Lab1 - Git

Set global config and create local repository

1. Set your global name and email
   git config --global user.name "Hamdi Brahim"
   git config --global user.email "brahim.hamdi.consult@gmail.com"
2. Create a local folder (Project1)
3. Initialize GIT repository in the folder.
   git init
   What is in .git new create folder ?

First lab

Continuing on the same repository:

1. Create the following tree :
   README.md
   MyConsoleApp
   | ***console.txt***
   MyWebApp
   | ***web.txt***
   | ***code.txt***
2. Check the state of Git repository
   git status
3. Add README.md to « staging area »
   git add README.md
   Then check the state of the local repository again.
4. Add README.md to the local repository as the first commit:
   git commit -m « My first lab - commit 1 »
5. Add all files into GIT repository as a single commit.
   git commit -am « My first lab – commit without staging !! »
   What's the problem ? How to solve it ?
6. Add this line to each file in your repository:
   « DevOps training – Git workshop »
7. Commit all changes as a single commit (commit msg : *My first lab commit 2*).
   git commit -am « My first lab commit 2 »
8. Show the commit history :
   git log
   What this command show ?
9. Create a « .gitignore » and « doc » files
   Check the state of local repository.

10. Add « doc » to « .gitignore » as first line, and then check the state of local repository again. What's the difference before and after adding « doc » to « gitignore » ? Why ?
    Commit changes.

Removing a file from a repository

Continuing on the same repository:

1. Remove the file in staging area:
   git rm MyConsoleApp/console.txt
   show the state of local repository.

   Since the « MyConsoleApp » should be empty, it should disappear from disk, verify with this system commad :
   ls -l
2. Check how the staging area looks like:
   git status
3. To undo the file deletion prior to commit, use following set of commands:
   git reset HEAD MyConsoleApp/console.txt
   Followed by:
   git checkout -- MyConsoleApp/console.txt
4. Remake command 1, and then commit changes
   git rm MyConsoleApp/console.txt
   git commit -m « commit removed file »
5. Revert to last commit (last version).
   git revert HEAD

Working with branches

Continuing on the same repository:

1. List branches :
   git branch
   asterisk marking currently active branch.
2. Create a new branch:
   git branch my_aple_app
   List branches.
3. Rename the new branche :
   git branch -m my_aple_app my_apple_app
   List branches
4. delete the branch :
   git branch -d my_apple_app
5. Create and switch to new branch
   git checkout -b my_apple_app
   git branch

6. Implement OS X version and commit it
   mkdir MyAppleApp
   echo "OS X implementation" > MyAppleApp/osx.txt
   git add MyAppleApp
   git commit -m "OS X version"
7. switch back to master
   git checkout master
   git branch
   Is `MyAppleApp` folder and its content in the working tree ? Why ?
   ls -l

Reviewing the repository history

1. Show one commit per line :
   git log --oneline
2. commit history of my_apple_app branch
   git log my_apple_app
3. commit history of code.txt file
   git log --oneline MyWebApp/code.txt
4. commit history of MyWebApp folder
   git log --oneline MyWebApp

Merging branches and conflict detecting (cloned repository)

1. Preparation
   git clone https://github.com/DevTrainings/test_merge_conflict.git
   cd test_merge_conflict
2. Join two or more development histories together
   git log --all --graph --oneline --decorate
3. The file `file` was changed in both branches `bar` and `foo` and we want to get those changes back into the master.
   git branch
   cat file
4. Merge bar to master
   git merge bar
   git log --all --graph --oneline --decorate
   cat file
5. do the same with `foo`
   git merge foo
   conflict !!

cat file

6. Resolve the conflict then,
   git add file
   git commit -m "Merge branch 'foo'"
7. Merge foo ok ?
   git log --all --graph --oneline –decorate

Working with remote repository

1. Preparation:

   git clone https://github.com/DevTrainings/premade_remote.git

   cd premade_remote

   How many branches are in the repository ?
2. What is the content of every branch ?
3. Create an account on Github.

   Create a new empty repository on your remote GitHub repository.

   (example : repos1).
4. Setup the remote repository for the local repository:

   git remote add origin https://github.com/brahimhamdi/repos1
5. Publish your entire repository to the server:

   git push origin master
6. remove the current mapping:

   git remote remove origin

   origin defines the name of remote.
7. Pull from another remote repository (for another user)