Docker Basic Questions

This category of Docker Interview Questions consists of questions that you're expected to know. These are the most basic questions. An interviewer will start with these and eventually increase the difficulty level. Let's have a look at them.

1. What is Hypervisor?

A hypervisor is a software that makes virtualization possible. It is also called Virtual Machine Monitor. It divides the host system and allocates the resources to each divided virtual environment. You can basically have multiple OS on a single host system. There are two types of Hypervisors:

- Type 1: It's also called Native Hypervisor or Bare metal Hypervisor. It runs directly on the underlying host system. It has direct access to your host's system hardware and hence does not require a base server operating system.
- Type 2: This kind of hypervisor makes use of the underlying host operating system. It's also called Hosted Hypervisor.

2. What is virtualization?

Virtualization is the process of creating a software-based, virtual version of something(compute storage, servers, application, etc.). These virtual versions or environments are created from a single physical hardware system. Virtualization lets you split one system into many different sections which act like separate, distinct individual systems. A software called Hypervisor makes this kind of splitting possible. The virtual environment created by the hypervisor is called Virtual Machine.

3. What is containerization?

Let me explain this is with an example. Usually, in the software development process, code developed on one machine might not work perfectly fine on any other machine because of the dependencies. This problem was solved by the containerization concept. So basically, an application that is being developed and deployed is bundled and wrapped together with all its configuration files and dependencies. This bundle is called a container. Now when you wish to run the application on another system, the container is deployed which will give a bug-free environment as all the dependencies and libraries are wrapped together. Most famous containerization environments are Docker and Kubernetes.

4. Difference between virtualization and containerization

Once you've explained containerization and virtualization, the next expected question would be differences. The question could either be differences between virtualization and containerization or differences between virtual machines and containers. Either way, this is how you respond.

Containers provide an isolated environment for running the application. The entire user space is explicitly dedicated to the application. Any changes made inside the container is never reflected on the host or even other containers running on the same host. Containers are an abstraction of the application layer. Each container is a different application.

Whereas in Virtualization, hypervisors provide an entire virtual machine to the guest(including Kernal). Virtual machines are an abstraction of the hardware layer. Each VM is a physical machine.

5. What is Docker?

Since its a Docker interview, there will be an obvious question about what is Docker. Start with a small definition.

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, 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. It wraps basically anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment.

6. What is a Docker Container?

Docker containers include the application and all of its dependencies. It shares 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. Docker containers are basically runtime instances of Docker images.

7. What are Docker Images?

When you mention Docker images, your very next question will be "what are Docker images".

Docker image is the source of Docker container. In other words, Docker images are used to create containers. When a user runs a Docker image, an instance of a container is created. These docker images can be deployed to any Docker environment.

8. What is Docker Hub?

Docker images create docker containers. There has to be a registry where these docker images live. This registry is Docker Hub. Users can pick up images from Docker Hub and use them to create customized images and containers. Currently, the Docker Hub is the world's largest public repository of image containers.

9. Explain Docker Architecture?

Docker Architecture consists of a Docker Engine which is a client-server application with three major components:

1. A server which is a type of long-running program called a daemon process (the docker command).
2. A REST API which specifies interfaces that programs can use to talk to the daemon and instruct it what to do.
3. A command line interface (CLI) client (the docker command).
4. The CLI uses the Docker REST API to control or interact with the Docker daemon through scripting or direct CLI commands. Many other Docker applications use the underlying API and CLI.

Refer to this blog, to read more about **Docker Architecture**.

10. What is a Dockerfile?

Let's start by giving a small explanation of Dockerfile and proceed by giving examples and commands to support your arguments.

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

The interviewer does not just expect definitions, hence explain how to use a Dockerfile which comes with experience. Have a look at this tutorial to understand how Dockerfile works.

11. Tell us something about Docker Compose.

Docker Compose is a YAML file which contains details about the services, networks, and volumes for setting up the Docker application. So, you can use Docker Compose to create separate containers, host them and get them to communicate with each other. Each container will expose a port for communicating with other containers.

12. What is Docker Swarm?

You are expected to have worked with Docker Swarm as it's an important concept of Docker.

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.

13. What is a Docker Namespace?

A namespace is one of the Linux features and an important concept of containers. Namespace adds a layer of isolation in containers. Docker provides various namespaces in order to stay portable and not affect the underlying host system. Few namespace types supported by Docker – PID, Mount, IPC, User, Network

14. What is the lifecycle of a Docker Container?

This is one of the most popular questions asked in Docker interviews. Docker containers have the following lifecycle:

- Create a container
- Run the container
- Pause the container(optional)
- Un-pause the container(optional)
- Start the container
- Stop the container
- Restart the container
- Kill the container
- Destroy the container

15. What is Docker Machine?

Docker machine is a tool that lets you install Docker Engine on virtual hosts. These hosts can now be managed using the docker-machine commands. Docker machine also lets you provision Docker Swarm Clusters.

Docker Basic Commands

Once you've aced the basic conceptual questions, the interviewer will increase the difficulty level. So let's move on to the next section of this Docker Interview Questions article. This section talks about the commands that are very common amongst docker users.

