VCS Task

TASKS

Work Flow:

- Create an EC2 instance with the help of AWS Management Console with linux OS of required configuration.
- Now, Connect an EC2 instance with an help of Windows Terminal or Gitbash or Vbox.
- To connect an EC2 instance the command is:
    - ssh -i "**key_file**" ec2-user@"**Public_IP_address**"

  Key_file --- Key file of the instance with the extension .pem

  Public_IP_address --- Public IP address of the instance.

1. **Establish a new directory, populate it with script files, initiate an empty repository on GitHub, convert the local directory into a Git repository, and link it to GitHub for pushing the code into the repository.Perform merge, rebase, stash commands in following github repo.**

Step 1: Create a New Directory

✓ Open your terminal and run:
- mkdir my-git-project
- cd my-git-project

Explanation:

✓ mkdir my-git-project creates a new folder named my-git-project.

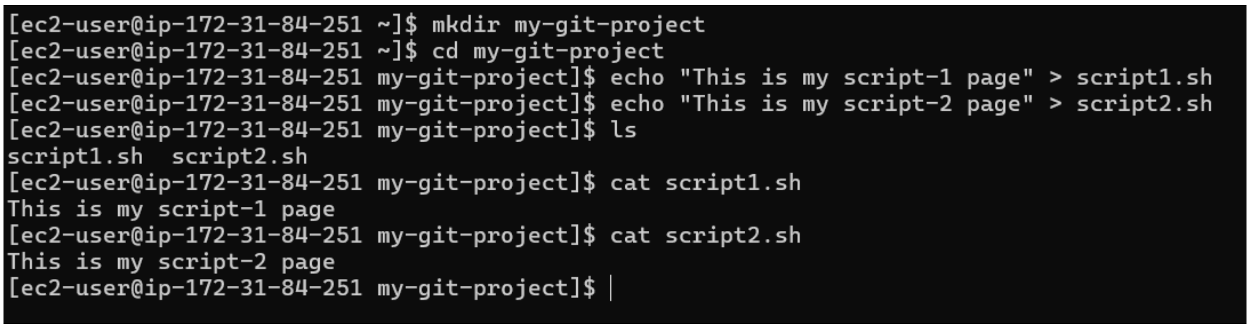
✓ cd my-git-project moves you into that directory.

Step 2: Add Script Files

✓ Inside the MyProject directory, create some sample script files:
- echo "This is my script-1 page" > script1.sh
- echo "This is my script-2 page" > script2.sh

Explanation:

✓ echo ... > filename writes text into a new file.



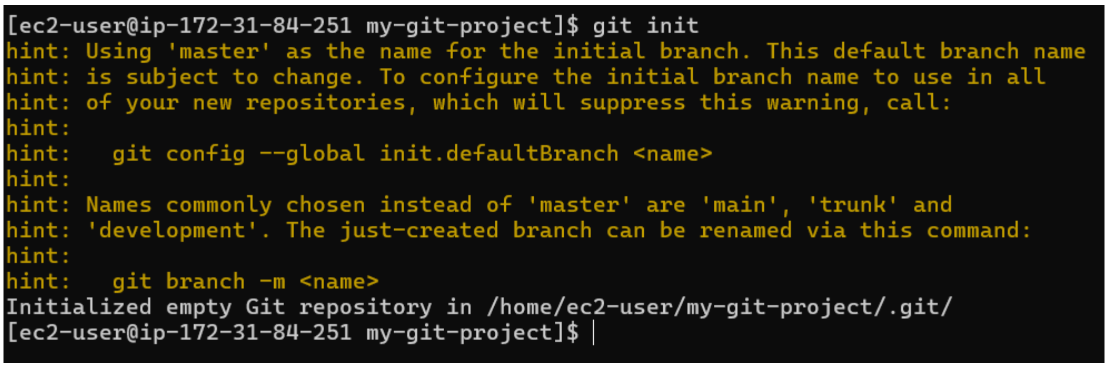
```
[ec2-user@ip-172-31-84-251 ~]$ mkdir my-git-project
[ec2-user@ip-172-31-84-251 ~]$ cd my-git-project
[ec2-user@ip-172-31-84-251 my-git-project]$ echo "This is my script-1 page" > script1.sh
[ec2-user@ip-172-31-84-251 my-git-project]$ echo "This is my script-2 page" > script2.sh
[ec2-user@ip-172-31-84-251 my-git-project]$ ls
script1.sh  script2.sh
[ec2-user@ip-172-31-84-251 my-git-project]$ cat script1.sh
This is my script-1 page
[ec2-user@ip-172-31-84-251 my-git-project]$ cat script2.sh
This is my script-2 page
[ec2-user@ip-172-31-84-251 my-git-project]$ |
```

