

Edge Insights for Industrial Get Started

Contents

Chapter 1: Edge Insights for Industrial

How it Works	3
Training and Learning Suite (TLS).....	6
Updating an Application-Over-The-Air	8
Get Started Guide	59
Requirements	60
Install Edge Insights for Industrial	60
Introduction to the Edge Software CLI	70
Tutorials.....	77
Defect Detection Demo	77
Textile Defect Classifier	84
Industrial Text Line Recognition.....	94
Industrial Surface Defect Detection.....	104
Weld Porosity Detection	113
Release Notes	121
Documentation Archive	122

Edge Insights for Industrial

1

Edge Insights for Industrial from Intel is a set of pre-validated ingredients for integrating video and time-series data analytics on edge compute nodes.

Edge Insights for Industrial helps to address various industrial and manufacturing usages, which include data collection, storage, and analytics on a variety of hardware nodes across the factory floor. See [How it Works](#).

Use the [Get Started Guide](#) for installation instructions and an introduction to the Edge Software command line interface to learn how to manage Intel® Developer Catalog packages.

When set up is complete, choose the [Tutorials](#) section for step-by-step, hands-on walkthroughs of how to use and configure modules in Edge Insights for Industrial.

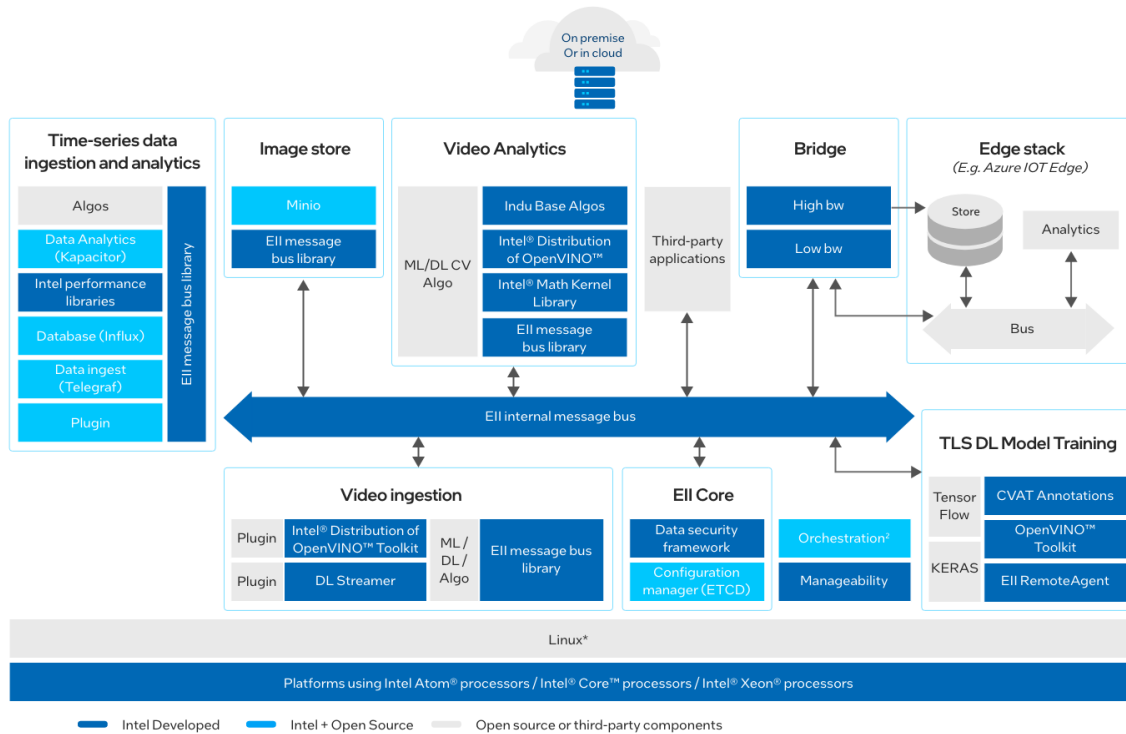
How it Works

Edge Insights for Industrial is a set of pre-integrated ingredients designed to accelerate the development and deployment of solutions for the industrial sector. The package is meant to be deployed on the device closest to the data generation, such as the tool or machine assembling a product. This enables the ingestion of video and time-series data, storage of data, performing analytics, closing the loop by transmitting a control message, and publishing the results.

Edge Insights for Industrial is based on [Open Edge Insights for Industrial](#) and offers additional components.

NOTE The Open Edge Insights for Industrial user guide details the capabilities of Open Edge Insights for Industrial modules and serves as a reference for developers. The Open Edge Insights for Industrial user guide is accessible at: <https://open-edge-insights.github.io/>.

This section provides an overview of the modules and services featured with Edge Insights for Industrial.



Modules and Services

For an overview of the EII modules, see below.

Video Ingestion and Analytics

Video Analytics

Classifier algorithms are executed in the Video Analytics container. The container uses the Edge Insights for Industrial Data Bus to subscribe to the input stream published by the Video Ingestion container. The algorithms included in the container are optimized for Intel hardware using the Intel® Distribution of OpenVINO™ toolkit (OpenVINO™ and the Intel® Math Kernel Library). You can create multiple instances of the Video Analytics container to classify multiple video input sources.

- The **Intel® Distribution of OpenVINO™ toolkit** is a software development kit (SDK) for deploying deep learning computer vision inference applications. It optimizes inferencing on your edge IoT device for Intel® architecture.
- The **Intel® Math Kernel Library** is a library of highly-optimized, threaded, and vectorized math routines designed to maximize performance on Intel® processors.

Watch a [video](#) on Video Analytics on the Edge Insights for Industrial [Architecture & Technical Training page](#).

Video Ingestion

Edge Insights for Industrial supports ingestion from:

- Video files.
- GigE cameras.
- USB cameras.
- RTSP cameras.

Edge Insights for Industrial uses **GStreamer*** and OpenCV to provide a preconfigured ingestion pipeline that you can modify for different camera types and preprocessing algorithms. The Video Ingestion container publishes video data, consisting of metadata and frames, to the Edge Insights for Industrial Data Bus. You

can create multiple instances of the Video Ingestion container to ingest video from multiple input sources. In addition, the Video Ingestion container itself can run UDFs performing analytics. This is recommended for faster processing of frames to avoid transmission delays.

Watch a [video](#) on Video Ingestion on the Edge Insights for Industrial [Architecture & Technical Training page](#) .

Training and Learning Suite (TLS)

The Training and Learning Suite has a web-based user interface for training deep learning models. The key TLS features are:

- Remote deployment of a TLS-trained model into an Edge Insights for Industrial system.
- Visualization of Edge Insights for Industrial video data classification results.

For details, see the [Training and Learning Suite \(TLS\)](#) section.

Image Store

The image store provides storage and retrieval of images as binary blobs. You can store the images persistently by using: In persistent storage with [MinIO*](#), an open source, distributed, object storage server.

Watch a [video](#) on Image Store on the Edge Insights for Industrial [Architecture & Technical Training page](#) .

Time Series Analytics

Edge Insights for Industrial leverages [Telegraf*](#) for time-series data ingestion into [InfluxDB*](#). [Kapacitor*](#) reads data from [InfluxDB*](#), generates alerts, and takes the appropriate action.

- [Telegraf*](#) is an agent for collecting, processing, aggregating and writing performance metrics.
- [InfluxDB*](#) is a distributed datastore for metrics, events, and real-time analytics.
- [Kapacitor*](#) is a data processing engine that handles both stream and batch data from [InfluxDB](#), acting on this data in real time via its programming language, TICKscript.

Watch a [video](#) on Time Series Data on the Edge Insights for Industrial [Architecture & Technical Training page](#).

Security and Configuration

The Data Security Framework enables security through a two-stage process involving provisioning and execution on Edge Compute Node. Its primary objective is to prevent unauthorized access of generated data from within the system or via external network interfaces.

Edge Insights for Industrial uses [etcd*](#) for configuration management. [etcd*](#) is a consistent, distributed key-value store that provides a reliable way to store data that needs to be accessed by a distributed system or cluster of machines.

Manageability

Edge Insights for Industrial uses [Telit*](#), [Azure*](#) and [ThingsBoard*](#) to support over-the-air updates and to enable cloud-to-edge manageability of application services.

For details, refer to the [Manageability](#) section.

Docker

[Docker*](#) is a container framework widely used in enterprise environments. It allows applications and their dependencies to be packaged together and run as a self-contained unit.

Edge Insights for Industrial Data Bus

Edge Insights for Industrial Data Bus is an abstraction over [ZeroMQ*](#), which is used for all inter-container communication.

[ZeroMQ*](#) is a brokerless message bus that transfers data from the source directly to the destination. [ZeroMQ*](#) is used in TCP and IPC mode with pub-sub and request-response patterns.

Training and Learning Suite (TLS)

TLS Server

The Intel® Training and Learning Suite 2.0 (TLS) enable data scientists to easily annotate images and create AI models in just a few clicks. For more details on using this tool, refer TLS user guide which is available in `Edge_Insights_for_Industrial_<version>/training-and-learning-suite/doc/631070_TLS2.0_UserGuide_Rev1.2.pdf`.

TLS Remote Agent

This module is a communication agent in EII to interact with Training and Learning Suite (TLS) server, which is another module application as part of EII to provide deep learning model training capability through web user interface. TLS can be setup as deep learning training server, either in a separate system or same local system with EII. Both TLS and EII communicate through RabbitMQ broker using mqtt protocol. To know details of TLS, one can refer to TLS repository, which included the documentation for setup and user guide.

NOTE The TLS remote agent is used along with video data classification use case only.

NOTE By default, the EII downloaded from Github opensource will not have TLS included in the package. The TLS full package can be downloaded via customize package in ESH.

TLS Remote Agent Configuration

Before starting the `ia_tls_remoteagent`, let's take a closer look at the TLS Remote Agent configuration in the EII Configuration Manager.

By default, `/TLS_RemoteAgent/config` in EII Configuration Manager has been configured to run with the `pcb_demo` sample application and to connect to TLS in localhost.

```
{
  "tls_host": "localhost",
  "username": "user",
  "passwd": "password",
  "user_labels":
  [{
    "VideoAnalytics": {
      "0": "D_MISSING",
      "1": "D_SHORT"
    }
  ]
}
```

Firstly, update the IP address of the TLS system into **tls_host** parameter. The default is **localhost** can be used only if the TLS system is being setup in the same local system with EII. Otherwise, the localhost should be replaced by the IP address of the remote system, where TLS has been setup.

Then, update the **username** and **passwd** that will be used for RabbitMQ broker's authentication. This is the second set of username and password being setup during the `./generateCert.sh` execution in TLS setup so RabbitMQ broker can establish a client connection.

User can define the labels for each of the classified results based on the trained model implemented in the classifier. For example, in the default `pcb_demo` sample application, the SubTopics in `docker-compose.yml` file is `VideoAnalytics/camera1_stream_results`, where `VideoAnalytics` is the AppName and `camera1_stream_results` is the topic name. In the `user_labels` parameter, use the AppName and define its labels in json format, as shown above. This will allow the labels to be displayed on TLS web UI based on the classified result.

For another example with multiple `VideoAnalytics` containers and combination with GVA-plugin used in `VideoIngestions`, refer to the user-defined labels example below through EII Configuration Manager.

```
{
  "tls_host": "localhost",
  "username": "user",
  "passwd": "password",
  "user_labels":
  [{
    "VideoAnalytics": {
      "0": "D_MISSING",
      "1": "D_SHORT"
    },
    "VideoIngestion2": {
      "1": "safety_helmet",
      "2": "safety_jacket",
      "3": "safe",
      "4": "violation"
    }
  ]
}
```

The above example of user labels is based on the assumption of use cases, as below:

1. **VideoAnalytics** as first-stream topic with **pcb_demo** and defect type in the classifier algo parsing object id for label mapping.
2. **VideoIngestion2** as second-stream topic, which used the GVA-plugin that the inferencing happened in `VideoIngestion` container using `GStreamer` and no `VideoAnalytics` is required. Therefore, the AppName being used here is "VideoIngestion" and not "VideoAnalytics". In this use case, `safety_gear_demo` and user-defined labels per the EII Configuration Manager are used because the GVA-plugin does not use the classifier algo in `VideoAnalytics` to parse the defect type.

TLS Remote Agent Setup

Pre-requisite:

1. TLS has been setup in the same local system with EII, or in a remote system with network connection established to EII system.
2. TLS has at least one model training completed so it can be configured to setup the connection with EII for trained model deployment.

Setup and Run TLS Remote Agent:

1. Follow the **Configuration** section above, configure TLS remote agent accordingly and save the new configuration through EII Configuration Manager web UI.
2. TLS has been designed to run only secured mode, where security certificates are required that covered in TLS setup guide. For EII running in **dev_mode**, steps below can be skipped. Otherwise, these steps are required if EII is running in **prod_mode**.

NOTE These steps need to be executed again whenever TLS system regenerated the new certificates using the **./generateCert.sh** tool.

1. From TLS system, go to **/thirdparty/security** directory, copy the 3 files below to **/build/provision/Certificates/TLSRemoteAgent** directory in EII system.
 - a. Under **security/ca** directory, copy **ca_certificate.pem** file.
 - b. Under **security** directory, copy **TLS_rabbitmq_cert.crt** and **TLS_rabbitmq_key.pem** files.
2. Go to the destination directory **/build/provision/Certificates/TLSRemoteAgent** in EII system, use below command to change the 3 files ownership.

```
$ sudo chown eiiuser:eiiuser ca_certificate.pem
$ sudo chown eiiuser:eiiuser TLS_rabbitmq_cert.crt
$ sudo chown eiiuser:eiiuser TLS_rabbitmq_key.pem
```

3. Optional: If the network for TLS and EII system is running behind proxy, add the TLS system IP to the **ia_tls_remoteagent** docker compose setting, under **environment -> no_proxy** in the consolidated **docker-compose.yml** file as below.

```
environment:
  no_proxy: ${eii_no_proxy},${ETCD_HOST},
```

4. Now, restart the TLS remote agent container by using command below:

```
$ docker-compose up -d --force-recreate ia_tls_remoteagent
```

Upon **ia_tls_remoteagent** started, it will read the SubTopics from **ia_tls_remoteagent** in the **docker-compose.yml** file to check how many camera/video stream topic being subscribed. Then it will based on each camera stream to create a unique UUID and display it on the screen. Alternatively, use the command below to view the **ia_tls_remoteagent** container log at terminal to obtain the UUID.

```
$ docker logs -f ia_tls_remoteagent
```

This unique UUID for each stream topic will be needed to add into TLS web UI in order to connect TLS system and EII system. For details, please refer to TLS user guide remote agent section. The naming convention format as following:

systemUUID_streamTopic

With **ia_tls_remoteagent** up and running, it will allow user to connect with TLS system to:

1. Deploy TLS trained model from TLS system to EII system. When user deploy TLS trained model from TLS web UI, the model files will be transferred to **ia_tls_remoteagent** container under **common/udfs/model_repo** directory. This directory is mounted on EII host system under **/opt/intel/eii/model_repo** directory as configured in **docker-compose.yml** file.
2. View EII classified result through TLS web UI. Each of the classified results stream topic from EII can be streamed back to TLS UI through the UUID setup and it can be controlled by **View** button on TLS web UI.

By enabling **ia_tls_remoteagent** module, it establish connection between TLS and EII for end to end deep learning deployment as user can train deep learning model base on own image dataset and deploy the trained model on EII edge inferencing which highly optimized on Intel platform.

Updating an Application-Over-The-Air

Application-Over-The-Air (AOTA) update enables cloud to edge manageability of application services running on the EII-enabled systems through the Device Manageability component. Device Manageability facilitates software updates and deployment from cloud to device which includes SOTA, FOTA, AOTA, and few system operations.

For EII use case, only **AOTA** features from Device Manageability are validated and will be supported through **Azure***, **ThingsBoard*** and **Telit*** cloud-based management front-end services. Depending on your preference you can select either one of these options.

The following section will walk you through - setting up **Azure***, **ThingsBoard***, and **Telit***. - creating and establishing connectivity with the target systems, as well as updating applications on those systems.

Refer to the user guide in **docs** for more detailed reference.

NOTE Edge Insights for Industrial (EII) was previously named Intel® Edge Insights Software (EIS). Remnants of the previous name still exist in some components. The name replacement is ongoing and will be completed in future releases.
Device Manageability was previously named Turtle Creek. Remnants of the previous name still exist in some components. The name replacement is ongoing and will be completed in future releases.

Installation - Device Manageability

Prerequisites

1. Ubuntu 18.04
2. Install the latest docker cli/docker daemon. Refer to the **Install using the repository** section and the **Install Docker Engine** section at <https://docs.docker.com/install/linux/docker-ce/ubuntu/#install-docker-ce>.

Also, to run docker without sudo refer to the Manage Docker as a non-root user section at <https://docs.docker.com/install/linux/linux-postinstall/>

3. Complete the following steps only if the node or system on which the docker setup is tried out is running behind a HTTP proxy server. Skip this step if that's not the case.
 - a. Configure proxy settings for docker client to connect to internet and for containers to access internet. For more information, see: <https://docs.docker.com/network/proxy/>.
 - b. Configure proxy settings for docker daemon. For more information, see <https://docs.docker.com/config/daemon/systemd/#httphttps-proxy>.
4. Install the docker-compose tool. For more information, see <https://docs.docker.com/compose/install/#install-compose>

All Device Manageability devices can be controlled using a cloud service application. To install Device Manageability complete the following:

NOTE This step is required only if you select cloud service **Azure***, **ThingsBoard***, and **Telit***.

1. Install the Device Manageability agent on the machine.
 - a. Run the following commands to install from the manageability folder:

```
sudo chmod +755 install-tc.sh
sudo ./install-tc.sh
```

NOTE Before installing the newer version check if the system has an old version of Device Manageability. If an older version exists then uninstall it first using the same version of uninstall script.

- b. After reading all the licenses, press 'q' to finish.
- c. Accept the License by entering 'Y'.

Figure below shows the Terminal output you should expect to see after the installation is complete.

```

user@user-2: ~/turtlecreek-v2.1-RC3
File Edit View Search Terminal Help
Ran systemctl daemon-reload
Setting up configuration-agent (0.37.2-1) ...
After install called
Found systemd
Activating configuration's apparmor policies
Ran systemctl daemon-reload
Setting up diagnostic-agent (0.37.2-1) ...
After install called
Found systemd
Activating diagnostic's apparmor policies
Ran systemctl daemon-reload
Setting up dispatcher-agent (0.37.2-1) ...
After install called
Found systemd
Activating dispatcher's apparmor policies
Ran systemctl daemon-reload
Setting up telemetry-agent (0.37.2-1) ...
After install called
Found systemd
Activating telemetry's apparmor policies
Ran systemctl daemon-reload
Agent Installation Complete
Turtle Creek Installation Complete
user@user-2:~/turtlecreek-v2.1-RC3$

```

- d. After the Device Manageability has been installed successfully, set the following values in the `/etc/intel_manageability.conf` file.

```
sudo vi /etc/intel_manageability.conf
```

- e. Under `<all>` `</all>` section change `db`s ON to OFF (refer to the following figures). DBS stands for Docker Bench Security, this feature of Device Manageability is not used for EII.

```

<?xml version="1.0" encoding="utf-8"?>
<!-- configuration-agent/fpm-template/etc/intel_manageability
<configurations>
  <all>
    <db>sON</db>s
  </all>
  <telemetry>
    <collectionIntervalSeconds>60</collectionIntervalSeconds>
    <publishIntervalSeconds>300</publishIntervalSeconds>
    <maxCacheSize>100</maxCacheSize>
    <containerHealthIntervalSeconds>600</containerHealthIntervalSeconds>
  </telemetry>

```



```
<?xml version="1.0" encoding="utf-8"?>
<!-- configuration-agent/fpm-template/etc/intel_manageability
<configurations>
  <all>
    <db>OFF</db>
  </all>
  <telemetry>
    <collectionIntervalSeconds>60</collectionIntervalSeconds>
    <publishIntervalSeconds>300</publishIntervalSeconds>
    <maxCacheSize>100</maxCacheSize>
    <containerHealthIntervalSeconds>600</containerHealthIntervalSeconds>
  </telemetry>
```

- f. Add the URL endpoint for the developer files to the TrustedRepositories as shown in the following figure.

```
<trustedRepositories>
https://api-dev.devicewise.com
</trustedRepositories>
```

- g. Save and exit. You must restart the machine before these changes can take effect.

Multi-Node Deployment

EII deployment on multiple nodes requires the use of the Docker* registry. The following section outlines some of the commands to be run on the primary node and on any newly added secondary nodes.

NOTE This step is required only if you select cloud service **Azure Portal***, **ThingsBoard***, or **Telit***.

1. Set up the Docker registry URL, build, and push images. This is required to run on the **primary** node.
 - a. Update the docker registry URL in the **DOCKER_REGISTRY** variable **<ip:port>**. The example uses **10.221.40.65:5000/** as the Docker registry.

```
sudo vi [WORK_DIR]/IEdgeInsights/build/.env
```

```
user@user-1: ~/EIS2.1/IEdgeInsights/docker_setup
File Edit View Search Terminal Help
# video analytics
SOCKET_DIR=/EIS/sockets

# DEV_MODE if set `true` allows one to run EIS in non-secure mode and provide
additional UX/DX etc.,
DEV_MODE=true
# PROFILING_MODE is set 'true' allows to generate profile/performance data
PROFILING_MODE=false

RTSP_CAMERA_IP=localhost
# All server certificates will be generated with below HOST_IP
# If HOST_IP is blank, HOST_IP will be automatically detected while generating
certificate
HOST_IP=
# Please provide docker registry details below for docker-compose push/pull
# Please provide full registry url with port trail by /
# e.g. localhost:5000/
DOCKER_REGISTRY=10.221.40.65:5000/

# EIS dependency image versions
UBUNTU_IMAGE_VERSION=18.04
GRAFANA_VERSION=6.4.3
INFLUXDB_VERSION=1.5.3
EIS_VERSION=2.1
```

- b.** In the same build repository, build the image using the following command:

```
docker-compose build
```

- a.** On a successful build, you will see the following terminal output:


```

user@user-1: ~/EIS2.1/IEdgeInsights/docker_setup
File Edit View Search Terminal Help
---> Using cache
---> 98592ed97bac
Step 27/31 : COPY --from=common ${GO_WORK_DIR}/common/cmake ${PY_WORK_DIR}/c
/cmake
---> Using cache
---> b520ec205a87
Step 28/31 : COPY --from=common /usr/local/lib /usr/local/lib
---> Using cache
---> c0b687d8bbb6
Step 29/31 : COPY --from=common /usr/local/lib/python3.6/dist-packages/ /usr
l/lib/python3.6/dist-packages
---> Using cache
---> c27491060d45
Step 30/31 : COPY . .
---> Using cache
---> 85d580eb49cd
Step 31/31 : ENTRYPOINT ["python3.6", "visualize.py", "-l"]
---> Using cache
---> 1f34e088421a

[Warning] One or more build-args [EIS_UID] were not consumed
Successfully built 1f34e088421a
Successfully tagged 10.221.40.65:5000/ia_visualizer:2.1
user@user-1: ~/EIS2.1/IEdgeInsights/docker_setup$

```

- a. Push the image to the registry:

```
docker-compose push
```

2. To run EII in multi-node, you must identify one primary node. To set the primary node, set ETCD_NAME in the [WORK_DIR]/IEdgeInsights/build/.env file to master.

```

# Etcd settings
ETCD_NAME=master
ETCD_VERSION=v3.4.0
ETCD_DATA_DIR=/EIS/etcd/data/
ETCD_RESET=true
ETCD_CLIENT_PORT=2379
ETCD_PEER_PORT=2380

```

3. You can proceed to provision the primary node using the following command. This will create the ETCD server on the primary edge node.

```

cd [WORK_DIR]/IEdgeInsights/build/provision
sudo ./provision.sh <path_to_eii_docker_compose_file>

```

4. After provisioning on the primary node, you need to provision for each of the worker node that will be included in the multi node deployment. Below step need to be executed at the primary node. Users need to ensure the ETCD_NAME in [WORK_DIR]/IEdgeInsights/build/.env have been set to other than "master" name.

```
cd [WORK_DIR]/IEdgeInsights/build/deploy
sudo python3.6 generate_eii_bundle.py -p
```

This step will generate the **eii_provisioning.tar.gz** at the same directory. Copy this to the worker node for provisioning.

5. After copying the generated file to the worker node, run the following commands on the worker node:

```
tar -xvzf eii_provisioning.tar.gz
cd eii_provisioning/provision/
sudo ./provision.sh
```

Generating EII Bundle for deployment

On the EII primary node, on which EII is running, complete the following steps to generate the EII_bundle.

This software will only work on Python* 3 and onwards.

1. Update the config.json file to include the services that you need for the bundle. Ensure to exclude the services that you do not want with the bundle.

```
$ sudo vi build/deploy/config.json

{
  "docker_compose_file_version": "<docker_compose_file_version which is compose file version
supported by deploying docker engine>",
  "exclude_services": [list of services which you want to exclude from your deployment]
  "include_services": [list of services needed in final deployment docker-compose.yml]
}
```

Note: Ensure that you have updated the DOCKER_REGISTRY in the build/.env file as mentioned in the Multi-Node deployment section.

2. Update the .env file at the [repo/build] directory for the worker node environment setting by changing below parameters

```
sudo gedit .env
ETCD_NAME=<any name other than `master`>
ETCD_HOST=<IP address of primary node>
DOCKER_REGISTRY=<Docker registry details>
```

3. Run the following script in the terminal to generate the eii_bundle

```
cd [WORK_DIR]/IEdgeInsights/build/deploy/
sudo python3.6 generate_eii_bundle.py
```

The default value for bundle name is **eii_bundle.tar.gz**, while the tag name is **eii_bundle**.

Using -t options can give you a custom tag name which you can use for bundle generation.

The default value for the docker-compose yml path is ../docker-compose.yml.

4. The file **eii_bundle.tar.gz** will be created in the same folder. This bundle will be used to deploy EII on the other node via ThingsBoard.

Creating an HTTP Local Server

NOTE This step is required only if you select cloud service Azure* or ThingsBoard* cloud services and for development purpose. During actual implementation, you need to have your own cloud http server to upload the EII bundle that being generated and use the URL to fetch the uploaded bundle on the AOTA step later.

The generated EII bundle need to available via http server to perform AOTA fetching from the cloud services.

Prerequisites

Run `sudo vi /etc/environment`.

Under `no_proxy` variable, add the local http server (example, 10.221.40.65).

Perform `sudo systemctl restart configuration`

1. Run the following command on the ThingsBoard machine to create the HTTP server for sharing the `eii_bundle`:

```
python3 -m http.server [any port number]
```

2. Add the fileservers endpoint to the configuration file on the node that you will access via the local HTTP server.

In the example, the steps below must be done on the new node. The new node will access the local HTTP server to fetch the `eii_bundle` that is being created.

- a. Open the configuration file

```
sudo vi /etc/intel_manageability.conf
```

- b. Under "**trustedRepositories**", add the bundle hosting server endpoint on a new line. <http://xx.xxx.xxx.xx> as used above.

```
<trustedRepositories>
  https://af01p-igk.devtools.intel.com/artifactory/SID-Docker-local/bmp
  https://ubit-artifactory-or.intel.com/artifactory/iotg-bmp-internal-1
  https://af01p-igk.devtools.intel.com/artifactory/iotg-bmp-test-local/
  https://af01p-igk.devtools.intel.com/artifactory/iotg-bmp-igk-local/
  http://ci_nginx:80
  https://api-dev.devicewise:433/file/
  http://10.221.40.65
</trustedRepositories>
```

NOTE The above example is running local HTTP server at 10.221.40.65:8000

- c. Perform `$ sudo systemctl restart configuration` on the machine.

Cloud Service: Azure Server Set up

All Device Manageability devices can be connected and controlled via the Azure portal. An Azure account can be created through the following link:

<https://azure.microsoft.com/en-us/free/>, select start free, and create the account.

Setting up an Azure IoT Central Application

1. To generate a premade Device Manageability (Intel Manageability) IoT Central application get the link from `/usr/share/cloudadapter-agent/azure_template_link.txt` file on Device Manageability machine.

2. Copy and paste the link in your browser, you will be redirect to the Azure UI as shown below.



Azure IoT Central



Build > New application



Intel Manageability v1.3



Use this template to interact with devices running Intel Manageability.



About your app

Application name * ⓘ

Custom 1yxec3lu6km

URL * ⓘ

custom-1yxec3lu6km

Pricing plan *

☒ Free

Try for **7 days** with no commitment

5 free devices

☐ Standard 1

For devices sending a **few messages per hour**

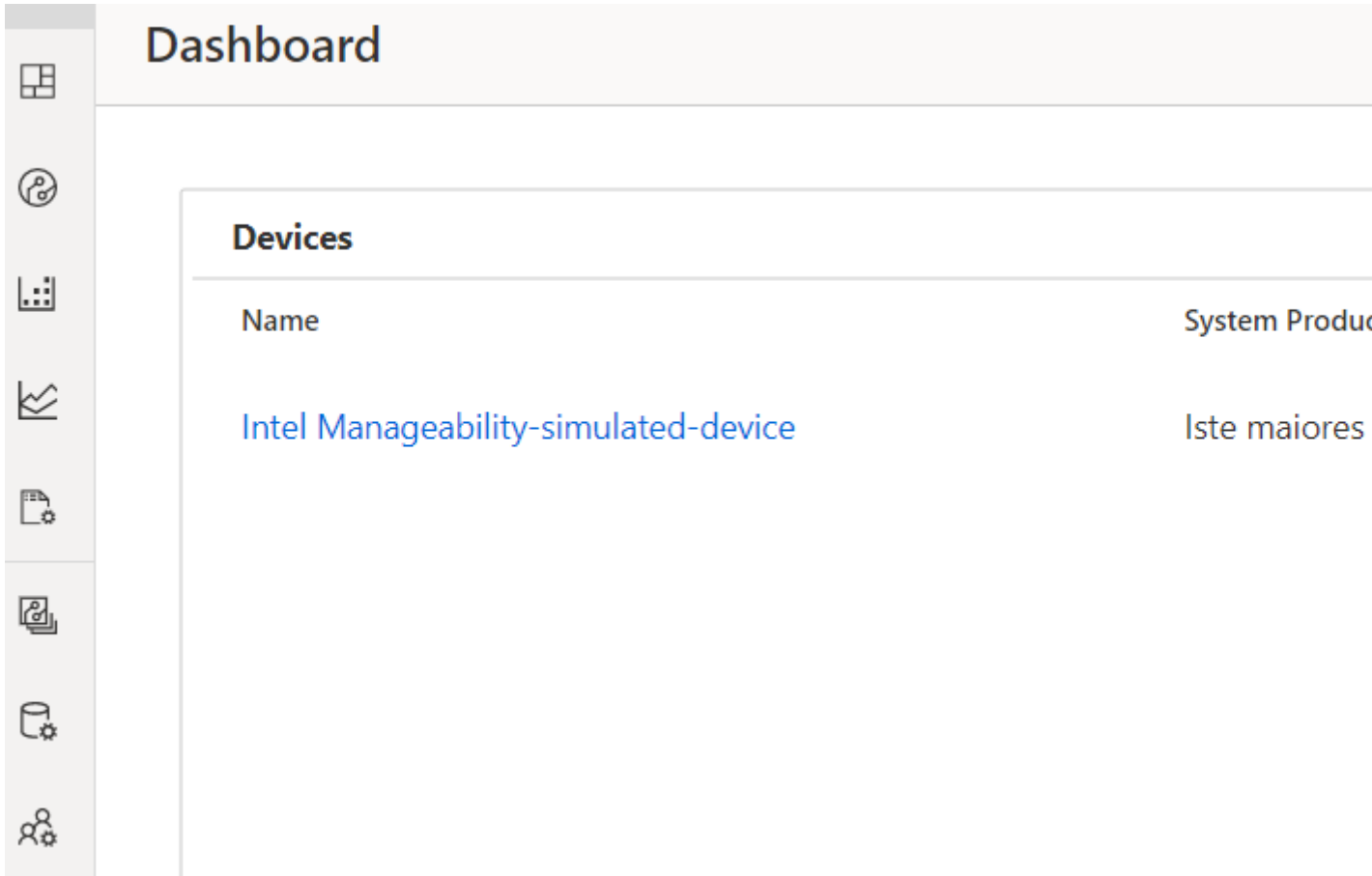
2 free devices **5,000** messages/mo

☐ Standard 2 (most popular)

For devices sending **messages every few minutes**

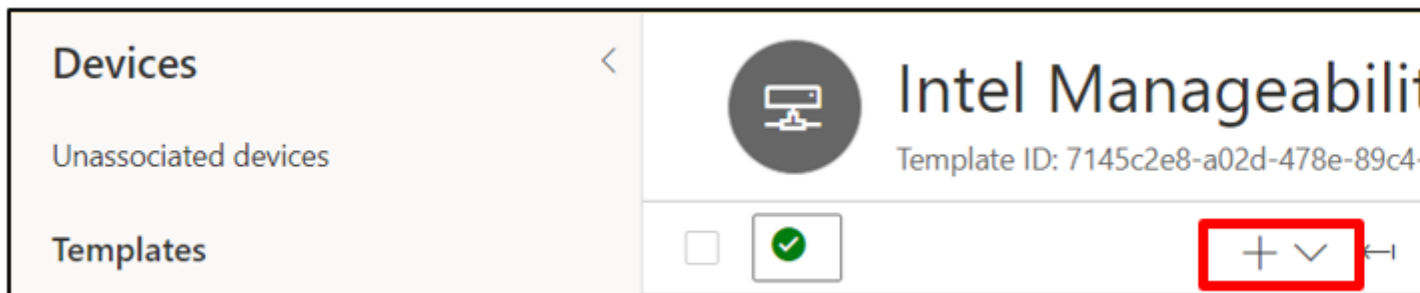
2 free devices **30,000** messages/mo

3. For the pricing plan, select free trial version. Fill up the necessary contact information and click on the create account button.
4. After provisioning, the IoT Central application with premade device templates and dashboards will appear. As noted before, this can be accessed at <https://apps.azureiotcentral.com> or through the Azure portal.

