

**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: Indhumathi S Department: CSE**

A black and white logo

Description automatically generated

**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.

**Objective:**

**Set Up the Local Server**: Use Node.js and Express to create a server that serves a static website from the public/ folder.

**Install Dependencies**: Install PM2 for process management and Chokidar to monitor file changes in the public/ folder.

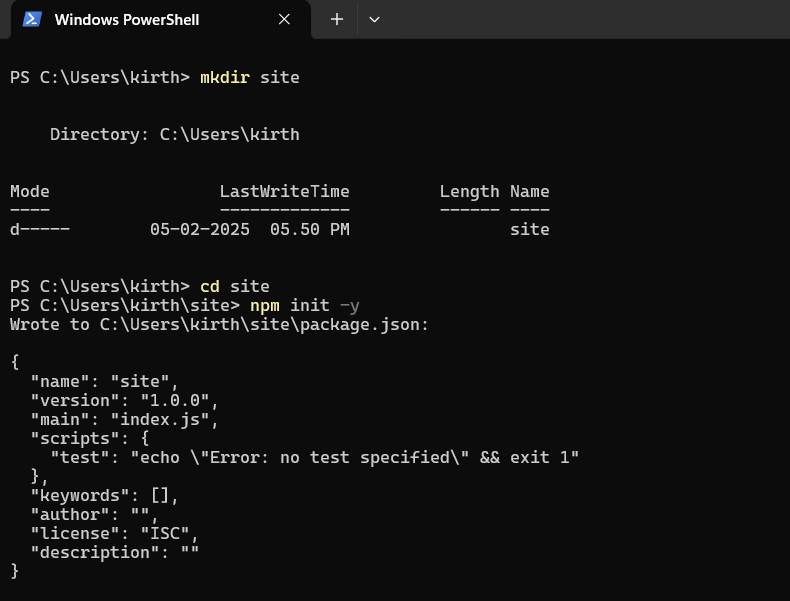
**Create File Watcher**: Implement a script (watcher.js) that uses Chokidar to detect changes in the public/ folder and automatically restarts the server when changes are made.

**Start the Server with PM2**: Use PM2 to start and manage the server, ensuring it runs continuously and restarts if necessary.

**Automate Server Restart**: Ensure that any changes in the public/ folder trigger a server restart, reflecting the updated content without manual intervention

**Step 1:**

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



**Step 2**

Create a new dictionary inside **site** 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.html 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.

****

**Step 4:**

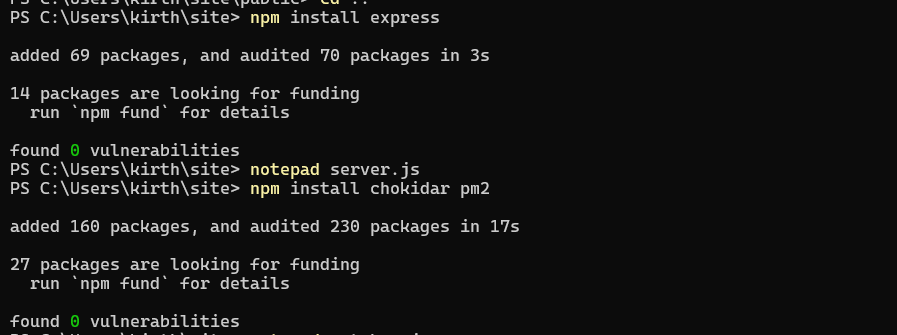
now move back to the site dictionary and create a new file called **server.js**

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).

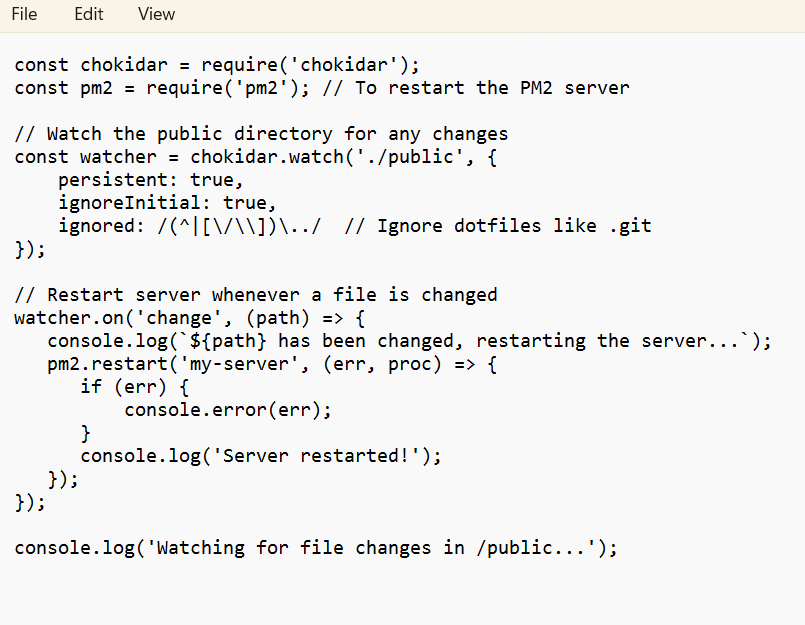
****

**Step 5:**

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.

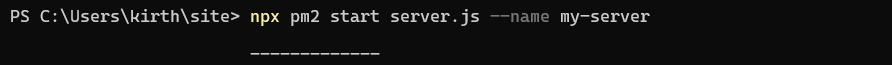
****



**Step 6:**

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  ****

****

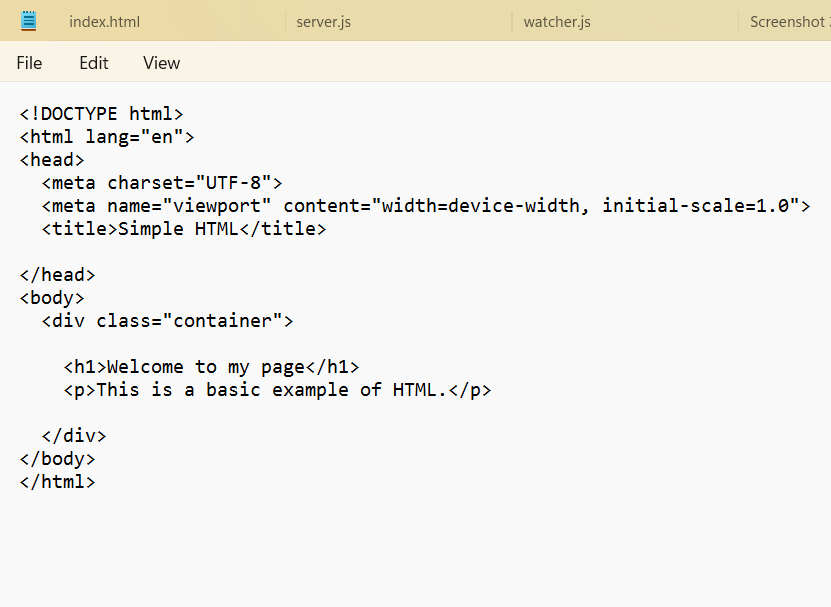
**Step 7:**

using command: **node watcher.js**

****

**Step 8:**

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

****

**Step 9:**

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

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

if it works 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.