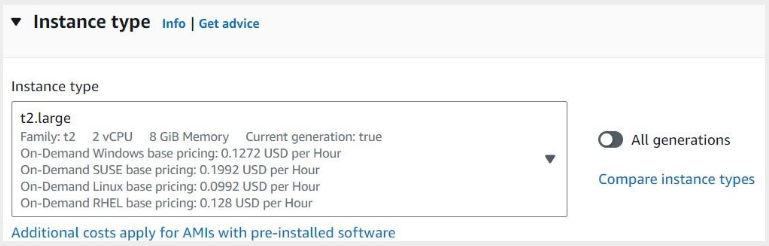
Kubeadm Installation

Reference: https://github.com/yeshwanthlm/Kubeadm-Installation-

Launch 2 instance with t2.large (Master Node & Worker Node)



Instance type Info | Get advice

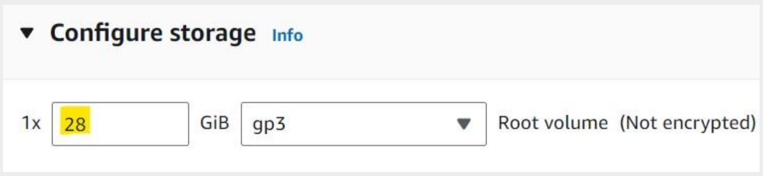
Instance type

t2.large
Family: t2 2 vCPU 8 GiB Memory Current generation: true
On-Demand Windows base pricing: 0.1272 USD per Hour
On-Demand SUSE base pricing: 0.1992 USD per Hour
On-Demand Linux base pricing: 0.0992 USD per Hour
On-Demand RHEL base pricing: 0.128 USD per Hour

All generations

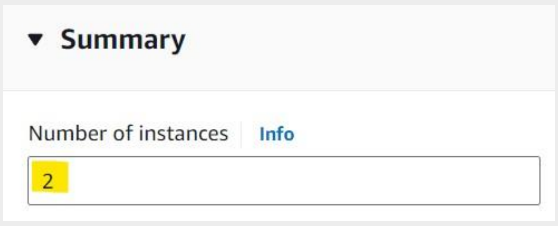
Compare instance types

Additional costs apply for AMIs with pre-installed software



Configure storage Info

1x 28 GiB gp3 Root volume (Not encrypted)



Summary

Number of instances Info

2

Set Hostname for both

```
$ sudo hostname Master
```

```
$ sudo hostname Worker
```

```
$ sudo -i
```

Run commands for Master Node & Worker Node

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

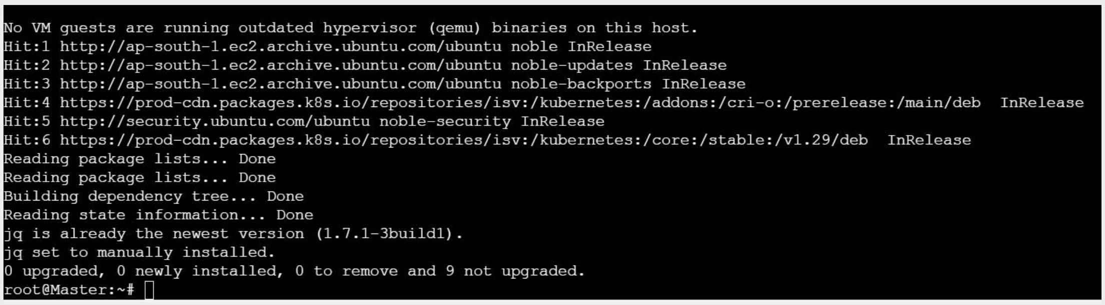
```
 curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sha256"
```

```
 echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
```

```
 sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl  chmod +x kubectl  mkdir -p

~/.local/bin  mv ./kubectl ~/.local/bin/kubectl

 # and then append (or prepend) ~/.local/bin to $PATH
```

```
kubectl version --client


# disable swap sudo

swapoff -a


# Create the .conf file to load the modules at bootup

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

overlay br_netfilter

EOF


sudo modprobe overlay sudo

modprobe br_netfilter


# sysctl params required by setup, params persist across reboots

cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf net.bridge.bridge-nf-

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

net.ipv4.ip_forward                = 1

EOF


# Apply sysctl params without reboot sudo

sysctl --system


## Install CRIO Runtime

sudo apt-get update -y sudo apt-get install -y software-properties-common curl apt-transport-

https ca-certificates gpg


sudo curl -fsSL https://pkgs.k8s.io/addons:/cri-o:/prerelease:/main/deb/Release.key | sudo gpg -
dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg

echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg]
https://pkgs.k8s.io/addons:/crio:/prerelease:/main/deb/ /" | sudo tee /etc/apt/sources.list.d/cri-
o.list
```

```
sudo apt-get update -y

sudo apt-get install -y cri-o


sudo systemctl daemon-reload sudo

systemctl enable crio --now sudo

systemctl start crio.service


echo "CRI runtime installed successfully"


# Add Kubernetes APT repository and install required packages

curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.29/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:/v1.29/deb/ /' | sudo tee /etc/apt/sources.list.d/kubernetes.list


sudo apt-get update -y sudo apt-get install -y kubelet="1.29.0-*" kubectl="1.29.0-

*" kubeadm="1.29.0-*" sudo apt-get update -y sudo apt-get install -y jq


sudo systemctl enable --now kubelet sudo

systemctl start kubelet
```



```
No VM guests are running outdated hypervisor (qemu) binaries on this host.
Hit:1 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://ap-south-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Hit:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/addons:/cri-o:/prerelease:/main/deb  InRelease
Hit:5 http://security.ubuntu.com/ubuntu noble-security InRelease
Hit:6 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.29/deb  InRelease
Reading package lists... Done
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
jq is already the newest version (1.7.1-3build1).
jq set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 9 not upgraded.
root@Master:~#
```

