Arpit Bhagat

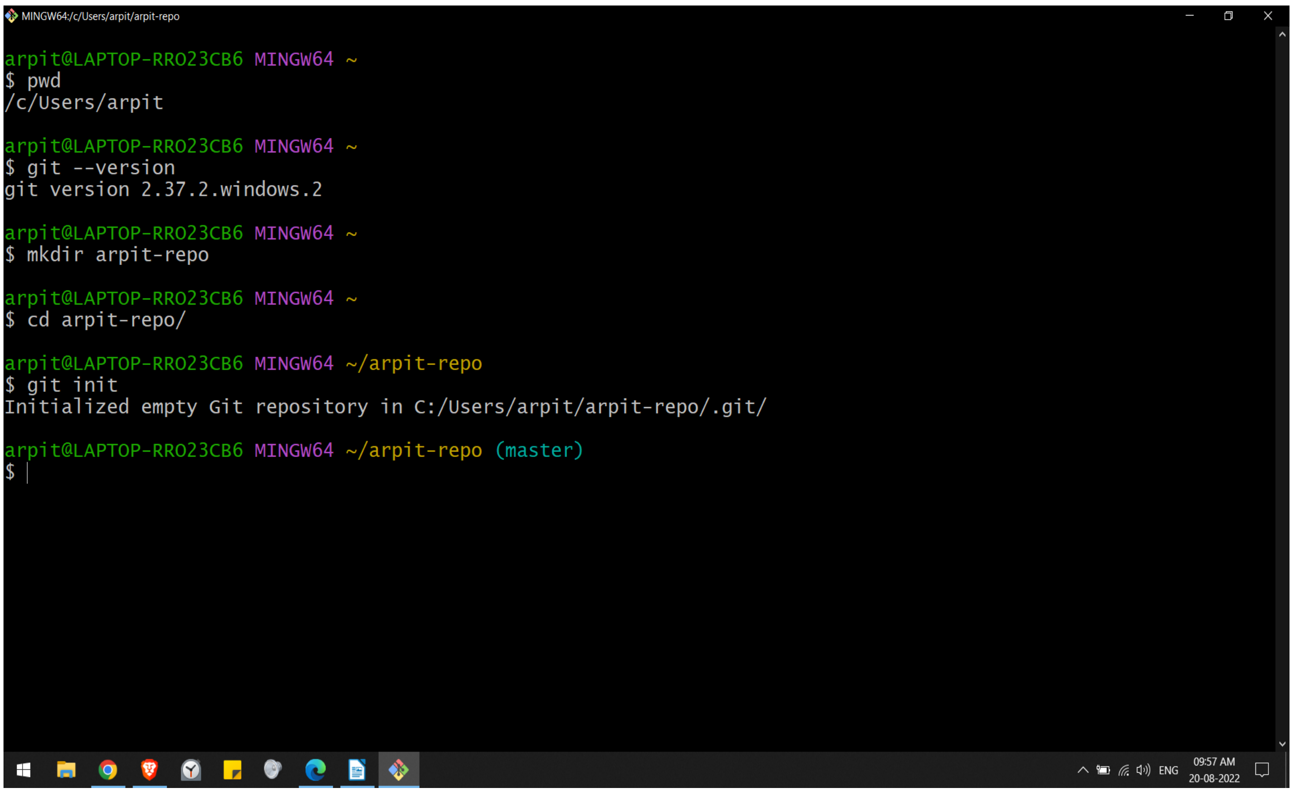
What is Git?

Git is a free and open source distributed version control system designed to handle everything from small to very large projects with speed and efficiency.
Git is easy to learn and has a tiny footprint with lightning fast performance. It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like cheap local branching, convenient staging areas, and multiple workflows.

Basic Git commands:

1) git init

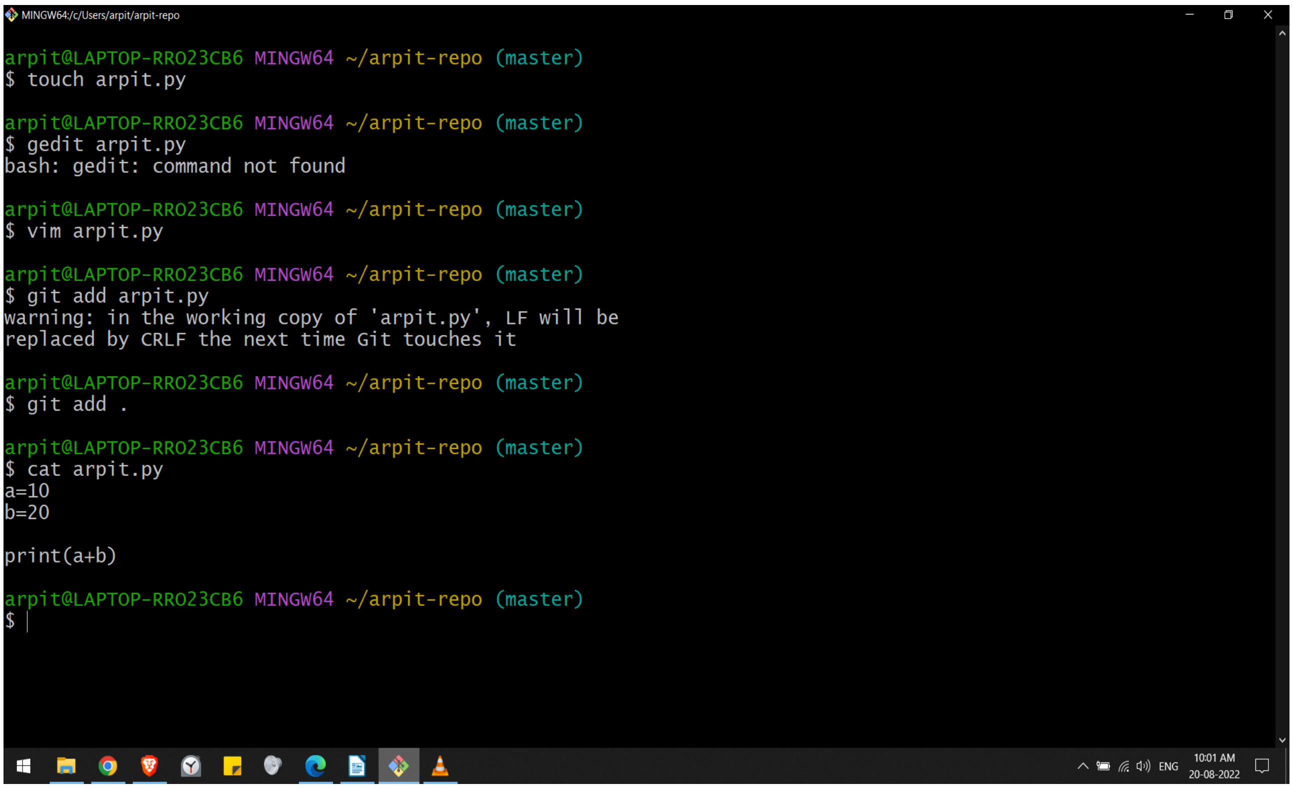
The git init command creates a new Git repository. It can be used to convert an existing, unversioned project to a Git repository or initialize a new, empty repository.



```
arpit@LAPTOP-RRO23CB6 MINGW64 ~
$ pwd
/c/Users/arpit

arpit@LAPTOP-RRO23CB6 MINGW64 ~
$ git --version
git version 2.37.2.windows.2

arpit@LAPTOP-RRO23CB6 MINGW64 ~
$ mkdir arpit-repo

arpit@LAPTOP-RRO23CB6 MINGW64 ~
$ cd arpit-repo/

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo
$ git init
Initialized empty Git repository in C:/Users/arpit/arpit-repo/.git/

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ |
```

