Week 9: Jenkins CD + Deployment

CSI403 Full Stack Development | Lab 8 (8%)

Semester 1/2569

Agenda

CI vs CD

CI - Continuous Integration

- Checkout code
- Install dependencies
- Run tests
- Generate reports

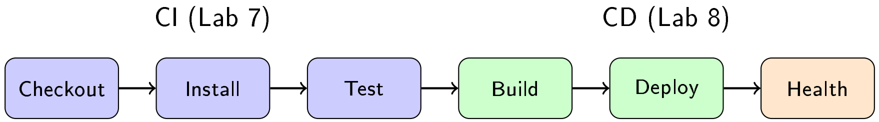
Already done in Lab 7!

CD - Continuous Deployment

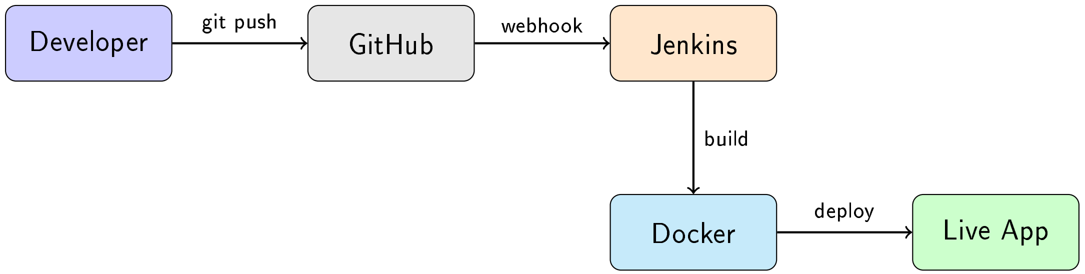
- Build Docker image
- Push to registry
- Deploy to server
- Health check

This week's focus!

Complete Pipeline



Deployment Architecture



1. Developer pushes code to GitHub
2. GitHub webhook triggers Jenkins
3. Jenkins builds and tests
4. Jenkins deploys new Docker container

Stage: Build Docker Image

```
stage('Build␣Docker␣Image') {
    when {
        branch 'main'
    }
    steps {
        sh '''
␣␣␣␣␣␣␣␣␣␣␣␣docker␣build␣-t␣taskflow:${BUILD_NUMBER}␣.
␣␣␣␣␣␣␣␣␣␣␣␣docker␣tag␣taskflow:${BUILD_NUMBER}␣taskflow:latest
␣␣␣␣␣␣␣␣'''
    }
}
```

Note: Only builds when on main branch

Stage: Stop Old Container

```
stage(’Stop␣Old␣Container’) {
    when {
        branch ’main’
    }
    steps {
        sh ’’’
␣␣␣␣␣␣␣␣␣␣␣␣echo␣"Stopping␣old␣container..."
␣␣␣␣␣␣␣␣␣␣␣␣docker␣stop␣taskflow-app␣||␣true
␣␣␣␣␣␣␣␣␣␣␣␣docker␣rm␣taskflow-app␣||␣true
␣␣␣␣␣␣␣␣’’’
    }
}
```

Note: `|| true` prevents failure if container doesn’t exist

Stage: Deploy New Container

```
stage('Deploy␣New␣Container') {
    when {
        branch 'main'
    }
    steps {
        sh '''
␣␣␣␣␣␣␣␣␣␣␣␣docker␣run␣-d␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--name␣taskflow-app␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--network␣taskflow-network␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣-p␣8000:8000␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣-e␣DATABASE_URL="${DATABASE_URL}"␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--restart␣unless-stopped␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣taskflow:latest
␣␣␣␣␣␣␣␣'''
    }
}
```

Stage: Health Check

```
stage('Health␣Check') {
    when {
        branch 'main'
    }
    steps {
        sh '''
␣␣␣␣␣␣␣␣␣␣␣␣echo␣"Waiting␣for␣app␣to␣start..."
␣␣␣␣␣␣␣␣␣␣␣␣sleep␣10

␣␣␣␣␣␣␣␣␣␣␣␣for␣i␣in␣1␣2␣3␣4␣5;␣do
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣curl␣-f␣http://localhost:8000/health;␣then
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣echo␣"App␣is␣healthy!"
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣exit␣0
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣fi
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣echo␣"Attempt␣$i␣failed,␣waiting..."
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣sleep␣5
␣␣␣␣␣␣␣␣␣␣␣␣done

␣␣␣␣␣␣␣␣␣␣␣␣echo␣"Health␣check␣failed!"
␣␣␣␣␣␣␣␣␣␣␣␣exit␣1
␣␣␣␣␣␣␣␣'''
    }
```

Post Actions

```
post {
    success {
        echo 'Pipeline␣completed␣successfully!'
        echo 'Application␣is␣live␣at␣http://localhost:8000'
    }
    failure {
        echo 'Pipeline␣failed!'
        // Could add: send email, Slack notification, etc.
    }
    always {
        // Clean up workspace
        cleanWs()
    }
}
```

