## **Objectives**

* Explain how to resolve the conflict during merge.

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”**

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”**

rajas@LAPTOP-N2GE05EJ MINGW64 ~

$ mkdir GitConflictDemo

rajas@LAPTOP-N2GE05EJ MINGW64 ~

$ cd GitConflictDemo

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo

$ git init

Initialized empty Git repository in C:/Users/rajas/GitConflictDemo/.git/

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ echo "Initial project setup" > readme.txt

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ git add readme.txt

warning: in the working copy of 'readme.txt', LF will be replaced by CRLF the next time Git touches it

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ git commit -m "Initial commit"

[main (root-commit) 6cc8d61] Initial commit

1 file changed, 1 insertion(+)

create mode 100644 readme.txt

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ git checkout -b GitWork

Switched to a new branch 'GitWork'

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (GitWork)

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

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (GitWork)

$ git add hello.xml

warning: in the working copy of 'hello.xml', LF will be replaced by CRLF the next time Git touches it

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (GitWork)

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

[GitWork bbb5da7] Add hello.xml in GitWork branch

1 file changed, 1 insertion(+)

create mode 100644 hello.xml

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (GitWork)

$ git checkout main

Switched to branch 'main'

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ echo "<greeting>Hello from main branch</greeting>" > hello.xml

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ git add hello.xml

warning: in the working copy of 'hello.xml', LF will be replaced by CRLF the next time Git touches it

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ git commit -m "Add hello.xml in main branch"

[main 0d64a31] Add hello.xml in main branch

1 file changed, 1 insertion(+)

create mode 100644 hello.xml

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

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

\* 0d64a31 (HEAD -> main) Add hello.xml in main branch

| \* bbb5da7 (GitWork) Add hello.xml in GitWork branch

|/

\* 6cc8d61 Initial commit

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (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.

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main|MERGING)

$ cat hello.xml

<<<<<<< HEAD

<greeting>Hello from main branch</greeting>

=======

<greeting>Hello from GitWork branch</greeting>

>>>>>>> GitWork

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main|MERGING)

$ cat > hello.xml

<greeting>Hello from both branches</greeting>

[2]+ Stopped cat > hello.xml

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main|MERGING)

$ git add hello.xml

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main|MERGING)

$ git commit -m "Resolve merge conflict in hello.xml"

[main 4b53996] Resolve merge conflict in hello.xml

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ echo "\*.orig" >> .gitignore

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ git add .gitignore

warning: in the working copy of '.gitignore', LF will be replaced by CRLF the next time Git touches it

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ git commit -m "Ignore merge backup files"

[main fd9d1b1] Ignore merge backup files

1 file changed, 1 insertion(+)

create mode 100644 .gitignore

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ git branch -d GitWork

Deleted branch GitWork (was bbb5da7).

rajas@LAPTOP-N2GE05EJ MINGW64 ~/GitConflictDemo (main)

$ git log --oneline --graph --decorate

\* fd9d1b1 (HEAD -> main) Ignore merge backup files

\* 4b53996 Resolve merge conflict in hello.xml

|\

| \* bbb5da7 Add hello.xml in GitWork branch

\* | 0d64a31 Add hello.xml in main branch

|/

\* 6cc8d61 Initial commit