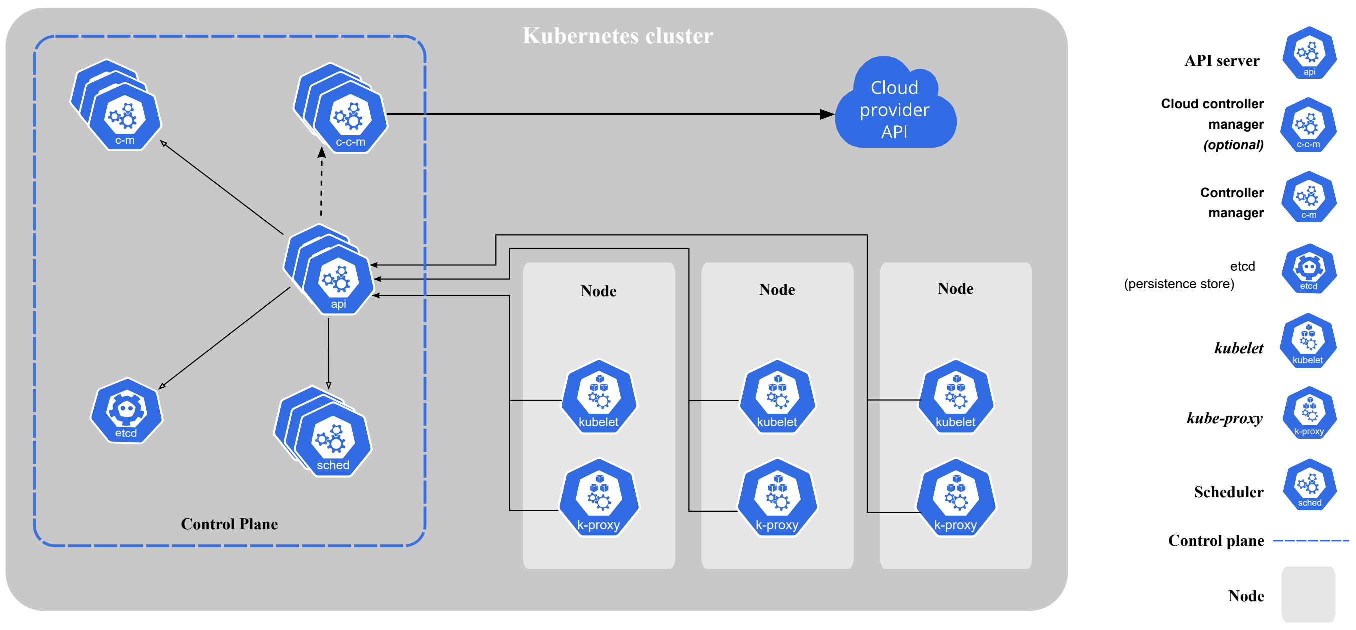
Kubernetes Architecture Explained

This document explains the key components that make up the architecture of a Kubernetes cluster, in simple terms.

Table of Contents

- Control Plane (Master Node Components)
- Worker Node Components
- Other Components



Control Plane (Master Node Components)

API Server

This is the "front desk" of Kubernetes. Whenever you want to interact with your cluster, your request goes through the API Server. It validates and processes these requests to the backend components.

etcd

Think of this as the "database" of Kubernetes. It stores all the information about your cluster—what nodes are part of the cluster, what pods are running, what their statuses are, and more.

Scheduler

The "event planner" for your containers. When you ask for a container to be run, the Scheduler decides which machine (Node) in your cluster should run it. It considers resource availability and other constraints while making this decision.

Controller Manager

Imagine a bunch of small robots that continuously monitor the cluster to make sure everything is running smoothly. If something goes wrong (e.g., a Pod crashes), they work to fix it, ensuring the cluster state matches your desired state.

Cloud Controller Manager

This is a specialized component that allows Kubernetes to interact with the underlying cloud provider, like AWS or Azure. It helps in tasks like setting up load balancers and persistent storage.

Worker Node Components

kubelet

This is the "manager" for each worker node. It ensures all containers on the node are healthy and running as they should be.

kube-proxy

Think of this as the "traffic cop" for network communication either between Pods or from external clients to Pods. It helps in routing the network traffic appropriately.

Container Runtime

This is the software used to run containers. Docker is commonly used, but other runtimes like containerd can also be used.

Other Components

Pod

The smallest unit in Kubernetes, a Pod is a group of one or more containers. Think of it like an apartment in an apartment building.

Service

This is like a phone directory for Pods. Since Pods can come and go, a Service provides a stable "address" so that other parts of your application can find them.

Volume

This is like an external hard-drive that can be attached to a Pod to store data.

Namespace

A way to divide cluster resources among multiple users or teams. Think of it as having different folders on a shared computer, where each team can only see their own folder.

Ingress

Think of this as the "front door" for external access to your applications, controlling how HTTP and HTTPS traffic should be routed to your services.

And there you have it! That's a simplified breakdown of Kubernetes architecture components.

```
```

Minikube Installation Guide for Ubuntu

This guide provides step-by-step instructions for installing Minikube on Ubuntu. Minikube allows you to run a single-node Kubernetes cluster locally for development and testing purposes.

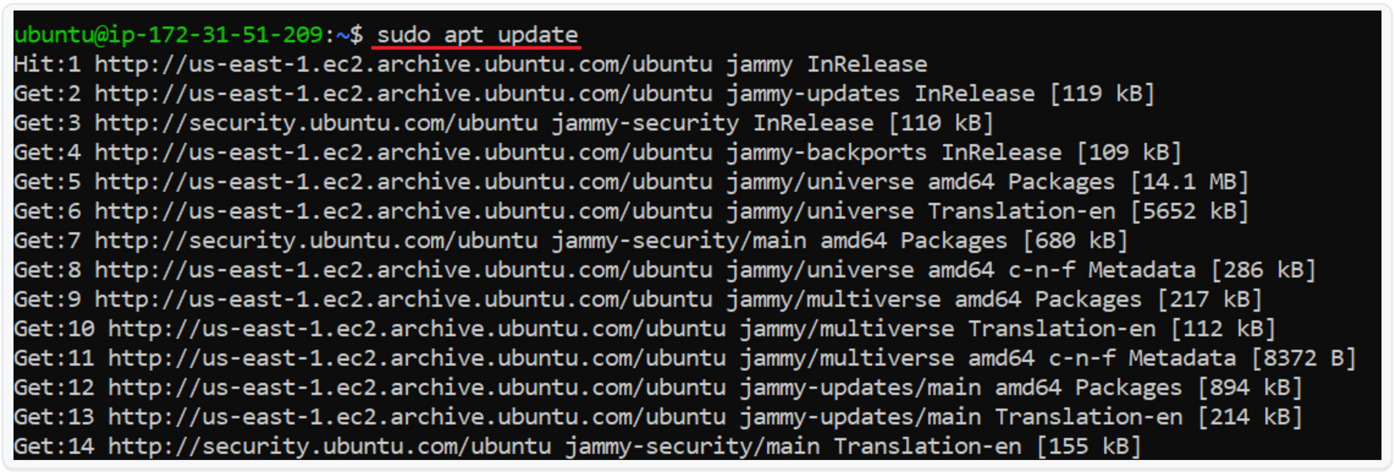
Pre-requisites

- Ubuntu OS
- sudo privileges
- Internet access
- Virtualization support enabled (Check with `egrep -c '(vmx|svm)' /proc/cpuinfo`)

Step 1: Update System Packages

Update your package lists to make sure you are getting the latest version and dependencies.

```
sudo apt update
```

