Placement Empowerment Program

Cloud Computing and DevOps Centre

Create a simple Backup script :

that backs up your entire git repository to a local website

Name: Keshika D Department: ADS

Introduction:

The task was to create a backup system for a Git repository. We set up a script that automatically pushes any new changes from your local repository to a backup repository on GitHub. When no changes were made, the script showed "up to date." But whenever you make changes, the script will commit and push those updates to GitHub, keeping your backup repository current. This ensures that your project is regularly backed up without manual intervention.

Overview

Here's what we will cover in this setup:

1. **Clone your local Git repository to your desktop.**
2. **Create a new backup repository on GitHub.**
3. **Write a batch script to commit and push changes to the backup repository.**
4. **Run the script to push changes (if any) to the backup repository.** 5. **Verify that the backup repository on GitHub is updated with any new changes.**

Objectives:

1. **Automate Backup**: Create an automated process to back up local Git repository changes to a remote GitHub repository.
2. **Ensure Version Control**: Keep a separate backup of your project on GitHub to prevent data loss.
3. **Simplify Regular Backups**: Set up a system that pushes new changes to the backup repo without manual intervention.
4. **Maintain Sync**: Ensure that the backup repository is always up-to-date with the latest changes from the local repository.
5. **Increase Data Security**: Safeguard your code by regularly backing it up to a remote, cloud-based platform like GitHub.

Step-by-Step Overview

Step 1:

1. Set Up Your Local Git Repository

Clone your existing Git repository (if not already cloned) from GitHub to your desktop using the following command

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

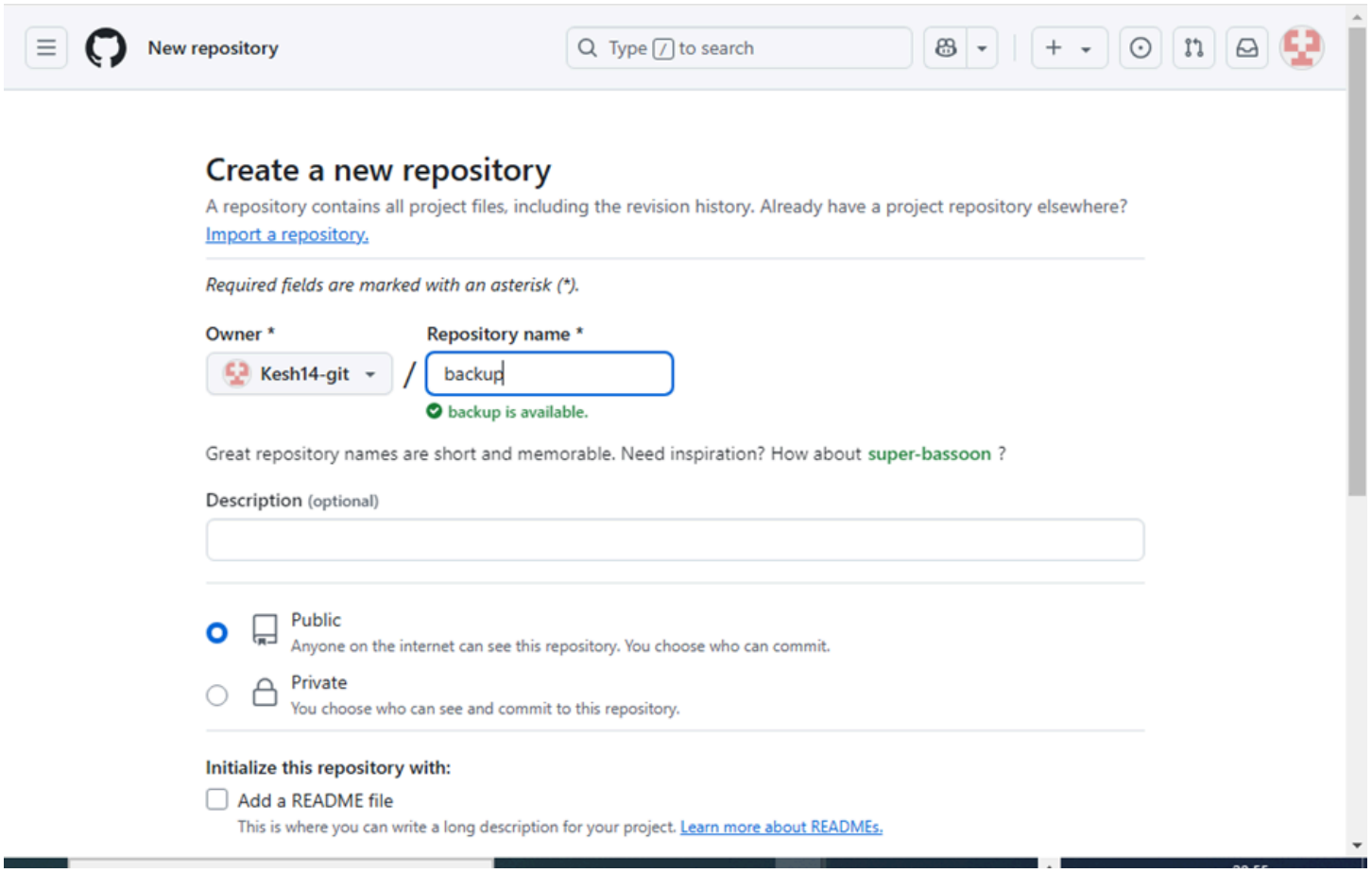
```
C:\Users\TEMP.DESKTOP-S0M6S0K.005>git clonehttps://github.com/Kesh14-git/Pep-Tasks
git: 'clonehttps://github.com/Kesh14-git/Pep-Tasks' is not a git command. See 'git --help'.

C:\Users\TEMP.DESKTOP-S0M6S0K.005>git clone https://github.com/Kesh14-git/Pep-Tasks
Cloning into 'Pep-Tasks'...
remote: Enumerating objects: 15, done.
remote: Counting objects: 100% (15/15), done.
remote: Compressing objects: 100% (13/13), done.
remote: Total 15 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (15/15), 7.20 MiB | 9.11 MiB/s, done.
Resolving deltas: 100% (3/3), done.
```

