Sagar Bhatia | 04 Jan, 2023

In this article

Top 50 Docker Interview Questions and Answers

Released in 2013, Docker is a useful tool for packing, shipping, and running applications within “containers.” The biggest tech firms, including Google, Amazon, and VMware use Docker as their go-to container technology. So, learning Docker can help you score a career with the best tech companies.

We have listed the top Docker interview questions, split into basic Docker interview questions and advanced Docker interview questions.

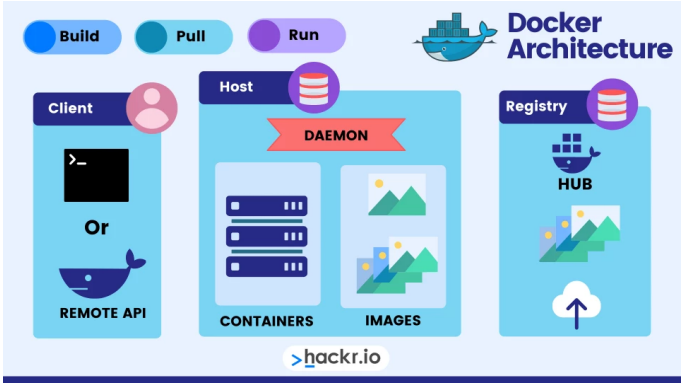
Top Docker Interview Questions

Basic Docker Interview Questions

1. What is the Docker Container?

The Docker container helps applications run smoothly. Essentially, it’s a software unit that holds code and all its dependencies — system tools, libraries, settings — everything you need to run an application.

2. Explain the components of Docker Architecture.



The components of Docker architecture are described below:

- **Host:** Docker Daemon, images, and containers fall under the host component
- **Client:** This allows communication with the Docker Host
- **Registry:** This component is used to store Docker Images. Docker Hub and Docker Cloud are public registries that anyone can use

3. Explain the Docker registry in detail.

Docker images are stored in the Docker registry, which acts as default image storage. It is a critical storage area that is regularly maintained as it holds container images. Another public registry is Docker Cloud.

4. Briefly explain the Docker container lifecycle.

The Docker container lifecycle entails the following:

- Creating the container
- Running the container
- Pausing the container
- Unpausing the container
- Starting the container
- Stopping the container
- Restarting the container
- Killing the container
- Destroying the container

5. State some important Docker commands.

Some important Docker commands include the following:

- **Build:** Builds an image file for Docker
- **Create:** Creates a new container
- **Kill:** Kills a container
- **Dockerd:** Launches the Docker daemon
- **Commit:** Creates a new image from container changes

6. What are namespaces?

Docker namespaces provide an isolated workspace in the form of a container. Docker creates namespaces for containers once they have been started, which offers a seclusion layer. Each container has its own unique namespace.

7. What is Docker Swarm?

Docker Swarm is an important tool that is used to cluster and schedule Docker containers. This makes it easy to create and maintain nodes in Docker or a solitary VS.

8. How do you identify the status of a Docker container?

Use the following command to return the status of every Docker container:

"**docker ps-a.**"

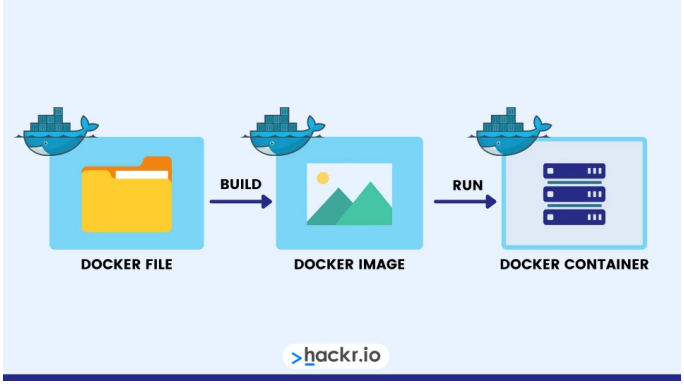
This command will return a list of all available Docker containers along with their status on the host. From the list, one can easily find the desired container to check its status.