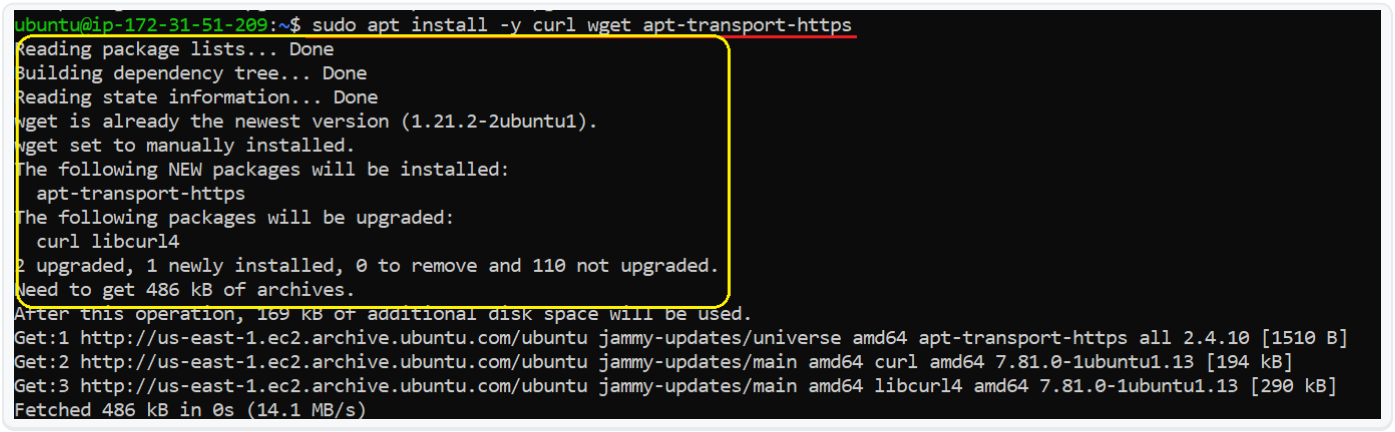


```
ubuntu@ip-172-31-51-209:~$ sudo apt update
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [680 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [894 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [214 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [155 kB]
```

Step 2: Install Required Packages

Install some basic required packages.

```
sudo apt install -y curl wget apt-transport-https
```

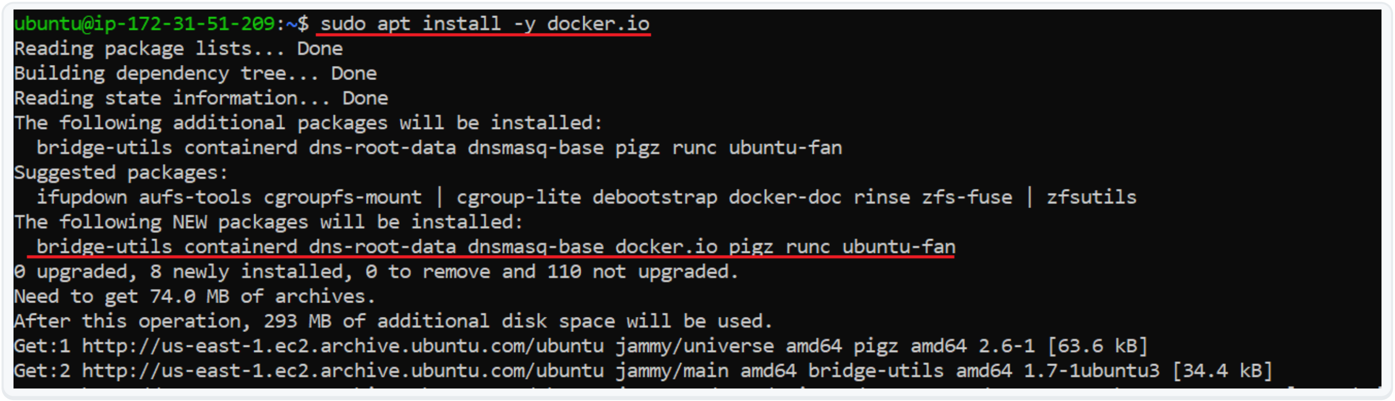


```
ubuntu@ip-172-31-51-209:~$ sudo apt install -y curl wget apt-transport-https
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
wget is already the newest version (1.21.2-2ubuntu1).
wget set to manually installed.
The following NEW packages will be installed:
  apt-transport-https
The following packages will be upgraded:
  curl libcurl4
2 upgraded, 1 newly installed, 0 to remove and 110 not upgraded.
Need to get 486 kB of archives.
After this operation, 169 kB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 apt-transport-https all 2.4.10 [1510 B]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 curl amd64 7.81.0-1ubuntu1.13 [194 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libcurl4 amd64 7.81.0-1ubuntu1.13 [290 kB]
Fetched 486 kB in 0s (14.1 MB/s)
```

Step 3: Install Docker

Minikube can run a Kubernetes cluster either in a VM or locally via Docker. This guide demonstrates the Docker method.

```
sudo apt install -y docker.io
```



```
ubuntu@ip-172-31-51-209:~$ sudo apt install -y docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 110 not upgraded.
Need to get 74.0 MB of archives.
After this operation, 293 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
```

Start and enable Docker.

```
sudo systemctl start docker
sudo systemctl enable docker
```

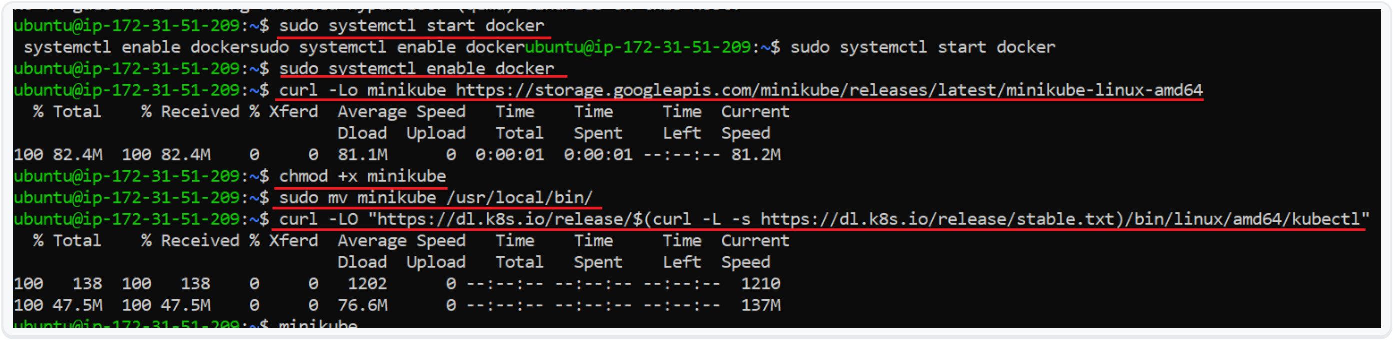
Step 4: Install Minikube

First, download the Minikube binary using `curl`:

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-li
```

Make it executable and move it into your path:

```
chmod +x minikube
sudo mv minikube /usr/local/bin/
```



```
ubuntu@ip-172-31-51-209:~$ sudo systemctl start docker
 systemctl enable dockersudo systemctl enable dockerubuntu@ip-172-31-51-209:~$ sudo systemctl start docker
ubuntu@ip-172-31-51-209:~$ sudo systemctl enable docker
ubuntu@ip-172-31-51-209:~$ curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 82.4M  100 82.4M    0     0  81.1M      0  0:00:01  0:00:01 --:--:-- 81.2M
ubuntu@ip-172-31-51-209:~$ chmod +x minikube
ubuntu@ip-172-31-51-209:~$ sudo mv minikube /usr/local/bin/
ubuntu@ip-172-31-51-209:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0   1202      0 --:--:-- --:--:-- --:--:--  1210
100 47.5M  100 47.5M    0     0  76.6M      0 --:--:-- --:--:-- --:--:--  137M
```

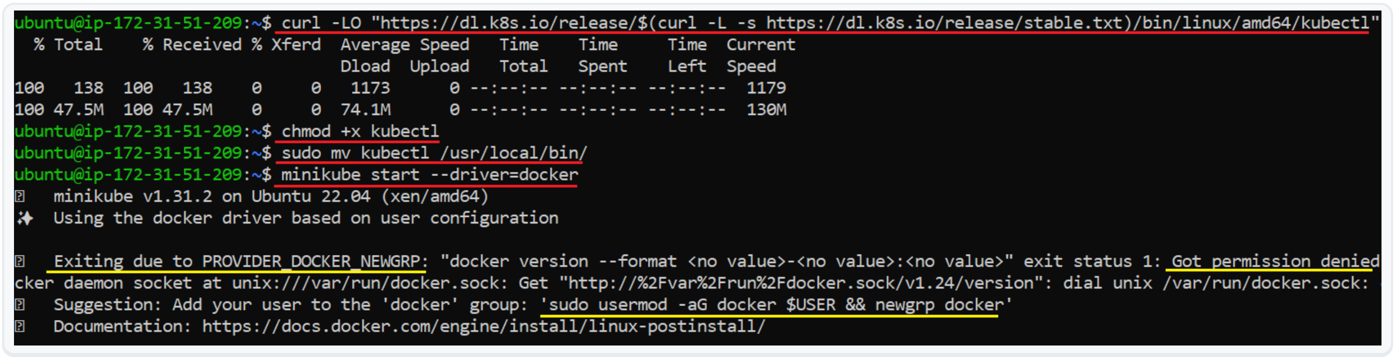
Step 5: Install kubectl

Download kubectl, which is a Kubernetes command-line tool.

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt
```

Check above image ⬆️ Make it executable and move it into your path:

```
chmod +x kubectl
sudo mv kubectl /usr/local/bin/
```



```
ubuntu@ip-172-31-51-209:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0   1173      0 --:--:-- --:--:-- --:--:--  1179
100 47.5M  100 47.5M    0     0  74.1M      0 --:--:-- --:--:-- --:--:--  130M
ubuntu@ip-172-31-51-209:~$ chmod +x kubectl
ubuntu@ip-172-31-51-209:~$ sudo mv kubectl /usr/local/bin/
ubuntu@ip-172-31-51-209:~$ minikube start --driver=docker
    minikube v1.31.2 on Ubuntu 22.04 (xen/amd64)
✨  Using the docker driver based on user configuration

    Exiting due to PROVIDER_DOCKER_NEWGRP: "docker version --format <no value>-<no value>:<no value>" exit status 1: Got permission denied
cker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/version": dial unix /var/run/docker.sock:
    Suggestion: Add your user to the 'docker' group: 'sudo usermod -aG docker $USER && newgrp docker'
    Documentation: https://docs.docker.com/engine/install/linux-postinstall/
```

Step 6: Start Minikube

Now, you can start Minikube with the following command:

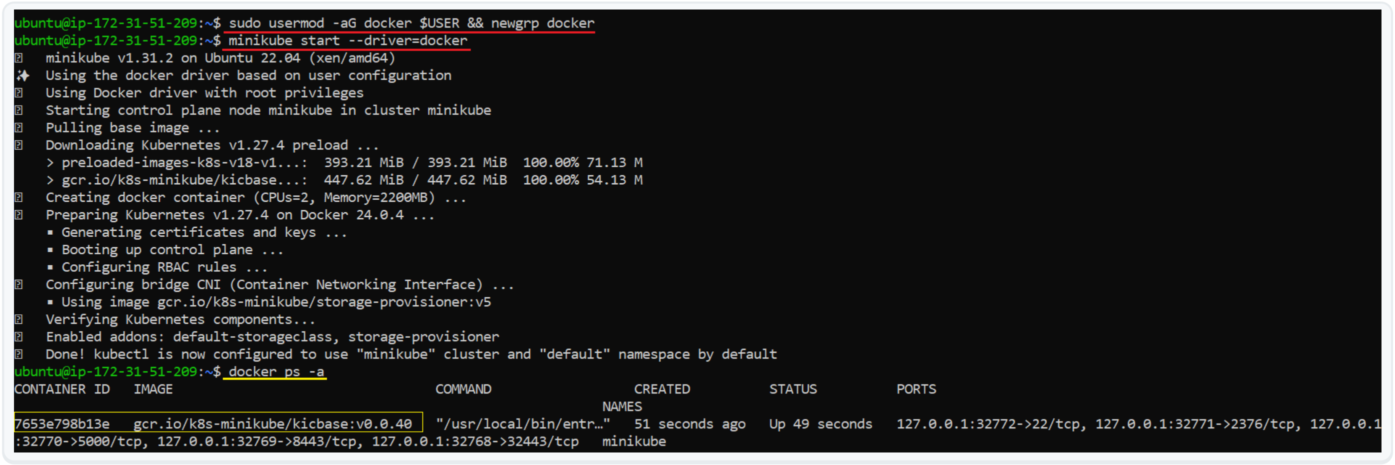
```
minikube start --driver=docker
```

This command will start a single-node Kubernetes cluster inside a Docker container.

Step 7: Check Cluster Status

Check the cluster status with:

```
minikube status
```



```
ubuntu@ip-172-31-51-209:~$ sudo usermod -aG docker $USER && newgrp docker
ubuntu@ip-172-31-51-209:~$ minikube start --driver=docker
    minikube v1.31.2 on Ubuntu 22.04 (xen/amd64)
✨  Using the docker driver based on user configuration
    Using Docker driver with root privileges
    Starting control plane node minikube in cluster minikube
    Pulling base image ...
    Downloading Kubernetes v1.27.4 preload ...
    > preloaded-images-k8s-v18-v1...:  393.21 MiB / 393.21 MiB  100.00% 71.13 M
    > gcr.io/k8s-minikube/kicbase...:  447.62 MiB / 447.62 MiB  100.00% 54.13 M
    Creating docker container (CPUs=2, Memory=2200MB) ...
    Preparing Kubernetes v1.27.4 on Docker 24.0.4 ...
    ▪ Generating certificates and keys ...
    ▪ Booting up control plane ...
    ▪ Configuring RBAC rules ...
    Configuring bridge CNI (Container Networking Interface) ...
    ▪ Using image gcr.io/k8s-minikube/storage-provisioner:v5
    Verifying Kubernetes components...
    Enabled addons: default-storageclass, storage-provisioner
    Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
ubuntu@ip-172-31-51-209:~$ docker ps -a
CONTAINER ID   IMAGE                                 COMMAND                  CREATED          STATUS          PORTS
                                                  NAMES
7653e798b13e   gcr.io/k8s-minikube/kicbase:v0.0.40   "/usr/local/bin/entr…"   51 seconds ago   Up 49 seconds   127.0.0.1:32772->22/tcp, 127.0.0.1:32771->2376/tcp, 127.0.0.1
:32770->5000/tcp, 127.0.0.1:32769->8443/tcp, 127.0.0.1:32768->32443/tcp   minikube
```

You can also use `kubectl` to interact with your cluster:

```
kubectl get nodes
```

Step 8: Stop Minikube

When you are done, you can stop the Minikube cluster with:

```
minikube stop
```

Optional: Delete Minikube Cluster

If you wish to delete the Minikube cluster entirely, you can do so with:

```
minikube delete
```

That's it! You've successfully installed Minikube on Ubuntu, and you can now start deploying Kubernetes applications for development and testing.

```
```

Kubeadm Installation Guide

This guide outlines the steps needed to set up a Kubernetes cluster using kubeadm.

