CPP CMAKE CI/CD Demo - Project Documentation

GitHub User ID - https://github.com/Nshravankumar4

Repository Link: https://github.com/Nshravankumar4/cpp-cmake-ci-demo.git

---

Overview

This project demonstrates how to build a C++ application using CMake with Continuous Integration and Continuous Delivery (CI/CD) pipelines automated through GitHub Actions and Jenkins. Jenkins is securely exposed to GitHub via ngrok webhooks to trigger builds on code changes.

Folder Structure

text

```
cpp-cmake-ci-demo/
 ├── CMakeLists.txt
 ├── include/
 │   └── hello.h
 ├── src/
 │    ├── hello.cpp
 │   └── main.cpp
 ├── .github/
 │   └── workflows/
 │       └── build.yml
 ├── Jenkinsfile
 ├── README.md
└── .gitignore
```

What You Have Done: Process Summary

1. Project Initialization and Source Setup

- Created folder and file structure.
- Implemented basic "Hello World" C++ components:
    - Header file: include/hello.h
    - Source files: src/hello.cpp and src/main.cpp
- Configured CMakeLists.txt to build the project.
- Verified successful local build and execution.

2. GitHub Repository

- Initialized git repository locally.
- Committed all source code and configuration files.
- Created GitHub repository at https://github.com/Nshravankumar4/cpp-cmake-ci-demo.
- Pushed code to remote main branch.

3. GitHub Actions CI

- Added .github/workflows/build.yml workflow defining:
    - Checkout source code.
    - Install dependencies (CMake, g++).
    - Build project using CMake.
    - Run the executable.
- Workflow triggers on push or pull request events to main.
- Viewed build results and logs directly on GitHub Actions interface.

4. Jenkins Integration

- Installed Jenkins on a local machine, running on default port 8080.
- Created Jenkins Pipeline job sourcing code from GitHub repo main branch.
- Added a Jenkinsfile at repository root defining pipeline stages:

    - Checkout, build (via CMake), run, and archive build artifacts.
- Verified builds executed successfully and outputs were archived.

5. ngrok Setup & Webhook Configuration

- Configured ngrok with your authtoken:

text

ngrok config add-authtoken <>

ngrok http 8080

- Obtained public ngrok URL that tunnels to local Jenkins service.
- Set up GitHub webhook with:
    - Payload URL: https://margherita-unpolished-unanswerably.ngrok-free.dev/github-webhook/
    - Content Type: application/json
    - SSL verification enabled.
    - Trigger events: Push only.
- Webhook actively triggers Jenkins builds on each new push.

6. GitHub Branch Protection

- Enabled protection on main branch with rules:
    - Require pull requests for all merges.
    - Enforce linear commit history.
    - Require passing GitHub Actions status checks.
    - Block force pushes and branch deletions.
    - Enforce signed commits.
    - Allow bypass for admins and specified apps (e.g., Vercel).

How the Process Works - From Start to Finish

1. **Code Development:** Develop or modify C++ source files locally.

2. **Commit and Push:** Commit changes and push to GitHub main branch.
3. **GitHub Actions:** Automatically builds and tests the app on GitHub-hosted runners.
4. **Webhook Event:** GitHub sends a webhook to the public ngrok endpoint.
5. **ngrok Tunnel:** Forwards webhook to Jenkins running locally.
6. **Jenkins Build:** Jenkins pipeline runs build, execution, and archiving stages.
7. **Status Reporting:** Build status and logs available in both Jenkins and GitHub.
8. **Branch Protection:** Ensures only successfully tested code is merged into main.

Jenkins Plugins Installed for CI/CD and Webhook Integration

Plugin Name	**Purpose**
Git plugin	Enables Jenkins to pull source code from GitHub repositories.
Pipeline plugin	Enables scripted and declarative pipelines using Jenkinsfile.
GitHub plugin	Integrates Jenkins with GitHub, providing webhook and status notification support.
Generic Webhook Trigger	Triggers Jenkins builds via arbitrary webhook POSTs, essential for GitHub webhook calls.
Pipeline: Stage View	Provides graphical visualization of pipeline stages in Jenkins UI.
Credentials Binding	Manages sensitive credentials securely for pipeline use (e.g., GitHub tokens).

These plugins empower your Jenkins to respond instantly to GitHub events, run automated pipelines, and provide rich visual insights.

Visual Attachments

Image	Description
ci-cd.png	Complete CI/CD process flow illustration.
build.png	Screenshot of a successful GitHub Actions build.
log.png	Jenkins pipeline console log showing execution.
ngrok.png	ngrok terminal showing public URL and tunnel.
local-run.png	Jenkins UI showing local pipeline run and artifacts.

Summary

You now have a fully functional CI/CD pipeline for a C++ project using modern tools:

- Automated builds and tests on GitHub with CMake and Actions.
- Local Jenkins server builds triggered securely by GitHub webhooks tunneled through ngrok.
- Branch protection rules to maintain code quality and safeguard your main branch.
- Visualization and logging that enable clear insight into your build process.

This setup allows fast development, reliable builds, and clean code deployment suitable for production and learning purposes.