**Git Commands Workflow:**

1. Create a local folder 🡪 mkdir Devops
2. cd Devops
3. add files 🡪 touch file1.txt file2.txt
4. git init
5. git status
6. git add .
7. git status
8. git commit -m “message”
9. git status
10. Synching to remote server (Create a Repo in Github)

* Git remote add origin “Repo URL”

1. Git push origin master
2. Git clone “Repo URL”

**Git branch Commands Workflow**

1. Git branch feature1 🡪 creating branch
2. Git branch 🡪 list the branches
3. Git branch -d feature1🡪 deleting branch
4. Git checkout feature1 🡪 switching to branch
5. Touch branchfile.txt
6. Git add .
7. Git commit -m “added to feature branch”
8. Git branch
9. Git checkout master
10. ls

Note: what ever we create in individual branches it will not reflect in another branch until you do a merge

1. git status
2. Git checkout feature1
3. Git status
4. Git push origin feature1
5. Git log 🡪 you can see the history

**Git Stashing example:**

1. Git checkout feature1
2. Vi stashfile.txt
3. Git status
4. git checkout master
5. ls
6. git checkout feature1
7. git add .
8. ls
9. git checkout master
10. ls
11. Git checkout feature1
12. Edit a common file between two branches
13. Git checkout master
14. Cat filename<common file you edited>

The above files will reflect in both the branches since you have not tracked those files

1. Git checkout feature1
2. Git stash
3. Git checkout master
4. Ls
5. Cat commonfile

**Now see the difference after stashing**

1. Git checkout feature1
2. Ls
3. Git stash pop
4. Ls
5. Cat commonfile
6. Git add .
7. Git commit -m “committing stashed changes”
8. Ls
9. Git status
10. Git checkout master
11. ls

**Git Revert workflow:**

Revert a commit to previous version

1. Git branch
2. Git log
3. Git revert commitID (better understanding select stash commit)
4. git checkout commitid
5. git checkout feature1
6. git diff commitid1 commitid2
7. git diff HEAD .