Lab 12. Install Kubernetes with Dashboard on OL 7

Learning Objectives

- Sequence 1. Create clone of tester1 VM
- Sequence 2. Setting up server, tester1 and tester2 for K8s
- Sequence 3. Set up K8s master node; server.
- Sequence 4. Deploy the Kubernetes Dashboard
- Sequence 5. Set up K8s worker nodes; tester1 and tester2.
- Sequence 6. Troubleshooting and Reset

Pre Requisite

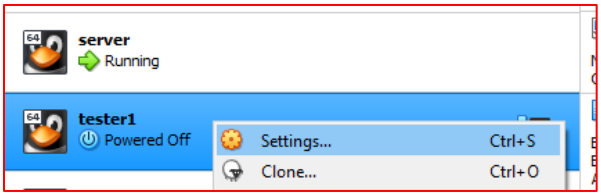
- An Oracle Linux 7 VM to install K8s and the required software.

Configuration	master Node	worker Node1	worker Node2
hostname	server.example.com	tester1.example.com	tester2.example.com
OS	CentOS 7/OL7	CentOS 7/OL7	CentOS 7/OL7
IP Address	10.10.0.100	10.10.0.101	10.10.0.102
rpms required	python3	python3	python3

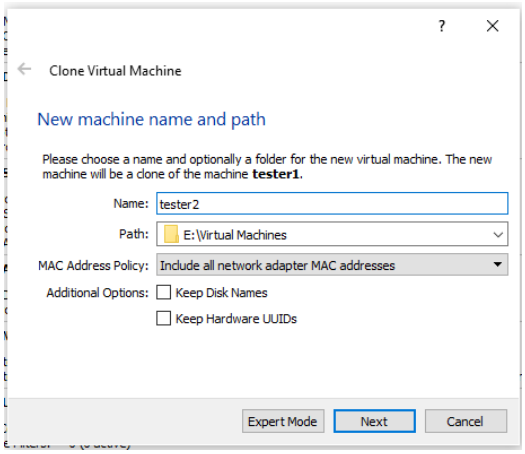
- Internet access on all VMs.
- Remove all Docker Images and Containers.

Sequence 1. Create clone of tester1 VM

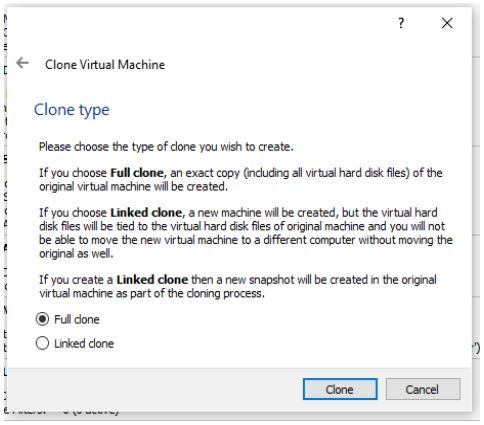
1. Shutdown tester1 VM and Create a clone of the VM.
   a. Right click on the VM Name and select clone.



   b. Change the Details as given in the screen shot

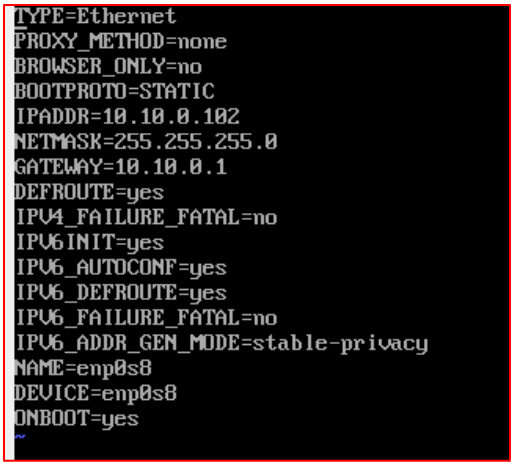


c. In the next screen keep default and click on clone button



2. Once the clone is created, start tester2 and make following changes
   a. Change the hostname in ***/etc/hostname*** to ***tester2.example.com***

   b. Change the IP address in ***/etc/sysconfig/network-scripts/ifcfg-enp0s8*** to 10.10.0.102



c. Reboot the VM, ***tester2***.

