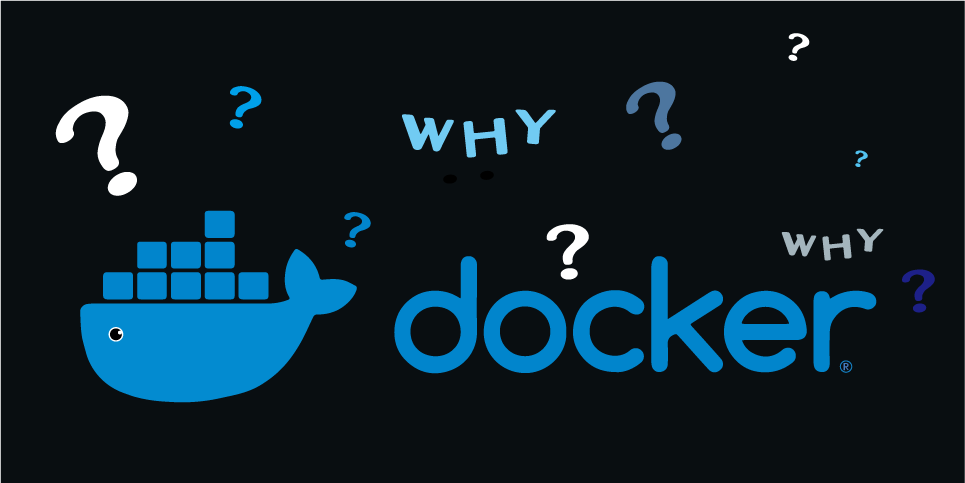
# Docker Documents



Index

[Docker Documents 1](#_vqnh2g8zg647)

[Index 2](#_z0l1qj6joexw)

[Introduction 3](#_fu0kscjrub81)

[History of docker 4](#_wagrqrai5i31)

[What is Docker 5](#_7uej07rh1kqn)

[Architecture of Docker 6](#_5ft6gwovqc9s)

[Types of Docker Storage 9](#_8ouqd73sjx8b)

[Docker Networking 10](#_exhzaqpenflo)

[Docker HUB 11](#_nyc9815a0gny)

[Docker installation and configuration steps 13](#_s2xyolycs27s)

[Docker commands 15](#_qhgsgkr45m79)

[Running Commands Inside Docker Container 20](#_ahei0pfvgsip)

[Method 1 :- 20](#_5uhuk4p98rmn)

[Method 2 :- 21](#_lgazklsz5i6s)

[Method 3 :- 22](#_766lh0t05py0)

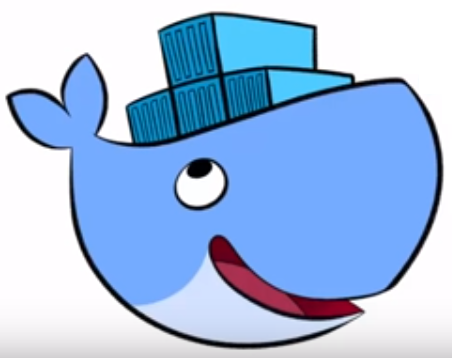
# 

# Introduction

Docker is basically a service which maintain containers, It helps developers to develop, arrange & execute application without installing any extra OS due to this docker make user very flexible and reliable

Docker worked separately from the base machine without using its hardware which make it flexible, portable and lightweight

Docker is a very popular choice by its continuous integration and continuous delivery (CI/CD). It is a set of program which make it easy to develop, make, & deploy software by using CI/CD.

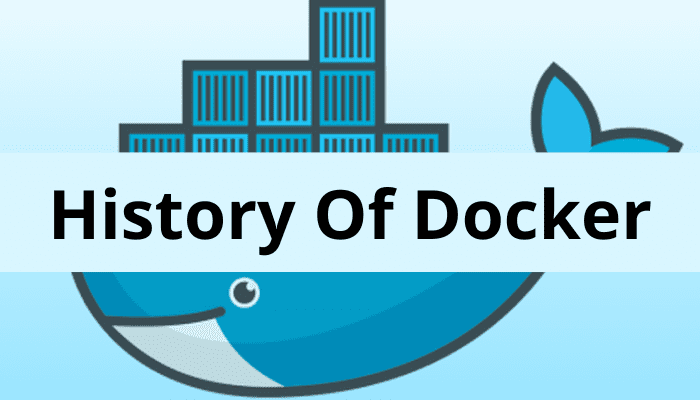


# History of docker

The first time we got familiar with the docker which official name was DotCloud in 2008 founded by solomon Hykes in paris (France). But on that time its was working as a PAAS (platform as a service) for developer to develop and run application.

