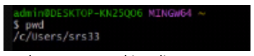
GIT & GITHUB

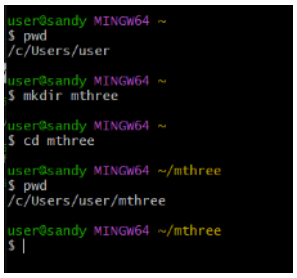
14 February 2025 09:49



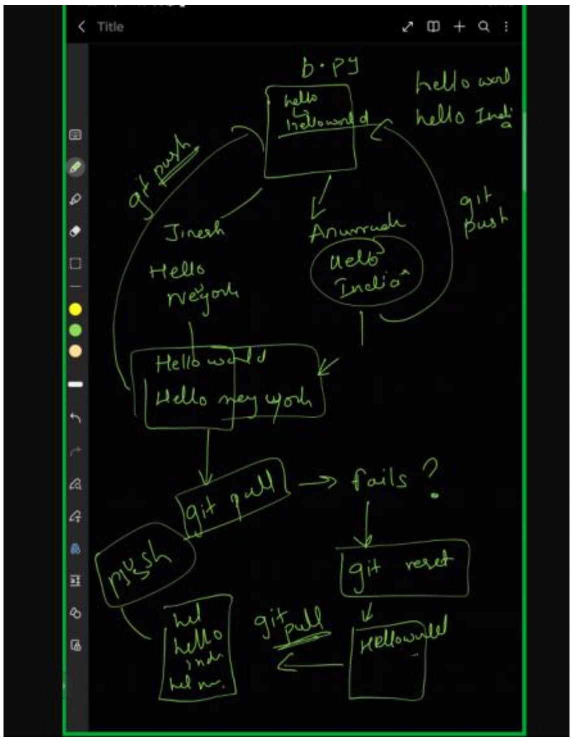
```
admin@DESKTOP-KN25Q06 MINGW64 ~
$ pwd
/c/Users/srs33
```

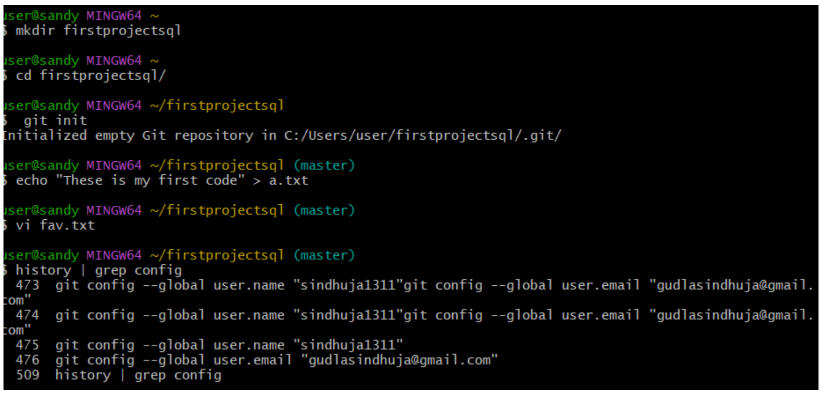
pwd - present working directory

mkdir - make directory



```
user@sandy MINGW64 ~
$ pwd
/c/Users/user

user@sandy MINGW64 ~
$ mkdir mthree

user@sandy MINGW64 ~
$ cd mthree

user@sandy MINGW64 ~/mthree
$ pwd
/c/Users/user/mthree

user@sandy MINGW64 ~/mthree
$ |
```

