ONAP OOM deployment - Set up Kubernetes cluster in Ubuntu

- Pre-requisite
  - VMs Info

    For example, The virtual machines is as follows

    10.12.26.59 Memory 128G cup16 core storage 156G k8s-node1 node 1

    10.12.26.60 Memory 128G cup16 core storage 156G k8s-node2 node 2

    10.12.26.61 Memory 128G cup16 core storage 156G k8s-node3 node 3

    10.12.26.58 Memory 8G cpu 8 core storage 97G k8s-store storage server

    docker private server address: 10.12.26.4:10001

  - Copy deployment scripts and resources

    master1 ---> 10.12.26.59 any directory

    master2 ---> 10.12.26.60 any directory

    master3 ---> 10.12.26.61 any directory

master1/kube/shell/master_nfs_node.sh ---> 10.12.26.58 any directory

- Modify host name (all nodes)

  vi /etc/hostname

  hostname xxx

  /etc/init.d/hostname.sh start #effective immediately

  For example: k8s-node1, k8s-node2, k8s-node3

- Add default route (all nodes)

  For example: vi /etc/rc.local

  ```
  route add default gw 10.12.26.1

  exit 0
  ```

- Add static ip (all nodes)

  For example: vi /etc/network/interfaces

  ```
  auto lo

  iface lo inet loopback


  auto ens3

  iface ens3 inet static

  address 10.12.26.59

  netmask 255.255.255.0

  gateway 10.12.26.1


  # Source interfaces
  ```

```
# Please check /etc/network/interfaces.d before changing this file

# as interfaces may have been defined in /etc/network/interfaces.d

# See LP: #1262951
```

- Start the nfs cluster on the storage server (10.12.26.58)

  ```
  ./master_nfs_node.sh 10.12.26.59 10.12.26.60 10.12.26.61
  ```

- Purge processes that occupy port 3306 (all nodes)

  The virtual machine we provided may have mysql installed and running by default, so stop mysql and release port 3306

  ```
  service mysql stop
  ```

- Modify ubuntu APT source (all nodes)

  ```
  vi /etc/apt/sources.list
  ```

- APT Install the following softwares (all nodes)

  ```
  apt-get update

  apt-get install -y unzip

  apt-get install build-essential libssl-dev libffi-dev python-dev gcc -y

  apt-get install -y keepalived

  apt-get install -y haproxy

  apt-get install -y sshpass
  ```

- Modify the docker configuration and use the company's image accelerator

  ```
  vi /lib/systemd/system/docker.service
  ```

```
# the default is not to use systemd for cgroups because the delegate issues still

# exists and systemd currently does not support the cgroup feature set required

# for containers run by docker

ExecStart=/usr/bin/dockerd -H fd:// --insecure-registry 10.12.26.4:10001 --insecure-registry 10.12.26.4:10003
```

- Confirm whether HAProxy is working (all nodes)

  service haproxy status

  If it is abnormal, it may be caused by the occupied port, such as port 3306 of mysql, execute the following command to solve

  ```
  service mysql stop

  service haproxy restart
  ```

- Pull k8s cluster related images and tag (all nodes)

  1. Pull the image and execute the following script

     ```
     cd $HOME/master1/kube/shell

     ./pull_images.sh
     ```

  2. Mark the mirror and execute the following script

     ```
     cd $HOME/master1/kube/shell

     ./docker_tag_modify.sh
     ```

- ssh logs in to each node remotely and exits.

After logging in once, the SSHPass operation can be performed normally (node 1)

```
ssh root@10.12.26.60

exit

ssh root@10.12.26.61

exit
```

- Deployment stage
  - Execute the init.sh script on node 1:

    master1/kube/shell/init.sh

    This script completes keepalived configuration, haproxy configuration, k8s configuration, /etc/hosts configuration, loading kernel ipvs module, starting nfs client and other operations

    E.g:

```
cd $HOME/master1/kube/shell

./init.sh 10.12.26.200 10.12.26.59 10.12.26.60 10.12.26.61
k8s-node1 k8s-node2 k8s-node3 ens3 ens3 ens3
10.12.26.58 k8s-store
```

Parameter Description:

10.12.26.200 is the virtual IP provided by keepalived for the cluster

10.12.26.59 node 1 ip

10.12.26.60 node 2 ip

10.12.26.61 node 3 ip

k8s-node1 node 1 machine name

k8s-node2 node 2 machine name

k8s-node3 node 3 machine name

ens3 NIC of node 1, used to configure keepalived

ens3 NIC of node 2 for configuring keepalived

ens3 NIC of node 3, used to configure keepalived

10.12.26.58 storage server ip

k8s-store storage server machine name

- Execute the init.sh script on node 2:

  ```
  master2/kube/shell/init.sh
  ```

  Node 2's init.sh is basically the same as node 1, except the configuration of Keepalived, node 1 as Keepalived's master node, and node 2 as Keepalived's child node, the configuration is slightly different.

