DOCKER

As a Docker developer, here are the key topics you should be familiar with:

### 1. Docker Basics:

* **Containerization Concept:** Understanding the concept of containerization and how Docker implements it.
* **Docker Engine:** Knowing the components of Docker, including Docker daemon, Docker client, and Docker images.
* **Dockerfile:** Writing Dockerfiles to define container configurations, including base images, dependencies, environment setup, and commands to run.

### 2. Docker Images:

* **Image Creation:** Building Docker images using Dockerfiles or pulling images from Docker Hub.
* **Image Registry:** Understanding Docker Hub or other image registries to store and share Docker images.

### 3. Docker Containers:

* **Container Lifecycle:** Managing the lifecycle of containers, including creation, starting, stopping, pausing, and deleting containers.
* **Container Networking:** Configuring networking for containers, understanding bridge networks, host networks, and overlay networks.
* **Volumes and Data Management:** Handling persistent data with volumes, binding mounts, and managing data within containers.

### 4. Docker Compose:

* **Multi-Container Applications:** Defining and managing multi-container applications using Docker Compose files.
* **Service Configuration:** Understanding services, networks, volumes, and other configurations in Docker Compose.

### 5. Docker Orchestration:

* **Container Orchestration:** Basics of container orchestration tools like Docker Swarm or Kubernetes for managing containerized applications at scale.
* **Scalability and High Availability:** Understanding scaling, load balancing, and ensuring high availability of containers in production environments.

### 6. Security and Best Practices:

* **Container Security:** Implementing security best practices, container isolation, and ensuring secure Docker configurations.
* **Docker Best Practices:** Following best practices for Dockerfile optimization, image size reduction, efficient layering, and containerization strategies.

### 7. Monitoring and Troubleshooting:

* **Logging and Monitoring:** Monitoring containers, collecting logs, and understanding container health.
* **Troubleshooting:** Identifying and troubleshooting issues related to containers, networking, or Docker configurations.

### 8. Continuous Integration/Continuous Deployment (CI/CD):

* **CI/CD Pipelines:** Integrating Docker with CI/CD pipelines for automated building, testing, and deploying containerized applications.

Mastering these topics will enable you to effectively use Docker for developing, deploying, and managing containerized applications, contributing to efficient and scalable software development processes.

Certainly! Docker is a popular platform for containerization, which enables developers to package applications and their dependencies into standardized units called containers. These containers can run consistently across different environments, providing isolation, portability, and scalability.

### Containerization Concept:

Containerization is a method of packaging, distributing, and running applications in containers. It encapsulates an application and its dependencies, including libraries and other necessary components, ensuring that it runs uniformly across different computing environments.

### Docker Components:

1. **Docker Daemon:** The Docker daemon (dockerd) is a background process that manages Docker containers, handling container creation, running, and monitoring.
2. **Docker Client:** The Docker client (docker) is a command-line tool that allows users to interact with the Docker daemon, issuing commands to build, manage, and run containers.
3. **Docker Images:** Docker images are read-only templates that contain the application code, libraries, dependencies, and configurations required to create a container.

### Dockerfile:

A Dockerfile is a text file that contains instructions for building a Docker image. It defines the steps needed to create a reproducible and consistent container environment. Here's an example of a simple Dockerfile:

Dockerfile

# Specify the base image

FROM python:3.9

# Set the working directory inside the container

WORKDIR /app

# Copy the application files into the container

COPY . /app

# Install dependencies

RUN pip install -r requirements.txt

# Set environment variables

ENV DEBUG\_MODE=True

# Specify the command to run when the container starts

CMD ["python", "app.py"]

### Explanation of Dockerfile Directives:

* **FROM:** Specifies the base image to build upon. Here, it uses the official Python image from Docker Hub.
* **WORKDIR:** Sets the working directory inside the container where subsequent commands will be executed.
* **COPY:** Copies files from the host machine into the container's filesystem.
* **RUN:** Executes commands to install dependencies or perform setup actions inside the container.
* **ENV:** Sets environment variables inside the container.
* **CMD:** Defines the default command to run when the container starts. It runs the Python application (**app.py** in this case).

