1. Have you worked with containers? What is a docker basically?

Docker container is an open source software development platform. Its main benefit is to package applications in “containers,” allowing them to be portable among any system running the Linux operating system (OS).

2. What is orchestration in cloud computing?

In most situations, cloud automation describes a task or function accomplished without human intervention. Cloud orchestration describes the arranging and coordination of automated tasks, ultimately resulting in a consolidated process or workflow. It is simplest to see this in an example.

3. What is Docker and what does it do?

Docker is a tool designed to make it easier to create, deploy, and run applications by using containers. Containers allow a developer to package up an application with all of the parts it needs, such as libraries and other dependencies, and ship it all out as one package.

Docker provides this same capability without the overhead of a virtual machine. It lets you put your environment and configuration into code and deploy it. The same Docker configuration can also be used in a variety of environments. This decouples infrastructure requirements from the application environment.

4. Do you have any expreience with CI tools? Which ones?

Jenkins, Bamboo, TeamCity, etc.

5. Describe your experience implementing continuous deployment in a production environment

Answer should describe personal experience on types of jobs run, systems integrated, etc.

6. What function does DNS play on a network?

The DNS plays a critical role in supporting the Internet infrastructure by providing a distributed and fairly robust mechanism that resolves Internet host names into IP addresses and IP addresses back into host names.

7. What is HTTP?

The Hypertext Transfer Protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.[1] HTTP is the foundation of data communication for the World Wide Web.

Hypertext is structured text that uses logical links (hyperlinks) between nodes containing text. HTTP is the protocol to exchange or transfer hypertext.

8. What commands do you know that can be used to check DNS records?

$ host example.com

$ nslookup example.com

$ dig example.com

9. How to get all environment variables and how can you use them?

All UNIX-like operating systems such as OpenBSD, Linux, Redhat, CentOS, Debian allows you to set environment variables. When you log in on UNIX, your current shell (login shell) sets a unique working environment for you which is maintained until you log out.

printenv\env - command to print all or part of environment

Full command:

$ printenv PATH HOME

$PATH - Display lists directories the shell searches, for the commands.

$HOME - User's home directory to store files.

10. What is version control and why should version control system (VCS) be used?

Define version control and talk about how this system records any changes made to one or more files and saves them in a centralized repository. VCS tools will help you recall previous versions and perform the following:

Go through the changes made over a period of time and check what works versus what doesn’t.

Revert specific files or specific projects back to an older version.

Examine issues or errors that have occurred due to a particular change.

Using VCS gives developers the flexibility to simultaneously work on a particular file and all modifications can be logically combined later.

11. What is Docker hub?

Docker hub is a cloud-based registry service which allows you to link to code repositories, build your images and test them, stores manually pushed images, and links to Docker cloud so you can deploy images to your hosts. It provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, and workflow automation throughout the development pipeline.

12. What is Dockerfile used for?

Dockerfile is nothing but a set of instructions that have to be passed on to Docker itself, so that it can build images automatically reading these instructions from that specified 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.

13. Tell us how you have used Docker in your past position?

This is a question that you could bring upon your whole experience with Docker and if you have used any other Container technologies before Docker. You could also explain the ease that this technology has brought in the automation of the development to production lifecycle management. You can also discuss about any other integrations that you might have worked along with Docker such as Puppet, Chef or even the most popular of all technologies – Jenkins. If you do not have any experience with Docker itself but similar tools from this space, you could convey the same and also show in your interest towards learning this leading containerization technology.

14. How to create Docker container?

You can create a Docker container out of any specific Docker image of your choice and the same can be achieved using the command given below:

docker run -t -i command name

The command above will create the container and also starts it for you. In order to check whether the Docker container is created and whether it is running or not, you could make use of the following command. This command will list out all the Docker containers along with its statuses on the host that the Docker container runs.

docker ps -a

15. How to stop and restart the Docker container?

The following command can be used to stop a certain Docker container with the container id as

CONTAINER\_ID:

docker stop CONTAINER\_ID

The following command can be used to restart a certain Docker container with the container id as

CONTAINER\_ID:

docker restart CONTAINER\_ID

16. How far do Docker containers scale?

Best examples in the Web deployments like Google, Twitter and best examples in the Platform Providers like Heroku, dotCloud run on Docker which can scale from the ranges of hundreds of thousands to millions of containers running in parallel, given the condition that the OS and the memory doesn’t run out from the hosts which runs all these innumerable containers hosting your applications.

17. What are the various states that a Docker container can be in at any given point in time?

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

• Running

• Paused

• Restarting

• Exited

19. Can you remove a paused container from Docker?

To answer this question blatantly, No, it is not possible to remove a container from Docker that is just paused. It is a must that a container should be in the stopped state, before it can be removed from the Docker container.

20. What is Mesos?

Mesos is a cluster manager that provides efficient resource isolation and sharing across distributed applications or frameworks. Mesos is a open source software originally developed at the University of California at Berkeley.

21. What is Marathon?

Marathon is a production-grade container orchestration platform for Mesosphere's Datacenter Operating System (DCOS) and Apache Mesos.

22. What are the basic parameters you have to pass in a json file to orchestrate a service on Mesos via Marathon?

Bare minimum parameters would be service name, cpu, memory and application to deploy and the containerizer to be used.

23. What is Git?

Git is a Distributed Version Control system (DVCS). It can track changes to a file and allows you to revert back to any particular change.

24. In Git how do you revert a commit that has already been pushed and made public?

It can be done in two ways:

By removing or fixing the bad file in a new commit and pushing it to the remote repository. Once the necessary changes to the file has been made, commit it to the remote repository. Use: git commit -m “commit message”

By creating a new commit that undoes all changes that were made in the bad commit. To do this, use command: git revert <name of bad commit>

25. What Is Ldap?

A GLDAP (Light-Weight Directory Access Protocol) determines how an object in an Active Directory should be named. LDAP is the industry standard directory access protocol, making Active Directory widely accessible to management and query applications. Active Directory supports LDAPv2 and LDAPv3.

26. According to an HTTP monitor, a website is down. You're able to telnet to the port, so how do you resolve the issue?

"Assuming the web page is up, I would investigate what could be wrong with the monitor. It could be a system overload or flapping, among other issues. Identifying the problem helps me prevent it in the future."