Sequence 2. Install required packages on all three VMs

1. Start all three VMs; ***server***, ***tester1*** and ***tester2*** and login as root user.
2. Setup yum repositories for required packages on all three VMs. Configure the Kubernetes repositories by manually creating repo file. Separate text file is available to save time and avoid errors due to control characters.

   # vi /etc/yum.repos.d/kubernetes.repo

   [kubernetes]

   name=Kubernetes

   baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64

   enabled=1

   gpgcheck=1

   repo_gpgcheck=1

   gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
   https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg

   Note – The *gpgkey* parameter is in one line with urls as given in screen shot.

3. On ***teste1*** and ***tester2***, add these to Existing Repo file /etc/yum.repos.d/oracle-linux-ol7.repo

   # vi /etc/yum.repos.d/oracle-linux-ol7.repo

   [ol7_developer]

   name=Oracle Linux $releasever Development Packages ($basearch)

   baseurl=https://yum.oracle.com/repo/OracleLinux/OL7/developer/$basearch/

   gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle

gpgcheck=1
enabled=1

[ol7_developer_EPEL]
name=Oracle Linux $releasever Development Packages ($basearch)
baseurl=https://yum.oracle.com/repo/OracleLinux/OL7/developer_EPEL/$basearch/
gpgkey=file:///etc/pki/rpm-gpg/RPM-GPG-KEY-oracle
gpgcheck=1
enabled=1

4. Run the package update command on all three VMs; server, tester1 and tester2. Note that it may take 15 to 20 Min. If the YUM package installer is busy, don't panic and interrupt it. This might be due to auto update running.

   # yum update -y

5. Disable swap on all three VMs. K8s will not work with swap on. For permanently disable swap, *comment out the last line* in /etc/fstab.

   # vi /etc/fstab

6. Enable Net packet filter with following command on ***three VMs***.

   # modprobe br_netfilter
   # echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables

```
[root@server ~]# setenforce 0
[root@server ~]# sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
[root@server ~]# firewall-cmd --permanent --add-port={10250-10252,10255,2379,2380,6443}/tcp
success
[root@server ~]# firewall-cmd --reload
success
[root@server ~]# modprobe br_netfilter
[root@server ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
[root@server ~]#
```

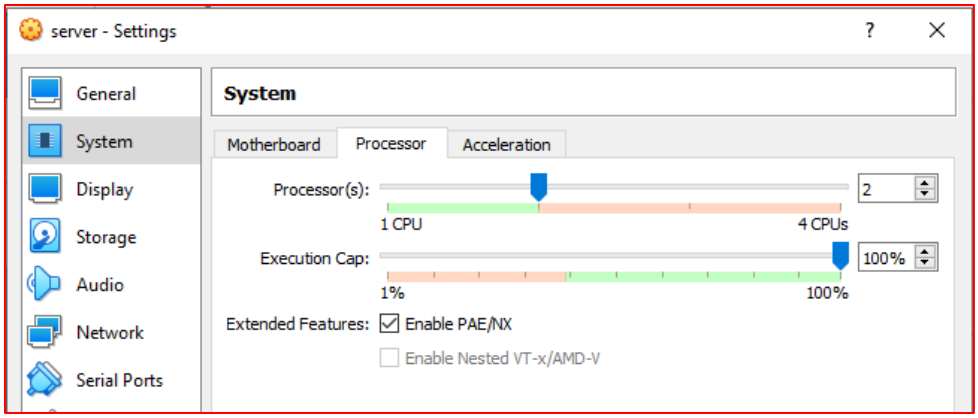
7. Verify your /etc/hosts file on all three VMs

```
[root@server ~]# cat /etc/hosts file
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.10.0.100     server  server.example.com
10.10.0.101     tester1 tester1.example.com
10.10.0.102     tester2 tester2.example.com
cat: file: No such file or directory
[root@server ~]#
```

8. Shutdown all three VMs.
   # poweroff

Sequence 3. Setting up master Node, server

1. Configure the server with two CPUs. Open the settings of server VM while it is in "poweroff" mode and click on processor tab under System as given in following screen shot.



2. Start the server VM and login as root.
3. Set the firewall rules with following command.

   # firewall-cmd --permanent --add-port={10248,10250-10252,10255,2379,2380,6443}/tcp
   # firewall-cmd --reload
4. Run the following command to install kubeadm.

