 

**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

Automate static website deployment locally: create a script that updates your server whenever change are pushed.

Name: S.KRITHIKA Department: ADS



**Introduction & Overview**

Static websites are simple, fast, and require minimal infrastructure compared to dynamic sites. However, manually updating them each time a change is made can be tedious and error-prone. Automating the deployment process ensures that updates are pushed seamlessly to a local server whenever changes are committed.

This guide outlines how to automate the deployment of a static website locally using a script that detects and updates the server whenever changes are pushed to the repository.

**Objective**

The goal of this guide is to enable developers to:

* Automate the deployment of static websites in a local environment.
* Reduce manual effort and errors in updating the local server.
* Ensure that the latest version of the website is always available on the local server.
* Improve development efficiency by automating repetitive tasks.

**Importance of Local Hosting & Automation**

1. **Faster Development & Testing:** Local hosting allows developers to test changes instantly before deploying to a live server.
2. **Reduced Downtime:** Automating updates ensures the latest version is deployed immediately, reducing the risk of outdated content.
3. **Consistency:** Automated deployment ensures that every team member works with the latest version without manual intervention.
4. **Error Reduction:** Manual deployments can lead to missed updates or misconfigurations. Automation minimizes these risks.

**Step-by-Step Implementation**

**Prerequisites**

* A static website (HTML, CSS, JavaScript files) hosted in a Git repository.
* A local server (e.g., Apache, Nginx, or a simple Python HTTP server).
* Basic knowledge of shell scripting and Git.

**Step 1: Set Up a Local Server**

1. Install a lightweight server (for example, Python’s HTTP server):

python3 -m http.server 8000

1. Ensure the server is running and accessible via <http://localhost:8000/>.

**Step 2: Clone the Repository Locally**

cd /path/to/local/server/directory

git clone <https://github.com/your-username/your-repository.git>

**Step 3: Create an Automation Script**

Create a shell script (deploy.sh) that pulls the latest changes and restarts the server.

#!/bin/bash

# Navigate to the website directory

cd /path/to/local/server/directory/your-repository || exit

# Pull latest changes

git pull origin main

# Restart the local server

pkill -f "python3 -m http.server"

nohup python3 -m http.server 8000 &

echo "Deployment successful! Website updated."

**Step 4: Automate Script Execution with Git Hooks**

Modify the post-merge Git hook to execute the script whenever changes are pulled.

1. Navigate to the Git hooks directory:

cd /path/to/local/server/directory/your-repository/.git/hooks

1. Create or edit the post-merge file:

nano post-merge

1. Add the following content:
2. #!/bin/bash

/path/to/local/server/directory/deploy.sh

1. Make the hook executable:

chmod +x post-merge

**Step 5: Test the Automation**

1. Push a change to the repository.
2. Pull the changes on the local machine:

git pull origin main

1. Verify that the website updates automatically and the server restarts.

**Expected Outcome**

* The website is updated automatically whenever changes are pushed to the repository.
* The local server reloads the latest version without manual intervention.
* Developers experience a more efficient workflow with reduced deployment errors.
* Time and effort spent on updating the local environment are minimized.