DOCKER

Docker Tutorial

Docker Tutorial is a series of blogs that will give you the conceptual & practical exposure to Docker – A new age containerization technology.

In this blog, I will focus on the below topics:

- What is the necessity behind Docker
- Introduction to Docker
- Virtualization vs Containerization

Docker's gaining popularity and its usage is spreading like wildfire. The reason for Docker's growing popularity is the extent to which it can be used in an IT organization. Very few tools out there have the functionality to find itself useful to both developers and as well as system administrators. Docker is one such tool that truly lives up to its promise of Build, Ship and Run.

In simple words, Docker is a software containerization platform, meaning you can build your application, package them along with their dependencies into a container and then these containers can be easily shipped to run on other machines.

For example: Lets consider a linux based application which has been written both in Ruby and Python. This application requires a specific version of linux, Ruby and Python. In order to avoid any version conflicts on user's end, a linux docker container can be created with the required versions of Ruby and Python installed along with the application. Now the end users can use the application easily by running this container without worrying about the dependencies or any version conflicts.

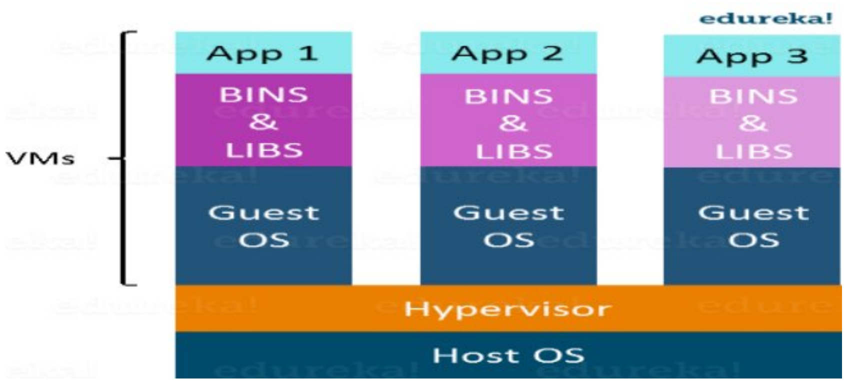
These containers uses Containerization which can be considered as an evolved version of Virtualization. The same task can also be achieved using Virtual Machines, however it is not very efficient.

I generally receive a question at this point, i.e. what is the difference between Virtualization and Containerization? These two terms are very similar to each other. So, let me first tell you What is Virtualization?

What is Virtualization?

Virtualization is the technique of importing a Guest operating system on top of a Host operating system. This technique was a revelation at the beginning because it allowed developers to run multiple operating systems in different virtual machines all running on the same host. This eliminated the need for extra hardware resource. The advantages of Virtual Machines or Virtualization are:

- Multiple operating systems can run on the same machine
- Maintenance and Recovery were easy in case of failure conditions
- Total cost of ownership was also less due to the reduced need for infrastructure



In the diagram on the right, you can see there is a host operating system on which there are 3 guest operating systems running which is nothing but the virtual machines.

As you know nothing is perfect, Virtualization also has some shortcomings. Running multiple Virtual Machines in the same host operating system leads to performance degradation. This is because of the guest OS running on top of the host OS, which will have its own kernel and set of libraries and dependencies. This takes up a large chunk of system resources, i.e. hard disk, processor and especially RAM.

Another problem with Virtual Machines which uses virtualization is that it takes almost a minute to boot-up. This is very critical in case of real-time applications.

Following are the disadvantages of Virtualization:

- Running multiple Virtual Machines leads to unstable performance
- Hypervisors are not as efficient as the host operating system
- Boot up process is long and takes time

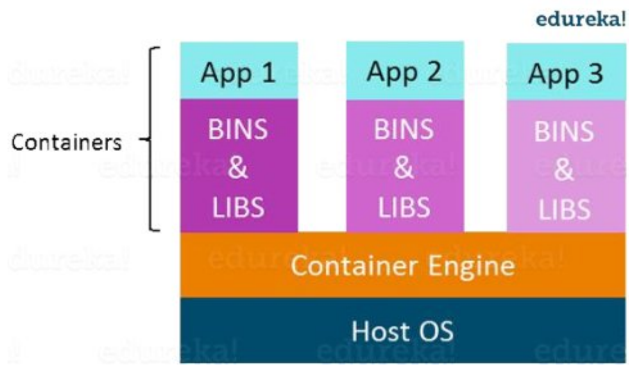
These drawbacks led to the emergence of a new technique called Containerization. Now let me tell you about Containerization.

What is Containerization?

