**Week 8: GIT**

**HandsOn 1:**

**Objectives**

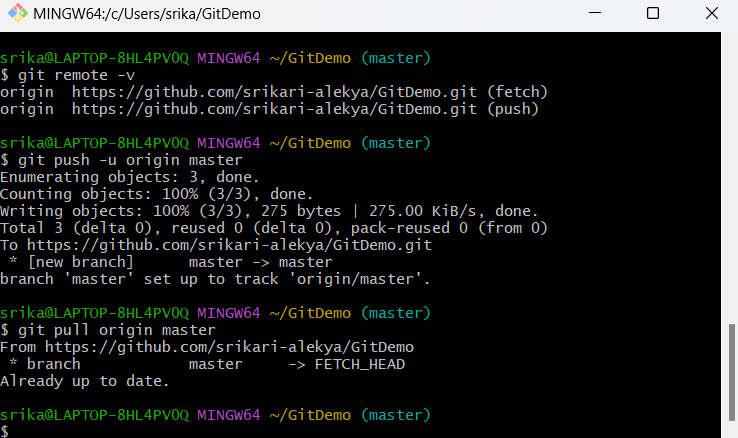
Familiar with Git commands like git init, git status, git add, git commit, git push, and git pull.

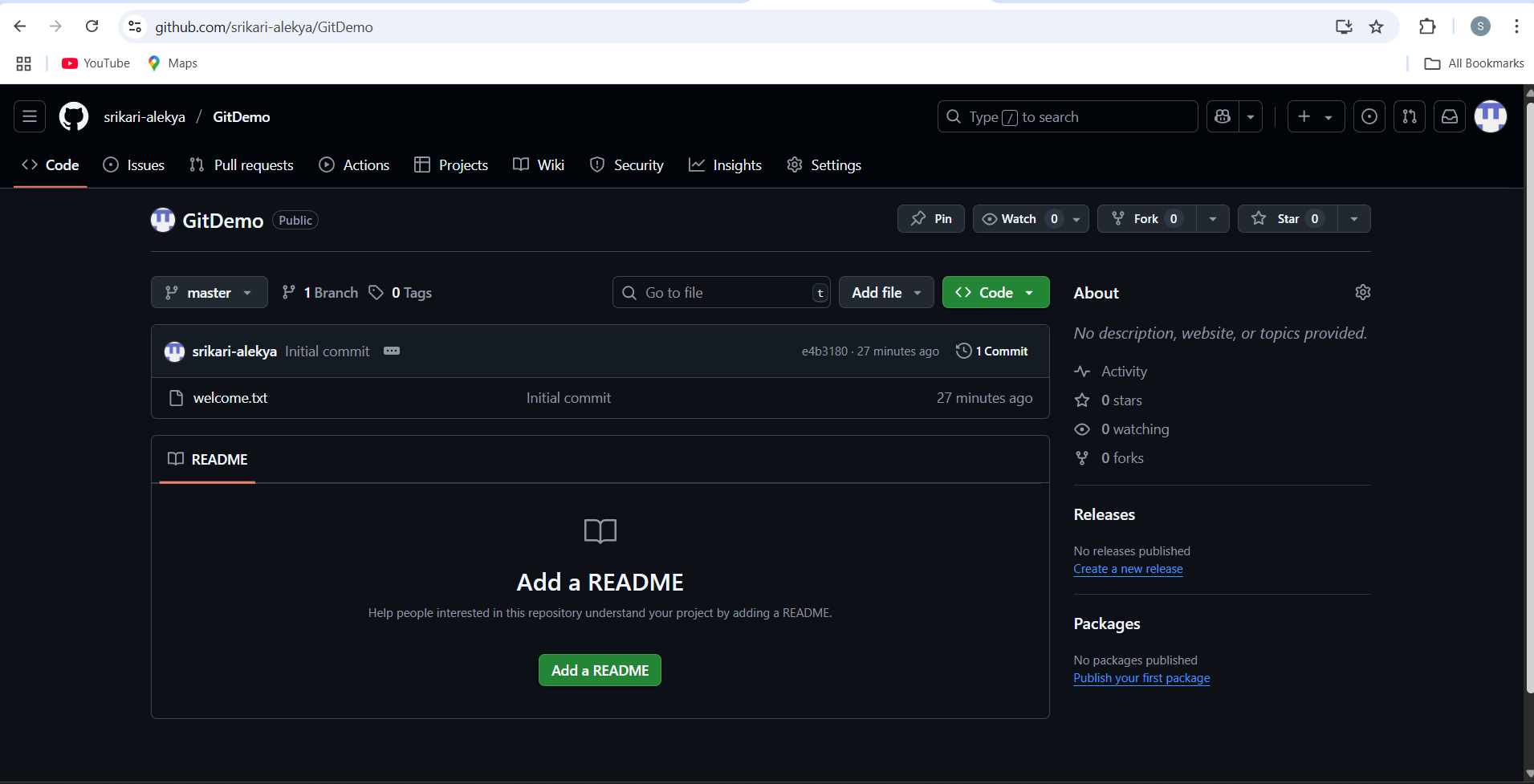
In this hands-on lab, you will learn how to