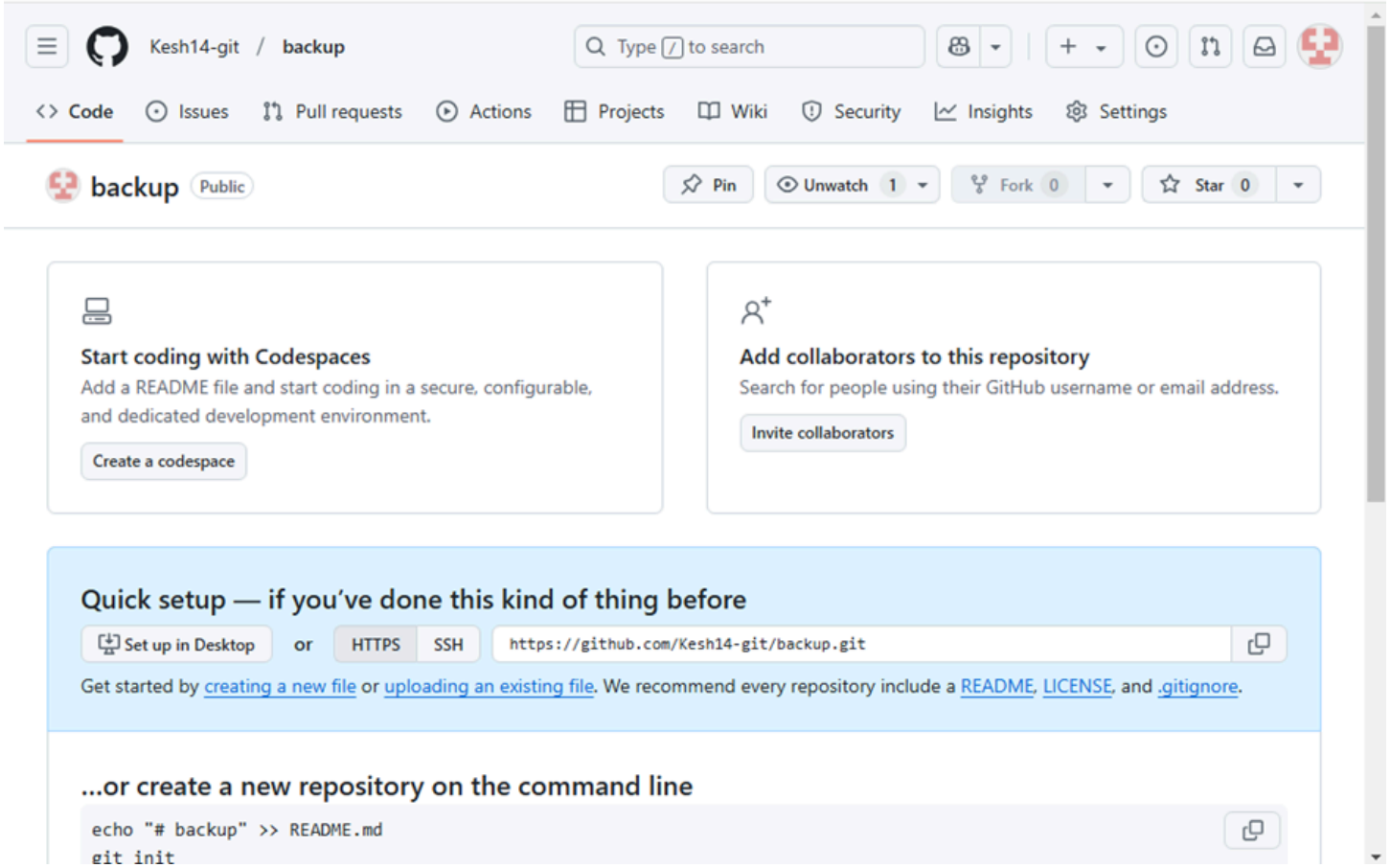
Step 2

Create a Backup Repository on GitHub

Go to GitHub and **create a new private or public repository** (e.g: backup-).
Do not initialize it with any files (like README

once you have created you will see a page like this


New repository
Type / to search
Create a new repository
A repository contains all project files, including the revision history. Already have a project repository elsewhere?
Import a repository.
Required fields are marked with an asterisk (*).
Owner *
Repository name *
Kesh14-git
/
backup
backup is available.
Great repository names are short and memorable. Need inspiration? How about super-bassoon ?
Description (optional)
Public
Anyone on the internet can see this repository. You choose who can commit.
Private
You choose who can see and commit to this repository.
Initialize this repository with:
Add a README file
This is where you can write a long description for your project. Learn more about READMEs.


Kesh14-git / backup
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
backup
Public
Pin
Unwatch 1
Fork 0
Star 0
Start coding with Codespaces
Add a README file and start coding in a secure, configurable, and dedicated development environment.
Create a codespace
Add collaborators to this repository
Search for people using their GitHub username or email address.
Invite collaborators
Quick setup — if you've done this kind of thing before
Set up in Desktop
or
HTTPS
SSH
https://github.com/Kesh14-git/backup.git
Get started by creating a new file or uploading an existing file. We recommend every repository include a README, LICENSE, and .gitignore.
...or create a new repository on the command line
echo "# backup" >> README.md
git init