  The execution approach is the same as node 1

- Execute the init.sh script on node 3:

  ```
  master3/kube/shell/init.sh
  ```

- Start the master.sh script on node 1:

  ```
  master1/kube/shell/master.sh
  ```

  This script is used to start k8s of node 1, deploy related components, and remotely copy some resource files from node 1 to node 2, node 3

  E.g:

  ```
  ./master.sh 10.12.26.59 10.12.26.60 10.12.26.61 k8s-node1 k8s-node2 k8s-node3 123456 123456 123456
  ```

  Parameter Description:

10.12.26.59 node 1 ip

10.12.26.60 node 2 ip

10.12.26.61 node 3 ip

k8s-node1 node 1 machine name

k8s-node2 node 2 machine name

k8s-node3 node 3 machine name

123456 The root password of node 1

123456 The root password of node 2

123456 The root password of node 3

After the script is executed, the k8s of node 1 is installed, the console will print out the registration-related information, retain this information, and use the k8s of other nodes to register to this node, as follows:

```
kubeadm join cluster.kube.com:16443 --token
7gix23.46nbslwpawg5wtc3 \

    --discovery-token-ca-cert-hash
sha256:bb916d3ebf1e8001530073a53c1bbcb5309d4fb06
9e74930031d365483848c09 \

    --experimental-control-plane
```

- Perform the previous registration command on node 2 and node 3 to register to the k8s cluster

  The registration command can also be obtained through the following command:

```
kubeadm token create --print-join-command
```

- Perform the following configuration on each child node to remove stains

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

By default, each node is a master of k8s, and components such as pods cannot be installed. After executing this command, the restrictions can be removed.

- Run helm.sh on node 1 and install the helm plugin
- OOM Deployment

0. Get OOM code

```
> git clone -b <BRANCH> http://gerrit.onap.org/
r/oom --recurse-submodules

> cd oom/kubernetes
```

Description:

1. The official version is as follows:

   4.0.0-ONAP for Dublin

   5.0.1-ONAP for El Alto

2. The A&AI and robot modules are separate sub-modules, which need to be downloaded separately

   git clone -o gerrit --recurse-submodules ...

It can also be downloaded via Github: https://github.com/onap/oom.git, select the elalto branch

1. Modify and adjust OOM code

In the downloaded OOM code, some places need to be modified, such as the mirror prefix, which should be changed to our own warehouse address, and some mirror addresses are directly written to death, and they must be modified individually.

In addition, except for the log module, other modules do not use nfs shared storage. In order to avoid some potential exceptions, the pods of each module need to be deployed on the same node, so the affinity of deployment, statefulset, job and other components must be modified.

2. Install helm extension

   The helm extension plug-in is provided in the community OOM code for sub-module deployment, or you can directly use the native helm without using the extension plugin

   ```
   > cp -R kubernetes/helm/plugins/ ~/.helm
   ```

3. Compile OOM code

   A Makefile script is provided in the community oom code, which is used to compile the oom code and store it in the local helm's charts warehouse

   The so-called compilation is to use the onap module as the core module, and the onap dependent submodules (aaf, aai, dmaap, etc.) are made into tgz compressed packages and placed in the onap dependency directory (charts)

   There are also dependencies between submodules. For example, the aaf module depends on the common module. When the compile command is executed, the tgz compressed package of the common module is first placed in the aaf dependency directory, and then the aaf module is made into a tgz compressed package. In the onap dependency directory

   The specific operation of compilation is as follows:

   Go to the oom/kubernetes directory and execute the make all command

   If no error is reported, the generated charts package will be placed in the local warehouse of helm. Run helm search onap -l to view the charts package of the local warehouse, as follows

