🚀 Automated CI/CD Pipeline for Web Application

This project demonstrates a complete **CI/CD (Continuous Integration & Continuous Deployment)** pipeline implementation using **GitHub Actions, AWS, and Docker**. The goal is to automate the entire software delivery workflow — from code commit to deployment — ensuring consistency, faster delivery, and improved reliability.

📌 Project Overview

This project implements an automated CI/CD pipeline for a sample **Java/Python web application**. Whenever code is pushed to GitHub, the pipeline automatically: - Builds the application - Runs unit tests - Performs linting & quality checks - Builds Docker images - Stores build artifacts - Deploys to AWS EC2 - Sends pipeline status notifications via AWS SNS

This pipeline follows modern DevOps practices to create a robust, efficient, and secure deployment workflow.

🧱 Architecture Overview

```
Developer Pushes Code → GitHub Actions CI Pipeline → Docker Build → Artifact Storage → AWS Deployment → SNS Notifications
```

CI: GitHub Actions
CD: Deployment to AWS EC2 via SSH + Docker
Notifications: AWS SNS email alerts

⚙️ Technologies Used

- **GitHub Actions** – CI automation
- **Docker** – Containerization
- **AWS EC2** – Application hosting
- **AWS S3** – Artifact & static file storage
- **AWS SNS** – Notification service
- **Git & GitHub** – Version control
- **Python / Java** – Application backend
- **Linux** – Server configuration

CI/CD Workflow Steps

1. Continuous Integration (CI)

Triggered on every push or pull request: - Checkout repository - Install dependencies - Run tests - Lint and format code - Build and package application (JAR/ZIP) - Upload build artifacts - Build Docker image - Push Docker image to Docker Hub or AWS ECR

2. Continuous Deployment (CD)

Triggered when CI is successful: - SSH into AWS EC2 - Pull latest Docker image - Stop existing container - Start new container - Verify health status of the application - Send success/failure notifications via SNS

Repository Structure

```
├── .github/workflows/
│   └── ci-cd.yml
├── app/
│   ├── src/
│   ├── tests/
│   └── requirements.txt / pom.xml
├── Dockerfile
├── README.md
└── deploy/
    └── scripts
```

Docker Setup

Sample Dockerfile

```
FROM python:3.9-slim
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
CMD ["python", "app.py"]
```

Sample GitHub Actions Workflow (ci-cd.yml)

```
name: CI/CD Pipeline

on:
  push:
    branches: [ "main", "dev" ]
```

```
  pull_request:
    branches: [ "main" ]

jobs:
  build:
    runs-on: ubuntu-latest

    steps:
    - uses: actions/checkout@v3

    - name: Set up Python
      uses: actions/setup-python@v4
      with:
        python-version: "3.9"

    - name: Install dependencies
      run: |
        pip install -r requirements.txt

    - name: Run tests
      run: |
        pytest

    - name: Build Docker image
      run: |
        docker build -t my-app:latest .

    - name: Deploy to EC2
      run: |
        ssh -o StrictHostKeyChecking=no ${{ secrets.AWS_EC2 }} "docker pull
my-app:latest && docker restart app-container"
```

AWS SNS Notifications

SNS notifies the team on: - Build success - Build failure - Deployment success - Deployment failure

This improves visibility and response time during releases.

Results & Impact

- Deployment time reduced from **40 minutes to under 5 minutes**
- Fully automated release cycle
- Higher code quality due to automated testing

- Reduced production issues with Dockerized environment
- Improved team efficiency and reliability

📌 Future Enhancements

- Add Terraform for Infrastructure-as-Code (IaC)
- Add monitoring using AWS CloudWatch
- Implement rolling deployments
- Add SonarQube for advanced code quality reports

👩‍💻 About the Author

Dhanushpriya T
Computer Science Engineer passionate about DevOps, automation, and cloud technologies.