Docker Tutorial

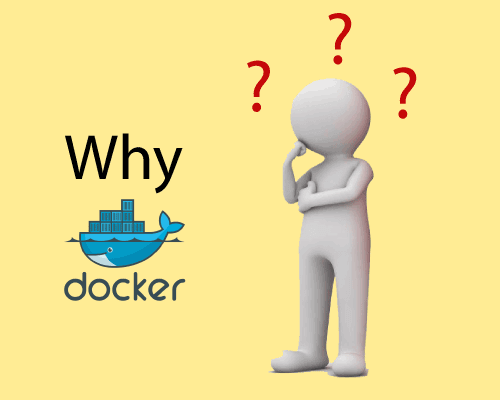
## What is Docker?

Docker is an **open-source centralized platform designed** to create, deploy, and run applications. Docker uses **container** on the host's operating system to run applications. It allows applications to use the same **Linux kernel** as a system on the host computer, rather than creating a whole virtual operating system. Containers ensure that our application works in any environment like development, test, or production.

### Containers Vs. Virtual Machine

|  |  |
| --- | --- |
| **Containers** | **Virtual Machine** |
| Integration in a container is faster and cheap. | Integration in virtual is slow and costly. |
| No wastage of memory. | Wastage of memory. |
| It uses the same kernel, but different distribution. | It uses multiple independent  operating systems. |

## Why Docker?



Docker is designed to benefit both the Developer and System Administrator. There are the following reasons to use Docker -

* Docker allows us to easily install and run software without worrying about setup or dependencies.
* Developers use Docker to eliminate machine problems, i.e. "**but code is worked on my laptop**." when working on code together with co-workers.
* Operators use Docker to run and manage apps in isolated containers for better compute density.
* Enterprises use Docker to securely built agile software delivery pipelines to ship new application features faster and more securely.
* Since docker is not only used for the deployment, but it is also a great platform for development, that's why we can efficiently increase our customer's satisfaction.

## Advantages of Docker

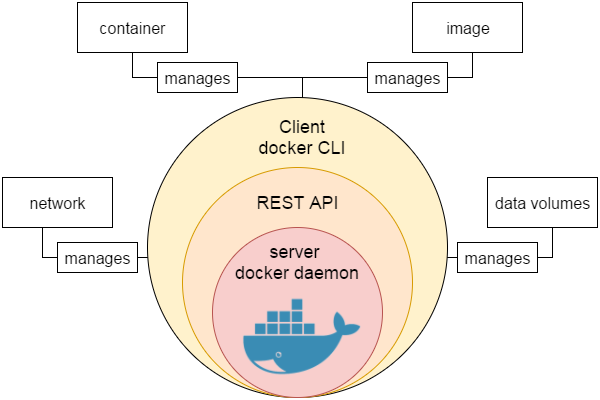
There are the following advantages of Docker -

* It runs the container in seconds instead of minutes.
* It uses less memory.

## Docker Engine

It is a client server application that contains the following major components.

* A server which is a type of long-running program called a daemon process.
* The REST API is used to specify interfaces that programs can use to talk to the daemon and instruct it what to do.
* A command line interface client.



Docker Features

Although Docker provides lots of features, we are listing some major features which are given below.

* Easy and Faster Configuration
* Increase productivity
* Application Isolation
* Swarm
* Routing Mesh
* Services
* Security Management

# Docker Architecture

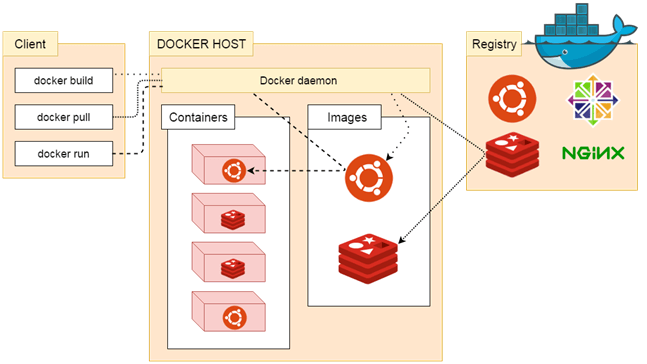
Before learning the Docker architecture, first, you should know about the Docker Daemon.

### What is Docker daemon?

Docker daemon runs on the host operating system. It is responsible for running containers to manage docker services. Docker daemon communicates with other daemons. It offers various Docker objects such as images, containers, networking, and storage. s

### Docker architecture

Docker follows Client-Server architecture, which includes the three main components that are **Docker Client**, **Docker Host**, and **Docker Registry**.



### 1. Docker Client

Docker client uses **commands** and **REST APIs** to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

#### Note: Docker Client has an ability to communicate with more than one docker daemon.

Docker Client uses Command Line Interface (CLI) to run the following commands -

docker build

docker pull

docker run

### 2. Docker Host

Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

### 3. Docker Registry

Docker Registry manages and stores the Docker images.

There are two types of registries in the Docker -

**Pubic Registry -** Public Registry is also called as **Docker hub**.

**Private Registry -** It is used to share images within the enterprise.

## Docker Objects

There are the following Docker Objects -

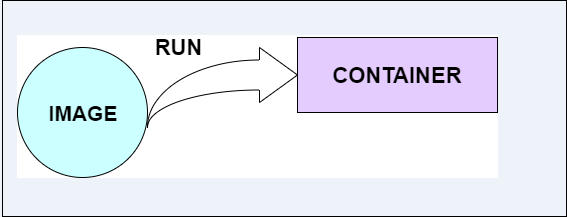
### Docker Images

Docker images are the **read-only binary templates** used to create Docker Containers. It uses a private container registry to share container images within the enterprise and also uses public container registry to share container images within the whole world. Metadata is also used by docket images to describe the container's abilities.

### Docker Containers

Containers are the structural units of Docker, which is used to hold the entire package that is needed to run the application. The advantage of containers is that it requires very less resources.

In other words, we can say that the image is a template, and the container is a copy of that template.



### Docker Networking

Using Docker Networking, an isolated package can be communicated. Docker contains the following network drivers -

* **Bridge -** Bridge is a default network driver for the container. It is used when multiple docker communicates with the same docker host.
* **Host -** It is used when we don't need for network isolation between the container and the host.
* **None -** It disables all the networking.
* **Overlay -** Overlay offers Swarm services to communicate with each other. It enables containers to run on the different docker host.
* **Macvlan -** Macvlan is used when we want to assign MAC addresses to the containers.

### Docker Storage

Docker Storage is used to store data on the container. Docker offers the following options for the Storage -

* **Data Volume -** Data Volume provides the ability to create persistence storage. It also allows us to name volumes, list volumes, and containers associates with the volumes.
* **Directory Mounts -** It is one of the best options for docker storage. It mounts a host's directory into a container.
* **Storage Plugins -** It provides an ability to connect to external storage platforms.