Containerization is the technique of bringing virtualization to the operating system level. While Virtualization brings abstraction to the hardware, Containerization brings abstraction to the operating system. Do note that Containerization is also a type of Virtualization. Containerization is however more efficient because there is no guest OS here and utilizes a host's operating system, share relevant libraries & resources as and when needed unlike virtual machines. Application specific binaries and libraries of containers run on the host kernel, which makes processing and execution very fast. Even booting-up a container takes only a fraction of a second. Because all the containers share, host operating system and holds only the application related binaries & libraries. They are lightweight and faster than Virtual Machines.

Advantages of Containerization over Virtualization are:

- Containers on the same OS kernel are lighter and smaller
- Better resource utilization compared to VMs
- Boot-up process is short and takes few seconds



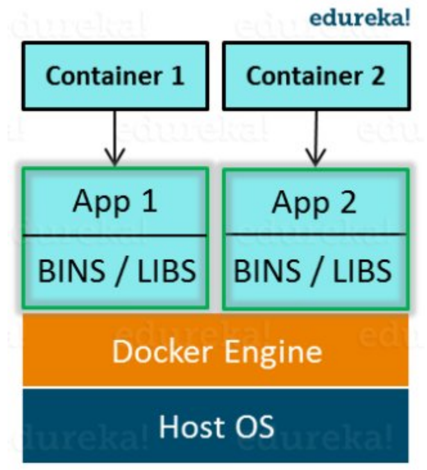
In the diagram on the right, you can see that there is a host operating system which is shared by all the containers. Containers only contain application specific libraries which are separate for each container and they are faster and do not waste any resources.

All these containers are handled by the containerization layer which is not native to the host operating system. Hence a software is needed, which can enable you to create & run containers on your host operating system.

Check out this Docker tutorial video for deep understanding of Docker.

Introduction To Docker

Docker is a containerization platform that packages your application and all its dependencies together in the form of Containers to ensure that your application works seamlessly in any environment.



As you can see in the diagram on the right, each application will run on a separate container and will have its own set of libraries and dependencies. This also ensures that there is process level isolation, meaning each application is independent of other applications, giving developers surety that they can build applications that will not interfere with one another.

As a developer, I can build a container which has different applications installed on it and give it to my QA team who will only need to run the container to replicate the developer environment.

Docker Interview Questions

Q1. What is Docker?

I will suggest you to start with a small definition of Docker.

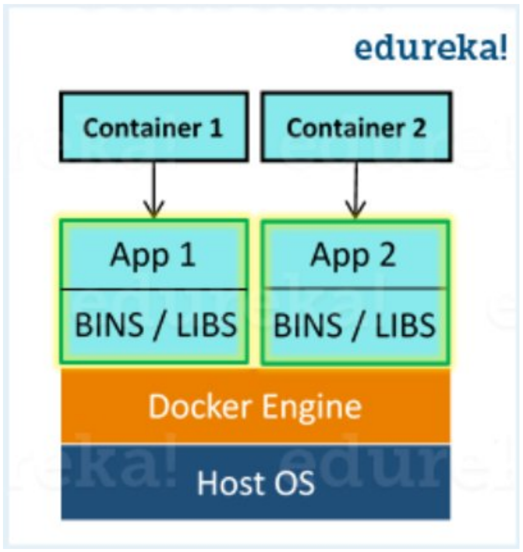
Docker is a containerization platform which packages your application and all its dependencies together in the form of containers so as to ensure that your application works seamlessly in any environment be it development or test or production.

Now you should explain Docker containers.

Docker containers, wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries etc. anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment.

You can refer the diagram shown below, as you can see that containers run on a single machine share the same operating system kernel, they start instantly as only apps need to start as the kernel is already running and uses less RAM.

Note: Unlike Virtual Machines which has its own OS Docker containers uses

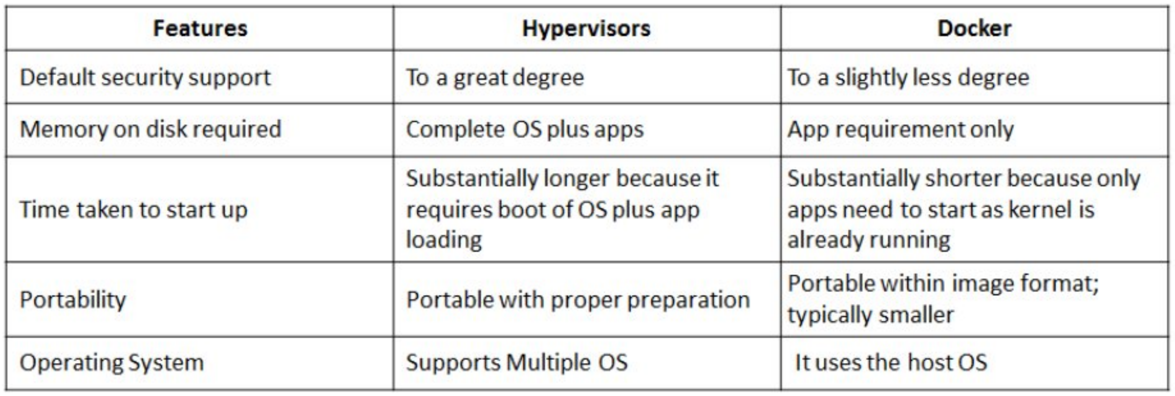


the host OS

As you have mentioned about Virtual Machines in your previous answer so the next question in this Docker Interview Questions blog will be related to the differences between the two.