Pre-requisites

- Ubuntu OS (Xenial or later)
- sudo privileges
- Internet access
- t2.medium instance type or higher

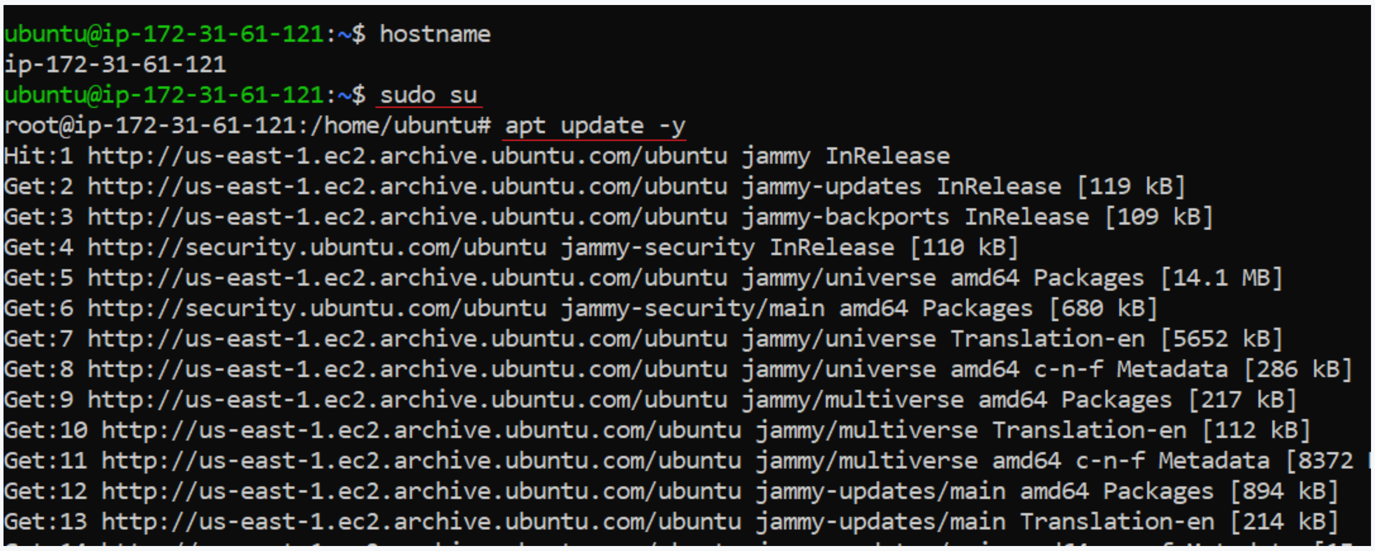
Both Master & Worker Node

Run the following commands on both the master and worker nodes to prepare them for kubeadm.

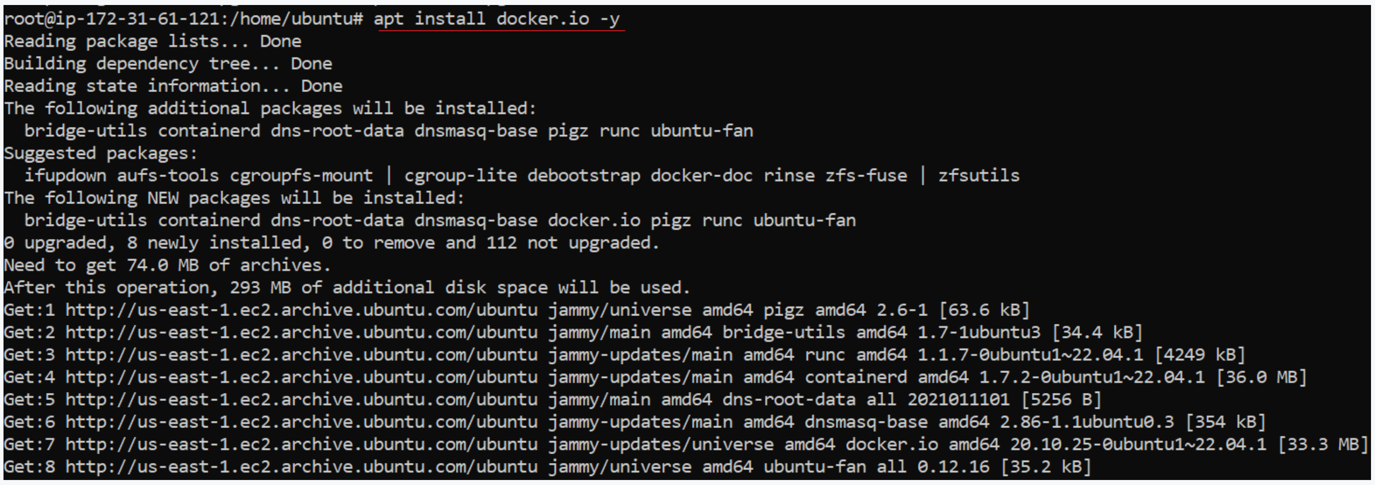
```
sudo su
apt update -y
apt install docker.io -y

systemctl start docker
systemctl enable docker

curl -fsSL "https://packages.cloud.google.com/apt/doc/apt-key.gpg" | sudo gpg --dearm
echo 'deb https://packages.cloud.google.com/apt kubernetes-xenial main' > /etc/apt/so
```

```
apt update -y
apt install kubeadm=1.20.0-00 kubectl=1.20.0-00 kubelet=1.20.0-00 -y
```

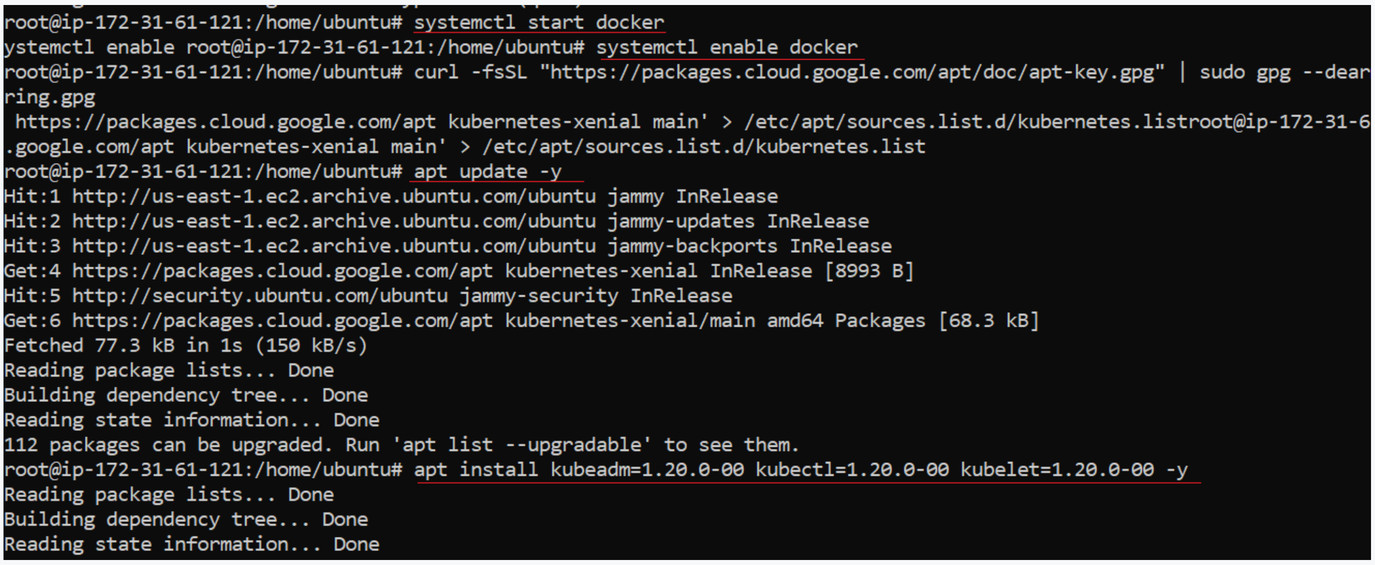
Sample Command run on master node



```
ubuntu@ip-172-31-61-121:~$ hostname
ip-172-31-61-121
ubuntu@ip-172-31-61-121:~$ sudo su
root@ip-172-31-61-121:/home/ubuntu# apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [680 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [894 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [214 kB]
```



```
root@ip-172-31-61-121:/home/ubuntu# apt install docker.io -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
The following NEW packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base docker.io pigz runc ubuntu-fan
0 upgraded, 8 newly installed, 0 to remove and 112 not upgraded.
Need to get 74.0 MB of archives.
After this operation, 293 MB of additional disk space will be used.
Get:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 pigz amd64 2.6-1 [63.6 kB]
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 bridge-utils amd64 1.7-1ubuntu3 [34.4 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 runc amd64 1.1.7-0ubuntu1~22.04.1 [4249 kB]
Get:4 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 containerd amd64 1.7.2-0ubuntu1~22.04.1 [36.0 MB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/main amd64 dns-root-data all 2021011101 [5256 B]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dnsmasq-base amd64 2.86-1.1ubuntu0.3 [354 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 docker.io amd64 20.10.25-0ubuntu1~22.04.1 [33.3 MB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 ubuntu-fan all 0.12.16 [35.2 kB]
```

