

TP 2 – 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 sur GitLab.

Paramétrage de la première utilisation de Git

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>

2. Avant de faire la configuration Git, vérifiez s'il est déjà configuré.

```
brahim@Training:~$ git config --list --show-origin
brahim@Training:~$
```

3. Si votre Git n'est pas encore configuré, ajoutez les paramètres de configuration suivants au niveau utilisateur :

- User.name : votre prénom et nom
- user.email : votre adresse mail
- core.editor : votre éditeur de texte préféré (nano, vi, notepad++, ...)

```
brahim@Training:~$ git config --global user.name "Brahim HAMDI"
brahim@Training:~$ git config --global user.email "brahim.hamdi.consult@gmail.com"
brahim@Training:~$ git config --global core.editor nano
brahim@Training:~$
```

- Vérifiez que Git est bien configuré au niveau utilisateur. Dans quel fichier a-t-il ajouté les paramètres ?

```
brahim@Training:~$ git config --list --global --show-origin
file:/home/brahim/.gitconfig      user.name=Brahim HAMDI
file:/home/brahim/.gitconfig      user.email=brahim.hamdi.consult@gmail.com
file:/home/brahim/.gitconfig      core.editor=nano
brahim@Training:~$
brahim@Training:~$ git config --list --system --show-origin
brahim@Training:~$ git config --list --local --show-origin
fatal: --local ne peut être utilisé qu'à l'intérieur d'un dépôt git
brahim@Training:~$
```

Configurer l'accès SSH à Gitlab

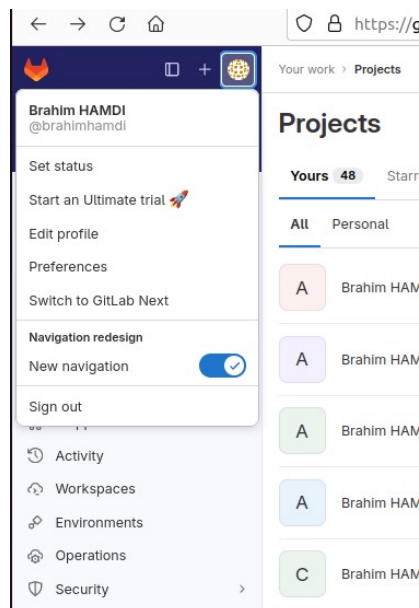
De nombreux serveurs Git utilisent une authentification par clés publiques SSH. Pour fournir une clé publique, chaque utilisateur de votre système doit la générer s'il n'en a pas déjà. Le processus est similaire sur tous les systèmes d'exploitation. Premièrement, l'utilisateur doit vérifier qu'il n'en a pas déjà une. Par défaut, les clés SSH d'un utilisateur sont stockées dans le répertoire `~/.ssh` du compte.

Dans cette partie, vous allez configurer SSH sur votre machine locale pour pouvoir établir une connexion sécurisée à <https://gitlab.com> pour télécharger et pousser les données de votre projet Git. Ainsi, pour se connecter à GitLab depuis Git, l'authentification s'effectuera par un échange de clés plutôt qu'un couple "identifiant / mot de passe".

