CONFIGURATION OF KUBERNETES CLUSTER

Cognitive Assistant for Networks (CAN) Release 5.5

JULY 6, 2021

AVANSEUS TECHNOLOGY PVT. LTD.

REVISION HISTORY

Version	Date	Change description	Created by	Updated by	Reviewed by
V 1.0	July, 2021	Initial Release	Hemanth/Umesh	Sandeep Singh	Chiranjib

avanseus

Table of Contents

1. Prerequisites

- We assume that the Docker and Kubernetes have been already installed successfully on the master and all the worker nodes. Helm should be installed only in the master node.
- Root access to the servers is mandatory.

2. Kubernetes Cluster Setup

2.1. Steps to set up Cluster for Kubernetes Master Node

Follow the below steps only in master node:

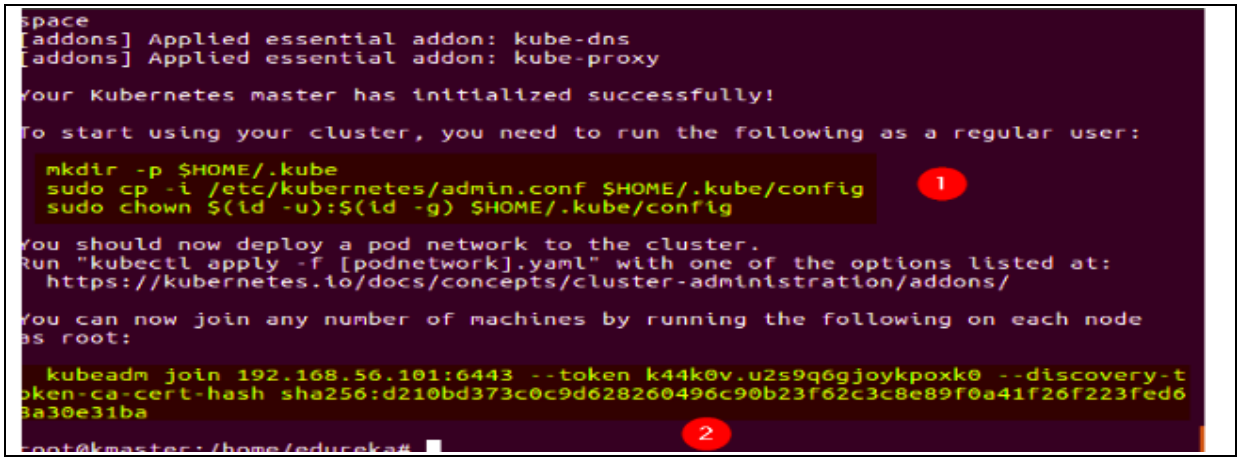
Step 1: Initializing Cluster from Master Node

Initialize Kubernetes cluster from the master machine. Run the below command based on your choice of CNI between calico or flannel. Calico is the most popular and preferred option.

```
$sudo su
$kubeadm init --apiserver-advertise-address=<private_ip-address-of-master> --pod-network-cidr=<CNI>
```

- **<private_ip-address-of-master>**: Private ip of the master need to be used in the command.
- **<CNI>:**
  - If you want to use Calico CNI (Preferred) then use CNI value as **192.168.0.0/16.**
  - If you want to use Flannel CNI then use CNI value as **10.244.0.0/16.**

You will get the output as shown in the below screenshot:



Now, the master has been initialized successfully. Note down few important points before you proceed with the cluster setup. There are few more steps that need to be followed in the master node.

Important Note:

Make a note of the join command (Marked as 2 in the above snapshot). This command will be used in section 2.2 while we join worker nodes into this newly created cluster.

Step 2. Enabling kubectl command

To start using the cluster, you need to use the following command as a regular user. Execute the following commands. (This step is marked as 1 in the above screenshot)

```
$ mkdir -p $HOME/.kube
$ sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
$ sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

This will enable to use kubectl from the CLI. To verify kubectl, execute the below command:

```
$kubectl version
```

```
[ec2-user@ip-172-31-25-155 ~]$
[ec2-user@ip-172-31-25-155 ~]$ kubectl version
Client Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.0", GitCommit:"af46c47ce925f4c4ad5cc8d1fca46c7b77d13b38", GitTreeState:"clean",
BuildDate:"2020-12-08T17:59:43Z", GoVersion:"go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"20", GitVersion:"v1.20.0", GitCommit:"af46c47ce925f4c4ad5cc8d1fca46c7b77d13b38", GitTreeState:"clean",
BuildDate:"2020-12-08T17:51:19Z", GoVersion:"go1.15.5", Compiler:"gc", Platform:"linux/amd64"}
[ec2-user@ip-172-31-25-155 ~]$
```

Use the below command to check the pods status:

```
$kubectl get pods -o wide --all-namespaces
```

You can see that some of the pod status is shown as "**Pending**", this is because CNI for the pod Network is not installed yet.