How to install docker on Windows

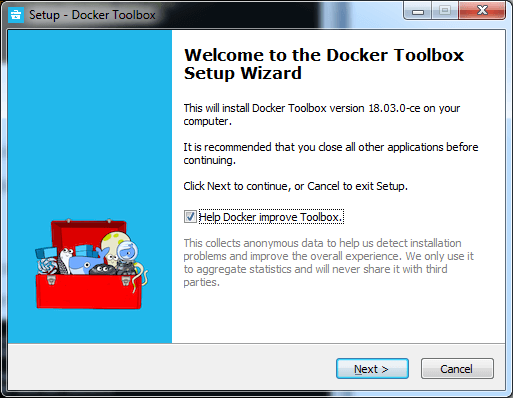
We can install docker on any operating system like **Windows, Linux,** or **Mac**. Here, we are going to install docker-engine on **Windows**. The main advantage of using Docker on Windows is that it provides an ability to run natively on Windows without any kind of virtualization. To install docker on windows, we need to download and install the **Docker Toolbox**.

Follow the below steps to install docker on windows -

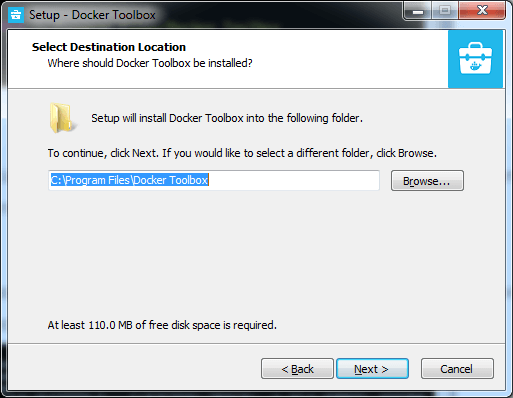
**Step 1:** Click on the below link to download DockerToolbox.exe. <https://download.docker.com/win/stable/DockerToolbox.exe>

**Step 2:** Once the **DockerToolbox.exe** file is downloaded, **double click** on that file. The following window appears on the screen, in which click on the **Next**.

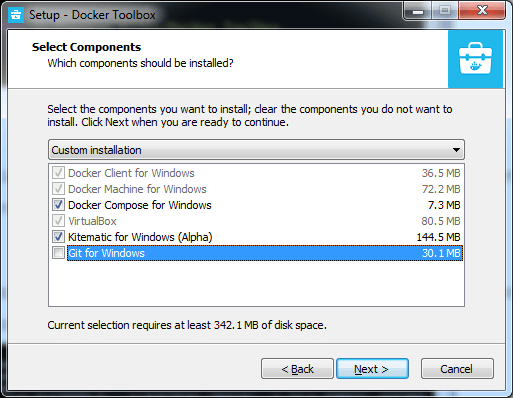
Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)



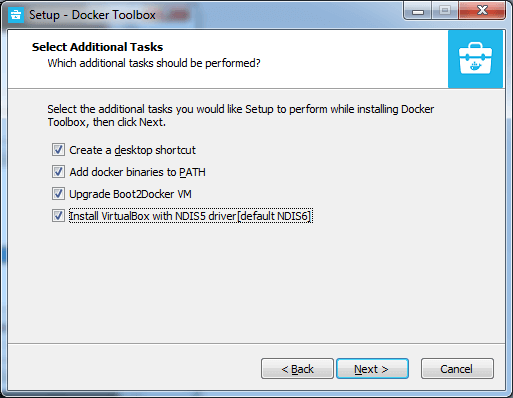
**Step 3: Browse the location** where you want to install the Docker Toolbox and click on the Next.



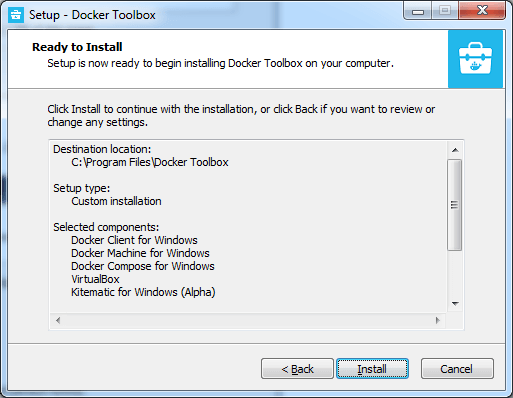
**Step 4: Select the components** according to your requirement and click on the **Next**.



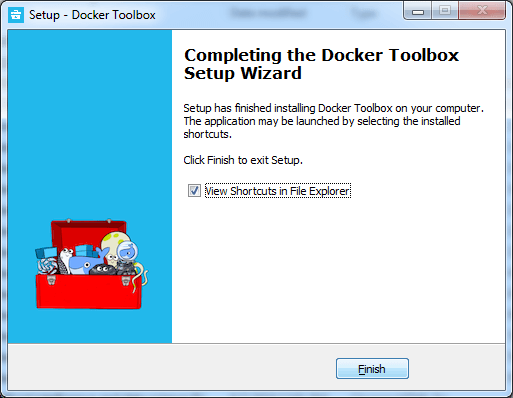
**Step 5: Select Additional Tasks** and click on the **Next**.



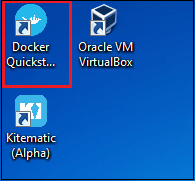
**Step 6:** The Docker Toolbox is ready to install. Click on **Install**.



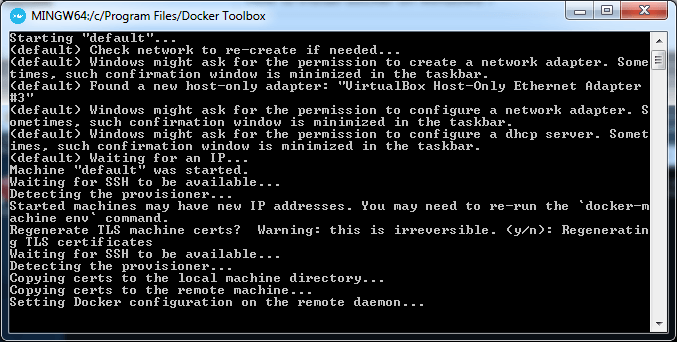
**Step 7:** Once the installation is completed, the following Wizard appears on the screen, in which click on the **Finish**.



**Step 8:** After the successful installation, three icons will appear on the screen that are: **Docker Quickstart Terminal, Kitematic (Alpha),** and **OracleVM VirtualBox**. **Double click** on the Docker Quickstart Terminal.



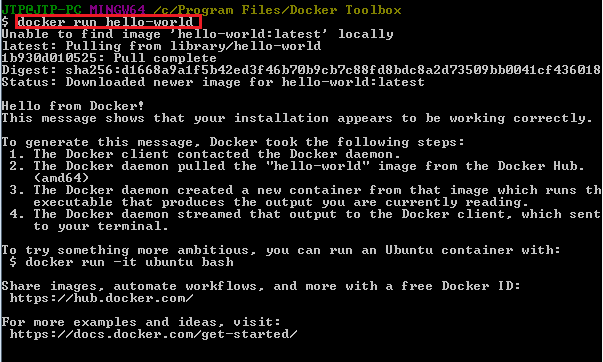
**Step 9:** A Docker Quickstart Terminal window appears on the screen.



To verify that the docker is successfully installed, type the below command and press enter key.

1. docker run hello-world

The following output will be visible on the screen, otherwise not.



You can check the Docker version using the following command.

