Lab 1 : CI/CD Pipeline

Learning objectives

By the end of this lab, students will be able to:

- Create a basic Node.js app and automated tests.
- Configure a robust GitHub Actions CI workflow.
- Add linting and test steps to CI.
- Read workflow logs and debug common failures.

Pre-requisites (check before starting)

- GitHub account.
- Node.js (v16+). Confirm with `node -v`.
- `npm` or Yarn installed. Confirm with `npm -v` or `yarn -v`.
- Basic git skills (clone, commit, push).

I. Create Repository & Initialize Project

1. **Create a new GitHub repository**
   - Go to GitHub → click **New Repository**
   - Name it: `lastname-firstname-lab1`
   - Choose **Public**
   - Do **not** initialize with README
2. **Clone your repository**

   ```
   git clone https://github.com/yourusername/lastname-firstname-lab1.git
   cd lastname-firstname-lab1
   ```

3. **Initialize a new Node.js project using Yarn**

   ```
   yarn init -y
   yarn add express
   yarn add --dev mocha chai supertest eslint
   ```

II. Create Application Files

1. app.js

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

app.get('/', (req, res) => {
```

```
  res.send('Hello, GitHub Actions!');
});

module.exports = app;
```

2. server.js

```
const app = require('./app');
const port = process.env.PORT || 3000;

app.listen(port, () => {
  console.log(`App listening at http://localhost:${port}`);
});
```

3. test/test.js

```
const request = require('supertest');
const app = require('../app');
const { expect } = require('chai');

describe('GET /', () => {
  it('should return Hello, GitHub Actions!', async () => {
    const res = await request(app).get('/');
    expect(res.status).to.equal(200);
    expect(res.text).to.equal('Hello, GitHub Actions!');
  });
});
```

4. .eslintrc.json

```
{
  "env": { "node": true, "mocha": true, "es2021": true },
  "extends": "eslint:recommended",
  "parserOptions": { "ecmaVersion": 12 },
  "rules": {}
}
```

5. Update `package.json` Scripts

```
"scripts": {
  "start": "node server.js",
  "test": "mocha --exit",
  "lint": "eslint . --ext .js"
}
```

III. Run Locally (Before CI)

Before pushing to GitHub, ensure everything works locally.

1. Start the application

```
yarn start
```

Visit http://localhost:3000
✅ Expected output in browser:

```
Hello, GitHub Actions!
```

2. Lint your code

```
yarn lint
```

✅ Expected: No linting errors.

3. Run automated tests

```
yarn test
```

✅ Expected:

```
  GET /
    ✓ should return Hello, GitHub Actions! (xx ms)

  1 passing (xx ms)
```

IV. Commit & Push

```
git add .
git commit -m "Initial Node.js project with test and lint"
git branch -M main
git remote add origin https://github.com/yourusername/lastname-firstname-lab1.git
git push -u origin main
```

V. Set Up GitHub Actions Workflow

1. **Create Workflow Directory**

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

2. **Create Workflow File** `.github/workflows/ci.yml`

   ```
   name: Node.js CI (Yarn)

   on:
     push:
       branches: [ main ]
     pull_request:
       branches: [ main ]

   jobs:
     build-and-test:
       runs-on: ubuntu-latest
       strategy:
         matrix:
           node-version: [16.x, 18.x]

       steps:
         - name: Checkout code
           uses: actions/checkout@v4

         - name: Setup Node.js
   ```

```
      uses: actions/setup-node@v4
      with:
        node-version: ${{ matrix.node-version }}
        cache: 'yarn'

    - name: Install dependencies
      run: yarn install --frozen-lockfile

    - name: Lint code
      run: yarn lint

    - name: Run tests
      run: yarn test
```

3. **Commit & Push**

```
git add .github/workflows/ci.yml
git commit -m "Add GitHub Actions CI workflow (Yarn)"
git push
```

VI. Monitor Your Workflow

1. Go to your repository on GitHub → **Actions** tab
2. Observe your workflow running automatically
3. Click the workflow → open logs for each step (checkout, setup, install, lint, test)
4. ✅ Success = All steps pass with green checkmarks

VII. Test Workflow Trigger

- Modify your code slightly (e.g., change message to `"Hello, CI/CD!"`)
- Commit & push again:
- `git add app.js`
- `git commit -m "Update greeting message"`
- `git push`
- Go back to **Actions** → confirm a new workflow run starts automatically.

IX. Advanced CI/CD Enhancements

1. Add Code Coverage Reporting

- Add coverage report using `nyc` + `mocha` and upload as a GitHub Actions artifact.
- You're adding a **code coverage** step to your CI workflow that measures **how much of your code is actually tested** when running your test suite.
- In GitHub Actions, **artifacts** are files generated during a workflow run that you can download later**.** E.g. test reports, logs, or coverage summaries.

2. Add Pull Request Workflow

Adjust the GitHub Actions workflow above to runs tests and linting on all pull requests before merging.

3. **Add Deployment Stage**
   Add a deployment step to Render, Vercel, Railway or Heroku to automatically deploy the app after all tests pass.

VIII. Submission

Submit a report that includes:

- **GitHub Repository**: Submit your full Node.js project including `.github/workflows/ci.yml` and enhancements (code coverage, PR workflow, deployment).
- **Evidence of CI/CD**: Screenshots or links showing workflow runs, linting, test results, code coverage, and deployment.
- **Reflection**: 200–300 words describing:
  - Challenges faced
  - How you debugged workflow failures
  - Key learnings about CI/CD and GitHub Actions
  - Ideas for improvement (optional)