Docker Interview Questions

Question: Explain Docker Container?
Answer: A container is the basic unit of software that holds the code and all its dependencies, in order to make the application run smoothly, quickly and reliably from one computing ecosystem to another. A Docker container may be created using a Docker image. It is an executable package of the software, which holds everything that is required to run an application, which are system tools, libraries, code, runtime, and settings.

Question: Explain the components of Docker Architecture.
Answer: The components in Docker architecture are given below:

Host: This component holds the Docker Daemon, Images, and Containers. While the Docker Daemon establishes a link with the Registry, the Docker Images act as metadata for the applications which are held in the Docker Containers.

Client: The Docker Client component runs operations to set up communication with the Docker Host.

Registry: This Docker Component is used to store the Docker Images. Docker Hub and Docker Cloud are public registries, which can be utilized by anyone.

Question: Explain the Docker Registry in detail.

significant public storehouse of the image containers, consistently maintained by a large number of developers, along with many individual contributors.

Question: Briefly explain the Docker Container lifestyle.
Answer: The lifecycle of a Docker Container is:

- Creation of the container
- Running the container
- Pausing the container
- Unpausing the container
- Starting the container
- Stopping the container
- Restarting the container
- Killing the container
- Destroying the container

Question: Name some important Docker commands
Answer: Below is some important Docker commands:

- build: to build an image file for Docker
- create: for creation of a new container
- kill: to kill a container
- dockerd: for launching Docker daemon
- commit: for creating a new image from the container changes

Question: What are Namespaces in Docker.
Answer: Docker Namespaces is a technology providing isolated workspaces knows as a container. Once a container is started, a set of namespaces is created for the said container. These namespaces provide a layer of seclusion for these containers as each container functions in a distinct namespace, with its access limited to the mentioned namespace.

containers. Using Docker Swarm, developers and IT supervisors can easily establish and manage a bunch of nodes in Docker or a solitary Virtual System (VS).

Question: How to identify the status of a Docker Container?
Answer: To identify the status of a Docker container, one should run the command

"docker ps-a."

This command will furnish the list of all available Docker containers with the respective status on the host. From the list, one can easily make out the intended container to check its status.

Question: What are the Docker Image and Docker Run Command?
Answer: A Docker Image is a group of files and an amalgamation of parameters that allow the creation of instances that run in distinct containers as isolated processes. An image is basically built using the instructions for a complete and executable version of an application, which relies on the host OS kernel. The Docker run command can be used to create the instance called container which can be run using the Docker image. When the Docker user runs an image, it becomes one or multiple instances of that container.

Question: State the functionalities and applications of Docker.
Answer: Below is some functionalities and applications of implementing Docker:

- It makes the configuration simpler and provides ease of configuration at the infrastructure level
- By helping the developer concentrate exclusively on business logic, it reduces development time and increases productivity
- It amplifies the debugging capabilities which provide useful functionalities
- It allows the isolation of the application
- It reduces the use of multiple servers in the form of containerization
- It facilitates rapid deployment at the OS level

Answer: Docker Images, Services, and Containers are termed as Docker Objects.

- Images: A read-only template with instructions for creating a Docker container
- Containers: A runnable instance of an image
- Services: It allows the scaling of containers across a variety of Docker Daemons, which all work together as a swarm. Other Docker Objects include Networks and Volumes.

Question: Which is more suitable for Docker Container, Stateless or Stateful application?
Answer: Stateless applications should be preferred over a Stateful application for Docker Container. We can create one container from our application and take out the app's configurable state parameters. Once it is one, we can run the same container with different production parameters and other environments. Through the Stateless application, we can reuse the same image in distinct scenarios. It is also easier to scale a Stateless application than a Stateful application when it comes to Docker Containers.

Question: Explain the use of Dockerfile.
Answer: Dockerfile contains many instructions passed on Docker to make possible the build process of images, which can automatically read these instructions. It can also be termed as a text document containing all the possible commands that a user may call on the command line to create an image.