Step 3: Initialize a Local Git Repository

- git init

Explanation:

✓ git init creates a new empty Git repository in your directory.



```
[ec2-user@ip-172-31-84-251 my-git-project]$ git init
hint: Using 'master' as the name for the initial branch. This default branch name
hint: is subject to change. To configure the initial branch name to use in all
hint: of your new repositories, which will suppress this warning, call:
hint:
hint:   git config --global init.defaultBranch <name>
hint:
hint: Names commonly chosen instead of 'master' are 'main', 'trunk' and
hint: 'development'. The just-created branch can be renamed via this command:
hint:
hint:   git branch -m <name>
Initialized empty Git repository in /home/ec2-user/my-git-project/.git/
[ec2-user@ip-172-31-84-251 my-git-project]$ |
```

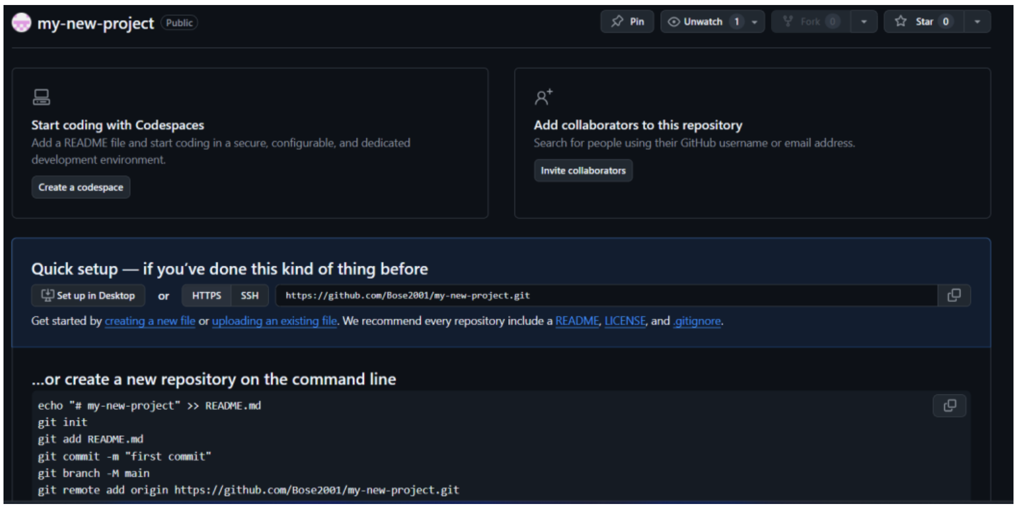
Step 4: Create a New Repository on GitHub

✓ Log into GitHub.

✓ Click on **New Repository**.

✓ Provide a name (e.g., my-new-project) and click **Create Repository**.

Note: Do not initialize it with a README.



Start coding with Codespaces

Add a README file and start coding in a secure, configurable, and dedicated development environment.

Add collaborators to this repository

Search for people using their GitHub username or email address.

