**WEEK-8**

**GIT HANDS ON**

**Superset ID: 6419740**

**EXERCISE 1:**

Step 1: Check Git Installation

Open Git Bash and run:

git –version

Step 2: Configure Git User Details

Set your name:

git config --global user.name "DarwinPrabu"

Set your email:

git config --global user.email "winddarwin27@gmail.com"

Verify configuration:

git config –list

Step 3: Set Notepad++ as Default Git Editor

If Notepad++ is not recognized, add its path to Environment Variables. Then run:

git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession"

Verify:

git config --global core.editor

Step 4: Create Repository and Push File

Initialize repository:

git init

Link to GitHub:

git remote add origin https://github.com/DarwinPrabu/GitDemo.git

Create a file:

echo "Hello Git" > file1.txt

Stage the file:

git add file1.txt

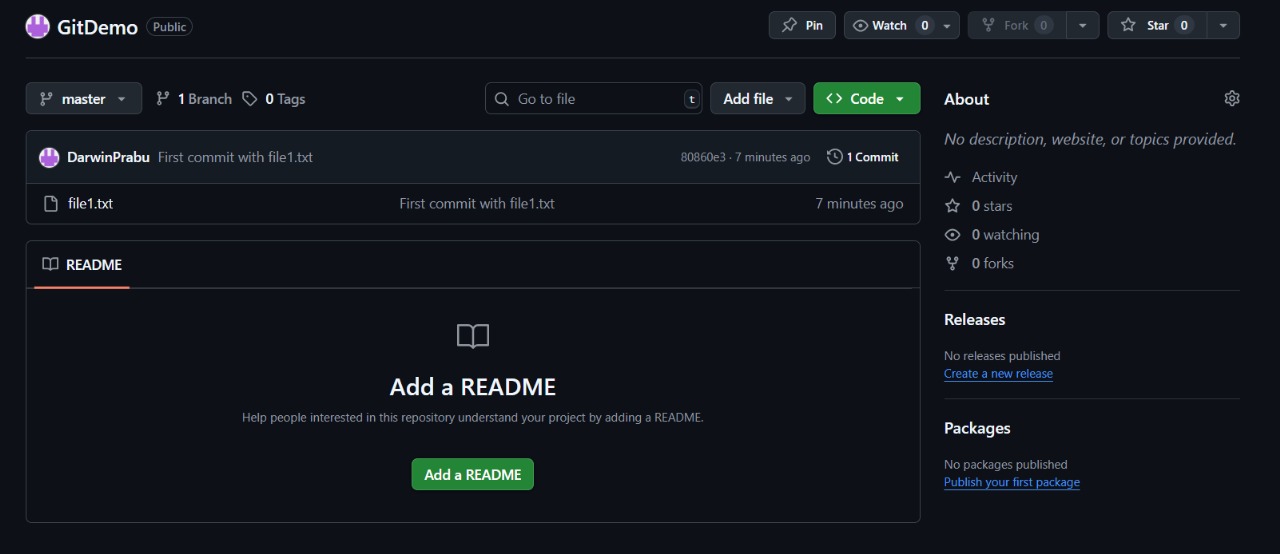
Commit changes:

git commit -m "First commit with file1.txt"

Push to GitHub:

git push -u origin master

**OUTPUT:**

****

**EXERCISE 2:**

**Step 1: Ensure Git is initialized**

git init

**Step 2: Create unwanted files and folder**

echo This is a log file > debug.log

mkdir log

echo This is inside the log folder > log\test.txt

**Step 3: Create .gitignore file**

echo \*.log > .gitignore

echo log/ >> .gitignore

**Step 4: Stage files (except ignored ones)**

git add .

**Step 5: Commit changes**

git commit -m "Added .gitignore to exclude .log files and log folder"

**step 6:Add Remote and Push**

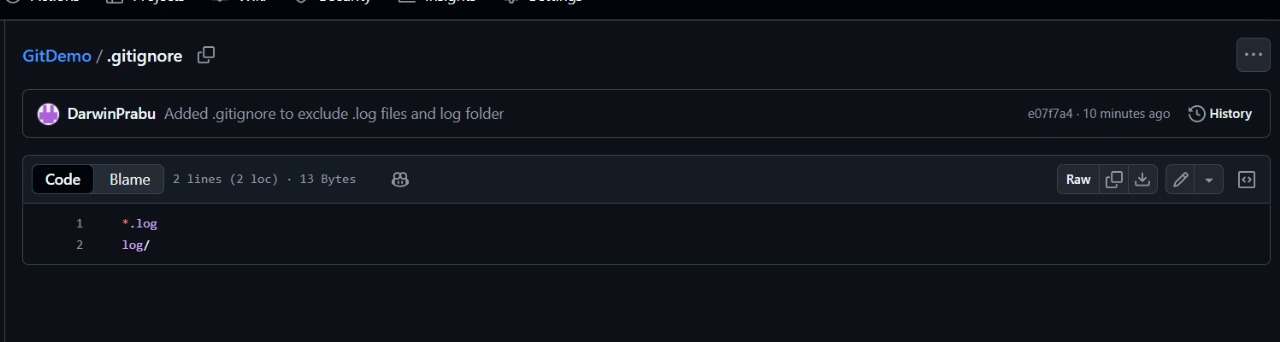
git remote add origin https://github.com/DarwinPrabu/GitDemo.git

