Lab 2: Implement a Single-Stage GitHub Actions Pipeline

Lab overview

In this lab, you will create a simple, but complete, CI/CD pipeline using GitHub Actions. You'll build a basic Node.js web application, write simple tests, and create workflows that demonstrate how code moves from development to production. This lab focuses on understanding the core concepts of CI/CD pipelines and automated testing without overwhelming complexity. By the end of this lab, you'll have hands-on experience with professional DevOps practices in a beginner-friendly format.

In this lab, you will:

- Create a simple Node.js web application with basic API endpoints
- Write easy-to-understand tests using Jest
- Build GitHub Actions workflows
- Implement automated testing and deployment simulation
- Monitor and troubleshoot pipeline execution

Estimated completion time

60 minutes

Task 1: Setting up your project

In this task, you will create a simple project structure and GitHub repository for your CI/CD pipeline.

1. Create a folder named **Lab2** on your Desktop.
2. Create a GitHub Repository.
   2.1. Go to **github.com** and sign in.
   2.2. Click **New** to create a new repository.
      - **Name:** simple-cicd-pipeline
      - **Description:** Simple CI/CD pipeline with GitHub Actions
   2.3. Make it public (free GitHub Actions).
   2.4. Check **Add a README file**.
   2.5. Click **Create repository**.
3. Clone the repository.
   3.1. Open a terminal and navigate to your Lab2 folder.

```
cd ~/Desktop/Lab2
```

   3.2. Clone your repository (replace *YOUR_USERNAME* with your GitHub username).

```
git clone https://github.com/YOUR_USERNAME/simple-cicd-pipeline.git
```

```
cd simple-cicd-pipeline
```

4. Create project folders.
   4.1. Create a basic project structure.

```
mkdir -p src tests .github/workflows
```

   4.2. Check the structure.

```
ls -la
```

Expected result: You should see **src/**, **tests/**, **.github/** folders along with **README.md**.

Task 2: Mapping real-world tasks to DevOps stages

In this task, you will build a basic web application with just a few endpoints.

> **Note**
>
> Sample files for all labs are located in the relevant lab folder under the **Desktop/Sample Lab Files** folder. These can be copied and pasted as required to assist with conducting these labs. Always review the contents against those listed in the lab steps.

1. Create a package.json file in the simple-cicd-pipeline directory (this could be via a Text Editor, VS Code, or copy and paste), with the following code.

```
{
  "name": "simple-cicd-pipeline",
  "version": "1.0.0",
  "description": "Simple web app for CI/CD learning",
  "main": "src/app.js",
  "scripts": {
    "start": "node src/app.js",
    "test": "jest",
    "build": "mkdir -p dist && cp src/*.js dist/"
  },
  "dependencies": {
    "express": "^4.18.2"
  },
  "devDependencies": {
    "jest": "^26.6.3",
    "supertest": "^6.1.6"
  },
```

```
  "jest": {
    "testEnvironment": "node"
  }
}
```

2. Create the main application file src/app.js.

```
const express = require('express');
const app = express();
const PORT = 3000;

// Simple welcome page
app.get('/', (req, res) => {
    res.json({
        message: 'Welcome to Simple CI/CD Demo!',
        version: '1.0.0'
    });
});

// Health check endpoint
app.get('/health', (req, res) => {
    res.json({
        status: 'healthy',
        timestamp: new Date().toISOString()
    });
});
```

```
// Simple API endpoint
app.get('/api/hello', (req, res) => {
    res.json({
        greeting: 'Hello from CI/CD Pipeline!',
        success: true
    });
});

// Start server (only if not in test mode)
if (process.env.NODE_ENV !== 'test') {
    app.listen(PORT, () => {
        console.log(`Server running on port ${PORT}`);
    });
}

module.exports = app;
```