Only for Master Node:

a) Initialize the Kubernetes master node.

```
sudo kubeadm config images pull
```

sudo kubeadm init

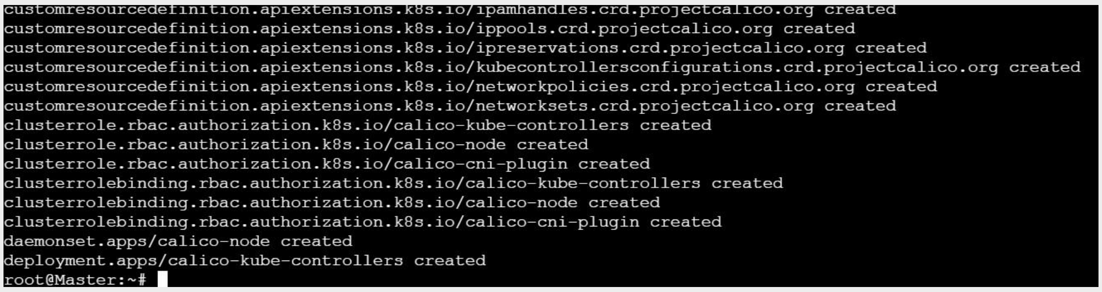
mkdir -p "$HOME"/.kube sudo cp -i

/etc/kubernetes/admin.conf "$HOME"/.kube/config sudo

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

# Network Plugin = calico

kubectl apply -f
https://raw.githubusercontent.com/projectcalico/calico/v3.26.0/manifests/calico.yaml

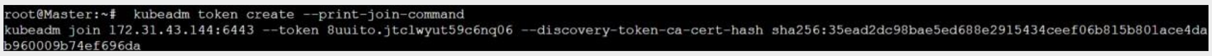


```
customresourcedefinition.apiextensions.k8s.io/ipamhandles.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ippools.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/ipreservations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/kubecontrollersconfigurations.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networkpolicies.crd.projectcalico.org created
customresourcedefinition.apiextensions.k8s.io/networksets.crd.projectcalico.org created
clusterrole.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrole.rbac.authorization.k8s.io/calico-node created
clusterrole.rbac.authorization.k8s.io/calico-cni-plugin created
clusterrolebinding.rbac.authorization.k8s.io/calico-kube-controllers created
clusterrolebinding.rbac.authorization.k8s.io/calico-node created
clusterrolebinding.rbac.authorization.k8s.io/calico-cni-plugin created
daemonset.apps/calico-node created
deployment.apps/calico-kube-controllers created
root@Master:~#
```

After succesfully running, your Kubernetes control plane will be initialized successfully.

