SEBA Testing

This page is about work in progress regarding SEBA validation and testing, more information should be added in the near future.

SEBA on ARM - porting and upstreaming work on this wiki describes the work done for porting and verifying SEBA on ARM servers in particular.

On the the more general subject of testing SEBA as a whole, there were some emails exchanged on the Akraino technical-discuss mailing list, where some valuable feedback was received from Zack Williams of Open Network Foundation:
https://lists.akraino.org/g/technical-discuss/message/138

More input was received from Aaron Byrdhere:
https://lists.akraino.org/g/technical-discuss/message/142

SEBA-in-a-Box and PONsim

For validation purposes, the SIAB guide shows how to easily deploy SEBA on a single server and install PONsim for emulating real OLT/ONU hardware. See the SEBA-in-aBox guide

Installing PONsim on IEC

The installation instruction on the SIAB guide asume you are installing SEBA either using automation-tools, or by deploying the components individually using the charts on https://gerrit.opencord.org/helm-charts

It should be possible however to use PONsim on the Akraino IEC platform, after installing SEBA as described in SEBA Installation Guide. These have not been verified on ARM servers yet and some errors occurs when pods get scheduled on different servers, since IEC platform defines 3 servers for the Kubernetes cluster.

Here are some crude, general instructions on how to install SEBA and PONsim on the IEC platform. The instructions have been tailored for working with the IEC platform, as per the instructions on IEC Blueprints Installation Overview, using the OPNFV Fuel installer.

Some networking aspects have been changed to match the network addresses used by the installer, but you can always compare with the SIAB installation guide to match your environment.

```
helm repo add cord https://charts.opencord.org
helm repo update
kubectl get pods --all-namespaces | grep tiller
helm install -n cord-platform cord/cord-platform --version=6.1.0
kubectl get crd | grep -i etcd | wc -l
kubectl get pod | grep -i Running | wc -l
helm install -n seba cord/seba --version=1.0.0
kubectl get pod --all-namespaces | grep -i Running | wc -l
helm install -n att-workflow cord/att-workflow --version=1.0.2
kubectl get pods --all-namespaces | grep att

mkdir cord
cd cord

# From the SIAB tutorial install ponnet and ponsimv2
wget https://charts.opencord.org/ponnet-1.0.0.tgz
tar xzf ponnet-1.0.0.tgz
sed -i 's/10.22/101.22/g' ponnet/templates/pon0-cni.yaml
sed -i 's/10.23/101.23/g' ponnet/templates/pon0-cni.yaml
helm install -n ponnet ponnet

wget https://charts.opencord.org/ponsimv2-1.0.3.tgz
tar xzf ponsimv2-1.0.3.tgz
helm install -n ponsimv2 ponsimv2

# on all nodes enable forwarding
sudo iptables -P FORWARD ACCEPT

# check voltha
kubectl -n voltha get pod

# get mininet charts
wget https://charts.opencord.org/mininet-0.2.0.tgz
tar xzf mininet-0.2.0.tgz
# Fix onos service name
sed -i 's/\<onos-openflow/cord-platform-onos-openflow/g' mininet/values.yaml
# on all nodes load openvswitch module
sudo modprobe openvswitch
# install chart
helm install -n mininet mininet
```

```
kubectl attach -ti deployment.apps/mininet
CTRL-P CTRL-Q
# find pon0
brctl show
# enable pon0 to forward EAPOL packets
echo 8 > /tmp/pon0_group_fwd_mask
sudo cp /tmp/pon0_group_fwd_mask /sys/class/net/pon0/bridge/group_fwd_mask

# install ponsim-pod chart
wget https://charts.opencord.org/ponsim-pod-1.1.0.tgz
tar xzf ponsim-pod-1.1.0.tgz
# replace ipv4loopback address in 030-fabric
sed -i 's/192.168.0/100.100.0/g' ponsim-pod/tosca/030-fabric.yaml
helm install -n ponsim-pod ponsim-pod

# Check that ponsim-pod is awaiting authentication
http -a admin@opencord.org:letmein GET \
http://127.0.0.1:30001/xosapi/v1/att-workflow-driver/attworkflowdriverserviceinstances | \
jq '.items[0].authentication_state' | grep AWAITING
# ONOS customizations to instruct the ONU to exchange untagged packets with the RG instead of packets tagged with
VLAN 0
http -a karaf:karaf POST \
http://127.0.0.1:30120/onos/v1/configuration/org.opencord.olt.impl.Olt defaultVlan=65535

# Validating the install; authenticate the RG
RG_POD=$( kubectl -n voltha get pod -l "app=rg" -o jsonpath='{.items[0].metadata.name}' )
kubectl -n voltha exec -ti $RG_POD bash
# inside the RG pod we need to run this command to authenticate;
# hit Ctrl-C after authentication is complete (about 5 seconds) and then exit the interactive shell
wpa_supplicant -i eth0 -Dwired -c /etc/wpa_supplicant/wpa_supplicant.conf
# Now the AttDriverWorkflow Service Instance should be in APPROVED state. To check use this command:
http -a admin@opencord.org:letmein GET \
http://127.0.0.1:30001/xosapi/v1/att-workflow-driver/attworkflowdriverserviceinstances | \
jq '.items[0].authentication_state' | grep APPROVED
# The FabricCrossconnect Service Instance should have a check in the Backend status column. You can check for
this by running:
http -a admin@opencord.org:letmein GET \
http://127.0.0.1:30001/xosapi/v1/fabric-crossconnect/fabriccrossconnectserviceinstances | \
jq '.items[0].backend_status'|grep OK

# Again on the RG POD, you need to run dhclient to obtain a valid IP address from the 172.18.0.0/24 subnet
ifconfig eth0 0.0.0.0
dhclient eth0
```

