**Week 8: GIT Hands-On**

**Student Name:** SHALU KUMARI

**Email:** 22052852@kiit.ac.in

**Superset ID:** 6363523

**1. Git-HOL: Foundational Git & GitHub Operations**

Objective:

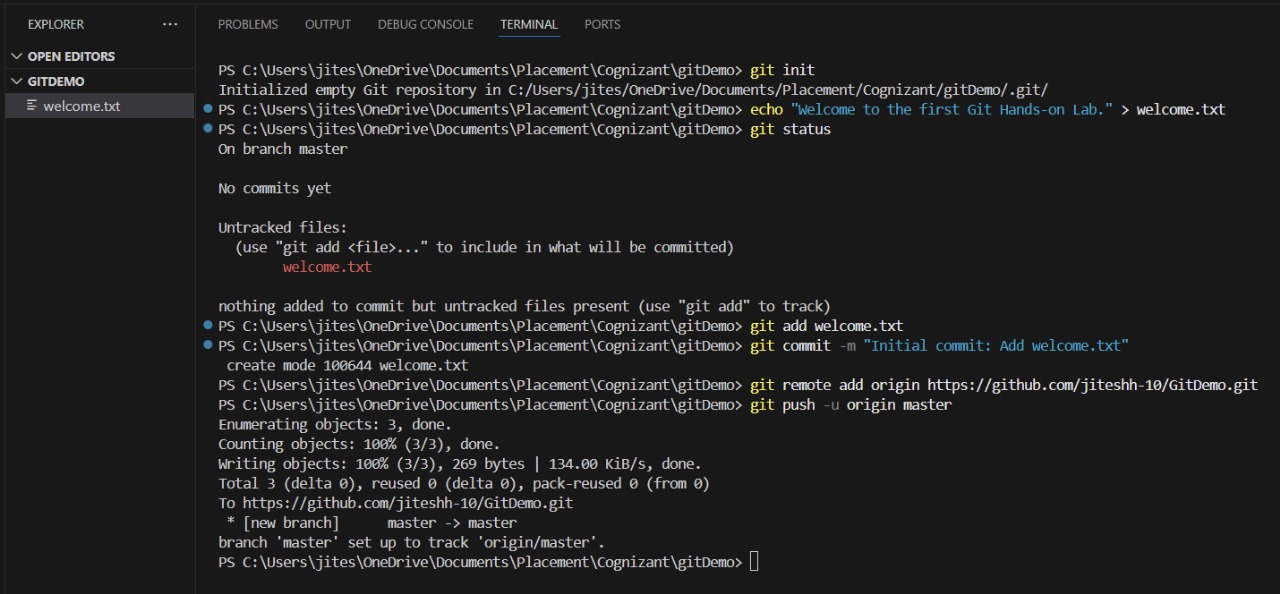
This lab's objective was to understand the fundamental Git workflow, including repository initialization, user configuration, and pushing a file to a remote GitHub repository.

Summary of Actions:

The process began by initializing the gitDemo directory as a Git repository using git init and configuring global user credentials. A welcome.txt file was then created, staged with git add, and committed to the local repository using git commit. A connection to the remote repository was established with git remote add origin, and the local commit was uploaded via git push -u origin master.

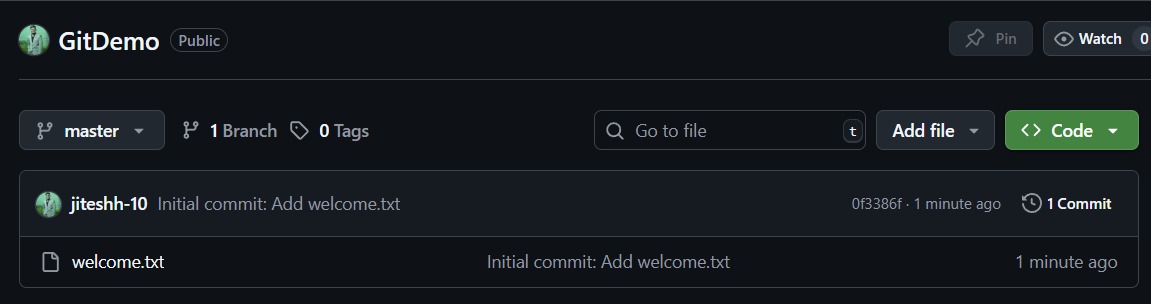
**Screenshots:**

* Terminal Output:



(This screenshot documents the successful execution of the command sequence, from repository initialization to the final push.)

* GitHub Repository after Push:



(This screenshot provides verification that the welcome.txt file and its corresponding commit were successfully populated in the remote GitHub repository.)

**2. Git-HOL: File Exclusion with** .gitignore

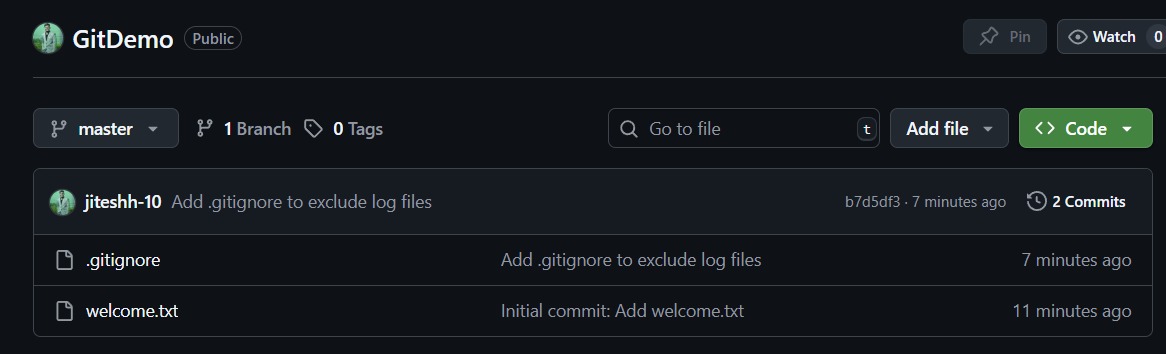
Objective:

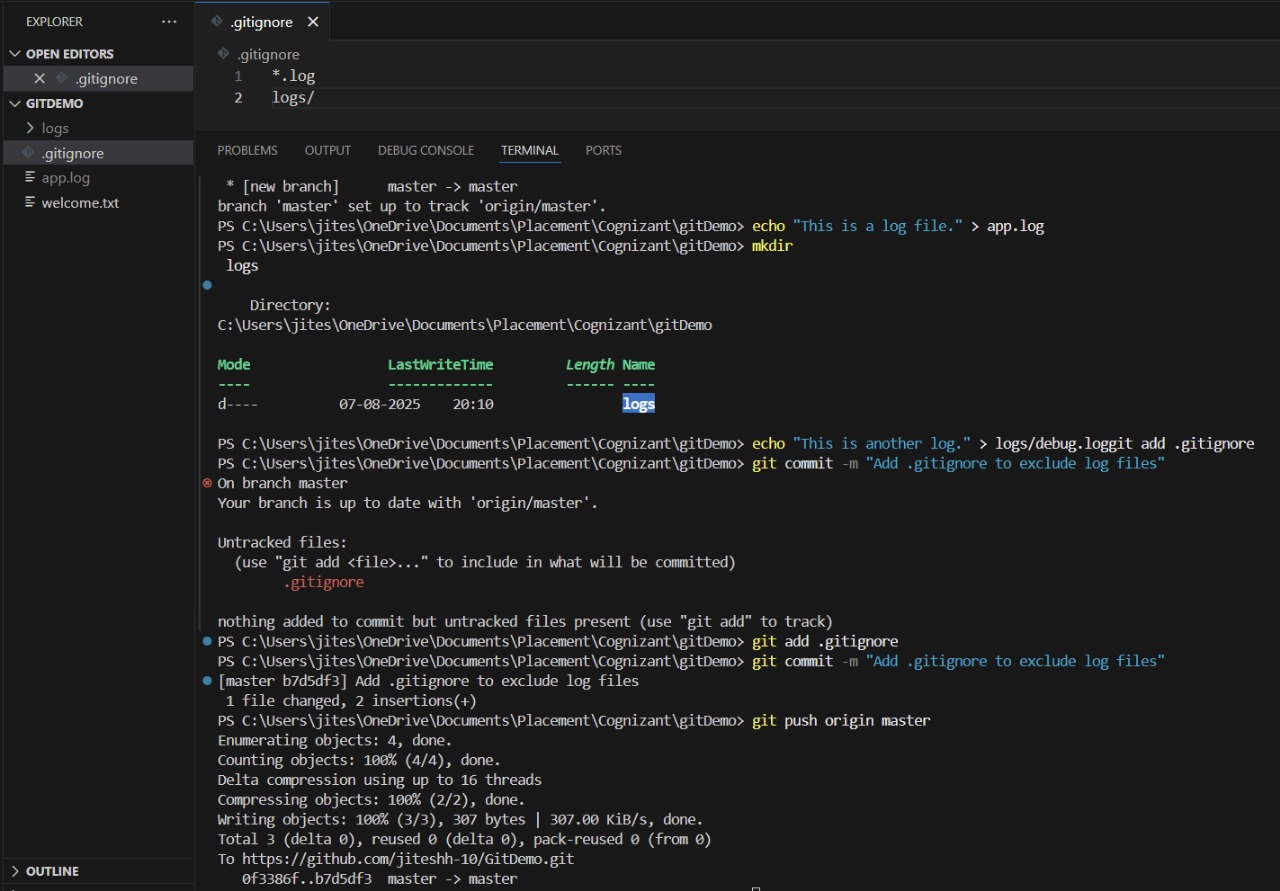
This exercise focused on implementing the .gitignore file to maintain repository integrity by preventing the tracking of non-essential assets like log files or dependency folders.

Summary of Actions:

To simulate a practical scenario, dummy log files and a logs directory were created. A .gitignore file was then generated in the project's root with patterns (\*.log, logs/) to exclude these items from version control. A git status check confirmed the exclusions were effective. The .gitignore file itself was then committed to the repository to ensure the rules are versioned and shared.

**Screenshots:**

* .gitignore File Content:
* Terminal Output after Ignoring:



**3. Git-HOL: Branching and Merging Strategies**

Objective:

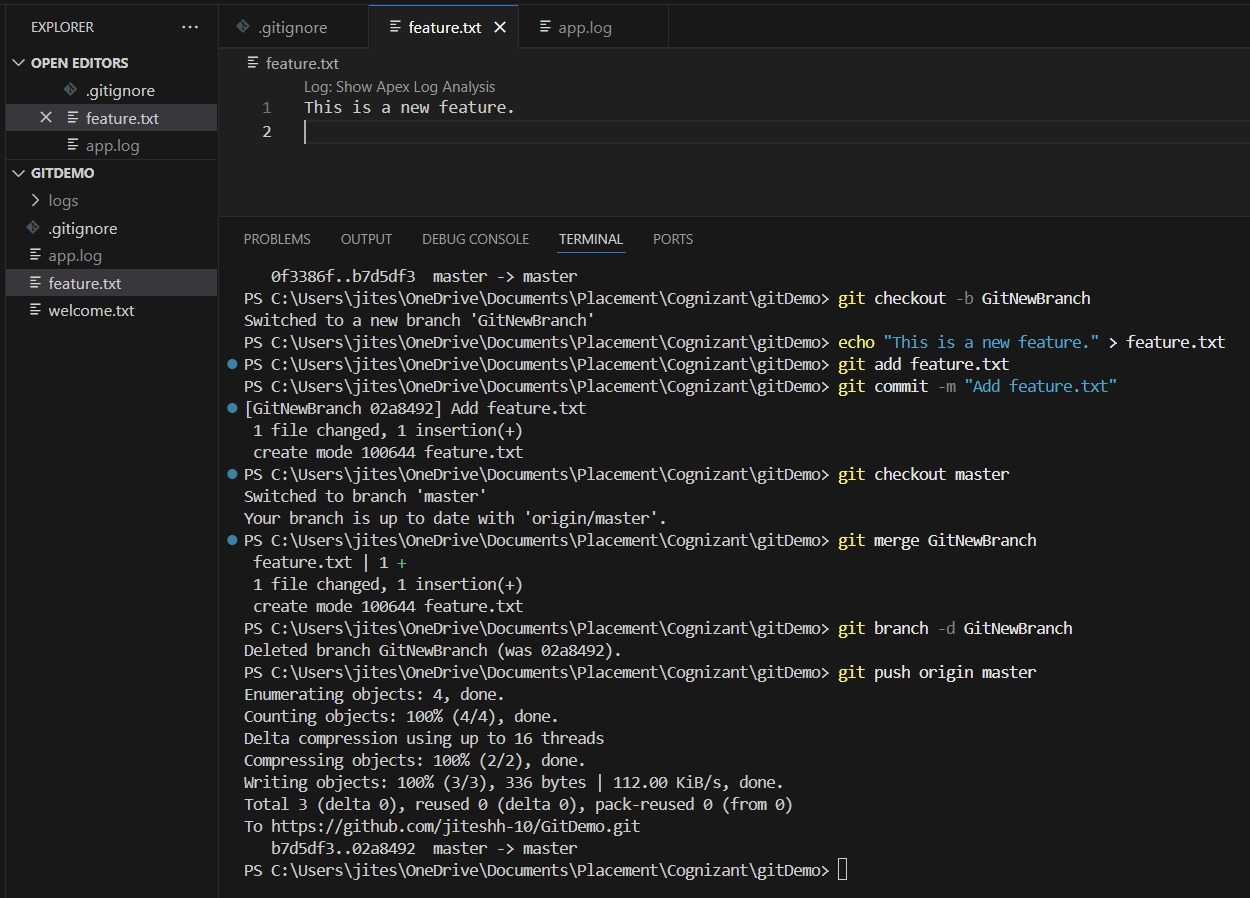
This module's objective was to understand the strategic use of Git branches for parallel development, which allows work on features in isolated environments to prevent disruption to the main master branch.

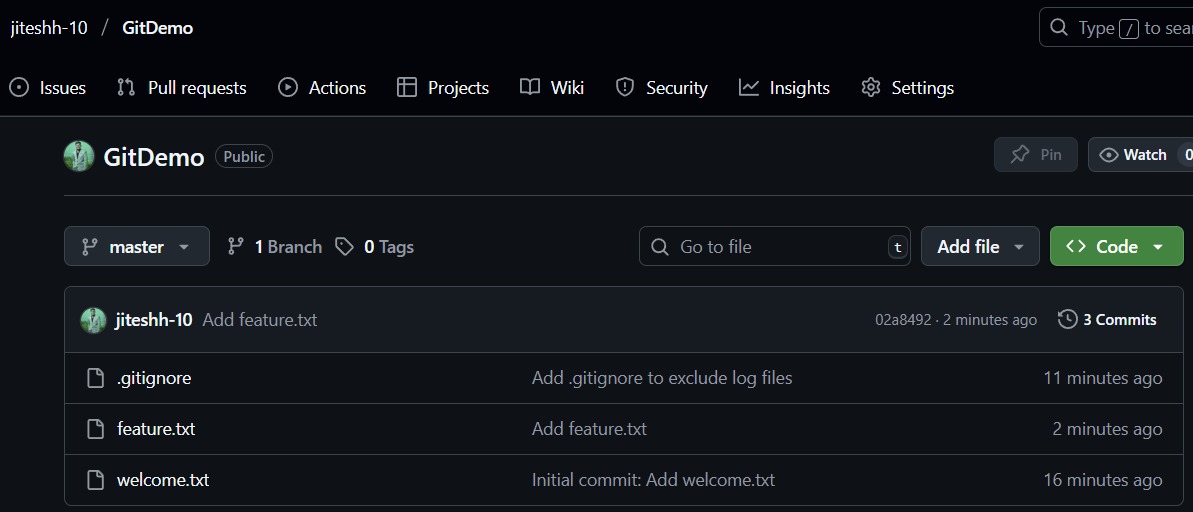
Summary of Actions:

A new branch, GitNewBranch, was created and activated using git checkout -b. On this branch, a feature.txt file was created and committed. After returning to the master branch, the changes from GitNewBranch were integrated using git merge. In adherence with best practices, the now-redundant feature branch was removed with git branch -d.

**Screenshots:**

* Branch Creation and Merging in Terminal:





**4. Git-HOL: Merge Conflict Resolution**

Objective:

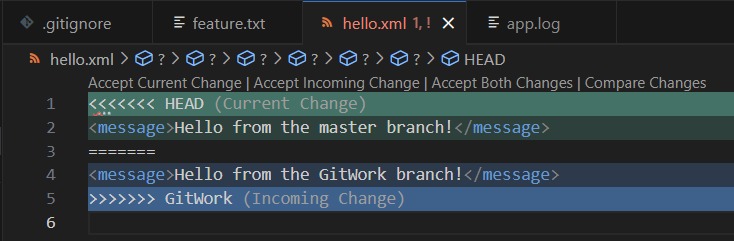
This lab addressed the practical scenario of resolving a merge conflict, which arises when Git cannot automatically merge competing changes made to the same file on different branches.

Summary of Actions:

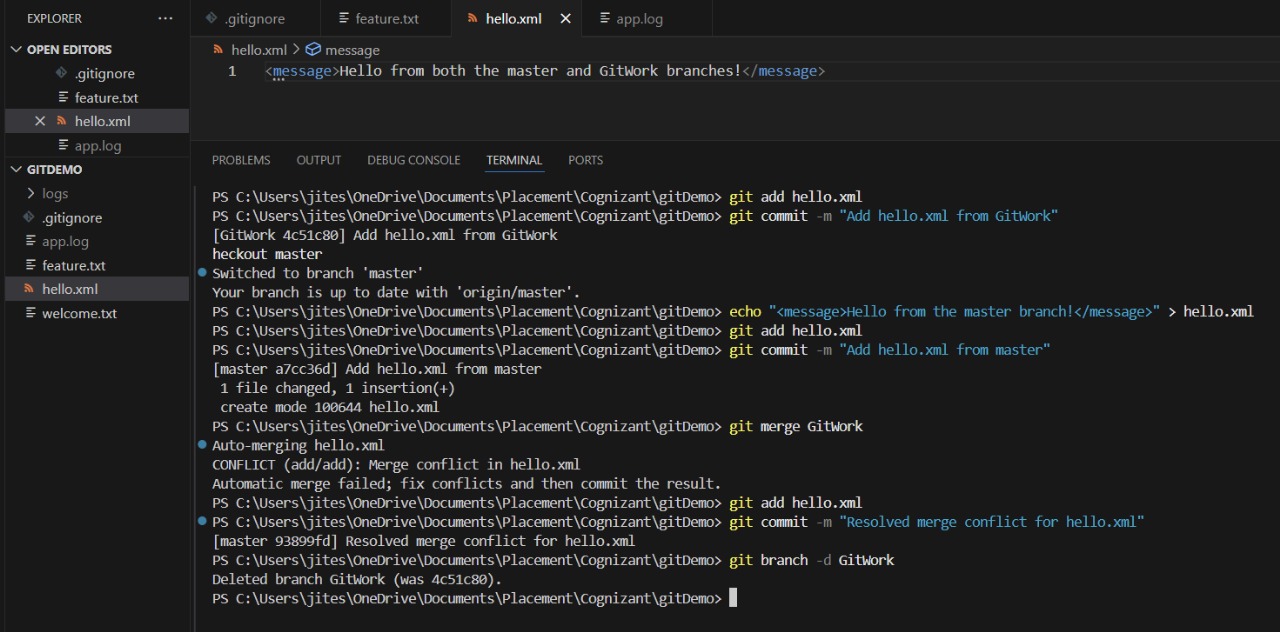
A merge conflict was engineered by creating a GitWork branch and committing a new hello.xml file, then creating a file with the same name but different content on the master branch. Attempting a git merge correctly resulted in a conflict. This was resolved within Visual Studio Code by manually editing the file to remove conflict markers and select the final content. The resolution was finalized by staging the corrected file with git add and completing the merge with git commit.

