Docker Interview Questions

Download PDF Print

Use the filter to quickly find topics like images/layers, containers, volumes/binds, networks, Dockerfiles, build cache, multi-stage builds, and security.

Filter (e.g., images, containers, volumes, networks, Dockerfile, Showing 51 of 51

1. ## What is Docker and what is its purpose?

   Docker is an open-source platform that automates the deployment, scaling, and management of applications inside containers. It provides a way to package and run applications in isolated environments, allowing for easier management, deployment, and scaling.

2. ## What is a container in Docker?

   A container in Docker is a standalone executable package that includes everything needed to run the application, including the application code, runtime, libraries, environment variables, and system tools.

3. ## What is a Docker image?

   A Docker image is a read-only template that contains a set of instructions for creating a container that can run on the Docker platform. It provides a convenient way to package up applications and preconfigured server environments, which you can use for your own private use or share publicly with other Docker users.

4. ## What is the difference between a Docker image and a container?

   A Docker image is a static, immutable, packaged version of a piece of software. A container is a running instance of a Docker image.

5. ## What is a Dockerfile and what is its purpose?

   A Dockerfile is a script that contains instructions to build a Docker image. It includes the base image to use, dependencies to install, and how to configure the application and its environment.

6. ## What is a Docker registry and why is it important?

   A Docker registry is a place where Docker images are stored and distributed to others. Docker Hub is a public registry that anyone can use, but you can also set up your own private registry. A registry is important because it provides a centralized place to manage images and control access to them.

7. ## What is a volume in Docker and what is its purpose?

   A volume in Docker is a way to persist data outside of a container's filesystem. It allows data to be stored in a designated location on the host and can be shared between containers. Volumes are useful for preserving data even if the container is deleted or recreated.

8. ## What is Docker Compose and what is it used for?

   Docker Compose is a tool for defining and running multi-container Docker applications. It allows you to define the services, networks, and volumes needed for an application in a single file, and then start and stop the services with a single command.

9. ## What is a Docker network and why is it important?

   A Docker network is a virtual network that allows containers to communicate with each other and with the host. Docker networks are important because they provide a way to isolate and secure the communication between containers and to manage their network configuration.

10. ## What is a Docker swarm and how does it relate to a Docker cluster?

    A Docker swarm is a native orchestration system for Docker that allows you to manage a cluster of Docker nodes as a single virtual system. A Docker cluster is a group of Docker nodes that are managed together. A swarm is a specific type of cluster that is optimized for deploying and managing Docker services.

11. ## What is the purpose of the Docker API and how is it used?

    The Docker API is the programming interface to the Docker daemon, which allows you to build and manage containers, images, networks, and volumes programmatically. The API can be used to automate tasks, integrate Docker into existing systems, or build new tools and applications on top of Docker.

12. ## How does Docker ensure the security of containers?

    Docker provides several security features to help ensure the security of containers, including user namespace remapping, seccomp profiles, control groups, and secure communication between containers and the host. Additionally, Docker images can be scanned for vulnerabilities, and Docker users can follow best practices for securing containers and the host system.

13. ## What is a Docker image layer and what is its purpose?

    A Docker image layer is a change to the file system of a Docker image. Each layer represents a separate command in a Dockerfile, and each layer is built on top of the previous one. The purpose of layers is to allow images to be constructed more efficiently and to take

advantage of image caching during builds.

14. ## What is the difference between a bind mount and a volume in Docker?

    A bind mount in Docker is a type of volume that links a directory on the host file system to a directory in a container. A volume in Docker is a separate, isolated section of the host file system that can be accessed and modified by containers. A bind mount is generally used for development purposes, while a volume is used for production and for storing data that should persist even if the container is deleted.

15. ## What is the Docker daemon and what is its role in the Docker architecture?

    The Docker daemon is a background process that manages containers, images, networks, and volumes on a host. The Docker daemon listens for API requests and communicates with the Docker client to execute commands and perform tasks. The Docker daemon is an essential component of the Docker architecture, as it is responsible for the execution of containers and for managing the state of the host.