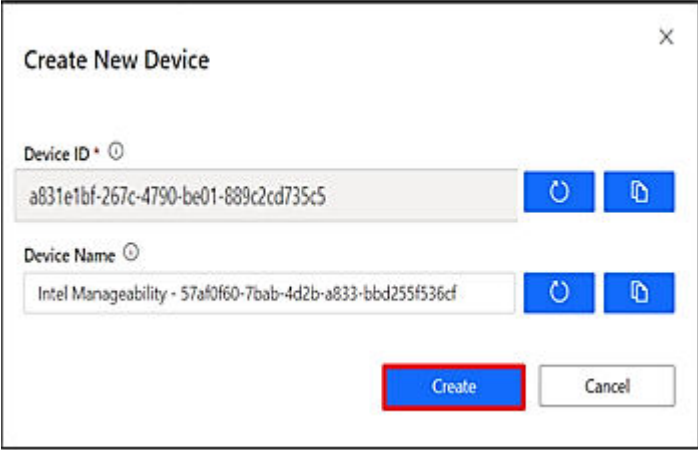


Obtaining device credentials

1. To add new device, click on the devices tab on the side bar, click on the add new device button.



2. Select real on the drop-down list, create new device form will appear as below. Click on create button.



The image shows a 'Create New Device' dialog box. It has a title bar with a close button (X). Inside, there are two input fields. The first is labeled 'Device ID' with a required field asterisk and a help icon. It contains the text 'a831e1bf-267c-4790-be01-889c2cd735c5'. To its right are two blue buttons: a power button and a copy button. The second input field is labeled 'Device Name' with a help icon. It contains the text 'Intel Manageability - 57af0f60-7bab-4d2b-a833-bbd255f536cf'. To its right are also two blue buttons: a power button and a copy button. At the bottom right, there are two buttons: a blue 'Create' button and a grey 'Cancel' button. The 'Create' button is highlighted with a red border.

3. The new added device will appear on the list of the devices page. To connect the device, click on that specific device and click on the connect button at the top bar right side.
4. Device connection information form will appear as below.

Device connection

ID scope ⓘ


0ne000CEFBC

Device ID ⓘ

a831e1bf-267c-4790-be01-889c2cd735c5

Credentials

Shared access signature (SAS) Certificates (X.509)

SAS security tokens are an attestation mechanism for devices to connect to IoT Central. The keys for this device are shown below. Use them to register your device with IoT Central. [more.](#) 

Primary key ⓘ

V49RzGEoDxUAsP/gRB+VS4llL9jjiQYnSwXJR6QErHs=

Secondary key ⓘ

Xnh/We2ghEh7A/3YUd+QAuArhSEaXV4+IZOeb7uDyTg=

Prerequisites : In order to perform this step, the Device Manageability device needs to be installed according to the instructions in the Installation Device Manageability section.

1. Provisioning can be done with or without the TPM security by setting 'PROVISION_TPM'. Using the following commands you can set 'PROVISION_TPM' to the Device Manageability device:
 - auto: use TPM if present; disable if not present; do not prompt
 - disable: do not use TPM.
 - enable: use TPM; return error code if TPM not detected.
 - (unset): default behavior; use TPM if present, prompt if not
2. On the Device Manageability device, run the following command:

```
sudo PROVISION_TPM=auto provision-tc
```

- To run without TPM security:

```
sudo PROVISION_TPM=disable provision-tc
```

3. If the device has been configured earlier, it will prompt as follows:

```
Enabling and starting mqtt (this may take some time to generate secrets)
A cloud configuration already exists: "azure"
Replace configuration?
[Y/N] y
```

- Press 'Y' to replace the existing configuration.
- Select '2' for cloud service option.

```
Please choose a cloud service to use:

1) Telit Device Cloud   3) ThingsBoard
2) Azure IoT Central    4) Custom
#? 2
```

4. A prompt for Device provision type appears; select the type of device authentication preferred: Choose 1 for SAS key authentication. If you choose option2 - Refer section [2.5 - Provisioning a Device] at ./docs/In-Band_Manageability_UserGuide_Azure.pdf for further steps.
5. All the device credentials from obtaining device credentials will be used here. Enter the following information accordingly.

```

Please enter the device Scope ID:
0ne000CEFBC

Please enter the Device ID:
a831e1bf-267c-4790-be01-889c2cd735c5

Please enter the device SAS Primary Key:
V49RzGEOdxUAsP/gRB+VS41lL9jjiQYnSwXJR6QErHs=

Successfully configured cloud service!

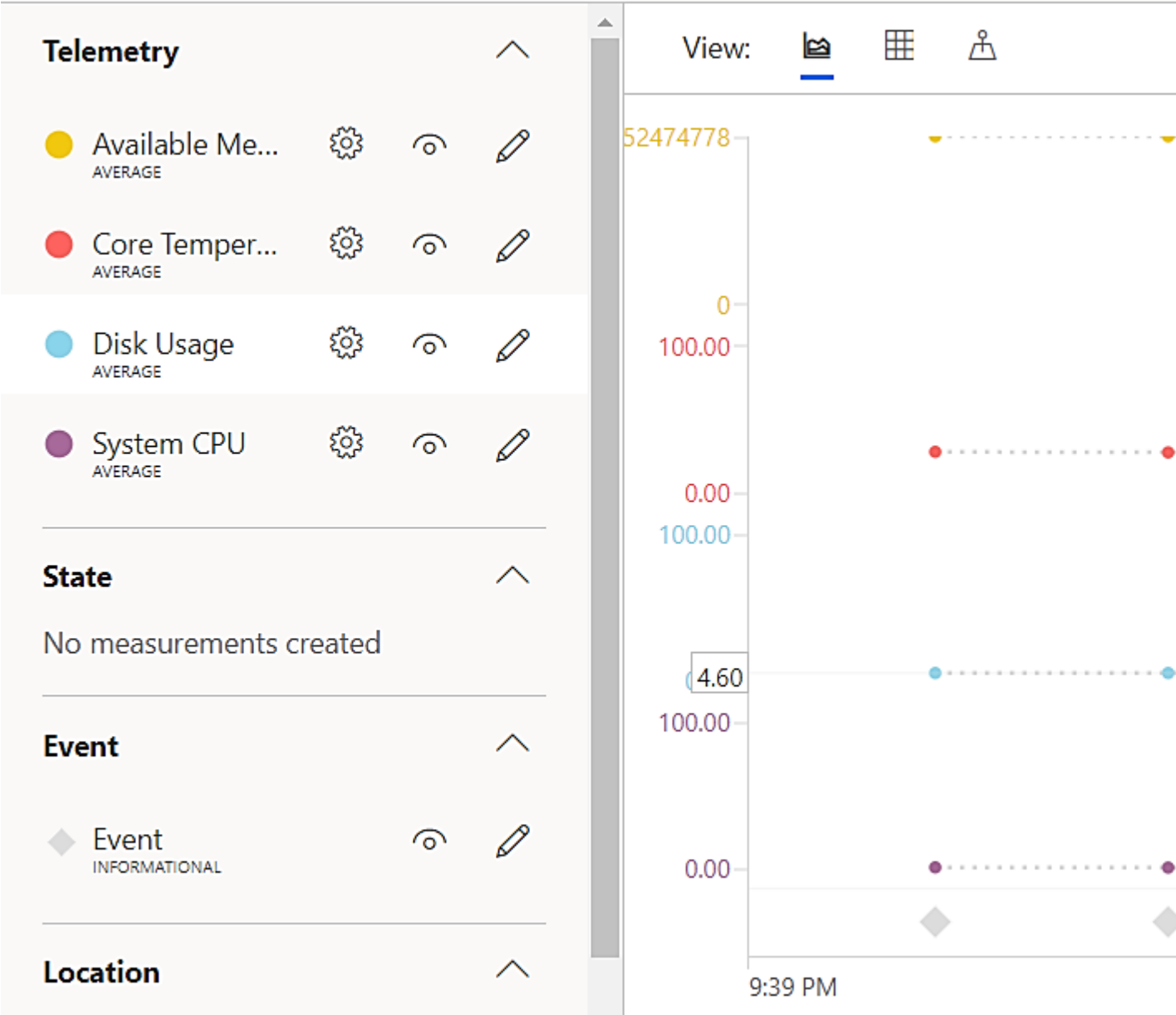
Enabling and starting agents...
Turtle Creek Provisioning Complete

```

6. The script will then start the Intel Manageability services; when the script finishes, the device should be able to be interacted with on its associated IoT Central Application. To verify connectivity: a. Check to see if telemetry/events are appearing. If the info is not seen, use `systemctl restart cloudadapter telemetry` command to restart the two agents give few seconds for the info to be populated and refresh the portal page. b. Alternatively, trigger a command like Reboot.
7. Run the provisioning script to change or update the cloud service configuration.
8. On the Azure portal, the device status should be changed from 'Registered' to 'Provisioned' as follows

<input type="checkbox"/> Name	Device ID	Device
<input type="checkbox"/> Intel Manageability - a831e1bf-267c-4790-be01-889c2cd735c5	a831e1bf-267c-4790-be01-889c2cd735c5	No

9. The following example shows the telemetry data of the connected devices that is available on the measurement tab in the Azure portal.



10. You can get the additional details information for the specific hardware and the event logs on the properties tab as follows:

The screenshot shows the TurtleCreek Azure IoT Central interface. On the left is a navigation sidebar with options: Dashboard, Devices, Device sets, Analytics, Jobs, App settings, Device Templates, Data export, and Administration. The main area displays the 'TEST' device page. At the top right of this page are tabs for Measurements, Settings, Properties, Commands, and Rules. Below the tabs is a descriptive sentence: 'The dashboard collects your device data and displays it in one centralized location.' The 'Summary' section contains a table of system information. Below this is an 'Events' section with a table of recent events.

Summary		
System Manufacturer	System Product Name	BIOS Release Date
		2019-05-10 00:00
BIOS Vendor	BIOS Version	CPU
INSYDE Corp.	V25.02.04	Intel(R) Core(TM)
Disk Information	OS Information	Total Memory
[{"NAME": "loop0...	Linux user-2 5.3.0-...	33525768192

Events			
Event	Time	Category	Value
Event	3/12/2020, 2:25:49 PM	Informational	network-information: {"cards
Event	3/12/2020, 2:25:48 PM	Informational	containers-cpu-percent: {"co
Event	3/12/2020, 2:25:43 PM	Informational	Connected

11. The following is the example of basic information that shown on the dashboard tab.

The screenshot displays the TurtleCreek Azure portal interface. On the left is a sidebar menu with options: Dashboard, Devices, Device sets, Analytics, Jobs, App settings, Device Templates, Data export, and Administration. The main content area is titled 'Device TEST' and has tabs for Measurements, Settings, Properties (which is selected and highlighted with a dashed border), and Commands. Below the tabs is a 'Save' button. The Properties tab shows several system information fields, each with a help icon (i):

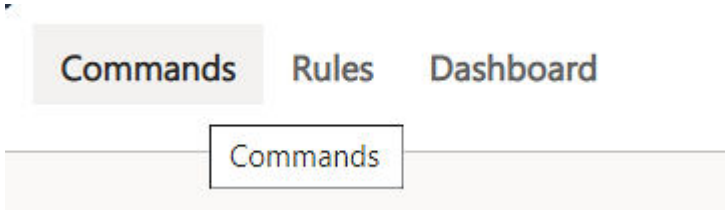
- System Product Name**: A text input field.
- OS Information**: A text input field containing 'Linux user-2 5.3.0-40-generic #32~18.04.1-Ubuntu SMP M'.
- Total Memory (bytes)**: A text input field containing '33525768192'.
- BIOS Vendor**: A text input field containing 'INSYDE Corp.'.
- BIOS Release Date**: A text input field containing '2019-05-10 00:00:00'.

Partial views of other fields are visible on the right side of the screen, including 'System Man...', 'CPU', 'Disk Inform...', and 'BIOS Versio...'.

Performing AOTA Updates Through the Azure Portal

Prerequisites: To perform this step, generate the EII bundle as mentioned in the **Generating EII Bundle for deployment** section. For testing, run the local http server as mentioned in the Creating an HTTP Local Server [ONLY FOR DEVELOPMENT] section. For actual implementation, you are need to have your own http server or any cloud services that provides a valid URL to fetch the EII bundle.

1. Open the Azure portal, select for the specific device that user will trigger the AOTA.
2. On the top menu bar, select the **Commands** tab.



- On the list command, fill up the trigger AOTA form as with details as follows. To perform this step, the EII bundle needs to be generated as mentioned in the **Generating EII Bundle for deployment section**.

Trigger AOTA ⓘ

App (docker, compose)

compose

Command (down, import, load, pull, snapshot, up, list, stats, remove)

up

Container Tag

eis_bundle

Fetch

http://10.221.40.65:8000/eis_bundle.tar.gz

- Above example assumes that the http server is being running at **http://10.221.40.65:8000/eii_bundle.tar.gz** and the EII bundle is generated using the default tag. You can leave all other sections empty and click **Run**.

The sample PCB demo example of the EII Visualizer application should appear on the new node. If the visualizer does not appear, run the following command on the new node terminal.

```
xhost +
```

You can check if EII has been successfully deployed on the new node. Run the following commands to view the list of running containers.

```
docker ps
```

The EII containers that are running are listed as output in the terminal.

You can also stop EII from running on the new node by repeating the AOTA method. Instead of passing command up parameter, you can pass command down parameter and the correct eii_bundle tag. You can check that EII has stopped by checking the list of running containers.


```

Activities  Terminal ▾

File Edit View Search Terminal Help

** (gedit:8403): WARNING **: 16:26:03.110: Set document metadata failed: Set
** (gedit:8403): WARNING **: 16:26:03.110: Set document metadata failed: Set
** (gedit:8403): WARNING **: 16:26:03.360: Set document metadata failed: Set
** (gedit:8403): WARNING **: 16:26:03.360: Set document metadata failed: Set
** (gedit:8403): WARNING **: 16:26:03.630: Set document metadata failed: Set
** (gedit:8403): WARNING **: 16:26:03.630: Set document metadata failed: Set
** (gedit:8403): WARNING **: 16:26:03.837: Set document metadata failed: Set
** (gedit:8403): WARNING **: 16:26:03.837: Set document metadata failed: Set
** (gedit:8403): WARNING **: 16:26:04.400: Set document metadata failed: Set
user@user-2:~$ systemctl restart configuration
user@user-2:~$ systemctl daemon-reload
user@user-2:~$ sudo gedit /etc/intel_manageability.conf

** (gedit:9172): WARNING **: 16:27:22.683: Set document metadata failed: Set
** (gedit:9172): WARNING **: 16:27:22.683: Set document metadata failed: Set
** (gedit:9172): WARNING **: 16:27:22.910: Set document metadata failed: Set
** (gedit:9172): WARNING **: 16:27:22.911: Set document metadata failed: Set
** (gedit:9172): WARNING **: 16:27:23.082: Set document metadata failed: Set
** (gedit:9172): WARNING **: 16:27:23.082: Set document metadata failed: Set
** (gedit:9172): WARNING **: 16:27:23.341: Set document metadata failed: Set
** (gedit:9172): WARNING **: 16:27:23.341: Set document metadata failed: Set
** (gedit:9172): WARNING **: 16:27:24.112: Set document metadata failed: Set
user@user-2:~$ sudo systemctl daemon-reload
user@user-2:~$ sudo systemctl daemon-reload
user@user-2:~$ history | grep xhost
    117  xhost +
    326  xhost +
    473  xhost +
    497  xhost +
    525  history | grep xhost
user@user-2:~$ docker ps
CONTAINER ID        IMAGE
ebb9cfff25d71      10.221.40.65:5000/ia_video_analytics:2.1
a2f523e29f02      10.221.40.65:5000/ia_visualizer:2.1
0980b3c07796      10.221.40.65:5000/ia_video_ingestion:2.1
COMMAND
"VideoAnalyti
"python3.6 vi
"VideoIngesti

```

Verifying Triggered AOTA in Event

Once an AOTA event is triggered, you can verify the log of the triggered call. This can be one of the verification during development phase.

1. Go to the Device Manageability machine and run the following command to view the log of commands:

```
journalctl -fu dispatcher & journalctl -fu cloudadapter
```

2. Note the event logs on the Azure portal server showing which commands have been run.

NOTE If the event log is not displayed then, follow the steps below.

```
Change settings from ERROR to DEBUG everywhere in the following files.
(Only for Development Purpose)
/etc/intel-manageability/public/dispatcher-agent/logging.ini
/etc/intel-manageability/public/cloudadapter-agent/logging.ini
```

3. Run the following commands

```
sudo systemctl restart dispatcher
sudo systemctl restart cloudadapter
sudo systemctl restart diagnostic
```

Cloud Service: ThingsBoard* Setup

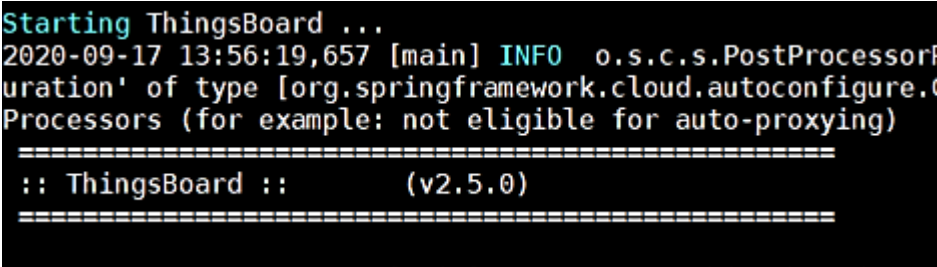
All Device Manageability devices can be added into ThingsBoard to provision or enable AOTA updates.

Starting ThingsBoard* as Docker [macOS*/Linux*]

1. Choose an edge device to run as the ThingsBoard server. The edge devices for ThingsBoard machine can be either within the cluster setup or any other edge devices.
2. Run the following command to start ThingsBoard on the ThingsBoard machine.

```
docker run -it -p 9090:9090 -p 1884:2883 -p 5683:5683/udp -v ~/.mytb-data:/data -v ~/.mytb-logs:/var/log/thingsboard -e MQTT_BIND_PORT=2883 -name mytb -restart always thingsboard/tb-postgres:2.5.0
```

Assuming that ThingsBoard has started correctly, you should expect to see the terminal output as shown in figure below.



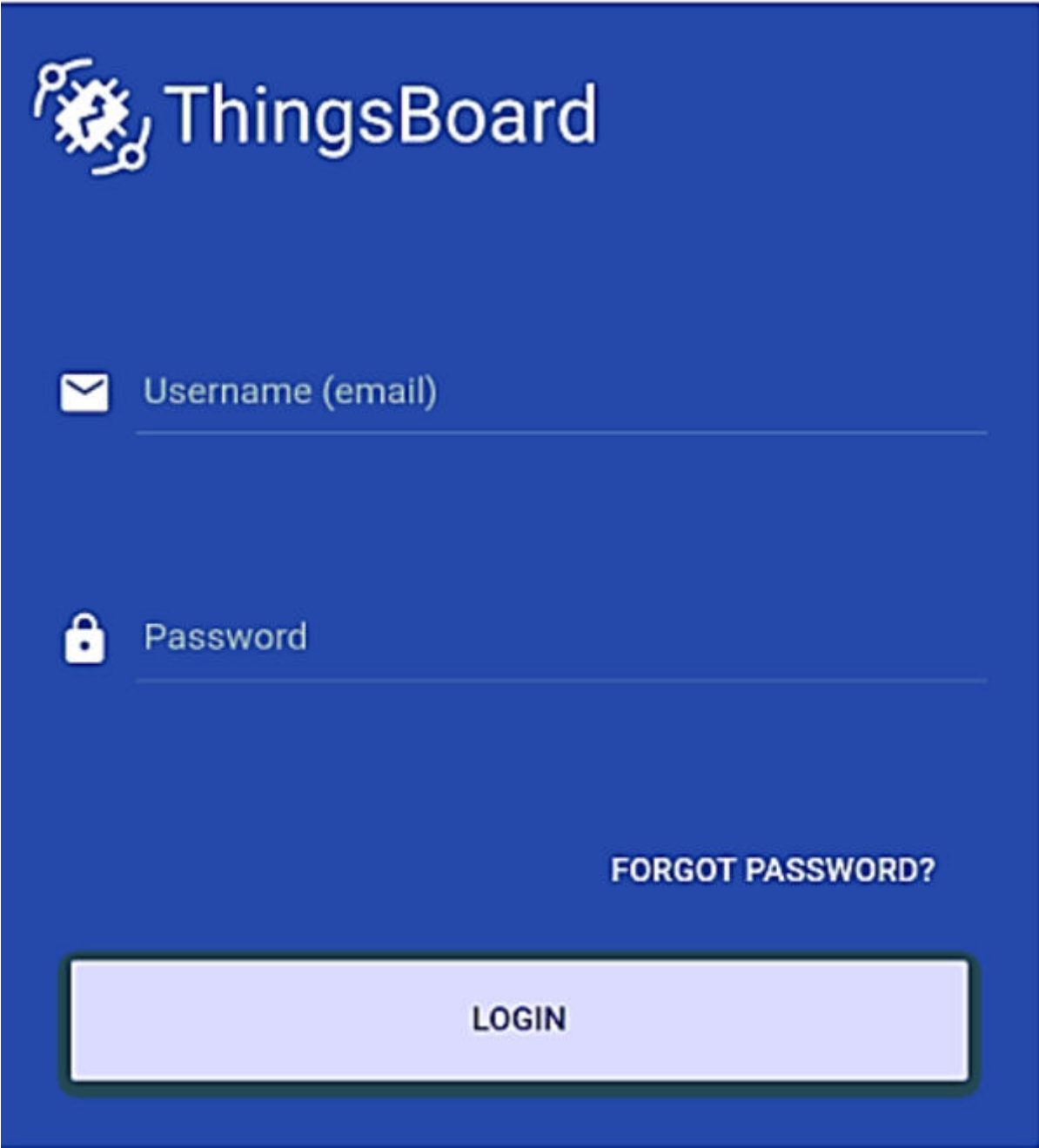
```
Starting ThingsBoard ...
2020-09-17 13:56:19,657 [main] INFO o.s.c.s.PostProcessor
uration' of type [org.springframework.cloud.autoconfigure.
Processors (for example: not eligible for auto-proxying)
=====
:: ThingsBoard :: (v2.5.0)
=====
```


3. After executing the command, you can browse the ThingsBoard server:


```
<http://<host-ip>:9090>
```


Example: <http://10.221.40.48:9090>

4. The ThingsBoard login page is displayed as follows:

The image shows the ThingsBoard login interface. It has a solid blue background. At the top left is the ThingsBoard logo, which consists of a white gear-like icon with a person figure inside, followed by the text 'ThingsBoard' in a white sans-serif font. Below the logo are two input fields. The first field is preceded by a white envelope icon and the text 'Username (email)'. The second field is preceded by a white padlock icon and the text 'Password'. To the right of the password field, the text 'FORGOT PASSWORD?' is displayed in white. At the bottom center, there is a large, light blue rectangular button with a dark blue border and the word 'LOGIN' in dark blue capital letters.

 ThingsBoard

 Username (email)

 Password

FORGOT PASSWORD?

LOGIN

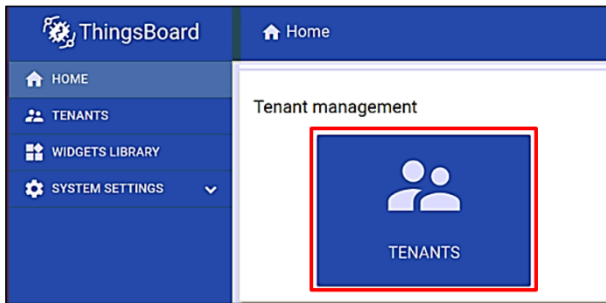
5. Enter the default username and password to access to the ThingsBoard home.

Username: `sysadmin@thingsboard.org`

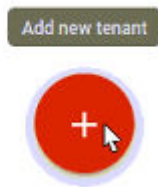
Password: `sysadmin`

Creating a ThingsBoard* Account

1. On the homepage, click **Tenant management**.



2. Click the plus button in the bottom right to add a new tenant. Refer to the following:



3. The Add Tenant form is displayed as follows. Complete the form, then click **add**.

Add Tenant

Title *

Description

Country ▼

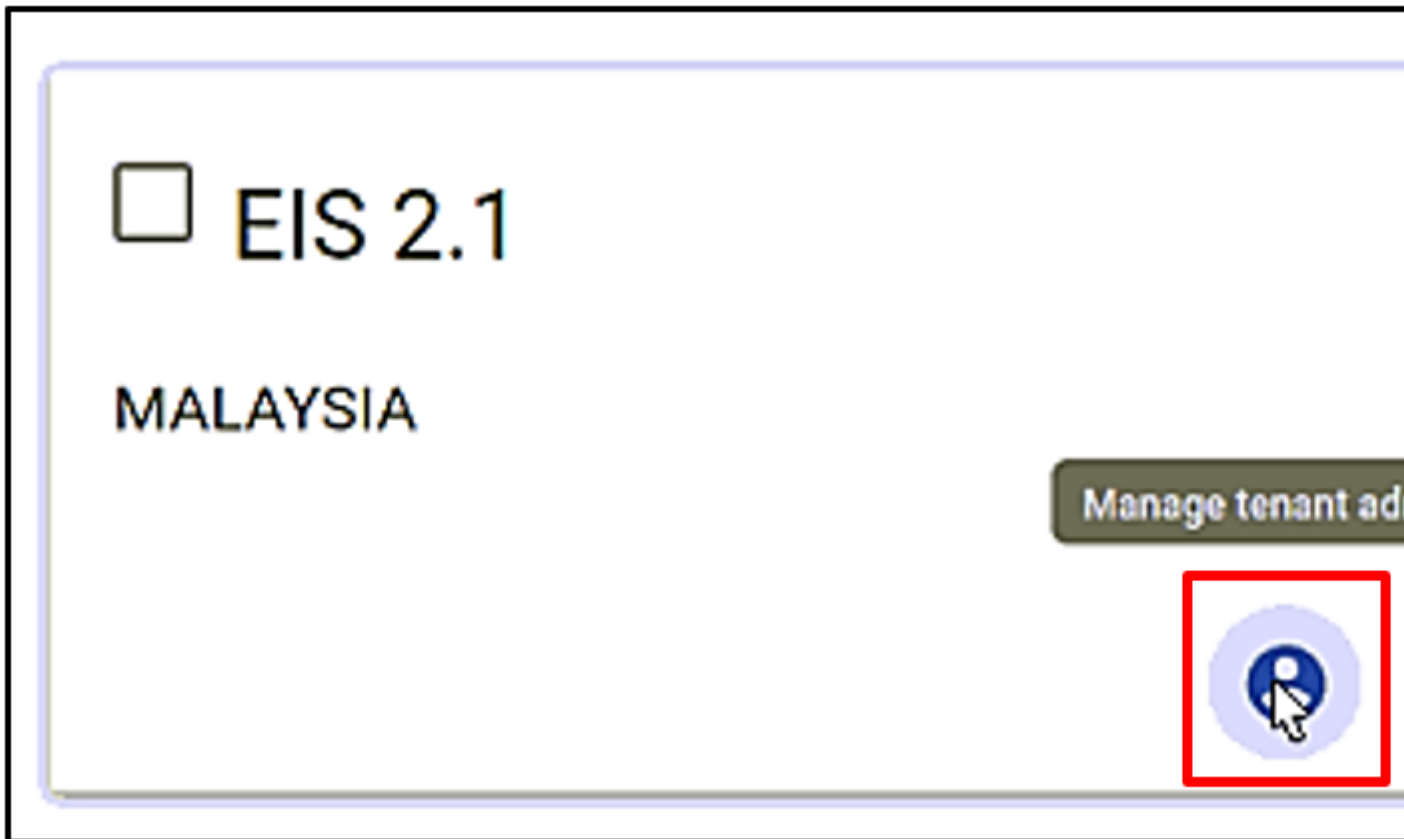
City State / Province Zip / Postal Code

Address

Address 2

ADD CANCEL

4. New tenant entry will appear in the tenant section as shown in figure below. Click **Manage tenant admins** to create a new user sign-in username and password.



5. Add User form will appear as shown in figure below. Complete the form accordingly. For the Activation method, click **display activation link** and then, click **ADD**.

User activation link

In order to activate user use the following [activation link](#) :

`http://10.221.40.48:9090/api/noauth/activate?activateToken=a71J8dEkr0j9UhSWv`

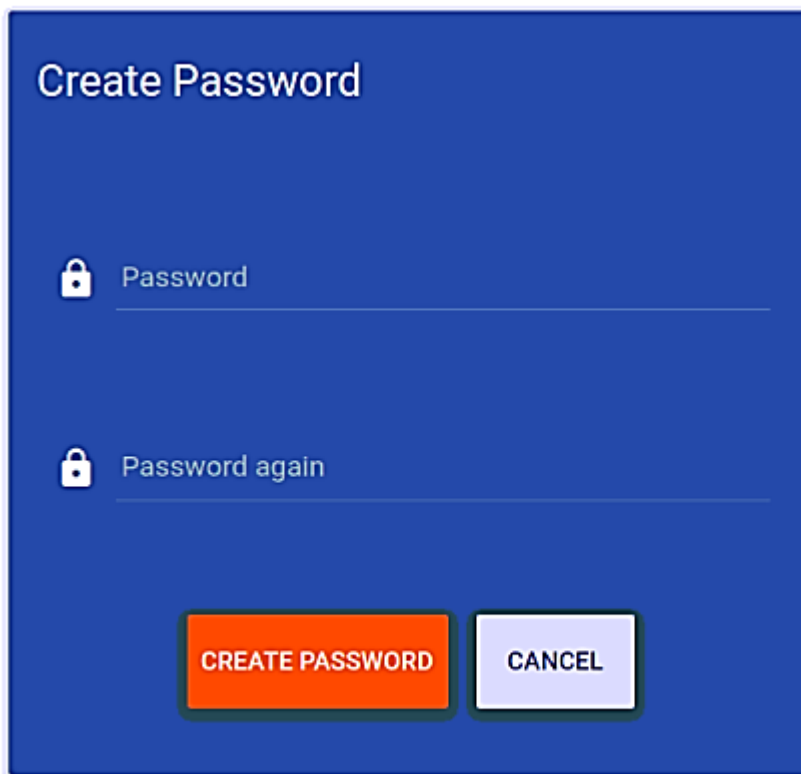
6. User activation link box will appear as shown in figure below. Click the activation link.

