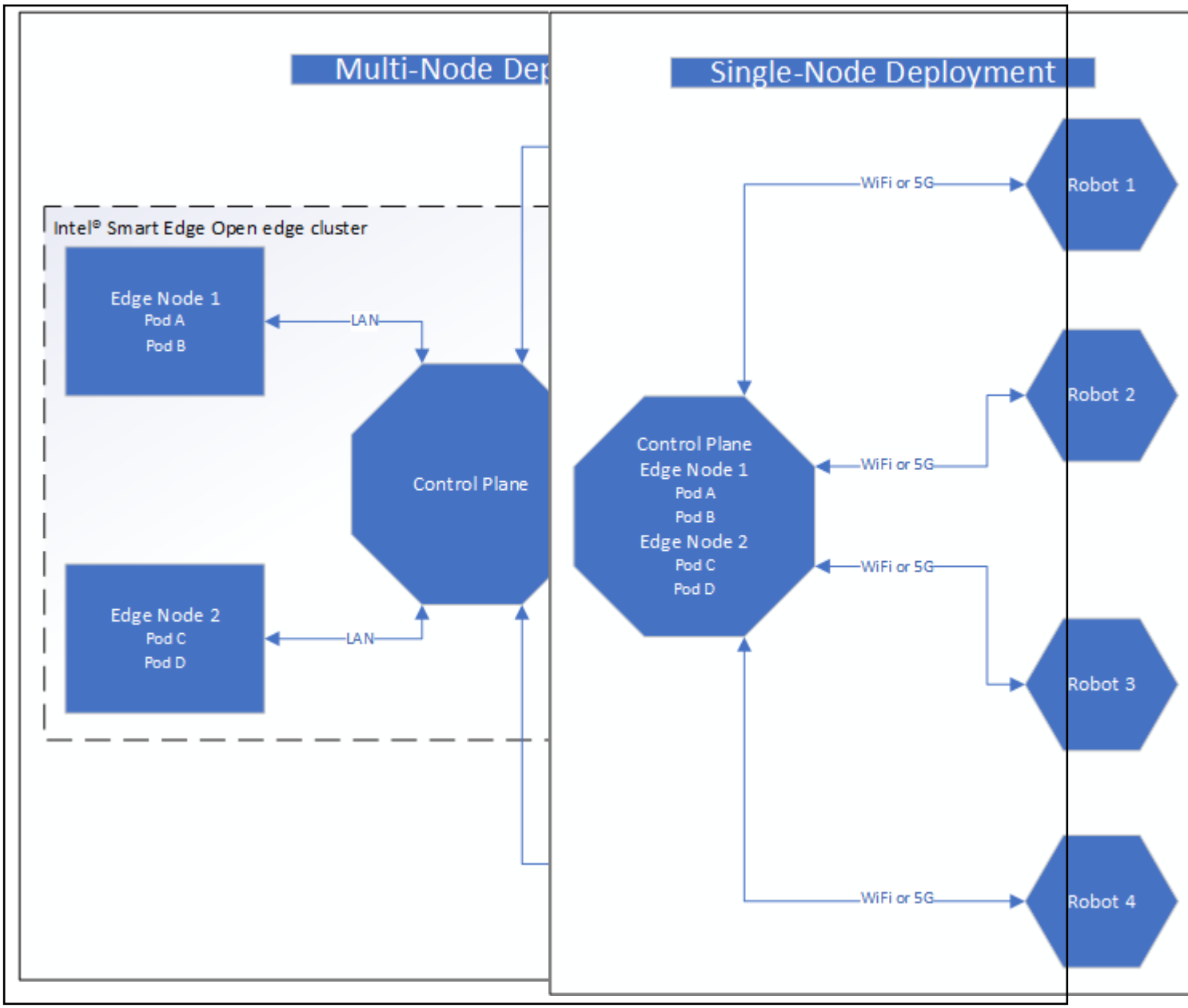
Edge Insights for Autonomous Mobile Robots (EI for AMR) Robot Orchestration Get Started Guide

Contents

Edge Insights for Autonomous Mobile Robots (EI for AMR) Robot Orchestration 1

There are multiple pieces in the robot orchestration setup: the control plane, edge nodes, and robots. The control plane and zero or more edge nodes form an Intel® Smart Edge Open edge cluster.



This Get Started Guide refers to the control plane, edge nodes, and robots in the following ways.

- Control plane: Used to control all edge nodes and robots through Kubernetes*. In a Single-Node deployment, the control plane and edge nodes(s) are installed on the same machine. In a Multi-Node deployment, the control plane and edge nodes(s) are on separate machines.

  - There can only be one control plane.
  - Only the control plane knows about the edge nodes. If a robot wants to offload to an edge node, it asks the control plane, and the control plane routes the request to an edge node.
- Edge node: One or more edge nodes which EI for AMR uses for resource intensive actions like:
  - Remote inference
  - Collaborative SLAM
  - Other actions that help EI for AMR robots perform their purpose efficiently

  From a Kubernetes* perspective, an edge node is an on-premise, stationary worker node.
  - An edge node may be virtual or a physical machine.
  - An edge node can have multiple pods. The control plane handles scheduling the pods across all edge nodes in the cluster.
- Pod: A pod encapsulates one or more application and always runs on a node. From a Kubernetes* perspective, a pod is the smallest execution unit.
- Robot: One or more robots with the EI for AMR Robot Complete Kit or Robot Base Kit installed.
  - From a Kubernetes* perspective, a robot is an on-premise, mobile worker node.
  - Robots are added after the initial configuration using onboarding procedures. Whether the initial deployment is Single-Node or Multi-Node does not affect onboarding procedures.

All devices need to be in the same network. Usually, the control plane and the edge nodes are connected with a LAN connection. The robots are connected to the control plane with a Wi-Fi or 5G connection.

The currently supported versions are:

- Base OS: Ubuntu* 20.04 LTS
- ROS 2 with data distribution service: Foxy
- Intel® Smart Edge Open: 21.12
- ThingsBoard*: 3.3.4.1-CVE22965

Requirements

Knowledge/Experience

- You have some knowledge of container orchestration.
- You are familiar with executing Linux* commands.
- You have basic Docker* experience.
- ROS 1 or ROS 2 background recommended.

Target System for the Server Complete Kit and Robot and Server Complete Kit

- Intel® processors:
  - Intel® Xeon® processor E3, E5, and E7 family
  - 2nd Generation Intel® Xeon® Scalable Processors
  - 3rd Generation Intel® Xeon® Scalable Processors

  NOTE Intel recommends the previously listed Intel® Xeon® processors for any system running resource intensive loads. Resource intensive uses include such things as remote inference and collaborative visual SLAM. If the server is only for fleet management, robot deployment, and robot onboarding, the following Intel® Core™ processors are sufficient.
  - 11th Generation Intel® Core™ processors with Intel® Iris® Xᵉ Integrated Graphics or Intel® UHD Graphics
  - 10th Generation Intel® Core™ processors with an integrated GPU and Intel® UHD Graphics
- 16 GB RAM

- 128 GB hard drive
- Ubuntu* 20.04 LTS

Step 1: Prepare Your Machines

1. Install a clean Ubuntu* OS on the control plane and, for Multi-Node deployments, on all edge nodes. This example uses Ubuntu* Server, but Ubuntu* Desktop is also supported. Here are the high-level steps.
   a. Download the Ubuntu* Server ISO file to your developer workstation for the control plane and for the edge nodes. For the supported version, see Requirements.
   b. Follow the prompts to install Ubuntu* with the default configurations.
   c. For detailed instructions, see the Ubuntu* guide.

   Expected result: Ubuntu* Server is successfully installed.
2. Verify that PATH is configured in `/etc/environment` to contain, at a minimum:

```
PATH='/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin'
```

   If a proxy is required to connect to the Internet:

   a. Add the proxy settings in `/etc/environment`:

```
export http_proxy="http://<http_proxy>:port"
export https_proxy="http://<https_proxy>:port"
export ftp_proxy="http://<ftp_proxy>:port"
export no_proxy="<no_proxy>"
```

   b. Apply the proxy settings:

```
source /etc/environment
```

3. On the control plane and all edge nodes, install the following packages:

```
sudo apt-get update
sudo apt-get install ssh
sudo apt-get install net-tools wget pipenv git ansible
pip install sh
```

4. Configure the control plane with a static IP. This is needed to avoid IP changes during control plane reboots. For help on how to set a static IP, go to Troubleshooting.

   NOTE This step is needed because the deployment reboots the control plane multiple times, and the deployment assumes the control plane has the same IP every time.

5. If the system does not have a host name, set `hostname`:

```
sudo hostname -b <hostname>
```

6. On the control plane and all edge nodes, make sure that password-less `ssh` access for `root` is set.
   a. Edit `/etc/ssh/sshd_config`:

```
sudo nano /etc/ssh/sshd_config
```

   b. Add the following line at the end of the file:

```
PermitRootLogin yes
```

   c. Restart the `ssh` service:

```
sudo service ssh restart
```

7. Log in as `root`. Do all of the remaining actions, after this point, in Step 1 as `root`.

```
sudo su -
```

8. On the control plane and all edge nodes, set a password for `root`.

If `root` does not already have a password, set a password for `root`:

```
passwd
```

9. Generate and add the `ssh` keys on the control plane.

   For a Single-Node Deployment

   a. Send the keys from the control plane to itself. This is needed because Intel® Smart Edge Open uses the `ssh` connection from the control plane to the worker node even though they are on the same machine.

```
ssh-keygen -f $HOME/.ssh/id_rsa -P ""
ssh-copy-id root@<hostname of the control plane>
chmod -R 700 ~/.ssh
```

   b. Update `/etc/hosts`:

```
127.0.0.1       localhost  <enter the system's hostname here>
127.0.1.1       <enter the system's hostname here>

# The following lines are desirable for IPv6 capable hosts
::1             localhost ip6-localhost ip6-loopback <enter the system's hostname here>
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

   For a Multi-Node Deployment

   a. Generate the `ssh` key.

```
ssh-keygen -f $HOME/.ssh/id_rsa -P ""
```

   b. On the control plane and all edge nodes, send the keys from the control plane to all edge nodes and from the control plane to itself:

```
ssh-copy-id root@<hostname of the control plane or edge node>
```

   c. On the control plane and all edge nodes, set the correct permissions for the `~/.ssh` directory:

```
chmod -R 700 ~/.ssh
```

   d. On the control plane and all edge nodes, update `/etc/hosts`:

```
127.0.0.1       localhost  <enter the system's hostname here>
127.0.1.1       <enter the system's hostname here>

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback <enter the system's hostname here>
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

10. Verify that all systems have the same date and time.

```
date
```

    If the date and time are not identical on all systems, synchronize the date and time between the control plane and edge nodes by running this command from the control plane to each edge node:

```
date +%Y%m%d%T -s "`ssh root@<hostname of the edge node> 'date "+%Y%m%d %T"'`"
```

NOTE

- This synchronizes the date and time between systems for this boot.
- A reboot may de-synchronize the systems.
- The systems should be configured to reflect the real world date and time.
- See Ubuntu's time synchronization for detailed instructions.

