# 1. Git-HOL

## Step 1 – Setup Git Configuration

1. Open Git Bash.

2. Check if Git is installed:

git --version

3. Configure your username and email:

git config --global user.name “CharithaCheeli"

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

4. Verify the configuration:

git config --list

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

1. Check if Notepad++ runs from Git Bash:

notepad++

- If it doesn’t work, add notepad++.exe to your Windows Environment Variables (Path).

2. Close and reopen Git Bash, then run:

notepad++

3. Create an alias for Notepad++:

alias npp='notepad++'

4. Set Notepad++ as the default Git editor:

git config --global core.editor "notepad++"

5. Verify editor setting:

git config --global -e

## Step 3 – Create a Local Repository

1. Open Git Bash in the desired folder.

2. Initialize repository:

git init GitDemo

3. Move into the folder:

cd GitDemo

4. Show hidden .git folder:

ls -a

## Step 4 – Create and Add a File

1. Create welcome.txt with some text:

echo "Welcome to Git Demo" > welcome.txt

2. Verify file exists:

ls

3. View file content:

cat welcome.txt

## Step 5 – Track and Commit the File

1. Check Git status:

git status

2. Stage the file:

git add welcome.txt

3. Commit with a message:

git commit -m "Added welcome.txt file"

4. Check status again:

git status

## Step 6 – Push to GitLab

1. Create a new project in GitLab named GitDemo.

2. Link the remote repository:

git remote add origin https://gitlab.com/CharithaCheeli/GitDemo.git

3. Pull any changes (if any):

git pull origin master

4. Push local repository to GitLab:

git push origin master

## Step 7 – Verify

• Open the GitLab repository in your browser.

• Confirm that welcome.txt appears with your commit message.

**2. Git-HOL**

**Objectives**

* Explain git ignore
* Explain how to ignore unwanted files using git ignore

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

* Implement git ignore command to ignore unwanted files and folders

**Prerequisites**

* Setting up Git environment
* Integrate notepad++ as a default editor
* A Git repository in the local system and a remote repository in GitLab

**Step 1 – Prepare Files and Folders to Ignore**

1. Navigate to your Git repository folder in Git Bash.
2. Create a log file:

touch error.log

1. Create a log folder:

mkdir logs

1. *(Optional)* Add files inside logs folder:

echo "Log data" > logs/debug.log

**Step 2 – Create or Edit .gitignore**

1. Open or create a .gitignore file in your repository root:
2. notepad++ .gitignore
3. Add the following lines to ignore all .log files and the logs folder:

\*.log

logs/

1. Save and close the file.

**Step 3 – Verify Ignored Files**

1. Check the status of your repository:

git status

1. Ensure that error.log and logs/ do **not** appear in the list of untracked files.
2. If they still appear, they might already be tracked by Git. Stop tracking them:

git rm --cached error.log

git rm -r --cached logs/

**Step 4 – Commit Changes**

1. Stage and commit the updated .gitignore file:

git add .gitignore

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

**Step 5 – Push to Remote Repository**

1. Push changes to GitLab:

git push origin master

**Step 6 – Final Verification**

* Check the remote repository on GitLab to confirm that error.log and logs/ are not uploaded.
* Verify that .gitignore exists in the repository with the correct rules.

# 3. Git-HOL

## Objectives

• Explain branching and merging

• Explain about creating a branch request in GitLab

• Explain about creating a merge request in GitLab

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

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

## Prerequisites

• Setting up Git environment with P4Merge tool for Windows

## Step 1 – Create a New Branch

1. Navigate to your Git repository in Git Bash.

2. Create a new branch named `GitNewBranch`:  
  
git branch GitNewBranch

3. List all local and remote branches:  
  
git branch -a  
- The branch with a `\*` is your current branch.

## Step 2 – Switch to the New Branch

1. Switch to `GitNewBranch`:  
  
git checkout GitNewBranch

2. Create and add new files to this branch:  
  
echo "Some new content" > branchfile.txt  
git add branchfile.txt  
git commit -m "Added branchfile.txt in GitNewBranch"

3. Verify changes:  
  
git status

## Step 3 – Merging the Branch into Master

1. Switch to the master branch:  
  
git checkout master

2. View differences between master and branch (command-line):  
  
git diff GitNewBranch

3. View differences visually with P4Merge:  
  
git mergetool

4. Merge the branch into master:  
  
git merge GitNewBranch

5. View merge history:  
  
git log --oneline --graph --decorate

6. Delete the merged branch:  
  
git branch -d GitNewBranch

7. Verify status:  
  
git status