

# Git Hands-On Lab Documentation

**Title:** Version Control with Git – Hands-On Lab

## Objective:

This lab introduces fundamental Git commands and workflows, guiding users through Git configuration, editor integration, and repository management using GitLab.

## Lab Objectives

By the end of this lab, you will be able to:

1. Configure Git on your local machine
2. Integrate Notepad++ as the default Git editor
3. Create and manage a Git repository
4. Push code to a remote GitLab repository

## Steps :

1. Git Configuration Setup
2. Integrate Notepad++ as Default Git Editor
3. Create and Manage Git Repository
4. Push to GitLab

## Output Snapshots :

```
ayush@AyushJainSparsh MINGW64 ~  
$ git config --global --list  
user.email=ayushjainsparsh2004.ajs@gmail.com  
user.name=Ayush Jain  
filter=diff --merge=diff --merge=diff --merge=diff
```

```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (master)  
$ echo "welcome to the version control" >> welcome.txt  
  
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (master)  
$ ls  
welcome.txt
```

```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (master)
$ cat welcome.txt
welcome to the version control

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
    welcome.txt

nothing added to commit but untracked files present (use "git add" to track)

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (master)
$ git add .
warning: in the working copy of 'welcome.txt', LF will be replaced by CRLF the n
ext time Git touches it
```

```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (master)
$ git commit -m "first"
[master (root-commit) b20cc86] first
 1 file changed, 1 insertion(+)
 create mode 100644 welcome.txt

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (master)
$ git status
On branch master
nothing to commit, working tree clean
```

```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git pull origin main
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 2.74 KiB | 254.00 KiB/s, done.
From https://gitlab.com/AyushJainSparsh/demo
 * branch                main          -> FETCH_HEAD
 * [new branch]           main          -> origin/main
Successfully rebased and updated refs/heads/main.
```

```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ ls
README.md  welcome.txt
```

```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git push origin main
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 321 bytes | 321.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://gitlab.com/AyushJainSparsh/demo.git
   af17b30..6bbd42c  main -> main
```

**Result:**

- Configured Git and set up your identity
- Integrated Notepad++ as your default Git editor
- Created a local Git repository and committed changes
- Pushed your code to a remote GitLab repository

# Git Hands-On Lab Documentation

## Objective:

Understand the purpose and functionality of .gitignore

1. Learn how to exclude specific files and folders from Git tracking
2. Implement .gitignore to ignore .log files and log folders

## Lab Objectives

By the end of this lab, you will be able to:

3. Initialise .gitignore file to stop unwanted files like .log to get store in gitlab

## Steps :

4. Initialize Git Repository (if not already done)
5. Create Files and Folders to Ignore
6. Create/Edit .gitignore File
7. Verify .gitignore Behavior

## Output Snapshots :

```
ayush@AyushJainSparsh MINGW64 ~  
$ cd onedrive/desktop/demo  
  
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)  
$ ls  
README.md  welcome.txt  
  
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)  
$ nano .log  
  
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)  
$ cat .log  
this is a default .log file  
  
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)  
$ nano .gitignore
```

```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)  
$ nano .gitignore  
  
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)  
$ git status  
On branch main  
Untracked files:  
  (use "git add <file>..." to include in what will be committed)  
    .gitignore  
  
nothing added to commit but untracked files present (use "git add" to track)  
  
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)  
$ ls  
README.md  welcome.txt  
  
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)  
$ ls -a  
./  ../  .git/  .gitignore  .log  README.md  welcome.txt
```

# Git Hands-On Lab Documentation

## Objective:

- Understand Git branching and merging workflows
- 1. Learn how to create a branch and merge it into the master (trunk)
- 2. Explore GitLab's branch and merge request features
- 3. Use P4Merge for visual diffing during merges

## Prerequisites

Ensure the following are ready before starting:

- Git environment installed and configured
- P4Merge tool installed and set as Git's diff/merge tool
- A local Git repository initialized
- A remote GitLab repository connected

## What is Branching in Git?

Branching allows developers to diverge from the main codebase (usually master or main) to work on features or fixes independently.