27. What are three attributes that make you a great systems engineer?

This question lends insight into candidates' self-awareness skills as well as whether their values match those of your business. Look for:

Answers that match areas of emphasis in your job description

At least one soft skill

Attributes that fit your company culture

"I'm analytical and curious. I always dig to find out why a problem occurred. Otherwise, it is liable to happen again and hurt the business. I am also a great communicator, able to share my insight with anyone in jargon-free language."

28. A Linux administrator wants to review the messages that scrolled up the screen during a system boot. How can this be accomplished?

The boot kernel messages are saved to the log file /var/log/dmesg. He can check this file.

29. Which utility could you use to repair the corrupted file system?

You can use fsck to repair the corrupted file system.

30. A user can not access a remote server. Which command he can use to verify that remote server is up and which command should he use to check if port 22 is open or closed?

He can use ping command to check whatever remote server is up or not and use the telnet command to check port accessibility.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1.what is docker image and docker registry?

A Docker image consists of many layers. Each layer corresponds to a command in an image’s Docker file. This image provides isolation for an application when you run a Docker image as a container.

You can run many containers from a single Docker image. Docker images can be built from a Docker file.

A Docker image registry is a storage area for Docker images. You can get images from them instead of building them.

3.What is Docker Swarm and which network driver should be used with it?

Docker Swarm is an open-source container orchestration tool that is integrated with the Docker engine and CLI. If you want to use Docker Swarm, you should use the overlay network driver. Using an overlay network enables the Swarm service by connecting multiple docker host daemons together.

4. Is there any problem with just using the latest tag in a container orchestration environment? What is considered best practice for image tagging?

If you’re running your image via the latest tag with a container orchestration environment like Kubernetes, it may cause a problem.

The problem is if you push a new image with just the latest tag, you lose your old image and your deployments will use the new image. If the new image has any problem, your deployments might fail, resulting in downtime.

5. **What is Docker Compose? What can it be used for?**

Docker Compose is a tool that lets you define multiple containers and their configurations via a YAML or JSON file.

The most common use for Docker Compose is when your application has one or more dependencies, e.g., MySQL or Redis

Once set up, you can bring all of these containers/dependencies up and running with a single docker-compose up command.

**1.What is Docker?**

Docker is a **containerization platform** that allows packaging an application with all its dependencies into a single entity called a container. Containers can be easily deployed and run on any machine that supports Docker. It provides a lightweight, portable solution for application deployment by isolating processes.

**2What are the features of Docker?**

Docker features include:

**Containerization**: Consistent deployment using lightweight containers.

**Resource Efficiency**: Efficient shared kernel utilization.

**Portability**: Seamless movement across environments.

**Security**: Isolation of containers, versioning, and automated builds.

**Pre-built Images**: A rich collection of pre-built images for streamlined development and deployment.

1. **What is a Docker image?**
   * **Answer**: A Docker image is a **read-only template** containing an application and its dependencies. Images serve as the basis for creating containers.
2. **How does Docker differ from virtualization?**
   * **Answer**: Docker containers share the host OS kernel, making them more lightweight than virtual machines (VMs). VMs run full OS instances, while containers isolate processes within a single OS.
3. **What is Docker Compose?**
   * **Answer**: Docker Compose is a tool for defining and running multi-container applications. It uses a YAML file to configure services, networks, and volumes.
4. **What is Docker Swarm?**
   * **Answer**: Docker Swarm provides native functionality for orchestrating Docker containers in a cluster. It simplifies scaling, load balancing, and service discovery.
5. **Scenario**: You need to deploy a microservices-based application. Which Docker orchestration tool would you choose, and why?
   * **Answer**: I would choose **Kubernetes** because it offers robust features for managing microservices, including scaling, rolling updates, and self-healing.
6. **Scenario**: How would you ensure data persistence in Docker containers?
   * **Answer**: I’d use **Docker volumes** to persist data outside the container. Volumes allow data to survive container restarts or removals.

**Docker Interview Questions and Answers for Freshers**

**1. What is Docker?**

Docker is an open-source lightweight containerization technology. It has gained widespread popularity in the cloud and application packaging world. It allows you to automate the deployment of applications in lightweight and portable containers.

**2. What are the advantages of using Docker container?**

Here, are a major advantage of using [Docker](https://www.guru99.com/docker-tutorial.html).

* Offers an efficient and easy initial set up
* Allows you to describe your application lifecycle in detail
* Simple configuration and interacts with Docker Compose.
* Documentation provides every bit of information.

**3. What are the important features of Docker?**

Here are the essential features of Docker:

* Easy Modelling
* Version control
* Placement/Affinity
* Application Agility
* Developer Productivity
* Operational Efficiencies

04:34

35:04

**4. What are the main drawbacks of Docker?**

Some notable drawbacks of Docker are:

* Doesn’t provide a storage option
* Offer a poor monitoring option.
* No automatic rescheduling of inactive Nodes
* Complicated automatic horizontal scaling set up

**5. What is Docker image?**

The Docker image help to create Docker containers. You can create the Docker image with the build command. Due to this, it creates a container that starts when it begins to run. Every docker images are stored in the Docker registry.

**6. What is Docker Engine?**

Docker daemon or Docker engine represents the server. The docker daemon and the clients should be run on the same or remote host, which can communicate through command-line client binary and full [RESTful API](https://www.guru99.com/restful-web-services.html).

**7. Explain Registries**

There are two types of registry is

* Public Registry
* Private Registry

Docker’s public registry is called Docker hub, which allows you to store images privately. In Docker hub, you can store millions of images.

**8. What command should you run to see all running container in Docker?**

$ docker ps

**9. Write the command to stop the docker container**

$ sudo docker stop container name

**10. What is the command to run the image as a container?**

$ sudo docker run -i -t alpine /bin/bash

**Docker Interview Questions for Experienced Professionals**

**11. What are the common instruction in Docker file?**

The common instruction in Docker file are: FROM, LABEL, RUN, and CMD.

**12. What is memory-swap flag?**

Memory-swap is a modified flag that only has meaning if- memory is also set. Swap allows the container to write express memory requirements to disk when the container has exhausted all the RAM which is available to it.

**13. Explain Docker Swarm?**

Docker Swarm is native gathering for docker which helps you to a group of Docker hosts into a single and virtual docker host. It offers the standard docker application program interface.

