70 Real Time Docker Interview Questions and Answers

November 30, 2023 by FOSS TechNix

In this article, We are going to cover Real time Docker Interview questions and answers for Fresher and Experienced candidate, Scenario Based Docker Interview Questions and Answers, Docker Troubleshooting Interview Questions and Answers.

We have friends who are working in DevOps,We have contacted them and collected below Practical docker interview questions and their answers which can be asked in interview.

Docker Interview Questions and Answers

1. What according to you is Docker?

A Docker can be a platform designed to ensure excellent efficiency and availability by containerizing the application and segregating them from each other for environments like production, testing or development.

Docker is a platform that allows developers to create, deploy, and run applications inside containers.

What is Container ?

A container is a lightweight, stand-alone, executable software 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 the host system.

2. Define Docker hub or Docker Hub Registry?

Docker hub is a cloud-based registry service that lets you attach, create and send your assets to application databases, save images that have been manually sent, and connects to Docker cloud so that you can you can deploy images to your hosts.

Docker hub provides a centralized resource for container image discovery, distribution and change management, user and team collaboration, workflow automation throughout the development pipeline.

3. What is command to pull docker image

Syntax

```
$ docker pull [OPTIONS] [PATH/]IMAGE_NAME[:TAG]
```

Example

```
$ docker pull ubuntu
```

4. How to pull docker image with specific version

```
$ docker pull ubuntu:18.04
```

5. How to pull the latest version of an image

```
$ docker pull ubuntu:latest
```

6. How to create a custom tag to docker image

```
$ docker tag ubuntu:latest fosstechnix/ubuntu:test
```

7. How to push docker image to docker hub registry using command line

```
$ docker push fosstechnix/ubuntu:demo
```

8. How to Remove all images in docker server

```
$ docker image rm -f <Image_id>
```

9. How to Pull your custom image from your docker account

```
$ docker pull fosstechnix/ubuntu:demo
```

10. How to pull Docker Image from Private Registry

First login to Private Registry

```
$ docker login myrepo.fosstechnix.com:9000
```

Then pull your images

Syntax:

```
$ docker pull [OPTIONS] ADDRESS:PORT[/PATH]/IMAGE_NAME[:TAG]
```

Example:

```
$ docker pull repo.fosstechnix.com:9000/demo-image
```

11. What do you mean by Docker container?

The program and all its components are included in Docker containers. The kernel is associated with several other containers and works in the host operating system as independent operations.

12. Tell me some about Hypervisor.

Hypervisor can also termed as the virtual machine monitor. This system basically divides the hosting system by delegating resources to each system equally. The possibility of turning virtualization into feasibility and practicality can be achieved only with Hypervisor.

13. Can you state the advantages of Docker over Hypervisor?

The prominent advantage of using Docker over Hypervisor is lightweight feature of the former one makes it highly feasible in terms of operations.

14. Define Docker images.

Docker images are required for establishing Docker containers. These images can be used for linking to any of the Docking environment.

15. How do you estimate the Docker server version and the client?

```
$ docker version
```

command will give the requisite information regarding the Docker client and server version.

16. Tell me something about Docker machine.

Docker machines are basically used for clustering of Docker swarms. Moreover, they are used for installation of Docker engines on the virtual hosting facilities, which can be managed using the commands from Docker machines.

17. Define Docker Swarms.

Docker swarms functions mainly by splitting a collection of Docker hosts into individual and virtual hosting spaces. As a result, the maintenance turns very easy and feasible in its operations.

18. Describe the usage of Dockerfile?

A Dockerfile is a series of specific instructions something we must send Docker to create the files. We can see the Dockerfile as a word document that has all the required commands to build a Docker Image.

OR

A **Dockerfile** is a text document that contains all the commands a user could call on the command line to assemble an image.

