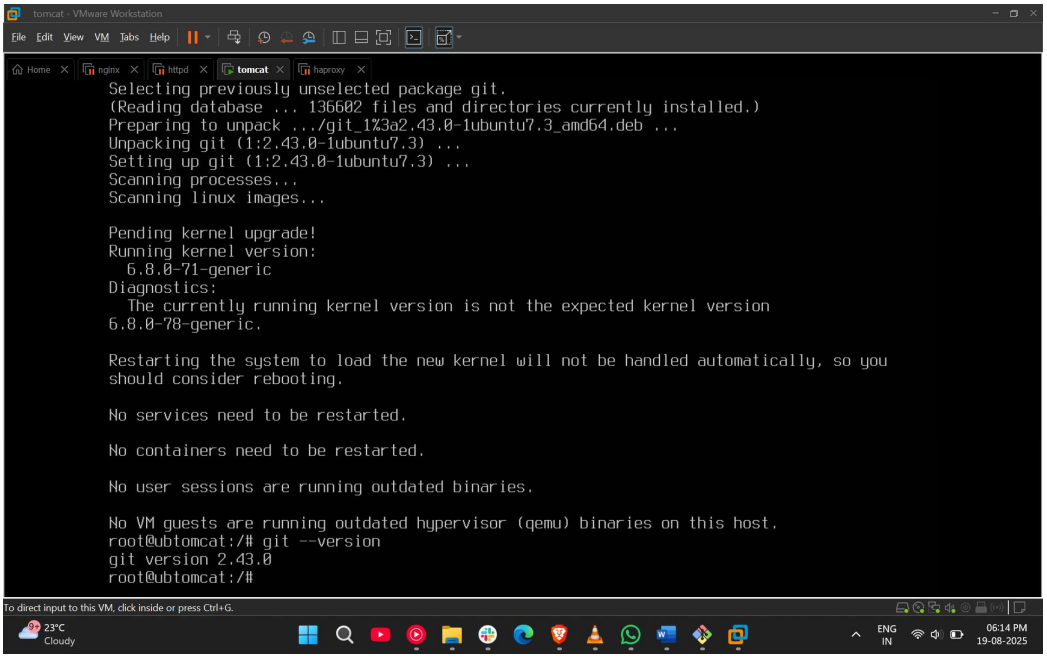
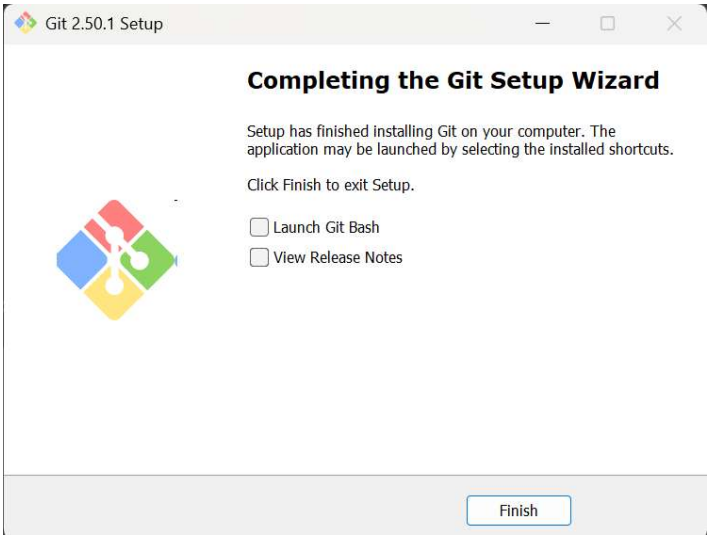
Tasks on Git and GitHub

1)Install git on the local machine.

Objective: Install Git on the local machine and show the output with screenshots.





Procedure:

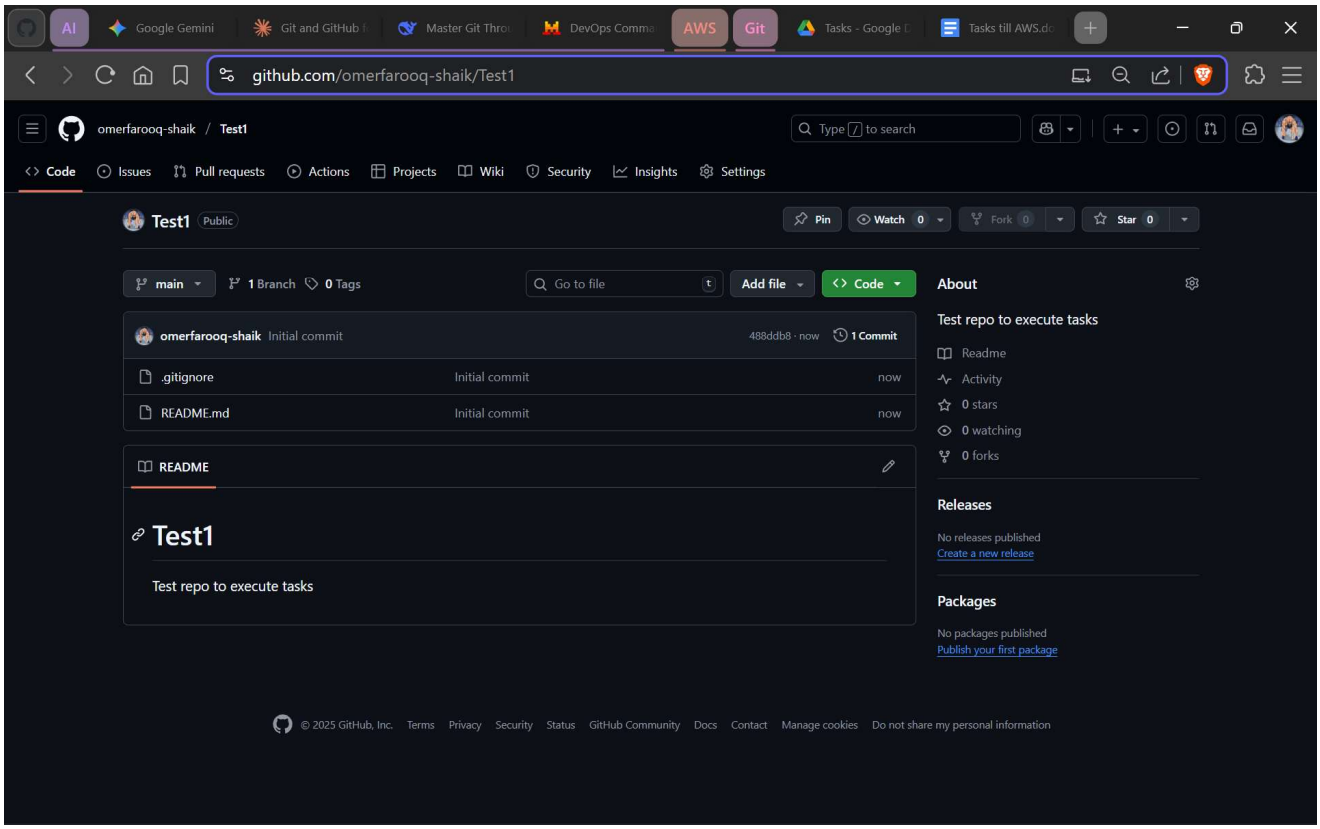
1. Steps to install Git in Windows:
   a) Open a web browser and search for “Git for Windows.”
   b) Select the Windows version of Git and download it.
   c) Open the downloaded file and click on install, and follow the installation process.
   d) Open Git is installed, click on finish to close the installation window.
   e) Open the Git Bash and type “git --version” to check if Git is installed successfully, or go to Program Files and check for the “Git” folder.
2. Steps to install Git in Linux:
   a) Open Linux using a virtual machine or on an Amazon EC2 instance.

b) For Ubuntu, type: “apt install git”, and for Amazon Linux, type: “yum install git”.
c) Once the installation is complete, check for the version of Git using – “git --version” command.

The outputs of both processes are attached above.

2)Create a repo in GitHub with README.md and .gitignore file.

Objective: Create a repository in GitHub with README.md and .gitignore file.



Steps to create a Repository:

a) Open a browser and open the “github.com” website
b) Create an account. If an account already exists, then directly jump to the next step.
c) Click on the “Repositories” section and click on “New”.
d) Create a new repository by giving it a name, enabling “readme”, and selecting a “.gitignore”. You can also choose to make it public or private.
e) Click on create repository to finish creating the repository.

Created a repository with readme.md and .gitignore files. The results can be seen above.

3. Clone the created repo on the local machine.

Objective: To clone a remote repository from GitHub to the local machine

```
iamom@Omer MINGW64 ~/Downloads (master)
$ git clone https://github.com/omerfarooq-shaik/Test1.git
Cloning into 'Test1'...
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 4 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Receiving objects: 100% (4/4), done.

iamom@Omer MINGW64 ~/Downloads (master)
$
```

Procedure:

1. Go to the GitHub website and copy the repository URL(HTTPS) from the drop-down.
2. In the local machine, type the following syntax:

$ git clone <url>

3. Once the cloning is done, a message with 'done' will be displayed on the screen, like in the screenshot.

4) Create two files in the local repo.

Objective: To create two files in the local repository that were cloned from GitHub.

```
iamom@Omer MINGW64 ~/Downloads (master)
$ cd Test1

iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$ ls
README.md

iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$ touch 1.txt 2.txt 3.txt

iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$ ls
1.txt  2.txt  3.txt  README.md

iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$
```

Procedure:

1. Navigate to the local repository with the 'cd' command, i.e., "cd repo_name"
2. Once in the local repository, create files with the 'touch' command.
3. To check the output, use the 'ls' command.
4. The created files can be seen in the above image.

5) Commit two files and push to the central Repository.

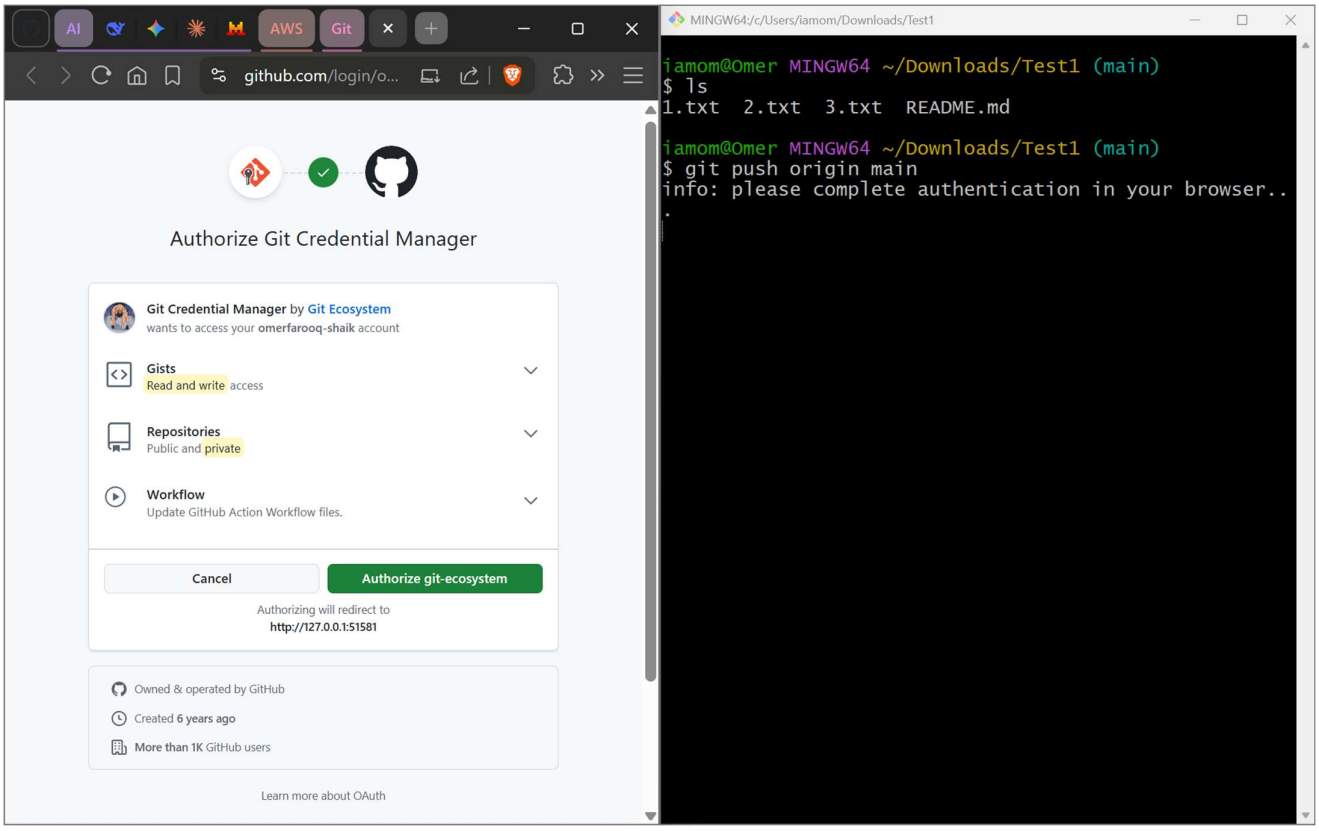
Objective: To commit the files in the local repository to the central repository