1. docker -version

install docker on Windows

# Docker Container and Image

Docker container is a running instance of an image. You can use Command Line Interface (CLI) commands to run, start, stop, move, or delete a container. You can also provide configuration for the network and environment variables. Docker container is an isolated and secure application platform, but it can share and access to resources running in a different host or container.

An image is a read-only template with instructions for creating a Docker container. A docker image is described in text file called a **Dockerfile**, which has a simple, well-defined syntax. An image does not have states and never changes. Docker Engine provides the core Docker technology that enables images and containers.

You can understand container and image with the help of the following command.

1. $ docker run hello-world

The above command **docker run hello-world** has three parts.

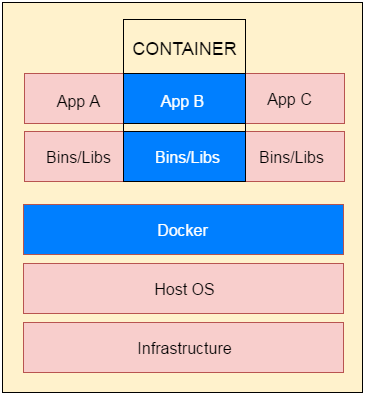
Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

1) **docker:** It is docker engine and used to run docker program. It tells to the operating system that you are running docker program.

2) **run:** This subcommand is used to create and run a docker container.

3) **hello-world:** It is a name of an image. You need to specify the name of an image which is to load into the container.

### Docker Container



**Fig**: docker-container

Docker Java Application Example

As, we have mentioned earlier that docker can execute any application.

Here, we are creating a Java application and running by using the docker. This example includes the following steps.

1. **Create a directory**

Directory is required to organize files. Create a director by using the following command.

* 1. $ mkdir  java-docker-app

See, screen shot for the above command.

Docker Java application 1

1. **Create a Java File**

Now create a Java file. Save this file as **Hello.java** file.

**// Hello.java**

* 1. **class** Hello{
  2. **public** **static** **void** main(String[] args){
  3. System.out.println("This is java app \n by using Docker");
  4. }
  5. }

Save it inside the directory **java-docker-app** as Hello.java.

1. **Create a Dockerfile**

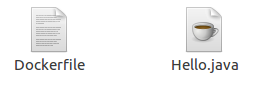
After creating a Java file, we need to create a Dockerfile which contains instructions for the Docker. Dockerfile does not contain any file extension. So, save it simple with **Dockerfile** name.

**// Dockerfile**

* 1. FROM java:8
  2. COPY . /var/www/java
  3. WORKDIR /var/www/java
  4. RUN javac Hello.java
  5. CMD ["java", "Hello"]

Write all instructions in uppercase because it is convention. Put this file inside **java-docker-app** directory. Now we have Dockerfile parallel to Hello.java inside the **java-docker-app** directory.

See, your folder inside must look like the below.

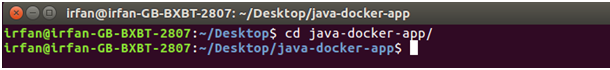


1. **Build Docker Image**

After creating Dockerfile, we are changing working directory.

* 1. $ cd   java-docker-app

See, the screen shot.



Now, create an image by following the below command. we must login as root in order to create an image. In this example, we have switched to as a root user. In the following command, **java-app**is name of the image. We can have any name for our docker image.

* 1. $ docker build -t java-app .

See, the screen shot of the above command.



After successfully building the image. Now, we can run our docker image.

1. **Run Docker Image**

After creating image successfully. Now we can run docker by using run command. The following command is used to run java-app.

* 1. $ docker run java-app

See, the screen shot of the above command.



Here, we can see that after running the java-app it produced an output.

Now, we have run docker image successfully on your system. Apart from all these you can also use other commands as well.

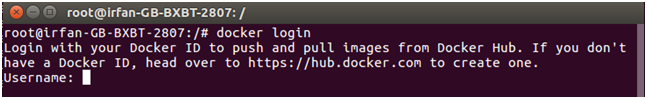
Docker Push Repository

We can push our Docker image to global repository. It is a public repository provided by Docker officially. It allows us to put our docker image on the server. It is helpful when we want to access our docker image from global. Follow the following steps to push custom image on the Docker hub.

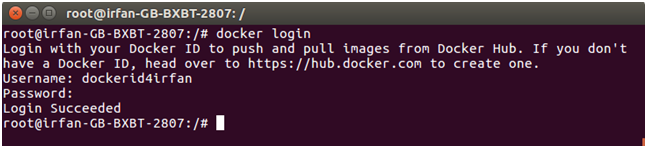
1. **login to hub.docker.com**

We need to login to our account of Docker hub. If you don't have, **create it first.**

* 1. $ docker login



It will ask for username. Enter the dockerid here and press enter.



After providing username, it asks for password. Enter your account password here and it will show you your login status as succeeded.

1. **Tag Docker Image**

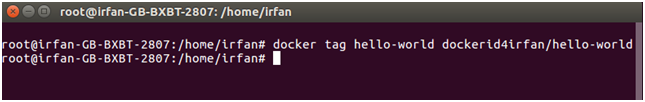
After login, we need to tag our docker image that we want to push. Following command is used to tag the docker image.

* 1. $ docker tag image-name username/image-name

**username** refers to our dockerid or the username which is used to login.

**image-name** is the name of our docker image present on our system.

See, screen shot of the above command.



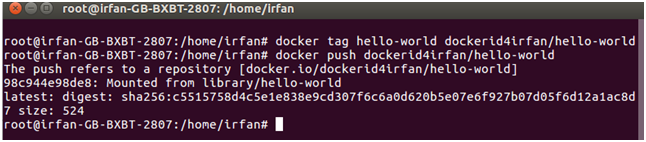
In the above command, we have tagged docker image ***hello-world***. Now, we need to push it to the repository. Let?s see it in the below command.

1. **Push Docker Image**

The following command is used to push docker image to docker hub repository.

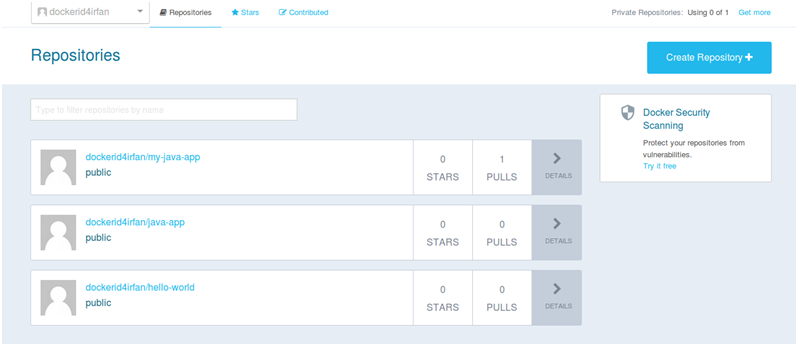
* 1. $ docker push  username/image-name

See, screen shot of the above command.



In the above screen shot, we can see that docker image has been pushed successfully.

Now, login into our account at hub.docker.com and check our dashboard. It will have a new docker image named ***dockerid4irfan/hello-world.***



Look at the screen shot, it has the newest one docker image which is just pushed. On the top, the first one is the newest image.