Quick setup — if you've done this kind of thing before

https://github.com/Bose2001/my-new-project.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 "# my-new-project" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/Bose2001/my-new-project.git
```

Step 5: Link Local Repository to GitHub

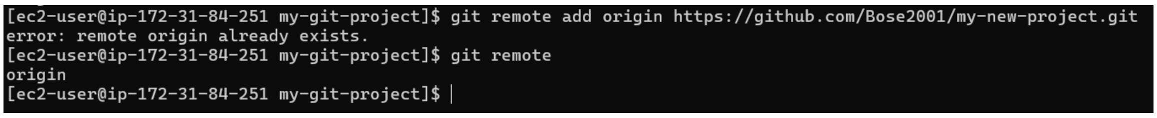
- git remote add origin https://github.com/YourUsername/MyProjectRepo.git
- git branch -M main
- git push -u origin main

Explanation:

✓ git remote add origin [URL] links your local repository.

✓ git branch -M main renames your branch to main.

✓ git push -u origin main pushes your code to GitHub.



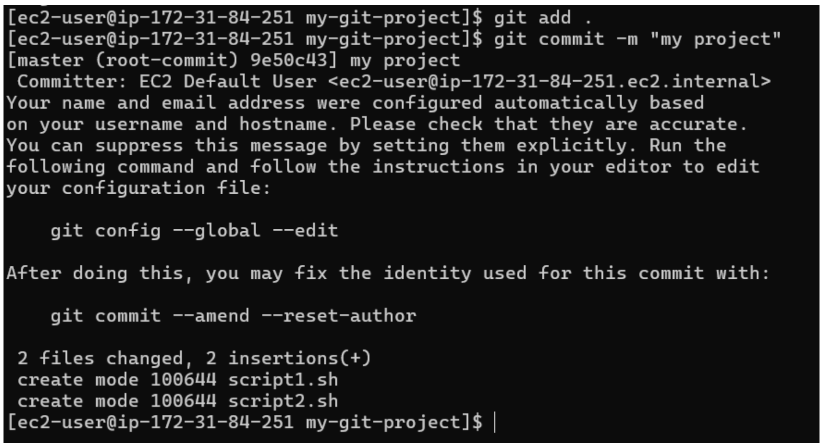
```
[ec2-user@ip-172-31-84-251 my-git-project]$ git remote add origin https://github.com/Bose2001/my-new-project.git
error: remote origin already exists.
[ec2-user@ip-172-31-84-251 my-git-project]$ git remote
origin
[ec2-user@ip-172-31-84-251 my-git-project]$ |
```

Step 6: Stage and Commit Your Files

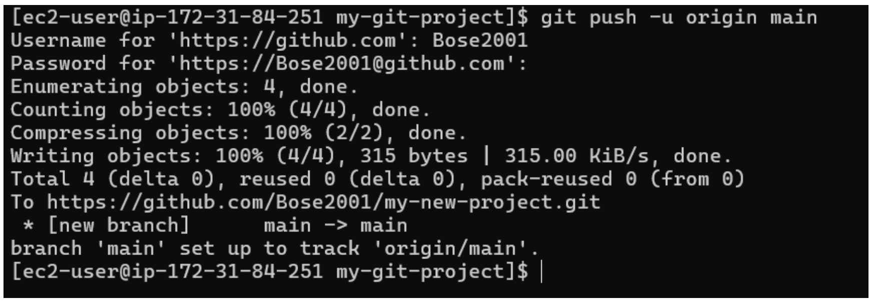
- git add .
- git commit -m "Initial commit with script files"

Explanation:

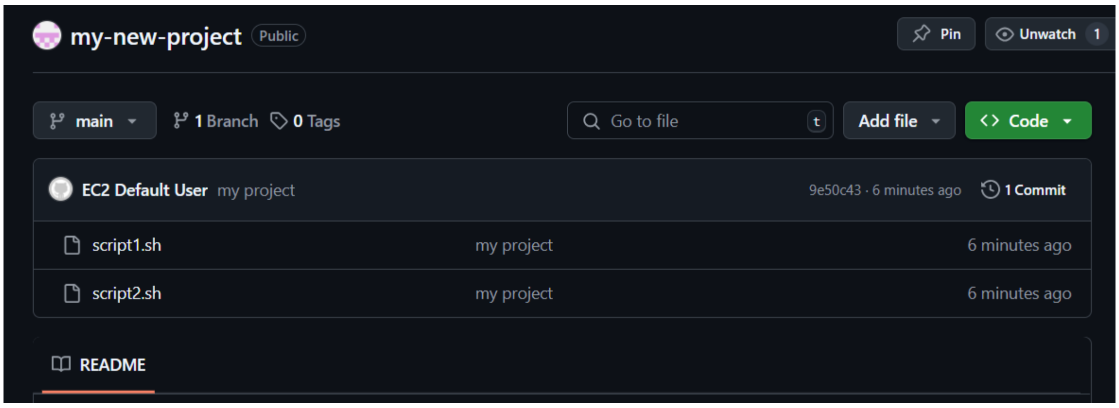
✓ git add . stages all changes in the directory.
✓ git commit -m "message" permanently records the staged changes.



```
[ec2-user@ip-172-31-84-251 my-git-project]$ git add .
[ec2-user@ip-172-31-84-251 my-git-project]$ git commit -m "my project"
[master (root-commit) 9e50c43] my project
 Committer: EC2 Default User <ec2-user@ip-172-31-84-251.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 2 files changed, 2 insertions(+)
 create mode 100644 script1.sh
 create mode 100644 script2.sh
[ec2-user@ip-172-31-84-251 my-git-project]$ |
```



```
[ec2-user@ip-172-31-84-251 my-git-project]$ git push -u origin main
Username for 'https://github.com': Bose2001
Password for 'https://Bose2001@github.com':
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (4/4), 315 bytes | 315.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Bose2001/my-new-project.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
[ec2-user@ip-172-31-84-251 my-git-project]$ |
```



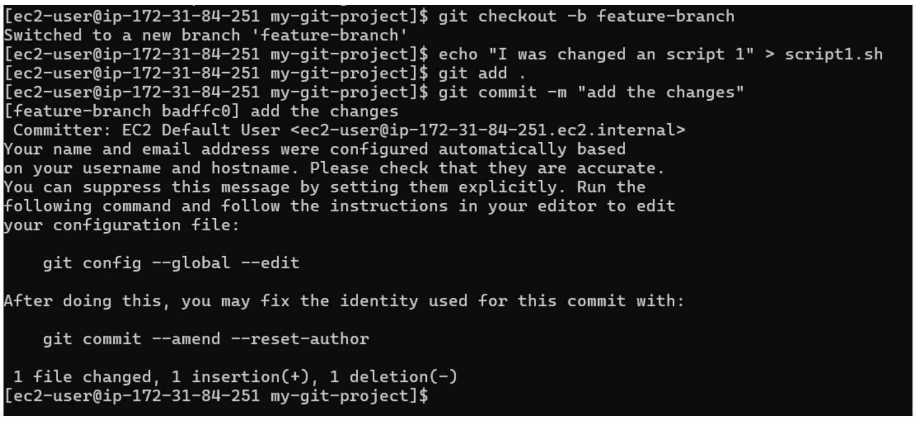
my-new-project Public

EC2 Default User my project		9e50c43 · 6 minutes ago · 1 Commit
script1.sh	my project	6 minutes ago
script2.sh	my project	6 minutes ago

Advanced Git Operations

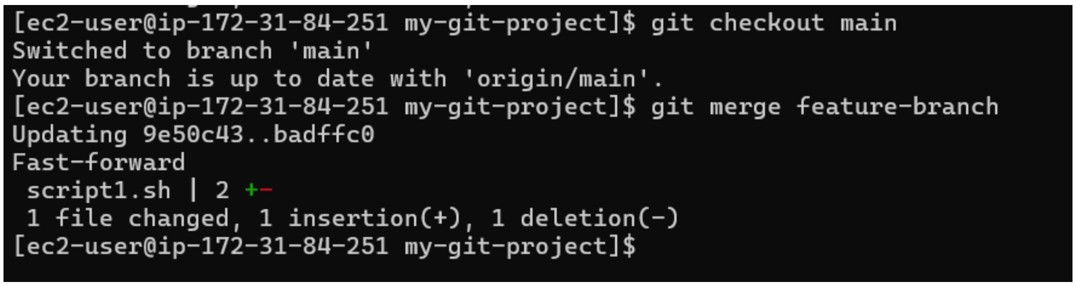
Merge

- ✓ Create a new branch:
    - git checkout -b feature-branch
- ✓ Make changes in your files, then:
    - git add .
    - git commit -m "Changes in feature branch"