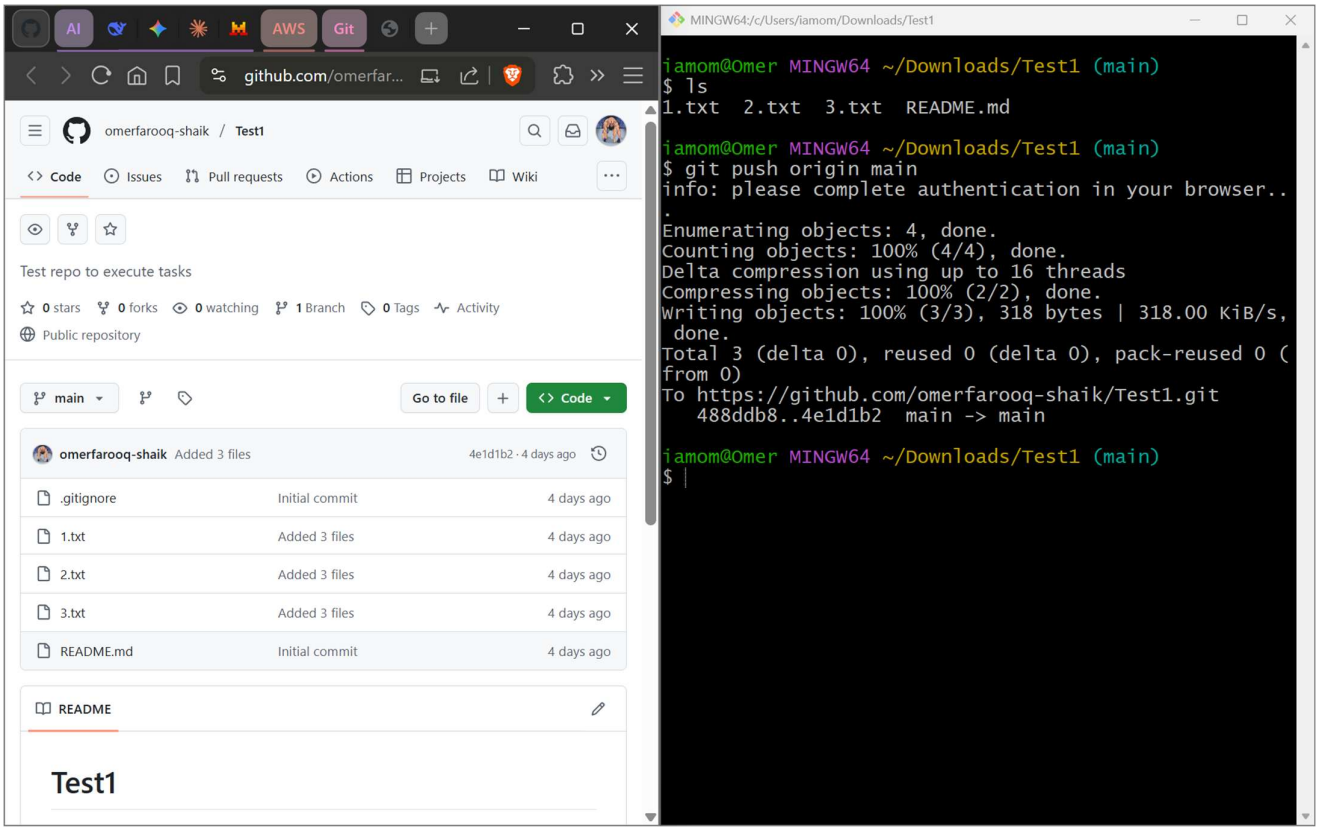
Commit:

```
iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$ ls
1.txt  2.txt  3.txt  README.md

iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$ git add 1.txt 2.txt 3.txt

iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   1.txt
        new file:   2.txt
        new file:   3.txt


iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$ git commit -m "Added 3 files"
[main 4e1d1b2] Added 3 files
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 1.txt
 create mode 100644 2.txt
 create mode 100644 3.txt

iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$
```

Push:

Objective: To push files from the local repository to the remote central repository.





Commit Procedure:

1. At first, we need to add the files of the working directory to the staging area. To do that, we use:
$ git add <filename> or git add . (To move all the files to the staging area) or git add *.extension (To move specific files with extension, eg: .txt, .py)

2. After moving the files to the staging area, we commit them and move them to the local repository.

3. This action does two things - it moves files to the local repo, and also takes a snapshot of all the changes. To perform it, we use the following command:
$ git commit -m “message” ex – $ git commit -m “Committed three files”

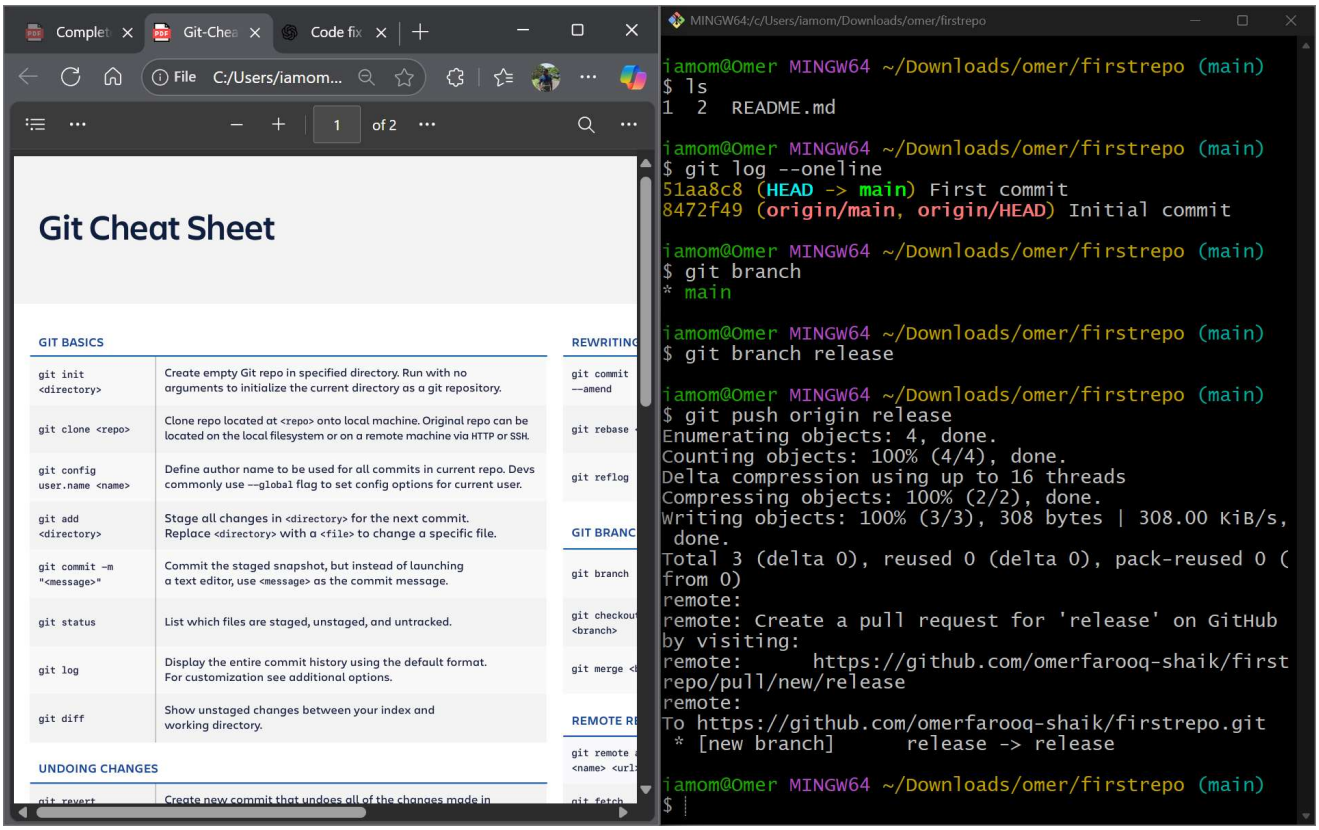
Push Procedure:

1. Once the files are committed, the next step is to push the files to the central (remote) repository. We do it to save the work done on the local machine by using the following command:
$ git push origin <main repo name> eg: $ git push origin main

2. It may ask you to log in if you haven’t already and authorize the procedure. Once done, the push procedure will be completed.
3. The results and steps of the whole procedure can be seen in the above images.

6) Create a branch in local and create a sample file and push to central.

Objective: To create a new branch in the local repository. Switch to the repo and create a sample file. Push it to the central repository.



Procedure:

1. Create a new branch in the local repository using any of the following commands:
$ git branch <branch_name> or $ git checkout -b <branch_name>.

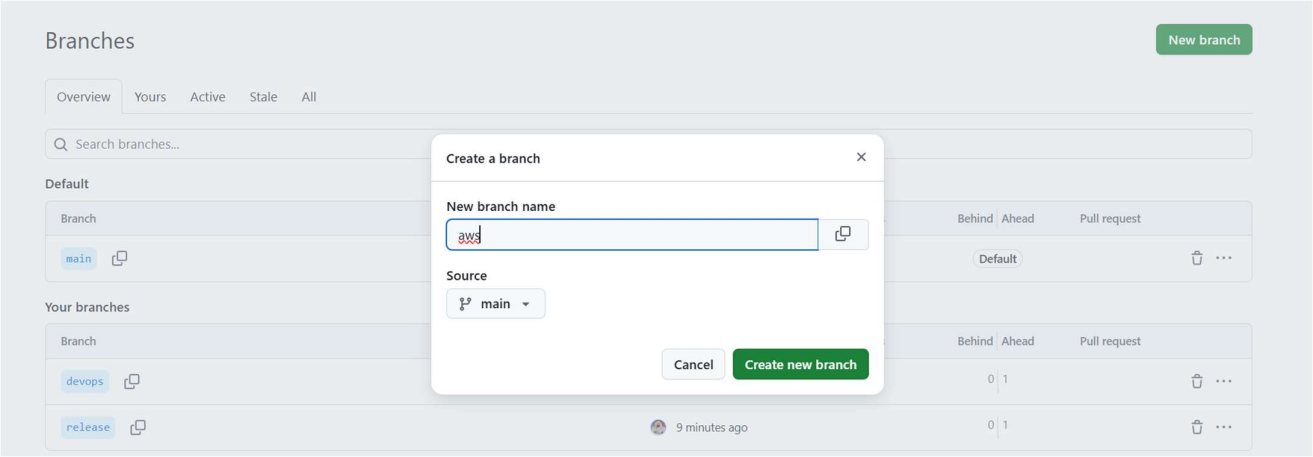
2. Create a sample file in the newly created branch.
3. Add the file to the staging area using:
$ git add <filename>

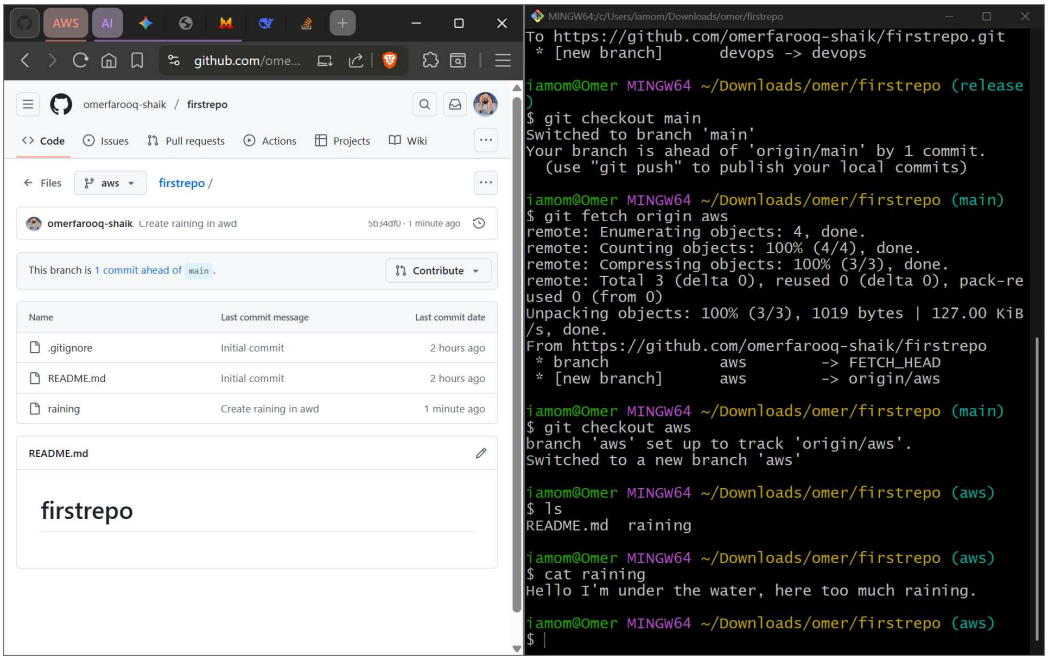
4. Commit the work using the following command:
$ git commit -m “commit message”

5. Now push the work to the central repository using the following command:
$ git push origin main
6. The result can be seen in the above image.

7) Create a branch in GitHub and clone it to the local repository.

Objective: To create a branch in the local repository and clone it to the central/remote repository.





Procedure:

1. Open the GitHub website in a browser and log in to your GitHub account.

2. On GitHub, navigate to the main page of the repository.

3. From the file tree view on the left, select the branch dropdown menu, then click **View all branches**. You can also find the branch dropdown menu at the top of the integrated file editor.

