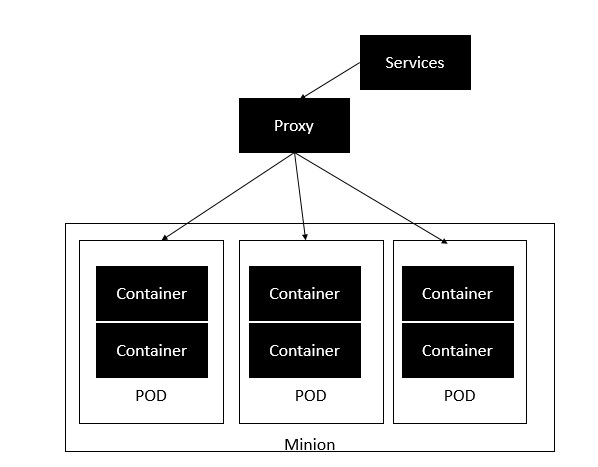
# Docker - Kubernetes Architecture

Kubernetes is an orchestration framework for Docker containers which helps expose containers as services to the outside world. For example, you can have two services − One service would contain **nginx** and **mongoDB**, and another service would contain **nginx** and **redis**. Each service can have an IP or service point which can be connected by other applications. Kubernetes is then used to manage these services.

The following diagram shows in a simplistic format how Kubernetes works from an architecture point of view.



The **minion** is the node on which all the services run. You can have many minions running at one point in time. Each minion will host one or more POD. Each **POD** is like hosting a service. Each POD then contains the Docker containers. Each POD can host a different set of Docker containers. The proxy is then used to control the exposing of these services to the outside world.

Kubernetes has several components in its architecture. The role of each component is explained below &mius;

**etcd** − This component is a highly available **key-value** store that is used for storing **shared configuration** and **service discovery**. Here the various applications will be able to connect to the services via the **discovery service**.

**Flannel** − This is a backend network which is required for the containers.

**kube-apiserver** − This is an API which can be used to orchestrate the Docker containers.

**kube-controller-manager** − This is used to control the **Kubernetes services**.

**kube-scheduler** − This is used to schedule the containers on hosts.

**Kubelet** − This is used to control the launching of containers via **manifest files**.

**kube-proxy** − This is used to provide network proxy services to the outside world.

# Docker - Working of Kubernetes

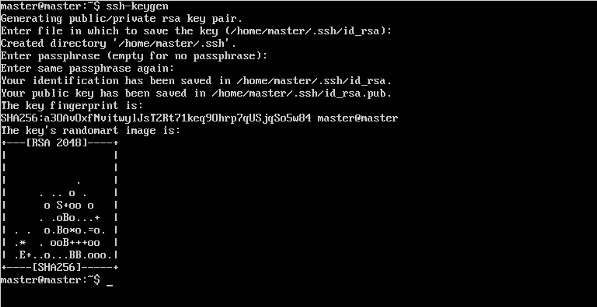
In this chapter, we will see how to install **Kubenetes** via **kubeadm**. This is a tool which helps in the installation of Kubernetes. Let’s go step by step and learn how to install Kubernetes.

**Step 1** − Ensure that the **Ubuntu server version** you are working on is **16.04**.

**Step 2** − Ensure that you generate a **ssh** key which can be used for **ssh** login. You can do this using the following command.

ssh-keygen

This will generate a key in your **home folder** as shown below.



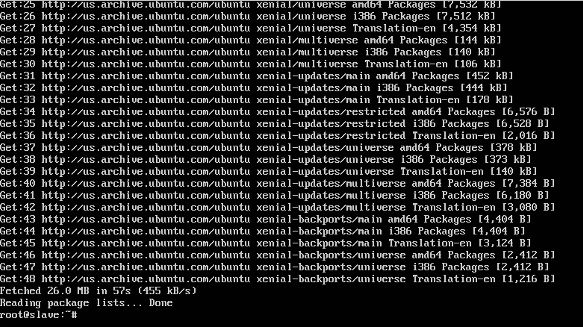
**Step 3** − Next, depending on the version of Ubuntu you have, you will need to add the relevant site to the **docker.list** for the **apt package manager**, so that it will be able to detect the **Kubernetes packages** from the **kubernetes** site and download them accordingly.

We can do it using the following commands.

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | apt-key add -

echo "deb http://apt.kubernetes.io/ kubernetes-xenial main” | sudo tee /etc/apt/sources.list.d/docker.list

**Step 4** − We then issue an apt-get update to ensure all packages are downloaded on the Ubuntu server.

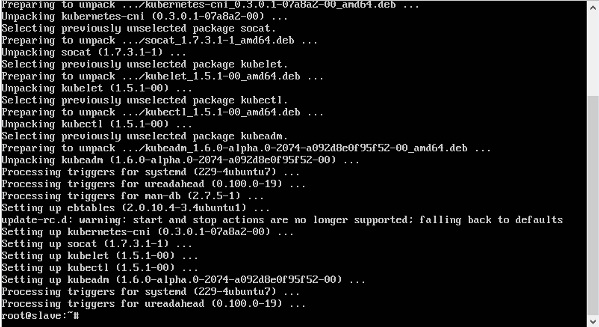


**Step 5** − Install the Docker package as detailed in the earlier chapters.

**Step 6** − Now it’s time to install **kubernetes** by installing the following packages −

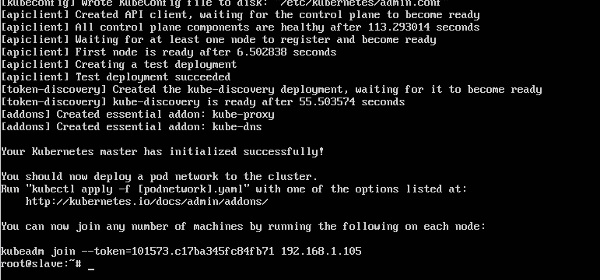
apt-get install –y kubelet kubeadm kubectl kubernetes-cni

IMG_258



**Step 7** − Once all **kubernetes** packages are downloaded, it’s time to start the kubernetes controller using the following command −

kubeadm init



Once done, you will get a successful message that the master is up and running and nodes can now join the cluster.