Docker was first invented by the Solomon Hykes in PyCon in march 2013 for the developers who were constantly looking for the platform to develop or coding to make an application.

Ben Golub who was the CEO of docker once he said, it is easy to ship a car from one country to another country but it is very difficult to ship an application form one sever to another sever, we don’t know an application which i have made it will also work one another server without any problem.

The main purpose was to invent a docker to get a platform where we can develop an application without the need of any other operations system, which will have all the basic system settings, required libraries, & packages.

# What is Docker

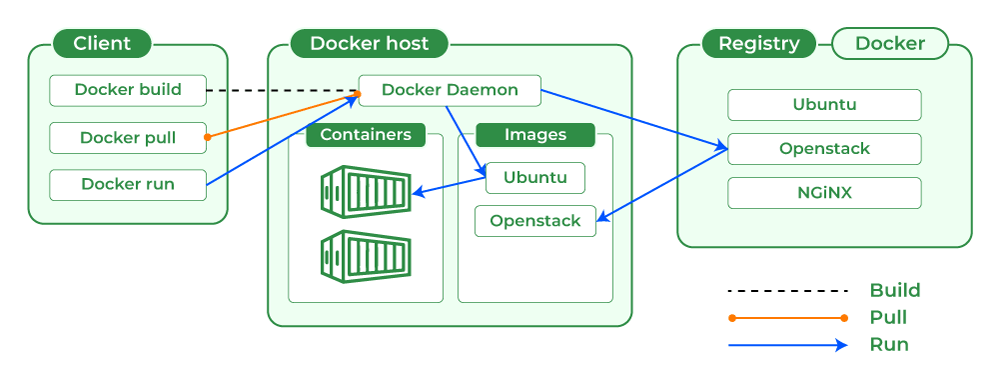
Docker is an open source platform, this is used to virtualize application container where we are able to develop, organize and execute applications. Docker is a containerized tool which help us to make application and execute quickly.

Docker has separate container for each and every application execution, which enable us to manage the infrastructure like we manage application without any delay and effectively.

# 

# Architecture of Docker

The architecture of a docker is used to make client server, docker client intract with the docker daemon for developer to build, run and distribute application on the same machine also we can connect the docker client and docker daemon remotely with the help of REST API.



* **Docker Client** :- as we know, Docker intract with the docker daemon to work with the help of docker client, Docker client using API, with the help of docker client we can connect with multiples of daemon.

When a docker client runs any docker command on the docker terminal then the terminal sends instructions to the daemon. The Docker daemon gets those instructions from the docker client withinside the shape of the command and REST API’s request.

## 

* **Docker Daemon** :- This is used to manage communication service in between client, It is also used to manage objects using docker API like, Image, Container, Networks and storage volumes.
* **Docker Host** :-A Docker host is a type of machine that is responsible for running more than one container. It comprises the Docker daemon, Images, Containers, Networks, and Storage**.**
* **Docker Registry** :- All the software having there own registry file, like docker also have there registry file, where the images have been stored automatically and it can be used by anyone which name is (**Docker Hub)**, this is a public registry in docker.

We can also use the private registry by using **(docker run & docker pull)** command, by this we can pull the stored images in configured registry, images can also be pushed in configured registry by using **docker** **push** command.

* **Docker Objects** :-Whenever we are using a docker, we are creating and use images, containers, volumes, networks, and other objects. Now, we are going to discuss docker objects:-

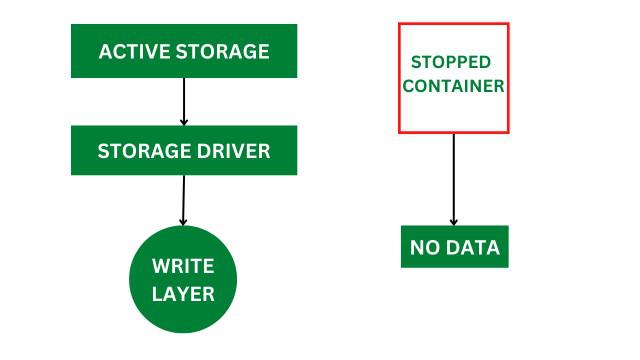
# 

* **Docker Images** :- An image contains instructions for creating a docker container. It is just a read-only template. It is used to store and ship applications. Images are an important part of the docker experience as they enable collaboration between developers in any way which is not possible earlier.
* **Docker containers** :- Containers are created from docker images as they are ready applications. With the help of Docker API or CLI, we can start, stop, delete, or move a container. A container can access only those resources which are defined in the image unless additional access is defined during the building of an image in the container.

