MANDATORY EXERCISE WEEK 8

## **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

## **Prerequisites**

* Install Git Bash client in your machine

Notes\*:

|  |
| --- |
| Please follow the below steps for creating a free account in GitHub.  Don’t use cognizant credentials to login to GitHub. |

Estimated time to complete this lab: **30 minutes.**

**Step 1: Setup your machine with Git Configuration**

To create a new repository, signup with GitLab and register your credentials

Login to GitLab and create a “GitDemo” project

1. To check if Git client is installed properly: Open Git bash shell and execute



If output shows Git with its version information that indicates, that Git Client installs properly.

1. To configure user level configuration of user ID and email ID execute



1. To check if the configuration is properly set, execute the following command.



**Step 2: Integrate notepad++.exe to Git and make it a default editor**

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

C:\Program Files\Notepad++\

notepad++

alias np='notepad++'

git config --global core.editor "'C:/Program Files/Notepad++/notepad++.exe' -multiInst -nosession"

git config --global -e

mkdir GitDemo

cd GitDemo

git init

OUTPUT:

Initialized empty Git repository in /c/Users/YourName/GitDemo/.git/

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

ls

cat welcome.txt

OUTPUT:

Welcome to Git Hands-on Lab!

git status

OUTPUT:

Untracked files:

(use "git add <file>..." to include in what will be committed)

welcome.txt

git add welcome.txt

git commit

Added welcome.txt file for initial setup.

PUSH

git remote add origin <https://gitlab.com/yourusername/GitDemo.git>

PULL

git pull origin master

PUSH

git push -u origin master

OUTPUT:

Enumerating objects: 3, done.

Counting objects: 100% (3/3), done.

Writing objects: 100% (3/3), done.

To https://gitlab.com/yourusername/GitDemo.git

\* [new branch] master -> master

OUTPUT



## **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**

The following are the pre-requisites to complete this hands-on lab:

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

Notes\*:

|  |
| --- |
| Please follow the below steps for creating a free account in GitHub.  Do not use cognizant credentials to login to GitHub. |

Estimated time to complete this lab: **20 minutes.**

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.

cd GitDemo # or your existing Git repo

echo "This is a test log" > error.log

mkdir log

echo "System logs here" > log/system.log

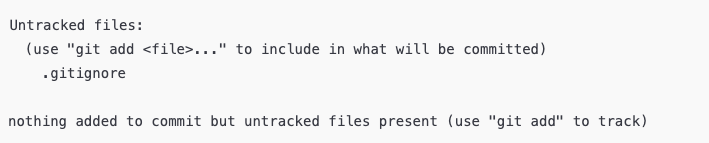
notepad++ .gitignore # Or use nano, vim, code, etc.

\*.log

log/

git status

OUTPUT



git add .gitignore

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

git add .gitignore

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

git push origin master

## **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**

The following are the pre-requisites to complete this hands-on lab:

* Setting up Git environment with P4Merge tool for Windows

Notes\*:

|  |
| --- |
| Please follow the below steps for creating a free account in GitHub.  Do not use cognizant credentials to login to GitHub. |

Estimated time to complete this lab: **30 minutes.**

Please follow the instruction to complete the hands-on. Each instruction expects a command for the Git Bash.

**Branching:**

1. Create a new branch **“GitNewBranch”.**
2. List all the local and remote branches available in the current trunk. Observe the “\*” mark which denote the current pointing branch.
3. Switch to the newly created branch. Add some files to it with some contents.
4. Commit the changes to the branch.
5. Check the status with **“git status”** command.

**Merging:**

1. Switch to the master
2. List out all the differences between trunk and branch. These provide the differences in command line interface.
3. List out all the visual differences between master and branch using **P4Merge tool**.
4. Merge the source branch to the trunk.
5. Observe the logging after merging using **“git log –oneline –graph –decorate”**
6. Delete the branch after merging with the trunk and observe the git status.

git checkout -b GitNewBranch

git branch -a

\* GitNewBranch

Master

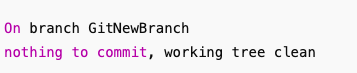
echo "This is a file from GitNewBranch" > branchfile.txt

git add branchfile.txt

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

git status

OUTPUT

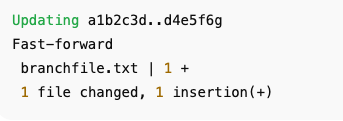


git checkout master

git diff master GitNewBranch

git merge GitNewBranch

OUTPUT



git log --oneline --graph –decorate

\* d4e5f6g (HEAD -> master) Added branchfile.txt in GitNewBranch

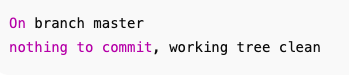
\* a1b2c3d initial commit

git branch -d GitNewBranch

Deleted branch GitNewBranch (was d4e5f6g).

git status

OUTPUT



## **Objectives**

* Explain how to resolve the conflict during merge.

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

* 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.

## **Prerequisites**

The following are the pre-requisites to complete this hands-on lab:

* Hands-on ID: **“Git-T03-HOL\_001”**

Notes\*:

|  |
| --- |
| Please follow the below steps for creating a free account in GitHub.  Do not use cognizant credentials to login to GitHub. |

Estimated time to complete this lab: **30 minutes.**

Please follow the instructions to complete the hands-on. Each instruction expect a command for the Git Bash.

1. Verify if master is in clean state.
2. Create a branch **“GitWork”.** Add a file “hello.xml”.
3. Update the content of “hello.xml” and observe the status
4. Commit the changes to reflect in the branch
5. Switch to master.
6. Add a file **“hello.xml”** to the master and add some different content than previous.
7. Commit the changes to the master
8. Observe the log by executing **“git log –oneline –graph –decorate –all”**
9. Check the differences with Git diff tool
10. For better visualization, use P4Merge tool to list out all the differences between master and branch
11. Merge the bran to the master
12. Observe the git mark up.
13. Use 3-way merge tool to resolve the conflict
14. Commit the changes to the master, once done with conflict
15. Observe the git status and add backup file to the .gitignore file.
16. Commit the changes to the .gitignore
17. List out all the available branches
18. Delete the branch, which merge to master.
19. Observe the log by executing **“git log –oneline –graph –decorate”**

git checkout master

git status

git checkout -b GitWork

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

git add hello.xml

git commit -m "GitWork: Added hello.xml with GitWork message"

git checkout master

echo "<message>Hello from Master</message>" > hello.xml

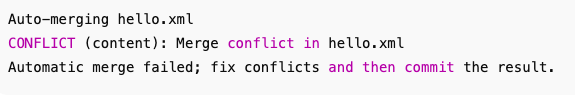
git add hello.xml

git commit -m "Master: Added hello.xml with master message"

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

git diff GitWork

git merge GitWork



XML

<<<<<<< HEAD

<message>Hello from Master</message>

=======

<message>Hello from GitWork</message>

>>>>>>> GitWork

git mergetool

<message>Hello from both branches</message>

git add hello.xml

git commit -m "Resolved conflict in hello.xml during merge"

echo "\*.orig" >> .gitignore

git add .gitignore

git commit -m "Ignore merge conflict backup files"

git branch

git branch -d GitWork

Deleted branch GitWork (was <commit-hash>).

git log --oneline --graph –decorate

## **Objectives**

* Explain how to clean up and push back to remote Git

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

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

## **Prerequisites**

The following are the pre-requisites to complete this hands-on lab:

* Hands-on ID: **“Git-T03-HOL\_002”**

Notes\*:

|  |
| --- |
| Please follow the below steps for creating a free account in GitHub.  Do not use cognizant credentials to login to GitHub. |

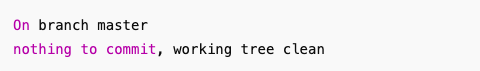
Estimated time to complete this lab: **10 minutes.**

Please follow the instructions to complete the hands-on. Each instruction expects a command for the Git Bash.

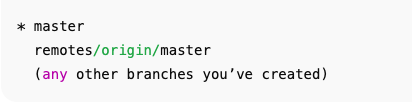
1. Verify if master is in clean state.
2. List out all the available branches.
3. Pull the remote git repository to the master
4. Push the changes, which are pending from **“Git-T03-HOL\_002”** to the remote repository.
5. Observe if the changes are reflected in the remote repository.

git checkout master

git status



git branch -a



git pull origin master



git push origin master

