

UNIVERSITÉ TECHNOLOGIE D'HAÏTI

(UNITECH)

Faculte de Genie et d'architechture

Departement des Sciences Informatiques

Niveau: L4

Prepare par: Jacquesima Jean Raynold

Propose par: Ismael Saint Amour

Dans le cadre du cours sécurité et cybersécurité Informatique

Date: 21-12-25

I. Description des résultats de la tâche

a) Expliquer en quelques lignes ce que tu devais faire (objectif du TD)

L'objectif de ce travail dirigé était d'apprendre à utiliser Git et GitHub pour gérer un projet informatique. Il fallait comprendre comment suivre les modifications d'un projet, enregistrer les versions et héberger les fichiers en ligne sur GitHub afin de pouvoir les partager et les sauvegarder..

. b) Décrire la démarche suivie

Pour réaliser ce TD, nous avons d'abord installé et configuré Git sur l'ordinateur. Ensuite, nous avons créé un compte GitHub et généré une clé SSH pour connecter notre machine à GitHub. Après cela, nous avons créé un dépôt GitHub nommé TD, que nous avons cloné sur le bureau.

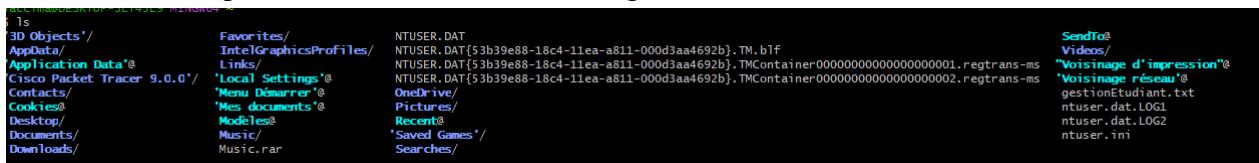
Nous avons ensuite créé les dossiers demandés, ajouté le rapport en PDF et les images. Enfin, nous avons utilisé les commandes git add, git commit et git push pour envoyer tous les fichiers vers GitHub.II.. Résultat de l'exécution des commandes

```
Jacquesima@DESKTOP-3L14JL9 MINGW64 ~
$ git config --global user.email "jacquesimajeanraynold@gmail.com"
```

Elle permet de configurer l'identité de l'utilisateur dans le Git.

```
Jacquesima@DESKTOP-3L14JL9 MINGW64 ~
$ git config --list
http.sslcainfo=C:/Program Files/Git/mingw64/etc/ssl/certs/ca-bundle.crt
http.sslbackend=openssl
diff.astextplain.textconv=astextplain
credential.helper=manager
core.autocrlf=true
core.fscache=true
core.symlinks=false
pull.rebase=false
credential.https://dev.azure.com.usehttppath=true
init.defaultbranch=master
user.email=jacquesimajeanraynold@gmail.com
```

Elle permet d'afficher toutes les configurations Git actives sur la machine .



La commande ls permet d'afficher la liste des fichiers présentes dans le répertoire courant.

```
Jaccima@DESKTOP-3LT4JL9 MINGW64 ~
$ cd desktop

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/desktop
$ mkdir projet-git

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/desktop
$ cd projet-git

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/desktop/projet-git
$ ^C

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/desktop/projet-git
$ git init
Initialized empty Git repository in C:/Users/Jaccima/Desktop/projet-git/.git/

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/desktop/projet-git (master)
$ echo "Hello git!" > README.md

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/desktop/projet-git (master)
$ git add README.com
fatal: pathspec 'README.com' did not match any files

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/desktop/projet-git (master)
$ git add README.md
warning: in the working copy of 'README.md', LF will be replaced by CRLF the next time Git touches it

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/desktop/projet-git (master)
$ git commit -m "premier avec powershell"
[master (root-commit) 3cb7657] premier avec powershell
 1 file changed, 1 insertion(+)
 create mode 100644 README.md

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/desktop/projet-git (master)
$ git status
On branch master
nothing to commit, working tree clean
```

À travers ce travail, nous avons utilisé les commandes de base de Git afin de gérer un projet de manière efficace. Nous avons commencé par nous déplacer dans les répertoires à l'aide de la commande cd, puis nous avons initialisé un dépôt Git avec la commande git init. Ensuite, un fichier README.md a été créé à l'aide de la commande echo, puis ajouté au dépôt grâce à la commande git add.