User activation link

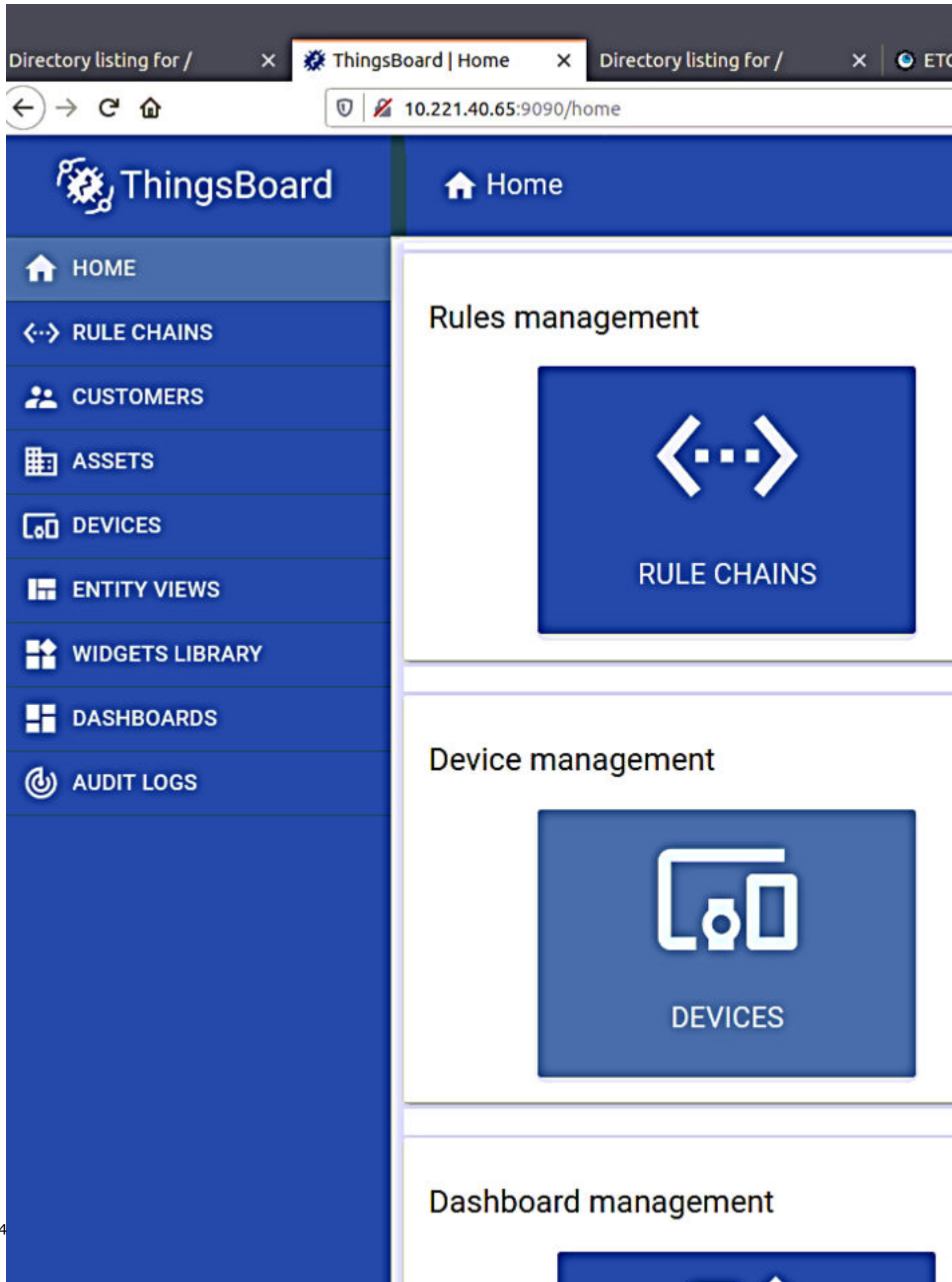
In order to activate user use the following [activation link](#) :

`http://10.221.40.48:9090/api/noauth/activate?activateToken=a71J8dEkr0j9UhSWv`

7. You will be redirected to create a new sign-in password. Click **create password** to create a new password as shown in the following figure:

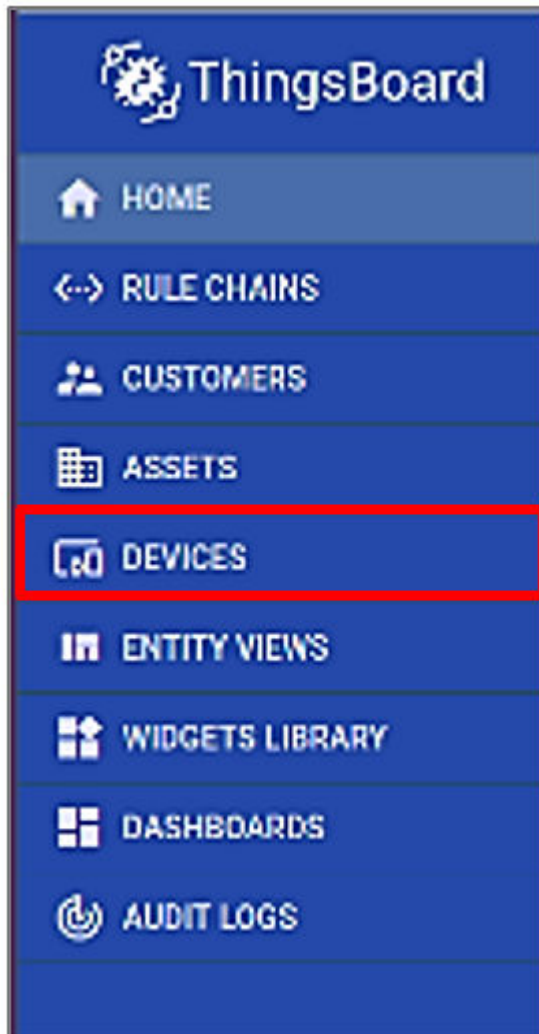
A screenshot of a 'Create Password' form. The form has a blue background. At the top, the title 'Create Password' is displayed in white. Below the title, there are two input fields, each preceded by a white padlock icon. The first field is labeled 'Password' and the second is labeled 'Password again'. At the bottom of the form, there are two buttons: an orange button labeled 'CREATE PASSWORD' and a light blue button labeled 'CANCEL'.

8. You can now sign in using the tenant username and password, after which you will be redirected to the ThingsBoard dashboard page as shown in the following figure:

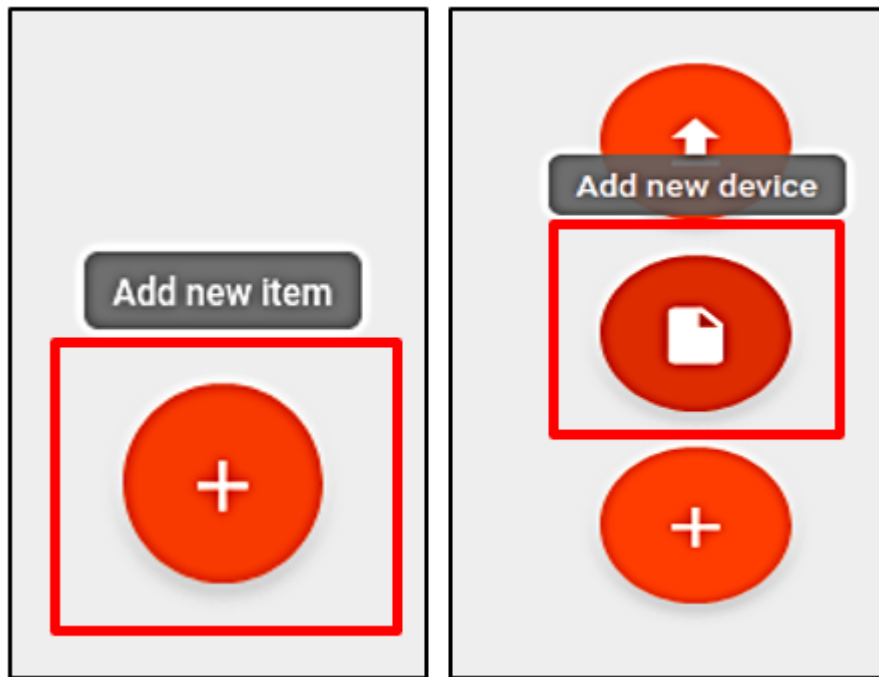


Adding Device Manageability Device to ThingsBoard*

1. Add the new Device Manageability device on ThingsBoard.
 - a. Log into the ThingsBoard page. On the left sidebar, click **Devices** as shown in the following figure:



- b. Click the 'plus' button on the bottom right and select **Add new device**.



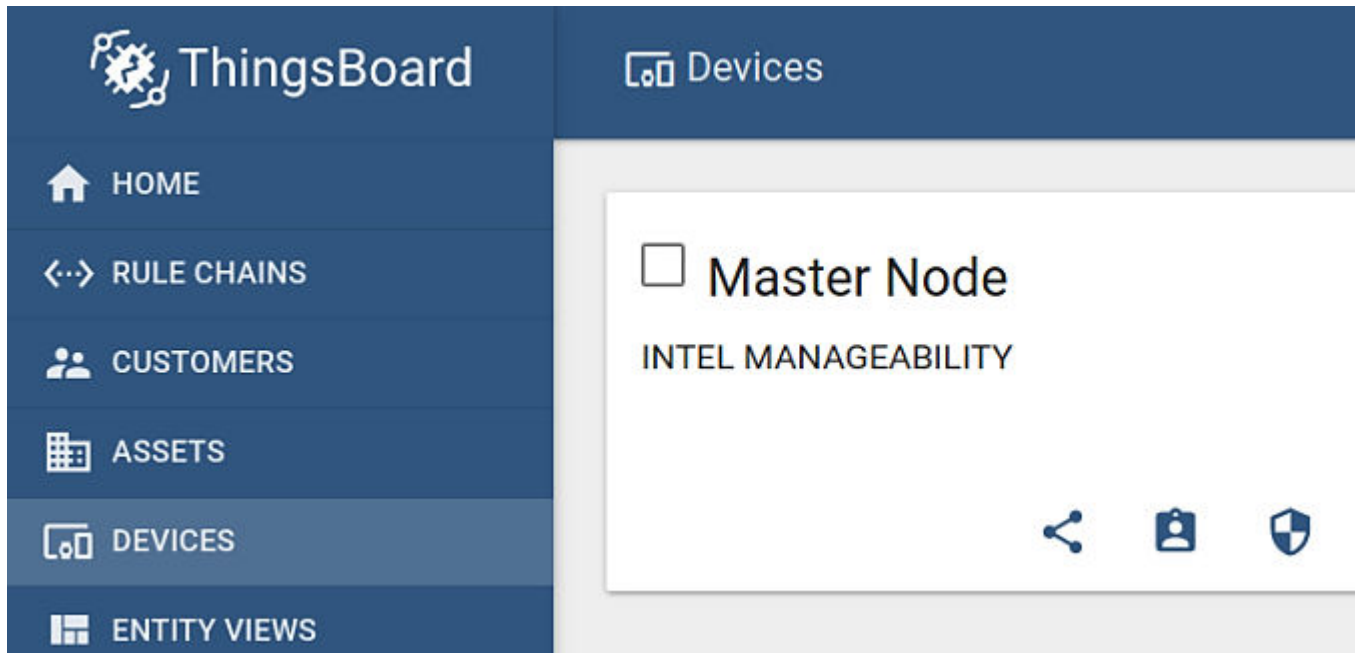
- c. On the **Add device** window, complete the fields, then click **ADD**.

The image shows two side-by-side 'Add Device' forms. Both forms have a dark blue header with the title 'Add Device', a help icon (?), and a close icon (X). The forms contain the following fields:

- Name ***: A text input field. In the right form, it contains 'Master Node'.
- Device type ***: A text input field. In the right form, it contains 'Intel Manageability', which is highlighted with a red rectangular box. A close icon (X) is visible to the right of the text.
- Label**: A text input field.
- Is gateway**: A checkbox.
- Description**: A text input field.

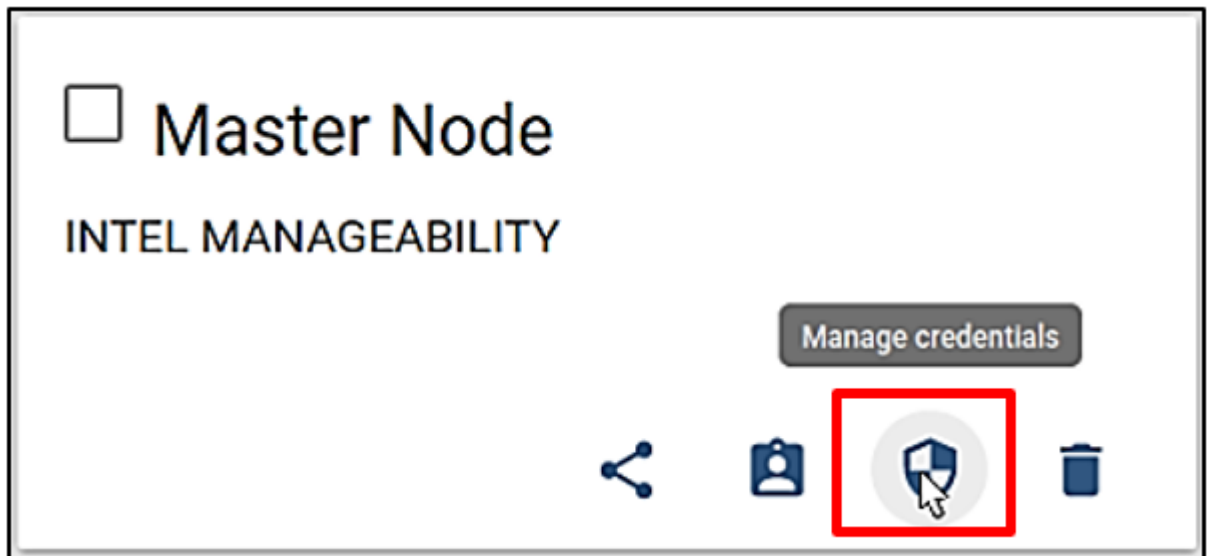
At the bottom of each form are two buttons: 'ADD' and 'CANCEL'. In the right form, the 'ADD' button is highlighted in blue, while the 'CANCEL' button is grey.

- d. Newly added device will be shown on the devices page as shown in figure below.

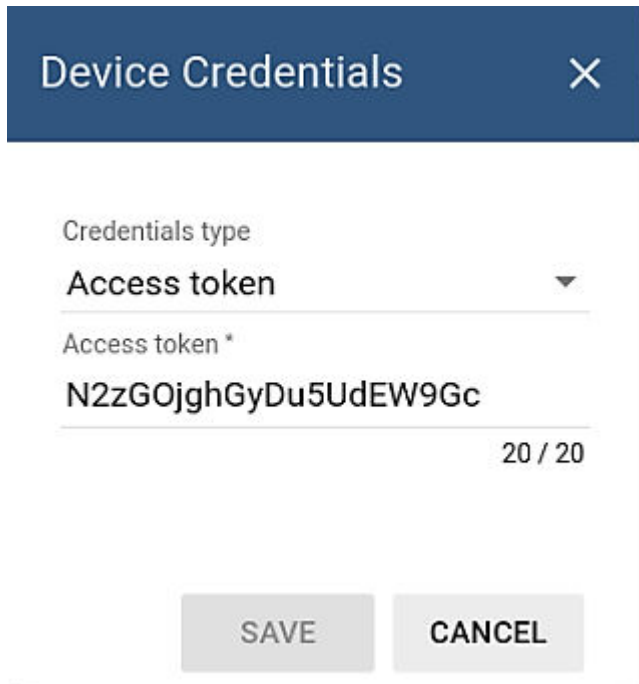


2. Obtaining device credentials.

- a. Click the manage credentials icon as shown in figure below.



- b. Device credentials window is displayed as follows. Access token is required for provisioning purposes.

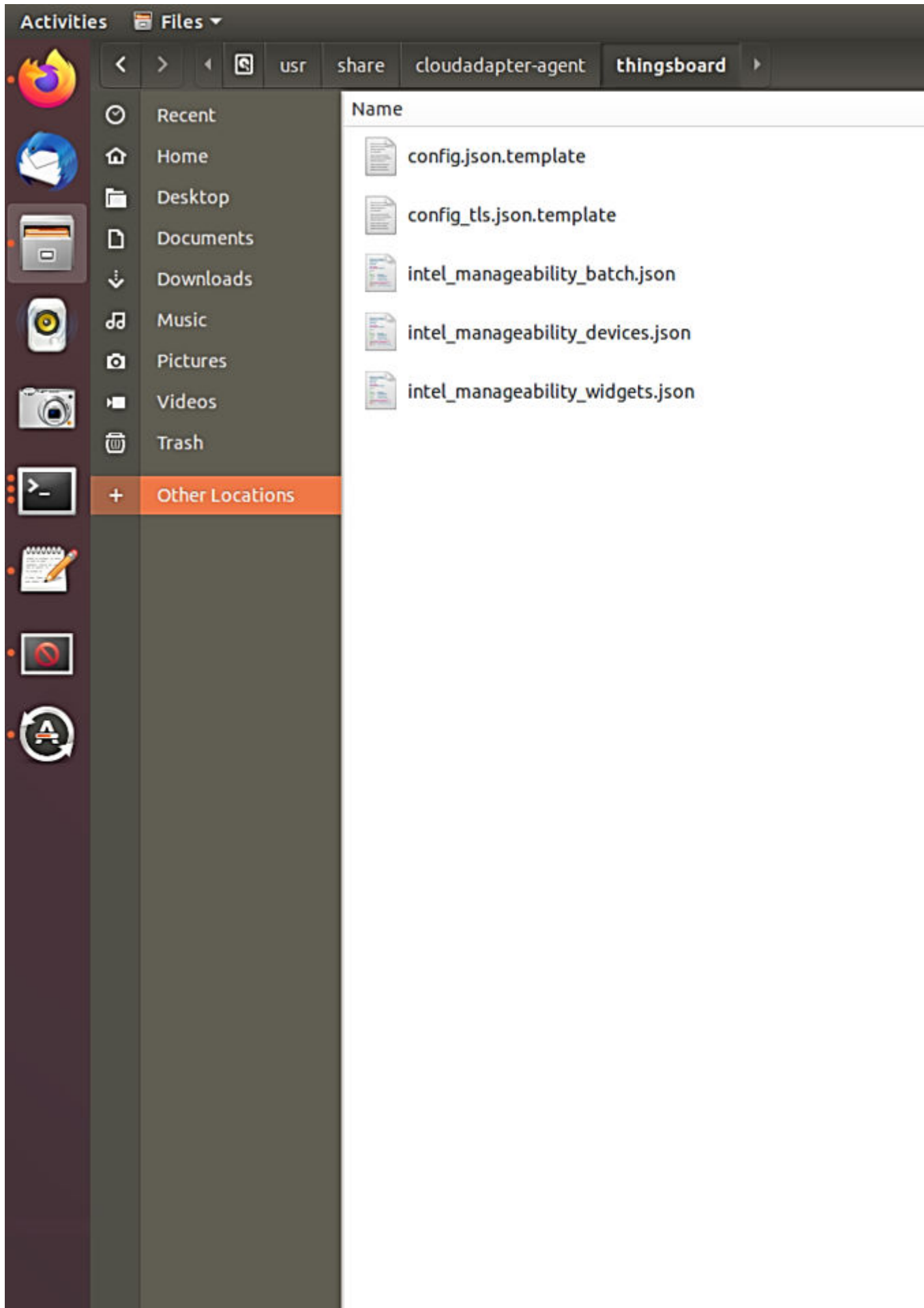


The image shows a 'Device Credentials' dialog box with a dark blue header and a close button (X). Inside, there is a 'Credentials type' dropdown menu set to 'Access token'. Below it is an 'Access token *' text field containing the value 'N2zGOjghGyDu5UdEW9Gc'. A character count '20 / 20' is displayed to the right of the text field. At the bottom, there are two buttons: 'SAVE' and 'CANCEL'.

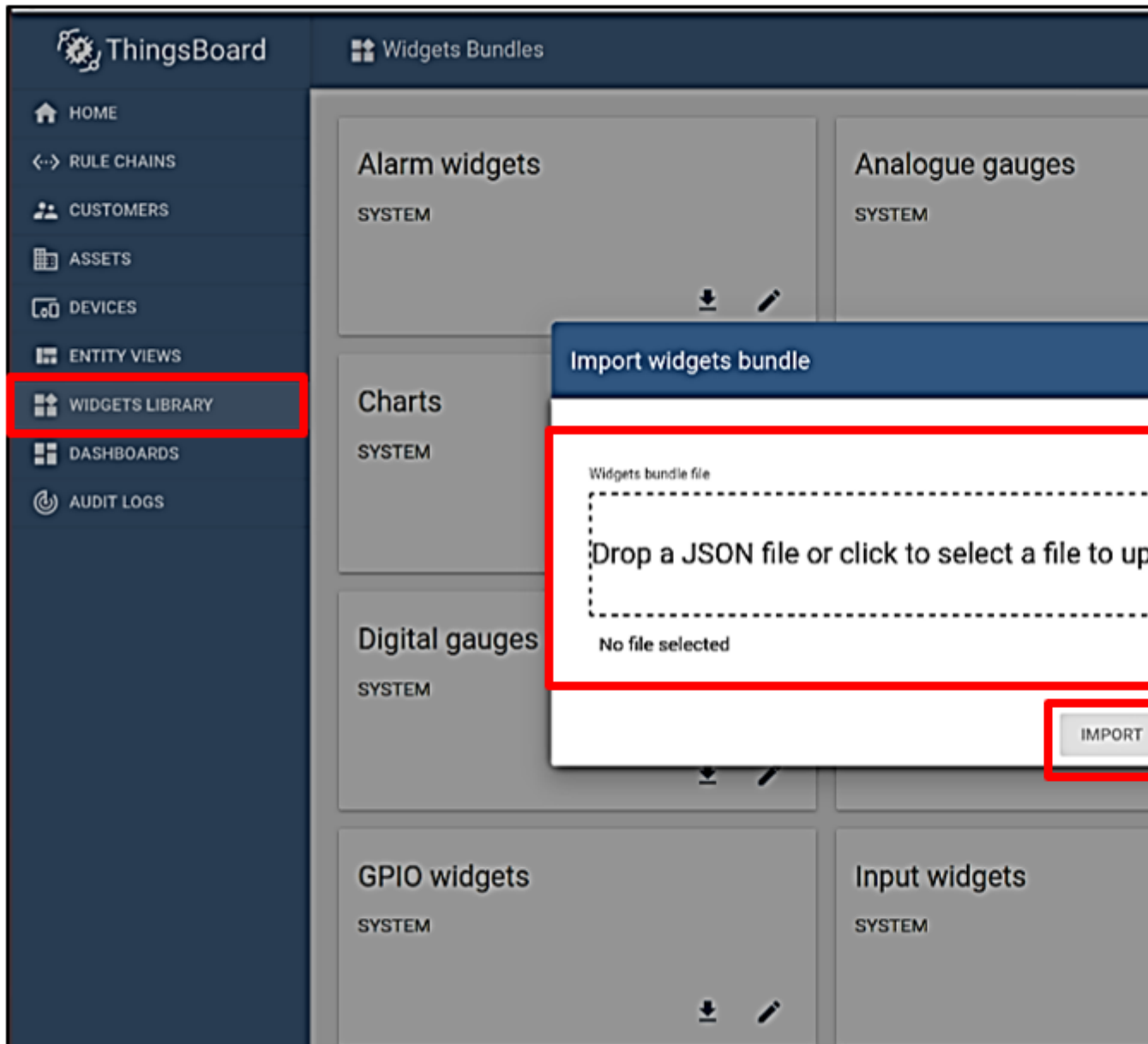
3. Setting up the dashboard.

- a.** Some files need to be copied from the Device Manageability machine to ThingsBoard page in the next step. These files can be found in the following directory.

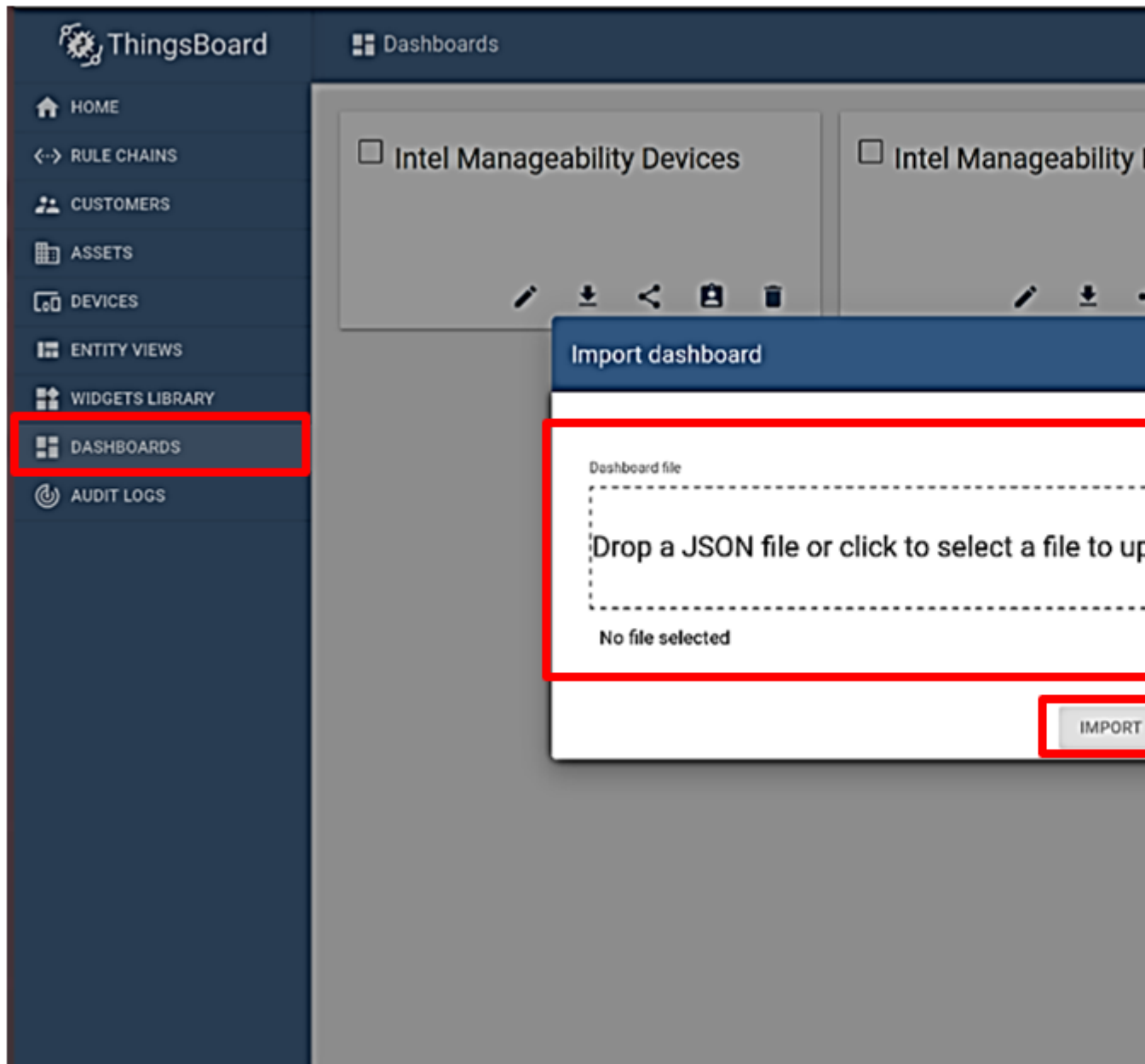
```
cd /usr/share/cloudadapter-agent/thingboard
```



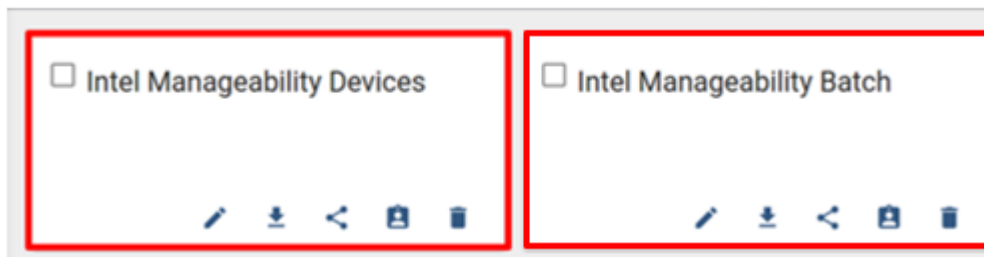
- b. On the left side bar menu, click the **Widget Library**, then click **Add new widget** in the bottom right, and the **Import widgets bundle**. Drag the `intel_manageability_widget.json` file into the box provided and then click **Import**.



- c. On the left sidebar menu, click on **Dashboard**, then click **Add new dashboard**** in the bottom right. Drag `intel_manageability_devices.json` into the box provided and then click **Import**. Repeat this step for `intel_manageability_batch.json`.



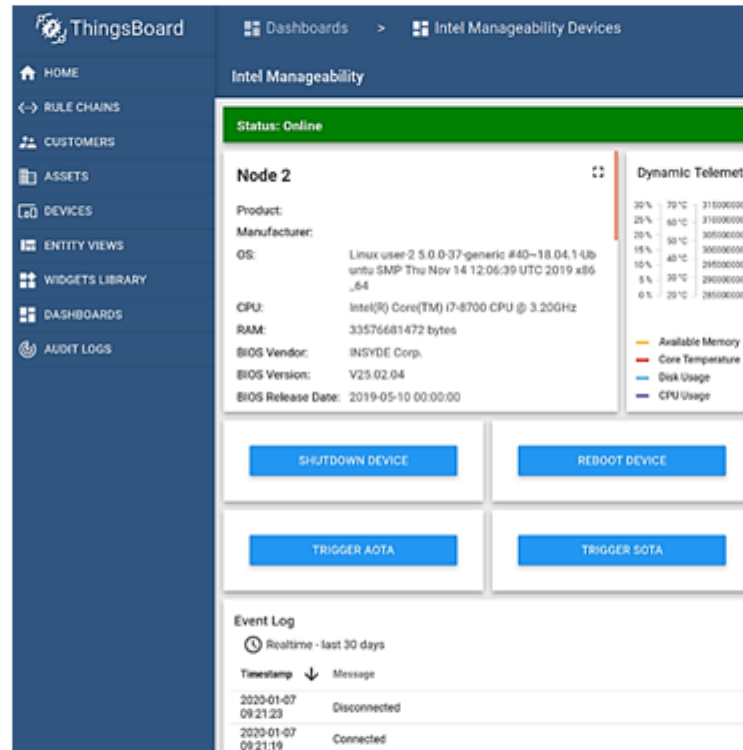
- d. Both newly added dashboards will be displayed.



- e. To access the dashboard, click **Dashboard** on the left sidebar, then click **Intel Manageability Devices**. Basic information about the added devices is available here.

If you have more than one device, then you can choose which device to view by clicking the highlighted button at the top right.

Buttons are also available in the lower part of the screen to view the entire event log, to search for specific event logs, or to expand the window to full screen.



Provisioning a Device

Prerequisites: To perform this step, Device Manageability needs to be installed following the instructions in the Installation - Device Manageability section.

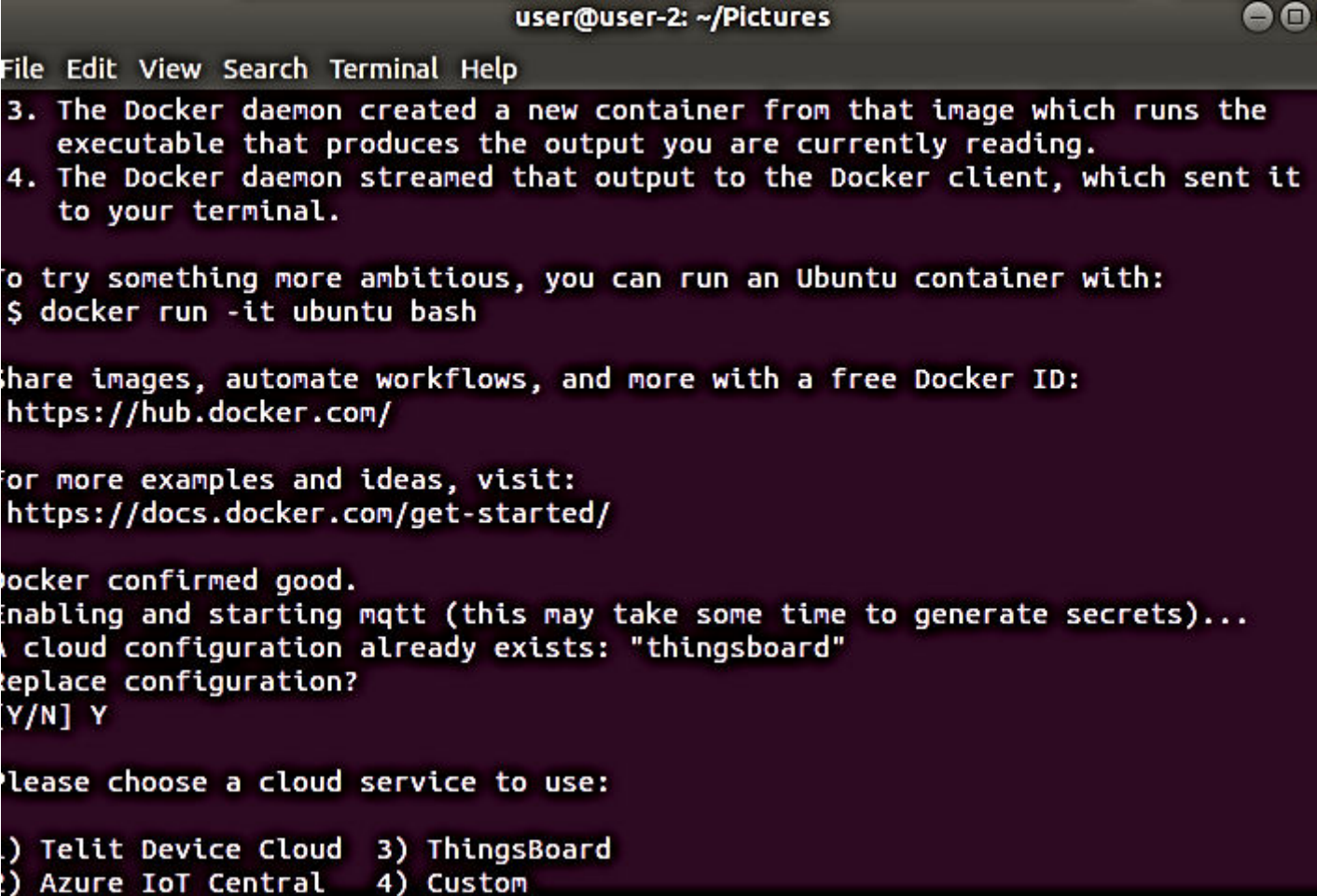
You must provision the Device Manageability device for it to connect to ThingsBoard.