2) git add

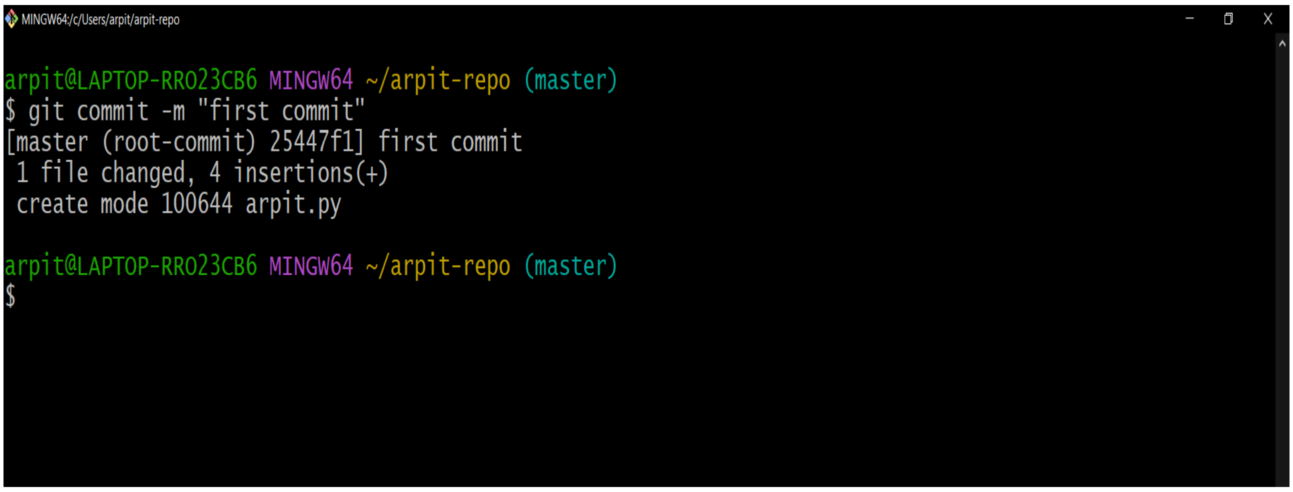
The git add command adds a change in the working directory to the staging area. It tells Git that you want to include updates to a particular file in the next commit.



```
arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ touch arpit.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ gedit arpit.py
bash: gedit: command not found

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ vim arpit.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git add arpit.py
warning: in the working copy of 'arpit.py', LF will be
replaced by CRLF the next time Git touches it

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git add .

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ cat arpit.py
a=10
b=20

print(a+b)

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ |
```

3) git commit

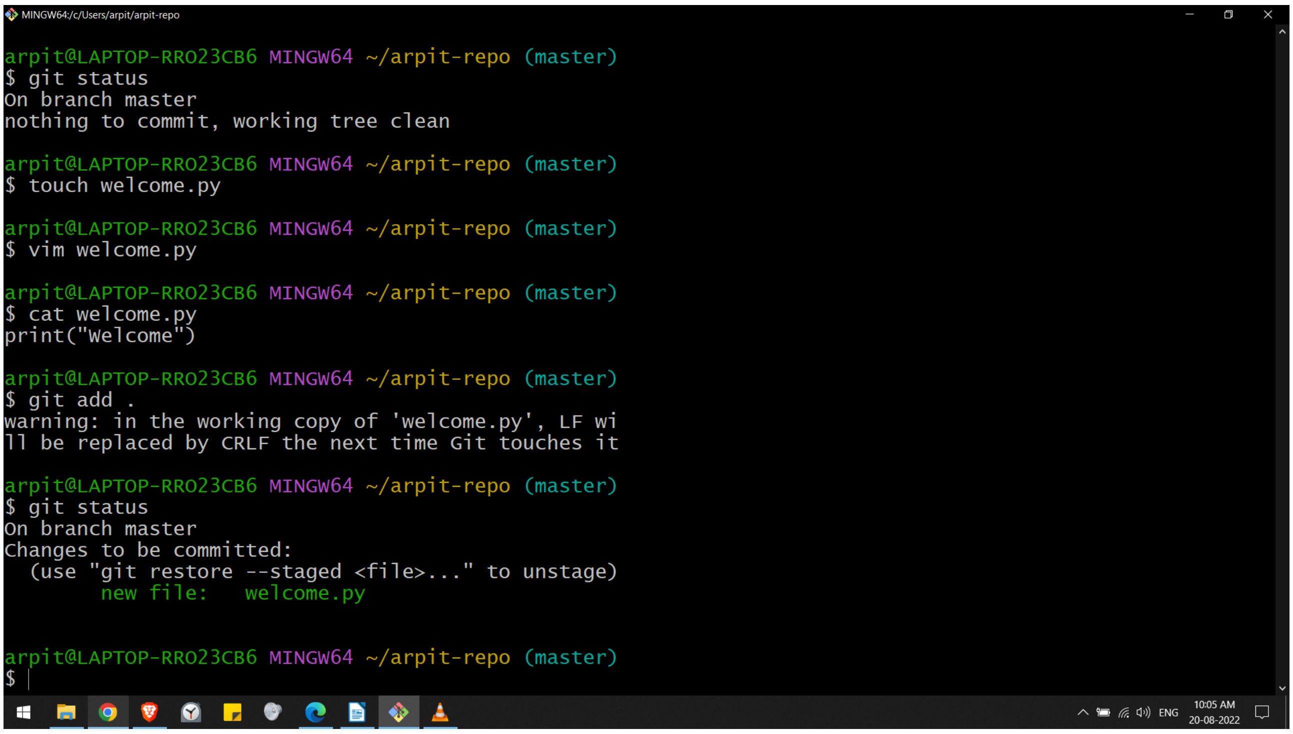
The git commit command captures a snapshot of the project's currently staged changes The "commit" command is used to save your changes to the local repository, and the -m "message" adds a message.



```
arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git commit -m "first commit"
[master (root-commit) 25447f1] first commit
 1 file changed, 4 insertions(+)
 create mode 100644 arpit.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$
```

4) git status

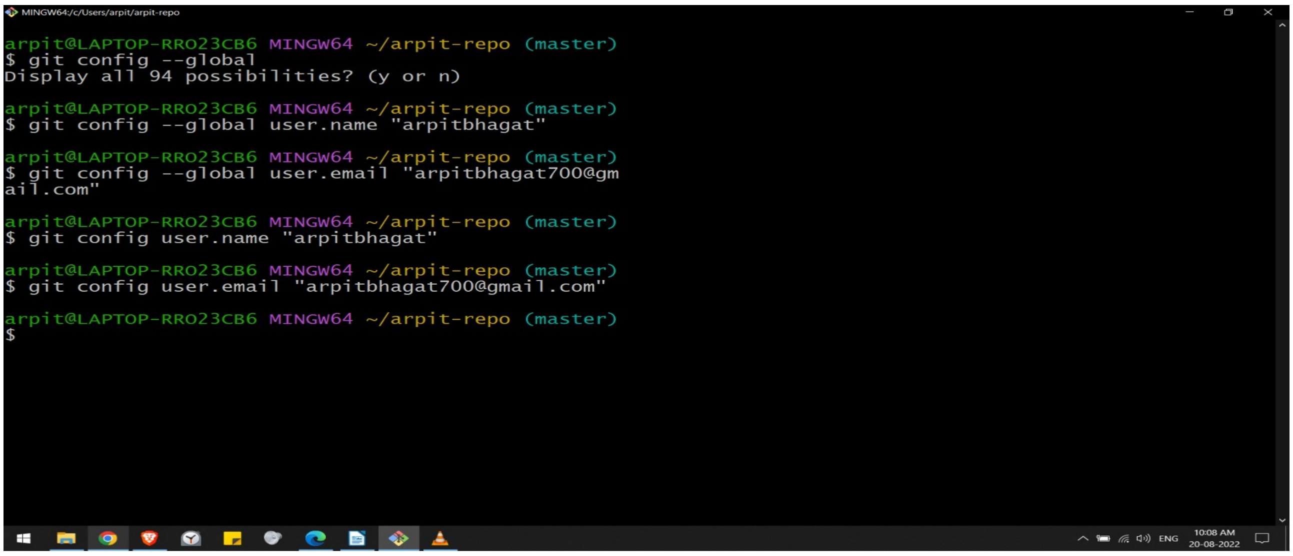
The git status command displays the state of the working directory and the staging area. It lets you see which changes have been staged, which haven't, and which files aren't being tracked by Git.



```
MINGW64:/c/Users/arpit/arpit-repo

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git status
On branch master
nothing to commit, working tree clean

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ touch welcome.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ vim welcome.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ cat welcome.py
print("Welcome")

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git add .
warning: in the working copy of 'welcome.py', LF wi
ll be replaced by CRLF the next time Git touches it

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   welcome.py


arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ |
```

