**Placement Empowerment Program**

***Cloud Computing and DevOps Centre***

**Create a Simple Backup Script :** Create a script

that backs up your entire Git repository to a local

folder daily.

Name: Bhavadharani S Department: CSE



**Introduction**

Data backup is a critical practice in software development and system administration, ensuring the safety and recoverability of important files. Automating the backup process helps reduce the risk of data loss while saving time and effort. In this Proof of Concept (POC), we will create a simple backup script to secure the contents of a Git repository.

The script will allow you to compress your repository and save it to a designated backup folder. Additionally, we will automate the backup process using scheduling tools like cron on Linux or Task Scheduler on Windows to run the script daily. Automating backups is an essential step for maintaining reliable workflows and safeguarding against unexpected data loss or corruption.

**Overview**

In this POC, we will cover the following steps:

1. **Installing Tools**: Verify and set up required tools (tar for Linux, robocopy for Windows).
2. **Writing the Backup Script**: Create a script that compresses and saves the Git repository to a backup folder.
3. **Testing the Script**: Ensure the script works correctly and successfully creates backups.
4. **Automating the Process**: Use **cron** or **Task Scheduler** to automate the script’s execution daily.
5. **Verifying Backups**: Validate that backups are properly created and stored.

**Objectives**

By the end of this POC, you will:

1. **Learn the Basics of Backup Processes**: Understand the importance of automating backups for version-controlled projects.
2. **Write a Backup Script**: Gain hands-on experience in creating and running a backup script using tar or robocopy.
3. **Automate Script Execution**: Configure task scheduling tools to run the script daily.
4. **Validate Backup Success**: Learn how to verify the completeness and integrity of backups.
5. **Safeguard Your Work**: Establish a reliable backup mechanism for your Git repository.

**Importance of Automating Backups**

1. **Data Safety**: Regular backups ensure that your project data is always recoverable in case of system failure or accidental deletions.
2. **Efficient Workflow**: Automation eliminates the need for manual intervention, saving time and reducing the risk of forgetting to back up critical files.
3. **Version Control**: Backing up a Git repository preserves both the project files and the version history, enabling complete restoration when needed.
4. **Collaboration Readiness**: Maintains a reliable backup mechanism, even when working on a shared project with multiple contributors.
5. **Recovery from Errors**: Enables quick rollback to a stable state without losing progress.

**Step-by-Step Overview**

Step 1:

**Create the Necessary Folder**

Ensure the directory C:\Projects\BackupPOC exists:

1. Open **File Explorer**.
2. Navigate to C:\Projects\.
3. Create a folder named BackupPOC.

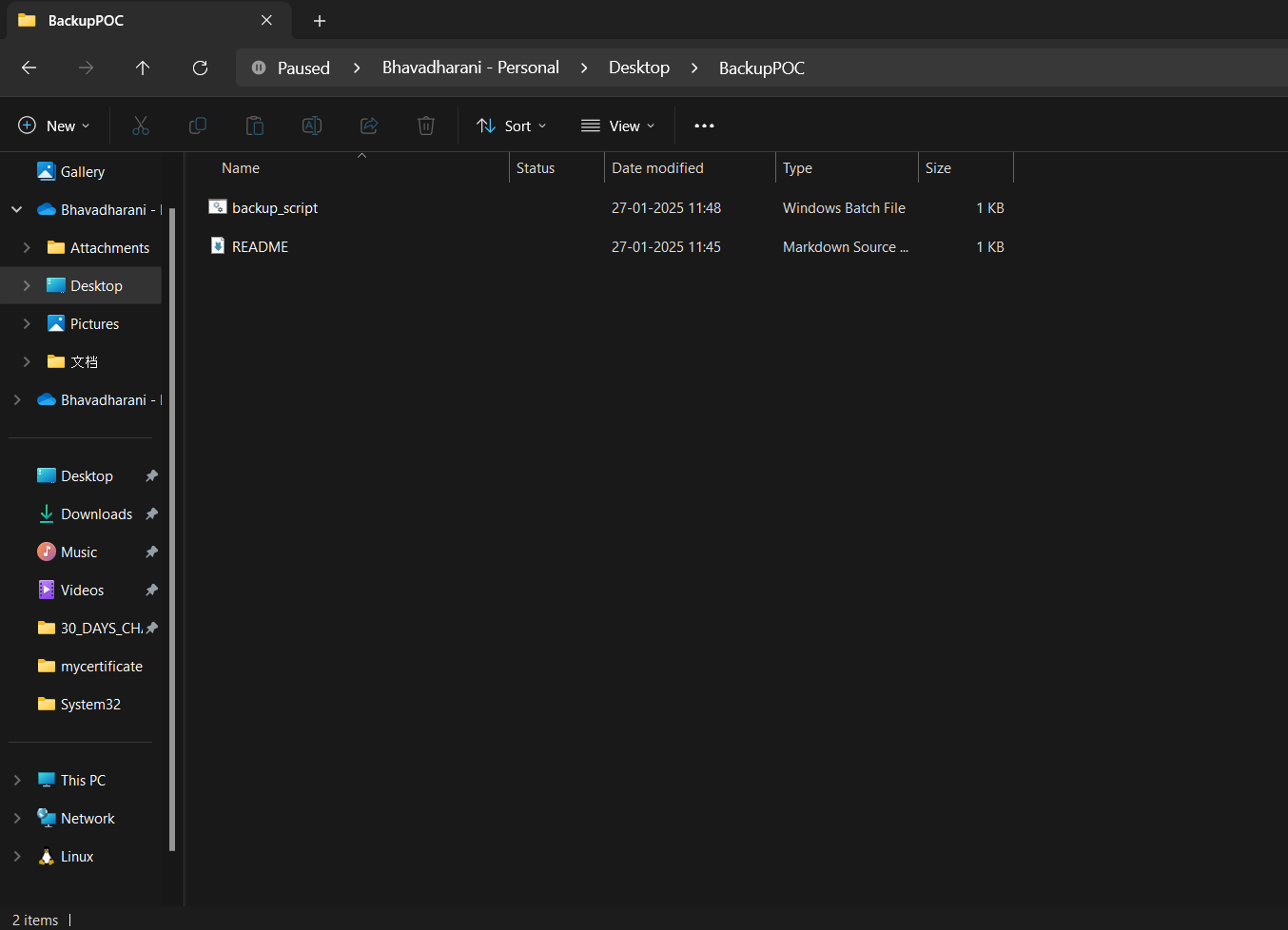


Step 2:

**Move the Script to the Correct Location**

Save the backup\_script.bat file in the C:\Projects\BackupPOC

Folder



Step 3:

**Step 3: Run the Script in PowerShell**

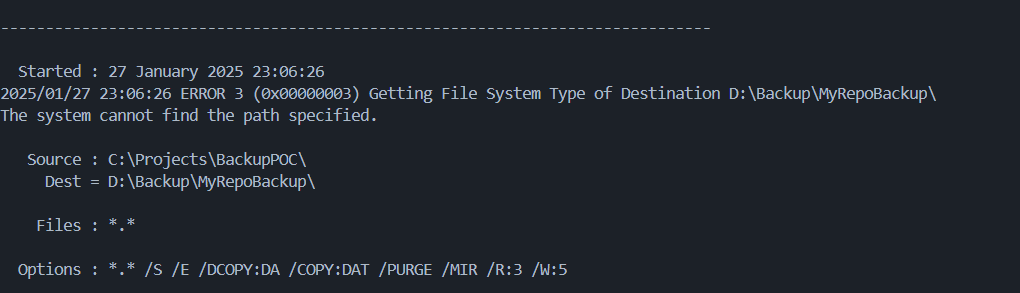
1. Open PowerShell.
2. Change the directory to C:\Projects\BackupPOC:

***cd C:\Projects\BackupPOC***

1. Run the script by explicitly specifying the current path:

***.\backup\_script.bat***

* + The .\ indicates that the script is in the current directory.



Step 4 :

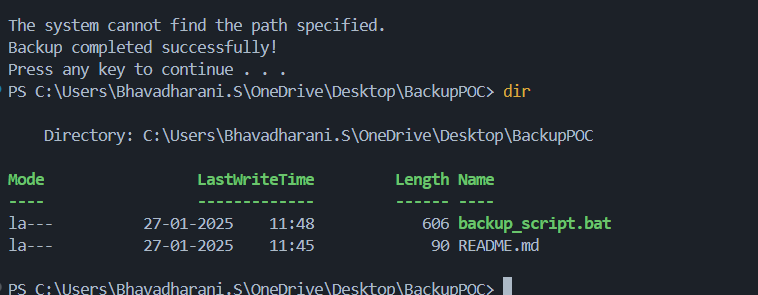
Debug Any Errors

If you still face issues:

* Verify the file name is exactly backup\_script.bat.
* Use dir in PowerShell to list the files in the current directory:

***dir***

Ensure backup\_script.bat is listed.



Step 5 :

**Test the Backup Process**

After the script runs successfully, verify that the destination folder (e.g., D:\Backup\MyRepoBackup) contains the backed-up files.

Let me know if you encounter further issues!

Step 6:

**Verify the Backup**

1. Open the destination folder you specified in the script (e.g., D:\Backup\MyRepoBackup).
2. Check if:
   * The repository files and folders are copied correctly.
   * The backup file (if compressed using tar or robocopy) is present.

If any files are missing or the backup isn't working as expected, troubleshoot the script to ensure the paths and commands are correct.

Step 7:

**Automate the Script**

To ensure your backup runs daily, set up automation using **Task Scheduler** in Windows.

**Steps to Automate Using Task Scheduler:**

1. **Open Task Scheduler:**
   * Press Win + S, search for **Task Scheduler**, and open it.
2. **Create a New Task:**
   * In Task Scheduler, click **Create Task** on the right-hand side.
3. **General Settings:**
   * Give your task a name, e.g., "Daily Git Repository Backup."
   * Under **Security Options**, select "Run whether the user is logged on or not."
4. **Trigger:**
   * Go to the **Triggers** tab and click **New**.
   * Set the trigger to **Daily** and specify a start time.
5. **Action:**
   * Go to the **Actions** tab and click **New**.
   * For **Action**, select **Start a Program**.
   * In the **Program/script** field, type the full path to cmd.exe.
   * In the **Add arguments** field, type:

/c "C:\Projects\BackupPOC\backup\_script.bat"

1. **Save the Task:**
   * Click **OK** and provide your credentials if prompted.
2. **Test the Task:**
   * Right-click the task and select **Run** to verify it works.

Step 8**:**

**Monitor Logs**

If you've set up logging in your script, check the logs daily to confirm the backup ran successfully.

* If no logging is included in the script, modify it to append the status of each run to a log file:

***robocopy C:\Path\To\Repo D:\Backup\MyRepoBackup /E /LOG+:D:\Backup\backup.log***

This adds output details to backup.log.

**Step 9: Confirm Automation**

After setting up automation:

1. Wait until the next scheduled time.
2. Check the destination folder and logs to confirm the script executed as planned.