## 

* **Docker Storage** :- Docker using docker layer for storing image layer, but for storing data it’s using writable layer of a container. Container writable layer does not exist once container got deleted but it could be able to strot short term data at the running time.

Storage drivers are able to optimize space efficiency but it’s storing speed is too slow in comparison of native file system performance.



# Types of Docker Storage

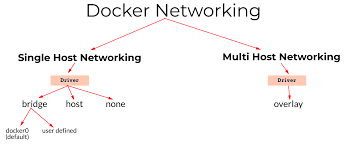
In the docker, there are total 4 types of storage which is listed below in brief.

* **Data Volumes** :- Data Volumes can be mounted directly into the filesystem of the container and are essentially directories or files on the Docker Host filesystem.
* **Volume Container** :- In order to maintain the state of the containers (data) produced by the running container, Docker volumes file systems are mounted on Docker containers. independent container life cycle, the volumes are stored on the host. This makes it simple for users to exchange file systems among containers and backup data.
* **Directory Mounts** :- A host directory that is mounted as a volume in your container might be specified.
* **Storage Plugins** :- Docker volume plugins enable us to integrate the Docker containers with external volumes like Amazon EBS by this we can maintain the state of the container.

# Docker Networking

Docker networking provides complete isolation for docker containers. It means a user can link a docker container to many networks. It requires very less OS instances to run the workload.

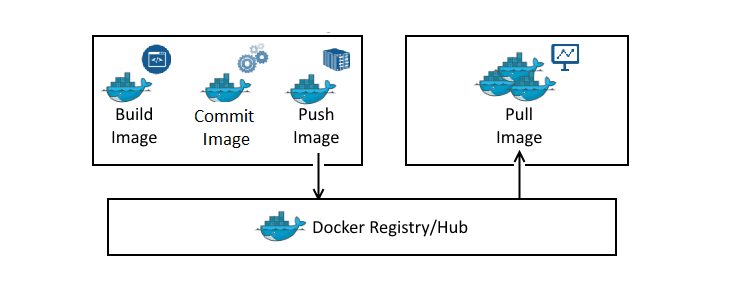
**Types of Docker Network**

1. **Bridge:** It is the default network driver. We can use this when different containers communicate with the same docker host.
2. **Host:** When you don’t need any isolation between the container and host then it is used.
3. **Overlay:** For communication with each other, it will enable the swarm services.
4. **None:** It disables all networking.
5. **macvlan:** This network assigns MAC(Media Access control) address to the containers which look like a physical address.

# Docker HUB

Docker hub is basically a storage point service where people are able to push and pull docker container image anythime and anywhere from the internet, it also provides features to push/pull container image in public or private.

Docker hub is open source tool which is available for all the operating system.



**Docker Hub Features**

* Storage, management, and sharing of images with others are made simple via Docker Hub.
* Docker Hub runs the necessary security checks on our images and generates a full report on any security flaws.
* Docker Hub can automate the processes like Continuous deployment and Continuous testing by triggering the Webhooks when the new image is pushed into Docker Hub.
* With the help of Docker Hub, we can manage the permission for the users, teams, and organizations.
* We can integrate Docker Hub into our tools like GitHub, jenkns which makes workflows easy

**Advantages of Docker Hub**

* Docker Container Images are light in weight.
* We can push the images within a minute and with help of a command.
* It is a secure method and also provides a feature like pushing the private image or public image.
* Docker hub plays a very important role in industries as it becomes more popular day by day and it acts as a bridge between the developer team and the testing team.
* If a person wants to share their code, software any type of file for public use, you can just make the images public on the docker hub.

