

TP 1 – Git et son utilisation avec GitLab

Objectif :

- Dans cet atelier, vous vous entraînerez à cloner un dépôt ; créer, utiliser et fusionner une branche ; éditer et valider un fichier ; et pousser et récupérer des modifications vers et depuis un référentiel distant.

A. Vérifier que Git est installé localement

1. Ouvrez un terminal et tapez la commande `git version` pour voir la version git installée sur votre machine.

```
brahim@Training:~$ git version
git version 2.37.3
brahim@Training:~$
```

Si la sortie affiche un numéro de version, Git est installé. Si ce n'est pas le cas, installez-le sur votre ordinateur en suivant les instructions du lien suivant :

<https://git-scm.com/book/fr/v2/D%C3%A9marrage-rapide-Installation-de-Git>

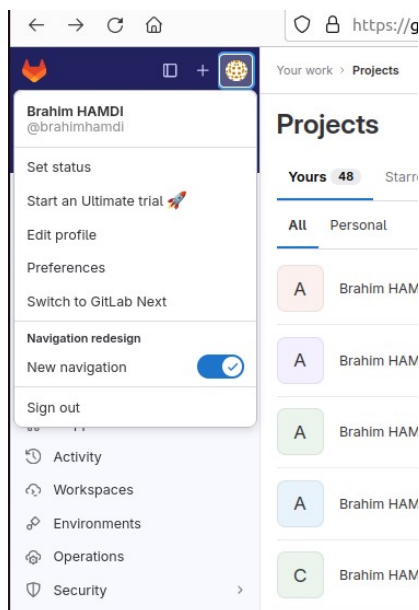
B. Générer une paire de clés SSH

1. Créez une paire de clés (publique et privée) en exécutant cette commande `ssh-keygen` dans votre terminal.
- Lorsque vous y êtes invité, appuyez sur **Enter** pour accepter l'emplacement de la clé par défaut.
 - Lorsque vous y êtes invité, appuyez sur **Enter** pour utiliser une phrase secrète (passphrase) vide.

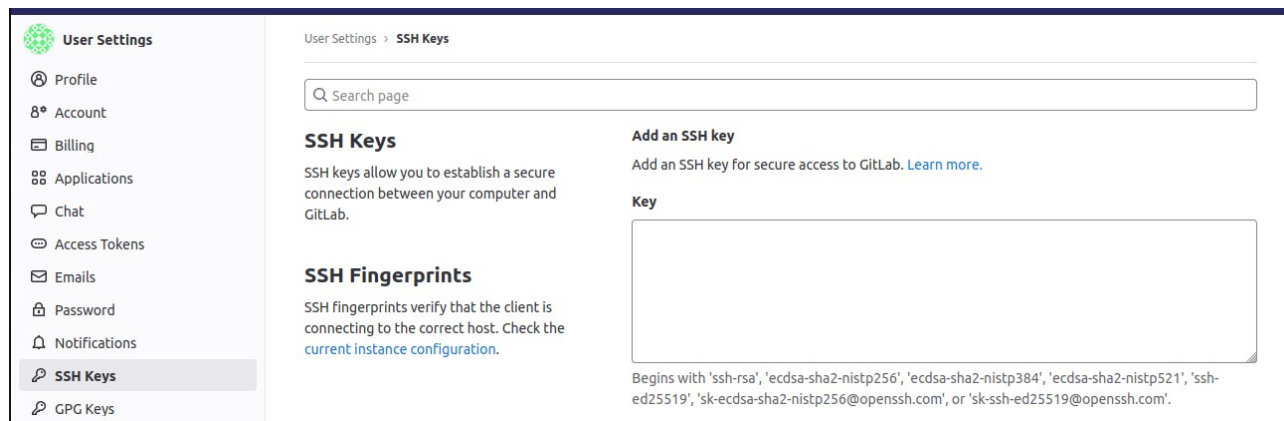
```
brahim@Training:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/brahim/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/brahim/.ssh/id_rsa
Your public key has been saved in /home/brahim/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:BeqjEaiXrkRTPDh/OsDe8nEc3Eb1TR1ucYHX33lv0xk brahim@Training
The key's randomart image is:
+---[RSA 3072]-----+
|      .. .o+=|
| o. .... o.o+|
| o.+ .. .. .o+|
|..+.ooo . .+|
|. =o..+ooS   o|
|oo+ +oo.    Eo|
| +.=.o      .+|
|..o +       + |
|. .         .|
+---[SHA256]-----+
brahim@Training:~$
```

C. Ajouter une clé SSH à votre profil GitLab

1. De retour dans GitLab, dans le coin supérieur gauche, cliquez sur le menu déroulant.



- Dans le menu déroulant, sélectionnez **Modifier le profil**.
- Dans le volet de navigation de gauche, sélectionnez **Clés SSH**.