Below is workflow to create Docker Container from Dockerfile

Dockerfile –> Docker Image –> Docker Container

19. Can you please write Dockerfile to create Ubuntu Docker Image

```
FROM ubuntu:18.04
MAINTAINER FOSS TECHNix support@fosstechnix.com
LABEL version="1.0" \
RUN apt-get update && apt-get install -y apache2 && apt-get clean
ENV APACHE_RUN_USER www-data
ENV APACHE_RUN_GROUP www-data
ENV APACHE_LOG_DIR /var/log/apache2
EXPOSE 80
COPY index.html /var/www/html
CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
```

20. Can you please write Dockerfile to create Node Js Docker Image

```
FROM node:10
RUN mkdir -p /home/nodejs/app
```

```
WORKDIR /home/nodejs/app
COPY package.json .
RUN npm install
COPY . .
CMD [“node.js”, “index.js”]
EXPOSE 3000
```

21. What is command to build docker image from Dockerfile ?

```
$ docker build -t image_name .
```

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

```
$ docker run -it ubuntu /bin/bash
```

Here **i** -> interactive, **t** -> terminal

23. What is command to list docker images

```
$ docker images
```

24. What is command to list specific docker image

```
$ docker image ls <imagename>
```

25. How to apply port mapping a running container?

We can apply port forwarding while running a container using below command.

```
$ docker run -p <public_port>:<private_port> -d <image>
```

26. What is command to login docker container

```
$ docker exec -it <container_Name> /bin/bash
```

27. What is command to stop a docker container

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

28. What is command to start a docker container

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

29. What is command to remove a Docker container

```
$ docker rm <container_id>
```

30. What is command to list out Running Docker containers

```
$ docker ps
```

31. What are the common instructions in Dockerfile

Below are some common instructions in Dockerfile

FROM, MAINTAINER, RUN, CMD, LABEL, EXPOSE, ENV, ADD, COPY, ENTRYPOINT, VOLUME, USER, WORKDIR, ARG, ONBUILD, STOPSIGNAL, SHELL, HEALTHCHECK

32. What is difference between ADD and COPY in Dockerfile

COPY : Copies a file or directory from your host to Docker image, It is used to simply copying files or directories into the build context.

Syntax:

```
COPY <source> <destination>
```

Example:

```
COPY index.html /var/www/html
```

ADD: Copies a file and directory from your host to Docker image, however can also fetch remote URLs, extract TAR/ZIP files, etc. It is used downloading remote resources, extracting TAR/ZIP files.

Syntax:

```
ADD <source> <destination>
```

Example:

```
ADD java/jdk-8u231-linux-x64.tar  /opt/jdk/
```

33. Detail us with some of the specific advantages of Docker over other containerization technologies.

- Different types of files can be downloaded from a central space along with the Docker hub.
- Docker also enables us to share our contents with our created containers.
- It can be assessed from the official IT systems or even from the personal computers also.

34. Enumerate the lifecycle process of Docker containers

- Establishing the containers.
- Using the container.
- Pausing
- Unpausing
- Halting the container.
- Restarting
- Destroying the container.

35. What are Docker Namespaces?

Docker Namespaces is a platform that provides a container known as independent workspaces. A variety of namespaces are generated for this database after a container is launched.

Such namespaces include a seclusion framework for the containers since each container operates in a different namespace with limited access to the namespace defined.

36. What is the EXPOSE command? What is its role?

```
EXPOSE <port> [<port>/<protocol>...]
```

During publishing of the ports, this command can be used for mapping of the documentations.

37. Why is it necessary to monitor a Docker?

Active monitoring ensures higher productivity and better outcomes. Docker monitoring also notifies about any default in the system.

38. Define Docker compose?

Docker Compose is a method for specifying various containers and their parameters in a YAML or JSON. Docker Compose more usually uses one or several dependencies, for instance MySQL or MongoDB, for your program.

This dependencies are usually set up locally throughout innovation — a process which then has to be reworked before going to a production set-up. You can use Docker Compose to prevent these deployment and configuration bits.

OR

Docker compose is used to Defining and Running multi container Docker Application, you can use JSON or YAML file to write docker compose

39. Can Please create Nodejs, MySQL, MongoDB docker environments using docker-compose file ?

```
version: "2"
services:
web:
build: .
ports:
- "3000:3000"
depends_on:
- mongo
- mysql
mongo:
image: mongo
volumes:
- my-datavolume:/var/lib/mongo
volumes:
my-datavolume:
ports:
- "27017:27017"
mysql:
env_file:
- mysql-server.env
```

```
image:mysql
volumes:
- my-datavolume:/var/lib/mysql
volumes:
my-datavolume:
ports:
- "3306:3306"
```

40. What is command to run Docker compose

```
$ docker-compose up
```

41. What is depends_on in Docker compose

It is used to Link to containers in another service and also express dependency between services.

42. Can We use JSON instead of YAML for my compose file in Docker

Yes. We can Use.

