Edgecore

**Device Manager**

Installation & Deployment Guide

v1.0.1

November 16, 2020

Edgecore Networks

# **Device-Management Repository**

This Repo contains the code for Device-Manager and related functionality. Device-manager is module that collects the device data from the devices that support REDFISH and publishes onto kafka bus. User-application is another software that listens on kafka bus and makes the data available to the dashboard for user.

# **Device-Manager Platform**

Device Manager gets the device details from devices and periodically collects data using REDFISH RESTful APIS based on HTTP. The interface (gRPC) is between Device-Manager and user-application. Device-Manager also registers specific Redfish APIs for events from the device like alerts, removal/insertion events. It then publishes data on kafka bus to collect the data.

# **Preparation**

The host system need to install necessary packages (ex: git, curl, unzip and docker)

* sudo apt update
* sudo apt upgrade
* sudo apt install git curl unzip

# **Installation Procedures**

## **Download Device Management**

The all needed files are located in the Edge-core Github. You could use git command to download all files.

## **Install Kubernetes Environment**

The device management based on the k8s environment to cooperate with others PODs (ex: core-kafka-0).

## **Install Docker tool**

* make install-docker

After this command, you need to logout/reboot the host system to take effect on the running system.

## **Install Kubernetes tools and kube-system Pods**

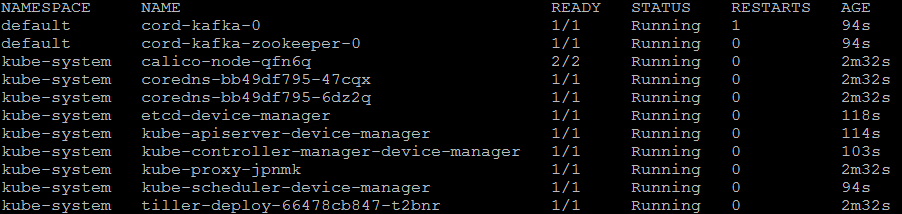
* make k8s

Before this command, you need to add the “nameserver” variable (ex: nameserver 8.8.8.8) to "/etc/resolv.conf".

## **Check kube-system Pods**

After installed the k8s Pods, you could use the command to check the status of Pods.

* kubectl get pods --all-namespaces



## **Download and build Device-Manager Docker image**

The images of device management will be downloaded to the host, and build those source files.

The following by this command to build the Device Management docker image.

* make build-dm

If you encountered that fails to download images, you need to use this command to fix it.

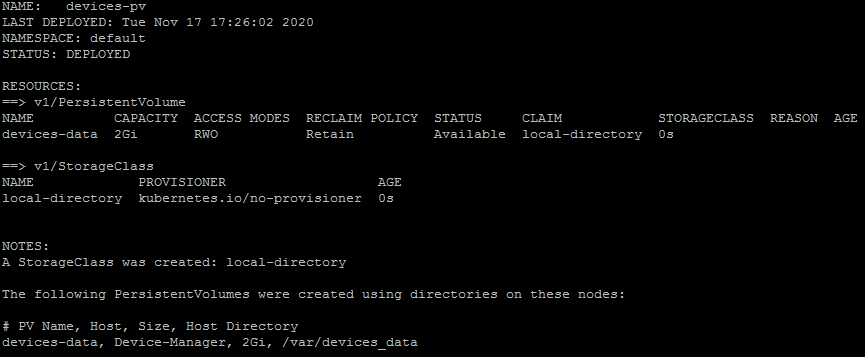
* sudo systemctl restart docker

## **Install Device-Manager Pod**

The device management would follow commands to bring up the Pod.

Bring up the Device Persistent Volume first (Default: /var/devices\_data) . The device data file could store in the host platform.

* make dpv



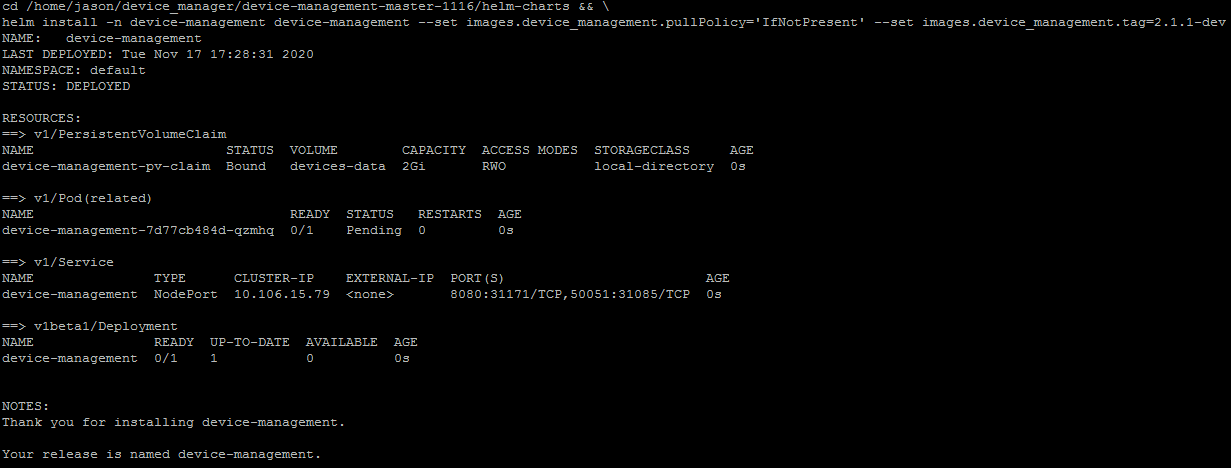
Displaying the device persistent volume status