La commande git commit nous a permis d'enregistrer le premier état du projet avec un message descriptif, tandis que la commande git status a servi à vérifier l'état du dépôt et à s'assurer que toutes les modifications ont bien été prises en

compte. Ce travail nous a permis de comprendre les principes fondamentaux du contrôle de version et l'importance de Git dans la gestion des projets informatiques.

```
PS C:\Windows\system32> ssh-keygen -t ed25519 -C "jacquesimajeanraynold@gmail.com"
Too many arguments.
usage: ssh-keygen [-q] [-a rounds] [-b bits] [-C comment] [-f output_keyfile]
                  [-m format] [-N new_passphrase] [-O option]
                  [-t dsa | ecdsa | ecdsa-sk | ed25519 | ed25519-sk | rsa]
                  [-w provider] [-Z cipher]
  ssh-keygen -p [-a rounds] [-f keyfile] [-m format] [-N new_passphrase]
              [-P old_passphrase] [-Z cipher]
  ssh-keygen -i [-f input_keyfile] [-m key_format]
  ssh-keygen -e [-f input_keyfile] [-m key_format]
  ssh-keygen -y [-f input_keyfile]
  ssh-keygen -c [-a rounds] [-C comment] [-f keyfile] [-P passphrase]
  ssh-keygen -l [-v] [-E fingerprint_hash] [-f input_keyfile]
  ssh-keygen -B [-f input_keyfile]
  ssh-keygen -D pkcs11
  ssh-keygen -F hostname [-lv] [-f known_hosts_file]
  ssh-keygen -H [-f known_hosts_file]
  ssh-keygen -K [-a rounds] [-w provider]
  ssh-keygen -R hostname [-f known_hosts_file]
  ssh-keygen -r hostname [-g] [-f input_keyfile]
  ssh-keygen -M generate [-O option] output_file
  ssh-keygen -M screen [-f input_file] [-O option] output_file
  ssh-keygen -I certificate_identity -s ca_key [-hU] [-D pkcs11_provider]
              [-n principals] [-O option] [-V validity_interval]
              [-z serial_number] file ...
  ssh-keygen -L [-f input_keyfile]
  ssh-keygen -A [-a rounds] [-f prefix_path]
  ssh-keygen -k -f krl_file [-u] [-s ca_public] [-z version_number]
              file ...
  ssh-keygen -Q [-l] -f krl_file [file ...]
  ssh-keygen -Y find-principals -s signature_file -f allowed_signers_file
  ssh-keygen -Y match-principals -I signer_identity -f allowed_signers_file
  ssh-keygen -Y check-novalidate -n namespace -s signature_file
  ssh-keygen -Y sign -f key_file -n namespace_file [-O option] ...
  ssh-keygen -Y verify -f allowed_signers_file -I signer_identity
              -n namespace -s signature_file [-r krl_file] [-O option]
```

La commande ssh-keygen -t ed25519 -C permet de générer une paire de clés SSH sécurisées afin d'authentifier l'utilisateur lors de connexions distantes, notamment pour l'accès à GitHub sans utilisation de mot de passe.

```
Jaccima@DESKTOP-3LT4JL9 MINGW64 ~
$ ssh-keygen -t ed25519 -C "jacquesimajeanraynold@gmail.com"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/c/Users/Jaccima/.ssh/id_ed25519):
Created directory '/c/Users/Jaccima/.ssh'.
Enter passphrase for "/c/Users/Jaccima/.ssh/id_ed25519" (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /c/Users/Jaccima/.ssh/id_ed25519
Your public key has been saved in /c/Users/Jaccima/.ssh/id_ed25519.pub
The key fingerprint is:
SHA256:Y2l9BdCwBs+kkerG242YUC9CAJqu1FF9KKNS4qMkzXo jacquesimajeanraynold@gmail.com
The key's randomart image is:
+--[ED25519 256]--+
|... . o.++. |
|o.+o o ..B ... |
|+*..o .... = . |
|+=.. o + . |
|==.. + .S . . |
|= E o =o.. . |
|.. + * o |
| + o . |
+---[SHA256]
```

La commande ssh-keygen -t ed25519 -C permet de générer une paire de clés SSH sécurisées associées à une adresse e-mail, afin de permettre une authentification sécurisée lors des échanges avec des plateformes comme GitHub.

```
Jaccima@DESKTOP-3LT4JL9 MINGW64 ~
$ cat ~/.ssh/id_ed25519
-----BEGIN OPENSSH PRIVATE KEY-----
b3B1bnNzaC1rZXktdjEAAAAABG5vbmlUAAAEBm9uZQAAAAAAAAAAwAAAAtzc2gtZW
QyNTUxOQAAACBg8xgD7Ak/HRxKFhNCM3PSXFyGwa1QKCFqiwXReAt0gwAAAKiscj5zrHI+
cwAAAtzc2gtZWQyNTUxOQAAACBg8xgD7Ak/HRxKFhNCM3PSXFyGwa1QKCFqiwXReAt0gw
AAAECHexyf0A2BY2e0/VKtBf55ER+xLRnGwEaEwVFkuyRCgWDzGAPsCT8dHEoWE0Izc9Jc
XIbBrVAoIWqlBdF4C3SDAAAAH2phY3F1ZXNpbWFqZWFncmF5bm9sZEbnbWFpbC5jb20BAG
MEBQY=
-----END OPENSSH PRIVATE KEY-----
```

