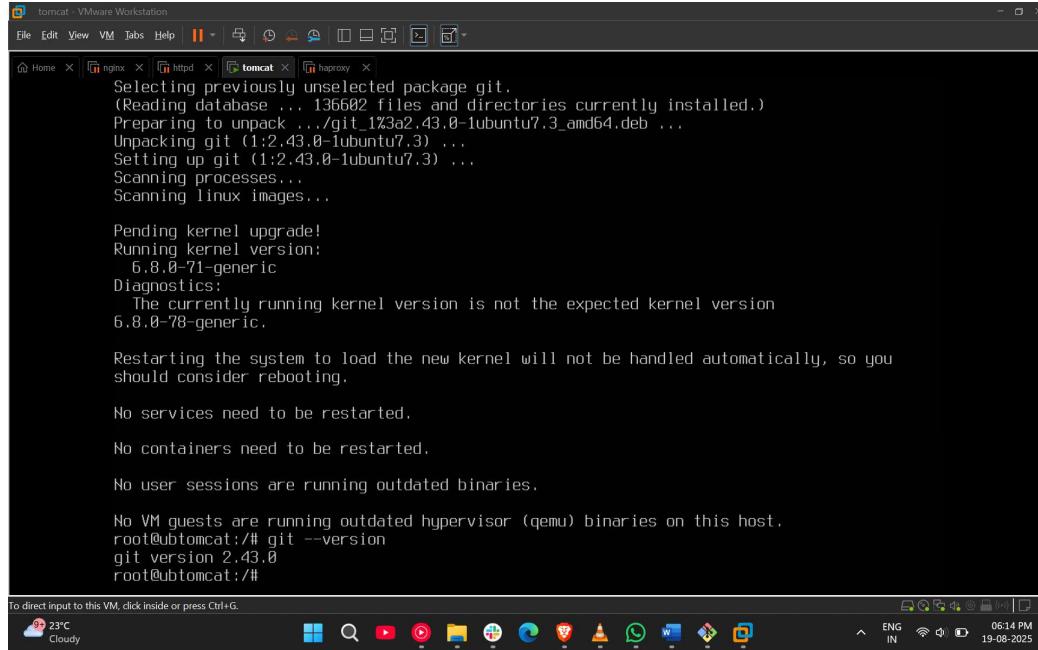


Tasks on Git and GitHub

1) Install git on the local machine.

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



```
Selecting previously unselected package git.
(Reading database ... 136602 files and directories currently installed.)
Preparing to unpack .../git_1%3d2.43.0-1ubuntu7.3_amd64.deb ...
Unpacking git (1:2.43.0-1ubuntu7.3) ...
Setting up git (1:2.43.0-1ubuntu7.3) ...
Scanning processes...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.8.0-71-generic
Diagnostics:
  The currently running kernel version is not the expected kernel version
  6.8.0-78-generic.

Restarting the system to load the new kernel will not be handled automatically, so you
should consider rebooting.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ubtomcat:/# git --version
git version 2.43.0
root@ubtomcat:/#
```



Procedure:

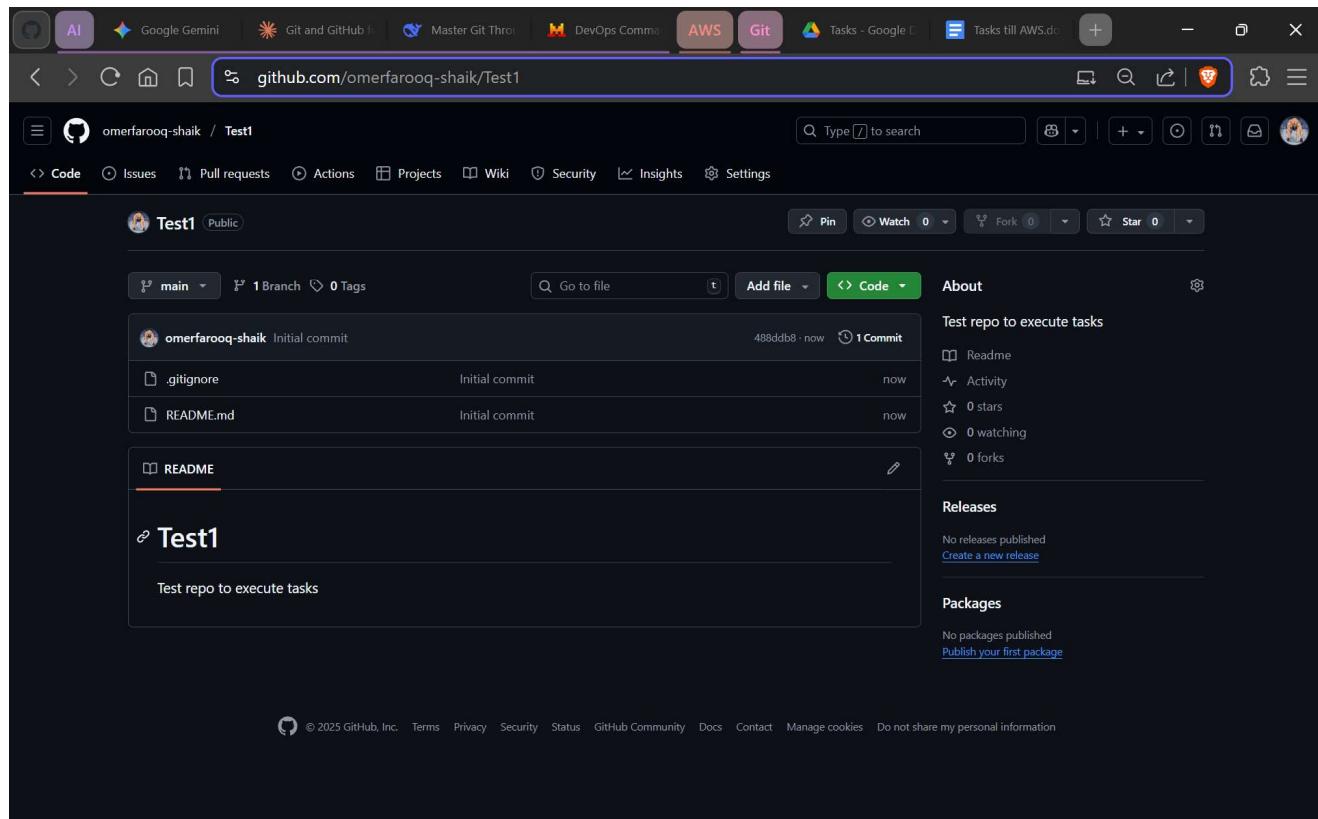
1. Steps to install Git in Windows:
 - a) Open a web browser and search for “Git for Windows.”
 - b) Select the Windows version of Git and download it.
 - c) Open the downloaded file and click on install, and follow the installation process.
 - d) Once Git is installed, click on finish to close the installation 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

