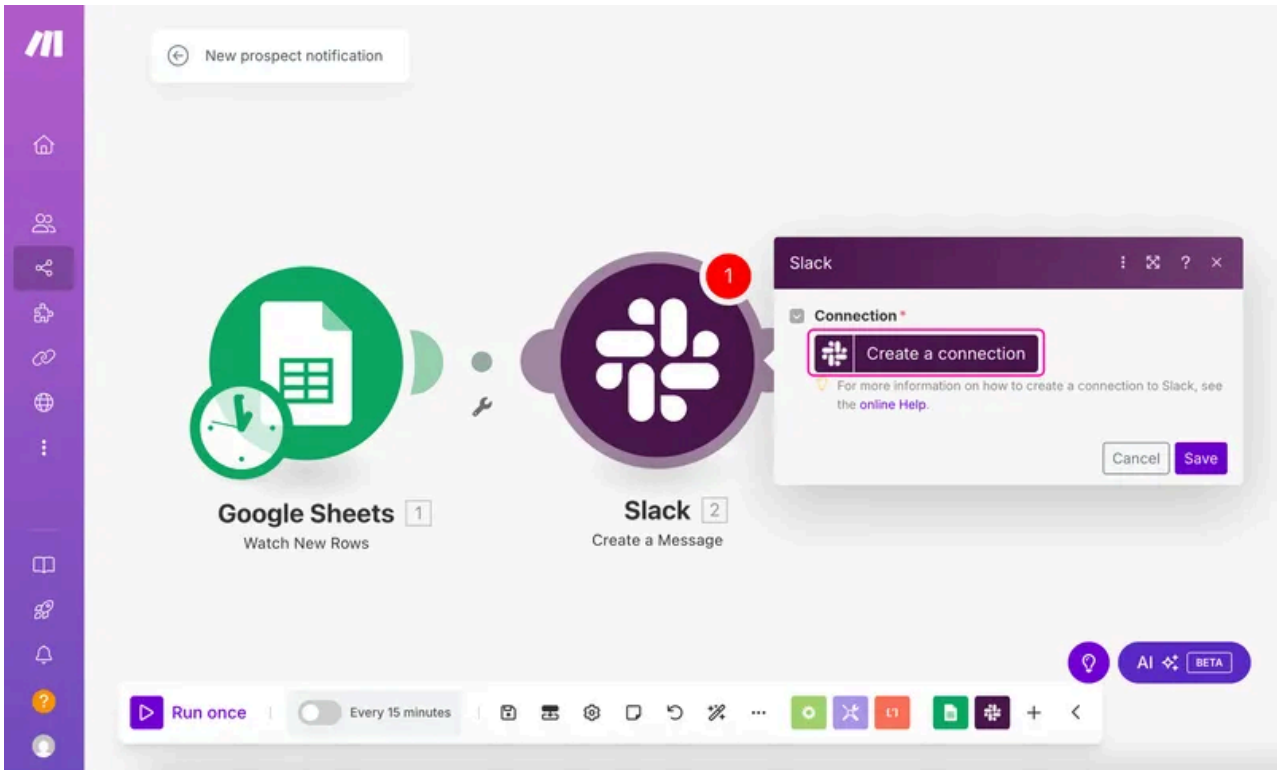
2 Search for "Slack" in the search bar or browse the available apps.



3 Select "Slack" and then choose "Create a Message" from the available modules.

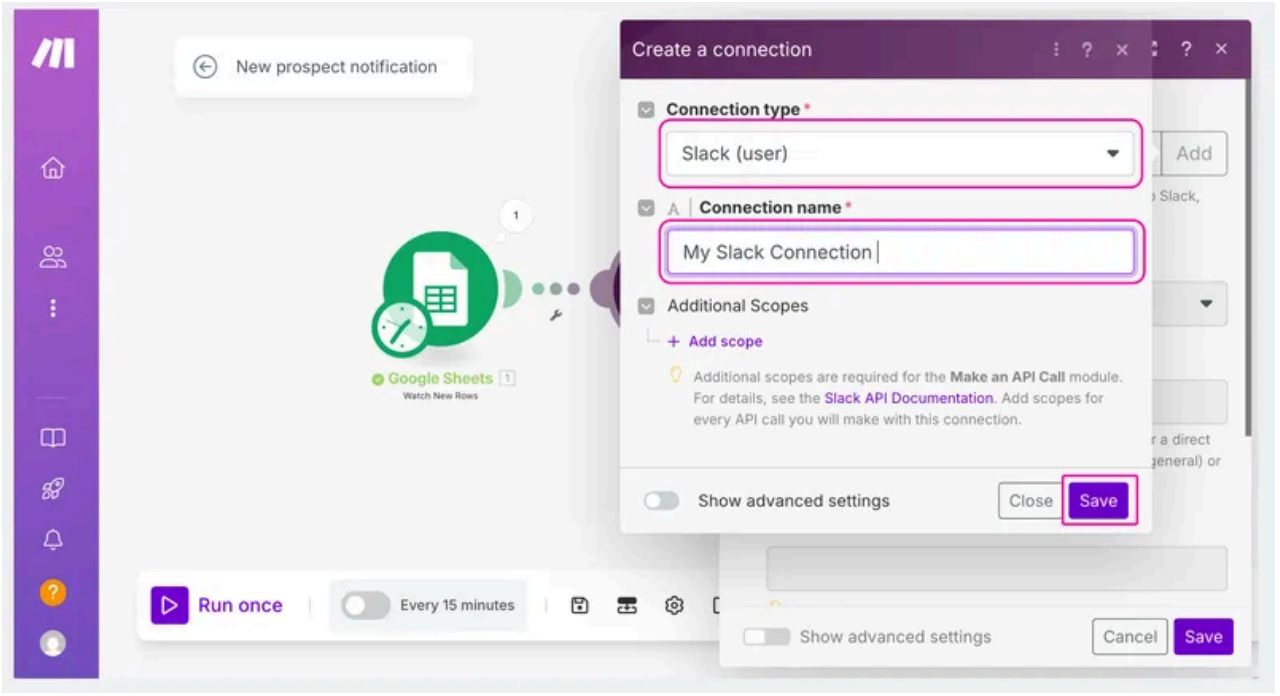


4 After you add the module to your scenario, the connection settings will appear. Click **Create a connection.**



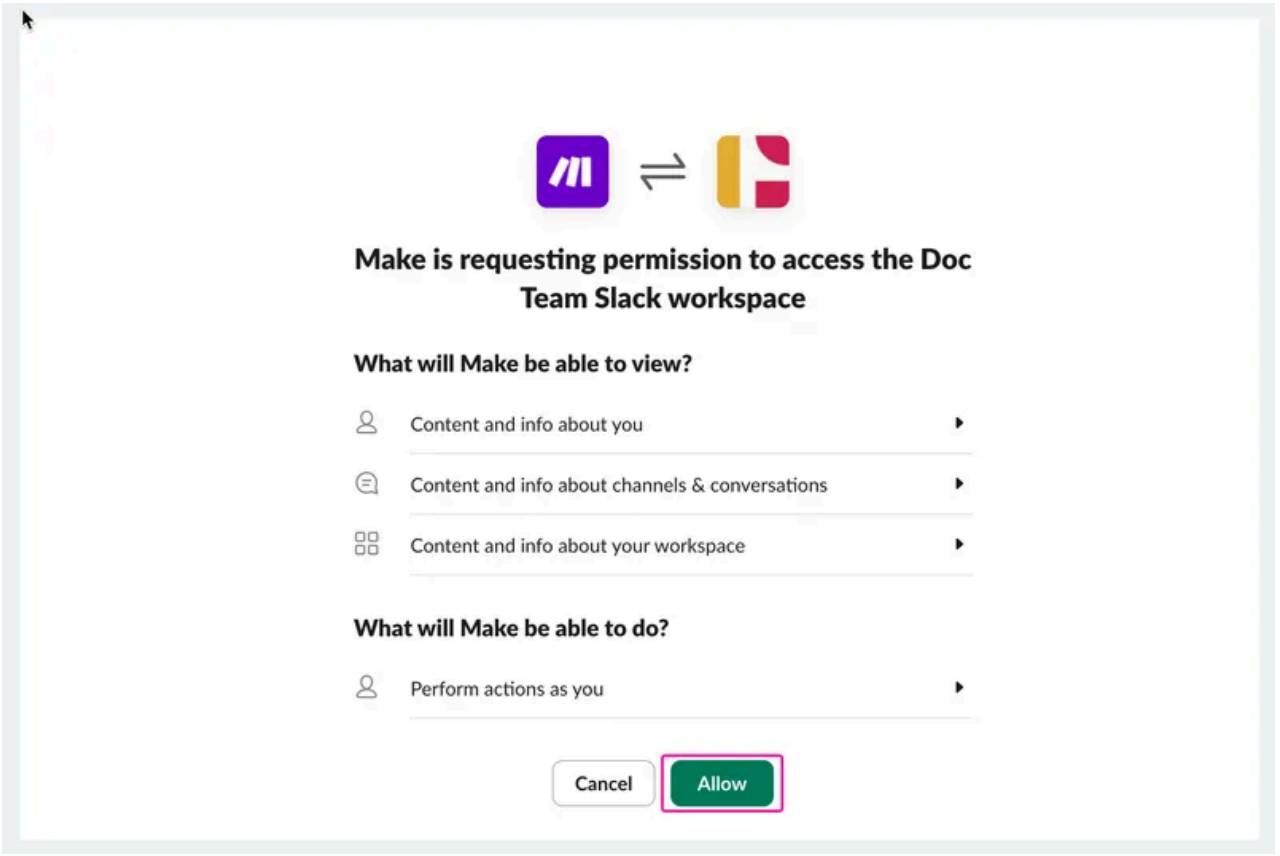
5 For **Connection type**, select "Slack (user)".

6 Name your connection, for example: "My Slack Connection", and click **Save**.



7 A Slack sign-in window will open. Click **Allow** to authorize Make to access your Slack account.

8



> If you have multiple Slack workspaces, make sure to select the one where you created the "sales-team" channel.

With your Slack module added to the scenario, you're now ready to configure it to send notifications using data from your Google Sheets trigger. In the next step, you'll map data between these modules.

1.2.8. Step 8. Map data

Mapping is how you send data between modules. It means taking information (output) from one module and telling Make how you want to use it in the next module (input).

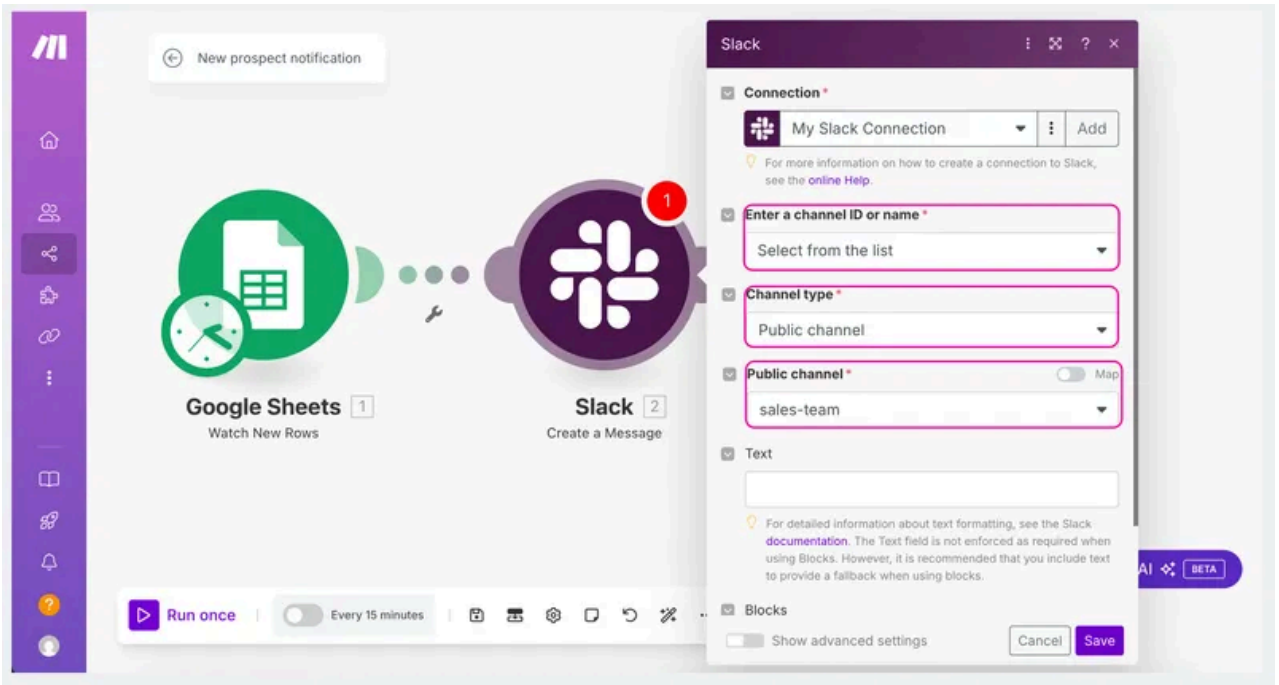
In this scenario, you're taking prospect information from Google Sheets (output) and using it in specific spots in your Slack message (input).

To configure your Slack message with data from Google Sheets:

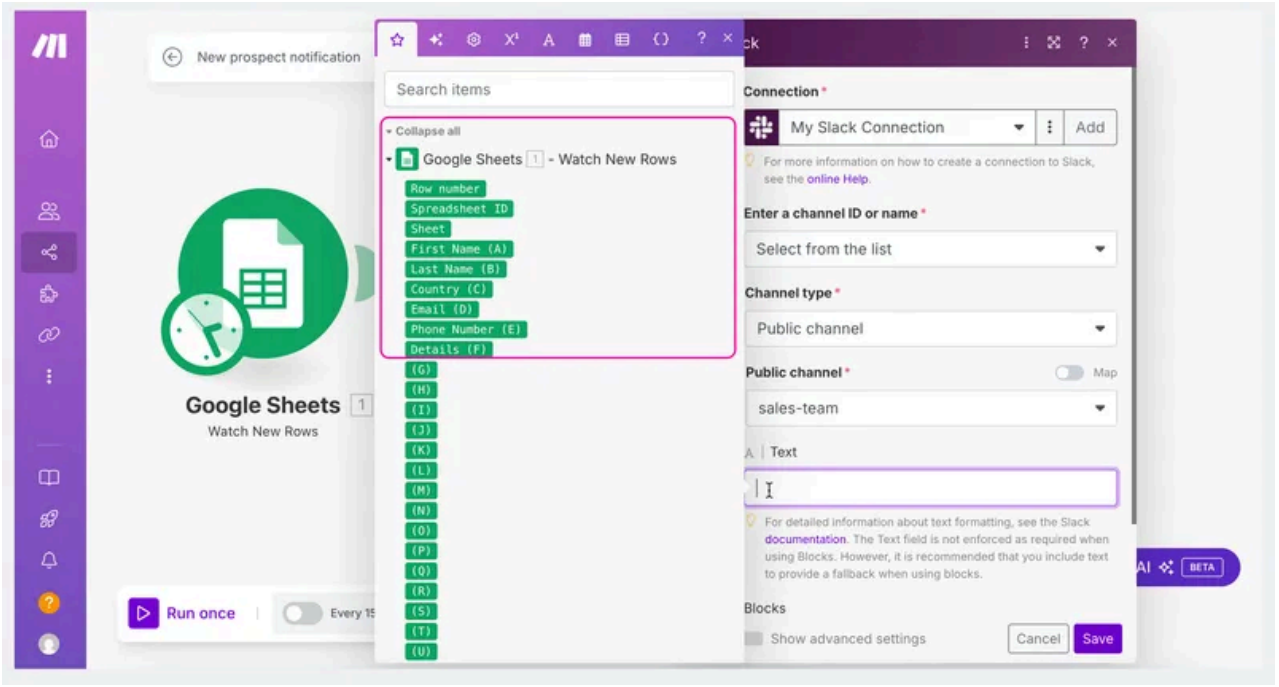
1. For **Enter a channel ID or name**, select "Select from the list".

2. In the **Channel type**, select "Public channel".

3. In the **Public channel**, select "sales-team" from the dropdown list.

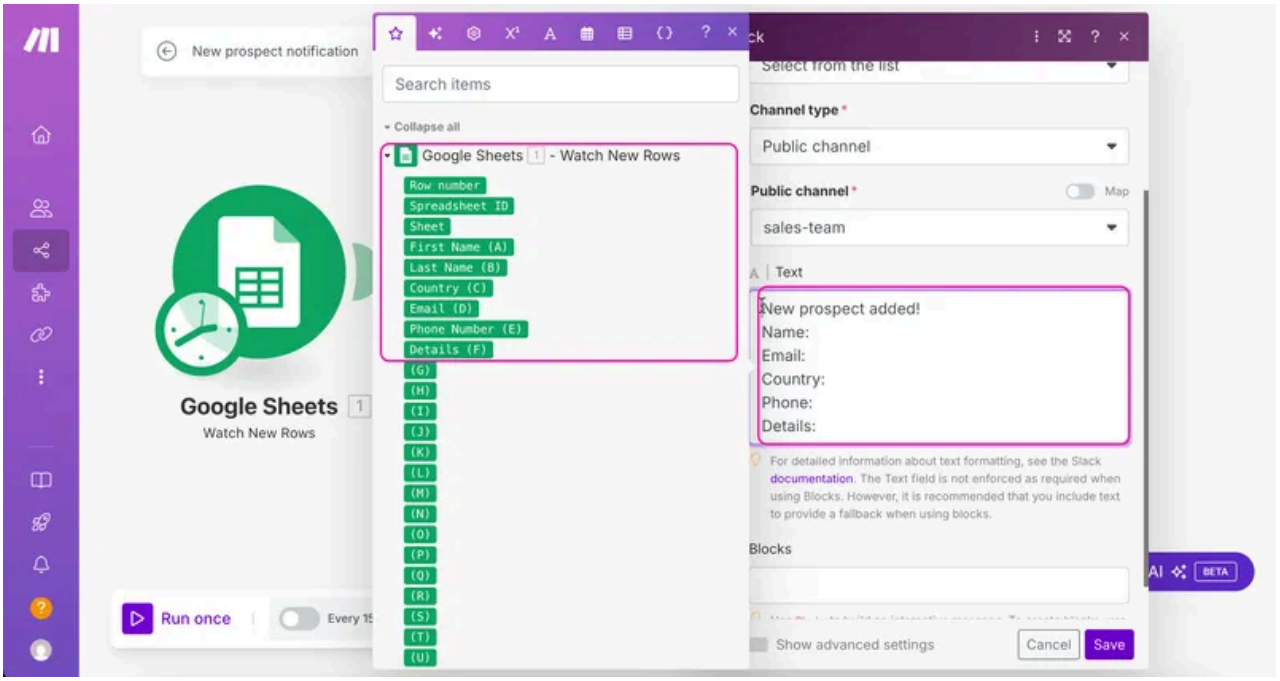


4. Click in the **Text** field, and a mapping panel will appear showing data from your Google Sheets module.

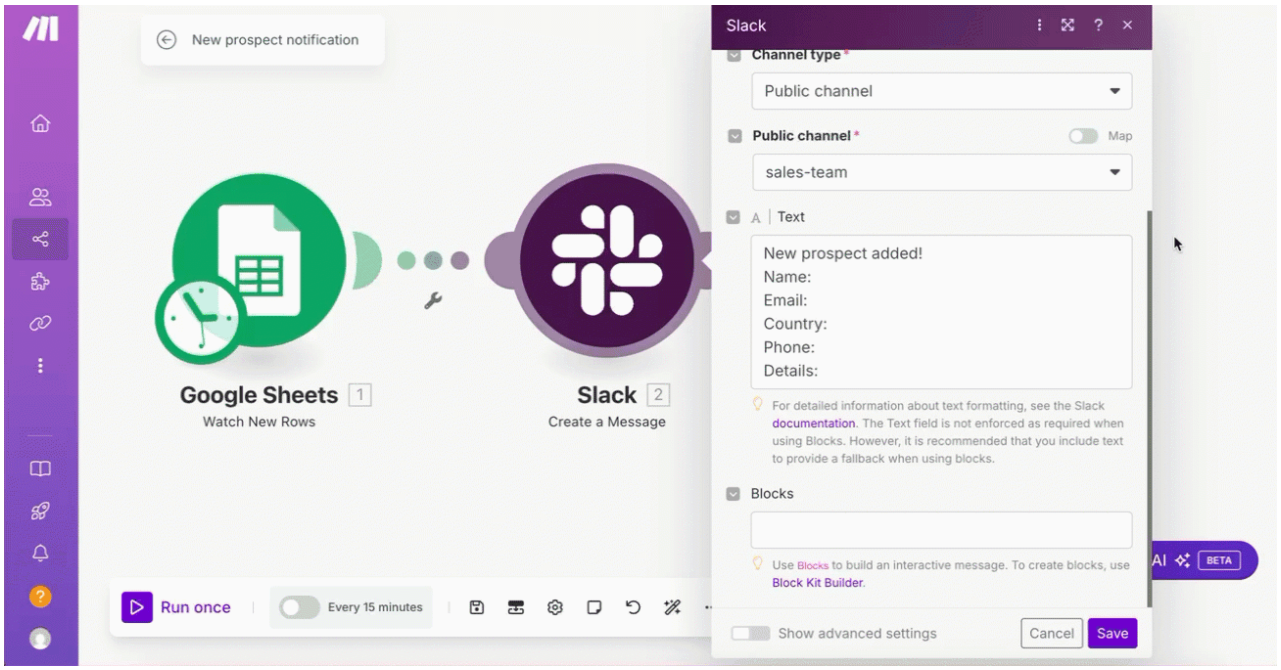


5 Create your notification message. You can copy and paste the following template into the **Text** field or create your own message.

```
New prospect added!
Name:
Email:
Country:
Phone:
Details:
```



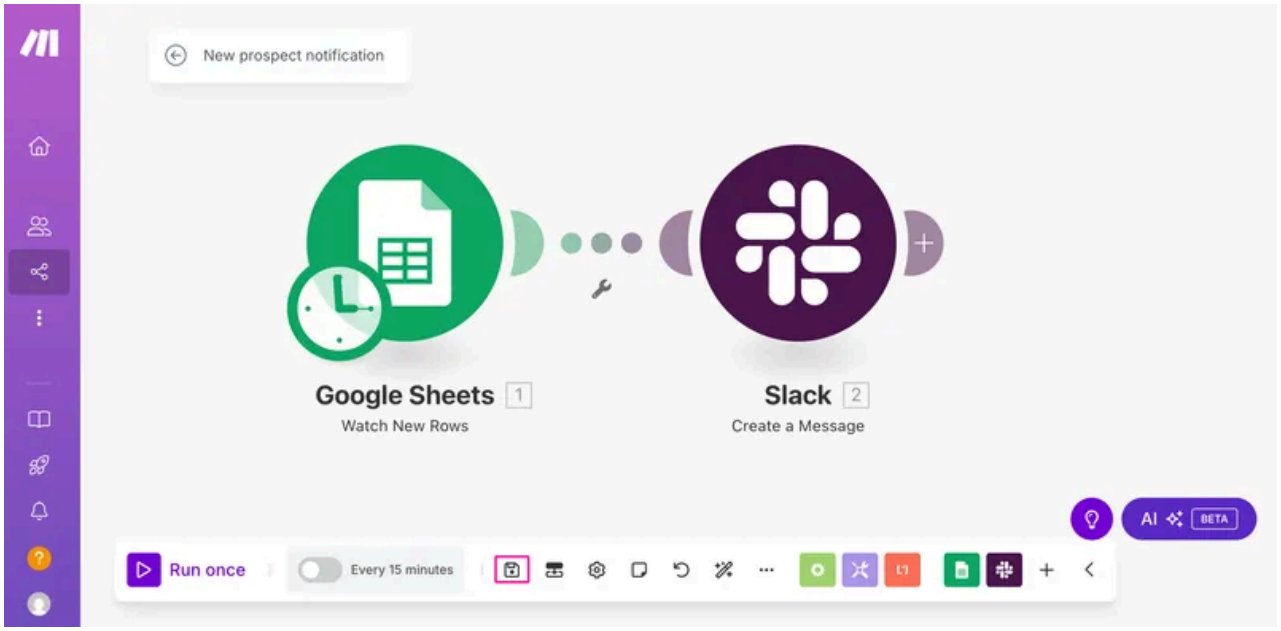
6 Drag and drop the values from the Google Sheets module to the **Text** field, or simply click on the values to place them into your message.



This is mapping data - taking prospect information from Google Sheets and placing it into specific spots in your Slack message.

7 Click **Save** to save your module settings.

8 Click the **Save** icon on the Scenario Builder toolbar to save your entire scenario.



Your scenario now has two connected modules:

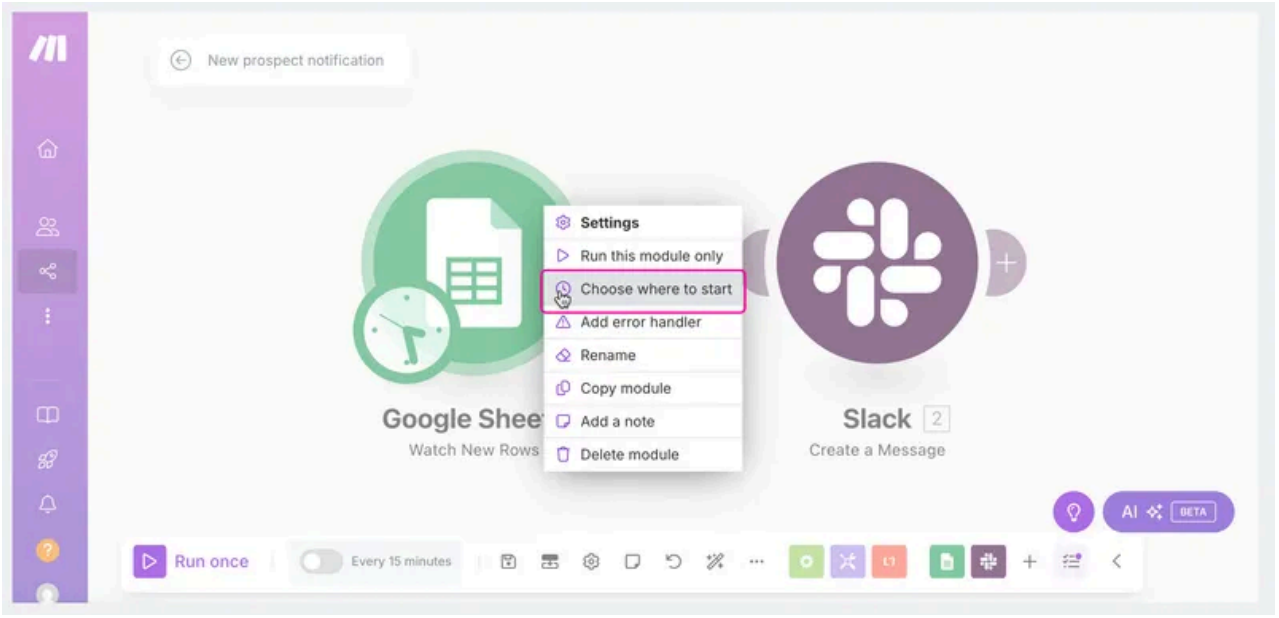
- Google Sheets "Watch New Rows" to detect new prospects
- Slack "Create a Message" to send notifications to your team

In the next step, you'll test your scenario to make sure both modules work together properly.

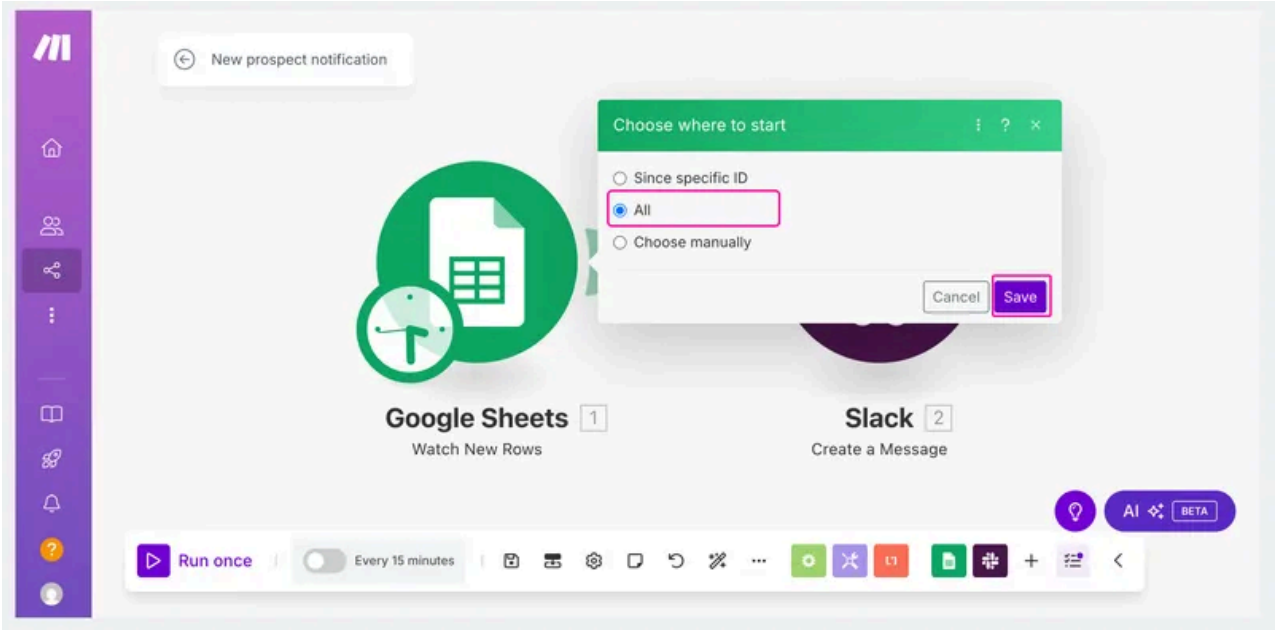
1.2.9. Step 9. Test your scenario

To make sure that both modules work together, you need to test your scenario.

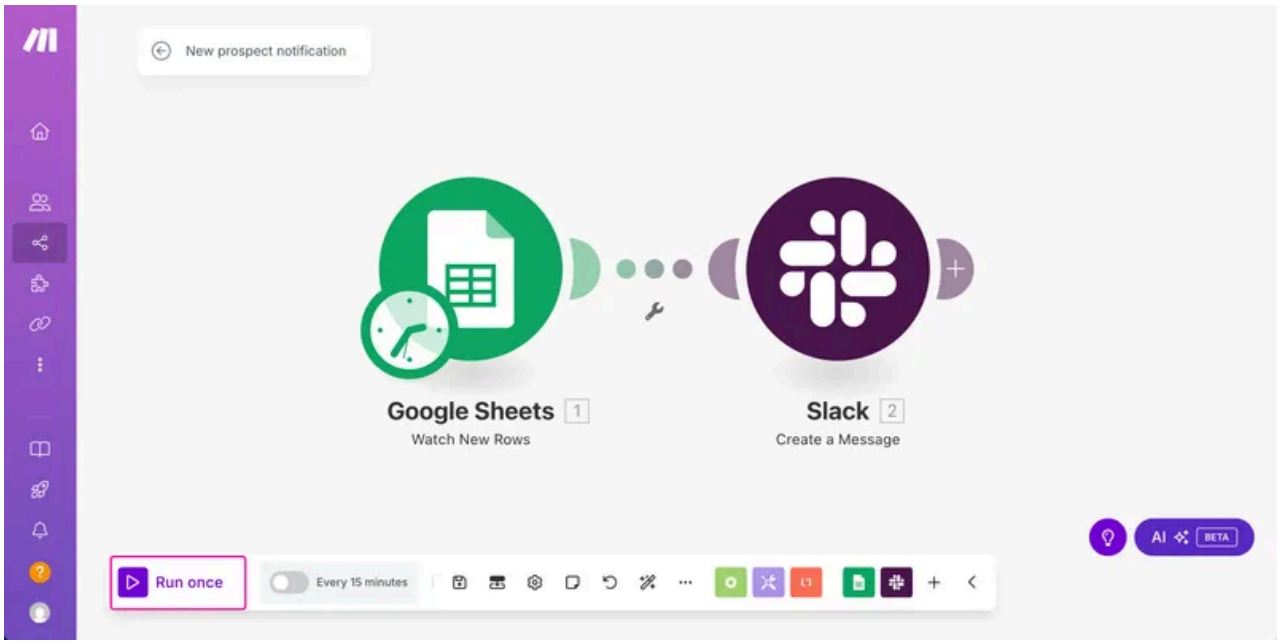
1. Right-click the Google Sheets module and select **Choose where to start**.



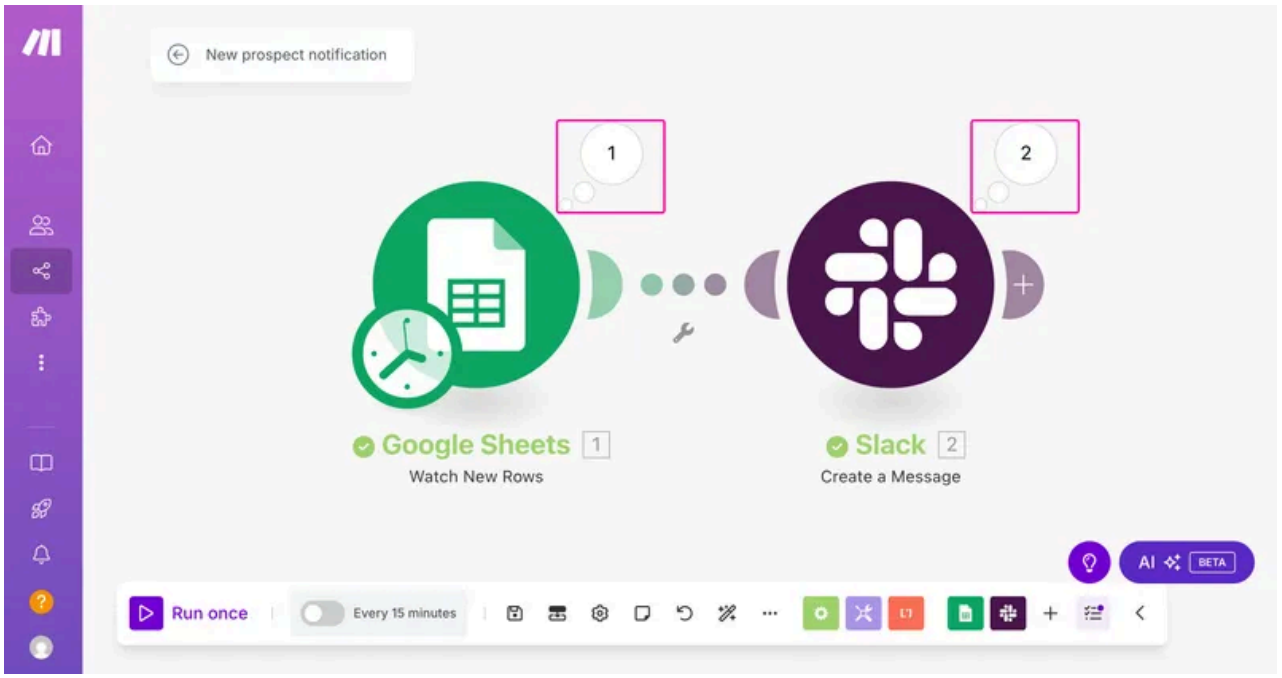
2. Select "All" and click **Save.**



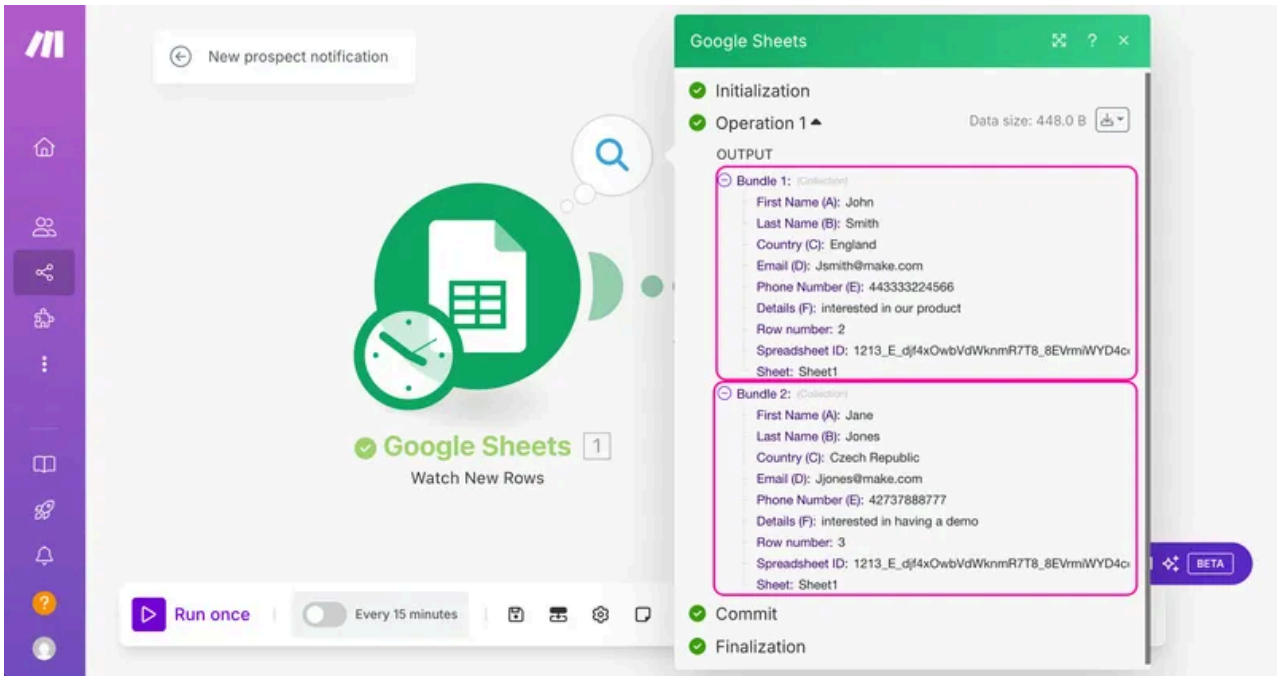
3. In the Scenario Builder toolbar, click the **Run once** button.



4 Check the operation bubbles above each module to see the results. The numbers shown in the bubbles indicate how many operations were processed.

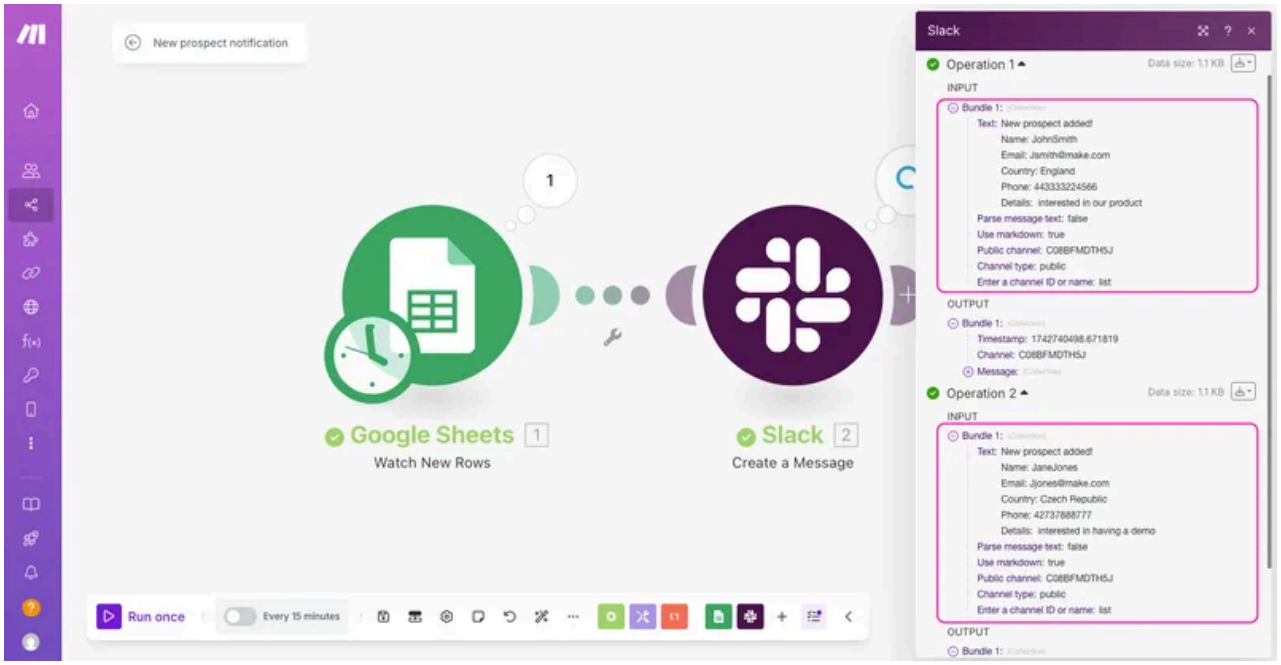


5 Click on the operation bubble above the Google Sheets module to see all rows processed in the Output section. Since we selected "All" in step 2, it processed all rows in your Google sheet (up to the limit of 20 you set earlier).

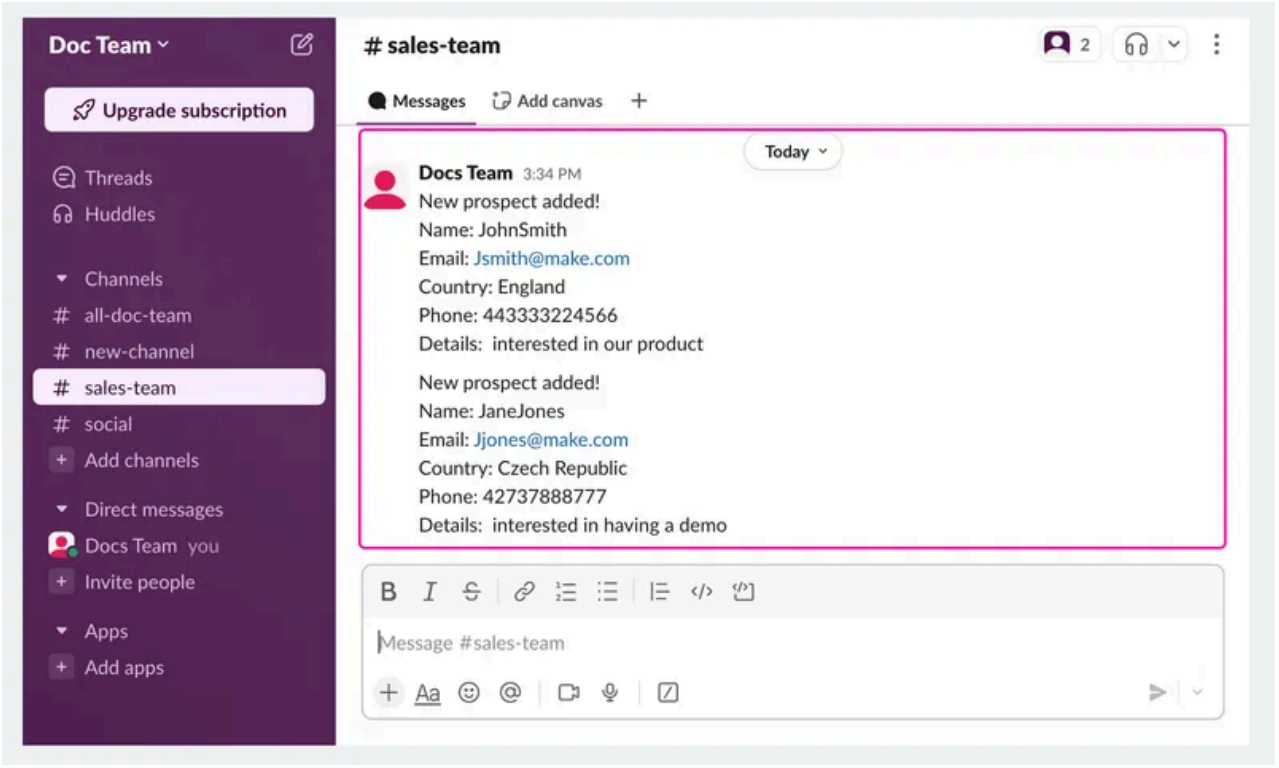


Click on the operation bubble above the Slack module to see:

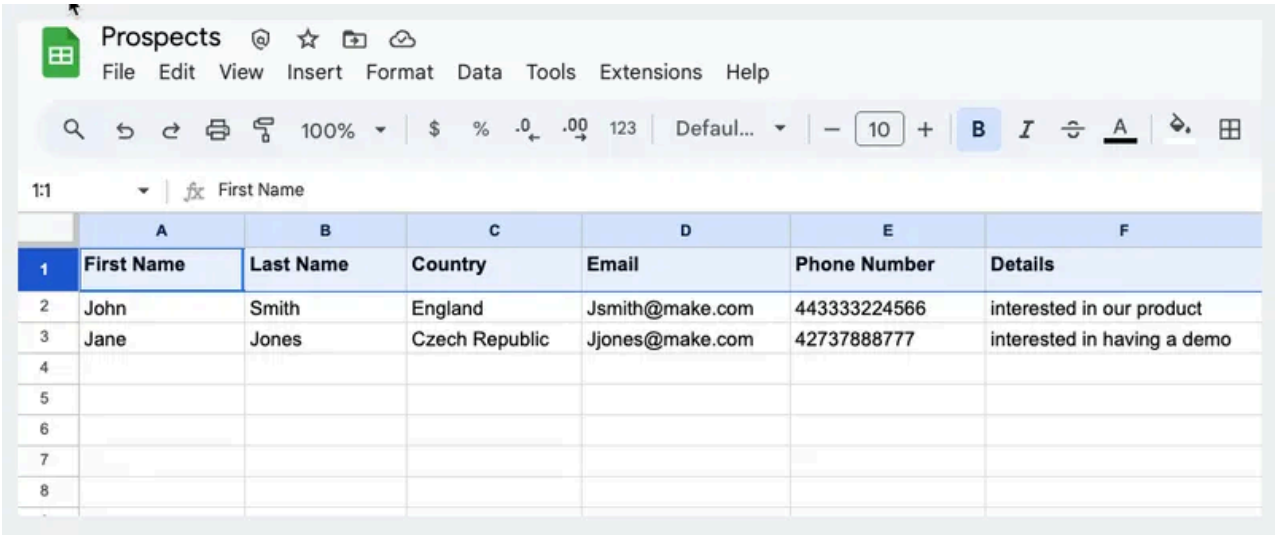
- The Input information it received from Google Sheets.
- The Output information about the Slack message it sent.



6 Go to your Slack channel and check that notification appeared correctly.



7 Compare the data in your Slack message with your Prospects spreadsheet to ensure all information was transferred correctly.



Understanding bundles

A bundle is a collection of data delivered together. It can contain a single piece of information (like an email address) or multiple pieces of data (like spreadsheet rows). When you click on the operation bubbles, you'll see the bundles that were processed:

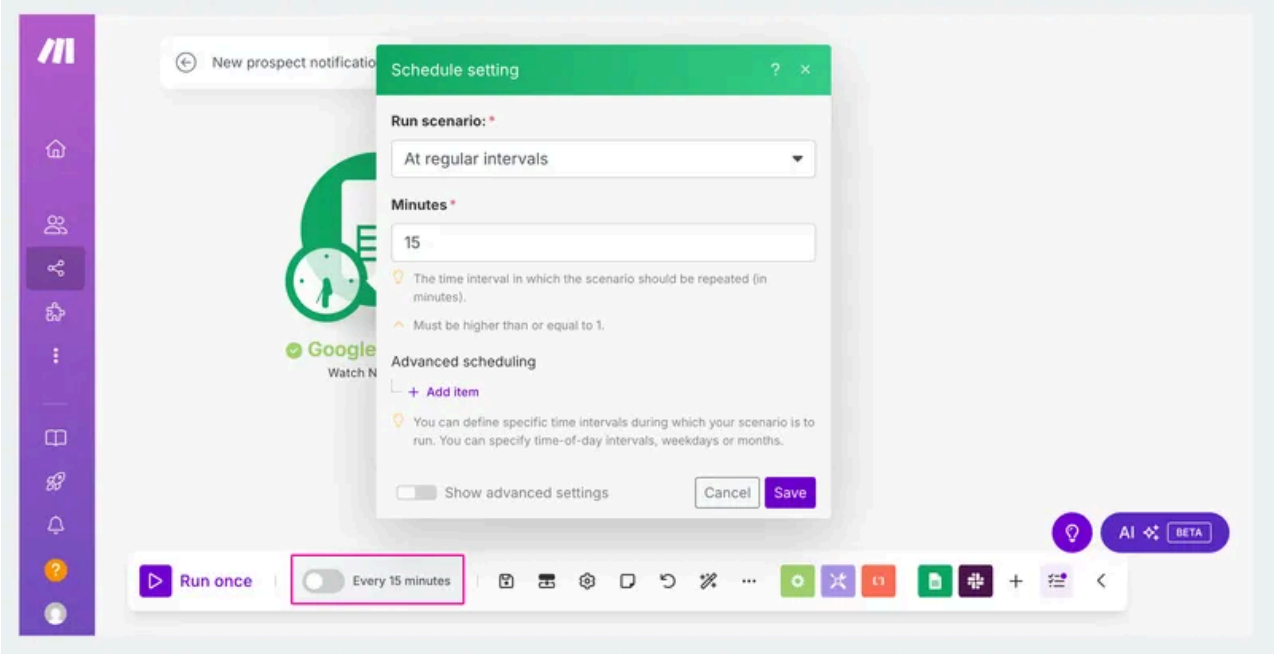
- **Input bundle**: Contains the information you entered into the module and data you mapped from previous modules.
- **Output bundle**: Contains the results of the actions taken in the application and any data generated that can be mapped to the next module.
- Each bundle processed counts as one operation for action modules (like your Slack module).
- Trigger modules (like your Google Sheets module) only use one operation, even if they output multiple bundles.

For example, if your Google sheet has 3 rows and you selected "All," the Google Sheets module will generate 3 output bundles but only use 1 operation because it's a trigger module. These become input bundles for the Slack module, which processes them and sends 3 messages. The Slack module uses 3 operations because it processes three input bundles.

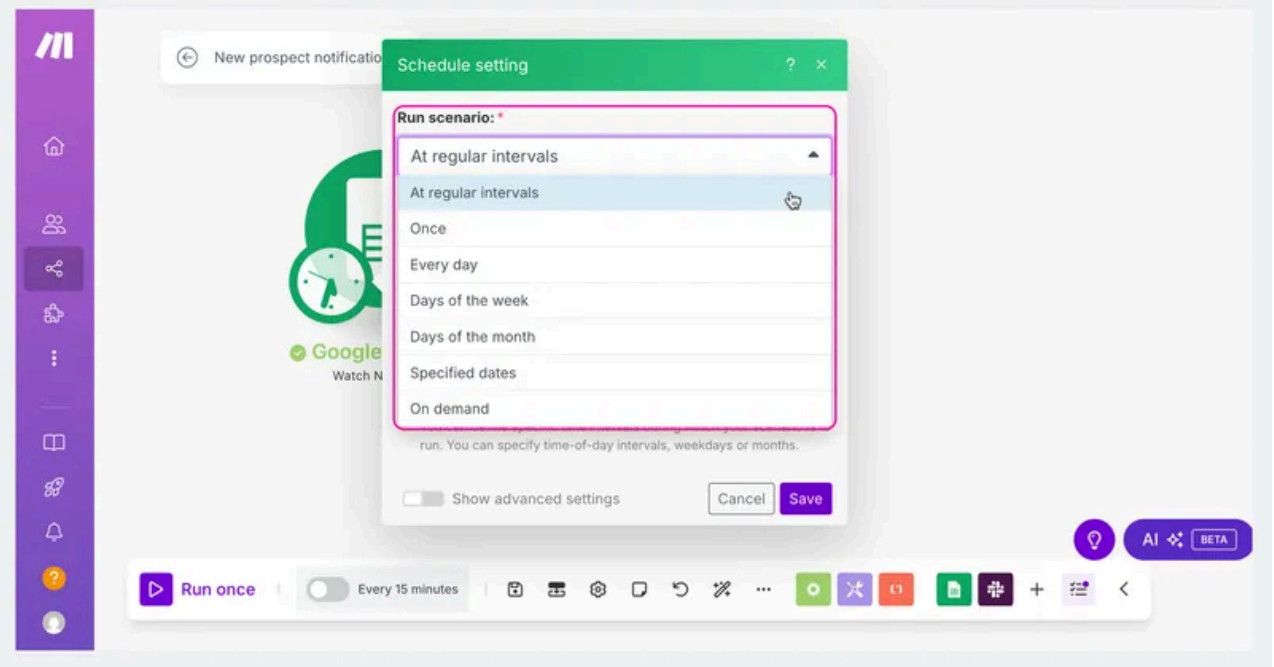
1.2.10. Step 10. Schedule your scenario

The final step is to set up how often your scenario should run:

1. In the Scenario Builder toolbar, click **Every 15 minutes** to open the schedule settings.



2. In the **Schedule setting** window that appears, set how frequently your scenario should run.
   For this scenario, keep the default "Every 15 minutes" option. This means your Google Sheets module will check for new rows every 15 minutes. You can adjust this based on how quickly you need to be notified about new prospects.



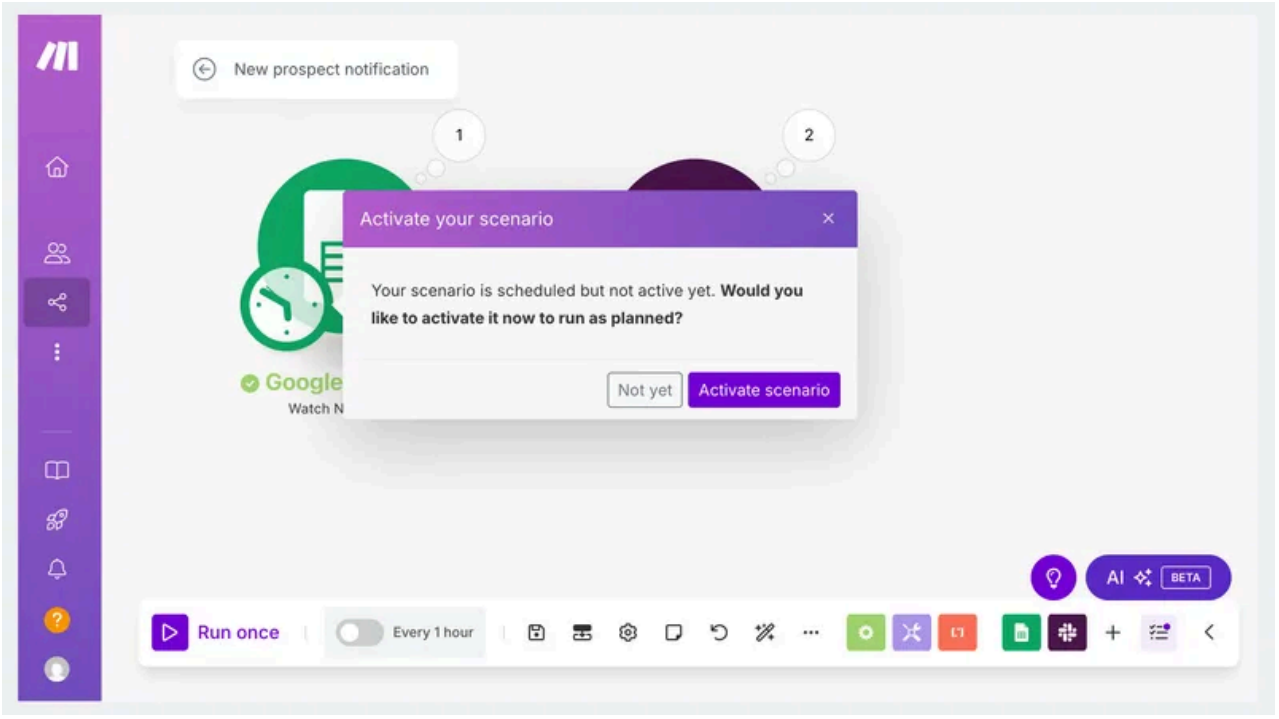
When scheduling a scenario, consider how frequently you need to check for new data, as this affects your operation consumption:

- Running every 5 minutes = 288 operations per day (60/5 × 24), or 8,640 operations per month
- Running every hour = 24 operations per day, or 720 operations per month
- Running once daily = 30 operations per month

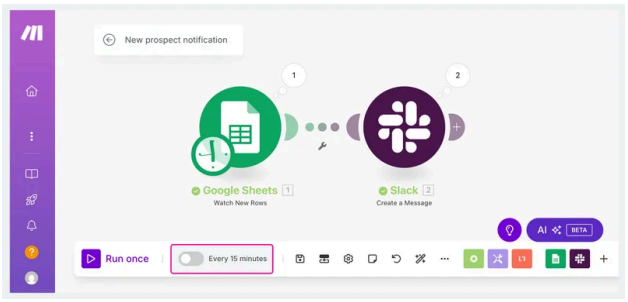
These numbers show the operation consumption for the trigger module only. The complete scenario will consume additional operations based on the amount of new data processed and the complexity of the modules in your scenario.

3 Click **Save**.

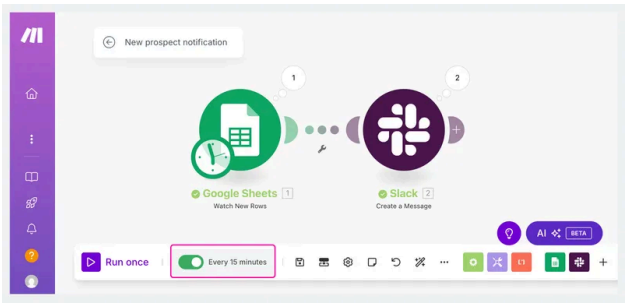
4 After you schedule your scenario, you'll be prompted to activate it. Click **Activate scenario**.



If you schedule and activate a scenario only for testing purposes, we recommend deactivating it after or adjusting the schedule to avoid using operations unnecessarily.



Scenario is not activated



Scenario is activated

Well done! You've created your first scenario in Make. It will now run according to your schedule, monitoring your Google Sheet for new prospects and sending notifications to your Slack channel.

What's next?

You can stop at this step or continue and expand your scenario with the following:

- Add a router to create multiple paths in your workflow.
- Use filters to process data conditionally.
- Implement aggregators to combine multiple items into one.

These features will be covered in the **Expand your scenario** section.

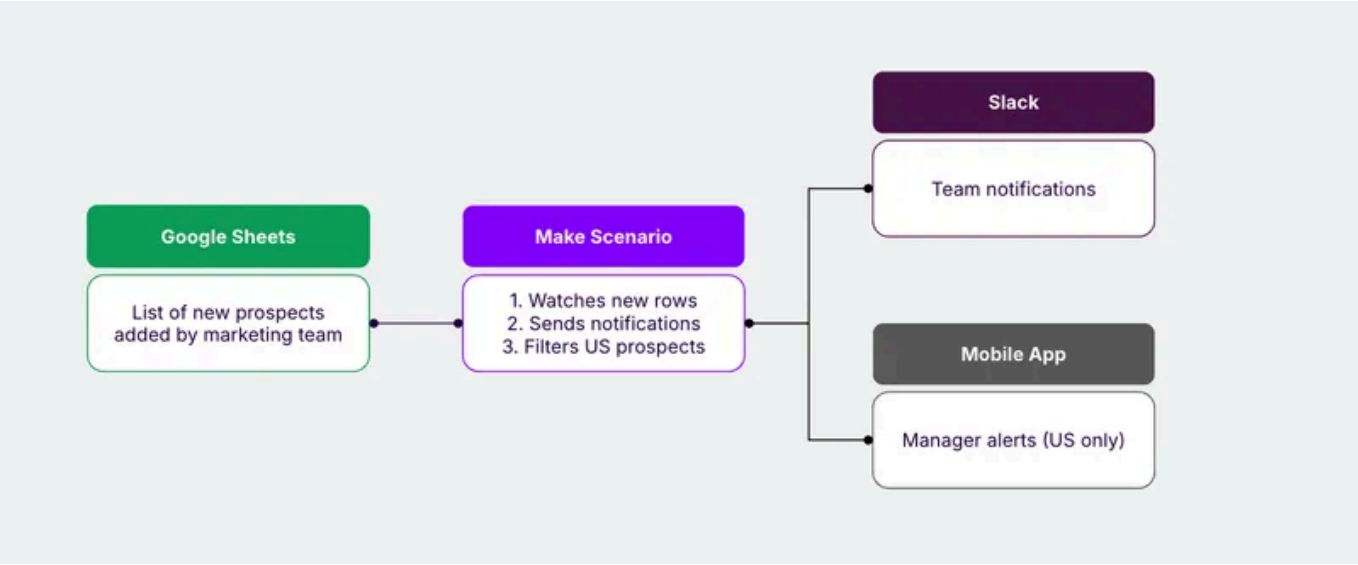
1.3. Expand your scenario

Now that you've **created your first scenario** that monitors Google Sheets for new prospects and sends Slack notifications, you can take your scenario further. This guide will walk you through using routers, filters, and aggregators to create targeted notifications that make your automation more effective.

What you'll build

Your expanded Make scenario will:

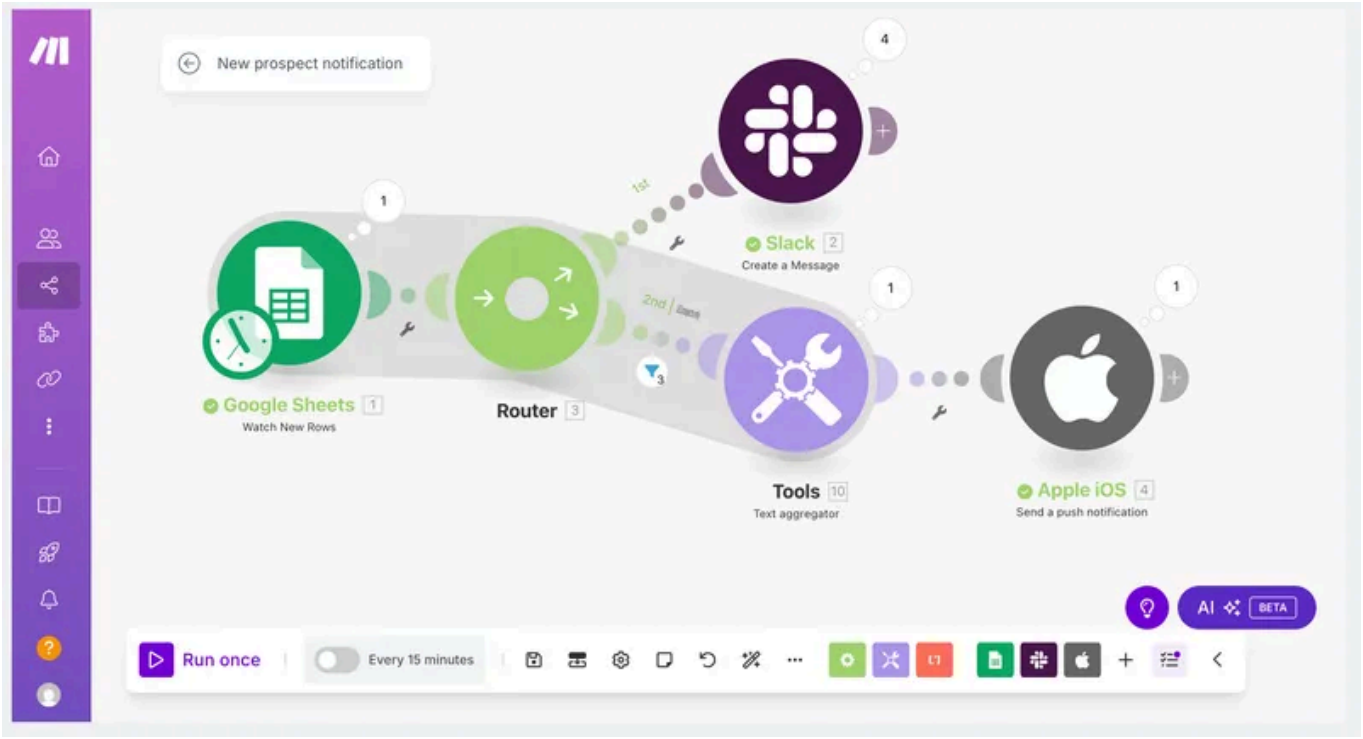
- Monitor Google Sheets for new prospect entries by the marketing team.
- Send Slack notifications to the sales team when new prospects are added.
- Send mobile notifications to the Sales Manager for US-based prospects.
- Aggregate multiple US prospects into a single notification.



What you'll need

- Your Make scenario that you created following the steps in **Create your first scenario**
- Make mobile app (**iOS** / **Android**)

Here's a preview of what your expanded scenario will look like:


New prospect notification
Google Sheets 1
Watch New Rows
Router 3
Slack 2
Create a Message
Tools 10
Text aggregator
Apple iOS 4
Send a push notification
Run once
Every 15 minutes
AI BETA

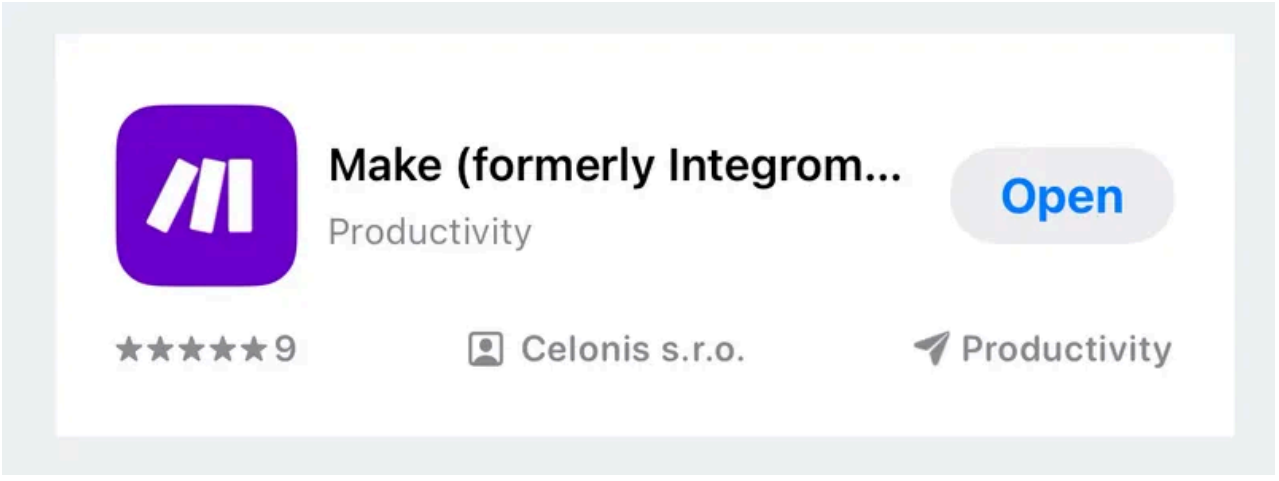
1.3.1. Step 1. Get your app ready

In the **first part of this guide**, you set up Slack and Google apps to create a basic notification system. Now, to expand your scenario and send targeted mobile notifications to the Sales Manager about any US-based prospects received, you need to set up the Make mobile app.

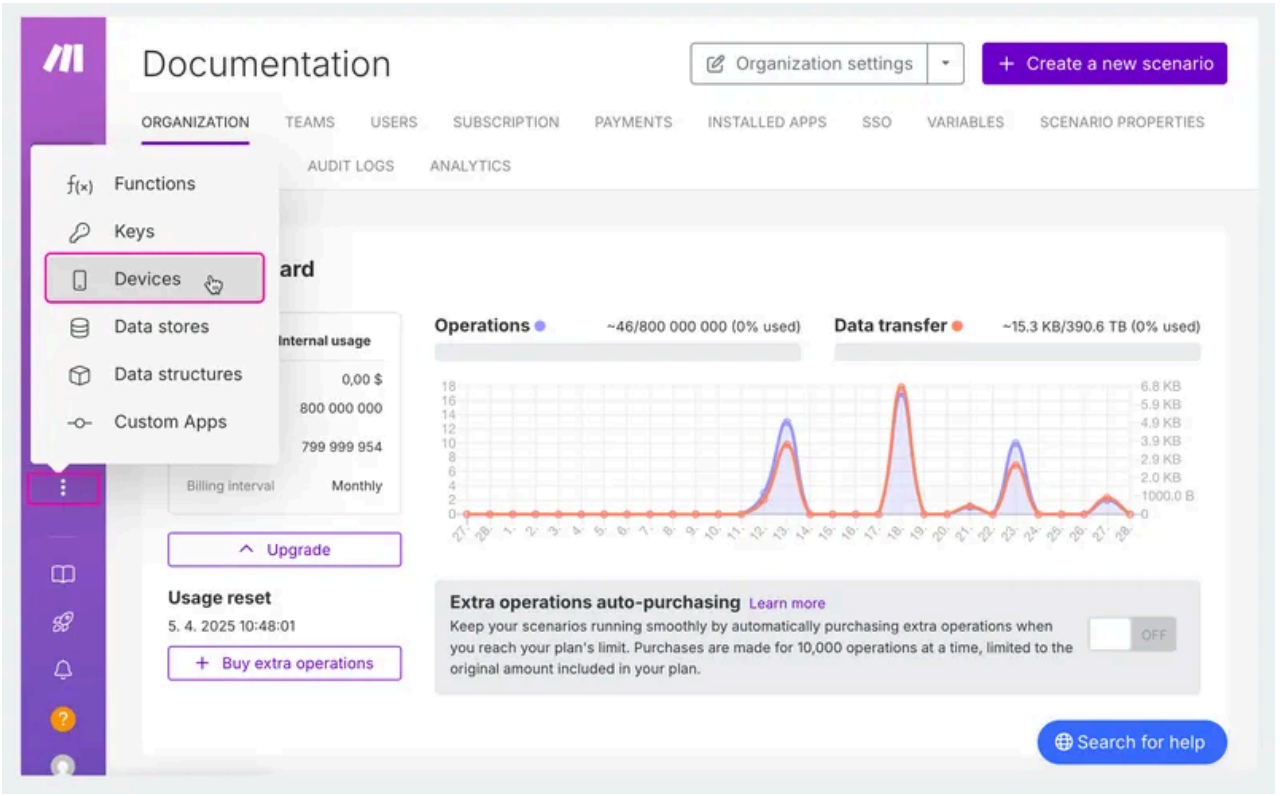
> The Make mobile app brings your automations to your smartphone, sending you push notifications about important events. This helps team members receive real-time alerts without checking Slack or email. For Sales Managers on the go, these instant notifications ensure they never miss high-priority prospects.

To set up the Make mobile app:

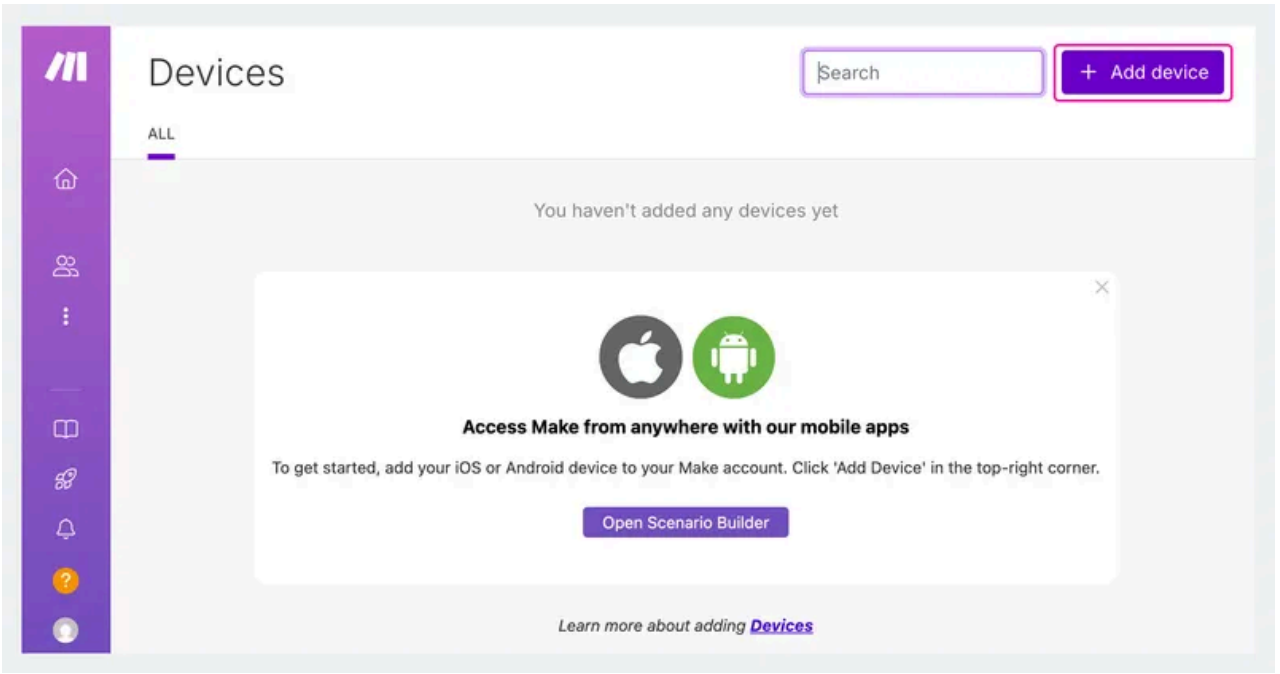
1. Go to your device's app store:
   - **iOS App Store**
   - **Google Play Store**
2. Search for "Make" or "Make (formerly Integromat)".
3. Download and install the Make app.



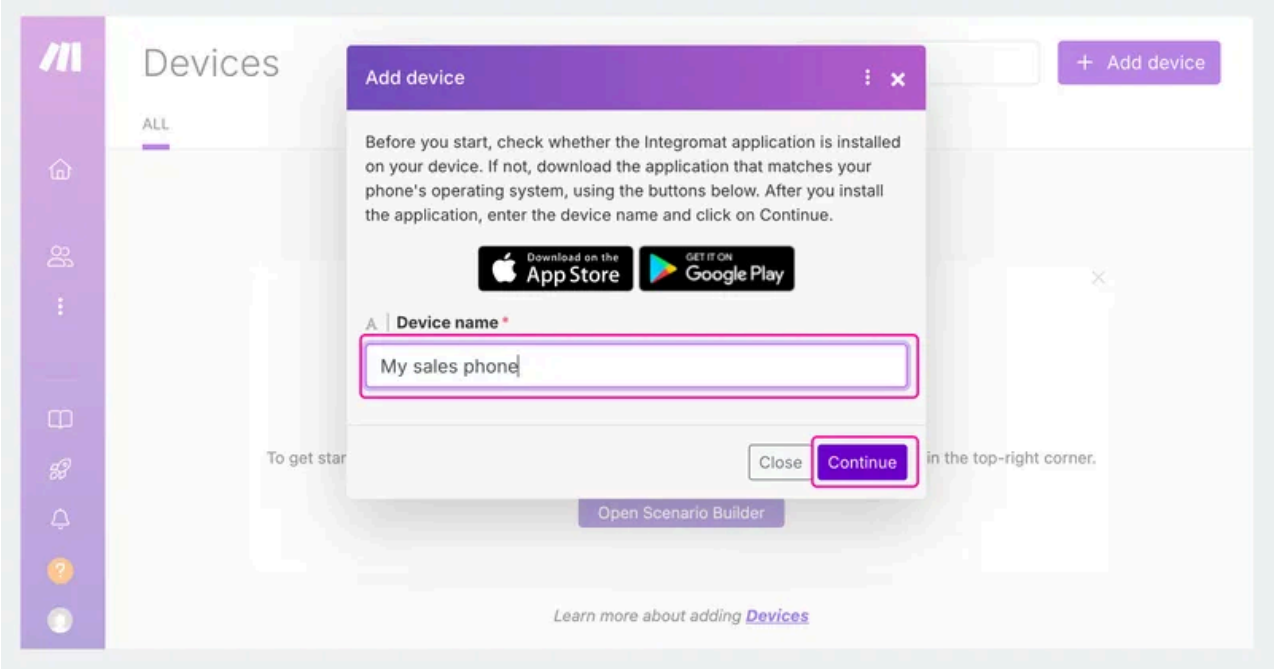
4 Log in to Make. In the left side navigation menu, click **More > Devices**.



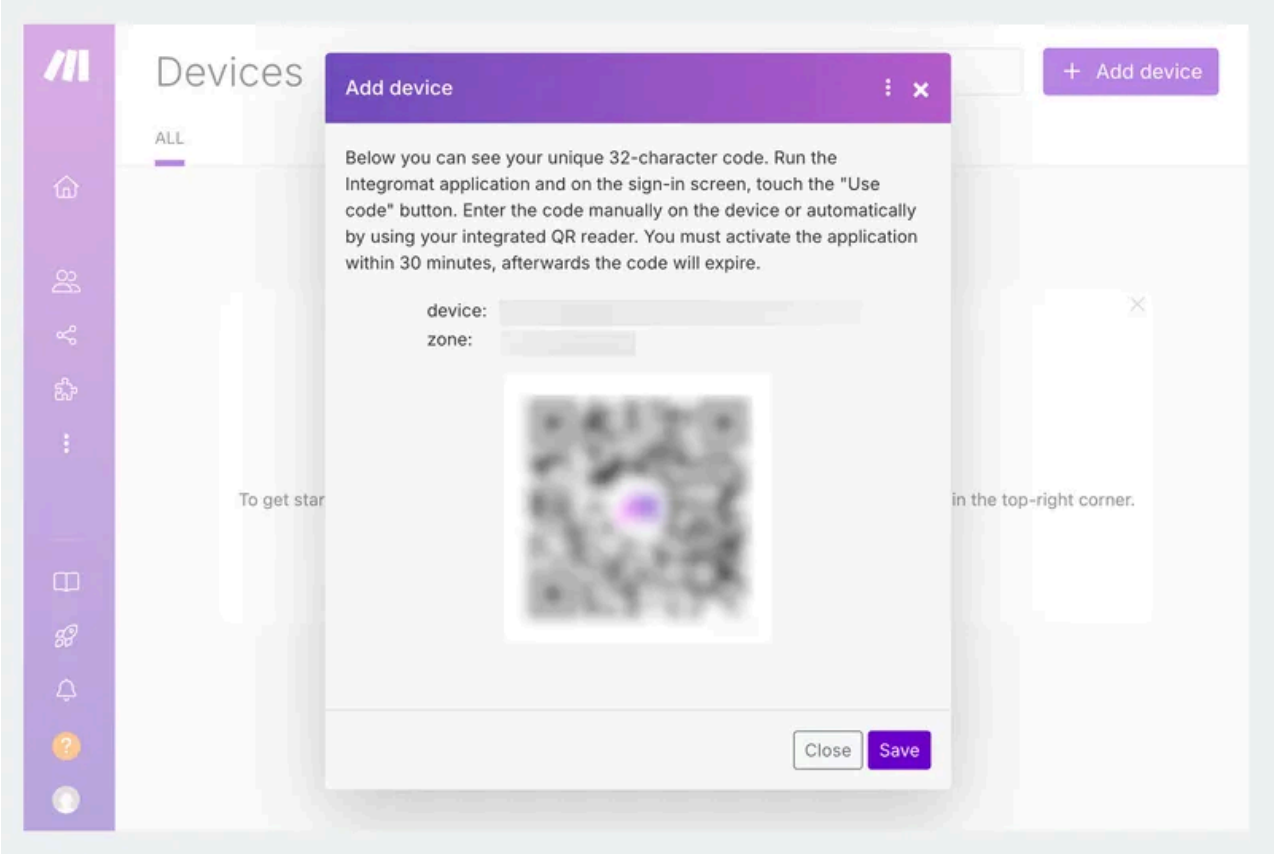
5 Click **+ Add device** in the top right corner.



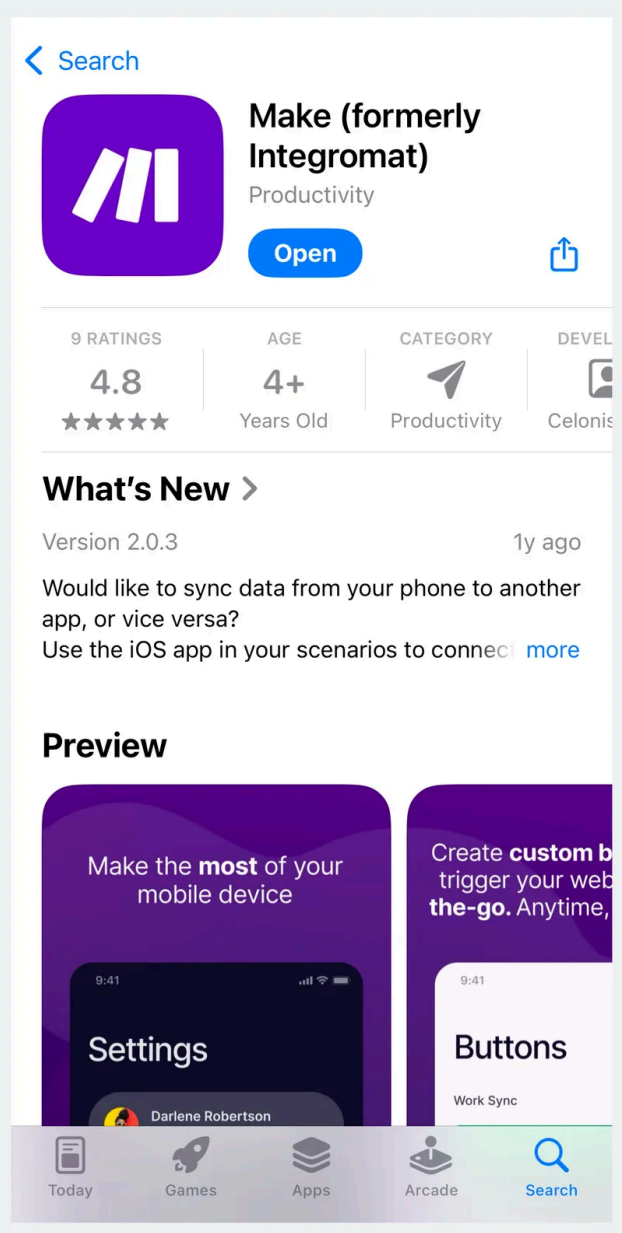
6 In the Add device window, type a name for your device (for example, "My sales phone") and click **Continue.**

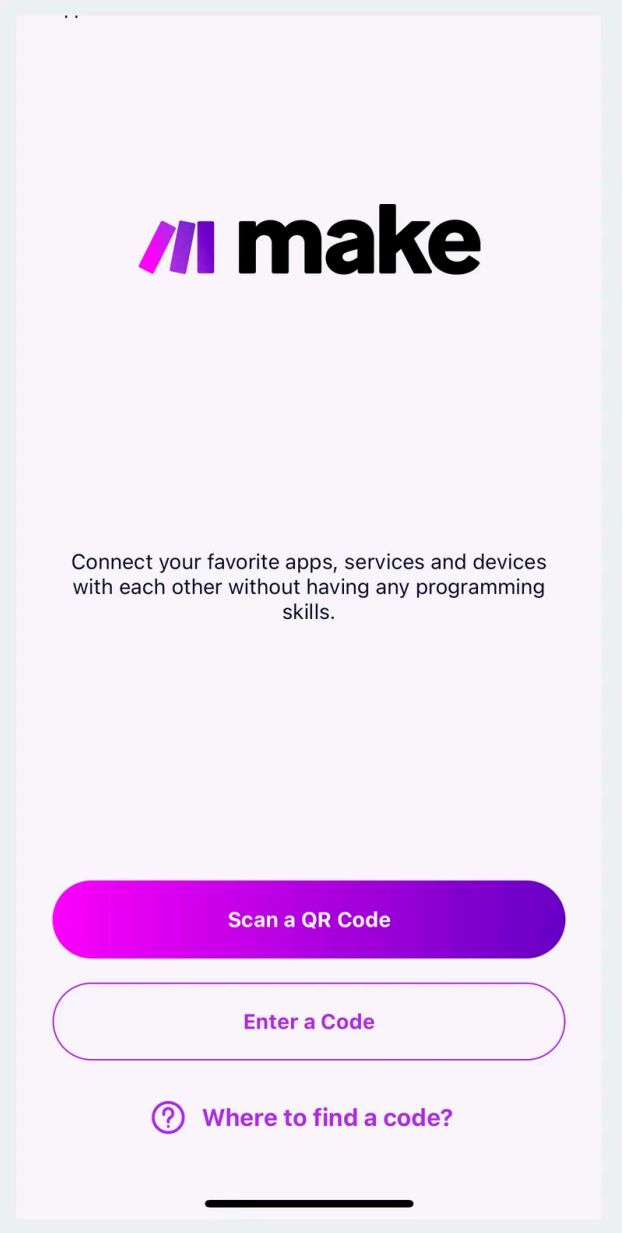


The screen with the QR code will appear.

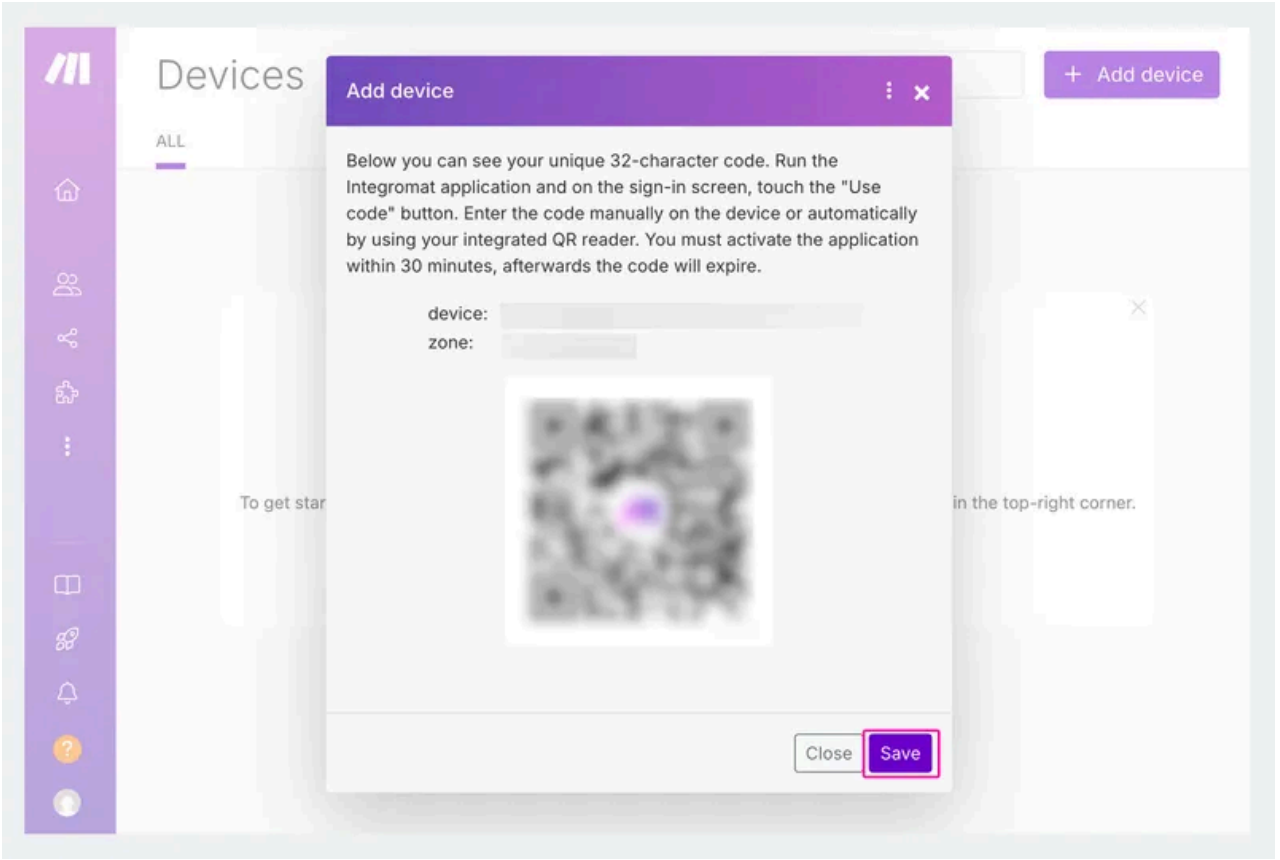


7 Go back to your mobile phone and open the **Make** app. In the screen that appears, tap **Scan a QR Code**. Scan the code you see in Make. If scanning doesn't work, you can manually enter the code displayed.

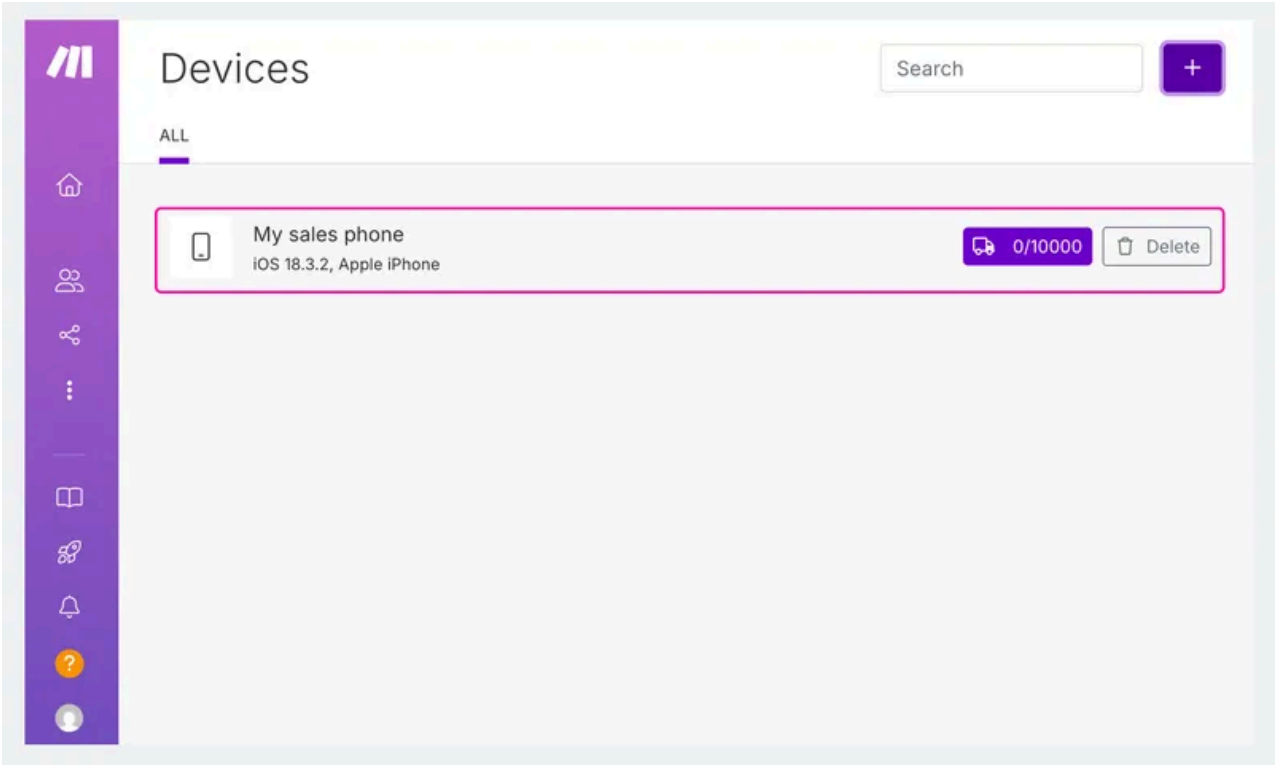




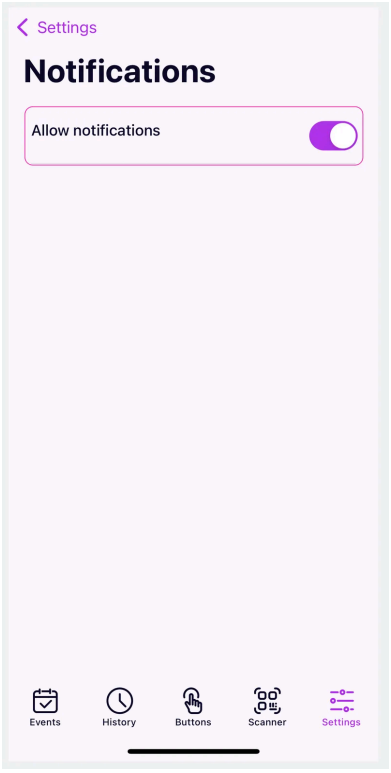
8 Come back to Make and click **Save.**



You should see the confirmation message that the device was added successfully and the device you added displayed on the Devices page.



9 In the Make mobile app, go to **Settings > Notifications**, and switch on the **Allow notifications** toggle. This will allow you to be notified whenever a new prospect is added to your spreadsheet.



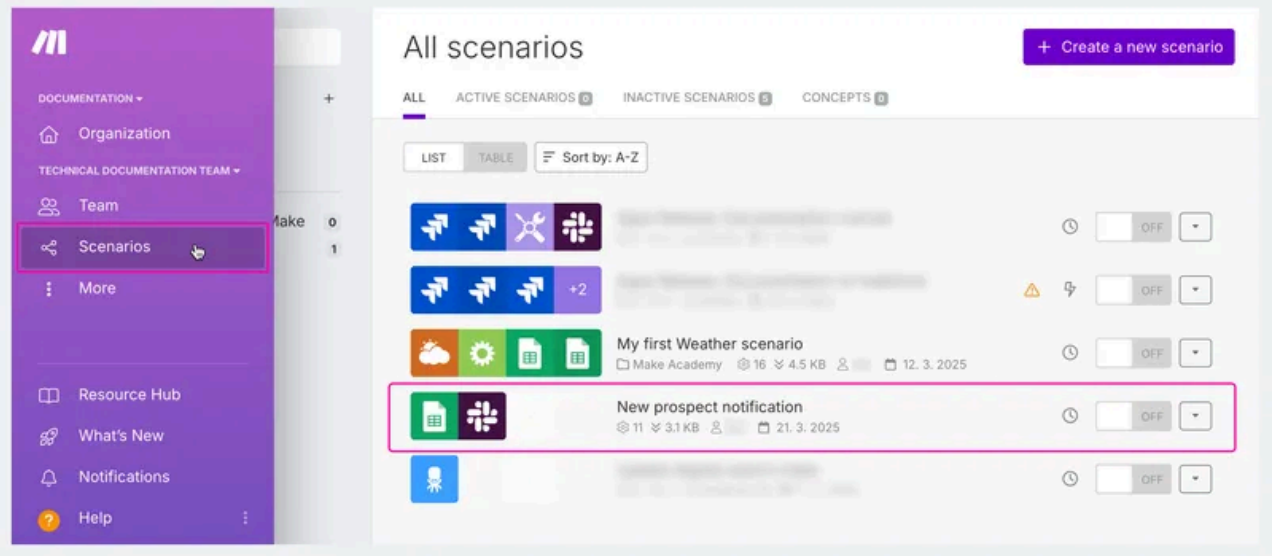
Your Make mobile app is installed and set up in Make. You are ready for the next step — adding a router to direct your scenario so that all prospects go to Slack, while only US-based prospects trigger mobile notifications.

1.3.2. Step 2. Add a router

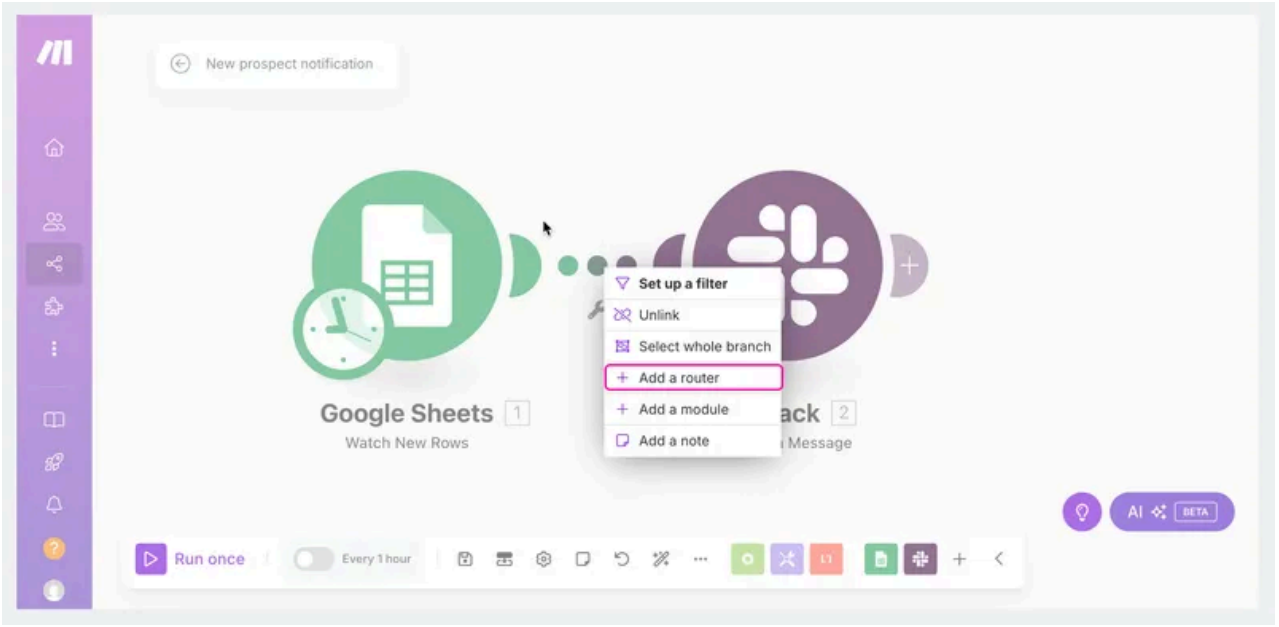
A router allows you to split your scenario flow into multiple routes. Think of it like a fork in the road where your data can follow different paths based on your scenario requirements. This lets you process the same information in different ways - for example, sending all prospect notifications to Slack while only sending specific ones to mobile devices.

To add a router to your scenario:

1. In the left side navigation menu, click **Scenarios** and select the "New prospect notification" scenario you built in the **Create your first scenario** section.



2. Right-click the dots between the Google Sheets and Slack modules and select **Add a router**.



A router module will appear between the modules.

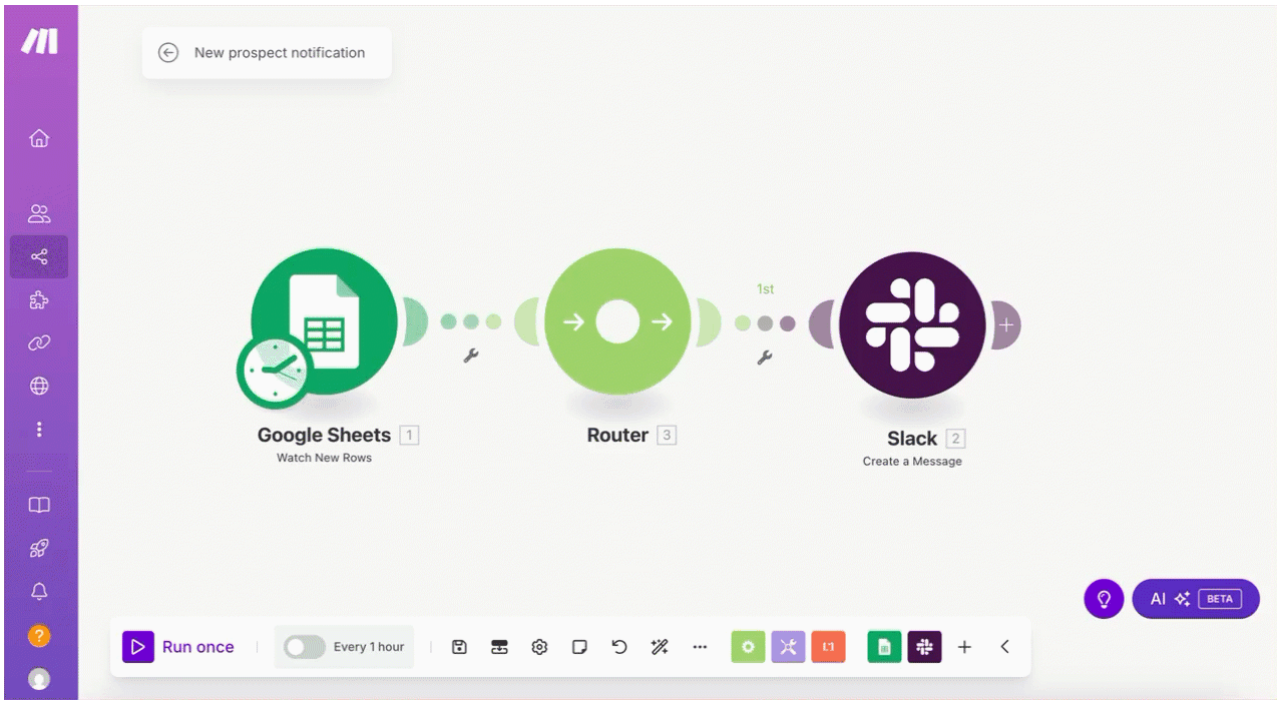


Alternatively, you can add a router the same way you add any other module:

- Click **+ Add another module** on the right side of your module, search for "Router" in the search field, and select **Router** from the results.

Please note that the right-click option is recommended when you need to place a router between two existing modules. The alternative method adds the router at the end of your scenario, which is useful when building a scenario from scratch but requires additional reconfiguration when working with existing modules.

3 Hover over the router and click the **+** icon that appears. This creates a second route that branches from the router.



Your scenario now has two routes: the original route to Slack where all prospects are sent, and a new route that you'll set up in the next step to send only US-based prospects to the mobile app.

Understanding routers

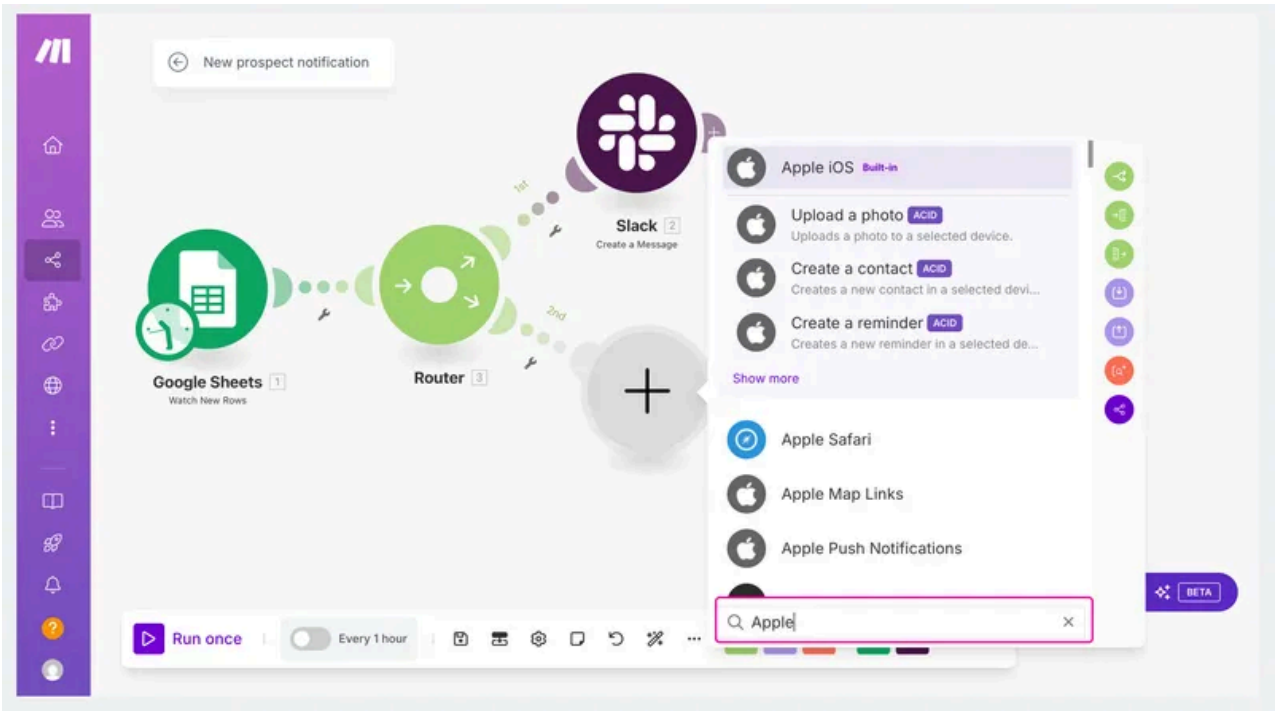
A router is a Make tool that allows you to branch or split your scenario flow into multiple routes:

- Each route after a router can process data differently
- You can add conditions to each route using filters (which we'll cover in **Step 4. Add a filter**)
- You can direct specific data to different destinations based on your requirements
- Routers don't consume operations, making them efficient for complex workflows

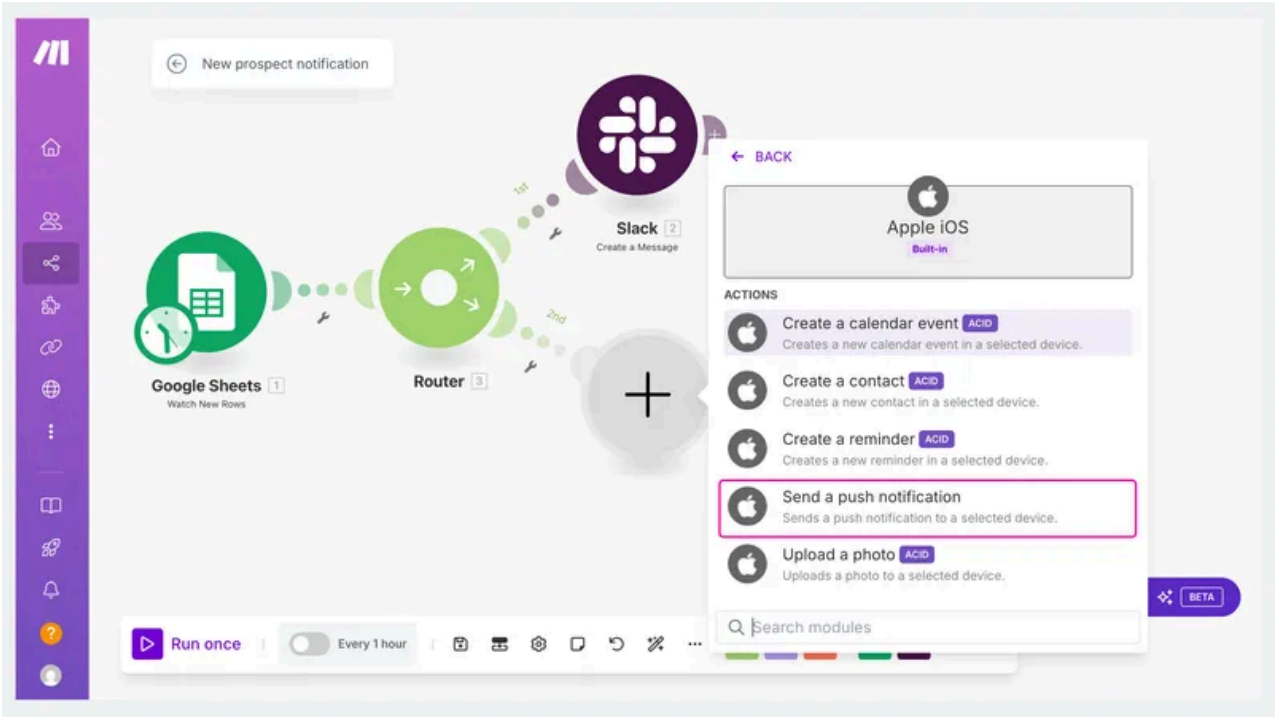
1.3.3. Step 3. Set up another module

Now you'll set up a module on the new route to send notifications to the Sales Manager's mobile device.

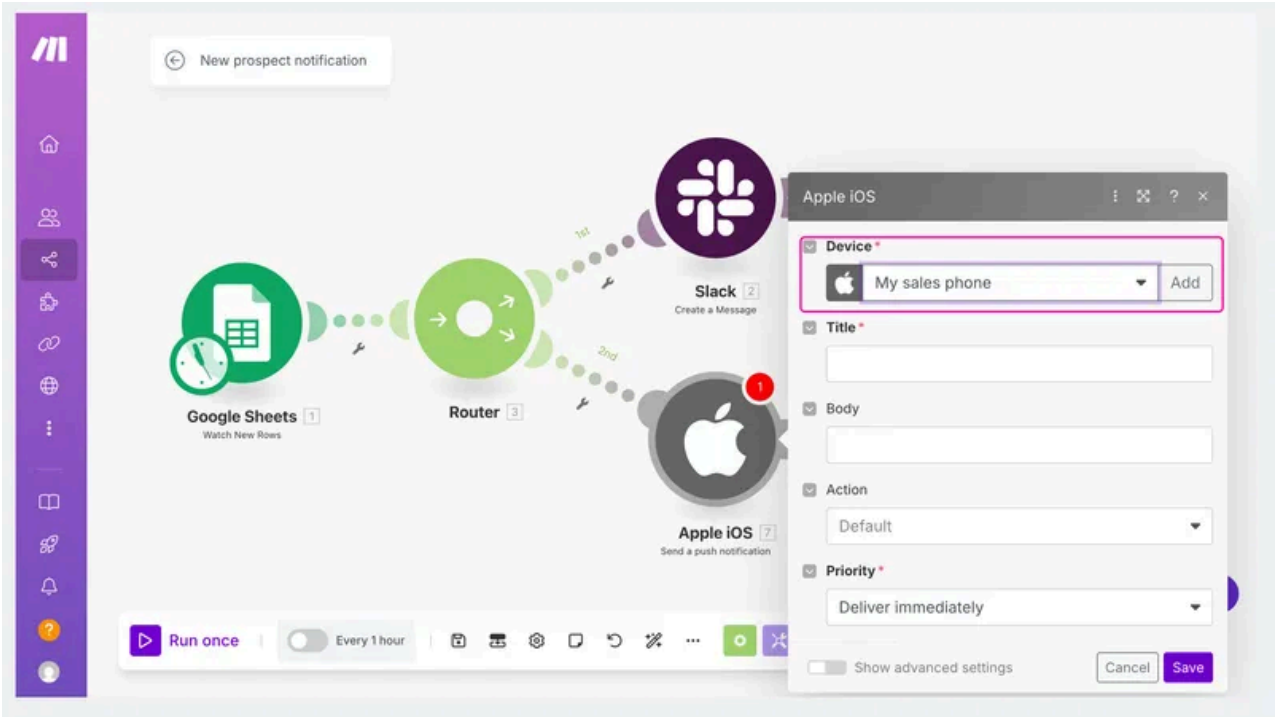
1. Click the **+** icon on the empty module at the end of the new route. Search for "Android" or "Apple iOS" in the search field, depending on the system where you installed the mobile app. In this example, we use the iOS version.



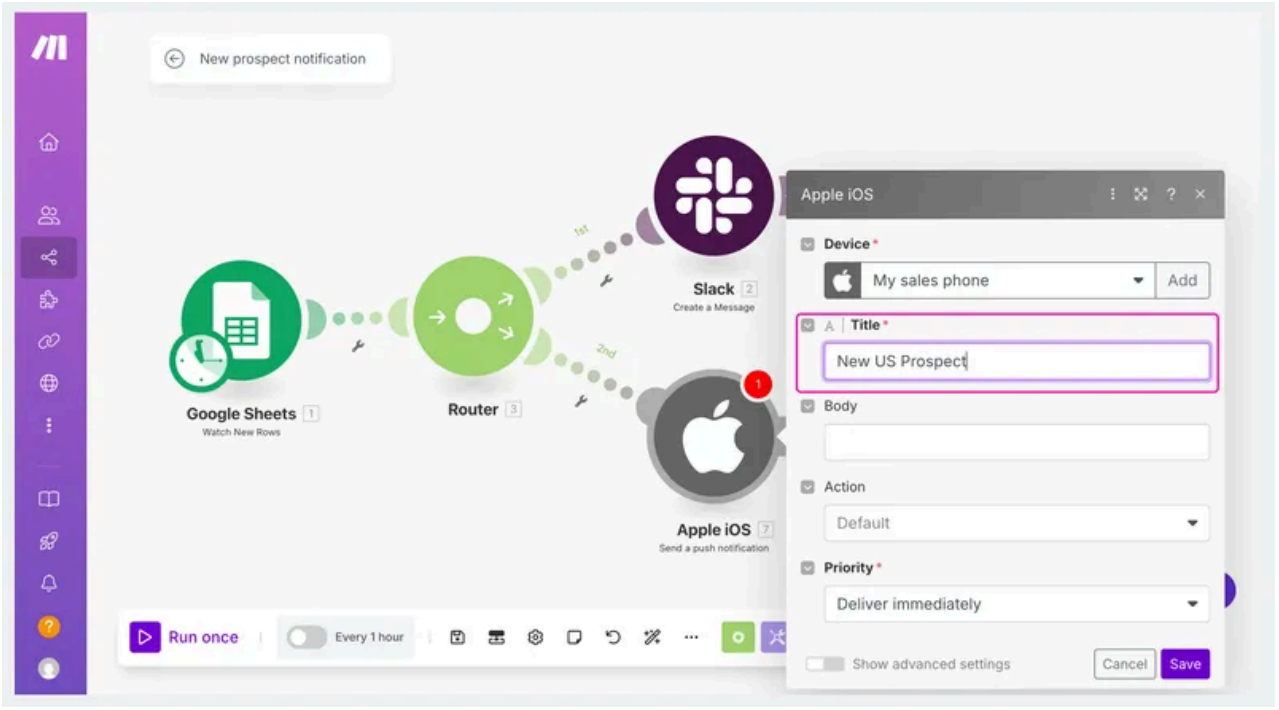
2. Select the app and then choose "Send a push notification" from the available modules. The module settings window will appear where you'll set up your notification.



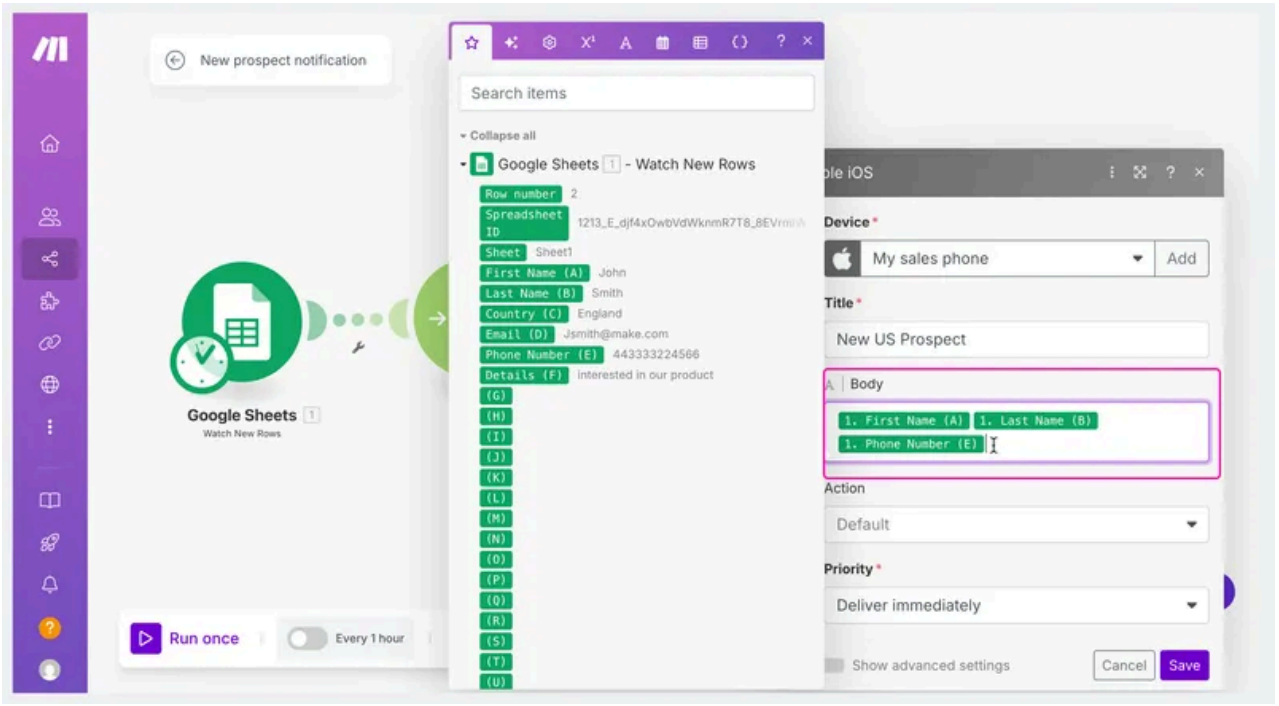
3 In the **Device** field, select your mobile device from the dropdown list.



4 In the **Title** field, enter "New US Prospect" or any other title you prefer.

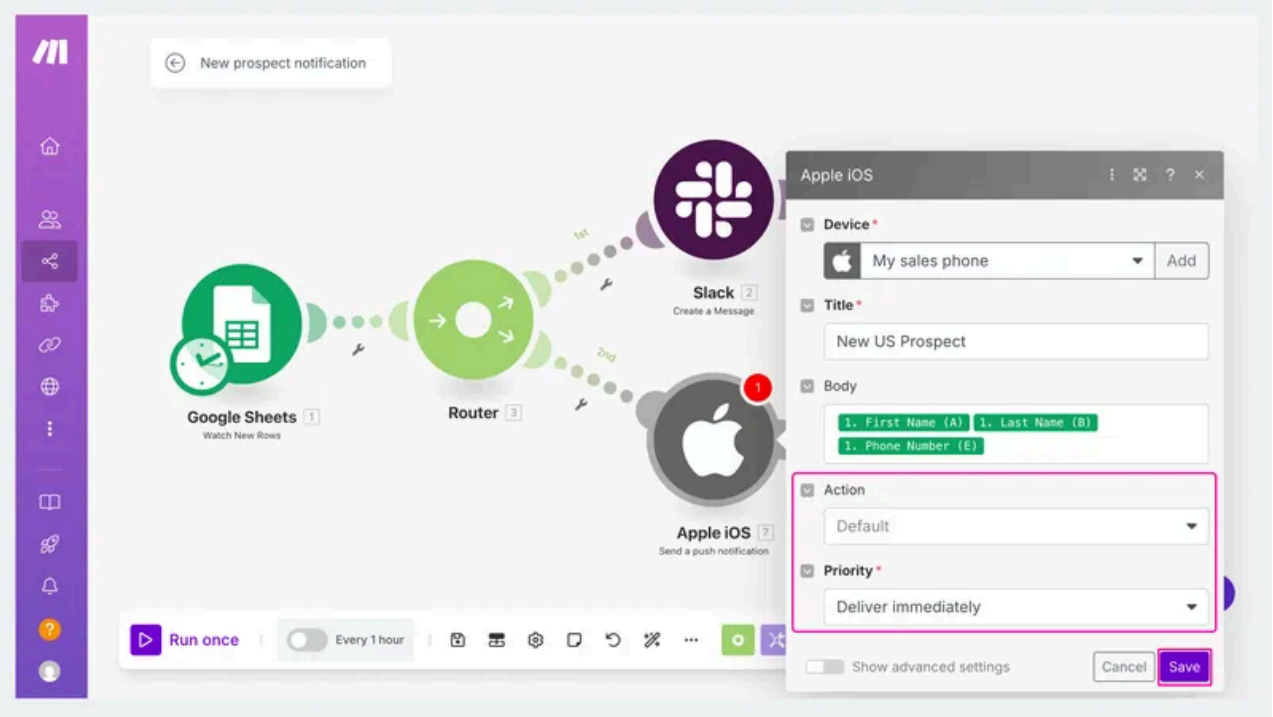


5 In the **Body** field, drag and drop the values (First Name, Last Name, Phone Number) from the Google Sheets module, or simply click on the values to place them into the field.



6 Keep **Action** as the default value and **Priority** as "Deliver immediately".

7 Click **Save** to save the module settings.



> Remember to click the **Save** icon in the Scenario Builder toolbar to save your entire scenario.

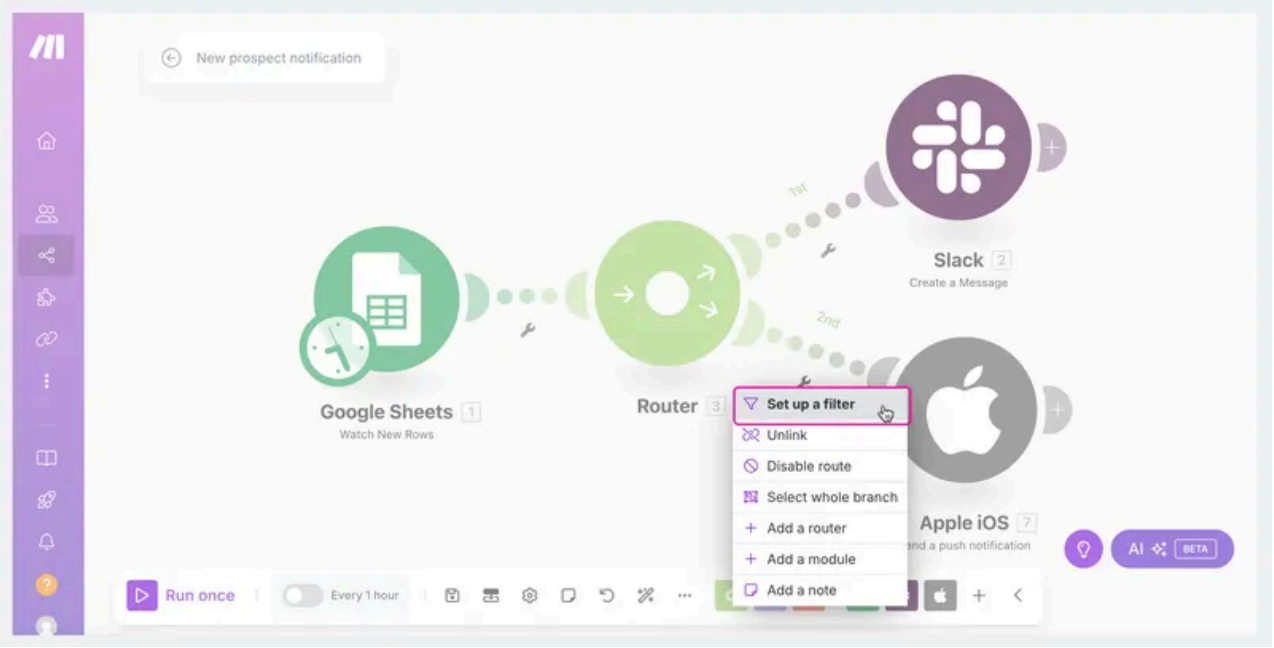
At this point, your scenario would send mobile notifications for all prospects, not just those from the USA. Next, you'll add a filter to this route to make the notifications country-specific.

1.3.4. Step 4. Add a filter

Filters allow you to control when a module runs, or what data is passed to a module based on specific conditions. In this case, you want to send mobile notifications only for prospects from the USA.

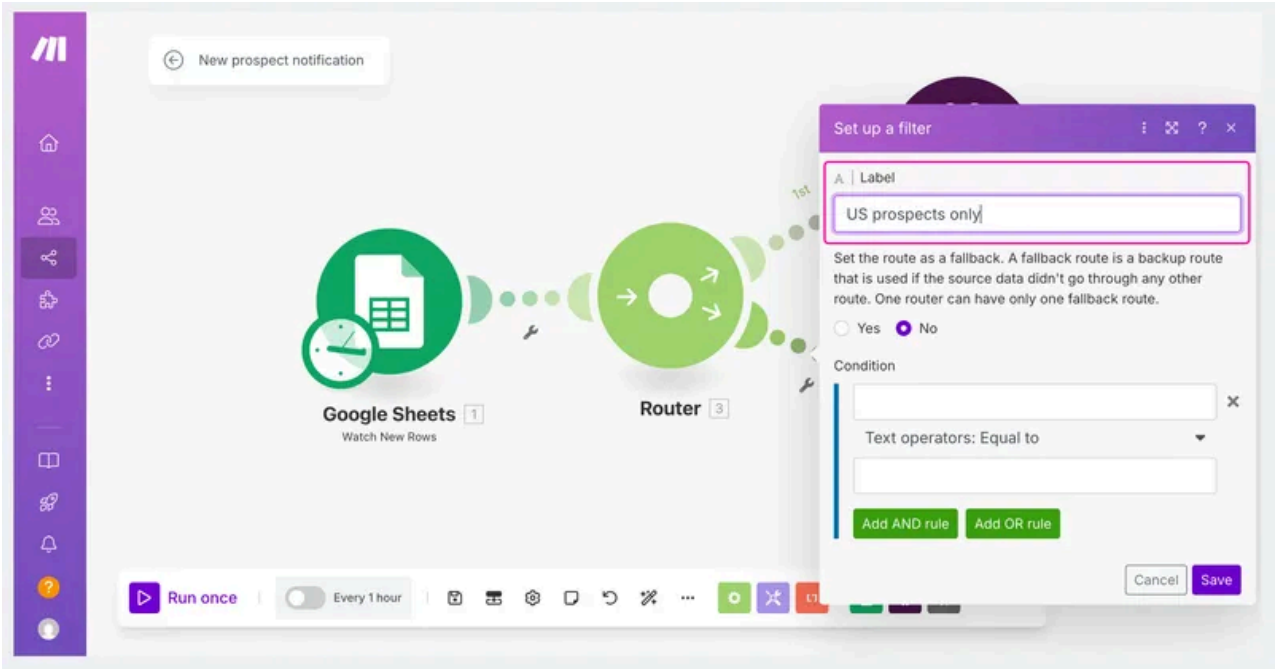
To add a filter:

1. Click on the wrench icon on the dots between the router and the Apple iOS module, and select **Set up a filter**.



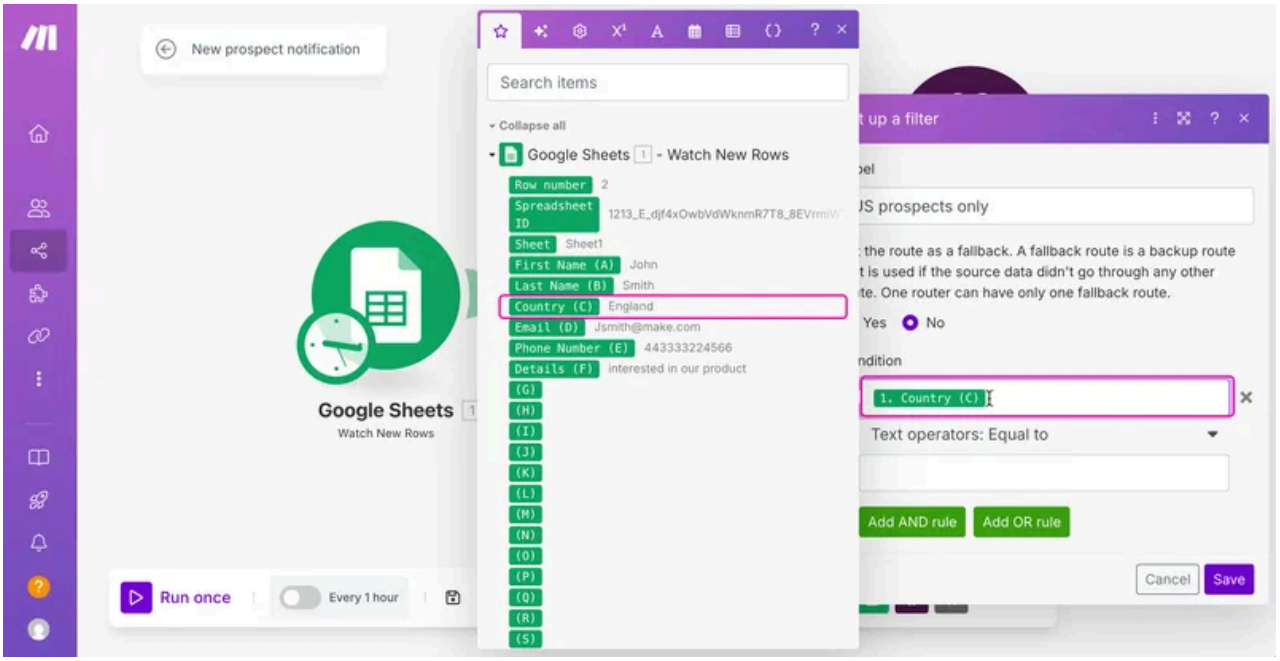
> Alternatively, you can right-click on the dots between modules and select **Set up a filter** or left-click directly on the dots between modules, which will open the **Set up a filter** window.

2. In the **Label** field, enter "US prospects only" as a label for the filter.



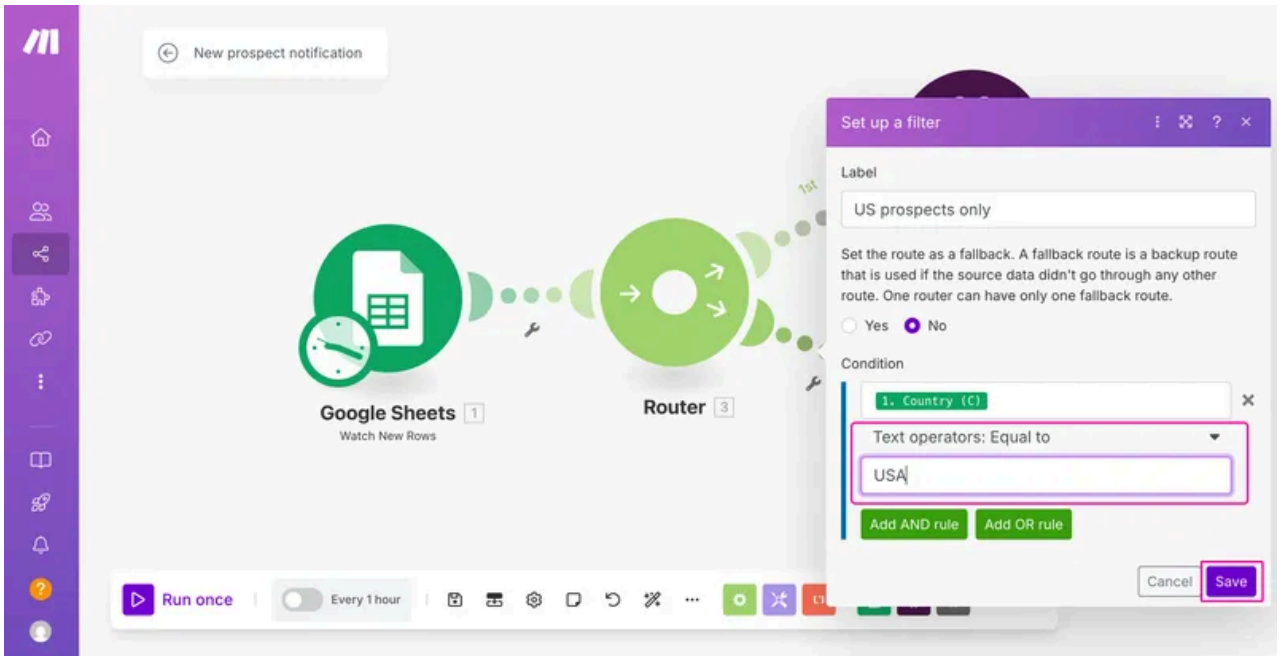
> The label of the filter is important because it appears in the Scenario Builder, making it easier to understand the flow of data through your scenario.

3 In the **Condition** field, drag and drop or click on the value (Country) from Google Sheets.



4 Keep the **Text Operators** dropdown as "Equal to" and enter "USA" in the field below.

5 Click **Save.**

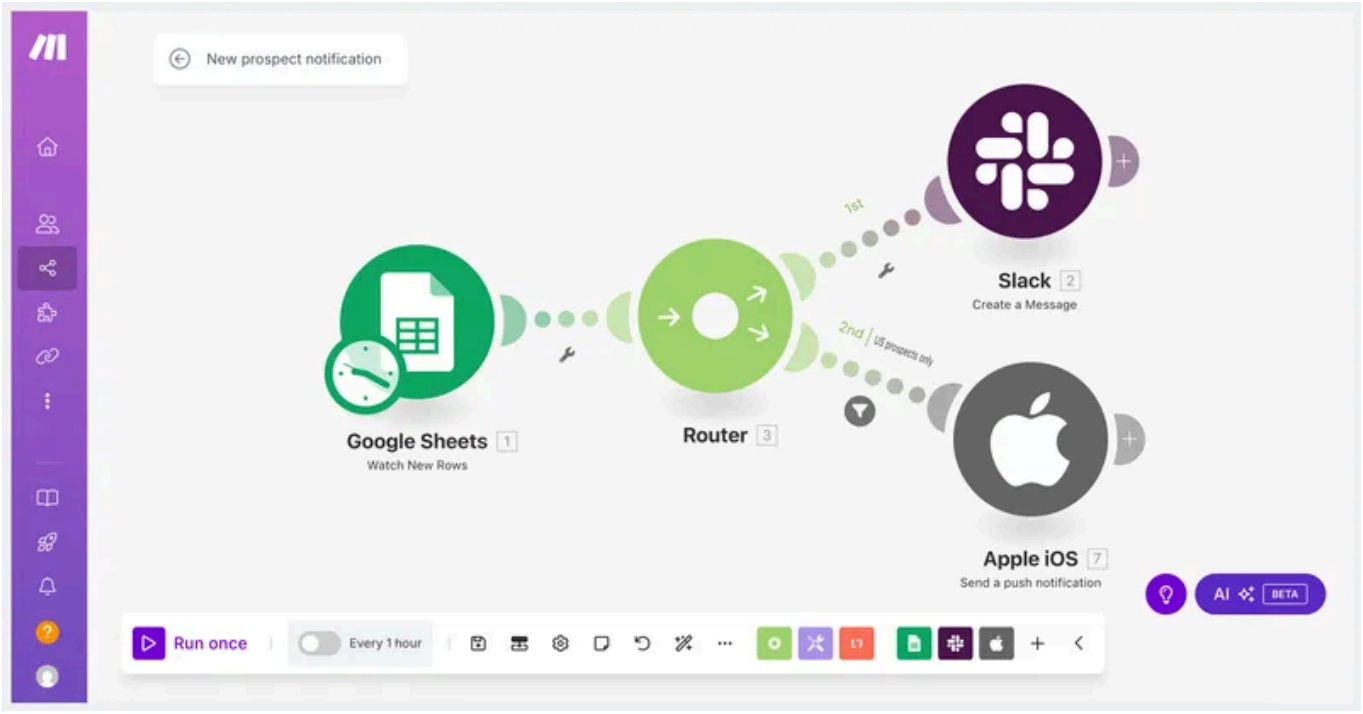


> Remember to click the **Save** icon in the Scenario Builder toolbar to save your entire scenario.

Now mobile notifications will only be sent when a prospect's country is "USA". The sales team will still receive all Slack notifications through the other route.

> **Understanding filters**
>
> Filters are tools that allow specific data to be passed or restricted within the flow of your scenario. Think of them as gatekeepers that only let through data that meets your criteria.
>
> Filters are useful to:
>
> - Send data down specific paths only when it meets your conditions.
> - Check if data matches specific criteria (equals, contains, greater than, etc.).
> - Build complex rules by combining multiple filters together.
> - Save operations because they stop the flow when your conditions aren't met.

Your scenario is now set up to send targeted notifications.



Next, you'll test it to make sure it works as expected with different types of prospects.

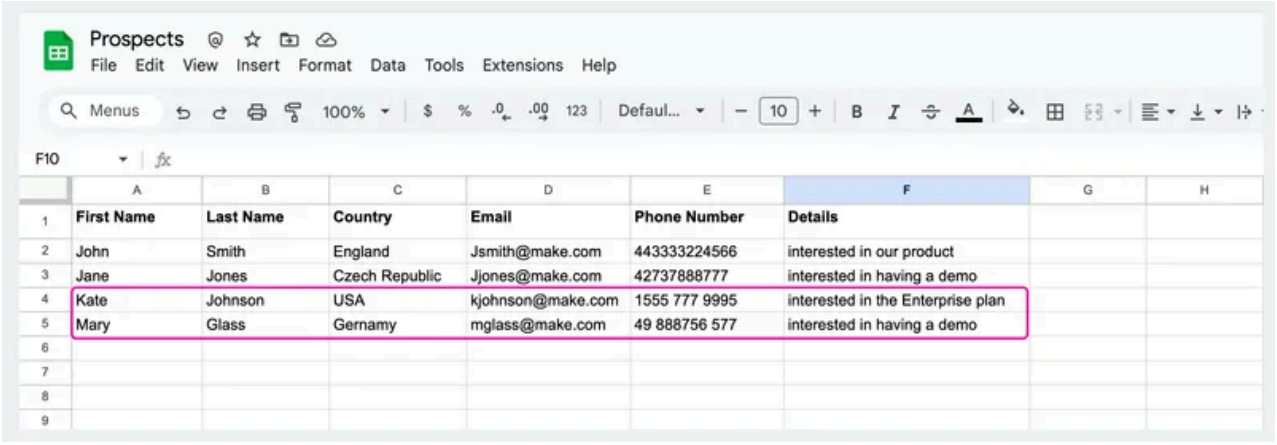
1.3.5. Step 5. Test your scenario

Now that you've set up the filter, it's time to test your scenario with different types of prospects.

To test your scenario:

1. Go to your Prospects spreadsheet and add two new rows:

    - One with country = USA
    - Another with a different country



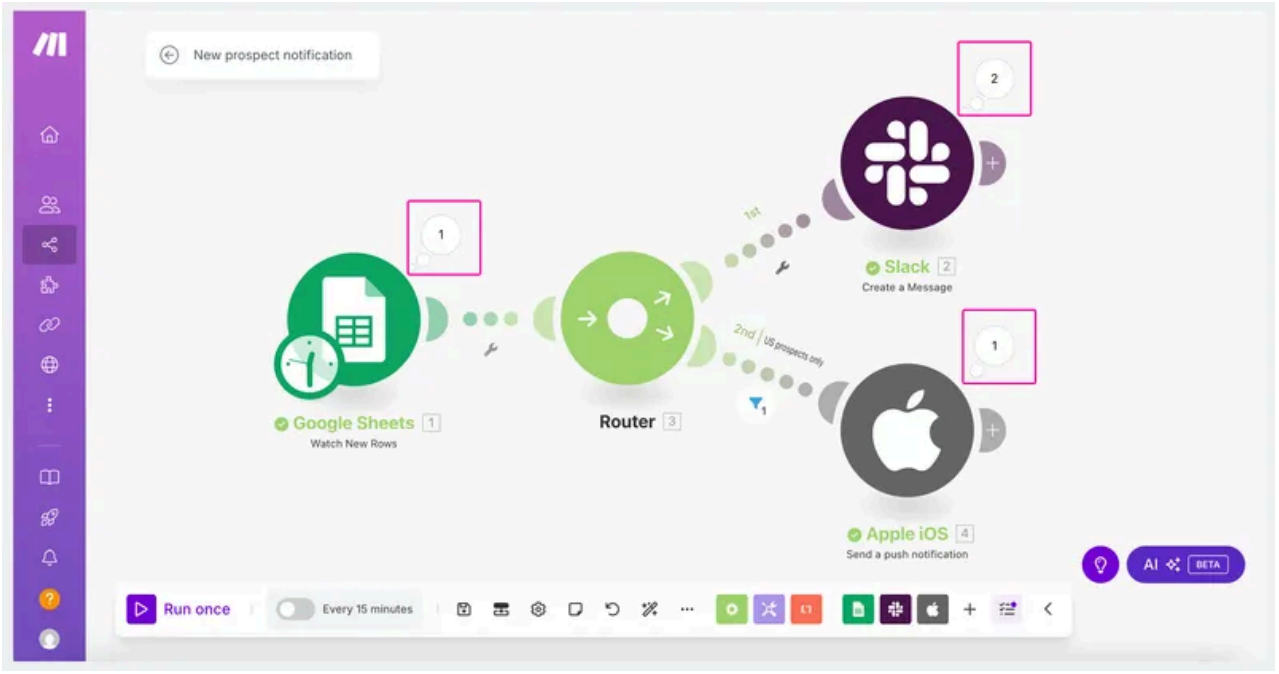
	First Name	Last Name	Country	Email	Phone Number	Details
2	John	Smith	England	Jsmith@make.com	443333224566	interested in our product
3	Jane	Jones	Czech Republic	Jjones@make.com	42737888777	interested in having a demo
4	Kate	Johnson	USA	kjohnson@make.com	1555 777 9995	interested in the Enterprise plan
5	Mary	Glass	Gernamy	mglass@make.com	49 888756 577	interested in having a demo

2. In the Scenario Builder toolbar, click the **Run once** button.



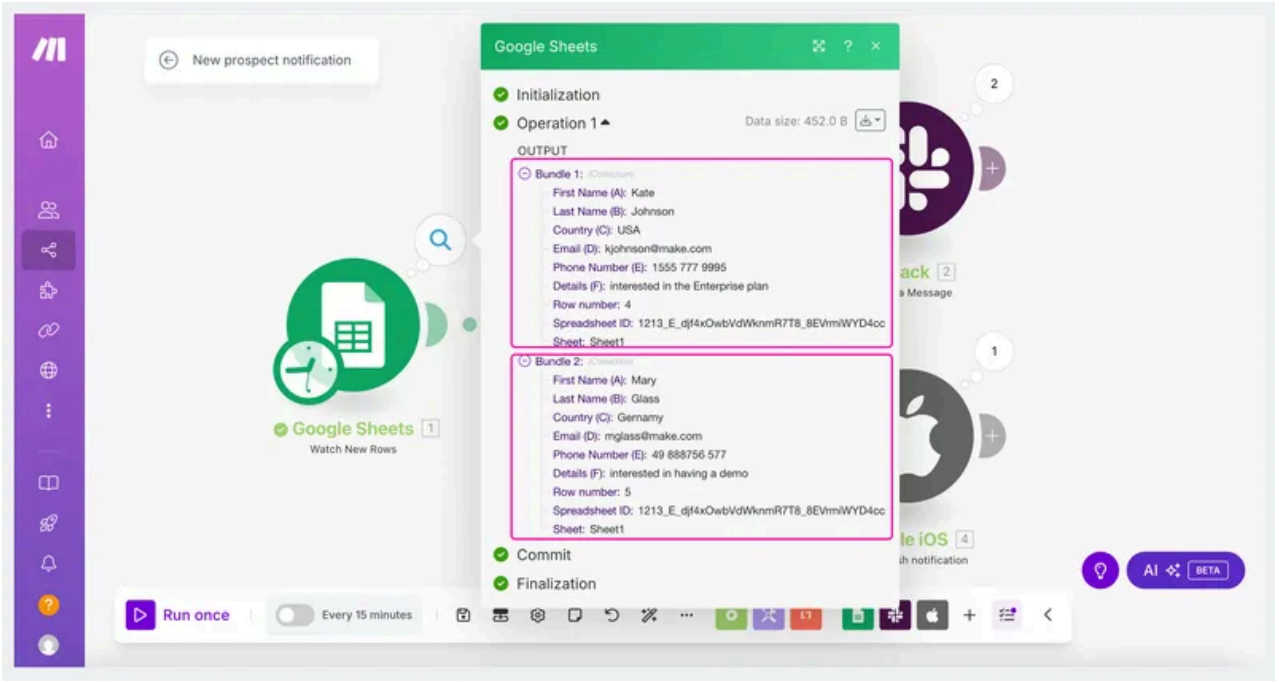
> By default, the Google Sheets "Watch New Rows" module will only find the new rows added since you last ran your scenario.
>
> If you want to process all rows again for testing, you can reset it: right-click the Google Sheets module and select **Choose where to start**. Select "All" and click **Save**. The next time you run the scenario, it will include all the rows again.

3 Check the operation bubbles above each module to see the results. The numbers shown in the bubbles indicate how many operations were used.

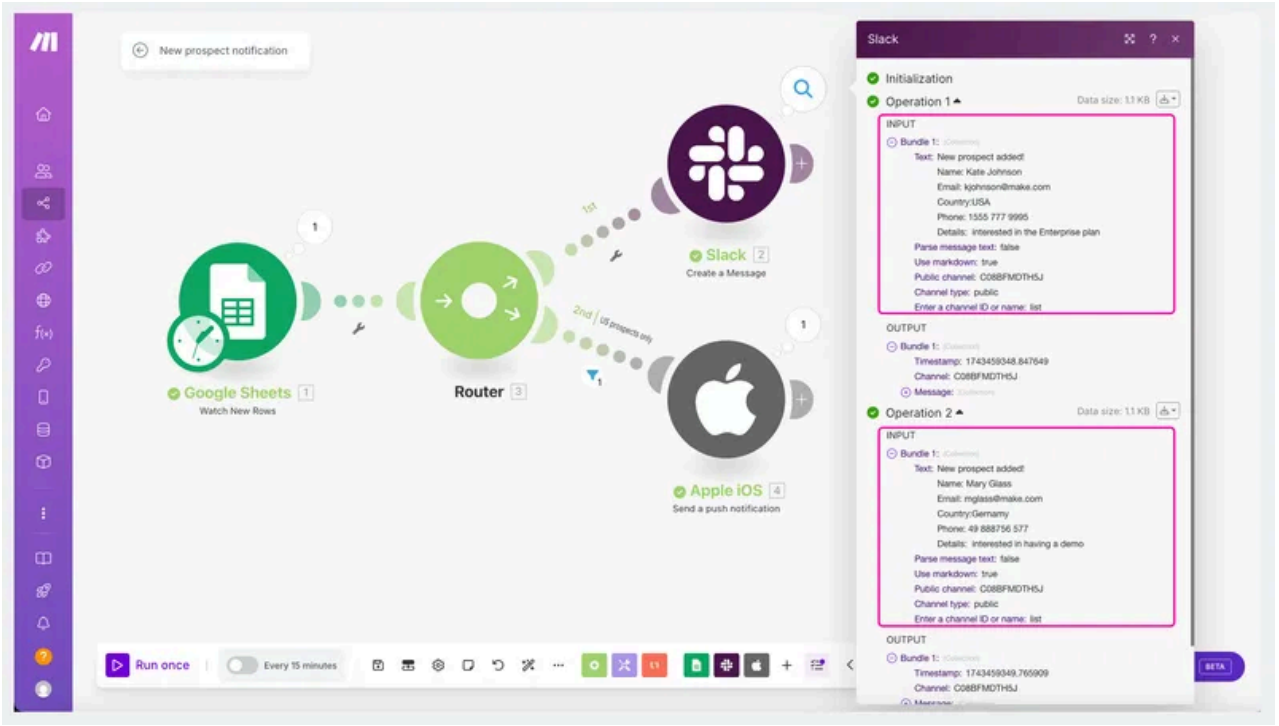


4 Click on the operation bubbles to view the details.

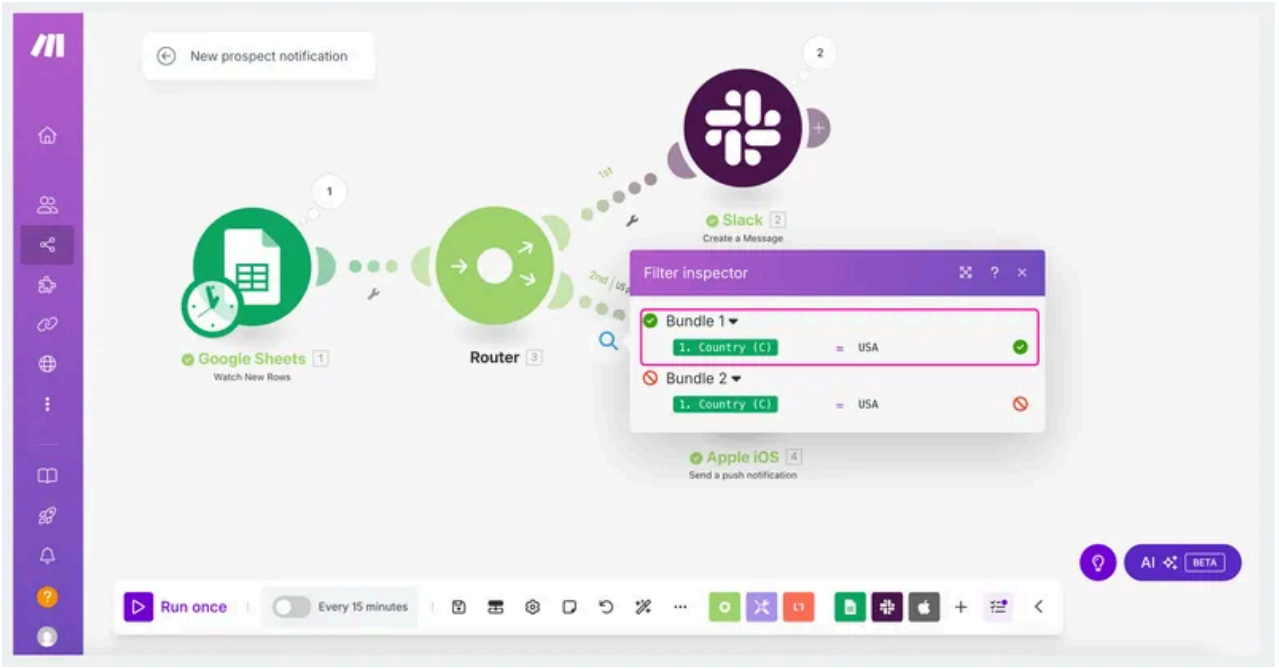
The Google Sheets module should show both prospects:



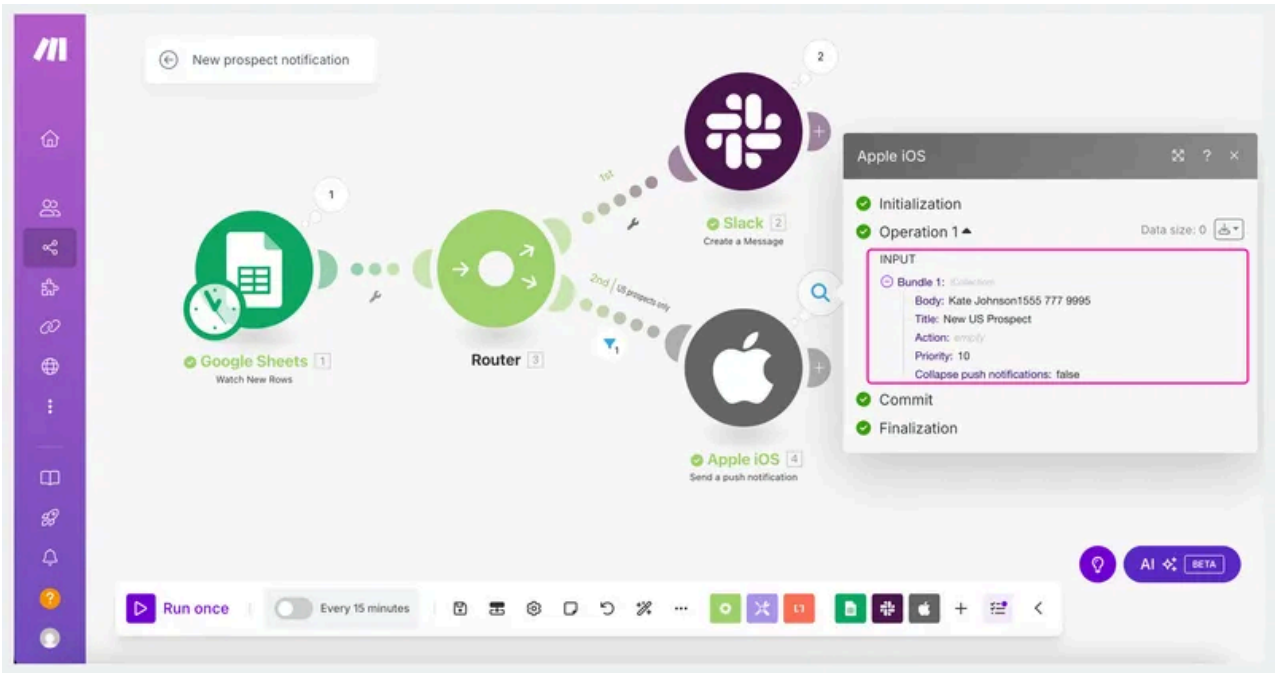
The Slack module should receive and process both prospects:



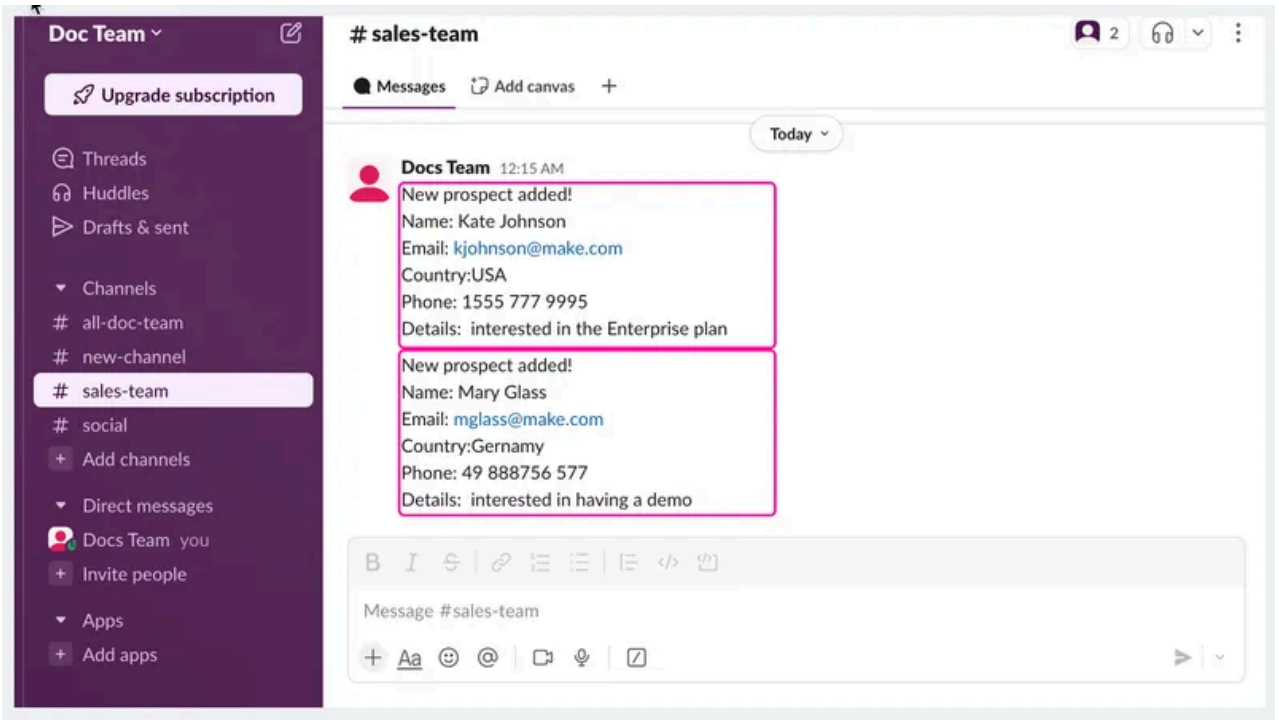
The filter should only let the USA prospect through:



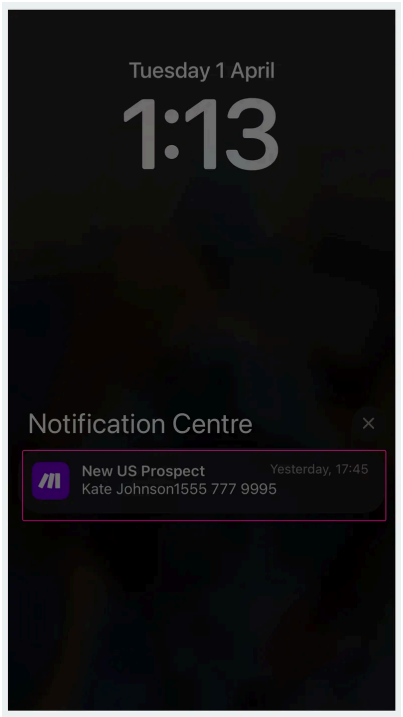
The Apple iOS module should only process the USA prospect:



5 Check your Slack channel to confirm both notifications arrived.



6 Check your mobile device to confirm you only received a notification for the USA prospect.



This test shows that the scenario works properly:

- All prospects are sent as Slack notifications

- Only US prospects trigger mobile notifications

This targeted approach ensures the Sales Manager gets alerts only for US market prospects, while the Slack channel receives updates about all prospects.

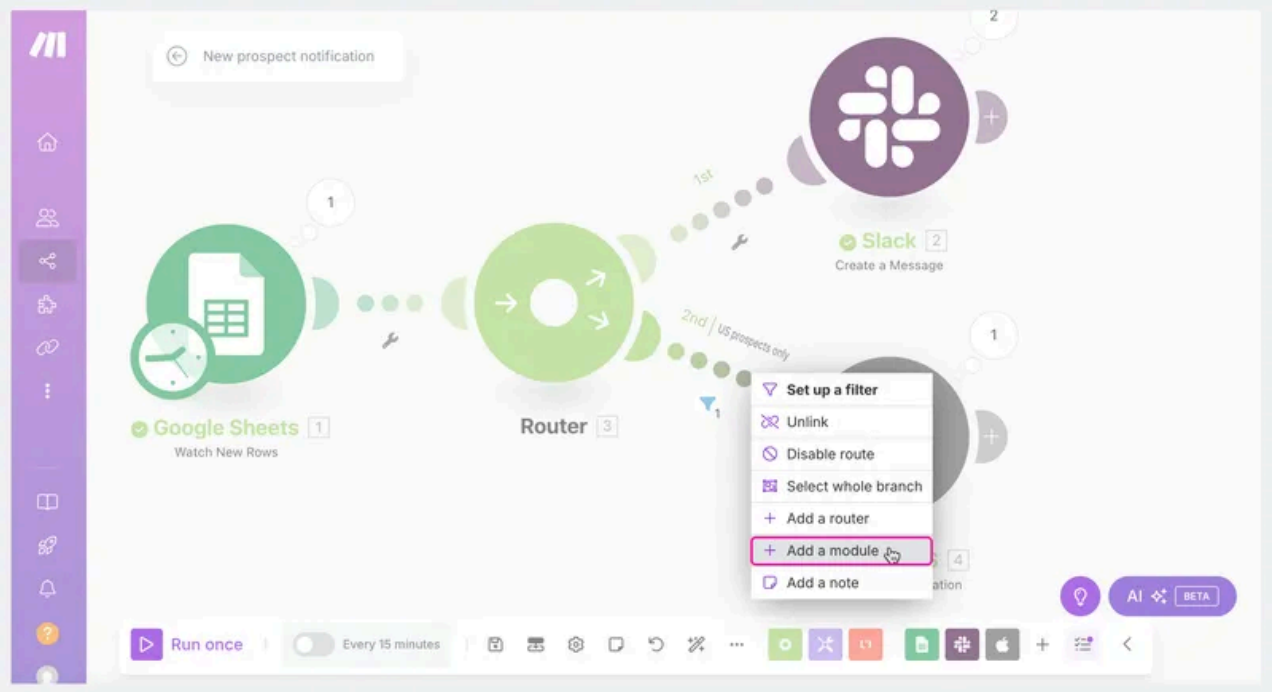
However, if many US prospects are added at once, the Sales Manager would receive multiple separate notifications. In the next step, you'll add an aggregator to combine multiple US prospects into a single notification.

1.3.6. Step 6. Add an aggregator

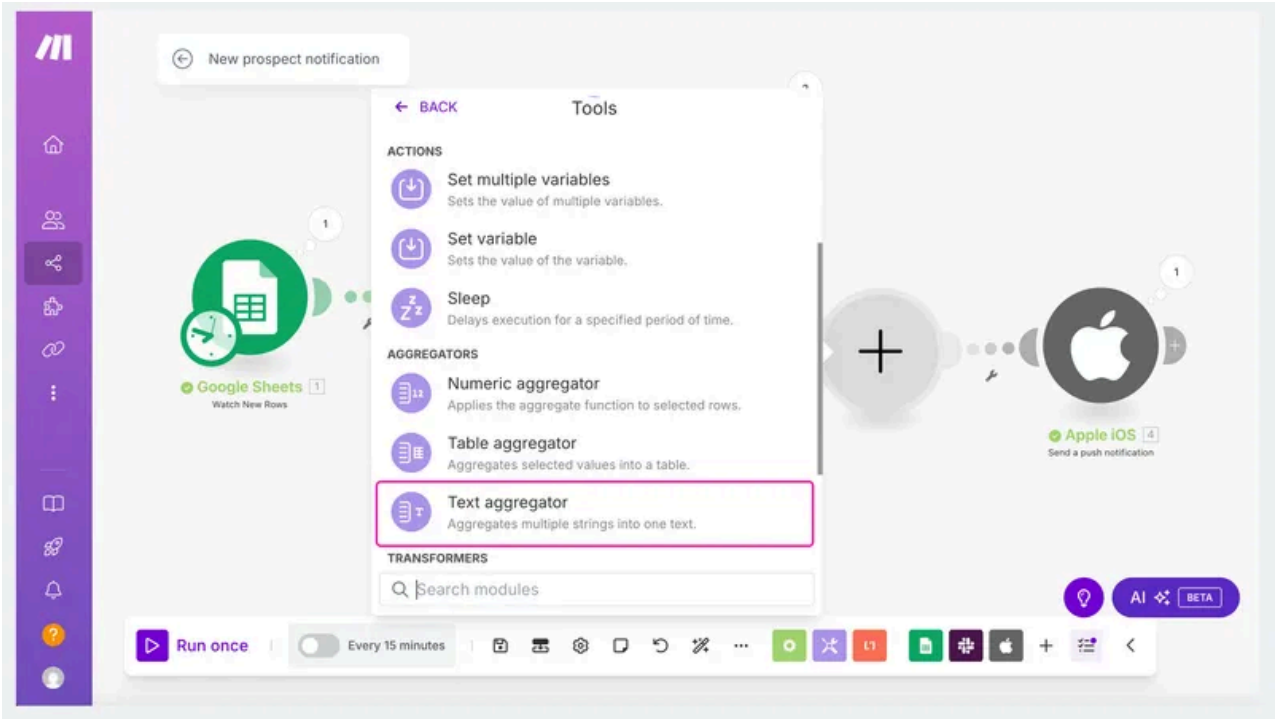
Imagine your marketing team adds multiple US prospects in a single day, your Sales Manager could quickly become overwhelmed with a flood of individual mobile notifications. By adding an aggregator to your scenario, you can combine all those prospects into a single notification - making the information more manageable while still ensuring timely alerts.

To add an aggregator:

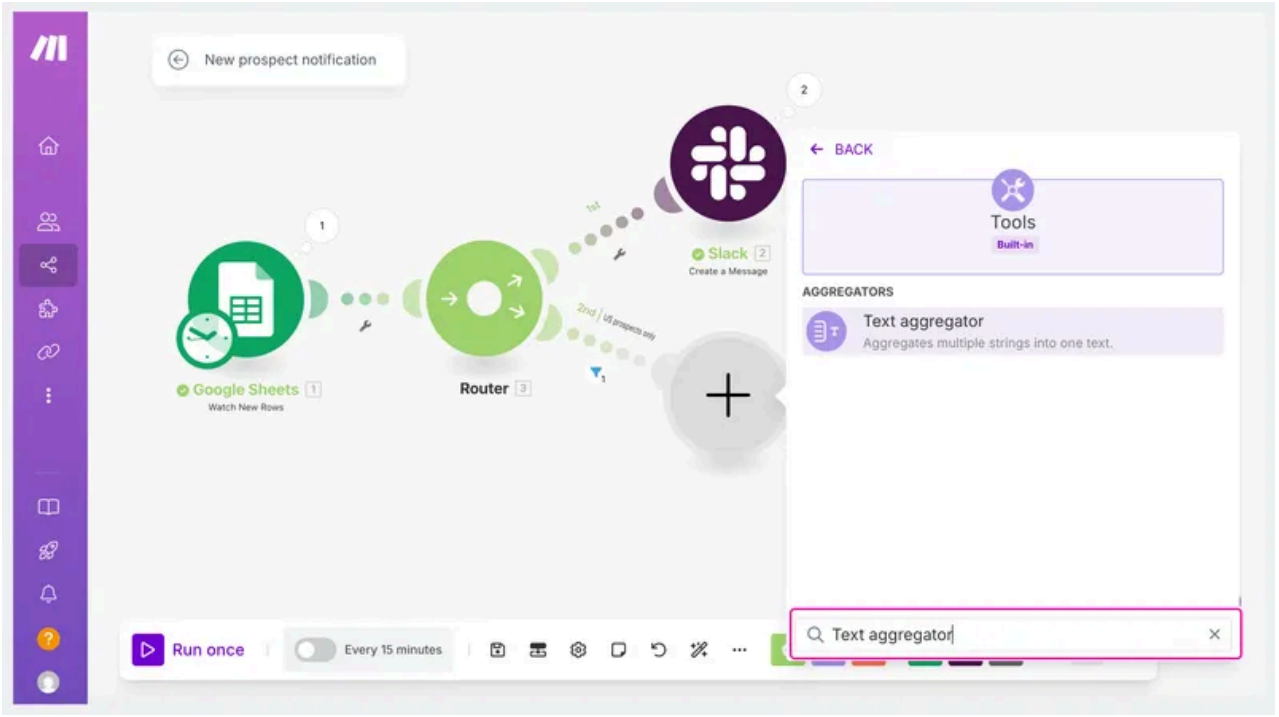
1. Right-click on the dots between the filter and the Apple iOS module and select **Add a module**. This will open a window with the list of apps and modules that you can add to your scenario.



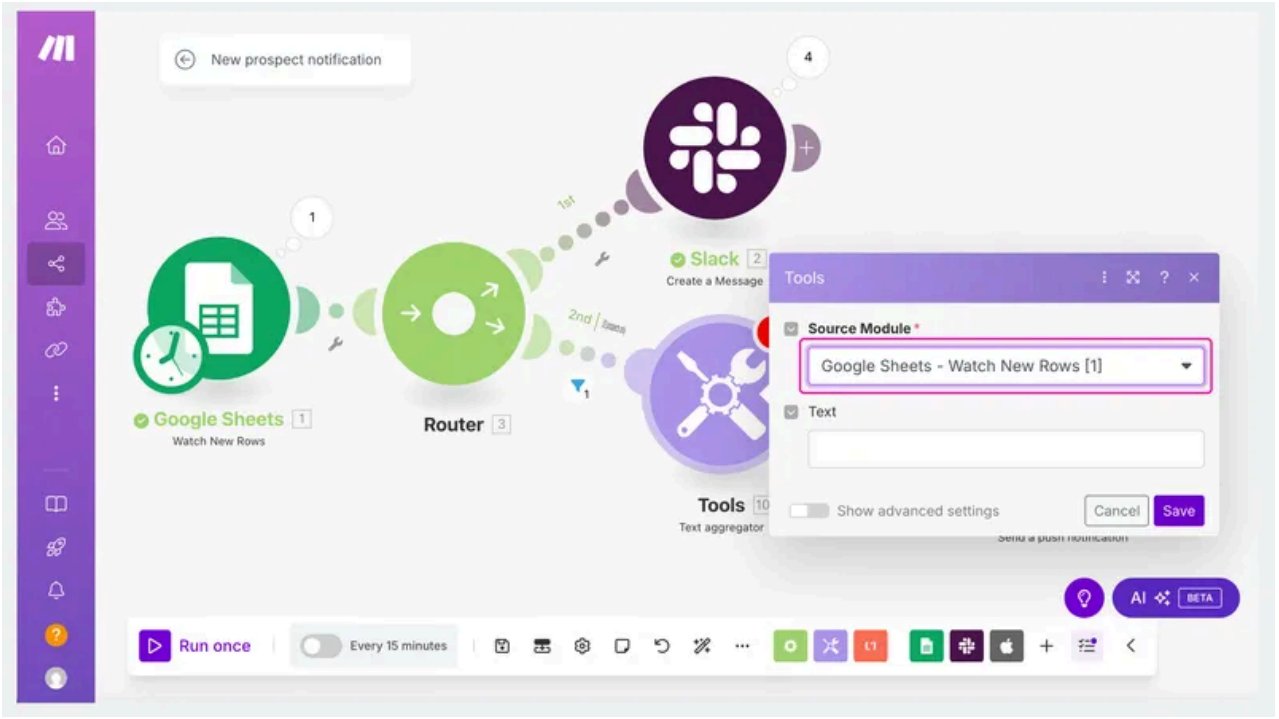
2. Search for "Tools" in the search bar and then choose "Text Aggregator" from the available modules.



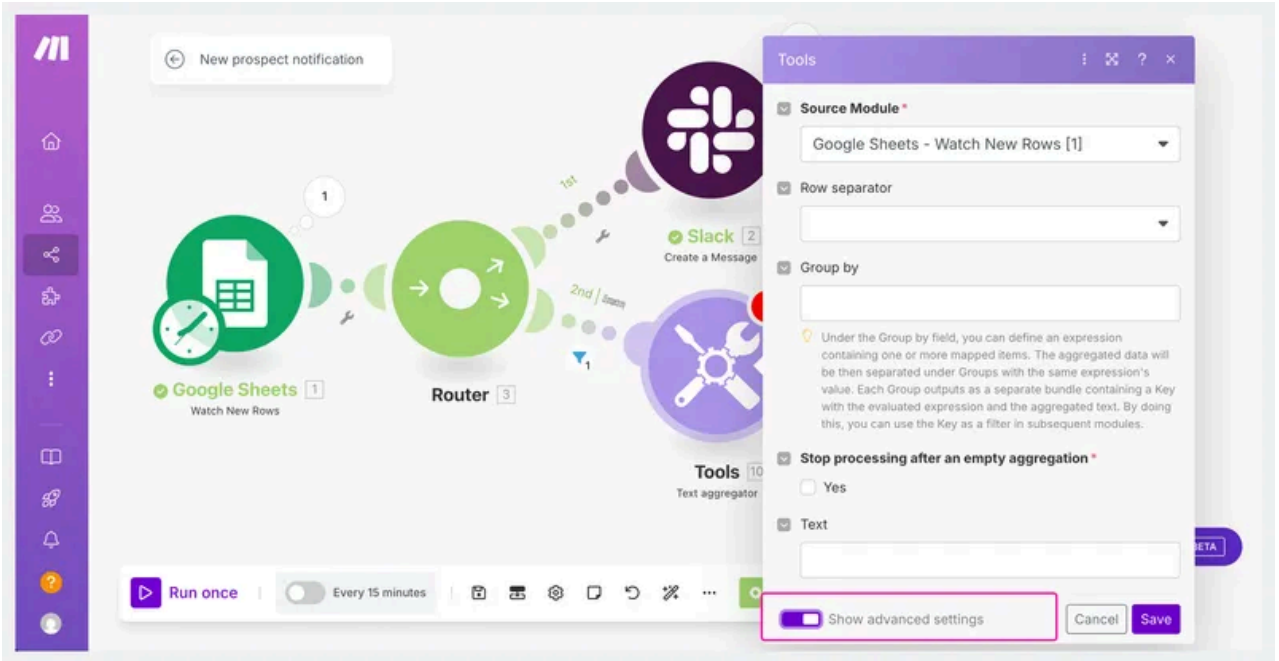
Alternatively, you can enter "Text Aggregator" in the search bar to find it faster.



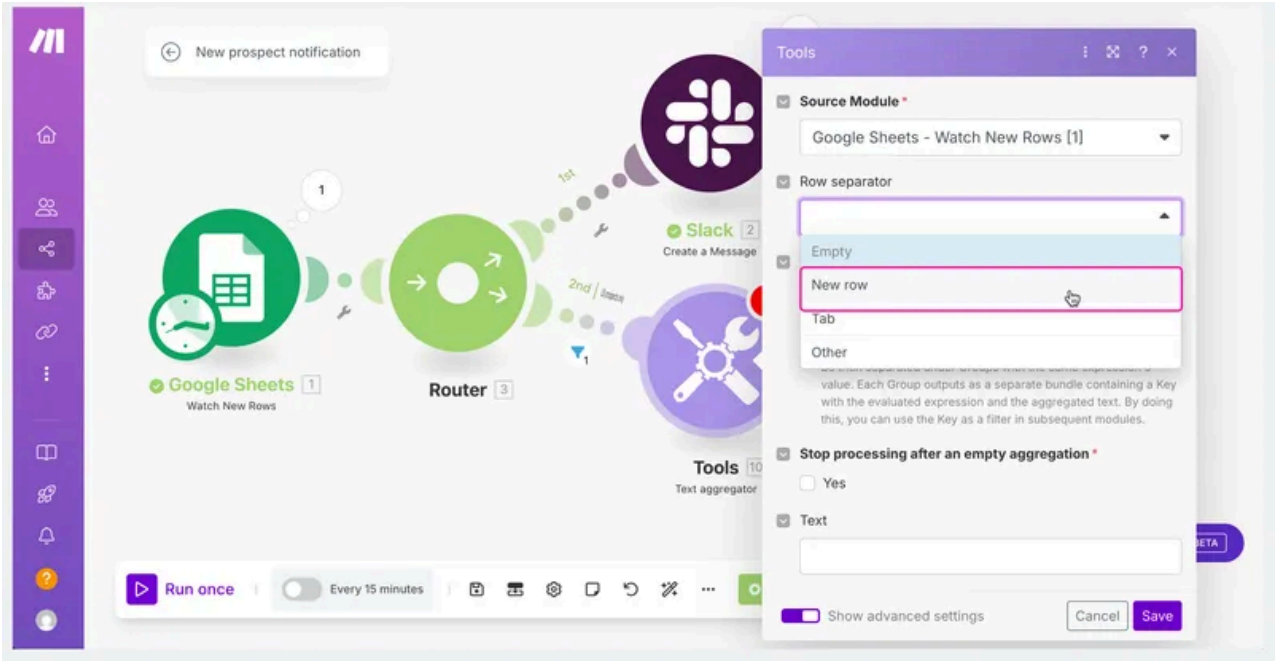
3 In the **Tools** settings, select "Google Sheets - Watch New Rows (1)" from the dropdown list under the **Source Module** field.



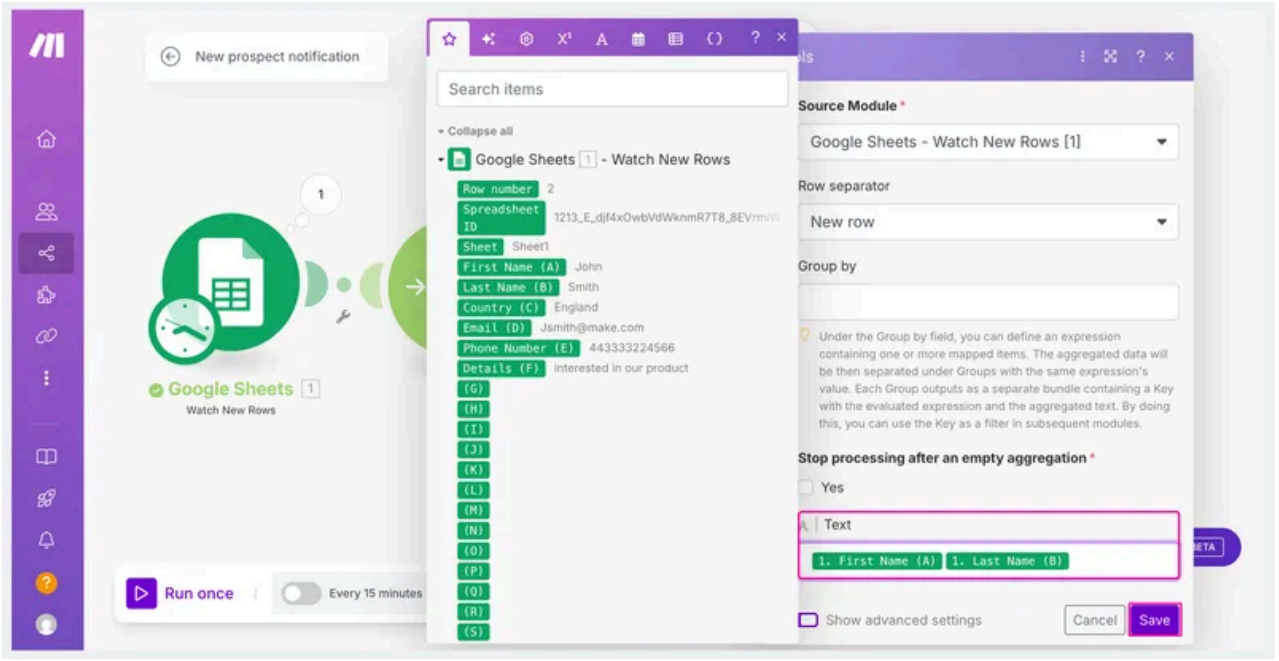
4 Switch on the **Show advanced settings** toggle at the bottom of the window to open additional options.



5 In the **Row separator** field, select "New row" from the dropdown.

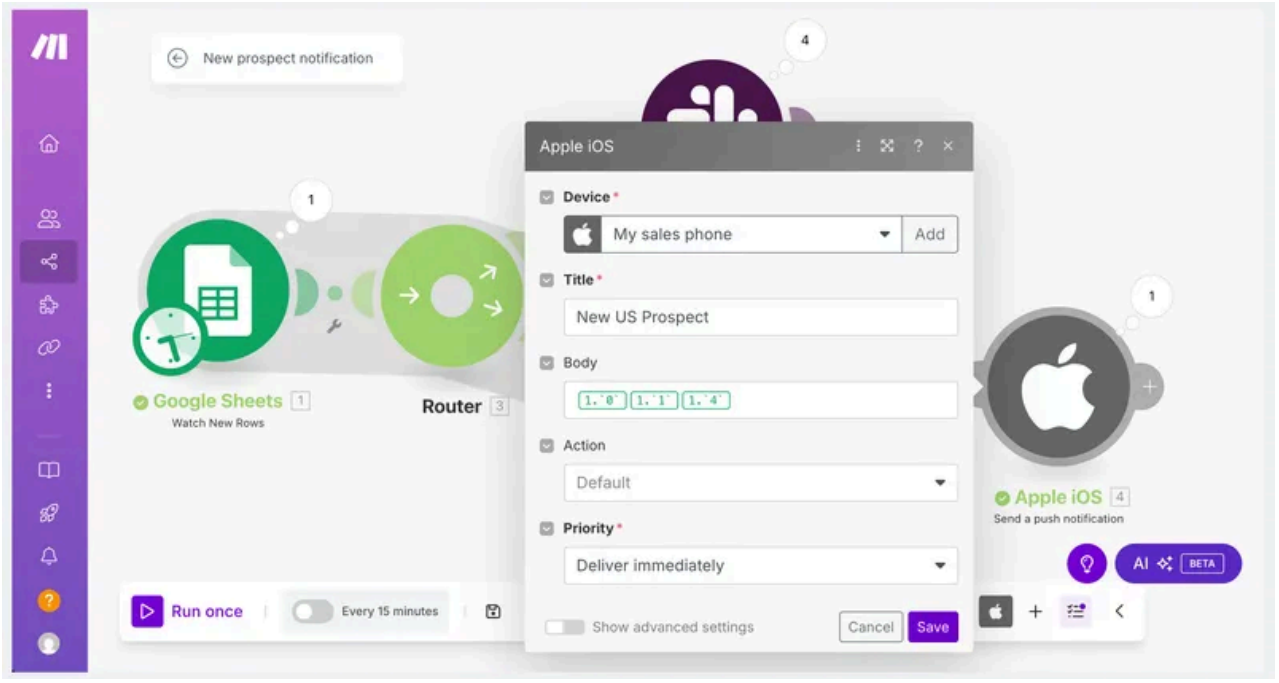


6 In the **Text** field, drag and drop or click on (First Name) and (Last Name) values from the Google Sheets module.

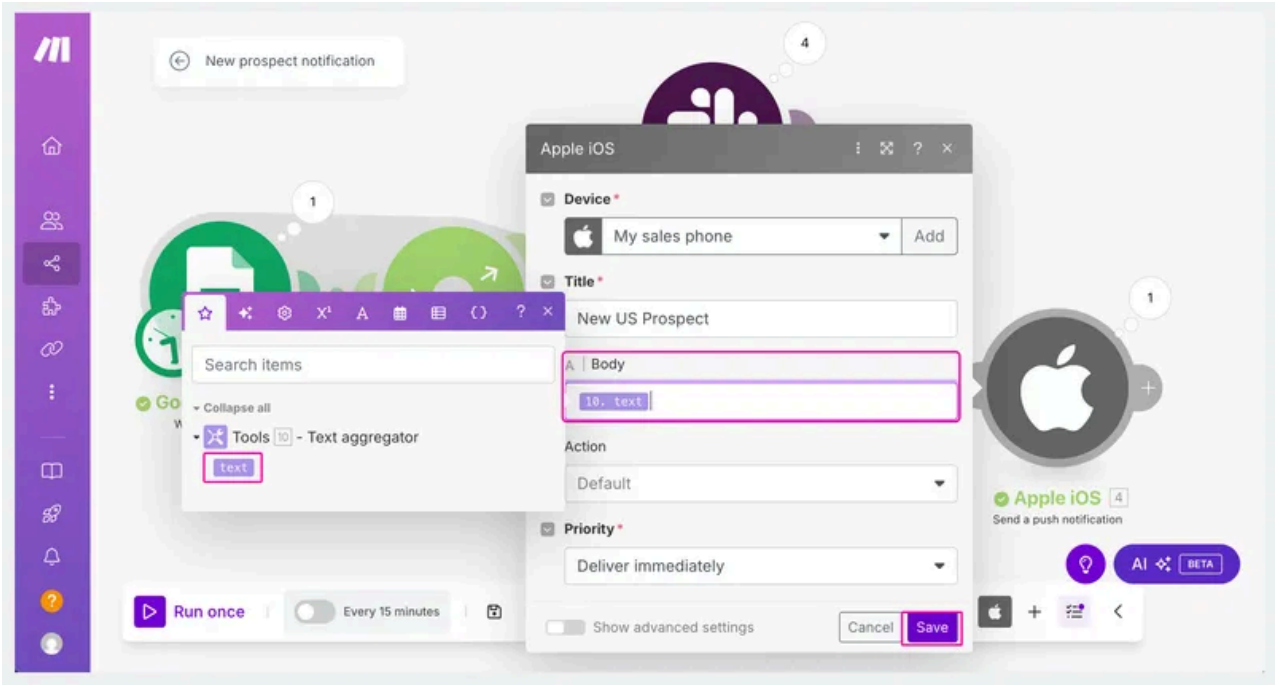


7 Click **Save** to save the aggregator settings.

8 Next, you need to update the Apple iOS "Send a push notification" module to use the aggregated text. Click on the Apple iOS module to open its settings.

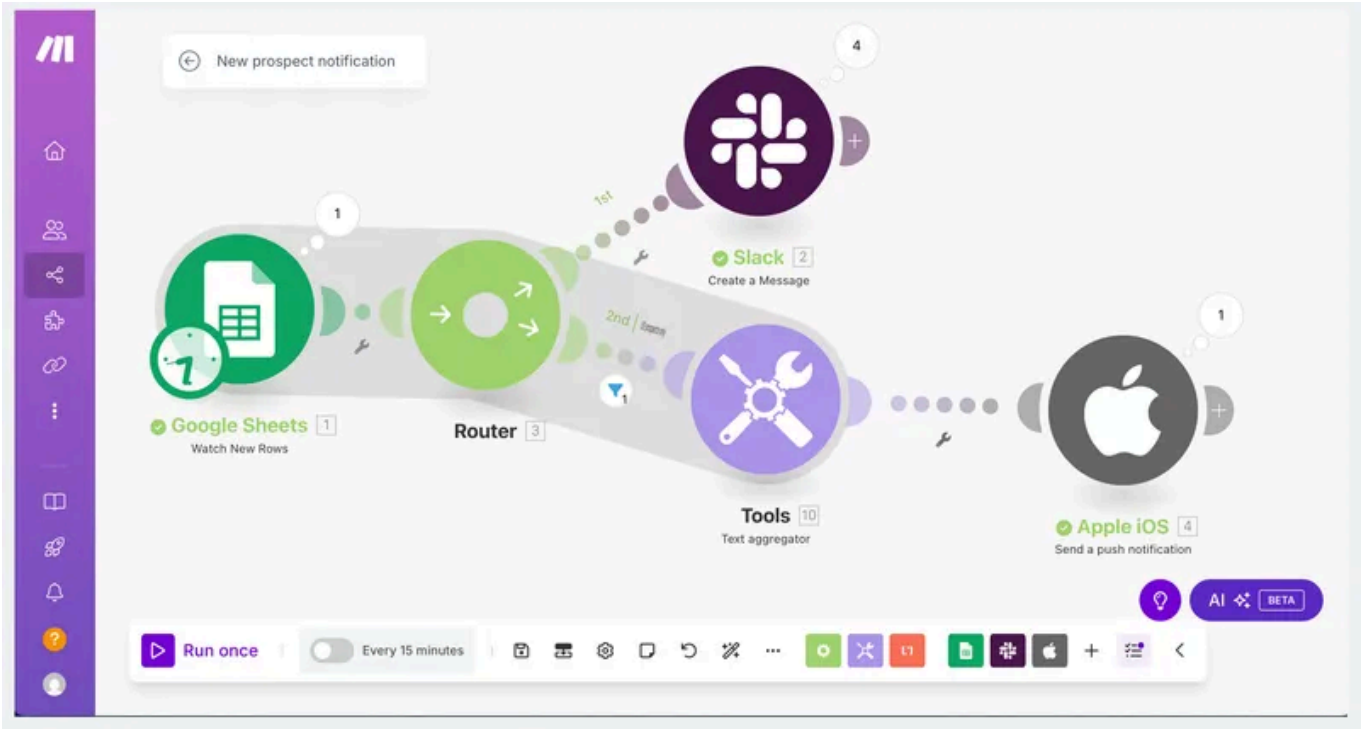


9 In the **Body** field, remove any existing mapped values and instead drag and drop or click on the (text) value.



10 Click **Save** to save the changes and make sure to save your entire scenario by clicking the **Save** icon in the Scenario Builder toolbar.

Your scenario now has an aggregator that combines new US prospects that arrive into a single notification. In the next step, you'll test this entire scenario.



Understanding aggregators

Aggregators combine multiple bundles of data into a single bundle. Without an aggregator, each prospect would trigger a separate notification. With an aggregator:

- Multiple prospects are combined into one notification
- The Sales Manager receives one comprehensive alert instead of many
- The scenario uses fewer operations

Make offers different types of aggregators:

- **Text aggregator**: Combines text with separators (the one we used in this scenario)
- **Numeric aggregator**: Performs calculations (sum, average, count)
- **Array aggregator**: Groups data into structured arrays

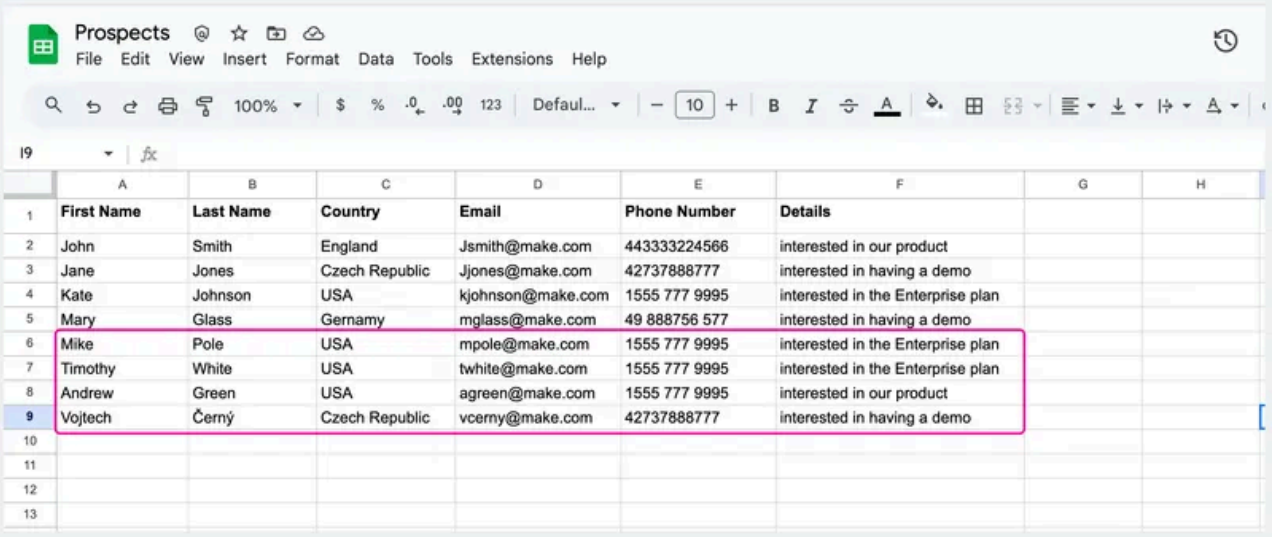
In our scenario, the text aggregator helps prevent notification overload by sending one organized list of all US prospects.

1.3.7. Step 7. Test the final scenario

Now that you've added an aggregator to your scenario, it's time to test the complete scenario to make sure everything works as expected.

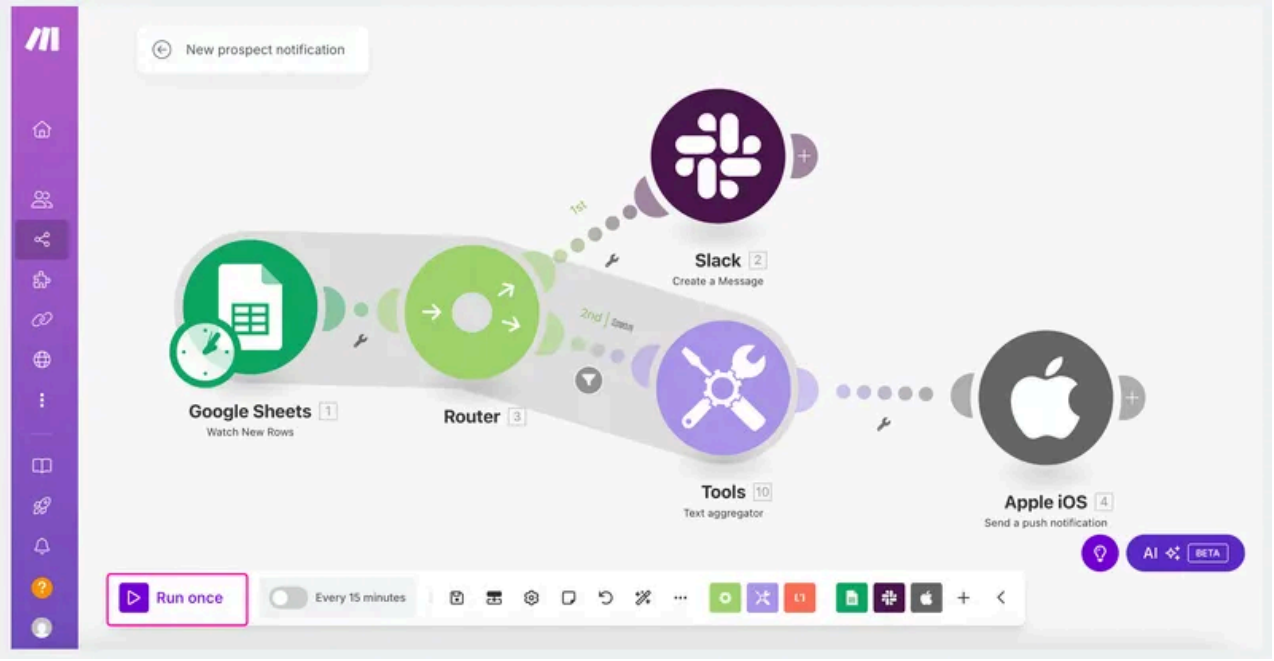
1. Go to your Prospects spreadsheet and add three or more new rows:

    - At least two with Country = USA
    - One with a different country



	First Name	Last Name	Country	Email	Phone Number	Details
2	John	Smith	England	Jsmith@make.com	443333224566	interested in our product
3	Jane	Jones	Czech Republic	Jjones@make.com	42737888777	interested in having a demo
4	Kate	Johnson	USA	kjohnson@make.com	1555 777 9995	interested in the Enterprise plan
5	Mary	Glass	Gernamy	mglass@make.com	49 888756 577	interested in having a demo
6	Mike	Pole	USA	mpole@make.com	1555 777 9995	interested in the Enterprise plan
7	Timothy	White	USA	twhite@make.com	1555 777 9995	interested in the Enterprise plan
8	Andrew	Green	USA	agreen@make.com	1555 777 9995	interested in our product
9	Vojtech	Černý	Czech Republic	vcerny@make.com	42737888777	interested in having a demo

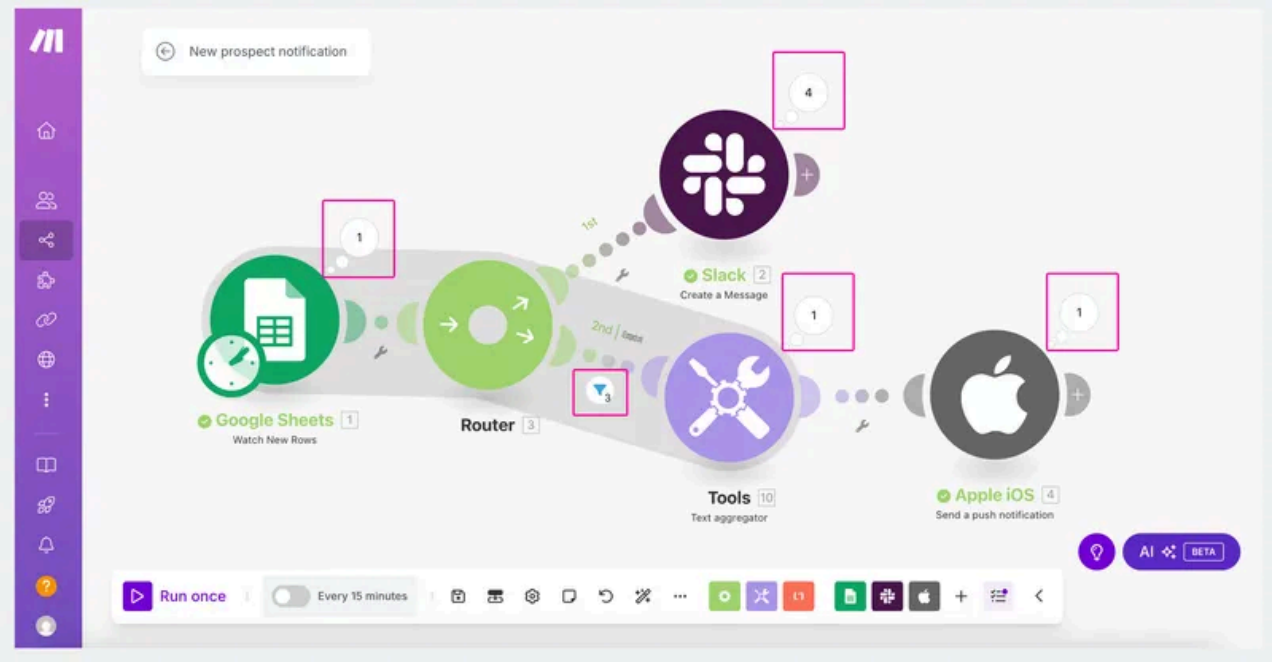
2. In the Scenario Builder toolbar, click the **Run once** button.



By default, the "Watch New Rows" module will only find the new rows added since you last ran your scenario. To process all rows again:

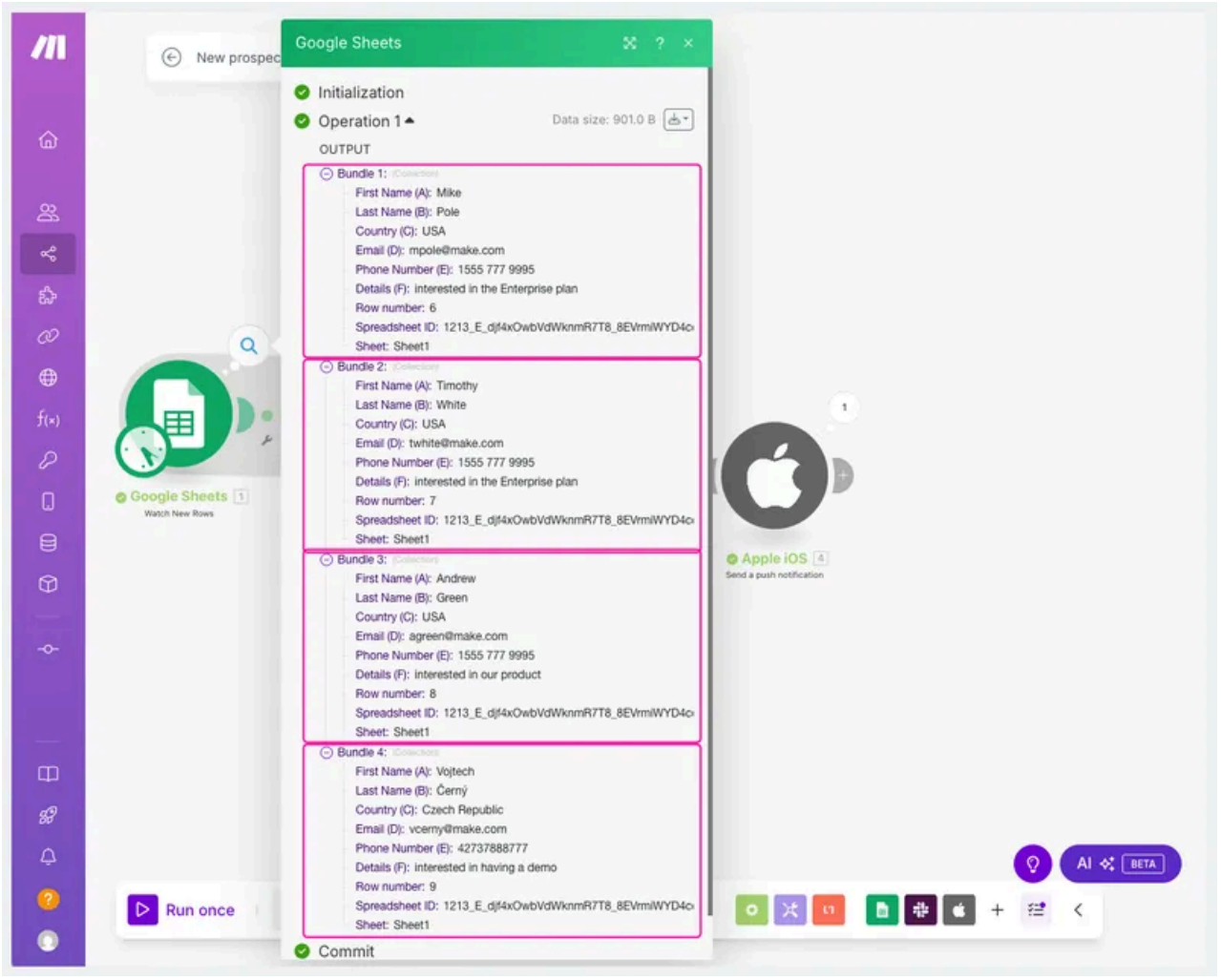
- Right-click the Google Sheets module
- Select **Choose where to start**
- Choose "All" and click **Save**

3 Check the operation bubbles above each module to see the results. The numbers shown in the bubbles indicate how many operations were used.