16. ## What is the difference between the Docker CE (Community Edition) and Docker EE (Enterprise Edition)?

    Docker CE is the free, open-source version of Docker that is designed for developers and small teams. Docker EE is a commercial product that provides additional features and support for enterprise users, including a certified container runtime, centralized image management, and integrated tools for security and compliance.

17. ## How does Docker handle resource constraints for containers?

    Docker allows resource constraints to be set for containers, including limits on CPU, memory, and other resources. These constraints help to ensure that containers do not consume too many resources and negatively impact the host or other containers. Resource constraints can be set using the `--cpus`, `--memory`, and other command-line options, or using the Docker API.

18. ## What is the purpose of a Dockerfile and how is it used?

    A Dockerfile is a script that contains instructions for building a Docker image. The Dockerfile specifies the base image to use, the files to include in the image, and any other configuration or setup that is needed. The Dockerfile is used with the `docker build` command to create an image, which can then be run as a container.

19. ## What is the difference between a Docker image and a container?

    A Docker image is a read-only template that defines the contents and structure of a container. A container is a runnable instance of a Docker image that can be started, stopped, and deleted. An image can be used to create many containers, and changes made to a container are not persisted when the container is deleted.

20. ## What is the difference between a Docker registry and a Docker repository?

    A Docker registry is a service that stores and distributes Docker images. A Docker repository is a collection of related images within a registry. A registry can contain many repositories, and a repository can contain multiple images for different purposes or stages of development.

21. ## What is the purpose of the Docker CLI and how is it used?

    The Docker CLI is the command-line interface to the Docker daemon. The Docker CLI can be used to build images, run containers, manage containers and images, and perform other tasks related to Docker. The CLI is used by typing commands in a terminal or command prompt, and it communicates with the Docker daemon to execute the commands.

22. ## What is the difference between `COPY` and `ADD` in a Dockerfile?

    `COPY` and `ADD` are both used to copy files from the host file system into a Docker image. The main difference between the two is that `ADD` has some additional functionality: it can automatically extract files from a tar archive if they are added to the image with a `.tar` extension. However, it is generally recommended to use `COPY` instead of `ADD` because it is simpler and less prone to unexpected behavior.

23. ## How does Docker ensure the security of containers?

    Docker provides several security features to help ensure the security of containers, including: - Isolation of containers from the host and from each other - User namespaces to separate the root user in the container from the root user on the host - Control groups to limit the resources available to a container - Image signing and verification to ensure the authenticity of images - Secure communication between the Docker client, daemon, and registry These security features help to prevent malicious containers from accessing sensitive data on the host or other containers, and from disrupting the host or other containers.

24. ## What is the difference between the `CMD` and `ENTRYPOINT` instructions in a Dockerfile?

    `CMD` and `ENTRYPOINT` are both used to specify the command that should be run when a container is started from an image. The main difference between the two is that `ENTRYPOINT` specifies the command that should always be run when the container starts, while `CMD` provides default arguments that can be overridden when the container is started. `ENTRYPOINT` is typically used to specify the main application that should be run in a container, while `CMD` is used to provide arguments that can be customized when the container is started.

25. ## What is a Docker volume and how is it used?

A Docker volume is a way to store data outside of a Docker container's filesystem. Volumes can be used to persist data even if the container is deleted, and can be shared between containers or between the host and a container. Volumes can be created and managed using the `docker volume` command, and can be mounted into a container using the `--mount` or `-v` option when starting a container.

26. ## What is the difference between a `docker run` command and a `docker start` command?

    The `docker run` command is used to create and start a new container from an image, while the `docker start` command is used to start an existing, stopped container. The `docker run` command creates a new container each time it is run, while the `docker start` command starts a container that has been previously created and stopped.

27. ## What is the Docker Swarm mode and how does it work?

    Docker Swarm mode is a native orchestration solution for Docker containers. It allows you to manage a group of Docker nodes as a single, virtual host, making it easier to deploy, scale, and manage containerized applications. Swarm mode works by automatically distributing containers across multiple nodes, providing automatic failover and high availability, and providing a single API endpoint for interacting with the swarm. To use Swarm mode, you create a swarm, add nodes to the swarm, and then deploy services to the swarm.

