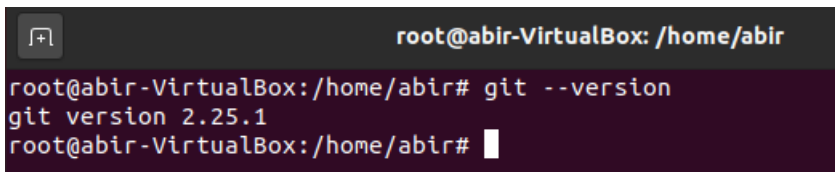


Lab 1. Utilisation de Git

Étape 1 : Installation de Git

1. Ouvrez le terminal Ubuntu.
2. Vérifiez si Git est déjà installé en exécutant la commande

```
git --version
```

A terminal window with a dark background. The prompt is 'root@abir-VirtualBox: /home/abir'. The command 'git --version' has been entered, and the output is 'git version 2.25.1'.

```
root@abir-VirtualBox: /home/abir# git --version
git version 2.25.1
root@abir-VirtualBox: /home/abir#
```

Dans notre cas, git est déjà installé sur notre machine ubuntu dans sa version 2.25.1.

Si Git n'est pas installé, installez-le en utilisant la commande :

```
sudo apt update
sudo apt install git
```

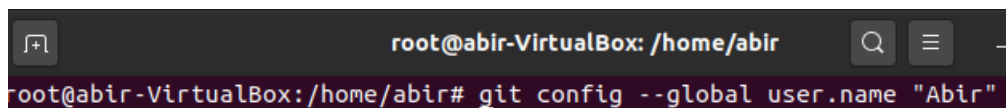
Vérifiez à nouveau la version de Git pour confirmer l'installation :

```
git --version
```

Étape 2 : Configuration initiale de Git

1. Configurez votre nom d'utilisateur Git

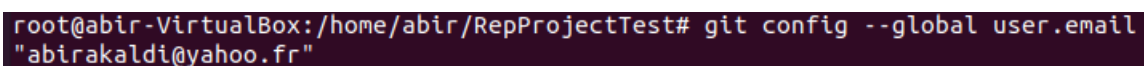
```
git config --global user.name "Votre Nom"
```

A terminal window with a dark background. The prompt is 'root@abir-VirtualBox: /home/abir'. The command 'git config --global user.name "Abir"' has been entered.

```
root@abir-VirtualBox: /home/abir# git config --global user.name "Abir"
```

2. Configurez votre adresse e-mail Git :

```
git config --global user.email "votre@email.com"
```

A terminal window with a dark background. The prompt is 'root@abir-VirtualBox: /home/abir/RepProjectTest#'. The command 'git config --global user.email "abirakaldi@yahoo.fr"' has been entered.

```
root@abir-VirtualBox: /home/abir/RepProjectTest# git config --global user.email
"abirakaldi@yahoo.fr"
```

3. Vérifiez la configuration Git :

```
git config --list
```

```
root@abir-VirtualBox: /home/abir
root@abir-VirtualBox:/home/abir# git config --list
user.name=Abir
user.mail=abirakaldi@yahoo.fr
```

Étape 3 : Création d'un nouveau repository Git

1. Créez un nouveau dossier pour votre projet nommé **RepProjectTest**:

```
mkdir RepProjectTest
cd RepProjectTest
```

```
root@abir-VirtualBox: /home/abir/RepProjectTest
root@abir-VirtualBox:/home/abir# mkdir RepProjectTest
root@abir-VirtualBox:/home/abir# cd RepProjectTest/
root@abir-VirtualBox:/home/abir/RepProjectTest#
```

2. Initialisez un nouveau repository Git dans ce dossier :

```
git init
```

```
root@abir-VirtualBox:/home/abir/RepProjectTest# git init
Dépôt Git vide initialisé dans /home/abir/RepProjectTest/.git/
```

Le projet **RepProjectTest** est bien initialisé avec git .

Nous pouvons explorer les fichiers au niveau du répertoire `.git` avec la commande.

```
ls .git
```

```
root@abir-VirtualBox: /home/abir/RepProjectTest
root@abir-VirtualBox:/home/abir/RepProjectTest# ls .git/
branches  config  description  HEAD  hooks  info  objects  refs
root@abir-VirtualBox:/home/abir/RepProjectTest#
```

Descriptions du contenu de `.git` :

Le répertoire `.git` est le cœur de chaque dépôt Git. Il contient tous les fichiers et métadonnées nécessaires pour suivre l'historique des versions, les branches, les commits, et bien plus encore. Voici une description des fichiers et répertoires les plus importants à l'intérieur du répertoire `.git` :

1. **HEAD**: Ce fichier pointe vers la branche actuelle. Il contient généralement une référence symbolique (un chemin de fichier) vers le fichier `refs/heads/nom_de_branche` qui représente la branche active.

