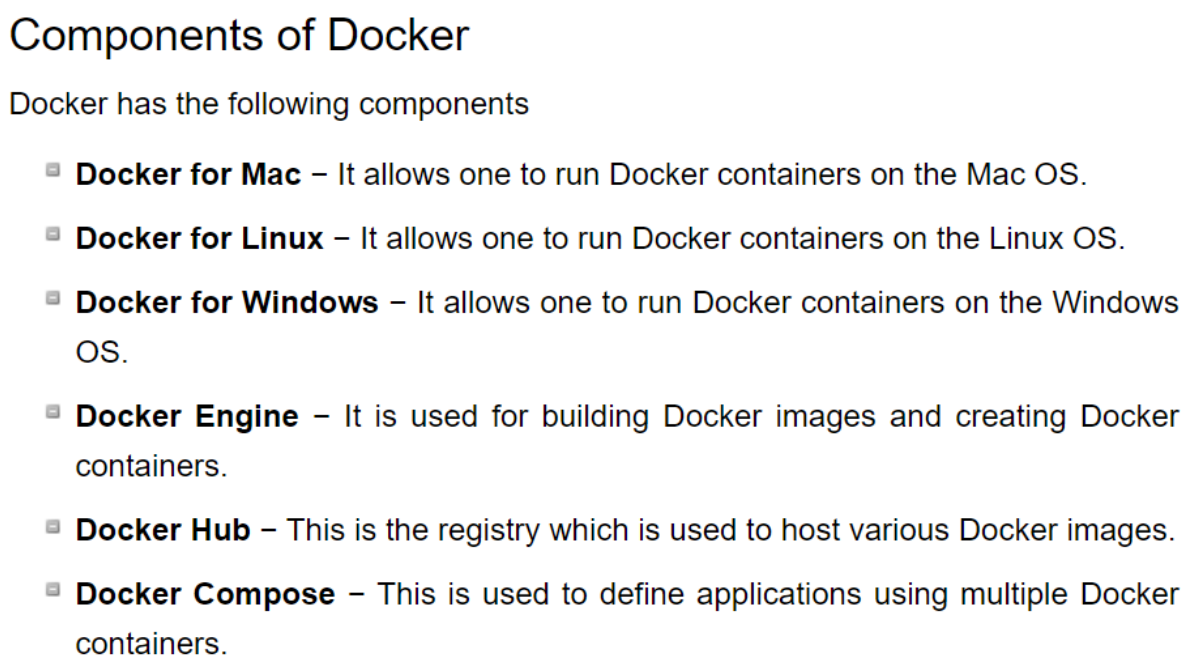
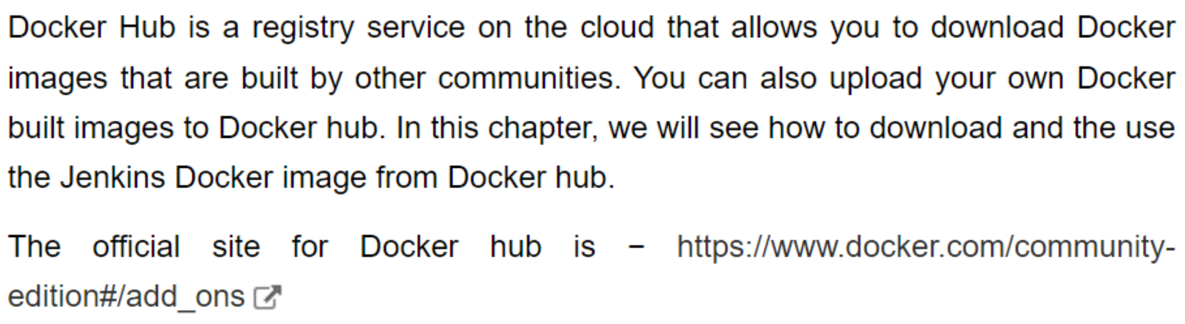
Docker is a container management service. The keywords of Docker are **develop, ship** and **run** anywhere. The whole idea of Docker is for developers to easily develop applications, ship them into containers which can then be deployed anywhere.



Components of Docker

Docker has the following components

- **Docker for Mac** – It allows one to run Docker containers on the Mac OS.
- **Docker for Linux** – It allows one to run Docker containers on the Linux OS.
- **Docker for Windows** – It allows one to run Docker containers on the Windows OS.
- **Docker Engine** – It is used for building Docker images and creating Docker containers.
- **Docker Hub** – This is the registry which is used to host various Docker images.
- **Docker Compose** – This is used to define applications using multiple Docker containers.