Step 2: SFTP Server Setup

In this example, the SFTP server sits on the edge node. In other configurations, it can be on any system in the same network as the control plane.

1. Install ssh:

```
sudo apt install ssh
```

2. Edit the `/etc/ssh/sshd_config` file:

```
sudo nano /etc/ssh/sshd_config
```

   Add the following lines at the end of the file:

```
Match group sftp
ChrootDirectory /home
X11Forwarding no
AllowTcpForwarding no
ForceCommand internal-sftp
```

3. Restart the ssh service by running:

```
sudo systemctl restart ssh
```

4. Create group and user:

```
sudo addgroup sftp
sudo useradd -m fdo_sftp -g sftp
sudo passwd fdo_sftp
sudo chmod 0700 /home/fdo_sftp
```

Step 3: Install Intel® Smart Edge Open, Tailored for EI for AMR

Perform all of Step 3 on the control plane only, not the edge nodes.

1. Log in as `root`. Do all steps in Step 3 as `root`.

```
sudo su -
```

2. Clone the `open-developer-experience-kits` repository to the control plane:

```
git clone -b smart-edge-open-21.12 https://github.com/smart-edge-open/open-developer-experience-kits.git ~/dek
cd ~/dek
git checkout 1848a355586d2c40420b6e5576efeac9396150de
```

3. Take note of the control plane and edge node IPs (you need them for the next step):

```
ifconfig
```

4. Edit the `~/dek/inventory.yml` file.
   a. Name your cluster in `cluster_name`; use _ instead of spaces.
   b. Indicate if it is a single or multi-node deployment:
      - **For a Multi-Node Deployment**, set the `single_node_deployment` value to `false`.
      - **For a Single-Node Deployment**, set the `single_node_deployment` value to `true`.
   c. Provide the control plane and edge node IPs.
      - **For a Multi-Node Deployment**, provide an IP for the control plane and each edge node.
      - **For a Single-Node Deployment**, provide the same IP for the control plane and node01. node02 is not required.
   d. Change the `ansible_user` from `smartedge-open` to `root`.

Example:

```
# SPDX-License-Identifier: Apache-2.0
# Copyright (c) 2021 Intel Corporation

---
all:
   vars:
   cluster_name: dek_test        # Use `_` instead of spaces.
   deployment: dek               # Available deployment type: Developer experience kits (dek).
   single_node_deployment: false  # Request a single node deployment (true/false).
   limit:                        # Limit ansible deployment to certain inventory group or hosts
controller_group:
   hosts:
   controller:
      ansible_host: <ip_from_control plane>
      ansible_user: root
edgenode_group:
   hosts:
   node01:
      ansible_host: <ip_from_edge_node01>
      ansible_user: root
   node02:
      ansible_host: <ip_from_edge_node02>
      ansible_user: root
```

5. If a proxy is required to connect to the Internet, edit the proxy variables in the `~/dek/inventory/default/group_vars/all/10-default.yml` file.

Example:

```
# SPDX-License-Identifier: Apache-2.0
# Copyright (c) 2019-2021 Intel Corporation

---
# This file contains variables intended to be configured by user.
# It allows feature enabling and configuration.
# Per-host variables should be places in `inventory/default/host_vars` directory.
# Features should not be configured by changing roles' defaults (i.e. role/defaults/main.yml)

################################################
##### User settings

### Proxy settings
proxy_env:
   # Proxy URLs to be used for HTTP, HTTPS and FTP
   http_proxy: "http://proxy.example.org:3128"
   https_proxy: "http://proxy.example.org:3129"
   ftp_proxy: "http://proxy.example.org:3128"
   # No proxy setting contains addresses and networks that should not be accessed using proxy
(e.g. local network, Kubernetes* CNI networks)
   no_proxy: "127.0.0.1/32"
```

6. Update the `~/dek/inventory/default/group_vars/all/10-default.yml` file with:

```
sriov_network_operator_enable: false

## SR-IOV Network Operator configuration
sriov_network_operator_configure_enable: false

### Software Guard Extensions
# SGX requires kernel 5.11+, SGX enabled in BIOS and access to PCC service
```

```
sgx_enabled: false

# Install isecl attestation components (TA, ihub, isecl k8s controller and scheduler extension)
platform_attestation_node: false

install_hwe_kernel_enable: false
```

7. Update the `~/dek/roles/telemetry/grafana/templates/prometheus-tls-datasource.yml` file by running the following sed command on a terminal:

```
sed -i "s/indent(width=13, indentfirst=False)/indent(width=13, first=False)/g" ~/dek/roles/
telemetry/grafana/templates/prometheus-tls-datasource.yml
```

8. Start deployment:

```
./deploy.sh
```

   Expected result: The script reboots the control plane.

9. After the reboot, run ./deploy.sh again:

```
sudo su -
cd ~/dek
./deploy.sh
```

   Expected result example:

```
kubernetes/harbor_registry/controlplane ------------------------------ 566.66s
infrastructure/docker ------------------------------------------------ 54.81s
.
.
.
infrastructure/setup_offline ----------------------------------------- 0.04s
stat ----------------------------------------------------------------- 0.03s
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
total ---------------------------------------------------------------- 767.77s
2021-11-05 22:45:45.898 INFO: dek_test single_node_network_edge.yml: succeed.
2021-11-05 22:45:46.899 INFO: ====================
2021-11-05 22:45:46.900 INFO: DEPLOYMENT RECAP:
2021-11-05 22:45:46.900 INFO: ====================
2021-11-05 22:45:46.900 INFO: DEPLOYMENT COUNT: 1
2021-11-05 22:45:46.900 INFO: SUCCESSFUL DEPLOYMENTS: 1
2021-11-05 22:45:46.901 INFO: FAILED DEPLOYMENTS: 0
2021-11-05 22:45:46.901 INFO: DEPLOYMENT "dek_test": SUCCESSFUL
2021-11-05 22:45:46.901 INFO: ====================
```

10. For onboarding, copy Kubernetes* and Docker* certificates to the SFTP server. If the SFTP server is not configured, see Step 2: SFTP Server Setup.

    a. Open an SFTP terminal:

```
sftp fdo_sftp@<sftp_server_ip>
```

    b. Copy the Kubernetes* Intel® Smart Edge Open certificate, Kubernetes* client key, and Kubernetes* client certificate:

```
mkdir /fdo_sftp/pki/
cd /fdo_sftp/pki/
lcd /etc/kubernetes/pki/
put ca.crt
put apiserver-kubelet-client.crt
put apiserver-kubelet-client.key
```

c. Copy the Docker* configuration file:

```
mkdir /fdo_sftp/root/
mkdir /fdo_sftp/root/.docker/
cd /fdo_sftp/root/.docker/
lcd /root/.docker/
put config.json
```

d. Copy the Docker* daemon configuration file:

```
mkdir /fdo_sftp/etc/
mkdir /fdo_sftp/etc/docker/
cd /fdo_sftp/etc/docker/
lcd /etc/docker/
put daemon.json
```

e. Copy the Docker* certificate file:

```
mkdir /fdo_sftp/etc/docker/certs.d/
mkdir /fdo_sftp/etc/docker/certs.d/<control_plane_ip>:30003/
cd /fdo_sftp/etc/docker/certs.d/<control_plane_ip>:30003/
lcd /etc/docker/certs.d/<control_plane_ip>:30003/
put ca.crt
```

f. Copy the Docker* proxy configuration file:

```
mkdir /fdo_sftp/etc/systemd/
mkdir /fdo_sftp/etc/systemd/system/
mkdir /fdo_sftp/etc/systemd/system/docker.service.d/
cd /fdo_sftp/etc/systemd/system/docker.service.d/
lcd /etc/systemd/system/docker.service.d/
put http-proxy.conf
```

g. Exit from SFTP terminal:

```
exit
```

Step 4: Install the EI for AMR Server Complete Kit

Perform all of Step 4 on the control plane only, not the edge nodes.

1. Log in as default user, not `root`.
2. Download the latest release.
   a. Go to Product Download.
   b. Select **Server Complete Kit** or **Robot and Server Complete Kit**.
   c. Click **Download**.
3. Copy the zip file to your target system.

> **NOTE** EI for AMR is delivered as a 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 to be installed, and a configuration file (config.ini).

4. Extract and install the software:

```
unzip edge_insights_for_amr.zip
cd edge_insights_for_amr
chmod 775 edgesoftware
export no_proxy="127.0.0.1/32,devtools.intel.com"
sudo ./edgesoftware install
sudo groupadd docker
```

```
sudo usermod -aG docker $USER
newgrp docker
source /etc/environment
```

5. Log in as `root`. Do all of the remaining actions, after this point, from Step 4 as `root`.

```
sudo su -
```

6. Disable swap:

```
sed -ri '/\sswap\s/s/^#?/#/' /etc/fstab
swapoff -a
```

7. Download zenoh images:

```
docker pull eclipse/zenoh-bridge-dds:0.5.0-beta.9
docker pull eclipse/zenoh:0.5.0-beta.9
```

8. Get all Intel® Smart Edge Open edge nodes:

```
kubectl get nodes
```

9. Delete Intel® Smart Edge Open node labels for all edge nodes:

```
kubectl label nodes <node-name-from-the-previous-step> tier-
kubectl label nodes <node-name-from-the-previous-step> environment-
```

10. Set the node labels:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_*
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/smart_edge_open/seo_cluster_labeling.yaml
```

11. Verify that the labels were correctly updated: the environment is `production`, and the tier is `edge-server`:

```
kubectl get nodes -L environment
kubectl get nodes -L tier
```

12. For onboarding, copy seo_install.sh and k8s_apply_label.py to the SFTP server. If the SFTP server is not configured, see Step 2: SFTP Server Setup.

    a. Open an SFTP terminal:

```
sftp fdo_sftp@<sftp_server_ip>
```

    b. Copy the scripts to the SFTP server:

```
cd /fdo_sftp/
lcd <AMR_Server_Containers>/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/smart_edge_open/
put seo_install.sh
put k8s_apply_label.py
```

    c. Exit the SFTP terminal:

```
exit
```

13. Install the collaborative SLAM DDS router playbook:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/collab_dds_router/collab_dds_router_install.yaml
```

14. Install the collaborative SLAM server playbook:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/collab_slam/collab_slam_install.yaml
```

15. Install the onboarding playbook. There are two supported robots types: `amr-aaeon` and `amr-pengo`.

    - `amr-aaeon`: UP Xtreme i11 Robotic Kit

1. Install the onboarding playbook. The number of `pod_replicas` is up to you. The number of `ovms_host` is the control plane hostname.

