**#DAY21 TASK**

- What is the Difference between an Image, Container and Engine?

- What is the Difference between the Docker command COPY vs ADD?

- What is the Difference between the Docker command CMD vs RUN?

- How Will you reduce the size of the Docker image?

- Why and when to use Docker?

- Explain the Docker components and how they interact with each other.

- Explain the terminology: Docker Compose, Docker File, Docker Image, Docker Container?

- In what real scenarios have you used Docker?

- Docker vs Hypervisor?

- What are the advantages and disadvantages of using docker?

- What is a Docker namespace?

- What is a Docker registry?

- What is an entry point?

- How to implement CI/CD in Docker?

- Will data on the container be lost when the docker container exits?

- What is a Docker swarm?

- What are the docker commands for the following:

- view running containers

- command to run the container under a specific name

- command to export a docker

- command to import an already existing docker image

- commands to delete a container

- command to remove all stopped containers, unused networks, build caches, and dangling images?

A Docker image is non-changeable file containing libraries, source code, tools and other files needed to run applications.

Docker Container is an open-source software development platform that bundles applications in "containers". Those containers can easily be ported from any system running the Linux OS. In other words, docker container is a lightweight, easy-to-deploy virtualization operating system environment.

Docker is what most of us look up to when it to Containers, right? There definitely seems to be no contradiction on this at least. Even though Containers are not VM replacements, the companies have started to utilize it in that way. Docker uses a client-server architecture. In this blog, I will be covering everything you have to know about Docker Architecture.

|  |  |
| --- | --- |
| COPY COMMAND | ADD COMMAND |
| COPY is a docker file command that copies files from a local source location to a destination in the Docker container. | ADD command is used to copy files/directories into a Docker image. |
| Syntax: COPY <src> <dest> | Syntax: ADD source destination |
| It only has only one assigned function. | It can also copy files from a URL. |

RUN executes commands and creates new image layers.

CMD sets the command and its parameters to be executed by default after the container is started. However, CMD can be replaced by docker run command line parameters.

The following are the methods by which we can achieve docker image optimization.

Using distroless/minimal base images

Multistage builds

Minimizing the number of layers

Understanding caching

Using Docker ignore

Keeping application data elsewhere

8 Key Components of Docker

The software comprises various components, each contributing significantly to the platform. These are:

1. The Docker Engine

The Docker Engine, also known as simply DE, is the essential central component of the Docker system. It must be downloaded and installed on the host computer. Next, a lightweight runtime system and the client-server technology that lies on top of the DE are used in creating and managing containers. There are three parts to the Docker Engine:

Server: The Docker daemon (dockerd) is in charge of creating and managing containers.

Rest API: The Rest API enables the communication between applications and Docker and gives Dockerd instructions.

Command Line Interface (CLI): Docker commands are executed using the CLI.

2. Docker images

The building blocks for containers are called Docker images. Like snapshots for virtual machines, Docker images are immutable, read-only files that include the source code. It also contains libraries, tools, dependencies, and additional files to run an application.

Images are essential for reducing disk utilization, enhancing reusability, and accelerating Docker build. The importance of maintaining compact images cannot be overstated because you want your containers to be quick and light. A Docker file, which contains detailed instructions for constructing a specific Docker image, is used to create each image. You can develop images and customized containers more quickly and easily if you’ve mastered producing Docker images from Docker files.

3. Docker file

A Docker file is a script with instructions on how to make a Docker image. In these instructions, you can find information about the operating system, languages, environment variables, file locations, network ports, and other details needed to run the image. The commands in the file are placed in groups, and those groups are automatically executed.

An image contains several layers. A new read-write layer is introduced once a Docker image has been launched to construct a container. The container layer is another name for this. With the additional layer, you can modify the base image and then commit your modifications to produce a fresh Docker image for usage in the future.

4. The Docker Hub

Docker Hub is the most extensive cloud-based archive of Docker’s container images. More than 100,000 images produced by open-source initiatives, software companies, and the Docker community are made accessible for use. The platform enables you to swiftly integrate your applications into a development pipeline, engage with team members, and ship your applications anywhere.

Like GitHub, developers can choose whether to maintain their private or public container images on Docker Hub by pushing and pulling them. Users of Docker Hub can freely distribute their images. To use the Docker filesystem as a starting point for any containerization project, they can also download prepared base images from it.

See More: What Is Enterprise Data Management (EDM)? Definition, Importance, and Best Practices

5. Docker volumes

When preserving data generated by a container that is already operating, using Docker volumes is a superior choice to adding extra layers to an image. With this feature’s assistance, users can store data, transfer it across containers, and mount it to new ones.

Because they are kept on the host, Docker volumes are immune to any changes that may occur throughout the container’s life cycle. When starting a container, you may generate and mount a Docker volume in several ways. These methods are all available.

6. Docker Compose

The process of creating and testing multi-container applications is made more accessible with the help of Docker Compose. It chooses the services to include in the application and generates a YAML Ain’t Markup Language (YAML) file. One can use a single command to deploy and run containers. Docker Compose is a helpful tool to streamline the process of running and managing numerous containers simultaneously.

It connects the containers that must cooperate and manages them with a single coordinated command. You can also define nodes, storage, and service dependencies using Docker Compose. Other technologies like Kubernetes and Docker Swarm allow more complex variations of similar tasks, known as container orchestration.