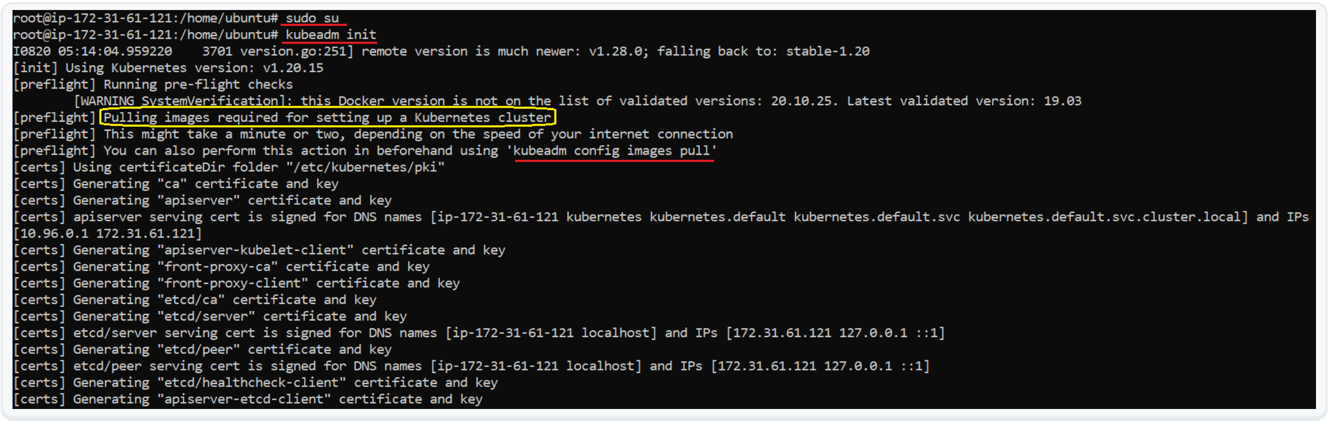


```
root@ip-172-31-61-121:/home/ubuntu# systemctl start docker
ystemctl enable root@ip-172-31-61-121:/home/ubuntu# systemctl enable docker
root@ip-172-31-61-121:/home/ubuntu# curl -fsSL "https://packages.cloud.google.com/apt/doc/apt-key.gpg" | sudo gpg --dear
ring.gpg
 https://packages.cloud.google.com/apt kubernetes-xenial main' > /etc/apt/sources.list.d/kubernetes.listroot@ip-172-31-6
.google.com/apt kubernetes-xenial main' > /etc/apt/sources.list.d/kubernetes.list
root@ip-172-31-61-121:/home/ubuntu# apt update -y
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Get:4 https://packages.cloud.google.com/apt kubernetes-xenial InRelease [8993 B]
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Get:6 https://packages.cloud.google.com/apt kubernetes-xenial/main amd64 Packages [68.3 kB]
Fetched 77.3 kB in 1s (150 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
112 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ip-172-31-61-121:/home/ubuntu# apt install kubeadm=1.20.0-00 kubectl=1.20.0-00 kubelet=1.20.0-00 -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
```

Master Node

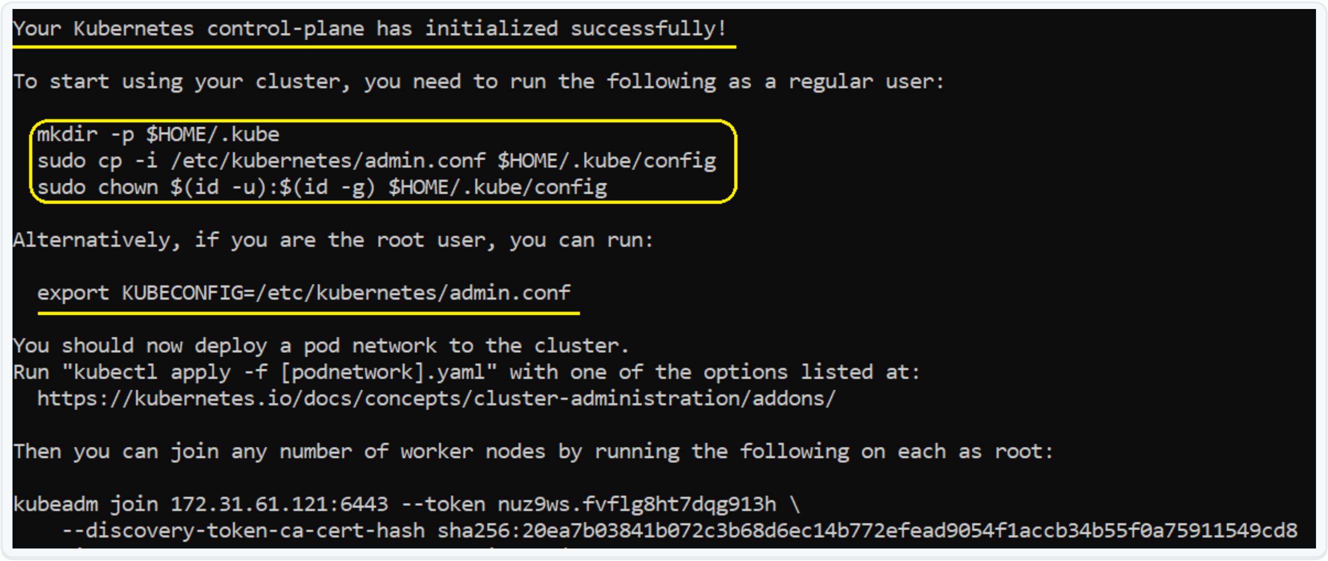
1. Initialize the Kubernetes master node.

```
sudo su
kubeadm init
```



```
root@ip-172-31-61-121:/home/ubuntu# sudo su
root@ip-172-31-61-121:/home/ubuntu# kubeadm init
I0820 05:14:04.959220    3701 version.go:251] remote version is much newer: v1.28.0; falling back to: stable-1.20
[init] Using Kubernetes version: v1.20.15
[preflight] Running pre-flight checks
        [WARNING SystemVerification]: this Docker version is not on the list of validated versions: 20.10.25. Latest validated version: 19.03
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connection
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull'
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [ip-172-31-61-121 kubernetes kubernetes.default kubernetes.default.svc kubernetes.default.svc.cluster.local] and IPs
[10.96.0.1 172.31.61.121]
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [ip-172-31-61-121 localhost] and IPs [172.31.61.121 127.0.0.1 ::1]
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [ip-172-31-61-121 localhost] and IPs [172.31.61.121 127.0.0.1 ::1]
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "apiserver-etcd-client" certificate and key
```