**14. How can you monitor the docker in production environments?**

Docker states and Docker Events are used to monitoring docker in the production environment.

**15. What the states of Docker container?**

Important states of Docker container are:

* Running
* Paused
* Restarting
* Exited

**16. What is Docker hub?**

Docker hub is a cloud-based registry that which helps you to link to code repositories. It allows you to build, test, store your image in Docker cloud. You can also deploy the image to your host with the help of Docker hub.

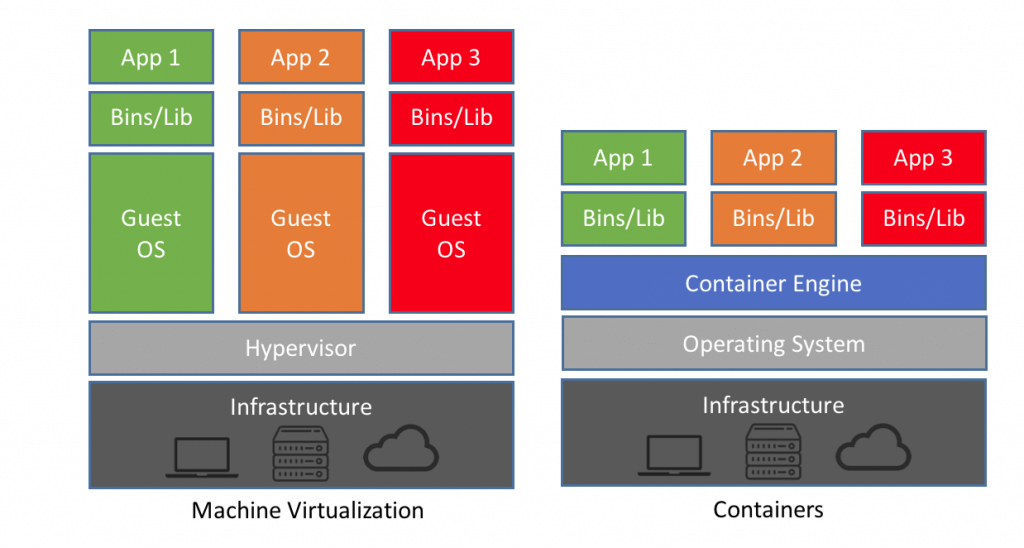
**17. What is Virtualization?**

Virtualization is a method of logically dividing mainframes to allow multiple applications to run simultaneously.

However, this scenario changed when companies and open source communities were able to offer a method of handling privileged instructions. It allows multiple OS to run simultaneously on a single x86 based system.

**18. What is Hypervisor?**

The hypervisor allows you to create a virtual environment in which the guest virtual machines operate. It controls the guest systems and checks if the resources are allocated to the guests as necessary.



Virtualization in Docker vs Hypervisor

**19. Explain Docker object labels**

Docker object labels is a method for applying metadata to docker objects including, images, containers, volumes, network, swam nodes, and services.

**20. Write a Docker file to create and copy a directory and built it using python modules?**

FROM pyhton:2.7-slim

WORKDIR /app

COPY . /app

docker build –tag

**21. Where the docker volumes are stored?**

You need to navigate:

/var/lib/docker/volumes

**22. List out some important advanced docker commands**

| **Command** | **Description** |
| --- | --- |
| docker info | Information Command |
| docker pull | Download an image |
| docker stats | Container information |
| Docker images | List of images downloaded |

**23. How does communication happen between Docker client and Docker Daemon?**

You can communicate between Docker client and Docker Daemon with the combination of Rest API, socket.IO, and TCP.

**24. Explain Implementation method of Continuous Integration(CI) and Continues Development (CD) in Docker?**

You need to do the following things:

* Runs Jenkins on docker
* You can run integration tests in Jenkins using docker-compose

**25. What are the command to control Docker with Systemd?**

systemctl start/stop docker

service docker start/stop

**26. How to use JSON instead of YAML compose file?**

docker-compose -f docker-compose.json up

**27. What is the command you need to give to push the new image to Docker registry?**

docker push myorg/img

**28. How to include code with copy/add or volumes?**

In docker file, we need to use COPY or ADD directive. This is useful to relocate code. However, we should use a volume if we want to make changes.

**29. Explain the process of scaling your Docker containers**

The Docker containers can be scaled to any level starting from a few hundred to even thousands or millions of containers. The only condition for this is that the containers need the memory and the OS at all times, and there should not be a constraint when the Docker is getting scaled.

**30. What is the method for creating a Docker container?**

You can use any of the specific Docker images for creating a Docker container using the below command.

docker run -t -i command name

This command not only creates the container but also start it for you.

**Docker Interview Questions for 5 Years Experience**

**31. What are the steps for the Docker container life cycle?**

Below are the steps for Docker life cycle:

* Build
* Pull
* Run

**32. How can you run multiple containers using a single service?**

By using docker-compose, you can run multiple containers using a single service. All docker-compose files uses yaml language.

**33. What is CNM?**

CNM stands for Container Networking Model. It is a standard or specification from Docker, Inc. that forms the basis of container networking in a Docker environment. This docker’s approach provides container networking with support for multiple network drivers.

**34. Does Docker offer support for IPV6?**

Yes, Docker provides support IPv6. IPv6 networking is supported only on Docker daemons runs on Linux hosts. However, if you want to enable IPv6 support in the Docker daemon, you need to modify /etc/docker/daemon. json and set the ipv6 key to true.

**35. Can you lose data when the container exits?**

No, any data that your application writes to disk get stored in container. The file system for the contain persists even after the container halts.

**36. What are a different kind of volume mount types available in Docker?**

Bind mounts- It can be stored anywhere on the host system

**37. How to configure the default logging driver under Docker?**

To configure the Docker daemon to default to a specific logging driver. You need to set the value of log-driver to the name of the logging drive the daemon. jason.fie.

**38. Explain Docker Trusted Registry?**

Docker Trusted Registry is the enterprise-grade image storage toll for Docker. You should install it after your firewall so that you can securely manage the Docker images you use in your applications.

**39. What are Docker Namespaces?**

The Namespace in Docker is a technique which offers isolated workspaces called the Container. Namespaces also offer a layer of isolation for the Docker containers.