# Docker Useful Commands

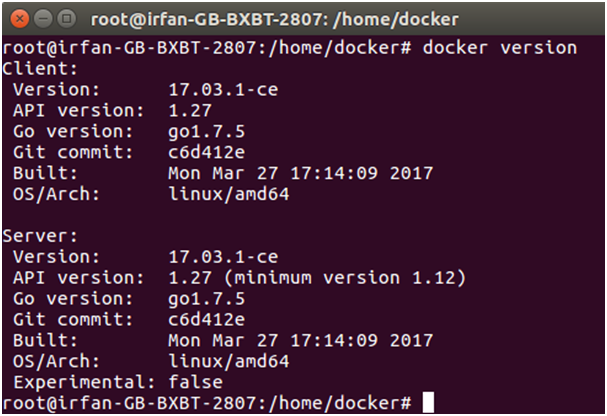
Docker is natively Linux based software so that it provides commands to interact and work in the client-server environment.

Here, we have listed some important and useful Docker commands.

### Check Docker version

1. $ docker version

It shows docker version for both client and server. As given in the following image.



### Build Docker Image from a Dockerfile

1. $ docker build -t image-name docker-file-location

**-t**: it is used to tag Docker image with the provided name.

Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

### Run Docker Image

1. $ docker run -d image-name

**-d** : It is used to create a daemon process.

### Check available Docker images

1. $ docker images

### Check for latest running container

1. $ docker ps -l

**-l** : it is used to show latest available container.

### Check all running containers

1. $ docker ps -a

**-a** : It is used to show all available containers.

### Stop running container

1. $ docker stop container\_id

**container\_id** : It is an Id assigned by the Docker to the container.

### Delete an image

1. $ docker rmi image-name

### Delete all images

1. $ docker rmi $(docker images -q)

### Delete all images forcefully

1. $ docker rmi -r $(docker images -q)

**-r** : It is used to delete image forcefully.

### Delete all containers

1. $ docker rm $(docker ps -a -q)

### Enter into Docker container

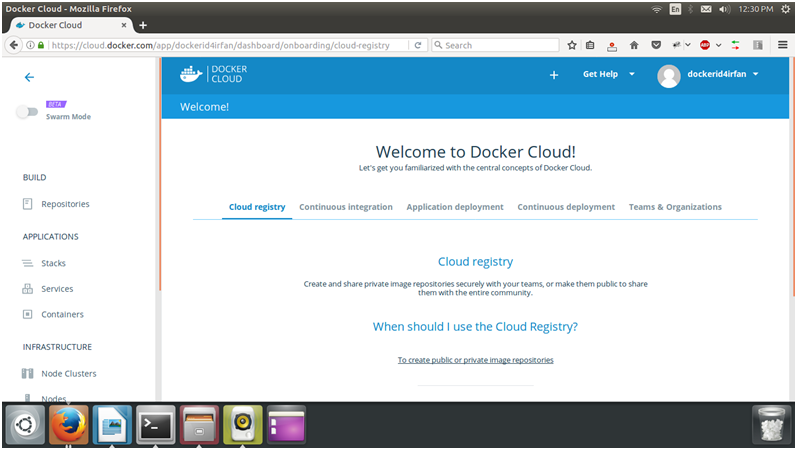
1. $ docker exec -it container-id bash

# Docker Cloud

Docker provides us the facility to store and fetch docker images on the cloud registry. We can store dockerized images either privately or publicly. It is a full GUI interface that allows us to manage builds, images, swarms, nodes and apps.

We need to have Docker ID to access and control images. If we don't have, create it first.

Here, in the following screenshot, we have logged in to Docker cloud. It shows a welcome page.

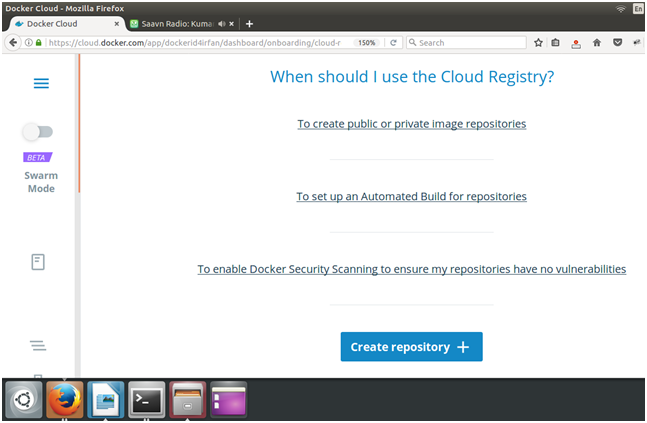


In the left panel, we can see that it provides lots of functionalities that we use on the cloud. Apart from all these, let's create a repository first.

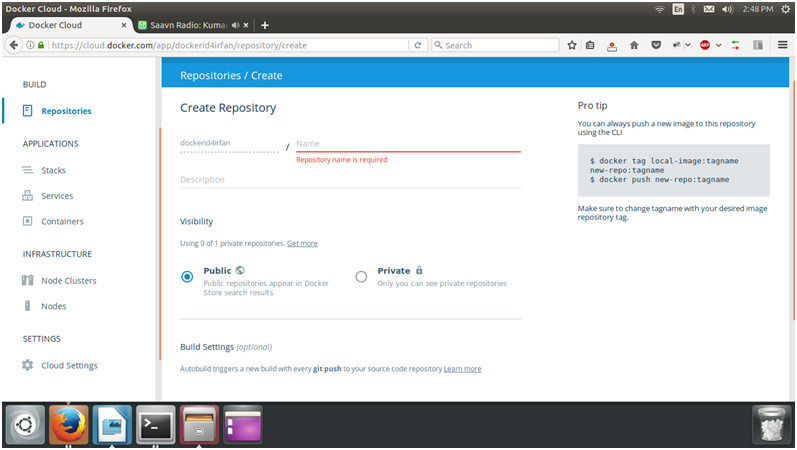
Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

## Creating Repository

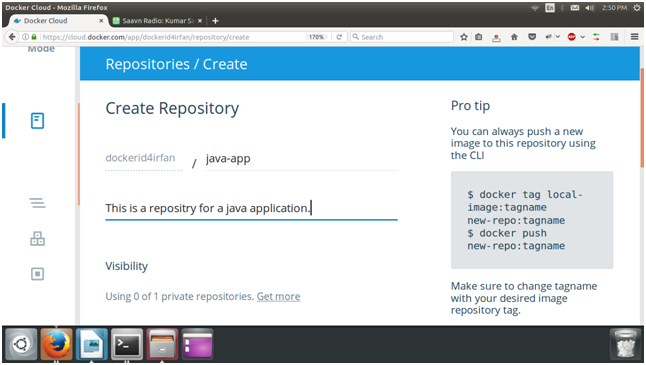
To create Docker cloud repository, click on the create repository +button available on the welcome page at the bottom.



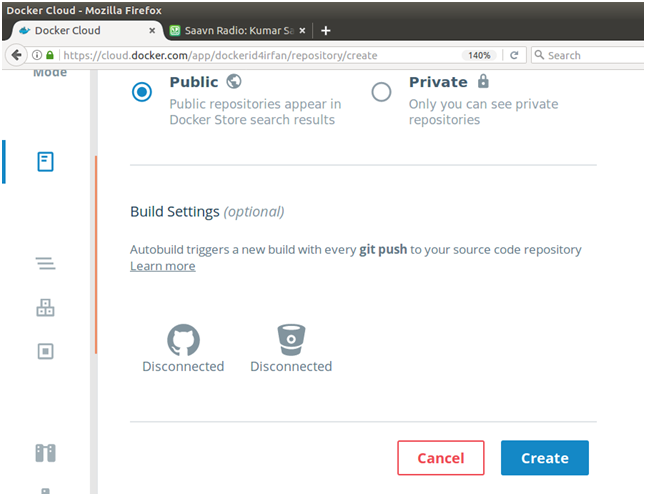
After clicking, it displays a form to enter the name of the repository. The page looks looks like the following.



It asks for the repository name to create a new one. The following screen-shot show the description.



After filling the details, we should make this repository public. Now, just click on the create button at the bottom. It will create repository for us.



So, we can see that it provides the other tools also to manage and control Docker cloud.

# Docker vs. Kubernetes

Today, both Docker and Kubernetes are leading container orchestration tools in the DevOps lifecycle. Docker uses a containerization platform for configuring, building, and distributing containers, while Kubernetes is an Ecosystem for managing a cluster of Docker containers.

Since Docker and Kubernetes are much similar to each other, so before knowing the difference between Docker and Kubernetes, first, you should know about what is Docker? And what is Kubernetes?

### What is Docker?

Docker provides a **containerization platform** which supports various operating systems such as Linux, Windows, and Mac. It allows us to easily build applications, package them with all required dependencies, and ship it to run on other machines. The advantage of using Docker is that it provides benefits for both developers as well as a system administrator. For develops it focuses on writhing the code without worrying about the system. For a system administrator, it provides flexibility to reduces the number of systems for testing the applications.

Docker includes various features such as easy and faster configuration, manages security, use Swarm, routing mesh, application isolation, and increase productivity.

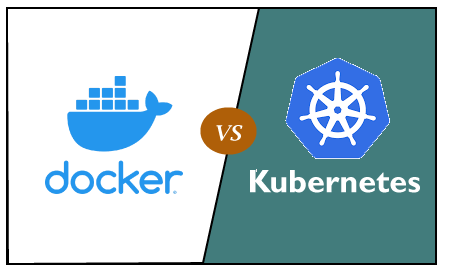
Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

### What is Kubernetes?

Kubernetes (also known as k8s) is an **open**-**source platform** developed by **Google**. It offers powerful, useful, and scalable tools for managing, deploying complicated containerized applications. The advantage of using Kubernetes is that it provides the best solution for scaling up the containers.

Kubernetes includes various features such as runs everywhere, automated rollouts and rollback, storage orchestration, Batch execution, secret and configuration management, horizontal scaling, and offers additional services.

### Docker vs. Kubernetes



The below table shows the difference between Docker and Kubernetes -

|  |  |
| --- | --- |
| **Docker** | **Kubernetes** |
| Docker is developed by **Docker Inc**. | Kubernetes is developed by **Google**. |
| It was first released in **2013**. | It was first released in **2014**. |
| It is a container based technology used to create isolated environment for applications. | It is an infrastructure for managing multiple containers. |
| It allows us to use **third**-**party tools** like ELK for logging and monitoring. | It allows us to use **in-built tools** for logging and monitoring. |
| Its public cloud service provider is **only Azure**. | Its public cloud service providers are **Google**, **Azure**, and **AWS**. |
| It is **less customizable**. | It is **highly customizable**. |
| Its container limit is **95000**. | Its container limit is **300000**. |
| It is **easy** to install. | It is **complex** to install. |
| It **cannot** do **auto-scaling.** | It can **do auto-scaling**. |
| It **does not** provide any **dashboard.** | It provides a **Web UI dashboard**. |

# Kubernetes Tutorial



## What is Kubernetes?

**Kubernetes** is also known as **'k8s'.** This word comes from the Greek language, which means a **pilot** or **helmsman**.  
**Kubernetes** is an extensible, portable, and open-source platform designed by **Google** in **2014**. It is mainly used to automate the deployment, scaling, and operations of the container-based applications across the cluster of nodes. It is also designed for managing the services of containerized apps using different methods which provide the scalability, predictability, and high availability.

It is actually an enhanced version of '**Borg**' for managing the long-running processes and batch jobs. Nowadays, many cloud services offer a Kubernetes-based infrastructure on which it can be deployed as the platform-providing service. This technique or concept works with many container tools, like **docker,** and follows the client-server architecture.

### Key Objects of Kubernetes

Following are the key objects which exist in the Kubernetes:

**Pod**

Play Videox[](https://campaign.adpushup.com/get-started/?utm_source=banner&utm_campaign=growth_hack)

It is the smallest and simplest basic unit of the Kubernetes application. This object indicates the processes which are running in the cluster.

**Node**

A **node** is nothing but a single host, which is used to run the virtual or physical machines. A node in the Kubernetes cluster is also known as a minion.

**Service**

A **service** in a Kubernetes is a logical set of pods, which works together. With the help of services, users can easily manage load balancing configurations.

**ReplicaSet**

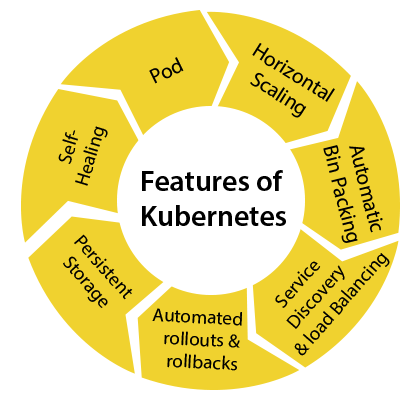
A **ReplicaSet** in the Kubernetes is used to identify the particular number of pod replicas are running at a given time. It replaces the replication controller because it is more powerful and allows a user to use the "set-based" label selector.

**Namespace**

**Kubernetes** supports various virtual clusters, which are known as namespaces. It is a way of dividing the cluster resources between two or more users.

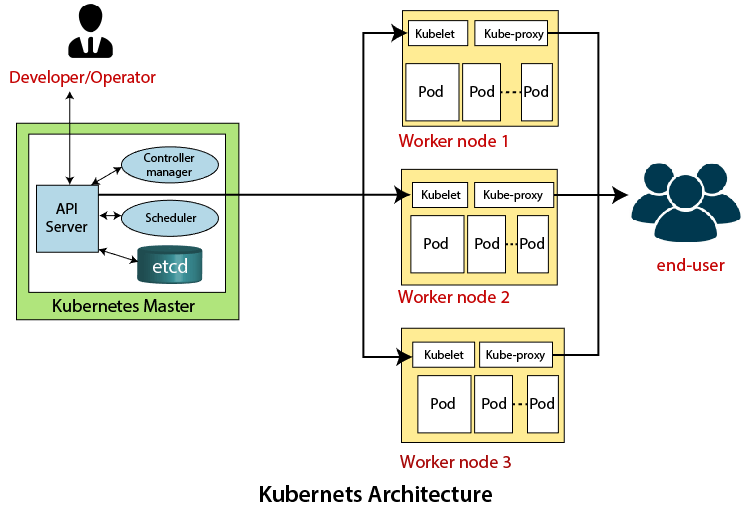
## Features of Kubernetes

Following are the essential features of Kubernetes:



1. **Pod:** It is a deployment unit in Kubernetes with a single Internet protocol address.
2. **Horizontal Scaling:** It is an important feature in the Kubernetes. This feature uses a **HorizontalPodAutoscalar** to automatically increase or decrease the number of pods in a deployment, replication controller, replica set, or stateful set on the basis of observed CPU utilization.
3. **Automatic Bin Packing:** Kubernetes helps the user to declare the maximum and minimum resources of computers for their containers.
4. **Service Discovery and load balancing:** Kubernetes assigns the IP addresses and a Name of DNS for a set of containers, and also balances the load across them.
5. **Automated rollouts and rollbacks:** Using the rollouts, Kubernetes distributes the changes and updates to an application or its configuration. If any problem occurs in the system, then this technique rollbacks those changes for you immediately.
6. **Persistent Storage:** Kubernetes provides an essential feature called '**persistent storage'** for storing the data, which cannot be lost after the pod is killed or rescheduled. Kubernetes supports various storage systems for storing the data, such as **Google Compute Engine's Persistent Disks (GCE PD) or Amazon Elastic Block Storage (EBS).** It also provides the distributed file systems: **NFS or GFS**.
7. **Self-Healing:** This feature plays an important role in the concept of Kubernetes. Those containers which are failed during the execution process, Kubernetes restarts them automatically. And, those containers which do not reply to the user-defined health check, it stops them from working automatically.

## Kubernetes Architecture



The architecture of Kubernetes actually follows the client-server architecture. It consists of the following two main components:

1. Master Node (Control Plane)
2. Slave/worker node

### Master Node or Kubernetes Control Plane

The master node in a Kubernetes architecture is used to manage the states of a cluster. It is actually an entry point for all types of administrative tasks. In the Kubernetes cluster, more than one master node is present for checking the fault tolerance.

Following are the four different components which exist in the Master node or Kubernetes Control plane:

1. API Server
2. Scheduler
3. Controller Manager
4. ETCD

**API Server**

The Kubernetes API server receives the REST commands which are sent by the user. After receiving, it validates the REST requests, process, and then executes them. After the execution of REST commands, the resulting state of a cluster is saved in '**etcd**' as a distributed key-value store.

**Scheduler**

The scheduler in a master node schedules the tasks to the worker nodes. And, for every worker node, it is used to store the resource usage information.  
In other words, it is a process that is responsible for assigning pods to the available worker nodes.

**Controller Manager**

The Controller manager is also known as a controller. It is a daemon that executes in the non-terminating control loops. The controllers in a master node perform a task and manage the state of the cluster. In the Kubernetes, the controller manager executes the various types of controllers for handling the nodes, endpoints, etc.

**ETCD**

It is an open-source, simple, distributed key-value storage which is used to store the cluster data. It is a part of a master node which is written in a GO programming language.

Now, we have learned about the functioning and components of a master node; let's see what is the function of a slave/worker node and what are its components.

### Worker/Slave node

The Worker node in a Kubernetes is also known as minions. A worker node is a physical machine that executes the applications using pods. It contains all the essential services which allow a user to assign the resources to the scheduled containers.

Following are the different components which are presents in the Worker or slave node:

**Kubelet**

This component is an agent service that executes on each worker node in a cluster. It ensures that the pods and their containers are running smoothly. Every **kubelet** in each worker node communicates with the master node. It also starts, stops, and maintains the containers which are organized into pods directly by the master node.

**Kube-proxy**

It is a proxy service of Kubernetes, which is executed simply on each worker node in the cluster. The main aim of this component is request forwarding. Each node interacts with the Kubernetes services through **Kube-proxy**.

**Pods**

A **pod** is a combination of one or more containers which logically execute together on nodes. One worker node can easily execute multiple pods.

## Installation of Kubernetes on Linux

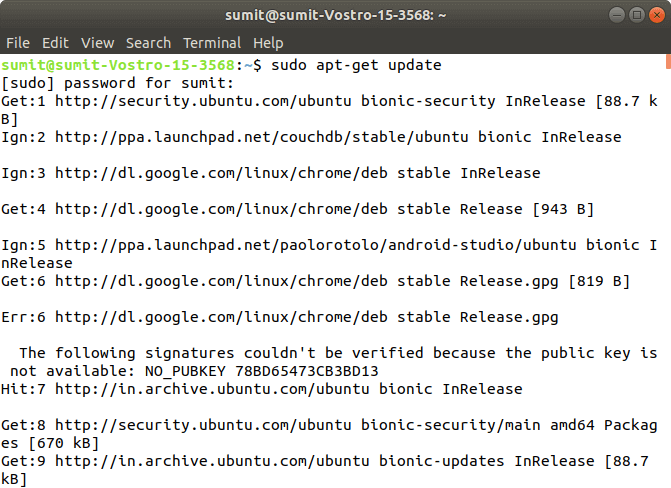
The installation of Kubernetes on Linux is a straight forward process. Follow the below steps to install the Kubernetes. In the installation of Kubernetes, each step is mandatory.

**Step 1:** In this step, we have to update the necessary dependencies of a system using two commands.

The first command is used to get all the updates. Execute the following command in the terminal; it will ask to enter the system's password.

1. sudo apt-get update

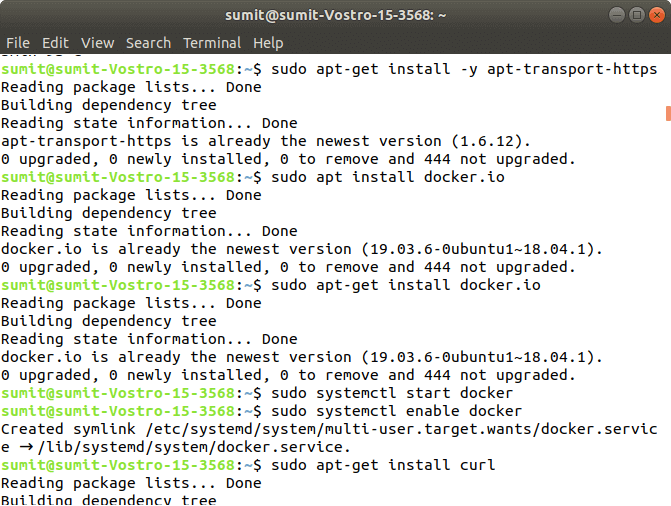
**Output:**



When the first command is successfully executed, type the following second command, which is used to make the repositories.

