St. JOSEPH'S GROUP OF INSTITUTIONS
OMR, CHENNAI - 119

aws

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: Madhumitha H

Department: ADS

Introduction:

This task involves automating the deployment of a static website on a local server. By using Node.js and the Express framework, the website is served on a local machine. A file-watching mechanism is implemented using **Chokidar** to monitor changes in the website's public/ folder. Whenever a change is detected, the server is automatically restarted using **PM2**, ensuring the latest updates are always reflected. This setup eliminates the need for manual server restarts, streamlining the development process.

Overview:

1. **Set Up the Local Server: Use Node.js and Express to create a server that serves a static website from the public/ folder.**
2. **Install Dependencies: Install PM2 for process management and Chokidar to monitor file changes in the public/ folder.**
3. **Create File Watcher: Implement a script (watcher.js) that uses Chokidar to detect changes in the public/ folder and automatically restarts the server when changes are made.**
4. **Start the Server with PM2: Use PM2 to start and manage the server, ensuring it runs continuously and restarts if necessary.**
5. **Automate Server Restart: Ensure that any changes in the public/ folder trigger a server restart, reflecting the updated content without manual intervention.**

Objectives:

The objective of this task is to automate the deployment of a static website on a local server using Node.js. It aims to streamline development by detecting file changes in real time and restarting the server automatically. The task involves using **PM2** for process management and **Chokidar** for file watching. The goal is to eliminate manual restarts and ensure the latest website updates are always served.

Step-by-Step Overview

Step 1: **Set Up the Project Directory**

You created a project directory (e.g., my-static-website) to store your static website files.

```
Windows PowerShell

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\madhu> mkdir my-static-website


    Directory: C:\Users\madhu


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        02-02-2025     19:46                my-static-website


PS C:\Users\madhu> cd my-static-website
PS C:\Users\madhu\my-static-website> npm init -y
Wrote to C:\Users\madhu\my-static-website\package.json:

{
  "name": "my-static-website",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

Step 2

Create a new dictionary inside my-static-website dictionary called public

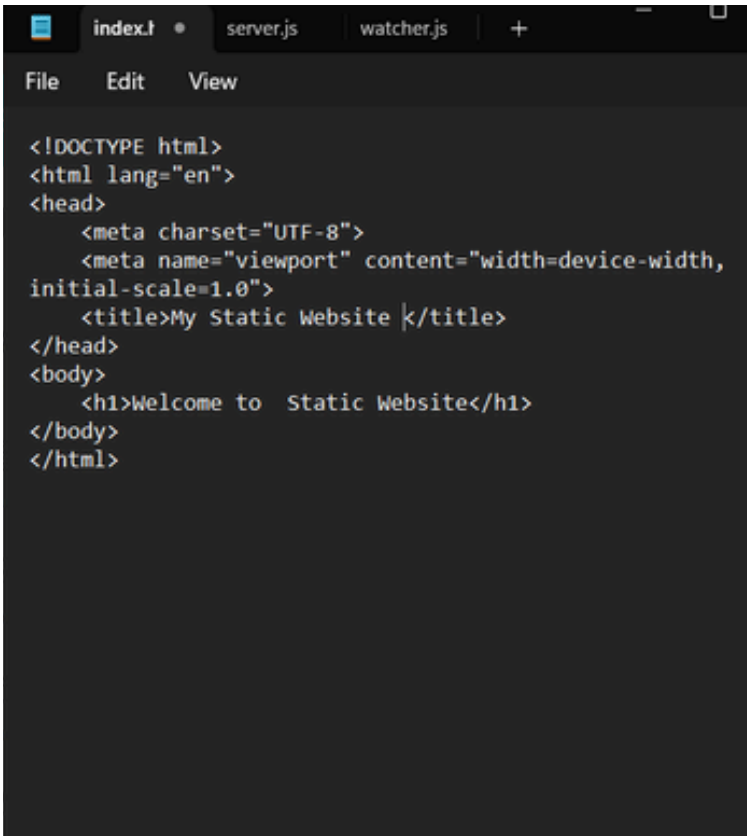
Step 3

Create a new file inside public called index.html and put some html code.

you can do this by directing into the public dictionary inside my-static-website folder and create a file called index.hmtl by the command : notepad index.html

this will directly open on to notepad asking whether you want to create a new file named index

press enter and type in the content.




