Module 2: Version Control with Git

Demo Document - 3

Demo – 3: Branching and Merging

- To create a new branch from your current branch
  Syntax: git branch <branchname>

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git branch feat1
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

- You can then switch to this newly created branch
  Syntax: git checkout <ranchname>

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git checkout feat1
Switched to branch 'feat1'
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

- Creating and switching to a new branch can be done with using **-b flag**
  Syntax: git checkout -b <branchname>

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git checkout -b feat1
Switched to a new branch 'feat1'
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

- **Branch** command lists all the branches and also points to the current working branch
  Syntax: git branch

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git branch
* feat1
  master
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

- Different modified branches can be **merged** together using merge
  Syntax: git merge <branchname>
- The branch mentioned is merged into the current branch

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git merge feat1
Updating 96d29b0..0ebb201
Fast-forward
 Demo2.java | 1 +
 1 file changed, 1 insertion(+)
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

- **Merged** branches can be deleted using -**d** flag
  Syntax: git branch -d <branchname>

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git branch -d feat1
Deleted branch feat1 (was 0ebb201).
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

- **Unmerged** branches can be deleted using **-D** flag
  Syntax: git branch -D <branchname>

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git branch -D feat1
Deleted branch feat1 (was fb0ba99).
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

Merge Conflict:

- Merge Conflict arises on merging branches

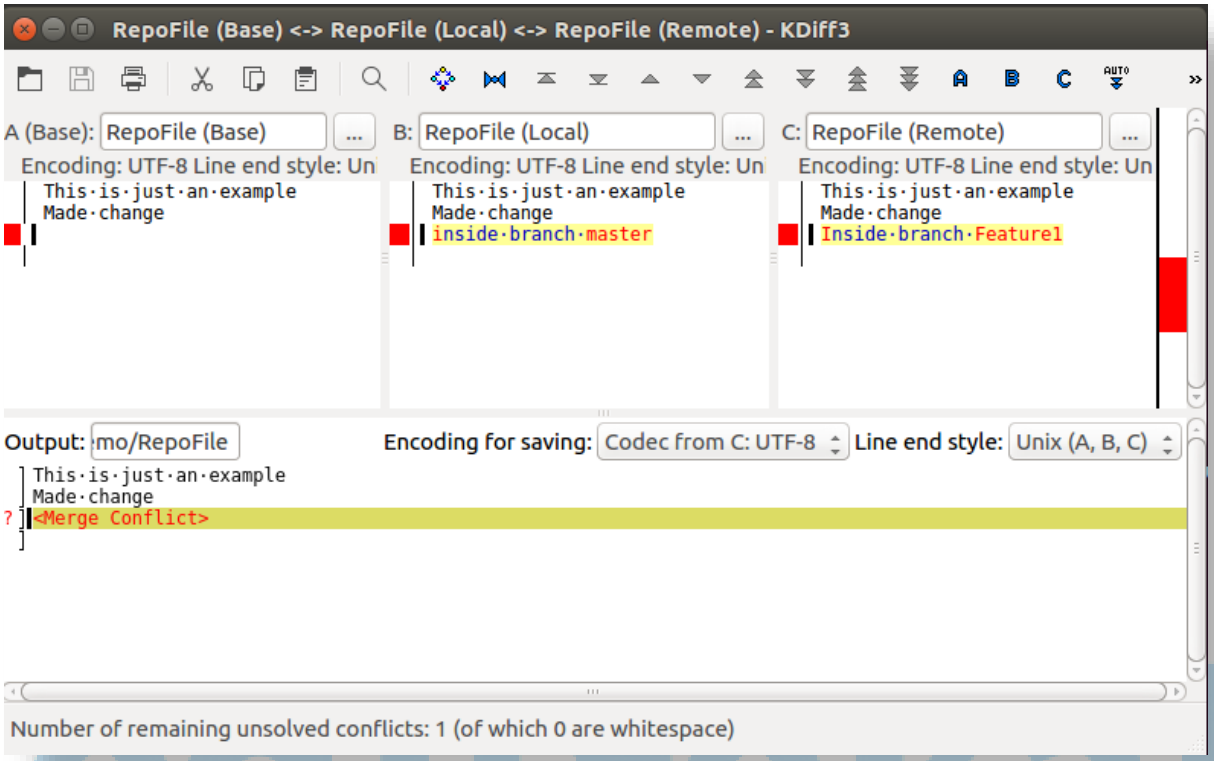
```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git merge feature1
Auto-merging RepoFile
CONFLICT (content): Merge conflict in RepoFile
Automatic merge failed; fix conflicts and then commit the result.
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

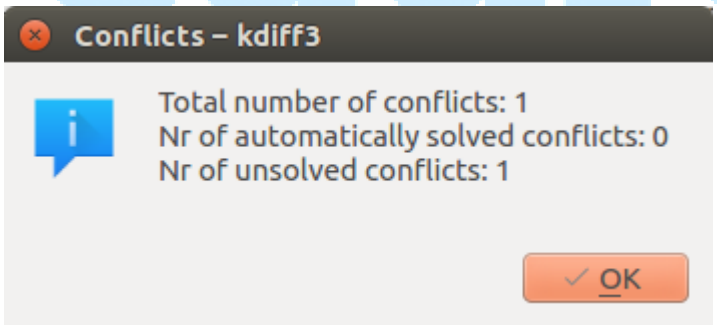
- Set a default merge tool before resolving conflict
  Syntax: git config --global merge.tool <toolname>

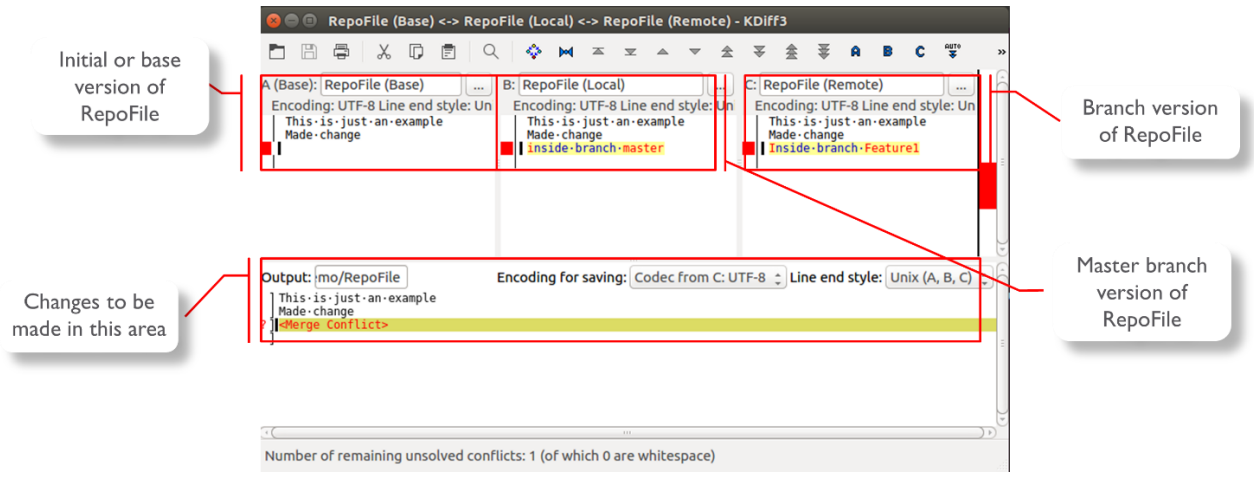
```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git config --global merge.tool kdiff3
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