4. Click New Branch. Under "Branch name", type a name for the branch. Click **Create branch**.

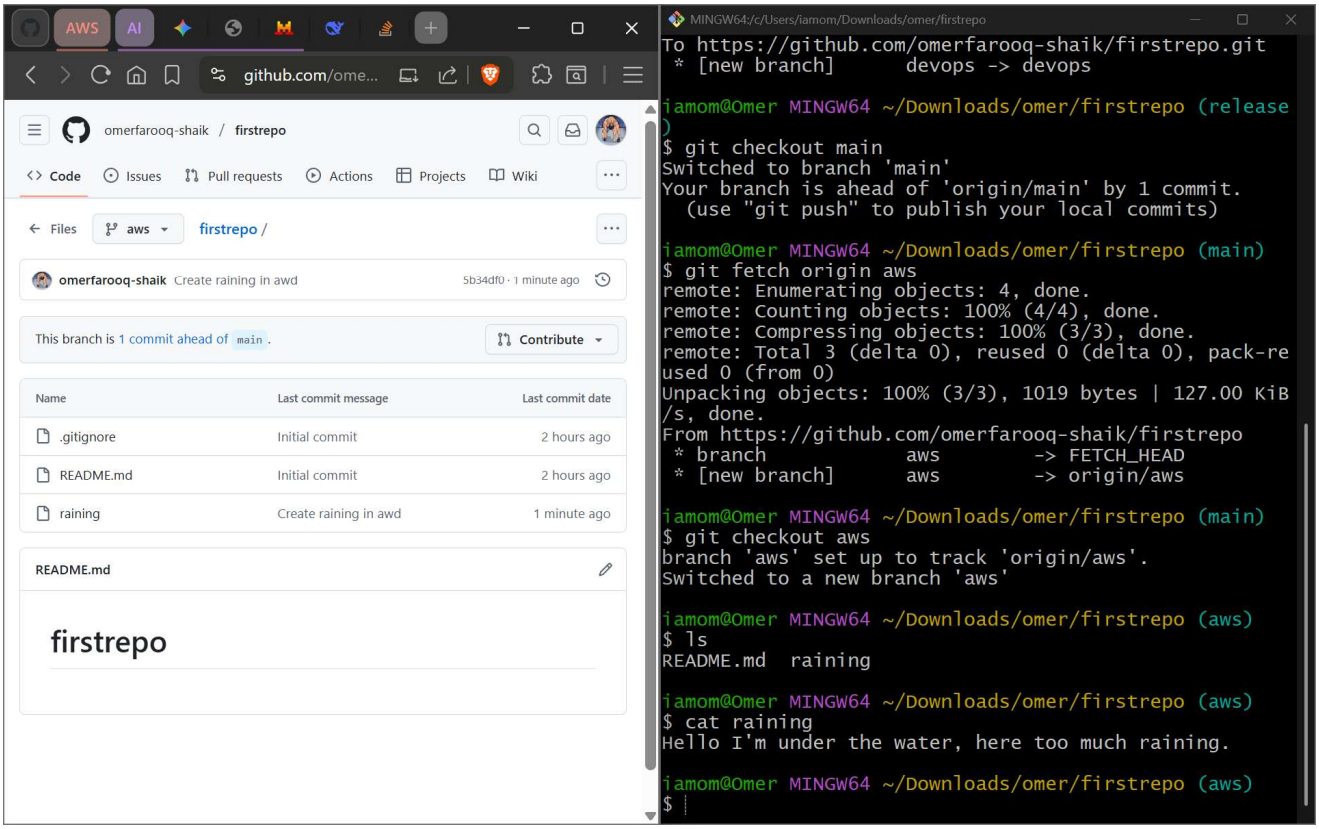
5. On the local machine, run Git Bash. Open the repository that was cloned from the remote repository.

6. Type the following command to clone the branch from the remote to the local
$ git clone --branch <branch_name> url(of the remote repository).

7. The branch from the remote repository will be cloned to the local repository. The results can be seen in the above image.

8) Merge the created branch with master in local Git.

Objective: To create a branch in the central repository and merge it into the local Git.



Procedure:

1. Log in to the GitHub account and look for the branches tab.

2. Click on the branches tab and create a new branch by giving it a name and selecting the appropriate source.

3. Click on ‘Create a new branch’. This creates a new branch.

4. On the local machine, open Git Bash.

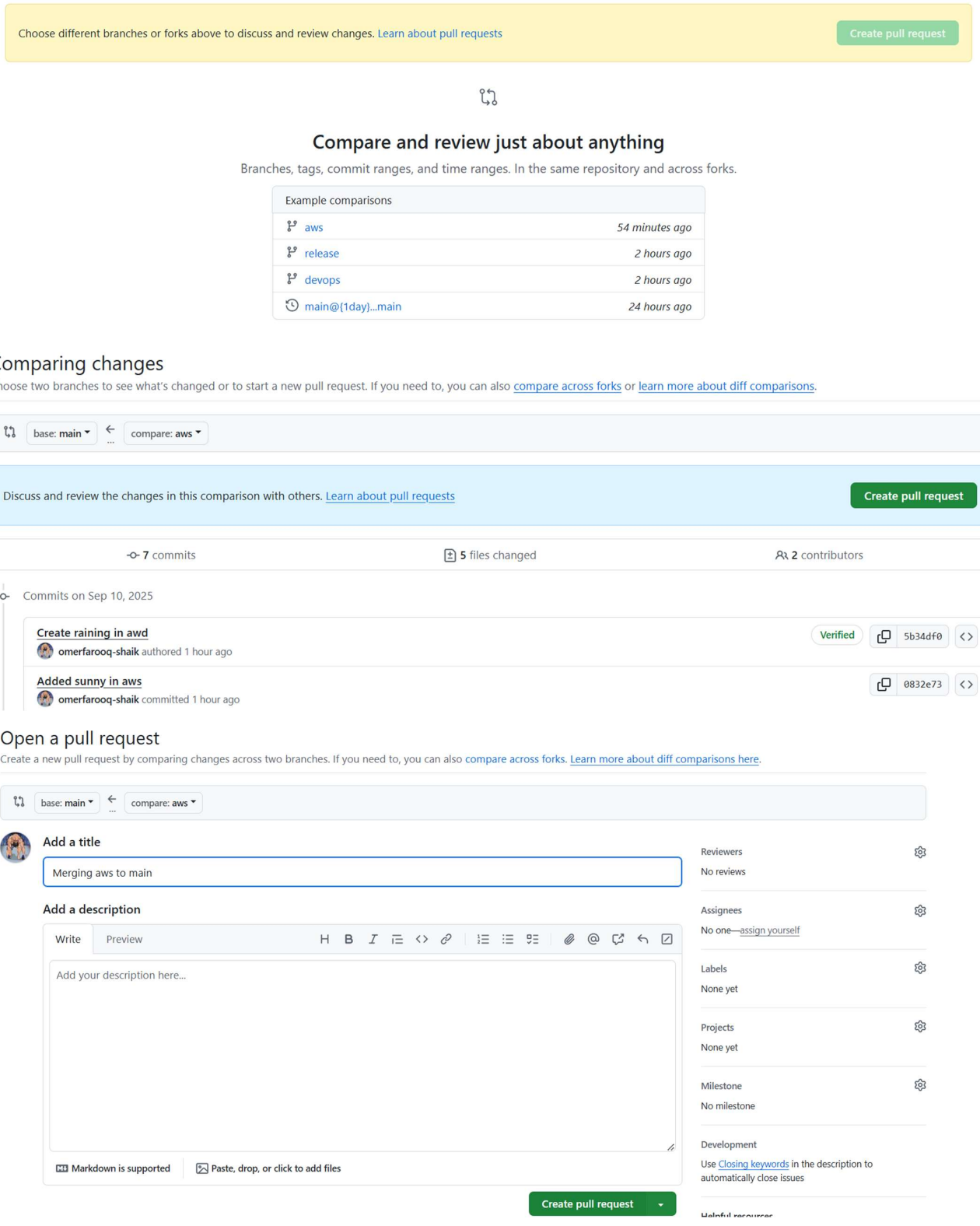
5. To merge the branch created on the central/remote repository, use the following command:
$ git fetch origin <branch_name>.
$ git fetch origin aws -> ‘aws’ is the name of the branch we are trying to merge into the local repository.

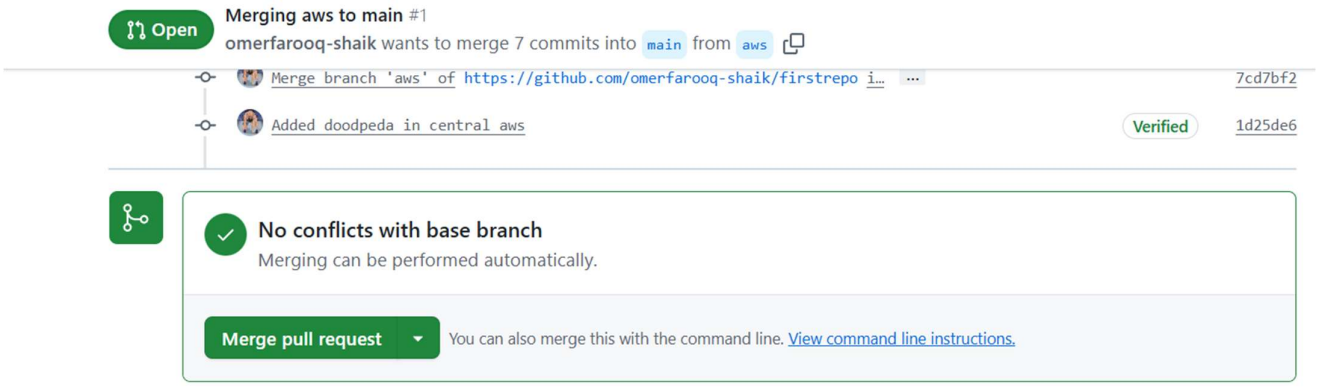
6. To check if the branch is merged successfully, use the following command:
$ git checkout <branch_name>, in our case it is $ git checkout aws

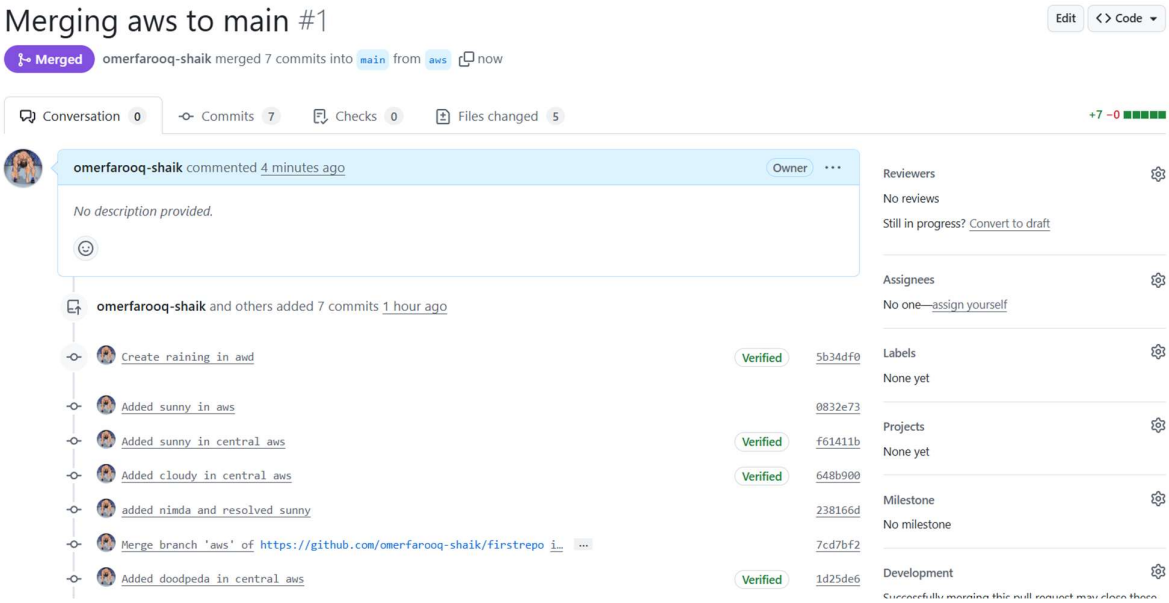
7. The remote and local branches are now successfully merged, and the output of the task can be seen in the above image.

9) Merge the created branch with master in GitHub by sending a pull request.