```
edureka@kmaster: ~
edureka@kmaster:~$ kubectl get pods -o wide --all-namespaces
AMESPACE     NAME                                READY     STATUS    RESTARTS   AGE       IP
ube-system   etcd-kmaster                        1/1       Running   0          4s        192.168.56.101
ube-system   kube-apiserver-kmaster              1/1       Running   0          4s        192.168.56.101
ube-system   kube-controller-manager-kmaster     1/1       Running   0          4s        192.168.56.101
ube-system   kube-dns-86f4d74b45-ggg8z           0/3       Pending   0          12m       <none>
ube-system   kube-proxy-85tp2                    1/1       Running   0          12m       192.168.56.101
ube-system   kube-scheduler-kmaster              1/1       Running   0          4s        192.168.56.101
edureka@kmaster:~$
```

Step 3. Installation of CNI for the pod network

There are two choices of CNI:

1. Calico (Preferred)
2. Flannel

To use Calico CNI for the pod network, execute the below command:

```
$kubectl apply -f https://docs.projectcalico.org/manifests/calico.yaml
```

To use Flannel CNI for the pod network, execute the below command:

```
$kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
```

After applying the CNI pod network, check the status of each pod using the below command:

```
$kubectl get pods -o wide --all-namespaces
```

```
edureka@kmaster: ~
edureka@kmaster:~$ kubectl get pods -o wide --all-namespaces
NAMESPACE     NAME                                READY     STATUS    RESTARTS   AGE       IP
kube-system   calico-etcd-b46dk                   1/1       Running   0          2m        192.168.56.101
kube-system   calico-kube-controllers-5d74847676-lrhvc   1/1       Running   0          2m        192.168.56.101
kube-system   calico-node-n9v8k                   2/2       Running   0          2m        192.168.56.101
kube-system   etcd-kmaster                        1/1       Running   0          1m        192.168.56.101
kube-system   kube-apiserver-kmaster              1/1       Running   0          1m        192.168.56.101
kube-system   kube-controller-manager-kmaster     1/1       Running   0          1m        192.168.56.101
kube-system   kube-dns-86f4d74b45-ggg8z           3/3       Running   0          23m       192.168.189.1
kube-system   kube-proxy-85tp2                    1/1       Running   0          23m       192.168.56.101
kube-system   kube-scheduler-kmaster              1/1       Running   0          1m        192.168.56.101
```

You can clearly see that all the pods are up and have status as "**Running**".

All the steps in the master have been completed.

2.2. Steps to set up Cluster for Kubernetes Worker Nodes

Get your worker node to join the cluster. (This is probably the only step that you will be doing on the worker node, after installing kubernetes on it).

Use the join command output, which we noted previously while configuring the cluster, in the master node.

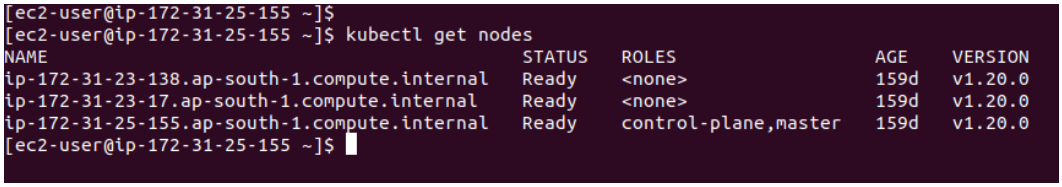
Note: The below join command is an example of how the join command looks. Please use the join command which was noted during the cluster configuration of the master node.

```
$sudo kubeadm join 172.31.25.155:6443 --token dmdm0d.nrkard0g3vs80549 --discovery-token-ca-cert-hash sha256:2ac8343de35f6316234bc14ec937e706e8f7fbcaea6703503b591de714020078
```

After all the setup has been done, verify whether the cluster is ready by executing the below command in master node:

```
$kubectl get nodes
```

The status of all the nodes should be "**Ready**" as shown in the below screenshot:



The kubernetes cluster is setup. If you face any issues in terms of cluster setup like core-DNS not getting resolved, worker node not able to join the cluster, unhealthy worker node etc., then try reinstalling the cluster from scratch. Deletion of clusters is discussed in next section.

3. Troubleshooting the Kubernetes Cluster creation

This section needs to be followed if you face any issues or unseen challenges while creating the kubernetes cluster or if joining worker nodes to the cluster create issues. This section helps in purging all the kubernetes cluster related dependencies before setting up a new cluster.

3.1. Steps to follow in Kubernetes Master Node

Execute the below commands to reset the complete cluster in master node:

```
$kubeadm reset -f
$rm -rf /etc/cni /etc/kubernetes /var/lib/dockershim/var/lib/etcd
/var/lib/kubelet /var/run/kubernetes ~/.kube/*
```

3.2. Steps to follow in Kubernetes Worker Nodes

Execute the below commands to reset the complete cluster in worker nodes:

```
$rm -rf /etc/cni /etc/kubernetes /var/lib/dockershim var/lib/etcd /var/lib/kubelet /var/run/kubernetes ~/.kube/*
$systemctl daemon-reload && systemctl restart kubelet
```

Once the above commands are executed, start setting up the cluster again. To set up the cluster again, refer this document from start.