**Screenshots:**

* Conflict Markers in VS Code:



* Terminal Showing Conflict and Resolution:



**5. Git-HOL: Remote Repository Synchronization**

Objective:

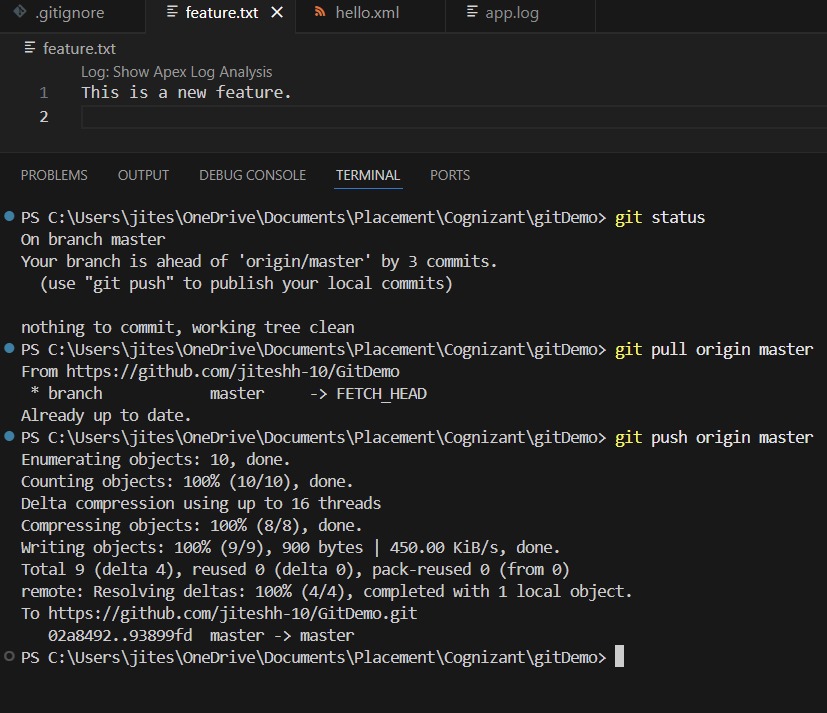
This concluding lab focused on synchronizing the local repository with its remote GitHub counterpart and deploying all accumulated local changes.

Summary of Actions:

In accordance with best practices, a git pull origin master command was executed to update the local repository and minimize potential conflicts. Following this, git push origin master was used to upload all local commits, including new files, the .gitignore configuration, and the results of the merge and conflict resolution exercises. A final review of the GitHub repository verified the successful update of the project's history.

**Screenshots:**

* Final Push Command in Terminal:



* Updated GitHub Repository History:

