**1.What Is Docker?**

Docker is an open-source software platform. It helps to create, run and deploy the applications packaged as containers.

**2.What is Docker Image?**

A [Docker](https://www.techtarget.com/searchitoperations/definition/Docker) image is an immutablefile used to execute code in a Docker container. Docker images act as a set of instructions to build a Docker [container](https://www.techtarget.com/searchitoperations/definition/container-containerization-or-container-based-virtualization), like a template.

A Docker image has many layers, and each image includes everything needed to configure a container environment -- system libraries, tools, dependencies, and other files.

<https://www.techtarget.com/searchitoperations/definition/Docker-image>

**3.What is a Container ?**

A container is a runnable **instance** of an image. Containers are executable units of software in which application code is packaged, along with its libraries and dependencies, in common ways so that it can be run anywhere, whether it be on desktop, traditional IT, or the cloud.

**4.Docker Container vs Docker Image?**

A Docker container is a virtualized runtime environment used in application development. It is used to create, run and deploy applications that are isolated from the underlying hardware. A Docker container can use one machine, share its [kernel](https://www.techtarget.com/searchdatacenter/definition/kernel) and **virtualize the OS** to run more isolated processes. As a result, Docker containers are lightweight.

A Docker image is like a snapshot in other types of VM environments. It is a record of a Docker container at a specific point in time. Docker images are also immutable. While they can't be changed, they can be duplicated, shared or deleted. The feature is useful for testing new software or configurations because whatever happens, the image remains unchanged.

Containers need a runnable image to exist. Containers are dependent on images, because they are used to construct runtime environments and are needed to run an application.

**5.Container vs VM ?**

Container :-

abstraction at all application layer packages the code and dependencies together.multiple containers run in same machine

### **Anatomy of a Docker image**

A Docker image has many layers, and each image includes everything needed to configure a container environment -- system libraries, tools, dependencies and other files. Some of the parts of an image include:

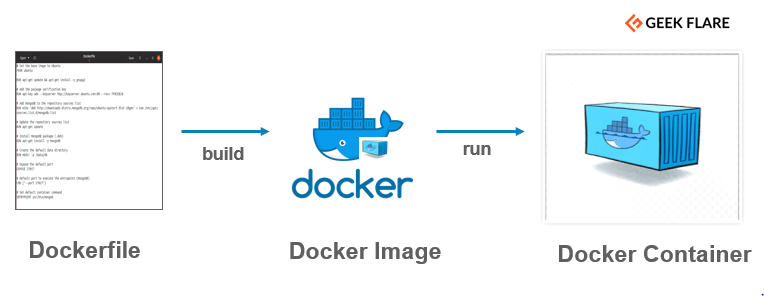
* **Base image**. The user can build this first layer entirely from scratch with the build command.
* **Parent image**. As an alternative to a base image, a parent image can be the first layer in a Docker image. It is a reused image that serves as a foundation for all other layers.
* **Layers**. Layers are added to the base image, using code that will enable it to run in a container. Each layer of a Docker image is viewable under /var/lib/docker/aufs/diff, or via the Docker history command in the command-line interface (CLI). Docker's default status is to show all top-layer images, including [repository](https://www.techtarget.com/searchitoperations/definition/container-repository), tags and file sizes. Intermediate layers are [cached](https://www.techtarget.com/searchstorage/definition/cache), making top layers easier to view. Docker has storage drives that handle the management of image layer contents.
* **Container layer.** A Docker image not only creates a new container, but also a writable or container layer. This layer hosts changes made to the running container and stores newly written and deleted files, as well as changes to existing files. This layer is also used to customize containers.
* **Docker manifest**. This part of the Docker image is an additional file. It uses [JSON](https://www.theserverside.com/definition/JSON-Javascript-Object-Notation) format to describe the image, using information such as image tags and digital signature.

**6. What is Docker File?**

It is a simple text file with a set of commands or instructions. These commands/instructions are executed successively to perform actions on the base image to create a new docker image.

Docker file Workflow

* Create a Docker file and mention the instructions to create your docker image
* Run docker build command which will build a docker image
* Now the docker image is ready to be used, use docker run command to create containers

****

**Docker File Syntax**

****

**FROM** – Defines the base image to use and start the build process.

**RUN** – It takes the command and its arguments to run it from the image.

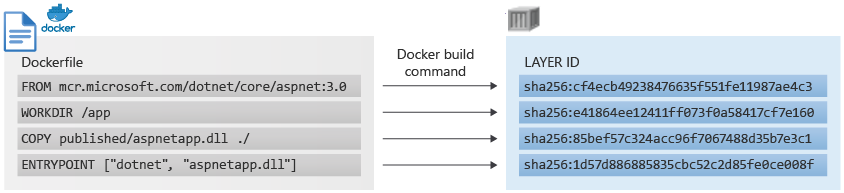
**CMD** – Similar function as a RUN command, but it gets executed only after the container is instantiated.

