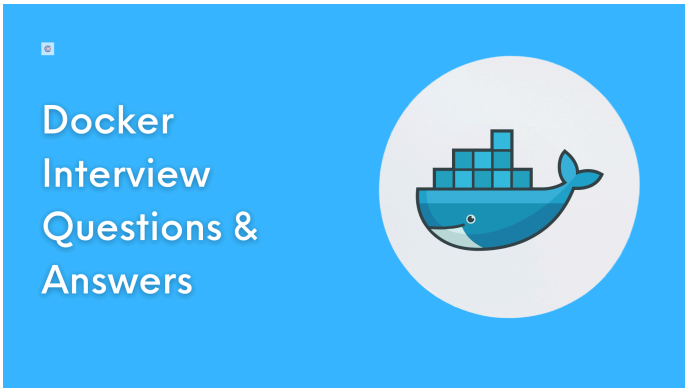
Coursesity

50 Must-Prepare Docker Interview Questions and Answers in 2025

Whether you already have experience with Docker or you're a complete newbie, this article will help you to ace your interview and get that dream job!

Coursesity Team
Apr 24, 2025 • 13 min read



Since its introduction in 2013, Docker has made a massive impact on the IT industry. In terms of container images, it became an instant success with 8 billion downloads by the end of 2017. As the demand for Docker increased, job openings increased exponentially. This article provides a comprehensive listing of all the 50 most important Docker interview questions to help you ace your interview round. But before that, let's clear

some essential concepts of Docker, such as What Docker is and why you should learn Docker.

Container engines such as Docker will play a significant role in deploying and running software from the Cloud in the upcoming future. Therefore, it is highly advisable to learn Docker with the best Docker tutorials to simplify both the development and deployment of software projects.

What is Docker?

It's a containerization platform that packages your application and all its dependencies together in the form of containers, ensuring that your application runs smoothly in any environment, be it development, test, or production. Furthermore, Docker containers enclose a piece of software in a complete filesystem that contains all of the files it needs to run: code, runtime, system tools, system libraries, etc. It wraps anything installed on the server. In this way, the software is guaranteed to run the same everywhere, no matter what environment it is running in.

Why should we learn Docker?

It takes a lot more than just writing code to develop an application! Many behind-the-scenes things go into the development process, such as using multiple frameworks and architectures for every stage of the development process, which makes the whole process complex and challenging. Using containerization eases the application development workflow, allowing developers to leverage the best technologies and environments of their choice while streamlining their workflow.

Following key points to help you out:

- Think of containers as a lightweight preinstalled box with all the packages, dependencies, and software required by your application. Just deploy to production with minimal changes to the configuration.
- Containers can be deployed on multiple platforms, including bare instances, virtual machines, Kubernetes, etc. depending on scale and platform requirements.
- To increase productivity, speed up development, consider application scalability and more effectively manage resources, it has become increasingly important to know these aspects as a developer.
- Companies like PayPal, Spotify, Uber, and many others use Docker to simplify operations and make more secure applications by bringing infrastructure and security closer together.

Best Docker Interview Questions For Beginners and Experienced

First, let's start with the most basic and most-asked questions for beginners, and then we will increase the difficulty level to experts.

1. What is Hypervisor?

Answer: A hypervisor is software that helps in enabling virtualization possible. Its other name is Virtual Machine Monitor. Its main task is to divide the host system. Additionally, it allocates resources to individually separated virtual environments. Hypervisors fall into two categories:

- **Bare Metal Hypervisor or Native Hypervisor:** A hypervisor like this runs directly on your primary host system. As it has access to your host hardware, it does not require any base server operating system.
- **Hosted Hypervisor:** Hosted hypervisors use an underlying host operating system, hence the name.

2. What is a Docker container?

Answer:

- In simple words, Docker containers contain applications and all their dependencies.
- Containers share kernel and system resources with other containers but run as isolated systems within their host operating system.
- Docker containers are intended to eliminate the infrastructure dependency when deploying and running applications. In other words, containerized applications can run on any platform regardless of what infrastructure is used.
- They are merely runtime instances of Docker images.