Docker Hub is a registry service on the cloud that allows you to download Docker images that are built by other communities. You can also upload your own Docker built images to Docker hub. In this chapter, we will see how to download and the use the Jenkins Docker image from Docker hub.

The official site for Docker hub is – https://www.docker.com/community-edition#/add_ons

In Docker, everything is based on Images. An image is a combination of a file system and parameters. Let's take an example of the following command in Docker.

```
docker run hello-world
```

- The Docker command is specific and tells the Docker program on the Operating System that something needs to be done.
- The **run** command is used to mention that we want to create an instance of an image, which is then called a **container**.
- Finally, "hello-world" represents the image from which the container is made.

Now let's look at how we can use the CentOS image available in Docker Hub to run CentOS on our Ubuntu machine. We can do this by executing the following command on our Ubuntu machine –

```
sudo docker run -it centos /bin/bash
```

Note the following points about the above **sudo** command –

- We are using the **sudo** command to ensure that it runs with **root** access.
- Here, **centos** is the name of the image we want to download from Docker Hub and install on our Ubuntu machine.
- **─it** is used to mention that we want to run in **interactive mode**.
- **/bin/bash** is used to run the bash shell once CentOS is up and running.

Displaying Docker Images

To see the list of Docker images on the system, you can issue the following command.

```
docker images
```

This command is used to display all the images currently installed on the system.

Syntax

```
docker images
```

Options

None

Return Value

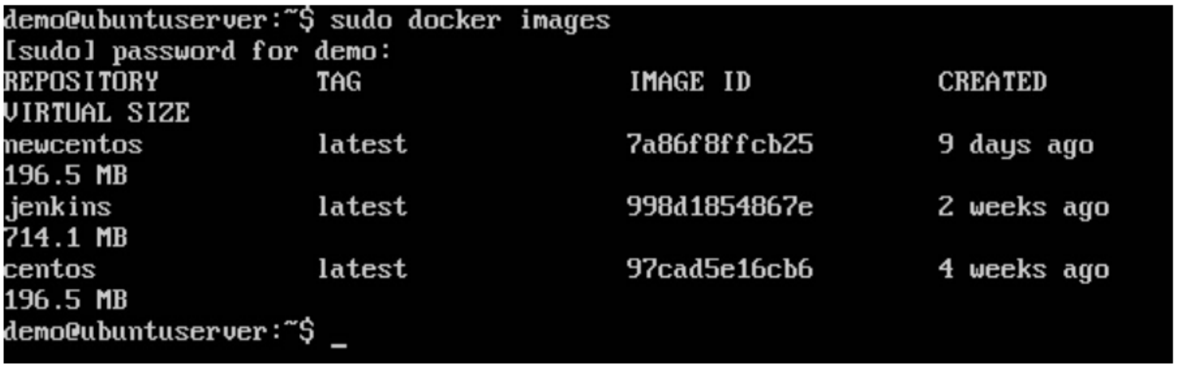
The output will provide the list of images on the system.

Example

```
sudo docker images
```

Output

When we run the above command, it will produce the following result −



```
demo@ubuntuserver:~$ sudo docker images
[sudo] password for demo:
REPOSITORY          TAG                 IMAGE ID            CREATED
VIRTUAL SIZE
newcentos           latest              7a86f8ffcb25        9 days ago
196.5 MB
jenkins             latest              998d1854867e        2 weeks ago
714.1 MB
centos              latest              97cad5e16cb6        4 weeks ago
196.5 MB
demo@ubuntuserver:~$ _
```

From the above output, you can see that the server has three images: **centos, newcentos,** and **jenkins**. Each image has the following attributes −

- **TAG** − This is used to logically tag images.
- **Image ID** − This is used to uniquely identify the image.
- **Created** − The number of days since the image was created.
- **Virtual Size** − The size of the image.

Downloading Docker Images

Images can be downloaded from Docker Hub using the Docker **run** command. Let's see in detail how we can do this.

Syntax

The following syntax is used to run a command in a Docker container.

```
docker run image
```

Options

- **Image** − This is the name of the image which is used to run the container.

Return Value

The output will run the command in the desired container.

