# HandsOn 1:

## Step 1: Setup your machine with Git Configuration

1. Install Git Bash from https://git-scm.com/ and complete the installation.  
2. Open Git Bash and verify installation:  
 git --version  
3. Configure your Git username and email:  
 git config --global user.name "Your Name"  
 git config --global user.email "your.email@example.com"  
4. Verify configuration:  
 git config --global --list

## Step 2: Integrate Notepad++ as Default Git Editor

1. Download and install Notepad++ from https://notepad-plus-plus.org/.  
2. Add Notepad++ installation path (e.g., C:/Program Files/Notepad++) to your system environment PATH.  
3. Configure Git to use Notepad++:  
 git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession -wait"  
4. Verify editor configuration:  
 git config --global -e

## Step 3: Create a New Repository Locally

1. Navigate to your preferred drive (example: D drive):  
 cd /d  
2. Create and enter a new project folder:  
 mkdir GitDemo  
 cd GitDemo  
3. Initialize a new Git repository:  
 git init

## Step 4: Create and Commit a File

1. Create a new file with sample content:  
 echo "Welcome to Git!> welcome.txt  
2. Check file status:  
 git status  
3. Stage the file:  
 git add welcome.txt  
4. Commit the file:  
 git commit -m "Added welcome.txt with welcome message"

## Step 5: Create Remote Repository on GitHub

1. Login to your GitHub account.  
2. Click 'New repository'.  
3. Name it 'GitDemo', keep it Public, and click 'Create repository'.  
4. Copy the repository URL (HTTPS).

## Step 6: Connect Local Repo to Remote Repo

1. In Git Bash, link the remote repository:  
 git remote add origin https://github.com/<your-username>/GitDemo.git  
2. Rename the branch to main:  
 git branch -M main

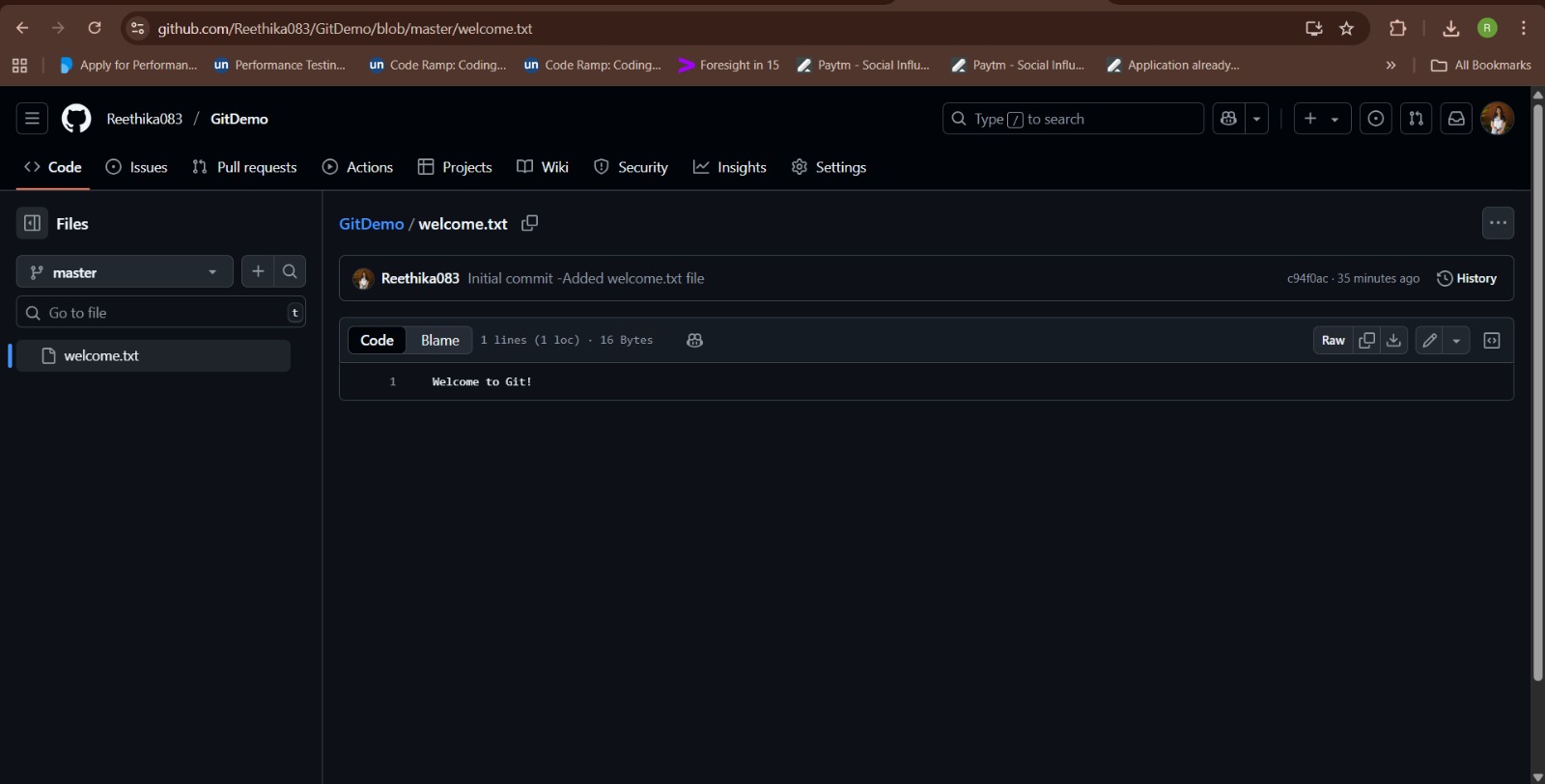
## Step 7: Setup GitHub Authentication via Personal Access Token

