GANDAKI COLLEGE OF ENGINEERING AND SCIENCE

Lamachaur, Pokhara

LAB REPORT OF

Agile Software Development

LAB – 3

SUBMITTED BY:

Ansh Gurung

Roll No: 9

6th Semester

BE Software

SUBMITTED TO:

Er. Rajendra Bdr. Thapa

Objective

To explore and evaluate various deployment tools commonly used in Agile environments, focusing on automation, continuous integration/continuous deployment (CI/CD), and containerization.

Tools Used

- **Docker** – for containerization
- **Docker Compose** – for multi-container setups
- **Jenkins** – for automating the CI/CD pipeline
- **GitHub** – for version control and source code management

Methodology

The deployment process was divided into the following phases:

Phase 1: Containerization with Docker

- Created Dockerfiles for both frontend and backend applications.
- Used multi-stage builds for optimized container images.
- Configured .env files for environment-specific variables.

Phase 2: Docker Compose Setup

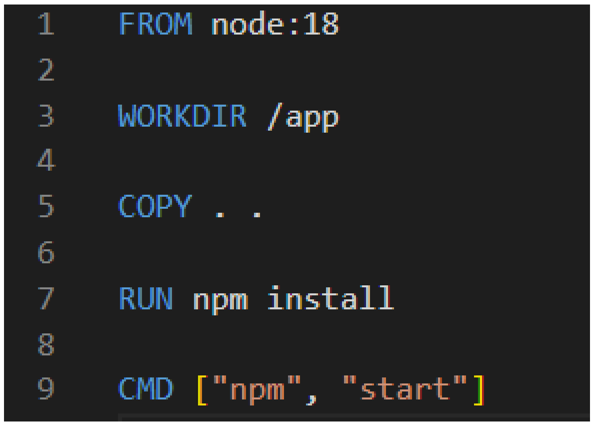
- Combined services using docker-compose.yml.
- Linked services: frontend, backend, database, and cache (Redis).
- Configured volumes and networks.

Phase 3: Jenkins CI/CD Pipeline

- Installed Jenkins and configured the pipeline using a Jenkinsfile.
- Set up automatic build triggers on GitHub push.
- Integrated build, test, and deployment stages.
- Monitored pipeline status and logs.

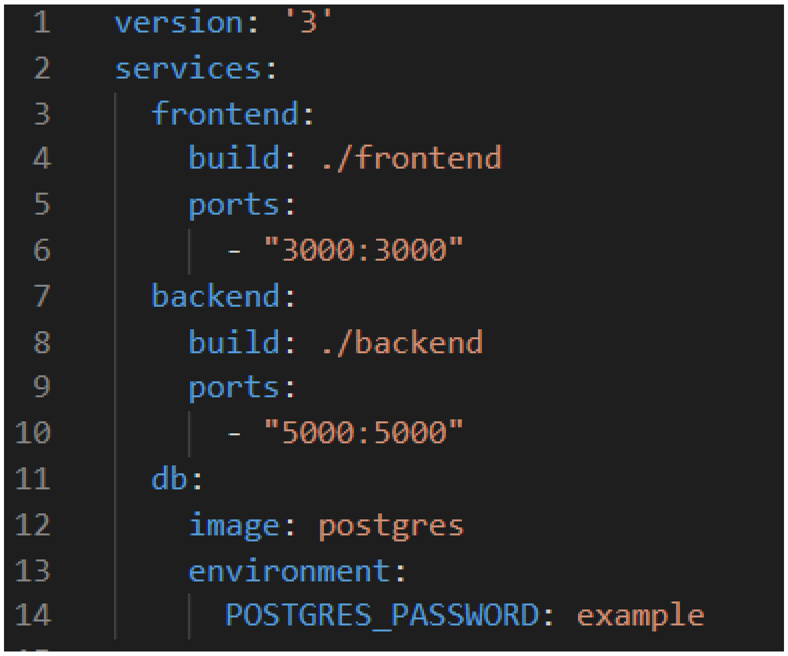
Implementation

- Dockerfile:



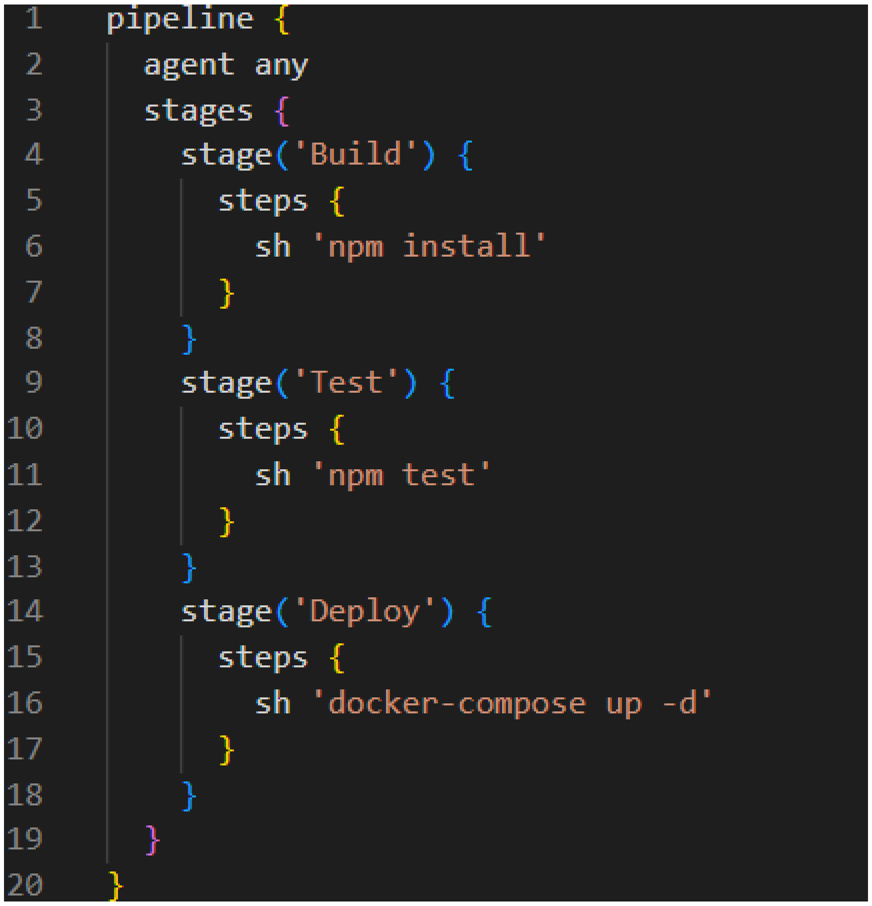
```dockerfile
FROM node:18

WORKDIR /app

COPY . .

RUN npm install

CMD ["npm", "start"]
```

- docker-compose.yml



```yaml
version: '3'
services:
  frontend:
    build: ./frontend
    ports:
      - "3000:3000"
  backend:
    build: ./backend
    ports:
      - "5000:5000"
  db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: example
```

- Jenkinsfile:



```
pipeline {
  agent any
  stages {
    stage('Build') {
      steps {
        sh 'npm install'
      }
    }
    stage('Test') {
      steps {
        sh 'npm test'
      }
    }
    stage('Deploy') {
      steps {
        sh 'docker-compose up -d'
      }
    }
  }
}
```

Results

- Successfully containerized the application and deployed using Docker Compose.
- Jenkins pipeline triggered automatically on code push and completed build-test-deploy cycle without errors.
- Reduced deployment time and eliminated manual errors.
- Enabled seamless integration between development and operations teams.

Conclusion

Deployment tools like Docker and Jenkins greatly enhance Agile workflows by promoting automation, repeatability, and efficiency. Through this lab, we gained hands-on experience in setting up containers, automating pipelines, and aligning deployment practices with Agile principles.