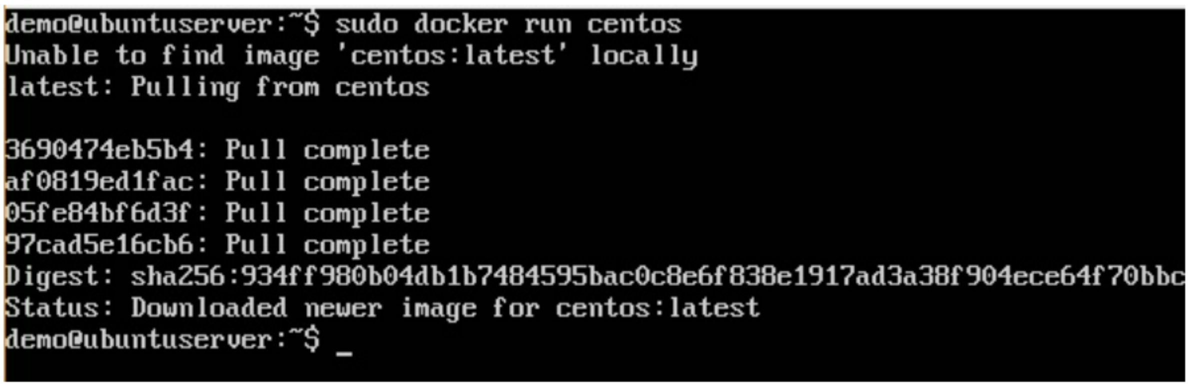
Example

```
sudo docker run centos
```

This command will download the **centos** image, if it is not already present, and run the OS as a container.

Output

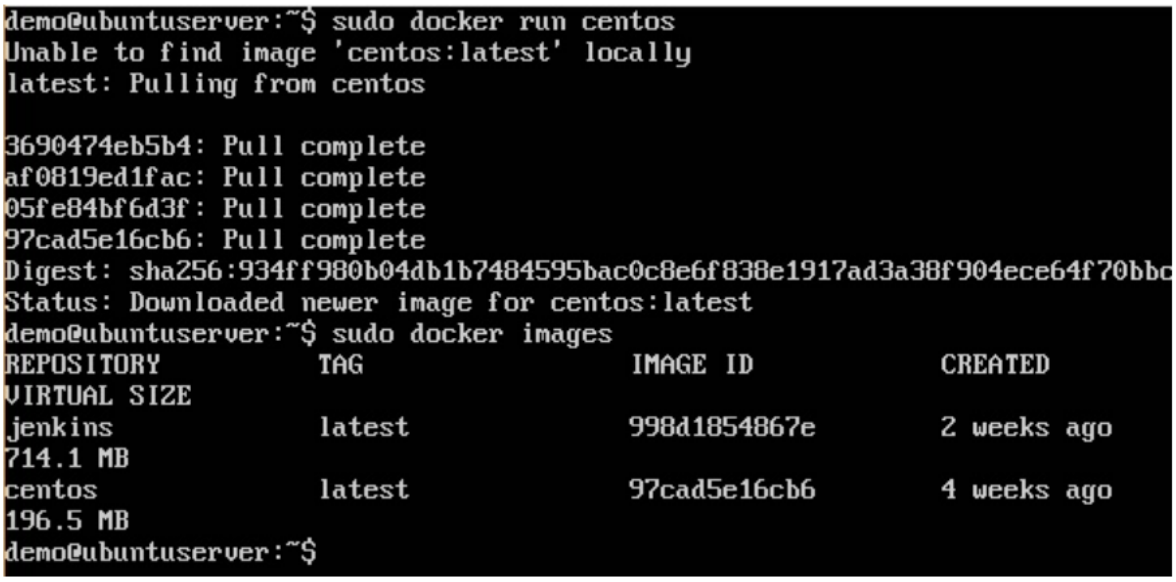
When we run the above command, we will get the following result −



```
demo@ubuntuserver:~$ sudo docker run centos
Unable to find image 'centos:latest' locally
latest: Pulling from centos

3690474eb5b4: Pull complete
af0819ed1fac: Pull complete
05fe84bf6d3f: Pull complete
97cad5e16cb6: Pull complete
Digest: sha256:934ff980b04db1b7484595bac0c8e6f838e1917ad3a38f904ece64f70bbc
Status: Downloaded newer image for centos:latest
demo@ubuntuserver:~$ _
```

