**Containerization**

1. **Install Docker and Docker Compose**

**# On Ubuntu:**

**sudo apt update**

**sudo apt install -y docker.io**

**sudo systemctl enable --now docker**

**# Download the latest version of Docker Compose**

sudo curl -L "https://github.com/docker/compose/releases/download/$(curl -s https://api.github.com/repos/docker/compose/releases/latest | grep -oP '"tag\_name": "\K(.\*)(?=")')/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

**# Make it executable**

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

**Verify Installation**:

docker –version

docker-compose --version

1. **Create a Docker Compose File**
2. **Create a Working Directory**

**#mkdir wordpress-docker**

**#cd wordpress-docker**

**4. Run the Docker Compose Setup**:

In the terminal, navigate to the directory containing our docker-compose.yml file, and run:

{docker-compose up -d}

This command will:

* Download the WordPress and MySQL images if these are not already on our system.
* Create and start the containers in detached mode.

**4-Access WordPress**:

* Once the containers are running, we can access our WordPress site by navigating to http://localhost:8080 in our browser. We will see the WordPress installation screen where we can configure our new site.

**5-Persistent Storage**:

* The wordpress\_data volume will ensure that any changes made to the wp-content directory (such as installing plugins, uploading media, etc.) are retained even if the container is restarted.
* Similarly, db\_data keeps the MySQL data persistent.

**CI/CD Pipeline**

**Set Up Jenkins**

**1. Install Jenkins**:

* Install Jenkins on a server (local or cloud, such as an EC2 instance).
* Ensure Jenkins has Docker and AWS CLI installed, as well as permissions to access your Docker daemon and AWS resources.

**2. Install Plugins**:

* In Jenkins, go to **Manage Jenkins > Manage Plugins** and install the following plugins:
  + **Docker**: For building Docker images.
  + **Pipeline**: For writing Jenkinsfiles.
  + **AWS Pipeline** or **AWS CLI**: For interacting with AWS services.

**3. Create a Jenkins Pipeline (Jenkinsfile)**

**4. Configure Environment Variables**

* Store sensitive data in Jenkins as credentials. We can add these under Manage Jenkins > Manage Credentials.
* Use credentials() function in the Jenkinsfile to reference these values securely.

**5. Set Up AWS Resources**

**AWS ECR**:

* + Create an ECR (Elastic Container Registry) repository for both my\_wordpress\_app and my\_mysql\_db.
  + Update <your-account-id> and AWS\_REGION in the Jenkinsfile accordingly.

**IAM Permissions**:

* Ensure the Jenkins server has IAM permissions for ECR and ECS actions (ecr:GetAuthorizationToken, ecs:UpdateService, etc.)

**6. Basic Testing**

* The **test stage** in the Jenkinsfile uses curl to check if the WordPress homepage loads successfully.
* This simple test ensures that the WordPress container runs correctly before deploying.

**7. Run the Jenkins Pipeline**

* In Jenkins, create a new pipeline job and point it to your Git repository containing the Jenkinsfile.
* Start the job to trigger the CI/CD pipeline, which will build, test, and deploy our WordPress and MySQL containers to AWS.