### Q1: What is the need for DevOps?

Topic: DevOps  
Difficulty: ⭐

Nowadays instead of releasing big sets of features, companies are trying to see if small features can be transported to their customers through a series of release trains. This has many advantages like quick feedback from customers, better quality of software etc. which in turn leads to high customer satisfaction. To achieve this, companies are required to:

1. Increase deployment frequency
2. Lower failure rate of new releases
3. Shortened lead time between fixes
4. Faster mean time to recovery in the event of new release crashing

DevOps fulfills all these requirements and helps in achieving seamless software delivery.

🔗 Source: [edureka.co](https://www.edureka.co/blog/interview-questions/top-devops-interview-questions-2016/)

### Q2: What is Docker?

Topic: Docker  
Difficulty: ⭐

* 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.
* 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.

🔗 Source: [edureka.co](https://www.edureka.co/blog/interview-questions/docker-interview-questions/)

### Q3: What are the advantages of DevOps?

Topic: DevOps  
Difficulty: ⭐⭐

Technical benefits:

* Continuous software delivery
* Less complex problems to fix
* Faster resolution of problems

Business benefits:

* Faster delivery of features
* More stable operating environments
* More time available to add value (rather than fix/maintain)

🔗 Source: [edureka.co](https://www.edureka.co/blog/interview-questions/top-devops-interview-questions-2016/)

### Q4: What is the function of CI (Continuous Integration) server?

Topic: DevOps  
Difficulty: ⭐⭐

CI server function is to continuously integrate all changes being made and committed to repository by different developers and check for compile errors. It needs to build code several times a day, preferably after every commit so it can detect which commit made the breakage if the breakage happens.

🔗 Source: [linoxide.com](https://linoxide.com/linux-how-to/devops-interview-questions-answers/)

### Q5: How to build envrionment-agnostic systems with Docker?

Topic: Docker  
Difficulty: ⭐⭐

There are three main features helping to achieve that:

* Volumes
* Environment variable injection
* Read-only file systems

🔗 Source: [rafalgolarz.com](http://rafalgolarz.com/blog/2017/03/11/docker_interview/)

### Q6: What is the difference between the COPY and ADDcommands in a Dockerfile?

Topic: Docker  
Difficulty: ⭐⭐

Although ADD and COPY are functionally similar, generally speaking, COPY is preferred.

That’s because it’s more transparent than ADD. COPY only supports the basic copying of local files into the container, while ADD has some features (like local-only tar extraction and remote URL support) that are not immediately obvious. Consequently, the best use for ADD is local tar file auto-extraction into the image, as in ADD rootfs.tar.xz /.

🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/16647069/should-i-use-vagrant-or-docker-for-creating-an-isolated-environment)

### Q7: What is Docker image?

Topic: Docker  
Difficulty: ⭐⭐

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.

🔗 Source: [edureka.co](https://www.edureka.co/blog/interview-questions/docker-interview-questions/)

### Q8: What is Docker container?

Topic: Docker  
Difficulty: ⭐⭐

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.

🔗 Source: [edureka.co](https://www.edureka.co/blog/interview-questions/docker-interview-questions/)

### Q9: What is Docker hub?

Topic: Docker  
Difficulty: ⭐⭐

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.

🔗 Source: [edureka.co](https://www.edureka.co/blog/interview-questions/docker-interview-questions/)

### Q10: What are the various states that a Docker container can be in at any given point in time?

Topic: Docker  
Difficulty: ⭐⭐

There are four states that a Docker container can be in, at any given point in time. Those states are as given as follows:

* Running
* Paused
* Restarting
* Exited

🔗 Source: [mindmajix.com](https://mindmajix.com/docker-interview-questions)

### Q11: Is there a way to identify the status of a Docker container?

Topic: Docker  
Difficulty: ⭐⭐

We can identify the status of a Docker container by running the command

docker ps –a

which will in turn list down all the available docker containers with its corresponding statuses on the host. From there we can easily identify the container of interest to check its status correspondingly.

🔗 Source: [mindmajix.com](https://mindmajix.com/docker-interview-questions)

### Q12: What are the most common instructions in Dockerfile?

Topic: Docker  
Difficulty: ⭐⭐

Some of the common instructions in Dockerfile are as follows:

* FROM: We use FROM to set the base image for subsequent instructions. In every valid Dockerfile, FROM is the first instruction.
* LABEL: We use LABEL to organize our images as per project, module, licensing etc. We can also use LABEL to help in automation. In LABEL we specify a key value pair that can be later used for programmatically handling the Dockerfile.
* RUN: We use RUN command to execute any instructions in a new layer on top of the current image. With each RUN command we add something on top of the image and use it in subsequent steps in Dockerfile.
* CMD: We use CMD command to provide default values of an executing container. In a Dockerfile, if we include multiple CMD commands, then only the last instruction is used.

🔗 Source: [knowledgepowerhouse.com](https://www.knowledgepowerhouse.com/what-are-the-most-common-instructions-in-dockerfile/1224)

### Q13: What type of applications - Stateless or Stateful are more suitable for Docker Container?

Topic: Docker  
Difficulty: ⭐⭐

It is preferable to create Stateless application for Docker Container. We can create a container out of our application and take out the configurable state parameters from application. Now we can run same container in Production as well as QA environments with different parameters. This helps in reusing the same Image in different scenarios. Also a stateless application is much easier to scale with Docker Containers than a stateful application.

🔗 Source: [mindmajix.com](https://www.knowledgepowerhouse.com/what-type-of-applications-stateless-or-stateful-are-more-suitable-for-docker-container/1208)

### Q14: Explain basic Docker usage workflow

Topic: Docker  
Difficulty: ⭐⭐⭐

1. Everything starts with the Dockerfile. The Dockerfile is the source code of the Image.
2. Once the Dockerfile is created, you build it to create the image of the container. The image is just the "compiled version" of the "source code" which is the Dockerfile.
3. Once you have the image of the container, you should redistribute it using the registry. The registry is like a git repository -- you can push and pull images.
4. Next, you can use the image to run containers. A running container is very similar, in many aspects, to a virtual machine (but without the hypervisor).

+------------+ docker build +--------------+ docker run -dt +-----------+ docker exec -it +------+

| Dockerfile | --------------> | Image | ---------------> | Container | -----------------> | Bash |

+------------+ +--------------+ +-----------+ +------+

^

| docker pull

|

+--------------+

| Registry |

+--------------+

🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/23735149/what-is-the-difference-between-a-docker-image-and-a-container/23736802" \l "23736802)

### Q15: What is the difference between Docker Image and Layer?

Topic: Docker  
Difficulty: ⭐⭐⭐

* Image: A Docker image is built up from a series of read-only layers
* Layer: Each layer represents an instruction in the image’s Dockerfile.

The below Dockerfile contains four commands, each of which creates a layer.

FROM ubuntu:15.04

COPY . /app

RUN make /app

CMD python /app/app.py

Importantly, each layer is only a set of differences from the layer before it.

🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/23735149/what-is-the-difference-between-a-docker-image-and-a-container/23736802" \l "23736802)

### Q16: What is virtualisation?

Topic: Docker  
Difficulty: ⭐⭐⭐

In its conceived form, virtualisation was considered a method of logically dividing mainframes to allow multiple applications to run simultaneously. However, the scenario drastically changed when companies and open source communities were able to provide a method of handling the privileged instructions in one way or another and allow for multiple operating systems to be run simultaneously on a single x86 based system.

The net effect is that virtualization allows you to run two completely different OS on same hardware. Each guest OS goes through all the process of bootstrapping, loading kernel etc. You can have very tight security, for example, guest OS can't get full access to host OS or other guests and mess things up.

The virtualization method can be categorized based on how it mimics hardware to a guest operating system and emulates guest operating environment. Primarily, there are three types of virtualization:

* Emulation
* Paravirtualization
* Container-based virtualization

🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine)

### Q17: What is Hypervisor?

Topic: Docker  
Difficulty: ⭐⭐⭐

The hypervisor handles creating the virtual environment on which the guest virtual machines operate. It supervises the guest systems and makes sure that resources are allocated to the guests as necessary. The hypervisor sits in between the physical machine and virtual machines and provides virtualization services to the virtual machines. To realize it, it intercepts the guest operating system operations on the virtual machines and emulates the operation on the host machine's operating system.

The rapid development of virtualization technologies, primarily in cloud, has driven the use of virtualization further by allowing multiple virtual servers to be created on a single physical server with the help of hypervisors, such as Xen, VMware Player, KVM, etc., and incorporation of hardware support in commodity processors, such as Intel VT and AMD-V.

🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine)

### Q18: What is Docker Swarm?

Topic: Docker  
Difficulty: ⭐⭐⭐

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.

🔗 Source: [edureka.co](https://www.edureka.co/blog/interview-questions/docker-interview-questions/)

### Q19: How will you monitor Docker in production?

Topic: Docker  
Difficulty: ⭐⭐⭐

Docker provides tools like docker stats and docker events to monitor Docker in production. We can get reports on important statistics with these commands.

* Docker stats: When we call docker stats with a container id, we get the CPU, memory usage etc of a container. It is similar to top command in Linux.
* Docker events: Docker events are a command to see the stream of activities that are going on in Docker daemon.

Some of the common Docker events are: attach, commit, die, detach, rename, destroy etc. We can also use various options to limit or filter the events that we are interested in.

