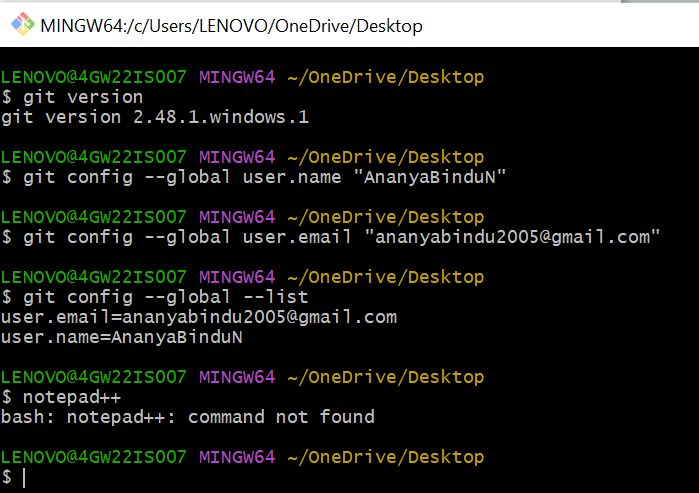
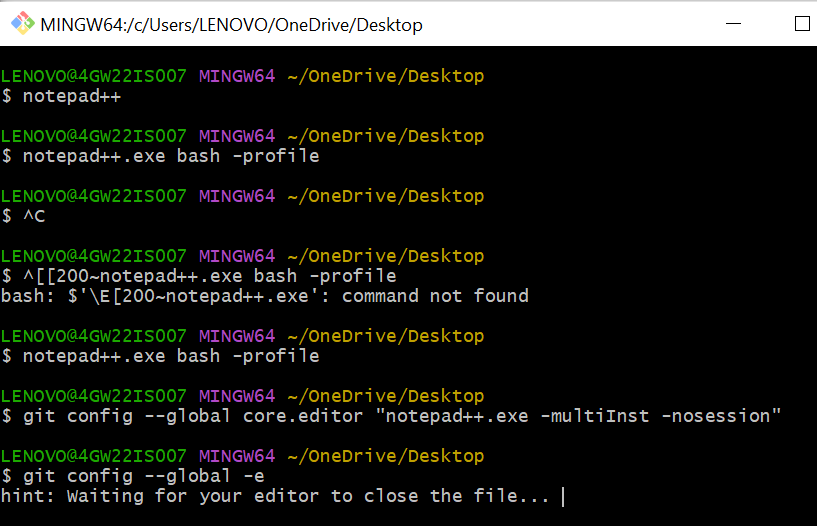
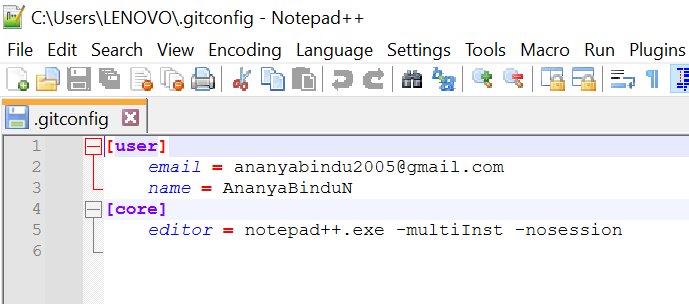
Week-8

