Docker

What is Docker?

It is a platform for building, running, and shipping applications in a consistent manner.

- If your application is working fine on the developer machine it needs to run on other machines also but sometimes it's not.

  Why?

  - One or more files missing.
  - Software version mismatch.
  - Different configuration settings.

- With docker, we can easily package our application with everything it needs and run it anywhere and anytime.

  Example: If your application package has Jave, HTML, and CSS configurations and versions now you can take your package and run it on any machine with the help of Docker. So, it means this application will work on Dev, Test, Prod, and any other machines!

Why Docker?

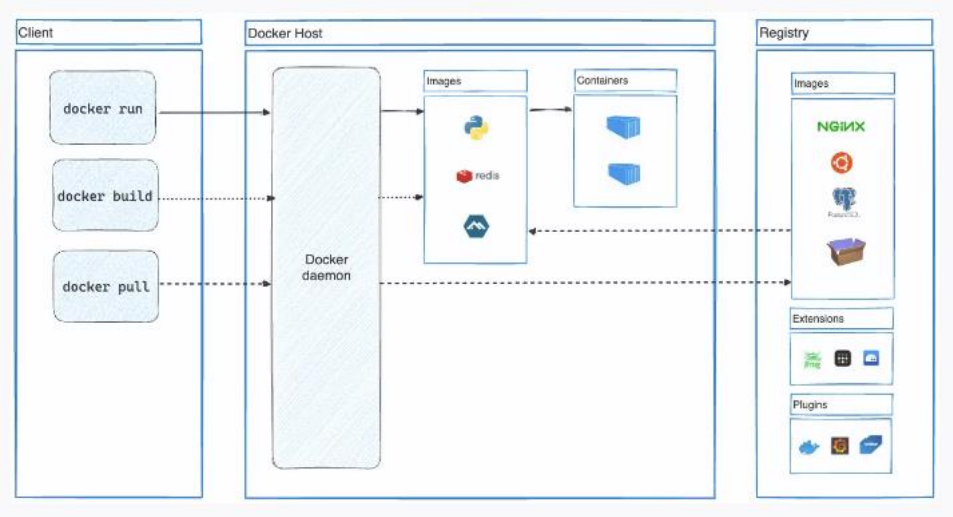
It offers portability, consistency, and scalability for deploying applications in different environments.

- Portability: Docker images can be run on any operating system, making it easy to deploy applications to different environments.
- Consistency: Docker containers provide a consistent environment for applications to run in. This means that developers can be confident that their applications will work the same way on any machine that they are run on.
- Scalability: Docker containers can be easily scaled up or down, making it easy to manage applications with varying workloads.

How does Docker work?

- Docker uses a client-server architecture. The Docker client talks to the Docker daemon, which does the heavy lifting of building, running, and distributing your Docker containers.
- The Docker client and daemon can run on the same system, or you can connect a Docker client to a remote Docker daemon.

Docker Architecture:



Docker follows Client-Server architecture, which includes three main components that are Docker Client, Docker Host, and Docker Registry.

1. Docker Client:

The Docker client uses commands and REST APIs to communicate with the Docker Daemon (Server). When a client runs any docker command on the docker client terminal, the client terminal sends these docker commands to the Docker daemon. Docker daemon receives these commands from the docker client in the form of command and REST API's request.

2. Docker Host:

Docker Host is used to provide an environment to execute and run applications. It contains the docker daemon, images, containers, networks, and storage.

3. Docker Registry:

Docker Registry manages and stores the Docker images.

There are two types of registries in the Docker -

Pubic Registry - A Public Registry is also called a Docker hub.

Private Registry - It is used to share images within the enterprise.

What is the use of Docker image?

- A Docker image is a read-only template containing a set of instructions for creating a container that can run on the Docker platform.
- It provides a convenient way to package up applications and preconfigured server environments.
- Which you can use for your private use or share publicly with other Docker users.

What is the use of Docker Container?

- A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing.

Installation of docker:

Link: https://hub.docker.com/

You can download the docker desktop by using the above link. Docker is available for Windows, Mac, and Linux.

Initially, docker was created only for Linux OS but later docker was developed for Windows and Mac OS's.

Commands:

- If any new joiner joins your team they do not need to install all the setups they just use the below command.

```
docker-compose up
```

It will download and run all the required installations and configurations on their machines.

- If you just want to remove all the unwanted installations and configurations in your machine use the below command to remove.

```
docker-compose down – rmi all
```

- List of images we have locally?

```
C:\Users\ShaikSri>docker images
```

- To pull the image use the below command,

```
C:\Users\ShaikSri>docker pull mongo
```

- To pull the containers use the below command,

```
C:\Users\ShaikSri>docker run mongo
```

- List of containers we have locally?

```
C:\Users\ShaikSri>docker ps
```

- Use Ctrl+C to exit the cmd process.
- By using the '-d' or '-detach' it runs containers in the background and prints the counter ID.

```
C:\Users\ShaikSri>docker run -d nginx:1.25
```

- Docker logs {container} = View logs from the service running inside the container (Which are present at the time of execution).

```
C:\Users\ShaikSri>docker logs 3280564c35fc
```

- We can run Containers without pulling the image. (Shortcut)

```
C:\Users\ShaikSri>docker run redis:6.0-alpine
```

- Docker stop {container} = stop one or more running containers

```
C:\Users\ShaikSri>docker stop 9abf501d070f
```

- By using -p or –publish = Publish a container's port to the host

```
C:\Users\ShaikSri>docker run -d -p 8000:80 nignx
```

- By using -a or -all = list of all containers (stopped and running)

  C:\Users\ShaikSri>docker ps -a

- By using this command, we can run the stopped containers. docker start {container}

  C:\Users\ShaikSri>docker start 7118acdad979

- We can add names to our containers.

