**Docker**

Docker is an open source platform used for containerization, allowing to package an application along with it’s dependencies into a standardized unit called container. Containers are light-weight, portable and isolated environments that ensure consistent behaviour across different computing environments. Docker is just like build once and use anywhere.

Docker simplifies the process of shipping and deploying applications by encapsulating them in containers. Benefits of Docker includes faster application deployment, resource efficiency, improved scalability and simplifies DevOps workflows.

**Docker Architecture**

**Docker Engine:** The core component of Docker that runs and manages containers.

**Docker client:** CLI used to interact with the docker images.

**Docker Daemon:** Background service responsible for managing containers, images.

**Docker Registry:** Stores docker images.

Ex: Dockerhub

Advantages:

1)Easy Deployment.

2)High Scalability

3)Increased Overall productivity.

In order work with Docker images and containers, must install the docker. In EC2 instance launch an instance named docker. Open the terminal (Git Bash) a CLI to interact with the instances. Move to the directory in the CLI where the key pair file is saved, which is generated during the creation of an instance.

Command to enter into ec2 instance as root:

Ssh -i “Linux.pem”<key\_pair\_file\_name> ec2-user@public\_ip\_address.

sudo su -changes the user to the root user, a root user can able to perform all the operations, it has access to every operation.

To install docker in the instance:

**yum install docker -y**

The above mentioned command will download and install the docker in to the ec2 instance.

Command: **service docker start**

The service docker start command is used to start the Docker service on Linux-based systems. If Docker is installed and its service is available, this command ensures that the Docker daemon is running and ready to manage containers.

Command: **docker --version**

The docker --version command is used to check the currently installed version of Docker on your system.

Command: **cd /var/lib/docker**

The /var/lib/docker directory is the default storage location where Docker stores all its data. This includes images, containers, volumes, networks, and other related metadata.

**Pulling an Image into Docker:**

To pull a Docker image from a repository (usually Docker Hub), use the docker pull command.

**Pull an Image**: Use the docker pull command followed by the image name.

Command: **docker pull nginx**

By default, this pulls the latest tag of the image.

**Specify a Version (Optional)**: If you want a specific version of NGINX, append the tag (e.g., 1.23.1)

Command: **docker pull nginx:1.23.1**

**Verify the Image**: List the downloaded images to confirm:

Command: **docker images**

**Using the pulled images:**

Run the nginx container:

**docker run -itd –name mynginx -p 8080:80 nginx:1.23.1**

-d: Runs the container in detached mode.

--name my-nginx: Names the container as "mynginx".

-p 8080:80: Maps port 8080 on the host to port 80 in the container.

**Access NGINX:**

http://<server\_ip\_address>:8080

**Verifying the running container:**

Command: **docker ps**

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

872e53c425c0 nginx:1.23.1 "/docker-entrypoint.…" 5 seconds ago Up 4 seconds 0.0.0.0:8080->80/tcp, :::8080->80/tcp mynginx

**Creating Tags and Pushing Images to Docker Hub:**

**1)Log In to Docker Hub**

Before pushing an image, log in to your Docker Hub account:

Command: **docker login**

Enter your **Docker Hub username** and **password** when prompted.

**2)Tag the Image**

Docker images need to be tagged to specify the repository and version for pushing to Docker Hub. The format for tagging is:

**docker tag <image-id> <dockerhub-username>/<repository-name>:<tag>**

Command: **docker tag nginx:1.23.1 koushikrdy/dev:1.23.1**

**3**)**Push the image**

Push the tagged image into the dockerhub

**docker push <dockerhub\_userbame>/<repository\_name>:<tag>**

command: **docker push koushikrdy/dev:1.23.1**

**4: Verify the Image on Docker Hub**

1. Log in to your Docker Hub account in a browser.
2. Navigate to the "Repositories" section.
3. Check for your newly pushed image (e.g., nginx with tag 1.23.1).

**Basic Docker commands**

**1. Docker Version**

Displays the installed version of Docker.

docker --version

Example Output:

Docker version 20.10.12, build e91ed57

**2. Start Docker**

Starts the Docker service:

service docker start

To check if Docker is running:

service docker status

**3. Pull an Image**

Downloads an image from Docker Hub or a custom registry.

docker pull <image-name>:<tag>

Example:

docker pull nginx:latest

**4. List Images**

Displays all downloaded Docker images.

docker images

Example Output:

REPOSITORY TAG IMAGE ID CREATED SIZE

nginx latest 4cdc5dd7eaad 3 days ago 142MB

**5. Run a Container**

Runs a container from an image.

docker run <options> <image-name>

Options:

* -d: Run in detached mode (background).
* -it: Interactive terminal.
* --name: Assign a custom name to the container.
* -p: Map ports between the host and the container.

Example:

docker run -itd --name my-nginx -p 8080:80 nginx

**6. List Running Containers**

Shows all currently running containers.

Example Output:

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d0e6fdb97c91 nginx "/docker-entrypoint.…" 2 minutes ago Up 2 minutes 0.0.0.0:8080->80/tcp my-nginx

**7. List All Containers**

Displays all containers (running and stopped).

docker ps -a

**8. Stop a Container**

Stops a running container.

docker stop <container-id|container-name>

Example:

docker stop my-nginx

**9. Start a Stopped Container**

Starts a container that has been stopped.

docker start <container-id|container-name>

Example:

docker start my-nginx

**10. Remove a Container**

Deletes a stopped container.

docker rm <container-id|container-name>

Example:

docker rm my-nginx

**11. Remove an Image**

Deletes an unused image.

docker rmi <image-id|image-name>

Example:

docker rmi nginx:latest

**12. View Logs of a Container**

Fetches logs of a container.

docker logs <container-id|container-name>

Example:

docker logs my-nginx

**13. Inspect a Container**

Displays detailed information about a container.

docker inspect <container-id|container-name>

**14. Execute a Command in a Running Container**

Runs a command inside an active container.

docker exec -it <container-id|container-name> <command>

Example:

docker exec -it my-nginx bash

**15. Check Docker System Resources**

Shows system-wide information about Docker usage.

docker system df

**17. Create a Tag for an Image**

Tags an existing image for pushing to a registry.

docker tag <image-id|image-name> <repository-name>:<tag>

Example:

docker tag nginx:latest myusername/nginx:1.0

**18. Push an Image**

Uploads a tagged image to Docker Hub.

docker push< user\_name>/<repository-name>:<tag>

Example:

docker push myusername/nginx:1.0