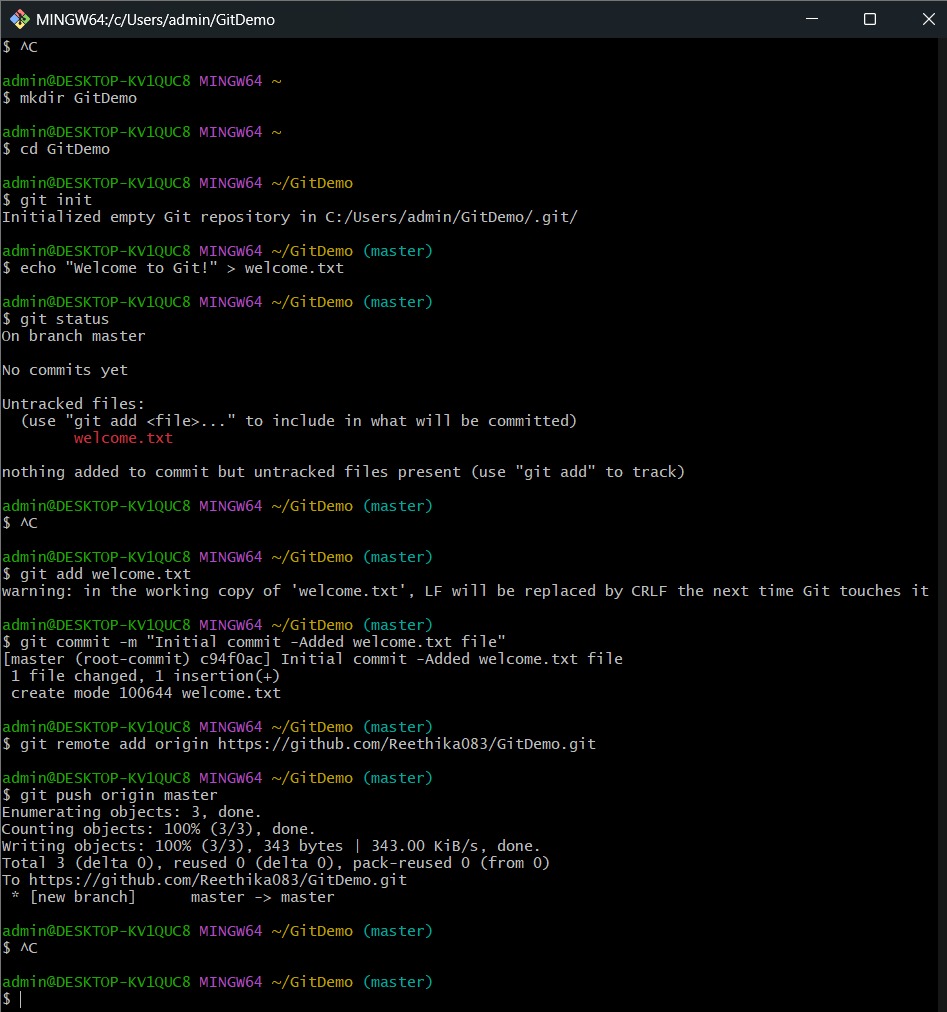
1. On GitHub, go to Settings > Developer settings > Personal Access Tokens > Tokens (classic).  
2. Click 'Generate new token (classic)'.  
3. Select 'repo' scope and generate token.  
4. Copy the token.  
5. Back in Git Bash, configure credential storage:  
 git config --global credential.helper store  
6. Push to GitHub:  
 git push -u origin main  
7. When prompted:  
 Username: your GitHub username  
 Password: paste the Personal Access Token