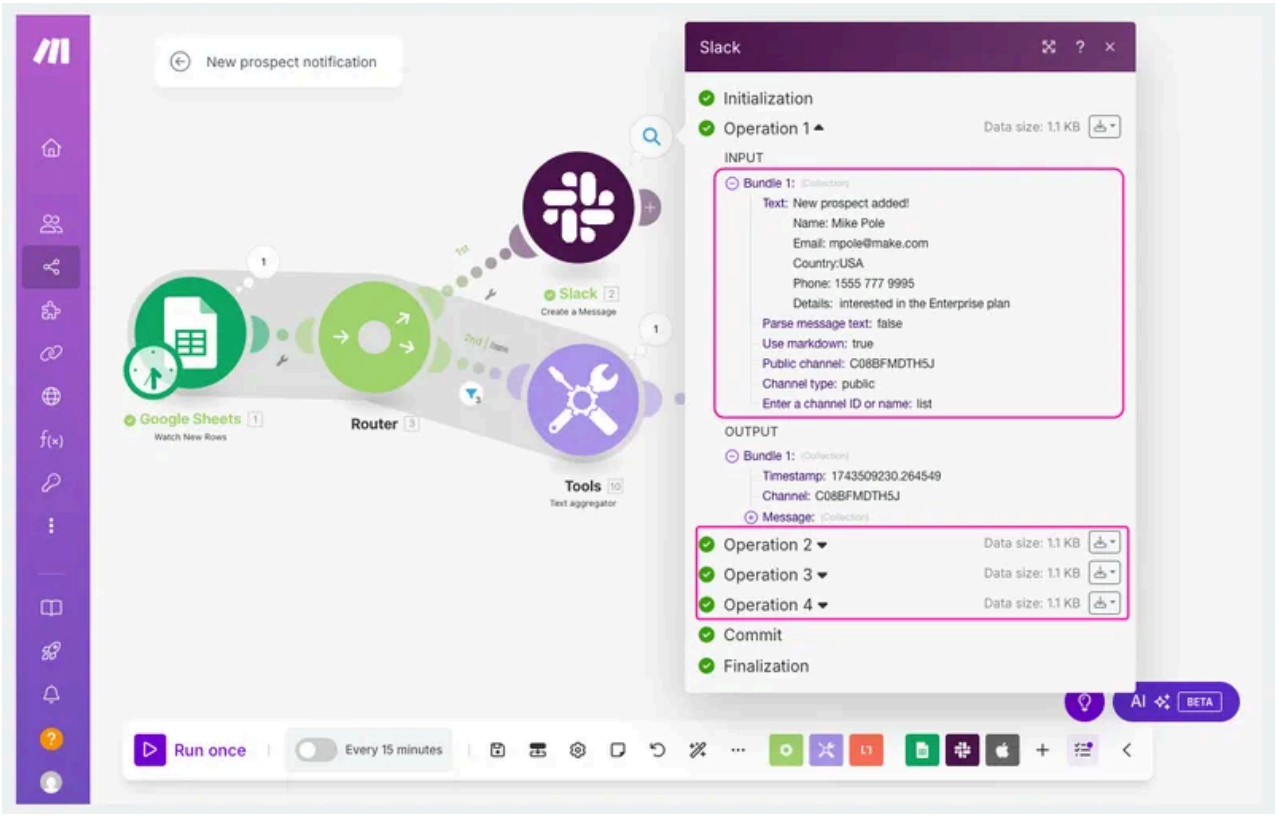


4 Click on the bubbles to view the details.

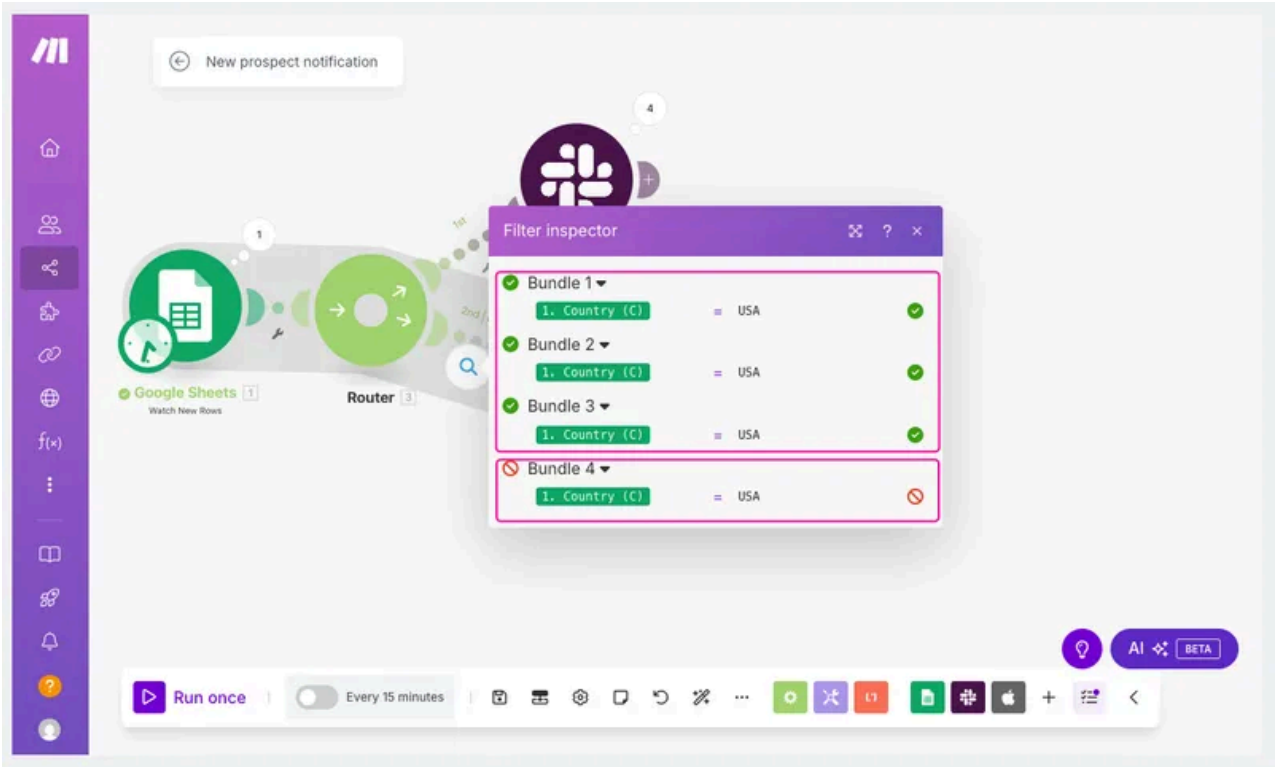
The Google Sheets module should show all new prospects you added, depending on the limit you set:



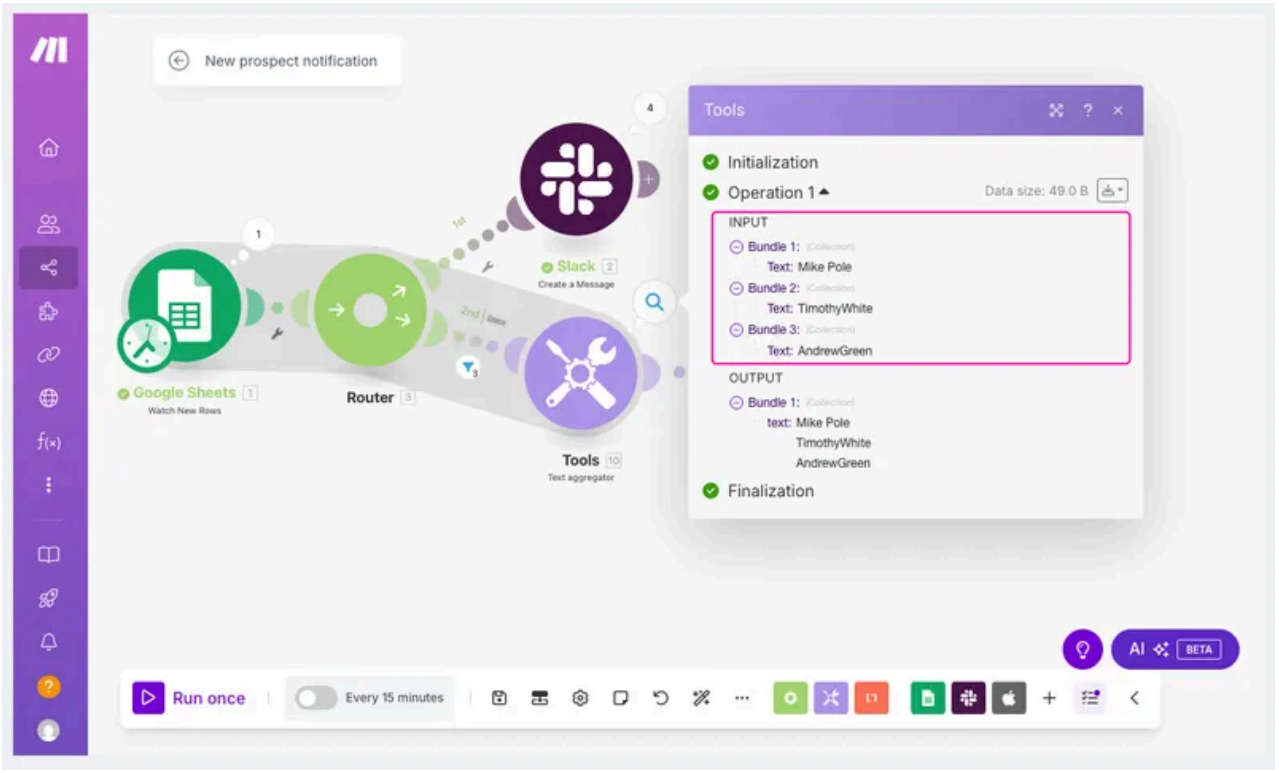
The Slack module should receive and process all prospects:



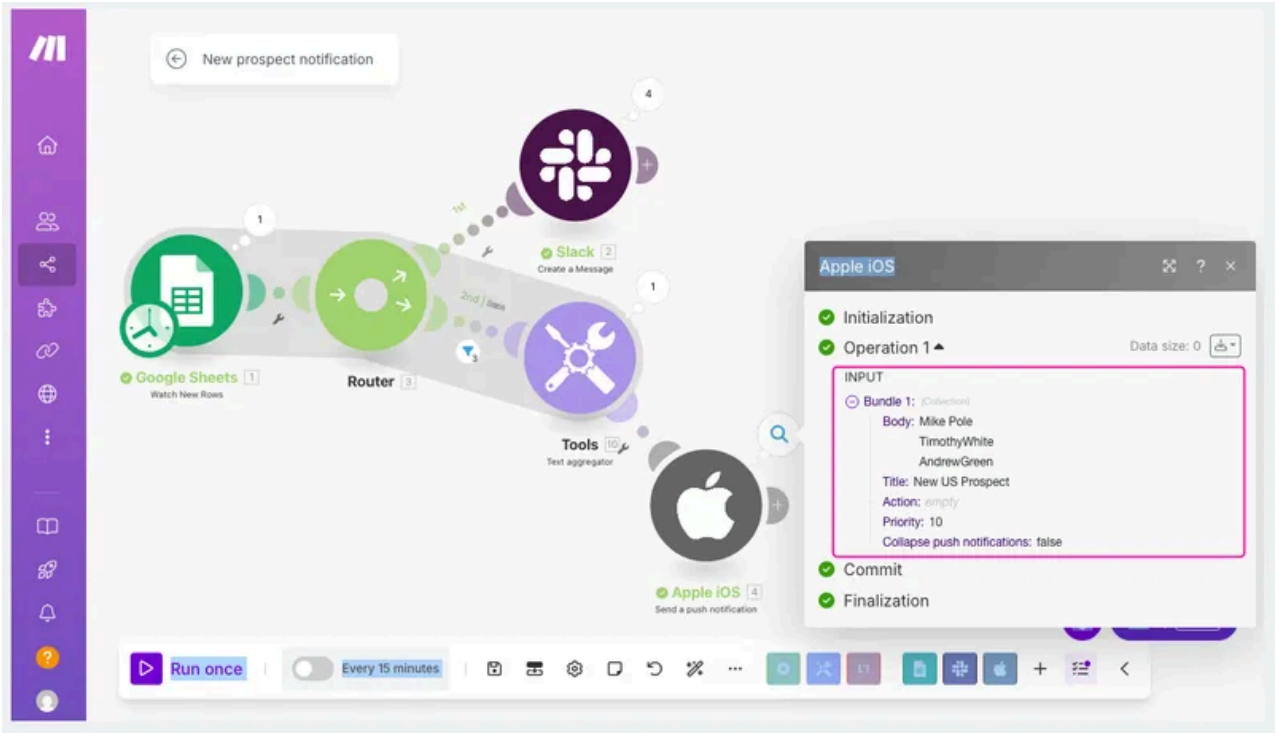
The filter should only let the USA prospects through:



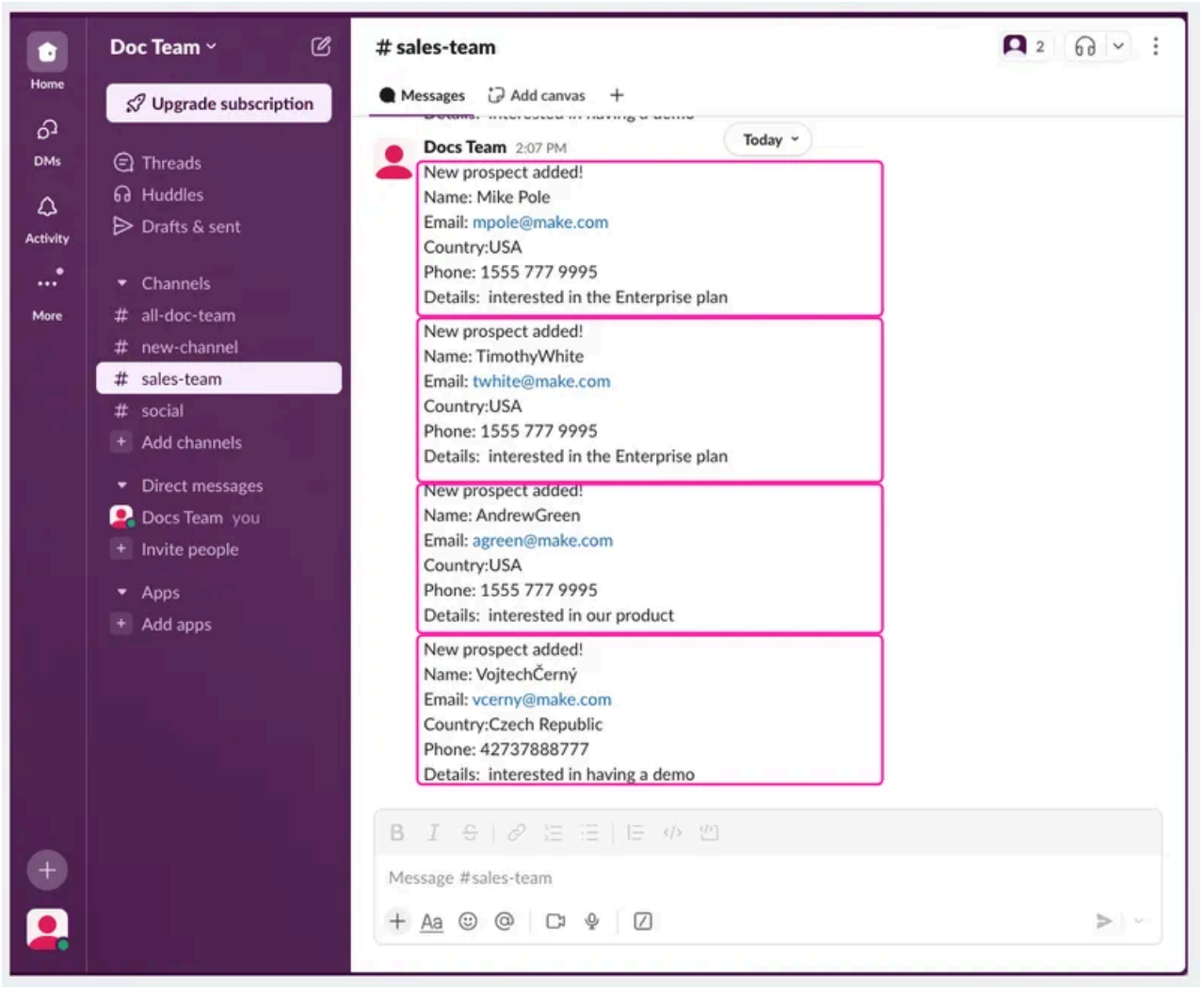
The Text aggregator should combine the USA prospects into a single bundle:



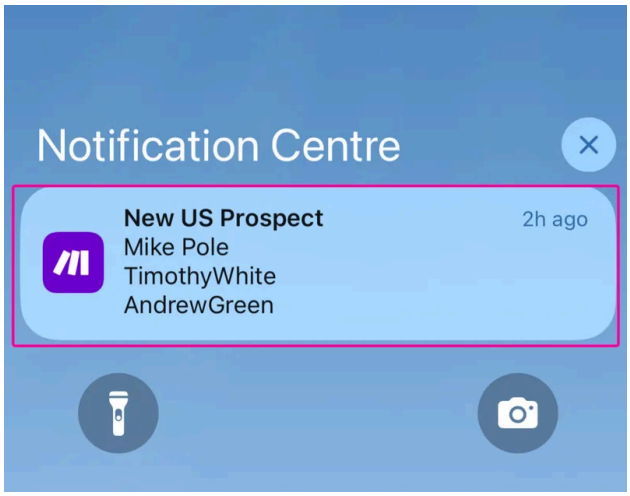
The Apple iOS module should only process this single aggregated bundle:



5 Check your Slack channel to confirm all notifications arrived.

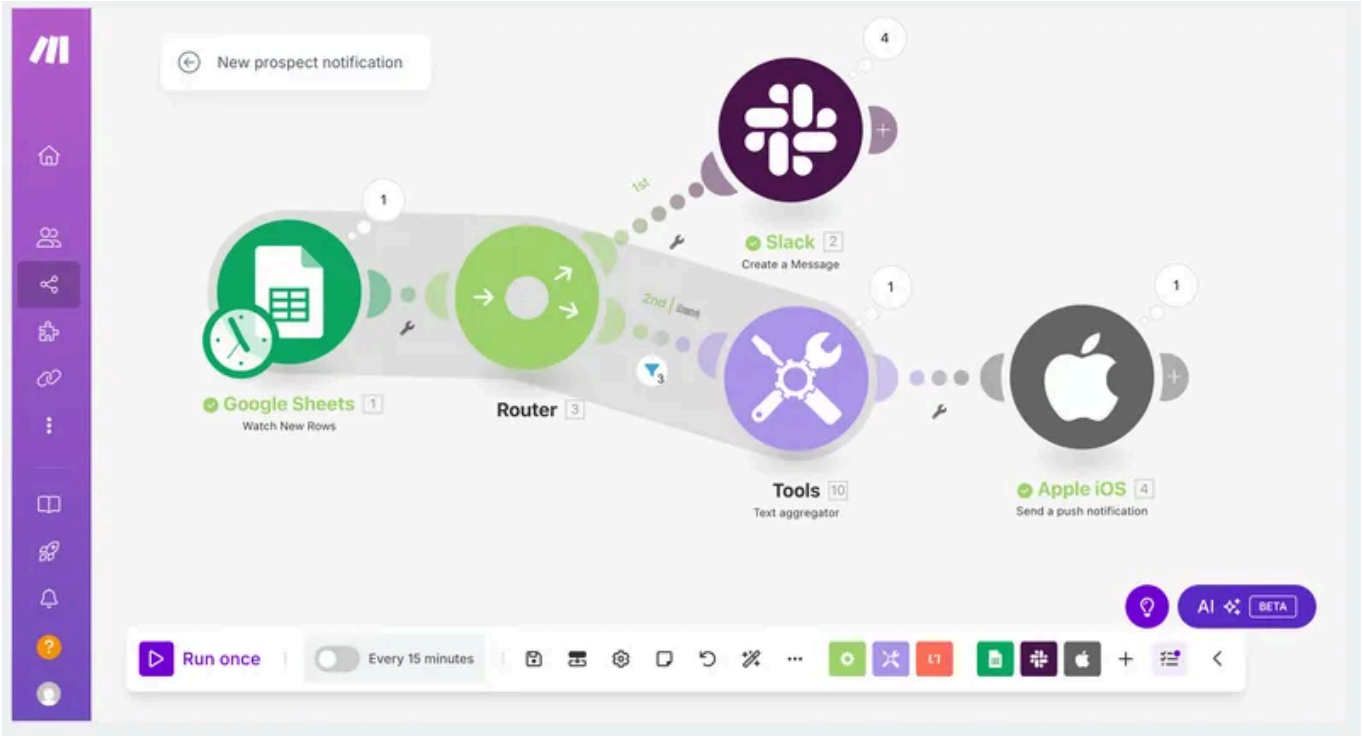


6 Check your mobile device to confirm you received only one notification that contains all the US prospects.



You've now created a scenario that:

- Monitors a Google Sheet for new prospects
- Sends all prospects to a Slack channel
- Sends only US-based prospects to the Sales Manager's mobile device
- Combines multiple US prospects into a single mobile notification



What's next?

Now that you've learned how to use routers to create multiple paths, filters to process data conditionally, and aggregators to combine related information, you can continue expanding your scenario by:

- Adding more filters for other countries or criteria
- Integrating additional apps like email or CRM systems
- Exploring other aggregator types for different data formats
- Learning about error handlers to make your scenarios more robust

To advance your Make knowledge, explore our free online courses at **Make Academy**.

2. Key concepts

Understand the fundamentals to start automating with Make:

Scenarios & connections

Learn the basics of scenarios, connections, and how to count operations

Apps & modules

Explore different types of apps, modules, and learn the fundamentals of webhooks

Data & mapping

Understand how to work with data and send it between modules

Tools

Get familiar with the essential tools for transforming data

Resources

Build your knowledge with glossary terms and keyboard shortcuts

2.1. Scenarios & connections

Learn the basics of scenarios and connections in these articles:

→ **Connect an application**

→ **Operations**

→ **Schedule a scenario**

→ **Active and inactive scenarios**

→ **Scenario execution flow**

→ **Scenario history**

→ **Scenario blueprints**

→ **Scenario templates**

2.1.1. Connect an application

To allow your scenario to get information from and send it to third-party applications, you need to create a connection. For example, a connection lets Make check your Gmail and enter the sender and subject into Google Sheets.

Because third parties have different technologies and standards, there are different ways to create connections in Make. For some apps, creating a connection requires logging in to the third-party app and clicking that you agree to let Make access your profile and data. Other third parties require an API key or OAuth credentials.

Make offers two types of connections:

1. Standard - the most common kind of connection and the type explained in this article.
2. Dynamic - a feature for Enterprise customers that lets you easily switch between multiple connections. This way you can have a separate connection for each team or department in your company and change them as needed.

Create a connection

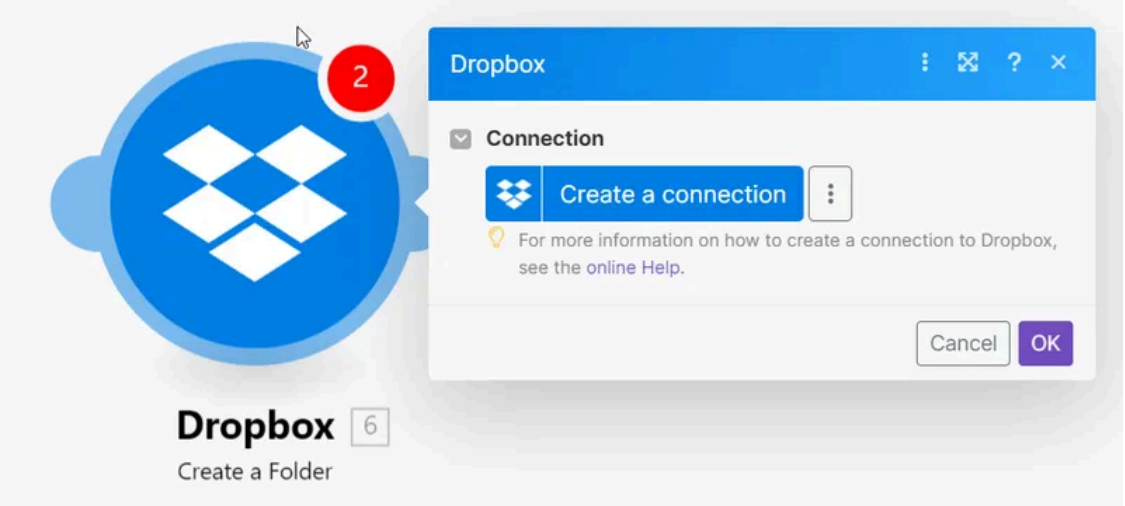
You create connections in the Scenario Builder when you add a module. All users in your team can use any connections you create in that team. Use your personal third-party account and credentials carefully, especially when your team has other users.

> All team members can manage connections as well. If you want to connect Make with your personal account or data, consider creating a separate team where you are the only member.

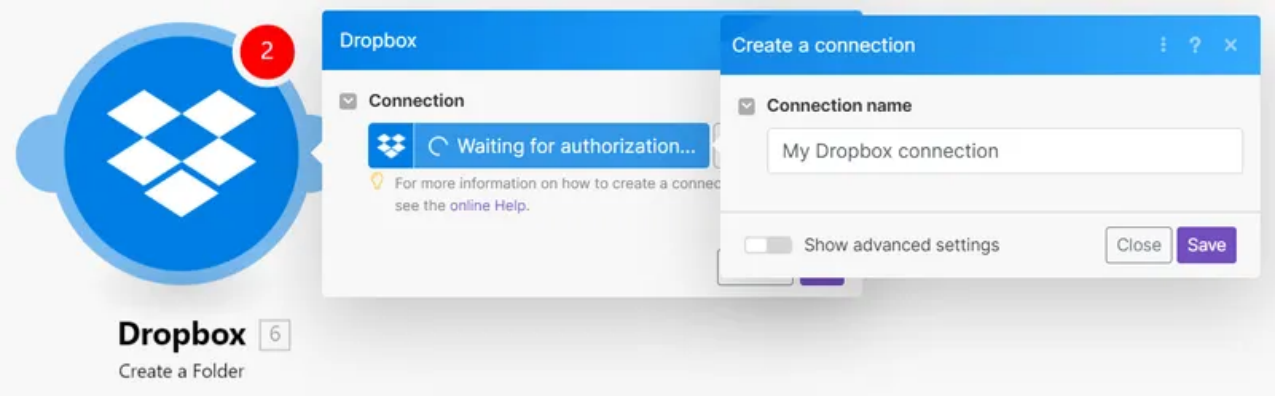
The following is the general way to create a connection. For more details, see the article on the specific app you want to connect.

1. Go to your scenario and add a module. For example, click **+** and search for **Dropbox**. Then select the module you want, such as **Create a Folder**.

2. Click **Create a connection**.



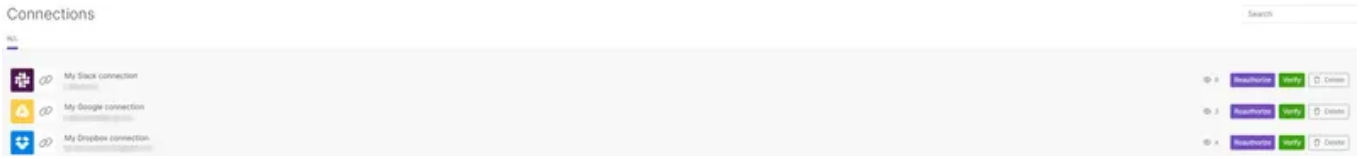
3. In the **Connection name** field, you can name your connection. For example, if you have two different Dropbox accounts, you might name one Dropbox-Company, and the second Dropbox-Personal.



4. If your app has fields for credentials such as username and password, API Key, Client ID and client secret, or any other values, refer to the article on the specific app you want to connect for more information.
5. Click **Save**. A new window prompts you to grant Make access to your data.
6. Grant Make access to your Dropbox account and data.

> Steps 5 and 6 can vary. Carefully follow on-screen instructions when granting consent.

You created a connection between your Dropbox account and Make and you can continue setting up the module. You can find the connection in the **Connections** section:



Manage connections

The connections tab lets you do any of the following to your connections:

- Edit
- Verify
- Reauthorize
- Delete

You can find the connections by clicking **Connections** in the left sidebar.

Edit connections

You can edit your connections after you create them. Click **Edit** to open a pop-up where you can fill in the new data for the connection.

Editing a connection replaces the original data with the new data, so you have to provide all the credentials necessary to create the connection. Make doesn't keep the original connection.

After you edit a connection, all modules that use the connection will start using the new credentials automatically.

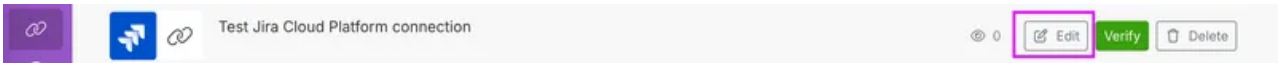
> Keep in mind that the API key or OAuth connection must have proper scopes to allow Make modules to do actions on your behalf. Before you edit a connection, make sure that the new connection has access to all the features you need, especially if you use the **Make an API call** module of the app.
>
> Make cannot ensure that the edited connection has the scopes required for all modules that use the connection.
>
> If a module uses a connection with insufficient scopes, the scenario ends with the `AccountValidationError`.

Edit API key connections

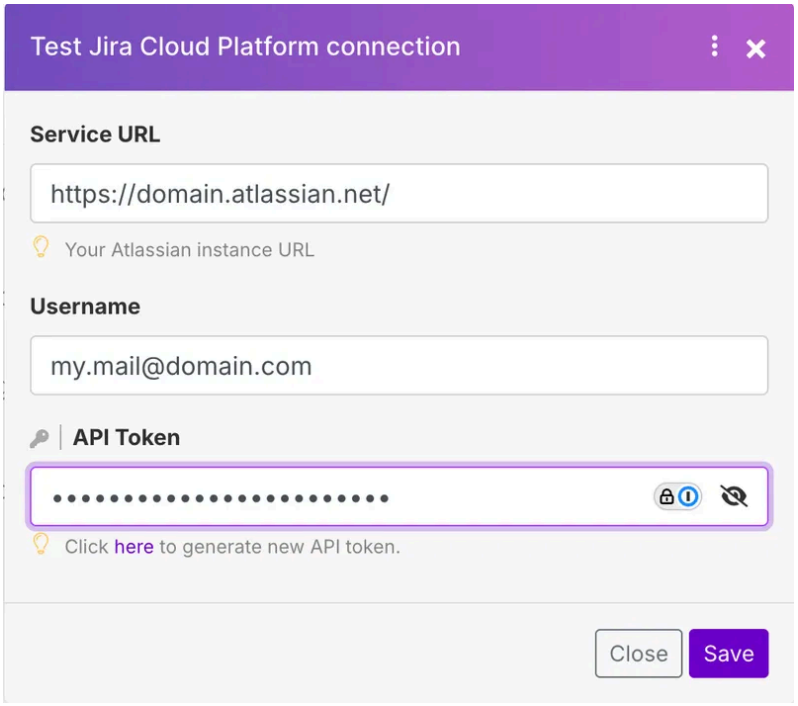
Editing an API key connection is similar to creating the connection, except that Make prefills the connection data based on the original connection.

To edit an API key connection:

1. Go to the **Connections** page with the list of connections associated with your team.
2. Click the **Edit** button in the row with the connection you want to update.



3. A pop-up appears. Fill in the new data in the form. Make creates a new connection from the data and replaces your old connection. This means, that even if your API key didn't change, you have to provide the key again to authenticate the updated data. Different apps may have a slightly different connection editing form. The example uses the **JIRA Cloud Platform** connection.



4. Confirm changes with the **Save** button.
   If you click the **Close** button, Make closes the form and keeps your original connection.

Make applies the changes immediately. After confirming, all modules that were using the old connection use the new one instead.

Edit OAuth connections

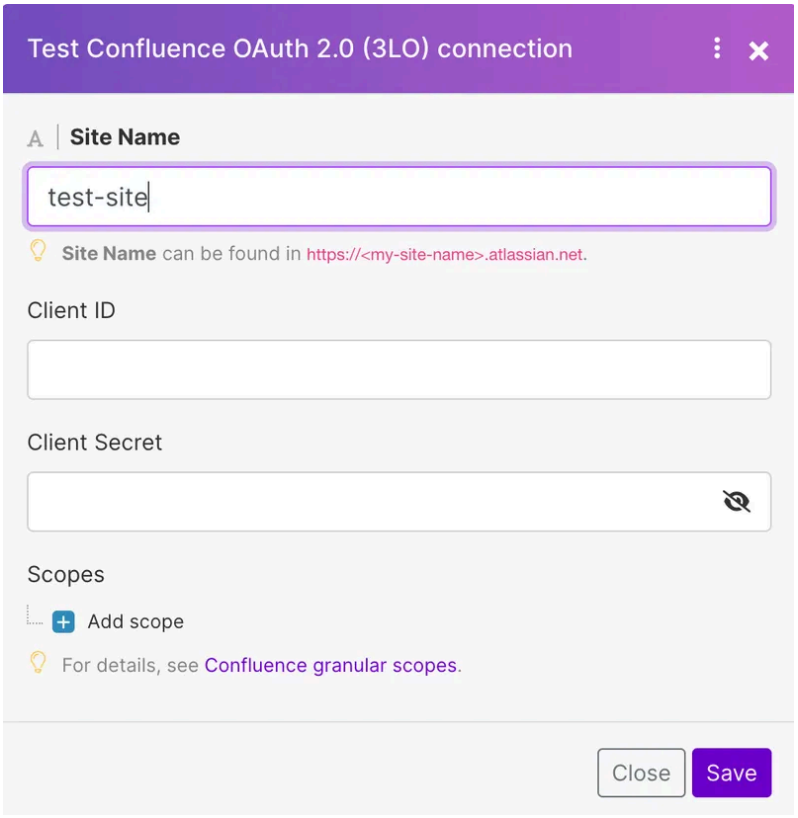
Editing an OAuth connection is similar to creating the connection, except that Make prefills the connection data based on the original connection.

To edit an OAuth connection:

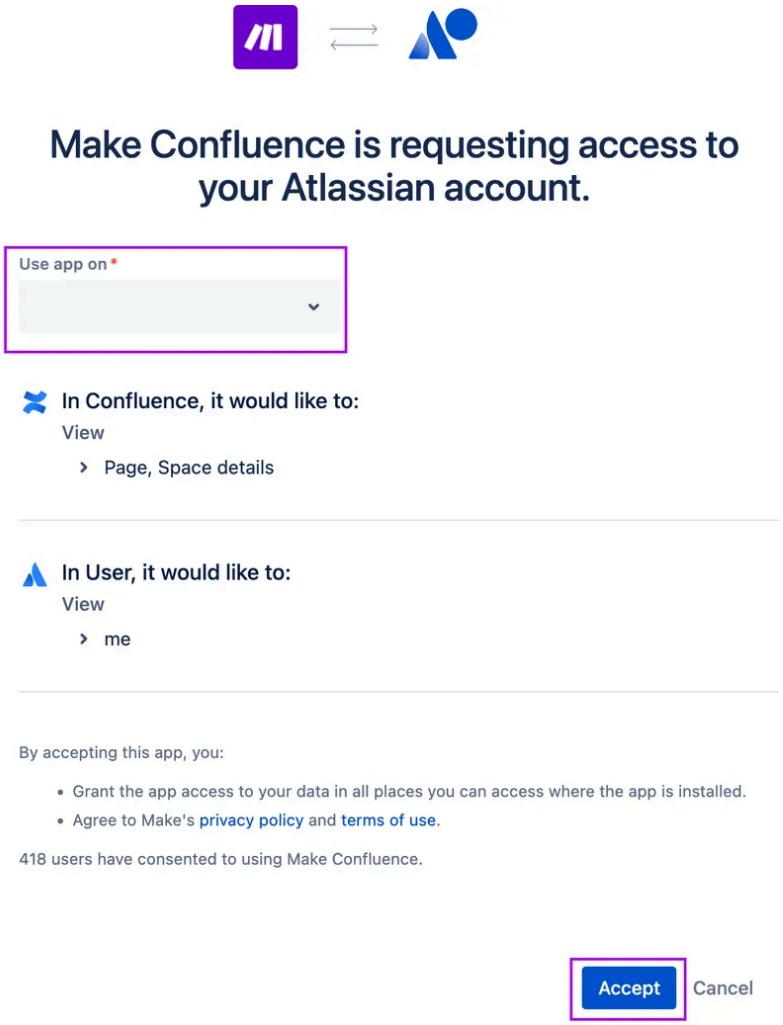
1. Go to the **Connections** page with the list of connections associated with your team.
2. Click the **Edit** button in the row with the connection you want to update.



3. A pop-up appears. Fill in the new data in the form. Make creates a new connection from the data and replaces your old connection. This means, that even if your API key didn't change, you have to provide the key again to authenticate the updated data. Different apps may have a slightly different connection editing form. The example uses the **Confluence** connection.



4. Confirm changes with the **Save** button.
   If you click the **Close** button, Make closes the form and keeps your original connection.
5. Fill in the OAuth access form. Check the required fields to create the connection and review the connection scopes. If everything is correct confirm the connection update with the **Accept** button.



Make applies the changes immediately. After confirming, all modules that were using the old connection use the new one instead.

Verify connections

You can see whether your connection works by clicking **verify**. The button briefly changes to **Verifying ...** as Make checks. A green check mark in a circle appears when successful.

Reauthorize connections

Some third parties, such as Google only grant access for a limited time. In this case, you may need to occasionally reauthorize your connection. Just click **Reauthorize** and follow any on-screen prompts that appear.

Delete connections

If you need to delete a connection, click **Delete**. The button turns red and asks again to confirm that you want to erase that connection. Click again to delete your connection.

> Deleting a connection may cause your scenarios to stop working. Be sure that the connection is not used in any scheduled or active scenarios.

2.1.2. Operations

Each time a module performs an action, it uses an operation. For example, adding a row to your Google Sheet or getting data from your Gmail account, each uses one operation.

Understanding your operation usage is important because Make charges based on the total number of operations performed by the modules in your scenario. More modules and bundles processed in a scenario result in a higher number of operations.

If you exceed your monthly operation limit, your scenarios will be paused until the next usage reset. To resume your scenario immediately, you can purchase additional operations or you can enable the extra operations auto-purchasing to prevent your scenarios from pausing in the future. You can learn more about the available options below.

> In addition to the number of operations, each plan gives you a certain amount of data that your scenarios can process during the usage period. You can find more information about your usage allowance **here**.

How modules use operations

Each time a module in a scenario performs an action, it uses one operation. If a module outputs multiple bundles of information (like a *List* module returning multiple records), it makes the following module run once for each bundle, processing each record separately. The total number of used operations for a module equals the number of performed actions.

Operations by module type

All **module types** use one to many operations, depending on their place and function in the scenario:

- **Trigger** modules use one operation regardless of whether they receive any data or not.
- **Search** modules use one operation to run, but can output multiple bundles. For example, the Google Sheets *Search Row* module will run once and use one operation, but it can return information about multiple rows after one run. Each row is a separate bundle of data.

- **Action** modules (*Add*, *Update*, *Delete*, etc.) use as many operations as many runs they need to make to process all the input data. If you have 10 rows returned by the Google Sheets *Search Row* module the next *Delete Rows* module will run 10 times to delete all the 10 rows and will use 10 operations.
- **Aggregator** modules combine multiple bundles into one. The aggregator uses one operation for each aggregation. If the aggregator creates one array from your Google Sheets records, it will use just one operation.
- **Iterator** modules are opposite to the aggregator modules. Iterators take one array bundle and split it into multiple bundles as an output, so they use one operation for the entire array, but modules after them will receive more than one input.

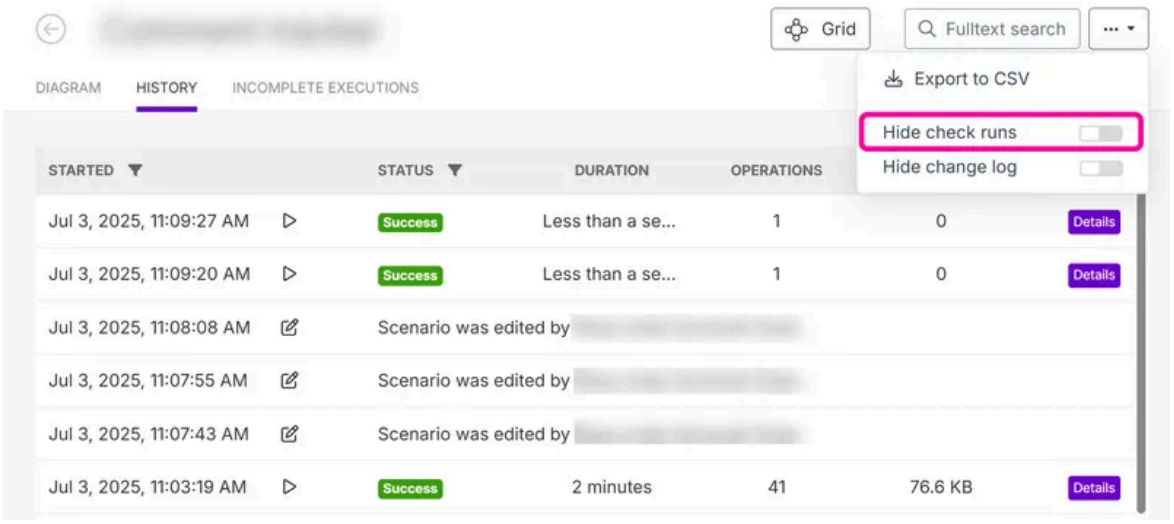
The following modules do not consume operations:

- **Error handler modules** (Rollback, Break, Resume, Commit, Ignore)
- **Router** modules and **filters**

Check runs

A scenario run always consumes at least one operation to account for the trigger module's check for new data. Even when no new data is found, the run still consumes operations. This type of scenario run is a check run. A check run is when a scheduled scenario runs, but no data is processed.

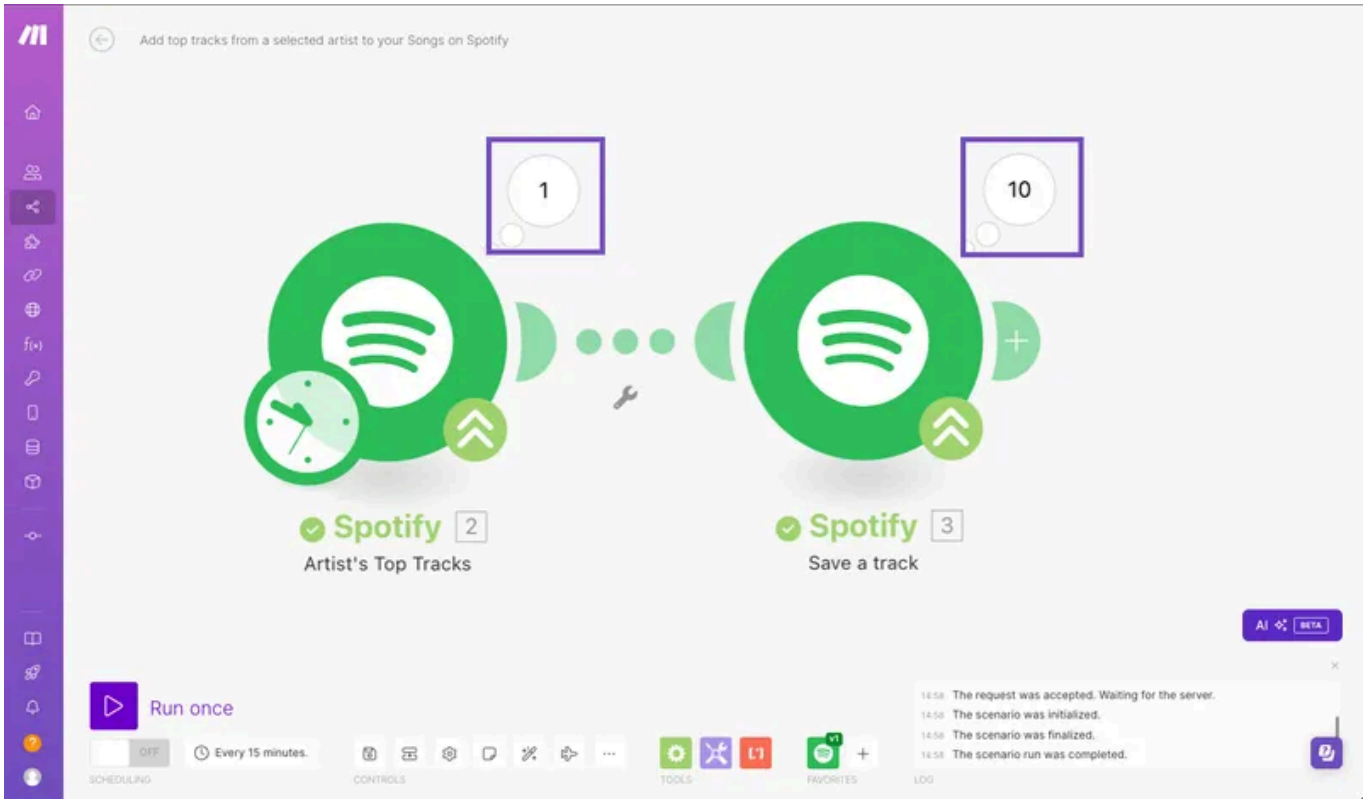
You can view check runs in the **History** tab in your scenario. If you don't see them, ensure that the **Hide check runs** toggle is off.



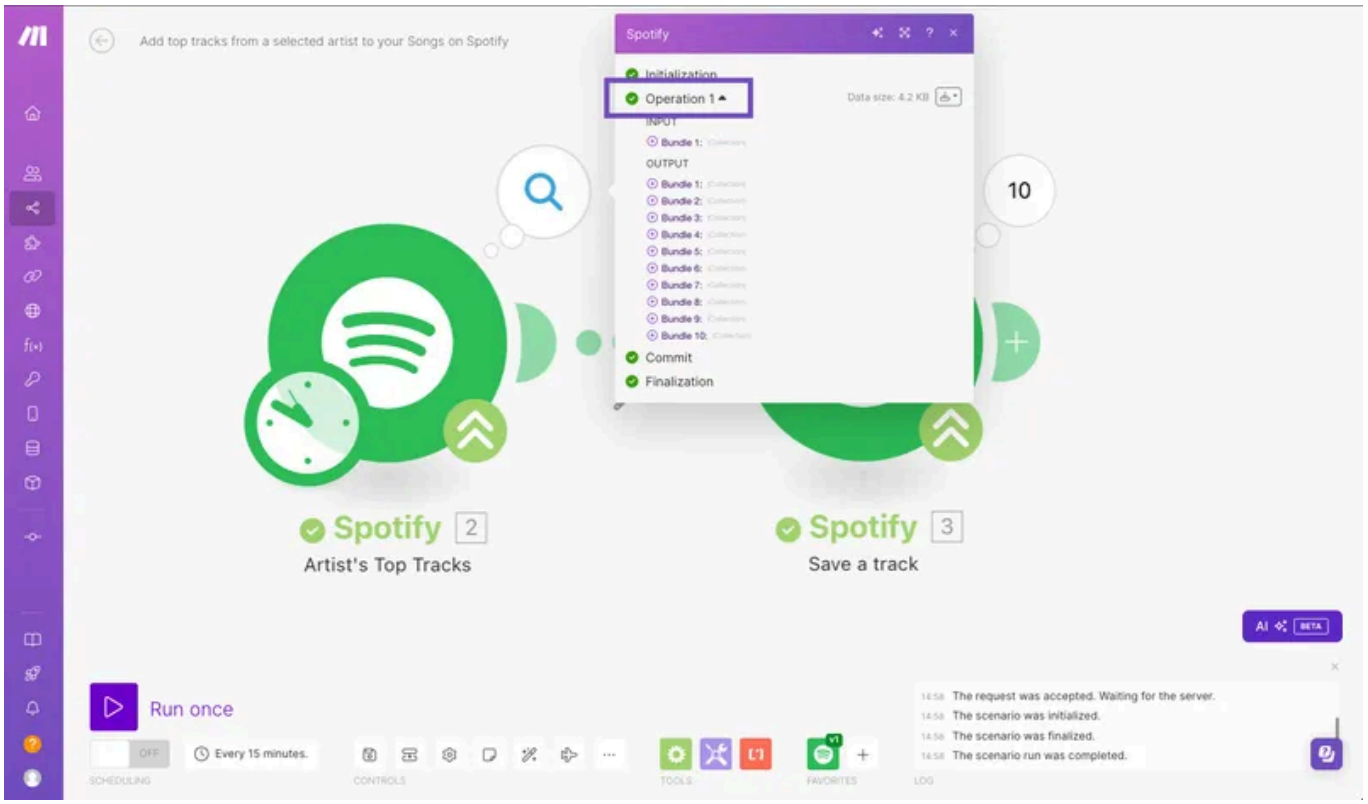
Example: Add top tracks from a selected artist to your liked songs on Spotify

In this scenario, Make gets the list of the Top 10 songs of an artist you select, and adds it to the list of the songs you liked.

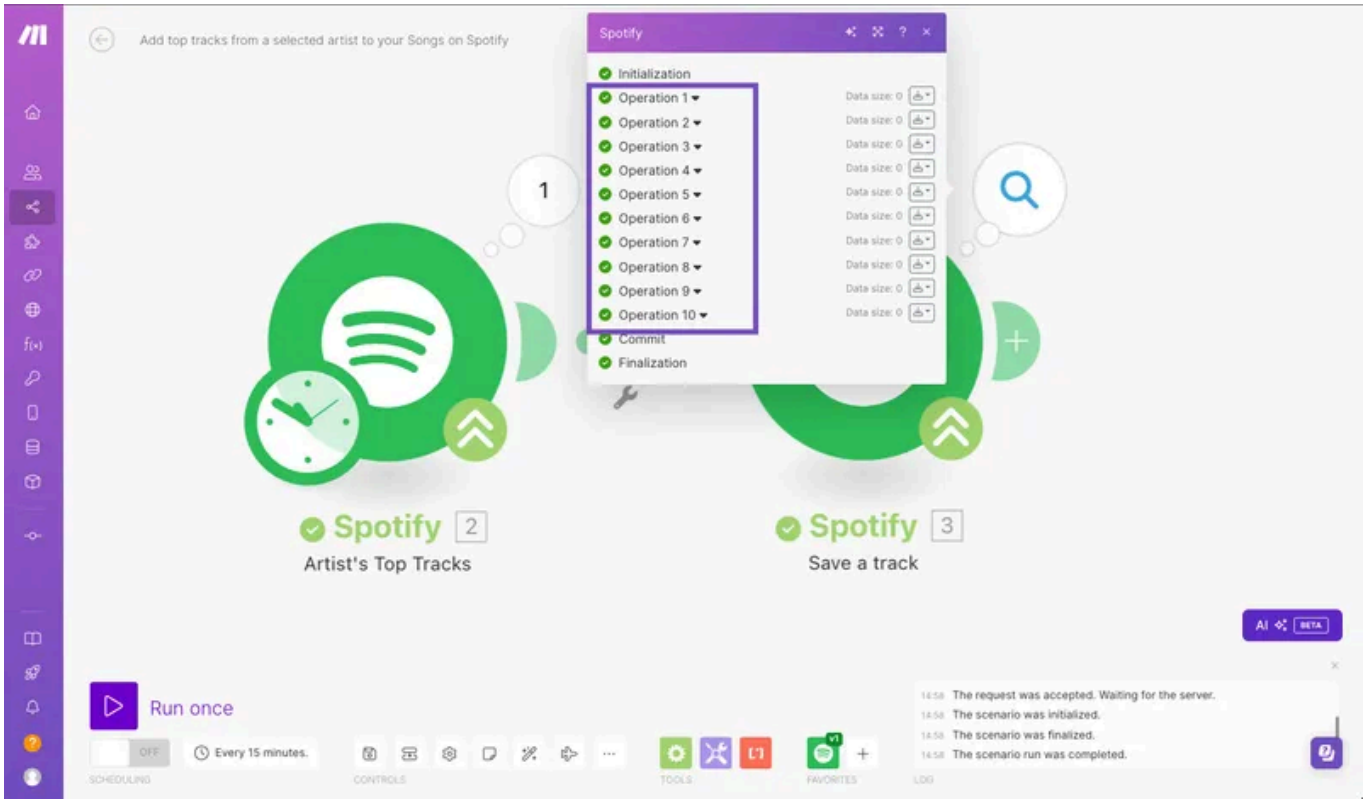
Once the execution of a scenario is completed, Make displays the number of operations performed by each module in the little white circle above the module.



In this example, the first Spotify module returned 10 songs and performed 1 operation while getting the information. Each song is returned as a bundle of information, so the next module has to work with 10 bundles.



The second Spotify module had to add each of the 10 songs to the playlist, requiring it to perform this action 10 times. As a result, the module processed 10 bundles and used 10 operations.



The run of this scenario required 11 operations in total.

In this example, Spotify limits each artist to 10 top songs. However, you can control the number of operations your scenario uses by restricting the number of results for *Search* and *List* action modules. For example, you can create a scenario that adds new songs from an artist to your playlist but limit the *List* module to return only 5 songs. This way, only 5 new songs are added per run instead of an entire album of 30 songs, saving operations. Additionally, you can use filters to process only specific bundles returned by modules. This ensures that only the bundles passing the filter are processed, consuming fewer operations.

Checking the total operations count for a scenario

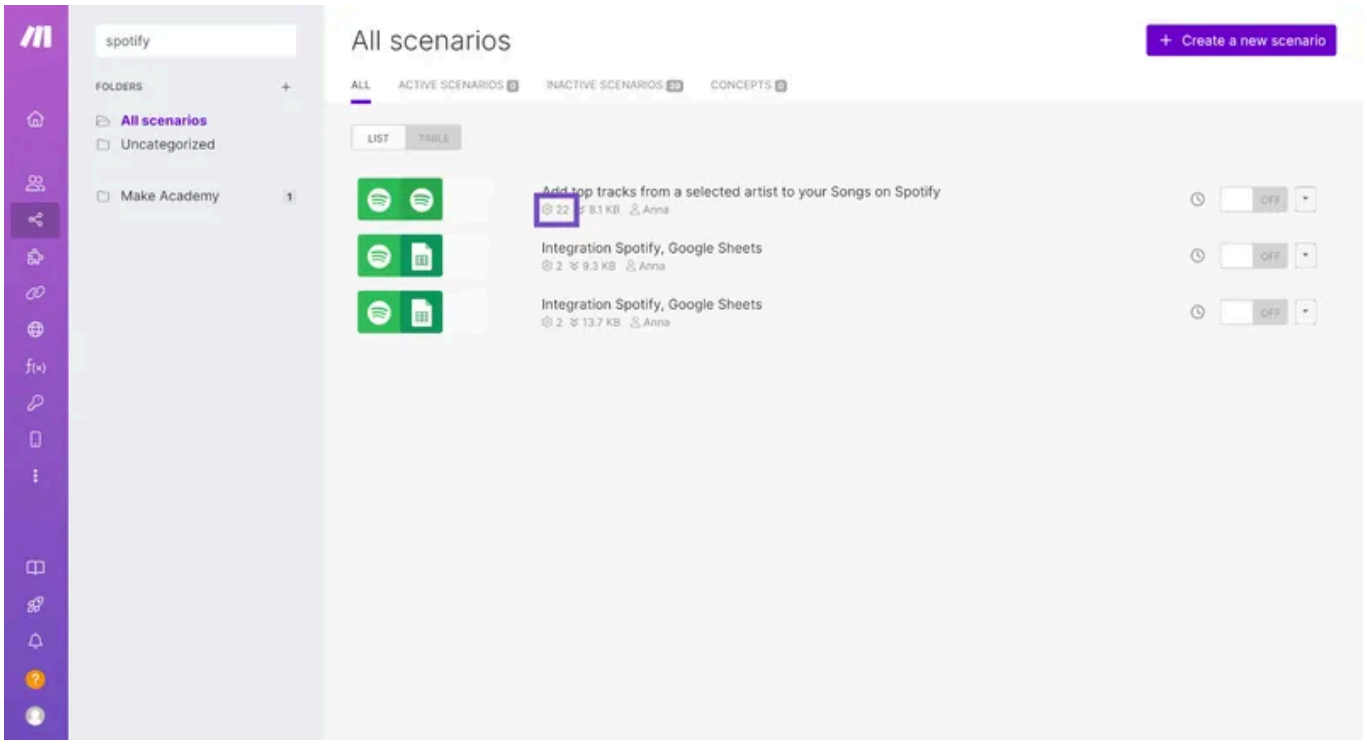
A scheduled scenario consumes operations each time it runs, and its total operations can add up quickly. For example, if your Spotify scenario runs every hour, it consumes 264 operations per day (11×24). If you schedule it to run every 5 minutes, it would consume 3168 operations per day (11×24×12).

Schedule your scenarios carefully to optimize the total operations count. You can reduce operations used by scheduling certain scenarios to run less frequently.

You can easily check the total operations used in your scenarios in the scenario list or scenario history.

From the scenario list

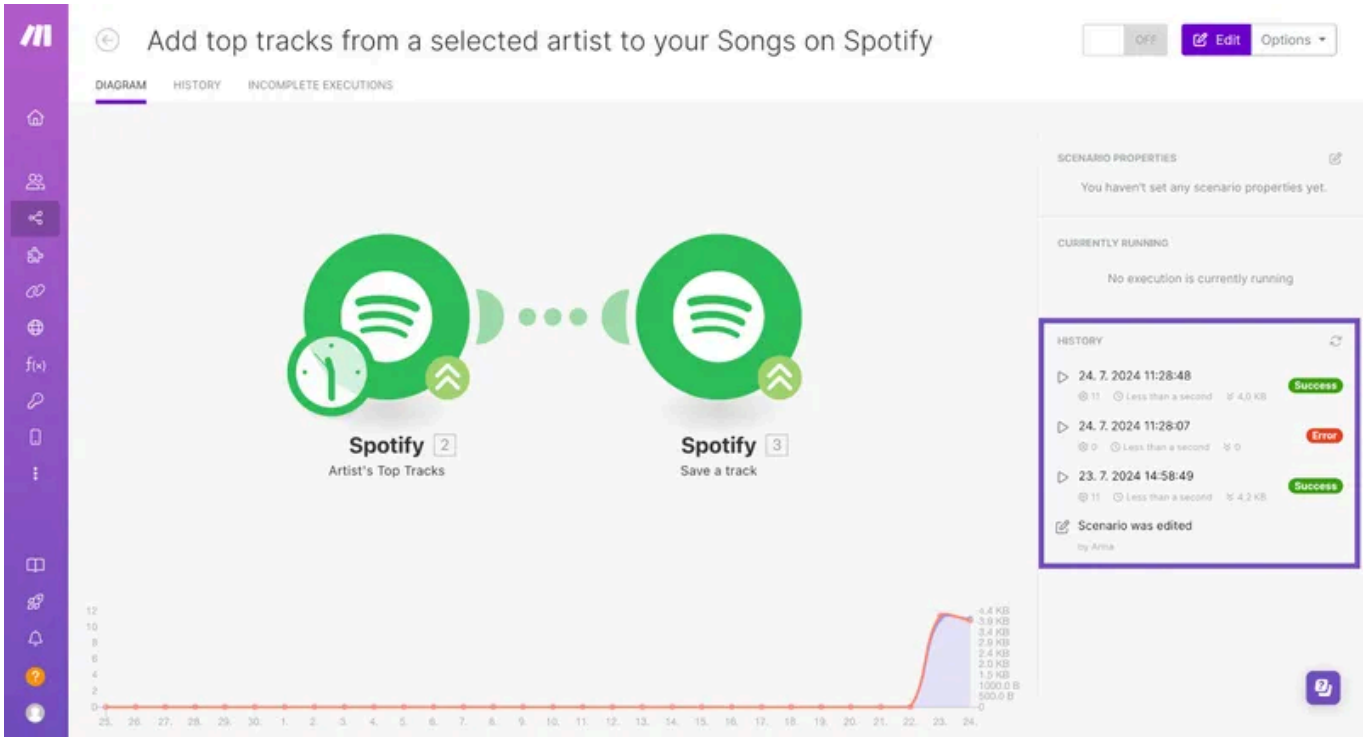
When you open a list of scenarios, you can see how many operations each scenario has consumed under its name. If you hover over the number you can see the date when the count started (the beginning of the usage period).



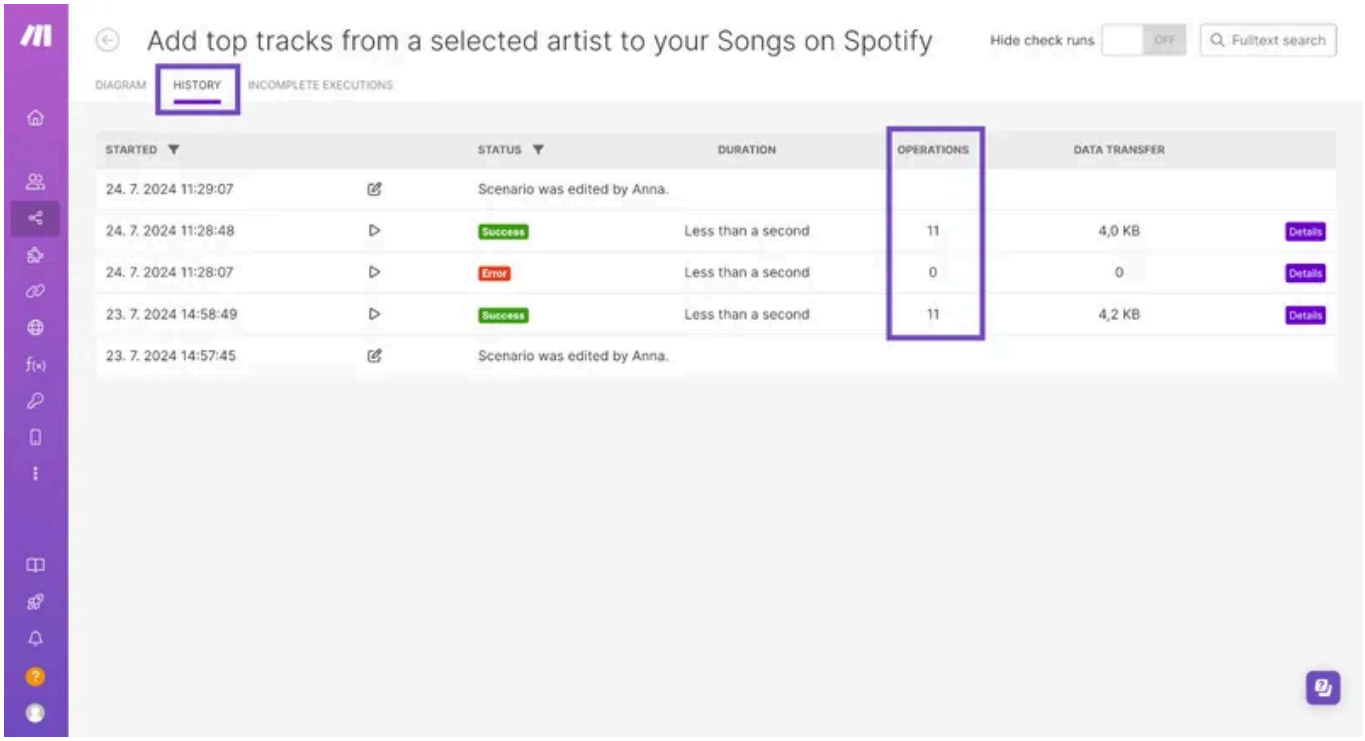
For yearly subscriptions, the label displays the reset date, but the data represents an aggregated usage within the last 60 days.

From the scenario history

To open the scenario execution history, click on the scenario from the list of scenarios. Under **History** on the right, you can see the number of operations consumed for each scenario run.



Similarly, you can switch to the **History** tab and see the number of consumed operations for each run there in a table format.



> When a regular execution ends with a warning or error, an **incomplete execution** is created. Any attempts to fix it are tracked in the **Incomplete Executions** tab, with each attempt consuming operations. These operations add to the total used by the scenario and are included in the operations usage displayed in the list of scenarios, even though they aren't visible in the **History** tab. You can view the operations used to resolve an incomplete execution in the incomplete execution details. Note that, by default, the storage of incomplete executions is disabled, but you can enable it individually for each scenario.

Checking the total operations count for the Organization or Team

While it is important to know how many operations each scenario consumes, it is also important to know how many operations your Organization and each Team within the Organization use. This can be easily checked from both Organization and Team dashboards.

To open the dashboard, click **Organization** or **Team** in the left-side navigation.

From the Organization dashboards it is also possible to see the limits you have according to your plan, how many operations were used during the current period, and to check the options of purchasing additional operations if needed.

If you are running out of operations

When you run out of operations, your scenario will pause until additional operations are added or the next billing cycle begins. Incoming webhooks will be queued based on your available webhook storage and processed once your scenarios resume with the new operations. Additionally, scenarios that poll for data will search for available records since the last successful run.

> If you are on the Enterprise plan, your scenarios will continue to run thanks to the **operations overage protection**. During this time, your Account Manager will contact you to resolve the issue and prevent any interruptions.

We'll notify you when you're approaching 75%, 90%, and 100% of your purchased operations. This way, you can ensure you have enough operations to meet your needs or take action to prevent your scenarios from pausing.

Here’s what you can do if you run out of operations before the next billing cycle:

- Upgrade your **subscription** before the end of the cycle.
- Purchase **extra operations** in bundles of 1,000 or 10,000 operations at the fixed price defined in your subscription. Extra operations work the same as pre-purchased operations and expire after the current billing period (monthly or yearly).
- Enable the auto-purchasing: once you use all 10,000 extra operations, Make auto-purchases 10,000 operations again. Make limits auto-purchasing to the number of operations in your subscription. For example, if your plan has 80,000 operations, auto-purchases can happen up to 8 times. Make charges you for the 10,000 operations based on your plan, plus a 30% surcharge. You can learn more about this option **here** .

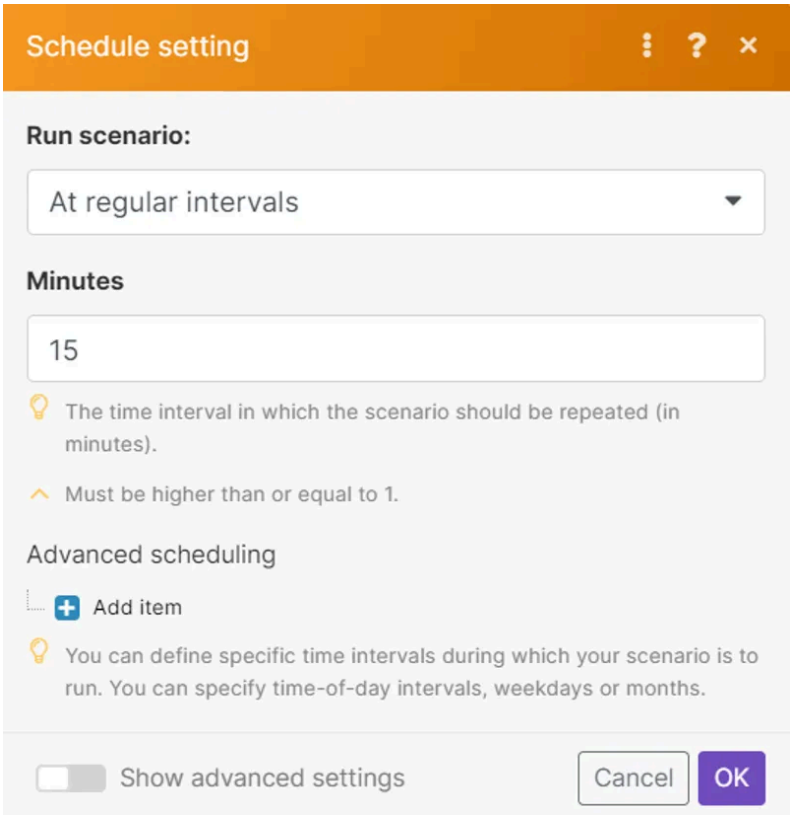
2.1.3. Schedule a scenario

Make allows you to define when and how often an **active** scenario executes.

1. Go to the Scenario Builder.
2. Click "Every 15 minutes" in the Scenario Builder toolbar.
3. The **Schedule setting panel** appears. Use this panel to customize your scenario schedule.

Schedule setting panel

The schedule setting panel lets you adjust the running schedule of the scenario. By default, a scenario runs every 15 minutes, but within this panel, you can customize the run schedule and the interval.



Run a scenario

You can select the following schedule options:

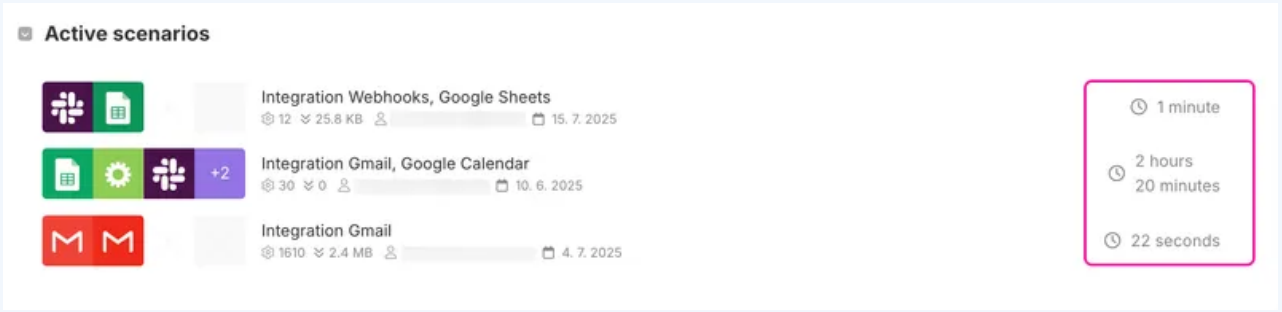
- Immediately

- At regular intervals
- Once
- Every day
- Days of the week
- Days of the month
- Specified dates
- On demand

When choosing any of these options, you may have to further define how often you want the run to occur. For example, if you select the **At regular intervals** option you must define the time interval (in minutes) between two consecutive scenario runs.

The minimum length of the interval depends on your **plan**.

The **Immediately** option is available only for some triggers. For more information on triggers, please see the **types of modules** help guide.

> You can always see how much time is left until the next scenario execution on the organization dashboard.
>
> 
>

On demand scheduling

With the On demand scheduling option, your scenario is waiting to be triggered by an API call or the Run once button. The scenario does not have any schedule and it does not run unless you start it manually.

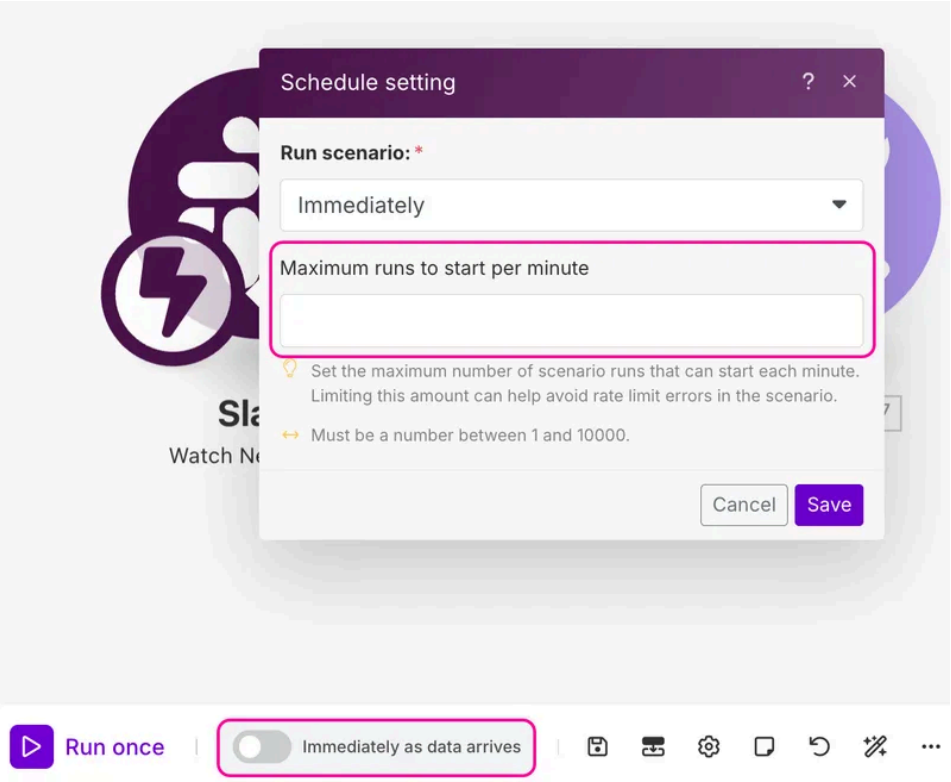
Start and end date

To define the time range in which an activated scenario runs, check the **Show advanced settings** box and fill in the **Start** and **End** date.

Set up scenario rate limit

If your scenario has an instant trigger that's triggered frequently within a short amount of time (tens of runs per minute) and connects to services like Slack or Shopify, it may exceed those services' rate limits. When this happens, the scenario will return an error for sending too many requests in a short time.

To avoid this, you can configure the **Maximum runs to start per minute** in the scenario's scheduling options.



For example, if you set the maximum to 30 runs per minute, and that limit is reached, Make will keep subsequent requests in the queue and process them gradually as the configured limit allows it.

> If your scenario has the **webhook response** module in it and exceeds the rate limit, the caller of the webhook will receive an **HTTP 429 Too many requests** error. This means the caller should wait and retry the request later.

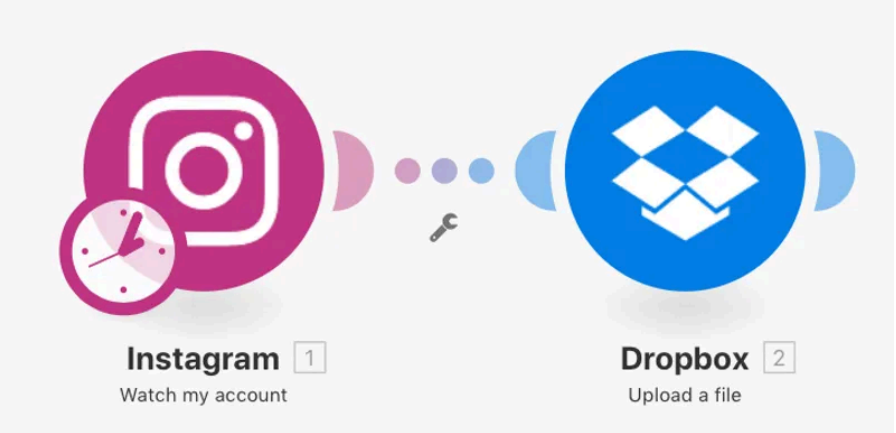
2.1.4. Active and inactive scenarios

There are two ways you can work with scenarios in Make. Depending on how you want to use your scenario, you will either activate or inactivate it.

- **Active** scenarios run periodically according to their defined **schedule**.
- **Inactive** scenarios are executed once and whenever you wish to run them.

You may change between the two statuses whenever you like (active scenarios can become inactive and vice versa).

When to activate or deactivate



There are specific purposes for activating your scenarios as well as other reasons for making your scenarios inactive.

Using the example of connecting Instagram to Dropbox, you want to activate your scenario when you are wanting to consistently watch your account on a long-term basis. In this case, whenever you add a photo, it will automatically upload this photo to Dropbox. It will do so according to its **schedule**.

On the other hand, the inactive scenario is great for when you simply want to select photos from a specific time interval. Clicking the **Run once** icon in the toolbar will start a one-time execution of the scenario.

Activating scenarios

A newly created scenario is, by default, inactive. You can activate it by clicking the **ON/OFF** button in the scenario details page.

2.1.5. Scenario execution flow

In this section, you will learn how a scenario executes and how data flows through a scenario. It will also show you where you can find information about your processed data and how to read it.

Once a scenario is **set-up** correctly and **activated**, it will execute according to its defined **schedule**.

The scenario begins with the first **module** responding to an event it has been set to watch for. If it returns any bundles (data), the bundles are then passed on to the next module and the scenario continues. If it does not return any bundles, the scenario does not continue and ends after the first module. In the case where there are returned bundles after the first module, the bundles will then go through each proceeding module one-by-one. If the bundles are processed correctly throughout all of the modules, the scenario is then marked as **Successful**.

Example: Connecting Facebook to Dropbox

The example below shows how Make connects three modules in a scenario. It explains how Make downloads photos from **Facebook**, converts them to another format, and sends them to a selected **Dropbox** folder.

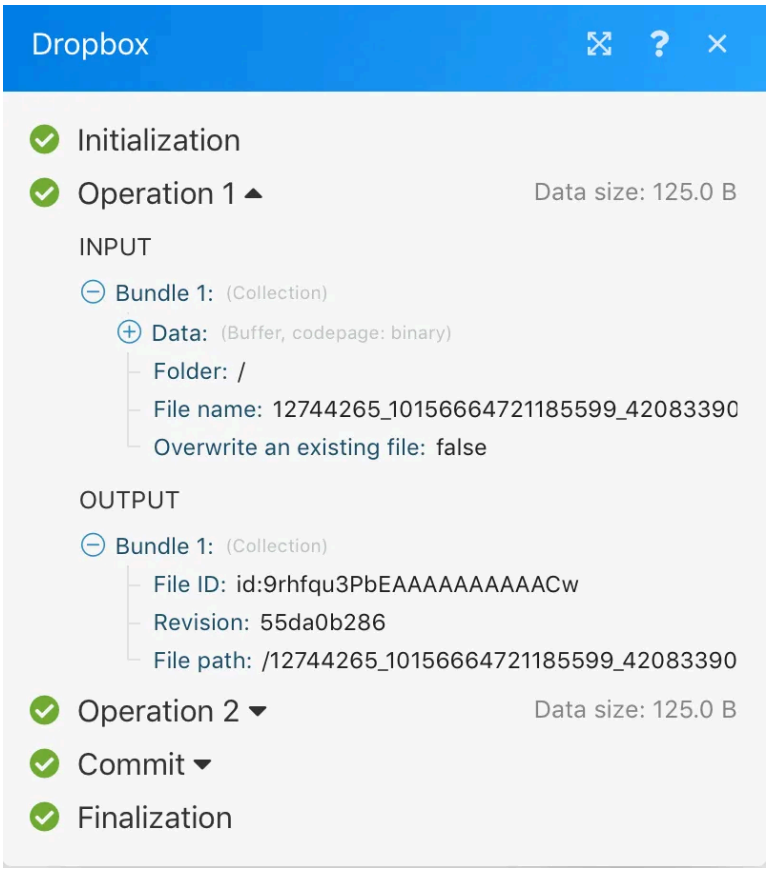


When the scenario begins, the first step is to watch for bundles. In this case, it is to watch for photos on Facebook. If it does not return a bundle (a photo), the processing of the scenario does not continue and ends after the first module.

If a bundle is returned, the bundle then passes through the rest of the scenario. The bundle is first recieved through the **Watch photos** module, then it goes through the **Convert a format** module for the Image app, and then goes through the **Upload a file** module for Dropbox to reach its final destination, the Dropbox folder.

It is also important to note that if Facebook returns multiple bundles, for example two bundles, the processing of the latter bundle will not start until the first bundle is converted and uploaded to Dropbox.

Information about processed bundles



For each module, the bundle goes through a 4 step process before going on to the next module or reaching its final destination. The 4 step process is **Initalization, Operation, Commit/Rollback, and Finalization**. This is called **transaction processing** and it helps to explain how data was processed in a module.

Once a scenario run is complete, each module displays an icon showing the **number of operations** performed. Clicking this icon will display the detailed information about the processed bundles, in the format described above. You can see which **modules settings** were used and which bundles were returned by which module.

The picture above illustrates the processing of the last module used in the scenario above, the Dropbox module, **Upload a file**.

The module received the following input information:

1. Converted image
2. Selected folder where the image shall be uploaded to
3. Original name of the Facebook image

After processing, the module returned this output information:

- Image ID assigned by Dropbox
- Full path where in Dropbox Make uploaded the file

The above information is captured for each bundle separately, as marked by the drop down boxes Operation 1 and Operation 2 in the image.

Errors while executing a scenario

An error may occur during the scenario run. For example, if you delete the Dropbox folder that you have set as the target folder in the module setting, the scenario will terminate with an error message. For more information on this and to learn how to handle errors, please see the **Error handling** help guide.

2.1.6. Scenario history

When you click on a scenario in the scenario list, you arrive at scenario details. From there, navigate to the **History** tab. Scenario history contains information about scenario executions and user changes to the scenario (change log).

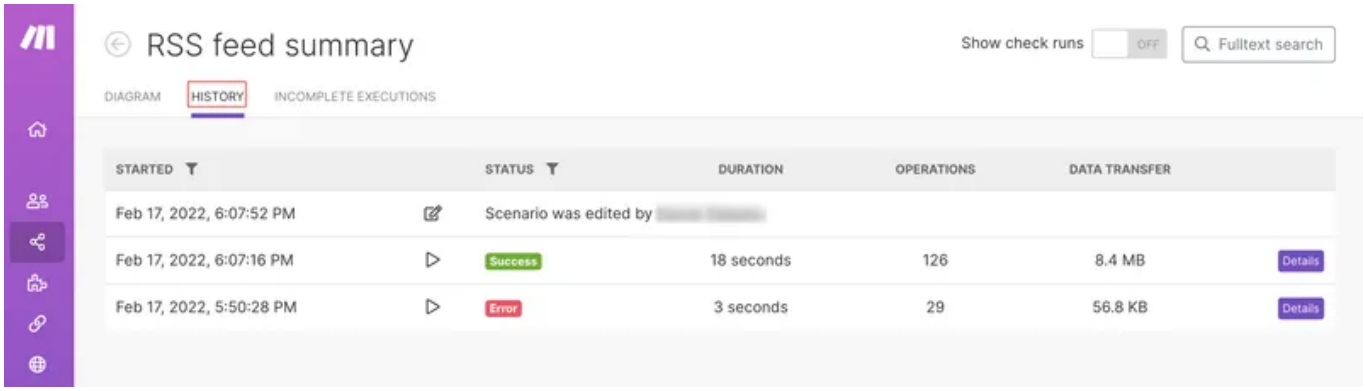
Execution and change log entries

In scenario history, execution entries include the following information:

- Execution date
- Icon indicating the scheduling type
- Status (Success, Warning, Error)
- Execution duration
- **Number of operations**
- Size of transferred data
- **Details**

Change log entries include user actions such as:

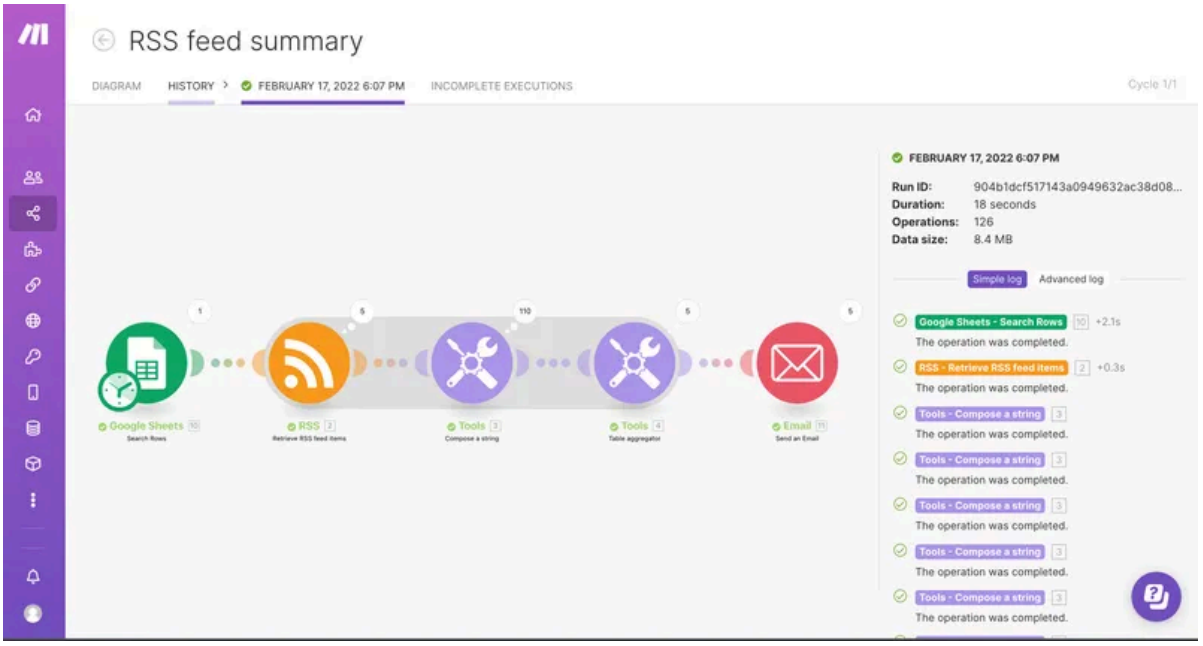
- Scenario scheduling changes
- Scenario edits
- Scenario activation



> The number of days your execution history entries are stored depends on your pricing plan. See the **Make pricing page**.

Scenario history details

In scenario history, click **Details** on the right-hand side of an entry for detailed information about the specific scenario execution, including which bundles were processed. Here you can view both a simple log and an advanced log.



Scenario history filtering

You can customize the entries displayed in your scenario history log, such as restrict entries to execution history only. In the **History** tab, click on the three dots in the upper right-hand corner to **hide check runs** and/or **change log** entries.

2.1.7. Scenario blueprints

In Make, you can save scenarios as **Blueprints** and share them with other Make users within or outside your organization, or reuse them later for your needs.

A Blueprint is a reusable version of a scenario that includes:

- Scenario's modules
- Module settings
- Mapped values

Blueprints help to set up automations quickly, without having to start from scratch. You can:

- Back up your scenarios in case you lose access to or switch to another Make account.
- Export a blueprint to reuse a scenario later.
- Share a blueprint with other users, who can then import it and run the scenario as-is or adapt it to their specific needs.

> While Blueprints include scenario's modules, settings, and mapped values, users still need to **create connections** for their accounts after importing the blueprint.

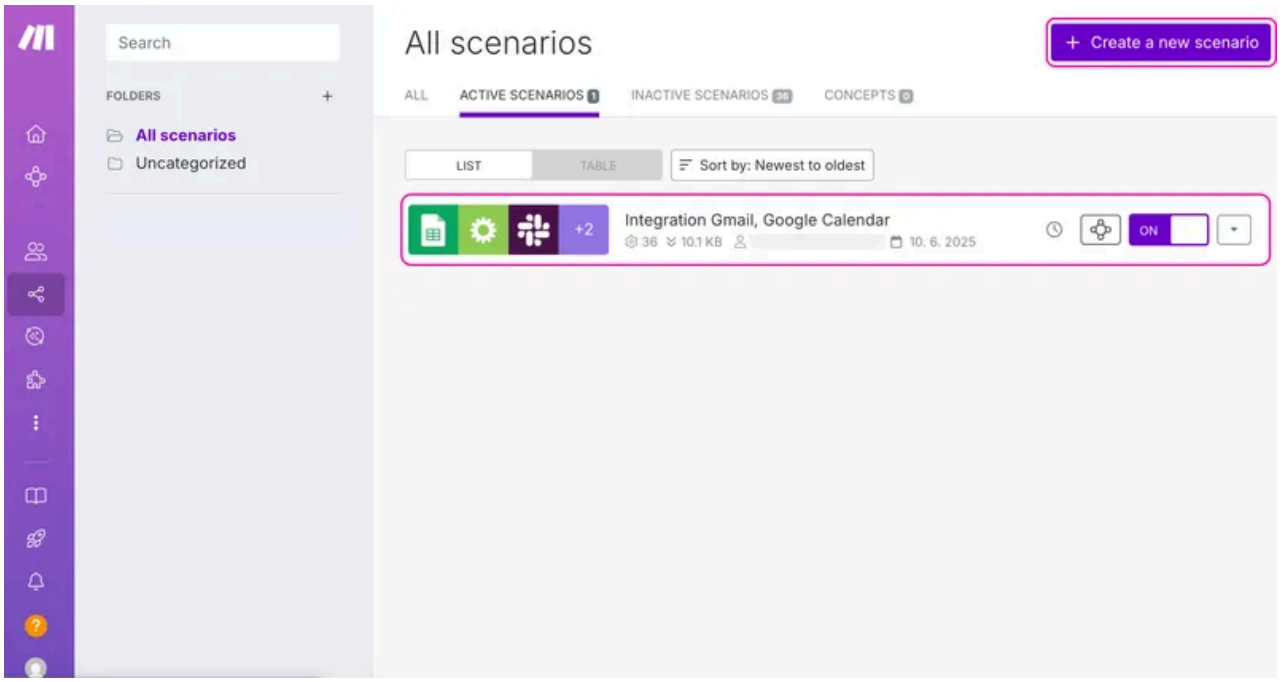
How to export and import blueprints

You can export a blueprint of the scenario as a `.json` file and import a blueprint in the same format.

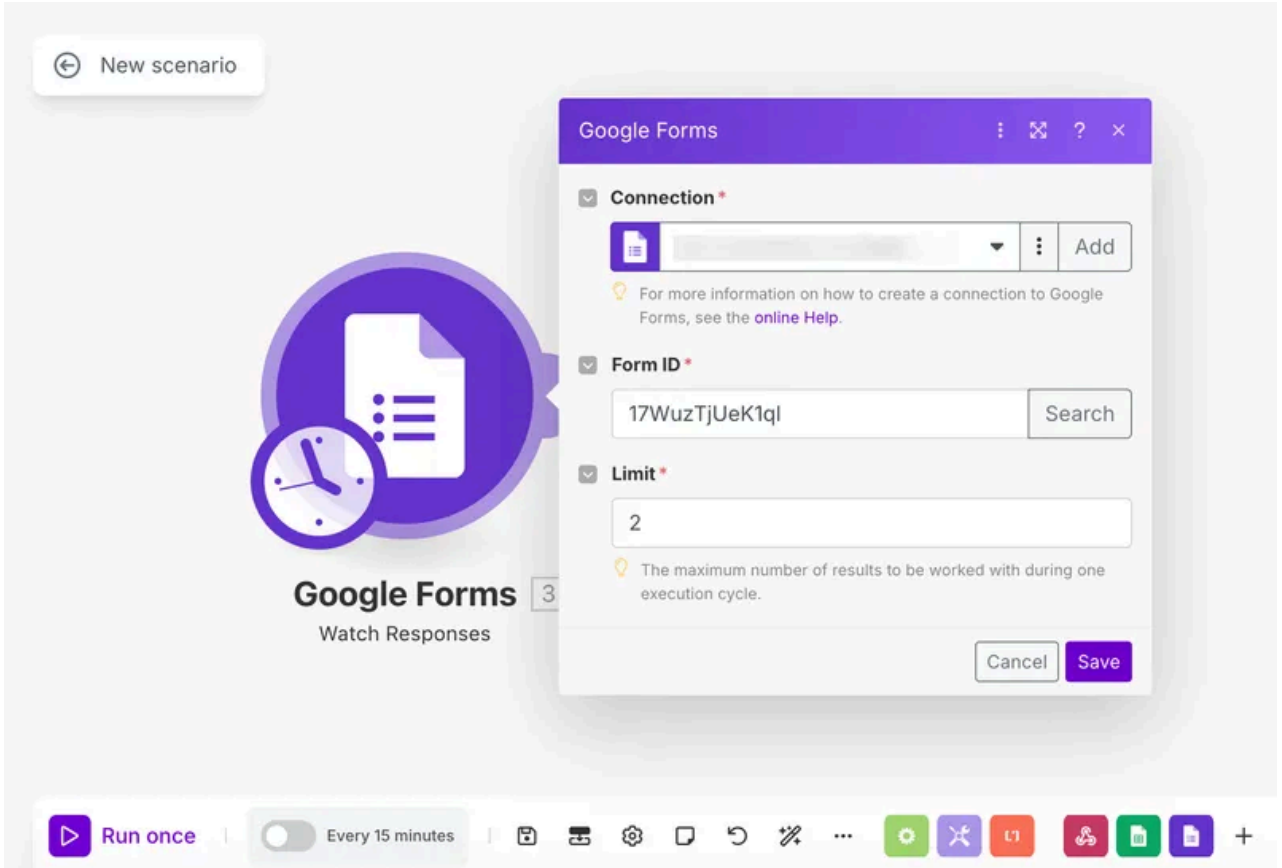
Export a blueprint

To export a blueprint:

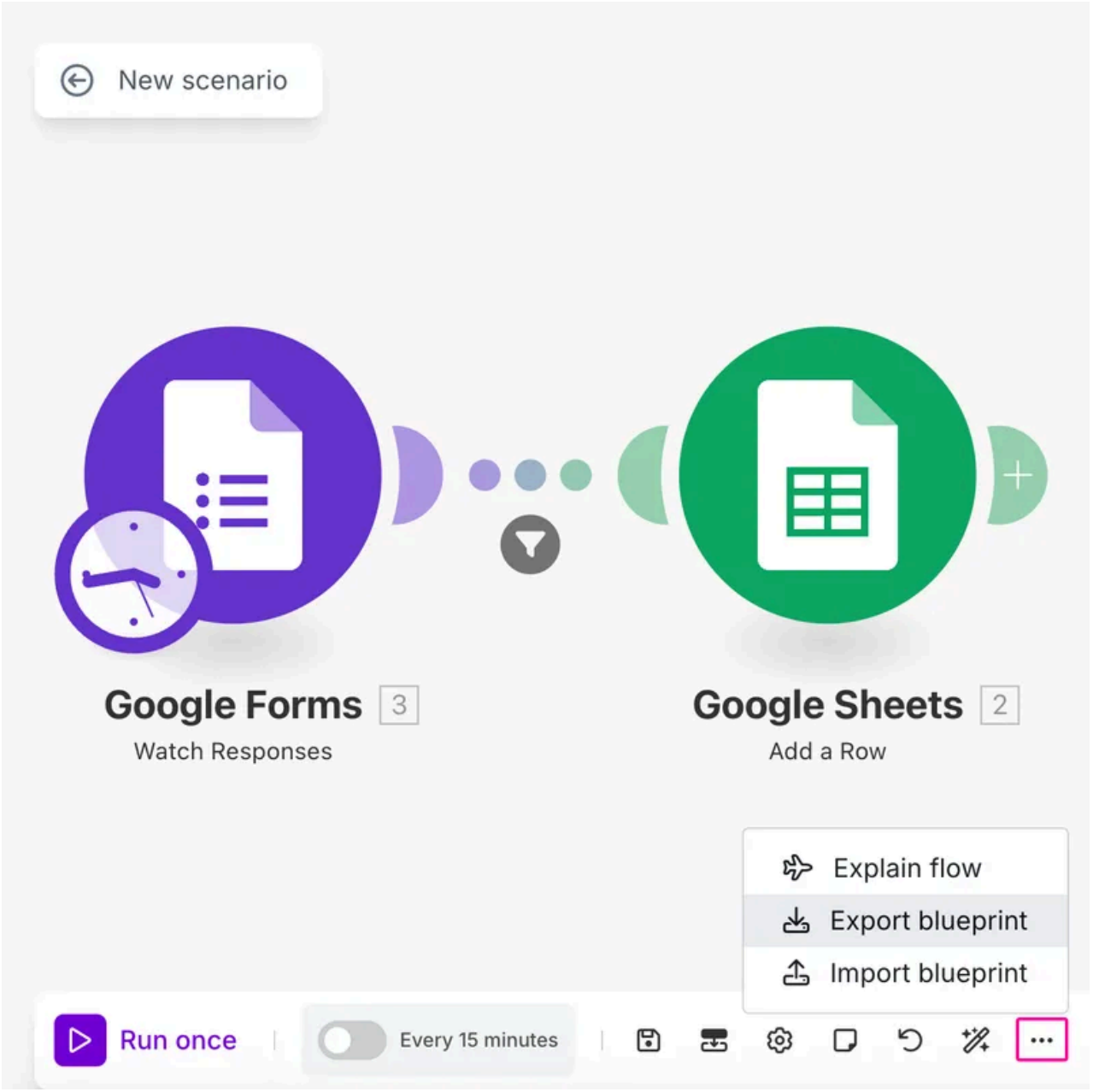
1. Click Scenarios in the left sidebar. Choose an existing scenario, or click **+ Create a new scenario**, if you need to create the scenario first.



2 If it's a new scenario, add modules and **configure the module settings**. If it's an existing scenario, check that all of them are set up correctly.



3 On the Scenario Builder toolbar, click the **three dots > Export blueprint**.



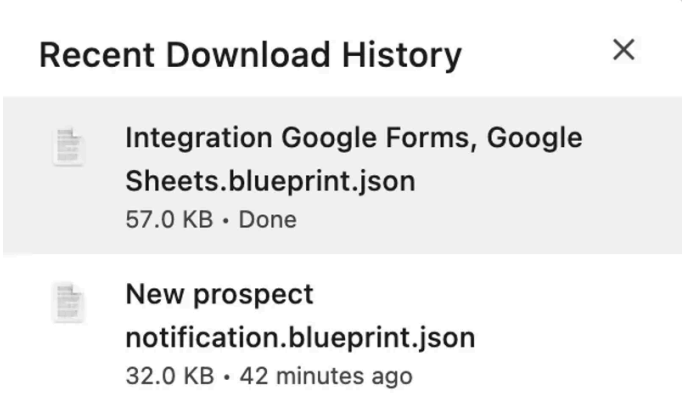
Once exported, the blueprint downloads to your device as a `.json` file.

Integration Google Forms, Google Sheets.blueprint.json


```
{
    "name": "Integration Google Forms, Google Sheets",
    "flow": [
        {
            "id": 3,
            "module": "google-forms:watchResponses",
            "version": 2,
            "parameters": {
                "__IMTCONN__": 8975087,
                "formId": "                                    ",
                "limit": 2
            },
            "mapper": {},
            "metadata": {
                "designer": {
                    "x": 0,
                    "y": 0
                },
                "restore": {
                    "parameters": {
                        "__IMTCONN__": {
                            "label": "                                    ",
                            "data": {
                                "scoped": "true",
                                "connection": "google"
                            }
                        }
                    }
                },
                "parameters": [
                    {
                        "name": "__IMTCONN__",
                        "type": "account:google",
                        "label": "Connection",
                        "required": true
```


The file is typically named after your scenario. For example:

- If your scenario's name is "New prospect notification", the file will be called `New prospect notifcation.blueprint.json.`
- If the scenario has a default "New scenario" name, Make names the file after the modules in the scenario, e.g., `Integration Google Forms, Google Sheets.blueprint.json` .



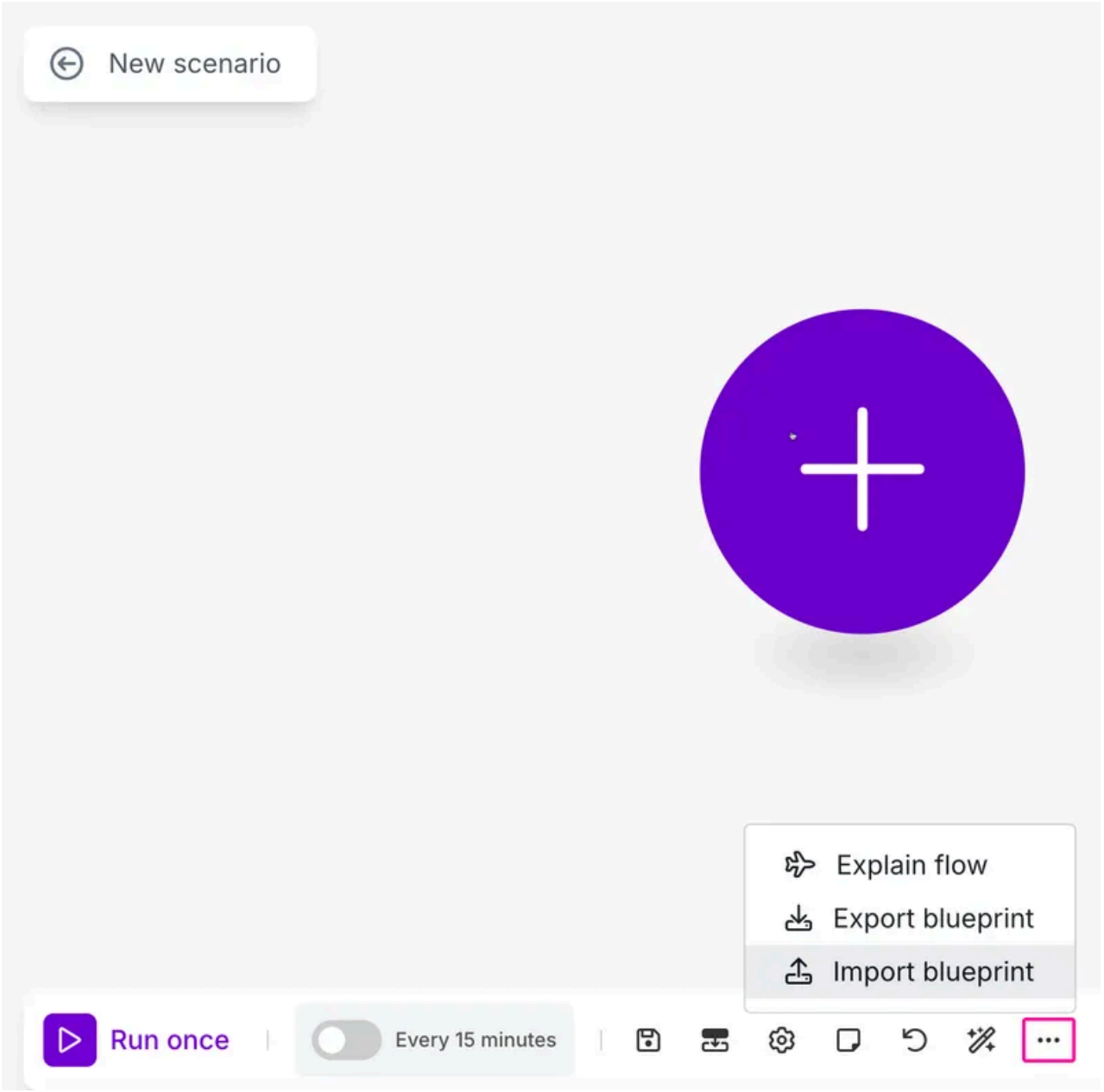
You can now share this file with other Make users or keep it for your own future use.

Import a blueprint

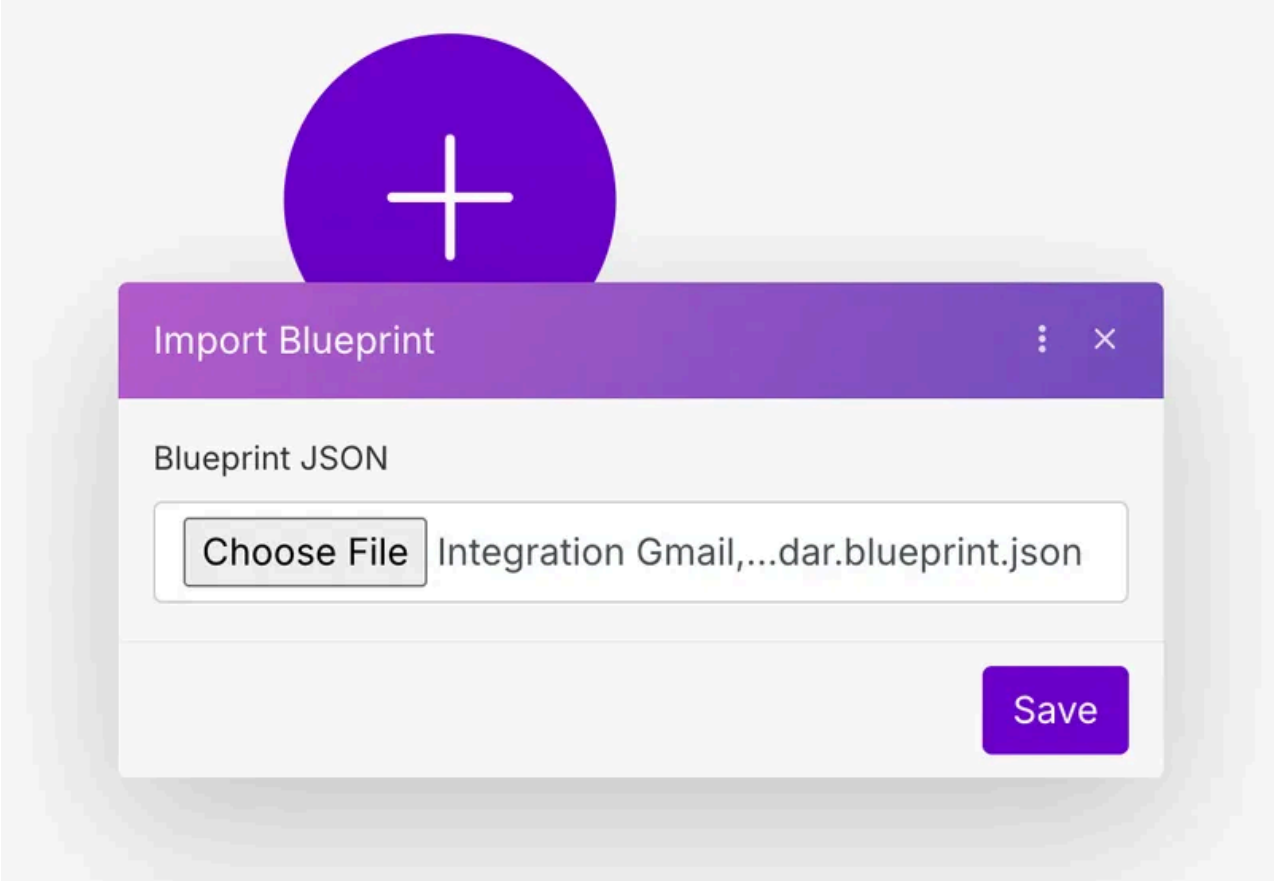
> We recommend creating a new scenario and importing the blueprint there.
>
> If you import a blueprint while editing another scenario that hasn't been saved, all the changes in that scenario will be lost.

To import a blueprint:

1. On the Scenario Builder toolbar, click the **three dots > Import blueprint**.



2. Click **Choose File,** select the blueprint file in .json format, and click **Save**.



The imported scenario appears in the Scenario Builder. You can now configure your own account connections, make any necessary edits to the scenario, schedule it, and click the **Save** icon.

> Check out the **Make Academy Blueprints course** to walk through a use case and see how blueprints work in action.

2.1.8. Scenario templates

Templates are predefined scenarios that you can expand and customize to create new scenarios. You can then **share** these with friends and colleagues.

Benefits of templates

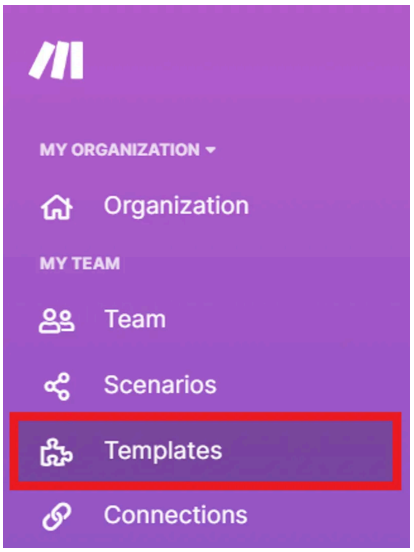
- Creating your scenarios becomes much easier. Instead of starting each new scenario from scratch, you can use a template as a foundation to expand upon.
- Share your templates with anyone.
- Save your templates for future use.

Who can create templates?

Anyone with a **Teams** plan can create templates. See the available **subscription plans**.

Using templates

You can find the **Templates** tab under your **Team** in the left-hand menu.



Search for templates using the search bar in the top-right corner.

Template types

Public templates

A collection of templates created by Make and our partners that features many common integrations. Anyone can use them to create scenarios.

See **Creating a scenario from a template**.

Team templates

> Only available for **Teams** and **Enterprise** organizations.

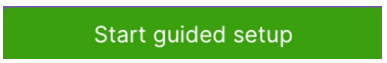
Team templates are those you and your team create. From the team templates screen, you can create, customize, save, and share your templates with others.

Creating a scenario from a template

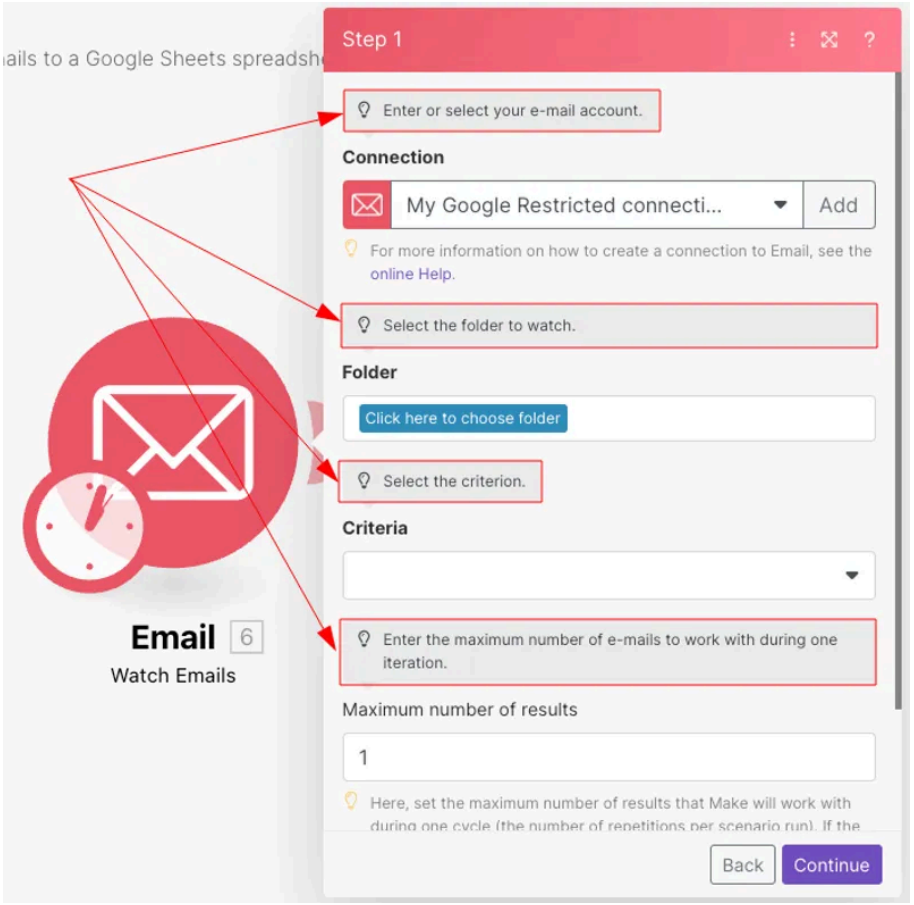
Using a public template

Make creates and maintains a library of templates. You can find it by clicking **Templates** on the left sidebar and looking under the **Public** tab.

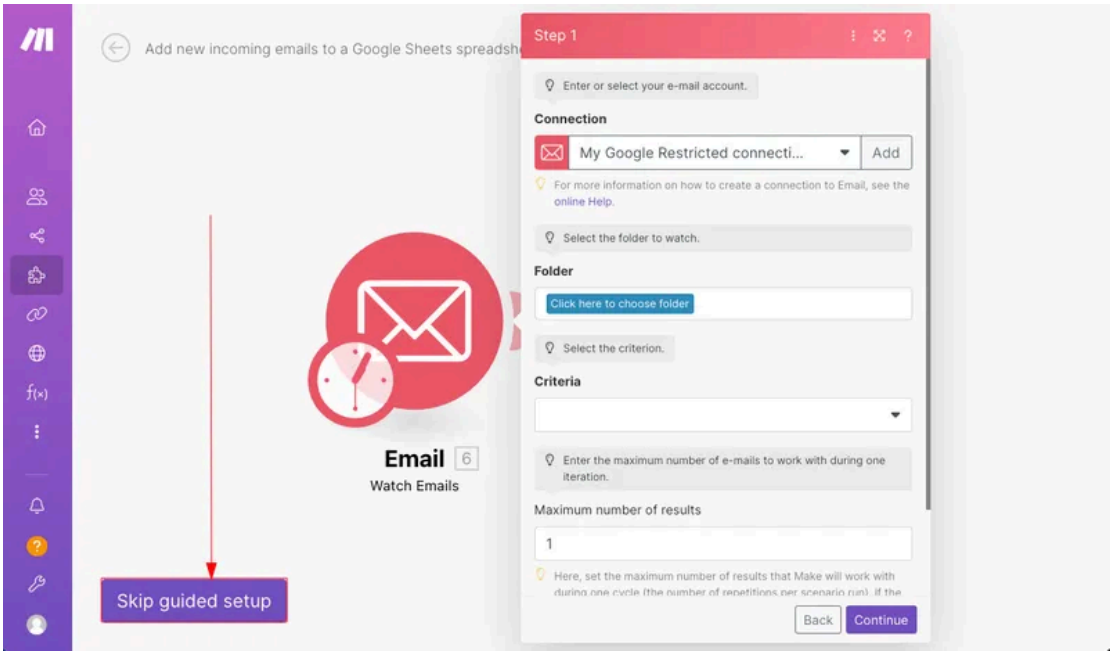
1. Go to **Templates > Public** and click the template you want to use.
2. Choose whether you want a guided setup experience or simply a new scenario from the template.
    1. **Guided setup** - This option includes on-screen prompts to help you configure the template for your needs.
        1. Click **Start guided setup**.



        2. Follow the on-screen prompts to configure your scenario.



3. O*ptional* Click **Skip guided setup** at any time to end the guided setup. You still need to configure all modules but on-screen prompts no longer appear.



2. **New scenario** - This option instantly creates a new scenario in your organization from the template. You need to manually configure module settings, such as **Connections**.
    1. Click **Create a new scenario from template**.

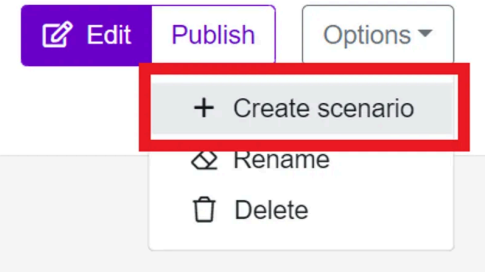
Create new scenario from template

2. Edit your new scenario the same way as any scenario.

3. Name and save your scenario.
4. Test and schedule your scenario.

Using a team template

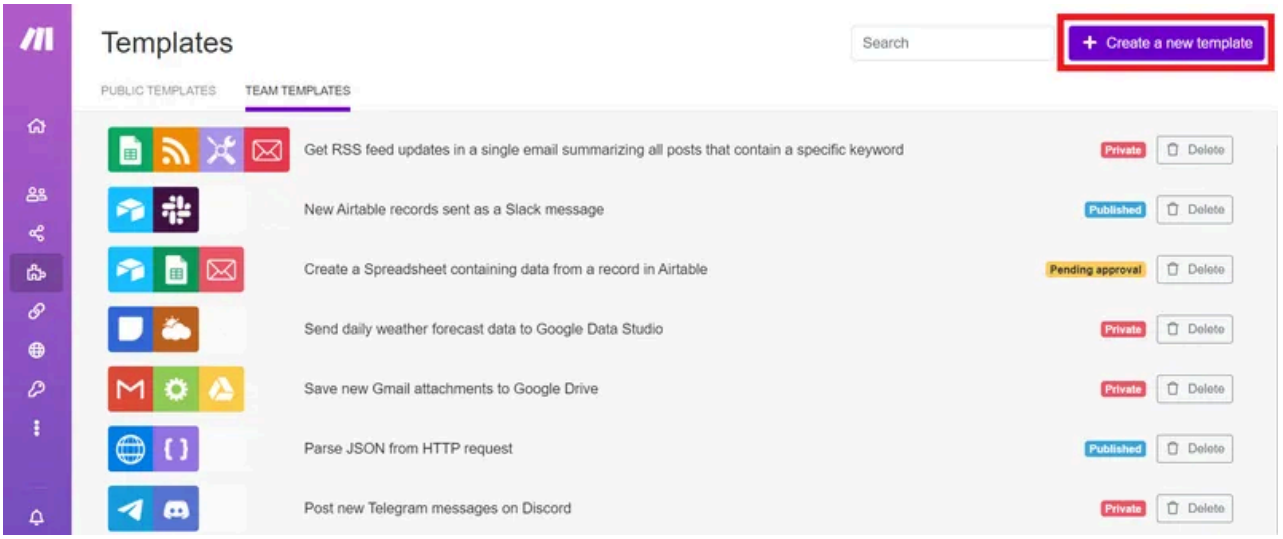
1. Go to **Templates > Team** and click the template you want to use.
2. Click **Options**.
3. Click **+ Create scenario**.



4. Choose which **organization** you want to create your scenario for.
5. Click **Save**.
   You are now ready to edit your scenario.

Creating templates

1. Click **Templates** in the left-hand menu.
2. Click **+ Create a new template**.



The Scenario Builder appears. Use the builder to create templates the same way you create regular scenarios.

> You can only edit your *private* templates.
>
> You cannot edit any *published* templates.
>
> Templates cannot start with a **webhook**.

Renaming templates

When you create a new template, Make names it “New Template” by default.

Edit the name by clicking the title displayed at the top of the page, or follow the steps below:

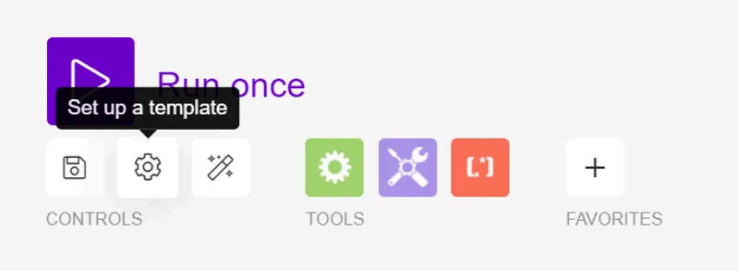
1. Click **Options** in the top-right corner.
2. Click **Rename**.

> You can only rename *private* templates.

Adding a description

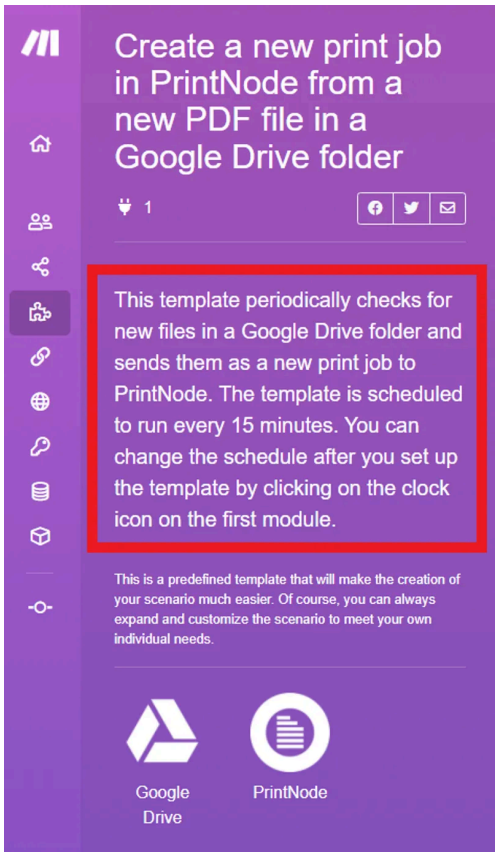
You can include a description of your template to inform others of its functionality.

1. Click the **gear icon** in the **Controls** panel below your template.



2. Add your description in the box provided (Max. 240 characters).
3. Click **Save**.

Your description appears in the template information panel.



Editing templates

Editing a template is similar to editing a scenario with extra options for each field:

- Use in Wizard - checking this box includes the field in the guided setup. Select it for fields that users need to personalize or edit.
- Use as a default value - checking this box makes the value you defined the default value for that field. Select it to include a recommended value that users can edit.

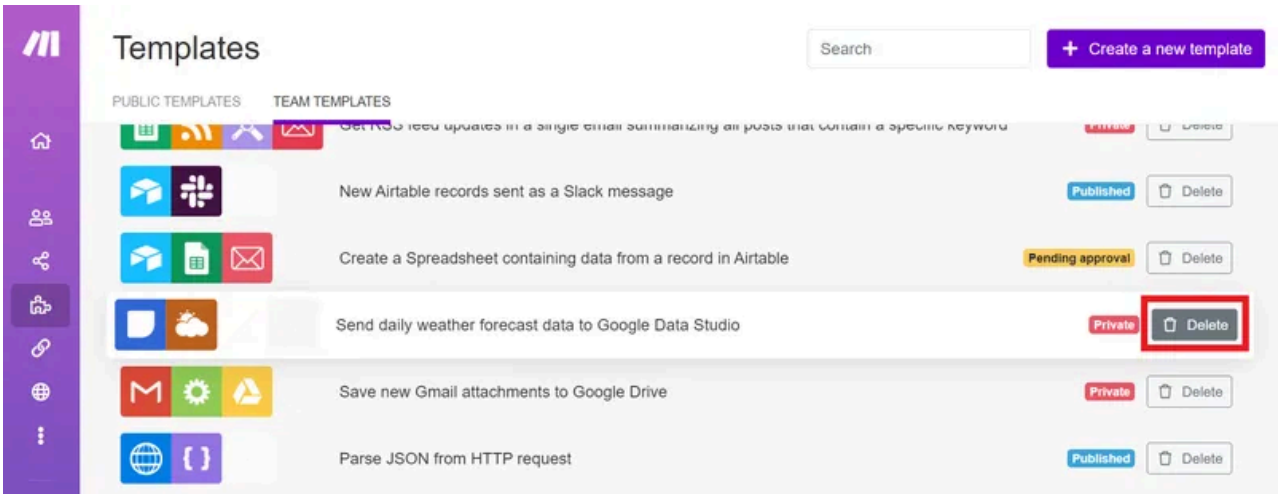
Known issues:

- When adding a module with dynamic, customizable fields, such as **Set multiple variables**, you need to check ***both***:
  - **Use in Wizard**
  - **Use as a default value**
  - Selecting both check boxes ensure that users can see the field when they create a scenario from your template.
- If the **Use in Wizard** and **Use as a default value** checkboxes do not appear:
  - In the **Module settings** window, click OK to save your changes and close the window.
  - Right click the module and select **Settings**.
    The **Use in Wizard** and **Use as a defualt value** checkboxes now appear.

Deleting templates

You can delete any of your *private* or *published* templates.

- From your template list, click **Delete** to the right of any template.



- From a template, click **Options** in the top-right corner, then click **Delete**.

Publishing templates

Publishing

To share your template with others, you must first publish it.

To publish a template:

1. Choose which *private* template you want to share.
2. Click **Publish** in the top right corner.

> You cannot edit a *published* template. You need to first unpublish your template. Then you can edit and republish.

Share a public link

Once you publish your template, you can share it with others.

Click **Share public link** in the top-right corner to copy the link to your clipboard.

> Template recipients cannot change or clone the template.

Requesting approval

You can send your *published* templates for approval to make them available in the public templates library.

- Click **Request approval** in the top-right corner.

The button will now read “Approval pending”.

A Make administrator reviews your template for public use. After approval, Make adds your template to the public templates library.

> If your template contains a custom app, we can't add the template to the public library. You can still share the template using the public link.

2.2. Apps & modules

Learn the basics of apps, modules, and webhooks in these articles:

→ **Introduction to Make apps**

→ **Types of modules**

→ **Module settings**

→ **Update legacy modules with new modules**

→ **Webhooks**

→ **Selecting the first bundle**

2.2.1. Introduction to Make apps

Make is a no-code integration platform that allows you to connect multiple apps and transfer data between them. With the modules in these apps, you can create scenarios that can be scheduled to fit your workflows.

The advantage of Make is that you don't need to be a coder to share data across apps. Read more about how Make works.

What are Make apps?

Make apps are integrations of third-party APIs or Make built-in tools. Each app consists of **modules**.

Types of Make apps

Make divides apps into app types based on their origin and availability:

- verified apps: apps verified by Make developers
- community apps: apps shared on the **Apps Marketplace**
- custom apps: apps made by Make users in the **Apps Platform**

Verified apps

Verified apps are developed either by Make developers or by other Make users. Verified apps made by a third party have to pass the Make app review process and have to be actively maintained.

Verified apps further differ in their availability in Make **pricing plans**.

- standard apps: All Make users can use standard apps
- enterprise apps: Users in organizations with the Enterprise plan can use enterprise apps.

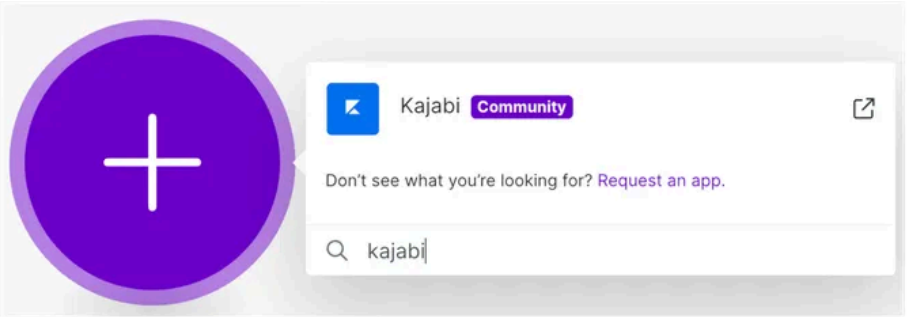
The enterprise apps are in the scenario editor marked with a label:

Greenhouse Recruiting **premium tier 3**

Community apps

Community apps are developed by Make partners and power users who have taken dedicated training. In contrast to verified apps, the community apps don't go through the app reviews to be approved by Make.

The community apps are shared as part of the **Apps Marketplace**. You can check all the community apps on the **Make Apps & services page** when you filter to show the community apps only. In the scenario editor, you can identify the community apps by the **Community** label:



Community apps may require an additional purchase to use. Make does not earn money or commission on these purchases. You purchase the community apps directly from the developer.

Custom apps

Custom apps are developed by other Make users. You can create a custom app anytime. To learn more about creating custom apps, check the **custom apps documentation**.

Custom apps are available to Make users based on their visibility:

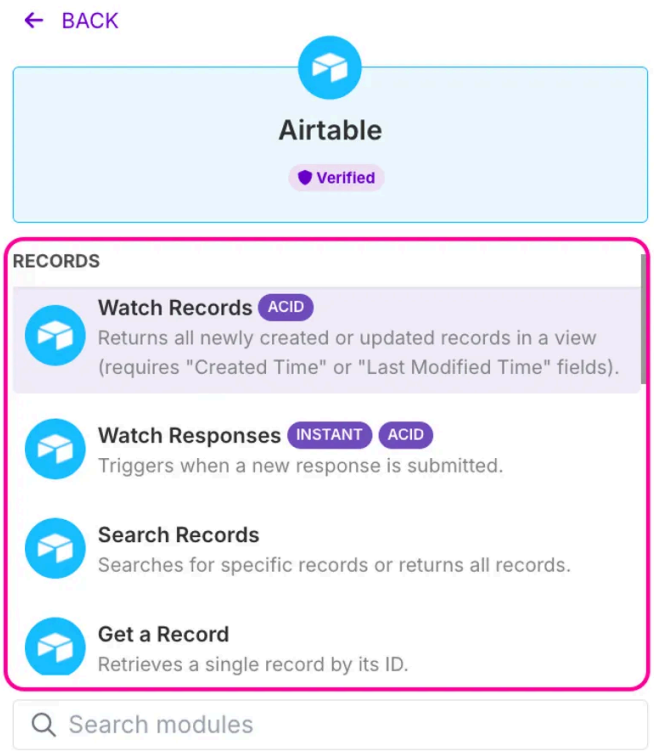
- private apps: The custom app developer and users in all organizations they are part of can use the app
- invite-only apps: Access to the app is on an invite-only basis. The app developer has to share an invite link with you.

2.2.2. Types of modules

In Make, a module is a building block you use to create a scenario. Think of it as bricks that you put together to automate your processes.

Each module performs a particular action, such as retrieving data from a service, creating or updating a record, downloading a file, or searching for specific data based on certain conditions. For example, your scenario can have three modules: one module watches for new customer data in a CRM, the second module converts the data into another format, and the last module sends that information to a different service.

You can find all available modules in the app list when selecting a specific app in the Scenario Builder.



Understanding each module type helps you navigate Make more easily, build advanced scenarios, and **optimize operations** while reducing costs.

In this article, you'll discover the different types of modules in Make and how to use them in your scenarios. If you're an advanced user, be sure to check out **the section on tips and examples** for more in-depth insights.

Triggers, Searches, Actions, and Universal modules

There are four types of modules in Make based on what they do with data:

- **Triggers**
- **Searches**
- **Actions**
- **Universal modules**

Triggers

A trigger tracks changes in a service and brings them to Make so you can use the data in your scenario. A trigger only shows new data from your service account. Each time a scenario processes trigger data, it's data that hasn't been processed before in that scenario.

For example, a trigger might detect when a new record is created, or an old record is deleted.

You can add a trigger only once in the scenario as the first module. This ensures that the trigger initiates the scenario by detecting the relevant changes and pulling in the data for further processing.

> We highly recommend starting your scenario with a trigger. Still, you can start your scenario with any possible module. See the **Advanced tips and examples section** to learn more.

Trigger types

There are two types of triggers in Make:

- **Polling triggers**
- **Instant triggers**

Polling triggers

A polling trigger checks for new data in a service account since the last scenario run, based on the scenario's **schedule**. You can recognize this trigger type by its name, which often starts with "Watch": "Watch a record," "Watch a row," and so on.

The polling trigger sends a request to the service. If new data exists, the scenario runs, and you see the data in the module's output as bundles. If not, you see no bundles.

A scheduled scenario run that checks for new data in a service, but returns no data, is called a check run. You can view check runs in the **scenario history**, located in the **History** tab of scenario details.

> Click **Schedule settings** in the scenario toolbar to set how often a trigger module runs.

You can choose a starting point for the module to check for new data in a service. Learn more in **Selecting the first bundle for polling triggers**.

Instant triggers

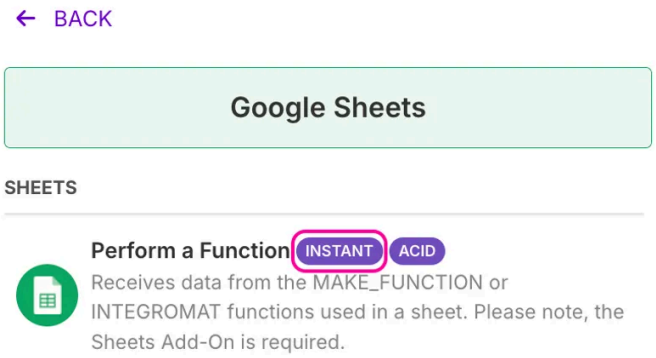
With an instant trigger, the service notifies Make when new data arrives, starting the scenario immediately. In essence, an instant trigger is **a webhook**: a link that the service uses to send new data to Make.

When you add an instant trigger, Make asks you to create a webhook. Some services may require you to create a connection beforehand.

> If you add an instant trigger, you automatically set the scenario schedule as *Immediately as data arrives*.
>
> You can reschedule a scenario if needed. In this case, a webhook receives data as soon as it arrives and stores it in a queue until the next scenario run according to a new schedule.

You can recognize an instant trigger by a lightning icon and an **instant** tag:





To understand how instant and polling triggers differ, think of buying plane tickets. While a polling trigger is like regularly visiting an airline's website to check for discounts, an instant trigger is like the website notifying you when discounts are available.

Searches

A Search module helps you to get specific data from a service such as records, profiles, or other objects.

Search modules are flexible. You can add as many as you need and place them anywhere within your scenario.

Search types

There are two types of search modules in Make:

- A Search module allows filtering and using search queries to find specific data. For example, Search records, Search rows, etc.
- A List module retrieves all data from a service without any filtering. Make returns all objects that you have in your service account. For example, List records, List sheets, List customers, etc.

In most cases, when using searches, you can get up to 3,200 objects per module run. Also, additional third-party API limits may exist. See the **Module limits section** below for more details.

Actions

An Action module processes the data retrieved from a service. It's one of the most commonly used modules in Make.

You can include as many as you need and position them anywhere in your scenario.

There are the following types of actions in Make:

- Get
- Create
- Update
- Delete

When using **Delete** modules in Make (e.g., deleting records, tasks, or other items), be aware that some services may not support recovery of deleted items for actions performed through the API, or in general.

This means that deletions may be **permanent** and **irreversible**, and you won't be able to recover items deleted using the **Delete** module in Make.

There are also some actions that are specific to the service, such as pin, save, or download.

Unlike Search modules that provide information about all objects or objects that match module filters, a Get module helps to receive information about one specified item. You should specify an object ID to get information about it. Read more in the **Advanced Tips and Examples section**.

Universal modules

A Universal module allows you to make a custom API call to a service when Make doesn't provide a pre-built module for an API endpoint you need. This module, usually called the *Make an API Call* module, is available for most services. You may need to refer to the service API documentation to see a full list of available API endpoints.

You can add as many Universal modules as needed and place them anywhere in your scenario.

> If Make doesn't have an app for a service you need, you can use the **HTTP app** to make API calls.

Apps and tools

Modules can be split into two groups based on whether they need to connect to a service or use a third-party API. There are two types:

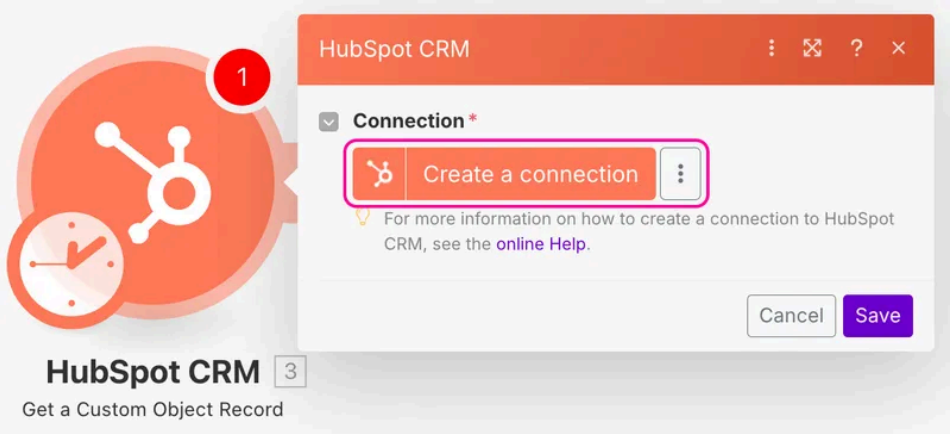
- **Apps**
- **Tools**

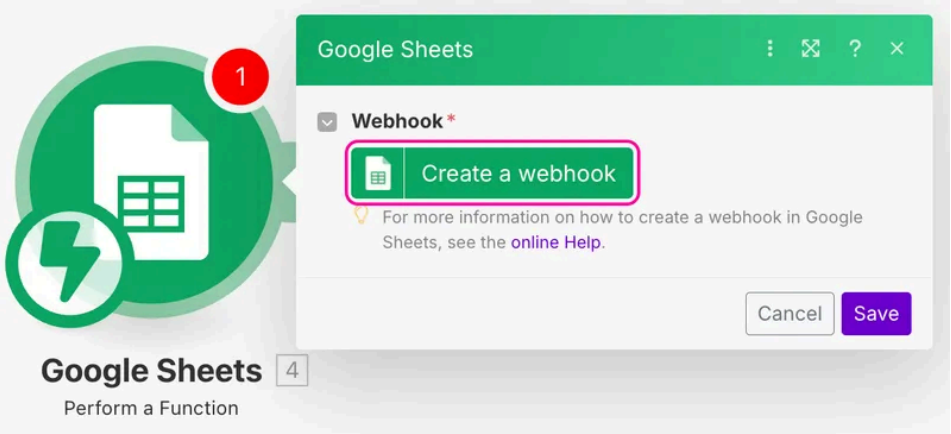
Apps

When adding modules of an app, you need to create a connection to start working with them. Each module serves a specific action. For example, to get data, create a record, or delete a profile in your service account. You can associate a certain app module with a specific service API endpoint.

LinkedIn, Google Sheets, HubSpot CRM, Trello and **many more** are apps in Make. When you add a module, you see the **Create a connection** or **Create a webhook** buttons. In the **Connections section**, you can learn more about connections in Make.

> Some apps don't require creating a connection, but they still use a third-party API. You may face third-party API limits when working with these apps. See the **Module limits section** below for more details.

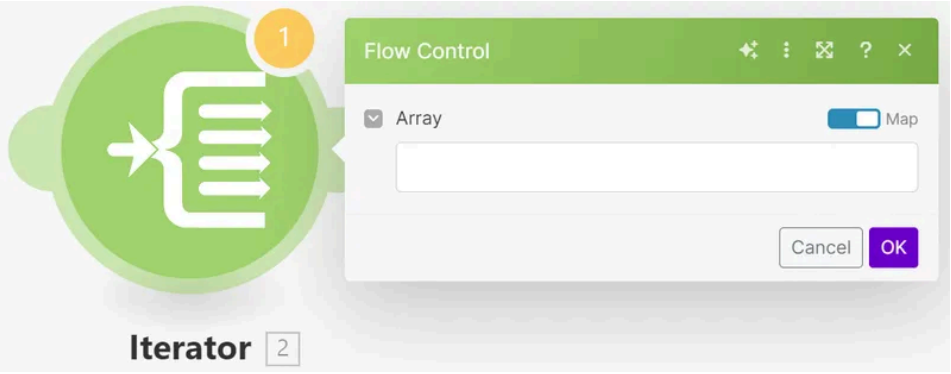


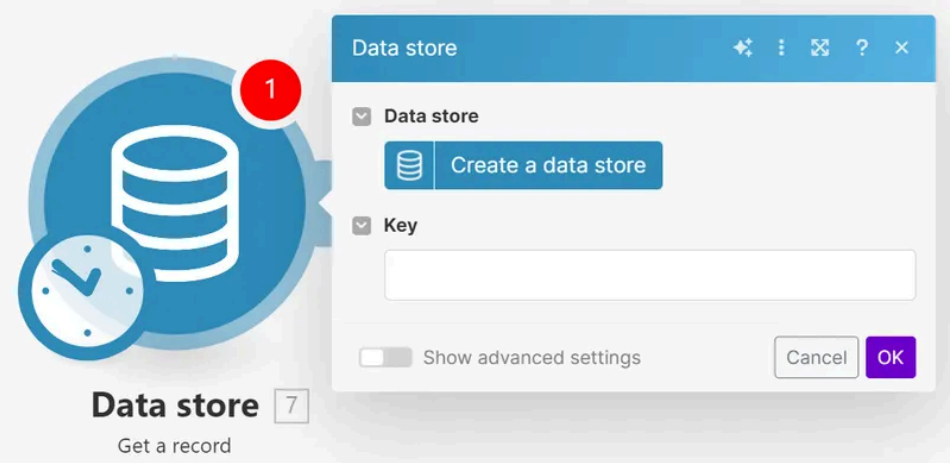


Tools

With tools, you don't need to set up a connection nor use a third-party API. Instead, you enter your data or customize module settings, and the module is ready to work. Examples include Iterator, Aggregator, Data store, Compose a string, Set variable, and many more.

These modules are ready to go as soon as you add them. However, some tools might require some setup before you can use them. For example, Data store might ask you to create a data store within Make.





Advanced tips and examples

In this section, you can find advanced tips and practical examples for using modules in Make.

Module limits

While using modules, you might run into limits that could cause errors in your scenario.

Most Searches can only process up to 3,200 objects or 5MB of data per run. You only see the first 3,200 objects searched or as many as fit within the 5MB limit, even if there are more. Some modules only have a data size limit: each module can receive or process up to 5MB of data per run.

Keep in mind that the service's API limits may apply. For example, if the API allows only 1,000 objects, that's all you will receive no matter which type of modules you use. You can refer to the service API documentation to learn more about its limits.

Here are two best practices to avoid errors and make sure your scenario runs smoothly:

- When available, use the **Limit** field to define how many objects you want to receive during one run.
- Use a Search module if you need specific data. Search modules let you narrow down results with filters and search queries.

If you use the **Limit** field, it's essential to set the right limit for the data being returned per one run. For example, with a **polling trigger**, you need to find a balance:

- If the limit is too low, Make will only process a part of data, and the rest will have to wait until the next run. This could cause delays in data processing if a lot of new data

arrives.

- If the limit is too high, you might retrieve too many objects in one run. This also increases the risk of hitting the 40-minute scenario run time limit if you need to perform many actions on each object.

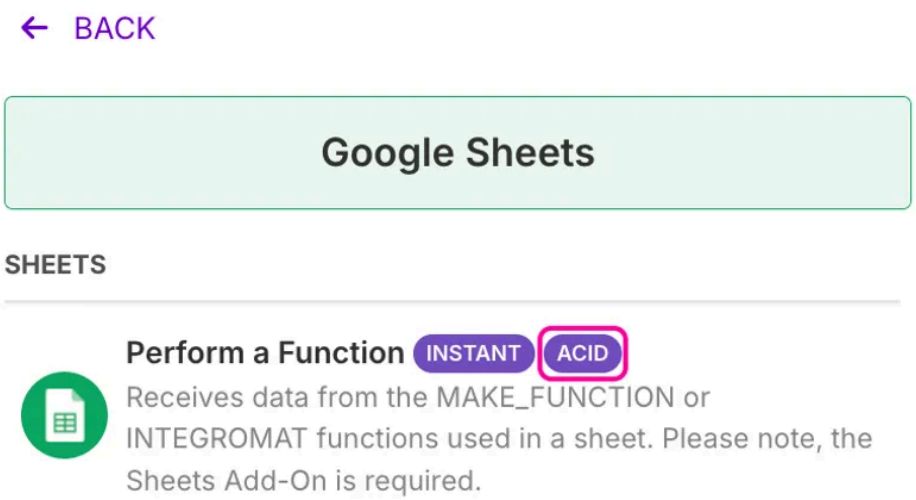
We recommend setting the limit according to the number of new objects you plan to get for each scenario run.

ACID modules

Modules tagged as ACID support rollback. Make will undo all the actions an ACID module takes if an error occurred in subsequent modules.

Imagine your ACID module receives two bundles. The next module processes the first one but encounters an error with the second one. In this case, Make rolls back the ACID module as if nothing was received. On the next run, the ACID module gets the same two bundles again. The next module, being non-ACID, will try to process both bundles once again.

You can recognize the ACID module by the ACID tag:



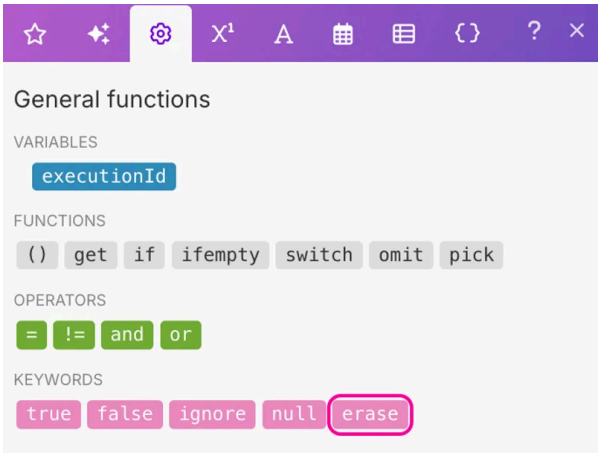
Difference between Search and Get modules

Unlike Search modules that provide information about all objects or objects that match module filters, a Get module helps to receive information about one specified object. You should specify an object ID to get information about it.

For data, this means Search or List modules usually return several bundles because there are multiple objects to retrieve. Get modules, however, always return a single object, which means only one bundle.

Using Update modules

With an Update module, you can perform three actions:

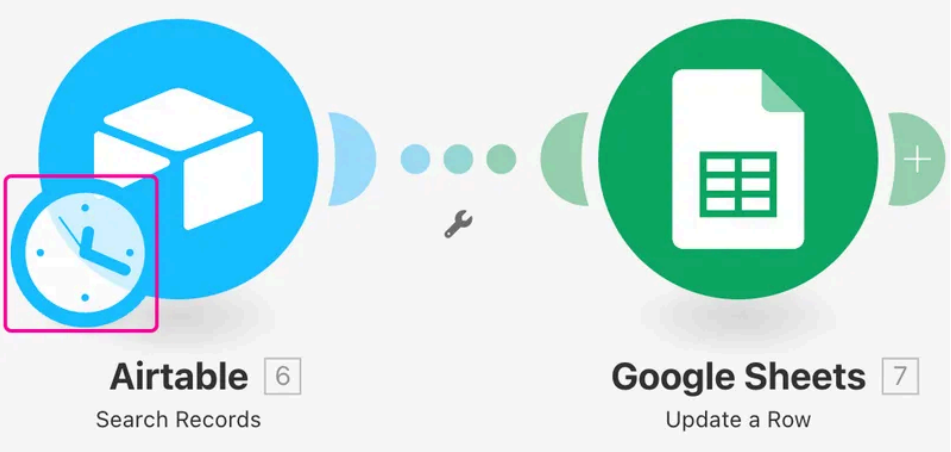
- Erase content using the **erase** keyword in a content field. You can find it in the **General functions** tab.
    - 

> Use this keyword only if you make a PUT API call. If you perform a PATCH API call, leave a space in a content field.

- Leave content as it is by leaving a content field empty.
- Overwrite content by entering new data in a content field.

Using different modules as a trigger

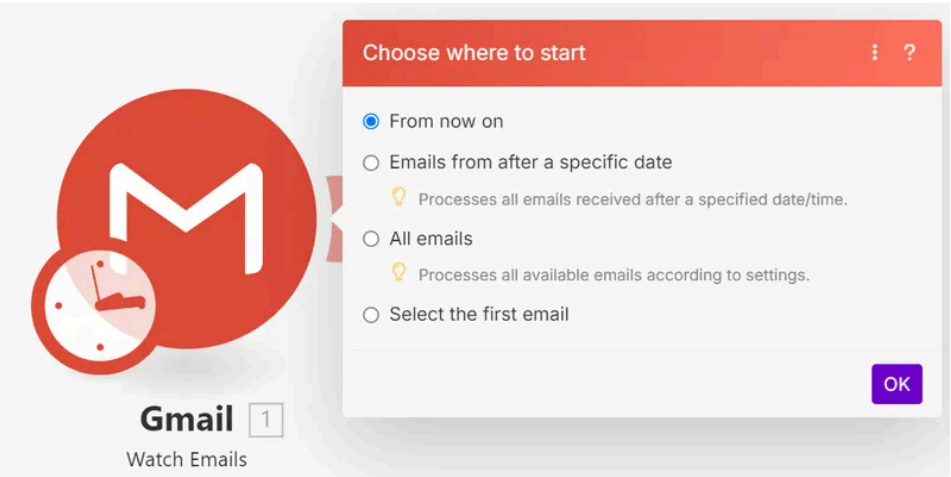
While we recommend starting a Make with a **Watch** module, you can also begin with a **Search** module or **Action** module, depending on your goal. For example, if you need specific data, a **Search** or **List** module can act as a polling trigger starting a Make according to the Make schedule.



> If you make a non-Watch module a polling trigger, you cannot add a Watch module afterward.

Selecting the first bundle for a polling trigger

You can decide where to start to track changes made in the service. To do it, right-click the module and then click **Choose where to start**.



Typically, you have the following options:

- The current moment
- A specific date
- A specific ID or a record, or an email
- The first record or email

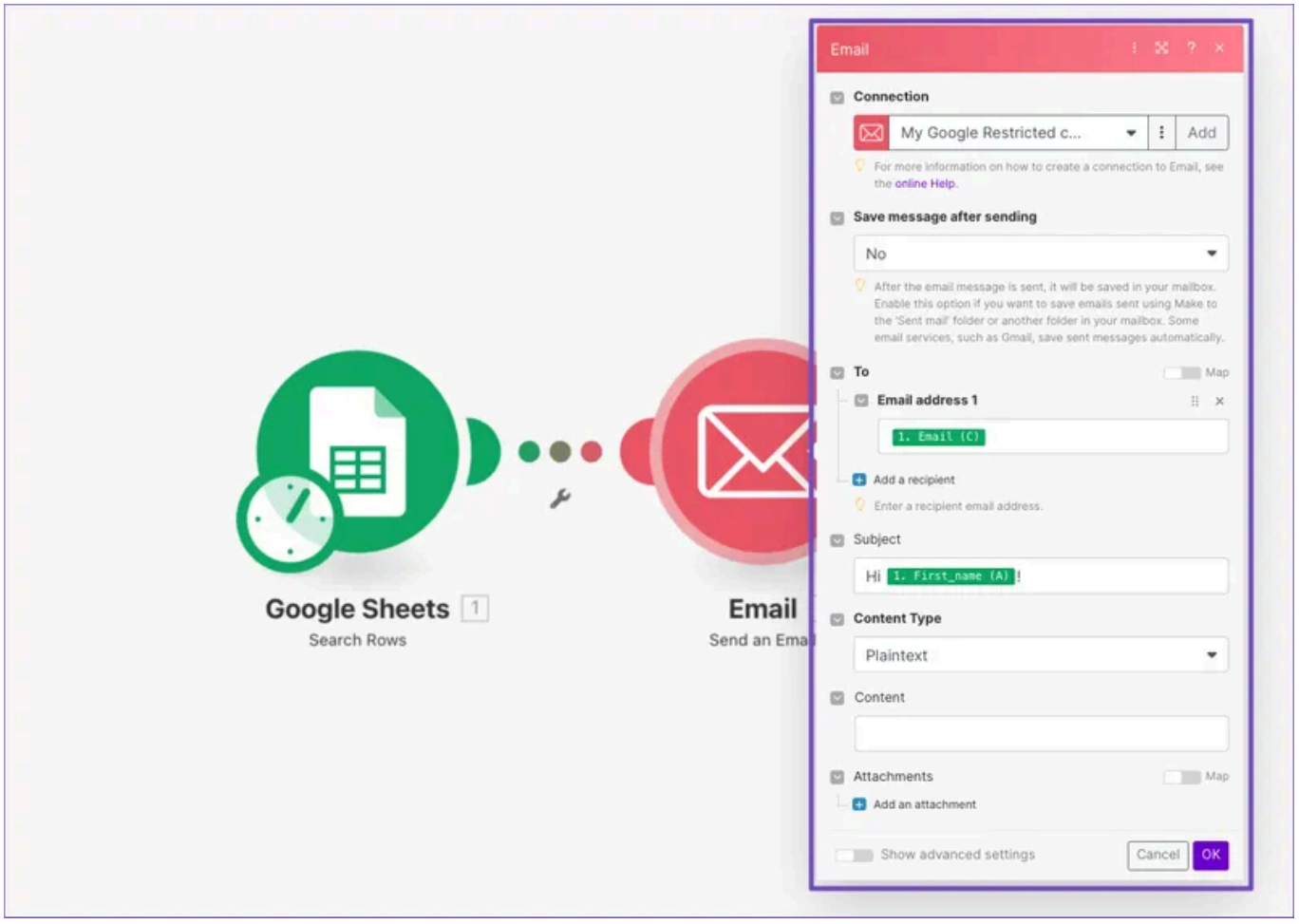
Settings may vary depending on the app.

The option that you select in the **Choose where to start** menu only applies to the first run of the module. Subsequent runs track changes that occurred in the third-party app since the previous run.

2.2.3. Module settings

For each module in your scenario, you need to configure its settings - the fields you see when you click on the module. These fields define the data you want to receive or send to an application, the information you want to pass between modules, and specific details regarding what actions each module should take.

For example, the **Dropbox** modules require you to specify the target folder for uploading files. For the **Email** modules, you need to enter the email address to which emails should be sent. Every setting is unique for each module.



The module settings open automatically when you add a new module to your scenario. You can also open or re-open the module settings in the **Scenario editor** by clicking on the module.

Connection settings

In almost every app's module settings, you will find connection settings. This is where you create a connection to your user account for the given app, and is the first step when adding a new module to your scenario. Some fields you might see in the connection settings include Username, Password, API Key, or Client Credentials. For more information and instructions on how to create a connection to any of your accounts see **Connecting to services**.

Required and optional fields

Some fields in the module settings are required, while others are optional. You can identify which fields are required by their name written in **bold**. If you try to save your module settings without filling in a required field, you will receive an error. Your scenario cannot run unless all required fields are filled in.

In each field, you can either type the appropriate text into the field or fill it in by **mapping an item** from another module in your scenario.

> For some parameters (mostly arrays with several fields that Make retrieves depending upon the selected connection) you can see a **Map** toggle. Switching on the **Map** toggle brings up a text field where you can **map items** from the preceding modules.

Standard fields

When you open the module settings, you will see all of the module's standard fields. These include the most frequently used fields for the module, and are typically the default items you see when going to the website of the service you are using. Standard fields can be both optional and required.

Advanced fields

Some modules also include advanced fields that can be found in the module's advanced settings. These fields include more complex items that may require specific technical knowledge to use. To open the advanced settings, switch on the **Show advanced settings** toggle.

Data types

Each field in the module settings accepts a specific data type, signifying what format of information you can enter or map into the field. This can be text, date, number, etc. You can identify a field's data type by hovering over the name of the field. For more information, see **Item data types**.

2.2.4. Update legacy modules with new modules

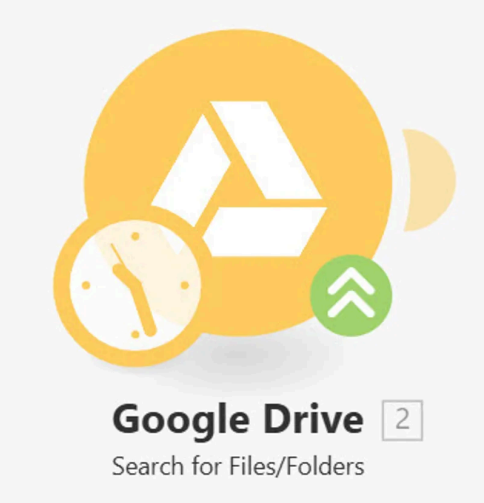
Most Make apps rely on external services, particularly their APIs (Application Programming Interfaces). As services evolve, so do their APIs. When services release a new API version, their Make app eventually becomes deprecated and shuts down. The result is a legacy (outdated) module in Make.

Make regularly implements new app versions that align with services' latest APIs. To keep your scenarios running, make sure to update legacy modules to the latest versions.

Here's a step-by-step guide to updating legacy modules in Make, using Google Drive as an example.

1 Step 1. Identify the legacy scenario

A green double arrow icon (below) indicates a legacy module:

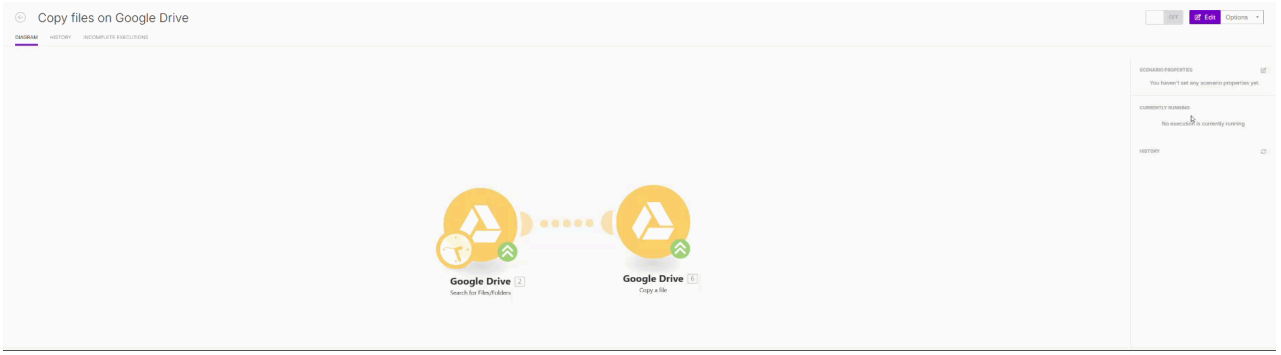


2 Step 2. Clone the legacy scenario

Clone the scenario to make a copy. You will use this scenario from now on.

1. In the scenario diagram, open the **Options** drop-down menu in the top-right corner.
2. Select **Clone**.
3. In **Name of new scenario**, name the copied scenario.

4. To enable the scenario to pick up where its old version left off, select **Yes** in **Keep the states of any modules the same as those being duplicated**.
5. Click **Save**.



> We recommend opening the legacy scenario in a new tab to easily compare its configuration settings with those of the new scenario.

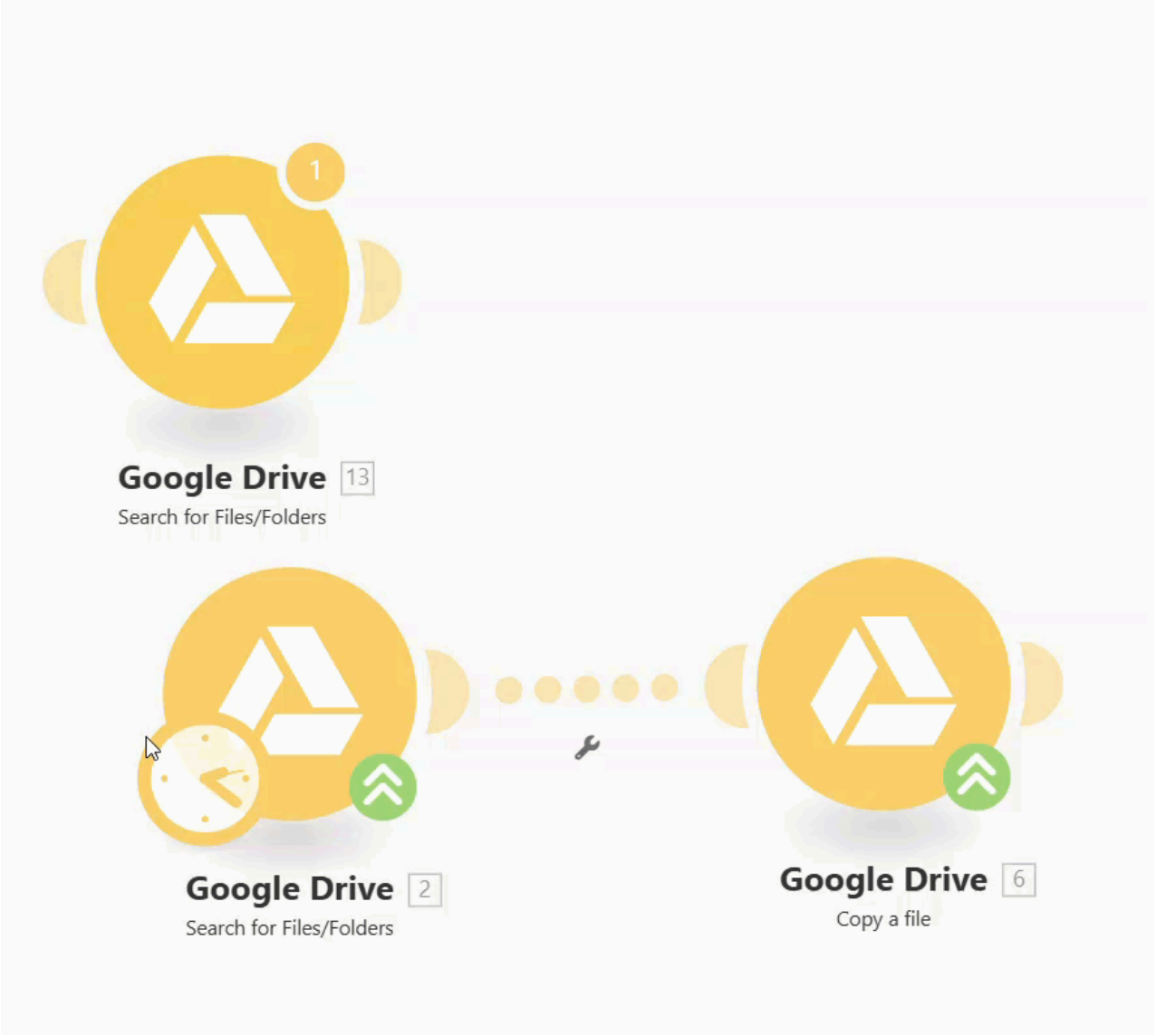
3 Step 3. Access and select new modules

1. In your new (copied) scenario, choose the legacy module you need to upgrade and click on its green double arrow icon.
2. Click **Show me new modules** and select the corresponding module. The new module will appear in the scenario builder, but remains unconnected.

> In the new modules list, some modules may no longer be available. In this case, choose the next-best fit.

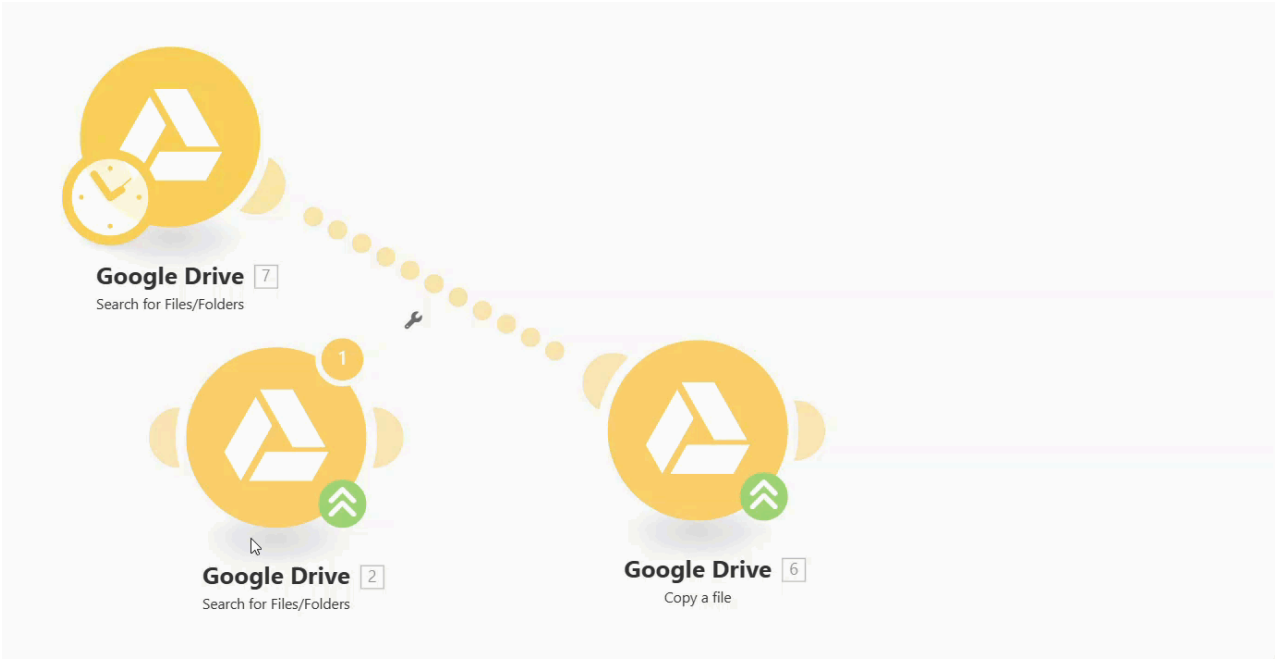
4 Step 4. Update the trigger module

1. Move the clock icon from the legacy module to the new module.
2. Unlink the legacy module and subsequent module, and link the new module with the desired module.
3. Set the values of the new module. Keep in mind that some fields may be different.



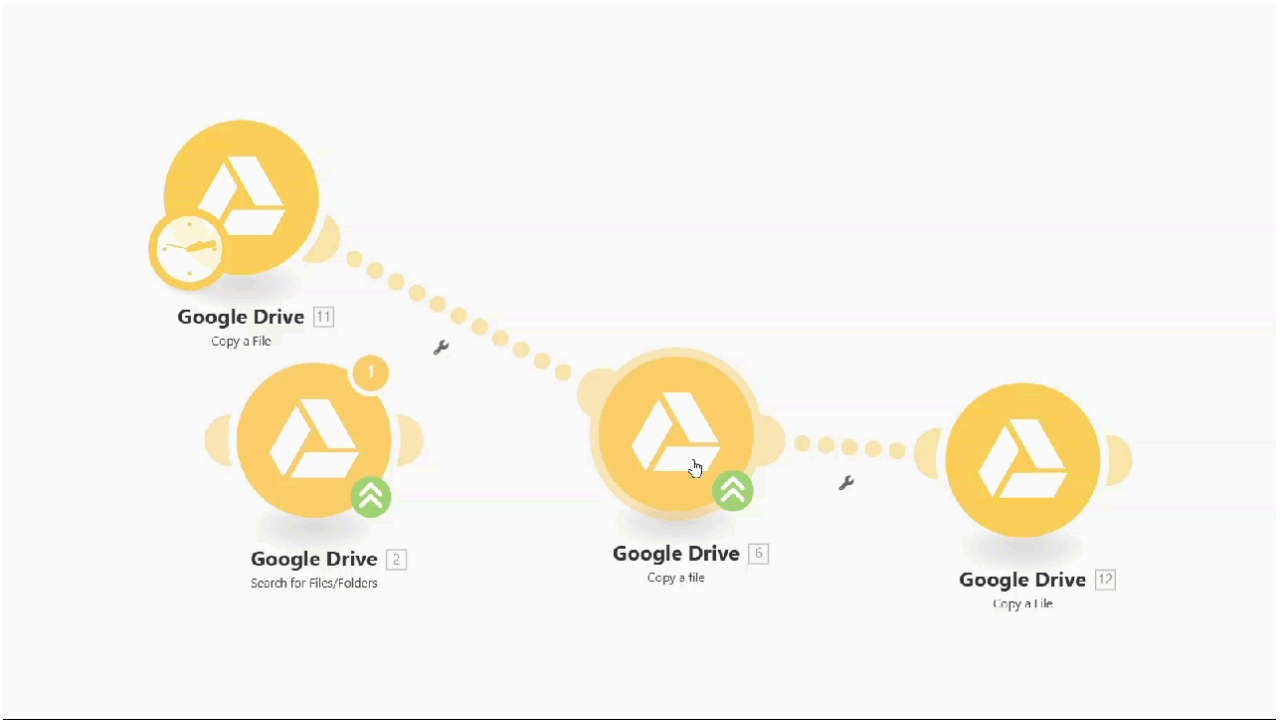
5 Step 5. Update other modules

1. Update the next legacy module, following the same process outlined in Step 3 (Access and Select New Modules).
2. Connect the new module to (after) its legacy module.
3. Set the values of the new module. If any items are missing in the mapping panel, see our **Mapping** documentation.
4. Repeat this process for any remaining legacy modules.



6 Step 6. Clean up and finalize

1. Remove all legacy modules from your new scenario with right-click and **Delete module**.
2. Click **Save** to save your scenario.



You have now updated your legacy modules to their newest versions. This process ensures your scenarios continue to run properly as external services evolve their APIs.

2.2.5. Webhooks

Webhooks allow you to send data to Make over HTTPS. Webhooks create a URL that you can call from an external app or service, or from another Make scenario. Use webhooks to trigger the execution of scenarios.

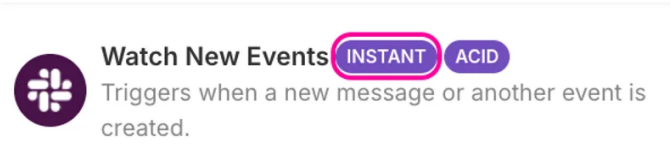
Webhooks usually act as instant triggers. Contrary to scheduled triggers, which periodically ask a given service for new data to be processed, webhooks execute the scenario immediately after the webhook URL receives a request.

Make supports the following types of webhooks:

- App-specific webhooks listen for data coming out of a specific app, also called instant triggers.
- Custom webhooks allow you to create a URL to which you can send any data.

Creating app-specific webhooks

Many apps provide webhooks to execute scenarios whenever a certain change occurs in the app. These are called **instant triggers**. Instant triggers are marked with the label **INSTANT** in the list of an app's modules:



If an app does not provide webhooks, use **polling triggers** to periodically poll the service for new data.

Creating custom webhooks

To create a custom webhook, you must use Make's Webhooks app. You can find information regarding how to set up the Custom webhooks module in the **Webhooks** app documentation.

Scheduling webhooks processing

By default, when Make receives data on a webhook, your scenario executes immediately. If you don't want to run your scenario immediately after a webhook receives data, you can schedule your scenario to process all webhook requests periodically.

1. Edit the scenario which is triggered by your webhook.
2. Edit the scenario schedule settings or edit the schedule settings of the webhook module.
3. Set up your desired schedule.

When a scheduled webhook receives data, Make stores the data in the webhook's queue. The whole queue is then processed every time your schedule criteria are met.

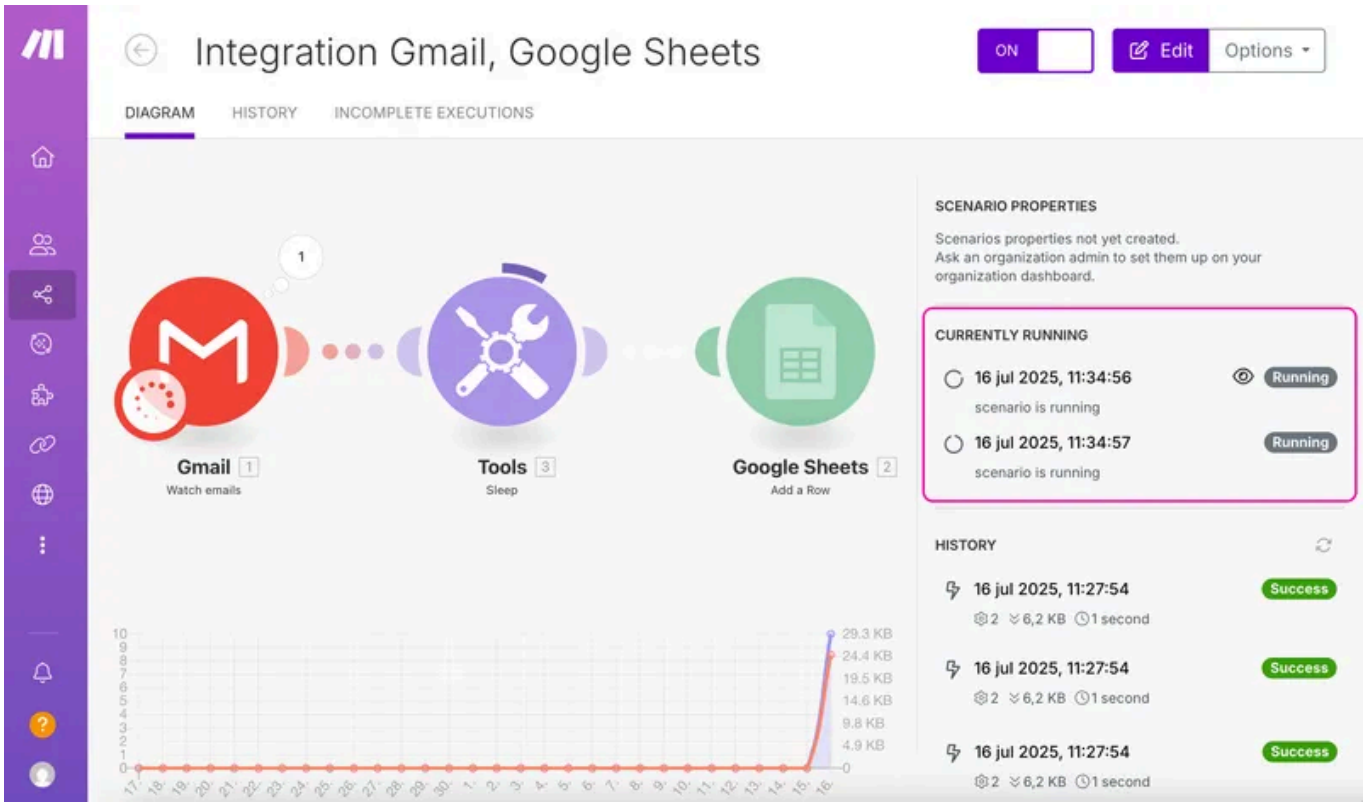
How Make processes webhooks

When a webhook receives a request, the system stores the request in the webhook's queue. Each webhook has its own queue. Go to the **Webhooks** section in the left menu to view all webhooks and their queues.

Parallel vs. sequential processing

If you are using instant webhooks, Make starts processing each request immediately as the request is received. By default, scenarios **with instant webhooks are processed in parallel**. Even if a previous scenario execution is still being processed, Make does not wait for its processing to complete.

You can inspect all running executions in the scenario detail. Click an item in the list of running executions to view the graphical representation of that particular execution. The execution that is currently displayed is marked with an eye icon.



To turn off parallel processing, open the settings of your scenario and select **Sequential processing**. With sequential processing enabled, Make waits until the previous execution is complete before starting the next one. Also, use sequential processing when you need to process your webhook requests in the order that they came in.

Processing scheduled webhooks

If you are using scheduled webhooks, requests accumulate in the queue until the schedule criteria are met. When schedule criteria are met, Make processes the queued requests based on the *Maximum number of results* that you set for the webhook.

For example, if your scenario is scheduled to run every hour and your *Maximum number of results* is set to the default value of 2, Make processes two items from the queue every hour. If your webhook queue is filling up with requests, increase the *Maximum number of results* or adjust the schedule to execute the scenario more often.

Instant trigger modules have the *Maximum number of cycles* parameter instead of the *Maximum number of results*.

Set the *Maximum number of cycles* in the **instant trigger** modules to get the same data processing behavior as is with webhooks and the *Maximum number of results* parameter.

Webhook response module

The Webhook response module is typically used to send a response back to the service that triggered the webhook, confirming receipt or providing additional information. **Without** this module in your scenario, the default responses are:

	HTTP status code	Body
Webhook accepted in the queue	200	Accepted
Webhook queue full	400	Queue is full
Rate limit check failed	429	Too many requests

Learn more about setting up a rate limit **here**.

You can further customize this response with the **Webhook response module**.

If you place the Webhook response module in the middle of a scenario and an error occurs in a subsequent module, the scenario will not be deactivated. This can result in increased operational consumption as you will not receive notification of the error.

If you place the Webhook response module last, you will be notified if any error occurs at any point in the scenario.

Webhook queue details

When data arrives to a webhook and the call is not processed instantly, Make stores it in the webhook processing queue.

The limit for the number of webhook queue items depends on your usage allowance, which is a part of your subscription. For every 10,000 operations licensed per month, you can have up to 667 items in each webhook's queue. The maximum number is 10,000 items in the webhook's queue.

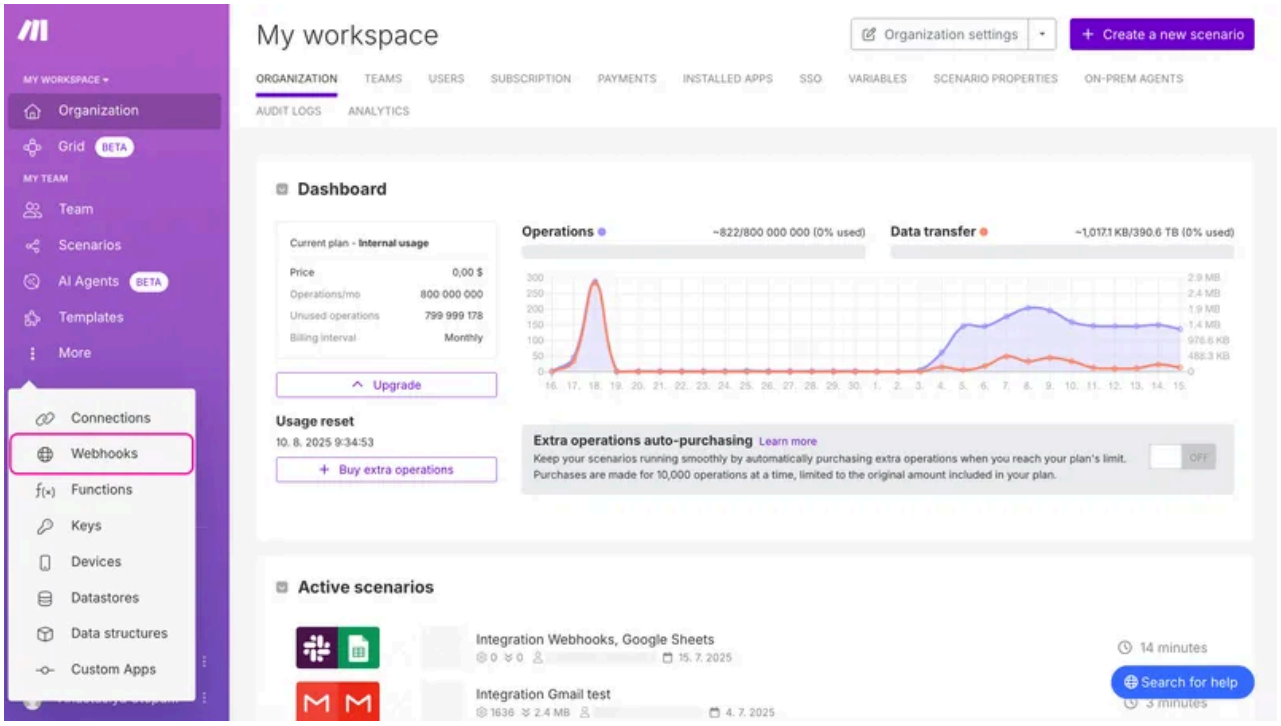
When the webhook queue is full, Make rejects all incoming webhook data which is over the limit.

Incoming webhook data is always stored in the queue regardless of the data is confidential option settings. As soon as the data is processed in a scenario, it is permanently deleted.

View webhook queue

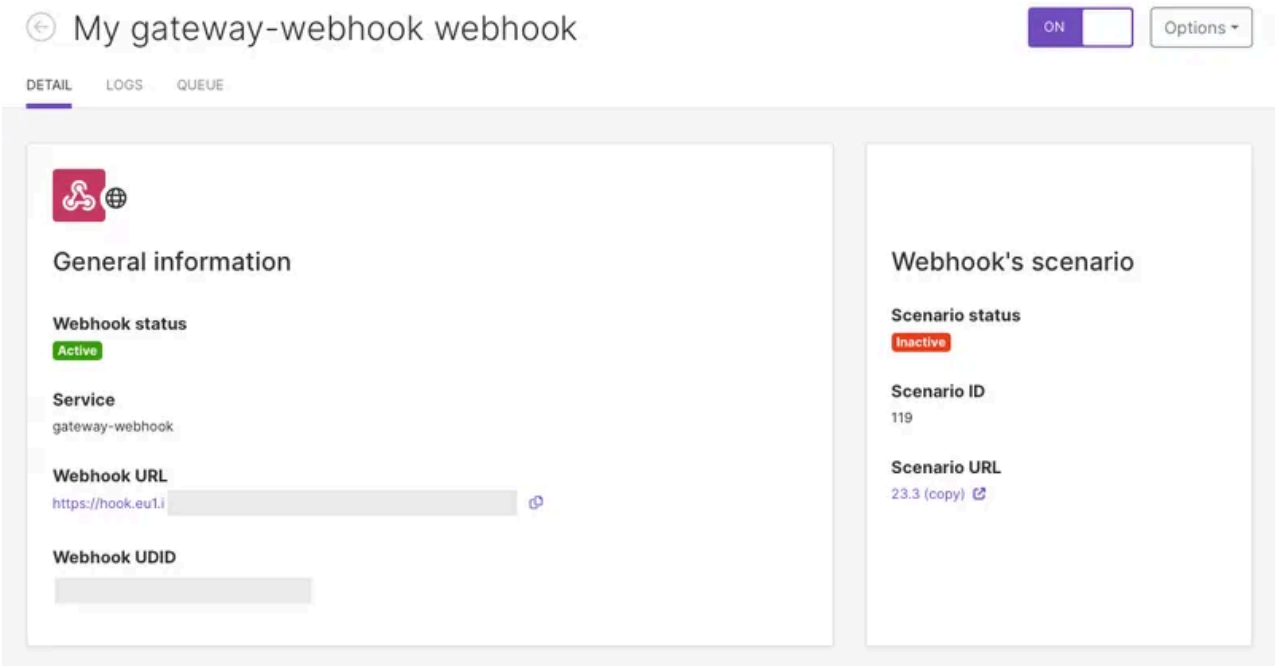
To view the content of the queue, follow the steps below.

1. In the left sidebar, click the **three dots** > **Webhooks**.



2. Find the webhook whose queue you want to view.

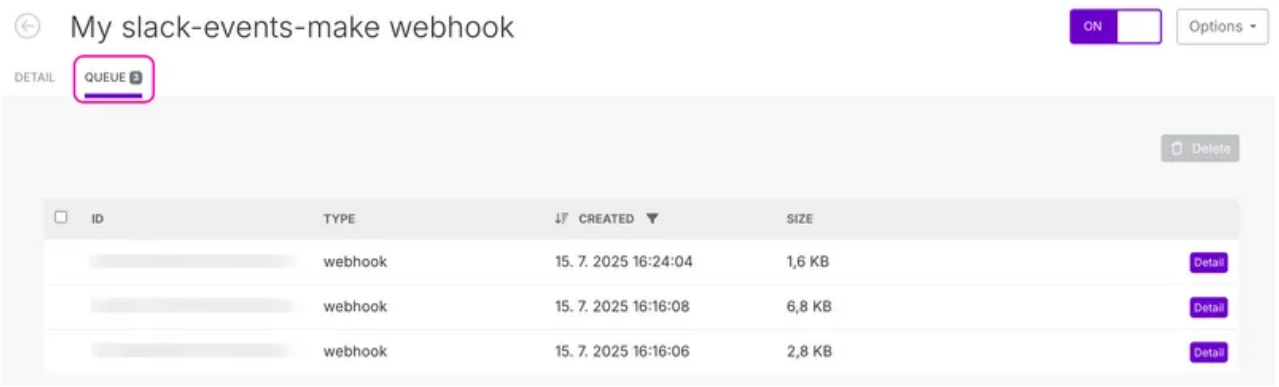
3. Click the specific webhook on the list to inspect its details.



You can see:

- Webhook status
- Webhook URL and webhook UDID (unique webhook identifier)
- Status of your scenario
- Scenario ID and scenario URL

4 To see the webhook's queue, click **Queue**.

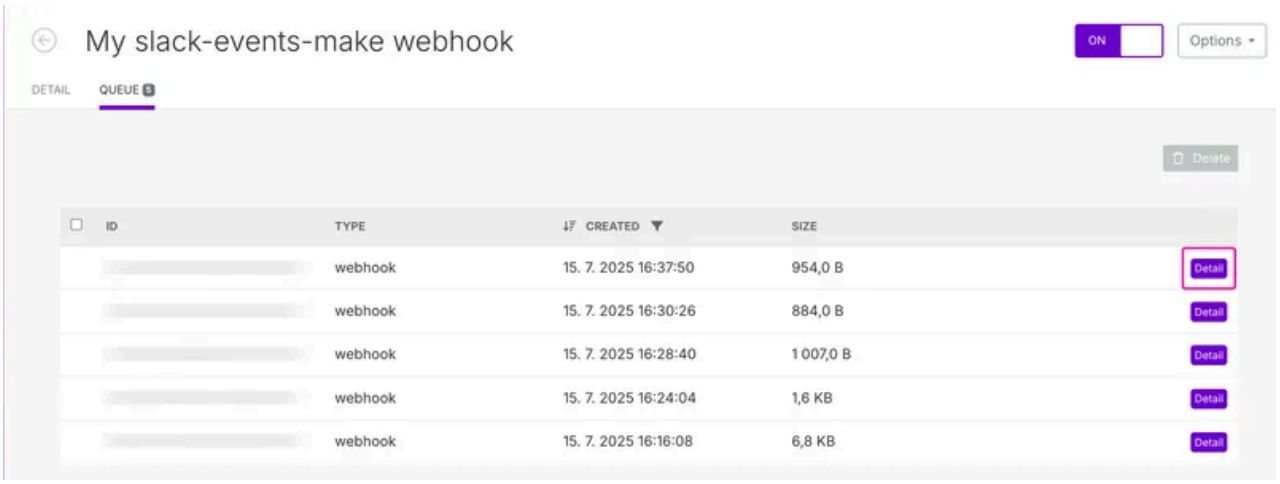


You can also click the button with the truck icon on the **Webhooks** page.



The webhook's queue displays.

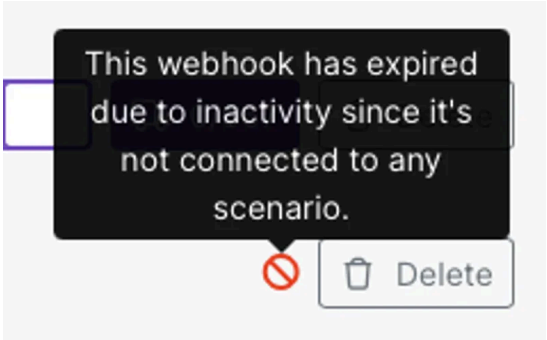
5 Click **Detail** by the webhook you want to inspect.



You can see the parsed items.

Expiration of inactive webhooks

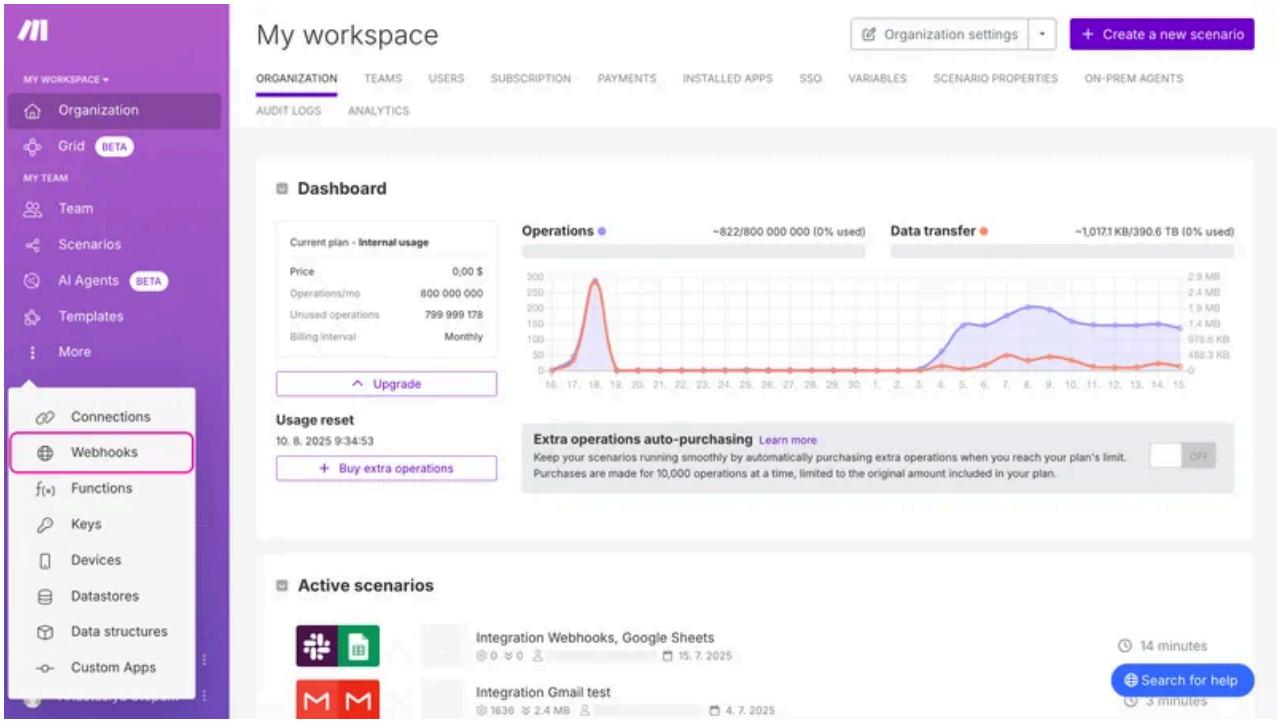
Make automatically deactivates webhooks that are not connected to any scenario for more than 5 days (120 hours). The hook return `410 Gone` status code.



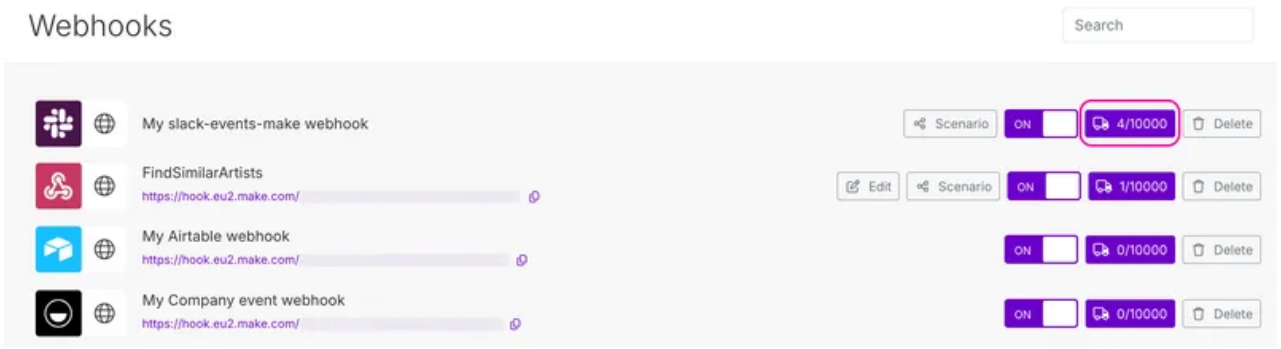
Delete webhook item from a queue

To delete a webhook item from a queue:

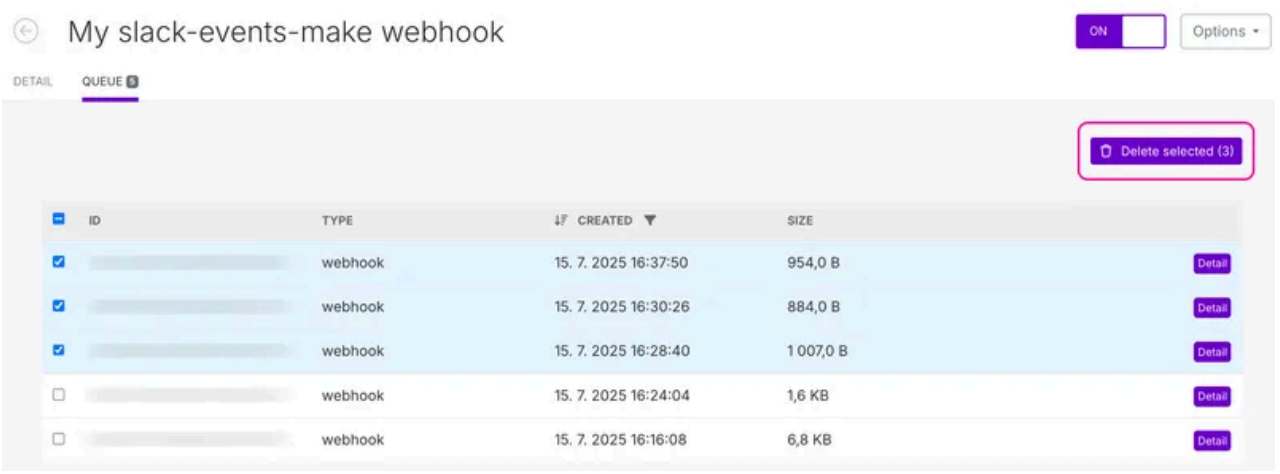
1. In the left sidebar, click the **three dots** > **Webhooks**.



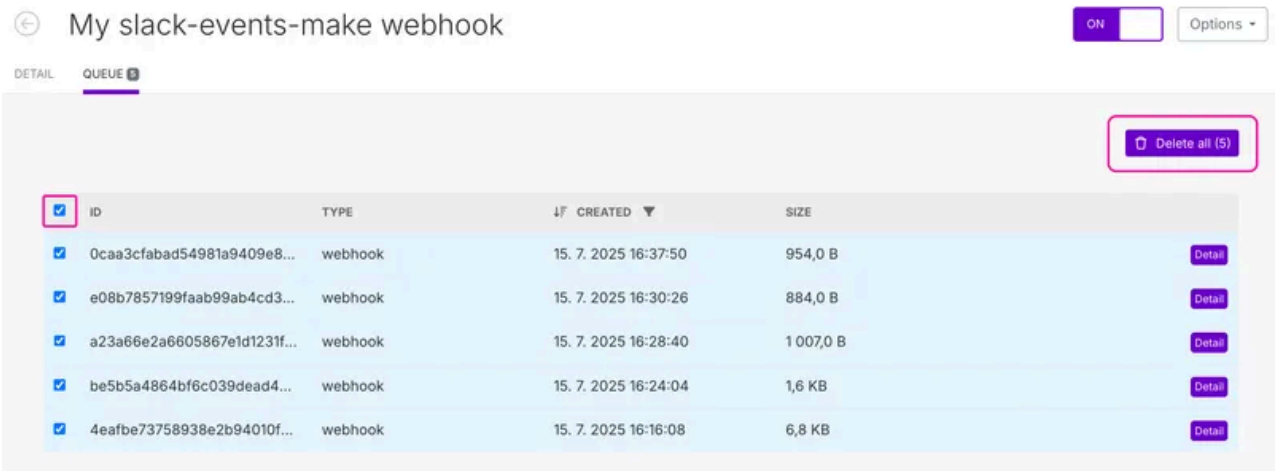
2. To see the webhook's queue, click the button with the truck icon.



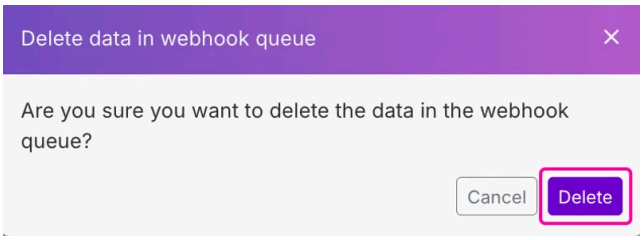
3. Tick the boxes on the left of the entries you want to delete, and click **Delete selected**.



To delete all, tick the first box on the left and then **Delete all**.



Click **Really?**, and then **Delete** to confirm.



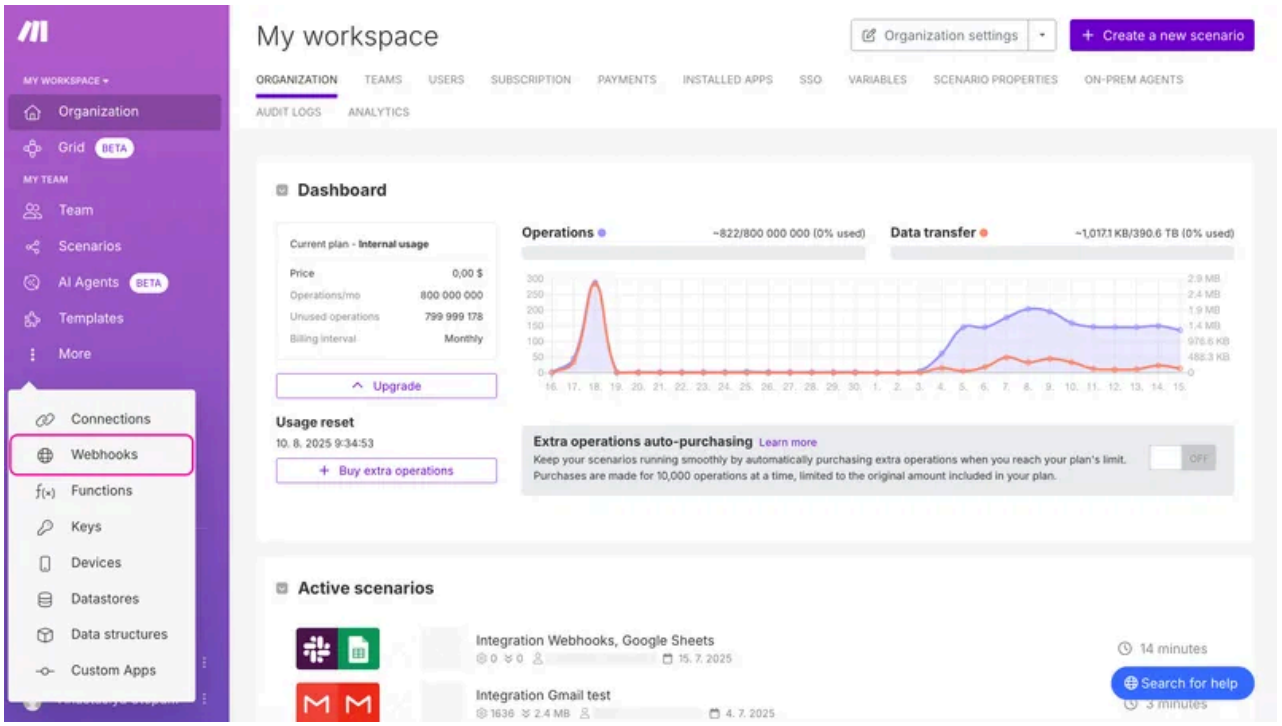
You have deleted the incoming webhook item(s) from the queue.

Webhook logs

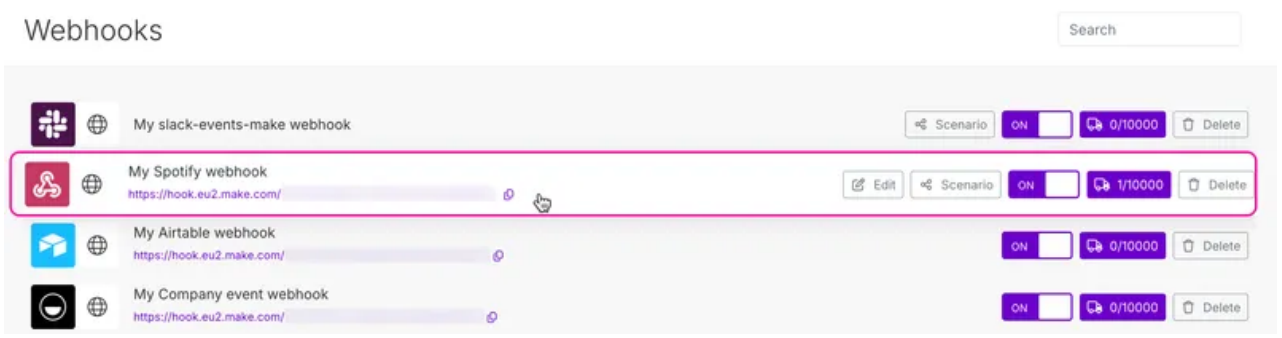
Make stores webhook logs for 3 days. For the organizations on the Enterprise plan, Make keeps the webhook logs for 30 days. Make deletes logs older than other retention limit.

To view webhook logs, follow the steps below.

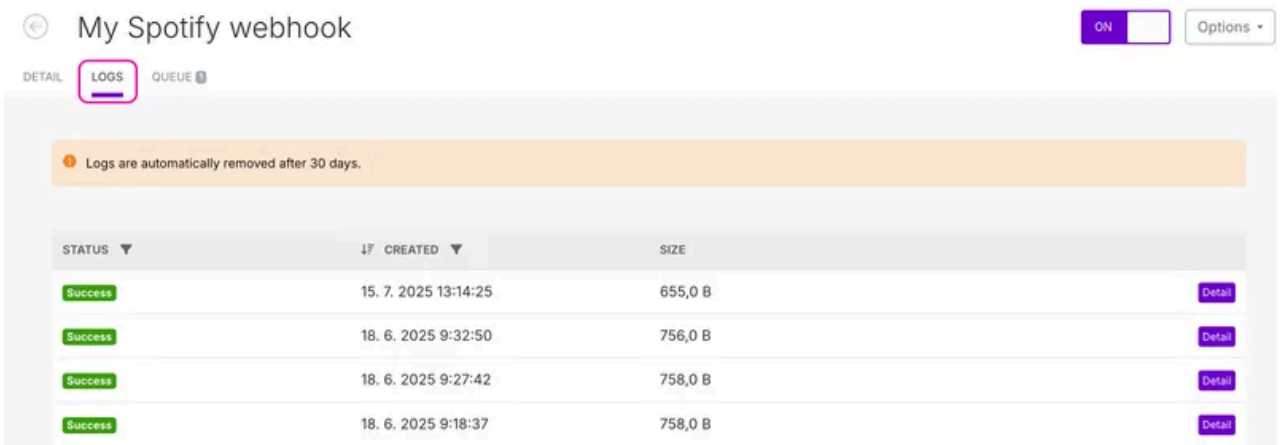
1. In the left sidebar, click the **three dots** > **Webhooks**.



2 Click the specific webhook on the list to inspect its details.

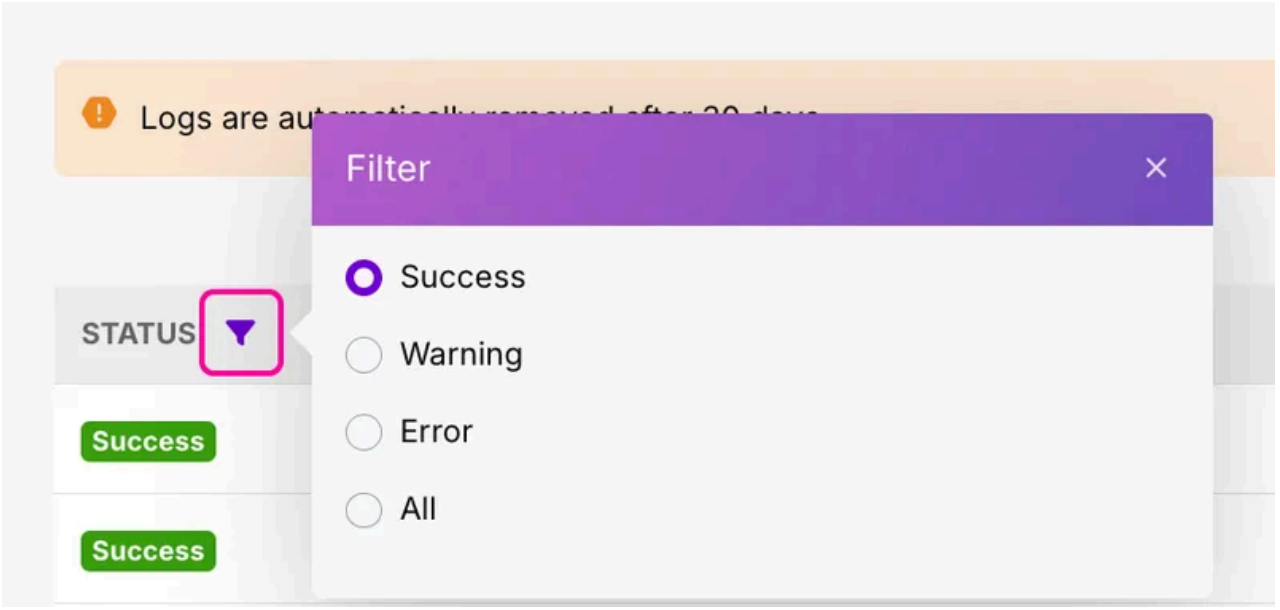


3 Click **Logs**.

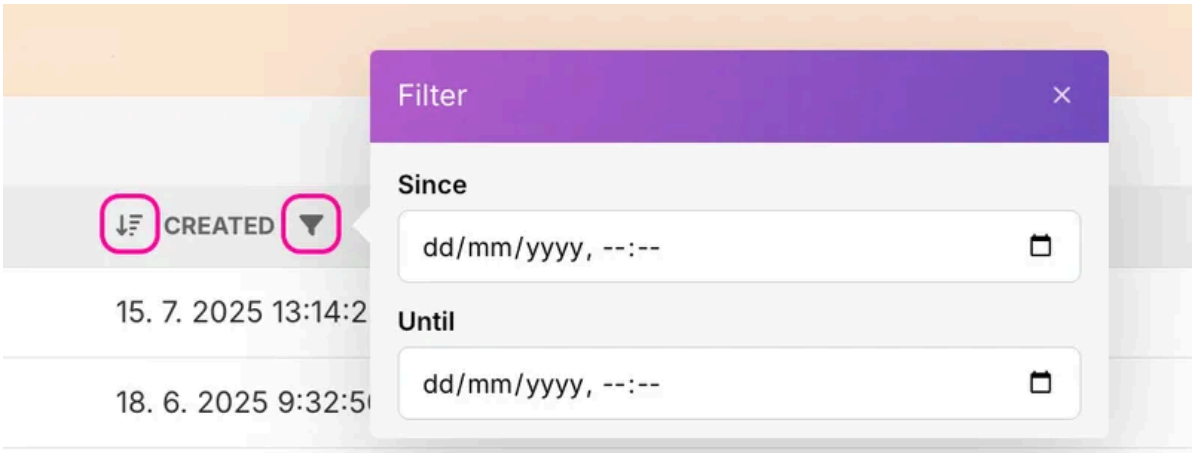


You can see:

- Status of the webhook call (success, warning, error, or all)
  To filter the webhook logs by status, click the filter icon.

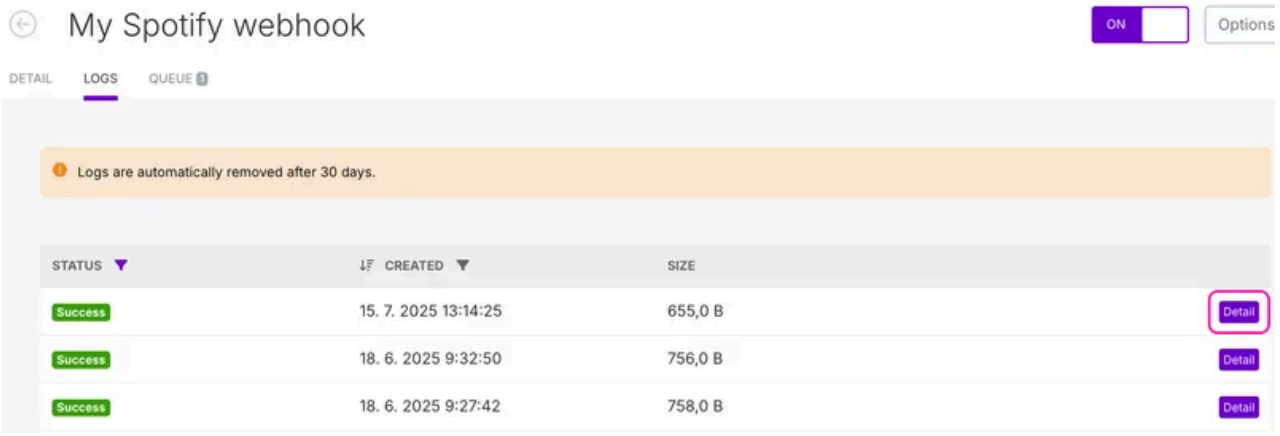


- Date and time of the incoming webhook
  To sort the webhook log by date and time, click the arrow icon. To filter the webhook logs by date of creation, click the filter icon on the right and enter a date.



- Webhook execution log size

4 To see the details of the specific webhook log, click **Detail**.



You can see:

- Webhook request (timestamp, URL, method, headers, query, body)
- Webhook response (status, headers, body)
- Parsed items
  Parsed items combine the query parameters and body of the webhook request in one bundle.

Updating webhook settings

You can update a webhook's settings when you use the Custom Webhooks module only. To learn more, see the **Webhooks** app documentation.

Error Handling

When there is an error in your scenario with a webhook, the scenario:

- stops immediately - when the scenario is set to run *Immediately*.
- stops after 3 unsuccessful attempts (3 errors) - when the scenario is set to run as **scheduled**.

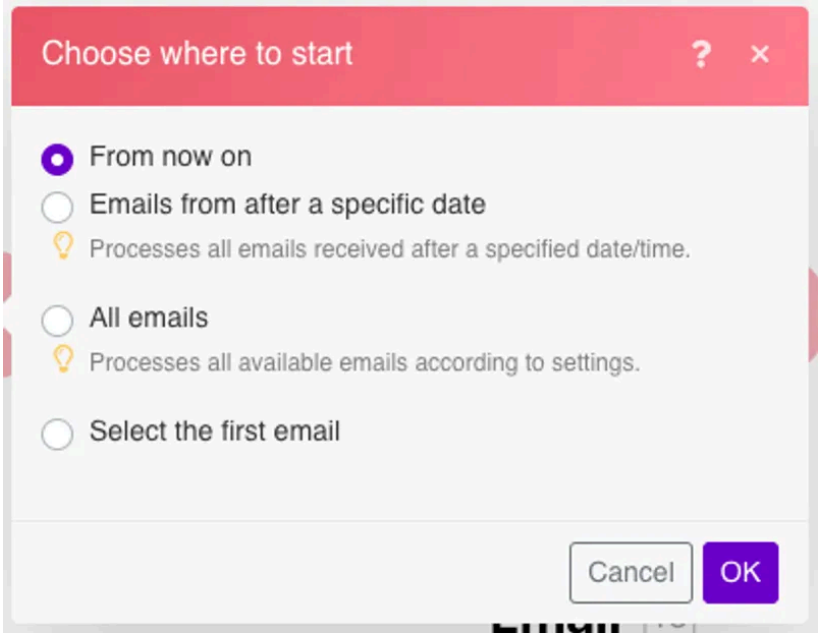
Webhook rate limit

Make can process up to 30 incoming webhook requests per second.

If you send more than 30 requests per second, the system returns an error with status code 429.

2.2.6. Selecting the first bundle

Some **triggers** allow you to select the first bundle where you want the retrieving of bundles to start from. You can also set whether to retrieve all bundles or just the bundles from after a specific date.



Displaying the panel

The panel to select where the first bundle should be retrieved displays automatically after you save a trigger or when you make a substantial change in the trigger **settings**.

You can also display the panel at any time by right-clicking on the trigger icon in the **Scenario editor**.

The options available in the panel differ trigger from trigger, depending on the the possibilities of a given service. In fact, some triggers do not offer any options at all.

Setting options

The panel for selecting the first bundle may contain 1 to 5 options:

- **From now on** (default) - retrieves all bundles added or updated (depending on settings) from now on.

- **From after a specific date** - retrieves all bundles added or updated (depending on settings) after a specified date/time.
- **With ID greater than or equal to a specific value** - retrieves all bundles with an ID greater than or equal to a specified ID.
- **All bundles** - retrieves all available bundles.
- **Select the first bundle** - allows you to select the first bundle from which the retrieval of bundles is to start.

2.3. Data & mapping

Learn about mapping and working with data in these articles:

→ **Item data types**

→ **Type coercion**

→ **Mapping**

→ **Filtering**

→ **Mapping arrays**

→ **Working with files**

2.3.1. Item data types

Every bundle contains several items. The type of item may differ. Make describes the different types of items below.

> Detailed information on which types of items Make allows for conversion between one other can be found in the **Type coercion** article.

Text

Also called **string**, text data contains characters such as letters, numbers, and special characters. Make validates text to make it meets any length requirements.