Question: Which networks all available by default in Docker?
Answer: The default networks available in Docker are:

- bridge: Default network which the containers connect to if the network is not specified otherwise
- none: Connects to a container-specific network stack lacking a network interface
- host: Connects to the host's network stack

Question: List the steps in a deploy process for Dockerized Apps stored In A Git Repo.
Answer: While the deploy process changes with your production environment, a

- Build an application through Docker Build located in the code directory

- Perform the test of an image
- Push the new image to registry docker
- Notify the remote application server to get hold of the image from the registry and run it
- Port swapping in HTTP proxy
- Stop the older container

Question: Explain how Docker is different from other container technologies.
Answer: Docker is one of the latest container technologies and has emerged as one of the most popular. Built-in the cloud era, Docker comes with a lot of new features that were missing in older container technologies. One of Docker's finest features is that it can run on any infrastructure, be it your home machine or the Cloud.

Through Docker, more applications can now run on the old servers, and it also allows the process to package and ship programs. Docker also has a Container Hub that acts as a repository for containers, which are easy to download and use. Moreover, these containers can also be shared by your applications. It is also very well documented, which makes it better than other container technologies.

Question: If you were to exit the Docker Container, will you lose your data?
Answer: When a Docker Container is exited, no data loss occurs as all the data is written to the disk by the application for the sole purpose of preservation. This process is consistently repeated until and unless the container is unequivocally deleted. Moreover, the file system for the Docker container persists even after the Docker container is halted.

Question: How is Docker monitored in production?
Answer:;To monitor Docker in production, tools such as Docker stats and Docker events are available. Through these tools, one can get reports on important statistics. Once Docker stats are called with a container ID, it returns the container's CPU and memory usage. It is similar to the top command in Linux. On the other hand, Docker Events are commands to see a list of activities in process in Docker

Question: Shed some light on the workflow of Docker usage.
Answer: Below is a brief explanation of the workflow of Docker usage:

- Since the Dockerfile is the source code of the image, everything starts with it
- Once it is created, the Dockerfile is used to build the image of the container. This image is only the compiled version of the Dockerfile
- This image is then redistributed using the registry, which is like a repository of images.
- Further, the image can be used to run containers. A container, while it is running, is very similar to a VM without the hypervisor.

Question: Explain the disparity between the commands 'Docker run' and 'Docker creates.'
Answer:;The primary difference between Docker run and Docker create is that if you use the latter, the container is created in a 'stopped' state. Also, Docker creates can be used to store and output container ID for use later. The best way to do it is to use 'docker run' with —cidfile FILE_NAME as running it again won't allow overwriting the file.

Question: What is Visualization?
Answer: Virtualization, in its earlier days, was termed as a method of logically dividing mainframes to allow multiple applications to run concurrently. But as time progressed and the industry was able to allow for multiple operating systems to be run simultaneously on a single x86 based system, the meaning of virtualization changed considerably.

The net result? Virtualization allows the user to run two different OS on the same hardware. While the primary OS is the administrator, every guest OS goes through the processes such as bootstrapping, loading kernel, and more. It is also perfect for security, as every guest OS may not be allowed full access to the host OS, leading to the data breach.

- Paravirtualization

- Emulation
- Container-based virtualization

Question: What is the difference between a registry and a repository?
Answer: The Docker Registry is a service for hosting and distributing images, whereas the Docker Hub is the default registry. On the other hand, the Docker Repository is the collection of Docker images that are related. That is, they have the same name but different tags.

Question: Can JSON be used instead of YAML for the compose file in Docker? If yes, how?
Answer: Yes, JSON can be used instead of YAML for the Docker compose file. To use the JSON file with composing, the filename should be specified as the following:

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

Question: Explain CMD and ENTRYPOINT in a Dockerfile?
Answer: In a Dockerfile, both CMD and ENTRYPOINT instructions define which command will be executed while running a container. For their cooperation, there are some rules, such as:

The Dockerfile should specify at least one command from CMD or ENTRYPOINT
While using the container as an executable, ENTRYPOINT needs to be defined
When running the container with an alternative argument, CMD will be overridden