```
[ec2-user@ip-172-31-84-251 my-git-project]$ git checkout -b feature-branch
Switched to a new branch 'feature-branch'
[ec2-user@ip-172-31-84-251 my-git-project]$ echo "I was changed an script 1" > script1.sh
[ec2-user@ip-172-31-84-251 my-git-project]$ git add .
[ec2-user@ip-172-31-84-251 my-git-project]$ git commit -m "add the changes"
[feature-branch badffc0] add the changes
 Committer: EC2 Default User <ec2-user@ip-172-31-84-251.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+), 1 deletion(-)
[ec2-user@ip-172-31-84-251 my-git-project]$
```

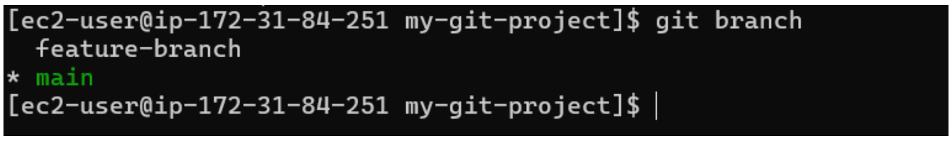
- ✓ Switch back to main and merge:
    - git checkout main
    - git merge feature-branch



```
[ec2-user@ip-172-31-84-251 my-git-project]$ git checkout main
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
[ec2-user@ip-172-31-84-251 my-git-project]$ git merge feature-branch
Updating 9e50c43..badffc0
Fast-forward
 script1.sh | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
[ec2-user@ip-172-31-84-251 my-git-project]$
```

Explanation:

- ✓ git checkout -b [branch-name] creates and switches to a new branch.
- ✓ git merge [branch-name] merges the specified branch into your current branch.

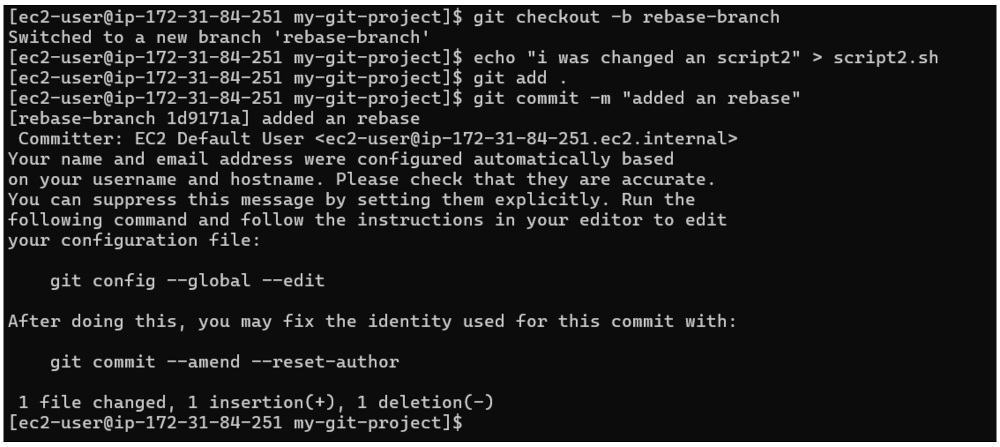


```
[ec2-user@ip-172-31-84-251 my-git-project]$ git branch
  feature-branch
* main
[ec2-user@ip-172-31-84-251 my-git-project]$ |
```

Rebase

- ✓ Suppose you're on feature-branch and want to rebase onto main:
    - git checkout -b rebase-branch

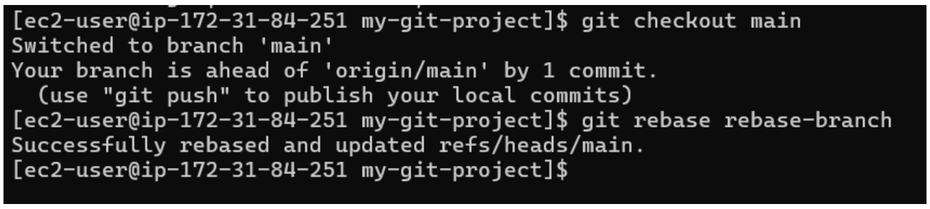
- ✓ Make changes and commit:
    - echo "I was changed an script2" > script2.sh
    - git add .
    - git commit -m “added an rebase”



