ODL COE

Release master

OpenDaylight Project

Aug 29, 2020

CONTENTS

This documentation provides critical information needed to help you understand the Container Orchestration Engine (COE) project in OpenDaylight. The project develops a framework for integrating the Container Orchestration Engine Kubernetes and OpenDaylight.

Contents:

CHAPTER

ONE

SETTING UP COE DEV ENVIRONMENT

For COE to work end to end, the below modules are required :

1. ODL K8S Watcher
2. OVS CNI plugin
3. ODL netvirt
4. Kubernetes Orchestration

Subsequent sections explain how to set up each of the above modules in a development environment. A VagrantFile for doing the below steps is already available in coe repository under coe/resources folder.

1.1 Building COE Watcher and CNI Plugin

Watcher and odlovscni modules reside within ODL COE repository. These modules are written in golang and for compiling the binaries you need a golang environment.

- install golang
  - download golang from https://golang.org/doc/install?download
  - mkdir $HOME/opt
  - tar -xvf golang tar filename (inside opt)
  - mkdir $HOME/go
  - mkdir $HOME/go/bin
  - export GOPATH=$HOME/go
  - export GOROOT=$HOME/opt/go
  - export PATH=$PATH:$GOROOT/bin:$GOPATH/bin
- install dep
  - curl https://raw.githubusercontent.com/golang/dep/master/install.sh | sh
  - check for dep in $HOME/go/bin
- install git and clone coe repository
  - sudo apt install git
  - go get -d git.opendaylight.org/gerrit/p/coe.git
  - check for git in $HOME/go/src

- build coe watcher binary
  - cd $GOPATH/src/git.opendaylight.org/gerrit/p/coe.git/watcher
  - dep ensure -vendor-only
  - a vendor file gets created in the same folder
  - go build
  - once the above step is completed, a "watcher" binary will be generated in the same folder
- build coe cni plugin
  - cd $GOPATH/src/git.opendaylight.org/gerrit/p/coe.git/odlCNIPlugin/odlovs-cni
  - dep ensure -vendor-only
  - a vendor file gets created in the same folder
  - go build
  - cni binary "odlovs-cni" will be created under same folder

1.1.1 Setting Up ODL Netvirt

- clone and build netvirt repository
  - This step is required only if you want to make some changes to the existing code and experiment. Else you can just download the latest odl distribution go get git.opendaylight.org/gerrit/p/netvirt.git cd netvirt mvn clean install
- **Run ODL**
  - cd karaf/target/assembly/bin
  - karaf clean
  - Once the karaf console comes up, install odl-netvirt-coe feature which will bring up all required modules for k8s integration
  - opendaylight-karaf>feature:install odl-netvirt-coe

1.1.2 Setting Up K8S Master and Minions

VMs need to be setup to run K8S Master and Minions. OVS, Docker and K8S need to be installed on all 3 VMs based on the steps specified below. The same is available as part of the Vagrant setup script under coe/resources/K8s-ODL-Vagrant/provisioning/setup-Req.sh [https://github.com/opendaylight/coe/blob/master/resources/K8s-ODL-Vagrant/provisioning/setup-Req.sh]

1.1.3 Install OVS

- sudo apt-get build-dep dkms
- **sudo apt-get install -y autoconf automake bzip2 debhelper dh-autoreconf** libssl-dev libtool openssl procps python-six dkms
- git clone https://github.com/openvswitch/ovs.git
- pushd ovs/
- sudo DEB_BUILD_OPTIONS='nocheck parallel=2' fakeroot debian/rules binary

- popd
- sudo dpkg -i openvswitch-datapath-dkms*.deb
- **sudo dpkg -i openvswitch-switch*.deb openvswitch-common*.deb** python-openvswitch*.deb libopenvswitch*.deb

1.1.4 Install Docker

- sudo apt-get update
- sudo apt-get install -y apt-transport-https ca-certificates
- sudo apt-key adv –keyserver hkp://p80.pool.sks-keyservers.net:80 –recv-keys 58118E89F3A912897C070ADBF76221572C52609D
- sudo su -c "echo "deb https://apt.dockerproject.org/repo ubuntu-xenial main" >> /etc/apt/sources.list.d/docker.list"
- sudo apt-get update
- sudo apt-get purge lxc-docker
- sudo apt-get install -y linux-image-extra-$(uname -r) linux-image-extra-virtual
- sudo apt-get install -y docker-engine bridge-utils
- sudo service docker start