   ```
   root@k8s-node1:~/deploy/oom/debug/kubernetes/aaf/charts# helm search onap -l
   ```

```
NAME                    CHART
VERSION                 APP VERSION
  DESCRIPTION

local/onap              5.0.0
El Alto                 Open Network
Automation Platform (ONAP)

local/aaf               5.0.0
El Alto                 ONAP Application
Authorization Framework

local/aai               5.0.0
El Alto                 ONAP Active and
Available Inventory

local/cassandra
5.0.0                   El Alto
ONAP cassandra

local/cds               5.0.0
El Alto                 ONAP Controller
Design Studio (CDS)

local/clamp             5.0.0
El Alto                 ONAP Clamp

local/consul            5.0.0
El Alto                 ONAP Consul
Agent

local/contrib           5.0.0
El Alto                 ONAP optional
tools

local/dcaegen2
5.0.0                   El Alto
ONAP DCAE Gen2

local/dmaap             5.0.0
El Alto                 ONAP DMaaP
components
```

```
local/esr                     5.0.0
El Alto                       ONAP External
System Register

local/log                     5.0.0
El Alto                       ONAP Logging
ElasticStack

local/msb                     5.0.0
El Alto                       ONAP
MicroServices Bus

local/multicloud              5.0.0
El Alto                       ONAP multicloud
broker

local/oof                     5.0.0
El Alto                       ONAP
Optimization Framework

local/policy                  5.0.0
El Alto                       ONAP Policy
Administration Point

local/portal                  5.0.0
El Alto                       ONAP Web Portal

local/postgres                5.0.0
El Alto                       ONAP Postgres
Server

local/robot                   5.0.0
El Alto                       A helm Chart for
kubernetes-ONAP Robot

local/sdnc-prom
5.0.0                         El Alto
ONAP SDNC Policy Driven Ownership
Management
```

```
local/so                5.0.0
El Alto                 ONAP Service
Orchestrator

local/vid               5.0.0
El Alto                 ONAP Virtual
Infrastructure Deployment
```

If we modify the OOM code, we need to execute the compile command before deploying

4. Deploy OOM

There are multiple deployment methods, you can use the helm extension plug-in deployment, you can also use the original helm function deployment.

0. Use the helm extension to deploy

```
> helm deploy dev local/onap --namespace
onap
```

dev is the release name of this deployment

ONAP is the specified namespace, if there is no such namespace in k8s, it will automatically create a

1. Use native helm deployment

```
> helm install local/onap -n dev --namespace
onap
```

This approach will deploy all the enabled modules in one release at a time, which is not conducive to maintenance.

You can modify the onap/values.yaml file and set the module to be deployed to enable to deploy a module separately

```
…

aaf:
  enabled: true

aai:

  enabled: false

…
```

For example, to deploy the aaf module, first set aaf enabled to true, and then execute the helm deployment command

```
> helm install local/onap -n aaf --namespace onap
```

-n aaf specifies the release name of this deployment

When deploying the aai module, set the aai's enabled to true, and the other modules' enabled to false

```
…

aaf:

  enabled: false

aai:

  enabled: true

…
```

```
> helm install local/onap -n aai --namespace onap
```

Other modules can be deduced by analogy. In this way, the submodules are deployed one by one in the native way of helm.

You can view the helm deployment status by the following instructions

```
root@k8s-node1:~# helm list



NAME
REVISION
UPDATED
                STATUS
CHART                         APP
VERSION
NAMESPACE

aaf                           1
  Wed Mar 25 23:35:09 2020

DEPLOYED
onap-5.0.0                    El Alto
                onap

aai                           1
  Wed Apr  1 10:44:06 2020
                DEPLOYED
onap-5.0.0                    El Alto
                onap

appc                          1
  Thu Apr  2 12:26:08 2020
                DEPLOYED
onap-5.0.0                    El Alto
                onap

cds                           1
  Sun Apr  5 13:01:58 2020
                DEPLOYED
onap-5.0.0                    El Alto
                onap

clamp                         1
  Thu Mar 26 00:27:44 2020
```

```
                DEPLOYED
onap-5.0.0                  El Alto
                onap

consul                      1
  Thu Mar 26 12:47:53 2020
                DEPLOYED
onap-5.0.0                  El Alto
                onap

contrib                     1
  Thu Mar 26 00:32:14 2020
                DEPLOYED
onap-5.0.0                  El Alto
                onap

dcae                        1
  Mon May 11 14:54:04 2020

DEPLOYED
onap-5.0.0                  El Alto
                onap

dev                         1
  Wed Mar 25 23:31:18 2020

DEPLOYED
onap-5.0.0                  El Alto
                onap

dmaap                       1
  Thu May  7 15:50:10 2020
                DEPLOYED
onap-5.0.0                  El Alto
                onap

esr                         1
  Thu Mar 26 12:19:29 2020
                DEPLOYED
```

```
onap-5.0.0                El Alto
             onap

log                       1
  Sun Apr 26 17:14:50 2020
             DEPLOYED
onap-5.0.0                El Alto
             onap

msb                       1
  Thu Mar 26 12:40:18 2020
             DEPLOYED
onap-5.0.0                El Alto
             onap

mutcld                    1
  Thu Mar 26 17:51:22 2020
             DEPLOYED
onap-5.0.0                El Alto
             onap

oof                       1
  Thu Mar 26 16:27:25 2020
             DEPLOYED
onap-5.0.0                El Alto
             onap

policy                    1
  Mon May 11 14:33:51 2020

DEPLOYED
onap-5.0.0                El Alto
             onap

portal                    2
  Thu Mar 26 17:35:05 2020
             DEPLOYED
onap-5.0.0                El Alto
             onap
```

```
robot                       1
  Thu Apr  2 12:11:49 2020
               DEPLOYED
onap-5.0.0                  El Alto
               onap

sdc                         2
  Mon May 11 16:15:17 2020

DEPLOYED
onap-5.0.0                  El Alto
               onap

sdnc                        1
  Mon Mar 30 11:48:08 2020
               DEPLOYED
onap-5.0.0                  El Alto
               onap

so                          1
  Thu Apr  2 12:20:06 2020
               DEPLOYED
onap-5.0.0                  El Alto
               onap

vid                         1
  Wed Apr  1 11:25:57 2020
               DEPLOYED
onap-5.0.0                  El Alto
               onap
```

