**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.

**Displaying Docker Images**

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

**Syntax**

docker images

**Return Value**

The output will provide the list of images on the system

sudo docker images

**Output**

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

Displaying Docker Images



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

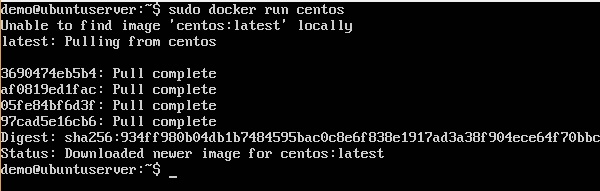
The output will run the command in the desired container.

**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:



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.



**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.

**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.

**Command:**

sudo docker run –it centos /bin/bash

Then hit Crtl+p and you will return to your OS shell.

**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.

**Syntax:**

docker ps

Return Value

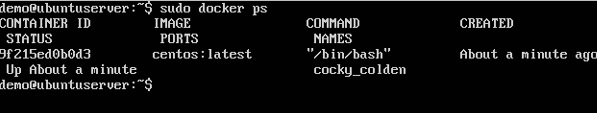
The output will show the currently running containers.

**Example**

sudo docker ps

**Output**

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



**docker ps -a**

This command is used to list all of the containers on the system

**Syntax**

docker ps -a

─a − It tells the docker ps command to list all of the containers on the system

**Example**

sudo docker ps -a

**Output**

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



**Docker history**

With this command, you can see all the commands that were run with an image via a container.

**Syntax**

docker history ImageID

ImageID − This is the Image ID for which you want to see all the commands that were run against it.

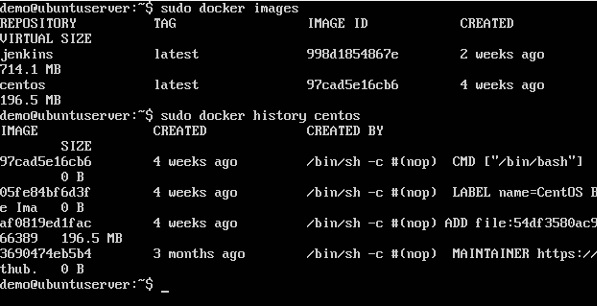
**Example**

sudo docker history centos

The above command will show all the commands that were run against the centos image.

**Output**

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

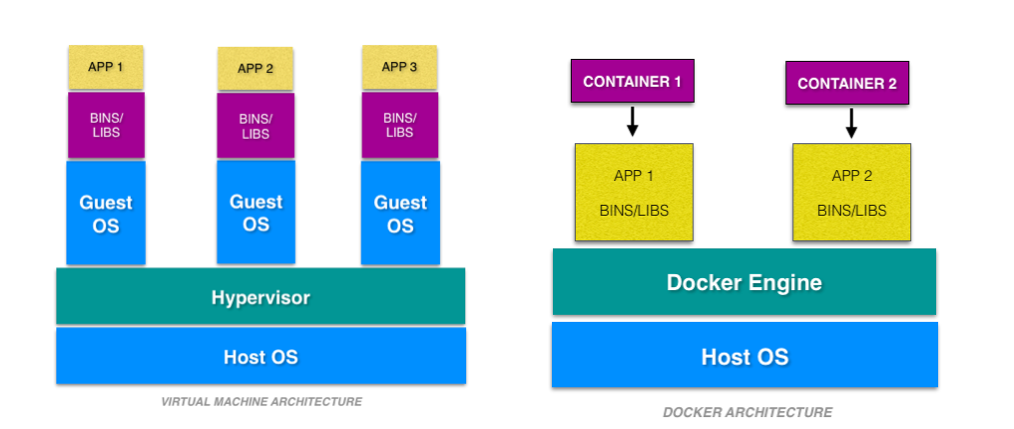


**Understand difference between Virtual Machine and docker containers**

**Docker vs.** **VM**

The significant differences between docker containers and virtual machines. Well, the significant differences are their operating system support, security, portability, and performance.

**Operating system support**



From the image above, you can see each virtual machine has its guest operating system above the host operating system, which makes virtual machines heavy. While on the other hand, Docker containers share the host operating system, and that is why they are lightweight.

The docker containers are suited for situations where you want to run multiple applications over a single operating system kernel. But if you have applications or servers that need to run on different operating system flavors, then virtual machines are required.

**Security**

A container have a lot of security risks, and vulnerabilities as the containers have shared host kernel.

