**Installing kubeadm**

This page shows how to install the kubeadm toolbox. For information on how to create a cluster with kubeadm once you have performed this installation process, see the [Creating a cluster with HYPERLINK "https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/"kubeadm](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/) page.

**Before you begin**

* A compatible Linux host. The Kubernetes project provides generic instructions for Linux distributions based on Debian and Red Hat, and those distributions without a package manager.
* 2 GB or more of RAM per machine (any less will leave little room for your apps).
* 2 CPUs or more.
* Full network connectivity between all machines in the cluster (public or private network is fine).
* Unique hostname, MAC address, and product\_uuid for every node. See [here](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/) for more details.
* Certain ports are open on your machines. See [here](https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/) for more details.
* Swap disabled. You **MUST** disable swap in order for the kubelet to work properly.

**Check required ports**

These [required ports](https://kubernetes.io/docs/reference/networking/ports-and-protocols/) need to be open in order for Kubernetes components to communicate with each other. You can use tools like netcat to check if a port is open. For example:

nc 127.0.0.1 6443

The pod network plugin you use may also require certain ports to be open. Since this differs with each pod network plugin, please see the documentation for the plugins about what port(s) those need.

**Installing a container runtime**

Installing Docker

*Uninstall old versions*

Older versions of Docker went by the names of docker, docker.io, or docker-engine, you might also have installations of containerd or runc. Uninstall any such older versions before attempting to install a new version:

sudo apt-get remove docker docker-engine docker.io containerd runc

apt-get might report that you have none of these packages installed.

*Install using the apt repository*

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

**Set up the repository**

* Update the apt package index and install packages to allow apt to use a repository over HTTPS:

sudo apt-get update

sudo apt-get install \

ca-certificates \

curl \

gnupg

* Add Docker’s official GPG key:

sudo mkdir -m 0755 -p /etc/apt/keyrings

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

* Use the following command to set up the repository:

echo \

"deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg] <https://download.docker.com/linux/ubuntu> \

"$(. /etc/os-release && echo "$VERSION\_CODENAME")" stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

*Install Docker Engine*

* Update the apt package index:

sudo apt-get update

Receiving a GPG error when running apt-get update?

Your default umask may be incorrectly configured, preventing detection of the repository public key file. Try granting read permission for the Docker public key file before updating the package index:

sudo chmod a+r /etc/apt/keyrings/docker.gpg

sudo apt-get update

* Install Docker Engine, containerd, and Docker Compose.

To install the latest version, run:

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y

* Verify that the Docker Engine installation is successful by running the hello-world image:

sudo docker run hello-world

This command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

Installing kubeadm, kubelet and kubectl

* Update the apt package index and install packages needed to use the Kubernetes apt repository:

sudo apt-get update

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

* Download the Google Cloud public signing key:

sudo curl -fsSLo /etc/apt/keyrings/kubernetes-archive-keyring.gpg <https://packages.cloud.google.com/apt/doc/apt-key.gpg>

* Add the Kubernetes apt repository:

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

* Update apt package index, install kubelet, kubeadm and kubectl, and pin their version:

sudo apt-get update

sudo apt-get install -y kubelet kubeadm kubectl

sudo apt-mark hold kubelet kubeadm kubectl

sudo systemctl enable kubelet

sudo systemctl start kubelet

sudo swapoff -a

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

vim /etc/fstab comment the /swap.img

sudo reboot now

###Note: If we are getting Container runtime is not running error

sudo rm /etc/containerd/config.toml

sudo systemctl restart containerd

sudo reboot now

Install all the required tools on both master and workers nodes, follow the above mentioned procedure.

To initialize the control-plane node run:

kubeadm init <args>

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

kubeadm init first runs a series of prechecks to ensure that the machine is ready to run Kubernetes. These prechecks expose warnings and exit on errors. kubeadm init then downloads and installs the cluster control plane components. This may take several minutes. After it finishes you should see:

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:

/docs/concepts/cluster-administration/addons/

You can now join any number of machines by running the following on each node

as root:

kubeadm join <control-plane-host>:<control-plane-port> --token <token> --discovery-token-ca-cert-hash sha256:<hash>

execute this on the master node

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

sudo chmod 777 /etc/kubernetes/admin.conf

kubectl get nodes

kubectl apply -f <https://github.com/flannel-io/flannel/releases/latest/download/kube-flannel.yml>

#once the kube-flannel pods are up then the nodes will be ready

kubectl get nodes

Now copy the join token from the master node and execute in the worker node

Example:

kubeadm join 172.31.43.2:6443 --token wl3td9.0ragjxhjwz1deo5t \

--discovery-token-ca-cert-hash sha256:c9eaf691a2700c7880b33b9ca8f0b4992a4458205906993947a146f750919492

kubectl get nodes

To regenrate the tokens

kubeadm token create --print-join-command

Now the Kubernetes cluster is ready

When you get Below Error on Worker node

E1222 10:48:23.353706 2262 memcache.go:265] couldn't get current server API group list: Get "<http://localhost:8080/api?timeout=32s>": dial tcp 127.0.0.1:8080: connect: connection refused

The connection to the server localhost:8080 was refused - did you specify the right host or port?

That means Cluster information (Configuartion) is missing on Worker node, So copy the .kube/config file from master to worker

curl <https://raw.githubusercontent.com/projectcalico/calico/v3.25.1/manifests/calico.yaml> -O

kubectl apply -f calico.yaml

sudo kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-advertise-address=54.209.29.212