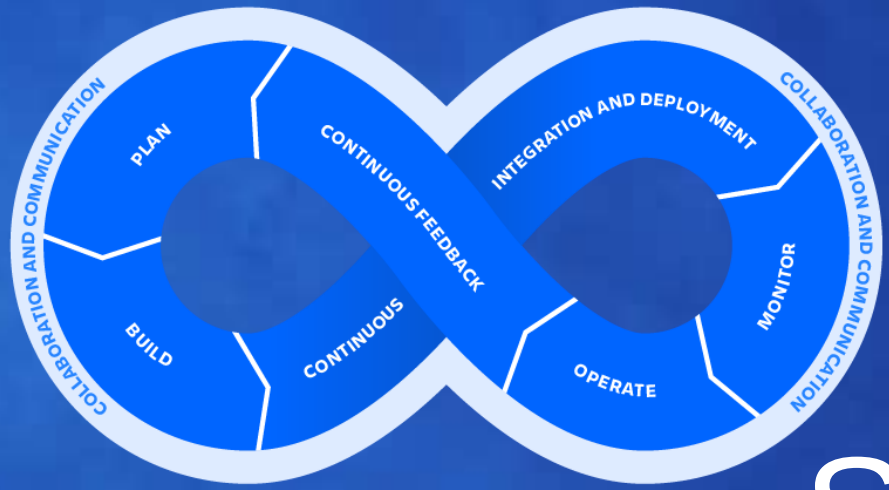


Session 16. Introduction to Kubernetes

Ram N Sangwan

www.rnsangwan.com

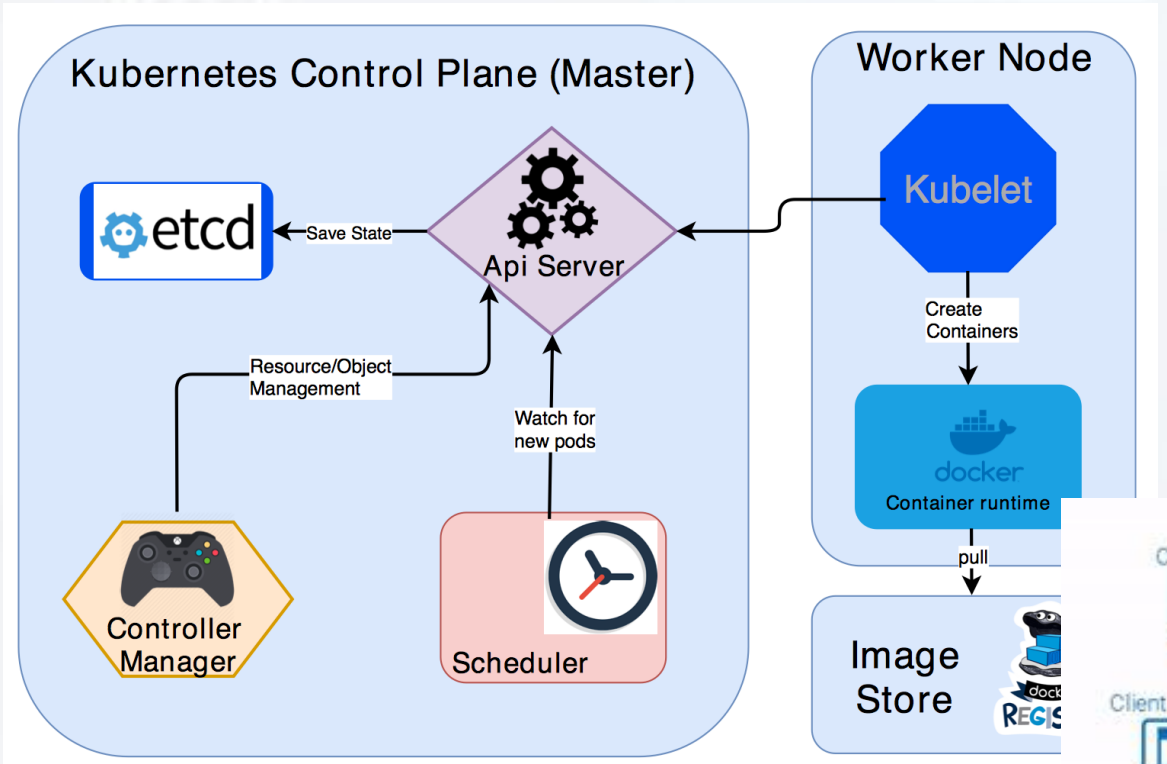
Agenda

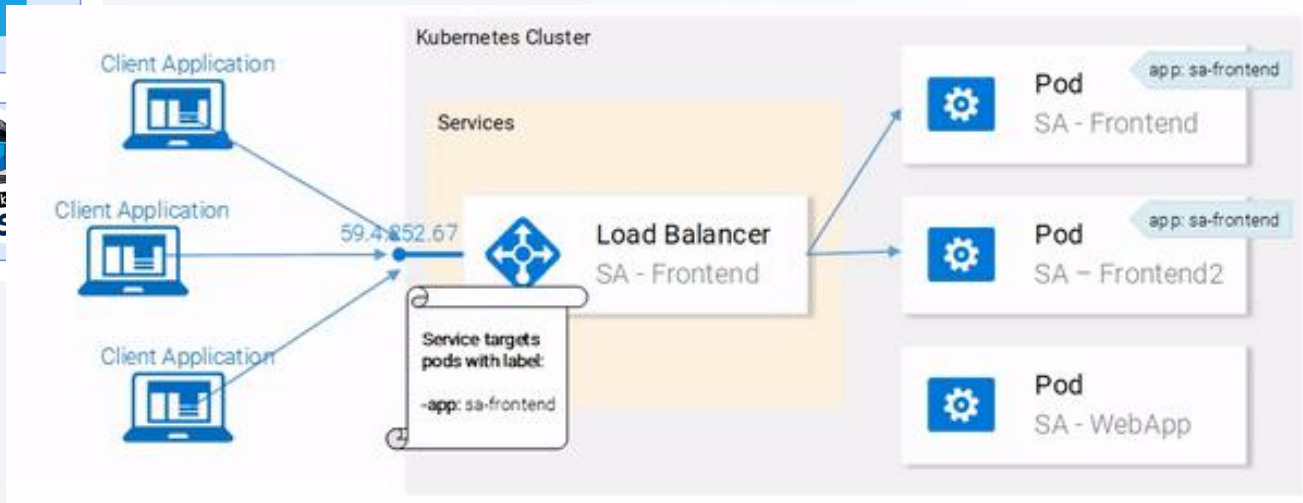
- What is Kubernetes & why
- Kubernetes Terminology
- Kick start Kubernetes
- Installation
- Initialize the cluster
- Setup the POD network
- Build HA cluster
- Validate the cluster

Kubernetes

- A very popular option for managing containers (and potentially other things).
  - Mature open source project
  - Based on learning from internal Google projects.
  - Also known as k8s, was developed by Google and donated to “Cloud Native Computing foundation”
  - Extensible / flexible API architecture
  - API is expressed in object types
  - Not necessarily just about containers
  - Can group multiple containers into a single logical unit for management and deployment.

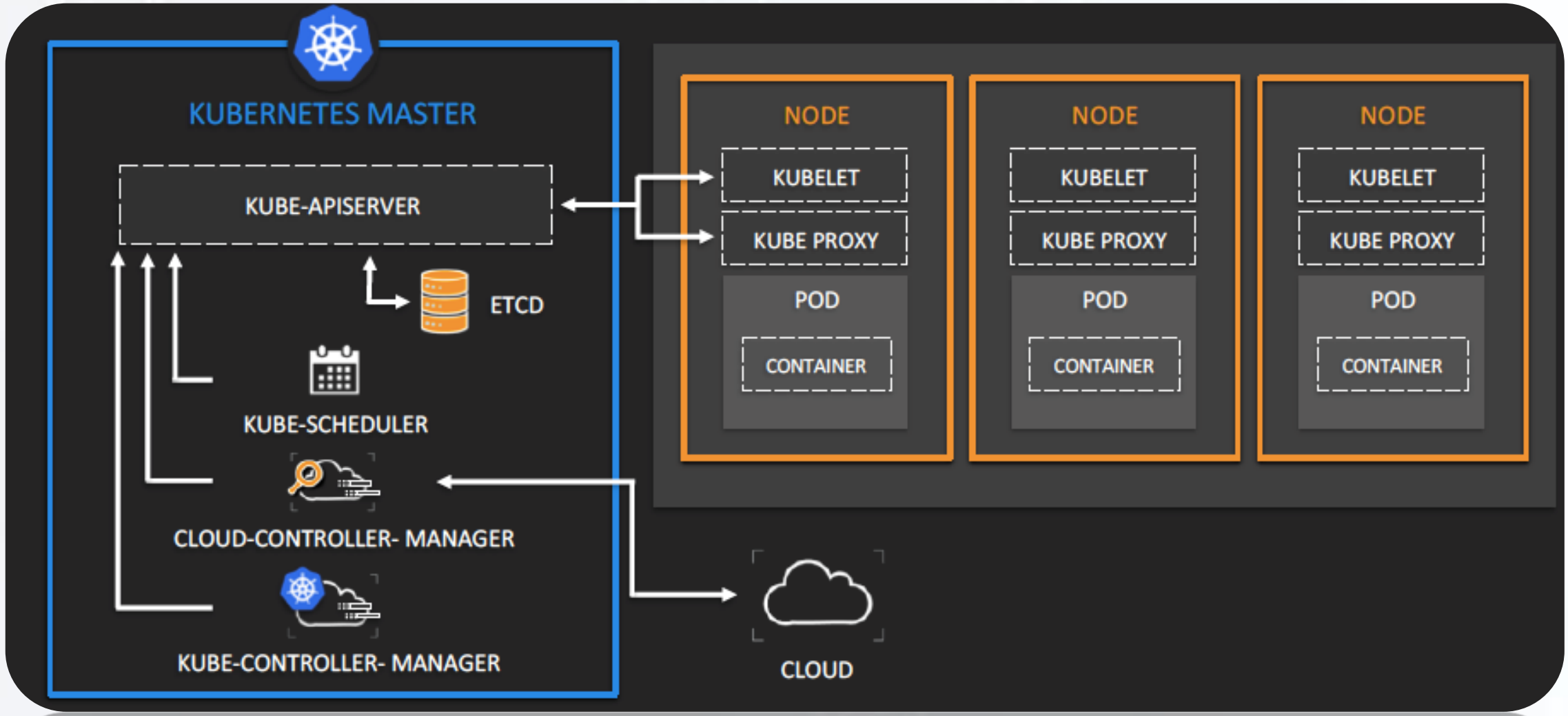
Kubernetes Architecture





https://kubernetes.io/docs/concepts/architecture/

Basic Kubernetes Architecture



Kubernetes Installation Considerations

Download Kubernetes from upstream repositories on VM or physical servers.

Criteria to consider while evaluating a Kubernetes solution:

- High Availability – Does your Kubernetes solution install clusters that are highly available, with replication of the underlying metadata for recovery against failures?
- Upgrades – Kubernetes community delivers a major upgrade every 3-4 months.
  - What is your Kubernetes upgrade strategy?
  - What downtimes will upgrades require and is that acceptable for business?

Installation Considerations..

- Support for hybrid – Does your Kubernetes solution equally support the private data center and public cloud endpoints that your business needs to deliver Kubernetes on?
- Federation support – Does your Kubernetes solution support installation of federated clusters that can grow across private and public clouds for robustness of infrastructure and dynamic burstability?
- Enterprise-ready features – What additional enterprise-readiness features does your Ops team need to run Kubernetes at scale for large scale of users?
  - Are they supported by your Kubernetes solution of choice?
  - Some examples include, SSO support, RBAC, isolated networking, persistent storage.

Install Kubernetes on CentOS 7 / OL 7

- In Kubernetes setup we have one master node and multiple nodes.
- Cluster nodes are known as worker nodes or Minions.
- From the master node we manage the cluster and its nodes using ‘kubeadm‘ and ‘kubectl‘ command.
- Kubernetes can be installed and deployed using:
  - Minikube (It is a single node ***kubernetes*** cluster)
  - Kubeadm (Multi Node Cluster in our own premises)

Master Node Components

- **API Server** – It provides kubernetes API using Jason/Yaml over http, states of API objects are stored in etcd
- **Scheduler** – It is a program on master node which performs the scheduling tasks like launching containers in worker nodes based on resource availability
- **Controller Manager** – Main Job of Controller manager is to monitor replication controllers and create pods to maintain desired state.
- **etcd** – It is a Key value pair data base. It stores configuration data of cluster and cluster state.
- **Kubectl utility** – It is a command line utility which connects to API Server on port 6443. It is used by administrators to create pods, services etc.

Worker Nodes Components

- **Kubelet** – It is an agent which runs on every worker node, it connects to docker and takes care of creating, starting, deleting containers.
- **Kube-Proxy** – It routes the traffic to appropriate containers based on ip address and port number of the incoming request. In other words we can say it is used for port translation.
- **Pod** – Pod can be defined as a multi-tier or group of containers that are deployed on a single worker node or docker host.

Setup the Environment

Disable SELinux & setup firewall rules

- Login to your kubernetes server (master node) and disable selinux

```
# setenforce 0
# sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/selinux/config
```

- Set the following firewall rules.

```
# firewall-cmd --permanent --add-port=6443/tcp
# firewall-cmd --permanent --add-port={2379,2380}/tcp
# firewall-cmd --permanent --add-port={10250,10251,10252,10255}/tcp
# firewall-cmd --reload
# modprobe br_netfilter
# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

Local Resolver and Swap

- Verify /etc/hosts file on master and worker nodes

  10.10.0.100 server

  10.10.0.101 tester1

  10.10.0.101 tester2

- Disable Swap in all nodes using “swapoff -a” command and remove or comment out swap partitions or swap file from fstab file

Configure Kubernetes Repository

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

```
[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-p
ackage-key.gpg
~
```

Install Kubeadm and Docker

- Run the following command to install kubeadm. Assuming that you have already installed docker.

  # yum install kubeadm -y

- Start and enable kubectl service

  # systemctl restart docker

  # systemctl restart kubelet && systemctl enable kubelet

Initialize Master with *kubeadm init*

- Run the following command to initialize and setup kubernetes master.

  # kubeadm init --apiserver-advertise-address=<your ipv4 IP>

- Output of above command would be something like:

```
[root@server ~]# kubeadm init --apiserver-advertise-address=10.10.0.100
W0701 10:52:32.452036   13631 configset.go:202] WARNING: kubeadm cannot
roups [kubelet.config.k8s.io kubeproxy.config.k8s.io]
[init] Using Kubernetes version: v1.18.5
[preflight] Running pre-flight checks
        [WARNING Firewalld]: firewalld is active, please ensure ports [6
may not function correctly
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docke
ver is "systemd". Please follow the guide at https://kubernetes.io/docs/
        [WARNING FileEx
[preflight] Pulling ima
[preflight] This might
[preflight] You can als
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 ~]#
```

Use the Cluster as root

- 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

- Run ***kubectl get nodes*** to get status of cluster nodes.
- Run ***kubectl get pods --all-namespaces*** to get status of cluster pods.

```
[root@server ~]# kubectl get nodes
NAME                 STATUS     ROLES    AGE     VERSION
server.example.com   NotReady   master   6m48s   v1.18.5
[root@server ~]# kubectl get pods --all-namespaces
NAMESPACE     NAME                                          READY   STATUS    RESTARTS   AGE
kube-system   coredns-66bff467f8-922d8                      0/1     Pending   0          7m26s
kube-system   coredns-66bff467f8-kl4nb                      0/1     Pending   0          7m25s
kube-system   etcd-server.example.com                       1/1     Running   0          7m29s
kube-system   kube-apiserver-server.example.com             1/1     Running   0          7m29s
kube-system   kube-controller-manager-server.example.com    1/1     Running   0          7m29s
kube-system   kube-proxy-k77g6                              1/1     Running   0          7m26s
kube-system   kube-scheduler-server.example.com             1/1     Running   0          7m29s
```

Setup the POD network

- To make the cluster status ready and ***kube-dns*** status running, deploy the pod network so that containers of different host communicated each other. POD network is the overlay network between the worker nodes.
- Run the following command to deploy network.

export kubever=$(kubectl version | base64 | tr -d '\n')

kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"