3. What is a DockerFile?

Answer: In general, DockerFile is a text file that contains all the commands for building a given image.

4. What are docker images?

Answer: Docker images are executable packages (packed with application code & dependencies, software packages, etc.) for creating containers. You can deploy Docker images to any Docker environment and run containers from there to start the application.

5. What is the docker command that lists the status of all docker containers?

Answer: Run the following command to get the status of all containers:

```
docker ps -a
```

6. What can you tell about Docker Compose?

Answer: Docker Compose is a tool that enables you to help define and share multi-container applications. With Compose, we can create a YAML file to define the services, and with one command, we can spin everything up or tear it all down.

7. What are some down sights of Docker?

Answer: Even though Docker is a viable platform for many use cases, it still has some disadvantages:

- **Containers don't run at bare-metal speeds:** In comparison with virtual machines, containers use resources more efficiently. Overlay networking, interfacing between containers and the host system, and so on still incur performance overhead. To achieve 100 percent bare-metal performance, you must use bare metal, not containers.
- **Persistent data storage is complicated:** If you don't first save the data somewhere else, all data inside a container disappears forever when the container shuts down. Docker offers ways to save data persistently, such as Docker Data Volumes, but this is undoubtedly a challenge that will take some time to address seamlessly.
- **Not all applications benefit from containers:** Containers are most beneficial to applications that are designed to run as a collection of discrete microservices. In addition, Docker's only real benefit is its ability to simplify application delivery by offering an easy packaging mechanism.
- **Graphical applications don't work well:** The purpose of Docker is to deploy server applications without requiring a graphical interface. Although there are some creative ways (such as using X11 video forwarding) that you can use to run a GUI app inside a container, these solutions are awkward at best.
- **The container ecosystem is fractured:** Although Docker's core platform is open source, some container products do not work with others - usually due to competition between the companies behind them. For instance, OpenShift, Red Hat's

container-as-a-service platform, works exclusively with the Kubernetes orchestrator.

8. What is Virtualization?

Answer: It is the process of creating a virtual, software-based version of anything, such as servers, computer storage, applications, etc. All it takes is a single hardware system to accomplish this. Hypervisor software is used to divide a single system into various sections. In turn, these split sections work like distinct, separate independent systems.

9. What do you know about the docker namespace?

Answer: A namespace is a feature of Linux that enables OS resources to be partitioned exclusively. Introducing namespaces into containers creates a layer of isolation between them, which is the fundamental principle behind containerization. Namespaces in Docker ensure that containers are portable and that the underlying host is not affected. Currently, Docker supports the following namespace types: PID, Mount, User, Network, and IPC.

10. Explain Docker Architecture.

Answer: The Docker Architecture has a client-server application called the Docker engine. There are three main components:

1. **A Daemon Process:** Server or long-running program. (The Docker Command)

2. **A REST API:** Specifies how programs can communicate with daemons and instruct them what to do.
3. **A CLI Client:** To control or interact with the Daemon, it uses REST API. It takes place through CLI commands or scripting.

11. How do you build a Docker file?

Answer: Once a docker file is written, it must be built so that the created image matches the given specifications. To build a Docker file, run the following command:

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

12. How can you check the Docker Client and also the Docker Server version?

Answer: By using the version command, you can perform this task:

```
$ docker version
```

13. How can you create a Docker container using an image?

Answer: You can create a Docker container using any image from the Docker repository. Here is the command to use:

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

14. How can you know of the number of containers paused, running, or stopped?

Answer: You can get detailed information about the docker installed by running the following command:

```
$ docker info
```

With this command, you can find out how many containers are running or paused or how many images, containers are stopped, and much more.

15. What is the command to see all the running containers?

Answer: You can list all containers running using the following command:

```
$ docker ps
```

16. When a docker container exits, will your data discard?

Answer: Exiting a docker container won't wipe your data. Upon writing to the container, all the data gets automatically saved. You will only be able to delete it if you intentionally do so.