1.1.5 Install Kubernetes

- sudo curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -
- sudo su -c "echo "deb http://apt.kubernetes.io/ kubernetes-xenial main" >> /etc/apt/sources.list.d/kubernetes.list"
- sudo apt-get update
- sudo apt-get install -y kubelet kubeadm kubernetes-cni

1.1.6 Setting Up K8S CNI Plugin

The below steps can be found under the ReadMe file at https://github.com/opendaylight/coe/tree/master/resources/K8s-ODL-Vagrant

- sudo mkdir -p /etc/cni/net.d/
- copy the appropriate conf files present in coe repo to net.d folder
  - cd $GOPATH/src/git.opendaylight.org/gerrit/p/coe.git/resources/K8s-ODL-Vagrant/example
  - sudo cp master.odlovs-cni.conf /etc/cni/net.d/ [For minions, copy the worker conf file instead of master.conf]
- sudo mkdir -p /opt/cni/bin
- copy the odlovs-cni binary which we compiled from coe repo, to the cni/bin folder.
  - cd $GOPATH/src/git.opendaylight.org/gerrit/p/coe.git/odlCNIPlugin/odlovs-cni
  - sudo cp odlovs-cni /opt/cni/bin

1.1.7 Start Kubernetes Cluster

- sudo kubeadm init –apiserver-advertise-address={K8S-Master-Node-IP}
  - note: read the command output in order to use the kubectl command after
  - note: in the minion VMs you will use the join command instead ex:
  - vagrant@k8sMinion2:~$ sudo kubeadm join –token {given_token} {K8S-Master-Node-IP}:6443
  - mkdir -p $HOME/.kube
  - sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  - sudo chown $(id -u):$(id -g) $HOME/.kube/config

1.1.8 Start COE Watcher on K8S Master

- cd $GOPATH/src/git.opendaylight.org/gerrit/p/coe.git/watcher
- ./watcher odl
- The above step will start the coe watcher, which watches for events from kubernetes, and propagate the same to ODL. note : check for master config file in /etc/cni/net.d before running watcher note : for watcher to start properly, .kube/config file should be setup properly, this will be explained in the output of kubeadm init command.

1.1.9 Bring up PODs and test connectivity

- You can now bring up pods and see if they are able to communicate to each other.
- Pods can be brought up on same minion or on different minions.
- **Tunnels have to be created so that pods created on different minions will be able to communicate with other.**
  - To create tunnels run the following commands on master and minions:
  - sudo ovs-vsctl set O . other_config:local_ip={LOCAL_IP}
  - sudo ovs-vsctl set O . external_ids:br-name={BRIDGE_NAME} (bridge name will be the same as the value of “ovsBridge” in /etc/cni/net.d/master.odlovs-cni.conf)
- Nodes have to be given labels so that random allocation of pods can be avoided.For the pod to be eligible to run on a node, the node must have each of the indicated key-value pairs as labels.
  - **kubectl label nodes <node-name> <label-key>=<label-value>** eg:kubectl label nodes minion disk-type=ssd
- Also node selector has to be added at the end of pod configuration file.
- **L2 connectivity(bringing up pods on same minion):**
  - **Include a node selector at the end of your .yaml file(eg:busybox.yaml)**

    nodeSelector: <label-key>=<label-value>

    eg:disktype=ssd
  - Create pods using kubectl create -f {.yamlfile}
  - To get .yaml files from the web use, kubectl create -f https://raw.githubusercontent.com/kubernetes/kubernetes/master/hack/testdata/recursive/pod/pod/busybox.yaml

  - **To check the status of pods run kubectl get pods -o wide**

    eg: NAME READY STATUS RESTARTS AGE IP NODE busybox1 1/1 Running 1 1h 10.11.2.210 *minion*

    busybox2 1/1 Running 1 1h 10.11.2.211 *minion*

  - **Try pinging from one pod to another,** kubectl exec -it busybox1 ping 10.11.2.211