```
[root@server ~]# export kubever=$(kubectl version | base64 | tr -d '\n')
[root@server ~]# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
[root@server ~]#
```

Verify Master/Controller Working

```
[root@server ~]# export kubever=$(kubectl version | base64 | tr -d '\n')
[root@server ~]# kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$kubever"
serviceaccount/weave-net created
clusterrole.rbac.authorization.k8s.io/weave-net created
clusterrolebinding.rbac.authorization.k8s.io/weave-net created
role.rbac.authorization.k8s.io/weave-net created
rolebinding.rbac.authorization.k8s.io/weave-net created
daemonset.apps/weave-net created
[root@server ~]#
```

Now run the following commands to verify the status

```
# kubectl get nodes
```

```
[root@server ~]# kubectl get nodes
NAME                 STATUS   ROLES    AGE   VERSION
server.example.com   Ready    master   12m   v1.18.5
[root@server ~]#
```

Disable SELinux & Configure firewall rules on Nodes

```
# setenforce 0
# sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
# firewall-cmd --permanent --add-port={10250,10255,30000-32767,6783}/tcp
# firewall-cmd  --reload
# modprobe br_netfilter
# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
```

```
[root@server ~]# ssh tester1
Last login: Wed Jul  1 11:13:17 2020 from server
[root@tester1 ~]# setenforce 0
[root@tester1 ~]# sed -i --follow-symlinks 's/SELINUX=enforcing/SELINUX=disabled/g' /etc/sysconfig/selinux
[root@tester1 ~]# firewall-cmd --permanent --add-port={10250,10255,30000-32767,6783}/tcp
success
[root@tester1 ~]# firewall-cmd  --reload
success
[root@tester1 ~]# modprobe br_netfilter
[root@tester1 ~]# echo '1' > /proc/sys/net/bridge/bridge-nf-call-iptables
[root@tester1 ~]#
```

Configure Repositories on worker nodes

- Create Repo File on both worker nodes, tester1 and tester2

```
# 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
```

Install kubeadm & Docker on Nodes

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

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

- Start and enable docker service

[root@tester1 ~]# systemctl restart docker && systemctl enable docker

[root@tester2 ~]# systemctl restart docker && systemctl enable docker

Join worker Nodes to Master

- To join worker nodes to Master node, a token is required. Whenever kubernetes master initialized then in the output we get command and token.
- Copy that command and run on both nodes.

[root@tester2 ~]# kubeadm join 10.10.0.100:6443 --token kndxd6.zolzvjaj8bifonoj \

--discovery-token-ca-cert-hash sha256:f0f118ebe0ab7f6ed2510e6e1bff2d23fb2b58f4f5ee8e69ed92323750659aa0

- 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 sha256:9c0374e
0daa08baf49a2568570a02f7fce1a38bee11c3a1d3c157a5d638fb8eb
W0701 11:32:14.549052    2296 join.go:346] [preflight] WARNING: JoinControlPane.controlPlane settings will be ignored when contro
l-plane flag is not set.
[preflight] Running pre-flight checks
        [WARNING IsDockerSystemdCheck]: detected "cgroupfs" as the Docker cgroup driver. The recommended driver is "systemd". Ple
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-system namespace
[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.
```

Verify the Installation

- Now verify Nodes status from master node using kubectl command

  # kubectl get nodes

- For a detailed status of nodes use

  # kubectl describe nodes

- Troubleshooting Commands

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

More on Tokens and Nodes

- By default, tokens expire after 24 hours.
  # kubeadm token create
- If you don't have the value of --discovery-token-ca-cert-hash, you can get it by running.
  # openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | \
  openssl dgst -sha256 -hex | sed 's/^.* //‘
- Controlling your cluster from machines other than the control-plane node
  # scp root@<control-plane-host>:/etc/kubernetes/admin.conf .
  # kubectl --kubeconfig ./admin.conf get nodes
- Remove the node
  # kubeadm reset
  # kubectl drain <node name> --delete-local-data --force --ignore-daemonsets

Thank You