2. **refs** : Ce répertoire contient des sous-répertoires `refs/heads`, `refs/tags`, et `refs/remotes` qui stockent respectivement les pointeurs vers les branches locales, les tags (étiquettes), et les branches distantes (pour les dépôts distant).

- **refs/heads** : Ce répertoire contient un fichier pour chaque branche locale. Chaque fichier contient le SHA-1 du dernier commit de cette branche.
- **refs/tags** : Ce répertoire stocke les tags, qui sont des points fixes dans l'historique des commits pour marquer des versions spécifiques.

- **refs/remotes** : Si vous travaillez avec des dépôts distants, ce répertoire contient des références aux branches distantes, telles que `refs/remotes/origin/nom_de_branche`.

3. **objects** : Ce répertoire stocke tous les objets Git, y compris les commits, les arbres, et les blobs (les fichiers en eux-mêmes).

- **objects/commit**: Contient les commits (chaque commit a un fichier avec son SHA-1).
- **objects/tree**: Contient les arbres, qui sont des structures de données Git qui représentent l'état du projet à un moment donné.
- **objects/blob**: Contient les blobs, qui sont les données réelles des fichiers.

4. **config**: Ce fichier contient la configuration du dépôt Git, y compris les informations sur l'utilisateur (nom, adresse e-mail) et d'autres paramètres de configuration spécifiques au dépôt.

5. **description**: Ce fichier contient une courte description du dépôt, généralement utilisée par les serveurs Git.

6. **hooks** : Ce répertoire peut contenir des scripts de hook, qui sont des scripts exécutés à des moments spécifiques lors des opérations Git (par exemple, avant un commit).

7. **index** : Ce fichier est le staging area (zone de préparation). Il contient des informations sur les fichiers qui sont prêts à être commités dans le prochain commit.

8. **logs** : Ce répertoire peut contenir des fichiers de logs qui enregistrent l'historique des références (par exemple, `refs/heads/nom_de_branche`) et les opérations effectuées sur elles.

9. **info** : Ce répertoire peut contenir des fichiers supplémentaires de configuration et d'informations.

🌈 Ces fichiers et répertoires forment l'infrastructure interne de Git qui permet de gérer efficacement les versions et de suivre l'historique des commits dans un projet.

Étape 4 : Ajout de fichiers et commits

1. Créez un fichier **samplegit.py** pour votre projet **RepProjectTest** et taper le code python suivant :

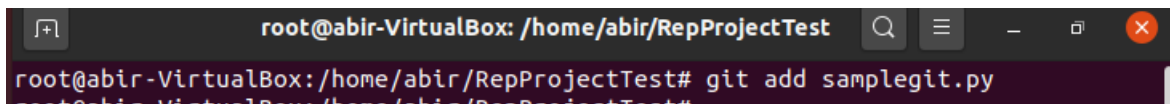
```
gedit samplegit.py
```



```
root@abir-VirtualBox: /home/abir/RepProjectTest
root@abir-VirtualBox:/home/abir/RepProjectTest# gedit samplegit.py
1 print("My first example for git Test")
2
```

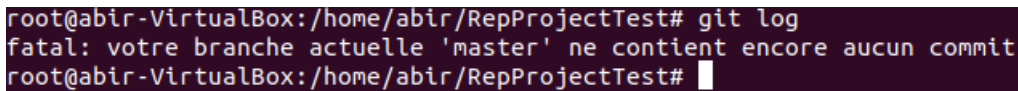
2. Ajoutez le fichier au staging area

```
git add samplegit.py
```



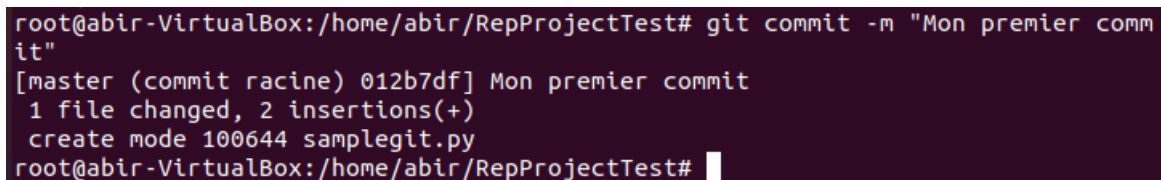
Avant de faire un commit, il faut vérifier que notre projet dans la branche master ne contient aucun commit avec la commande

```
git log
```