Question: Explain the process to run an application inside a Linux Container using Docker
Answer: Below are the steps on how to run an application inside the Linux Container using Docker

- Install and run Docker
- Fetch Fedora 21 (Linux based OS) base image from the Docker hub

- Check the containers in the system

- Start or stop Docker container
- Go inside a Docker container
- Remove container or image

Question: What is a Hypervisor?
Answer: A Virtual machine monitor, known as Hypervisor, is software to create and run virtual machines. It allows a single host computer to support more than one guest VMs. This is done by sharing resources like memory, processing, etc., thus reducing the memory, space, and maintenance requirements. There are two types of hypervisor:

- Type I: it is like a lightweight operating system that runs on the host's hardware.
- Type II: runs like software programs on an operating system.

Question: Explain containerization?
Answer: Docker containers contain different software packages like code, system tools, libraries, runtime, and settings required to run an application. The apps reside on top of the Docker engine layer. This is called containerization. It helps applications to run smoothly regardless of the environment.;

Question: Explain the main difference between containerization and virtualization.
Answer: Through virtualization, you can run many operating systems on a single physical server. Containerization occurs on the same operating system, where applications are packaged as containers and run under a single server or VM.

Question: Explain Docker Images, Docker Hub, Docker File?
Answer: Docker images: These are files that contain multiple layers and are used to execute code inside the Docker container. Images are built from instructions for an executable version of an application. Images speed up docker build by allowing each step to be cached.;

Docker hub: It is a service that finds and shares container images within a team. You can push and pull images, access private repositories of container images, build container images automatically from Github (or Bitbucket), and push them to DockerHub. Docker itself provides the service. Read more.

Docker file: It is a text document used to build an image. It contains instructions and commands to build the image. Docker reads the commands and assembles the image automatically.

[options]. If we do not give any options, then Docker gives all the version related information about client and server. For example, to get only the server version, use:

$ docker version –format '{{.Server.Version}}'

Question: Explain the login procedure to Docker Repository?
Answer: To log in to the Docker repository, use the following command:

docker login [OPTIONS] [SERVER]

For example, to login to a self-hosted (local) registry, you can add the server name:

$ docker login localhost:8080

Question: Explain various Docker Basic Commands?
Answer: Some Docker commands are:

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

Question: Explain how Docker Container is different from other containerization methods?
Answer: Docker containers can be easily deployed to any cloud platform. Also, developers can create ready-to-run containerized applications faster and manage and deploy applications easily. Containers can also be shared with applications. These features are not present in other containerization methods.

architectures), s390x, ppc64le.

Question: Is it possible for a container to restart by itself?
Answer: Yes, it is possible. Docker defines certain policies to restart the container. These are Off: container won't be restarted if it stops or fails,

- On-failure: container restarts only when a failure that occurred is not due to the user,
- Unless-stopped: container restarts only when a user executes the command to stop it,
- Always: the container is always restarted irrespective of error or other issues.

The command is:

$ docker run -dit — restart [unless-stopped|off|on-failure|always] [CONTAINER]

Question: Is it possible for the cloud to overtake the use of Containerization?
Answer: In this type of question, you can give your personal opinion. For example, as per my understanding, although the cloud is a good competitor, it cannot replace containerization. Most companies are using cloud and containerization in tandem to get the best out of both technologies.

Question: What are the various possible states of the Docker Container?
Answer: The different states of the Docker container are:

- Created – a container that is created but not active.
- Restarting – a container that is in the process of getting restarted.
- Running – running container.
- Paused – container whose processes are paused.
- Exited – a container that ran and completed.
- Dead – a container that the daemon tried and failed to stop.

Question: Explain the container orchestration and why we need to use it?
Answer: Container orchestration helps in managing the containers running in a

- Provisioning and deployment of containers,
- Load balancing,
- Allocation of resources between containers,
- Monitoring the health of containers and hosts,
- Scaling of containers,
- Switching containers from one host to another when the host is unavailable or lacking resources.