5) git config

The git config command is a convenience function that is used to set Git configuration values on a global or local project level. These configuration levels correspond to . gitconfig text files. Executing git config will modify a configuration text file.



```
MINGW64:/c/Users/arpit/arpit-repo
arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git config --global
Display all 94 possibilities? (y or n)

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git config --global user.name "arpitbhagat"

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git config --global user.email "arpitbhagat700@gm
ail.com"

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git config user.name "arpitbhagat"

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git config user.email "arpitbhagat700@gmail.com"

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$
```

6) Branching command:

1> git branch:

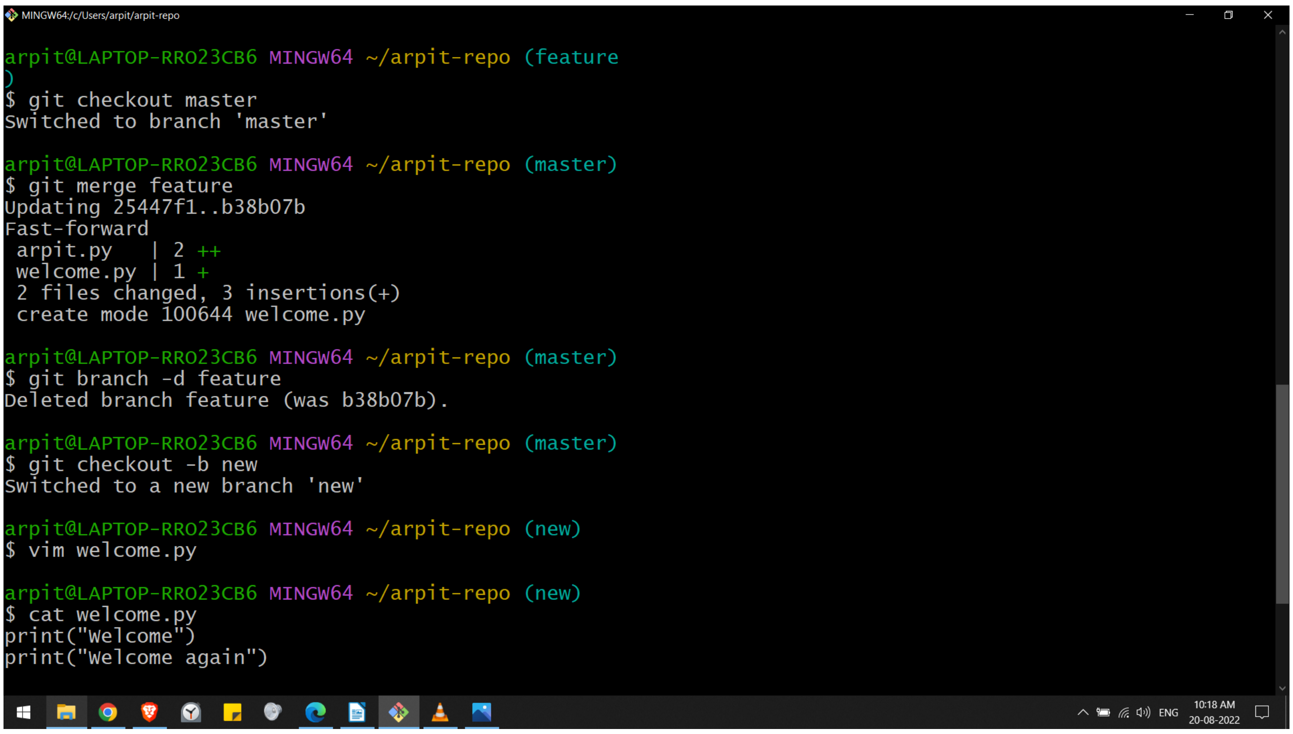
The git branch command lets you create, list, rename, and delete branches. It doesn't let you switch between branches or put a forked history back together again. For this reason, git branch is tightly integrated with the git checkout and git merge commands.

2> git merge:

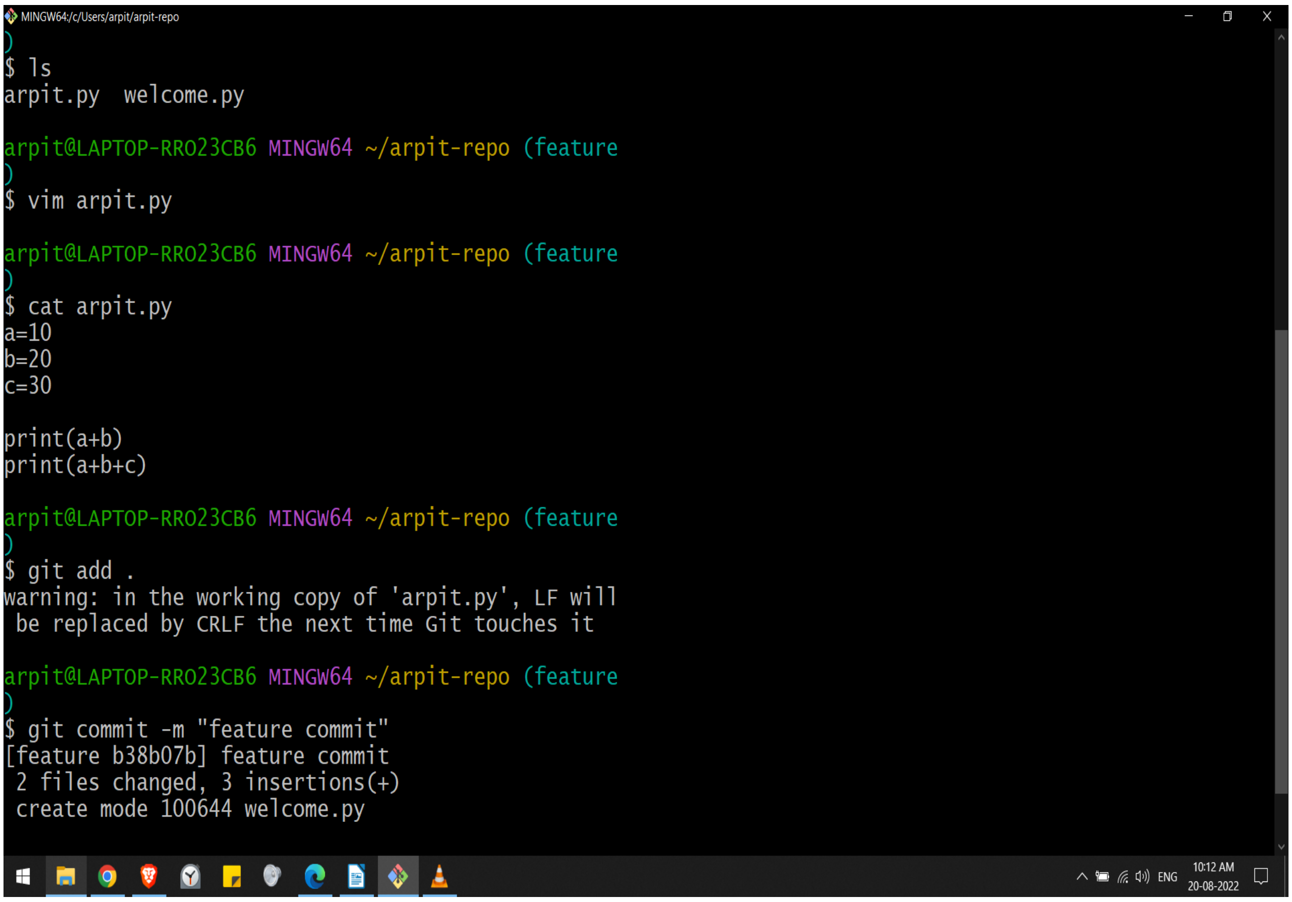
Merging is Git's way of putting a forked history back together again. The git merge command lets you take the independent lines of development created by git branch and integrate them into a single branch.