2. Revenez à votre terminal et listez le contenu du répertoire `.ssh`

```
brahim@Training:~$ ls .ssh
id_rsa  id_rsa.pub  known_hosts  known_hosts.old
brahim@Training:~$
```

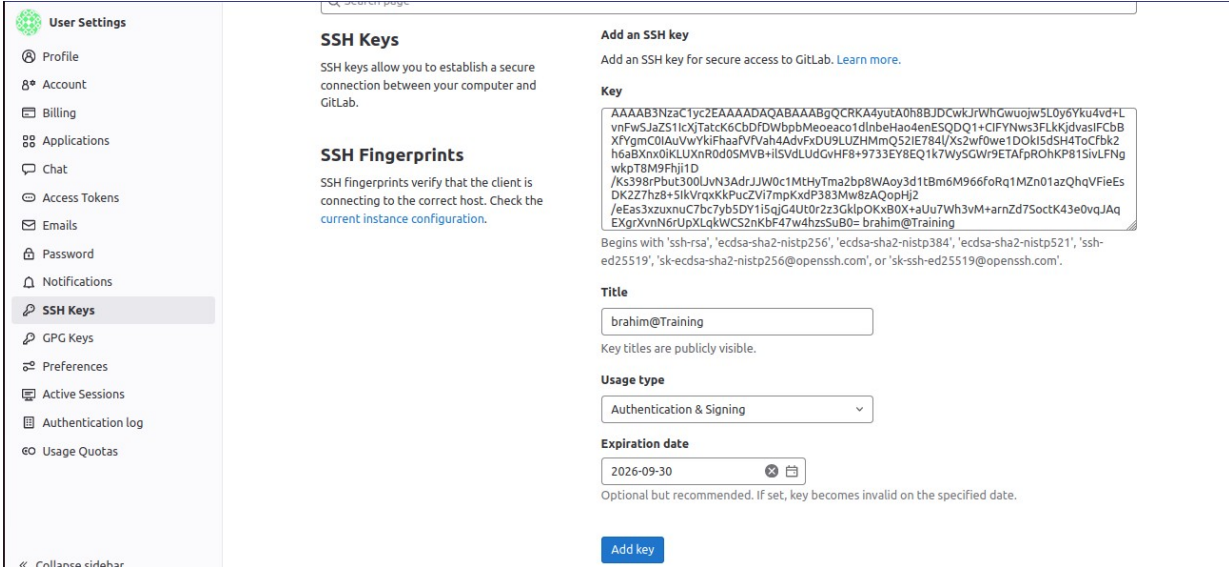
Vous devriez voir deux fichiers clés : une clé publique et une clé privée. La clé publique se termine par `.pub` et est ce que vous devez partager avec GitLab.

- Affichez le contenu de votre clé publique.

```
brahim@Training:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQCCKA4yutA0h8BJDCwkJrWhGwuoJw5L0y6Yku4vd+LvnFwSJaZ51IcXjTatC6CbDFDwbpMeoeaco1dInbeHao4enESQDQ1+CIFYNws3
FLKKjdvasIFCbBXfYgmC0IAuVwYkLFhaafVfVah4AdvFXDU9LUZHMmQ52IE784L/Xs2wf0we1D0K15dSH4ToCfbk2h6aBxnx0LKLUXnR0d0SMVB+lLSVdLudGvHF8+9733EY8EQ1k7WysG
Wr9ETAfPR0hKP81SivLFNgwkpT8M9Fhj1D/Ks398rPbut300LjvN3AdrJJW0c1MhYtMa2bp8WAoy3d1tBm6M966f0Rq1MZn01azQhQVFleESDK2Z7hz8+5IkVrqqKkPucZvi7mpKxdP3
83Mw8ZAQopHj2/eEas3xzuxnuC7bc7yb5DY1i5qjG4U0r2z3GklpOKx80X+aUu7Wh3vM+arnZd7SocTK43e0vqJAQExgrXvnN6rUpXLqkWC52nKbF47w4hzsSuB0= brahim@Training
brahim@Training:~$
```

- Copiez le contenu de `id_rsa.pub` dans votre presse-papiers.

3. De retour dans GitLab.com, collez le contenu de la clé publique dans le champ **Clé**, entrez le titre de votre choix dans le champ **Titre** et sélectionnez **Ajouter une clé**.



The screenshot shows the 'SSH Keys' section of the GitLab user settings. On the left is a sidebar with navigation links: Profile, Account, Billing, Applications, Chat, Access Tokens, Emails, Password, Notifications, **SSH Keys**, GPG Keys, Preferences, Active Sessions, Authentication log, and Usage Quotas. The main content area has two columns. The left column contains 'SSH Keys' (with a description) and 'SSH Fingerprints' (with a description and a link to 'current instance configuration'). The right column is titled 'Add an SSH key' and contains a form. The form has a text area for the 'Key' (containing the copied public key), a text input for the 'Title' (with 'brahim@Training' entered), a dropdown for 'Usage type' (set to 'Authentication & Signing'), and a date picker for 'Expiration date' (set to '2026-09-30'). Below the form is an 'Add key' button.