Benefits:

- Isolated development
- Parallel workflows
- Safe experimentation

## What is Merging in Git?

Merging integrates changes from one branch into another. Typically, feature branches are merged into master after testing.

## Lab Objectives

By the end of this lab, you will be able to:

- 4. Initialise .gitignore file to stop unwanted files like .log to get store in gitlab

## Steps :

- 5. Create a New Branch
- 6. Switch to the New Branch Add Files and Make Changes
- 7. Commit the Changes Check Git Status
- 8. Switch Back to Master & List CLI Differences
- 9. View Visual Differences with p4merge

10. Merge the branches

**Output Snapshots :**

```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git branch GitNewBranch

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git branch -a
  GitNewBranch
* main
  remotes/origin/main

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git checkout GitNewBranch
Switched to branch 'GitNewBranch'

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (GitNewBranch)
$ git branch -a
* GitNewBranch
  main
  remotes/origin/main

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (GitNewBranch)
$ echo "This is a new feature file" > feature.txt

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (GitNewBranch)
$ git add .
warning: in the working copy of '.gitignore', LF will be replaced by CRLF
warning: in the working copy of 'feature.txt', LF will be replaced by CRLF

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (GitNewBranch)
$ git commit -m "add fetaures"
[GitNewBranch 86da64c] add fetaures
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 feature.txt

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (GitNewBranch)
$ git status
On branch GitNewBranch
nothing to commit, working tree clean
```

```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git config --global diff.tool p4merge

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git config --global merge.tool p4merge

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git branch
  GitNewBranch
* main

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git merge GitNewBranch
Updating 6bbd42c..86da64c
Fast-forward
 .gitignore | 1 +
 feature.txt | 1 +
 2 files changed, 2 insertions(+)
 create mode 100644 .gitignore
 create mode 100644 feature.txt

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git log --oneline --graph --decorate
* 86da64c (HEAD -> main, GitNewBranch) add fetaures
* 6bbd42c (origin/main) demo
* af17b30 Initial commit

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git branch -d GitNewBranch
Deleted branch GitNewBranch (was 86da64c).

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    sh.exe.stackdump

nothing added to commit but untracked files present (use "git add" to track)

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git branch
* main
```

# Git Hands-On Lab Documentation

## Objective:

- Understand how merge conflicts occur in Git
- Learn how to resolve conflicts using Git and a 3-way merge tool
- Use P4Merge for visual conflict resolution
- Update .gitignore to exclude backup files

## Prerequisites

Before starting, ensure the following are ready:

- Git environment installed and configured
- P4Merge tool installed and integrated with Git
- Hands-on ID: Git-T03-HOL\_001
- A local Git repository initialized and connected to a remote

## What is a Merge Conflict?

A merge conflict occurs when Git cannot automatically reconcile differences between two branches. This typically happens when the same file is modified in both branches in overlapping lines.

## Lab Objectives

By the end of this lab, you will be able to:

11. Initialise .gitignore file to stop unwanted files like .log to get store in gitlab

## Steps :

12. Verify main is clean
13. Create a new branch. Add and modify hello.xml in branch
14. Commit the Changes Check Git Status
15. Switch Back to main and add conflicting hello.xml in main
16. Commit changes and check difference
17. Merge the branches
18. Resolve conflict using 3 way merge tool
19. Add backup file to .gitignore
20. Commit and list all branches and delete final log