   # yum install kubeadm -y
5. Change some settings for Docker and start & enable kubectl and docker service

   # cat > /etc/docker/daemon.json <<EOF
   {"exec-opts": ["native.cgroupdriver=systemd"]}
   EOF

   # systemctl restart docker
   # systemctl enable --now kubelet

```
[root@server ~]# cat > /etc/docker/daemon.json <<EOF
> {"exec-opts": ["native.cgroupdriver=systemd"]}
> EOF
[root@server ~]# cat /etc/docker/daemon.json
{"exec-opts": ["native.cgroupdriver=systemd"]}
[root@server ~]# systemctl restart docker
[root@server ~]# systemctl enable --now kubelet
[root@server ~]#
```

   Service will not start yet. If you check /var/log/messages, it will complain missing /var/lib/kubelet/config.yaml. Don't Worry.
6. Run the following commands to pull images, initialize and setup kubernetes server.

   # kubeadm config images pull
   # kubeadm init \
   --apiserver-advertise-address=10.10.0.100 \
   --pod-network-cidr=10.244.0.0/16 \

   Output of above command would be something like

```
[root@server ~]# kubeadm init \
> --apiserver-advertise-address=10.10.0.100 \
> --pod-network-cidr=10.244.0.0/16 \
>
[init] Using Kubernetes version: v1.22.0
[preflight] Running pre-flight checks
        [WARNING Firewalld]: firewalld is active, please ensure ports [
        [WARNING FileExisting-tc]: tc not found in system path
[preflight] Pulling images required for setting up a Kubernetes cluster
```

7. Execute the following commands to use the cluster as root user.

   # mkdir -p $HOME/.kube

   # cp -i /etc/kubernetes/admin.conf $HOME/.kube/config

   # chown $(id -u):$(id -g) $HOME/.kube/config

```
To start using your cluster, you need to run the following as a regular user:

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

You should now deploy a pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options listed at:
  https://kubernetes.io/docs/concepts/cluster-administration/addons/

Then you can join any number of worker nodes by running the following on each as root:

kubeadm join 10.10.0.100:6443 --token ddynn0.4g0y958pwtg8lude \
    --discovery-token-ca-cert-hash sha256:9c0374ebdaa08baf49a2568570a02f7fce1a38bee11c3a1d3c157a5d638fb8eb
[root@server ~]#
```

8. **Take note** of the command to be executed on worker nodes, ***tester1*** and ***tester2.*** ***Copy this command in a notepad file. You will need it later. The command is similar to:***

   kubeadm join 10.10.0.100:6443 --token ddynn0.xxxxxxxxxxxxde \

   --discovery-token-ca-cert-hash sha256:xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

9. Run the following command to deploy network.

   # kubectl apply -f \

   https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml

10. Now run the following commands to verify the status. Please note that it may take few minutes to change the status to "***ready***"

    # kubectl get nodes

```
[root@server ~]# kubectl get nodes
NAME                STATUS   ROLES                  AGE     VERSION
server.example.com   Ready    control-plane,master   3m46s   v1.20.2
[root@server ~]#
```

    # kubectl get pods --all-namespaces

```
[root@server ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                          READY   STATUS
kube-system   coredns-74ff55c5b-glnt6                       1/1     Running
kube-system   coredns-74ff55c5b-jj4q6                       1/1     Running
kube-system   etcd-server.example.com                       1/1     Running
kube-system   kube-apiserver-server.example.com             1/1     Running
kube-system   kube-controller-manager-server.example.com    1/1     Running
kube-system   kube-flannel-ds-6bxv2                         1/1     Running
kube-system   kube-proxy-4f8b4                              1/1     Running
kube-system   kube-scheduler-server.example.com             1/1     Running
[root@server ~]#
```

Sequence 4. Deploy the Kubernetes Dashboard

1. Confirm the Namespace by running following command

```
[root@server ~]# kubectl get ns -o wide
NAME                   STATUS   AGE
default                Active   30m
kube-node-lease        Active   30m
kube-public            Active   30m
kube-system            Active   30m
kubernetes-dashboard   Active   3m25s
```

2. On the master node, deploy the Kubernetes Dashboard by running the following command

kubectl apply -f
https://raw.githubusercontent.com/kubernetes/dashboard/v2.3.1/aio/deploy/recommended.yaml

```
[root@server ~]# kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.3.1/aio/deploy/recommended.yaml
namespace/kubernetes-dashboard created
serviceaccount/kubernetes-dashboard created
service/kubernetes-dashboard created
secret/kubernetes-dashboard-certs created
secret/kubernetes-dashboard-csrf created
secret/kubernetes-dashboard-key-holder created
configmap/kubernetes-dashboard-settings created
```

