DevOps Internship Project Report

Project Title:

Automated CI/CD for React App using GitHub Actions and WSL Self-Hosted Runner

Intern:

Rakesh Kolipaka

Submission Date:

August 28, 2025

Table of Contents

---

1. Introduction

This project demonstrates a DevOps workflow for automating the build, test, and deployment of a React application using GitHub Actions. The workflow is executed via a self-hosted runner on Windows Subsystem for Linux (WSL), enabling fully local CI/CD without relying on cloud infrastructure.

2. Project Objectives

- Set up a React project using Vite.
- Integrate version control with Git and GitHub.
- Automate build, test, and deployment processes using GitHub Actions.
- Run workflows on a local self-hosted runner (WSL).
- Achieve automatic deployment updates upon every code push to the main branch.

---

3. Environment Setup

- System: Windows 10/11 with WSL enabled (Ubuntu).
- Tools Installed:
    - Node.js and npm
    - Vite (for React scaffolding)
    - Git

    - OpenSSH server
    - GitHub Actions Runner

---

4. React App Initialization

- Initialized new React app with Vite:

```
npm create vite@latest my-react-app -- --template react

cd my-react-app

npm install

npm run dev
```

- Added components (Header.jsx, Footer.jsx, Card.jsx) in the /src/components folder.

5. Version Control Integration

- Initialized local git repository.
- Connected to GitHub repository (rakeshkolipakaace/Devops).
- Pushed initial code and ongoing changes to main branch.

---

6. GitHub Actions Workflow Configuration

- Created .github/workflows/deploy.yml for CI/CD pipeline.
- Configured two jobs:

- **build-test:** Runs on self-hosted, checks out code, installs dependencies, runs linter/tests, builds app, uploads artifacts.
- **deploy:** Runs on self-hosted, SSH into server, runs deploy script to update live app.

Example deploy.yml:

```
name: React CI

on:
  push:
    branches:
      - main

jobs:
  build-test:
    runs-on: self-hosted
    steps:
      - name: Checkout
        uses: actions/checkout@v3
      - name: Setup Node.js
        uses: actions/setup-node@v3
```

```yaml
        with:
          node-version: "18"
      - name: Install dependencies
        run: npm install
      - name: Build
        run: npm run build
      - name: Upload build
        uses: actions/upload-artifact@v4
        with:
          name: react-build
          path: dist/

  deploy:
    runs-on: self-hosted
    needs: build-test
    steps:
      - name: Deploy to server
        run: ssh -o StrictHostKeyChecking=no rakesh@localhost "bash /home/rakesh/deploy.sh"
```

---

7. Self-Hosted Runner Setup (WSL)

- Created actions-runner directory.
- Downloaded and extracted the GitHub Actions Runner package for Linux.
- Registered runner using received token and ran ./run.sh.
- Verified runner shows as online in GitHub repo.
- Both build and deploy jobs now execute on the local runner.

---

8. Automated Build, Test, and Deploy Pipeline

- Upon each push to main, workflow:
  1. Builds the React app (npm run build)
  2. Runs tests (if configured)
  3. SSHs to localhost, executing deploy.sh script
  4. deploy.sh copies contents of dist/ to app's serving directory, optionally restarts the web server

Sample deploy.sh:

```
#!/bin/bash

SRC_DIR="/home/rakesh/my-react-app/dist"

DEST_DIR="/var/www/html"

cp -r $SRC_DIR/* $DEST_DIR/

echo "Deployment complete."
```

- Automatic updates verified by making changes in components like Header.jsx, pushing to main, and confirming updates appear at http://localhost.

---

9. Troubleshooting and Resolutions

- Faced issues with SSH keys, service configuration, and sudo permissions.
- Resolved runner registration 404 errors by using correct repo URL and token.
- Fixed sudo password prompts by adjusting user permissions or removing sudo where possible in scripts.
- Ensured all workflow jobs use the self-hosted runner to maintain local-only deployment.

---

10. Conclusion

A fully automated CI/CD workflow was implemented for a React app using GitHub Actions, utilizing a self-hosted runner within WSL. All deployments are triggered automatically on each code push, allowing rapid iteration and integration, with no cloud services or paid resources required.