```
ansible-playbook  --extra-vars '{ "pod_replicas":5}' --extra-vars '{ "http_proxy":http://
http_proxy:port}' --extra-vars '{ "https_proxy":https://https_proxy:port}' --extra-vars
'{ "ovms_host":<CONTROL_PLANE_IP>}' AMR_server_containers/01_docker_sdk_env/docker_orchestration/
ansible-playbooks/01_amr/onboarding/onboarding_install.yaml
```

2. Verify that services, pods, and deployment are running:

```
$ kubectl get all --output=wide --namespace onboarding
   NAME                                          READY   STATUS    RESTARTS   AGE   IP
NODE     NOMINATED NODE   READINESS GATES
   pod/onboarding-deployment-7fff9ddc47-cqzgd   0/16    Pending   0          18s   <none>
<none>   <none>           <none>
   pod/onboarding-deployment-7fff9ddc47-fgc99   0/16    Pending   0          18s   <none>
<none>   <none>           <none>
   pod/onboarding-deployment-7fff9ddc47-kdn74   0/16    Pending   0          18s   <none>
<none>   <none>           <none>
   pod/onboarding-deployment-7fff9ddc47-s7slz   0/16    Pending   0          18s   <none>
<none>   <none>           <none>
   pod/onboarding-deployment-7fff9ddc47-ttgdg   0/16    Pending   0          18s   <none>
<none>   <none>           <none>

   NAME                         TYPE       CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
SELECTOR
   service/onboarding-service   NodePort   10.108.200.30   <none>        8883:32759/TCP   18s
app.kubernetes.io/instance=onboarding-abcxzy,app.kubernetes.io/name=onboarding

   NAME                                     READY   UP-TO-DATE   AVAILABLE   AGE
CONTAINERS


IMAGES









                                                                                SELECTOR
   deployment.apps/onboarding-deployment   0/5     5            0           18s   dds-bridge,amr-
fleet-management,vda5050-ros2-bridge,amr-realsense,amr-ros-base-camera-tf,amr-aaeon-amr-
interface,amr-ros-base-teleop,amr-battery-bridge,amr-object-detection,imu-madgwick-filter,robot-
localization,amr-collab-slam,amr-fastmapping,amr-nav2,amr-wandering,amr-vda-navigator
10.237.22.198:30003/intel/eclipse/zenoh-bridge-dds:0.5.0-beta.9,10.237.22.198:30003/intel/amr-
fleet-management::0.5.0-beta.9,10.237.22.198:30003/intel/amr-vda5050-ros2-
bridge:2022.3,10.237.22.198:30003/intel/amr-realsense:2022.3,10.237.22.198:30003/intel/amr-ros-
base-camera-tf:2022.3,10.237.22.198:30003/intel/amr-aaeon-amr-
interface:2022.3,10.237.22.198:30003/intel/amr-ros-base-teleop:2022.3,10.237.22.198:30003/intel/
amr-battery-bridge:2022.3,10.237.22.198:30003/intel/amr-object-
detection:2022.3,10.237.22.198:30003/intel/amr-imu-madgwick-filter:2022.3,10.237.22.198:30003/
intel/amr-robot-localization:2022.3,10.237.22.198:30003/intel/amr-collab-
slam:2022.3,10.237.22.198:30003/intel/amr-fastmapping:2022.3,10.237.22.198:30003/intel/amr-
nav2:2022.3,10.237.22.198:30003/intel/amr-wandering:2022.3,10.237.22.198:30003/intel/amr-vda-
navigator:2022.3   app.kubernetes.io/instance=onboarding-abcxzy,app.kubernetes.io/name=onboarding

   NAME                                           DESIRED   CURRENT   READY   AGE
CONTAINERS
```

```
IMAGES







                                                                              SELECTOR
   replicaset.apps/onboarding-deployment-7fff9ddc47   5         5         0       18s   dds-
bridge,amr-fleet-management,vda5050-ros2-bridge,amr-realsense,amr-ros-base-camera-tf,amr-aaeon-
amr-interface,amr-ros-base-teleop,amr-battery-bridge,amr-object-detection,imu-madgwick-
filter,robot-localization,amr-collab-slam,amr-fastmapping,amr-nav2,amr-wandering,amr-vda-
navigator   10.237.22.198:30003/intel/eclipse/zenoh-bridge-dds:0.5.0-beta.9,10.237.22.198:30003/
intel/amr-fleet-management::0.5.0-beta.9,10.237.22.198:30003/intel/amr-vda5050-ros2-
bridge:2022.3,10.237.22.198:30003/intel/amr-realsense:2022.3,10.237.22.198:30003/intel/amr-ros-
base-camera-tf:2022.3,10.237.22.198:30003/intel/amr-aaeon-amr-
interface:2022.3,10.237.22.198:30003/intel/amr-ros-base-teleop:2022.3,10.237.22.198:30003/intel/
amr-battery-bridge:2022.3,10.237.22.198:30003/intel/amr-object-
detection:2022.3,10.237.22.198:30003/intel/amr-imu-madgwick-filter:2022.3,10.237.22.198:30003/
intel/amr-robot-localization:2022.3,10.237.22.198:30003/intel/amr-collab-
slam:2022.3,10.237.22.198:30003/intel/amr-fastmapping:2022.3,10.237.22.198:30003/intel/amr-
nav2:2022.3,10.237.22.198:30003/intel/amr-wandering:2022.3,10.237.22.198:30003/intel/amr-vda-
navigator:2022.3   app.kubernetes.io/instance=onboarding-abcxzy,app.kubernetes.io/
name=onboarding,pod-template-hash=7fff9ddc47
```

- `amr-pengo`: Pengo

    1. Install the onboarding-pengo playbook. The number of `pod_replicas` is up to you. The number of `ovms_host` is the control plane hostname.

```
ansible-playbook  --extra-vars '{ "pod_replicas":5}' --extra-vars '{ "http_proxy":http://
http_proxy:port}' --extra-vars '{ "https_proxy":https://https_proxy:port}' --extra-vars
'{ "ovms_host":<CONTROL_PLANE_IP>}' AMR_server_containers/01_docker_sdk_env/docker_orchestration/
ansible-playbooks/01_amr/onboarding_pengo/onboarding_pengo_install.yaml
```

    2. Verify that services, pods, and deployment are running:

```
$ kubectl get all --output=wide --namespace onboarding-pengo
   NAME                                         READY   STATUS    RESTARTS   AGE   IP
NODE     NOMINATED NODE   READINESS GATES
   NAME                                               READY   STATUS        RESTARTS
AGE     IP              NODE              NOMINATED NODE   READINESS GATES
   pod/onboarding-pengo-deployment-557b496cf8-9j27c   0/10    Pending       0
5h21m   <none>          <none>            <none>           <none>
   pod/onboarding-pengo-deployment-557b496cf8-lfsgb   0/10    Pending       0
3h49m   <none>          <none>            <none>           <none>
   pod/onboarding-pengo-deployment-557b496cf8-ndwgs   0/10    Pending       0
5h21m   <none>          <none>            <none>           <none>
   pod/onboarding-pengo-deployment-557b496cf8-njppn   0/10    Pending       0
5h21m   <none>          <none>            <none>           <none>
   pod/onboarding-pengo-deployment-557b496cf8-pxz29   0/10    Pending       0
5h21m   <none>          <none>            <none>           <none>
   pod/onboarding-pengo-deployment-557b496cf8-twtlv   0/10    Pending       0
5h21m   <none>          <none>            <none>           <none>

   NAME                             TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE     SELECTOR
   service/onboarding-pengo-service   NodePort   10.103.247.202   <none>        8883:32761/TCP
5h21m   app.kubernetes.io/instance=onboarding-pengo-abcxzy,app.kubernetes.io/name=onboarding-
```

```
pengo

   NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
CONTAINERS

IMAGES




                            SELECTOR
   deployment.apps/onboarding-pengo-deployment   0/5     5            0           5h21m   dds-
bridge,amr-fleet-management,vda5050-ros2-bridge,amr-realsense,amr-kobuki,amr-ros-base-camera-
tf,amr-collab-slam,amr-nav2,amr-wandering,amr-vda-navigator   10.237.22.198:30003/intel/eclipse/
zenoh-bridge-dds:0.5.0-beta.9,10.237.22.198:30003/intel/amr-fleet-
management:2022.3,10.237.22.198:30003/intel/amr-vda5050-ros2-bridge:2022.3,10.237.22.198:30003/
intel/amr-realsense:2022.3,10.237.22.198:30003/intel/amr-kobuki:2022.3,10.237.22.198:30003/intel/
amr-ros-base-camera-tf:2022.3,10.237.22.198:30003/intel/amr-collab-
slam:2022.3,10.237.22.198:30003/intel/amr-nav2:2022.3,10.237.22.198:30003/intel/amr-
wandering:2022.3,10.237.22.198:30003/intel/amr-vda-navigator:2022.3   app.kubernetes.io/
instance=onboarding-pengo-abcxzy,app.kubernetes.io/name=onboarding-pengo

   NAME                                                DESIRED   CURRENT   READY   AGE
CONTAINERS

IMAGES




                            SELECTOR
   replicaset.apps/onboarding-pengo-deployment-557b496cf8   5         5         0       5h21m
dds-bridge,amr-fleet-management,vda5050-ros2-bridge,amr-realsense,amr-kobuki,amr-ros-base-camera-
tf,amr-collab-slam,amr-nav2,amr-wandering,amr-vda-navigator   10.237.22.198:30003/intel/eclipse/
zenoh-bridge-dds:0.5.0-beta.9,10.237.22.198:30003/intel/amr-fleet-
management:2022.3,10.237.22.198:30003/intel/amr-vda5050-ros2-bridge:2022.3,10.237.22.198:30003/
intel/amr-realsense:2022.3,10.237.22.198:30003/intel/amr-kobuki:2022.3,10.237.22.198:30003/intel/
amr-ros-base-camera-tf:2022.3,10.237.22.198:30003/intel/amr-collab-
slam:2022.3,10.237.22.198:30003/intel/amr-nav2:2022.3,10.237.22.198:30003/intel/amr-
wandering:2022.3,10.237.22.198:30003/intel/amr-vda-navigator:2022.3   app.kubernetes.io/
instance=onboarding-pengo-abcxzy,app.kubernetes.io/name=onboarding-pengo,pod-template-
hash=557b496cf8
```

NOTE These steps prepare the control plane to deploy and control edge nodes and robots. After the device is onboarded, the pods are installed on the EI for AMR device.

Step 5: Install the ThingsBoard* Reference

This ThingsBoard* reference includes a pre-configured database customized for Intel's EI for AMR solution.

Configure ThingsBoard*

Do all steps on the **control plane**.

1. Log in as `root`. Do all steps as `root`.