Objective:







Procedure:

1. Go to your repository on GitHub in a web browser.

2. Click the "Pull requests" tab. Click the "New pull request" button.

3. Select main as the base branch and aws as the compare branch.

4. Click on "Create pull request" and add a title and description for your PR, explaining the changes you made.

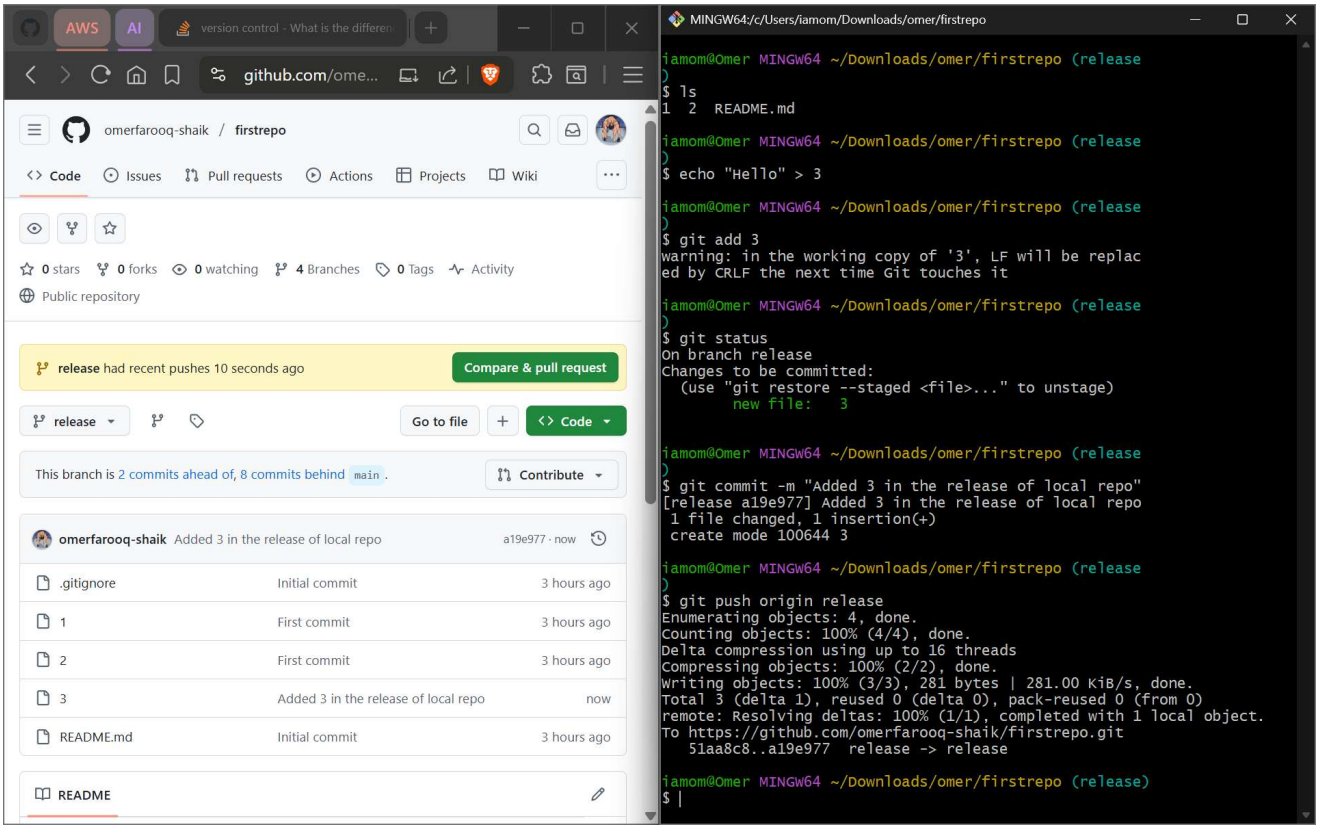
5. Switch to the main branch. Click on the “Pull requests” tab.

6. Check the newly created PR, review it, and click on the “merge pull request” tab.

7. Once merged successfully, the next page displays all the details of the merged branch.

10) Create a file in the local repo and send that to a branch in GitHub.

Objective: To create a file in the local repository and push it to the central repository (GitHub).



Procedure:

1. Create a file in the local repository
$ echo “file_content” > file_name i.e., $ echo “Hello” > 3

2. Add the file to the staging area.
$ git add <file_name>, i.e., $ git add 3

3. Commit the file to the local repository.
$ git commit -m “commit_message”, i.e., $ git commit -m “Added file 3 in the release branch of local repo”

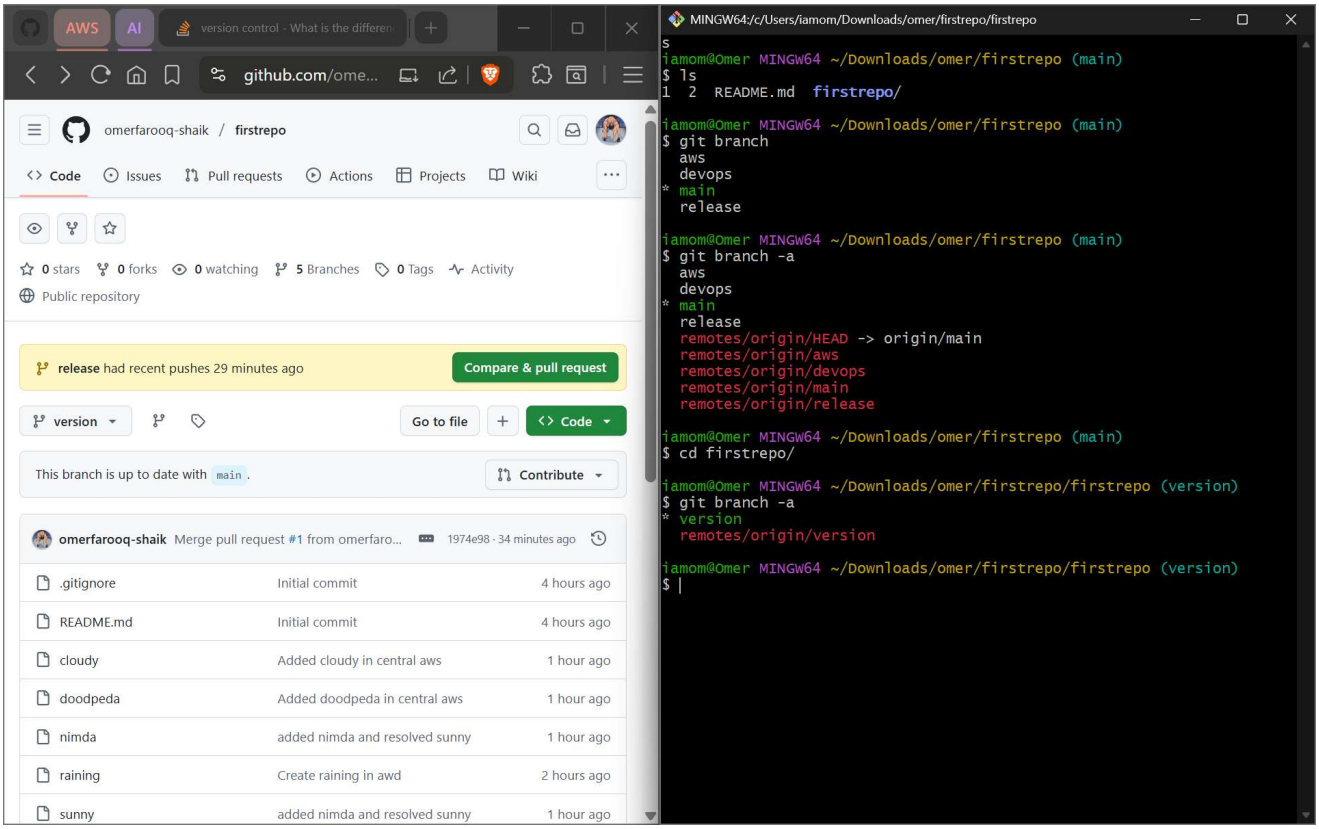
4. To send the file to the central repository (GitHub), we use the ‘push’ method. Git push allows us to transfer files from the local repository to the remote repository.

5. To do that, we use the following command:
$ git push origin <branch_to_push>
$ git push origin release

6. The file is now successfully copied to the remote repository, i.e., pushed to the central repository and the results can be seen in the above image.

11) Clone only a branch from GitHub to the local.

Objective: To clone a single branch from the remote repository to the local working area.



Procedure:

1. A repository may have different branches, but we may not need all the branches at the same time. Sometimes, we may need only one branch to work on it.

2. In any real-life scenario, and at the workplace, we do not have access to the full repository. We work on a small part of a big project.

3. In that case, we only pull one branch from the remote/central repository.

4. To pull a single branch from the central repository, use the following command:
$ git clone -b <branch_name> --single-branch <GitHub_Repository_URL>
$ git clone -b aws --single-branch https://github.com/omerfarooq-shaik/firstrepo.git

5. A new repository will be created where we clone the branch. It consists of a clone of the remote repository that we requested.

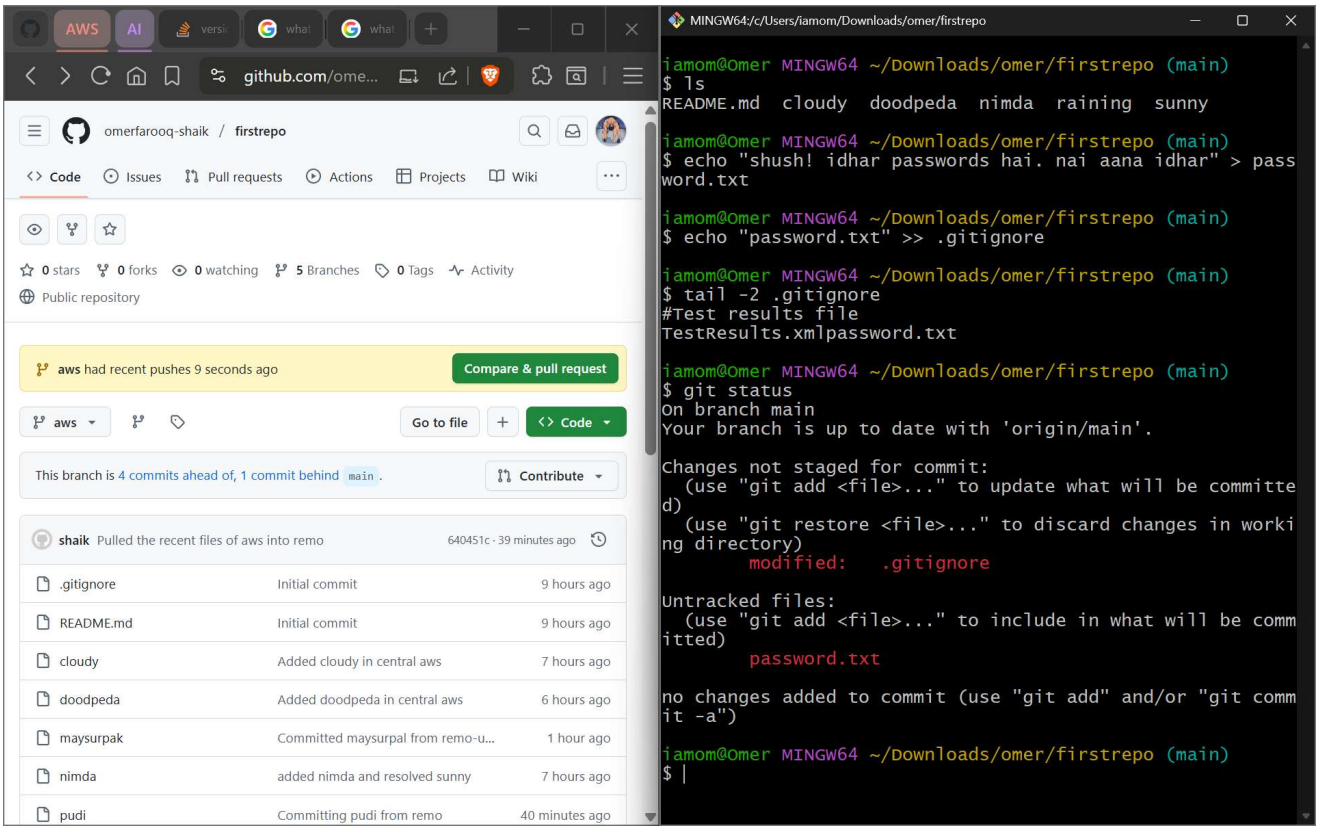
6. To check if the branch is cloned or not, open the directory using:
$ cd <Repository_Name> i.e., $ cd firstrepo/

7. Inside the repository, check if the details of the remote branch are cloned or not using:
$ ls && $ git branch -a

8. The above procedure cloned the branch. Result can be seen in the above image.

12) Create a file with all passwords and make that untrackable with git.

Objective: To create a file with sensitive data, like passwords, and make it untrackable by adding it to the .gitignore.



Procedure:

1. When we create a repository in GitHub, it will give us an option to select the .gitignore file. We must choose that file in order to perform this task. If the .gitignore file is not there in the cloned repository, this task cannot be performed.

2. Create a file and give it a name. Ex – echo “email_password” password.txt

3. Follow the step to add the file to the .gitignore file:
$ echo “password.txt” >> .gitignore

4. Check if the file is already tracked.
$ git ls-files | grep password.txt

5. If nothing shows up, you’re safe. If it shows, Git is already tracking it.

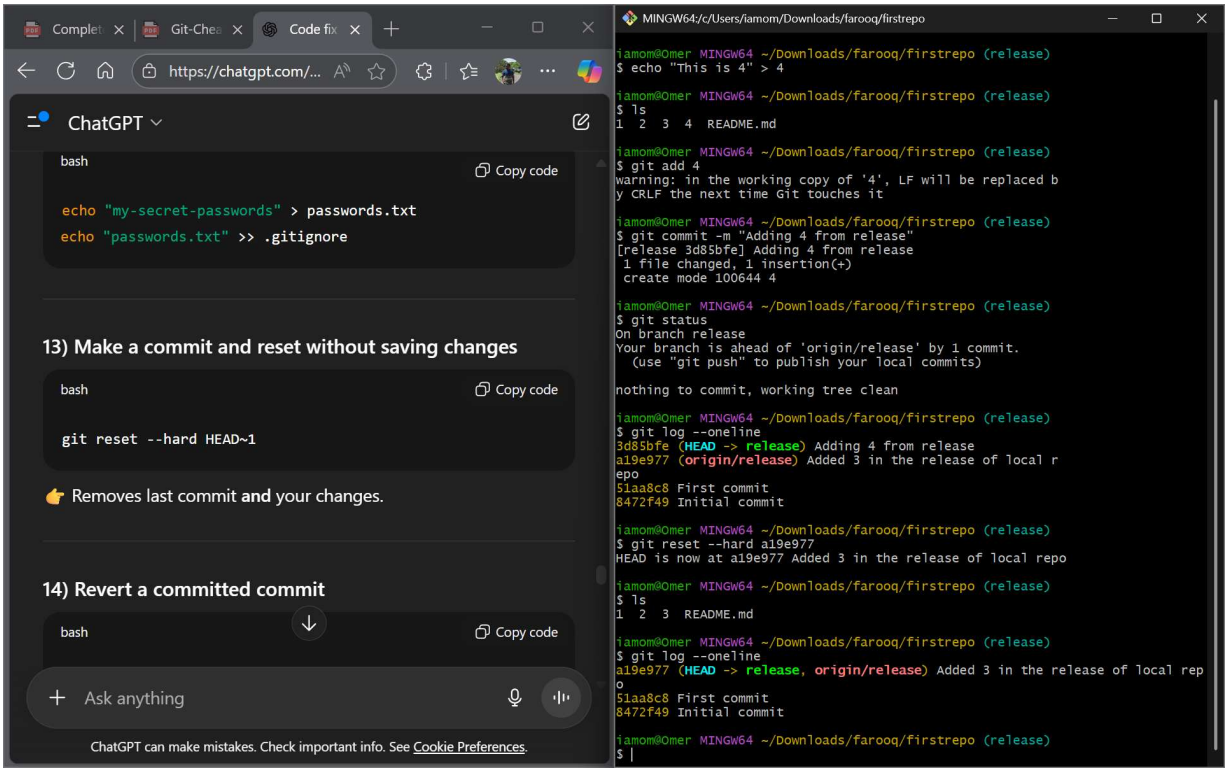
6. If it’s already tracked, untrack it using:
$ git rm --cached password.txt

7. Commit the changes and push to the remote.
$ git commit -m “Added password.txt to .gitignore” && git push origin main

8. We have successfully moved the file to .gitignore, and it’s untrackable.

13) Make a commit and make that commit reset without saving changes.