Q2. What are the differences between Docker and Hypervisors?

You can refer the below differences:



Features	Hypervisors	Docker
Default security support	To a great degree	To a slightly less degree
Memory on disk required	Complete OS plus apps	App requirement only
Time taken to start up	Substantially longer because it requires boot of OS plus app loading	Substantially shorter because only apps need to start as kernel is already running
Portability	Portable with proper preparation	Portable within image format; typically smaller
Operating System	Supports Multiple OS	It uses the host OS

Next set of Docker interview questions will focus on various components of Docker.

Q3. What is Docker image?

I will suggest you to go with the below mentioned flow:

Docker image is the source of Docker container. In other words, Docker images are used to create containers. Images are created with the build command, and they'll produce a container when started with run. Images are stored in a Docker registry such as registry.hub.docker.com because they can become quite large, images are designed to be composed of layers of other images, allowing a minimal amount of data to be sent when transferring images over the network.
Tip: Be aware of Dockerhub in order to answer questions on pre-available images.

Q4. What is Docker container?

This is a very important question so just make sure you don't deviate from the topic and I will advise you to follow the below mentioned format:

Docker containers include the application and all of its dependencies, but share the kernel with other containers, running as isolated processes in user space on the host operating system. Docker containers are not tied to any specific infrastructure: they run on any computer, on any infrastructure, and in any cloud.
Now explain how to create a Docker container, Docker containers can be created by either creating a Docker image and then running it or you can use Docker images that are present on the Dockerhub.

Docker containers are basically runtime instances of Docker images.

Q5 What is Docker hub?

Answer to this question is pretty direct.

Docker hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker cloud so you can deploy images to your hosts. It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.

Q6. How is Docker different from other container technologies?

According to me, below, points should be there in your answer:

Docker containers are easy to deploy in a cloud. It can get more applications running on the same hardware than other technologies, it makes it easy for developers to quickly create, ready-to-run containerized applications and it makes managing and deploying applications much easier. You can even share containers with your applications.
If you have some more points to add you can do that but make sure the above the above explanation is there in your answer.

Q7. What is Docker Swarm?

You should start this answer by explaining Docker Swarn.

Docker Swarm is native clustering for Docker. It turns a pool of Docker hosts into a single, virtual Docker host. Docker Swarm serves the standard Docker API, any tool that already communicates with a Docker daemon can use Swarm to transparently scale to multiple hosts.

I will also suggest you to include some supported tools:

- Dokku
- Docker Compose
- Docker Machine
- Jenkins

Q8. What is Dockerfile used for?

This answer, according to me should begin by explaining the use of Dockerfile.

Docker can build images automatically by reading the instructions from a Dockerfile.

Now I will suggest you to give a small definition of Dockerfle.

A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image. Using docker build users can create an automated build that executes several command-line instructions in succession.

Now, the next set of Docker interview questions will test your experience with Docker.

Q9. Can I use json instead of yaml for my compose file in Docker?

You can use json instead of yaml for your compose file, to use json file with compose, specify the filename to use for eg:

```
docker-compose -f docker-compose.json up
```

Q10. Tell us how you have used Docker in your past position?

Explain how you have used Docker to help rapid deployment. Explain how you have scripted Docker and used Docker with other tools like Puppet, Chef or Jenkins.

If you have no past practical experience in Docker and have past experience with other tools in a similar space, be honest and explain the same. In this case, it makes sense if you can compare other tools to Docker in terms of functionality.

Q11. How to create Docker container?

I will suggest you to give a direct answer to this.

We can use Docker image to create Docker container by using the below command:

```
docker run -t -i command name
```

This command will create and start a container.

You should also add, If you want to check the list of all running container with the status on a host use the below command:

```
docker ps -a
```

Q12. How to stop and restart the Docker container?

In order to stop the Docker container you can use the below command:

```
docker stop container ID
```

Now to restart the Docker container you can use:

```
docker restart container ID
```

Q13 How far do Docker containers scale?