3. Edit Dashboard Deployment and add **- --token-ttl=43200**

   # kubectl -n kubernetes-dashboard edit deployments kubernetes-dashboard

```
template:
  metadata:
    creationTimestamp: null
    labels:
      k8s-app: kubernetes-dashboard
  spec:
    containers:
    - args:
      - --auto-generate-certificates
      - --namespace=kubernetes-dashboard
      - --token-ttl=43200
      image: kubernetesui/dashboard:v2.0.0
      imagePullPolicy: Always
      livenessProbe:
        failureThreshold: 3
        httpGet:
          path: /
          port: 8443
          scheme: HTTPS
        initialDelaySeconds: 30
        periodSeconds: 10
        successThreshold: 1
        timeoutSeconds: 30
      name: kubernetes-dashboard
      ports:
      - containerPort: 8443
```

4. Verify that all pods are running with

   # kubectl get pods --all-namespaces -o wide

```
[root@server ~]# kubectl get pods --all-namespaces -o wide
NAMESPACE              NAME                                         READY   STATUS    RESTARTS   AGE     IP             NODE
kube-system            coredns-78fcd69978-ptqvq                     1/1     Running   0          4m43s   10.244.0.2     server.
kube-system            coredns-78fcd69978-qtktx                     1/1     Running   0          4m43s   10.244.0.3     server.
kube-system            etcd-server.example.com                      1/1     Running   0          4m56s   10.10.0.100    server.
kube-system            kube-apiserver-server.example.com            1/1     Running   0          4m56s   10.10.0.100    server.
kube-system            kube-controller-manager-server.example.com   1/1     Running   0          4m58s   10.10.0.100    server.
kube-system            kube-flannel-ds-sjtp7                        1/1     Running   0          2m55s   10.10.0.100    server.
kube-system            kube-proxy-4c8pf                             1/1     Running   0          4m44s   10.10.0.100    server.
kube-system            kube-scheduler-server.example.com            1/1     Running   0          4m56s   10.10.0.100    server.
kubernetes-dashboard   dashboard-metrics-scraper-856586f554-v47mp   1/1     Running   0          97s     10.244.0.5     server.
kubernetes-dashboard   kubernetes-dashboard-79b875f7f8-5fk9b        1/1     Running   0          26s     10.244.0.6     server.
[root@server ~]#
```

5. Start the Dashboard with

   # kubectl proxy &

```
[root@server ~]# kubectl proxy
Starting to serve on 127.0.0.1:8001
```

6. ***Open another Tab*** in the terminal and create the required user (ServiceAccount) for the Kubernetes Dashboard with cluster-admin privileges. Text file is available for directl copy/paste to save time.

   # vi kubernetes-dashboard-admin-user.yaml
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: admin-user

```
  namespace: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: admin-user
  namespace: kubernetes-dashboard
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: admin-user
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: cluster-admin
subjects:
- kind: ServiceAccount
  name: admin-user
  namespace: kubernetes-dashboard
```

7. Apply this yaml

   # kubectl apply -f ~/kubernetes-dashboard-admin-user.yaml

```
[root@server ~]# kubectl apply -f ~/kubernetes-dashboard-admin-user.yaml
serviceaccount/admin-user created
clusterrolebinding.rbac.authorization.k8s.io/admin-user created
```