- **L3 connectivity(bringing up pods on different minions):**
  - Label the node with a different label-value and include the node selector in yaml file. eg: kubectl label nodes minion2 disktype=ssl
  - **To check the status of pods run kubectl get pods -o wide**

    eg: NAME READY STATUS RESTARTS AGE IP NODE busybox1 1/1 Running 1 1h 10.11.2.210 *minion*

    busybox2 1/1 Running 1 1h 10.11.2.211 *minion*

    busybox3 1/1 Running 1 1h 10.11.3.5 *minion2*
  - Tunnels have to be created between nodes before L3 ping is done.
  - **Try pinging from one pod to another,** kubectl exec -it busybox1 ping 10.11.2.211

1.1.10 Bring Up Services And Test Connectivity

Currently OpenDaylight has been tested to work only for masquerade mode of Kubernetes services, with kube-proxyIPTables mode. To get this to work a series of steps are required, an ansible version of this is already available in the coe.yml at https://github.com/opendaylight/coe/tree/master/resources/K8s-ODL-Vagrant/playbooks

- The master and worker config files needs to be updated to include the service IP range and the default gateway also requires an update. Complete sample configuration can be found at https://github.com/opendaylight/coe/tree/master/resources/K8s-ODL-Vagrant/playbooks/templates/odlovs-cni2.conf.j2
- Enable kube-proxy masquerade mode using:

  [vagrant@k8sMaster ~]$ kubectl edit configmap kube-proxy –namespace=kube-system. The above command will open the kube-proxy config map, in which the below attributes need to be updated.
  - masqueradeAll: true
  - clusterCIDR: “10.11.0.0/16”
- **Once kube-proxy configuration is changed, the kube-proxy pods need to be restarted to pick up the new mode.**
  [vagrant@k8sMaster ~]$ kubectl get pods –all-namespaces

  NAMESPACE NAME READY STATUS RESTARTS AGE

  kube-system etcd-k8smaster 1/1 Running 0 10m

  kube-system kube-apiserver-k8smaster 1/1 Running 0 10m

  kube-system kube-controller-manager-k8smaster 1/1 Running 0 10m

  kube-system kube-proxy-cpwqw 1/1 Running 0 10m

  kube-system kube-proxy-hkxbm 1/1 Running 0 2m

  kube-system kube-proxy-xwqdw 1/1 Running 0 6m

  kube-system kube-scheduler-k8smaster 1/1 Running 0 10m

  kube-system odlwatcher-hp5nm 1/1 Running 1 10m

[vagrant@k8sMaster ~]$ kubectl delete pod –namespace kube-system kube-proxy-cpwqw

pod “kube-proxy-cpwqw” deleted

[vagrant@k8sMaster ~]$ kubectl delete pod –namespace kube-system kube-proxy-hkxbm

pod “kube-proxy-hkxbm” deleted

[vagrant@k8sMaster ~]$ kubectl delete pod –namespace kube-system kube-proxy-xwqdw

pod “kube-proxy-xwqdw” deleted

- **A default-gateway port needs to be created on OVS, to take all packets destined to services to the IPTables. The following**

  [vagrant@k8sMinion1 ~] ovs-vsctl set Interface {{veth-default-gateway-port}} external-ids:iface-id=’{{cluster-id}}:minion-services’ external-ids:attached-mac=”{{veth-port-mac}}” external-ids:is-service-gateway=true

  [vagrant@k8sMinion1 ~] sudo ip link set br-int down

  [vagrant@k8sMinion1 ~] sudo ip addr add {{service-ip-address}}/24 dev {{veth-default-gateway-port}}

  [vagrant@k8sMinion1 ~] sudo ip link set dev {{veth-default-gateway-port}} up

  {{veth-default-gateway-port}} can be any veth port name, provided the same port name is not used already on OVS.