**40. What are the three components of Docker Architecture**

* Client
* Docker-Host
* Registry

**41. What is client?**

Docker provides Command Line Interface tools to the client to interact with Docker daemon.

**42. What is the purpose of Docker\_Host?**

It contains container, images, and Docker daemon. It offers a complete environment to execute and run your application.

**43. How do I run multiple copies of Compose file on the same host?**

Compose uses the project name which allows you to create unique identifiers for all of a project’s containers and other resources. To run multiple copies of a project, set a custom project name using the -a command-line option or using COMPOSE\_PROJECT\_NAME environment variable.

## ntermediate Docker Interview Questions

### 15. Can You Tell the Difference Between CMD and ENTRYPOINT?

#### CMD vs ENTRYPOINT

* **CMD** is used for setting default commands and arguments that will be executed at the start of runtime of the containers. It is oftenly overridden by providing command-line arguments during container startup.
* **ENTRYPOINT** configures a container to run as an executable, defining the command that has to be executed when the container starts. It is more strict than CMD and is oftenly used when the container should have to behave like an executable.

Know more about [CMD vs ENTRYPOINT](https://www.geeksforgeeks.org/difference-between-run-vs-cmd-vs-entrypoint-docker-commands/).

### 16. What are the Key Distinctions Between Daemon Level Logging and Container Level Logging in Docker?

#### Daemon Level Logging vs. Container Level Logging

* **Daemon Level Logging:**It involves with configuring the Docker daemon’s logging behavior globally. This will affect all containers that are running on the host. Configuration settings will be applied at the daemon level.
* **Container Level Logging:** This will be connected to logs specific to an individual container. They can be accessed using the docker logs <container\_id> command , providing clear understanding regard the container’s runtime activities.

docker logs <container\_id>

### 17. What Does the Docker Info Command Do?

docker info provides detailed information regarding the Docker system. It includes information such as the number of containers, images, storage driver that are used and much more. It’s a valuable command for gaining details on overview of the Docker environment.

### 18. Where are Docker Volumes Stored in Docker?

Docker volumes will be stored on the host machine in thedirectory**/var/lib/docker/volumes** . This ensures persistance of the data storage even if the container is removed.

### 19. Can You Tell the Differences Between a Docker Image and Layer?

A Docker Image can be considered as a snapshot of a file system and application dependencies. It is composed of multiple layers, where each layer will represent a set of filesystem changes. These Layers facilitate in efficient image creation and sharing common components among the images.

### 20. Can a Container Restart By Itself?

Yes, a container itself can restart automatically by setting up the –restart option during the creation period of time. For example using **`docker run –restart**` always. This will ensure that the container restarts irrespective of its exit status.

docker run --restart

### 21. Name the Essential Docker Commands and What They Do.

The essential Docker Commands are listed here:

* **docker run:** [Docker run command](https://www.geeksforgeeks.org/difference-between-run-vs-cmd-vs-entrypoint-docker-commands/) is used to run the docker image as a docker container.
* **docker ps:** Docker os command will list all the running container in the docker.
* **docker exec:**It helps in execute the commands in a running container.
* **docker stop:**It will stops a container which is running in the docker.

### 22. What are Docker Object Labels?

Docker object labels are key-value mapping pair applied to the docker objects for better organizational and metadata purposes. For example, **`docker run –label environment=production <image\_name>`** adds a label to a container.

docker run --label environment=production <image\_name>

### 23. How Do You Check the Versions of Docker Client and Server?

Use `**docker version`**command to obtain the detailed information about the Docker client and server, including their respective version numbers.

docker version

### 24. Why is Docker System Prune Used? What Does It Do?

**`**[**docker system prune**](https://www.geeksforgeeks.org/docker-prune/)**`**is used for removal of unused data on inclusion of stopped containers, [docker networks,](https://www.geeksforgeeks.org/basics-of-docker-networking/) and [dangling images.](https://www.geeksforgeeks.org/remove-all-containers-and-images-in-docker/) It helps in freeing up the disk space on cleaning unnecessary resources.

docker system prune

### 25. What is Docker Swarm?

[Docker Swarm](https://www.geeksforgeeks.org/introduction-to-docker-swarm-mode/) is an inherented native clustering that comes up with a orchestration solution for the Docker software. It helps in simplifying the management of a swarm of Docker nodes on allowing the seamless scaling of the applications across various multiple nodes within the network. It provides built-in load balancing and will ensure the high availability of containerized applications.

### 26. How Do You Scale Docker Containers Horizontally?

Horizontal scaling is achieved through replicating the services across multiple nodes. Tools like [Docker Compose](https://www.geeksforgeeks.org/docker-compose/) or Docker Swarm facilitate this process. For example, using **`docker-compose up –scale web=3`** command will replicates the **“web”** service to three instances, distributing the workload across them horizontally.

docker-compose up --scale web=3

### 27. What Is the Difference Between Docker Restart Policies “no”, “on-failure,” And “always”?

These restart policies will provide flexibility in managing the container behavior based on specific requirements. The restart policy**“no”**gives full control over restarts, **“on-failure”**handles irregular issues, and **“always”**will ensures the constant availability. Choose the appropriate policy based over the nature and importance of that particular containerized application.

| **Restart Policy** | **Description** |
| --- | --- |
| **“no”** | No automatic restart will be done. The container will not restart automatically, in any case of the exit status. It will be suitable for the scenarios where manual intervention is preferred or when the container is perfered for a one-time execution.  Example: **docker run –restart no my\_container** |
| **“on-failure”** | It will restarts the container only if it exited with a non-zero status. This policy is useful when anticipating occasional failures and wanting the container to automatically recover from such failures. Example: **docker run –restart on-failure:3 my\_container** ( restarts up to 3 times on failure ) |
| **“always”** | The container will restart regardless of its exit status. It will be useful in critical services that should be always running and ensuring continuous operation even if the container exits. Example: **docker run –restart always my\_container** |

### 28. How Do You Inspect the Metadata of a Docker Image?

By using the **`docker inspect <image\_name>`** command , you can examine into detailed metadata about the Docker image. This contains the information regarding labels, layers, and the configuration settings.