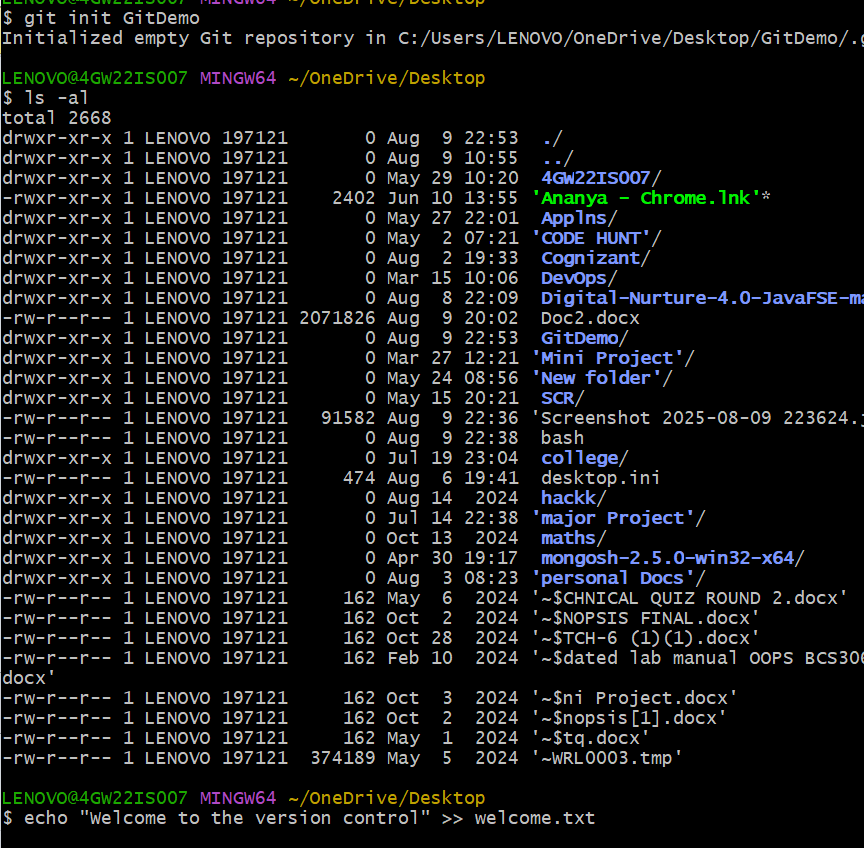
Skill name : GIT-HOL

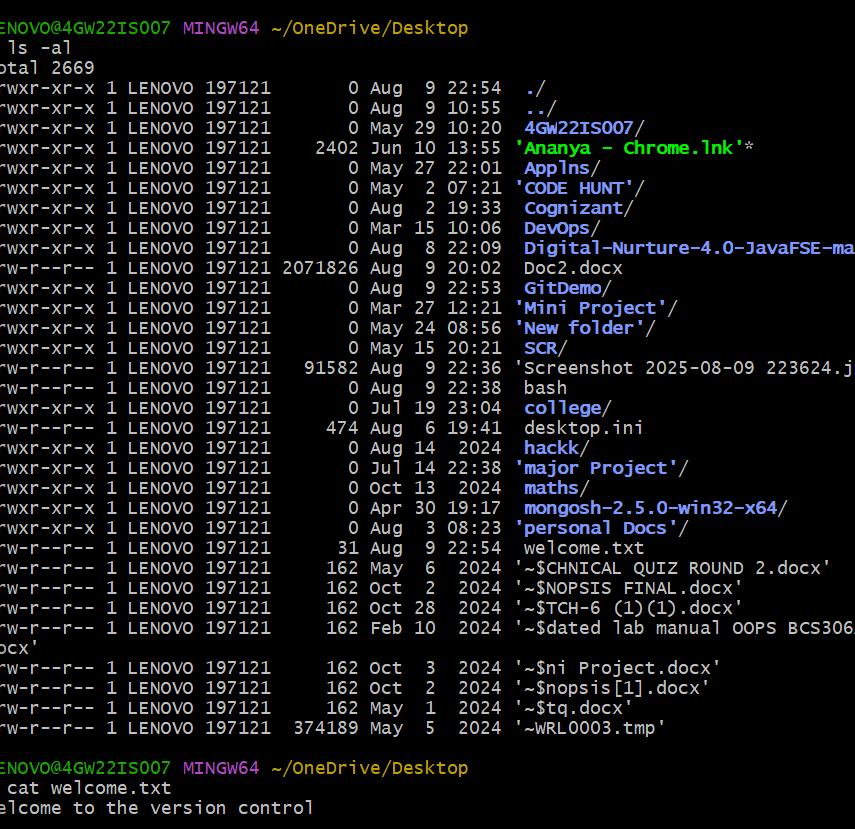
Hands-on 1: Familiar with Git commands like git init, git status, git add, git commit, git push, and git pull.





