**3 . GIT - HOL**

**Branching and Merging in Git**

**Objectives**

* Explain branching and merging

**Branching** in Git allows you to create a separate line of development from the main codebase (usually called main or master). This enables multiple developers to work on different features or fixes independently without affecting the main project. Each branch can have its own commits and changes isolated from others.  
  
**Merging** is the process of integrating changes from one branch into another (commonly from a feature branch back into the main branch). When merging, Git combines the histories and changes, resolving conflicts if the same parts of files were changed differently. Merging helps to consolidate work and keep the project updated.

* 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

**Note:** Please create a free GitHub account. Do not use Cognizant credentials to login to GitHub.

Estimated time to complete this lab: 30 minutes.

**Branching**

**Step 0: Setup or Navigate to Git Repository**

* Open Git Bash
* Navigate to your Git repository folder or create a new one:

mkdir GitDemo

cd GitDemo

git init

* If folder exists, just navigate:

cd GitDemo

git init

**Step 1: Create a new branch named GitNewBranch**

git branch GitNewBranch

**Step 2: List all local and remote branches**

git branch -a

Observe the \* mark which denotes the current branch.

**Step 3: Switch to the newly created branch**

git checkout GitNewBranch

**Step 4: Add a file to the branch**

echo "This is a sample file in GitNewBranch." > example.txt

**Step 5: Stage (add) the file**

git add example.txt

**Step 6: Commit the changes to the branch**

git commit -m "Add example.txt with sample content to GitNewBranch"

**Step 7: Check the status to confirm commit**

git status

**Merging**

**Step 1: Switch back to the main branch**

git checkout main

**Step 2: View differences between main and GitNewBranch**

git diff main GitNewBranch

**Step 3: Visualize differences using P4Merge tool**

Set P4Merge as your difftool if not set:

git config --global diff.tool p4merge

git config --global difftool.p4merge.cmd "p4merge \$LOCAL \$REMOTE"

git difftool main GitNewBranch

**Step 4: Merge the source branch into main**

git merge GitNewBranch

**Step 5: View commit history with graph, one line, and decorate options**

git log --oneline --graph --decorate

**Step 6: Delete the branch after successful merge**

git branch -d GitNewBranch

Check branches again:

git branch -a