Objective: To commit and reset the commit while also deleting the changes in the logs.



Procedure:

1. There are two ways to perform the above task. I have adopted a long version instead of a short. The long version explains what's happening in detail. The first method can be seen on the left side of the image.

2. To perform the task, we first create a file with any name, in my case, 4.

3. Add the file to the staging area and then commit it.
$ git add 4 && $ git commit -m "Adding 4 from release branch"

4. Check the logs to see the recent commit. Every commit comes with a commit ID. That can be seen in the image attached. The most recent commit has HEAD beside the commit ID.
$ git log (or) git log --oneline

5. To reset the commit and also delete the commit ID of the second last commit. Why? Because we want to go back to that commit and delete the present commit.

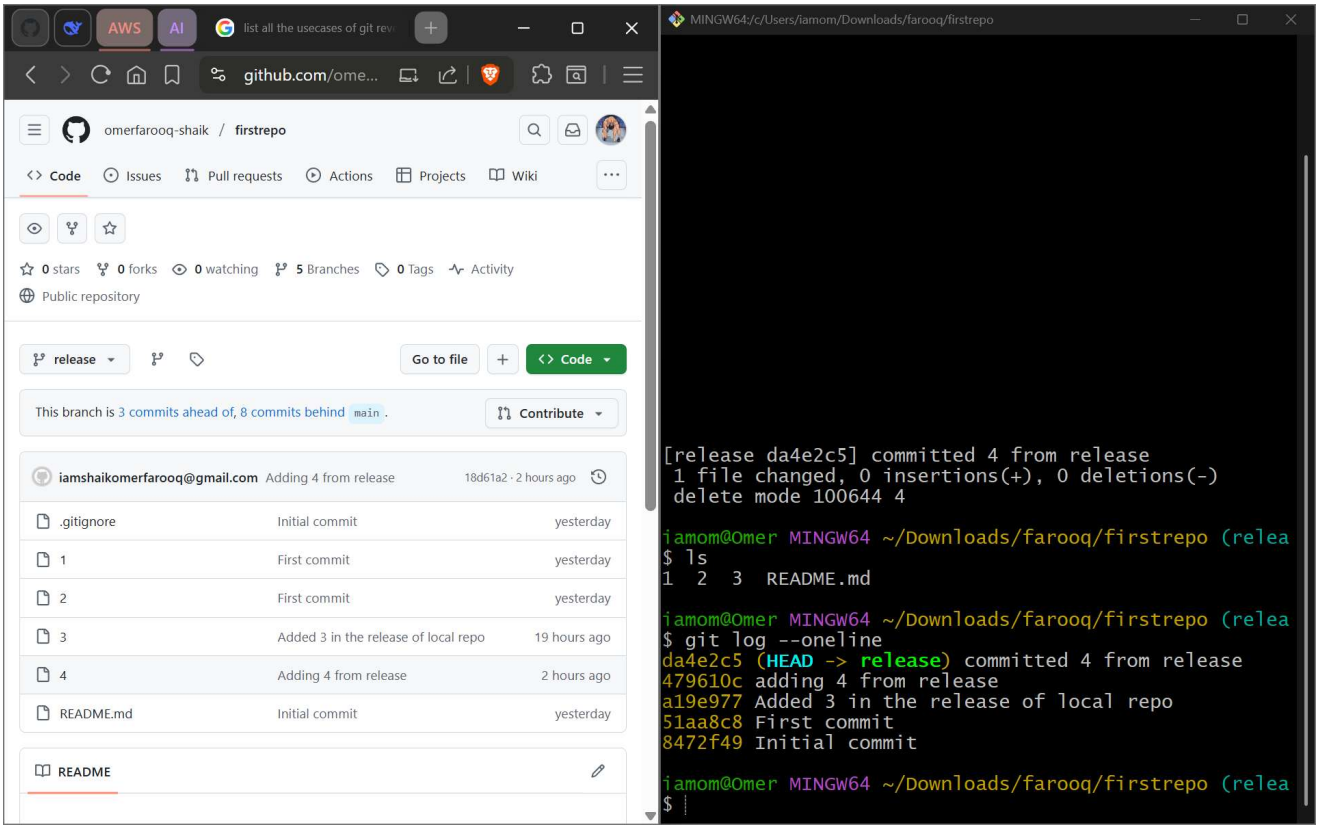
6. To do that, use the 'reset' command. Syntax is as follows:
$ git reset –hard <previous_commit>

7. There are 3 ways we use the reset command – 'git reset --hard', 'git reset --soft', and just 'git reset'.

8. The difference between 3 is that:
i. The hard reset deletes the commit ID as well as the file that we created and committed.
ii. The soft reset just deletes the commit ID, doesn't delete the file, and keeps the file in the staging area.
iii. The reset command deleted the commit ID, doesn't delete the file, removes it from the staging area, and adds it to the working directory.

14) Revert a committed commit to the older version.

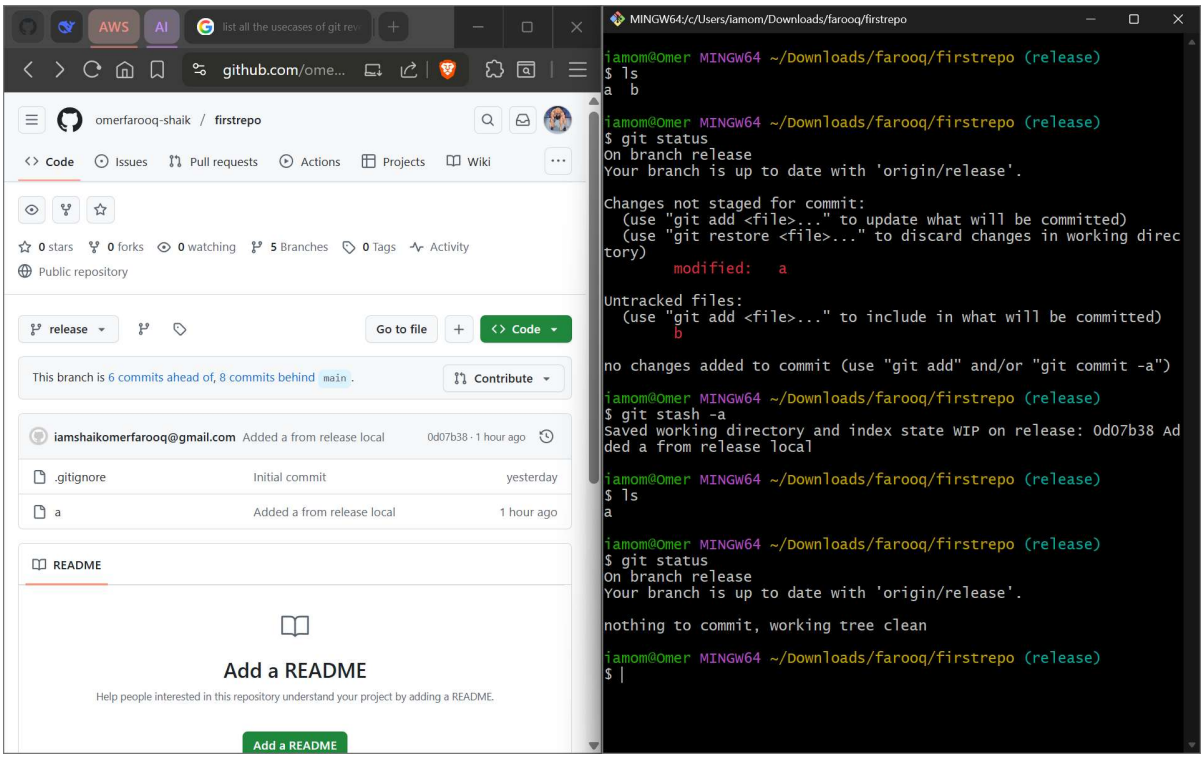
Objective:



Procedure:

15) Push a file to stash without saving the changes and work on another file.

Objective: To push a file without saving the changes to stash and work on another file which is located elsewhere.



Procedure:

1. We use stash if we're working on both files at the same time and we aren't ready to stage them yet.

2. Stash works on committed, pushed, and untracked files.

3. To perform it and understand how exactly stash works, I created two files – a and b. The file "a" is pushed to the central repository, while "b" is still an untracked file.

4. I have made some changes to file "a", which was already pushed. Then, I started working on file "b". I haven't staged and committed both the files after completing the work, and upon checking the log, file "a" will be in a modified state, while "b" will be an untracked file. Ref the above image.

5. Now, if I want to switch to other work and also don't want my current work to stop or save it from loss, I stash it and remove it from the untracked area temporarily.

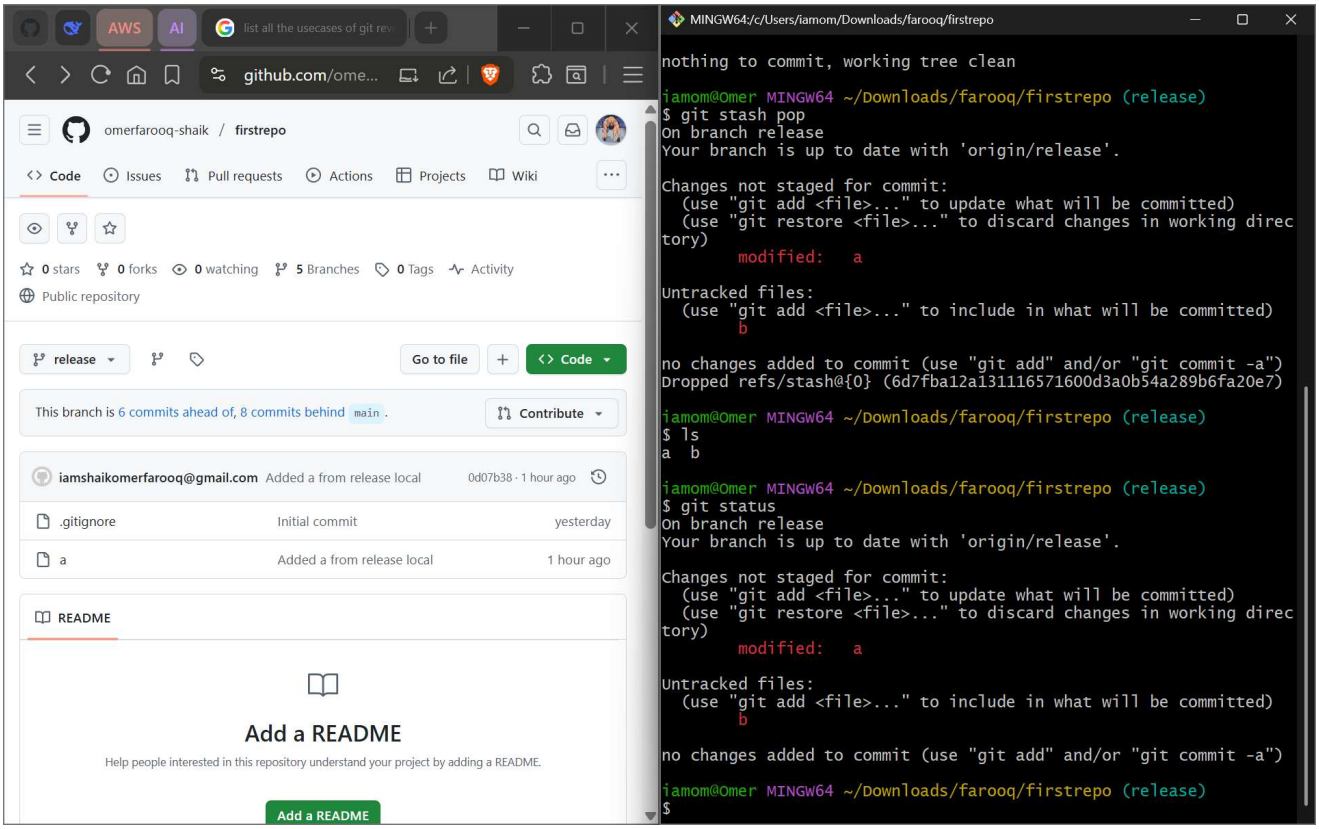
6. To do that, use the following command:
$ git stash -a (a means all)
Note: To stash, the "git stash" command can also be used, but it only stashes files that were never committed before. There's another command, "git stash -S", which is used to stash files from the staging area.