* helm ls



Bring up the Device-Manager Pod

* make dm



After bring up the Pods, you could use the command to check the status of device management Pod.

* kubectl get pods --all-namespaces | grep device-management



## **Unload Device-Manager Pod and Device Persistent Voluem**

The devi

The command is unloading the Device-Manager pod

* make clean-dm



The command is unloading the device persistent volume helm chart.

* make clean-dpv



# **Build and Run demo test**

Before you build the demotest tool, some of packages needs to install, For example: go packages.

* cd demo\_test

## **Build demo test**

Install go complier tool to host platform

* make go-install

Take effect the GO environment variables

* . ~/.bashrc

Install go library, APIs, and “protoc” tool

* make prereq

The demotest is a daemon that create the connection interface for accessing the device.

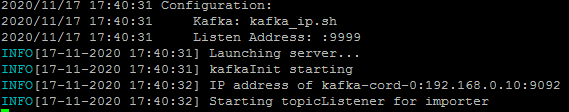
* cd demo\_test
* make demotest



## **Run demo test**

After built the demotest, You could run the daemon in the foreground and listen by the "dm" program command.

* cd demo\_test
* ./demotest



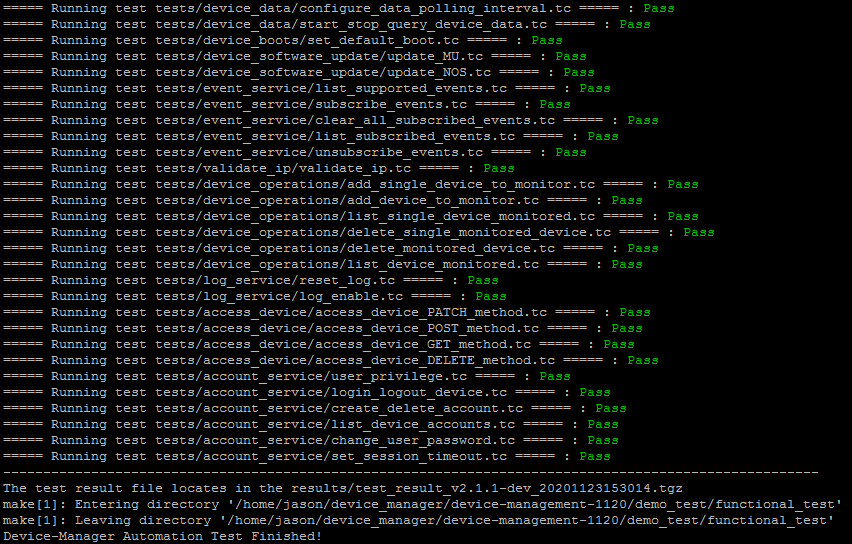
# **Test Physical Device**

The automation test needs two physical devices to perform the test cases that include getting device data and functionalities.

## **Automation Test**

Test cases utilizing 'dm' provided in the functional\_test/ sub-directory. The test results will save a tarball file and locates in the "results" directory. They can execute through Makefile at command line, type

* cd demo\_test/functional\_test
* make test IP1=<ip of 1st device> PORT1=<RF port of 1st device> IP2=<ip of 2nd device> PORT2=<RF port of 2nd device>



The test case could specific by the "TESTSDIR" option (for exmaple: tests/account\_service)

* cd demo\_test/functional\_test
* make test IP1=<ip of 1st device> PORT1=<RF port of 1st device> IP2=<ip of 2nd device> PORT2=<RF port of 2nd device> TESTSDIR=<test case directory>

## **Manual testing at command line**

The 'dm' test tool needs to build at the command line the following by

* cd demo\_test/functional\_test
* make



For running 'dm', please make and launch 'demotest' first.

# **Reset kubernetes (k8s) environment**

The command is removing all pods and helm chart.

* make reset-pods

