Device Manager

Installation & Deployment Guide

v1.0.1

NOVEMBER 16, 2020

EDGECORE NETWORKS

1. 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.

2. 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.

3. 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

4. Installation Procedures

4.1 Download Device Management

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

4.2 Install Kubernetes Environment

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

4.2.1 Install Docker tool

- make install-docker

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

4.2.2 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".

4.2.3 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

```
NAMESPACE     NAME                                      READY   STATUS    RESTARTS   AGE
default       cord-kafka-0                              1/1     Running   1          94s
default       cord-kafka-zookeeper-0                    1/1     Running   0          94s
kube-system   calico-node-qfn6q                         2/2     Running   0          2m32s
kube-system   coredns-bb49df795-47cqx                   1/1     Running   0          2m32s
kube-system   coredns-bb49df795-6dz2q                   1/1     Running   0          2m32s
kube-system   etcd-device-manager                       1/1     Running   0          118s
kube-system   kube-apiserver-device-manager             1/1     Running   0          114s
kube-system   kube-controller-manager-device-manager    1/1     Running   0          103s
kube-system   kube-proxy-jpnmk                          1/1     Running   0          2m32s
kube-system   kube-scheduler-device-manager             1/1     Running   0          94s
kube-system   tiller-deploy-66478cb847-t2bnr            1/1     Running   0          2m32s
```

4.3 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

4.4 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

```
NAME:   devices-pv
LAST DEPLOYED: Tue Nov 17 17:26:02 2020
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/PersistentVolume
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS     CLAIM            STORAGECLASS  REASON  AGE
devices-data  2Gi       RWO           Retain          Available  local-directory  0s

==> v1/StorageClass
NAME            PROVISIONER                    AGE
local-directory  kubernetes.io/no-provisioner  0s


NOTES:
A StorageClass was created: local-directory

The following PersistentVolumes were created using directories on these nodes:

# PV Name, Host, Size, Host Directory
devices-data, Device-Manager, 2Gi, /var/devices_data
```

Displaying the device persistent volume status

- helm ls

```
NAME            REVISION        UPDATED                         STATUS          CHART                           APP VERSION     NAMESPACE
cord-kafka      1               Mon Nov 16 11:21:14 2020        DEPLOYED        kafka-0.13.3                    5.0.1           default
devices-pv      1               Tue Nov 17 17:26:02 2020        DEPLOYED        local-directory-0.1.0-dev0                      default
```

Bring up the Device-Manager Pod

- make dm

```
cd /home/jason/device_manager/device-management-master-1116/helm-charts && \
helm install -n device-management device-management --set images.device_management.pullPolicy='IfNotPresent' --set images.device_management.tag=2.1.1-dev
NAME:   device-management
LAST DEPLOYED: Tue Nov 17 17:28:31 2020
NAMESPACE: default
STATUS: DEPLOYED

RESOURCES:
==> v1/PersistentVolumeClaim
NAME                        STATUS  VOLUME        CAPACITY  ACCESS MODES  STORAGECLASS     AGE
device-management-pv-claim  Bound   devices-data  2Gi       RWO           local-directory  0s

==> v1/Pod(related)
NAME                                  READY  STATUS   RESTARTS  AGE
device-management-7d77cb484d-qzmhq    0/1    Pending  0         0s

==> v1/Service
NAME              TYPE      CLUSTER-IP    EXTERNAL-IP  PORT(S)                        AGE
device-management  NodePort  10.106.15.79  <none>       8080:31171/TCP,50051:31085/TCP  0s

==> v1beta1/Deployment
NAME              READY  UP-TO-DATE  AVAILABLE  AGE
device-management  0/1    1           0          0s


NOTES:
Thank you for installing device-management.

Your release is named device-management.
```

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

```
default         device-management-7d77cb484d-qzmhq        1/1     Running   0          70s
```

4.5 Unload Device-Manager Pod and Device Persistent Voluem

The devi

The command is unloading the Device-Manager pod

- make clean-dm

```
release "device-management" deleted
```

The command is unloading the device persistent volume helm chart.