4. Dans le terminal, testez la connexion ssh :

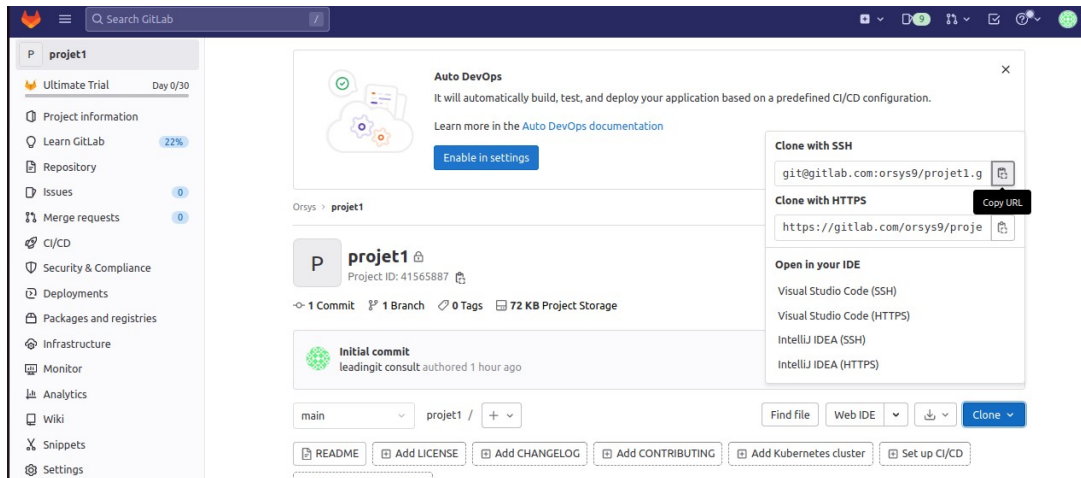
```
brahim@Training:~$ ssh -T git@gitlab.com
The authenticity of host 'gitlab.com (172.65.251.78)' can't be established.
ED25519 key fingerprint is SHA256:eUXGGM1YGsMAS7vkcx6JOJdOGHPem5gQp4taicfCLB8.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitlab.com' (ED25519) to the list of known hosts.
Welcome to GitLab, @leadingit.consult!
brahim@Training:~$
```

Si la commande se termine par un message de bienvenue au lieu d'une erreur, votre connexion SSH est correctement configurée.

D. Cloner un dépôt de projet GitLab sur votre ordinateur local

1. Sur *gitlab.com*, créez un nouveau groupe privé **orsys9** avec le rôle **DevOps Engineer**. Puis créez un nouveau projet blanc **projet1** sous **orsys9**.

- Sélectionnez **Cloner** . Dans la section **Cloner avec SSH** , sélectionnez l'icône **Copier l'URL**.



2. Sur votre terminal, créez un nouveau répertoire **formation** dans le répertoire personnel.

```
brahim@Training:~$ mkdir formation
brahim@Training:~$ cd formation/
brahim@Training:~/formation$
```

- Sous le répertoire formation, copiez l'url (git et pas https) du référentiel distant **projet1** git clonez-le sur votre machine locale.

```
brahim@Training:~/formation$ git clone git@gitlab.com:orsys9/projet1.git
Clonage dans 'projet1'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Réception d'objets: 100% (3/3), fait.
brahim@Training:~/formation$
brahim@Training:~/formation$ ls
projet1
brahim@Training:~/formation$
```

- Déplacez-vous dans le référentiel que vous venez de cloner. Tous les fichiers de ce répertoire seront suivis par Git, et toutes les commandes Git que vous exécutez dans cet atelier doivent être exécutées à partir de ce répertoire.

Affichez le contenu du répertoire, y compris les fichiers cachés et les répertoires commençant par un point. Notez la présence du répertoire **.git**, qui transforme ce répertoire en référentiel Git.

```

brahim@Training:~/formation$ cd projet1/
brahim@Training:~/formation/projet1$ ls
README.md
brahim@Training:~/formation/projet1$
brahim@Training:~/formation/projet1$ ls -a
.  .. .git README.md
brahim@Training:~/formation/projet1$
brahim@Training:~/formation/projet1$ tree .git
.git
├── branches
├── config
├── description
├── HEAD
├── hooks
│   ├── applypatch-msg.sample
│   ├── commit-msg.sample
│   ├── fsmonitor-watchman.sample
│   ├── post-update.sample
│   ├── pre-applypatch.sample
│   ├── pre-commit.sample
│   ├── pre-merge-commit.sample
│   ├── prepare-commit-msg.sample
│   ├── pre-push.sample
│   ├── pre-rebase.sample
│   ├── pre-receive.sample
│   ├── push-to-checkout.sample
│   └── update.sample
├── index
├── info
│   └── exclude
├── logs
│   └── HEAD

```

- Affichez l'état des fichiers de votre référentiel local. Quel est l'état de README.md ?

```

brahim@Training:~/formation/projet1$ git status
Sur la branche main
Votre branche est à jour avec 'origin/main'.

rien à valider, la copie de travail est propre
brahim@Training:~/formation/projet1$ 

```

Vous verrez rien à valider, la copie de travail est propre dans la sortie, ce qui signifie que les fichiers de ce répertoire ont le même contenu que les versions de ces fichiers qui sont stockées dans la base de Git.

E. Travailler sur une branche

1. Listez les branches que vous avez dans le référentiel.

```

brahim@Training:~/formation/projet1$ git branch
* main
brahim@Training:~/formation/projet1$ 

```

Il y a une seule branche qui s'appelle *main*, c'est la branche créée par défaut lorsque vous initialisez un nouveau projet/référentiel.

- Créez une nouvelle branche nommée **branche_temporaire** dans votre référentiel.

```
brahim@Training:~/formation/projet1$ git branch branche_temporaire
brahim@Training:~/formation/projet1$
brahim@Training:~/formation/projet1$ git branch
  branche_temporaire
* main
brahim@Training:~/formation/projet1$
```

Notez bien que la branche active est toujours la branche *main* (marqué par *).

- Basculez sur la nouvelle branche, et vérifiez que c'est la branche en cours.

```
brahim@Training:~/formation/projet1$ git checkout branche_temporaire
Basculement sur la branche 'branche_temporaire'
brahim@Training:~/formation/projet1$
brahim@Training:~/formation/projet1$ git branch
* branche_temporaire
  main
brahim@Training:~/formation/projet1$
```

Maintenant c'est la branche *branche_temporaire* qui est active.

- Listez toutes les branches du référentiel.

```
brahim@Training:~/formation/projet1$ git branch -a
* branche_temporaire
  main
  remotes/origin/HEAD -> origin/main
  remotes/origin/main
brahim@Training:~/formation/projet1$
```

Les branches en rouge se trouvent sur le serveur distant, qu'est l'instance *GitLab* dans votre environnement de formation. L'astérisque (*) indique la branche sur laquelle vous vous trouvez actuellement.

F. Modifier un fichier

1. À l'aide de n'importe quel éditeur de texte, ajoutez cette ligne à la fin de `README.md` et enregistrez le fichier.

une ligne ajoutée localement à branche_temporaire

- Voyez si Git a remarqué que le fichier a été modifié.

```
brahim@Training:~/formation/projet1$ git status
Sur la branche branche_temporaire
Modifications qui ne seront pas validées :
  (utilisez "git add <fichier>..." pour mettre à jour ce qui sera validé)
  (utilisez "git restore <fichier>..." pour annuler les modifications dans le répertoire de travail)
    modifié :      README.md

aucune modification n'a été ajoutée à la validation (utilisez "git add" ou "git commit -a")
brahim@Training:~/formation/projet1$
```


La sortie montre que Git a détecté que vous avez modifié un fichier dans votre référentiel local, mais comme vous n'avez pas *validé* ce fichier, Git n'a pas encore stocké cette modification dans sa base.

G. Ajouter le fichier modifié à la zone de préparation de Git (staging area) et valider les modifications

1. Ajoutez le fichier à la zone de préparation. Si la commande réussit, il n'y aura pas de sortie. Assurez-vous qu'il `README.md` est maintenant prêt à être validé.

```
brahim@Training:~/formation/projet1$ git add .
brahim@Training:~/formation/projet1$
brahim@Training:~/formation/projet1$ git status
Sur la branche branche_temporaire
Modifications qui seront validées :
  (utilisez "git restore --staged <fichier>..." pour désindexer)
    modifié :      README.md
brahim@Training:~/formation/projet1$
```

La commande `git add` ne déplace pas `README.md` sur votre système de fichiers, mais il l'ajoute à la "zone de préparation" de Git.

2. Avant de lancer la première validation, vous devez ajouter vos paramètres Git.
 - Lancez les commandes suivantes pour configurer votre Git (tout en remplaçant les valeurs par les vôtres).

git config --global user.name "Brahim HAMDI"

git config --global user.email "brahim.hamdi.consult@gmail.com"

git config --global core.editor nano

- Validez maintenant par un **commit**. Assurez-vous que la zone de préparation est à nouveau vide.

```
brahim@Training:~/formation/projet1$ git commit -m "Add a line to README.md"
[branche_temporaire abe74a3] Add a line to README.md
 1 file changed, 2 insertions(+)
brahim@Training:~/formation/projet1$ git status
Sur la branche branche_temporaire
rien à valider, la copie de travail est propre
brahim@Training:~/formation/projet1$
```

Vous avez maintenant créé un instantané (*snapshot*) du fichier auquel vous pourrez vous référer ultérieurement, si nécessaire.

H. Poussez vos modifications vers l'instance GitLab

1. Poussez la nouvelle branche **branche_temporaire** dans le référentiel Git distant **projet1** sur le serveur GitLab.

```
brahim@Training:~/formation/projet1$ git remote -v
origin  git@gitlab.com:orsys9/projet1.git (fetch)
origin  git@gitlab.com:orsys9/projet1.git (push)
brahim@Training:~/formation/projet1$
brahim@Training:~/formation/projet1$ git push -u origin branche_temporaire
Énumération des objets: 5, fait.
Décompte des objets: 100% (5/5), fait.
Compression par delta en utilisant jusqu'à 8 fils d'exécution
Compression des objets: 100% (2/2), fait.
Écriture des objets: 100% (3/3), 341 octets | 341.00 Kio/s, fait.
Total 3 (delta 1), réutilisés 0 (delta 0), réutilisés du pack 0
remote:
remote: To create a merge request for branche_temporaire, visit:
remote:   https://gitlab.com/orsys9/projet1/-/merge_requests/new?merge_request%5Bsource_branch%5D=branche_temporaire
remote:
To gitlab.com:orsys9/projet1.git
 * [new branch]      branche_temporaire -> branche_temporaire
la branche 'branche_temporaire' est paramétrée pour suivre 'origin/branche_temporaire'.
brahim@Training:~/formation/projet1$
```

Si vous n'êtes jamais sûrs de la commande exacte pour envoyer vos modifications au serveur distant, tapez `git push` et Git affichera un message d'erreur avec la commande correcte à copier et coller.

