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.

   What is in .git new create folder ?

First lab

Continuing on the same repository.

1. Create the following tree :

```
hasher
└── hasher.rb
README.md
rng
└── rng.py
webui
├── files
└── webui.js
worker
└── worker.py
```

2. Check the state of Git repository

3. Add README.md to « staging area »

   Then check the state of the local repository again.

4. Add README.md to the local repository as the first commit.

5. Add all files into GIT repository as a single commit (without git add).

   What's the problem ? How to solve it ?

6. Add these lines to worker/worker.py :
   import logging

```
import os
from redis import Redis
import requests
import time
```

Then show the state.

7. Commit all changes as a single commit.

8. Show the commit history.

   What this command show ?

9. Create a « .gitignore » and « architecture.pdf » files

   Check the state of local repository.

10. Add « architecture.pdf » to « .gitignore » as first line, and then check the state of local repository again.

    What's the difference before and after adding « architecture.pdf » to « .gitignore » ? Why ? Commit changes.

Removing a file from a repository

Continuing on the same repository:

1. Remove worker/worker.py with git rm command.

   show the state of local repository.

   Since the « worker » directory should be empty, it should disappear from disk, verify with system commad.

2. Check how the staging area looks like.

3. Undo the file deletion prior to commit.

   Is the file in the workdir ? How to resove problem ?

4. Remake command 1, and then commit changes

5. Revert to last commit (last version).

Working with branches

Continuing on the same repository:

1. List branches.

   asterisk marking currently active branch.
2. Create a new branch *my_aple_app.*

   List branches.

3. Rename the new branche to *my_apple_app.*

   List branches

4. delete the branch.

5. Create and switch to new branch my_apple_app in one command.

6. Implement OS X version and commit it

7. Switch back to master and verfy if new changes in workdir.

Reviewing the repository history

On the same local repository :

1. Show all commit history (one per line).

2. Show commit history of my_apple_app branch

3. Show commit history of worker/worker.py file.

Merging branches and conflict detecting (cloned repository)

1. Quit the Projet1 workdir, then clone this remote repository :
   https://github.com/DevTrainings/test_merge_conflict.git
   Change to *test_merge_conflict* directory

   How many branches in this repository ?

2. The file `file` was changed in both branches `bar` and `foo` and we want to get those changes back into the master.
   What's the content of `file` in master, foo and bar branches ?

3. Merge bar to master.

   What the content of `file` in master branch now ?

4. Do the same with `foo`

   What happened ? Is the merge completed ?

5. Resolve manually the conflict then commit chages.

   Is merge foo ok ?

Collaborative working on remote repository

On the Project1 repository :

1. Create an account on Github.

   Create a new empty repository on your remote GitHub account (example : remote_repos1).

2. Setup the remote repository for the local repository:

3. Publish your entire repository to the server.

4. Quit Project1 repository, then clone remote repository in « collaborator » folder.

   Set your local name and email (différent from the global).

5. Add these line to hasher/hasher.rb :

   require 'digest'
   require 'sinatra'
   require 'socket'

6. Commit changes, then push to remote repository.

7. On the Project1 repository :
   pull the changes from remote repository.