docker inspect <image\_name>

### 29. How Do You Limit the CPU and Memory Usage of a Docker Container?

Om using the **–cpus** option you can set the CPU limitsand with **-m**option you can set memory limits**.**The following example illustrates usuage of CPU and memory for a docker container.

docker run --cpus=3 -m 1024M <image\_name>

### 30. What are the Differences ****B****etween Docker Community Edition (CE) and Docker Enterprise Edition (EE)?

Usage of Docker Community Edition will be peferable for individuals and small-scale projects, It provides the essential features of containerization for free. On the other hand, Docker Enterprise Edition deals in providing the enterprise needs with advanced features and support for the large-scale projects in production environments. The choice between these two will depends on the scale, requirements, and support needed for the Docker deployment.

| **Feature** | **Docker Community Edition (CE)** | **Docker Enterprise Edition (EE)** |
| --- | --- | --- |
| **Pricing** | Free for individual use and will be suitable for development and testing. | It requires the subscription and offers advanced features for the production environments. |
| **Support** | Provides community support through forums and community resources | Provides enterprise-grade level support through service-level agreements (SLAs). |
| **Certification** | Limited certification for specific platforms. | These are Certified and tested for a wide range of operating systems, cloud providers, and plugins. |
| **Security** | Contains Basic security features | Enhanced security with additional features like image signing and scanning are available. |
| **Orchestration Tools** | Has only Basic orchestration capabilities | Advanced orchestration tools like Docker Swarm and Kubernetes for large-scale deployments are available. |
| **Image and Container Management** | Core image and container management features. | Additional management tools and features, including role-based access control (RBAC). |
| **Environment Support** | Ideal for development and small-scale deployments. | Tailored for large-scale enterprise environments with optimized performance. |
| **Networking** | Basic networking capabilities. | Advanced networking features, including multi-host networking and DNS. |
| **Plugins and Extensions** | Plugins and extensions are limited in the community edition. | A wide range support of certified plugins and extensions are available for various integrations. |

## Advanced Docker Interview Questions

### 31. What Is the Purpose of the “docker checkpoint” Command?

The**“docker checkpoint”** command is vital for the creation of snapshots of a running container’s state , including its file system and the memory. It is particularly useful for experimental mode of scenarios such as debugging or migration.  
For example to checkpoint a container named “my\_container,” the command would be:

docker checkpoint create my\_container checkpoint\_name

### 32. Can We Use JSON Instead Of YAML While Developing a Docker-Compose File in Docker?

Yes, Docker Compose has support for both YAML and JSON formats for defining the configuration of services. While YAML is more commonly used due to its readability and clearness , you can also use JSON as an alternative. The choice between of two will depends on personal preference on requirements of the projects. To use JSON, simply try on creating a **`docker-compose.json`**file instead of a **`docker-compose.yml`** file, and define your services in JSON format.

### 33. Describe the Lifecycle of a Docker Container.

In the lifecycle of docker container , it goes through the following states:

* **Creation:** On using `**docker run`** container is created.
* **Execution:**In this state specific required commands are executed inside the container.
* **Pausing/Unpausing:** These are optional states for temporarily halting a container.
* **Stopping:**On using the `**docker stop`** command container is gracefully halted in this state.
* **Removal:**Using `**docker rm`**command the container can be deleted.
* **Restarting:**With the `**docker restart`** command containers can be restarted.

### 34. How Will You Ensure Container 1 Runs Before Container 2 While Using docker-compose?

In Docker Compose, the order of the services startup is determined by their dependencies. By specifying container dependencies with the**“depends\_on”**key in the docker-compose.yml file, you can ensure the desired startup order.  
A sample example on usage of depends\_on provided here , In this even though container1 is listed firsted because of the **depends\_on** key container2 will be startup and then container1 will be queue order.

services:

container1:

depends\_on:

- container2

...

container2:

....

### 35. How Does Docker Handle Container Isolation and Security?

Docker uses containerization concept to isolate the processes by imiting their access to the host system. Features like namespaces and cgroups provides the resource isolation for the containers and Docker Security Scanning helps in identifying the vulnerabilities in images.

### 36. How do the Docker Daemon and the Docker Client Communicate With Each Other?

The Docker daemon and client communicate on using REST APIs. The [Docker client](https://www.geeksforgeeks.org/using-cli-to-manage-docker-volumes/)will send the commands to the daemon using the API, and the daemon will execute those commands on managing containers, images, and other Docker objects.

### 37. Can You Implement Continuous Development (CD) and Continuous Integration (CI) in Docker?

Yes, Docker is an integral part to the [CI/CD pipelines](https://www.geeksforgeeks.org/what-is-ci-cd/). Developers can use Docker images for the consistent environments, and CI tools can perform the automate testing and deployment using Docker containers for ensure reproducibility.

### 38. Is it a Good Practice to Run Stateful Applications on Docker?

Docker is primarily designed for the stateless applications. On using Docker volumes or persistent storage stateful applications can be runnable but it’s crucial to carefully manage data persistence and backup to avoid data loss.

### 39. What Is the Purpose of Docker Secrets?

Docker secrets are used mostly to securely store sensitive information, such as passwords or API keys in Docker swarm. Secrets are encrypted and can only be accessible by services that have explicit permission to use them.  
Example:

docker secret create db\_password mysecretpassword

### 40. How Do You Create a Multi-stage Build in Docker?

A multi-stage build in Docker involves with using multiple “FROM” instructions in a Dockerfile. Each “FROM” instruction will begin a new stage, allowing you to build and copy the artifacts from previous stages for reducing the final image size.  
Example of Dockerfile with multi-stage build:

FROM builder as build

# Build stage

FROM alpine

# Final stage

COPY --from=build /app /app

### 41. How Do You Update a Docker Container Without Losing Data?

To update a Docker container without losing data, you can try on using a combination of Docker volumes or bind mounts to make the data persistant outside the container. When updating, create a new container with the updated image and then link it to the existing data volume.

### 42. How Do You Manage Network Connectivity Between Docker Containers And the Host Machine?

Docker provides several ways to manage network connectivity between containers and the host machine. The choice of networking options depends on the specific requirements of the application and the desired level of isolation.