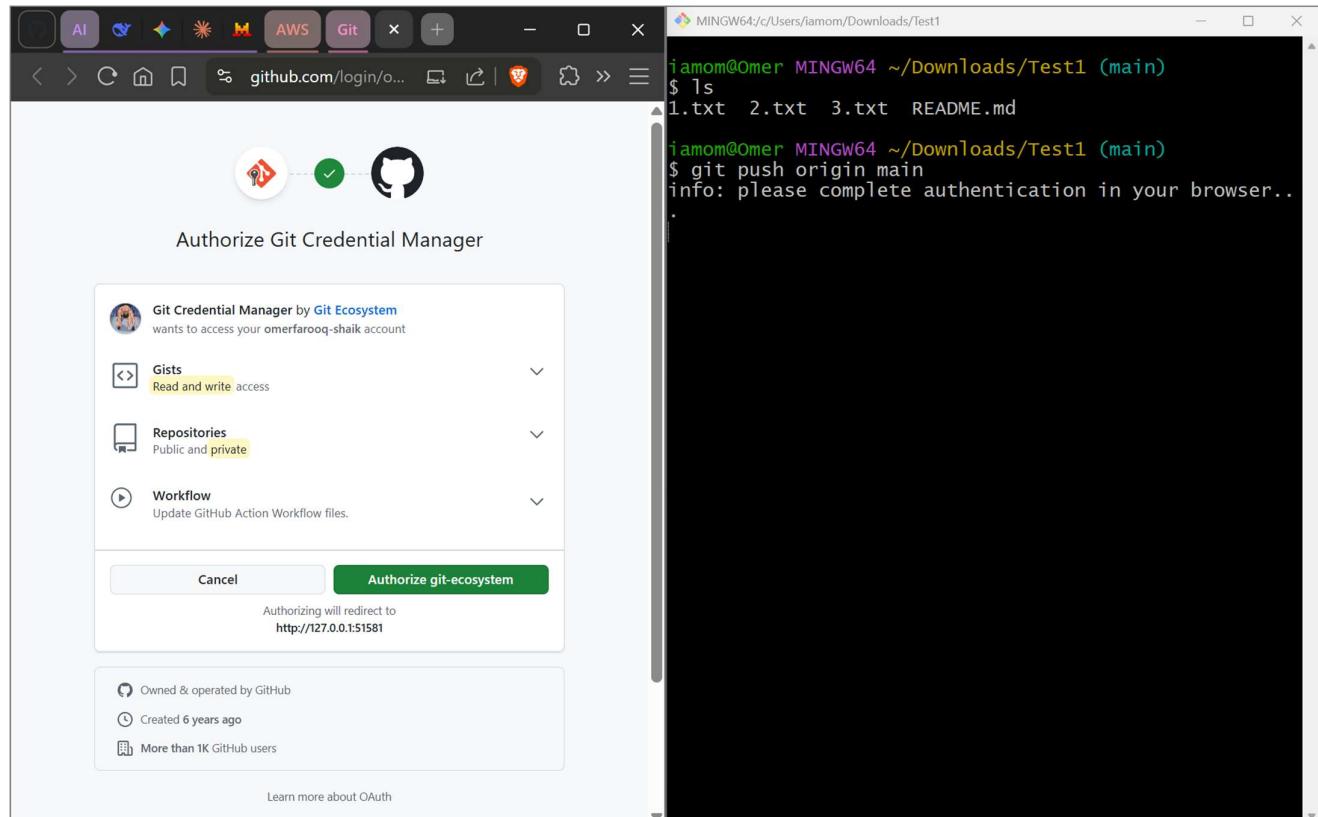
iamom@Omer MINGW64 ~/Downloads/Test1 (main)
$ git status
On branch main
Your branch is ahead of 'origin/main' by 1 commit.
  (use "git push" to publish your local commits)

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.



The screenshot shows a split-screen interface. On the left is a GitHub repository page for 'Test1'. It displays basic repository statistics: 0 stars, 0 forks, 0 watching, 1 Branch, 0 Tags, and Activity. Below this is a list of commits:

- Added 3 files (by omerfarooq-shaik, 4 days ago)
- .gitignore (Initial commit, 4 days ago)
- 1.txt (Added 3 files, 4 days ago)
- 2.txt (Added 3 files, 4 days ago)
- 3.txt (Added 3 files, 4 days ago)
- README.md (Initial commit, 4 days ago)

On the right is a terminal window titled 'MINGW64/c/Users/iamom/Downloads/Test1'. It shows the command \$ git push origin main being run, followed by a message: 'info: please complete authentication in your browser..'. The terminal then continues with object enumeration, counting, compression, and writing processes, concluding with a total of 3 objects (delta 0), reused 0 (delta 0), and pack-reused 0 (from 0). The final output is 'To https://github.com/omerfarooq-shaik/Test1.git 488ddb8..4e1d1b2 main -> main'.

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.

The screenshot shows a browser window with a tab titled "Git Cheat Sheet". The page contains a table of Git commands under the "GIT BASICS" section. To the right of the table is a terminal window showing the execution of several Git commands:

```
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ ls
1 2 README.md

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ git log --oneline
51aa8c8 (HEAD -> main) First commit
8472f49 (origin/main, origin/HEAD) Initial commit

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ git branch
* main

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ git branch release

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ git push origin release
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 308 bytes | 308.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
remote:
remote: Create a pull request for 'release' on GitHub by visiting:
remote: https://github.com/omerfarooq-shaik/firstrepo/pull/new/release
remote:
To https://github.com/omerfarooq-shaik/firstrepo.git
 * [new branch] release -> release

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

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.

The screenshot shows the GitHub 'Branches' page. A modal window titled 'Create a branch' is open, prompting for a 'New branch name' (set to 'aws') and a 'Source' (set to 'main'). Below the modal, two branches are listed: 'devops' and 'release'. At the top right of the main area, there is a green button labeled 'New branch'.

The screenshot shows a GitHub repository page for 'omerfaroq-shaik / firstrepo'. The 'aws' branch is selected. On the right, a terminal window displays the command-line process of cloning the 'aws' branch from the remote repository to a local machine. The terminal output includes:

```
To https://github.com/omerfaroq-shaik/firstrepo.git
* [new branch] devops -> devops
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (release)
$ 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)

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ git fetch origin aws
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1019 bytes | 127.00 KiB/s, done.
From https://github.com/omerfaroq-shaik/firstrepo
 * branch      aws      -> FETCH_HEAD
 * [new branch] aws      -> origin/aws

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ git checkout aws
branch 'aws' set up to track 'origin/aws'.
Switched to a new branch 'aws'

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (aws)
$ ls
README.md raining

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (aws)
$ cat raining
Hello I'm under the water, here too much raining.

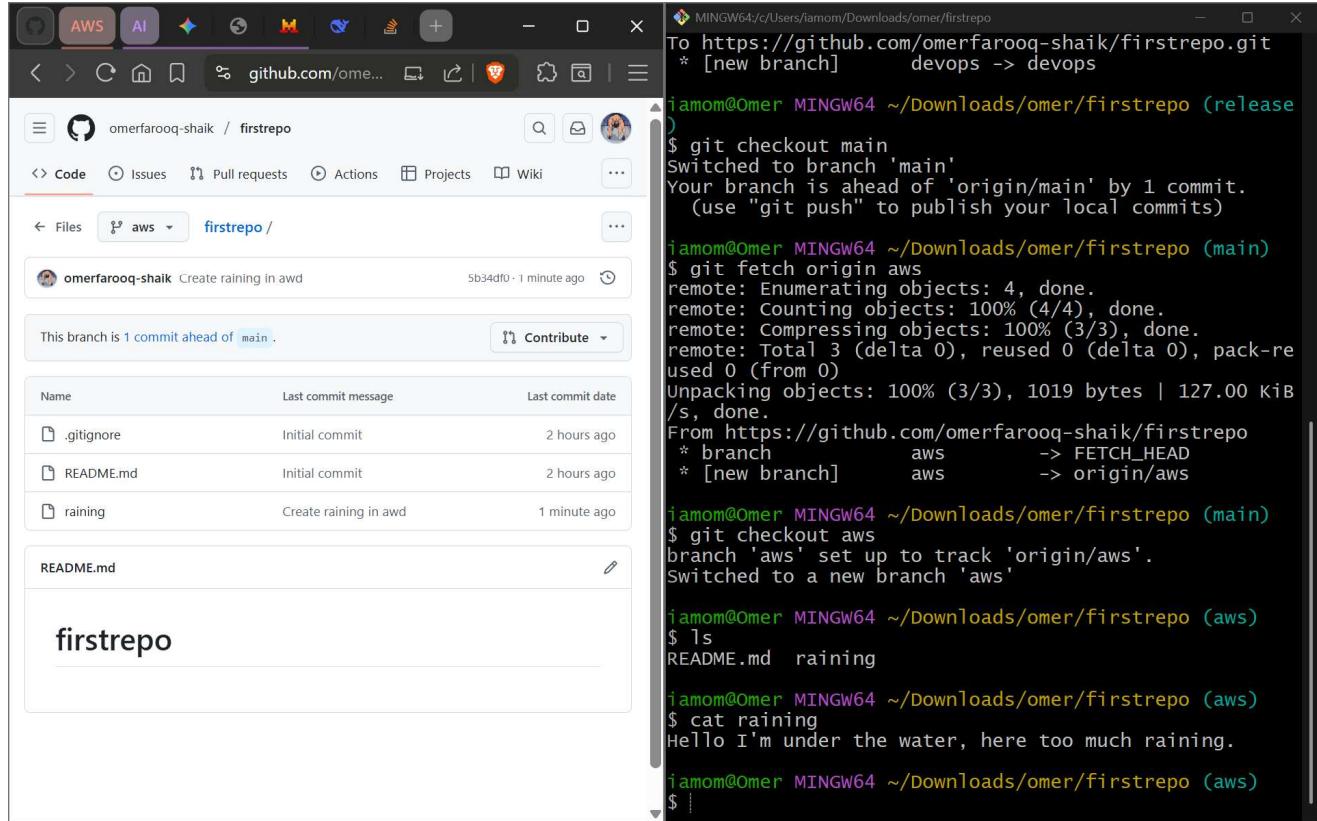
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (aws)
$ |
```

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.



The screenshot shows a split-screen interface. On the left is a browser window displaying the GitHub repository 'omerfarooq-shaik / firstrepo'. The repository has one branch, 'main', and three commits. The most recent commit is 'Create raining in awd' by 'omerfarooq-shaik' made 1 minute ago. On the right is a terminal window titled 'MINGW64/c/Users/iamom/Downloads/omer/firstrepo'. The terminal output shows the user's actions:

```
To https://github.com/omerfarooq-shaik/firstrepo.git
 * [new branch] devops -> devops

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

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ git fetch origin aws
remote: Enumerating objects: 4, done.
remote: Counting objects: 100% (4/4), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 1019 bytes | 127.00 KiB/s, done.
From https://github.com/omerfarooq-shaik/firstrepo
 * branch      aws      -> FETCH_HEAD
 * [new branch] aws      -> origin/aws

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ git checkout aws
branch 'aws' set up to track 'origin/aws'.
Switched to a new branch 'aws'

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (aws)
$ ls
README.md  raining

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (aws)
$ cat raining
Hello I'm under the water, here too much raining.

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (aws)
$
```

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:

Choose different branches or forks above to discuss and review changes. [Learn about pull requests](#)

[Create pull request](#)



Compare and review just about anything

Branches, tags, commit ranges, and time ranges. In the same repository and across forks.

Example comparisons	
 aws	54 minutes ago
 release	2 hours ago
 devops	2 hours ago
 main@{1day}...main	24 hours ago

Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#) or [learn more about diff comparisons](#).

Discuss and review the changes in this comparison with others. [Learn about pull requests](#)

[Create pull request](#)

• 7 commits 5 files changed 2 contributors

Commits on Sep 10, 2025

[Create raining in awd](#) Verified 5b34df0

 omerfarooq-shaik authored 1 hour ago

[Added sunny in aws](#) 0832e73 (diff)

 omerfarooq-shaik committed 1 hour ago

[Open a pull request](#)

Create a new pull request by comparing changes across two branches. If you need to, you can also compare across forks. Learn more about diff comparisons here.

The screenshot shows a GitHub pull request interface. At the top, there are buttons for 'base: main' and 'compare: aws'. The title 'Merging aws to main' is displayed in a large input field. Below it, a section for 'Add a description' includes tabs for 'Write' (selected) and 'Preview', and a rich text editor with various formatting tools. A placeholder text 'Add your description here...' is present. To the right, there are sections for 'Reviewers' (no reviews), 'Assignees' (no one, with a note to 'assign yourself'), 'Labels' (none yet), 'Projects' (none yet), 'Milestone' (no milestone), and 'Development' (using closing keywords to automatically close issues). At the bottom, a green button says 'Create pull request'.

Merging aws to main #1

omerfarooq-shaik wants to merge 7 commits into `main` from `aws`

- `Merge branch 'aws' of https://github.com/omerfarooq-shaik/firstrepo i... 7cd7bf2`
- `Added doodpeda in central aws 1d25de6`

No conflicts with base branch
Merging can be performed automatically.

Merge pull request You can also merge this with the command line. [View command line instructions.](#)

Merging aws to main #1

`Merged` omerfarooq-shaik merged 7 commits into `main` from `aws` now

Conversation 0 Commits 7 Checks 0 Files changed 5 +7 -0

omerkarooq-shaik commented 4 minutes ago
No description provided.

omerkarooq-shaik and others added 7 commits 1 hour ago

- `Create raining in awd` Verified 5b34df0
- `Added sunny in aws` Verified 0832e73
- `Added sunny in central aws` Verified f61411b
- `Added cloudy in central aws` Verified 648b900
- `added nimda and resolved sunny`
- `Merge branch 'aws' of https://github.com/omerfarooq-shaik/firstrepo i... 7cd7bf2` Verified 238166d
- `Added doodpeda in central aws` Verified 1d25de6

Reviewers: No reviews Still in progress? Convert to draft

Assignees: None yet Assign yourself

Labels: None yet

Projects: None yet

Milestone: None yet

Development: Successfully merging this pull request may close these

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).

The screenshot shows a browser window with the GitHub URL `github.com/omerfarooq-shaik/firstrepo`. The repository details are visible, including 0 stars, 0 forks, 0 watching, 4 branches, and 0 tags. A yellow banner at the top indicates "release had recent pushes 10 seconds ago". Below this, a message says "This branch is 2 commits ahead of, 8 commits behind main.". The commit history shows three commits from "omerfarooq-shaik": "Added 3 in the release of local repo" (a19e977), "Initial commit" (.gitignore), and "First commit" (1, 2, 3). The terminal session on the right shows the command-line steps to add, commit, and push the changes:

```
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (release)
$ ls
1 2 README.md

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (release)
$ echo "Hello" > 3

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (release)
$ git add 3
warning: in the working copy of '3', LF will be replaced by CRLF the next time Git touches it

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (release)
$ git status
On branch release
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file: 3

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (release)
$ git commit -m "Added 3 in the release of local repo"
[release a19e977] Added 3 in the release of local repo
1 file changed, 1 insertion(+)
create mode 100644 3

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (release)
$ git push origin release
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 16 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 281 bytes | 281.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1) completed with 1 local object.
To https://github.com/omerfarooq-shaik/firstrepo.git
  51aa8c8..a19e977  release -> release

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

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.

The screenshot shows a browser window with the GitHub URL <https://github.com/omerfaroq-shaik/firstrepo>. The repository page displays basic information like 0 stars, 0 forks, and 5 branches. A yellow banner at the top indicates a recent push. Below, a list of commits is shown, including a merge pull request and several commits from the 'main' branch. On the right, a terminal window titled 'MINGW64' shows the command-line process of cloning the 'aws' branch. It starts with navigating to the repository directory, listing files, and checking branches. Then, it runs 'git clone -b aws' followed by 'git branch -a' to show the cloned branch alongside the remote ones.

```
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo/firstrepo
$ ls
1 2 README.md firstrepo/
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ git branch
aws
devops
* main
release
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ git branch -a
aws
devops
* main
release
remotes/origin/HEAD -> origin/main
remotes/origin/aws
remotes/origin/devops
remotes/origin/main
remotes/origin/release
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo/firstrepo (version)
$ cd firstrepo/
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo/firstrepo (version)
$ git branch -a
* version
remotes/origin/version
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo/firstrepo (version)
$ |
```

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/omerfaroq-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.

The screenshot shows a browser window for GitHub with a repository named 'firstrepo'. The repository has 0 stars, 0 forks, 0 watching, 5 branches, and 0 tags. A recent push from 'aws' is visible. Below the repository details is a list of commits from 'shaik'. The terminal session on the right shows the following steps:

```
iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ ls
README.md  cloudy  doodpeda  nimda  raining  sunny

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ echo "shush! idhar passwords hai. nai aana idhar" > password.txt

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ echo "password.txt" >> .gitignore

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ tail -2 .gitignore
#Test results file
TestResults.xmlpassword.txt

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

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
    (use "git restore <file>..." to discard changes in working directory)
      modified:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    password.txt

no changes added to commit (use "git add" and/or "git commit -a")

iamom@Omer MINGW64 ~/Downloads/omer/firstrepo (main)
$ |
```

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.

The screenshot shows a ChatGPT interface with a terminal window. On the left, there's a sidebar with a 'ChatGPT' section and a 'Ask anything' input field. The main area has a title '13) Make a commit and reset without saving changes'. Below it, a terminal window shows the following commands and output:

```
echo "my-secret-passwords" > passwords.txt
echo "passwords.txt" >> .gitignore
```

Then, the user asks:

git reset --hard HEAD~1

ChatGPT responds:

👉 Removes last commit and your changes.

On the right, another terminal window shows the command history:

```
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ echo "This is 4" > 4
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ ls
1 2 3 4 README.md
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ git add 4
warning: in the working copy of '4', LF will be replaced by CRLF the next time Git touches it
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ git commit -m "Adding 4 from release"
[release 3d85bfef] Adding 4 from release
1 file changed, 1 insertion(+)
create mode 100644 4
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ git status
On branch release
Your branch is ahead of 'origin/release' by 1 commit.
  (use "git push" to publish your local commits)
nothing to commit, working tree clean
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ git log --oneline
3d85bfef (HEAD -> release) Adding 4 from release
a19e977 (origin/release) Added 3 in the release of local repo
51aa8c8 First commit
8472f49 Initial commit
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ git reset --hard a19e977
HEAD is now at a19e977 Added 3 in the release of local repo
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ ls
1 2 3 README.md
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ git log --oneline
a19e977 (HEAD -> release, origin/release) Added 3 in the release of local repo
51aa8c8 First commit
8472f49 Initial commit
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ |
```

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.

```
$ 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:

GitHub Repository Page (omerkarimfarooq / firstrepo):

- Code (selected)
- Issues
- Pull requests
- Actions
- Projects
- Wiki

Branch: release

This branch is 3 commits ahead of, 8 commits behind main.

Commit	Author	Message	Date
18d61a2	iamshaikomerfarooq@gmail.com	Adding 4 from release	2 hours ago
1		Initial commit	yesterday
2		First commit	yesterday
3		Added 3 in the release of local repo	19 hours ago
4		Adding 4 from release	2 hours ago
README.md		Initial commit	yesterday

Terminal Log (MINGW64 shell):

```
[release da4e2c5] committed 4 from release
1 file changed, 0 insertions(+), 0 deletions(-)
delete mode 100644 4
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (releas
$ ls
1 2 3 README.md
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (releas
$ git log --oneline
da4e2c5 (HEAD -> release) committed 4 from release
479610c 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 (releas
$
```

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.

The screenshot shows a GitHub repository page for 'fistrepo' and a terminal window on a Windows machine (MINGW64). In the GitHub interface, file 'a' is shown as modified and file 'b' as untracked. The terminal shows the following sequence of commands:

```
iamom@Omer MINGW64 ~/Downloads/farooq/fistrepo (release)
$ ls
a b

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

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   a

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        b

no changes added to commit (use "git add" and/or "git commit -a")

iamom@Omer MINGW64 ~/Downloads/farooq/fistrepo (release)
$ git stash -a
Saved working directory and index state WIP on release: 0d07b38 Added a from release local

iamom@Omer MINGW64 ~/Downloads/farooq/fistrepo (release)
$ ls
a

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

nothing to commit, working tree clean

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

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:

The screenshot shows a GitHub repository page for 'omerfarooq-shaik / firstrepo'. The repository is public, has 0 stars, 0 forks, 0 watching, 5 branches, and 0 tags. The 'release' branch is selected. The terminal window on the right shows the command \$ git stash pop being run in a MINGW64 shell. The output indicates there is nothing to commit, the working tree is clean, and the branch is up to date with 'origin/release'. It also shows changes not staged for commit (files 'a' and 'b') and untracked files ('a'). The terminal then lists the contents of the 'a' directory, which contains 'a' and 'b'. Finally, it shows the status after stash popping, where the branch is still up to date with 'origin/release'.

```
nothing to commit, working tree clean
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ git stash pop
On branch release
Your branch is up to date with 'origin/release'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   a

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        b

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (6d7fba12a13116571600d3a0b54a289b6fa20e7)

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

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

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   a

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        b

no changes added to commit (use "git add" and/or "git commit -a")

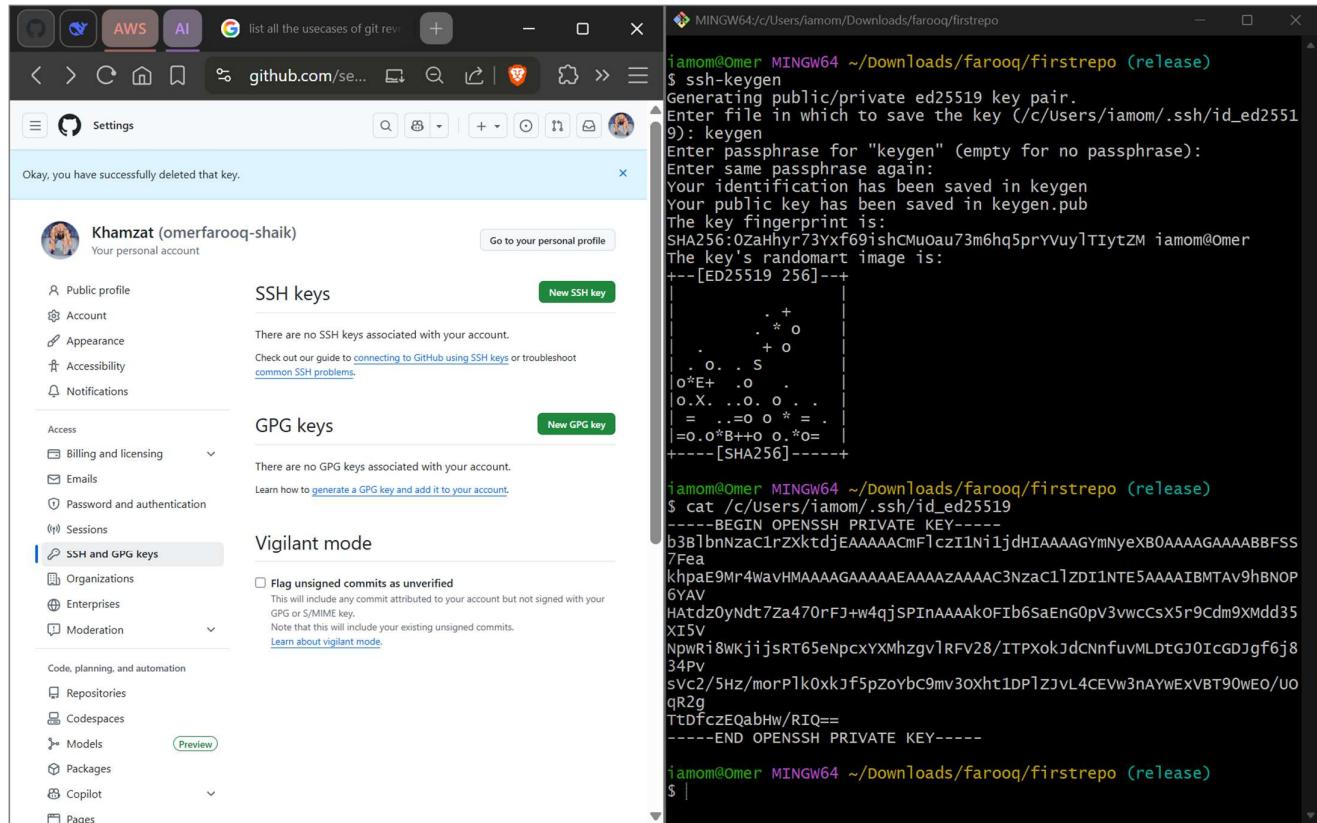
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$
```

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.

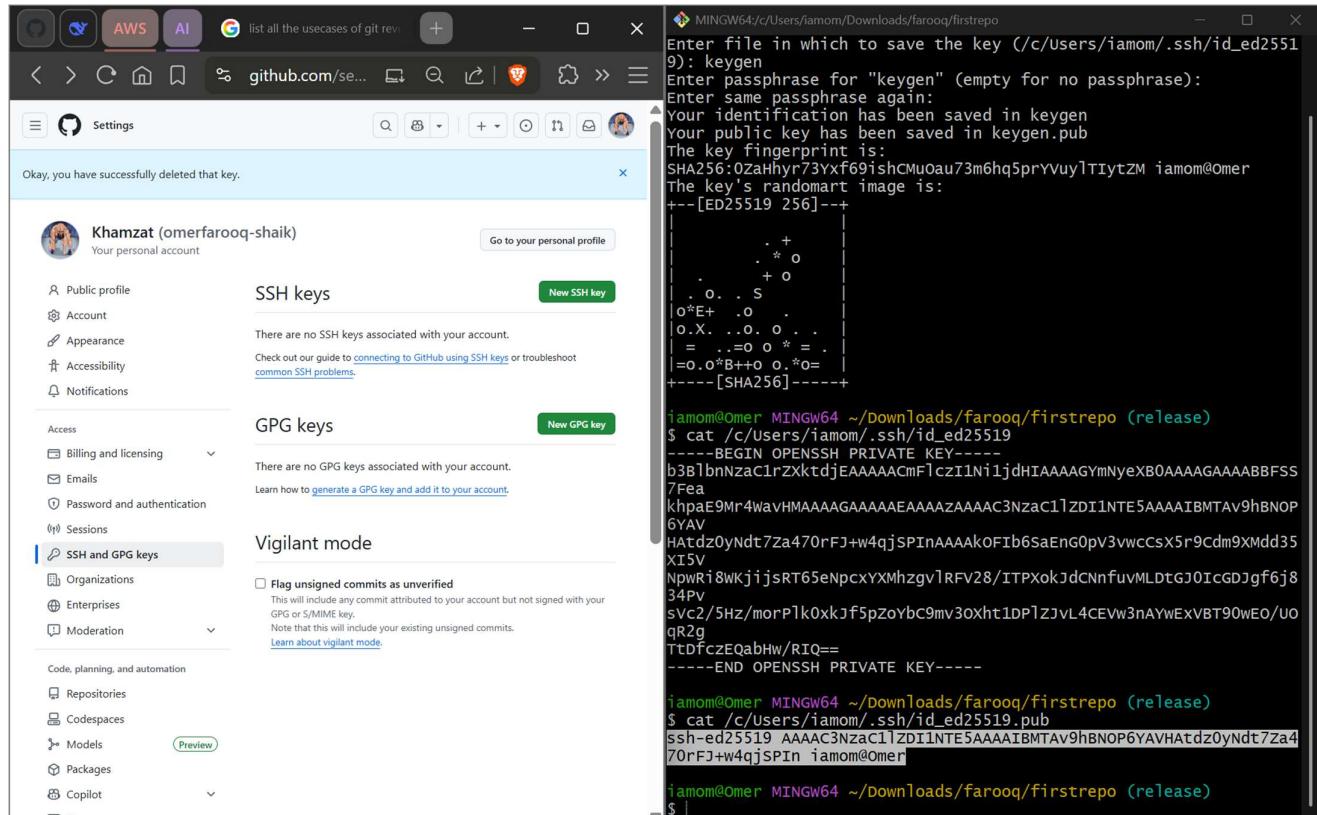
17) Generate an SSH key and configure it in GitHub.

Objective:



The screenshot shows a browser window for GitHub settings and a terminal window running on MINGW64. The GitHub settings page displays account information, SSH keys (none), GPG keys (none), and Vigilant mode options. The terminal window shows the process of generating an SSH key pair, including prompts for file save, passphrase, and key fingerprint, followed by the generated key content and its SHA256 hash.

```
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ ssh-keygen
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/iamom/.ssh/id_ed25519): 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:
SHA256:0ZaHyr73Yxf69ishCMQoau73m6hq5prYvuy1TiytzM iamom@Omer
The key's randomart image is:
+--[ED25519 256]--+
| . +
| . * o
| . + o
| . o . S
| o^E+ . o .
| o.X . . o . .
| = ..=o o * = .
|=o.o*B++o o.*o=
+---[SHA256]---+
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ cat /c/Users/iamom/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaci1rzXktdjEAAAACmF1czI1n1jdHIAAAAGYmNyexB0AAAAGAAAABBFSS
7Fea
khpaE9Mr4WavHMAAAAGAAAAAAEAAAAC3NzaC1lZDI1NTE5AAAIBMTAv9hBNOP
6YAV
HAtdz0yNdt7za470rFJ+w4qjSPInAAAakOFib6saEngOpV3vwccsX5r9cdm9Xmd35
XISV
NpwRi8WkjijRT65eNpcxyMhzgv1RFV28/ITPxokJdCNnfuvMLDtGJ0IcGDJgf6j8
34Pv
svC2/5Hz/morPlk0xkjf5pzoYbc9mv3oxht1Dp1zJvL4CEVw3nAYwExVBT90wEO/u
qr2g
TtDfczEqabHW/RIQ==
-----END OPENSSH PRIVATE KEY-----
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ |
```



The second screenshot shows the same GitHub settings and terminal window. The terminal now displays the configuration of the generated SSH key in GitHub, including the private key content and the public key being added to the GitHub SSH keys section.

```
Enter file in which to save the key (/c/Users/iamom/.ssh/id_ed25519): 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:
SHA256:0ZaHyr73Yxf69ishCMQoau73m6hq5prYvuy1TiytzM iamom@Omer
The key's randomart image is:
+--[ED25519 256]--+
| . +
| . * o
| . + o
| . o . S
| o^E+ . o .
| o.X . . o . .
| = ..=o o * = .
|=o.o*B++o o.*o=
+---[SHA256]---+
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ cat /c/Users/iamom/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaci1rzXktdjEAAAACmF1czI1n1jdHIAAAAGYmNyexB0AAAAGAAAABBFSS
7Fea
khpaE9Mr4WavHMAAAAGAAAAAAEAAAAC3NzaC1lZDI1NTE5AAAIBMTAv9hBNOP
6YAV
HAtdz0yNdt7za470rFJ+w4qjSPInAAAakOFib6saEngOpV3vwccsX5r9cdm9Xmd35
XISV
NpwRi8WkjijRT65eNpcxyMhzgv1RFV28/ITPxokJdCNnfuvMLDtGJ0IcGDJgf6j8
34Pv
svC2/5Hz/morPlk0xkjf5pzoYbc9mv3oxht1Dp1zJvL4CEVw3nAYwExVBT90wEO/u
qr2g
TtDfczEqabHW/RIQ==
-----END OPENSSH PRIVATE KEY-----
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAIBMTAv9hBNOP6YAVHAtdz0yNdt7za4
70rFJ+w4qjSPIn iamom@Omer
iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo (release)
$ |
```

Khamzat (omerfarooq-shaik)
Your personal account

Add new SSH Key

Title: keygen

Key type: Authentication Key

```
ssh-ed25519
AAAAC3nzaC1ZDI1NTE5AAAAIBMTAv9hBNOP6YAVHAtdz0yNdt7za4
70rFJ+w4qjSPIn iamom@Omer
```

Add SSH key

MINGW64: MINGW64 ~ /Downloads/farooq/firstrepo (release)

```
Enter file in which to save the key (/c/users/iamom/.ssh/id_ed25519)
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:
SHA256:0zaHyr73Yxf69ishCMuoau73m6hq5prYvuy1TIytZM iamom@Omer
The key's randomart image is:
+---[ED25519 256]---+
| . +
| . * o
| . + o
| . o . S
| o^E+ .o .
| o.X . o. o. .
| = ..=o o * = .
|=o.o*B++o o.*o=
+---[SHA256]---+
```

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

```
$ cat /c/Users/iamom/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaci1rzXktdjEAAAACmFlczI1N1jdHIAAAAGYmNyexB0AAAAAGAAAABFSS
7Fea
khpaE9Mr4wavHMAAAGAAAAAAZAAAC3NzaC1lZDI1NTE5AAAAIBMTAv9hBNOP6YAVHAtdz0yNdt7za4
6YAV
HAtdz0yNdt7za470rFJ+w4qjSPInAAAAKoFib6SaEng0pV3vwccsX5r9cdm9Xmd35
XISV
NpwRi8WkjijRT65eNpcxyMhzgv1RFV28/ITPxokJdCnfuvMLDtGJOICGDJgf6j8
34Pv
sVC2/5Hz/morPlk0xkjf5pzoYbc9mv3oxht1DpLzjvL4CEVw3nAYwExVBT90wEO/u
qr2g
TdfczEqabHw/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)

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.

MINGW64: MINGW64 ~ /Downloads/farooq/firstrepo (release)

```
Enter file in which to save the key (/c/users/iamom/.ssh/id_ed25519)
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:
SHA256:0zaHyr73Yxf69ishCMuoau73m6hq5prYvuy1TIytZM iamom@Omer
The key's randomart image is:
+---[ED25519 256]---+
| . +
| . * o
| . + o
| . o . S
| o^E+ .o .
| o.X . o. o. .
| = ..=o o * = .
|=o.o*B++o o.*o=
+---[SHA256]---+
```

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

```
$ cat /c/Users/iamom/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaci1rzXktdjEAAAACmFlczI1N1jdHIAAAAGYmNyexB0AAAAAGAAAABFSS
7Fea
khpaE9Mr4wavHMAAAGAAAAAAZAAAC3NzaC1lZDI1NTE5AAAAIBMTAv9hBNOP6YAVHAtdz0yNdt7za4
6YAV
HAtdz0yNdt7za470rFJ+w4qjSPInAAAAKoFib6SaEng0pV3vwccsX5r9cdm9Xmd35
XISV
NpwRi8WkjijRT65eNpcxyMhzgv1RFV28/ITPxokJdCnfuvMLDtGJOICGDJgf6j8
34Pv
sVC2/5Hz/morPlk0xkjf5pzoYbc9mv3oxht1DpLzjvL4CEVw3nAYwExVBT90wEO/u
qr2g
TdfczEqabHw/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:

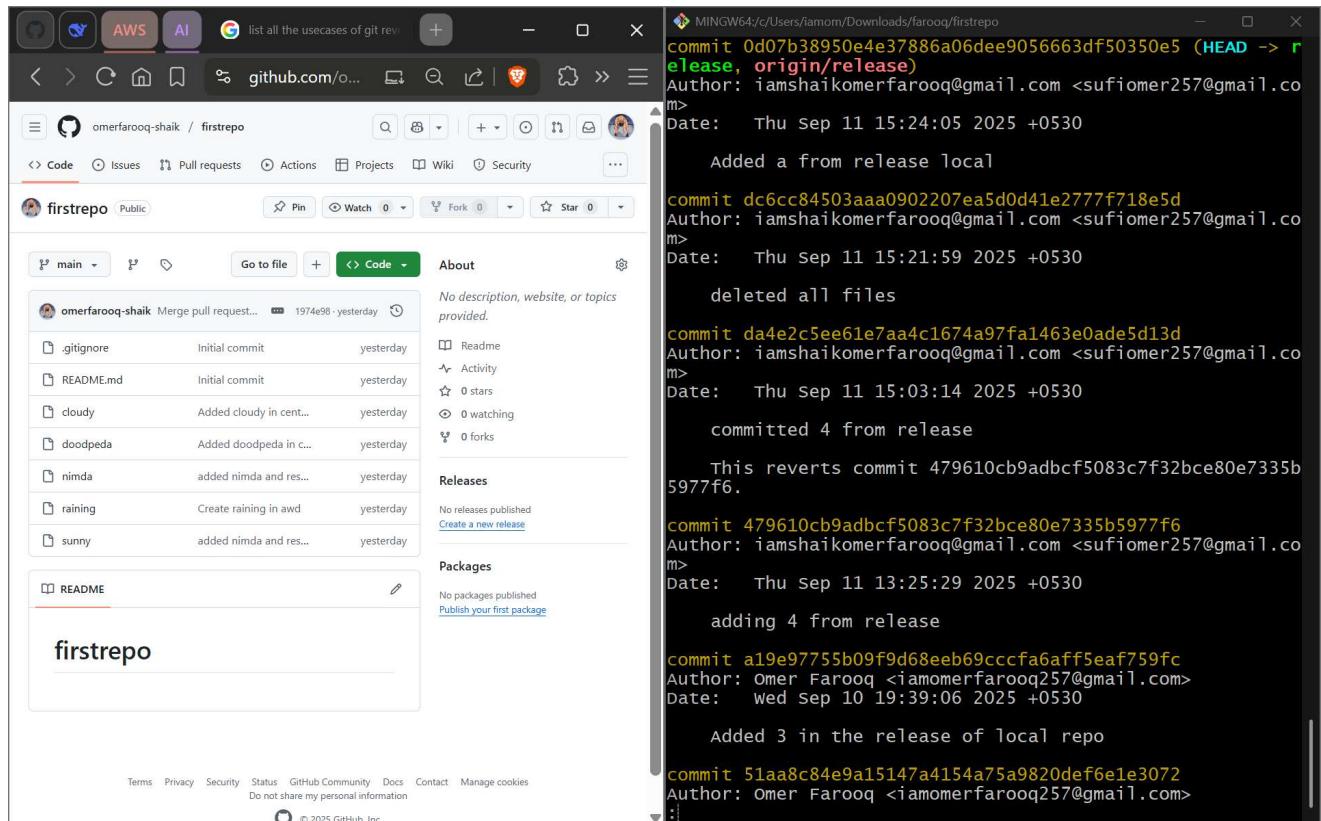
The .git folder contains Git's metadata and object database.

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:



The screenshot shows a dual-pane interface. On the left is a GitHub repository page for 'firstrepo' owned by 'omerfarooq-shaik'. It lists several commits and pull requests. On the right is a terminal window running on MINGW64, showing a log of Git commits. The commits are as follows:

```
MINGW64:/c/Users/iamom/Downloads/farooq/firstrepo
commit 0d07b38950e4e37886a06dee9056663df50350e5 (HEAD -> r
elease, origin/release)
Author: iamshaikomerfarooq@gmail.com <sufiomer257@gmail.co
m>
Date: Thu Sep 11 15:24:05 2025 +0530

    Added a from release local

commit dc6cc84503aaa0902207ea5d0d41e2777f718e5d
Author: iamshaikomerfarooq@gmail.com <sufiomer257@gmail.co
m>
Date: Thu Sep 11 15:21:59 2025 +0530

    deleted all files

commit da4e2c5ee61e7aa4c1674a97fa1463e0ade5d13d
Author: iamshaikomerfarooq@gmail.com <sufiomer257@gmail.co
m>
Date: Thu Sep 11 15:03:14 2025 +0530

    committed 4 from release

    This reverts commit 479610cb9adbcf5083c7f32bce80e7335b
5977f6.

commit 479610cb9adbcf5083c7f32bce80e7335b5977f6
Author: iamshaikomerfarooq@gmail.com <sufiomer257@gmail.co
m>
Date: Thu Sep 11 13:25:29 2025 +0530

    adding 4 from release

commit a19e97755b09f9d68eeb69ccfa6aff5eaf759fc
Author: Omer Farooq <iamomerfarooq257@gmail.com>
Date: Wed Sep 10 19:39:06 2025 +0530

    Added 3 in the release of local repo

commit 51aa8c84e9a15147a4154a75a9820def6e1e3072
Author: Omer Farooq <iamomerfarooq257@gmail.com>
:|
```

The screenshot shows a GitHub repository named 'firstrepo' on the left and a terminal window on the right. The GitHub page displays a list of commits, including 'Initial commit' and 'First commit'. The terminal window shows the command '\$ git log --oneline' output, which details 17 commits from 'iamom@Omer' on MINGW64, each with a unique SHA-1 hash and a brief description.

```

iamom@Omer MINGW64 ~/Downloads/farooq/firstrepo
$ git log --oneline
0d07b38 (HEAD -> release, origin/release) Added a from release local
dc6cc84 deleted all files
da4e2c5 committed 4 from release
479610c 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)
$ git reflog
0d07b38 (HEAD -> release, origin/release) HEAD@{0}: reset:
moving to HEAD
0d07b38 (HEAD -> release, origin/release) HEAD@{1}: rebase
(finish): returning to refs/heads/release
0d07b38 (HEAD -> release, origin/release) HEAD@{2}: rebase
: fast-forward
dc6cc84 HEAD@{3}: rebase: fast-forward
da4e2c5 HEAD@{4}: rebase: fast-forward
479610c HEAD@{5}: rebase: fast-forward
a19e977 HEAD@{6}: rebase: fast-forward
51aa8c8 HEAD@{7}: rebase: fast-forward
8472f49 HEAD@{8}: rebase: fast-forward
7e79935 HEAD@{9}: rebase (start): checkout 7e79935d8c0f95c
438d80e182fef5b904547ef30
0d07b38 (HEAD -> release, origin/release) HEAD@{10}: reset
: moving to HEAD-0
0d07b38 (HEAD -> release, origin/release) HEAD@{11}: reset
: moving to HEAD
0d07b38 (HEAD -> release, origin/release) HEAD@{12}: reset
: moving to HEAD
0d07b38 (HEAD -> release, origin/release) HEAD@{13}: reset
: moving to HEAD
0d07b38 (HEAD -> release, origin/release) HEAD@{14}: reset
: moving to HEAD
0d07b38 (HEAD -> release, origin/release) HEAD@{15}: reset
: moving to HEAD
0d07b38 (HEAD -> release, origin/release) HEAD@{16}: commi

```

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.

```
MINGW64/c/Users/iamom/Downloads/farooq/firstrepo
pick 8472f49 # Initial commit
pick 51aa8c8 # First commit
pick a19e977 # Added 3 in the release of local repo
reword 9de1825 # adding 4 from release local
pick d40e994 # committed 4 from release
pick 2e02a05 # deleted all files
pick f168aa1 # Added a from release local

# Rebase f168aa1 onto b0cf263 (7 commands)
#
# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup [-C | -c] <commit> = like "squash" but keep only the previous
#                               commit's log message, unless -C is used, in which case
#                               keep only this commit's message; -c is same as -C but
#                               opens the editor
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit>] [-c <commit>] <label> [<oneline>]
#           create a merge commit using the original merge commit's
#           message (or the oneline, if no original merge commit was
#           specified); use -c <commit> to reword the commit message
# u, update-ref <ref> = track a placeholder for the <ref> to be updated
#                      to this position in the new commits. The <ref> is
#                      updated at the end of the rebase
#
# These lines can be re-ordered; they are executed from top to bottom.
#
.git/rebase-merge/git-rebase-todo[+] [unix] (17:30 11/09/2025)
-- INSERT --
```

```
[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.

jamom@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

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

jamom@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 5th 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.