I. Modifier, valider et envoyer à nouveau le fichier

1. Dans l'éditeur de texte de votre ordinateur local (et non dans l'éditeur de navigateur de *gitlab.com*), ajoutez cette nouvelle ligne à la fin de votre copie locale de `README.md` et enregistrez le fichier.

Une ligne ajoutée à README.md

- Dans votre terminal, déplacez le fichier modifié vers la zone de préparation de Git. Puis, validez ce changement.

```
brahim@Training:~/formation/projet1$ vim README.md
brahim@Training:~/formation/projet1$ git add README.md
brahim@Training:~/formation/projet1$ git commit -m "Modify README.md"
[branche_temporaire 72baac6] Modify README.md
1 file changed, 2 insertions(+)
brahim@Training:~/formation/projet1$
```

- Affichez l'historique des commits (validations) de votre dépôt.


```

brahim@Training:~/formation/projet1$ git log
commit 72baac6fd7b2c81a4738b9be2e3d4596d07b78ac (HEAD -> branche_temporaire)
Author: Brahim Hamdi <brahim.hamdi.consult@gmail.com>
Date: Sat Dec 3 09:19:18 2022 +0100

    Modify README.md

commit abe74a302033014708b3af5821b91a920f4c25d9 (origin/branche_temporaire)
Author: Brahim Hamdi <brahim.hamdi.consult@gmail.com>
Date: Sat Dec 3 09:04:31 2022 +0100

    Add a line to README.md

commit 87bebe30a3afe95cd5cec4dd20a530c6901b7c5c (origin/main, origin/HEAD, main)
Author: leadingit consult <leadingit.consult@gmail.com>
Date: Sat Dec 3 06:25:20 2022 +0000

    Initial commit
brahim@Training:~/formation/projet1$

```

- Poussez vos mises à jour vers le référentiel distant **projet1** sur le serveur GitLab.

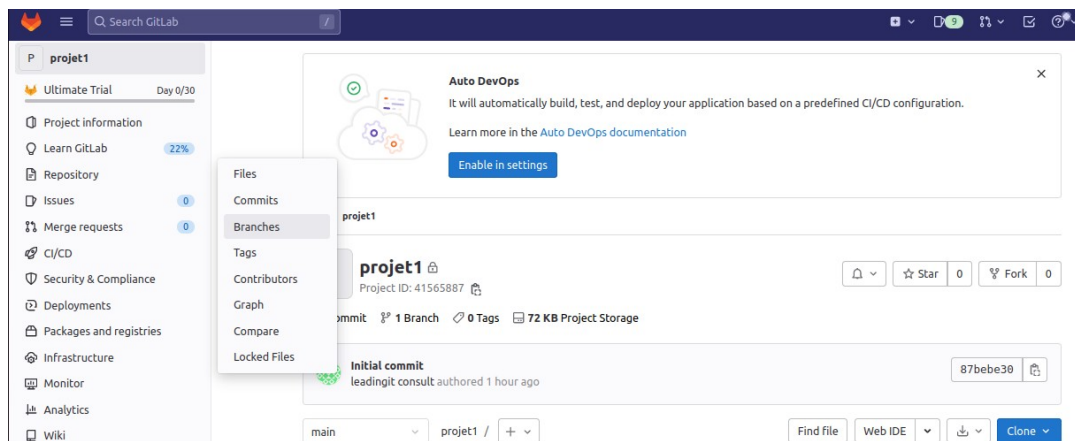
```

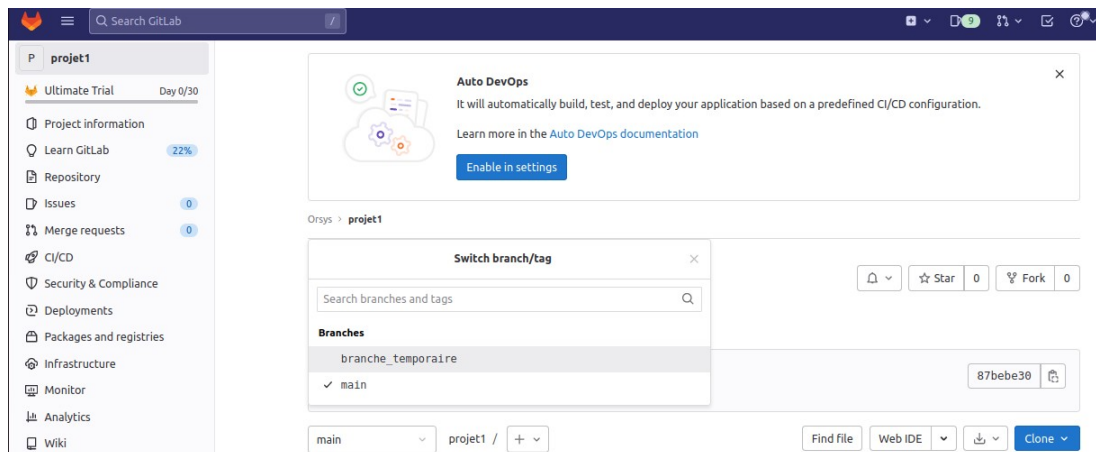
brahim@Training:~/formation/projet1$ git push
Énumération des objets: 5, fait.
Décompte des objets: 100% (5/5), fait.
Compression par delta en utilisant jusqu'à 8 fils d'exécution
Compression des objets: 100% (2/2), fait.
Écriture des objets: 100% (3/3), 320 octets | 320.00 Kio/s, fait.
Total 3 (delta 1), réutilisés 0 (delta 0), réutilisés du pack 0
remote:
remote: To create a merge request for branche_temporaire, visit:
remote: https://gitlab.com/orsys9/projet1/-/merge_requests/new?merge_request%5Bsource_branch%5D=branche_temporaire
remote:
To gitlab.com:orsys9/projet1.git
abe74a3..72baac6  branche_temporaire -> branche_temporaire
brahim@Training:~/formation/projet1$