43. Describe the role of Docker load and save command.

Docker save command makes its possible to export a Docker image as an archive with command:

```
$ docker save -o <container-export-path>.tar <container-name>
```

44. While this exported image can be easily imported to other hosts using the load command.

```
$ docker load -i <container-path>.tar
```

45. Why and how to identify the status of a Docker container?

Identification of the status helps us to make any conduct to the Docker containers accordingly. Therefore, for identifying the same we need to run the following command.

```
$ docker ps —a
```

46. Which is the most feasible type of application for Docker containers – Stateless or Stateful?

It is best to generate a Stateless Docker container application. From our code we could build a container and remove programmable state parameters from framework. Now in development and in QA environments we can operate the same container with different parameters.

This allows to recreate the same image in various scenarios. A stateless framework is also much simpler than a superb program to scale with Docker Containers.

47. Differentiate between a Docker layer and a image.

The layers in a Docker represents the set of instructions from a Docker image, while the image is nothing but the set of read-only layers.

48. Say something about virtualization.

A way of logically separating mainframes is seen as virtualization what allows multiple software to run concurrently. The situation changed radically, though, as businesses and open source networks could provide a way to handle delegated instructions in one manner or another, allowing multiple operating systems to operate on a single x86-based machine concurrently.

49. What if you have accidentally out of the Docker containers, will you loose the files?

There is no way we can loose our progress in a Docker container unless we have implemented the deleting program in the container itself.

50. What are factors that decides the number of containers you can run on?

Factors such as app size and strength of the CPU can deliberately influence the count of Docker containers. Thus, it can be understood that if we are equipped goof CPU strength we can easily hell load of containers with extreme ease.

51. What is the difference between CMD and ENTRYPOINT in a Dockerfile?

CMD in Dockerfile Instruction is used to execute a command in Running container, There should be one **CMD** in a Dockerfile.

ENTRYPOINT in Dockerfile Instruction is used you to configure a container that you can run as an executable.

52. What is Dockerfile Instructions ?

Dockerfile contains a set of Instructions to build Docker Image -> from Docker Image -> running Docker container

We have covered docker interview questions and answers for fresher and experienced candidate.

53. What are Docker Lifecycle commands ?
Below are some Docker Lifecycle commands for every Docker Container

1. Docker create
2. docker run
3. docker pause
4. docker unpause
5. docker stop
6. docker start
7. docker restart
8. docker attach
9. docker wait
10. docker rm
11. docker kill

54. What is Docker Prune ?

Using Docker prune we can delete unused or dangling containers, Images , volumes and networks

Why is Docker Important for DevOps?

1. Consistency: containers package up all dependencies, an application will run the same regardless of where the container is run. This eliminates the "it works on my machine" problem.

2. Isolation: Containers ensure that applications run in isolation from each other.This means if one application crashes, it won't affect others.

3. Scalability: With Docker, it's easy to scale applications up or down, depending on the demand, by simply starting or stopping containers.

4. Portability: You can build a container on your local machine, then deploy it to various environments (e.g., Integration, staging, production) without changes.

5. Infrastructure Efficiency: Containers are lightweight compared to virtual machines, as they share the host system's OS, rather than needing their own operating system.

What are key components of Docker?

1. Docker Images: Docker uses images to create containers. An image is a lightweight, stand-alone, executable package that includes everything needed to run a piece of software.

2. Docker Containers: A container is a runtime instance of an image. It's the "live" version of the Docker image.

3. Dockerfile: A script with commands to build a Docker image. It defines how the image should be built, what software it should contain, and how it should run.

4. Docker Hub: A cloud-based registry where Docker users and partners create, test, store, and distribute container images.

5. Docker Compose: A tool for defining and running multi-container Docker applications.With Compose, you use a YAML file to define the services, networks, and volumes, and then start all services with a single command.

6. Docker Volumes: Used to store data outside of containers, ensuring that data persists even if the container is deleted.

Please Explain How Does Docker Work?

At a high level, Docker works by using a technology called containerization. Unlike traditional virtualization, where each application requires a separate operating system instance, containerization allows multiple containers to run on a single OS instance. The Docker engine, which is responsible for building and running containers, achieves this.

Explain What is difference between Docker vs. Traditional VMs?

Performance: Containers are lightweight and share the host OS, while VMs have their own full OS instance.

Size: VM images are often several GBs because they include a full OS. Docker images are much smaller as they only include the application and its dependencies.

Startup Time: Containers can start almost instantly, whereas VMs can take several minutes.

Advanced Docker Interview Questions and Answers

What is the difference between containerization and virtualization?