Also, since docker resources are shared and not namespaced, an attacker can exploit all the containers in a cluster if he/she gets access to even one container.

In a virtual machine, you don’t get direct access to the resources, and hypervisor is there to restrict the usage of resources in a VM.

**Portability**

Docker containers are easily portable because they do not have separate operating systems.

On the other hand, virtual machines have separate OS, so porting a virtual machine is difficult as compared to containers, and it also takes a lot of time to port a virtual machine because of its size.

For development purposes where the applications must be developed and tested in different platforms, Docker containers are the ideal choice.

**Performance**

the lightweight architecture of docker its less resource-intensive feature makes it a better choice than a virtual machine. As a result, of which containers can startup very fast compared to that of virtual machines

Virtual machines, there is no need to allocate resources permanently to containers. Scaling up and duplicating the containers is also an easy task compared to that of virtual machines, as there is no need to install an operating system in them.

**Understand docker in windows and docker in linux**

Docker’s popularity has been driven in large part by its promise of letting admins build an application once and then deploy it anywhere. But this promise can be misleading when you are trying to use Docker on Linux and Windows hosts at the same time.

Although Docker runs natively on both operating systems, the approach you to take to using Docker in both platforms isn’t the same. What’s more, the rationales for using containers don’t apply equally to both Linux and Windows.

**On Windows, Not All Versions Are Supported**

In most cases, Docker runs on any Linux system with a Linux kernel of 3.10 or later. That kernel version appeared in 2011, so most Linux distributions released since then work with Docker. And indeed, for Linux system admins, the fact that you can take a container running on one type of Linux system and move it to a different Linux server with minimal effort is a big part of Docker’s appeal.

The same can’t be said of the Windows world. Docker works only with Windows 10 desktop systems and Windows Server 2016 and 2019. There are other versions of Windows that Microsoft continues to support—and therefore may still be used in production—but don’t work with Docker.

For this reason, Docker loses some of its appeal in a Windows ecosystem, because it doesn’t let you deploy the same containerized application on any modern version of Windows.

**GUI Apps**

Docker was designed for containerizing applications that have a command-line interface. On both Windows and Linux, it lacks a native way of connecting to a graphical interface inside an application.

There are certainly ways of connecting to graphical interfaces using a protocol such as RDP or VNC. But they require extra effort to set up, and they could create additional security risks.

This problem applies to both Windows and Linux. However, given that it’s more common for Windows apps than Linux ones to have a graphical interface, this limitation is more significant in Windows environments. If you are running a Linux server, you probably know your way around a Bash terminal, and might not have a graphical front end installed on it at all; Windows admins, on contrast, are more accustomed to being able to administer their systems and applications graphically.

**Docker on Linux is ‘Pure-Play’**

In the Linux world, Docker is a (mostly) open source platform that works with any Linux distribution from any vendor. Docker didn’t work closely with any particular company in the Linux space to develop containers or make them work with Linux.

The same is not true of Windows. Docker and Microsoft worked closely to bring containers to Windows.

This difference might matter to you if you are thinking about the longevity or flexibility of containers on Windows. If Microsoft decides in the future to stop supporting containers, that will probably be the end of containers on Windows. Users are also beholden to Microsoft to decide which versions of Windows they can containers on (see above).

On Linux, these risks don’t exist. Even if Docker decided to cease development, the container ecosystem is now so dynamic and large that other open source projects would likely pick the slack and ensure that Linux containers remain viable.

**Docker file**

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

The following steps explain how you should go about creating a Docker File.

Step 1 − Create a file called Docker File and edit it using vim. Please note that the name of the file has to be "Dockerfile" with "D" as capital.



Step 2 − Build your Docker File using the following instructions.

#This is a sample Image

FROM ubuntu

MAINTAINER demousr@gmail.com

RUN apt-get update

RUN apt-get install –y nginx

CMD [“echo”,”Image created”]

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

* The first line "#This is a sample Image" is a comment. You can add comments to the Docker File with the help of the # command
* The next line has to start with the FROM keyword. It tells docker, from which base image you want to base your image from. In our example, we are creating an image from the ubuntu image.
* The next command is the person who is going to maintain this image. Here you specify the MAINTAINER keyword and just mention the email ID.
* The RUN command is used to run instructions against the image. In our case, we first update our Ubuntu system and then install the nginx server on our ubuntu image.
* The last command is used to display a message to the user.

Step 3 − Save the file.

