## **Docker**

## 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.

## **Features of Docker**

1.Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.

2.With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.

3.You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.

4.Since Docker containers are pretty lightweight, they are very easily scalable.

**Components of Docker**

**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**

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.

**Docker - Images**

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

**Displaying Docker Images**

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

docker images

Containers are instances of Docker images that can be run using the Docker run command. The basic purpose of Docker is to run containers. Let’s discuss how to work with containers.

**Running a Container**

Running of containers is managed with the Docker run command. To run a container in an interactive mode, first launch the Docker container.

sudo docker run –it centos /bin/bash

**Listing of Containers**

One can list all of the containers on the machine via the **docker ps** command. This command is used to return the currently running containers.

docker ps

Syntax

docker ps

**Docker – Configuring**

Now,we will look at the different options to configure Docker.

service docker stop

This command is used to stop the Docker **daemon** process.

Syntax

service docker stop

**Docker Public Repositories**

Public repositories can be used to host Docker images which can be used by everyone else. An example is the images which are available in Docker Hub. Most of the images such as Centos, Ubuntu, and Jenkins are all publicly available for all. We can also make our images available by publishing it to the public repository on Docker Hub.

**Docker Private Registries**

You might have the need to have your own private repositories. You may not want to host the repositories on Docker Hub.

Use the Docker run command to download the private registry. This can be done using the following command.

sudo docker run –d –p 5000:5000 –-name registry registry:2

# Docker - Building a Web Server Docker File

Let’s see how we can build a web server image which can be used to build containers.

In our example, we are going to use the Apache Web Server on Ubuntu to build our image. Let’s follow the steps given below, to build our web server Docker file.

**Step 1** − The first step is to build our Docker File. Let’s use **vim** and create a Docker File with the following information.

RUN apt-get update

RUN apt-get install –y apache2

RUN apt-get install –y apache2-utils

RUN apt-get clean

EXPOSE 80 CMD [“apache2ctl”, “-D”, “FOREGROUND”]

The following points need to be noted about the above statements −

* We are first creating our image to be from the Ubuntu base image.
* Next, we are going to use the RUN command to update all the packages on the Ubuntu system.
* Next, we use the RUN command to install apache2 on our image.
* Next, we use the RUN command to install the necessary utility apache2 packages on our image.
* Next, we use the RUN command to clean any unnecessary files from the system.
* The EXPOSE command is used to expose port 80 of Apache in the container to the Docker host.

**Step 2** − Run the Docker **build** command to build the Docker file. It can be done using the following command −

sudo docker build –t=”mywebserver”

**Step 3** − Now that the web server file has been built, it’s now time to create a container from the image. We can do this with the Docker **run** command.

sudo docker run –d –p 80:80 mywebserver

The following points need to be noted about the above command −

* The port number exposed by the container is 80. Hence with the **–p** command, we are mapping the same port number to the 80 port number on our localhost.
* The **–d** option is used to run the container in detached mode. This is so that the container can run in the background.

**Docker - Building Files**

The Docker File can be built with the following command −

docker build

**Docker build**

This method allows the users to build their own Docker images.

Syntax

docker build -t ImageName:TagName dir

**Docker - Networking**

Docker takes care of the networking aspects so that the containers can communicate with other containers and also with the Docker Host. If you do an **ifconfig** on the Docker Host, you will see the Docker Ethernet adapter. This adapter is created when Docker is installed on the Docker Host.

## IFCONFIG

## **Listing All Docker Networks**

This command can be used to list all the networks associated with Docker on the host.

Syntax

docker network ls

**Docker - Compose**

Docker Compose is used to run multiple containers as a single service. For example, suppose you had an application which required NGNIX and MySQL, you could create one file which would start both the containers as a service without the need to start each one separately. how to get the docker-compose version.

sudo ./docker-compose -version

**Docker – Storage**

**Storage Drivers**

Docker has multiple storage drivers that allow one to work with the underlying storage devices. The following table shows the different storage drivers along with the technology used for the storage drivers.

|  |  |
| --- | --- |
| **Technology** | **Storage Driver** |
| OverlayFS | overlay or overlay2 |
| AUFS | aufs |
| Btrfs | brtfs |
| Device Manager | devicemanager |
| VFS | vfs |
| ZFS | zfs |

Let us now discuss some of the instances in which you would use the various storage drivers −

**AUFS**

* This is a stable driver; can be used for production-ready applications.
* It has good memory usage and is good for ensuring a smooth Docker experience for containers.
* There is a high-write activity associated with this driver which should be considered.
* It’s good for systems which are of Platform as a service type work.

**Device mapper**

* This is a stable driver; ensures a smooth Docker experience.
* This driver is good for testing applications in the lab.
* This driver is in line with the main Linux kernel functionality.

**Btrfs**

* This driver is in line with the main Linux kernel functionality.
* There is a high-write activity associated with this driver which should be considered.
* This driver is good for instances where you maintain multiple build pools.

**Ovelay**

* This is a stable driver and it is in line with the main Linux kernel functionality.
* It has a good memory usage.
* This driver is good for testing applications in the lab.

**ZFS**

* This is a stable driver and it is good for testing applications in the lab.
* It’s good for systems which are of Platform-as-a-Service type work.

To see the storage driver being used, issue the **docker info** command.

Syntax

docker info