Step 3

backup_git_repo - Notepad

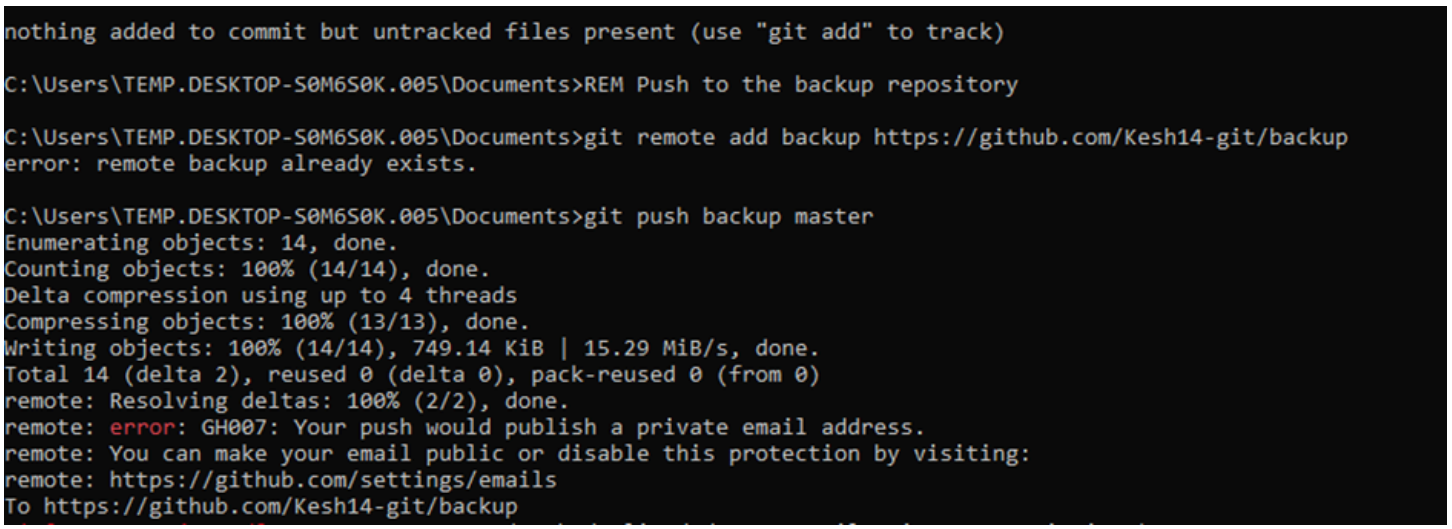
File Edit Format View Help

```
@eco off
set REPO_PATH=E:\Keshika\PEP TASKS
set BACKUP_REPO_URL=https://github.com/Kesh14-git/backup
set TIMESTAMP=%DATE:~10,4%-DATE:~4,2%-%DATE:~7,2%_%TIME:~0,2%-%TIME:~3,2%-%TIME
set BACKUP_NAME=git_backup_%TIMESTAMP%

REM Navigate to your repository folder
cd "%REPO_PATH%"

REM Add a new backup commit
git add .
git commit -m "Backup on %TIMESTAMP%"

REM Push to the backup repository
git remote add backup %BACKUP_REPO_URL%
git push backup master

echo Backup pushed to Github repository at: %BACKUP_REPO_URL%
pause
```

Write the Backup Script

Open **Notepad** and create a new script (backup_git_repo.bat) with the following code:

Step 4: Save and Run the Script



Step 5

you will get an output saying everything is backup to your backup- repository in

the end

Since no changes were made to the local repository after the backup script was first created, the script output showed **"up to date, no new changes to be made"**.

This simply means that the local repository is already **synchronized** with the backup repository on GitHub.

However, whenever **new changes** are made to the local repository (such as modifications, additions, or deletions of files), the backup script will automatically commit those changes and push them to the backup repository.

So, when new changes occur, they will be reflected in the **backup repository** on GitHub.

Task Outcomes:

The outcome of this task are:

Automated Backup Process: Successfully set up a system that automatically backs up changes from a local Git repository to a remote GitHub backup repository.
Up-to-Date Backup: Ensures that the backup repository on GitHub remains synchronized with the local repository, reflecting all changes made in t project.
Improved Data Security: Provides an additional layer of data protection by keeping a secure backup of the project on GitHub, reducing the risk of data loss.
Time-Saving: Eliminates the need for manual backups by automating the process, making it easier to maintain an updated backup.