* Setup your machine with Git Configuration
* Integrate notepad++.exe to Git and make it a default editor
* Add a file to source code repository

**Steps:**

1. **Installed** Git & Git Bash and verified installation with git --version.
2. **Installed** Notepad++ as required for Git editor setup.
3. **Configured** system PATH to include Notepad++ for Git Bash access.
4. **Configured** Git with username and email using git config --global.
5. **Configured** Notepad++ as the default Git editor via core.editor setting.
6. **Created** a local Git repository (GitDemo) using git init.
7. **Created** welcome.txt file with welcome message.
8. **Staged & committed** welcome.txt to the local repository using Git and Notepad++.
9. **Created** a remote repository (GitDemo) on GitHub.
10. **Linked** the local repository to the remote using git remote add origin.
11. **Pushed** the local repository to the remote branch (master).
12. **Pulled** updates from the remote repository using git pull.





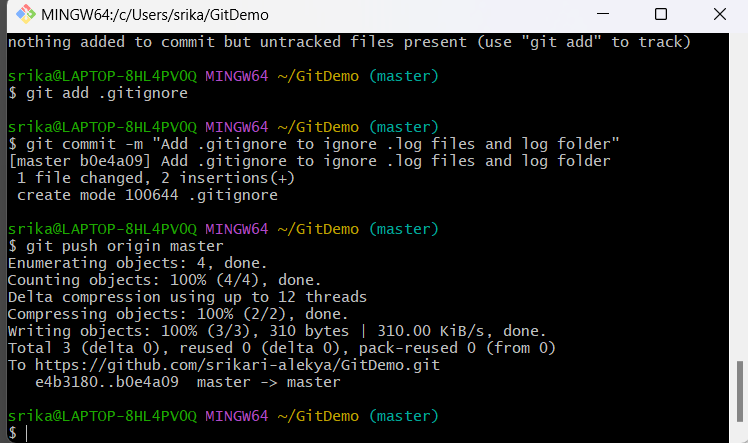
**HandsOn 2:**

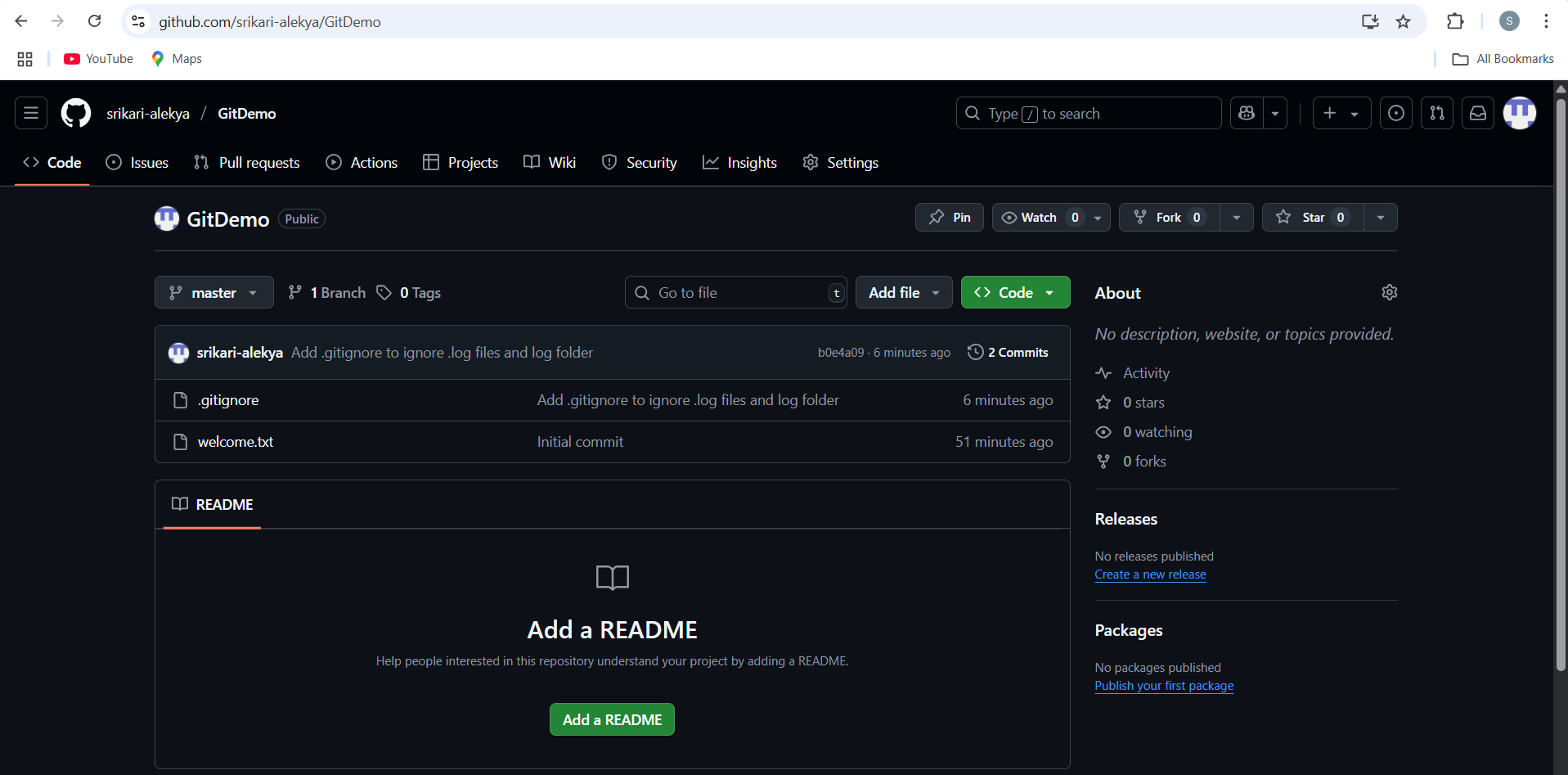
Create a **“.log”** file and a **log folder** in the working directory of Git. Update the **.gitignore** file in such a way that on committing, these files (.log extensions and log folders) are ignored.

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