### Dockerfile Usage:

1. **Building an Image:** Use the **docker build** command with the path to the directory containing the Dockerfile:

Copy code

docker build -t myapp .

1. **Running a Container:** Start a container using the built image with **docker run**:

arduinoCopy code

docker run myapp

Dockerfiles play a crucial role in defining the configuration and setup required for running applications in containers. They enable developers to create portable and reproducible environments, facilitating efficient application deployment.

Top of Form

### Image Creation:

#### **Using Dockerfiles:**

* **Dockerfiles:** These are text files that contain a set of instructions to assemble a Docker image. They specify the base image, environment variables, dependencies, and commands needed to run an application.
* **Build Process:** Dockerfiles are used with the **docker build** command to create images. This command reads the Dockerfile's instructions and builds an image accordingly.

#### **Pulling Images from Docker Hub:**

* **Docker Hub:** It's a cloud-based registry service by Docker that allows developers to share and store Docker images publicly or privately.
* **Pulling Images:** Users can download pre-built Docker images from Docker Hub using the **docker pull** command. These images can be used as base images for applications or services.

### 2. Image Registries:

#### **Docker Hub:**

* **Repository:** Docker Hub hosts repositories containing Docker images. Users can push (upload) their images to Docker Hub or pull (download) images from public repositories.
* **Private Repositories:** Docker Hub also offers private repositories for teams or individuals to store images privately.

#### **Other Image Registries:**

* Apart from Docker Hub, there are other image registries available such as:
  + **AWS Elastic Container Registry (ECR):** Amazon's fully managed Docker container registry.
  + **Google Container Registry (GCR):** Google Cloud's managed registry for Docker images.
  + **Azure Container Registry (ACR):** Microsoft Azure's registry service for storing and managing container images.

### Workflow Example:

1. **Creating Docker Image with Dockerfile:**
   * Write a Dockerfile specifying the application's dependencies, configuration, and build steps.
   * Use the **docker build** command to build the image:

Arduino

docker build -t myapp-image .

1. **Pulling Images from Docker Hub:**
   * Pull a pre-built image from Docker Hub:

**docker pull image-name:tag**

1. **Pushing Images to Registries (like Docker Hub):**
   * After building an image, push it to Docke

Understanding image creation and usage, along with using image registries like Docker Hub, is essential for managing and sharing Docker-based applications effectively within development teams or the broader community.

Top of Form



What is Virtualization?

The process of creating virtual server by using the base machine. We can create multiple virtual instances on the single physical machine for better enhancement, resource utilization, etc., ex are vm ware,hyper v

* Host OS can have Multiple Virtual Guest OS with the help of Virtualization
* Example: Windows 11 OS Host machine, can have multiple Virtual Box Instances having OS say Ubuntu, Windows etc.
* Why Software Testers need Virtualization?
  + We have to test in multiple application supported Environments
* *Hypervisor* is required to setup VMs
* But Virtualization is Costly and the Default things which come with OS and are not required by our Application, result in waste of space

 What is Containerization?

* Containers has no OS
* Containers use the things from Host OS
* Containers contain only libraries and binaries that are required by the Application.
* Containers are light weight, faster, low cost and don't waste space

 What is a Docker?

* Docker is a Software Tool
* Using Docker we can create containers
* Docker tool packages application under test and its required dependencies together as *Docker Container*
* We can perform Testing on Application using these containers

 Docker Image

* We can execute the Docker Image to create the required Docker Container
* Docker Image like exe files contains the executable instructions to get the required Docker Container for running the application

 Docker Hub

* Cloud Repository where we can create, find, test, store and share Docker Images
* We can find and get Docker Image from Docker Hub
* We can upload our Docker Image to Docker Hub
* Sharing of Docker Image is possible via Docker Hub

