**--- Kubernetes ---**

1. **Installation & setup and post installation task ( Unit II)**

**Kubernetes installation**

1. **Control plane : -** All four services ( etcd, api, controller and scheduler ) are installed on the same server on the controller. All four services are installed using separate containers on the same server. This isolates each service from each other. 90% of companies prefer this method.
2. **Hard way :-** All services installed on the same service as a systemd on the same isolate. This setup doesn’t isolate each service from each other just like containers.

**Control plane -**

**Server required :-** **1 master node** (Min 2GB RAM) and **2 worker nodes** ( worker1 & worker2 - 1GBRAM) for practice

OS - RHEL 7 / CentOS 7 / Ubuntu etc.

**Installation & configuration of Master Node :-**

System specification -

CentOS 7 instance ( t2 small - 1cpu 2GB 10 GB HDD)

1. **System prerequisite -**
2. System should ping with name ( don’t change hostname after installation )

Set hostname -

Eg - master.example.com

Enter IP and hostname in /etc/hosts file

1. Selinux should be disabled / permissive (k8s doesn’t support selinux)

Vim /etc/sysconfig/selinux

Disable

1. Swap space should be off

* In cloud swap is off already for physical check swap is off or not (swapon -a ) .

**2. In the control plane method Docker is required on the master node also. Because all four services are installed in containers.**

1. Install Docker CE (Community Edition) on master

<https://docs.docker.com/engine/install/>

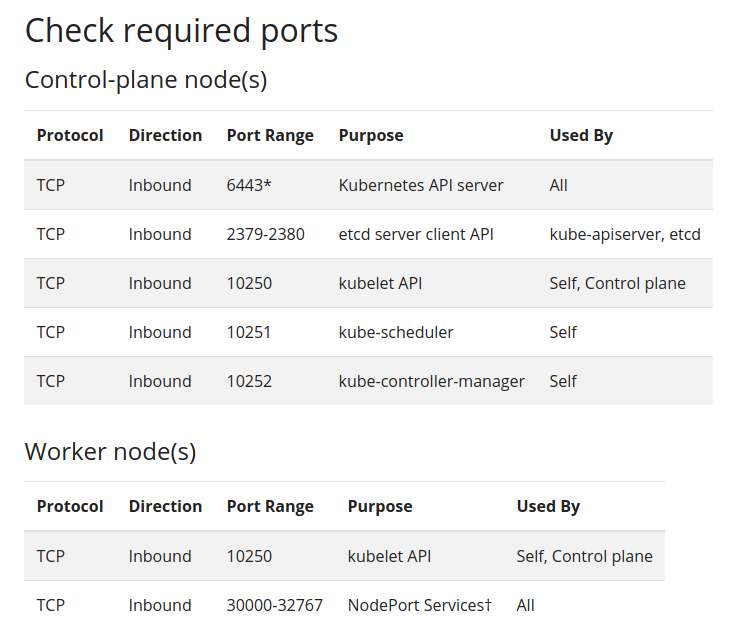
**3. Install kubernetes administrator ( kubeadm ) on control node**

kubeadm : bootstrap utility ( install and manage all four services in container on control node via cmd). Utility to make/setup a master node.

\*\*\* kubeadm is required for control plane installation and upgradation only.

* Go to <https://kubernetes.io/>
* Go to Documentation <https://kubernetes.io/docs/home/>
* Search for install kubeadm and go to https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

**Port no ( Usually asked in interviews )**



cat <<EOF | sudo tee /etc/modules-load.d/k8s.conf

br\_netfilter

EOF

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf

net.bridge.bridge-nf-call-ip6tables = 1

net.bridge.bridge-nf-call-iptables = 1

EOF

sudo sysctl --system

* Setup the repository

RHEL/CentOS

baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-\$basearch

Ubuntu/Debian

echo "deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee /etc/apt/sources.list.d/kubernetes.list

* Install the kubeadm

Eg -

RHEL/CentOS

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

Ubuntu/Debian

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

* Enable and start docker and kubelet

# systemctl enable docker

# systemctl enable kubelet

# systemctl start docker

# systemctl start kubelet

\*\*\* kube-proxy will run containers based on the control plane. Kubelet and docker can’t be run as container based, these will run directly on the master node.

**# kubeadm init** (This command will make this system master node.)

You will get alerts for less h/w for 1 cpu.

This command will launch 4 containers and make it a master node.

**Kubeadm -** During the installation time of the control method only.

**Kubelet -** agent, to make/manage (like add,delete) any node master or worker

**Kubectl -** command utility to administrate the whole cluster.

Flow of control

**Kubectl --** reach to **-> API --** API validate from **-> Etcd**

$ netstat -tlpn | grep 6443

Api is running on port 6443 on the master node.

To start using cluster as a regular user

$ sudo mkdir -p $HOME/.kube.config

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

To get added node

$ kubectl get node

You will get error, now run

$ kubectl get node --kubeconfig=/etc/kubernetes/admin.cong

OR

Make export the file

$sudo export KUECONFIG=/etc/kubernetes/admin.conf

To make it permanent

sudo cp /etc/kubernetes/admin ~/.kube/config

Now run kubectl command directly

$ kubectl get nodes ( get all worker )

$ kubectl get ns ( Get namespace of all kubernetes )

$ kubectl get po -n kube-system (list out pod of four services )

$ kubectl get po -n kube-system

Pod is the group of container

Name - DNS name to communicate between two pod ( internal DNS service to resolve pod name)

1/1 - first 1 is no of container in a pod

Pending - core task is pending, discuss further.

**## EKS service of AWS provides pre pre-installed master nodes we need to connect only.**

**which we did till now**

**#### 2. How to configure and register on master, worker nodes. ###**

1. **System prerequisite -**
2. System should ping with name ( don’t change hostname after installation )

worker1.example.com

1. Selinux should be disabled / permissive (k8s doesn’t support selinux)
2. Swap space should be off

**2. Install required packages (docker, kubelet and kube-proxy) by their respective repository.**

1. install docker from official repo
2. Install kubeadm , it will install kubelet automatically.
3. Container will be created for kube-proxy. When we add the workers nodes to the master node, the master node will automatically launch the kube-proxy container from etcd.

Now start and enable docker and kubelet service.

**\*\* Note - Don’t run $ kubeadm init command on any of the worker nodes. Otherwise it will make it a master node.**

**We will have to run the $ kubeadm join command on the worker node.**

**3. Get the kubeadm join command form master node and run on worker node.**

$ kubeadm token create ( To create a token on master.)

$ $ kubeadm token delete ( To delete token on master)

$ kubeadm token list ( To list token, this token is valid for 23h only)

**To create a fresh token on master and join workers to master.**

$ kubeadm token create --help

$ kubeadm token create --print-join-command ( This will create a token and give a command to join the worker on master node. Now run this printed command on worker node)

**To check added worker node**

$ kubectl get nodes

**Repeat same step to add all worker node.**

**### 3. post installation task ###**

**Bring joined worker in ready state ( $ kubectl get node )**

**To bring it in a ready state, we will have to create a k8s network. Kubernetes needs this network to allocate ip to it’s pod/container.**

**To create bridge network : Deploy CNI ( container network interface )**

**CNI is a s/w defined network which will run all nodes and allocate all containers an IP.**

**CNI Type -**

**Flannel : previously used**

**Weave : previously used**

**Calico : Currently we are using calico CNI in the kubernetes cluster.**

**To Install Calico on nodes, Follow below step on master node**

1. **Go to official website ( google calico kubernetes )**

<https://docs.projectcalico.org/getting-started/kubernetes/>

1. **Go to docs.projectcalico.org**
2. **Go to** [**Self-managed on-premises**](https://docs.projectcalico.org/getting-started/kubernetes/self-managed-onprem/)
3. **Download the yaml file.**

curl https://docs.projectcalico.org/manifests/calico.yaml -O

1. **Run the command**

$ kubectl apply -f calico.yaml

**This will create a bridge n/w container on all nodes (workers and master ).**

**To verify run command**

$ kubectl get po -n kube-system

$ kubectl get po -n kube-system -o wide | grep calico

command method (kubectl)

file method

############################

# How to deploy pod in k8s #

############################

### 1. simple pod ###

eg.

$ kubectl get ns

$ kubectl get pod -n kube-system

$ kubectl run test --image=nginx --dry-run -o yaml > pod.yml

$ kubectl run test --image=nginx OR kubectl create -f pod.yml

$ kubectl get po

$ kubectl exec -it test /bin/bash

If we delete docker container manually from worker then k8s controller will automatically create the node

$ kubectl describe pod test ( To check the container details like uptime)

$ kubectl delete pod test

$ kubectl get po

$ kubectl edit po test ( To edit pod in etcd database ls -ld /var/lib/etcd)

### 2.By pod's controller ####

Replication controller resource : It creates replicas of pod and maintain pod's state.

my\_rc.yml

$ kubectl create -f my\_rc.yml

$ kubectl describe pod my-nginx | grep -i controlled

$ kubectl get rc

$ kubectl scale --replicas=4 rc nginx

$ kubectl get po

$ kubectl apply -f my\_rc.yml ( to apply the changes of file my\_rc.yml)

-------

controllers in k8s

RC

RS

Deployment

Deamonsets

statefulset

--------