  Once the veth port is created on OVS, the configuration for the default-gateway needs to added to ODL as well.

  curl -v -X PUT -u admin:admin -H “Content-Type: application/json” -d @default-gateway-pod.json http://localhost:8181/restconf/config/pod:coe/pods/{{unique-uuid-to-represent-default-gateway-veth}}

  default-gateway-pod.json can be found at

  > https://github.com/opendaylight/coe/tree/master/resources/K8s-ODL-Vagrant/playbooks/templates/odl-pod.json.j2

  Make sure that the attributes used in default-gateway-pod.json match against the {default-gateway-veth-port} external-id setting. A sample external-id setting and corresponding JSON are given below:

  [vagrant@k8sMinion1 ~]ifconfig veth11111111

  veth11111111 Link encap:Ethernet HWaddr be:36:e8:01:bd:34 inet addr:10.11.2.254 Bcast:0.0.0.0 Mask:255.255.255.0

  ovs-vsctl set Interface veth11111111 external-ids:iface-id=‘00000000-0000-0000-0000-000000000001:minion-services’ external-ids:attached-mac=”be:4b:bd:ca:a9:98” external-ids:is-service-gateway=true

```
{
    "pods": [
        {
            "uid": "ab86e792-1a1d-11e9-bd54-080027bc132b",
            "interface": [
                {
                    "uid": "9caa3d73-1a17-11e9-bd54-080027bc131a",
                    "network-id": "00000000-0000-0000-0000-000000000000",
                    "network-type": "VXLAN",
                    "ip-address": "192.11.2.254"
                }
            ],
            "cluster-id": "00000000-0000-0000-0000-000000000001",
            "port-mac-address": "",
            "network-NS": "default",
```

(continues on next page)

(continued from previous page)

```
                "name": "minion-services",
                "host-ip-address": "192.168.56.101"
            }
        ]
}
```

- **IP Routes have to be added on Kubernetes nodes to distinguish between pod traffic and service traffic** ip route add 10.11.0.0/16 via {{ gateway }} ip route add 10.96.0.0/12 via {{ services_ip_address }}

  gateway and service_ip_address needs to be derived from the /etc/cni/net.d/{cni.conf} config file.

- **Create Kubernetes Service** kubectl apply -f https://github.com/opendaylight/coe/tree/master/resources/K8s-ODL-Vagrant/playbooks/examples/apache-e-w.yaml

  Verify if the service got created by the below command :

  [vagrant@k8sMaster ~]$ kubectl get service

  NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE

  apacheservice ClusterIP 10.107.87.106 <none> 8800/TCP 4h

  kubernetes ClusterIP 10.96.0.1 <none> 443/TCP 5h

- **Deploy apache webservice with a set of pods** kubectl apply -f https://github.com/opendaylight/coe/tree/master/resources/K8s-ODL-Vagrant/playbooks/examples/apache-deployment.yaml

  Verify if all the pods got deployed under the service by checking :

  [vagrant@k8sMaster ~]$ kubectl get pods -o wide NAME READY STATUS RESTARTS AGE IP NODE

  apache-deployment-8bf8f9b5c-g6c7j 1/1 Running 0 2h 10.11.2.35 k8sMinion1

  apache-deployment-8bf8f9b5c-gflb7 1/1 Running 0 2h 10.11.2.36 k8sMinion1

  apache-deployment-8bf8f9b5c-pl5v2 1/1 Running 0 2h 10.11.2.37 k8sMinion1

  busybox1 1/1 Running 4 4h 10.11.2.28 k8sMinion1

- **A temporary hack needs to be done on OVS to tweek the netvirt pipeline to get the services working.** ovs-ofctl -OOpenflow13 add-flow br-int table=21,priority=0,actions=resubmit\(,17\)

- **Check if kubernetes node to pod connectivity works.** This does not work without the default flow in Table=21 which is created in the previous step. Node to Pod connectvitiy is mandatory for services to work.

  [vagrant@k8sMaster ~]$ ping 10.11.2.35 -c 3

  PING 10.11.2.35 (10.11.2.35) 56(84) bytes of data.