3> git checkout

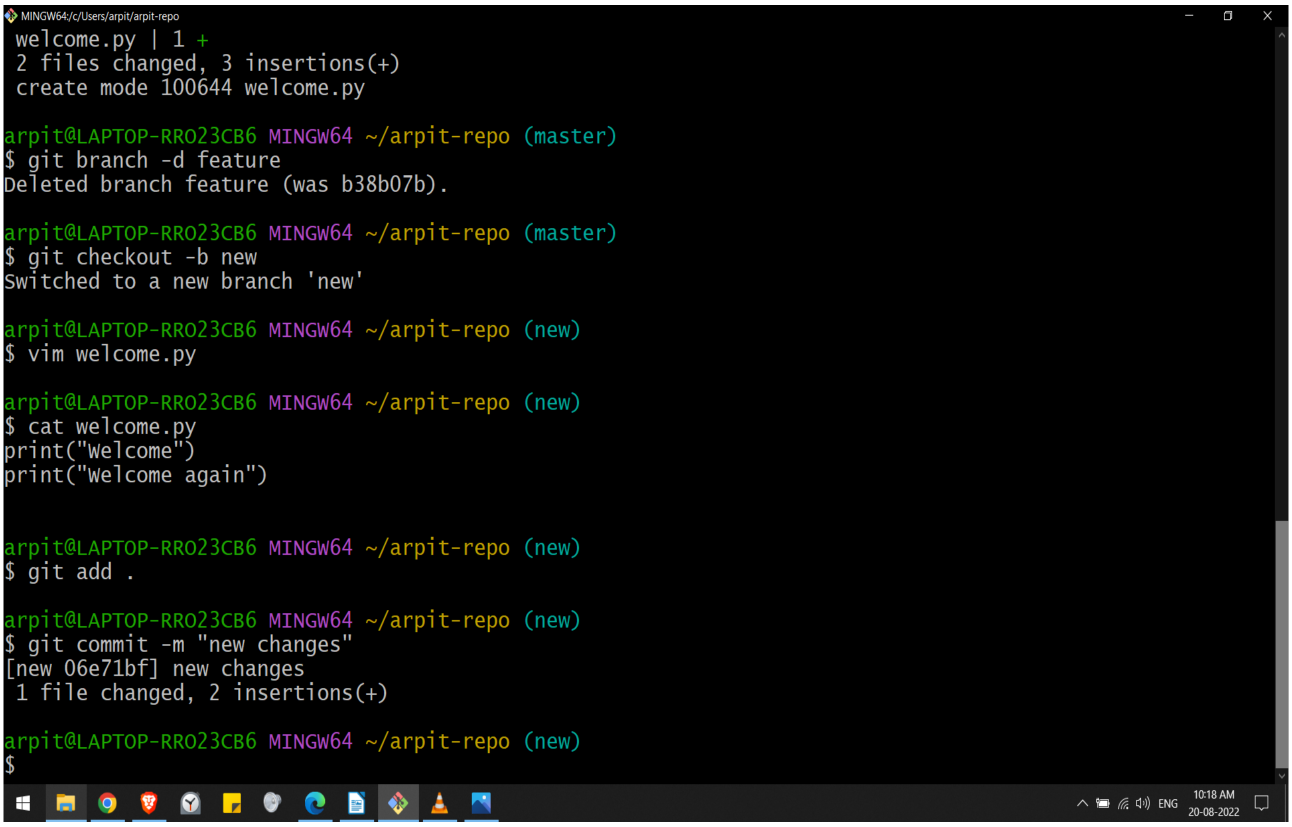
The git checkout command lets you navigate between the branches created by git branch . Checking out a branch updates the files in the working directory to match the version stored in that branch, and it tells Git to record all new commits on that branch.



```
arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (feature
)
$ git checkout master
Switched to branch 'master'

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git merge feature
Updating 25447f1..b38b07b
Fast-forward
 arpit.py   | 2 ++
 welcome.py | 1 +
 2 files changed, 3 insertions(+)
 create mode 100644 welcome.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git branch -d feature
Deleted branch feature (was b38b07b).

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git checkout -b new
Switched to a new branch 'new'

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (new)
$ vim welcome.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (new)
$ cat welcome.py
print("Welcome")
print("Welcome again")
```



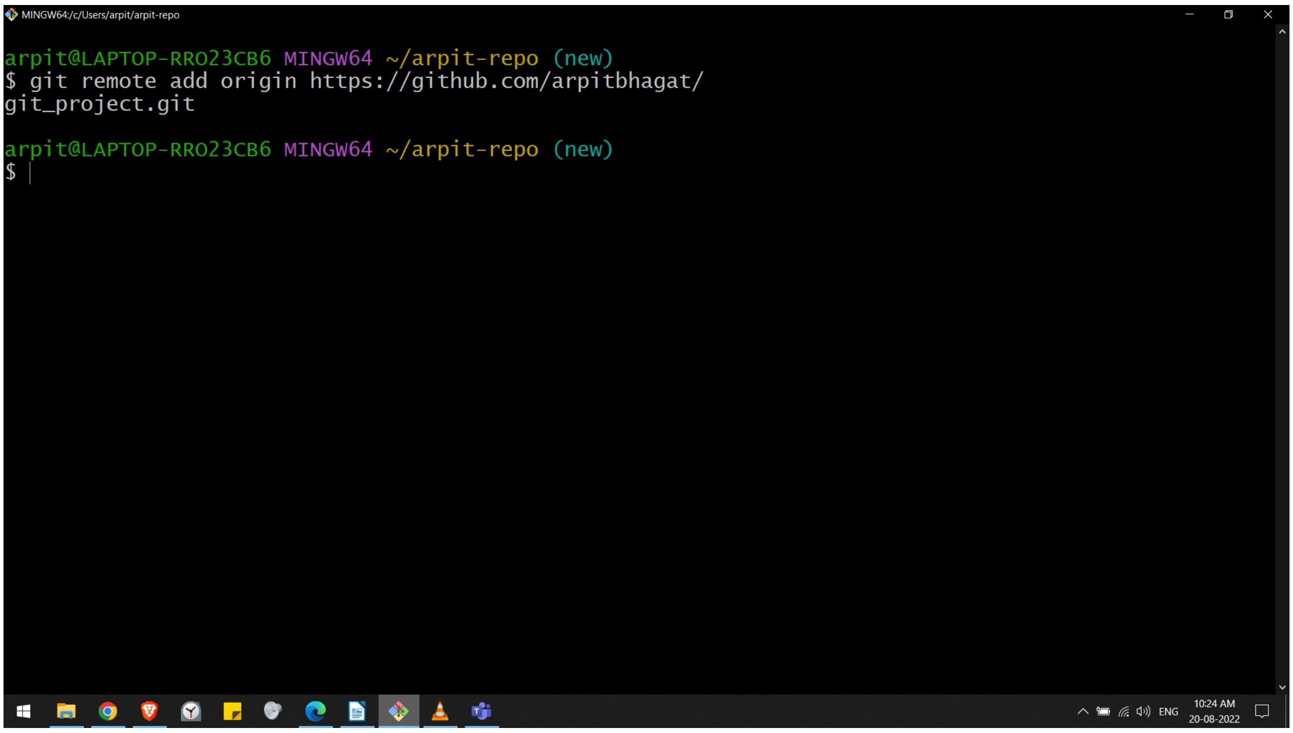
```
)
$ ls
arpit.py  welcome.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (feature
)
$ vim arpit.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (feature
)
$ cat arpit.py
a=10
b=20
c=30

print(a+b)
print(a+b+c)

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (feature
)
$ git add .
warning: in the working copy of 'arpit.py', LF will
 be replaced by CRLF the next time Git touches it

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (feature
)
$ git commit -m "feature commit"
[feature b38b07b] feature commit
 2 files changed, 3 insertions(+)
 create mode 100644 welcome.py
```



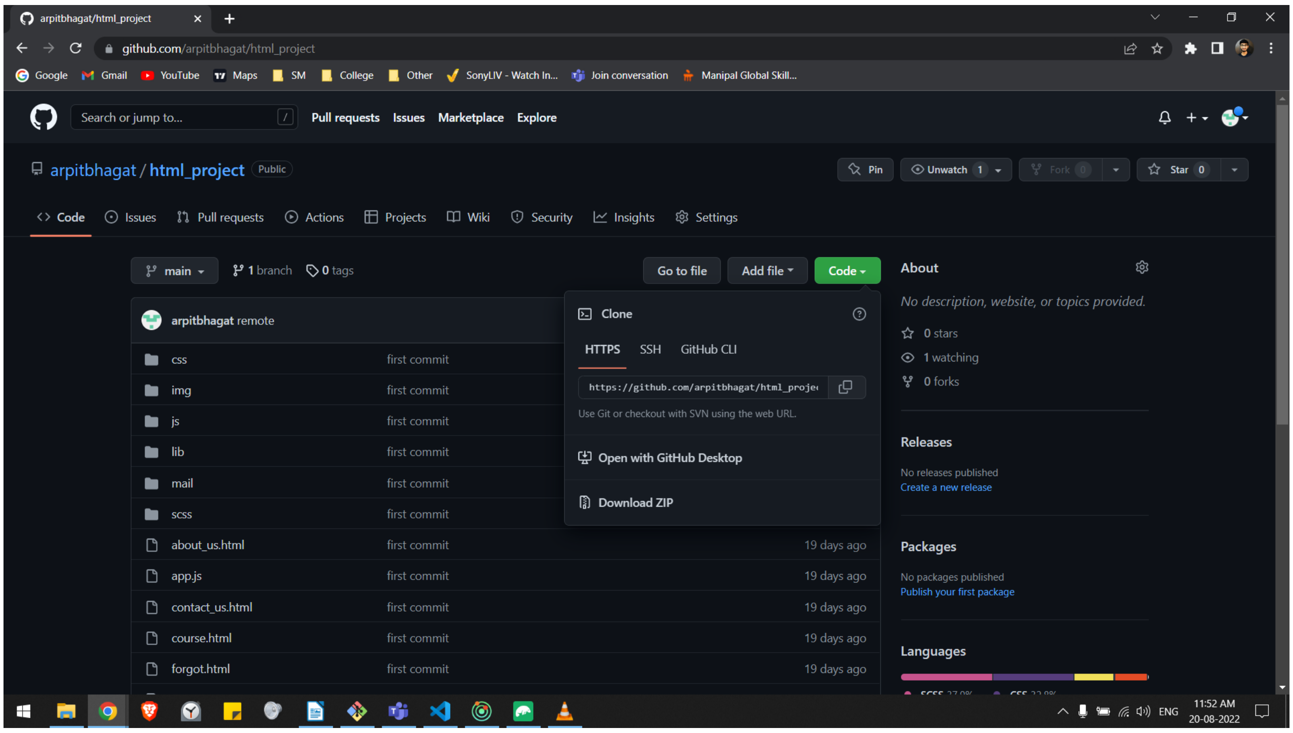
```
 welcome.py | 1 +
 2 files changed, 3 insertions(+)
 create mode 100644 welcome.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git branch -d feature
Deleted branch feature (was b38b07b).

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (master)
$ git checkout -b new
Switched to a new branch 'new'

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (new)
$ vim welcome.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (new)
$ cat welcome.py
print("Welcome")
print("Welcome again")


arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (new)
$ git add .

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (new)
$ git commit -m "new changes"
[new 06e71bf] new changes
 1 file changed, 2 insertions(+)

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (new)
$
```

7) git remote

The git remote command lets you create, view, and delete connections to other repositories. Remote connections are more like bookmarks rather than direct links into other repositories.



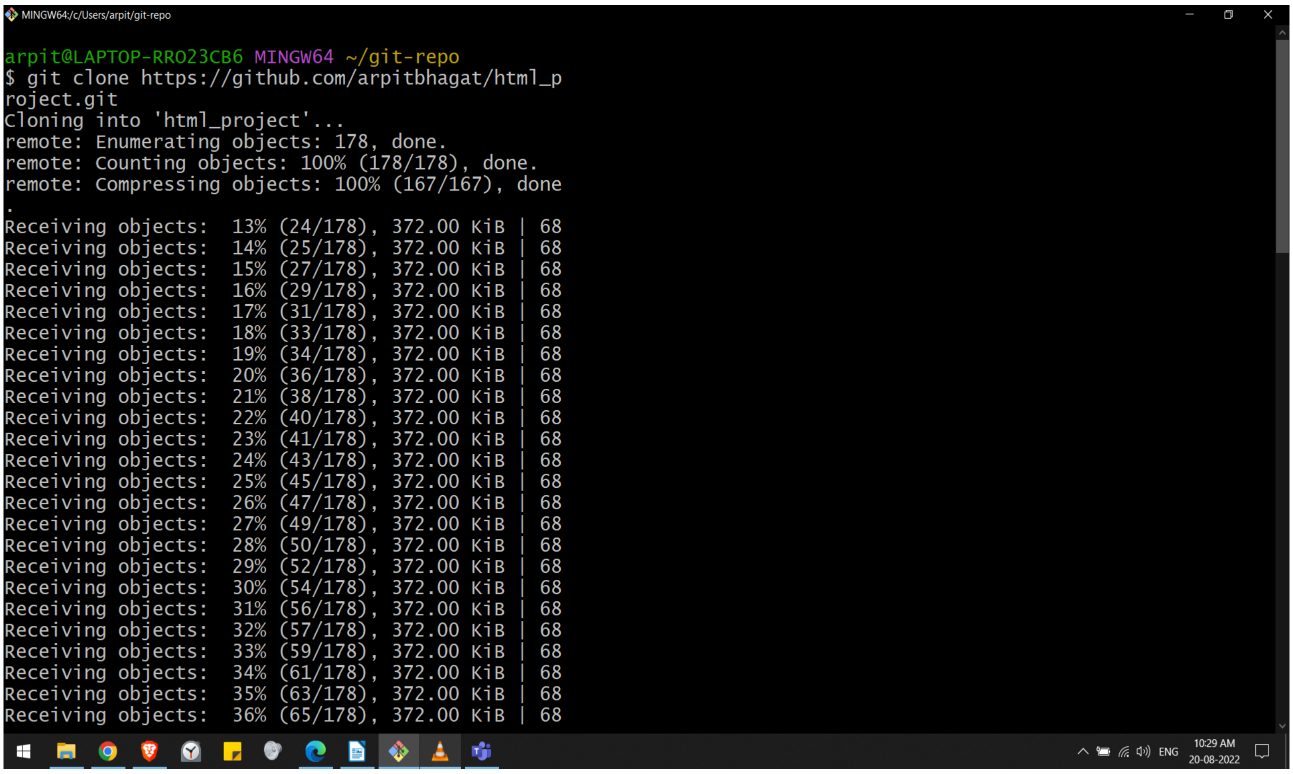
```
MINGW64:/c/Users/arpit/arpit-repo

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (new)
$ git remote add origin https://github.com/arpitbhagat/
git_project.git

arpit@LAPTOP-RRO23CB6 MINGW64 ~/arpit-repo (new)
$ 
```



```
arpitbhagat/html_project
github.com/arpitbhagat/html_project

arpitbhagat / html_project  Public

main   1 branch   0 tags                Go to file   Add file   Code

arpitbhagat remote
css               first commit
img               first commit
js                first commit
lib               first commit
mail              first commit
scss              first commit
about_us.html     first commit    19 days ago
app.js            first commit    19 days ago
contact_us.html   first commit    19 days ago
course.html       first commit    19 days ago
forgot.html       first commit    19 days ago

Clone
HTTPS   SSH   GitHub CLI
https://github.com/arpitbhagat/html_proje
Use Git or checkout with SVN using the web URL.
Open with GitHub Desktop
Download ZIP

About
No description, website, or topics provided.
0 stars
1 watching
0 forks

Releases
No releases published
Create a new release

Packages
No packages published
Publish your first package

Languages
```