* **Bridge Networks:** Bridge networks are the default networks , these are created when a Docker daemon starts. Through this network containers on the same bridge network can communicate with each other.
* **Host Networks:** In this containers will share the host’s network namespace. so that containers can directly use the host machine’s network interfaces.
* **Custom Networks:** Custom bridge networks can used to create isolate containers and control their communication. Containers on custom networks can communicate with each other with the help of container names as hostnames.
* **Overlay Networks (Swarm Mode):**Overlay networks are used in Docker Swarm mode for communication between the services that running on different nodes. They provide multi-host networking for orchestration of containers.
* **Macvlan Networks:** Macvlan networks will allow containers to have their own MAC addresses on the physical network, providing them with the direct access to the host’s network.

Examples for creating the networks are listed here:

# Create a bridge network

docker network create my\_bridge\_network

# To Run a container with host network

docker run --name container1 --network host -d my\_image

# To create a custom bridge network

docker network create my\_custom\_network

# To create an overlay network

docker network create --driver overlay my\_overlay\_network

### 43. How Do You Debug Issues in a Docker Container?

Debugging techniques will provide a comprehensive approach for troubleshoot and to the resolve issues within Docker containers. Depending on the nature of the problem on following these guided commands you can understands the details of the container’s behavior.

* **Container Logs:**On running this **`docker logs <container\_id>`**command you can view the standard output and error logs.

docker logs <container\_id>

* **Interactive Shell:** Helps in accessing the interactive shell inside the container for detailing.

docker exec -it <container\_id> /bin/bash

* **Inspect Container Details:** helps in retrieving the detailed information about the container.

docker inspect <container\_id>

* **Process Listing:**Provide the list of running processes inside the container.

docker exec -it <container\_id> ps aux

* **Network Troubleshooting:**Check network connectivity of the container through the hostname

docker exec -it <container\_id> ping <hostname>

* **Resource Utilization:** Helps in monitoring the CPU and memory usage.

docker stats <container\_id>

### 44. How Does Docker Handle Service Discovery in Swarm Mode?

In Docker Swarm Mode, service discovery is automatically handled through maintaining an internal [DNS service](https://www.geeksforgeeks.org/domain-name-system-dns-in-application-layer/) that automatically assigns DNS names to the containers on enabling easy service discovery within the swarm.

## Scenario-Based Docker Interview Questions

### 45. Which Is the Preferred Method for Removing a Container: Using the “docker stop” Command Followed By the “docker rm” Command, or Simply Using the “docker rm” Command By Itself?

The recommend approach for the container removal is to use the combined**“docker sto**p” and **“docker rm”**commands, as it makes sure a safely stopping of the container before removing it. This two-step process will helpful in avoiding potential issues related to the active processes within the container. However, if you are sure about that the container is not running, then you can go for using the “docker rm” command alone to remove it.

### 46. Suppose You Have 3 Containers Running, and Out Of These, You Wish to Access One Of Them. How Do You Access a Running Container?

To access the running container, you can use the **“docker exec**” command. Here is the general syntax:

docker exec -it <container\_id\_or\_name> /bin/bash

The `[**docker exec**](https://www.geeksforgeeks.org/mastering-docker-exec/)**`**command is useful for the execute of a command inside a running container. It provides a interactive session for pseudo terminal **-TTY** with option -it, allowing you to interact with the container **<container\_id\_or\_name>.** Try on replacing this with the actual ID or name of the container that you want to access. The /bin/bash program specifies the command that to be executed in the container. In this case, it starts with interactive Bash shell, but you can also replace it with the appropriate required command for your needs. This will opens up the bash shell inside the specified container on enabling you to run commands, inspecting the container’s filesystem, or troubleshoot. After once you are done with it, you can exit from the shell and it won’t affect the container’s running state.

### 47. Considering a Server With 16 GB RAM and a Quad-core CPU, What Factors Determine the Maximum Number Of Containers You Can Run on the Host for a Microservices App?

The maximum number of containers a host can support will mainly depends on the available resources like RAM and CPU cores. With the 16 GB of RAM, on assuming each container utilizes 512 MB efficiently, you could potentially run on around nearly 32 containers. However, cautious allocation, monitoring tools, and the container orchestration are essential factors for optimal resource utilization and scalability.

### 48. How Will You Monitor Docker in Production?

For monitoring the docker in production environment, generally utilize tools like Docker Stats, cAdvisor, and the [Prometheus](https://www.geeksforgeeks.org/working-with-prometheus-and-grafana-using-helm/) for real-time insights retriving for container performance. Implementation of centralized logging with solutions such as [ELK Stack](https://www.geeksforgeeks.org/what-is-elastic-stack-and-elasticsearch/)or [Splunk](https://www.geeksforgeeks.org/difference-between-hadoop-and-splunk/) to track container logs are used in common. Additionally, consideration of using container orchestration platforms like [Kubernetes](https://www.geeksforgeeks.org/introduction-to-kubernetes-k8s/) or Docker Swarm, which offer better built-in monitoring and scaling capabilities. Regularly review metrics such as [CPU](https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/), memory usage, and network activity to ensure optimal performance and address potential issues promptly.

### 49. Are You Aware of Load Balancing Across Containers And Hosts? How Does It Work?

[Load balancing](https://www.geeksforgeeks.org/load-balancing-in-cloud-computing/) across the containers and hosts is critical for distributing traffic efficiently in the containerized environment. Container orchestration tools like Kubernetes or Docker Swarm employes load balancers to evenly distribute the requests among the container instances or nodes. This will enhance the scalability, fault tolerance, and resource utilization by directing the traffic to healthy containers or hosts. The Load balancing plays a vital role in maintaining the stability and optimization on overall system performance in both the dynamic and scalable containerized applications.

### 50. How Do You Share Data Between Containers in Docker?

In Docker, you can share data between the containers on using **volumes** or by utilizing the `**–volumes-from`** option. Volumes will provide a persistent and the shared storage mechanism, allowing the data to be accessed and modified by multiple containers. Alternatively, the**–volumes-from** option allows the container to access the volumes of another container. This provides the seamless data sharing and collaboration between the containers, through facilitating the communication and coordination in complex multi-container setups.

### 51. Can a Container Restart By Itself?

Yes, The containers can be configured to restart by themselves automatically. Docker provides a restart policy that allowing you to define the container behavior when it exits. On using the options such as **“–restart always”**.  
you can instruct Docker to restart the container automatically, in any case of the exit status. This will be useful for ensuring the continuous availability of critical services within the containerized environment. Other restart policies include **“unless-stopped”** and **“on-failure”** offering the flexibility in handling the container lifecycle events.