```
PS C:\Users\madhu\my-static-website> cd public
PS C:\Users\madhu\my-static-website\public> notepad index.html
```

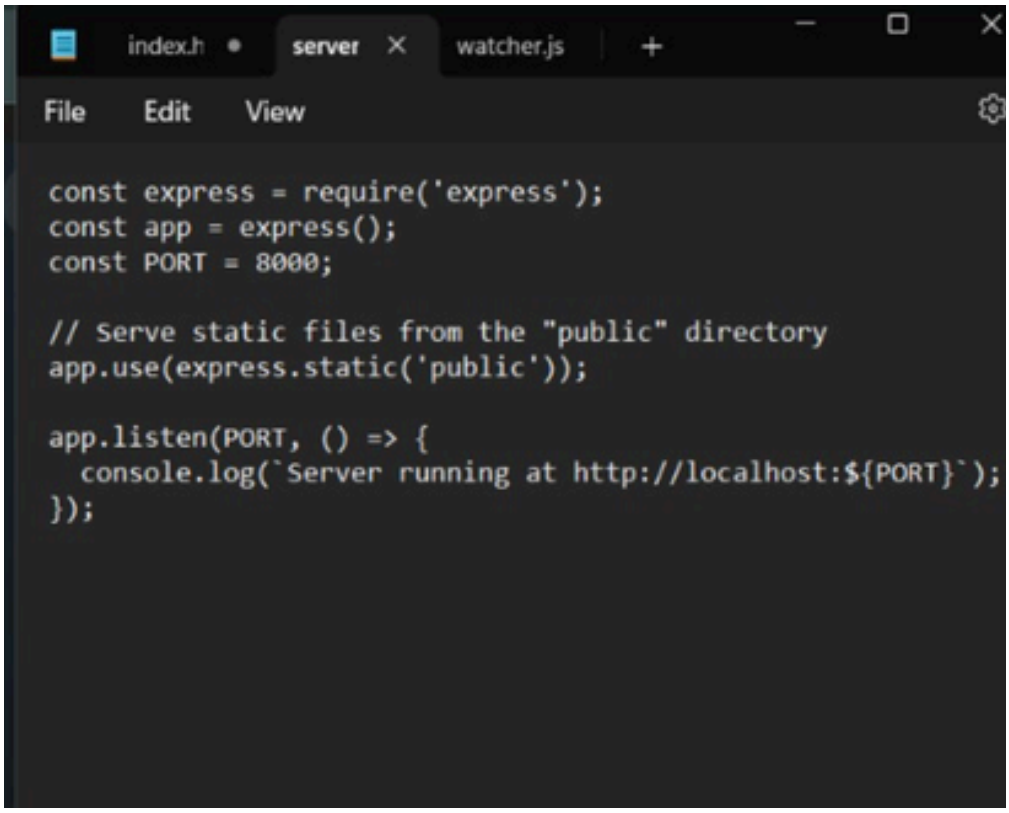
Step 4:

now move back to the my-static-website dictionary and create a new file called server.js

```
PS C:\Users\madhu\my-static-website\public> cd ..
PS C:\Users\madhu\my-static-website> npm install express

added 69 packages, and audited 70 packages in 6s

14 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS C:\Users\madhu\my-static-website> notepad server.js
```

Step 5

fill in the server.js file with the following content



Step 6:

Install Dependencies (PM2 and Chokidar)

- You installed the necessary dependencies:
  - **PM2** for process management (ensuring the server runs continuously).
  - **Chokidar** for file watching (to detect file changes and restart the server automatically).

```bash
npm install chokidar pm2
```

Step 7:

Create the File Watcher (watcher.js)

- You created a **watcher.js** script that uses **Chokidar** to monitor changes in the public/ folder and restarts the server whenever files change.

index.h • server.js watch ×

File Edit View

```
const chokidar = require('chokidar');
const pm2 = require('pm2');  // To restart the PM2 server

// Watch the public directory for any changes
const watcher = chokidar.watch('./public', {
  persistent: true,
  ignoreInitial: true,
  ignored: /(^|[\/\\])\../  // Ignore dotfiles like .git
});

// Restart server whenever a file is changed
watcher.on('change', (path) => {
  console.log(`${path} has been changed, restarting the
server...`);
  pm2.restart('my-server', (err, proc) => {
    if (err) {
      console.error(err);
    }
    console.log('Server restarted!');
  });
});

console.log('Watching for file changes in /public...');
```

Ln 23 Col 1 636 characters 100% Windows (CRLF) UTF-8

Step 8:

Start the Server Using PM2

- You started your **Node.js server** using PM2 so it can run in the background and automatically restart if it crashes:

put the following command

```
pm2 start server.js --name my-server
```

```
PS C:\Users\madhu\my-static-website> pm2 start server.js --name my-server
[PM2] Starting C:\Users\madhu\my-static-website\server.js in fork_mode (1 instance)
[PM2] Done.
```

id	name	mode	↺	status	cpu	memory
0	my-server	fork	0	online	0%	41.0mb

Step 9:

Run the File Watcher (watcher.js)

- You ran the **file watcher script** (watcher.js), which started monitoring changes in the public/ directory:

using command : node watcher.js

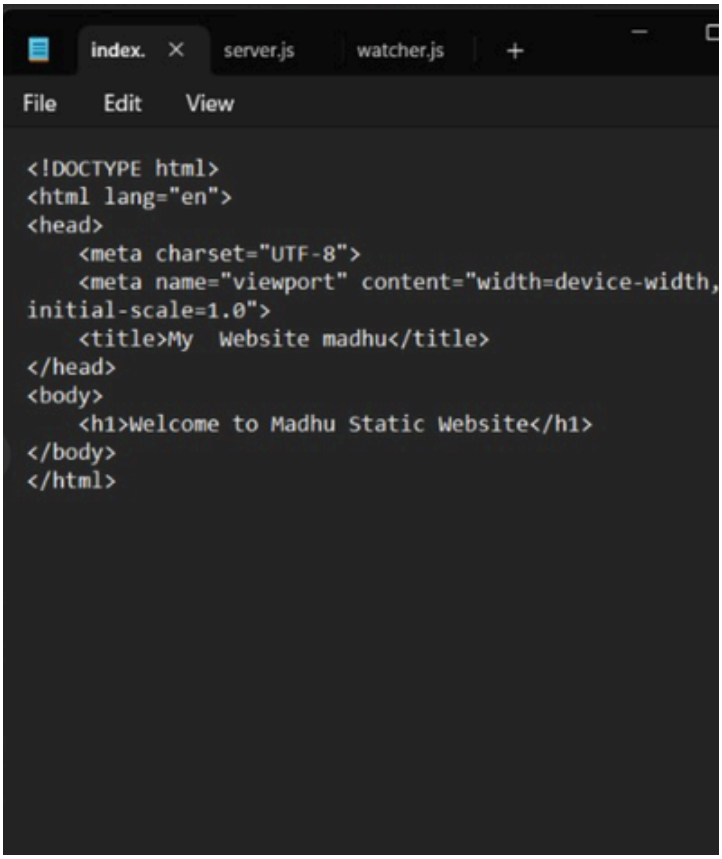
```
PS C:\Users\madhu\my-static-website> node watcher.js
Watching for file changes in /public...
```

you must see some output like this. since no changes are made in any file inside th public dictionary there wont be anything to specify and this will be the output

Step 10:

do some changes to your index.html file in the public dictionary and then save it.

then run the command: node watcher.js to see it it detects the change made in the file



i had added my name n cut the word static here from the script .

now run the node watcher.js command to see if it detects in the terminal

```
PS C:\Users\madhu\my-static-website> node watcher.js
Watching for file changes in /public...
public\index.html has been changed, restarting the server...
Server restarted!
```

if it wokrs properly this will be its final output .

Outcomes

1. Automated Static Website Deployment:

- **Static website deployment** is now automated: the server automatically updates itself whenever you make changes to files in the public/ folder, saving you time and manual effort.

2. Continuous Server Management with PM2:

- The **server** is continuously running in the background using **PM2**. If the server crashes or needs to restart for any reason, PM2 ensures that it restarts without any manual intervention.

3. Real-Time File Change Detection:

- **File changes in the public/ folder** are automatically detected using **Chokidar**. When a file (HTML, CSS, JS, etc.) is updated, the file watcher script detects it and triggers a server restart.

4. Improved Development Workflow:

- **Efficient workflow**: As a developer, you no longer need to manually restart the server after every change. This allows for faster iteration and more convenient testing of updates on the site.

5. Automatic Server Restart:

- When files are modified in the public/ folder, the server is **automatically restarted** to reflect the changes on the website. This ensures that the latest content is always served without needing to stop and start the server manually.

6. Running Processes in the Background:

- By using **PM2** to manage both the server and the watcher script, both processes are running in the background, ensuring the website is always up and the file changes are continuously monitored.

7. Task Completion Without Git:

- The task was completed **without using Git** or a version control system. Instead, you relied entirely on a **file-watching mechanism** and **PM2** for managing updates and server restarts.

8. Reliable Setup:

- The server setup is now **reliable** and does not require you to manually intervene when content changes are made. This is especially useful for **local development** where frequent changes are common.