```
[ec2-user@ip-172-31-84-251 my-git-project]$ git checkout -b rebase-branch
Switched to a new branch 'rebase-branch'
[ec2-user@ip-172-31-84-251 my-git-project]$ echo "i was changed an script2" > script2.sh
[ec2-user@ip-172-31-84-251 my-git-project]$ git add .
[ec2-user@ip-172-31-84-251 my-git-project]$ git commit -m "added an rebase"
[rebase-branch 1d9171a] added an rebase
 Committer: EC2 Default User <ec2-user@ip-172-31-84-251.ec2.internal>
Your name and email address were configured automatically based
on your username and hostname. Please check that they are accurate.
You can suppress this message by setting them explicitly. Run the
following command and follow the instructions in your editor to edit
your configuration file:

    git config --global --edit

After doing this, you may fix the identity used for this commit with:

    git commit --amend --reset-author

 1 file changed, 1 insertion(+), 1 deletion(-)
[ec2-user@ip-172-31-84-251 my-git-project]$
```

✓ Instead of merging, **rebase** onto master:

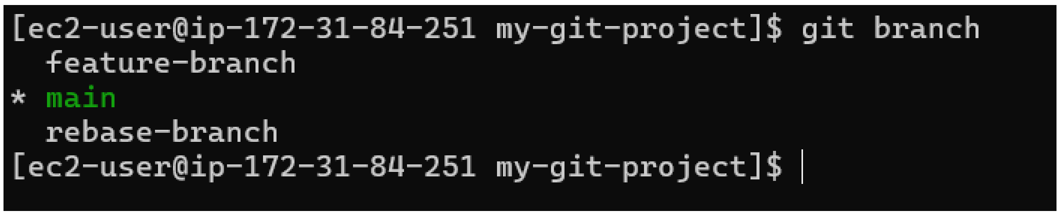
- git checkout main
- git rebase rebase-branch



```
[ec2-user@ip-172-31-84-251 my-git-project]$ git checkout main
Switched to branch 'main'
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)
[ec2-user@ip-172-31-84-251 my-git-project]$ git rebase rebase-branch
Successfully rebased and updated refs/heads/main.
[ec2-user@ip-172-31-84-251 my-git-project]$
```

Explanation:

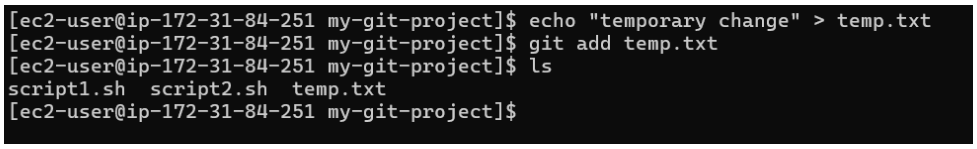
✓ git rebase [branch-name] reapplies your branch commits on top of the specified branch.