b) Generate a token for worker nodes to join:

kubeadm token create --print-join-command

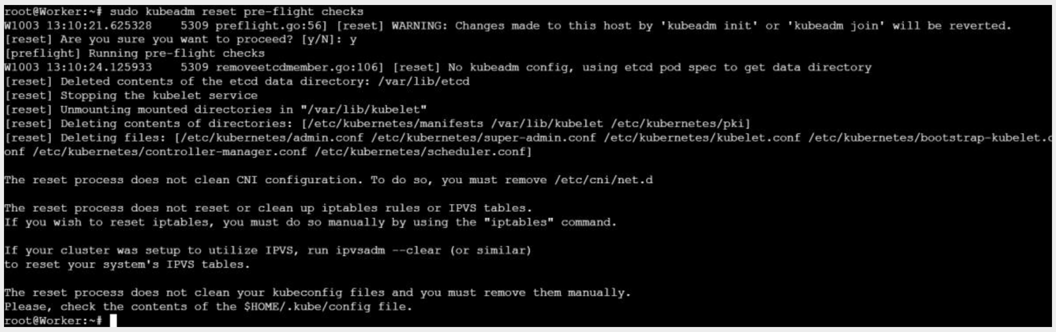


```
root@Master:~#  kubeadm token create --print-join-command
kubeadm join 172.31.43.144:6443 --token 8uuito.jtclwyut59c6nq06 --discovery-token-ca-cert-hash sha256:35ead2dc98bae5ed688e2915434ceef06b815b801ace4da
b960009b74ef696da
```

c) Expose port 6443 in the Security group for the Worker to connect to Master Node

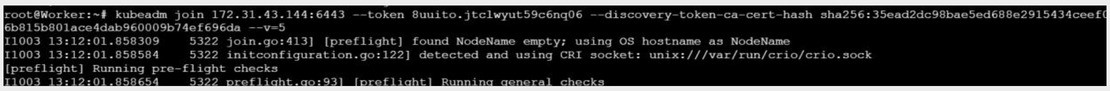
Only for Worker Node: sudo

kubeadm reset pre-flight checks

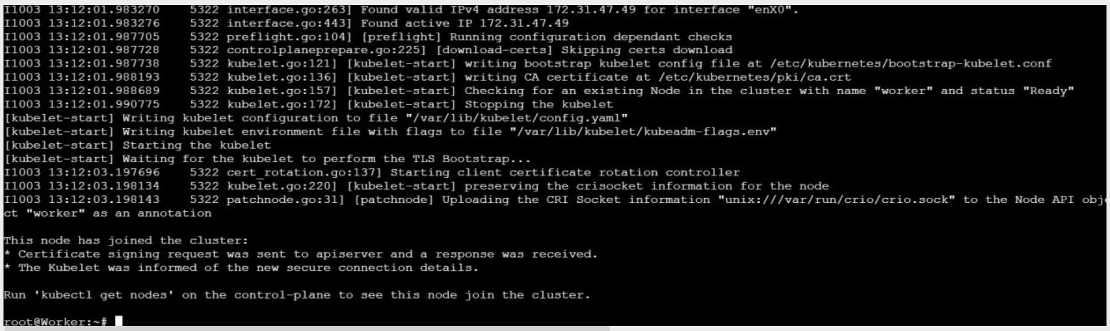


```
root@Worker:~# sudo kubeadm reset pre-flight checks
W1003 13:10:21.625328    5309 preflight.go:56] [reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]: y
[preflight] Running pre-flight checks
W1003 13:10:24.125933    5309 removeetcdmember.go:106] [reset] No kubeadm config, using etcd pod spec to get data directory
[reset] Deleted contents of the etcd data directory: /var/lib/etcd
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Deleting contents of directories: [/etc/kubernetes/manifests /var/lib/kubelet /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/super-admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.c
onf /etc/kubernetes/controller-manager.conf /etc/kubernetes/scheduler.conf]

The reset process does not clean CNI configuration. To do so, you must remove /etc/cni/net.d

The reset process does not reset or clean up iptables rules or IPVS tables.
If you wish to reset iptables, you must do so manually by using the "iptables" command.

If your cluster was setup to utilize IPVS, run ipvsadm --clear (or similar)
to reset your system's IPVS tables.

The reset process does not clean your kubeconfig files and you must remove them manually.
Please, check the contents of the $HOME/.kube/config file.
root@Worker:~#
```

Paste the join command you got from the master node and append --v=5 at the end. Make sure either you are working as sudo user or usesudo before the command kubeadm join 172.31.43.144:6443 --token 8uuito.jtclwyut59c6nq06 --discovery-token-ca-cert-hash sha256:35ead2dc98bae5ed688e2915434ceef06b815b801ace4dab960009b74ef696da --v=5



