## **Objectives**

* **Explain how to resolve the conflict during merge.**

**Git** is a distributed version control system used to track and manage changes in code or files.

It stores changes as **snapshots** (commits) so you can move between versions easily.

Developers can work on **branches** independently without affecting the main code.

Changes are merged back into the main branch, optionally after resolving conflicts.

Git works locally but can sync with remote repositories like **GitHub** or **GitLab**.

Ask ChatGPT

In this hands-on lab, you will learn how to:

* 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.

## **Prerequisites**

The following are the pre-requisites to complete this hands-on lab:

* Hands-on ID: **“Git-T03-HOL\_001”**

Notes\*:

|  |
| --- |
| Please follow the below steps for creating a free account in GitHub.  Do not use cognizant credentials to login to GitHub. |

Estimated time to complete this lab: **30 minutes.**

Please follow the instructions to complete the hands-on. Each instruction expect a command for the Git Bash.

1. Verify if master is in clean state.
2. Create a branch **“GitWork”.** Add a file “hello.xml”.
3. Update the content of “hello.xml” and observe the status
4. Commit the changes to reflect in the branch
5. Switch to master.
6. Add a file **“hello.xml”** to the master and add some different content than previous.
7. Commit the changes to the master
8. Observe the log by executing **“git log –oneline –graph –decorate –all”**
9. Check the differences with Git diff tool
10. For better visualization, use P4Merge tool to list out all the differences between master and branch
11. Merge the bran to the master
12. Observe the git mark up.
13. Use 3-way merge tool to resolve the conflict
14. Commit the changes to the master, once done with conflict
15. Observe the git status and add backup file to the .gitignore file.
16. Commit the changes to the .gitignore
17. List out all the available branches
18. Delete the branch, which merge to master.
19. Observe the log by executing **“git log –oneline –graph –decorate”**

Outputs:







