Full Stack Development
Lab Management 3

Branching and merging

1. Lab objectives

This lab is intended for you to get a solid understanding of working with branches

2. Creating the initial repository

Create a new git repository and add several versions of a test file by adding a new line to each version.

```
D:\gitlab>git init
Initialized empty Git repository in D:/gitlab/.git/

… three version of test.txt created and committed …

D:\gitlab>git ll
* 36a088b (HEAD -> master) third
* a7d0182 second
* 1542feb intial

D:\gitlab>type test.txt
This is a first version
This is a second version
This is a third version
```

3. List and create a development branch

The git branch command lists all of the branches and shows the current branch that is checked out in the working directory. You can also create a new branch using the command with the name of the new branch specified

```
D:\gitlab>git branch
* master

D:\gitlab>git branch dev

D:\gitlab>git branch
  dev
* master
```

Check the status of the new dev branch. Notice that is currently pointing to the same commit as the main branch.

```
D:\gitlab>git ll
* 36a088b (HEAD -> master, dev) third
* a7d0182 second
* 1542feb intial
```

Switch to the dev branch by checking it out. Add a line to the test file and commit it.

```
D:\gitlab>git checkout dev
Switched to branch 'dev'

D:\gitlab>git branch
* dev
  master

D:\gitlab>type test.txt
This is a first version
This is a second version
This is a dev branch version

D:\gitlab>git add .

D:\gitlab>git commit -m "dev changes"
[dev bfafd0d] dev changes
 1 file changed, 1 insertion(+), 1 deletion(-)

D:\gitlab>git ll
* bfafd0d (HEAD -> dev) dev changes
* 36a088b (master) third
* a7d0182 second
* 1542feb intial
```

At this point, the dev branch is said to be a commit ahead of the main branch.

Check out each branch in turn and look at the differences in the files

```
D:\gitlab>git ll
* bfafd0d (HEAD -> dev) dev changes
* 36a088b (master) third
* a7d0182 second
* 1542feb intial

D:\gitlab>type test.txt
This is a first version
This is a second version
This is a dev branch version

D:\gitlab>git checkout master
Switched to branch 'master'

D:\gitlab>git ll
* bfafd0d (dev) dev changes
* 36a088b (HEAD -> master) third
* a7d0182 second
* 1542feb initial

D:\gitlab>type test.txt
This is a first version
This is a second version
This is a third version
```

4. Fast forward merge

Since no changes were made on the master branch, merging the changes from dev into the master branch must means updating the commit that master points to. To do a merge, you should have the branch you are merging into checked out.

Once the dev branch has served its purpose, it can be deleted. Note that nothing is lost, you are just removing the pointer to the commit.

```
D:\gitlab>git branch
  dev
* master


D:\gitlab>git merge dev
Updating 36a088b..bfafd0d
Fast-forward
 test.txt | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)

D:\gitlab>git ll
* bfafd0d (HEAD -> master, dev) dev changes
* 36a088b third
* a7d0182 second
* 1542feb initial

D:\gitlab>git branch
  dev
* master

D:\gitlab>git branch -d dev
Deleted branch dev (was bfafd0d).

D:\gitlab>git branch
* master
```

5. Merge conflicts

This happens when conflicting changes take places in two different branches which are then merged. To set up this scenario, first create another dev branch, and check it out.

```
D:\gitlab>git branch dev2

D:\gitlab>git branch
  dev2
* master

D:\gitlab>git checkout dev2
Switched to branch 'dev2'
```

Make conflicting changes, change line 2 for example, in each version of the file.

```
D:\gitlab>git checkout dev2
Switched to branch 'dev2'

D:\gitlab>type test.txt
This is a first version
This is a second version -dev2
This is a dev branch version

D:\gitlab>git add .

D:\gitlab>git commit -m "dev 2 edit"
[dev2 7415ad4] dev 2 edit
 1 file changed, 1 insertion(+), 1 deletion(-)

D:\gitlab>git checkout master
Switched to branch 'master'

D:\gitlab>type test.txt
This is a first version
This is a second version -master
This is a dev branch version

D:\gitlab>git add .

D:\gitlab>git commit -m "master edit"
[master 6a39168] master edit
 1 file changed, 1 insertion(+), 1 deletion(-)
```

If you look at the log files, you can see that there are now diverging branches, we have to non-linear commits.

```
D:\gitlab>git ll
* 6a39168 (HEAD -> master) master edit
| * 7415ad4 (dev2) dev 2 edit
|/
* bfafd0d dev changes
* 36a088b third
* a7d0182 second
* 1542feb intial
```

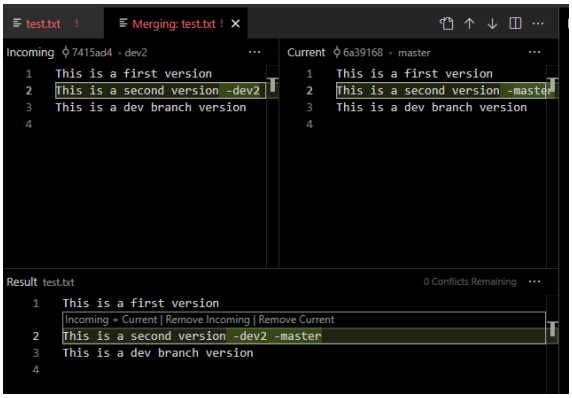
If you try and do a merge, git complains that there are conflicting changes.

```
D:\gitlab>git merge dev2
Auto-merging test.txt
CONFLICT (content): Merge conflict in test.txt
Automatic merge failed; fix conflicts and then commit the result.

D:\gitlab>git s
On branch master
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
        both modified:   test.txt

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

At this point you could just abort the merge. However, since you are still in a merge, the conflicts are temporarily in the file to be merge that has the conflicts.

```
D:\gitlab>type test.txt
This is a first version
<<<<<<< HEAD
This is a second version -master
=======
This is a second version -dev2
>>>>>>> dev2
This is a dev branch version
```

Now edit the file to keep both changes as was shown in the demo.



```
D:\gitlab>type test.txt
This is a first version
This is a second version -dev2 -master
This is a dev branch version

D:\gitlab>git s
On branch master
All conflicts fixed but you are still merging.
  (use "git commit" to conclude merge)

Changes to be committed:
        modified:   test.txt


D:\gitlab>git commit -m "merge done"
[master e424023] merge done

D:\gitlab>git ll
*   e424023 (HEAD -> master) merge done
|\
| * 7415ad4 (dev2) dev 2 edit
* | 6a39168 master edit
|/
* bfafd0d dev changes
* 36a088b third
* a7d0182 second
* 1542feb intial
```

Now delete the dev2 branch

End Lab