How to setup Master Slave Kubernetes cluster - Kishan ray

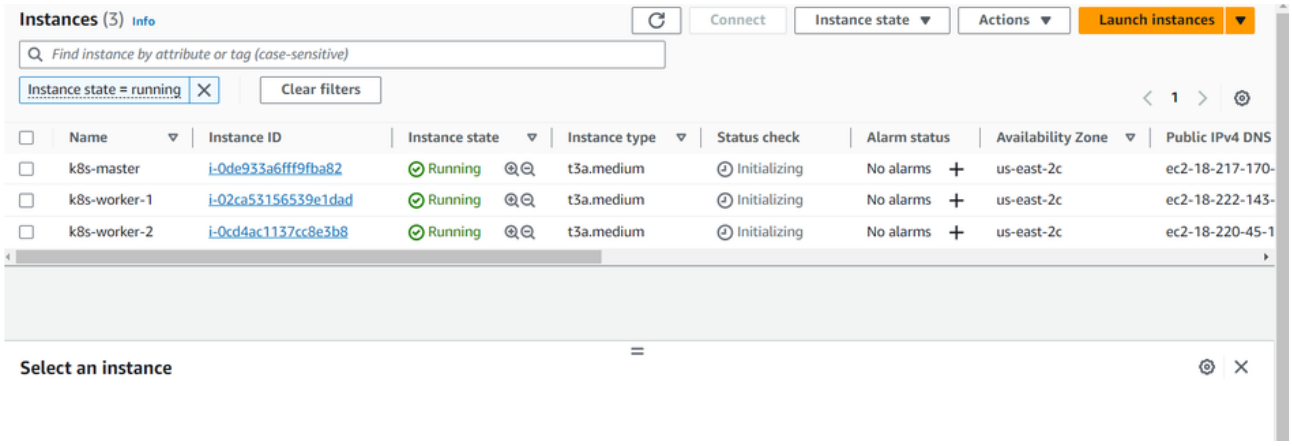
Before you can follow the steps below, you'll need access to a machine running an updated Ubuntu OS (this guide uses 22.04).

Two or more hosts is preferable to ensure that there are multiple worker nodes for your cluster.

reference:- Stand up a k8s cluster on Ubuntu from scratch

Launch 3 ec2-instances using ubuntu 22.04 image and with t3.medium instance type (all traffic open)

Name the instances as master,node1,node2



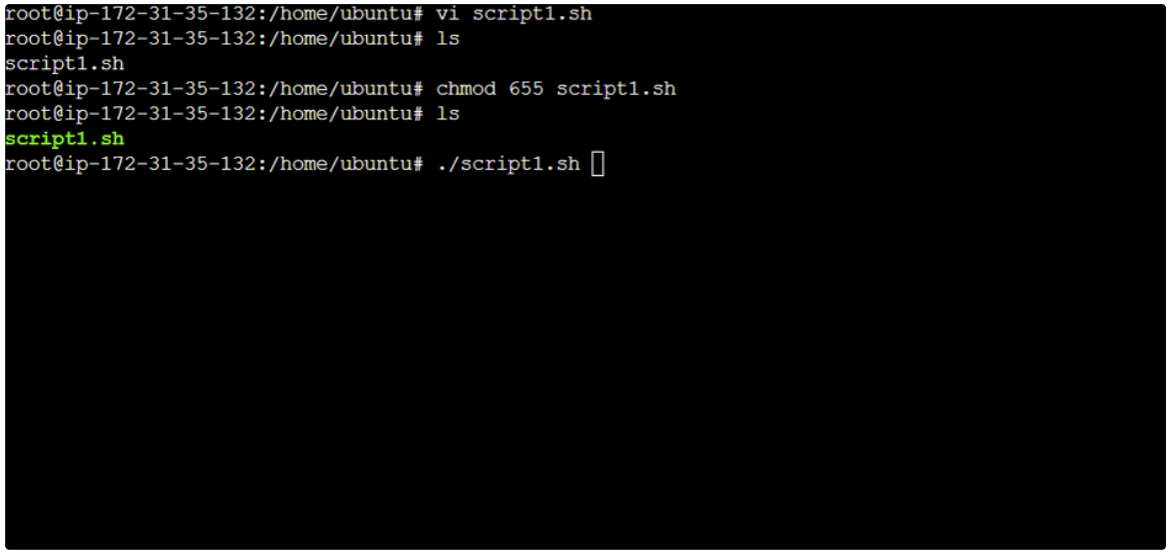
For Installation using kubeadm refer :- Installing kubeadm

Login to all three machine parallel and run script1 file in **all three machine.**

```
sudo su root
clear
```

```
vi script1.sh

#!/bin/bash

sudo swapoff -a

sudo modprobe br_netfilter

echo br_netfilter | sudo tee /etc/modules-load.d/kubernetes.conf

sudo apt update

sudo apt install ca-certificates curl gnupg lsb-release -y


```

```
sudo mkdir -p /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/ubuntu/gpg |
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
 https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
 sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update

sudo apt install -y containerd.io

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

sudo service containerd status
```



Container runtimes

Note: This section links to third party projects that provide functionality required by Kubernetes. The Kubernetes project authors aren't responsible for these projects, which are listed alphabetically. To add a project to this list, read the content guide before submitting a change. More information.

containerd

This section outlines the necessary steps to use containerd as CRI runtime.

To install containerd on your system, follow the instructions on getting started with containerd. Return to this step once you've created a valid `config.toml` configuration file.

- Linux
- Windows

You can find this file under the path `/etc/containerd/config.toml`.

On Linux the default CRI socket for containerd is `/run/containerd/containerd.sock`. On Windows the default CRI endpoint is `npipe://./pipe/containerd-containerd`.