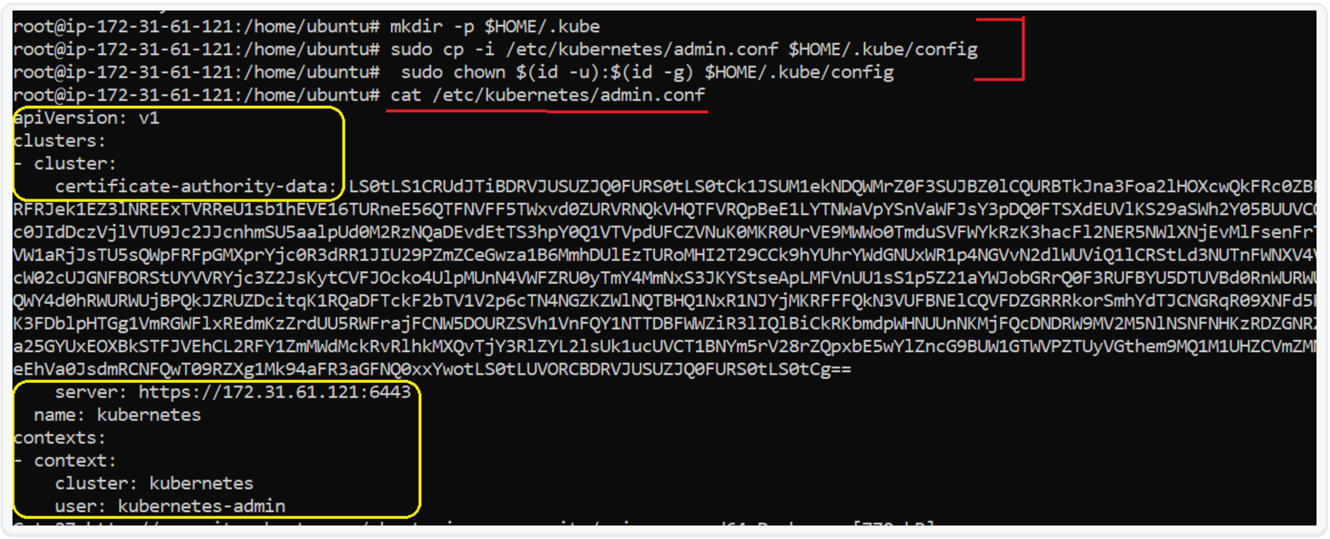
After succesfully running, your Kubernetes control plane will be initialized successfully.



```
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 172.31.61.121:6443 --token nuz9ws.fvflg8ht7dqg913h \
    --discovery-token-ca-cert-hash sha256:20ea7b03841b072c3b68d6ec14b772efead9054f1accb34b55f0a75911549cd8
```

2. Set up local kubeconfig (both for root user and normal user):

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

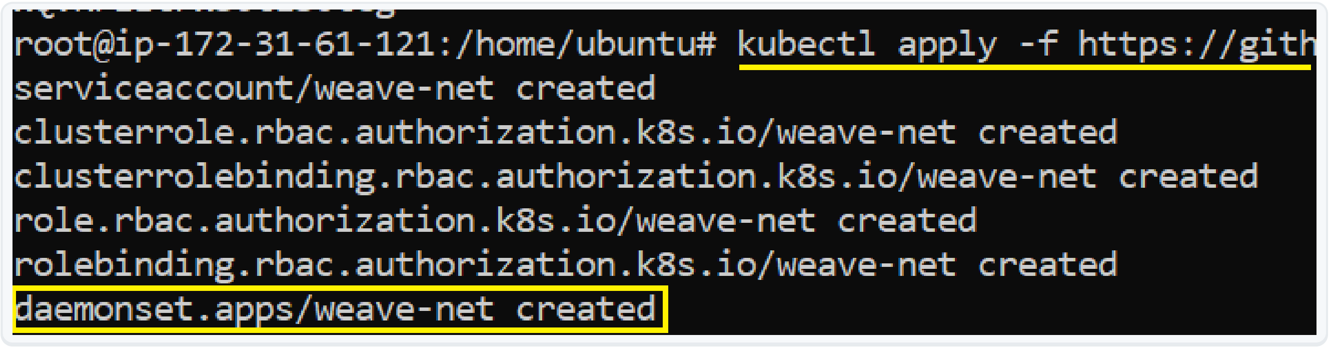


```
root@ip-172-31-61-121:/home/ubuntu# mkdir -p $HOME/.kube
root@ip-172-31-61-121:/home/ubuntu# sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
root@ip-172-31-61-121:/home/ubuntu#  sudo chown $(id -u):$(id -g) $HOME/.kube/config
root@ip-172-31-61-121:/home/ubuntu# cat /etc/kubernetes/admin.conf
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUM1ekNDQWMrZ0F3SUJBZ0lCQURBTkJna3Foa2lHOXcwQkFRc0ZB
RFRJek1EZ3lNREExTVRReU1sb1hEVE16TURneE56QTFNVFF5TWxvd0ZURVRNQkVHQTFVRQpBeE1LYTNWaVpYSnVaWFJsY3pDQ0FTSXdEUVlKS29aSWh2Y05BUUVC
c0JIdDczVjlVTU9Jc2JJcnhmSU5aalpUd0M2RzNQaDEvdEtTS3hpY0Q1VTVpdUFCZVNuK0MKR0UrVE9MWWo0TmduSVFWYkRzK3hacFl2NER5NWlXNjEvMlFsenFr
VW1aRjJsTU5sQWpFRFpGMXprYjc0R3dRR1JIU29PZmZCeGwza1B6MmhDUlEzTURoMHI2T29CCk9hYUhrYWdGNUxWR1p4NGVvN2dlWUViQ1lCRStLd3NUTnFWNXV4
cW02cUJGNFBORStUYVVRYjc3Z2JsKytCVFJOcko4UlpMUnN4VWFZRU0yTmY4MmNxS3JKYStseApLMFVnUU1sS1p5Z21aYWJobGRrQ0F3RUFBYU5DTUVBd0RnWURW
QWY4d0hRWURWUjBPQkJZRUZDcitqK1RQaDFTckF2bTV1V2p6cTN4NGZKZWlNQTBHQ1NxR1NJYjMKRFFFQkN3VUFBNElCQVFDZGRRRkorSmhYdTJCNGRqR09XNFd5
K3FDblpHTGg1VmRGWFlxREdmKzZrdUU5RWFrajFCNW5DOURZSVh1VnFQY1NTTDBFWWZiR3lIQlBiCkRKbmdpWHNUUnNKMjFQcDNDRW9MV2M5NlNSNFNHKzRDZGNR
a25GYUxEOXBkSTFJVEhCL2RFY1ZmMWdMckRvRlhkMXQvTjY3RlZYL2lsUk1ucUVCT1BNYm5rV28rZQpxbE5wYlZncG9BUW1GTWVPZTUyVGthem9MQ1M1UHZCVmZM
eEhVa0JsdmRCNFQwT09RZXg1Mk94aFR3aGFNQ0xxYwotLS0tLUVORCBDRVJUSUZJQ0FURS0tLS0tCg==
    server: https://172.31.61.121:6443
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
```

3. Apply Weave network:

feature/add-ima... kubernetes-kickstarter / kubeadm_installation.md ↑ Top

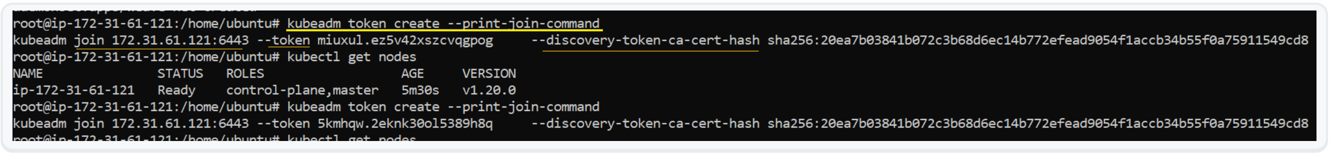
Preview Code Blame Raw



```
root@ip-172-31-61-121:/home/ubuntu# kubectl apply -f https://gith
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
```

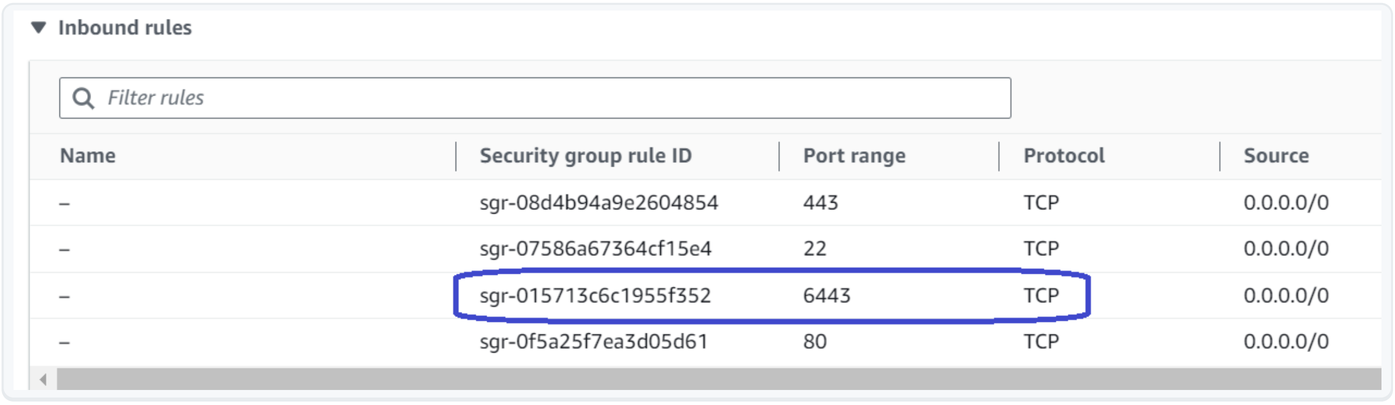
4. Generate a token for worker nodes to join:

```
kubeadm token create --print-join-command
```



```
root@ip-172-31-61-121:/home/ubuntu# kubeadm token create --print-join-command
kubeadm join 172.31.61.121:6443 --token miuxul.ez5v42xszcvqgpog     --discovery-token-ca-cert-hash sha256:20ea7b03841b072c3b68d6ec14b772efead9054f1accb34b55f0a75911549cd8
root@ip-172-31-61-121:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES                  AGE     VERSION
ip-172-31-61-121   Ready    control-plane,master   5m30s   v1.20.0
root@ip-172-31-61-121:/home/ubuntu# kubeadm token create --print-join-command
kubeadm join 172.31.61.121:6443 --token 5kmhqw.2eknk30ol5389h8q     --discovery-token-ca-cert-hash sha256:20ea7b03841b072c3b68d6ec14b772efead9054f1accb34b55f0a75911549cd8
root@ip-172-31-61-121:/home/ubuntu# kubectl get nodes
```

5. Expose port 6443 in the Security group for the Worker to connect to Master Node



▼ Inbound rules

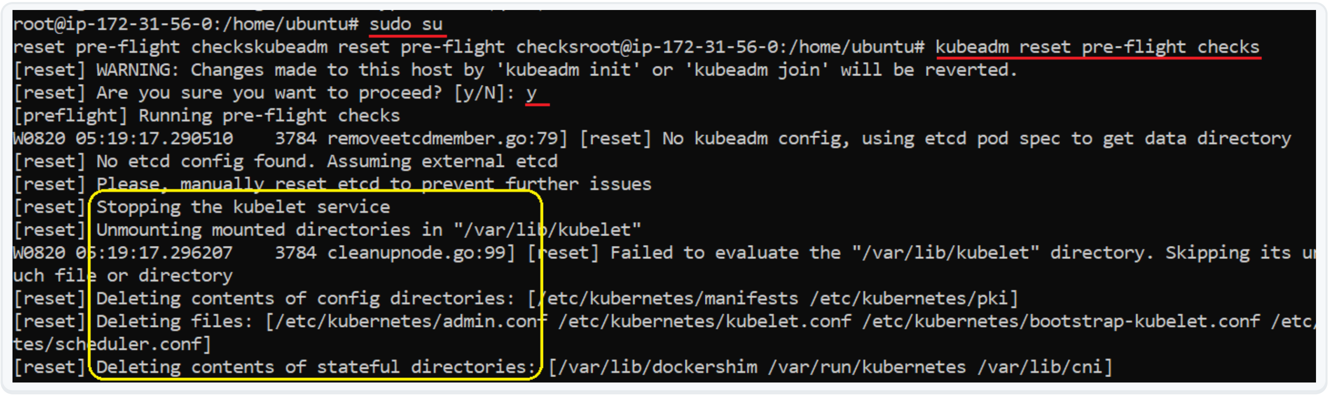
Filter rules

Name	Security group rule ID	Port range	Protocol	Source
–	sgr-08d4b94a9e2604854	443	TCP	0.0.0.0/0
–	sgr-07586a67364cf15e4	22	TCP	0.0.0.0/0
–	sgr-015713c6c1955f352	6443	TCP	0.0.0.0/0
–	sgr-0f5a25f7ea3d05d61	80	TCP	0.0.0.0/0

Worker Node

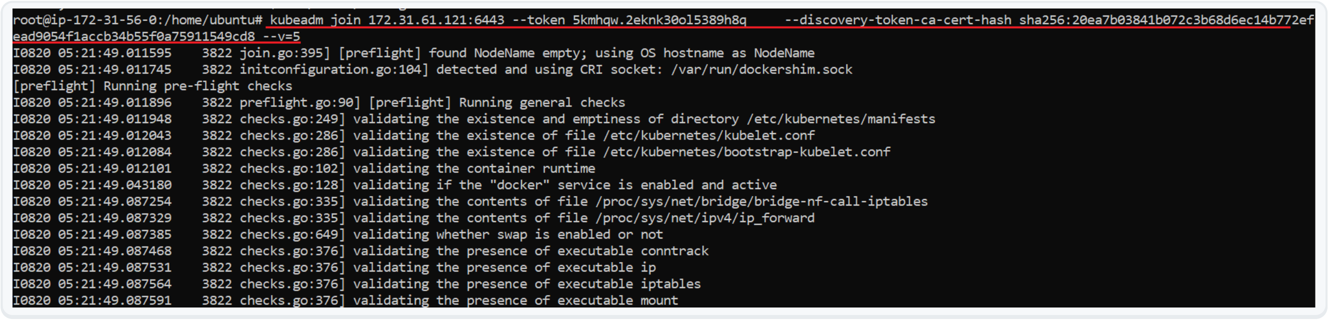
1. Run the following commands on the worker node.

```
sudo su
kubeadm reset pre-flight checks
```



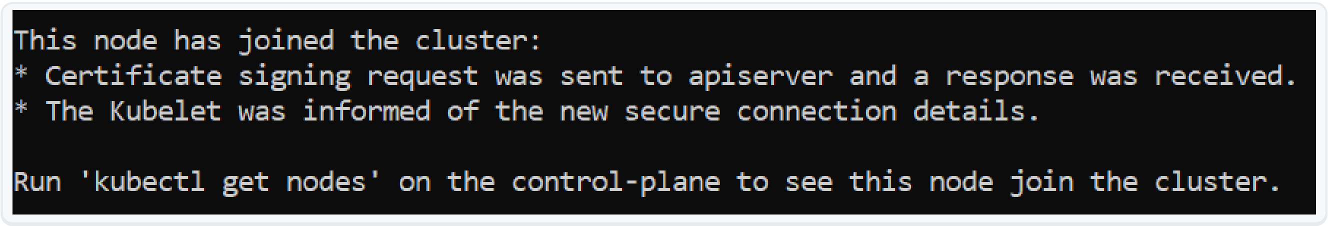
```
root@ip-172-31-56-0:/home/ubuntu# sudo su
reset pre-flight checkskubeadm reset pre-flight checksroot@ip-172-31-56-0:/home/ubuntu# kubeadm reset pre-flight checks
[reset] WARNING: Changes made to this host by 'kubeadm init' or 'kubeadm join' will be reverted.
[reset] Are you sure you want to proceed? [y/N]: y
[preflight] Running pre-flight checks
W0820 05:19:17.290510    3784 removeetcdmember.go:79] [reset] No kubeadm config, using etcd pod spec to get data directory
[reset] No etcd config found. Assuming external etcd
[reset] Please, manually reset etcd to prevent further issues
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
W0820 05:19:17.296207    3784 cleanupnode.go:99] [reset] Failed to evaluate the "/var/lib/kubelet" directory. Skipping its u
uch file or directory
[reset] Deleting contents of config directories: [/etc/kubernetes/manifests /etc/kubernetes/pki]
[reset] Deleting files: [/etc/kubernetes/admin.conf /etc/kubernetes/kubelet.conf /etc/kubernetes/bootstrap-kubelet.conf /etc/
tes/scheduler.conf]
[reset] Deleting contents of stateful directories: [/var/lib/dockershim /var/run/kubernetes /var/lib/cni]
```

