 

**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: S.KRITHIKA Department: ADS



**Introduction & Overview**

Version control systems like Git play a crucial role in software development by tracking changes and facilitating collaboration. However, relying solely on remote repositories (e.g., GitHub, GitLab, or Bitbucket) can pose risks such as data loss due to accidental deletion, service outages, or security breaches. To mitigate these risks, creating a local backup of your Git repository ensures that you always have a copy of your work available.

This document outlines a simple automated script that will back up your entire Git repository to a designated local folder daily.

**Objective**

The primary goal of this script is to:

* Automate the backup process for Git repositories.
* Store backups in a local folder to prevent data loss.
* Ensure developers can restore their repositories in case of corruption or accidental deletion.
* Maintain version control integrity while providing redundancy.

**Importance of Local Hosting**

While cloud-based repositories provide great accessibility and security, local backups add an extra layer of protection:

1. **Redundancy:** Having multiple backups reduces the risk of losing critical data.
2. **Offline Access:** Local backups allow developers to access their repositories without an internet connection.
3. **Version Recovery:** If an accidental deletion occurs on the remote repository, a local copy ensures a quick recovery.
4. **Security:** Local backups protect sensitive code from unauthorized access in case of cloud security breaches.

**Step-by-Step Overview**

The backup script follows these steps:

1. **Specify Repository Path**
   * Define the location of the Git repository you want to back up.
2. **Specify Backup Directory**
   * Set a local folder where backups will be stored.
3. **Create a Timestamped Backup Folder**
   * Use a timestamp to differentiate each backup.
4. **Use Git to Clone the Repository**
   * Utilize git clone --mirror to create a full backup of the repository.
5. **Automate the Process with a Cron Job (Linux) or Task Scheduler (Windows)**
   * Schedule the script to run daily.

**Backup Script (Example in Bash for Linux/macOS)**

#!/bin/bash

# Set variables

REPO\_PATH="/path/to/your/repository"

BACKUP\_DIR="/path/to/backup/location"

TIMESTAMP=$(date +"%Y-%m-%d\_%H-%M-%S")

BACKUP\_PATH="$BACKUP\_DIR/git\_backup\_$TIMESTAMP"

# Ensure backup directory exists

mkdir -p "$BACKUP\_PATH"

# Clone repository with mirror option

git clone --mirror "$REPO\_PATH" "$BACKUP\_PATH"

# Log the backup process

echo "Backup completed at: $(date)" >> "$BACKUP\_DIR/backup\_log.txt"

**Automating the Backup**

**For Linux/macOS (Using Cron Job)**

1. Open the terminal and type crontab -e to edit the cron jobs.
2. Add the following line to schedule the script to run daily at midnight:

0 0 \* \* \* /path/to/backup\_script.sh

**For Windows (Using Task Scheduler)**

1. Open Task Scheduler and create a new task.
2. Set the trigger to run the script daily at a specified time.
3. Set the action to run bash.exe (if using Git Bash) with the script path as an argument.

**Expected Outcome**

After implementing this script:

* A local backup of the Git repository will be created daily.
* Backup logs will be available for tracking past backups.
* Developers will have an offline, versioned copy of their repository.
* Recovery from accidental data loss will be quick and efficient.