**Steps**

1. Checked current directory
2. Created a .log file
3. echo "This is a sample log file" > error.log
4. Created a log folder and a .log file inside it
5. mkdir log
6. echo "Log file content" > log/sample.log
7. Created a .gitignore file
8. touch .gitignore
9. Opened .gitignore in Notepad++ and added ignore rules
   * Added:
   * \*.log
   * log/
10. Saved and closed .gitignore
11. Checked the status to verify ignored files
12. git status
    * .log files and log folder do not appear in tracked changes.
13. Stage .gitignore for commit
14. git add .gitignore
15. Commited the changes
16. git commit -m "Add .gitignore to ignore .log files and log folder"
17. Pushed to the remote repository
18. git push origin master



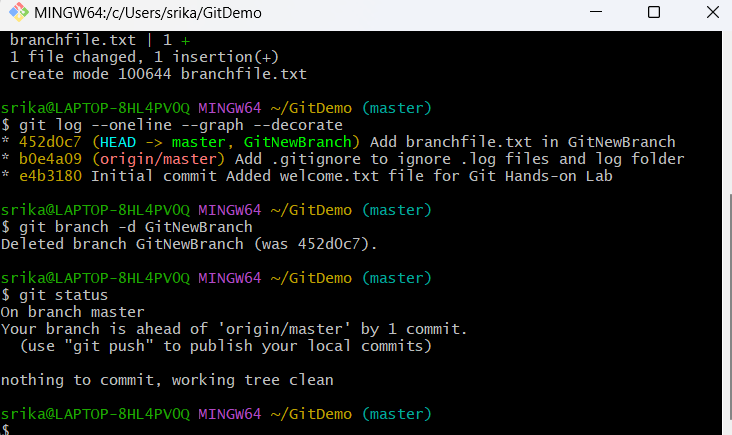


**HandsOn 3:**

Construct a branch, do some changes in the branch, and merge it with master (or trunk)

**Steps:**

1. Created a new branch named GitNewBranch from the master branch.
2. Added a new file with some sample content in the new branch.
3. Committed the changes in the branch.
4. Switched back to the master branch.
5. Compared differences between master and GitNewBranch using:
   * Command line (git diff)
   * P4Merge visual tool.
6. Merged GitNewBranch into master locally.
7. Verified merge success using git log --oneline --graph --decorate.
8. Deleted the GitNewBranch after merge.