## Step 8: Verify Everything

1.Run git status — should show 'nothing to commit, working tree clean'.  
2. Run git log — should show your commit message.  
3. On GitHub, open the GitDemo repository and confirm that welcome.txt is present.

**OUTPUT:**

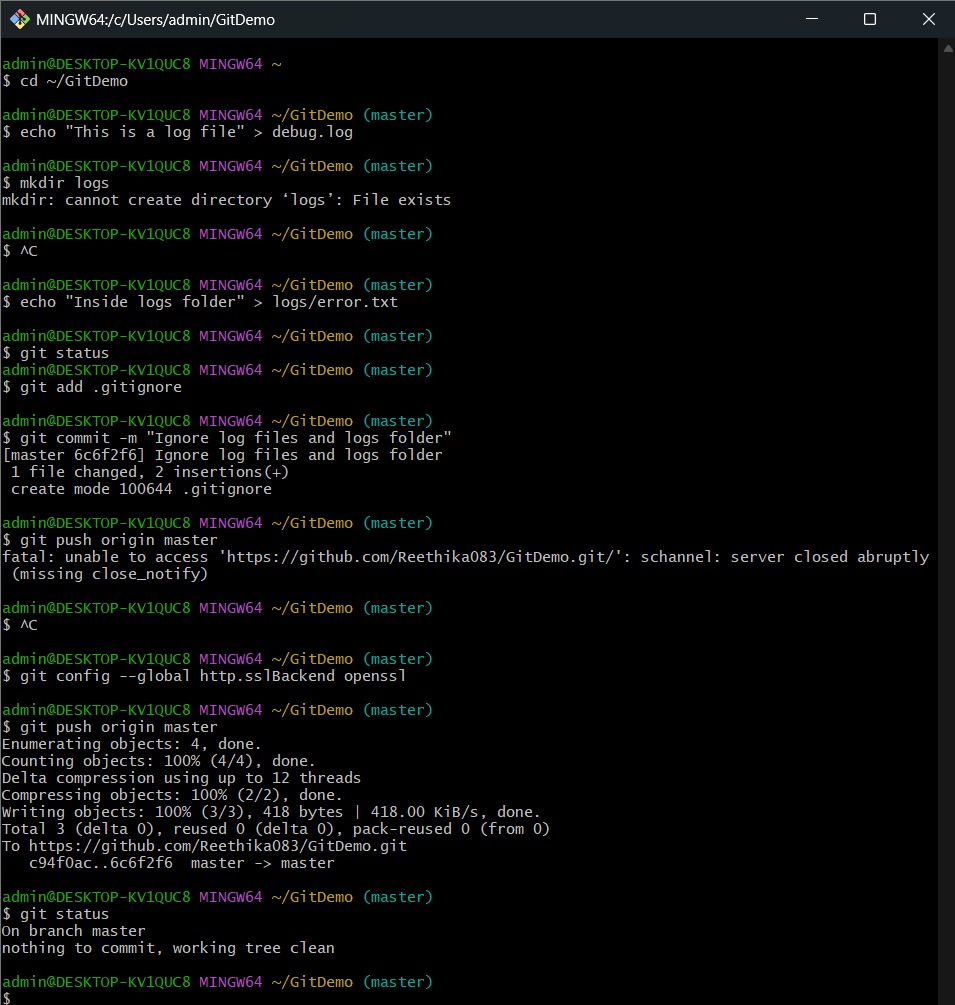




## HandsOn2:

1. cd ~/GitDemo
2. echo "This is a log file" > debug.log
3. mkdir logs
4. echo "Inside logs folder" > logs/error.txt
5. notepad .gitignore
6. # Inside .gitignore, add:
7. # \*.log
8. # logs/
9. git status
10. git add .gitignore
11. git commit -m "Ignore log files and logs folder"
12. git push origin master