1. Provisioning can be done with or without TPM security by setting 'PROVISION_TPM'. Run the following commands to set 'PROVISION_TPM' to the Device Manageability device:
 - auto: use TPM if present; disable if not present; do not prompt.
 - disable: do not use TPM.
 - enable: use TPM; return error code if TPM not detected.
 - (unset): default behavior; use TPM if present, prompt if not.
2. On the Device Manageability device, run the following command, change the bold parameter accordingly as mentioned above.

```
sudo PROVISION_TPM=auto provision
```

Read though the license and press **Y** to accept.

3. If the device was previously provisioned, the following message will appear. To override the previous cloud configuration, press **Y**.



```
user@user-2: ~/Pictures
File Edit View Search Terminal Help
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://hub.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/get-started/

Docker confirmed good.
Enabling and starting mqtt (this may take some time to generate secrets)...
A cloud configuration already exists: "thingsboard"
Replace configuration?
[Y/N] Y

Please choose a cloud service to use:

1) Telit Device Cloud   3) ThingsBoard
2) Azure IoT Central   4) Custom
```

4. Select ThingsBoard as the cloud service by pressing **3** and **[ENTER]**.


```
Please choose a cloud service to use:
```

```
1) Telit Device Cloud   3) ThingsBoard
2) Azure IoT Central   4) Custom
#? 3
```

```
Configuring to use ThingsBoard...
```

5. A prompt for Device provision type appears; select the type of device authentication preferred: Choose 1 for Token authentication. If you choose option2 - Refer section [2.5 - Provisioning a Device] at ./docs/In-Band_Manageability_UserGuide_ThingsBoard.pdf for further steps.
6. Provide the IP address of the server running ThingsBoard.

```
Please enter the server IP:
10.221.40.48
```

7. Provide the device token based on Step 3b.

```
Please enter the device token:
N2zGOjghGyDu5UdEW9Gc
```

8. Use 1884s for the port configuration.

```
Please enter the server port (default 1883):
1884
```

9. Prompts for 'Configure TLS' will appear and enter 'N' for this.

```
Configure TLS?
[Y/N] N
```

10. The following screen will appear if cloud provisioning has been completed successfully.

```
Successfully configured cloud service!

Enabling and starting agents...
Turtle Creek Provisioning Complete
```

The script will then start the Intel® Manageability Services. When the script finishes, you should be able to interact with the device via the ThingsBoard dashboard.

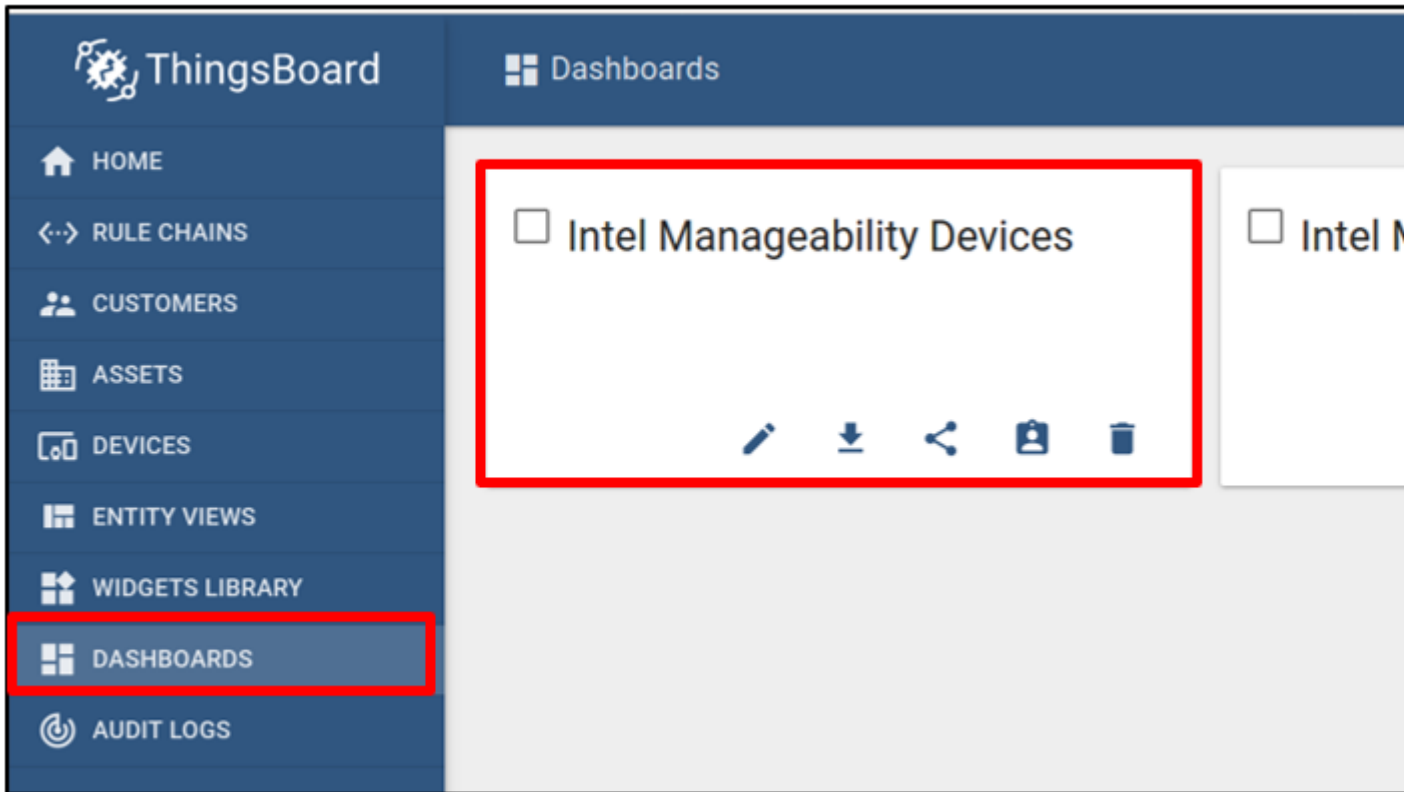
If at any time the cloud service configuration needs to be changed or updated, run this provisioning script again.

To add more than one device, repeat each of the previous steps, except for Step 4 (Setting Up the Dashboard).

Performing AOTA Updates Through the ThingsBoard* Page

Prerequisites: In order to perform this step EII bundle need to be generated as mentioned in section Generating EII Bundle for deployment and local http server (for development) need to be run as mentioned in Creating an HTTP Local Server [ONLY FOR DEVELOPMENT].

1. Open the ThingsBoard page, then click on the **Dashboard** page. Select **Intel Manageability Devices**.



2. Select one of the previously added Device Manageability devices. In this case, the user can select the new node (node 2) in which they want to deploy the eii_bundle.
3. Click **Trigger AOTA**.



4. Once the Trigger AOTA window opens, complete each field per the information below:

Trigger AOTA

METHOD	RESULTS
App	
<div><div>docker-compose</div><div></div></div>	
Command	
<div><div>up</div><div></div></div>	
Container Tag	
<div><div>eis_bundle</div><div></div></div>	
Fetch	
<div><div>http://10.221.40.65:8000/eis_bundle.tar.gz</div><div></div></div>	
Signature	
Version	
Server Username	
Server Password	

For the fetch information, provide the local HTTP server being set up in section Creating an HTTP Local Server [ONLY FOR DEVELOPMENT]. The example uses **http://10.221.40.48:9000/eii_bundle.tar.gz** in this case.

Leave the other section empty and click **Send**.

In the step above, the new node will access the primary node through the local HTTP server that being set up to fetch the eii_bundle.

In the ThingsBoard log, the user can see that the eii_bundle has been fetched from the local server and was deployed successfully:

```
{'status': 200, 'message': 'COMMAND SUCCESS'}
```

```
Package: http://10.221.40.65:8000/eis_bundle.tar.gz Fetch Result: {'status': 200, 'message': 'OK'}
```

The sample PCB demo example of the EII Visualizer application should appear on the new node. In case the visualizer does not appear, run the following command on the new node terminal.

```
xhost +
```

The user can also verify that EII has been successfully deployed on the new node by checking the list of running containers using the following command:

```
docker ps
```

This action should yield the following terminal output listing the running EII containers.

CONTAINER ID	IMAGE	COMMAND
ebb9cff25d71	10.221.40.65:5000/ia_video_analytics:2.1	"VideoAnalytics/va_
a2f523e29f02	10.221.40.65:5000/ia_visualizer:2.1	"python3.6 visualiz
0980b3c07796	10.221.40.65:5000/ia_video_ingestion:2.1	"VideoIngestion/vi_
d8aff3d38637	ia_etcd:2.1	"./start_etcd.sh"
a0c2e6174211	registry:2	"/entrypoint.sh /et

The user can also stop EII from running on the new node by repeating the AOTA method. Instead of passing command up parameter, the user can pass the command down parameter and the correct eii_bundle tag. The user can verify that EII has stopped by checking on the list of running containers.

Verifying Triggered AOTA in Event

Once an AOTA event is triggered, we can verify the log of the triggered call. This can be one of the verification done during development phase.

1. Go to the Device Manageability machine and run the following command to view the log of commands:

```
journalctl -fu dispatcher & journalctl -fu cloudadapter
```

2. Note the event logs on the ThingsBoard server showing which commands have been run.

NOTE If the event log does not appear, follow the steps below.

```
Change settings from ERROR to DEBUG everywhere in below files.
(Only for Development Purpose)
/etc/intel-manageability/public/dispatcher-agent/logging.ini
/etc/intel-manageability/public/cloudadapter-agent/logging.ini
```

3. Run the following commands

```
sudo systemctl restart dispatcher  
sudo systemctl restart cloudadapter
```

Cloud Service: Telit* Cloud Setup

It is necessary to creating a Telit account and to receive an org token from Telit to provision/enable AOTA updates. You will also need to import the Thing Definition, which provides the buttons created to support AOTA functionality in EII.

A connection to Telit can only be made if the user has a group/organization on the service. Visit the deviceWise domain to create an account and get an "org" or "application".

Create Your Telit* IoT Portal Account

Telit provides a Getting Started Guide for creating an account. This guide can be referenced here:

<https://docs.devicewise.com/Content/GettingStarted/IoT-Portal-Part-1—Creating-your-account.htm>

After setting up your password and accepting Telit's terms, the first time you click on Things you will be asked to set up live updates. Although this is optional, Intel recommends setting the frequency of live updates to 5 seconds.

Live updates for Things

Things support live updating. You can have the list of Things, or an individual Thing, automatically update live while leaving the page open.

This feature can be enabled/disabled and timing adjusted from your User profile.

To set this up now, just select Enable and a frequency below.

Note: Use of this feature will result in increased API usage.

The screenshot shows a user profile settings interface. At the top, there is a checkbox labeled "Enable live update for things" which is checked. Below this, there is a section titled "Frequency" with a dropdown menu currently set to "5 seconds". At the bottom of this section is a blue button labeled "Update".

Import Thing Definition

Pre-Requisites: In order to perform this step Device Manageability need to be installed based on Installation Device Manageability section.

To work with EII's AOTA capability, import INTEL-MGB as a Thing.

1. Login to Telit* at <https://portal.telit.com/app/login>
2. Ensure the correct org in the top right corner, left of the gear wheel.
3. Select the **Developer** tab along the top bar.
4. Select **Thing Definitions** at the bottom left sidebar menu.
5. From the **Things** sub-screen, find **Import** in the upper-right corner.
6. Click **Attach File** and then select ****thingsdefinition.json*** that is available inside the turtlecreek repository.

7. Click **Import**. You should see the new Things definition added to your account.

Thing definitions

20 ▼ 15 thing definitions found.

	Name ↓
   	Device Manager
   	INTEL_MGB

8. The new Thing definition should now be visible within the Telit list of things under **Things definitions**, in the **Developer** tab.

Copy Application Token

Within the Telit portal, view the “Developer” tab and click **Applications** from the menu on the left. Copy and retain the token, which will be required during the provisioning script or step.




This token is used to link your Thing to the Telit cloud and is used during provisioning.

1. Log in to your Telit IoT Portal: <https://portal.devicecloud.windriver.com>

NOTE The first time you log in, you will be asked to Enable Live Updates for Things. Click the box, set 5 seconds for the setting.

2. Retrieve Token.
- Click **Developer**.
 - Click **Application**.
 - Copy the encrypted token.

Applications

	Name	Token
  	Default Application	3sqX*****

NOTE Alternately, you can paste the token into a text editor.

Provision with the Token

- Once you have copied your Application Token, you are ready to provision Device Manageability. You will need the provisioning script.
- Provisioning can be done with or without TPM security by setting 'PROVISION_TPM'. 'PROVISION_TPM' can be set to the turtlecreek device, users need to run below command.
 - auto: use TPM if present; disable if not present; do not prompt
 - disable: do not use TPM.
 - enable: use TPM; return error code if TPM not detected.

- (unset): default behavior; use TPM if present, prompt if not
3. On the Device Manageability device, users need to run the command below, change the bold parameter accordingly as mentioned above.
- a. `sudo PROVISION_TPM=auto provision-tc`
 - b. You will be asked to replace configuration, if there is any existing configured cloud service. Type 'Y'.
 - c. Choose Telit Cloud service.
 - d. Select development for Telit host.
 - e. Insert application token that have been copied on previous step.
 - f. Leave the Telit thing key blank.

User will be prompted by 'successfully configured cloud service' and 'Device Manageability provision complete' message.

```
Docker confirmed good.
Enabling and starting mqtt (this may take some time to generate secret.
A cloud configuration already exists: "telit"
Replace configuration?
[Y/N] y

Please choose a cloud service to use:

1) Telit Device Cloud    3) ThingsBoard
2) Azure IoT Central    4) Custom
#? 1

Configuring to use Telit...

Please select the Telit host to use:
1) Production (api.devicewise.com)
2) Development (api-dev.devicewise.com)
#? 2

Provide Telit token:
AnuIAwETthly6E6n

Provide Telit Thing Key (leave blank to autogenerate):

Thing Key: 4c017063-4af7-4df9-8f7b-39f991db7596

Successfully configured cloud service!

Enabling and starting agents...
Turtle Creek Provisioning Complete
```


4. Verify your Connectivity.

- a. Click **Things**.
- b. Verify that your system is now visible on Telit.

Things

defkey disconnected

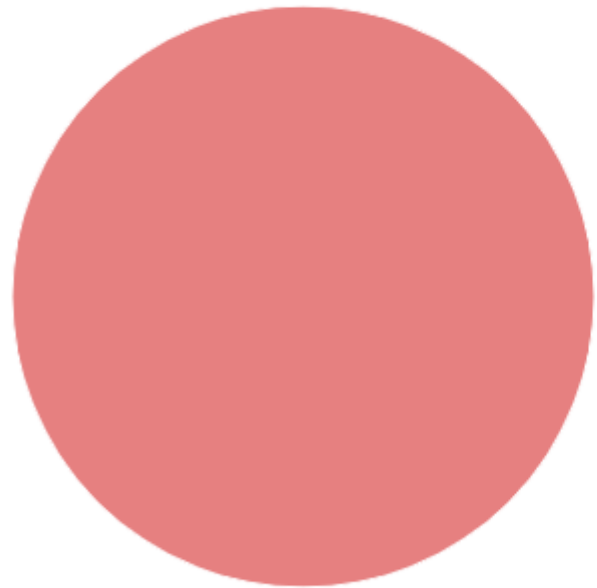
Search t

 Dashboard


 Table

 Map

Connected things



Recently added things

Key	Name	Added o
 6bce725f-c53f-43e7-86cc-732b1e011118- iot-dispatcher	auto:6bce725f-c53f-43e7-86cc- 732b1e011118-iot-dispatcher	on May 1

Once your device is identified as a Thing in Telit, your system can be updated through Telit.

Telit serves as the UI to support the following OTAs (Over-the-Air) without manual intervention:

- **AOTA** (Application Over the Air update)
- **FOTA** (Firmware Over the Air update)
- **SOTA** (Software/OS Over the Air update)
- **Config Update** (configuration parameter update)
- **Power Management** (Remote Shutdown and Restart)

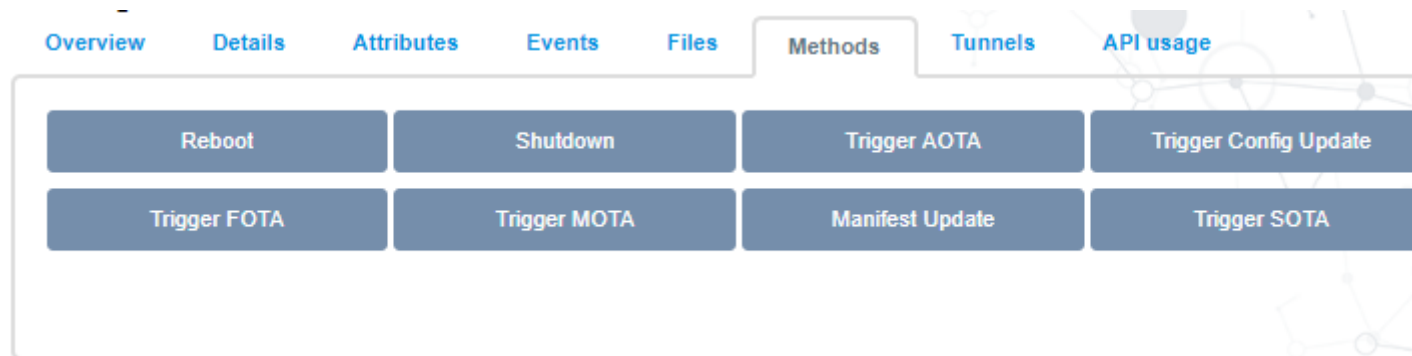
NOTE Only AOTA is currently supported and validated with EII.

Link to Thing definition

1. Set Thing definition
 - a. Click **Things**.
 - b. Click the **(eye)** icon for your Thing.
 - c. Set **Thing Definition**, then click **(book)**.
 - d. Choose definition: **INTEL_MGB**.
 - e. Submit, Okay.



2. After the Things definition is installed, you will see the **Methods** tab is enabled, which will show you the widgets that EII supports.



NOTE Your 'Methods' might look different.

The Telit website provides instructions for navigating the user interface to complete the steps of importing a thing definition and performing required configuration tasks. The following steps are generalized, as the name and placement of specific buttons and menus within the Telit website can change without notice.

View and Name Your Thing

1. Name your Thing (optional)
 - a. Click **Action**.
 - b. Click **Edit**.
 - c. Under Name* <enter name>.
 - d. Update, Okay

Overview

Details

Attributes


Events

Files

Id

5cf94dae14c97843

Thing definition



INTEL_MGB 

Key

1350391e-621f-447

NOTE This helps to identify your device.

Recently added things

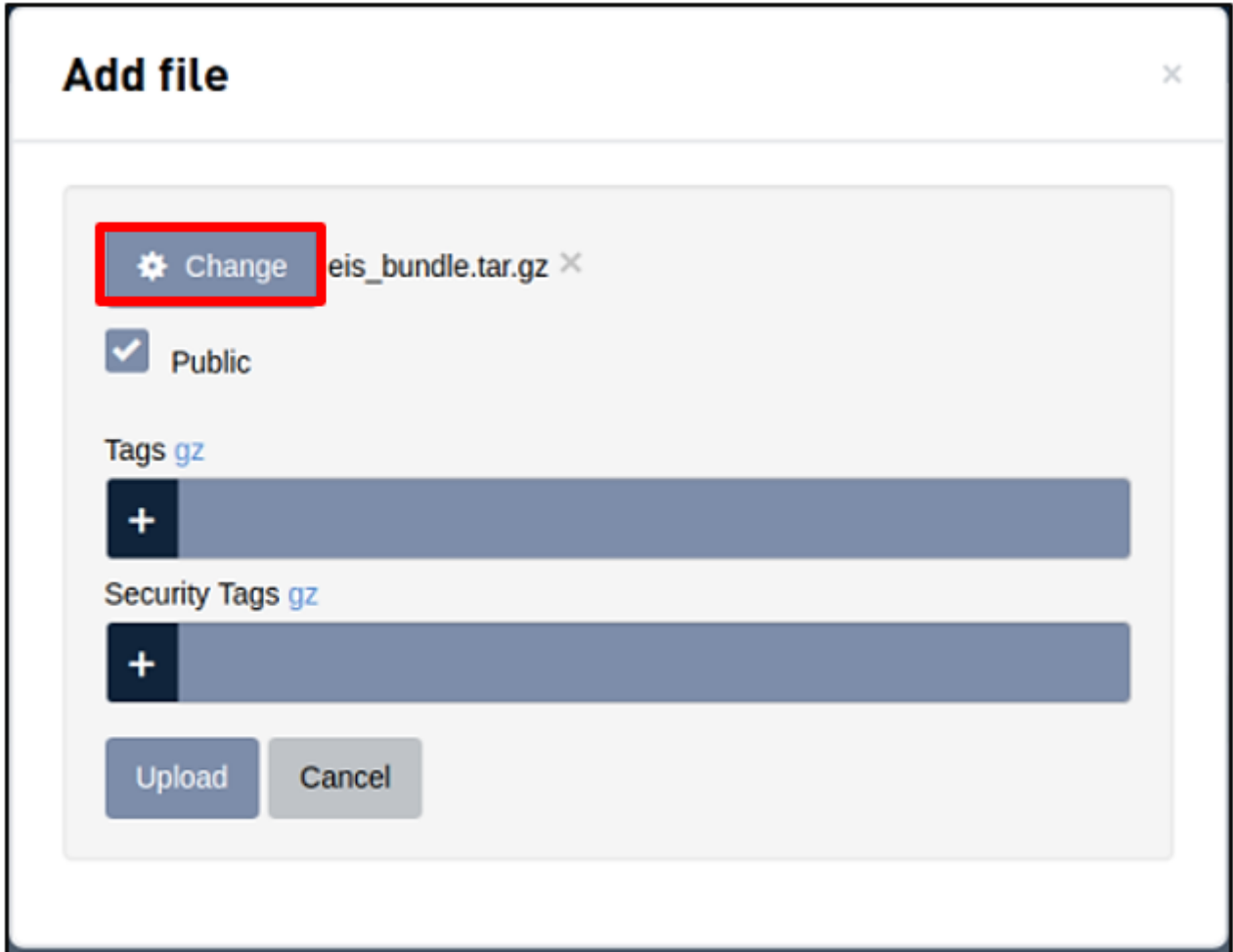
Key	Name	Added on
 8795ef55-8f4c-4ab3-a68f-7cae1fba51a8- iot-dispatcher	auto:8795ef55-8f4c-4ab3-a68f- 7cae1fba51a8-iot-dispatcher	2 weeks, 6 d
 1350391e-621f-4470-b488- 0b03276497be-iot-dispatcher	Turtle Creek Demo Kit	3 weeks, 5 d

2. Identify thing on system (Thing Key)
 - a. Open terminal.
 - b. `$ cat/usr/share/dispatcher-agent/device_id`

Uploading the Telit deployment Bundle in the Telit Portal

Prerequisites: To perform this step EII bundle need to be generated as mentioned in the **Generating EII Bundle for Deployment** section. The EII bundle need to be in the same system as you access the Telit portal.

1. Go to the **Developer Page** and select **Files** general menu on the left side panel.
2. On the **Files** page, click **New File**.
3. In the **Add File** dialog, click **Attach File** and attach the generated TC-Telit Bundle **eii_bundle.tar.gz**.
4. Select the **Public** checkbox.
5. Click **Upload**.



6. The new file will be listed in the **Files** section. Copy the **Url** of the uploaded file for the AOTA method.



Performing AOTA Updates Through Telit

1. Go to the [Things](#) page.
2. Click the eye icon to select the provisioned device.
3. Select **Methods**.
4. Select **Trigger AOTA** as follows:



5. Fill up the trigger AOTA form accordingly.

- Container tag should be **eii_bundle** if you generate the bundle using default value.
- Fetch should be the copied URL from the upload bundle in the uploading the Telit deployment bundle in the Telit* portal.
- The version should be the docker compose file version number used for the tc-telit bundle generation.
- Skip other sections and then, click **execute**.

Trigger AOTA

App

docker-compose

Command

down

Container Tag

eis_bundle

Fetch

https://api-dev.devicewise.com:443/file/5e6af83014c97877fcdf8547/eis_bur

Signature

Version

19.03.5

1. Click the **event** tab to check on the AOTA status. If the AOTA is executed successfully, a success message is shown as follows:

Overview

Details

Attributes

Events

Files

Methods

Date	Level	Message
2020-03-13 03:12:46 +0000	Information	{"status": 200, "message": "COMMAND SUCCESS
2020-03-13 03:12:34 +0000	Information	Command: install_check passed. Message: Install
2020-03-13 03:12:33 +0000	Information	AOTA Triggered

Multi-node Deployment Without Using Device Manageability

This step deploys EII on multi-node by using the native method without any installation of Device Manageability and Cloud service. To continue on this section, it is required to complete the steps mentioned in the **Multi-node deployment** section and **Generating EII Bundle for Deployment** section.

NOTE Ensure to copy and untar the above bundle on a secure location having root only access as it contains secrets.

```
sudo tar -xvf <eii_bundle.gz generated>
cd build
docker-compose up
```

After completing this step, you can see that the container generated on the EII bundle is running.

Get Started Guide

This step-by-step guide takes you through installing the Edge Insights for Industrial on Linux and introduces you to the Edge Software command line interface from which you manage the Intel® Developer Catalog packages. After you complete this guide you will be ready to use a tutorial.

Refer to the [Requirements](#) section before you get started with installation.

To use these instructions, you must download the [Edge Insights for Industrial package](#). The download file name is `edge_insights_industrial.zip`.

NOTE
Important

Save the email message you get when you download the package. **Keep this message safe!** This message includes a product key that is required to complete the installation. If you do not get the email message, use the [Support Forum](#).

Edge Insights for Industrial is delivered as compressed .zip file that is compatible with the operating system you selected during the download. The .zip contains a binary executable file, a manifest file that lists the modules that will be installed, and a readme file.

See [Troubleshooting](#) if you run into problems installing the software.

After installation, follow the [Introduction to the Edge Software CLI](#) . This introduces you to the Edge Software command line interface from which you manage the Intel® Developer Catalog packages.

Requirements

In addition to the [Edge Insights for Industrial package](#), you must have the following:

Target System

- One of the following processors:
 - 6th, 7th, or 8th generation Intel® Core™ processor.
 - 6th, 7th, or 8th generation Intel® Xeon® processor.
 - Intel® Pentium® processor N4200/5, N3350/5, N3450/5 with Intel® HD Graphics.
- At least 16 GB RAM for video ingestion and analytics. (At least 2 GB RAM for time-series ingestion and analytics.)
- At least 64 GB hard drive.
- An Internet connection.
- Ubuntu* 18.04.3 LTS

Knowledge/Experience

You are familiar with executing Linux* commands.

Install Edge Insights for Industrial

During the installation, you will be prompted to enter your product key. This key is in the email you received from the Intel® Registration Center. Contact [Support Forum](#) if you do not have this email message.

The steps below explain how to:

- Prepare your target system.
- Copy the package.
- Complete the installation steps.

NOTE Be aware that screenshots may show a package version number that is different from the current release. See the [Release Notes](#) for information on the current release.

Step 1: Prepare the Target System

Make sure your target system has a fresh installation of Ubuntu* Linux that corresponds to the version of Edge Insights for Industrial that you downloaded. If you need help installing Ubuntu* Linux, follow these steps:

1. Download [Ubuntu* Linux* Desktop ISO file](#) to your developer workstation.
2. Use an imaging application to create a bootable flash drive.
3. Power off your target system, insert the USB drive, and power on the system.

If the target system doesn't boot from the USB drive, change the boot priority in the system BIOS.

4. Follow the prompts to install Ubuntu* Linux with the default configurations. For detailed instructions, see the [Ubuntu guide](#).
5. Power down your target system and remove the USB drive.
6. Power up the target system. You will see Ubuntu* Linux is successfully installed.
7. Log on as the root user:

```
su root
```

Step 2: Download and Copy the Edge Insights for Industrial .zip File to the Target System

In this step you download the package and copy Edge Insights for Industrial to your target system.

1. Download the [Edge Insights for Industrial package](#).
2. Copy `edge_insights_industrial.zip` from the developer workstation to the Home directory on your target system. You can use a USB flash drive to copy the file. The icon looks like this:

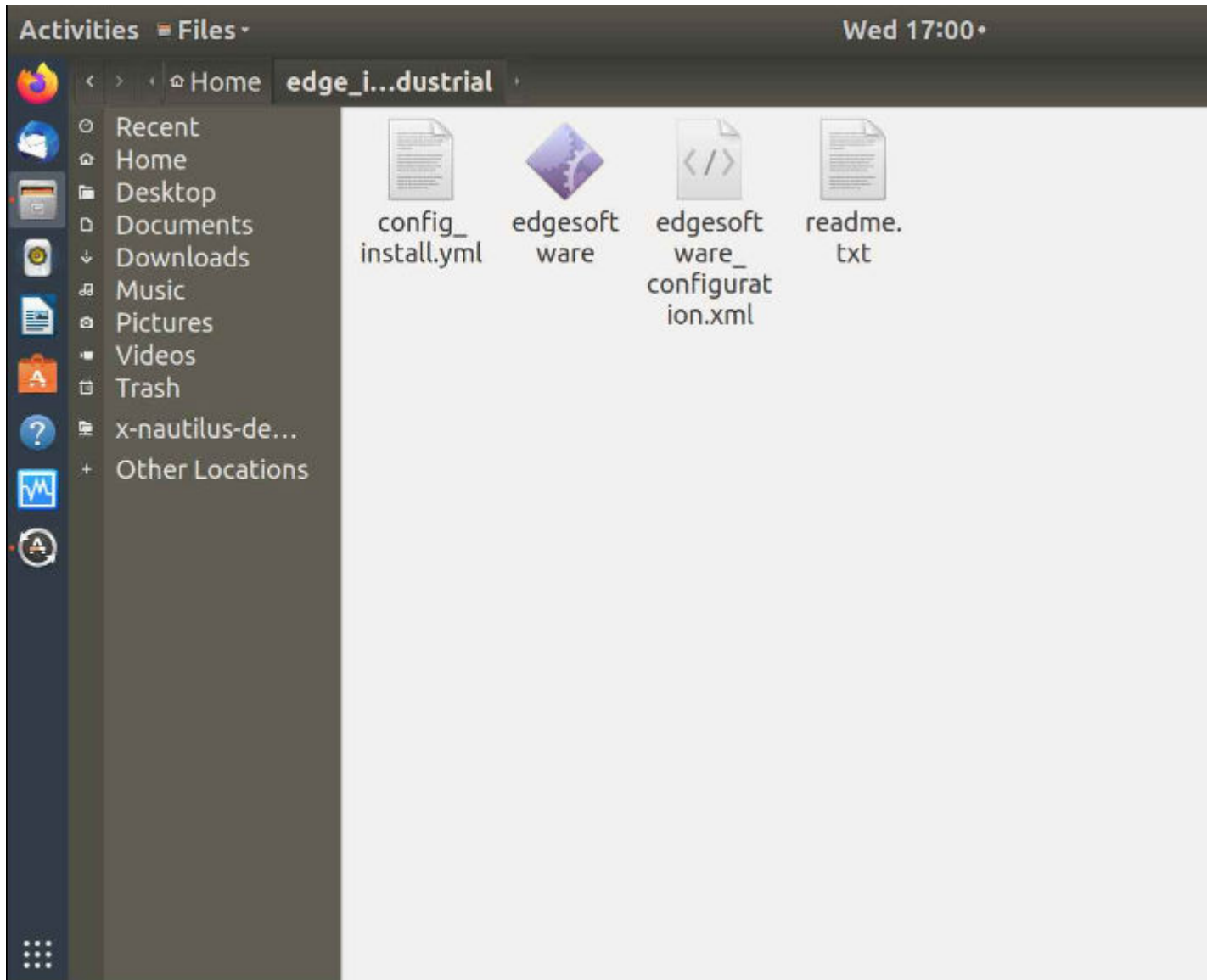
Step 3: Extract the Edge Insights for Industrial Software

In this step you extract `edge_insights_industrial.zip`. You need to be on the target system to complete these steps.

1. Make sure you have a working Internet connection.
2. Open a new terminal.
3. Extract the package:

```
unzip edge_insights_industrial.zip
```

NOTE If you download the file more than once, each download beyond the first is appended by a number.



4. (Optional) Use the link in the `readme` file to open this Get Started Guide on the target system for an easy way to copy and paste commands.

Step 4 (Optional): Configure your Installation

You can use the `config_install.yml` file to configure the installation behavior of the components in your target system.

Step 5: Install the Edge Insights for Industrial Software

NOTE If you are running behind a proxy server, please be sure that the proxy settings are configured correctly. The edgesoftware tool uses these proxy settings to download and install the modules.