# Docker installation and configuration steps

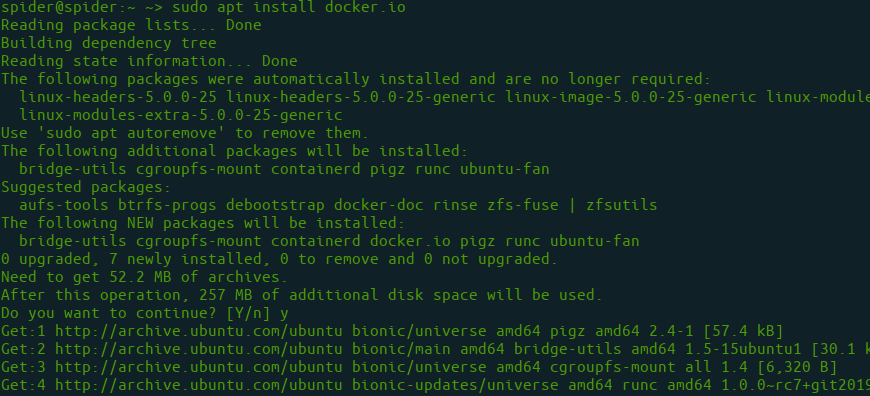
Docker is a platform and service-based product which utilizes OS-level virtualization to deliver software in packages called containers. Containers are separated from one another and bundle their own software, libraries, and configuration files. Docker is written in the Go language. Docker can be installed in two versions Docker CE(Community Edition) and Docker EE(Enterprise Edition).

**Step 1**:- First of all, update the system software repository by using the below command.

**#sudo apt update**

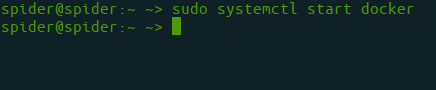
**Step 2**:- Install docker program using the below command.

**#sudo apt install docker.io -y**

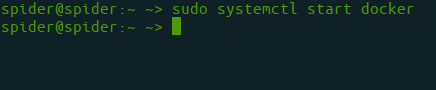
****

**Step 3**:- Enable and start Docker.

**#sudo systemctl enable docker**

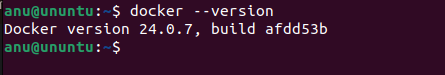
****

**#sudo systemctl start docker**

****

**Step 4**:- Check Docker Version.

**#docker --version**

****

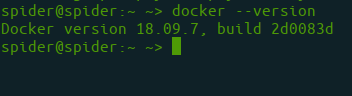
**Step 5**:- We will get a permission denied error as a regular user doesn’t have permission to execute docker commands, so we need to add an Ubuntu user to the docker group.

**#sudo usermod -aG docker $USER**

or

**#sudo usermod -aG docker ubuntu**

**Step 6**:- Leave the current SSH terminal and re-login with SSH. then carry out.

**#docker ps**

# Docker commands

Docker is a open source platform which automate the deployment of application as movable and independent container that can be run locally or cloud based, by using the command we could be able to manage the infrastructure as we managing the application.

Docker also using lots of command which is not possible to discuss in a single documents but from this document we will discuss about some basic command which can be used to run a container and manage infra.

**Docker Run command :-** Below command is the combination of two command, to create a docker and start a docker

**#docker run image\_name**

To provide name of container.

**#Docker run –-name container\_name Image\_name**

**Docker pull command:**- This command is used to pull image file directly from the docker hub or repository. It can be pull the latest image and also we can use the image name to pull any specific image.

**#Docker pull image\_name**

**Docker PS:-** By using this command we can see the list of all running containers, also we can use the flags (options command).

* **-a flag:**  shows us all the containers, stopped or running.
* **-l flag:** shows us the latest container.
* **-q flag**: shows only the Id of the containers.

**#Docker ps (option command)**

**Docker stop**:- this command is used to stop container if we want to stop and switchf from one container to another container.

**#docekr stop container\_ID or Container\_name**

**Docker start:-** If you want to run any new container or existing container then below command is used to run.

**#docekr start container\_ID or Container\_name**

**Remove container:-** Below command used to delete particular container bu using there name or ID.

**#docker rm container\_name or Container\_ID**

We can also use the below flags with docker rm command.

* **f flag:** remove the container forcefully.
* **-v flag:** remove the volumes.
* **-l flag:** remove the specific link mentioned.