  64 bytes from 10.11.2.35: icmp_seq=1 ttl=64 time=0.335 ms

- **Check if pod to service IP connectivity works** [vagrant@k8sMaster ~]$ kubectl exec -it busybox1 wget 10.107.87.106:8800

  Connecting to 10.107.87.106:8800 (10.107.87.106:8800)

  index.html 100% |*******************************| *7 0:00:00 ETA*

Note:

- For more details on ITM tunnel auto-configuration refer, https://docs.opendaylight.org/en/stable-oxygen/submodules/genius/docs/specs/itm-tunnel-auto-config.html
- For details on iptables based kubeproxy implementation in netvirt, refer, https://docs.opendaylight.org/projects/netvirt/en/latest/specs/neon/coe-service-integration.html

CHAPTER

TWO

SETTING UP COE IN VAGRANT

1. **Run vagrant up**
    - If Fedora wants to use libvirtd as the default provider then set it explicitly with –provider=virtualbox or export VAGRANT_DEFAULT_PROVIDER=virtualbox.
2. ssh to the VM as vagrant ssh k8s-master
3. **for each VM, run the following commands:**
    - start the Kubernetes cluster using the following command:

```
vagrant@k8sMaster:~$ sudo kubeadm init --apiserver-advertise-address=192.
↪168.33.11
```

Note: If you get the swap issue, then do the following:

1. open the /etc/systemd/system/kubelet.service.d/10-kubeadm.conf file:

```
$ sudo vi /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

2. add the following line then save and close:

```
Environment="KUBELET_EXTRA_ARGS=--fail-swap-on=false"
```

3. disable swap and restart kubelet:

```
$ sudo swapoff -a
$ sudo systemctl daemon-reload
$ sudo systemctl restart kubelet
```

4. create k8s cluster using:

```
$ sudo kubeadm init --apiserver-advertise-address=192.168.33.11 --
↪ignore-preflight-errors Swap
```

Note: Read the command output in order to use the kubectl command after.

Note: In the minion VMs, you will use the join command instead ex:

```
vagrant@k8sMinion2:~$ sudo kubeadm join --token {given_token} 192.168.
↪33.11:6443
```

- In order to create pods example use the following commands:

```
vagrant@k8sMaster:~$ sudo kubectl create namespace sock-shop

vagrant@k8sMaster:~$ sudo kubectl apply -n sock-shop -f "https://github.
↪com/microservices-demo/microservices-demo/blob/master/deploy/kubernetes/
↪complete-demo.yaml?raw=true"
```

- Check the pods status by executing:

```
vagrant@k8sMaster:~$ sudo kubectl -n sock-shop get pods -o wide
```

2.1 Verification and Troubleshooting

1. Tunnels: a full overlay mesh is established between all three nodes. Each node will have two ports beginning with tun on br-int.:

```
$ vagrant ssh k8s-master
$ sudo ovs-vsctl show
[vagrant@k8sMaster cni]$ sudo ovs-vsctl show
ba282931-cf8e-4440-8982-4ae3ea71e014
    Manager "tcp:192.168.33.11:6640"
    Bridge br-int
        Controller "tcp:192.168.33.11:6653"
        fail_mode: secure
        Port "tun27b98be53d8"
            Interface "tun27b98be53d8"
                type: vxlan
                options: {key=flow, local_ip="192.168.33.11", remote_ip="192.168.
↪33.13"}
        Port "veth39657fe3"
            Interface "veth39657fe3"
                error: "could not open network device veth39657fe3 (No such␣
↪device)"
        Port "tun75d56d08394"
            Interface "tun75d56d08394"
                type: vxlan
                options: {key=flow, local_ip="192.168.33.11", remote_ip="192.168.
↪33.12"}
        Port br-int
            Interface br-int
                type: internal
    ovs_version: "2.8.2"
```

2. **veth ports: each node will have a veth on br-int.**
   - Use the same command for the tunnels verification step.
3. nodes
   - kubectl get nodes:

```
[vagrant@k8sMaster cni]$ kubectl get nodes
NAME          STATUS     ROLES     AGE       VERSION
k8smaster     Ready      master    3d        v1.9.4
k8sminion1    NotReady   <none>    3d        v1.9.4
k8sminion2    NotReady   <none>    3d        v1.9.4
```