🔗 Source: [knowledgepowerhouse.com](https://www.knowledgepowerhouse.com/what-are-the-different-kinds-of-namespaces-available-in-a-container/1228)

### Q20: What is an orphant volume and how to remove it?

Topic: Docker  
Difficulty: ⭐⭐⭐⭐

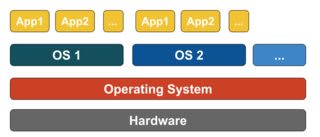
An orphant volume is a volume without any containers attached to it. Prior Docker v. 1.9 it was very problematic to remove it.

🔗 Source: [rafalgolarz.com](http://rafalgolarz.com/blog/2017/03/11/docker_interview/)

### Q21: What is Paravirtualization?

Topic: Docker  
Difficulty: ⭐⭐⭐⭐

Paravirtualization, also known as Type 1 hypervisor, runs directly on the hardware, or “bare-metal”, and provides virtualization services directly to the virtual machines running on it. It helps the operating system, the virtualized hardware, and the real hardware to collaborate to achieve optimal performance. These hypervisors typically have a rather small footprint and do not, themselves, require extensive resources.



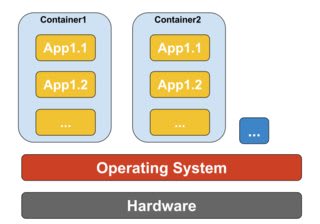
Examples in this category include Xen, KVM, etc.

🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine)

### Q22: How is Docker different from a virtual machine?

Topic: Docker  
Difficulty: ⭐⭐⭐⭐

Docker isn't a virtualization methodology. It relies on other tools that actually implement container-based virtualization or operating system level virtualization. For that, Docker was initially using LXC driver, then moved to libcontainerwhich is now renamed asrunc. Docker primarily focuses on automating the deployment of applications inside application containers. Application containers are designed to package and run a single service, whereas system containers are designed to run multiple processes, like virtual machines. So, Docker is considered as a container management or application deployment tool on containerized systems.



* Unlike a virtual machine, a container does not need to boot the operating system kernel, so containers can be created in less than a second. This feature makes container-based virtualization unique and desirable than other virtualization approaches.
* Since container-based virtualization adds little or no overhead to the host machine, container-based virtualization has near-native performance
* For container-based virtualization, no additional software is required, unlike other virtualizations.
* All containers on a host machine share the scheduler of the host machine saving need of extra resources.
* Container states (Docker or LXC images) are small in size compared to virtual machine images, so container images are easy to distribute.
* Resource management in containers is achieved through cgroups. Cgroups does not allow containers to consume more resources than allocated to them. However, as of now, all resources of host machine are visible in virtual machines, but can't be used. This can be realized by running top or htop on containers and host machine at the same time. The output across all environments will look similar.

🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine)

### Q23: Can you explain dockerfile ONBUILD instruction?

Topic: Docker  
Difficulty: ⭐⭐⭐⭐

The ONBUILD instruction adds to the image a trigger instruction to be executed at a later time, when the image is used as the base for another build. This is useful if you are building an image which will be used as a base to build other images, for example an application build environment or a daemon which may be customized with user-specific configuration.

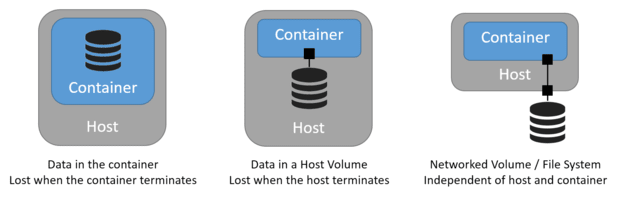
🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/34863114/dockerfile-onbuild-instruction)

### Q24: Is it good practice to run stateful applications on Docker? What are the scenarios where Docker best fits in?

Topic: Docker  
Difficulty: ⭐⭐⭐⭐

he problem with statefull docker aplications is that they by default store their state (data) in the containers filesystem. Once you update your software version or want to move to another machine its hard to retrieve the data from there.

What you need to do is bind a volume to the container and store any data in the volume.



if you run your container with: docker run -v hostFolder:/containerfolderany changes to /containerfolder will be persisted on the hostfolder. Something similar can be done with a nfs drive. Then you can run you application on any host machine and the state will be saved in the nfs drive.

🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/42997550/stateful-applications-on-docker)

### Q25: Can you run Docker containers natively on Windows?

Topic: Docker  
Difficulty: ⭐⭐⭐⭐

With Windows Server 2016 you can run Docker containers natively on Windows, and with Windows Nano Server you’ll have a lightweight OS to run inside containers, so you can run .NET apps on their native platform.

