Placement Empowerment Program
Cloud Computing and DevOps Centre

Automate Static Website Deployment Locally:

Create a script that updates your server whenever changes are pushed.

Name: Sylashri Rajendran

Department: IT

Introduction

In this Proof of Concept (PoC), we aim to automate the deployment of a static website to a local Apache server using Git hooks. Specifically, the post-receive Git hook will be configured to trigger a deployment process after changes are pushed to the Git repository. By setting up this automation, we can ensure that the latest code changes are immediately reflected on the live website without needing manual intervention.

Overview

This PoC demonstrates the integration of Git hooks, Apache, and a local Git repository for automating the deployment of static websites. The process consists of:

1. **Setting up a Bare Git Repository**: A repository that acts as a remote for receiving code changes.

2. **Configuring the post-receive Git Hook**: A script that will run after code is pushed, which pulls the latest changes and deploys them to the Apache htdocs directory.

3. **Automating the Deployment**: The script will automate the process of pulling the latest code and restarting the Apache server to reflect the changes immediately.

In this PoC, the local machine will be used to demonstrate the deployment process on a Windows environment with Apache running the static website.

Objectives

The primary objectives of this PoC are:

1. **Automate Static Website Deployment**: Set up an automated system where changes pushed to a Git repository are automatically deployed to the Apache web server.

2. **Enhance Development Efficiency**: Simplify the process of updating a static website without requiring manual file transfers or server restarts.

3. **Implement Version Control for Static Websites**: Use Git as a version control system to track changes to the static website code and deploy those changes automatically.

4. **Test Automation on Local Setup**: Set up and test the automation on a local environment before applying it to production systems.

Importance

Automating static website deployment using Git hooks is important for the following reasons:

1. Efficient Updates: With the post-receive hook in place, any changes pushed to the repository are automatically deployed, reducing the time spent manually copying files to the server and restarting services.

2. Consistency: This automation ensures that the correct version of the website is always deployed, reducing the chances of human errors during deployment.

3. Streamlined Development Workflow: Developers can focus on writing code rather than manually managing deployments, leading to a smoother workflow and more rapid development cycles.

Step-by-Step Overview

Step 1:

Open GitBash and Create a bare repository to receive the code updates:

```
sylas@LAPTOP-DG79B52P MINGW64 ~
$ cd ~

sylas@LAPTOP-DG79B52P MINGW64 ~
$ mkdir my-site.git

sylas@LAPTOP-DG79B52P MINGW64 ~
$ cd my-site.git

sylas@LAPTOP-DG79B52P MINGW64 ~/my-site.git
$ git init --bare
Initialized empty Git repository in C:/Users/sylas/my-site.git/
```

Step 2:

Navigate to your Apache htdocs directory. Based on your setup, it should be:

```
sylas@LAPTOP-DG79B52P MINGW64 ~/my-site.git (BARE:master)
$ cd /c/Users/sylas/Downloads/httpd-2.4.63-250122-win32-vs17/Apache24/htdocs

sylas@LAPTOP-DG79B52P MINGW64 ~/Downloads/httpd-2.4.63-250122-win32-vs17/Apache24/htdocs
$ ls
index.html
```

Step 3:

Next, we'll set up the post-receive hook to automatically deploy your code whenever changes are pushed and **Save and close the file.**

Press Ctrl+O and then press Enter to save. Press Ctrl+X to exit.

```
sylas@LAPTOP-DG79B52P MINGW64 ~/Downloads/httpd-2.4.63-250122-win32-vs17/Apache24/htdocs
$ cd ~/my-site.git

sylas@LAPTOP-DG79B52P MINGW64 ~/my-site.git (BARE:master)
$ nano hooks/post-receive
```

MINGW64:/c/Users/sylas/my-site.git

```
  GNU nano 8.2
#!/bin/bash
TARGET="/c/Users/sylas/Downloads/httpd-2.4.63-250122-win32-vs17/Apache24/htdocs"
GIT_DIR="/C/Users/sylas/my-site.git"
BRANCH="master" # Adjust this if your branch is different

echo "Deploying updates...."
git --work-trees=$TARGET --git-dir=$GIT_DIR checkout -f $BRANCH
echo "Deployment completed!"
```