  C:\Users\ShaikSri>docker run --name sri-sk -d nginx

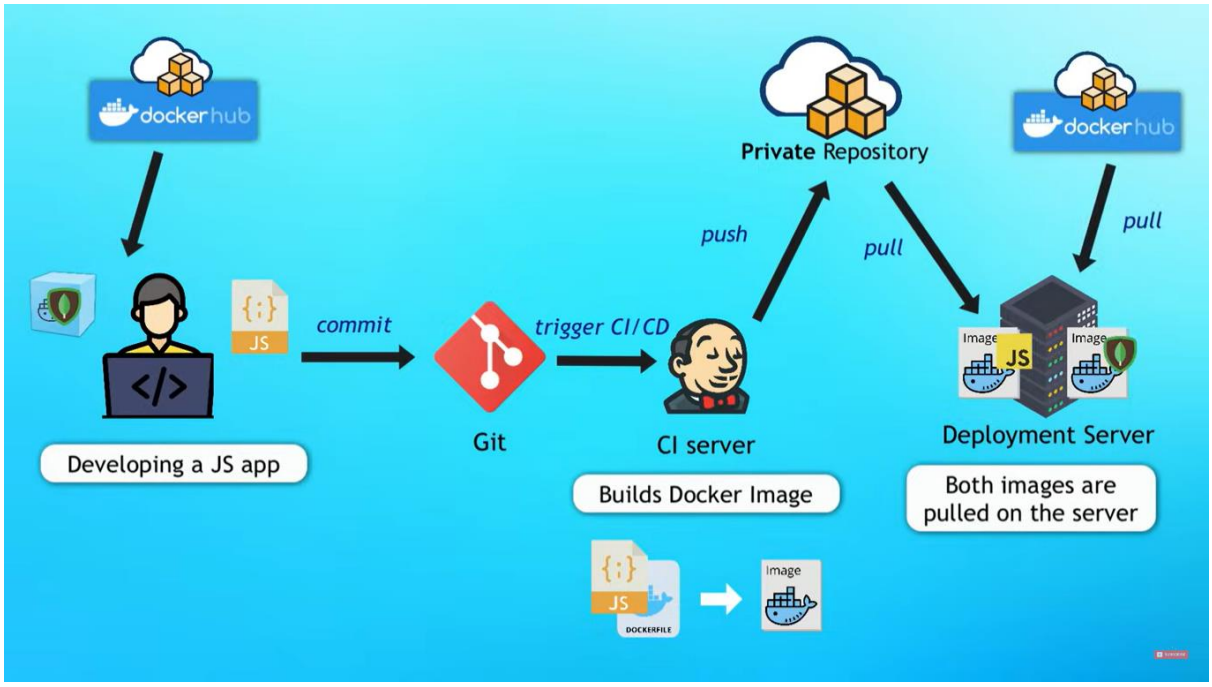
  By using names also, we can find logs

  C:\Users\ShaikSri>docker logs sri-sk

Commands:

- Build - Build an image from a Dockerfile
- History - Show the history of an image
- Import - Import the contents from a tarball to create a filesystem image
- Inspect - Display detailed information on one or more images
- Load - Load an image from a tar archive or STDIN
- Ls - List images
- Prune - Remove unused images
- Pull - Download an image from a registry
- Push - Upload an image to a registry
- rm - Remove one or more images
- save - Save one or more images to a tar archive (streamed to STDOUT by default)
- tag - Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Docker – real-life development process:



Imagine if you're a developing JavaScript application on your laptop in a development environment.

Your javascript application uses mango db databases. So now, instead of installing it on your laptop, you download a docker container from the docker hub.

So now you connect your application with mango db and start developing and you developed.

Now you want to test it right so you commit your code in repo or git or some other apps and then this will trigger a CI a Jenkins build or whatever you configured.

Then your application and docker files produce artifacts it will produce a docker image artifact.

Once docker images are created they get pushed to the private docker repository and then the docker image is deployed in the deployment server.

Then this pulls into the docker image and also pulls the mango db from the docker hub.

Now you have two containers one is my custom container and another one is a publicly available mango db dev server.

Both containers communicate with each other and run as an application.

Now whoever what to test this app, will test it in stage dev and prod.

YT: https://www.youtube.com/watch?v=pg19Z8LL06w (Short Course)

YT: https://www.youtube.com/watch?v=3c-iBn73dDE&t=17s (Full Course)