**GIT**

1. **Git-HOL**

**# Repository setup**

mkdir GitDemo

cd GitDemo

git init

echo "Welcome to Git Hands-On Lab" > welcome.txt

git status

git add welcome.txt

git commit

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

git push -u origin master

**#Configure git**

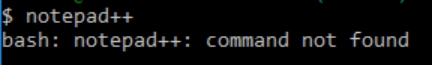
git --version

git config --global user.name "Your Name"

git config --global user.email "your.email@example.com"

git config –list

1. To check, if notepad++.exe execute from Git bash



If Git bash could not able to recognize notepad++ command that implies notepad++.exe is note added to the environment path variable.

To add path of notepad++.exe to environment variable, go to control panel -> System -> Advanced System settings. Go to Advanced tab -> Environment variables -> Add path of notepad++.exe to the path user variable by clicking on “Edit”

1. Exit Git bash shell, open bash shell and execute



Now, notepad++ will open from Git bash shell

1. To create an alias command for notepad++.exe, execute



It will open notepad++ from bash shell, and create a user profile by adding the line in notepad++

1. To configure the editor, execute the command

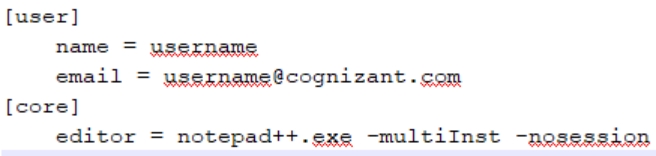


1. To verify if notepad++ is the default editor, execute the command



Here ‘-e’ option implies editor

It will show the entire global configuration as shown below,

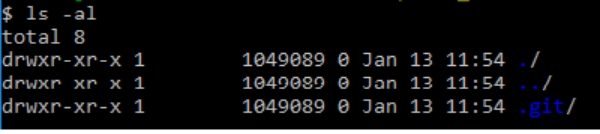


**Step 3: Add a file to source code repository**

1. Open Git bash shell and create a new project “**GitDemo**” by executing the command



1. Git bash initializes the “**GitDemo**” repository. To verify, execute the command

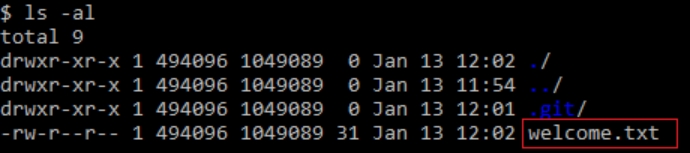


It will display all the hidden files in the Git “working directory”.

1. To create a file **“welcome.txt”** and add content to the file, execute the command



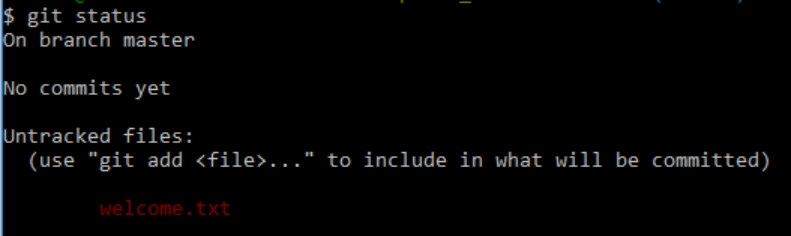
1. To verify if the file “welcome.txt” is created, execute



1. To verify the content, execute the command

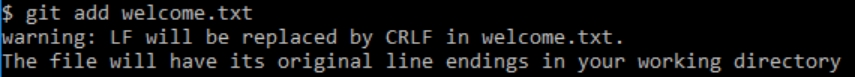


1. Check the status by executing



Now the file **“welcome.txt”** is available in Git “working directory”

1. To make the file to be tracked by Git repository, execute the command

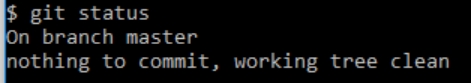


1. To add multi line comments, we are opening default editor to comment. Execute the command



Notepad++ editor will open and to add multi-line comment with default editor

1. To check if local and “Working Directory” git repository are same, execute git status



**welcome.txt** is added to the local repository.

1. Signup with GitLab and create a remote repository **“GitDemo”**
2. To pull the remote repository, execute

git pull origin master

1. To push the local to remote repository, execute

git push origin master

1. **Git-HOL**

**#Creating a local github repo**

mkdir GitIgnoreDemo

cd GitIgnoreDemo

git init

**#Create a .gitignore File**

touch .gitignore

notepad++ .gitignore

# Inside .gitignore, add:

\*.log

log/

**#Create Files or Folders to Ignore**

echo "This is a sample log file" > sample.log

mkdir log

echo "Some logs inside log folder" > log/error.log

**#Check Git Status**

Git status

**#Add Remaining Files and Commit**

echo "README for GitIgnoreDemo" > README.md

git add README.md

git commit -m "Added README, ignored .log files and log folder"

1. **Git-HOL**

**# Prerequisites**

Git installed and configured

P4Merge installed and configured:

Download: https://www.perforce.com/products/helix-core-apps/merge-diff-tool-p4merge

#mcd

git config --global merge.tool p4merge

git config --global mergetool.p4merge.cmd '"C:/Program Files/Perforce/p4merge.exe" "$BASE" "$LOCAL" "$REMOTE" "$MERGED"'

git config --global diff.tool p4merge

git config --global difftool.p4merge.cmd '"C:/Program Files/Perforce/p4merge.exe" "$LOCAL" "$REMOTE"'

**#Merge Steps**

git checkout master

git diff master GitNewBranch

git difftool master GitNewBranch

git merge GitNewBranch

git log --oneline --graph --decorate

git branch -d GitNewBranch

git status

**#Makes Changes**

"This is from the new branch" > feature.txt

git add feature.txt

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

git status

**#Branching Instructions**

# Create and switch

git branch GitNewBranch

git branch -a

git checkout GitNewBranch

1. **Git-HOL**

**#View History and Differences**

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

git diff master GitWork

git difftool master GitWork

**#Use 3-Way Merge Tool (P4Merge)**

git mergetool

P4Merge opens with:

BASE (common ancestor)

LOCAL (master)

REMOTE (GitWork)

**#Prerequisites**

Git and Git Bash installed

Notepad++ and P4Merge installed

A clean master branch in your local repository

**#Merge and Resolve Conflict**

git config --global diff.tool p4merge

git config --global difftool.p4merge.cmd '"C:/Program Files/Perforce/p4merge.exe" "$LOCAL" "$REMOTE"'

git merge GitWork

**#Finalize the Merge**

git add hello.xml

git commit -m "Resolved conflict in hello.xml while merging GitWork into master"

**#Delete Branch and View Final Log**

git branch -d GitWork

git log --oneline --graph –decorate

**#Create GitWork Branch and Add File**

git checkout -b GitWork

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

git status

git add hello.xml

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

**#Create Conflicting File in Master**

git checkout master

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

git add hello.xml

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

**#Cleanup and Ignore Merge Backup Files**

echo "\*.orig" >> .gitignore

git add .gitignore

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

1. **Git-HOL**

**#Verify master branch is clean**

git checkout master

git status

**#Push pending local commits to remote**

git push origin master

**#Pull latest changes from remote**

git pull origin master

**#Prerequisites**

You have already completed Git-T03-HOL\_002 (from previous HOLs)

Remote Git repo is already set up (GitHub/GitLab)

Internet connection is active

You’re on the master branch

**#List all available branches**

git branch -a