**Week-8**

**GIT**

**1. Git-HOL**

**Verify Git is installed**

git --version

**Configure Git user (one-time setup)**

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

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

Verify:

git config --global --list.

**Make Notepad++ your Git editor**

**Add Notepad++ to PATH so can type notepad++ from Git Bash**

1. Open **Control Panel → System → Advanced system settings**.
2. Advanced → **Environment Variables…**
3. Under **User variables**, select **Path** → **Edit** → **New** and paste the Notepad++ folder, e.g.:

C:\Program Files\Notepad++

1. Click OK, close/reopen Git Bash.

**Command-line method** (PowerShell as Admin):

setx PATH "%PATH%;C:\Program Files\Notepad++"

Now you can run:

notepad++ welcome.txt

**Create a local repo and add a file**

mkdir GitDemo

cd GitDemo

git init

ls -la

**Create welcome.txt and add some content:**

echo "Welcome to GitDemo" > welcome.txt

cat welcome.txt

Check status:

git status

**5 — Stage and commit the file**

Add the file to the staging area:

git add welcome.txt

Check git status — now it will be staged.

Commit it:

* If you want to use Notepad++ as commit-editor:

git commit

Notepad++ will open; type a multi-line commit message, save & close the editor, commit completes.

* Or do a quick single-line commit:

git commit -m "Add welcome.txt"

Check log:

git log --oneline

**Create a remote repository on GitHub (or GitLab)**

Open your browser:

**GitHub**

1. Sign in to <https://github.com> (create an account if needed).
2. Click **+ → New repository**.
3. Repository name: GitDemo (or whatever prefer).
4. **Do not** initialize with README (leave empty) — this avoids merge issues.
5. Create repository.
6. GitHub will show the remote URL (HTTPS). Copy it (e.g. https://github.com/yourusername/GitDemo.git).

**GitLab** is similar (<https://gitlab.com> → Projects → New project → create).

**Link local repo to remote and push**

In Git Bash (inside your GitDemo folder):

First ensure default branch name. New repos often use main. Set local branch to main:

git branch -M main

Add the remote origin (use the URL you copied):

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

Push your commits to remote and set upstream:

git push -u origin main

If Git asks for username/password:

* For GitHub in 2024+, use a Personal Access Token (PAT) instead of password for HTTPS pushes. Or set up SSH keys and use the SSH URL.

If push is rejected (remote has commits), you can first pull remote changes and merge:

git pull origin main --allow-unrelated-histories

git add .

git commit -m "Merge remote"

git push origin main

**8 — Pull changes from remote**

To update your local copy with remote changes:

git pull origin main

or simply

git pull

**2. Git-HOL**

**Open your Git repository**

1. Go to your **local Git repository** folder in your system.
2. Open it in **VS Code** or any terminal where Git is available.

**Create a .log file**

1. In the repository folder, create a sample log file:

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

Or manually create **test.log** in VS Code.

**Create a log folder**

1. Make a folder named **log** inside your repo:

mkdir log

1. Create a file inside it (just for testing):

echo "log folder file" > log/example.txt

**Create/Edit .gitignore file**

1. In the **root folder** of the repo, create a .gitignore file if it doesn’t exist:

touch .gitignore

1. Open .gitignore in VS Code and add:

\*.log

log/

This means:

* + Ignore all files ending with .log
  + Ignore the **log** folder completely

**Check Git status**

git status

**Commit changes (without the ignored files)**

1. Add and commit:

git add .

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

1. Push to remote:

git push

**Verify**

* Your .log file and log folder will **remain in your local machine** but won’t be tracked in Git.
* .gitignore file will be in Git to ensure others also ignore those files.

**3.Git-HOL**

**Open Git Bash in your project folder**

Make sure you’re inside your Git repository (already initialized with git init and having a master/main branch).

**Create a new branch**

git branch GitNewBranch

**List all branches (local & remote)**

git branch -a

* \* indicates the branch you are currently on.
* You’ll see something like:

\* master

GitNewBranch

**Switch to the new branch**

git checkout GitNewBranch

or (modern way):

git switch GitNewBranch

**Add some files in this branch**

echo "This is a new branch file" > branchfile.txt

**Stage and commit the changes**

git add branchfile.txt

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

**Check the branch status**

git status

You should see **“nothing to commit, working tree clean”** if changes are committed.

**Merging Process**

**Switch back to master**

git checkout master

or:

git switch master

**See differences between master and branch (command line)**

git diff master GitNewBranch

**See differences visually in P4Merge**

If P4Merge is configured as merge tool:

git difftool master GitNewBranch

This will open the P4Merge visual diff.

**Merge GitNewBranch into master**

git merge GitNewBranch

If no conflicts, it will merge directly.

**Check commit history after merge**

git log --oneline --graph --decorate

You’ll see the merge commit in the log.

**Delete the merged branch**

git branch -d GitNewBranch

If you want to delete from remote (if pushed):

git push origin --delete GitNewBranch

**Final check**

git status

Should show:

On branch master

nothing to commit, working tree clean

**4.Git-HOL**

**Verify master is in a clean state**

git status

**Create a branch GitWork**

git checkout -b GitWork

This creates and switches to a new branch named GitWork.

**Add hello.xml in GitWork**

1. Create the file:

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

1. Check status:

git status

**Commit changes in GitWork**

git add hello.xml

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

**Switch to master branch**

git checkout master

**Add hello.xml in master with different content**

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

git add hello.xml

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

**View logs with branch visualization**

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

You should see master and GitWork diverging.

**Compare differences**

git diff master GitWork

*(If you have P4Merge installed)*

git mergetool

Select P4Merge to see a visual difference.

**Merge GitWork into master**

git merge GitWork

This will cause a merge conflict because hello.xml is different in both.

**Resolve conflict**

Open hello.xml in your editor. You’ll see conflict markers:

<<<<<<< HEAD

<message>Hello from master branch</message>

=======

<message>Hello from GitWork branch</message>

>>>>>>> GitWork

Edit to resolve, for example:

<message>Hello from both branches</message>

Save the file.

**Mark conflict resolved & commit**

git add hello.xml

git commit -m "Merge GitWork into master with conflict resolved"

**Add .gitignore for backup files**

echo "\*.orig" >> .gitignore

git add .gitignore

git commit -m "Add .gitignore to ignore merge backup files"

**Delete merged branch**

git branch -d GitWork

**View final commit graph**

git log --oneline --graph –decorate

**5.Git-HOL**

**Open Git Bash**

* Go to the folder where your Git repository is located.
* Right-click → select **"Git Bash Here"**.

**Verify if master branch is in a clean state**

git status

* If it says **"working tree clean"**, your master branch has no pending changes.
* If there are uncommitted changes, commit them or stash them:

git add .

git commit -m "Saving changes before cleanup"

OR

git stash

**List all available branches**

git branch -a

* This will show all **local** and **remote** branches.

**Step 4: Pull latest changes from remote master**

git checkout master

git pull origin master

* Ensures your master branch is up-to-date with the remote repository.

**Push pending changes from “Git-T03-HOL\_002” to remote**

1. Switch to your working branch:

git checkout Git-T03-HOL\_002

1. Push your branch to the remote repository:

git push origin Git-T03-HOL\_002

**Verify changes in remote repository**

* Go to **GitHub** in your browser.
* Open your repository and check if branch **Git-T03-HOL\_002** contains the latest changes.