7. Most important thing to notice here is that file "a", which was committed and pushed to the remote and later modified, is visible in the working directory, while the file "b", which was never staged, committed, or pushed, is not moved to the stash.

8. The task is successfully executed, and the result can be seen in the above attached image.

16) Undo the stash file and start working on that again.

Objective:



Procedure:

1. Check the above task for detailed information about how to stash a file.

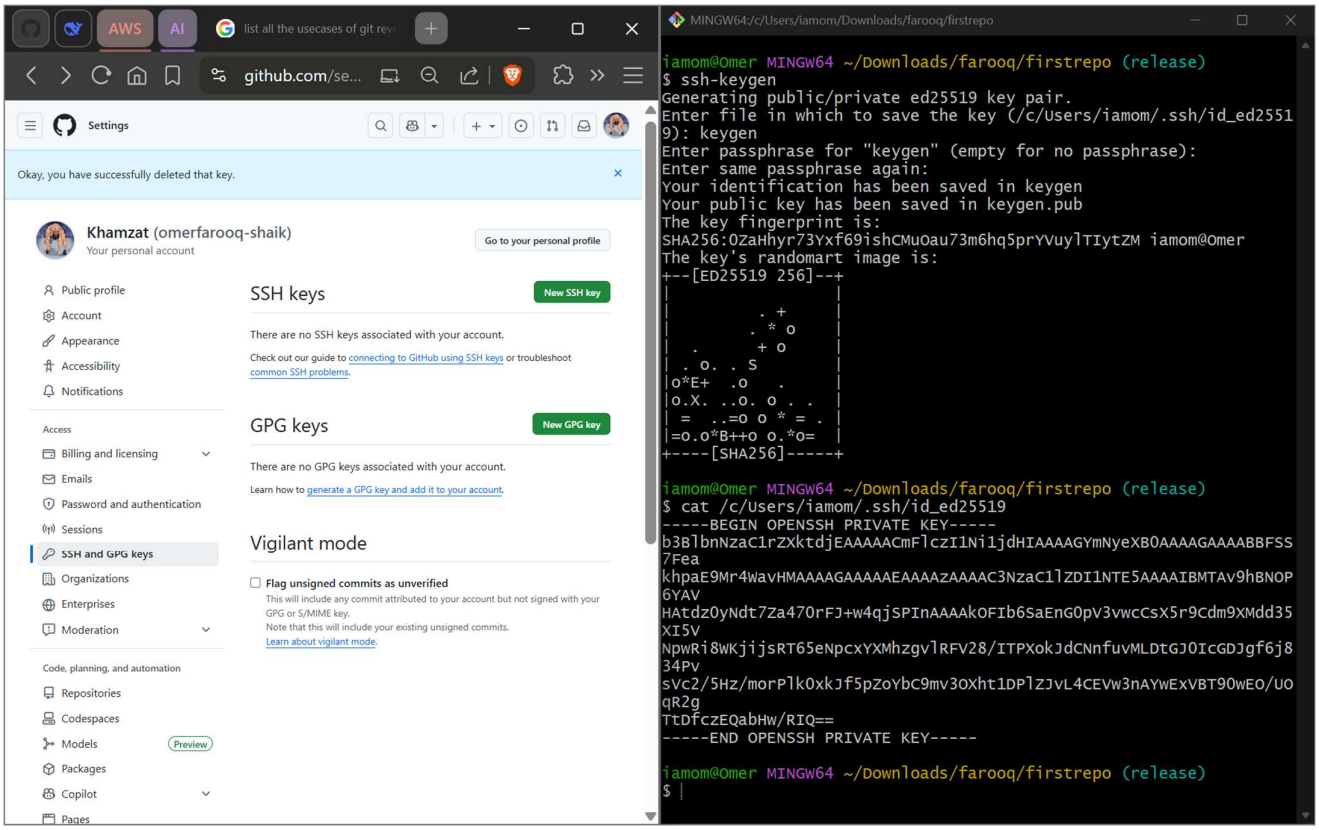
2. To unstash, use the following command:
$ git stash pop

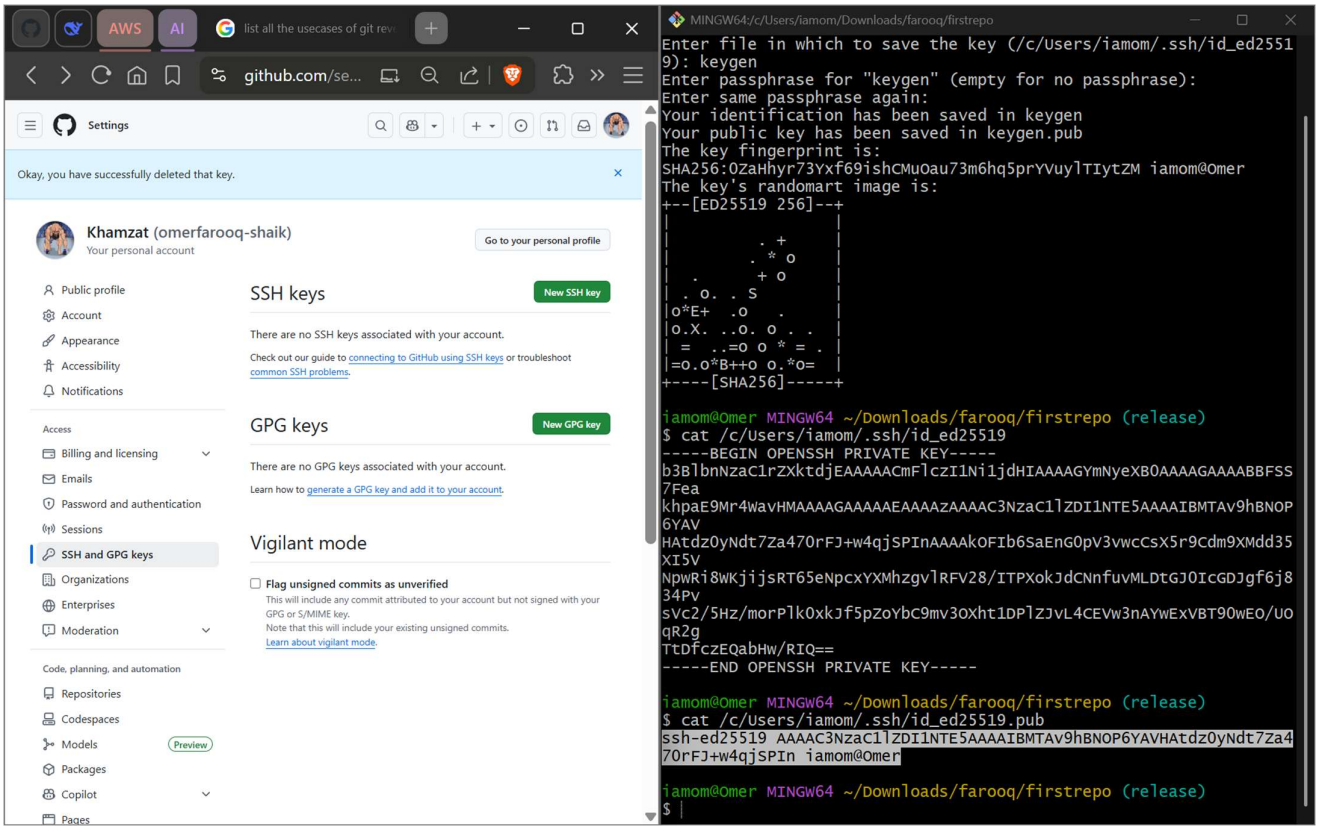
3. All the files that were stashed will be unstashed, and the files that were moved will now be visible in the working directory.

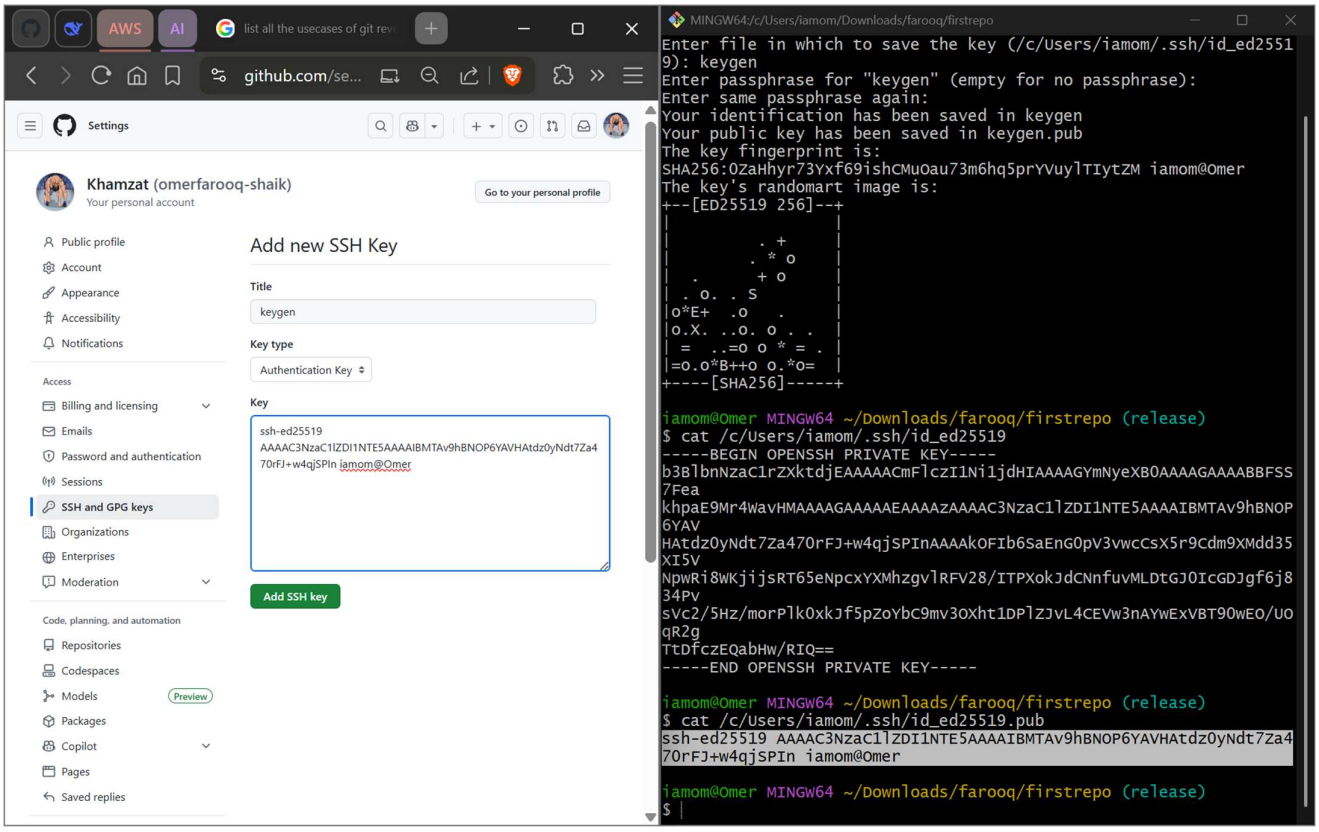
4. Check the above image for reference.

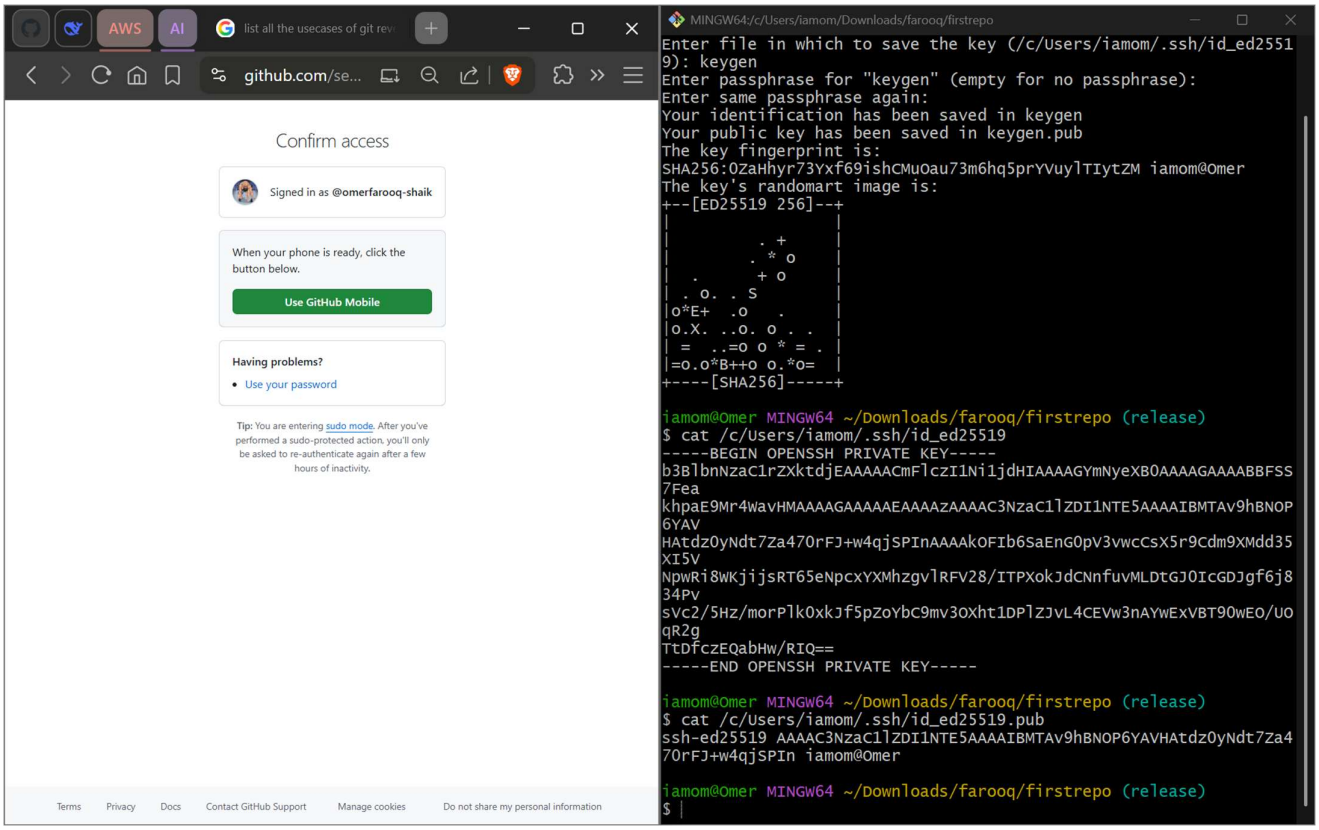
17) Generate an SSH key and configure it in GitHub.