28. ## What is the purpose of the Docker `network` command?

    The `docker network` command is used to manage Docker networks. Docker networks provide a way to isolate containers from each other, while still allowing them to communicate with each other and with the host system. With the `docker network` command, you can create and manage custom networks, connect and disconnect containers from networks, and view information about networks and their connected containers.

29. ## What is the Docker image layering system and how does it work?

    Docker uses a layering system for images. Each image is built from a series of layers, where each layer represents a change or an addition to the previous layer. When you create a new container from an image, Docker adds a new, writable layer on top of the image layers. This allows you to make changes to the container's filesystem, while still sharing the underlying image layers with other containers. This layering system is what makes Docker images and containers lightweight and fast to start.

30. ## What is the purpose of the Docker registry?

    A Docker registry is a centralized place to store and distribute Docker images. The Docker registry makes it easy to share images with others, and to use images created by others in your own applications. Docker Hub is a public registry that is free to use, and that provides a large collection of images for a wide variety of applications. You can also run your own private registry if you need to store sensitive images or control access to images.

31. ## What is a Docker container?

    A Docker container is a standalone executable package that includes everything needed to run a piece of software, including the code, runtime, system tools, libraries, and settings. Containers are isolated from each other and from the host system, providing a secure and reliable way to run applications. Containers are created from images, which are snapshots of a container's filesystem at a particular point in time.

32. ## What is the difference between a Docker image and a Docker container?

    A Docker image is a read-only template that is used to create Docker containers. An image includes all the software and dependencies required to run an application. A Docker container is a running instance of a Docker image, and it includes a writable layer on top of the image layers that can be used to make changes to the filesystem. In other words, a Docker image is a blueprint for a container, while a Docker container is a live, running instance of that blueprint.

33. ## What is the `Dockerfile` used for?

    A Dockerfile is a script used to automate the creation of a Docker image. It specifies the base image to use, the software to install, the configuration to set up, and the command to run when the container starts. When you run the `docker build` command, Docker reads the Dockerfile and creates a new image based on the instructions in the Dockerfile. Dockerfiles are a convenient and repeatable way to create images, and they allow you to share your images with others and to automate your build process.

34. ## What is the Docker `run` command used for?

    The `docker run` command is used to create and start new containers from images. When you run the `docker run` command, Docker looks for the image in your local image cache, and if the image is not found, it pulls the image from the Docker registry. Then Docker creates a new container from the image and starts the container, running the command specified in the image's definition or in the `docker run` command. The `docker run` command is the main way to interact with Docker containers, and it provides a lot of options for customizing how containers are created and run.

35. ## What is the purpose of the Docker `exec` command?

    The `docker exec` command is used to run a command in a running Docker container. The `docker exec` command allows you to interact with a container's process, or to run additional commands inside the container. For example, you might use `docker exec` to check the contents of a file inside a container, or to run a shell script inside the container. The `docker exec` command provides a convenient way to run commands in containers without having to start a new container or login to the container directly.

36. ## What is the Docker `push` command used for?

    The `docker push` command is used to upload a Docker image to a Docker registry. The `docker push` command is used after you have created an image and you want to share the image with others, or you want to store the image in a remote registry so that you can use it on other systems. The `docker push` command takes the image name and the registry address, and it uploads the image to the specified registry. After the image is pushed, others can use the image by pulling it from the registry with the `docker pull` command.

37. ## What is the difference between the Docker `commit` and `save` commands?

    The `docker commit` and `docker save` commands are used to create a new image from a container, but they have different use cases and output formats. The `docker commit` command is used to create a new image from the changes made to a container's filesystem. The `docker commit` command creates a new image in the Docker image format, and the new image can be pushed to a Docker registry. The `docker save` command is used to save an image to a tar archive, which can be used to transfer images between systems or to backup images. The `docker save` command creates a tar archive that can be loaded into Docker using the `docker load` command.

38. ## What is the purpose of the Docker `networks` command?