17. What platforms does Docker support?

Answer: Docker supports the following platforms and servers:

- ArchLinux
- CentOS 6+
- CRUX 3.0+
- Debian

- Fedora 19/20+
- Gentoo
- macOS
- openSUSE 12.3+
- Raspbian
- RHEL 6.5+
- SLES
- Ubuntu 12.04, 13.04 et al
- Windows
- Binaries

With the given services, Docker can also be used in production on Cloud platforms:

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

18. Distinguish between virtualization and containerization.

Answer: Virtualization is the technology that simulates your physical hardware (like your CPU, memory, disk) and portrays it as a separate machine. It has its own Guest OS, Kernel, processes, drivers, etc.

Therefore, it is hardware-based virtualization. VirtualBox and VMware are the most popular technologies in this field.

On the other hand, Containerization is os-based virtualization. It does not simulate the entire physical machine. It just simulates the OS. As a result, multiple applications can use the same operating system kernel. Containers are similar to virtual machines but lack hardware virtualization. Among the most popular container technologies is Docker.

19. What is Docker Swarm?

Answer: Since Docker Swarm is an important concept, it is expected that you have worked with it. Docker Swarm is a native clustering solution for Docker. By combining several Docker hosts into one, it becomes a single virtual Docker host. Swarm exposes the standard Docker API, so any tool which already communicates with Docker daemons can use Swarm all the time.

20. What is Docker Machine?

Answer: Docker machine is a tool that allows you to install Docker Engine on virtual hosts. You can now manage these hosts using the docker-machine command. Moreover, you can also create Docker Swarm Clusters with a Docker machine.

21. Differentiate between COPY and ADD commands that are used in a Dockerfile?

Answer: Both commands offer similar functionality, but COPY offers a higher level of transparency than ADD. COPY supports copying local files

into the container, whereas ADD provides additional features like remote URLs and tar extraction.

22. Where are docker volumes stored in docker?

Answer: Non-docker entities are unable to access volumes created and managed by Docker. Docker host files are located in - `/var/lib/docker/volumes`

For Windows, it's stored in - `"C:\ProgramData\docker\volumes".`

23. Can a container get automatically restart?

Answer: The flag restart remains false by default. Therefore, a container cannot automatically restart.

On the other hand, Yes, it can get automatically restarted if you add the attribute `"--restart"` with the correct Policy value.

24. Is it okay to run Compose in production?

Answer: According to my experience, docker-compose in production is one of its most popular uses. During the defining of applications with composing, developers can use it at various production stages such as CI, testing, staging, etc.

25. Is it okay to run stateful applications over Docker?

Answer: Statement applications store their data on the local file system. As a result, moving the application to another device will make it difficult

for you to retrieve data. As a result, running stateful apps here is not recommended.

26. Can we use JSON in place of YAML for composing files in Docker?

Answer: Definitely. Since YAML is a superset of JSON, any JSON file should be valid Yaml. If using Compose, you should specify the filename to use to work with JSON files. For example:

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

27. Can you tell the approach to login to the docker registry?

Answer: You can log into your cloud repository using the docker login command.

28. Does Docker supports IPv6?

Answer: Docker supports IPv6. IPv6 networking, however, it is only available for Docker daemons running on Linux hosts. IPv6 support has been available since Docker Engine 1.5 release.

29. Does Docker support IPv6?

Answer: Docker supports IPv6. IPv6 networking, however, is only available for Docker daemons running on Linux hosts. IPv6 support has been available since Docker Engine 1.5 release.

```
{
  "ipv6": true
```

```
}
```

Make sure that you reload the Docker configuration file.

```
$ systemctl reload docker
```

If you want IPv6 support in the Docker daemon, you mustedit/etc/docker/daemon.json and set the ipv6 key to true.

30. How is Docker different from other containerization methods?

Answer: Docker containers are simple to deploy in any cloud platform. Compared to other technologies, it can get more applications running on the same hardware, developers can quickly create ready-to-run containerized applications, and it is easier to manage and deploy applications. It is even possible to share containers with your applications.