```
sudo su -
```

2. Import the repository signing key:

```
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -
```

3. Add the repository contents to your system:

```
RELEASE=$(lsb_release -cs)
echo "deb http://apt.postgresql.org/pub/repos/apt/ ${RELEASE}"-pgdg main | sudo tee  /etc/apt/
sources.list.d/pgdg.list
```

4. Install and launch the postgresql service:

```
apt update
apt -y install postgresql-12
service postgresql start
```

5. Set the password for postgres:

```
su - postgres
psql
\password
```

   You are asked to set a password here, set `postgres`.

```
\q
exit
```

6. Create the .mytb-data and .mytb-logs directories:

```
mkdir -p ~/.mytb-data && chown -R 799:799 ~/.mytb-data
mkdir -p ~/.mytb-logs && chown -R 799:799 ~/.mytb-logs
```

7. Generate a key pair for ThingsBoard* to EI for AMR robot communications. Run script using the **control plane** server hostname and IP as input parameters:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_<version>/
AMR_server_containers
chmod +x 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fleet_management/thingsboard/tb-
key-gen.sh
./01_docker_sdk_env/artifacts/02_edge_server/edge_server_fleet_management/thingsboard/tb-key-
gen.sh <TB-Hostname> <TB-IP>
```

8. Copy the keys to the SFTP server. If the SFTP server is not configured, see Step 2: SFTP Server Setup.

```
sftp fdo_sftp@<sftp_server_ip>
cd /fdo_sftp/
lcd <AMR_Server_Containers>/01_docker_sdk_env/artifacts/02_edge_server/
edge_server_fleet_management/thingsboard/
put thingsboard.pub.pem
exit
```

Install the ThingsBoard* Reference

On the **control plane**, build the ThingsBoard* and install the playbook:

```
source ./01_docker_sdk_env/docker_compose/common/docker_compose.source
export DISPLAY=0:0
chmod 775 -R 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fleet_management/
preconfigured_tb_server_database/db
chmod +x 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fleet_management/thingsboard/tb-
server-reset-db.sh
docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml build
```

```
fleet-management
ansible-playbook 01_docker_sdk_env/docker_orchestration/ansible-playbooks/02_edge_server/
fleet_management/fleet_management_playbook_install.yaml
```

If the playbook fails to install, see Troubleshooting.

Check the ThingsBoard* Reference Installation

1. On the **control plane**, verify that the services, pods, and deployment are running:

```
$ kubectl get all --output=wide --namespace fleet-management
NAME                                        READY   STATUS    RESTARTS      AGE
pod/fleet-deployment-8449fdc54f-m4fhb       1/1     Running   2 (21s ago)   81s

NAME                                        TYPE       CLUSTER-IP      EXTERNAL-IP
PORT(S)                                                    AGE
service/fleet-service                       NodePort   10.97.216.230   <none>
9090:32764/TCP,1883:32765/TCP,7070:32766/TCP,8883:32767/TCP   42s

NAME                                        READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/fleet-deployment            1/1     1            1           81s

NAME                                        DESIRED   CURRENT   READY   AGE
replicaset.apps/fleet-deployment-8449fdc54f   1         1         1       81s
```

   If they are not running, see Troubleshooting.
2. On each **edge node**, verify that the Docker* container is running:

```
docker ps | grep fleet
dd22be830f82   10.237.23.152:30003/intel/fleet-management              "/usr/bin/start-tb.sh"
52 minutes ago   Up 52 minutes             k8s_fleet_fleet-deployment-858494f866-7jmhh_fleet-
management_13d09334-4223-4409-8cd9-c0cac60cd04c_0
```

   If the container was not deployed, see Troubleshooting.

Open the ThingsBoard* Reference

1. ThingsBoard* runs on the control plane, but it can be accessed from any machine on the same network by:

```
# Open Firefox and go to:
<control plane IP Address>:32764
```

> **Warning** Installing VNC on the control plane node breaks the Intel® Smart Edge Open installation. Intel recommends that you access ThingsBoard* on the control plane without VNC or from any other system in the same network.

   Expected result: A ThingsBoard* login interface appears.

> **Warning** If the ThingsBoard* login interface does not appear, set your browser settings to `No Proxy`. If the ThingsBoard* login interface still does not appear, see Troubleshooting.

Troubleshooting

- If the playbook does not install, or the services, pods, or deployment are not running, first check if the installation was successful even if ansible-playbook failed.

If the installation was unsuccessful, uninstall and reinstall the playbook.

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_uninstall.yaml
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_install.yaml
```

If the ThingsBoard* login interface does not appear, check if your network blocks certain ports. ThingsBoard* uses port 9090. Intel® Smart Edge Open maps this port on the control plane to 32764.

To get more details on what ports and IPs are used:

```
kubectl describe node | grep 'Addresses:' -A 4 | grep -B1 $(kubectl get nodes | grep control-
plane | awk '{print $1}') | grep InternalIP | awk '{print $2}'
```

For detailed EI for AMR target installation steps, see the Get Started Guide for Robots.

- To perform the switch to the preconfigured database, use the following commands:

```
docker exec -it <container-id-of-edge-fleet-management> bash
./tb-server-reset-db.sh
exit
docker restart <container-id-of-edge-fleet-management>
```

- The container might restart several times. If is restarting more than 10 minutes continuously, uninstall and reinstall the playbook.

  If the error persist, recreate the two database directories from the above steps.

```
sudo su -
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_uninstall.yaml
service postgresql restart
rm -rf ~/.mytb-data && mkdir -p ~/.mytb-data && chown -R 799:799 ~/.mytb-data
rm -rf ~/.mytb-logs && mkdir -p ~/.mytb-logs && chown -R 799:799 ~/.mytb-logs
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_install.yaml
```

Step 6: OpenVINO™ Model Server Setup

Configure the OpenVINO™ Model Server

Perform configuration steps on the **control plane**.

1. Go to the `AMR_containers` folder:

```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_*/
AMR_server_containers
```

2. Generate the keys used for the OpenVINO™ model server remote inference setups:

```
chmod +x 01_docker_sdk_env/artifacts/02_edge_server/edge_server_fleet_management/ovms/
generate_ovms_certs.sh
sudo ./01_docker_sdk_env/artifacts/02_edge_server/edge_server_fleet_management/ovms/
generate_ovms_certs.sh <control_plane_IP>
```

> **NOTE** The IP of the control plane is used to generate certificates.

3. Copy the keys to the SFTP server. If the SFTP server is not configured, see Step 2: SFTP Server Setup.

a. Open an SFTP terminal:

```
sftp fdo_sftp@<sftp_server_ip>
```

b. Copy the OpenVINO™ model server certificates to the SFTP server:

```
mkdir /fdo_sftp/etc/amr/
mkdir /fdo_sftp/etc/amr/ri-certs/
cd /fdo_sftp/etc/amr/ri-certs/
lcd <AMR_Server_Containers>/01_docker_sdk_env/artifacts/02_edge_server/
edge_server_fleet_management/ovms/keys/
put server.pem
put client.key
put client.pem
```

c. Exit the SFTP terminal:

```
exit
```

Start the OpenVINO™ Model Server

On the **control plane**, install the playbook:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/openvino_model_server/ovms_playbook_install.yaml
```

If the playbook fails to install, see Troubleshooting.

Check the OpenVINO™ Model Server Installation

1. On the **control plane**, verify that the services, pods, and deployment are running:

```
$ kubectl get all --output=wide --namespace ovms-tls

 NAME                                        READY   STATUS    RESTARTS      AGE
IP             NODE            NOMINATED NODE   READINESS GATES
 pod/ovms-deployment-75c7dffdc5-cgxp9        1/1     Running   1 (50m ago)   50m
10.245.202.9   glaic3roscube   <none>           <none>

 NAME                                        TYPE       CLUSTER-IP       EXTERNAL-IP
PORT(S)                         AGE   SELECTOR
 service/ovms-service                        NodePort   10.110.177.249   <none>
3335:32762/TCP,2225:32763/TCP   50m   app.kubernetes.io/instance=ovms-tls-
abcxzy,app.kubernetes.io/name=ovms-tls

 NAME                                        READY   UP-TO-DATE   AVAILABLE   AGE
CONTAINERS   IMAGES                                    SELECTOR
 deployment.apps/ovms-deployment             1/1     1            1           50m   ovms-
tls     10.237.23.153:30003/intel/ovms-tls:2022.2   app.kubernetes.io/instance=ovms-tls-
abcxzy,app.kubernetes.io/name=ovms-tls

 NAME                                        DESIRED   CURRENT   READY   AGE   CONTAINERS
IMAGES                                    SELECTOR
 replicaset.apps/ovms-deployment-75c7dffdc5   1         1         1       50m   ovms-tls
10.237.23.153:30003/intel/ovms-tls:2022.2   app.kubernetes.io/instance=ovms-tls-
abcxzy,app.kubernetes.io/name=ovms-tls,pod-template-hash=75c7dffdc5
```

NOTE CLUSTER-IP is a virtual IP that is allocated by Kubernetes* to a service. It is the Kubernetes* internal IP. Two different pods can communicate using this IP.

2. On each **edge node**, verify that the Docker* container is running:

```
docker ps | grep ovms-tls
6b64514a4b9a   c9e7db04fe06                       "/usr/bin/dumb-init …"   10 minutes ago   Up
10 minutes             k8s_ovms-tls_ovms-deployment-5856948447-t78tz_ovms-tls_3cd84b35-
c604-4948-9228-e381fd0714fa_1
ceeaa7673bcd   k8s.gcr.io/pause:3.5               "/pause"                 10 minutes ago   Up
10 minutes             k8s_POD_ovms-deployment-5856948447-t78tz_ovms-tls_3cd84b35-c604-4948-9228-
e381fd0714fa_1
```

Step 7: Device Onboarding Control Plane Setup

These steps are used to configure ThingsBoard* for device onboarding. Do all steps on the **control plane**.

Install the MQTT Service

1. Make sure that the common name is the hostname of **control plane**.
2. Install the Eclipse Mosquitto* broker and client for device onboarding and application over-the-air (AOTA) message queuing telemetry transport (MQTT) messages:

```
sudo apt-add-repository ppa:mosquitto-dev/mosquitto-ppa
sudo apt-get update
sudo apt-get install mosquitto
sudo apt-get install mosquitto-clients
sudo apt clean
```

3. Generate the certificates for the Mosquitto secure sockets layer (SSL) and the `server.key` for the Mosquitto SSL:

```
cd  /etc/mosquitto/certs
openssl genrsa -des3 -out ca.key 2048
openssl req -new -x509 -days 1826 -key ca.key -out ca.crt
openssl genrsa -out server.key 2048
openssl req -new -out server.csr -key server.key
```

```
openssl x509 -req -in server.csr -CA ca.crt -CAkey ca.key -CAcreateserial -out server.crt -days
360
chmod 777 *
```

NOTE Use the machine hostname of **control plane** as the common name.