### 52. How Do You Perform a Live Migration Of Docker Containers Between Hosts?

For performing a live migration of docker containers between hosts can be achieved through using container orchestration tools like Docker Swarm or Kubernetes. These tools will manage the seamless containers movements across the hosts on ensuring minimal downtime. Through Utilizing the features such as Swarm’s **`docker service update`** or Kubernetes’ **`kubectl drain`** and **`kubectl uncordon`** commands, you can initiate the live migrations by allowing containers to be moved to different hosts while maintaining the availability of application.

## Conclusion

In this Article , we have  gone through covering most of the Docker Interview Questions. We discussed the interview questions in classifying them into sections based on the complexity level and depth of Docker concepts. Beginning from the fundamentals and diving into Intermediate level, Advanced and Scenario based Concepts, to keep your understanding with the flow while moving to the complexity level. In the field of DevOps, The role of the container platform tool has been crucial in providing lightweight Platforms for deploying and encapsulating the applications. Docker brought creativity and Agility in the Software Development Life Cycle by laying a step for DevOps culture.

Once you get a thorough understanding of Docker with the preparation of these Interview Questions, you can move forward with confidence for expose insights of learning with effective DevOps practices. This article is more enough your docker in-depth preparation for giving Interviews on the Docker tool.

Lost in the complex landscape of DevOps? It's time to find your way! Enroll in our [DevOps Engineering Planning to Production](https://www.geeksforgeeks.org/courses/devops-live?utm_source=geeksforgeeks&utm_medium=article_bottom_text&utm_campaign=courses) Live Course and set out on an exhilarating expedition to conquer DevOps methodologies with precision and timeliness.  
**What We Offer:**