3. From the Terminal window, install dependencies.

```
npm install
```

4. Build the application locally.

   4.1. Test the build.

```
npm run build
```

   4.2. Check if files were created.

```
ls -la dist/
```

Expected result: You should see app.js copied to the dist/ folder.

```
student@labuser-virtual-machine:~/Desktop/Lab2/simple-cicd-pipeline$ ls -la dist/
total 12
drwxrwxr-x 2 student student 4096 Oct 29 08:25 .
drwxrwxr-x 7 student student 4096 Oct 29 08:25 ..
-rw-rw-r-- 1 student student  790 Oct 29 08:25 app.js
student@labuser-virtual-machine:~/Desktop/Lab2/simple-cicd-pipeline$
```

Task 3: Creating simple tests

In this task, you will write basic tests to ensure your application works correctly.

1. Create a test file tests/app.test.js.

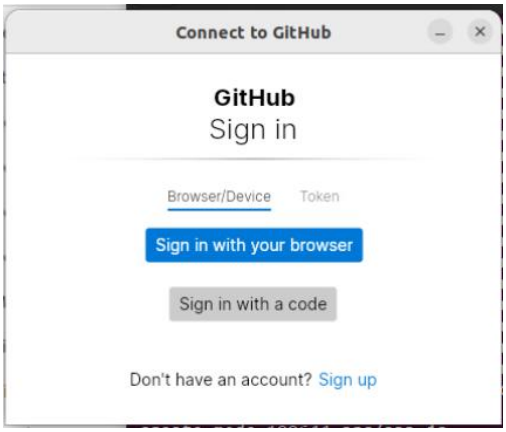
```
const request = require('supertest');

const app = require('../src/app');


describe('Simple App Tests', () => {

    test('Welcome page should work', async () => {

        const response = await request(app).get('/');

        expect(response.status).toBe(200);

        expect(response.body.message).toBe('Welcome to Simple CI/CD
Demo!');

    });


    test('Health check should work', async () => {

        const response = await request(app).get('/health');

        expect(response.status).toBe(200);

        expect(response.body.status).toBe('healthy');

    });


    test('API endpoint should work', async () => {

        const response = await request(app).get('/api/hello');
```

```
        expect(response.status).toBe(200);
        expect(response.body.success).toBe(true);
    });
});
```

2. Commit your basic application.

```
git config --global user.email "you@example.com"
git config --global user.name "Your Name"
git add .
git commit -m "Add simple Node.js application with tests"
git push origin main
```

3. If you are asked, sign in with your browser.



Task 4: Creating basic CI workflow

In this task, you will create your first GitHub Actions workflow that automatically tests your code.