1. sudo apt-get install -y apt-transport-https

**Output:**

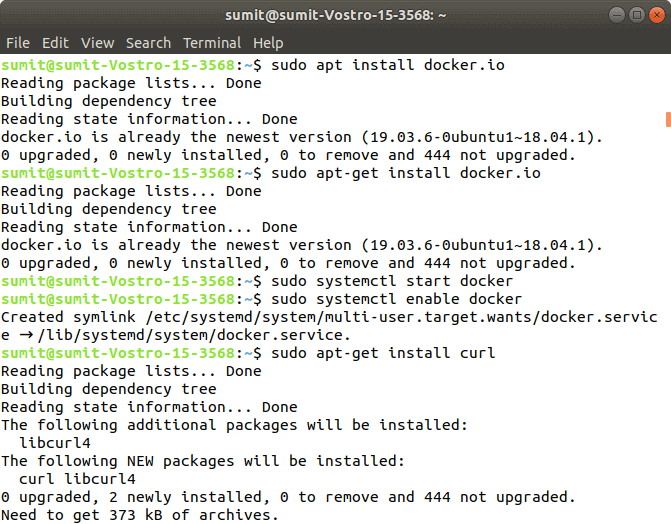


**Step 2:** After the above steps are successfully executed, we have to install the dependencies of docker in this step.

Type the following command to install the docker. In the installation process, we have to choose Y for confirmation of the installation.

1. sudo apt install docker.io

**Output:**



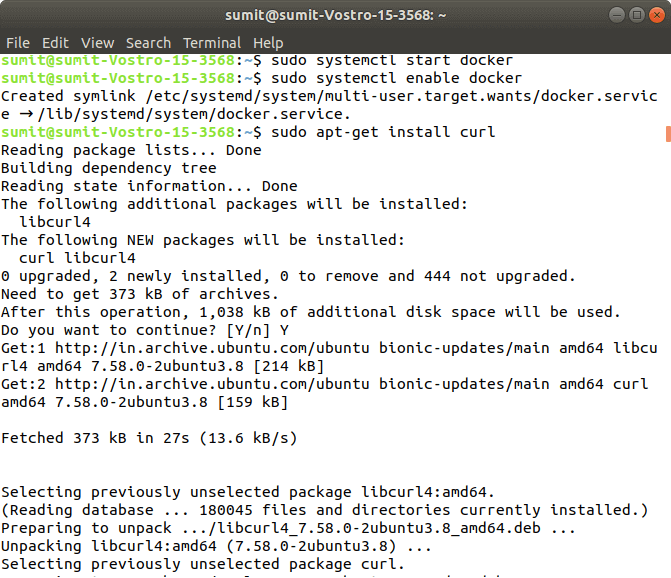
After installing the docker, we have to type the different two commands for starting and enabling the docker. Type the following first command, which starts the docker:

1. sudo systemctl start docker

Now, type the following second command, which enables the docker:

1. sudo systemctl enable docker

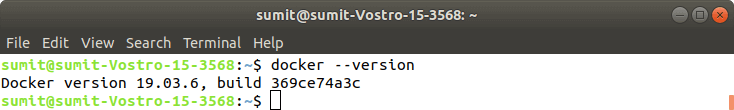
**Output:**



Now, we can check the version of docker by typing the following command:

1. Docker -version

**Output:**

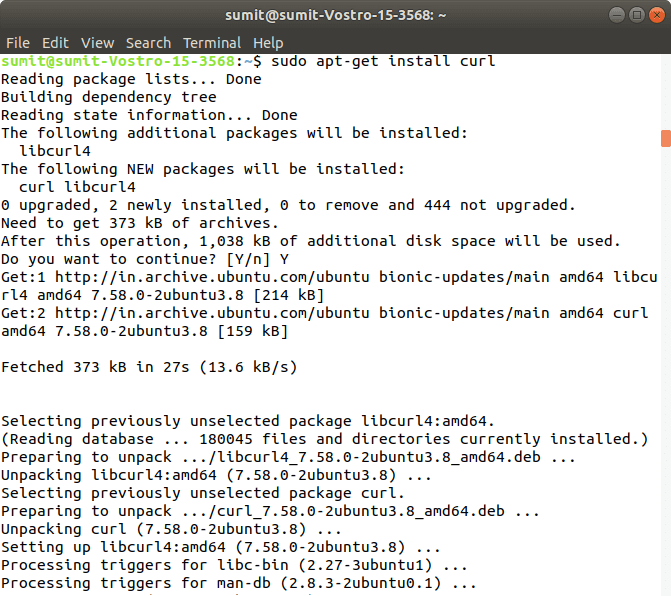


**Step 3:** After the successful execution of all the commands of the second step, we have to install the curl command. The curl is used to send the data using URL syntax.

Now, install the curl by using the following command. In the installation, we have to type Y.

1. sudo apt-get install curl

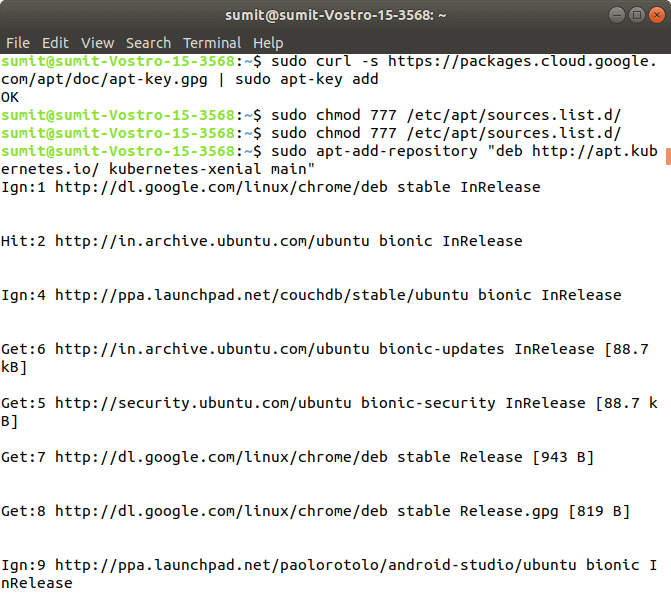
**Output:**



Now, we have to download the add package key for Kubernetes by the following command:

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

**Output:**

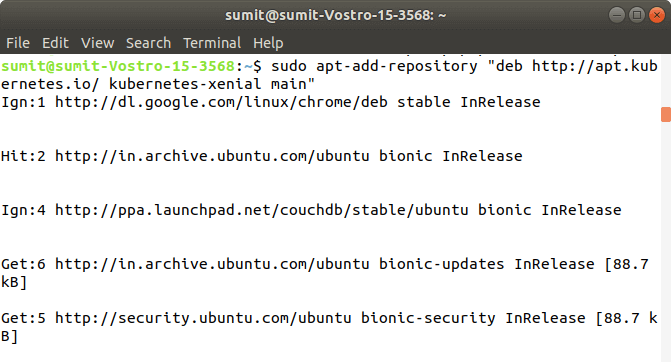


If you get an error from the above command, then it means your curl command is not successfully installed, so first install the curl command, and again run the above command.