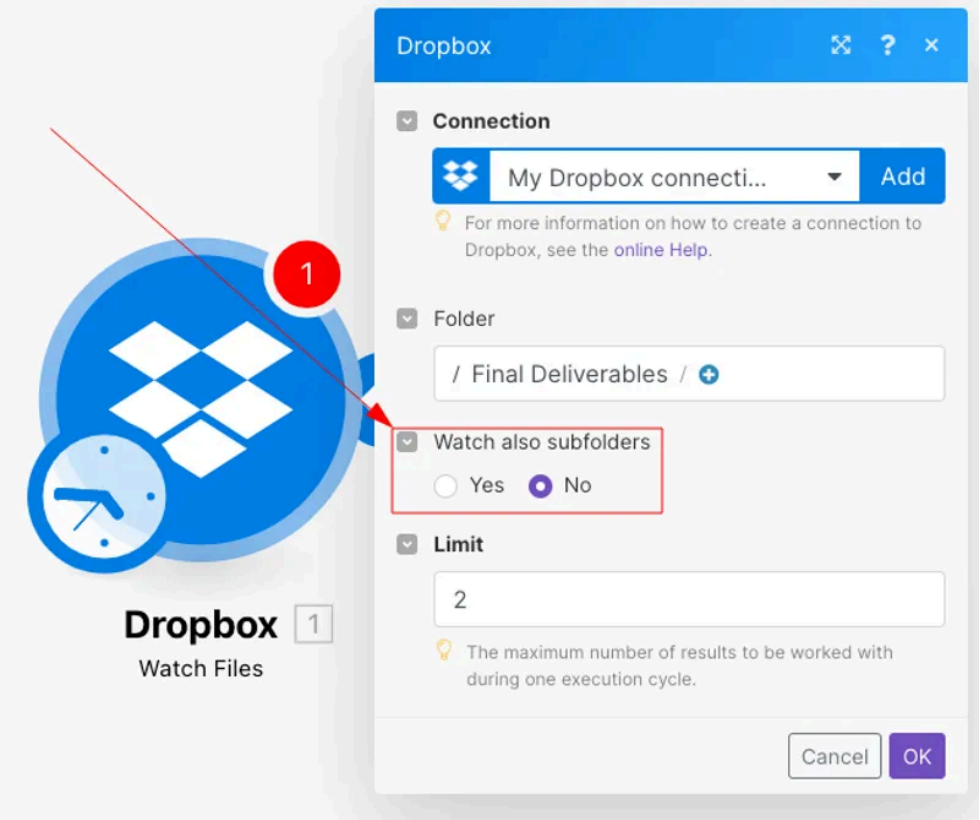


Number

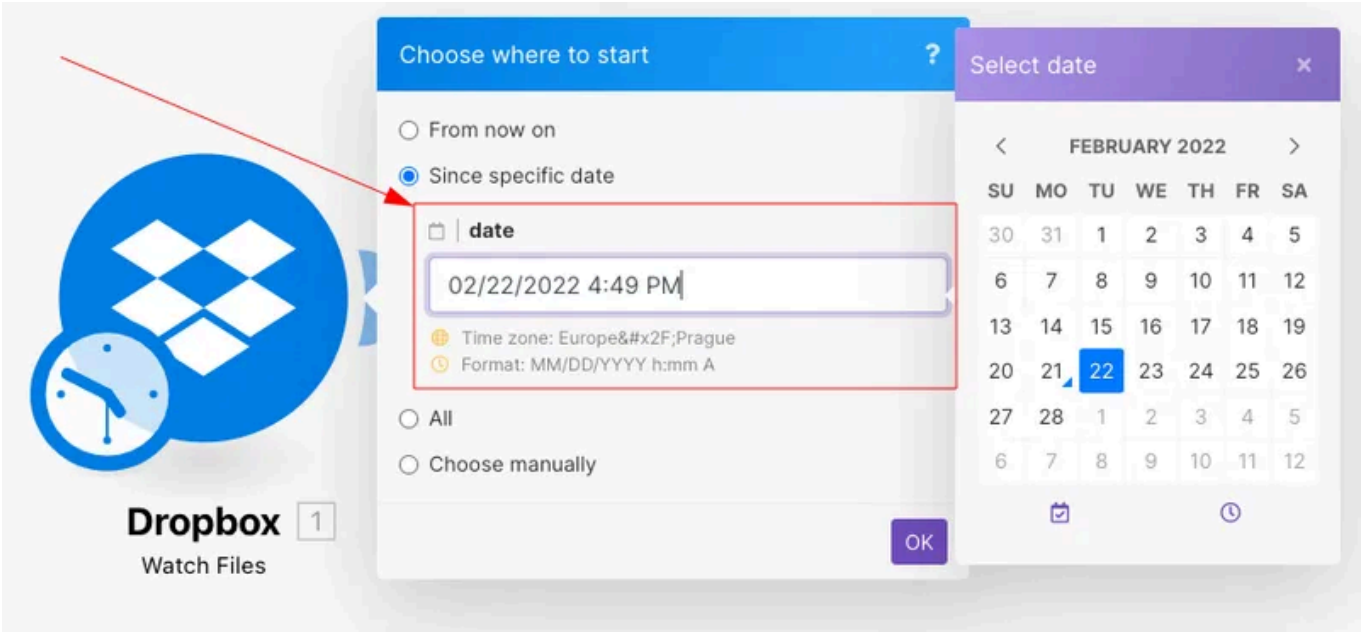
For some numerical items, Make may validate the input for a specified range (the minimum or maximum allowed value).

Boolean (Yes/No)

This type is used for items with only two possible values: true or false. An example is the *Watch also subfolders* option shown in the **Dropbox > Watch files** module.



Date



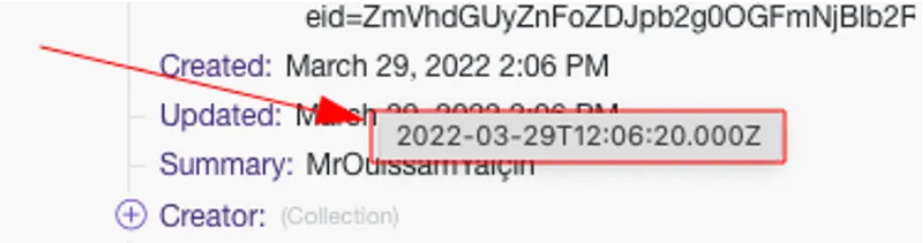
The date is entered in the **ISO 8601** date format, e.g. `2015-09-18T11:58Z`

The time zone can be adjusted in the profile settings. If you fill in a field that requires entering a date, the pop-up calendar will be displayed in the module settings. The time is not required for some items.

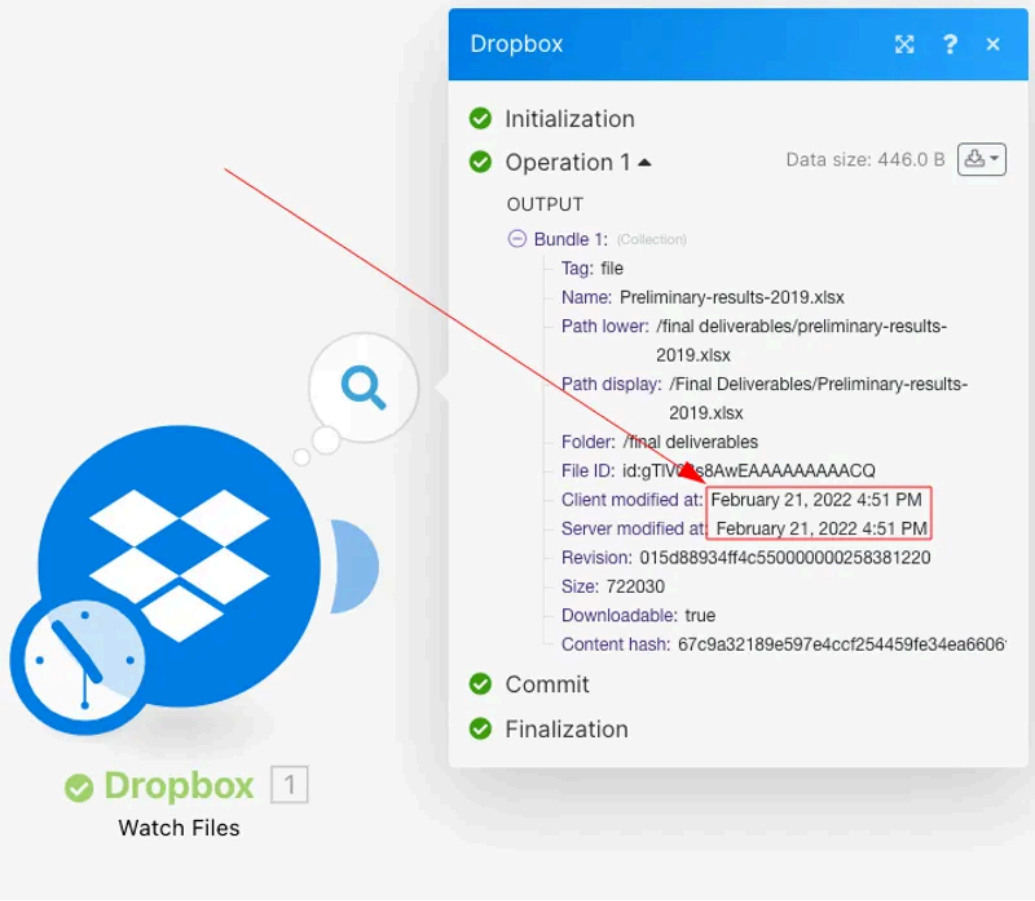
How Make displays date values

Values of Date items in bundle inspector are formatted using the locale and Web's timezone specified in your Profile in the *Web* section of the *Time zone options* tab.

It is possible to display a date item's value expressed according to ISO 8601 by hovering mouse cursor over the item.



> If the ISO value does not show up, the item is not a date but probably text.



Time

The time is entered in the `hours:minutes:seconds` format, e.g. `14:03:52`

Buffer (binary data)

File content is usually sent as Buffer type (image content, video file and others). In some cases, there is text data included in this type (e.g. text file). Make is able to automatically convert text data in binary code to text (and vice versa). For more information on working with files see the **Working with files** article.

Collection

A collection is an item consisting of several subitems. An example of a collection is the Sender item in an email message. It contains the sender name (text type) and the sender email address (text type).

Array

The array type is used for working with several values of the same type, including collections. An example is the Email modules return an array of attachments and each attachment contains name, content, size, etc.

For more information about arrays please see the **Working with arrays** section.

Validation

Make may perform validation on each type of item. If an item does not pass the validation, the processing of the module will terminate because of the DataError. The Twitter modules, for example, do not allow a tweet (text type) that exceeds the maximum of 280 characters to be sent. If you try to send a tweet with more than 280 characters, your scenario will encounter this error and end.

2.3.2. Type coercion

This document describes how Make behaves in situations when it receives values in expected and unexpected data formats.

When array is expected:

Received	Description
array	The value is handed over unchanged.
other	If the received value is not of the array type, Make will create an array and the first (and the only) element will be the received value.

When Boolean is expected:

Received	Description
boolean	The value is handed over unchanged.
number	The value is converted to logical `Yes`, even if the value is `0`
text	If the value is equal to `false` or the value is empty, it is converted to logical `No`. If not, it is converted to logical `Yes`
other	The value is converted to logical `Yes` whenever the received value exists (is not `null`).

When Buffer is expected:

Received	Description
buffer	The value is handed over unchanged only if the codepage is as expected. If the `codepage` differs, Make will try to convert the received value to the requested `codepage`. If this conversion is not supported, Make will return a validation error.
boolean	The value is converted to text (`true` / `false`) and then to binary data following the steps mentioned above for converting to **text**.
date	The value is converted to ISO 8601 text and then to binary data following the steps mentioned for converting to **text**.
number	The value is converted to text and then to binary data following the steps mentioned above for converting to **text**.
text	The value is converted to binary data and encoded as expected. If the expected encoding is not specified, `utf8` encoding will be used.
other	Make returns a validation error.

When Collection is expected:

Received	Description
collection	The value is handed over unchanged.
other	Make returns a validation error.

When Date is expected:

Received	Description
date	The value is handed over unchanged.
text	Make will try to convert the text to a date. If the conversion fails, it will return a validation error. Date must contain day, month and year. Date may contain time and time zone. Default time zone is based on your settings. See these **examples**.
number	Make will treat the number as milliseconds since Jan 01 1970 (UTC) and convert it to a corresponding date.
other	Make returns a validation error.

Examples of date format

- `2016-06-20T17:26:44.356Z`
- `2016-06-20 19:26:44 GMT+02:00`
- `2016-06-20 19:26+0200`
- `2016-06-20 17:26:44`
- `2016-06-20`
- `2016/06/20 17:26:44`
- `2016/06/20 19:26:44+02:00`
- `2016/06/20 17:26`
- `2016/06/20 5:26 PM`
- `2016/06/20`
- `06/20/2016 17:26:44`
- `06/20/2016 19:26:44+02:00`
- `06/20/2016 17:26`
- `06/20/2016 5:26 PM`
- `06/20/2016`
- `20.6.2016 17:26:44`
- `20.6.2016 19:26:44+02:00`
- `20.6.2016 19:26:44+02:00`
- `20.6.2016`

When Number is expected:

Received	Description
number	The value is handed over unchanged.
text	Make will try to convert the text to a number. If the conversion fails, it will return a validation error.
other	Make returns a validation error.

When Text is expected:

Received	Description
text	The value is handed over unchanged.
array	If the given array supports conversion to text, the value will be converted. If not, Make will return a validation error.
collection	The value is converted to a JSON string. An array of collections is converted separately to a comma-separated list.
boolean	The value is converted to text (`true` / `false`).
buffer	If text encoding is specified for binary data, the value will be converted to text. If not, Make will return a validation error
date	The value is converted to ISO 8601 text.
number	The value is converted to text.
other	Make returns a validation error.

When Time is expected:

Received	Description
time	The value is handed over unchanged.
text	Make will try to convert time to the hours:minutes:seconds format. If the conversion fails, it will return a validation error.
other	Make returns a validation error.

2.3.3. Mapping

What is mapping?

Mapping is how Make knows what information to get from one app and which app to send it to. You can use mapping to tell Make how to transform data in your scenario. For example:

- You want to automatically send information from emails to a colleague in a Slack message. In this case, map the subject or body of email messages to Slack.
- When a customer submits a form to request support, you want a Jira issue created automatically. Here, you can map the information from the form into Jira.
- Your business gets valuable leads from Facebook, so you want a database of this information and the same information entered into Pipedrive. Just map the data from Facebook leads to Airtable and Pipedrive.

In the same way that a map leads you from your starting point to your destination, mapping in Make tells your data where to go in your scenario.

What can I map?

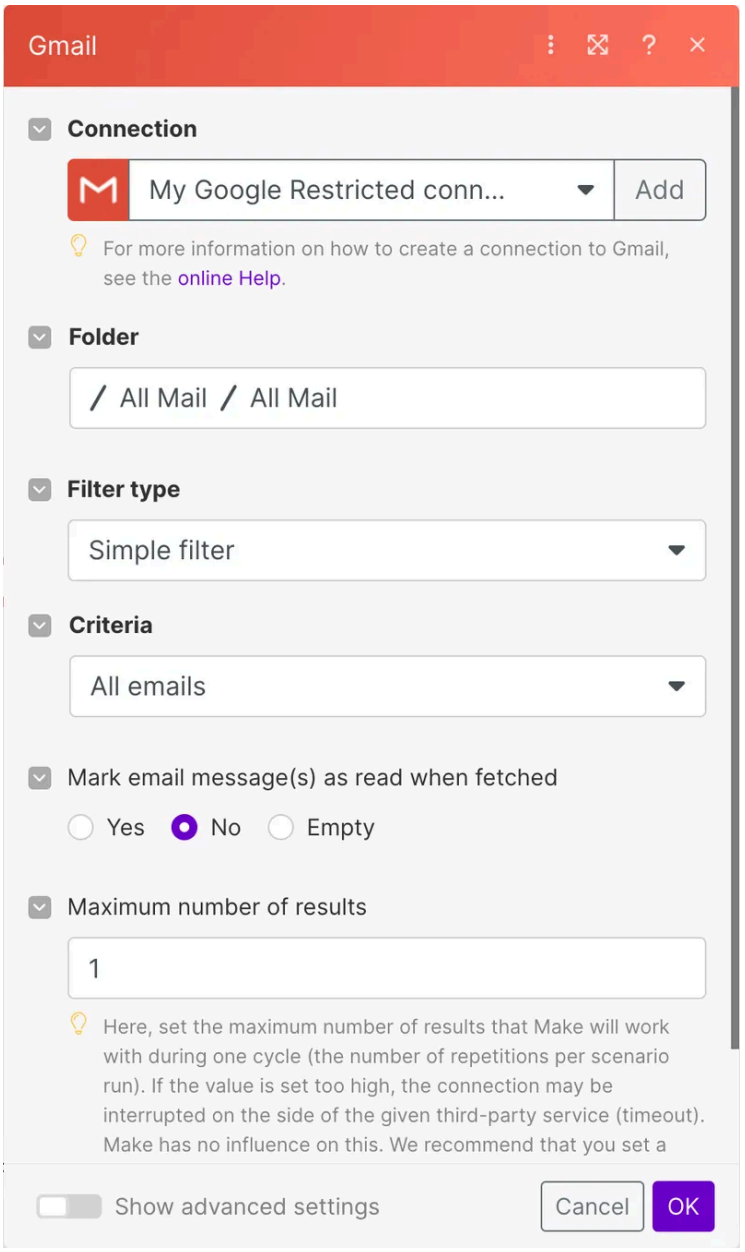
You can map any information that a module gets in a bundle. A **bundle** is like a box of information. Some boxes are large; some are small; some have just one thing inside them; some might have smaller boxes in them. Bundles are similar. Sometimes a bundle is just one piece of information, for example, a number or email address. Other times a bundle contains a variety of information.

When bundles have a lot of information, you might see the data organized into an **array** or **collection**. An array is like a menu organized into different sections like soups, starters, desserts, and so on. There's a lot of information but it's the same kind of information: food. An array has information of only one kind. For example, it might have multiple dates or names but not both. Collections can have different kinds of information and are like a moving box. When you move, you might put various things into one box: books, small gadgets, a box of letters. Collections contain a similar variety of information, for example, dates, numbers, and text descriptions.

Getting a bundle

To know what you can map from a module, you run only that module and look at its bundles. Once you know what you can map, you can start mapping. Let's use the first example above to explore how this works. We want to take information from emails and send them to a colleague. Before we do any mapping, we begin by finding what information we can get from the email module. Because the email module provides the information we want, we call it the **source module**.

1. Go to the **Scenario Builder** and add the module **Email > Watch emails**.
2. Create a connection.
3. Select a folder and criteria.
4. Click **OK**.



5. Choose where you want Make to start getting your emails and click **OK**.

6. Right-click the module.

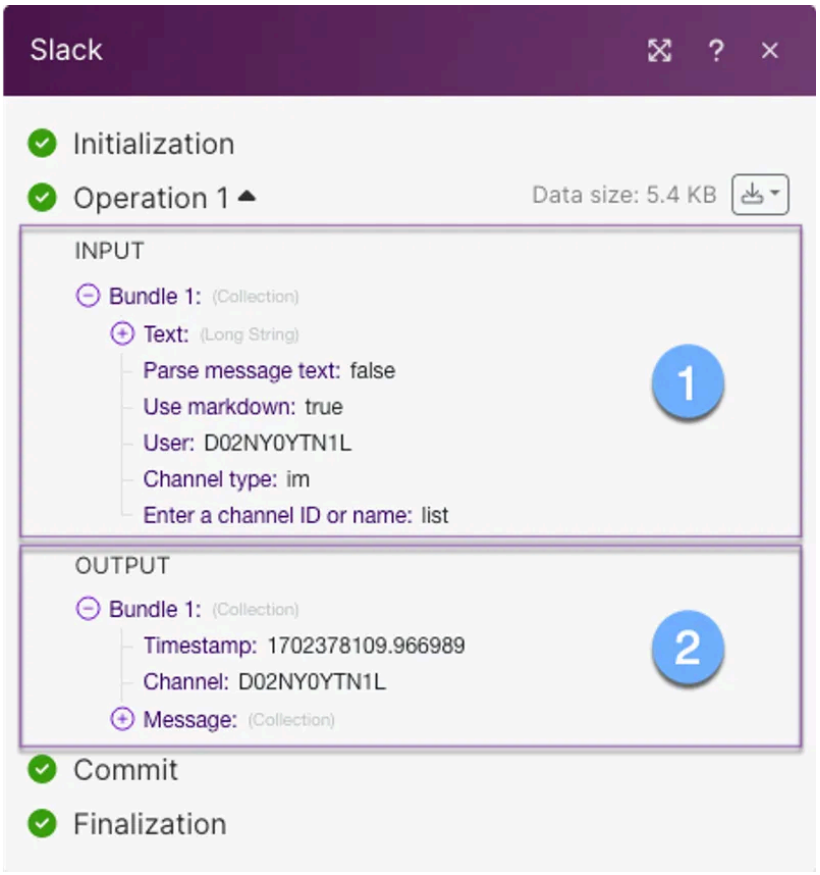
7. Select **Run this module only**.

8. A small circle with 1 inside appears beside your module. Click inside the small circle.

A window opens and shows you all of the information that your email module got. You can map anything that appears. Click **+** to expand items.

What's in a bundle?

Let's take a closer look at the information in the bundle window. In this example, we see the bundles from the **Slack > Create a message** module.



1. Input - the information Make sends to the third-party app.

   In this example, you can see:

   - Text - the content of the message sent. Click + to expand for further details.
   - User - the user ID of the person receiving your Slack message.

2. Output - the information Make gets back from the third-party app.

   In this example, you can see:

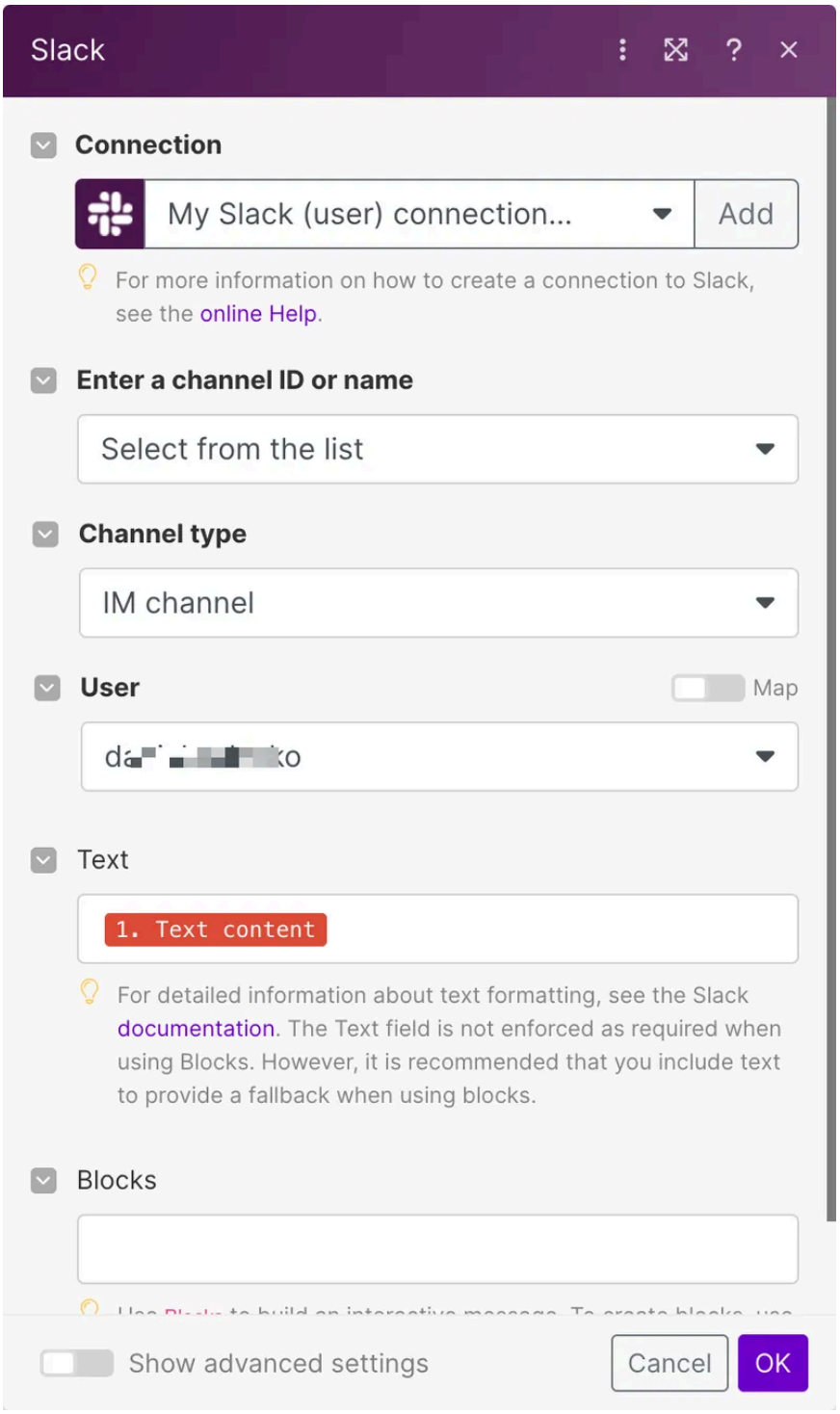
   - Message - the text content of the sent message, similar to Text in the Input. Click + to expand for further details.

- Channel - the ID of the channel where the message was sent. In this case, it's the same as User from the input. That's because we sent a direct message to a specific user.

Mapping to the target

Now we know what information we can send to other modules. Next, we map that information. To do that, we first add the module we want to receive the information. In our example, that's **Slack > Create a message**. If you're still in the Scenario Builder, continue with these steps:

1. Add the module **Slack > Create a message**.
2. Create a connection.
3. Under **Enter a channel ID or name**, select **Choose from a list**.
4. Under **Channel type**, select **IM channel**.
5. Under **User** search or scroll to find your intended recipient.
6. Under **Text**, click inside the empty field. A new window appears.
7. In the new window, click **Text content**. A red box labeled **Text content** appears in the **Text** field.
8. Click **OK**.



Your data is now mapped so that Slack automatically receives email content and sends it in a message.

Tips and solutions

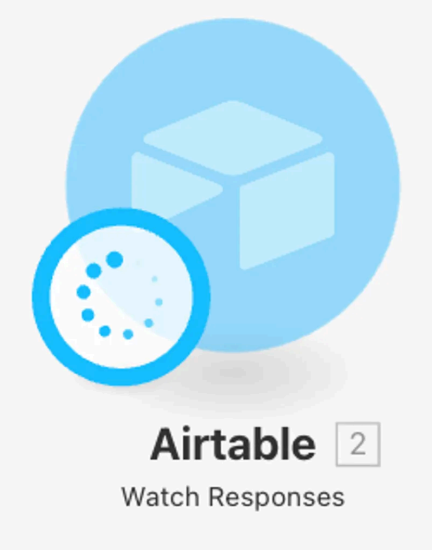
Find the source module

While mapping a scenario, you can find the source module by hovering over the mapped data. As you keep your mouse on a mapped item, the source module pulses.

When there's no info for an instant trigger

When you click Run once or Run this module only, an instant trigger waits to receive information. The instant trigger needs incoming info to run. So the trigger waits until it receives incoming data. To run the module, you need to provide incoming information. Let's explore with an example: **Airtable > Watch responses**.

1. Right-click on **Airtable > Watch responses** and select **Run this module only**. The module waits for new responses.



2. Go to your Airtable form and submit a response.

The information submitted goes to your **Airtable > Watch responses** module. You can inspect the bundle to check that your configuration is working.

When there's no info for a polling trigger

You can use the **above procedure** to generate a bundle for most modules, including polling triggers. For instant modules, see the solution for instant triggers. The following are the general steps to get a bundle.

1. Go to the **Scenario Builder** and find the module you want to get a bundle for.

2. Right-click the module.

3. Select **Choose where to start**.

4. To get a bundle, select one of the following options and click **OK**.

   a. **From a specific date option**: for example **Emails from after a specific date**. Enter a date you know there is data for. Example: enter a date you know you received emails.

   b. **Select the first option**: when available, this option lets you choose from a list.

5. Right-click the module.

6. Select **Run this module only**.

A small circle with 1 inside appears beside your module. Click inside the small circle.

2.3.4. Filtering

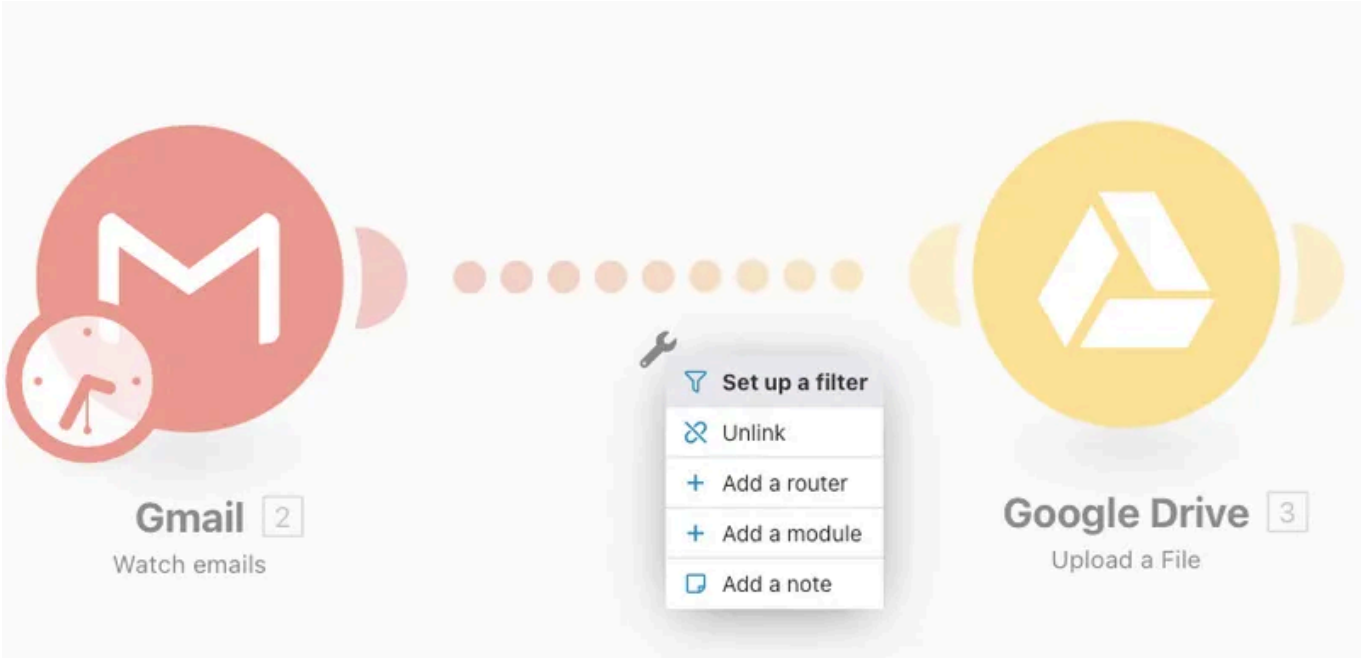
In some scenarios, you may need to only work with bundles that fit specific criteria. Filters will help you to select those bundles.

You can add a filter between two modules and check whether bundles received from the preceding modules fulfill specific filter conditions or not. If yes, the bundles will be passed on to the next module in the scenario. If not, their processing will be terminated.

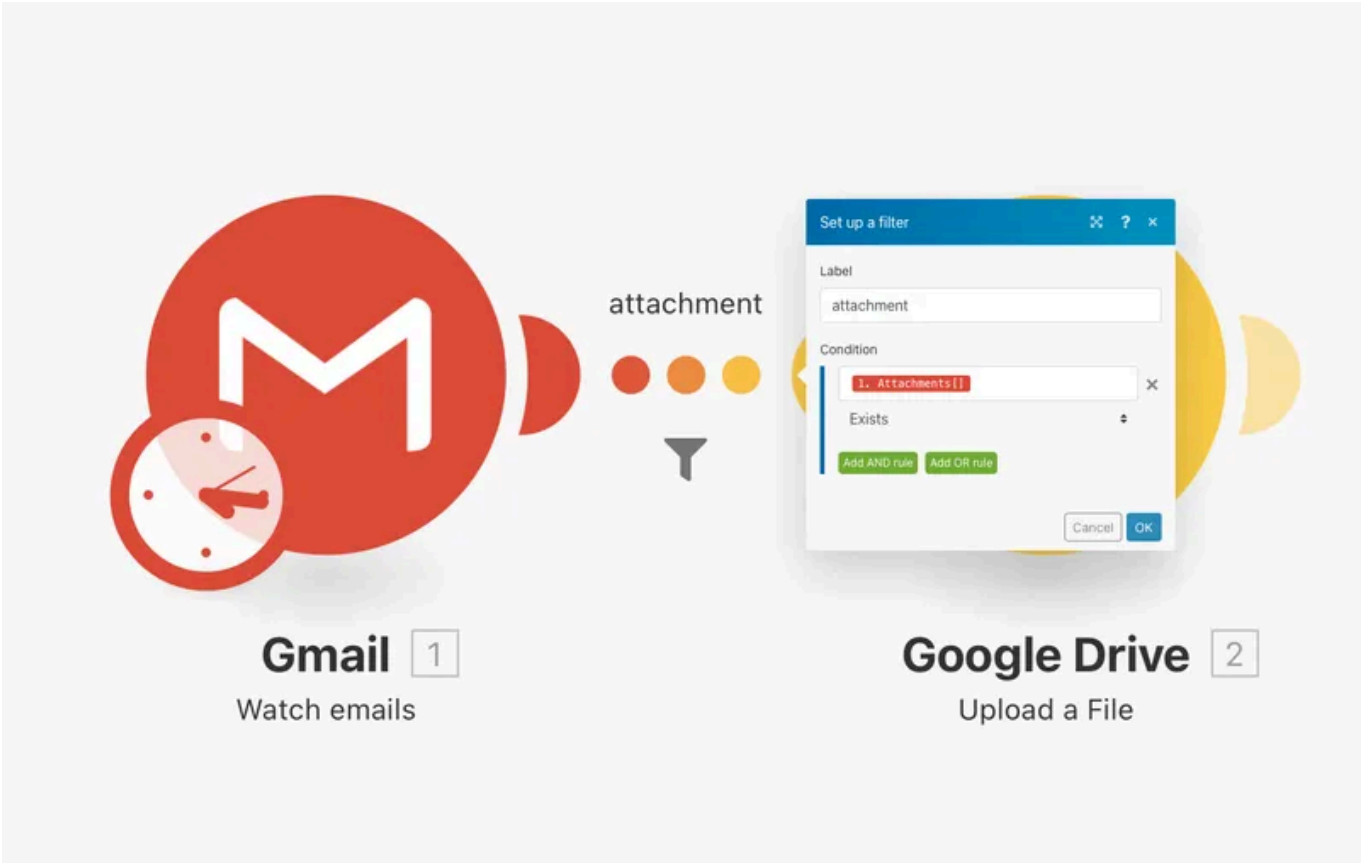
For example, if you want to create a scenario with the **Facebook** trigger *Watch posts* and you want to work only with posts containing a specific word or posts written by a specific author, a filter would make sure you receive only these posts and nothing else.

Adding a filter

To add a filter between two modules, click on the connecting line between them.



This brings up a panel where you can enter the name for the filter that is to be created and define one or more filter conditions.



For each condition, you can enter one or two operands and an operator that will determine the relation between them. In the operand field, you can enter values in the same way as you would **map** them.

In the example above, you can see how to connect the **Gmail** trigger *Watch emails* and the **Google Drive** action *Upload a file*. The filter automatically applies the condition to incoming bundles from the first module and only bundles containing attachments are allowed to pass on to the next module.

Operators

For each condition, you can use one of several different operators.

Basic operators

- **Exists** - checks whether a specific bundle item is filled in. Using this operator, you can create a filter that permits, for example, only Facebook posts that contain a photo to go through to the next module in a scenario.
- **Not exists** - the opposite of *exists*. It permits only those bundles where a specific item is not filled in.

Other operators

There are a number of other operators you can use: text comparison operators, numerical operators, time and date operators, and operators for working with **arrays**.

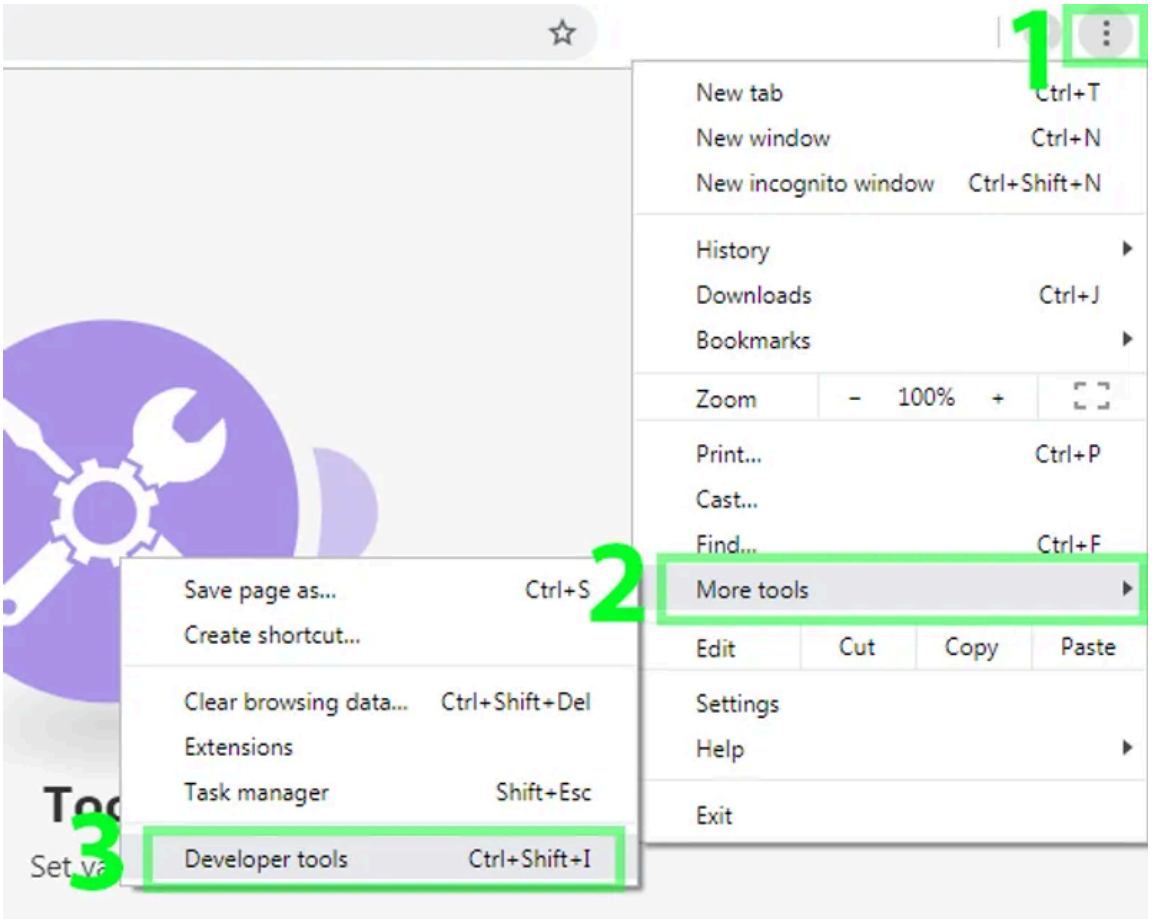
Copying a filter

Copying filters is currently not natively supported by the **Scenario editor**, though if you use the Google Chrome web browser, the following workaround can be employed:

1. Install **Make DevTool** Chrome extension.

2. In Make, open your scenario.

3. In Chrome, open the **Developer tools**. You can do so either by choosing the **Developer tools** command from the the Chrome main menu or just press Ctrl+Shift+I or F12:



4. In the **Developer tools**, click on the Make tab.

5. Click on the **Tools** icon in the left side bar.

6. .Click on the **Copy filter** tool and configure it in the right side panel.

7. Set the **Source Module** field - the module that's right after the filter you wish to copy.

8. Set the **Target Module** field - the module before which you wish to copy the filter.

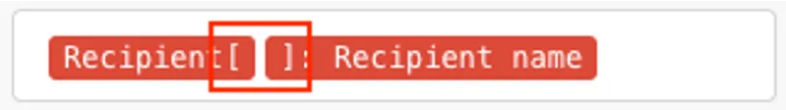
9. Click on the **Run** button.

2.3.5. Mapping arrays

An array is a special type of item. A simple array contains one or more text values. A complex array contains one or more collections of the same type. An example of a complex array is the email attachment. The Watch emails module returns an array of attachments for every email. Every attachment represents a collection that may contain a name, content, size, etc.

Mapping an array's first element

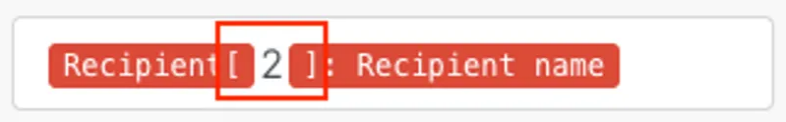
If you map the array's `Recipient name` item, it will appear in the field like this:



The number between the square brackets is an index that determines which element of the array will be used. Leaving it empty defaults to the first element.

Mapping an array's element

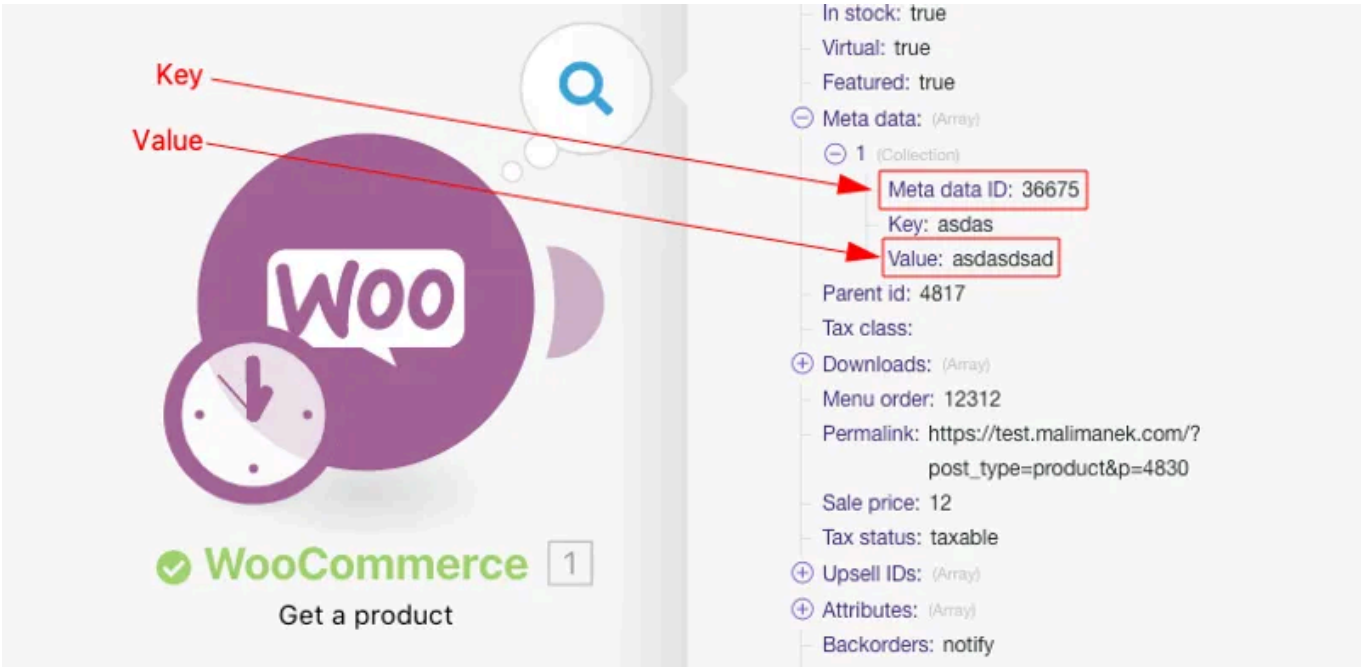
If you wish to access another element, enter or map a value between the square brackets. In the below example, enter 2 to select the second element.



Mapping an array's element with a given key

Some arrays contain several collections with *key* and *value* items. These are typically various metadata, attributes, etc.

The following example shows the output of the *WooCommerce > Get a product* module that contains the item `Meta data`, which is an array of collections. Each collection contains the key item `Meta Data ID` and the value item `Value`:



The typical requirement is to lookup an element by its given *key* value and to obtain the corresponding value from the *value* item. This can be achieved with a **formula** employing a combination of the **map ()** and **get ()** functions.

The following example shows how to obtain the value of the `Value` item of the element with *key* `Meta data ID` item value equal to `20642`

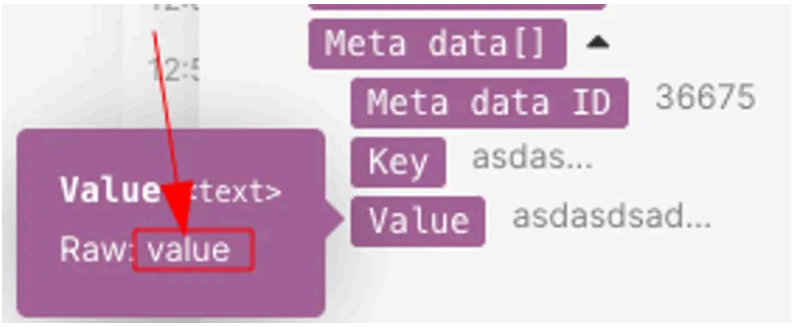
```
get( map( 4. Meta data[] ; value ; id ; 20642 ) ; 1 )
```

The result of the formula will be "no".

The detailed breakdown of the formula follows:

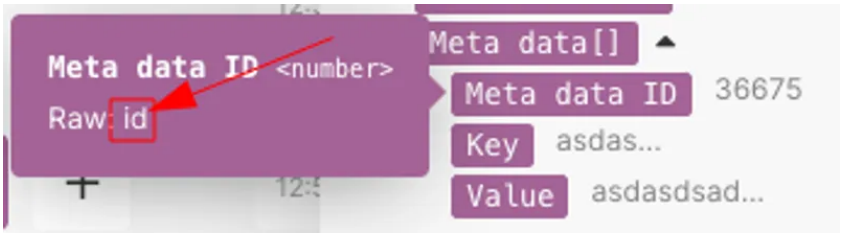
1. The 1st parameter of the `map()` function is the whole array item.

2. The 2nd parameter is the **raw name** of the value item. To obtain the raw name, hover the mouse cursor over the item in the mapping panel:



> All parameters are case sensitive. Even though in this particular example the item's label differs from its raw name only in capitalization, it is necessary to use the raw name, which is all lowercase `value` in contrast to the label `Value`

3. The 3rd parameter is the raw name of the *key* item:



4. The 4th parameter is the given *key* value.

Because the `map()` function returns an array (as there could be more elements with the given key value), it is necessary to apply the `get()` function to get its first element:

1. The 1st parameter of the `get()` function is the result of the `map()` function.

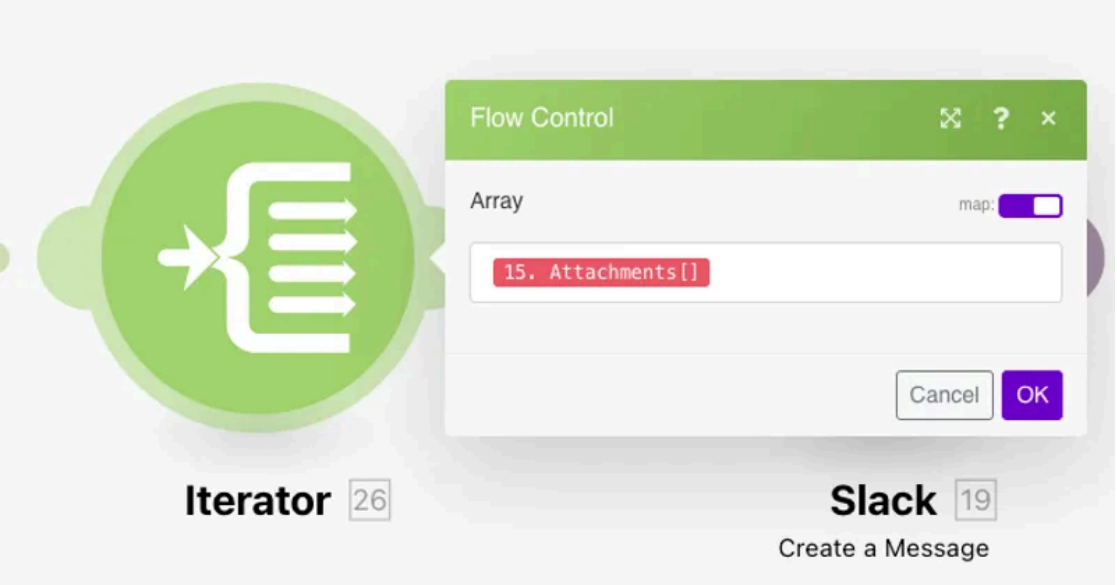
2. The 2nd parameter is the element's index - one.

See also our **Extract an item and/or its value from an array of collections** video tutorial.

How to Extract an Item And/Or Its Value From an Array of Collections

Converting elements to a series of bundles

Arrays can be converted to a series of bundles using the **Iterator** module:



> The outputs from modules wrapped between an Iterator and Aggregator are not accessible beyond the Aggregator module.

2.3.6. Working with files

Some modules have the capability to process files. These modules can either return an output file to be sent for further processing or require a file to be passed to them for processing. For these modules to work together to process files they have to be mapped to each other.

How to map a file

Modules that have the ability to work with files require two pieces of information:

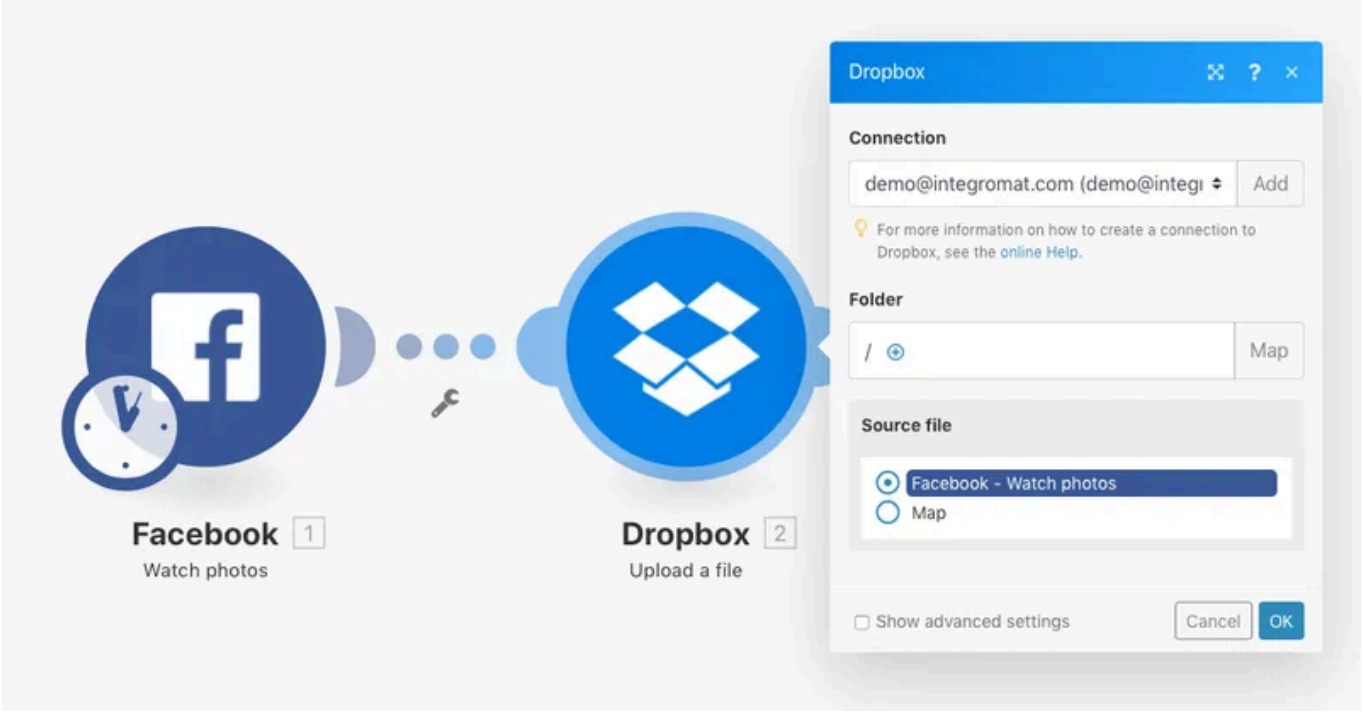
1. File name
2. File content (data)

When you map a file, you choose from which module(s) in your scenario you want to obtain the data from. The file name and file content are then automatically mapped as they are.

> If you need to process a file from a URL, we recommend to use the **HTTP > Get a File** module to download the file from the URL, and then map the file from the **HTTP > Get a File** module to the desired module's field in your scenario.

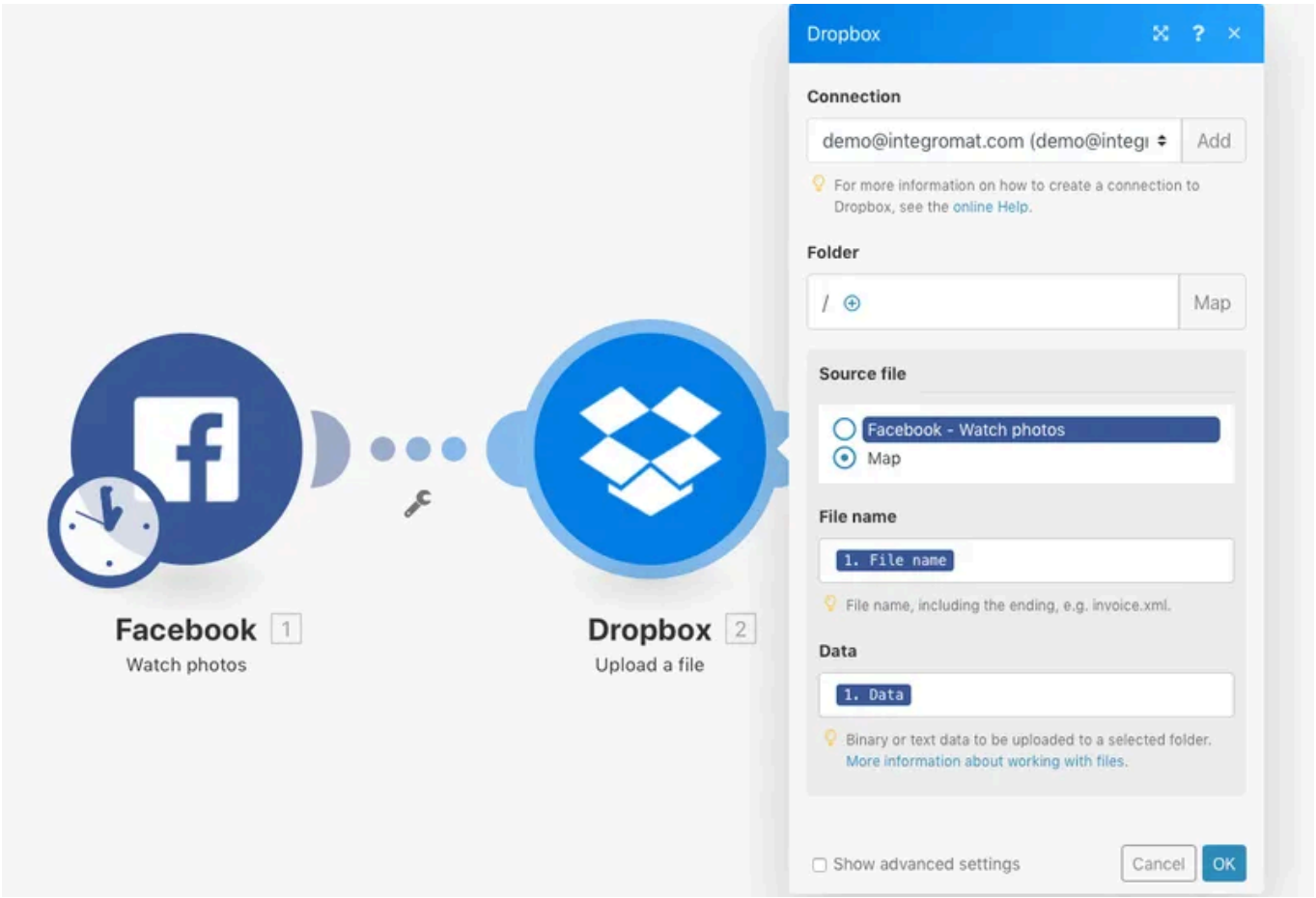
Example: Mapping Facebook photos to Dropbox

The example below shows how to map photos (files) from Facebook to Dropbox. The facebook trigger **Watch photos** returns detailed information about each downloaded photo, including its name and content (data). To map this information to Dropbox so it can be uploaded, the **Dropbox** action **Upload a file** requires you to specify the source file that this information will be mapped from. By selecting the **Facebook > Watch photos** option under the source file, Make will automatically map the file name and file content. From now on, all photos from Facebook will be uploaded to the specified Dropbox folder.



However, it can happen that you want to rename the file, but want to keep the data as it is. In such case, use the **Map** option (as seen in the image below) to map the file name and file content independently of each other.

You will have to enter the full file name, including the ending (e.g. invoice.xml). Both binary (photos, videos, PDF and the like) and text formats are supported.



Maximum file size

The maximum file size depends on the plan you are subscribed to (FREE - 5 MB, CORE - 100 MB, PRO - 250 MB, TEAMS - 500 MB, ENTERPRISE - 1,000 MB). If it is exceeded, Make proceeds in accordance with the settings of the Enable data loss option.

2.4. Tools

Learn about tools for transforming data in these articles:

→ **Aggregator**

→ **Iterator**

→ **Router**

2.4.1. Aggregator

An **aggregator** is a **type of module** designed to merge several bundles of data into a single bundle.

When an aggregator is executed, it:

1. accumulates all the bundles it receives (during a single source module's operation)
2. outputs a single bundle with an array containing one item per each accumulated bundle. The content of the array's items depends on particular aggregator module and its setup.

A typical example of an aggregator module is the **Array aggregator** module. Aggregators usually feature the following fields:

Source Module	The module from which the bundle aggregation will start. The source module is usually an **iterator** or a search module that outputs a series of bundles. Once you setup the aggregator's Source Module (and close the aggregator's setup), the route between the source module and the aggregator will be wrapped in a grey area to visualize the start and the end of the aggregation.Searches
Group by	The aggregator's output can be split into several groups with the help of the *Group by* field. The *Group by* field can contain a formula that is evaluated for each aggregator's input bundle. The aggregator then outputs one bundle per each distinct formula's value. Each bundle contains two items: • `Key` contains the distinct value. • `Array` contains the aggregated data from the bundles for which the formula evaluated to the `Key` value.
Stop processing after an empty aggregation	By default, the aggregator outputs the result of the aggregation even in case no bundles reached the aggregator (e.g. because they have been all filtered out on their way). If the *Stop processing after an empty aggregation* option is enabled, the aggregator will not produce any output bundle in this case and the flow will stop.

> Bundles outputted from the source module and any other modules between the source module and the aggregator module are not outputted by the aggregator and thus items in these bundles are not accessible by the modules in the flow after the aggregator.

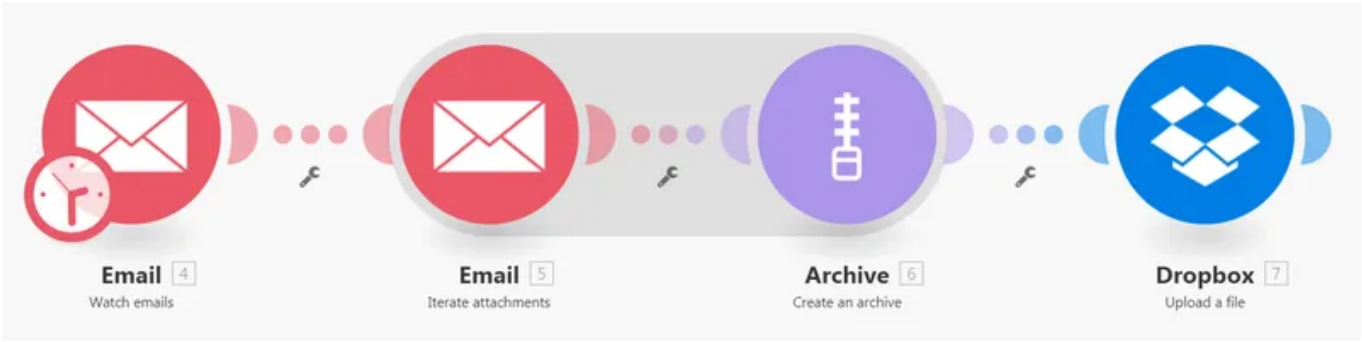
If you need to access items from bundles outputted from the source module and any other modules between the source module and the aggregator module, make sure to include them in the aggregator's setup, e.g. in the *Aggregated fields* field in the setup of the **Array aggregator** module.

Example

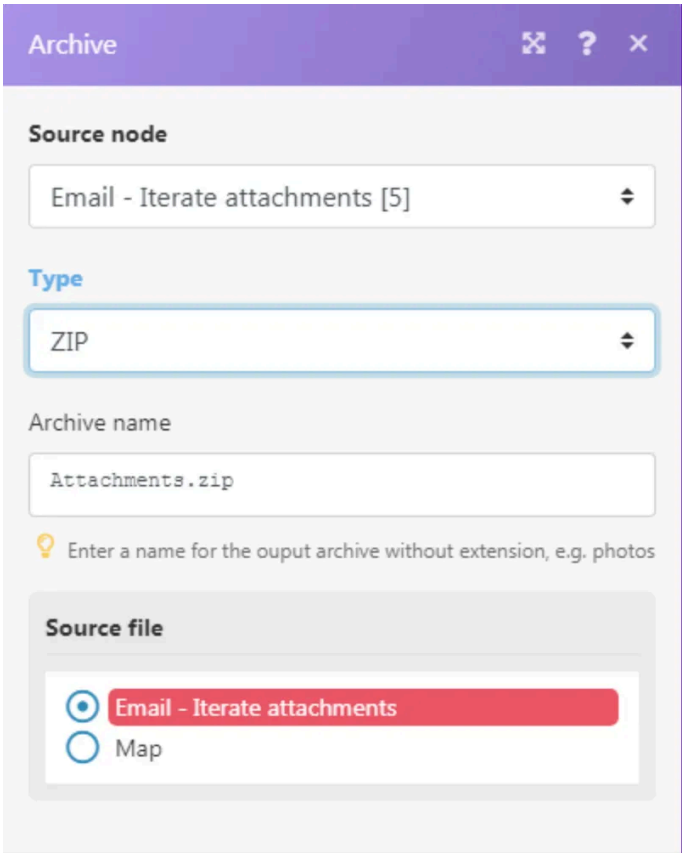
Use case: Zipping all email attachments and uploading the ZIP file to Dropbox

The scenario below shows how to:

1. Watch a mailbox for incoming emails: **Email > Watch emails** trigger will output a bundle with item `Attachments[]` , which is an array containing all the email's attachments.
2. Iterate the email's attachments: **Email > Iterate attachments** iterator takes the items from the `Attachments[]` array one by one and sends them further as separate bundles.
3. Aggregate the bundles outputted by the **Email > Iterate attachments** module: **Archive > Create an archive** aggregator accumulates all the bundles it receives and outputs a single bundle containing the ZIP file
4. Upload the resulting ZIP file to Dropbox: Dropbox > Upload a file obtains the ZIP file from the **Archive > Create an archive** module and uploads it to Dropbox.



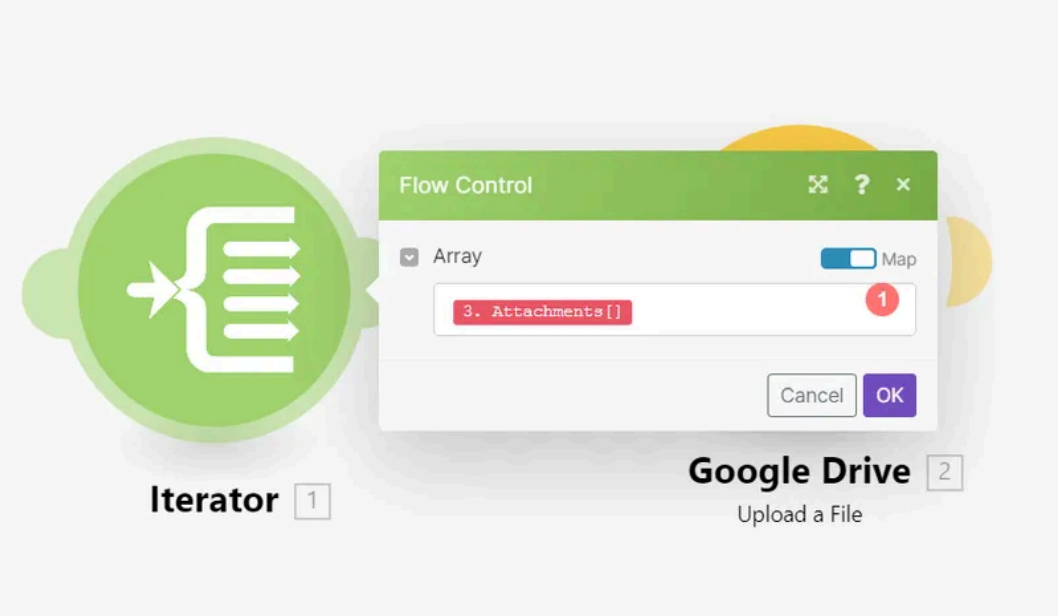
Below is a sample setup of the **Archive > Create an archive** aggregator:



2.4.2. Iterator

Iterator is a special type of module that converts an array into a series of bundles. Each array item will output as a separate bundle.

Setting up an iterator



Setting up an iterator is done in the same way as **setting any other module**. The Array field contains the array to be converted/split into separate bundles.

Examples:

Save email attachments to Google Drive

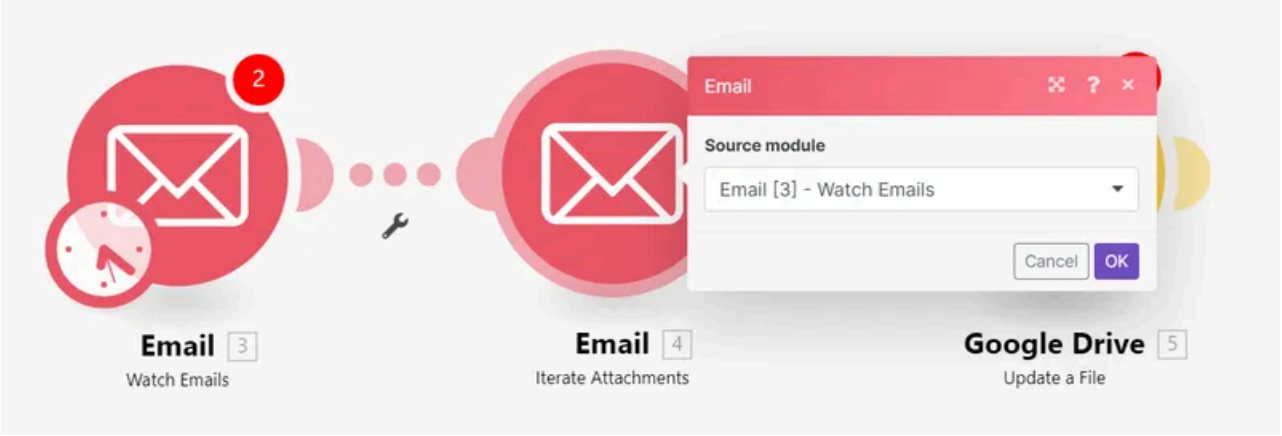
The scenario below shows how to retrieve emails with attachments and save the attachments as single files in a selected **Google Drive** folder.

Emails can contain an array of attachments. The **Iterator** module inserted after the first module enables you to handle each attachment separately. The **Iterator** splits the array of attachments into single bundles, each bundle with one attachment will then save one at a time in a selected Google Drive folder. The **Iterator** module set up is shown above - the Array field should contain the `Attachments[]` array.



Specialized iterators

For your convenience, many Make apps offer specialized iterator modules with a simplified setup. For example, the **Email** app contains the special iterator **Email > Iterate attachments** that will produce the same results as the general **Iterator** without having to specify the array, just the source module.



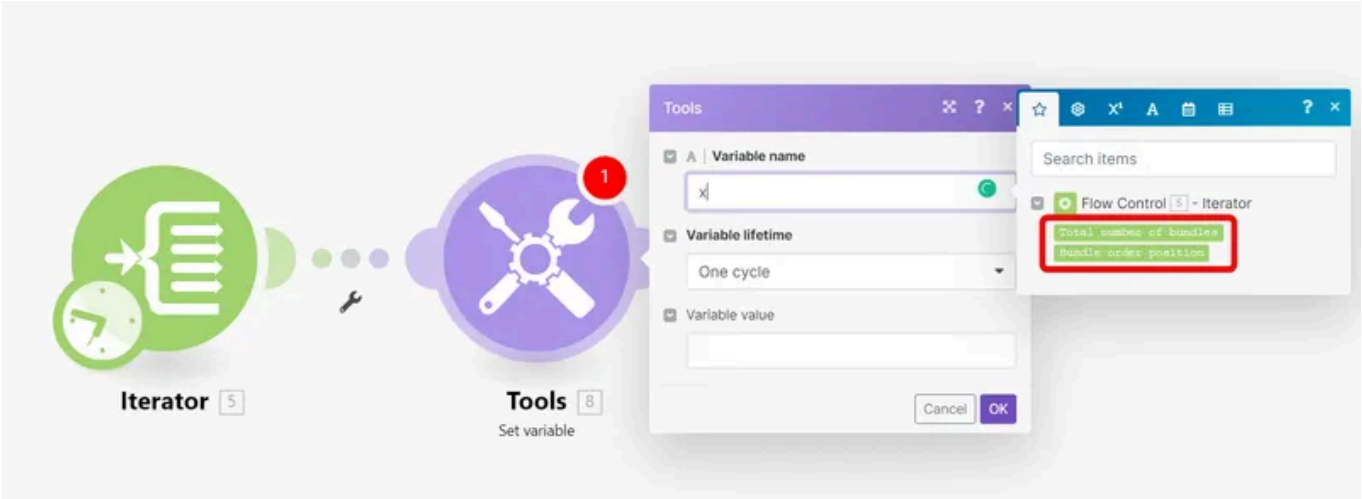
Learn when to use an Iterator in your scenarios

The video below is module 1 of the 3 part lesson titled **Iterator and Array Aggregator**. It explains the purpose of the Iterator and the Array Aggregator and with the help of a sample scenario, explains when to use an Iterator and what to do with the output.

Iterator & Array Aggregator | Part 1: Introduction and Iterator Demo

Troubleshooting: Mapping panel does not display mappable items under the Iterator module

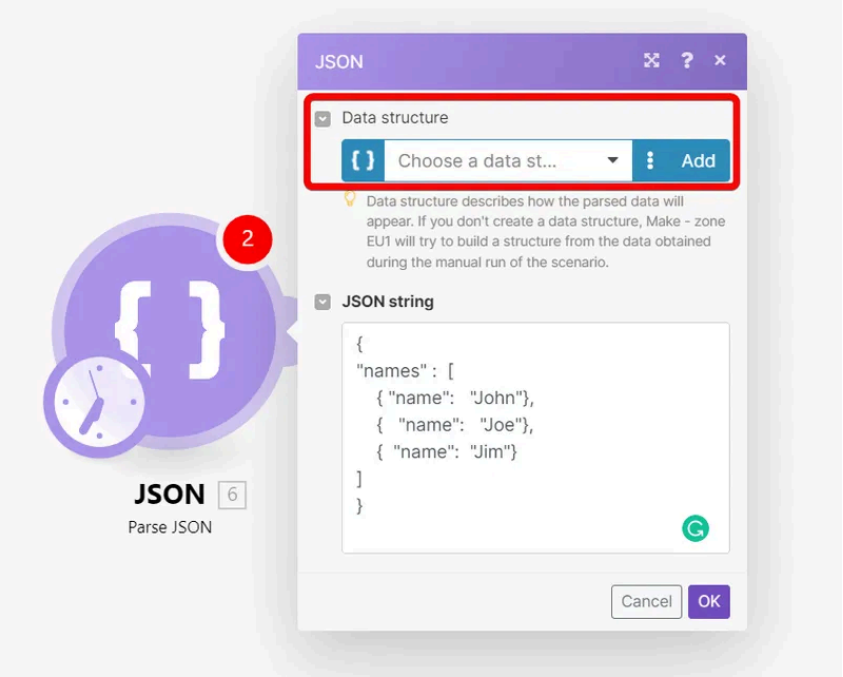
When an **Iterator** does not have information about the structure of the array's items, the mapping panel in the modules following the **Iterator** will display only two items under the **Iterator**: `Total number of bundles` and `Bundle order position`:



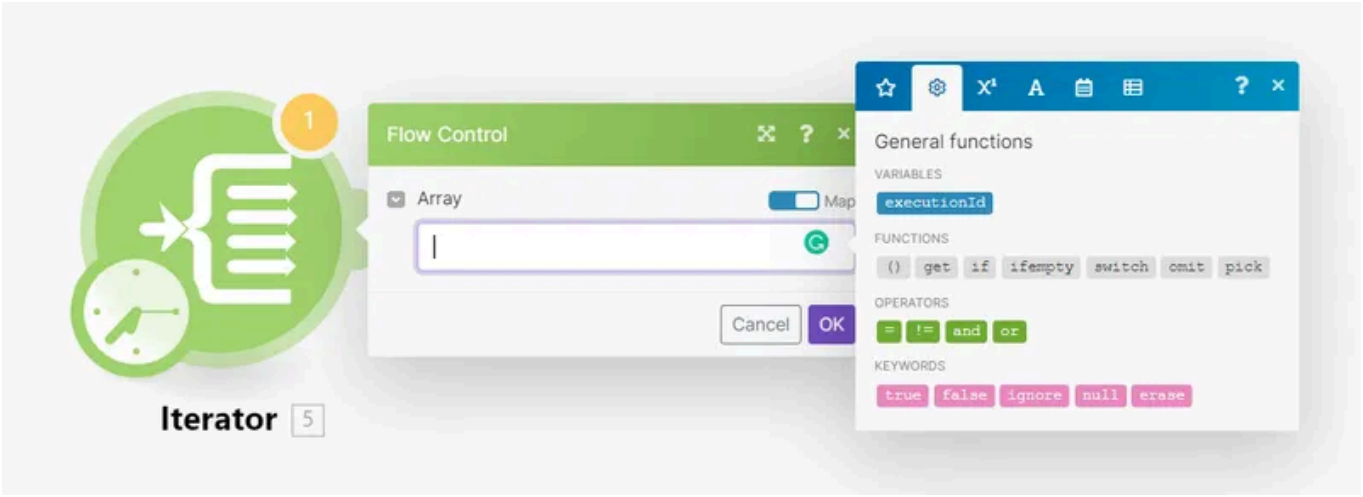
The reason for this is that in Make each module is responsible for providing information about items it outputs so these items can be properly displayed in the mapping panel in the following modules. However, there are several modules that might be unable to provide this information in some cases, e.g. **JSON** > **Parse JSON** or **Webhooks** > **Custom Webhook** modules with missing **data structure**.

The solution is to manually execute the scenario to make the module learn about the items it outputs so it can provide the information to the following modules.

For example, if you have a **JSON > Parse JSON** module without a data structure as below:



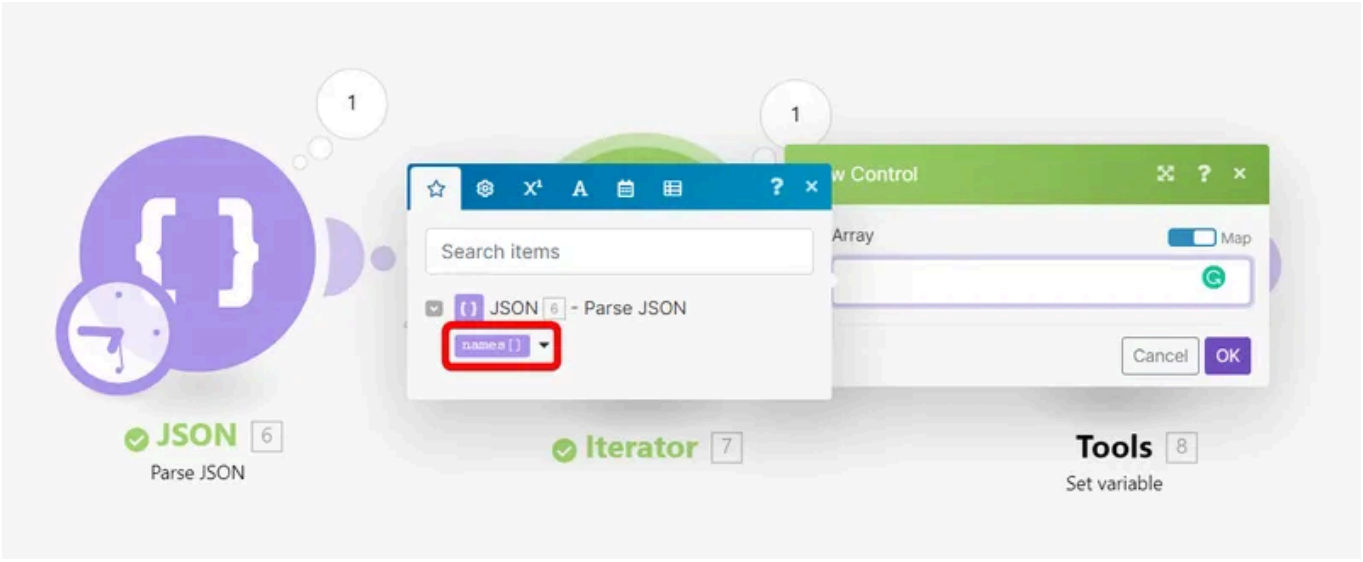
And then if you connect an **Iterator** module to it, you will not be able to map the output of the module to the *Array* field in the setup panel of the **Iterator**:



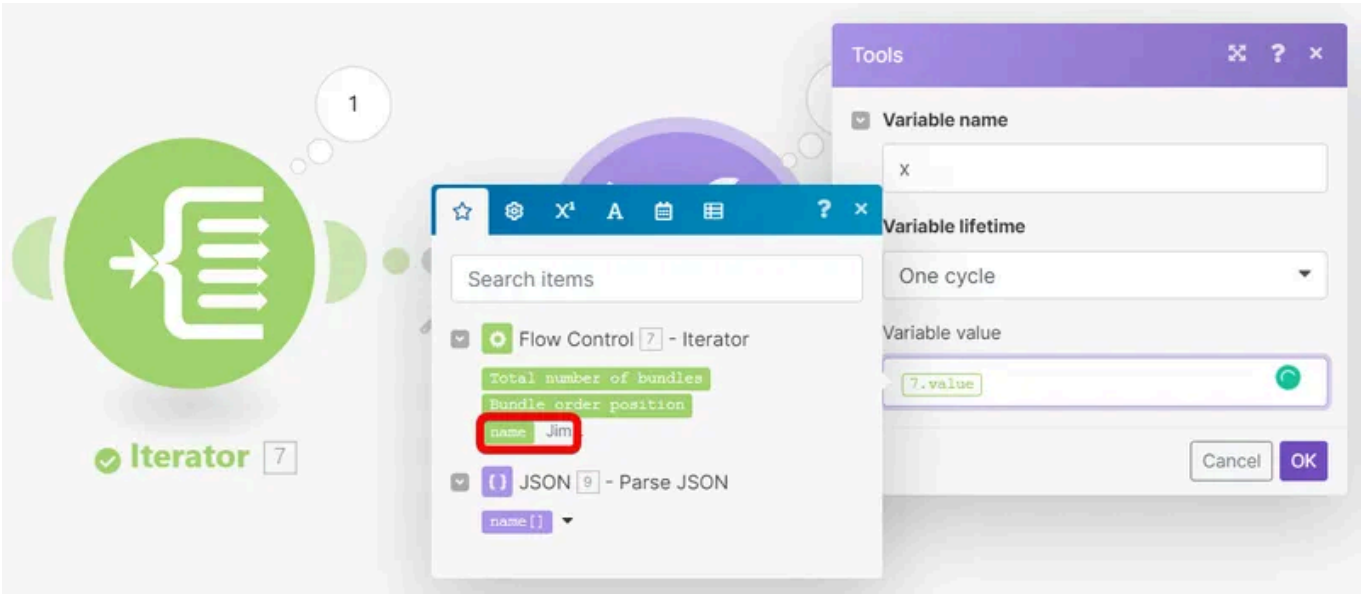
To resolve this, just manually start the scenario in the Scenario editor. You can un-link the modules after the **JSON > Parse JSON** module to prevent the flow from proceeding further or right-click the **JSON > Parse JSON** module and choose "Run this module only" from the context menu to execute only the **JSON > Parse JSON** module.

Once the **JSON > Parse JSON** has been executed, it learns about the items it outputs and provides this information to all the following modules including the **Iterator**. The mapping

panel in the **Iterator's** setup will then display the items:



Moreover, the mapping panel in the modules connected after the **Iterator** will display the items contained in the array's items:



In summary: if you cannot see some items in a module's mapping panel, simply run the scenario once so all the modules can learn about the items they output and provide this information to the following modules.

2.4.3. Router

A router allows you to branch the scenario flow into several chains of modules. Each route processes the data differently according to the condition you set. Filters help you to determine conditions via different operators such as `less than`, `greater than`, and so on.

Order routes in the sequence you want and set up a fallback route that will process data that doesn't fit other routes.

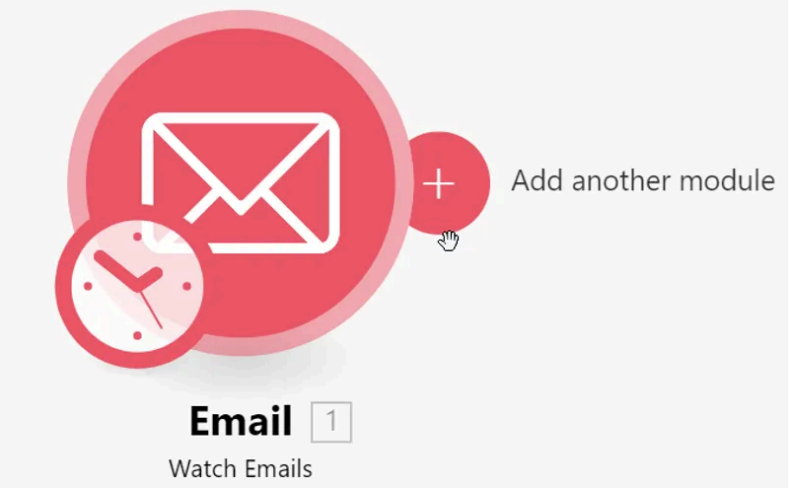
> See our scenario template for the **Controlled distribution of data flow**.

Adding a router to a scenario

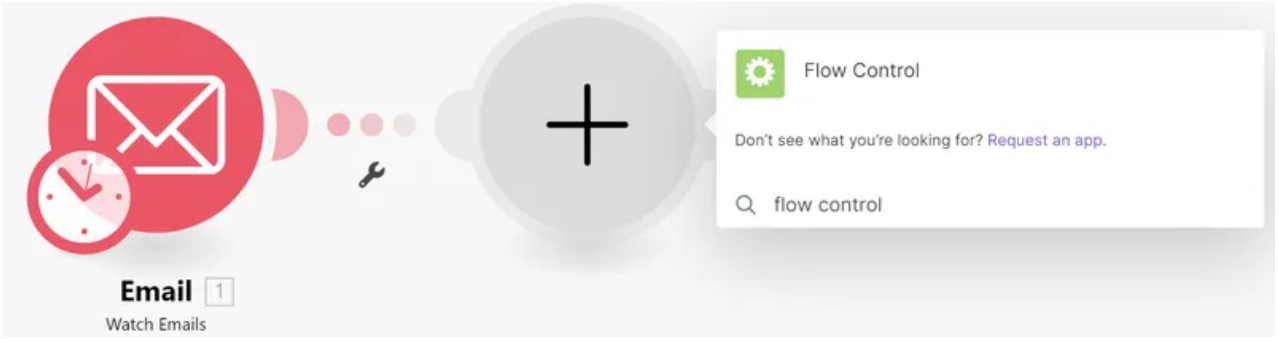
You can add a router in two different ways:

- Connect a router to a module:

1. Click **Add another module**.



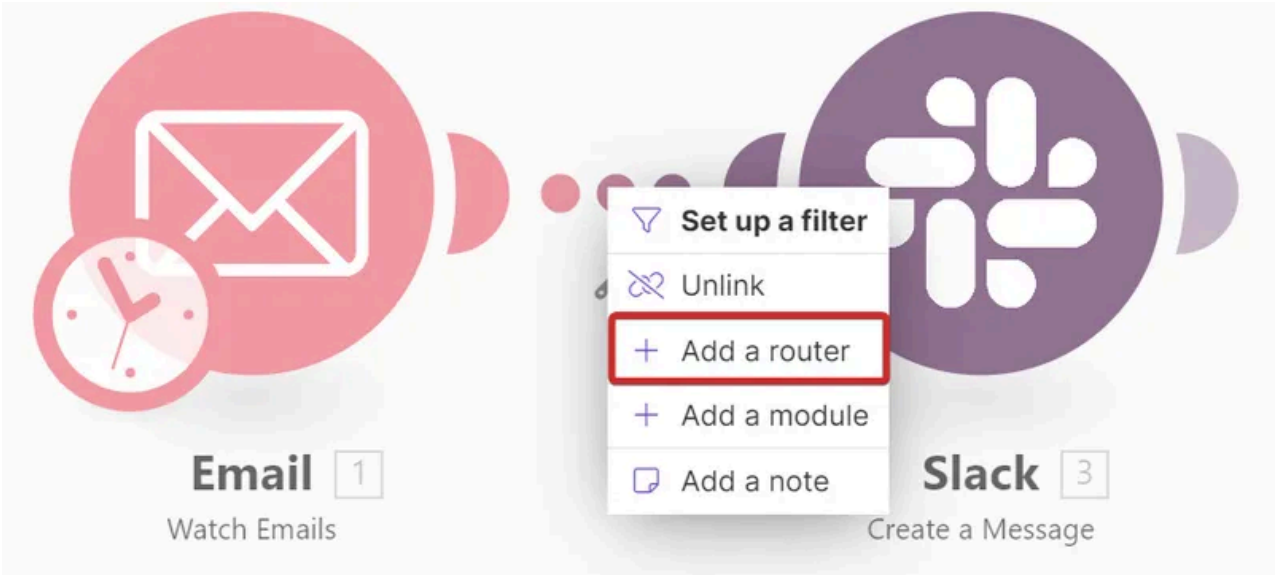
2. In the search box, type **Flow controls** and click it.



3 Select **Router**.

- Insert a router between two modules:

1 Right-click the bridge between two modules, and select **Add a router**.



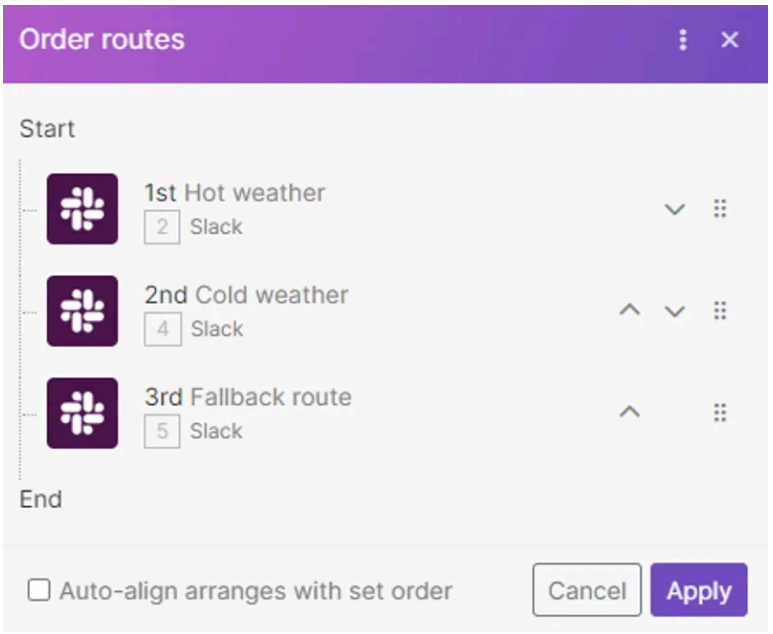
Order routes

You can set the order of routes in which Make processes them in the scenario.

This example shows the router that determines which hint to send you on Slack according to tomorrow's weather.

1 Click the router that contains the routes you want to order.

2 Right-click and select **Order routes**. A window appears.



3 Click arrows and move routes according to your needs.

4 *Optional*. Select **Auto-align arranges with set order** to visually arrange modules on the scenario canvas according to the set order.

5 Click **Apply**.

> Routes are processed sequentially, not in parallel. Make won't process the second route unless it finishes processing the first one.

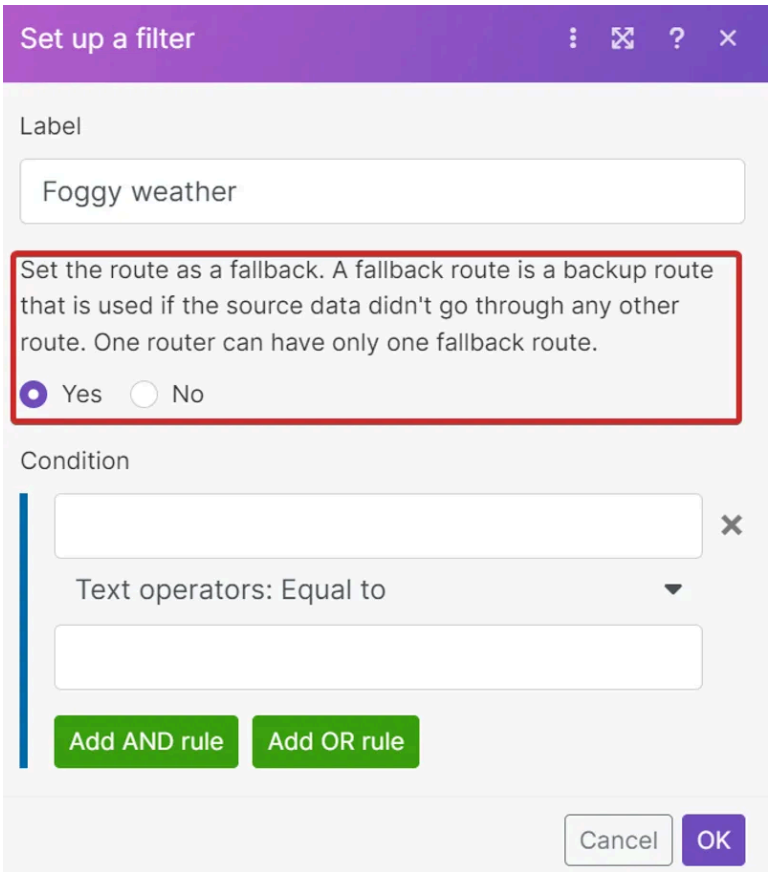
The fallback route

A fallback route processes data that doesn't fit the condition of all other routes. You can mark a route as a fallback if you want it to be executed last.

> You can set up a filter for a fallback route same as for other routes.

To set up a fallback route, follow the steps:

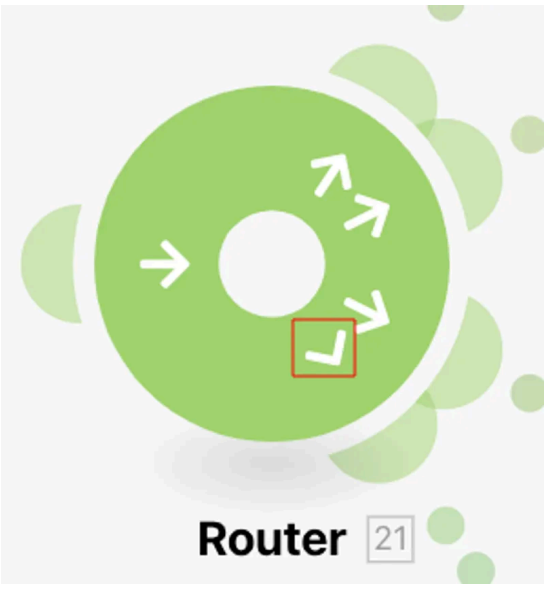
1 Click the route you want to mark as a fallback. A filter window appears.



2 Select **Yes**.

3 Click **Save**.

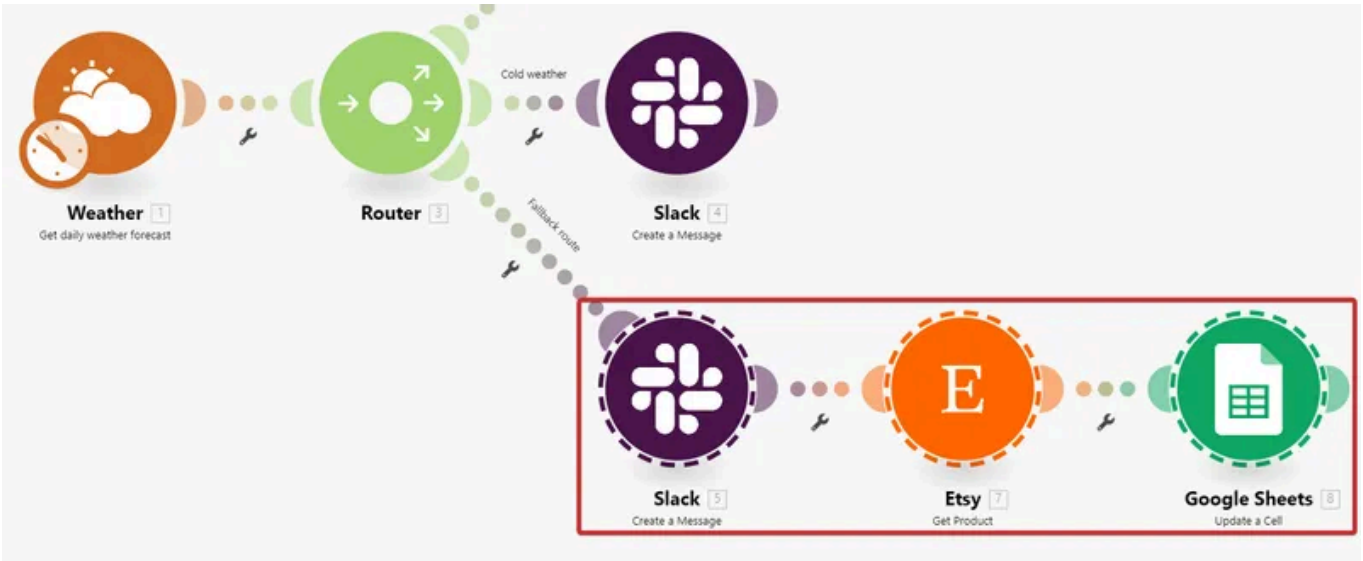
You can recognize the fallback route by the special arrow icon on the router module:



Select a whole branch

You can manage all modules in the branch at once.

Click the route menu, then click **Select whole branch**. It selects all the following modules.



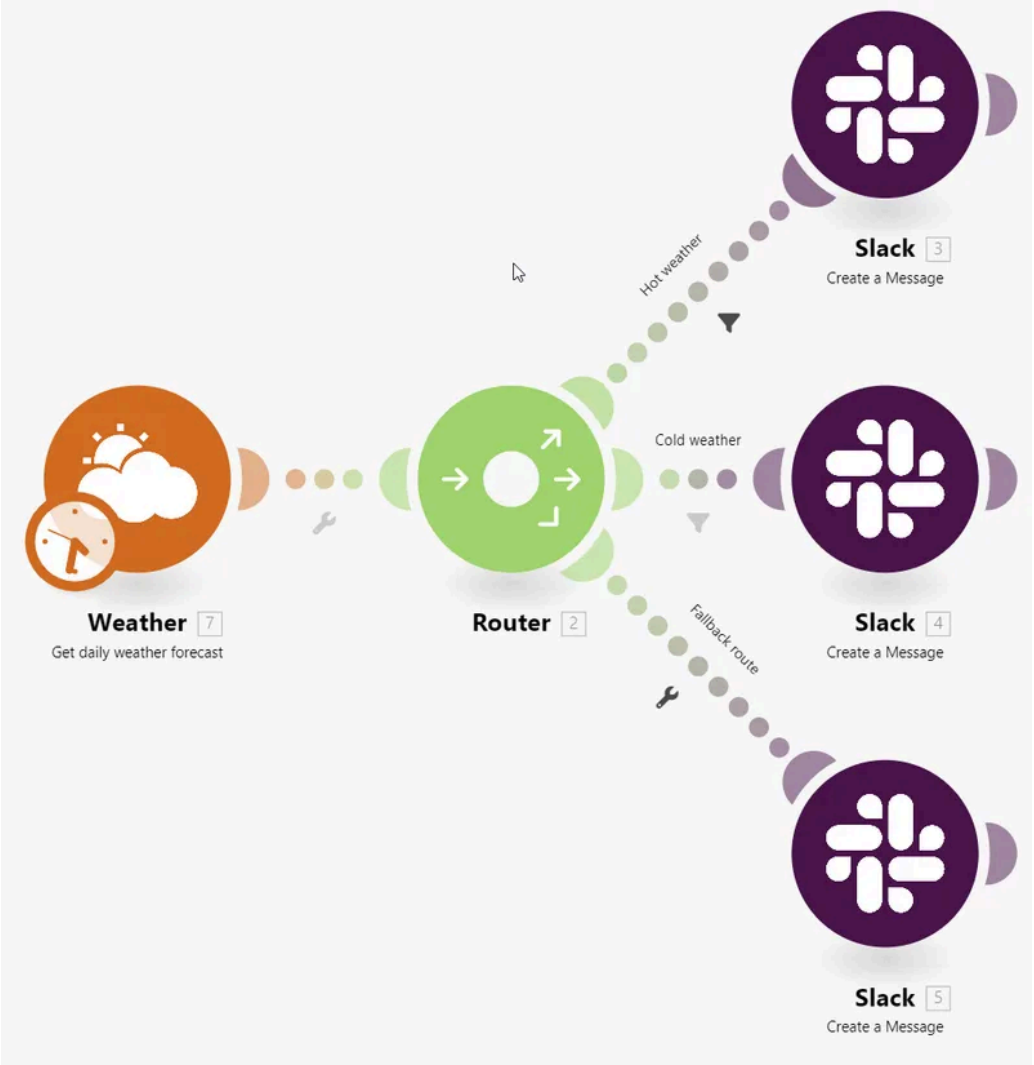
You can copy or delete all selected modules at once.

Example of a router with a fallback route

You need to receive a message on Slack depending on tomorrow's weather:

- if the weather is hot, the message is `wear shorts`.
- if the weather is cold, the message is `wear a jacket`.
- if the weather is neither hot nor cold, the message is `better stay at home`.

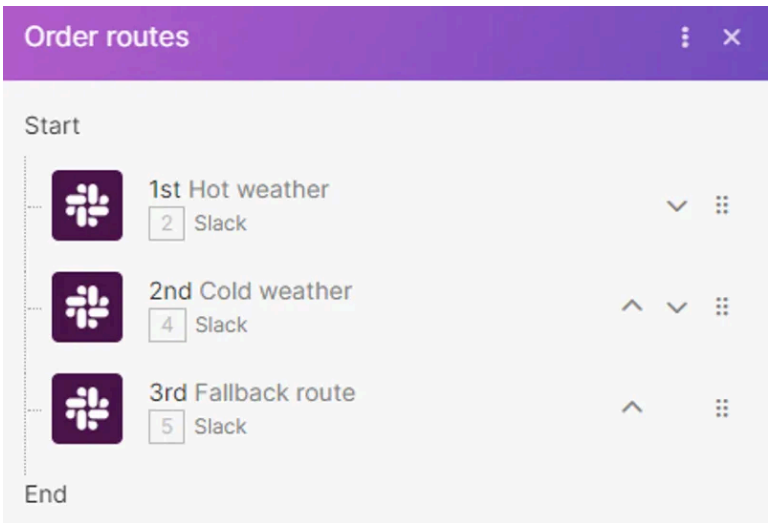
The scenario looks like that:



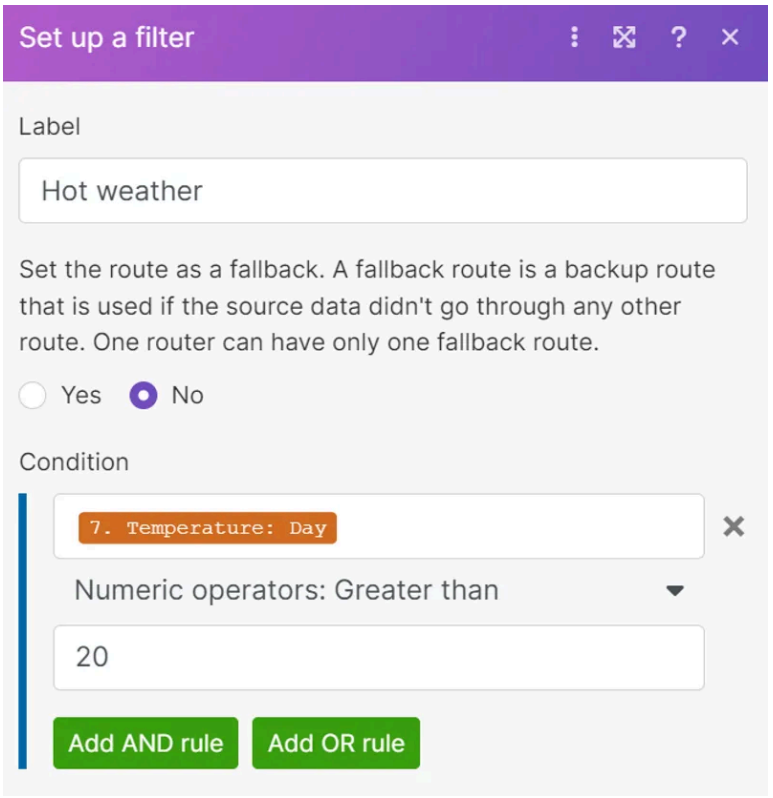
The scenario flow is:

1. The Weather module gets data about tomorrow's weather.

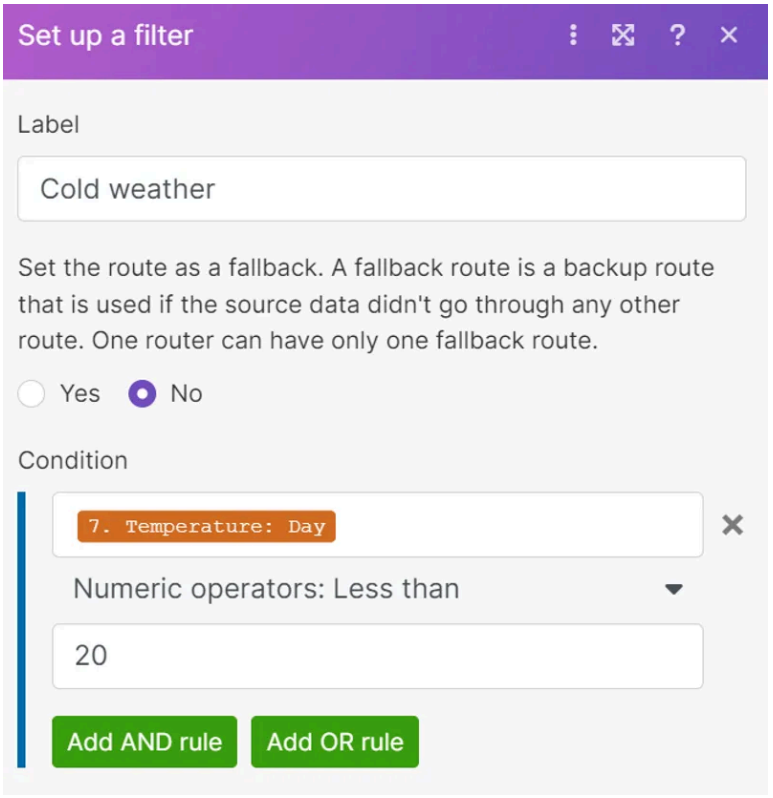
2. Data goes to the router that processes routes in the determined order:



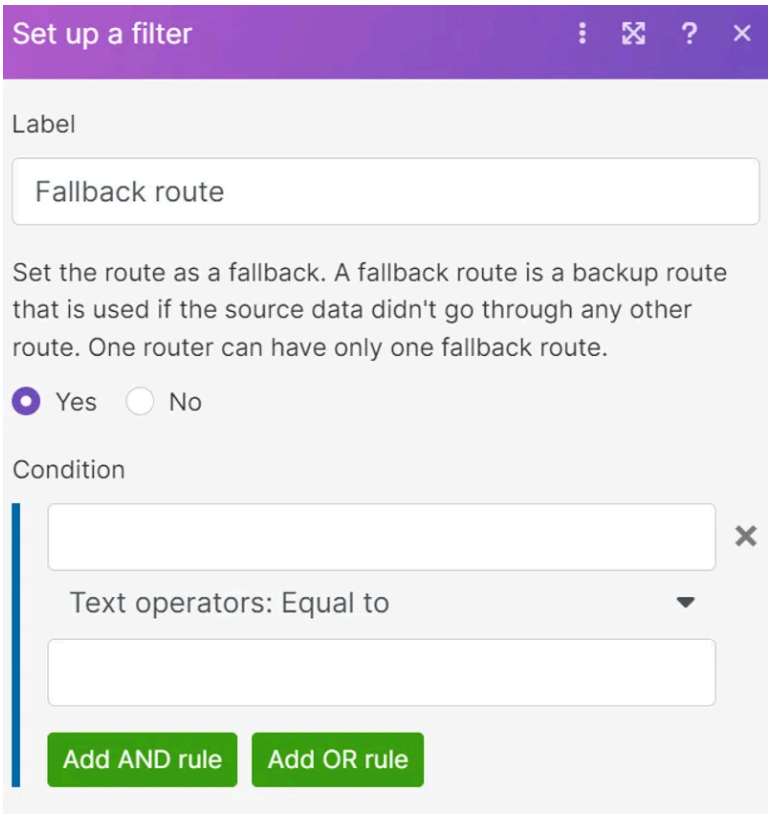
a. The **Hot weather** route sends the message to Slack, if data fits the filter condition:



b. The **Cold weather** route sends the message to Slack, if data fist the filter condition:



c. The **Fallback** route sends the message to Slack, if data doesn't fit previous routes.


Set up a filter
Label
Fallback route
Set the route as a fallback. A fallback route is a backup route that is used if the source data didn't go through any other route. One router can have only one fallback route.
Yes
No
Condition
Text operators: Equal to
Add AND rule
Add OR rule

2.5. Resources

Learn about different keyboard shortcuts in this article:

→ **Keyboard shortcuts**

→ **Glossary** (coming soon...)

2.5.1. Keyboard shortcuts

You can save time when editing scenarios with keyboard shortcuts.

Make features these shortcuts:

Shortcut	Action
+	Zoom in
-	Zoom out
Win: Ctrl+S **Mac:** ⌘+S	Save Make
Win: Ctrl+C on module **Mac:** ⌘+C on module	Copy module
Win: Ctrl+V **Mac:** ⌘+V	Paste module
Space+drag on module	Move all modules
Win: Ctrl+Enter **Mac:** ⌘+Enter	Run once
Shift+drag	Create a box to select multiple modules

3. Explore more

Discover powerful tools and features to automate without limits:

Scenarios

Dive deeper into scenario capabilities to build on the next level

Connections

Find more connection options including our on-premise agent

Tools

Build complex scenarios with the help of these advanced tools

Functions

Transform and format data using our range of functions

Data stores

Learn how to use Make's built-in database tool to store and manage data

Developers

Discover the Make DevTool to debug your scenarios like a pro

3.1. Scenarios

Dive deeper into scenario capabilities in these articles:

→ **Scenario settings**

→ **Scenario inputs and outputs**

→ **Subscenarios**

→ **Create scenario templates**

→ **Custom scenario properties**

→ **Variables**

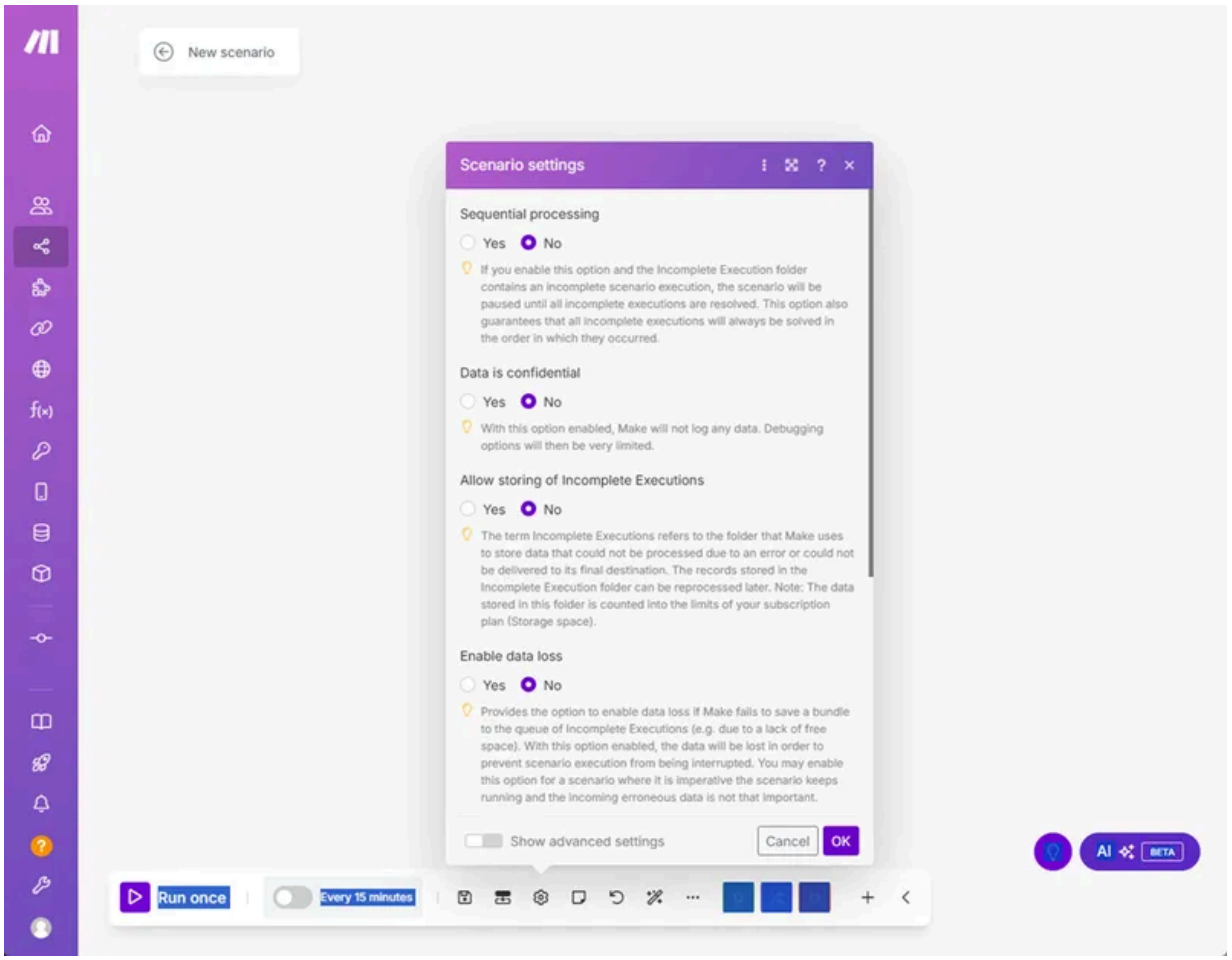
→ **Incomplete executions**

→ **Scenario execution, cycles, and phases**

→ **Restore a previous scenario version**

3.1.1. Scenario settings

You can access the scenario settings panel by clicking the gear icon in the scenario editor. Here you can set various advanced settings.



Sequential processing

You can allow Make to store information about incomplete Make executions.

The sequential processing setting determines how Make processes recurring incomplete executions. The **incomplete executions** folder must contain Make data.

- If enabled, Make stops executing the scenario until you resolve all incomplete executions. This option guarantees that Make will always solve incomplete executions in sequential order.
- If disabled, the scenario continues to run according to its **schedule**, regardless of errors.

> Sequential processing also applies to **webhooks**. By default, Make processes webhooks in parallel. When you enable sequential processing, Make waits until the previous execution is complete before starting the next one.

Data is confidential

After a scenario executes, you can **display information** about data processed by the modules. This happens by default.

If you do not want to store this information, enable this setting.

> If enabled, there are very limited options to solve errors that occur in a scenario execution.

Allow storing of incomplete executions

This setting determines what happens if a scenario run encounters an **error**. You can choose how Make will process the data.

- If enabled, the scenario is paused and moved to the **incomplete executions** folder. This gives you the possibility to fix the issue and continue from where the scenario stopped.
- If disabled, the scenario run stops and starts a **rollback** phase.

You can resolve each incomplete execution either manually or automatically.

> The data in this folder counts towards the storage limits of your **subscription plan**.

Enable data loss

Make may fail to save a data bundle to the queue of **incomplete executions** (e.g. due to a lack of free space). With this setting enabled, Make does not save the lost data. This is to prevent interruptions in the Make execution.

This option is well-suited for scenarios where continuous execution is the highest priority. The incoming data is less important.

Modules can encounter files larger than the **maximum allowed size**. In this case, Make proceeds according to the enable data loss setting and displays a warning message.

> The maximum file size depends on your **subscription plan**.

Auto commit

This setting applies to **transactions** and defines the way to process a scenario. This setting is enabled by default.

- If enabled, the **commit** phase on each module starts immediately after the operation phase. Data is committed right away and cannot be restored in the case of an error.
- If disabled, no commit occurs until operations are executed for all modules.

> Not every module supports transactionality. Only modules marked with the tag 'ACID' support transactions.

Commit trigger last

This setting defines the module **commit** order after a successful scenario operation phase. This setting is enabled by default.

- If enabled, the commit phase skips the trigger and processes that module last.
- If disabled, the commit phase occurs in the default order.

Max number of cycles

This setting defines the maximum number of cycles allowed during a scenario execution.

Setting more **cycles** can be useful when you want to prevent connection interruption to third-party services. This can also ensure all records are processed within the scenario run.

If you execute the scenario manually by clicking the **Run once** button, the setting is ignored and **only one cycle** will be performed.

Number of consecutive errors

This setting defines the maximum number of consecutive execution attempts before the scenario deactivates (though there are exceptions listed in the **error handling overview**.

> If a scenario starts with an instant trigger, the setting is ignored and the scenario is deactivated immediately once the first error has occurred.

3.1.2. Scenario notes

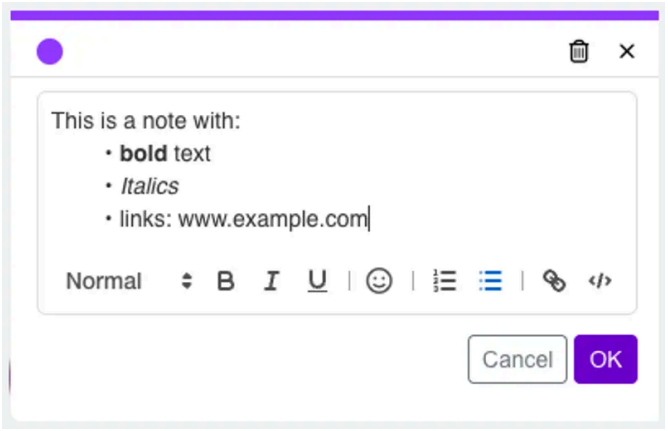
Scenario notes allow you to document and communicate essential details about your scenario with other Make users. Notes serve as an effective way to provide context, instructions, or explanations about different parts of your scenario.

> Before you start using notes, keep in mind:
>
> - You can attach a note to any module or any route in your scenario.
> - Each module or route can contain up to 200 notes.
> - Each note has a 10,000-character limit.
> - When you use a lot of formatting, like emojis and links, you might reach the note length limit faster than with no formatting.
> - Make stores your notes in HTML format, which uses extra characters to set formatting.

Create notes

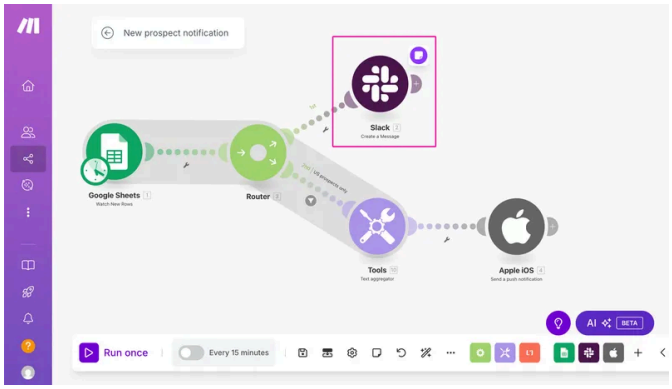
How to add a note to a module or route:

1. Right-click on the module or route where you want to add a note.
2. Select **Add a note** from the context menu.
3. Enter your content in the text editor.

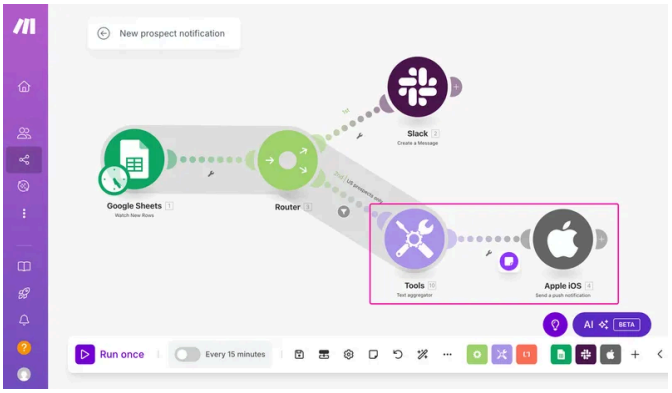


4 Optional: Use the formatting toolbar to style your text, add links, or insert emojis.

5 Click **OK** to save your note.



Note attached to a module



Note attached to a route

View notes

You have several options to view notes in your scenario:

What to do	What you'll see
Mouse over the note marker	A preview of the note
Click once on a note marker	The complete note content with scrolling for longer notes
Click the sticky note icon in the toolbar	The note panel showing previews of all scenario notes

Tips: Double-click on a note to open it in edit mode.

Manage notes

How to edit notes:

1. Double-click on the note you want to modify.
2. Make your changes in the text editor.
3. Click **OK** to save your updates or **Cancel** to discard them.

How to delete notes:

1. Open the note you want to remove.
2. Click the trash icon in the top-right corner of the note window.
3. Confirm you want to remove the note when prompted.

When you delete a note, this action is permanent and cannot be undone.

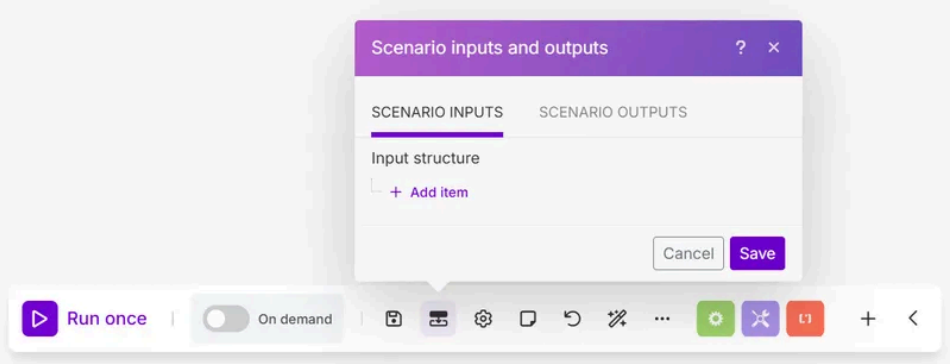
Best practices

- Use notes to document complex logic or explain non-obvious workflow decisions.
- Add implementation details that might help other team members understand your approach.
- Document any specific requirements or dependencies for modules
- Explain the purpose of different routes to clarify workflow branching.
- Keep notes concise but informative for better team collaboration.

3.1.3. Scenario inputs and outputs

Scenario inputs and outputs let you control what data goes into your scenario and what data comes out. They define the information that scenarios expect and share when communicating with other scenarios or external systems.

To find scenario inputs and outputs, navigate to the Scenario Builder and click on the **Scenario inputs and outputs** icon on the toolbar.



What scenario inputs and outputs do

Scenario inputs define what data a scenario receives when triggered.

Scenario outputs define what data the scenario returns after execution.

When a scenario calls another scenario (subscenario), inputs specify what data gets passed to the subscenario, while outputs specify what data gets returned to the calling scenario.

When you need scenario inputs and outputs

You will need to **configure scenario inputs and outputs** when building scenarios that communicate with other scenarios or external systems. This includes scenarios that:

- Call other scenarios (subscenarios)
- Get triggered by external systems via API
- Function as tools in AI systems

Some cases when you need scenario inputs and outputs:

- Syncing customer information across multiple systems
- Adding leads to CRM and email lists while creating proposals
- Creating records, invoices, emails, or orders with specific input parameters

Where to use scenario inputs and outputs

Scenario inputs and outputs enable your scenarios to function as tools across different systems: subscenarios, APIs, Make AI agents, and Make MCP server.

Regardless of which system you are working with, scenarios with scenario inputs and outputs use three modules: **Call a subscenario**, **Start a subscenario**, and **Return output**.

Consider this flow to understand how these modules interact:

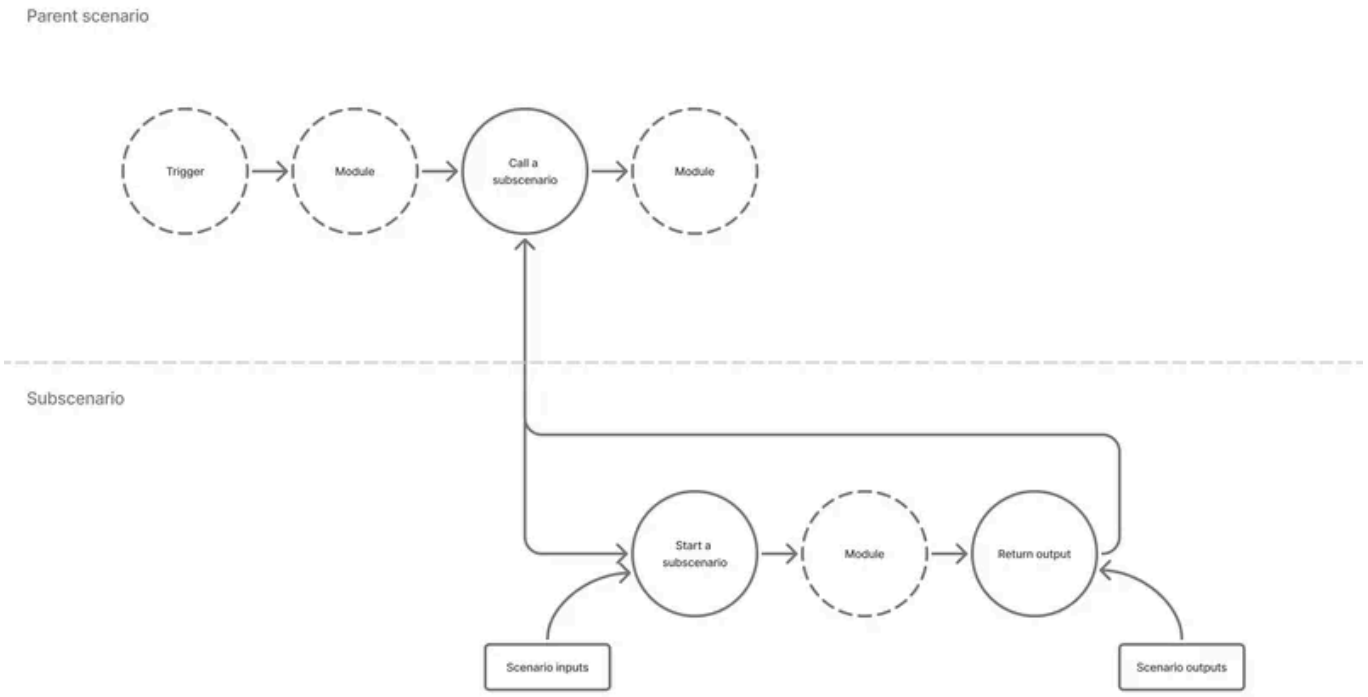
1. In the parent scenario, you add a **Call a subscenario** module, then select an existing (sub)scenario or create a new one.
2. In the subscenario, add a **Start a subscenario** module, which allows you to map scenario inputs to modules.
3. Add more modules depending on what the subscenario aims to do.
4. Add a **Return output** module to send output from the subscenario to the parent.

This flow applies to all cases using scenario inputs and outputs.

Subscenarios

Subscenarios break complex workflows into smaller, reusable components. Each subscenario handles a specific task or business function. Scenario inputs and outputs define the data exchanged between parent scenario and subscenarios.

The **Start a subscenario** module triggers the subscenario and passes input data. The **Return output** module returns data from the subscenario to the calling scenario.



Benefits of inputs and outputs in subscenarios:

- **Reusability:** Updates to a subscenario's inputs and outputs automatically apply to all scenarios using that subscenario.
- **Structured data sharing:** Inputs and outputs clearly define what data gets shared between scenarios.

Example: A scenario that retrieves restaurant menus could use subscenarios for individual restaurants. Each subscenario receives restaurant name/ID and date as inputs, then returns menu items, prices, and last-updated timestamps as outputs.

APIs

The **Make API** allows external systems to trigger scenarios. Scenario inputs and outputs define the structure of API requests and responses.

When calling a scenario via API, the defined inputs become the request body parameters. After execution, the scenario returns the defined outputs in the response structure.

Example: An external application triggers a restaurant menu scenario via API, sending date and restaurant ID as inputs. The scenario returns structured menu data as outputs.

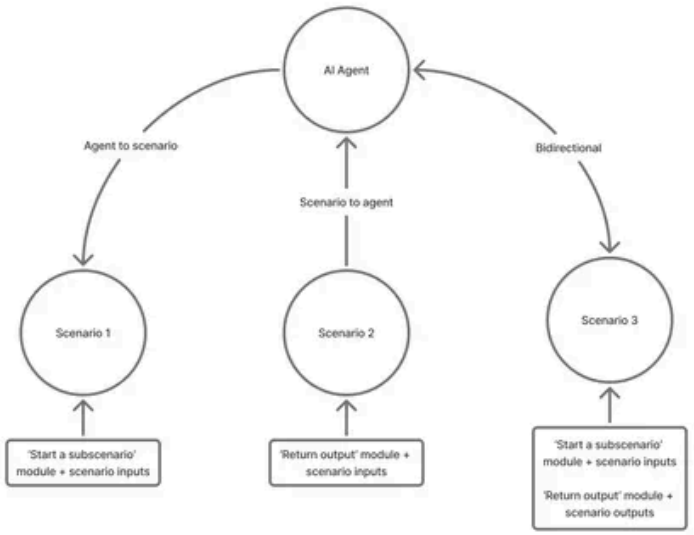
Make AI agents

Make AI agents can use scenario as tools. Scenario inputs and outputs help agents understand what data to provide and what to expect in return.

Input and output names and descriptions help AI agents select appropriate data, similar to how tool names and descriptions help agents choose the right tools.

Inputs and outputs in data flow patterns:

- **Agent to scenario:** When scenario inputs are defined
- **Scenario to agent:** When scenario outputs are defined
- **Bidirectional:** When both inputs and outputs are defined

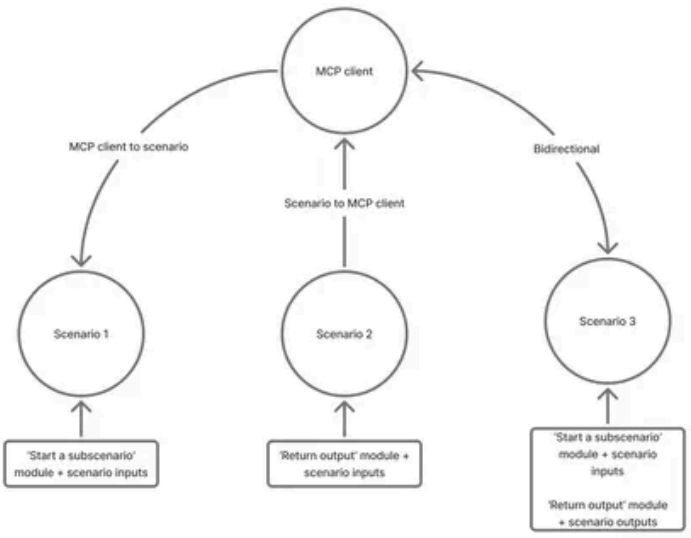


Example: A simple AI agent for retrieving daily restaurant menus uses subscenarios for each restaurant as its tools. Based on the user prompt, the agent calls the right scenario and returns a menu. The agent used the outputs structure in the subscenario to understand the menu data to give in the response.

Make MCP server

The **Make MCP server** enables scenarios to run from MCP-enabled AI systems like Claude. Scenario inputs and outputs serve as instructions for AI systems by defining the information to exchange during interactions.

Scenario inputs are parameters that AI systems fill when using scenarios. Scenario outputs are parameters for data that scenarios return to AI systems.



Example: When you Claude, an MCP client, about a restaurant's daily menu, Claude calls the relevant scenarios through the MCP server. The scenario outputs provide the menu information for Claude's response.

Scenario input and output types

To help you correctly define inputs and outputs, the following table outlines the input and output types available in scenarios, including descriptions and when to use them.

Type	Description
Array	A list of similar items. For example, a list of email attachments, where each item is a single file.
Collection	A group of related details about a single item. For example, a "Contact" collection containing a name, email address, and phone number.
Date	A specific point in time, including the year, month, and day. Date is in ISO 8601 format (e.g., 2015-09-18T11:58Z).
Text	Any type of plain text, from a word to a paragraph. Use this for names, descriptions, or messages.
Number	A numerical value. Use this for quantities, prices, ratings, or any other data that consists of digits.
Boolean	A type with only two options: true/false or yes/no. Good for toggles.
Select	A dropdown list of predefined options. Use this when you want a user to choose from a list of specific items, such as a country.
Dynamic collection	Use this type to map a collection/object or paste a valid JSON string into the field.

3.1.3.1. Create the structure of scenario inputs or outputs

Before using scenario inputs or scenario outputs, you need to define their structure. When creating the scenario input structure, you can set any parameter as required, which allows you to validate the structure of the input data.

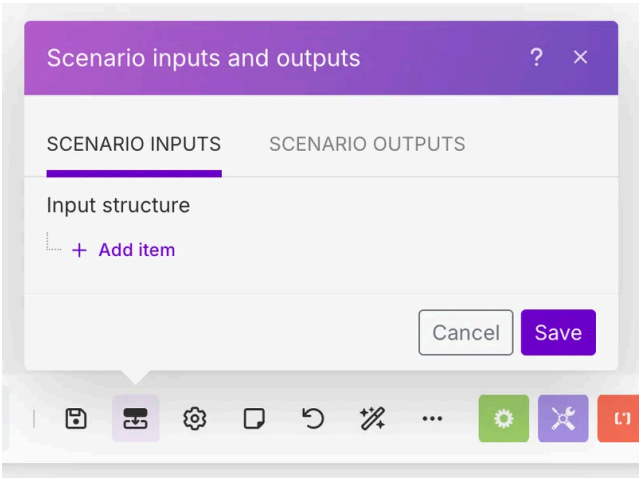
You can also add a description to each input parameter to document the data the parameter contains.

You can add scenario inputs and outputs anytime.

1. In the Scenario Builder, click the scenario inputs and outputs icon.



2. Select the **Scenario inputs** or **Scenario outputs** tab and click **Add item**.



3. Enter a value in the **Name** field. This field is mandatory.

   You can't use spaces or special characters in the name. You can use letters, numbers, and the underscore symbol. You can only start with a letter or underscore.

4 In the **Type** field, select the data type of the item from the dropdown menu.

- If you use the type **Select**, add options for the dropdown selection in the **Options** field.
- If you use the type **Array**, set the type of the array items in the nested **Type** field.
- If you use the type **Collection**, you can set the structure of the collection in the **Specification** field. Make validates the input data against the specification. If you keep the **Specification** empty, Make will accept any collection for the input data.
- If you use the type **Dynamic collection**, you can map a collection or a JSON string to the output field. The **Return output** module parses the JSON and returns it as output bundles. You can use the **Create JSON** module to create a custom data output.

5 Optional: In the **Description** field, add information about the input or output item. For example, how is the item used in the scenario or what information the item contains.

6 Optional: In the **Default** field, enter the default value that is used if the value is missing in the original input.

7 Under **Required**, select whether this input is required or not to start your scenario.

If you have required inputs, your scenario scheduling type must be **On demand**. For more information, see the **scheduling a scenario** article.

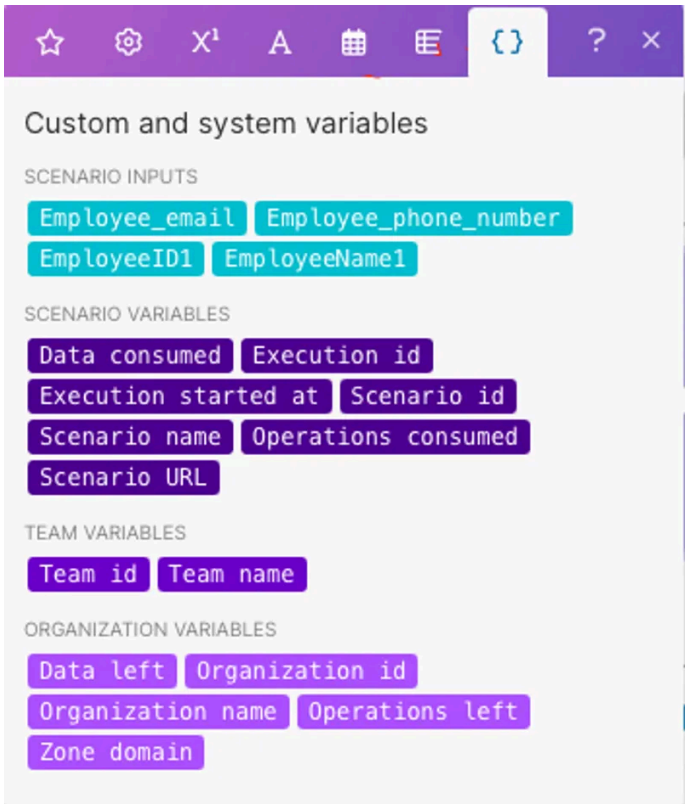
8 Under **Multi-line**, select how you want your text to be displayed if you selected **Text**, **Array**, **Collection** or **JSON** as the item data type**.**

Selecting **Yes** shows multiple lines of text. Selecting **No** shows only one line of text.

9 Repeat the process if you need to add more items.

10 Click **Save** in the **Scenario inputs** box and save your scenario.

You can use the scenario inputs and outputs as mappable values in your scenario. You can find them under the **Variables** tab in the mapping dialog.



To use scenario outputs, you need to add the **Scenarios** > **Return output** module. Check the next section.

3.1.3.2. Use scenario inputs

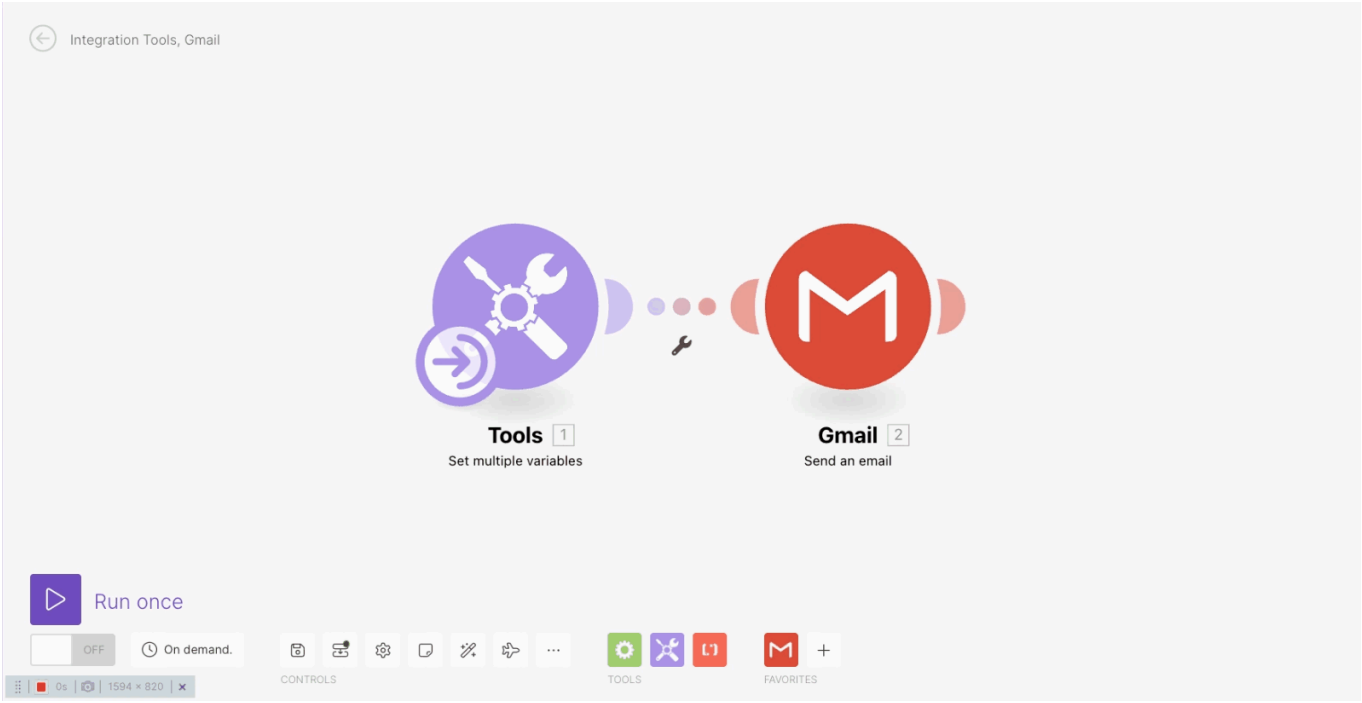
Once you define your scenario inputs, they appear as mappable values under the **Variables** tab in the mapping dialog.

You can use scenario inputs in the following ways:

- You can provide the inputs manually when you want to run the scenario. This is great for processes that you need to run on demand and that need human input and verification.
- You can send the inputs from another scenario. In this case, you need to use the **Call a subscenario** module to trigger the scenario.
- You can add the inputs to the request body when triggering the scenario with the **Make API**.

> For combining subscenarios and scenario inputs, we recommend you use the **Start a subscenario** module as the trigger for the child scenario.

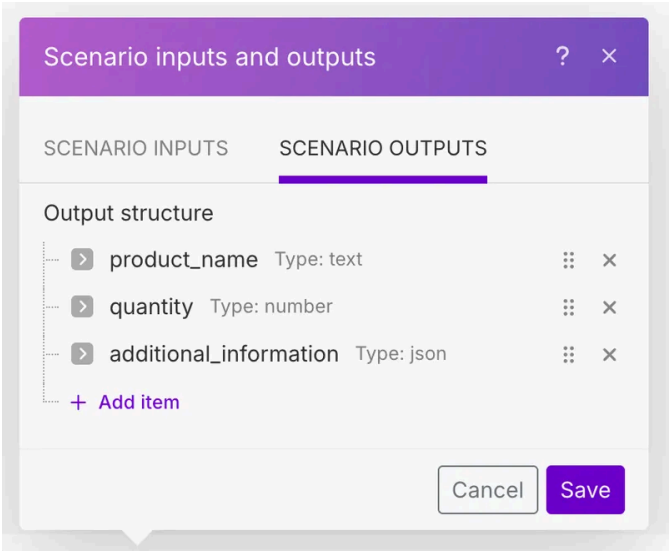
With defined scenario inputs, every time you trigger a scenario manually with the **Run Once** button, the scenario inputs window pops up. The scenario can run only if you fill in all the required fields with the correct data types.

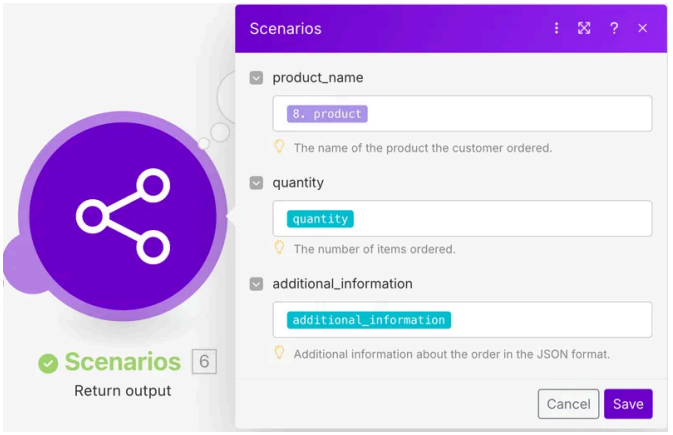


3.1.3.3. Use scenario outputs

Once you define your scenario outputs, you need to add the dedicated **Scenarios** > **Return output** module. The **Return output** module has module fields based on scenario outputs setting.

Map the data you want to output from your scenario to the **Return output** module fields.





> Make is not billing you for operations used by the **Return output** module. You can use scenario outputs as much as you like.

The **Return output** module always finishes the scenario run. It works similarly as the `return` statement in programming. You cannot add any subsequent modules after the **Return output** module in the current route.

Keep in mind that when using routers, a **Return output** module in a route finishes the scenario. The modules in the subsequent routes won't run.

> If you have multiple **Return output** modules in your scenario, Make runs only the one that is reached first in the scenario flow. Subsequent **Return output** modules don't run.

Scenario outputs are available to the entity that triggered the scenario:

- If the scenario was triggered through the **Call a subscenario** module, then scenario outputs are a part of the module output bundles.
- If the scenario was triggered through the Make API, then scenario outputs are in the request response.

3.1.3.4. Edit the structure of a scenario input or output

1. Navigate to the scenario inputs and outputs icon in the scenario builder.
2. Click on the item you want to edit.
3. Make your changes.

   Changing the scenario input name will cause the scenario to fail. If you need to change the input name, we recommend deleting the old item and adding a new one with the new name.

   If you change the inputs from not required to required, you need to change your scheduling to **On-demand**.
4. Click **Save**.

Once you save your changes, the changes update automatically everywhere you use the specific input mappable value.

> Editing the input or output structure item(s) in an active scenario might lead to data inconsistencies and scenario failure.
>
> If you are connecting the scenarios through the **Call a subscenario** and **Return output** modules, check the **Relation tree** tab to find related scenarios quickly. **Find out more about the Relation tree tab**.

3.1.3.5. Delete the scenario input or output fields

1. Navigate to the scenario inputs icon in the scenario editor.
2. Find the item you want to delete.
3. Click **X** to delete the item.
4. Click **Save**.

Editing the input or output structure item(s) in an active scenario might lead to data inconsistencies and scenario failure.

If you are connecting the scenarios through the **Call a subscenario** and **Return output** modules, check the **Relation tree** tab to find related scenarios quickly. **Find out more about the Relation tree tab**.

3.1.3.6. Related APIs

The Make API allows you to define scenario inputs and outputs programmatically, as well as use the API to pass input values to a scenario and run it.

In the API, the definition of scenario inputs and outputs is called **scenario interface**.

You can also use the **Call a subscenario** module of the Make app.

The following endpoints are available:

- **Get scenario interface**
- **Update scenario interface**
- **Run a scenario**

3.1.4. Subscenarios

Subscenarios allow you to link multiple scenarios together to form a **scenario chain**. When the first scenario in the scenario chain finishes, it triggers the next scenario. This is useful in workflows with different stages or processes that need to run in a specific order. Creating a scenario chain with subscenarios brings in the following features:

- **Run in sequence:** when the first scenario finishes, it automatically triggers the execution of the next scenario. This ensures that tasks are carried out in the desired order without manual intervention.
- **Data transfer:** The first scenario can send data to the second scenario. This allows for using the result of the first scenario as an input for the second scenario.

  For example, you can send an order number to a scenario that updates the warehouse inventory and to another scenario that sets up tracking of the order delivery.
- **Modularity:** You can split a complex scenario into multiple chained scenarios. By breaking down a complex workflow into smaller, manageable scenarios, it becomes easier to design, debug, and maintain. In addition, you can link the scenario with other scenarios in the future, to avoid rebuilding the same logic.

  For example, you can have multiple scenarios to send customers notification emails and order invoices. Each of these scenarios starts with a few modules retrieving user data. With scenario chaining, you can extract the repeating modules to a separate scenario and put it at the start of the scenario chain.

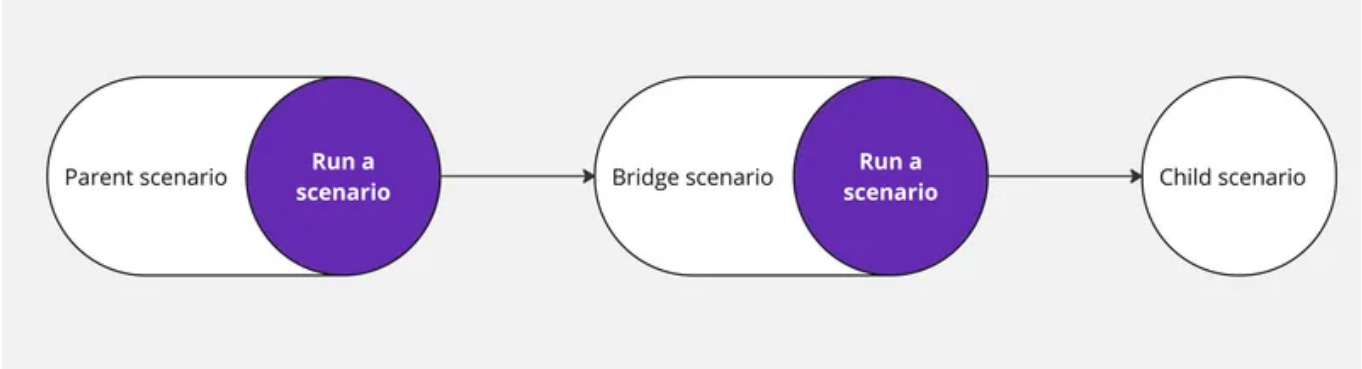
Types of scenarios in a scenario chain

There are three types of scenarios in a scenario chain:

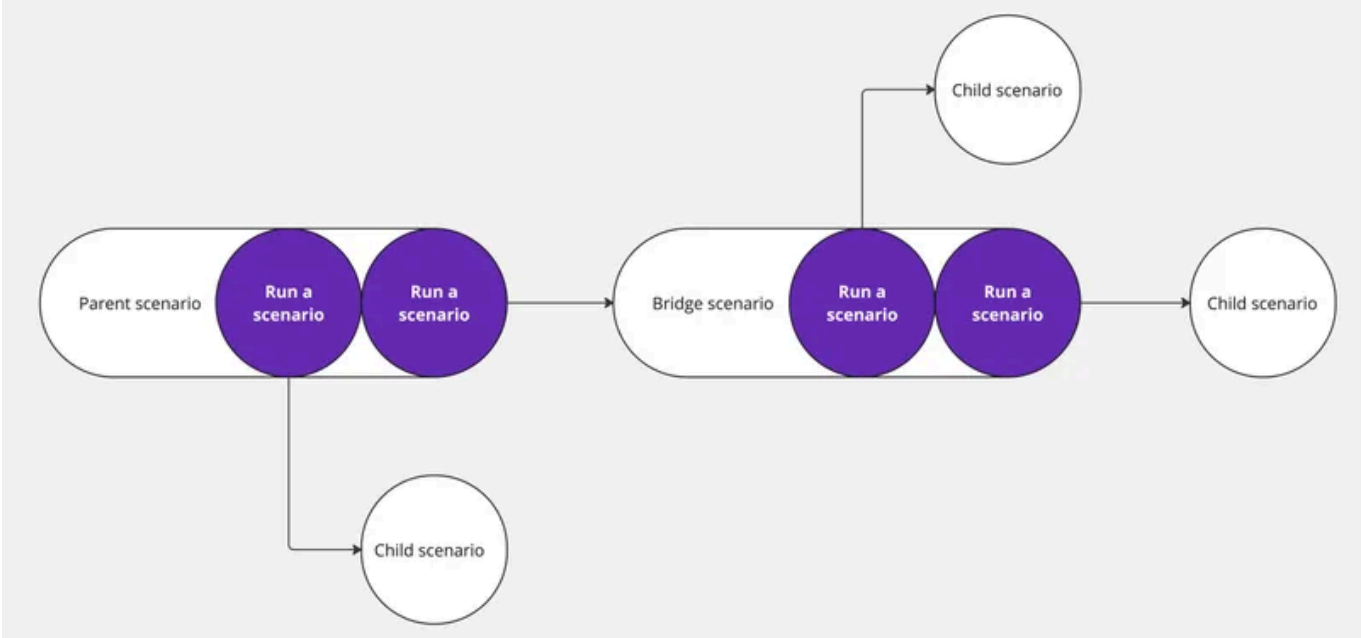
- **Parent** scenario
  - Starts the scenario chain.
  - Triggers bridge or child scenarios.
  - Can have multiple **Run a scenario** modules.
- **Bridge** scenario

    - Connects child scenarios or other bridge scenarios with parent scenarios in the scenario chain.
    - Is triggered by a parent scenario or another bridge scenario. In addition, the bridge scenario triggers another bridge scenario or a child scenario.
    - Can have multiple **Run a scenario** modules.
- **Child** scenario
    - Ends the scenarios chain.
    - Is triggered by parent or bridge scenarios.
    - Doesn't have any **Run a scenario** modules.

The following diagram shows the relation between parent, bridge, and child scenarios.



The scenario chains can even create branches, like this:



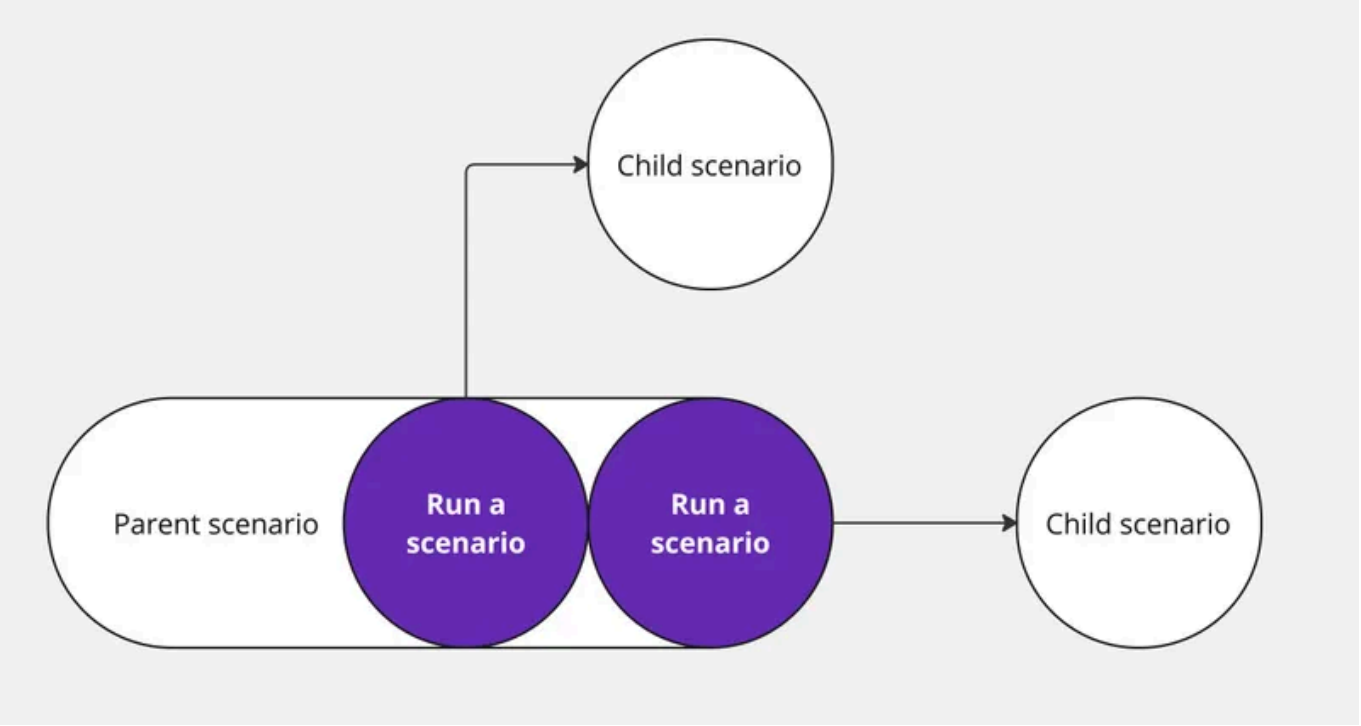
Create a scenario chain

To create a scenario chain, you need to create multiple scenarios. The following steps focus on a quick and simple showcase, where you don't need to build complicated scenarios to try out scenario chaining. Keep in mind that scenario chaining brings the most benefits when you want to split a complex scenario into multiple parts.

Overview of the steps to create a scenario chain:

1. Create the parent scenario.
2. Create the child scenarios.
3. Connect the scenarios with **Run a scenario** modules to form a chain from the parent and child scenarios.
4. Set the scheduling for the scenarios in the scenario chain.

The scenario chain can be visualized like this:



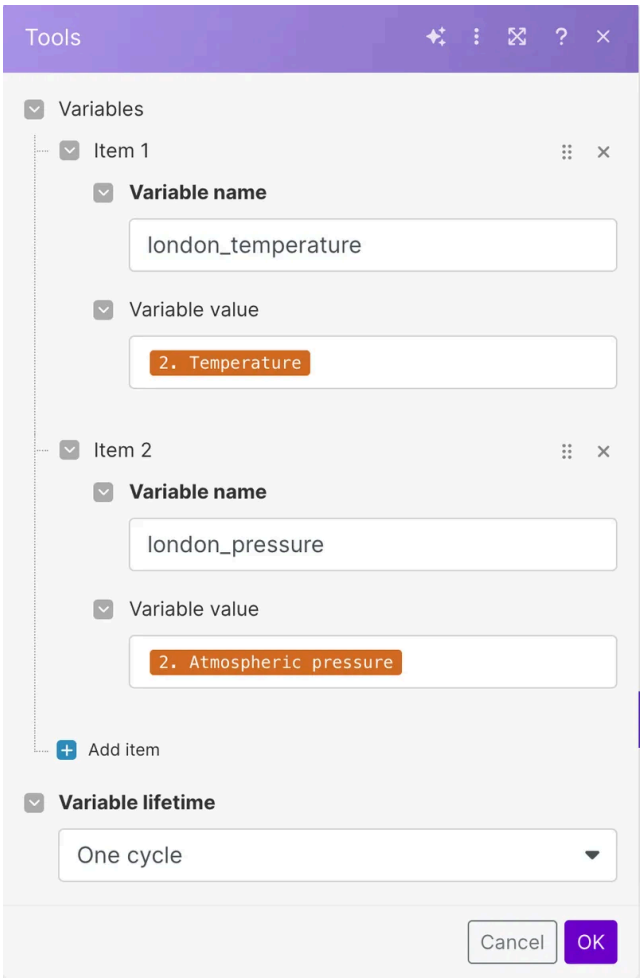
Create the parent scenario

In the parent scenario, we will use the **OpenWeather** app to get weather data. We will use the **Set multiple variables** module to set the data we will be sending to the child scenarios.

1. In Make, click **Create a new scenario**.
2. In the scenario editor, click the big plus icon and search for the **OpenWeather** app.

3 Connect the **Get current weather** module with the **Set mutliple variables** module. In the **Set multiple variables** module, create new variables and map the data you want to send to the child scenarios.

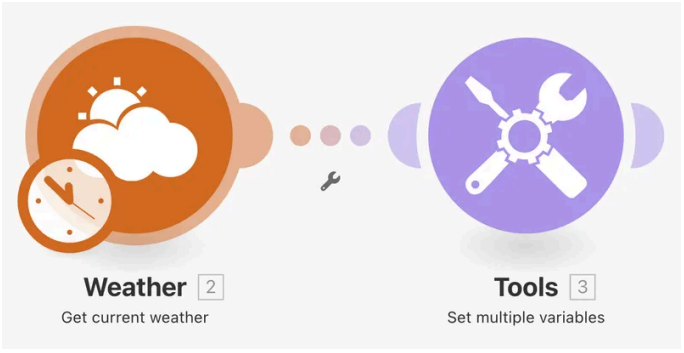
For example, map the **Temperature** and **Pressure** pills from the **Get current weather** module to the `london_temperature` and `london_pressure` variables:



4 Connect the **Get current weather** module with the **Set mutliple variables** module. In the **Set multiple variables** module, create new variables and map the data you want to send to the child scenarios.

You created a starting piece of the parent scenario in the scenario chain.

Your scenario should look like this:



Create the child scenarios

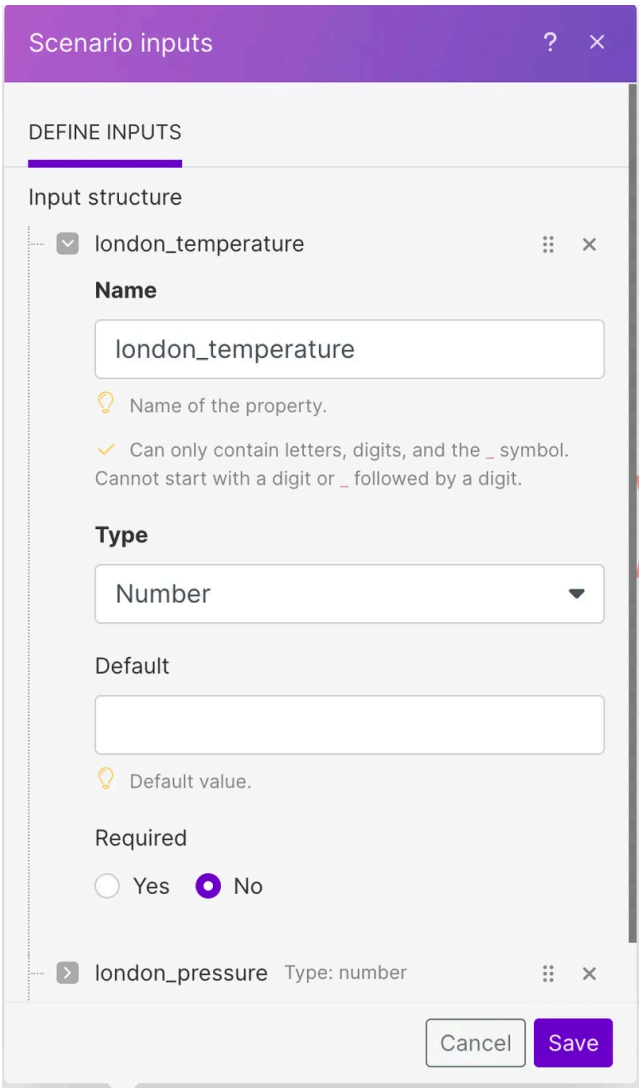
In the following steps, we will create two child scenarios. Both child scenarios will use **scenario inputs** to receive data from the parent scenario.

1. Click **Create a new scenario**.

2. In the scenario editor, click the big plus icon and search for the **Gmail** app.

   In search results, select the **Send an email** module.

3. Set up the inputs for the scenario. The scenario will expect the `london_temperature` and `london_pressure` variables as its inputs.



4 Configure the **Send an email** module. Use the scenario inputs in the email message.

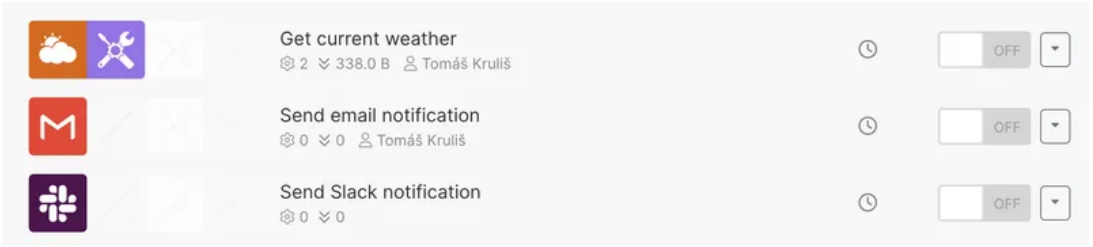


5 Confirm the module settings and save the scenario.

6 Create another child scenario.

For example, you can use another messaging app if you have created a connection for it, like the **Slack** app.

You have three new scenarios in your Make team, but they are not connected yet. Let's fix that.



Connect the parent scenario with the child scenarios

The following steps describe how to use the **Scenarios** > **Run a scenario** modules to connect the parent scenario with the child scenarios.

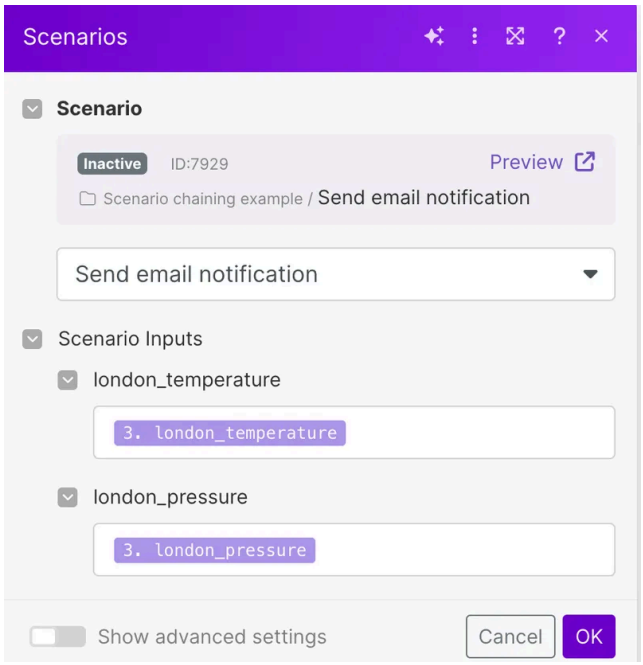
1. Go to the first scenario with the **Get current weather** and **Set multiple variables** modules.

   Add the **Scenarios** > **Run a scenario** module to the scenario.

2. In the **Scenario** field, select the scenario you want to trigger with the module. If you are following the example, select the scenario which sends the email notifications.

   Note that the scenarios are organized in scenario folders.

3. Make retrieves the scenario inputs of the scenario you select. Map the `london_temperature` and `london_pressure` variables to the scenario input fields:



Note that at the top of the module settings, there's the **Inactive** label informing you that the child scenario is not active. There's also the child scenario ID, the scenario folder and name and a **Preview** link that opens a new tab with the child scenario open in the Scenario Builder.

4 Click **OK** to confirm the settings and save the parent scenario.

5 Open the **Run a scenario** module settings again and click the **Preview** link to go to the child scenario.

Set the child scenario scheduling to **On demand**. Enable the scenario and save.

6 Repeat the same steps for the second child scenario.

When you connect both of the scenarios with the parent scenario, your list of scenarios should look like this:



Note the second to last column in the scenario list. The column shows whether the scenario is a parent, bridge, or child scenario in a scenario chain. Users in organizations with the Teams and Enterprise pricing plans can see the column.

See **Make pricing** for more information.

Set the scheduling for scenarios in the scenario chain

In the previous section, we've set the scenario scheduling to on demand to have the scenarios enabled for testing without running them in intervals.

The scenarios in the scenario chain have independent scheduling. A child scenario can trigger with its own scheduling, in addition to being triggered by a parent or a bridge scenario.

For example, if the child scenario in a scenario chain runs every two hours, but the parent (or bridge) scenario triggers it every hour, then the child scenario will run four times in three hours:

- 1st run: The child scenario is triggered by the parent scenario an hour after the start.
- 2nd run: The child scenario triggers by it's own scheduling two hours after the start.
- 3rd run: The child scenario is triggered by the scenario two hours after the start.
- 4th run: The child scenario is triggered by the parent sscenario three hours after the start.

When setting scenario scheduling for scenarios in the scenario chain, keep in mind that the scheduling of the linked scenarios combines.

If you want to run scenarios in a chain in a sequence, use on demand scheduling.

In the example, set scheduling for the parent scenario, but keep the child scenarios running on demand.

The child scenarios use the scenario inputs from the parent scenario with no default values. If the child scenarios would run with a schedule on their own, they wouldn't contain any values for the scenario inputs.

> You can prevent running a scenario with empty scenario inputs by setting the scenario input field as **Required** and the scheduling to **On Demand**.

Migrate from using the Run a scenario module

The **Run a scenario** module from the Make app enables you to trigger a scenario run from another scenario. The **Run a scenario** module from the **Scenarios** app works similarly.

The major advantage of scenario chaining over triggering a scenario with the Make app is the management utility and auditability of scenario chaining. You can view connected scenarios in the relation tree tab and you can see chained scenarios in the scenario list and history.

For a full comparison of the individual approaches to trigger a scenario from another scenario, check the **comparison section**.

To start using scenario chaining instead of the **Run a scenario** module from the **Make** app:

1. Go to the scenario that uses the **Make** > **Run a scenario** module.
2. Swap the **Make** > **Run a scenario** module for the **Scenarios** > **Run a scenario** module.
3. In the **Scenario** field, select the scenario you want to trigger.
4. Optional: If you transfer data between scenarios, map scenario inputs to the **Run a scenario** module.
5. Review the scheduling of the child scenario. The child scenario can have its own scheduling in addition to being triggered from the scenario chain.

   You can access the child scenario from the **Preview** link in the **Run a scenario** module settings.

Check the relation tree tab in the details of any of the scenarios. The relation tree tab is now showing two scenarios.

The parent scenario is the scenario that contains the **Run a scenario** module, and the child scenario is the scenario triggered by the module.

Migrate from using the HTTP and custom webhook apps

With the **HTTP** > **Make a request** module, you can send data to a **Webhooks** > **Custom webhook** module. The **Scenarios** > **Run a scenario** module works differently. **Run a scenario** module sends data to a scenario through scenario inputs, instead of using the HTTP app.

The major advantage of scenario chaining is the management utility and auditability of scenario chaining. You can view connected scenarios in the relation tree tab and you can see chained scenarios in the scenario list and history.

You can also use the scenario inputs to validate the data you send to the child scenario instead of checking the data after the webhook receives them.

For a full comparison of the individual approaches to trigger a scenario from another scenario, check the **comparison section**.

To start using scenario chaining instead of the **HTTP** and **Custom webhook** modules:

1. Go to the scenario with the **HTTP** > **Make a request** module.
2. Swap the **HTTP** > **Make a request** module for the **Scenarios** > **Run a scenario** module.
3. In the **Scenario** field, select the scenario you want to trigger.
4. Optional: Set up scenario inputs to transfer data from the parent scenario to the child scenario. Make will evaluate the structure of the inputs before running the scenario, which increases the reliability of the scenario chain.
5. Remove the **Custom webhook** module from the child scenario.
6. Review the scheduling of the child scenario. The child scenario can run on demand when triggered from the parent scenario, or can have its own scheduling in addition to being triggered from the scenario chain.

Check the relation tree tab in the details of any of the scenarios. The relation tree tab is now showing two scenarios .

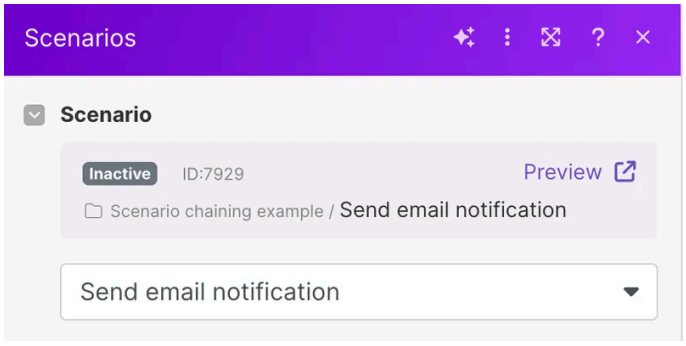
The parent scenario is the scenario that contains the **Run a scenario** module, and the child scenario is the scenario triggered by the module.

Manage scenario chains

Scenario chaining simplifies the management of scenario sequences, where one scenario triggers another. This approach eliminates the need for naming conventions or organizing sequences in folders. Instead, you can easily monitor and manage the scenarios within a scenario chain from multiple places in Make:

> The scenario relations, like:
>
> - the **Relation tree** tab
> - scenario relations in the list of scenarios
> - scenario relations in the scenario history
>
> show only to users in organizations with the Teams and Enterprise pricing plans. Users in other organizations have to track scenario relations differently.
>
> See **Make pricing** for more information.

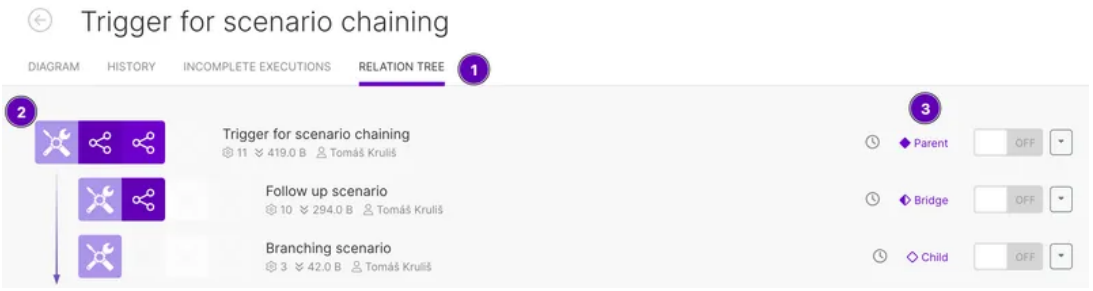
Run a scenario module: When you select a scenario in the **Scenarios** > **Run a scenario** module, you can see at the top of the module settings information about the scenario triggered by the module:



You can see:

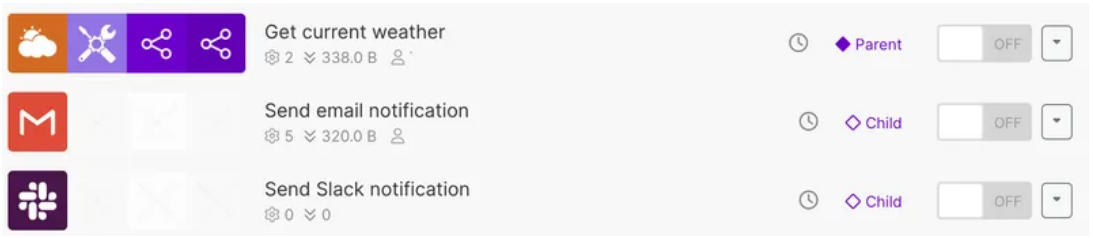
- whether the scenario is enabled (Active) or disabled (Inactive)
- scenario ID
- the scenario folder name
- the Preview link. Clicking the preview link opens a new browser tab with the linked scenario open in the Scenario Builder

Relation tree tab: Scenarios that are part of a scenario chain have the **Relation tree** tab:

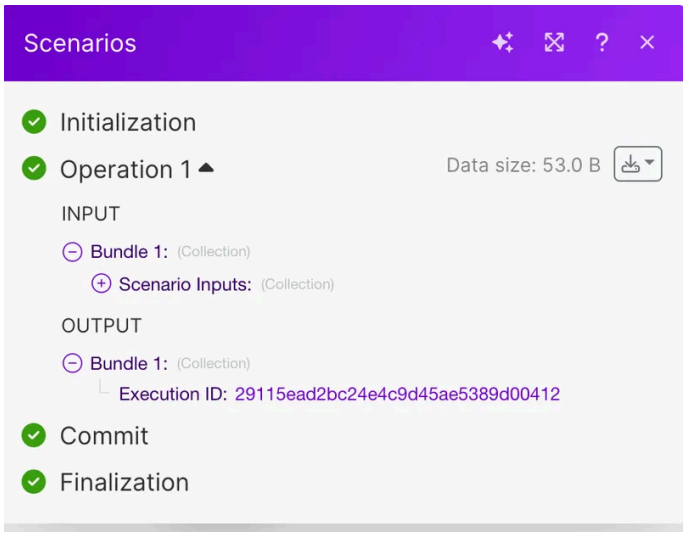


- The **Relation tree** tab shows scenarios directly connected to the currently open scenario in the scenario chain.
- The open scenario is outdented in the scenario relation tree.
- A column in the relation tree shows the relation of the scenario in the scenario chain.

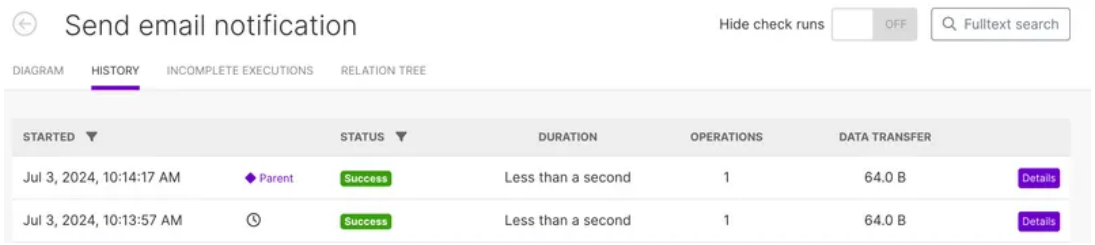
The scenario list also shows the scenario chain relations:



Bundles output by the **Run a scenario** module contain a link to the triggered scenario run.



Scenario history contains a link to the scenario that triggered the scenario run.



> There is no link if the scenario in the scenario chain runs because of its own scheduling instead of being triggered by a parent or a bridge scenario.

Add scenarios to a scenario chain

To add a scenario to an existing scenario chain:

1. Create the scenario you want to add to the chain.
2. Go to the scenario from which you want to trigger the new scenario.
3. In the parent scenario, add the **Scenarios** > **Run a scenario** module.
4. In the **Scenario** field, select the scenario you want to trigger (the scenario created in step 1).

You connected the scenarios to a scenario chain. Check the **Relation tree** tab of the added scenario to see the scenario connected to the parent scenario.

> Make sure to review the scenario scheduling and enable the added scenario.

Remove scenarios from a scenario chain

To remove a scenario from a scenario chain:

1. Go to the parent scenario which triggers the scenario you want to remove.
2. In the parent scenario, remove the **Scenarios** > **Run a scenario** module.

You removed the scenario from a scenario chain. The removed scenario doesn't have a **Relation tree** tab anymore and Make doesn't highlight the scenario as a part of a scenario chain.

Scenario chaining comparison with other options to connect scenarios

If you are a power user of Make, you might know of other options on how to connect scenarios into a sequence. The following table compares the other approaches with scenario chaining:

	Scenario chaining	Make app: Run a scenario	Trigger a webhook with the HTTP app
Scenario triggering	**Scenarios** > **Run a scenario**	**Make** > **Run a scenario**	**HTTP** > **Make a request** the follow-up scenario has to start with a webhook module
Setting next scenario	Select from a list of scenarios, categorized in folders. The scenario has to be in the same team.	Search scenarios available through the Make app connection. You can run scenarios in any organization or team to which you have access.	You have to know the webhook URL. You can trigger any webhook if you know its URL.
Sending data	Use the **scenario inputs** to accept data from the previous scenario.	Use the **scenario inputs** to accept data from the previous scenario.	Send the data in the HTTP request `body` or in the `query` parameters.
Scenario sequence management	• Each scenario in the chain has the relation tree overview. • In the scenario list, you can see which scenario is a parent, bridge, or child in a chain. • In the scenario execution history, you can distinguish between scenario runs triggered by	There are no dedicated management options. You can put sequence scenarios in a dedicated folder, use **custom scenario properties**, or create your own naming convention.	There are no dedicated management options. You can put sequence scenarios in a dedicated folder, use **custom scenario properties**, or create your own naming convention.

	Scenario chaining	Make app: Run a scenario	Trigger a webhook with the HTTP app
	another scenario in the chain. • The bundles generated by the **Run a scenario** module contain a link to the execution log of the nextscenario in the chain.		
Scenarios running in parallel	You can trigger the next scenario in a chain before it finishes it's current run. You can have multiple scenariosrunning in parallel.	You can have multiple scenarios running in parallel.	You can run webhook-triggered scenarios in parallel.

Scenario chaining limitations

Scenario chaining has certain feature limitations to help you avoid creating complex or unmanageable chains that could result in infinite loops. These limitations are as follows:

- You cannot use the **Run a scenario** module to trigger the current scenario itself.
- You cannot form a circular scenario chain. If a scenario is already part of a scenario chain, you cannot add it again to the same chain.
- You cannot connect scenarios from different teams. You can only create a scenario chain from scenarios within the same team.

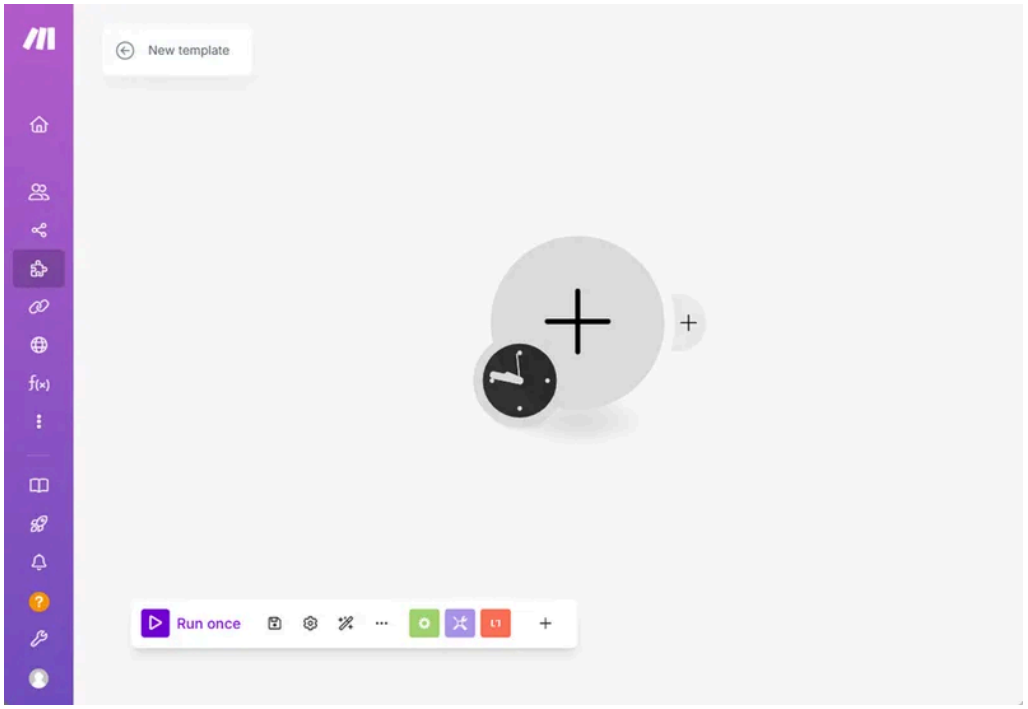
3.1.5. Create scenario templates

Templates are predefined scenarios that you can customize to create new scenarios. Templates can be saved for future use and shared.

Create a scenario template

> This feature is available to anyone with a **Teams plan**.

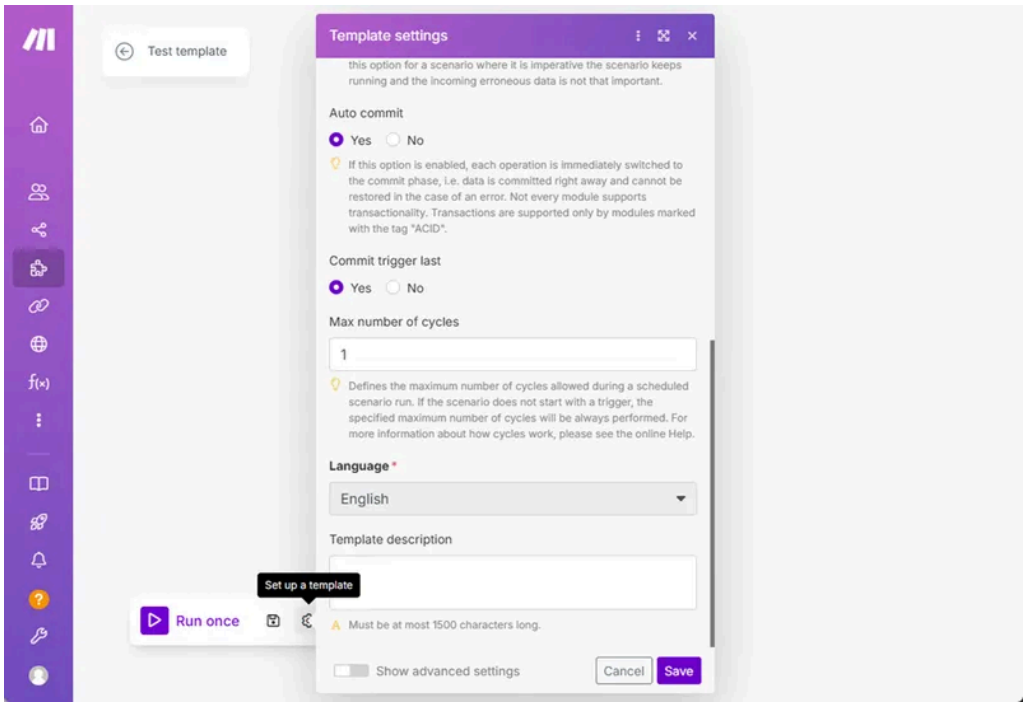
1. In the left sidebar, click **Templates**.
2. In the upper-right corner, click **+ New template**.
3. Your template will have the default name New template. Click this title to rename the template.



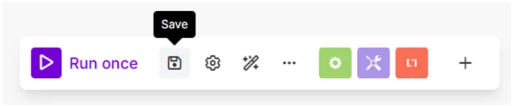
4. In the scenarioeditor, **create your scenario**.

5 Optional: Click the gear icon at the bottom left and enter a **Template description**. The description can be a maximum of 240 characters.

Click **Save**.



6 Click the Save icon to save the template.



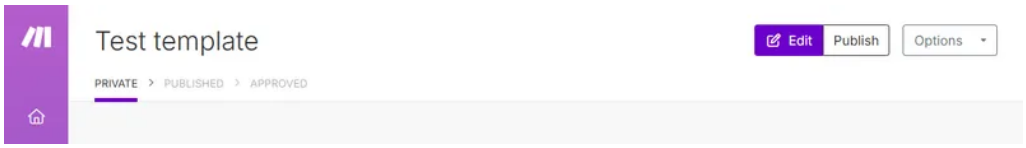
Publish a scenario template

To share your template with others, you must first publish it.

> You can't edit a published template. If you need to edit a published template, first unpublish your template, complete your edits, and then publish again.

1 Select the private template you wish to share to open it in the template editor.

2 Click **Publish**.



> One you publish your template, you can share the link with others. In the upper-right corner, click **Share public link** to copy the link to your clipboard.

Request approval of your template

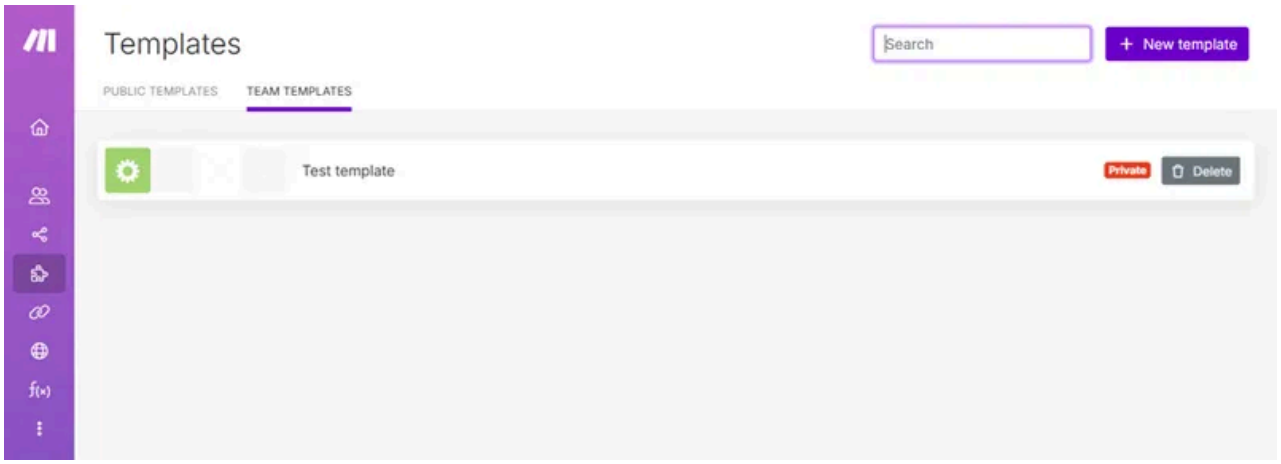
You can send your published templates for approval to make them available in the public templates library.

Click **Request approval** in the upper-right corner of the template editor. The button will now read **Approval pending**.

A Make administrator will review your template for public use. After approval, we will add your template to the public template library.

Delete a scenario template

1. In the left sidebar, click **Templates**.
2. Click the **Team templates** tab to view your Private templates.
3. Next to the template you with to delete, click **Delete**.



4 Click **Really?** to confirm.

3.1.6. Custom scenario properties

> This feature is available to Enterprise customers.

To help organize and sort scenarios, you can use custom scenario properties to add customizable metadata. The properties you create appear on the scenario detail page and also in sortable columns in a table view of your scenario. You can add as many custom properties as you like and edit which properties appear in your table view.

All organization members can view and use custom scenario properties. This includes using custom properties to filter the table view of scenarios.

For all properties, you can define the following:

- **Name** - a descriptive name used as a unique identifier that only appears when you manage your custom properties.
- **Label** - the name visible on the table view and detail page.
- **Hint text** - a short message you add to help your organization members use your custom property.
- **Field type** - the property type. For example, dropdown or boolean.
- **Required** - select **Yes** to make a property mandatory to include when applying custom properties to a scenario.

Custom scenario properties support the following field types:

- **Short text** - up to 200 characters; useful for email addresses, and URLs.
- **Long text** - up to 1,000 characters; supports multiple lines.
- **Number** - supports integers.
- **Boolean** - a yes/no value that appears as radio buttons.
- **Date** - date and time according to **ISO 8601.**
- **Dropdown** - a list of items where users can select only one.
- **Multichoice** - a list of items where users can select multiple values.

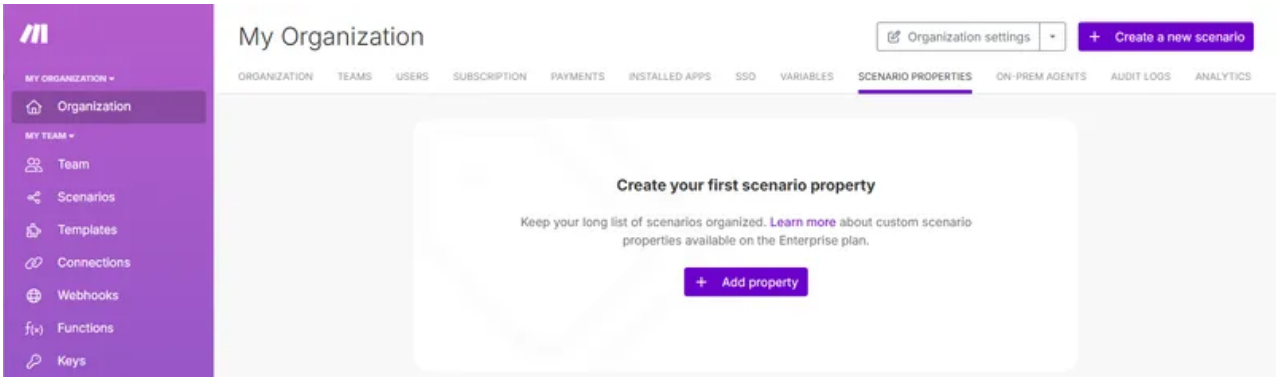
Create custom properties

Your organization dashboard has a **Scenario properties** tab where you can create and manage your custom properties.

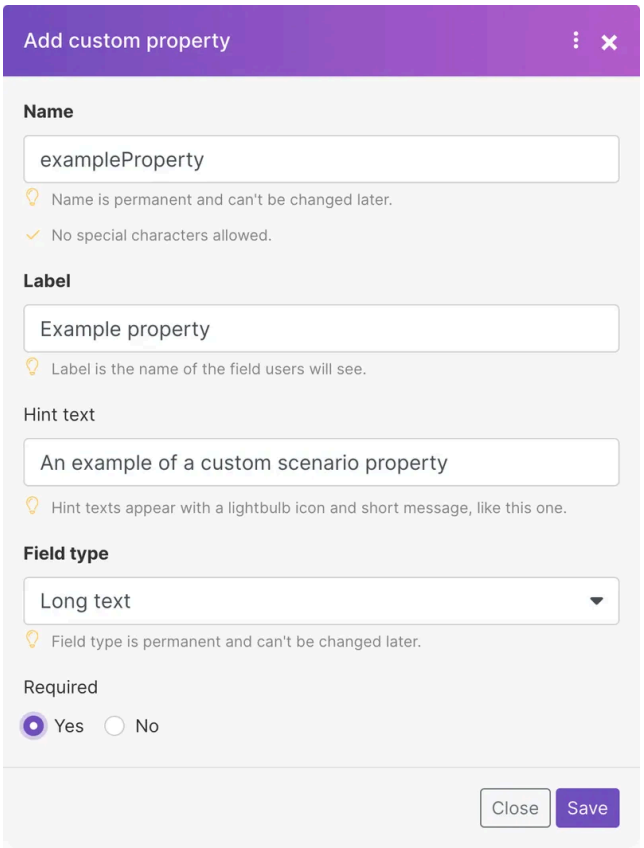
1. In the left sidebar, click **Organization**.

2. Click the **Scenario properties** tab.

3. Click **Add property**.



4. In the **Add custom property** dialog, enter and select your information for each field. Click **Save** to create your custom property.



Your custom property appears on the list on the **Scenario properties** tab.

> Only organization owners and admins can create and manage custom properties.

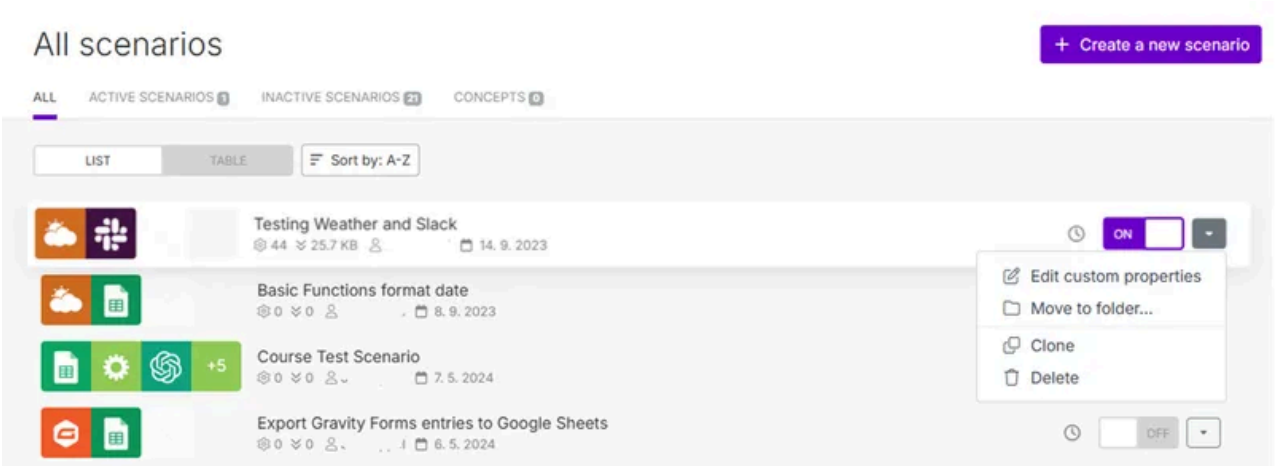
Use custom Make properties

All organization members can:

- Apply and edit custom properties to Make.
- Use custom properties to filter and sort the Make list.

Apply and edit custom scenario properties

1. In the left sidebar, click **Scenarios**.
2. Click the down arror icon next to the scenario whose custom properties you want to edit.
3. Select **Edit custom properties**.



4. In the dialog box, edit your scenario properties

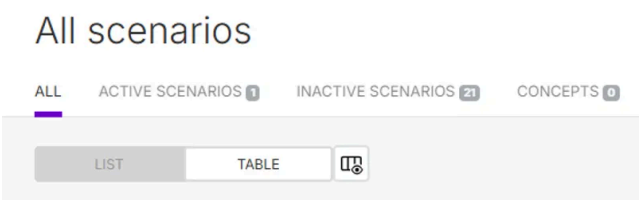
5 Click **Save**.

> You can also apply and edit custom properties from the scenario detail page.

Filter the scenario list

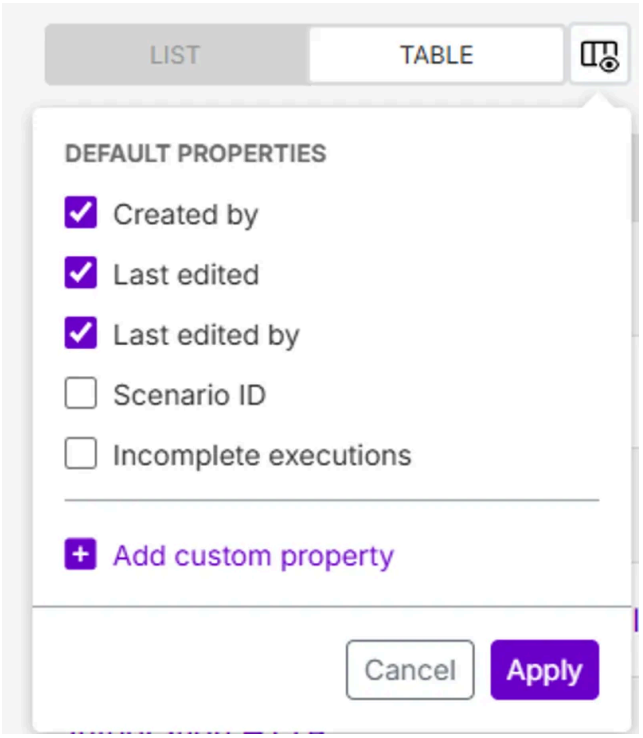
1 In the left sidebar, click **Scenarios**.

2 Click the toggle to enter **Table** view.



3 Click the icon to the right of the **Table** view toggle.

4 In the dialog box, select the properties you want to see in the table.



5 Click **Apply**.

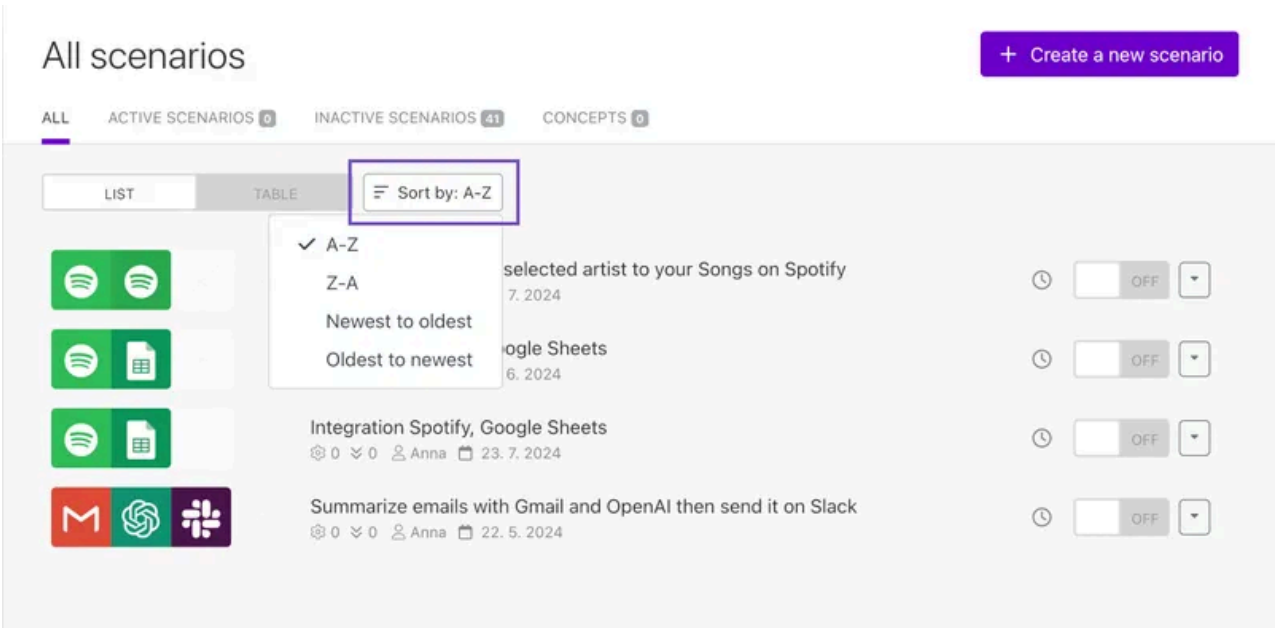
Sort the scenario list

By default, scenarios are sorted by their names in alphabetical order, with active scenarios at the top of the list. However, it is possible to change the sorting in both the List and Table views.

From the List view:

1 In the left sidebar, click **Scenarios**.

2 Click the **Sort by** button.



3 Select a sorting option. You can sort scenarios by their name or creation date.

From the Table view:

1 In the left sidebar, click **Scenarios**.

2 Click the toggle to enter **Table** view.

3. Click the title header of a column to sort the list by that property. Click again to see a descending order list.

Manage custom properties

You can edit or delete your custom scenario properties from the **Scenario properties** tab of your organization dashboard.

Edit custom properties

1. In the left sidebar, click **Organization**.
2. Click the **Scenario properties** tab.
3. Find the custom property you want to edit and click **Edit.**
4. In the dialog box, change the information as needed.
5. Click **Save**.

A confirmation message appears at the bottom of the screen and your changes appear in the table.

Delete a custom property

1. In the left sidebar, click **Organization**.
2. Click the **Scenario properties** tab.
3. Find the custom property you want to delete and click the down arrow icon.

4. Click **Delete**.

The property disappears from the table.

3.1.7. Variables

Variables are containers for storing data. You can think of a variable as a “box” for data, with a uniquely-named sticker on it.

Imagine you are moving to a new home and you need to pack up all the things in your house. You might want a box for clothes, one for books, and one for the dishes. When you pack the boxes, it's helpful to add a label to each box. That way, you know what is inside.

Variables store data as boxes store items.

However, unlike a box, you can store only one thing in a variable. Variables have a fixed name and a value that can change. You can reference the name in your scenario. The name is then replaced by the current value of your variable when the scenario executes.

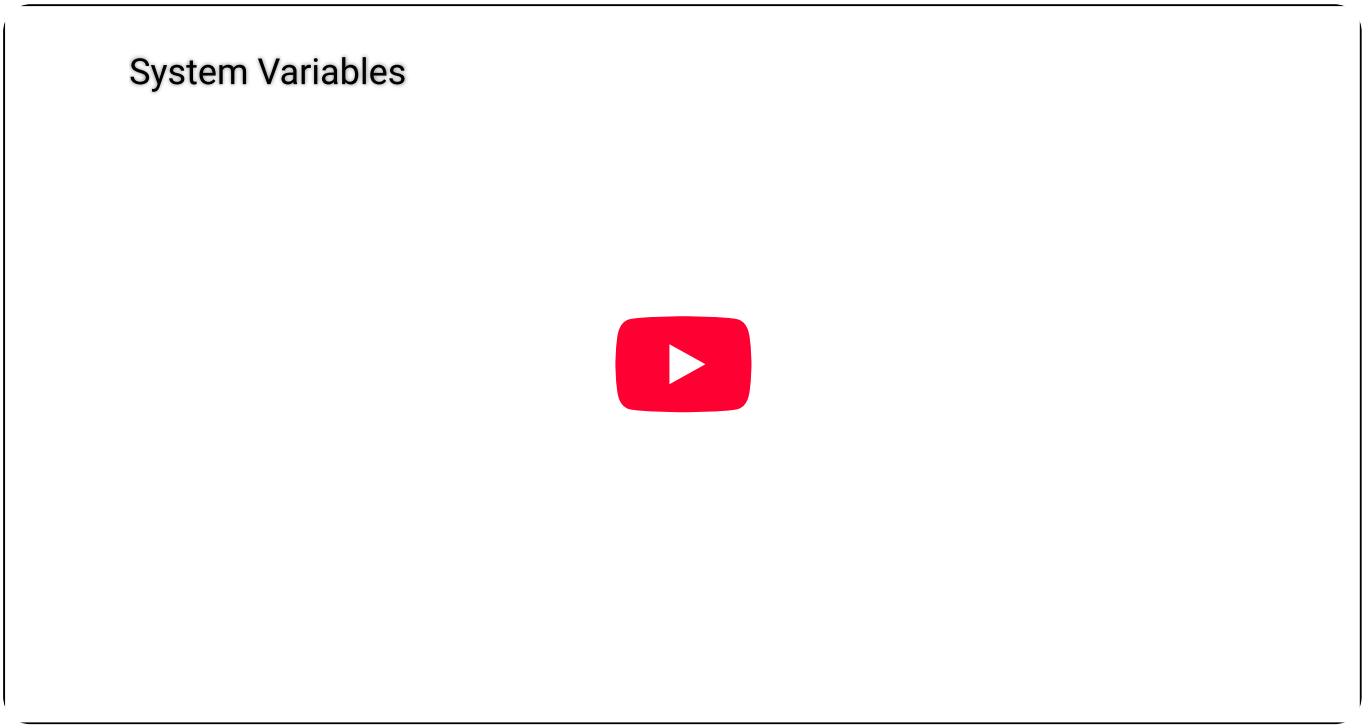
When using Make, you might need to save certain pieces of information and assign a label to the information so that you can retrieve it later. A username, the day of the week, or available inventory left in your e-shop - these are just some examples of data that can be stored in variables as reusable elements that can be mapped to any module in the module's input fields.

Make offers four types of variables:

- **System variables** are provided by Make. You can’t modify or delete system variables. They can be used in modules with input fields, filters within the scenario editor, and in templates.
- **Scenario variables** are created within a scenario for use in scenario executions. You can use the **Set variable** and **Set multiple variables** tools to define these variables.
- **Custom variables** are similar to scenario variables, but they are created at the Organization or Team level and can be accessed and modified within a scenario execution. Custom variables are only available with Pro, Teams, and Enterprise **pricing plans**.
- **Incremental variables** are created for each scenario. You can increment them any time in a scenario execution. The value of the incremental variable can be kept within multiple scenario executions.

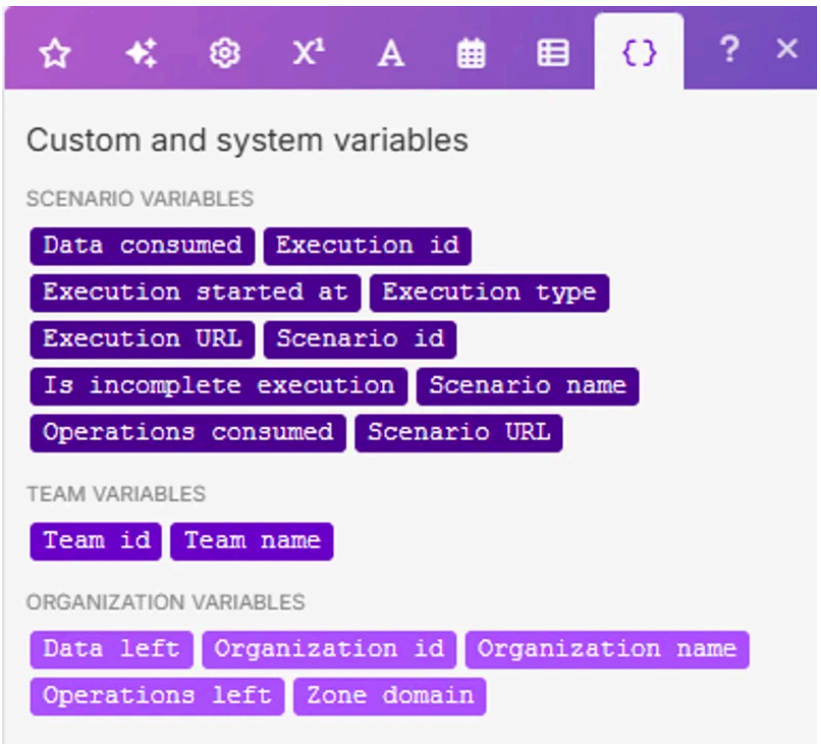
Variable type	Scope/Visibility	Created	Modified	Deleted
System variables	In all scenarios Visability varies (global, team, current scenario)	By Make	By Make	Never deleted
Scenario variables	Only in the current scenario execution	Only in the scenario	Only in the scenario	Can't be deleted, but they no longer exist when the scenario stops
Custom variables	Organization-level variables are visible for all scenarios in the organization Team-level variables are only visible in scenarios built within that team	By the admin of the organization or team	By the admin of the organization or team Within a scenario, by using the Make API or the **Update a Custom Variable** module	By the admin of the organization or team
Incrementable variable	In the current scenario	By Make	Incremented in the scenario, using the **Tool>Increment Function** module	Value resets for each scenario execution, or never resets, based on the parameters chosen in the **Tool>Increment Function** module

3.1.7.1. System variables



System variables are provided by Make. You can't modify or delete system variables. They can be used in modules with input fields, filters within the scenario editor, and in templates.

You can access the scenario variables under the **Custom and system variables** tab when mapping a value in a module.



System variables include information about:

- the scenario itself (Scenario ID, Scenario name, Scenario URL)
- the specific scenario execution (Execution ID, Operations consumed, Data consumed, Execution start date/time)
- the team the scenario is in (Team ID, Team name)
- the organization the scenario is in (Organization ID, Organization name, Operations left, Data left, Zone domain)

System variables are useful for error handling and scenario activity tracking.

For example, if there is an error in your scenario, you can use the scenario ID or organization name variables to pinpoint the exact scenario execution and organization in which the error occurred.