```

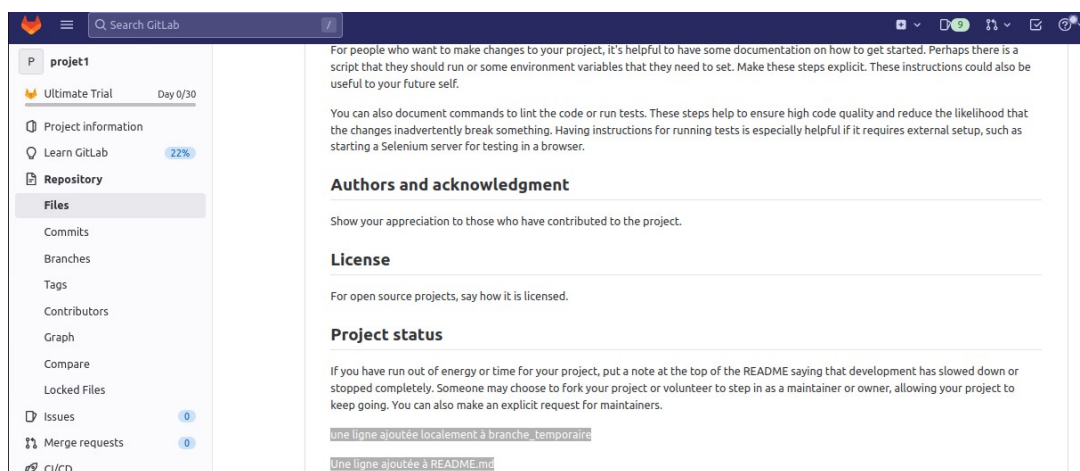
Pour valider vos modifications dans la branche en amont (c'est-à-dire une branche déjà existante sur le référentiel distant portant le même nom que la branche sur votre ordinateur local), vous pouvez simplement exécuter git push. Le système n'a besoin de définir la branche en amont qu'une seule fois.

2. Accédez à votre projet dans GitLab. Une fois sur l'interface principale du projet, accédez au volet de navigation de gauche, sélectionnez **Code** > **Branches**, puis sélectionnez **branche_temporaire** pour basculer vers cette branche.





- Confirmez que les modifications que vous avez apportées à `README.md` sur votre branche locale ont été transmises au référentiel distant.