 Docker Container

* We can run, stop and delete the docker containers

Installing Docker

* Pre-requisite: Windows 10 64-bit or higher
  + Type winver in windows search and check the windows version
* Pre-requisite: Minimum 4 GB Ram
  + Type system information in windows search and check the RAM
* Pre-requisite: BIOS-Level Virtualization is enabled
  + Search for task manager in windows search, go to Performance Tab and check Virtualization is enabled
* Pre-requisites: Enable WSL2
  + Enable WSL2 (i.e. Windows Subsystem for Linux) feature in Windows
    - Search for 'Turn windows feature on or off' in windows search and make sure the below 'Windows Features' options are selected:
      * Windows Subsystem for Linux
      * Virtual Machine Platform
    - Restart the PC
    - Check Step#2 at <https://docs.microsoft.com/en-us/windows/wsl/install-manual>
      * Builds lower than 18362 do not support WSL 2
      * Type winver in windows search and check
    - Check Step#4 at <https://docs.microsoft.com/en-us/windows/wsl/install-manual>
      * Download the Linux kernel update package and install
    - Check Step#5 at <https://docs.microsoft.com/en-us/windows/wsl/install-manual>
      * Open Powershell and run the command wsl --set-default-version 2
    - Install Ubuntu 18.04 LTS at Microsoft Store at link [https://www.microsoft.com/en-in/p/ubuntu-1804-lts/9n9tngvndl3q?rtc=1&activetab=pivot:overviewtab](https://www.microsoft.com/en-in/p/ubuntu-1804-lts/9n9tngvndl3q?rtc=1&amp;activetab=pivot:overviewtab)
      * Install and open it
      * Set username and password
      * We can later open it by searching it in windows
* Search 'Download Docker for Windows' in Google
  + Install Docker with default settings
  + Launch it by searching 'Docker' in windows
* Check the below Commands in command prompt
  + docker -v
* Post steps: Restart the machine after installing Docker

# Docker Commands

* docker --version
  + Gives docker version
* docker -v
  + Gives docker version same as docker --version
* docker info
  + Gives details about how many containers running/paused/stopped, images etc.
* docker --help
  + Gives the list of docker commands we can run
* docker command --help
  + Get the help details about the specified command
  + example: get info --help
* docker login
  + login to docker hub - <https://hub.docker.com/>
  + It will ask for username and password
* docker logout
  + logout from docker hub
* docker images
  + Displays the list of docker images already available in your machine
* docker pull
  + To download the docker images to our machine
  + example: docker pull ubuntu
  + docker images
* docker rmi imageid
  + To remove the docker image from our machine
  + example: docker rmi ba6acccedd29
  + docker images
* docker ps
  + Displays the list of docker containers running in our machine
* docker run ubuntu
  + Downloads the docker image if not available in our machine
  + Actual purpose is to creates the container
  + After running trigger:
    - docker images
    - docker ps
* docker run -it ubuntu
  + It will run the given ubuntu container
  + In new command prompt, run the below commands:
    - docker images
    - docker ps
* docker stop containerid
  + In new command prompt, run the below commands:
    - docker ps
    - docker stop containerid
    - docker ps
* docker start containerid
  + In new command prompt, run the below comands:
    - docker ps
    - docker start cointainerid
    - docker ps
* docker stats
  + Gives statistics about the containers that are currently running
* docker system df
  + Gives information about the how many containers are actively running etc.
* docker system prune -f
  + Deletes all the stopped containers
  + Images won't be deleted
  + docker system df
* docker rmi imageid
  + To remove the docker image from our machine
  + example: docker rmi ba6acccedd29
  + docker images

how to connect to an EC2 instance through cmd

ssh -I pemfile.pem ubuntu@public ip iddress

if you find any error like unprotected key file then use the following commands

icacls future\_ride.pem /inheritance:r

icacls future\_ride.pem /grant "%USERNAME%":(F)

then again use ssh command

now update the systems

sudo apt-get update

sudo apt-get upgrade -y

sudo apt install django

to install docker in ec2 use sudo systemctl install docker.io -y

to check the docker status

sudo systemctl status docker

check if docker running or not by using docker tun hello world a simple commandif nor running then use command like

sudo usermod -aG docker ubuntu the restart the instance by logout and login again

token github passsword

ghp\_qDUwErtZJBOK7ZilUigUHKo8DGoGML0Z3hPW