8) git clone

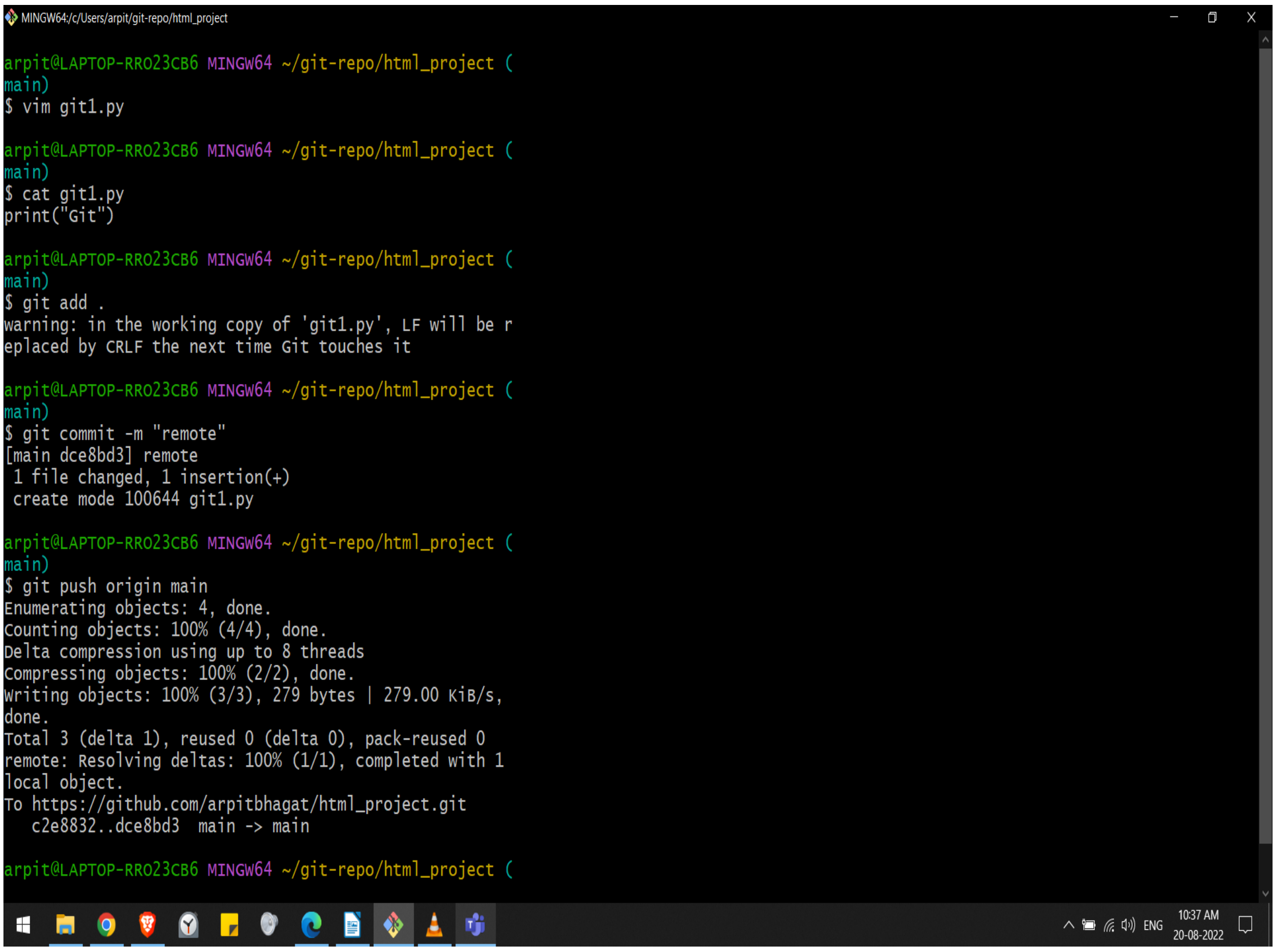
Usage. git clone is primarily used to point to an existing repo and make a clone or copy of that repo at in a new directory, at another location. The original repository can be located on the local filesystem or on remote machine accessible supported protocols. The git clone command copies an existing Git repository.