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

Answer: Platform providers such as Heroku and dotCloud, as well as large web deployments like Google and Twitter, all use container technology. It is possible to scale containers up to hundreds of thousands or even millions concurrently. Concerning requirements, containers must have memory and OS available, along with a way to use this memory efficiently when scaled.

32. Will cloud overtake the use of Containerization?

Answer: This is my personal opinion, not an authentic response. Despite the popularity of Docker containers, Cloud services are giving a good fight as well. Cloud will never replace Docker, in my opinion. Cloud services combined with containerization will elevate the game. Organizations need to take into consideration their requirements and dependencies and decide what's best for them. Cloud integration is common in most companies. In this way, they can make the most of both technologies.

33. Where can you use Docker?

Answer: Docker has applications in the following areas, depending on its use:

- Application Isolation
- Multi-tenancy
- Debugging Capabilities
- Code Pipeline Management
- Developer Productivity
- Rapid Deployment
- Simplifying Configuration

34. 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?

Answer: Yes, it will. Docker-compose runs according to dependency order. Dependencies include specifications like depend_on, link, volume_from, etc.

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

Answer: Docker-compose in production is the best practical use of docker-compose. When you define applications with compose, you can use this compose definition in various production stages lifecycle, including CI, staging, testing, etc.

36. How will you monitor Docker in production?

Answer: Monitoring an application system in production requires a set of tools. Typically, these include: Infrastructure, Network, Application features and performance, and Last-mile monitoring and log analytics (Last-mile monitoring refers to checking on user experience.)

You can monitor Docker in production by using the following tools:

- AppOptics Docker Monitoring with APM
- Sematext
- Dynatrace
- cAdvisor
- Datadog
- Prometheus & Grafana
- Elasticsearch & Kibana

- SolarWinds Server & Application Monitor
- Splunk
- ManageEngine Applications Manager
- Sumo Logic
- Sysdig

37. How many containers can run per host?

Answer: You can have as many containers as you want per host. Docker does not restrict it in any way. However, it is crucial to keep in mind that every container needs storage space, CPU, and memory, which the hardware must provide. Additionally, you need to consider the application size. Containers are lightweight but highly dependent on the host OS.

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

Answer: The best way to remove a container is to stop it and then remove it using the remove command.

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

Stopping and removing the container will allow sending SIG_HUP signals to recipients. As a result, this will provide enough time for all

containers to complete their tasks. It is considered good practice since it avoids unwanted errors.

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

Answer: Describe how Docker has helped you with rapid deployment. Tell us how you have scripted Docker and used it with other tools such as Puppet, Chef, or Jenkins. If you do not have experience with Docker but have used other tools in a similar space, be honest and state that too. In this case, it makes sense to compare Docker to other applications.

The interviewer will likely ask you this question, so keep this answer in mind.

40. How to use docker for multiple application environments?

Answer:

- The Docker-compose feature of Docker can help here. Docker-compose enables us to set up multiple services, networks, and containers in a clean manner, along with volume mappings, and then we can simply run the command "docker-compose up".
- In the case of multiple environments - i.e., development, staging, uat, or production - we would need to specify the server-specific dependencies and processes for running the application. In this case, we can create environment-specific docker-compose files with the name "docker-compose. [environment].yml" and based on the environment, we can set up and run the application.

41. What is the difference between a registry and a repository?

Answer: Unlike the Docker Hub (default registry), the Docker Registry hosts and distributes images. In contrast, the Docker Repository is a collection of related Docker images. Thus, they have a common name but different tags.

42. Explain CMD and ENTRYPOINT in a Dockerfile?

Answer: In a Dockerfile, both CMD and ENTRYPOINT instructions specify which command should be executed while running a container. There are some rules for their cooperation, including:

- It is recommended that Dockerfiles specify at least one command from CMD or ENTRYPOINT.
- When using the container as an executable, ENTRYPOINT must be defined.
- Running the container with an alternative argument will override CMD.

43. Explain Docker object labels.

Answer: Docker object label is a key-value pair stored as a string. Using labels, we can apply metadata. A label can be applied to Docker objects like images, containers, volumes, networks, nodes, and services. Every object should have a unique pair of key values. Labels remain static throughout the lifetime of an object.