4. Update `/etc/mosquitto/mosquitto.conf`:

```
# Place your local configuration in /etc/mosquitto/conf.d/
#
# A full description of the configuration file is at
# /usr/share/doc/mosquitto/examples/mosquitto.conf.example

persistence true
persistence_location /var/lib/mosquitto/

log_dest file /var/log/mosquitto/mosquitto.log

include_dir /etc/mosquitto/conf.d

listener 18883
allow_anonymous true
```

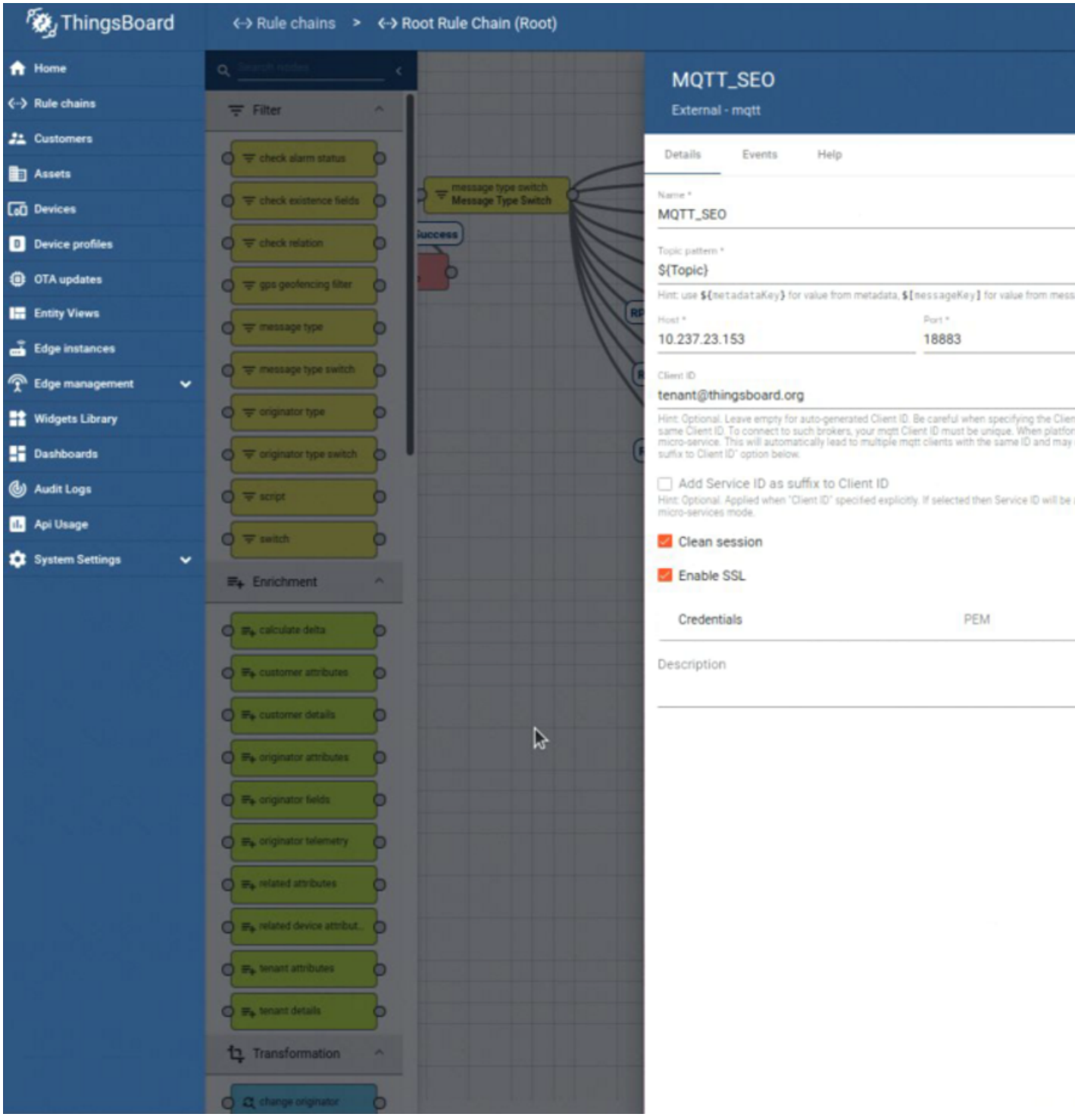
```
cafile /etc/mosquitto/certs/ca.crt
certfile /etc/mosquitto/certs/server.crt
keyfile /etc/mosquitto/certs/server.key
```

5. Go to AMR_server_containers folder, and start the MQTT service:

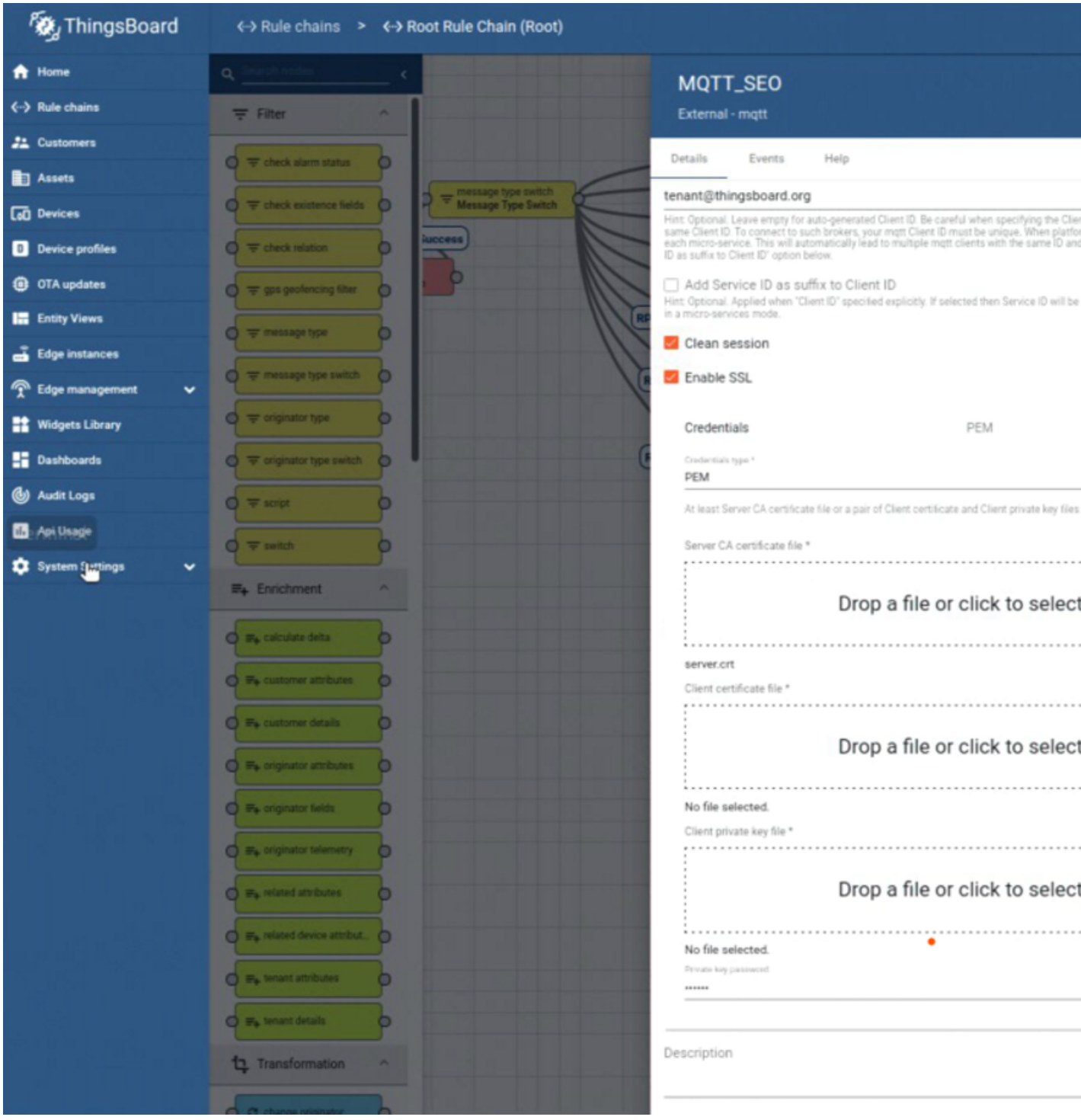
```
cd <edge_insights_for_amr_path>/Edge_Insights_for_Autonomous_Mobile_Robots_*/
AMR_server_containers
ansible-playbook 01_docker_sdk_env/docker_orchestration/ansible-playbooks/02_edge_server/
smart_edge_open/fleetmanagement_interface.yaml
ufw allow 18883
```

   For errors, go to Troubleshooting.

6. Open a browser, use the controller IP, and open <IP Address>:32764. Use the following credentials:
   - account: `tenant@thingsboard.org`
   - password: `tenant`
7. Go to the **Rule Chain** page, and select `MQTT_SEO`.



8. Assign the **control plane** IP to the variable `Host*`.
9. Select the `Enable SSL` option.
10. Assign `PEM` to the variable `Credentials`.
11. Upload the `/etc/mosquitto/certs/server.crt` certificate that was generated above, and apply the changes.