**OUTPUT:**



**HandsOn3:**

**Branching:**

**1. Create a new branch named 'GitNewBranch':**

**git branch GitNewBranch**

**2. List all the local and remote branches:**

**git branch -a**

**(Observe the '\*' mark indicating the current branch)**

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

**git checkout GitNewBranch**

**4. Add a new file to this branch:**

**echo "This is a file in GitNewBranch" > branchfile.txt**

**5. Stage the new file:**

**git add branchfile.txt**

**6. Commit the changes to the branch:**

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

**7. Check the status of the branch:**

**git status**

**Merging:**

**1. Switch back to the master branch:**

**git checkout master**

**2. List out all the differences between master and GitNewBranch:**

**git diff master GitNewBranch**

**3. (Optional) View visual differences using P4Merge tool.**

**4. Merge the source branch into master:**

**git merge GitNewBranch**

**5. View the commit history after merging:**

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

**6. Delete the branch after merging:**

**git branch -d GitNewBranch**

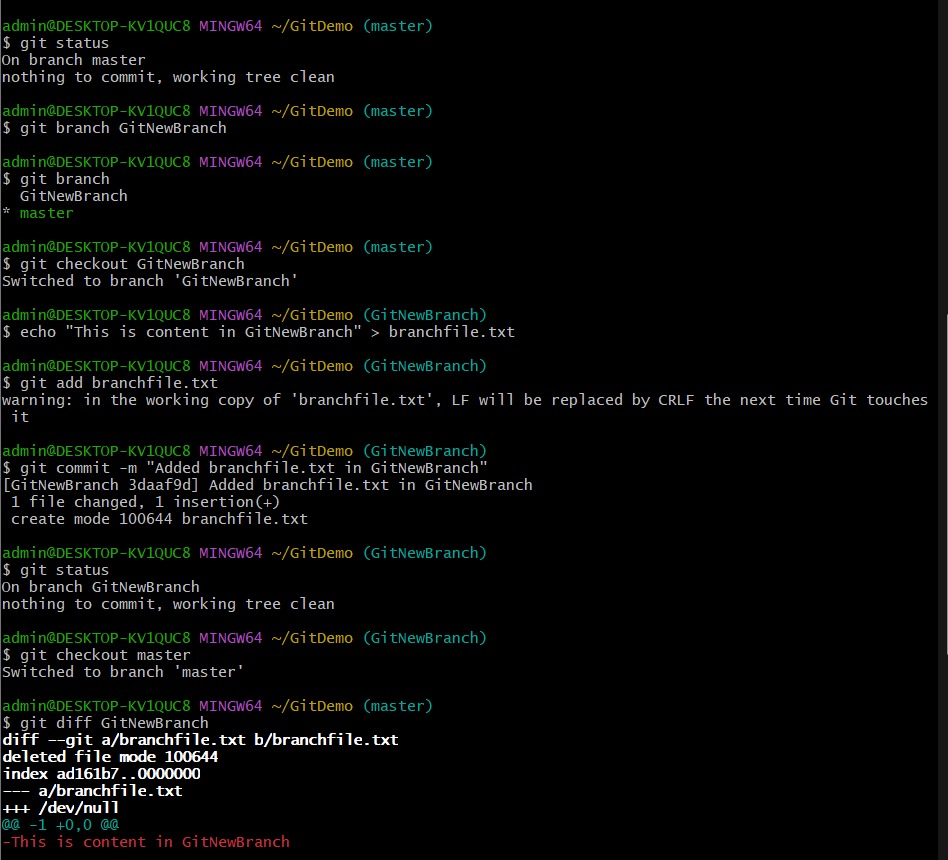
**7. Push the updated master to the remote repository:**