    The `docker network` command is used to manage Docker networks, which are virtual networks that containers can use to communicate with each other and with the host system. The `docker network` command allows you to create, inspect, and delete Docker networks, and to connect and disconnect containers from networks. Docker provides several network drivers that allow you to configure different network types, such as bridge networks, host networks, and overlay networks. The `docker network` command is a powerful tool for managing the networking of your Docker containers, and it is an important part of the Docker networking model.

39. ## What is the Docker `volume` command used for?

    The `docker volume` command is used to manage Docker volumes, which are persistent data storage for containers. The `docker volume` command allows you to create, inspect, and delete Docker volumes, and to connect and disconnect volumes from containers. Docker volumes provide a way to persist data outside of a container's filesystem, so that the data can survive even if the container is deleted. Docker volumes are stored on the host system, and they can be backed by a variety of storage drivers, such as local disk, NFS, and cloud storage services.

40. ## What is the Docker `stats` command used for?

    The `docker stats` command is used to display live resource usage statistics for running Docker containers. The `docker stats` command shows CPU usage, memory usage, network I/O, and other resource usage metrics for all containers, or for a specific container. The `docker stats` command provides real-time performance monitoring for your containers, and it can help you to troubleshoot performance issues, to optimize resource usage, and to monitor the health of your containers. The `docker stats` command is a useful tool for monitoring and managing your Docker containers, and it is an important part of the Docker toolset.

41. ## What is the Docker `exec` command used for?

    The `docker exec` command is used to run a command in a running Docker container. The `docker exec` command takes the container name or ID and the command to run as arguments, and it runs the command inside the container's environment. The `docker exec` command can be used to run administrative tasks, such as inspecting the contents of a container, updating configuration files, and debugging applications. The `docker exec` command is also useful for attaching to a running container and running commands interactively. The `docker exec` command is a powerful tool for managing and troubleshooting Docker containers.

42. ## What is the purpose of the Docker `system` command?

    The `docker system` command is used to perform administrative tasks on the Docker system, such as managing images, containers, and networks, and managing system-wide settings and configuration. The `docker system` command provides several subcommands, such as `docker system df` to show disk usage statistics, `docker system prune` to remove unused data, and `docker system events` to display system-wide events. The `docker system` command is a powerful tool for managing and maintaining your Docker installation, and it provides a central location for performing administrative tasks.

43. ## What is the difference between a Docker `image` and a `container`?

    A Docker image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software, including the code, a runtime, libraries, environment variables, and config files. Docker images are created from a set of instructions called a Dockerfile, and they can be stored in a registry, such as Docker Hub, for distribution and reuse. A Docker container is a runnable instance of a Docker image. When you run a Docker container, you are creating a new instance of a Docker image and running the software inside the container. Containers provide a way to isolate and manage the software and its dependencies, and they can be easily created, started, stopped, moved, and deleted.

44. ## What is the purpose of the `Dockerfile`?

    A Dockerfile is a script that contains a set of instructions for building a Docker image. The Dockerfile specifies the base image to use, the application code and dependencies to include, and any configuration and runtime settings. The Dockerfile is used by the `docker build` command to create a new Docker image, which can then be run as a Docker container. The Dockerfile is a key component of the Docker platform, and it provides a repeatable and automated way to build, package, and distribute software. The Dockerfile is written in a simple syntax that is easy to read and understand, and it provides a powerful and flexible way to define the components and dependencies of your application.

45. ## What is the difference between `docker run` and `docker start` commands?

    The `docker run` command is used to run a new instance of a Docker image as a container. The `docker run` command takes the image name, creates a new container based on the image, and starts the container. The `docker run` command is typically used to run a container for the first time. The `docker start` command is used to start a stopped container. The `docker start` command takes the container name or ID as an argument, and it restarts the container based on the configuration defined in the original `docker run` command. The `docker start` command is typically used to restart a stopped container, or to start multiple instances of the same image.

46. ## What is a Docker registry?