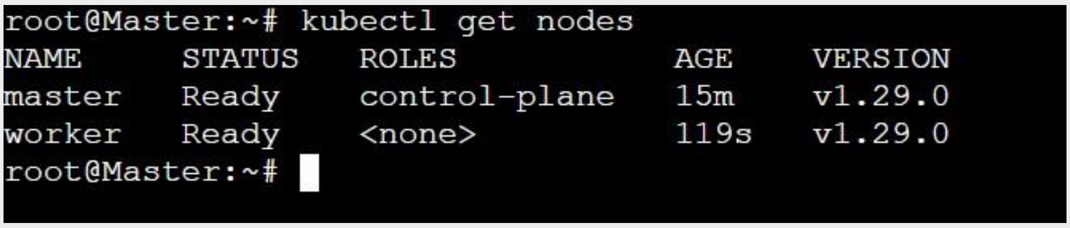
```
root@Worker:~# kubeadm join 172.31.43.144:6443 --token 8uuito.jtclwyut59c6nq06 --discovery-token-ca-cert-hash sha256:35ead2dc98bae5ed688e2915434ceef0
6b815b801ace4dab960009b74ef696da --v=5
I1003 13:12:01.858309    5322 join.go:413] [preflight] found NodeName empty; using OS hostname as NodeName
I1003 13:12:01.858584    5322 initconfiguration.go:122] detected and using CRI socket: unix:///var/run/crio/crio.sock
[preflight] Running pre-flight checks
I1003 13:12:01.858654    5322 preflight.go:93] [preflight] Running general checks
```



```
I1003 13:12:01.983270    5322 interface.go:263] Found valid IPv4 address 172.31.47.49 for interface "enX0".
I1003 13:12:01.983276    5322 interface.go:443] Found active IP 172.31.47.49
I1003 13:12:01.987705    5322 preflight.go:104] [preflight] Running configuration dependant checks
I1003 13:12:01.987728    5322 controlplaneprepare.go:225] [download-certs] Skipping certs download
I1003 13:12:01.987738    5322 kubelet.go:121] [kubelet-start] writing bootstrap kubelet config file at /etc/kubernetes/bootstrap-kubelet.conf
I1003 13:12:01.988193    5322 kubelet.go:136] [kubelet-start] writing CA certificate at /etc/kubernetes/pki/ca.crt
I1003 13:12:01.988689    5322 kubelet.go:157] [kubelet-start] Checking for an existing Node in the cluster with name "worker" and status "Ready"
I1003 13:12:01.990775    5322 kubelet.go:172] [kubelet-start] Stopping the kubelet
[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...
I1003 13:12:03.197696    5322 cert_rotation.go:137] Starting client certificate rotation controller
I1003 13:12:03.198134    5322 kubelet.go:220] [kubelet-start] preserving the crisocket information for the node
I1003 13:12:03.198143    5322 patchnode.go:31] [patchnode] Uploading the CRI Socket information "unix:///var/run/crio/crio.sock" to the Node API obj
ct "worker" as an annotation

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.

root@Worker:~#
```

Verify if it is working as expected on Worker Node:

kubectl get nodes



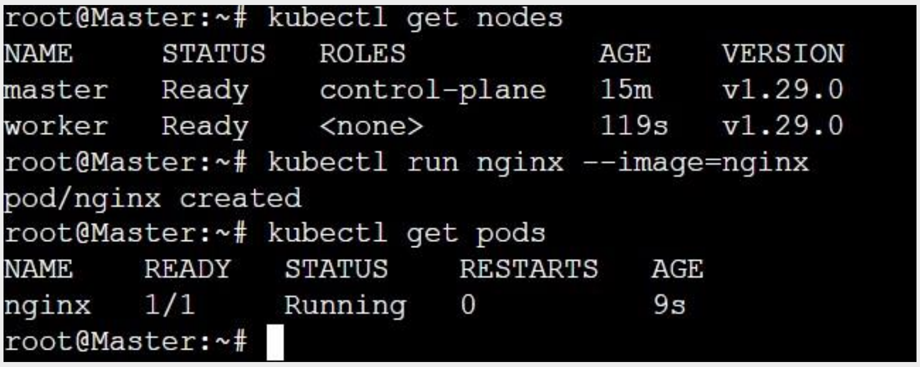
```
root@Master:~# kubectl get nodes
NAME     STATUS   ROLES           AGE    VERSION
master   Ready    control-plane   15m    v1.29.0
worker   Ready    <none>          119s   v1.29.0
root@Master:~#
```

Just install nginx:

$ kubectl get nodes

$ kubectl run nginx --image=nginx

$ kubectl get pods



```
root@Master:~# kubectl get nodes
NAME     STATUS   ROLES           AGE    VERSION
master   Ready    control-plane   15m    v1.29.0
worker   Ready    <none>          119s   v1.29.0
root@Master:~# kubectl run nginx --image=nginx
pod/nginx created
root@Master:~# kubectl get pods
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          9s
root@Master:~#
```