9. What are Docker images and run commands?

Docker images are groups of files that enable instances to be created and run in distinct containers. Each instance is a separate process.

Images are created using the information required to run an executable application.

You can use the Docker run command to create and initialize a container instance using a Docker image. If an image is running, it can be linked to any number of instances (or containers).



10. What are Docker's functionalities and applications?

The following are some of Docker's functionalities and applications:

- Easy configuration at the infrastructure level
- Helps the developer concentrate exclusively on business logic as it reduces development time, thereby increasing productivity
- Amplifies debugging capabilities
- Allows an application to be isolated
- Containerization decreases the need for multiple servers.
- Facilitates rapid deployment at the OS level

11. What are Docker objects?

Docker images, services, and containers are termed Docker objects.

- **Image**: Contains instructions to create a Docker container
- **Containers**: A runnable instance of an image
- **Service**: Container scaling across various Docker Daemons as a swarm

Other Docker objects include networks and volumes.

12. Which is more suitable for a Docker container: a stateless or stateful application?

Stateless applications are more suitable for Docker containers because we can create a single, reusable container for our application, which allows us to detach state configuration from the app and the container.

By doing this, we can run multiple instances of the same container with varying production parameters, etc.

So, stateless applications give us the freedom to use the same image in a range of scenarios, including various prod and test environments.

13. What is Dockerfile used for?

Dockerfile contains a range of instructions to build an image that can then be used to initialize a container. This text document includes every command you would enter at the command line to create an image.

14. Which networks are available by default in Docker?

Default available networks include:

- **Bridge**: Default network that containers will connect to if the network has not been otherwise specified
- **None**: Connects to a container-specific network stack that doesn't have a network interface
- **Host**: Connects to the host's network stack

15. How is Docker monitored in production?

When running Docker in production, essential statistics can be gathered and examined by using tools like Docker Stats and Docker Events.

Docker Stats can be called from within a container to return data relating to the container's CPU and memory usage. This is similar to the Linux top command which can be used to examine all running processes and their current computational load.

Docker Events represent a list of commands that can be used to analyze any ongoing activities that are being processed by the Docker Daemon. These commands include attached, commit, rename, destroy, and die.

16. Shed some light on Docker's workflow.

Here's a quick run through of Docker's workflow:

- Everything starts with the Dockerfile, the image's source code
- Once created, the Dockerfile helps build the container's image, which is a compiled version of the Dockerfile
- After, it is redistributed through the registry and used to run containers

17. What is the difference between Docker run and Docker create?

If you use Docker create, the container is created in a 'stopped' state. This allows you to store and output the container ID later. If you use 'docker run' with —cidfile FILE_NAME, it won't allow you to overwrite the file.

18. What is virtualization?

Initially, virtualization enabled the logical division of mainframe systems to enable several applications to run at the same time on a system. As technology progressed, the meaning of this term evolved to represent the running of multiple (and possibly varying) operating systems (OS) on an individual x86 system.

These days, this term is broadly used to refer to the process of running multiple OS on the same hardware. In this scenario, the primary OS serves as the admin, and guests must follow the pre-defined procedures for bootstrapping, loading the kernel, etc. This ensures greater security and prevents guest systems from obtaining full system access, which could lead to a data breach.

Here are the three types of virtualization:

- Paravirtualization
- Emulation
- Container-based virtualization

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

The Docker Registry hosts and distributes images and the Docker Hub is the default registry. The Docker repository (or repo) allows you to store a collection

of Docker image versions. This means that images will have the same names, but their tags will vary to represent the different versions.

20. What are Docker Images, Docker Hub, and Docker File?

Docker images: These multi-layer files are used to create instances of a Docker container, and they are built using terminal command instructions or a pre-defined Dockerfile which contains each of these instructions. Using an image can speed up Docker build times due to caching at each step of the build sequence.

Docker hub: This is a service provided by Docker that can be used to find and share Docker images with others in a team. In the same way that GitHub is used to provide a distributed file store (with version control), Docker hub allows you to push and pull images, access private repos that store Docker images, and auto-build Docker images from GitHub or BitBucket repositories, before pushing these to Docker hub.

