**3. Git-HOL**

## Prerequisites

1. Git installed and configured on our machine.  
2. Git Bash for executing commands.  
3. (Windows) P4Merge installed and configured as the difftool/mergetool (optional but required for visual diffs).

## Part A — Setup P4Merge (Windows)

1. Download P4Merge from the Perforce website and install it on Windows.

2. Configure Git to use P4Merge (run these in Git Bash):

git config --global diff.tool p4merge

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

git config --global merge.tool p4merge

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

3. Verify configuration:

git config --global --list

## Part B — Branching

Follow these steps in order. Replace <path-to-repo> with our repository path and GitNewBranch with the branch name if you want a different name.

1. Open Git Bash and go to your repository folder:

cd /path/to/your/local/repo

2. Ensure you are on master (or trunk):

git checkout master

3. Update local master from remote:

git pull origin master

4. Create a new branch and switch to it (single command):

git checkout -b GitNewBranch

(or) git branch GitNewBranch && git checkout GitNewBranch

5. Confirm current branch (the \* shows the active branch):

git branch

6. List all local and remote branches:

git branch -a

7. Create a new file and add some content (example):

echo "This is the first file in GitNewBranch" > file1.txt

8. Stage the file:

git add file1.txt

9. Check the status:

git status

10. Commit the change with a meaningful message:

git commit -m "Add file1.txt on GitNewBranch"

11. (Optional) Repeat steps 7–10 to add more files or changes.

12. Push the branch to the remote and set upstream:

git push -u origin GitNewBranch

## Part C — Creating a Branch Request (GitLab UI)

If you prefer to create a branch in GitLab UI instead of pushing from local:

1. Login to GitLab and open your project.

2. Go to Repository -> Branches.

3. Click "New branch".

4. Enter the branch name (GitNewBranch) and select the source branch (master).

5. Click "Create branch". The new branch will appear in the branch list.

## Part D — Merge (Command-line + Visual checks)

Do these steps to merge your branch into master locally, verify, and then push.

1. Switch to master:

git checkout master

2. Update local master:

git pull origin master

3. See the differences between master and the branch (text):

git diff master..GitNewBranch

4. See only commits on the branch (useful to review what will be merged):

git log master..GitNewBranch --oneline

5. Open a visual diff using P4Merge (if configured):

git difftool master..GitNewBranch

6. Merge the source branch into master locally (non-fast-forward to keep history):

git merge --no-ff GitNewBranch -m "Merge GitNewBranch into master"

If Git reports conflicts, resolve them:

a) Run git status to see conflicted files.

b) Run git mergetool to open P4Merge and resolve conflicts.

c) After resolving, run git add <file> for the resolved files and then git commit (if required).

7. Push merged master to remote:

git push origin master

8. View compact commit graph/log:

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

9. Delete the local branch (after successful merge):

git branch -d GitNewBranch

10. Delete the remote branch (optional, if you no longer need it):

git push origin --delete GitNewBranch

11. Confirm repository status:

git status

## Part E — Creating a Merge Request (GitLab UI)

After you push the branch to remote, create a Merge Request in GitLab:

1. Login to GitLab and open your project.

2. Go to Merge Requests -> New merge request.

3. Choose the source branch: GitNewBranch and the target branch: master.

4. Click "Compare branches and continue".

5. Fill in the Title (e.g., "Merge GitNewBranch into master") and add a clear Description of your changes.

6. Assign reviewer(s), add labels, milestone if needed, and set the appropriate assignee.

7. Click "Create merge request".

8. Reviewers will verify changes, CI pipelines (if any) will run, and comments may be added.

9. If there are conflicts, GitLab will show them; you can either resolve locally and push or use the Web IDE to resolve simple conflicts.

10. Once approvals are complete and pipeline (if any) passes, click "Merge". You may check the "Remove source branch" option to auto-delete the branch on merge.

## Part F — Post-merge cleanup and verification

1. After merge in GitLab, sync your local repo:

git checkout master

git pull origin master

2. Remove the local branch if not already deleted:

git branch -d GitNewBranch

3. Verify branch removal on remote (optional):

git branch -a

4. Check repository status and logs:

git status

git log --oneline --graph --decorate -n 20