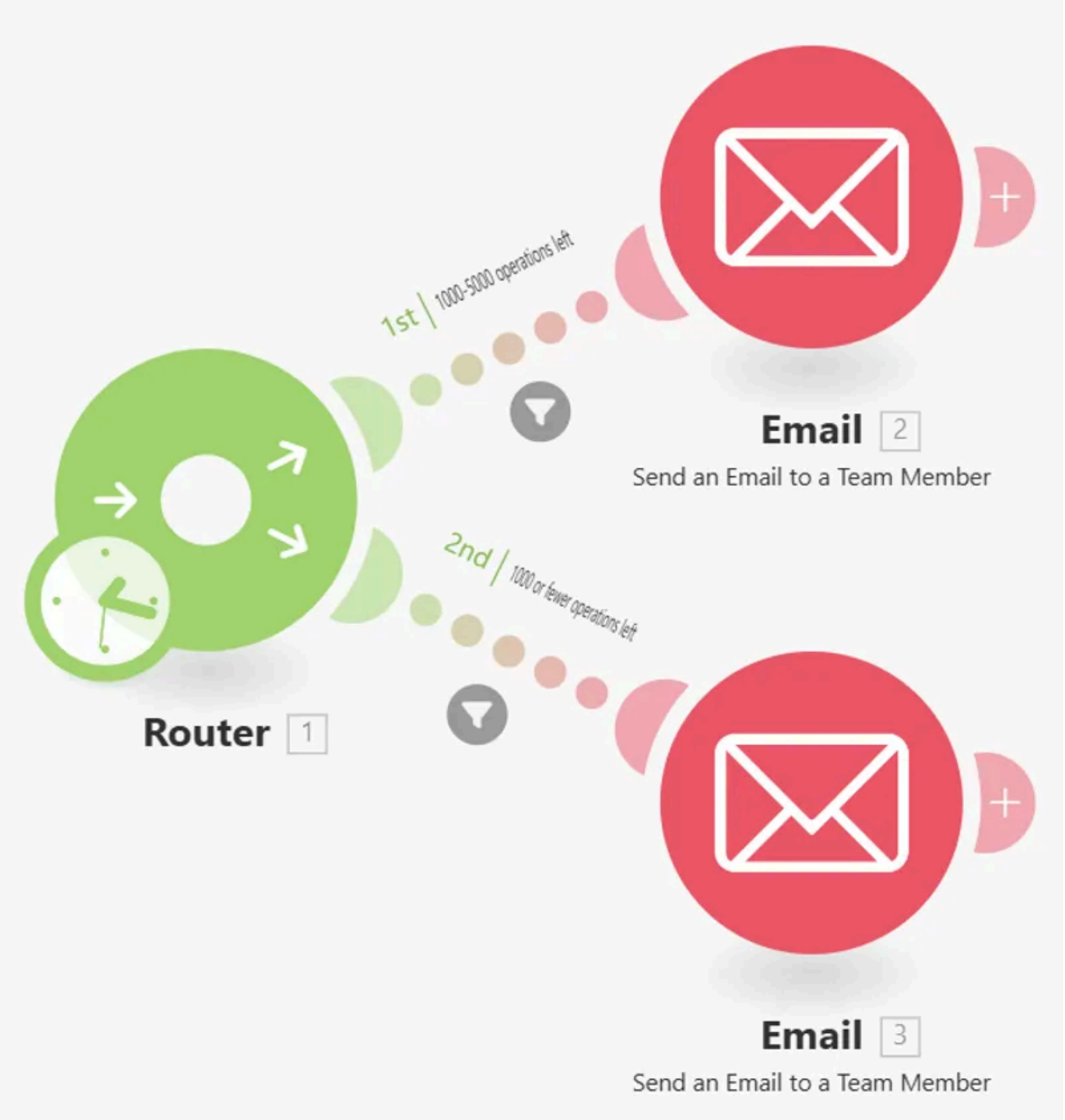
Here are some other situations for which you might use system variables:

- Create a slack message, email, or support ticket that contains a link to the scenario execution that created the message.
- Set email notifications to notify you once the operations left value reaches a certain number.

You can also reuse system variables and map them to any module in the module's input fields.

Example: Get notified before you reach your operations limit

In this example, we imagine that at some point in your scenario you want to notify a team member of the number of operations left in your organization. We use a router to send an email when there are 1000-5000 operations left and when there are under 1000 operations left.



1. In your scenario, add a router.

2 In the first route, click the wrench icon to set up a filter.

Label: 1000-5000 operations left

Condition: Map the condition to the `Operations left` system variable.

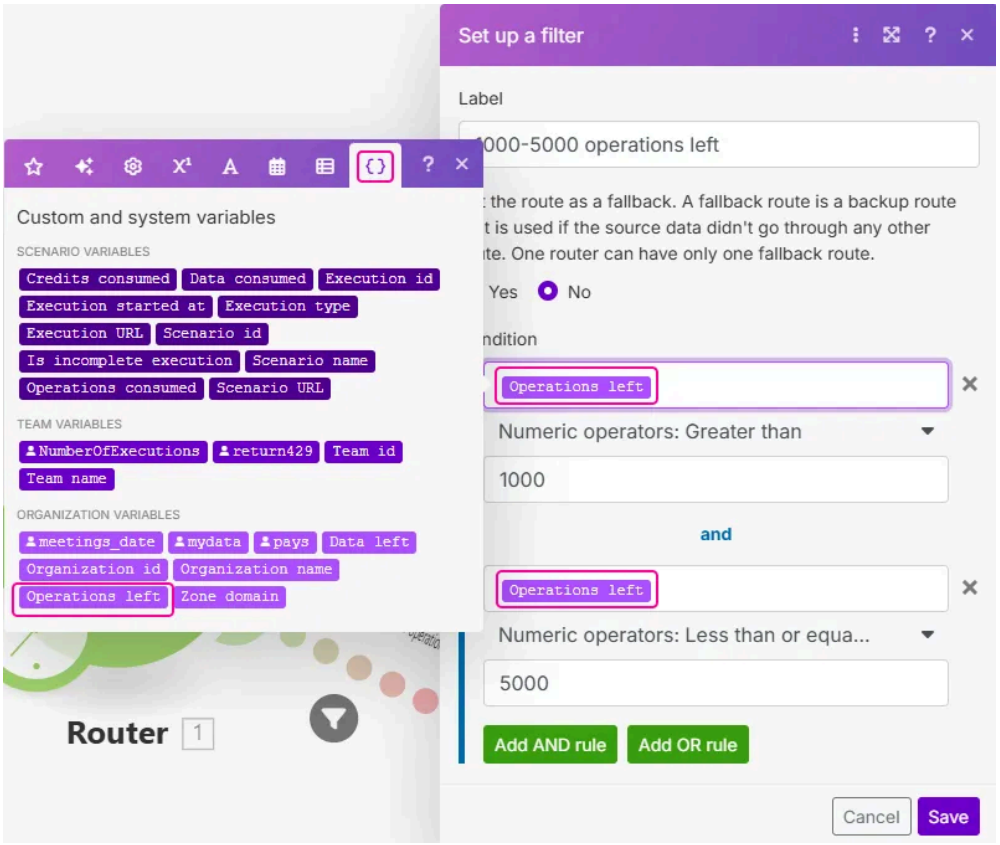
Change the operator to **Numeric operators: Greater than** and enter `1000` in the field.

Click **Add AND rule**.

Condition: Map the second condition to the **Operations left** system variable.

Change the operator to **Numeric operators: Less than or equal to** and enter `5000` in the field.

Click **Save**.



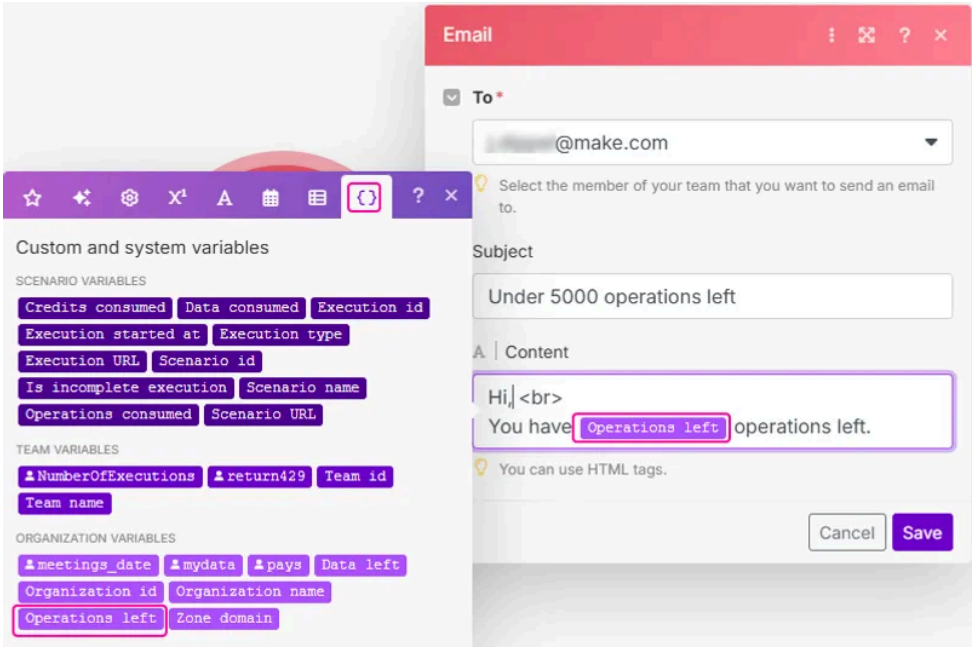
3 After this filter in the first route, add an **Email > Send an Email to a Team Member** module and **Create a connection**.

In the **To** field, select the member of your team you want to send an email to.

In the **Subject** field, enter a relevant subject.

In the **Content** field, enter the text of your email, mapping the Operations left variable. You can use HTML tags.

Click **Save**.



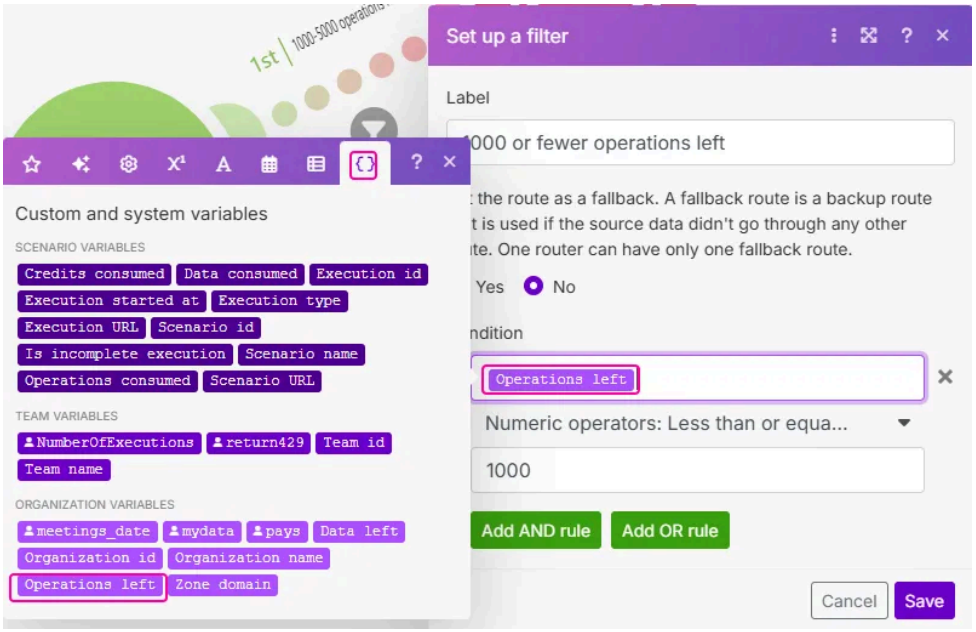
4 In the second route, click the wrench icon to set up a filter.

Label: 1000 or fewer operations left

Condition: map the condition to the `Operations left` system variable.

Change the operator to **Numeric operators: Less than or equal to** and enter `1000` in the field.

Click **Save**.



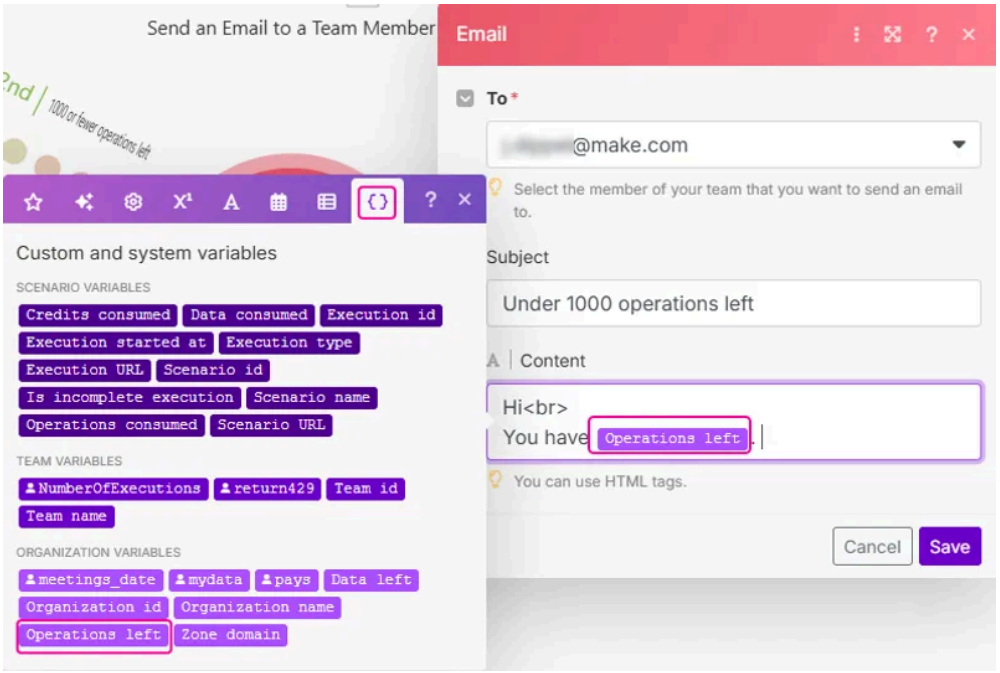
5 After this filter in the second route, add an **Email > Send an Email to a Team Member** module.

In the **To** field, select the member of your team you want to send an email to.

In the **Subject** field, enter a relevant subject.

In the **Content** field, enter the text of your email, mapping the `Operations left` variable. You can use HTML tags.

Click **Save**.



As the number of operations left drops to the filtered amounts, your team member receives email notifications.

3.1.7.2. Scenario variables

Scenario variables are useful when you need to reuse the same information multiple times. They make scenarios easier to manage and maintain.

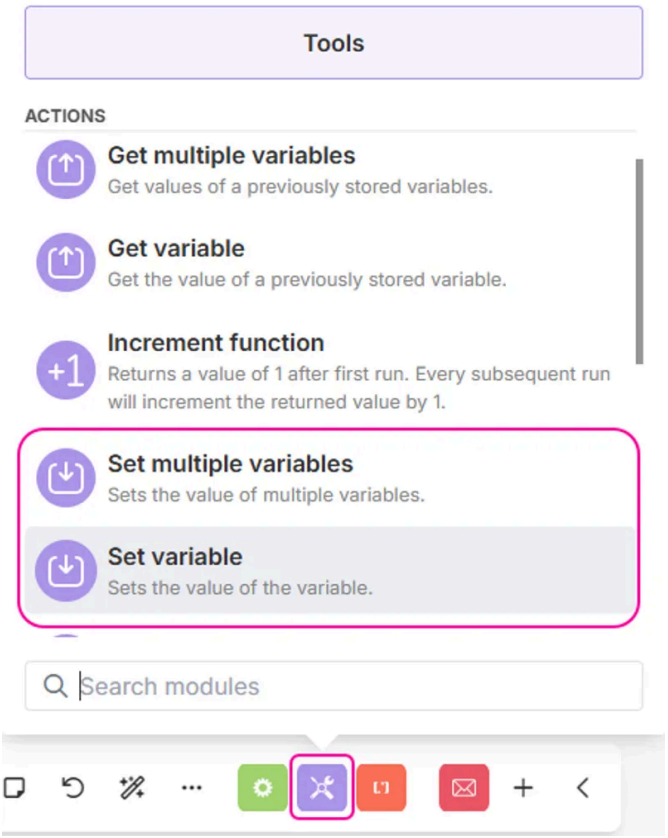
To use scenario variables, use the **Set variable** or **Set multiple variable** tools in your scenario.

If you need to access the defined variable(s) in a different route than where they were set, use the **Get Variable** or **Get Multiple Variable** tools.

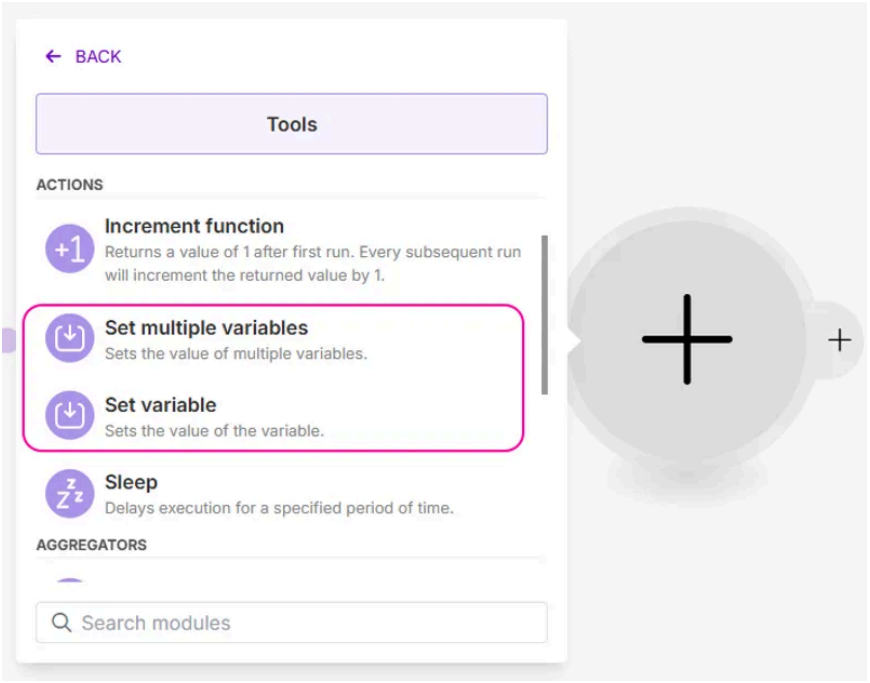
Create a scenario variable

You can create scenario variables in your scenario.

1. Click the **Tool** icon on the Scenario Builder toolbar to select the **Set variable** tool.



You can also click the giant plus in your canvas to select **Tools < Set variable**.



> To set multiple variables at once, select the **Set multiple variables** tool.

2 Fill in the following information:

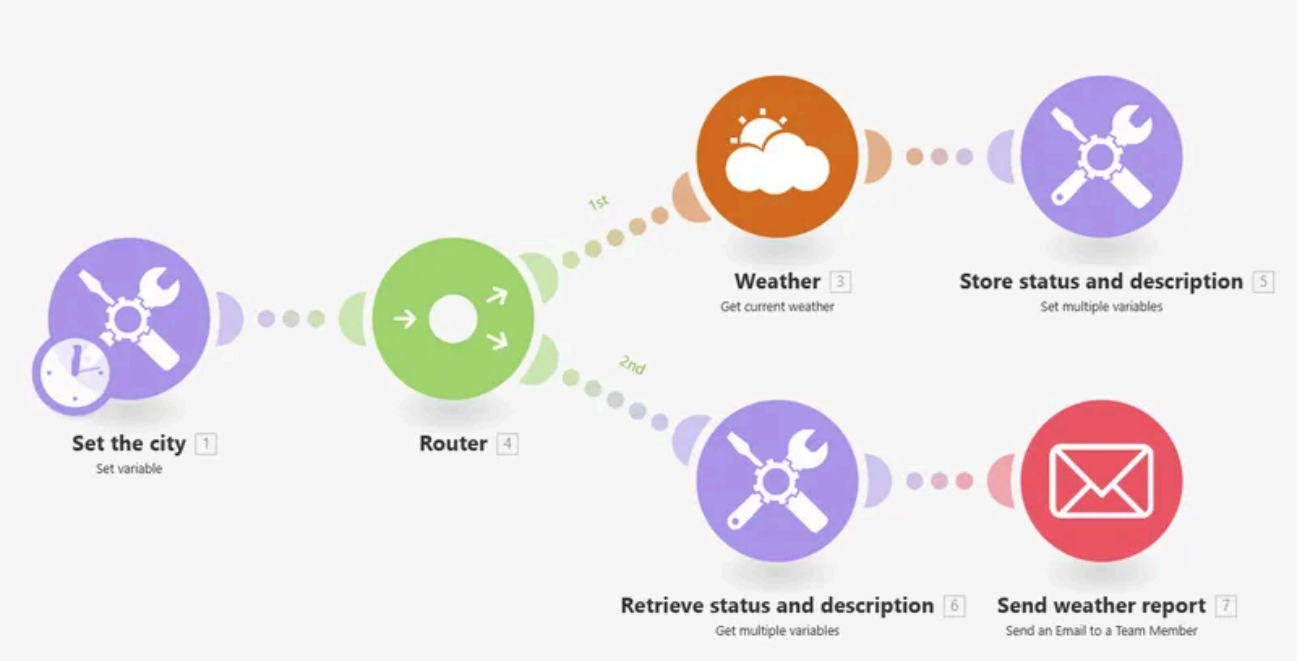
Field	Description
Variable name	• This field is mandatory. • The name is the identifier of the variable.
Variable value	• Enter the value for your variable. • You can use letters, digits, spaces, and special characters.
Variable lifetime	Select the lifetime of your variable: • One cycle • One execution

If you are using the **Set multiple variables** tool, repeat this for each variable.

3 Click **Save** to save your variable.

Example: Send an email with weather conditions for a selected city

In this example, we use the **Set variable** module to select a city, retrieve weather information for the city, and use the **Set multiple variables** and **Get multiple variables** modules to send the information in an email to a team member.



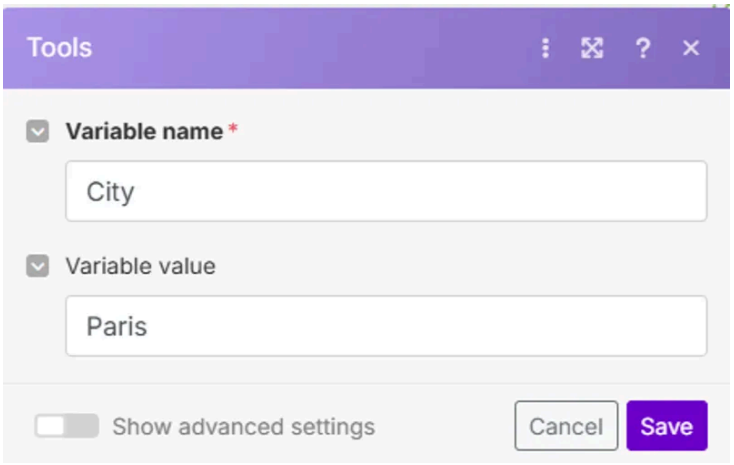
1. In your scenario, click the **Tool** icon on the Scenario Builder toolbar to select the **Set variable** tool.

   You can also click the giant plus in your canvas to select **Tools < Set variable**.

2. In the **Variable name** field, enter `City`.

   For this example, we have set the **Variable value** to `Paris`.

   Click **Save**.



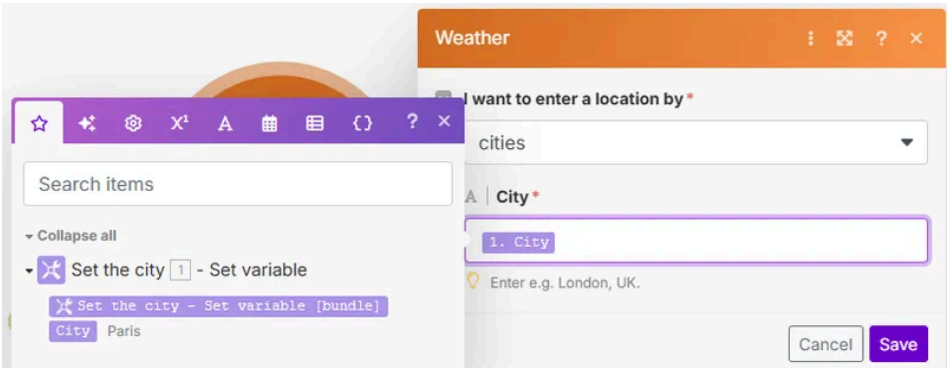
3 Add a **Router**.

4 In the first route at the top, add a **Weather** module.

In the **I want to enter a location by** field, select `cities`.

In the **City** field, map the value to the `City variable` that you set in step 1.

Click **Save**.

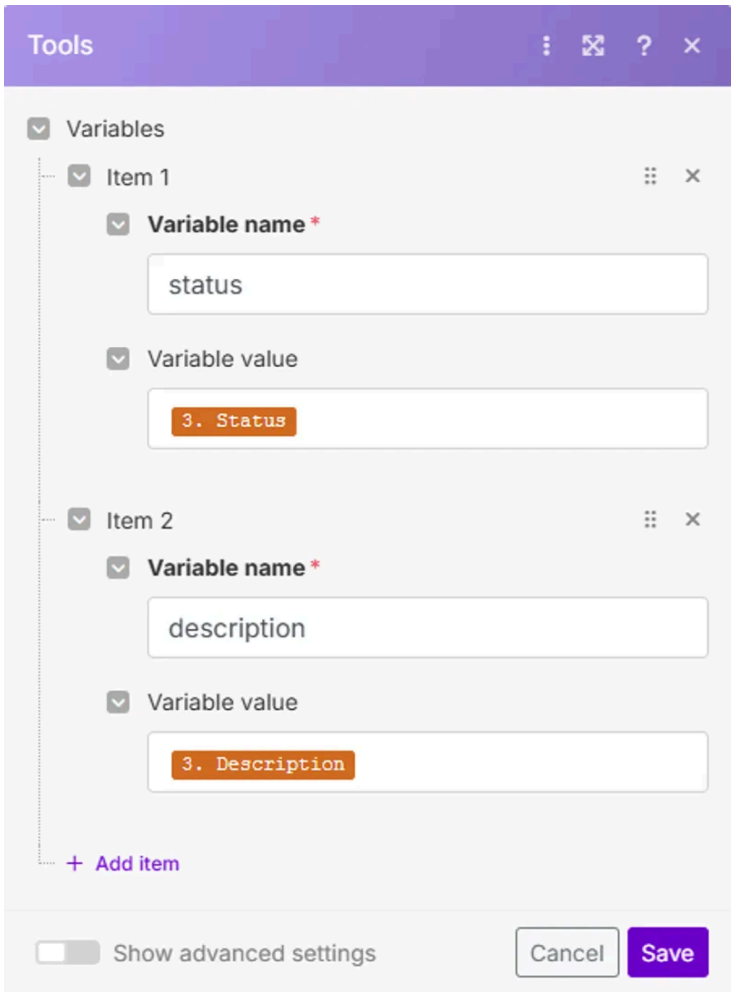


5 After the **Weather** module in the first route, add a **Set multiple variables** module.

Add two variables:

- `status` - mapped to the `Status` value from the Weather module.
- `description` - mapped ot the `Description` value from the Weather module.

Click **Save.**

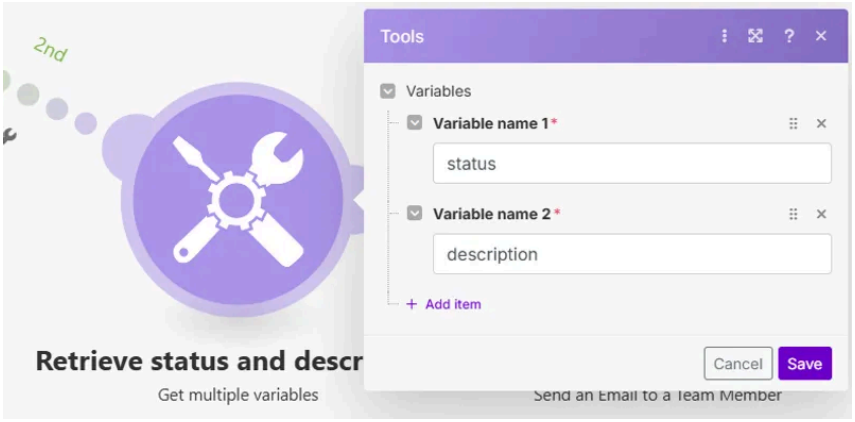


6 In the second route at the bottom, add a **Get multiple variables** module to retrieve the information from the previously set variables.

For **Variable name 1**, enter `status`.

For **Variable name 2**, enter `description`.

Click **Save**.



> Note that this step is not necessary if you are not using a router.
>
> In this example, we've chosen to split the scenario into two routes to demonstrate both the **Set mutliple variables** and **Get multiple variables** modules.

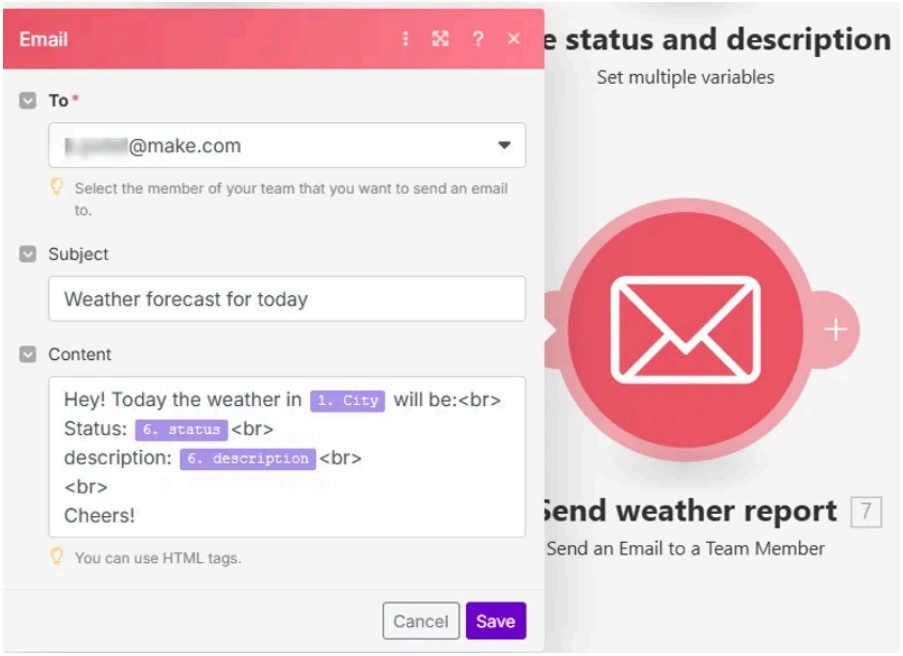
7 After the **Get multiple variables** module in the second route, add an **Email > Send an Email to a Team Member** module and **Create a connection**.

In the **To** field, select the member of your team you want to send the email to.

In the **Subject** field, enter a relevant subject.

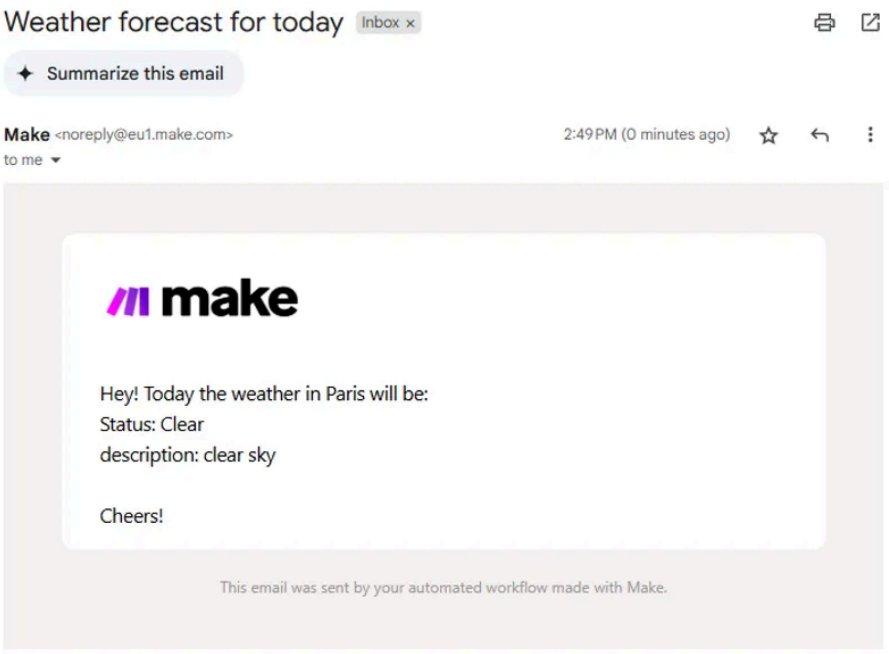
In the **Content** field, enter the text of your email, mapping the `City`, `status`, and `description` variables. You can use HTML tags.

Click **Save**.



8 Click **Run once** to test your scenario.

The person you selected in the Email module receives an email like this:



3.1.7.3. Custom variables



Custom variables are similar to **scenario variables**, but they are defined at the organization or team level.

> Custom variables are available only on Pro, Teams, and Enterprise pricing plans. See **Make pricing** for more information.

For each variable, you need to identify the:

- **Name** (permanent variable name)
- **Data type** (text, number, Boolean, or date)
- **Value** (the actual value of the variable)

Here are some situations for which you might use custom variables:

- Keep track of a value between multiple scenario executions
- Share a value between different scenarios
- Set a global variable that various scenarios leverage. For example `TestMode = true/false` so the scenario routes differently according to the value.

- Change the behavior of a scenario without modifying the scenario blueprint. For example, `myLimit=20` set as a custom variable in the Limit field of a module.

> Variables are not meant to store secrets. Don't use them for anything sensitive!
>
> Variable values are not encrypted as they are stored in plain text. Don't use variables to store passwords or any other sensitive data. Be aware that other team members and organization admins can access all custom variables.

Create a custom variable

You can create custom variables at the organization level and at the team level.

1. Decide if you want to create an organization variable or a team variable.

   - For organization variables, go to the organization dashboard and click **Variables > Add organization variable**.
   - For team variables, go to the team dashboard and click **Variables > Add team variable**.

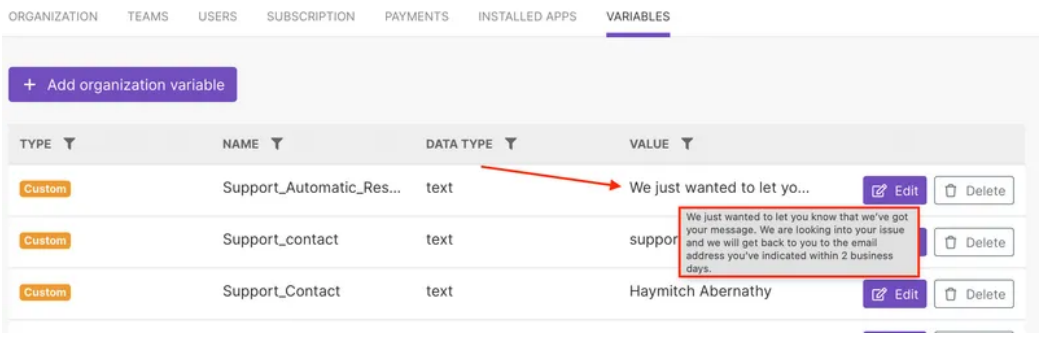
2. Fill in the following information:

Field	Description
Name	• This field is mandatory and you cannot change it after you save it. • The name is the identifier of the variable. • The name must contain only letters, digits, or the symbols $ and _. • The name cannot start with a digit. • The name must contain a maximum of 128 characters.
Data type	Choose the variable data type from the dropdown menu • Number • Text • Boolean • Date
Value	• Enter the value for your variable. • You can use letters, digits, spaces, and special characters. • Value cannot be empty.

3 Click **Save** to save your custom variable.

Your new variable will appear in the list of variables.

To preview the variable value, hover over the individual value field.

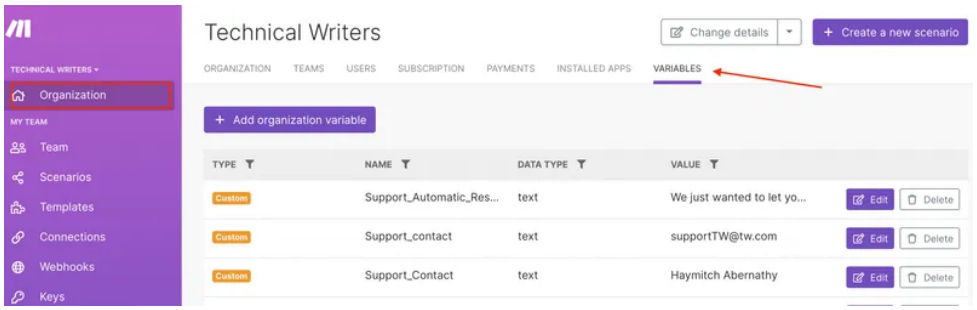


Edit a custom variable

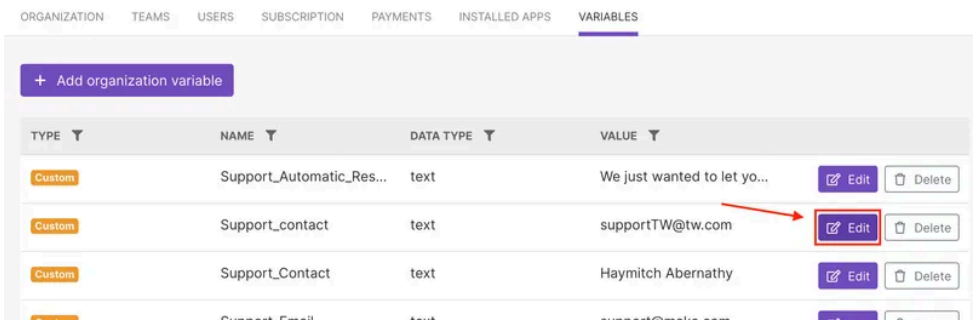
> If your other team members or organization admins use the same variable in their scenarios, the changes will affect them too.

1. Go to the variable you want to edit.

   - For organization variables, go to the organization dashboard and click **Variables**.
   - For team variables, go to the team dashboard and click **Variables**.



2. In the list of your custom variables, find the one you want to edit and click **Edit** next to it.



3. Edit the variable as needed. You can edit the variable data type and value. You can't edit the name.

4. Click **Save** to save your changes.

Your changes are saved and your updated variable will appear on the list of variables. Hover over the individual value field to preview the variable value.

The changes automatically update in the scenarios that already use the variable.

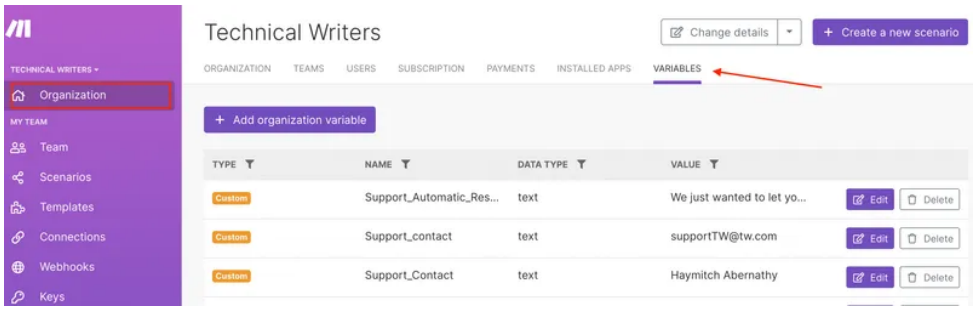
> The value of a custom variable can be changed within a scenarioas well. The value changes only **after** the scenario finishes running. The new value is available for the next run of the scenario for everyone in the organization or team.

Delete a custom variable

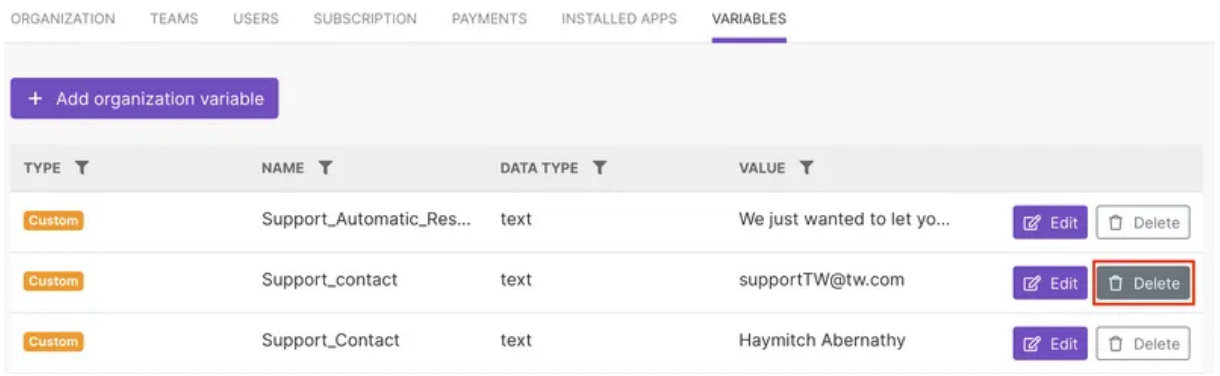
> Deleting a variable that you already use in one or more scenarios can affect other users who use the same variable in their scenarios. After you delete a variable, the variable becomes inactive in all scenarios where it is used and it stops returning expected values.

1. Decide if you want to delete an organization variable or a team variable.

   - For organization variables, go to the organization dashboard and click **Variables**.
   - For team variables, go to the team dashboard and click **Variables**.



2. In the list of your custom variables find the one you want to delete and click **Delete** next to it.



Delete_a_custom_variable_.png

3. Click **OK** to confirm that you want to delete the variable.

> If the custom variable you are deleting is already used, you will see a pop-up that lists all scenarios that will be impacted. If you still want to delete it, click **OK**.

The variable is deleted and disappears from your list of custom variables.

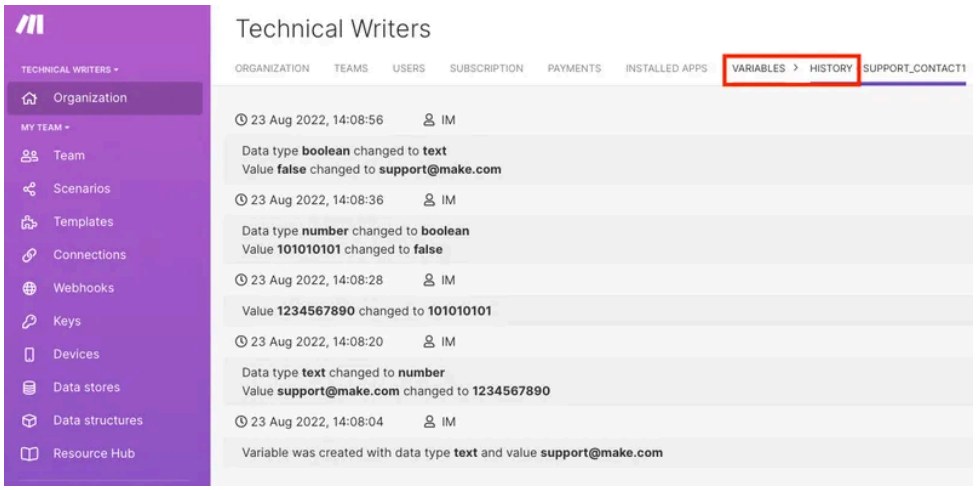
Check custom variable history

You can see who changed the custom variable, when, and what the changes are.

1. Go to the custom variable whose history you want to check.

   - For organization variables, go to the organization dashboard and click **Variables**.
   - For team variables, go to the team dashboard and click **Variables**.

2. Click the dropdown menu next to **Edit**. Select **Show history**.



A new window opens where you can see the variable history. The last change appears on the top.

User permissions for custom variables

Organization variables

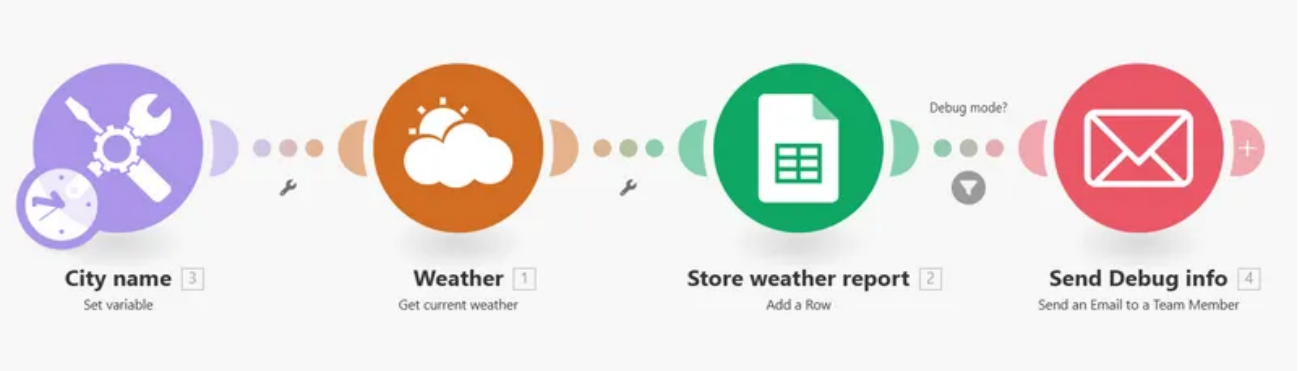
	Owner	Admin	Member	Accountant	App Developer
Can access organization variables	X	X	X		X
Can edit organization variables	X	X			
Can add organization variables	X	X			
Can delete organization variables	X	X			

Team variables

	Team Admin	Team Member	Team Monitoring	Team Operator	Team Restricted Member
Can access team variables	X	X		X	X
Can add team variables	X	X			X
Can edit team variables	X	X			X
Can delete team variables	X	X			X

Example: Use a custom variable to control a scenario

You can use a custom variable at the organization or team level to easily switch the value and control how a scenarioworks. Here, a custom variable called `debugMode` is boolean, and if the value is `true` an email will be sent to a team member. If you change the value to `false` at the team level, the scenario will end after adding a row to the Google Sheets document.



1. Decide if you want to create an organization variable or a team variable.

   - For organization variables, go to the organization dashboard and click **Variables > Add organization variable**.
   - For team variables, go to the team dashboard and click **Variables > Add team variable**.

   In this example, we are using a team variable.

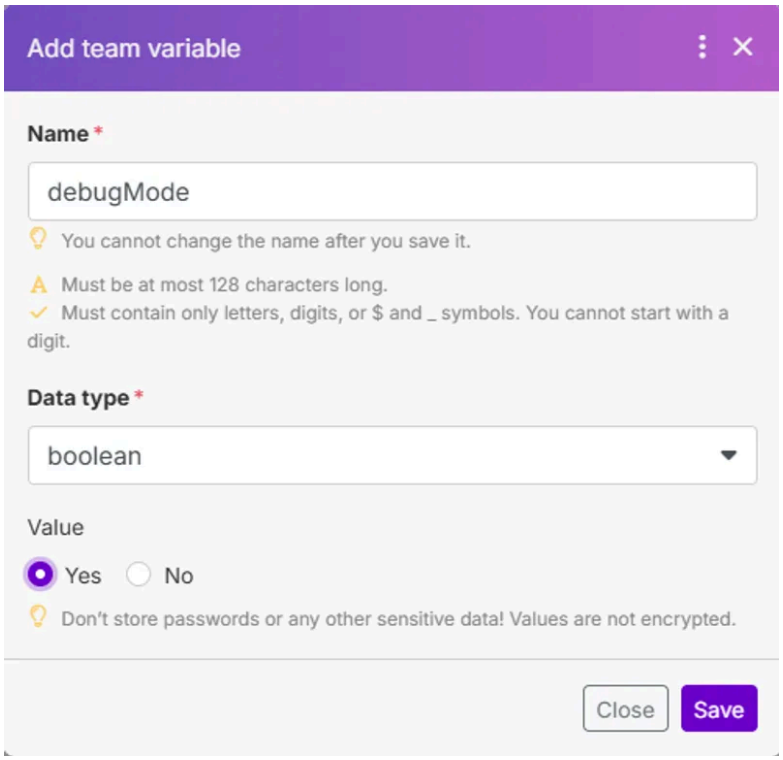
2. Enter your variable **Name**, **Data type**, and **Value**.

   Name: debugMode

   Data type: boolean

   Value: Yes

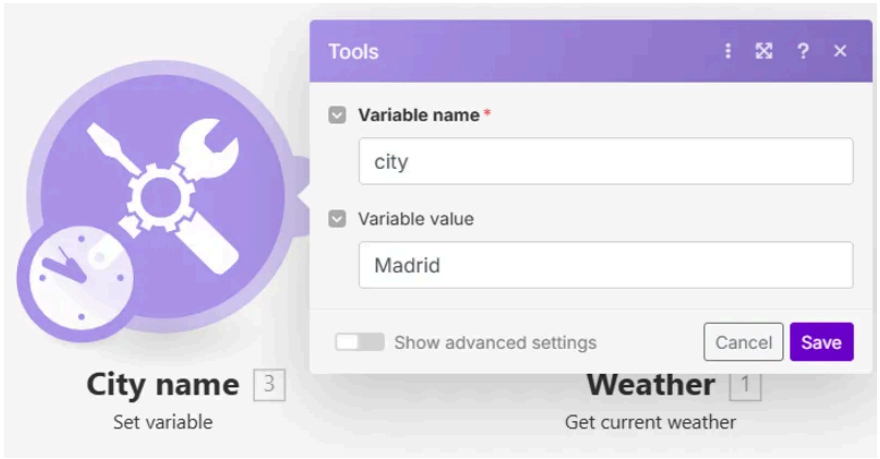
   Click **Save** to save your custom variable.



3 In your scenario, add a **Tools > Set variable** module.

Set the **Variable name** to `city` and the **Variable value** to `Madrid`.

Click **Save**.

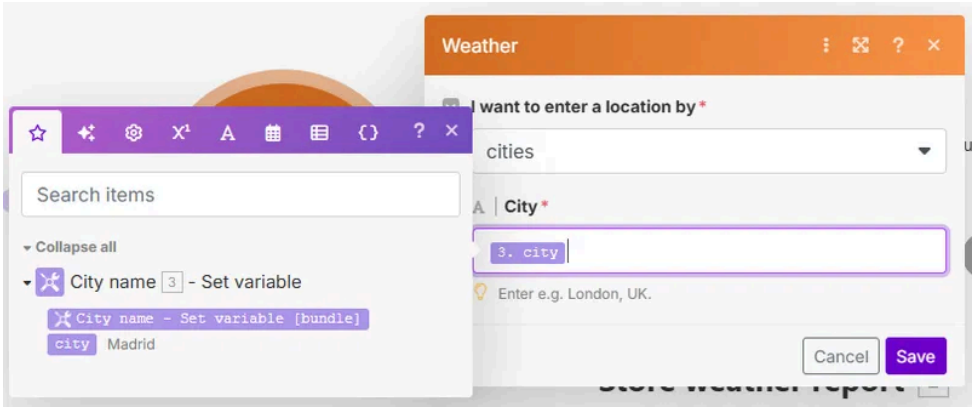


4 Add a **Weather > Get current weather** module.

In the **I want to enter a location by** field, select `cities`.

In the **City** field, map the value to the `city` variable that you set in step 3.

Click **Save**.



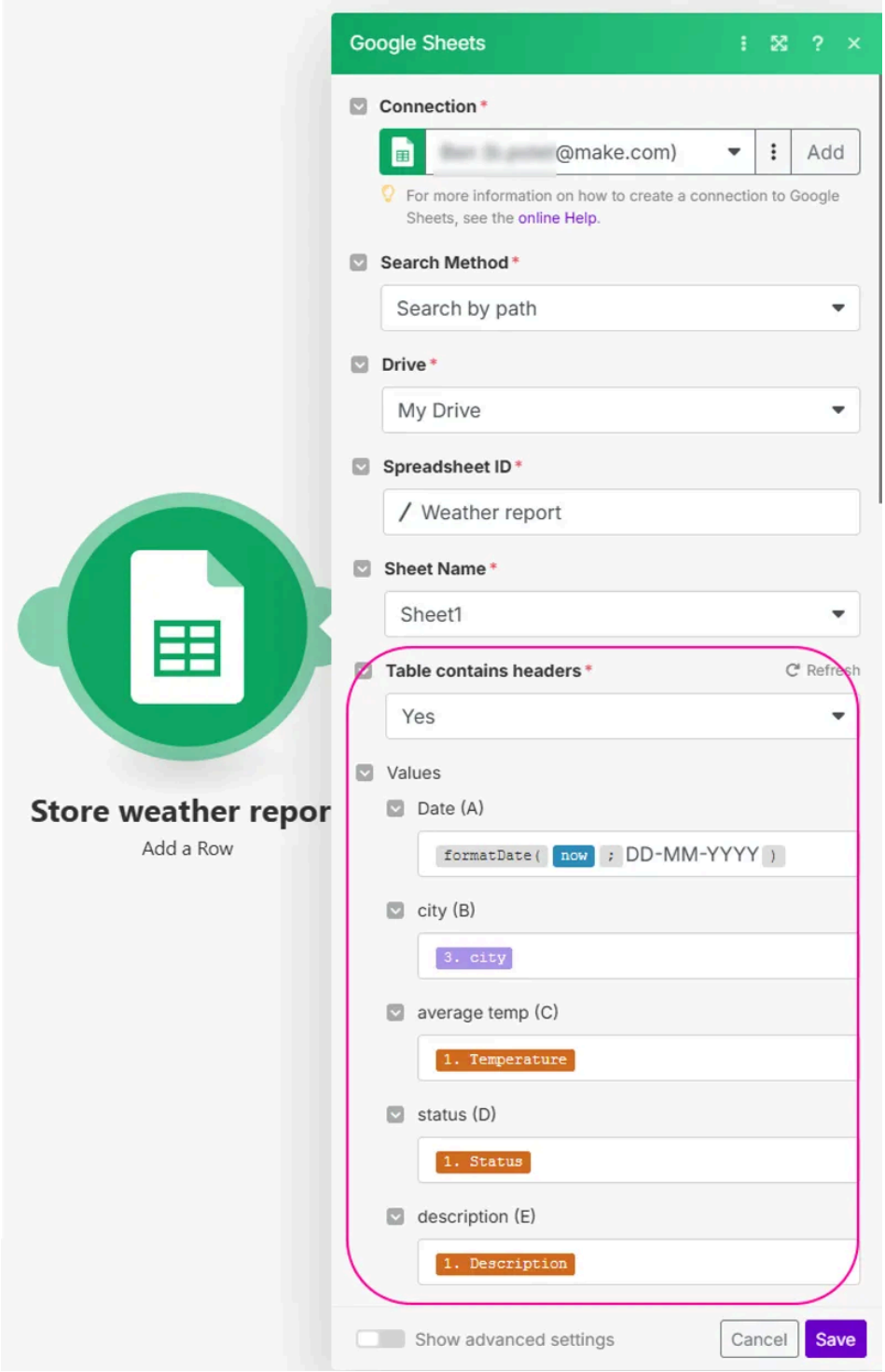
5 Add a **Google Sheets > Add a row** module and create a connection.

In this example, we have a Google Sheets document called **Weather report** saved in **My Drive**. This file has headers for the **date**, **city**, **average temp**, **status**, and **description**.

Connect to this file and map values as shown:

Field	Mapped value
date	formatDate(now; "DD-MM-YYYY")
city	Scenario variable `city`
average temp	Weather app value `Temperature`
status	Weather app value `Status`
description	Weather app value `Description`

Click **Save**.



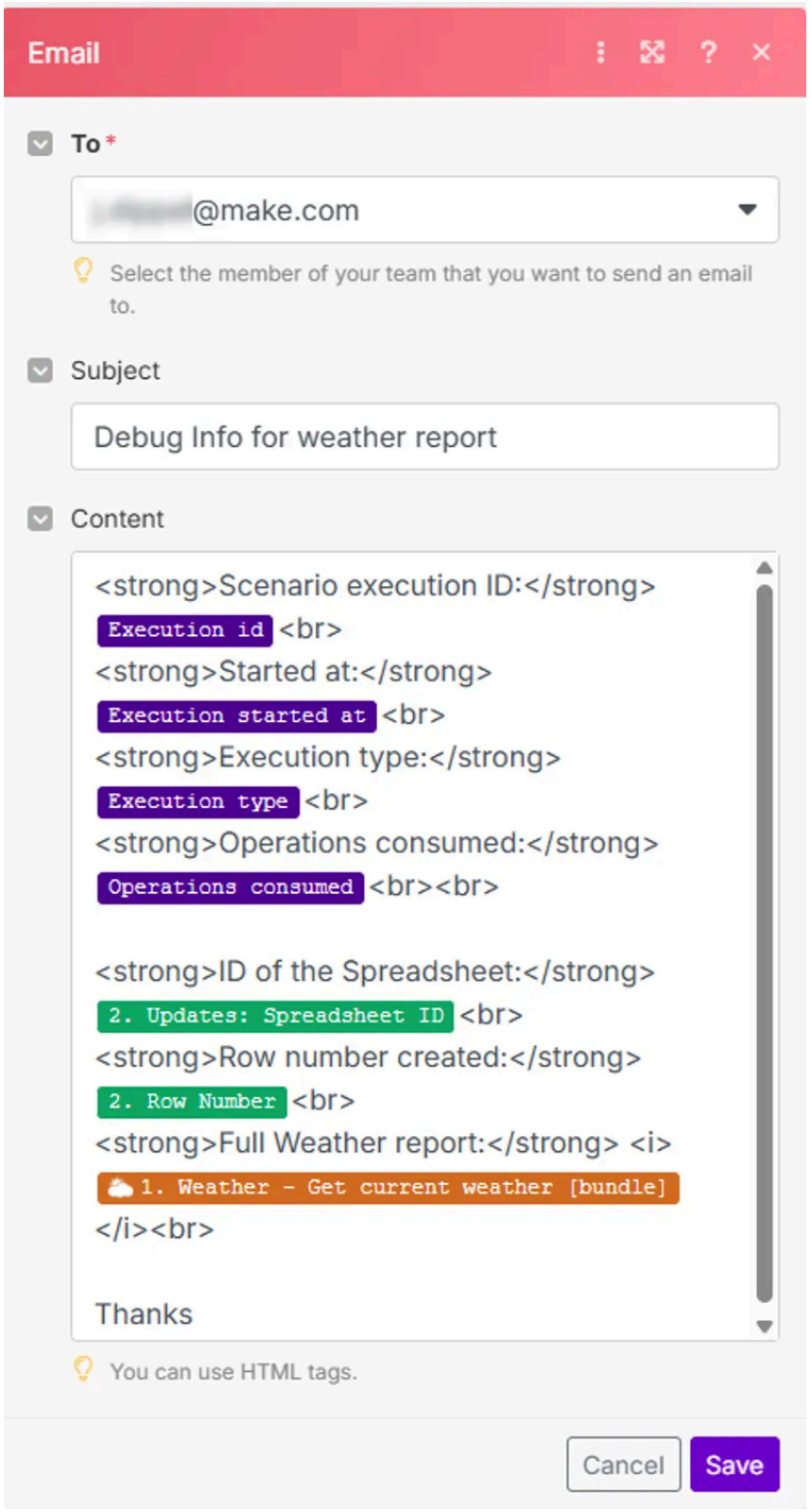
6 Add an **Email > Send an Email to a Team Member** module and create a connection.

In the **To** field, select the member of your team you want to send the email to.

In the **Subject** field, enter a relevant subject.

In the **Content** field, enter the text of your email, mapping the information you want to share with your team member. You can use HTML tags.

Click **Save**.



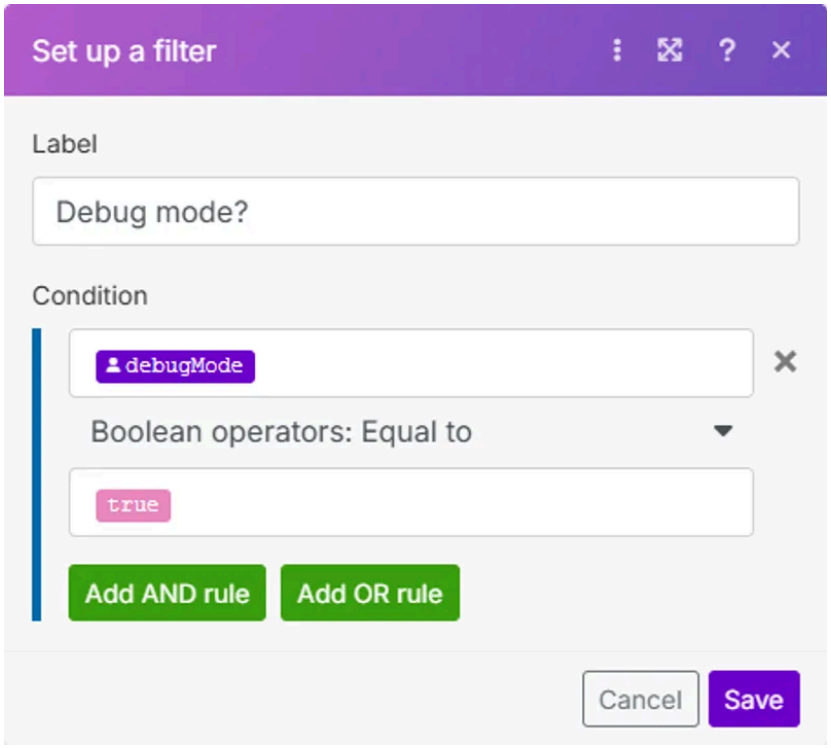
7 In your scenario, between the **Google Sheets** and **Email** modules, click the **wrench icon** to set up a filter.

Label: Debug mode?

Condition: Map the condition to the `debugMode` custom variable.

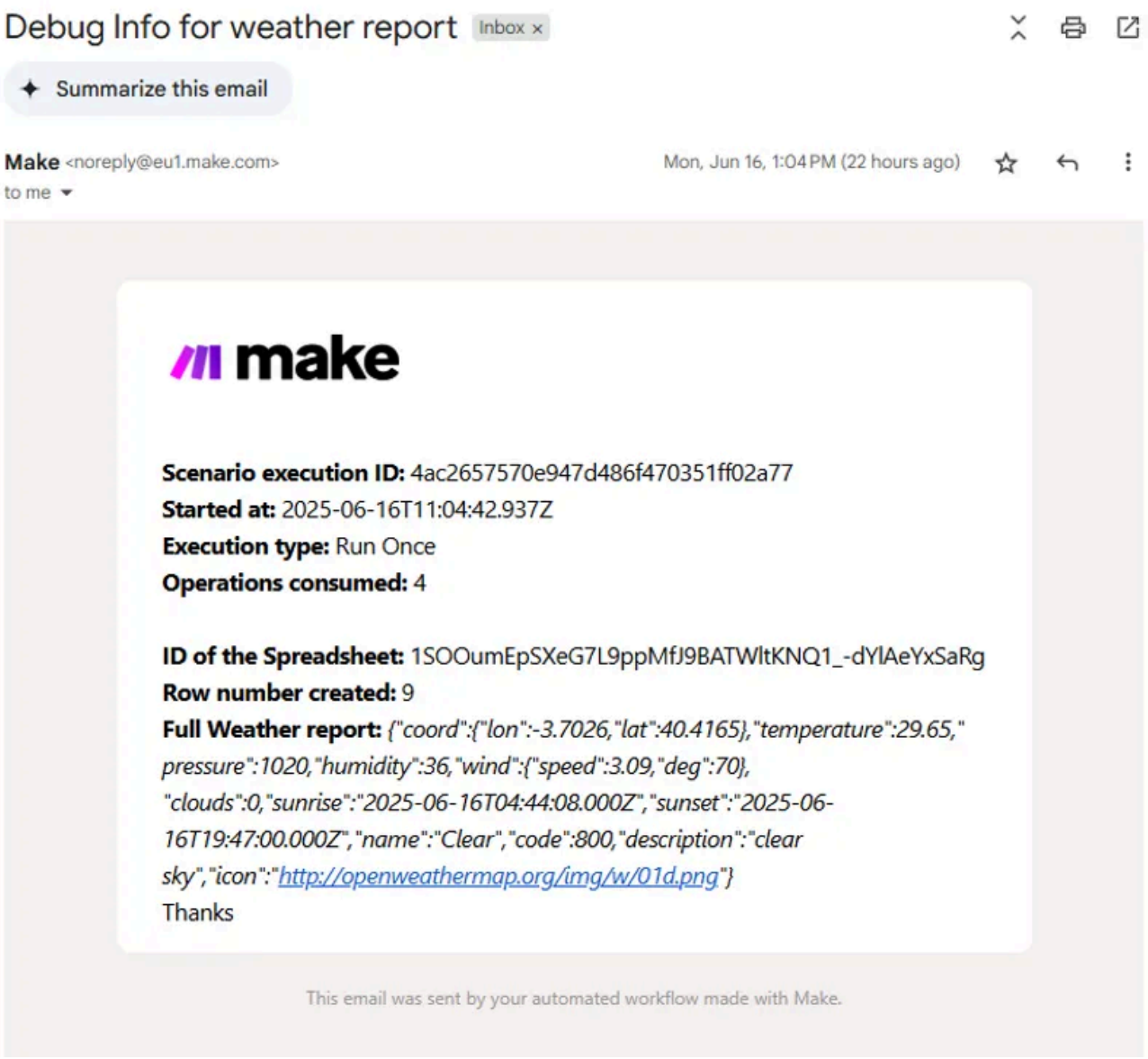
Change the operator to **Boolean operators: Equal to** and enter `true` in the field.

Click **Save**.



8 Click the save icon to save your scenario and click **Run once** to test.

If your custom variable value is `yes`, an email is sent to your team member.



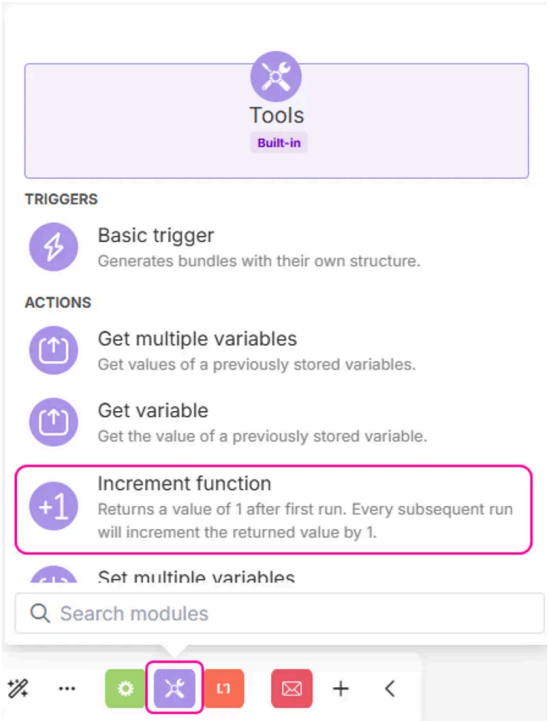
If the custom variable value is `no`, the email is not sent after the weather information is stored in the Google Sheets document.

3.1.7.4. Incremental variables

Incremental variables are variables that can be used when you want to keep track of a value when a scenario or a route in a scenario runs.

The incremental variable returns a value of `1` after the first run. The subsequent value is based on your choice for the value to never reset, reset after one cycle, or reset after one scenario run.

You can add an incremental variable to your scenario by clicking on to the **Tools** icon and selecting the **Increment function** module.



Example: Rotating the assignment of tasks to users in a group

Imagine that you receive a form submission requesting the completion of a task. In the following scenario, an incremental variable is set to count each time a request is submitted and send the task by email to different people, alternating the assignment of the task.

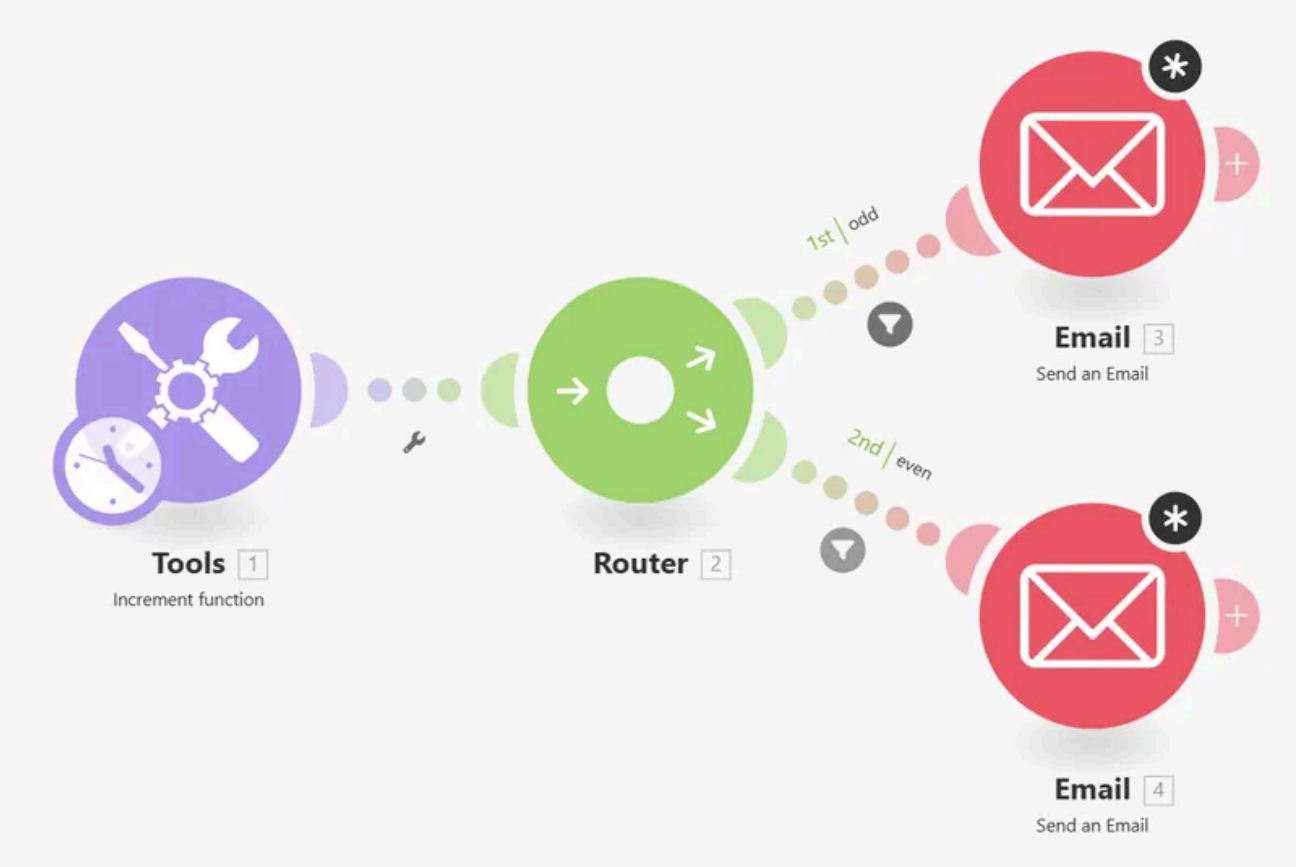
To alternate the tasks, we are using the `mod` function to filter the assignments between two people.

The `mod` function returns the remainder after dividing one number by another.

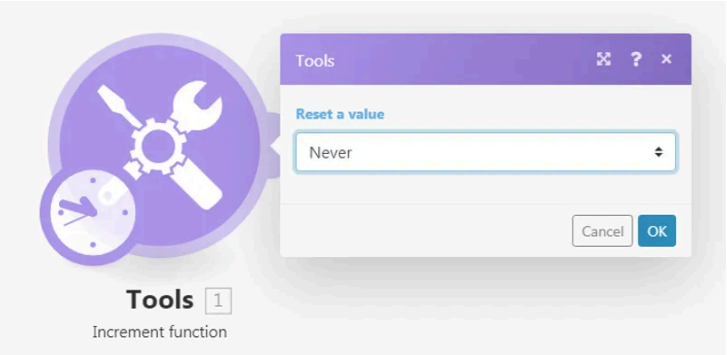
For example, 12 `mod` 2 = 0, because there is no remainder after dividing 12 by 2.

13 `mod` 2 = 1, because after dividing 13 by 2, there is a remainder of 1.

In this scenario, the incremental variable value is divided by 2 using the `mod` function, and the tasks are routed to different people based on the remainder being even (`0`) or odd (`1`).



1. Add a **Tool > Increment function** module and configure the value to never reset.



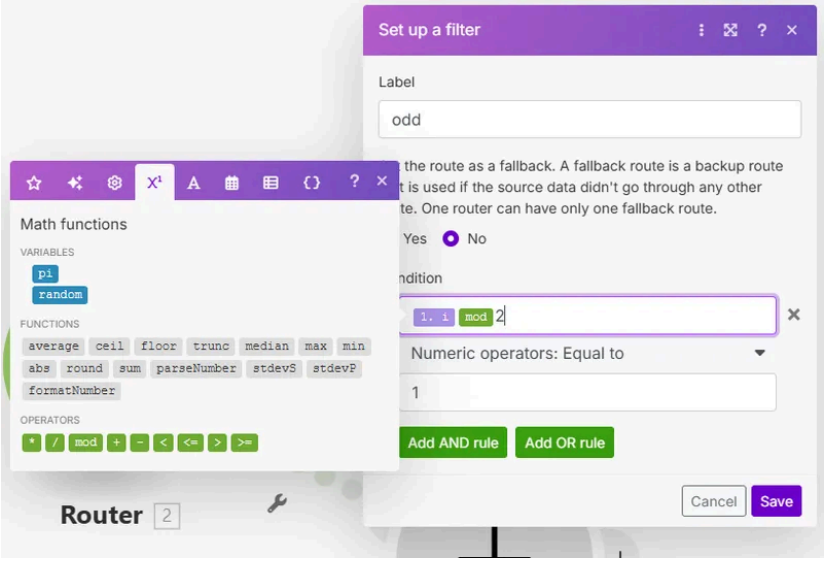
2. Add a router. You will see two default routes added.

3. Click the wrench icon for the first route and set the filter:

   Label: Odd

Condition: Map the incremental variable and use the `mod` math function and add the number `2`

Change the **Equal to** operator from the default **Text operator** to the **Numeric operator** and type `1` in the comparison field.

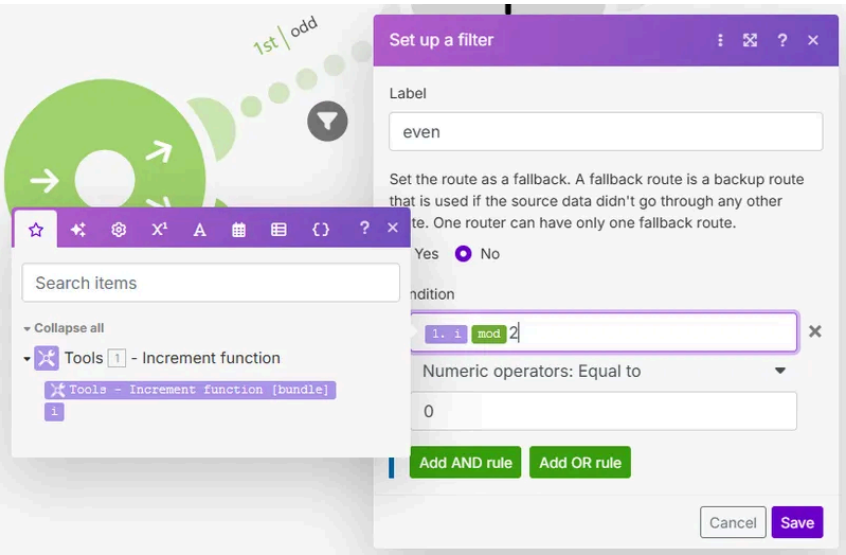


4 Click the wrench icon for the second route and set the filter:

Label: Even

Condition: Map the incremental variable and use the mod math function and add the number `2`.

Change the **Equal to** operator from the default **Text operator** to the **Numeric operator** and type `0` in the comparison field.



5. Complete your scenario by adding an Email module to the end of each route, notifying the recipient of their assignment.

As information comes in, the task assignment alternates between the two recipients.

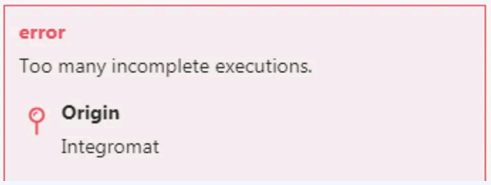
3.1.8. Incomplete executions

Incomplete executions are a safety feature that protects your scenarios from stopping due to errors and from data loss that could happen. When a scenario encounters an error with incomplete executions enabled, Make stores the unfinished scenario run in the **Incomplete executions** tab.

> Incomplete executions are disabled by default. To store incomplete executions, enable the **Allow storing incomplete executions** option in the scenario settings.

You can handle incomplete executions in the following ways:

- let Make retry the incomplete executions automatically for the **supported error types** or use the **Break** error handler
- **resolve** the incomplete executions manually
- **delete** the incomplete executions

> The maximum number of incomplete executions from all scenarios in all teams in an organization depends on your usage allowance. If the limit is exceeded, you will get the following error:
>
> 
>

3.1.8.1. Automatic retry of incomplete executions

Make checks the origin of every incomplete execution when it's created. Make automatically retries incomplete executions that have been created because of:

- `RateLimitError`
- `ConnectionError`
- `ModuleTimeoutError`

and incomplete executions that were created with the **Break error handler** with the automatic run completion enabled.

Make takes the following steps when Make retries incomplete executions:

1. Make schedules the incomplete execution retries.
2. Make retries the incomplete execution.
3. Based on the incomplete execution retry result Make schedules another attempt or marks the incomplete execution resolved.

Automatic retry scheduling

Make schedules automatic retries with the **incomplete executions backoff** schedule. The backoff schedule prevents the situation when you would get the same error multiple times in a row.

For example, when you get the `ConnectionError` because the app is unavailable, it might take some time until it's back. Make spaces out the retry attempts to get a successful retry even at a later time after the original error.

Error type	Retry schedule
• `RateLimitError` • `ConnectionError` • `ModuleTimeoutError`	1. 1 minute (1 minute after the original scenario run). 2. 10 minutes (11 minutes after the original scenario run). 3. 10 minutes (21 minutes after the original scenario run). 4. 30 minutes (51 minutes after the original scenario run). 5. 30 minutes (1 hour 21 minutes after the original scenario run). 6. 30 minutes (1 hour 51 minutes after the original scenario run). 7. 3 hours (4 hours 51 minutes after the original scenario run). 8. 3 hours (7 hours 51 minutes after the original scenario run).
Error handled by the **Break** error handler	If you enable automatic scenario run completion in the error handler settings. The default is: • maximum number of retry attempts: 3 • retry delay: 15 minutes You can customize the defaults in the error handler settings.

Other **error types** usually require changes in the incomplete execution and **manual resolving**. Make doesn't retry these error types automatically by default.

Automatic retry processing

After Make schedules the retries, Make runs the scenario again, starting with the module that caused the error.

For each scenario, there is a limit of 3 incomplete execution retries running in parallel. If there are more incomplete executions scheduled from the same scenario, Make retries them in batches of 3 after the previous batch finishes.

In addition, the retry doesn't start when the original scenario is running already.

The 3 parallel retries limit applies to retries from the same scenario. When Make retries incomplete executions from multiple scenarios, then each of them has their own limit.

> This limitation is to prevent your scenarios from getting follow-up rate limit errors if you are retrying a lot of incomplete executions at the same time.

For example:

You have a scenario that runs for 10 minutes every hour. There was a disruption of a third party service for 5 hours.

- The scenario now has 5 incomplete executions scheduled for automatic retry.
- Make first waits until the original scenario finishes if it's running already. This takes 10 minutes if the scenario started just now.
- After the scenario finishes, Make retries the first 3 incomplete executions. This takes an additional 10 minutes (20 in total).
- Make retries the remaining 2 incomplete executions after the previous batch finishes. This takes another 10 minutes (30 in total).
- After another 30 minutes (1 hour in total), Make starts the scenario again according to the scenario schedule.

Automatic retry result

If a retry attempt succeeds, Make marks the incomplete execution as **Resolved** and stops retrying.

If all of the retry attempts fail, Make marks the incomplete execution as **Unresolved**. You can then retry the incomplete execution when the app that caused the error is available again or you can resolve the incomplete execution manually.

3.1.8.2. Manage incomplete executions

When dealing with scenario incomplete executions, you should first decide if you want to finish the run and process the rest of the data in the scenario. If you want to finish the scenario run, you should check the error that created the incomplete execution:

1. Go to the **Incomplete executions** tab.
2. Find the incomplete execution and click **Details**. The Scenario Builder opens.
3. Click the warning sign on the module that caused the error. You can also check the module settings or the **scenario execution history**.



If the error is a temporary error, like the `ConnectionError` or the `RateLimitError`, you can **retry the incomplete executions**. Retrying an incomplete execution is like another attempt to run a scenario. The incomplete execution runs with the same module settings as when the error happened.

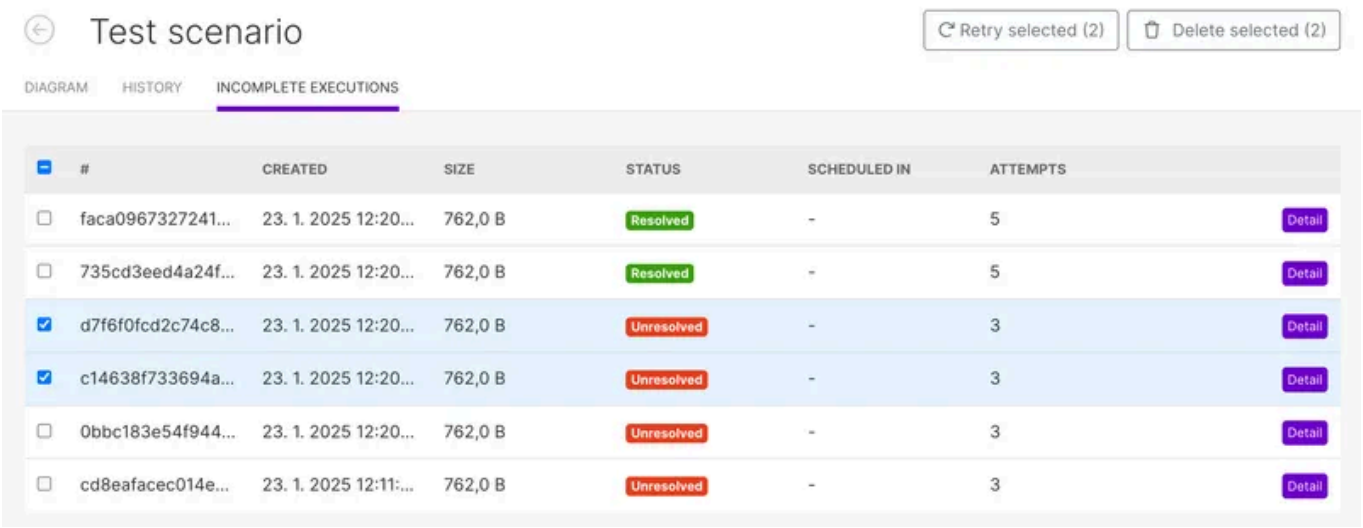
> You can retry multiple incomplete executions at once.

If the error requires changes in the module settings or in the general scenario blueprint, you need to **resolve the incomplete execution** manually.

If you find out that you don't need to run the rest of the scenario, you can **delete the incomplete execution**.

The incomplete executions tab

To open the incomplete executions tab, select the scenario in the list of scenarios and switch from the **Diagram** tab to the **Incomplete executions** tab.



In the scenario's **Incomplete executions** tab, you can find the following information for each incomplete execution:

- **Selection check box**: You can select incomplete executions to either retry or delete. You can select multiple incomplete executions, or you can use the check box in the table header to select or de-select all incomplete executions of a scenario.
- **Incomplete execution ID**: A unique text generated by Make to identify the incomplete execution. You can use the incomplete execution ID with the **Make app** or with the **Make API**.
- **Created**: The date and time when Make created the incomplete execution.
- **Size**: The size of the incomplete execution blueprint and the data that the module was processing when it failed.
- **Status**: Possible incomplete execution statuses are:
  - **Unresolved**: The incomplete execution is not set to retry and hasn't been resolved. To fix this, you can **retry** or **manually resolve** the incomplete execution.
  - **Pending**: The incomplete execution is set to retry. Check the **Scheduled in** column to see approximately when Make will retry the incomplete execution.

    - **In progress**: Make is currently retrying the incomplete execution.
    - **Resolved**: The incomplete execution has been resolved successfully. Make will delete the resolved incomplete execution automatically after 30 days.
- **Scheduled in**: The approximate time until Make will retry the incomplete execution.
- **Attempts**: The number of attempts already made to resolve the incomplete execution.
- **Detail**: The **Detail** of the incomplete execution opens the Scenario Builder with the incomplete execution. In the incomplete execution detail, you can check the cause of the incomplete execution and **resolve the incomplete execution**.

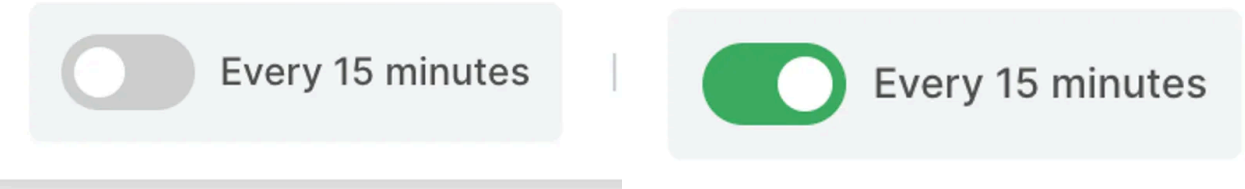
Retry incomplete executions

Retrying incomplete executions runs the incomplete execution again, starting with the module that caused the error. Retried incomplete executions run with the same configuration as when the error happened, so a retry works best for temporary errors, like the `RateLimitError` or the `ConnectionError`.

To retry incomplete executions:

1. Make sure that the scenario is active. You can retry incomplete executions only when the scenario is active.

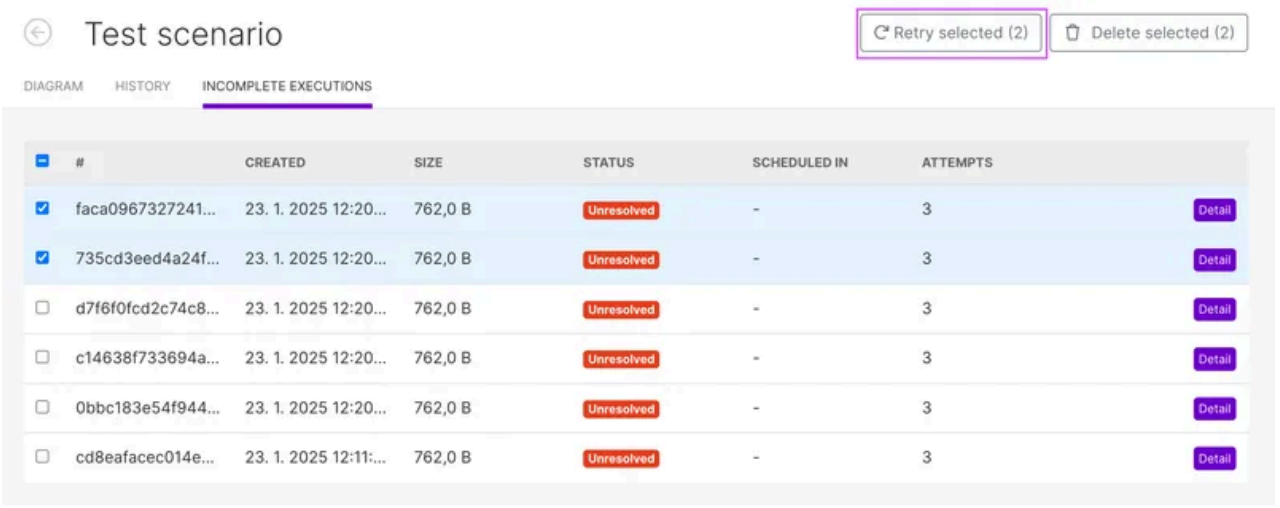
   To activate a scenario, go back to the **Diagram** tab or the Scenario Builder and click the switch next to scenario scheduling on the scenario toolbar:

2. Go to the **Incomplete executions** tab.

3 Select incomplete executions you want to retry. You can use the checkbox at the top of the incomplete executions table to either select or de-select all incomplete executions of the scenario.

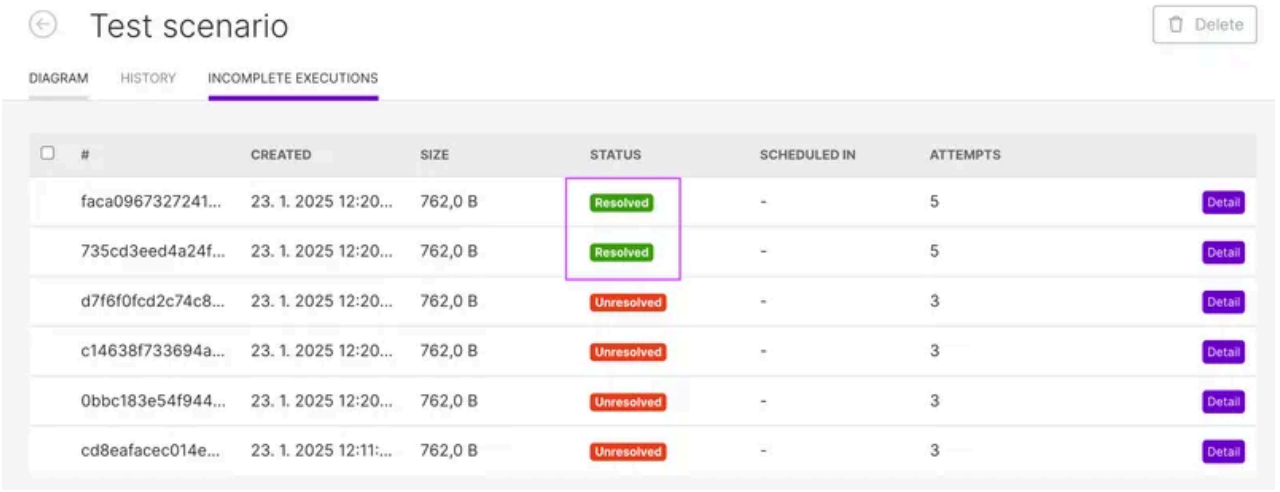
4 Click the **Retry selected** button to retry the incomplete executions.



The status of the selected incomplete executions turns into **Scheduled in,** and to **In Progress** when Make starts retrying them.

If you are retrying a lot of incomplete executions, it might happen that some of them get scheduled for later processing instead. The **Scheduled in** column contains the information of when approximately Make retries the incomplete execution.

5 If the incomplete execution is resolved successfully, the status of the incomplete execution turns into **Resolved**. Otherwise, the status stays **Unresolved**.



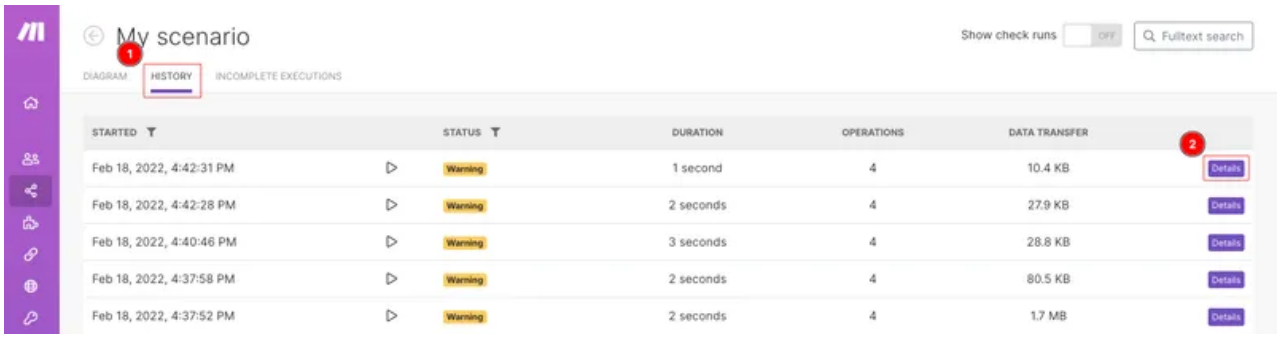
Resolve an incomplete execution

Some errors create incomplete executions that require manual changes before the incomplete execution can be resolved. For example, the `RuntimeError` and `DataError` most of the time have to be resolved manually instead of retrying. You can resolve an incomplete execution manually either from the scenario run history, or from the incomplete executions tab:

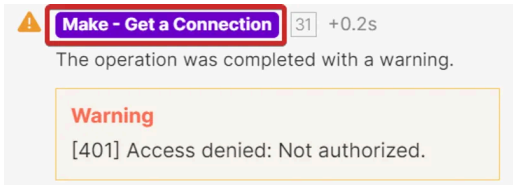
You might want to check the scenario execution log first to investigate why the error happened:

1. Go to the **History** tab.

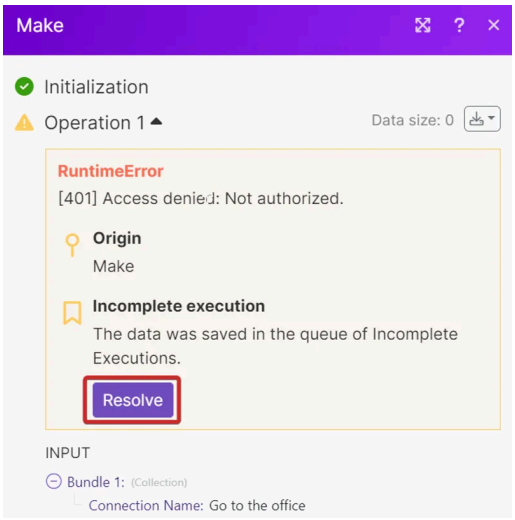
2. Locate the failed execution log and click **Details**.



3. Click the module that has a warning status.



4. Click **Resolve**. Make redirects you to the incomplete execution.



5. Check the module that caused the error and fix it.

6. Save the updated module settings with the **Save** button.

7. Click **Run once**.

The incomplete execution runs again, starting with the module that caused the incomplete execution with the original input. If the run succeeds, the incomplete execution is resolved.

Otherwise, if resolving fails on the same module, the incomplete execution is not resolved. You need to fix all errors on the module to resolve the incomplete execution.

If the incomplete execution fails on a different module, Make creates a new incomplete execution, starting with the module that caused the new error.

If you want to go directly to the incomplete execution:

1. Go to the **Incomplete executions** tab.

2. Find the incomplete execution and click **Details.**

3. Check the module that caused the error and fix it.

4. Save the updated module settings with the **Save** button.

5 Click **Run once**.

The incomplete execution runs again, starting with the module that caused the incomplete execution with the original input. If the run succeeds, the incomplete execution is resolved.

Otherwise, if resolving fails on the same module, the incomplete execution is not resolved. You need to fix all errors on the module to resolve the incomplete execution.

If the incomplete execution fails on a different module, Make creates a new incomplete execution, starting with the module that caused the new error.

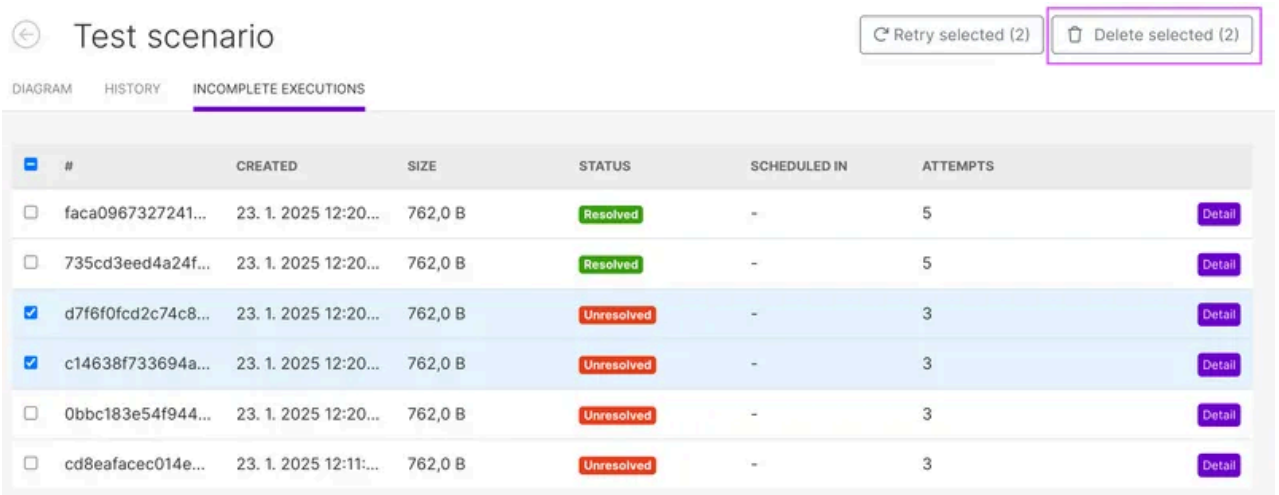
Delete incomplete executions

When you don't need to finish the scenario workflow, or you don't need the data from the incomplete execution anymore, you can delete the incomplete exectution. To delete incomplete executions:

1 Go to the **Incomplete executions** tab.

2 Select incomplete executions you want to delete. You can use the checkbox at the top of the incomplete executions table to either select or de-select all incomplete executions of the scenario.

3 Click the button **Delete selected** to delete the incomplete executions.



The deleted incomplete executions are no longer in the list of incomplete executions.

> Once you delete an incomplete execution, it cannot be put back.

3.1.8.3. Errors that don't create incomplete executions

Most of the errors that can happen in Make are connected with the flow of data in your scenario (like the `DataError`) or with the third party application (like the `ConnectionError`).

However, some errors don't create an incomplete execution:

- When the error happens on the first module in the scenario.
  However, you can add the **Break** error handler ot the first module in the scenario. With the **Break** error handler, Make stores the incomplete execution even when the first module in the scenario outputs an error.
- When your incomplete executions storage is full. If your incomplete executions storage is full, Make checks the **enable data loss** setting:
  - If the data loss is disabled, Make disables the scenario.
  - If the data loss is enabled, Make keeps scheduling scenario runs and discards the incomplete execution if it cannot be stored in your account.
- When the scenario runs longer than the scenario run duration limit. You can check the limit for your plan on the Make **pricing**.
- When an error happens during the initialization or rollback **scenario phase**. Since these errors happen outside of the scenario operation phase, there is no incomplete scenario run.

3.1.8.4. Options related to incomplete executions

The following options in Make scenario settings determine if and how the incomplete executions are stored:

- **Allow storing incomplete executions**
  This option enables incomplete executions for the scenario. If this option is disabled, Make doesn't store incomplete executions of the scenario.
- **Sequential processing**
  Sequential processing ensures that the scenario runs in a sequence. If there is an incomplete execution of the scenario, Make pauses further scheduling of the scenario to keep the processing in the chronological sequence. Make activates the scenario again after the scenario has no incomplete executions.

  If the scenario has instant scheduling, Make stores the arriving bundles in the webhook queue.
- **Enable data loss**
  Data loss setting controls what should happen when Make cannot create an incomplete execution of a failed scenario run. This happens in most cases because the incomplete execution storage is full. Incomplete execution storage is limited based on the usage allowance.

  If you keep data loss disabled, Make pauses scheduling of the scenario to avoid losing any more scenario runs, until you clear the incomplete execution storage and enable the scenario again.

  If you enable data loss, Make will continue scheduling the scenario even if it couldn't store the incomplete execution.

3.1.9. Scenario execution, cycles, and phases

Make is a transactional system, similar to **relational databases**. Each scenario execution starts with the **initialization phase**, continues with at least one **cycle** composed of the **operation and commit/rollback phases**, and ends with the **finalization phase.**

During the **initialization phase**, all necessary connections are created. They are also checked if each module is capable of performing their intended operation(s).

Each **cycle** represents an undividable unit of work composed of a series of operations. It is possible to set the maximum number of cycles in the scenario settings. The default number is 1.

During the **operation phase,** reading and writing operations are performed.

- The reading operation consists of obtaining data from a service that will then be processed by other modules according to a predefined scenario. For example, the **Dropbox > Watch files** module returns new bundles (files) created since the last scenario execution.
- The writing operation consists of sending data to a given service for further processing. For example, the **Dropbox > Upload a file** module uploads a file to a Dropbox folder.

If the operation phase is successful for all modules, the **commit phase** begins, during which all operations performed by the modules are committed. This means that Make sends information to all the services involved in the operation phase about its success.

If an error occurs during the operation or commit phase for any module, the phase is aborted and the **rollback phase** is started, making all operations during the given cycle void. Some modules do not support rollback and operations performed by these modules cannot be taken back.

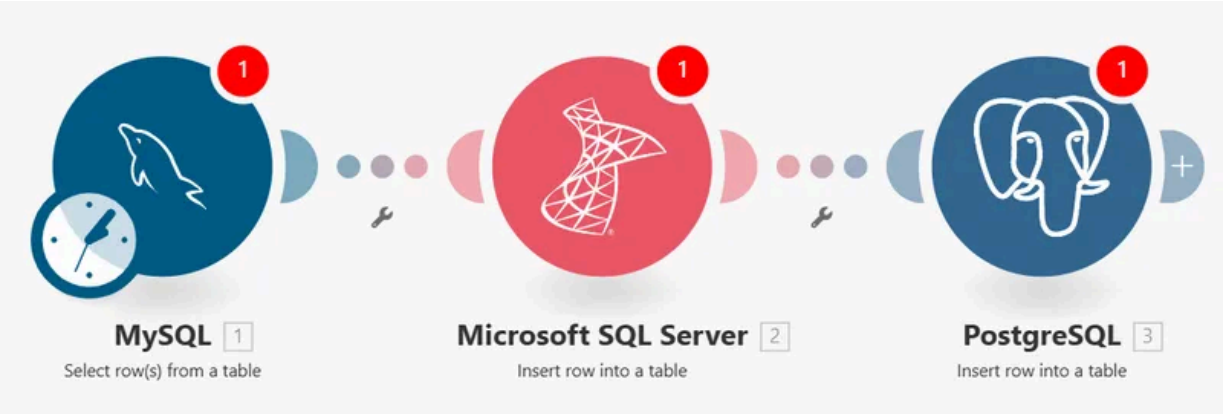
> Modules that support rollback and marked with the `ACID` tag. Modules without this tag do not support rollback and cannot be reverted back to their initial state in case of an error in other modules.

During the **finalization phase**, open connections (e.g. FTP connections, database connections, etc.) are closed and the scenario is completed.

Example

Transfer of bundles between databases

The following example shows how to connect three `ACID` modules. The aim of the scenario is to get new rows from a MySQL database, insert (transfer) them into an MSSQL database, and then insert the IDs of the rows from the MSSQL database into a PostgreSQL database.



When the scenario starts, the **initialization phase** is performed first. Make verifies connections to the MySQL, MSSQL and PostgreSQL databases one at a time. If everything goes well and the connections are successful, Make moves on to the **operation phase**. If an error occurs, the **finalization phase** starts instead of the **operation phase** and the scenario is terminated.

If there isn't an error, the **operation phase** begins. A preset procedure selects (reads) the table rows (bundles) from MySQL. Those rows are then passed to the next module that writes them to a selected table in the MSSQL database. If everything is in order, the last PostgresSQL procedure is called to insert the row IDs returned by the preceding module into the table.

If the operation phase is completed successfully, the **commit phase** begins. Make calls the SQL COMMIT command for each database and the write operations are committed.

However, if the operation or commit phase fails due to an error, (e.g. connection failure), Make calls for a rollback. During the **rollback phase**, Make goes through all modules and

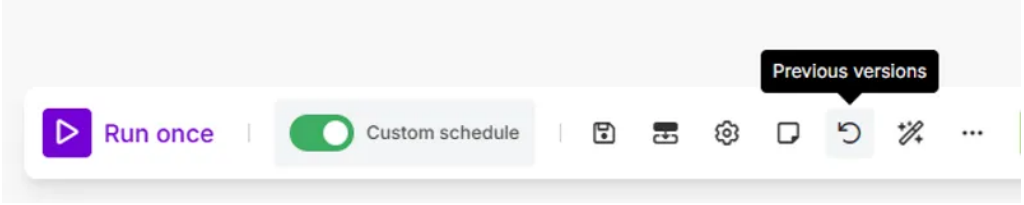
executes the SQL ROLLBACK command for each module to revert each database back to its initial state.

Finally, during the **finalization phase**, each module closes its connection to the database.

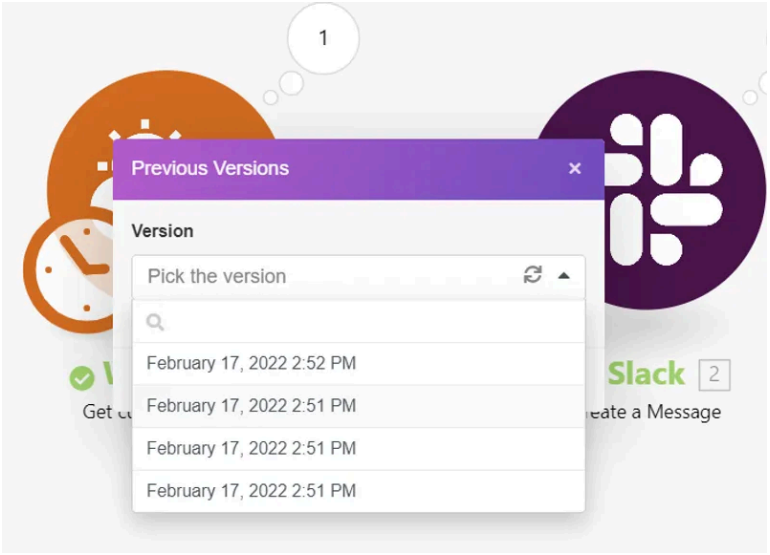
3.1.10. Restore a previous scenario version

1. Open your scenario.

2. Click on the **Previous versions** icon in the bottom of the scenario editor.

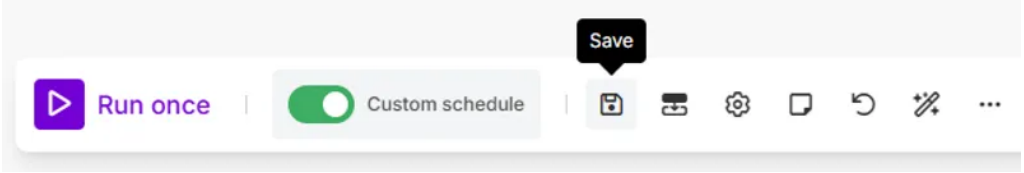


3. Select a previous version from the menu that you would like to restore.



4. Click **OK** to confirm.

5. Click the **Save** icon to save the scenario with the restored version.



The restored version of the scenario is not automatically saved. If you wish to save the restored version of the scenario, you have to do so manually, as described in step 5.

3.2. Connections

Find more connection options in these articles:

→ **Devices**

→ **On-premise agent**

→ **Keys**

→ **Dynamic connections**

→ **Certificates and keys**

→ **Allow connections to and from Make IP addresses**

→ **Connect to Google services using a custom OAuth client**

→ **Call Google APIs with the 'HTTP Make a OAuth 2.0 request' module**

→ **Connect Make to any web service that uses OAuth2 authorization**

3.2.1. Devices

To use Make for Apple iOS or Make for Android apps, it is necessary to add your iOS or Android device to your Make account.

You can add your device to Make by opening the mobile app and scanning a QR code generated on the **Devices** page.

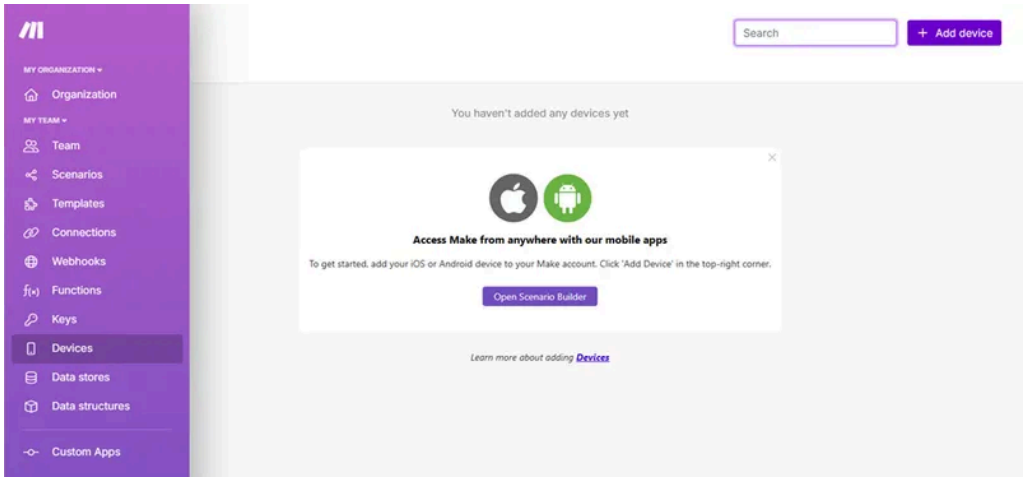
Prerequisites:

- Make for Apple iOS app or Make for Android app installed on your device.

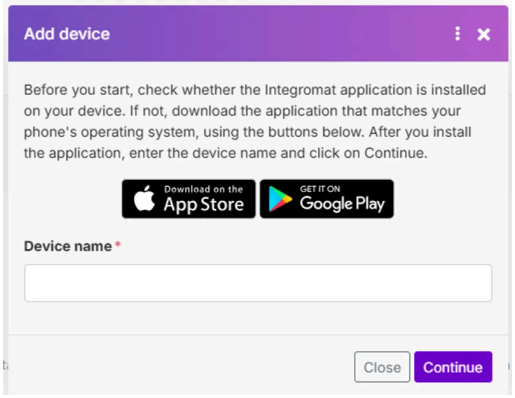
Add a device using a QR Code

1. In the left sidebar, click **Devices**.

2. Click **+Add device**.



3. Under **Device name**, enter the name of your device and click **Continue**.

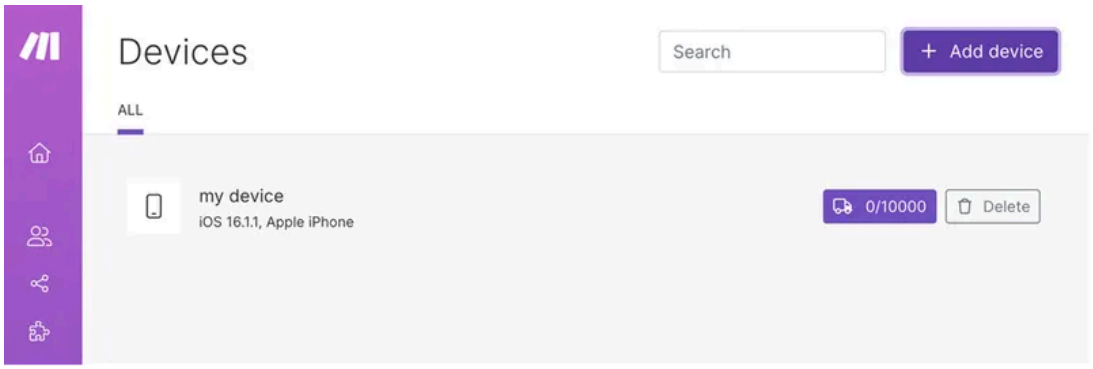


4. Open the Make app on your iPhone or Android device, and tap **Scan QR Code**.

5. Scan the QR code.

6. Click **Save**.

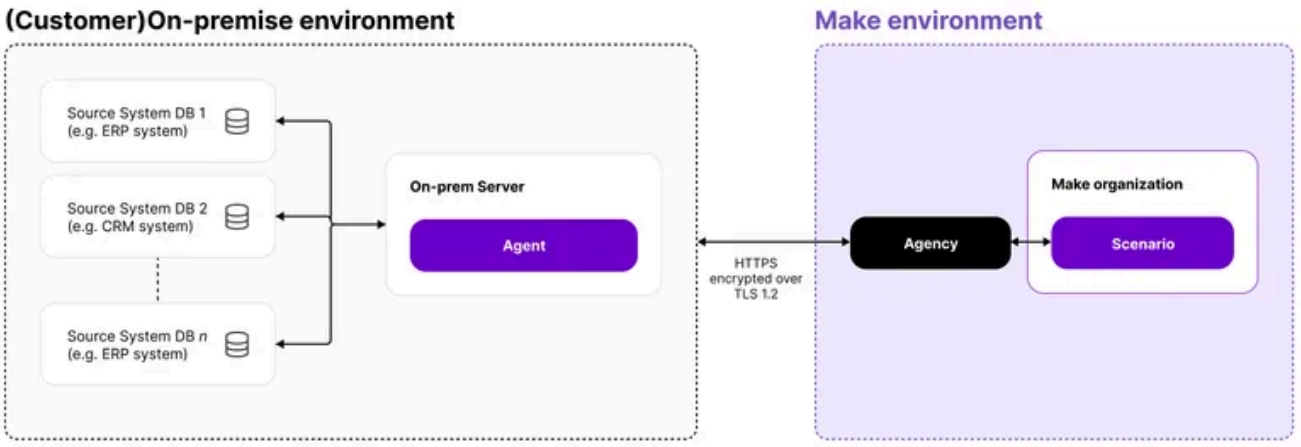
Your mobile device now displays your account settings. Your browser now lists your device on the Devices page.



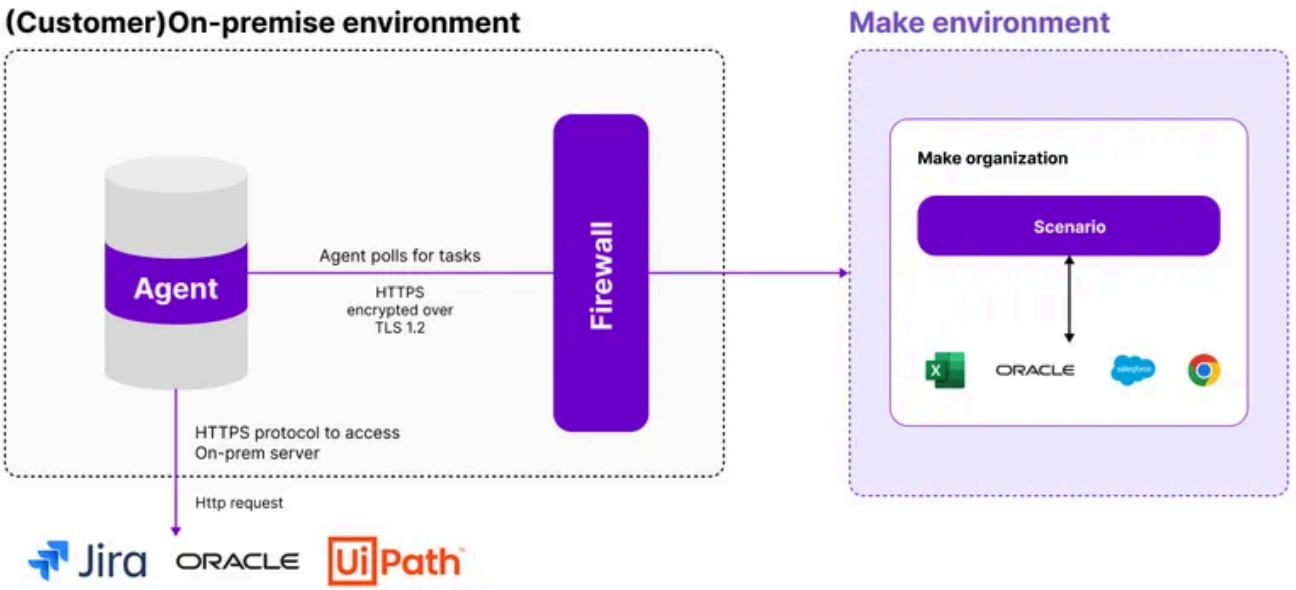
3.2.2. On-premise agent

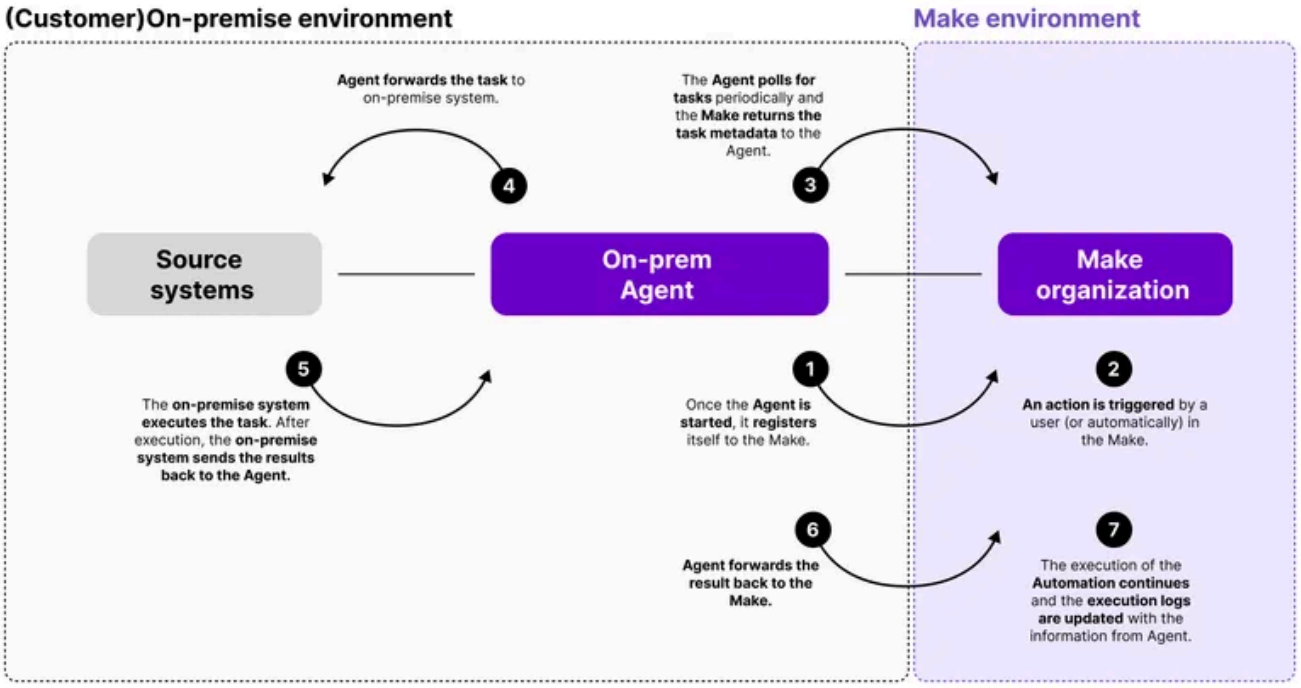
> This feature is available to Enterprise customers.

With the On-premise agent (On-prem agent) installed on your system, your scenarios can access applications and databases on your local network without changing the settings of your firewall.



One On-prem agent can connect to many applications that are located in the same network. Currently, we provide an **HTTP Agent** app that allows for a connection with other custom systems that provide an API or a web service.





To start using an On-prem agent in your scenario, you need to:

1. **Create** a new On-prem agent in the **On-prem agents** section of the **Organization** dashboard.
2. **Download** the On-prem agent and **install** it on your device.
3. **Connect a system** (an application) to the installed On-prem agent.
4. **Add a connection** between Make and the connected system when adding the application module in your scenario.

Add an On-prem agent

The installation process consists of three steps:

- Creating an On-prem agent in the **On-prem agents** section, in Make.
- Downloading the On-prem agent to your device.
- Installing the On-prem agent on your device.

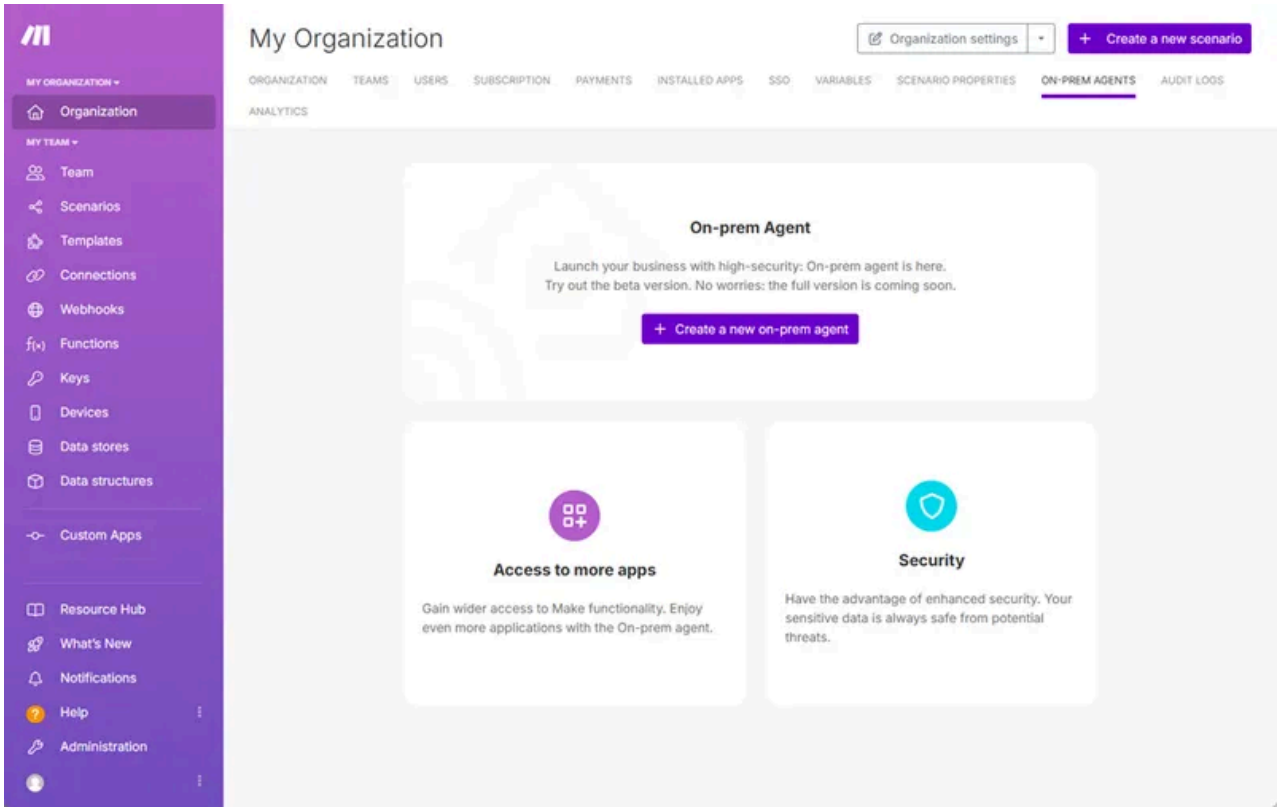
> Only users with the **Owner**, **Admin**, and **App dev** organization roles can perform all three steps.

Create an On-prem agent

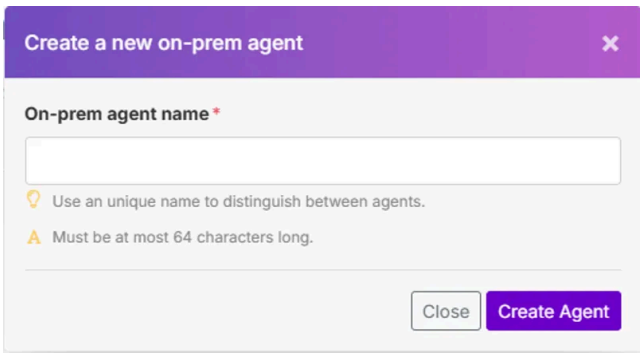
1. In the left sidebar, click **Organization**.

2. Click the **On-prem agents** tab.

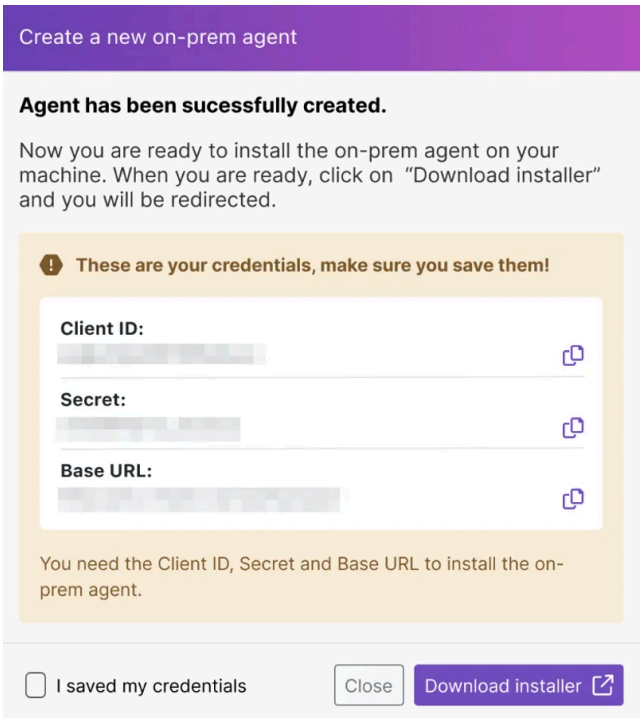
3. Click **+ Create a new on-prem agent**.



4. In the **On-prem agent name** field, enter the agent name.



5. Click **Create agent**. A dialog box with credentials appears.



6 Save the **Client ID**, the **Secret**, and the **Base URL** in a safe place. You will need these credentials in the installation stage later.

> Make sure to save the credentials in a secure location. This is the only time when Make shows the On-prem agent credentials.

7 Check **I saved my credentials**.

8 Click **Download installer**.

Make redirects you to the download page, where you can choose your device OS and **download the On-prem agent installer**.

Download the On-prem agent

You need to download the installation package for the On-prem agent to run on your device.

1 Select your device's operating system and click **Download**.

For macOS, download the file for Linux.



If the auto-download doesn't start automatically, use the link instead.

2. Define where you want to save the On-prem agent installation package on your device.

3. Click **Save**.

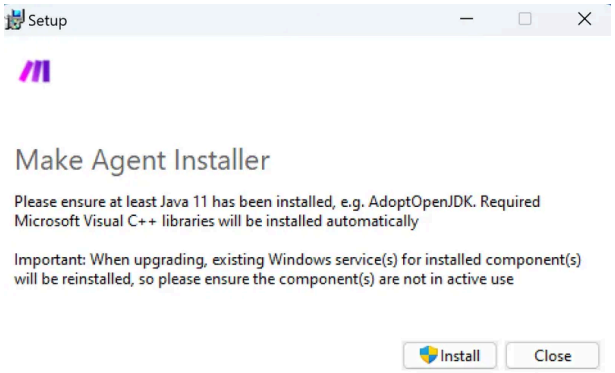
Install the On-prem agent

Installation is the last step you need to start using the On-prem agent in Make. Here you will find steps for installing the On-prem agent on Windows, macOS, and Linux.

Installation for Windows

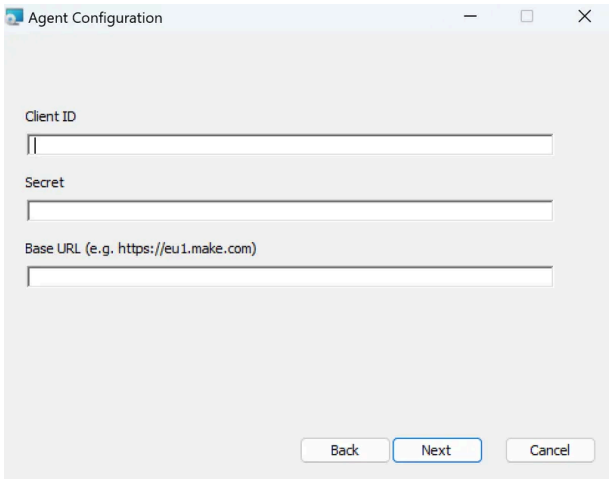
> You need to have Java 11 or above already installed on your device:

1. Go to the folder where you saved the On-prem agent when downloading, right-click the file, and select **Run as Administrator**. An installation window appears. Click **Install**.

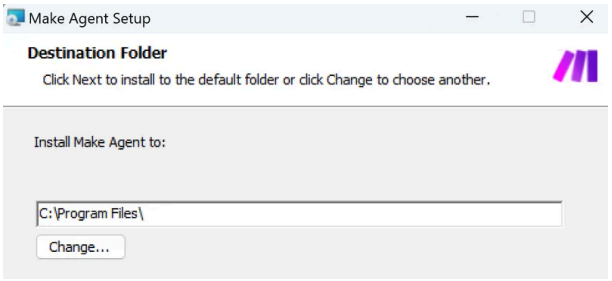


You might see a warning. Click **Yes** to continue the installation process.

2. Insert the credentials you saved when **creating the On-prem agent** in Make and click **Next**.



3. Select the path for installation and click **Next**.



4. Make sure your internet connection is stable. Click **Install**.

5. Complete the installation by clicking **Finish**.

6. A confirmation window appears. Click **Close**.

The On-prem agent status changes to **Active** in Make after refreshing the On-prem agents page. You can **connect a system** to it.



You can also check your system to see if the installation was successful:

1. Open the **Services** app on your computer.

2. In the list of services, find **Make Agent**. If the status is **Running**, you can start using the On-prem agent in Make. If there is no status, check the **troubleshooting guide**.

> The installer works only for the agent you've downloaded. If you need to run more than one agent on your server, you need to download the installer once again and repeat the installation procedure.

Installation for macOS

> You need to have Java 11 or above already installed on your device:

1. Go to the folder where you saved the .zip archive.
2. Unzip the archive.
3. Open the `application-local.yml` file and fill in the following fields with the data you saved when **adding an On-prem agent** in Make.
   - `make: authentication: client-id`
   - `make: authentication: client-secret`
   - `make: authentication: base-url`
4. Save the changes in the file.
5. Open the Terminal.
6. Open the folder that you unzipped from the archive in the Terminal. It is called **Linux-Mac**.
7. Launch the file using this command: `java -jar agent.jar`



The system launches the On-prem agent. If you see the `Successfully connected Agent with ID...` and `Started to poll for tasks` strings, it means the agent is installed successfully. You can connect a system to it.

Installation for Linux

> You need to have Java 11 or above already installed on your device:

1. Go to the folder where you saved the .zip archive.
2. Unzip the archive.
3. Copy the `application-example.yml` file, and rename the copied file to `application-local.yml`.
4. Open the `application-local.yml` file and fill in the following fields with the data you saved when **adding an On-prem agent** in Make.
   - `make: authentication: client-id`
   - `make: authentication: client-secret`
   - `make: authentication: base-url`
5. Save the changes in the file.
6. Open the Terminal.
7. Open the folder that you unzipped from the archive in the Terminal. It is called **Linux-Mac**.
8. Launch the file using this command: `java -jar agent.jar`



The system launches the On-prem agent. If you see the `Successfully connected Agent with ID...` and `Started to poll for tasks` strings, it means the agent is installed successfully. You can connect a system to it.

Connect a system

Once you installed the On-prem agent, you can connect it to an application (a connected system). You can connect several systems to one On-prem agent.

> Only users with the **Owner**, **Admin**, and **App dev** organization roles can connect a system.

Connect the HTTP system

1. In the left sidebar, click **Organization**.
2. Click the **On-prem agents** tab.
3. Click **Connect system**.
4. In the **Name** field, insert a connection name. This name appears when adding an application module to a Make.
5. In the **Choose app to connect** dropdown list, select the HTTP Agent app.
6. Define the base URL to the application API.

> The URL contains a host, a port, and a definition of API for a particular app. If the app runs on the same device where the On-prem agent is installed, the host is `localhost`. If the app runs on another device, the host is the device IP address. The port and the API definition vary based on the app.
>
> In this example URL `https://localhost:3000/api`
>
> - the host is `localhost`
> - the port is `3000`
> - the definition of API is `/api`

7 In the **URL** field, insert the URL to the application API.

8 Click **Save**.

In the connected systems list, you will see a new connection that works with this specific On-prem agent. Now you can start building a scenario.

> You might be asked to renew the access to the connected system after a certain period of time. If so, click **Revoke** in the **Connected systems** section.

Create a connection in the scenario

To create a **connection**, go to the Scenario Builder.

1 Click **Create a new scenario**.

2 Select the **HTTP Agent** from the dropdown list.

3 Select a module and connect it to Make.

4 Finishg configuring the module settings.

After adding the module, you can continue building your scenario. The module uses your selected On-prem agent.

> Currently, Make supports only **HTTP Agent** application for the On-prem agent usage.

On-prem agent status

The following table explains On-prem agent statuses.

Status	Description	Recommended next steps
Active	The On-prem agent is installed on your device and works under normal conditions. You can connect a system to it and use it in scenarios.	
Not responding	The On-prem agent is disconnected. Make checks its activity every four minutes. If the On-prem agent doesn't respond, it means the Internet connection is lost.	Check **On-prem agent troubleshooting**. When in Not responding status, associated scenarios still run, but the module using the On-prem agent shows a **500 error.**
Registered	The On-prem agent is created and appears in the **On-prem agents** section.	• Make sure to complete the installation of the On-prem agent. • Check if your firewall or anti-virus software is blocking the On-prem agent.
Stopped	The On-prem agent was stopped manually.	Make your On-prem agent active again if needed.

Troubleshooting

Here you can find solutions for the most common issues when working with the On-prem agent in Make.

Windows installer broke in the middle of the installation

If the installation process breaks, and the progress bar remains the same, do the following:

1. Check if you have these drives installed and updated:
   a. Visual Studio 2019
   b. AdoptOpenJDK 11
   c. Java 11 and above
2. Restart the installation process.

Error regarding wrong Java version when installing the On-prem agent for Windows

The following error appears if you have the wrong version of Java installed on your device:



1. Open the Command line.
2. Use the `echo %JAVA_HOME%` command to make sure Java is in the Windows PATH variable.
3. Use the `java -version` command to make sure Java 11 or above is installed.

No permissions to run the Windows installer

Right-click the installer and click **Run as Administrator**.

On-prem agent in Not responding status in Make

Check the following:

- The device where the On-prem agent is installed is operational.
- The device is connected to the Internet.
- The Internet connection is stable.
- Your firewall and anti-virus software don't block the On-prem agent.
- Restart the On-prem agent on your device (Windows only):

On-prem agent not running on a Windows machine

If you open the **Services** app on Windows and see that the **Make Agent** app doesn't have the **Running** status, you can do the following:

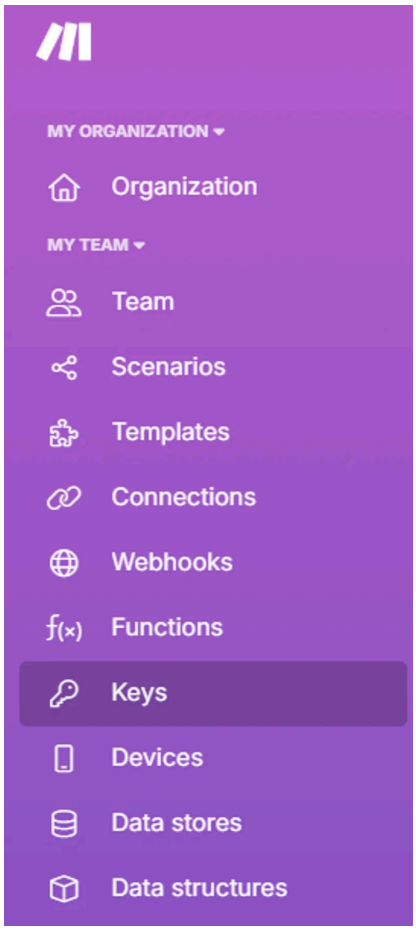
1. Go to the folder where you saved the On-prem agent on your machine.
2. Open the **application-local** file and check your credentials: Client ID, Secret, and Base URL.
3. Open the **logs** folder and locate the problem.

3.2.3. Keys

The keychain in Make will help you to administer public and private keys. Keys are used, for example, by the **Encryptor** app for encrypting or decrypting PGP messages.

If a module requires a public or private key, you can add it to the keychain in the **module settings**.

To administer all created keys, click **Keys** in the left sidebar.

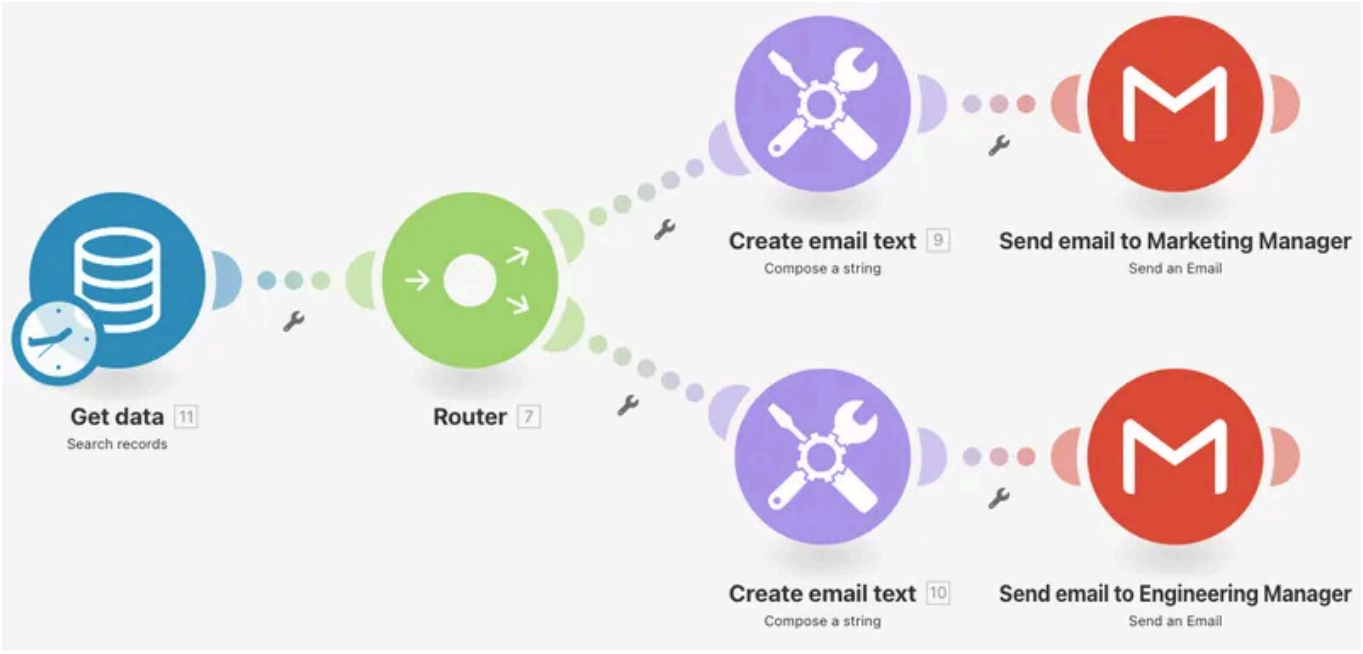


3.2.4. Dynamic connections

> This feature is available to Enterprise customers.

A **dynamic connection** is a variable that contains a connection.

Imagine that you have a scenario that automatically sends emails to multiple email addresses. Your team doesn't have a team email so you want your colleagues to use their work emails when they run the scenario:



You can set a dynamic connection in both Gmail modules. That way, your teammates can use their own Gmail connections to send the emails when they run the scenario.

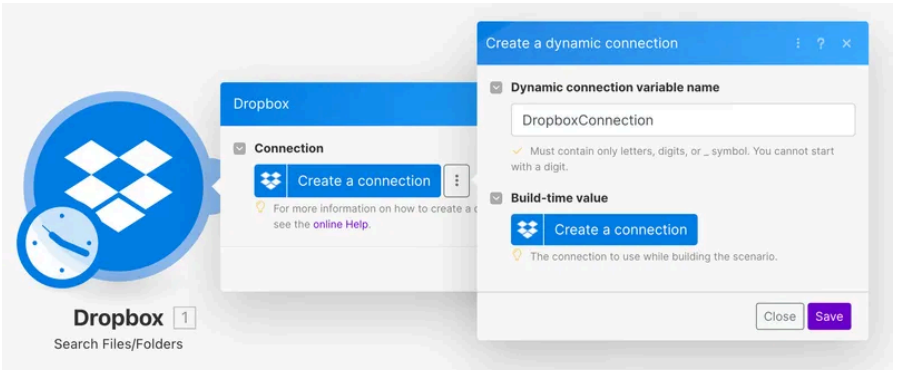
Create a scenario with a dynamic connection

Almost every module requires a connection to work properly. To create a scenario with a dynamic connection, you have to assign a connection to build the scenario. Make calls the initial connection as the "build-time value."

When you finish the scenario, you can assign a different connection to the dynamic connection anytime.

1 Go to your scenario and select an app module.

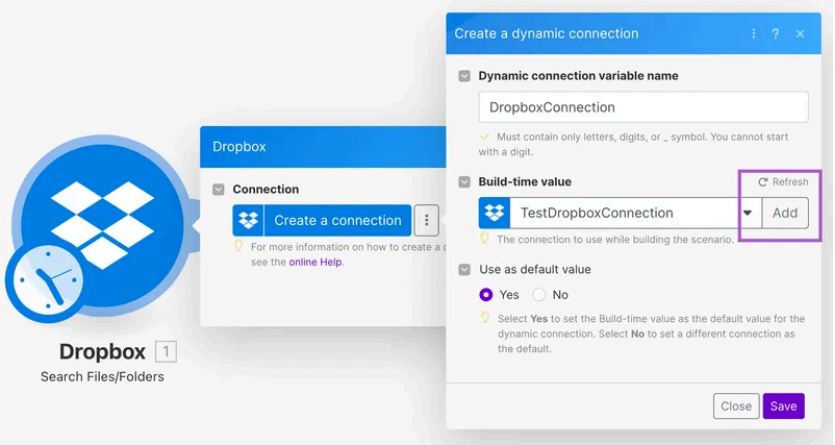
2 Click the ellipsis button, and then click **Create a dynamic connection**.



3 In the **Dynamic connection name** field, type the name of the connection.

4 Set the **Build-time value** for the dynamic connection:

1. In the **Build-time value**, choose a connection to use when building the scenario or click **Add** to create a new connection.

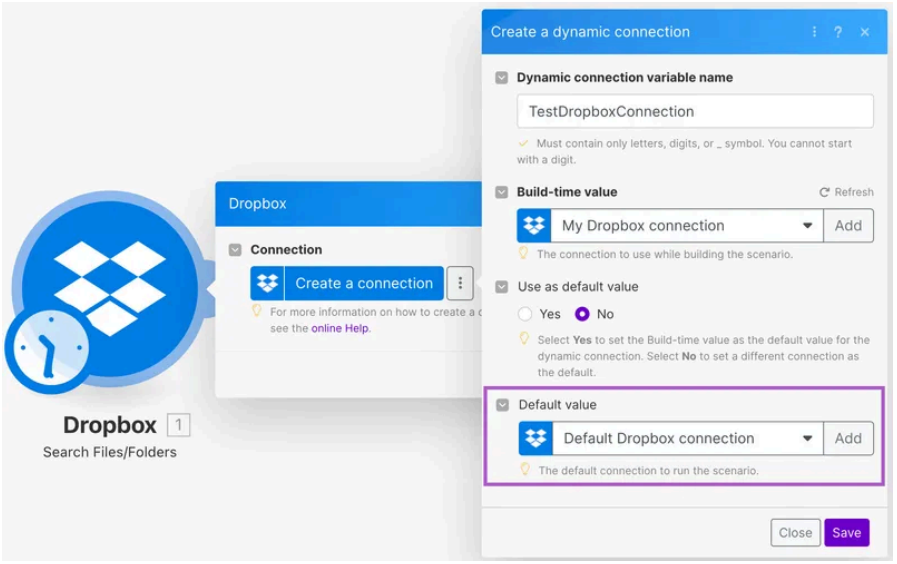


2. In the **Connection name** field, enter a name for the connection.
3. Click **Save**. Dropbox prompts you to grant Make access to your data.
4. Grant Make access to your Dropbox account.
5. Click **Save**.

You have set the build-time value for the dynamic connection. You can see the connection in the **Build-time value** field.

5 Select whether you want to use the build-time connection as the default or not. Make uses the default value to run the scenario. You can select a different connection when you run the scenario **on-demand**.

6 (Optional): You can select the dynamic connection default value in the **Default value** selection box or create a new connection with the **Add** button.

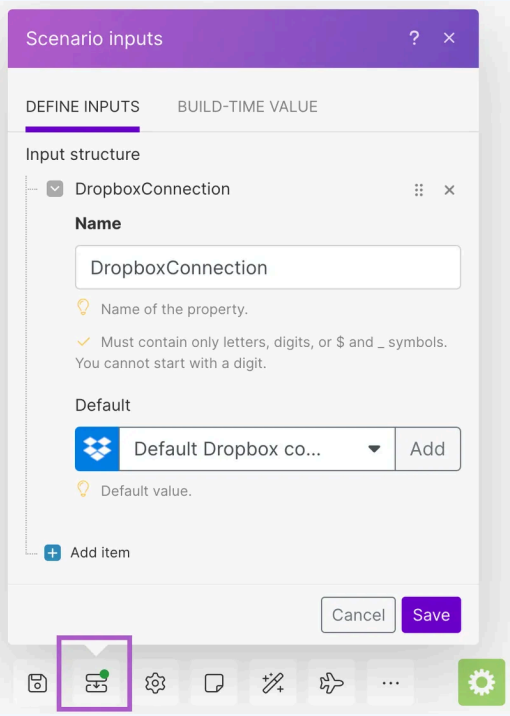


> If you don't provide a default for the dynamic connection, you can schedule the scenario to **On-demand** only.

7 Click **Save**.

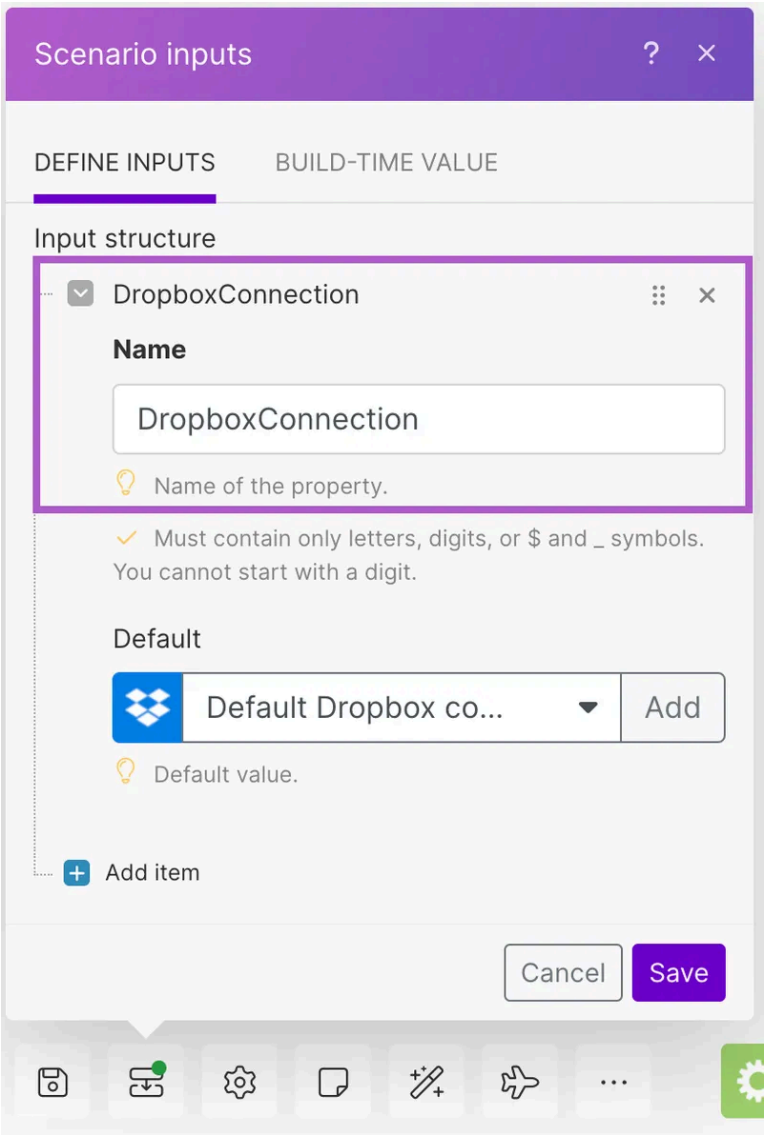
You created a dynamic connection. Now you can continue configuring the module.

You can set the default for the dynamic connection in the scenario inputs interface as well:

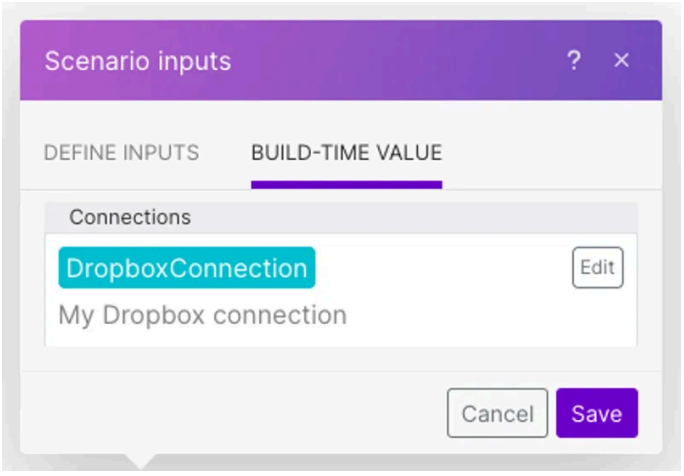


You can find the list of dynamic connections in:

Module settings | **Scenario inputs window**



In the **Build-time value** tab, the dynamic connection with the connection its using:



Run a scenario with a dynamic connection

A dynamic connection is like a variable or a container for connections, which is empty until you assign a connection to it. To run a scenario with a dynamic connection, you have to assign a connection to it or use a default. You can either create a new dynamic connection or use an existing one.

Use an existing dynamic connection

To run a scenario with a dynamic connection, use the scenario inputs. In scenario inputs, assign a connection to the dynamic connection for your scenario run or use the default.

> To run your scenario on schedule, you have to provide a default for your dynamic connections. Make uses the default connection in scheduled scenario runs.

1. Go to your scenario.
2. Click **Run once**. A scenario inputs window appears.
3. You can either set up a new connection or use an existing connection.
4. Click **OK**.

Make runs the scenario with the selected connections.

Update a dynamic connection

Compared to other connections, dynamic connections are flexible and can change the value they contain. For example, you might want to assign a different connection to the dynamic connection with each scenario run. This might be practical when you want to frequently use different credentials in the scenario.

Another option is to update just the default for the dynamic connection for scheduled scenario runs.

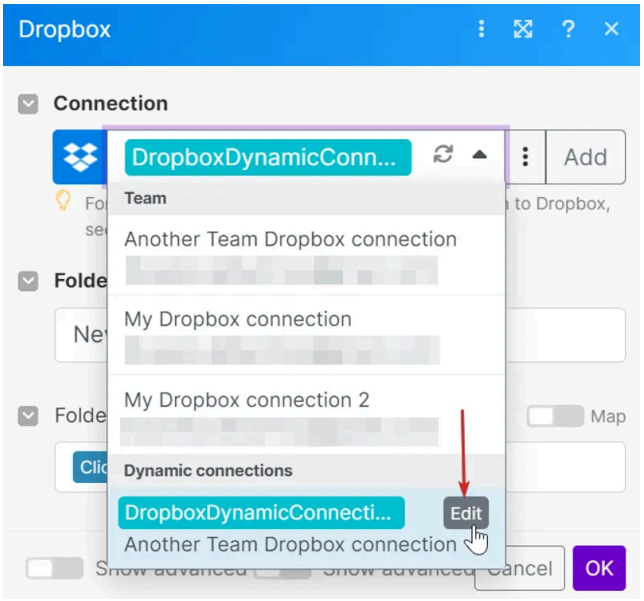
You can update a dynamic connection:

- in the app module
- in the scenario inputs

Update dynamic connections in the app module

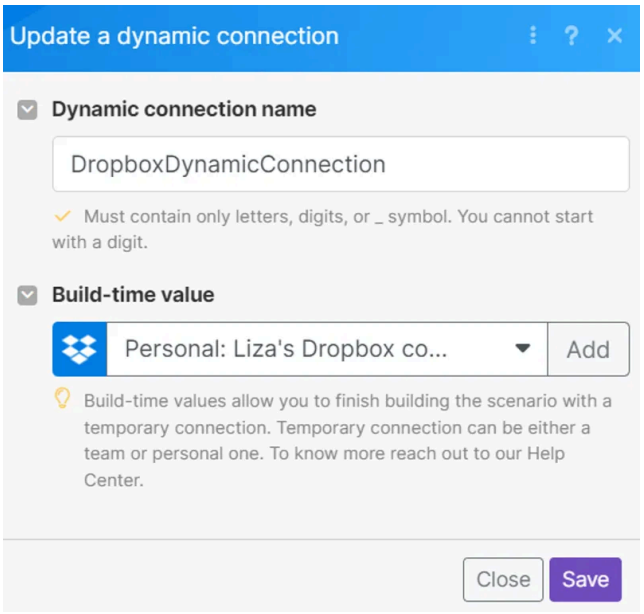
1. Open the **Connections** dropdown list in the app module and find the dynamic connection you want to edit.

2. Click **Edit**.



3. Optional: You can change the dynamic connection name as well.

4. In the **Build-time value** dropdown list, select the connection you want to assign to the dynamic connection. Click **Add**, if you want to create a new connection.



5 Click **Save**.

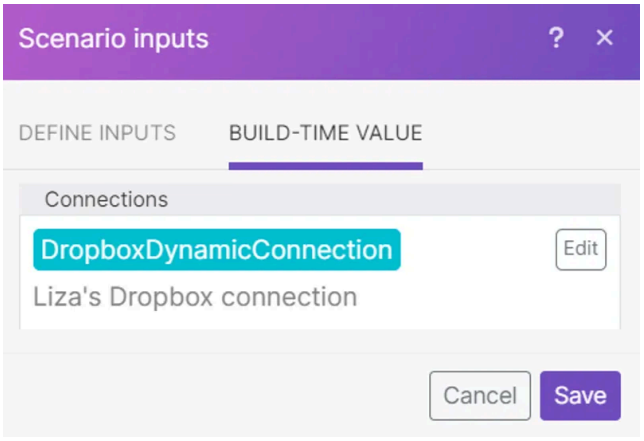
You have updated the dynamic connection. When the scenario runs, the dynamic connection will use the connection you assigned.

Update dynamic connections in the scenario inputs

1 Go to the scenario in which you want to update the dynamic connection.

2 Click **scenario inputs**.

3 Go to the **Build-time value** tab.



4. Click **Edit**.

5. In the dropdown list, select another connection you want to assign to the dynamic connection. Click **Add** if you want to create a new connection.

6. Click **Save**.

You have updated the dynamic connection. When the scenario runs, the dynamic connection will use the connection you assigned.

3.2.5. Certificates and keys

There are two ways of inserting a key or certificate.

1. **Direct insert**
2. **Extract from the file** (P12, PFX or PEM)

Direct insert

You need the file that contains **RSA PRIVATE KEY**.

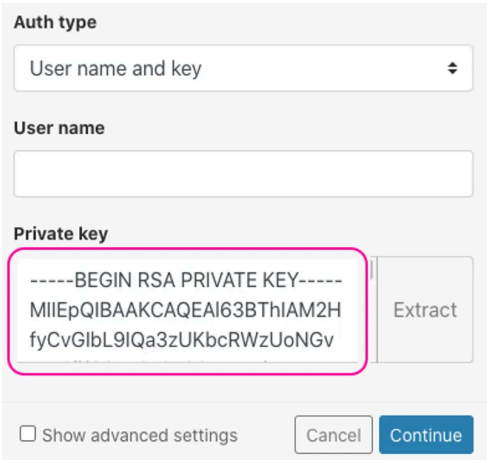
1. Copy your key from the file. It should begin and end like this:

   -----BEGIN RSA PRIVATE KEY-----

   .....

   -----END RSA PRIVATE KEY-----

2. Paste the key directly in the following field:



> OPENSSH PRIVATE KEY is not supported. This has to be converted using the following command in Terminal: `ssh-keygen -p -m PEM -f <pathToTheKey>`.
> For Windows, you can use the PuTTy key generator.

Extract from the file

You can extract a Private Key or Certificate using our key extraction function. This function also allows you to extract the key from encrypted files.

The supported file formats are **P12**, **PFX**, and **PEM**.

You can extract either the private key or the certificate. Select the type of information you want to extract from the file in the **Extract** field.

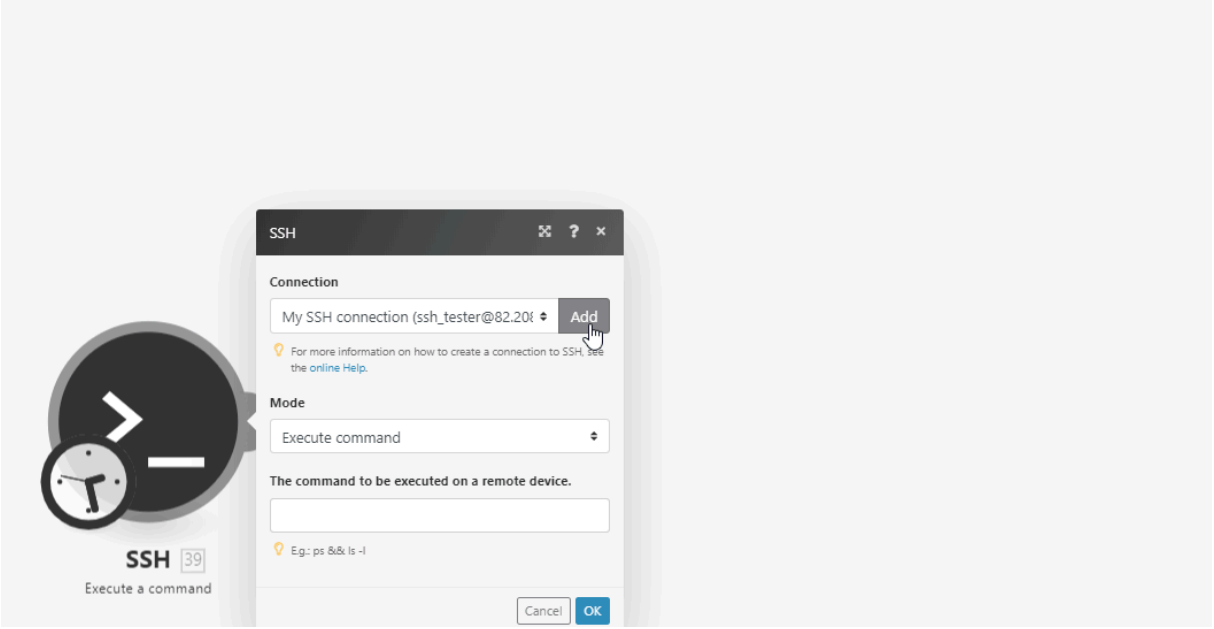
For example, when you create a Linux Based EC2 instance on AWS, you receive the login credentials in the PEM file:



The file contains the private key that is used to connect to the instance.

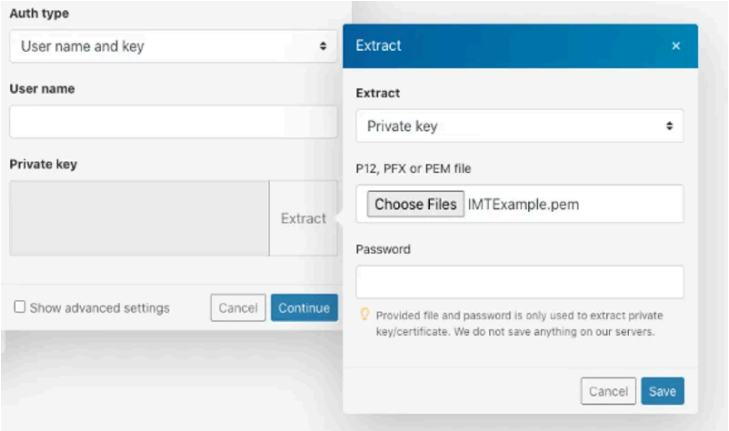


1. Go to the module's **Create a connection** dialog and click the **Extract** button in **Advanced Settings**.



2 In the **Extract** dialog, select the **Private Key** option to extract the private key from the provided file.

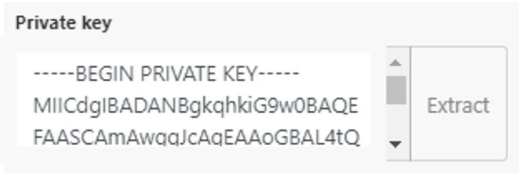
3 Click the **Choose Files** button, browse and **Open** the desired file.



4 Enter the password to the respective field, if needed.

5 Click **Save**.

The private key has been exported from your file and will be used to connect to the desired service.


Private key
-----BEGIN PRIVATE KEY-----
MIICdgIBADANBgkqhkiG9w0BAQE
FAASCAmAwggJcAgEAAoGBAL4tQ
Extract

3.2.6. Allow connections to and from Make IP addresses

If you need Make to connect to servers that are behind a firewall or that allow connections only from specific IP addresses, white-list the following IP addresses:

Zone	Egress IPs (outbound from Make)
us2.make.com	44.209.150.16 44.210.162.163 35.170.163.230
us1.make.com	54.209.79.175 54.80.47.193 54.161.178.114
eu2.make.com	34.254.1.9 52.31.156.93 52.50.32.186
eu1.make.com	54.75.157.176 54.78.149.203 52.18.144.195
us1.make.celonis.com	44.196.246.20 3.94.51.90 52.4.48.212
eu1.make.celonis.com	3.125.27.86 3.68.125.41 18.193.24.45

> Make uses dynamic **Ingress IPs**.

3.2.7. Connect to Google services using a custom OAuth client

> This article is currently in progress. We are actively working to improve the writing style and visual elements, while making sure it reflects the most up-to-date information.

This article will show you how to create your own project in Google Cloud Platform and a custom OAuth client. This is useful for connecting restricted Google services, like Google Drive or Gmail, to Make.

To connect, you must have a Google account. You can create an account at **accounts.google.com/signin**.

The following procedure is intended for:

- **Personal use** (*@gmail* and *@googlemail.com* users)
- **Internal use** (Google Workspace users that prefer to use a custom OAuth client)

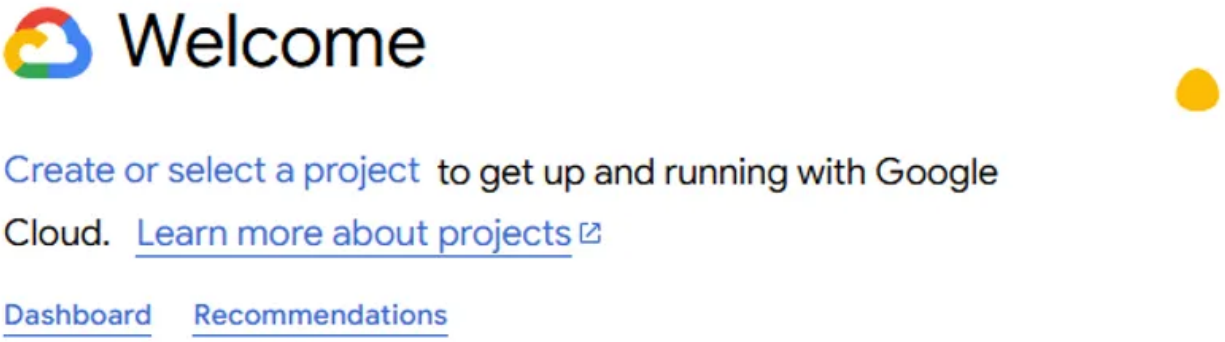
Configure a Google Cloud Platform project

There are five steps to configure a Google Cloud Platform project.

1. **Create a Google Cloud Platform project**
2. **Enable APIs**
3. **Configure your OAuth consent screen**
4. **Create your client credentials**
5. **Connect to Make with custom credentials**

Create a Google Cloud Platform project

1. Log in to the **Google Cloud Platform** using your Google credentials.
2. On the welcome page, click **Create or select a project** > **New project**.



3. Enter a **Project name** and select the **Location** for your project.
4. Click **Create**.
5. In the top menu, check if your new project is selected in the **Select a project** dropdown. If not, select the project you just created.

> To create a new project or work in the existing one, you need to have the `serviceusage.services.enable` permission. If you don't have this permission, ask the Google Cloud Platform Project Owner or Project IAM Admin to grant it to you.

Enable APIs

1. Open the left navigation menu and go to **APIs & Services** > **Library**.
2. Search for the APIs you need for your project.
   For information regarding specific APIs, refer to the **Make Help Center** for the app you are using.
3. Click the relevant API, then click **Enable**.

Configure your OAuth consent screen

1. In the left sidebar, click **Google Auth Platform**.
2. Click **Get Started**.
3. In the **Overview** section, under **App information**, enter *Make* as the app name and provide your Gmail address. Click **Next**.
4. Under **Audience**, select **External**.
   For more information regarding user types, refer to **Google's Exceptions to verification requirements documentation**.
5. Under **Contact Information**, enter your Gmail address..

6. Under **Finish**, agree to the Google User Data Policy.
7. Click **Continue** > **Create**.
8. In the **Branding** section, under **Authorized domains**, add `make.com` and `integromat.com`. Click **Save**.
9. Optional: In the **Audience** section, add your Gmail address on the **Test users** page, then click **Save and continue** if you want the project to remain in the **Testing** publishing status.
10. In the **Data Access** section, click **Add or remove scopes**, add any desired scopes, and click **Update**.

    Here are some examples of scopes you may need for Gmail and Google Drive. For additional scopes, refer to the **Make Help Center** for the app you are using.
11. Click **Save**.

- `https://mail.google.com`
- `https://www.googleapis.com/auth/userinfo.email`
- `https://www.googleapis.com/auth/drive`
- `https://www.googleapis.com/auth/drive.readonly`

Publishing Status

Testing: If you keep your project in the **Testing** status, you will be required to reauthorize your connection in Make every week. To avoid weekly reauthorization, update the project status to **In production**.

In production: If you update your project to the **In production** status, you will not be required to reauthorize the connection weekly. To update your project's status, go to the **Google Auth Platform**, the **Audience** section, and click **Publish app**. If you see the notice **Needs verification**, you can choose whether to go through the **Google verification process** for the app or to connect to your unverified app. Currently connecting to unverified apps works in Make, but we cannot guarantee the Google will allow connections to unverified apps for an indefinite period.

For more information regarding the publishing status, refer to the Publishing status section of **Google's Setting up your OAuth consent screen help**.

Create your client credentials

1. In Google Auth Platform, click **Clients**.
2. Click **+ Create Client**.
3. In the **Application type** dropdown, select **Web application**.
4. Update the **Name** of your OAuth client. This will help you identify it in the console.
5. In the **Authorized redirect URIs** section, click **+ Add URI** and enter the redirect URI.

   Here is a redirect URI you need for Gmail and Google Drive:
   `https://www.integromat.com/oauth/cb/google-restricted`
   For additional redirect URIs, refer to the **Make Help Center** for the app you are using.
6. Click **Create**.
7. Click the OAuth 2.0 Client you created, copy your **Client ID** and **Client secret** values, and store them in a safe place.

You will use these values in the **Client ID** and **Client Secret** fields in Make.

Establish the connection in Make

To establish the connection in Make:

1. Log in to your Make account, add a Google app module to your Make, and click **Create a connection**.
2. Optional: In the **Connection name** field, enter a name for the connection.
3. Switch on the **Show advanced settings** toggle and enter your Google Cloud Platform project client credentials.
4. Click **Sign in with Google**.
5. If prompted, authenticate your account and confirm access.

You have successfully established the connection. You can now edit your Make and add more modules.

Common problems

100 Logins Limit Per Day Has Been Reached

This happens rarely, but when it does, we recommend creating another OAuth client.

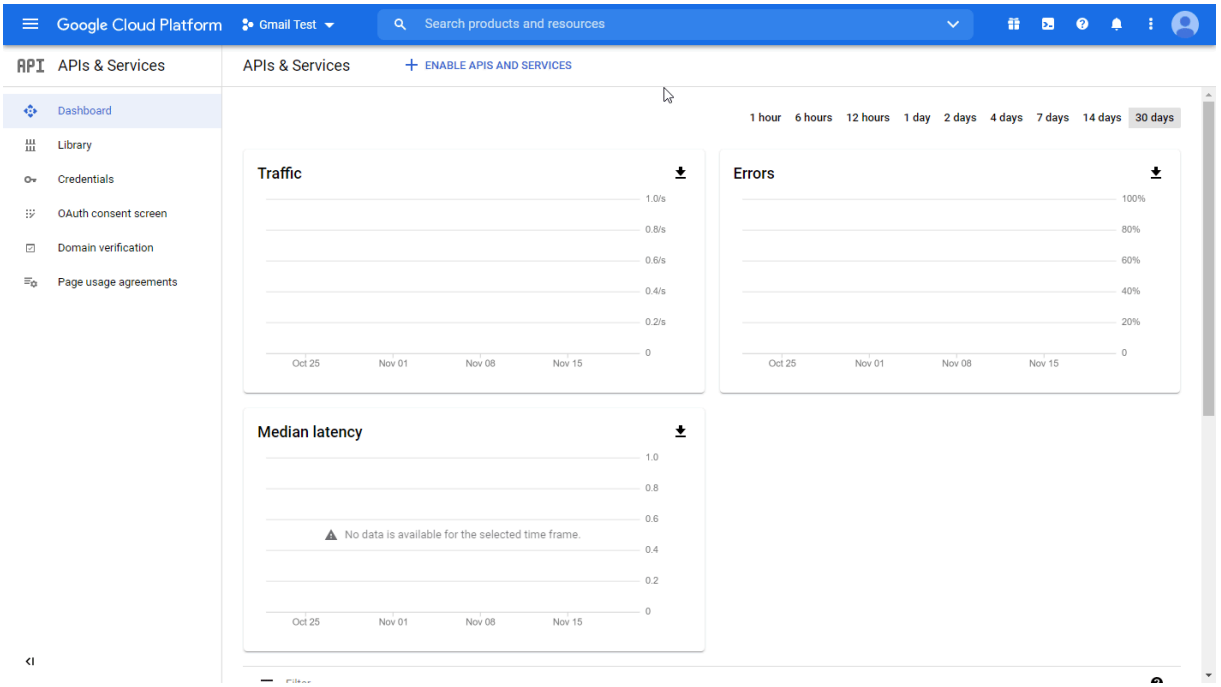
[403] Access Not Configured

If this error message appears, you need to enable the corresponding API in your Google Cloud Platform.

Authorization Error - Error 403: access_denied

Google has added the required settings for the Consent screen. You'll need to add the email address associated with the Google account you want to connect with Make as a Test user.

1. Sign in to the **Google Cloud Platform** using your Google credentials.
2. Go to **APIs & Services > OAuth consent screen**.
3. In the **Test Users** section, click **Add users** to add a test user. Enter the email address associated with the Google account you want to connect with Make, and click **Save**.



4. Now, go to **Make,** and connect to the desired Google service.

Failed to verify connection 'My Google Restricted connection'. Status Code Error: 400

Your connection has expired and is no longer valid. You need to reauthorize the connection.

This error affects **non**-Google Workspace accounts. For more details please refer to the **Google OAuth documentation**.

Due to Google's updated security policy, unpublished apps can only have a 7-day authorization period. After the OAuth security token expires, the connection is no longer authorized and any module relying on it will fail.

Solutions:

Option 1:

To avoid weekly reauthorization, you can update the publishing status of your project.

If you update your project to the **In production** status, you will not be required to reauthorize the connection weekly.

Change the status of your project by following these steps:

1. Log in to the Google Cloud console.
2. Navigate to the **Oauth consent screen**.
3. Click the **Publish app** button next to your app.
4. If you see the notice **Needs verification**, you can choose whether to go through the **Google verification process** for the app or to connect to your unverified app. Currently connecting to unverified apps works in Make, but we cannot guarantee the Google will allow connections to unverified apps for an indefinite period.

For more information regarding publishing statuses, refer to the **Publishing status** section of **Google's Setting up your OAuth consent screen help** and our **Community page**.

Option 2

If you keep your project in the **Testing** status, you will be required to reauthorize your connection in Make every week.

Reauthorize your Google connection by following these steps:

1. Log in to Make.
2. Go to **Connections**.
3. Find your Google connection and click **Reauthorize** button.

3.2.8. Call Google APIs with the 'HTTP Make a OAuth 2.0 request' module

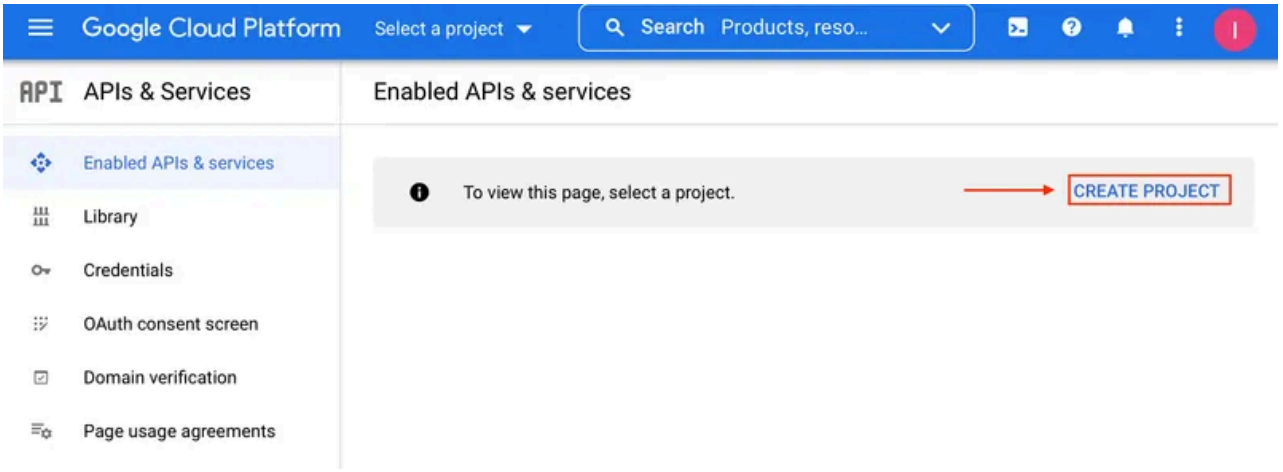
> This article is currently in progress. We are actively working to improve the writing style and visual elements, while making sure it reflects the most up-to-date information.

This is a step-by-step guide on calling **Google APIs** like:

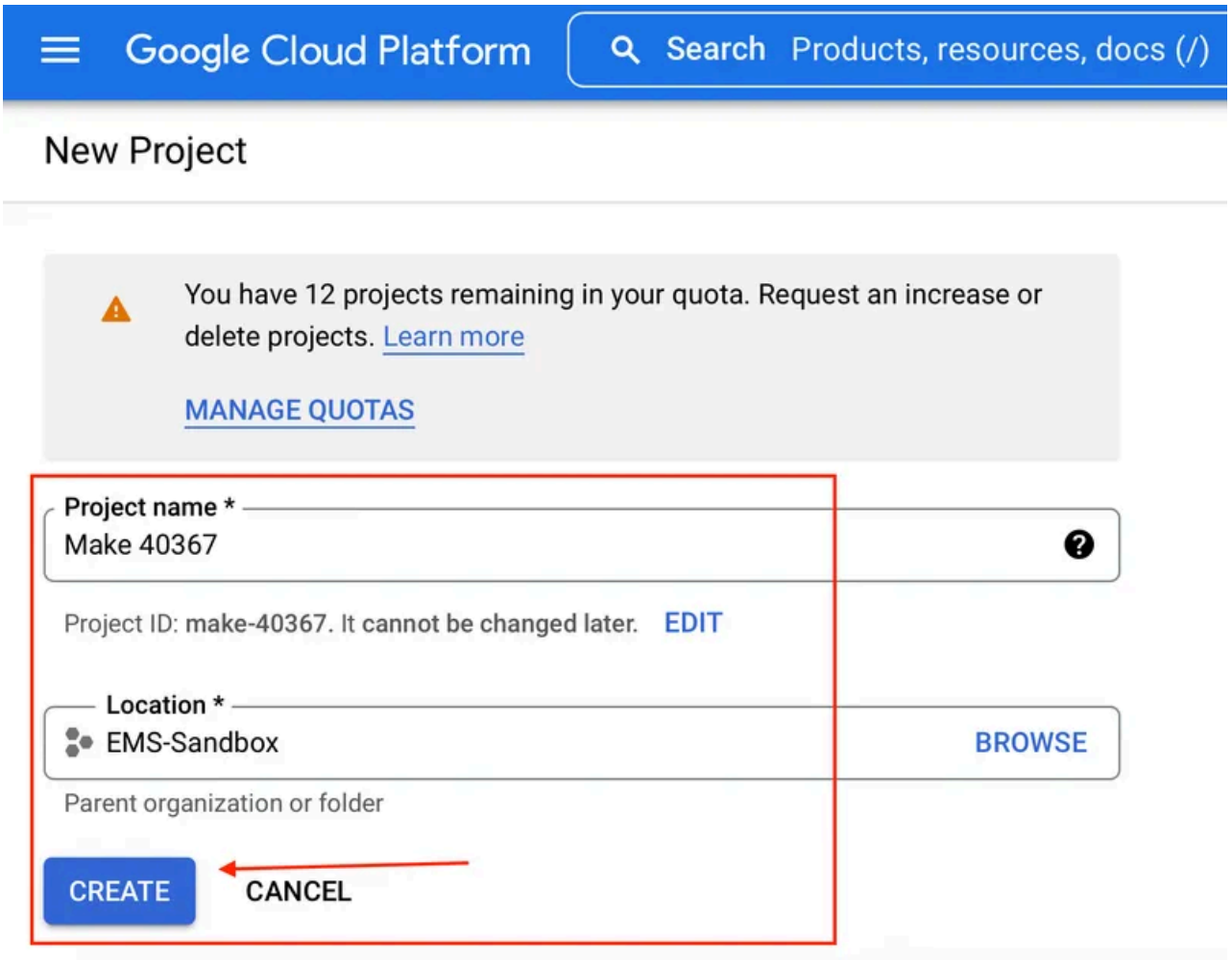
- **Google Drive API**
- **Google Tag Manager API**
- **Google Docs API**
- **Google Ads API Beta**
- **Google My Business API**
- **Google Cloud Pub/Sub API**
- **Google Search Console API**
- **G Suite Groups Settings API**
- **G Suite Directory API**
- **BigQuery API**
- **Google Cloud Natural Language API**
- **G Suite Reseller API**
- **Google Sites**
- **Google Apps Script**
- **Gmail API**
- and many more

For more information on required APIs, scopes, and connection details, please see our documentation for the Google service you are using. If we don't have a Make app available for the Google service you want to connect to, refer to that service's API documentation for information regarding required APIs and scopes.

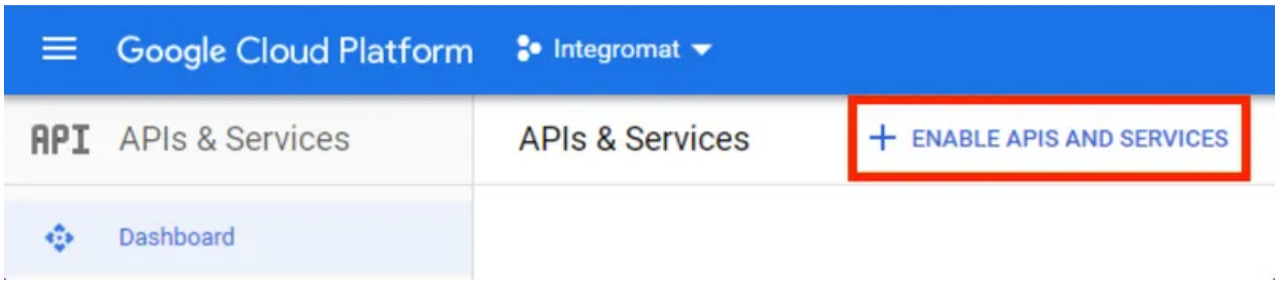
1. Open the **Google Cloud Platform APIs & Services dashboard** and login with your Google account.
2. Click **CREATE** to create a project.



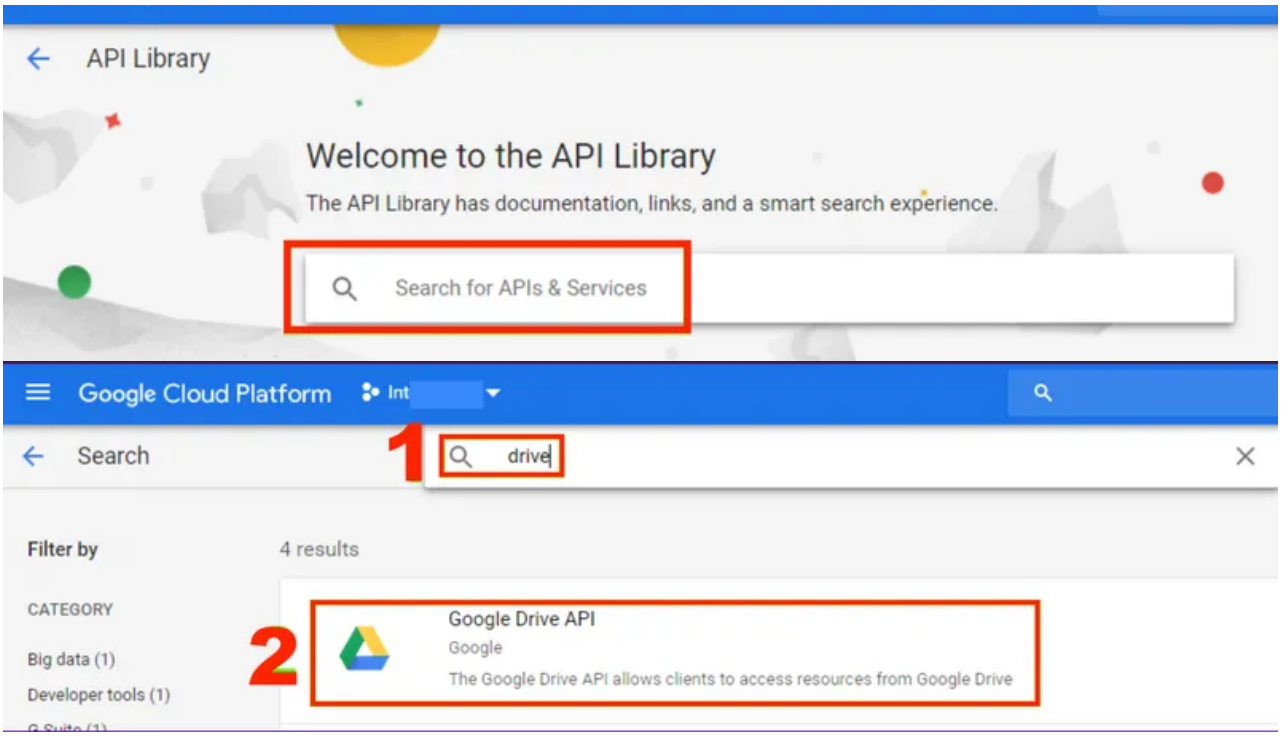
Enter **Project name** and choose **Location**. Click **CREATE**.



3. Click **ENABLE APIS AND SERVICES**.

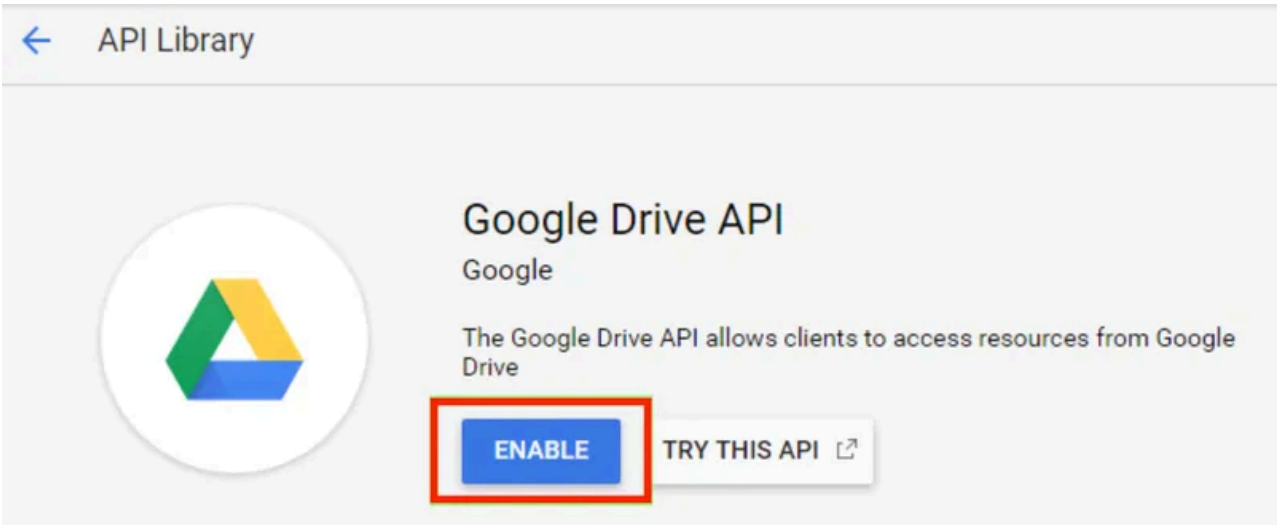


4. Type the name of the API you want to call in the **Search for API & Services** field and click on it.



search_for_api_and_services_.png

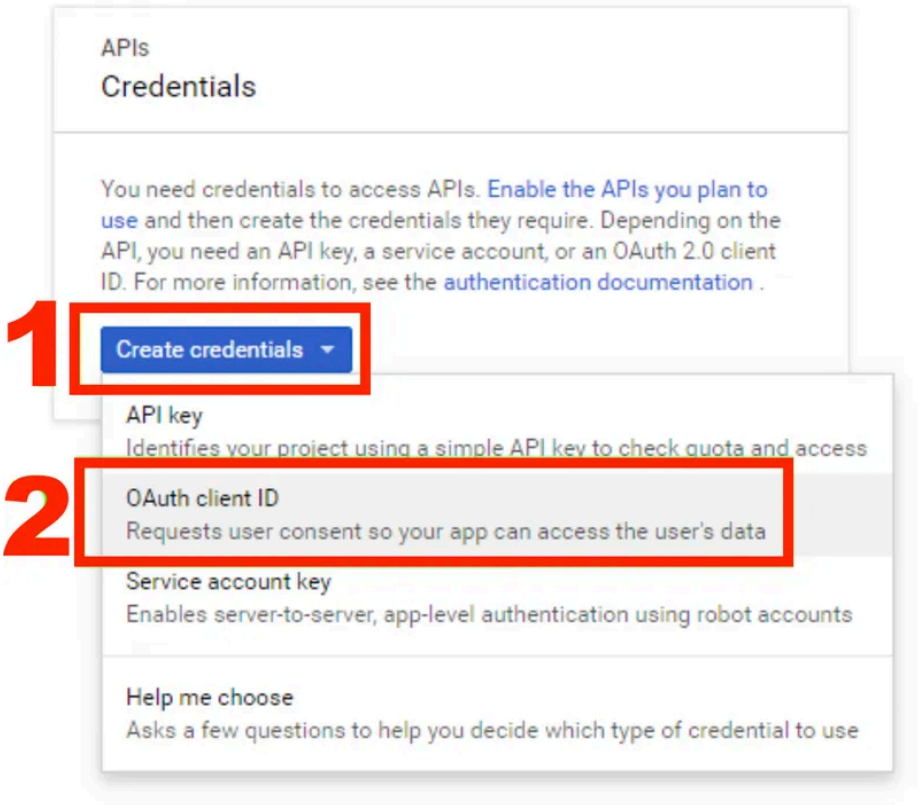
5. Click **ENABLE** to enable the API.



6. Open the **Google Cloud Platform APIs & Services Credentials OAuth consent screen tab.**

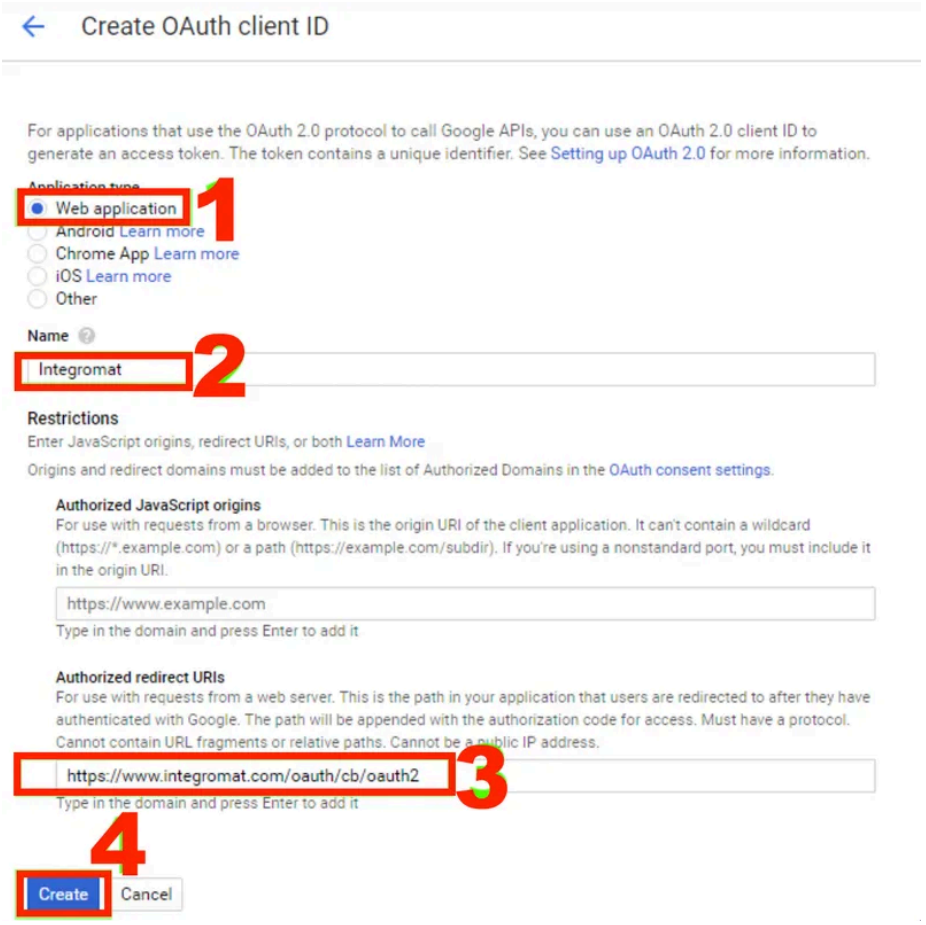
7. Fill out the form as follows and click **Save** to save the settings:

8. Open the **Google Cloud Platform APIs & Services Credentials**.
9. Click **Create credentials** dropdown and choose the **OAuth client ID** option from the menu.

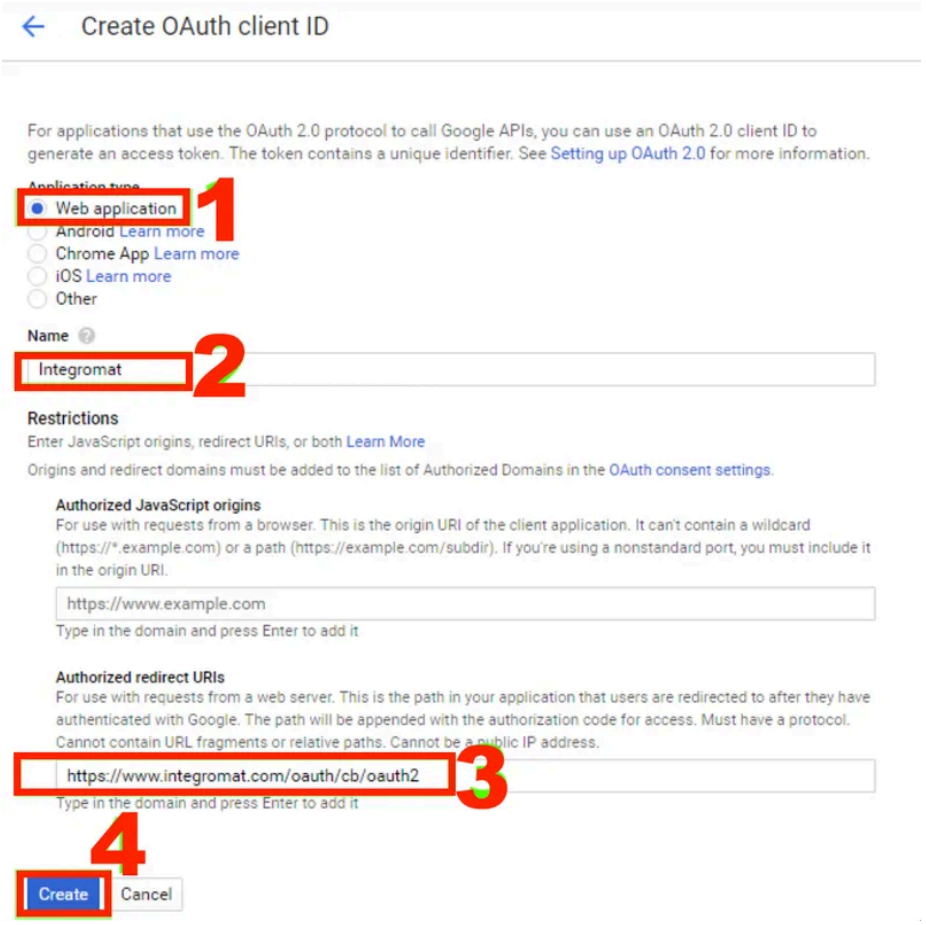


10. Fill out the form as below and click **Create** to create the app.

    Notice that the redirect URI starts with `https://www.integromat.com` instead of `https://www.make.com`. This is currently a known issue in Make.
    Make was formerly called Integromat, which means you can trust this URL as much as any Make URL.
    Please make sure all your OAuth redirect URIs point to `https://www.integromat.com/oauth/cb/oauth2/`.

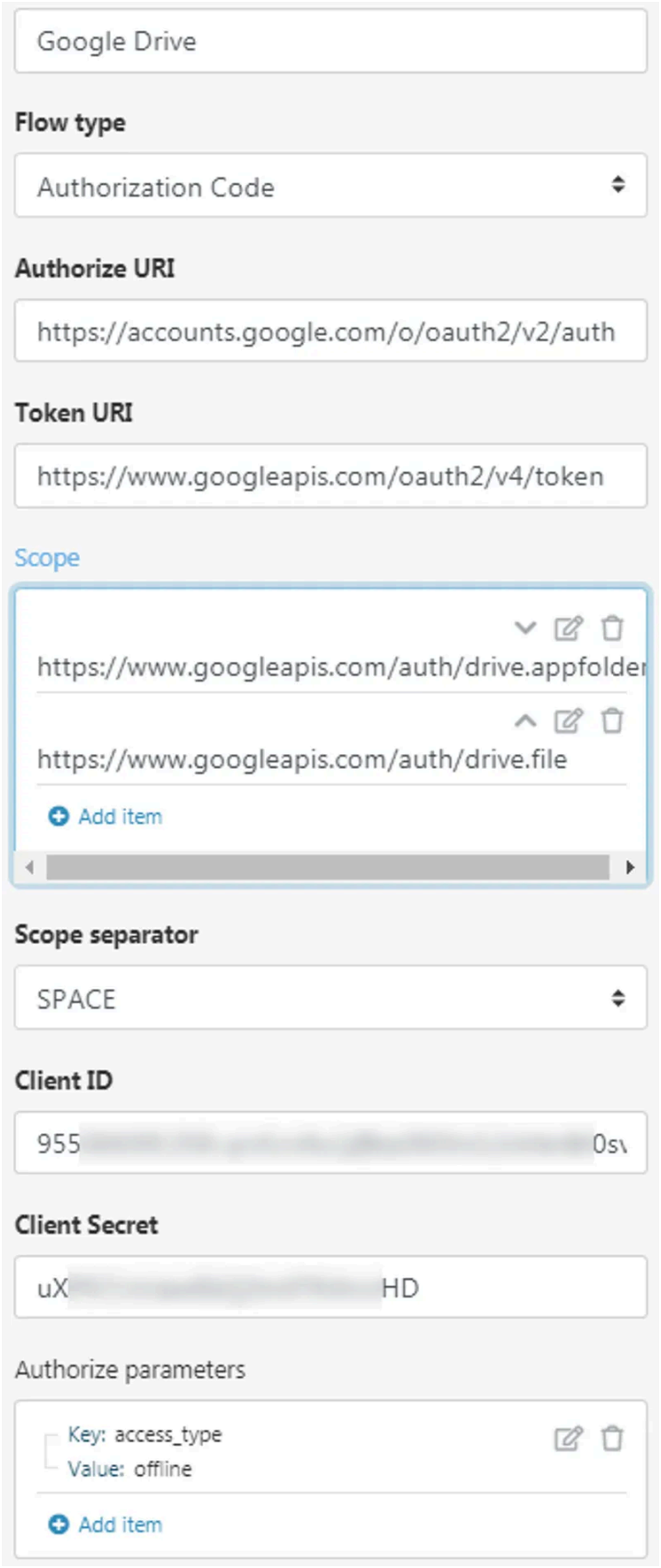


11. A dialog containing app's Client ID and Client Secret pops up.



12. In Make, create a new scenario.

13. Insert the **HTTP > Make an OAuth 2.0** request module.
14. Open the module's configuration and click **Add** button next to the **Connection** field to create a new connection.
15. Tick the **Show advance settings** option.
16. Fill the connection setup as follows:
17. The connection setup should look like this:



18. Click **Continue**.
19. If a popup titled "This app isn't verified" appears, click on the "Advanced" link and then on the "Go to Make (unsafe)" link.
20. A window pops up. Review the permissions and click **Allow**.
    The connection is successfully created.

21. Configure the module to make an API call.

3.2.9. Connect Make to any web service that uses OAuth2 authorization

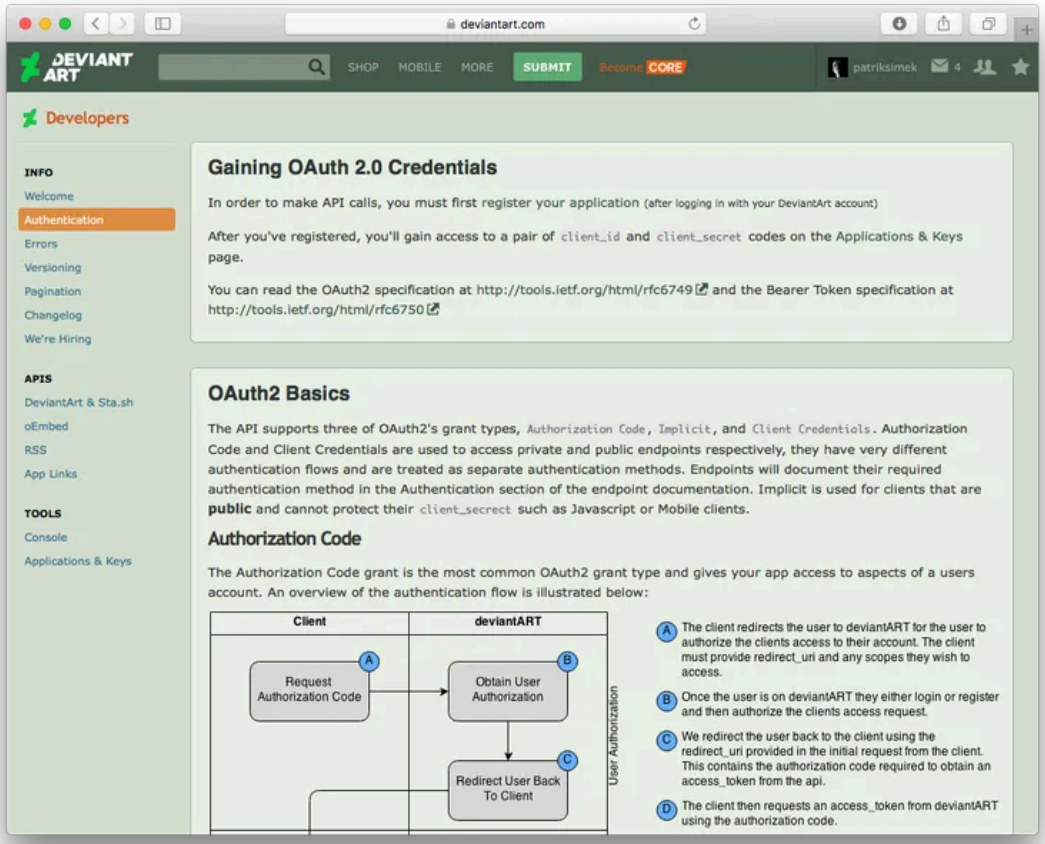
> This article is currently in progress. We are actively working to improve the writing style and visual elements, while making sure it reflects the most up-to-date information.

In this article, you will learn how to integrate the DeviantArt service, one of the largest networks for creative people. Our goal is to get a list of the latest deviations. Before configuring this scenario, you have to find out some basic information about the service.

Firstly, go through the Development section and find out what method DeviantArt uses for user authentication, and which API is appropriate to perform the desired function.

Make supports the following authorization types:

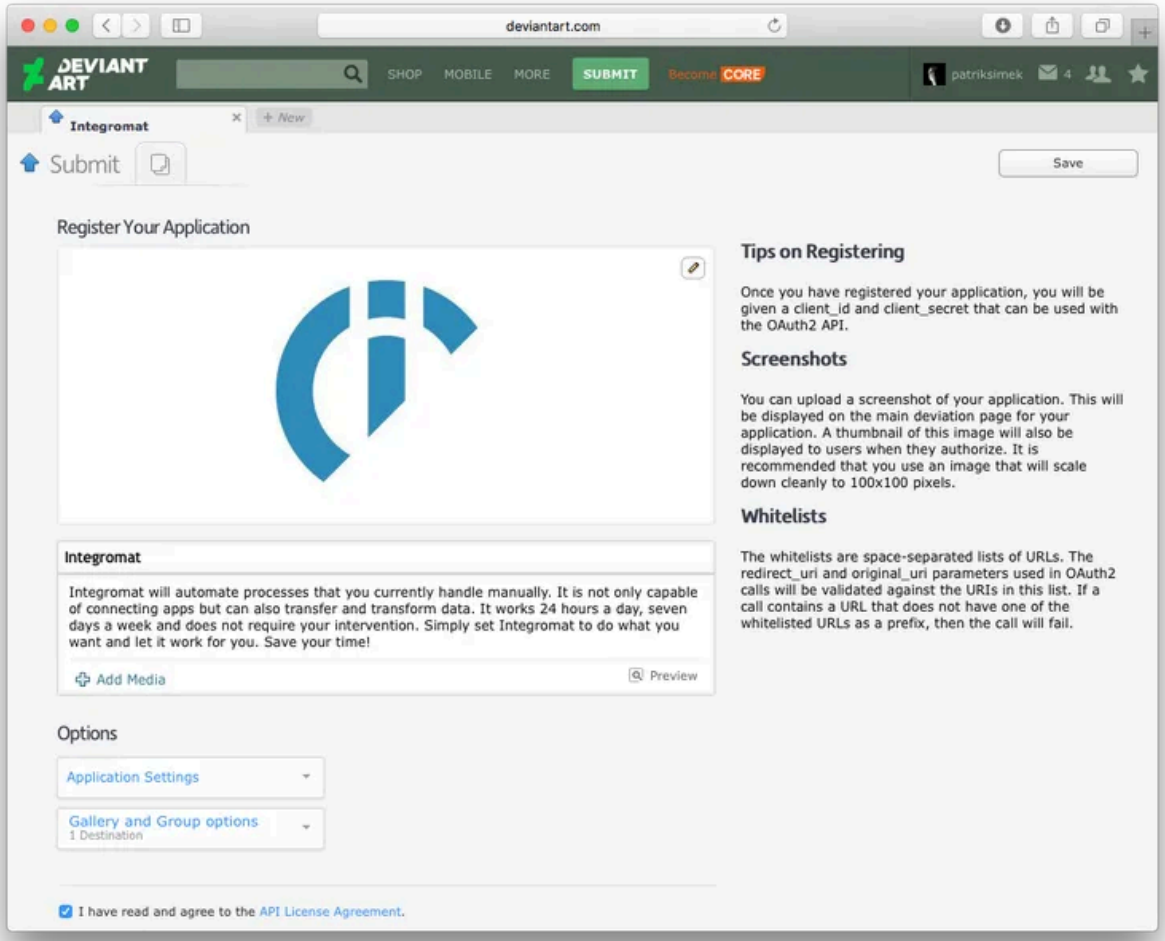
- OAuth2
- Basic Authorization
- API key



Getting ready

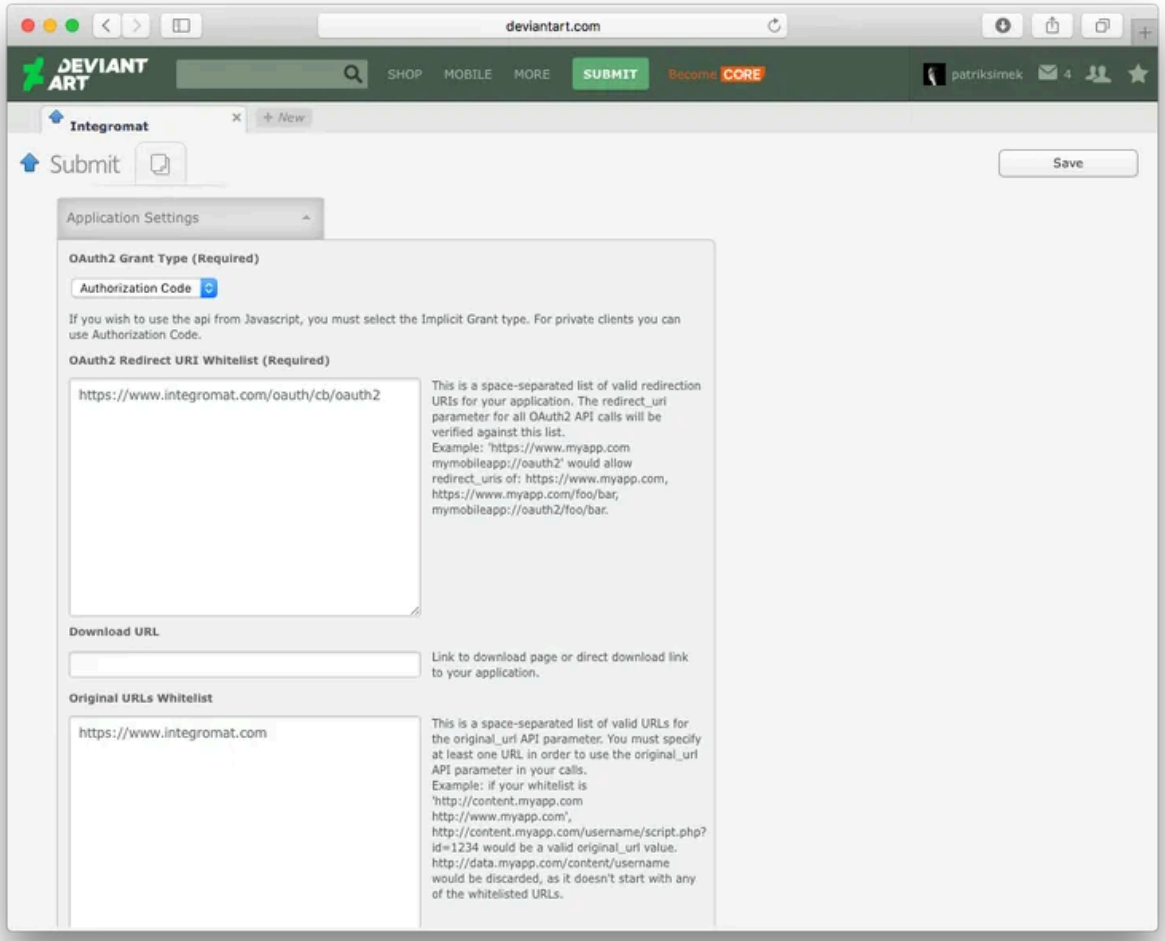
DeviantArt uses a standardized OAuth2 protocol for authorization. You can see that in the *Authentication* section.