You will now see the CentOS Docker image downloaded. Now, if we run the Docker **images** command to see the list of images on the system, we should be able to see the **centos** image as well.



```
demo@ubuntuserver:~$ sudo docker run centos
Unable to find image 'centos:latest' locally
latest: Pulling from centos

3690474eb5b4: Pull complete
af0819ed1fac: Pull complete
05fe84bf6d3f: Pull complete
97cad5e16cb6: Pull complete
Digest: sha256:934ff980b04db1b7484595bac0c8e6f838e1917ad3a38f904ece64f70bbc
Status: Downloaded newer image for centos:latest
demo@ubuntuserver:~$ sudo docker images
REPOSITORY          TAG                 IMAGE ID            CREATED
VIRTUAL SIZE
jenkins             latest              998d1854867e        2 weeks ago
714.1 MB
centos              latest              97cad5e16cb6        4 weeks ago
196.5 MB
demo@ubuntuserver:~$
```

Removing Docker Images

The Docker images on the system can be removed via the **docker rmi** command. Let's look at this command in more detail.

```
docker rmi
```

This command is used to remove Docker images.

Syntax

```
docker rmi ImageID
```

Options

- **ImageID** − This is the ID of the image which needs to be removed.

Return Value

The output will provide the Image ID of the deleted Image.

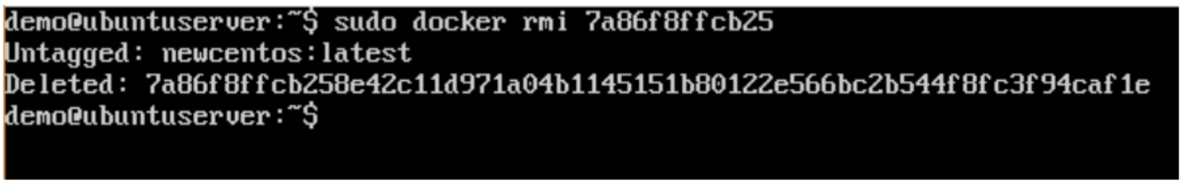
Example

```
sudo docker rmi 7a86f8ffcb25
```

Here, **7a86f8ffcb25** is the Image ID of the **newcentos** image.

Output

When we run the above command, it will produce the following result −



```
demo@ubuntuserver:~$ sudo docker rmi 7a86f8ffcb25
Untagged: newcentos:latest
Deleted: 7a86f8ffcb258e42c11d971a04b1145151b80122e566bc2b544f8fc3f94caf1e
demo@ubuntuserver:~$
```

Let's see some more Docker commands on images.

docker images -q

This command is used to return only the Image ID's of the images.

Syntax

```
docker images
```

Options

- **q** − It tells the Docker command to return the Image ID's only.

Return Value

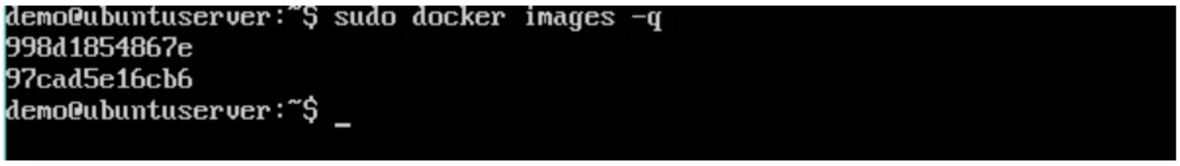
The output will show only the Image ID's of the images on the Docker host.

Example

```
sudo docker images -q
```

Output

When we run the above command, it will produce the following result −



```
demo@ubuntuserver:~$ sudo docker images -q
998d1854867e
97cad5e16cb6
demo@ubuntuserver:~$ _
```

docker inspect

This command is used see the details of an image or container.

Syntax

```
docker inspect Repository
```

Options

- **Repository** − This is the name of the Image.