8. Get a Token to access the Kubernetes Dashboard.

   # kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}')

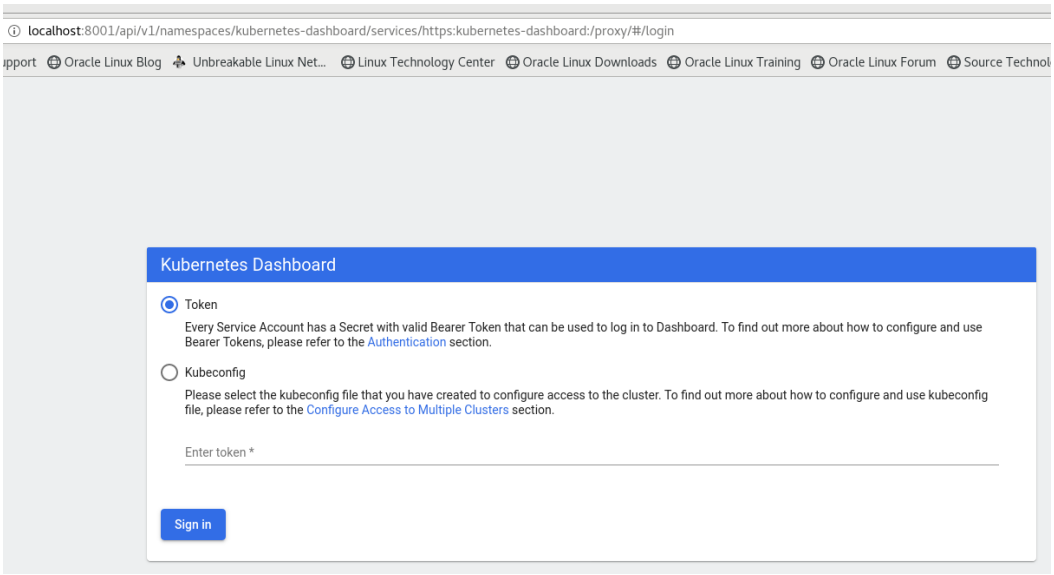
```
[root@server ~]# kubectl -n kubernetes-dashboard describe secret $(kubectl -n kubernetes-dashboard get secret | grep admin-user | awk '{print $1}')
Name:         admin-user-token-9zm5g
Namespace:    kubernetes-dashboard
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: admin-user
              kubernetes.io/service-account.uid: 3a4ea869-3366-4972-9d4b-ada6f0fbaf29

Type:  kubernetes.io/service-account-token

Data
====
ca.crt:     1066 bytes
namespace:  20 bytes
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IkQ2SllpMzUxQ1RpOUszQTd1SmdaOFU1NzNRUTh2NDBSQ0NrWGhPZEFmM0kifQ.eyJpc3MiOiJrdWJlcm5ldGVzL3NlcnZpY2VhY2NvdW50Ii
lYWNjb3VudC9uYW1lc3BhY2UiOiJrdWJlcm5ldGVzLWRhc2hib2FyZCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VjcmV0Lm5hbWUiOiJhZG1pbi1c2VyLXRva2VuLTl6bTVnIiwi
WNjb3VudC9zZXJ2aWNlLWFjY291bnQubmFtZSI6ImFkbWluLXVzZXIiLCJrdWJlcm5ldGVzLmlvL3NlcnZpY2VhY2NvdW50L3NlcnZpY2UtYWNjb3VudC51aWQiOiIzYTRlYTg2OS0zMzY2LTQ5Nz
zdWIiOiJzeXN0ZW06c2VydmljZWFjY291bnQ6a3ViZXJuZXRlcy1kYXNoYm9hcmQ6YWRtaW4tdXNlciJ9.Iy32t5FbvBpz3bblQcAvDuueEe-bfE80RSIqg6HnQmWkfC_vOPMgQK4II7uHRfEjgM_
CdI9B1wroaszseSskyaYMY_a5hKEeQsI0MKdZC-yPbW2K4ZkGjVPhvQCDaBTsoQ2ZcGzYCe5mZ2yV4k8yYIWh1YiJXPLG8AmJuIcKoNksTGecTQUDaOdscF58LC2ySEuK5Jy29qu3W4cLpSw3n-25
chIZKayzXAfXSmR-CVtl783MQflsOhkFO5k-LoyO7GIi24PFQKqg2XzBuL-DoJPCAQug
[root@server ~]#
```

