## **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”**

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**

### **Theory**

1. **What is a Merge Conflict?**
   1. A merge conflict occurs when Git cannot automatically combine changes from two branches because the same part of a file was modified differently in both branches.
   2. Git stops the merge and lets you manually resolve the conflict.
2. **When does it happen?**
   1. When two people edit the same lines of a file in different branches.
   2. When one branch deletes a file and the other modifies it.
3. **How to resolve it?**
   1. Git marks the conflict in the file with:

<<<<<<< HEAD  
code from current branch  
=======  
code from branch being merged  
>>>>>>> branch-name

* 1. You edit the file to keep the correct changes, remove these markers, save, and then commit.

1. **Tools to help:**
   1. **Git Diff** – Shows conflicts in the terminal.
   2. **P4Merge** or other merge tools – Provides a visual side-by-side view.

## **Hands-On**

### **Step 1 – Check that master is clean**

$ git checkout master  
$ git status  
On branch master  
nothing to commit, working tree clean

**Step 2 – Create a branch and add file**

$ git branch GitWork  
$ git checkout GitWork  
$ echo "<greeting>Hello from GitWork</greeting>" > hello.xml  
$ git add hello.xml  
$ git commit -m "Added hello.xml in GitWork branch"  
[GitWork a1b2c3d] Added hello.xml in GitWork branch  
 1 file changed, 1 insertion(+)  
 create mode 100644 hello.xml

**Step 3 – Update file in GitWork**

$ echo "<greeting>Hello from GitWork - updated</greeting>" > hello.xml  
$ git status  
On branch GitWork  
Changes not staged for commit:  
 modified: hello.xml

Commit:

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

### **Step 4 – Switch to master and create different change**

$ git checkout master  
$ echo "<greeting>Hello from Master branch</greeting>" > hello.xml  
$ git add hello.xml  
$ git commit -m "Added hello.xml in Master branch"

**Step 5 – View logs**

$ git log --oneline --graph --decorate --all  
\* 2b3c4d5 (HEAD -> master) Added hello.xml in Master branch  
| \* a1b2c3d (GitWork) Updated hello.xml in GitWork branch  
|/  
\* e5f6g7h Initial commit

### **Step 6 – Check differences**

$ git diff master GitWork

$ git mergetool

**Step 7 – Merge and cause conflict**

$ git merge GitWork  
Auto-merging hello.xml  
CONFLICT (content): Merge conflict in hello.xml  
Automatic merge failed; fix conflicts and then commit the result.

### **Step 8 – Resolve conflict**

<<<<<<< HEAD  
<greeting>Hello from Master branch</greeting>  
=======  
<greeting>Hello from GitWork - updated</greeting>  
>>>>>>> GitWork

Edit to:

<greeting>Hello from Master and GitWork</greeting>

**Step 9 – Commit resolved merge**

$ git add hello.xml  
$ git commit -m "Resolved merge conflict between Master and GitWork

### **Step 10 – Ignore backup files**

$ echo "\*.orig" >> .gitignore  
$ git add .gitignore  
$ git commit -m "Ignored backup files"

**Step 11 – Delete branch**

### $ git branch -d GitWork Deleted branch GitWork (was a1b2c3d).

### **Step 12 – Final log**

$ git log --oneline --graph --decorate  
\* 9x8y7z6 (HEAD -> master) Ignored backup files  
\* 7a6b5c4 Resolved merge conflict between Master and GitWork  
\* 2b3c4d5 Added hello.xml in Master branch  
\* e5f6g7h Initial commit