You will now run a script that will download components and install Edge Insights for Industrial on your target system.

The software installation can take 1 to 3 hours depending on the package you chose. The completion time depends on your target system and Internet connection.

1. Run these commands:

```
cd edge_insights_industrial/  
chmod 775 edgesoftware  
./edgesoftware install
```

NOTE If you encounter any Docker* pull-related issues during the installation process, refer to the **Troubleshooting <install-edge-insights-for-industrial>** section at the end of this document.

2. Type the product key at the prompt:

```

intel@edgesoftware: ~/edge_insights_industrial
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial$ ./edgesoftware install
Please enter the Product Key. The Product Key is contained in the email you received from Intel confi
[REDACTED]
Starting the setup...
ESB CLI version: 2020.3
Target OS: Ubuntu
-----SOFTWARE INFO-----
Operating System: Ubuntu 18.04.4 LTS
Kernel: 5.4.0-42-generic
-----HARDWARE INFO-----
Hardware Architecture: x86_64
Processor: Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
Memory size: 15G
Total disk space (GB): 468
Free disk space (GB): 432
-----HARDWARE ACCELERATOR-----
No hardware accelerator found
Python version: 3.6.9
Checking Internet connection
Connected to the Internet
Checking for prerequisites
All dependencies met
Validating product key
Successfully validated Product Key
Starting installation
Downloading modules...
Downloading component esb_common
ZIP file for module 5e8c4742e02f17002a2a6976 already exists. Validating it...
Module validation passed for 5e8c4742e02f17002a2a6976
Skipping download...
Downloading component Docker_Community_Edition_CE
ZIP file for module 5f21392e9e63c9002a6fd88d already exists. Validating it...
Module validation passed for 5f21392e9e63c9002a6fd88d
Skipping download...
Downloading component Docker_Compose
ZIP file for module 5f213aae9e63c9002a6fd88e already exists. Validating it...
Module validation passed for 5f213aae9e63c9002a6fd88e
Skipping download...
Downloading component IEdgeInsights
ZIP file for module 5ef1c13e1c1b39002a68448b already exists. Validating it...
Module validation passed for 5ef1c13e1c1b39002a68448b
Skipping download...
Downloading component IEdgeInsights/Samples
ZIP file for module 5ef2e867fa5828002aa646da already exists. Validating it...

```

NOTE See the Edge Insights for Industrial based Reference Implementation (RI) tutorial requirements in the table below.

1. No	RI Tutorial	Applicable Use Case	Requirement
1	Defect Detection Demo	Video Analytics or Video Analytics and Time series	Download the installer package through the Download Recommended Configuration option by choosing one of the use cases (see Applicable Use Case column) in the Edge Insights for Industrial package .
2	Weld Porosity Detection Demo	Video Analytics or Video Analytics and Time series	Download the installer package through the Customize Download option by choosing one of the use cases (see Applicable Use Case column) in the Edge Insights for Industrial package and additionally selecting the Weld Porosity Detection RI.
3	Industrial Text Line Recognition	Video Analytics or Video Analytics and Time series	Download the installer package through the Customize Download option by choosing one of the use cases (see Applicable Use Case column) in the Edge Insights for Industrial package and additionally selecting the Industrial Text Line Recognition RI.
4	Industrial Surface Defect Detection	Video Analytics or Video Analytics and Time series	Download the installer package through the Customize Download option by choosing one of the use cases (see Applicable Use Case column) in the Edge Insights for Industrial package and additionally selecting the Industrial Surface Defect Detection RI.

1. No	RI Tutorial	Applicable Use Case	Requirement
5	Textile Defect Classifier	Video Analytics or Video Analytics and Time series	Download the installer package through the Customize Download option by choosing one of the use cases (see Applicable Use Case column) in the Edge Insights for Industrial package and additionally selecting the Textile Defect Classifier RI.

Based on components selected and system configuration, you might be prompted for additional actions if you have not configured the `config_install.yml` file correctly.

When the installation is complete, you see the message `Installation of package complete` and the installation status for each module.


```

Activities  Terminal  Thu 15:34
intel@edgesoftware: ~/edge_insights_industrial

File Edit View Search Terminal Help
Creating edgeinsightssoftware_ia_azure_simple_subscriber_1 ... done
Creating ia_grafana ... done
Creating ia_video_ingestion ... done
Verifying Docker images
SUCCESS : Verified openedgeinsights/ia_azure_bridge
SUCCESS : Verified openedgeinsights/ia_azure_simple_subscriber
SUCCESS : Verified openedgeinsights/ia_factoryctrl_app
SUCCESS : Verified openedgeinsights/ia_grafana
SUCCESS : Verified openedgeinsights/ia_imagestore
SUCCESS : Verified openedgeinsights/ia_influxdbconnector
SUCCESS : Verified ia_kapacitor
SUCCESS : Verified openedgeinsights/ia_opcua_export
SUCCESS : Verified openedgeinsights/ia_rest_export
SUCCESS : Verified ia_tls_remoteagent
SUCCESS : Verified openedgeinsights/ia_telegraf
SUCCESS : Verified openedgeinsights/ia_video_analytics
SUCCESS : Verified openedgeinsights/ia_video_ingestion
SUCCESS : Verified openedgeinsights/ia_visualizer
SUCCESS : Verified openedgeinsights/ia_web_visualizer
SUCCESS : Verified openedgeinsights/ia_zmq_broker
Building Insights Images [.....]
Building Insights Images [.....]
100%
Edge Insights Visualizer [
Edge Insights Visualizer [.....]
Edge Insights Visualizer [.....]
100%
Successfully installed eii_installer took 5 minutes 39.83 seconds
Installation of package complete
***Recommended to reboot system after installation***

+-----+-----+-----+
| Id | Module | Status |
+-----+-----+-----+
| 5f21392e9e63c9002a6fd88d | Docker Community Edition CE | SUCCESS |
| 5f213aae9e63c9002a6fd88e | Docker Compose | SUCCESS |
| 60c727ad4b40e5002ad9b795 | eii installer | SUCCESS |
+-----+-----+-----+

intel@edgesoftware:~/edge_insights_industrial$

```

NOTE For Time series or Video Analytics and Time series use cases, the demo for time series does not start automatically, unlike the video use case. Please follow the step below to see the time series demonstration:

- Launch Grafana to see the time series visualization. For details, refer to [Open Edge Insights Grafana](#) information.
-

To confirm your installation was successful and for a quick look at how it works, use the [Introduction to the Command Line Interface](#).

Steps for Manual Installation of Manageability Module

NOTE Manageability is downloaded by default, but not auto-installed.

- Go to `edge_insights_industrial/Edge_Insights_for_Industrial_<version>/manageability/` where `<version>` indicates the downloaded version of Edge Insights for Industrial.
- Refer to the steps in `README.md` for manual installation of manageability component.

Steps for Manual Installation of Training and Learning Suite Module and TLSRemoteAgent Configuration

NOTE Training and Learning Suite is available via custom download, but not auto-installed.

- Go to `edge_insights_industrial/Edge_Insights_for_Industrial_<version>/training-and-learning-suite` where `<version>` indicates the downloaded version of Edge Insights for Industrial.
- Run the command: `chmod 755 -R .`
- Refer to the steps at `README.md` for TLS server installation.
- Refer to `edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights/TLSRemoteAgent/README.md` for configuration of TLSRemoteAgent service.

Troubleshooting

- Make sure you have an active internet connection during the full installation. If you lose Internet connectivity at any time, the installation might fail.
- Make sure you are using a fresh Ubuntu* Linux installation. Earlier software, especially Docker* and Docker Compose*, can cause issues.
- If the `eii installer` component fails to install after running the `./edgesoftware install` command, check the running containers list using `docker ps` command and check the logs of the failing containers to find out why they are failing. Mostly services like InfluxDBConnector, Grafana, Imagestore, etc., fail if the port is already used by other process on the host m/c as EII uses `docker host` network. Please make sure to free up that port and re-run the `./edgesoftware install` command or follow the steps below:
 - Go to `edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights/build` where `<version>` indicates the downloaded version of Edge Insights for Industrial.
 - Run `sudo sg docker -c docker-compose up -d` command to relaunch services.

Restart Mode

The following services may be in **restart mode** after successful installation. Refer to the respective README.md files to troubleshoot, where `<version>` indicates the downloaded version of Edge Insights for Industrial:

- **Visualizer:**`edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights/Visualizer/README.md`
- **FactoryControlApp:**`edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights/FactoryControlApp/README.md`
- **RestDataExport:**`edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights/RestDataExport/README.md`
- **TLSRemoteAgent:**`edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights/TLSRemoteAgent/README.md`

Docker Image Pull Issue

This issue can be because of the latest pull rate limitations introduced by Docker hub.

- Please check this [Docker site article](#) to help determine the exact pull limit that's applicable on the system where you are trying to pull the publicly available Docker hub images, such as Docker, Python, etc.

NOTE This limit is only applicable for the 6-hour window.

- If you see this issue with an anonymous user (pull limit of 100), i.e., without Docker login, you can [create an account](#) and try to do the build after running the command: `docker login -u <username> -p <password>`. The alternative is to use a [paid subscription](#).

If you're unable to resolve your issues, go to the [Support Forum](#).

Introduction to the Edge Software CLI

edgesoftware is a command line interface (CLI) that helps you manage packages on the Intel® Developer Catalog.

This guide describes the CLI commands and their usage. In this guide you will:

- Try out commands and get familiar with the CLI and the package you installed.
- Learn to update modules.
- Learn to install custom components.
- Learn to export the package you installed, including custom modules, so you can install it on other edge nodes.

Get Started with the edgesoftware CLI

Use the information in this section to try out the edgesoftware CLI commands.

NOTE Be aware that screenshots may show a package version number that is different from the current release. See the [Release Notes](#) for information on the current release.

To begin:

1. Open a terminal window.
2. Go to the `edge_insights_industrial/` directory.
3. Try out the following commands.

Get Help or List the Available Commands

- Command:

```
./edgesoftware --help
```

- Response:

```
Usage: edgesoftware [OPTIONS] COMMAND [ARGS]...
A CLI wrapper for management of Intel® Edge Software Hub packages

Options:
  -v, --version    Show the version number and exit.
  --help           Show this message and exit.

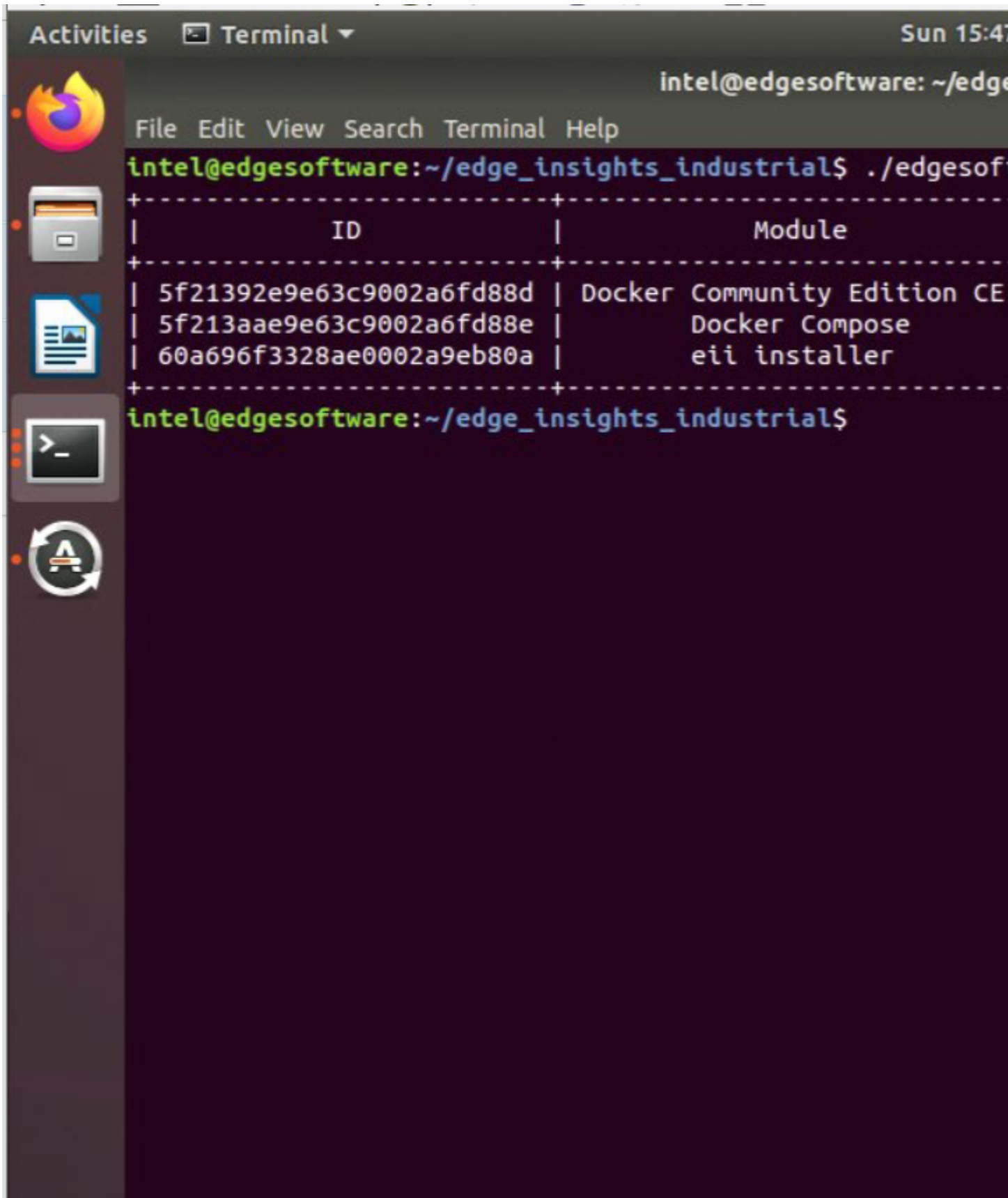
Commands:
  download  Download modules of a package.
  export    Exports the modules installed as a part of a package.
  install   Install modules of a package.
  list      List the modules of a package.
  log       Show log of CLI events.
  pull      Pull Docker image.
  uninstall Uninstall the modules of a package.
  update    Update the modules of a package.
  upgrade   Upgrade a package.
```

Download Package Modules

- Command:

```
./edgesoftware download
```

- Response: Downloads and unzips the modules of the package.



The image shows a terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Search", "Terminal", and "Help". The prompt is "intel@edgesoftware: ~/edge". The command executed is `./edgesof`. The output is a table with two columns: "ID" and "Module".

ID	Module
5f21392e9e63c9002a6fd88d	Docker Community Edition CE
5f213aae9e63c9002a6fd88e	Docker Compose
60a696f3328ae0002a9eb80a	eii installer

The prompt is now "intel@edgesoftware: ~/edge_insights_industrial\$".

View the Software Version

- Command:

```
./edgesoftware --version
```

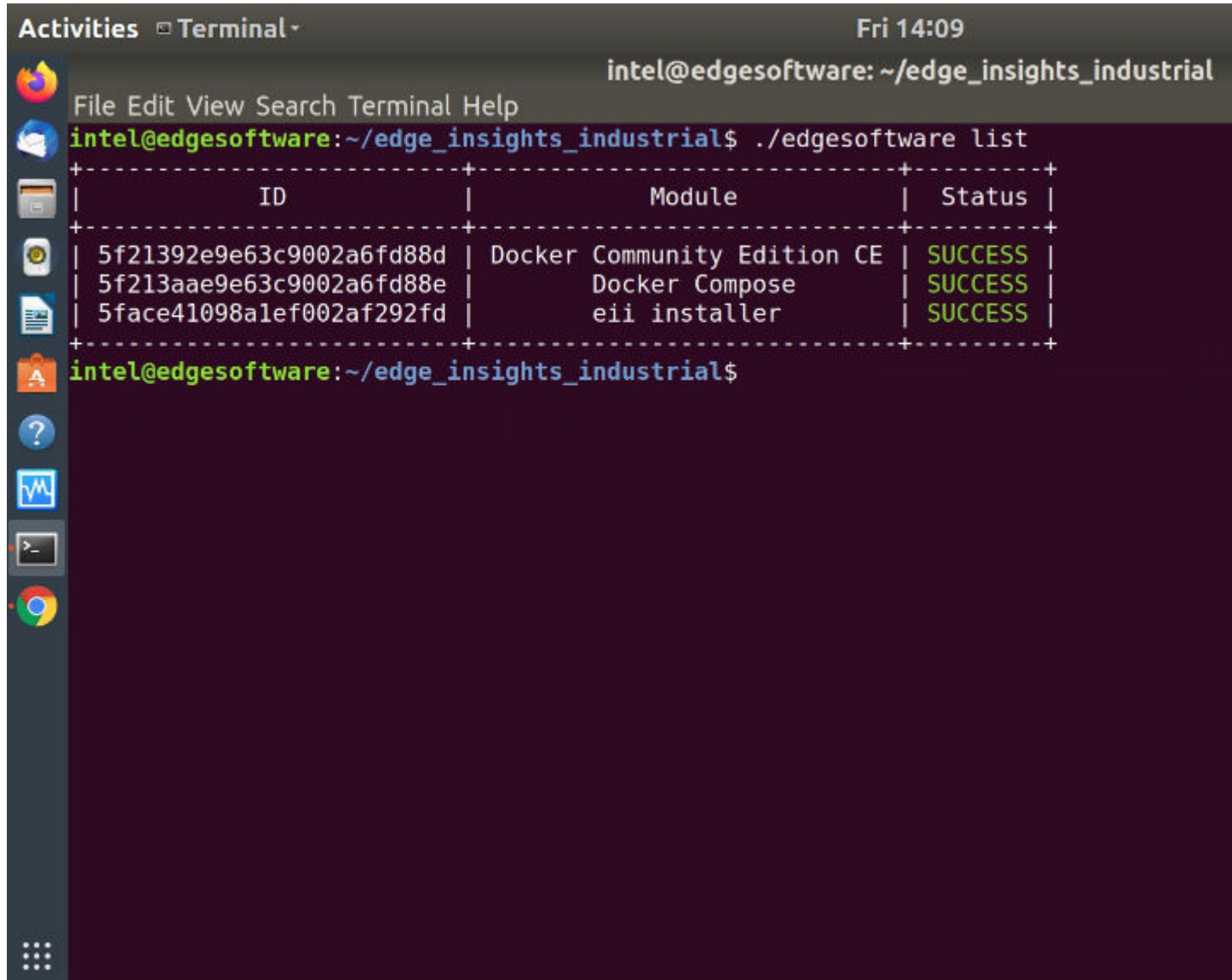
- Response: The edgesoftware version, build date, and target OS.

List the Package Modules

- Command:

```
./edgesoftware list
```

- Response: The modules installed and status.



```

Activities  Terminal  Fri 14:09
intel@edgesoftware: ~/edge_insights_industrial
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial$ ./edgesoftware list
+-----+-----+-----+
| ID | Module | Status |
+-----+-----+-----+
| 5f21392e9e63c9002a6fd88d | Docker Community Edition CE | SUCCESS |
| 5f213aae9e63c9002a6fd88e | Docker Compose | SUCCESS |
| 5face41098a1ef002af292fd | eii installer | SUCCESS |
+-----+-----+-----+
intel@edgesoftware:~/edge_insights_industrial$
  
```

List Modules Available for Download

- Command:

```
./edgesoftware list --default
```

- Response: All modules available for download for that package version, modules ID and version.

Display the CLI Event Log

- Command:

```
./edgesoftware log
```

- Response: CLI event log information, such as:

- target system information (hardware and software)
- system health
- installation status
- modules you can install

```

intel@edgesoftware: ~/edge_insights_industrial
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial$ ./edgesoftware log
=====Start of installer log=====
Tue Aug 18 11:02:29 IST 2020 - INFO - ESB CLI version: 2020.2.08.17
Tue Aug 18 11:02:30 IST 2020 - INFO - -----SOFTWARE INFO-----
Tue Aug 18 11:02:30 IST 2020 - INFO - Operating System: Ubuntu 18.04.4 LTS
Tue Aug 18 11:02:30 IST 2020 - INFO - Kernel: 5.3.0-28-generic

Tue Aug 18 11:02:30 IST 2020 - INFO - -----HARDWARE INFO-----
Tue Aug 18 11:02:30 IST 2020 - INFO - Hardware Architecture: x86_64
Tue Aug 18 11:02:30 IST 2020 - INFO - Processor: Intel(R) Core(TM) i7-6700K CPU @ 4.00GHz
Tue Aug 18 11:02:30 IST 2020 - INFO - Memory size: 15G
Tue Aug 18 11:02:30 IST 2020 - INFO - Total disk space (GB): 468
Tue Aug 18 11:02:30 IST 2020 - INFO - Free disk space (GB): 431

Tue Aug 18 11:02:30 IST 2020 - INFO - -----HARDWARE ACCELERATOR-----
Tue Aug 18 11:02:30 IST 2020 - INFO - No hardware accelerator found
Tue Aug 18 11:02:30 IST 2020 - INFO - Python version: 3.6.9
Tue Aug 18 11:02:30 IST 2020 - INFO - Checking Internet connection
Tue Aug 18 11:02:31 IST 2020 - INFO - Connected to the Internet
Tue Aug 18 11:02:31 IST 2020 - INFO - Checking for prerequisites
Tue Aug 18 11:02:34 IST 2020 - INFO - Installing prerequisites. This may take some time...
Tue Aug 18 11:04:39 IST 2020 - INFO - -----Successfully installed prerequisites-----
Tue Aug 18 11:04:39 IST 2020 - INFO - All dependencies met
Tue Aug 18 11:04:39 IST 2020 - INFO - Validating product key
Tue Aug 18 11:04:40 IST 2020 - INFO - Successfully validated Product Key
Tue Aug 18 11:04:40 IST 2020 - INFO - Starting installation
Tue Aug 18 11:04:40 IST 2020 - INFO - Downloading modules...
Tue Aug 18 11:04:40 IST 2020 - INFO - Downloading component esb_common
Tue Aug 18 11:04:40 IST 2020 - INFO - Sending request to download module 5e8c4742e02f17002a2a6976
Tue Aug 18 11:04:42 IST 2020 - INFO - Sending request to validate module 5e8c4742e02f17002a2a6976 with hash value
f25be6a391a4322e56ce2216fe4
Tue Aug 18 11:04:42 IST 2020 - INFO - Module validation passed for 5e8c4742e02f17002a2a6976
Tue Aug 18 11:04:42 IST 2020 - INFO - Successfully downloaded module esb_common
Tue Aug 18 11:04:42 IST 2020 - INFO - Downloading component Docker_Community_Edition_CE
Tue Aug 18 11:04:42 IST 2020 - INFO - Sending request to download module 5f21392e9e63c9002a6fd88d
Tue Aug 18 11:04:44 IST 2020 - INFO - Sending request to validate module 5f21392e9e63c9002a6fd88d with hash value
cb30328bcec83af377fe239d5f3
Tue Aug 18 11:04:45 IST 2020 - INFO - Module validation passed for 5f21392e9e63c9002a6fd88d
Tue Aug 18 11:04:45 IST 2020 - INFO - Successfully downloaded module Docker_Community_Edition_CE
Tue Aug 18 11:04:45 IST 2020 - INFO - Downloading component Docker_Compose
Tue Aug 18 11:04:45 IST 2020 - INFO - Sending request to download module 5f21392e9e63c9002a6fd88d

```

See the Installation Event Log for a Module

- Command:

```
./edgesoftware log <MODULE_ID>
```

You can specify multiple <MODULE_ID> arguments by listing them with a space between each.

NOTE To find the module ID, use

```
./edgesoftware list
```

- Response: The installation log for the module.

Install Package Modules

This edgesoftware command installs package modules on the target system. To do so, the command looks at `edgesoftware_configuration.xml` that was downloaded from the Intel® Developer Catalog when you installed the Edge Insights for Industrial software. This file contains information about the modules to install.

During the installation, you will be prompted to enter your product key. The product key is in the email message you received from Intel confirming your Edge Insights for Industrial download.

NOTE

Important

Do not manually edit `edgesoftware_configuration.xml`.

1. Open a terminal window.
2. Go to the `edge_insights_industrial/` directory.
3. Run the install command:

```
./edgesoftware install
```

Update the Package Modules

NOTE On a fresh Linux installation, you might need to use the `install` command at least once before performing an update. `install` makes sure all dependencies and packages are installed on the target system.

```
./edgesoftware install
```

When you are ready to perform the update, use:

```
./edgesoftware update <MODULE_ID>
```

During the installation, you will be prompted to enter your product key. The product key is in the email message you received from Intel confirming your Edge Insights for Industrial download.

NOTE To find the module ID, use

```
./edgesoftware list --default
```

Export the Package for Installation

The edgesoftware CLI lets you package the installed modules, customer applications, and dependencies as part of a package. The export is provided in a `.zip` file that includes installation scripts, XML files, and an edgesoftware Python* executable.

Command:

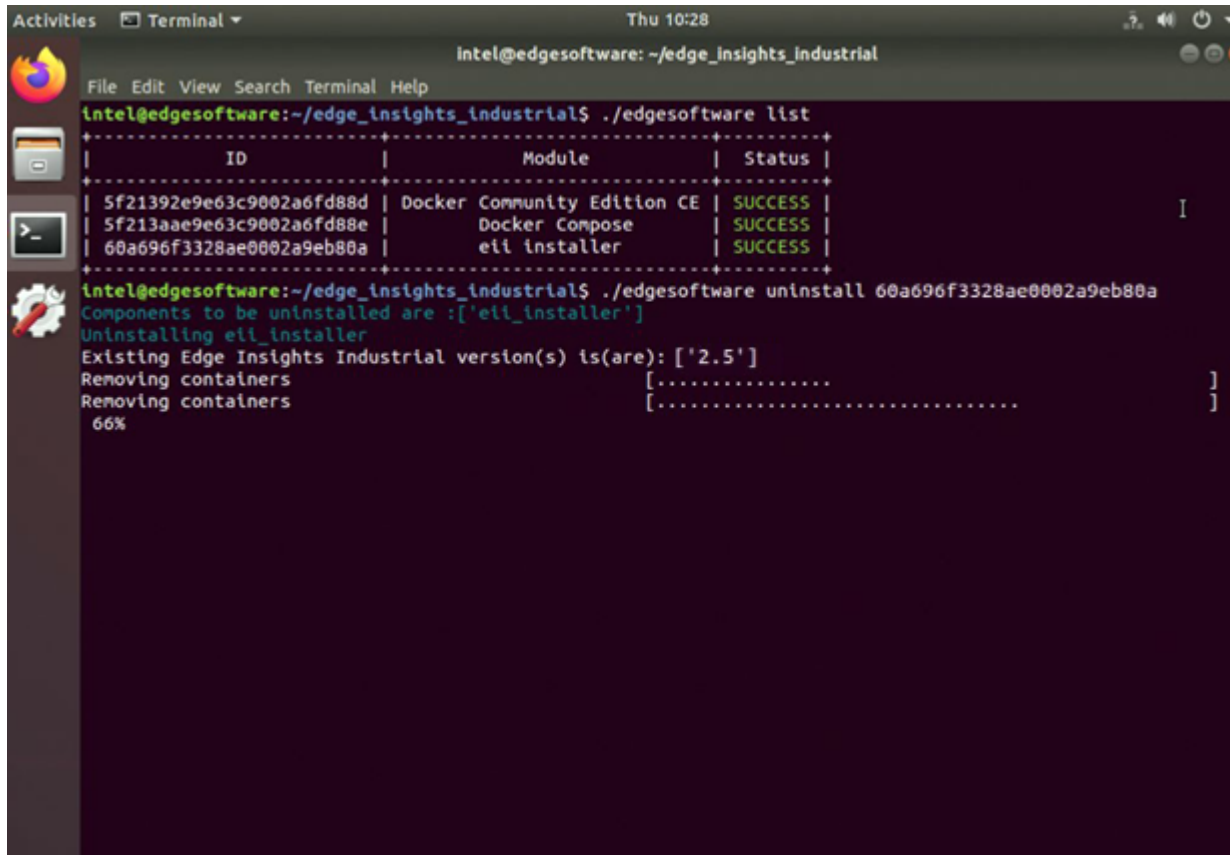
```
./edgesoftware export
```

Uninstall the Packages

The edgesoftware CLI lets you uninstall the complete package or individual components from the package.

To uninstall individual package, run the following command:

```
./edgesoftware uninstall  
<Module-ID>
```



The screenshot shows a terminal window titled "Intel@edgesoftware: ~/edge_insights_industrial". The user runs the command `./edgesoftware list`, which displays a table of installed modules. The table has three columns: ID, Module, and Status. The modules listed are Docker Community Edition CE, Docker Compose, and eii_installer, all with a status of SUCCESS. The user then runs the command `./edgesoftware uninstall 60a696f3328ae0002a9eb80a`. The output shows that the components to be uninstalled are ['eii_installer']. It then displays the existing Edge Insights Industrial version(s) as ['2.5']. The process of removing containers is shown with a progress bar at 66%.

```
Intel@edgesoftware:~/edge_insights_industrial$ ./edgesoftware list
+-----+-----+-----+
| ID | Module | Status |
+-----+-----+-----+
| 5f21392e9e63c9002a6fd88d | Docker Community Edition CE | SUCCESS |
| 5f213aae9e63c9002a6fd88e | Docker Compose | SUCCESS |
| 60a696f3328ae0002a9eb80a | eii_installer | SUCCESS |
+-----+-----+-----+

Intel@edgesoftware:~/edge_insights_industrial$ ./edgesoftware uninstall 60a696f3328ae0002a9eb80a
Components to be uninstalled are :['eii_installer']
Uninstalling eii_installer
Existing Edge Insights Industrial version(s) is(are): ['2.5']
Removing containers [.....]
Removing containers [.....]
66%
```

To uninstall all the packages, run the following command:

```
./edgesoftware uninstall -a
```



```

Activities Terminal
Fri 15:33
intel@edgesoftware: ~/edge_insights_industrial

File Edit View Search Terminal Help
Removing image : ['6ad5ca31daf9', '2.5', 'ia_connon']
Successfully removed : 6ad5ca31daf9
Removing image : ['efb21d423ea1', '2.5', 'ia_etibase']
Successfully removed : efb21d423ea1
Removing image : ['52a693e7cdef', '2.5', 'ia_etcd_provision']
Successfully removed : 52a693e7cdef
Removing image : ['cb4f2ecdafa32', '2.5', 'ia_etcd']
Successfully removed : cb4f2ecdafa32
Removing Images [.....] 100%
Removing Volumes [.....] 100%
Removing Networks [.....] 100%
Successfully uninstalled eii_installer took 1 minutes 34.81 seconds
Uninstalling Docker Compose [.....] 100%
Uninstalling Docker Compose [.....] 100%
Successfully uninstalled Docker_Compose took 11.75 seconds
Uninstalling Docker Community Edition CE [.....] 100%
Uninstalling docker [.....] 100%
Successfully uninstalled Docker_Community_Edition_CE took 56.18 seconds
Uninstall Finished

+-----+-----+-----+
| Id | Module | Status |
+-----+-----+-----+
| 60a696f3328ae0002a9eb80a | eii installer | SUCCESS |
| 5f213aae9e63c9002a6fd88e | Docker Compose | SUCCESS |
| 5f21392e9e63c9002a6fd88d | Docker Community Edition CE | SUCCESS |
+-----+-----+-----+

intel@edgesoftware:~/edge_insights_industrial$

```

Summary and Next Steps

By following this guide, you tried a few commands to familiarize yourself with the features of the edgesoftware CLI.

Go to [Tutorials](#) to start using Edge Insights for Industrial.

Tutorials

Follow the tutorials in this section to learn how to use and configure Edge Insights for Industrial for different use cases.

With step-by-step instructions covering real world usage scenarios, tutorials provide a learning path for developers to follow for mastering the usage of Edge Insights for Industrial.

Get started in your learning journey:

- [Defect Detection Demo](#)
- [Textile Defect Classifier](#)
- [Industrial Text Line Recognition](#)
- [Industrial Surface Defect Detection](#)
- [Weld Porosity Detection](#)

Defect Detection Demo

Prerequisite: The Defect Detection Demo tutorial requires video pipeline services, which are available in the **Video Analytics** or **Video Analytics and Time series** use cases installer package downloaded through the **Download** option for the [Edge Insights for Industrial package](#).

In this tutorial, you'll run the Defect Detection Demo to verify that Edge Insights for Industrial was installed successfully and to start getting familiar with its modules and structure. By following this tutorial, you will learn:

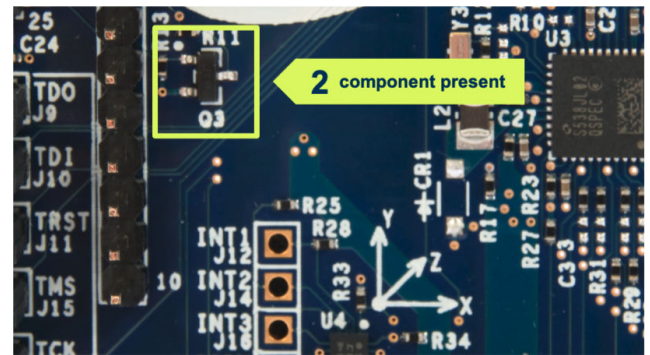
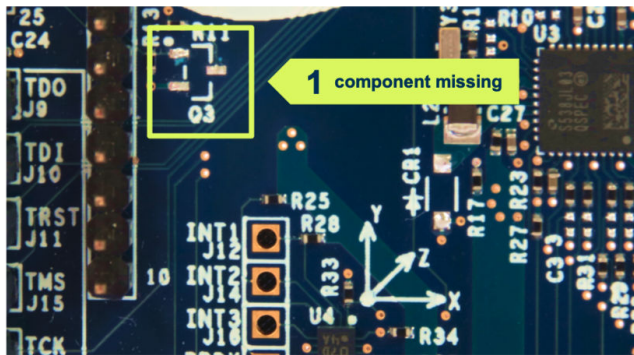
- How to start Edge Insights for Industrial
- How to visualize the results of the demo application
- How the application works at a high level

NOTE Be aware that screenshots may show a package version number that is different from the current release. See the [Release Notes](#) for information on the current release.

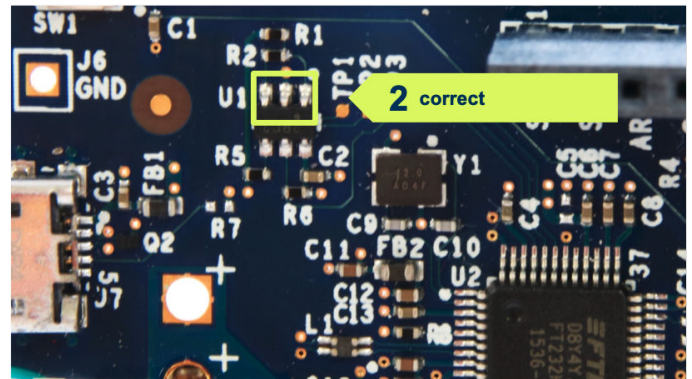
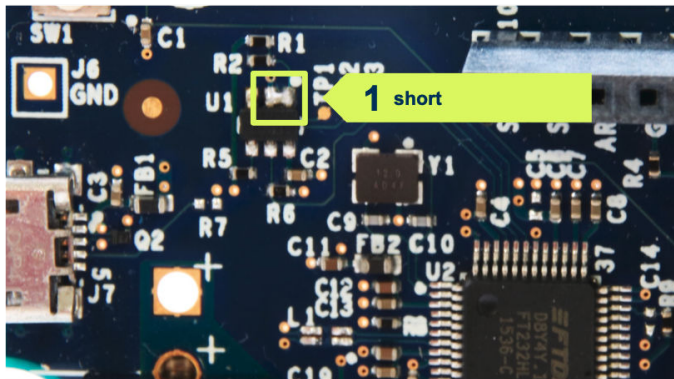
How it Works

The Defect Detection Demo performs a quality control inspection on a video of the printed circuit boards (PCBs). In this scenario, the PCBs are inspected for quality control and detects two types of defects:

- A missing component. In the following illustration, PCB #1, on the left, is missing a component. PCB #2, on the right, shows the component in place:



- A component short: In the following illustration, PCB #1, on the left, has two solder joints connected that should not be connected, resulting in a short. PCB #2, on the right, shows the correct solder:



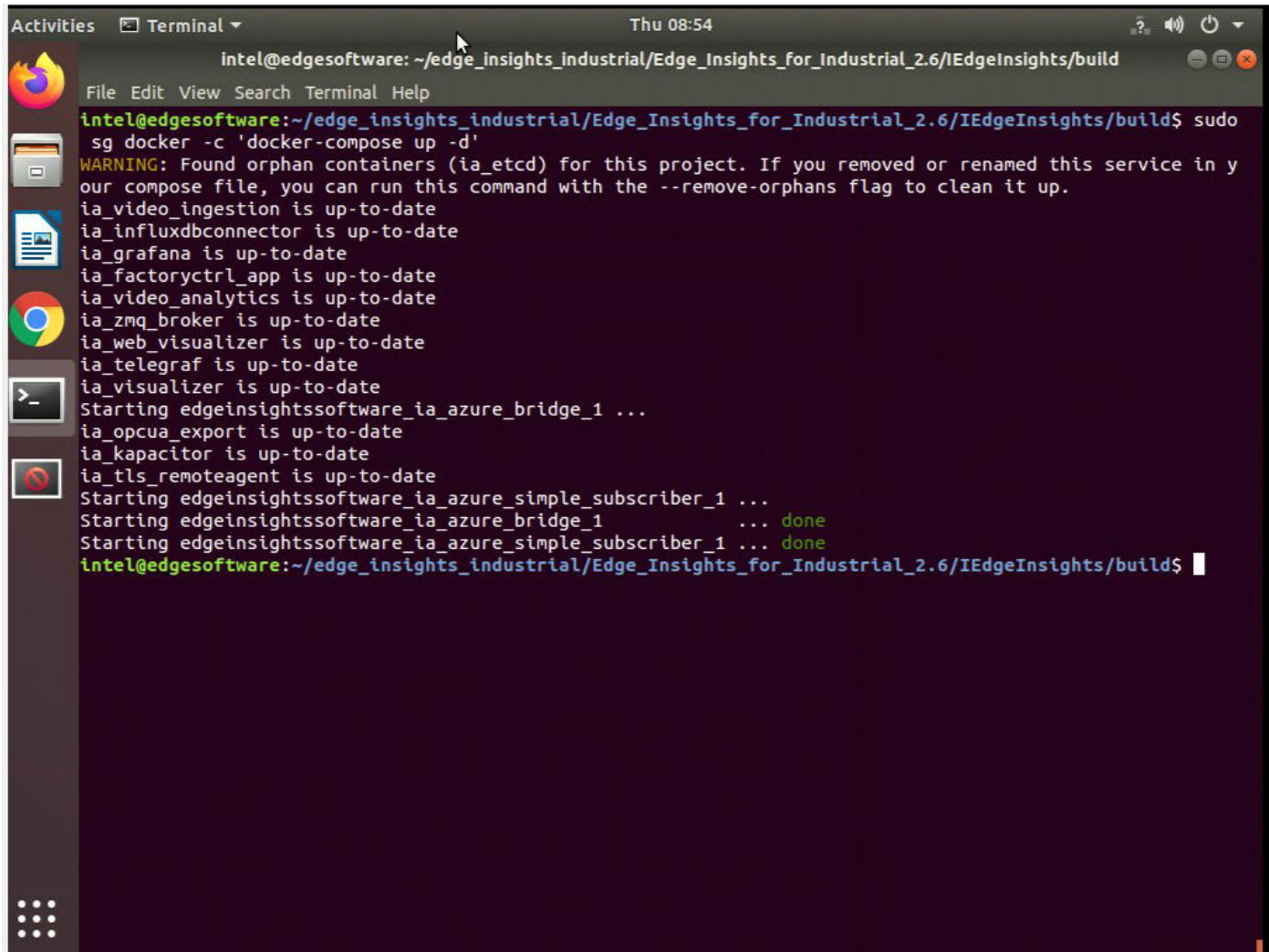
Step 1: Start the Edge Insights for Industrial Containers

1. Run the Edge Insights for Industrial:

```
xhost +
cd $HOME/edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights/build
sudo sg docker -c 'docker-compose up -d'
```

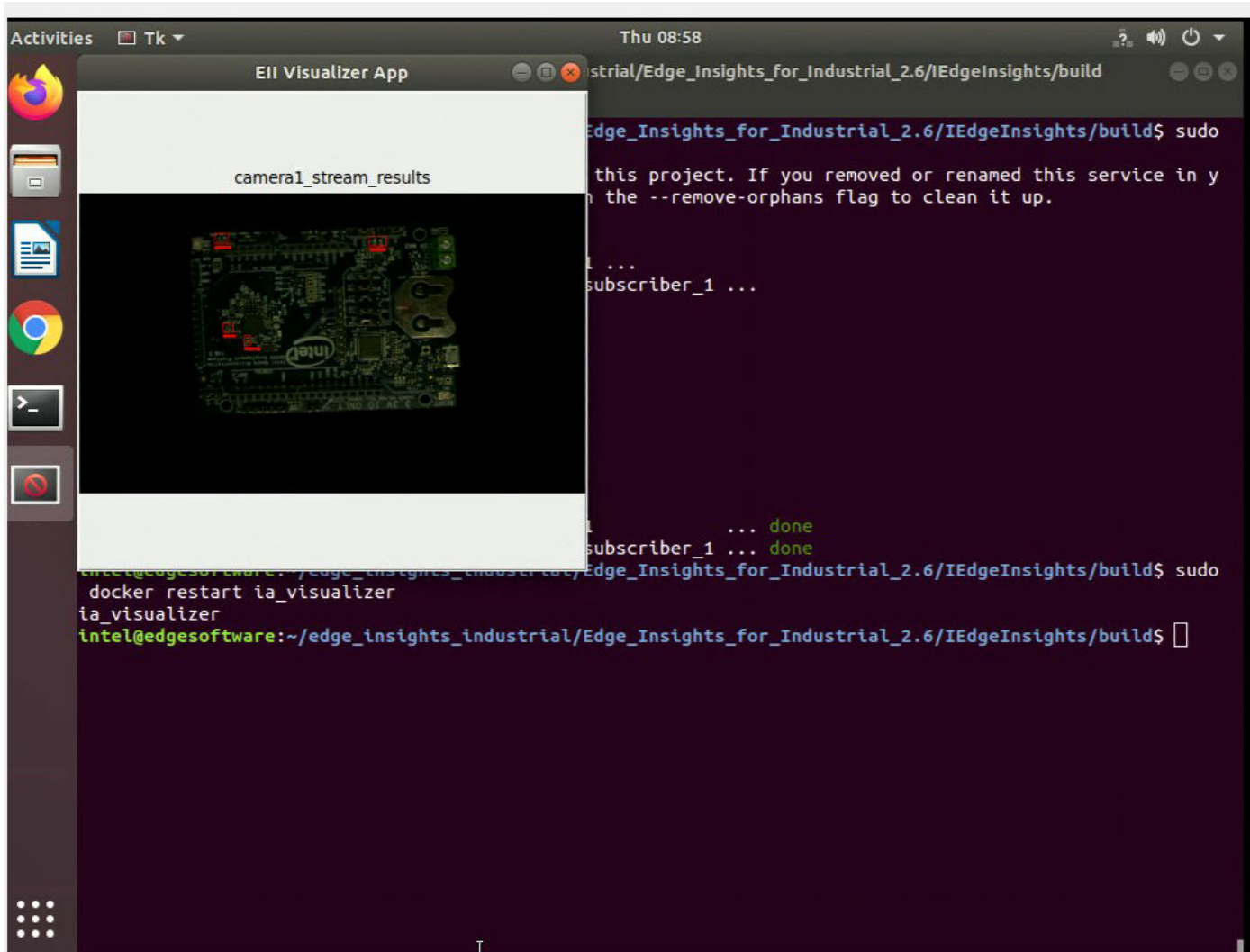
where <version> indicates the downloaded version of Edge Insights for Industrial.

Success is indicated by a screen similar to:

A terminal window titled 'Terminal' with a dark background. The window shows the command 'sudo sg docker -c 'docker-compose up -d'' being executed. The output lists various services as 'up-to-date' and shows the starting of 'edgeinsightssoftware_ia_azure_bridge_1' and 'edgeinsightssoftware_ia_azure_simple_subscriber_1' with 'done' status. The terminal window is part of a desktop environment with a sidebar on the left containing icons for Firefox, Files, Documents, Applications, and a Dash icon at the bottom. The top of the window shows the system clock as 'Thu 08:54' and window control buttons on the right.

```
intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$ sudo
sg docker -c 'docker-compose up -d'
WARNING: Found orphan containers (ia_etcd) for this project. If you removed or renamed this service in y
our compose file, you can run this command with the --remove-orphans flag to clean it up.
ia_video_ingestion is up-to-date
ia_influxdbconnector is up-to-date
ia_grafana is up-to-date
ia_factoryctrl_app is up-to-date
ia_video_analytics is up-to-date
ia_zmq_broker is up-to-date
ia_web_visualizer is up-to-date
ia_telegraf is up-to-date
ia_visualizer is up-to-date
Starting edgeinsightssoftware_ia_azure_bridge_1 ...
ia_opcua_export is up-to-date
ia_kapacitor is up-to-date
ia_tls_remoteagent is up-to-date
Starting edgeinsightssoftware_ia_azure_simple_subscriber_1 ...
Starting edgeinsightssoftware_ia_azure_bridge_1 ... done
Starting edgeinsightssoftware_ia_azure_simple_subscriber_1 ... done
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$
```

In addition, the visualizer displays an image that displays metadata and a separate window that shows the PCB image with the defects outlined by red boxes:



NOTE If the Visualizer UI does not show up and you notice the error `couldn't connect to display ":0"` after running the command `docker logs -f ia_visualizer`, check the value of the `DISPLAY` environment variable on the host machine. Use the command:

```
env | grep DISPLAY
```

Set the value for the `DISPLAY` environment variable in the `ia_visualizer` service in the `HOME/edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights/build/docker-compose.yml` file. Then bring up the `ia_visualizer` UI with the command:

```
docker-compose up ia_visualizer
```

For example,

```
$ env | grep DISPLAY
DISPLAY=:1
```

Set `=:1` as the `DISPLAY` environment value in the `ia_visualizer` service in the `docker-compose.yml` file.

2. Verify the containers are running:

```
docker ps
```

The result looks similar to:

```

intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$ sudo
docker ps
CONTAINER ID   IMAGE                                COMMAND                                            CREATED        ST
ATUS
                    NAMES                                PORTS
55cb9f319c5b   openedgeinsights/ia_grafana:2.6-RC2  "/.Grafana/run.sh"                             9 hours ago   Up
9 hours (healthy)                                0.0.0.0:3000->3000/tcp
                    ia_grafana
c0d973ab8e2e   ia_kapacitor:2.6-RC2                "/.classifier_startu..."                       9 hours ago   Up
9 hours (healthy)                                0.0.0.0:9092->9092/tcp
                    ia_kapacitor
73f530dac9fb   openedgeinsights/ia_factoryctrl_app:2.6-RC2  "python3 factoryctrl..."                     9 hours ago   Re
starting (255) 20 seconds ago
                    ia_factoryctrl_app
e870087ffbd    openedgeinsights/ia_influxdbconnector:2.6-RC2  "/.startup.sh"                                9 hours ago   Up
9 hours (healthy)                                0.0.0.0:8086->8086/tcp, 0.0.0.0:65030-65034->65030-65034/tcp, 0
.0.0.0:65145->65145/tcp
                    ia_influxdbconnector
d4c34f4c89a5   openedgeinsights/ia_telegraf:2.6-RC2         "python3 telegraf_st..."                     9 hours ago   Up
9 hours (healthy)                                0.0.0.0:65077->65077/tcp
                    ia_telegraf
ab02a28b5c78   openedgeinsights/ia_imagestore:2.6-RC2       "/.image-store"                                9 hours ago   Up
9 hours (healthy)                                0.0.0.0:5669->5669/tcp
                    ia_imagestore
91450b1f39f5   ia_tls_remoteagent:2.6-RC2              "/.tls_remote_agent_..."                     9 hours ago   Up
Less than a second (health: starting)
                    ia_tls_remoteagent
b9ba6a39f341   openedgeinsights/ia_opcua_export:2.6-RC2     "/.OpcuaExport/Opcua..."                     9 hours ago   Up
9 hours (healthy)                                0.0.0.0:65003->65003/tcp
                    ia_opcua_export
f9ba4209bc0a   openedgeinsights/ia_rest_export:2.6-RC2     "/.RestDataExport"                             9 hours ago   Up
9 hours (healthy)                                0.0.0.0:8087->8087/tcp
                    ia_rest_export
184987318fb3   openedgeinsights/ia_video_analytics:2.6-RC2  "/.VideoAnalytics/va..."                     9 hours ago   Up
9 hours (healthy)                                0.0.0.0:65013->65013/tcp
                    ia_video_analytics
e1a32fa60ecf   openedgeinsights/ia_zmq_broker:2.6-RC2       "/.zmq-broker"                                9 hours ago   Up
9 hours (healthy)                                0.0.0.0:60514-60515->60514-60515/tcp
                    ia_zmq_broker

```

The following table describes the contents of the screen.

Note that <version> indicates the downloaded version of Edge Insights for Industrial.

Image Column Content	Description
ia_video_ingestion:<version>	Ingests video frames from a video source, like a video file or camera, using the GStreamer* pipeline. Data, consisting of a frame and metadata, is published to the message bus.
ia_video_analytics:<version>	Use OpenVINO™ on the data to perform inference. The data are received from the video ingestion and new data is published to the message bus.
ia_visualizer:<version>	Use a Python-based visualizer to display the frame sent by video analytics.

Image Column Content	Description
ia_etcd:<version>	etcd* provides endpoint configurations to establish the message bus and configuration of Edge Insights for Industrial containers.
ia_etcd_ui:<version>	Web user interface for etcd* configurations.

3. Check the log files to verify the data pipeline in Edge Insights for Industrial is working correctly:

```
sudo docker logs -f ia_video_analytics
```

```

intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$ sudo
docker logs -f ia_video_analytics
[setupvars.sh] OpenVINO environment initialized
(root): extra property 'queue_size' found.
[Wed Jun 23 19:01:19 2021] INFO:validate_json:49: JSON schema validation passed !
[Wed Jun 23 19:01:19 2021] WARN:get_config_value:202: JSON does not contain key: zmq_recv_hwm
[Wed Jun 23 19:01:19 2021] WARN:get_config_value:202: JSON does not contain key: brokered
[Wed Jun 23 19:01:19 2021] WARN:get_config_value:202: JSON does not contain key: BrokerAppName
[Wed Jun 23 19:01:19 2021] WARN:get_config_value:202: JSON does not contain key: zmq_recv_hwm
[Wed Jun 23 19:01:19 2021] WARN:get_config_value:202: JSON does not contain key: zmq_connect_retries
[Wed Jun 23 19:01:19 2021] WARN:get_config_value:202: JSON does not contain key: brokered
[Wed Jun 23 19:01:19 2021] INFO:UdfManager:105: max_workers: 4
[Wed Jun 23 19:01:19 2021] INFO:run:228: UDFManager thread started
[Wed Jun 23 19:01:19 2021] INFO:run:228: UDFManager thread started
[Wed Jun 23 19:01:19 2021] INFO:run:228: UDFManager thread started
[Wed Jun 23 19:01:19 2021] INFO:run:228: UDFManager thread started
[Wed Jun 23 19:01:30 2021] WARN:get_config_value:202: JSON does not contain key: zmq_connect_retries
[Wed Jun 23 19:01:30 2021] INFO:start:184: Publisher thread started...
[Wed Jun 23 19:01:30 2021] INFO:start:189: Started udf manager
[Wed Jun 23 19:01:30 2021] INFO:start:194: Subscriber thread started...

```

4. Press Ctrl+C on your keyboard to stop the log file.

NOTE You must stop the containers to close the visualizer.

Step 2: Stop the Edge Insights for Industrial Containers

1. Stop the Edge Insights for Industrial containers:

```
cd $HOME/edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights/build
sudo sg docker -c 'docker-compose down'
```


where <version> indicates the downloaded version of Edge Insights for Industrial.

```

intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$ sudo
sg docker -c 'docker-compose down'
Stopping ia_grafana ... done
Stopping ia_kapacitor ... done
Stopping ia_factoryctrl_app ... done
Stopping ia_influxdbconnector ... done
Stopping ia_telegraf ... done
Stopping ia_imagestore ... done
Stopping ia_tls_remoteagent ... done
Stopping ia_opcua_export ... done
Stopping ia_rest_export ... done
Stopping ia_video_analytics ... done
Stopping ia_zmq_broker ... done
Stopping ia_visualizer ... done
Stopping ia_video_ingestion ... done
Stopping ia_web_visualizer ... done
WARNING: Found orphan containers (ia_etcd) for this project. If you removed or renamed this service in y
our compose file, you can run this command with the --remove-orphans flag to clean it up.
Removing ia_grafana ... done
Removing ia_kapacitor ... done
Removing ia_factoryctrl_app ... done
Removing ia_influxdbconnector ... done
Removing edgeinsightssoftware_ia_azure_bridge_1 ... done
Removing edgeinsightssoftware_ia_azure_simple_subscriber_1 ... done
Removing ia_telegraf ... done
Removing ia_imagestore ... done
Removing ia_tls_remoteagent ... done
Removing ia_opcua_export ... done
Removing ia_rest_export ... done
Removing ia_video_analytics ... done
Removing ia_zmq_broker ... done
Removing ia_visualizer ... done
Removing ia_video_ingestion ... done
Removing ia_web_visualizer ... done
Network eii is external, skipping
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$

```

How the PCB Defect Detection Demo Works

Video frames are sent to a Python* application named **filter** in the Video Ingestion container where unwanted frames are filtered out and frames of interest are passed to a Python application named **classifier**. The classifier application is also in the Video Analytics container. This container is for deep learning inference via the data bus.

Results are saved to a database and passed to the data bus, which is used by the Visualizer container that runs a sample Python application to display the images on the target system.

Summary and Next Steps

In this tutorial, you learned to start Edge Insights for Industrial and see the results of the demo application as it performed a quality control inspection on a video of PCBs. You also learned how the application works at a high level and are more familiar with its modules and structure. For more Edge Insights for Industrial features/configurations, refer to `edge_insights_industrial/`

`Edge_Insights_for_Industrial_<version>/IEdgeInsights/README.md`, where <version> indicates the downloaded version of Edge Insights for Industrial.

As a next step, see the [Textile Defect Classifier](#) tutorial.

Textile Defect Classifier

In this tutorial, you'll run the Textile Defect Classifier. By following this tutorial, you will learn:

- How to add a new custom/user application to Edge Insights for Industrial
- How to rebuild Edge Insights for Industrial with this new application
- How to visualize the results of the custom application

Textile Defect Classifier is a reference implementation that can be executed as part of the Edge Insights for Industrial package. This reference implementation provides an AI-enabled approach to classify an input frame from a textile inspection camera as defective or good.

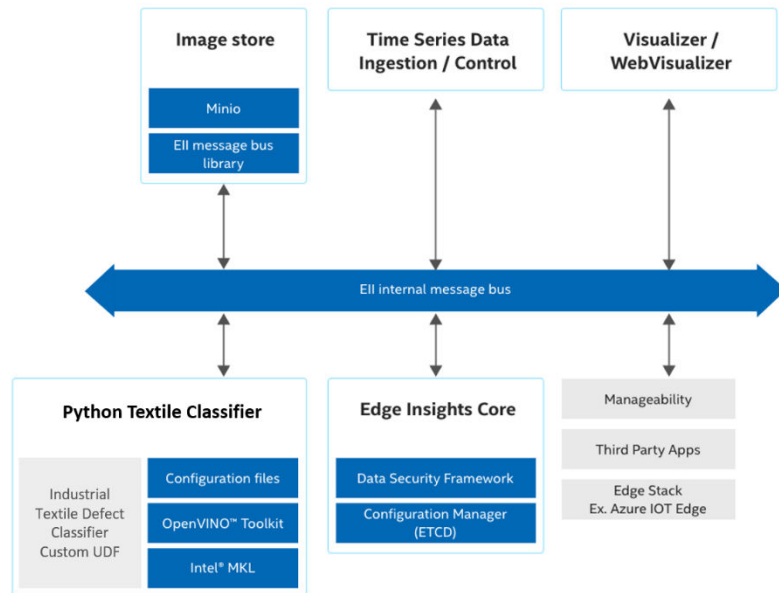
NOTE Be aware that screenshots may show a package version number that is different from the current release. See the **Release Notes <release-notes>** for information on the current release.

How the Textile Defect Classifier Works

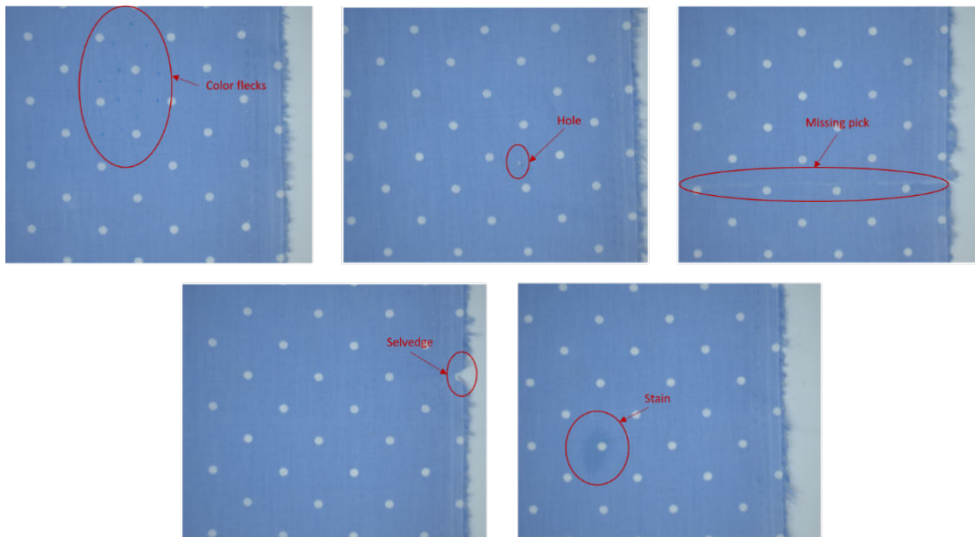
The tutorial shows an example of how the Intel® Distribution of OpenVINO™ toolkit (OpenVINO™) optimized classification networks can be leveraged in industrial quality inspection applications and deployed using Edge Insights for Industrial. This tutorial includes:

- Textile inspection video: Sample video that is used to simulate a textile inspection camera stream. The video includes 5 types of common textile defects, some of which are not easily noticeable to the human eye.
- Textile Defect Classifier Custom UDF: Ingest the frames from the sample test video and performs inference on each frame to classify it as defective or not. The rate at which each frame is classified is also recorded and displayed on the output Visualizer window of the Edge Insights for Industrial software stack. The UDF utilizes the Inference Engine module from the Intel® Distribution of OpenVINO™ toolkit to optimize the inference on Intel hardware.
- Configuration files: Visualizer and WebVisualizer configuration files to add the textile defect classifier UDF to the Edge Insights for Industrial pipeline.
- Intel hardware-optimized models: Deep learning classification models to detect textile defects optimized using the Intel® Distribution of OpenVINO™ toolkit; IR files generated by the Model Optimizer.

All communication between the image acquisition from sample video and analytics to detect defects (custom UDF) to visualizing the inspection results (Visualizer) occur over the Edge Insights Internal Message Bus, as shown in the following figure.



The sample textile data included as part of this application has 5 types of common textile defects, as shown in the following figure.



Get Started

The Textile Defect Classifier reference implementation is a plug-and-play application developed for the Edge Insights for Industrial package.

NOTE This application is a reference implementation of how to deploy an OpenVINO™ optimized classification model using the Edge Insights for Industrial pipeline. Please be advised that the deep learning model included as part of this package is not intended to be a ready-to-deploy commercial textile defect classification model. We recommend that you retrain the model with a factory collected dataset before deploying it on the factory floor.

Prerequisites - Edge Insights for Industrial Installation Options

Download and install the Edge Insights for Industrial package after selecting the Textile Defect Classifier Module in the custom download section of the video or video-timeseries use case. Both the use cases will download the video pipeline related services (custom UDF, Visualizer, and WebVisualizer) required to satisfy the prerequisites for this application.

Step 1: Add Textile Defect Classifier UDF components

1. Change working directory to the Textile Defect Classifier package and create an environment variable \$EII_HOME to point to the IEdgeInsights install path for ease of reference:

```
cd $WORKDIR/edge_insights_industrial/Edge_Insights_for_Industrial_<version>/
textile_defect_classifier/textile_defect_classifier/

EII_HOME=$WORKDIR/edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights
```

NOTE \$WORKDIR in this tutorial refers to the host system path where the Edge Insights for Industrial package was downloaded and unzipped.
<version> indicates the downloaded version of Edge Insights for Industrial.

2. Copy the Textile Defect Classifier UDF file to the Edge Insights for Industrial UDFs directory:

```
cp -r PyTextileClassificationIngestion/ $EII_HOME/CustomUdfs/.
```

3. Modify the Visualizer and WebVisualizer configuration files to add Textile Defect Classifier to the IEdgeInsights pipeline.

- a. Make a copy of the original Visualizer and WebVisualizer config.json files:

```
mv $EII_HOME/Visualizer/config.json $EII_HOME/Visualizer/config.json_original
```

```
mv $EII_HOME/WebVisualizer/config.json $EII_HOME/WebVisualizer/config.json_original
```

- b. Copy the new Visualizer and WebVisualizer config.json files to execute the textile classification application.

Optionally, edit the PyTextileClassificationIngestion/config.json to use the inference hardware of choice [CPU/GPU/Myriad/HDDL]. Default is CPU.

Note that the deep learning model used in the reference implementation is a deep model, and hence will take a few minutes for the initial loading into an accelerator (GPU/Myriad/HDDL) memory.

```
cp vis_config.json $EII_HOME/Visualizer/config.json
cp webvis_config.json $EII_HOME/WebVisualizer/config.json
```

- c. Copy the custom UDF yml file to include the custom textile UDF container:

```
cp video-streaming-textileUDF.yml $EII_HOME/build/usecases/.
```

4. Follow the Edge Insights for Industrial instructions to provision and launch the containers:

```
# Configure the IEdgeInsights pipeline to use UDF, Visualizer & WebVisualizer containers
cd $EII_HOME/build/
sudo python3 builder.py -f usecases/video-streaming-textileUDF.yml

# Provision
cd $EII_HOME/build/provision
sudo -E ./provision.sh ../docker-compose.yml

# Build
cd $EII_HOME/build/
sudo sg docker -c 'docker-compose -f docker-compose-build.yml build
python_textile_classification'
sudo sg docker -c 'docker-compose up -d'

# Enable visualizer display
xhost +
```

The result looks similar to:

```
intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_f
File Edit View Search Terminal Help
Step 1/13 : ARG EII_VERSION
Step 2/13 : ARG EII_USER_NAME
Step 3/13 : ARG DOCKER_REGISTRY
Step 4/13 : FROM ${DOCKER_REGISTRY}openedgeinsights/ia_video_ingestion:$EII_VERSION
---> 7e07ff0b3451
Step 5/13 : LABEL description="Weld Textile Defect Classification UDF Image"
---> Using cache
---> a4ecf0b6d19f
Step 6/13 : HEALTHCHECK NONE
---> Using cache
---> d6a9a2a331b3
Step 7/13 : WORKDIR /app
---> Using cache
---> b50c8326a82d
Step 8/13 : COPY ./textile ./textile
---> Using cache
---> 7f2968a1fa47
Step 9/13 : COPY ./textile.mp4 ./test_videos/textile.mp4
---> Using cache
---> a5dcdce1d908
Step 10/13 : USER root
---> Using cache
---> 14d04e899229
Step 11/13 : COPY requirements.txt .
---> Using cache
---> 641287824c86
Step 12/13 : RUN pip3 install --user -r requirements.txt
---> Using cache
---> 4197557ca95a
Step 13/13 : USER $EII_USER_NAME
---> Running in ce26b2d6d34d
Removing intermediate container ce26b2d6d34d
---> 37cb9b346a6b
Successfully built 37cb9b346a6b
Successfully tagged python_textile_classification:2.6
Use 'docker scan' to run Snyk tests against images to find vulnerabilities and lea
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IED
```

```
intel@edgesoftware: ~/edge_insights_i
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_fo
WARNING: Found orphan containers (ia_etcd, ia_etcd_provision)
you can run this command with the --remove-orphans flag to cle
Creating ia_visualizer ... done
Creating python_textile_classification ... done
Creating ia_web_visualizer ... done
Creating ia_etcd_ui ... done
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_fo
```

NOTE Be aware that screenshots may show a package version number that is different from the current release. See the [Release Notes](#) for information on the current release.