1. Create a so-called client to be able to communicate with the API. Click **Register your application** link to create a client.
2. A form opens up that allows you to create a new application. Fill in the name and description of the application and upload the icon.

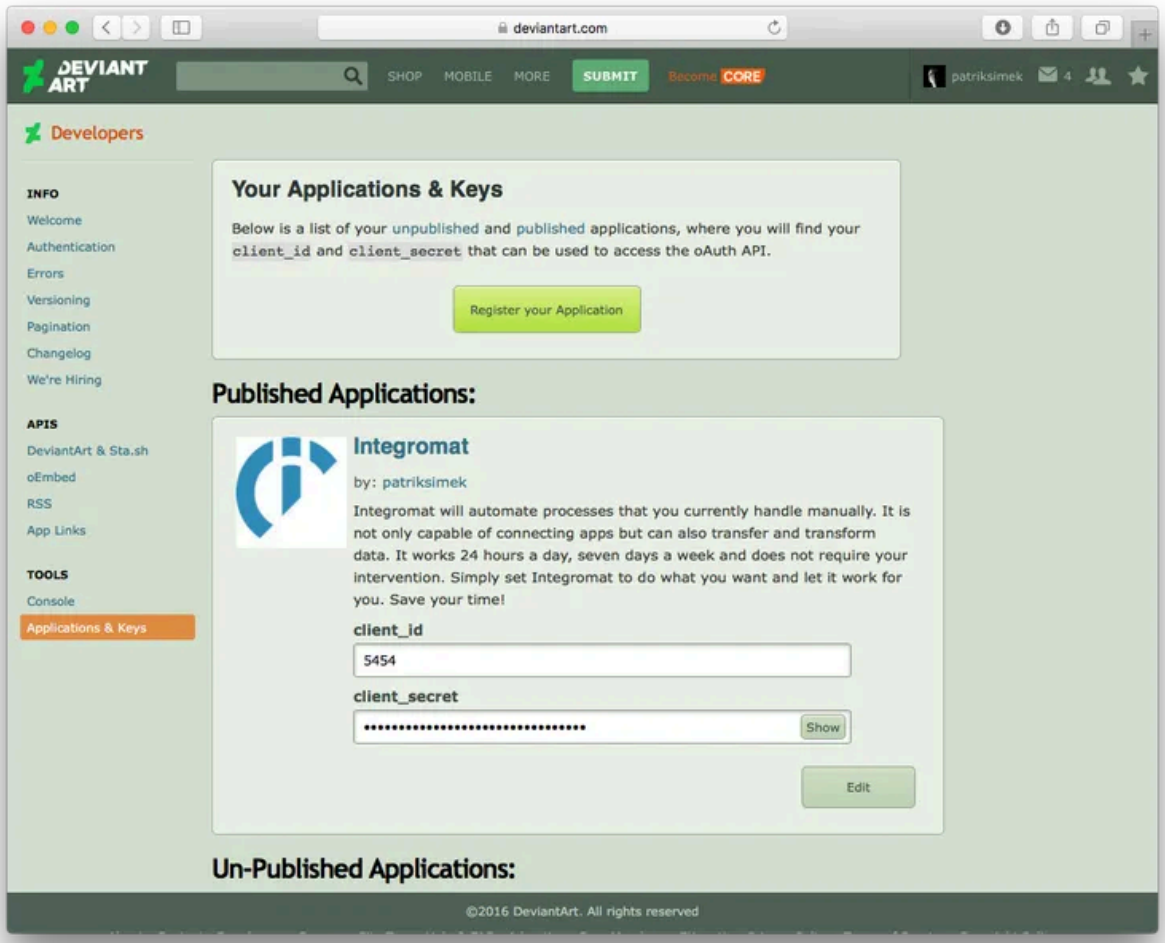


3. In *Application settings* section, fill out the *OAuth2 Redirect URI Whitelist* field. All services that use standard OAuth2 protocol for authorization require this parameter.

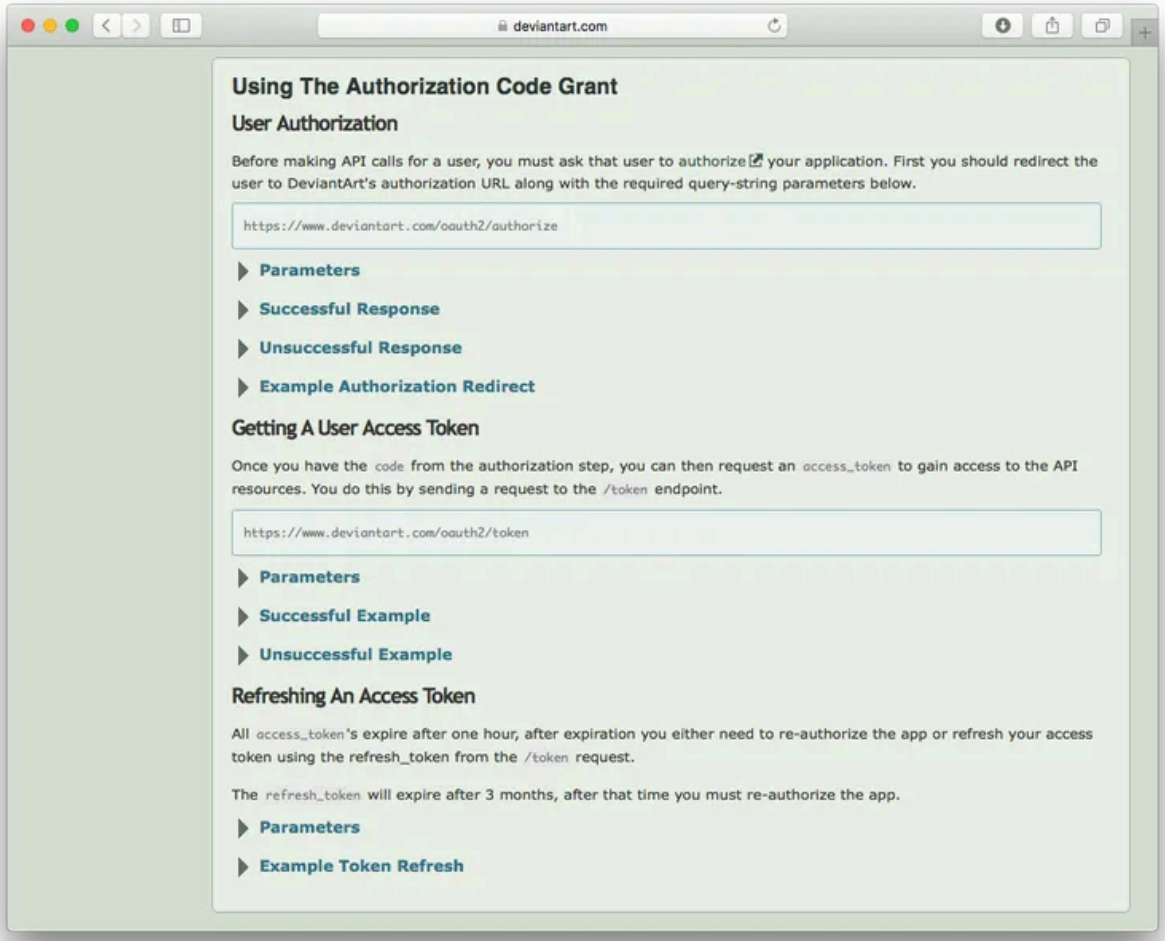
   The value of this parameter is always the same - *https://www.integromat.com/oauth/cb/oauth2*. This is the URL, that a user is redirected to after they authorize Make to access the given service. This way, Make learns that the authorization has been successful, and creates a new connection with the service.



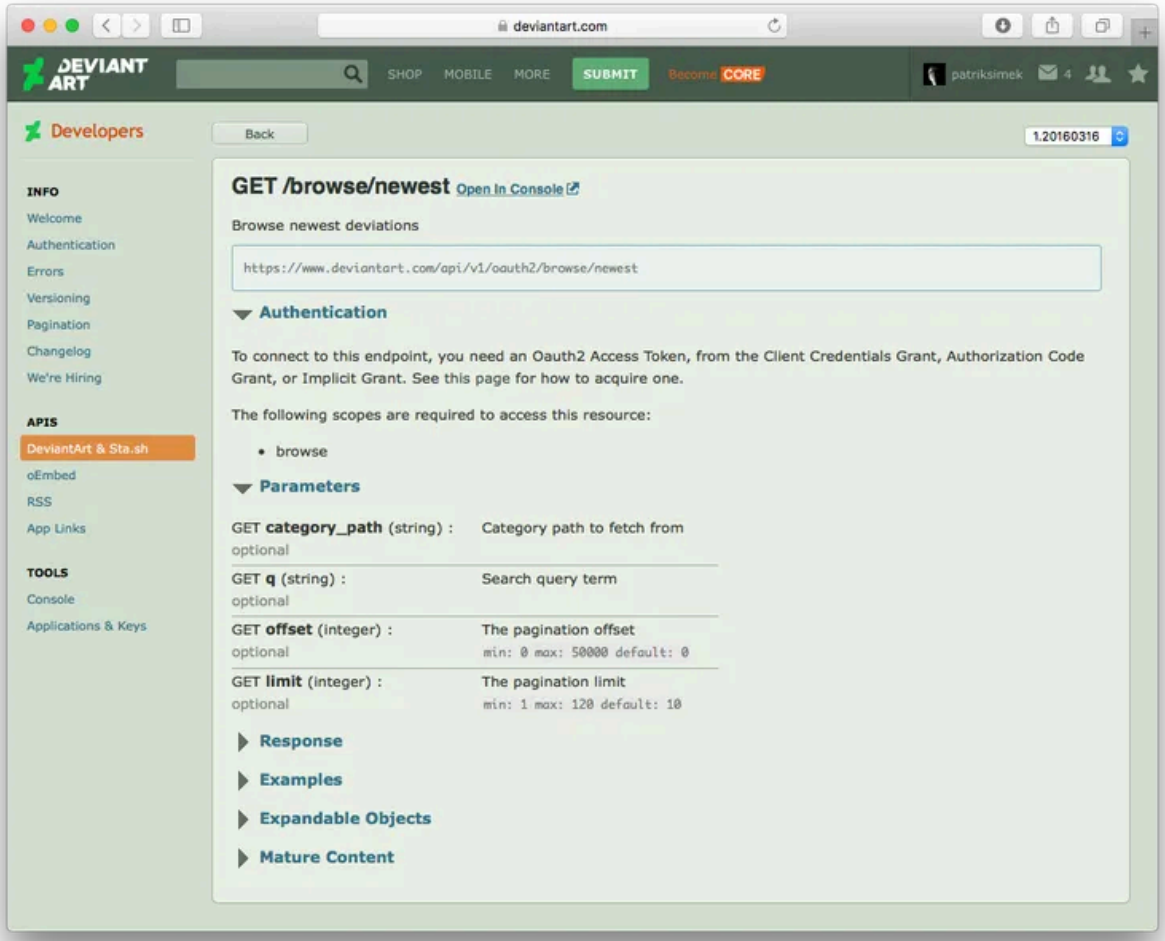
4. Once you have completed the registration, a screen opens where you can view two important values - **client_id** and **client_secret**. Write down these two values, you need them later on.



5. Go back to the **Authentication** section and get authorization URL and access token URL. All services that use standard authorization OAuth2 protocol always include both URL addresses in their documentation.



6. Go to the API section which describes all API calls in detail. The API you need is in **GET /browse/newest**.
7. Expand the **Authentication** and **Parameters** sections to learn which scope is required to call this API and which parameters you can send along with the request.

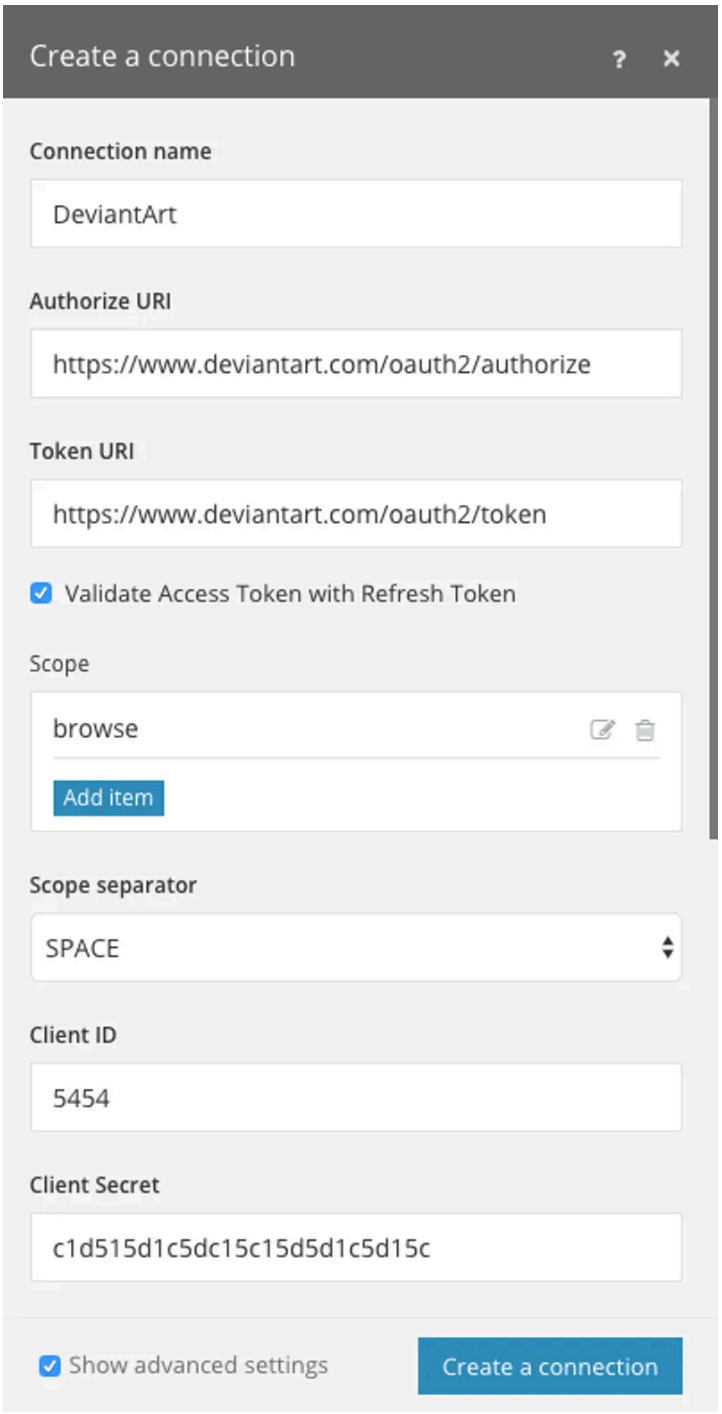


Creating a scenario

> Before you start setting up your scenario in Make, we recommend that you go through all APIs that you plan to call and get a list of all scopes that are needed. This way you can acquire all the necessary scopes at once and avoid the hassle of expanding the scopes in the future.

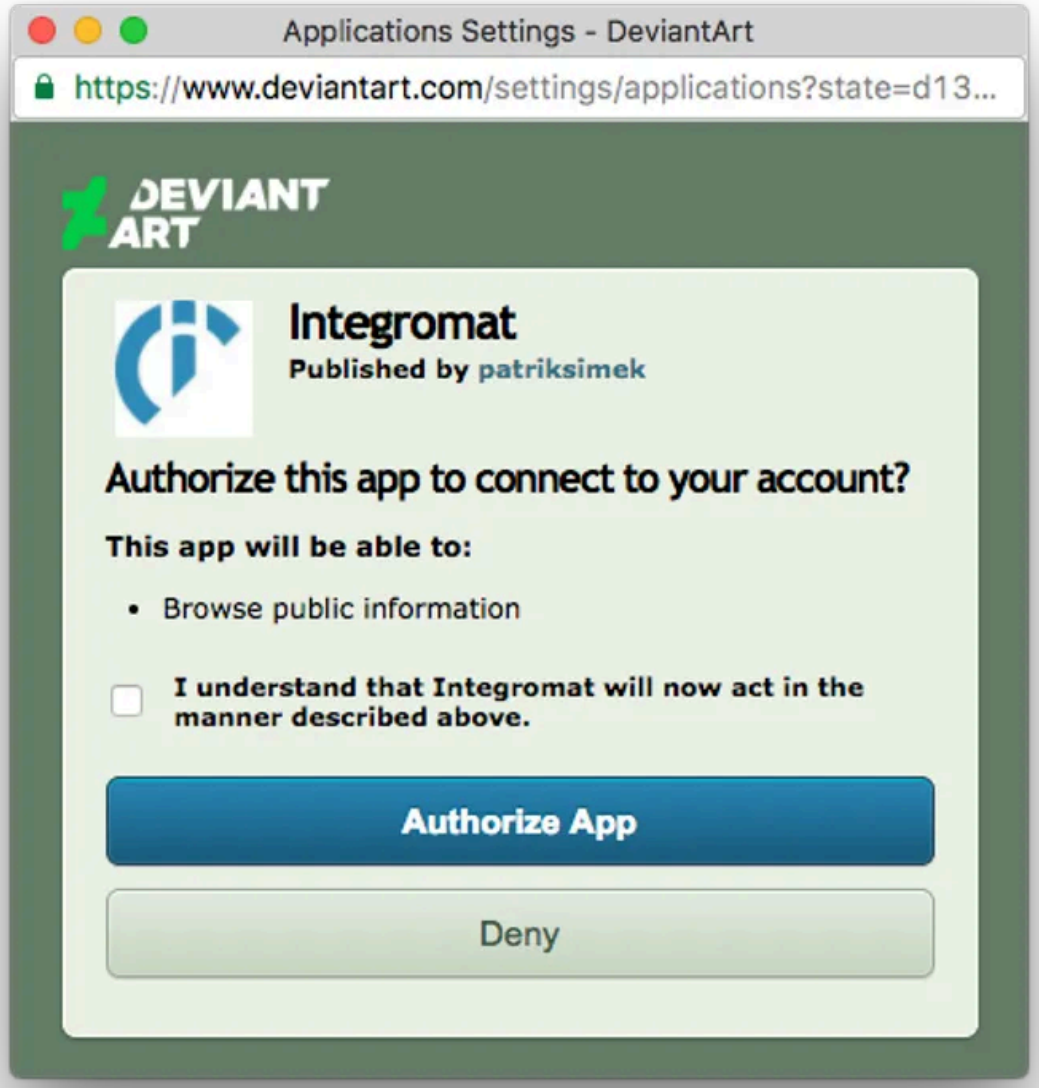
1. Go to Make and create a new scenario.
2. Choose the **OAuth 2.0 HTTP(S) request and response processing module** from the HTTP app and add it to your scenario.

   In the open configuration window, click **Add**. A new window opens that allows you to add a new OAuth2 connection. In this step, enter the information that you have found in the documentation:



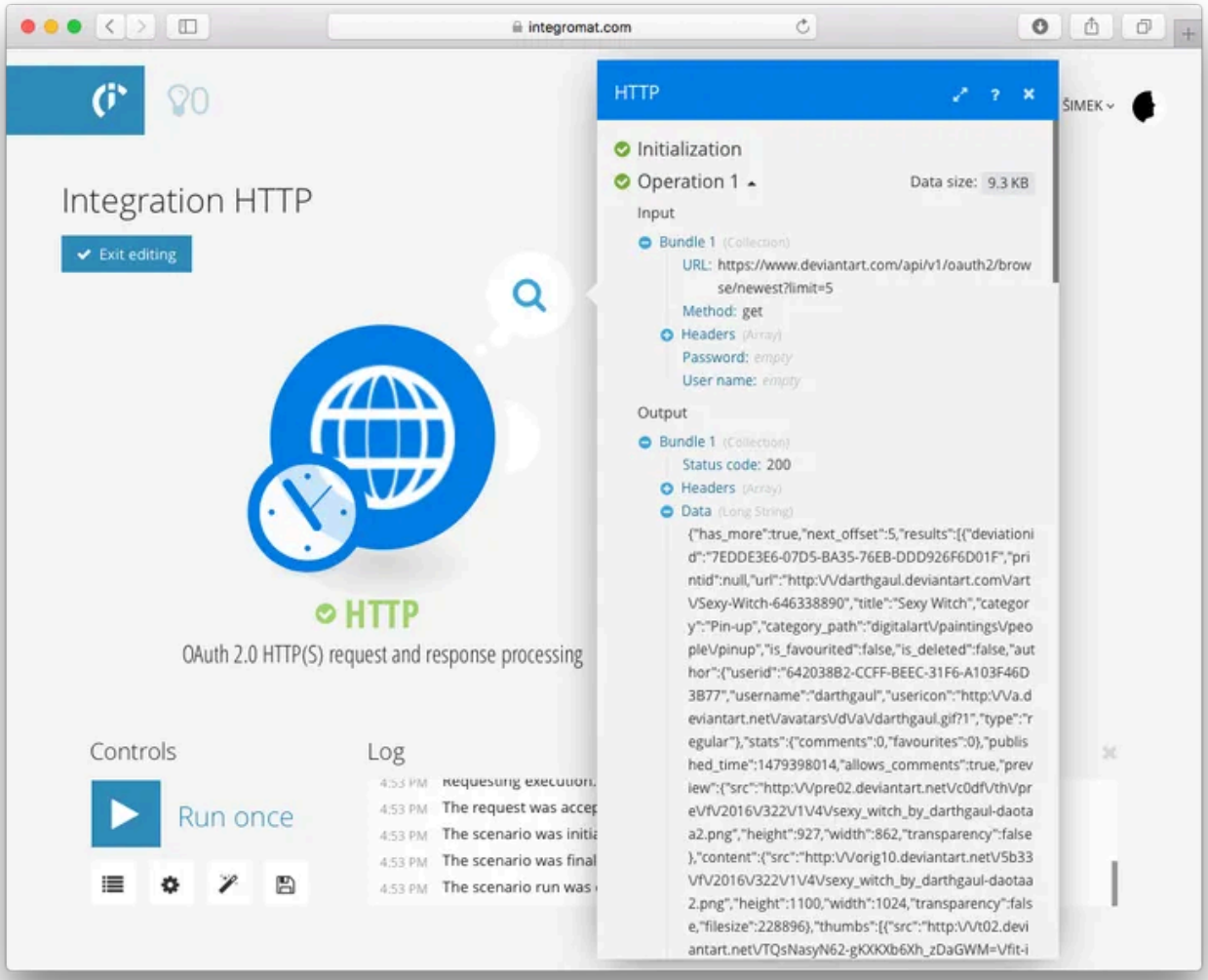
3. Click **Create a connection**.

   A new window pops up. Click Authorize App to authorize Make to access the given service via the application you have created.

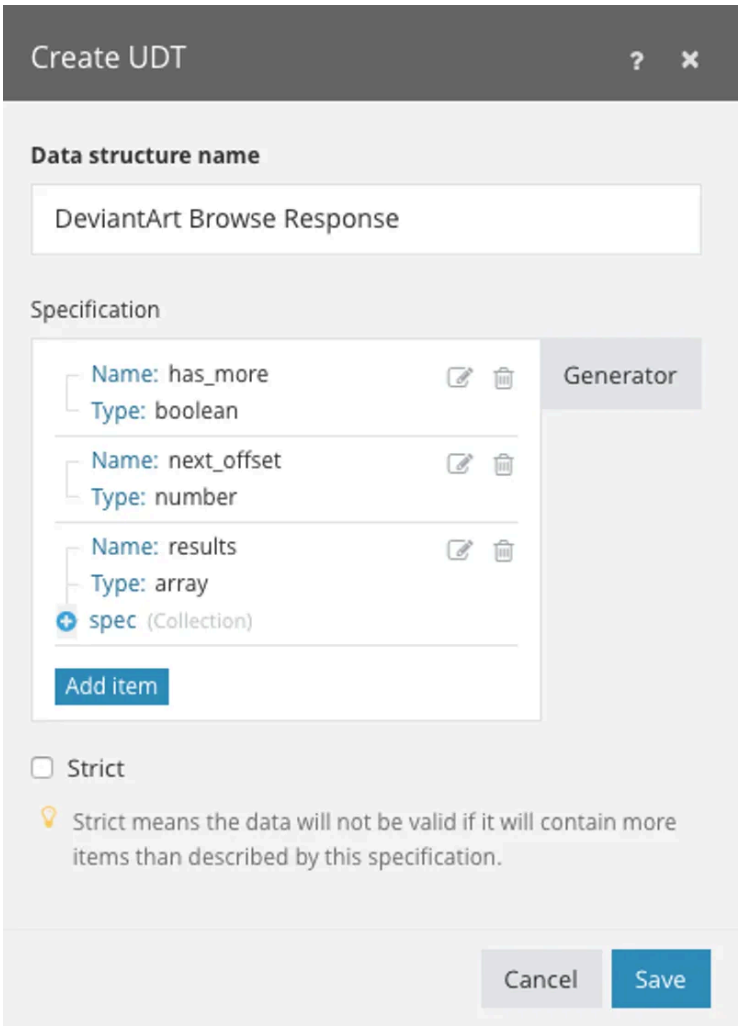


The access tokens exchange in the background automatically. The configuration window automatically populates with the newly created connection.

4. Set up the URL address of the API endpoint according to the specifications given in the API documentation and specify optional parameters (in our case, we will enter `https://www.deviantart.com/api/v1/oauth2/browse/newest`).
When creating a scenario, we recommend working with a smaller number of bundles.


integromat.com
Integration HTTP
Exit editing
HTTP
OAuth 2.0 HTTP(S) request and response processing
Controls
Run once
Log
HTTP
Initialization
Operation 1
Data size: 9.3 KB
Input
Bundle 1
URL: https://www.deviantart.com/api/v1/oauth2/browse/newest?limit=5
Method: get
Headers
Password: empty
User name: empty
Output
Bundle 1
Status code: 200
Headers
Data
ŠIMEK

Testing

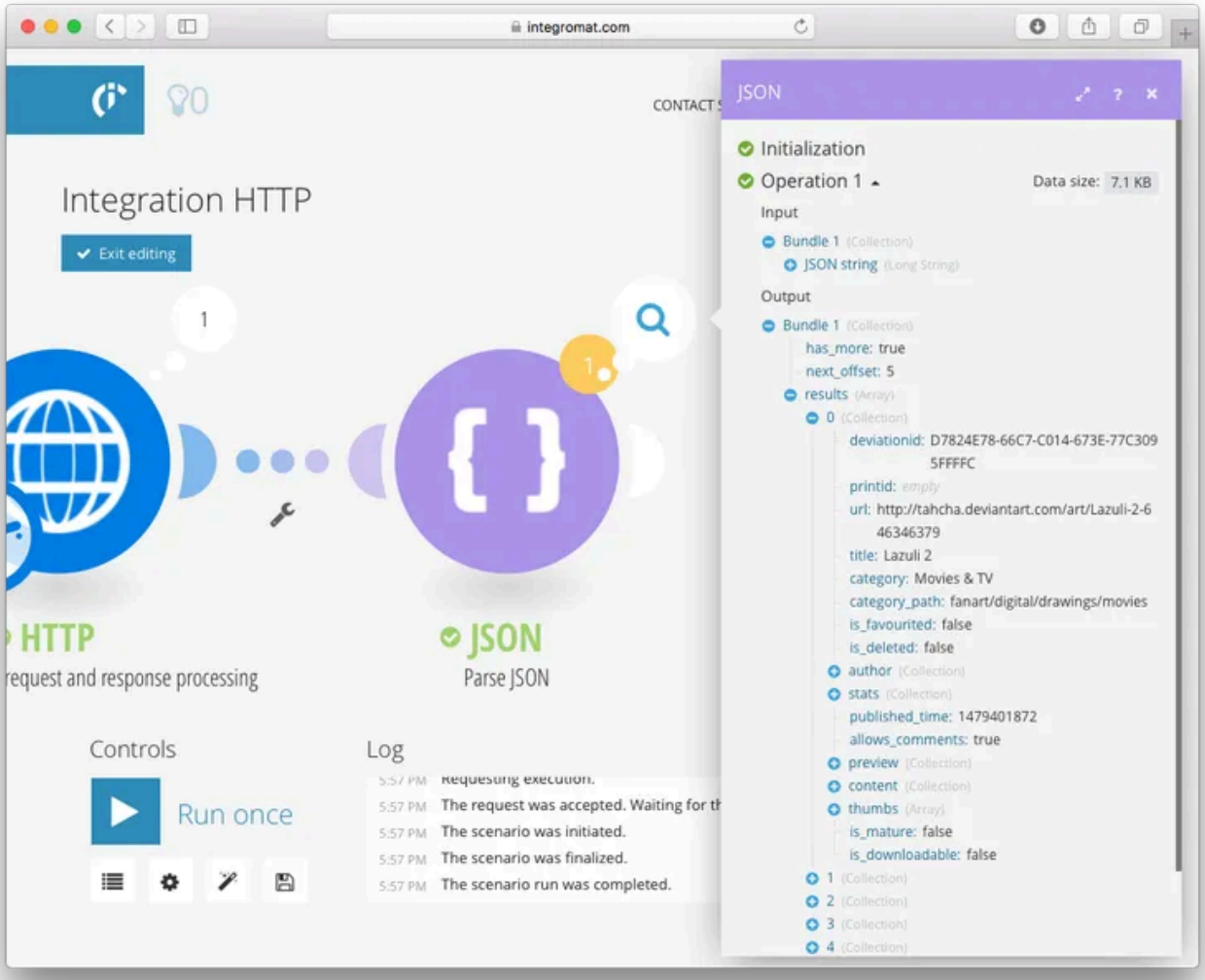


1. Run your scenario.
   If the execution results show that the call ended with a 200 status, the run completed successfully. The response details contain complete JSON data that is available for further use.
2. Copy the whole JSON response to the system clipboard - you'll need it in order to generate a data structure.
   Thanks to the data structures, Make knows exactly what structure the received data has.
3. Use **JSON app** > **Parse JSON** module to parse your data in order to use the data in your scenario.
4. In the settings of **Parse JSON** module, map the **Data** variable from the HTTP app to the **JSON string** field.
5. Next to the **Data structure** field, click **Add**. A panel opens. Select **Generator**.
6. In the data structure generator, choose the JSON value as **Content type**. In the **Sample data** field paste the JSON response from the server that you have copied to the system clipboard.
   Once you confirm your entries, the data structure generates automatically.

7. Give a name to the Data structure and click **Save**.
   Click **Save** to confirm the settings.

Run the scenario.

Once the scenario run completes, view details of the JSON module run. If everything has been processed without errors, you will see a list of 5 new DeviantArt deviations. You can then easily browse through the data structure and map obtained values to other modules.



3.3. Tools

Build complex scenarios with the help of these advanced tools:

→ **Flow control**

→ **Converger**

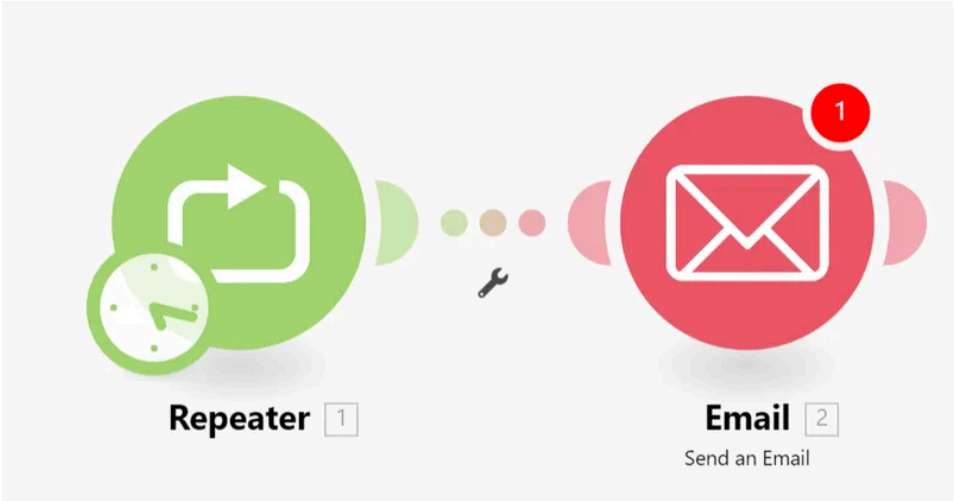
→ **Tools**

→ **Text parser**

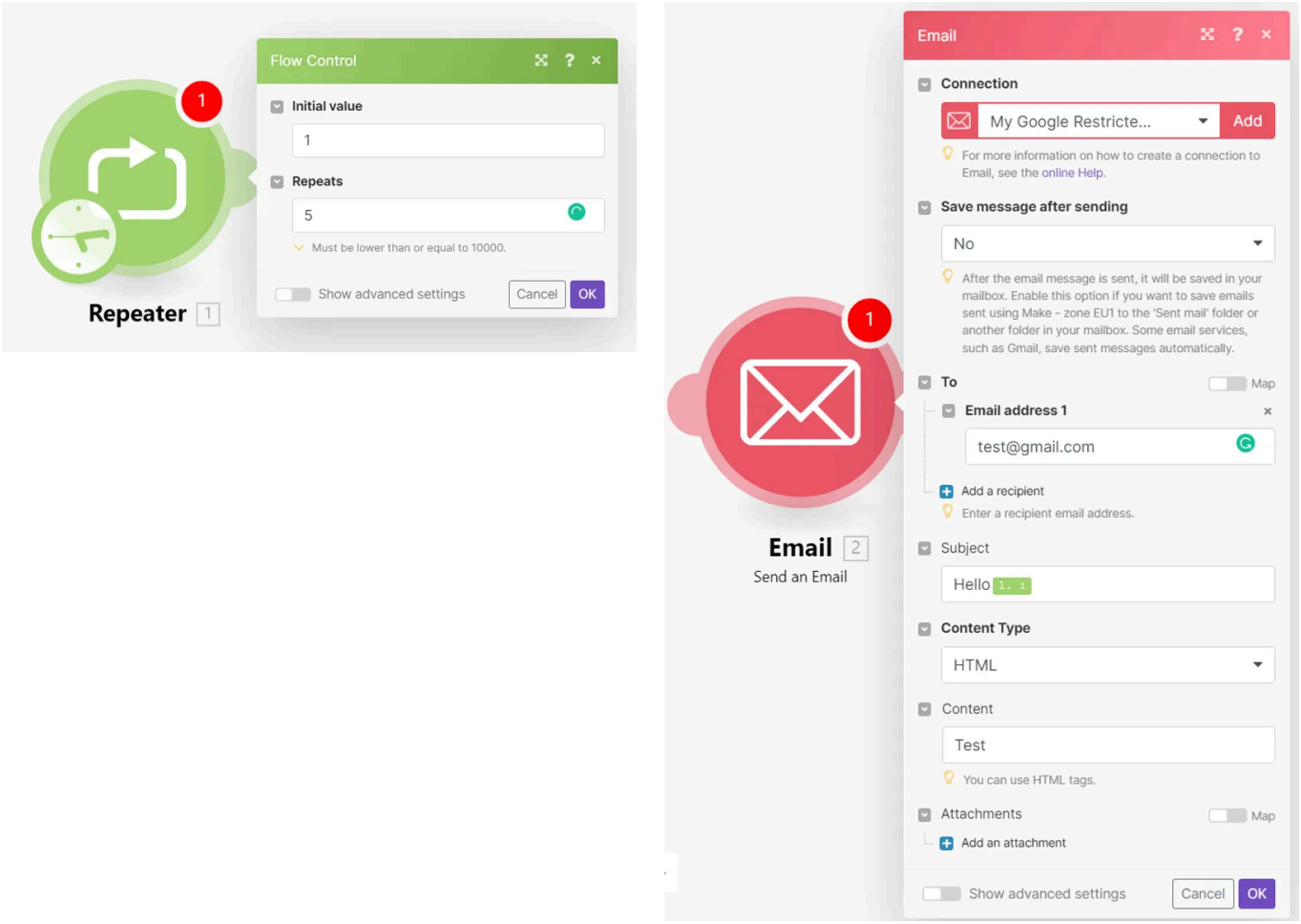
3.3.1. Flow control

Repeater

A repeater is used in cases where you wish to repeat a task a set number of times. For example, if you would like to send five emails with subjects "Hello 1", "Hello 2", ... "Hello 5", this could be accomplished with connecting the **Email > Send an email** module after the **Repeater** module:



Module configuration details:



Flow_Control_3.png

You can imagine the **Repeater** module as a generator of bundles outputting one bundle after another. Each bundle contains one item named `i` of type Number. The initial value of the `i` item is specified in the **Initial value** field. The number of repetitions (= number of bundles) is specified in the **Repeats** field. The value of the `i` item is increased in each repetition by the value specified in the **Step** field, which is 1 by default (toggle the **Show advanced settings** box to reveal it).

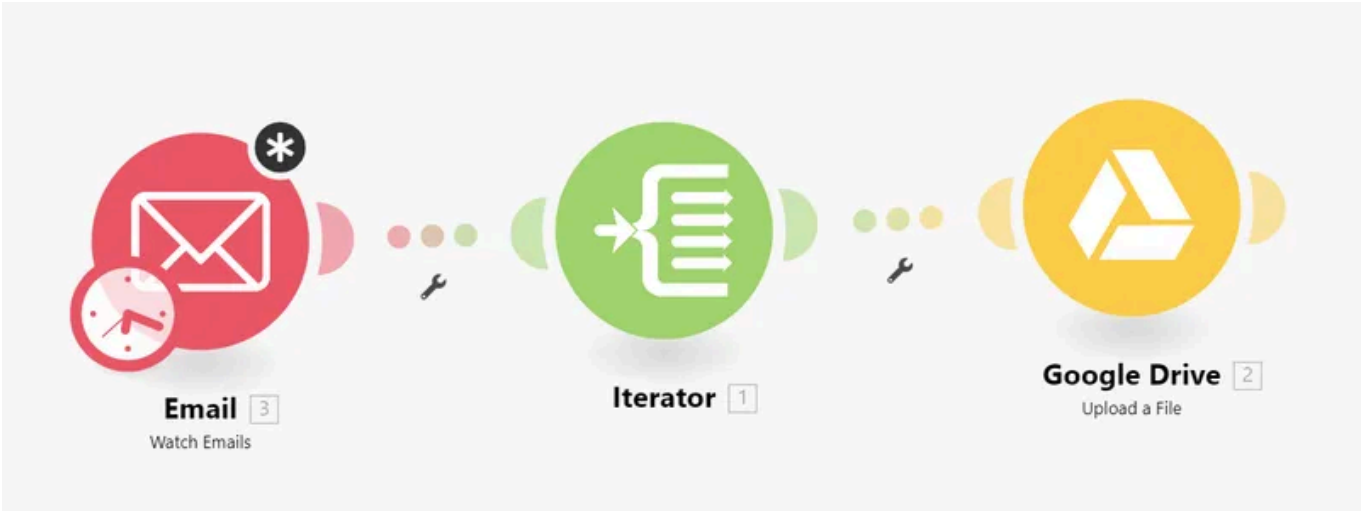
Iterator

An iterator is a special type of module that converts an array into a series of bundles. Each array item will output as a separate bundle.

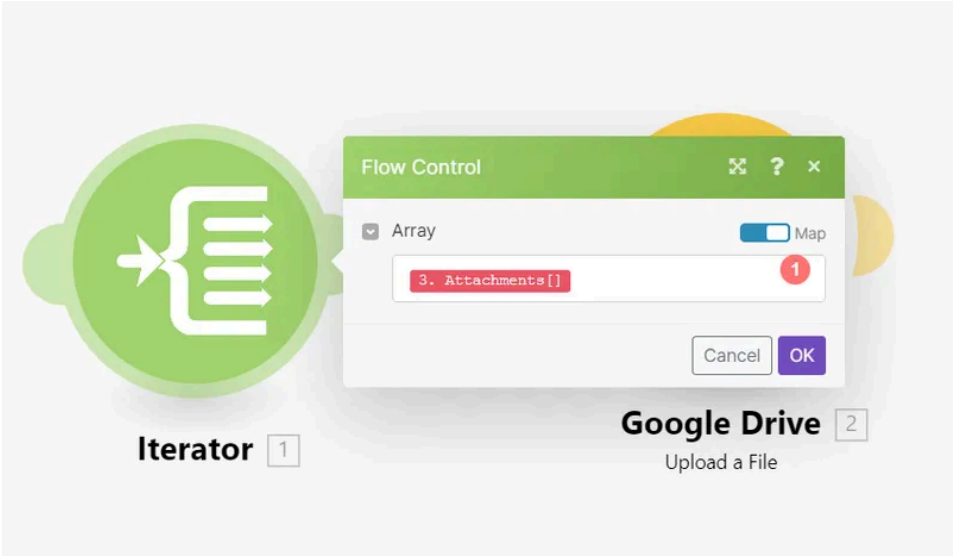
Setting up an iterator

Setting up an iterator is done in the same way as **setting any other module**. The **Array** field contains the array to be converted/split into separate bundles.

In this example, the scenario below shows how to retrieve emails with attachments and save the attachments as single files in a selected Google Drive folder.



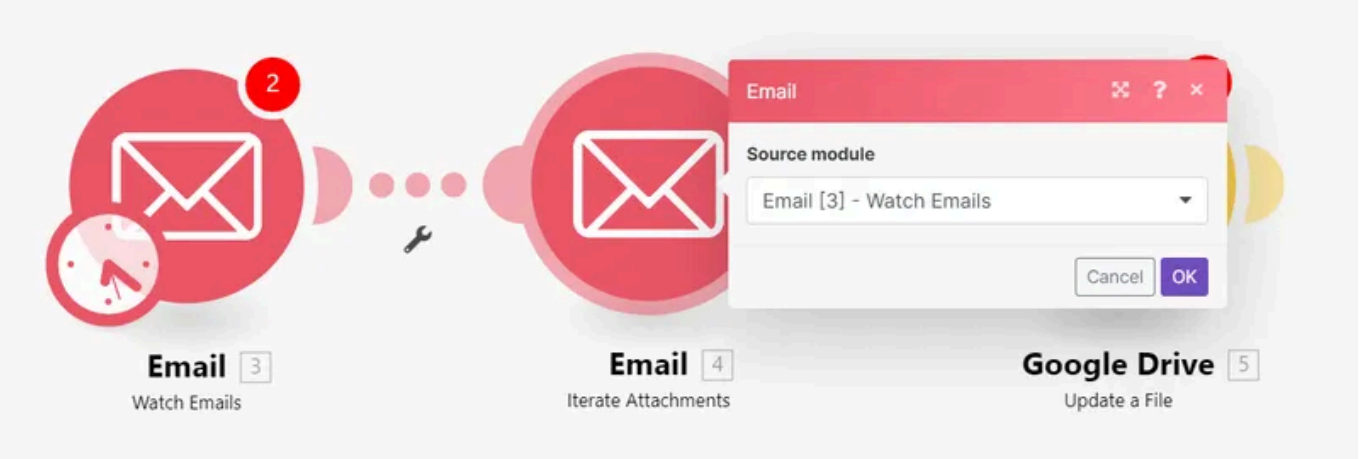
Emails can contain an array of attachments. The **Iterator** module inserted after the first module enables you to handle each attachment separately. The iterator splits the array of attachments into single bundles, each bundle with one attachment will then save one at a time in a selected Google Drive folder. The **Array** field should contain the `Attachments[]` array.



Specialized iterators

For your convenience, many Make apps offer specialized iterator modules with a simplified setup. For example, the Email app contains the special iterator **Iterate attachments** that will

produce the same results as the general **Iterator** module without having to specify the array, just the source module.



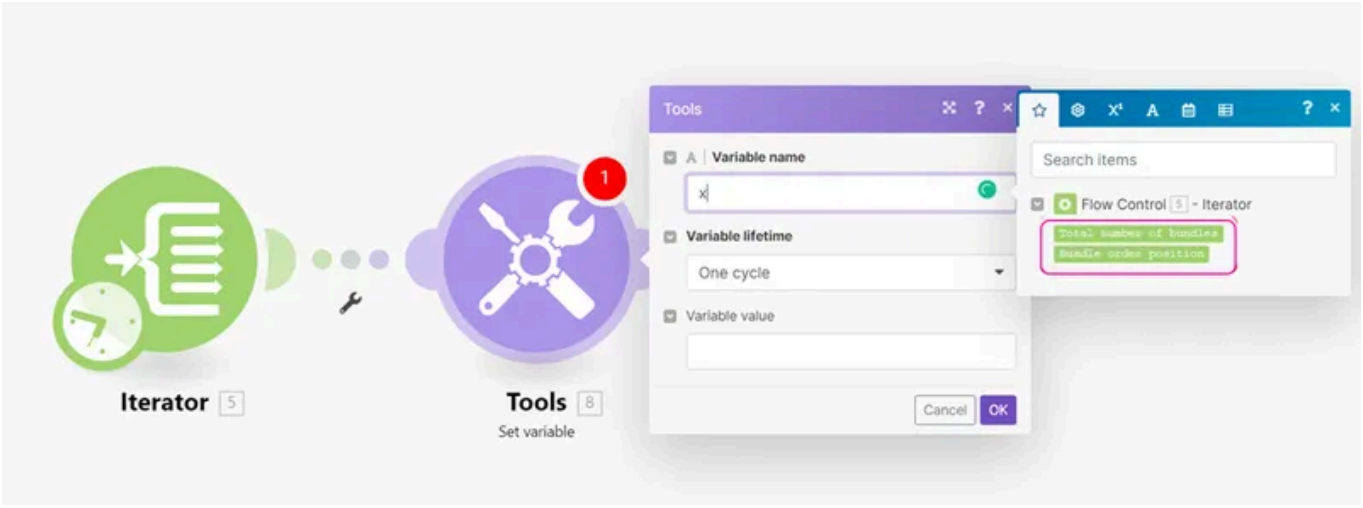
When to use an iterator in your scenario

The video below is module one of the three-part lesson titled **Iterator and Array Aggregator**. It explains the purpose of the iterator and the array aggregator and with the help of a sample scenario, when to use an iterator, and what to do with the output.

Iterator & Array Aggregator | Part 1: Introduction and Iterator Demo

Mappable items under the Iterator module

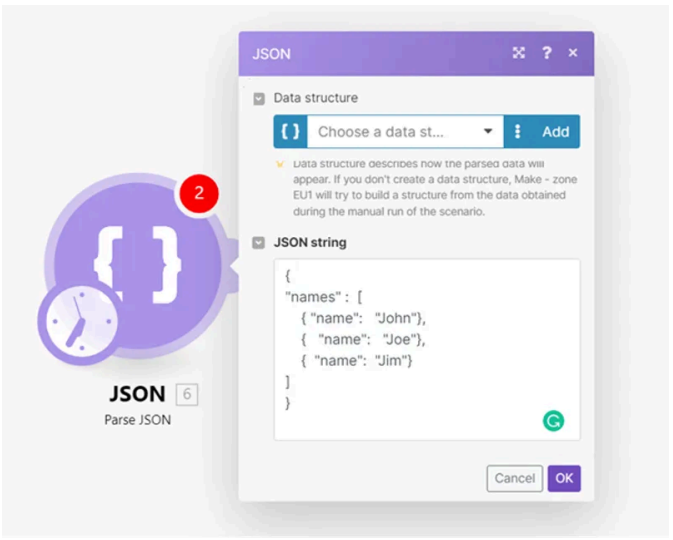
When an iterator does not have information about the structure of the array's items, the mapping panel in the modules following the iterator will display only two items under the iterator: `Total number of bundles` and `Bundle order position`:

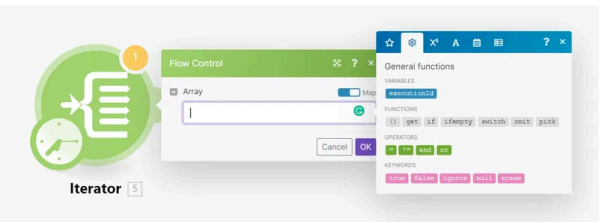


The reason for this is that in Make, each module is responsible for providing information about items it outputs so these items can be properly displayed in the mapping panel in the following modules. However, there are several modules that might be unable to provide this information in some cases, e.g. **JSON > Parse JSON** or **Webhooks > Custom Webhook** modules with missing **data structures**.

The solution is to manually execute the scenario to make the module learn about the items it outputs so it can provide the information to the following modules.

For example, if you have a **JSON > Parse JSON** module without a data structure and then you connect an **Iterator** module to it, you will not be able to map the output of the module to the **Array** field in the setup panel of the iterator.

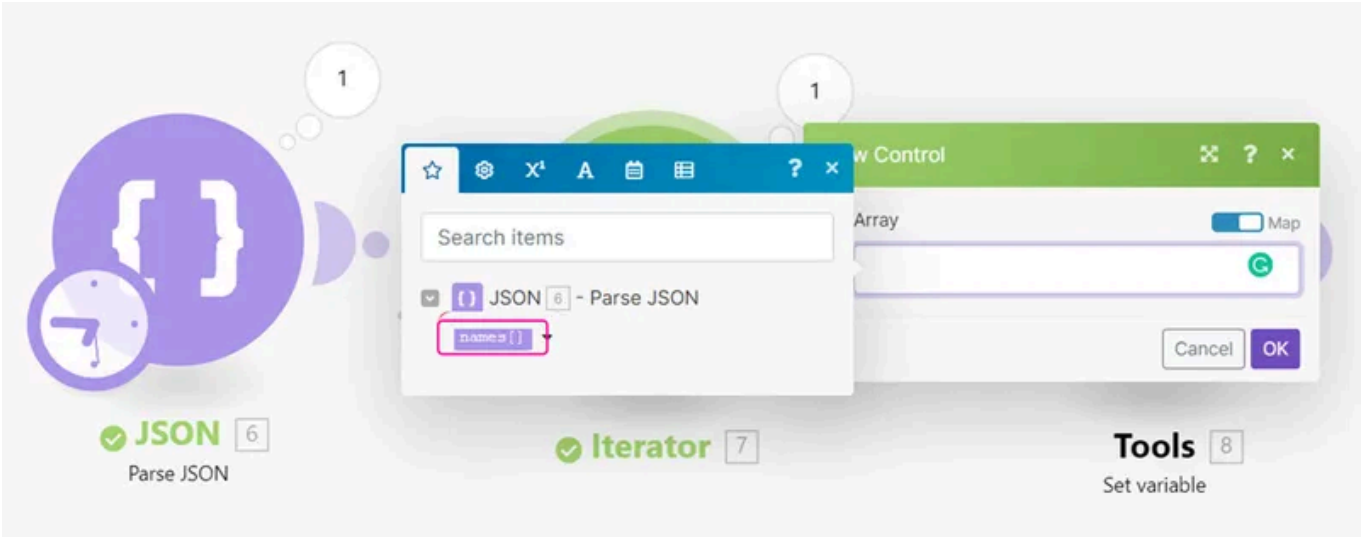




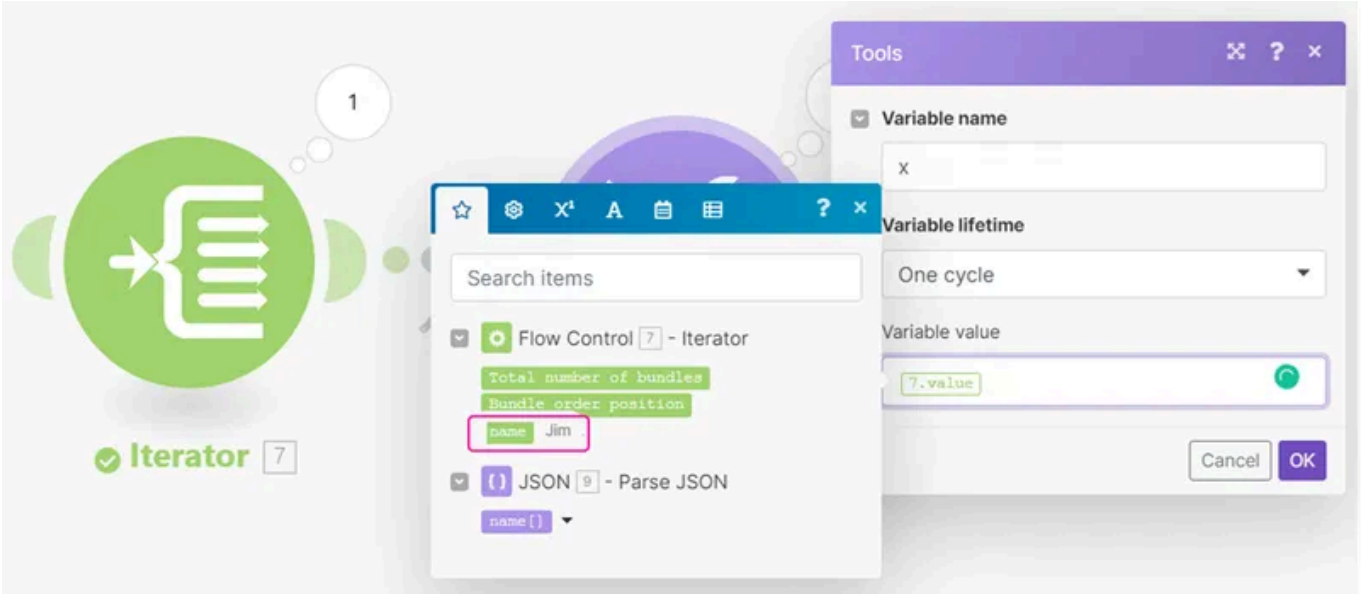
To resolve this, manually start the scenario in the scenario editor. You can un-link the modules after the **JSON > Parse JSON** module to prevent the flow from proceeding further

or right-click the **JSON > Parse JSON** module and choose **Run this module only** from the context menu to execute only the **JSON > Parse JSON** module.

Once the **JSON > Parse JSON** has been executed, it learns about the items it outputs and provides this information to all the following modules including the iterator. The mapping panel in the iterator's setup will then display the items:



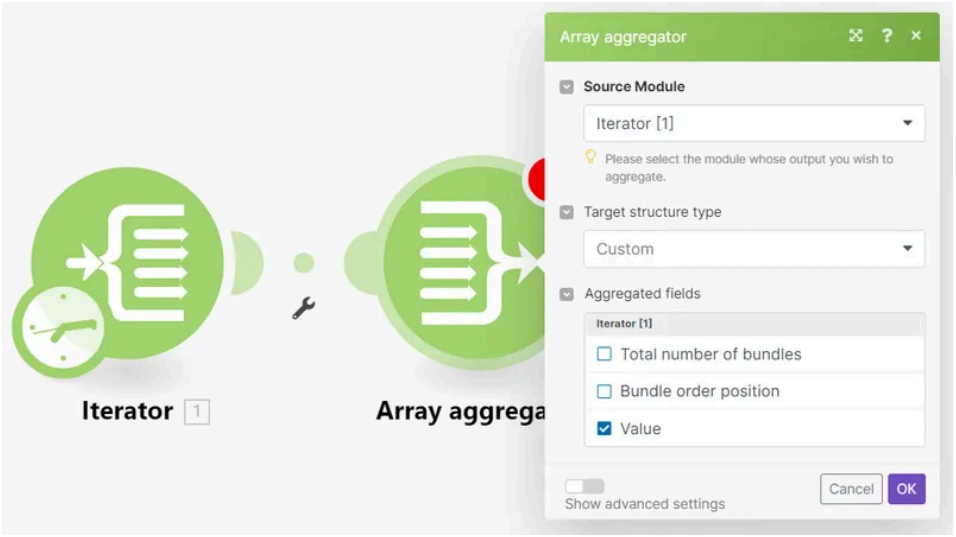
The mapping panel in the modules connected after the iterator will display the items contained in the array's items:

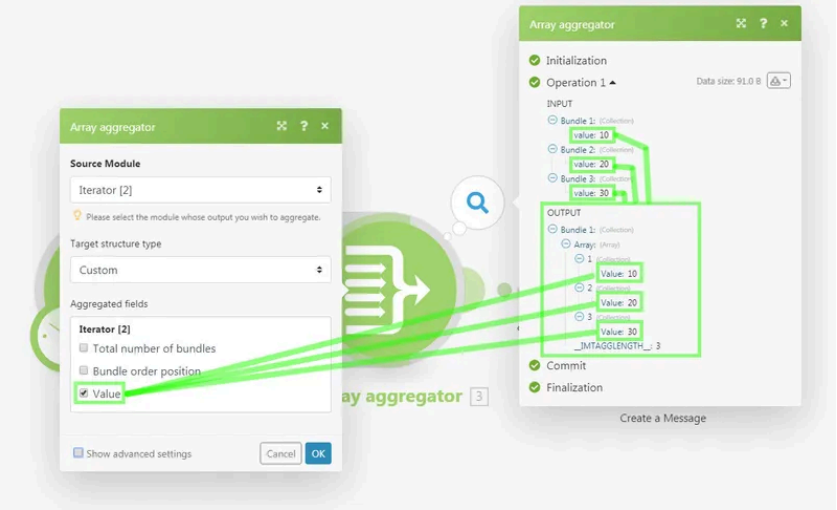


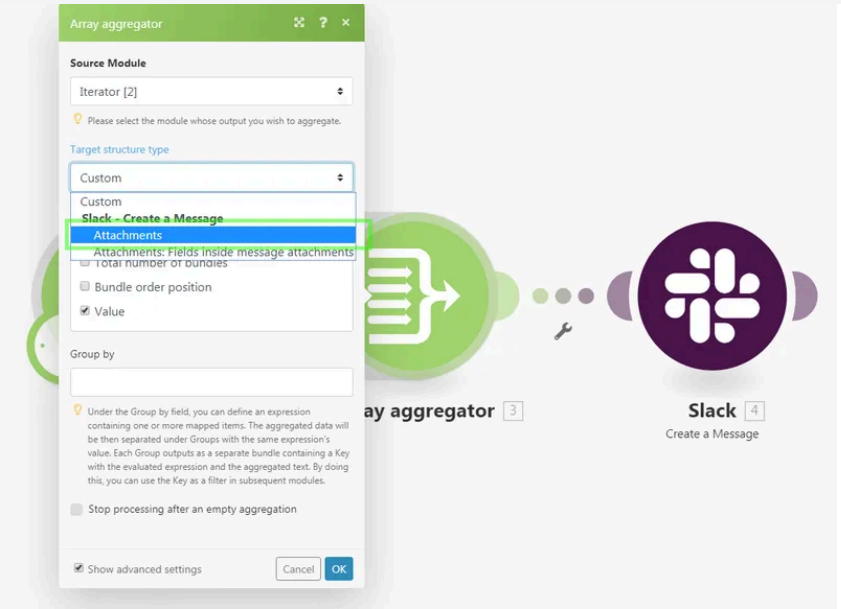
If you cannot see some items in a module's mapping panel, simply run the scenario once so all the modules can learn about the items they output and provide this information to the following modules.

Array aggregator

An array aggregator is an aggregator module that allows you to merge several bundles into one single bundle. The following image shows a typical setup of the **Array aggregator** module.



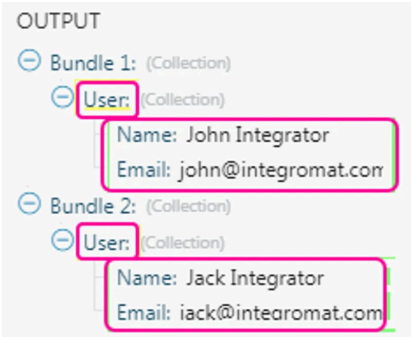
Field	Description
Source Module	The module from which the bundle aggregation will start. The source module is usually an iterator or a search module that outputs a series of bundles. Once you setup the aggregator's Source Module (and close the aggregator's setup), the route between the source module and the aggregator will be wrapped in a grey area to visualize the start and the end of the aggregation.
Target structure type	The target structure into which the data shall be aggregated. The default option is Custom that enables you to choose items that should be aggregated into the Array aggregator's output bundle's `Array` item:  Once you connect more modules after the **Array aggregator** module and get back to the module's setup, the **Target structure type** dropdown will contain all the following modules and their fields that are of type Array of Collections, like e.g. **Attachments** field of the **Slack > Create a Message** module:

Field	Description
	
Group by	The aggregator's output can be split into several groups with the help of the **Group by** field. The **Group by** field can contain a formula that is evaluated for each aggregator's input bundle. The aggregator then outputs one bundle per each distinct formula's value. Each bundle contains two items: • `Key` contains the distinct value. • `Array` contains the aggregated data from the bundles for which the formula evaluated to the `Key` value.
Stop processing after an empty aggregation	By default, the aggregator outputs the result of the aggregation even in case no bundles reached the aggregator (e.g. because they have been all filtered out on their way). If the **Stop processing after an empty aggregation** option is enabled, the aggregator will not produce any output bundle in this case and the flow will stop.

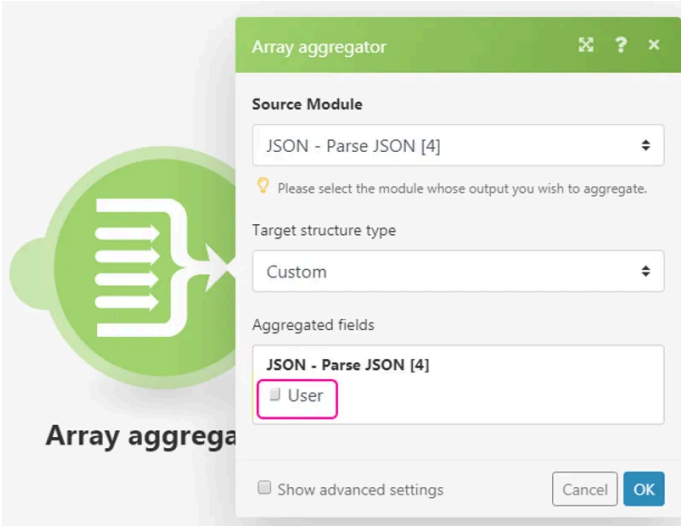
> Bundles output from the source module and any other modules between the source module and the aggregator module are not output by the aggregator and thus items in these bundles are not accessible by the modules in the flow after the aggregator.

If you need to access items from bundles output from the source module and any other modules between the source module and the aggregator module, make sure to include them in the **Aggregated fields** field in the setup of the **Array aggregator** module.

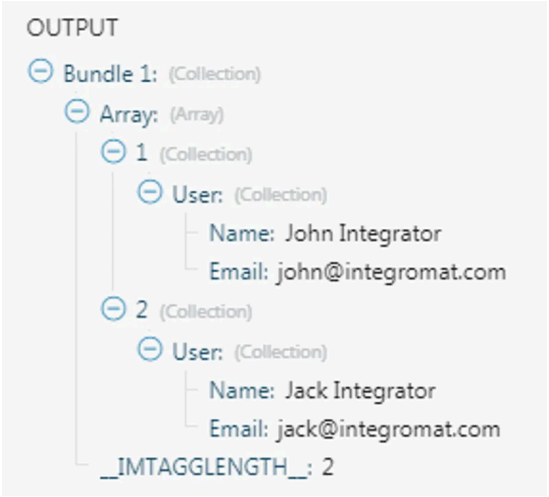
If items are nested (i.e. contained in a collection item) they currently cannot be easily selected in the array aggregator's **Aggregated fields** field. For example, if bundles contain collection item `User` with two items `Name` and `Email`:



Then only the `User` collection item can be selected:



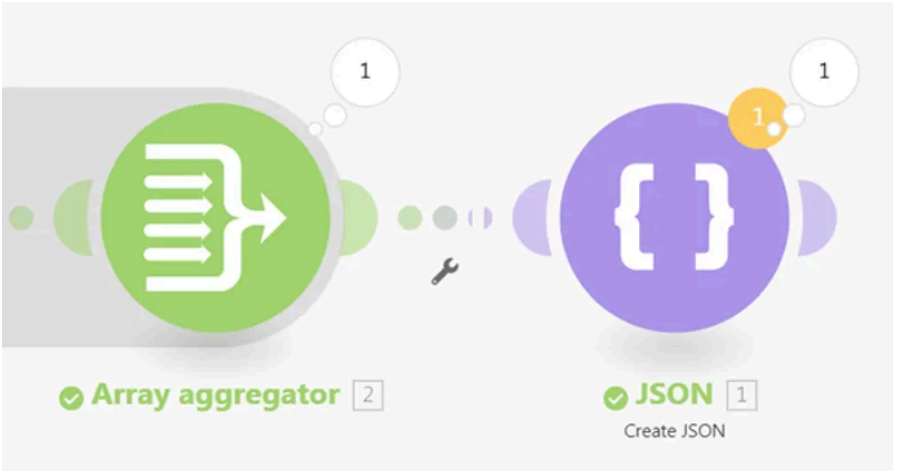
This setup will produce the following output:



Customize the output

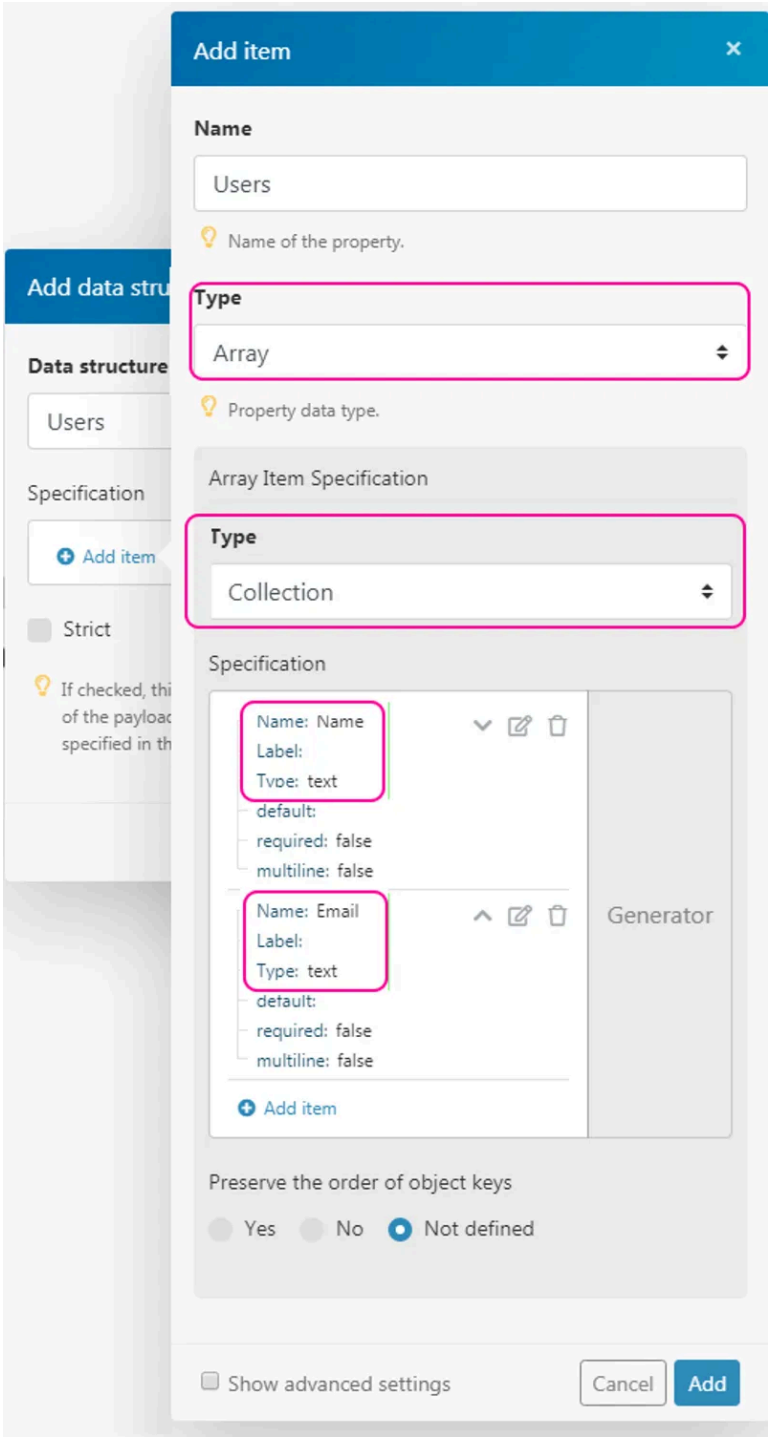
If you wish to fully customize the **Array aggregator**'s output structure:

1. Insert the **JSON > Create JSON** module after the **Array aggregator** module:



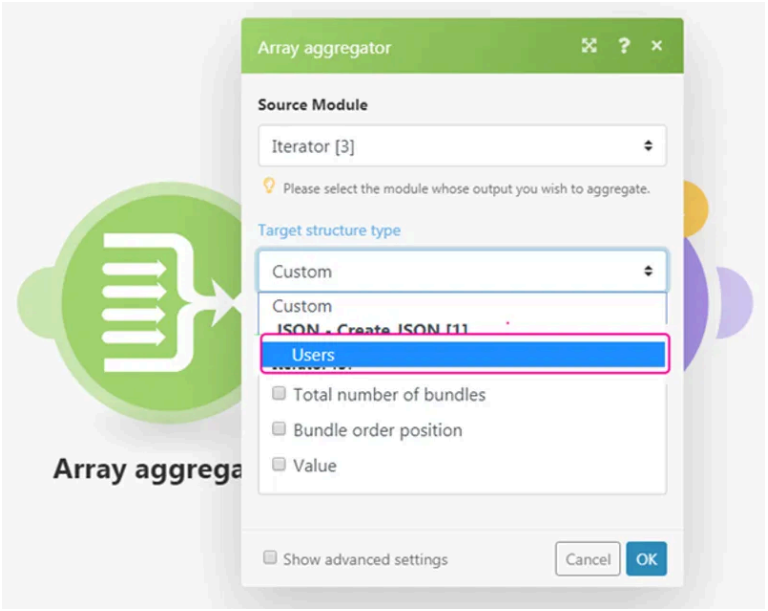
2. Open the **JSON > Create JSON** module's setup.

3. Setup a Data structure for the items you want to be output from the array aggregator. The data structure should be an array of collections and the collections should contain the items you want to include in the output. Here is a sample data structure with two text items `Name` and `Email`:

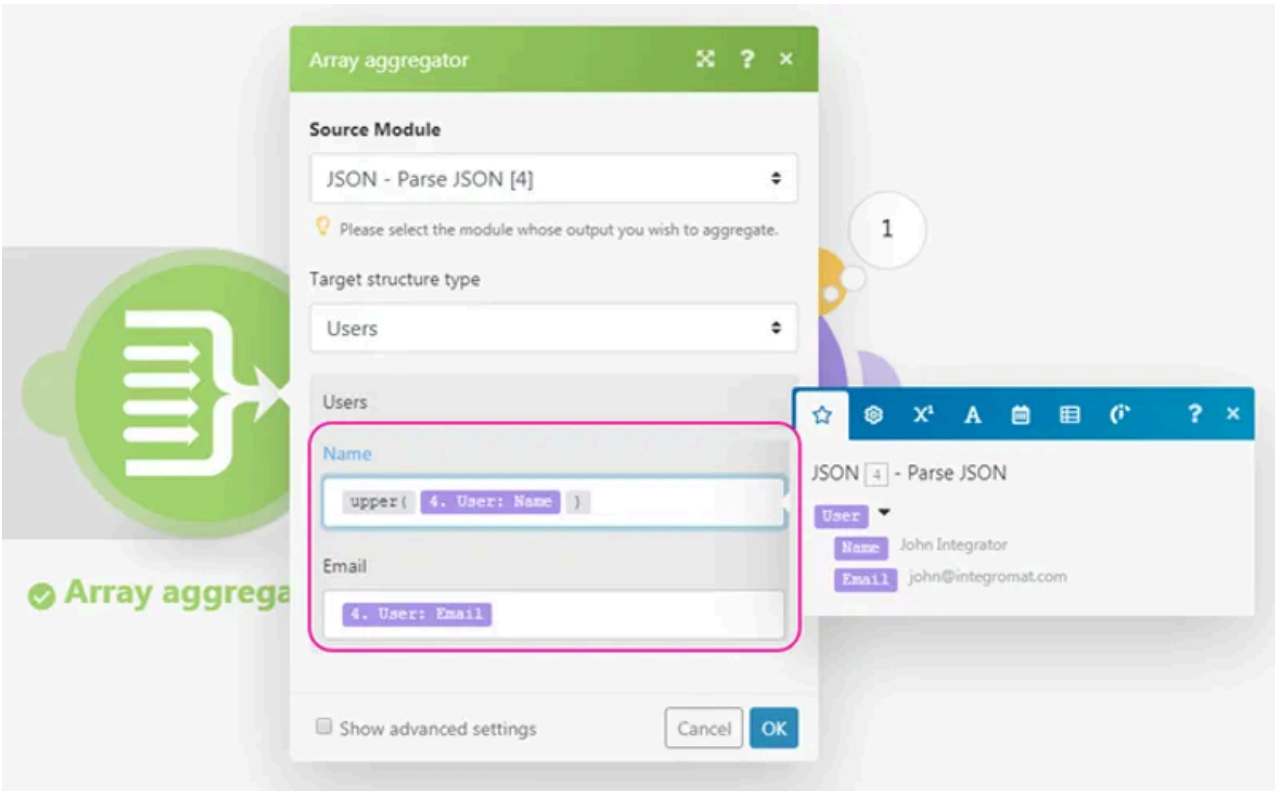


4 Open the **Array aggregator** module's setup.

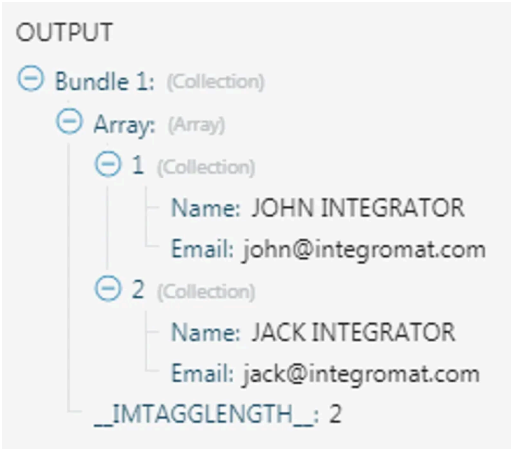
5 In the **Target structure type** field, choose the **JSON > Create JSON** module's array field:



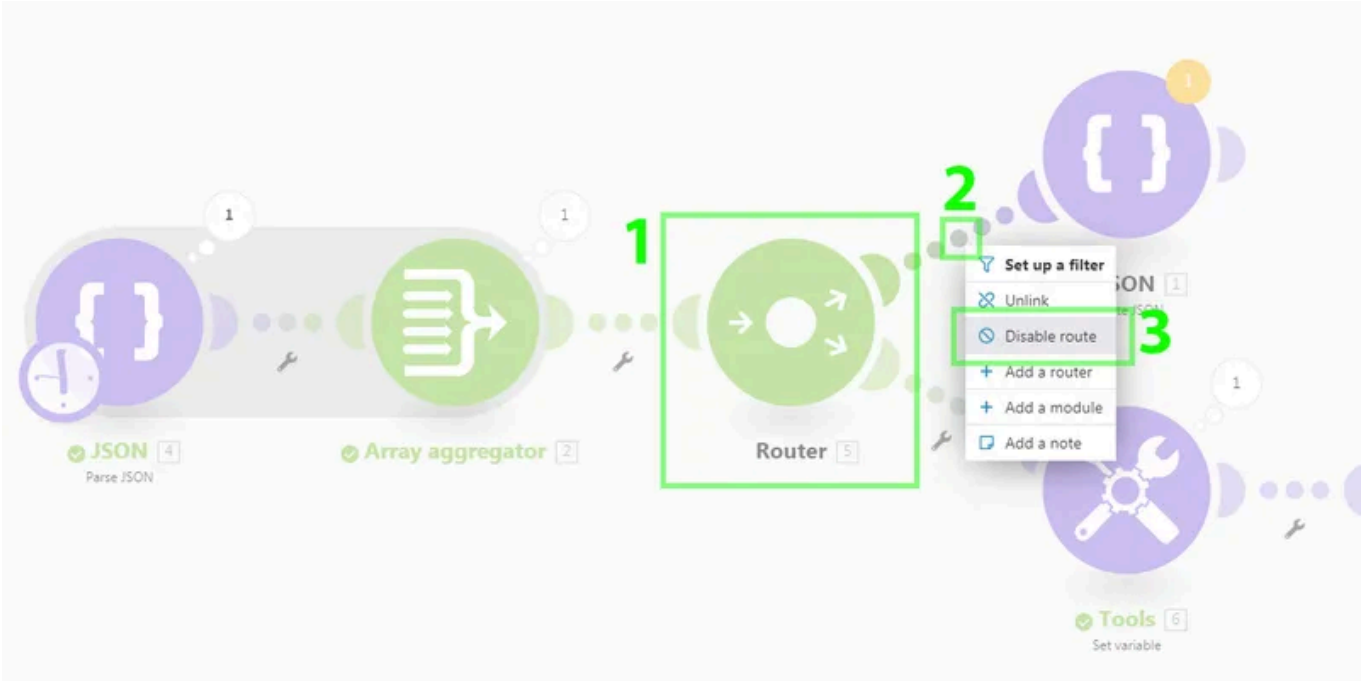
6 Fields corresponding to the Data structure created in step 3 will appear in the setup of the **Array aggregator** module. Map any items into the fields. You can now easily map nested items using the mapping panel and even use formulas:



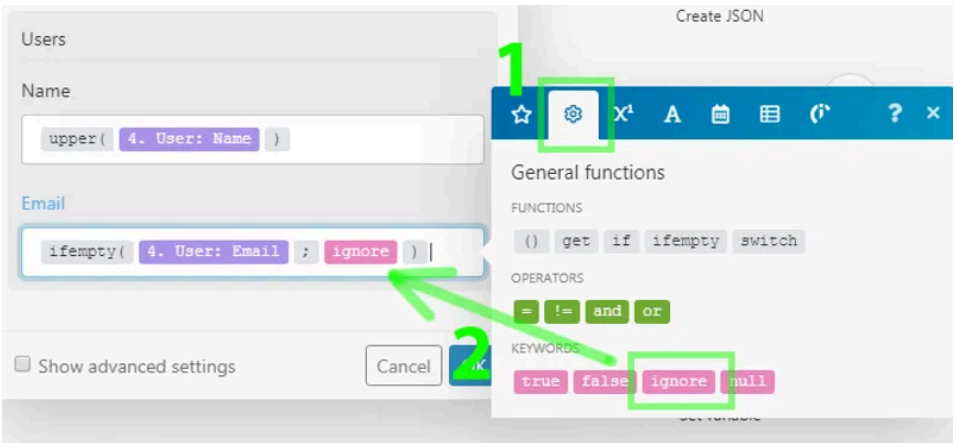
7 The **Array aggregator** module's output will now look like this:



If you wish to save the operation performed by the **JSON > Create JSON** module, put it on a disabled route after a **Router**:

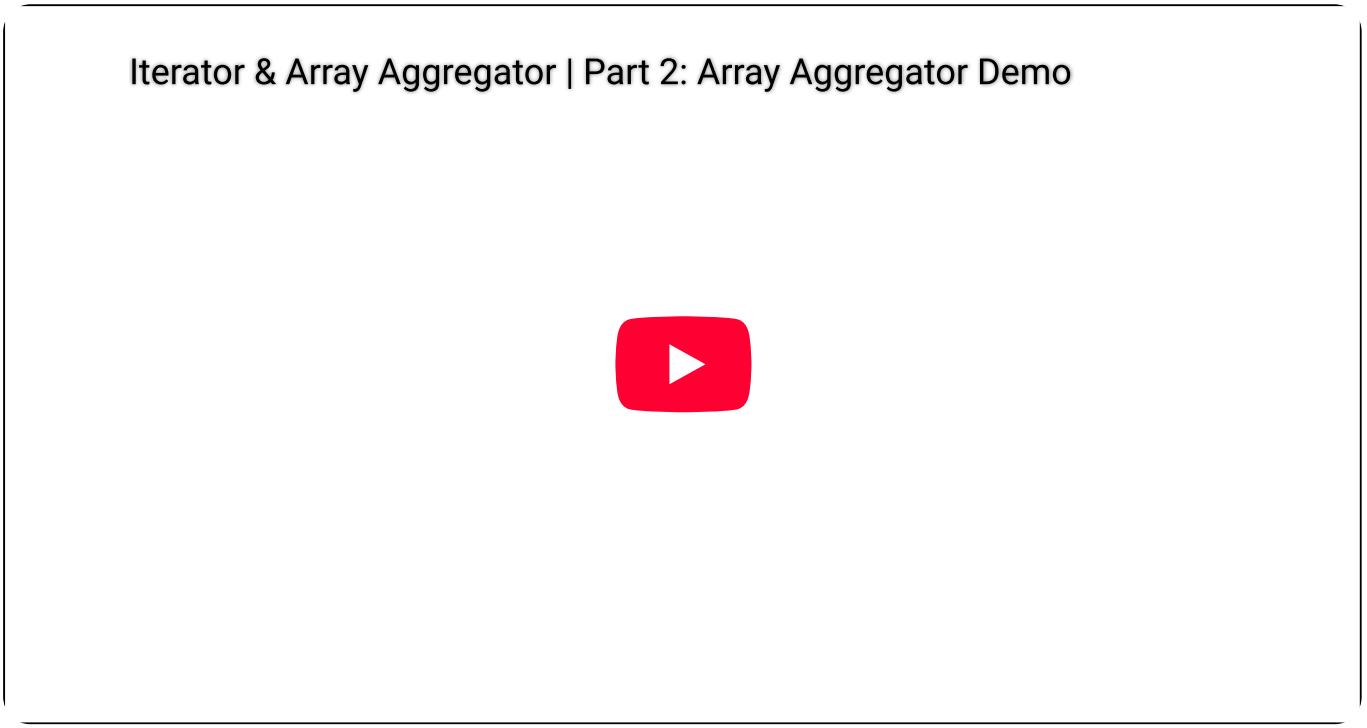


If you wish to conditionally omit an item from the module's output, use a formula that evaluates to `ignore` keyword:

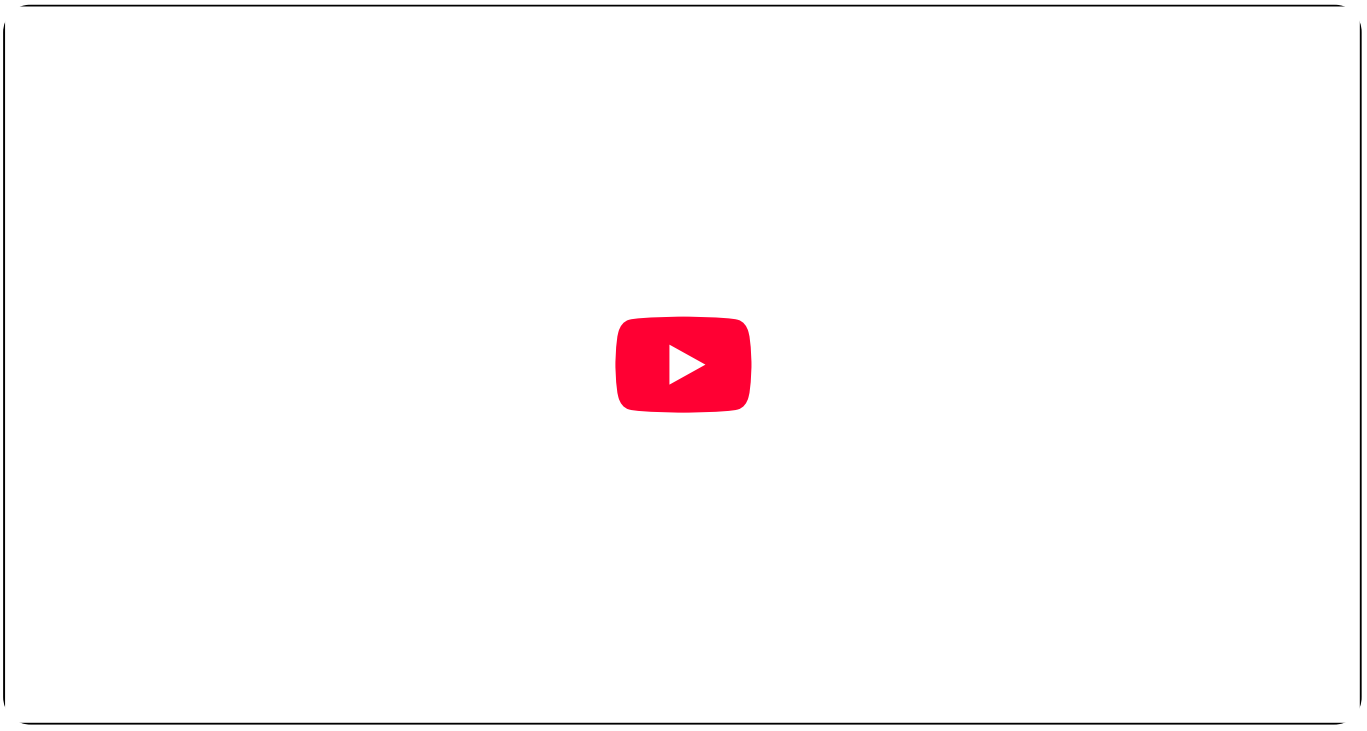


If the `4. User: Email` is empty then the `Email` item will be completely omitted from the output:

When to use an array aggregator in your scenarios



How modules connected after an array aggregator affects the target structure type dropdown



3.3.2. Converger

> This article describes alternatives and workarounds to a converger module, often requested by our customers.

Please note that there is no converger module available in Make. At the moment, it is just a concept that provides a counterpart to the Router module, allowing you to reduce duplication of modules in different routes.

Solution

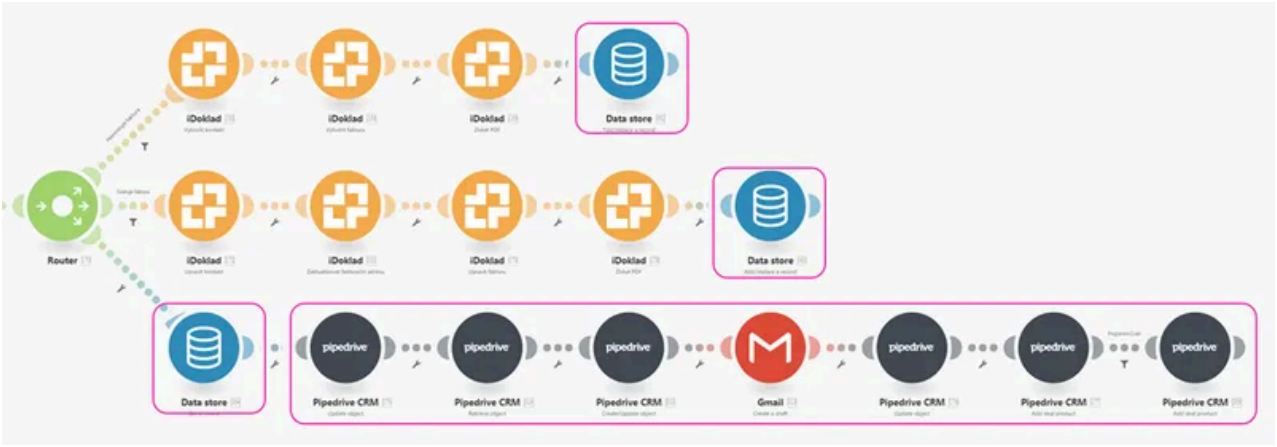
To implement the converger concept, use one of the following workarounds to avoid the duplication of the common sequence.

Data store

1. Add an extra filter-free route to the router to connect the common sequence (the one you would put after a converger module).

2. Add **Data store > Add/replace a record** modules at the end of each router's route (except the new extra route) to store the data output by the modules on the route that should be passed to the common sequence. The data store would contain just one record, the record's key can be, for example, `MyKey`.

3. Add a **Data store > Get a record** module at the beginning of the common sequence to obtain the previously stored data.

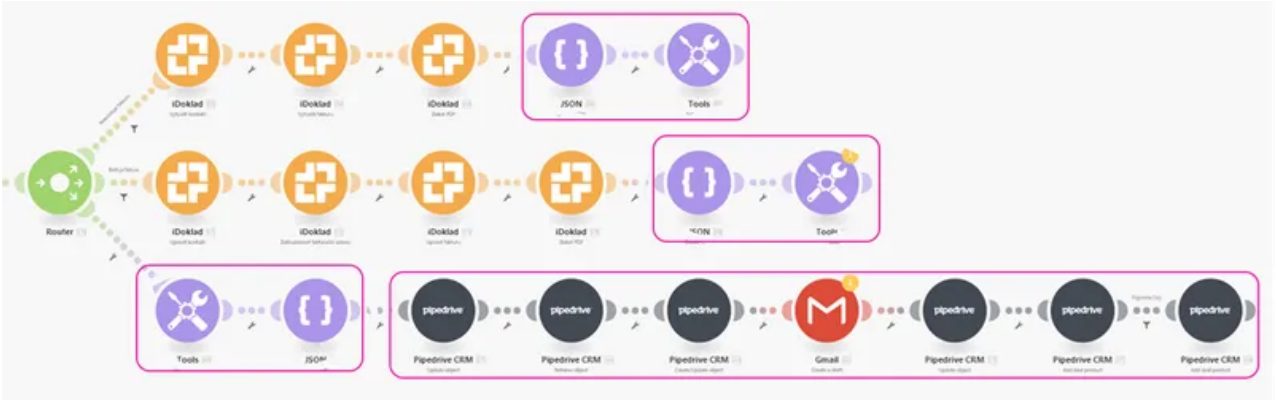


JSON

If you wish to avoid the use of the data store you can:

1. Use a **JSON > Create JSON** module followed by **Tools > Set variable** to store the resulting JSON in a variable (e.g. `MyBundle`).

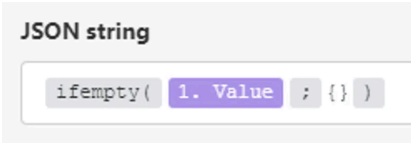
2. Use **Tools > Get variable** to obtain the previously stored variable followed by **JSON > Parse JSON**.



Please note that if no route is executed, the **JSON > Parse JSON** will throw the following error:

BundleValidationError
Validation failed for 1 parameter(s).
> Missing value of required parameter 'JSON string'.

To avoid this, you may employ the `ifemtpy()` function as show below:



Set a variable

If there is a single value that you need to pass to the common sequence (e.g. ID), you can use the **Tools > Set variable** and **Get variable** modules.

Make also allows you to use the **Set multiple variables** module.

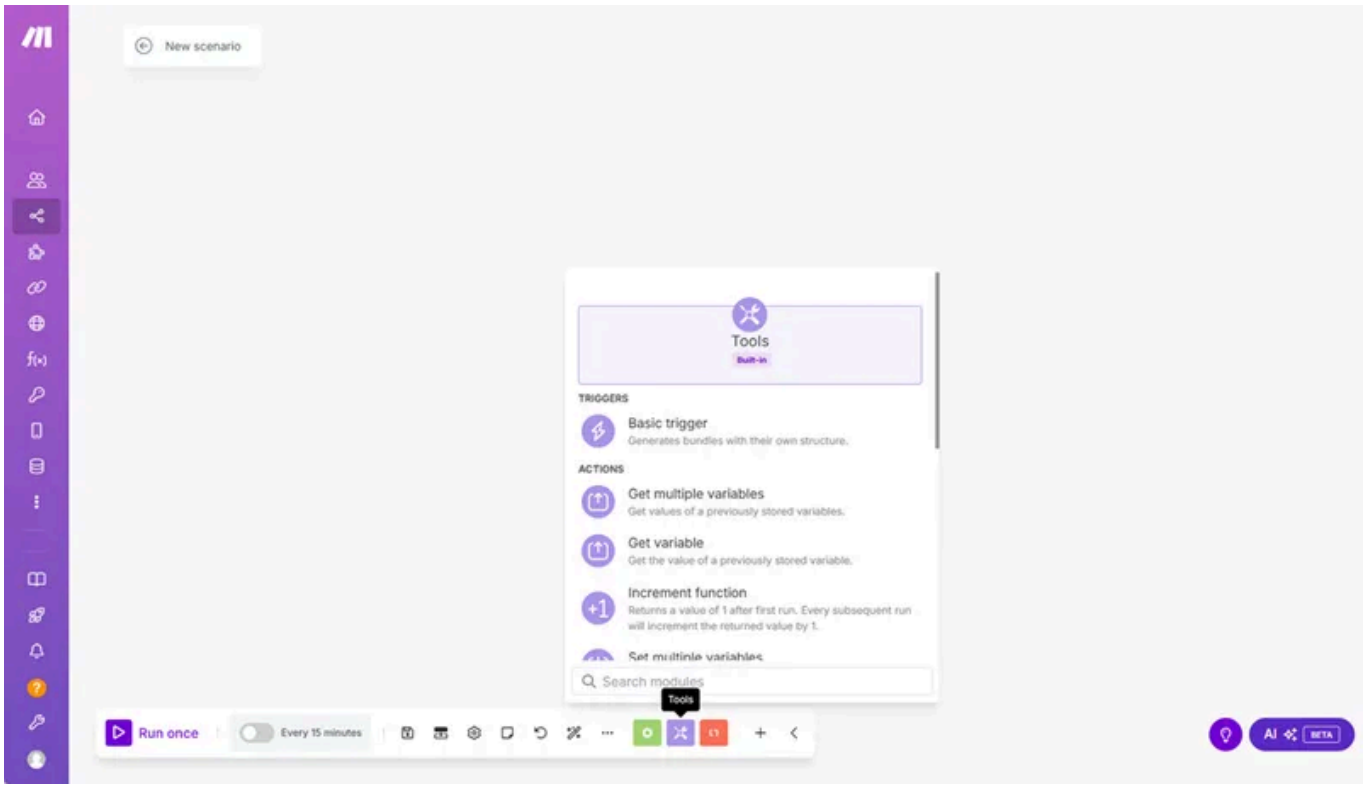
Use a separate new scenario

You may also place the common sequence to a separate new scenario and then:

1. Use the **HTTP > Make a request** module at the end of each route to pass the data to the new scenario.
2. Use the **Webhooks > Custom webhook** module at the beginning of the new scenario to receive the data.

3.3.3. Tools

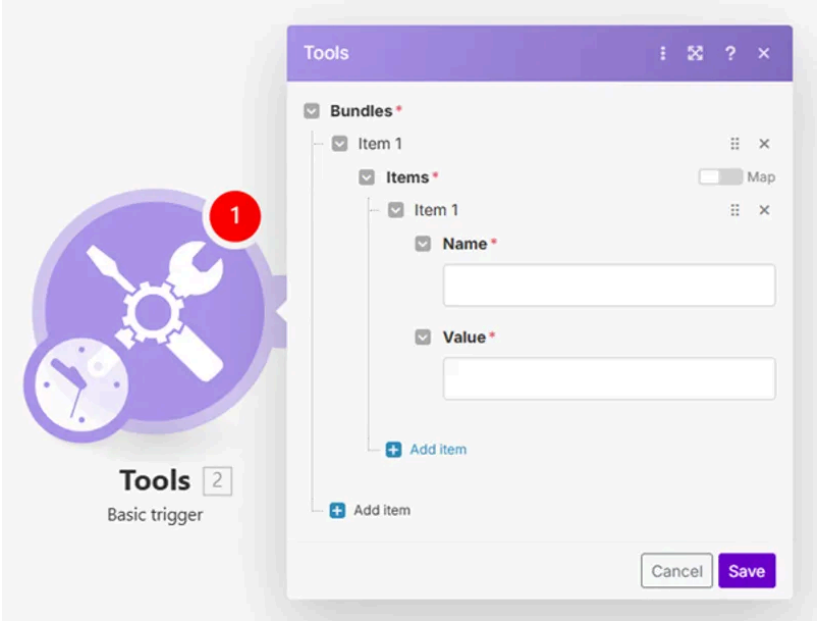
Our Tools section includes several useful modules that can enhance your scenario.



Triggers

Basic trigger

Allows you to create a custom trigger and define its input bundles.



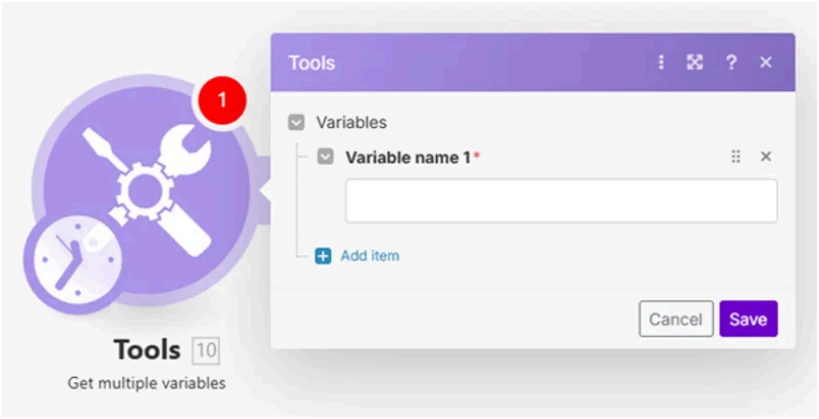
Create custom bundles by adding array items. The array consists of the *name-value* pairs.

For example, you can use this tool for contacts or any other list that is scheduled to be sent to a specified email address (**Email > Send an Email**, **Gmail > Send an Email** modules), or as a simple reminder to be triggered whenever it is necessary.

Actions

Get multiple variables

Retrieves values that were previously created by the **Set multiple variables** module within a single operation.



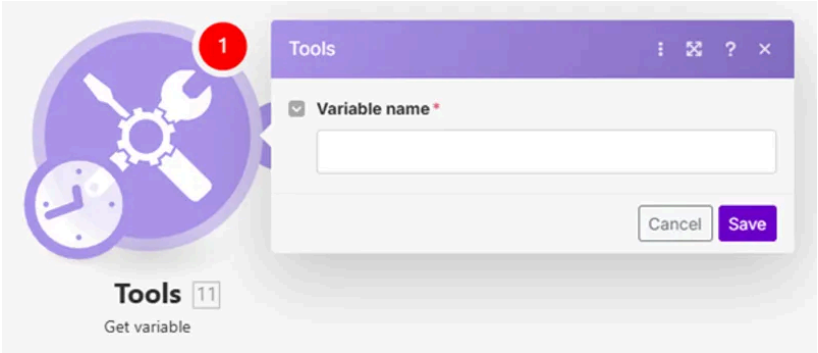
The main benefits of the **Set multiple variables** module are:

- one **Get multiple variables** module can replace a whole series of **Get variable** modules.

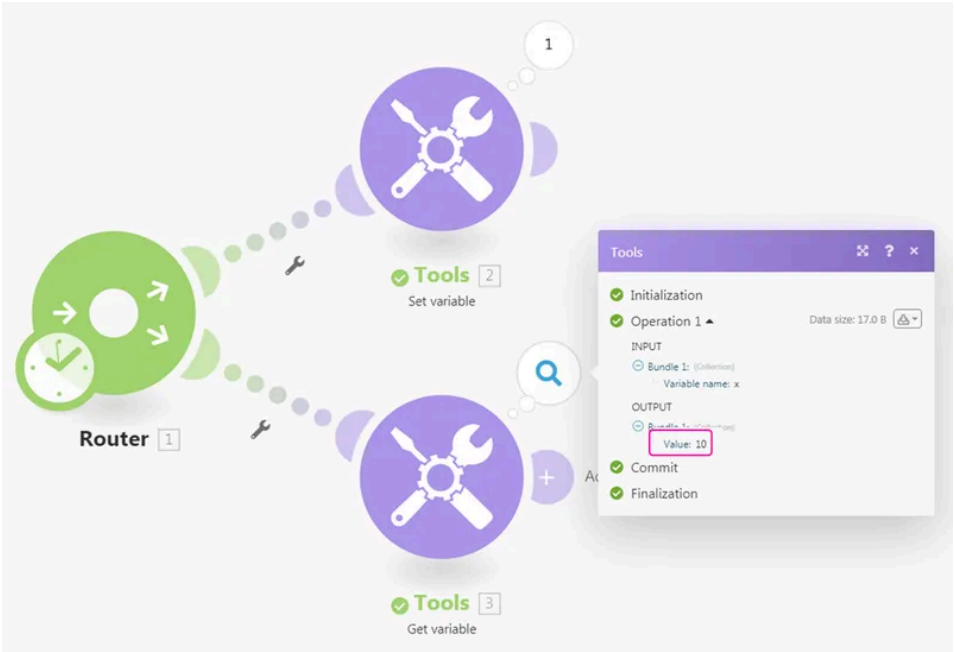
- one **Get multiple variables** module consumes just a single operation.

Get variable

Retrieves a value that was previously created by the **Set variable** module.



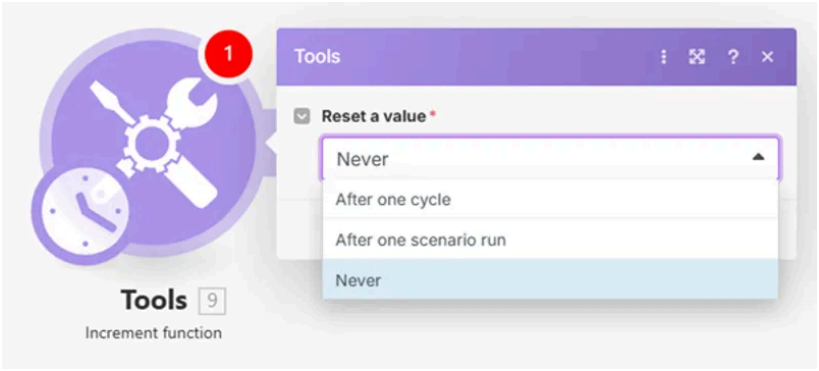
Note that this module can read a variable that was set anywhere in the scenario. The only requirement is that the **Tools > Set Variable** module is executed before (in time) the **Tools > Get Variable** module. See the documentation for the **Router** module for information about the order in which routes are processed.



Increment function

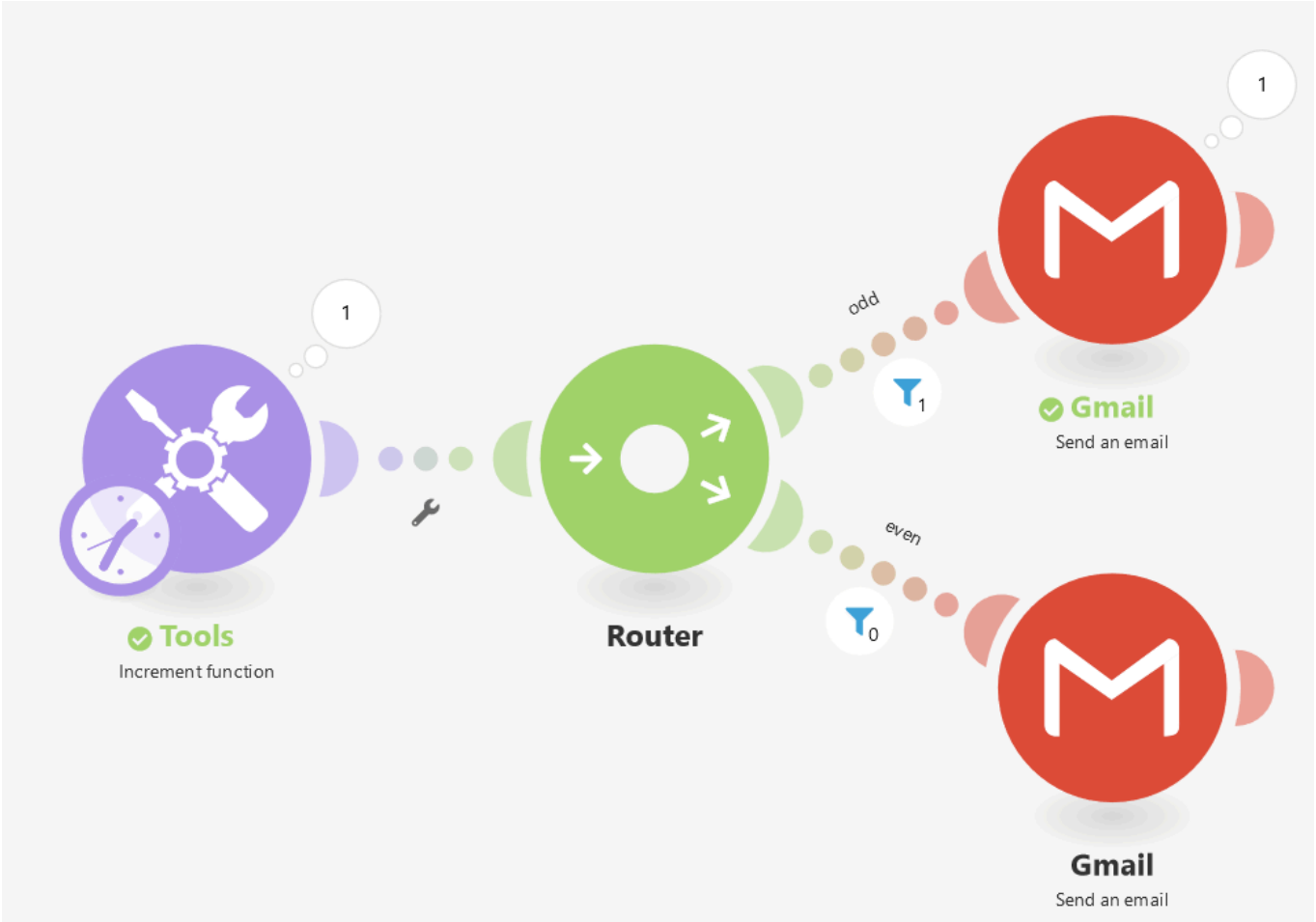
> See our template **Controlled distribution of data flow** for an example of the increment function tool.

Returns a value incremented by 1 after each module's operation. It is possible to configure the module to reset the value.

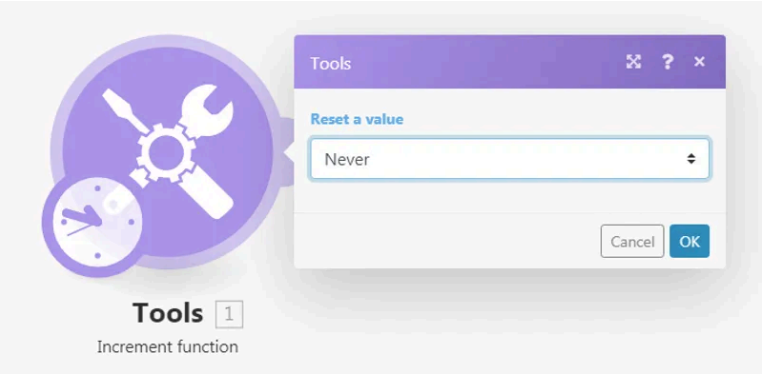


One of this tool's uses is to implement a round robin assignment of tasks to users in a group. The algorithm chooses the assignees from a group in some rational order, usually going from the top to the bottom of a list and then repeating until finished (like you would deal a deck of cards).

The following scenario sends an email to the first recipient after every odd scenario run, and to the second recipient after every even scenario run.



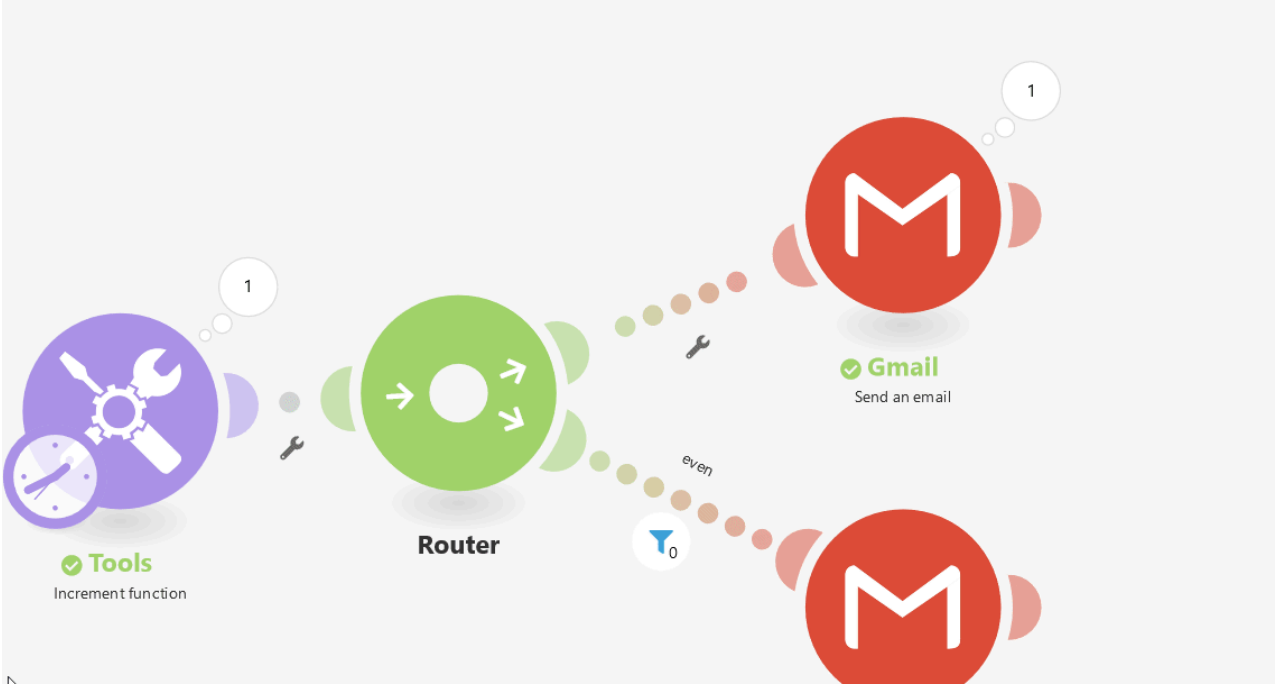
1 Configure the module to never reset the value:



2 Add a router.

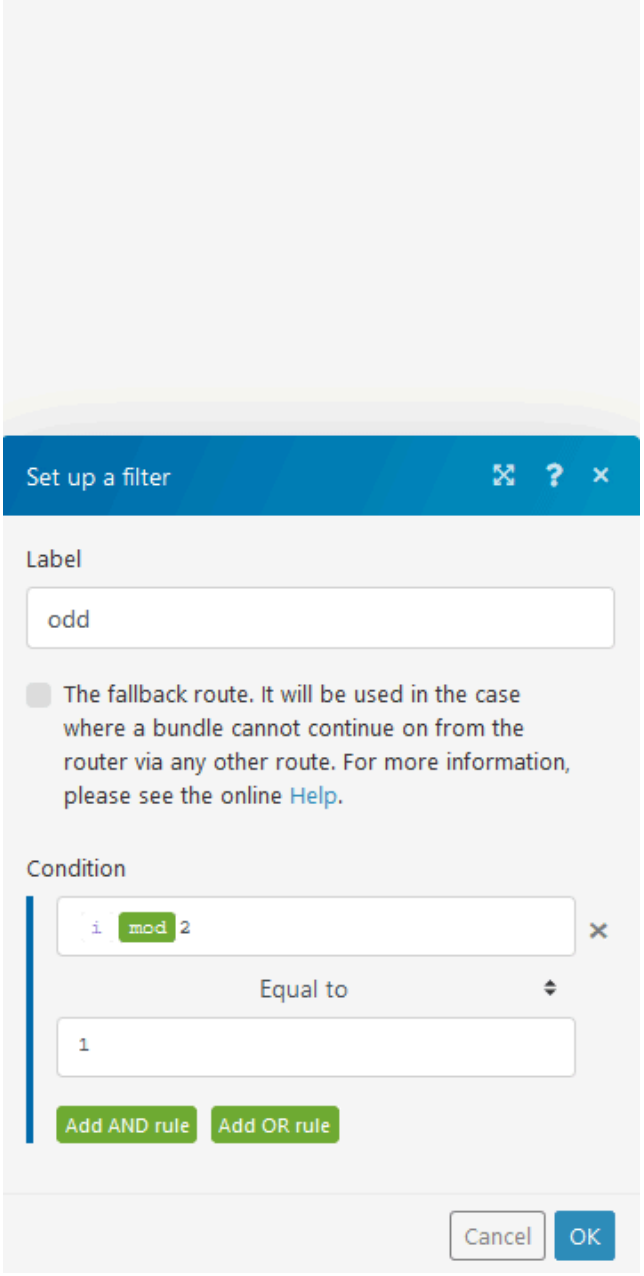
3 Set the first condition after the router:

Odd - use the `mod` math function set equal to `1`.



61d5a3ca16a69.gif

Do not forget to change the **Equal to** operator from the default *Text operator* to the **Numeric operator.**



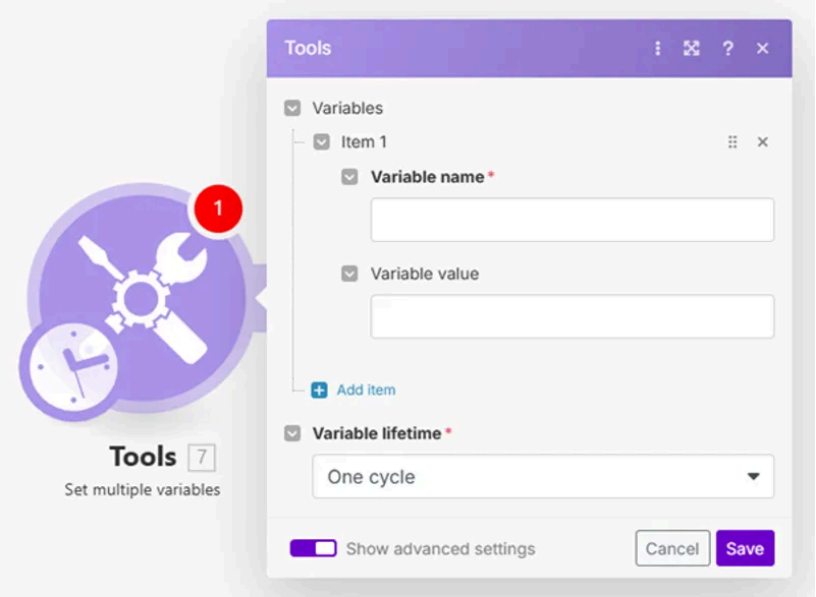
4 Set the second condition after the router.

Even – use the `mod` math function that equals `0`:



Set multiple variables

Creates multiple variables that can be mapped by other modules in the route or by the **Get Multiple Variables** module for every route in the scenario **within a single operation**.



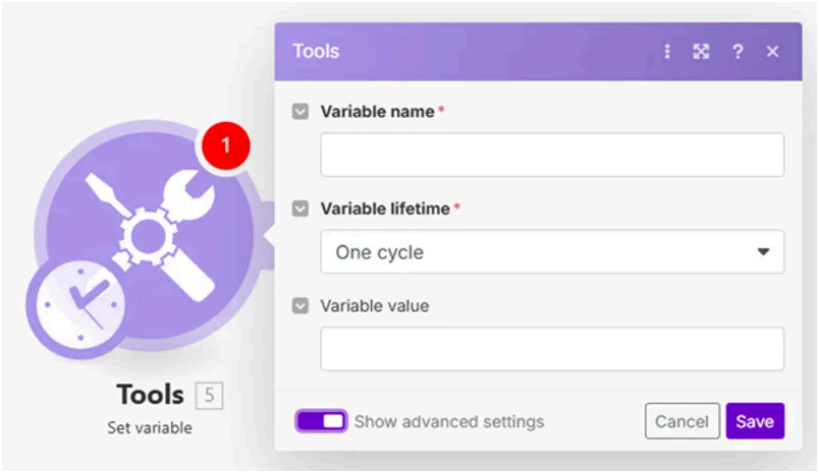
The main benefits of the **Set multiple variables** module are:

- one **Set multiple variables** module can replace a whole series of **Set variable** modules
- one **Set multiple variables** module consumes just a single operation

Variables	Add multiple variables you want to set. • Variable name: Enter the variable name. This name will be displayed when mapping the variable in other modules. • Variable value: Enter the value of the variable.
Variable lifetime	One cycle : the variable is valid only for one cycle. Useful when multiple webhooks in one scenario run are received (more webhooks = more cycles). One execution: The variable is valid for one execution. One execution can contain more cycles.

Set variable

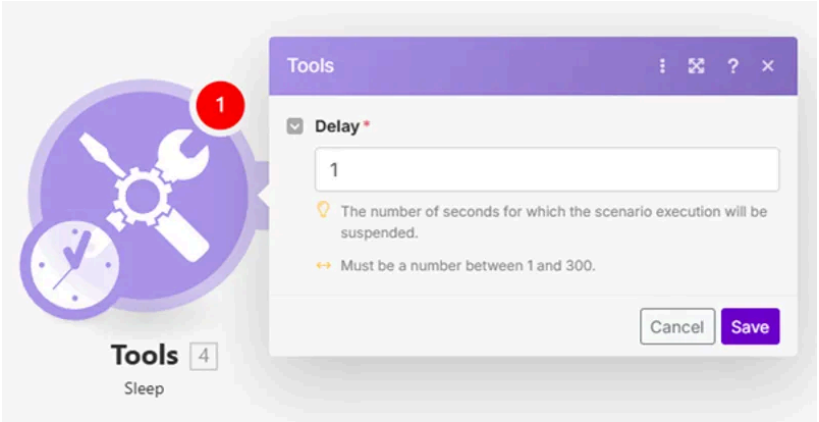
Creates a variable that can be mapped by other modules in the route or by the **Get variable** module for every route in the scenario.



Field	Description
Variable name	Enter the variable name. This name will be displayed when mapping the variable in other modules.
Variable lifetime	One cycle : the variable is valid only for one cycle. Useful when multiple webhooks in one scenario run are received (more webhooks = more cycles). One execution: The variable is valid for one execution. One execution can contain more cycles.
Variable value	Enter the value of the variable.

Sleep

Allows you to delay the scenario flow for up to 300 seconds (5 minutes).



This function can be useful, for example, if you want to lower the target service server load or to simulate more human behavior when sending bulk SMS or emails.

If you wish to pause the flow for longer periods of time, we suggest splitting your scenario into two scenarios.

1. The first scenario would contain the part before the pause.
2. The second scenario would contain the part after it.

The first scenario would store all the necessary information in a data store together with the current timestamp. The second scenario would periodically check the data store for

records with a timestamp older than the intended delay, retrieve the records, finalize the processing of the data, and remove the records from the data store.

Aggregators

Numeric aggregator

Allows you to retrieve numerical values, then apply one of the selected functions (SUM, AVG, COUNT, MAX,...), and return the result in one bundle.

In this example, the module sums up values under the *number* parameter.

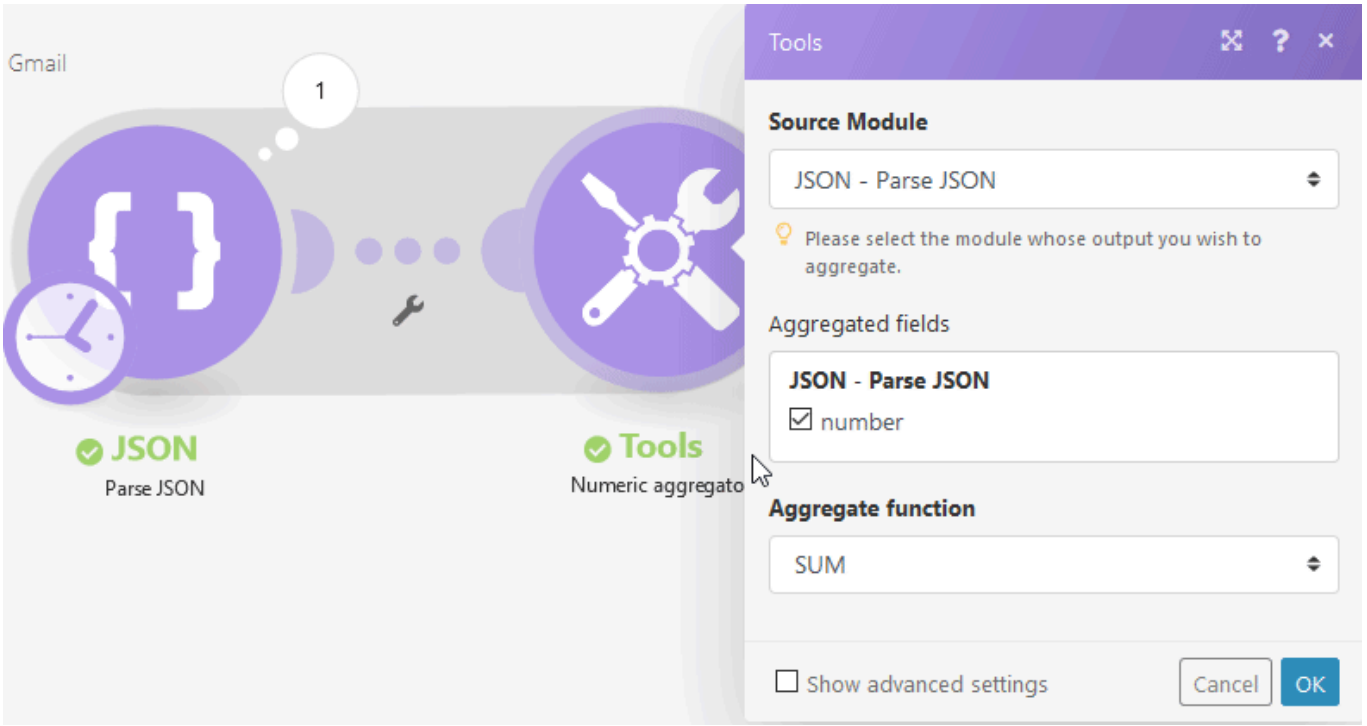
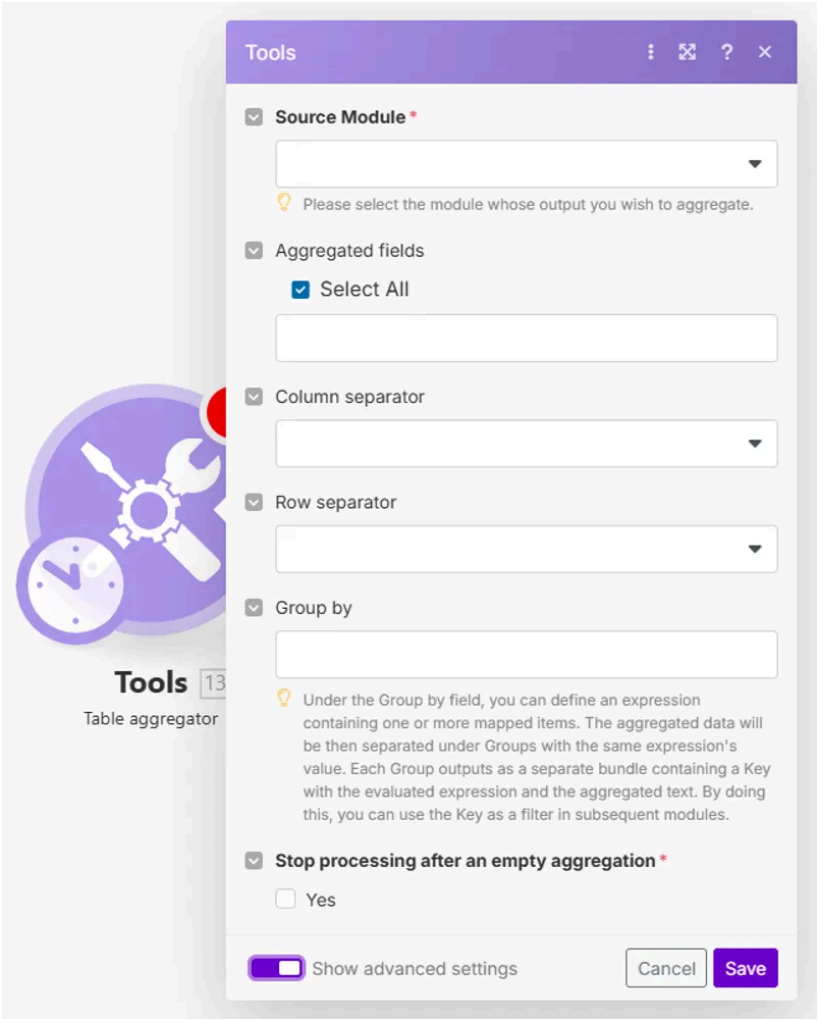


Table aggregator

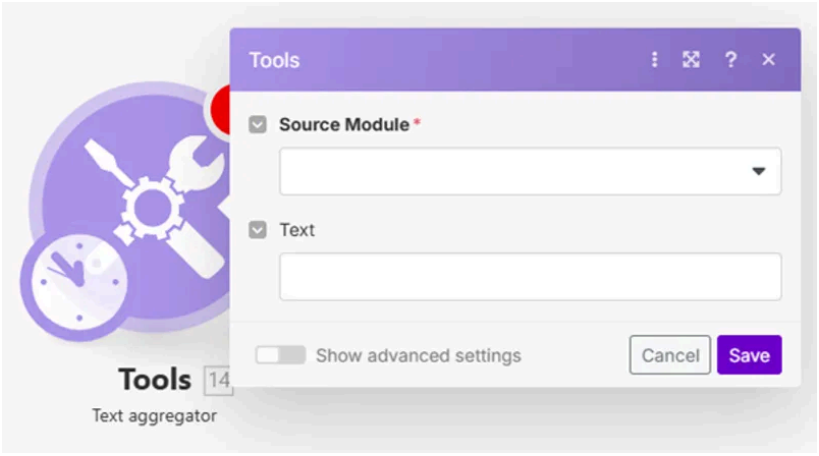
Merges values from the selected fields of received bundles into a single bundle using a specified column and row separator, allowing you to create a table.



Field	Description
Source module	Select the module you want to aggregate fields from.
Aggregated fields	Select the fields from the module selected above whose values you want to aggregate into one bundle.
Column separator	Select or enter the type of separator that will separate the field value columns in the resulting bundle.
Row separator	Select or enter the type of separator that will separate the field value rows in the resulting bundle.
Group by	Define an expression containing one or more mapped items. The aggregated data will then be separated into groups with the same expression's value. Each group outputs a separate bundle containing a key with the evaluated expression and the aggregated text. By doing this, you can use the key as a filter in subsequent modules.

Text aggregator

Merges values from the selected fields of received bundles into a single bundle.

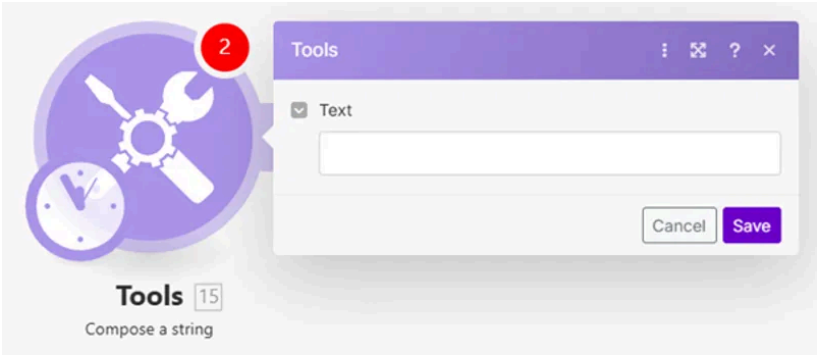


You can use the text aggregator tool to insert more values (e.g. customer names or notes) into a single bundle and send an email containing all the values in the email body or the email subject.

Transformers

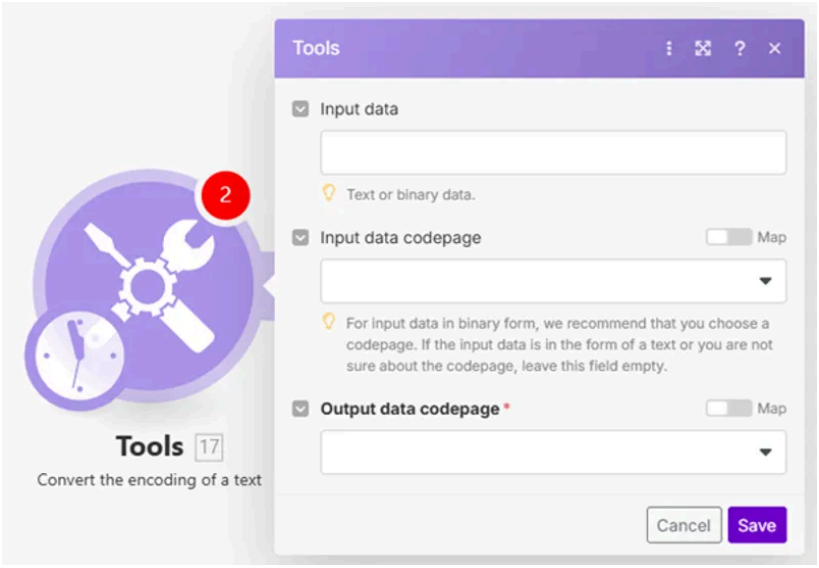
Compose a string

Converts any value to a string data type (text), making the mapping easier when mapping, for example, binary data.



Convert the encoding of the text

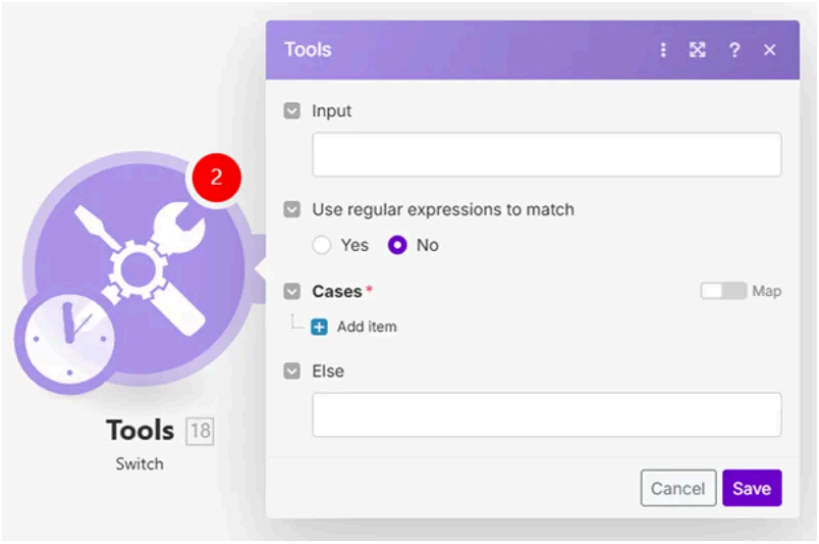
Converts entered input text (or binary data) to the selected encoding.



Field	Description
Input data	Enter the content you want to convert.
Input data codepage	Enter the input data encoding type. This is important for the binary form of data.
Output data codepage	Select the target encoding of your data.

Switch

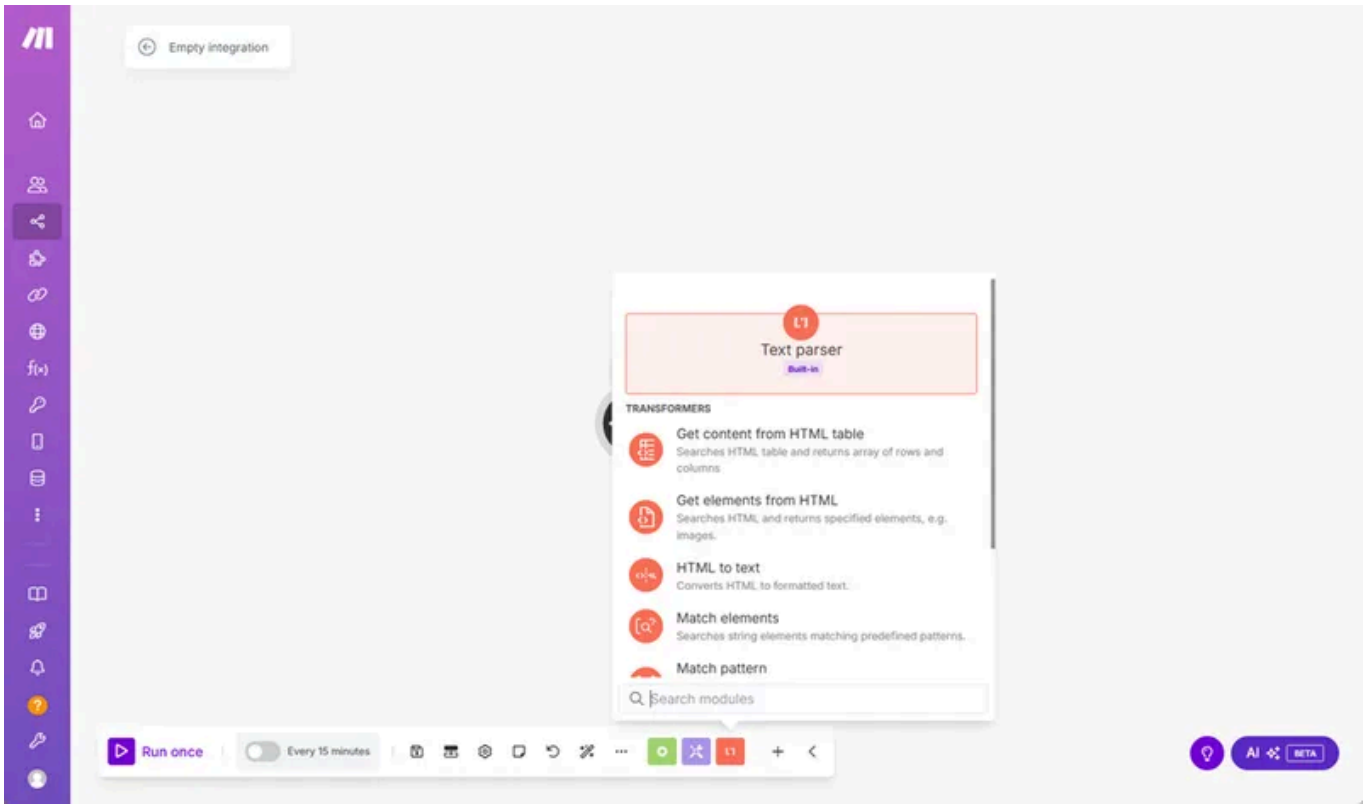
Checks the input value for a match with the provided list of values. Returns output based on the result.



Field	Description
Input	Enter the expression you want to evaluate.
Cases	If the **Input** contains a value entered to the **Pattern** field, then the value entered to the **Output** field is returned. If the condition is not met, then no output is returned OR the value from the **Else** field (below) is returned. When using regex patterns, make sure to adhere to the ECMAScript flavor.
Else	Enter the value that is returned when the criteria set in the **Cases** field is not met.

3.3.4. Text parser

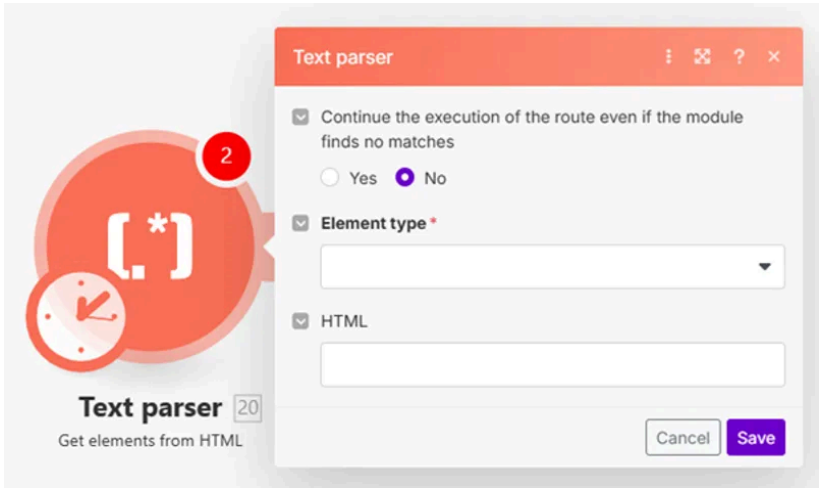
Our Text parser section includes several useful modules that can enhance your scenario.



Transformers

Get elements from HTML

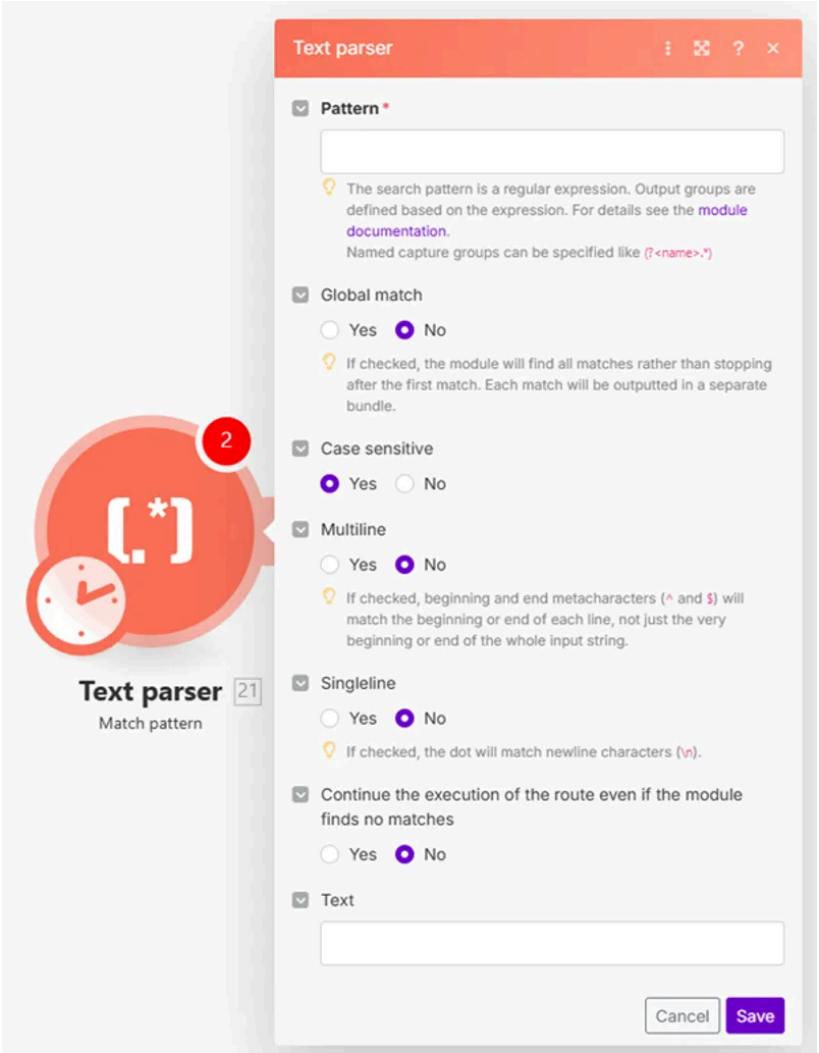
Retrieves the desired elements from an HTML code.



Field	Description
Continue the execution of the route even if the module returns no results	If enabled, the scenario will not be stopped by this module.
Element type	Select the type of element you want to retrieve from the HTML code such as an image, link, or iframe element(s).
HTML	Enter the HTML code you want to retrieve the specified element types from.

Match pattern

Enables you to find and extract string elements matching a search pattern from a given text.



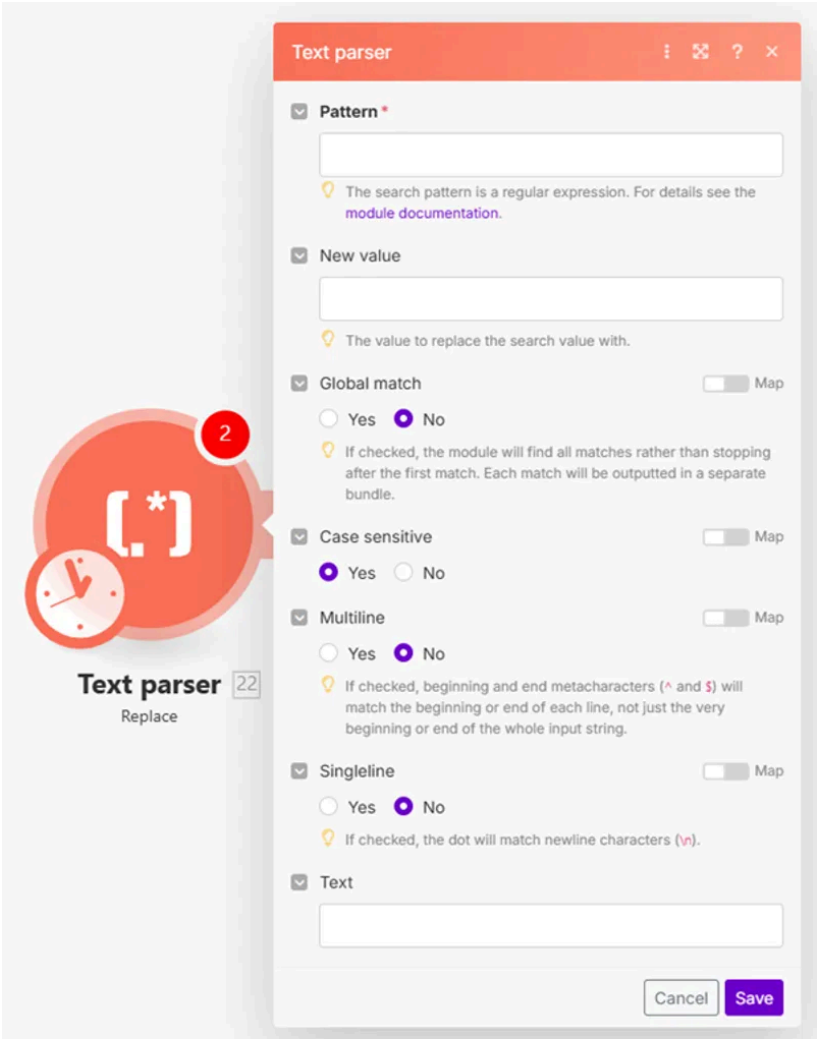
Field	Description
Pattern	Enter the regular expression pattern. For example, `[+-]?(\d+(\.\d+)?|\.\d+)([eE][+-]?\d+)?` extracts all numerals in the provided text. The pattern will contain at least one capture group in parenthesis `()` for the output bundle to contain some items. If the pattern does not contain any capture groups, the output bundle will be empty
Global match	If enabled, then the module retrieves all matches in the text. If disabled, then the module retrieves only the first entry.
Case sensitive	You can disable the case sensitivity by disabling this option (default=case sensitive).
Multiline	If checked, beginning and end metacharacters (`^` and `$`) will match the beginning or end of each line, not just the very beginning or end of the whole input string.
Continue the execution of the route even if the module returns no results	If enabled, the scenario will not be stopped by this module.
Text	Enter the text you want to match the pattern.

The search pattern is a **regular expression** (aka regex or regexp), which is a sequence of characters in which each character is either a metacharacter, having a special meaning, or a regular character that has a literal meaning.

- The complete list of metacharacters can be found on the **MDN web docs website**.
- For a tutorial on how to create regular expressions, we recommend the **RegexOne website**.
- For an easy, quick regex generator, try the **Regular Expressions generator**.
- For experimenting with regular expressions, we recommend the **regular expressions 101 website.** Just make sure to tick the ECMAScript (JavaScript) FLAVOR in the left panel:

Replace

Searches the entered text for a specified value or regular expression, and replaces the result with the new value.



Field	Description
Pattern	Enter the search term. You can also use a regular expression. For more details about the regular expression, refer to the **Match pattern** module.
New value	Enter a value that will replace the search term.
Global match	If this option is enabled, the module will find all matches rather than stopping after the first match. Each match will be output in a separate bundle.
Case sensitive	If this option is enabled, the search will be case sensitive.
Multiline	If checked, the beginning and end metacharacters (`^` and `$`) will match the beginning or end of each line, not just the very beginning or end of the whole input string.
Text	Enter the text to be searched.

Data scraping

Data scraping, sometimes called web scraping, data extraction, or web harvesting is the process of collecting data from websites and storing it in your local database or spreadsheets. If you wish to scrape data from a website and you are not familiar with regular expressions, you may use a data scraping tool:

- **Apify** is an excellent tool, and we already have it integrated
- **Web Data Extractors 2025**

If the data scraping tool provides a REST API, you can connect to it via our universal **HTTP** and **Webhooks** modules. You can also create an app on your own using the **Make App SDK**.

3.4. Functions

Transform and format data using our range of functions:

→ **Use functions**

→ **General functions**

→ **Math functions**

→ **Math variables**

→ **String functions**

→ **Date and time functions**

→ **Tokens for date/time parsing**

→ **Tokens for date/time formatting**

→ **Array functions**

→ **Custom functions**

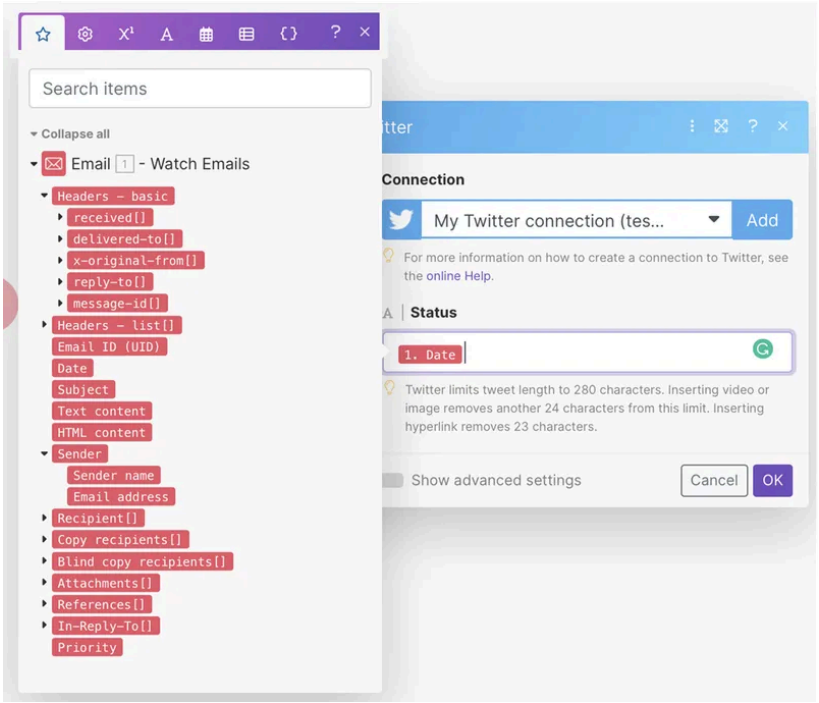
3.4.1. Use functions

When **mapping items**, you can use functions to create complex formulas. The functions available in Make are similar to functions in Excel or those in other programming languages. The functions let you perform various transformations of item values, such as converting a text to uppercase, trimming a text, converting a date into a different format, and many others.

Insert functions into fields

Functions can be inserted into fields the same way as items. Click a field to view the mapping panel.

The mapping panel contains several tabs:



The first tab (shown upon opening the panel) displays the items that you can map from other modules.

The other tabs contain various functions:

- **General functions**
- **Math functions**

- **String functions**
- **Date and time functions**
- **Array functions**
- **Custom functions**
- **Custom and system variables**

To insert a function into a field, click the function name or drag it to the field.

Example

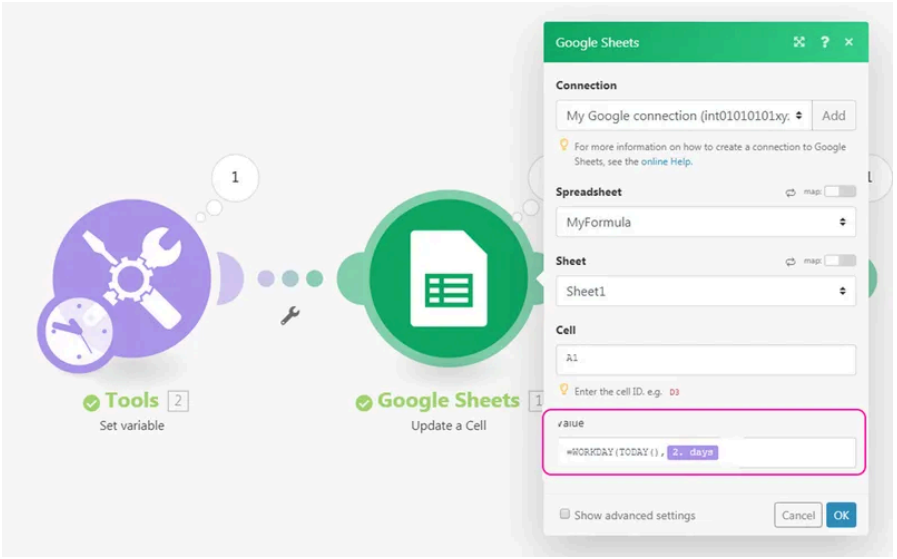
When posting a new tweet with the **Twitter > Create a tweet** module, you should not exceed Twitter's 280-character limit. Otherwise, the module throws an error. If you map a text item that can contain more than 280 characters into the module's **Status** field, you can use the `substring()` function to limit the tweet to 280 characters:



Google Sheets functions

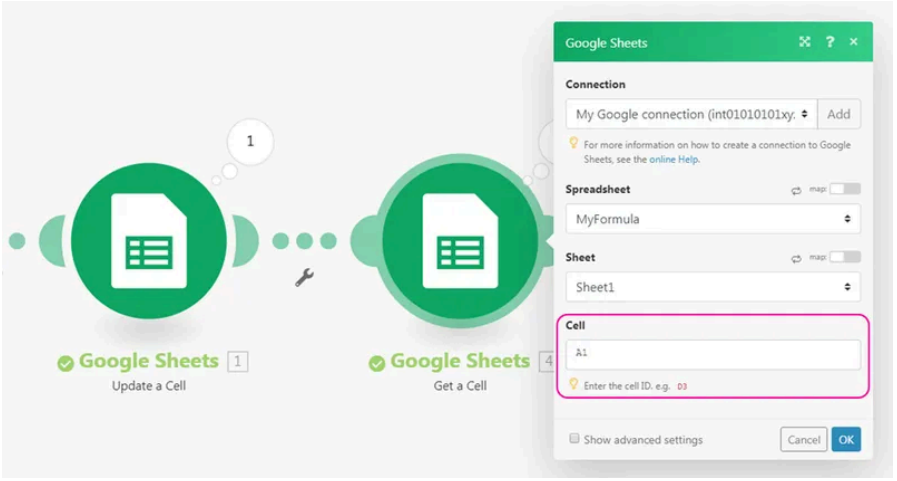
If you miss a function, but it is featured by **Google Sheets**, you may use it like this:

1. In Google Sheets, create a new empty spreadsheet.
2. In Make open your scenario.
3. Insert the **Google Sheets > Update a cell** module.
4. Configure the module:
    1. Choose the newly created spreadsheet in the **Spreadsheet** field.
    2. Instert your formula containing the Google Sheets function into the Value field.



5 Insert the **Google Sheets > Get a cell** module to obtain the calculated result.

6 Configure the module. Fill in the same **Cell** ID as you filled in step 4.



3.4.2. General functions

General functions let you fine-tune your mapping in a scenario. For example, you can select or leave out items in an array or apply conditional criteria to your mapping. Below is a list of general functions you can use.

get

get (object or array; path)

Returns the value path of an object or array. To access nested objects, use dot notation. The first item in an array is index 1.

Text

```
get(array;1+1)
```

Text

```
get(array;5.raw_name)
```

Text

```
get(object;raw_name)
```

Text

```
get(object;raw_name.sub_raw_name)
```

if

if (expression; value1; value2)

Returns value 1 if the expression is evaluated to be true. Otherwise, it returns value 2.

```Text
if(1=1;A;B)
= A
```

```Text
if(1=2;A;B)
= B
```

ifempty

ifempty (value1; value2)

Returns value 1 if this value is not empty. Otherwise, it returns value 2.

```Text
ifempty(A;B)
= A
```

```Text
ifempty(unknown;B)
= B
```

```Text
ifempty("";B)
= B
```

switch

switch (expression; value1; result1; [value2; result2; ...]; [else])

Evaluates one value (called the expression) against a list of values, and returns the result corresponding to the first matching value.

```
switch(B;A;1;B;2;C;3)
= 2
```

```
switch(C;A;1;B;2;C;3)
= 3
```

```
switch(X;A;1;B;2;C;3;4)
= 4
```

omit

omit (collection; key1; [key2; ...])

Removes elements with the given keys from a collection. Returns a collection with the remaining elements. Useful when you want to pass a collection to an API, which expects an exact number of elements in the collection. Using this function, you can make sure your collection meets the requirements of the API.

pick

pick (collection; key1; [key2; ...])

Picks elements with the given keys from a collection. Returns a collection that contains only the picked elements. Useful when you want to pass a collection to an API, which

expects an exact number of elements in the collection. Using this function, you can make sure your collection meets the requirements of the API.

3.4.3. Math functions

Use a math function to apply mathematical operations to data. For example, math functions let you get an average of an array, round a value up or down, or change the format of a number (1 000 000 to 1,000,000 or 1.000.000). Below is a list of supported math functions and a brief description of each one.

abs

abs (number)

Returns the absolute value of an integer.

Text

```
abs( -5 ) = 5
```

Text

```
abs( 5 ) = 5
```

Text

```
abs( 0 ) = 0
```

Text

```
abs( -3.7 ) = 3.7
```

average

average ([array of values]) average(value1; [value2], ...)

Returns the average value of the numeric values in a specific array, or the average value of numerical values entered individually.

ceil

ceil (number)

Returns the smallest integer greater than or equal to a specified number.

```Text
ceil( 1.2 ) = 2
```

```Text
ceil ( 4 ) = 4
```

floor

floor (number)

Returns the largest integer less than or equal to a specified number.

```Text
floor( 1.2 ) = 1
```

```Text
floor( 1.9 ) = 1
```

```Text
floor( 4 ) = 4
```

formatNumber

formatNumber (number; decimalPOINTS; [decimalSeparator]; [thousandsSeparator])

Returns a number in the requested format. Decimal point is `,` by default, Thousands separator is `.` by default.

Text

```
formatNumber( 123456789 ; 3 ; , ; . ) = 123.456.789,000
```

max

max ([array of values]), max(value1;value2; ...)

Returns the largest number in a specified array, or the largest number among numbers entered individually.

median

median ([array of values])

Returns the median of the values in a specified array, or the median of numbers entered individually.

Text

```
median( 3; 5; 7 ) = 5
```

Text

```
median( 2; 3; 5; 8 ) = 4
```

Text

```
median( 2.5; 3.5; 2; 4.5; 1 ) = 2.5
```

min

min ([array of values]), min(value1;value2; ...)

Returns the smallest number in a specified array, or the smallest number among numbers entered individually.

parseNumber

parseNumber (number; decimal separator)

Parses a string with a number and returns the number.

Example: `parseNumber(` 1 756,456 `;` , `)`

round

round (number)

Rounds a numeric value to the nearest integer.

Text

```
round( 1.2 ) = 1
```

Text

```
round( 1.5 ) = 2
```

Text

```
round( 1.7 ) = 2
```

Text

```
round( 2 ) = 2
```

stdevP

stdevP ([array of values])

Returns the standard deviation of a specified array of population values, or the standard deviation of numbers entered individually.

Text

```
stdevP( 1; 2; 3; 4; 5) = 1.4142135623730951
```

Text

```
stdevP( {{array}} )
```

stdevS

stdevS ([array of values])

Returns the standard deviation of a specified array of sample values, or the standard deviation of numbers entered individually.

Text

```
stdevS( 1; 2; 3; 4; 5) = 1.5811388300841898
```

Text

```
stdevS( {{array}} )
```

sum

sum ([array of values]), sum(value1;value2; ...)

Returns the sum of the values in a specified array, or the sum of numbers entered individually.

trunc

trunc (number)

Truncates a number to an integer by removing the fractional part of the number.

Text

```
trunc( 3.789) = 3
```

Text

```
trunc( 3.789; 2) = 3.78
```

Text

```
trunc( -3.789; 2 ) = -3.7
```

Text

```
trunc( 123.456; -2 ) = 100
```

3.4.4. Math variables

Use math variables to insert a random number into your functions.

random

Returns a floating-point, pseudo-random number in the range [`0` , `1`) (inclusive of `0` , but not `1`).

Use the following formula to generate an integer pseudo-random number in the range [`min` , `max`] (inclusive of both, `min` and `max`):



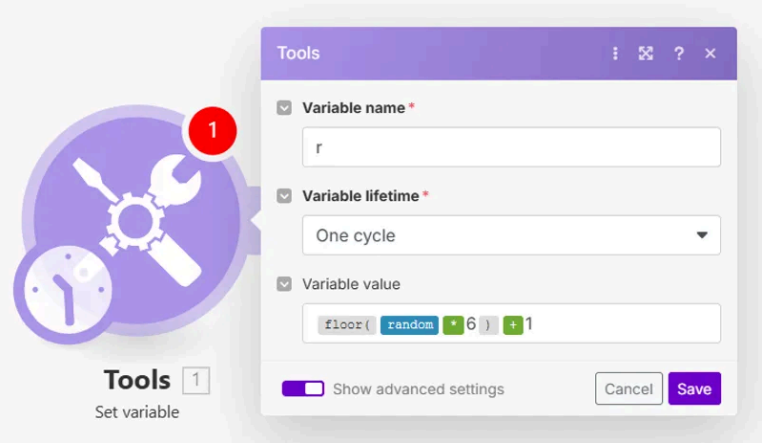
Text

```
{{floor(random * (1.max - 1.min + 1)) + 1.min}}
```

Example: Roll the dice

You can employ the `random` variable to create a dice game that randomly picks a number, for example between 1 and 6, and then returns that number to a user.

1. In the mapping panel of the **Tools > Set variable** module, select the **Math functions** tab.



2 Select the `floor` function.

3 Insert the following:

Text

```
{{floor(random * 6) + 1}}
```

3.4.5. String functions

String functions let you modify and transform string data, also known as text. Use a string function to, for example, change capitalization, remove diacritics and accent marks, combine, split, encode, or decode text. Below is a list of supported string functions and a description of each.

ascii

ascii (text; [remove diacritics])

Removes all non-ascii characters from a text string.

Text

```
ascii( ěMščařžkýáeíé)
= Make
```

Text

```
ascii( ěščřž ; true )
= escrz
```

base64

base64 (text)

Transforms text to base64.

Text

```
base64( Make )
=TWFrZQ==
```

Combine the `toBinary()` and `toString()` functions to transform base64 encoded text to plain text, as seen below:

Text

```
toString( toBinary( TWFrZQ== ; base64 ) )
= Make
```

capitalize

capitalize (text)

Converts the first character in a text string to uppercase.

Text

```
capitalize(make)
= Make
```

contains

contains (text; search string)

Verifies if text contains the search string.

Text

```
contains( Hello World ; Hello )
= true
```

Text

```
contains( Hello World ; Bye )
= false
```

decodeURL

decodeURL (text)

Decodes special characters in URL to text.

Text

```
decodeURL( Automate%20your%20workflow )
= Automate your workflow
```

encodeURL

encodeURL (text)

Encodes special characters in a text to a valid URL address.

escapeHTML

escapeHTML (text)

Escapes all HTML tags in text.

Text

```
escapeHTML( <b>Hello</b> )
= &lt;b&gt;Hello&lt;/b&gt;
```

indexOf

indexOf (string; value; [start])

Returns the position of the first occurrence of a specified value in a string. This method returns '-1' if the value searched for never occurs.

Text

```
indexOf(Make; k )
= 2
```

Text

```
indexOf(We love Make ; x )
= -1
```

Text

```
indexOf(We love Make ; e ; 7 )
= 11
```

length

length (text or buffer)

Returns the length of text string (number of characters) or binary buffer (buffer size in bytes).

Text

```
length( hello )
= 5
```

lower

lower (text)

Converts all alphabetical characters in a text string to lowercase.

Text

```
lower( Hello )
= hello
```

md5

md5 (text)

Calculates the md5 hash of a string.

Text

```
md5(Make)
= 529a05ca6c1263aab080ec4f20754411
```

replace (text; search string; replacement string)

Replaces the search string with the new string.

Text

```
replace( Hello World ; Hello ; Hi )
= Hi World
```

Regular expressions (enclosed in `/.../`) can be used as search string with a combination of flags (like `g`, `i`, `m`) appended:

Text

```
replace ( All these numbers 187 23 1 9565 will be replaced with X ; /\d+g ; X
= All these numbers X X X X will be replaced with X
```

The replacement string can include the following special replacement patterns:

- `$&` = Inserts the matched substring

- `$n` = Where n is a positive integer less than 100, inserts the nth parenthesized submatch string. Note that this is 1-indexed

Text

```
replace ( Phone number is 777111222 ; /\d+g ; $& )
Phone number is +420777111222
```

Text

```
replace ( Phone number is 777111222 ; / is (d) ;:  +42$1 )
Phone number is +420777111222
```

> Do not use named capture groups like `/ is (?<number>\d+)/` in the replacement string argument as this will throw an error.

replaceEmojiCharacters

replaceEmojiCharacters (text)

Replaces emoji characters with the new string.

`replaceEmojiCharacters(` Hello, world!🌍🙂 `;` `emptystring` `)`
= Hello, world!

`replaceEmojiCharacters(` Automate🤖your📈workflow🔧easily! `;` `space` `)`
= Automate your workflow easily!

sha1

sha1 (text; [encoding]; [key])

Calculates the sha1 hash of a string. If the key argument is specified, sha1 HMAC hash is returned instead. Supported encodings: *hex* (default), *base64* or *latin1*.

Text

```
sha1(Make)
= a94431ee22f05f141107f9355ed3127d0f0c4d5a
```

sha256

sha256 (text; [encoding]; [key]; [key encoding])

Calculates the sha256 hash of a string. If the key argument is specified, sha256 HMAC hash is returned instead. Supported encodings: *hex* (default), *base64* or *latin1*.

Text

```
sha256(Make)
= ccdd25d4230fb997a7ee1166f8afabd157248eb812ea55ec7c3d1b7c8af7fa11
```

sha512

sha512 (text; [output encoding]; [key]; [key encoding])

Calculates the sha512 hash of a string. If the key argument is specified, sha512 HMAC hash is returned instead. Supported encodings: *hex* (default), *base64* or *latin1*. Supported key encodings: *text* (default), *hex*, *base64* or *binary*. When using *binary* key encoding, a key must be a buffer, not a string.

Text

```
sha512(Make)
= e8000cd0fb8fae18caa8daa677269b0034380d3ec549e0113a0722d8b8dc05b0f7037f33f32
```

split

split (text; separator)

Splits a string into an array of strings by separating the string into substrings.

Text

```
split( John, George, Paul ; , )
=
John
George
Paul
```

startcase

startcase (text)

Capitalizes the first letter of every word and lower cases all other letters.

Text

```
startcase( hello WORLD )
= Hello World
```

stripHTML

stripHTML (text)

Removes all HTML tags from text.

Text

```
stripHTML( <b>Hello</b> )
= Hello
```

substring

substring (text; start; end)

Returns a portion of a text string between the "start" position and "the end" position.

Text

```
substring( Hello ; 0 ; 3 )
= Hel
substring( Hello ; 1 ; 3 )
= el
```

toBinary

toBinary (value)

Converts any value to binary data. You can also specify encoding as a second argument to apply binary conversions from hex or base64 to binary data.

Text

```
toBinary(Make)
= 4d 61 6b 65
```

toString

toString (value)

Converts any value to a string.

Text

```
toString(toBinary(TWFrZQ==;base64)
= Make
```

trim

trim (text)

Removes space characters at the start or end of the text.

upper

upper (text)

Converts all alphabetical characters in a text string to uppercase.

Text

```
upper( Hello )
= HELLO
```

3.4.6. Date and time functions

Use date and time functions to convert and transform date and time data. For example, you can change the date format, convert time based on timezones, convert text to date or time data, and more. Below is a list of supported date and time functions with descriptions and details for each.

formatDate (date; format; [timezone])

When to use it: You have a **date** value that you wish to convert (format) to a **text** value (textual human-readable representation) like `12-10-2019 20:30` or `Aug 18, 2019 10:00 AM`

Parameters

The second column indicates the expected type. If different type is provided, **type coercion** is applied.

Parameter	Expected type	Description
date	date	Date value to be converted to a text value.
format	text	Format specified using **tokens for date/time formatting**. Example: `DD.MM.YYYY HH:mm`
timezone	text	Optional. The timezone used for the conversion. See **List of tz database time zones**, column "TZ database name" for the list of recognized timezones. If omitted, Make uses the organization's timezone. You can **edit your time zone**.

Return value and type

Text representation of the given Date value according to the specified format and timezone. Type is Text.

Example

The Organization's and Web's timezone were both set to `Europe/Prague` in the following examples.

Function	Result
`formatDate(1. Date created;` MM/DD/YYYY `)`	10/01/2018
`formatDate(1. Date created;` YYYY-MM-DD hh:mm A `)`	2018-10-01 09:32 AM
`formatDate(1. Date created;` DD.MM.YYYY HH:mm `;` UTC `)`	01.10.2018 07:32
`formatDate(now;` MM/DD/YYYY HH:mm `)`	19/03/2019 15:30

parseDate (text; format; [timezone])

When to use it: You have a **text** value representing a date (e.g. `12-10-2019 20:30` or `Aug 18, 2019 10:00 AM`) and you wish to convert (parse) it to a **Date** value (binary machine-readable representation).

Parameters

The second column indicates the expected type. If different type is provided, **Type Coercion** is applied.

Parameter	Expected type	Description
text	text	Text value to be converted to a date value.
format	text	Format specified using **tokens for date/time formatting**. Example: `DD.MM.YYYY HH:mm`
timezone	text	Optional. The timezone used for the conversion. See **List of tz database time zones**, column "TZ database name" for the list of recognized timezones. If omitted, Make uses the organization's timezone. You can **edit your time zone**. Example: `Europe/Prague`, `UTC`

Return value and type

Date representation of the given text value according to the specified format and timezone. Type is date.

Examples

Please note that in the following examples the returned date value is expressed according to ISO 8601, but the actual resulting value is of type date.

Text

```
parseDate( 2016-12-28 ; YYYY-MM-DD )
= 2016-12-28T00:00:00.000Z
```

Text

```
parseDate( 2016-12-28 16:03 ; YYYY-MM-DD HH:mm )
= 2016-12-28T16:03:00.000Z
```

Text

```
parseDate( 2016-12-28 04:03 pm ; YYYY-MM-DD hh:mm a )
= 2016-12-28T16:03:06.000Z
```

Text

```
parseDate( 1482940986 ; X )
= 2016-12-28T16:03:06.000Z
```

addDays (date; number)

Returns a new date as a result of adding a given number of days to a date. To subtract days, enter a negative number.

Text

```
addDays( 2016-12-08T15:55:57.536Z ; 2 )
= 2016-12-10T15:55:57.536Z
```

Text

```
addDays( 2016-12-08T15:55:57.536Z ; -2 )
= 2016-12-6T15:55:57.536Z
```

addHours (date; number)

Returns a new date as a result of adding a given number of hours to a date. To subtract hours, enter a negative number.

Text

```
addHours( 2016-12-08T15:55:57.536Z ; 2 )
= 2016-12-08T17:55:57.536Z
```

Text

```
addHours( 2016-12-08T15:55:57.536Z ; -2 )
= 2016-12-08T13:55:57.536Z
```

addMinutes (date; number)

Returns a new date as a result of adding a given number of minutes to a date. To subtract minutes, enter a negative number.

Text

```
addMinutes( 2016-12-08T15:55:57.536Z ; 2 )
= 2016-12-08T15:57:57.536Z
```

Text

```
addMinutes( 2016-12-08T15:55:57.536Z ; -2 )
= 2016-12-08T15:53:57.536Z
```

addMonths (date; number)

Returns a new date as a result of adding a given number of months to a date. To subtract months, enter a negative number.

Text

```
addMonths( 2016-10-08T15:55:57.536Z ; 2 )
= 2016-12-08T15:57:57.536Z
```

Text

```
addMonths( 2016-10-08T15:55:57.536Z ; -2 )
= 2016-08-08T15:57:57.536Z
```

addSeconds (date; number)

Returns a new date as a result of adding a given number of seconds to a date. To subtract seconds, enter a negative number.

Text

```
addSeconds( 2016-12-08T15:55:57.536Z ; 2 )
= 2016-12-08T15:57:57.536Z
```

Text

```
addSeconds( 2016-12-08T15:55:57.536Z ; -2 )
= 2016-12-08T15:53:57.536Z
```

addYears (date; years)

Returns a new date as a result of adding a given number of years to a date. To subtract years, enter a negative number.

Text

```
addYears( 2016-12-08T15:55:57.536Z ; 2 )
2018-08-08T15:55:57.536Z
```

Text

```
addYears( 2016-12-08T15:55:57.536Z ; -2 )
2014-08-08T15:55:57.536Z
```

setSecond (date; number)

Returns a new date with the seconds specified in parameters. Accepts numbers from 0 to 59. If a number is given outside of this range, it will return the date with the seconds from the previous or subsequent minute(s), accordingly.

Text

```
setSecond( 2015-10-07T11:36:39.138Z ; 10 )
= 2015-10-07T11:36:10.138Z
```

Text

```
setSecond( 2015-10-07T11:36:39.138Z ; 61 )
= 2015-10-07T11:37:01.138Z
```

setMinute (date; number)

Returns a new date with the minutes specified in parameters. Accepts numbers from 0 to 59. If a number is given outside of this range, it will return the date with the minutes from the previous or subsequent hour(s), accordingly.

Text

```
setMinute( 2015-10-07T11:36:39.138Z ; 10 )
= 2015-10-07T11:10:39.138Z
```

Text

```
setMinute( 2015-10-07T11:36:39.138Z ; 61 )
= 2015-10-07T12:01:39.138Z
```

setHour (date; number)

Returns a new date with the hour specified in parameters. Accepts numbers from 0 to 59. If a number is given outside of this range, it will return the date with the hour from the previous or subsequent day(s), accordingly.

Text

```
setHour( 2015-10-07T11:36:39.138Z ; 10 )
= 2015-08-07T06:36:39.138Z
```

Text

```
setHour( 2015-10-07T11:36:39.138Z ; 61 )
= 2015-08-06T18:36:39.138Z
```

setDay (date; number/name of the day in english)

Returns a new date with the day specified in parameters. It can be used to set the day of the week, with Sunday as 1 and Saturday as 7. If the given value is from 1 to 7, the resulting date will be within the current (Sunday-to-Saturday) week. If a number is given outside of the range, it will return the day from the previous or subsequent week(s), accordingly.

Text

```
setDay( 2018-06-27T11:36:39.138Z ; monday )
= 2018-06-25T11:36:39.138Z
```

Text

```
setDay( 2018-06-27T11:36:39.138Z ; 1 )
= 2018-06-24T11:36:39.138Z
```

Text

```
setDay( 2018-06-27T11:36:39.138Z ; 7 )
= 2018-06-30T11:36:39.138Z
```

setDate (date; number)

Returns a new date with the day of the month specified in parameters. Accepts numbers from 1 to 31. If a number is given outside of the range, it will return the day from the previous or subsequent month(s), accordingly.

Text

```
setDate( 2015-08-07T11:36:39.138Z ; 5 )
= 2015-08-05T11:36:39.138Z
```

Text

```
setDate( 2015-08-07T11:36:39.138Z ; 32 )
= 2015-09-01T11:36:39.138Z
```

setMonth (date; number/name of the month in English)

Returns a new date with the month specified in parameters. Accepts numbers from 1 to 12. If a number is given outside of this range, it will return the month in the previous or subsequent year(s), accordingly.

Text

```
setMonth( 2015-08-07T11:36:39.138Z ; 5 )
= 2015-05-07T11:36:39.138Z
```

Text

```
setMonth( 2015-08-07T11:36:39.138Z ; 17 )
= 2016-05-07T11:36:39.138Z
```

Text

```
setMonth( 2015-08-07T11:36:39.138Z ; january )
= 2015-01-07T12:36:39.138Z
```

setYear (date; number)

Returns a new date with the year specified in parameters.

Text

```
setYear( 2015-08-07T11:36:39.138Zv ; 2017 )
= 2017-08-07T11:36:39.138Z
```

Calculate n-th day of the week in a month

If you need to calculate a date corresponding to the n-th day of week in a month (e.g. 1st Tuesday, 3rd Friday, etc.), you can use the following formula:

Text

```
{{addDays(setDate(1.date; 1); 1.n * 7 - formatDate(addDays(setDate(1.date; 1)
```

The formula contains the following items:

Value	Description
`1.n`	n-th day: • `1` for **1**st Tuesday • `2` for **2**nd Tuesday • `3` for **3**rd Tuesday • etc.
`2.dow`	Day of the week: • `1` for Monday • `2` for Tuesday • `3` for Wednesday • `4` for Thursday • `5` for Friday • `6` for Saturday • `7` for Sunday
`1.date`	The date determines the month. To calculate n-th day of week in **current** month use the `now` variable

If you want to calculate only one specific case, e.g. 2nd Wednesday, you may replace the items `1.n` and `2.dow` in the formula with the corresponding numbers. For 2nd Wednesday in the current month you would use the following values:

- `1.n` = `2`
- `1.dow` = `3`
- `1.date` = `now`
- `setDate(now;1)` returns first of current month
- `formatDate(....;E)` returns day of week (1, 2, ... 6)
- see the **original source** for the rest

Calculate days between dates



Text

```
{{round((2.value - 1.value) / 1000 / 60 / 60 / 24)}}
```

The values of `D1` and `D2` above have to be of type date. If they are of type string (e.g. "20.10.2018"), use the `parseDate()` function to convert them to type date.

The `round()` function is used for cases when one of the dates falls within the daylight savings time period and the other does not. In these cases, the difference in hours is by one hour less/more and dividing it by 24 gives a non-integer results.

Calculate the last day/millisecond of a month

When specifying a date range (e.g. in a search module) spanning the whole previous month as a closed interval (the interval that **includes** both its limit points), it is necessary to calculate the last day of the month.

2019-09-01 ≤ D ≤ **2019-09-30**

Text

```
{{addDays(setDate(now; 1); -1)}}
```

In some cases, it is necessary to calculate not only the last day of month, but its last millisecond:

2019-09-01T00:00:00.000Z ≤ D ≤ **2019-09-30T23:59:59.999Z**

Text

```
{{parseDate(parseDate(formatDate(now; "YYYYMM01"); "YYYYMMDD"; "UTC") - 1; "x
```

If the result should respect your timezone settings, simply omit the UTC argument:

```
Text
{{parseDate(parseDate(formatDate(now; "YYYYMM01"); "YYYYMMDD") - 1; "x")}}
```

However, it is preferable to use a half-open interval instead (the interval that **excludes** one of its limit points), specifying the first day of the following month instead and replacing the **less or equal than** operator with **less than**:

2019-09-01 ≤ D **< 2019-10-01**

2019-09-01T00:00:00.000Z ≤ D **< 2019-10-01T00:00:00.000Z**

Transform seconds into hours, minutes and second

```
Text
{{floor(1.seconds / 3600)}}:{{floor((1.seconds % 3600) / 60)}}:{{((1.seconds
```

Values of `second` should be number type. This function is suited only if the second value is less than 86400 (less than a day).

3.4.7. Tokens for date/time parsing

Date/time parsing tokens are special combinations of letters you can use to read and extract date/time data. The following tables list the tokens you can use to parse data in your functions:

Year, month, and day tokens

Input	Example	Description
`YYYY`	`2014`	4 or 2 digit year
`YY`	`14`	2 digit year
`Y`	`-25`	Year with any number of digits and sign
`Q`	`1..4`	Quarter of a year Sets month to first month in quarter
`M MM`	`1..12`	Month number
`MMM MMMM`	`Jan..December`	Month name
`D DD`	`1..31`	Day of month
`Do`	`1st..31st`	Day of month with ordinal numbers
`DDD DDDD`	`1..365`	Day of year
`X`	`1410715640.579`	Unix timestamp
`x`	`1410715640579`	Unix ms timestamp

Week year, week, and weekday tokens

Input	Example	Description
`ddd dddd`	`Mon...Sunday`	Day name
`GGGG`	`2014`	ISO 4 digit week year
`GG`	`14`	ISO 2 digit week year
`W WW`	`1..53`	ISO week of year
`E`	`1..7`	ISO day of week

Hour, minute, second, millisecond, and offset tokens

Input	Example	Description
`H HH`	`0..23`	Hours (24 hour time)
`h hh`	`1..12`	Hours (12 hour time used with `a A`)
`k kk`	`1..24`	Hours (24 hour time from 1 to 24)
`a A`	`am pm`	Post or ante meridiem (Note the one character `a p` are also considered valid)
`m mm`	`0..59`	Minutes
`s ss`	`0..59`	Seconds
`S SS SSS`	`0..999`	Fractional seconds
`Z ZZ`	`+12:00`	Offset from UTC as `+-HH:mm`, `+-HHmm`, or `Z`

3.4.8. Tokens for date/time formatting

Use date/time formatting tokens to change the format of date/time data. For example, you can convert a timestamp to a quarter of the year or day of the week. The following tables present the tokens supported for changes in formatting.

Year, month, and day tokens

Token	Output	Description
`YY`	`70 71 ... 29 30`	2 digit year
`YYYY`	`1970 1971 ... 2029 2030`	4 digit year
`Y`	`1970 1971 ... 9999 +10000 +10001`	Year with any number of digits and sign
`Q`	`1 2 3 4`	Quarter of year
`Qo`	`1st 2nd 3rd 4th`	Quarter of year with ordinal
`M`	`1 2 ... 11 12`	Month number
`Mo`	`1st 2nd ... 11th 12th`	Month with ordinal
`MM`	`01 02 ... 11 12`	Month number with leading zero
`MMM`	`Jan Feb ... Nov Dec`	Month abbreviation
`MMMM`	`January February ... November December`	Month name
`D`	`1 2 ... 30 31`	Day of month
`Do`	`1st 2nd ... 30th 31st`	Day of month with ordinal
`DD`	`01 02 ... 30 31`	Day of month with leading zero
`DDD`	`1 2 ... 364 365`	Day of year
`DDDo`	`1st 2nd ... 364th 365th`	Day of year with ordinal
`DDDD`	`001 002 ... 364 365`	Day of year with leading zero

Week year, week, and weekday tokens

Token	Output	Description
`d`	`0 1 ... 5 6`	Day of week
`do`	`0th 1st ... 5th 6th`	Day of week with ordinal
`dd`	`Su Mo ... Fr Sa`	Day abbreviation
`ddd`	`Sun Mon ... Fri Sat`	Day abbreviation
`dddd`	`Sunday Monday ... Friday Saturday`	Day name
`E`	`1 2 ... 6 7`	Day of week (ISO)
`w`	`1 2 ... 52 53`	Week of year
`wo`	`1st 2nd ... 52nd 53rd`	Week of year with ordinal
`ww`	`01 02 ... 52 53`	Week of year with leading zero
`W`	`1 2 ... 52 53`	Week of year (ISO)
`Wo`	`1st 2nd ... 52nd 53rd`	Week of year with ordinal (ISO)
`WW`	`01 02 ... 52 53`	Week of year with leading zero (ISO)
`gg`	`70 71 ... 29 30`	Week year
`gggg`	`1970 1971 ... 2029 2030`	Week year
`GG`	`70 71 ... 29 30`	Week year (ISO
`GGGG`	`1970 1971 ... 2029 2030`	Week year (ISO)

Hour, minute, second, millisecond, and offset tokens

Token	Output	Description
`H`	`0 1 ... 22 23`	24 hour time
`HH`	`00 01 ... 22 23`	24 hour time with leading zero
`h`	`1 2 ... 11 12`	12 hour time
`hh`	`01 02 ... 11 12`	12 hour time with leading zero
`k`	`1 2 ... 23 24`	24 hour time
`kk`	`01 02 ... 23 24`	24 hour time with leading zero
`A`	`AM PM`	Post or ante meridiem (upper case)
`a`	`am pm`	Post or ante meridiem (lower case)
`m`	`0 1 ... 58 59`	Minutes
`mm`	`00 01 ... 58 59`	Minutes with leading zero
`s`	`0 1 ... 58 59`	Seconds
`ss`	`00 01 ... 58 59`	Seconds with leading zero
`S`	`0 1 ... 8 9`	Fractional seconds
`SS`	`00 01 ... 98 99`	Fractional seconds with leading zero
`SSS`	`000 001 ... 998 999`	Fractional seconds with two leading zeros
`SSSS ... SSSSSSSSS`	`000[0..] 001[0..] ... 998[0..] 999[0..]`	Fractional seconds
`Z`	`-07:00 -06:00 ... +06:00 +07:00`	Time zone
`ZZ`	`-0700 -0600 ... +0600 +0700`	Time zone
`X`	`1360013296`	Unix Timestamp
`x`	`1360013296123`	Unix Millisecond Timestamp

3.4.9. Array functions

You can use array functions to transform **array data**. Array functions can, for example, search and sort the items in an array, perform mathematical operations, convert arrays to collections, reorder the items in an array, and more. Below is a list of supported array functions and a description what of each function does.

add

add (array; value1; value2; ...)

Adds values specified in parameters to an array and returns that array.

contains

contains (array; value)

Verifies if an array contains the value.

deduplicate

deduplicate (array)

Removes duplicates from an array.

distinct

distinct (array; [key])

Removes duplicates inside an array. Use the *key* argument to access properties inside complex objects. To access nested properties, use dot notation. The first item in an array is index 1.

Text

```
distinct( Contacts[] ; name )
Removes duplicates inside an array of contacts by comparing the name property
```

first

first (array)

Returns the first element of an array.

flatten

flatten (array)

Creates a new array with all sub-array elements concatenated into it recursively up to the specified depth.

More details about this function can be found in the **Array.prototype.flat** documentation.

join

join (array; separator)

Concatenates all the items of an array into a string, using a specified separator between each item.

keys

keys (object)

Returns an array of a given object's or array's properties.

last

last (array)

Returns the last element of an array.

length

length (array)

Returns the number of items in an array.

map

map (complex array; key;[key for filtering];[possible values for filtering separated by a comma])

Returns a primitive array containing values of a complex array. Allows filtering values. Use raw variable names for keys.

Text

```
map( Emails[] ; email )
Returns a primitive array with emails.
```

Text

```
map( Emails[] ; email ; label ; work,home )
Returns a primitive array with emails that have a label equal to work or home
```

How to Extract an Item And/Or Its Value From an Array of Collections



merge

merge (array1; array2; ...)

Merges two or more arrays into one array.

remove

remove (array; value1; value2; ...)

Removes values specified in the parameters of an array. Effective only in case of primitive arrays of text or numbers.

reverse

reverse (array)

The first element of the array becomes the last element and vice versa.

shuffle

shuffle (array)

Shuffles (randomly reorders) elements of an array.

slice

slice (array; start; [end])

Returns a new array containing only selected items. The first item in the array has an index of 0.

sort

sort (array; [order]; [key])

Sorts values of an array. The valid values of the `order` parameter are:

- `asc` (default) - ascending order: 1, 2, 3, ... for type Number. A, B, C, a, b, c, ... for type Text.
- `desc` - descending order: ..., 3, 2, 1 for type Number. ..., c, b, a, C, B, A for type Text.
- `asc ci` - case insensitive ascending order: A, a, B, b, C, c, ... for type Text.
- `desc ci` - case insensitive descending order: ..., C, c, B, b, A, a for type Text.

Use the `key` parameter to access properties inside complex objects. Use raw variable names for keys. To access nested properties, use dot notation. The first item in an array is index 1.

Text

```
sort( Contacts[] ; name )
Sorts an array of contacts by the name property in default ascending order.
```

Text

```
sort( Contacts[] ; desc ; name )
Sorts an array of contacts by the name property in descending order.
```

Text

```
sort( Contacts[] ; asc ci ; name )
Sorts an array of contacts by the name property in case insensitive ascending
```

Text

```
sort( Emails[] ; sender.name )
Sorts an array of emails by the sender.name property.
```

toArray

toArray (collection)

Converts a collection into an array of key-value collections.

toCollection

toCollection (array; key; value)

Converts an array containing objects with key-value pairs into a collection.

3.4.10. Custom functions

> This feature is available to Enterprise customers.

Custom functions allow you to enhance the built-in data transformation capabilities available in the scenario designer.

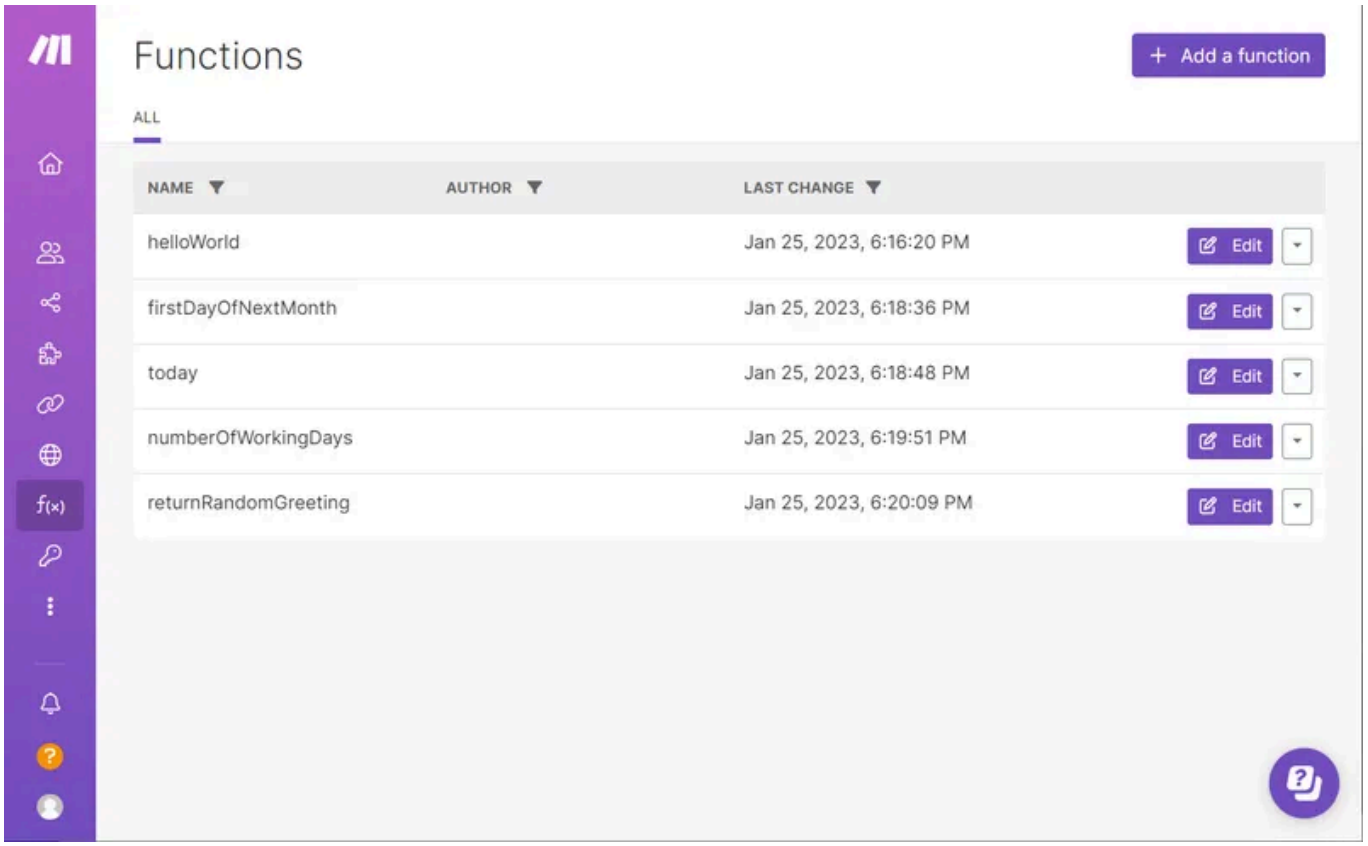
Custom functions are created using JavaScript.

> Make supports the infrastructure and system behavior that takes care of managing and using custom functions. However, our customer care team cannot provide support on how to write and debug your own custom functions.

Create and manage custom functions

You create and manage custom functions in the Functions section.

Custom functions belong to a team. All users with the Team Member role can view and use custom functions. To create and edit custom functions, you need to have the Team Admin role.

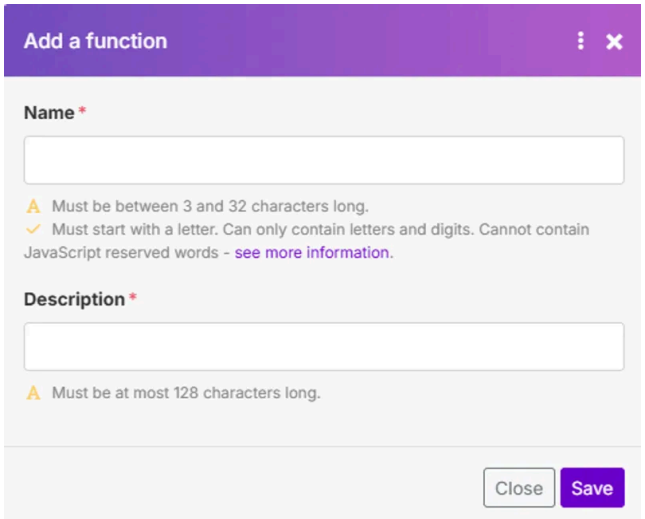


To create a new custom function:

1. In the left sidenav, click **Functions**.

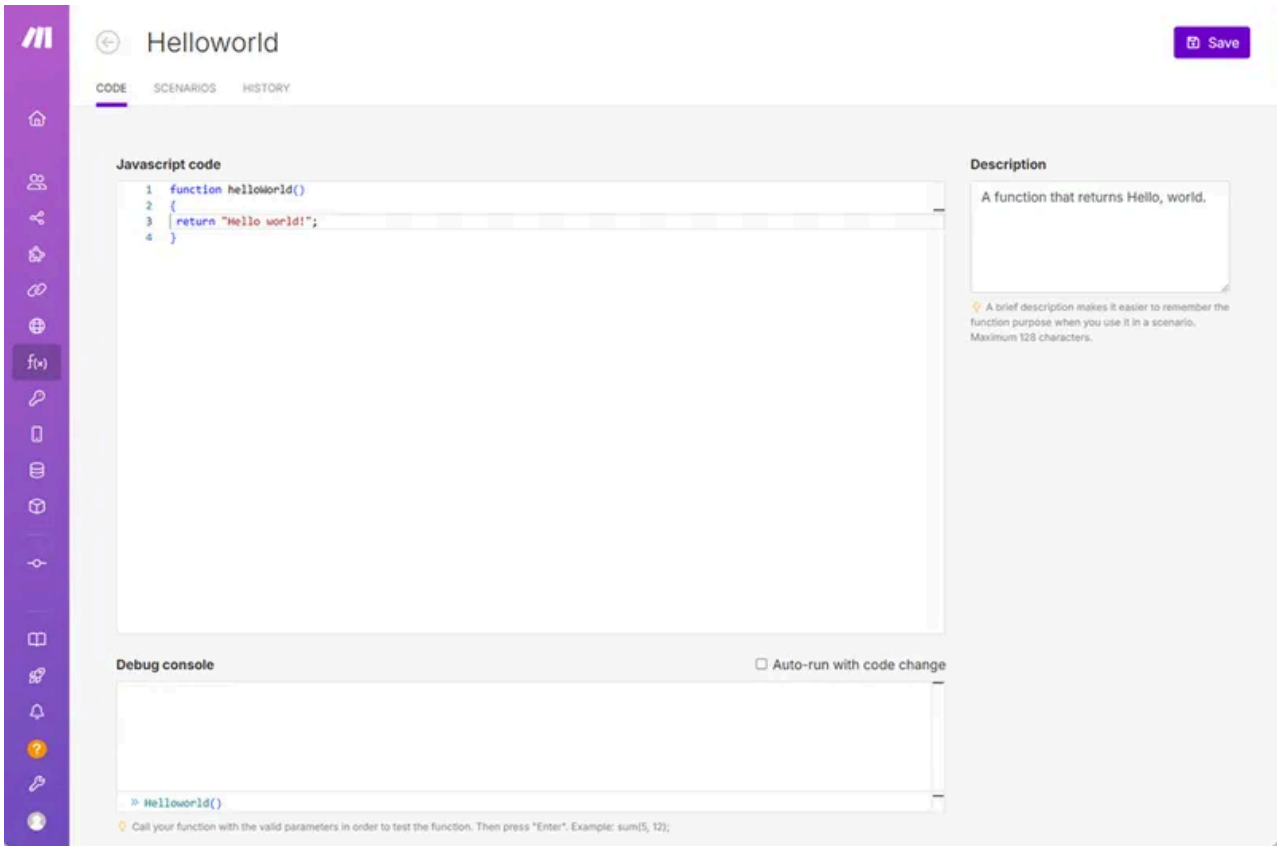
2. Click **+ Add a function**.

3. Enter a **name** and **description**. The name and description will appear in the list of functions available in the scenario designer.



4 Click **Save**.

5 Write the code of the function. Use the keyword `return` to define the return value.



6 Click **Save**.

When you save the function for the first time, it immediately becomes available in the scenario designer. Custom functions are marked with an icon to distinguish them from the built-in functions.

Basic structure of custom functions

The structure of custom functions reflects the structure of regular JavaScript functions. Each function must contain the following:

- Function header - the name of the function and the parameters it accepts.
- Function body enclosed in curly brackets.
- Exactly one `return` statement.

Supported languages

Custom functions support **JavaScript ES6**. No 3rd party libraries are supported.

The code runs on Make's back end. This means that we do not support functionality that only makes sense in the context of writing browser-executed JavaScript.

Validation of custom functions

Make does not validate the code of your functions. If there are errors in your code, they will appear as errors in the execution of scenarios that use these functions. Invalid functions will cause your scenarios to fail.

Limitations of custom functions

The following limits apply to all custom functions that you create:

- A single custom function can't run for more than 300 milliseconds.
- The code of a custom function cannot have more than 5000 characters.
- Functions must be synchronous. You can't run asynchronous code in your functions.
- You can't make HTTP requests from your functions.
- You can't call other custom functions from inside of a custom function.
- You can't use recursion in your custom functions.
- In order to use a custom variable with an iterator, you need to:
    - Set the value with the custom function in a **Set Variable** module before the **Iterator**.
    - Map the output of the **Set Variable** to the **Iterator**.

Custom functions version history

Every time you save a custom function, the system creates a new history record. This allows you to compare versions and revert your function to a previous version.



Select a history record to compare the function with the previous version. You can only compare two consecutive versions.

Click **Restore this version** to revert to the version that you're currently viewing. This creates a new version of the function with the content of the restored version.

Work with dates inside custom functions

Custom functions are executed on Make's servers. Therefore, whenever you work with dates inside a function, the dates will take into account the time zone set for your organization.

Call built-in functions from inside of your custom functions

You can use Make's built-in functions in your custom function code.

Built-in functions are available as methods of the `iml` object.

The following example uses the Length built-in function.

Text

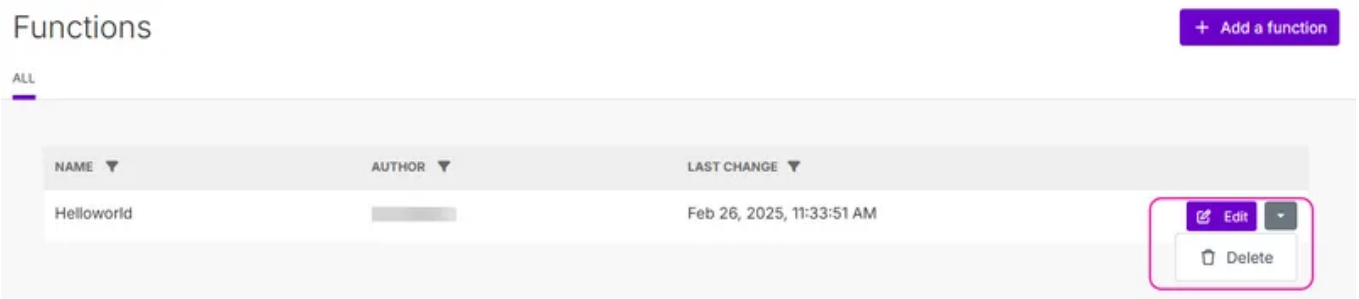
```
function useBuiltInFunction(text){
    return iml.length(text);
}
```

Debug custom functions

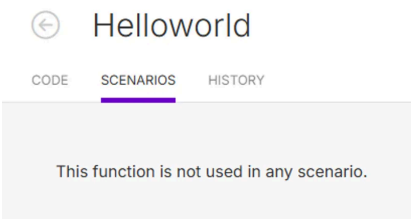
You can execute your custom function code in the debug console, which is located under the code editor.

Delete custom functions

You can delete your custom functions in the list in the Functions section.



Before you delete a function, click **Edit** next to it and then switch to the **Scenarios** tab. Here, you can check in which scenarios the function is used.



When you delete a function that is used in a scenario, the scenario will fail to execute on its next run.

The scenario that uses the deleted function will fail with the following error message:

`Failed to map '<field name>': Function '<function name>' not found!`

Examples

Hello world

Returns the string "Hello world."

Text

```
function helloWorld()
{
    return "Hello world!";
}
```

Count the number of working days in a month

Calculates how many working days there are in the month specified by the month's number and a year in the function's arguments. Does not take into account local holidays.

Text

```
function numberOfWorkingDays(month, year) {
  let counter = 0;
  let date = new Date(year, month - 1, 1);
  const endDate = new Date(date.getFullYear(), date.getMonth() + 1, 0);

  while (date.getTime() < endDate.getTime()) {
    const weekDay = date.getDay();
    if (weekDay !== 0 && weekDay !== 6) {
      counter += 1;
    }
    date = new Date(date.getFullYear(), date.getMonth(), date.getDate() + 1);
  }
  return counter;
}
```

Calculate the number of days between two dates

Calculates the number of days between two dates passed to the function as arguments.

Text

```
function numberOfDays(start, end) {
  const startDate = new Date(start);
  const endDate = new Date(end);
  return Math.abs((startDate.getTime() - endDate.getTime()) / (1000 * 60 * 60
}
```

Randomization: return a random greeting from an array of greetings

Accepts an array as an argument. Returns a random item from the array.

Text

```
function randomGreeting(greetings) {
  const index = Math.floor(Math.random() * greetings.length);
  return greetings[index];
}
```

3.5. Data stores

Learn how to use Make's built-in database tool to store and manage data:

→ **Data stores**

→ **Data structures**

3.5.1. Data stores

Data stores allow you to store data from a scenario or transfer data between individual scenarios or scenario runs. You can use data stores to store data from apps during scenario execution. Data stores are similar to a simple database.

Data storage allowance

Before creating a data store, take note of your data storage allowance. Total data storage is based on number of operations in your plan, with every 1,000 operations equaling 1 MB of data storage.

> As 1 MB is the minimum size of a data store in Make, you will need at least 1 MB of data storage available to create a new data store.

Consider the following data storage allowances for data stores, depending on your **plan**:

Free plan users | **Enterprise users** | All other users

As an Enterprise user, calculate your total data store by determining your average monthly operations, and then dividing your monthly operations by 1,000.

For example, with 1,000,000 operations per year, you have 83.3 MB of data storage available to allocate to your data stores.

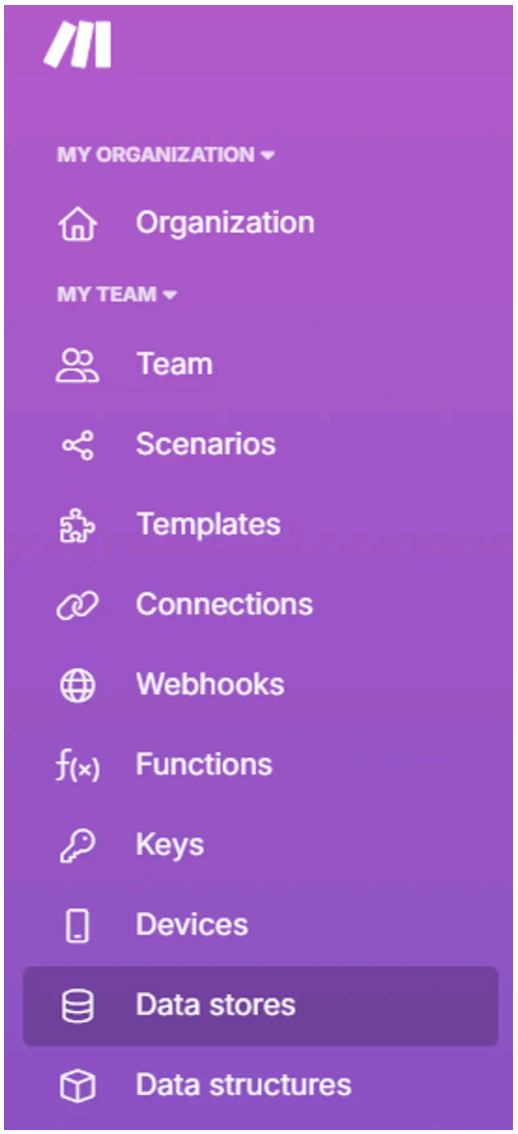
1,000,000 operations / 12 months = 83,333 operations
83,333 operations / 1,000 = 83.33 MB

Each data store is at least 1 MB in size.

To adjust the data storage allocated to your existing data stores, see **Out of space error**.

How to create a data store

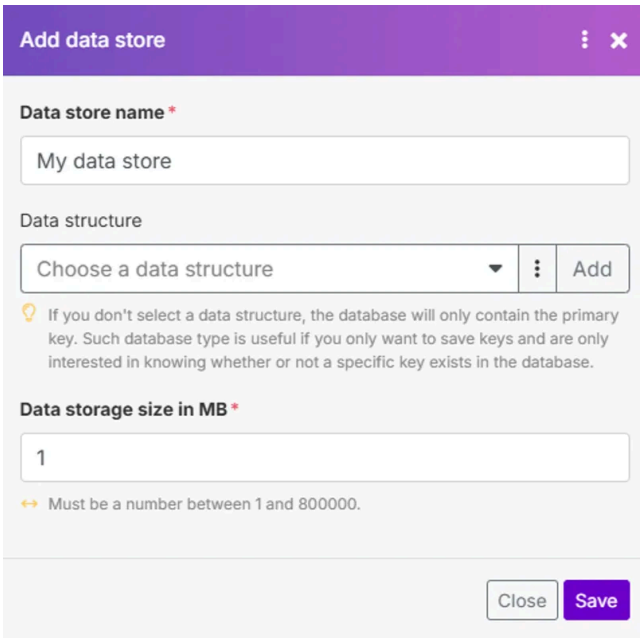
1 In the left sidebar, click **Data stores**.



2 Click **+ Add data store**.



3 Enter settings for the new data store.



Field	Description
Data store name	Enter the name for the data store. For example, `Contacts`.
Data structure	A data structure is a list of the columns for a table that indicates the column name and data type. You have three options: • **Select a data structure that has been already created** • **Leave the field empty** If you don't select a data structure, the database will only contain the primary key. Such a database type is useful if you only want to save keys and are only interested in knowing whether or not a specific key exists in the database. • **Add a new data structure** Click the **Add** button to create a new data structure.
Data storage size in MB	Allocate the size for the data store from your total data storage. The amount can be changed at any time. 1 MB is the minimum data storage size per data store. You can see your available storage space in the text below the **Data storage size in MB** field.

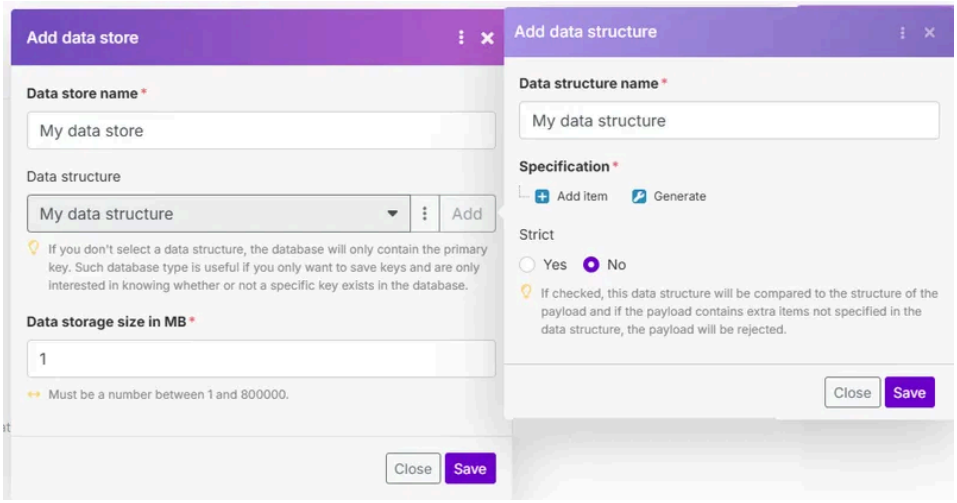
Manage the data structure of a data store

Add a new data structure

During the process of setting up a data store, you can set up a new data structure.

To set up the data structure while adding a data store, click the Add button for the Data structure field

You can access this dialog by clicking the **Add** button when creating or editing the data store:



Data structure name	Enter the name for the data structure. The data structure name is its unique identifier and cannot be changed later.

Specification	There are two options for how you can specify the data store columns.
	• Click the **Add item** button to specify the properties of one column manually.
	Enter the **Name** and **Type** for the data store column and define the corresponding properties.
	• Use the *Generator button* to determine the columns from the sample data you provide.
	For example, the following JSON sample data creates three columns (name, age, phone number) with phone number as a collection of mobile and landline:
	`{ "name":"John",`
	`"age":30,`
	`"phone number": {`
	`"mobile":"987654321",`
	`"landline":"123456789"`
	`}`
	`}`

Strict	If enabled, the data structure will be compared to the structure of the payload and if the payload contains extra items not specified in the data structure, the payload will be rejected.