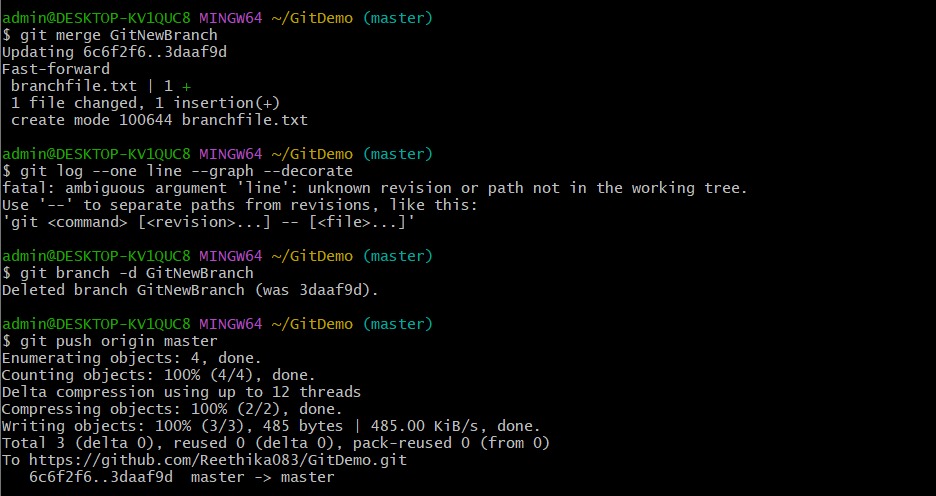
**git push origin master**

**8. Check the final status:**

**git status**

**OUTPUT:**





# HandsOn 4:

**1. Verify master is in clean state**

git checkout master

git status

**2. Create a branch “GitWork”**

git branch GitWork

git checkout GitWork

**3. Add a file “hello.xml”**

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

**4. Check status**

git status

**5. Commit changes in branch**

git add hello.xml

git commit -m "Added hello.xml in GitWork branch"

**6. Switch to master**

git checkout master

**7. Add hello.xml with different content**

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

**8. Commit changes in master**

git add hello.xml

git commit -m "Added hello.xml in master branch"

**9. Observe log**

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

**10. Check differences**

git diff master GitWork

# (Optional) View differences using P4Merge

# p4merge hello.xml

**11. Merge branch into master**

git merge GitWork

**12. Git will show a conflict — open hello.xml**

# You will see conflict markers <<<<<<<, =======, >>>>>>>

**13. Use a 3-way merge tool or manually edit hello.xml to resolve conflict**

**14. After resolving, stage the file**

git add hello.xml

**15. Commit merge**

git commit -m "Resolved merge conflict between master and GitWork"

**16. Add backup file to .gitignore (if created by merge tool)**

echo "\*.bak" >> .gitignore

git add .gitignore

git commit -m "Added backup files to .gitignore"