```
[ec2-user@ip-172-31-84-251 my-git-project]$ git branch
  feature-branch
* main
  rebase-branch
[ec2-user@ip-172-31-84-251 my-git-project]$
```

Stash

✓ First create an simple text file named as temp.txt



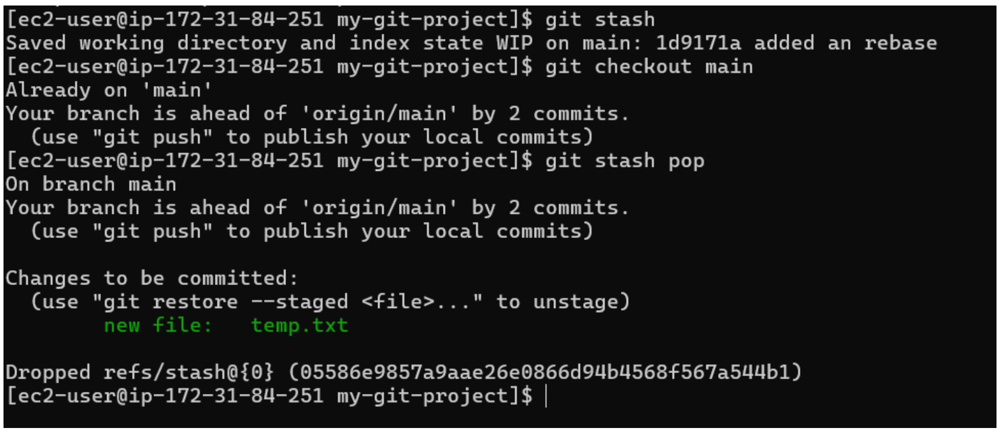
```
[ec2-user@ip-172-31-84-251 my-git-project]$ echo "temporary change" > temp.txt
[ec2-user@ip-172-31-84-251 my-git-project]$ git add temp.txt
[ec2-user@ip-172-31-84-251 my-git-project]$ ls
script1.sh  script2.sh  temp.txt
[ec2-user@ip-172-31-84-251 my-git-project]$
```

✓ Suppose you have uncommitted changes but want to switch branches quickly:

- git stash
- git checkout main

✓ Later, you can reapply your stashed changes:

- git stash pop

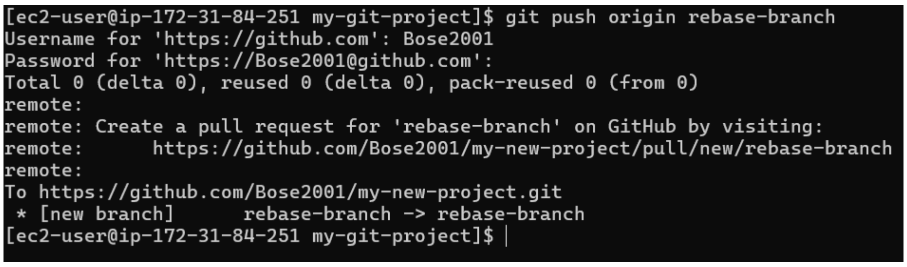


```
[ec2-user@ip-172-31-84-251 my-git-project]$ git stash
Saved working directory and index state WIP on main: 1d9171a added an rebase
[ec2-user@ip-172-31-84-251 my-git-project]$ git checkout main
Already on 'main'
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)
[ec2-user@ip-172-31-84-251 my-git-project]$ git stash pop
On branch main
Your branch is ahead of 'origin/main' by 2 commits.
  (use "git push" to publish your local commits)