- make clean-dpv

```
release "devices-pv" deleted
```

5. Build and Run demo test

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

- cd demo_test

5.1 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

```
GO111MODULE=on go build -mod=vendor -i -v -o demotest
```

5.2 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

```
2020/11/17 17:40:31 Configuration:
2020/11/17 17:40:31     Kafka: kafka_ip.sh
2020/11/17 17:40:31     Listen Address: :9999
INFO[17-11-2020 17:40:31] Launching server...
INFO[17-11-2020 17:40:31] kafkaInit starting
INFO[17-11-2020 17:40:32] IP address of kafka-cord-0:192.168.0.10:9092
INFO[17-11-2020 17:40:32] Starting topicListener for importer
```

6. Test Physical Device

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

6.1 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>

```
===== Running test tests/device_data/configure_data_polling_interval.tc ===== : Pass
===== Running test tests/device_data/start_stop_query_device_data.tc ===== : Pass
===== Running test tests/device_boots/set_default_boot.tc ===== : Pass
===== Running test tests/device_software_update/update_MU.tc ===== : Pass
===== Running test tests/device_software_update/update_NOS.tc ===== : Pass
===== Running test tests/event_service/list_supported_events.tc ===== : Pass
===== Running test tests/event_service/subscribe_events.tc ===== : Pass
===== Running test tests/event_service/clear_all_subscribed_events.tc ===== : Pass
===== Running test tests/event_service/list_subscribed_events.tc ===== : Pass
===== Running test tests/event_service/unsubscribe_events.tc ===== : Pass
===== Running test tests/validate_ip/validate_ip.tc ===== : Pass
===== Running test tests/device_operations/add_single_device_to_monitor.tc ===== : Pass
===== Running test tests/device_operations/add_device_to_monitor.tc ===== : Pass
===== Running test tests/device_operations/list_single_device_monitored.tc ===== : Pass
===== Running test tests/device_operations/delete_single_monitored_device.tc ===== : Pass
===== Running test tests/device_operations/delete_monitored_device.tc ===== : Pass
===== Running test tests/device_operations/list_device_monitored.tc ===== : Pass
===== Running test tests/log_service/reset_log.tc ===== : Pass
===== Running test tests/log_service/log_enable.tc ===== : Pass
===== Running test tests/access_device/access_device_PATCH_method.tc ===== : Pass
===== Running test tests/access_device/access_device_POST_method.tc ===== : Pass
===== Running test tests/access_device/access_device_GET_method.tc ===== : Pass
===== Running test tests/access_device/access_device_DELETE_method.tc ===== : Pass
===== Running test tests/account_service/user_privilege.tc ===== : Pass
===== Running test tests/account_service/login_logout_device.tc ===== : Pass
===== Running test tests/account_service/create_delete_account.tc ===== : Pass
===== Running test tests/account_service/list_device_accounts.tc ===== : Pass
===== Running test tests/account_service/change_user_password.tc ===== : Pass
===== Running test tests/account_service/set_session_timeout.tc ===== : Pass
-----------------------------------------------------------------------------------------------
The test result file locates in the results/test_result_v2.1.1-dev_20201123153014.tgz
make[1]: Entering directory '/home/jason/device_manager/device-management-1120/demo_test/functional_test'
make[1]: Leaving directory '/home/jason/device_manager/device-management-1120/demo_test/functional_test'
Device-Manager Automation Test Finished!
```

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>

6.2 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

```
go build -i -v -o dm
device-management/demo_test/functional_test
```

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

7. Reset kubernetes (k8s) environment

The command is removing all pods and helm chart.

- make reset-pods

```
sudo kubeadm reset -f || true
[sudo] password for jason:
[preflight] running pre-flight checks
[reset] stopping the kubelet service
[reset] unmounting mounted directories in "/var/lib/kubelet"
[reset] deleting contents of stateful directories: [/var/lib/kubelet /etc/cni/net.d /var/lib/dockershim /var/run/kubernetes /var/lib/etcd]
[reset] deleting contents of config directories: [/etc/kubernetes/manifests /etc/kubernetes/pki]
[reset] deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.conf /etc/kubernetes/controller-manager.conf /etc/kubernetes/scheduler.conf
sudo iptables -F && sudo iptables -t nat -F && sudo iptables -t mangle -F && sudo iptables -X
sudo rm -f /var/lib/cni/networks/pon*/* || true
sudo rm -f /var/lib/cni/networks/nni*/* || true
sudo rm -f /var/lib/cni/networks/k8s-pod-network/* || true
```