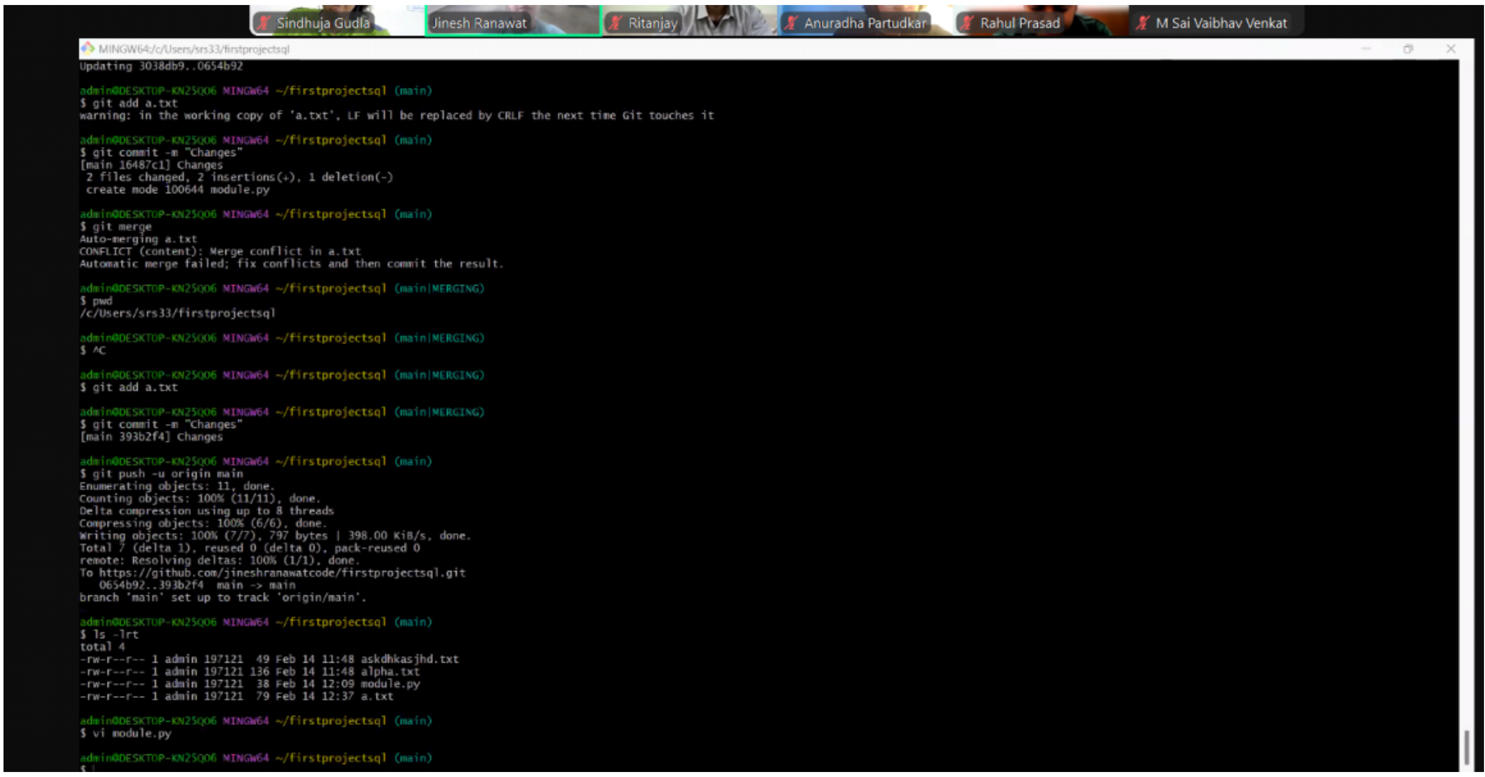




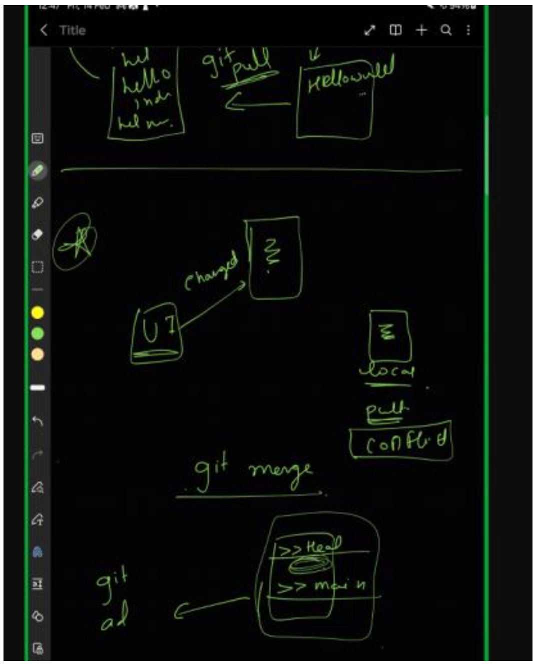
```
ser@sandy MINGW64 ~
$ mkdir firstprojectsql

ser@sandy MINGW64 ~
$ cd firstprojectsql/

ser@sandy MINGW64 ~/firstprojectsql
$  git init
nitialized empty Git repository in C:/Users/user/firstprojectsql/.git/

ser@sandy MINGW64 ~/firstprojectsql (master)
$ echo "These is my first code" > a.txt

ser@sandy MINGW64 ~/firstprojectsql (master)
$ vi fav.txt

ser@sandy MINGW64 ~/firstprojectsql (master)
$ history | grep config
  473  git config --global user.name "sindhuja1311"git config --global user.email "gudlasindhuja@gmail.
om"
  474  git config --global user.name "sindhuja1311"git config --global user.email "gudlasindhuja@gmail.
om"
  475  git config --global user.name "sindhuja1311"
  476  git config --global user.email "gudlasindhuja@gmail.com"
  509  history | grep config
```