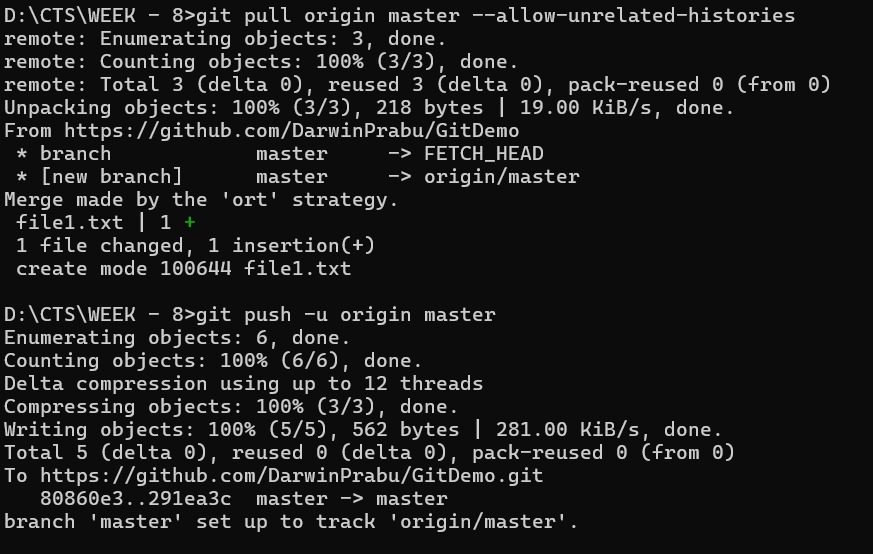
**Step 7: Push to GitHub**

git pull origin master --allow-unrelated-histories

git push -u origin master

**OUTPUT:**





**EXERCUISE 3:**

**Branching**

1. Create a New Branch

git branch GitNewBranch

2. List All Branches (Local & Remote)

git branch -a

* The \* shows your current branch.

3. Switch to the New Branch

git checkout GitNewBranch

4. Add a New File & Add Content

echo "This is content for the new branch" > branchfile.txt

5. Stage and Commit Changes

git add branchfile.txt

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

6. Check Status

git status

Merging

1. Switch Back to Master

git checkout master

2. View Differences Between Master & Branch (CLI)

git diff master GitNewBranch

3. View Differences Using P4Merge (if installed)

git difftool master GitNewBranch

4. Merge Branch into Master

git merge GitNewBranch

5. View Merge History

git log --oneline --graph --decorate

6. Delete the Branch After Merging

git branch -d GitNewBranch

7. Final Status Check

git status

**Branching Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

**A computer screen shot of a black screen

AI-generated content may be incorrect.Merging Output:**

**EXERCISE 4:**

1. Go to your repository directory

*cd D:\CTS\WEEK-8\Ex4*

2. Verify if master is in clean state

*git checkout master*

*git status*

3. Create a branch GitWork and switch to it

*git checkout -b GitWork*

4. Add a file hello.xml

*echo "<message>Hello from GitWork branch</message>" > hello.xml*

5. Check status

*git status*

6. Stage and commit changes

*git add hello.xml*

*git commit -m "Add hello.xml in GitWork branch"*

7. Switch to master

*git checkout master*

8. Add a file hello.xml in master with different content

*echo "<message>Hello from Master branch</message>" > hello.xml*

9. Stage and commit changes

*git add hello.xml*

*git commit -m "Add hello.xml in Master branch"*

10. View commit history (all branches)

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

11. View CLI differences

*git diff master GitWork*

12. View differences using P4Merge

*git difftool master GitWork*

13. Merge branch into master (this will cause conflict)

*git merge GitWork*

14. Resolve the conflict

* Open hello.xml in your editor (e.g., Notepad++).
* You’ll see conflict markers like:

<<<<<<< HEAD

<message>Hello from Master branch</message>

=======

<message>Hello from GitWork branch</message>

>>>>>>> GitWork

* Edit the file to keep the correct final version.
* Save the file.

15. Mark the conflict as resolved

*git add hello.xml*

16. Commit the merge

*git commit -m "Resolved merge conflict in hello.xml"*

17. Add backup file to .gitignore

*echo "\*.bak" >> .gitignore*

18. Stage and commit .gitignore

*git add .gitignore*

*git commit -m "Add .bak files to .gitignore"*

19. List all branches

*git branch*

20. Delete the merged branch

*git branch -d GitWork*

21. View final commit graph

*git log --oneline --graph –****decorate***

**A screenshot of a computer program

AI-generated content may be incorrect.Output:**

**A screenshot of a computer program

AI-generated content may be incorrect.**

**EXERCISE 5:**

1. Navigate to your working directory

*cd "D:/CTS/WEEK – 8/Ex5"*

2. Verify if master is in clean state

*git checkout master*

*git status*

3. List all branches

*git branch -a*

4. Pull latest changes from remote master

*git pull origin master*

5. Push pending changes to remote

*git push origin master*

**A computer screen shot of a black screen

AI-generated content may be incorrect.A screenshot of a computer screen

AI-generated content may be incorrect.OUTPUT:**

**SExercise 1 to 5:**

**A screenshot of a computer

AI-generated content may be incorrect.**