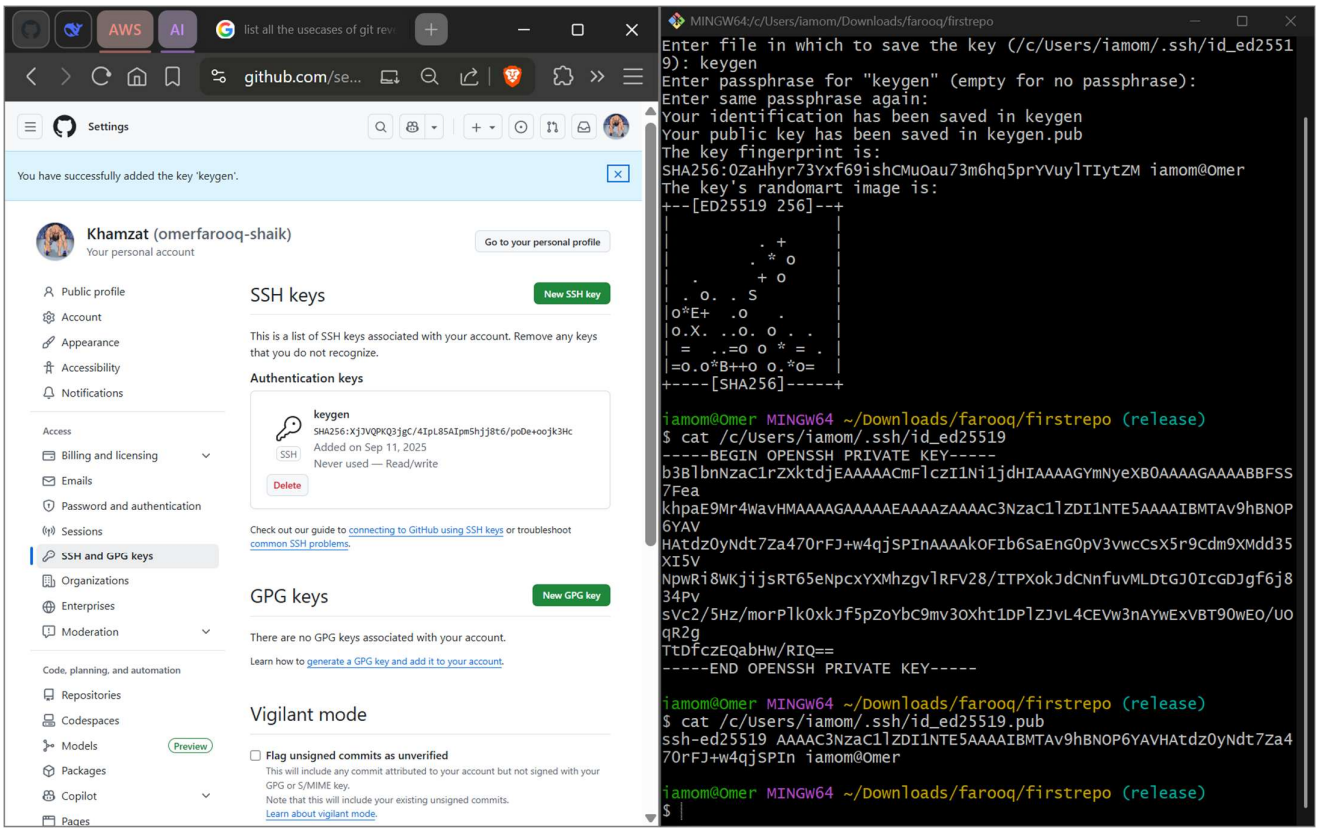
Objective:






github.com/se...
Settings
Khamzat (omerfarooq-shaik)
Your personal account
Go to your personal profile
Public profile
Account
Appearance
Accessibility
Notifications
Access
Billing and licensing
Emails
Password and authentication
Sessions
SSH and GPG keys
Organizations
Enterprises
Moderation
Code, planning, and automation
Repositories
Codespaces
Models
Preview
Packages
Copilot
Pages
Saved replies
Add new SSH Key
Title
keygen
Key type
Authentication Key
Key
ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAAIBMTAv9hBNOP6YAVHAtdz0yNdt7Za4
70rFJ+w4qjSPIn iamom@Omer
Add SSH key
MINGW64:/c/Users/iamom/Downloads/farooq/firstrepo
Enter file in which to save the key (/c/Users/iamom/.ssh/id_ed2551
9): keygen
Enter passphrase for "keygen" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in keygen
Your public key has been saved in keygen.pub
The key fingerprint is:
SHA256:0ZaHhyr73Yxf69ishCMuOau73m6hq5prYVuylTIytZM iamom@Omer
The key's randomart image is:
+--[ED25519 256]--+
+----[SHA256]-----+
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ cat /c/Users/iamom/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABBFSS
7Fea
khpaE9Mr4WavHMAAAAGAAAAAEAAAAzAAAAC3NzaC1lZDI1NTE5AAAAIBMTAv9hBNOP
6YAV
HAtdz0yNdt7Za47OrFJ+w4qjSPInAAAAkOFIb6SaEnGOpV3vwcCsX5r9Cdm9XMdd35
XI5V
NpwRi8WKjijsRT65eNpcxYXMhzgvlRFV28/ITPXokJdCNnfuvMLDtGJOIcGDJgf6j8
34Pv
sVc2/5Hz/morPlk0xkJf5pZoYbC9mv3OXht1DPlZJvL4CEVw3nAYwExVBT9OwEO/UO
qR2g
TtDfczEQabHw/RIQ==
-----END OPENSSH PRIVATE KEY-----
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ cat /c/Users/iamom/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBMTAv9hBNOP6YAVHAtdz0yNdt7Za4
70rFJ+w4qjSPIn iamom@Omer
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$


github.com/se...
Confirm access
Signed in as @omerfarooq-shaik
When your phone is ready, click the button below.
Use GitHub Mobile
Having problems?
Use your password
Tip: You are entering sudo mode. After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.
Terms
Privacy
Docs
Contact GitHub Support
Manage cookies
Do not share my personal information
MINGW64:/c/Users/iamom/Downloads/farooq/firstrepo
Enter file in which to save the key (/c/Users/iamom/.ssh/id_ed2551
9): keygen
Enter passphrase for "keygen" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in keygen
Your public key has been saved in keygen.pub
The key fingerprint is:
SHA256:0ZaHhyr73Yxf69ishCMuOau73m6hq5prYVuylTIytZM iamom@Omer
The key's randomart image is:
+--[ED25519 256]--+
+----[SHA256]-----+
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ cat /c/Users/iamom/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAACmFlczI1Ni1jdHIAAAAGYmNyeXB0AAAAGAAAABBFSS
7Fea
khpaE9Mr4WavHMAAAAGAAAAAEAAAAzAAAAC3NzaC1lZDI1NTE5AAAAIBMTAv9hBNOP
6YAV
HAtdz0yNdt7Za47OrFJ+w4qjSPInAAAAkOFIb6SaEnGOpV3vwcCsX5r9Cdm9XMdd35
XI5V
NpwRi8WKjijsRT65eNpcxYXMhzgvlRFV28/ITPXokJdCNnfuvMLDtGJOIcGDJgf6j8
34Pv
sVc2/5Hz/morPlk0xkJf5pZoYbC9mv3OXht1DPlZJvL4CEVw3nAYwExVBT9OwEO/UO
qR2g
TtDfczEQabHw/RIQ==
-----END OPENSSH PRIVATE KEY-----
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ cat /c/Users/iamom/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBMTAv9hBNOP6YAVHAtdz0yNdt7Za4
70rFJ+w4qjSPIn iamom@Omer
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$



Procedure:

1. Generate a key from the local machine. Use:
$ ssh-keygen

2. Follow the on-screen steps, like giving a name to the file and fingerprint

3. It creates a public key and a private key. We can choose both. In my task, I have taken a public key.

4. To open the key file, check the generated location of the public key and open it.
$ cat <path_to_file>

5. Copy the key. Open GitHub on a browser and log in to your account.

6. Go to Settings -> SSH and GPG keys. Check for the 'Authentication keys' tab.

7. Click on the 'New SSH Key' button and paste the key, which was copied from the local.

8. Click on 'Add SSH key'. If you have connected mobile authentication to your GitHub account, it will ask for the authentication, as in my case.

9. Follow the authentication process and done. We have successfully generated and add key to the remote.

18) Configure webhooks to GitHub.

Objective: NA

Procedure: NA

19) Basic understanding of the .git file.

Objective: To explain the '.git' file with its contents.

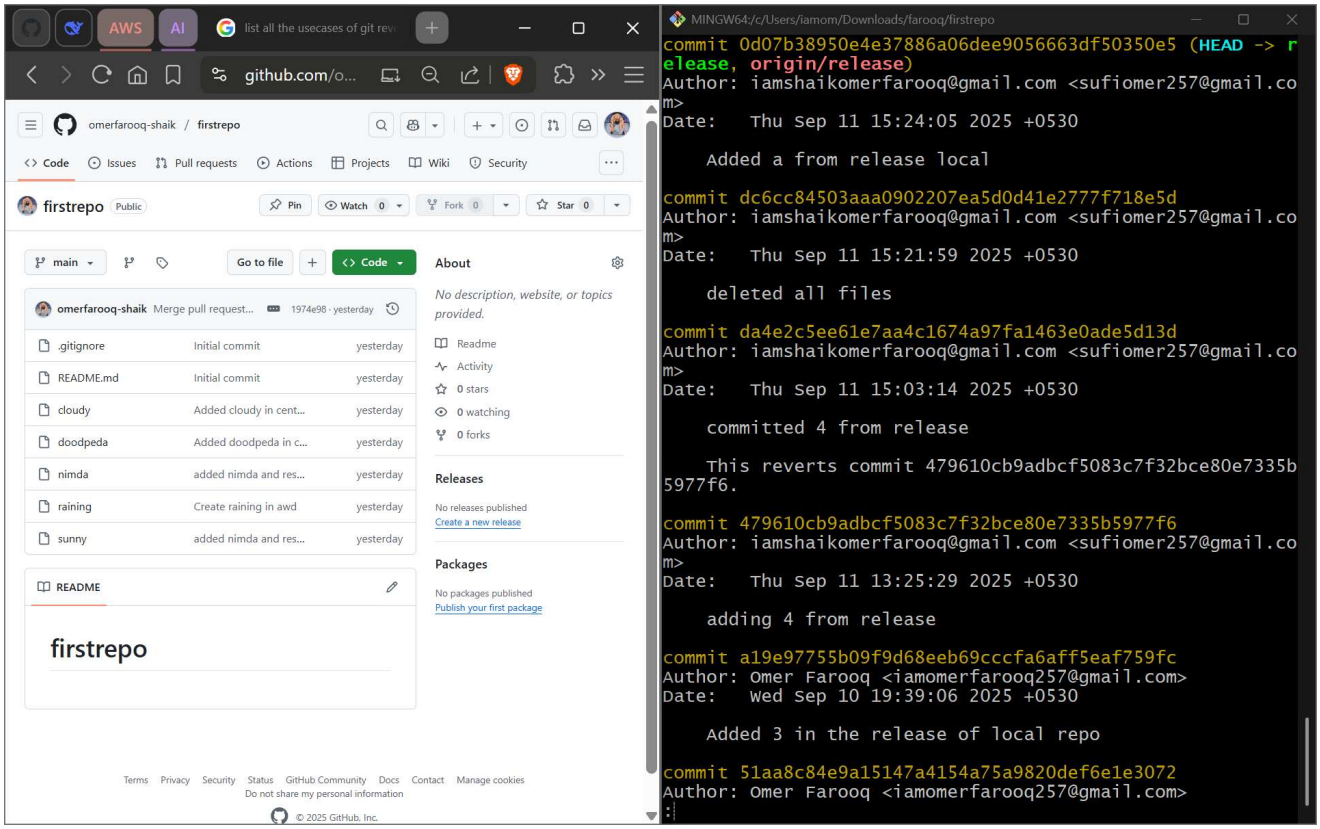
Explanation:

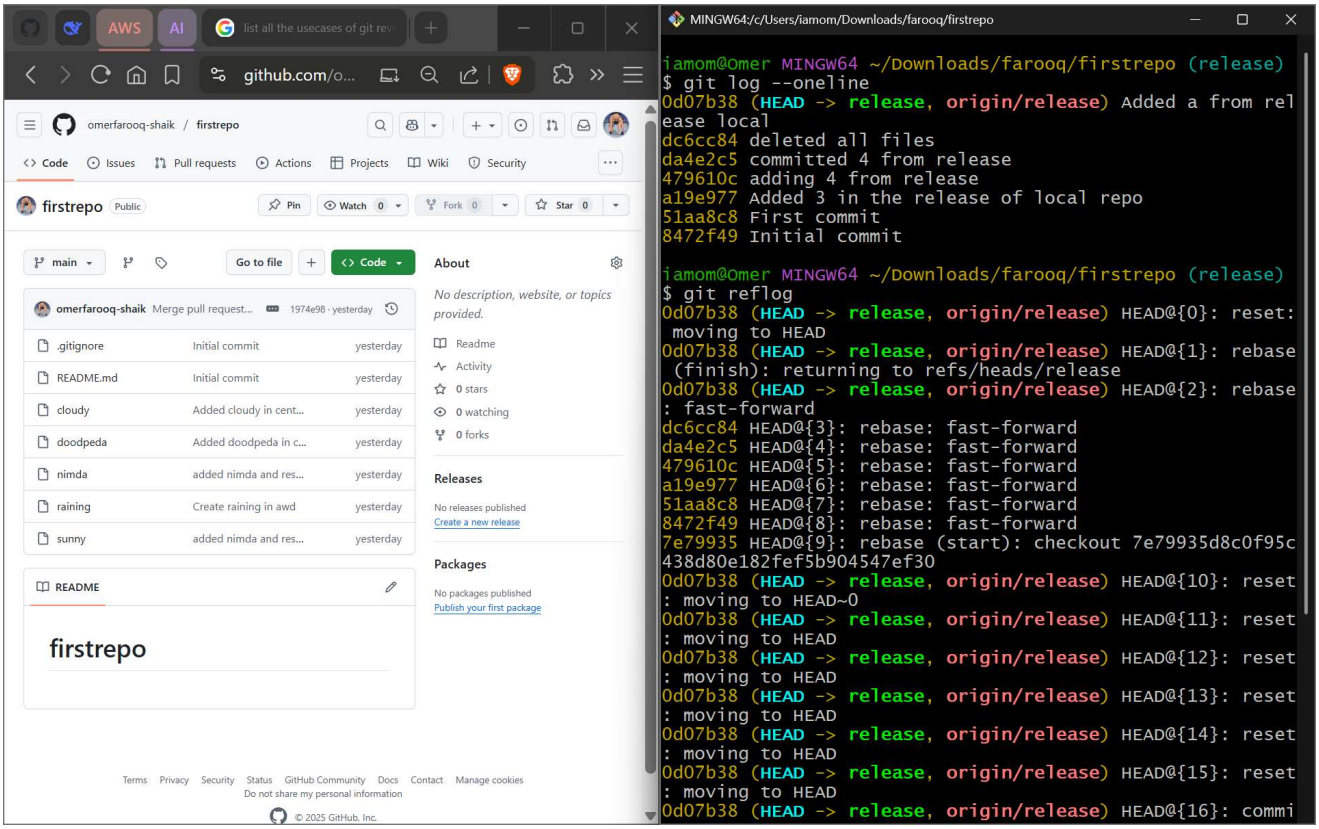
The .git folder contains Git's metadata and object database.
Key Files:

- HEAD: Points to the current branch.
- config: Repository-specific settings.
- objects/: Stores all commits, files, and trees.
- refs/heads/: Contains pointers to commits for each branch.

20) Check all the logs of git.

Objective:





Procedure:

1. The ‘git log’ command displays the history of committed changes in a Git repository.

2. By default, it shows commits in reverse chronological order, meaning the most recent commits appear first.

3. For each commit, git log typically provides the following information:

- Commit ID (SHA-1 checksum): A unique identifier for the commit.
- Author: The name and email address of the person who made the commit.
- Date: The date and time the commit was made.
- Commit Message: A description provided by the author explaining the changes made in that commit.

4. The git log command can be customized and filtered using various options to display specific information, limit the number of commits shown, or format the output differently.

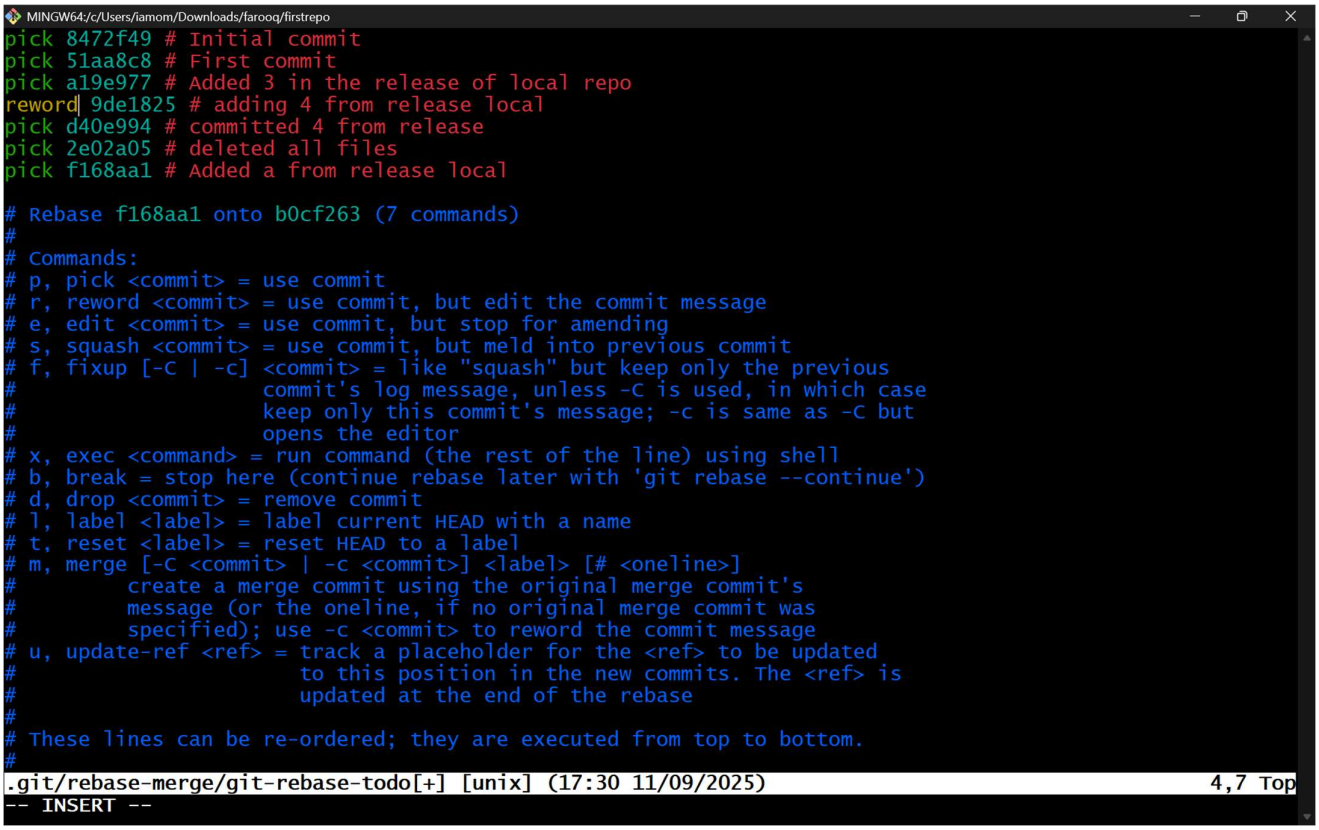
5. For instance, you can use options like filter by author, date, or specific files.

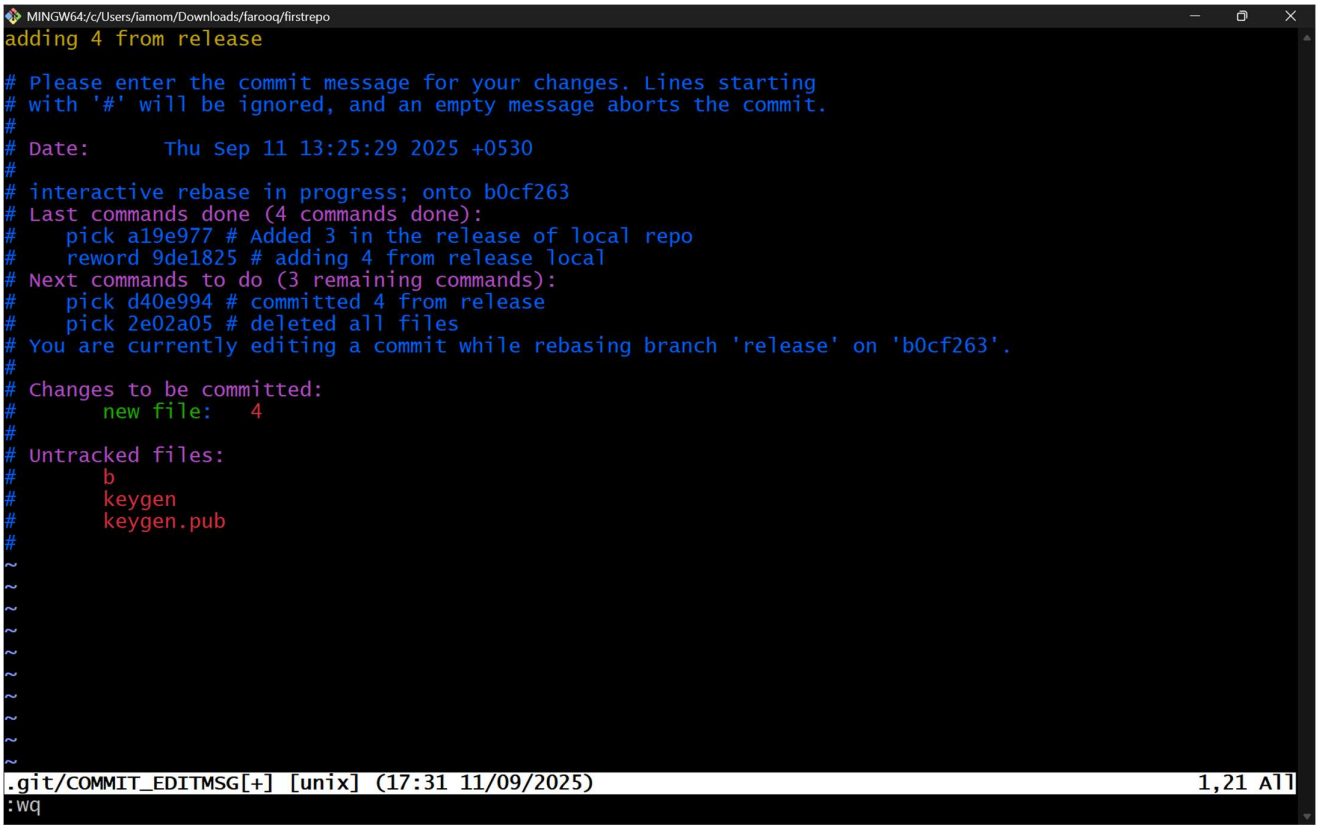
6. Below are some of the ways you can check logs:
$ “git log –oneline” for a concise view
$ “git log –graph” to visualize the commit history as a graph

7. Check the attached images for reference.

21) Rename the commit message.

Objective: To rename the commit message of any commit, either present or past commit.





```
[detached HEAD 2b42b31] adding 4 from release
 Date: Thu Sep 11 13:25:29 2025 +0530
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 4
Successfully rebased and updated refs/heads/release.

iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ git log --oneline
75fbd58 (HEAD -> release) Added a from release local
eb02359 deleted all files
6931453 committed 4 from release
2b42b31 adding 4 from release
a19e977 Added 3 in the release of local repo
51aa8c8 First commit
8472f49 Initial commit

iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$
```

Procedure:

1. First, find the hash/ID of the commit you want to rename using git log.
2. Start an interactive rebase with the command:
   $ git rebase -i <commit_ID>.
3. In the editor that opens, find the line corresponding to the commit you want to rename and change the word “pick” to “reword” or “r”.
4. Save and close the editor.
5. A new editor session will open for the commit you marked with reword.
6. Edit the commit message and save and close the file.
7. After the rebase completes, check the changes with:
   $ git log --oneline – The changes can be seen in the window.
8. Push the changes to remote.
9. Results can be seen in the above attached image.

22) Merge multiple commits into a single commit.

Objective:

```
[detached HEAD 08e3688] Added a and b after failing rebase in first attempt
 Date: Thu Sep 11 18:04:58 2025 +0530
 3 files changed, 5 insertions(+), 1 deletion(-)
 create mode 100644 a
 create mode 100644 b
Successfully rebased and updated refs/heads/release.

iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ git log --oneline
08e3688 (HEAD -> release) Added a and b after failing rebase in first attempt
ababe36 deleted all files
ab24ebe Added 3 in the release of local repo
51aa8c8 First commit
8472f49 Initial commit

iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ ls
a  b  keygen  keygen.pub

iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$
```

Procedure:

1. To merge multiple commits into single commit, use the below steps:
$ git rebase -i –root

2. In the editor that opens, find the line corresponding to the commits you want to merge and change the word “pick” to “squash”.

3. Save and close the editor.

4. A new editor session will open for the commits you marked with squash.

5. Edit the first commit message and save and close the file.
Note: If we have a total of 10 commits, and want to merge the top 4, the 4 commits will be merged to the next commit, i.e., the 5^{th} commit.

6. After the rebase completes, check the changes with:
$ git log --oneline – The changes can be seen in the window.

7. Push the changes to remote.