🔗 Source: [rafalgolarz.com](http://rafalgolarz.com/blog/2017/03/11/docker_interview/)

### Q26: How does Docker run containers in non-Linux systems?

Topic: Docker  
Difficulty: ⭐⭐⭐⭐⭐

The concept of a container is made possible by the namespaces feature added to Linux kernel version 2.6.24. The container adds its ID to every process and adding new access control checks to every system call. It is accessed by the clone() system call that allows creating separate instances of previously-global namespaces.

If containers are possible because of the features available in the Linux kernel, then the obvious question is that how do non-Linux systems run containers. Both Docker for Mac and Windows use Linux VMs to run the containers. Docker Toolbox used to run containers in Virtual Box VMs. But, the latest Docker uses Hyper-V in Windows and Hypervisor.framework in Mac.

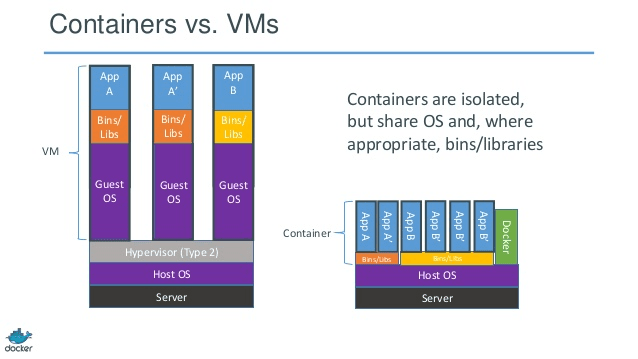
🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine)

### Q27: How containers works at low level?

Topic: Docker  
Difficulty: ⭐⭐⭐⭐⭐

Around 2006, people including some of the employees at Google implemented new Linux kernel level feature called namespaces (however the idea long before existed in FreeBSD). One function of the OS is to allow sharing of global resources like network and disk to processes. What if these global resources were wrapped in namespaces so that they are visible only to those processes that run in the same namespace? Say, you can get a chunk of disk and put that in namespace X and then processes running in namespace Y can't see or access it. Similarly, processes in namespace X can't access anything in memory that is allocated to namespace Y. Of course, processes in X can't see or talk to processes in namespace Y. This provides kind of virtualization and isolation for global resources.

This is how Docker works: Each container runs in its own namespace but uses exactly the same kernel as all other containers. The isolation happens because kernel knows the namespace that was assigned to the process and during API calls it makes sure that process can only access resources in its own namespace.



🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine)

### Q28: Name some limitations of containers vs VM

Topic: Docker  
Difficulty: ⭐⭐⭐⭐⭐

Just to name a few:

* You can't run completely different OS in containers like in VMs. However you can run different distros of Linux because they do share the same kernel. The isolation level is not as strong as in VM. In fact, there was a way for "guest" container to take over host in early implementations.
* Also you can see that when you load new container, the entire new copy of OS doesn't start like it does in VM.
* All containers share the same kernel. This is why containers are light weight.
* Also unlike VM, you don't have to pre-allocate significant chunk of memory to containers because we are not running new copy of OS. This enables to run thousands of containers on one OS while sandboxing them which might not be possible to do if we were running separate copy of OS in its own VM.

🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/16047306/how-is-docker-different-from-a-virtual-machine)

### Q29: How to use Docker with multiple environments?

Topic: Docker  
Difficulty: ⭐⭐⭐⭐⭐

You’ll almost certainly want to make changes to your app configuration that are more appropriate to a live environment. These changes may include:

* Removing any volume bindings for application code, so that code stays inside the container and can’t be changed from outside
* Binding to different ports on the host
* Setting environment variables differently (e.g., to decrease the verbosity of logging, or to enable email sending)
* Specifying a restart policy (e.g., restart: always) to avoid downtime
* Adding extra services (e.g., a log aggregator)

For this reason, you’ll probably want to define an additional Compose file, say production.yml, which specifies production-appropriate configuration. This configuration file only needs to include the changes you’d like to make from the original Compose file.

docker-compose -f docker-com

🔗 Source: [stackoverflow.com](https://stackoverflow.com/questions/42287426/docker-multiple-environments)

### Q30: Why Docker compose does not wait for a container to be ready before moving on to start next service in dependency order?

Topic: Docker  
Difficulty: ⭐⭐⭐⭐⭐

Compose always starts and stops containers in dependency order, where dependencies are determined by depends\_on, links, volumes\_from, and network\_mode: "service:...".

However, for startup Compose does not wait until a container is “ready” (whatever that means for your particular application) - only until it’s running. There’s a good reason for this:

* The problem of waiting for a database (for example) to be ready is really just a subset of a much larger problem of distributed systems. In production, your database could become unavailable or move hosts at any time. Your application needs to be resilient to these types of failures.
* To handle this, design your application to attempt to re-establish a connection to the database after a failure. If the application retries the connection, it can eventually connect to the database.
* The best solution is to perform this check in your application code, both at startup and whenever a connection is lost for any reason.