There is a note here, there are some common components in the onap module, under the directory oom/kubernetes/onap/templates

If you use this helm native sub-module deployment method, you need to extract the public components in this directory and put them in separate modules, and deploy in advance to avoid repeated deployment of these public components.

2. Customised deployment

   You can put the customised configuration in a separate file, and specify this file with -f during deployment

   ```
   > helm deploy dev local/onap --namespace onap -f onap/resources/overrides/openstack.yaml --timeout 900
   ```

   The content in the customised file will replace the corresponding default configuration in the OOM code

3. Debugging

   Before deployment, you can debug and deploy the source file, the command is as follows

   ```
   > helm install local/onap -n aai --dry-run --debug> /tmp/install.log
   ```

   You can view the source file information of the A&AI module through install.log, including the template content, the image used, and which components such as secret configmap service deployment statefulset are defined

5. View deployment

Check pod status:

```
root@nsk8s-node1:/home/ubuntu/test/kubernetes# kubectl get pod -n onap -o wide

NAME                              READY   STATUS
RESTARTS   AGE   IP              NODE
NOMINATED NODE   READINESS GATES
```

```
dev-aaf-aaf-cass-67fbc9bfc8-plxcg      1/1
Running     0         48m   100.119.62.183
nsk8s-node1   <none>          <none>

dev-aaf-aaf-cm-567f77f74d-8wnl5        1/1
Running     0         48m   100.119.62.154
nsk8s-node1   <none>          <none>

dev-aaf-aaf-fs-665bdb44f9-54r7q        1/1
Running     0         48m   100.119.62.181
nsk8s-node1   <none>          <none>

dev-aaf-aaf-gui-5664788c89-lg646       1/1
Running     0         48m   100.119.62.185
nsk8s-node1   <none>          <none>

dev-aaf-aaf-locate-8569676b6f-28v9h    1/1
Running     0         48m   100.119.62.177
nsk8s-node1   <none>          <none>

dev-aaf-aaf-oauth-6998487496-4lsjc     1/1
Running     0         48m   100.119.62.165
nsk8s-node1   <none>          <none>

dev-aaf-aaf-service-67fb8cb9d9-55hwn   1/1
Running     0         48m   100.119.62.153
nsk8s-node1   <none>          <none>

dev-aaf-aaf-sms-5db54d6b98-w697r       1/1
Running     0         48m   100.119.62.130
nsk8s-node1   <none>          <none>

dev-aaf-aaf-sms-preload-rc65p          0/1
Completed   0         48m   100.119.62.132
nsk8s-node1   <none>          <none>

dev-aaf-aaf-sms-quorumclient-0         1/1
Running     0         48m   100.119.62.131
nsk8s-node1   <none>          <none>
```

```
dev-aaf-aaf-sms-quorumclient-1         1/1
Running     0          47m   100.119.62.157
nsk8s-node1   <none>           <none>

dev-aaf-aaf-sms-quorumclient-2         1/1
Running     0          47m   100.119.62.186
nsk8s-node1   <none>           <none>

dev-aaf-aaf-sms-vault-0                2/2
Running     1          48m   100.119.62.151
nsk8s-node1   <none>           <none>

dev-aaf-aaf-sshsm-distcenter-5hg52     0/1
Completed   0          48m   100.119.62.159
nsk8s-node1   <none>           <none>

dev-aaf-aaf-sshsm-testca-bj675         0/1
Completed   0          48m   100.119.62.188
nsk8s-node1   <none>           <none>

dev-cassandra-cassandra-0              1/1
Running     0          48m   100.119.62.140
nsk8s-node1   <none>           <none>

dev-cassandra-cassandra-1              1/1
Running     0          45m   100.119.62.161
nsk8s-node1   <none>           <none>

dev-cassandra-cassandra-2              1/1
Running     0          43m   100.119.62.147
nsk8s-node1   <none>           <none>

dev-mariadb-galera-mariadb-galera-0    1/1
Running     0          48m   100.119.62.138
nsk8s-node1   <none>           <none>

dev-mariadb-galera-mariadb-galera-1    1/1
Running     0          47m   100.119.62.179
nsk8s-node1   <none>           <none>
```

```
dev-mariadb-galera-mariadb-galera-2    1/1
Running     0          46m   100.119.62.133
nsk8s-node1   <none>           <none>
```

