DevOps Internship - Task 1 Submission Automate Code Deployment Using CI/CD Pipeline

Soumen Das

November 13, 2025

Abstract

This document outlines the solution for Task 1: "Automate Code Deployment Using CI/CD Pipeline (GitHub Actions)". The objective is to create a complete CI/CD pipeline that automatically builds a Docker image for a Node.js application, and pushes it to Docker Hub whenever changes are pushed to the `main` branch on GitHub.

1 Task Overview

- **Objective:** Set up a CI/CD pipeline to build and deploy a web app.
- **Tools:** GitHub, GitHub Actions, Node.js, Docker.
- **Deliverable:** A GitHub repository with a `.yml` CI/CD workflow file.

2 Solution Implementation

The solution consists of three core components:

1. A sample Node.js application (`app.js` and `package.json`).
2. A `Dockerfile` to containerize the application.
3. A GitHub Actions workflow file (`.github/workflows/main.yml`) to automate the build and push process.

2.1 1. Sample Node.js Application

First, a minimal Node.js application using the Express framework is created to serve as our web app.

`package.json`:

```
{
  "name": "nodejs-demo-app",
  "version": "1.0.0",
  "description": "Sample Node.js app for CI/CD pipeline",
  "main": "app.js",
  "scripts": {
    "start": "node app.js"
  },
  "dependencies": {
    "express": "^4.17.1"
  }
}
```

app.js:

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

app.get('/', (req, res) => {
  res.send('Hello, World! This is the CI/CD Task 1 submission by Soumen Das.');
});

app.listen(PORT, () => {
  console.log(`Server is running on port ${PORT}`);
});
```

2.2 2. Dockerfile

A `Dockerfile` is required to build a container image of our application. This file defines the steps to install dependencies and run the app.

Dockerfile:

```
# Use an official Node.js runtime as a parent image
FROM node:18-alpine

# Set the working directory in the container
WORKDIR /usr/src/app

# Copy package.json and package-lock.json (if available)
COPY package*.json ./

# Install project dependencies
RUN npm install

# Copy the rest of the application source code
COPY . .

# Expose the port the app runs on
EXPOSE 3000

# Define the command to run the application
CMD [ "npm", "start" ]
```

2.3 3. GitHub Actions CI/CD Workflow

This is the core of the task. The `main.yml` file defines the pipeline. It is placed in the `.github/workflows/` directory of the repository.

.github/workflows/main.yml:

```
# Name of the workflow
name: CI/CD Pipeline - Build and Push to Docker Hub

# Controls when the workflow will run
```

```yaml
on:
  # Triggers the workflow on push events to the "main" branch
  push:
    branches: [ "main" ]

# A workflow run is made up of one or more jobs
jobs:
  # This job builds the app and pushes it to Docker Hub
  build-and-push:
    # The type of runner that the job will run on
    runs-on: ubuntu-latest

    # Steps represent a sequence of tasks
    steps:
      # 1. Checks-out your repository under $GITHUB_WORKSPACE
      - name: Check out the repo
        uses: actions/checkout@v4

      # 2. Log in to Docker Hub
      # Uses secrets stored in GitHub settings
      - name: Log in to Docker Hub
        uses: docker/login-action@v3
        with:
          username: ${{ secrets.DOCKER_USERNAME }}
          password: ${{ secrets.DOCKER_PASSWORD }}

      # 3. Build and push Docker image
      - name: Build and push Docker image
        uses: docker/build-push-action@v5
        with:
          context: .
          push: true
          # Tag the image with the Docker Hub username and repo name
          # IMPORTANT: Change 'your-dockerhub-username' to your actual username
          tags: your-dockerhub-username/nodejs-demo-app:latest
```

3 Setup and Deployment Instructions

To use this solution, follow these steps:

1. **Create GitHub Repository:** Create a new GitHub repository (e.g., `devops-task-1`).
2. **Add Files:** Add the `app.js`, `package.json`, and `Dockerfile` to the root of your repository.
3. **Create Workflow:** Create a directory named `.github/workflows` and add the `main.yml` file inside it.
4. **IMPORTANT - Update `main.yml`:** In the `main.yml` file, you **must** change the line `tags: your-dockerhub-username/nodejs-demo-app:latest` to use your own Docker Hub username.
5. **Set GitHub Secrets:**
   - In your GitHub repository, go to **Settings** > **Secrets and variables** > **Actions**.
   - Click **New repository secret**.
   - Create a secret named `DOCKER_USERNAME` and enter your Docker Hub username as the value.

- Create another secret named `DOCKER_PASSWORD`. For the value, it is highly recommended to use a **Docker Hub Access Token** instead of your main password. You can generate one in your Docker Hub account settings.

6. **Trigger the Pipeline:** Commit and push all the files to the `main` branch of your GitHub repository.
7. **Verify:** Go to the **Actions** tab in your GitHub repository. You will see the workflow running. Once it completes successfully, check your Docker Hub account to find the new `nodejs-demo-app:latest` image.

4 Conclusion

This solution successfully fulfills all requirements of Task 1. It provides a simple Node.js application, a `Dockerfile` for containerization, and a complete, automated CI/CD pipeline using GitHub Actions that triggers on a push to the `main` branch, builds the image, and pushes it to Docker Hub.