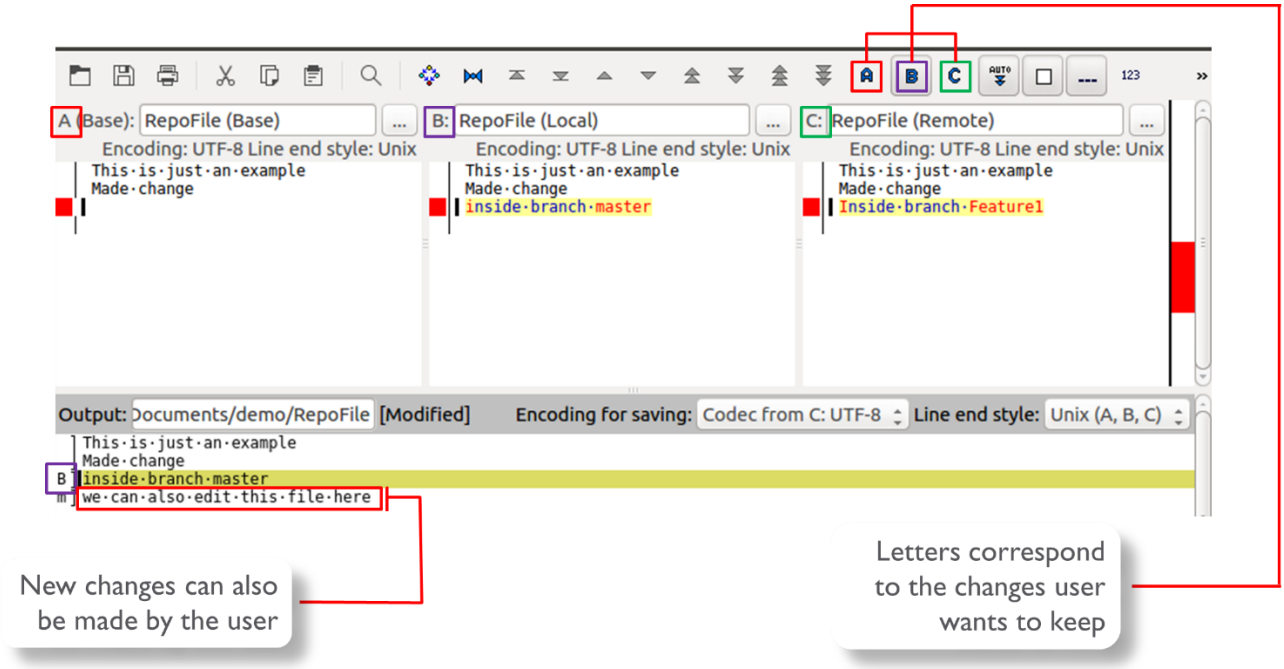
- Merge-tool automatically detects the conflicts and displays them
  Syntax: git mergetool

