**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

* Docker has the ability to reduce the size of development by providing a smaller footprint of the operating system via containers.
* With containers, it becomes easier for teams across different units, such as development, QA and Operations to work seamlessly across applications.
* You can deploy Docker containers anywhere, on any physical and virtual machines and even on the cloud.
* Since Docker containers are pretty lightweight, they are very easily scalable.

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.

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

Images

• Images are read only templates used to create containers.

• Images are created with the docker build command, either

by us or by other docker users.

• Images are composed of layers of other images.

• Images are stored in a Docker registry.

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.

## 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.

## 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.

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

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.

## Listing All Docker Networks

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

### **Syntax**

docker network ls

### **Options**

None

### **Return Value**

The command will output all the networks on the Docker Host.

## nspecting a Docker network

If you want to see more details on the network associated with Docker, you can use the Docker **network inspect** command.

### **Syntax**

docker network inspect networkname

### **Options**

* **networkname** − This is the name of the network you need to inspect.

### **Return Value**

The command will output all the details about the network.

## Creating Your Own New Network

One can create a network in Docker before launching containers. This can be done with the following command −

### **Syntax**

docker network create –-driver drivername name

### **Options**

* **drivername** − This is the name used for the network driver.
* **name** − This is the name given to the network.

### **Return Value**

The command will output the long ID for the new network.

The following image shows the standard and traditional architecture of **virtualization**.

None Network

• Provides the maximum level of network protection.

• Not a good choice if network or Internet connection is required.

• Suites well where the container require the maximum level of

network security and network access is not necessary.

Host Network

• Minimum network security level.

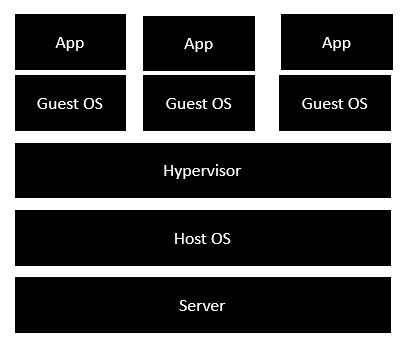
• No isolation on this type of open containers, thus leave the

container widely unprotected.

• Containers running in the host network stack should see a higher

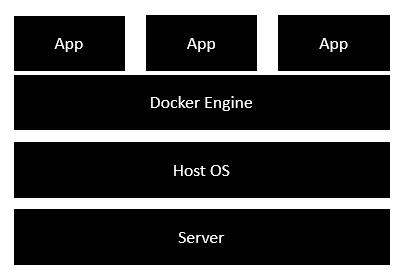
level of performance than those traversing the docker0 bridge and

iptables port mappings.



* The server is the physical server that is used to host multiple virtual machines.
* The Host OS is the base machine such as Linux or Windows.
* The Hypervisor is either VMWare or Windows Hyper V that is used to host virtual machines.
* You would then install multiple operating systems as virtual machines on top of the existing hypervisor as Guest OS.
* You would then host your applications on top of each Guest OS.

The following image shows the new generation of virtualization that is enabled via Dockers. Let’s have a look at the various layers.



* The server is the physical server that is used to host multiple virtual machines. So this layer remains the same.
* The Host OS is the base machine such as Linux or Windows. So this layer remains the same.
* Now comes the new generation which is the Docker engine. This is used to run the operating system which earlier used to be virtual machines as Docker containers.
* All of the Apps now run as Docker containers.