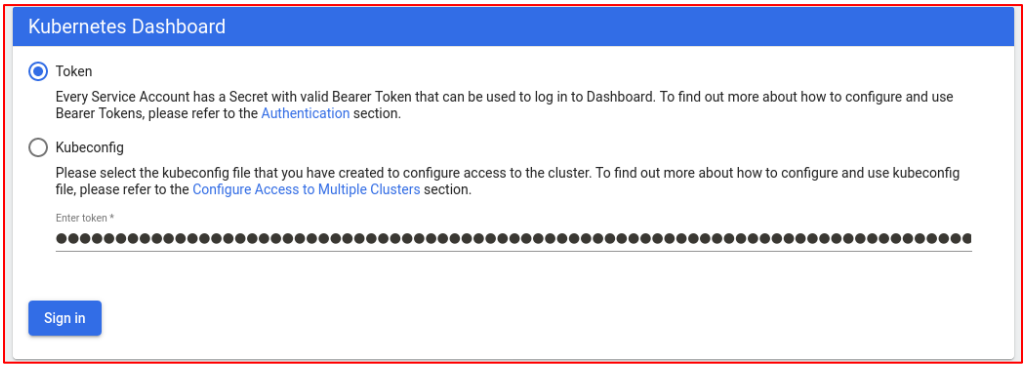
9. Open the Dashboard available at

http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/

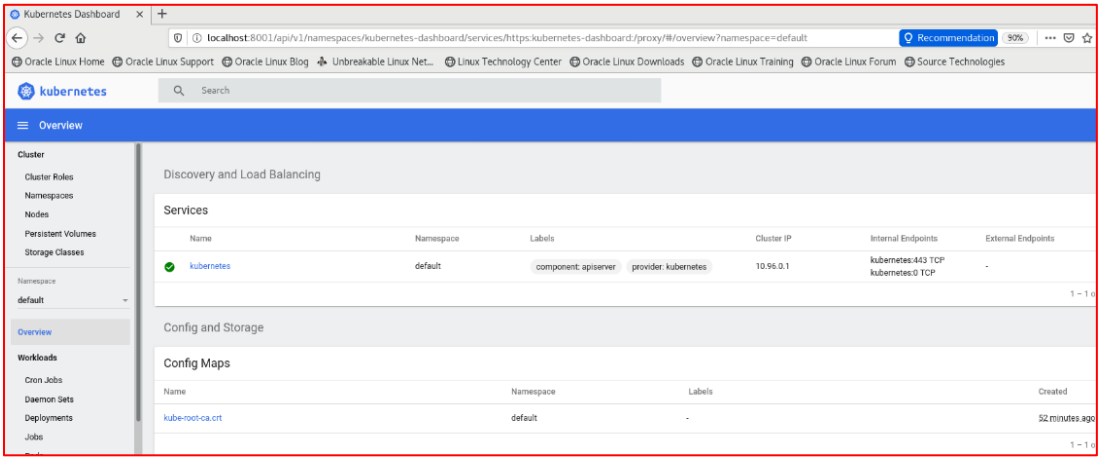


The UI can *only* be accessed from the machine where the command is executed. See ***kubectl proxy --help*** for more options.

10. Copy the Token displayed and paste it in the login page by selecting ***Token*** to login as given below.



11. Dashboard will open as given below.



12. If you lose the token, you can get with following commands.

a. List secrets using

kubectl get secrets

```
[root@server ~]# kubectl get secrets
NAME                  TYPE                                  DATA   AGE
default-token-4bbdp   kubernetes.io/service-account-token   3      14m
```

b. Use kubectl describe to get the access token. Use secret name from above command to get the token.

kubectl describe secret default-token-4bbdp

```
[root@server ~]# kubectl get secrets
NAME                  TYPE                                  DATA
default-token-4bbdp   kubernetes.io/service-account-token   3
[root@server ~]# kubectl describe secret default-token-4bbdp
Name:         default-token-4bbdp
Namespace:    default
Labels:       <none>
Annotations:  kubernetes.io/service-account.name: default
              kubernetes.io/service-account.uid: 0755c252-3f73-4

Type:  kubernetes.io/service-account-token

Data
====
token:      eyJhbGciOiJSUzI1NiIsImtpZCI6IkNub2xRU0FRVWduYUZEVGZv
dC9uYW1lc3BhY2UiOiJkZWZhdWx0Iiwia3ViZXJuZXRlcy5pby9zZXJ2aWNlYWNj
iZGVmYXVsdCIsImt1YmVybmV0ZXMuaW8vc2VydmljZWFjY291bnQvc2VydmljZS1
F1bHQifQ.aEXwrSvBBXnUv-QrUdYpo8hF890GoPmEqtJvt-E9IbiqWStG3nn7xkA
_sGgnHjepBntZw_RlRT73vTcCQZGURr43SzpEGBKhZcIhrSrE2w-K0fMB99_K27X
ca.crt:     1066 bytes
namespace:  7 bytes
[root@server ~]#
```

Sequence 5. Set up worker nodes; tester1 and tester2

1. Configure firewall rules on both the nodes; ***tester1*** and ***tester2***.

   # firewall-cmd --permanent --add-port={10250,10255,30000-32767,6783}/tcp

   # firewall-cmd --reload

2. Install kubeadm and docker package on both nodes

   [root@tester1 ~]# yum install kubeadm docker -y

   [root@tester2 ~]# yum install kubeadm docker -y

3. Change Docker Configuration to use system and start & enable docker service

   [root@tester1 ~]# systemctl enable --now docker

   [root@tester2 ~]# systemctl enable --now docker

   cat > /etc/docker/daemon.json <<EOF

   {"exec-opts": ["native.cgroupdriver=systemd"]}

   EOF

   [root@tester1 ~]# systemctl restart docker

[root@tester2 ~]# systemctl restart docker

4. Now use the command similar to following, to join the K8s cluster.
   To join worker nodes to Server node, a token is required. Whenever kubernetes server is initialized, then in the output we get command and token. Copy that command and run on both nodes. (Refer to Sequence 3, step 7 and 8)

   [root@tester1 ~]# kubeadm join 10.10.0.100:6443 --token kndxd6.zolzvjaj8bifonoj \
   --discovery-token-ca-cert-hash sha256:f0f118ebe0ab7f6ed2510e6e1bff2d23fb2b58f4f5ee8e69ed92323750659aa0

   [root@tester2 ~]# kubeadm join 10.10.0.100:6443 --token kndxd6.zolzvjaj8bifonoj \
   --discovery-token-ca-cert-hash sha256:f0f118ebe0ab7f6ed2510e6e1bff2d23fb2b58f4f5ee8e69ed92323750659aa0

5. Output of above command would be something like

```
[root@tester1 ~]# kubeadm join 10.10.0.100:6443 --token ddynn0.4g0y958pwtg8lude     --discovery-token-ca-cert-hash
odaa08baf49a2568570a02f7fce1a38bee11c3a1d3c157a5d638fb8eb
N0701 11:32:14.549052    2296 join.go:346] [preflight] WARNING: JoinControlPane.controlPlane settings will be ignore
l-plane flag is not set.
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is
ase follow the guide at https://kubernetes.io/docs/setup/cri/
        [WARNING FileExisting-tc]: tc not found in system path
        [WARNING Service-Kubelet]: kubelet service is not enabled, please run 'systemctl enable kubelet.service'
[preflight] Reading configuration from the cluster...
[preflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -oyaml'
[kubelet-start] Downloading configuration for the kubelet from the "kubelet-config-1.18" ConfigMap in the kube-sys
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
[kubelet-start] Starting the kubelet
[kubelet-start] Waiting for the kubelet to perform the TLS Bootstrap...

This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

Run 'kubectl get nodes' on the control-plane to see this node join the cluster.
```

6. Now verify Nodes status from **server** node using kubectl command

   # kubectl get nodes

```
[root@server ~]# kubectl get nodes
NAME                  STATUS   ROLES    AGE     VERSION
server.example.com    Ready    master   18m     v1.18.5
tester1.example.com   Ready    <none>   7m22s   v1.18.5
tester2.example.com   Ready    <none>   7m16s   v1.18.5
[root@server ~]#
```

7. To assign a role to tester1 and tester2, use the following command:

   # kubectl label node tester1.example.com node-role.kubernetes.io/worker=worker

   # kubectl label node tester2.example.com node-role.kubernetes.io/worker=worker

```
[root@server ~]# kubectl label node tester1.example.com node-role.kubernetes.io/worker=worker
node/tester1.example.com labeled
[root@server ~]# kubectl get nodes
NAME                  STATUS   ROLES    AGE     VERSION
server.example.com    Ready    master   25m     v1.18.6
tester1.example.com   Ready    worker   2m14s   v1.18.6
```