Dockerfile: This is a text document that is used to store build instructions for a Docker image. When run, Docker executes the commands to automatically an image.

21. How do you check versions of the Docker Client and Docker Server?

The docker version [options] command allows us to do this. If we omit the options we simply receive all of the relevant version information about the client and server. This is the command:

```
$ docker version --format'{{.Server.Version}}'
```

22. What is the login procedure for Docker Repository?

To log in to the Docker repository, use:

```
dockerlogin [OPTIONS] [SERVER]
```

To login to a self-hosted (local) registry, just add the server name:

```
$ docker login localhost:8080
```

23. What are some basic Docker commands?

Some basic Docker commands are:

- **docker push:** pushes a repository or image to a registry
- **docker run:** runs a command in a new container
- **docker pull:** pulls a repository or image from a registry
- **docker start:** starts one or more containers
- **docker stop:** stops one or more running containers
- **docker search:** searches for an image in Docker hub
- **docker commit:** commits a new image

24. How is a Docker container different from other containerization methods?

You can easily deploy Docker containers to any cloud platform. It is also possible to use ready-to-run containerized applications more quickly, as well as manage and deploy applications more easily. Docker containers can also be shared with applications, whilst other containerization methods do not have these methods.

25. What platforms does Docker run on?

Docker runs on Windows (x86-64) and Linux (on x86-64, ARM, and other CPU architectures), s390x, and ppc64le.

26. What is the memory-swap flag?

The 'memory-swap' flag is a modifier that can be combined with the run command to give a container access to additional virtual memory when it has utilized all of its provisioned physical memory (RAM). This command requires that the 'memory' flag be preset when executing the run command.

For example: –memory = "256m"; –memory-swap = "512m"; With this setup, a container is provisioned with 256MB of physical memory, and with an additional virtual swap space of 256MB (512m-256m).

27. Where are Docker volumes stored?

Volumes are stored in the Docker host filesystem: /var/lib/docker/volumes/. It is the most efficient way to ensure data persistence in Docker.

28. What is CNM?

CNM, or Container Network Model, formally defines the steps for the networking of containers, while also maintaining the abstraction used to support multiple network drivers. Sandbox, endpoint, and network are the three components.

29. What are the different kinds of mount types available in Docker?

The three types are:

- **Bind mounts:** These can be stored anywhere on the host system
- **Volume mount:** Stored in the host filesystem and Docker manages them
- **tmpfs mount:** Stored in the host system's memory and they can never be written to the host's filesystem

30. What are Docker object labels?

This is a key-value pair stored as a string. We can apply metadata using labels, which can be used for images, containers, volumes, networks, local daemons, swarm nodes, and services. Every object should have a unique label and these are static for an object's entire lifetime.

Advanced Docker Interview Questions

31. List the steps in the deployment process for Dockerized Apps stored In a Git Repo.

The deployment process can vary as a function of the production environment, but the basic process will include the following:

- Build the application with Docker Build
- Test an image
- Push a new image to the Docker registry
- Inform the remote application server to obtain the new image from the Docker registry and then run the image
- Utilize HTTP proxy for port-swapping
- Stop any older containers

32. Explain how Docker is different from other container technologies.

Whilst Docker is a relatively new container technology, it has become one of the most adopted and popular. A product of the cloud computing era, Docker provides several features that were absent in older container offerings. One of Docker's standout features is the ability to run on any type of infrastructure, both on local premises (on-prem) or in the cloud.

These days, Docker also allows the execution of more applications, the processing of these into packages, and their shipment to old servers. It can also serve as a repository for convenient containers and these can also be shared by your other applications. Finally, it is very well documented.

33. Will you lose your data if you were to exit the Docker Container?

There is no loss of data when you exit a container since it is written to the disk. This continues until the container has been entirely deleted. The container's file system is also persisted after it is halted.

34. Can JSON be used instead of YAML for the compose file in Docker? If yes, how?

Yes, it can. To do this, specify the filename as follows:

"docker-compose -f docker-compose.json up."

