Cognizant - DN 4.0 I Deep Skilling

Week-8

**1] 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**

git –version

git config --global user.name "Meghana"

git config --global user.email [meghanajakkanagari@gmail.com](mailto:meghanajakkanagari@gmail.com)

git config –list

"C:\Program Files\Notepad++\notepad++.exe"

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

git config --global -e

mkdir my-git-lab

cd my-git-lab

git init

echo "Hello Git" > hello.txt

git status

git add hello.txt

git commit -m "Initial commit: Added hello.txt"

**2]**

* **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**

.gitignore is a special text file in a Git repository.

It tells Git which files or folders to ignore when tracking changes.

Common use cases:

* Ignoring temporary files (\*.tmp, \*.log)
* Ignoring system-generated files (.DS\_Store, Thumbs.db)
* Ignoring build/output directories (/bin, /dist)
* Ignoring sensitive config files (like .env)

mkdir git-ignore-lab

cd git-ignore-lab

git init

cd path/to/your/repo

notepad++ .gitignore

# Ignore log files

\*.log

# Ignore temporary files

\*.tmp

# Ignore node\_modules folder

node\_modules/

# Ignore build directory

/dist/

# Ignore system files

.DS\_Store

Thumbs.db

git status

git rm --cached filename

git commit -m "Stop tracking filename and update .gitignore"

git add .gitignore

git commit -m "Added .gitignore file to ignore unwanted files"

git push origin main

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

git branch

git branch feature-update

git checkout feature-update

git switch feature-update

echo "This is a new feature" > feature.txt

git add feature.txt

git commit -m "Added feature.txt in feature-update branch"

git checkout master

git switch master

git merge feature-update

git branch -d feature-update

git push origin master

**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.**

git init conflict-lab

cd conflict-lab

echo "Version 1" > file.txt

git add file.txt

git commit -m "Initial commit with file.txt"

git branch feature-update

git checkout feature-update

echo "Feature branch changes" > file.txt

git add file.txt

git commit -m "Updated file.txt in feature-update branch"

git checkout master

echo "Master branch changes" > file.txt

git add file.txt

git commit -m "Updated file.txt in master branch"

git merge feature-update

CONFLICT (content): Merge conflict in file.txt

Automatic merge failed; fix conflicts and then commit the result.

<<<<<<< HEAD

Master branch changes

=======

Feature branch changes

>>>>>>> feature-update

Master branch changes + Feature branch changes

git add file.txt

git commit -m "Resolved merge conflict between master and feature-update"

git push origin master

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

git branch -d feature-update

git branch -D feature-update

git push origin --delete feature-update

git branch -a

git status

git push origin master

git pull origin master

git clean -fd