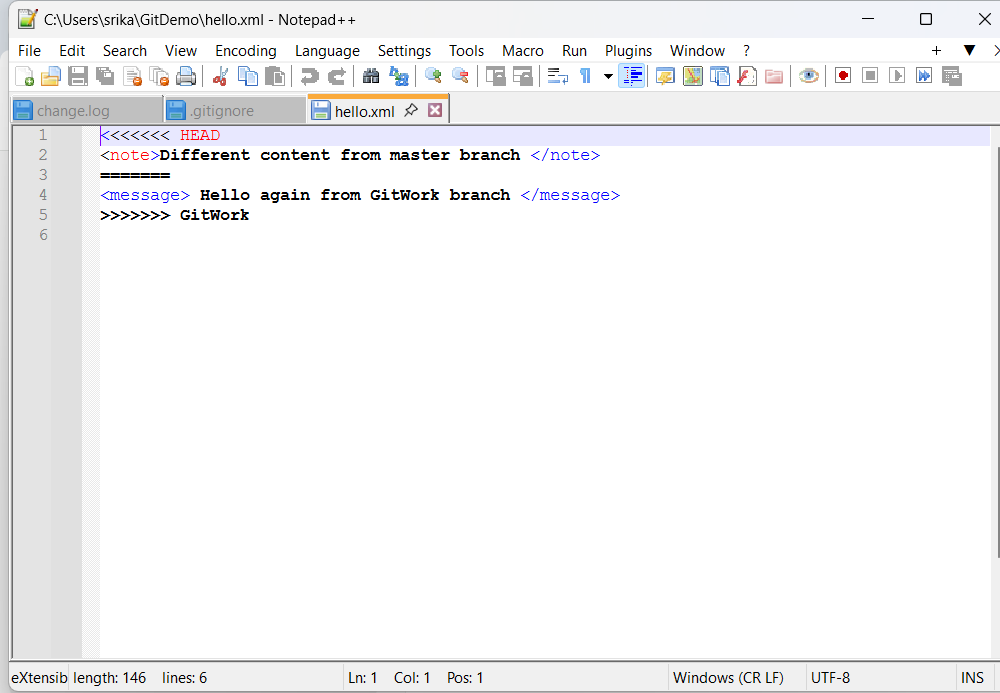


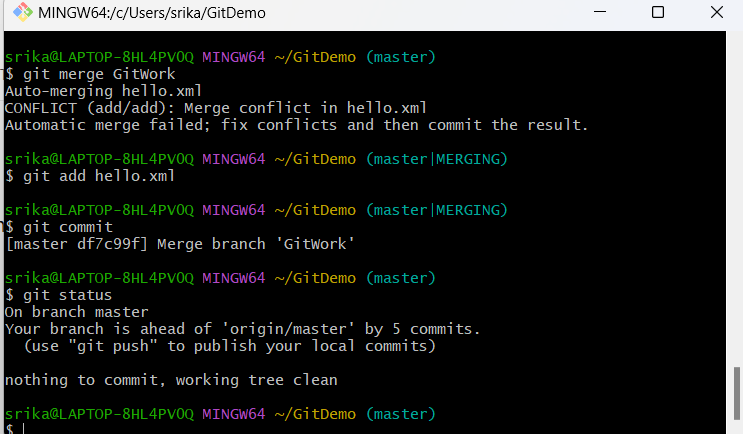
**HandsOn 4:**

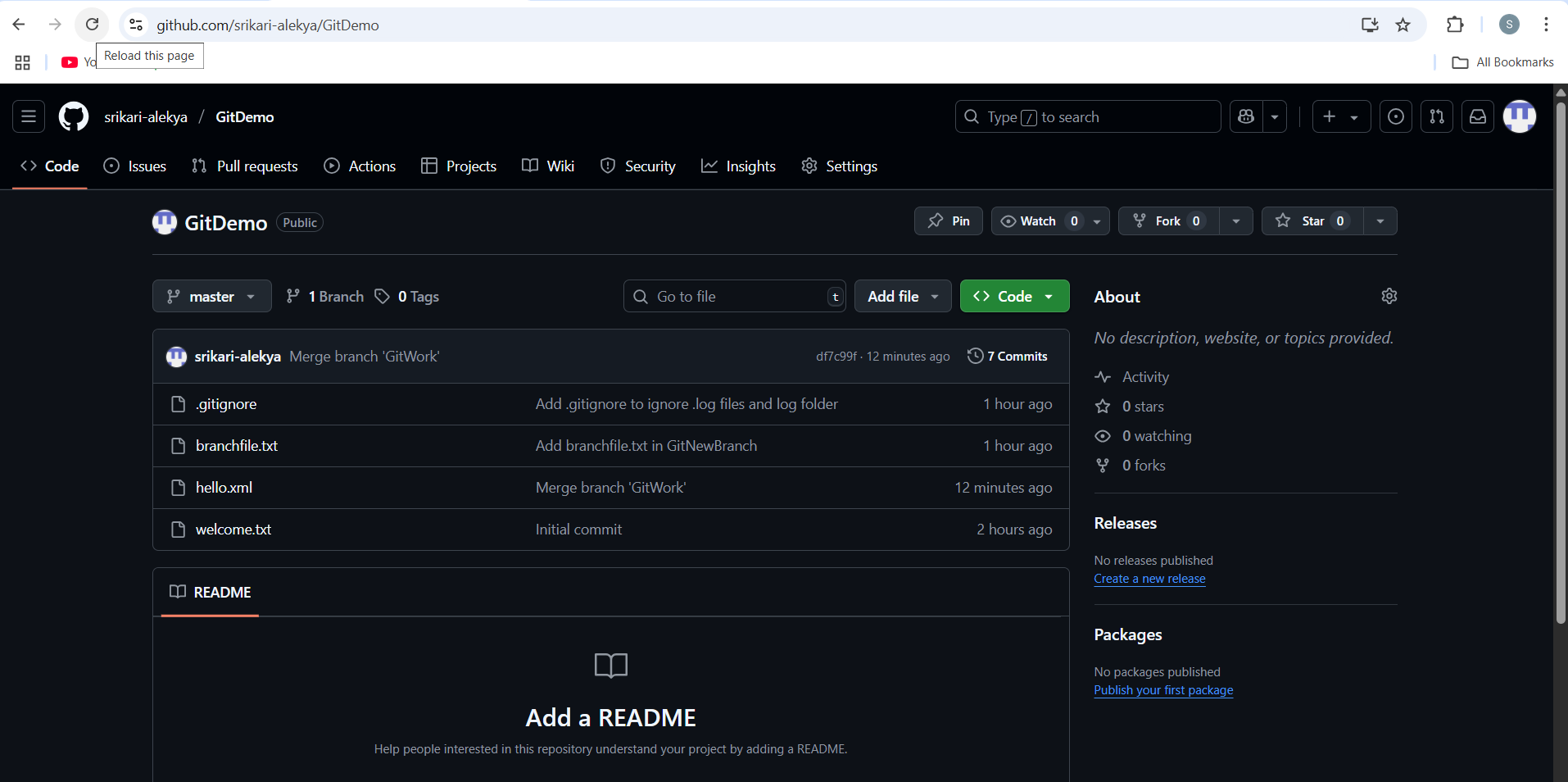
Implement conflict resolution when multiple users are updating the trunk (or master) in such a way that it results into a conflict with the branch’s modification.

**Steps:**

1. Checked the file with conflicts and located the conflict markers <<<<<<<, =======, >>>>>>>.
2. Manually edited the file to keep both required changes:
3. <note>Different content from master branch </note>
4. <message> Hello again from GitWork branch </message>
5. Removed the conflict markers so the file is clean and valid.
6. Saved the file after resolving the conflicts.
7. Added the resolved file to the staging area:
8. git add hello.xml
9. Committed the merge resolution:
10. git commit
11. Pushed the changes to the remote repository:
12. git push
13. Verified on the repository that the file shows the merged content without conflict markers.

****

****

****

**HandsOn 5:**

Execute steps involving clean up and push back to remote Git.

**Steps:**

1. Switched to master branch
2. git checkout master
3. Verified master branch is in clean state
4. git status
5. Listed all available branches
6. git branch -a
7. Pulled latest changes from remote master
8. git pull origin master
9. Pushed pending changes to remote repository
10. git push origin master