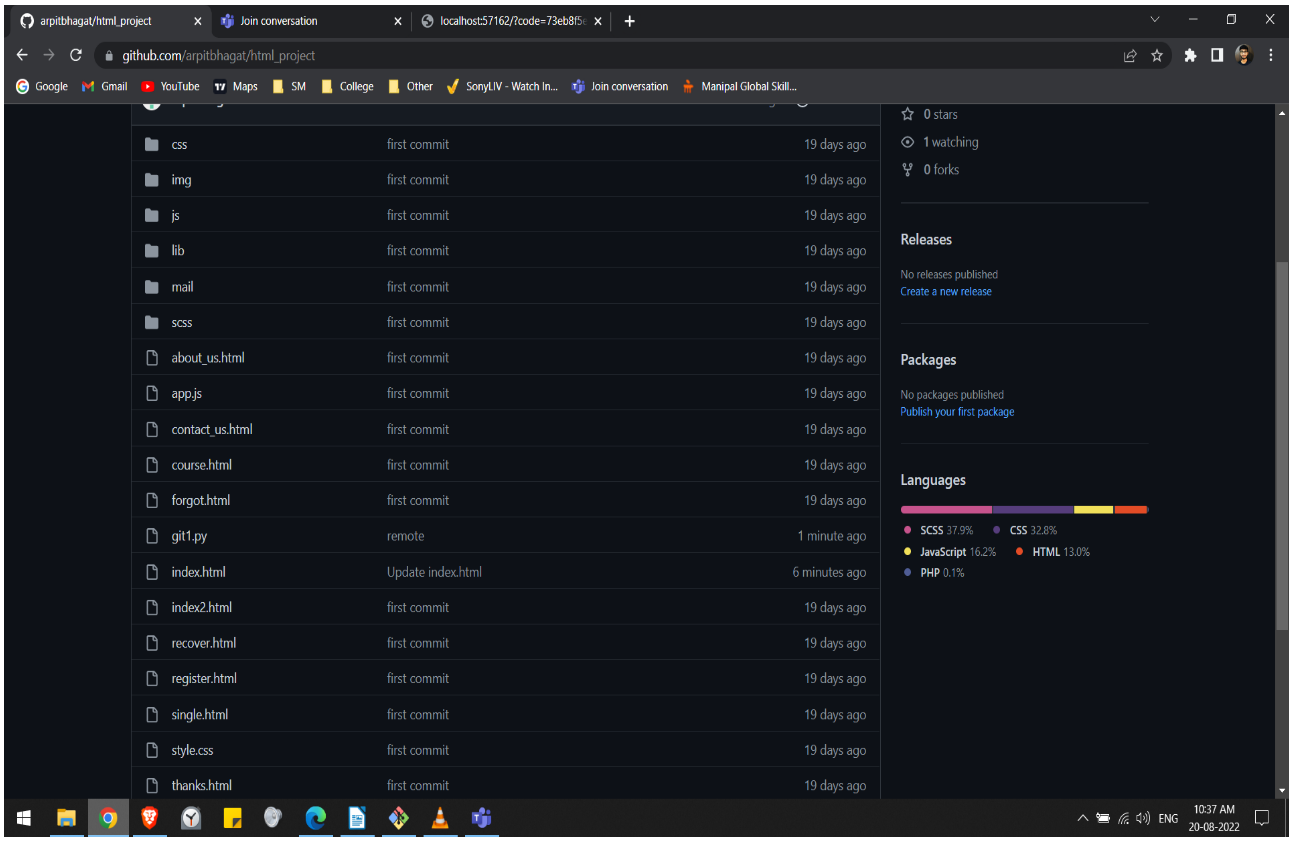
```
MINGW64:/c/Users/arpit/git-repo

arpit@LAPTOP-RRO23CB6 MINGW64 ~/git-repo
$ git clone https://github.com/arpitbhagat/html_p
roject.git
Cloning into 'html_project'...
remote: Enumerating objects: 178, done.
remote: Counting objects: 100% (178/178), done.
remote: Compressing objects: 100% (167/167), done
.
Receiving objects:  13% (24/178), 372.00 KiB | 68
Receiving objects:  14% (25/178), 372.00 KiB | 68
Receiving objects:  15% (27/178), 372.00 KiB | 68
Receiving objects:  16% (29/178), 372.00 KiB | 68
Receiving objects:  17% (31/178), 372.00 KiB | 68
Receiving objects:  18% (33/178), 372.00 KiB | 68
Receiving objects:  19% (34/178), 372.00 KiB | 68
Receiving objects:  20% (36/178), 372.00 KiB | 68
Receiving objects:  21% (38/178), 372.00 KiB | 68
Receiving objects:  22% (40/178), 372.00 KiB | 68
Receiving objects:  23% (41/178), 372.00 KiB | 68
Receiving objects:  24% (43/178), 372.00 KiB | 68
Receiving objects:  25% (45/178), 372.00 KiB | 68
Receiving objects:  26% (47/178), 372.00 KiB | 68
Receiving objects:  27% (49/178), 372.00 KiB | 68
Receiving objects:  28% (50/178), 372.00 KiB | 68
Receiving objects:  29% (52/178), 372.00 KiB | 68
Receiving objects:  30% (54/178), 372.00 KiB | 68
Receiving objects:  31% (56/178), 372.00 KiB | 68
Receiving objects:  32% (57/178), 372.00 KiB | 68
Receiving objects:  33% (59/178), 372.00 KiB | 68
Receiving objects:  34% (61/178), 372.00 KiB | 68
Receiving objects:  35% (63/178), 372.00 KiB | 68
Receiving objects:  36% (65/178), 372.00 KiB | 68
```

9) Git push

The git push command is used to upload local repository content to a remote repository. Pushing is how you transfer commits from your local repository to a remote repo. It's the counterpart to git fetch , but whereas fetching imports commits to local branches, pushing exports commits to remote branches.



```
MINGW64:/c/Users/arpit/git-repo/html_project

arpit@LAPTOP-RRO23CB6 MINGW64 ~/git-repo/html_project (
main)
$ vim git1.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/git-repo/html_project (
main)
$ cat git1.py
print("Git")

arpit@LAPTOP-RRO23CB6 MINGW64 ~/git-repo/html_project (
main)
$ git add .
warning: in the working copy of 'git1.py', LF will be r
eplaced by CRLF the next time Git touches it

arpit@LAPTOP-RRO23CB6 MINGW64 ~/git-repo/html_project (
main)
$ git commit -m "remote"
[main dce8bd3] remote
 1 file changed, 1 insertion(+)
 create mode 100644 git1.py

arpit@LAPTOP-RRO23CB6 MINGW64 ~/git-repo/html_project (
main)
$ git push origin main
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), 279 bytes | 279.00 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 https://github.com/arpitbhagat/html_project.git
   c2e8832..dce8bd3  main -> main

arpit@LAPTOP-RRO23CB6 MINGW64 ~/git-repo/html_project (
```



css	first commit	19 days ago
img	first commit	19 days ago
js	first commit	19 days ago
lib	first commit	19 days ago
mail	first commit	19 days ago
scss	first commit	19 days ago
about_us.html	first commit	19 days ago
app.js	first commit	19 days ago
contact_us.html	first commit	19 days ago
course.html	first commit	19 days ago
forgot.html	first commit	19 days ago
git1.py	remote	1 minute ago
index.html	Update index.html	6 minutes ago
index2.html	first commit	19 days ago
recover.html	first commit	19 days ago
register.html	first commit	19 days ago
single.html	first commit	19 days ago
style.css	first commit	19 days ago
thanks.html	first commit	19 days ago