**ENTRYPOINT** – It targets your default application in the image when the container is created.

**ADD** – It copies the files from source to destination (inside the container).

**ENV** – Sets environment variables.

[**https://geekflare.com/dockerfile-tutorial/**](https://geekflare.com/dockerfile-tutorial/)

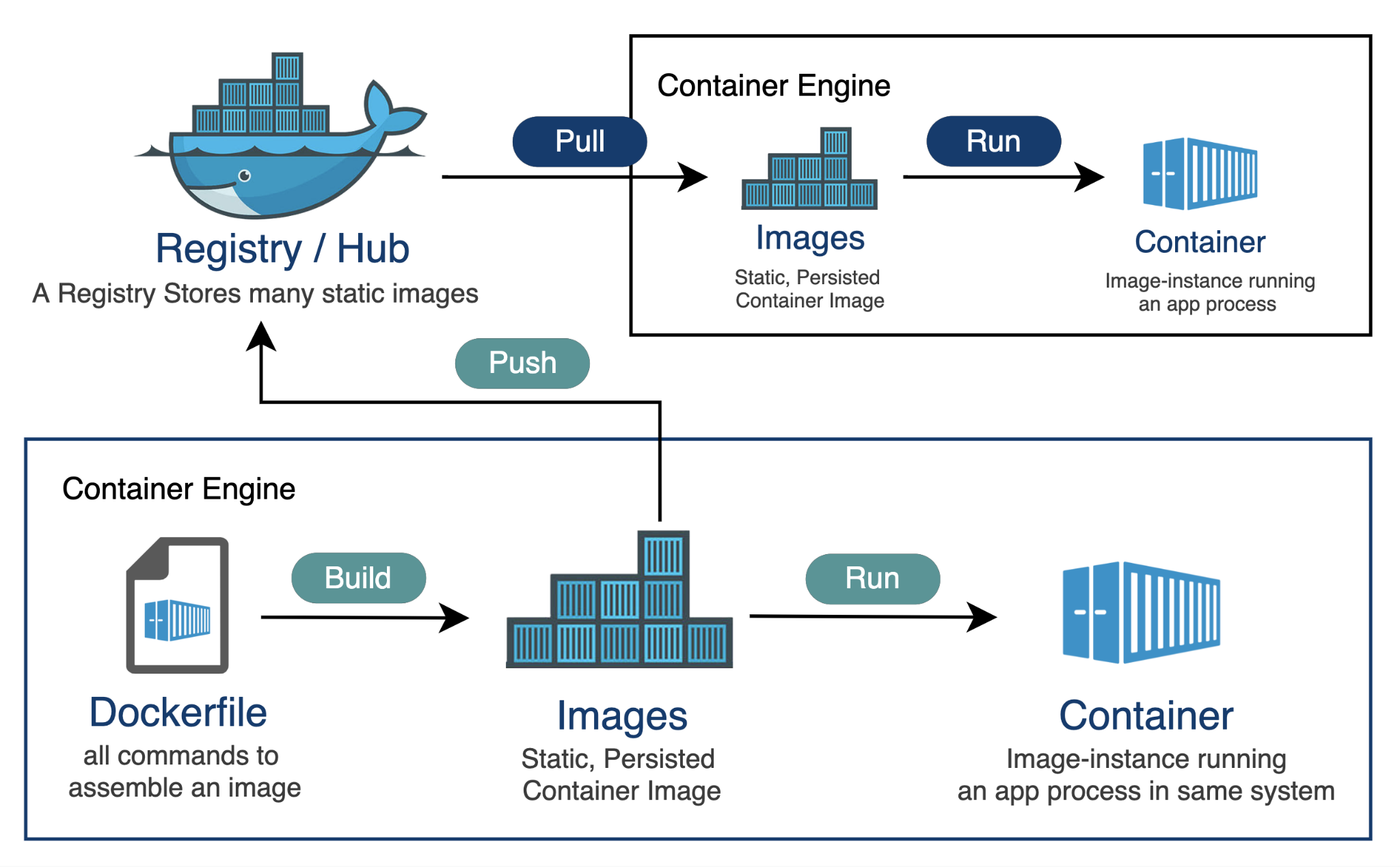
****

**7.Benefits of Docker ?**

Packaged Applications in a container that can be moved across platform and run everywhere

Easy for developers to quickly create, ready to run containerized applications

**Registries, Repositories, Images, and Tags**

****

**8.How do you share Docker images?**

Docker Hub is a service provided by Docker for finding and sharing container images.