Test cases to validate Host OS Environment for SEBA installation ***NOT COMPLETE - Needs to be updated*** *- Feel free to add validation (where needed)*

NOTE: SEBA specific functional testing & validation is documented in SEBA upstream project - https://guide.opencord.org/cord-6.1/cord-tester/

1. OS Version Test:

cat /etc/*release

Expected Output: *(expect this version to be updated*)

DISTRIB_ID=Ubuntu

DISTRIB_RELEASE=16.04

DISTRIB_CODENAME=xenial

DISTRIB_DESCRIPTION="Ubuntu 16.04.6 LTS"

NAME="Ubuntu"

VERSION="16.04.6 LTS (Xenial Xerus)"

ID=ubuntu

ID_LIKE=debian

PRETTY_NAME="Ubuntu 16.04.6 LTS"

VERSION_ID="16.04"

Note: SEBA setup tested Ubuntu v16.04. Other versions need to be tested for compatibility.

2. Repo Connectivity Test: (Offline install)

validation of connected Repo?? *(Need to update to show validation step)*

Previous connectivity was for internet access (online install)

3. NIC Interface Test:

lshw -class network

Expected Output: Current SEBA setup requires at least 2-4 Physical Network Interface cards.

4. RAM Test:

free -h

Expected Output: Minimum RAM requirement for installing SEBA architecture is approx. 32Gb.

5, Storage Test:

df -h

Expected Output: Minimum Storage requirement for installing SEBA architecture is approx. 100Gb.

6. Root Privileges:

sudo bash

Expected Output: Above command should run successfully without any errors.

7. Debian packages:

sudo apt install openssh-server

Expected Output:

Reading package lists... Done

Building dependency tree

Reading state information... Done

openssh-server is already the newest version (1:7.2p2-4ubuntu2.8).

Additional Notes: Above command should run successfully without any errors.

8. Docker Version Test:

#docker --version

Expected Output: Docker version 18.06.1-ce, build e68fc7a

Additional Notes: Current SEBA setup is tested with Docker version 17.03.2~ce-0~ubuntu-xenial

9. Kubernittis Cluster – check Health/validation

kubectl get pods --all-namespaces

10. VM Validation:

Management VM – Provides connectivity to BNG system

11. Validate the following Services

- Repository – keeps sw images and configuration files (helm charts , ONOS apps)
- DNS – used for internal server connectivity (repo server , inventory server , policy server , ansible server)