2. Paste the join command you got from the master node and append `--v=5` at the end.



```
root@ip-172-31-56-0:/home/ubuntu# kubeadm join 172.31.61.121:6443 --token 5kmhqw.2eknk30ol5389h8q     --discovery-token-ca-cert-hash sha256:20ea7b03841b072c3b68d6ec14b772ef
ead9054f1accb34b55f0a75911549cd8 --v=5
I0820 05:21:49.011595    3822 join.go:395] [preflight] found NodeName empty; using OS hostname as NodeName
I0820 05:21:49.011745    3822 initconfiguration.go:104] detected and using CRI socket: /var/run/dockershim.sock
[preflight] Running pre-flight checks
I0820 05:21:49.011896    3822 preflight.go:90] [preflight] Running general checks
I0820 05:21:49.011948    3822 checks.go:249] validating the existence and emptiness of directory /etc/kubernetes/manifests
I0820 05:21:49.012043    3822 checks.go:286] validating the existence of file /etc/kubernetes/kubelet.conf
I0820 05:21:49.012084    3822 checks.go:286] validating the existence of file /etc/kubernetes/bootstrap-kubelet.conf
I0820 05:21:49.012101    3822 checks.go:102] validating the container runtime
I0820 05:21:49.043180    3822 checks.go:128] validating if the "docker" service is enabled and active
I0820 05:21:49.087254    3822 checks.go:335] validating the contents of file /proc/sys/net/bridge/bridge-nf-call-iptables
I0820 05:21:49.087329    3822 checks.go:335] validating the contents of file /proc/sys/net/ipv4/ip_forward
I0820 05:21:49.087385    3822 checks.go:649] validating whether swap is enabled or not
I0820 05:21:49.087468    3822 checks.go:376] validating the presence of executable conntrack
I0820 05:21:49.087531    3822 checks.go:376] validating the presence of executable ip
I0820 05:21:49.087564    3822 checks.go:376] validating the presence of executable iptables
I0820 05:21:49.087591    3822 checks.go:376] validating the presence of executable mount
```

After succesful join->

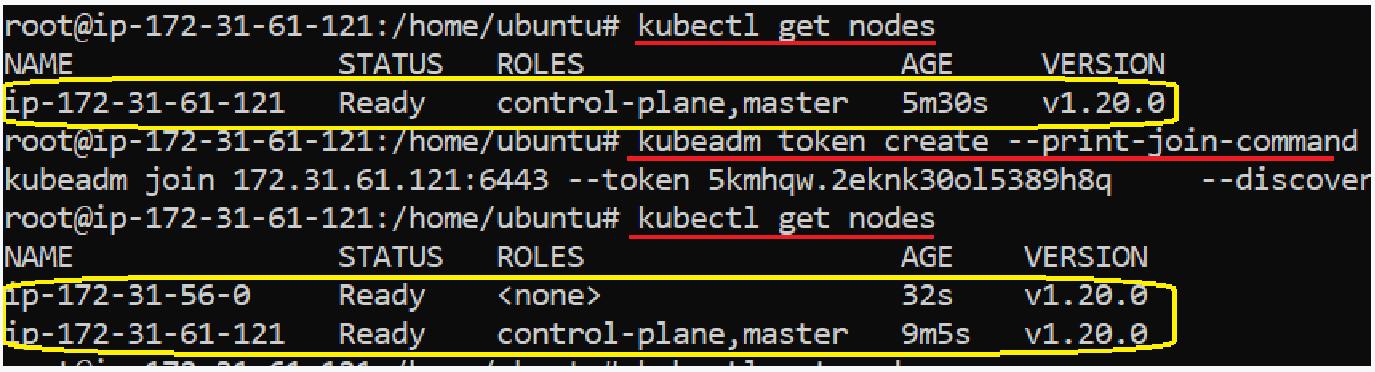


```
This node has joined the cluster:
* Certificate signing request was sent to apiserver and a response was received.
* The Kubelet was informed of the new secure connection details.

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

Verify Cluster Connection

On Master Node:

```
kubectl get nodes
```



```
root@ip-172-31-61-121:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES                  AGE     VERSION
ip-172-31-61-121   Ready    control-plane,master   5m30s   v1.20.0
root@ip-172-31-61-121:/home/ubuntu# kubeadm token create --print-join-command
kubeadm join 172.31.61.121:6443 --token 5kmhqw.2eknk30ol5389h8q     --discover
root@ip-172-31-61-121:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES                  AGE    VERSION
ip-172-31-56-0     Ready    <none>                 32s    v1.20.0
ip-172-31-61-121   Ready    control-plane,master   9m5s   v1.20.0
```

Optional: Labeling Nodes

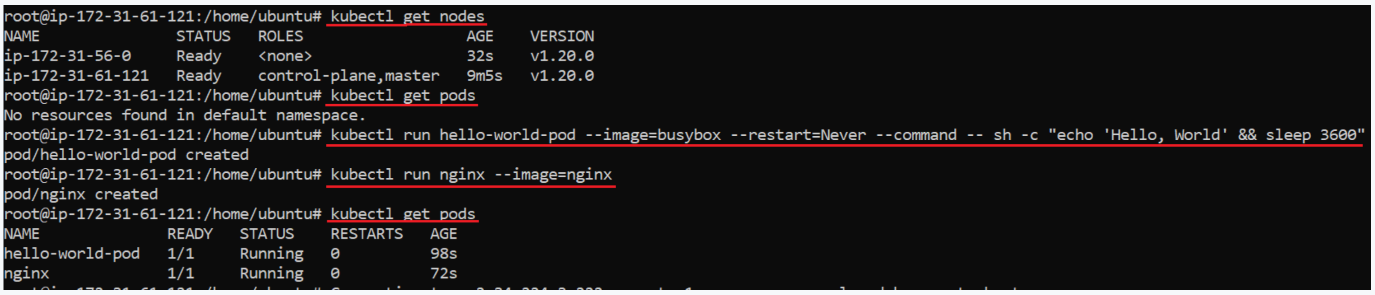
If you want to label worker nodes, you can use the following command:

```
kubectl label node <node-name> node-role.kubernetes.io/worker=worker
```

Optional: Test a demo Pod

If you want to test a demo pod, you can use the following command:

```
kubectl run hello-world-pod --image=busybox --restart=Never --command -- sh -c "echo
```



```
root@ip-172-31-61-121:/home/ubuntu# kubectl get nodes
NAME               STATUS   ROLES                  AGE    VERSION
ip-172-31-56-0     Ready    <none>                 32s    v1.20.0
ip-172-31-61-121   Ready    control-plane,master   9m5s   v1.20.0
root@ip-172-31-61-121:/home/ubuntu# kubectl get pods
No resources found in default namespace.
root@ip-172-31-61-121:/home/ubuntu# kubectl run hello-world-pod --image=busybox --restart=Never --command -- sh -c "echo 'Hello, World' && sleep 3600"
pod/hello-world-pod created
root@ip-172-31-61-121:/home/ubuntu# kubectl run nginx --image=nginx
pod/nginx created
root@ip-172-31-61-121:/home/ubuntu# kubectl get pods
NAME              READY   STATUS    RESTARTS   AGE
hello-world-pod   1/1     Running   0          98s
nginx             1/1     Running   0          72s
```