Containerization and virtualization are both technologies for isolating applications from each other. However, they work in different ways.

- **Virtualization** creates a virtual machine (VM) that is a complete replica of a physical machine, including the operating system, hardware, and applications. VMs are typically larger and slower than containers.

- **Containerization** packages an application and its dependencies into a lightweight container that shares the host operating system’s kernel. Containers are typically smaller and faster than VMs.

What are the benefits of using Docker?

There are many benefits to using Docker, including:

- **Isolation:** Docker containers isolate applications from each other, which can help to prevent conflicts and security vulnerabilities.

- **Portability:** Docker containers can be run on any machine with Docker installed, which makes them very portable.

- **Reproducibility:** Docker containers are always in the same state, which makes them very reproducible.

- **Scalability:** Docker containers can be easily scaled up or down, which makes them very scalable.

What are the components of Docker?

The main components of Docker are:

- **Docker image:** A Docker image is a read-only template that contains the instructions for creating a container.

- **Docker container:** A Docker container is an instance of a Docker image. It is a running instance of an application that is isolated from the host operating system.

- **Docker Hub:** Docker Hub is a public registry where Docker images can be stored and shared.

What is a Docker registry?

A Docker registry is a repository where Docker images are stored and shared. Docker Hub is a public Docker registry, but there are also many private registries.

What are some common Docker networking concepts?

Some common Docker networking concepts include:

- **Bridge network:** The default network for Docker containers. It connects containers to each other and to the host machine.

- **Overlay network:** A network that can span multiple Docker hosts.

- **External network:** A network that connects containers to external resources, such as the internet.

What are some security considerations for using Docker?

Some security considerations for using Docker include:

- **Image scanning:** Scanning Docker images for vulnerabilities before running them.

- **User isolation:** Using different user accounts for different Docker containers.

- **Network isolation:** Restricting network access to Docker containers.

How can you troubleshoot Docker problems?

Some common ways to troubleshoot Docker problems include:

- **Checking the Docker logs:** The Docker logs can provide valuable information about errors and warnings.

- **Inspecting Docker containers:** You can inspect Docker containers to get more information about their state and configuration.

- **Using the Docker CLI tools:** There are a number of Docker CLI tools that can be used to troubleshoot problems, such as `docker ps`, `docker logs`, and `docker info`.

Advanced Docker Interview Questions and Answers

What are some advanced Docker features?

Some advanced Docker features include:

- **Docker Swarm:** A tool for managing clusters of Docker hosts.

- **Docker Compose:** A tool for defining and running multi-container applications.

- **Dockerfiles with multiple stages:** A way to build Docker images with multiple stages, which can improve image efficiency.

- **Docker Secrets:** A way to store sensitive information securely in Docker containers.

- **Docker BuildKit:** A tool for building Docker images that can improve build performance.

How do you keep your Docker images up to date?

There are a number of ways to keep your Docker images up to date, including:

- **Using a continuous integration (CI) pipeline:** A CI pipeline can automatically build and test Docker images.

- **Using a vulnerability scanner:** A vulnerability scanner can scan Docker images for vulnerabilities.

Scenario Based Docker Interview Questions and Answers

Scenario 1: Deploying a web application

You are tasked with deploying a new web application to production. The application is a microservices-based architecture and consists of several different containers. How would you approach this task using Docker?

Answer:

1. **Create Dockerfiles for each microservice:** This will ensure that each microservice is always built in the same way and that the development environment is consistent with the production environment.

2. **Build Docker images for each microservice:** This will create a portable and reproducible image for each microservice that can be run on any machine with Docker installed.

3. **Store Docker images in a registry:** This will make it easy to deploy and manage the microservices in production.

4. **Use a container orchestration tool, such as Kubernetes, to manage the microservices in production:** This will make it easy to scale the microservices up or down and to manage their health and availability.

Scenario 2: Troubleshooting a Docker container

A Docker container is not starting. How would you troubleshoot this problem?

Answer:

1. **Check the Docker logs:** The Docker logs can provide valuable information about errors and warnings that may be preventing the container from starting.

2. **Inspect the Docker container:** You can inspect the Docker container to get more information about its state and configuration. This can help you to identify any problems with the container's configuration or with the application that is running in the container.

3. **Use the Docker CLI tools:** There are a number of Docker CLI tools that can be used to troubleshoot problems, such as `docker ps`, `docker logs`, and `docker info`.

