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: Hariharan v

Department: ADS

St. JOSEPH'S COLLEGE OF ENGINEERING | St. JOSEPH'S INSTITUTE OF TECHNOLOGY

AUTONOMOUS INSTITUTIONS, AFFILIATED TO ANNA UNIVERSITY

Introduction

Efficient deployment processes are crucial for maintaining up-to-date static websites. Automating the deployment of static websites ensures that the latest changes are reflected on the server as soon as they are pushed to the repository. This document outlines how to create a script that updates the server with new changes and restarts the necessary services, using Git hooks and scripting techniques.

Objective

The objectives of this task are:

- **Understand the use of Git hooks for automating deployment.**
- **Learn to write scripts for pulling changes and managing server services.**
- **Automate the deployment process to minimize manual intervention.**

Step 1: Understanding the Requirements

Key Concepts:

- **Git Hooks: Scripts that Git executes automatically on specific events like commits or pushes.**
- **File Deployment: Updating the server files with the latest code changes.**
- **Server Management: Restarting services (e.g., Apache or Nginx) to apply changes.**

Tools/Commands:

- **post-receive Git hook - Triggers after a push to the repository.**
- **git pull - Pulls the latest changes from the repository.**
- **systemctl restart or equivalent - Restarts the server services.**

Step 2: Instructions

1. Configure the Git Hook

1. **Navigate to Your Repository**

cd /path/to/your/repository/.git/hooks

2. **Create or Edit the post-receive Hook Write the following script in the post-receive file:**

```
#!/bin/bash

# Define the deployment directory
DEPLOY_DIR="/var/www/html"

# Navigate to the deployment directory
cd "$DEPLOY_DIR"

# Pull the latest changes
git pull origin main

# Restart the server (adjust for your server type)
sudo systemctl restart apache2

echo "Deployment complete. Server restarted."
```

3. **Make the Hook Executable**

chmod +x post-receive

---

2. Test the Hook

1. **Push changes to the repository:**

git push origin main

2. **Verify that the changes are deployed and the server has restarted.**

---

3. Alternative: Standalone Script

If you prefer not to use Git hooks, create a standalone script to automate the deployment.

1. **Write the Deployment Script**

```
#!/bin/bash

# Define the deployment directory
DEPLOY_DIR="/var/www/html"

# Navigate to the deployment directory
cd "$DEPLOY_DIR"

# Pull the latest changes
git pull origin main

# Restart the server (adjust for your server type)
sudo systemctl restart apache2

echo "Deployment complete. Server restarted."
```

2. **Make the Script Executable**

chmod +x deploy.sh

3. **Run the Script**

./deploy.sh

Step 3: Automate the Script

- **Linux: Use cron to run the script at regular intervals.**

crontab -e

Add the following line to schedule the script every hour:

0 * * * * /path/to/deploy.sh

- **Windows: Use Task Scheduler to execute the script periodically.**

Step 4: Best Practices

1. **Backup Files: Ensure you have backups before deploying new changes.**
2. **Access Control: Restrict script execution to authorized users.**
3. **Error Handling: Add logging and error handling to the script to capture failures.**

Conclusion

Automating the deployment of static websites improves efficiency and reduces the risk of human error. By leveraging Git hooks or custom scripts, you can ensure that your server always serves the latest changes. Follow this guide to set up a reliable and automated deployment process for your static website.