Lab: Version Control with Git

The goal of this lab is for you to clone a remote repository, apply some changes, commit these changes as well as merge branches together again.

Please follow through with the following operations

1. Clone the remote repository from https://github.com/squ4rks/se-for-ase-sample_code
2. In the folder 00_git_lab create a new text file, insert your favorite cisco BU and add it to the repository
3. Commit the changes.

Next we are going to create a branch. Please

1. Create a new branch called "feature"
2. Add (as in append) another line to your previously created text file and commit it
3. Switch back to the master branch (and observe that the file changes)
4. Merge your new feature branch into the master branch

Lastly, we want to resolve a merge conflict. A merge conflict occurs when git is not able to automatically merge two branches because the changes occur on the same line. We will now artificially create such a merge conflict (and then resolve it).

1. On your master branch, create a new branch called "conflict"
2. In this new conflict branch change your favorite BU to something else (Collaboration is a great choice). Add the file and commit the change.
3. Switch over to your feature branch and change your favorite BU to something different (maybe IoT?). Add the file and commit the change.
4. On your feature branch try to merge the conflict branch into your feature branch
5. To resolve the conflict you will have to manually decide which lines from which version(or branch) you want to keep. Git will indicate all conflicting changes with "<<<<<<<" and "=======" signs.
6. Resolve the issue (don't forget to remove all of gits additional things like the branch names and "<<<<<<" etc.
7. Add the file to your commit and commit it. Congratulations. You just resolved a merge conflict.

Additional tasks in case you are already done

1. Check the difference between a normal merge and a fast-forward merge. Create two branches and then merge them without fast-forward. Any idea why this might be desirable?
2. Create some changes to your repository (don’t commit them), stash the changes and then apply (“pop”) them on another branch.
3. Instead of merging you can also rebase a branch. Create a diverging branch and rebase it into master instead of merging it.
4. Reset your repository to a previous clean working state by fetching all remote information and resetting your HEAD