Step 4:

Make the hook executable:

```
sylas@LAPTOP-DG79B52P MINGW64 ~/my-site.git (BARE:master)
$ chmod +x hooks/post-receive
```

Step 5:

Now, we'll link your development repository to the bare repository and push some changes to test the deployment.

```
sylas@LAPTOP-DG79B52P MINGW64 ~/my-site.git (BARE:master)
$ cd ~

sylas@LAPTOP-DG79B52P MINGW64 ~
$ mkdir my-site

sylas@LAPTOP-DG79B52P MINGW64 ~
$ cd my-site

sylas@LAPTOP-DG79B52P MINGW64 ~/my-site
$ git init
Initialized empty Git repository in C:/Users/sylas/my-site/.git/
```

Step 6:

Create or edit an index.html file. This is just for testing:

```
sylas@LAPTOP-DG79B52P MINGW64 ~/my-site (master)
$ echo '<h1>Hello, Automated Deployment!<h1>' > index.html
```

Step 7:

Add the files, commit the changes:

```
sylas@LAPTOP-DG79B52P MINGW64 ~/my-site (master)
$ git add .
warning: in the working copy of 'index.html', LF will be replaced by CRLF the next time Git touches it

sylas@LAPTOP-DG79B52P MINGW64 ~/my-site (master)
$ git commit -m "Initial commit for deployment test"
[master (root-commit) d313f68] Initial commit for deployment test
 1 file changed, 1 insertion(+)
 create mode 100644 index.html
```

Step 8:

Add the correct remote URL and push the changes

```
sylas@LAPTOP-DG79B52P MINGW64 ~/my-site (master)
$ git remote add live /c/Users/sylas/my-site.git

sylas@LAPTOP-DG79B52P MINGW64 ~/my-site (master)
$ git push live master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 272 bytes | 272.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Deploying updates....
remote: unknown option: --work-trees=C:/Users/sylas/Downloads/httpd-2.4.63-250122-win32-vs17/Apache24/htdocs
remote: usage: git [-v | --version] [-h | --help] [-C <path>] [-c <name>=<value>]
remote:            [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
remote:            [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--no-lazy-fetch]
remote:            [--no-optional-locks] [--no-advice] [--bare] [--git-dir=<path>]
remote:            [--work-tree=<path>] [--namespace=<name>] [--config-env=<name>=<envvar>]
remote:            <command> [<args>]
remote: Deployment completed!
To C:/Users/sylas/my-site.git
 * [new branch]      master -> master
```
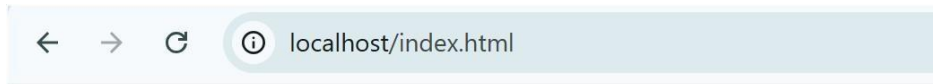
Step 9:

1. **Open your browser** and go to http://localhost.
You should see the message: **"Hello, Automated Deployment!"** from the index.html file you just pushed.

2. **Verify the website update:**
If you see the changes, then the deployment is working as expected!
The post-receive hook has automatically deployed your code after the push.

Hello, Automated Deployment!

Outcomes

By completing this PoC of automating static website deployment using Git hooks, you will:

1. Set Up and Configure a Bare Git Repository:
Learn to create and configure a bare Git repository for automatic deployment.

2. Automate Deployment Using Git Hooks:
Implement the post-receive Git hook to automate deployment whenever updates are pushed to the repository.

3. Script Deployment and Server Restart:
Gain experience scripting the process of pulling code changes and restarting the server to deploy them.

4. Streamline Development Workflow:
Save time by automating the process of updating the live website, reducing manual effort in the deployment process.

5. Troubleshoot Deployment Issues:
Build problem-solving skills by addressing common issues related to code pushes, file permissions, and server configurations.