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git mergetool
```









- Commit the resolved changes
- Merge the branch again

```
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git commit -a -m 'resolved merge conflict'
[master b166145] resolved merge conflict
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git merge feature1
Already up-to-date.
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$ git branch --merged
  feature1
* master
  merge
amrinderjeet@amrinderjeet-VirtualBox:~/Documents/demo$
```

Stashing:

- To Create a stash of your current working directory
  Syntax: git stash save ‘message’

- To list all the saved stashes
  Syntax: git stash list
- Stash list uses a stack structure to save the list

```
amr@amr-VirtualBox:~/Documents/demo$ git checkout master
Switched to branch 'master'
amr@amr-VirtualBox:~/Documents/demo$ git stash list
stash@{0}: On feat1: Saving changes made to the RepoFile
amr@amr-VirtualBox:~/Documents/demo$
```

- Saved stashes can be applied at anytime on any branch
  Syntax: git stash apply <stash id>
- After applying a stash it can still be accessed elsewhere because it remains in the stash

```
amr@amr-VirtualBox:~/Documents/demo$ git stash apply stash@{0}
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   RepoFile

no changes added to commit (use "git add" and/or "git commit -a")
amr@amr-VirtualBox:~/Documents/demo$
```

- Pop command can be used to apply the most recent stash and removing it from the stash stack
  Syntax: git stash pop

```
amr@amr-VirtualBox:~/Documents/demo$ git stash list
stash@{0}: On master: Updated new stash
stash@{1}: On feat1: Saving changes made to the RepoFile
amr@amr-VirtualBox:~/Documents/demo$
```

```
amr@amr-VirtualBox:~/Documents/demo$ git stash pop
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   RepoFile

no changes added to commit (use "git add" and/or "git commit -a")
Dropped refs/stash@{0} (4bbae06658e7435e16ea9a29f00c254526c1e788)
amr@amr-VirtualBox:~/Documents/demo$ git stash list
stash@{0}: On feat1: Saving changes made to the RepoFile
amr@amr-VirtualBox:~/Documents/demo$
```

- Stashes can be deleted from the stash list
  Syntax: git stash drop <stack id>

```
amr@amr-VirtualBox:~/Documents/demo$ git stash drop stash@{0}
Dropped stash@{0} (6275777018dcd9bd801d02172bb45feb9538cebf)
amr@amr-VirtualBox:~/Documents/demo$
```

- The entire stack can also be deleted using one comman
  Syntax: git stash clear

```
amr@amr-VirtualBox:~/Documents/demo$ git stash clear
amr@amr-VirtualBox:~/Documents/demo$
```

Rebasing, reverting and resseting:

- To Rebase a branch
  Syntax: git rebase <rebase branch>

```
amr@amr-VirtualBox:~/Documents/demo$ git rebase master
First, rewinding head to replay your work on top of it...
Fast-forwarded feat2 to master.
amr@amr-VirtualBox:~/Documents/demo$
```

- Revert or **undo** the changes made in the previous commit

- New commit is created without the changes made in the other commit
  Syntax: git revert <commit id>
- Old commit still resides in the history

```
amr@amr-VirtualBox:~/Documents/demo$ git revert HEAD
[feat2 a50d898] Revert "revert edit"
 1 file changed, 1 deletion(-)
amr@amr-VirtualBox:~/Documents/demo$
```

- Reset command can be used to undo changes at different levels
- Modifiers like --hard, --soft and --mixed can be used to decide the degree to which to reset
  Syntax: git reset <modifier> <commit id>

```
amr@amr-VirtualBox:~/Documents/demo$ git reset v1.5
Unstaged changes after reset:
M       RepoFile
amr@amr-VirtualBox:~/Documents/demo$
```