**9.What is Docker Registry?**

A container registry is a service that stores and distributes container images and related artifacts. Docker Hub is an example of a public container registry that serves as a general catalog of Docker container images.

**10.Docker Repository ?**

Repositories let you share container images with your team, customers, or the Docker community at large.

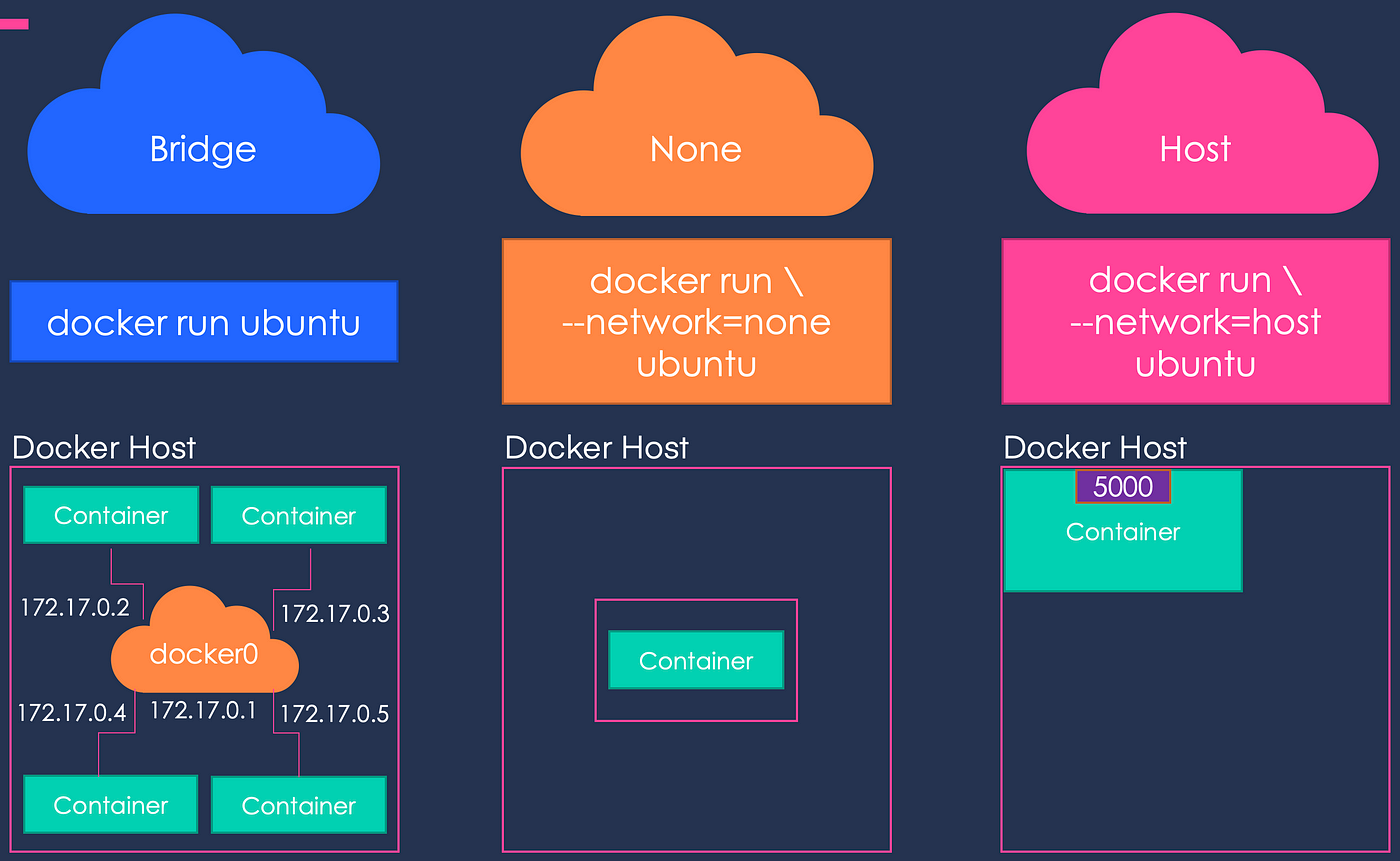
A single Docker Hub repository can hold many Docker images which are stored as tags. Docker images are pushed to Docker Hub through the docker push command.

**11.What is the Docker network?**

In Docker, a network is a virtual network that allows containers to communicate with each other and with the outside world. Docker provides three types of networks: bridge, host, and overlay.

* **Bridge network**: This is the default network in Docker. Containers in a bridge network can communicate with each other and with the host system but are isolated from other networks. By default, Docker creates a bridge network for each new container.
* **Host network**: Containers in a host network share the host system's network stack, meaning they can access the host's network interfaces directly. This provides better network performance but may compromise isolation.
* **Overlay network**: This type of network is used to connect containers running on different Docker hosts. Overlay networks use the VXLAN protocol to create a virtual network that spans multiple Docker hosts.

In addition to these built-in networks, Docker also allows you to create custom networks using the docker network command. Custom networks can be used to provide fine-grained control over container communication and can also be used to connect containers with external networks, such as a legacy application running on a different network.



**12.What are Docker Volumes ?**

Docker volumes are a way to store and manage persistent data in Docker containers.

Docker containers are used to run applications in an isolated environment. By default, all the changes inside the container are lost when the container stops. If we want to keep data between runs, Docker volumes and bind mounts can help.

In Docker, a volume is a way to store and share persistent data between containers or between a container and the host system. It helps to manage data that needs to persist beyond the lifetime of a container.

Docker volumes are created and managed using the docker volume command or the Docker API.Volumes can be mounted in a container at a specific path, allowing the container to read and write data to the volume. Volumes can also be shared between multiple containers, making it possible for containers to share data and collaborate on a task.

**13.Docker VS Virtual Machine?**

|  |  |  |
| --- | --- | --- |
|  | **Docker** | **Virtual Machine(VM)** |
| **Boot-Time** | **Boots in a few seconds.** | **It takes a few minutes for VMs to boot.** |
| **Runs on** | **Dockers make use of the execution engine.** | **VMs make use of the hypervisor.** |
| **Memory Efficiency** | **No space is needed to virtualize, hence less memory.** | **Requires the entire OS to be loaded before starting the surface, so less efficient.** |
| **Isolation** | **Prone to adversities as no provisions for isolation systems.** | **Interference possibility is minimum because of the efficient isolation mechanism.** |
| **Deployment** | **Deploying is easy as only a single image, containerized, can be used across all platforms.** | **Deployment is comparatively lengthy as separate instances are responsible for execution.** |
| **Usage** | **Docker has a complex usage mechanism consisting of both third party and docker managed tools.** | **Tools are easy to use and simpler to work with.** |

**14.What is Docker compose?**

Docker Compose is a tool that was developed to help define and share multi-container applications. With Compose, we can create a YAML file to define the services and with a single command, can spin everything up or tear it all down.

**15.How to debug an application inside a docker container?**

**16.How to manage containers in docker?**

To manage containers in Docker, you can use a variety of Docker commands. Here are some common container management commands:

docker ps - This command lists all running containers.

docker ps -a - This command lists all containers, including stopped containers.

docker run - This command is used to create and start a new container.

docker start - This command is used to start a stopped container.

docker stop - This command is used to stop a running container.

docker restart - This command is used to restart a running container.

docker rm - This command is used to remove a stopped container.

docker logs - This command is used to display the logs of a container.

docker inspect - This command is used to display detailed information about a container.

docker exec - This command is used to run a command inside a running container.

docker stats - This command is used to display real-time resource usage statistics for all running containers.

docker top - This command is used to display the running processes in a container.

docker port - This command is used to display the public-facing port of a container.

In addition to these commands, you can also use Docker Compose to manage multiple containers as part of a larger application. With Docker Compose, you can define and run multiple containers in a single configuration file.

Docker also provides a web-based user interface called Docker Dashboard, which provides a graphical way to manage containers and other Docker resources.

**17.How to access Docker Container shell ?**

Using the **docker exec** command. We can access the shell. In order to have an interaction with the shell. You can use **docker exec -it** which will create an interactive terminal of the container’s shell

**18.Docker Best Practices?**

**19.DockerFile Best Practices ?**

* Getting base os from a certified registry like docker hub , quay by identifying the secured ones.
* You have to create a user in order to build rootless containers as it is safe to use rootless containers instead of root containers. The root containers have more file permissions when compared to the rootless containers.

**20.Podman vs docker**

**21. Basic Docker commands**

docker run - This command is used to create and start a new Docker container.

docker ps - This command is used to list all running containers.

docker ps -a - This command is used to list all the containers including the stopped and active containers

docker images - This command is used to list all available Docker images.

docker stop - This command is used to stop a running container.

docker rm - This command is used to remove a stopped container.

docker rmi - This command is used to remove a Docker image.

docker pull - This command is used to download a Docker image from a registry.

docker push - This command is used to upload a Docker image to a registry.

docker exec -it - This command is used to run a command inside a running container with an interactive terminal(bash shell)

docker build - This command is used to build a Docker image from a Dockerfile.

docker network - This command is used to manage Docker networks.

docker volume - This command is used to manage Docker volumes.

docker-compose - This command is used to manage multi-container Docker applications, allowing you to define and run multiple containers in a single configuration file.

docker build -t jobs . => jobs . is image name.

docker run jobs => run the docker images.