* Comprehensive DevOps Curriculum
* Expert Guidance for Streamlined Learning
* Hands-on Experience with Real-world Scenarios
* Proven Track Record with 100,000+ Successful DevOps Enthusiasts
* **1. What is Docker and how does it differ from a virtual machine?**
* Answer: Docker is a containerization platform that allows you to package applications and their dependencies into a single unit called a container. Unlike virtual machines, containers share the host OS kernel and are more lightweight, efficient, and portable.
* **2. What is a Docker image?**
* Answer: A Docker image is a lightweight, standalone, and executable software package that includes everything needed to run a piece of software, including the code, runtime, libraries, and system tools.
* **3. How do you create a Docker image?**
* Answer: A Docker image is created using a Dockerfile, which contains instructions to build the image. You use the docker build command to create an image from a Dockerfile.
* **4. What is Docker Hub?**
* Answer: Docker Hub is a cloud-based repository where you can find and share Docker images. It allows you to store and distribute Docker images for easy access and deployment.
* **5. How do you run a Docker container from an image?**
* Answer: You use the docker run command followed by the name of the image to run a container. For example, docker run -it ubuntu will run an interactive Ubuntu container.
* **6. Explain the concept of a Docker container lifecycle.**
* Answer: A Docker container's lifecycle involves creating, starting, stopping, restarting, and deleting containers. Docker manages the entire lifecycle, ensuring consistency and portability.
* **7. How do you link containers in Docker?**
* Answer: In modern Docker networking, you use user-defined networks to connect containers. The docker network create command creates a network, and containers can be added to that network using the --network flag.
* **8. What is Docker Compose?**
* Answer: Docker Compose is a tool for defining and running multi-container Docker applications. It uses a YAML file to configure the services, networks, and volumes required for an application.
* **9. How do you scale Docker containers in a cluster?**
* Answer: Docker Swarm and Kubernetes are popular tools for container orchestration. With Docker Swarm, you can use the docker service scale command to scale services up or down based on demand.
* **10. What is Kubernetes and how does it relate to Docker?**
* Answer: Kubernetes is an open-source container orchestration platform that automates the deployment, scaling, and management of containerized applications. While Docker provides the containers, Kubernetes manages their deployment and scaling.
* **11. What are Docker volumes?**
* Answer: Docker volumes are used to persist data generated by and used by Docker containers. They allow data to be stored outside the container filesystem and survive container restarts.
* **12. How can you ensure security in Docker containers?**
* Answer: Security in Docker containers involves using official images, regularly updating containers, avoiding unnecessary privileges, and restricting network access to minimize potential vulnerabilities.
* **13. Explain the difference between Docker and Docker Swarm.**
* Answer: Docker is a platform for containerization, while Docker Swarm is a native clustering and orchestration solution that simplifies the deployment and management of Docker services in a swarm mode.
* **14. How do you troubleshoot container-related issues?**
* Answer: Troubleshooting container issues involves checking container logs, inspecting the container's environment, analyzing resource usage, and identifying any misconfigured settings.
* **15. Can you explain the concept of containerization in software development?**
* Answer: Containerization is a method of packaging applications and their dependencies into isolated environments called containers. This ensures consistent behavior across different environments, from development to production.
* **16. What is Docker Registry?**
* Answer: A Docker Registry is a service that stores and manages Docker images. Docker Hub is a public registry, but organizations can also set up private registries to store proprietary images.
* **17. How does Docker help in microservices architecture?**
* Answer: Docker enables microservices architecture by encapsulating each microservice within its own container. This isolation simplifies deployment, scaling, and management of microservices.
* **18. How do you update a Docker image?**
* Answer: To update a Docker image, you update the Dockerfile with the necessary changes, rebuild the image using the docker build command, and then recreate the containers based on the updated image.
* **19. What is the purpose of the Dockerfile?**
* Answer: A Dockerfile is a text file containing instructions to build a Docker image. It defines the base image, environment variables, commands, and settings required to create a container image.
* **20. How can you share a Docker image with others?**
* Answer: You can share a Docker image by pushing it to a repository on Docker Hub or a private registry. Others can then pull the image from the repository using the docker pull command.
* **21. Explain the concept of Docker layering.**
* Answer: Docker uses a layered filesystem to create images efficiently. Each instruction in a Dockerfile creates a new layer, allowing for incremental changes and efficient image caching.
* **22. How do you manage environment variables within Docker containers?**
* Answer: You can set environment variables in a Docker container using the -e flag with the docker run command or by specifying them in a Dockerfile.
* **23. What is the difference between a Docker container and an image?**
* Answer: An image is a blueprint for creating containers, while a container is an instance of an image running as a process. Containers are the runnable instances created from images.
* **24. How do you clean up unused Docker resources?**
* Answer: To clean up unused Docker resources, you can use commands like docker system prune to remove stopped containers, unused images, and networks.
* **25. Can you explain the concept of Docker networking?**
* Answer: Docker networking allows containers to communicate with each other or with external networks. Docker provides various network drivers to create and manage networks.
* **26. How do you ensure high availability for Docker containers in production environments?**
* Answer: Docker Swarm and Kubernetes offer features for high availability, such as automatic container recovery, load balancing, and scaling based on demand.
* **27. Describe the use case for Docker volumes versus Docker bind mounts.**
* Answer: Docker volumes are managed by Docker and are more portable, while bind mounts allow you to directly access files from the host system. Volumes are preferred for most use cases.
* **28. What is the significance of the ENTRYPOINT and CMD instructions in a Dockerfile?**
* Answer: The ENTRYPOINT specifies the command that will be executed when the container starts. The CMD provides default arguments for the ENTRYPOINT.
* **29. How do you upgrade Docker Compose services without downtime?**
* Answer: To upgrade Docker Compose services without downtime, you can use the docker-compose up command with the --scale flag to increase the desired number of instances gradually.
* **30. What is the purpose of health checks in Docker containers?**
* Answer: Health checks monitor the status of a container's processes and determine whether the container is healthy or not. They help in automatic recovery and load balancing.
* **31. Explain how you can configure network communication between containers and the host system.**
* Answer: By default, containers can communicate with the host using the special IP address host.docker.internal. Host-to-container communication can be achieved using port mapping.
* **32. How do you manage container logs in Docker?**
* Answer: Docker captures container logs, which can be accessed using the docker logs command. Logs can also be configured to output to files or external logging systems.
* **33. What is the purpose of Docker Healthchecks?**
* Answer: Docker Healthchecks are commands used to determine the health status of a container. They help ensure that only healthy containers are used in services.
* **34. How do you ensure that Docker containers are running as non-root users for security reasons?**
* Answer: You can specify a non-root user using the USER directive in the Dockerfile, which ensures that the container runs with reduced privileges.
* **35. Explain the concept of Docker Orchestration.**
* Answer: Docker Orchestration refers to the automated management, deployment, scaling, and coordination of containerized applications using tools like Docker Swarm or Kubernetes.
* **36. What is the purpose of the docker-compose.yml file in Docker Compose?**
* Answer: The docker-compose.yml file defines the services, networks, and volumes required for a Docker Compose application. It allows you to define a multi-container application configuration.
* **37. How do you handle data persistence in Docker containers?**
* Answer: Data persistence can be achieved using Docker volumes or bind mounts to store data outside of the container filesystem. Volumes are recommended for data persistence.
* **38. Can you explain the difference between a container and an instance?**
* Answer: A container is a runtime instance of an image, while an instance is a general term referring to a single execution of a program, which can include containers.
* **39. How can you ensure that Docker containers are automatically started when the host system restarts?**
* Answer: You can use the --restart flag with the docker run command to specify the restart policy for containers, ensuring they start automatically on system restart.
* **40. Explain the concept of container port mapping in Docker.**
* Answer: Container port mapping allows you to expose container ports to the host system, enabling external access to services running within the container.
* **41. How do you upgrade a Docker image used by multiple containers in a service?**
* Answer: You can update a Docker image for a service in a swarm using the docker service update command, specifying the new image version.
* **42. What is the main difference between Docker and other virtualization technologies?**
* Answer: Docker uses containerization, where multiple containers share the host OS kernel, resulting in smaller resource overhead and improved performance compared to traditional virtualization.
* **43. How can you limit the resources (CPU and memory) used by a Docker container?**
* Answer: You can limit resources using the --cpu and --memory flags with the docker run command or by specifying resource limits in a docker-compose.yml file.
* **44. Explain the role of Docker labels in container management.**
* Answer: Docker labels provide metadata to containers, helping in categorization, organization, and management of containers. Labels can be used for various purposes, including monitoring and automation.
* **45. What is the difference between Docker Swarm and Kubernetes for container orchestration?**
* Answer: Docker Swarm is simpler to set up and use, making it suitable for smaller deployments. Kubernetes is more feature-rich and designed for managing complex containerized applications at scale.
* **46. How do you manage secrets in Docker Swarm services?**
* Answer: Docker Swarm secrets allow secure storage and sharing of sensitive information, such as passwords, API keys, and certificates, among services in a swarm.
* **47. Explain the concept of Docker context.**
* Answer: A Docker context is a way to manage different Docker environments and target remote Docker hosts. It lets you switch between different contexts to manage containers on different hosts.
* **48. How can you troubleshoot connectivity issues between Docker containers in a network?**
* Answer: You can use commands like docker network inspect and docker exec to investigate networking settings and connectivity within containers.
* **49. Can you explain the concept of a multi-stage Docker build?**
* Answer: A multi-stage Docker build involves using multiple FROM instructions in a single Dockerfile to create intermediate images. This helps in building efficient and smaller final images.
* **50. What are Docker plugins and how do they extend Docker's functionality?**
* Answer: Docker plugins are external tools that extend Docker's functionality. They allow you to integrate external services, storage drivers, and network drivers into Docker.
* These Docker & Container interview questions and answers cover a wide range of topics related to containerization, Docker, and container management. Customize your responses based on your experience and the specific requirements of the role you're interviewing for.

-\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Links :\_ [[2023] Top 50 Docker & Container Interview Questions and Answers - Web Asha](https://www.webasha.com/blog/top-docker-container-interview-questions-answers)

[Top 50+ Docker Interview Questions & Answers (2024) (geeksforgeeks.org)](https://www.geeksforgeeks.org/docker-interview-questions/)

[Top 40 Docker interview questions (with answers) (educative.io)](https://www.educative.io/blog/top-40-docker-interview-questions)

[100+ Docker Interview Questions and Answers 2024 - Turing](https://www.turing.com/interview-questions/docker)

[100 Latest Azure Interview Questions and Answers 2024 - Turing](https://www.turing.com/interview-questions/azure)