Now, we have to add the Kubernetes repositories by the following command:

1. sudo apt-add-repository "deb http://apt.kubernetes.io/ kubernetes-xenial main"

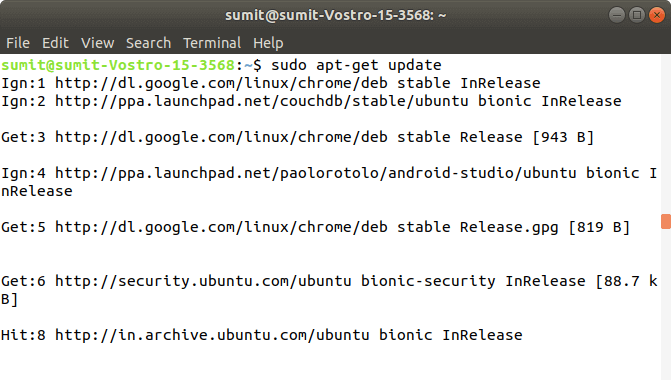
**Output:**



After the successful execution of the above command, we have to check any updates by executing the following command:

1. sudo apt-get update

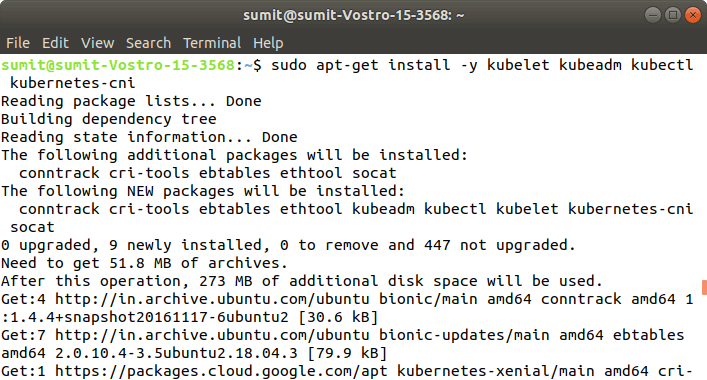
**Output:**



**Step 4:** After the execution of the above commands in the above steps, we have to install the components of Kubernetes by executing the following command:

1. sudo apt-get install -y kubelet kubeadm kubectl kubernetes-cni

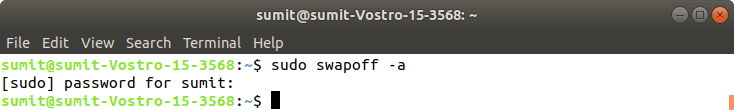
**Output:**



**Step 5:** After the above installation is done, we have to initialize the kubeadm by executing the following command. The following command disables the swapping on other devices:

1. sudo swapoff -a

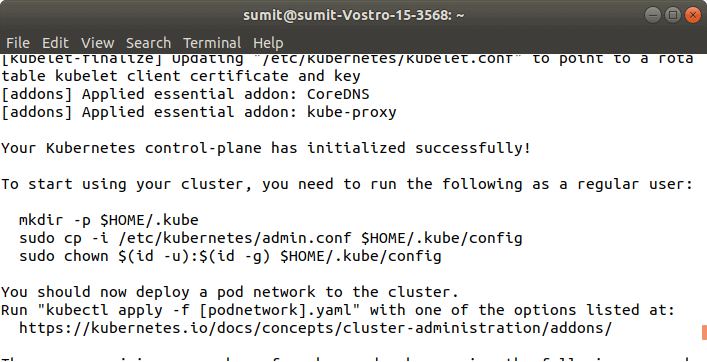
**Output:**



Now, we have to initialize the kubeadm by executing the following command:

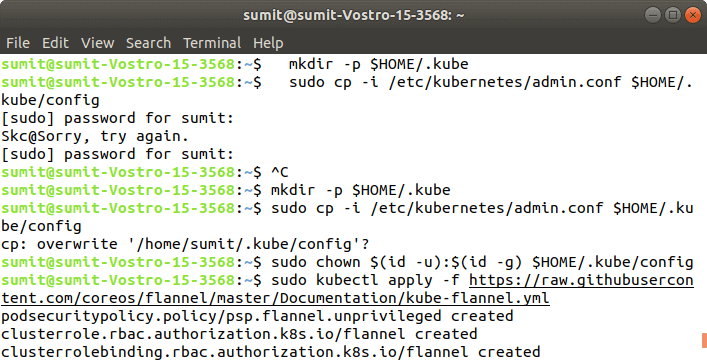
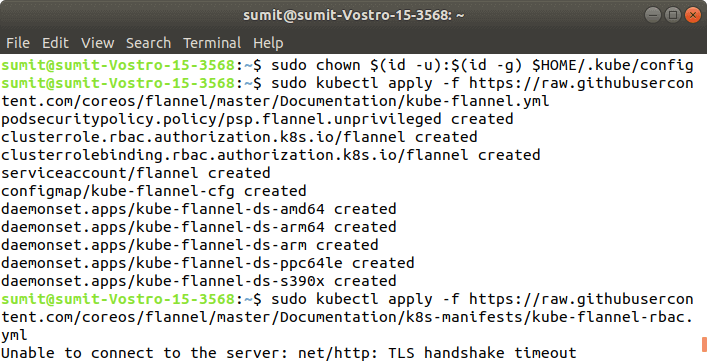
1. sudo kubeadm init

**Output:**



**Step 6:** After the above command is successfully executed, we have to run the following commands, which are given in the initialization of kubeadm. These commands are shown in the above screenshot. The following commands are used to start a cluster:

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

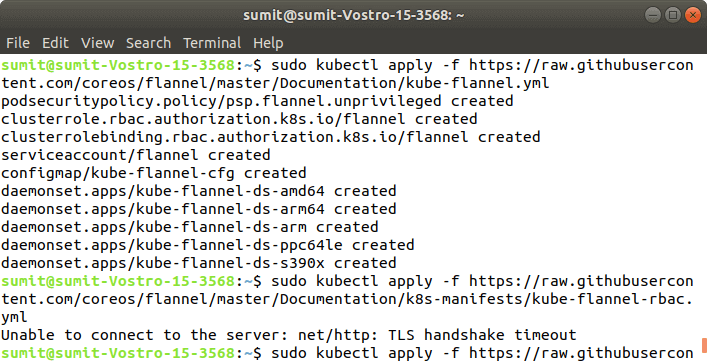
  


**Output:**

**Step 7:** In this step, we have to deploy the paths using the following command:

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

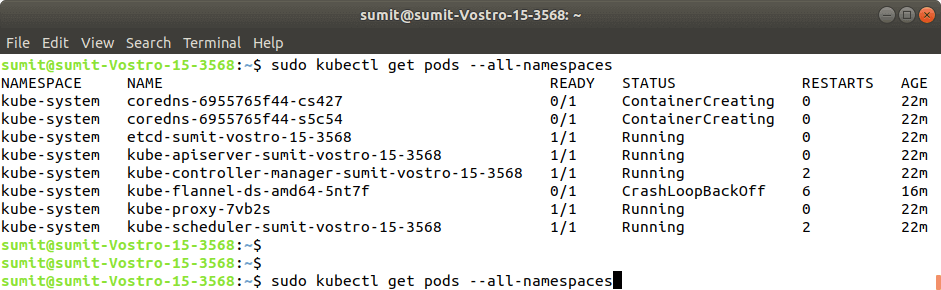
**Output:**



**Step 8:** After the execution of the above command, we have to run the following command to verify the installation:

1. sudo kubectl get pods --all-namespaces

**Output:**



If the output is displayed as shown in the above screenshot. It means that the Kubernetes is successfully installed on our system.