CNF deployment on OpenShift

04/22/2020
Sandeep Sharma
ssharma@aarnanetworks.com
The Setup

● ONAP Dublin
  ○ OOM deployment
  ○ Deployed across 6 node K8 cluster on same bare metal server

● Openshift CRC (Red Hat Core ready) VM on same bare metal server
  ○ CRC: A preconfigured Openshift disk image meant for development and testing.
    ■ CPUs: 4
    ■ RAM: 8G
    ■ DISK: 35G
    ■ oc client library used to interact with the Openshift cluster
  ○ Openshift version 4.3.1
  ○ Kubernetes version 1.16.2
The Setup (Continued…)

Onap Dublin 6 node cluster

<table>
<thead>
<tr>
<th>Onap Master</th>
<th>N1</th>
<th>N2</th>
<th>N3</th>
<th>N4</th>
<th>N5</th>
</tr>
</thead>
</table>

1. The openshift cloud registered with ONAP, using onap4k8s API

2. CNF deployed on openshift, using onap4K8s APIs

Core ready container
- Configuration: 16 cpu/64G mem
- Edit Cluster network operator in order to create network attachments. Required by the CNF.
- Installed kubervirt operator and CR

Centos 7 server
Qemu-kvm virtualization
Register Openshift Cluster with ONAP

- Call onap4K8s plugin API in order to upload the kubecfg.

```bash
curl -i -F "metadata=<${payload};type=application/json" -F file=@$HOME/.kube/config -X POST ${base_url}
base_url='http://{ONAPMASTER}:30280/api/multicloud-k8s/v1/v1/connectivity-info'
```

Payload:
```json
{
    "cloud-region": "crc",
    "cloud-owner": "owner",
    "other-connectivity-list": {
        "connectivity-records": [
            { "ssl-initiator": "false",
              "user-name": "kubeadmin",
              "password": "db9Dr-J2csc-8oP78-9sbmf"
            }
        ]
    }
}
```
Prepare CNF Helm Chart

- CNF used: Open source vFW
  - Sink: Container
  - FW: VM
  - Packer generator: VM
- Helm Version: Helm3
- VM on Openshift
  - Kubervirt operator/CRD
  - `oc apply -f kubervirt_operator.yaml`
  - `oc apply -f kubevirt_crd.yaml`
  - Helm chart with heat template of VM
- Specify multiple network requirements by CNFs in Helm chart...
SINK CNF Networks Setup

OAM network: Bridge
CNI plugin

Pod default network port.
CNI: openshiftSDN

SINK

Private network: Bridge
CNI plugin
SINK CNF Networks Setup (Continued…)

- **Openshift (CNO) Cluster network operator**
  - Comes up with default CNI openshiftSDN (or OVN)
  - CNO has capability of adding more CNI plugins.
  - Users can create network attachments by adding CNI.
  - CNO creates networkattachmentDefinition CR (custom resource) for Multus CNI.
  - Multus handles CNI chaining.

- **For Sink we used the bridge CNI as network attachment.**
  - `oc edit networks.operator.openshift.io cluster`

```yaml
spec:
  additionalNetworks:
    - name: unprotected-private-net
      rawCNIConfig: '{
        "cniVersion": "0.3.1",
        "type": "bridge",
        "master": "eth1",
        "ipam": {
          "type": "static"
        }
      }'
  type: Raw
```
• After creating the attachment definition specify the networks in Helm chart annotations:

```yaml
k8s.v1.cni.cncf.io/networks: '[
    {
        "name": "unprotected-private-net",
        "interface": "eth1",
        "ips": "192.168.10.10/24"
    }
],'
```
CNF Deployment on OpenShift

1. POST: CNF Service CSAR
2. Get the Kube cloud related information.
3. Extract CSAR.
4. Validate the Helm charts.
5. Pass the Helm charts to kubernetes cluster.
Onap4K8s APIs (V1)

- Design in SDC
  - Create VSP from the vFW helm package
  - Test and Approve
  - Distribute: This step will upload the package to the K8s plugin

- API
  - curl -i -d @create_rbdefinition.json -X POST http://$NODE_IP:30280/api/multicloud-k8s/v1/v1/rb/definition
  - curl -i --data-binary @vfw_cloudtech_k8s_charts.tgz -X POST http://$NODE_IP:30280/api/multicloud-k8s/v1/v1/rb/definition/test-vfw2/v1/content
  - curl -i -d @create_rbprofile.json -X POST http://$NODE_IP:30280/api/multicloud-k8s/v1/v1/rb/definition/test-vfw2/v1/profile
  - curl -d @create_rbinstance.json http://$NODE_IP:30280/api/multicloud-k8s/v1/v1/instance
Add CAs for local Container registry to OpenShift

- Configure Additional CA
  - Create a configmap in openshift-config NS
  - `oc create configmap registry-config --from-file='myregistrydomain.com:443'=./ca.crt -n openshift-config`
  - ca.crt is the certificate
  - mystegistrydomain.com:443 is the registry name and port number

- Edit the image.config.openshift.io

  `oc edit image.config.openshift.io cluster`

  `spec:`

  ```
  AdditionalTrustedCA:
      name: registry-config
  ```
Run POD as USER ID in Openshift

- Openshift giver unique userd ID to applications.
- Problem: Application might want to run as specified user.
- Solution: Create service accounts in the residing project with specific service context constraints (SCC).

```bash
# oc create ns useanyuid
# oc project useanyuid
# oc create serviceaccount useanyuid
# oc adm policy add-scc-to-user anyuid -z useanyuid --as system:admin
```

In the Deployment of the application specify the serviceaccount,

```
spec:

  serviceAccountName: useanyuid
```
• Direct changes to Openshift Controller nodes is discouraged.
• For a Day-2 kernel configuration change create MachineConfig for the running
  Openshift Container platform.
• Example: set kernel loglevel

```yaml
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
name: 99_openshift-machineconfig_master-kargs
spec:
  kernelArguments:
  - 'loglevel=7'
```

EOF
THANK YOU