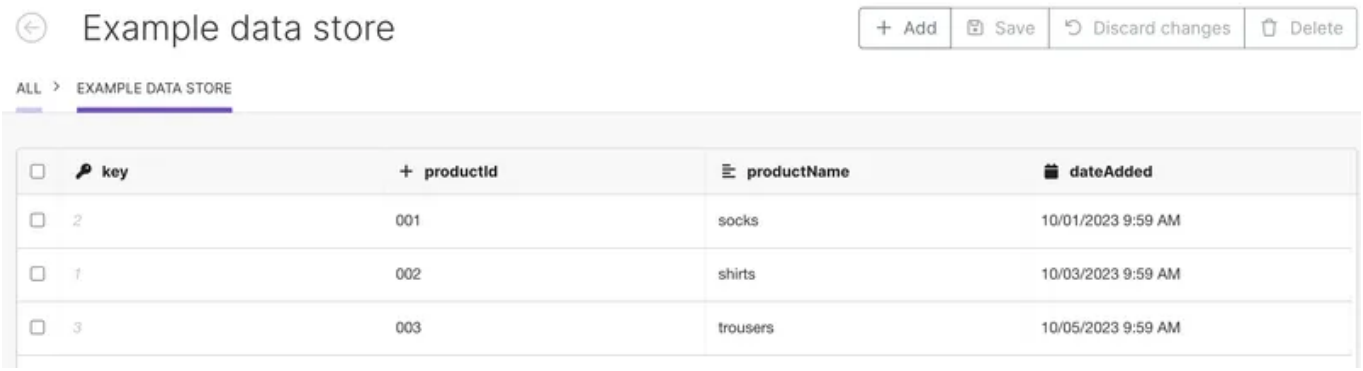
Update the data structure of a data store

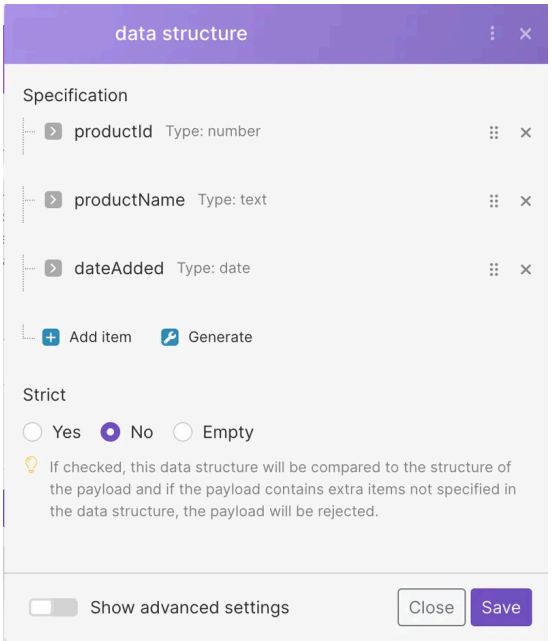
> Be careful when updating the data structure of a data store. Before updating the structure, make a backup of the data.
>
> Changes in the data structure of a data store might lead to unexpected results.

When you want to update the data structure of a data store, you should keep in mind that:

- The data structure field names are unique identifiers of the data store columns. When you rename a field of a data structure, Make cannot retrieve the original data in the data store column, because they use a different column identifier.
- You can update the data structure field label anytime without the effects mentioned in the previous point.
- The changes to the data store structure apply only to the new data you put in the data store. Make doesn't change or validate the original data to fit the updated structure.

The best approach to updating the data structure of a data store is to create temporary fields with copies of your data. Update the data in the temporary fields and make sure they conform to the final data structure.





Rename a field of the data store structure

You can update the **Label** of a data store structure field anytime.

To change the data store structure field **Name**:

1. **Create a backup of the data in the data store**.
2. Create a field in your data store with the new name.
3. Copy all data from the oroginal column to the new column.
4. Update all data in the original column to empty fields. This step prevents storing the data in the original column alongside the data in the new column.

You have put the data from the original data store column into the new data store column. In addition, you have a data store backup to check that the update was successful.

Change the type of a field of the data store structure

1. **Create a backup of the data in the data store**.
2. Either create a temporary field for the updated data type, or you can skip this step and update the field type in place.
3. Use a conversion function to update the type of all values in the data store column to the new type. For example, to convert `text` to `date`, use the `parseDate` function.

Actions

Add/Replace a Record

Adds or replaces a record in the data store.

Field	Description
Data store	Select or add the data store where you want to create a record.
Key	Enter the unique key. The key can be used later to retrieve the record. If you leave this field blank, the key will be generated.
Overwrite an existing record	Enable this option to overwrite the record. The record you want to overwrite must be specified in the **Key** field above.
Record	Enter the desired values to the record's fields. The maximum size of the record in the data store is 15 MB.

> The module throws an error when you try to add the record which is already in the data store under the same name and the **Overwrite an existing record** option is disabled.

Update a Record

Updates a record in the selected data store.

Field	Description
Data store	Select or add the data store where you want to create a record.
Key	Enter the unique key of the record you want to update.
Insert missing record	Enable this option to create a new record if the record with the specified key doesn't already exist.
Record	Enter the desired values to the record's fields that you want to update. The maximum size of the record in the data store is 15 MB.

Get a Record

Retrieves a record from the selected data store.

Field	Description
Data store	Select the data store you want to retrieve a record from.
Key	Enter the unique key of the record you want to retrieve.
Return Wrapped Output	Choose if you want the output to be returned in the same way that the `Search records` module returns data.

Check the Existence of a Record

Returns the value `true` if the record exists in the specified data store or `false` if the record doesn't exist in the data store.

Field	Description
Data store	Select the data store you want to check for the record existence.
Key	Enter the key of the record you want to check for existence

Delete a Record

Deletes a specified record from the selected data store.

Field	Description
Data store	Select the data store you want to check for the record existence.
Key	Enter the key of the record you want to delete.

Delete All Records

Deletes all records from the selected data store.

Field	Description
Data store	Select the data store you want to delete all records from.

Search Records

Performs a search for records based on filter settings.

Field	Description
Data store	Select the data store you want to check for the record's existence.
Filter	Set the filter for the search. Select the column, operator and required value (search term) for the search. If you use the **datetime** operators, you need to provide a value in a date format. Use the `parseDate` function for this purpose.
Sort	• Key: Select the column name you want to sort the results by. • Order : Select whether you want to sort results in ascending or descending order.
Limit	Set the maximum number of search results Make will return during one execution cycle.
Continue the execution of the route even if the module returns no results	If enabled, the scenario will not be stopped by this module.

Count Records

Returns the number of records in the selected data store.

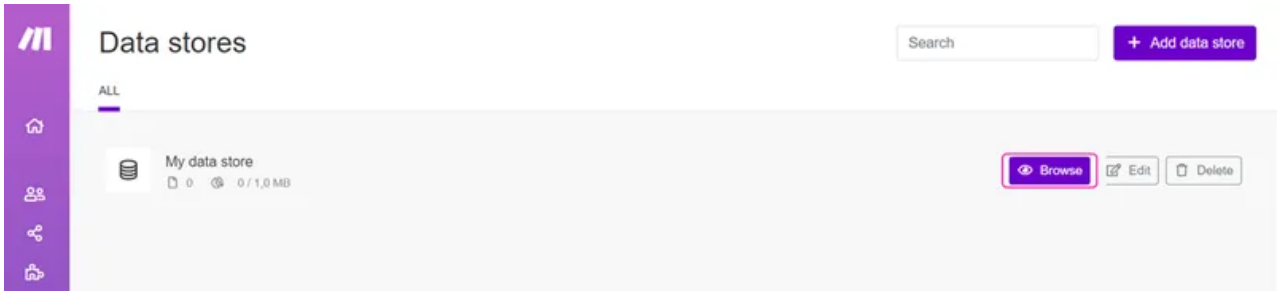
Field	Description
Data store	Select the data store whose records you want to count.

Manage records in a data store

Make allows you to view, update, and delete the records in your data store.

1. In the left sidebar, click **Data stores**.

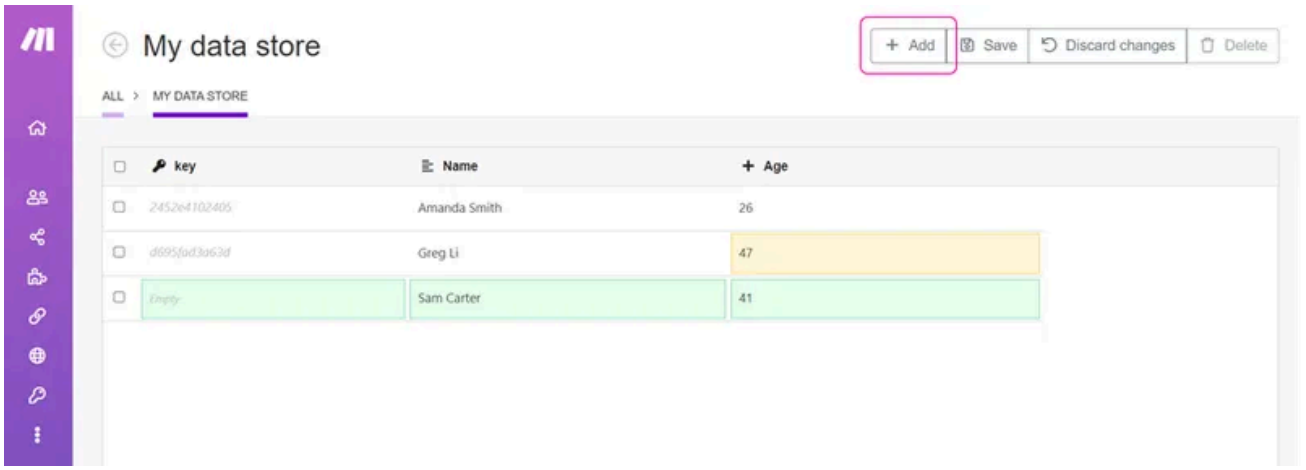
2. Click **Browse** next to your data store.



Add or edit records

Click Add to add new records to the data store. You can add multiople records by clicking Add multiople times. Newly inserted records are highlighted in green.

Click an existing field to change it. Changed records are highlighted in yellow.



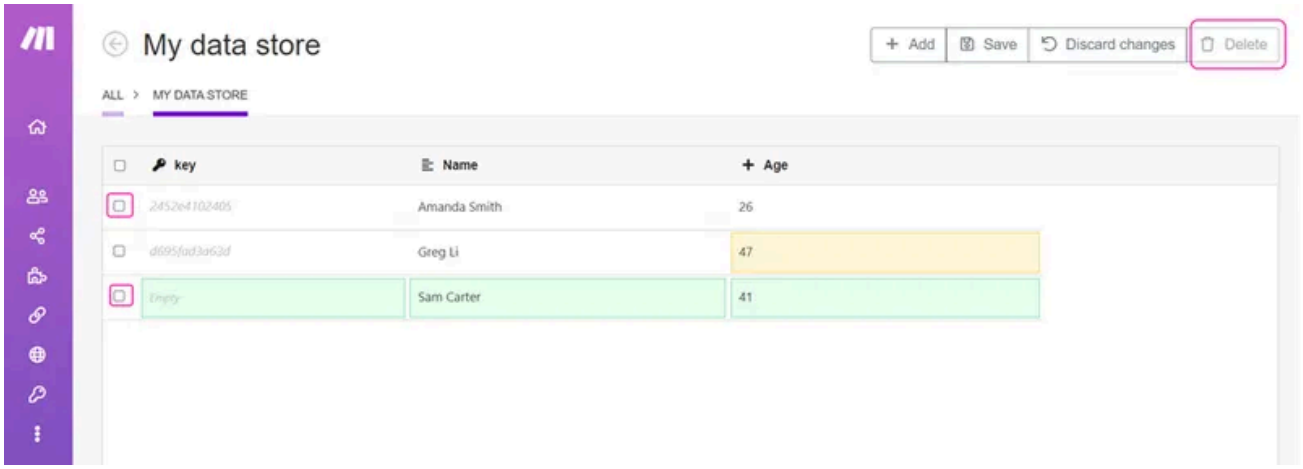
Click **Save** to save all your changes to the data store. Click **Discard changes** to throw away any changes you have made, including added records and edited records.

You cannot use **Discard changes** to get back records that you have deleted.

Delete records

To delete records from your data store, first select the records you want to delete by selecting the check boxes next to the records. Then, click the **Delete** icon.

You cannot roll back deleted records.



Back up data in the data store

When you want to make large-scale changes in your data store, like deleting a high number of records or changing the data store structure, you should make a backup of your data first.

There are two options of how to backup the data in your data store with Make:

- Create a Make scenario
- Use the **Make API**.

To back up your data using a Make scenario:

1. Clone the data structure of the original data store. Name the data structure to clearly show that it's for the backup data store.
2. Create a data store with the backup data structure. Name the data store to clearly show that it's a backup of your data.
3. Create a Makescenario. Name the scenario to clearly show that it's for backing up data in the data store.
4. Add the **Data Store** > **Search Records** module to the scenario. In the **Data Store** field, select the data store which contains the data you want to back up.
5. Add the **Data Store** > **Add/Replace Record** module to the scenario.
6. Run the scenario.

The backup data store now contains all data from the original data store.

> You can reduce the number of operations by using the **Text Aggregator** and **CSV** or SQL-processing tools and apps.

Troubleshooting

Restore lost data from your data store

There is no automated process to restore lost values in your data stores. However, there is a manual approach that you can use to fix the issue.

To locate and restore the missing data:

1. Open the specific scenario where the missing items were stored.
2. Review the execution history and identify the instances where items were inserted into the data store.
3. Copy the identified missing data into your data store.

Update the data store structure

There are two challenges that occur when you update the data structure of a data store:

- When you rename a field of the data structure, the data in the field becomes inaccessible.

  If you want to access the data again, add a field to the data store structure with the original field name.

  With the data in the original field accessible again, you can follow the steps to **update the data store structure**.

- When you change the type of a field in a data structure, the original data keeps the original data type and the new data has the new type.

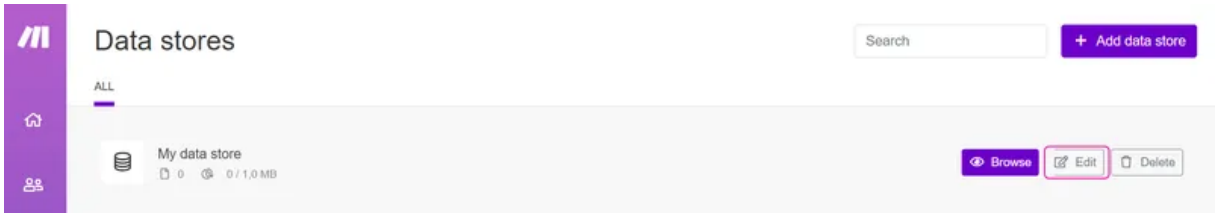
> Before updating the data structure of a data store, **create a backup of your data**.

Out of space error

You get this error message if you currently have a datastore that has already been assigned your allocated datastore storage.

You can edit any of your existing data stores to free up space.

1. In the left sidebar, click **Data stores**.
2. Click **Edit** next to a data store.



3. Reduce the data storage size in the **Data storage size in MB** field.

You can now add a new data store.

> Make sure that when you create a new data store you do not assign all of your space to only one record unless you need it.

Date/time displayed as date for text value

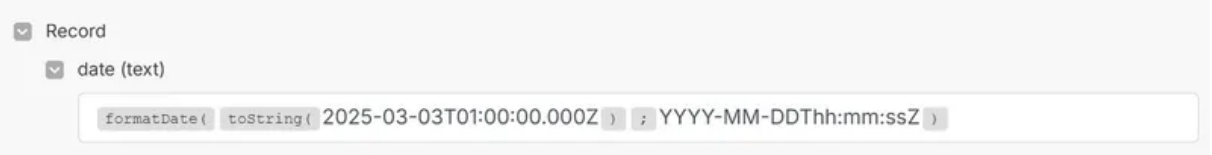
When you input a raw date/time string in ISO format ending with Z in a text-based field of a data store, it will display in a parsed (date) format.

Add **functions** to convert a date in raw ISO format to text:

1. In the **data structure** of your data store, set the **Type** of the relevant field to Text.

2. In the relevant scenario, add a **Data store > Add/replace a record** module.

    1. In its configuration settings, locate the field that corresponds to the relevant column in the data store.
    2. Add the **formatDate()** and **toString()** functions as shown below, with the raw ISO date as the first input value.



    For more on formatting date and time data, see **Tokens for date/time formatting**.

3. Click **Save**.

3.5.2. Data structures

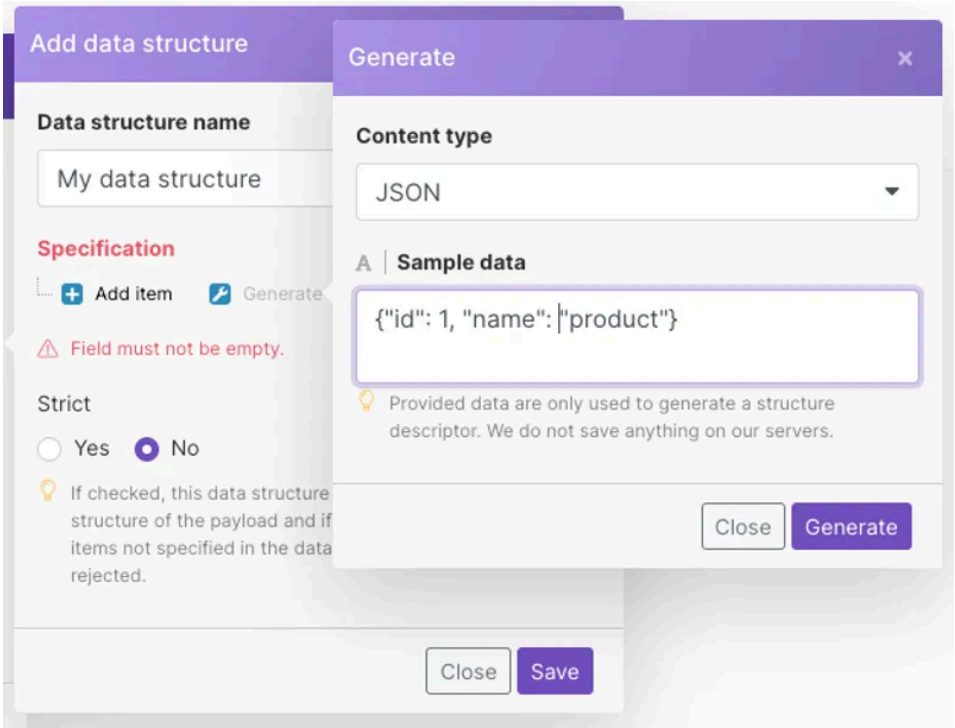
A data structure is a document that describes in detail the format of the data being transferred to Make. Based on this document, the scenario editor is able to figure out which module returns or receives which kind of data. The data structure documents are most commonly used for serializing/parsing data formats such as JSON, XML, CSV and others.

You can create a data structure when **adding a new data store.**

You can read more about **supported data types**.

Data structure generator

Data structures don't always have to be created. You can make it easier by using a template from our built-in generator. By providing a data sample, the generator will automatically create a data structure based off of the data sample you entered. The created data structure can then be manually modified.



3.6. Developers

Discover the Make DevTool to debug your scenarios like a pro:

→ **Make DevTool**

3.6.1. Make DevTool

Make DevTool allows you to debug your Makescenarios by adding an extra pane to the **Chrome Developer Tools**. Using this new debugger pane, you can check all the manual runs of your scenarios, review all the performed operations, and see the details of every API call performed. Additionally, with Make DevTools you can see which module, operation, or response caused an error in your scenario. This helps you debug your scenario and get your scenario back on track.

Install Make DevTool

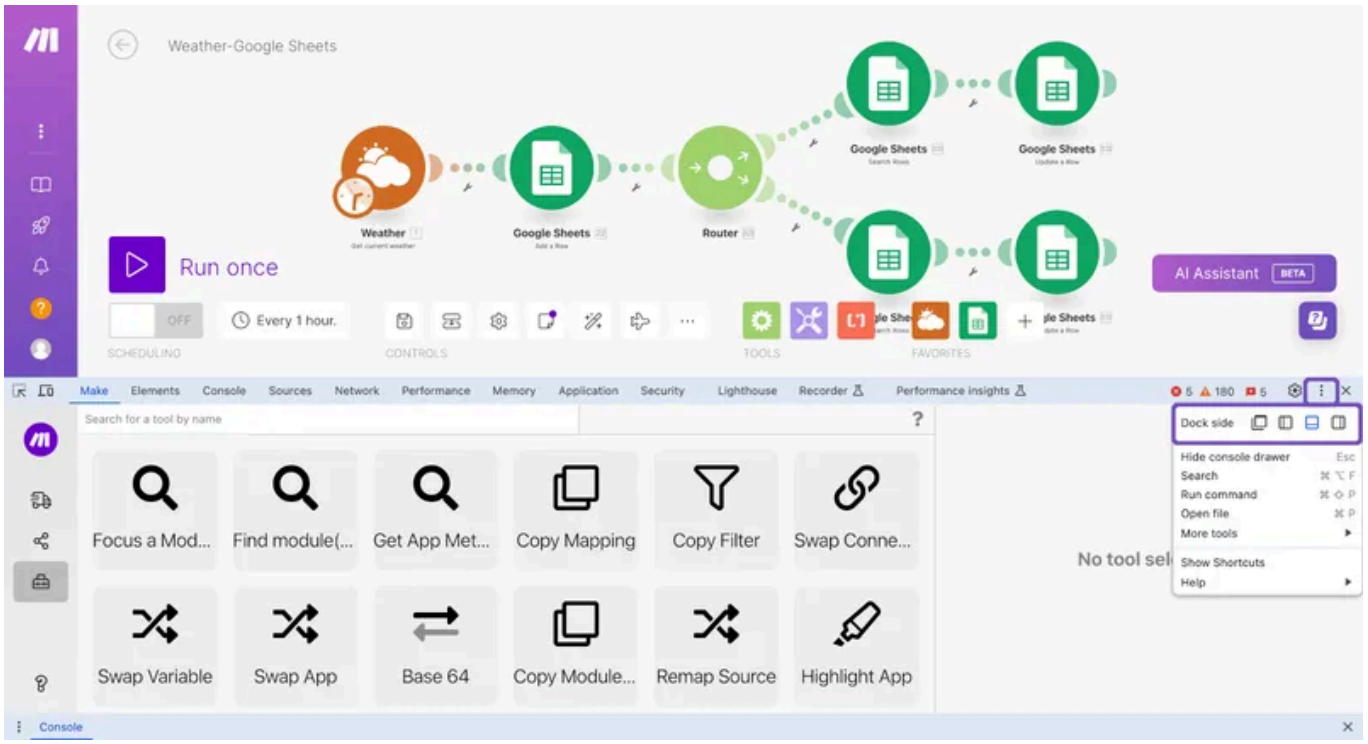
1. Open the **Chrome Web Store** and search for Make DevTool. You can also use **this direct link** to install the extension.

2. Click the **Add to Chrome** button. Confirm your decision in the pop-up window that opens.

Make DevTool is installed in the Chrome Developer Tools pane.

To start using Make DevTool, open a scenario in Make, press **Control+Shift+I** or **F12** (Windows) or **Command+Option+I** (Mac) on your keyboard to open Chrome Developer Tools, and switch to the Make tab.

We recommend docking the Chrome Developer Console to the bottom to maintain a better view of your modules.



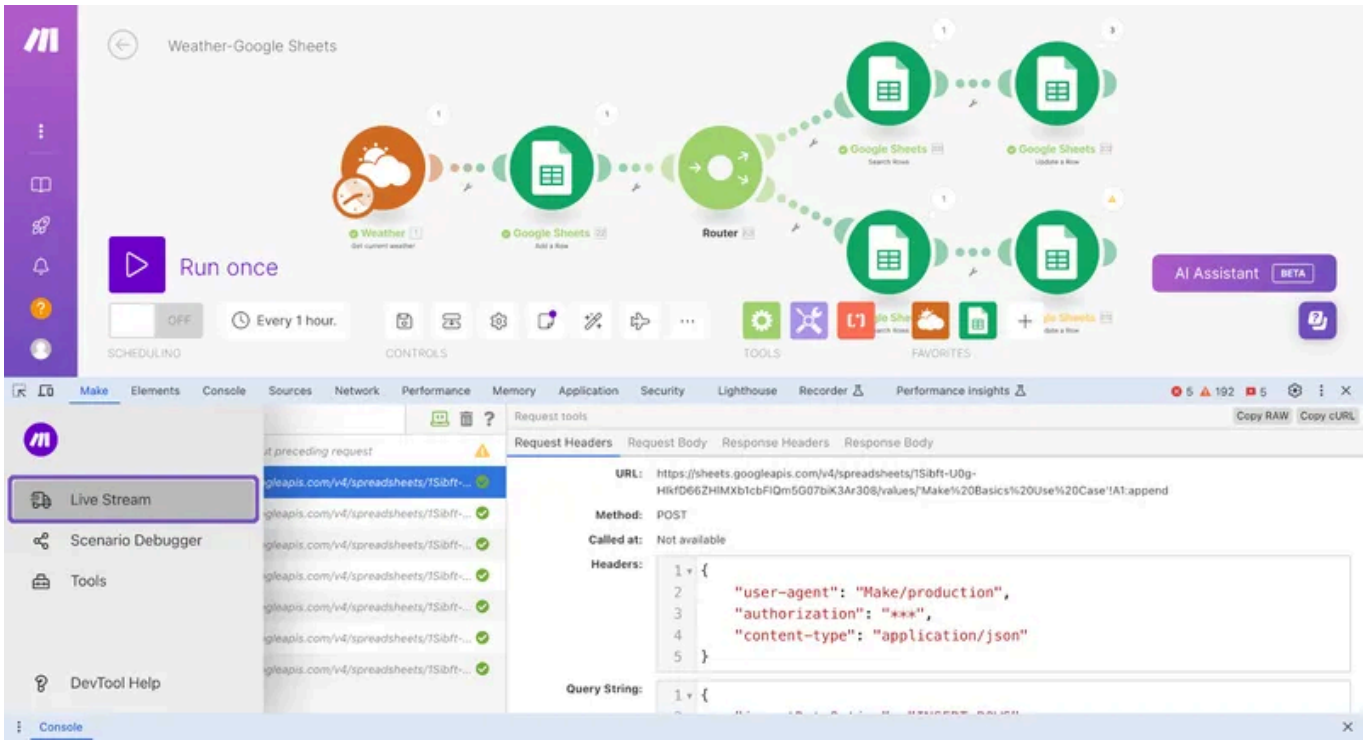
Use Live Stream

Live Stream displays what is happening in the background of your scenario when you hit the **Run once** button.

It allows you to view the following information for each module in your scenario:

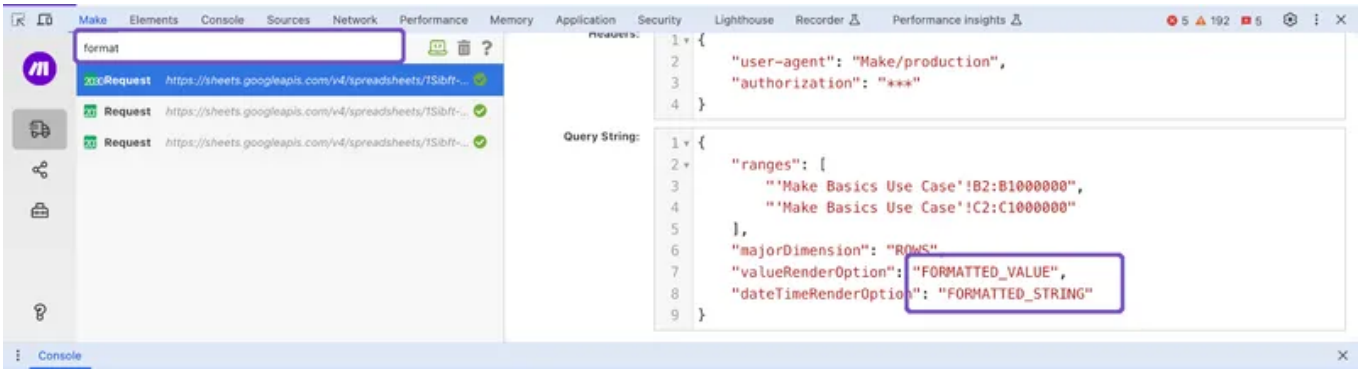
- Request Headers (API endpoint URL, HTTP method, time and date the request has been called at, request headers, and query string)
- Request Body
- Response Headers
- Response Body

After you run a scenario, select **Live Stream** from the left-side panel and click one of the tabs in the right panel of Make DevTool to view the information.



Search in logs

Enter the search term into the search field in the left panel of Make DevTool to display requests and responses that contain the search term.



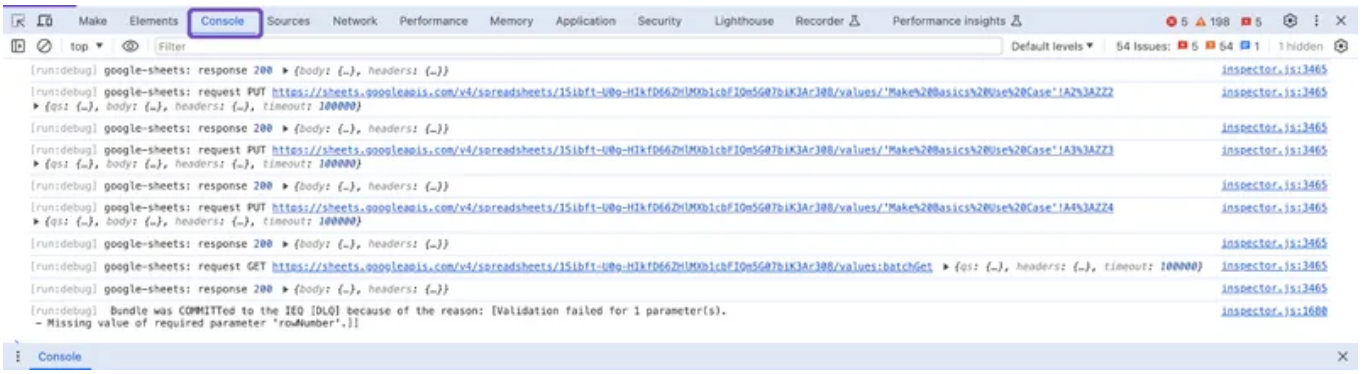
Clear the list of recorded requests

To clear the list of requests recorded by Make DevTool, click the bin icon next to the search field.

Enable console logging

To enable logging into the console, click the computer icon next to the search field.

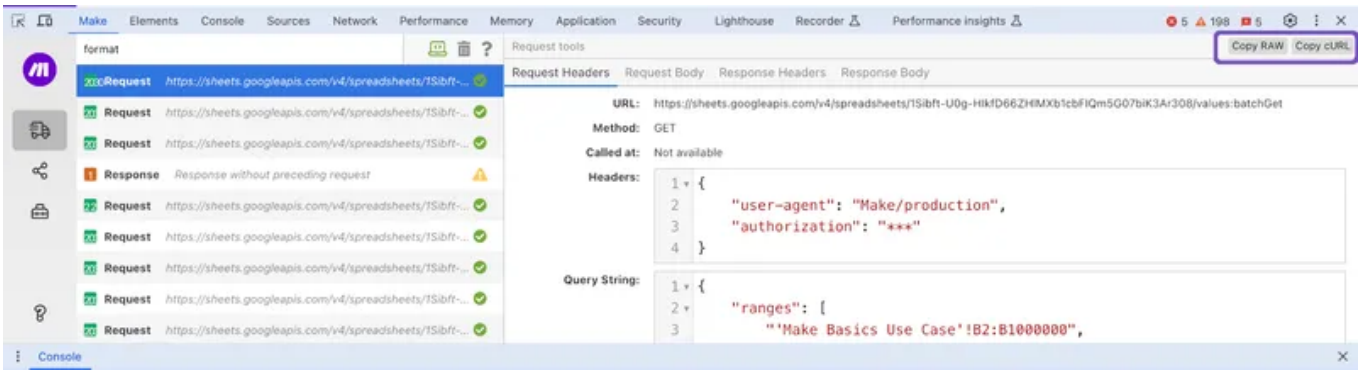
When logging is enabled, the color of the computer icon switches to green.



Click the same icon one more time if you want to disable logging. The icon turns gray when the feature is disabled.

Retrieve the request in the raw JSON format or cURL

To retrieve the raw JSON content of the request, click the **Copy RAW** button in the top-right corner of the DevTool's panel.



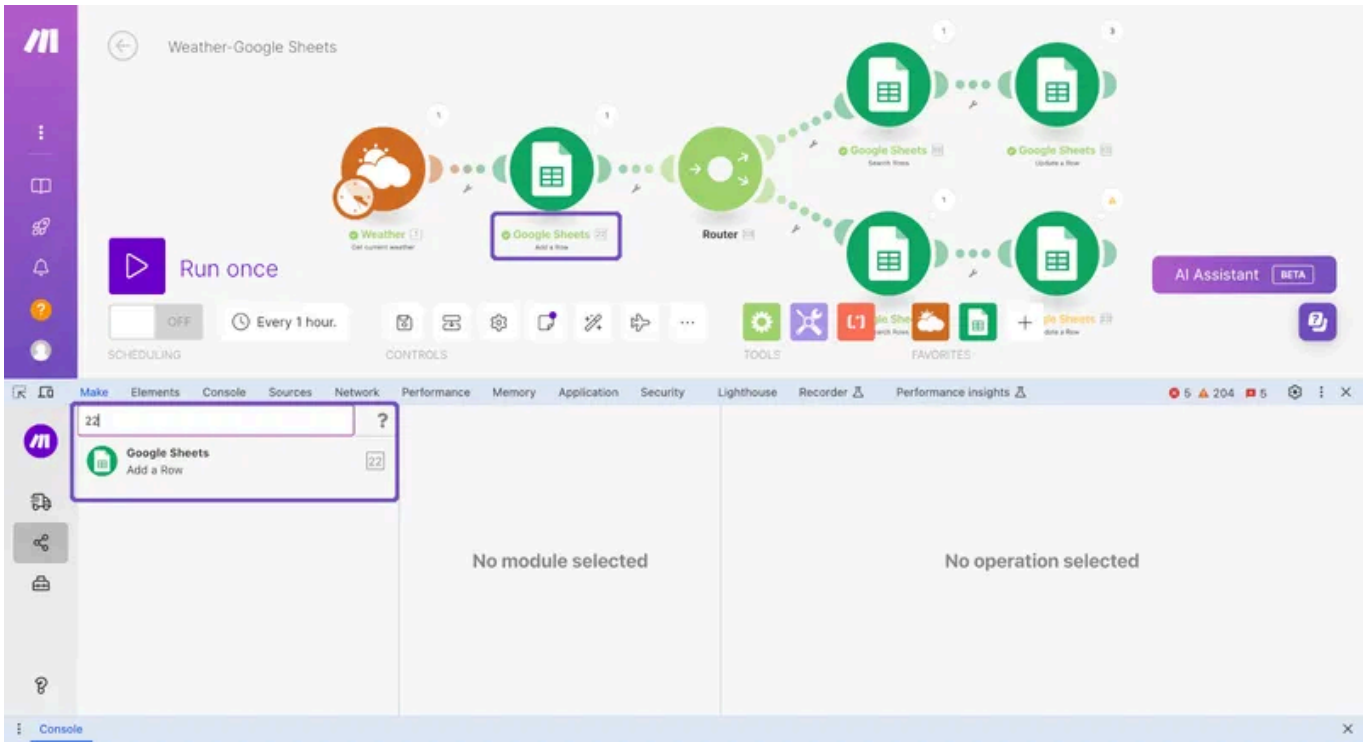
Similarly, you can retrieve cURL using the **Copy cURL** button next to **Copy RAW** in the top-right corner of Make DevTool's panel.

Scenario Debugger

Scenario Debugger shows you the historical logs of your scenario. The Scenario Debugger displays the history of the scenario runs and enables you to search for modules by their names or IDs.

Search for modules

To search for the module by its name or ID number, enter the search term in the search field in the left panel of the DevTool.

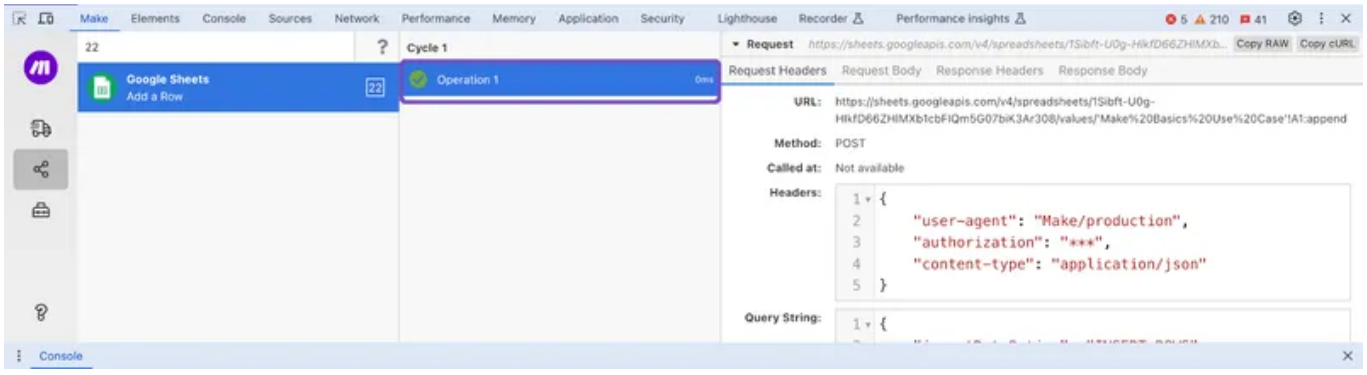


Open module settings

Double-click the module's name in the list to open its settings in the Scenario Builder.

View request or response details

You can open the request details by clicking the operation in the list.



Tools

The following set of tools makes setting up your {{}} easier.

Focus a module

Opens settings of the selected module.

> Module ID is the number you see in your Scenario Builder next to the name of the module. The same apps used multiple times in one scenario will have different IDs for each module.

Field	Description
Module ID	Enter the ID of the module for which you want to open settings and click the **Run** button

Find module(s) by mapping

Allows you to search modules' values for the specified term.

Field	Description
Keyword	Enter the term you want to search for and click the **Run** button. The numbers in the output are IDs of modules that contain the term you have searched for.
Use Only Values	You can enable this option to only search in module fields' values, or disable it to also search in module fields' names.

> The search is performed through the *name* and *label* parameters.

Get app metadata

Retrieves metadata of the app by the app's module name or ID. This is useful, for example, when you need to retrieve the app's version used in your scenario for the technical support or the developer of the app.

Field	Description
Source Module	Select the module for which you want to retrieve metadata.

Copy mapping

Copies values from the source module to the target module.

> Make sure you set the correct source and target modules because values in the target module will be overwritten. If you select different types of modules, values in the target module will be deleted.

Field	Description
Source Module	Select the module or enter the ID of the module from which you want to copy field values.
Target Module	Select the module or enter the ID of the module into which you want to insert the source module values.

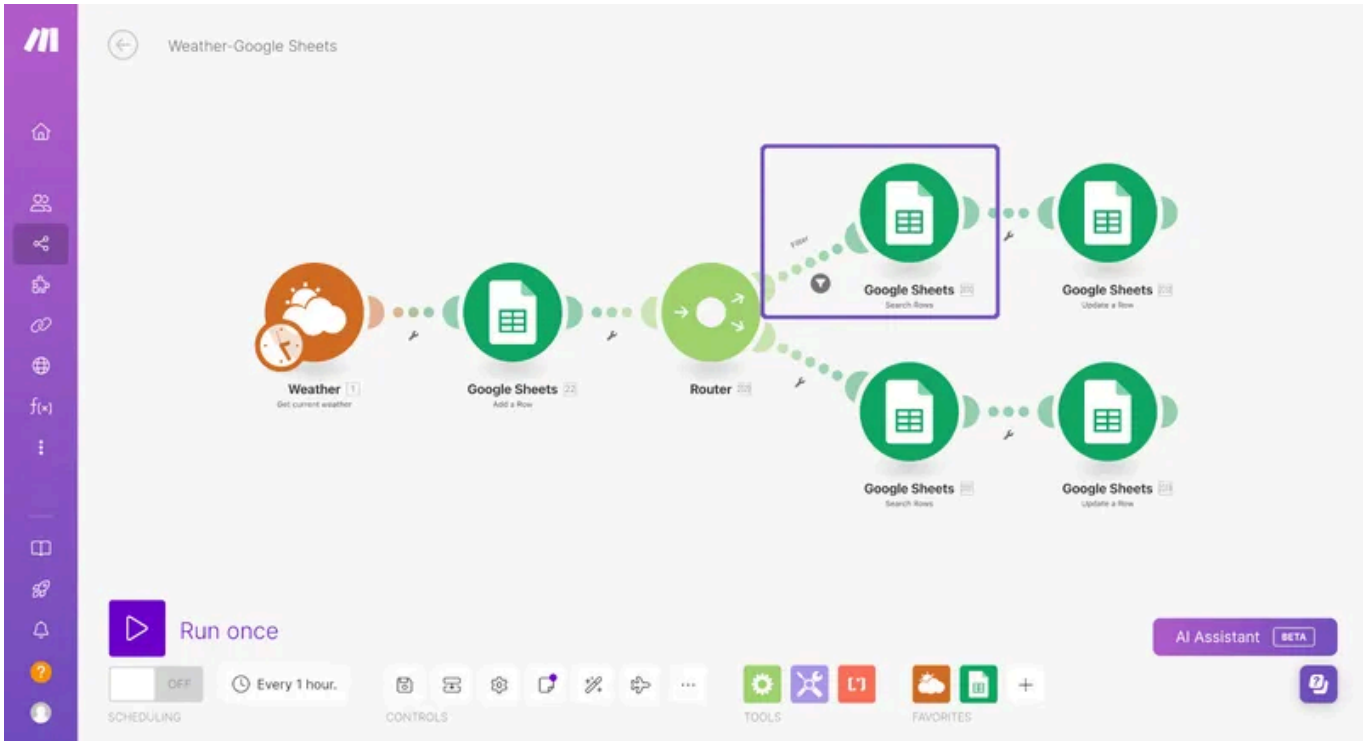
Switch on the **map** toggle to manually enter the ID of the modules for mapping.

When the source and target modules are specified, click the **Run** button to perform the action.

Copy filter

Copies the filter settings from the source module to the target module.

The copy action is performed on the filter placed on the left side of the selected module.



Field	Description
Source Module	Select the module or enter the ID of the module from which you want to copy filter values.
Target Module	Select the module or enter the ID of the module into which you want to insert the filter values from the source module. Note that the values in the target module will be overwritten.
Preserve Fallback Route setting	If enabled and the source filter is set as the fallback route, then the target filter is also set as the fallback route.

When the source and target modules are specified, click the **Run** button to copy the filter.

Swap connection

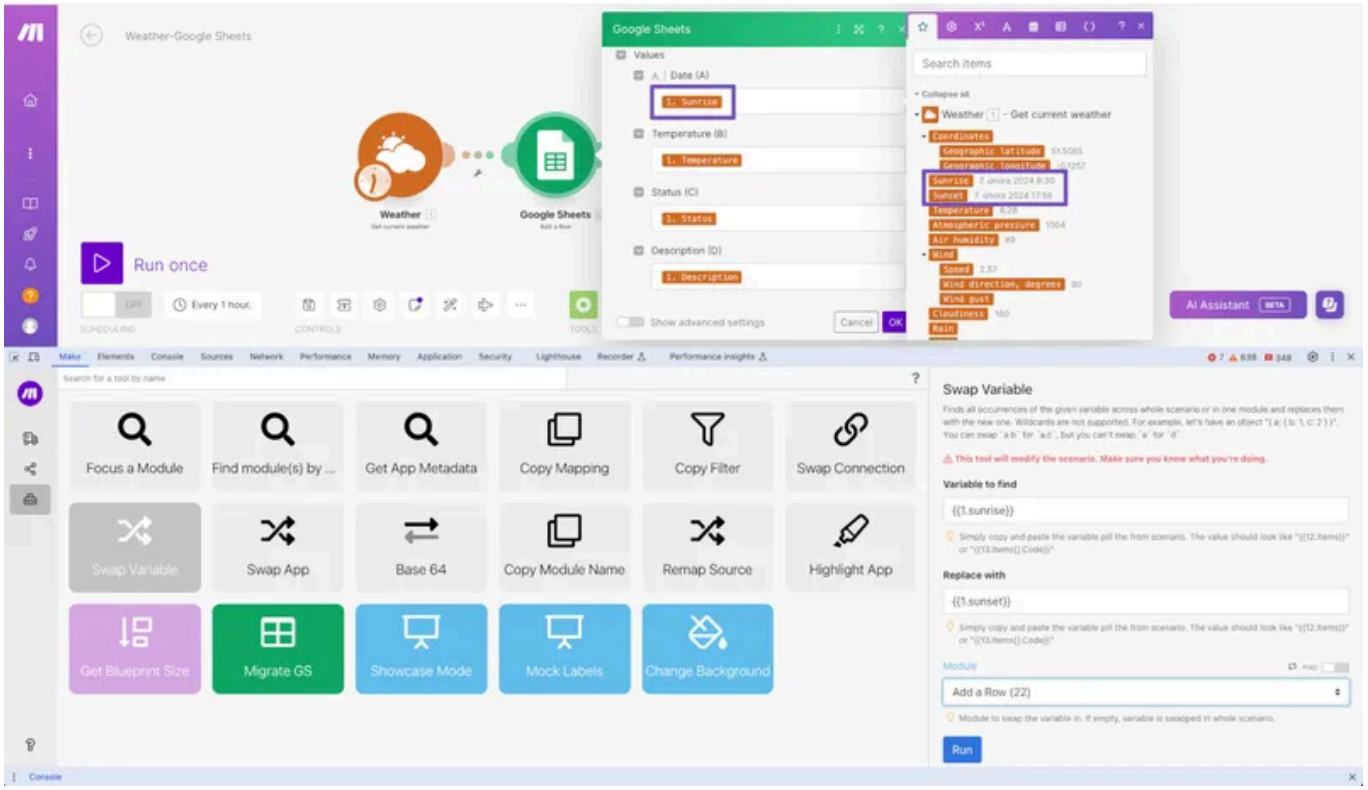
Duplicates a connection from the source module to every module of the same app in the scenario.

Field	Description
Source Module	Select the module or enter the ID of the module from which you want to duplicate the connection and set the same connection for every module of the same app in your scenario.

Swap variable

Searches for specified variables in the scenario and replaces them with a new variable.

Field	Description
Variable to Find	Select the variable which you want to be replaced from the module in your scenario and copy it to the **Variable to Find** field. You can also drag and drop the variable into the field. After it is done the format of the value in the field will look like this: `{{5.value}}`
Replace With	Copy and paste, or drag and drop the variable you want to use instead of the variable specified in the field above.The format of the value in the field will look like this: `{{5.value}}`
Module	Select the module in which you want to replace the variable. If no module is selected, the variable will be replaced in the entire scenario.



When all is set up, click the **Run** button to perform the action.

Swap app

Replaces the selected app version in your scenario with another app version.

This tool can be used, for example, to upgrade Gmail and Email apps' modules to the latest version.

> Before swapping make sure that the version you've selected supports all the modules and functions you might need for your scenario.

Base 64

Allows you to encode the entered data to Base64 or to decode Base64. This tool can be useful when you want to search for particular data in the encoded request.

Field	Description
Operation	Select whether you want to encode the data from the **Raw Data** field (below) to Base64 or decode Base64 to Raw Data.
Raw Data	Enter the data you want to encode to Base64 or Base64 you want to decode to raw data, depending on the option selected in the **Operation** field above.

When the input is specified, click the **Run** button to perform the selected action.

Copy module name

Copies the name of the selected module to your clipboard.

Field	Description
Module	Select the module whose name you want to copy.

When the module is selected, click the **Run** button to copy the module's name.

Remap source

Allows you to change the mapping source from one module to another. You can do it for the already existing module in your scenario as well as add a new one.

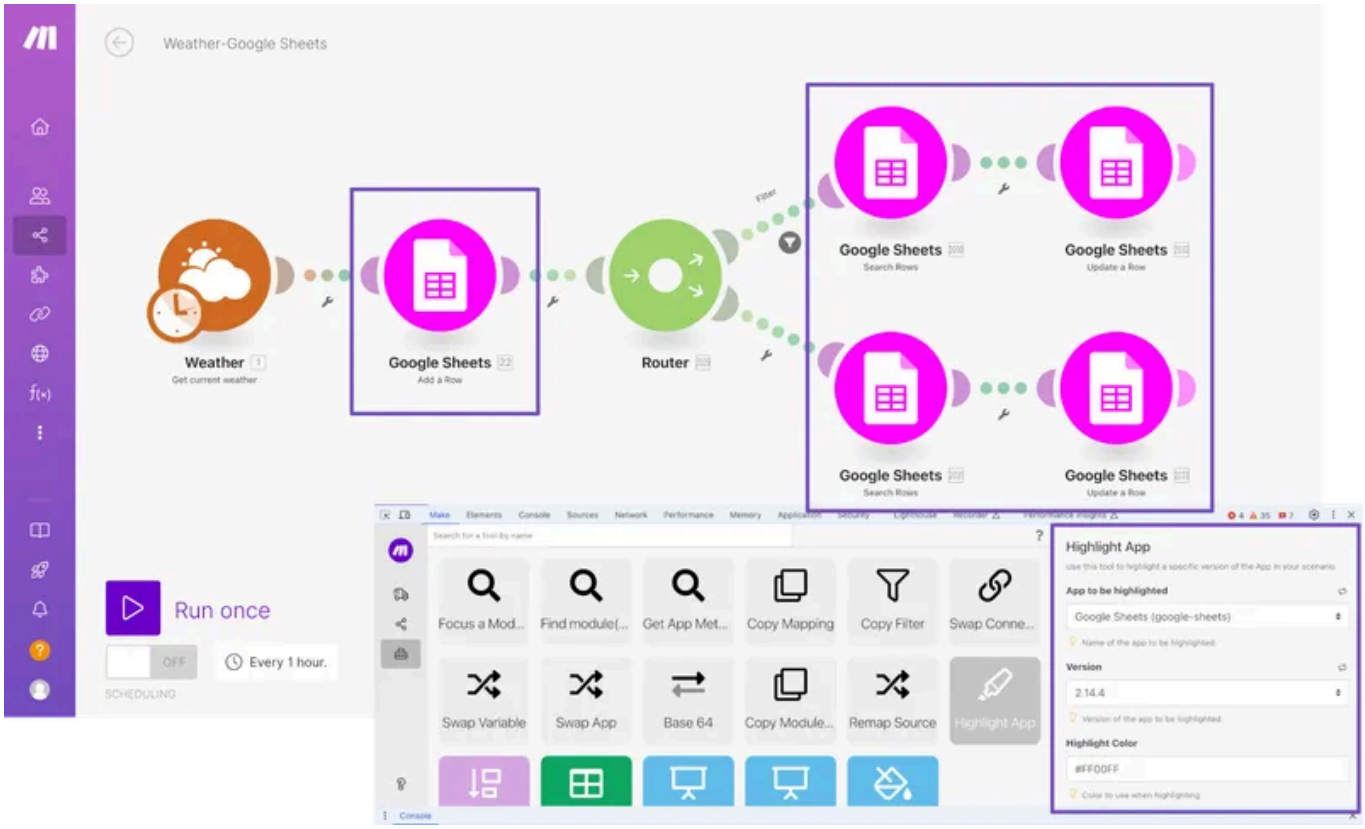
Field	Description
Source Module	Select the module you want to be replaced as the mapping source for other modules in your scenario.
Target Module	Select the module you want to use as a new mapping source.
Module to Edit	In case you don't want to change the mapping in the entire Make, select the module you want to change the mapping for.

When all is set up, click the **Run** button to perform the action.

Highlight app

Highlights modules of the specified app in your Make.

Field	Description
App to be Highlighted	Select the app you want to be highlighted in your Make.
Version	Select the version of the app you want to be highlighted.
Highlight Color	Enter the color hex you want to use for highlighting modules.



Migrate GS

This tool is made especially to upgrade Google Sheets (legacy) modules to the latest Google Sheets version. It adds a new version of the module just after the legacy version of the module in the scenario route.

4. Make AI Agents

Learn about agentic automation with Make AI Agents:

→ **Introduction to AI agents**

→ **AI agent best practices**

→ **Manage AI agents**

→ **Make AI agent reference**

→ **AI agent use case**

4.1. Introduction to AI agents

> Make AI Agents is available in **Open Beta** to all customers on paid plans. As a beta feature, both product functionality and pricing may change.

An AI agent is an autonomous system that acts to achieve its goals while following certain rules and conditions. Think of it like a skilled assistant who performs tasks for you after understanding what you want to do.

Make AI agents consist of a few fundamental components that can be thought of as a brain (LLM), instructions (system prompt), memory (context), and tools (scenarios and MCP tools):

- A Large Language Model (LLM) acts as the agent's brain, powering its reasoning for decision-making.
- The system prompt instructs the agent on how to behave. Like a job description, it defines its role, limitations, and processes.
- Context, or files stored in the agent's long-term memory, provides extra information that helps the agent tailor its responses to your needs. For example, an internal knowledge base or company policies.
- Tools provide the agent with the capabilities to do its job. The agent runs tools for you to complete your requested tasks. In Make, the agent's tools can include scenarios and MCP tools. Scenarios used as tools for agents must be scheduled on demand or triggered by a Custom webhook.

The agent uses tool names and descriptions to know which tool to use to achieve a goal. It uses **scenario input and input** names and descriptions to understand the data it receives or sends back to you.

In the next section, you will explore these concepts in more detail.

> The Make AI agent's reasoning is transparent and easy to review. However, LLMs are still evolving, which could impact agent outcomes. Always check outputs to ensure their accuracy and reliability.

4.2. AI agent best practices

AI agents are powerful AI systems that can run workflows for you with minimal human intervention. However, agents are only as good as their setup and instructions – a good design drives consistent and desirable outcomes.

This guide outlines tips for building effective AI agents in Make, including tool naming, prompting, data access, and more.

After reading, you'll have the foundational knowledge to design AI agents that can perform as expected.

Tools

If a Large Language Model (LLM) is an AI agent's brain, tools are its hands and actions – they give the agent the capabilities to do its job. Currently in Make, your agent's tools are scenarios and MCP tools.

Tool naming

An agent chooses the right tool for a task based on your tool names and descriptions. Write concise names and descriptions that reflect the tool's purpose and when to run it.

For example: "Add customer email to Customer Contacts table" as the scenario name and "This adds a new customer email address to the Customer Contacts table. Do not run this if '@' is missing" as the scenario description.

> Well-named tools are essential when sharing scenarios as tools to external AI systems, such as agents, through **Make MCP Server.** These systems rely on tool metadata (not the system prompt) to make the right decisions.

Input and output naming

Well-defined scenario inputs and outputs help the AI agent understand what data to receive and share. Inputs and output fields include:

- Name
- Type
- Description
- Default value
- Required (Yes/No)
- Multi-line (Yes/No)

Take advantage of this configuration to specify your requirements for inputs or outputs. For example: "customer_email" as the scenario input name and "Full name of the customer" as the description. It is a required text input.

You can also add stylistic requirements to input descriptions, if relevant. For example, you can specify that a string (text) input must be all lowercase, sentence case, all caps, and so on.

Debugging in tools

If you notice your agent returns suboptimal outputs, first check if the issue is with your tools. You can debug by adjusting tool-specific logic in tool names, descriptions, and input and output schemas.

For example, you can add a hint in your tool description about when the agent should run the tool ("Only run when you have a bank user ID").

Also, check that all scenarios with an agent are activated; scenario errors can cause deactivation. (For handling errors in scenarios, see **Error handling**.)

You can adjust the logic in the system prompt if that change applies to all tools.

Prompting

Make AI agent prompts serve two distinct but related functions: system instructions and user-agent communication.

The **System prompt** serves as the core instructions guiding agent behavior, including role, limitations, and processes.

Prompting (or communicating with) agents involves task requests and instructions. You can send prompts in:

- Scenario-based **Messages** in the **AI Agent > Run an agent** module
- The chat window, **Testing & Training**, in the **AI Agents** configuration tab

Global system prompts

Keep the system prompt broad enough to enable your agent to work across different scenarios (e.g., quoting, inventory, and customer support). An overly specific prompt can risk restricting your agent's effectiveness to a limited set of tasks.

For example, consider these two approaches to writing system prompts – specific or simple – for a content agent you expect to use across multiple content creation scenarios:

Specific and restrictive prompt	**Simple and global prompt**
"You're a content agent designed to generate Instagram content under 150 characters. Drawing from various sources, you will write captions for lifestyle brands. Use an upbeat tone, two relevant hashtags, and three emojis. End the caption with a question to the audience."	"You're a content agent designed to generate and refine text across formats and platforms. You can produce engaging, informative, or persuasive content (including short-form, long-form, outlines, and more) tailored to audience needs and platform requirements."

The specific prompt restricts the agent in platform (Instagram), form (character count, emojis, hashtags), and task (social media caption writing), which limits its ability to help with other content-related tasks like SEO-optimized blog writing.

Keep system prompts general and include more specific instructions in the relevant scenario's description or **Messages** field.

Redundancy in AI-assisted prompts

If you are not an advanced prompt writer, you may benefit from using the **Improve** button in **System prompt**, which enhances your prompt with AI. Overly lengthy prompts can increase the risk of LLM confusion or hallucinations; using AI to write prompts may contribute to this risk if you don't review its work.

Always review your AI-enhanced prompts for redundancies, such as repetitive or obvious instructions, and remove them. Similarly, edit or remove the Workflow section if it does not align with your process.

Additional prompts in tools

After writing your system prompt, you can add further specifications in tools. You can do this in **Messages** and **Additional system instructions** in the **Make AI Agents > Run an agent** module.

In **Messages**, you can write a prompt or input that tells the agent what to do in that specific scenario, including limitations and conditions.

For example, in a social media caption scenario, you can instruct the agent to limit the length to 150 characters, avoid complex punctuation like em dashes or semicolons, and request human validation for specific topics.

In the optional **Additional system instructions**, you can add extra contextual information that the agent wouldn't get from the user.

For example, if users call your agent through communication channels like Slack or Telegram, you can use this field to pass metadata such as user type, timestamp, and profile name from that channel.

Data security and access

LLMs are imperfect systems and still vulnerable to risk, such as PII disclosure in outputs or prompt hacking. The best precaution is to assume that anyone could successfully access what you share with your agent, including its tools and data.

Agent access to information

Agents only use the personal data you explicitly provide through tools and context files. To reduce data security risk, limit the agent's access to the data it needs to do its job, and avoid exposing it to sensitive, non-essential information.

For example, instead of connecting the agent to a tool that exports your entire calendar, its access could be limited to a public calendar, with only free/busy slots listed.

Limits of user-imposed constraints

An agent's system prompt includes limitations and constraints, or rules. Limitations guide the agent's behavior by defining desirable and undesirable actions. For example, you can instruct your AI agent to:

- Limit engagement with off-topic queries
- Avoid sharing sensitive information in outputs
- Flag harmful or unsafe inputs
- Prioritize internal knowledge bases as references
- Prompt a human to validate outputs

However, even with thorough, explicit guardrails, the agent could still behave unpredictably. In AI systems, constraints themselves have limitations: agents may not always follow them. Given the implications for data security, the most reliable constraint in this area is giving the agent minimal access to sensitive information.

Model configuration

In **Agent settings** in the **AI Agents** configuration tab, you can improve how your agent functions by setting limits on its output tokens, execution steps, and thread history.

Max output tokens

Agents can provide lengthy responses, which can get costly. Use **Max Output Tokens** to limit the number of tokens an agent outputs. If you set the maximum tokens too low, you may get incomplete responses. One token equals about 4 characters of text.

Steps per agent call

In rare cases, agents can get stuck in a loop, repeating the same actions without achieving their goal. Prevent this by limiting the execution steps an agent can take. A safe estimate for execution steps is to multiply your maximum expected number of tool calls by 10.

Max number of agent runs in thread history

Agents use thread history, similar to a chat history, to remember interactions with you. However, you may not want to keep the entire thread history. Doing so consumes additional tokens, as the agent needs to read the entire history during each execution. To save tokens, limit the number of previous messages the system stores. For example, if you only want to keep the last 5 messages, set this field to 5.

Knowledge enhancement

The LLM behind your agent doesn't know your internal knowledge and processes if they aren't publicly available. Improve your agent's decision-making by uploading private files to **Context** in the **AI Agents** configuration tab. These files serve as reference information for the agent.

Context files should contain fixed information that applies to all tools using an agent. For example:

- Company guidelines
- Internal knowledge bases (e.g., Confluence)
- Support tickets
- Community posts
- Company Slack conversations

Your files shouldn't include sensitive information when the agent will be used by users who shouldn't access that data.

When deciding on files to upload, consider the garbage in, garbage out (GIGO) principle: poor-quality or unrepresentative inputs lead to similarly flawed outputs. Upload information that reflects the results you want the agent to reproduce. For example, do not upload poorly executed work as examples to follow.

Cost optimization

When using Make AI agents, you incur fees for Make operations and LLM provider (e.g., ChatGPT) tokens.

Overall costs can rise quickly if using one thread ID, multiple tools, and advanced reasoning LLMs. Keep LLM costs low by optimizing your token usage and choosing a cost-

effective LLM.

Token usage

Token usage depends on two main factors: the amount of information the LLM processes in requests and the number of user-to-LLM interactions.

Data scope

To reduce the amount of information an agent handles in a request, specify the data you want to pass to the agent and the data you want it to return. For example, instead of instructing the agent to scan an entire database, narrow the scope of its search to entries after June 1st, 2025.

To limit the data handled in requests:

- Define scenario inputs and outputs.
- Filter inputs before passing them into the agent. In a scenario, you can add a filter in the route before the **AI Agents > Run an agent** module.
- Store files with global information (e.g., contact details) in the agent's **Context**.

Interactions

User-to-LLM interactions, which include your inputs and the LLM's outputs, also consume tokens.

To reduce interactions, prioritize clarity in prompts and consolidate data requests instead of making multiple sequential requests. These strategies enable complete, accurate outputs sooner.

Additionally, when you use the same thread ID across **Make AI Agents** module runs, the entire thread gets passed into the agent. If you don't need to reference your conversation history with an agent, leave the **Thread ID** field blank. This action creates a new thread ID, which can significantly save costs.

> You can monitor token usage in the **Make AI Agents > Run an agent** output bundle in **tokenUsageSummary**, and on your LLM provider account.

Choice of starter LLM

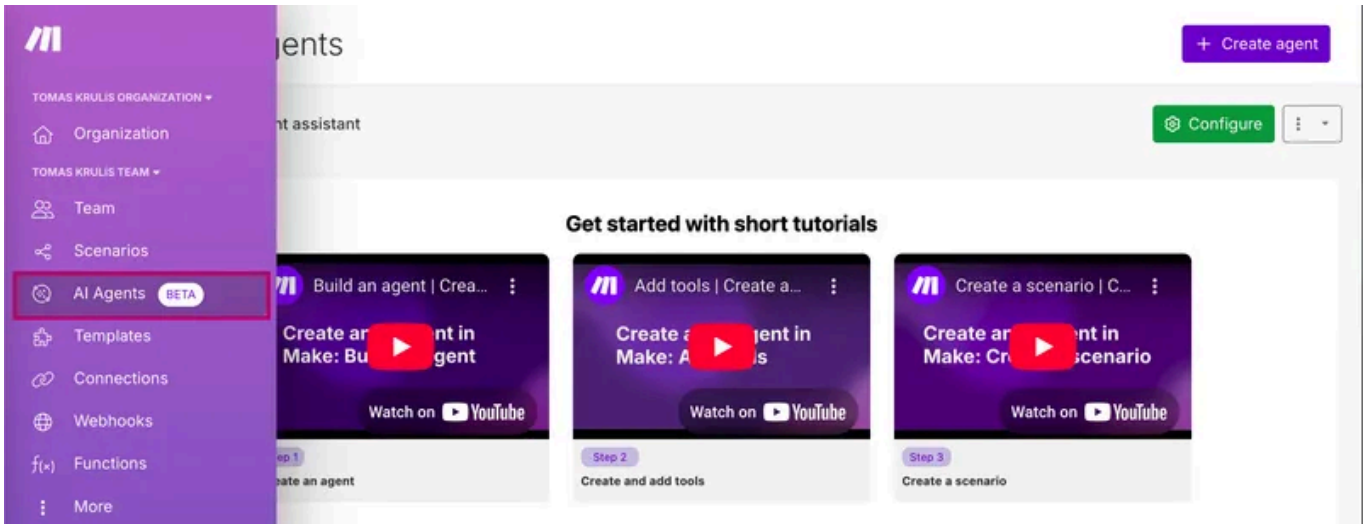
You have multiple LLMs to choose from for powering your AI agent, each with varying costs depending on factors like speed and reasoning abilities.

When creating an AI agent, start with an LLM with a good speed-to-cost ratio. The ideal starter model is fast and inexpensive, such as OpenAI's GPT-4.1 mini. More advanced, slower models, like OpenAI's o3, can rapidly increase costs. Test how well an affordable model can achieve your goals, and scale up as needed.

While a fast LLM performs well in many cases, the right model depends on what you want to do with an agent. Some LLMs are better suited for certain use cases than others, and many LLM providers classify their models accordingly.

4.3. Manage AI agents

When you create multiple AI agents, managing them through scenarios can become quite a challenge. To make managing AI agents easier, we added a new tab to the Make platform navigation called **AI Agents**.



The **AI Agents** tab shows you the list of AI agents that available in your team.

> AI agents are shared across all team members, like connections. If you want to have a private agent, use a team where you are the only member.

In the **AI Agents** tab, you can:

- Create and duplicate agents
- Configure agents
- Delete agents

Create agents

To create a new AI agent, click the button **Create agent** in the top right corner. A **Create agent** pop-up appears:

1. In the **Connection**, select your AI service provider connection or click **Add**. **Create a connection** pop-up appears.

1. In the **Connection type** drop-down, select your AI service provider.
2. Fill in the rest of the form according to the documentation for creating a connection to the AI service provider. Check the list of **AI service providers**.
3. Click **Save** to finish creating the connection.

2 In the **Agent name** field, specify name for the agent.

3 In the **Model** field, select the AI service model that the agent should use for reasoning.

4 In the **System prompt** field, define the agent's purpose and constraints. We recommend providing a short system prompt for the agent and leverage the **Additional system instructions** to customize the agent for a specific scenario.

> We recommend starting with a short system prompt and checking your resource (token) usage in your AI service provider.
>
> If your agent is using a lot of tools or is processing long descriptions your AI service usage might increase.

5 Click **Save** to confirm agent settings.

Next, to add the agent's tools and context, click **Configure** on the right-hand side of your agent in the **Make AI Agents** tab:

1 In **Context**, click **Add** to upload files that the agent can use.

Context allows you to add external information that enhances the agent's knowledge base, enabling it to effectively achieve its goal. Examples include internal knowledge bases or reference tables. The files in **Context** are stored in a database and enable long-term AI memory.

Current limitations for uploading in **Context**:

- TXT, PDF, DOCX, CSV, and JSON files
- 20 MB maximum per file
- 50 files maximum per team
- 100 files maximum per organization
- 20 files maximum per AI agent

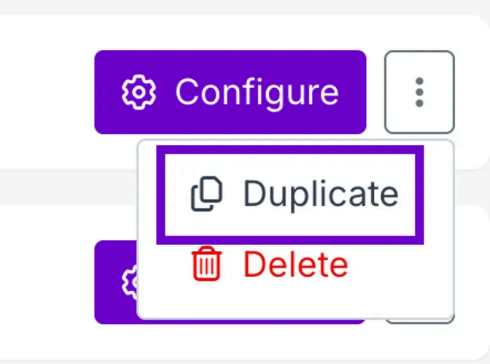
2. In **System tools**, click **Add** to select the scenarios the agent can use.

3. Click **Save**.

You have created an AI agent. You can now use it in a **Run an agent** module, or you can continue configuring it.

Duplicate agents

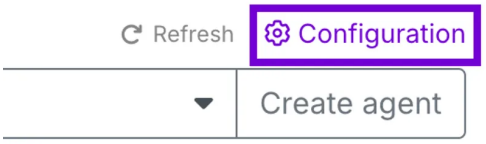
If you need to create a new agent based on an existing agent, you can create a duplicate of an agent to set up the new agent faster. To create a copy of an agent, click the three dots next to the **Configure** button and select **Duplicate**.



Configure agents

When using AI agents, you will need to adapt them to your processes or update them to improve their efficiency. To configure an agent, go to the **AI Agents** tab and click

Configure. Alternatively, you can click the **Configuration** link in the **Run an agent** module settings:



Options for configuring or reconfiguring your agent:

- Update agent's system prompt in **System prompt**
- Add background information or data for the agent to use in **Context**
- Add scenarios the agent can use as its tools in **System tools**

The scenarios you select in the agent's settings are always available to the agent, in addition to scenarios you add in the **Run an agent** module.

> Keep in mind that updating agent settings influences all **Run an agent** modules where you use the agent. If you would need to revert changes, you need to put them back manually.

To make further changes to the agent's model, click the **Agent settings** button. The pop-up form allows you to:

- Change the name of the agent
- Change the language model the agent is using if the AI service has different models
- If the AI service supports configuring the model, you can change the model configuration. The configuration options are available based on the agent's language model. Example configuration options:
    - **Maximum tokens**: The maximum amount of tokens that the agent can generate and send back to the **Run an agent** module.
    - **Maximum steps**: The maximum number of iterations the agent can run before providing the final reply to your request.
    - **Maximum history**: The maximum number of messages that the agent adds to its context from the communication thread. This setting influences the agent's context only if you specify the **Thread ID** in the **Run an agent** module settings.

> You can't change an agent's AI service provider or agent's connection. If you need to use a different AI service provider or if you need to change agent's connection, you have to create a new agent instead.

Delete agents

To delete an agent, navigate to the agent settings, click the button with the three dots and select **Delete**.

> Deleting an AI agent is irreversible. If you remove an AI agent, all modules that use the agent stop working.

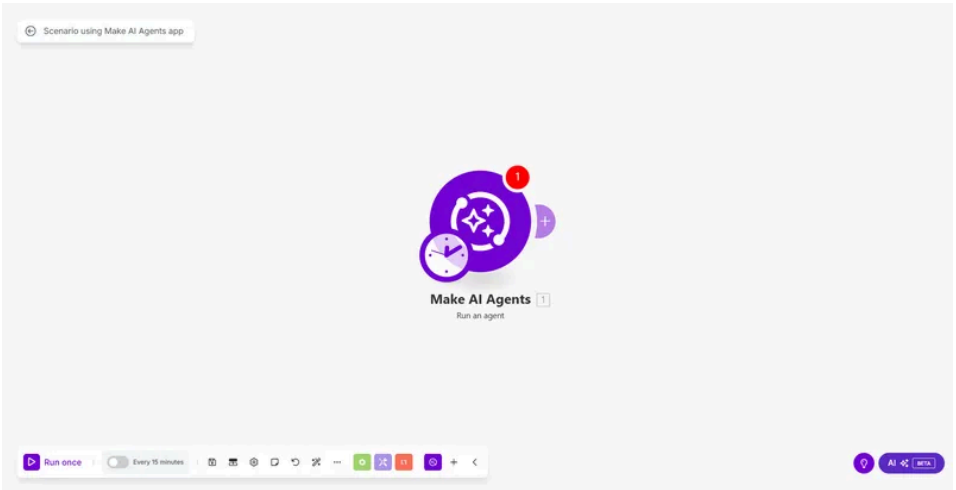
4.4. Make AI agent reference

This article contains reference information on Make AI agent in the **AI Agents** configuration tab and the **Make AI Agents** app.

- **AI Agents**, on the left sidebar, is where you can create and configure AI agents.



- **Make AI Agents**, in Scenario Builder, is the app for using your AI agent. It includes the **Run an agent** module.



AI Agents configuration

This section outlines the fields in the **AI Agents** configuration tab.

System prompt

The **System prompt** is your agent's core instructions. It outlines its purpose, behavior, and any guidelines or constraints it should follow. The agent follows these instructions across all workflows.

The **Improve** button on the lower right-hand side is an optional prompt improvement feature. Use it to improve your existing prompt with AI and enable more consistent outcomes. After you click **Improve**, you can leave the field empty for an auto-improvement or suggest specific improvements. Click **Save** to apply the changes.

Context

In **Context**, you can upload external information to improve the agent's knowledge base. Examples include internal knowledge bases or reference tables. Make stores the files in **Context** in a RAG database, enabling long-term AI memory.

Current limitations for uploading in **Context**:

- TXT, PDF, DOCX, CSV, and JSON files only
- 20 MB maximum per file
- 50 files maximum per team
- 100 files maximum per organization
- 20 files maximum per AI agent

You can also view your uploaded **Context** files in the **Make AI Agents** app in the Scenario Builder.

MCP

In **MCP**, you can connect your agent to MCP servers to access additional tools.

Click the **Add** button to create a connection to an available MCP server, then toggle on the tools that you want the agent to access. Click **Save**.

System tools

System tools are scenarios your agent uses to make decisions and complete tasks. These are tools the agent can always access across all workflows. Changing system tools impacts all scenarios using an agent.

When an agent processes requests, it uses tool names and descriptions to choose the right tool for the task. Similarly, it uses **scenario inputs and outputs** names and descriptions to choose the right data.

> Your agent uses tools at two levels: **System tools** in the **AI Agents** configuration tab for core tools, and **Additional tools** in the **Make AI Agents** > **Run an agent** module for scenario-level tools.

Testing & Training

Testing & Training is a chat interface that enables direct communication with your agent, eliminating the need to connect to external tools like Telegram or Slack. Use it to test, debug, and refine your agent's responses.

When you interact with your agent in **Testing & Training**, the agent can show you the tools it uses and their inputs and outputs.

You can click the **New chat** button to start a new chat.

Make AI Agents app

This section outlines the **Make AI Agents** app modules and their settings.

The **Make AI Agents** > **Run an agent** module allows you to:

- Create AI agents
- Provide additional scenarios to the agent
- Send messages to the agent, either as additional instructions to modify its context or messages to give it tasks

Agent

Select the agent you want to use or click the **Create agent** button to create a new agent.

If you want to create a new agent:

1. In the **Connection** field, select your connection.

2. In the **Agent name** field, fill in your agent's identification.
3. In the **Model** field, select the language model and version provided by your AI service provider.
4. In the **System prompt** field, define the agent's purpose and constraints. The system prompt defines the agent in all the scenarios it uses. In each scenario, you can further customize the agent's description with additional instructions.

Context	In **Context**, you can view the **Context** files the agent uses for reference.
Thread ID	In **Thread ID**, you can specify a custom thread ID to keep all of your interactions with the agent in the same communication thread. If you leave the **Thread ID** field empty, a new thread ID appears in the output.
Additional system instructions	In **Additional system instructions**, you can share extra contextual information with the agent. Dynamically provide context by mapping data from the scenario to this field.
Messages	In **Messages**, send tasks and specific requirements to the agent. You can send multiple user messages at once.
Timeout	In **Timeout**, set the maximum number of seconds the agent can execute until it is terminated. If you leave the **Timeout** field empty, the timeout is 300 seconds (5 minutes). The maximum timeout is 600 seconds (10 minutes).
Continue scenario run while agent is working	Enable this option to forward the agent's response to a specific URL.

Sometimes the agent might take a while to respond. If the **Run an agent** module doesn't get a response in 180 seconds, it would output the `ModuleTimeoutError` and stop the scenario. If you think the agent could take a long time to respond, enable this option.

In the **Webhook URL** field, specify a webhook URL. If you don't have a webhook ready:

1. Create a **custom webhook**.
2. Add the webhook to a scenario.
3. Set the scenario scheduling to **Immediately**.
4. Activate the scenario.
5. Paste the webhook URL in the **Webhook URL** field of the **Run an agent** module.

When the **Run an agent** module runs, it replies immediately with a notification that Make will forward the agent's reply to the specified webhook. When the agent finishes its tasks, the **Custom webhook** trigger receives its reply.

> With **Continue scenario run while agent is working** enabled, as the **Run an agent** module doesn't receive data from the agent, the data is unavailable for mapping in the scenario. The agent's reply is forwarded to the webhook. If you want to use the agent's results in a scenario, you need to move the processing to the scenario with the webhook.

4.5. AI agent use case

In the following use case, we will build an agent that helps us manage the stock inventory of a small shop.

We will use Slack to submit tasks for the agent and receive responses about their completion. The agent will use scenarios and other information we provide to him to list our stock information and order new items.

In this use case, we will:

1. **Set up the AI agent**
2. **Create the AI agent's tools**:
    1. Scenario to list shop inventory
    2. Scenario to order more stock
3. **Create a scenario to send tools to the AI agent**

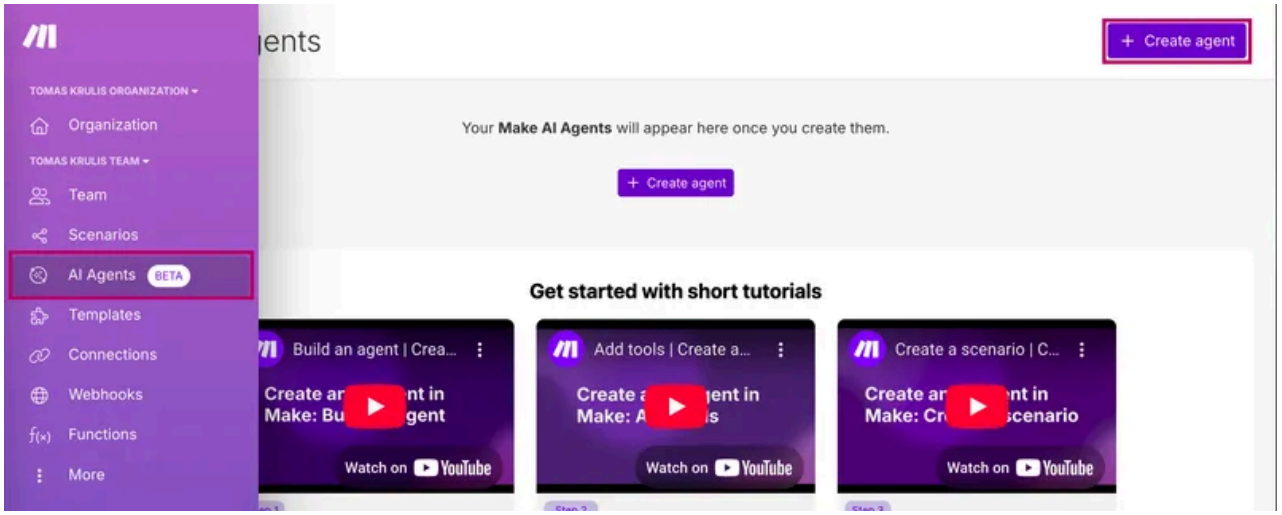
Prerequisites:

- API key of the AI service provider. Check the documentation of your AI service provider to find out how to get it.

4.5.1. Step 1. Set up the AI agent

First, we need to create the AI agent and connect it to the AI service provider. We can do that in the **AI Agents** tab.

1. Go to the **AI Agents** tab and click the **Create agent** button.



> Alternatively, you can create a new AI agent through the **Run an agent** module. In the **Run an agent** module settings, click the **Create agent** button.

2. In the **Connection**, select your AI service provider connection.
    1. If you don't have a connection to a AI service provider, click **Add**. **Create a connection** pop-up appears.
    2. In the **Connection type** drop-down, select your AI service provider.
    3. Fill in the rest of the form according to the documentation for creating a connection to the AI service provider. Check the list of **AI service providers**.
    4. Click **Save** to finish creating the connection.

3. In the **Agent name** field, specify the name for the agent.

4. In the **Model** field, select the AI service model that the agent should use for reasoning.

5. In the **System prompt** field, define the agent's purpose and constraints. For our shop inventory agent, fill in: "You are a stock inventory bot."

   We recommend providing a short system prompt for the agent and leverage the **Additional agent instructions** to customize the agent for a specific scenario.

6. Click **Save** to confirm the agent settings.

You have created an AI agent for managing your stock inventory. However, your agent doesn't have any means to do its job. In the next section, we create scenarios that work as tools for the AI agent.

4.5.2. Step 2. Create the AI agent's tools

AI agents in Make need tools to do their job – these tools are scenarios. Agents use information provided by you to reason about how and when to use these tools.

To inform the AI agent's context, Make sends the following information to the AI service provider:

- System prompt
- The name and description of each tool (scenario)
- The name and description of **inputs or outputs** of these tools

In addition to providing context, all scenarios used as tools for AI agents must be **active** and either scheduled **on demand** or triggered by a **Custom webhook.**

In the next sections, we will create a tool for our agent to list our shop inventory and another to order more stock if we're low.

> The aim of the following sections is to showcase the AI agent's reasoning. The example scenarios provided have been simplified to streamline their setup.

Tool 1: Scenario to list shop inventory

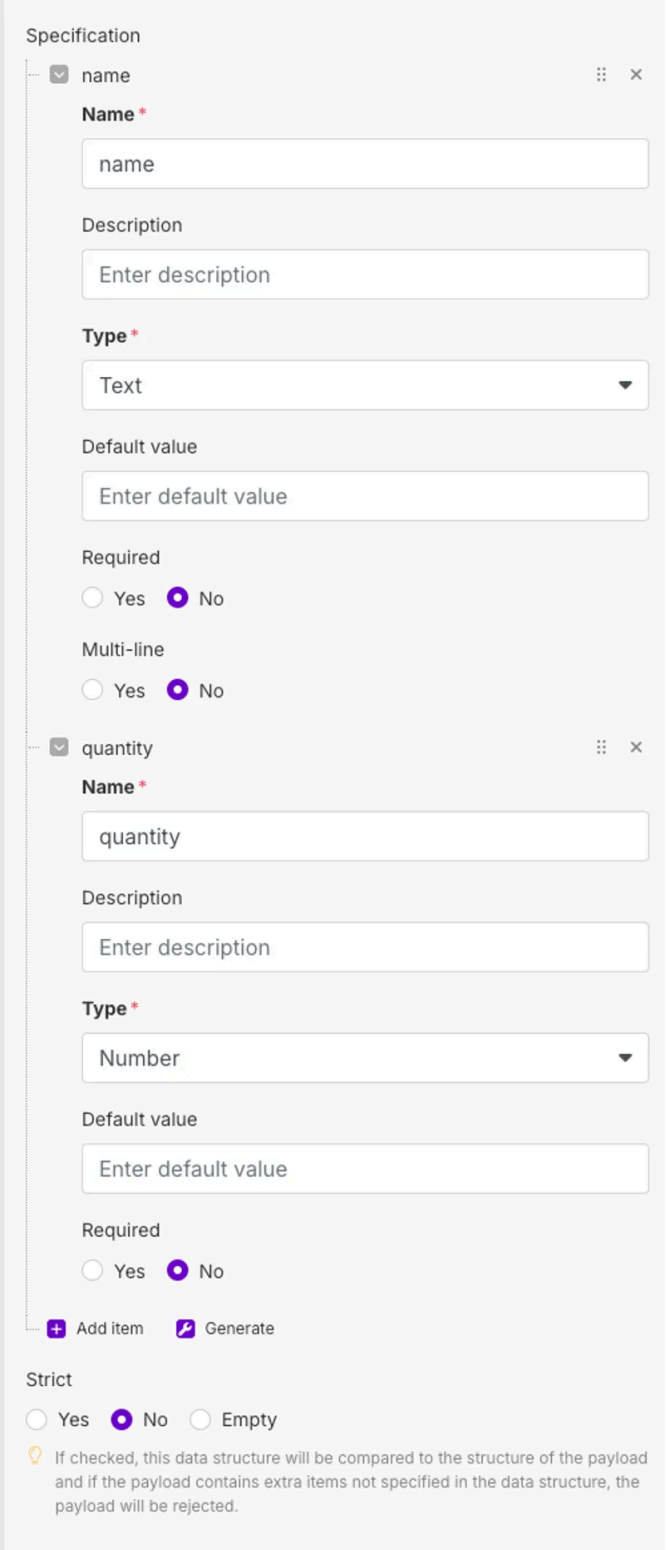
We will provide our agent with a scenario to list our shop inventory. Since we want to send data from the scenario to the agent, we have to use **scenario outputs** and the **Scenarios** > **Return output** module.

To create the scenario:

1. Click the **Create a new scenario** button in your organization dashboard or in the list of scenarios.

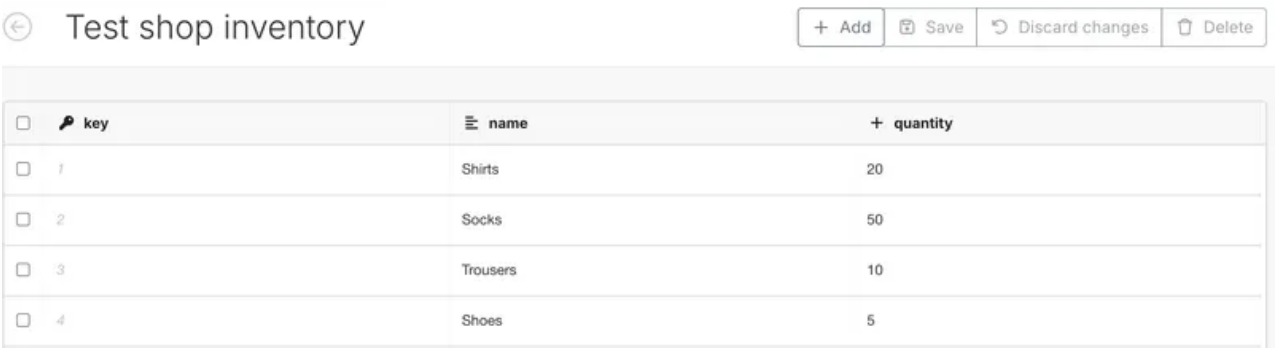
2. If you don't have testing data ready, set up your shop data:

    1. Add the **Data store** > **Search record** module to your scenario.

2. In the **Data store** field, select **Add** to create a new data store.
3. In the **Data store name** field, fill in the name for your inventory data storage.
4. In the **Data structure** box, click **Add** to define a structure for your data store.
5. 

6. Click **Save** to confirm the data structure and **Save** to create the data store.
7. Go to the list of data store and open the new data store.

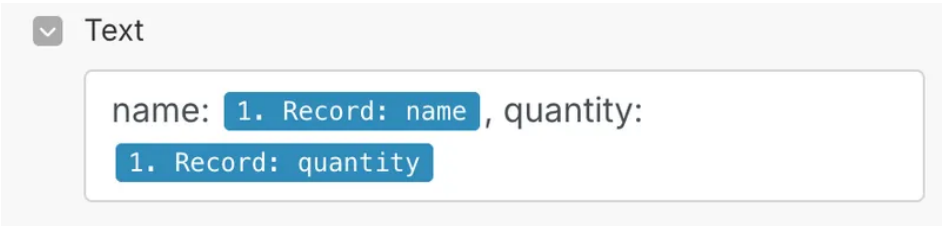
8. Click the **Add** button to add data to your data store:



3 Update the name of the scenario. The AI agent uses the scenario name to decide if it should run the scenario. Fill in: "List shop inventory".

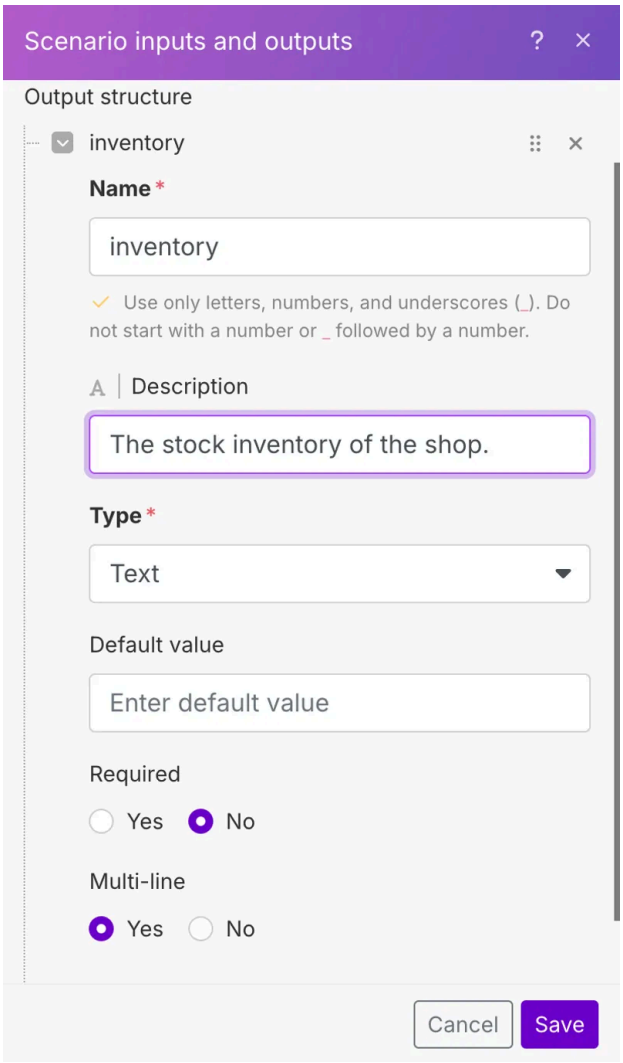
4 Add the **Tools** > **Text aggregator** module

1. In the **Source module** field, keep the data store module.
2. Enable the **Show advanced settings** toggle at the bottom of the module settings.
3. In the **Row separator** field, select **New row**.
4. In the **Text** field, map the `name` and `quantity` fields from your data store:



5 Add the **Scenarios** > **Return output** module.

1. Click **Add scenario outputs** to set scenario outputs for the return data module.
2. In the **Scenario outputs** tab, define the scenario output structure and fill in the `description` of each item in the output:
3.



4. In the module settings, map the `text` variable to the `inventory` field.
5. Confirm module settings with **Save**.

6 Set scenario scheduling:

1. Set the scenario scheduling to **On demand**. The agent will run the scenario for you when needed.
2. Confirm the scenario scheduling with **Save**.
3. Activate the scenario.
4. Save your scenario.

7 Add scenario description for the agent:

1. Go to the scenario **Diagram** tab.
2. Click **Options** > **Edit description**.

3. Add description to the scenario. The AI agent uses the scenario name and description to decide if it should run the scenario. Fill in: "Lists the shop inventory."
4. Click **Save**.

You have created the scenario for your agent to list the shop inventory. We will make the scenario available to your agent in the following sections.

Tool 2: Scenario to order more stock

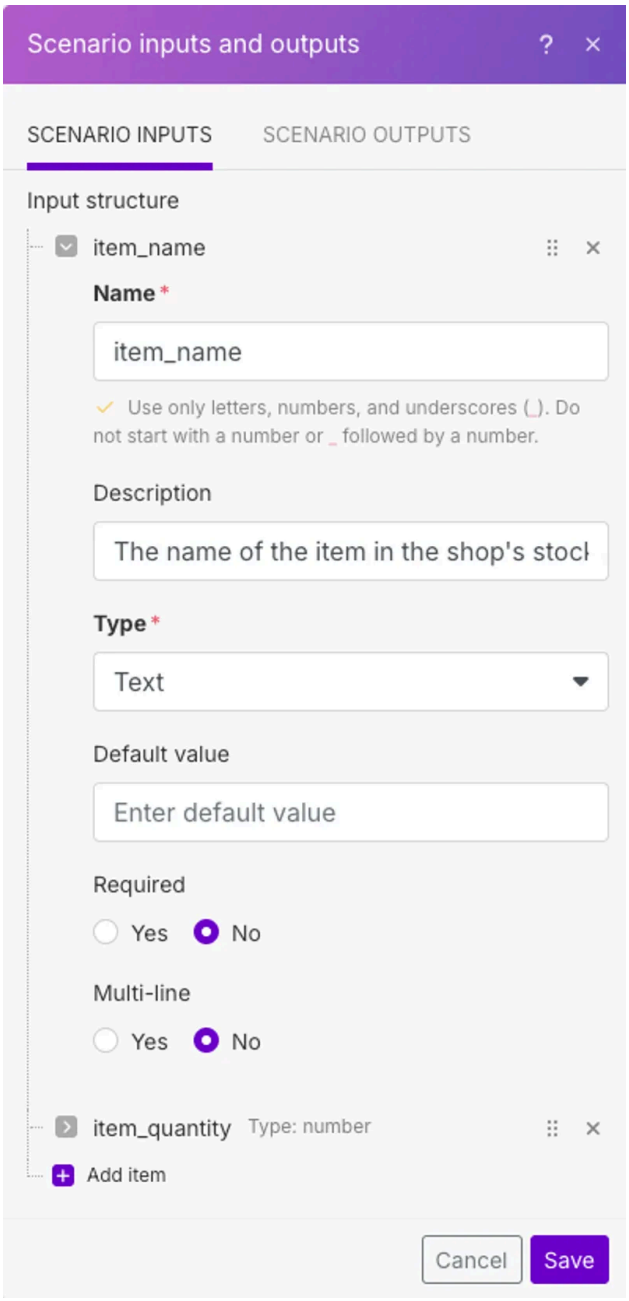
We will provide the agent with another tool: a scenario to order more items for our shop inventory. The scenario will receive order information from the agent with scenario inputs.

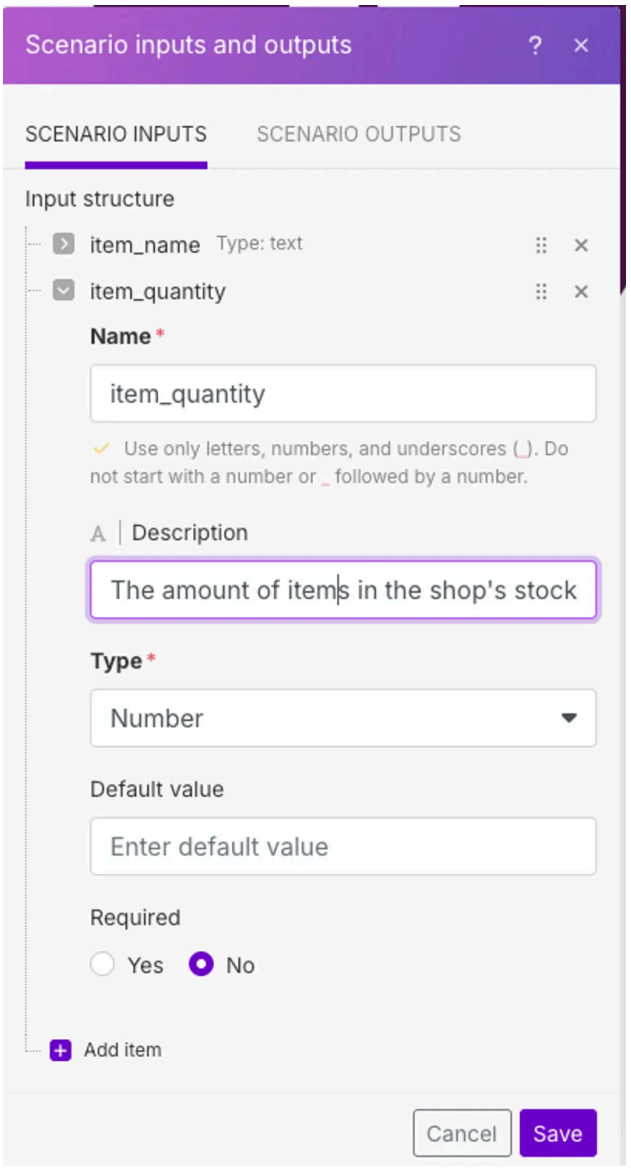
For our testing purposes, the scenario will just send messages to a selected chat.

To create the scenario:

1. Click the **Create a new scenario** button in your organization dashboard or in the list of scenarios.

2. In **Scenario inputs and outputs** in the **Scenario inputs** tab, set the scenario inputs structure and description:

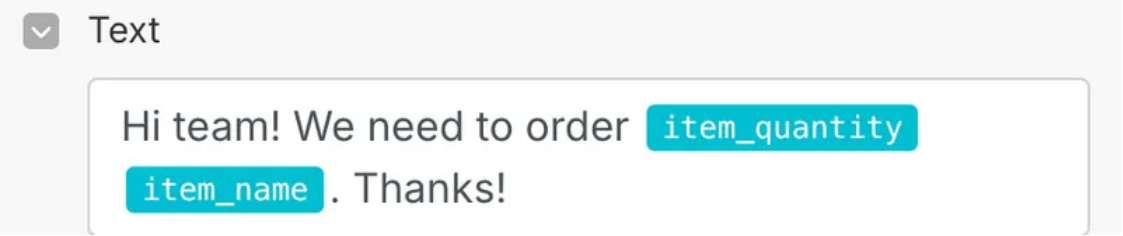




3 Add the **Slack** > **Create a message** module.

4 Set up the **Create a message** module:

1. In the **Connection** selection box, select your connection. If you don't have a connection, click the **Add** button to create it.
2. Select the channel where you want to receive the messages about new orders created by the agent.
3. In the module settings, use the scenario input in the **Text** field. For example:



5 Update the name of the scenario. The AI agent uses the scenario name to decide if it should run the scenario. Fill in: "Create buy stock order."

6 Set scenario scheduling:

1. Set the scenario scheduling to **On demand**. The agent will run the scenario for you when needed.
2. Confirm the scenario scheduling with **Save**.
3. Activate the scenario.
4. Save your scenario.

7 Add scenario description for the agent:

1. Go to the scenario **Diagram** tab.
2. Click **Options** > **Edit description**.
3. Add description to the scenario. The AI agent uses the scenario name and description to decide if it should run the scenario. Fill in: "Creates orders to refill the shop inventory."
4. Click **Save**.

You have created the scenario for your agent to create orders to refill the shop inventory. We will make the scenario available to your agent in following sections.

4.5.3. Step 3. Create a scenario to send tasks to the AI agent

In this final step, we create the scenario to communicate with our agent. We will equip the agent with scenarios as tools to perform the agent's tasks: list our shop inventory and order more items. We will use Slack to send tasks to our agent.

1. Click the **Create a new scenario** button in your organization dashboard or in the list of scenarios.

2. Add the **Slack** > **Watch new events** module to the scenario.

3. Set up the module:

    1. In the **Webhook** field, select a Slack event webhook or click **Add** to create a new one.
    2. In the **Event type** drop-down, select **New channel message**.
    3. In the **Connection** field, select your Slack connection.
    4. In the **Channel** field, select a channel you can use for testing the AI agent.
    5. Set the scenario scheduling to **Immediately when data arrives**.
    6. Click **Save** to confirm the module settings.

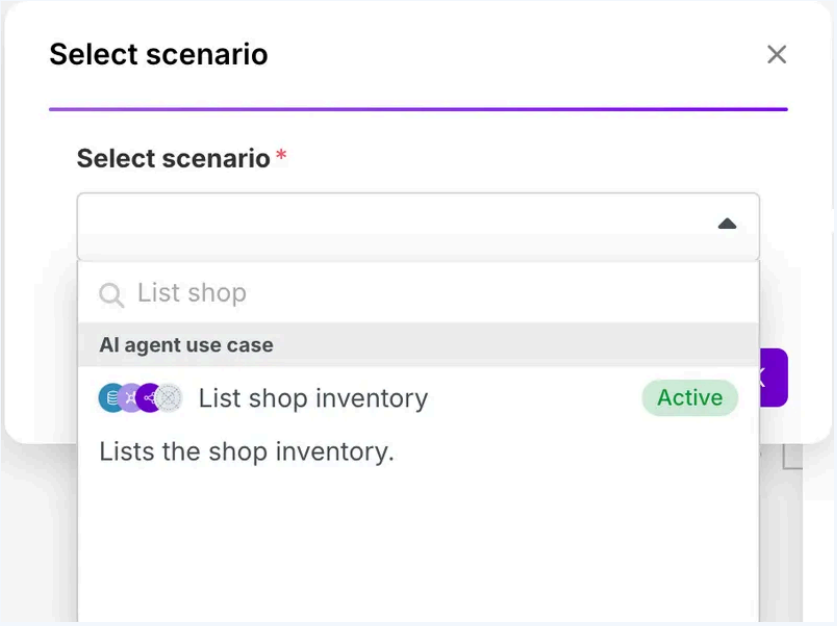
4. Click the module placeholder to add a new module to your scenario. Search "AI agent" to get the **Make AI agent** > **Run an agent** module and add it to your scenario.

5. In the **Agent** field, select the agent you created in the **previous section**.

6. In the **Additional tools**, click **Add tool**.

7. Add the scenarios to the AI agent's tools:

1. Click the empty drop-down under **Select scenario**. The list of scenarios in your team rolls down. Select the "List shop inventory" scenario we created in the **previous section**.

> Use the search row at the top of the selection list to find the scenario. Note the scenario description that appears under the name of the scenario and the name of the scenario folder.
>
> 
>

1. Click **OK** to confirm the selection.
2. Add the "Create buy stock order" scenario created in the **previous section** in the same way.
3. Click **Save**.

8 Finish setting up the **Run an agent** module:

1. In the **Thread ID** field, map a unique identifier to keep the same thread for all communication with the agent. This optimizes the context sent to the agent with each module run and allows the agent to keep the history of previous conversations.
2. In the **Message 1** field, map the `Text` from the **Watch new events** module.
3. Click **Save** to confirm changes.

9 Add the **Ignore** error handler to the **Run an agent** module. Sometimes, the agent can take time to respond.

> If the response time gets longer than 180 seconds, the module outputs the `ModuleTimeoutError` and Make stops the scenario.

There are more sophisticated options to prevent the `ModuleTimeoutError`, but they are not part of this use case. If you want to learn more, check the module settings reference.

10 Add the **Slack** > **Create a message** module after the **Run an agent** module.

11 Set up the module:

1. In the **Connection** field, select your connection.
2. In the **Enter a channel ID or name** and **Channel type** fields, select values relevant for the channel where you want to send agent's replies and pick the channel in the next field.
3. In the **Text** field, map the `response` from the AI agent.
4. Click **Save** to confirm changes.

12 Update the name of your scenario.

13 Save the scenario.

You should now test your agent. Send a message to the channel that is watched by the **Watch new events** module that contains the intent for the agent to either send the inventory information or order new stock. For example, request the agent to list the current shop inventory.

Check the channel where is the **Create a message** module sending agent's replies. You should see a reply from the agent listing the same data as we defined in the data store when we were setting up the scenarios available to the agent.

5. Error handling

Dive into resolving errors and keep your scenarios running smoothly:

→ **Introduction to errors and warnings**

→ **How to handle errors**

→ **Error handlers**

→ **Types of errors**

→ **Types of warnings**

→ **Exponential backoff**

→ **Throw**

5.1. Introduction to errors and warnings

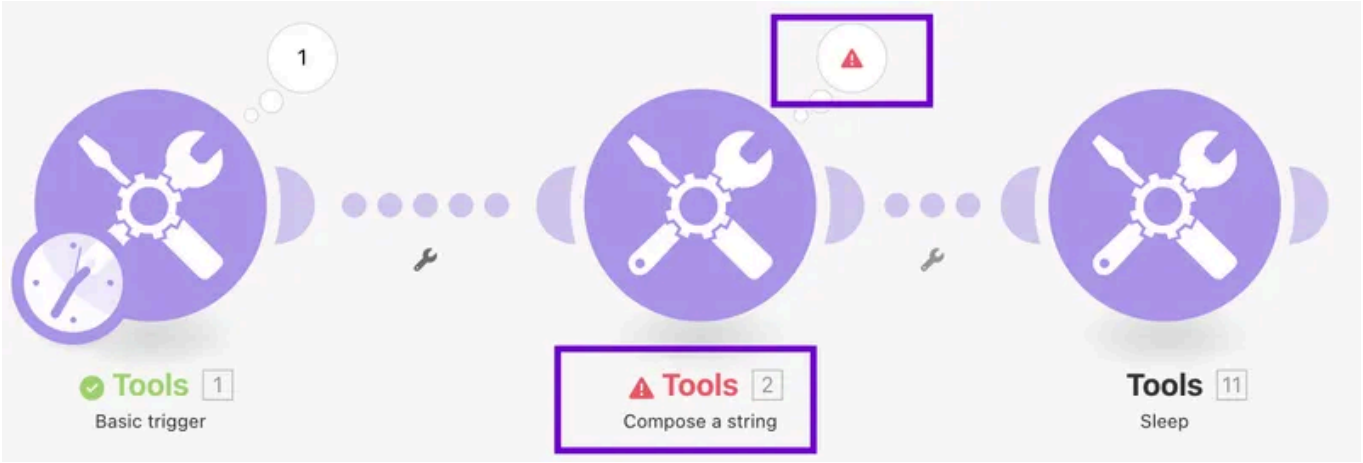
Sometimes, automation doesn't go the way you planned and takes a wrong turn. When this happens, Make gives you a warning or an error based on the situation.

You can also assume another point of view on errors and warnings: They protect your scenario by preventing the processing of unexpected data, and therefore saving operations usage.

Errors in Make

Errors notify you that your Make encountered an unexpected event that is not handled by an **error handler**. Because of the situation, you should check the scenario.

A module outputs an error when the module receives incorrect data from the previous modules or the module app. When you open your scenario in the scenario editor, Make highlights the module that outputs the error with the "Caution" sign:



When a module outputs an error, Make stops the current scenario run and starts the **rollback scenario phase**. The rollback phase reverts changes if possible and puts back the **polling trigger epoch** to the time before the scenario run.

When there are **consecutive** scenario runs that end with an error, Make disables further scheduling of the scenario. Disabling the scenario allows you to check the error and prevents consuming operations on scenario runs that finish with an error.

The most common situations when a module outputs an error include:

- Mapping a value to a required field in a module when the value is sometimes empty and causes missing required data.
- When you exhaust your resources in the third-party app. For example, when you can't store more data in the app.
- When the app is unavailable. For example, when the app is down for maintenance.
- When there is a change to your authentication or authorization in the app and you don't update your connection. For example, when your API key expires or when you change teams and lose access to some of the app features.

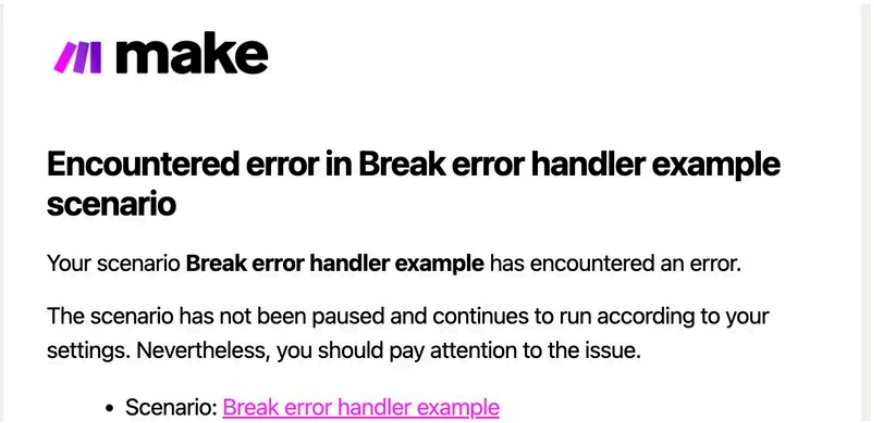
The best way to deal with errors in your scenario is to use an error handler. An error handler connects to a module with the error handling route. When the module outputs an error, the error handling route activates and runs the error handler.

When all errors are handled, Make keeps scheduling scenario runs instead of disabling the scenario.

For more information, check the **Overview of error handling in Make** or pages about **error types**, **error handlers** or specific **error handling strategies**.

Error notifications

When an error happens and the error is not handled by any error handler, Make sends you an email notification:



Make also sends out a notification when your scenario gets disabled because of repeated errors.

You can learn more about Make email notifications and their settings **here**.

Warnings in Make

Warnings alert you that there was an issue during the scenario run, but not as serious as an error. Also, scenarios can have the warning status when there were errors handled with your error handling.

When a module in a scenario returns a warning, your scenario keeps running and stays enabled. But it's a good idea to check the **scenario execution history** for the cause of the warning.

The situations when you get a warning include:

- When a module outputs an error, but you have enabled the **storing of incomplete executions** in the scenario settings.
- When you use up all of the capacity of a data store in your scenario.
- When the duration of the scenario run exceeds the time limit for your subscription.

Learn more about warnings **here**.

5.2. How to handle errors

→ Overview of error handling

→ Fixing rate limit errors

→ Fixing connection errors

→ Fixing missing data errors

5.2.1. Overview of error handling

{}When you are creating scenarios, you might encounter unexpected data or unexpected circumstances. Make notifies you about these events with an error to keep your scenarios reliable and functioning.

You can use error handlers to deal with errors or unexpected events in your scenario. An error handler connects to a module with the error handling route. When the module outputs an error, the error handling route activates and runs the error handler.

If an error occurs that is handled by an error handler, Make keeps scheduling your scenario because Make can assume that you anticipated the situation and prepared for it.

There are 5 error handlers in Make:

- **Ignore**
- **Resume**
- **Commit**
- **Rollback**
- **Break**

The error handling route

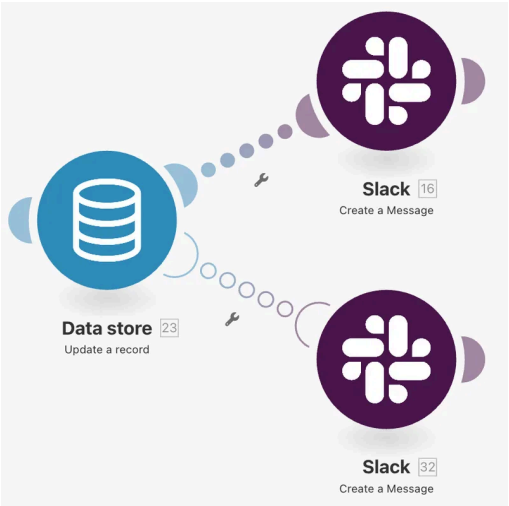
An error handler is always at the end of the error handling route. The error handling route has a transparent filling:

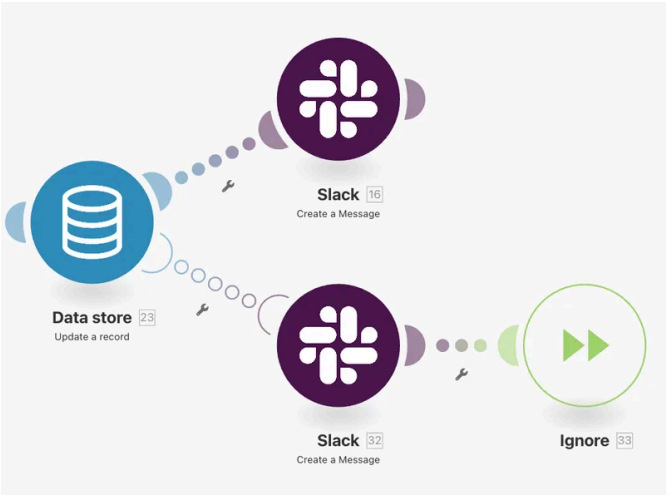


When an error handler activates, it doesn't consume operations. Make doesn't bill you for handling unexpected events in your scenario.

The error handling route doesn't have to have an error handler. For example, in the error handling route, there can be only a **Slack** > **Create a message** module to send you a Slack notification when an error happens.

If no module outputs an error in the error handling route, Make ignores the error. That means that the two error handling routes on the pictures work the same:



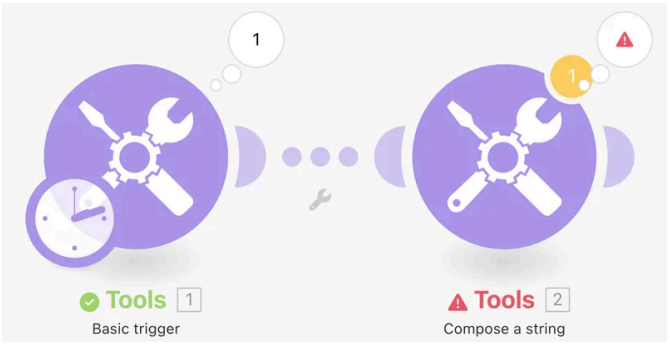


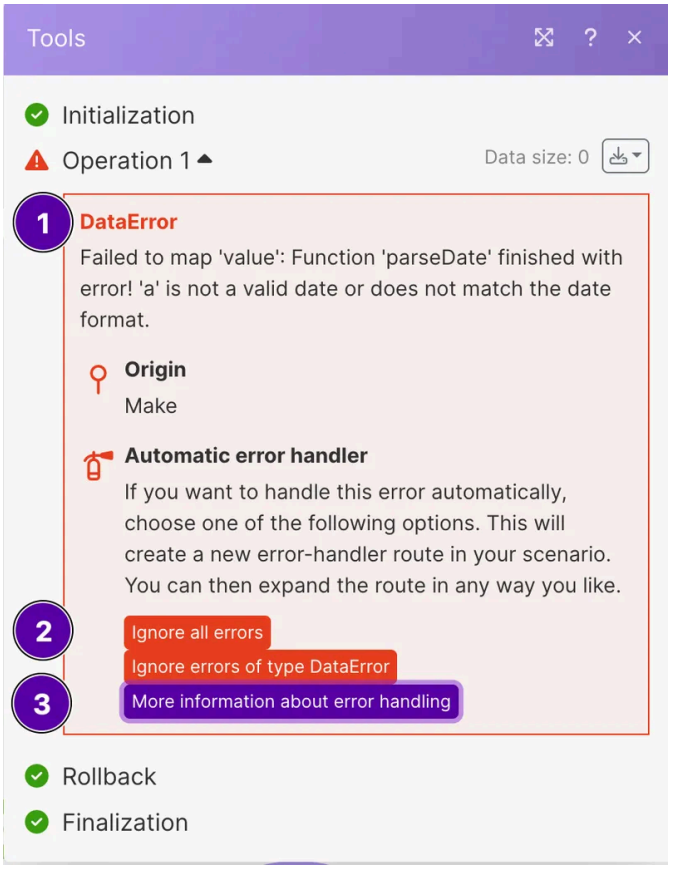
If a module in the error handling route outputs an error, the scenario run ends with an error.

How to identify errors

When you are building or checking your scenario in the Scenario Builder, Make highlights the module that caused the error with a warning sign in front of the module name and in the list of bundles.

When you click the bubble with the warning sign, you can check the bundle that caused the error in the module:





1. Shows the **error type** and the error message.
2. The red fields are quick action buttons. You can click the buttons to connect the **Ignore error handler** to the module.
   The first button connects the **Ignore** error handler with the module, to ignore all errors the module outputs.
   The second button inserts the Ignore error handler with a filter. The filter only allows errors that match the current error type to pass through. With the example on the pictures above, Make would ignore only the `DataError`.
3. The purple action button opens the Make Help Center in a pop-up window.

How to approach error handling

You have multiple options on how to handle errors in Make. Make handles some of the most frequent errors by default, and you'll typically need custom error handling only if you need to deal with a specific issue.

The most frequent errors in running scenarios are the `RatelimitError` and `ConnectionError`. You can rely on the default error handling for these errors if you enable **incomplete executions**. The details of the error handling of these errors are in their **dedicated articles**.

If you need to handle a different error type or if you need to customize the default error handling, you should consider:

- How important is the data the scenario processes?
  For important scenarios, Make can store partial scenario runs in incomplete executions. You can resolve the incomplete scenario runs manually one by one or retry multiple of them at once.
- What type of error does the module output and how frequently?
  If the error occurs rarely and it's a temporary error, like the `RateLimitError`, you can rely on the default scenario error handling. But if the error is critical, like the `InvalidAccessTokenError` or `the InconsistencyError`, you should set up error handling.
- What is the impact of the error?
  If the error has no impact on your data or your processes, you could ignore the error with the **Ignore error handler**. If the error has a high impact on your processes, you should consider enabling incomplete executions in scenario settings.

For ideas about specific error handling strategies, you can check the **list of dedicated articles**.

Scenario settings that impact error handling

The scenario settings play a key role in error handling. The following list focuses on how some scenario settings influence error handling. For more info about Make settings, check the **dedicated article**.

Allow storing of incomplete executions

Enable this option to store the incomplete scenario run when a module in the scenario outputs an error.

With incomplete executions enabled, Make stores the scenario state when the error happened as an incomplete execution. You can then check the scenario run, investigate why the error happened, and fix it to finish the scenario run successfully. In addition, all scenario errors turn into warnings.

- When you use the **Break error handler** in your scenario, you have to enable incomplete executions.
- Make doesn't store the incomplete scenario run in these conditions:
  - When the error happens on the first module in the scenario.
    However, you can add the **Break** error handler to the first module in the scenario. With the **Break** error handler, Make stores the incomplete execution even when the first module in the scenario outputs an error.
  - When your incomplete executions storage is full. If your incomplete executions storage is full, Make checks the **enable data loss** setting:
    - If the data loss is disabled, Make disables the scenario.
    - If the data loss is enabled, Make keeps scheduling scenario runs and discards the incomplete execution if it cannot be stored in your account.

You can read more about incomplete executions in the **dedicated article**.

Sequential processing

Enable this option to postpone running the scenario until the previous run finishes and until all scenario incomplete executions are resolved. Sequential processing makes sure that:

- The scenario runs finish in the same order as they were triggered.
- There is only one scenario execution running at the same time.

Sequential processing has the highest impact on scenarios that start with an instant trigger (a webhook) or on scenarios that have incomplete executions enabled.

Scenarios that start with an instant trigger run in parallel by default. For example:

If you have a scenario that starts with a webhook that runs for 5 minutes and you receive a webhook bundle at 12:00 and another one at 12:03, then from 12:03 to 12:05 there will be two instances of the scenario running at the same time in parallel.

In addition, if the scenario instance that started at 12:00 runs longer than usual, for example, until 12:12, the Make instance that started at 12:03 finishes sooner (at 12:08), even though it started later.

If you want to make sure that the scenario doesn't start before the previous run finishes, enable the sequential processing.

See **webhooks** for more information.

> The same applies to scenarios with the incomplete executions enabled. When there is an error and Make creates an incomplete execution, Make postpones the next scenario run until you resolve the incomplete execution or until it's resolved automatically with the **Break** error handler.

You can read more in the **scenario settings**.

Enable data loss

The enable data loss scenario setting influences the scenario incomplete executions storage. Enable this option to keep scheduling scenario runs regardless of not having enough space to store incomplete executions.

If you enable data loss, Make discards the data that doesn't fit into the size limits and continues running the scenario on schedule. Otherwise, Make disables the scenario scheduling instead.

Make sets the size limits based on your organization plan. Check the Make **pricing** or read more about the **scenario settings**.

Number of consecutive errors

This setting allows you to set how many times in a row the scenario can finish with an error and still keep being scheduled by Make for subsequent runs. When the scenario finishes with an error the specified number of times in a row, Make disables the scenario.

To access the setting of the number of consecutive errors, switch the advanced settings toggle in scenario settings. The default number of consecutive errors is 3.

The number of consecutive errors doesn't apply:

- When the scenario is triggered with an **instant trigger** (webhook). Make disables instantly triggered scenario immediately if an error happens.
- When an error happens with one of the following types:
  - `AccountValidationError`
  - `OperationsLimitExceededError`
  - `DataSizeLimitExceededError`
    Make disables the scenario scheduling immediately after the error happens.
- When you get a warning. If a scenario finishes with a warning, Make will keep scheduling subsequent scenario runs.

Auto-commit

Enable this option to commit changes right after they happen. For example, when a user triggers a scenario that updates their details.

If you disable this option, Make commits the changes after all modules finish successfully.

The setting affects only modules that support transactions. The modules supporting transactions are labeled with the "ACID" tag. They use a database app most of the time, like the Data Store or MySQL apps.

The modules that don't support transactions make changes immediately and don't provide the rollback functionality.

You can read more about the auto-commit option in the **scenario settings** article.

Commit trigger last

Enable this option to commit changes made by the first module in the scenario last. Otherwise, Make commits the changes in the same order as they happen.

The setting affects only modules that support transactions. The modules supporting transactions are labeled with the "ACID" tag. They use a database app most of the time, like the Data Store or MySQL apps.

The modules that don't support transactions make changes immediately and don't provide the rollback functionality.

You can read more about the commit trigger last option in the **scenario settings** article.

5.2.2. Fixing rate limit errors

App modules return the `RateLimitError` when you reach the API rate limit for the app. The rate limit is the number of requests you can send to an app over a period of time, typically over a second or a minute. The `RateLimitError` corresponds to the **HTTP error code 429**.

> Make follows the standard error codes and their definitions. Note that it is possible that the third party may not fully comply with the standard.

When you reach the app rate limit, the app will block further requests until the limiting time period passes. For example, **these are the API rate limits** for the **Google Sheets app**.

When a module returns the **RateLimitError** and you don't use any error handling, Make acts depending on the following attributes:

- scenario scheduling
- enabling of the incomplete executions

	Incomplete executions disabled	Incomplete executions enabled
Scheduled scenario	Make pauses the scheduling of the scenario for 20 minutes. Make doesn't rerun the scenario.	Make pauses the scheduling of the scenario for 20 minutes. Make retries the incomplete execution with the incomplete execution **backoff** .
Instant scenario	Make reruns the incomplete execution with the scenario **backoff**.	Make retries the incomplete execution with the incomplete execution **backoff** .

> Keep in mind that a third-party app tracks the rate limiting in your account in the app. The rate limiting of the app is shared between all your scenarios.
>
> If your scenario has a third-party app module that consumes a lot of operations or if you use an app module in your scenarios frequently, consider checking the rate-limiting of the third-party API or using the following error handling strategies preemptively to avoid frequent reruns of your scenario.

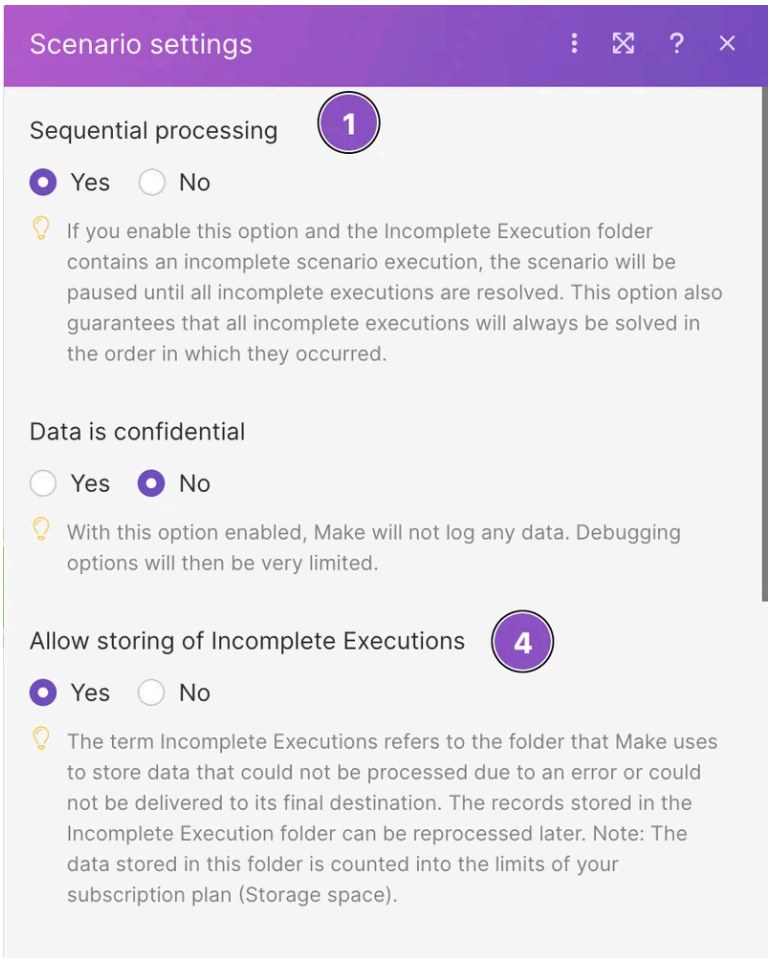
How to customize the handling of rate limit errors

You can use the following strategies to handle the `RateLimitError`:

- Add a delay with the **Sleep module** before the app module to delay the requests.
- Add the **Break error handler** to the module to handle errors and prevent turning off the scenario schedule.
- Combine both of the options together.

In the following steps, we combine a delay with the **Sleep** module and the **Break** error handler to handle the error if it occurs even after the delay.

1. If your scenario triggers with an **instant trigger** (for example, a **custom webhook module**), consider enabling **sequential processing** in scenario settings. With sequential processing, the trigger module processes incoming data one by one in the order they arrive.



Otherwise, skip this step.

> If a scenario has incomplete executions and sequential processing enabled, Make pauses the scenario until the incomplete executions are processed to keep the order of incoming data.

2 Add the **Sleep** module before the module that is causing the error. Set the delay to fit into the number of requests in the time limit. For example, if the app is limited to 10 requests per minute, set the delay to 6 seconds.

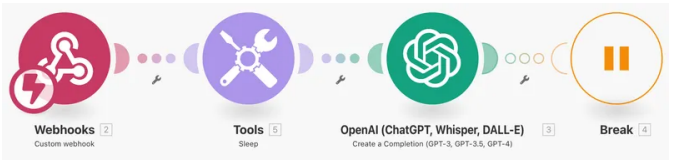
Look into the **advanced strategies** if you want to have finer control over the delays.

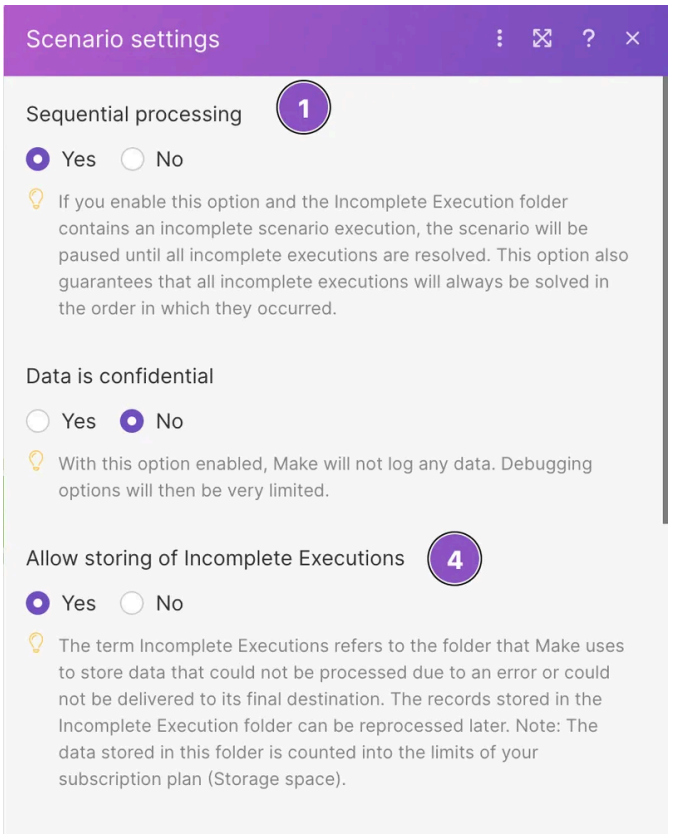
3 Add the **Break** error handler to the module that is causing the errors.

If the **Sleep** module delay doesn't prevent the error, the **Break** handler runs the module with the rest of the scenario flow again after another delay. Set the number of retries and the retry delay in the handler settings.

4. Enable **incomplete executions** in the scenario settings. Make will save bundles that caused the error.

For example, if you would use the **Webhook** trigger to send data with the **OpenAI** module, but the **OpenAI** module is causing errors, your scenario and scenario settings with error handling would look like this:





Advanced rate limit error handling strategies

When the strategies we mentioned above don't fit your use case or don't work for you, you can find more information in the following section. Here, we share advanced approaches to handling the rate limit errors in a scenario.

Fine-tuning the Sleep delay

This strategy can help you if you want to trigger the delay after the module processes a specific number of bundles.

If the modules you use in your scenario output a value you can use to number them, you can have a "delay route." You set the frequency of the delay with a filter and the duration of the delay with the **Sleep** module.

Options to enumerate bundles include:

- If you create bundles with an **Iterator** module, you can use the meta-variable **Bundle order position**.
- You can convert incoming bundles to an array with the **Array aggregator** module and then use the **Iterator**.
- You can use the **Increment function** module.

> Every bundle that goes through the **Increment function** module uses 1 operation. If you need to count a lot of bundles, consider a different option.

In the following example, we create a scenario with the **Array aggregator** and **Iterator** modules, because it's universal and uses just a few operations.

1. Set up your scenario. In our example, we will use the **Data Store** > **Search Records** and **HTTP** > **Make a request** modules. The **Make a request** module returns the `RateLimitError`.

2. Add the Array aggregator after the Data Store > Search Records module.

    1. Set the **Source module** to the **Search Records** module.
    2. Select any data from the data store to create the array.

3. Add the **Iterator** module after the **Array aggregator**. Iterate the array output from the aggregator.

    With the **Array aggregator** → **Iterator** flow, we create an array that we iterate to get the **Bundle order position** meta-variable in the **Iterator** output. In the next steps, we use this variable to delay the **Make a request** module to avoid the `RateLimitError`.

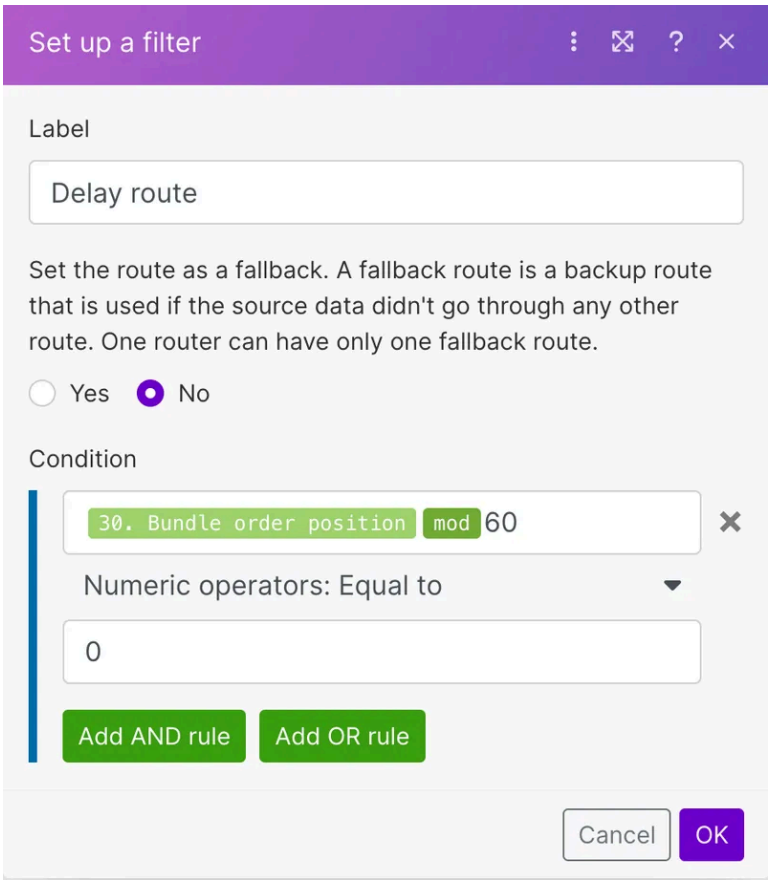
4. Add a **Router** after the **Iterator** module.

5 Add the **Sleep** module to the first route from the router. This route will be your "delay route". Set the time delay in the **Sleep** module settings.

6 Create a filter between the **Router** and **Sleep** modules.

1. Set the filter condition to use the **Bundle order position** variable output from the **Iterator** module.
2. Use the **mod** mathematic operator to set the frequency of the delay.
3. Set the filter operator to **Numeric: Equals to** and the other number to 0.

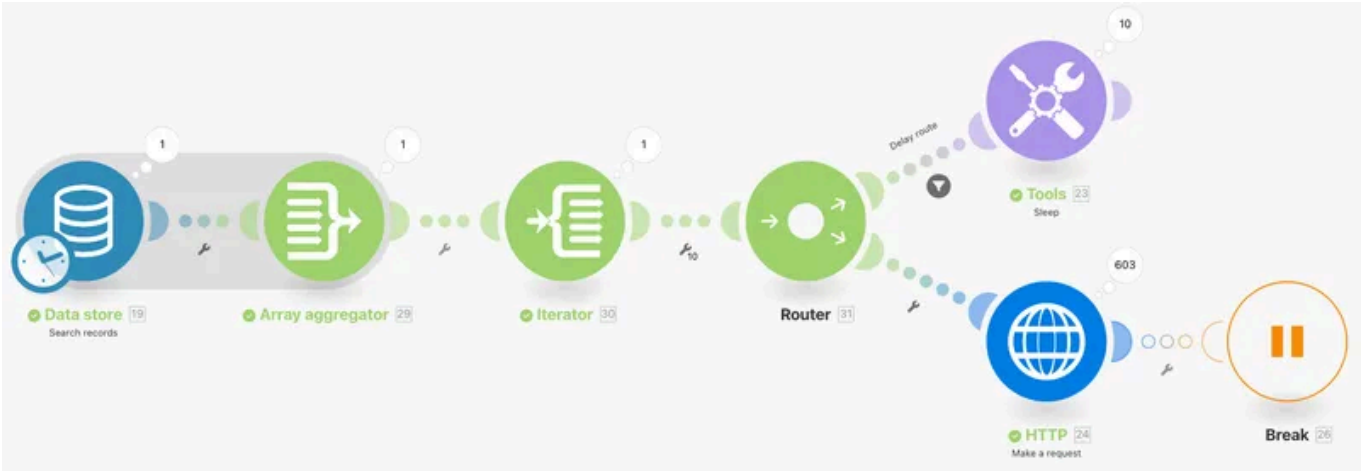
For example, the following filter triggers the delay with every 60th bundle:



After processing 60 bundles, the **Sleep** delay triggers. If the app API has a rate limit of 60 requests per second, adding a 1 second delay with the **Sleep** module prevents the `RateLimitError`.

7 Add the **HTTP** > **Make a request** module to the second route from the **Router**. You can add the **Break** error handler for the **Make a request** module too.

Our finished example scenario looks like this:



When you run the scenario, every 60th bundle triggers the **Sleep** module delay. The setup consumes only two extra operations to count the bundles.

5.2.3. Fixing connection errors

App modules output the `ConnectionError` when the app is unavailable. For example, the app might be offline for maintenance.

Make uses the **HTTP 503 and 504 status codes** to identify the `ConnectionError`.

> Make follows the standard error codes and their definitions. Note that it is possible that the third party may not fully comply with the standard.

When a module in your scenario outputs the `ConnectionError`, you should check the status page of the module app. Chances are that the status page will have the URL `https://status.domain`, for example **https://status.make.com**.

When Make recognizes the module output as the **ConnectionError** and you don't use any error handling, Make acts depending on the following attributes:

- scenario scheduling
- enabling of the incomplete executions

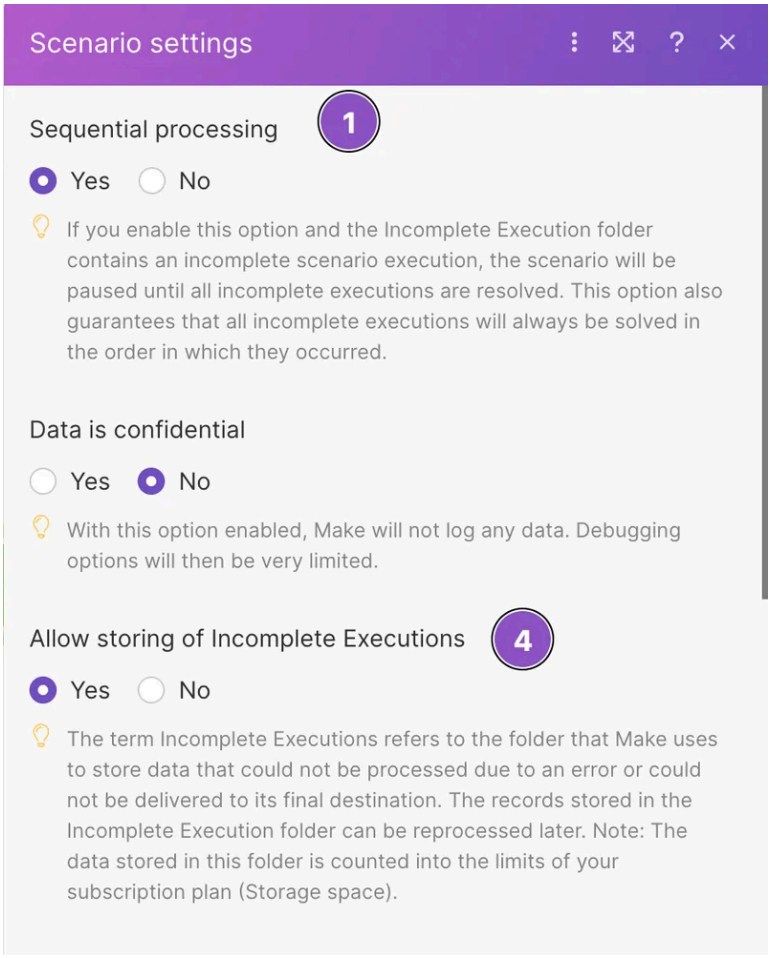
	Incomplete executions disabled	Incomplete executions enabled
Scheduled scenario	Make pauses the scheduling of the scenario for 20 minutes. Make doesn't rerun the scenario.	Make pauses the scheduling of the scenario for 20 minutes. Make retries the incomplete execution with the incomplete execution **backoff**.
Instant scenario	Make reruns the incomplete execution with the scenario **backoff**.	Make retries the incomplete execution with the incomplete execution **backoff**.

How to handle the ConnectionError

To handle the `ConnectionError`, you can use the **strategies for handling the RateLimitError**. The most efficient strategy is to use the **Break error handler** to rerun the

scenario after a delay:

1. If your scenario triggers with an **instant trigger** (for example, a custom webhook module), consider enabling sequential processing in scenario settings. With sequential processing, the trigger module processes incoming data one by one in the order they arrive.



   Otherwise, skip this step.

   > If a scenario has incomplete executions and sequential processing enabled, Make pauses the scenario until the incomplete executions are processed to keep the order of incoming data.

2. Add the **Break** error handler to the module that is causing the errors.

   Consider setting the delay and the number of attempts according to the importance and the schedule of your scenario.

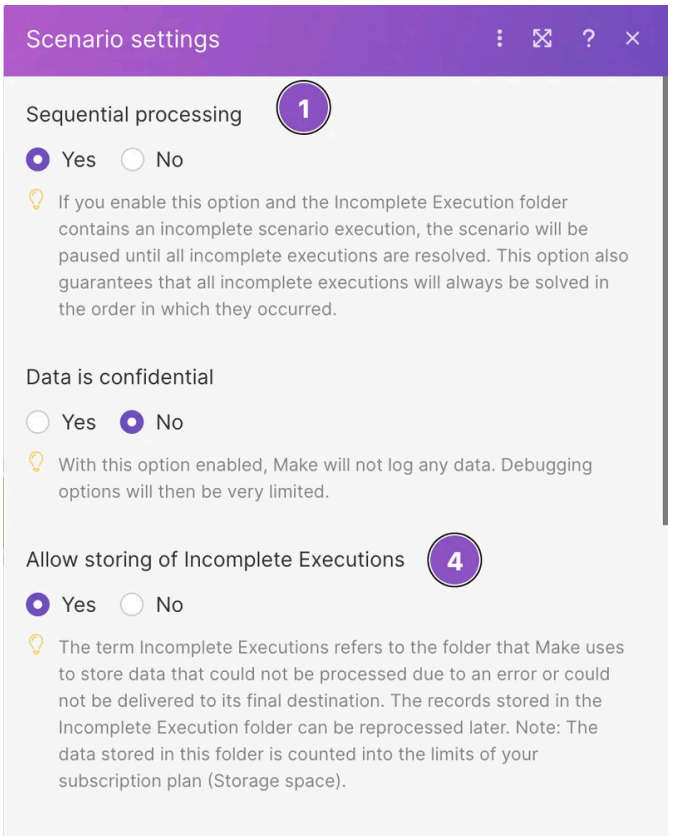
For example, if the app has occasional downtime for maintenance for a few hours with no availability, it might be best to set a lower number of attempts with longer time periods between them.

On the other hand, if the app is occasionally unavailable because it's overloaded, and the scenario is important for you, it might be best to use a short time period (a few minutes) with a higher number of attempts.

3. Enable incomplete executions in the scenario settings. Make will save bundles that caused the error.

For example, if you would use the **Webhook** trigger to ask questions to **ChatGPT**, but the **ChatGPT** app is sometimes overloaded with requests and sending back errors, your Make and Make settings with error handling could look like this:





Whenever the **Create a completion** module outputs the `ConnectionError` because the OpenAI servers are overloaded or unavailable, Make creates an incomplete execution with the **Create a completion** module.

After the delay set in the **Break** error handler, Make reruns the **Create a completion** module. If the rerun succeeds, Make will continue scheduling new scenario runs.

If the rerun fails, Make reruns the module again after the delay, up to the number of attempts set in the **Break** error handler settings.

5.2.4. Fixing missing data errors

App modules output the `BundleValidationError` when the input bundle is missing the required data. For example, if you map data from a search module that didn't return any results.

When a module in your scenario outputs the `BundleValidationError`, you should check the reason for the error. You should consider redesigning your scenario if you can avoid the error to make your automation more robust. You can use the **if** or **ifempty** functions or use **Filtering**.

Otherwise, you will have to handle the error to keep your scenario active.

In the following examples, we will consider two situations:

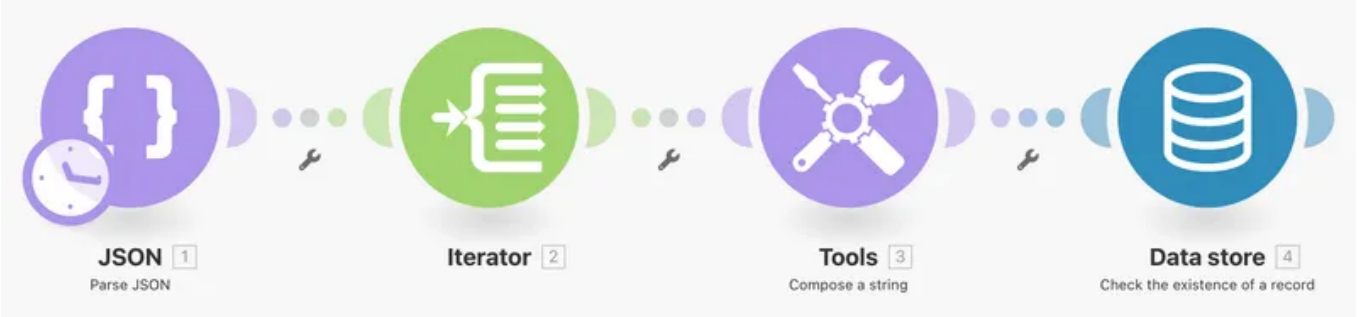
1. Ignore the missing data or use a placeholder.
   The missing data are not a problem. You just want to keep your scenario enabled.
2. Get notified about errors and store the scenario run.
   The missing data are not acceptable. If you get the `BundleValidationError`, you want to check what happened in detail.

To learn more about how to approach error handling in Make you can check the **error handling overview**.

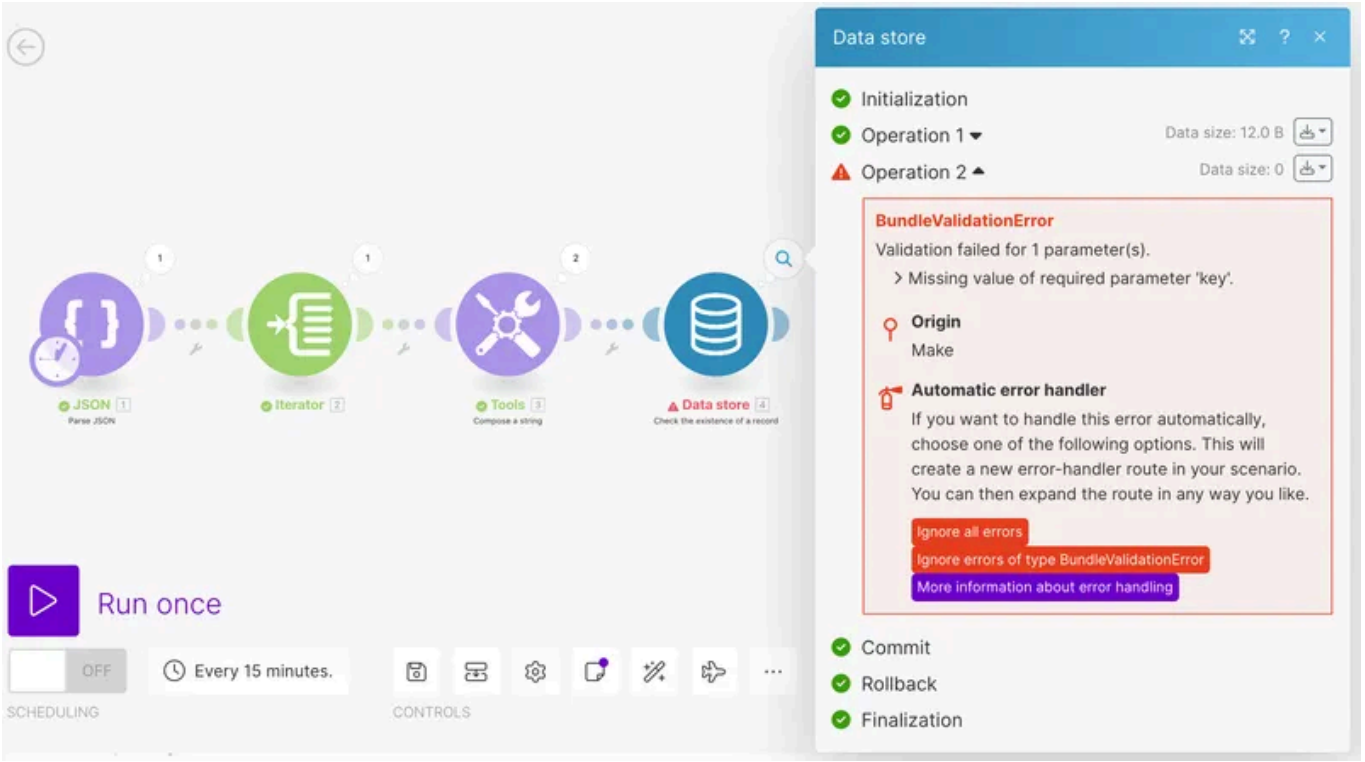
Ignore the missing data or use a placeholder

You might work with data that sometimes contain missing values. You don't need to fix them or investigate why they are missing. Your main goal is to avoid disabling your scenario.

We will use the following scenario for testing:



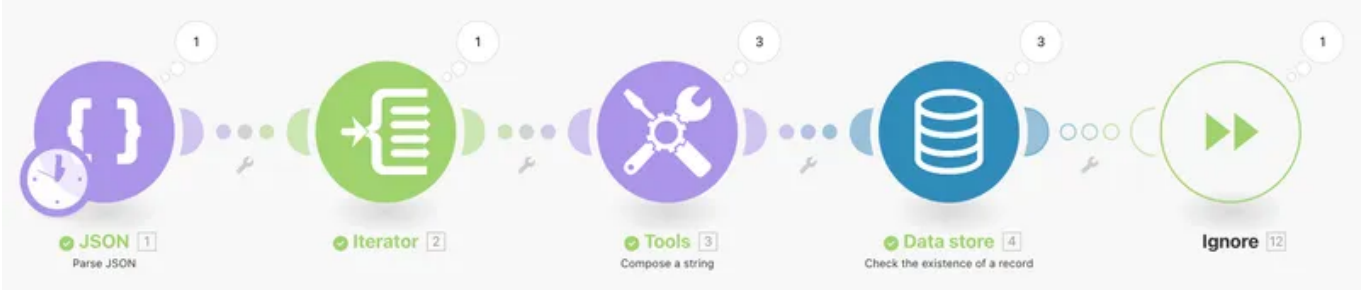
The scenario reads data, modifies them and uses them for search in the data store. One piece of data is empty, which causes the `BundleValidationError` in the search module.



To keep your scenario running even when the error happens, you can use the **Ignore** or **Resume** error handlers.

1. Right-click the module that is causing the error.
2. Select **Add an error handler**.
3. A pop-up appears. From the list of error handlers, select the **Ignore** error handler.

The final scenario should look like this:

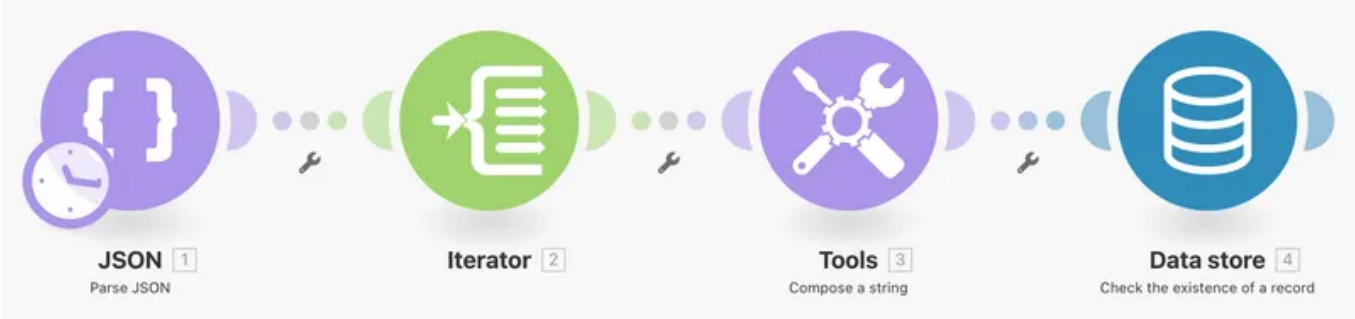


Get notified about errors and store the Make run

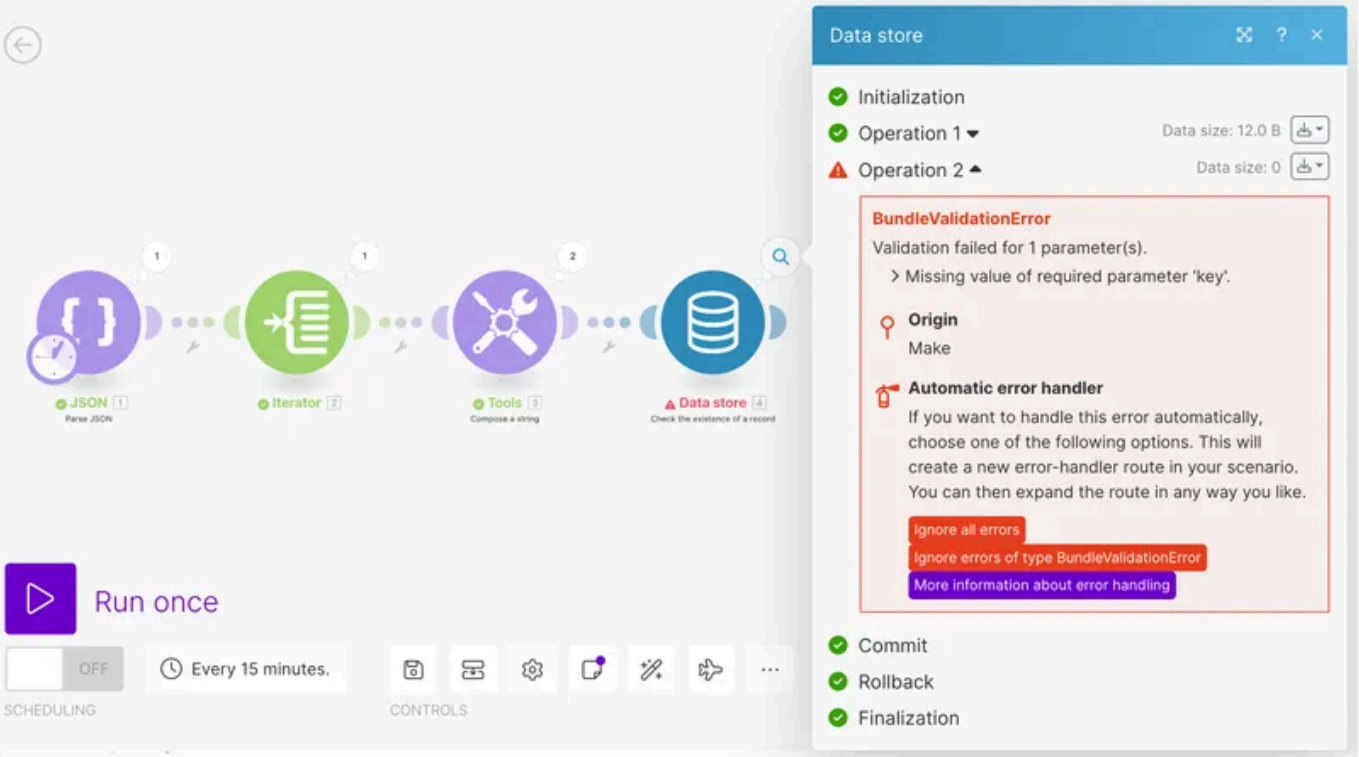
There might be scenarios you want to check closely when an error happens. You can do that with a custom error handling setup. If an error occurs in your scenario, you get an

email with the error description and a link to the scenario. In addition, you store the error bundle as incomplete execution to resolve it manually.

We will use the following scenario for testing:



The scenario reads data, modifies them and uses them for search in the data store. One piece of data is empty, which causes the `BundleValidationError` in the search module.

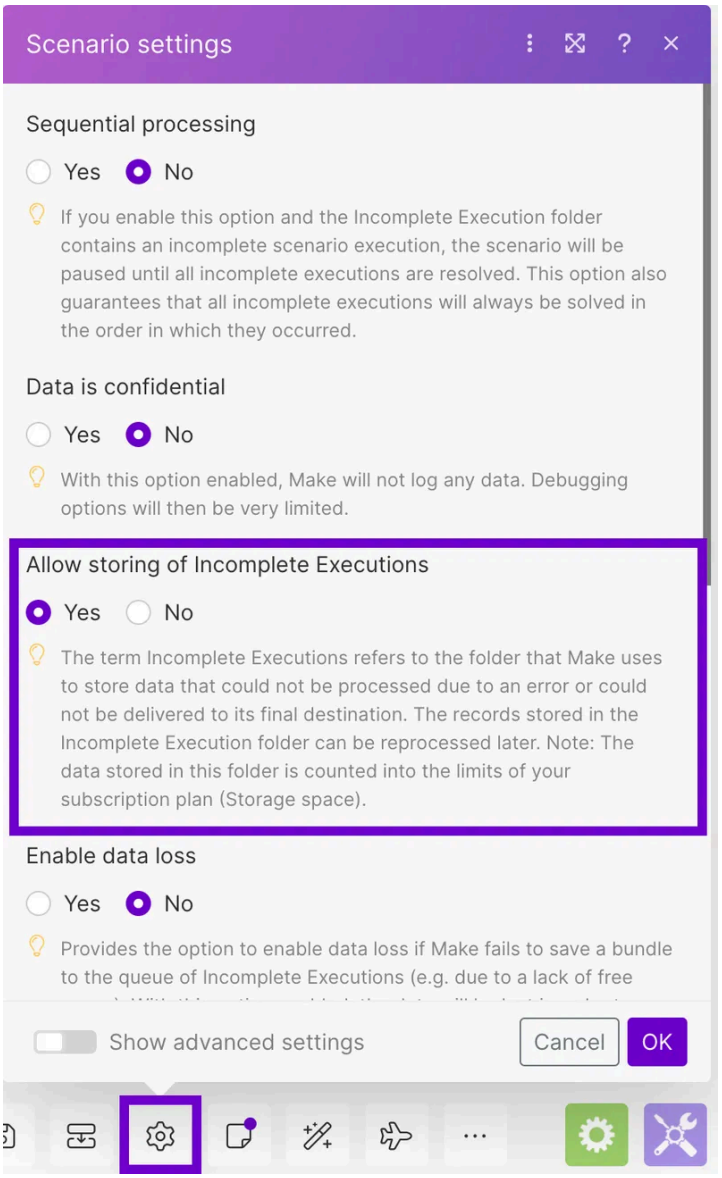


To store unfinished scenario runs and get a custom email notification when an error occurs, follow these steps:

1. Right-click the module that is causing the error.
2. Select **Add an error handler**.
3. A pop-up appears. From the list of error handlers, select the **Break** error handler.
4. A pop-up appears with the **Break** error handler settings. Set automatic completion to **No** to always resolve the error manually. Confirm the settings with the **OK** button. To

learn more about the **Break** error handler settings, check the **dedicated article**.

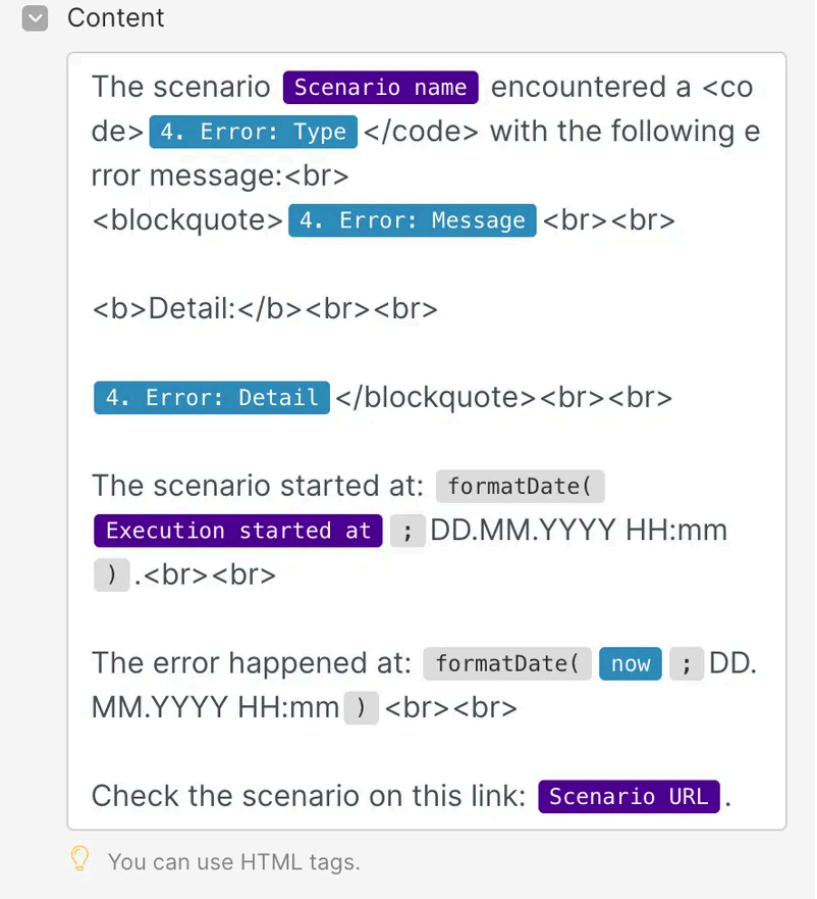
5. Click the scenario settings icon and enable storing of incomplete executions.



6. Right-click the connection between the erroring module and the **Break** error handler. Choose **Add a module**. Make inserts a gray module placeholder between the erroring module and the **Break** error handler.

7. Click the gray module placeholder and search for the **Gmail** app. Insert the **Send an email** module into the module placeholder.

8. Set up the **Send an email** module. In the **Content** field, add the notification message you want to receive when an error happens. Use the Make **system variables** to get the metadata. For example:



You set up error handling in your scenario to create an incomplete execution and notify you with an email when an error happens.

The example scenario with error handling looks like this:



5.3. Error handlers

→ Quick error handling reference

→ Break error handler

→ Commit error handler

→ Ignore error handler

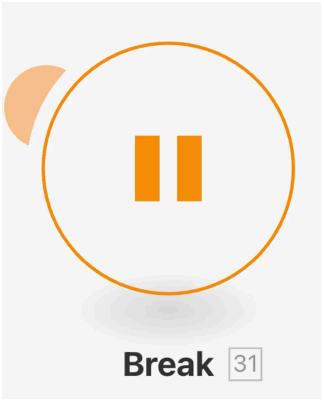
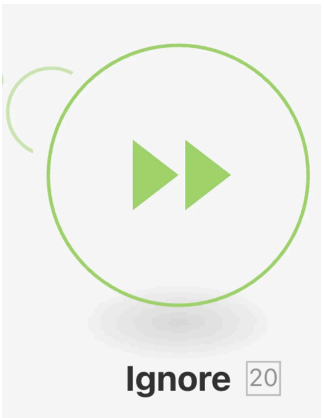
→ Resume error handler

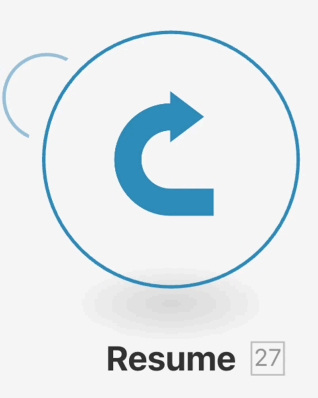
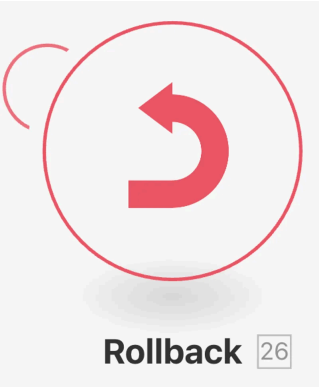
→ Rollback error handler

5.3.1. Quick error handling reference

The following table provides a quick reference to working with error handlers in Make.

If you want to learn more about individual error handlers, check the dedicated articles in this section of the Help Center. If you want to learn more about error handling, you can start with the **introduction**.

Break		Make stores the subsequent modules as an **incomplete execution**. Set the automatic completion to **Yes** to get similar functionality as a **Retry**. Otherwise, you have to resolve incomplete executions manually. Make processes the rest of the bundles in the scenario flow normally. **The scenario ends with the "warning" status.** Check the **Break error handler** for further information.
Commit		Make stops the scenario run and commits all changes. Make doesn't process the rest of the modules in the scenario flow. **The scenario ends with the "success" status.** Check the **Commit error handler** for further information.
Ignore		Make ignores the error. The bundle doesn't continue in the scenario flow. Make processes the rest of the bundles in the scenario flow normally. **The scenario ends with the "success" status.** Check the **Ignore error handler** for further information.
Resume		Specify a substitute mapping for when the module outputs an error. The substitute data continue through the rest of the scenario. Make processes the rest of the bundles in the scenario flow normally.

		**The scwnario ends with the "success" status.** Check the **Resume error handler** for further information.
Rollback		Make stops the scenario run and reverts changes in all modules that support transactions (**ACID modules**). Make doesn't process the rest of the modules in the scenario flow. Make stops scheduling the scenario after the **Rollback** activates for the **number of consecutive errors** in a row. **The scenario ends with the "error" status.**

Rollback is the default error handling if you don't set any error handling and when you keep **incomplete executions** disabled.

5.3.2. Break error handler

The **Break** error handler removes the erroring bundle from the scenario flow. Make stores the error message, mappings and the remaining scenario flow as an **incomplete executions**. Depending on the **Break** error handler settings, Make retries the incomplete scenario runs automatically or stores them until you resolve them yourself.

Automatic retry is most effective for temporary errors or errors where another attempt might be successful.

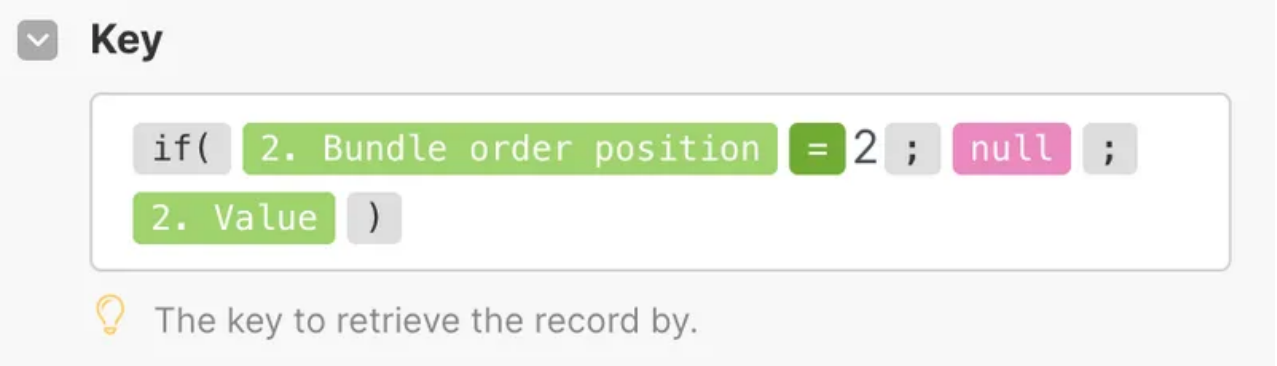
> Make automatically retries the most frequent temporary errors, the `ConnectionError` and `RateLimitError`, by default already. You don't need the **Break** error handler to automatically retry these types of errors.
>
> To automatically retry the `ConnectionError` and `RateLimitError`, you only need to enable incomplete executions in scenario settings.
>
> You can read more about automatic retry in the article about **incomplete executions**.

Make processes the rest of the bundles in the scenario flow.

To use the **Break** error handler in your scenario, you have to enable incomplete executions in scenario settings.

For example: This demo scenario contains five modules. The scenario is useful for tests and showing the effect of an error handler:

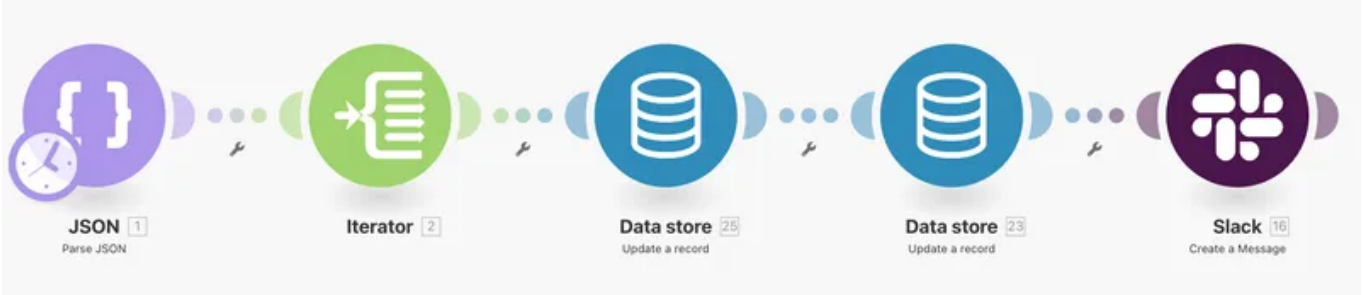
1. **JSON** - **Parse JSON** provides test data in the form of an array of three record IDs.
2. **Iterator** splits the array into individual bundles.
3. **Data store** - **Update a record**: Updates the data in the data store.
4. **Data store** - **Update a record**: This module updates the data again. This time the module works differently. In the module mapping, there is a mapping that intentionally creates an error:



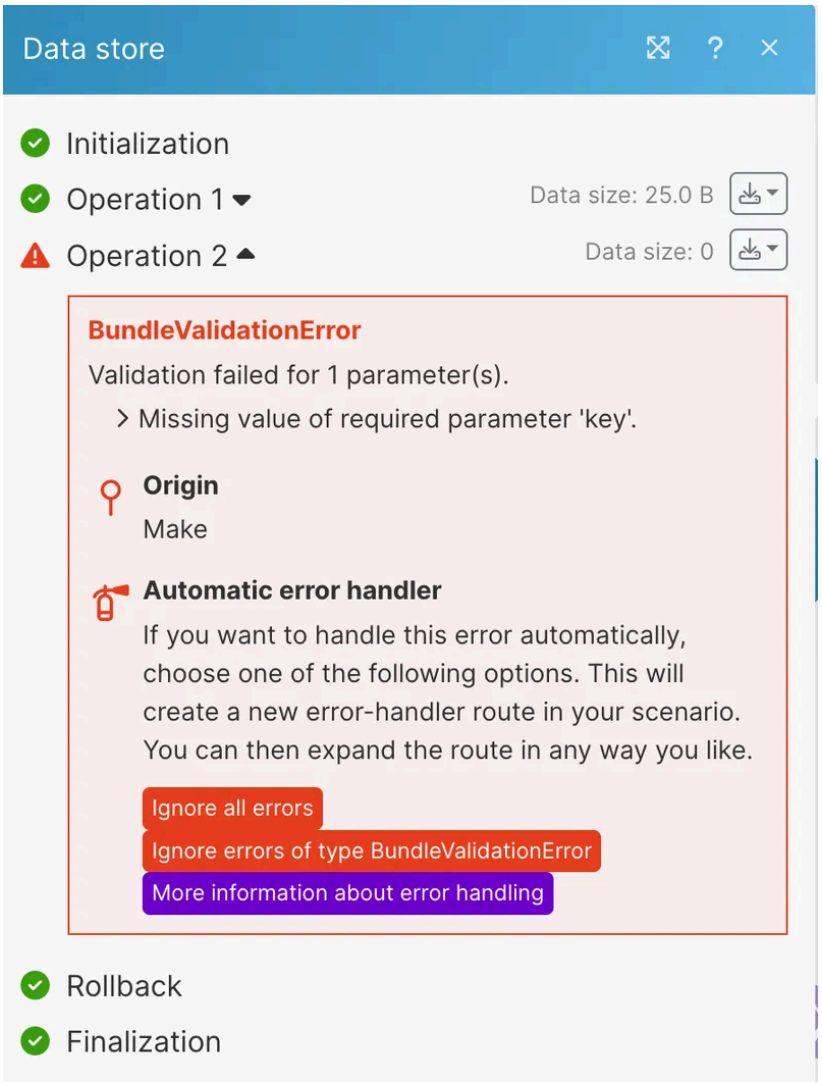
The mapping inserts a `null` value into the required **Key** field, which always creates the `BundleValidationError`.

5. **Slack** - **Send a message**: Sends a message to a private testing channel.

This is how the example scenario looks:



When we run the example scenario, we would get the `BundleValidationError`:



If we added the **Break** error handler to the **Update a record** module, the **Break** error handler would remove the bundle from the scenario flow. The bundle that caused the error doesn't continue through the rest of the scenario. Instead, Make creates an incomplete execution to store the error type, mappings, and the remaining scenario flow.

Make would then process the remaining bundles in the scenario flow.



For more information about error handling strategies check the **overview of error handling**.

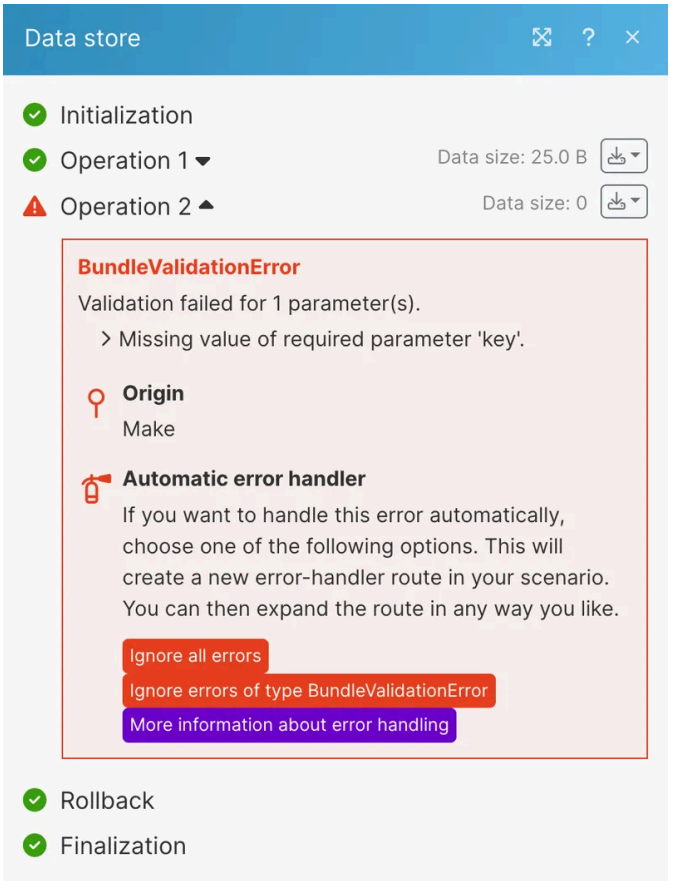
Store the scenario when an error happens

With the **Break** error handler, you can store the remaining scenario flow in the scenario incomplete executions when an error happens. Make also stores the error message, scenario mappings, and data.

You can finish the scenario run in the incomplete executions tab manually or Make can finish the run automatically.

For example, the following scenario outputs an error in the **Data Store** app module:





To store the scenario as an incomplete execution, follow the steps:

1. Right-click the module that is causing the error. In the menu, select **Add error handler**.
2. Select the **Break** error handler.

3. Optional: In the error handler settings, select if you want to automatically finish the incomplete executions. For automatic completions, you can set the number of attempts and the time delay between them.
4. Confirm the **Break** handler settings with the **OK** button.
5. Enable storing of incomplete executions in scenario settings.
6. Save your scenario.

You added the **Break** error handler to your scenario. When an error occurs in the **Data store** module, Make creates an incomplete execution. The scenario incomplete execution contains the error message, scenario mappings and data.



If you use the automatic scenario completion, Make attempts to finish the scenario automatically.

5.3.3. Commit error handler

The **Commit** error handler stops the scenario run and commits changes in your database apps. If your scenario is not using apps that support transactions, like **MySQL** or **Data store**, the **Commit** error handler just stops the scenario.

> Modules that support transactions are labeled with the "ACID" label.
>
> Before you use the **Rollback** or **Commit** error handlers, take a look at the **auto commit scenario setting** first.

The bundle that caused the error doesn't go through the rest of the scenario flow. Make doesn't process the rest of the bundles.

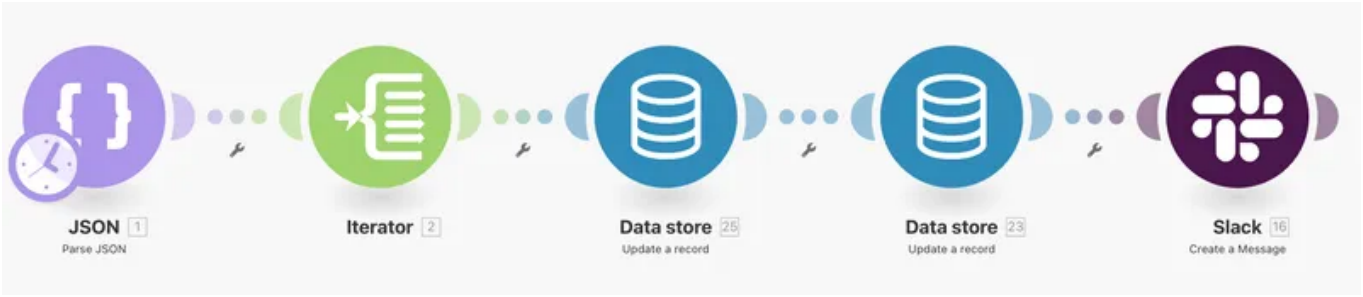
For example: This demo scenario contains five modules. The scenario is useful for tests and showing the effect of an error handler:

1. **JSON** - **Parse JSON** provides test data in the form of an array of three record IDs.
2. **Iterator** splits the array into individual bundles.
3. **Data store** - **Update a record**: Updates the data in the data store.
4. **Data store** - **Update a record**: This module updates the data again. This time the module works differently. In the module mapping, there is a mapping that intentionally creates an error:

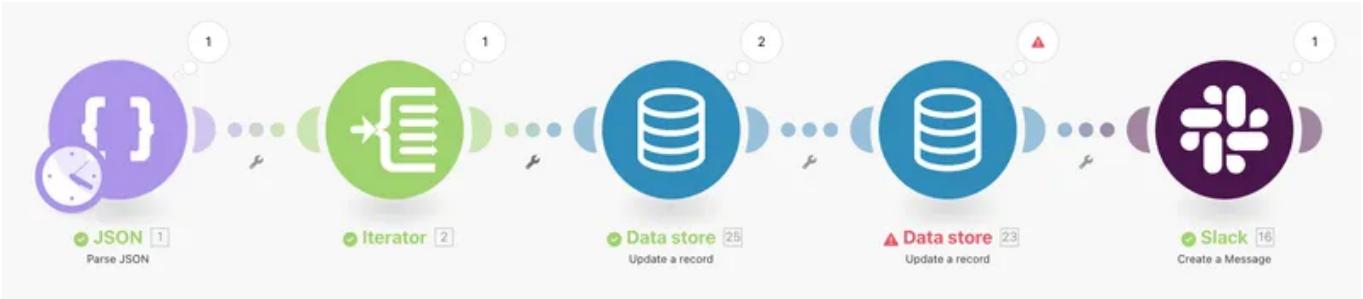
   

   The mapping inserts a `null` value into the required **Key** field, which always creates the `BundleValidationError`.
   Having two data store modules doing the same thing, but one of them failing, will make a good example for the **Commit** and **Rollback** error handlers.
5. **Slack** - **Send a message**: Sends a message to a private testing channel.

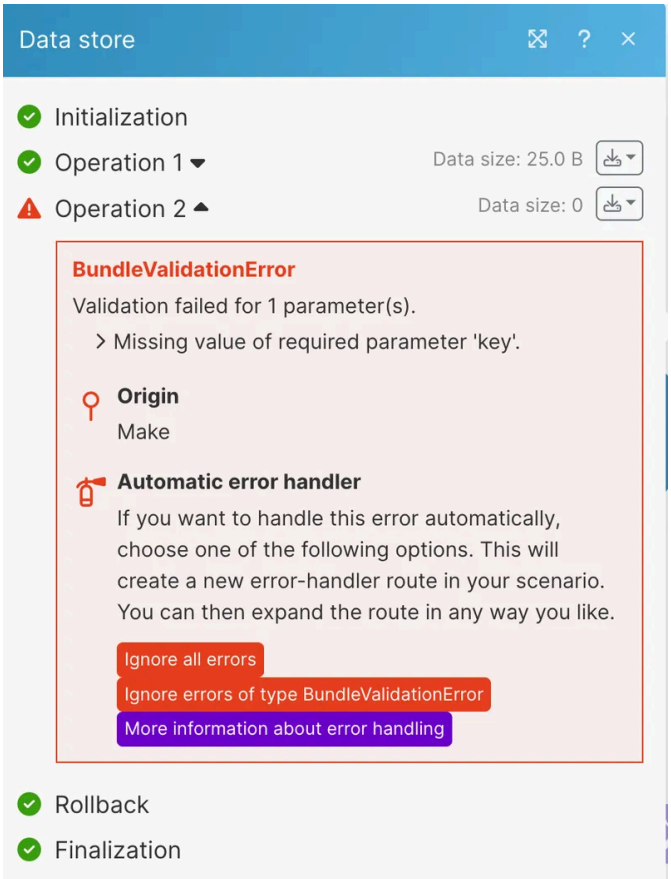
This is how the example scenario looks:



When we would run the example scenario, we would get the `BundleValidationError`:







If we added the **Commit** error handler to the **Update a record** module, the **Commit** error handler would stop processing the bundle in the scenario and save changes to your data in database apps. Make wouldn't process the remaining bundles.

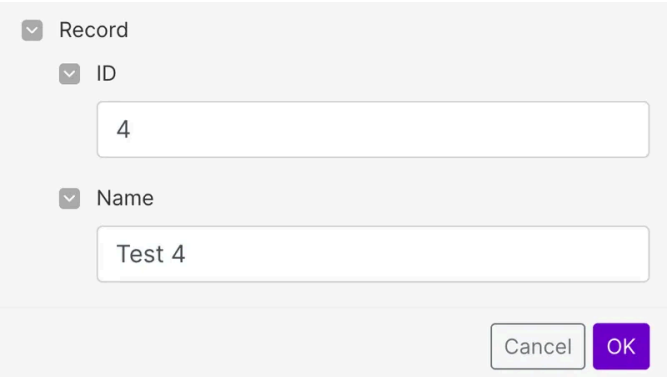


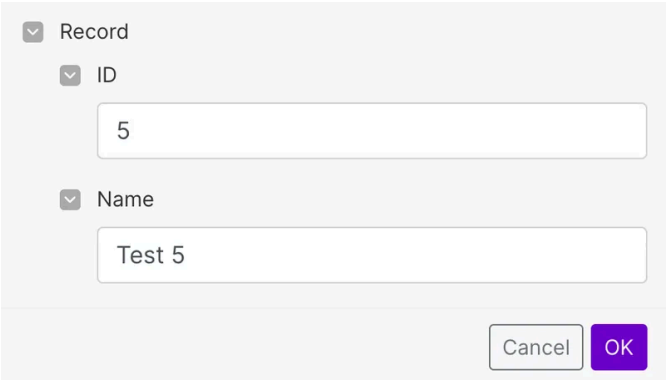
Let's check the data in the data store as well.

Before running the scenario, the data store contained the following data:



key	ID	Name
efdbe41d301a	1	Test 1
9f6528d6ab69	2	Test 2
7a956672551a	3	Test 3





The mappings for the **Update a record** modules. The first module updates the `ID` column to the number `4` and the `Name` column to the text `Test 4`.

The second module updates the `ID` column to the number `5` and the `Name` column to the text `Test 5`.

After running the scenario, Make would update the data in the data store:



key	ID	Name
efdbe41d301a	5	Test 5
9f6528d6ab69	4	Test 4
7a956672551a	3	Test 3

1. The first bundle of data gets through the scenario flow successfully and updates the first row of data in the data store both times. The first row contains the update from the second **Update a record** module: `ID = 5, Name = Test 5`.
2. The second bundle gets to the first **Update a record** module successfully, but causes an error in the second module.
   The **Commit** error handler saves the update in the first module, but prevents the update in the second module and stops the scenario. The second row contains the update from the first module only: `ID = 4, Name = Test 4`.
3. Make doesn't update the third row because the **Commit** error handler stopped the scenario run already. The data in the third row remain the same: `ID = 3, Name = Test 3`.

For more information about error handling strategies check the **overview of error handling**.

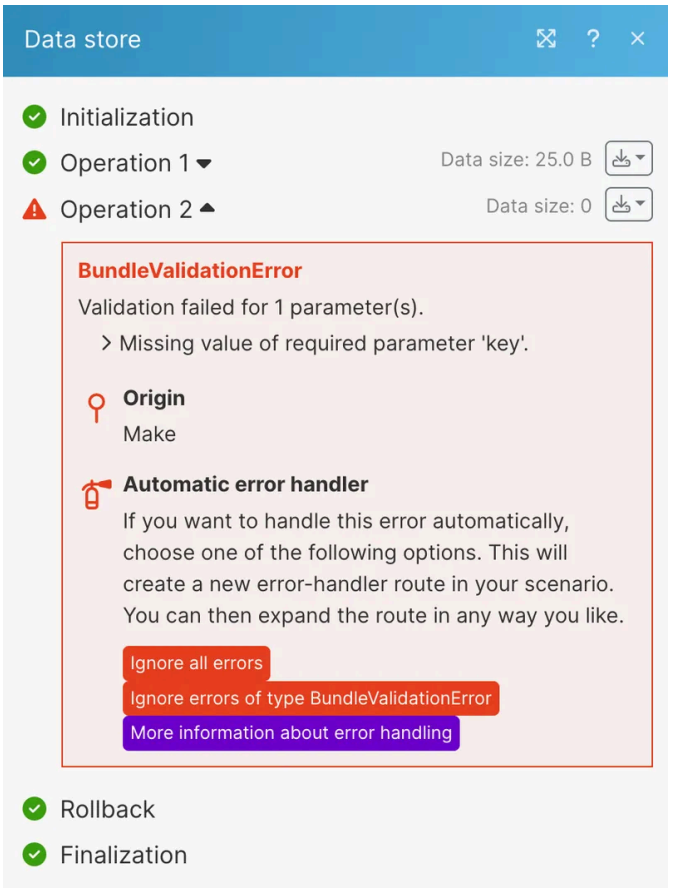
Stop the scenario when an error happens

With the **Commit** error handler, you can stop the scenario when an error happens. Make saves changes in the database apps in your scenario and doesn't process the rest of the bundles in the scenario flow.

> Modules that support transactions are labeled with the "ACID" label.
>
> Before you use the **Rollback** or **Commit** error handlers, take a look at the **auto commit scenario setting** first.

For example, the following scenario outputs an error in the **Data Store** app module:





To stop the scenario and save changes, follow the steps:

1. Right-click the module that is causing the error. In the menu, select **Add error handler**.
2. Select the **Commit** error handler.
3. Save your Make.

You added the **Commit** error handler to your scenario. When an error occurs in the **Data store** module, the scenario stops and Make saves changes in modules that support transactions.


1
1
2
2
1
JSON 1
Parse JSON
Iterator 2
Data store 25
Update a record
Data store 23
Update a record
Slack 16
Create a Message
Commit 27

5.3.4. Ignore error handler

The **Ignore** error handler ignores the error and removes the bundle from the scenario flow. The scenario run continues with the next bundle.

You can use the **Ignore** error handler when you know that there can be incorrect data in your scenario and they don't have an impact on your processes. The **Ignore** error handler prevents turning off the scenario when there's an error and marks the scenario run as a success even in case of errors.

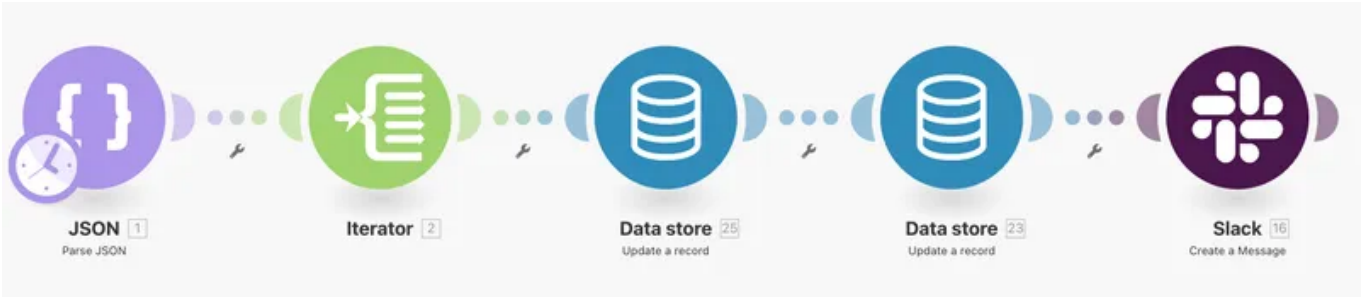
For example: This demo scenario contains five modules. The scenario is useful for tests and showing the effect of an error handler:

1. **JSON** - **Parse JSON** provides test data in the form of an array of three record IDs.
2. **Iterator** splits the array into individual bundles.
3. **Data store** - **Update a record**: Updates the data in the data store.
4. **Data store** - **Update a record**: This module updates the data again. This time the module works differently. In the module mapping, there is a mapping that intentionally creates an error:

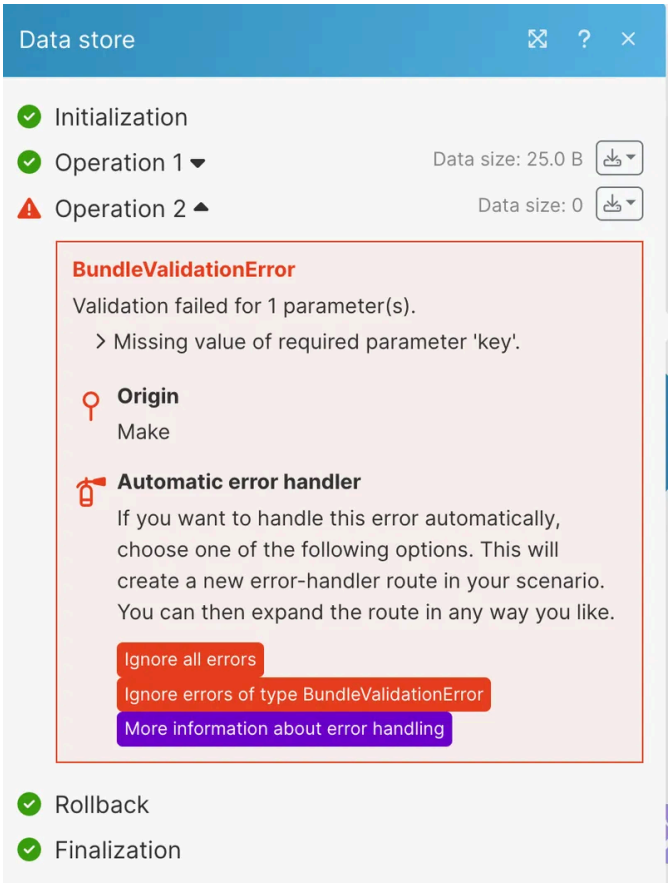
   The mapping inserts a `null` value into the required **Key** field, which always creates the `BundleValidationError`.
5. **Slack** - **Send a message**: Sends a message to a private testing channel.

This is how the example scenario looks:



When we would run the example scenario, we would get the `BundleValidationError`:





If we added the **Ignore** error handler to the **Update a record** module, the **Ignore** error handler would remove the bundle from the scenario flow. The bundle doesn't enter the fifth (**Send a message**) module. The third bundle runs through the scenario:



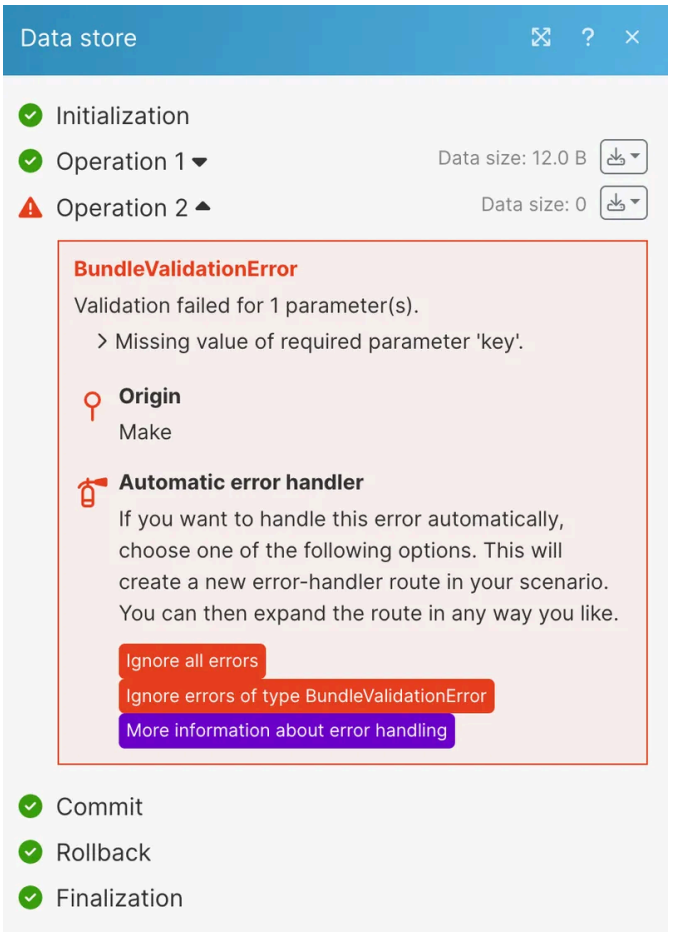
For more information about error handling strategies check the **overview of error handling**.

Keep your scenario running regardless of errors

With the **Ignore** error handler, you can remove the bundle that causes an error from the scenario flow and process the rest of the bundles in the scenario. In addition, Make will keep running your scenario on schedule instead of disabling scheduling because of an error.

For example, the following scenario outputs an error in the **Update a record** module:





To ignore the error and keep your scenario running regardless of errors, follow the steps:

1. Right-click the module that is causing the error. In the menu, select **Add error handler**.
2. Select the **Ignore** error handler.
3. Save your scenario.

Your scenario keeps running regardless of errors. When an error occurs in the **Data store** module, the **Ignore** error handler removes the bundle from the scenario flow.

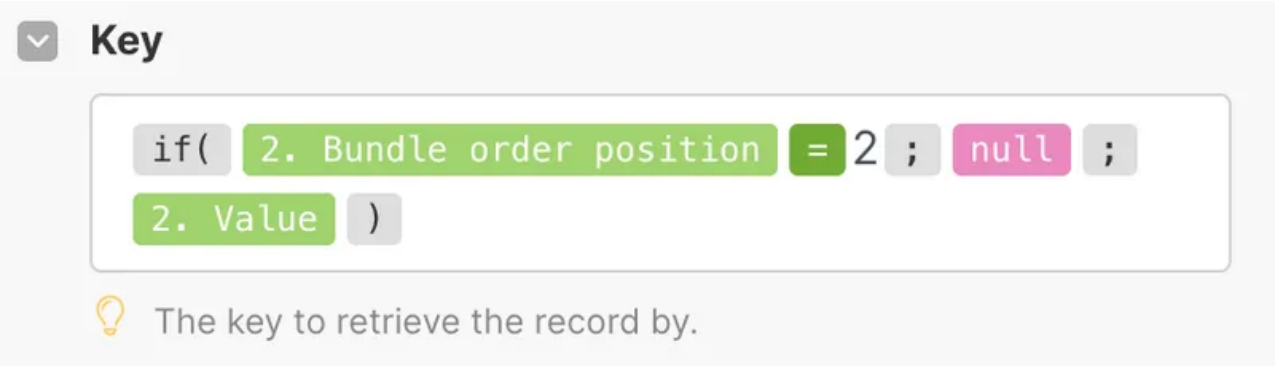

1
JSON 23
Parse JSON
1
Iterator 24
3
Data store 21
Update a record
3
Data store 22
Update a record
2
Slack 16
Create a Message
1
Ignore 20

5.3.5. Resume error handler

The **Resume** error handler replaces the module output with a substitute output when an error happens. You define the substitute output in the **Resume** error handler settings. The rest of the scenario uses the substitute output. Make processes the rest of the bundles in the scenario flow normally.

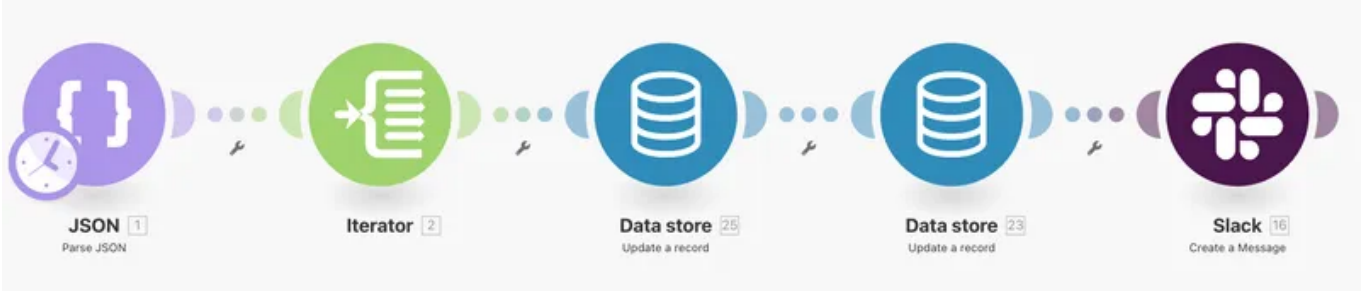
For example: This demo scenario contains five modules. The scenario is useful for tests and showing the effect of an error handler:

1. **JSON** - **Parse JSON** provides test data in the form of an array of three record IDs.
2. **Iterator** splits the array into individual bundles.
3. **Data store** - **Update a record**: Updates the data in the data store.
4. **Data store** - **Update a record**: This module updates the data again. This time the module works differently. In the module mapping, there is a mapping that intentionally creates an error:

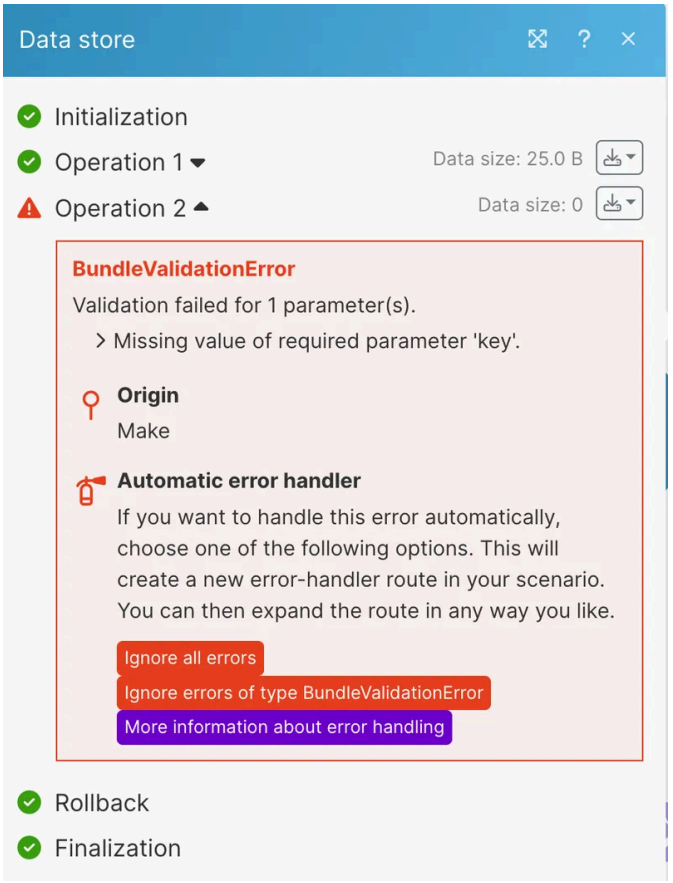
   The mapping inserts a `null` value into the required **Key** field, which always creates the `BundleValidationError`.
5. **Slack** - **Send a message**: Sends a message to a private testing channel.

This is how the example scenario looks:



When we would run the example scenario, we would get the `BundleValidationError`:





If we added the **Resume** error handler to the **Update a record** module, the **Resume** error handler would replace the bundle with the substitute mapping. When the module outputs an error, Make would use the substitute bundle instead. The third bundle runs through the scenario:



You can use the **Resume** error handler when you have a substitute mapping that fixes the bundle and allows the data bundle to continue in the scenario flow. You can also use the **Resume** error handler to add a placeholder that marks the data bundle to check later.

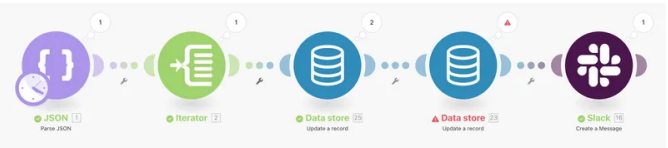
For more information about error handling strategies check the **overview of error handling**.

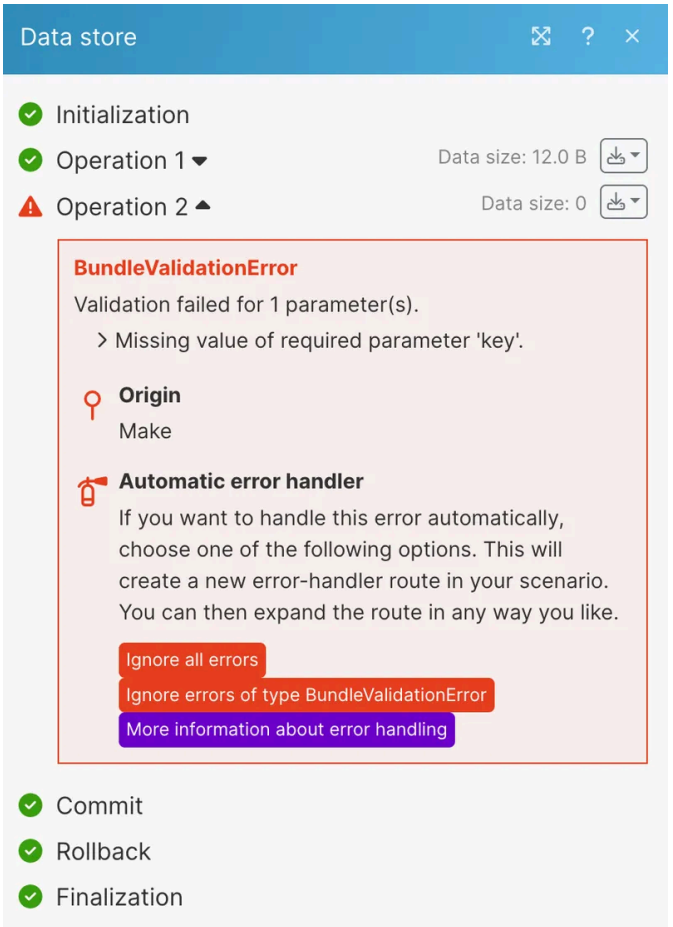
Change the bundle when it causes an error

With the **Resume** error handler, you can substitute the bundle that causes an error with your custom data. The custom data continue the rest of the scenario flow instead of the erroring bundle.

In the **Resume** error handler settings, you get the same fields as in the handled module settings. In the error handler fields, you can provide custom data that Make uses instead of the bundle that caused the error.

For example, the following scenario outputs an error in the **Update a record** module:





To change the erroring bundle for custom data and use them in the rest of the scenario, follow the steps:

1. Right-click the module that is causing the error. In the menu, select **Add error handler**.
2. Select the **Resume** error handler.
3. Fill in the **Resume** handler settings with your custom mapping.
4. Save your scenario.

You added the **Resume** error handler to your scenario. When an error occurs in the **Data store** module, the **Resume** error handler replaces the bundle with your custom mapping.



5.3.6. Rollback error handler

The **Rollback** error handler stops the scenario run and reverts changes made by modules that support transactions. They always use a database app, like **MySQL** or **Data store**. Make cannot undo actions made by modules that don't support transactions, like **Gmail > Send an email** or **Dropbox > Delete a file**.

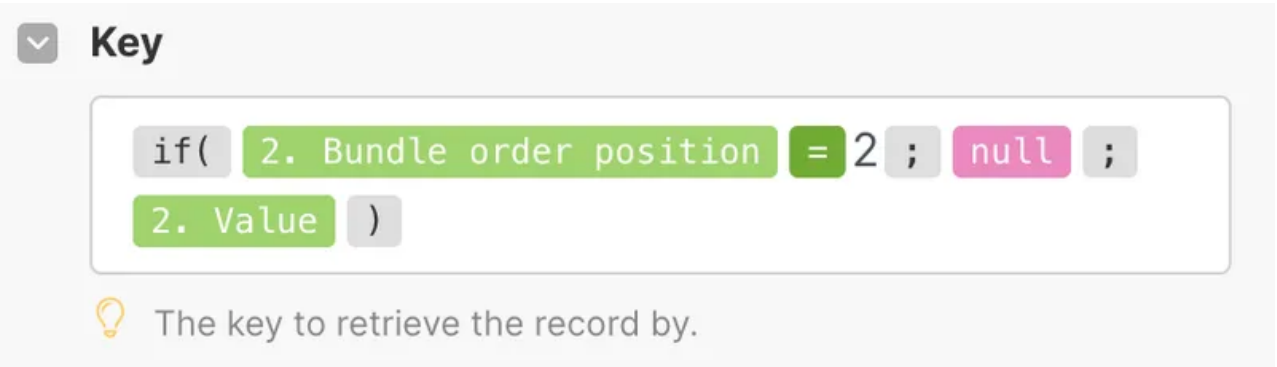
The bundle that caused the error doesn't continue in the scenario flow. In addition, Make doesn't process any remaining bundles.

Make marks the scenario run as an error in the scenario history. Make won't disable the scenario because of consecutive errors.

> Modules that support transactions are labeled with the "ACID" label.
>
> Before you use the **Rollback** or **Commit** error handlers, take a look at the **auto commit scenario setting** first.

For example: This demo scenario contains five modules. The scenario is useful for tests and showing the effect of an error handler:

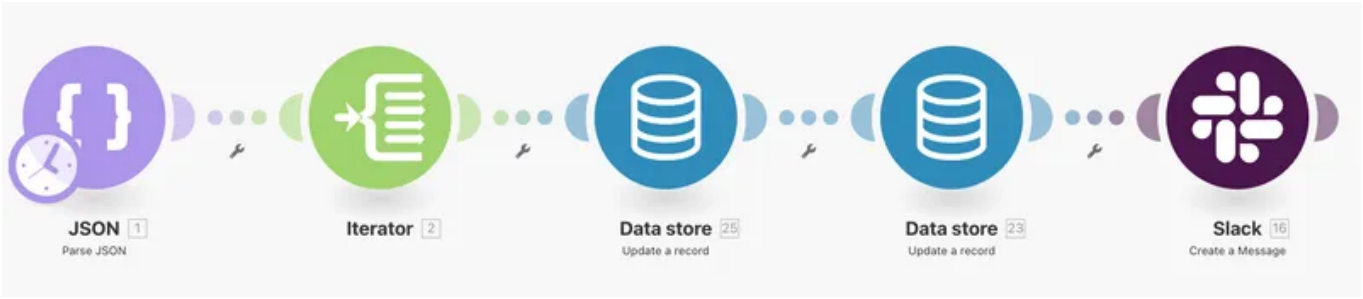
1. **JSON** - **Parse JSON** provides test data in the form of an array of three record IDs.
2. **Iterator** splits the array into individual bundles.
3. **Data store** - **Update a record**: Updates the data in the data store.
4. **Data store** - **Update a record**: This module updates the data again. This time the module works differently. In the module mapping, there is a mapping that intentionally creates an error:



The mapping inserts a `null` value into the required **Key** field, which always creates the `BundleValidationError`.
Having two data store modules doing the same thing, but one of them failing, will make a good example for the **Commit** and **Rollback** error handlers.