Return Value

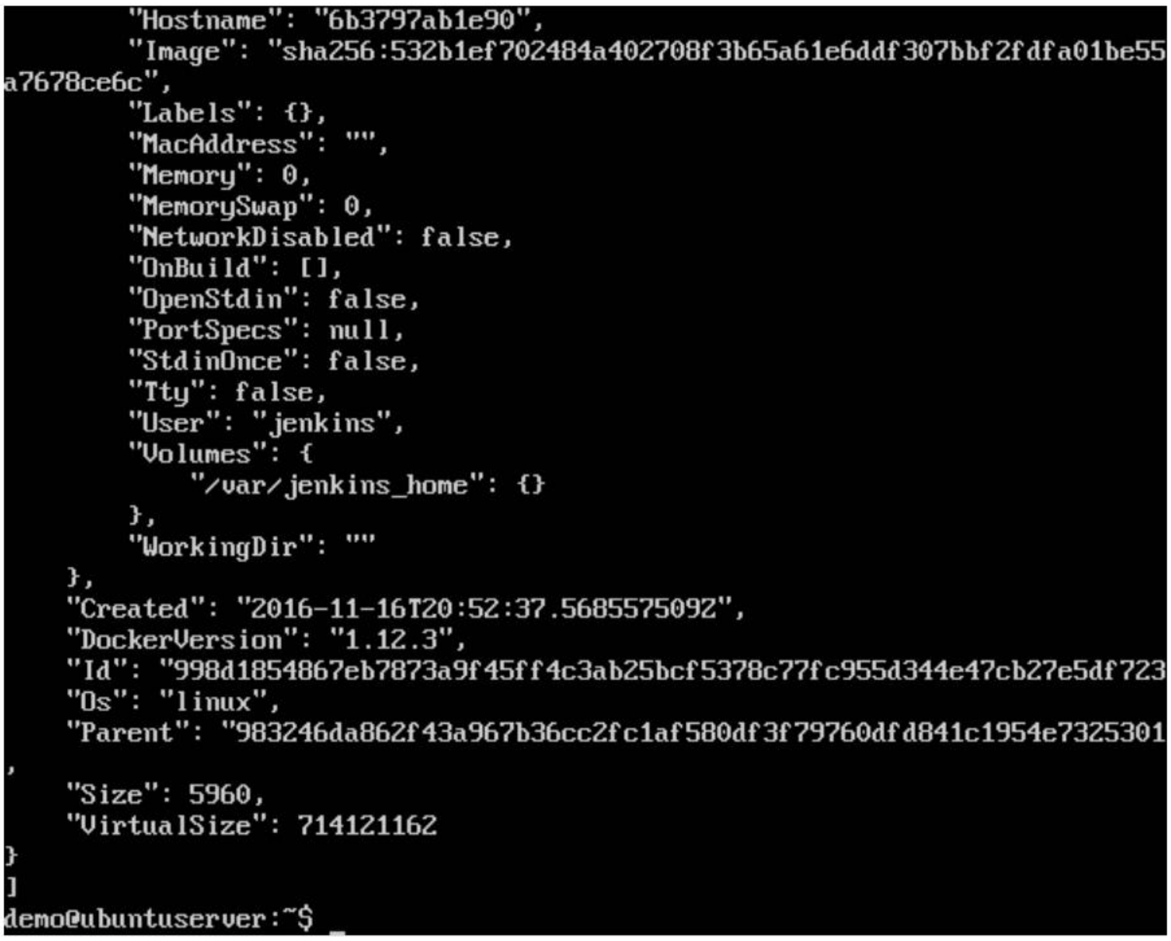
The output will show detailed information on the Image.

Example

```
sudo docker inspect jenkins
```

Output

When we run the above command, it will produce the following result −



```
        "Hostname": "6b3797ab1e90",
        "Image": "sha256:532b1ef702484a402708f3b65a61e6ddf307bbf2fdfa01be55
a7678ce6c",
        "Labels": {},
        "MacAddress": "",
        "Memory": 0,
        "MemorySwap": 0,
        "NetworkDisabled": false,
        "OnBuild": [],
        "OpenStdin": false,
        "PortSpecs": null,
        "StdinOnce": false,
        "Tty": false,
        "User": "jenkins",
        "Volumes": {
            "/var/jenkins_home": {}
        },
        "WorkingDir": ""
    },
    "Created": "2016-11-16T20:52:37.568557509Z",
    "DockerVersion": "1.12.3",
    "Id": "998d1854867eb7873a9f45ff4c3ab25bcf5378c77fc955d344e47cb27e5df723
    "Os": "linux",
    "Parent": "983246da862f43a967b36cc2fc1af580df3f79760dfd841c1954e7325301
,
    "Size": 5960,
    "VirtualSize": 714121162
}
]
demo@ubuntuserver:~$ _
```