Step 2: Check the Reference Implementation

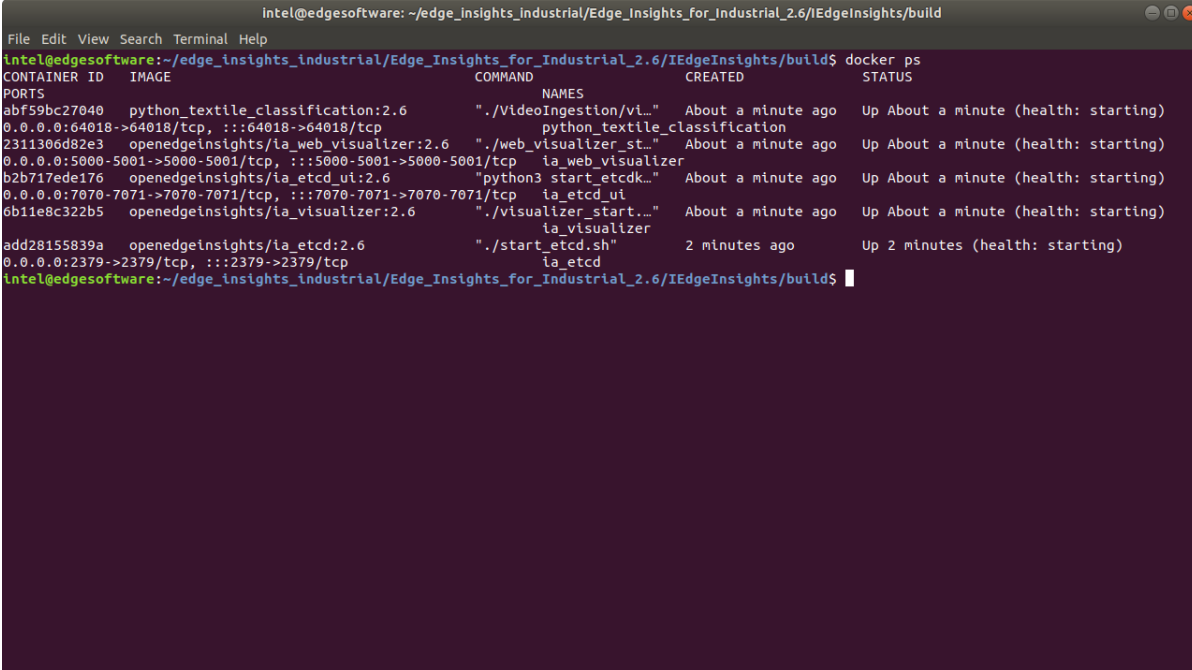
Edge Insights for Industrial builds the custom UDF, Visualizer, and WebVisualizer containers. The UDF container streams the sample textile video and performs classification on each frame to detect occurrence of defects. Once the frame has been classified, the image is displayed on the Visualizer output window with the classification result and inference time.

Execute the following command to ensure all the containers are running without errors:

```
sudo docker ps
```

Check for Success

If it was successful, the results will be similar to:

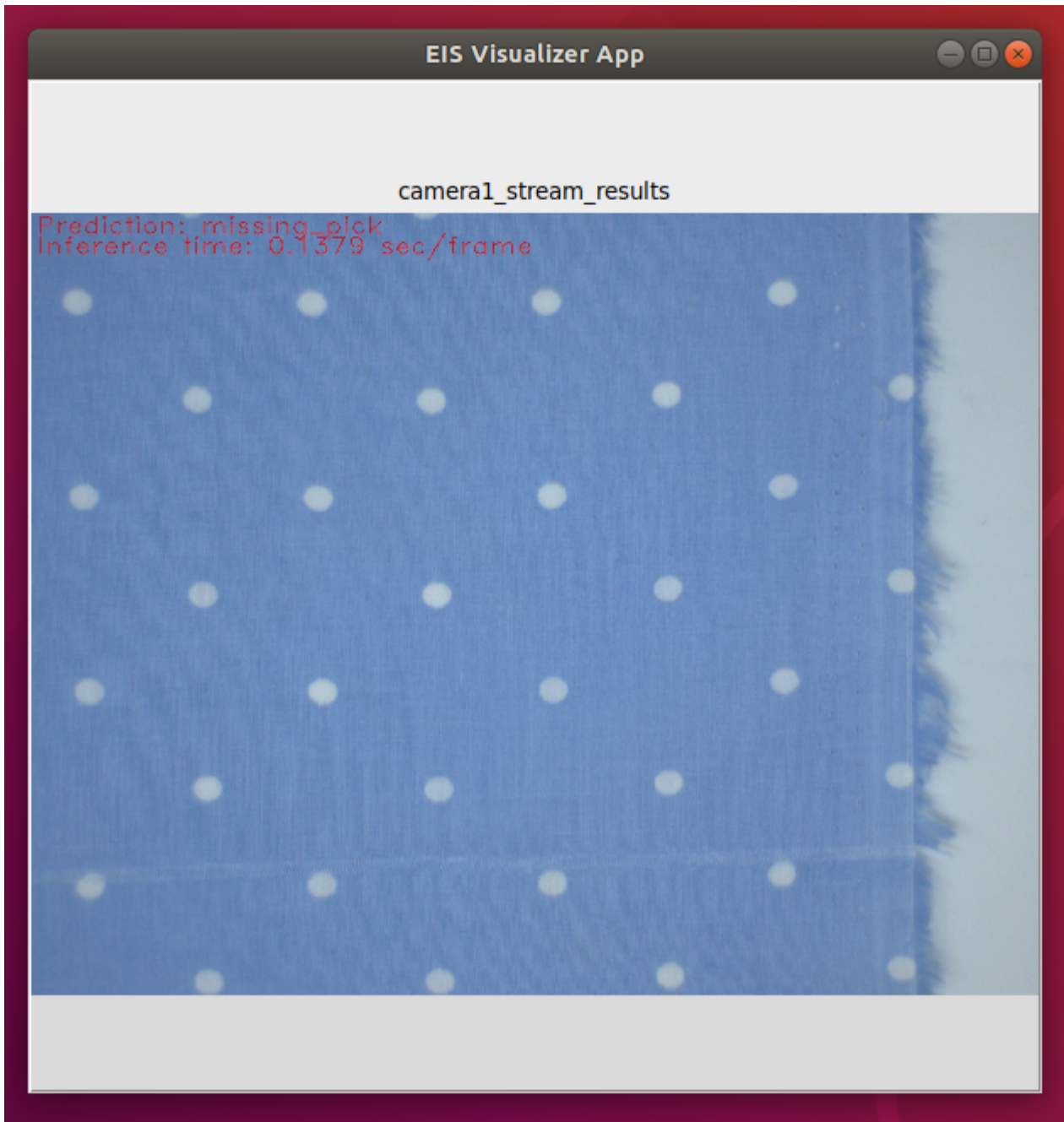


```

intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
PORTS
abf59bc27040   python_textile_classification:2.6    ".VideoIngestion/vi..." About a minute ago Up About a minute (health: starting)
0.0.0.0:64018->64018/tcp, :::64018->64018/tcp      python_textile_classification
2311306d82e3   openedgeinsights/ia_web_visualizer:2.6 ".web_visualizer_st..." About a minute ago Up About a minute (health: starting)
0.0.0.0:5000-5001->5000-5001/tcp, :::5000-5001->5000-5001/tcp  ia_web_visualizer
b2b717ede176   openedgeinsights/ia_etcd_ui:2.6      "python3 start_etcdk..." About a minute ago Up About a minute (health: starting)
0.0.0.0:7070-7071->7070-7071/tcp, :::7070-7071->7070-7071/tcp  ia_etcd_ui
6b11e8c322b5   openedgeinsights/ia_visualizer:2.6    "./visualizer_start..." About a minute ago Up About a minute (health: starting)
                                ia_visualizer
add28155839a   openedgeinsights/ia_etcd:2.6         "./start_etcd.sh"       2 minutes ago   Up 2 minutes (health: starting)
0.0.0.0:2379->2379/tcp, :::2379->2379/tcp          ia_etcd
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$

```

Visualizer Output



The sample textile video has 5 types of defects that can be monitored using the visualizer output.

Step 3: Stop the Vision Pipeline

To stop Edge Insights for Industrial software, execute the following command:

```
cd $EII_HOME/build  
sudo sg docker -c 'docker-compose down'
```

ATTENTION: To revert to the default PCB Demo application, move the Visualizer and WebVisualizer config.json files to the original copies [Step 1.3.a] and rebuild using the instructions in Step 1.4.

Troubleshooting

Troubleshooting a custom UDF application

The Textile Defect Classifier UDF file includes a `tester` code snippet that allows the file to be tested outside of the Edge Insights for Industrial environment on individual images to aid in debugging. Executing the application outside of the Edge Insights for Industrial deployment environment, requires OpenVINO™ to be installed on the development system.

Please follow instructions on the [OpenVINO official page](#) for installation steps for different inference hardware (CPU/GPU/Myriad).

To execute the application on the host development system, follow the instructions:

```
# Install Python dependencies
pip3 install opencv-python==4.4.0.42
pip3 install tensorflow==1.15.0

# Set OpenVINO environment variables
source /opt/intel/openvino_2021/bin/setupvars.sh -pyver 3.6
```

Run inference on a single test image, where `<version>` indicates the downloaded version of Edge Insights for Industrial:

```
cd $WORKDIR/edge_insights_industrial/Edge_Insights_for_Industrial_<version>/
textile_defect_classifier/textile_defect_classifier/

# Run inference on single test image
cd PyTextileClassificationIngestion/
python3 textile/textile_classifier.py \
    --model ./textile/ref/model.xml \
    --label ./textile/ref/labels.txt \
    --image ./sample_data/frame_a.png
```

Expected Output


```
intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_for_Industrial_2.4/textile_defect_classifier/textile_defect_classifier
File Edit View Search Terminal Help
(m_classf) intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.4/textile_defect_classifier/textile_defect_classifier$ source /opt/intel/openvino_2021/bin/setupvars.sh -pyver 3.6
python_version = 3.6
[setupvars.sh] OpenVINO environment initialized
(m_classf) intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.4/textile_defect_classifier/textile_defect_classifier$ python3 textile/textile_classifier.py \
> --model ./textile/ref/model.xml \
> --label ./textile/ref/labels.txt \
> --image ./sample_data/frame_a.png

Image : frame_a.png
INFO:TEXTILE_DEFECT_CLASSIFICATION:Prediction : color_flecks
INFO:TEXTILE_DEFECT_CLASSIFICATION:Inference time : 0.09683394432067871 sec
*****
(m_classf) intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.4/textile_defect_classifier/textile_defect_classifier$
```

Run inference on a directory of test images to calculate accuracy information of the model, where <version> indicates the downloaded version of Edge Insights for Industrial:

```
cd $WORKDIR/edge_insights_industrial/Edge_Insights_for_Industrial_<version>/
textile_defect_classifier/textile_defect_classifier/

# Run inference on a directory of test images
cd PyTextileClassificationIngestion/
python3 textile/textile_classifier.py \
    --model ./textile/ref/model.xml \
    --label ./textile/ref/labels.txt \
    --dir ./sample_data/
```

Expected Output

```
intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_for_Industrial_2.4/textile_defect_classifier/textile_defect_classifier
File Edit View Search Terminal Help
> --model ./textile/ref/model.xml \
> --label ./textile/ref/labels.txt \
> --dir ./sample_data/
['./sample_data/frame_b.png', './sample_data/frame_c.png', './sample_data/frame_nodect.png', './sample_data/frame_d.png', './sample_data/frame_e.png', './sample_data/frame_a.png']

Image : frame_b.png
INFO:TEXTILE_DEFECT_CLASSIFICATION:Prediction : hole
INFO:TEXTILE_DEFECT_CLASSIFICATION:Inference time : 0.10081005096435547 sec
*****

Image : frame_c.png
INFO:TEXTILE_DEFECT_CLASSIFICATION:Prediction : missing_pick
INFO:TEXTILE_DEFECT_CLASSIFICATION:Inference time : 0.08916974067687988 sec
*****

Image : frame_nodect.png
INFO:TEXTILE_DEFECT_CLASSIFICATION:Prediction : good
INFO:TEXTILE_DEFECT_CLASSIFICATION:Inference time : 0.08947467803955078 sec
*****

Image : frame_d.png
INFO:TEXTILE_DEFECT_CLASSIFICATION:Prediction : selvedge
INFO:TEXTILE_DEFECT_CLASSIFICATION:Inference time : 0.08903717994689941 sec
*****

Image : frame_e.png
INFO:TEXTILE_DEFECT_CLASSIFICATION:Prediction : stain
INFO:TEXTILE_DEFECT_CLASSIFICATION:Inference time : 0.0909273624420166 sec
*****

Image : frame_a.png
INFO:TEXTILE_DEFECT_CLASSIFICATION:Prediction : color_flecks
INFO:TEXTILE_DEFECT_CLASSIFICATION:Inference time : 0.0904548168182373 sec
*****
(m_classf) intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.4/textile_defect_classifier/textile_defect_classifier$
```

Summary and Next Steps

In this tutorial, you successfully ran the Textile Defect Classifier application and displayed the result using the Edge Insights for Industrial Visualizer output.

To get access to the deep learning training algorithm that was used to generate the Textile Defect classification model, please reach out to your Intel account manager.

As a next step, see the [Industrial Text Line Recognition](#) tutorial.

Industrial Text Line Recognition

In this tutorial, you'll run the Industrial Text Line Recognition reference implementation to verify that Edge Insights for Industrial was installed successfully and to start getting familiar with its modules and structure. By following this tutorial, you will learn:

- How to add a new custom/user application to Edge Insights for Industrial
- How to rebuild Edge Insights for Industrial with this new application
- How to visualize the results of the custom application

NOTE Be aware that screenshots may show a package version number that is different from the current release. See the [Release Notes](#) for information on the current release.

This reference implementation provides an AI-enabled approach to recognize characters from print text line using the deep learning method.

Industrial Text Line Recognition is a reference implementation that can be executed as part of the [Edge Insights for Industrial package](#). If you have not installed that package yet, you can download it [here](#) and then follow the [Edge Insights for Industrial installation instructions](#).

Target System Requirements

- Ubuntu* 18.04 LTS

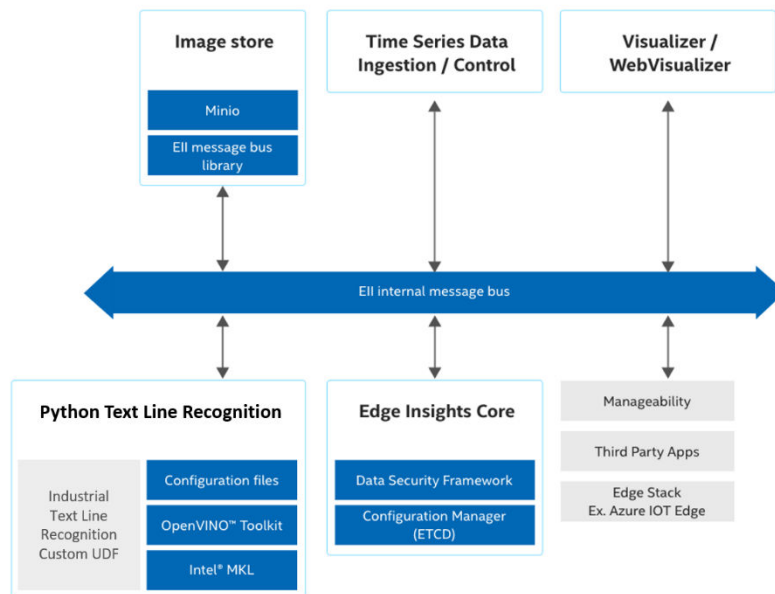
- Intel® Core™ i5 processor or above with 16GB of RAM

How it Works

The application uses the inference engine included in the Intel® Distribution of OpenVINO™ toolkit. The reference implementation shows an example of how Industrial Text Line Recognition algorithm can be leveraged in industrial scenarios.

- Print text line video: Sample video that is used to simulate a live camera stream. The video includes 300 print text images.
- Inference Module: Ingest the frames from the camera stream and performs inference on each frame to recognize characters. The predicted result of each frame is also recorded and displayed on the output Visualizer window of the Edge Insights for Industrial software stack.

All communication between modules occurs over the Edge Insights for Industrial Message Bus. This reference implementation is a sample application that can be executed using the established pipeline.



Get Started

The Industrial Text Line Recognition reference implementation is a plug-and-play application developed for the Edge Insights for Industrial package.

NOTE This application is a reference implementation of how to deploy an OpenVINO™ optimized Industrial Text Line Recognition model using the Edge Insights for Industrial pipeline and not a production-ready custom commercially deployable Industrial Text Line Recognition tool.

Industrial Text Line Recognition is a reference implementation that can be executed as part of the [Edge Insights for Industrial package](#). If you have not installed that package yet, you can download it [here](#) and then follow the [Edge Insights for Industrial installation instructions](#).

Step 1: Add Industrial Text Line Recognition UDF Components

1. Copy the Industrial Text Line Recognition UDF file to the Edge Insights for Industrial UDFs directory:

```
# Set Edge Insights for Industrial home path as environment variable
EII_HOME=<path_to_install_dir/IEdgeInsights/>
```

```
# Copy Text Line Recognition UDF file to the UDFs directory
cp -r PyTextLineRecognition/ $EII_HOME/CustomUdfs/.
```

2. Modify the Visualizer and WebVisualizer configuration files to add Industrial Text Line Recognition to the IEdgeInsights pipeline:

NOTE We recommend that you make copies of the original `$EII_HOME/Visualizer/config.json` and `$EII_HOME/WebVisualizer/config.json` files before proceeding. These files needed to be rewritten to modify the pipeline to use the Industrial Text Line Recognition application files instead of the default PCB demo application.

- a. Make copies of the original Visualizer and WebVisualizer `config.json` files:

```
mv $EII_HOME/Visualizer/config.json $EII_HOME/Visualizer/config.json_original
mv $EII_HOME/WebVisualizer/config.json $EII_HOME/WebVisualizer/config.json_original
```

- b. Copy the new Visualizer and WebVisualizer `config.json` files to execute the Industrial Text Line Recognition application.

Optionally, edit the `PyTextLineRecognition/config.json` to use the inference hardware of choice [CPU/GPU/Myriad/HDDL]. Default is CPU.

NOTE The deep learning model used in the reference implementation is a deep model, and hence would take a few minutes for the initial loading into an accelerator (Myriad/HDDL) memory.

```
cp vis_config.json $EII_HOME/Visualizer/config.json
cp webvis_config.json $EII_HOME/WebVisualizer/config.json
```

- c. Copy the custom UDF yml file to include the custom text line recognition UDF container:

```
cp video-streaming-textlineUDF.yml $EII_HOME/build/usecases/.
```

3. Follow the Edge Insights for Industrial instructions to provision and launch the containers:

```
# Configure the IEdgeInsights pipeline to use custom UDF, Visualizer & Web Visualizer containers
cd $EII_HOME/build/
sudo python3 builder.py -f usecases/video-streaming-textlineUDF.yml
```

```
# Provision
cd $EII_HOME/build/provision/
sudo -E ./provision.sh ../docker-compose.yml
```

```
# Build and run
cd $EII_HOME/build/
docker-compose -f docker-compose-build.yml build python_text_line_recognition
docker-compose up -d
```

```
# Enable visualizer display
xhost +
```

The output will look similiar to:

```
intel@edgesoftware: ~/edge_insights
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights
l build python_text_line_recognition
Building python_text_line_recognition
Sending build context to Docker daemon 41.79MB

Step 1/8 : ARG EII_VERSION
Step 2/8 : ARG DOCKER_REGISTRY
Step 3/8 : FROM ${DOCKER_REGISTRY}openedgeinsights/ia_video
---> 7e07ff0b3451
Step 4/8 : LABEL description="Text Line Recognition UDF Image"
---> Using cache
---> 1515e94b2b81
Step 5/8 : HEALTHCHECK NONE
---> Using cache
---> e8a3757e5c29
Step 6/8 : WORKDIR /app
---> Using cache
---> 04e3ef8692a9
Step 7/8 : COPY ./text_line ./text_line
---> Using cache
---> 6e03519ba10a
Step 8/8 : COPY ./text_line.avi ./test_videos/text_line.avi
---> Using cache
---> 5bcbd3566d67
Successfully built 5bcbd3566d67
Successfully tagged python_text_line_recognition:2.6
Use 'docker scan' to run Snyk tests against images to find
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights
WARNING: Found orphan containers (ia_etcd, ia_etcd_provisioning)
you can run this command with the --remove-orphans flag to
Creating python_text_line_recognition ... done
Creating ia_web_visualizer ... done
Creating ia_visualizer ... done
Creating ia_etcd_ui ... done
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights
```

Step 2: Check the Reference Implementation

The Edge Insights for Industrial builds the custom UDF, Visualizer and WebVisualizer containers. The custom UDF streams the sample video and performs Industrial Text Line Recognition on each frame to recognize the characters. Once the frame has been processed, the image is displayed on the Visualizer output window with the predicted result and inference time.

Execute the following command to ensure all the containers are running without errors:

```
sudo docker ps
```

Check for Success

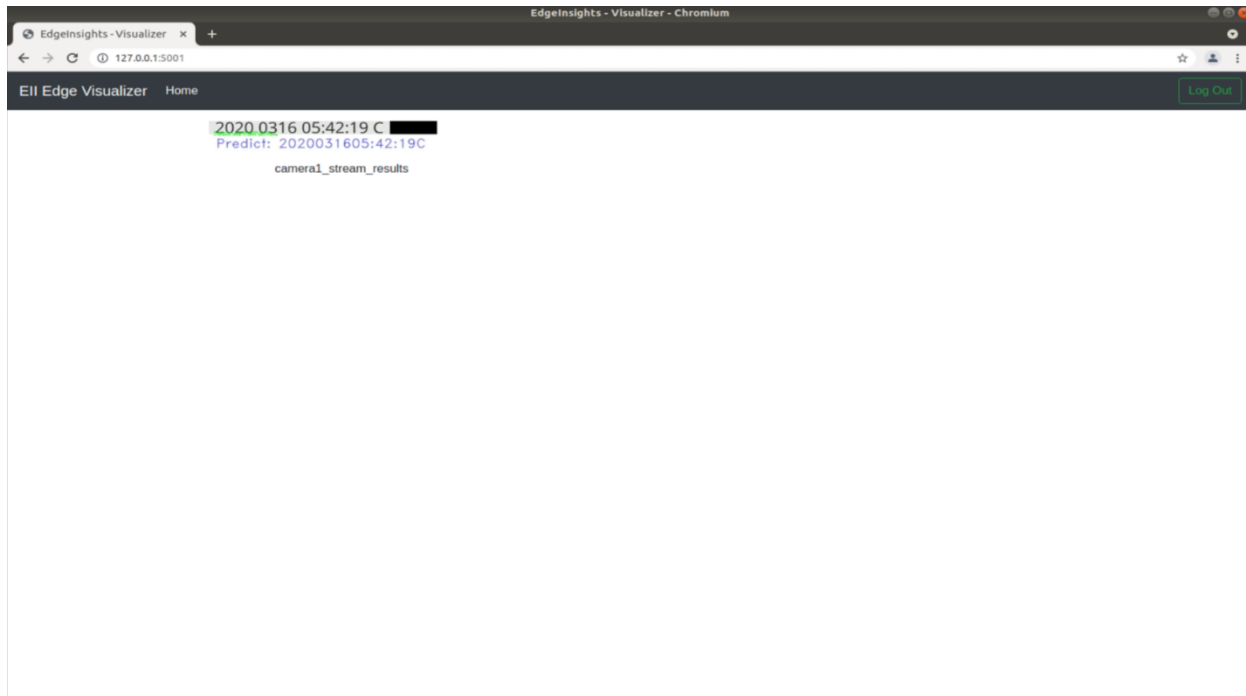
If it was successful, the results will be similar to:

```

intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_f
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IED
CONTAINER ID      IMAGE                                     COMMAND                                     C
NAMES
16694f5c0714      openedgeinsights/ia_visualizer:2.6      "/visualizer_start..."                 1
ia_visualizer
a4d8c7def58c      openedgeinsights/ia_web_visualizer:2.6  "/web_visualizer_st..."                 1
ia_web_visualizer
5000-5001->5000-5001/tcp, :::5000-5001->5000-5001/tcp
3fdd5bbf37ac      openedgeinsights/ia_etcd_ui:2.6         "python3 start_etcdk..."               1
ia_etcd_ui
7070-7071->7070-7071/tcp, :::7070-7071->7070-7071/tcp
e473b657cb23      python_text_line_recognition:2.6        "/VideoIngestion/vi..."                 1
python_text_line_recogniti
64018->64018/tcp, :::64018->64018/tcp
11b346c6662a      openedgeinsights/ia_etcd:2.6            "/start_etcd.sh"                         5
ia_etcd
2379->2379/tcp, :::2379->2379/tcp
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IED

```

WebVisualizer Output



The sample video contains 300 printed text line images similar to the ones shown below:

2019 0221 05:09:

2019 0320 02:02:

2019 0717 22:04:

2020 0510 02:48:

2020 1001 10:09:

(Optional) Stop the Vision Pipeline

To stop the Edge Insights for Industrial software, execute the following command:

```
cd $EII_HOME/build/  
docker-compose down
```

To revert to the default PCB Demo application, move the Visualizer and WebVisualizer `config.json` files to the original copies [Step 1.2.a] and rebuild using the instructions in Step 1.3.

(Optional) Standalone Testing

Testing the Industrial Text Line Recognition reference implementation outside of the Edge Insights for Industrial deployment environment requires OpenVINO™ to be installed on the development system. Please follow instructions on the [OpenVINO™ website](#) for installation steps for different inference hardware (CPU/Myriad/HDDL).

To execute the application on the host development system, follow the instructions below:

```
# Install dependencies
pip install opencv-python==4.4.0.42

# Set OpenVINO env variables
source /opt/intel/openvino/bin/setupvars.sh -pyver 3.6

# Run inference on single test image
python3 text_line_recognizer.py \
    --model <path/to/model/XML/file> \
    --label <path/to/labels/txt/file> \
    --image <path/to/input/test/image>

Ex.:
python3 text_line/text_line_recognizer.py \
    --model text_line/ref/model.xml \
    --label text_line/ref/labels.txt \
    --image ./sample_data/2020081707:39:59E_217.jpg

Expected output:
Image : ./sample_data/2020081707:39:59E_217.jpg
INFO:Textline_RECOGNITION:Prediction : 2020081707:39:59E
INFO:Textline_RECOGNITION:Inference time : 0.08637118339538574 sec
*****

# Run inference on directory of test images
python3 text_line_recognizer.py \
    --model <path/to/model/XML/file> \
    --label <path/to/labels/txt/file> \
    --dir <path/to/dir/of/test/images>

Ex.:
python3 text_line/text_line_recognizer.py \
    --model text_line/ref/model.xml \
    --label text_line/ref/labels.txt \
    --dir ./sample_data/

Expected output:
Image : ./sample_data/2019011413:26:44E_21.jpg
INFO:Textline_RECOGNITION:Prediction : 2019011413:26:44E
INFO:Textline_RECOGNITION:Inference time : 0.07046294212341309 sec
*****

Image : ./sample_data/2020091921:05:31M_148.jpg
INFO:Textline_RECOGNITION:Prediction : 2020091921:05:31M
INFO:Textline_RECOGNITION:Inference time : 0.0416107177734375 sec
*****
```

Summary and Next Steps

In this tutorial, you successfully ran the Industrial Text Line Recognition application and displayed the result using the Edge Insights for Industrial Visualizer output.

To get access to the deep learning training algorithm that was used to generate the Industrial Text Line Recognition model, please contact your Intel account manager.

As a next step, see the [Industrial Surface Defect Detection](#) tutorial.

Industrial Surface Defect Detection

In this tutorial, you'll run the Industrial Surface Defect Detection reference implementation to verify that Edge Insights for Industrial was installed successfully and to start getting familiar with its modules and structure. By following this tutorial, you will learn:

- How to add a new custom/user application to Edge Insights for Industrial
- How to rebuild Edge Insights for Industrial with this new application
- How to visualize the results of the custom application

NOTE Be aware that screenshots may show a package version number that is different from the current release. See the [Release Notes](#) for information on the current release.

This reference implementation provides an AI-enabled approach to segment defects from an inspection camera.

Industrial Surface Defect Detection is a reference implementation that can be executed as part of the [Edge Insights for Industrial package](#). If you have not installed that package yet, you can download it [here](#) and then follow the [Edge Insights for Industrial installation instructions](#).

Target System Requirements

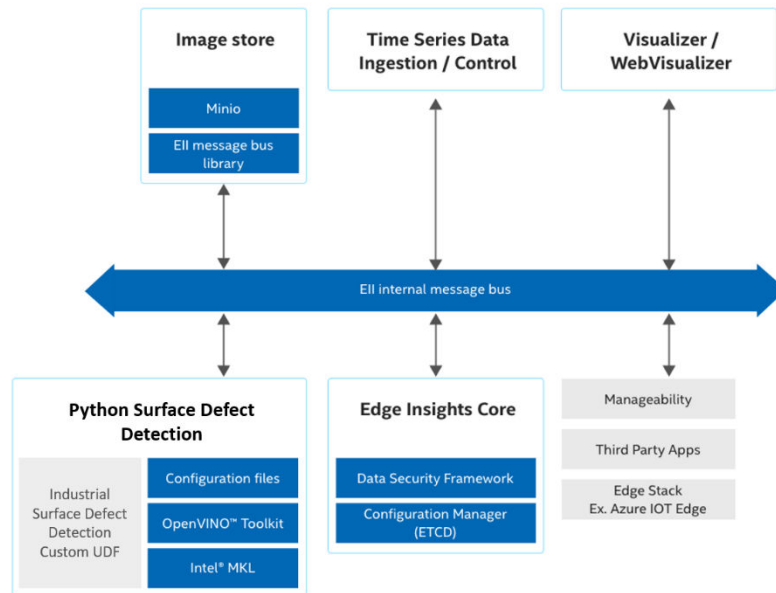
- Ubuntu* 18.04 LTS
- Intel® Core™ i5 processor or above with 16 GB of RAM

How it Works

The application uses the inference engine included in the Intel® Distribution of OpenVINO™ toolkit. The reference implementation shows an example of how segmentation networks can be leveraged in industrial quality inspection applications.

- Surface inspection video: Sample video that is used to simulate a surface defect inspection camera stream. The video includes different types of surface defects.
- Inference Module: Ingests the frames from the camera stream and performs inference on each frame to segment defects. The rate at which each frame is detected is also recorded and displayed on the output Visualizer window of the Edge Insights for Industrial software stack.

All communication between modules occurs over the Edge Insights for Industrial Message Bus. This reference implementation is a sample application that can be executed using the established pipeline.



Get Started

The Industrial Surface Defect Detection reference implementation is a plug-and-play application developed for the Edge Insights for Industrial package.

NOTE This application is a reference implementation of how to deploy an OpenVINO™ optimized segmentation model using the Edge Insights for Industrial pipeline and not a production-ready custom commercially deployable industrial surface defect detection tool.

Industrial Surface Defect Detection is a reference implementation that can be executed as part of the [Edge Insights for Industrial package](#). If you have not installed that package yet, you can download it [here](#) and then follow the [Edge Insights for Industrial installation instructions](#).

Step 1: Add Surface Defect Detection UDF Components

1. Copy the Industrial Surface Defect Detection UDF file to the Edge Insights for Industrial UDFs directory:

```
# Set Edge Insights for Industrial home path as environment variable
EII_HOME=<path_to_install_dir/IEdgeInsights/>

# Copy Surface Defect Detection UDF file to the UDFs directory
cp -r PySurfaceDefectDetection $EII_HOME/CustomUdfs/.
```

2. Modify the Visualizer and WebVisualizer configuration files to add Industrial Surface Defect Detection to the IEdgeInsights pipeline:

NOTE We recommend that you make copies of the original `$EII_HOME/Visualizer/config.json` and `$EII_HOME/WebVisualizer/config.json` files before proceeding. These files needed to be rewritten to modify the pipeline to use the Industrial Surface Defect Detection application files instead of the default PCB demo application.

- a.** Make copies of the original Visualizer and WebVisualizer `config.json` files:

```
mv $EII_HOME/Visualizer/config.json $EII_HOME/Visualizer/config.json_original
mv $EII_HOME/WebVisualizer/config.json $EII_HOME/WebVisualizer/config.json_original
```

- b.** Copy the new Visualizer and WebVisualizer `config.json` files to execute the Industrial Surface Defect Detection application.

Optionally, edit the `PySurfaceDefectDetection/config.json` to use the inference hardware of choice [CPU/GPU/Myriad/HDDL]. Default is CPU.

