**Project Overview**

In this project, A web application will be deployed to leveraging multiple Docker images. We will orchestrate a multi-container setup using Docker Compose, hosting the web application containers environment on the AWS cloud to make it accessible via the public internet.

**Project Setup**

1. **Web App Selection:** Use a suitable web application or find one on GitHub that meets the project requirements.
2. **Microservices Architecture:** Design the microservices architecture for the web application, ensuring they communicate effectively.
3. **Dockerization:** Write Dockerfiles for each component of the web application, specifying the build instructions and dependencies.
4. **DockerHub Account:** Create a DockerHub account if you don't have one already, and configure Docker to authenticate with DockerHub.
5. **Multi-Architecture Build:** Use Docker Buildx or similar tools to build Docker images for both x86\_64 and arm64 architectures, and push these images to DockerHub.
6. **AWS Deployment:** Set up an AWS account and navigate to the EC2 dashboard to launch a new EC2 instance.
7. **EC2 Instance Setup:** Launch a new EC2 instance, selecting an appropriate instance type and ensuring it meets the requirements for hosting Docker containers.
8. **Docker Engine and Docker Compose Installation:** Connect to the EC2 instance via SSH and install Docker Engine and Docker Compose following the official documentation.
9. **Compose Configuration:** Create a docker-compose.yaml file in your project directory, defining the services, networks, and volumes for your multi-container setup.
10. **Container Launch:** Use Docker Compose to deploy and start the containers on the EC2 instance, and verify accessibility via the public internet.

version: '3.8'

services:

api:

image: <your\_api\_image>:latest

ports:

- "80:80" # Assuming your API service runs on port 80

environment:

- DATABASE\_URL=<your\_database\_connection\_string>

depends\_on:

- database

networks:

- webnet

database:

image: <your\_database\_image>:latest

environment:

- POSTGRES\_DB=<your\_database\_name>

- POSTGRES\_USER=<your\_database\_user>

- POSTGRES\_PASSWORD=<your\_database\_password>

networks:

- webnet

networks:

webnet: