GIT & GIT HUB task-2

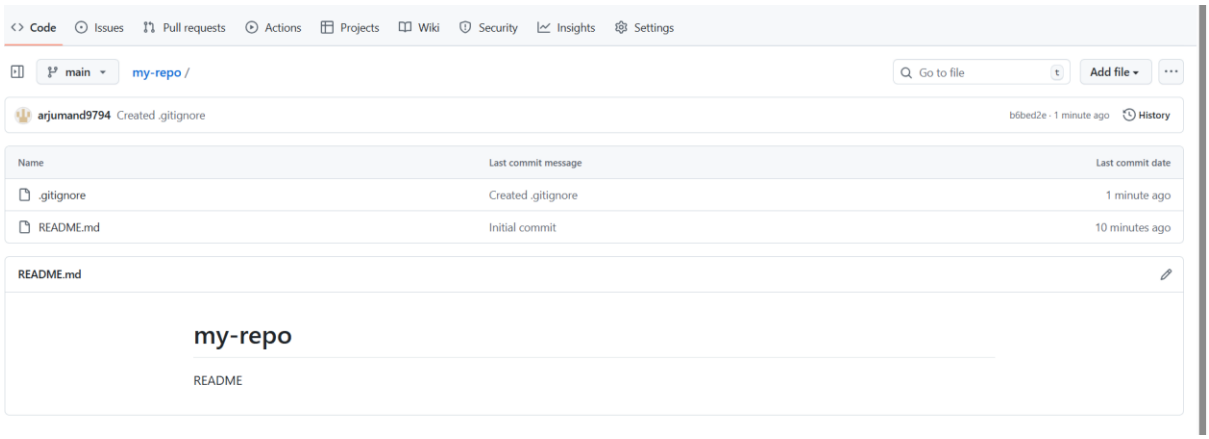
1. Install git.

Create

```
ArjumandM@Arjumand MINGW64 ~/docker (master)
$ git --version
git version 2.37.1.windows.1

ArjumandM@Arjumand MINGW64 ~/docker (master)
$
```

---

2. Created a repo in github with README.md and. ignore file.



3. Clone the created repo to local.

```
        clone

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git clone git@github.com:arjumand9794/arjumandm.git
Cloning into 'arjumandm'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 2), reused 5 (delta 1), pack-reused 0 (from 0)
Receiving objects: 100% (9/9), done.
Resolving deltas: 100% (2/2), done.

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ ls
README.md  a  add  arj  arjumandm/  b  c  d  e  f  file  file2  fileA  fileB

ArjumandM@Arjumand MINGW64 ~/docker (main)
$
```

Steps I Followed

Step 1 — Check existing remote

git remote -v

I saw that my folder was linked to the old repository.

Step 2 — Remove the old remote

git remote remove origin

This removed the old GitHub connection.

Step 3 — Add my NEW GitHub repository as remote

git remote add origin https://github.com/arjumand9794/my-repo.git

Now the folder was linked to the correct repo.

Step 4 — Rename my local branch from master → main

git branch -m master main

This made my local branch match GitHub's default branch.

Step 5 — Pull GitHub main branch (first-time merge)

git pull origin main --allow-unrelated-histories

GitHub had README.md and .gitignore, so I got a merge conflict.

I resolved it and committed the merge.

Step 6 — Add and commit my project files

git add .

git commit -m "merged repo and added project files"

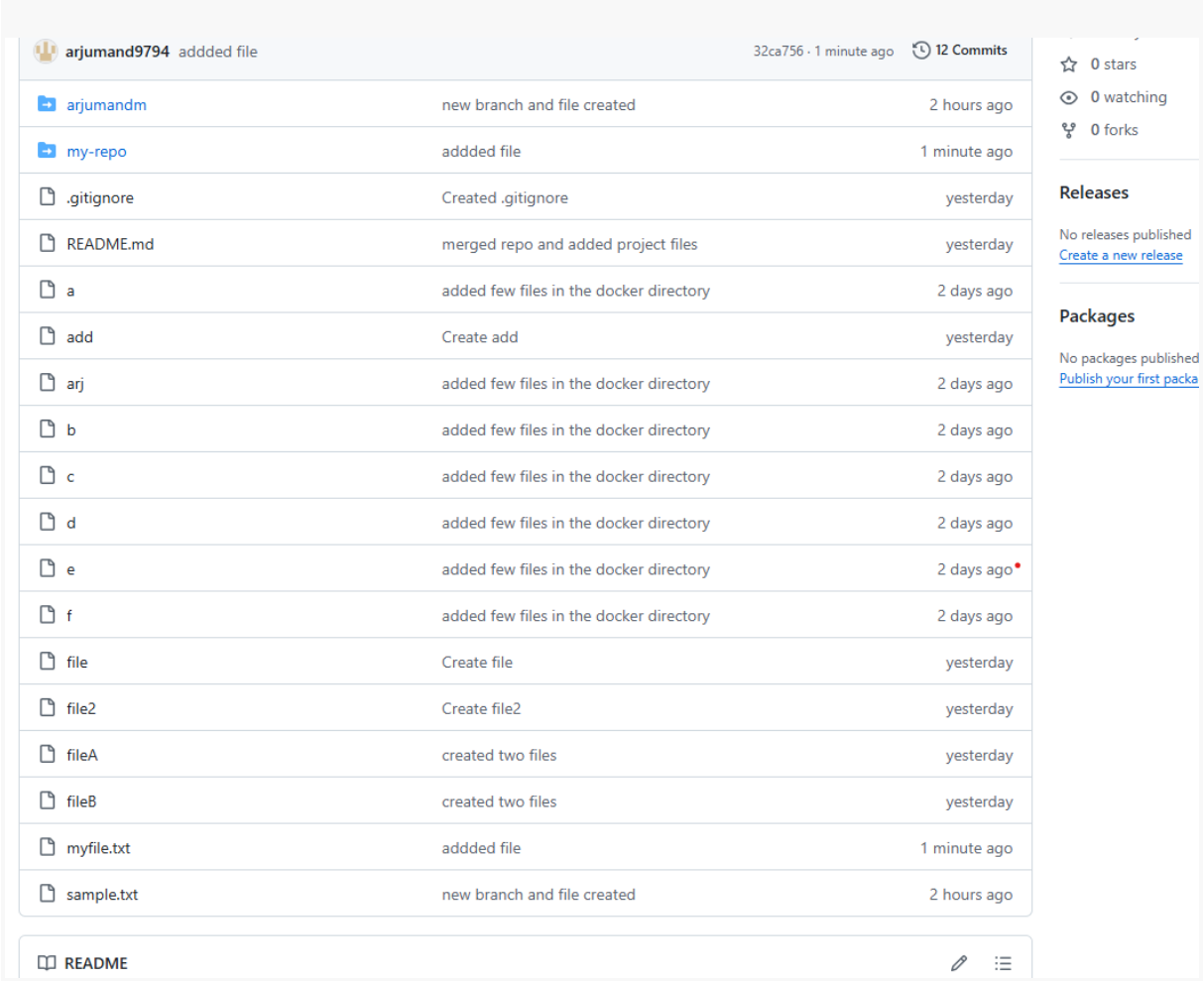
Step 7 — Push my updated project to the NEW repo

git push -u origin main

This successfully uploaded all my files to the new repository on the main branch.

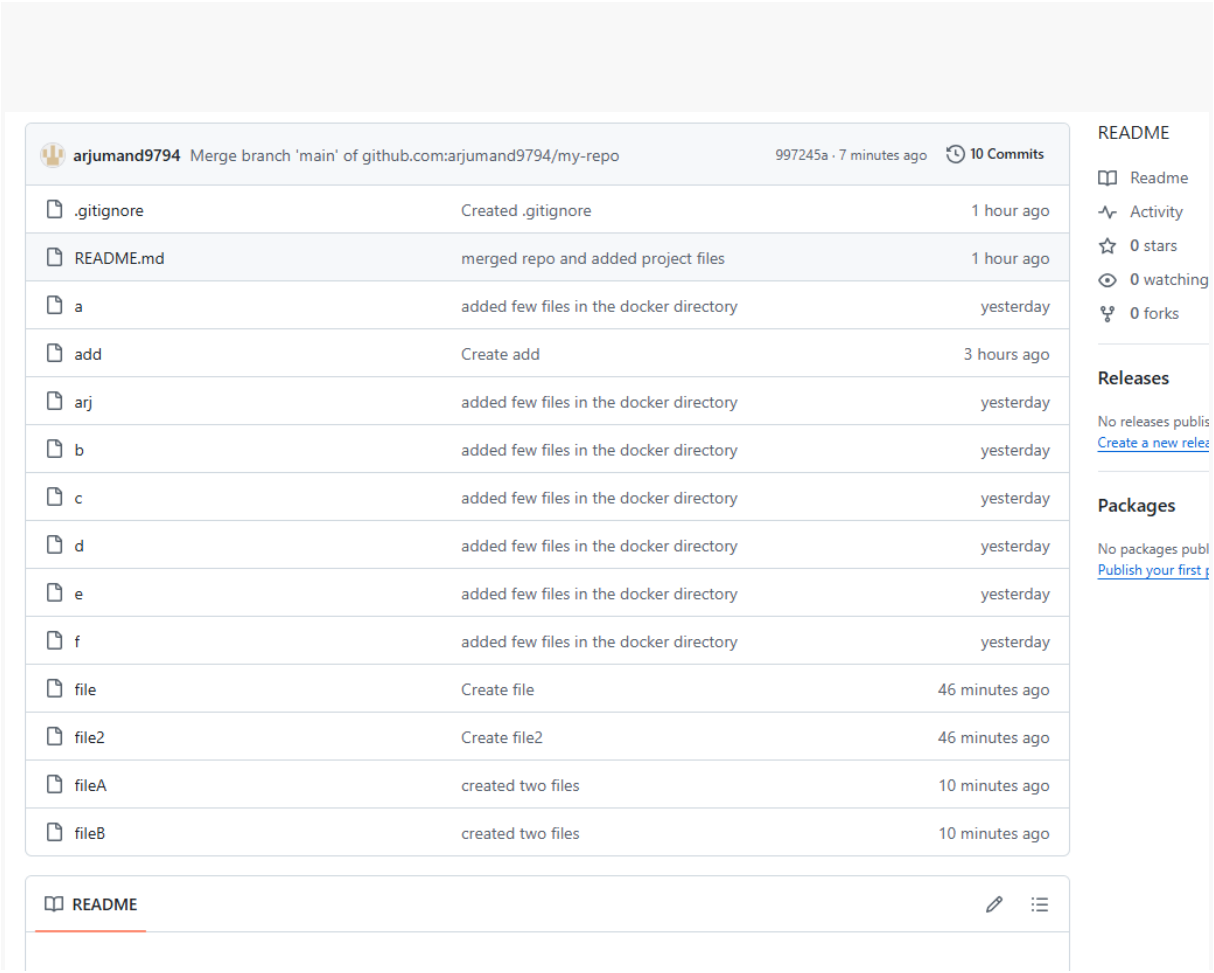
```
$ git push -u origin main
To github.com:arjumand9794/my-repo.git
 ! [rejected]        main -> main (non-fast-forward)
error: failed to push some refs to 'github.com:arjumand9794/my-repo.git'
hint: Updates were rejected because the tip of your current branch is behind
hint: its remote counterpart. Integrate the remote changes (e.g.
hint: 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git pull origin main --allow-unrelated-histories
From github.com:arjumand9794/my-repo
 * branch            main       -> FETCH_HEAD
Auto-merging README.md
CONFLICT (add/add): Merge conflict in README.md
Automatic merge failed; fix conflicts and then commit the result.

ArjumandM@Arjumand MINGW64 ~/docker (main|MERGING)
$ git add.
git: 'add.' is not a git command. See 'git --help'.

The most similar command is
        add

ArjumandM@Arjumand MINGW64 ~/docker (main|MERGING)
$ git add .

ArjumandM@Arjumand MINGW64 ~/docker (main|MERGING)
$ git commit -m "merged repo and added project files"
[main 095984e] merged repo and added project files

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git push origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 8 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (12/12), 1.69 KiB | 246.00 KiB/s, done.
Total 12 (delta 3), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (3/3), done.
To github.com:arjumand9794/my-repo.git
   b6bed2e..095984e  main -> main

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ READ.me (deleted)
bash: syntax error near unexpected token `deleted'

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ README.md (deleted)
bash: syntax error near unexpected token `deleted'

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ ls
README.md  a  add  arj  b  c  d  e  f

ArjumandM@Arjumand MINGW64 ~/docker (main)
$
```

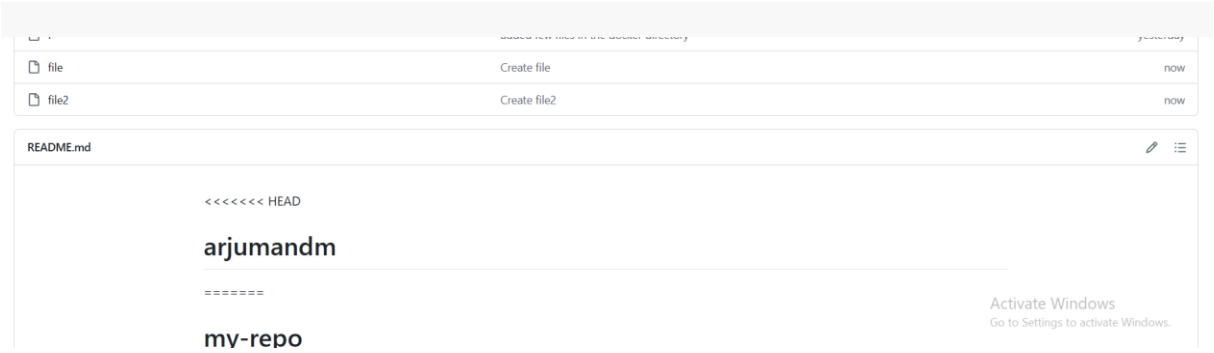
4. Created two files in local repo and in central repo as well. (fileA &fileB) and in
5. central repo files are file1 ansd file2



6. Commit two files and push to central Repository.

```
ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git status
On branch main
nothing to commit, working tree clean

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git status
On branch main
nothing to commit, working tree clean

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git push -u origin main
To github.com:arjumand9794/my-repo.git
 ! [rejected]        main -> main (fetch first)
error: failed to push some refs to 'github.com:arjumand9794/my-repo.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git pull origin main --allow-unrelated-histories
remote: Enumerating objects: 7, done.
remote: Counting objects: 100% (7/7), done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 2), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (6/6), 1.78 KiB | 91.00 KiB/s, done.
From github.com:arjumand9794/my-repo
 * branch            main       -> FETCH_HEAD
   095984e..58d1840  main       -> origin/main
Merge made by the 'ort' strategy.
 file  | 1 +
 file2 | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 file
 create mode 100644 file2

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git push origin main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 506 bytes | 506.00 KiB/s, done.
Total 4 (delta 2), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (2/2), completed with 1 local object.
To github.com:arjumand9794/my-repo.git
   58d1840..997245a  main -> main
```





7. Created a branch in local and create a sample file and push to central.

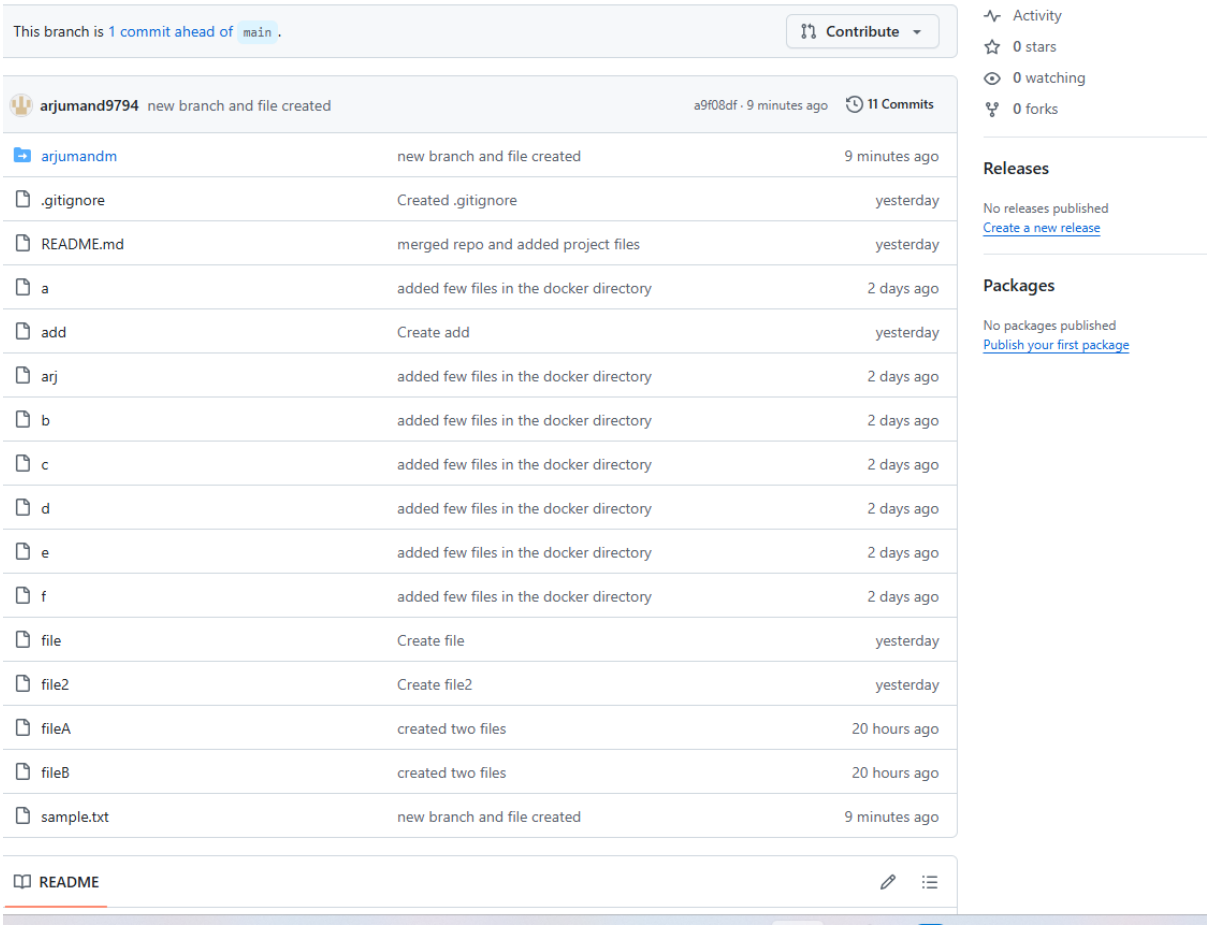
```
rjumandM@Arjumand MINGW64 ~/docker (main)
 ls
EADME.md  a  add  arj  arjumandm/  b  c  d  e  f  file  file2  fileA  fileB

rjumandM@Arjumand MINGW64 ~/docker (main)
 git checkout -b arjumand_m
witched to a new branch 'arjumand_m'

rjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
 branch list
ash: branch: command not found

rjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
 branch --list
ash: branch: command not found

rjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
 git branch
  arjumand_m
  main
```

```
ArjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ echo "this is my new sample file" > sample.txt

ArjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ ls
README.md  a  add  arj  arjumandm/  b  c  d  e  f  file  file2  fileA  fileB  sample.txt
```

Now I have pushed the file which I created in new branch

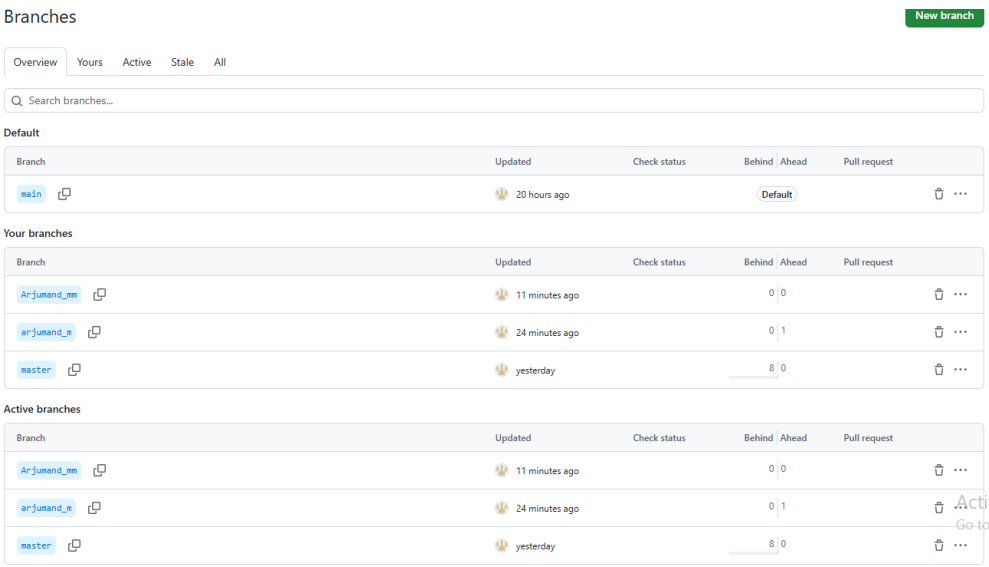
```
n branch arjumand_m
our branch is up to date with 'origin/arjumand_m'.

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

nothing added to commit but untracked files present (use "git add" to track)

ArjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ git add .
warning: adding embedded git repository: arjumandm
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> arjumandm
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached arjumandm
hint:
hint: See "git help submodule" for more information.
warning: in the working copy of 'sample.txt', LF will be replaced by CRLF the next time Git touches it

ArjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ git commit -m "new branch and file created"
[arjumand_m a9f08df] new branch and file created
 2 files changed, 2 insertions(+)
 create mode 160000 arjumandm
 create mode 100644 sample.txt

ArjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ git status
On branch arjumand_m
Your branch is ahead of 'origin/arjumand_m' by 1 commit.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean

ArjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ git push -u origin arjumand_m
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 351 bytes | 351.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:arjumand9794/my-repo.git
   997245a..a9f08df  arjumand_m -> arjumand_m
branch 'arjumand_m' set up to track 'origin/arjumand_m'.
```


Branch and my file sample.txt is successfully moved to central repository



8. Create a branch in github and clone that to local.

I have created a new branch in central repository “Arjumand_mm”



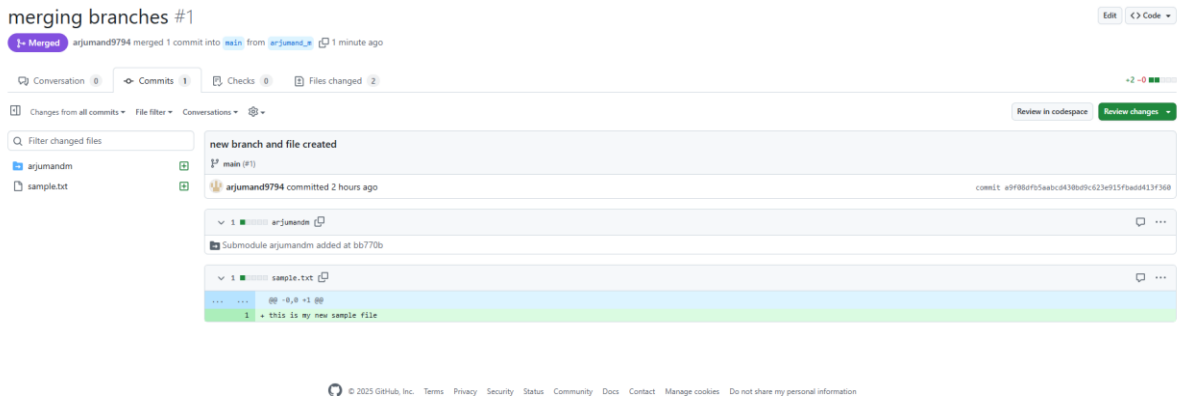
I have clone the branch which I have created.

```
ArjumandM@Arjumand MINGW64 ~/docker (master)
$ git clone --branch arjumand_mm --single-branch git@github.com:arjumand9794/my-repo.git
Cloning into 'my-repo'...
warning: Could not find remote branch arjumand_mm to clone.
fatal: Remote branch arjumand_mm not found in upstream origin

ArjumandM@Arjumand MINGW64 ~/docker (master)
$ git clone --branch Arjumand_mm --single-branch git@github.com:arjumand9794/my-repo.git
Cloning into 'my-repo'...
remote: Enumerating objects: 28, done.
remote: Counting objects: 100% (28/28), done.
remote: Compressing objects: 100% (15/15), done.
remote: Total 28 (delta 7), reused 16 (delta 5), pack-reused 0 (from 0)
Receiving objects: 100% (28/28), 5.59 KiB | 817.00 KiB/s, done.
Resolving deltas: 100% (7/7), done.

ArjumandM@Arjumand MINGW64 ~/docker (master)
$ ls
README.md  arjumandm/  my-repo/

ArjumandM@Arjumand MINGW64 ~/docker (master)
$ cd
```

9. Merged the branches using merge command

```
ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git branch
  arjumand_m
* main
  master

ArjumandM@Arjumand MINGW64 ~/docker (main)
$ git merge arjumand_m
Merge made by the 'ort' strategy.
 sample.txt | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 sample.txt

ArjumandM@Arjumand MINGW64 ~/docker (main)
$
```

10. Merging branches in github



11. Created the file in local repo and push it to branch in hit hub

```
rjumandM@Arjumand MINGW64 ~/docker (main)
$ git checkout arjumand_m
warning: unable to rmdir 'my-repo': Directory not empty
Switched to branch 'arjumand_m'
Your branch is up to date with 'origin/arjumand_m'.

rjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ echo "This is my new file for arjumand_m for arjumand_m branch" > myfile.txt

rjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ git add myfile.txt
warning: in the working copy of 'myfile.txt', LF will be replaced by CRLF the next time Git touches it

rjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ git add .
warning: adding embedded git repository: my-repo
hint: You've added another git repository inside your current repository.
hint: Clones of the outer repository will not contain the contents of
hint: the embedded repository and will not know how to obtain it.
hint: If you meant to add a submodule, use:
hint:
hint:   git submodule add <url> my-repo
hint:
hint: If you added this path by mistake, you can remove it from the
hint: index with:
hint:
hint:   git rm --cached my-repo
hint:
hint: See "git help submodule" for more information.

rjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ git commit -m "addded file"
[arjumand_m 32ca756] addded file
 2 files changed, 2 insertions(+)
 create mode 160000 my-repo
 create mode 100644 myfile.txt

rjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$ git push origin arjumand_m
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 347 bytes | 347.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:arjumand9794/my-repo.git
   a9f08df..32ca756  arjumand_m -> arjumand_m

rjumandM@Arjumand MINGW64 ~/docker (arjumand_m)
$
```

12. I have clone the branch only from github

```
    -X, --strategy-option <option=value>
                          option for selected merge strategy
    -S, --gpg-sign[=<key-id>]
                          GPG sign commit
    --allow-unrelated-histories
                          allow merging unrelated histories

Options related to fetching
    --all                 fetch from all remotes
    -a, --append          append to .git/FETCH_HEAD instead of overwriting
    --upload-pack <path>  path to upload pack on remote end
    -f, --force           force overwrite of local branch
    -t, --tags            fetch all tags and associated objects
    -p, --prune           prune remote-tracking branches no longer on remote
    -j, --jobs[=<n>]      number of submodules pulled in parallel
    --dry-run             dry run
    -k, --keep            keep downloaded pack
    --depth <depth>       deepen history of shallow clone
    --shallow-since <time>
                          deepen history of shallow repository based on time
    --shallow-exclude <revision>
                          deepen history of shallow clone, excluding rev
    --deepen <n>          deepen history of shallow clone
    --unshallow           convert to a complete repository
    --update-shallow      accept refs that update .git/shallow
    --refmap <refmap>     specify fetch refmap
    -o, --server-option <server-specific>
                          option to transmit
    -4, --ipv4            use IPv4 addresses only
    -6, --ipv6            use IPv6 addresses only
    --negotiation-tip <revision>
                          report that we have only objects reachable from this object
    --show-forced-updates
                          check for forced-updates on all updated branches
    --set-upstream        set upstream for git pull/fetch


ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ branch
bash: branch: command not found

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git branch
  Arjumand_mm
* master
```

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

```
ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git add .gitignore
warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git commit -m "stop tracking this file"
[master c99f9fc] stop tracking this file
 1 file changed, 1 insertion(+), 1 deletion(-)

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git status
On branch master
nothing to commit, working tree clean

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ cat .gitignore
secrete.txt

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ ls -1
README.md
a
add
arj
b
c
d
e
f
file
file2
fileA
fileB
fileee
secrete.txt

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$  ls
README.md  a  add  arj  b  c  d  e  f  file  file2  fileA  fileB  fileee  secrete.txt
```

14. First head log is deleted after making the file

```
rjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
 git log -2
ommit c99f9fc0944faf95847bdec0644937d4a65d14f8 (HEAD -> master)
uthor: Arjumand <Arjumand9794@gmail.com>
ate:   Fri Nov 14 18:00:12 2025 +0530

    stop tracking this file

ommit 529d20587c4daed0c581ae6cc5d2dec97ed38a88 (Arjumand_mm)
uthor: Arjumand <Arjumand9794@gmail.com>
ate:   Fri Nov 14 16:20:10 2025 +0530

    new file

rjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
 git reset --mixed HEAD~2
nstaged changes after reset:
        .gitignore

rjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
 git log -2
ommit 997245a6abf949af382410403a312a291ca4881e (HEAD -> master, origin/Arjumand_mm)
erge: 46ee45a 58d1840
uthor: Arjumand <Arjumand9794@gmail.com>
ate:   Thu Nov 13 19:04:13 2025 +0530

    Merge branch 'main' of github.com:arjumand9794/my-repo

ommit 46ee45a0f82c2cf64b7e32e9e33faa8f9329a144
uthor: Arjumand <Arjumand9794@gmail.com>
ate:   Thu Nov 13 19:00:57 2025 +0530

    created two files

rjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
```

I have changed the message of file2 by using command ‘git revert 58d1840.

```
ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git log --oneline
6910fcd (HEAD -> master) Revert "revert is now working"
997245a (origin/Arjumand_mm) Merge branch 'main' of github.com:arjumand9794/my-repo
46ee45a created two files
58d1840 Create file2
200320e Create file
095984e merged repo and added project files
b6bed2e Created .gitignore
5c69f79 Initial commit
bb770b2 Create add
8d667d0 first commit
833ed75 added few files in the docker directory

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git revert 58d1840
On branch master
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)
        fileee
        test.txt

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

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
```

15. Pushed a file to stash memory without saving the changes and work on another file.

I have moved a file called test.txt to stash memory using command git stash push -m “stash sigle file” -u – tst.txt”

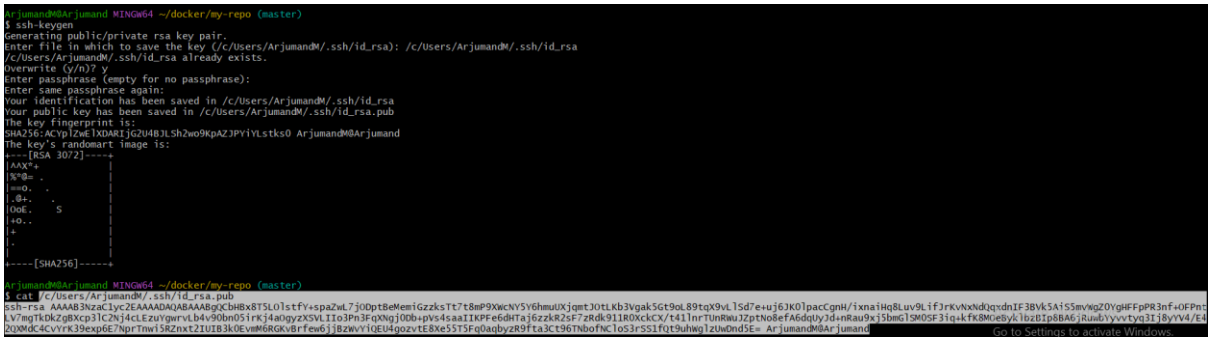
```
rjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
 git stash push -m "stash single file" test.txt
rror: pathspec ':(,prefix:0)test.txt' did not match any file(s) known to git
id you forget to 'git add'?

rjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
 git stash push -m "stash single file" -u -- test.txt
aved working directory and index state On master: stash single file

rjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
 git stash list
tash@{0}: On master: stash single file
tash@{1}: On master: moving to stash memory

rjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
 vi file1

rjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
 git status
```

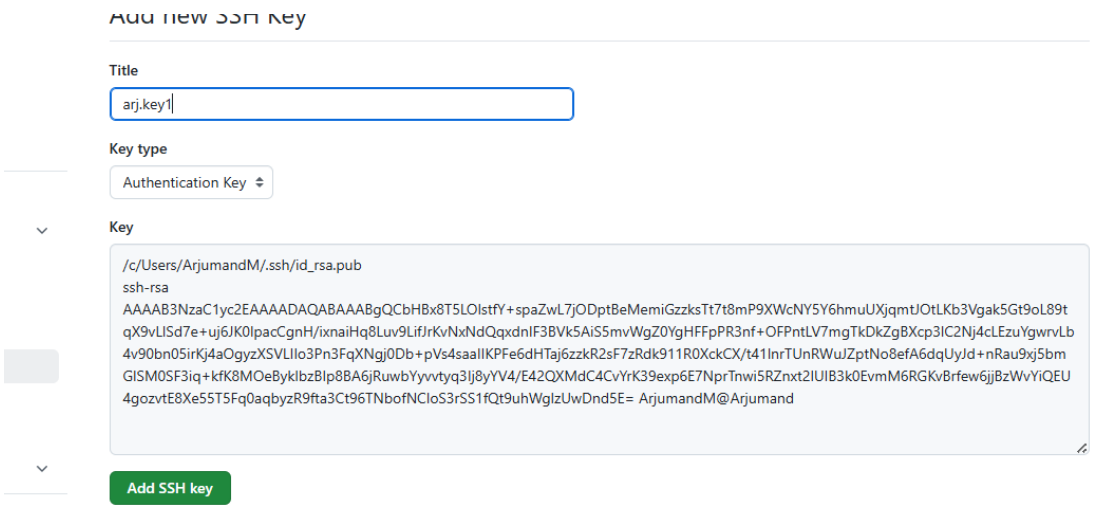
16. Undo the stash file and start working on that again.

I brought ‘test.txt’ file back untracking area from stash memory using command git stash pop

```
ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git stash pop
Already up to date.
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        file1
        fileee
        secrete.txt
        test.txt

nothing added to commit but untracked files present (use "git add" to track)
Dropped refs/stash@{0} (2770cbb623ec8dabcc1a813e79bdf8c94ddf6c0d)

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git stash list
stash@{0}: On master: moving to stash memory
```

17. Generate a ssh-keygen and configure into github.





18. Skipping as of now.

19. Basic understanding of .git file.

What is the .git Folder?

The .git folder is the brain of your Git repository.

It contains everything Git needs to track versions, branches, commits, logs, and configurations.

You can think of it like the database for your project.

It is automatically created when you run:

git init

or when you clone a repo:

git clone repo-url

🟩 Where is the .git folder located?

Inside your project folder:

my-repo/

.git/ ← THIS is the hidden Git database

file1

file2

README.md

🟧 Why is .git hidden?

It’s hidden because:

You should not modify anything manually inside it

It contains internal Git data

Changing files inside .git may corrupt your repo

🟪 What does the .git folder contain? (Basic understanding)

Inside .git you will find:

.git/

├– HEAD

├– config

├– objects/

├– refs/

├– hooks/

├– logs/

└– index

Here’s what they mean in simple words:

 HEAD

Shows which branch you are currently on.

Example content:

ref: refs/heads/master

Meaning:

✓ You are on master branch.

 config

Git settings for your repository.

Example:

[user]

  name = Arjumand

  email = arjumand@example.com

 objects/

Where Git stores actual data:

commits

file contents (blobs)

trees

histories

Git saves everything here as compressed snapshots.

This is the most important folder.

 refs/

Stores pointers to:

branches

tags

remote branches

Example:

refs/heads/master

refs/remotes/origin/main

 logs/

Stores history of actions:

commits

merges

checkouts

reverts

Useful for recovering deleted commits with git reflog.

 hooks/

Scripts that run on Git events like:

before commit

after push

(but usually unused by beginners)

 index

Staging area (your “git add” data).

20. Check all the logs of git

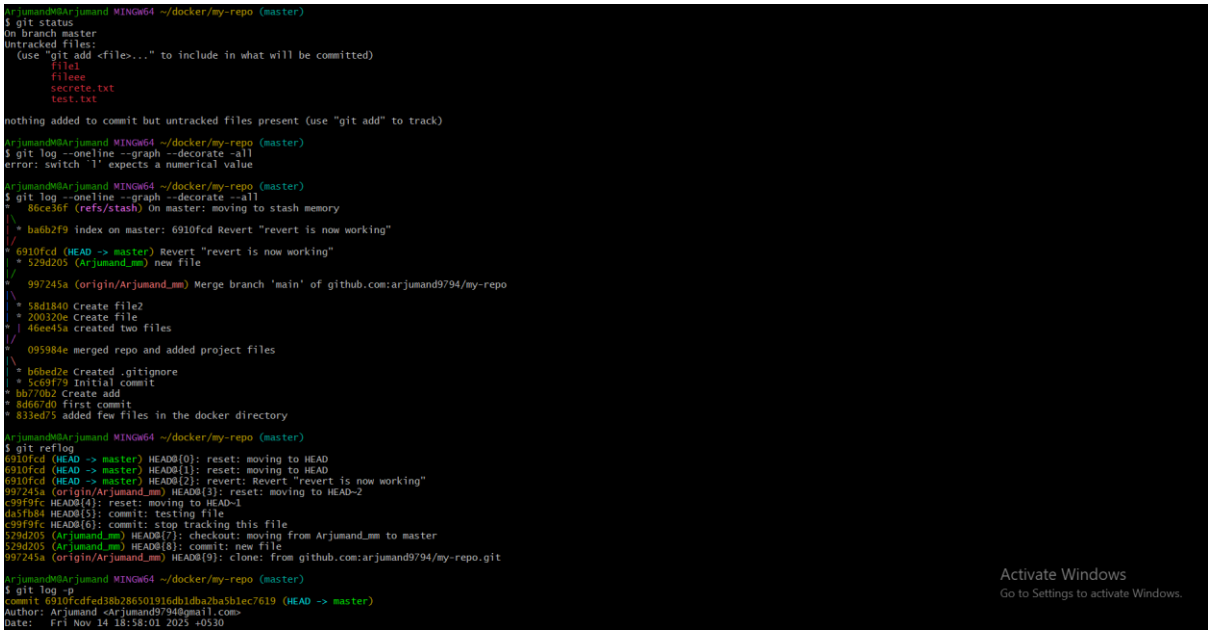
git status # current state

git log --oneline --graph --decorate --all

git reflog # history of HEAD moves

git show <commit> # details of a commit

git log -p # show patches



```
Author: Arjumand <Arjumand9794@gmail.com>
Date:   Fri Nov 14 18:58:01 2025 +0530

    Revert "revert is now working"

    This reverts commit 58d18403d3dace28c7a7e9c9ff1bd220472d17b6.

diff --git a/file2 b/file2
deleted file mode 100644
index c645d45..0000000
--- a/file2
+++ /dev/null
@@ -1 +0,0 @@
-this my second file in repo

commit 997245a6abf949af382410403a312a291ca4881e (origin/Arjumand_mm)
Merge: 46ee45a 58d1840
Author: Arjumand <Arjumand9794@gmail.com>
Date:   Thu Nov 13 19:04:13 2025 +0530

    Merge branch 'main' of github.com:arjumand9794/my-repo

commit 46ee45a0f82c2cf64b7e32e9e33faa8f9329a144
Author: Arjumand <Arjumand9794@gmail.com>
Date:   Thu Nov 13 19:00:57 2025 +0530

    created two files

diff --git a/fileA b/fileA
new file mode 100644
index 0000000..e69de29
diff --git a/fileB b/fileB
new file mode 100644
index 0000000..e69de29

commit 58d18403d3dace28c7a7e9c9ff1bd220472d17b6
Author: Arjumand Mohiuddin <arjumand9794@gmail.com>
Date:   Thu Nov 13 18:25:12 2025 +0530

    Create file2

diff --git a/file2 b/file2
new file mode 100644
index 0000000..c645d45
--- /dev/null
+++ b/file2
@@ -0,0 +1 @@
+this my second file in repo

commit 200320e096da5eca6b9972fbc299b5db0c49e921
Author: Arjumand Mohiuddin <arjumand9794@gmail.com>
Date:   Thu Nov 13 18:24:30 2025 +0530

    Create file

diff --git a/file b/file
:
```

21. Rename the commit messages

```
pick 46ee45a created two files
pick 200320e Create file
pick 58d1840 Create file2
pick 83e1145 New commit message

# Rebase 095984e..83e1145 onto 095984e (4 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
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
```

Now I have changed the first commit message

Procedure:

1. Rename the latest commit message

git commit --amend -m "New commit message"

git push --force # only if already pushed

2. Rename an older commit message

git rebase -i HEAD~<number_of_commits>

```
pick 46ee45a created two files and modifies by the text and renamed commit successfully
pick 200320e Create file
pick 58d1840 Create file2
pick 83e1145 New commit message

# Rebase 095984e..83e1145 onto 095984e (4 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
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
~
~
~
~
~
~
~
~
~
```

22. Merge multiple commits into single commit.

```
ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git rebase -i HEAD~3
Successfully rebased and updated refs/heads/master.

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ ^C

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git rebase -i HEAD~3
[detached HEAD c3c7281] Create file
 Author: Arjumand Mohiuddin <arjumand9794@gmail.com>
 Date: Thu Nov 13 18:24:30 2025 +0530
 1 file changed, 1 insertion(+)
 create mode 100644 file
Successfully rebased and updated refs/heads/master.

ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git log
```

```
pick 7f45a79 Create file
squash f5b4fc1 Create file2
squash af5fbe6 New commit message

# Rebase 46ee45a..af5fbe6 onto 46ee45a (3 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
#
# These lines can be re-ordered; they are executed from top to bottom.
#
# If you remove a line here THAT COMMIT WILL BE LOST.
#
# However, if you remove everything, the rebase will be aborted.
#
~
~
~
~
```

git rebase -i HEAD~3

In the editor:

First commit → keep pick

All others → change pick → squash

Save & close the editor.

Enter the final commit message when Git asks.

```
ArjumandM@Arjumand MINGW64 ~/docker/my-repo (master)
$ git log
commit c3c728187e70f5654f44e2511c361225739ccf81 (HEAD -> master)
Author: Arjumand Mohiuddin <arjumand9794@gmail.com>
Date:   Thu Nov 13 18:24:30 2025 +0530

    Create file

    Create file2

    New commit message
```