NOTE The deep learning model used in the reference implementation is a deep model, and hence would take a few minutes for the initial loading into an accelerator (Myriad/HDDL) memory.

```
cp vis_config.json $EII_HOME/Visualizer/config.json
cp webvis_config.json $EII_HOME/WebVisualizer/config.json
```

- c.** Copy the custom UDF `yml` file to include the custom surface defect detection UDF container:

```
cp video-streaming-surfacedefectUDF.yml $EII_HOME/build/usecases/.
```

Download the [DAGM](#) data and convert the Class1-Class6 test images to video `defect_segmentation.avi` with the convert code:

```
import cv2
import glob

img_path = 'Your/DAGM/TEST/IMAGE/PATH'
img_list = glob.glob(img_path + '*.PNG')
fourcc = cv2.VideoWriter_fourcc(*'XVID')
writer = cv2.VideoWriter('surface_defect.avi', fourcc, 2,
                        (512, 512))

for imgi in img_list:
    img = cv2.imread(imgi)
    writer.write(img)
writer.release()
```

- 3.** Follow the Edge Insights for Industrial instructions to provision and launch the containers:

```
# Configure the IEdgeInsights pipeline to use custom UDF, Visualizer & Web Visualizer containers
cd $EII_HOME/build/
sudo python3 builder.py -f usecases/video-streaming-surfacedefectUDF.yml

# Provision
cd $EII_HOME/build/provision/
sudo -E ./provision.sh ../docker-compose.yml

# Build and run
cd $EII_HOME/build/
docker-compose -f docker-compose-build.yml build python_surface_defect_detection
docker-compose up -d
```

```
# Enable visualizer display  
xhost +
```

The output will look similar to:

```
intel@edgesoftware: ~/edge_insights
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights
l build python_surface_defect_detection
Building python_surface_defect_detection
Sending build context to Docker daemon 28.39MB

Step 1/8 : ARG EII_VERSION
Step 2/8 : ARG DOCKER_REGISTRY
Step 3/8 : FROM ${DOCKER_REGISTRY}openedgeinsights/ia_video
---> 7e07ff0b3451
Step 4/8 : LABEL description="Surface Defect Detection UDF
---> Using cache
---> eb93b0ee0eec
Step 5/8 : HEALTHCHECK NONE
---> Using cache
---> 803e517b8fa2
Step 6/8 : WORKDIR /app
---> Using cache
---> 05368e6c68f8
Step 7/8 : COPY ./surface_defect ./surface_defect
---> Using cache
---> 831289f652af
Step 8/8 : COPY ./defect_segmentation.avi ./test_videos/def
---> Using cache
---> dda051558380
Successfully built dda051558380
Successfully tagged python_surface_defect_detection:2.6
Use 'docker scan' to run Snyk tests against images to find
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights
WARNING: Found orphan containers (ia_etcd, ia_etcd_provisio
you can run this command with the --remove-orphans flag to
Creating ia_visualizer ... done
Creating python_surface_defect_detection ... done
Creating ia_etcd_ui ... done
Creating ia_web_visualizer ... done
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights
```


Step 2: Check the Reference Implementation

The Edge Insights for Industrial builds the custom UDF, Visualizer and WebVisualizer containers. The custom UDF container streams the sample video and performs segmentation on each frame to detect defects. Once the frame has been segmented, the image is displayed on the Visualizer output window with the segmentation result and inference time.

Execute the following command to ensure all the containers are running without errors:

```
sudo docker ps
```

Check for Success

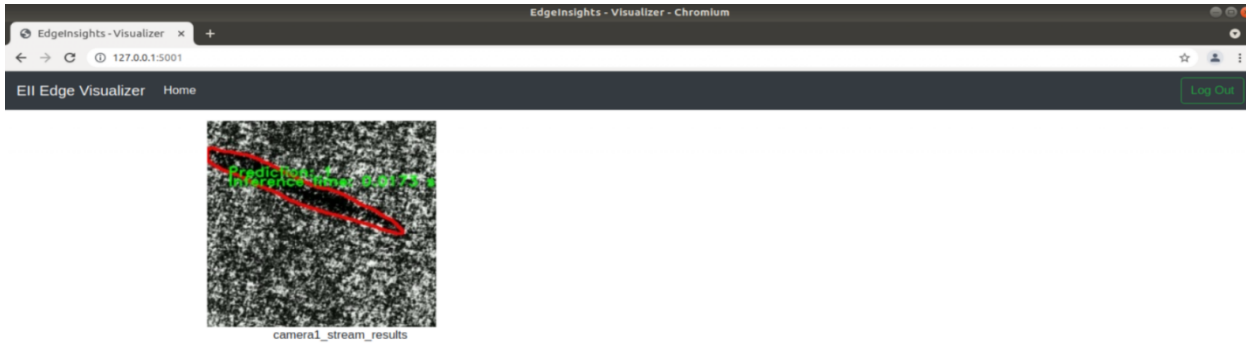
If it was successful, the results will be similar to:

```

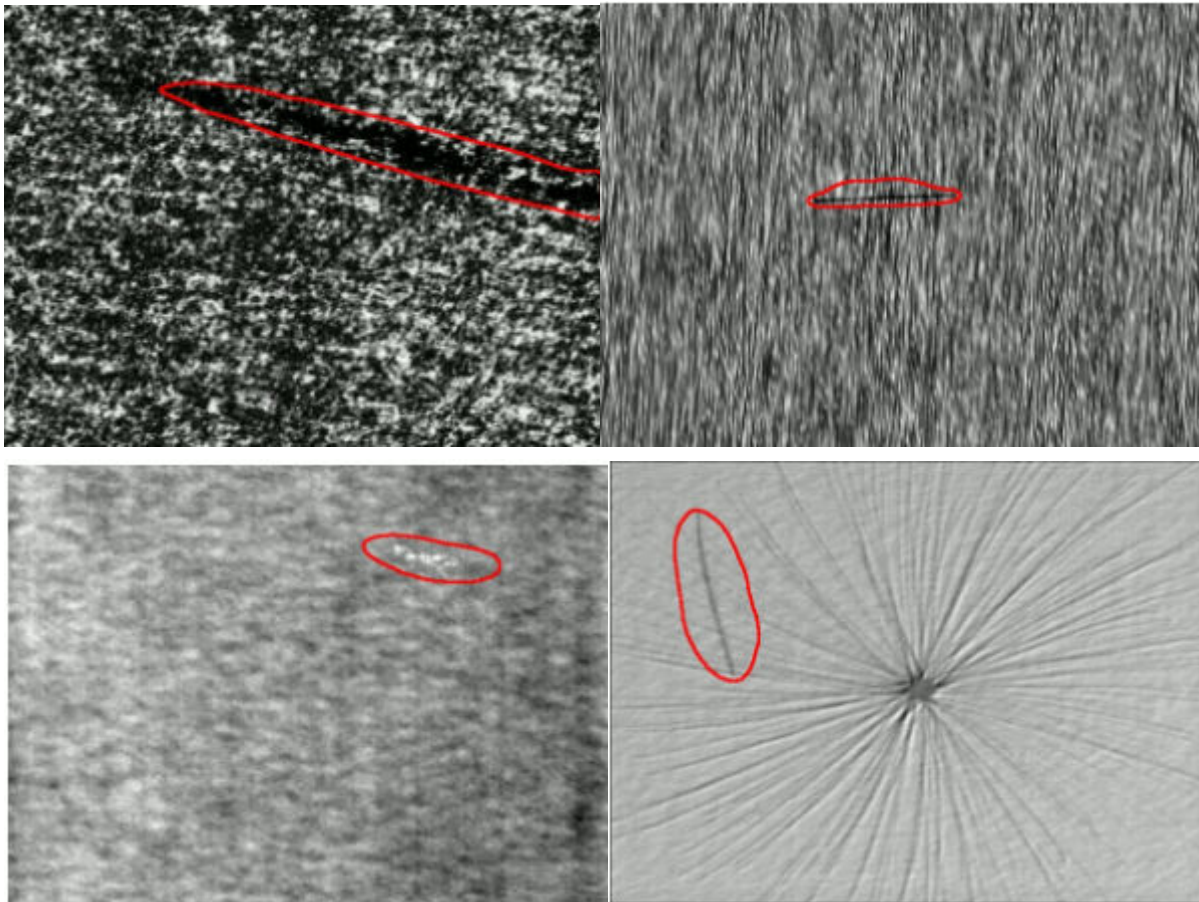
intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_f
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IED
CONTAINER ID      IMAGE                                     COMMAND                                     NAMES
0d3f8badb46d      python_surface_defect_detection:2.6     "/VideoIngestion/vi..."                6
4018->64018/tcp, :::64018->64018/tcp      python_surface_defect_detec
94b44b032622      openedgeinsights/ia_etcd_ui:2.6         "python3 start_etcdk..."                6
070-7071->7070-7071/tcp, :::7070-7071->7070-7071/tcp  ia_etcd_ui
eb65e5b4fb99      openedgeinsights/ia_web_visualizer:2.6   "/web_visualizer_st..."                6
000-5001->5000-5001/tcp, :::5000-5001->5000-5001/tcp  ia_web_visualizer
a80fb7822846      openedgeinsights/ia_visualizer:2.6       "/visualizer_start..."                6
070f8825c152      openedgeinsights/ia_etcd:2.6            "/start_etcd.sh"                        4
379->2379/tcp, :::2379->2379/tcp          ia_etcd
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IED

```

WebVisualizer Output



The sample video has different types of defects that can be monitored using the Web Visualizer:



(Optional) Stop the Vision Pipeline

To stop the Edge Insights for Industrial software, execute the following command:

```
cd $EII_HOME/build/
docker-compose down
```

To revert to the default PCB Demo application, move the Visualizer and WebVisualizer `config.json` files to the original copies [Step 1.2.a] and rebuild using the instructions in Step 1.3.

(Optional) Standalone Testing

Testing the Industrial Surface Defect Detection reference implementation outside of the Edge Insights for Industrial deployment environment requires OpenVINO™ to be installed on the development system. Please follow instructions on the [OpenVINO™ website](#) for installation steps for different inference hardware (CPU/Myriad/HDDL). The reference model is training with DAGM dataset Class1~Class6, you can download it [here](#).

To execute the application on the host development system, follow the instructions below:

```
# Install dependencies
pip install opencv-python==4.4.0.42

# Set OpenVINO env variables
source /opt/intel/openvino/bin/setupvars.sh -pyver 3.6

# Run inference on single test image
python3 defect_segmentation.py \
    --model <path/to/model/XML/file> \
    --label <path/to/labels/txt/file> \
    --image <path/to/input/test/image>

Ex.:
python3 surface_defect/defect_segmentation.py \
    --model surface_defect/ref/model.xml\
    --image sample_data/Class1_0837.png

Expected output:
Image : ../sample_data/Class1_0837.png
INFO:DETECT_SEGMENTATION:Prediction : 1
INFO:DETECT_SEGMENTATION:Inference time : 0.0775909423828125 sec
*****

# Run inference on director of test images
python3 defect_segmentation.py \
    --model <path/to/model/XML/file> \
    --label <path/to/labels/txt/file> \
    --dir <path/to/dir/of/test/images>

Expected output:
Image : ../sample_data/Class6_None-Defect.png
INFO:DETECT_SEGMENTATION:Prediction : 0
INFO:DETECT_SEGMENTATION:Inference time : 0.07543802261352539 sec
*****
```

```
Image : ../sample_data/Class4_0836.png
INFO:DEFECT_SEGMENTATION:Prediction : 1
INFO:DEFECT_SEGMENTATION:Inference time : 0.03906846046447754 sec
*****
...
```

Summary and Next Steps

In this tutorial, you successfully ran the Industrial Surface Defect Detection application and displayed the result using the Edge Insights for Industrial Visualizer output.

To get access to the deep learning training algorithm that was used to generate the Surface Defect Detection model, please contact your Intel account manager.

As a next step, see the [Weld Porosity Detection](#) tutorial.

Weld Porosity Detection

Overview

The Weld Porosity Detection tutorial provides an AI-enabled approach to detect porosity defects in an input frame from a weld inspection camera.

This tutorial can be executed as part of the [Edge Insights for Industrial package](#) . If you have not installed that package yet, you can download it [here](#) and then follow the [Edge Insights for Industrial installation instructions](#) .

Target System Requirements

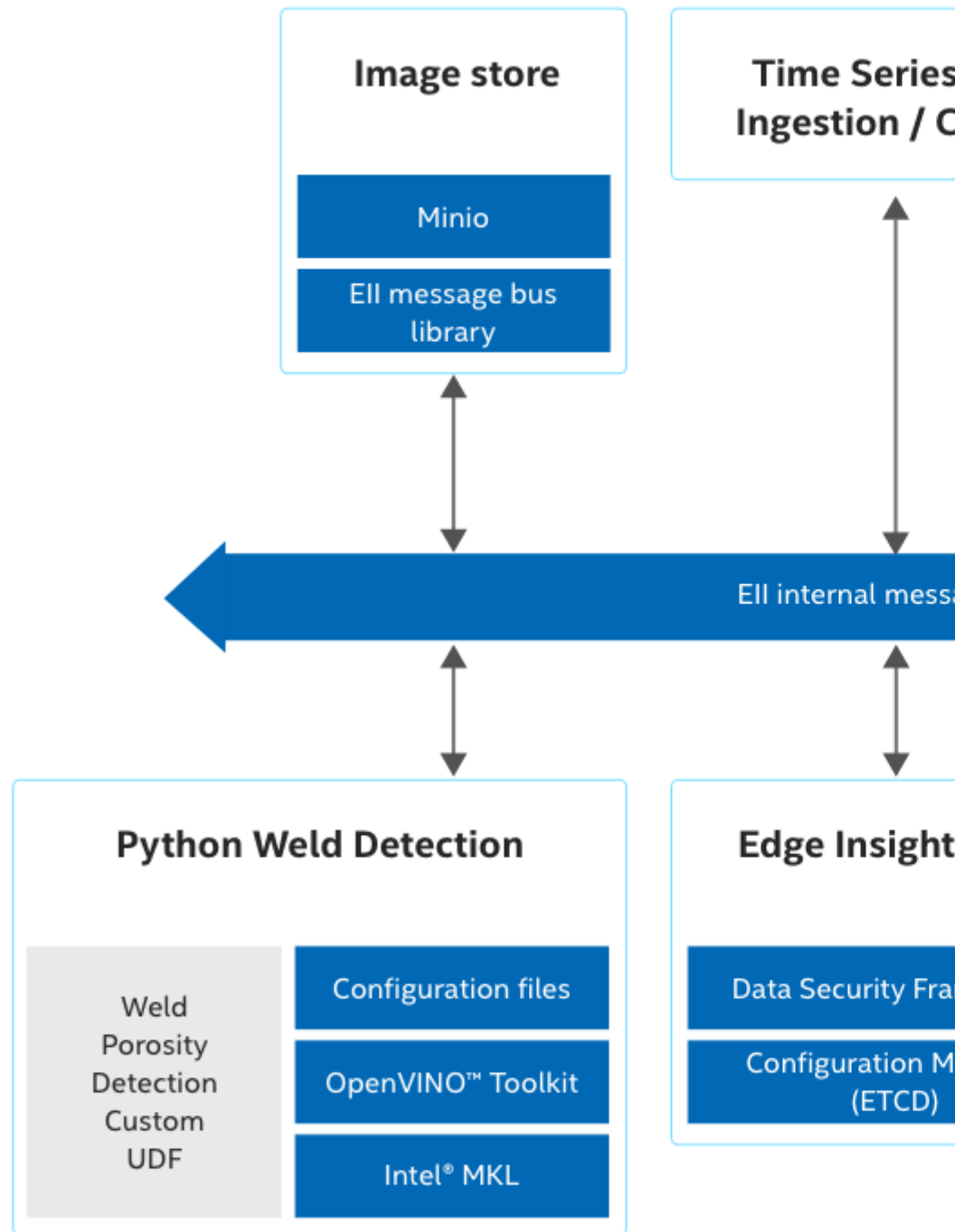
- Ubuntu* 18.04 LTS
- Intel® Core™ i5 processor or above with 16 GB of RAM

How It Works

The Weld Porosity Detection tutorial uses the inference engine included in the [Intel® Distribution of OpenVINO™ toolkit](#). The tutorial shows an example of how detection networks can be leveraged in industrial quality inspection applications.

- **Weld inspection video:** Sample video that is used to simulate a weld inspection camera stream. The video includes porosity defects that occur during the weld process and at random times.
- **Weld Porosity Detection Custom UDF:** Ingest the frames from the Video Ingestion pipeline and performs inference on each frame to classify it as no weld or normal weld or porosity weld. The rate at which each frame is classified is also recorded and displayed on the output Visualizer window of the Edge Insights for Industrial software stack. The UDF utilizes the Inference Engine module from the Intel® Distribution of OpenVINO™ toolkit to optimize the inference on Intel hardware.
- **Configuration files:** The weld porosity detection custom UDF configuration file to add the classification algorithm UDF to the Edge Insights for Industrial pipeline.
- **Intel hardware optimized models:** Deep learning classification models to detect porosity weld defects optimized using the Intel® Distribution of OpenVINO™ toolkit; IR files generated by the Model Optimizer.

All communication between the image acquisition from sample video and analytics to detect defects (VideoIngestion) to visualizing the inspection results (Visualizer) occur over the Edge Insights Internal Message Bus.



Get Started

The Weld Porosity Detection reference implementation is a plug-and-play application developed for the Edge Insights for Industrial package.

NOTE This application is a reference implementation of how to deploy an OpenVINO™ optimized action recognition model using the Edge Insights for Industrial pipeline. Please be advised that the deep learning model included as part of this package is not intended to be a ready-to-deploy commercial weld defect detection model. It is expected that the user would retrain the model with a factory collected dataset before deploying it on factory floor.

Prerequisites

Weld Porosity Detection can be executed as part of the [Edge Insights for Industrial package](#) . If you have not installed that package yet, you can download it [here](#) in the custom download section of the *video* or *video-timeseries* use case. Both the use cases will download the video pipeline related services required to satisfy the pre-requisites for this application. Next, follow the [Edge Insights for Industrial installation instructions](#) .

Step 1: Add Weld Porosity Detection UDF Components

1. Change working directory to the Weld Porosity Detection package and create an environment variable \$EII_HOME to point to the IEdgeInsights install path for ease of reference.

NOTE In this tutorial, the variable \$WORKDIR refers to the host system path where Edge Insights for Industrial package was downloaded and unzipped.
<version> indicates the downloaded version of Edge Insights for Industrial.

```
cd $WORKDIR/edge_insights_industrial/Edge_Insights_for_Industrial_<version>/
weld_porosity_detection/weld_porosity_detection/
EII_HOME=$WORKDIR/edge_insights_industrial/Edge_Insights_for_Industrial_<version>/IEdgeInsights/
```

2. Copy the Weld Porosity Detection custom UDF file to the Edge Insights for Industrial CustomUdfs directory:

```
cp -r PyWeldPorosityIngestion $EII_HOME/CustomUdfs/.
```

Optionally, edit the PyWeldPorosityIngestion/config.json to use the inference hardware of choice [CPU/GPU/Myriad/HDDL]. Default is CPU. Please note that the deep learning model used in the reference implementation is a deep model, and hence would take a few minutes for the initial loading into an accelerator (GPU/Myriad/HDDL) memory.

3. Modify the Visualizer and WebVisualizer configuration to subscribe to the custom UDF publisher port.

NOTE We recommend that you make copies of the original \$EII_HOME/Visualizer/config.json and \$EII_HOME/WebVisualizer/config.json files before proceeding. These files needed to be rewritten to modify the pipeline to use the Weld Porosity Detection application files instead of the default PCB demo application.

- a. Make copies of the original Visualizer and WebVisualizer config.json files:

```
mv $EII_HOME/Visualizer/config.json $EII_HOME/Visualizer/config.json_original
mv $EII_HOME/WebVisualizer/config.json $EII_HOME/WebVisualizer/config.json_original
```


- b. Copy the new Visualizer and WebVisualizer config.json files to subscribe to the the Weld Porosity Detection application:

```
cp vis_config.json $EII_HOME/Visualizer/config.json
cp webvis_config.json $EII_HOME/WebVisualizer/config.json
```

- c. Copy the usecase UDF yml file to include the custom weld UDF container:

```
cp video-streaming-weldUDF.yml $EII_HOME/build/usecases/.
```

4. Follow the Edge Insights for Industrial instructions to provision and launch the containers:

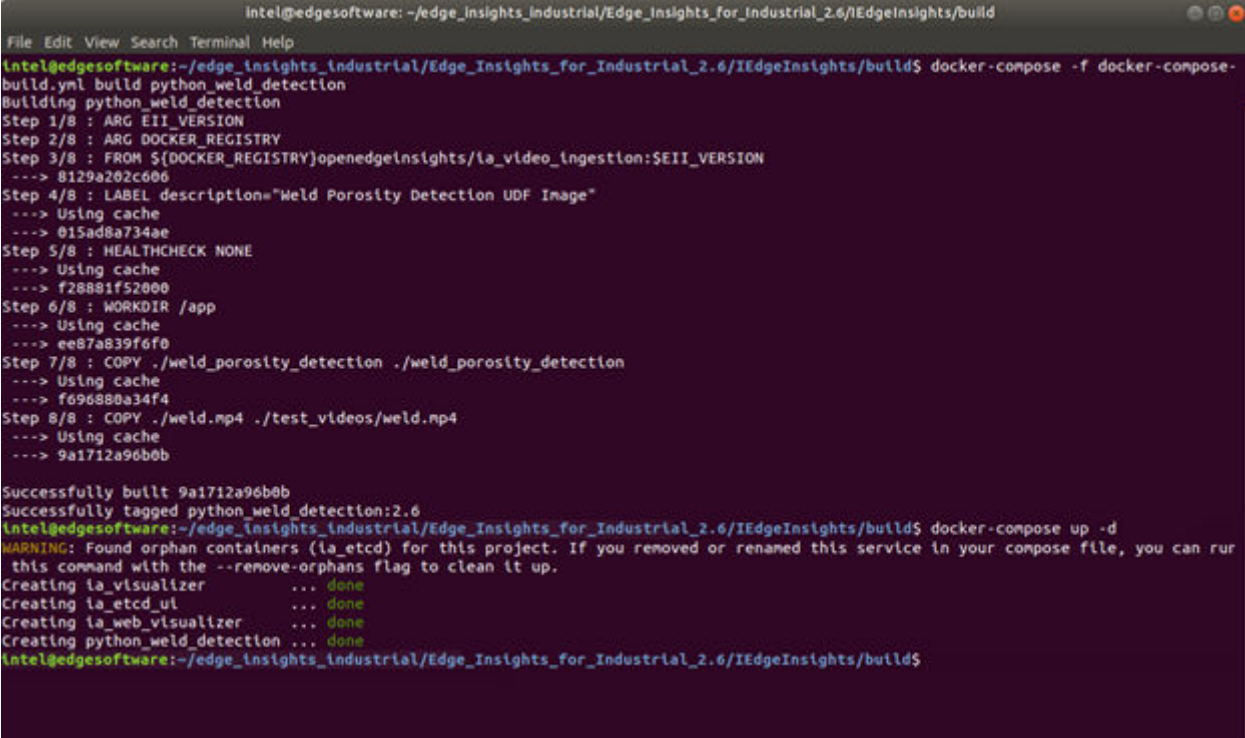
```
# Configure the IEdgeInsights pipeline to use the containers necessary for CustomUDF application
cd $EII_HOME/build/
sudo python3 builder.py -f usecases/video-streaming-weldUDF.yml
```

```
# Provision
cd $EII_HOME/build/provision/
sudo -E ./provision.sh ../docker-compose.yml
```

```
# Build and run
cd $EII_HOME/build/
sudo sg docker -c 'docker-compose -f docker-compose-build.yml build python_weld_detection'
sudo sg docker -c 'docker-compose up -d'
```

```
# Enable visualizer display
xhost +
```

The output will look similar to:



```
intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$ docker-compose -f docker-compose-
build.yml build python_weld_detection
Building python_weld_detection
Step 1/8 : ARG EII_VERSION
Step 2/8 : ARG DOCKER_REGISTRY
Step 3/8 : FROM ${DOCKER_REGISTRY}openedgeinsights/la_video_ingestion:$EII_VERSION
--> 8129a202c606
Step 4/8 : LABEL description="Weld Porosity Detection UDF Image"
--> Using cache
--> 015ad8a734ae
Step 5/8 : HEALTHCHECK NONE
--> Using cache
--> f28881f52000
Step 6/8 : WORKDIR /app
--> Using cache
--> ee87a839f6f0
Step 7/8 : COPY ./weld_porosity_detection ./weld_porosity_detection
--> Using cache
--> f696880a34f4
Step 8/8 : COPY ./weld.mp4 ./test_videos/weld.mp4
--> Using cache
--> 9a1712a96b0b
Successfully built 9a1712a96b0b
Successfully tagged python_weld_detection:2.6
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$ docker-compose up -d
WARNING: Found orphan containers (la_etcd) for this project. If you removed or renamed this service in your compose file, you can run
this command with the --remove-orphans flag to clean it up.
Creating la_visualizer ... done
Creating la_etcd_ui ... done
Creating la_web_visualizer ... done
Creating python_weld_detection ... done
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$
```

Step 2: Check the Reference Implementation

The Edge Insights for Industrial builds the PyWeldPorosityIngestion, Visualizer and WebVisualizer containers. The VideoIngestion container streams the sample weld video and performs inference on each frame to detect porosity defect. Once the frame has been classified, the image is displayed on the Visualizer output window with the inference result and time.

Execute the following command to ensure all the containers are running without errors:

```
sudo docker ps
```

Check for Success

If it was successful, the results will be similar to:

```

Intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build
Intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$ docker ps
CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS
cf07af64154d   openedgeinsights/ia_visualizer:2.6   "./visualizer_start..." About a minute ago Up About a minute (hea
lth: starting)
a041930105cd   python_weld_detection:2.6            "./VideoIngestion/vi..." About a minute ago Up About a minute (hea
lth: starting)
19fd084b1652   openedgeinsights/ia_web_visualizer:2.6   "./web_visualizer_st..." About a minute ago Up About a minute (hea
lth: starting)
a13a16c8c091   openedgeinsights/ia_etcd_ui:2.6         "python3 start_etcdk..." About a minute ago Up About a minute (hea
lth: starting)
d38a1e82ed2a   openedgeinsights/ia_etcd:2.6           "./start_etcd.sh"       3 minutes ago  Up 3 minutes (health:
starting)
Intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/IEdgeInsights/build$

```

Visualizer Output



The sample weld video has 3 types of classification results that can be monitored using the visualizer output: no weld, normal weld, porosity weld.

Step 3: Stop the Vision Pipeline

To stop the Edge Insights for Industrial software, execute the following command:

```
cd $EII_HOME/build/  
sudo sg docker -c 'docker-compose down'
```

To revert to the default PCB Demo application, move the Visualizer and WebVisualizer `config.json` files to the original copies [Step 1.3.a] and rebuild using the instructions below:

```
# Replace the original Visualizer & WebVisualizer config.json  
mv $EII_HOME/Visualizer/config.json_original $EII_HOME/Visualizer/config.json  
mv $EII_HOME/WebVisualizer/config.json_original $EII_HOME/WebVisualizer/config.json  
  
# Configure the IEdgeInsights pipeline to use default containers for PCB demo application  
cd $EII_HOME/build/  
python3 builder.py -f usecases/video-streaming.yml  
  
# Provision  
cd $EII_HOME/build/provision  
sudo -E ./provision.sh ../docker-compose.yml  
  
# Run  
sudo sg docker -c 'docker-compose up -d'  
  
# Enable visualizer display if not already  
xhost +
```

Troubleshooting

- If executing `docker ps` lists all containers as running, but the visualizer window does not display results, execute the below command:

```
xhost +
```

```
intel@edgesoftware: ~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/
File Edit View Search Terminal Help
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/I
access control disabled, clients can connect from any host
intel@edgesoftware:~/edge_insights_industrial/Edge_Insights_for_Industrial_2.6/I
```

This is a one-time command that needs to be executed after each system reboot.

- If the system is remote without access to monitor display, use the WebVisualizer to view the results while the custom UDF containers are running. Login from a Chrome browser using the IP address of the EII system and port 5000: `https://<ipaddress>:5000`

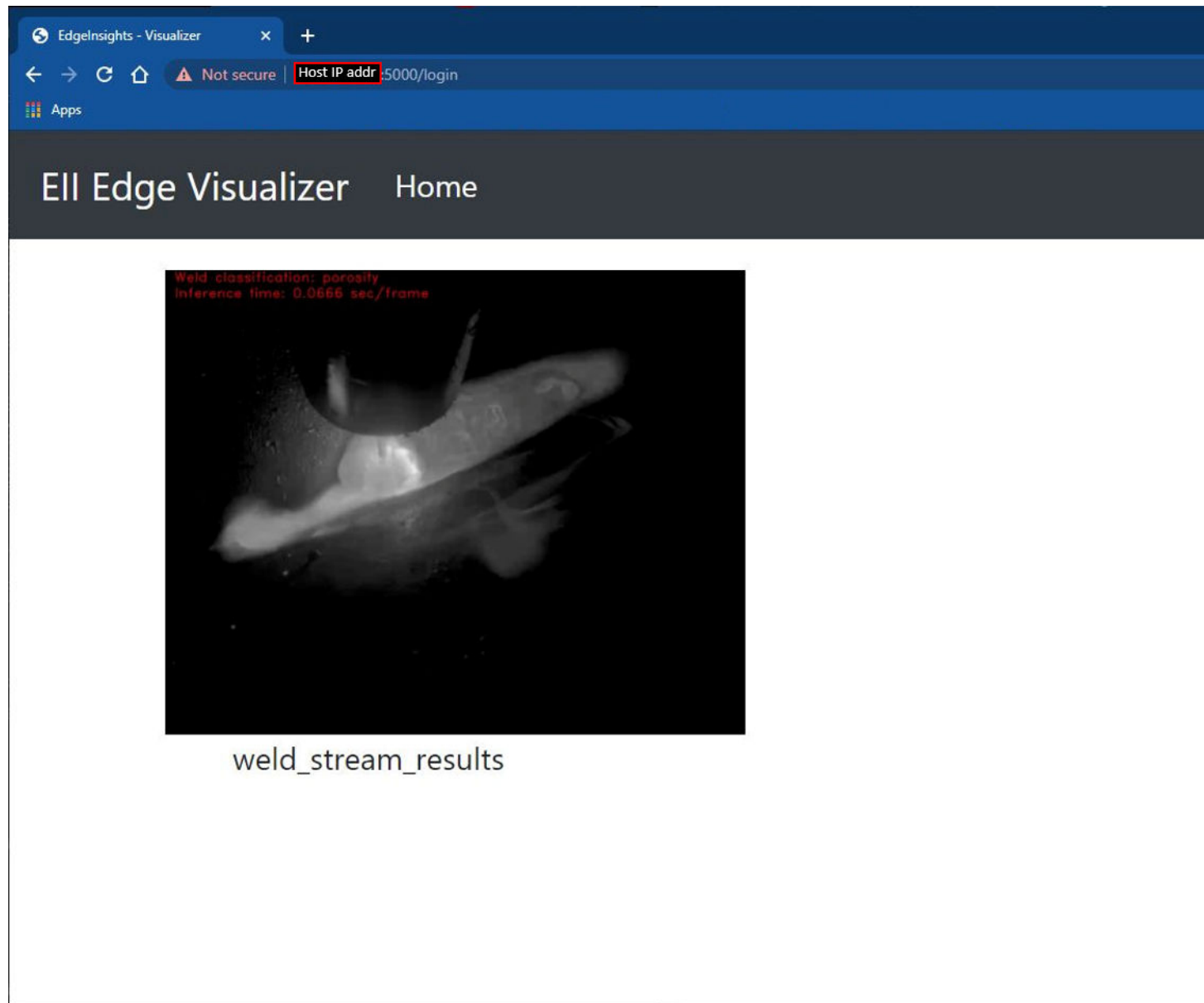
Enter using the default login ID and password provided in `webvis_config.json`.

Username: `admin`

Password: `admin@123`

Optionally, modify these values in `$EII_HOME/WebVisualizer/config.json` while building the containers to use custom username and passwords.

WebVisualizer Output



If you're unable to resolve your issues, go to the [Support Forum](#).

Summary and Next Steps

You successfully ran the Weld Porosity Detection application and displayed the result using the Edge Insights for Industrial Visualizer output.

To get access to the deep learning training algorithm that was used to generate the Weld Porosity Detection model, please contact your Intel account manager.

Release Notes

Edgesoftware CLI Features:

Initial Features for Recommended Configuration

- Installs Docker CE*
- Installs Docker Compose*
- Installs all the prerequisites and dependencies for Edge Insights for Industrial.
- Installs and sets up the Edge Insights for Industrial.
- Provisions Edge Insights for Industrial based on user input 1) Production or 2) Developer mode.
- Supports below use cases:
 - Video Analytics + Time Series
 - Video Analytics
 - Time Series
- Brings up all container images as per the use case selected during Edge Insights for Industrial installation.
- Downloads Manageability module, which can be installed manually with TPM and non-TPM capability.
- Downloads Intel® Training and Learning Suite (TLS) server and TLSRemoteAgent service, which needs to be installed manually and configured respectively.
- Custom flow enabled to allow users to select or deselect different modules as per their choice.

EII 2.6.3 Changes

- <https://open-edge-insights.github.io/pages/release.html#eii-v2-6-3>
- Removed the DOCKER_REGISTRY prefix in TLSRemoteAgent Dockerfile

EII 2.6.2 Changes

- <https://open-edge-insights.github.io/pages/release.html#eii-v2-6-2>

EII 2.6.1 Changes

- <https://open-edge-insights.github.io/pages/release.html#eii-v2-6-1>

EII 2.6 Features

- <https://open-edge-insights.github.io/pages/release.html#eii-v2-6>

Release Content

Subproject (component)	Location	Revision
IEdgeInsights	edge_insights_industrial/ Edge_Insights_for_Industrial 1_2.6.3/IEdgeInsights	v2.6.3
Manageability	edge_insights_industrial/ Edge_Insights_for_Industrial 1_2.6.1/manageability	v2.6.1
Training and Learning Suite	edge_insights_industrial/ Edge_Insights_for_Industrial 1_2.6.3/training-and- learning-suite	v2.6.3

Documentation Archive

View earlier version of Edge Insights for Industrial documentation:

- [2.6.2](#)

NOTE Only the most current version of the documentation is maintained and updated.