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   temp.txt

Dropped refs/stash@{0} (05586e9857a9aae26e0866d94b4568f567a544b1)
[ec2-user@ip-172-31-84-251 my-git-project]$
```

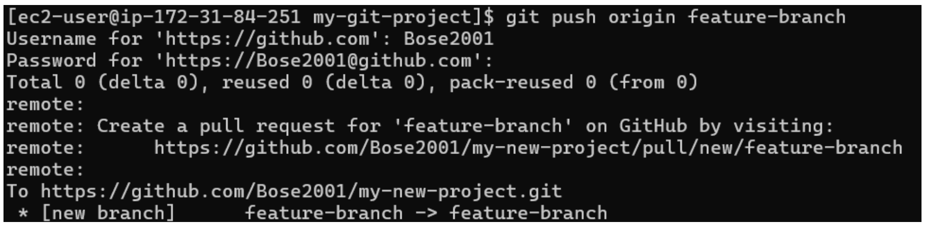
Explanation:

- ✓ git stash saves your uncommitted changes.
- ✓ git stash pop reapplies the most recent stashed changes.

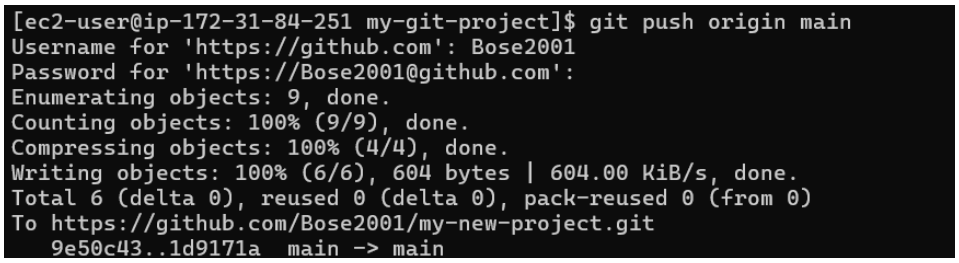
Finally push all the branch which was locally created move to the remote github



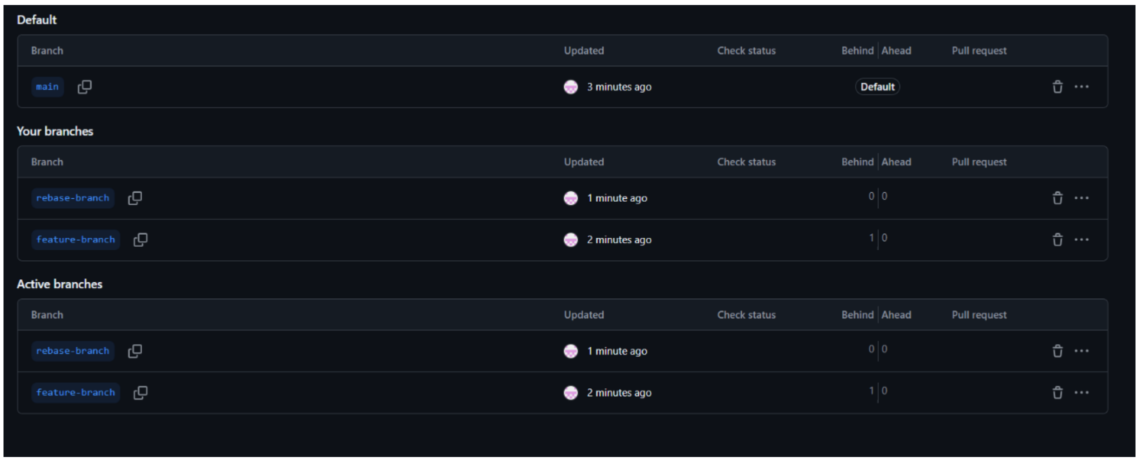
```
[ec2-user@ip-172-31-84-251 my-git-project]$ git push origin rebase-branch
Username for 'https://github.com': Bose2001
Password for 'https://Bose2001@github.com':
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'rebase-branch' on GitHub by visiting:
remote:      https://github.com/Bose2001/my-new-project/pull/new/rebase-branch
remote:
To https://github.com/Bose2001/my-new-project.git
 * [new branch]      rebase-branch -> rebase-branch
[ec2-user@ip-172-31-84-251 my-git-project]$
```



```
[ec2-user@ip-172-31-84-251 my-git-project]$ git push origin feature-branch
Username for 'https://github.com': Bose2001
Password for 'https://Bose2001@github.com':
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'feature-branch' on GitHub by visiting:
remote:      https://github.com/Bose2001/my-new-project/pull/new/feature-branch
remote:
To https://github.com/Bose2001/my-new-project.git
 * [new branch]      feature-branch -> feature-branch
```



```
[ec2-user@ip-172-31-84-251 my-git-project]$ git push origin main
Username for 'https://github.com': Bose2001
Password for 'https://Bose2001@github.com':
Enumerating objects: 9, done.
Counting objects: 100% (9/9), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 604 bytes | 604.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Bose2001/my-new-project.git
   9e50c43..1d9171a  main -> main
```



Default

Branch	Updated	Check status	Behind | Ahead	Pull request
main	3 minutes ago		Default	

Your branches

Branch	Updated	Check status	Behind | Ahead	Pull request
rebase-branch	1 minute ago		0 | 0	
feature-branch	2 minutes ago		1 | 0	

Active branches

Branch	Updated	Check status	Behind | Ahead	Pull request
rebase-branch	1 minute ago		0 | 0	
feature-branch	2 minutes ago		1 | 0	

Summary of Important Git Commands

Command	**Purpose**
git init	Initialize a new Git repository
git add .	Stage all files for commit
git commit -m "message"	Commit changes with a message
git remote add origin [URL]	Connect local repo to GitHub
git push -u origin main	Push code to GitHub
git checkout -b branch-name	Create and switch to a new branch
git merge branch-name	Merge a branch into another
git rebase branch-name	Reapply commits from one branch onto another
git stash	Save uncommitted changes temporarily
git stash pop	Reapply stashed changes

************** **TASK COMPLETED** **************