Login a container

```
> kubectl exec -it xxx /bin/bash -n onap
```

View service and exposed port information

```
root@nsk8s-node1:/home/ubuntu/test/
kubernetes# kubectl get svc -n onap

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

aaf-cass         ClusterIP   None              <none>
7000/TCP,7001/TCP,9042/TCP,9160/TCP
50m

aaf-cm           NodePort    10.106.87.10
<none>        8150:31114/TCP
50m

aaf-fs           NodePort    10.104.210.7
<none>        8096:31115/TCP
50m

aaf-gui          NodePort    10.101.244.203
<none>        8200:31113/TCP
50m

aaf-hello        NodePort    10.98.5.198
<none>        8130:31119/TCP
50m

aaf-locate       NodePort    10.106.169.8
<none>        8095:31111/TCP
50m
```

```
aaf-oauth       NodePort    10.103.198.198
<none>        8140:31112/TCP
50m

aaf-service      NodePort    10.109.195.115
<none>        8100:31110/TCP
50m

aaf-sms          ClusterIP   10.109.81.91
<none>        10443/TCP
50m

aaf-sms-db       ClusterIP   10.109.161.244
<none>        8200/TCP
50m

cassandra        ClusterIP   None            <none>
7000/TCP,7001/TCP,7199/TCP,9042/TCP,9160/
TCP,61621/TCP   50m

mariadb-galera   ClusterIP   None
<none>        3306/TCP
50m
```

6. Uninstall a module

helm list to see which releases are currently deployed through helm

```
root@nsk8s-node1:/home/ubuntu/test/
kubernetes# helm list




NAME
REVISION                    UPDATED
                            STATUS
CHART                                    APP
VERSION                     NAMESPACE
```

```
dev                                          1
                 Mon May 18 17:21:39 2020
                 FAILED
onap-5.0.0                                   El
Alto                            onap

dev-aaa                                      1
                 Mon May 18 17:21:39 2020
                 DEPLOYED
aaa-5.0.0                                    El
Alto                            onap

dev-aaf                                      1
                 Mon May 18 17:21:40 2020
                 DEPLOYED
aaf-5.0.0                                    El
Alto                            onap

dev-cassandra
1                               Mon May 18
17:21:47 2020
DEPLOYED                        cassandra-5.0.0
                 El Alto
onap

dev-mariadb-galera
1                               Mon May 18
17:21:48 2020
DEPLOYED                        mariadb-
galera-5.0.0                    El Alto
onap
```

Use the following command to delete the residual data volume related to mariadb

```
> kubectl delete pvc $(kubectl get pvc -n onap | grep
dev-mariadb-galera | awk -F " " '{print $1}') -n onap
```

Some modules will map a directory on the host machine through the data volume. If you want to uninstall completely, you also need to delete the file directory mapped on the host machine, because some modules cannot be deployed again if they are not cleanly deleted.

Usually the mapping directory is in /dockerdata-nfs

```
rm -rf /dockerdata-nfs/xxx
```

7. Uninstall the Kubernetes environment

```
kubeadm reset -f



modprobe -r ipip

lsmod

rm -rf ~/.kube/

rm -rf /etc/kubernetes/

rm -rf /etc/systemd/system/kubelet.service.d

rm -rf /etc/systemd/system/kubelet.service

rm -rf /usr/bin/kube*

rm -rf /etc/cni

rm -rf /opt/cni

rm -rf /var/lib/etcd

rm -rf /var/etcd

ipvsadm --clear

iptables --flush

iptables -tnat --flush

ip6tables --flush

ip6tables -tnat --flush
```

```
systemctl restart docker.service
```