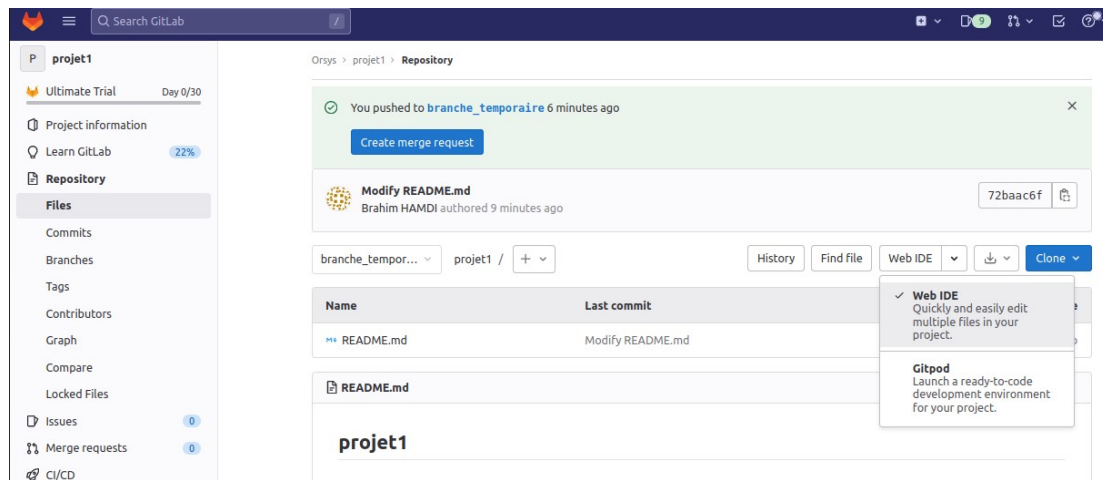


J. Modifier une branche distante

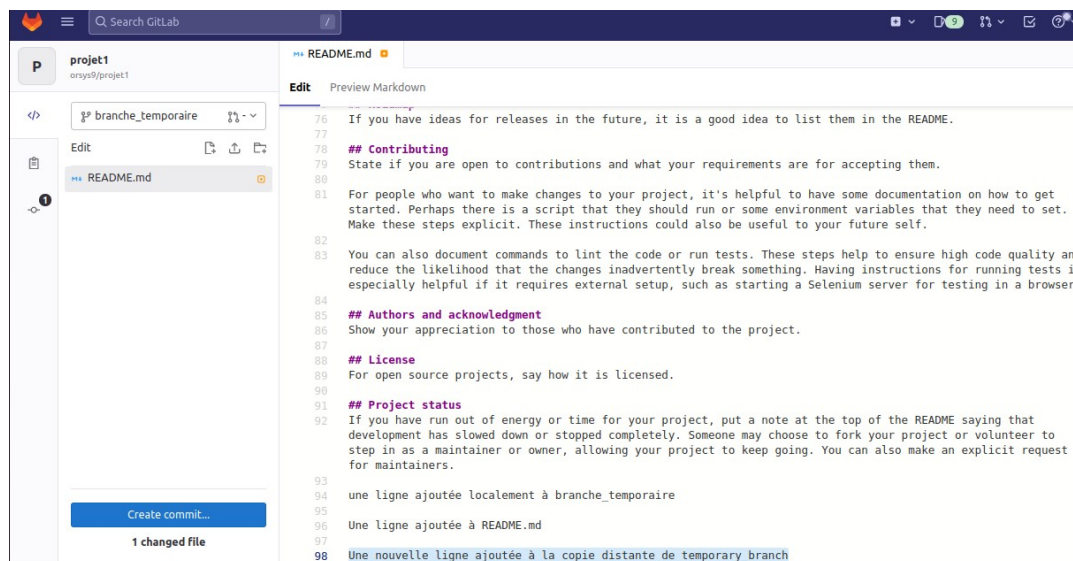
Supposant qu'un autre utilisateur a apporté une modification à la **branche temporaire** qui se trouve dans le référentiel distant de GitLab. Après cette modification, les versions distante et locale de cette branche seront différentes. Dans la section suivante, nous verrons comment concilier cette différence.

1. Sur le serveur GitLab, accédez à l'interface d'accueil de votre projet. Si vous n'êtes pas déjà sur **branche_temporaire**, accédez au volet de navigation de gauche et sélectionnez **Code > Branches > branche_temporaire**.

- Vous regardez maintenant les fichiers de la branche **branche_temporaire** . Cliquez sur **README.md** et passez en mode édition.



Ajoutez une nouvelle ligne à la fin du fichier.



- Puis validez en cliquant sur le bouton **Commit** pour finaliser les modifications sur la branche **branche_temporaire** du référentiel distant . Maintenant, le référentiel distant a désormais un commit *en avance* sur votre référentiel local.

Normalement, chaque branche dans laquelle vous vous engagez a besoin d'une demande de fusion associée y (**merge request**) pour pouvoir la fusionner avec la branche principale (main dans notre cas).

K. Obtenir des métadonnées sur les modifications apportées à la branche temporaire distante

Votre **branche_temporaire** locale n'est pas encore synchronisée avec la **branche_temporaire** distante sur GitLab. La commande `git fetch` obtient l'état mis à jour des branches distantes sans mettre à jour le contenu de vos branches locales. En d'autres termes, il vous indique combien de commits vos branches locales se trouvent derrière les branches distantes, mais il n'apporte aucune modification aux fichiers de vos branches locales.

1. Sur le terminal de votre machine, récupérez les mises à jour de la branche distante sans avoir les fusionner avec la branche locale.

```
brahim@Training:~/formation/projet1$ git log
commit 72baac6fd7b2c81a4738b9be2e3d4596d07b78ac (HEAD -> branche_temporaire)
Author: Brahim Hamdi <brahim.hamdi.consult@gmail.com>
Date: Sat Dec 3 09:19:18 2022 +0100

    Modify README.md

commit abe74a302033014708b3af5821b91a920f4c25d9
Author: Brahim Hamdi <brahim.hamdi.consult@gmail.com>
Date: Sat Dec 3 09:04:31 2022 +0100

    Add a line to README.md

commit 87bebe30a3afe95cd5cec4dd20a530c6901b7c5c (origin/main, origin/HEAD, ma
Author: leadingit consult <leadingit.consult@gmail.com>
Date: Sat Dec 3 06:25:20 2022 +0000

    Initial commit
brahim@Training:~/formation/projet1$

brahim@Training:~/formation/projet1$ git fetch
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
Dépaquetage des objets: 100% (3/3), 332 octets | 332.00 Kio/s, fait.
Depuis gitlab.com:orsys9/projet1
 72baac6..220790e branche_temporaire -> origin/branche_temporaire
brahim@Training:~/formation/projet1$
```

Laquelle des 2 versions (locale et distante) est en avance par rapport à l'autre ?

```
brahim@Training:~/formation/projet1$ git status
Sur la branche branche_temporaire
Votre branche est en retard sur 'origin/branche_temporaire' de 1 commit, et peut être mise à jour en avance rapide.
(utilisez "git pull" pour mettre à jour votre branche locale)

rien à valider, la copie de travail est propre
brahim@Training:~/formation/projet1$
```