La commande \$ cat ~/.ssh/id_ed25519 affiche la clé privée SSH, qui permet de s'authentifier sur un serveur ou GitHub. Cette clé est très sensible et ne doit jamais être partagée. Pour se connecter ou configurer GitHub, on utilise plutôt la clé publique (id_ed25519.pub).

```
Jaccima@DESKTOP-3LT4JL9 MINGW64 ~
$ cat ~/.ssh/id_ed25519.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIGDzGAPsCT8dHEoWE0Izc9JcXIbBrVAoIWqlBdF4C3SD jacquesimajeanraynold@gmail.com
```

\$ cat ~/.ssh/id_ed25519.pub affiche la clé publique SSH, que l'on peut partager pour s'authentifier sur un serveur ou GitHub. Contrairement à la clé privée, elle n'est pas secrète.

```
Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ ls
CV RODDLEY.docx*
CV X.docx*
Cisco Packet Tracer.lnk*
CorelDRAW Graphics Suite 2024 v25.0.0.230 (x64) + Fix {CrackHash}/*
IMAJ.jpg
InShot_20251209_142819135.png
'Osna Kesnel.pdf'
'Personal - Edge.lnk'*
'RAPPORT.DEV WOOD FINAL.pdf'
'RAPPORT.DEV WOOD.docx'
'RAPPORT.DEV WOOD.pdf'
'RAPPORT.DEVFINAL.pdf'
'RAPPORT.DEVFINAL.pdf'
'Rapport_mobile_2.pdf'
'Rood CV.pdf'
'Visual Studio 2015'/
'WebApplication1/'
desktop.ini
'devoir IA.docx'
'devoir IA.pdf'
'dossier virtuel kali'/
'jaccima cisco.pkt'
'jaccima cisco.pkt-'
'kotlinCompiler-2.2.21'
'kotlininc/'
'pl_oracle.sql'
'pratique python'/
'projet-git'/
'zx.pdf'
'~$voir IA.docx'
```

La commande ls permet d'afficher la liste des fichiers présentes dans le répertoire courant.

```
Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ cd mon-projet
bash: cd: mon-projet: No such file or directory

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ echo "Système d'exploitation!" > module.text

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ git add .
Fatal: not a git repository (or any of the parent directories): .git

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ git status
Fatal: not a git repository (or any of the parent directories): .git

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ git commit -m "Ajouter module"
Fatal: not a git repository (or any of the parent directories): .git

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ git branch -M main
Fatal: not a git repository (or any of the parent directories): .git

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ git push -u origin main
Fatal: not a git repository (or any of the parent directories): .git

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ cd TD
bash: cd: TD: No such file or directory

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ mkdir système

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ cd système
```

Ces commandes ont permis de :

- Créer une arborescence de dossiers (système, projet, présentation, réseau, image)
 - Se déplacer entre les dossiers avec cd
- Créer un fichier texte Module.text avec du contenu
 - Ajouter les fichiers au dépôt Git avec git add
 - Vérifier l'état du dépôt avec git status

```
Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ mkdir systeme

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ cd systeme

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop/systeme
$ mkdir image

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop/systeme
$ mkdir projet

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop/systeme
$ mkdir presentation

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop/systeme
$ cd ..

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ mkdir reseauI

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop
$ cd reseauI

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop/reseauI
$ mkdir image

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop/reseauI
$ mkdir projet

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop/reseauI
$ mkdir presentation

Jaccima@DESKTOP-3LT4JL9 MINGW64 ~/Desktop/reseauI
$ cd ..
```

Conclusion

Ce travail dirigé nous a permis de découvrir et de pratiquer l'utilisation de **Git** et **GitHub** pour la gestion de projets informatiques. Nous avons appris à créer un dépôt local, à suivre les modifications grâce aux commits, et à envoyer nos fichiers vers un dépôt distant sur GitHub.

.La tâche est **réussie**, car tous les fichiers demandés ont été correctement ajoutés et envoyés sur GitHub. Les principales difficultés rencontrées concernaient la configuration de Git et l'utilisation de certaines commandes, mais elles ont été surmontées en suivant les étapes fournies.

Ce TD nous a donné une base solide pour utiliser Git et GitHub dans nos futurs projets informatiques.