Prepare ThingsBoard* for OTA Updates

1. Prepare for the Intel® RealSense™ camera firmware update.
   a. Download the firmware from https://dev.intelrealsense.com/docs/firmware-releases.
   b. Place the `.bin` file that contains the firmware in a `.tar.gz` archive. Make sure that you do not archive the entire directory, only the `.bin` file.
   c. Set up a basic HTTP server, and upload the `.tar.gz` on it as a trusted repository server:

a. Install the apache2:

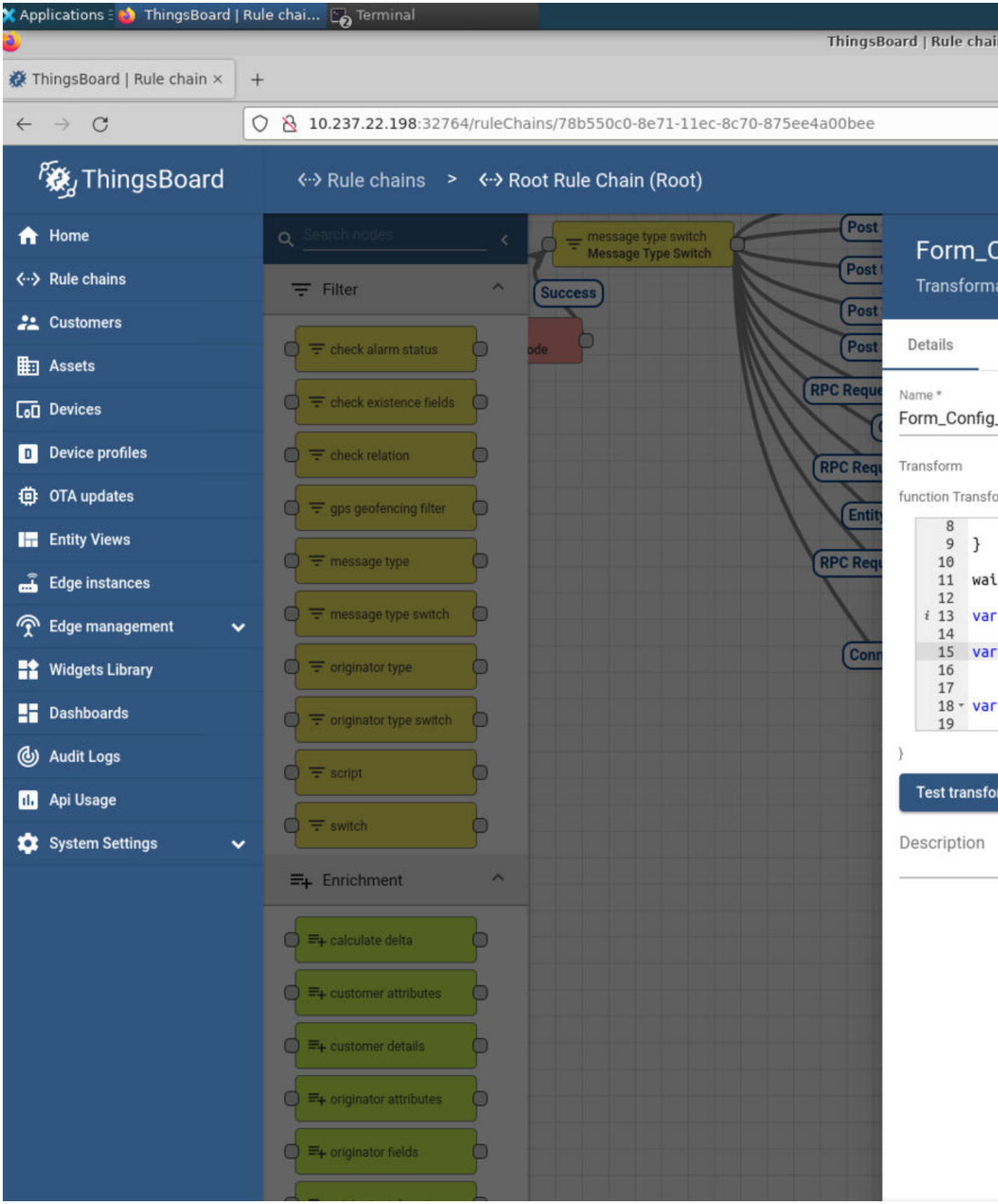
```
sudo apt update
sudo apt install apache2
```

b. Put the RealSense .bin file inside a .tar.gz, and place it on a http server:

```
tar -czvf rs_12_15.tar.gz Signed_Image_UVC_5_12_15_50.bin
sudo cp rs_12_15.tar.gz /var/www/html/
```

2. On ThingsBoard*, open **Rule Chain**.

3. Open **Form_Config_Update**, and, on line 15, update the URL of HTTP host that has the new firmware.



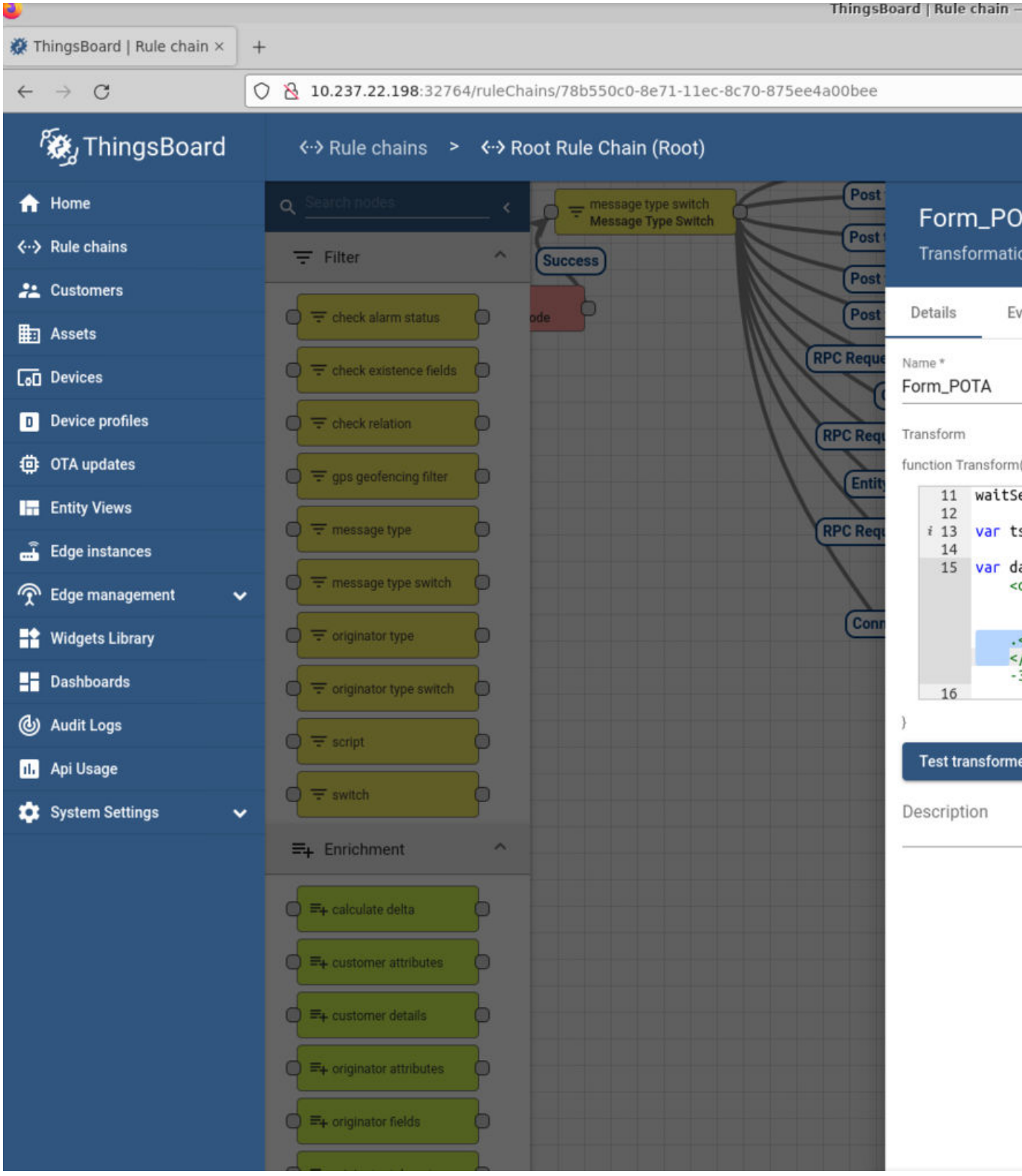
4. Open **Form_POTA**, and, on line 15, update the following.

a. The entire HTTP URL with the .tar.gz file for the firmware file.

NOTE The link should be similar to `http://<hostname>/<archive.tar.gz>`

b. The Manufacturer, Vendor, and the Product name with the output of the following commands. Execute these commands on the robot.

```
dmidecode -t system  | grep Product
dmidecode -t system  | grep Manufacturer
dmidecode -t bios  | grep Vendor
```


ThingsBoard | Rule chain
10.237.22.198:32764/ruleChains/78b550c0-8e71-11ec-8c70-875ee4a00bee
ThingsBoard
Rule chains > Root Rule Chain (Root)
Home
Rule chains
Customers
Assets
Devices
Device profiles
OTA updates
Entity Views
Edge instances
Edge management
Widgets Library
Dashboards
Audit Logs
Api Usage
System Settings
Search nodes
Filter
check alarm status
check existence fields
check relation
gps geofencing filter
message type
message type switch
originator type
originator type switch
script
switch
Enrichment
calculate delta
customer attributes
customer details
originator attributes
originator fields
message type switch
Message Type Switch
Success
Post
RPC Reque
RPC Requ
Entity
RPC Requ
Conn
Form_PO
Transformati
Details
Name *
Form_POTA
Transform
function Transform(
11 waitSe
12
13 var ts
14
15 var da
16
}
Test transforme
Description

NOTE Updating the Manufacturer, Vendor, and Product name needs to be done every time you onboard a new type of robot. If these values do not match the ones from the robot trying to onboard, the flow fails.

5. Save all changes.

If you encounter errors, see Troubleshooting.

Troubleshooting

If, even after reinstalling the playbook and verifying that the ports are not blocked by the firewall, ThingsBoard* does not work, contact Intel's Support Forum.

- Verify that the MQTT service is running:

```
systemctl status mosquitto.service
```

If the command above returns `Active: failed`:

```
chmod -R 755 /etc/mosquitto/
systemctl restart mosquitto.service
systemctl status mosquitto.service
systemctl restart mqtt_aota.service
systemctl status mqtt_aota.service
```

Expected result: The status of the mosquitto service is `Active: active`.

Deepen Your Knowledge

Robot orchestration with the EI for AMR Server Complete Kit is handled by a combination of EI for AMR, Intel® Smart Edge Open, and ThingsBoard*.

Collateral	Description
Get Started Guide for Robots	Robot Kit installation
Developer Guide	Complex scenarios, advanced features, and debugging
Release Notes	New features and known issues
Troubleshooting	Problems installing the software
Intel® Smart Edge Open	Intel® Smart Edge Open is built on top of Kubernetes*, a production-grade platform for managing containerized workloads and services. Intel® Smart Edge Open experience kits customize and extend the Kubernetes* control plane and edge node with microservices, third-party applications, extensions, and optimizations.
ThingsBoard*	ThingsBoard* is an open-source IoT platform for data collection, processing, visualization, and device management. This EI for AMR robot orchestration uses ThingsBoard* as the front-end for Kubernetes*, providing a graphical interface for the orchestration.

Troubleshooting

Setting a Static IP

Depending on your network setup, there are multiple ways to set a static IP.