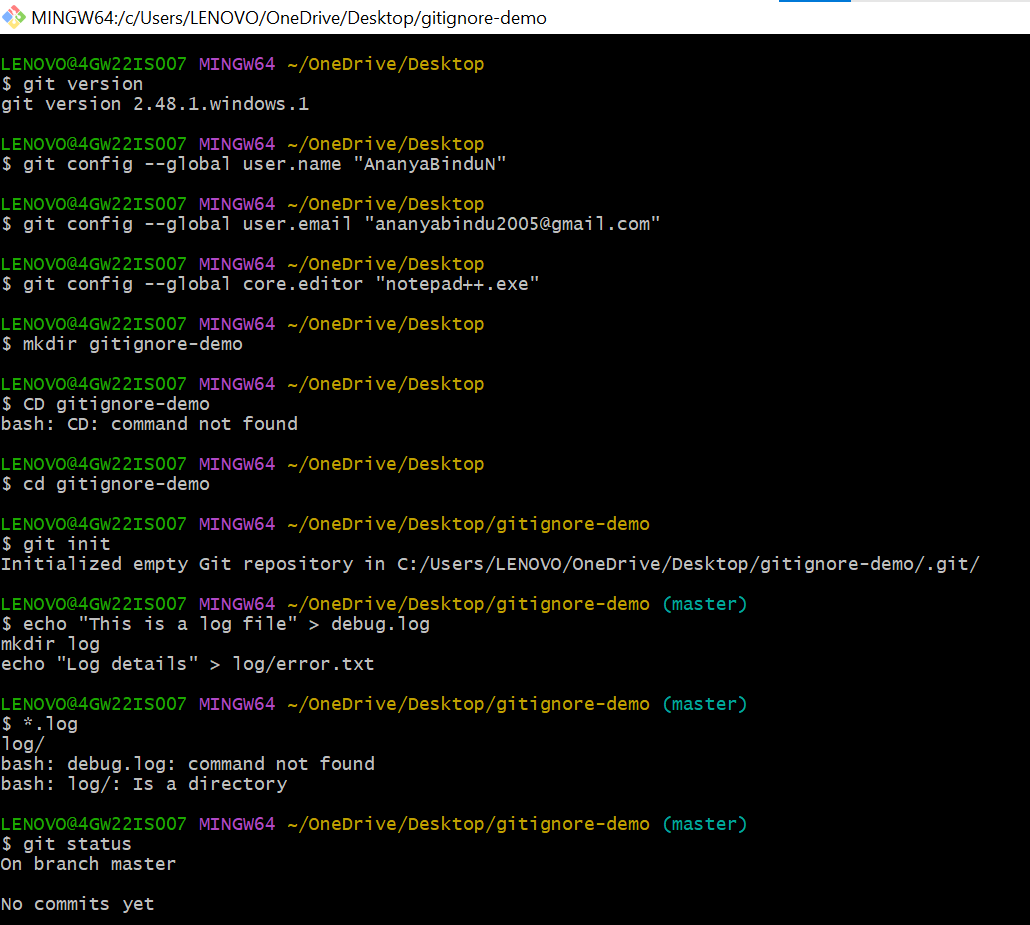


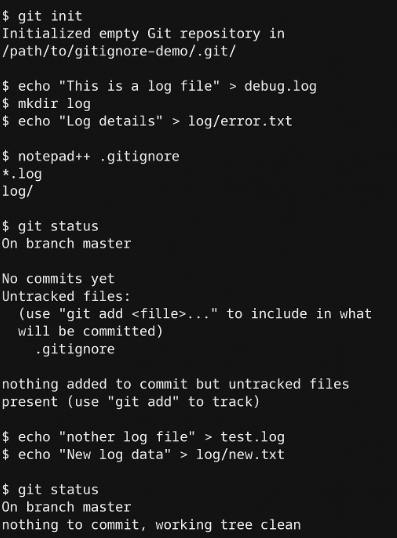




Hands-on 2:- Create a **“.log”** file and a **log folder** in the working directory of Git. Update the **.gitignore** file in such a way that on committing, these files (.log extensions and log folders) are ignored.

Verify if the git status reflects the same about working directory, local repository and git repository.





**3.Git HOL**

**Branching Steps**

**1. Create a new branch**

git branch GitNewBranch

**2. List all local & remote branches**

git branch -a

GitNewBranch

remotes/origin/master

**3. Switch to the newly created branch**

git checkout GitNewBranch

Output:

vbnet

Switched to branch 'GitNewBranch'

**4. Add some files with content**

echo "Hello from GitNewBranch" > newfile.txt

git add newfile.txt

**5. Commit the changes**

git commit -m "Added newfile.txt in GitNewBranch"

Output:

[GitNewBranch 123abc4] Added newfile.txt in GitNewBranch

1 file changed, 1 insertion(+)

create mode 100644 newfile.txt

**6. Check branch status**

git status

Output:

pgsql

On branch GitNewBranch

nothing to commit, working tree clean

**Merging Steps**

**1. Switch to master**

git checkout master

**2. List differences between master and branch**

git diff master GitNewBranch

This will show text differences in CLI.

**3. Visual differences with P4Merge**

If P4Merge is installed and configured:

git difftool master GitNewBranch

A visual diff tool will open showing file changes.

**4. Merge the branch into master**

git merge GitNewBranch

Output (if no conflicts):

nginx

Updating abc123..def456

Fast-forward

newfile.txt | 1 +

1 file changed, 1 insertion(+)

**5. Observe logging after merging**

git log --oneline --graph --decorate

Example:

markdown

\* def456 (HEAD -> master) Added newfile.txt in GitNewBranch

| \* 123abc4 (GitNewBranch) Added newfile.txt in GitNewBranch

|/

\* abc123 Initial commit

**6. Delete the branch after merging**

git branch -d GitNewBranch

Output:

java

Deleted branch GitNewBranch (was 123abc4).

