**Docker Setup and Commands Reference**

**Introduction to Docker**

**What is Docker?**  
Docker is a platform that enables you to build, run, and manage applications in lightweight, portable containers. Containers package an application and its dependencies into a single unit, ensuring that the app runs consistently across different environments—from a developer’s laptop to production servers.

**How Does Docker Work?**  
Docker uses containerization technology to isolate applications from the underlying operating system. This isolation makes it possible to run multiple applications on the same host without conflicts.

**Why Do We Need Docker?**

* **Consistency:** Ensures that an application behaves the same way across development, testing, and production environments.
* **Efficiency:** Containers share the host OS kernel, making them faster and more efficient than traditional virtual machines.
* **Scalability:** Simplifies scaling applications horizontally by running multiple container instances.

**Docker Architecture**  
Docker’s architecture comprises three main components:

* **CLI (Command Line Interface):**  
  Commands used to build, run, and manage Docker containers and images.
* **Daemon:**  
  A background service (dockerd) that manages Docker objects such as containers, images, networks, and volumes. The daemon listens for Docker API requests.
* **Registry:**  
  A storage and distribution system for Docker images. Docker Hub is the default public registry, though you can use private registries as well.

**1. Installing Docker on Ubuntu**

**Update Package List and Install Prerequisites**

sudo apt update

sudo apt install -y apt-transport-https ca-certificates curl software-properties-common

*Updates your package list and installs necessary tools for adding Docker’s repository.*

**Add Docker’s Official GPG Key and Repository**

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg

echo "deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list

*Downloads Docker’s GPG key and adds the Docker repository to your system.*

**Install Docker Engine, CLI, and Containerd**

sudo apt update

sudo apt install -y docker-ce docker-ce-cli containerd.io

*Installs Docker Community Edition along with the container runtime.*

**Allow Running Docker Commands Without sudo**

sudo usermod -aG docker $USER

*Adds your user to the Docker group, allowing you to run Docker commands without using sudo (log out and log back in for changes to take effect).*

**2. Project Setup and File Structure**

**Create Your Project Directory and Files**

mkdir python-docker-project

cd python-docker-project

mkdir src tests

touch src/\_\_init\_\_.py

touch src/main.py

touch requirements.txt

touch Dockerfile

touch .dockerignore

touch docker-compose.yml

*Creates a new project directory with a basic file structure for your application.*

**Open the Project in Your Editor**

code .

*Opens the project directory in Visual Studio Code (or your preferred editor).*

**3. Working with Docker Images and Containers**

**Build Your Docker Image**

docker build -t python-docker-app .

*Builds an image from your Dockerfile and tags it as python-docker-app.*

**Run Your Docker Container**

docker run -p 5000:5000 python-docker-app

*Starts a container from your image and maps port 5000 on the container to port 5000 on your host, making your application accessible at http://localhost:5000.*

**4. Using Docker Compose**

**Installing Docker Compose**

sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose

docker-compose --version

*Downloads, installs, and verifies Docker Compose.*

**Build and Run with Docker Compose**

docker-compose up --build

*Builds your images (if needed) and starts your application as defined in your docker-compose.yml file.*

**5. Managing Docker Hub Repository and Pushing Images**

**Log in to Docker Hub**

docker login

*Logs you into Docker Hub. Follow the prompts for authentication.*

**Tag Your Local Image for Docker Hub**

Assuming your Docker Hub repository name is utkarshkumar10/utkarshkumar:

docker tag python-docker-project-web:latest utkarshkumar10/utkarshkumar:latest

*Tags your local image so it is ready to be pushed to Docker Hub.*

**Push Your Image to Docker Hub**

If needed, run with sudo (or ensure your user is added to the Docker group):

sudo docker push utkarshkumar10/utkarshkumar:latest

*Pushes the tagged image to your Docker Hub repository.*

*Note:*  
If you encounter errors such as “denied: requested access to the resource is denied,” verify that you’re logged into the correct Docker Hub account and that the repository exists. You may also need to log in with sudo if running as root.

**6. Cleaning Up Docker Images**

**List Docker Images**

docker images

*Displays all your local Docker images.*

**Remove a Dangling Image**

If an image appears with <none> as the repository and tag:

sudo docker image prune

*Removes dangling images (unused layers).*

**Removing a Specific Image**

If an image is still referenced by a stopped container, remove the container first:

1. List containers:
2. docker ps -a
3. Remove the container (replace <container\_id> with the actual ID):
4. docker rm <container\_id>
5. Remove the image:
6. docker rmi <image\_id>

*Alternatively, force removal with:*

docker rmi -f <image\_id>

**7. Python Virtual Environment and Project Setup**

**Create and Activate a Virtual Environment**

python3 -m venv env

source env/bin/activate

*Creates and activates a Python virtual environment.*

**Install Python Dependencies**

pip install -r requirements.txt

*Installs the required Python packages as specified in requirements.txt.*

**Final Notes**

* **Docker Permissions:**  
  If you encounter “permission denied” errors when accessing /var/run/docker.sock, either use sudo or add your user to the Docker group:
* sudo usermod -aG docker $USER

Then log out and log back in.

* **Repository Naming:**  
  When tagging images for Docker Hub, use the format:
* <dockerhub-username>/<repository-name>:<tag>

For example, utkarshkumar10/utkarshkumar:latest.

* **Troubleshooting Push Errors:**  
  If you see errors like “denied: requested access to the resource is denied,” confirm that you’re logged into the correct Docker Hub account and that the repository exists on Docker Hub.