Kubernetes Installation - Ubuntu

**Below Commands on Both Master - Slave(worker) Node (run command as root user)**

sudo apt update -y

sudo apt install curl apt-transport-https -y

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg

echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list

sudo apt install -y kubeadm=1.28.1-1.1 kubelet=1.28.1-1.1 kubectl=1.28.1-1.1

sudo apt update -y

sudo apt install kubelet kubeadm kubectl -y

sudo apt-mark hold kubelet kubeadm kubectl

sudo swapoff -a

sudo sed -i '/ swap / s/^(.\*)$/#\1/g' /etc/fstab

free -m

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

overlay

br\_netfilter

EOF

sudo modprobe overlay

sudo modprobe br\_netfilter

sudo tee /etc/sysctl.d/kubernetes.conf<<EOF

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

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

net.ipv4.ip\_forward = 1

EOF

sudo sysctl --system

sudo apt install -y curl gnupg2 software-properties-common apt-transport-https ca-certificates

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /etc/apt/trusted.gpg.d/docker-archive-keyring.gpg

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

sudo apt update -y

sudo apt install -y containerd.io

sudo mkdir -p /etc/containerd

sudo containerd config default|sudo tee /etc/containerd/config.toml

sudo sed -i 's/SystemdCgroup \= false/SystemdCgroup \= true/g' /etc/containerd/config.toml

sudo systemctl restart containerd

sudo systemctl enable containerd

systemctl status containerd

sudo systemctl enable kubelet

**Master Only config:**

**Run kubeadm from root user:**

kubeadm init --apiserver-advertise-address=172.31.3.229(masternodeprivateip) --pod-network-cidr=10.244.0.0/16

Enter the commands shown as output as a **regular user(ubuntu)**. Copy and keep your join command for slave node

kubectl apply -f <https://raw.githubusercontent.com/flannel-io/flannel/master/Documentation/kube-flannel.yml>

**Slave only config:**

Run the join command as **root** user:

kubeadm join 172.31.3.229:6443 --token t6u4tu.w17z374p5vzd8sie --discovery-token-ca-cert-hash sha256:67582fe54e8b3dc7f8a39bf9ee4a12a9074a76be3c80d512b3a1ec3c6a6831e3

**(**Note you will get this join command in your master node after kubeadm init)

Confirm in master node, by running command   
kubectl get nodes

Manifests

Pod manifest podmanifest.yml

apiVersion: v1

kind: Pod

metadata:

name: nginx-pod

spec:

containers:

- name: nginx-container

image: nginx

ports:

- containerPort: 80

apply manifest

kubectl apply -f podmanifest.yml

Command Kubectl commands

list pods

kubectl get pods

Describe pod

kubectl describe pod nginx-pod (podname)

check logs

kubectl logs nginx-pod (podname)

how to get inside a pod

kubectl exec -it pod/nginx-pod -- /bin/bash

or

kubectl exec -it pod/nginx-pod -- sh

Delete pod

kubectl delete pod/nginx-pod

or

kubectl delete pod nginx-pod

list namespace

kubectl get ns

or

kubectl get namespace

Create ns

kubectl create ns web(namespacename)

list pods in specific namespace

kubectl get pods -n web(namespace name)

delete ns

kubectl delete ns/web

or

kubectl delete namespace web

**Replicaset:**

apiVersion: apps/v1

kind: ReplicaSet

metadata:

name: nginx-replicaset

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx-container

image: nginx

ports:

- containerPort: 80

kubectl apply -f replicamanifest.yml

kubectl get replicaset

kubectl delete replicaset/nginx-replicaset

**Deployment:**

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx-container

image: nginx:1.7.1

ports:

- containerPort: 80

kubectl apply –f deploymentmanifest.yml

kubectl get deployments

kubectl describe deployments/nginx-deployment

kubectl delete deployment nginx-deployment (To delete the particular deployment)

kubectl rollout status deployment nginx-deployment ( To view our rollout status)

kubectl rollout undo deployment nginx-deployment (To rollback our deployment)

## 

## 

## 

## **Services**

### **Node port svc manifest**

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx

type: NodePort

ports:

- protocol: TCP

port: 80

targetPort: 80

nodePort: 30080

access: awsec2publicip:30080

### 

### **Clusterip Svc manifest**

apiVersion: v1

kind: Service

metadata:

name: nginx-service

spec:

selector:

app: nginx

type: ClusterIP

ports:

- protocol: TCP

port: 80

targetPort: 80

### 

### **deploy centos pod - test curl clusteripsvcipadd:80 inside the pod**

apiVersion: v1

kind: Pod

metadata:

name: centos-pod

spec:

containers:

- name: centos-container

image: centos:latest

command: ["/bin/sleep", "infinity"]

stdin: true

tty: true

## **test curl**

go inside centos pod

kubectl exec -it pod/centos-pod -n test -- /bin/bash

or

kubectl exec -it pod/centos-pod -n test -- sh

curl clusterip:80

# **Kubernetes Ingress**

## **Configuration**

kubectl apply -f https://raw.githubusercontent.com/kubernetes/ingress-nginx/controller-v1.8.2/deploy/static/provider/baremetal/deploy.yaml

kubectl apply -f deploy.yaml

**Ingressroute:**

apiVersion: v1

kind: Ingress

metadata:

name: nginx-ingress

spec:

rules:

- host: xyz.com

http:

paths:

- path: /

pathType: Prefix

backend:

service:

name: nginx-service

port:

number: 80

# **Kubernetes Dashboard**

## **Configuration**

Step 1: Apply manifest

kubectl apply -f <https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml>

Step2: Make svc to NodePort

kubectl get svc -n kubernetes-dashboard

kubectl edit svc/kubernetes-dashboard -n kubernetes-dashboard

type: NodePort

Step 3: Access the dashboard

Access [https://publicipaws:nodeport/](about:blank)

Step 4:

kubectl create serviceaccount dashboard-admin -n kubernetes-dashboard

kubectl create clusterrolebinding cluster-admin-rolebinding --clusterrole=cluster-admin --serviceaccount=kubernetes-dashboard:dashboard-admin

Step5: Create the secret

kubectl create token dashboard-admin -n kubernetes-dashboard

Step 6: Enter the token secret

Enter the token in dashboard UI Kubernetes

**HPA config:**

apiVersion: autoscaling/v2beta2

kind: HorizontalPodAutoscaler

metadata:

name: example-hpa

namespace: default

spec:

scaleTargetRef:

apiVersion: apps/v1

kind: Deployment

name: nginx-deployment

minReplicas: 2

maxReplicas: 10

metrics:

- type: Resource

resource:

name: cpu

targetAverageUtilization: 60

d