0 stars
1 watching
0 forks

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

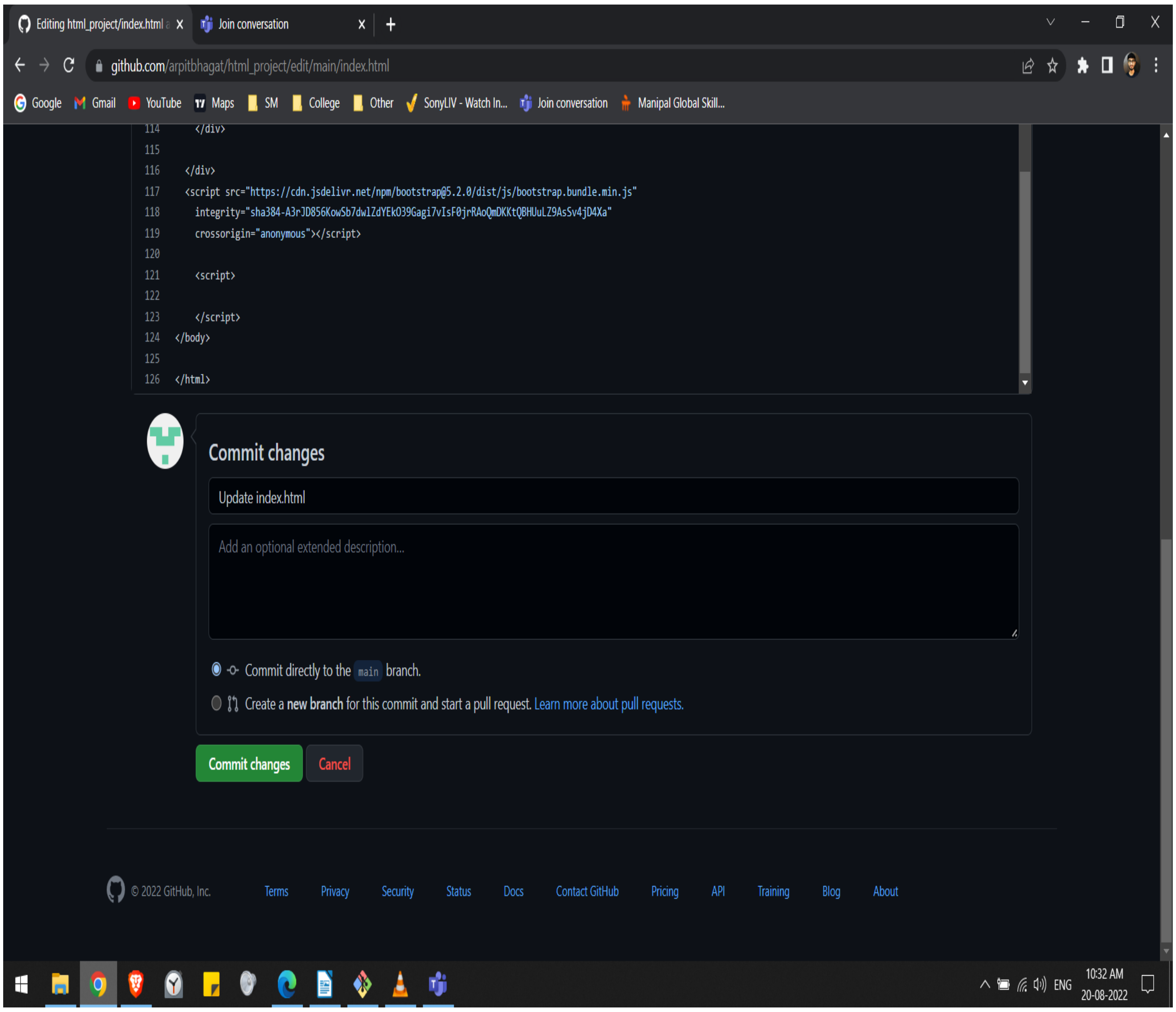
Languages

● SCSS 37.9% ● CSS 32.8%
● JavaScript 16.2% ● HTML 13.0%
● PHP 0.1%

GUI GitHub:

Using GUIthe Repository Creation, creating branch, comparing, editing can be done without the Command. Readymade Options are available. Along with the above options, Additional Option such as Code, Pull requests, issues, Actions, projects,Wiki, Insights,etc. Are avaialble

10) git commit through github



```
      </div>

    </div>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/js/bootstrap.bundle.min.js"
      integrity="sha384-A3rJD856KowSb7dwlZdYEk039Gagi7vIsF0jrRAoQmDKKtQBHUuLZ9AsSv4jD4Xa"
      crossorigin="anonymous"></script>

      <script>

      </script>
  </body>

  </html>
```

Commit changes

Update index.html

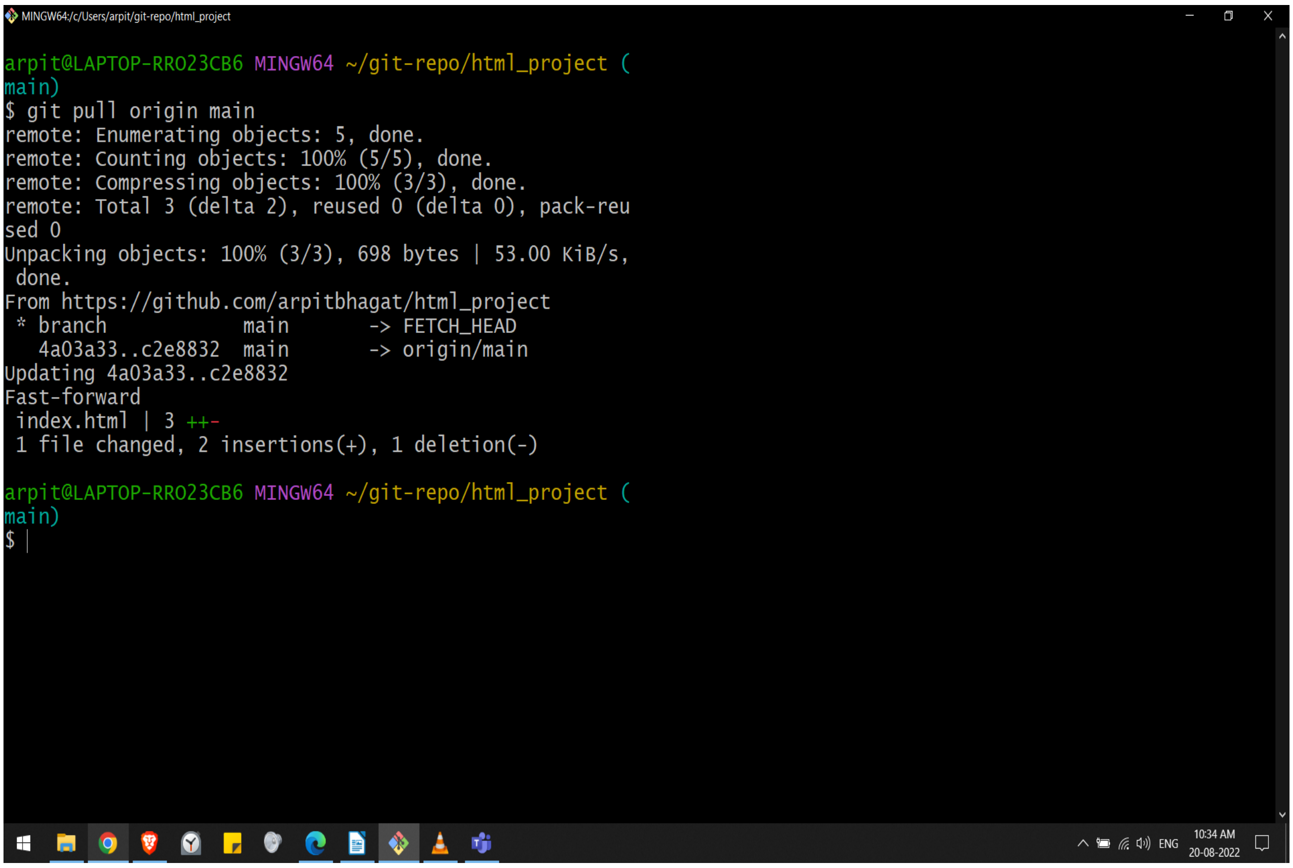
Add an optional extended description...

[✓ Commit directly to the main branch.]

Create a new branch for this commit and start a pull request. Learn more about pull requests.

11) git pulll

The git pull command is used to fetch and download content from a remote repository and immediately update the local repository to match that content. Merging remote upstream changes into your local repository is a common task in Git-based collaboration work flows.



```
arpit@LAPTOP-RRO23CB6 MINGW64 ~/git-repo/html_project (
main)
$ git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 2), reused 0 (delta 0), pack-reu
sed 0
Unpacking objects: 100% (3/3), 698 bytes | 53.00 KiB/s,
 done.
From https://github.com/arpitbhagat/html_project
 * branch            main       -> FETCH_HEAD
   4a03a33..c2e8832  main       -> origin/main
Updating 4a03a33..c2e8832
Fast-forward
 index.html | 3 ++-
 1 file changed, 2 insertions(+), 1 deletion(-)

arpit@LAPTOP-RRO23CB6 MINGW64 ~/git-repo/html_project (
main)
$ |
```

Conclusion:

In conclusion, Git provides a way of keeping track of past versions of software and papers, making collaboration between various authors easy, and provides backup for your software. It has proven very useful to the open-source community and in academia as well.