Scenario 3: Securing Docker containers

What are some security considerations for using Docker?

Answer:

1. **Image scanning:** Scanning Docker images for vulnerabilities before running them.

2. **User isolation:** Using different user accounts for different Docker containers.

3. **Network isolation:** Restricting network access to Docker containers.

4. **Least privilege:** Running Docker containers with the least amount of privilege necessary.

5. **Regular updates:** Regularly updating Docker images and the Docker daemon to the latest version.

Scenario 4: Migrating to Docker

You are tasked with migrating a legacy application to Docker. What are some of the challenges that you may face?

Answer:

1. **Packaging the application:** The application may not be designed to be run in a container, so you may need to make changes to the application code to make it compatible with Docker.

2. **Managing dependencies:** The application may have a number of dependencies that need to be installed and managed in the Docker container.

3. **Testing the application:** You need to make sure that the application works correctly when running in a container.

4. **Deploying the application:** You need to decide how to deploy the application to production, such as using Docker Swarm or a container orchestration tool like Kubernetes.

Scenario 5: You have a multi-container application, and one of the containers is not communicating with the others. How would you troubleshoot and resolve the issue?

- **Answer:**
  - Check container logs: Use `docker logs <container_id>` to inspect the logs of the misbehaving container for any error messages.
  - Inspect network connectivity: Ensure containers can reach each other by checking network configurations (`docker network inspect`) and using tools like `ping` or `telnet` between container IP addresses.
  - Verify container dependencies: Ensure that dependencies specified in your application (e.g., environment variables, network configurations) are correctly set.
  - Update container configurations: If necessary, modify container configurations, such as environment variables or network settings.

Scenario 6: You want to scale a service horizontally by adding more instances of a container. How would you achieve this using Docker?

- **Answer:**
  - Use Docker Compose: Define the service configuration in a `docker-compose.yml` file and then use the `docker-compose up --scale <service_name>=<desired_instances>` command to scale the service.
  - Use Docker Swarm: If working with a swarm, deploy the service with `docker service create --replicas <desired_instances> <service_name>`.
  - Use Docker Compose with Swarm mode: If using both Docker Compose and Swarm, define the services in `docker-compose.yml` and deploy them to the swarm with `docker stack deploy`.

Scenario 7: Your Docker host is running out of disk space. How would you identify and clean up unused Docker resources?

- **Answer:**
  - Use `docker system df`: This command shows the disk usage of Docker components. Look for items consuming the most space.
  - Remove unused containers: `docker container prune` removes stopped containers.
  - Remove unused images: `docker image prune` removes dangling (untagged) images.
  - Remove unused volumes: `docker volume prune` removes unused volumes.
  - Remove unused networks: `docker network prune` removes unused networks.

Scenario 8: You need to deploy a multi-container application on a remote server. How would you securely manage the deployment process?

- **Answer:**
  - Use Docker Swarm or Kubernetes: These orchestration tools provide built-in security features for managing and deploying multi-container applications.
  - Use SSH for remote access: Connect to the remote server securely using SSH for deploying Docker containers manually.
  - Set up TLS for Docker: Configure Docker daemon to use TLS to encrypt communication between Docker clients and the daemon for a secure remote connection.

Scenario 9: You want to update a running container with a new version of the application without downtime. How can you achieve zero-downtime deployments in Docker?

- **Answer:**
  - Use Blue-Green Deployments: Start a new set of containers (Green) with the updated version, divert traffic to them, and then stop the old containers (Blue).
  - Use Rolling Deployments: Update one container at a time while maintaining a specified number of instances running.
  - Use Orchestration Tools: Tools like Docker Swarm or Kubernetes can automatically manage rolling updates with minimal downtime.

Scenario 10: You are experiencing performance issues with your Dockerized application. How would you profile and optimize the performance of Docker containers?

- **Answer:**
  - Use Docker Stats: Monitor container resource usage using `docker stats <container_id>`.
  - Profile Application: Use tools like `strace`, `top`, or `htop` inside the container to identify resource-intensive processes.
  - Optimize Dockerfile: Ensure your Dockerfile is efficient, avoiding unnecessary layers and optimizing dependencies.
  - Adjust Resource Limits: Use Docker Compose or Docker Swarm to set resource limits for individual containers.

Conclusion:

We have covered Real time Docker Interview questions and answers for Fresher and Experienced candidate, Scenario Based Docker Interview Questions and Answers, Docker Troubleshooting Interview Questions and Answers.