DevOps

Day – 2

Assignment

Name : DEEPIKA P M

Roll No:22CSR041

**Project Setup**:

* Define the application architecture (e.g., microservices, monolith).
* Write application code and organize it into repositories.

**Create Dockerfiles**:

* Write Dockerfile for each service to define how containers should be built.
* Include necessary dependencies and configurations.

**Docker Compose**:

* Create a docker-compose.yml file to manage multiple services using Docker Compose.

**Containerization**:

* Build Docker images using docker build.
* Run containers using docker run.

**Version Control**:

* Use Git for source code management and maintain Dockerfile versions.

**CI/CD Pipeline**:

* Configure CI/CD pipelines using tools like Jenkins, GitLab CI, or GitHub Actions to automate builds, tests, and deployments.

**Container Registry**:

* Push Docker images to Docker Hub, AWS ECR, or other container registries.

**Orchestration**:

* Use Kubernetes or Docker Swarm to manage and orchestrate containers in production.

**Monitoring and Logging**:

* Integrate tools like Prometheus, Grafana, or ELK Stack for container monitoring and logging.

**Scaling and Management**:

* Implement load balancing and scaling using Kubernetes or other container orchestration tools.

**Commands :**

**Build Docker image:**

docker build -t my-app .

Create Docker Compose File

version: '3.8' services: web:

build: . ports: - "5000:5000" environment: - ENV=development volumes:

- .:/app

restart: always

docker-compose up

**Container Management**

Check running containers:

docker ps

Stop containers:

docker-compose down Restart containers:

docker-compose restart **Version Control** git init git add .

git commit -m "Initial commit" git branch -M main git remote add origin <repo-url> git push -u origin main

**Setup CI/CD Pipeline**

Install Jenkins using Docker: docker run -d -p 8080:8080 -p 50000:50000 jenkins/jenkins:lts

Configure pipeline using Jenkinsfile.

If using GitHub Actions:

Create .github/workflows/docker.yml

name: Docker Build and Push

on:

push: branches:

* main

jobs: build:

runs-on: ubuntu-latest steps:

* name: Checkout Code uses: actions/checkout@v4

* name: Build Docker Image run: docker build -t my-app:latest .

* name: Login to Docker Hub

run: echo "${{ secrets.DOCKER\_PASSWORD }}" | docker login -u "${{ secrets.DOCKER\_USERNAME }}" -password-stdin

* name: Push Docker Image

run: docker tag my-app:latest my-dockerhub-user/my-app:latest && docker push my-dockerhub-user/my-

app:latest  **Push to Docker Hub** Login to Docker Hub: docker login Tag the image:

docker tag my-app my-dockerhub-user/my-app:latest Push to Docker Hub:

docker push my-dockerhub-user/my-app:latest