Configuring the `systemd` cgroup driver

To use the `systemd` cgroup driver in `/etc/containerd/config.toml` with `runc`, set

Do in all three machines.

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
  ...
  [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
    SystemdCgroup = true
```

```
vi /etc/containerd/config.toml

edit the line

 [plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]
    SystemdCgroup = true


sudo service containerd restart
```

Create a script2 file and run in **all three machines:-**

ref:- Installing kubeadm

```
vi script2.sh

#!/bin/bash

sudo apt-get update

sudo apt-get install -y apt-transport-https ca-certificates curl

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-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl


sudo apt-get install iproute2

echo 1 > /proc/sys/net/ipv4/ip_forward
```

Now run This command in **master machine only.**

```
kubeadm init --pod-network-cidr=10.244.0.0/16

save token command
```

```
ot@ip-172-31-35-132:/home/ubuntu# kubeadm init --pod-network-cidr=10.244.0.0/16
nit] Using Kubernetes version: v1.28.1
reflight] Running pre-flight checks
reflight] Pulling images required for setting up a Kubernetes cluster
reflight] This might take a minute or two, depending on the speed of your internet connection
reflight] You can also perform this action in beforehand using 'kubeadm config images pull'
```

```
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes control-plane has initialized successfully!

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

Alternatively, if you are the root user, you can run:

  export KUBECONFIG=/etc/kubernetes/admin.conf

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 172.31.35.132:6443 --token rd5vnu.h9hhjxy80iv0u5pp \
        --discovery-token-ca-cert-hash sha256:ad99bc769756e0ad981c3990791ed5fa4fc1fcad653224bde9217cc82a2f28e5
root@ip-172-31-35-132:/home/ubuntu#
```

i-0de933a6fff9fba82 (k8s-master)

PublicIPs: 18.217.170.70 PrivateIPs: 172.31.35.132

Copy the kubeadm join command, for future use-cases.

Now create a script3 file and run it in **master only.**

```
vi script3.sh

#!/bin/bash

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


```

```
root@ip-172-31-35-132:/home/ubuntu# vi script3.sh
root@ip-172-31-35-132:/home/ubuntu# chmod 655 script3.sh
root@ip-172-31-35-132:/home/ubuntu# ls
script1.sh  script2.sh  script3.sh
root@ip-172-31-35-132:/home/ubuntu# ./script3.sh
root@ip-172-31-35-132:/home/ubuntu#
```

Try to run get nodes command in **master machine.**

```
kubectl get nodes
```

```
aws  Services  Search  [Alt+S]
root@ip-172-31-35-132:/home/ubuntu# kubectl get nodes
NAME                STATUS     ROLES           AGE     VERSION
ip-172-31-35-132    NotReady   control-plane   4m50s   v1.28.1
root@ip-172-31-35-132:/home/ubuntu#
```

Its not ready so to make it ready:-

Install flannel one networking tool :- master only

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

```
kubectl get nodes
```

```
t@ip-172-31-35-132:/home/ubuntu# kubectl get nodes
E                STATUS     ROLES           AGE     VERSION
-172-31-35-132   NotReady   control-plane   4m50s   v1.28.1
t@ip-172-31-35-132:/home/ubuntu# kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/master/Documentation/kube-flannel.yml
espace/kube-flannel created
sterrole.rbac.authorization.k8s.io/flannel created
sterrolebinding.rbac.authorization.k8s.io/flannel created
viceaccount/flannel created
figmap/kube-flannel-cfg created
monset.apps/kube-flannel-ds created
t@ip-172-31-35-132:/home/ubuntu# kubectl get nodes
E                STATUS   ROLES           AGE     VERSION
-172-31-35-132   Ready    control-plane   6m40s   v1.28.1
t@ip-172-31-35-132:/home/ubuntu#
```

To join worker node, go to worker node and hit kubeadm join command one by one.

```
This will be different in your case



kubeadm join 172.31.35.132:6443 --token rd5vnu.h9hhjxy80iv0u5pp
--discovery-token-ca-cert-hash
sha256:ad99bc769756e0ad981c3990791ed5fa4fc1fcad653224bde9217cc82a2f28e5
```

```
Services   Search   [Alt+S]   Ohio   devopstraining
t@ip-172-31-46-179:/home/ubuntu# kubeadm join 172.31.35.132:6443 --token rd5vnu.h9hhjxy80iv0u5pp --discovery-token-ca-cert-hash sha256:ad99bc769756e0ad981c3990791ed5fa4
cad653224bde9217cc82a2f28e5
eflight] Running pre-flight checks
eflight] Reading configuration from the cluster...
eflight] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
belet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
belet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubeadm-flags.env"
belet-start] Starting the kubelet
belet-start] Waiting for the kubelet to perform the TLS Bootstrap...

s node has joined the cluster:
rtificate signing request was sent to apiserver and a response was received.
he Kubelet was informed of the new secure connection details.

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

t@ip-172-31-46-179:/home/ubuntu#
```

```
Kubectl get nodes    #in master
```

```
root@ip-172-31-35-132:/home/ubuntu# kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-35-132    Ready    control-plane   6m40s   v1.28.1
root@ip-172-31-35-132:/home/ubuntu# kubectl get nodes
NAME                STATUS   ROLES           AGE     VERSION
ip-172-31-35-132    Ready    control-plane   8m29s   v1.28.1
ip-172-31-46-131    Ready    <none>          33s     v1.28.1
ip-172-31-46-179    Ready    <none>          38s     v1.28.1
root@ip-172-31-35-132:/home/ubuntu#
```

Hence our setup is completed.