## Output Snapshots :



```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git checkout -b GitWork
Switched to a new branch 'GitWork'

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (GitWork)
$ echo "<message>Hello from GitWork branch</message>" > hello.xml

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (GitWork)
$ git status
On branch GitWork
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    hello.xml
    sh.exe.stackdump

nothing added to commit but untracked files present (use "git add" to track)

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (GitWork)
$ git add hello.xml
warning: in the working copy of 'hello.xml', LF will be replaced by CRLF the
```

```
ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (GitWork)
$ git commit -m "Added hello.xml in GitWork branch"
[GitWork f7b6062] Added hello.xml in GitWork branch
1 file changed, 1 insertion(+)
create mode 100644 hello.xml

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (GitWork)
$ git checkout main
Switched to branch 'main'

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ echo "<message>Hello from master branch</message>" > hello.xml

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git add hello.xml
warning: in the working copy of 'hello.xml', LF will be replaced by CRLF

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git commit -m "Added hello.xml in master with different content"
[main 19bd64d] Added hello.xml in master with different content
1 file changed, 1 insertion(+)
create mode 100644 hello.xml

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git diff GitWork
diff --git a/hello.xml b/hello.xml
index 0bf4c19..82868a7 100644
--- a/hello.xml
+++ b/hello.xml
@@ -1,1 @@
-<message>Hello from GitWork branch</message>
+<message>Hello from master branch</message>

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git merge GitWork
Auto-merging hello.xml
CONFLICT (add/add): Merge conflict in hello.xml
Automatic merge failed; fix conflicts and then commit the result.

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main|MERGING)
$ git mergetool
Merging:
hello.xml
```

```

yush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main|MERGING)
git add hello.xml
git commit -m "Resolved merge conflict in hello.xml"
main 137ef93] Resolved merge conflict in hello.xml

yush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
ls -a
./ .git/ .log feature.txt hello_BACKUP_892.xml hello_
./ .gitignore README.md hello.xml hello_BASE_892.xml hello_

yush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
echo "*.orig" >> .gitignore

yush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
git branch -d GitWork
Deleted branch GitWork (was f7b6062).

yush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
git log --oneline --graph --decorate
    137ef93 (HEAD -> main) Resolved merge conflict in hello.xml
\
 * f7b6062 Added hello.xml in GitWork branch
| 19bd64d Added hello.xml in master with different content
/
86da64c add fetaures
6bbd42c (origin/main) demo
af17b30 Initial commit

```

# Git Hands-On Lab Documentation

## Objective:

- Ensure the local repository is clean and synchronized
  - Push pending changes to the remote repository
  - Confirm successful update on GitHub

## Prerequisites

Before starting, make sure:

1. Git is installed and configured
2. You have a GitHub account
3. Hands-on ID: Git-T03-HOL\_002
4. Remote repository is already set up and linked

## Steps :

5. Verify main is clean
6. List all available branches
7. Pull remote repo to main
8. Push pending changes to remote
9. Verify changes on Gitlab

## Output Snapshots :

```

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   .gitignore

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        hello_BACKUP_892.xml
        hello_BASE_892.xml
        hello_LOCAL_892.xml
        hello_REMOTE_892.xml
        sh.exe.stackdump

no changes added to commit (use "git add" and/or "git commit -a")

ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git branch
* main


ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git pull origin main

```


```






ayush@AyushJainSparsh MINGW64 ~/onedrive/desktop/demo (main)
$ git push origin main
Enumerating objects: 14, done.
Counting objects: 100% (14/14), done.
Delta compression using up to 12 threads
Compressing objects: 100% (9/9), done.
Writing objects: 100% (13/13), 1.20 KiB | 614.00 KiB/s, done.
Total 13 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
To https://gitlab.com/AyushJainSparsh/demo.git
   6bbd42c..137ef93  main -> main

```


**demo**

main
demo
+
Find file
Code


**Resolved merge conflict in hello.xml**
Ayush Jain authored 7 minutes ago
137ef939
History

Name	Last commit	Last update
 .gitignore	add fetaures	37 minutes ago
 README.md	Initial commit	1 hour ago
 feature.txt	add fetaures	37 minutes ago
 hello.xml	Resolved merge conflict in hello.xml	7 minutes ago
 welcome.txt	demo	58 minutes ago