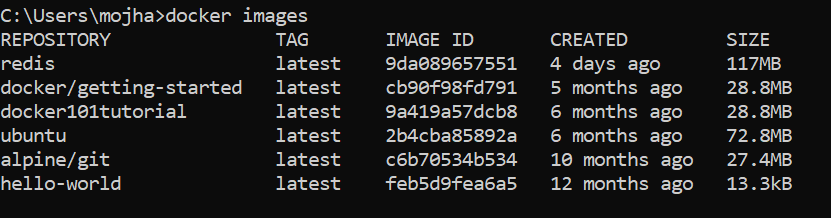
**$ docker rm (options) container\_name or container\_ID**

**Docker RMI:** docker RMI is used to remove existing image from the registry of docker or docker hub.

**#docker RMI image\_name/Image\_ID**

**Docker image:-** this command is used to show all the available image on your system.

**#Docker image**

****

**Docker exec :-** this command is used to run new command on running container, if you closed or restart the container then the command will not restart automatically. We can also use below flags with this command.

* -d flag: for running the commands in the background.
* -i flag: it will keep STDIN open even when not attached.
* -e flag: sets the environment variables

**#docker exec {options}**

### 

**Docker Ports (Port Mapping) :-** In order to access the [docker container](https://www.geeksforgeeks.org/containerization-using-docker/) from the outside world, we have to map the port on our host( Our laptop for example), to the port on the container. This is where port mapping comes into play.

**#docker run -d -p port\_on\_host port\_on\_container Container\_name**

So these were the 9 most basic docker commands that every beginner must know. Containerization is a very vast topic but you can start from the very basic commands and by practicing them daily you can master them.

**Docker Login :-** The Docker login command will help you to authenticate with the Docker hub by which you can push and pull your images.

**#docker login**

It will ask you to enter the username and password after that you will authenticate with DockerHub and you can perform the tasks.

**Docker Build :-** The docker build command is used to build the docker images with the help of Dockerfile.

**#docker build -t image\_name:tag**

In the place of image\_name use the name of the image you build with and give the tag number and . “dot” represents the current directory.

**Docker Stop :-**You can stop and start the docker containers where you can do the maintenance for containers. To stop and start specific containers you can use the following commands.

**#docker stop container\_name\_or\_id**

**Stop Multiple Containers** :- Instead of stopping a single container. You can stop multiple containers at a time by using the following commands.

**#docker stop container1 container2 container3**

**Docker Inspection** :- Docker containers will run into some errors in real time to debug the container’s errors you can use the following commands.

**#docker inspect container\_name\_or\_id**

**Docker Commit command :-** After running the containers by using the current image you can make the updates to the containers by interacting with the containers from that containers you can create an image by using the following commands.

**#docker commit container\_name\_or\_id new\_image\_name:tag**

# Running Commands Inside Docker Container

In the running docker container we need commands to install the packages for applications to access the file system, docker provide us multiple ways to install packages in the container.

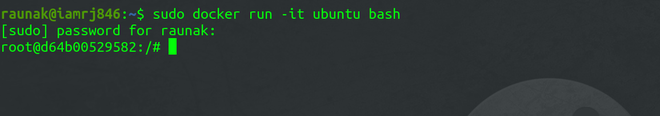
In this, will learn few ways that how commands work in docker terminal.

# **Method 1** :-

Using Bash.

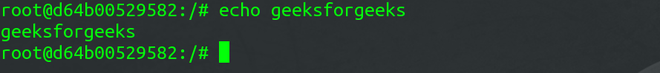
This is very easy to access bash in docker container and execute commands, we can access the bash in docker using below command.

**#sudo docker run -it ubuntu bash**



Once we get Bash access, we don’t need to enter password again and again. Its start execiting all command without any authorization acess, Like:-

**#Echo Geeksforgeek**



# **Method 2** :-

Using the Docker exec Command.

In order to run a command inside a Docker Container using the exec command, you have to know the Container Id of the Docker Container. You can get the Container Id using the following Command.

**#sudo docker container ls**

or

**#sudo docker ps -a**

****

Once you have the Container ID, you can use the Docker exec command. But you have to confirm that the Container is running before you can execute the exec command. To start the Container, use this command.

**#sudo docker start d64b00529582**

After that, execute the exec command.

**#sudo docker exec -it d64b00529582 echo "GeeksforGeeks"**

# **Method 3** :-

By using the Dockerfile.

When you are creating a large application, it is always advised that you execute your commands by specifying it inside the Dockerfile. However, you should only include those commands inside the Dockerfile which you want to execute while building the Container. For executing commands on the go, you can use any of the two methods above. To execute commands through Dockerfile, you can specify them using the Docker Run Commands.

FROM ubuntu:latest

**#RUN echo "geeksforgeeks"**

After you have created the above Dockerfile, you can build the images using the Docker build command.

**#sudo docker build -t sample-image**