Jinesh Ranawat 12:10

```
 173  git stash save "WIP: Intrest
calculation module"
  174  git add module.py
  175  git stash save "WIP: Intrest
calculation module"
  176  ls -lrt
  177  ls -lrt
  178  vi a.txt
  179  git add a.txt
  180  git commit -m "Bug fix
changes"
  181  git push -u origin main
  182  git stash list
  183  git stash shash@{0}
  184  git stash stash@{0}
  185  git stash show stash@{0}
  186  git checkout -b feature/intrest
  187  git stash pop
  188  ls -lrt
  189  history
```



```
Updating 3038db9..0654b92

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git add a.txt
warning: in the working copy of 'a.txt', LF will be replaced by CRLF the next time Git touches it

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git commit -m "Changes"
[main 16487c1] Changes
 2 files changed, 2 insertions(+), 1 deletion(-)
 create mode 100644 module.py

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git merge
Auto-merging a.txt
CONFLICT (content): Merge conflict in a.txt
Automatic merge failed; fix conflicts and then commit the result.

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main|MERGING)
$ pwd
/c/Users/srs33/firstprojectsql

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main|MERGING)
$ ^C

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main|MERGING)
$ git add a.txt

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main|MERGING)
$ git commit -m "Changes"
[main 393b2f4] Changes

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ git push -u origin main
Enumerating objects: 11, done.
Counting objects: 100% (11/11), done.
Delta compression using up to 8 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (7/7), 797 bytes | 398.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/jineshranawatcode/firstprojectsql.git
   0654b92..393b2f4  main -> main
branch 'main' set up to track 'origin/main'.

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ ls -lrt
total 4
-rw-r--r-- 1 admin 197121  49 Feb 14 11:48 askdhkasjhd.txt
-rw-r--r-- 1 admin 197121 136 Feb 14 11:48 alpha.txt
-rw-r--r-- 1 admin 197121  38 Feb 14 12:09 module.py
-rw-r--r-- 1 admin 197121  79 Feb 14 12:37 a.txt

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$ vi module.py

admin@DESKTOP-KN25Q06 MINGW64 ~/firstprojectsql (main)
$
```



Optimized Git Commands with Definitions

1. **git checkout feature/interest** – Switch to the feature/interest branch.
2. **vi a.txt** – Open a.txt in the vi editor for editing.
3. **git checkout main** – Switch back to the main branch.

4. **git merge feature/interest** – Merge feature/interest into main.
5. **git diff a.txt** – Show differences in a.txt between working directory and last commit.
6. **git pull origin main** – Fetch and merge the latest changes from the remote main branch.
7. **git add a.txt** – Stage changes in a.txt.
8. **git commit -m "Changes"** – Commit staged changes with a message.
9. **git push -u origin main** – Push changes to the remote main branch.
10. **vi module.py** – Open module.py for editing.
11. **git add module.py** – Stage module.py.
12. **git commit -m "Changes "** – Commit changes with a message.
13. **git pull** – Fetch latest changes from the remote branch.
14. **git merge** – Merge current branch with fetched changes.
15. **history** – Display command history.

Git Stash

Git **stash** is a feature that temporarily saves uncommitted changes, allowing you to switch branches or work on something else without committing. You can later retrieve and reapply the stashed changes.

- **git stash** → Saves uncommitted changes without committing, keeping the working directory clean.
- **git stash list** → Shows all stashed changes.
- **git stash apply** → Reapplies the most recent stash without removing it.
- **git stash pop** → Reapplies and removes the most recent stash.
- **git stash drop** → Deletes a specific stash.
- **git stash clear** → Removes all stashes.
- **git stash save "message"** → Saves stash with a custom message.
- **git stash show -p stash@{n}** → Shows changes in a specific stash.

In activity, we worked on a git repo as a collab where we worked on stashes and also pull and merge requests and merge conflicts

Merge Conflicts in Git

A **merge conflict** occurs when Git cannot automatically combine changes from different branches due to conflicting modifications in the same file.

Common Causes:

- Two branches modify the same line in a file.
- One branch deletes a file while another modifies it.

Resolving Conflicts:

1. **Identify Conflicts:**
   git status
2. **Edit the Conflicted File:**
   - Git marks conflicts like this:
     ```
     <<<<<<< HEAD
     Your changes
     =======
     Incoming changes
     >>>>>>> branch-name
     ```
   - Manually edit and keep the correct version.
3. **Stage and Commit:**
   git add <file>
   git commit -m "Resolved merge conflict"
4. **Continue Merge (if applicable):**
   git merge --continue

To **abort** the merge:
git merge --abort