**7. Check status**

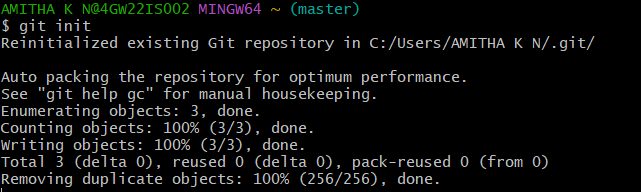
git status

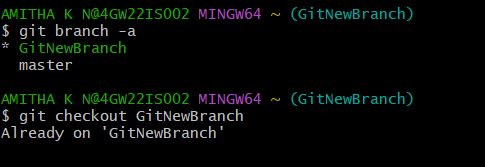
Output:

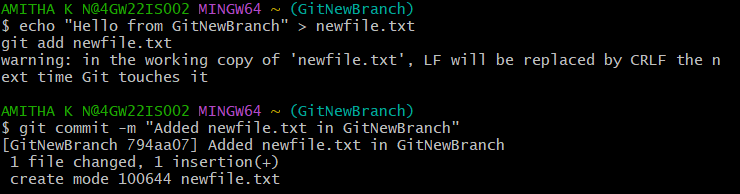
pgsql

On branch master

nothing to commit, working tree clean

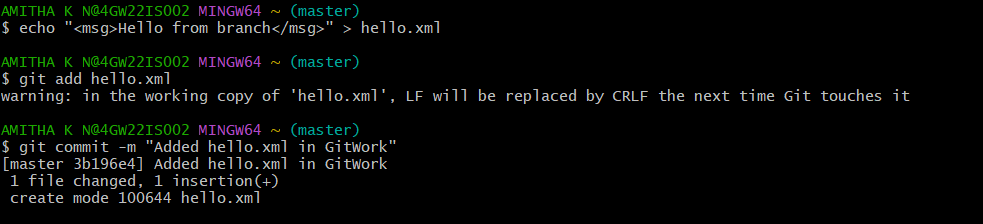


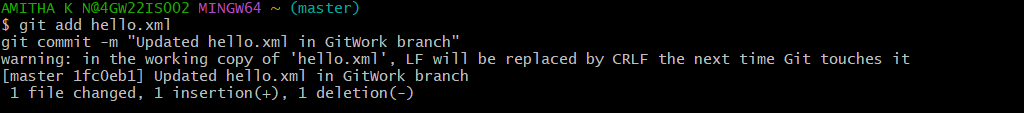




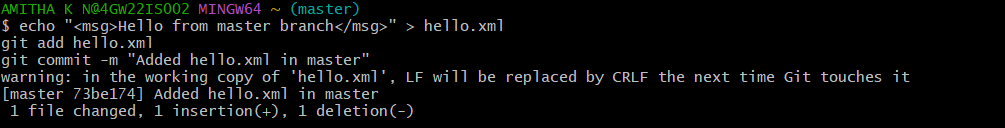
**4.Git HOL**

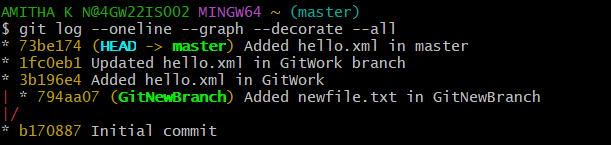
1. Verify if master is in clean state.
2. Create a branch **“GitWork”.** Add a file “hello.xml”.
3. Update the content of “hello.xml” and observe the status
4. Commit the changes to reflect in the branch
5. Switch to master.
6. Add a file **“hello.xml”** to the master and add some different content than previous.
7. Commit the changes to the master
8. Observe the log by executing **“git log –oneline –graph –decorate –all”**
9. Check the differences with Git diff tool
10. For better visualization, use P4Merge tool to list out all the differences between master and branch
11. Merge the bran to the master
12. Observe the git mark up.
13. Use 3-way merge tool to resolve the conflict
14. Commit the changes to the master, once done with conflict
15. Observe the git status and add backup file to the .gitignore file.
16. Commit the changes to the .gitignore
17. List out all the available branches
18. Delete the branch, which merge to master.
19. Observe the log by executing **“git log –oneline –graph –decorate”**











**5.Git HOL**

1. Verify if master is in clean state.
2. List out all the available branches.
3. Pull the remote git repository to the master
4. Push the changes, which are pending from “Git-T03-HOL\_002” to the remote repository.
5. Observe if the changes are reflected in the remote repository.