7. Docker Desktop

Docker Desktop was known initially as Docker for Windows and Docker for Mac. When using Docker Desktop, it takes a few minutes to start creating and operating containers on either Windows or Mac. The entire Docker development environment may be installed and set up quickly.

The environment may include components like Kubernetes and Credential Helper, and Docker features such as the DE, Docker Compose, Docker CLI client, and Docker Content Trust. On any cloud platform, the tool is used to create and share containerized apps and microservices in various languages and frameworks.

8. Docker containers

The active, operational instances of Docker images are known as containers. Docker images are read-only files, whereas containers contain executable, transient content. Users can interact with them, and administrators can use Docker commands to change their parameters.

Docker’s Key use cases

Here are just some of the use cases that provide a consistent environment at low overhead with the enabling technology of Docker.

1. Simplifying Configuration
2. Code Pipeline Management
3. Developer Productivity
4. App Isolation
5. Server Consolidation
6. Debugging Capabilities

The software that helps in the creation of Virtual machines where a virtual platform is provided to the operating systems to manage and execute the virtual machines is called Hypervisor which is otherwise called Virtual Machine Monitor or Emulator or Virtualizer. One system can control various virtual machines and this helps to manage the working of virtual machines via hypervisor.

Docker is a service for virtualization used in OS where software is delivered in containers with software, libraries, and configuration files. Written in Go language and developed by Solomon Hykes, the applications are created and deployed using containers and are developed as packages inside the same.

Advantages and Disadvantages of Docker

Below, we are discussing the major limitations and benefits of Docker, let’s look one by one –

i. Benefits of Docker

Following are some advantages of Docker, let’s discuss them in detail

a. Return on Investment and Cost Savings

Dockers first advantage is ROI. Especially for large, established companies, which need to generate steady revenue over the long term, the solution is only better if it can drive down costs while raising profits.

b. Rapid Deployment

It can decrease deployment to seconds. It is because of the fact that it can create a container for every process and even does not boot an OS. So, even without worrying about the cost to bring it up again, it would be higher than what is affordable, Data can be created as well as destroyed.

c. Security

Docker makes sure that applications that are running on containers are completely segregated and isolated from each other, from a security point of view, by granting us complete control over traffic flow and management.

d. Simplicity and Faster Configurations

The way Docker simplifies the matters is one of the key benefits of it. It gives flexibility to users to take their own configuration, put that into the code, and further deploy it without any problems.

However, the requirements of the infrastructure are no longer linked with the environment of the application, as Docker can be used in a wide variety of environments.

e. CI Efficiency

With the help of a Docker, we can build a container image and can further use that same image over every step of the deployment process.

The advantage of it is the ability to separate non-dependent steps and also run them in parallel. In addition, the duration of time it takes from build to production may speed up notably.

f. Continuous Integration

While it comes to Continuous Integration, Docker works well as part of its pipelines along with tools such as Travis, Jenkins, and Wercker.

These tools can save the new version as a Docker image, every time our source code is updated, just tag it with a version number and push to Docker Hub, then deploy it to production.

ii. Limitations of Docker

Some disadvantages of Docker are discussed, here:

a. Missing features

There are a ton of feature requests are under progress, like container self-registration, and self-inspects, copying files from the host to the container, and many more.

b. Data in the container

There are times when a container goes down, so after that, it needs a backup and recovery strategy, although we have several solutions for that they are not automated or not very scalable yet.

c. Run applications as fast as a bare-metal serve

In comparison with the virtual machines, Docker containers have less overhead but not zero overhead. If we run, an application directly on a bare-metal server we get true bare-metal speed even without using containers or virtual machines. However, Containers don’t run at bare-metal speeds.

d. Provide cross-platform compatibility

The one major issue is if an application designed to run in a Docker container on Windows, then it can’t run on Linux or vice versa. However, Virtual machines are not subject to this limitation.

So, this limitation makes Docker less attractive in some highly heterogeneous environments which are composed of both Windows and Linux servers.

e. Run applications with graphical interfaces

In general, Docker is designed for hosting applications which run on the command line. Though we have a few ways (like X11 forwarding) by which we can make it possible to run a graphical interface inside a Docker container, however, this is clunky.

Hence we can say, for applications that require rich interfaces, Docker is not a good solution.

f. Solve all your security problems

In simple words, we need to evaluate the Docker-specific security risks and make sure we can handle them before moving workloads to Docker.

The reason behind it is that Docker creates new security challenges like the difficulty of monitoring multiple moving pieces within a large-scale, dynamic Docker environment.

Docker uses a technology called namespaces to provide the isolated workspace called the container. When you run a container, Docker creates a set of namespaces for that container. These namespaces provide a layer of isolation.

A Docker registry is a storage and distribution system for named Docker images. The same image might have multiple different versions, identified by their tags. A Docker registry is organized into Docker repositories , where a repository holds all the versions of a specific image.

​a particular place where a person or thing can enter something or somewhere. The site has several entry points. entry point to something The emergency services closed all entry points to the square.

No, any data that your application writes to disk gets preserved in its container until you explicitly delete the container.

A Docker Swarm is a group of either physical or virtual machines that are running the Docker application and that have been configured to join together in a cluster.

view running containers: docker ps

command to run the container under a specific name: docker exec -it

command to export a docker: docker export

command to import an already existing docker image: docker import

commands to delete a container: docker rm -f < Container ID>

command to remove all stopped containers, unused networks, build caches, and dangling images: docker system prune