35. What are CMD and ENTRYPOINT in a Dockerfile?

Both of these instructions focus on the commands and parameters used by a container during execution. These instructions follow certain rules:

- The Dockerfile should specify at least one command from CMD or ENTRYPOINT
- ENTRYPOINT provides commands to determine how a container will be executed. These arguments and parameters cannot be overridden when using the run command the the command line (CLI)
- CMD specifies the default image to be executed when starting a container. Any parameters passed to this command can be overridden at the CLI if the user provides an alternative argument with the run command

36. Explain the process to run an application inside a Linux Container using Docker.

In order to run an application within a Linux container using Docker, follow these steps:

- Install Docker, then run it
- Pull the Fedora 21 (Linux OS) base image from Docker hub
- Using the Docker base image, load your application
- Run a container in interactive mode by using the new image
- Check the system containers
- Start or stop the Docker container
- Remove the image or the container

37. What is a Hypervisor?

A hypervisor is a form of management software that can be used to create and run virtual machines (VM). This enables a host system to accommodate multiple guest VMs and these will share computational resources including RAM and CPU. By allocating the required computational resources to each VM, it is possible to reduce physical hardware requirements and the maintenance of these.

The two types of hypervisors are:

- **Type I**: This is a lightweight OS that is run on the host system
- **Type II**: This runs like any other piece of software within an existing OS

38. Explain containerization.

Docker containers contain things including code, system tools, libraries, runtime, and the settings required for application execution, with the app existing on top of the Docker engine layer. This concept is called containerization

39. What is the main difference between containerization and virtualization?

Virtualization allows you to run multiple OS on a single physical server. The OS also handles containerization, which takes place under a single server or VM.

40. Is it possible for a container to restart by itself?

Yes, this is possible. However, Docker provides a range of behaviors that can be configured to control this:

- **Off:** If a container fails or stops, it will not be restarted
- **On-failure:** If the container fails due to non-user error, the container will restart
- **Unless-stopped:** The container will restart if the user stops it manually
- **Always:** Regardless of the error or reason for stopping, a container will always restart

The command is:

```
$ docker run -dit — restart [unless-stopped|off|on-failure|alwa
```

41. Is it possible for the cloud to overtake the use of Containerization?

With a question like this, it can only be answered subjectively, or with an opinion. As of today, many companies have come to rely on the combination of cloud computing and containerization to achieve a highly performant system design.

42. What are the various possible states of the Docker Container?

The different states of the Docker container are:

- **Created:** The container has been created, but is not active
- **Restarting:** The container is in the process of being restarted
- **Running:** The container is running
- **Paused:** The container's processes have been paused
- **Exited:** The container was run and it completed its process
- **Dead:** The container is not functioning or it was partly removed, but is still using resources. The daemon will try to remove it when it is restarted

43. Explain container orchestration. Why do we need it?

Container orchestration reduces the need for developers to manually manage container-related activities via automation of the following:

- Provisioning and deploying containers
- Network load-balancing
- Allocation of resources for current containers
- Health monitoring for containers and hosts
- Container scaling
- Failure prevention via the transfer of a container to a new host if the current host becomes unresponsive or lacks computational resources

44. What are some Advanced Docker Commands?

Some advanced Docker commands are:

- **docker -version:** See what version of docker is installed. Syntax: Docker --version

- **docker ps:** Lists all the docker containers that are running along with container details. Syntax: docker ps
- **docker ps -a:** Lists all the containers, including those that are running, stopped, or exited, along with relevant details. Syntax: docker ps -a
- **docker exec:** Access a container and run commands inside that container. Syntax: docker exec [options]
- **docker build:** Builds an image from a Dockerfile. Syntax: docker build [options] path|URL
- **docker rm:** Removes a container with the given container id. Syntax: docker rm <container_id>
- **docker rmi:** Removes a docker image with the given image id. Syntax: docker rmi <image_id>
- **docker info:** Returns detailed information about Docker installed on the system including number of images; containers running, paused, or stopped; server version; volume; runtimes; kernel version; and total memory etc. Syntax: docker info
- **docker cp:** Copies a file from a docker container to the local system. Syntax: docker cp <source_path> <dest_path>
- **docker history:** Displays the history of the docker image with the given image name. Syntax: docker history <img_name>

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

Docker Daemon can be started using the system:

```
$ sudo systemctl start docker
```

Use systemct1 to start services. If this is not available, use the service command:

```
$ sudo service docker start
```

To enable and disable a Daemon during boot time, use:

```
$ sudo systemctl enable docker

$ sudo systemctl disable docker
```

To modify Daemon options, use:

```
$ sudo systemctl edit docker
```

To view logs related to Docker service:

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

46. What is the process of scaling your Docker containers?

The docker-compose command can be used to horizontally scale the number of Docker containers that you require by starting the required number of additional instances. The syntax to achieve this is:

```
$] docker-compose --file docker-compose-run-srvr.yml scale <ser
```

In the command above, we are passing the docker-compose-run-srver.yml YAML file as the service name, and we must provide an integer value, 'n' , to represent the number of additional instances we require to scale horizontally. Lastly, we can check the details of these new containers with:

```
$] docker ps -a
```

47. What are the major actions in the Docker container life cycle?

Here are the steps:

- **Create container:** docker create --name <container-name> <image-name>
- **Run docker container:** docker run -it -d --name <container-name> <image-name> bash
- **Pause container**: docker pause <container-id/name>
- **Unpause container:** docker unpause <container-id/name>
- **Start container:** docker start <container-id/name>
- **Stop container:** docker stop <container-id/name>
- **Restart container:** docker restart <container-id/name>
- **Kill container:** docker kill <container-id/name>
- **Destroy container:** docker rm <container-id/name>

48. What is the Docker Trusted Registry?

The Docker Trusted Registry is used to store and manage Docker images. It is available both locally and on the cloud. It can also be used during CI/CD processes for building, delivering, and running applications. It is readily available, efficient, and has built-in access control.

49. What is the purpose of Docker_Host?

Docker_host specifies the URL or Unix socket path used to connect to the Docker API. The default value is: UNIX://var/run/docker.sock

To connect to the remote host, provide the TCP connection string as: TCP://192.0.1.20:3230

50. Is it possible to run multiple copies of a Compose file on the same host? If so, How?

Yes, this can be done by using the docker-compose up command with a YAML file that has been written to configure the application's services.

To do this, complete the following steps:

- Create a Dockerfile to configure an app environment, thus allowing it to be replicated anywhere
- Create a docker-compose.yml (YAML) file to define the services for the application
- Run the docker-compose up command to create and start the app

51. What is Docker Push?

This allows us to push or share a local Docker image or a repository to a central repository.

Start Preparing for Your Docker Interview Today

That ends our list of the top 50 Docker interview questions. Use them to prepare for your interview — and don't forget to get some practical experience under your belt.

If you want to learn more about Docker, check out the best Docker tutorials, submitted and recommended by the community. You should also check out courses like the Docker Crash Course from Udemy. Your final step to ace your Docker interview?

Explore these Cracking the Coding Interview: 189 Programming Questions and Solutions.

Frequently Asked Questions

1. What are Docker Swarm Interview Questions?

Docker swarm is a container orchestration tool that allows multiple containers across multiple host machines. The above interview questions on Docker contain related questions.

2. What is the Main Use of Docker?

Docker is a containerization platform that allows developers to build applications in containers. These are standalone executables that can run on any Operating System.

People are also reading:

- Jenkins Interview Questions
- Comparison between Nginx or Apache
- Jquery Interview Questions
- Linux Interview Questions
- Top DevOps Interview Questions and Answers
- Get the Difference between Linux vs. Windows
- Top SQL Interview Questions and Answers

By Sagar Bhatia

Sagar is an engineering graduate and a technology lover and has been writing across various disciplines for over 5 years now. An avid gamer himself, he wishes to create a venture revolving around the e-sports domain in India.

View all post by the author

Learn More

20 Social Media Interview Questions for Social Media Managers

Top 48 Networking Interview Questions and Answers To Get Hired in 2023

Top 20 REST API Interview Questions & Answers [2023]

Interview Questions Other

Interview Questions Networking Jobs

Interview Questions rest api