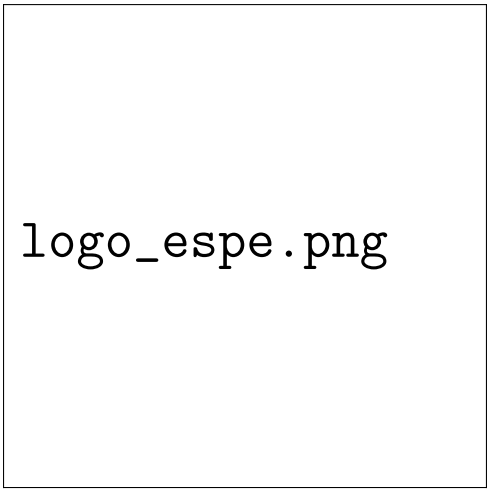

Laboratorio 6: CI/CD usando GitHub Actions

Universidad de las Fuerzas Armadas ESPE

Estudiante: [Gabriel Murillo]
Instructor: Ing. Enrique Calvopiña E, Mgtr.

22 de enero de 2026

1. Introducción

Este documento detalla el proceso de configuración de un flujo de Integración Continua (CI) y Entrega Continua (CD) utilizando GitHub Actions. Se implementan pruebas unitarias con Jest y análisis estático con ESLint sobre un servidor Express básico.

2. Objetivos

- Configurar un flujo de CI en GitHub Actions para automatizar pruebas y linting.
- Implementar pruebas unitarias utilizando Jest.
- Aplicar análisis estático de código con ESLint.

3. Desarrollo

3.1. Parte 1: Establecimiento de la estructura del proyecto base

Se creó la estructura del proyecto incluyendo archivos de configuración y dependencias. Se utilizó el puerto 3111 para el servidor.

[Espacio para captura de pantalla: Estructura de archivos]

3.2. Parte 2: Creación de archivos base

A continuación se presenta el código fuente desarrollado:

3.2.1. package.json

```
{
  "name": "lab6-ci-cd",
  "version": "1.0.0",
  "main": "index.js",
  "type": "module",
  "scripts": {
    "start": "node index.js",
    "test": "node --experimental-vm-modules node_modules/jest/bin/jest.
    js",
    "lint": "eslint ."
  }
}
```

3.2.2. index.js

```
import express from 'express';
const app = express();
const port = 3111;

app.get('/', (req, res) => {
  res.send('CI/CD Lab - GitHub Actions');
});

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

3.3. Parte 3: Configuración de Git y CI/CD

Esta sección cubre los 4 pasos fundamentales solicitados:
Paso 1: Crear repositorio en la cuenta de Git.

- a. Abrir la cuenta de Git en el navegador.
- b. Crear un nuevo repositorio vacío.

Paso 2: Ejecución de comandos para clonar al repositorio.

- a. `git init`
- b. `git add .`
- c. `git commit -m "Proyecto base con CI"`
- d. `git branch -M main`
- e. `git remote add origin https://github.com/USUARIO/REPO.git`
- f. `git push -u origin main`

Paso 3: Crear el workflow de GitHub Actions.

- a. Crear el archivo `.github/workflows/ci.yml`.
- b. Configurar disparadores (push, pull_request).
- c. Definir trabajos (jobs) y pasos (checkout, install, lint, test).

Paso 4: Probar la CI.

- a. Realizar cambios en el código.
- b. Ejecutar push.
- c. Verificar en la pestaña .ªctions"de GitHub.

4. Resultados y Evidencias

En esta sección se deben incluir capturas de pantalla del flujo de GitHub Actions ejecutándose correctamente.

[Espacio para captura de pantalla: Workflow aprobado en GitHub]

5. Conclusiones

1. GitHub Actions facilita la integración de pruebas automatizadas en el flujo de desarrollo.

2. El uso de linting y pruebas unitarias garantiza un código más limpio y libre de errores básicos antes del despliegue.

6. Recomendaciones

1. Se recomienda configurar umbrales de cobertura de pruebas para asegurar la calidad del software.

2. Utilizar secretos de GitHub para manejar información sensible en los flujos de trabajo.