**17. List branches**

git branch

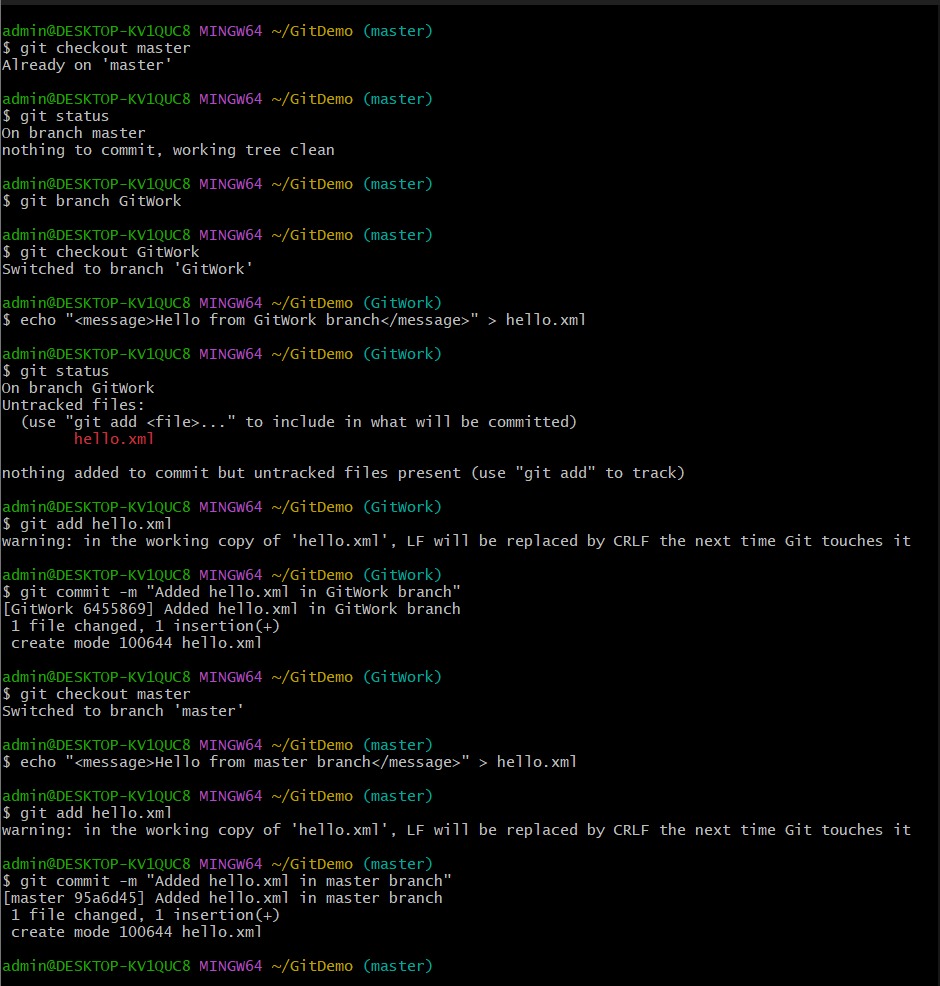
**18. Delete merged branch**

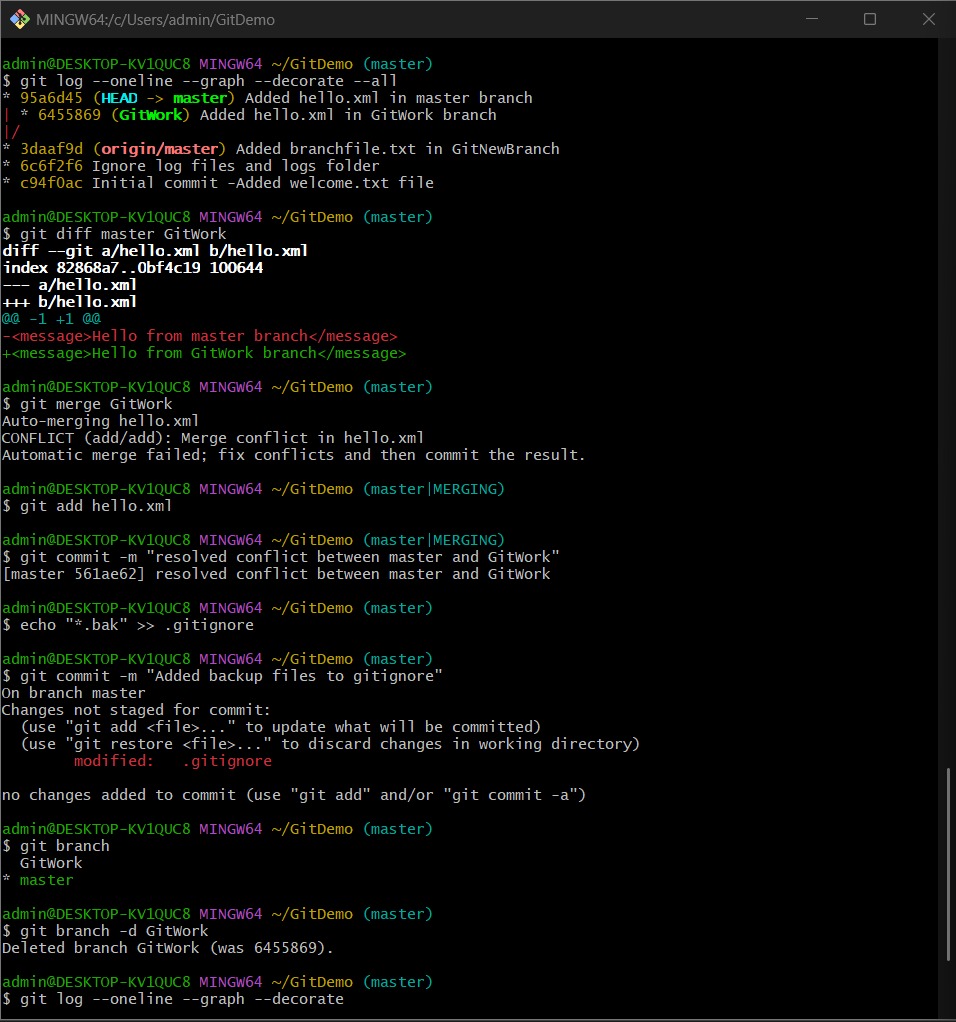
git branch -d GitWork

**19. Observe log after merge**

git log --oneline --graph –decorate

**OUTPUT:**







**HandsOn 5:**

**1. Verify if master is in a clean state**

git checkout master

git status

**2. List all available branches**

git branch -a

**3. Pull the remote repository into master**

git pull origin master

**4. Push any pending changes from local master to remote**

git push origin master

**5. Open your GitHub repository in the browser to verify that changes are reflected**

**OUTPUT:**