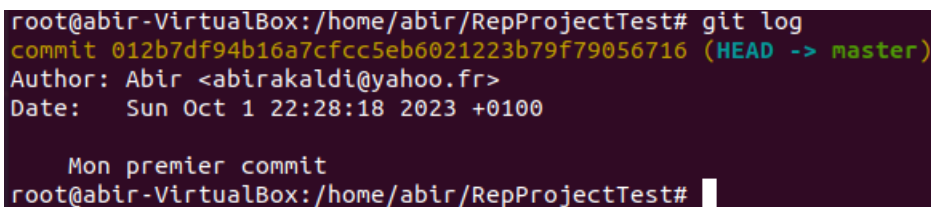
3. Faites un commit pour enregistrer les changements :

```
git commit -m "Premier commit"
```



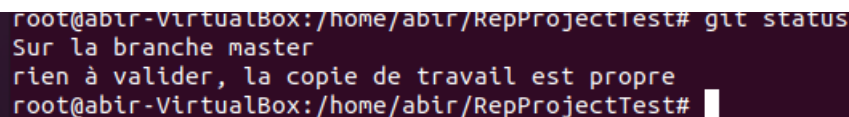
4. Affichez l'historique des commits :

```
git log
```

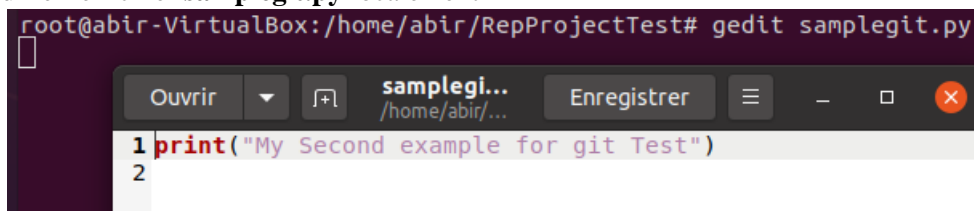


5. Vérifier l'état de la branche master

```
git status
```



6. Modifier le fichier **samplegit.py** localement



Étape 5 : Création de branches

1. Créer une branche nommée developer

```
git branch developer
```

```
root@abir-VirtualBox:/home/abir/RepProjectTest# git branch developer
```

2. Vérifier la création de la branche avec la commande

```
git branch
```

```
root@abir-VirtualBox:/home/abir/RepProjectTest# git branch
developer
* master
```

3. Basculer vers la branche developer avec la commande

```
Git checkout developer
```

```
root@abir-VirtualBox:/home/abir/RepProjectTest# git checkout developer
Basculement sur la branche 'developer'
```

Étape 6 : Connexion de Git avec GitHub

1. Créez un compte GitHub si vous n'en avez pas déjà un.
2. Connectez-vous à GitHub.
3. Créez un nouveau repository GitHub nommé « RepGitTest » en suivant les étapes sur la plateforme.
4. Associez le repository local avec le repository GitHub « RepGitTest » en utilisant la commande suivante (remplacez `votre_utilisateur` et `votre_projet` par vos informations) :

```
git remote add origin https://github.com/votre_utilisateur/RepProjectTest
```

```
root@abir-VirtualBox:/home/abir/RepProjectTest# git remote add origin https://github.com/AbirKaldi/RepProjectTest
```

5. Poussez le code local vers GitHub en utilisant la commande :

```
git push -u origin master
```

```
root@abir-VirtualBox:/home/abir/RepProjectTest# git push origin master
Username for 'https://github.com': AbirKaldi
Password for 'https://AbirKaldi@github.com':
Énumération des objets: 12, fait.
Décompte des objets: 100% (12/12), fait.
Compression des objets: 100% (6/6), fait.
Écriture des objets: 100% (12/12), 1.02 Kio | 1.02 Mio/s, fait.
Total 12 (delta 0), réutilisés 0 (delta 0)
remote:
remote: Create a pull request for 'master' on GitHub by visiting:
remote:   https://github.com/AbirKaldi/RepProjectTest/pull/new/master
remote:
To https://github.com/AbirKaldi/RepProjectTest.git
 * [new branch]      master -> master
```

6. Vous devrez peut-être vous authentifier avec votre nom d'utilisateur et votre token GitHub.
7. Accédez au repository GitHub pour vérifier que vos fichiers ont été poussés avec succès.