1. Create .github/workflows/simple-ci.yml.

```
name: Simple CI
# When to run this workflow
on:
  push:
```

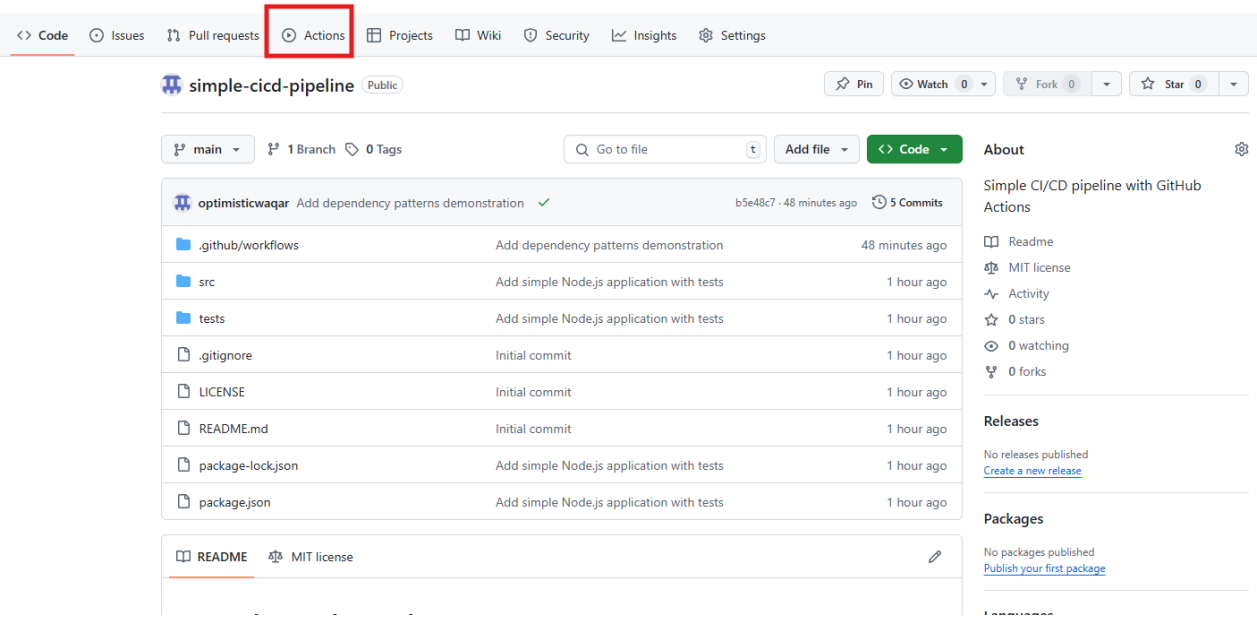
```
    branches: [ main ]
  pull_request:
    branches: [ main ]
jobs:
  test:
    name: Test Application
    runs-on: ubuntu-latest
    steps:
    # Get the code
    - name: Get Code
      uses: actions/checkout@v4
    # Setup Node.js
    - name: Setup Node.js
      uses: actions/setup-node@v4
      with:
        node-version: '18'
    # Install packages
    - name: Install Dependencies
      run: npm install

    # Run tests
    - name: Run Tests
      run: npm test

    # Build application
    - name: Build App
```

```
      run: npm run build

    # Show what was built
    - name: Show Build Results
      run: |
        echo "Build completed!"
        ls -la dist/
```

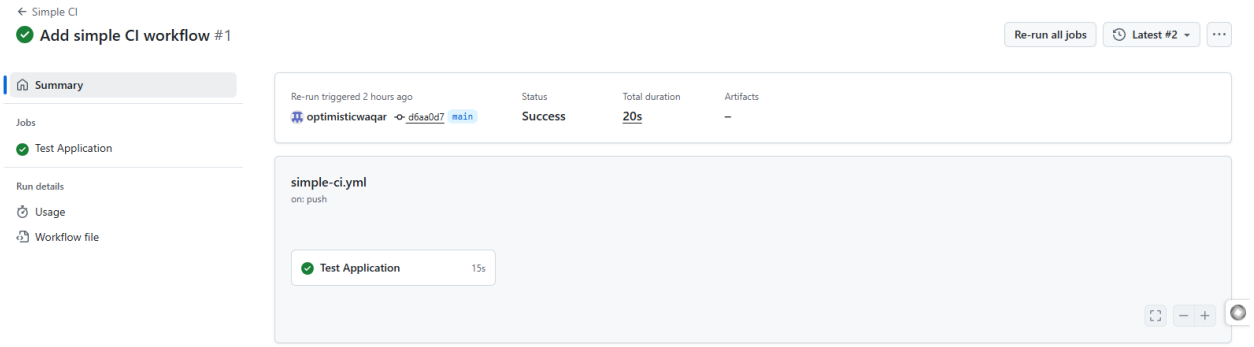
2. Commit and test the workflow.

```
git add .github/workflows/simple-ci.yml
git commit -m "Add simple CI workflow"
git push origin main
```

3. Check your workflow.
   3.1. Go to your GitHub repository.
   3.2. Click the **Actions** tab.



You should see your workflow running.



Expected result: The workflow should complete successfully with all green checkmarks.

Lab review

1. Which section of the GitHub Actions workflow defines when the workflow should be triggered?
   A. jobs
   B. steps
   C. on
   D. runs-on

STOP

You have successfully completed this lab.