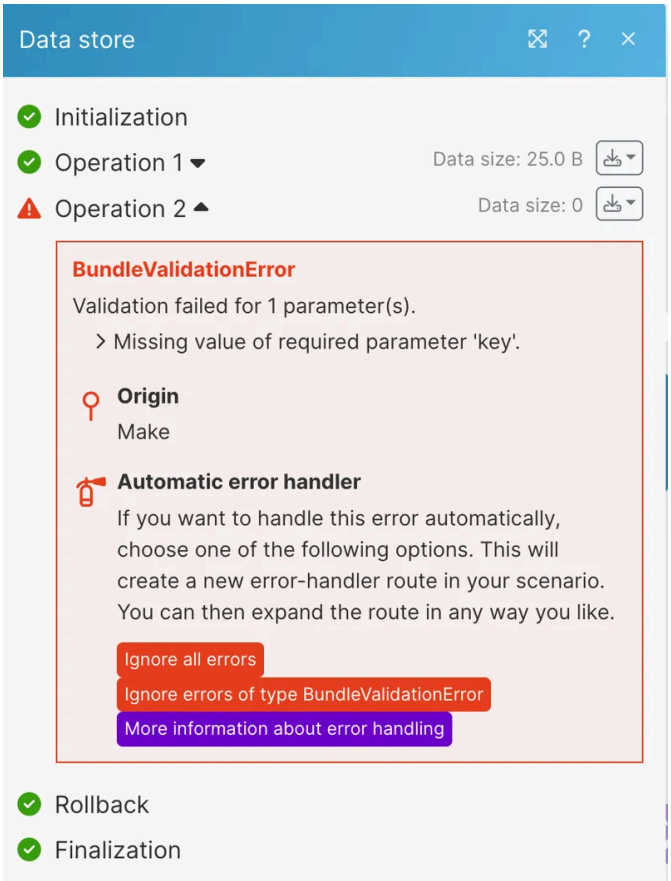
5. **Slack** - **Send a message**: Sends a message to a private testing channel.

This is how the example scenario looks:



When we would run the example scenario, we would get the `BundleValidationError`:





If we added the **Rollback** error handler to the **Update a record** module, the **Rollback** error handler would stop processing the bundle in the scenario. Make wouldn't process the remaining bundles.

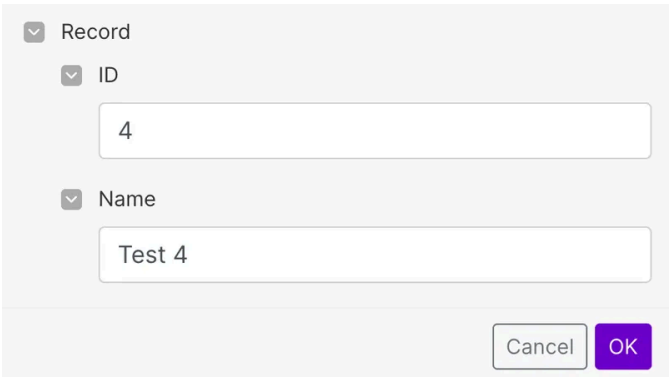


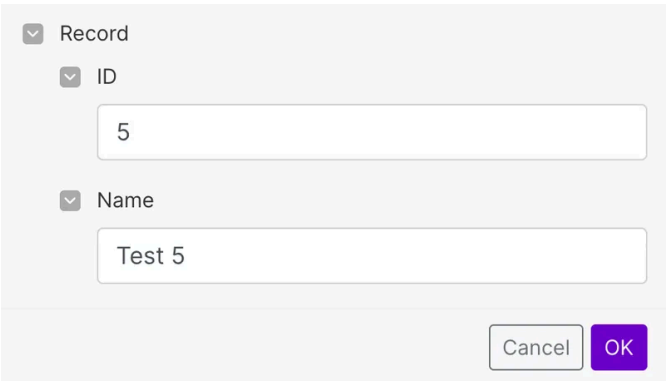
Let's check the data in the data store as well.

Before running the scenario, the data store contained the following data:



key	ID	Name
efdbe41d301a	1	Test 1
9f6528d6ab69	2	Test 2
7a956672551a	3	Test 3





The mappings for the **Update a record** modules. The first module updates the `ID` column to the number `4` and the `Name` column to the text `Test 4`.

The second module updates the `ID` column to the number `5` and the `Name` column to the text `Test 5`.

If you disable the **Auto-commit** option in the scenario settings, Make reverts the changes that happened when Make was processing the bundle in modules that support transactions.

key	ID	Name
efdbe41d301a	4	Test 4
9f6528d6ab69	2	Test 2
7a956672551a	3	Test 3

1. The first bundle of data gets through the scenario flow successfully and updates the first row of data in the data store both times. The first row contains the update from the second **Update a record** module: `ID = 5, Name = Test 5`.
2. The second bundle gets to the first **Update a record** module successfully, but causes an error in the second module. The **Rollback** error handler reverts the update from the second bundle and stops the scenario.
3. Make doesn't update the third row because the **Rollback** error handler stopped the scenario run already. The data in the third row remain the same: `ID = 3, Name = Test 3`.

If you keep the **Auto-commit** option enabled, Make reverts the changes made by the module that output the error if the module supports transactions.

key	ID	Name
efdbe41d301a	5	Test 5
9f6528d6ab69	4	Test 4
7a956672551a	3	Test 3

1. The first bundle of data gets through the scenario flow successfully and updates the first row of data in the data store both times. Make commits all changes and they cannot be rolled back later.
   The first row contains the update from the second **Update a record** module: `ID = 5, Name = Test 5`.
2. The second bundle gets to the first **Update a record** module successfully. Make commits all changes and they cannot be rolled back later. The second bundle causes an error in the second module.
   The **Rollback** error handler prevents the update in the second module and stops the scenario. The second row contains the update from the first module only: `ID = 4, Name = Test 4`.
3. Make doesn't update the third row because the **Rollback** error handler stopped the scenario run already. The data in the third row remain the same: `ID = 3, Name = Test 3`.

You can use the **Rollback** error handler to stop the scenario run and undo changes when the module outputs an error.

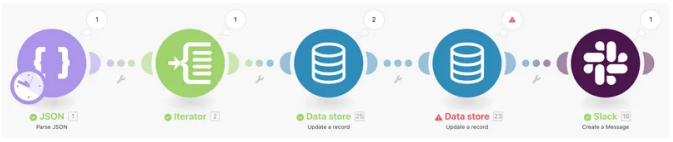
For more information about error handling strategies check the **overview of error handling**.

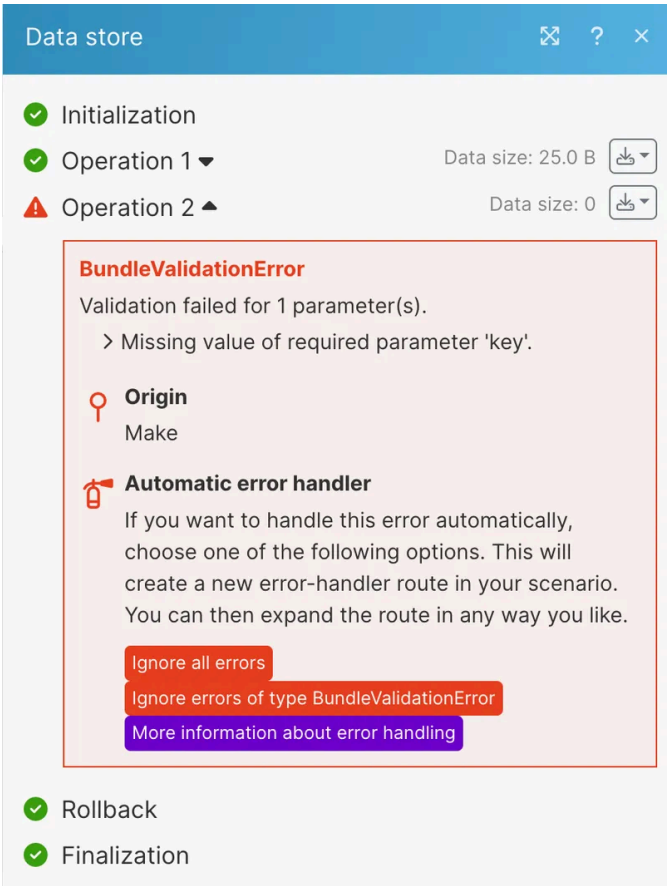
Undo changes to your data when an error happens

With the **Rollback** error handler, you can stop the scenario and undo changes when the module outputs an error. You can only undo changes in modules that support transactions.

> Modules that support transactions are labeled with the "ACID" label.
>
> Before you use the **Rollback** or **Commit** error handlers, take a look at the **auto commit scenario setting** first.

For example, the following scenario outputs an error in the **Data Store** app module:





To stop the scenario and undo changes where possible when an error happens, follow the steps:

1. Right-click the module that is causing the error. In the menu, select **Add error handler**.
2. Select the **Rollback** error handler.
3. Optional: Go to scenario settings and disable the **Auto-commit** option.
   When an error happens, the module that outputs the error reverts changes if the module supports transactions. If you disable the **Auto-commit** option, all modules in the scenario that support transactions undo changes.
4. Save your scenario.

You added the **Rollback** error handler to your scenario. When an error occurs in the **Data store** module, the scenario stops and Make reverts changes made by the erroring bundle in modules that support transactions.

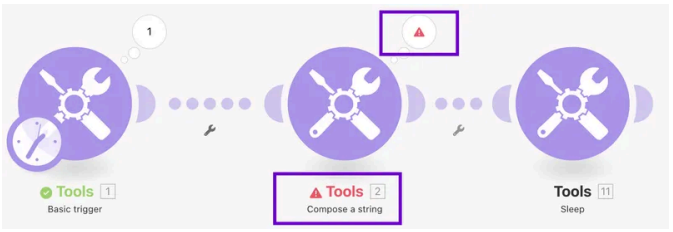


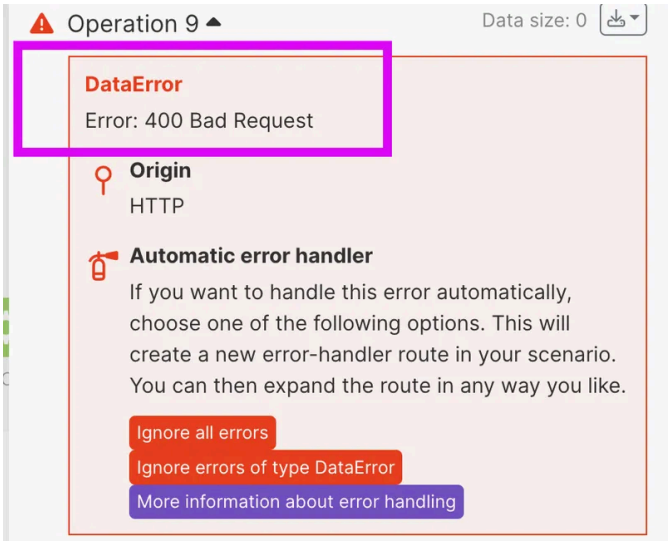
5.4. Types of errors

Within a scenario, you can have numerous modules and item mappings. Together, they create your scenario. Each of these elements can create an unexpected event -- an error. Make categorizes errors based on their origin and their cause.

For example, the `ConnectionError` happens when the app is unavailable because of maintenance or downtime on the third-party side and Make can't connect to the app servers.

When a module returns an error, Make highlights the module with the "Caution" sign. To check the error type, click the thought bubble above the module:





> Make follows the standard error codes and their definitions. Note that it is possible that the third party may not fully comply with the standard.

AccountValidationError

A module outputs the `AccountValidationError` when Make cannot authenticate you in the third-party app. For example, when you change your credentials in the app or your credentials expire and you don't update them in the connection, the app module will output the `AccountValidationError`.

The `AccountValidationError` appears also with the **HTTP status codes 401 and 403**.

If a module outputs the `AccountValidationError`, the scenario ends with an error. When your scenario finishes with an error for the **number of consecutive errors** in a row, Make disables the scheduling of your scenario.

> It is not possible to handle the `AccountValidationError` with an error handler, because the error happens during the **initialization of the scenario**.

To fix the `AccountValidationError`, review your credentials in the app and the connection in Make. If necessary, create a new connection for the app. If you are getting the `AccountValidationError` frequently, contact our **Support**.

BundleValidationError

A module outputs the `BundleValidationError` when the bundle entering the module doesn't pass validation. Validation means that before processing a bundle in a module, Make checks whether data types match in the module mappings and whether there are no missing values in the module required fields.

The `BundleValidationError` appears also with the **HTTP status codes 400 and 404**.

For example, you get the `BundleValidationError` when you map text to a module settings field that requires a date, or when you map an empty value to a required field in the module settings.

If a module outputs the `BundleValidationError` with no error handling, the scenario ends with an error. When your scenario finishes with an error for the **number of consecutive errors** in a row, Make disables the scheduling of your scenario.

The best way to handle this error is to review your mapping in the module settings.

For tips on how to handle missing data in a scenario check out the article about **how to fix missing data errors**.

ConnectionError

A module outputs the `ConnectionError` when the third-party app is unavailable. For example, the third-party service might be offline because of maintenance. This error uses

the **HTTP 502, 503 and 504 status codes**.

The default handling of the `ConnectionError` depends on the following attributes:

- scenario scheduling
- enabling of the incomplete executions

	Incomplete executions disabled	Incomplete executions enabled
Scheduled scenario	Make pauses the scheduling of the scenario for 20 minutes. Make doesn't rerun the scenario.	Make pauses the scheduling of the scenario for 20 minutes. Make retries the incomplete execution with the incomplete execution **backoff** .
Instant scenario	Make reruns the incomplete execution with the scenario **backoff**.	Make retries the incomplete execution with the incomplete execution **backoff** .

To fix the `ConnectionError`, check the **article about fixing ConnectionErrors**.

DataError

A module outputs the `DataError` when data sent by the module doesn't pass validation on the third-party side. For example, when you try to create a tweet with the **Twitter** > **Create a Tweet** module that has more than 280 characters, the **Create a Tweet** module outputs the `DataError` because tweets have a maximum length of 280 characters.

Another situation when you get the `DataError` is when you map an incorrect data type to a function. For example, when you map data with the `text` data type to the **parseDate** function.

If a module outputs the `DataError` with no error handling, the scenario ends with an error. When your scenario finishes with an error for the **number of consecutive errors** in a row, Make disables the scheduling of your scenario.

To fix the `DataError`, review your mapping and identify the reason why the error happens. If you cannot fix the error with different mapping, you can use the **Resume** and **Ignore** error handlers.

DataSizeLimitExceededError

A module outputs the `DataSizeLimitExceededError` when you run out of data transfer quota. Your data transfer limit is calculated from the operations limit.

If a module outputs the `DataSizeLimitExceededError` with no error handling, the scenario ends with an error. Because the `DataSizeLimitExceededError` is a fatal error, Make immediately disables the scenario scheduling, regardless of the **number of consecutive errors**.

To fix the `DataSizeLimitExceededError`, consider **purchasing extra operations** or upgrading your organization's subscription plan.

DuplicateDataError

A module outputs the `DuplicateDataError` when you send the same data to a module that doesn't allow duplicates. For example, when you try to create a new user in an app and the user's e-mail address has to be unique, but the e-mail address is used already. The module outputs the `DuplicateDataError`, because the e-mail address is registered with another user already.

If a module outputs the `DuplicateDataError` with no error handling, the scenario ends with an error. When your scenario finishes with an error for the **number of consecutive errors** in a row, Make disables the scheduling of your scenario.

If you get the `DuplicateDataError` in your scenario, you should review your scenario design. For example, if you are using a database, you could first check if the database record exists with a search module. Or with the email example, you could just ignore the error with the **Ignore error handler**.

IncompleteDataError

A module outputs the `IncompleteDataError` when the module can get only part of the data from the third-party app.

For example, when you are uploading new photos to Google Photos and you have a scenario that downloads the photos at the same time. Make tries to download the photo

that you are currently uploading. The photo file won't be complete and the Google Photos module will output the `IncompleteDataError`.

If a module outputs the `IncompleteDataError` with no error handling, the scenario ends with an error. Make pauses the scenario for 20 minutes and then runs the scenario again until the scenario succeeds or reaches the **number of consecutive errors**.

If you want to handle the `IncompleteDataError`, you can use the **Break error handler**.

InconsistencyError

A module outputs the `InconsistencyError` when an error happens during the scenario **rollback**. This error can occur when you make changes to a data store with two scenarios that run at the same time. If one scenario encounters an error and attempts to undo the changes in the rollback phase, but the other scenario already finished making changes, the changes cannot be safely undone and you get the `InconsistencyError` from the data store module in the first scenario.

For example, imagine two people making changes in the same part of a file. One of them saves changes before the other. What happens with the changes from the other person?

If a module outputs the `InconsistencyError` with no error handling, the scenario ends with an error. Because the `InconsistencyError` is a fatal error, Make immediately disables the scenario scheduling, regardless of the **number of consecutive errors**.

To fix the `InconsistencyError`, check your data and fix the data if necessary. If you are getting the `InconsistencyError` frequently, check the scenarios that use the database.

MaxFileSizeExceededError

A module outputs the `MaxFileSizeExceededError` when you try to process a file that exceeds the maximum file size. The maximum file size differs based on your organization's subscription. You can check the maximum file sizes in the Make **pricing**.

For example, if you use the **Google Drive** > **Move a file** module in an organization with the Core plan to move a file larger than 100 MB, you would get the `MaxFileSizeExceededError`.

If a module outputs the `MaxFileSizeExceededError` with no error handling, the scenario ends with an error. When your scenario finishes with an error for the **number of consecutive errors** in a row, Make disables the scheduling of your scenario.

To fix the `MaxFileSizeError`, you have to either make the file smaller (compress or split the file) or upgrade your organization's subscription plan.

ModuleTimeoutError

A module outputs the `ModuleTimeoutError` when the module is requesting or processing data for more than 40 seconds. For example, this error can happen when you are retrieving a large amount of data from the app or when the app has reduced availability.

The default handling of the `ModuleTimeoutError` depends on the following attributes:

- scenario scheduling
- enabling of the incomplete executions

	Incomplete executions disabled	Incomplete executions enabled
Scheduled scenario	Make pauses the scheduling of the scenario for 20 minutes. Make doesn't rerun the scenario.	Make pauses the scheduling of the scenario for 20 minutes. Make retries the incomplete execution with the incomplete execution **backoff** .
Instant scenario	Make reruns the incomplete execution with the scenario **backoff**.	Make retries the incomplete execution with the incomplete execution **backoff** .

To fix the `ModuleTimeoutError` with search modules, set the **Limit** in module settings to a lower number. To deal with reduced availability, check the **article about fixing a ConnectionError**.

OperationsLimitExceededError

A module outputs the `OperationsLimitExceededError` when you run out of operations. Your operations limit is set with your organization's subscription. You can check your

operations limit in the Make **pricing**.

If a module outputs the `OperationsLimitExceededError`, the scenario ends with an error. Because the `OperationsLimitExceededError` is a fatal error, Make immediately disables the scenario scheduling, regardless of the **number of consecutive errors**.

To fix the `OperationsLimitExceededError`, consider **purchasing extra operations** or upgrading your organization's subscription plan.

RateLimitError

A module outputs the `RateLimitError` when you make too many requests over a time period to the app API. This error uses the **HTTP 429 status code** and follows the **rate-limiting** rules of the third party.

For example, the **Make app** modules output the `RateLimitError` when you reach the number of API calls per minute based on your **subscription**.

The default handling of the `RateLimitError` depends on the following attributes:

- scenario scheduling
- enabling of the incomplete executions

	Incomplete executions disabled	Incomplete executions enabled
Scheduled scenario	Make pauses the scheduling of the scenario for 20 minutes. Make doesn't rerun the scenario.	Make pauses the scheduling of the scenario for 20 minutes. Make retries the incomplete execution with the incomplete execution **backoff** .
Instant scenario	Make reruns the incomplete execution with the scenario **backoff**.	Make retries the incomplete execution with the incomplete execution **backoff** .

To fix the `RateLimitError`, check the **article about fixing the rate-limit errors**.

RuntimeError

A module outputs the `RuntimeError` when the error reported by the third-party app doesn't meet the criteria for any other error type. For example, you get the `RuntimeError` when you use up all your tokens with the **OpenAI** > **Create a Completion** module.

If a module outputs the `RuntimeError` with no error handling, the scenario ends with an error. When your scenario finishes with an error for the **number of consecutive errors** in a row, Make disables the scheduling of your scenario.

There's no general rule to fixing the `RuntimeError`. Check the error message in the scenario history or try to reproduce the error with the **Make DevTool**.

The **overview of error handling** can help you create your error handling strategy.

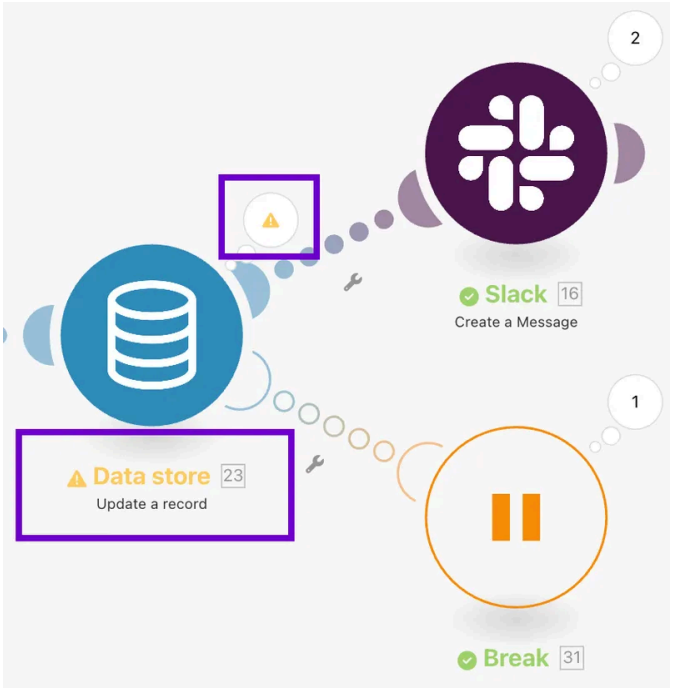
5.5. Types of warnings

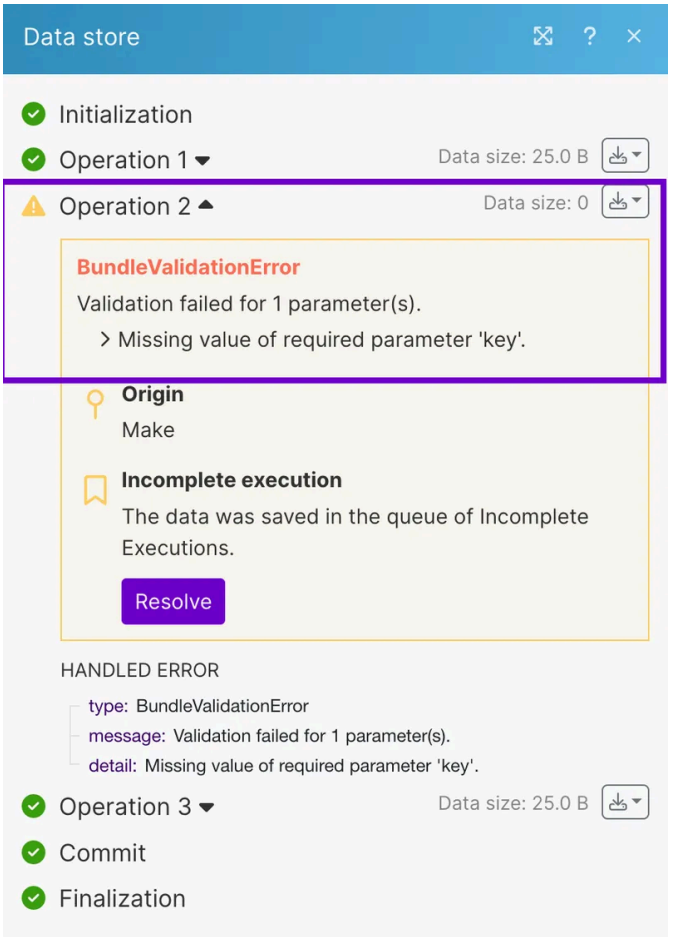
Within a scenario, you can have numerous modules and item mappings. Together, they create your scenario. Each of these elements can create an event that might need your attention -- a warning.

A warning can also be the result of handling an error with the **Break error handler**. With the warning Make informs you that the **Break** error handler created an **incomplete execution** of your scenario.

The main difference between a warning and an error is, that a warning doesn't disable the scheduling of your scenario and warnings don't count into the **number of consecutive errors**.

When a module returns a warning, Make highlights the module with the "Warning" sign. To check the error type, click the thought bubble above the module.





When you use error handling, you will see a warning when the error handler activates. In the example, the **Break error handler** handled the **BundleValidationError**. The error handler activation turned the original error into a warning.

ExecutionInterruptedError

A module outputs the `ExecutionInterruptedError` when the scenario runs for more than 40 minutes (5 minutes for the Free subscription). The module that is currently processing data outputs the warning.

When a module outputs the `ExecutionInterruptedError` the scenario ends with a warning. The scenario doesn't process the remaining bundles. Make will keep scheduling further scenario runs.

To fix the `ExecutionInterruptedError`, consider strategies to optimize your scenario:

- Split your scenario. You can use the combination of the following modules:

  For example, if you are processing a lot of bundles in a scenario with 20 modules and the scenario ends with the `ExecutionInterruptedError`, you can split the scenario into two scenarios with 10 modules each.
- Set the search modules' **Limit** to a lower number to reduce the amount of data processed by the scenario.
- Check the API documentation of the apps you use in the scenario. If the app's API has a suitable endpoint that supports batch requests, you can use the **JSON aggregator** and the **HTTP** app to send your data in batches. In addition, you save on operation usage.

OutOfSpaceError

Data store modules output the `OutOfSpaceError` when Make cannot store any more data in the data store. You also get the `OutOfSpaceError` when you fill your incomplete execution storage.

If a module outputs the `OutOfSpaceError` with no error handling, the scenario ends with a warning. The scenario doesn't process the remaining bundles. Make will keep scheduling further scenario runs.

To fix the `OutOfSpaceError` in a scenario, you can use a backup data store with the **Resume error handler**. You should also check the data in the data store or your incomplete execution storage. You can also review the Make **pricing**.

5.6. Exponential backoff

Make uses **exponential backoff** to automatically schedule scenario reruns with increasing time delays. The scenario reruns trigger when a module in the scenario outputs:

- `RateLimitError`
- `ConnectionError`
- `ModuleTimeoutError`

The time delays between the scenario reruns depend on whether **incomplete executions** are enabled or not:

Rerun number	Incomplete executions enabled	Incomplete executions disabled
1	1 minute	1 minute
2	10 minutes	2 minutes
3	10 minutes	5 minutes
4	30 minutes	10 minutes
5	30 minutes	1 hour
6	30 minutes	3 hours
7	3 hours	12 hours
8	3 hours	24 hours

This means that when a module in your scenario outputs the `ConnectionError` because the third-party service is unavailable, Make schedules the rerunning of the scenario one minute after the error occurs.

If a module during the rerun outputs the `ConnectionError` again, Make schedules another rerun after 2 minutes and so on. If the 8th attempt fails, Make disables scheduling of the scenario.

> Until Make disables scheduling the scenario, the scenario follows the run schedule in addition to the exponential backoff reruns (though there are **exceptions**).
>
> If you want Make to wait with scheduling the next scenario runs until the reruns finish, enable **sequential processing** in scenario settings.

If you enable the option to store incomplete executions in the scenario settings, Make stores the scenario run in incomplete executions.

Make then schedules scenario reruns from the incomplete execution data. If all rerun attempts fail, Make keeps the scenario incomplete execution for you to resolve manually.

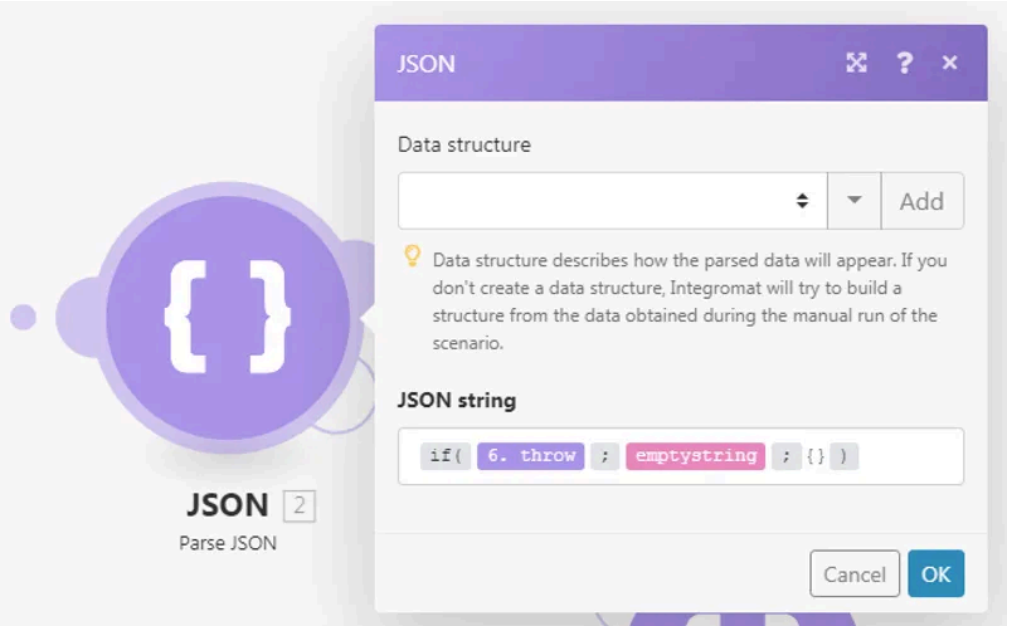
5.7. Throw

Make does not offer the **Throw** **error handling directive**. This feature implementation is being analyzed and evaluated.

> This article describes alternatives and workarounds to mimic the **Throw** error handling directive.

Alternate solution

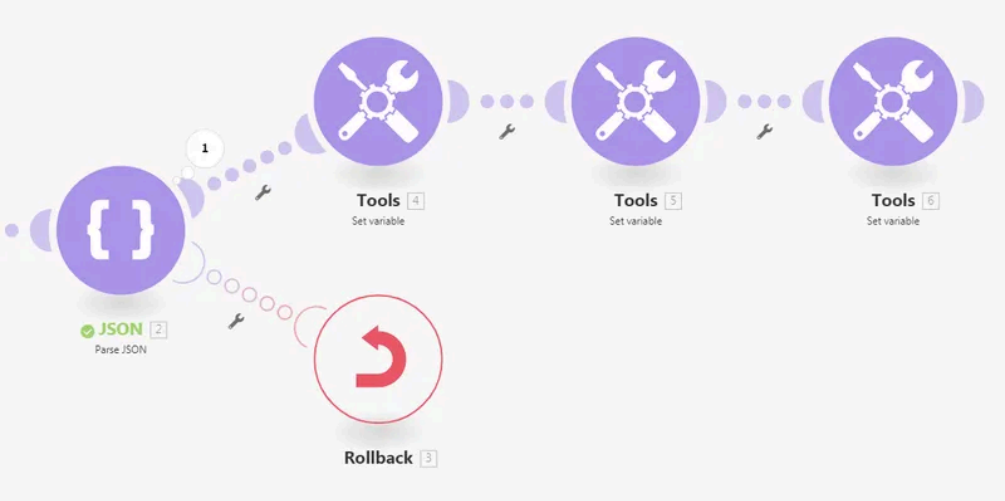
To conditionally throw an error you may configure a module to make it optionally purposely fail during its operation. One possibility is to employ **JSON > Parse JSON** module configured to optionally throw an error (`BundleValidationError` in this case):



You can then attach one of the **error handling directives** to the error handling route to:

- force the scenario execution to stop and perform the rollback phase: Rollback
- force the scenario execution to stop and perform the commit phase: Commit
- stop the processing of a route: Ignore
- stop the processing of a route and storing it in the queue of **incomplete executions** : Break

The following example shows the use of the Rollback directive:

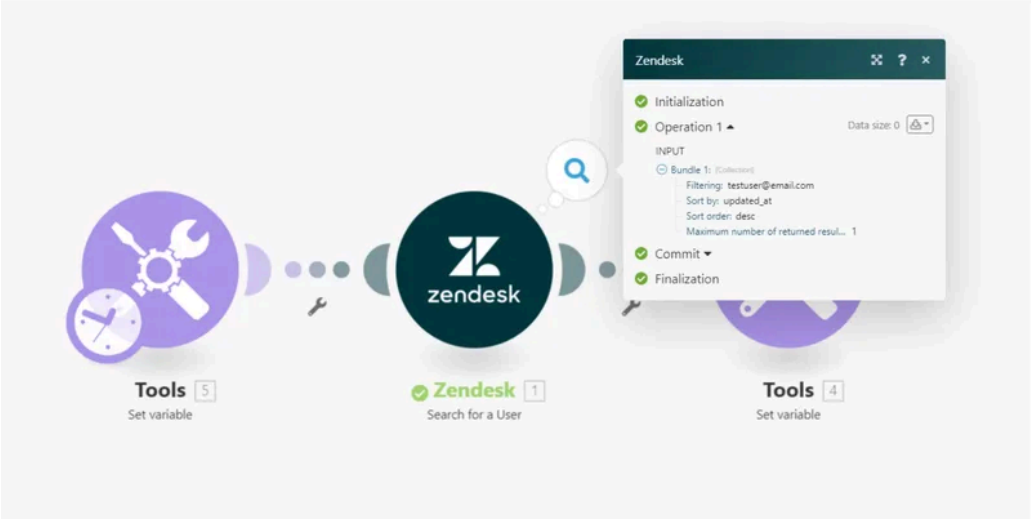


Workaround - Using HTTP Module

Usecase - Retry after some time if there is no record found using Break Directive. Usually, this is crucial when your record doesn't update instantly and you would like to process it later on in automation. Using this Break directive could be handy with the lesser complexity of the setup.

Current Barrier - Make does not offer a module that would enable you to easily conditionally generate (throw) errors.

To give you a better understanding here is the current setup without modification: This scenario search in Zendesk if there is no use it won't throw an error forcefully to search it again you would need to implement a complex procedure by saving the record.

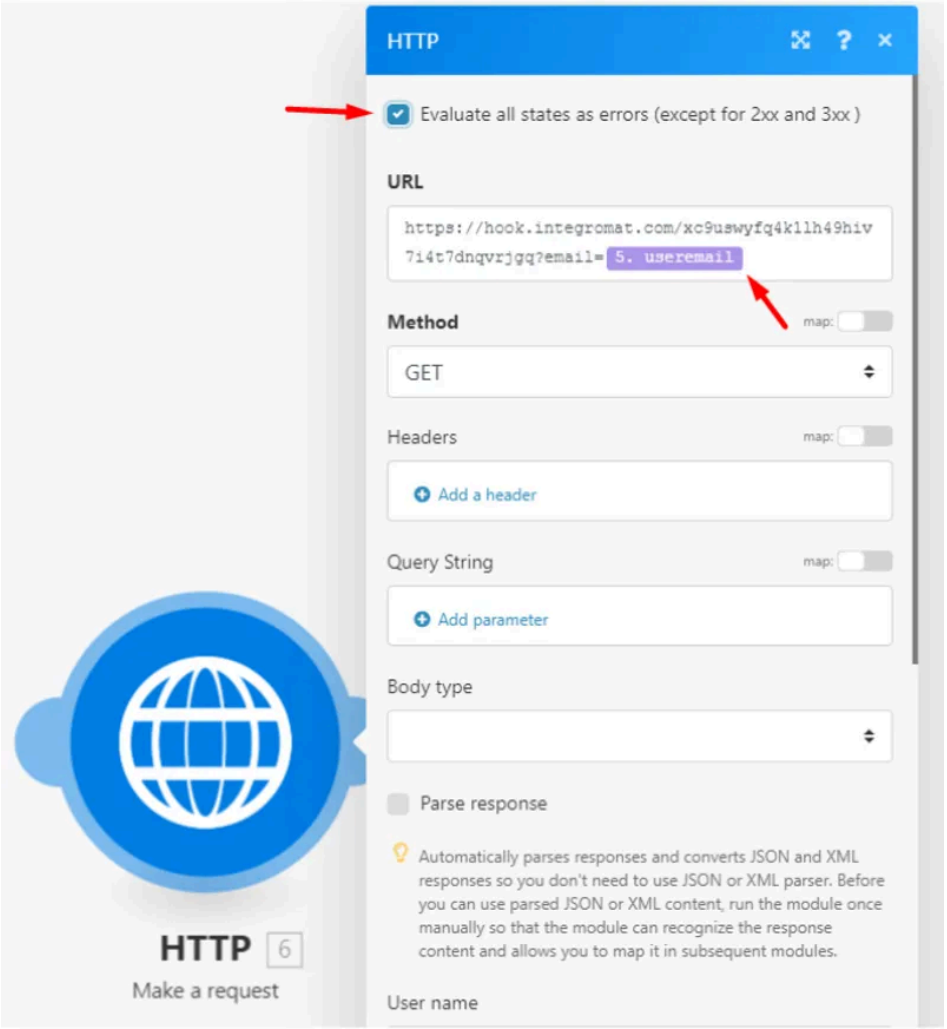


Solution - To conditionally throw an error you may replace the module where you want to throw an error, with an HTTP module then perform the search in a second scenario linking

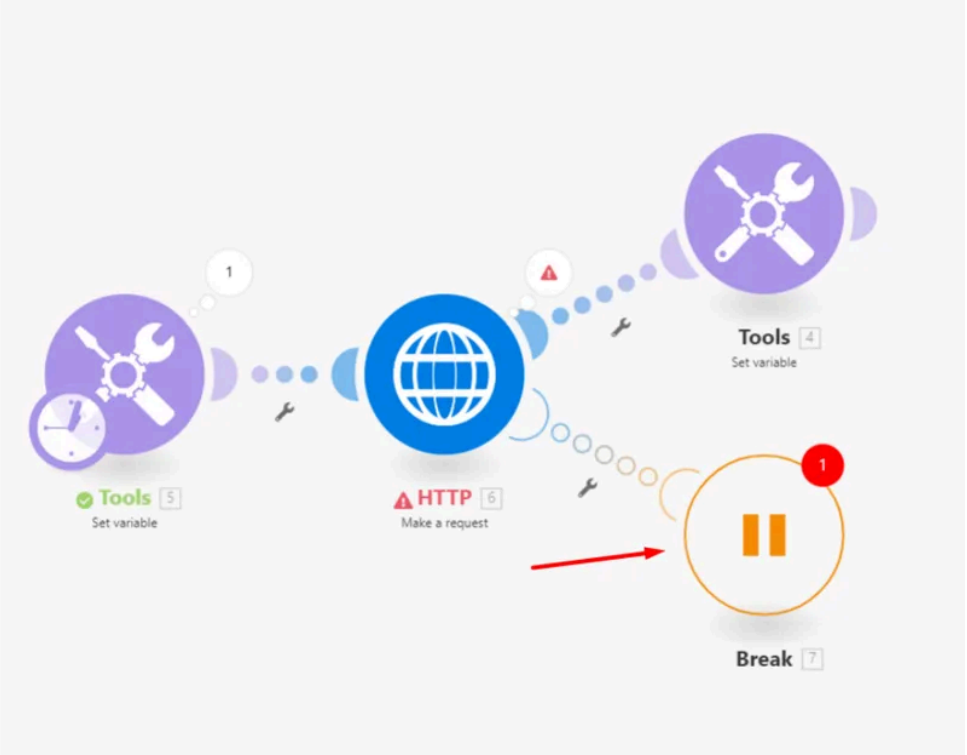
the HTTP module using webhook with a second scenario. If no result found you can customize the module to throw an error

Scenario One -

- Replace the Module where you want to throw an error with HTTP > Make a Request module
- Configure the URL within the query parameter that you will get from the Custom Webhook module and add an optional query parameter to search for the email
- Enable the advanced settings and check the evaluate all the states as errors.



- Add a Break handler in that HTTP module and configure the setup to run later.



Scenario 2 -

1. Setup Webhooks > Custom Webhooks as a trigger and copy the URL use it in the HTTP module as shown in the previous steps.
2. Here use the Zendesk > Search for a User module use the parameter from the HTTP module to perform a query. Enable Continue the execution of the route even if the module returns no results
3. Add a Router and create two routes
4. Consecutively setup the webhook response module

The following example returning the result - You will notice when the Zendesk module executes the API it doesn't send any error message but in the action, we're replicating the error using the HTTP module.


Integration Tools, HTTP
zendesk
Zendesk
Search for a User
Tools
Set variable
HTTP
Make a request
Break
Run once
OFF
SCHEDULING
CONTROLS
TOOLS
FAVORITES
1:15 PM Data is INVALID because of the error: [Error: 400 Bad Request]]
1:15 PM The operation was completed with a warning. Error: 400 Bad Request
1:15 PM The scenario was finalized.
1:15 PM The scenario run was completed.
LOG
Integration Webhooks, Zendesk
Webhooks
Custom webhook
Zendesk
Search for a User
Router
Match Found
Match Not Found
Webhooks
Webhook response
Webhooks
Webhook response
Stop
1:15 PM The scenario was finalized.
1:15 PM The scenario run was completed.
1:15 PM Preparing scenario for running.
1:15 PM Requesting execution.
The concept was saved.

6. Your organization

Learn about managing your subscription, organization, and teams:

Organizations & teams

Navigate organizations and teams, including adding users and managing roles

Subscription

Find details to manage your subscription, payments, and purchase extra operations

Administration

Discover features to manage organization usage and members

Access management

Set up SSO and domain claim to define Enterprise user log in

Make Managed Services (MMS)

Explore MMS capabilities for managing child organizations and their operations

6.1. Organizations & teams

Navigate organizations and teams, including adding users and managing roles:

→ **Organizations**

→ **Teams**

6.1.1. Organizations

Organizations are the basic containers into which all scenarios, users, and data belong.

Organizations usually represent a company that is a Make customer or partner. Each organization has a separate pricing plan and is billed separately. This means that you can also monitor your operations and data consumption for each organization separately.

Each organization allows you to select the geographical location of the data center where the organization's data will be stored and processed. Currently, Make supports two data center locations:

- United States (US)
- European Union (EU)

> The geographical location that you select for an organization does not represent where your company or your users are located, but where the data center that stores and processes your data is located.

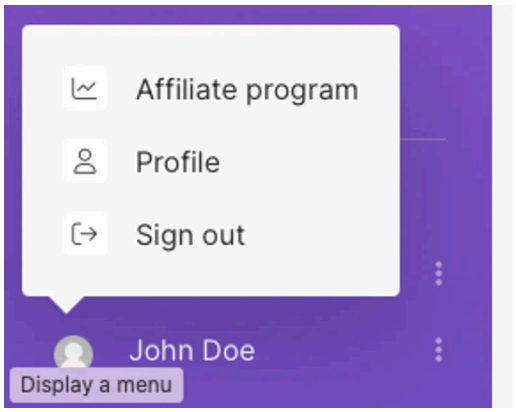
We recommend that you select the location closest to you to ensure low latency.

> You cannot change the location of the data center after you create the organization.

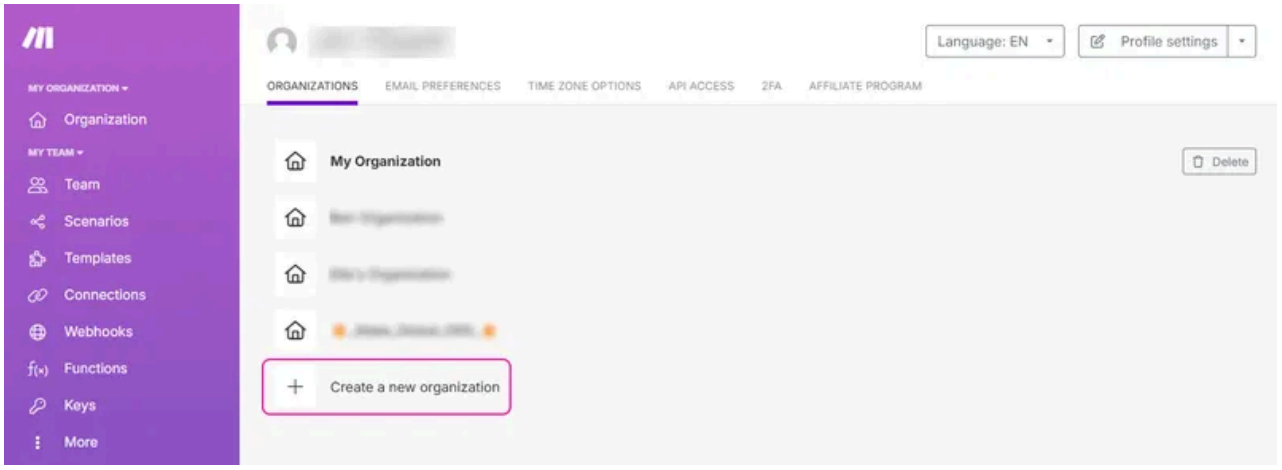
Create an organization

Any user can create their own organization, even if they are already a member of another organization.

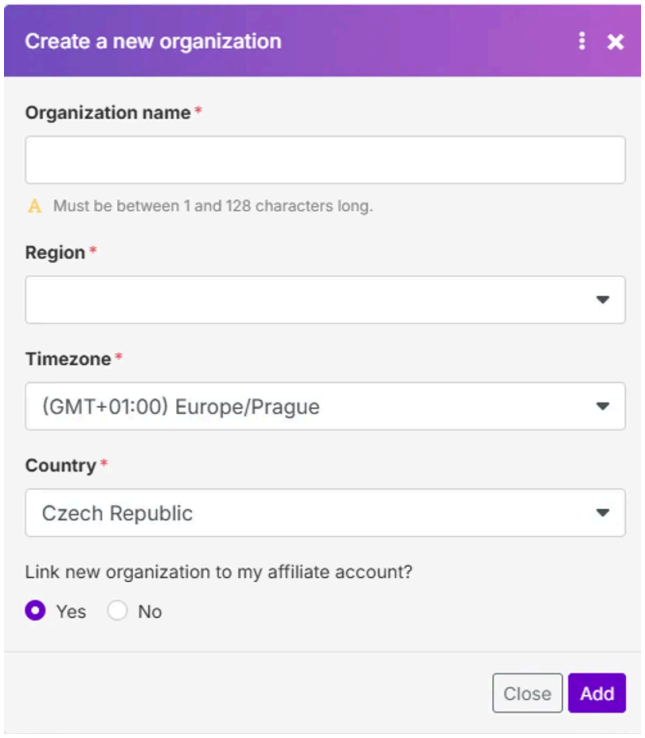
1. Click on your username in the bottom of the left sidebar.
2. Select **Profile**.



3 In the **Organizations** tab, click **Create a new organization**.



4 Fill in the details of the new organization. Note that **Region** represents the location of the data center that will store and process your data. **Country** represents your physical location but is currently not used by Make in any way.

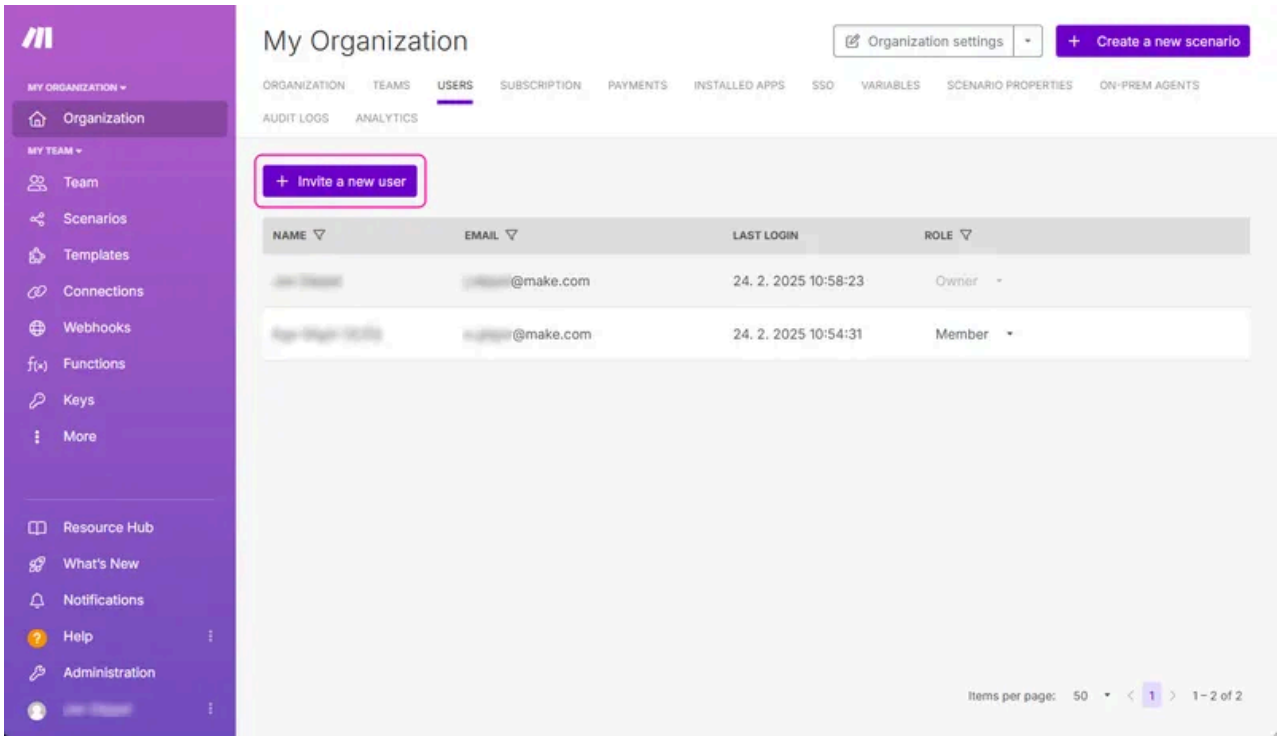


By default, the pricing plan for the organization is set to **Free**. The organization contains one team, called **My Team**.

Add users to an organization

To add users to an organization, you must be the **owner** or an **admin** of the organization. For more details about roles, refer to the **organization roles**.

1. In the left sidenav, click **Organization**.
2. Click the **Users** tab.
3. Click **Invite a new user**.



4 In the dialog box, fill in the details of the user.

- **Email** - The invited user's email address.
- **Name** - The name of the user. The user can change their name after they accept the invitation.
- **Role in the organization**
- **Team** - You must select at least one team.
- **Note** - an optional, custom message that is added to the invitation email.

5 Click **Save**.

The user will receive an invitation email with an **Accept invitation** button.

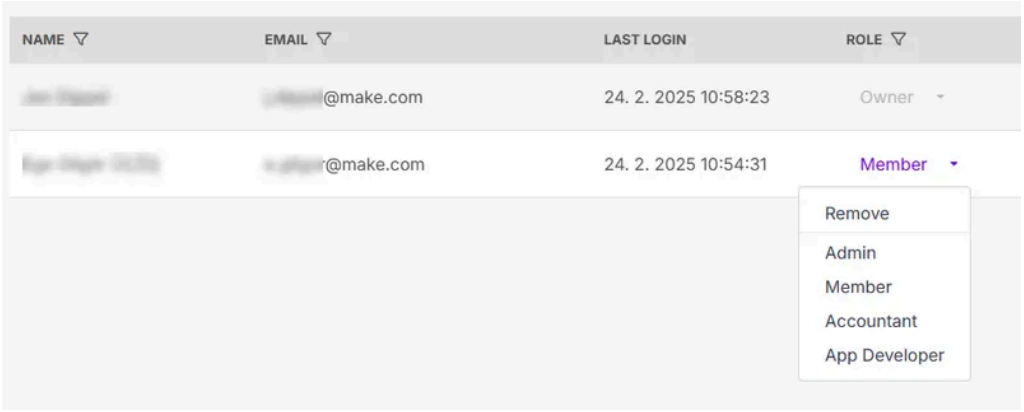
When the user accepts the invitation, Make adds them to the organization, as well as to the selected team with the assigned team member role.

> The invitation expires in 7 days. If the invited user does not accept the invitation:
>
> 1. Remove the invited user from your organization.
> 2. Send a new invitation.

Remove a user

To remove users from an organization, you must be the **owner** or an **admin** of the organization.

1. In the left sidenav, click **Organization**.
2. Click the **Users** tab.
3. Use the dropdown menu in the Role column to select **Remove**.



Organization roles

Make provides a two-level permission model. All users are members of at least one organization and one team. The permissions that users have are determined by the organization role and team role that they are assigned to.

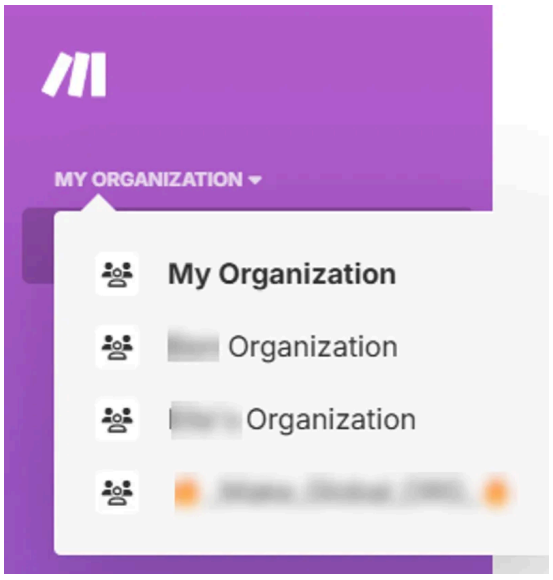
The following table describes actions that members of different organization roles can perform.

For information about team roles, **read the article about teams**.

	Owner	Admin	Member	Accountant	App Developer
View organization details	●	●	●	●	●
Modify organization	●	●			
Delete organization	●				
Transfer organization ownership	●				
Add and modify users in organization	●	●			
Manage all teams	●	●			
Install apps to organization	●	●			●
View payments	●	●		●	
Change payment methods	●	●			

Switch organizations

If you are a member of multiple organizations, you can change the active organization in the left menu. All items in the menu represent items that belong to the currently selected organization.

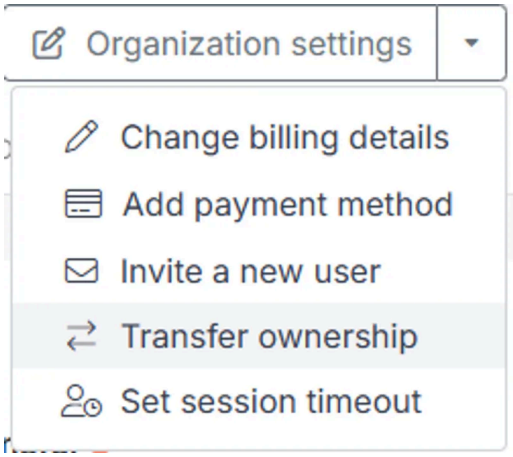


Transfer ownership

When you create an organization, you automatically become the owner of the organization. Organization owners can manage and access all teams in an organization. An organization

can have only one owner. If you are the owner of an organization, you can transfer ownership to any member of your organization.

1. In the left sidenav, click **Organization**.
2. Next to **Organization settings**, click the drop-down arrow to expand the menu.
3. Select **Transfer ownership**.



4. Use the drop menu to select the organization member you want to be the owner of the organization. All current organization members appear in the menu. Once a new member accepts the invitation, you can transfer ownership to them.
5. Click **Save**.

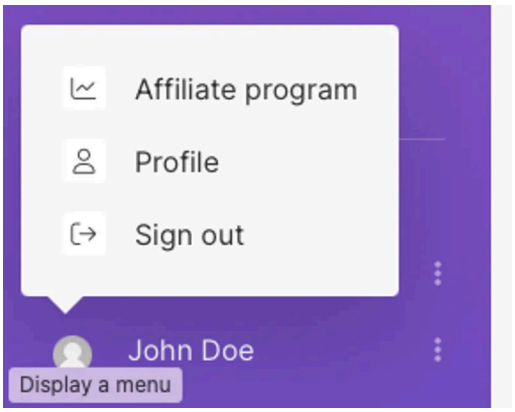
A purple pop-up appears confirming that ownership has been transferred successfully.

Delete an organization

You can delete an organization only if you are the owner of the organization. You can't delete an organization owned by other users.

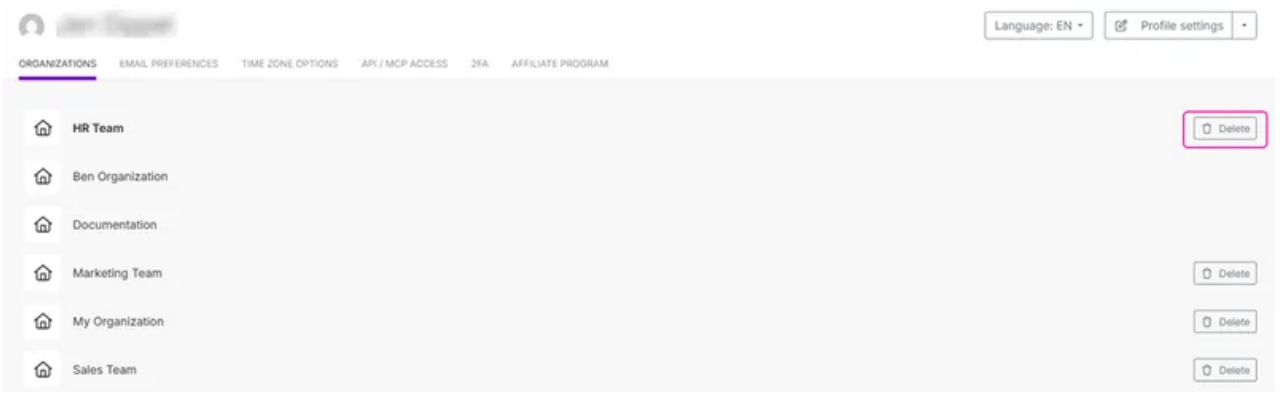
1. Click on your username in the bottom of the left sidebar.

2 Select **Profile**.

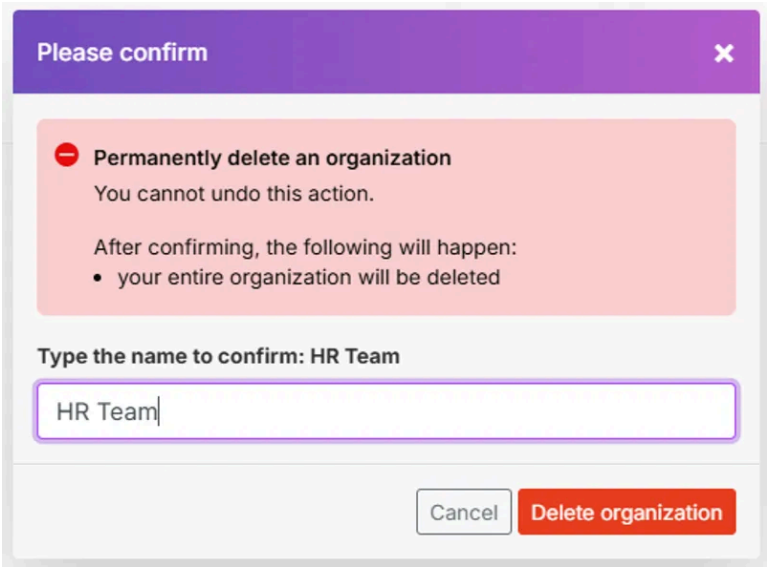


3 In the **Organizations** tab, you will see a list of all organizations you have access to.

4 Click **Delete** for the organization you want to delete.



5 Confirm that you want to delete the organization by typing the name of the organization in the field and click **Delete organization**.



> You can't undo this action. Your organization will be permanently deleted and it can't be retrieved.

Your organization is deleted.

6.1.2. Teams

To control access to Make and other Make data, you can assign users to teams. Members of a team have access only to scenarios and data that belong to their team.

The following items belong to a team:

- Scenarios
- Templates
- Connections
- Webhooks
- Keys
- Devices
- Data stores
- Data structures
- Custom functions

All of these items always belong to only one team. You can't change which team a particular item belongs to.

A user can be a member of any number of teams within the same organization.

Every organization must have at least one team.

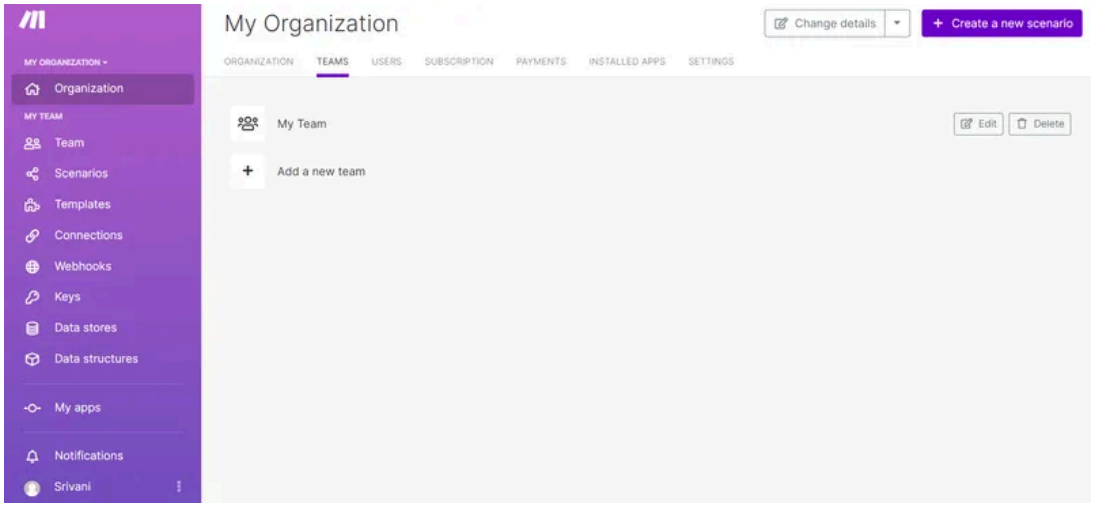
Teams and organizations

Organizations are the most basic organizational unit, or container, to which all elements within Make are assigned. An organization can contain any number of teams.

You register users into an organization. Then, you assign users to teams. A user can be a member of any number of organizations and any number of teams. A user can have a different role in each team.

> If you're on a plan lower than the **Teams pricing plan**, you can only have one team in your organization.

Create a team



You create teams inside organizations. You must be an owner or an admin of an organization to manage its teams.

1. In the left sidenav, click **Organization**.
2. Click the **Teams** tab.
3. Click **Add a new team**.
4. Enter the new team's name and click **Add**.

The new team appears in the list of teams. Click the team name or the **Edit** button to manage team members.

Manage teams

You can view the team's consumption of operations and data transfer, manage team members and their roles, and set up notifications.

Monitor operations and scenario executions

You can monitor operations and data transfer consumed by each team.

1. In the left sidenav, click **Organization**.
2. Click the **Teams** tab.
3. Select a team.

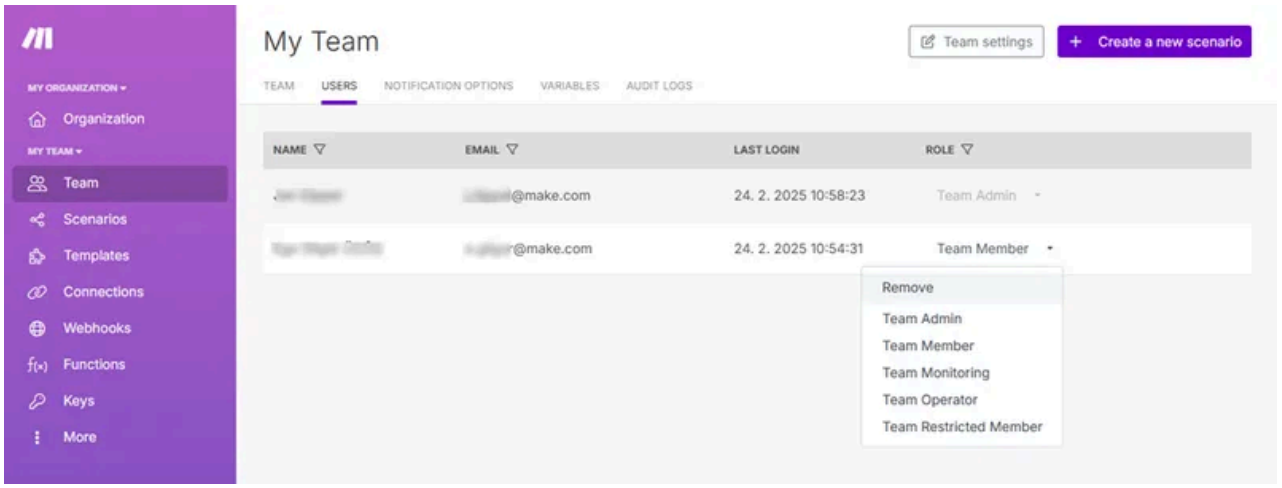
Since scenarios always belong to teams, their execution history is also tracked for each team separately.

Manage team members

When adding users to teams, you assign team roles to the users. Roles determine what permissions users have and what users can do with the team's data.

1. Click **Team** in the left sidebar.
2. Click the **Users** tab.

   The **Users** section shows all users in your organization and the role they have in the selected team. Users who don't belong to the selected team have their team role set to **None**.
3. To add a user to the team, click the **Role** menu and select a role other than **None**.



You can assign users to the following roles:

- **Team Admin** has full access to all data within the team. Only team admins can create custom functions.
- **Team Member** has full access to all data within the team, but cannot manage team members and cannot delete the team.
- **Team Monitoring** has read-only access to the team, scenarios, and templates.
- **Team Operator** has read-only access to all data within the team. In addition, the Team Operator can activate or stop scenarios and set up scenarios schedules.
- **Team Restricted Member** (deprecated) has full access to all data within the team, but cannot modify the team.

To **remove a user** from a team, select **None** from the **Role** menu. Make removes the user from its team role, which effectively removes the user from the team.

> Owners and admins of a team's parent organization always have Team Admin permissions.

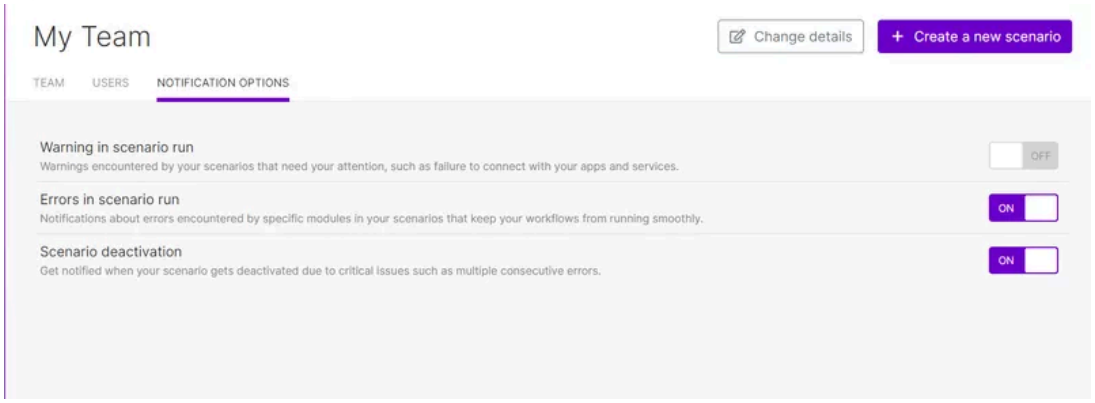
Manage team notification options

Using **Notification options**, you can control how Make notifies team members about warnings and errors in Make.

1. Click **Team** in the left sidebar.

2. Click the **Notification Options** tab.

3 Toggle the notification **On** or **Off** as needed.



6.2. Subscription

Find details to manage your subscription, payments, and purchase extra operations:

→ **Change your subscription**

→ **Cancel your subscription**

→ **Extra operations**

→ **Monthly and annual subscriptions**

→ **Change payment details**

→ **Coupons**

→ **Upgrade to the Enterprise plan**

→ **Request sales tax exemption**

6.2.1. Change your subscription

You can change your subscription on the Organization dashboard.

If you decide to upgrade or downgrade, we charge your payment method and the new billing cycle runs. These terms do not apply when downgrading to the Free plan.

> Make has a no cash refund policy.
>
> If you decide to upgrade, we convert your unused operations to credit. If you decide to downgrade, we transfer your unused operations to the new plan. Learn more in the **upgrade** and **downgrade** sections.

Upgrade your subscription

You can upgrade between plans and between tiers. Plans differ in features, and tiers refer to the number of operations.

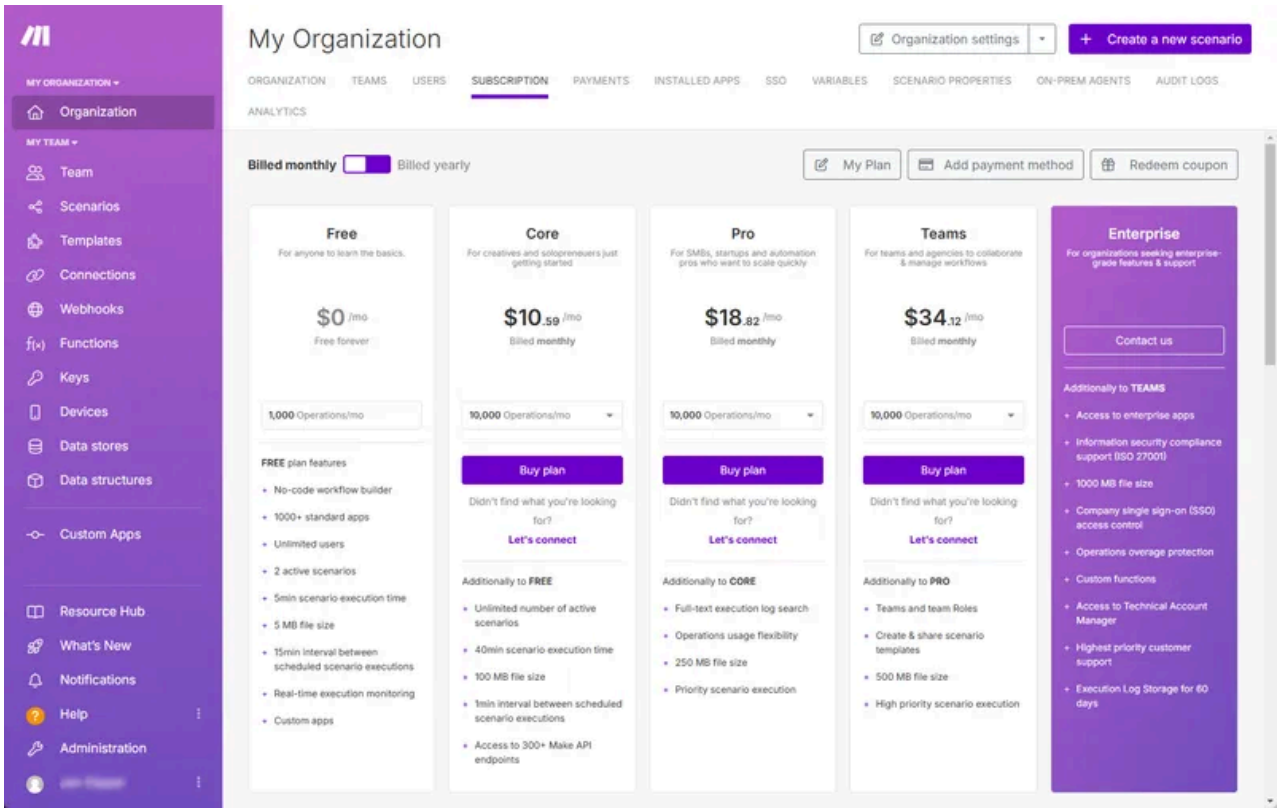
An upgrade can be:

- Changing from a lower plan to a higher plan, regardless of the tier (e.g. changing from the highest tier in Core to the lowest tier in Teams)
- Changing within the same plan to a higher tier (e.g. from 10,000 operations per month in Pro to 150,000 operations per month in Pro)
- Changing from a monthly to an annual subscription to an equal or higher tier (e.g. changing from 10,000 per month in Pro to 120,000 per year in Pro)

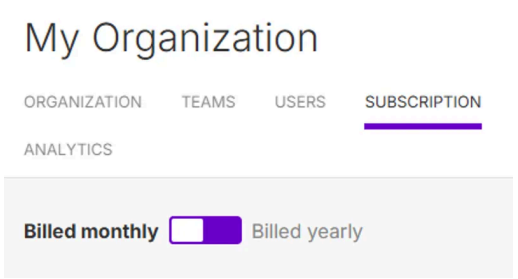
1. In the left sidebar, click **Organization**.

2. Click the **Subscription** tab.

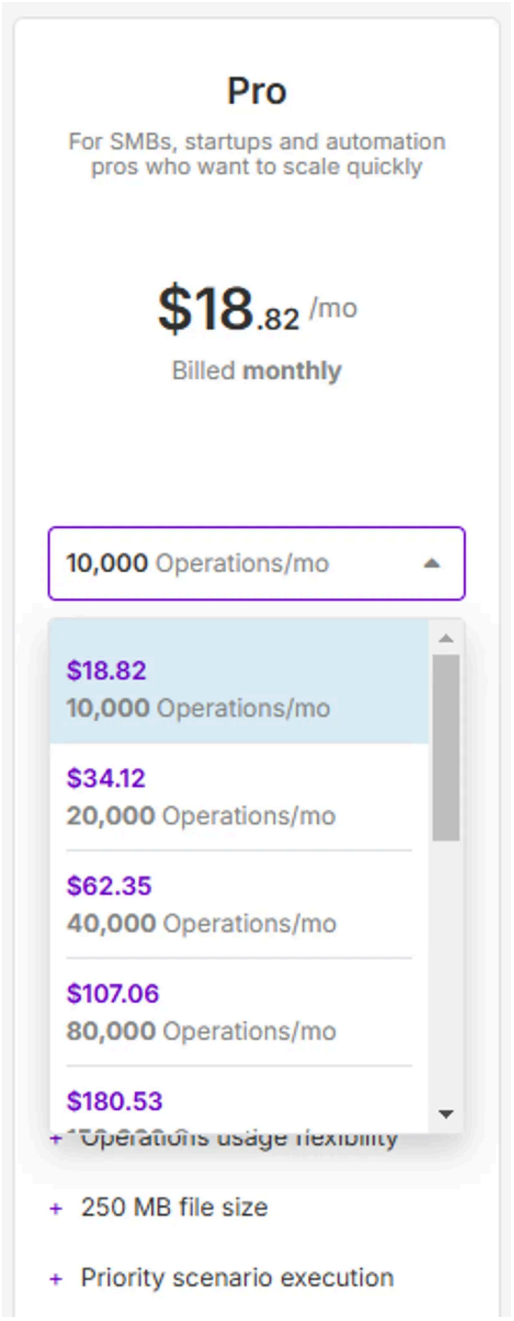
   A purple border and highlight mark your current plan.



3 Use a toggle to select a monthly or annual subscription.



4 Click the arrow to open a drop-down menu with different operation tiers. Choose the number of operations in your new plan.

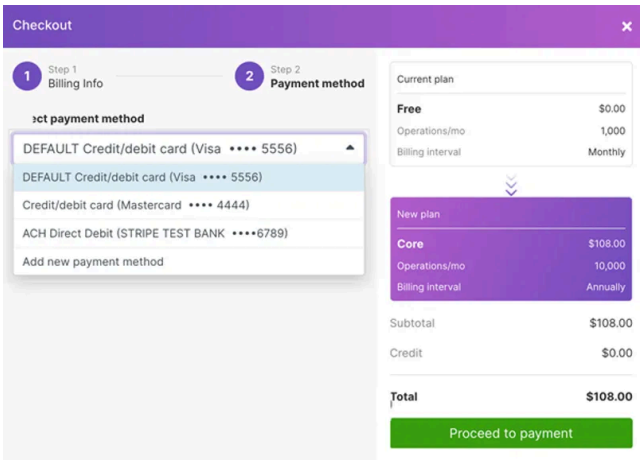


5 Click **Buy plan** for your selected plan and tier. Annual subscriptions appear as **Buy yearly plan**.

6 Check your billing info and use the edit button to update if necessary.

7 Click **Select payment method.**

8 Use the drop down menu to select a payment method. Click **Proceed to payment**.



9 Follow the on-screen instructions from Stripe checkout.

10 Reivew your payment information and click **Subscribe**.

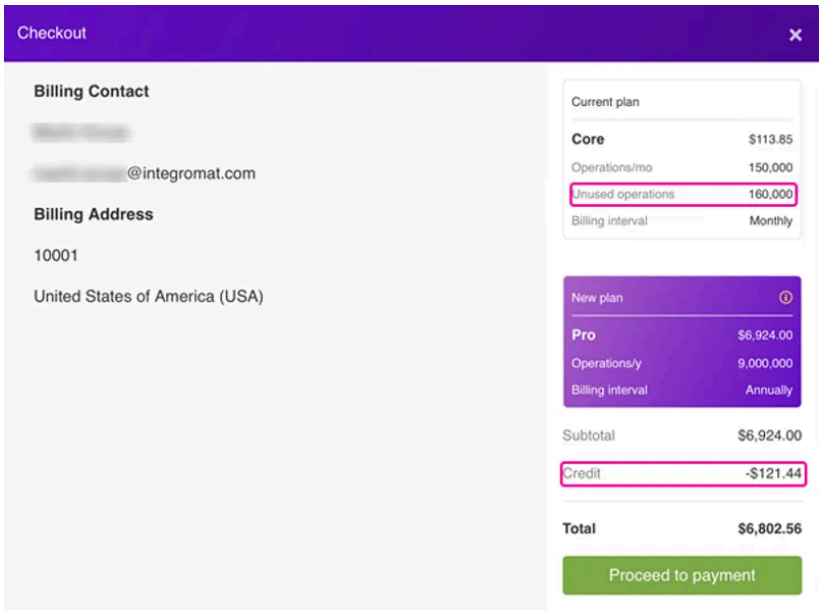
Credit

If you decide to upgrade, we convert your unused operations (including purchased extra operations) to credit in dollars according to the price per operation in your current plan.

We deduct this credit immediately from your payment for the new upgraded subscription. The credit is visible in the checkout page calculation.

For example, you want to upgrade from Core to Pro, and you have 160,000 unused operations. We convert these unused operations to credit. You can find the calculation logic below.

1. We calculate the price per single operation in your original price plan.
   We do this by dividing the price of the plan with the number of operations.
   In this example the original price of the Core plan is $113.85 and it has 150,000 operations per month.
   $113.85 divided by 150,000 operations equals $0,000759 per one operation.
2. We multiply the price per operation with the number of unused operations.
   Our price per operation is $0.00075 and there are 160,000 unused operations.
   $0.000759 multiplied with 160,000 unused operations equals $121.44, which is the amount of the credit you can see deducted on the checkout page.



The credit amount cannot exceed the amount of your total bill.

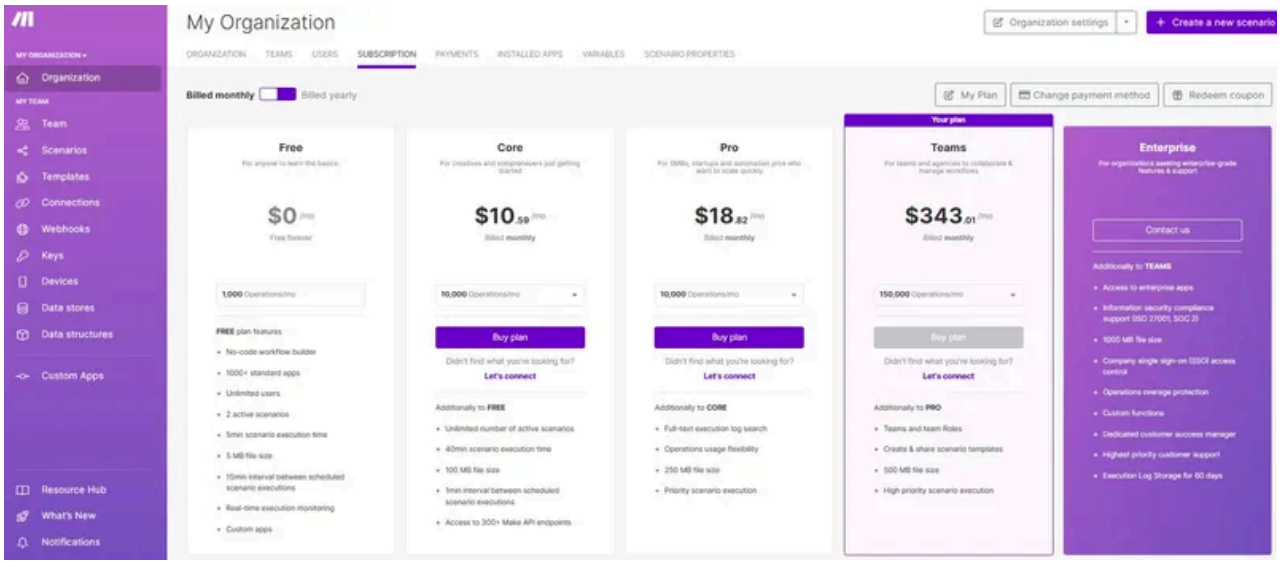
Downgrade your subscription

You can downgrade between plans and between tiers. Plans differ in features, and tiers refer to the number of operations.

A downgrade can be:

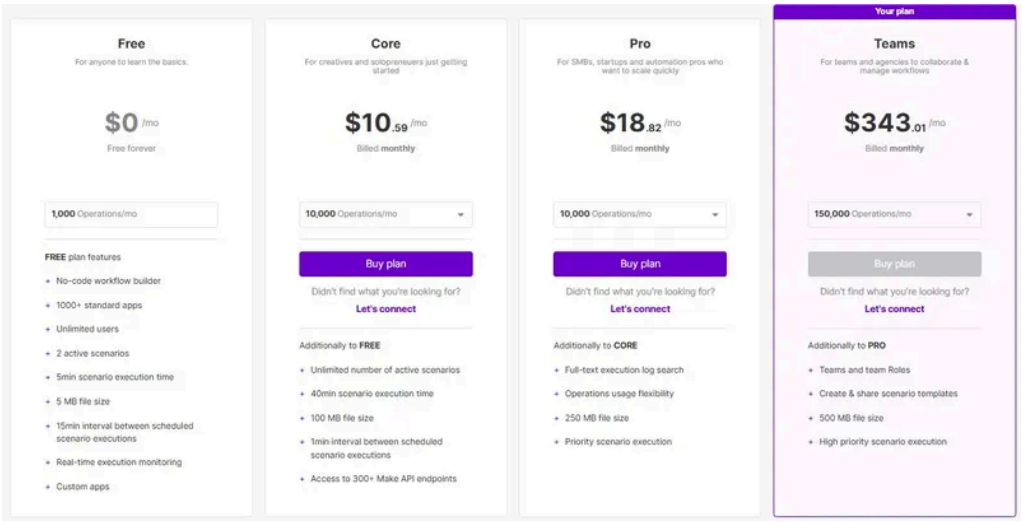
- a change from a higher plan to a lower plan, regardless of the tier (e.g. changing from the lowest tier in Teams to the highest tier in Core)
- a change within the same plan to a lower tier (e.g. changing from 150,000 operations in Pro to 10,000 operations in Pro)
- a change from an annual to a monthly subscription to a lower tier is a downgrade (e.g. changing from 240,000 operations per year in Pro to 20,000 operations per month in Pro).

1 On the Organization dashboard, click the **Subscription** tab.



2 Select your new plan and the number of operations.

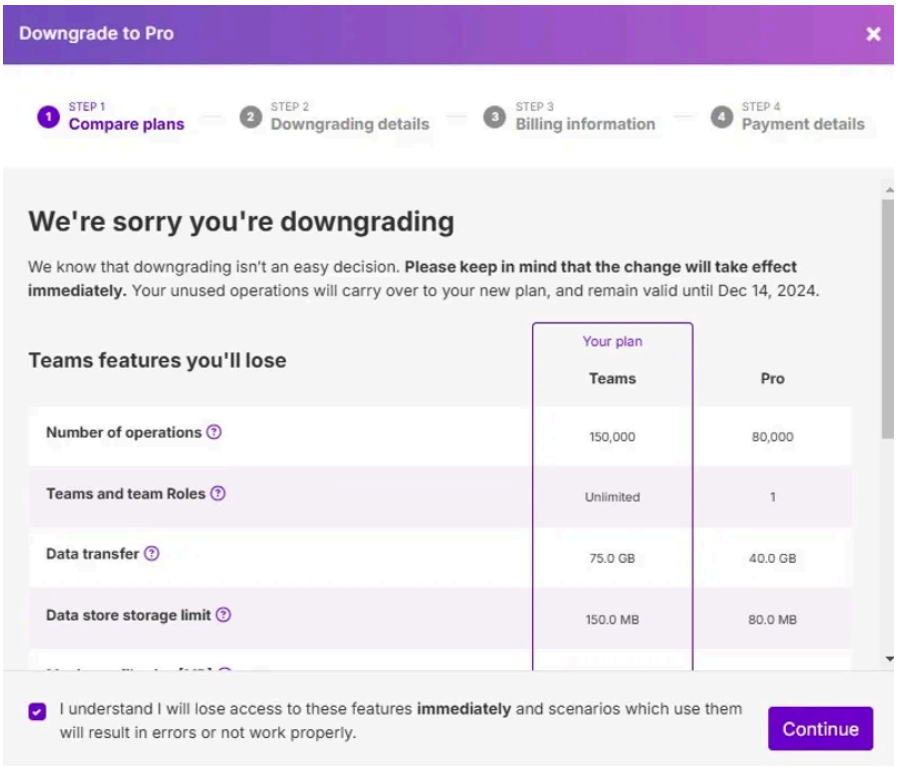
In the example below, the current plan is Teams with 150,000 operations per month. So you can downgrade to any tier in Core or Pro, or select fewer operations in the Teams plan.



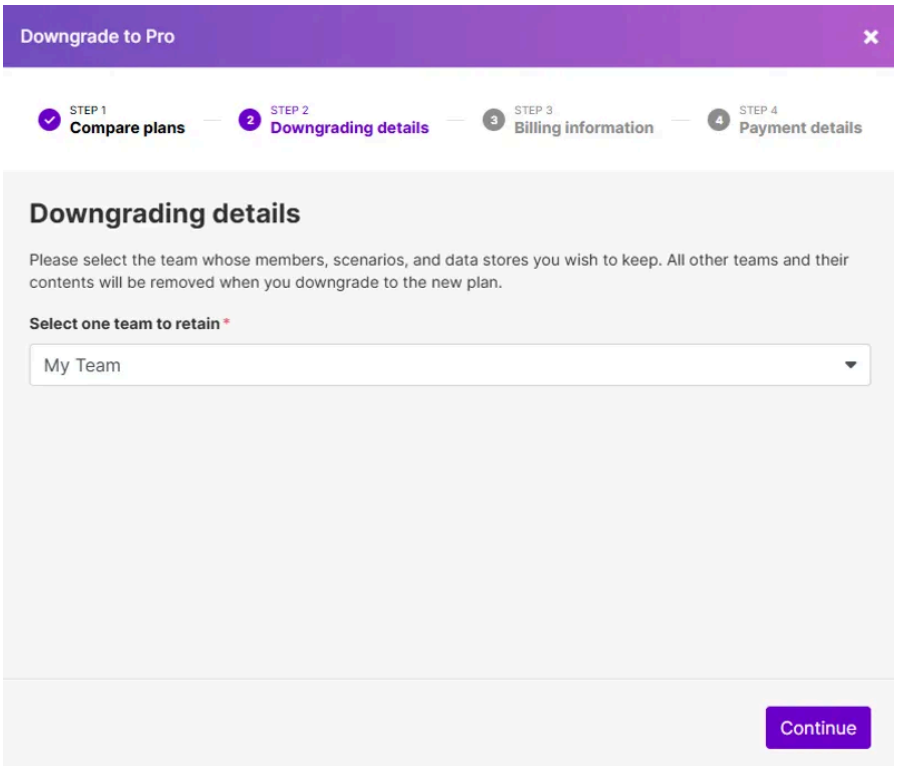
3 In your new plan and tier, click **Buy plan**.

4 In a new dialog box, you will see your current settings on the left side of the table and on the right side you will see how your plan will change.

Verify that you understand the changes by clicking the checkbox and then click **Continue**.

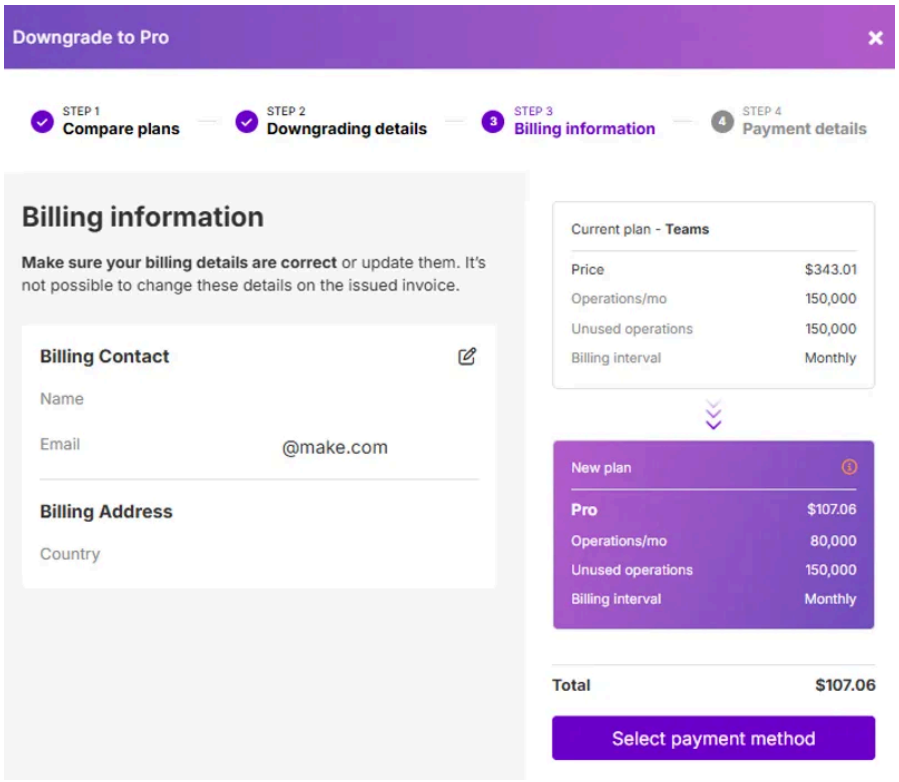


5 Depending on your downgrade choice, your new plan may allow for only one team. If so, select one team to retain and click **Continue**.

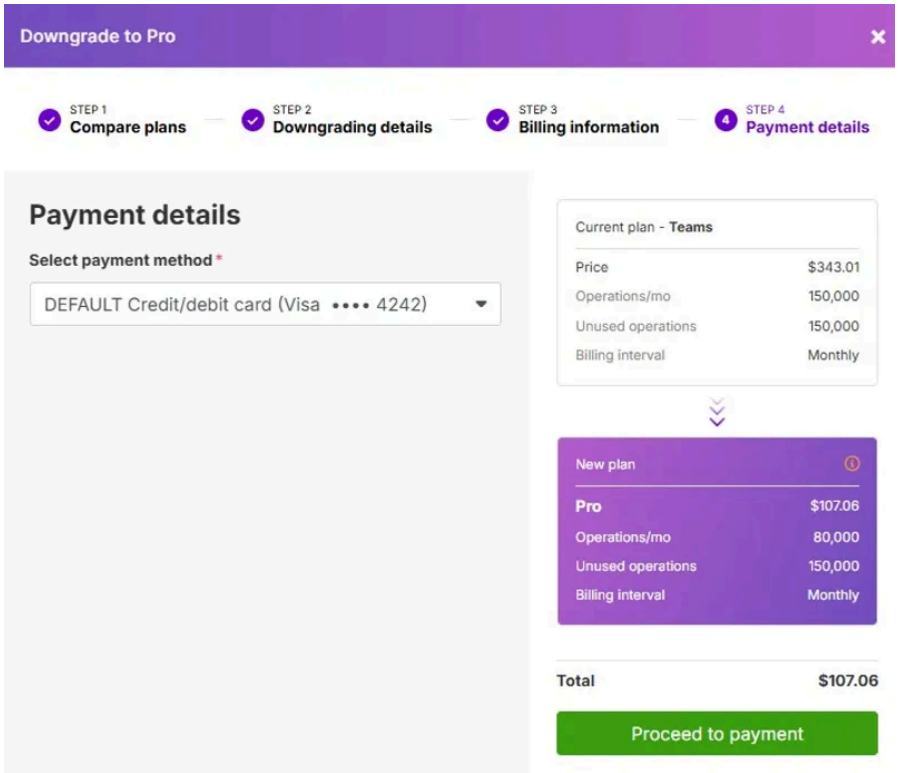


6 Verify that your billing information is correct. If necessary, make changes using the edit button.

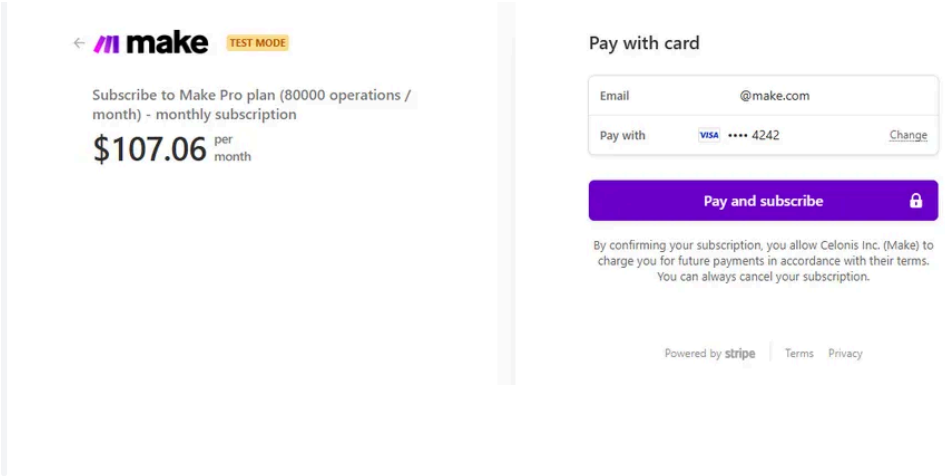
7 Click **Select payment method**.



8 Select a payment method and click **Proceed to payment**.



9 On the Stripe checkout screen, review your details and click **Pay and subscribe**.



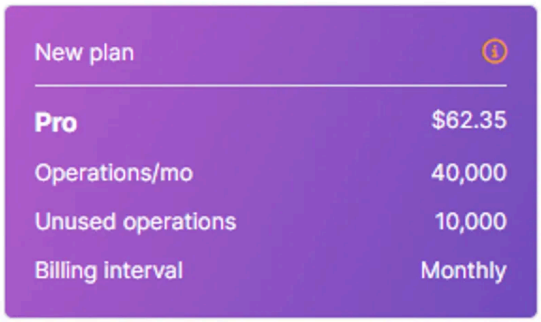
A notification pops up saying that your plan was successfully changed. Your new plan is highlighted on the Subscription page.

Transfer of unused operations

If you downgrade, we transfer your unused operations to the new downgraded plan. You can use the transferred operations in the new downgrade plan for the length of your new billing cycle:

- 30 days for a monthly plan
- 365 days for an annual plan

For example, you decide to downgrade to the Pro plan and you have 10,000 operations left. In the Pro plan, you select the tier with 40,000 operations/month, so in the first month of the downgrade, you have 50,000 operations available.

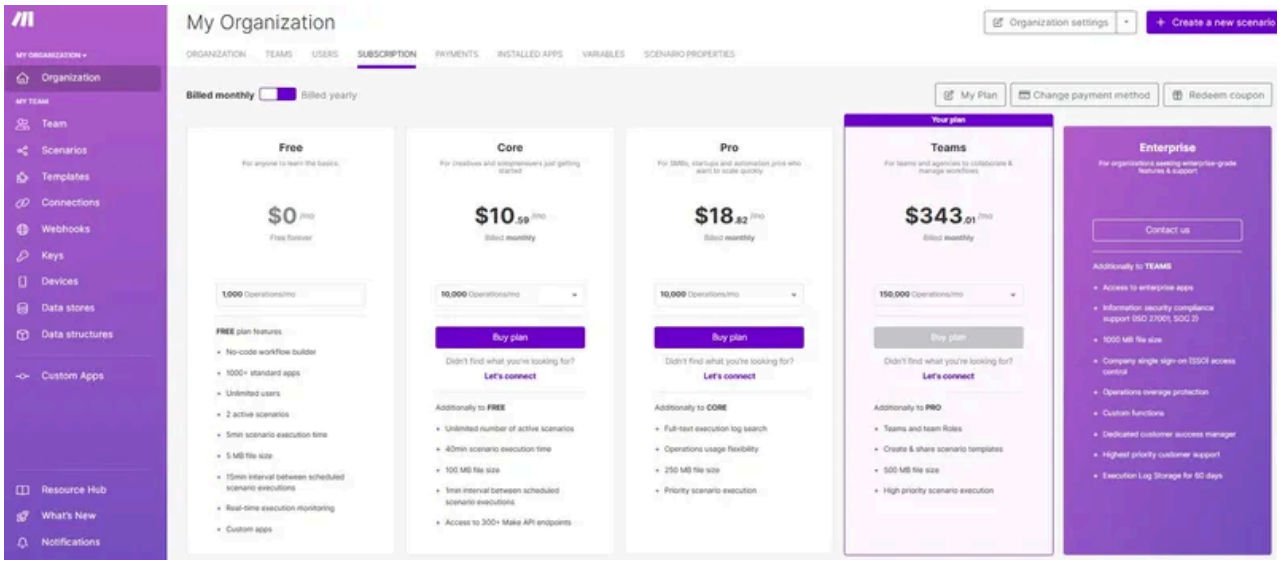


> For a Core plan with an annual subscription, operations reset every month and your extra operations expire with the monthly reset.

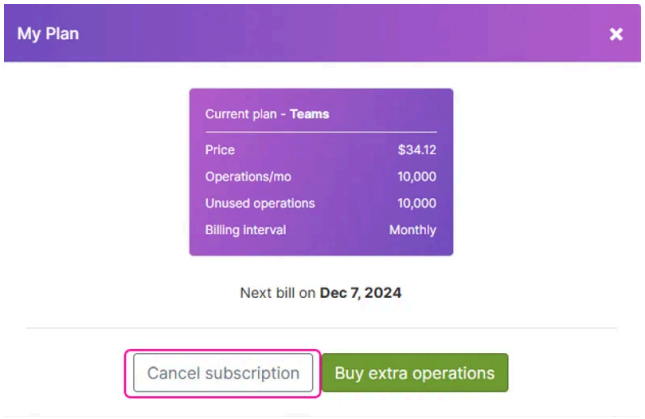
6.2.2. Cancel your subscription

You can cancel your subscription at any time.

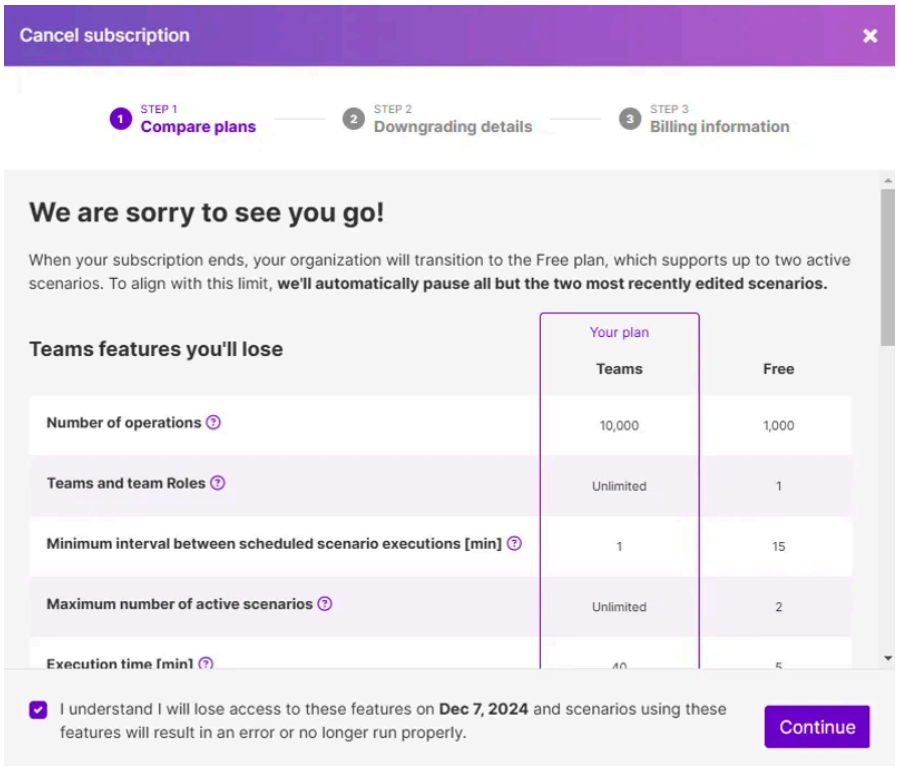
1. On the Organization dashboard, go to **Subscription > My Plan**.



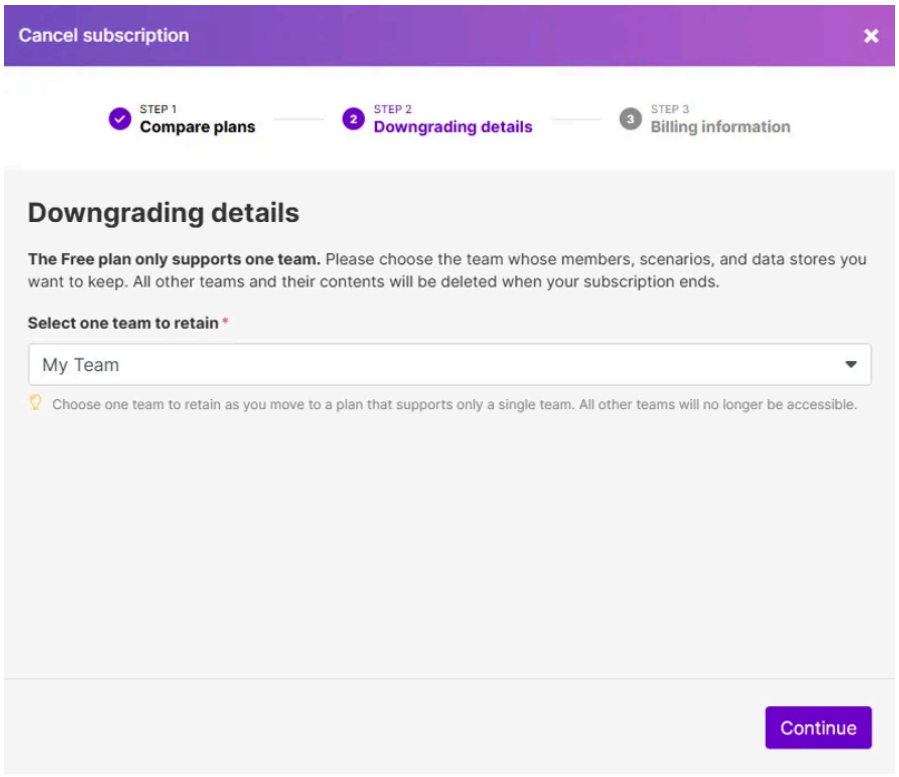
2. Click **Cancel subscription**.



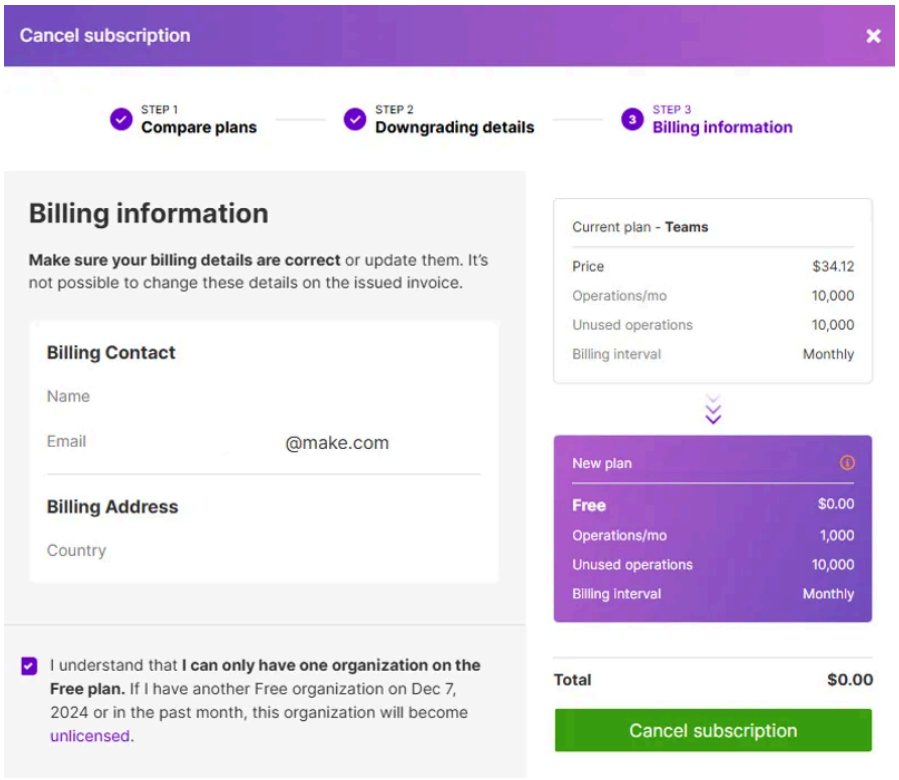
3. In a new dialog box, you will see details regarding the plan you are canceling and the features you will retain in the Free plan. Verify that you understand the changes by clicking the checkbox and then click **Continue**.



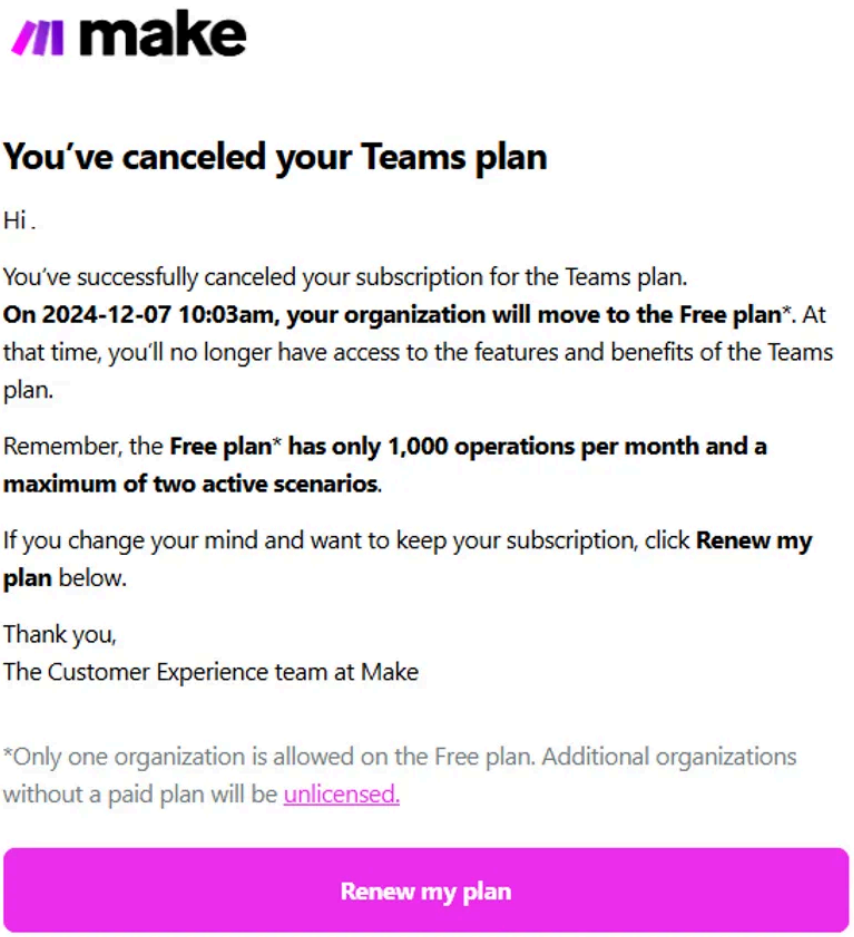
4 The Free plan only supports one team. Select one team to retain and click **Continue**.



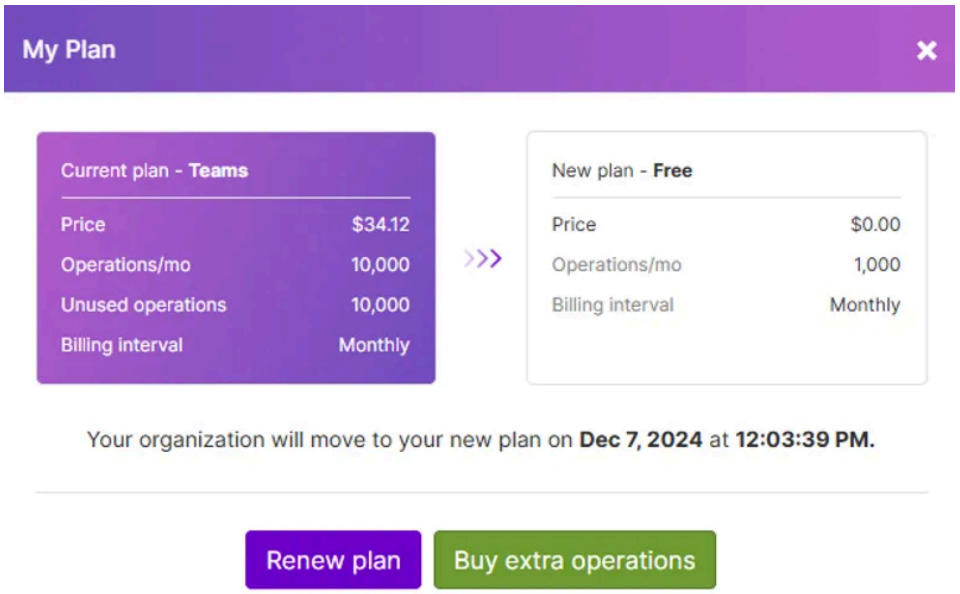
5 To finalize the cancelation, you will be asked to verify that you understand that you will only have one organization under the Free plan after your cancelation date. Additionally, if you have another Free organization, that organization will become unlicensed. Click **Cancel subscription**.



6 We will send you a confirmation email similar to the following example:



If you change your mind, you can renew your plan.



What happens after cancelation

We don't charge you for the next month, but you can still use your current subscription until the end of your billing cycle. After that, we switch you to the Free plan (assuming that you are not the owner of another organization with a Free plan).

We don't transfer your unused operations to that Free plan. You will have the standard Free subscription (i.e. 1,000 operations per month). This applies to both monthly and annual subscriptions.

> You can have only one free organization under your account.
>
> When you set an organization to a Free plan (either by creating a new free organization or setting a current one to free), for the next one month it is not possible to set another organization to a Free plan.
>
> If you have multiple organizations under your account and one subscription is already canceled, each subsequent organization you cancel is set as an organization without a license. You cannot run any scenarios under this organization until you subscribe to a new plan.

6.2.3. Extra operations

Validity of extra operations

Extra operations are valid for the current billing cycle. For example, you have monthly billing and your billing cycle is from the first until the last day of the month. If you buy extra operations on the 20th, your extra operations are valid until the last day of that month.

If you have an annual plan, extra operations expire at the end of your billing year.

> For the Core plan with an annual subscription, operations reset every month and your extra operations expire with the monthly reset.

Buy extra operations manually

You can buy extra operations at any time from your organization dashboard. Make consumes the operations included in your subscription first and then your extra operations. For example, if your subscription has 40,000 operations per month, you use those operations before consuming any extra operations you purchase. Operations 0 to 40,000 are your subscription. Operations 40,001 and onward are your extra operations.

> Extra operations also give you additional data transfer but do not impact other usage allowance parameters.

1. In the left sidebar, click **Organization**.
2. On the Organization dashboard, under **Usage reset**, click **Buy extra operations**.
3. Use the + and - buttons to enter how many extra operations in units of 1,000.

Price per extra operation

The price for the extra operations is identical to the price per operation within your pricing plan and tier.

For example, if you are subscribed to a $9 plan and have 10,000 operations, 1 operation costs you $0.0009 ($9 / 10,000 operations). If you decide to buy 1,000 extra operations, it costs you $0.9 (1,000 operations * $0.0009). If you decide to buy 10,000 extra operations, it costs you $9 (10,000 operations * $0.0009).

4. Click **Select payment method**.

5. Use the drop-down menu to select a payment card.

   You can only use credit or debit cards to pay for extra operations.

6. Click **Proceed to payment**.

7. You will see the extra operations on your dashboard.

Enable extra operations auto-purchasing

You can ensure your scenarios keep running with extra operations auto-purchasing. Extra operations auto-purchasing covers you when you reach your limit of operations by automatically purchasing extra operations.

When you use all of the operations in your subscription, Make automatically purchases 10,000 extra operations. All automatic purchases are in units of 10,000 operations. Make then charges you for the 10,000 extra operations.

If you use all 10,000 extra operations, Make auto-purchases 10,000 operations again. Make limits auto-purchasing to the number of operations in your subscription. For example, if your plan has 80,000 operations, Make auto-purchases a maximum of 8 times.

When you enable extra operations auto-purchasing, you will incur fees for the auto-purchased extra operations and a 30% surcharge. This surcharge covers the cost and flexibility of keeping your scenarios running when you reach your quota. See the **example**.

To enable extra operations auto-purchasing:

1. In the left sidebar, click **Organization**.

2. On the Organization dashboard, click the toggle for extra operations auto-purchasing.



3. In the pop-up, select a payment method and click **OK**.

   Extra operations auto-purchasing only accepts card payments.

Example

Imagine you have a monthly suscription of 80,000 operations and have enabled extra operations auto-purchasing. Once you reach your 80,000 limit, the following happens:

- Make automatically purchases 10,000 operations and keeps your scenarios running.
- Make charges you for the 10,000 operations based on your plan and adds a 30% surcharge.

Let's assume that your scenarios use all 10,000 extra operations before the monthly billing cycle renews. Extra operations auto-purchasing buys an additional 10,000.

So far, extra operations auto-purchasing has purchased a total of 20,000. Your auto-purchasing limit per monthly cycle is 80,000 operations. In this example, extra operations auto-purchasing is only possible six more times, i.e. 60,000 extra operations.

> If your payment fails, your scenarios pause until you update your payment method. Make sends you an email notification when your payment fails.

Disable extra operations auto-purchasing

1. In the left sidebar, click **Organization**.

2. On the Organization dashboard, click the toggle for extra operations auto-purchasing.

3. In the pop-up, click **Turn off**.



6.2.4. Monthly and annual subscriptions

You can choosen between a monthly or annual subscription.

Monthly subscription

If you have a monthly subscription, your billing cycle is one month. For example, if you subscribe on 12 January, your next payment is due on 12 February.

If you have a monthly subscription, you get the monthly allocation of operations in the tier you choose.

Annual subscription

If you have an annual subscription, you pay the whole yearly amount up front and we allocate all the operations that belong in the specific plan and tier you chose. There is no monthly operations allotment, but the bulk of operations is available for you to use flexibly over the year.

> With the Core plan, you always have monthly allocated operations, even if you have an annual subscription. Only Pro and Teams plans have an annual package of operations.

Change your billing cycle

If you change from a monthly to an annual subscription, and vice versa, we look at your current and future operations tier. The tier determines whether the action is an upgrade or a downgrade.

Changing from a monthly to an annual subscription to an equal or higher tier is an **upgrade** (e.g. changing from 10,000 per month in Pro to 120,000 per year in Pro).

Changing from an annual to a monthly subscription to a lower tier is a **downgrade** (e.g. changing from 240,000 operations per year in Pro to 20,000 operations per month in Pro).

6.2.5. Change payment details

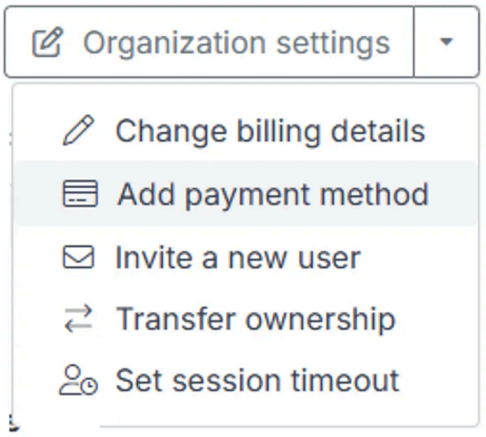
The organization dashboard lets you change your payment information at any time. Make accepts the following payment methods:

- Credit card
  - Valid for the following:
    - Monthly and annual subscriptions
    - Extra operations
- ACH direct debit (US bank accounts)
  - Valid for monthly and annual subscriptions only

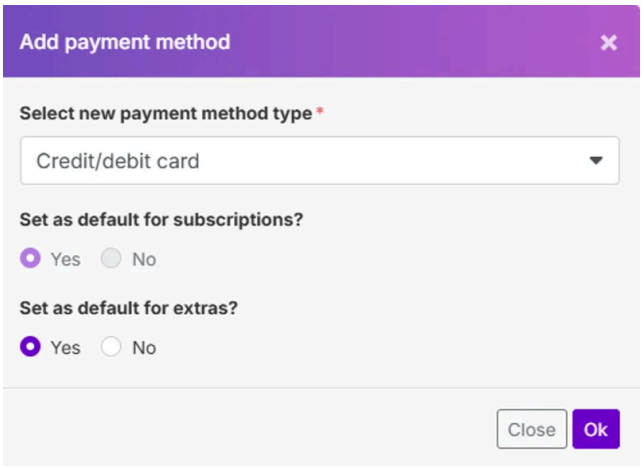
Enter credit card payment details

1. In the left sidebar, click **Organization**.

2. Under **Organization Settings**, select **Add payment method**.



3. Under **Select new payment method type**, select **Credit/debit card.**



4. Under **Set as default**, select **Yes** if you want the card you enter as your default payment method.

5. Click **OK**.

6. A pop-up appears. Follow the on-screen instructions and enter the required information.

7. Click **Save card**.

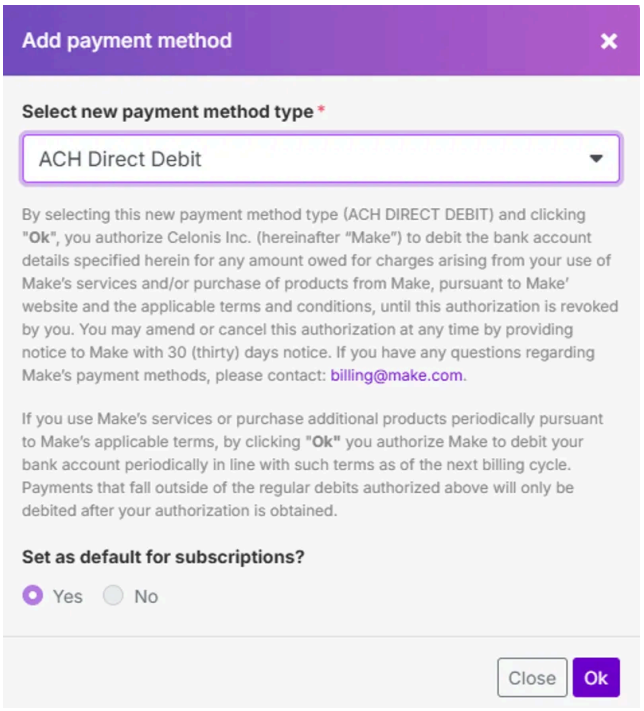
   A confirmation message appears in the lower right.

Enter ACH direct debit (US) details

1. In the left sidebar, click **Organization**.

2. Under **Organization Settings**, select **Add payment method**.

3. Under **Select new payment method type**, select **ACH Direct Debit**.



4. Under **Set as default**, select **Yes** if you want to save this account as your default payment method.

5. Click **OK**.

6. A pop-up appears. Follow the on-screen instructions and enter the required information.

   Upon completing the instructions, the popup disappears and a confirmation message appears in the lower right.

Change your billing address

You can change your billing address anytime you need to in Make.

1. In the left sidebar, click **Organization**.

2. Under **Organization Settings**, select **Change billing details**.

3. Enter your information in the allocated spaces (email, name, address).

If you have a company account, select **Yes** under **I am a business**.

If you want Make to issue the invoice to your company (and not to you personally), make sure to add your company name and Company Tax ID.

4 Click **Save**.

The new billing information automatically shows on your next invoice.

> Only customers in the US pay sales tax. We calculate sales tax based on your zip code.

What to do if your payment fails

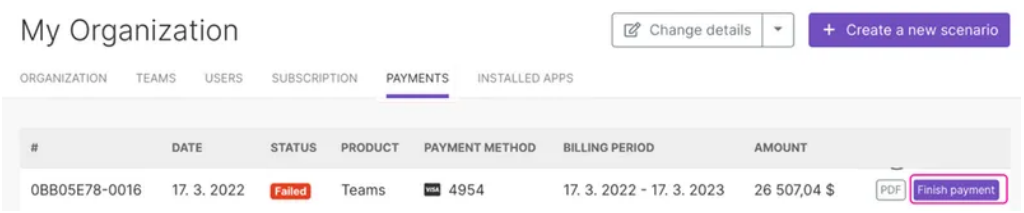
If Make cannot process your payment, you will receive an email. There is a seven-day grace period to update your payment method. When the grace period ends, your subscription is automatically canceled. If you use ACH direct debit for a failed payment, your subscription is renewed but may be canceled if the new ACH payment fails.

If your payment fails, follow these steps:

1 In the left sidebar, click **Organization**.

2 Click the **Payments** tab.

3 Click **Finish payment**.



4. A new tab opens. Follow the on-screen instructions to enter a new payment method. You can enter either credit card or direct debit payment information.

5. Review the information you entered. Click the **Pay** button to begin the transaction.

6. Confirm payment by checking the **Payments** tab.

> ACH direct debit payments take up to 5 business days to complete. As a result, the **Payments** tab may not reflect a recent direct debit payment.

View and download invoices

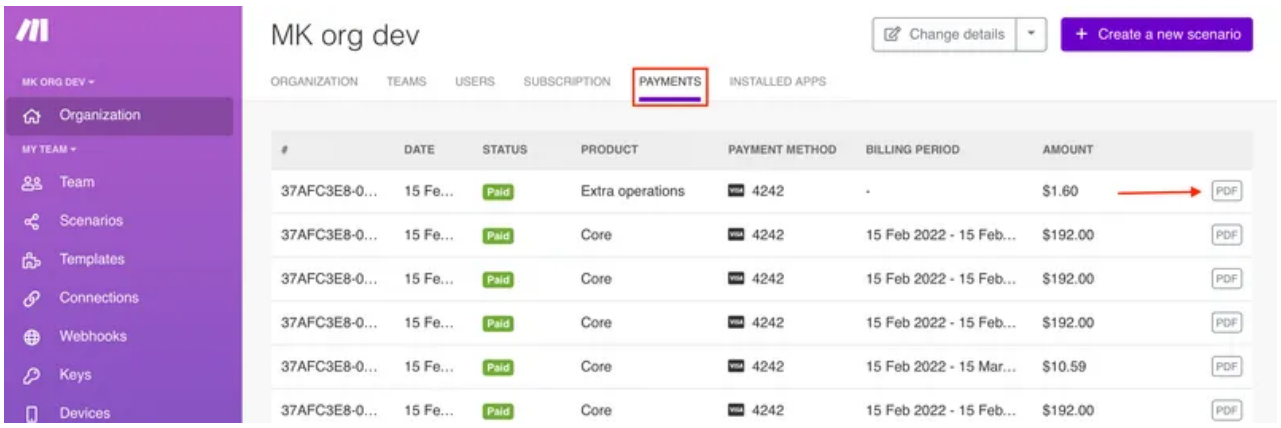
Make sends an invoice for every payment to the email address associated with your organization. You can **change your email address** at anytime.

You can download past invoices from your organization dashboard:

1. In the left sidebar, click **Organization**.

2. Click the **Payments** tab.

3. Click **PDF** for the invoice you want to download.



Your browser automatically downloads the PDF file. Check your Downloads folder or your browser's Downloads tab to find the file.

> ACH Direct debit payments take up to 5 business days to complete. As a result, the **Payments** tab may not reflect a recent direct debit payment.

6.2.6. Coupons

Occassionally, we send coupon codes via email. A coupon code may look like this: `xrCeZkqA`

A coupon may offer one of the following:

- Free trials of a subscription plan
- Discounts on subscription plans, such as your current plan or higher
- Extra operations

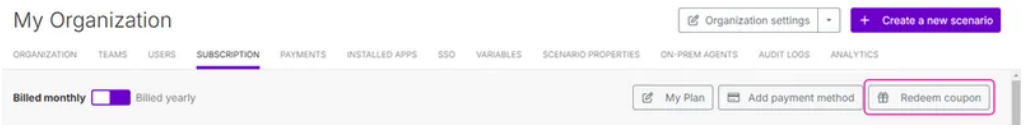
> Each coupon is unique and redeemable only once.

Redeem coupons

Only organization admins and owners can redeem coupons. If you are not an admin or the owner of an organization, contact them to redeem your coupon.

1. In the left sidebar, click **Organization**.

2. Under the **Subscription** tab, click **Redeem coupon.**



3. Enter your coupon code and click **Redeem coupon**.

4. In the payment checkout, your coupon appears in the details of your plan. Review the details and click **Redeem coupon**.

> Depending on the details of your discount and your plan, you may be redirected to a payment gatemey to finish redeeming your coupon.

To confirm the successful redemption of your coupon, check the details in the Organization panel, Subscription panel, or **Subscription** > **My Plan.**

Subscription coupons

Subscription coupons and your plan

Using a coupon for a subscription plan places you on that plan if you have a saved payment method.

To return to the plan you had before the coupon, change your subscription before your coupon expires.

If you do not have a valid payment method saved, you will have a Free plan when your coupon expires.

Subscription coupons and your billing cyle

After you redeem your subscription coupon, your billing cycle changes. The day you redeem your coupon becomes the new start date of your billing cycle.

Example:

You have a Core plan that renews on the 20th of each month. You receive a coupon for the Teams plan. On the 15th you redeem your coupon. The 15th is now the beginning of your monthly billing cycle.

Exceptions:

- Coupons for the same plan do not impact your billing cycle. For example, a discount on your current plan.
- Core plans with an annual subscription have different billing and reset cycles. See **the note on Core plans in our information regarding changing your subscription**.

> You can only have one subscription coupon applied to your subscription plan at a time. Redeeming a second plan coupon cancels the first. This limit does not apply to coupons for extra operations. You can combine coupons for plans and extra operations.

Extra operations coupons

You do not need a saved payment method to use coupons for extra operations.

Extra operations are valid for the length of your billing cycle and expire on your billing day.

Example:

You have a coupon for 10,000 extra operations and your monthly subscription renews on the 15th of the month. If you redeem the extra operations on the 14th, they are valid until the 15th.

> Extra operations are valid until the end of your billing cycle regardless of how many other coupons you redeem.

6.2.7. Upgrade to the Enterprise plan

With the Enterprise plan, you get access to the most advanced Make features, wider limits, priority support, and a separate Celonis Amazon Web Service environment.

After signing up for Enterprise, you will need to migrate your data to our dedicated server in your region. This will give you access to a more stable and reliable infrastructure. Make provides an effective tool to let you migrate your data fast and easily.

In this article, you will learn how to use the Make **Migration tool** to move your data and start enjoying the Enterprise plan.

Migration scope overview

The Migration tool has two modes:

- Fast mode enables automatic migration of selected scenarios and their associated data. In some cases, the system may require you to take manual steps.
- Manual mode allows you to choose which data to migrate and how to handle it in your Enterprise instance.

Which data can you migrate?

The Migration tool migrates the following items:

- Scenarios and scenario folders
- Connections (except those that are associated with **connected systems and on-prem agents**
- Webhooks and mailhooks
- Keys
- Data stores
- Data structures
- Custom apps

If the following items exist in your source instance, but no scenarios use them, the Migration tool **doesn't** move them to the target instance:

- Connections
- Webhooks and mailhooks
- Keys
- Data stores
- Data structures

Which data can't you migrate?

- User accounts
- Organizations
- Scenario executions history and logs
- Incomplete executions
- **Connections associated with connected systems and on-prem agents**
- Webhook and mailhook queues and logs
- **Devices**
- **Payment details and extra operations**

Prepare your data

The Migration tool has some peculiarities that you need to keep in mind when migrating your data either in fast or manual mode. To arrange your target instance better, consider taking some measures that allow you to overcome limits of the Migration tool.

Teams

Keep in mind that: Migration involves transferring data from one team in the source instance to another team in the target instance.

To have your data in different teams in the target instance:

1. In the target instance, create as many teams as needed.
2. In the Migration tool, select the team you want to migrate scenarios to, and then select the specific scenarios to be moved to the selected team.

> You can't move scenarios from one team to another after finishing migration. When migration is complete, you can clone scenarios and select the desired team for them.
>
> To avoid data inconsistency, we recommend not migrating data from the same team more than once.

Time zones

Keep in mind that: The Migration tool doesn't migrate the time zone settings. You need to set up time zones in the target instance.

If you aim to run scenarios in the target instance according to the time zone that you have in the source instance, manually set up a time zone in the target instance. Learn more about **managing time zones**.

Scenarios

Keep in mind that: If your scenarios contain apps that require you to whitelist IP addresses for access, you need them to have relevant IPs associated with the target instance.

Before migrating such scenarios, **add the following IPS to apps** depending on your zone: **eu1.make.celonis.com** or **us1.make.celonis.com**. You may delete those IPs from apps in the source instance after the migration is done.

Keep in mind that: The Migration tool doesn't migrate a webhook or mailhook queue. It can cause problems if you use **sequential processing** because scenario data may arrive to the target instance before the queue is processed in the source instance.

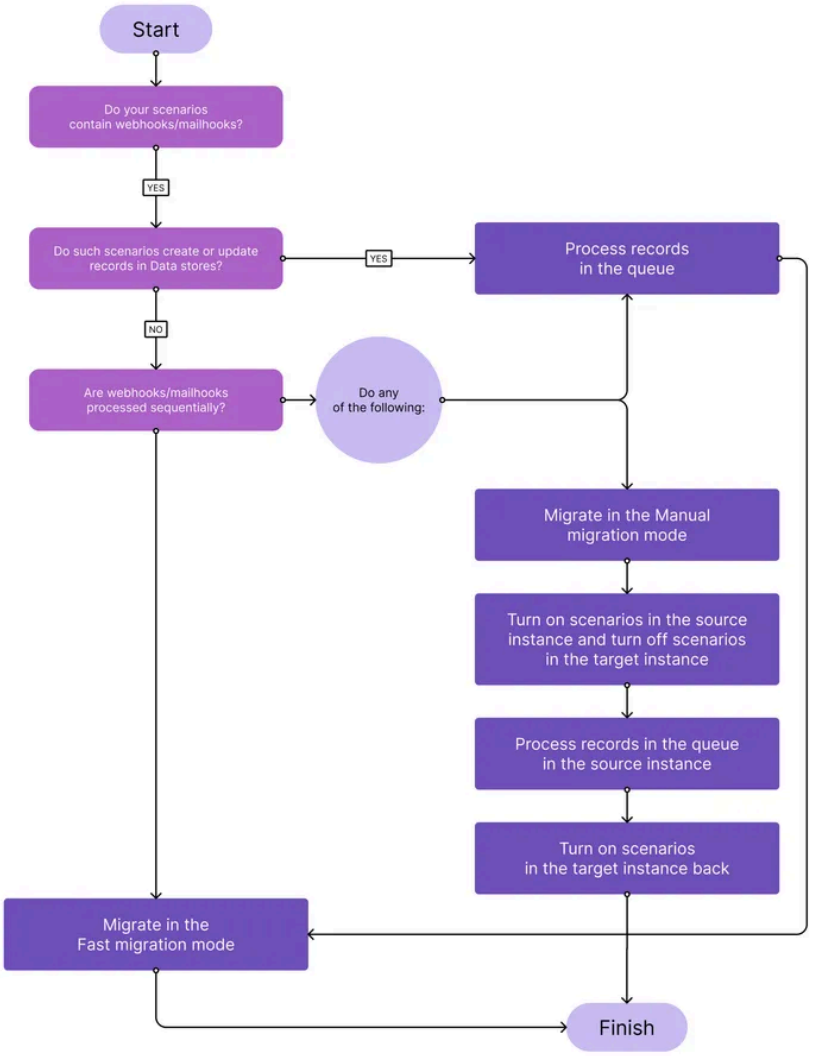
To ensure sequential processing, do one of the options:

Option 1: Process the entire queue in the source instance before migrating.

Option 2:

1. Migrate scenarios in the manual mode.
2. Disable scenarios in the target instance and enable scenarios in the source instance.
3. Process the webhook queue.
4. Once all the data is processed, enable scenarios in the target instance.

> Data received after migration goes to the new webhook URL and will be processed once you enable scenarios in the target instance.



Connections

Keep in mind that: The Migration tool doesn't replace one connection with another if they have different scopes and permissions.

Ensure that scopes and permissions of connections in the target instance match scopes and permissions of connections that you want to replace.

Data stores

Keep in mind that: Data may be inconsistent between the source and target instances if the Migration tool moves scenarios that update data stores more frequently than every five minutes.

To avoid inconsistency:

1. Stop all scenarios in the source instance before starting migration.

   > You only have to stop scenarios that contain the **Add/replace a record**, **Delete all records**, **Delete a record**, and **Update a record** modules.

2. Migrate the scenarios.

3. Run the scenarios in the target instance.

Custom apps

Keep in mind that: You cannot migrate scenarios with published custom apps if you didn't install those apps into the target instance beforehand.

To have scenarios with custom apps in the target instance:

1. If the custom app has the **Private** status, publish it. Refer to the **App Builder documentation** to learn more.

   > It is not possible to migrate apps that have the **Private** status. Publish the app after making all the necessary changes.
   >
   > When the custom app has the **Published** status, you cannot:
   >
   > - Make the custom app Private again
   > - Delete the custom app
   > - Delete a module from the custom app

2. Install the published custom app in the target instance.

3. Delete a module from the custom app.

Migrate your data

You can migrate as many scenarios as you want at once. If you don't migrate all scenarios at once, you can continue later.

Follow the next steps to start migration:

1. Go to the **Migration tool**.
2. Read all the information and click **Let's start**.
3. In the **Source** and **Target** sections, select Make.
4. In both sections, click **Sign in to source** and **Sign in to target** respectively, and log in to both Make instances. Then click **Continue.**
5. In both **Organization** dropdown lists, select the source organization to migrate data from, and the target organization to migrate the data into.
6. In both **Team** dropdown lists, select the source team to migrate data from, and the target team to migrate data into. Then click **Continue**.

You see the list of all scenarios that your source instance has. You can select the migration mode: fast or manual.

You see the list of all scenarios that your source instance has. You can select the migration mode: fast or manual.

> The Migration tool doesn't migrate sscenarios that have errors: you can find them in the **Non-migratable** scenarios tab. If you want to move those scenarios to the target instance, you must first resolve the errors.

Manage your data and finish the migration

Depending on the migration mode you chose, you need to decide how to handle your data from the source instance. Some options are available only for the manual mode.

We highly recommend using the fast migration mode to ensure smooth and easy data transferring.

1. Select custom apps to replace ones that migrated scenarios use.

2. Select the action for connections:

    - Duplicate a connection in the target instance
    - Replace a connection with the one that exists in the target instance
    - Use a previously migrated connection

> The set of scopes or permissions of a particular connection might differ between the source and the target instance. To see the scopes of a specific connection, go to the **Connection** section and click the eye icon.
>
> If you need to extend scopes:
>
> 1. Log in to either a source or a target instance depending on where you need to extend scopes.
> 2. Create a new scenario or open an existing one.
> 3. Add a module that requires additional scopes.
> 4. Create a new connection. Make will ask you to provide additional permissions. Click **Continue**.
> 5. On the app page where you're redirected to, confirm granting new scopes.
> 6. Optional: Delete the module or the newly created scenario if you don't need it.
>
> Some apps allow adding scopes manually when creating a new connection. If you created a custom application in a service itself, you may extend scopes in your service account.

3 Select the action for webhooks and mailhooks:

- Duplicate a webhook or mailhook
- Duplicate a webhook or mailhook and forward traffic

> We recommend always choosing the second option. If you just duplicate a webhook or mailhook, traffic will still be sent on the webhook or mailhook that you have in the source instance, and no data will appear in the target instance.
>
> A duplicated webhook will have a different URL.

4 If you have linked devices in your source instance, create a mock device in the target instance.

> You need to replace a mock device with a real device. See more information in the **devices section** below.

5 Select the action for data stores:

- Duplicate a data store and migrate its content
- Duplicate a data store without its content
- Replace a data store with the one that exists in the target instance

> The Migration tool will replace data in the data store in the target instance with data from the source instance. If you use auto-generated keys, Make will store data under different keys in the target instance.

6 Select the action for data structures:

- Duplicate a data structure
- Replace a data structure with the one that exists in the target instance

7 Select the action for keys:

- Duplicate a key
  A duplicated key contains the relevant data, but the key ID is different.
- Replace a key with the one that exists in the target instance

8 Select the action for folders:

- Duplicate a folder
- Replace a folder with the one that exists in the target instance

9 Select what you want to do with the scenarios both in the source and target instances.

> If at least one scenario contains webhooks or mailhooks, Make automatically disables all migrated scenarios. You may enable them manually after migration.

10 Click **Start migration**.

The Migration tool shows the list of scenarios and entities it migrated and confirms that the migration process was successful. Click **Show log** to see the migration log.

If the migration fails, the Migration tool rolls back all changes made both to the source and target instances. Check the migration log out to find out reasons for the failure.

Post-migration data setup

Using the Migration tool to move your data to a different server environment ensures smooth and easy migration. Still, you must consider specific details and take certain actions after migration to fully benefit from the Enterprise plan.

Your data will remain in your source instance after migration.

Subscriptions and coupons

Make doesn't automatically cancel your source instance subscription. After you fully set up your target instance, you need to manually cancel your subscription. After that, your source instance moves to **the Free pricing plan**. You can use all available features within this plan.

Extra operations remain in the source instance. Contact our Support to move extra operations to the target instance.

Scenarios

After migrating scenarios scheduled as **immediately**, they execute once the scenarios trigger receives new data.

Some scenarios require additional configuration after migration. Pay attention to scenarios that contain:

- Custom apps: They may require additional setup.
- Apps that require IP whitelisting: Make sure that the migrated scenarios have **updated IPs** in their apps.
- The Makeapp: Create a new connection and update all IDs mapped in the app modules. New IDs should be associated with the data based on the target instance.

On-prem connections

The Migration tool doesn't migrate connections associated with **connected systems and on-prem agents**. You need to manually create connections in the target instance and then link them to connected systems.

Webhooks and mailhooks

If during migration you selected duplicating webhooks or mailhooks and forwarding traffic, the old URL in the source instance still receives traffic, but redirects it to the new URL in the target instance. Don't delete the old URL in order not to lose traffic. Contact our Support to receive a list of pairs of old and new URLs.

Devices

The Migration tool creates mock devices in the target instance to migrate scenarios that use the iOS or Android app.

The mock device is called a deaf device in the target instance.

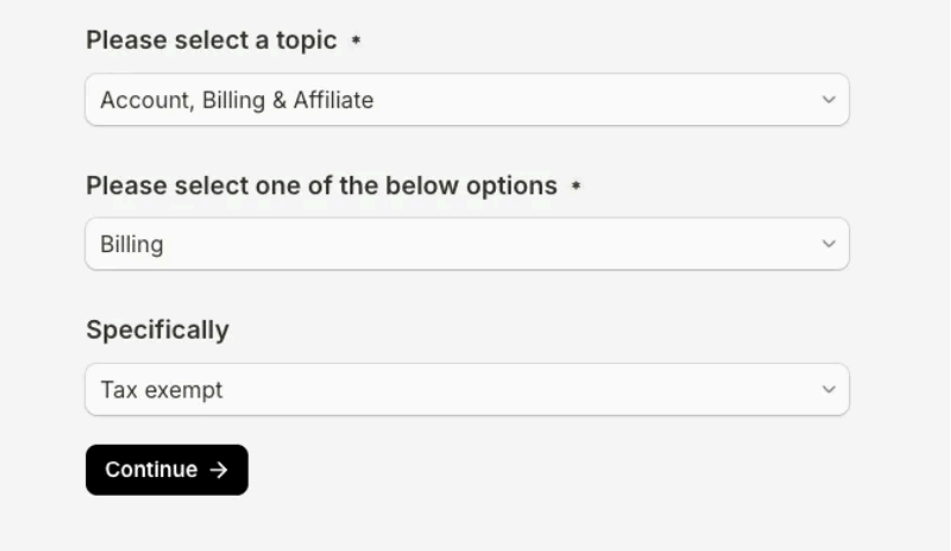
After migration, to use such scenarios, you must:

1. Create a new device manually in the target instance.
2. Replace a mock device with the newly created in all relevant scenarios.
3. Delete all mock devices.

6.2.8. Request sales tax exemption

If you are an organization exempt from sales tax, **contact us** using the following details:

- **Topic:** Account, Billing & Affiliate
- **Category:** Billing
- **Topic details:** Tax exempt



Once you reach out, we'll send you instructions on submitting your tax-exempt documentation. After we review and verify your documentation, you'll receive an email confirming your tax-exempt status has been approved.

6.3. Administration

Discover features to manage organization usage and members:

→ **Analytics dashboard**

→ **Operations per team management**

→ **Delete user data**

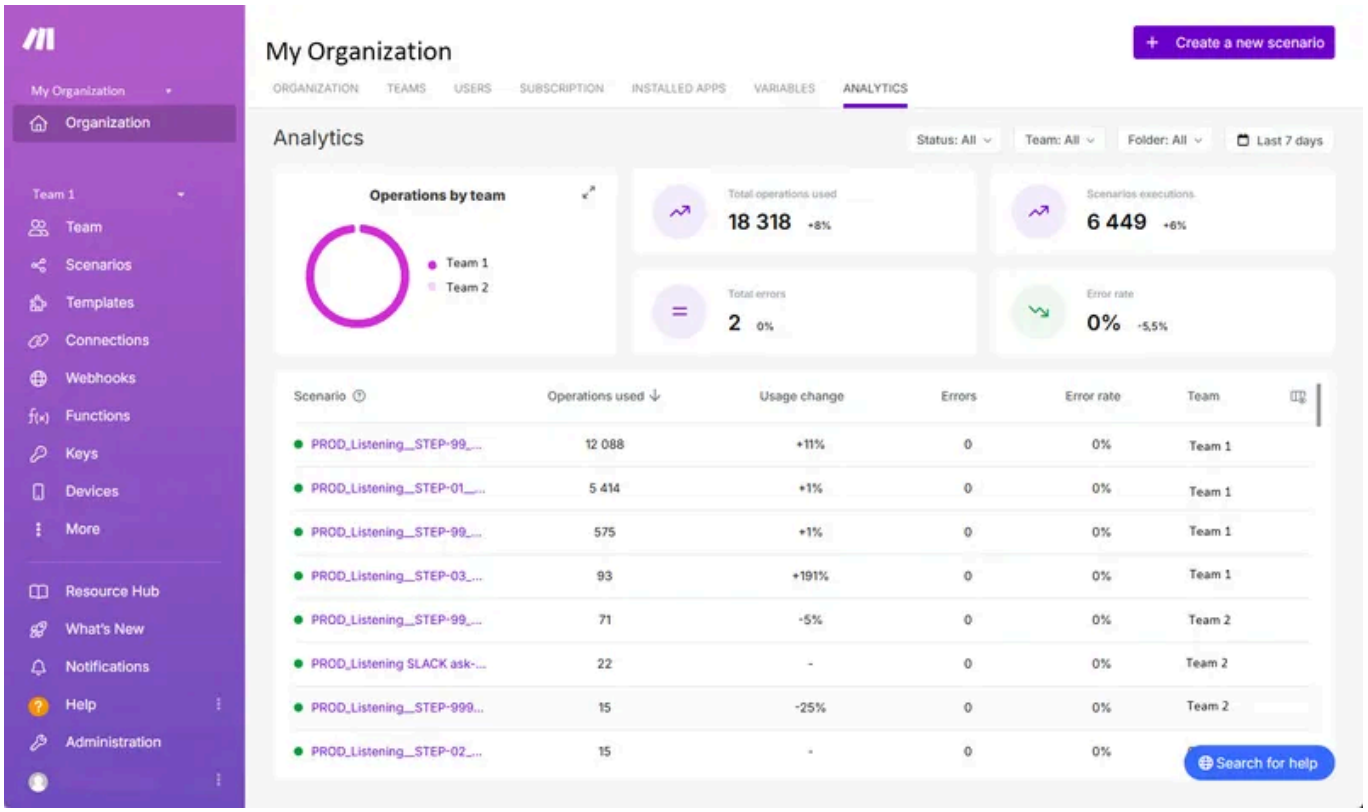
→ **Set the length of your session timeout**

→ **Audit logs**

6.3.1. Analytics dashboard

> This feature is available to Enterprise customers.

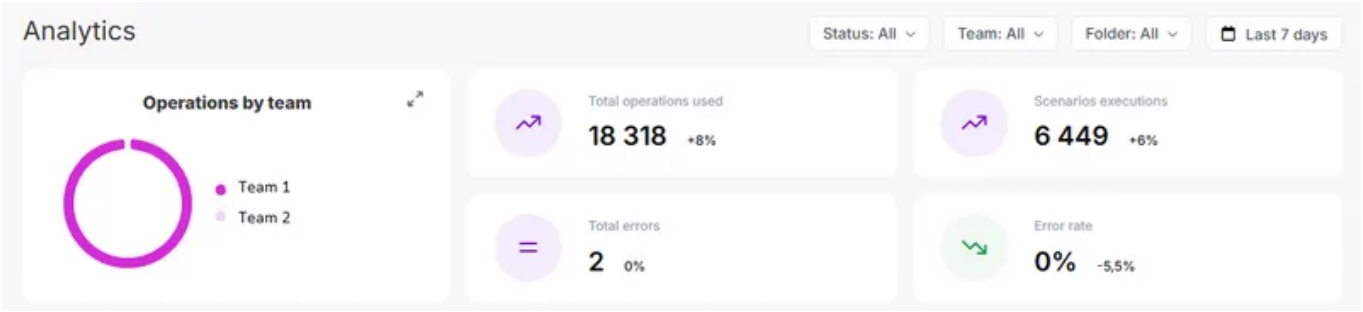
The Analytics dashboard gives you a clear view of your account's key metrics. You can easily see what's happening across all your scenarios so you can stay on top of things and make informed decisions to optimize performance.



Dashboard features

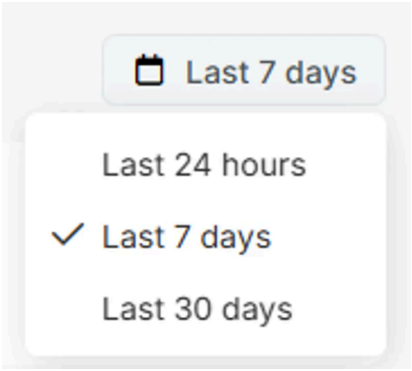
The top of the dashboard shows key metrics for your organization:

- operations by team
- total operations used
- total errors
- error rate
- scenario executions

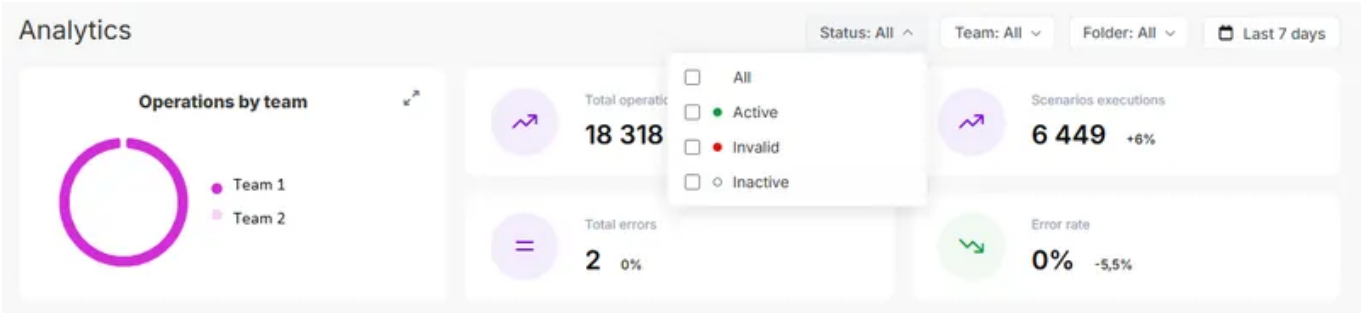


Next to each of these metrics is an icon, showing how the metric is trending. For example, a down arrow next to the error rate indicates a decreasing error rate within the indicated time frame.

You can change the time frame to view statistics for the last 24 hours, 7 days, or 30 days, allowing for more precise analytics.



Additional filters include **Status**, **Team**, and **Folder**. The values available under **Team** and **Folder** vary, depending on the setup of your organization.



By default, scenarios are listed according to the number of operations used in the selected time frame.

You can further customize the display by choosing the columns you want to view for each scenario.



Column	Description
Scenario	Only existing scenarios are shown. Data for deleted scenarios is unavailable. • Green dot: Active scenario • Red dot: Invalid scenario • Empty dot: Inactive scenario
Operations used	The total number of operations used by a scenario in the selected time frame.
Usage change	The percentage change in operations used by a scenario, compared to the previous time frame.
Executions	The total number of executions in a scenario in the selected time frame.
Executions change	The percentage change in executions for a scenario, compared to the previous time frame.
Errors	The total number of errors in a scenario in the selected time frame.
Error rate	The percentage of executions with errors in a scenario in the selected time frame.
Team	The team associated with a scenario.
Folder	The folder associated with a scenario.

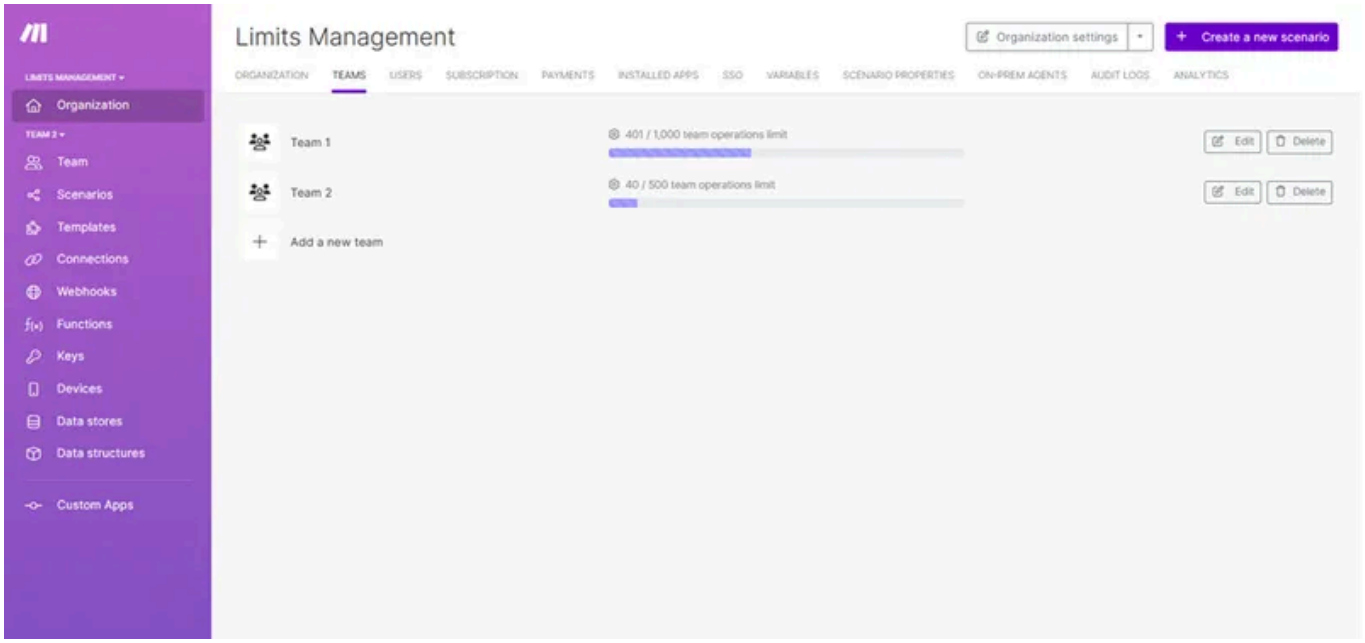
6.3.2. Operations per team management

> This feature is available to Enterprise customers.

Organization owners and administrators have the option to set operations consumption limits for teams. This feature offers many benefits:

- **Control over teams consumption**
  You can set operations limits for all teams within an organization.
- **Operational continuity**
  Operations limits remove the possibility of stopping organization scenarios due to a team's overconsumption.
- **Real-time operations usage monitoring**
  The team dashboard shows individual team consumption as well as total organizational usage in real time.
- **Flexibility in operations allocation**
  You can change allocated limits as needed.
- **Proactive usage notifications**
  You will receive an email at 75%, 90%, and 100% usage thresholds to allow for proactive operations management.

You can view the breakdown of operations usage by teams by clicking the **Teams** tab.



> Team operations consumption includes usage from mid-December 2024 to the present. Usage before this date is not included.

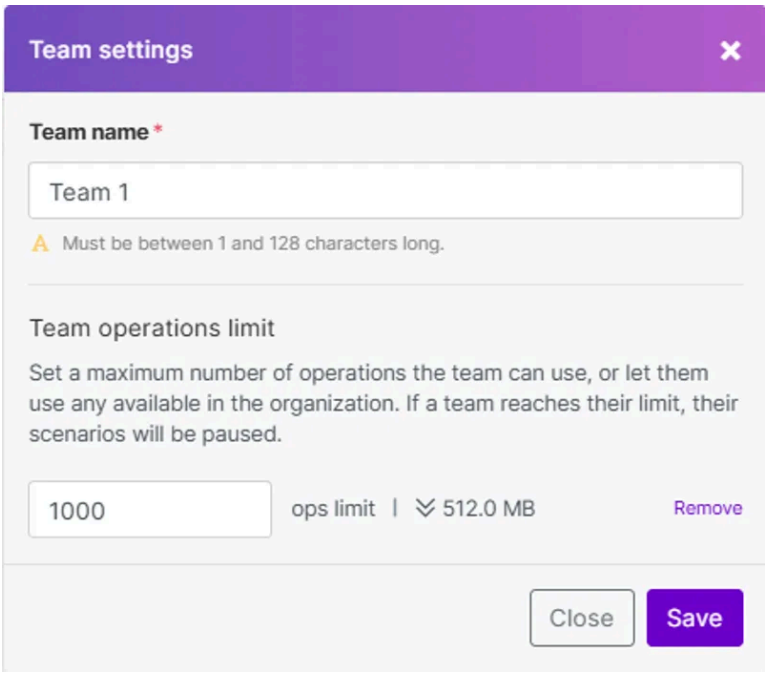
Set the team operations limit

1. Click **Organization > Teams** to view all of the teams in your organization.

2. Click **Edit** for the team for which you would like to set an operations limit.

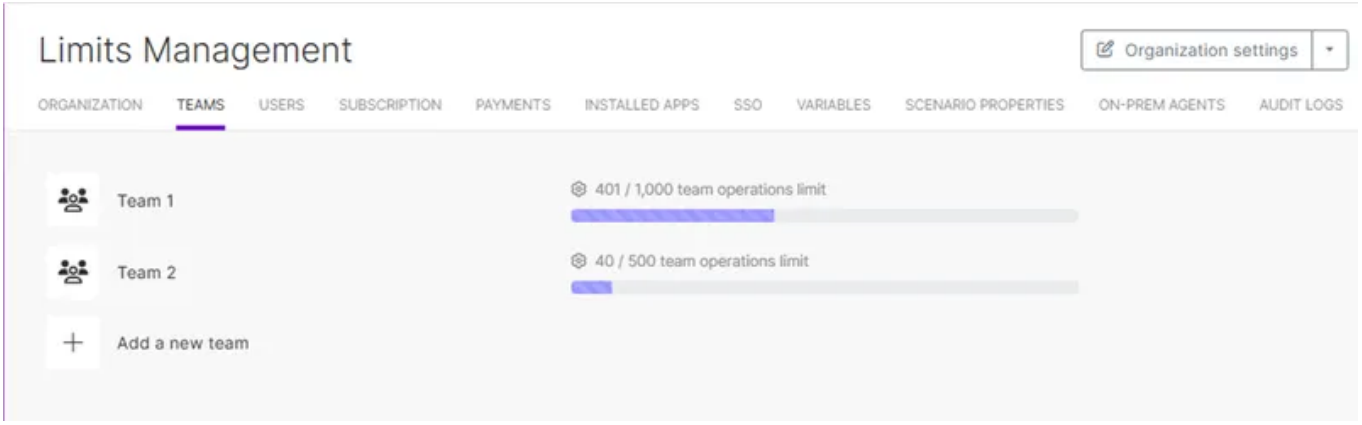
3. Click **Set team limit** and enter the maximum number of operations you would like to allocate for this team.

   If the team reaches their limit, their scenarios will be paused.



4. Click **Save**.

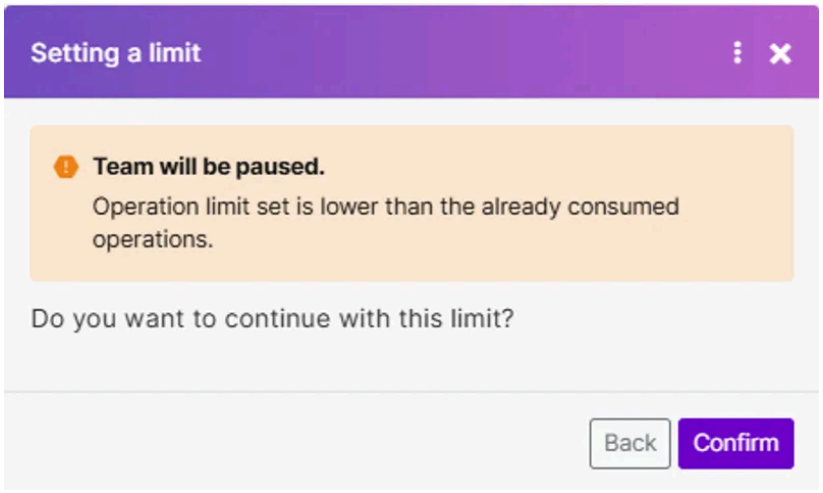
You will see the team operations limit and usage under the **Teams** tab.

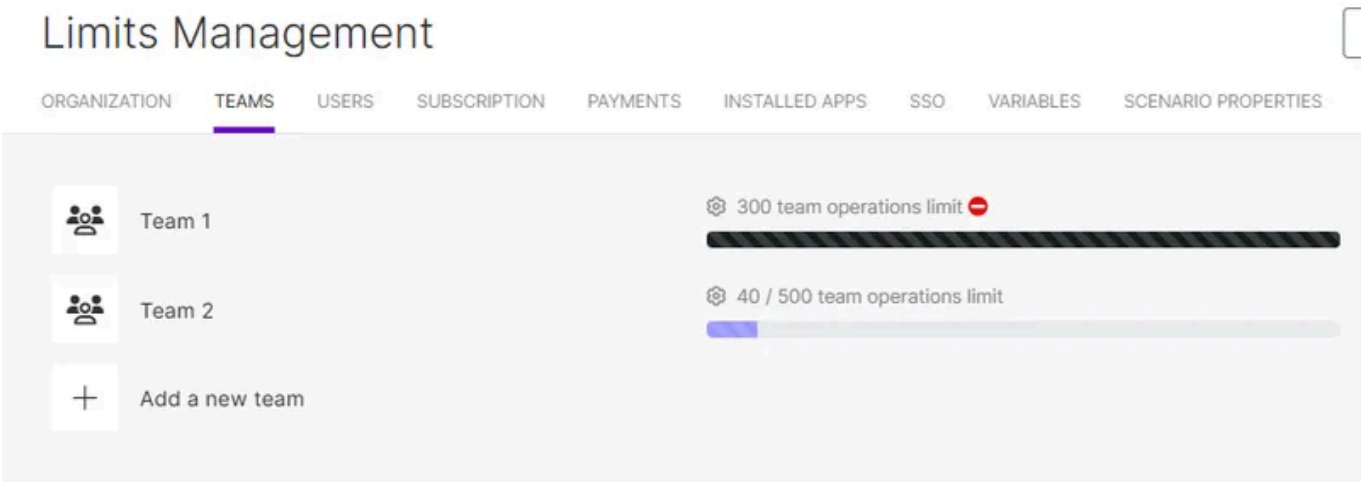


Change the team operations limit

1. Click **Organization > Teams** to view all of the teams in your organization.
2. Click **Edit** for the team for which you would like to change the operations limit.
3. Update the team operations limit value to the new maximum number of operations the team can use.
4. Click **Save**.

If you change the limit to an amount less than what has already been consumed by the team, the team will be paused.

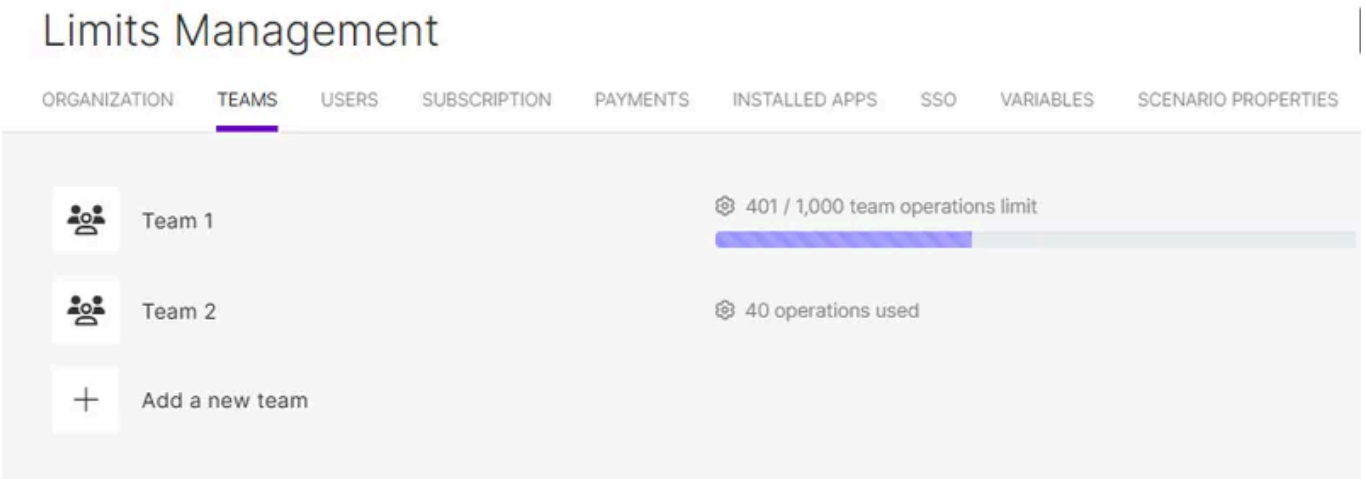




Remove the team operations limit

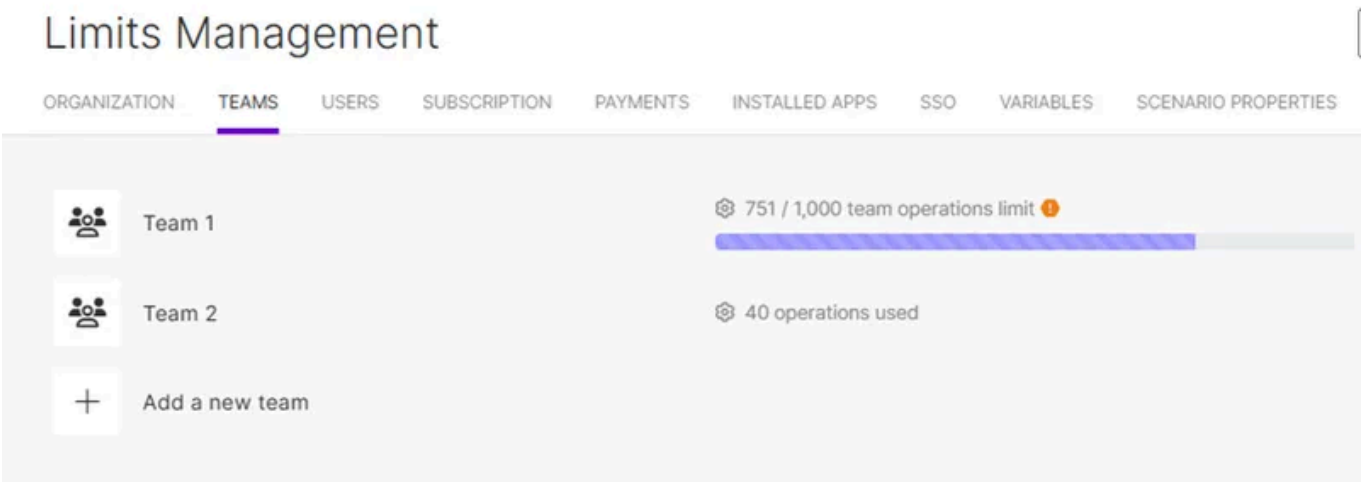
1. Click **Organization > Teams** to view all of the teams in your organization.
2. Click **Edit** for the team for which you would like to change the operations limit.
3. Click **Remove** to remove the team operations limit.
4. Click **Save**.

For a team without an operations limit, you will only see the total operations used. In this example, Team 1 has an operations limit of 1000, and Team 2 has no operations limit set.

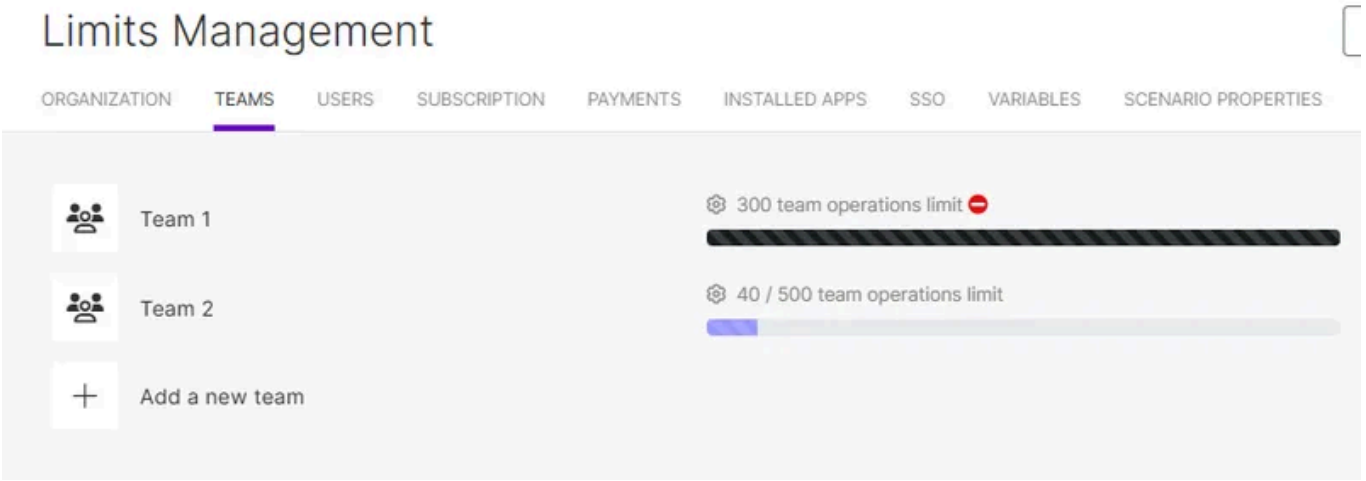


Usage notification

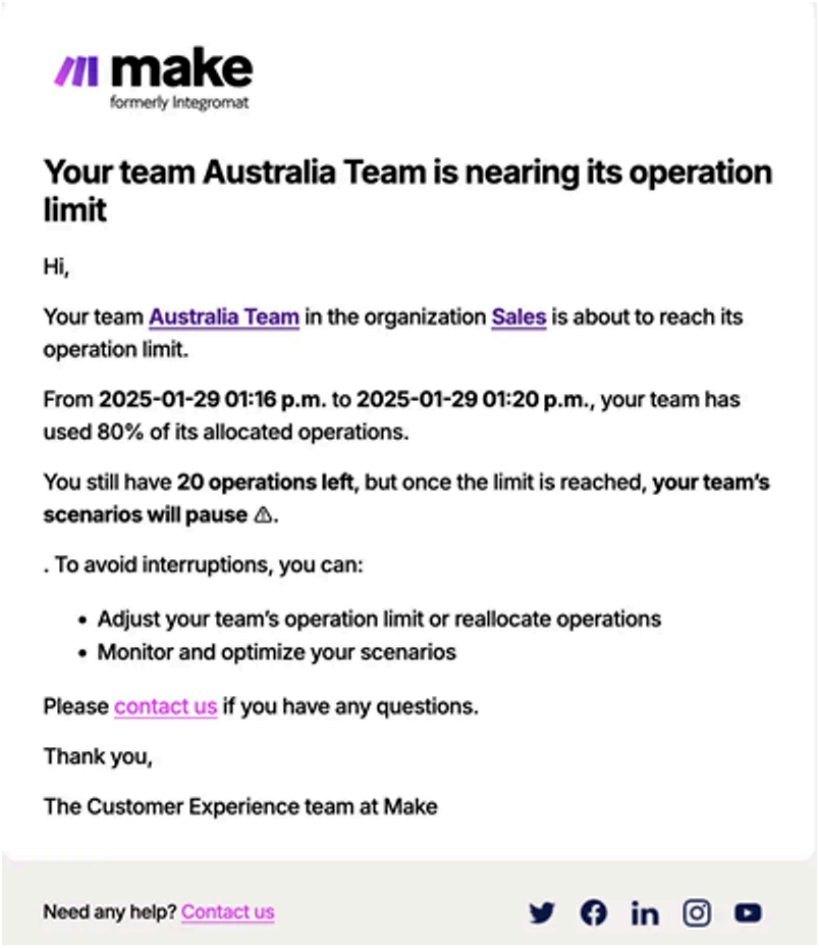
As a team approaches their limit, beginning with 75% usage, you will see an orange notification icon next to the team's operations limit.



When a team reaches their limit, you will see a red notification icon next to the team's operation limit, the bar will turn black, and all scenarios will pause.



When a team in your organization has reached 75%, 90%, and 100% of the allocated operations, organization owners and admins will receive an email detailing this information.



To avoid interruptions, the organization owner or admin can:

- adjust the team's operations limit or reallocate operations.
- monitor and optimize the team's scenarios.

Team admins will also receive an email with this information.

make
formerly Integromat

Your team Australia Team is nearing its operation limit

Hi,

Your team Australia Team in the organization Sales is about to reach its operation limit.

From **2025-01-29 01:16 p.m.** to **2025-01-29 01:41 p.m.**, your team has used 80% of its allocated operations.

You still have **20 operations left**, but once the limit is reached, **your team's scenarios will pause** ⚠.

To avoid interruptions, please reach out to your organization owner or admin to adjust the operation limits or reallocate operations.

Please contact us if you have any questions.

Thank you,

The Customer Experience team at Make

6.3.3. Delete user data

To completely delete all user data that is stored on Make servers, please follow the instructions below.

Scenarios

1. Click **Scenarios** in the left sidebar.
2. Click the **All** tab to see all scenarios.
3. To the right of a scenario, click the down arrow and select **Delete**.
4. Click **Really?** to confirm.
5. Repeat for each scenario.

Connections

1. Click **Connections** in the left sidebar.
2. To the right of a connection, click **Delete**.
3. Click **Really?** to confirm.
4. Repeat for each connection.

Webhooks

1. Click **Webhooks** in the left sidebar.
2. To the right of a webhook, click **Delete**.
3. Click **Really?** to confirm.
4. Repeat for each webhook.

Keys

1. Click **Keys** in the left sidebar.
2. To the right of a key, click **Delete**.
3. Click **Really?** to confirm.
4. Repeat for each key.

Devices

1. Click **Devices** in the left sidebar.
2. To the right of a device, click **Delete**.
3. Click **Really?** to confirm.
4. Repeat for each key.

Data stores

1. Click **Data Stores** in the left sidebar.
2. To the right of a data store, click **Delete**.
3. Click **Really?** to confirm.
4. Repeat for each data store.

User profile

1. Click your name in the bottom-left corner, then click **Profile**.
2. In the upper-right corner, under **Profile settings**, select **Delete profile**.
3. Enter your password to confirm and click **Delete**.

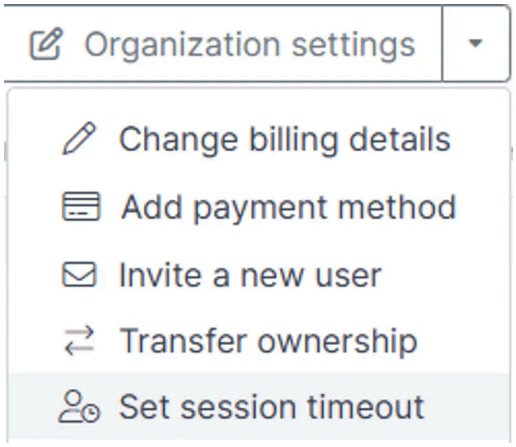
6.3.4. Set the length of your session timeout

For security purposes, you may wish to control the length of the session timeout.

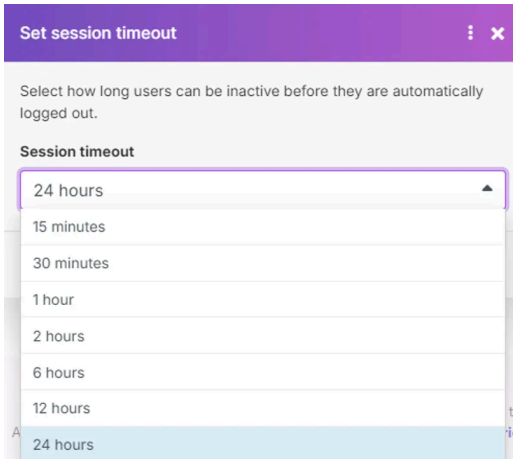
You can choose to set your session timeout from 15 minutes to 24 hours. **This option is only available to Organization administrators and owners.**

1. Click **Organization** in the left sidebar.

2. In the upper-right corner, under **Organization settings**, select **Set session timeout**.



3. Select how long users can be inactive before they are automatically logged out and click **Save**.



Your session timeout is saved.

If a user belongs to multiple organizations, the shortest session logout time applies.

6.3.5. Audit logs

> This feature is available to Enterprise customers.

Audit logs allow you to monitor user activity within your organization or team. This information is particularly useful for resolving issues and troubleshooting incidents. For example, if a crucial scenario transferring data between systems unexpectedly fails, audit logs can help you identify who updated the scenario and when it happened.

Audit logs are available on the Enterprise plan only. Organization owners and admins as well as team admins can access audit logs.

Audit logs are stored for 12 months.

Organization audit logs

Audit logs are available at both the organization and team levels. Organization audit logs give you visibility into everything that is happening in your organization, including such events as creating or updating the organization variable.

Organization audit logs are visible only to organization owners and admins.

To open audit logs from the Organization dashboard:

1. Click **Organization** in the left sidebar.

2. Click the **Audit Logs** tab.

   Once you open the tab you will see the audit logs for the organization you're in. You can filter what you see to show only specific events, specific time periods, or users by setting up filters on the filter panel.

   Additionally, you can select to see only events from specific teams in the audit logs.

3. Click the **All filters** button.

4. Select the teams for which you want to see the logs in the **Team** dropdown.

5. Click **Apply**.

Team audit logs

Team audit logs are available to team admins. Organization owners and admins can see team audit logs through the audit logs in the Organization dashboard or by opening the audit logs for a specific team from the Team dashboard.

To view audit logs from the Team dashboard:

1. Click **Team** in the left sidebar.

2. Click the **Audit Logs** tab.

You will see the audit logs for your team. You can filter the logs to display specific events or time periods.

Available events

The audit logs allow you to see what was changed, who made the change, when it was made, and for which scenario.

> Some of the events are not visible in the team audit logs. For example, you won't see events about organization variables, but you will see events connected to this specific team (team member added, removed, etc.).

Event category	Available events	Available for Organization	Available for Team
Scenarios	• Scenario created • Scenario updated • Scenario deleted • Scenario activated • Scenario deactivated	Yes	Yes
Connections	• Connection created • Connection deleted • Connection updated • Connection authorized/reauthorized	Yes	Yes
Webhooks	• Webhook created • Webhook deleted • Webhook updated • Webhook enabled • Webhook disabled	Yes	Yes
Keys	• Key created • Key updated • Key deleted	Yes	Yes
Team	• Team created • Team updated • Team deleted • Team role updated • Team member removed	Yes	Yes
Team Variables	• Team variable created	Yes	Yes

Event category	Available events	Available for Organization	Available for Team
	• Team variable updated • Team variable deleted		
Organization Variables	• Organization variable created • Organization variable updated • Organization variable deleted	Yes	No
Data Store	• Data store created • Data store updated • Data store deleted • Data store record created • Data store record updated • Data store record deleted	Yes	Yes
Data Store Records	• Data store record created • Data store record updated • Data store record deleted	Yes	Yes
Data Structure	• Data structure created • Data structure updated • Data structure deleted	Yes	Yes
Functions	• Function updated	Yes	Yes

Event category	Available events	Available for Organization	Available for Team
	• Function created • Function deleted		
Organization	• Organization created	Yes	No

> Events related to Data Store Records (like Data Store Record Updated) only apply to actions done within the Make interface or API. They don't apply to actions done through the datastore modules.

Event details

The audit log main page provides a summary of who performed specific actions and when they occurred. For more detailed information about an event, click the **Details** button next to the entry. A pop-up window will appear, displaying additional data about the event. To copy any information, click the **Copy** button within the pop-up.

6.4. Access management

Set up SSO and domain claim to define Enterprise user log in:

→ **Single Sign-on**

→ **Google SAML**

→ **MS Azure AD SAML**

→ **MS Azure AD OIDC**

→ **Okta SAML**

→ **SAML certificate management**

→ **Domain claim**

6.4.1. Single Sign-on

> This feature is available to Enterprise customers.

Single sign-on (SSO) allows you to use your own provider of user account management, authentication, and authorization services to register and log in to Make .

Make supports the following protocols:

- Open ID Connect (OIDC)
- SAML 2

Make supports the following identity providers (IdPs):

- Okta (**SAML**)
- Microsoft AD (**SAML** **OIDC**)
- Google (**SAML**)

You configure SSO for each of your organizations separately.

You can prevent your organization members from accidentally creating their own self-service accounts by **claiming a domain**. After you set up SSO, claim your email domain so Make can recognize your new users. Any new user who signs in with your claimed email domain gets a prompt to use SSO.

Enable single sign-on using Open ID Connect (OIDC) and SAML 2.0

> Double-check your SSO configuration before you click **Save** on the SSO settings page. When you click **Save**, Make enables SSO with the settings you provided. You will be logged out immediately. You won't be able to log in with your Make credentials anymore.

1. Click **Organization** in the left sidebar.

2. Click the **SSO** tab.

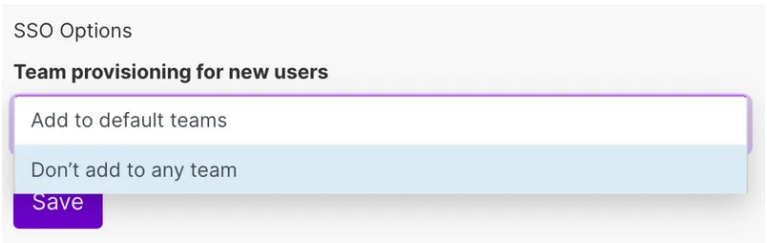
3. Click **SSO configuration**.

4. Enter a **Namespace**. You can enter any text that describes your organization. Users will need to enter your organization's namespace on the SSO login page. Namespace must include only lowercase characters and dashes. An underscore may lead to errors.

5. Select an **SSO type**.

6. Fill in the protocol-specific information as described in either the **Open ID Connect** or **SAML** section of this article.

7. Under **Team provisioning for new user**, select which teams new users who log in will become members of. You can choose to not add new users to any team.



8. Click **Save**.

Make enables SSO with the settings you provided and logs you out immediately. You can now log in with your SSO provider credentials. At the same time, you receive an email with a one-time link, which you can click to disable SSO.

When logging in using SSO for the first time, you must use an account that is the owner of the organization and has the same email address as the account that you used to configure SSO. Make sure that you assign the same email address to the user in your identity provider.

Open ID Connect (OAuth 2.0 settings)

The following fields appear once you select OAuth 2.0 from the SSO menu:

Field	Required	Description
User Information URL	required	URL obtained from your identity provider. Example: `https://example.com/oauth2/v1/userinfo`
Client ID	required	Obtained from your identity provider. Sometimes called Application ID.
Token URL	required	URL obtained from your identity provider. Example: `https://example.com/oauth2/v1/token`
Login scopes	optional	Parameters used when accessing your identity provider.
Scopes separator	optional	The character used between scopes, such as a space or a comma. If your separator is a space, use the spacebar on your keyboard.
Authorize URL	required	URL obtained from your identity provider. Example: `https://example.com/oauth2/v1/authorize`
Client secret	required	Obtained from your identity provider.
IML resolve	required	Because both Make and your Identity provider use attributes such as username and email, you need to **map these attributes using IML**. For Open ID Connect: `{"id":"{{sub}}","email":"{{email}}","name":"{{name}}"}`
Redirect URL	optional	The location where the identity provider sends the user once successfully authorized and granted access. Must be unique to your application/instance.

SAML 2.0 settings

The following fields appear once you select SAML 2.0 from the SSO menu:

Field	Required	Description
Service provider certificate	required	Make provides certificates for you. Click the down arrow to copy or download a `.pem` file of your certificate. At least one certificate must be active. If there is no active certificate, click **Activate**. For more information, see our article on **SAML certificate management**.
Identity provider certificate	required	An x.509 certificate created and stored by your IdP, for example, Google, Okta, or Microsoft Azure Directory. You can enter this information in the following ways: • Copy and paste from your IdP's UI.

Field	Required	Description

		• Copy and paste from your IdP's metadata XML file.
		• Extract from any of the following:
		  ◦ P12
		  ◦ PFX
		  ◦ PEM

IdP login URL	required	Also called an authorization URL. The IdP login URL is available from your IdP, for example, Google, or Okta. The IdP metadata typically contains this information in XML. The IdP metadata is usually downloadable from your Identity provider.
IdP logout URL	optional	A URL created by your IdP to enable Single Log Out (SLO). Leave this field empty to disable SLO.
Login IML resolve	required	Because both Make and your Identity provider use attributes such as username and email, you need to **map these attributes using IML**.
Redirect URL	optional	The location where the identity provider sends the user once successfully authorized and granted access. Must be unique to your application/instance.
Allow unencrypted assertions	optional	Your IdP may not support SAML 2.0 assertions with encryption. Check with your IdP to determine whether you need to enable this option.
Allow unsigned responses	optional	Your IdP may not support a signed SAML 2.0 response. Check with your IdP to determine whether you need to enable this option.
Sign requests	optional	Your IdP may require a signed SAML 2.0 response. Check with your IdP to determine whether you need to enable this option.
Audience	optional	Optional field to define the intended target. Typically this is a URL but can also be formatted as any string

Field	Required	Description
		of data.
Audience URI	optional	This read-only field provides you with the path for metadata.xml file. This information might be needed to set up the SAML settings on the SSO provider side.

SAML certificate rotation

For more information on rotating your service provider certificate, see our article on **SAML certificate management**.

Create and enter Login IML resolve

To support a broad choice of identity providers (IdPs), Make lets you map values related to identifying users. The IML resolve maps the values from your IdP to Make's internal values by using IML, a JavaScript-based function notation. Your IML resolve must be specific to your IdP. You must map the following properties:

Property	Description
email	You can map this to any valid email. Aliases and alternate email suffixes can create problems. Be sure to map the most appropriate email in your IML resolve.
name	Used as the user's name in the application. You can reuse email for this property. If left blank, creates a user without a name that must be updated later.
id	External user ID Can be `an integer` or `string` but must be mapped to an identifier.

In the following example, the resolve maps the following values:

Make	Your IdP
email	`user.attribute.email`
name	`user.attributes.firstName` and `user.attributes.last` concatenated together
id	`user.name_id`

Javascript

```
{
    "email":"{{get(user.attributes.email, 1)}}",
    "name":"{{get(user.attributes.firstName, 1)}} {{get(user.attributes.last}
    "id":"{{user.name_id}}"
}
```

Log in using SSO

When Make is configured to use SSO, users don't use the default sign-in form. Instead, they use the dedicated SSO sign-in options.

1. Go to **https://www.make.com/en/login**
2. Click **Sign in with SSO**.
3. Enter the namespace you chose for your organization.
4. Log in using your identity provider and consent to Make's access to your user data.

The user is now logged in. If the user was not assigned to your organization before, the system creates a new user account for them and assigns them to the selected default team.

> If a user with the same email address already existed in the organization before you configure SSO, they will not have access to the organization's data. To solve this, delete the user from the organization and ask them to log in again using SSO.

6.4.2. Google SAML

> This feature is available to Enterprise customers.

The following manual configuration creates an SAML SSO configuration for your Enterprise organization.

Prerequisites

- `Owner` role in an Enterprise organization
- Google Admin console account

Supported features

This configuration supports the following:

- Service Provider initiated SSO
- Single Log Out [optional]

Configuration Steps

Before configuring SSO, you need to assign a namespace and download your service provider certificate in Make. These steps provide information you need to enter later.

Create your namespace in Make

1. Click **Organization** in the left sidebar.
2. Click the **SSO** tab.
3. Under **Namespace**, enter the namespace you want for your organization. For example, `acmecorp`. Your organization members enter this namespace when they log in via SSO.

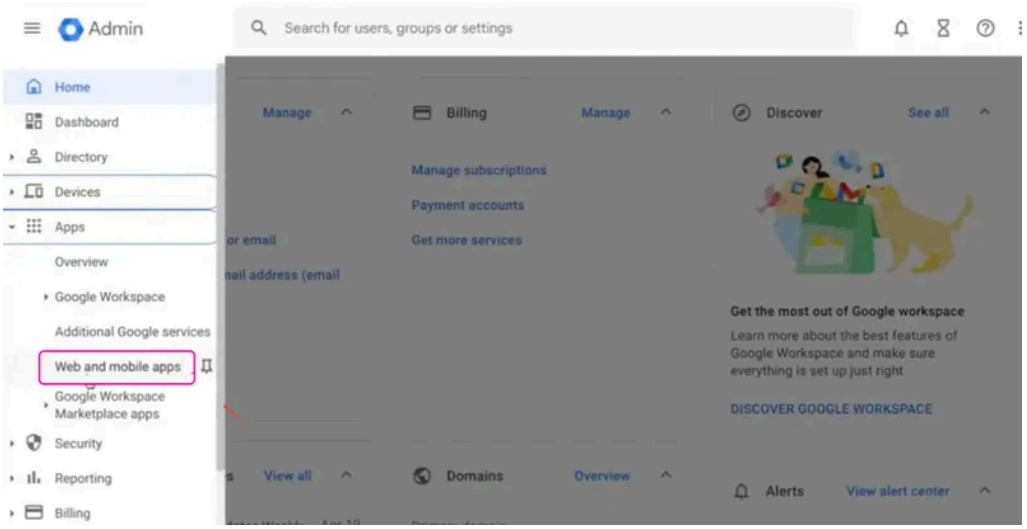
4 Under **SSO type**, select **SAML 2.0**.

5 Copy the **Redirect URL** and save it in a safe place. You will use this later when you create your SAML integration in the Google admin portal.

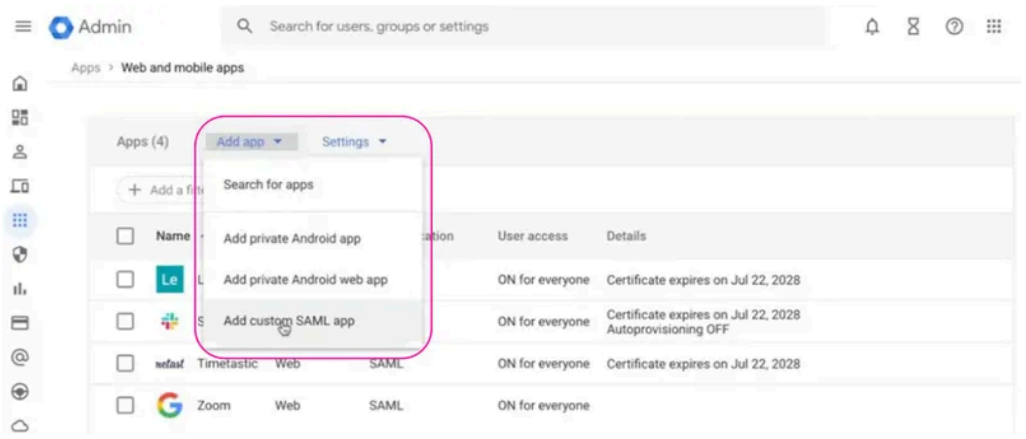
Create an SAML application in the Google admin portal

1 Login to the Google admin console.

2 From the dashboard's left menu, click **Apps > Web and mobile apps**.



3 Click **Add app** and select **Add custom SAML app**.



4 Enter an **App name** and **Description**.

5 Copy the **SSO URL** and save it in a safe place. You will use this later.

6 On the same screen, download the certificate and save it in a safe place.

7 Click **Continue**.

8 Enter the **Service provider details**. You can find thise information in the Make SSO configuration tab.

ACS URL: `https://www.make.com/sso/saml/{namespace}`

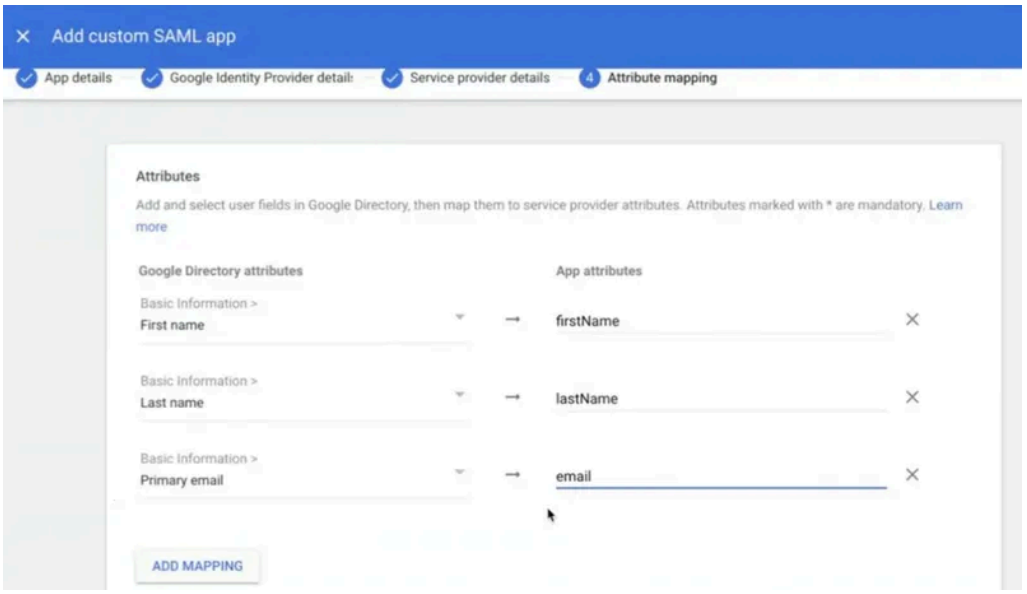
Entity ID: `https://www.make.com/sso/saml/{namespace}/metadata.xml`

Replace {namespace} with your **namespace**.

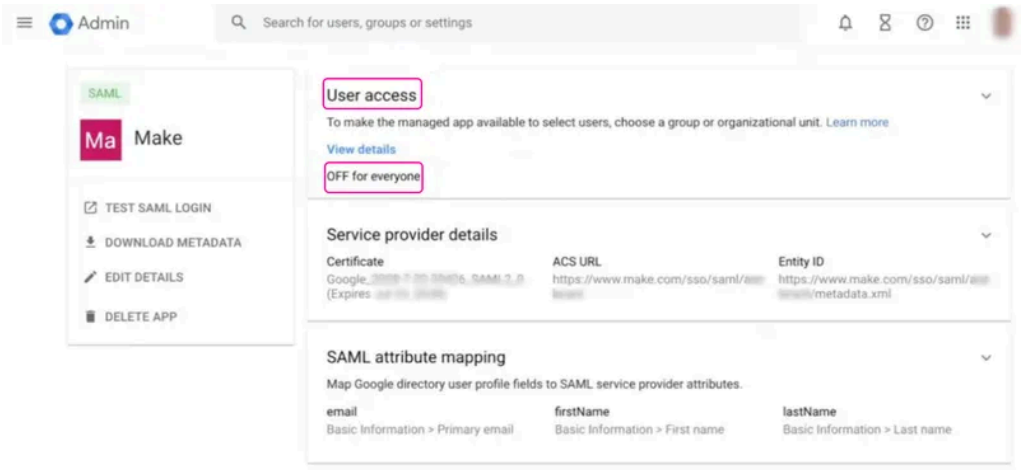


9 Click **Continue**.

10 Enter the App attributes.



11 Update the **User access** to On for everyone.



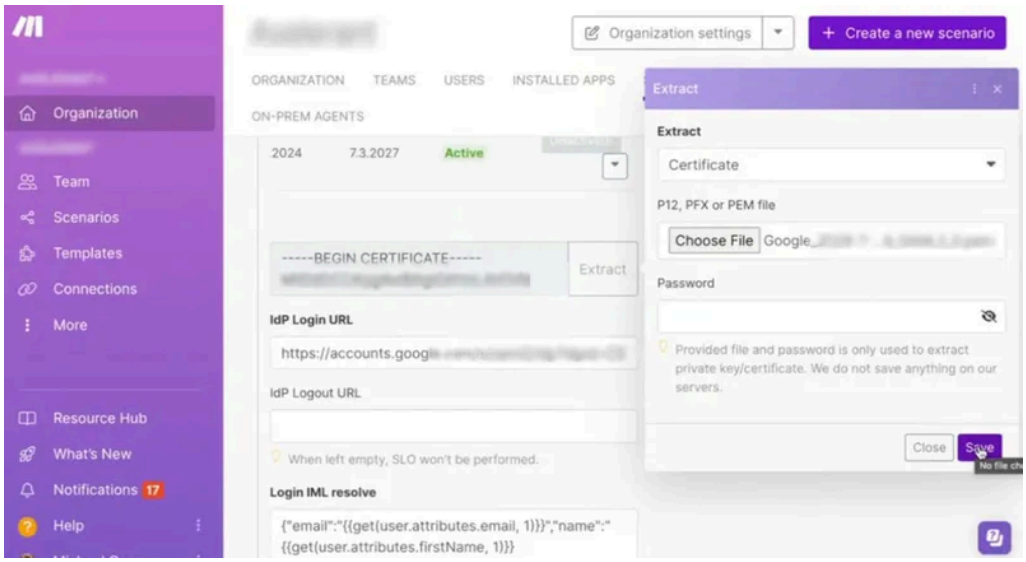
Update the SSO in Make

1 Click **Organization** in the left sidebar.

2 Click the **SSO** tab.

3 Activate the **Service Provider Certificate** and download it.

4 In the **IDP certificate** section, upload the certificate downloaded from step 6 of **Create an SAML application in the Google admin portal** above.



5 Enter the SSO URL from step 5 of **Create an SAML application in the Google admin portal** above and paste it into the **IDP Login URL** field in Make.

6 Enter the **Login IML resolve**.

JS JS

```
{
    "email": "{{get(user.attributes.email, 1)}}",
    "name": "{{get(user.attributes.firstName, 1)}}{{get(user.attributes.
    "id": "{{get(user.attributes.email, 1)}}"
}
```

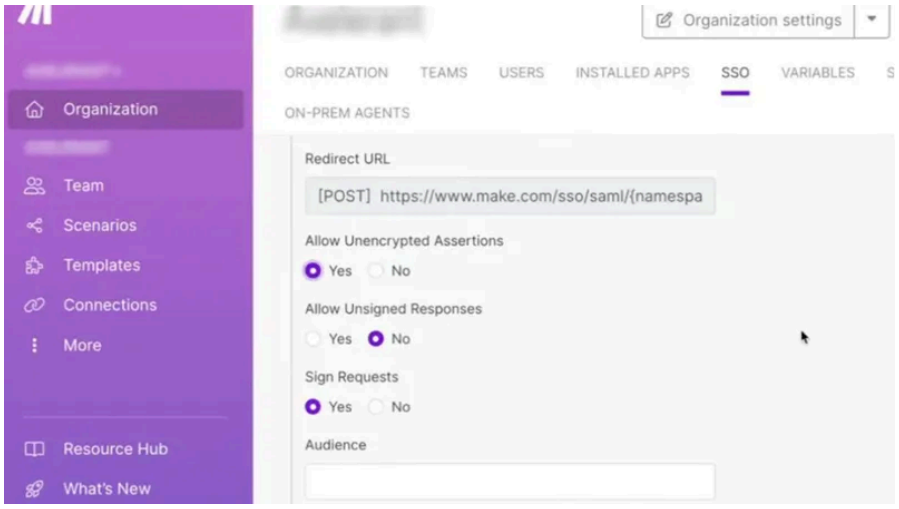
Optional: It is a good practice to validate the JSON string in IML Resolve to ensure it is correct. You can use the **JSONLint** website to perform this validation.

7 Enter the following additional information:

Allows Unencrypted Assertions: Yes

Allow Unsigned Responses: No

Sign Requests: Yes



Service provider initiated SSO

1. Go to **make.com**.
2. Click **Sign in with SSO**.
3. Enter the namespace you chose for your organization.
4. Log in using your Microsoft credentials and consent to Make's access to your user data.

6.4.3. MS Azure AD OIDC

> This feature is available to Enterprise customers.

The following manual configuration creates an OIDC SSO configuration for your Enterprise organization.

Prerequisites

- `Owner` or `admin` role in an Enterprise organization
- Administrative access to your organization's Microsoft Azure AD portal

Supported features

This configuration supports the following:

- Service provider initiated SSO
- Single Log Out [optional]

Configuration steps

Before configuring SSO, you need to assign a namespace and make files of your service provider certificate and private key. These steps provide the information you need to enter later.

Create your namespace in Make

1. Click **Organization** in the left sidebar.
2. Click the **SSO** tab.
3. Under **Namespace**, enter the namespace you want for your organization. For example, `acmecorp`. Your organization members enter this namespace when they

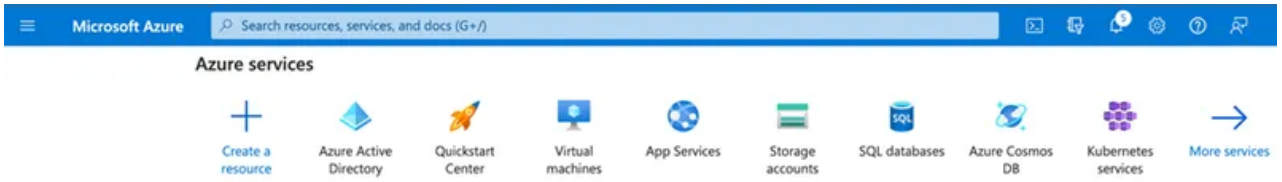
log in via SSO.

4 Under **SSO type**, select **SAML 2.0**.

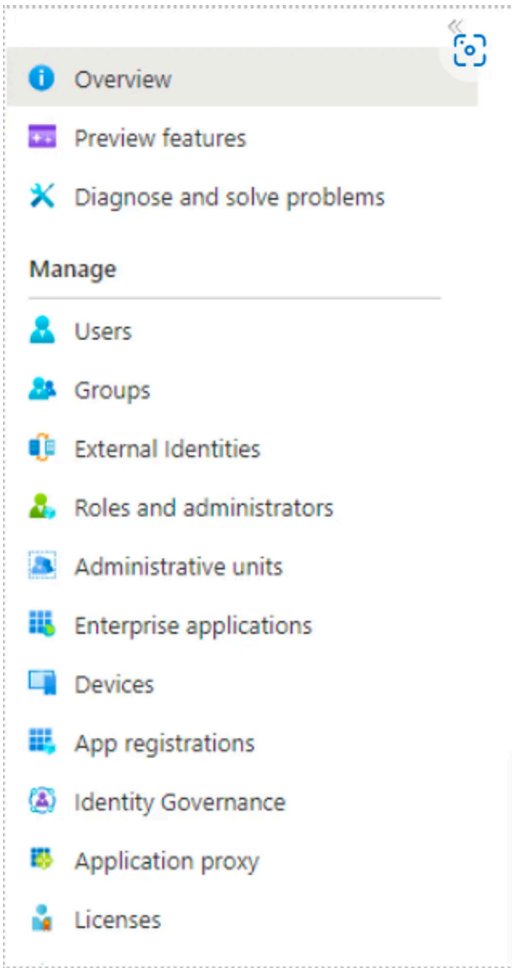
5 Copy the **Redirect URL** and save it in a safe place. You will use this later when you create your SAML integration in the Microsoft Azure AD portal.

Create an OIDC application in the MS Azure portal

1 Log in to the **Microsoft Azure portal** and navigate to the Azure Active Directory.



2 In the left navigation, click **Enterprise applications.**



3 Click **+ New Application**.

4 Click **+ Create your own application**.

5 Enter a name for your app and select **Register an application to integrate with Azure AD (App your're developing)**.

6 Click **Create**.

7 Enter and select the following:

Field on the Register an application page	Required information
Name	Enter a name for your OIDC SSO app.
Supported account types	Select the optoin that is best for your user case. For example, use **Accounts in this organizational directory** only if your application is only for internal use within your organization.
Redirect URI (optional)	Although Microsoft marks this field as optional, successful implementation with Make requires the following: Select a platform - Web - https://next.integromat.com/sso/login

8 Click **Register**.

Create your client credentials in the MS Azure portal

1 In the Microsoft Azure AD portal go to **Home > Enterprise applications > {your OIDC app} > Single Sign-on** and click **Go to application**.

2 Under **Essentials**, find **Application (client) ID**. Copy this value and save it in a secure place. This is the required information for the **Client ID** field in your Make SSO configuration.

3 In the left navigation under **Manage**, click **Certificates & secrets**.

4 Click **+ New client secret**.

5 In the new dialog box, enter a short description and click **Add**.

6. Find the new client secret on the list. Copy the **Value** and save it in a secure place. This is the required information for the **Client secret** field in your Make SSO configuration.

Configure tokens and optional claims in the MS Azure portal

1. In the left navigation under **Manage**, click **Token Configuration**.
2. Click **+ Optional claim**.
3. In the new dialog box, select **ID**.
4. A list appears. Select **Email**.

Add API permissions in the MS Azure portal

1. In the left navigation under **Manage**, click **API Permissions**.
2. Click **+ Add permission**.
3. In the new dialog box, click **Microsoft Graph**.
4. Click **Application permissions**.
5. Use the search bar to find `User.Read.All`.
6. Select `User.Read.All` and click **Add permissions**.

Add users to your application in the MS Azure portal

To provide access to your organization members, you need to add these users to your app in the MS Azure portal.

1. In the Microsoft Azure AD portal go to **Home > Enterprise applications > {your OIDC app}**.
2. Click **Users and groups**.
3. Click **+ Add user/group** to add the users you want to access your Make organization.

Update the SSO in Make

1. Click **Organization** in the left sidebar.
2. Click the **SSO** tab.
3. Enter the following information:

Field	Value
User information URL	`https://graph.microsoft.com/v1.0/me`
Client ID	Enter the **Application (client) ID** you copied in step 2 of **how to create your client credentials**.
Token URL	`https://login.microsoftonline.com/1234etc/oauth2/v2.0/token`
Login scopes	`User.Read.All`
Scopes separator	Enter a single space.
Authorize URL	To find your Authorize URL: 1. In the MS Azure portal, go to **Home > Enterprise applications > {your OIDC app} > Single Sign-on** and click **Go to application.** 2. Click **Endpoints**. A window appears. 3. Find **OAuth 2.0 authorization endpoint (v1)**. Copy and paste this URL into your Make configuration.
Client secret	Enter the **Value** you copied in step 6 of **how to create your client credentials**.
User infomration IML resolve	`{"id":"{{id}}","email":"{{mail}}","name":"{{givenName}}"}`
Redirect URL	No action required
Team provisioning for new users	Select an option based on your needs.

4 Click **Save**.

You will receive an email with the subject "Activation complete: SSO ready for your organization" upon successful activation. If you encounter any issues while logging in using SSO, disable SSO using the "one-time link" (valid for 24 hours).

Service provider initiated SSO

1. Go to **make.com**.
2. Click **Sign in with SSO**.
3. Enter the namespace you chose for your organization.
4. Log in using your Microsoft credentials and consent to Make's access to your user data.

6.4.4. MS Azure AD SAML

> This feature is available to Enterprise customers.

The following manual configuration creates an SAML SSO configuration for your Enterprise organization.

Prerequisites

- `Owner` role in an Enterprise organization
- Administrative access to your organization's Microsoft Azure AD portal

Supported features

This configuration supports the following:

- Service provider initiated SSO
- Single Log Out [optional]

Configuration steps

Before configuring SSO, you need to assign a namespace and download your service provider certificate in Make. These steps provide information you need to enter later.

Create your namespace in Make

1. Click **Organization** in the left sidebar.
2. Click the **SSO** tab.
3. Under **Namespace**, enter the namespace you want for your organization. For example, `acmecorp`. Your organization members enter this namespace when they log in via SSO.

4 Under **SSO type**, select **SAML 2.0**.

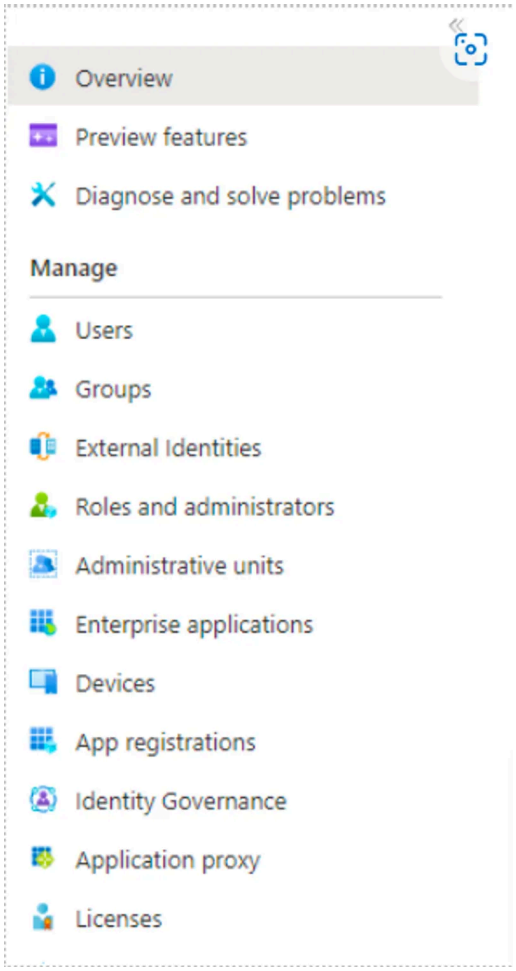
5 Copy the **Redirect URL** and save it in a safe place. You will use this later when you create your SAML integration in the Microsoft Azure AD portal.

Create an SAML application in the MS Azure portal

1 Log in to the **Microsoft Azure portal** and navigate to the Azure Active Directory.



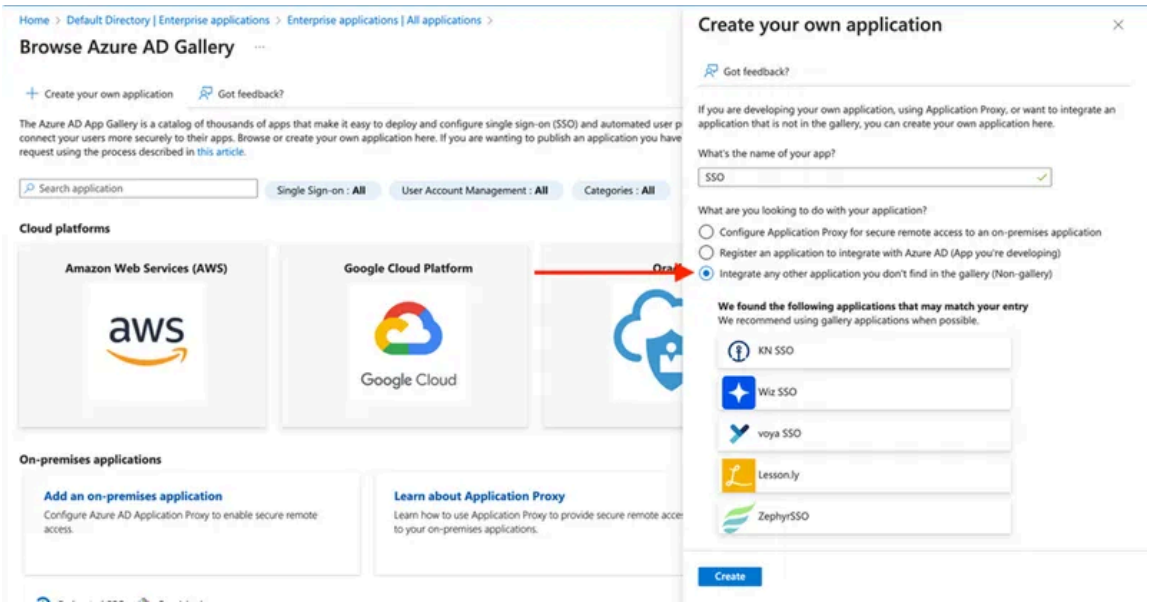
2 In the left navigation, click **Enterprise applications.**



3 Click **+ New Application**.

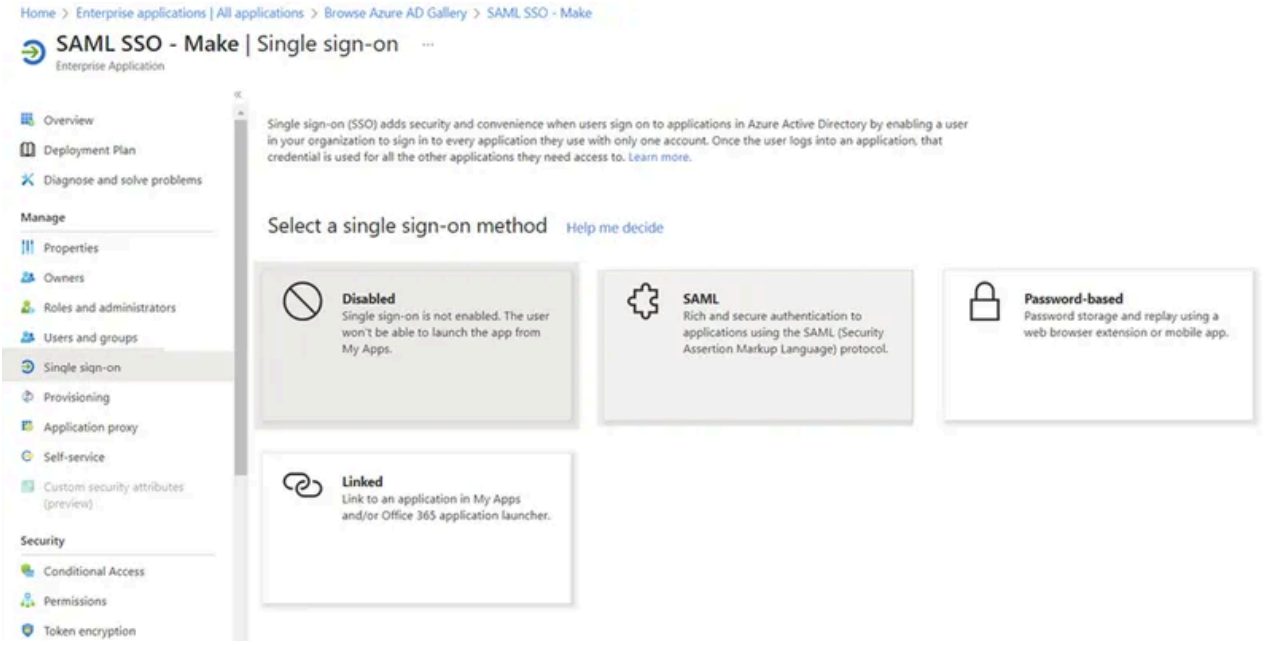
4 Click **+ Create your own application**.

5 Enter a name for your app and select **Integrate any other application you don't find in the gallery**.



6 Click **Create**.

7 In the left navigation, click **Single Sign-on.**



8 Click **SAML**.

9 Configure your Basic SAML settings using the Redirect URL (`https://www.make.com/sso/saml/{namespace}`) that you copied in the **the steps above**.

Field	Value
Entity ID	`https://www.make.com/sso/saml/{namespace}/metadata.xml`
Reply URL	`https://www.make.com/sso/saml/{namespace}`
Logout URL	`https://www.make.com/sso/saml/{namespace}`



10 In the **Attributes & Claims** section, click **Edit** to rename your attributes.



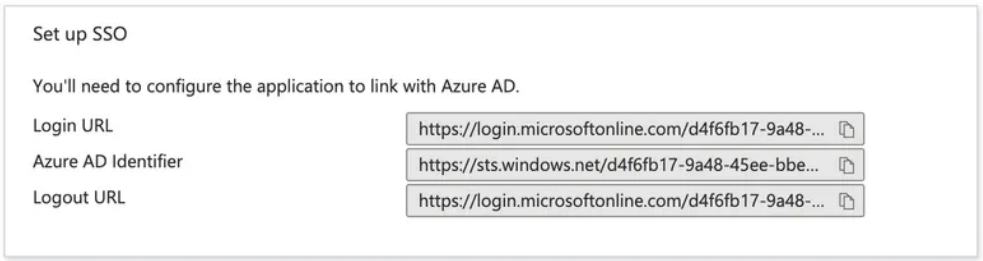
11 Under **Additional claims**, find the value you want to edit and click that row.

12 Enter the new name in the Name field. Use the following chart to find the names required for your IML resolve.

Field	Value
email	user.mail
name	user.displayname
id	user.userprincipalname
Unique User Identifier	user.userprincipalname

13 Click **Save**.

14 Copy the **Login URL** and save it in a safe place.



Download the SAML certificate

You need to download the base 64 SAML certificate from Microsoft Azure and upload it to the **Identity Provider Certificate** field of the **SSO** tab in your Make organization.

1. Find the **SAML Certificates** section of your single sign-on settings in the Microsoft Azure portal.

2. Next to **Certificate (Base64)**, click **Download**.

Your browser automatically downloads the `.cer` file.

Update the SSO in Make

1. Click **Organization** in the left sidebar.

2. Click the **SSO** tab.

3. Under **Identity Provider Certificate**, click **Extract**. A pop-up appears.

4. Under **P12, PFX or PEM file**, click **Choose file** and select the `.cer` file you downloaded.

5. Enter the following information from MS Azure into the **IdP login URL** and **Identify provider certificate** fields.

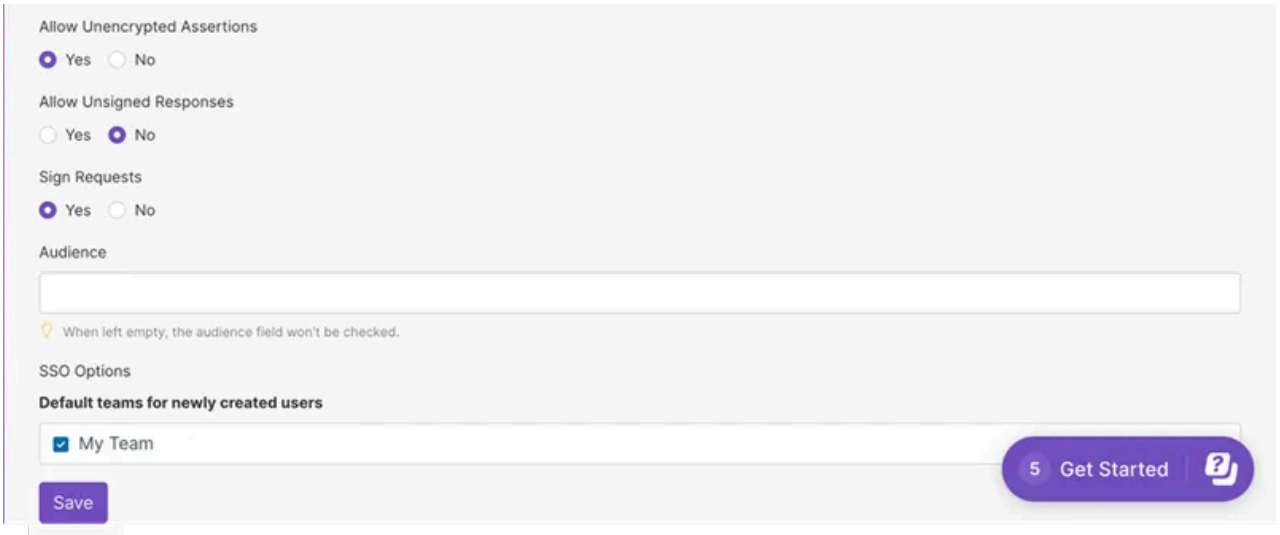
Field	Value to enter from MS Azure
IdP login URL	Login URL
Identity provider certificate	Certificate (Base 64)

6. Enter the following in the **Login IML resolve** field:

```JS
{"email":"{{user.name_id}}","id":"{{get(user.attributes.id,1)}}","name":
```

7 Select the following settings:

Field	Value
Allow Unencrypted Assertions	Yes
Allow Unsigned Responses	No
Sign Requests	Yes



8 Click **Save**.

> You will receive an email with the subject "Activation complete: SSO ready for your organization" upon successful activation. If you encounter any issues while logging in using SSO, disable SSO using the "one-time link" (valid for 24 hours).

Service Provider initiated SSO

1 Go to **make.com**.

2 Click **Sign in with SSO**.

3 Enter the namespace you chose for your organization.

4 Log in using your Microsoft credentials and consent to Make's access to your user data.

6.4.5. Okta SAML

> This feature is available to Enterprise customers.

The following manual configuration creates an SAML SSO configuration for your Enterprise organization.

Prerequisites

- `Owner` role in an Enterprise organization
- Okta account with admin access

Supported features

This configuration supports the following:

- Service Provider initiated SSO
- Single Log Out [optional]

Configuration steps

Before configuring SSO, you need to assign a namespace and create a Service Provider certificate and private key. These important steps provide information you need to enter later.

Create your namespace

1. Click **Organization** in the left sidebar.
2. Click the **SSO** tab.
3. Under **Namespace**, enter the namespace you want for your organization. For example, `acmecorp`. Your organization members enter this namespace when they

log in via SSO.

4. Under **SSO type**, select **SAML 2.0**.

5. Copy the **Redirect URL** and save it in a safe place. You will use this later when you create your SAML integration in Okta.

Download your Make Service Provider certificate

1. Click **Organization** in the left sidebar.

2. Click the **SSO** tab.

3. Scroll down to find **Service Provider Certificates**.

4. Find your new certificate. Refer to the **Valid from** and **Expires** dates if you are unsure.

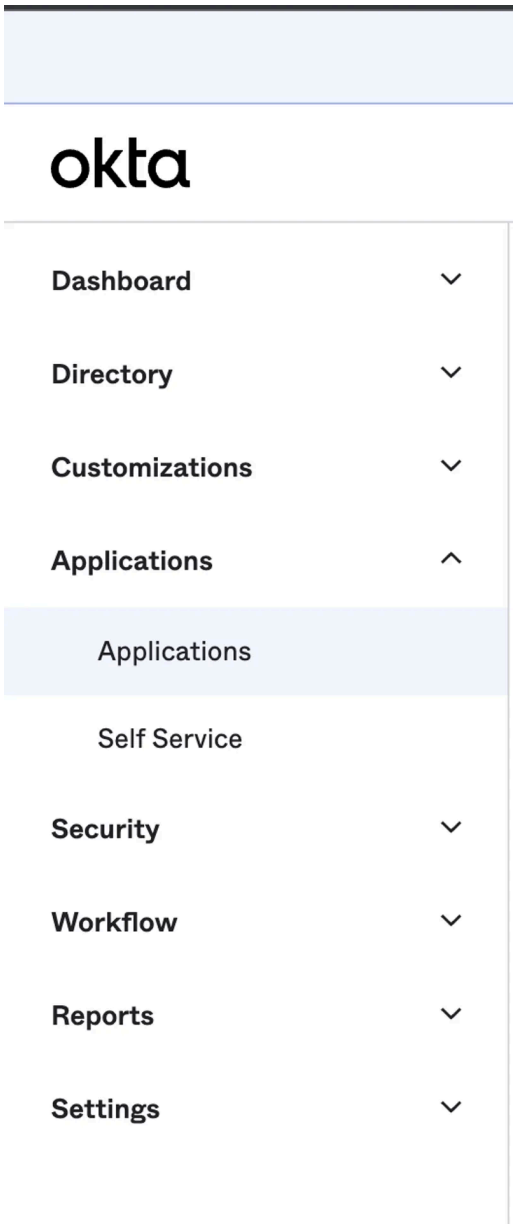
5. On the right side of the row for your certificate, click the icon.

6. Select **Download**.

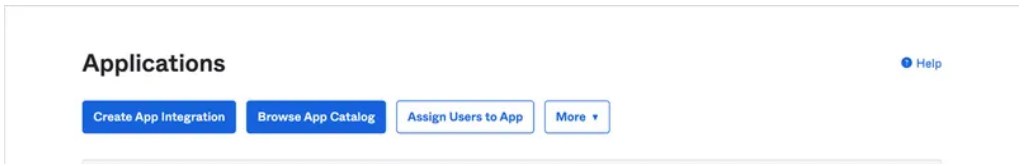
Your browser downloads your SP certificate as a `.pem` file. You can find it in your downloads folder.

Create an SAML integration

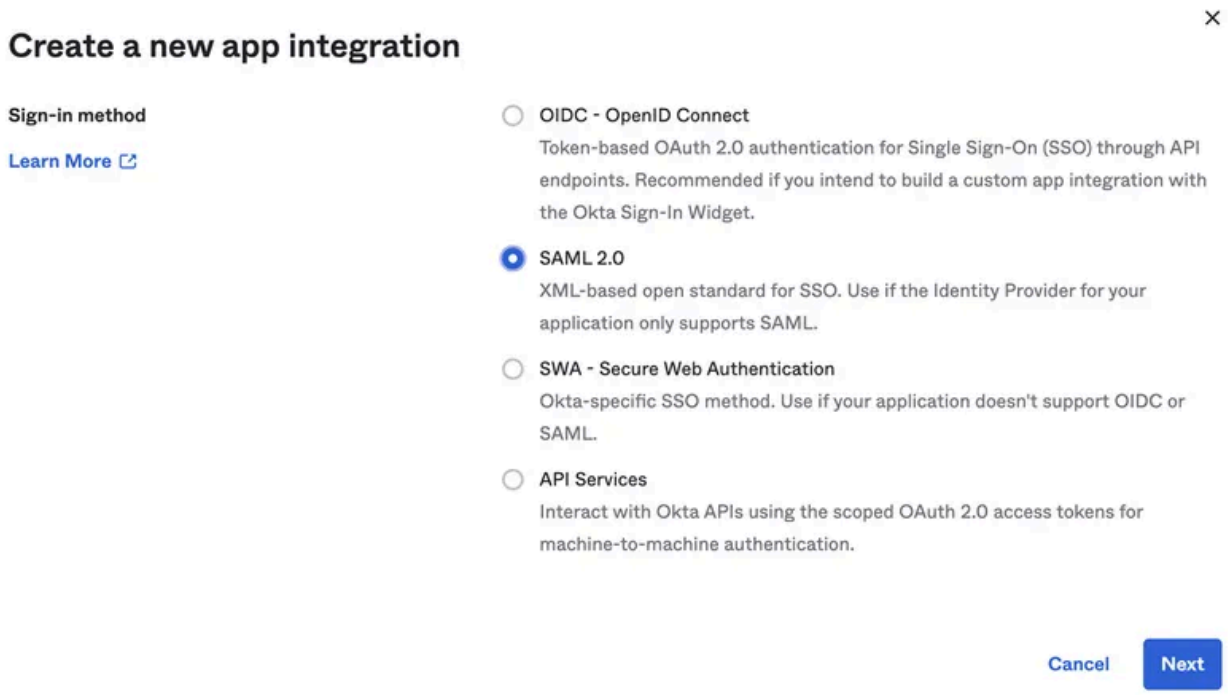
1. Log in to Okta and go to **Applications > Applications**.



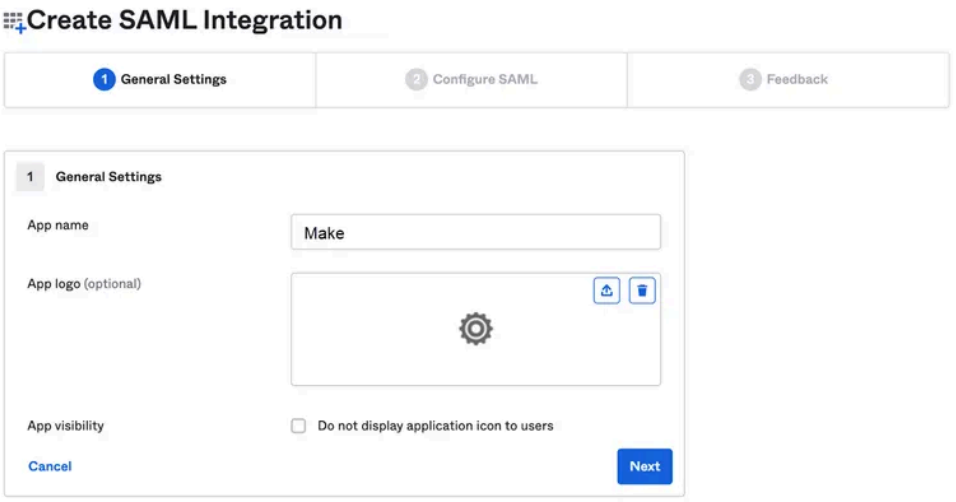
2 Click **Create App Integration**.



3 Select **SAML 2.0** in the popup winder and click **Next**.

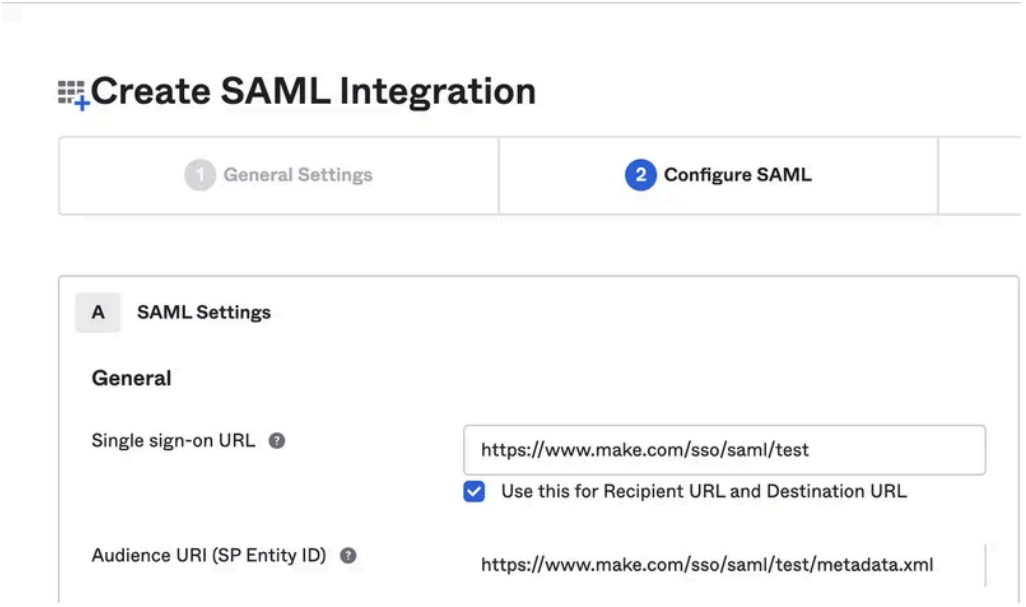


4 In the **General Settings** tab, name your app and upload your icon.



5 Click **Next**.

6 In the **Configure SAML** tab, enter the Single sign-on URL that you copied in the steps to **create your namespace** above.



7 Enter the Audience URI (SP Enttity ID) as `https://www.make.com/sso/saml/{{namespace}}/metadata.xml`.

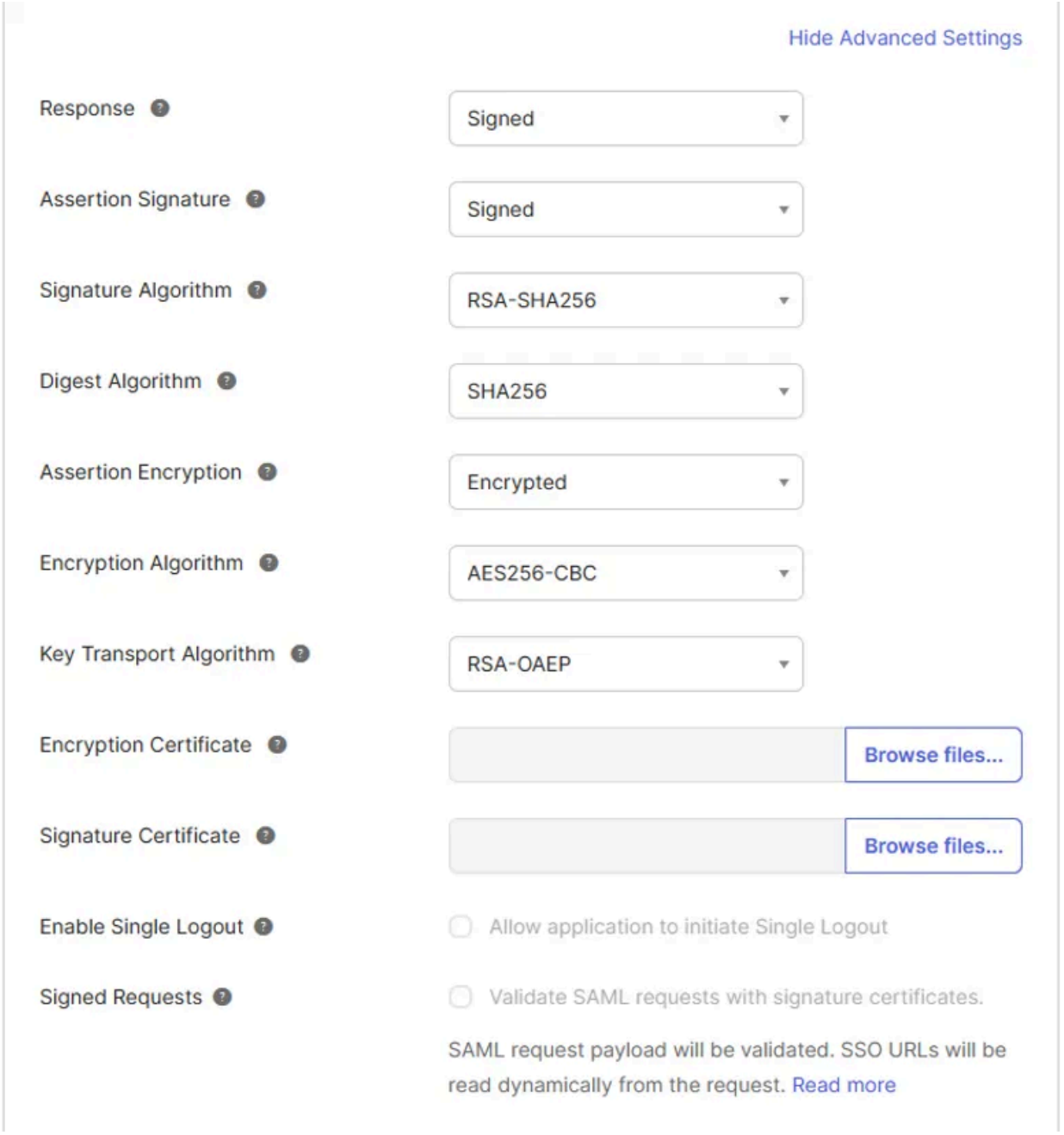
8 Keep the **Default Relay State** blank.

9 Enter the following information:

Field	Value
Name ID format	EmailAddress
Application Username	Okta username
Update application username on	Create and update

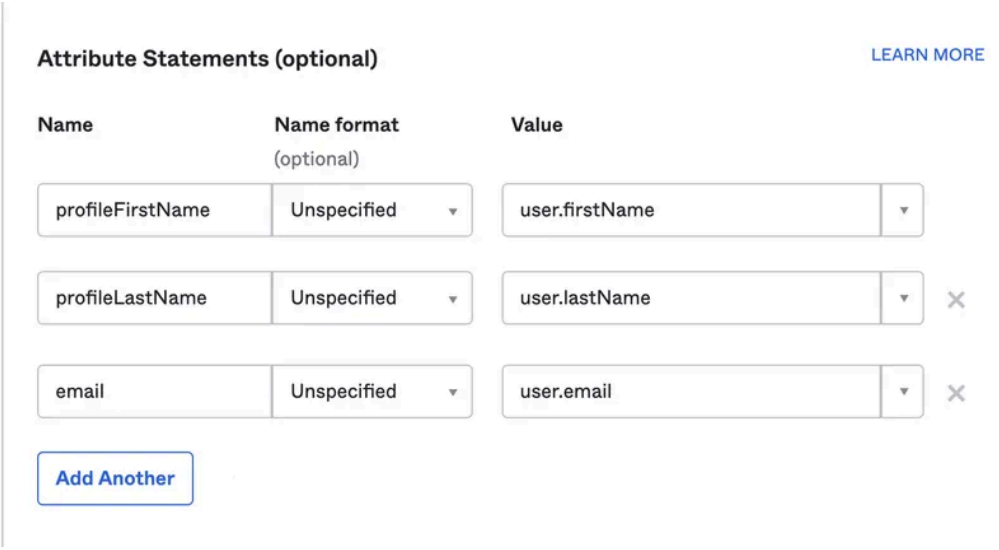
10 Click **Show Advanced Settings**.

11 Verify that all of the information provided matches the following:

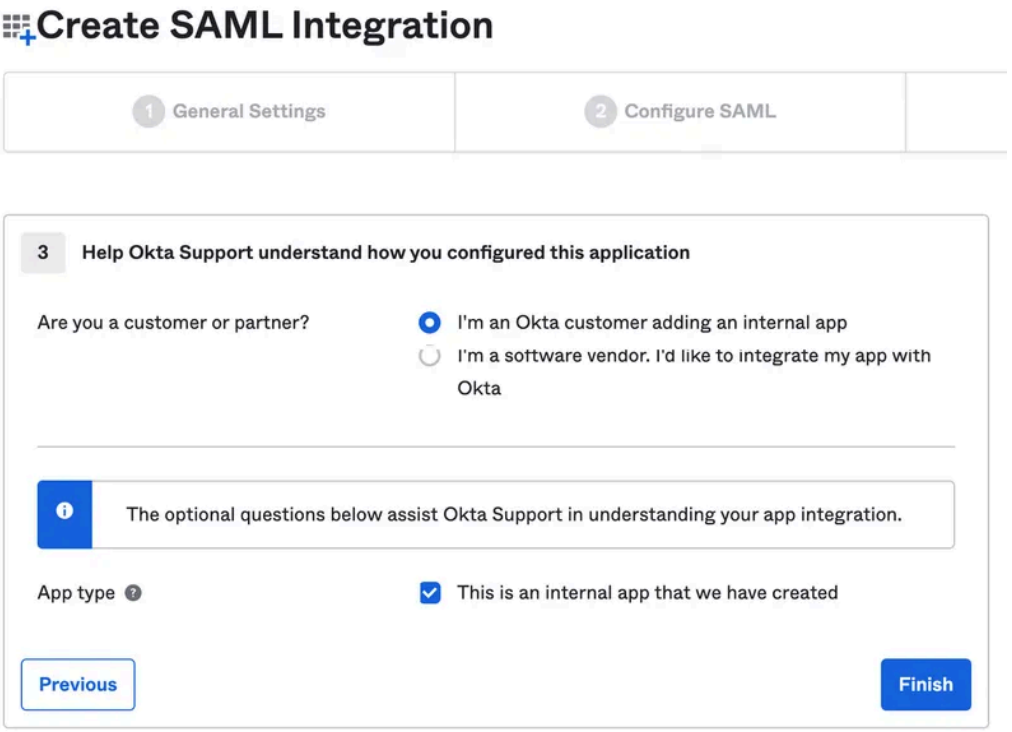


Set **Assertion Encryption** to **Encrypted**. For the Signature Certificate field, upload the Make **Service Provider Certificate** you downloaded above.

12 Under **Attibute Statements (optional)**, add the attribute as shown in the image and click **Next** to save.



13 Select the following options and click **Finish**.



14 Assign people to your Make application under the **Assignments** tab in Okta.

15 Under the **Sign On** tab in Okta, view the SAML setup instructions.

16 Copy the **Identity Provider Single Sign-On URL** and the **Identify provider certficate** and save them in a safe place.

17 In Make, under **Organization > SSO**, update the **IdP log URL** field and the **Identify provider certficate** field with the information obtained in Okta in step 16.

18 Enter the following in the **Login IML resolve** field:

```
{"email":"{{get(user.attributes.email, 1)}}","name":"{{get(user.attribut
```

19 Set:

- **Allow Unecrypted Assertions** to **No**
- **Allow Unsigned Responses** to **No**
- **Sign Requests** to **Yes**

20 Select the team and **Save**.

Once saved, the page will reload. Sign out.

> You will receive an email with the subject **Activation complete: SSO ready for your organization** upon successful activation. If you encounter any issues while logging in using SSO, disable SSO using the "one-time link" (valid for 24 hours).

Service Provider initiated SSO

1. Go to **make.com**.
2. Click **Sign in with SSO**.
3. Enter the namespace you chose for your organization.
4. Log in using your Okta credentials and consent to Make's access to your user data.

6.4.6. SAML certificate management

> This feature is available to Enterprise customers.

The SSO setup page lets you manage your service provider (SP) certificates. You can activate, deactivate, copy, or download your SP certificates.

Make provides a new certificate when your active SP certificate is close to expiring. Email notifications let you know when it's time to rotate your certificates.

Rotate service provider certificates

To maintain the security of your SSO setup, Make supports service provider (SP) certificate rotation on a three-year basis. When your SP certificate is 90 days from expiring, Make provides a new certificate and sends you an email. Rotate your certificate before it expires to avoid login failure. You can see when your certificate expires by looking under the **Expires** column of the **Service Provider Certificates** section of your SSO setup.

You can activate your new certificate and copy or download it with the following steps:

1. Click **Organization** in the left sidebar.
2. Click the **SSO** tab.
3. Under **SSO configuration**, find your **Service Provider Certificates**.
4. Find your new certificate. Refer to the **Valid from** and **Expires** dates if you are unsure.
5. On the right side of the row for your new certificate, click **Activate**.

6. Next to **Activate**, click the icon to download or copy your certificate, depending on how you need to enter your certificates with your identity provider.

7. Go to your identity provider and update your service provider certificate. If you use Okta, refer to our **Okta SAML page** for details.

If you have more than one active certificate, Make deactivates the certificate that expires first. You can check the **Expires** column to see when your certificates expire.

> Only click **Save** if you make other changes to your setup. Clicking **Save**:
>
> - is not required to activate and rotate your certificate.
> - immediately logs out all organization members.

Activate a certificate

You can see which certificates are active by looking in the **Status** column.

Active means the certificate is in use in your SAML SSO configuration. No further action required.

Inactive means the certificate is not used in your SAML SSO configuration. Make automatically deactivates certificates that expire as long as you have another valid active certificate.

1. Click **Organization** in the left sidebar.

2. Click the **SSO** tab.

3. Under **SSO configuration**, find your **Service Provider Certificates**.

4. Find the certificate in the list.

5. Under **Actions**, click **Activate**.

6. A popup asks you to confirm activation. Click **Activate**.

> Only click **Save** if you make other changes to your setup. Clicking **Save**:
>
> - is not required to activate your certificate.
> - immediately logs out all organization members.

Deactivate a certificate

You can only deactivate a certificate if there is another active certificate. This prevents accidental deactivation of your only active certificate. At least once certificate must be active.

If you have more than one active certificate, Make deactivates the older certificate for you when it expires. Don't worry, Make won't deactivate your only active certificate.

1. Click **Organization** in the left sidebar.

2. Click the **SSO** tab.

3. Under **SSO configuration**, find your **Service Provider Certificates**.

4. Find the certificate in the list.

5. Under **Actions**, click **Deactivate**.

Copy a certificate

If your identity provider (IdP) lets you paste your service provider (SP) certificates into your setup, you can copy your SP certificate into your clipboard.

1. Click **Organization** in the left sidebar.
2. Click the **SSO** tab.
3. Under **SSO configuration**, find your **Service Provider Certificates**.
4. Find the certificate in the list.
5. On the right side of the row for your certificate, click the icon.
6. Select **Copy**.

Your SP certificate is copied to your clipboard and ready to paste into your IdP setup.

Download a certificate

If your identity provider (IdP) lets you upload your service provider (SP) certificates into your setup, you can download your SP certificate as a `.pem` file.

1. Click **Organization** in the left sidebar.
2. Click the **SSO** tab.
3. Under **SSO configuration**, find your **Service Provider Certificates**.
4. Find the certificate in the list.

5. On the right side of the row for your certificate, click the icon.

6. Select **Download**.

Your browser downloads your SP certificate as a `.pem` file. You can find it in your downloads folder.

6.4.7. Domain claim

> This feature is available to Enterprise customers.

Claiming a domain prevents your members from accidentally creating their own self-service accounts. After you set up single sign-on (SSO), claim your email domain so Make can recognize your new users. Any new user who signs in with your claimed email domain gets a prompt to use SSO. You can provision your new members by:

- Creating a global provision that assigns all new members to a team you choose.
- **Using an API endpoint to create new users and assign them to a team.**

To claim a domain:

1. **Add your domain on Make's SSO setup page**.
2. **Add your verification code to your domain's DNS**.
3. **Verify your domain on Make**.

Once verified, Make recognizes login and registration attempts using your email domain and redirects them to your SSO.

Add your domain on Make

1. Click **Organization** in the left sidebar.
2. Click the **SSO** tab.
3. Click **Domain verification** to expand that section.
4. Click **+Add domain**.

5. In the dialog box, for **Domain**, enter your organization's domain. For example: `acme.com`, `mybusiness.net`

6. Click **Add**.

The dialog box displays the verification code you need to add as a TXT record to your domain's DNS.

Your verification code can be found at **Organization > SSO > Domain verification**.

Add your verification code to your domain's DNS records

Once you have your verification, add it to your domain's DNS records as TXT. The details vary based on your domain host.

The following steps describe the general process. Contact your domain host for detailed help.

1. In a separate browser window or tab, go to your domain host and sign in.

2. Find the DNS records section of your domain host. Possible names are **DNS Management**, **Name Server Management**, **Control Panel**, **Advanced Settings** or similar.

3. Find and select the option to add a new record.

4. Use the following information to add a new record:

Possible field name	Information to enter
Record type or **Type**	TXT
Name, **Host, Hostname, or Alias**	Leave blank or enter @ Check with your domain host's support documentation for more information.
Value, Data, Answer, Destination, or Points to	Paste your verification code from Make.

5 Save your new record.

Verify your domain on Make

After you add your verification to your DNS, you can complete the process on Make.

1 Click **Organization** in the left sidebar.

2 Click the **SSO** tab.

3 Click **Domain verification** to expand that section.

4 Find your domain on the list.

5 Under **Actions**, click **Verify**.

Your status changes to **Verified** and the domain claim is successfully configured.

Verification status

Status	Explanation and steps required
Unverified	You have not yet verified your domain. Complete the steps in this article to resolve this status.
Verifed	You have successfully claimed your domain. No further action required.
Suspended	You have disabled your SSO configuration. Enable SSO and try to verify again.
Failed	Possible reasons include: • The DNS is unavailable: Check with your domain host. • The DNS record does not exist: Wait and try again. If the problem continues, contact your domain host. • The DNS record contains an incorrect value: Review the code you entered with your domain host and correct any errors. • Another organization has verified the domain: Check that you do not have another organization with a domain claim.

Provisioning new users

Without a claimed domain, you can use the SSO settings to globally assign users to a team that you specify. This provisioning happens when a new user logs in using your identity provider. Make gets information such as the email address and user name from your identity provider during login.

After you claim a domain, you can use an API endpoint to create new users and assign them to specific teams.

Create and provision users for Enterprise SSO

You can create a new user in your organization and assign their SSO external ID by calling the `POST /users/user-sso-create` endpoint. The `addUserToDefaultSsoTeams` parameter lets you define whether the new user is automatically added to the default team(s) designated in your SSO configuration. If the new user is not assigned to your default team(s), you can assign the new user to any team in your organization by using the API call `POST/users/{userId}/user-team-roles/{teamId}`.

`POST` /users/user-sso-create

Required permission: `users sso edit`

This endpoint creates a new user in your organization and assigns their SSO external ID in one API call. Use the `addUserToDefaultSsoTeams` parameter to define whether the new user is automatically added to the team(s) designated as default in your SSO configuration. If the new user is not assigned to your default team(s), you need to **assign new users to your teams**.

Parameter name	Data type	Required?	Description
`email`	string	**Yes**	The user's email. Example: john@doe.com
`name`	string	**Yes**	The name of the user. Example: John Doe
`countryId`	integer	**Yes**	The ID of the user's country. Get the list of country IDs with the API call `GET /enums/countries`. Example: 1
`timezoneId`	integer	No	The ID of the user's timezone. Get the list of the timezone IDs with the API call `GET /enums/timezones`. Example: 113
`language`	string	No	The language of the user's UI. Make supports: English: value `en` Czech: value `cs`
`localeId`	integer	No	The ID of the user's locale. Get the list of locale IDs with the API call `GET /enums/locales`. Example: 18
`ssoId`	string	**Yes**	The external ID for SSO. You can associate any parameter in the SSO payload to identify users, such as email or an ID from your Identity provider. Examples: example.employee@example.com 12EXample3456id78string90

Parameter name	Data type	Required?	Description
`organizationId`	integer	**Yes**	The ID of your organization. Example: 22
`addUserToDefaultSsoTeams`	boolean	No	This parameter is deprecated. The SSO configuration in the UI has an option to not assign new users to a team.

Example request:

TypeScript

```
{
    "email": "a.example@example.com",
    "name": "Anne Example",
    "countryId": 1,
    "timezoneId": 113,
    "language": "cs",
    "localeId": 11,
    "ssoId": "a1e8af1bd9b4d2c602fb8c687182633c6854b0e7",
    "organizationId": 54,
    "addUserToDefaultSsoTeams: false
}
```

Example response:

TypeScript


```
{
    "user": {
        "user_id": 86,
        "email": "a.example@example.com",
        "name": "Anne Example",
        "country_id": 1,
        "language": "cs",
        "locale_id": 11,
        "timezone_id": 113,
        "sso_id": "a1e8af1bd9b4d2c602fb8c687182633c6854b0e7",
        "organization_id": 54
    }
}
```


The response includes `user_id` and `organization_id` which you can map in a scenario to provision your users.

6.5. Make Managed Services (MMS)

Make Managed Services (MMS) is a product offering that lets distributors of Make manage multiple organizations under a single entity.

With MMS, organizations can now easily distribute Enterprise licenses to clients and manage their operations in one place, or create automations for clients after an easy, quick user setup.

Who can use it

Make Managed Services is useful to distributors that need to manage organizations on behalf of their clients, such as Make resellers or automation experts who build for organizations.

What you can do with it

Make Managed Services allows organizations to:

- Create child organizations
- Allocate and adjust operations across child organizations
- Assign ownership of child organizations
- Pause child organizations
- Unlink child organizations

MMS offers separate data between child and distributor organizations, flexible operation distribution, and integrated Single Sign-On (SSO) capabilities across organizations.

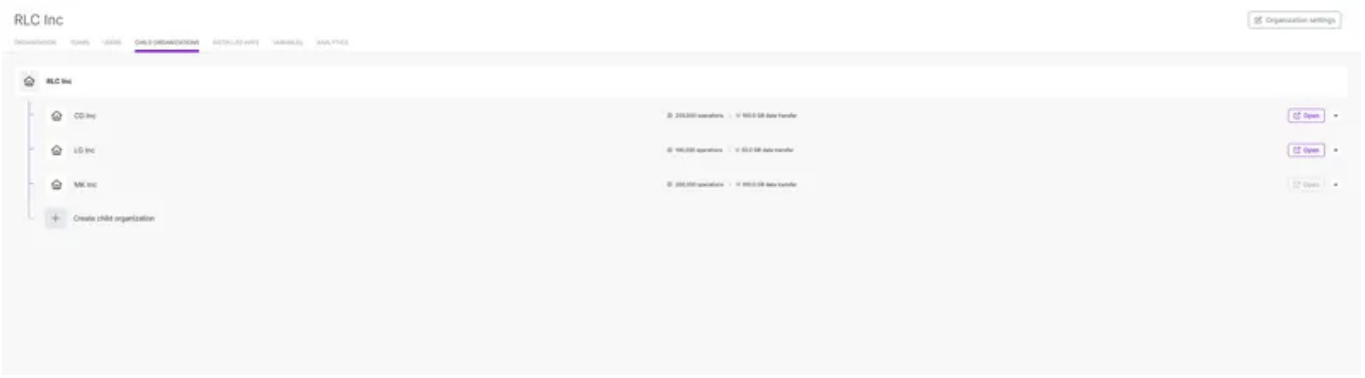
How it works

Make Managed Services is a space in Make to manage child organizations and allocate their operations.

MMS consists of a distributor organization and its child organizations:

- The **distributor organization** is the organization that can create and manage child organizations, assign their owners, and allocate operations.
- The **child organization** is the organization linked to the distributor organization. Child organizations have access to all Enterprise user benefits, except for allocating their own operations. They can invite distributor organizations (including via **SSO**) to access their organization.

In this example MMS overview, RLC Inc. is the distributor organization and CD Inc., LG Inc., and MK Inc. are its child organizations:



Distributor organizations allocate operations to child organizations by drawing from their total operations.

To learn more about Make Managed Services, **talk to sales**.

7. Your profile

Find resources to manage your profile and explore Make's programs:

Profile settings

Manage your profile settings, 2FA, and create API keys

Make programs

Read about our Affiliate and NGO programs

7.1. Profile settings

Manage your profile settings, 2FA, and create API keys:

→ **Change your email address**

→ **Manage your email preferences**

→ **Delete your profile**

→ **Manage time zones**

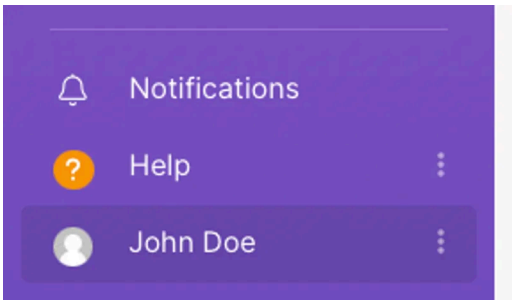
→ **API key**

→ **Two-factor authentication**

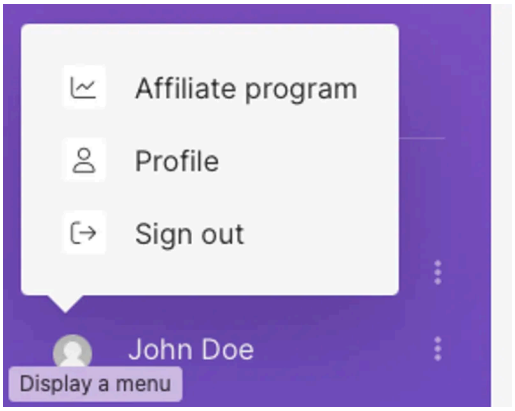
7.1.1. Change your email address

You can change your email address from your **Profile** page.

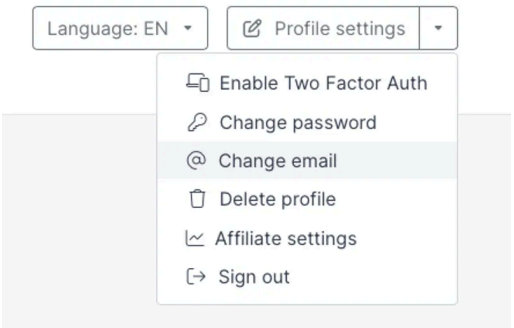
1. Click on your username in the bottom of the left sidebar.



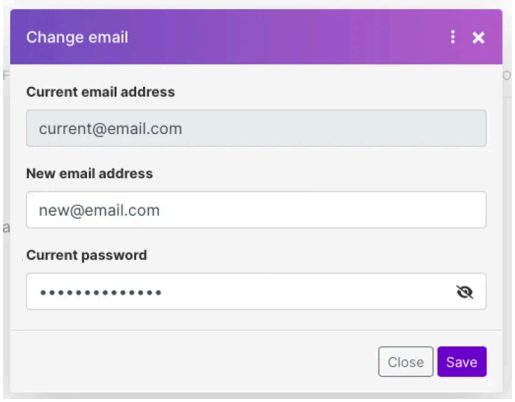
2. Select **Profile**.



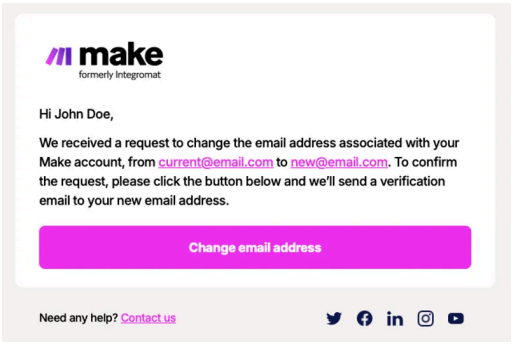
3. In the top-right corner of the Profile page, select **Profile settings > Change email**.



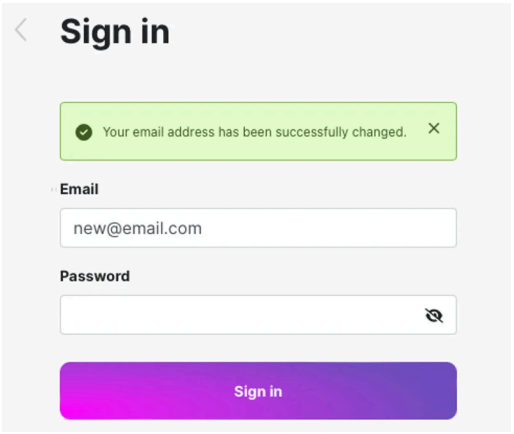
4 Enter your new email address in the **New email address** field, enter your **Current password**, and click **Save**.



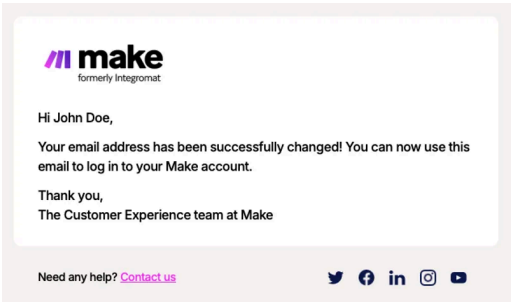
5 A confirmation email will be sent to your **current email address.** Click the **Change email address** confirmation button in the email to approve the change.



6 Upon clicking the **Change email address** button, you'll be redirected to the login page. Your new email will be displayed in the **Email** field and you will see a message indicating your email address has been successfully changed.



7 You'll also receive a confirmation email, sent to the new email address associated with your account.



For security purposes, verification of your change email request must be completed within 24 hours, at which point the links will expire.

7.1.2. Manage your email preferences

You can manage your email notification preferences from your **Profile** page. The options on the **Email preferences** tab let you customize which email notifications you receive from Make . You can opt-in to emails with information about specific topics and decide which scenario-related notifications you receive from each organization you are a member of.

Global notifications

The Global notifications section includes options for communication about Make overall. For example, you can choose to receive emails about updates to Make, special offers and promotions, or announcements about the Make Academy.

Organizations

Make can automatically send notification emails when your scenario encounters a warning, an error, or a deactivation due to errors. By default, you receive emails in the following cases:

- An error prevents your scenario from successfully running.
- Make automatically disables your scenario due to errors.

You can customize your email notifications for each organization you belong to.

- **Deactivation** - these notifications inform you whenever Make disables a scenario because of errors.
- **Warning** - these notifications inform you of issues in a scenario that require your attention. For example, issues with an app's connections.
- **Errors** - these notifications inform you of issues that prevent a scenario from successfully completing.

> Make sends an email immediately when there is a warning or error. Then, Make watches for further warnings or errors and sends you a digest email (after 15 minutes for warnings and after 5 minutes for errors). This second email links to the scenario history where you can find more details.

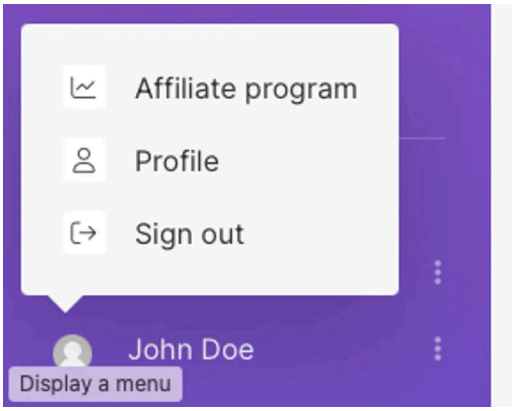
7.1.3. Delete your profile

We're sad to see you go!

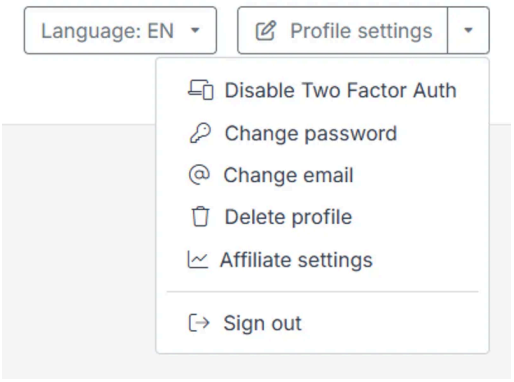
> Once you delete your profile, you won't be able to restore it. We suggest you consider canceling the subscription and **downgrading to a Free plan** in case you still want to have access to your created scenarios.

1. Click on your username at the bottom of the left sidebar.

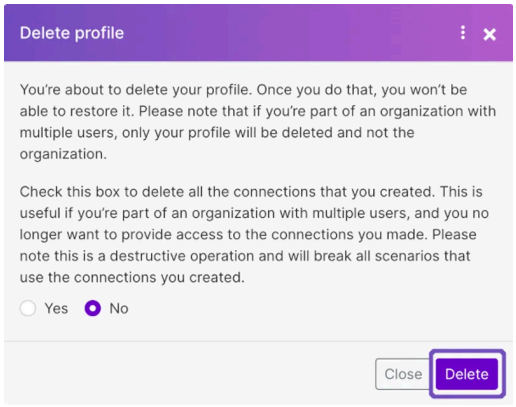
2. Select **Profile**.



3. Under **Profile settings** in the upper-right corner, select **Profile settings > Delete profile**.



4 In the pop-up that opens, you can select if you want to delete all the connections you've created. If you decide to delete your connections, all the scenarios using these connections will stop working. If you decide to keep the connections, other users in your organization will still be able to use them. Click **Delete** to confirm your decision.



If you're a part of the organization with multiple users, only your profile will be deleted, not the whole organization. If you are the only user in the organization, the whole organization will be deleted together with your profile.

7.1.4. Manage time zones

In Make, every **organization** and every user has a definable time zone parameter. However, Make uses organization time zones and user time zones differently:

- The **organization time zone** defines the time used when executing scenarios and modules. For example, a scenario schedule of "4:00 pm every day" means 4:00 pm in the organization's time zone.
- The **user time zone** is similar to **locale** and defines how Make displays time, for example, in the **scenario execution history**.

As a result, organization members in a time zone different from the organization's time zone may see different times displayed. See the **example** below for more details.

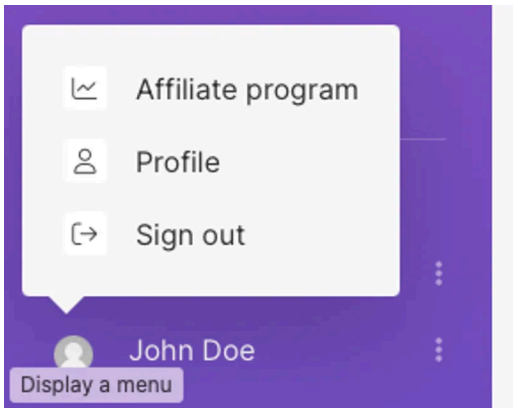
Your profile page lets you view and edit time zone settings from the **Time Zone Options** tab.

Manage your organization time zone

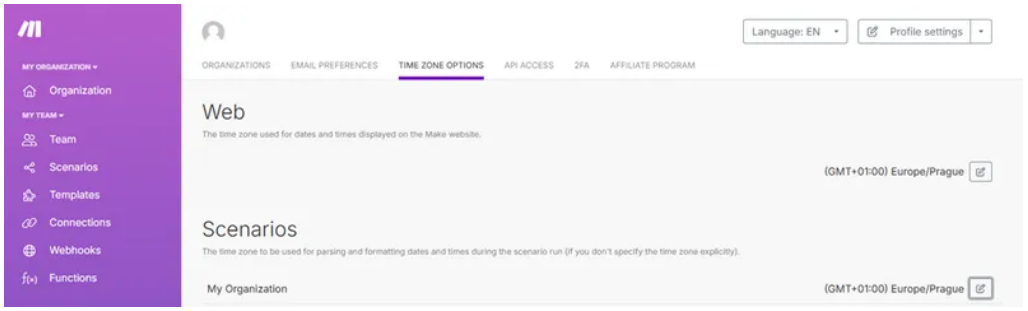
Make uses an organization's time zone to define when a scheduled scenario executes. The organization time zone defines scenario execution and functions such as `formatDate` and `parseDate`. Unless otherwise specified, Make uses the organization time zone as the default time for time and date-related functions and scenario execution. You can view and edit your organization time zones on the **Time Zone Options** tab of the profile page. afsa

> Only organization owners and admins can edit the organization's time zone.

1. Click on your username at the bottom of the left sidebar.
2. Select **Profile**.



3 Click the **Time Zone Options** tab.

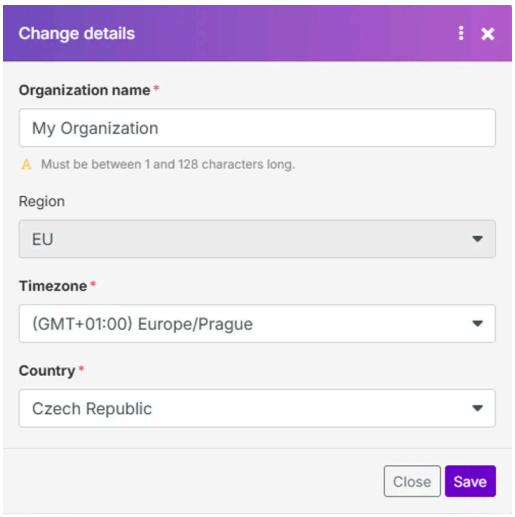


4 Make lists your organizations under Scenarios and displays the current time zone setting.

5 Click the edit icon to edit an organization's setting.



6 In the dialog box, use the menu to search for and select a new time zone.



7 Click **Save**.

The selected time zone now appears on the **Time Zone Options** page.

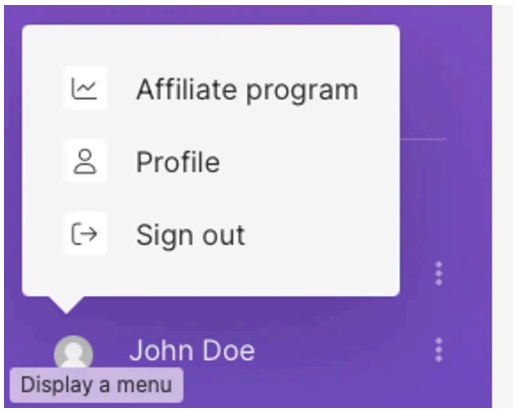
Any scheduled scenarios execute at the scheduled time in the time zone you assigned in step 4.

Manage your user time zone

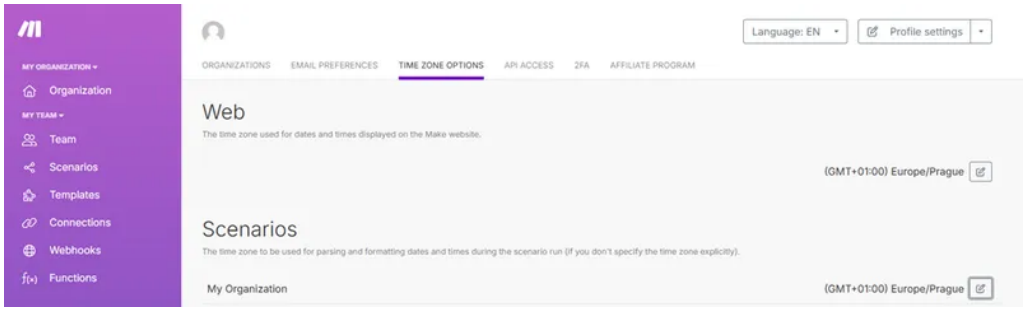
Make uses your user time zone to display time and date information. For example, execution logs display the time in your user time zone. The user time zone only impacts how Make displays information, such as the date and time displayed in module output, bundles, or a log. Changing the user time zone does not impact scenario execution. You can view and edit your user time zone anytime on the **Time Zone Options** tab of the profile page.

1 Click on your username at the bottom of the left sidebar.

2 Select **Profile**.

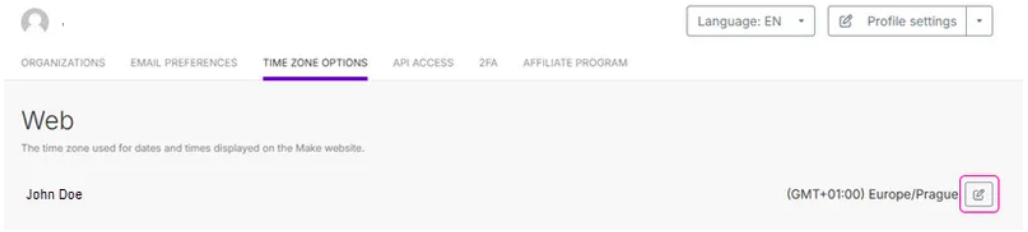


3 Go to your Profile page and click **Time Zone Options**.

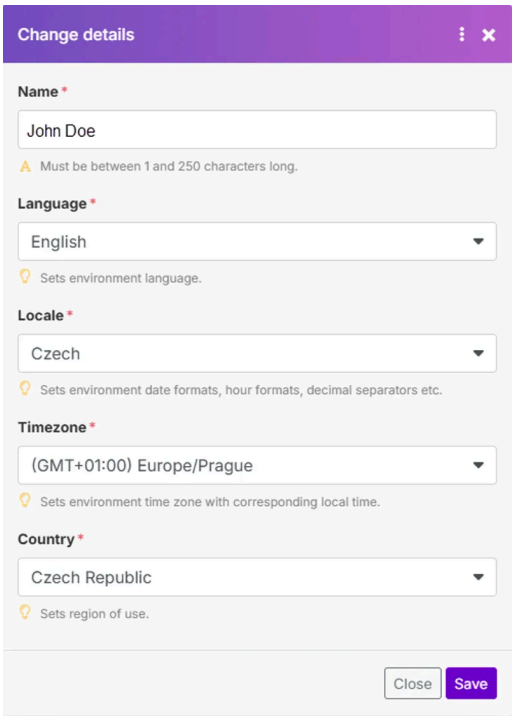


4 Find your name in the **Web** section. Make displays the current time zone setting.

5 Click the edit icon to edit your settings.



6 In the dialog box, use the menu to search for and select a new time zone.



7 Click **Save**.

The selected time zone now appears on the **Time Zone Options** page.

Example

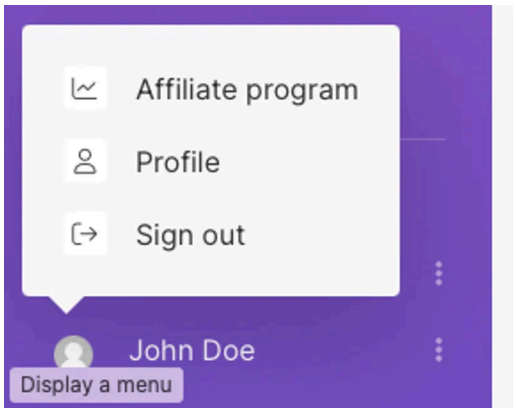
A Make organization based in New York City has members in New York City, Los Angeles, and London. When members view scenario execution histories, they see that information relative to their local time zone. Members in London see the time scenarios executed in London time, Los Angeles members see those execution times in LA time, and so on. However, when they schedule scenarios, Make uses New York City time regardless of the user's location.

User city and time zone	Organization time zone	Time zone that the user sees in the scenario editor and logs	Time zone used for scenario execution
New York (GMT-05:00) America/New_York	(GMT-05:00) America/New_York	(GMT-05:00) America/New_York	(GMT-05:00) America/New_York
Los Angeles (GMT-08:00) America/Los_Angeles	(GMT-05:00) America/New_York	(GMT-08:00) America/Los_Angeles	(GMT-05:00) America/New_York
London (GMT+0:00) Europe/London	(GMT-05:00) America/New_York	(GMT+0:00) Europe/London	(GMT-05:00) America/New_York

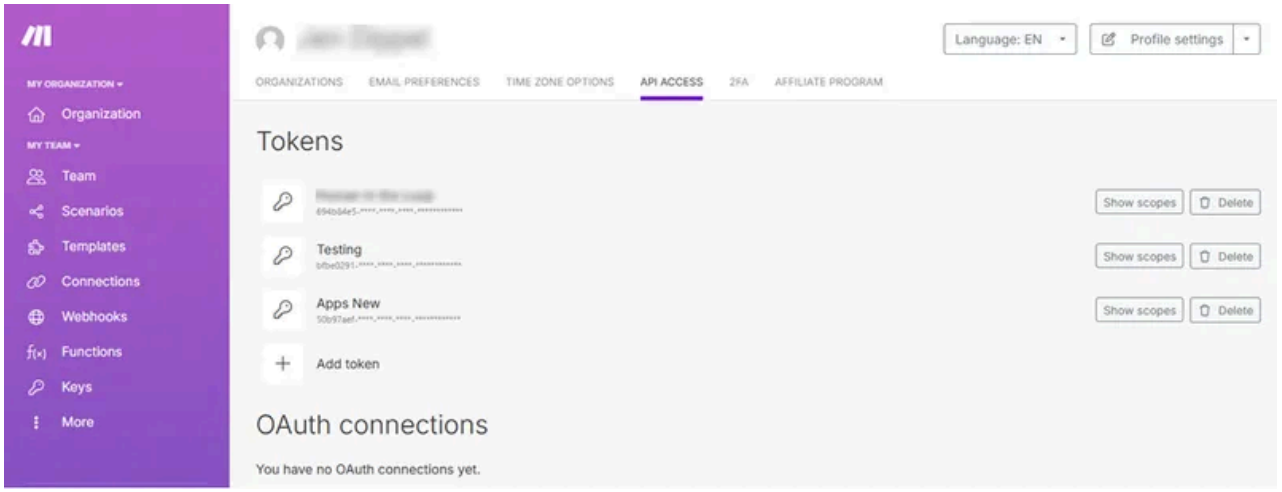
7.1.5. API key

1. Click on your username at the bottom of the left sidebar.

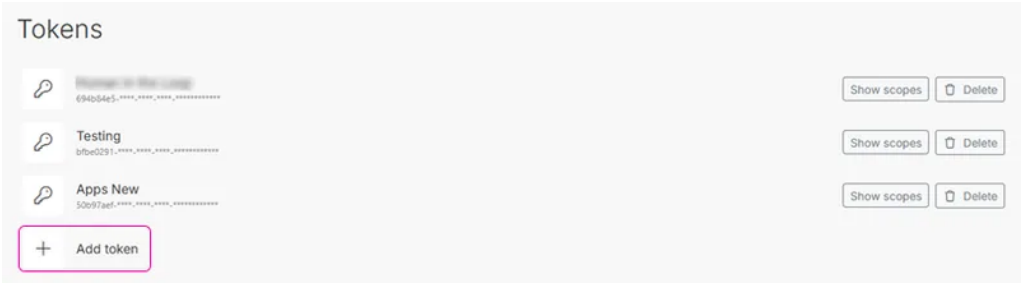
2. Select **Profile**.



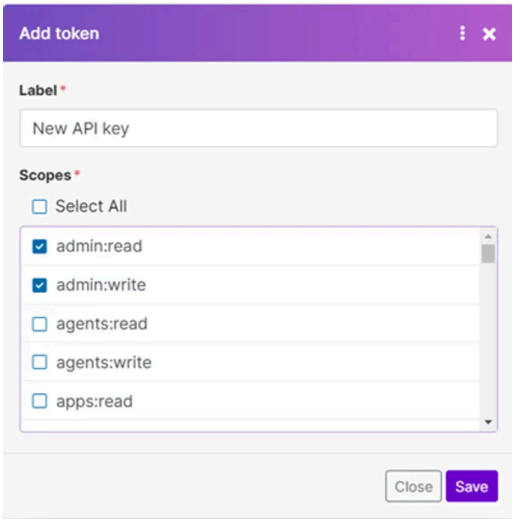
3. Click the **API Access** tab.



4. Click the **Add token** button.



5 Enter a **Label** for your token and select the necessary **Scopes**.



You can review the scopes in the **Make API documentation**.

6 Click the **Save** button to confirm your selections.

7 Copy your new API key to your clipboard and store it in a safe place. **This is the only time when you can see the whole API key.** The API key is required to set up the VS code extension.

Your Make API keys are listed in your profile in the **API Access** tab where you can view and manage your API keys.

7.1.6. Two-factor authentication

You can make your Make account more secure using two-factor authentication (2FA). 2FA means that there are two different factors (your Makecredentials and an additional verification code) needed to authenticate you as a user when logging into your Make account.

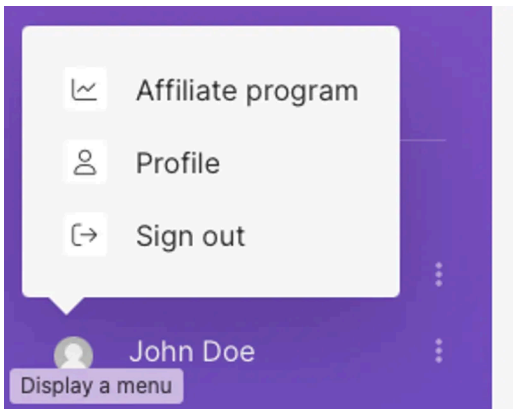
You must have an authenticator app installed on your mobile device.

Enable two-factor authentication

There are many authenticator apps available (e.g. Google Authenticator, Authy, Microsoft Authenticator, or 1password). We will use the **Google Authenticator** app in our example.

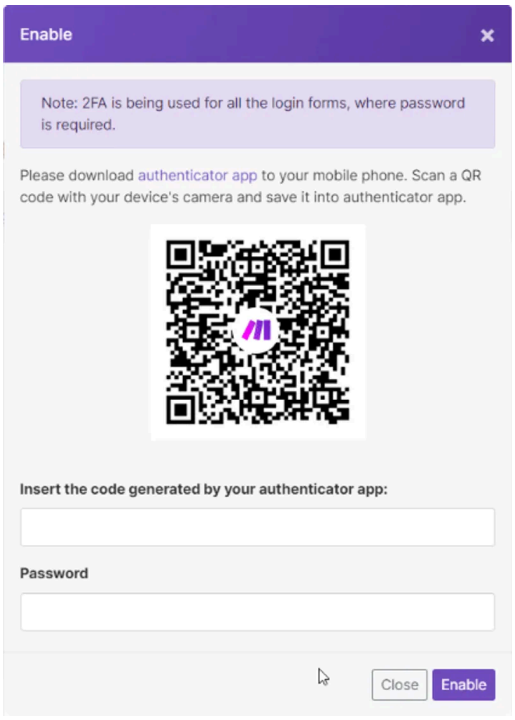
1. Click on your username at the bottom of the left sidebar.

2. Select **Profile**.



3. Click the **2FA** tab for two-factor authentication.

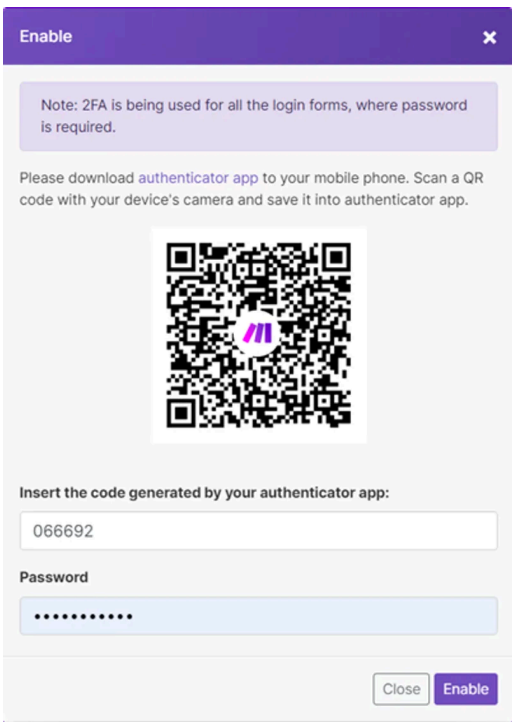
4. Click the **Enable** button to display the **Enable Two-Factor Authentication** dialog box with a QR code.



5 Install and open the **Google Authenticator** app on your mobile device (**Android**, **iOS**).

6 In the **Google Authenticator** app, click **Scan a QR code**.

7 Enter the six-digit **verification code** generated by the **Google Authenticator** app and your Make password in the **Enable Two-Factor Authentication** dialog box, and click **Enable**.



8. Click the **Download codes** button to download the TXT file containing all your one-time passwords.

> Save the provided one-time passwords in a safe place (not on your mobile device). You can use these one-time passwords to recover access to your account if your mobile device has been stolen or lost, or if the authenticator mobile app has been removed from your phone.

Sign in using two-factor authentication

After signing out, you can now log in using 2FA.

1. Go to the Make login page.
2. Enter your Make credentials and click Sign In (or sign in using Google, Facebook, or GitHub).
3. Open the authenticator app (e.g. Google Authenticator) on your mobile device to get the verification code.
4. Enter the verification code provided by the authenticator mobile app in the **2FA code** field, and click **Sign in**.

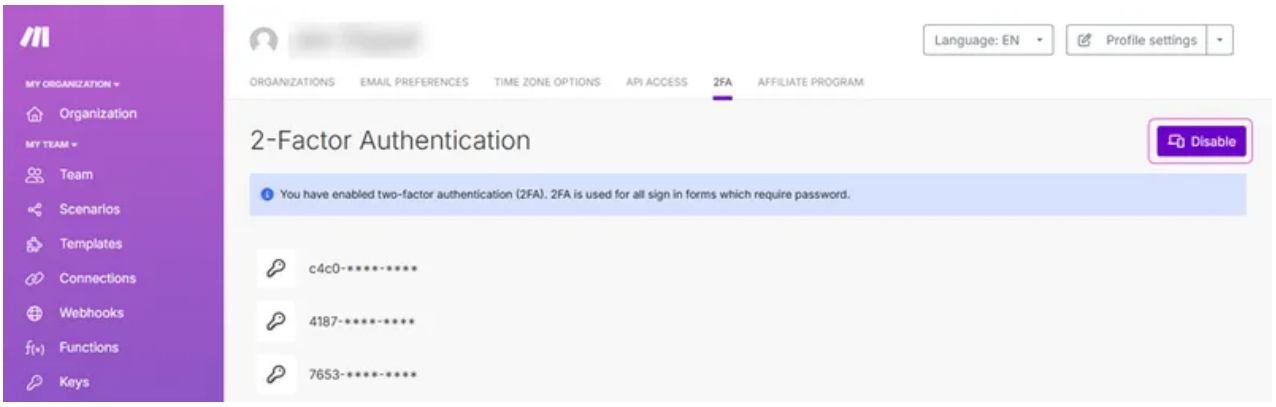
> If you have lost your mobile device, or accidentally deleted the authenticator app, you can use one of the one-time passwords you store after enabling the 2FA on your Make account (step 6 in the **How to enable two-factor authentication** section above).

Disable two-factor authentication

1. Log in to your Make account.

2. Go to your Profile page and click the **2FA** tab for two-factor authentication.

3. Click the **Disable** button, enter the verification code generated by your authenticator app on your mobile device, enter your Make password, and click **Disable**.



Two-factor authentication is now disabled on your account.

Switch to another authenticator app or change your mobile device

If you want to use another app for the two-factor authentication or another phone, you first need to disable the two-factor authentication on your Make account using your current authenticator app or one-time password.

1. **Disable the two-factor authentication** on your Make account.

2. Install another authenticator app on your mobile device (or the same app on the new mobile device).

3. **Enable the two-factor authentication** on your Make account using your newly installed authenticator app.

7.2. Make programs

Read about our Affiliate and NGO programs:

→ **Affiliate program**

→ **NGO program**

7.2.1. Affiliate program

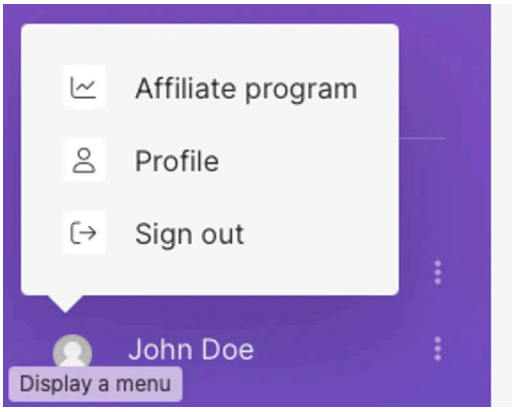
The Make affiliate program allows you to earn commission with successful Make account activation through tracked links.

When a visitor signs up using your unique referral link, we give you a commission. Anyone can join the program and start monetizing their audience through these referrals.

Register for the Make affiliate program

1. Click on your username at the bottom of the left sidebar.

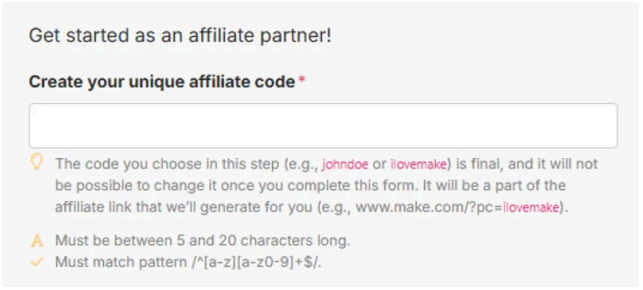
2. Select **Affiliate program**.



3. In the **Create your unique affiliate code** field, enter a code of your choice between 5 and 20 characters. For example, `johndoe`.

   Your code must contain lower case letters. It can contain numbers.

   The code you choose in this step is final and it will not be possible to change it. It will be part of the affiliate link we will generate for you.



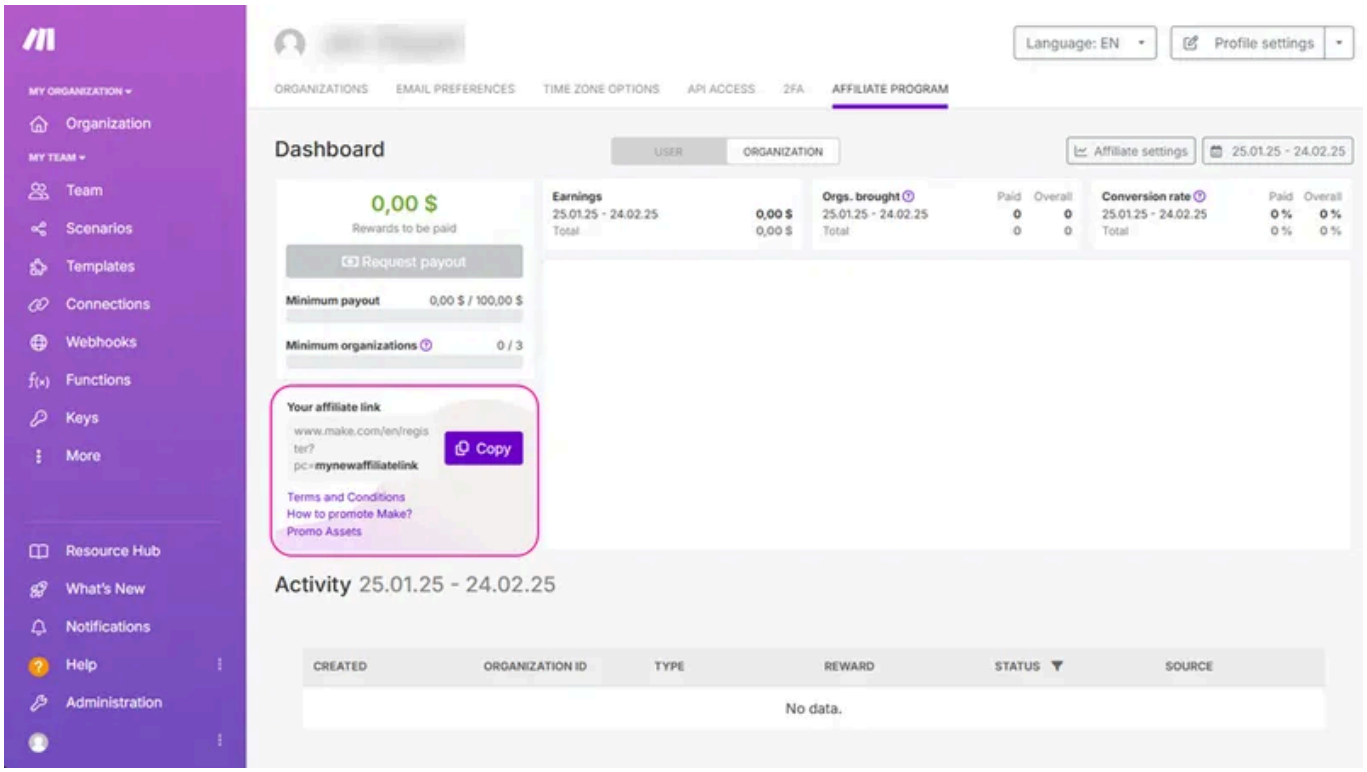
4 In the **Insert your Wise email** field, enter your Wise email.

5 Select your **Wise account type:** Person or Business.

6 Select your **Preferred payout currency**.

7 Complete the short survey, agree to the terms of service, and click **Get your affiliate link.**

You can view the affiliate link in your affiliate dashboard. By registering for the affiliate program, you can earn a 20% commission on all your referrals from your affiliate link for 24 months.



What happens when someone registers using your affiliate link?

For each visitor that signs up to Make using your affiliate link and makes a payment for a plan, you will earn 20% of their payments for the next 24 months.

The 24-month time period starts right after an organization registers with the affiliate link.

Example

An organization registers with an affiliate link on June 1, 2023.

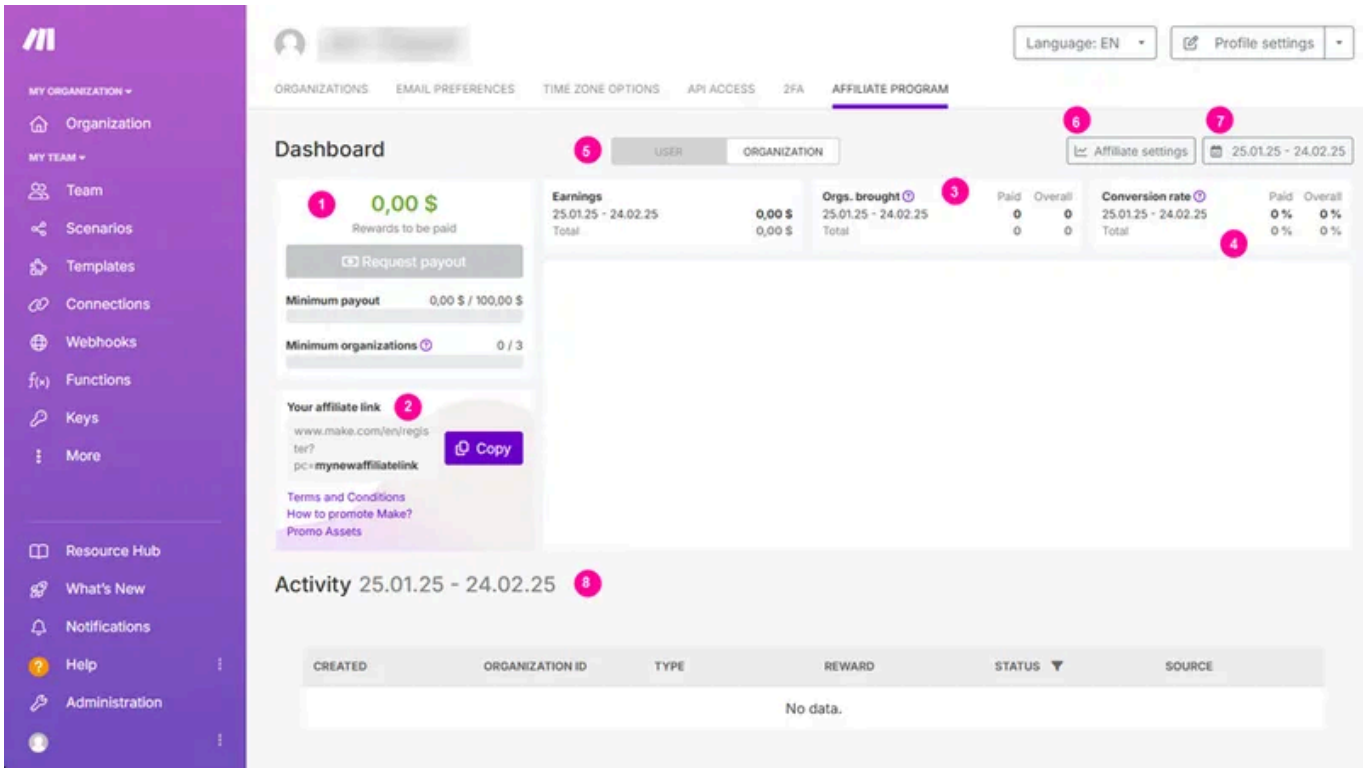
The organization starts paying Make on January 1, 2024 and they continue to pay for the next five years.

The affiliate partner will get 20% of every payment between the period of June 1, 2023 and June 1, 2025.

View your affiliate statistics and commission

You can view your affiliate link, commission information, and user statistics from the affiliate dashboard. The dashboard also lets you define specific dates to filter statistics such as registrations and rewards.

For each visitor that signs up to Make using your affiliate link and makes payments for a plan, you will earn 20% of their payments for the next 24 months.



The dashboard has the following:

1. **Earning panel**: Displays the total earnings, balance amount, minimum payout, and organizations required to qualify for receiving the commission. You can click **Request payout** to receive your commission.
2. **Your affiliate link:** You share this link with your users for registering or upgrading to Make.
3. **Orgs. brought:** Displays the number of organizations and users registered using your affiliate referral link.
4. **Conversion rate:** Displays data for both paid (registration vs. paid) and overall (clicks vs. registrations) conversion rates.
5. **User and Organization toggle**: Switches the statistics between users and organizations.
6. **Affiliate settings button:** Opens a dialogue box to change settings, such as your Wise account link.
7. **Date filter:** Defines the date range for the statistics shown on the dashboard. The default setting is for the past 30 days. You can customize this date range as needed, to analyze trends in more detail.
8. **Registrations and rewards chart:** Color-coded line graph of registrations and rewards earned.

Change the timeframe of data shown

The affiliate dashboard displays statistics for the past 30 days by default. The date filter lets you customize the range of dates for the data displayed on your dashboard. You can define both the start and end date and view the statistics for that period of time. This flexibility lets you analyze trends in more detail from your dashboard.

1. Go to your **affiliate dashboard**.
2. Click on the **Date filter**.
3. Define the **Date from** and **Date to** by clicking on the field and using the calendar to select a date.
4. Click **Apply**.

You can now see the dates in the following places on your dashboard:

- Earning panel
- Orgs brought
- Conversion rate
- Registrations and rewards chart
- Activity log

Use a shortened link to get source statistics

The Make Affiliate Program lets you use shortened links, such as TinyURL, and still get information about the source of your users and organizations. You can add the query parameter `affiliateSource` and any value that you define. Create as many short URLs as you like to get statistics for registrations through a specific website, campaign, or any other source you like.

1. Copy and paste your affiliate link into TinyURL or a similar product.

   ```
   https://www.make.com/en/register?pc=myAffiliate
   ```

2. Add `&affiliateSource=yourCustomValue` to your affiliate link.

   `https://www.make.com/en/register?pc=myAffiliate&affiliateSource=myBlog`

3. Create your shortened URL.

4. Share your shortened URL to get information for new registrations via a specific site or campaign.

Information about each specific sources appears in your Affiliate dashboard.

Receive payments

To get your first commission, you have to bring three unique customers. The minimum payout amount is $100. When an affiliate reaches $100, they will receive the payout.

Some Wise accounts have an "email discoverable" option. If you have this option for your account, it is **not mandatory** to enable this to receive your payments.

However, if you want to update this setting in your Wise account, you can take the following steps to do so:

1. Log in to your Wise account.

2. Go to **Settings >Security and Privacy > Find me by**.

3. Toggle Email to **ON** to ensure that your account is discoverable.

4. Save your settings.

View your Wise account information

Make stores your Wise account information and uses it to send your payouts. Existing affiliates can view and change their Wise account information in the affiliate dashboard. This allows you to directly update this information in case of entry errors or updates in your Wise account.

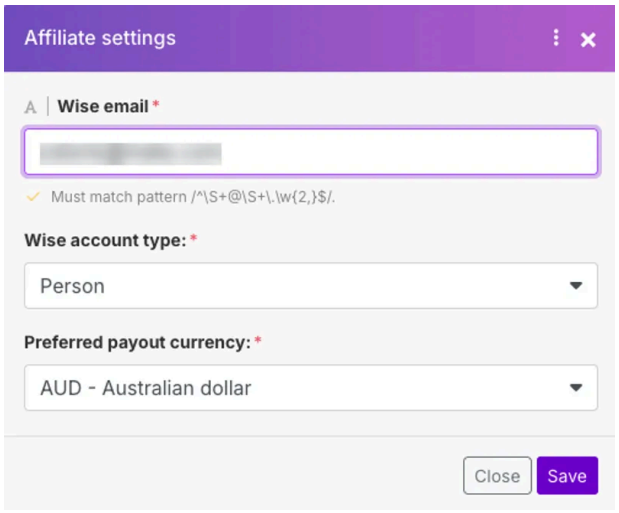
The **Affiliate settings** button opens a window where you can view and verify your Wise email. The link displayed is your current, saved information.

1. Go to the **Affiliate dashboard**.
2. Click **Affiliate settings**.
3. A window opens displaying your current Wise account information. Click **Close** to close the window without changing your saved information.

Update your Wise account information

The **Affiliate settings** window also lets you update your Wise email.

1. Go to the **Affiliate dashboard**.
2. Click **Affiliate settings**.
3. A dialog opens displaying your current Wise account information. Update your **Wise email**, **Wise account type**, and **Preferred payout currency**.



4 Click **Save**.

The dialog closes and a notification appears in the lower right confirming the change.

7.2.2. NGO program (link)

The **Make NGO program** is one of our initiatives to support causes that go beyond our own interests. We engage with NGOs from all over the world, particularly in the areas of childhood & education, health & hunger, and the earth & environment – but we are open to supporting NGOs with different focuses as well.

As part of the program, Make provides selected NGOs with a license for 12 months, with the opportunity to renew after the initial period.

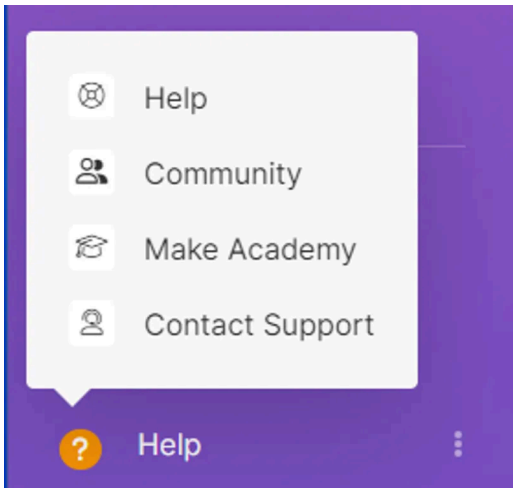
Additionally, we choose a handful of NGOs every year to participate in our high-engagement program where they get the chance to work with our employees to learn how to use Make to its maximum potential and create marketing content to inspire others to use Make.

The Make NGO program is limited to NGOs without any political or religious affiliation. Make reserves the right to change the terms of the program at its discretion.

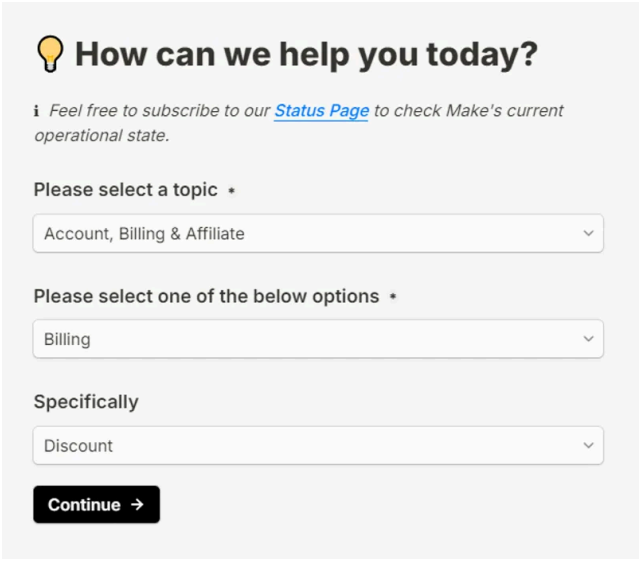
How to apply to our NGO program

You can apply through the **online form** on our website or in your Make account.

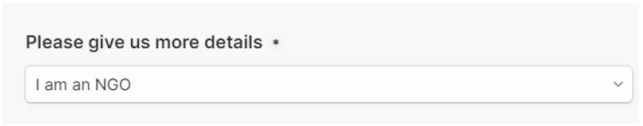
1. Log in to your Make account and click **Help** > **Contact Support**.



2. Select **Account, Billing & Affiliate** > **Billing** > **Discount** and then click **Continue.**



3 When asked for more details, select **I am an NGO.**



4 Complete the rest of the form with the required information as indicated and click **Submit**.

Your submission will be assessed by automatic review, followed by our careful selection process. The review process takes approximately five business days.

8. Release notes

→ 2025

→ 2024

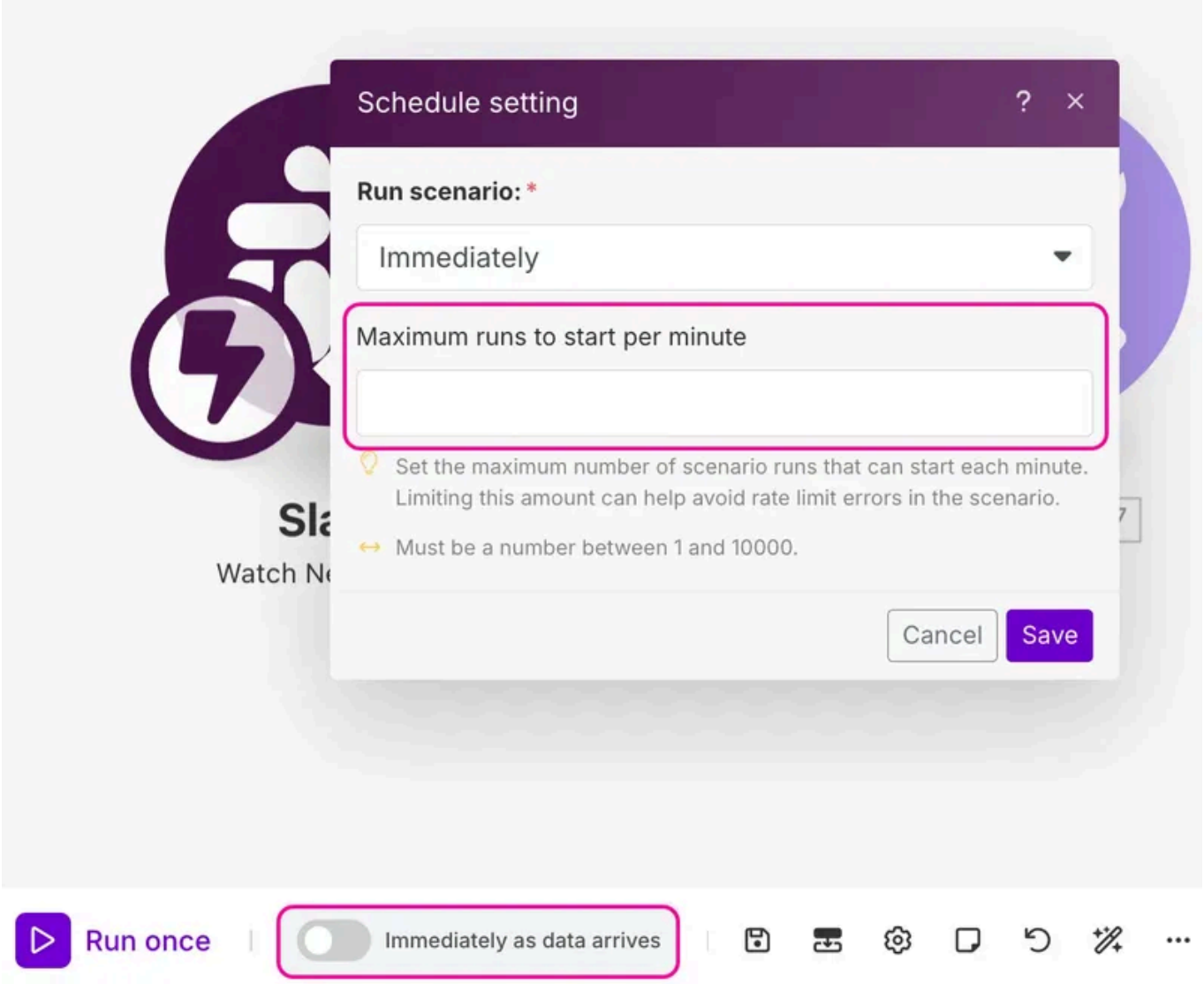
8.1. 2025

July 28, 2025	**Scenario rate limits**
July 24, 2025	**Make AI Content Extractor, new Shopify version, app updates**
June 24, 2025	**Make Grid public beta is now live!**
May 30, 2025	**Latest AI models for OpenAI and Gemini, UI updates to module rename panel**
May 20, 2025	**Make AI Tools now available in Open Beta**
May 2, 2025	**Make TypeScript SDK & App updates**
April 25, 2025	**Make AI Agents: The next step in automation**
April 17, 2025	**Affiliate payouts, incomplete execution retries, new apps**
April 3, 2025	**X (formerly Twitter) app integration discontinued**
February 21, 2025	**OAuth 2.0 integration, enhanced analytics, new apps**
February 7, 2025	**Enhanced notes & team-level operations management**
January 27, 2025	**Incomplete executions retry**
January 10, 2025	**Apps updates**

8.1.1. Scenario rate limits

Previously, scenarios with instant triggers often received rate limit errors due to exceeding third-party rate limits.

Schedule setting in the Scenario Builder now includes a **maximum runs per minute limit** to help prevent third-party rate limit errors. When a scenario reaches its configured scenario run limit, it queues and processes requests gradually as the limit allows.



Private key extraction fix

Previously, some users encountered errors when attempting to extract password-protected private keys. This issue has been resolved, and private keys can now be successfully extracted.

8.1.2. Make AI Content Extractor, new Shopify version, app updates

New apps

	Make AI Content Extractor - We've released a new built-in app that extracts structured text and metadata from files like PDFs, Word documents, images, and audio recordings—directly within your Make scenarios. The app is currently in open beta, available to all customers on paid plans, except for the Enterprise plan at the moment.
	Pictory - This AI-powered platform enables you to create and edit engaging videos quickly and easily. You can manage transcriptions, summaries, and video generation in your Pictory account using this new app.
	Postiz - This open-source social media tool lets you create, update, and delete posts, upload files, and retrieve post data from your Postiz account—directly within your scenarios.

Updated apps

	**OpenAI** - We've added new parameters to our **Create a Model Response** module and **Create a Completion** module.
	**Shopify** – We've released an updated version of the Shopify app using the GraphQL API, offering better performance and improved data handling. To learn more, see **Shopify tips and examples**.
	**Webhooks** - You can now attach multiple API keys to your custom webhook. This feature provides an extra layer of security to control access management.
	**Make** - Connections to the Make app can now be established using OAuth2 or an API key.
	**Salesforce Pardot** - It's now possible to connect using OAuth client credentials, streamlining the authentication process.
	**Calendly** - We've added a new module: **Cancel an Event**.
	**LiveChat** - A new connection procedure has been imlpemented for LiveChat, requiring your client credentials.

	**OpenAI** - We've added new parameters to our **Create a Model Response** module and **Create a Completion** module.
	**Printful** - The API key connection type has been deprecated. You can now create a connection using client credentials.

8.1.3. Make Grid public beta is now live!

June 24, 2025

We're excited to announce that **Make Grid** is now available in public beta for all paid plans!

What is Make Grid?

Make Grid is the first-of-its-kind, fully observable, auto-generated visual map that gives you a complete view of your automation and AI landscape. Think of it as your automation command center — a fully interactive world where your scenarios, apps, data stores, and AI components take form as visual objects you can explore and orchestrate in real-time.

Why Grid changes everything

As your automation grows, complexity grows with it. What starts as a few simple workflows can quickly become hundreds of moving parts across different teams and use cases. Make Grid transforms that complexity into a visual landscape you can see, understand, and control — no matter how sophisticated your system becomes.

What you can do with Grid

✨ **Visualize your entire automation ecosystem** at a glance
 Collaborate better with visual documentation that everyone can understand
 Debug faster with instant visibility into dependencies and connections
 Scale with confidence using real-time insights and monitoring

Unlike static documentation, Grid updates automatically as you build and modify your automation, so you always have an accurate view of your system.

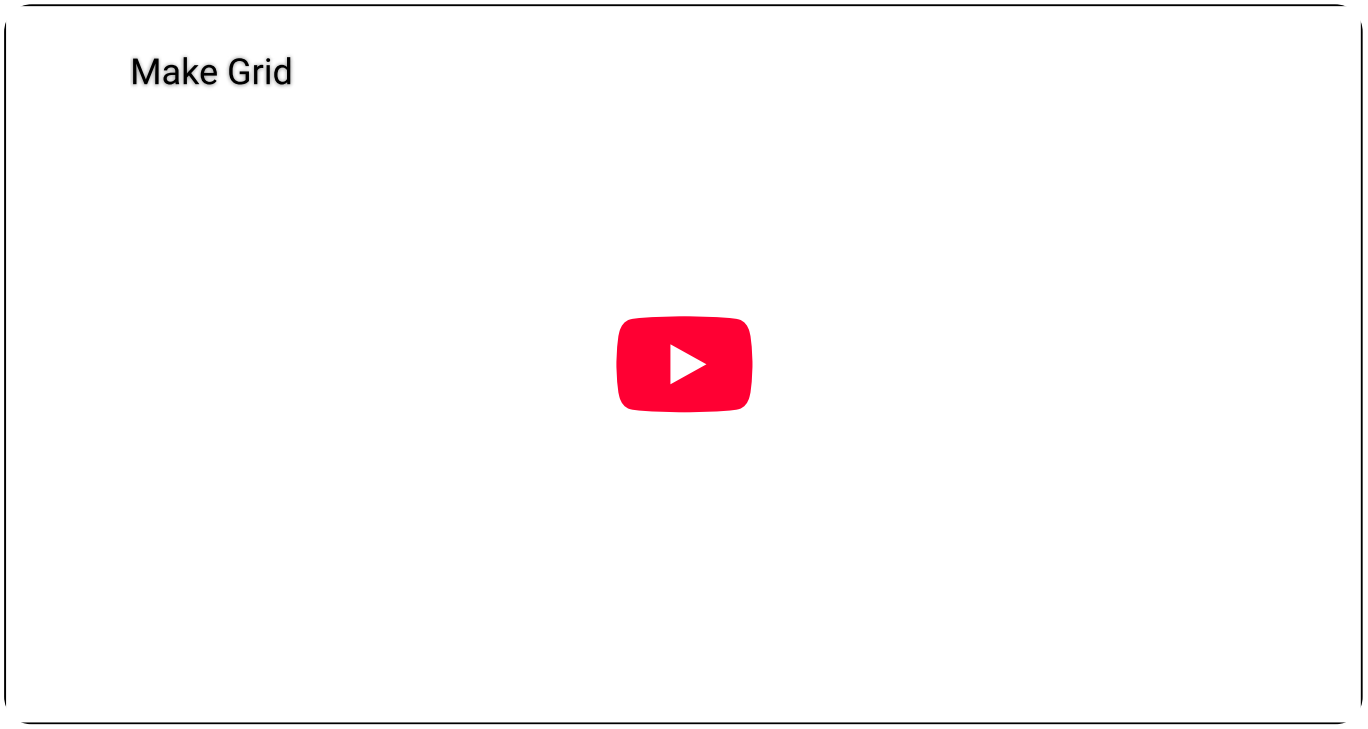
Who's it for?

Grid is designed for teams with serious scale — multiple users, dozens of scenarios, and fast-moving priorities. Whether you're a technical builder or a decision-maker steering strategy, Grid gives everyone the same clear view and the confidence to move fast.

Ready to get started?

Make Grid is available now in public beta for all paid plans. See your automation landscape like never before and discover new ways to optimize, collaborate, and scale.

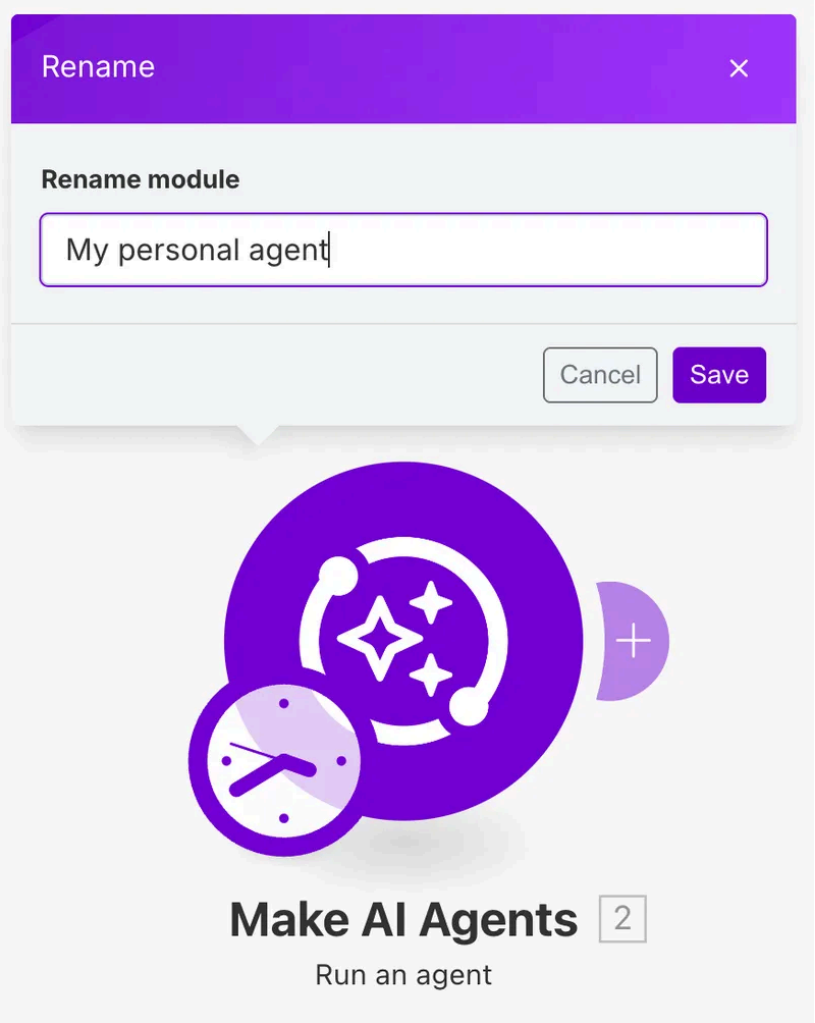
Explore Make Grid →



8.1.4. Latest AI models for OpenAI and Gemini, UI updates to module rename panel

UI updates to module rename panel

We've enhanced the module renaming experience with an updated panel design that's more intuitive and user-friendly.



Apps updates



OpenAI - You now have access to OpenAI's most advanced models:

- **GPT-4.1** - Enhanced reasoning and problem-solving capabilities
- **o4** - Optimized for complex tasks with improved accuracy



Google Vertex AI (Gemini) - New Gemini models are now available in the app:

- **Gemini 2.5 Flash** - Faster responses for everyday tasks
- **Gemini 2.5 Pro** - Advanced capabilities for complex scenarios



YouTube - We have two new modules available in the app:

- **Delete a Video** - Remove videos directly from your channel
- **Make an API Call** - Access any YouTube endpoint for custom integrations

8.1.5. Make AI Tools now available in Open Beta

> Make AI Tools is currently in beta. There may be changes to functionality or pricing.

We're excited to launch Make AI Tools for all paid plan customers. This pre-built app includes 9 AI modules that handle common tasks like sentiment analysis, text categorization, and language translation — all without needing to write prompts or set up third-party AI accounts.

The new modules include Analyze Sentiment, Categorize Text, Extract Information, Summarize Text, Translate Text, Identify Language, Standardize Text, Chunk Text, and Request Anything for custom prompts. Perfect for automating customer feedback analysis, support ticket routing, document processing, and content translation.

Technical details

Make AI Tools uses GPT-4.1 nano and includes 10 million input and 10 million output tokens per month per organization at no additional cost during the Open Beta. Your data is not stored or used for AI training — we only keep interaction logs for 30 days as part of standard Make scenario logging.

Getting started

Available now for all paid plan customers in your Make dashboard. Start with simple tasks like analyzing customer sentiment or categorizing support tickets, then expand to more complex workflows as you get comfortable with the tools.

> For detailed module descriptions, examples, and implementation guidance, see the **Make AI Tools documentation**.

8.1.6. Make TypeScript SDK & App updates

Improvements and changes

Make TypeScript SDK now available

We've added a new client library to our API documentation – the **Make TypeScript SDK**. This library provides developers with:

- Comprehensive type definitions
- Support for most Make API endpoints
- Built-in error handling
- Response typing

The SDK is fully compatible with pure JavaScript projects, allowing seamless integration regardless of your preferred development approach.

Improved collection handling

We've simplified your workflow with automatic collection conversion. When mapping a collection to a text field, Make now automatically converts it to a JSON string for you. This eliminates the manual conversion step previously required, saving you time and reducing potential errors in your scenarios.

Apps updates

New apps

	**SAP SuccessFactors** - A cloud-based human capital management software that helps companies manage employee information. You can now retrieve and search for the records in your SAP SuccessFactors account.
	**XLSX** - The XLSX app is our built-in app available on all plans. It allows you to aggregate data in an XLSX file without the need to connect to the Microsoft 365 Excel app.
	**Amazon Bedrock** - We've released the new Amazon app. You can now build AI-powered automations that create text, images, or chat responses using your Bedrock setup.
	**Clay** - The new Clay app lets you create new webhook records in your Clay tables.

Updated apps

	**OpenAI** - The OpenAI app now includes four new modules that help you manage model responses more effectively: • List Input Items • Get a Model Response • Create a Model Response • Delete a Model Response
	**xAI** - A new module called **Generate an image** is now available. It lets you create images based on a prompt.
	**Monday.com** - We have two new modules that will let you monitor the events and search items in the board by column values: • Watch Events • Search Item sin the Board by Column Values (advanced)
	**Facebook Pages** - A new module is now available that lets you publish Reels directly to your Facebook Page: • Publish a Reel
	**Crowdin** - The app now supports both OAuth and API key authentication, giving you two different ways to create a connection.

8.1.7. Make AI Agents: The next step in automation

We're excited to introduce Make AI Agents, now available on all paid plans. Transform your automation strategy with intelligent agents that work seamlessly within your existing Make scenarios.

What's new?

Make AI Agents enable you to build, customize, and manage intelligent AI workflows that automate complex business processes with adaptive decision-making capabilities.



Key features

- **Centralized management**: Create reusable agents that work across multiple workflows, reducing redundancy.
- **Flexible customization**: Maintain consistency with global system prompts while allowing scenario-specific adaptations.
- **Model selection**: Choose from various LLMs including OpenAI-compatible models to support real-time decision making.

Availability

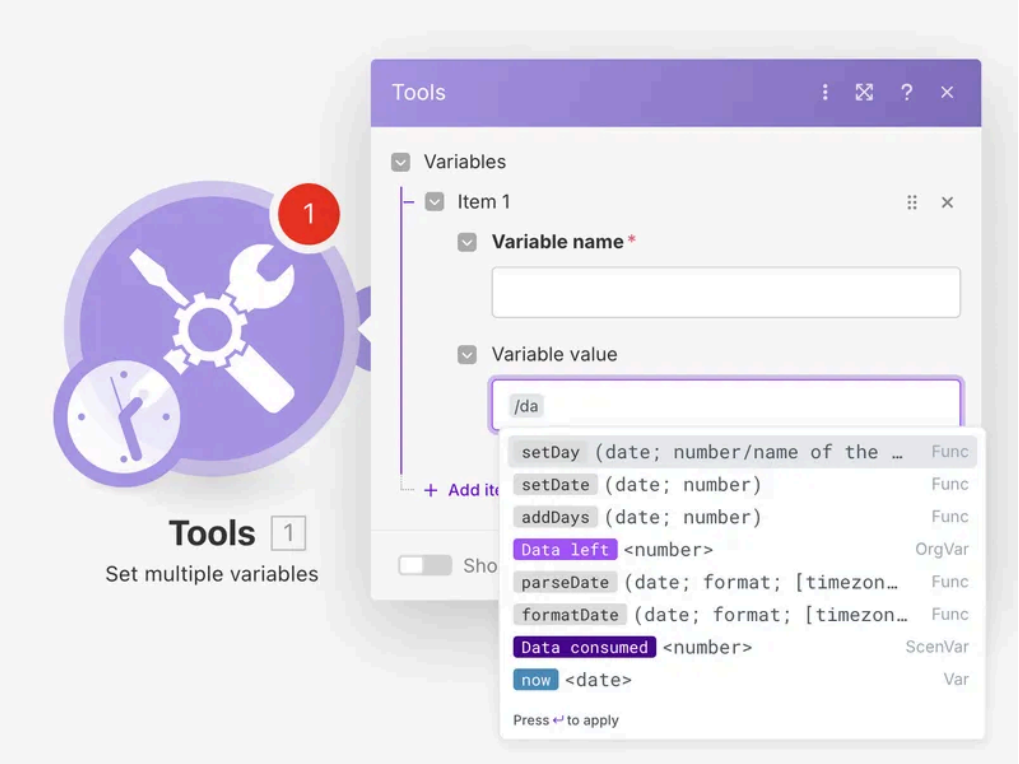
Available now for all paid Make plans.

> Learn more in our **Make AI Agents documentation**.

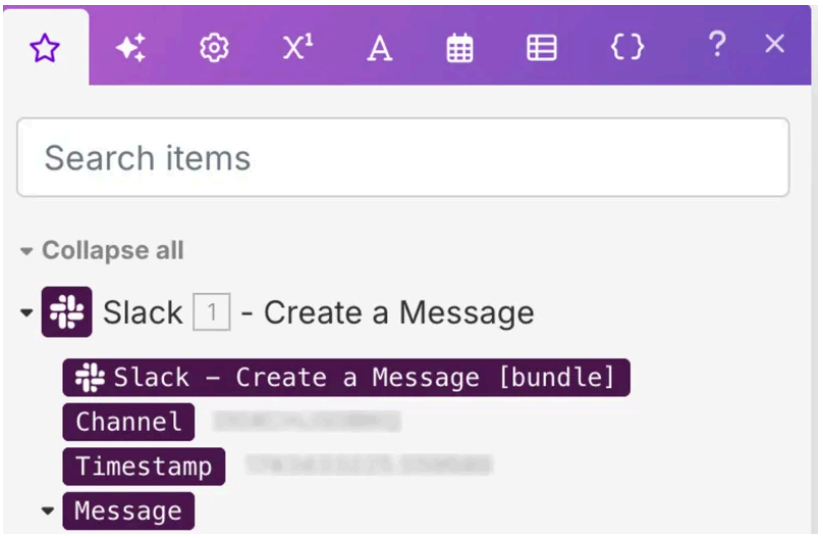
8.1.8. Affiliate payouts, incomplete execution retries, new apps

Improvements and changes

- To ensure faster and more seamless payments, Wise has replaced PayPal as our exclusive payment provider for affiliates. When you register for the **affiliate program**, you'll be asked to provide your Wise email address, account type, and preferred payout currency. If you're already part of the program, you can **update your account information** at any time.
- We've improved the way incomplete executions are retried. When retrying a large number of incomplete executions, we now limit how many are processed at the same time. This helps prevent retries from failing due to `RateLimitError` caused by too many simultaneous requests.
- You can now write `/` in mapping fields to get completion suggestions based on what you write next.



- You can now map the entire output bundle from one module to another.



Fixed issues

- Sometimes, a confirmation prompt to discard changes to scenario notes appeared even when you didn't make any. This is fixed now.
- The app search wasn't visible in rare cases when adding the first module to a new scenario. We fixed the issue so this doesn't happen anymore.

Apps updates

New apps

	Adobe Workfront - A work management platform designed to help teams plan, execute, and deliver projects efficiently. This new app lets you connect with Adobe Workfront to manage tasks, projects, issues, and users directly from Make.
	Amazon Rekognition - This new app lets you connect with Amazon Rekognition to detect objects, scenes, and faces; moderate content; and extract text from images and videos.
	Kimi - AI language model developed by Moonshot AI that allows you to create chat completions and list the available models.

Updated apps

	**OpenAI** - Now we support GPT-4.5 models. You can see the new GPT-4.5 group in the model selection dropdown.
	**Hugging Face** - Two new modules are now available: • Create a Chat Completion (Prompt) • Generate an Image
	**Twilio** - A new module, **Make an API Call for Lookup**, is now available. This module lets you send custom API requests to Twilio's Lookup API, helping you retrieve information about phone numbers - such as line type, carrier, or caller name.
	**ServiceTitan** - We've added a new connection type to reflect recent changes in the ServiceTitan app. You can now connect using either your own ServiceTitan application or the default Make ServiceTitan application.
	**Azure DevOps** – From now you can choose between **Azure DevOps** and **Azure DevOps (Azure App OAuth)** when setting up a connection, giving you more flexibility in how you authenticate.

8.1.9. X (formerly Twitter) app integration discontinued

Starting today, we are removing the X (formerly Twitter) app integration from our offering due to X's API policy requirements and pricing that prevent us from providing a reasonable integration to our customers.

Timeline

- **Effective immediately (April 3, 2025**): You will not be able to create any new scenarios using the X (formerly Twitter) integration.
- **May 30, 2025**: Your existing scenarios using X (formerly Twitter) will stop working.
- **After May 30**, you will still be able to view existing scenarios using X (formerly Twitter) in your Make account, however, these scenarios will fail upon execution.

Recommended actions

We recommend exploring alternative social media apps on Make to serve your objectives and taking necessary steps to avoid disruptions in your workflows.

Alternative options include:

- Social media management apps like Buffer and Hootsuite
- Social media platforms like Bluesky, Mastodon, and others

Frequently Asked Questions

Why is the X (formerly Twitter) app integration being decommissioned?

We are removing the X (formerly Twitter) app integration from our offering due to requirements in their API policy and pricing that prevent us from providing a reasonable integration to our customers.

Why can’t Make pay for X’s API access like other platforms?

We are unable to providing a reasonable integration to our customer based on X’s API policy requirements and pricing.

When will the X (formerly Twitter) integration be removed?

Starting from April 3, 2025, the X (formerly Twitter) app integration will no longer be accessible in our app search. Existing scenarios using the X integration will continue running until May 30, 2025, after which the scenarios will fail.

Can I still create new scenarios with X (formerly Twitter)?

No. From April 3, 2025, you will no longer be able to create scenarios using the X (formerly Twitter) app integration.

What will happen to scenarios currently leveraging the X (formerly Twitter) app integration after the decommission date?

Existing scenarios using the X (formerly Twitter) app integration will continue running until May 30, 2025, after which the scenarios will fail and return an error upon execution.

Will I have access to my X (formerly Twitter) scenarios after they stop working?

You will be able to view scenarios with the X (formerly Twitter) app after it has been decommissioned, however, your scenarios will fail and return an error upon execution.You will be able to view scenarios with the X (formerly Twitter) app after it has been decommissioned, however, your scenarios will fail and return an error upon execution.

What are alternative approaches to connecting to the X (formerly Twitter) platform?

We recommend that you please explore alternative social media apps on Make to serve your objective and take the necessary steps to avoid any disruptions in your workflows. Alternative social platform integrations available include: social media management apps like Buffer and Hootsuite, social media channels like Bluesky, Mastodon and more.

Does this impact the xAI app on Make as well?

No. At the time of this writing, X and xAI have separate API policies and requirement

8.1.10. OAuth 2.0 integration, enhanced analytics, new apps

Improvements and changes

- You can now connect third-party apps or clients with Make using the **OAuth 2.0** protocol. Check out the latest updates in our **Make API documentation update.**
- We've added more filter options to our **Analytics dashboard**, available to **Enterprise** users. Additional filters include **Status**, **Team**, and **Folder** for overall stats, with **Executions Change** and **Folder** now available in the scenario breakdown.
- Scenario execution history can now be exported into a CSV file. This is a useful option for users who want to analyze their scenario execution history in an external system. Before exporting, you can choose to hide check runs or the change log, depending on what data you want to include in the file.
- You can now rename keys without specifying their values. Previously, when renaming keys, you had to provide the current values. If you did not, the values would be replaced with blank entries.

Apps updates

New apps

	**Bluesky** - A social media platform designed for open and independent networking. This new app lets you connect with Bluesky to manage users, posts, messages, media, likes, and lists in your account.
	**ClickFunnels 2.0** - We have a new version of ClickFunnels. This version allows you to monitor contacts, orders, and form submissions, as well as manage contacts and orders in your ClickFunnels account.

Updated apps

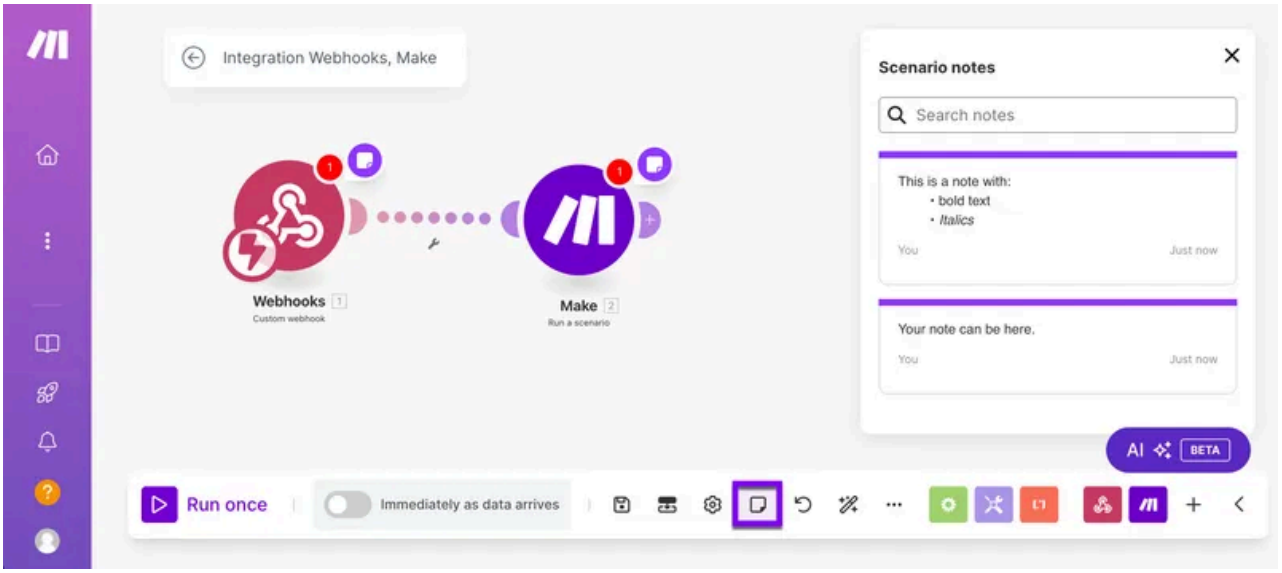
	**Google Vertex AI (Gemini)** - We have added new modules that allow you to create and process images: • Generate an Image • Generate an Image Caption • Generate an Answer for an Image
	**Runway** - In the **Generate a Video from Image(s)** module, we have added two new fields: **First URL** and **Last URL**, which require an HTTPS URL pointing to an image used to generate the video. The **Ratio** field has also updated values to select. Additionally, we have added a new field **API version** into the **Make an API Call** module.

- **Front** – We have updated the Front app logo to match the latest branding.
- **ClickFunnels Classic** - The previous version of ClickFunnels has been renamed due to the API deprecation. The new version, **ClickFunnels 2.0**, is now available.
- **Base.com** – The BaseLinker app has been updated and renamed to Base.com, along with a new logo.

8.1.11. Enhanced notes & team-level operations management

Improvements and changes

- We have improved the notes experience, making it easier and more enjoyable to create and manage notes. In addition to a fresh new look, notes now support rich text formatting. For more information, refer to our new **Scenario notes** documentation.



- Managing operations at the team level is now easier with the new **Enterprise** feature, **Operations per Team Management**. Benefits include improved control over team consumption, operational continuity, real-time monitoring of operations usage, flexibility in operations allocation, and proactive usage notifications.

Updated apps

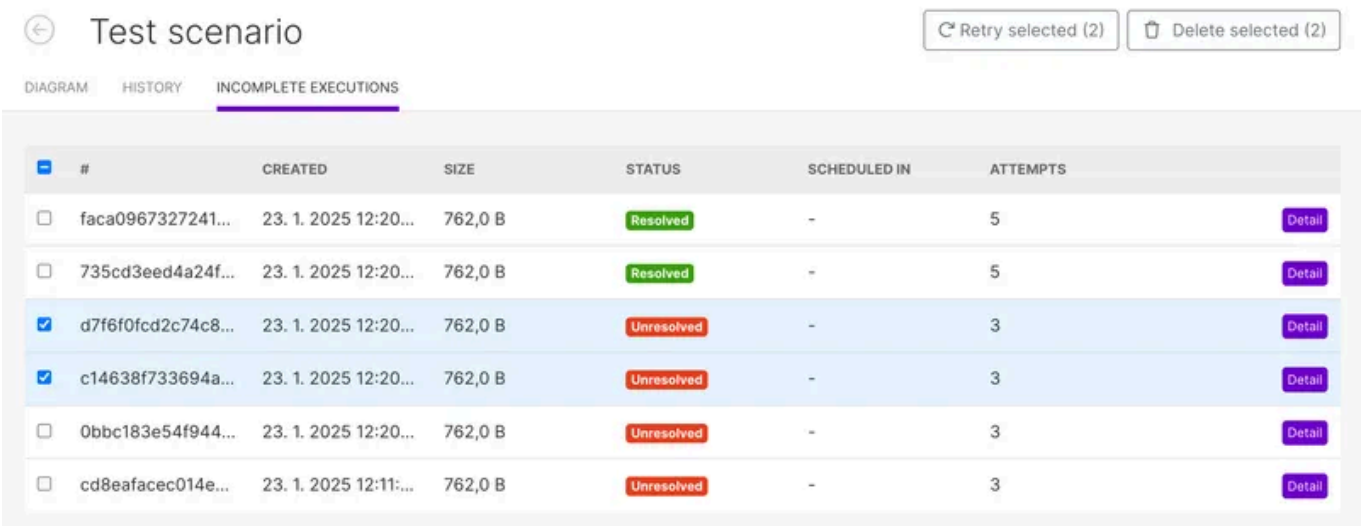
- **OpenAI** - The **GPT-4o** and **GPT-4o-mini** series of models now include a new **Predicted Outputs** parameter in the **Create a Completion** module. Additionally, **o3** models are now supported in this module, and this update has also been reflected in the documentation.
- **Workday Human Capital Management** - A new **Make a REST API Call** module has been released to enable interaction with Workday's API for greater flexibility.
- **Make** - We have released the following **Incomplete Scenario Executions** modules that allow you to manage and monitor incomplete executions effectively:
  - List scenario incomplete executions
  - Get incomplete execution detail

    - Retry incomplete executions in a scenario
    - Delete incomplete executions in a scenario
- **Workday Financial Management** - The **Make a REST API Call** module has been added, allowing users to interact with Workday's API for enhanced flexibility.
- **Salesforce** - A new module, **Make an API Call (Advanced)**, has been added, allowing for advanced API interactions with Salesforce.
- **Ablefy** - Elopage has been rebranded as Ablefy. This change has been reflected in our documentation to ensure consistency and clarity.

8.1.12. Incomplete executions retry

Improvements and changes

Users can now retry multiple incomplete executions at once. You can either select specific incomplete executions or attempt to retry all of them. You can see approximately when an incomplete execution will be rerun in the scenario incomplete executions tab.



Apps updates

We've improved the way that custom app logos are processed for more accurate representation. For more details on how single color, multicolor, and grayscale logos are processed, view the specifications and examples **here**.

8.1.13. Apps updates

New apps

- **Braze** - Braze is a platform that helps businesses deliver personalized messaging and experiences to their users across various channels. It enables brands to build stronger relationships with customers through targeted campaigns, real-time data analytics, and user insights. In Make, you can manage user profiles, catalogs, and catalog items, as well as send messages and make API calls.
- **Snapchat Campaign Management** - We now have a new Snapchat app that enables you to manage your ad accounts, ad squads, and organization members within your Snapchat Ads Manager account.

Updated apps

- **OpenAI** - We have renamed the module **Create a Completion (Prompt) (GPT-3, GPT-3.5, GPT-4)** to **Create a Completion (Prompt) (GPT and o1 Models)**.
- **Instagram for Business (Facebook login)** - We have renamed our **Instagram for Business** app to **Instagram for Business (Facebook login)** allowing you to configure Facebook authentication and access features such as managing media, posts, stories and more for Instagram Business accounts.
- **OneSignal** - The process for creating a connection with OneSignal in Make has changed. You now need to create a custom application in your OneSignal account and obtain an API key to use when setting up the connection.

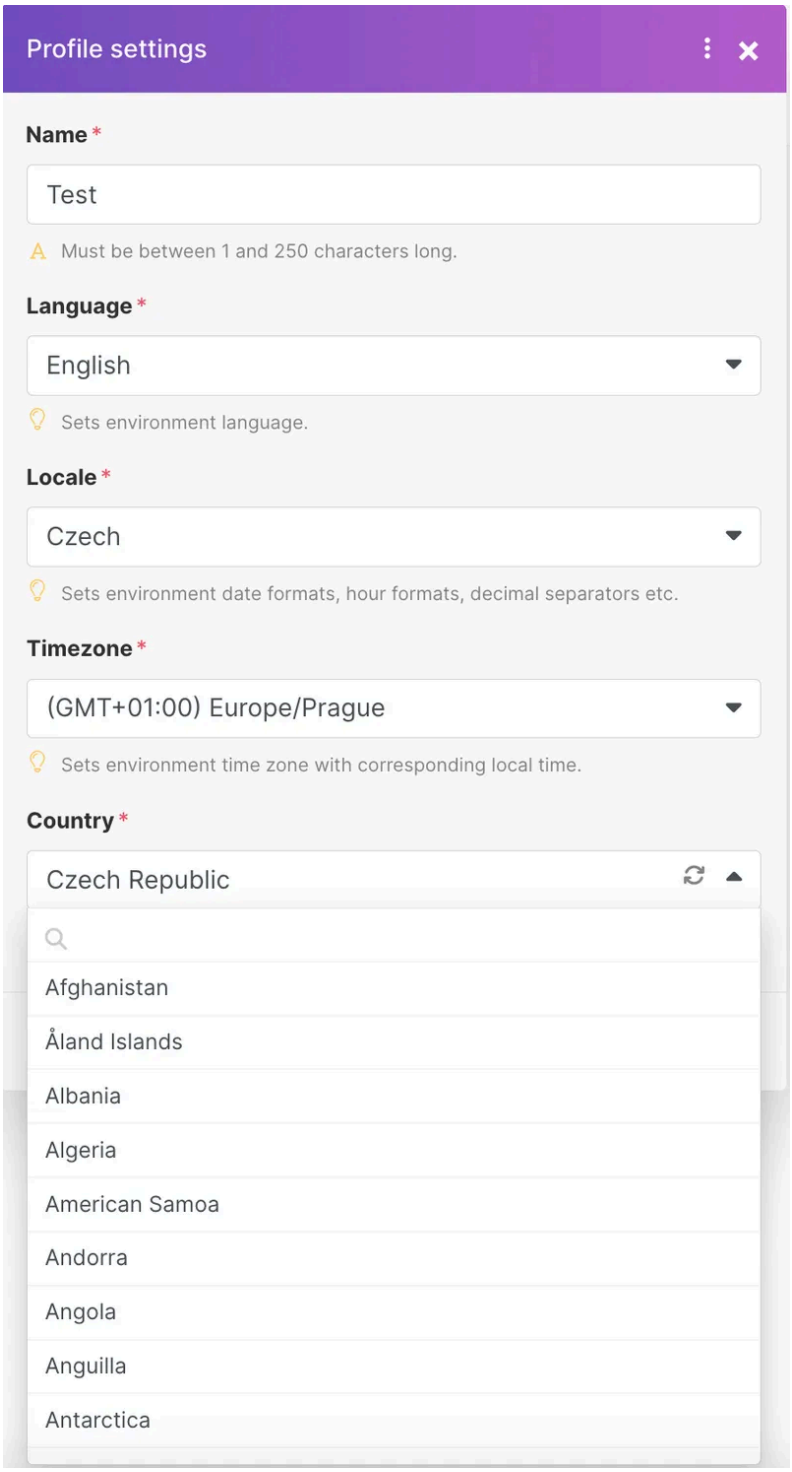
8.2. 2024

December 13, 2024	**Analytics dashboard, profile setting updates, and more**
November 29, 2024	**New Snapchat integration**
November 19, 2024	**Scenario inputs now available for all users**
November 1, 2024	
October 18, 2024	
October 5, 2024	
September 23, 2024	
September 6, 2024	
August 23, 2024	
August 16, 2024	
July 29, 2024	
July 12, 2024	
June 28, 2024	
June 14, 2024	
June 3, 2024	
May 3, 2024	
April 18, 2024	
April 5, 2024	
April 5, 2024	
March 13, 2024	
March 4, 2024	
February 12, 2024	
January 29, 2024	
January 16, 2024	
January 11, 2024	

8.2.1. Analytics dashboard, profile setting updates, and more

Improvements and changes

- Our new **Analytics dashboard**, available to **Enterprise** users, gives you a clear view of your account's key metrics. You can easily see what's happening across all of your scenario, so you can stay on top of things and make informed decisions to optimize performance.
- We've updated your options in the profile settings. Before, you were unable to change the country you first chose when registering. Now, you can change your country if necessary.



- We've updated our **downgrading and cancelation** process. Now, if you downgrade to a plan that only offers one team, or if you cancel and revert to the Free plan, you will be asked to choose the team you want to keep. You will also see a table, comparing the features of your old and new plan, before you commit to the change. For more information, refer to our **Changing your subscription** article.
- You can now schedule resolving multiple incomplete executions at once with the Make API. This feature is best for resolving incomplete executions caused by temporary errors like rate limit or connection errors, or for incomplete executions where you fixed the error already.

Apps updates

Updated apps

- **UiPath** - We've added a new connection type for UiPath. You can choose to use the original OAuth connection or the new version with client credentials.
- **Google Ads Campaign Management** - There is a new module named **Search Objects (SearchStream Query)**. Additionally, the **Search Objects** module was renamed to **Search Objects (Search Query)**.

8.2.2. New Snapchat integration

New apps

- **Snapchat Conversions** - This new app allows you to directly send conversions event data to Snapchat's servers, improving conversion tracking accuracy and enhancing campaign performance through better measurement.

Updated apps

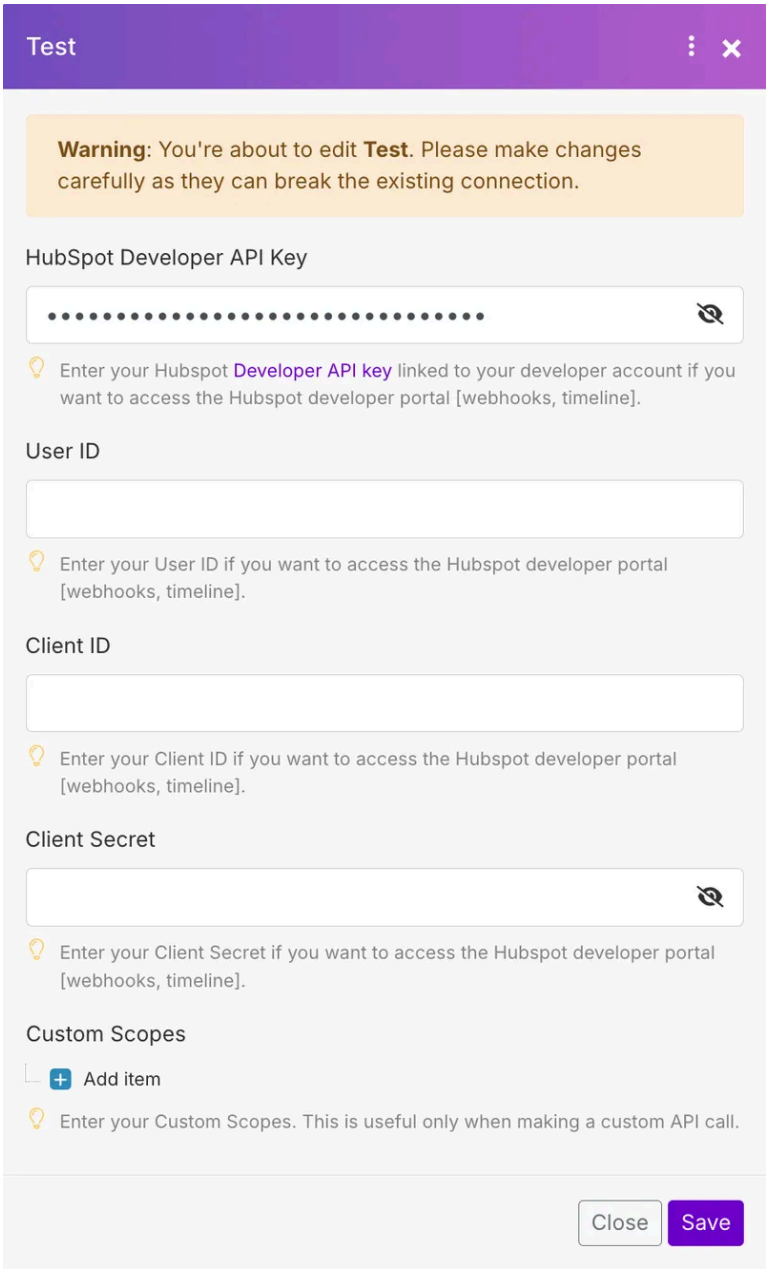
- **ServiceNow** - We have a new connection type that uses custom app client credentials. Our documentation has been updated to explain both connection types.
- **LinkedIn Conversion API** - Now you can associate multiple campaigns with a conversion rule: just add as many campaigns as you want by their IDs.
- **Klaviyo** - We removed the following modules due to API deprecation:

8.2.3. Scenario inputs now available for all users

Improvements and changes

- Users can now update their existing connections when their credentials change, instead of creating and reconfiguring new connections.

  Note: This feature is being enabled in stages. Some apps may not yet support connection updates.



- The **scenario inputs** are now available to all Make users. Previously, scenario inputs were available only to users in organizations with the **Pro** plan or higher.
- The scenario entity in the Make API now has the `isActive` parameter. This parameter shows whether a scenario **is active**. The `islinked` parameter is now Make API still returns the `islinked` parameter for backward compatibility.
- Custom app developers now have the ability to utilize data structures directly within custom apps. Users can create a data structure and integrate it into their app, similar to how data structures function in data stores.
- Previously, when a Team Admin only had an organization member role, they were unable to add new users to the team. Now, a Team Admin with a member role can see all the organization users and add users to the team.
- Custom app developers can now add a banner to the module settings. You can use banners in module settings to highlight new features or announce changes.

Fixed issues

- When parsing a date with the `parseDate` function in the timestamp format in milliseconds, users had to convert the timestamp to a number first. This is no longer required. The `parseDate` function converts timestamps in seconds or in milliseconds directly.
- Previously, attempting to add a custom property would result in an error stating that the property data already exists if there was an empty value for that property. Now, the function recognizes an empty object as a condition and allows the user to add the new property without receiving an error.

Apps updates

New apps

- **TikTok Audiences** - A new app that enables you to manage custom audiences, customer audience contacts, customer file audiences, and saved audiences directly from your TikTok Business account.
- **Lusha** - This new app helps businesses find accurate and verified contact and company information for easy prospecting and research.
- **xAI** - An app enabling users to create text completions from prompts or chats, supporting streamlined automation and enhanced workflows.

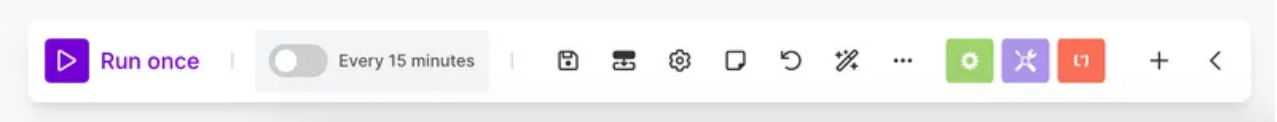
Updated apps

- **LinkedIn Conversions API** - The documentation has been updated with additional details, including Salesforce configuration guidance, required LinkedIn account permissions, and steps for setting up conversion events.
- **Snack Prompt** - We've updated the documentation to include more detailed information about the modules in the app.

8.2.4. November 1, 2024

Improvements and changes

- We revamped the toolbar in our scenario editor! We have made numerous visual and functionality improvements, like moving the Run once button into the toolbar or making the toolbar expandable. Read more about the new toolbar **here**.



- The **subscenarios** are have left the closed beta and are now available to users in organizations with the **Teams** or **Enterprise** plan.
- The custom IML function for developing custom apps has been re-enabled! It was disabled because of security reasons, and is now available again.
  We are migrating the existing custom apps to the new technology in batches, so it might take time until you can edit a custom IML function in an already existing custom app. You can read more about the re-enablement **here**.

8.2.5. October 18, 2024

Fixed issues

- We fixed a bug that was causing an error when multiple scenarios were called in an array using the scenario ID. Now, multiple scenario ID values are passed in the scenario URL.

Apps updates

New apps:

- **Schogini Image Wizard** - Easily edit images with the new Schogini Image Wizard app, allowing you to blur, resize, rotate, and more, all within Make.
- **Poper** - Watch new leads in your Poper account using the new app in Make.

Updated apps:

- **SAP ECC Agent** - We've created a custom function that lets you apply padding to any values across all SAP ECC Agent modules.
- **MailerSend** - Starting November 1, MailerSend has made changes that affect all modules. Check out the article and take the necessary steps to ensure smooth work with MailerSend.
- **Anthropic Claude** - There is a new **Make an API Call** module for Anthropic Claude.

8.2.6. October 5, 2024

Improvements and changes

- As an increased security measure, we've updated our process for deleting an account. Now, if you'd like to delete your account, you'll be asked to provide the same level of verification that is required for administering your account. For example, if you have 2FA enabled, that will be required for deleting an account.
- It's now possible to add or change your payment method on the Subscriptions page. You can still add your payment method from the Organization settings dropdown as well.
- We've added a new section in the organization dashboard: now you can monitor all your scenarios that Make automatically disabled due to some error appeared in a scenario.

Fixed issues

- Users rarely couldn't see the operation usage in their organization dashboards. This is fixed now.
- Under specific conditions, the polling trigger starting point could reset to the original setting when you created the trigger. We fixed the issue, so the polling trigger metadata are always correct.
- When the HTTP module returned invalid Unicode characters after you ran the scenario manually, you couldn't save the scenario. This is no longer happening.

Apps updates

New apps:

- **Skai** - With a new Skai app, you can configure a report and run it, as well as download a report file.

Updated apps:

- **Bigin by Zoho CRM** - Deals are now called Pipelines in Bigin by Zoho CRM. Our modules have been updated to match this change.

- **LinkedIn Offline Conversions** - The app is called **LinkedIn Conversions API** and now includes a new module, **Send Conversion Events**. Additionally, we added a new connection type that allows connecting via OpenID Connect for enhanced security.

8.2.7. September 23, 2024

Fixed issues

- Previously, unsaved filter settings didn't display correctly when you returned to finish them. This has now been fixed, and the correct filter values are shown when you come back to update a filter with unsaved changes.
- Rarely, opening a module output in scenario history resulted in an endless loading loop. We fixed the issue, so no endless loading should occur when you open module outputs.
- Sometimes incomplete executions were wrongly marked as resolved. This is now fixed.
- When you swapped an instant trigger with a polling trigger, the scenario didn't run on schedule. We fixed the issue so your scenarios run on schedule when you swap instant trigger with a polling trigger.
- In rare cases, the scenario history displayed a scenario as running even though it wasn't. We've fixed the running scenario visualization, and now the scenario status in the history shows correctly.

Apps updates

New apps:

- **Microsoft Advertising Reports** - This new app allows you to manage reports in your Microsoft Advertising account, giving you the tools to track and analyze your ad performance more effectively.

Updated apps:

- **LinkedIn Lead Forms** - We've added a helpful use case: Automated Lead Integration and Management from LinkedIn to Google Sheets.
- **Cflow** - This app allows you to automate workflows by creating and managing approval processes directly from your Cflow account, streamlining business operations.

- **SectorFlow.AI** - We've updated the documentation for this app to make it easier to navigate and find the information you need.
- **DataForSEO** - We now have a newly updated and detailed documentation for this app, offering clear guidance to help you make the most of its features.

8.2.8. September 6, 2024

Improvements and changes

- Our **DevTool**, which helps users debug their scenarios, has been rebranded and is now called **Make DevTool** in the Chrome Web Store. If you've been using it before the name change, there's no need to worry - it will continue to work as usual, just under a new name.
- Users with API access can now create up to **100 API tokens**. The former limit was 5.
- We've added new sorting options to the Scenarios list, allowing you to sort scenarios alphabetically from A to Z or by creation date, either from newest to oldest or oldest to newest. Previously, sorting was only available in the **Table** view, and scenarios in the **List** view were sorted by status (active/inactive) and then alphabetically from A to Z.
- We're excited to announce that we've added new functions!
- When creating a new module in your custom app, you now have three options to choose from:

  These options help you speed up the process of writing boilerplate code.
- User search no longer requires accents to find a match. Now, searching for a name like "Tomas" will return results for "Tomáš", "Tomás", and "Tomas", making it easier to find users with different accent marks.

Fixed issues

- Previously, some scenarios would reach their run time limit earlier than expected. This issue has now been resolved, ensuring all scenarios run within their specified time limits.
- When a scenario was cloned in the **Table** view, the **Created By** column was left empty. Now, it shows the name of the user who created the original scenario. Please note that the **Table** view is available only for **Enterprise** customers.
- The links to incomplete executions in the notification emails sometimes didn't work. This issue has been fixed, and the links now function correctly.
- When manually resolving an incomplete execution, Make didn't create a new incomplete execution if another module returned an error. This issue has been fixed,

so a new incomplete execution is created if an error occurs during the resolution process.

- Before, when a scenario was cloned, the **Created By** column was empty. Now the name of the user who created the original scenario is shown.

Apps updates

New apps:

- **Microsoft Advertising Conversions** - A new app allows managing offline and online conversions in your Microsoft Adversiting account.

Updated apps:

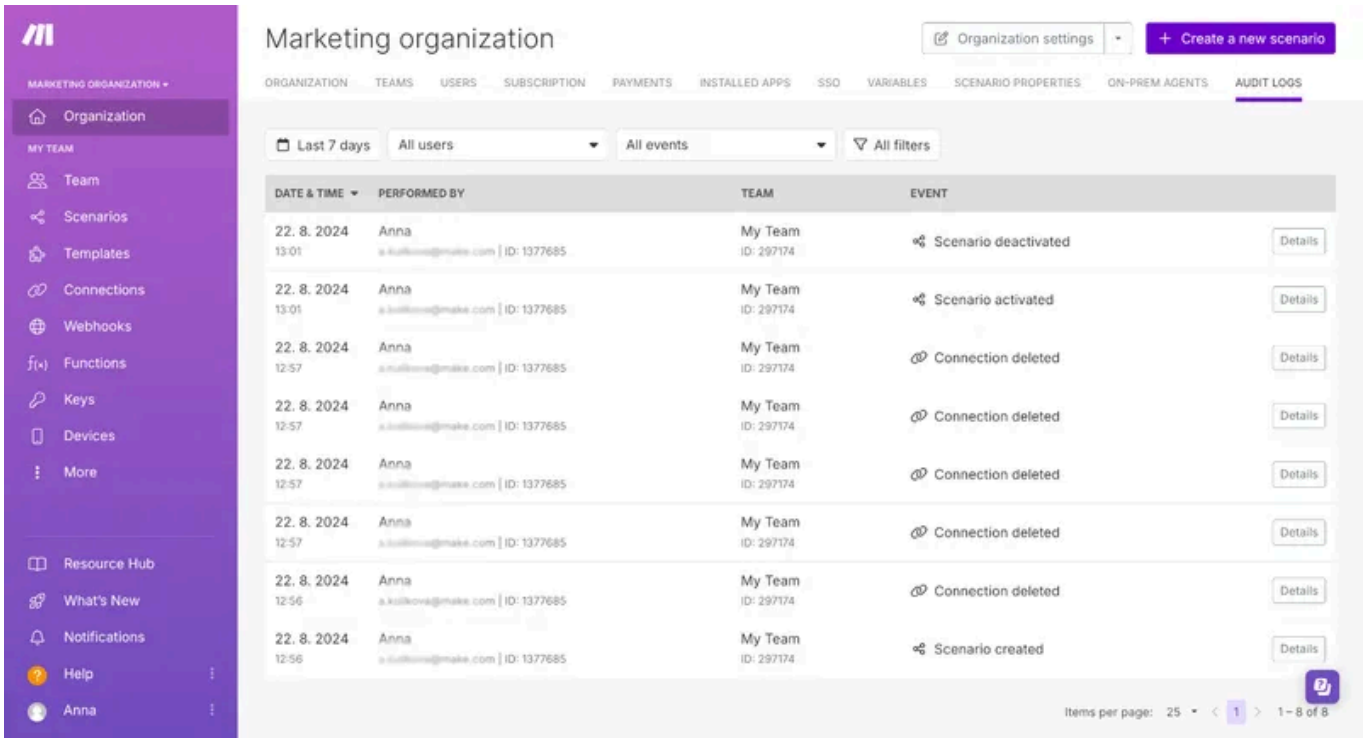
- **LinkedIn** - We have a new, secure connection method using **OpenID** that enhances privacy and security for your LinkedIn integrations. This feature is also available in certain LinkedIn apps.
- **Pipedrive CRM** - We added a couple of new modules that allows you to list deals according to different criteria as well as watch emails.
- **Adobe Acrobat Sign** - Now you can define scopes for your connection if you have a custom application in your Adobe Acrobat Sign account.
- **UiPath** - The On-prem connection type is no longer available to use.
- **Cin7 Core** - Dear Inventory has rebranded to Cin7 Core. We have updated our documentation to ensure it accurately reflects this change.
- **Active Campaign** - We added new modules for working with custom objects in your Active Campaign account: **Create or Update Custom Object Records** and **List Custom Object Records**.
- **LinkedIn Lead Forms** - Previously we had two LinkedIn Lead Forms apps for working with forms and responses. Now you don't have to split work - just use a new **LinkedIn Lead Forms** app.

8.2.9. August 23, 2024

Improvements and changes

We're excited to introduce **Audit Logs**, a powerful new feature designed to help you monitor activities within your Make account! With Audit Logs, you can:

- **Track Key Events:** Stay informed about significant events in your Organization or Team, such as scenario creation or deletion, connection updates, user removals, and more.
- **Identify User Actions:** Easily see who did what and when, as Audit Logs provide the user's name and a timestamp for each event.
- **Filter and Sort Easily:** Focus on the information that matters most by selecting a specific timeframe, filtering by event type, or tracking the actions of a particular user.
- **View Detailed Event Information:** Access comprehensive details for each event logged.



Audit Logs are available with the **Enterprise** plan, giving you enhanced visibility and control over your environment.

8.2.10. August 16, 2024

Improvements and changes

- Non-admin users can now decide to leave an organization, with the option to either keep or delete their connections. A confirmation window will appear to help them decide how to manage their connections.
- We've added helpful tooltips to the app search. Now, when you hover over the app type badge (Verified, Built-in, Community, etc.), a tooltip will appear, providing a clear explanation of what each badge represents.

Fixed issues

- Sometimes, module outputs failed to load properly. We've fixed this bug and now you can see module outputs without any issues.

Apps updates

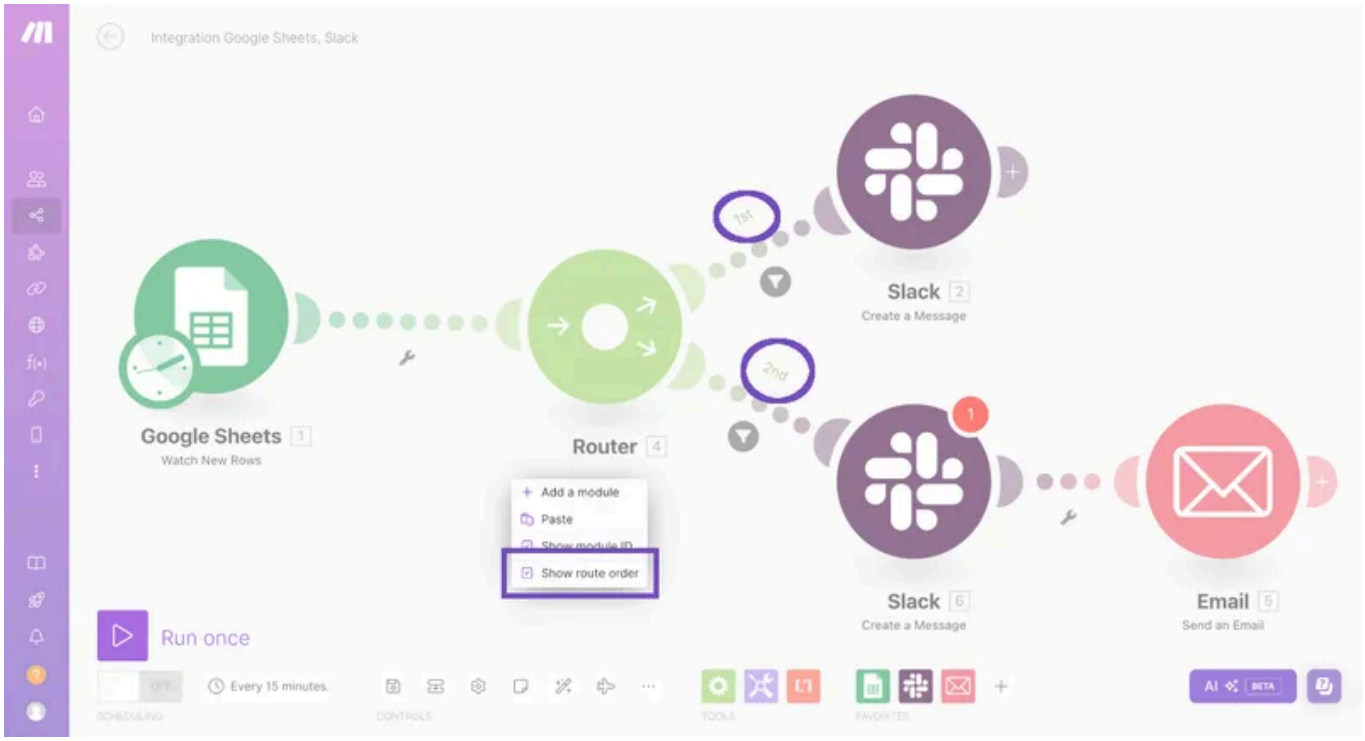
Updated apps:

- **Bird** - This app, formerly known as MessageBird, has been rebranded. We've updated our documentation to reflect this change.
- **TikTok Campaign Management** - The existing TikTok app has been renamed to TikTok Campaign Management. This app allows you to manage your campaigns and ads in your TikTok Business account efficiently.
- **BambooHR** - The API key connection method has been deprecated. Now, only the OAuth connection is available for connecting the app.
- **OpenAI** - In the **Message an Assistant** module, you can now select a tool that a model will call.
- **Shortcut** - Previously called Clubhouse, this app is now named Shortcut. Our documentation has been updated to align with the new name.
- **Ortto** - The app formerly known as Autopilot has been rebranded to Ortto. We've made updates to our documentation to reflect this rebranding.

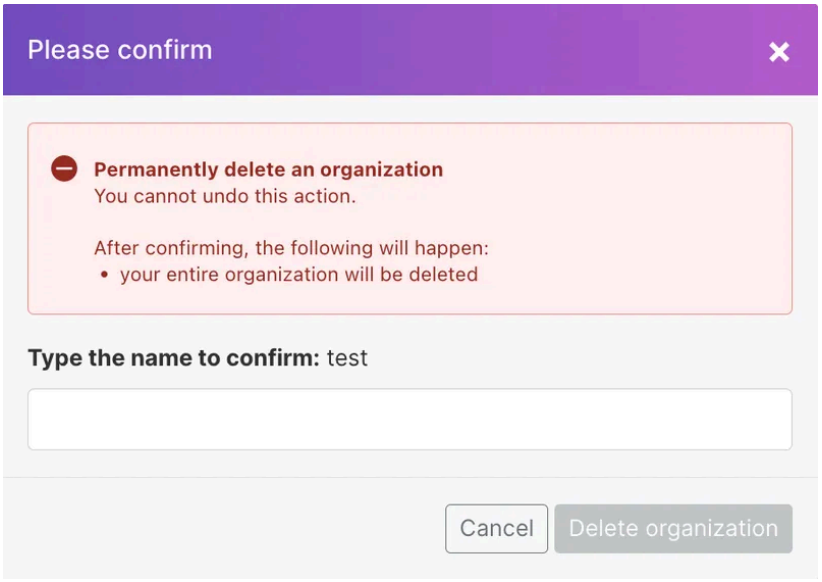
8.2.11. July 29, 2024

Improvements and changes

- Previously, the route order numbers were hidden by default when using a router in your Make scenario. We have now enabled the **Show route order** option as the default setting, making it easier to view and manage route orders.



- To help avoid accidental deletion of organizations or teams, we've added an additional step. Users now need to type the name of the organization or team to confirm the deletion.



- You can now update your keys in the **Keys** tab. To do so, navigate to the **Keys** tab, locate the key you want to update, and click the **Edit** button.
  Note that you can only overwrite the existing value with a new one. Both the previous and updated key values will not be visible when editing.
- We've added a **Remember this device** option for two-factor authentication (2FA). When enabled, this feature allows users to skip 2FA on the same device for 30 days. This option makes accessing your account faster and easier on trusted devices.
- Previously, the cloned Make scenarios were saved in the **Uncategorized** folder. Now, they stay in the same folder as the original scenario.

Fixed issues

- When using the **Search** or **Get** modules, if the processed data exceeded 3.5 MB and there were many output bundles, only the last set of bundles would be shown in the module's output. This issue has been resolved. Now, regardless of the amount of processed data or the number of output bundles, all bundles are displayed correctly in the module output.

Apps updates

New apps:

- **Canva** - This app provides an intuitive design platform that allows users to create professional-quality graphics and documents with ease, regardless of their design experience. Within Make, you can manage your design assets, folders, and other items to automate working with your projects.
- **Zoom** - Experience seamless virtual communication with high-quality video conferencing and easy screen sharing. Before Make had only Zoom for administrators, but now you can enjoy Zoom as a user: handle your webinars, meetings, chats, and cloud recordings from the user prospective.
- **Motion** - Motion is a powerful productivity app designed to streamline task and project management with automated scheduling and time-blocking features. With Make modules, you can manage your tasks, projects, and comments.

Updated apps:

- The already existing Zoom app is now called **Zoom Administration**: you can use it if you have the administrator role in the Zoom account.

- **Active Campaign** - We added several new modules such as **Watch Tasks**, **Watch Deals**, **Create a Deal** and others to let you leverage your data in a more flexible way.

8.2.12. July 12, 2024

Improvements and changes

- We made an improvement to the scenario property creation process. Now, you can add new scenario properties directly from the table view without needing to go back to the **Organization** → **Scenario properties** tab. Newly created properties are immediately visible in the table. Note that the Table view is only available for the **Enterprise** accounts.
- Previously, if a folder had a long name, it was not possible to see the full name in the left sidebar. Now, hovering over a folder with a long name displays a tooltip showing the full name.
- The scenario editor has been updated for better usability. Previously, to add a new module, you had to hover over the last module to see the **+Add another module button**. Now, the **+** button is always visible and shows a hint upon hover.
- We have released a new major version of our **Visual Studio Code** extension. The new version brings in the local development for apps, which enables you to use `git` versioning for custom app development.

  Local development for apps is currently in beta. Feel free to share your feedback!

Fixed issues

- An issue was resolved where scenarios rarely ended with an error when processing items from the webhook queue with sequential processing enabled. The processing of webhook queue items with sequential processing enabled now works as expected.

Apps updates

New apps:

- **Chatdata** - It is a platform that allows you to create AI chatbots using your chosen data or data provided by the service.
- **Iterable** - This app allows you to manage user data and engagement efficiently. With Iterable, you can manage user information, track events, and lists, improving your communication with users.

Updated apps:

- **Hubspot CRM** - There is a new module named **Create or Update a Contact**. It creates a new contact if the specified email address in the properties does not exist; otherwise, updates the existing contact.
- **Google Sheets** - We've added two new modules: **Bulk Add Rows (advanced)** and **Bulk Update Rows (advanced)** that works with multiple rows.
- **Anthropic Claude** - The **Create a Message** module has been renamed to **Create a Prompt**.
- **Facebook Conversions API for CRM** - Previously named Facebook Conversion Leads, this app has a new video to walk you through the setup process.
- **Airtable** - Four new advanced modules for bulk operations have been added to help you manage multiple records.
- **NetSuite** - There are two new modules added: **Search Records (Saved Search)** and **Watch Records (Saved Search)**.
- **Slack** - We've updated the terminology within the app for better clarity. The following changes have been made:
- **Mem** - The **Create a Mem** module has been renamed to **Create a Mem (Prompt)**.
- **Azure OpenAI** - The **Create a Completion** module has been renamed to **Create a Completion (Prompt)**.

8.2.13. June 28, 2024

Fixed issues

- We've updated how we track scenario operations. Now, the list includes operations used to resolve incomplete executions. Plus, the Make API endpoint for listing **Make operations usage** also covers these operations.

Apps updates

New apps:

- **SuiteDash** - We're excited to announce a new powerful CRM tool integration with Make! This highly requested app allows you to manage contacts and companies directly within your SuiteDash account. Now, you can automatically create tasks, projects, and update client information in SuiteDash based on triggers from other apps. Additionally, this integration ensures that data for contacts, invoices, and project statuses are synced and up-to-date across all platforms.

Updated apps:

- **YouTube** - We've added new modules that will let you manage your videos, channels, and playlists.
- **SugarCRM** - In the advanced settings section, we've added a new field API Version that will allow you to enter the version you want to work with.
- **OpenAI** - Introducing Batch Modules! Now you can work in bulk with our app's functionality. We've added five new Batch modules and an Upload a File module.
- **Smartsheet** - We've added a new Search Rows module.
- **BigQuery** - We've added a new Get Query Results by Job ID module.

8.2.14. June 14, 2024

Improvements and changes

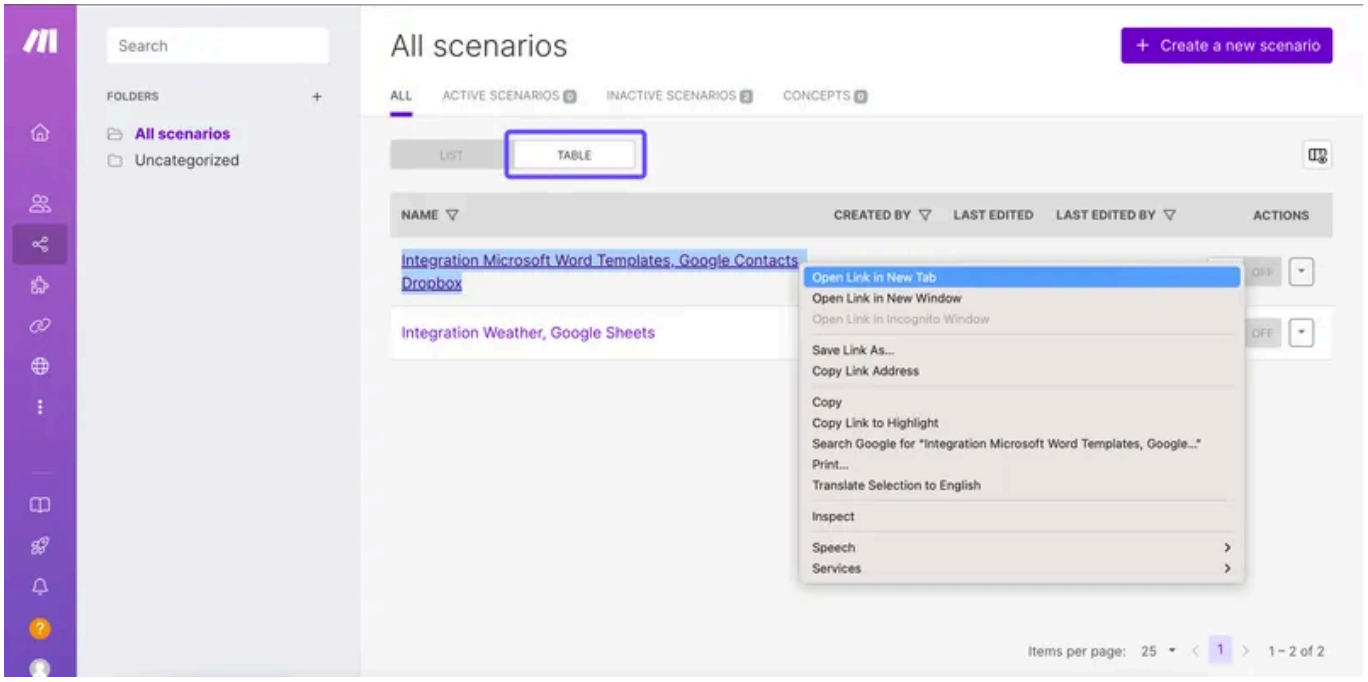
- We created new Make API endpoints to help you monitor the usage of operations in your scenario.

  Learn more about them in the **Make API documentation**.

- We updated the timezones for operations and data transfer in your organization and team dashboards. Instead of showing usage in GMT+0, the dashboards will now display data based on each user's account timezone.
- The organization and team dashboards now include data from incomplete executions, covering both operations and data transfer usage.

> If your Make scenarios frequently have incomplete executions, you may notice an increase in usage metrics on the dashboards.

- Previously, it was not possible to open a scenario in a new tab from the Table view of the **Scenarios** tab. Now, you can right-click on a scenario from the list and select the option to open it in a new tab. Note that the Table view is only available for the **Enterprise** accounts.



Fixed issues

- We resolved an issue where, in rare cases, scenarios were running indefinitely and blocking the processing of webhook queue items. With this fix, scenarios will no longer get stuck, allowing the webhook queue items to be processed smoothly.

Apps updates

New apps:

- **Sage Intacct** - This app allows you to keep accounting, staff management tools, and reports in one place. Within Make, you can handle records in your Sage Intacct account.
- **Google Ads Reports** - This app is an integral part of Google Ads that is responsible for marketing reports based on data in your Google Ads account.

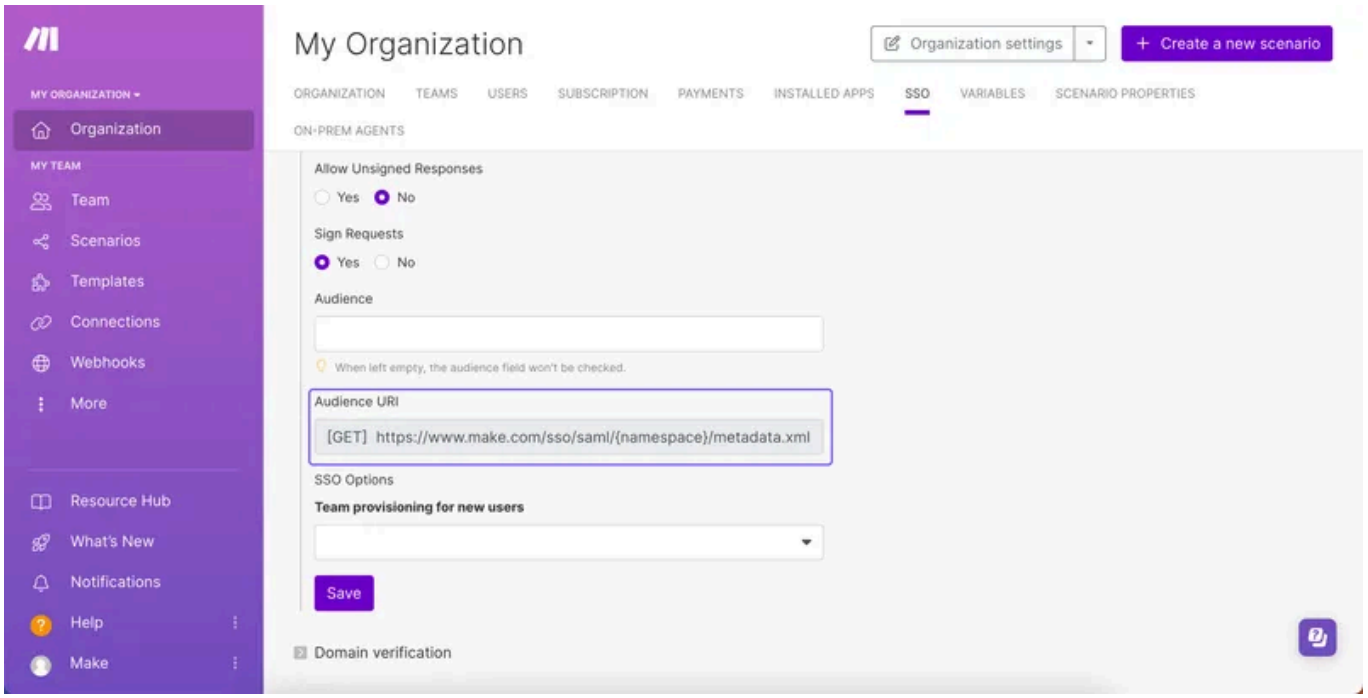
Updated apps:

- **Celoxis** - The API URL for Celoxis has been updated, ensuring all app parts and documentation reflect this mandatory change.
- **Custify** - The app documentation has been updated with revised module names and improved structure.

8.2.15. June 3, 2024

Improvements and changes

- We added a new read-only **Audience URI** field to the SSO settings, where you can find the path to the `metadata.xml`. This can be useful for setting up SSO on the provider's side. Also, we included a link to the documentation in the SSO settings so that users can access the information more easily.



Apps updates

New apps:

- **Crossbeam** - This app helps you find potential partnerships among your customers and identify account overlap to uncover new leads and boost your sales.
- **watsonx.ai** - This AI app lets you generate responses to a prompt using a variety of large language models (LLMs).

Updated apps:

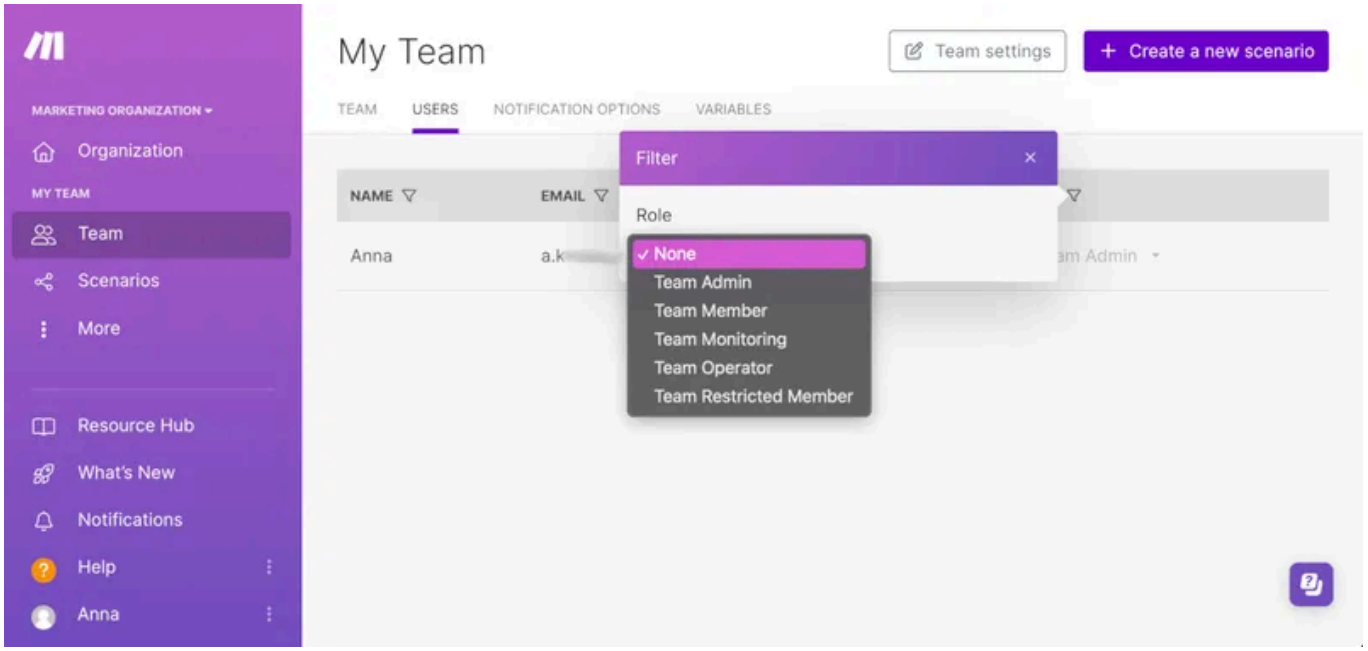
- **Firebase Cloud Messaging** - It was required to enter additional credentials to create a connection before. Now you can just sign in to your Google account to work with Firebase Cloud Messaging and enjoy!

- **Stripe** - In the beginning of June, Stripe will deprecate all API keys that you used to create a connection in Make. To prepare for this, we created a new connection type: Restricted API key. Don’t forget to switch to the new connection type to continue working with Stripe!
- **Google Chrome (v2)** - We’ve released version 2 of our Chrome app to align with the new API, so you can continue to send notifications through your browser.

8.2.16. May 3, 2024

Improvements and changes

- We've added a small but important improvement to the **Users** tab. Now you can filter users by their role at both organization and team levels. Simply click on the filter icon and select the role that you would like to see in the list.



Fixed issues

- Recently, when searching for modules in the scenario editor, the **Instant** tag was missing from instant triggers and they were mistakenly labeled as **ACID** (polling) triggers. This issue has been resolved, and it now works as expected.

Apps updates

New apps:

- Google Search Console - This app provides additional online marketing support to monitor and troubleshoot your website's Google Search results.
- Flodesk - This app allows users to create visually engaging emails for designing and managing email marketing campaigns.

8.2.17. April 18, 2024

Fixed issues

- When rerunning an incomplete execution caused by an error in a route in a router, only the route where the error occurred would be executed while resolving the incomplete executions. Any subsequent routes would be ignored. Now, all subsequent routes are also processed when resolving incomplete executions.
- Previously, custom functions couldn't handle parameters with binary data types, such as email attachments or pictures. This caused failures when trying to process files with custom functions. We fixed it and now you can successfully use custom functions to process raw binary data.

Apps updates

New apps:

- **TikTok Conversions** - Introducing an app that allows you to track conversion events that occur outside of TikTok and optimize your ads accordingly.
- **TikTok Reports** - With this app you can create detailed reports on the performance of your TikTok advertising campaigns.
- **BigCommerce** - Using this app you can manage your BigCommerce orders, carts, customers, products, and blog posts.
- **Azure OpenAI** - This AI app lets you generate images and create transcriptions, translations, and chat completions.

Updated apps:

- **LinkedIn** - A new module, **Search Organizations**, has been added to help you easily locate organizations by vanity name or email domain.
- **LinkedIn Matched Audiences** - Four modules have been deprecated due to a LinkedIn API change: **Attach a Seed Audience**, **Create a Lookalike Audience**, **List Lookalike Audience's Seeds**, and **Update a Campaign Target**.

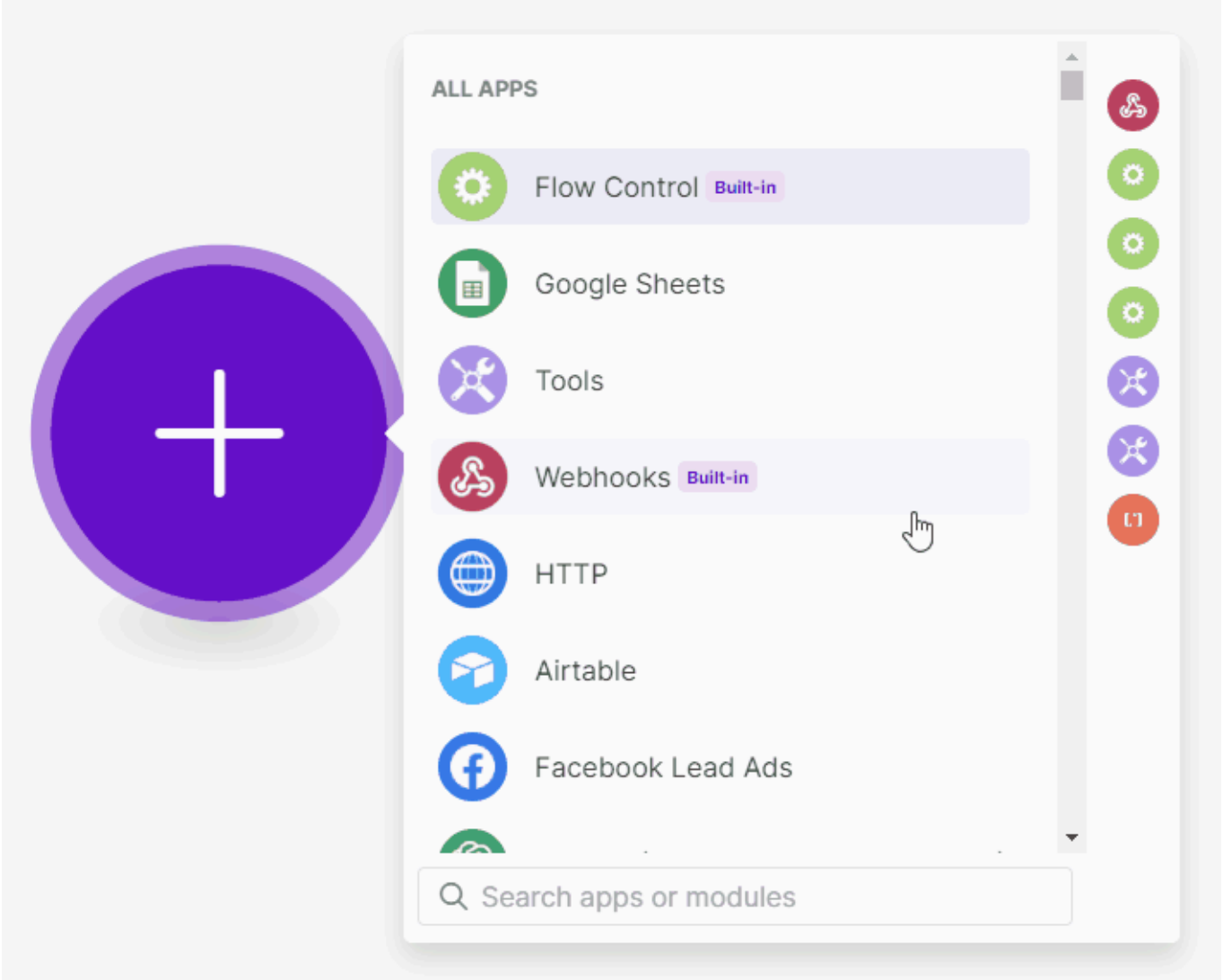
8.2.18. April 5, 2024

Improved app search

We're excited to introduce our improved in-product app search designed to help you discover apps and build scenarios faster!

What’s new:

- Improved search speed and reliability.
- Apps related to your search query show up in the app search window with added insight into available modules.
- Your recently used apps move to the top for a speedy build.
- The new right-hand bar gives you quick access to common tools and functions.
- While creating or editing the scenario you see a new section in the list of apps called **Apps in scenario**. It helps you get quicker access to already used apps.

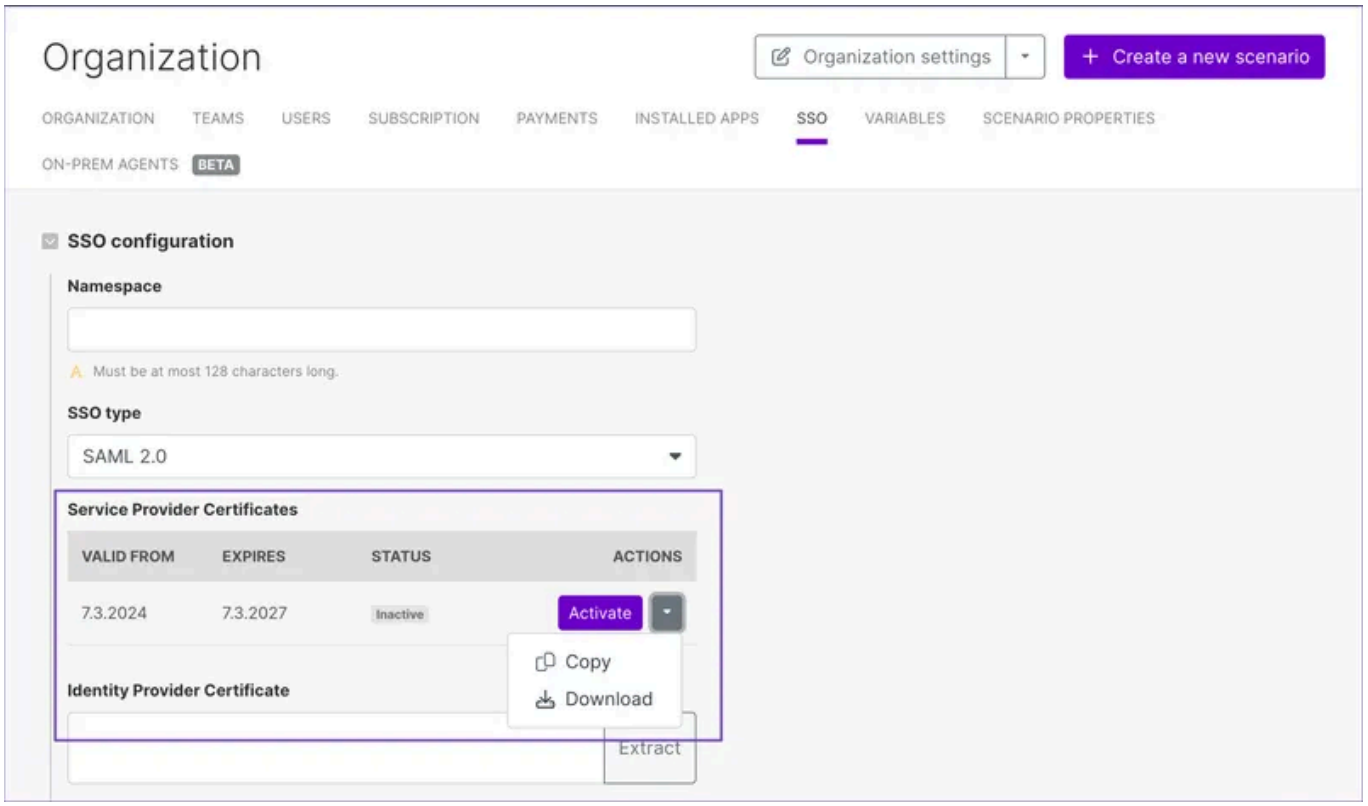


If you want to learn more or leave feedback for this feature go to **this Community post**.

8.2.19. April 2, 2024

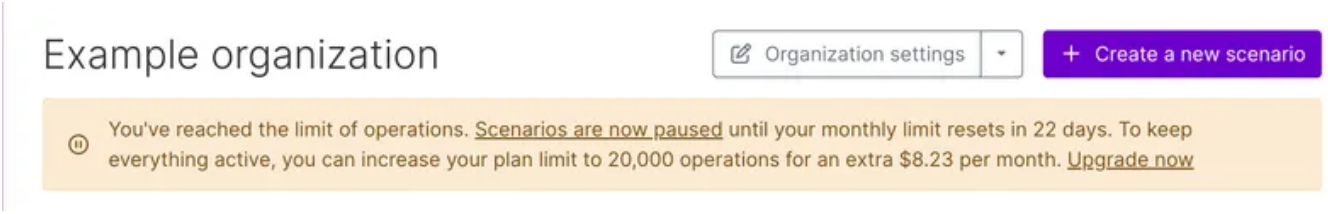
Improvements and changes

- **Managing and rotating** SAML service provider (SP) certificates became easier for the Enterprise customers. Organization owners can now activate, deactivate, copy, and download SP certificates within the SSO settings. They can also see which certificates are active. When it's time to rotate certificates, users will receive a notification from Make, giving them time to schedule the switch and minimize downtime.
  Additionally, we've extended the validity of SP certificates to three years, instead of just one.



To learn more, check out our **SAML certificate management** article.

- We've updated the banners that let you know when you've reached your operations limit. Now, when you see one of these banners, you'll get all the details about your upgrade options right there. If you decide to upgrade or purchase more operations, you can just click the link in the banner.



Fixed issues

- Make built-in functions that use timezone information, like the `addDays` function, didn't work in custom functions. We fixed the custom function's context, so now you can use the date functions in your custom functions.

Apps updates

New and updated apps:

- **OpenAI** - We've added some new functionality to our OpenAI app. Not only can you generate an image using OpenAI modules, now you can also edit or extend an image with the new module called **Edit an Image**. Don't forget to include transparency in the image file, or a mask layer, to make sure that the image can be edited later.
- **Anthropic Claude** - We've added a new module called **Create a Message**. Also, we now support the latest model Claude 3 Haiku.
- **ZoomInfo** - This new app lets you manage companies, contacts, scoops, and more.
- **Confluence** - Our new productivity app will help you keep track of all your Confluence content, including pages and blog posts.
- **Deel** - We switched to the newest API version. Make sure to use the correct URL with the new API version when using the **Make an API Call** module. Also, the **Create a Record** module now allows creating the *Global Payroll Contract* and *Contact for Contractors* types of records.
- **Tiktok Lead Forms** - This new app with an instant trigger allows you to trigger a scenario whenever you receive a new lead.
- **Cohere** - With this new AI app, you can create conversations and classify texts.
- **X (formerly Twitter)** - Our app has a new name to match the recent rebranding.

8.2.20. March 13, 2024

Improvements and changes

- Previously, you could only filter and sort scenarios in the Table view. But now, you can also search for specific scenarios using the search bar in the upper left corner. This option is only available to Enterprise customers.
- The template guided setup makes sure you don't miss any important fields when setting up each module:

Fixed issues

- Clicking the **Details** button in the webhook queue resulted in an error. We fixed it, so you can see all the data that webhooks received.

Apps updates

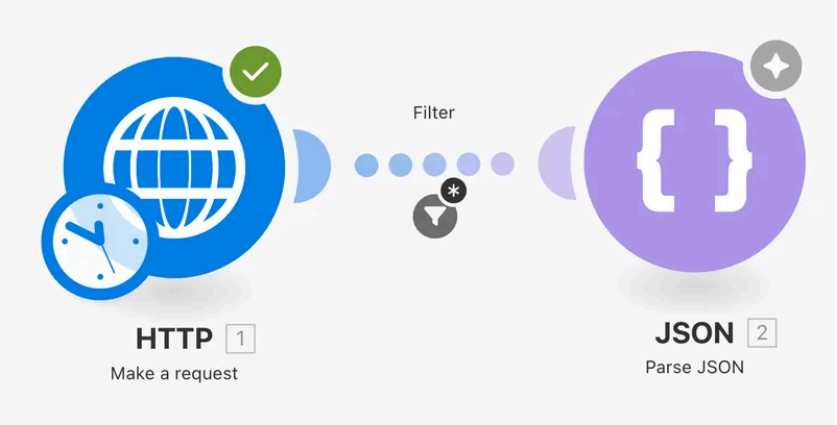
New and updated apps:

- **Salesforce Pardot** - Salesforce Pardot is a marketing platform for email campaigns and lead management. Our app allows users to manage their records, files, and folders in their Pardot account.
- **HighLevel LeadConnector** - A new **Watch Events** module was added. It can be used as an alternative to **Watch Contact** and **Watch Opportunities**. Also we switched to API v.2 for this app, making sure that the overall experience is better.

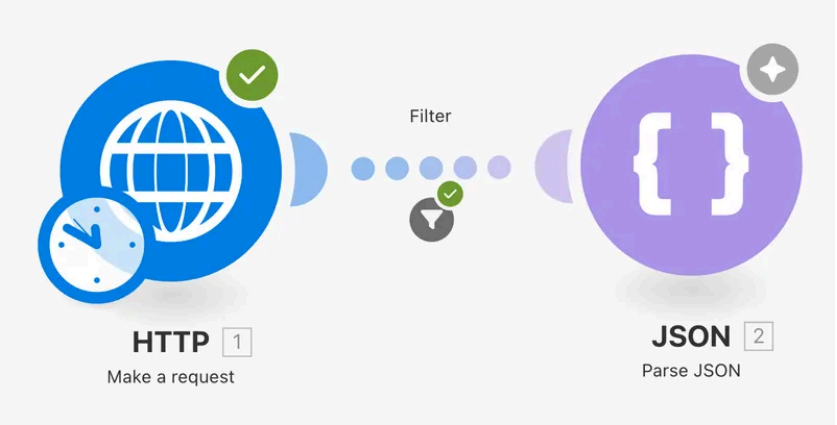
8.2.21. March 4, 2024

Improvements and changes

- During guided template setup, filters now have setup status badges:
  - Your filter has unsaved changes:



  - Your filter is set up and ready:



Fixed issues

- Custom functions sometimes took too long to finish, which caused an error and disabled your scenario. We reviewed the timeout settings on custom functions. Before using custom functions, check the **limits**.
- Webhooks migrated from the Integromat platform sometimes returned a `ConnectionError`. We fixed the issue so your legacy webhooks remain stable.
- In some cases, domain verification did not work for SSO domain claim. Verification now works as expected.
- Sharable links to templates went to the organization dashboard. These links now go directly to the specific template.

- Switching from one team to another sometimes resulted in an error message. Navigating to a different team now works just fine.

Apps updates

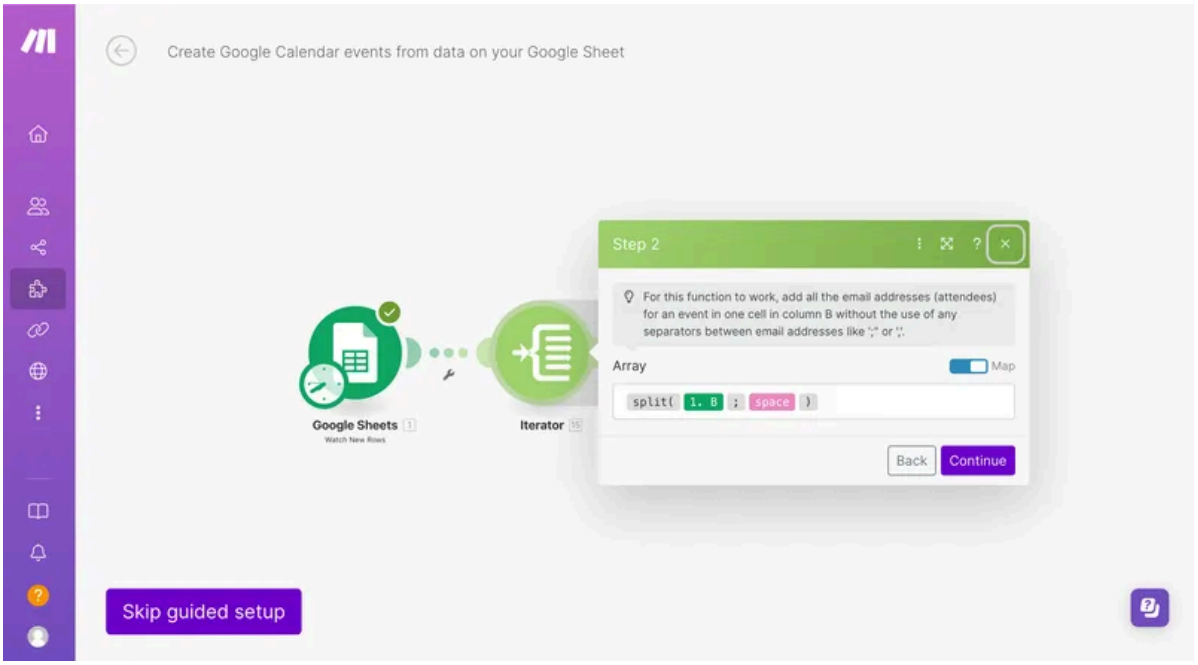
New and updated apps:

- **Leonardo.AI** - This new app helps you generate images using the latest AI art generator, Leonardo AI, a popular alternative to Midjourney.
- **TickTick** - Manage your to-do lists and tasks with this helpful productivity app.
- **Google Vertex AI (Gemini)** - A powerful new app that can answer anything with the **Create a Completion** module using the gemini-pro model, as well as analyze image and video files with the gemini-pro-vision model. This app is a replacement for the recently deprecated Google Cloud Vision app.
- **NetSuite** - It is now possible to trigger a scenario when a custom date field is updated. Use the **Watch Records** module for this purpose. Also, we added new hints in the **Create or Update a Record** module.
- **Workday Financial Management** and **Workday Human Capital Management** - You can select the format in which to get a RaaS report with the help of the **Get RaaS Report** module. Also, we added helping hints to let users know where to get credentials to create an OAuth2 connection.

8.2.22. February 12, 2024

Improvements and changes

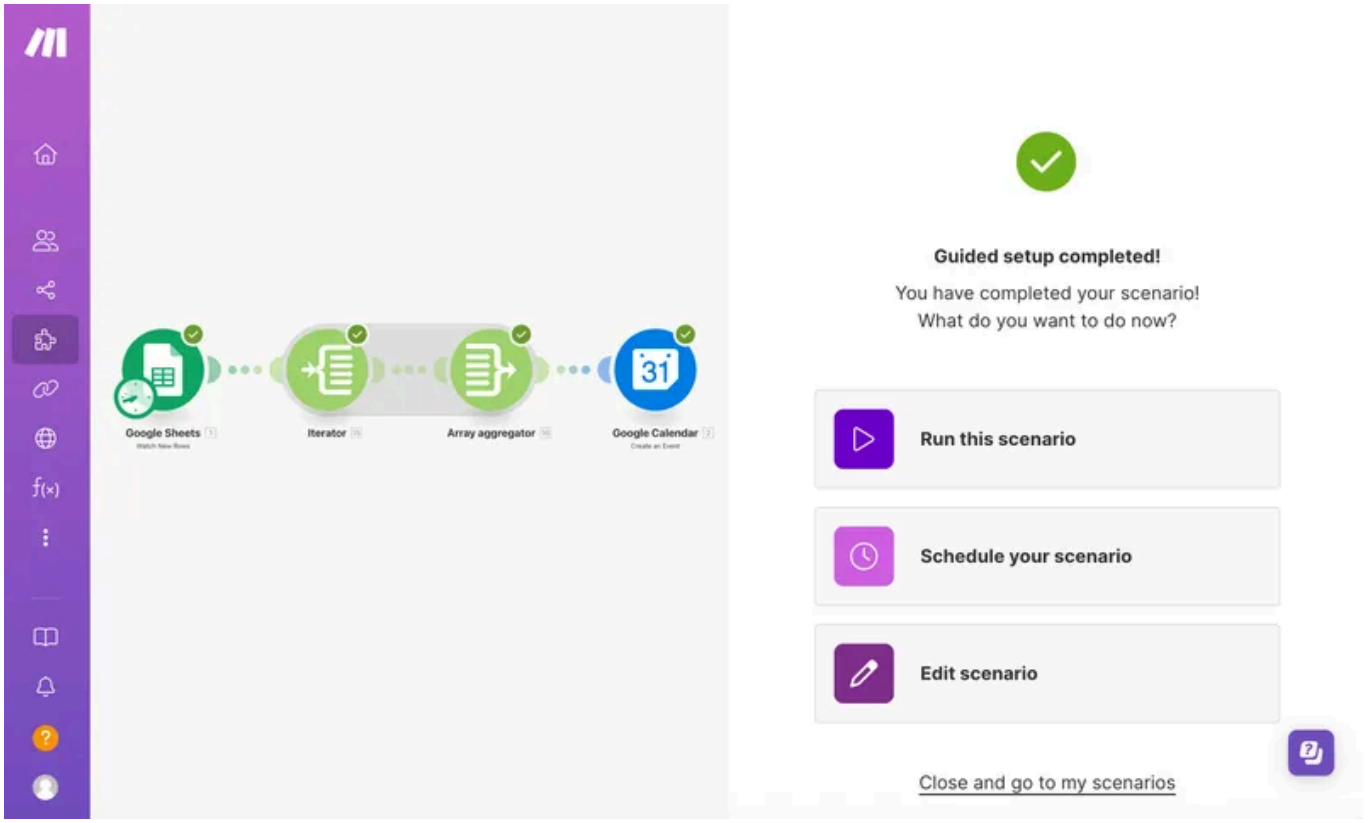
- Using templates to create.scenarios is now much easier and more flexible.
  - We added an "X" icon that allows you to easily close the module settings. This means you can now open and set up modules in any order you prefer.



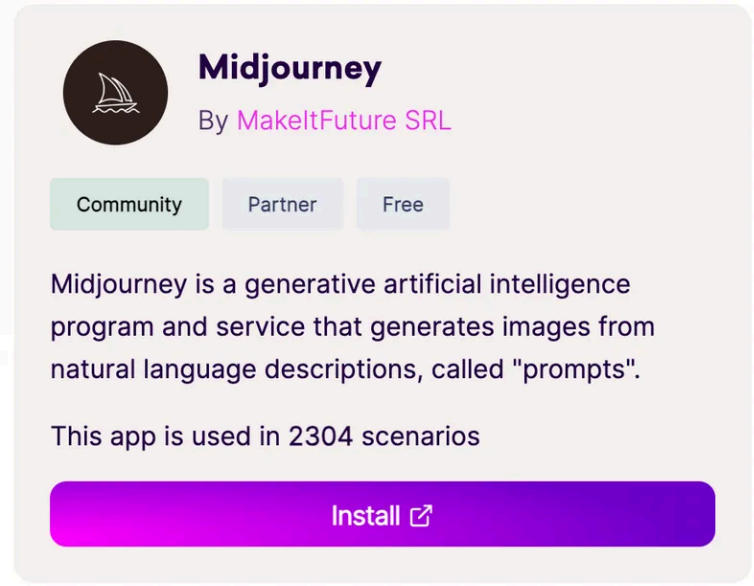
- We introduced badges for each module to make sure you don't miss any important information:

	A green badge indicates that all required fields are filled in.
	An orange badge signals that you need to provide some additional information.
	A gray badge reminds you that you still need to get started with setting up the module.

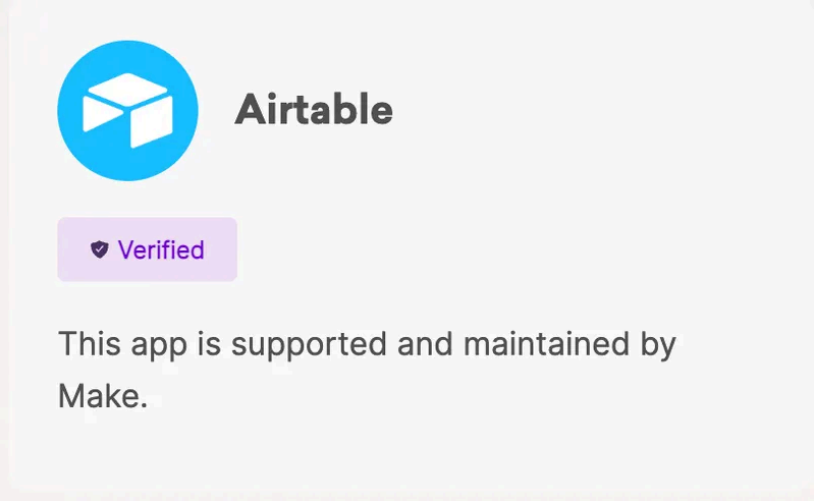
Once you complete the setup and all your badges turn green, you will see a side panel that gives you the option to run, edit, or schedule your scenario.



- We renamed the **add-on apps** to **community apps**. In addition, on the Make **integrations** page, we show the number of scenarios that are using the community app.



- On the Make **integrations** page, we show the verification badge for Make verified apps with their author and maintainer. This brings more clarity to who developed the app and who you should reach out to if you experience issues.



Fixed issues

- If a scenario contained an iterator followed by an aggregator, sometimes it would get stuck when processing a large number of bundles. The scenario then failed to create scenario logs. Now scenarios run smoothly and create scenario logs.

8.2.23. January 29, 2024

Improvements and changes

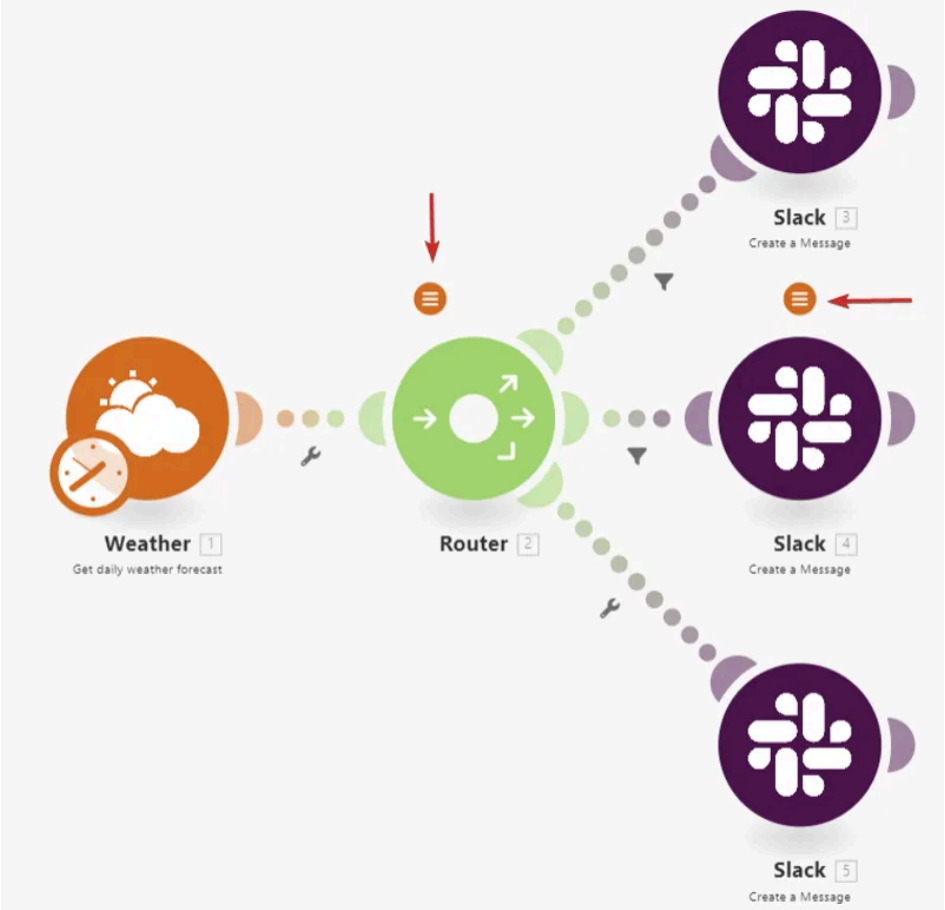
- Enterprise customers can now claim and verify an email domain for their Single Sign-On (SSO) setup, allowing for SSO enforcement. Make recognizes users with the claimed email domain and prompts them to use SSO. See our **article on domain claim** for information on setup and provisioning options.
- Using a coupon for a higher plan now changes your subscription. Once the coupon expires, you automatically stay on your new plan. Or, you can manually downgrade to your original plan. See our **article on coupons** for details.

Fixed issues

- Under the SSO option **Default teams for newly created users**, the list of teams showed deleted teams. This list now shows only valid teams.
- It wasn't possible to delete records in a webhook queue: the **Delete** button wasn't clickable.
- The infinite scroll in the scenario history wasn't infinite: it got stuck and didn't show all records. Now you have access to the full history.
- The hints for creating names of custom functions and scenario inputs were incorrect. The hints now show the correct requirements for custom function and scenario inputs names.
- Doubled and unaligned texts appeared when you clicked the **Create a connection** button. We solved it, so you have beautifully aligned and correct texts.
- When you tried to resolve incomplete executions, the **Run once** button wasn't clickable. Also, a module that caused an error lost its mapping. We fixed both issues.
- Mapping pills showed their raw name. Pills now have descriptive names.



- When you ran the **Explain flow** option and then deleted one or more modules, the dot showing the flow was stuck. We unfroze the dot, and now it shows the full flow (and takes into account the deleted modules).



- The built-in option that Make uses to decode URLs in the output refused to decipher links. Now URLs in the output are legible and don't contain odd symbols.

8.2.24. January 16, 2024

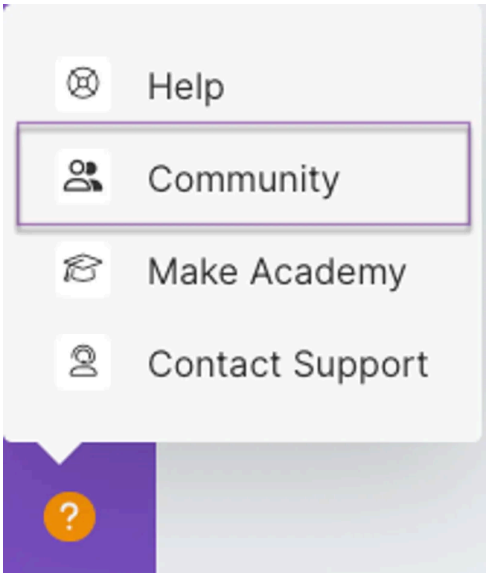
Dynamic connections

With dynamic connections, you and your teammates use connections as **scenario inputs**. You can create a scenario with your connections, and your colleagues provide their connections to run the scenario. To learn more about how to use dynamic connections, check out the **article in our Help Center**. Dynamic connections are available for **Enterprise** plan users only.

8.2.25. January 11, 2024

Improvements and changes

- A link to **Make Community** now appears in the sidebar **Help** submenu.



- Enterprise customers can now see their current subscription period and reset date on their organization dashboard.