Rollback on Failure

```
post {
    failure {
        sh '''
␣␣␣␣␣␣␣␣␣␣␣␣echo␣"Deployment␣failed!␣Rolling␣back..."

␣␣␣␣␣␣␣␣␣␣␣␣#␣Stop␣failed␣container
␣␣␣␣␣␣␣␣␣␣␣␣docker␣stop␣taskflow-app␣||␣true
␣␣␣␣␣␣␣␣␣␣␣␣docker␣rm␣taskflow-app␣||␣true

␣␣␣␣␣␣␣␣␣␣␣␣#␣Start␣previous␣version
␣␣␣␣␣␣␣␣␣␣␣␣docker␣run␣-d␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--name␣taskflow-app␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣--network␣taskflow-network␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣-p␣8000:8000␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣taskflow:previous

␣␣␣␣␣␣␣␣␣␣␣␣echo␣"Rolled␣back␣to␣previous␣version"
␣␣␣␣␣␣␣␣'''
    }
}
```

Complete Jenkinsfile (Part 1)

```
pipeline {
    agent any

    environment {
        DATABASE_URL = credentials('database-url')
    }

    stages {
        stage('Checkout') {
            steps {
                checkout scm
            }
        }

        stage('Install') {
            steps {
                sh 'pip␣install␣-r␣requirements.txt'
            }
        }
```

Complete Jenkinsfile (Part 2)

```
        stage('Test') {
            steps {
                sh 'pytest␣--junitxml=results.xml␣-v'
            }
            post {
                always { junit 'results.xml' }
            }
        }

        stage('Build') {
            when { branch 'main' }
            steps {
                sh 'docker␣build␣-t␣taskflow:${BUILD_NUMBER}␣.'
            }
        }
```

Complete Jenkinsfile (Part 3)

```
        stage('Deploy') {
            when { branch 'main' }
            steps {
                sh 'docker␣stop␣taskflow-app␣||␣true'
                sh 'docker␣rm␣taskflow-app␣||␣true'
                sh '''
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣docker␣run␣-d␣--name␣taskflow-app␣\
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣-p␣8000:8000␣taskflow:${BUILD_NUMBER}
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣'''
            }
        }

        stage('Health␣Check') {
            when { branch 'main' }
            steps {
                sh 'sleep␣10␣&&␣curl␣-f␣http://localhost:8000/health'
            }
        }
    }
}
```

Configure Jenkins Credentials

Steps:

1. Go to Jenkins > Manage Jenkins > Credentials
2. Click “Add Credentials”
3. Kind: Secret text
4. ID: database-url
5. Secret: Your database connection string
6. Save

Configure GitHub Webhook

In GitHub Repository:

1. Settings > Webhooks > Add webhook
2. Payload URL: http://your-jenkins:8080/github-webhook/
3. Content type: application/json
4. Events: Just push events
5. Active: Yes

Pipeline Trigger Options

- **GitHub webhook** - Trigger on push (recommended)
- **Poll SCM** - Check for changes periodically
- **Manual** - Click “Build Now” button
- **Scheduled** - Cron-like schedule

Phase 1 Complete!

Congratulations!

You have built a complete Full Stack application!

What you built:

- FastAPI backend with CRUD operations
- MSSQL database with SQLAlchemy ORM
- Jinja2 templates with Bootstrap UI
- Docker containers
- Automated tests with pytest
- CI/CD pipeline with Jenkins

Phase 1 Score Summary

Lab	Topic	Weight
1	Git + Python + Setup	8%
2	FastAPI CRUD	8%
3	Database Integration	8%
4	Frontend Basics	8%
5	Jinja2 Templates	8%
6	Docker + Compose	8%
7	Testing + CI	8%
8	CD + Deployment	8%
	Total Phase 1	**64%**

Phase 2 Preview: Group Project (36%)

Week	Activity	Weight
10	Team Formation + Proposal	5%
11	System Design	5%
12	Checkpoint Demo	8%
13-14	Development Sprints	-
15	Final Presentation	18%

Requirements:

- Team of 3-4 students
- Use all technologies from Phase 1
- Your own project idea

Lab 8: Jenkins CD (8%)

Requirements:

1. Complete Jenkinsfile with all stages
2. Configure credentials in Jenkins
3. Pipeline triggers on push to main
4. Health check verifies deployment
5. Post actions for success/failure
6. Document setup in README

Lab 8 Grading Rubric

Criteria	**Points**
CI stages (Checkout, Install, Test)	2%
CD stages (Build, Deploy)	2%
Health Check stage	1.5%
Post actions + Error handling	1%
Pipeline runs successfully	1%
Documentation	0.5%
Total	**8%**

Questions?

Phase 1 Complete!

Push - Test - Build - Deploy - Live!

Next Week: Form Teams + Project Proposal