- In a home network, see your router's instructions for how to set a static IP using your MAC address.
- In a corporate network, contact your local support team for how to set a static IP.
- To set it from your computer's operating system:

  1. Make sure that your computer has the correct date:

```
date
```

     If the date is incorrect, contact your local support team for help setting the correct date and time.
  2. Find the gateway:

```
ip route | grep default
```

  3. Name the servers by finding your interface name and replacing it:

```
nmcli device show <interface name> | grep IP4.DNS
```

  4. Follow the "Static IP Address Assignment" steps from Ubuntu* here.

virtualenv Error

If the following error is displayed:

```
Virtualenv location:
Warning: There was an unexpected error while activating your virtualenv. Continuing anyway…
Traceback (most recent call last):
File "./deploy.py", line 24, in <module>
from scripts import log_all
ImportError: cannot import name 'log_all' from 'scripts' (/home/test/.local/lib/python3.8/site-
packages/scripts/__init__.py)
```

Remove the `~/.local/lib/python3.8/` directory and run the following commands:

```
pip install --user -U pip
pip freeze --user | cut -d'=' -f1 | xargs pip install --user -U
```

Python Error

If the following error is displayed:

```
Failed to install wget. b'    ERROR: Command errored out with exit status 1:\n
command: /usr/bin/python3 -c \'import sys, setuptools, tokenize; sys.argv[0] = \'"\'"\'/tmp/pip-
install-6hcmet6a/wget/setup.py\'"\'"\'; __file__=\'"\'"\'/tmp/pip-install-6hcmet6a/wget/setup.py
\'"\'"\';f=getattr(tokenize, \'"\'"\'open\'"\'"\', open)
(__file__);code=f.read().replace(\'"\'"\'\\r\\n\'"\'"\', \'"\'"\'\\n
\'"\'"\');f.close();exec(compile(code, __file__, \'"\'"\'exec\'"\'"\'))\' egg_info --egg-
base /tmp/pip-pip-egg-info-7_3nl4xa\n         cwd: /tmp/pip-install-6hcmet6a/wget/\n    Complete
output (17 lines):\n    Traceback (most recent call last):\n      File "<string>", line 1, in
<module>\n      File "/tmp/pip-install-6hcmet6a/wget/setup.py", line 15, in <module>\n
setup(\n      File "/usr/local/lib/python3.8/dist-packages/setuptools/_distutils/core.py", line
147, in setup\n        _setup_distribution = dist = klass(attrs)\n      File "/usr/local/lib/
python3.8/dist-packages/setuptools/dist.py", line 476, in __init__\n
_Distribution.__init__(\n      File "/usr/local/lib/python3.8/dist-packages/setuptools/
_distutils/dist.py", line 280, in __init__\n        self.finalize_options()\n      File "/usr/
local/lib/python3.8/dist-packages/setuptools/dist.py", line 899, in finalize_options\n
for ep in sorted(loaded, key=by_order):\n      File "/usr/local/lib/python3.8/dist-packages/
setuptools/dist.py", line 898, in <lambda>\n        loaded = map(lambda e: e.load(), filtered)
\n      File "/usr/local/lib/python3.8/dist-packages/setuptools/_vendor/importlib_metadata/
__init__.py", line 196, in load\n        return functools.reduce(getattr, attrs, module)\n
AttributeError: type object \'Distribution\' has no attribute \'_finalize_feature_opts\'\n
```

```
----------------------------------------\nERROR: Command errored out with exit status 1: python
setup.py egg_info Check the logs for full command output.\nWARNING: You are using pip version
20.2.4; however, version 22.2.2 is available.\nYou should consider upgrading via the \'/usr/bin/
python3 -m pip install --upgrade pip\' command.\n'
```

Remove the `~/.local/lib/python3.8/` directory, and run the following commands:

```
python3 -m pip uninstall setuptools
python3 -m pip install testresources
python3 -m pip install launchpadlib
python3 -m pip install setuptools
python3 -m pip install --user -U pip
```

termcolor Error

If the following error is displayed:

```
Failed to install termcolor. b'/usr/local/lib/python3.8/dist-packages/pkg_resources/
__init__.py:122:
```

```
python3 -m pip uninstall setuptools
python3 -m pip install testresources
python3 -m pip install setuptools
```

Failed OpenSSL Download

If the following error is displayed:

```
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (4
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (4
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (3
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (3
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (2
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (2
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (1
retries left).
FAILED - RETRYING: OpenSSL download from https://www.openssl.org/source/openssl-1.1.1i.tar.gz (1
retries left)
```

Run the following commands:

```
wget --directory-prefix=/tmp http://certificates.intel.com/repository/certificates/
IntelSHA2RootChain-Base64.zip

sudo unzip -o /tmp/IntelSHA2RootChain-Base64.zip -d /usr/local/share/ca-certificates/

rm /tmp/IntelSHA2RootChain-Base64.zip

update-ca-certificates
```

"Isecl control plane IP not set" Error

If the following error is displayed:

```
TASK [Check control plane IP]
*********************************************************************************************
*********************************************************************************************
*************
task path: /root/dek/roles/security/isecl/common/tasks/precheck.yml:7
Wednesday 16 February 2022  15:36:34 +0000 (0:00:00.047)       0:00:05.373 ****
fatal: [node01]: FAILED! => {
   "changed": false
}

MSG:

Isecl control plane IP not set!
fatal: [node02]: FAILED! => {
   "changed": false
}

MSG:

Isecl control plane IP not set!
fatal: [controller]: FAILED! => {
   "changed": false
}

MSG:

Isecl control plane IP not set!
```

Update the `~/dek/inventory/default/group_vars/all/10-default.yml` file with:

```
# Install isecl attestation components (TA, ihub, isecl k8s controller and scheduler extension)
platform_attestation_node: false
```

"PCCS IP address not set" Error

If the following error is displayed:

```
TASK [Check PCCS IP address]
*********************************************************************************************
*********************************************************************************************
**************
task path: /root/dek/roles/infrastructure/provision_sgx_enabled_platform/tasks/
param_precheck.yml:7
Wednesday 16 February 2022  15:39:59 +0000 (0:00:00.060)       0:00:05.688 ****
fatal: [node01]: FAILED! => {
   "changed": false
}

MSG:

PCCS IP address not set!
fatal: [node02]: FAILED! => {
   "changed": false
}

MSG:
```

```
PCCS IP address not set!
fatal: [controller]: FAILED! => {
   "changed": false
}

MSG:

PCCS IP address not set!
```

Update the `~/dek/inventory/default/group_vars/all/10-default.yml` file with:

```
### Software Guard Extensions
# SGX requires kernel 5.11+, SGX enabled in BIOS and access to PCC service
sgx_enabled: false
```

"no supported NIC is selected" Error

If the following error is displayed:

```
sriovnetwork.sriovnetwork.openshift.io/sriov-vfio-network-c1p1 unchanged
STDERR:
Error from server (no supported NIC is selected by the nicSelector in CR sriov-netdev-net-c0p0):
error when creating "sriov-netdev-net-c0p0-sriov_network_node_policy.yml": admission webhook
"operator-webhook.sriovnetwork.openshift.io" denied the request: no supported NIC is selected by
the nicSelector in CR sriov-netdev-net-c0p0
Error from server (no supported NIC is selected by the nicSelector in CR sriov-netdev-net-c1p0):
error when creating "sriov-netdev-net-c1p0-sriov_network_node_policy.yml": admission webhook
"operator-webhook.sriovnetwork.openshift.io" denied the request: no supported NIC is selected by
the nicSelector in CR sriov-netdev-net-c1p0
Error from server (no supported NIC is selected by the nicSelector in CR sriov-vfio-pci-net-
c0p1): error when creating "sriov-vfio-pci-net-c0p1-sriov_network_node_policy.yml": admission
webhook "operator-webhook.sriovnetwork.openshift.io" denied the request: no supported NIC is
selected by the nicSelector in CR sriov-vfio-pci-net-c0p1
Error from server (no supported NIC is selected by the nicSelector in CR sriov-vfio-pci-net-
c1p1): error when creating "sriov-vfio-pci-net-c1p1-sriov_network_node_policy.yml": admission
webhook "operator-webhook.sriovnetwork.openshift.io" denied the request: no supported NIC is
selected by the nicSelector in CR sriov-vfio-pci-net-c1p1
```

Update the `~/dek/inventory/default/group_vars/all/10-default.yml` file with:

```
sriov_network_operator_enable: false

## SR-IOV Network Operator configuration
sriov_network_operator_configure_enable: false
```

"Unexpected templating type error"

If the following error is displayed:

```
MSG:
AnsibleError: Unexpected templating type error occurred on (# SPDX-License-Identifier: Apache-2.0
# Copyright (c) 2020 Intel Corporation
apiVersion: v1
kind: ConfigMap
metadata:
   name: grafana-datasources
   namespace: telemetry
   labels:
      grafana_datasource: '1'
```

```
data:
   prometheus-tls.yaml: |-
      apiVersion: 1
      datasources:
      - name: Prometheus-TLS
         access: proxy
         editable: true
         orgId: 1
         type: prometheus
         url: https://prometheus:9099
         withCredentials: true
         isDefault: true
         jsonData:
            tlsAuth: true
            tlsAuthWithCACert: true
         secureJsonData:
            tlsCACert: |
               {{ telemetry_root_ca_cert.stdout | trim | indent(width=13, indentfirst=False) }}
            tlsClientCert: |
               {{ telemetry_grafana_cert.stdout | trim | indent(width=13, indentfirst=False) }}
            tlsClientKey: |
               {{ telemetry_grafana_key.stdout | trim | indent(width=13, indentfirst=False) }}
         version: 1
         editable: false
): do_indent() got an unexpected keyword argument 'indentfirst'
```

Update the `~/dek/roles/telemetry/grafana/templates/prometheus-tls-datasource.yml` file with:

```
-             {{ telemetry_root_ca_cert.stdout | trim | indent(width=13, indentfirst=False) }}
+             {{ telemetry_root_ca_cert.stdout | trim | indent(width=13, first=False) }}
-             {{ telemetry_grafana_cert.stdout | trim | indent(width=13, indentfirst=False) }}
+             {{ telemetry_grafana_cert.stdout | trim | indent(width=13, first=False) }}
-             {{ telemetry_grafana_key.stdout | trim | indent(width=13, indentfirst=False) }}
+             {{ telemetry_grafana_key.stdout | trim | indent(width=13, first=False) }}
```

"Wait till all Harbor resources ready" Message

If the following log is displayed:

```
TASK [kubernetes/cni : Wait till all Harbor resources ready]
*********************************************************************************************
*************************************************************************
task path: /home/user/dek/roles/kubernetes/cni/tasks/main.yml:20
Tuesday 16 November 2021 14:41:58 +0100 (0:00:00.070) 0:04:39.646 ******
FAILED - RETRYING: Wait till all Harbor resources ready (60 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (59 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (58 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (57 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (56 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (55 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (54 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (53 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (52 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (51 retries left).
FAILED - RETRYING: Wait till all Harbor resources ready (50 retries left).
```

Wait approximately 30 minutes. The Intel® Smart Edge Open deployment script waits for the Harbor resources to be ready.

Installation Stuck

If the installation remains stuck with the following log:

```
TASK [infrastructure/os_setup : enable UFW]
*********************************************************************************************
*********************************************************************************************
task path: /root/dek/roles/infrastructure/os_setup/tasks/ufw_enable_debian.yml:12
Wednesday 16 February 2022  15:53:04 +0000 (0:00:01.627)       0:08:03.425 ****
NOTIFIED HANDLER reboot server for controller
changed: [controller] => {
   "changed": true,
   "commands": [
      "/usr/sbin/ufw status verbose",
      "/usr/bin/grep -h '^### tuple' /lib/ufw/user.rules /lib/ufw/user6.rules /etc/ufw/
user.rules /etc/ufw/user6.rules /var/lib/ufw/user.rules /var/lib/ufw/user6.rules",
      "/usr/sbin/ufw -f enable",
      "/usr/sbin/ufw status verbose",
      "/usr/bin/grep -h '^### tuple' /lib/ufw/user.rules /lib/ufw/user6.rules /etc/ufw/
user.rules /etc/ufw/user6.rules /var/lib/ufw/user.rules /var/lib/ufw/user6.rules"
   ]
}

MSG:

Status: active
Logging: on (low)
Default: deny (incoming), allow (outgoing), deny (routed)
New profiles: skip

To                         Action      From
--                         ------      ----
22/tcp                     ALLOW IN    Anywhere
22/tcp (v6)                ALLOW IN    Anywhere (v6)
```

Type `Ctrl-c`, and restart the installation. (Run the `./deploy.sh` script again.)

Pod Remains in “Terminating” State after Uninstall

After uninstall, if the pod does not stop but remains in “Terminating” state, enter the following commands:

```
kubectl get pods -n fleet-management
kubectl delete -n <pod_name_from_above_command> --grace-period=0 --force
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_uninstall.yaml
```

docker-compose Failure

If you see an error message that `docker-compose` fails with some variables not defined, add the environment variables to `.bashrc` so that they are available to all terminals:

```
export DOCKER_BUILDKIT=1
export COMPOSE_DOCKER_CLI_BUILD=1
export DOCKER_HOSTNAME=$(hostname)
export DOCKER_USER_ID=$(id -u)
export DOCKER_GROUP_ID=$(id -g)
export DOCKER_USER=$(whoami)
# Check with command
env | grep DOCKER
```

Keytool Not Installed

The keytool utility is used to create the certificate store. Install any preferred Java* version. For development, Intel used:

```
sudo apt install default-jre
# Check your Java version:
java -version
```

Corrupt Database or Nonresponsive Server

Reset the ThingsBoard* server with the following steps.

1. Uninstall the playbook:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_uninstall.yaml
```

2. After uninstalling the playbook, wait several seconds for all fleet related containers to stop. Verify that there are no fleet containers running:

```
docker ps | grep fleet
```

3. Reinstall the playbook:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_install.yaml
```

ThingsBoard* Server Errors

These errors can be fixed directly on the hosting machine using Docker* Compose. However, this requires automated steps using Ansible* playbooks, so try these fixes last.

- Reset the database to a pristine state (without customizations from Intel):

```
# delete database and start the server
# The state of server is - without any customization from Intel.
sudo rm -rf ~/.mytb-data/db ~/.mytb-data/.firstlaunch ~/.mytb-data/.upgradeversion
docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml down
CHOOSE_USER=thingsboard docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-
server.all.yml up fleet-management
```

> **NOTE** This only restarts the ThingsBoard* server, without Intel® Smart Edge Open.

- Reset the database to the preconfigured state (with customizations from Intel), and restart the server:

```
# Start the server with old/corrupted database
CHOOSE_USER=thingsboard docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-
server.all.yml up fleet-management
# attach to running container from another terminal:
docker exec -it  edge-server-sdk-fleet-management bash

# inside the container: replace the database with Intel-customized-database:
# Just press tb<tab>. The tb-server-reset-db.sh is present in /usr/local/bin folder, so it is
accessible from anywhere.
tb-server-reset-db.sh
# When asked press y and enter. Done.
# Now exit the container. and run below commands again to re-launch the server with
preconfigured-state of database (With Intel Customizations):
docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-server.all.yml down
CHOOSE_USER=thingsboard docker-compose -f 01_docker_sdk_env/docker_compose/02_edge_server/edge-
server.all.yml up fleet-management
```

NOTE This only restarts the ThingsBoard* server, without Intel® Smart Edge Open.

- When you deploy the ThingsBoard* container using Intel® Smart Edge Open Ansible* playbook, sometimes the server cannot start due to following error:

```
edge-server-sdk-fleet-management | 2021-11-25 15:24:34,345 [main] ERROR
com.zaxxer.hikari.pool.HikariPool - HikariPool-1 - Exception during pool initialization.
edge-server-sdk-fleet-management | org.postgresql.util.PSQLException: Connection to
localhost:5432 refused. Check that the hostname and port are correct and that the postmaster is
accepting TCP/IP connections.
edge-server-sdk-fleet-management |      at
org.postgresql.core.v3.ConnectionFactoryImpl.openConnectionImpl(ConnectionFactoryImpl.java:303)
edge-server-sdk-fleet-management |      at
org.postgresql.core.ConnectionFactory.openConnection(ConnectionFactory.java:51)
edge-server-sdk-fleet-management |      at
org.postgresql.jdbc.PgConnection.<init>(PgConnection.java:223)
edge-server-sdk-fleet-management |      at org.postgresql.Driver.makeConnection(Driver.java:465)
edge-server-sdk-fleet-management |      at org.postgresql.Driver.connect(Driver.java:264)
```

  If, after waiting for some time, the server is not up and running, and the server URL localhost:9090 is not showing the server page, uninstall and reinstall the playbook:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_uninstall.yaml
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_install.yaml
```

  Result: The database is reset to the preconfigured database provided by Intel.

Fleet Management Server Dashboard over LAN Issues

If the Dashboard is not accessible from the client, the first step is to make sure that the client and server nodes are in the same subnet. This helper page can be used to find out: https://www.meridianoutpost.com/resources/etools/network/two-ips-on-same-network.php

If the client and server are in the same subnet, then it is possible that you are using proxies that prevent the connection. To check this on Linux, run the following command:

```
wget -q -T 3 -t 3 --no-proxy http://<IP>:9090/ && echo "COMMAND PASSED"
```

Where `<IP>` is the IP of your server.

If `COMMAND PASSED` is displayed, then you should configure your browser to NOT use proxy when accessing the IP/hostname of the server.

Playbook Install Errors

If you start the basic fleet management server right after a server reboot, you may encounter the error:

```
fatal: [localhost]: FAILED! => {"changed": false, "msg": "Logging into 10.237.22.88:30003 for
user admin
failed - 500 Server Error for http+docker://localhost/v1.41/auth: Internal Server Error
(\"Get \"https://10.237.22.88:30003/v2/\": dial tcp 10.237.22.88:30003: connect: connection
refused\")"}
```

1. Wait two minutes until the server is up and running.
2. Verify that all pods are running and no errors are reported:

```
kubectl get all -A
```

3. After all pods and services are up and running, restart the basic fleet management server:

```
ansible-playbook AMR_server_containers/01_docker_sdk_env/docker_orchestration/ansible-playbooks/
02_edge_server/fleet_management/fleet_management_playbook_install.yaml
```

Terminology

Term	Description
ADBSCAN	Adaptive Density-Based Spatial Clustering of Applications with Noise
AGV	Autonomous Guided Vehicle
AI	Artificial Intelligence
amcl	adaptive Monte-Carlo localizer
AOTA	Application Over The Air
API	Application Programming Interface
ARIAC	Agile Robotics for Industrial Automation Competition
AT	ATtention
BRIEF	Binary Robust Independent Elementary Feature
CNDA	Corporate Non-Disclosure Agreement
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DDS	Data Distribution Service
DI	Device Initialization protocol
DL	Deep Learning
DMS	Device Management Service
DNS	Domain Name System
DPC++	Data Parallel C++
DRM	Deterministic Road Map
EI for AMR	Edge Insights for Autonomous Mobile Robots
EOF	End-Of-File
FAST	Features from Accelerated and Segments Test
FDO	FIDO Device Onboard
FIDO	Fast IDentity Online
FLANN	Fast Library for Approximate Nearest Neighbors

Term	Description
FM	FastMapping
FOTA	Firmware Over The Air
GEAR	Gazebo Environment for Agile Robotics
GPU	Graphics Processor Unit
GPS	Global Positioning System
GSLAM	General Simultaneous Localization and Mapping
GUI	Graphical User Interface
ICP	Iterative Closest Point
IDE	Integrated Development Environment
IE	Inference Engine
IMU	Inertial Measurement Unit
IP	Intellectual Property
IPU	Image Processing Unit
Intel® SDO	Intel® Secure Device Onboard
ITS	Intelligent sampling and Two-way Search
JIT	Just-In-Time
KVM	Kernel-based Virtual Machine
LAN	Local Area Network
LIDAR	LIght Detection And Ranging
MBIM	Mobile Interface Broadband Model
MLS	Moving Least Squares
MQTT	Message Queuing Telemetry Transport
MSM	Mobile Station Modem
NFS	Network File System
NN	Neural Network
ORB	Oriented FAST and Rotated BRIEF
OSRF	Open Source Robotics Foundation
OS	Operating System
OTA	Over The Air
PCL	Point Cloud Library
POTA	Programming Over The Air (includes SOTA and FOTA)
PRM	Probabilistic Road Map

Term	Description
QCDM	Qualcomm Diagnostic Monitor
QMI	Qualcomm MSM Interface
RDC	Resource and Documentation Center
RGBD	Red, Green, Blue plus Depth
ROS	Robot Operating System
RPLIDAR	360-degree 2D LIDAR solution developed by SLAMTEC
RPM	Red Hat* Package Manager
RTAB-Map	Real-Time Appearance-Based Mapping
RV	RendezVous
SCTP	Stream Control Transmission Protocol
SDK	Software Development Kit
SFTP	SSH File Transfer Protocol
SLAM	Simultaneous Localization And Mapping
SOTA	operating System Over The Air
SPIR	Standard Portable Intermediate Representation
SSD	Single-Shot multi-box Detection
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TMI	Test Module Interface
UDP	User Datagram Protocol
UEFI	Unified Extensible Firmware Interface
VNC	Virtual Network Computing
vSLAM	visual Simultaneous Localization And Mapping
WWAN	Wireless Wide Area Network

Notices and Disclaimers

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

Intel technologies may require enabled hardware, software or service activation.

No product or component can be absolutely secure.

Your costs and results may vary.

Code names are used by Intel to identify products, technologies, or services that are in development and not publicly available. These are not “commercial” names and not intended to function as trademarks.

Performance varies by use, configuration and other factors. Learn more at www.Intel.com/PerformanceIndex.