Question: Explain the memory-swap flag?
A memory-swap flag is a modifier flag that allows a container to write excess memory requirements into a disk when it has used all the available RAM. It is set only when the –memory flag is set. Example, if memory = "400m" and memory-swap = "1g", then the container can use 400m of memory and swap of 600m (1g-400m).

Question: Where are the docker volumes stored?
Answer: Volumes are created and managed by Docker (not to be accessed by Non-Docker processes) and are stored in a part of Docker host filesystem: /var/lib/docker/volumes/. Volumes are the most efficient way to persist data in Docker.

Question: Explain the various Docker Advanced Commands?
Answer: Some important Docker commands are:

- docker -version: to know the installed docker version. Syntax, Docker –version
- docker ps: lists all the docker containers that are running along with the container details. Syntax: docker ps
- docker ps -a: lists all the containers, including those that are running, stopped, exited, along with the details. Syntax: docker ps -a
- docker exec: Access the container and run commands inside that container. Syntax: docker exec [options]
- docker build: builds an image from Dockerfile. Syntax: docker build [options]

docker rm <container_id>

- docker rmi: Removes the docker image with the mentioned image id. Syntax: docker rmi <image_id>
- docker info: Gets detailed information about Docker installed on the system like the number of containers, images, running, paused, stopped, server version, volume, runtimes, kernel version, total memory etc. Syntax: docker info;
- docker cp: Copies a file from a docker container to a local system. Syntax: docker cp <source_path> <dest_path>
- docker history: displays the history of the docker image with the mentioned image name. Syntax: docker history <img_name>

Question: What are the commands to control Docker with Systemd?
Answer: To start the Docker daemon, many Linux distributions use the system. To start the services, use the command systemct1. If systemct1 is not available, the service command is used.;

$ sudo systemctl start docker
$ sudo service docker start

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

$ sudo systemctl enable docker
$ sudo systemctl disable docker

To modify daemon options, use:;

$ sudo systemctl edit docker

To view logs related to Docker service:

$ journalctl -u docker;

Question: What is the process of scaling your Docker containers

- Scale the server container and start 'n' instances of the server using:

$] docker-compose –file docker-compose-run-srvr.yml scale <service_name>=<n>

In the above command, the service name is defined in docker-compose-run-srvr.yml and are scaling it to 'n' times, where n can be any integer value.

- After scaling the Docker container, to check the container details, execute the following command:

$] docker ps -a

Question: Tell us about the steps for the Docker container life cycle.
Answer: Here are the steps:

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

Question: What is CNM?
Answer: CNM or Container Network Model is a specification that formally defines the steps needed to provide networking for containers while maintaining abstraction used to support multiple network drivers. CNM is built on three components, namely, sandbox, endpoint, and Network.

Answer: The three types are:

- Bind mounts: These can be stored anywhere on the host system
- Volume mount: they are managed by Docker and are stored in a part of the host filesystem.
- tmpfs mount: they are stored in the host system's memory. These mounts can never be written to the host's filesystem.

Question: Explain Docker Trusted Registry?
Answer: It is an image storage solution to store and manage the Docker images securely. Docker Trusted Registry is available on-premises or private cloud. DTR can be used during CI/CD processes for building, delivering, and running applications. DTR is highly available, efficient and has built-in access control.

Question: What is the purpose of Docker_Host?
Answer: Docker_host specifies the URL or Unix socket path used to connect to the Docker API. The default value is: UNIX://var/run/docker.sock

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

Question: Is it possible to run multiple copies of a Compose file on the same host? How?
Answer: This is done through the use of docker-compose. With Docker Compose, we can use a YAML file to configure the application's services. After this, with a single command, all the services can be created and started. To use Compose, follow the below steps:

- Define the app environment in the Dockerfile so that it can be replicated anywhere
- Define all the services of your application in the docker-compose.yml file.;
- Run docker-compose up to create and start the entire app.

Question: Explain Docker object labels.

containers, volumes, networks, local daemons, swarm nodes, and services. The key-

value pair should be unique for each object. Labels are static for the entire lifetime of the object.

Like

Be the first to like this.

Self Online Training / WordPress