44. What is the purpose of Docker_Host?

Answer: Docker object label is a key-value pair stored as a string. Using labels, we can apply metadata. A label can be applied to Docker objects like images, containers, volumes, networks, nodes, and services. Every object should have a unique pair of key values. Labels remain static throughout the lifetime of an object.

45. Explain the memory-swap flag?

Answer: When the container has used all the RAM, the memory-swap flag instructs it to write the excess memory requirements to disk. Moreover, applications that swap memory frequently to disk experience a performance penalty.

46. Explain the container orchestration and why we need to use it?

Answer: Container orchestration enables the management of containers running in a dynamic and large environment. The following tasks can be controlled and automated by container orchestration:

- Load balancing
- Scaling of containers
- Resource allocation between containers
- Provisioning and container deployment
- Monitoring the health of hosts and containers.

- Moving containers from one host to another if the host is not available or lacking resources.

47. What is CNM?

Answer: CNM stands for Container Network Model. It is a specification that defines the steps needed to implement networking in containers while maintaining the abstractions used to support multiple network drivers. CNM is based on three components, namely, sandbox, endpoint, and Network.

48. Explain Docker Trusted Registry?

Answer: Essentially, it is a way to securely store and manage Docker images over an image storage environment. It is possible to install Docker Trusted Registry on-premises or on a private cloud. CI/CD processes can use DTR to build, deliver, and run applications. DTR is highly available, efficient, and comes with built-in access control.

49. What are the commands to control Docker with Systemd?

Answer: Many Linux distributions use the system to start the Docker daemon. Start the services with the command systemct1. In the absence of systemct1, use the service command.

```
$ sudo systemctl start docker
$ sudo service docker start
```

At boot time, use the following commands to enable or disable a daemon:

```
$ sudo systemctl enable docker
$ sudo systemctl disable docker
```

Use the following command to modify daemon options:

```
$ sudo systemctl edit docker
```

View the Docker service logs:

```
$ journalctl -u docker;
```

50. Have you used Kubernetes? Which one would you prefer, Docker or Kubernetes?

Answer: Answer such questions honestly. Talk about how you have used Kubernetes and Docker Swarm. List the key areas where the docker swarm is more efficient and vice versa.

It is not only Docker that you will be asked about during your Docker interview, but also similar tools. Thus, be ready with tools/technologies that will compete with Docker. Kubernetes is one such example.

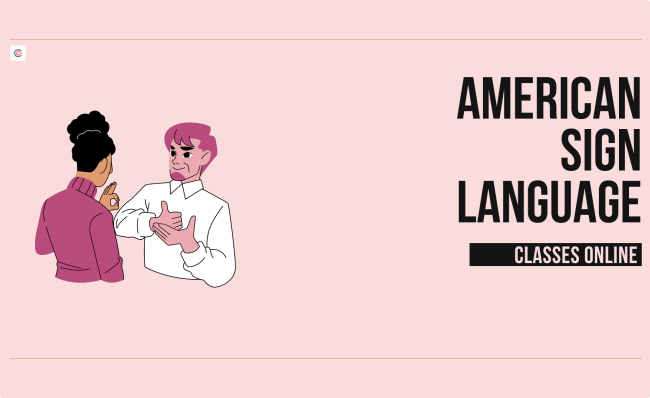
If you have made it this far, then certainly you are willing to learn more about cloud computing. Here are some more resources related to cloud computing that we think will be useful to you.

- 8 Best Snowflake Training Courses
- 5 Best Microsoft Azure Exam Certifications Courses
- 7 Best AWS Certifications & Training

Choosing What to Study: Strategies to Simplify the Process

Image Source Picking a college major is huge. It's one of those decisions that can shape your whole life after high school. Now, that…

Coursesity
May 27, 2025 • 5 min read



9 Best American Sign Language Courses - Learn ASL Online

American Sign Language (ASL) is one of the most commonly used languages in the U.S. and ranks as the fourth most-studied secon…

Coursesity Team
May 26, 2025 • 10 min read