8. Verify and Fix. Use the following command to check the status of all nodes on master

   # kubectl get pods --all-namespaces -o wide

```
[root@server ~]# kubectl get pods --all-namespaces -o wide
NAMESPACE              NAME                                          READY   STATUS    RESTARTS   AGE     IP             NODE
kube-system            coredns-78fcd69978-ptqvq                      1/1     Running   0          12m     10.244.0.2     server.ex
kube-system            coredns-78fcd69978-qtktx                      1/1     Running   0          12m     10.244.0.3     server.ex
kube-system            etcd-server.example.com                       1/1     Running   0          12m     10.10.0.100    server.ex
kube-system            kube-apiserver-server.example.com             1/1     Running   0          12m     10.10.0.100    server.ex
kube-system            kube-controller-manager-server.example.com    1/1     Running   0          12m     10.10.0.100    server.ex
kube-system            kube-flannel-ds-bt7cl                         1/1     Running   0          45s     10.10.0.101    tester1.e
kube-system            kube-flannel-ds-sjtp7                         1/1     Running   0          10m     10.10.0.100    server.ex
kube-system            kube-flannel-ds-trkrh                         1/1     Running   0          54s     10.10.0.102    tester2.e
kube-system            kube-proxy-2ss55                              1/1     Running   0          54s     10.10.0.102    tester2.e
kube-system            kube-proxy-4c8pf                              1/1     Running   0          12m     10.10.0.100    server.ex
kube-system            kube-proxy-l2259                              1/1     Running   0          45s     10.10.0.101    tester1.e
kube-system            kube-scheduler-server.example.com             1/1     Running   0          12m     10.10.0.100    server.ex
kubernetes-dashboard   dashboard-metrics-scraper-856586f554-v47mp    1/1     Running   0          9m25s   10.244.0.5     server.ex
kubernetes-dashboard   kubernetes-dashboard-79b875f7f8-5fk9b         1/1     Running   0          8m14s   10.244.0.6     server.ex
[root@server ~]#
```

9. IF ***tester1*** and ***tester2*** failed to acquire lease. This means, the pod didn't get the podCIDR. To fix it, from the master-node, first find out your funnel CIDR

   # cat /etc/kubernetes/manifests/kube-controller-manager.yaml | grep -i cluster-cidr

   Output:

   - --cluster-cidr=10.244.0.0/16

   Then run the following commands from the master node:

   # kubectl patch node tester1.example.com -p '{"spec":{"podCIDR":"10.244.0.0/16"}}'

   # kubectl patch node tester2.example.com -p '{"spec":{"podCIDR":"10.244.0.0/16"}}'

10. After 2-3 Minutes, use the following command to check the status of all nodes on master

    # kubectl get pods --all-namespaces -o wide

```
[root@server ~]# kubectl get pods --all-namespaces -o wide
NAMESPACE              NAME                                          READY   STATUS    RESTARTS       AGE     IP
kube-system            coredns-78fcd69978-ptqvq                      1/1     Running   0              14m     10.244.0.2
kube-system            coredns-78fcd69978-qtktx                      1/1     Running   0              14m     10.244.0.3
kube-system            etcd-server.example.com                       1/1     Running   0              14m     10.10.0.100
kube-system            kube-apiserver-server.example.com             1/1     Running   0              14m     10.10.0.100
kube-system            kube-controller-manager-server.example.com    1/1     Running   0              14m     10.10.0.100
kube-system            kube-flannel-ds-bt7cl                         1/1     Running   1 (117s ago)   2m54s   10.10.0.101
kube-system            kube-flannel-ds-sjtp7                         1/1     Running   0              12m     10.10.0.100
kube-system            kube-flannel-ds-trkrh                         1/1     Running   0              3m3s    10.10.0.102
kube-system            kube-proxy-2ss55                              1/1     Running   0              3m3s    10.10.0.102
kube-system            kube-proxy-4c8pf                              1/1     Running   0              14m     10.10.0.100
kube-system            kube-proxy-l2259                              1/1     Running   0              2m54s   10.10.0.101
kube-system            kube-scheduler-server.example.com             1/1     Running   0              14m     10.10.0.100
kubernetes-dashboard   dashboard-metrics-scraper-856586f554-v47mp    1/1     Running   0              11m     10.244.0.5
kubernetes-dashboard   kubernetes-dashboard-79b875f7f8-5fk9b         1/1     Running   0              10m     10.244.0.6
[root@server ~]#
```

11. For a detailed status of nodes use

    # kubectl describe nodes

Sequence 6. Troubleshooting and Reset

1. Get a comprehensive report on your nodes with:

   ```
   # kubectl describe nodes
   # kubectl get pods -n kube-system -o wide
   # kubectl cluster-info dump
   ```

2. Reset the Cluster on all three VMs with

   ```
   # kubeadm reset
   ```