Large web deployments like Google and Twitter, and platform providers such as Heroku and dotCloud all run on container technology, at a scale of hundreds of thousands or even millions of containers running in parallel.

Q14. What platforms does Docker run on?

I will start this answer by saying Docker runs on only Linux and Cloud platforms and then I will mention the below vendors of Linux:

- Ubuntu 12.04, 13.04 et al
- Fedora 19/20+
- RHEL 6.5+
- CentOS 6+
- Gentoo
- ArchLinux
- openSUSE 12.3+
- CRUX 3.0+

Cloud:

- Amazon EC2
- Google Compute Engine
- Microsoft Azure
- Rackspace

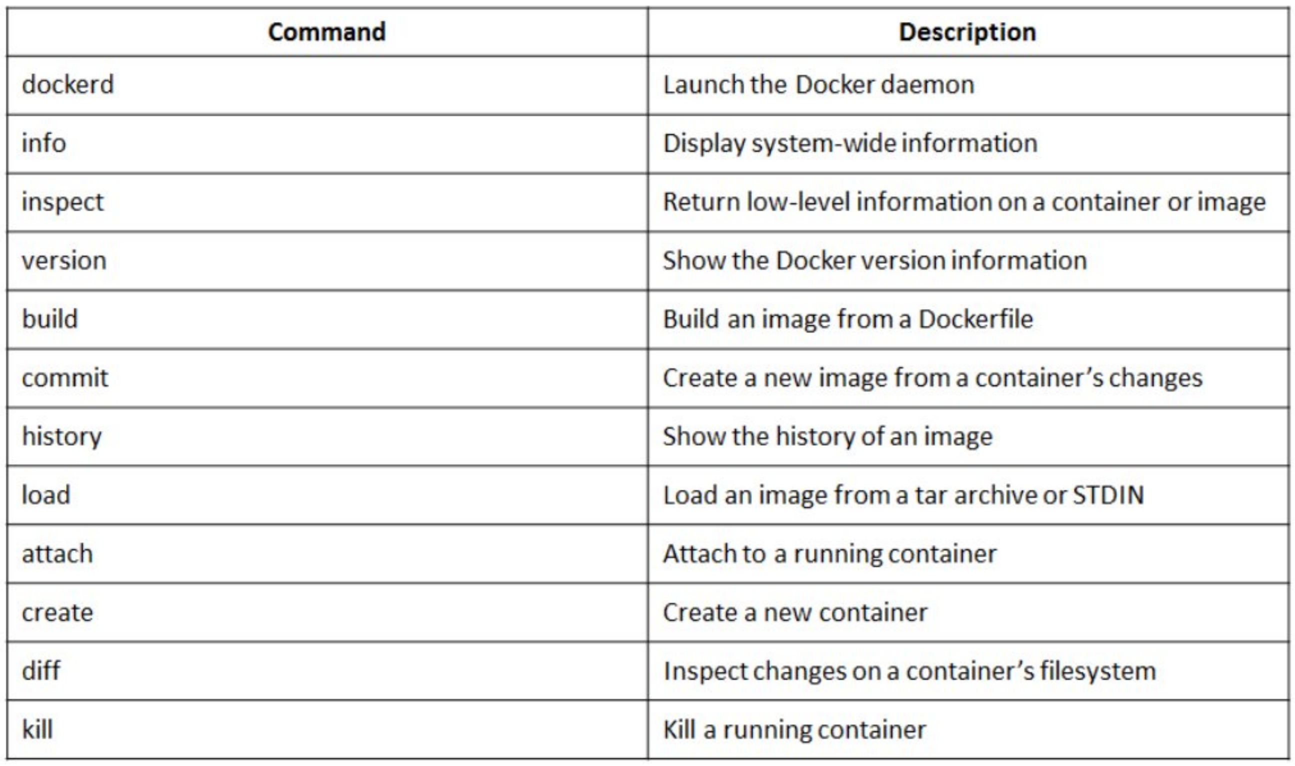
Note that Docker does not run on Windows or Mac.

Q15. Do I lose my data when the Docker container exits?

You can answer this by saying, no I won’t lose my data when Docker container exits, any data that your application writes to disk gets preserved in its container until you explicitly delete the container. The file system for the container persists even after the container halts.

Q16. Mention some commonly used Docker command?

Below are some commonly used Docker commands:



Command	Description
dockerd	Launch the Docker daemon
info	Display system-wide information
inspect	Return low-level information on a container or image
version	Show the Docker version information
build	Build an image from a Dockerfile
commit	Create a new image from a container’s changes
history	Show the history of an image
load	Load an image from a tar archive or STDIN
attach	Attach to a running container
create	Create a new container
diff	Inspect changes on a container’s filesystem
kill	Kill a running container