16. How to check for Docker Client and Docker Server version?

The following command gives you information about Docker Client and Server versions:

```
$ docker version
```

17. How do you get the number of containers running, paused and stopped?

You can use the following command to get detailed information about the docker installed on your system.

```
$ docker info
```

DevOps Certification Training

Instructor-led Sessions

Real-life Case Studies

Assignments

Lifetime Access

Explore Curriculum

You can get the number of containers running, paused, stopped, the number of images and a lot more.

18. If you vaguely remember the command and you'd like to confirm it, how will you get help on that particular command?

The following command is very useful as it gives you help on how to use a command, the syntax, etc.

```
$ docker --help
```

The above command lists all Docker commands. If you need help with one specific command, you can use the following syntax:

```
$ docker <command> --help
```

19. How to login into docker repository?

You can use the following command to login into hub.docker.com:

```
$ docker login
```

You'll be prompted for your username and password, insert those and congratulations, you're logged in.

20. If you wish to use a base image and make modifications or personalize it, how do you do that?

You pull an image from docker hub onto your local system

It's one simple command to pull an image from docker hub:

```
$ docker pull <image_name>
```

21. How do you create a docker container from an image?

Pull an image from docker repository with the above command and run it to create a container. Use the following command:

```
$ docker run -it -d <image_name>
```

Most probably the next question would be, what does the '-d' flag mean in the command?

-d means the container needs to start in the detached mode. Explain a little about the detach mode. Have a look at this blog to get a better understanding of different docker commands.

22. How do you list all the running containers?

The following command lists down all the running containers:

```
$ docker ps
```

23. Suppose you have 3 containers running and out of these, you wish to access one of them. How do you access a running container?

The following command lets us access a running container:

```
$ docker exec -it <container id> bash
```

The exec command lets you get inside a container and work with it.

24. How to start, stop and kill a container?

The following command is used to start a docker container:

```
$ docker start <container_id>
```

and the following for stopping a running container:

```
$ docker stop <container_id>
```

kill a container with the following command:

```
$ docker kill <container_id>
```

25. Can you use a container, edit it, and update it? Also, how do you make it a new and store it on the local system?

Of course, you can use a container, edit it and update it. This sounds complicated but its actually just one command.

```
$ docker commit <conatainer id> <username/imagename>
```

26. Once you've worked with an image, how do you push it to docker hub?

```
$ docker push <username/image name>
```

27. How to delete a stopped container?

Use the following command to delete a stopped container:

```
$ docker rm <container id>
```


evOps Training

DEVOPS CERTIFICATION TRAINING

DevOps Certification Training

Reviews
5(59529)

AWS CERTIFIED DEVOPS ENGINEER TRAINING

AWS Certified DevOps Engineer Training

Reviews
5(2030)

DOCKER TRAINING AND CERTIFICATION

Docker Training and Certification

Reviews
5(4300)

CONTINUOUS INTEGRATION WITH JENKINS CERTIFICATION TRAINING

Continuous Integration with Jenkins Certification Training

Reviews
5(7088)

28. How to delete an image from the local storage system?

The following command lets you delete an image from the local system:

```
$ docker rmi <image-id>
```

29. How to build a Dockerfile?

Once you've written a Dockerfile, you need to build it to create an image with those specifications. Use the following command to build a Dockerfile:

```
$ docker build <path to docker file>
```

The next question would be when do you use ".dockerfile_name" and when to use the entire path?

Use ".dockerfile_name" when the dockerfile exits in the same file directory and you use the entire path if it lives somewhere else.

30. Do you know why *docker system prune* is used? What does it do?

```
$ docker system prune
```

The above command is used to remove all the stopped containers, all the networks that are not used, all dangling images and all build caches. It's one of the most useful docker commands.

Docker Advanced Questions

Once the interviewer knows that you're familiar with the Docker commands, he/she will start asking about practical applications This section of Docker Interview Questions consists of questions that you'll only be able to answer when you've gained some experience working with Docker.

31. Will you lose your data, when a docker container exists?

No, you won't lose any data when Docker container exits. Any data that your application writes to the container gets preserved on the disk until you explicitly delete the container. The file system for the container persists even after the container halts.

32. Where all do you think Docker is being used?

When asked such a question, respond by talking about applications of Docker. Docker is being used in the following areas:

- Simplifying configuration: Docker lets you put your environment and configuration into code and deploy it.
- Code Pipeline Management: There are different systems used for development and production. As the code travels from development to testing to production, it goes through a difference in the environment. Docker helps in maintaining the code pipeline consistency.

- Developer Productivity: Using Docker for development gives us two things – We're closer to production and development environment is built faster.
- Application Isolation: As containers are applications wrapped together with all dependencies, your apps are isolated. They can work by themselves on any hardware that supports Docker.
- Debugging Capabilities: Docker supports various debugging tools that are not specific to containers but work well with containers.
- Multi-tenancy: Docker lets you have multi-tenant applications avoiding redundancy in your codes and deployments.
- Rapid Deployment: Docker eliminates the need to boost an entire OS from scratch, reducing the deployment time.

33. How is Docker different from other containerization methods?

Docker containers are very easy to deploy in any cloud platform. It can get more applications running on the same hardware when compared to 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 explanation is there in your answer.

34. 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 JSON filename to use, for eg:

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

35. How have you used Docker in your previous position?

Explain how you have used Docker to help rapid deployment. Explain how you have scripted Docker and used it with other tools like Puppet, Chef or Jenkins. If you have no past practical experience in Docker and instead have 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.

36. How far do Docker containers scale? Are there any requirements for the same?

Large web deployments like Google and Twitter and platform providers such as Heroku and dotCloud, all run on container technology. Containers can be scaled to hundreds of thousands or even millions of them running in parallel. Talking about requirements, containers require the memory and the OS at all the times and a way to use this memory efficiently when scaled.

37. What platforms does docker run on?

This is a very straightforward question but can get tricky. Do some company research before going for the interview and find out how the company is using Docker. Make sure you mention the platform company is using in this answer.

Docker runs on various Linux administration:

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

It can also be used in production with Cloud platforms with the following services:

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

38. Is there a way to identify the status of a Docker container?

There are six possible states a container can be at any given point – Created, Running, Paused, Restarting, Exited, Dead.

Use the following command to check for docker state at any given point:

```
$ docker ps
```

The above command lists down only running containers by default. To look for all containers, use the following command:

```
$ docker ps -a
```

39. Can you remove a paused container from Docker?

The answer is no. You cannot remove a paused container. The container has to be in the stopped state before it can be removed.

40. Can a container restart by itself?

No, it's not possible for a container to restart by itself. By default the flag -restart is set to false.

41. Is it better to directly remove the container using the rm command or stop the container followed by remove container?

Its always better to stop the container and then remove it using the remove command.

```
$ docker stop <coontainer_id>
$ docker rm -f <container_id>
```

Stopping the container and then removing it will allow sending SIG_HUP signal to recipients. This will ensure that all the containers have enough time to clean up their tasks. This method is considered a good practice, avoiding unwanted errors.

42. Will cloud overtake the use of Containerization?

Docker containers are gaining popularity but at the same time, Cloud services are giving a good fight. In my personal opinion, Docker will never be replaced by Cloud. Using cloud services with containerization will definitely hype the game. Organizations need to take their requirements and dependencies into consideration into the picture and decide what's best for them. Most of the companies have integrated Docker with the cloud. This way they can make the best out of both the technologies.

43. How many containers can run per host?

There can be as many containers as you wish per host. Docker does not put any restrictions on it. But you need to consider every container needs storage space, CPU and memory which the hardware needs to support. You also need to consider the application size. Containers are considered to be lightweight but very dependant on the host OS.

DevOps Certification Training

Weekday / Weekend Batches

See Batch Details

44. Is it a good practice to run stateful applications on Docker?

The concept behind stateful applications is that they store their data onto the local file system. You need to decide to move the application to another machine, retrieving data becomes painful. I honestly would not prefer running stateful applications on Docker.

45. Suppose you have an application that has many dependant services. Will docker compose wait for the current container to be ready to move to the running of the next service?

The answer is yes. Docker compose always runs in the dependency order. These dependencies are specifications like depends_on, links, volumes_from, etc.

46. How will you monitor Docker in production?

Docker provides functionalities like docker stats and docker events to monitor docker in production. Docker stats provides CPU and memory usage of the container. Docker events provide information about the activities taking place in the docker daemon.

47. Is it a good practice to run Docker compose in production?

Yes, using docker compose in production is the best practical application of docker compose. When you define applications with compose, you can use this compose definition in various production stages like CI, staging, testing, etc.

48. What changes are expected in your docker compose file while moving it to production?

These are the following changes you need make to your compose file before migrating your application to the production environment:

- Remove volume bindings, so the code stays inside the container and cannot be changed from outside the container.
- Binding to different ports on the host.
- Specify a restart policy
- Add extra services like log aggregator

49. Have you used Kubernetes? If you have, which one would you prefer amongst Docker and Kubernetes?

Be very honest in such questions. If you have used Kubernetes, talk about your experience with Kubernetes and Docker Swarm. Point out the key areas where you thought docker swarm was more efficient and vice versa. Have a look at this blog for understanding differences between Docker and Kubernetes.

You Docker interview questions are not just limited to the workarounds of docker but also other similar tools. Hence be prepared with tools/technologies that give Docker competition. One such example is Kubernetes.

50. Are you aware of load balancing across containers and hosts? How does it work?

While using docker service with multiple containers across different hosts, you come across the need to load balance the incoming traffic. Load balancing and HAProxy is basically used to balance the incoming traffic across different available(healthy) containers. If one container crashes, another container should automatically start running and the traffic should be re-routed to this new running container. Load balancing and HAProxy works around this concept.

This brings us to the end of the *Docker Interview Questions* article. With increasing business competition, companies have realized the importance of adapting and taking advantage of the changing market. Few things that kept them in the game were faster scaling of systems, better software delivery, adapting to new technologies, etc. That's when docker swung into the picture and gave these companies boosting support to continue the race.