    A Docker registry is a centralized repository for storing and distributing Docker images. Docker registries are used to store and share Docker images with others. The most popular public registry is Docker Hub, which is hosted by Docker, Inc. and provides a free service for hosting and distributing Docker images. Private registries, such as the Docker Trusted Registry, can be hosted on-premises or in the cloud, and they provide an enterprise-grade solution for hosting and distributing Docker images within an organization. Docker registries are used to store and distribute images for deployment, testing, and production, and they provide a secure and scalable solution for managing and distributing Docker images.

47. ## What is the purpose of the `--rm` option in the `docker run` command?

    The `--rm` option in the `docker run` command is used to automatically remove the container when it exits. The `--rm` option is used to automatically clean up containers when they are no longer needed, and it helps to reduce clutter and conserve disk space. When the `--rm` option is used, the container is automatically removed when it exits, and its file system and network settings are destroyed. The `--rm` option is particularly useful for containers that are used for one-off tasks or for testing purposes, and it helps to ensure that resources are conserved and that the system remains clean and organized.

48. ## What is the purpose of the `-it` option in the `docker run` command?

    The `-it` option in the `docker run` command is used to run the container in interactive mode. The `-it` option allows you to attach to the container's console, and it provides an interactive shell that you can use to run commands inside the container. The `-it` option is used to run a container in an interactive manner, and it is typically used for debugging, testing, and development purposes. The `-it` option makes it easy to inspect the state of the container and its environment, and it provides a way to interact with the container in real-time. The `-it` option is a useful tool for managing and troubleshooting Docker containers.

49. ## What is a Docker volume?

    A Docker volume is a way to persist data generated by a container, even after the container has been deleted. A Docker volume is a separate area for storing data outside of a container's filesystem. Docker volumes are stored on the host system and can be mounted into containers at a specified location. This allows data to be persisted even if the container is deleted or recreated. Docker volumes are useful for storing database data, logs, configuration files, and other types of data that need to persist even if the container is deleted. Docker volumes can be managed using the `docker volume` command.

50. ## What is a Docker network?

    A Docker network is a virtual network that provides communication between containers. A Docker network provides a way for containers to communicate with each other, and with the host system, over a virtual network. Docker networks are created using the `docker network` command, and they provide a way to isolate containers from each other and from the host system. Docker networks are useful for creating isolated network segments for different types of containers, such as front-end and back-end containers, or for isolating containers from the host system. Docker networks can also be used to create a private network for communication between containers, and to connect containers to other networks, such as the host system's network.

51. ## What is the difference between a Docker image and a Docker container?

    A Docker image is a static, read-only definition of a container, while a Docker container is a running instance of a Docker image. A Docker image is a blueprint for creating a container, and it defines all the required configurations, files, and dependencies for a container. A Docker container is created from a Docker image, and it is a running instance of the image. A Docker container is a live instance of a Docker image, and it can be modified and interacted with. Docker images are stored in a Docker registry, and they can be pulled and run as containers on any host system that has Docker installed. A single Docker image can be used to create multiple containers, and each container can be configured and customized as needed.

Tip: In answers, call out how layers and cache work, image size optimization (alpine, multi-stage), least-privilege containers, and compose vs. swarm/k8s.

Java 8	Q&A
Spring	Q&A
Hibernate	Q&A
React JS	Q&A
SQL	Q&A
MongoDB	Q&A
JavaScript	Q&A
Kafka	Q&A
Design Patterns	Q&A
Microservices	Q&A
REST API	Q&A
Behavioral	Q&A

QUICK ACTIONS

Start System Design | Practice Coding | Earn a Certificate

RECOMMENDED READING

- Learn Java 17 Programming
- Learn JavaScript Quickly
- SQL QuickStart Guide
- System Design Interview

See all ➜

SPONSORED

Ads help keep content free.

NEED HELP?

Have a topic request or correction? We'd love to hear from you.

Contact Us

THANK YOU!

Your comments and complements are always the inspiration that keeps us motivated.

ABOUT

PROJECTS

ABOUT US

BLOG

AWARDS

ADDRESS

Sterling, Virginia, 20166, US

info@techventurellc.com

641 451 3463

641 451 3463

FOLLOW US