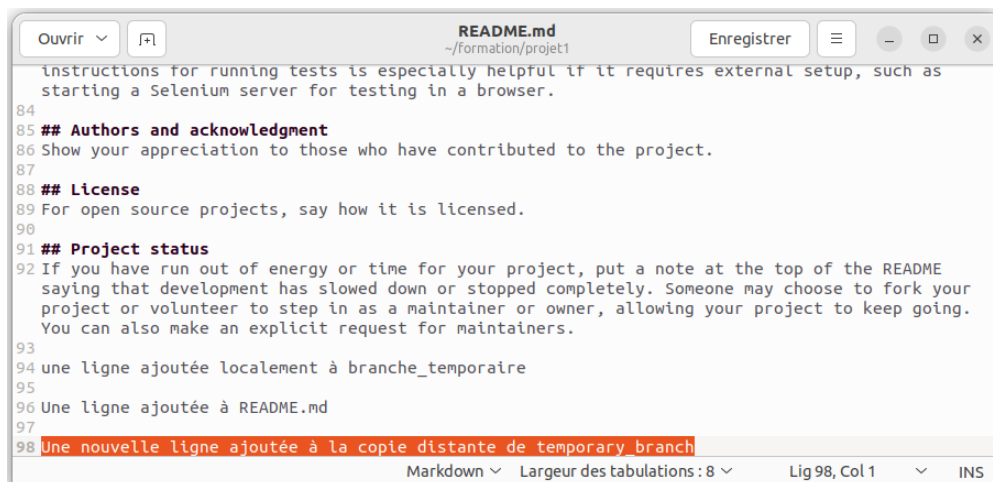
2. Dans votre terminal, fusionnez la copie distante dans votre copie locale en utilisant la commande `git pull`.

```
brahim@Training:~/formation/projet1$ git pull
Mise à jour 72baac6..220790e
Fast-forward
 README.md | 2 ++
 1 file changed, 2 insertions(+)
brahim@Training:~/formation/projet1$
```

Vérifiez la sortie pour voir combien de fichiers Git a mis à jour localement.

Quelle est la différence entre `git push` et `git fetch` ?

- Affichez le contenu mis à jour du fichier. Vous devriez voir la dernière ligne que vous avez ajoutée dans l'IDE Web GitLab.



M. Fusionner les changements dans la branche principale

Maintenant que votre **branche_temporaire** locale est identique à **branche_temporaire** distante, vous pouvez la fusionner dans votre branche principale locale. Cela ajoutera vos modifications à la base de code stable qui réside dans **main**.

1. Voyez sur quelle branche vous travaillez actuellement et basculez vers la branche principale **main**.

```
brahim@Training:~/formation/projet1$ git branch
* branche_temporaire
  main
brahim@Training:~/formation/projet1$ git checkout main
Basculement sur la branche 'main'
Votre branche est à jour avec 'origin/main'.
brahim@Training:~/formation/projet1$ git branch
* main
  branche_temporaire
brahim@Training:~/formation/projet1$
```

- Incorporez toutes les modifications de votre branche **branche_temporaire** locale (dans ce cas, uniquement la modification README.md) dans votre branche principale locale.

```
brahim@Training:~/formation/projet1$ git merge branche_temporaire
Mise à jour 87bebe3..220790e
Fast-forward
 README.md | 6 +++++
 1 file changed, 6 insertions(+)
brahim@Training:~/formation/projet1$
```

- Vérifiez avec `git log` l'ajout d'un nouveau commit à la branche *main*.

```
brahim@Training:~/formation/projet1$ git log
commit 220790e45cda48612add21092e2fe8433bcf5cbb (HEAD -> main, origin/branche_temporaire, branche_temporaire)
Author: leadingit consult <leadingit.consult@gmail.com>
Date: Sat Dec 3 08:39:57 2022 +0000

    Update README.md

commit 72baac6fd7b2c81a4738b9be2e3d4596d07b78ac
Author: Brahim Hamdi <brahim.hamdi.consult@gmail.com>
Date: Sat Dec 3 09:19:18 2022 +0100

    Modify README.md

commit abe74a302033014708b3af5821b91a920f4c25d9
Author: Brahim Hamdi <brahim.hamdi.consult@gmail.com>
Date: Sat Dec 3 09:04:31 2022 +0100

    Add a line to README.md

commit 87bebe30a3afe95cd5cec4dd20a530c6901b7c5c (origin/main, origin/HEAD)
Author: leadingit consult <leadingit.consult@gmail.com>
Date: Sat Dec 3 06:25:20 2022 +0000

    Initial commit
brahim@Training:~/formation/projet1$
```

N. Mettre à jour le référentiel distant

1. En utilisant la commande `git log`, assurez-vous qu'il n'y a pas de fichiers modifiés que vous devez préparer ou valider et confirmer sur la branche principale .
- En utilisant `git push`, Mettez à jour la copie distante de la branche principale avec toutes les modifications de votre copie locale.
 - Revenez à GitLab dans votre navigateur et affichez les modifications que vous venez de pousser vers la copie distante de *main*.