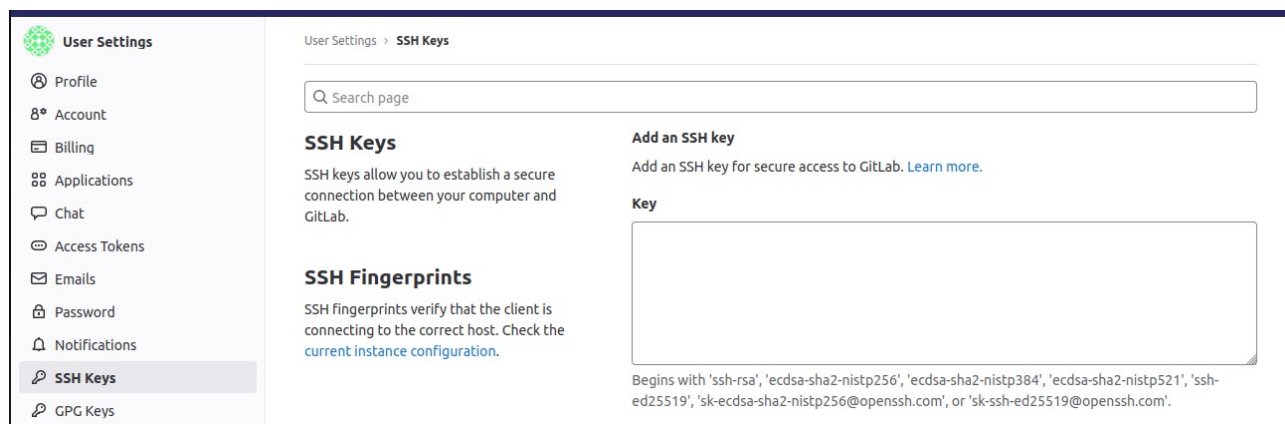
4. Créez une paire de clés (publique et privée) en exécutant cette commande `ssh-keygen` dans votre terminal.
- Lorsque vous y êtes invité, appuyez sur **Enter** pour accepter l'emplacement de la clé par défaut.
- Lorsque vous y êtes invité, appuyez sur **Enter** pour utiliser une phrase secrète (passphrase) 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+|
|..+.oooo . .+.|
|.o..+ooS      o|
|oo+ +oo.      Eo|
| +.=.o        .+|
|..o +         + |
|. .           .|
+---[SHA256]---+
brahim@Training:~$
```

5. 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**.



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

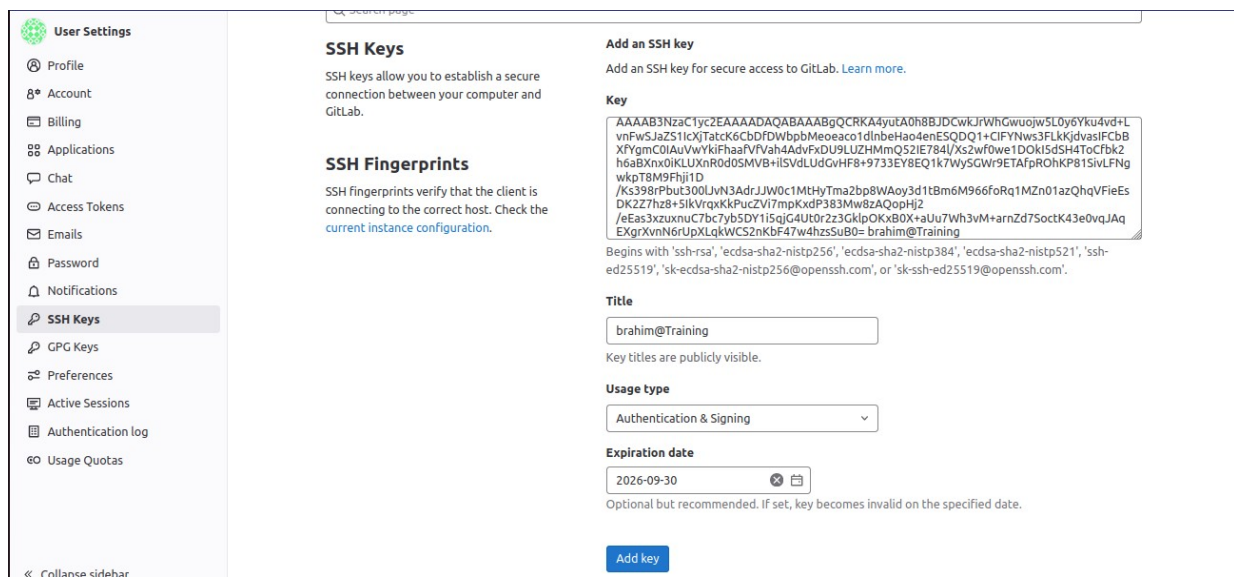
```
brahim@Training:~$ ls .ssh
id_rsa  id_rsa.pub  known_hosts  known_hosts.old
brahim@Training:~$
```

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.

- Copiez le contenu de votre clé publique (fichier `id_rsa.pub`) dans votre presse-papiers.

```
brahim@Training:~$ cat .ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQgQCCKA4yutA0h8BJDCwkJrWhGwuojw5L0y6Yku4vd+LvnFwSJaZ51IcXjTatcK6CbDFDwbpbMeoeaco1dlnbeHao4enESQDQ1+CIFYNws3
FLKkjdvafICbBXfYgmC0IAuVwYkLFhaafVfVah4AdvFXDU9LUZHMmQ52IE784L/Xs2wf0we1DOKI5dSH4ToCfbk2h6aBxnx0iKLUXnR0d0SMVB+lLSVdLUdGvHF8+9733EY8EQ1k7WysG
Wf9ETAFpR0hKP81SivLFNgwkpT8M9Fhj1D/Ks398rPbut300LJvN3AdrJJW0c1MthYTma2bp8WAoy3d1tBm6M966f0Rq1Mzn01azQhQVFIEsDK2Z7hz8+5IkVrqqKkPucZVi7mpKxdP3
83Mw8zAQopHj2/eEas3xzuxnuC7bc7yb5DY1i5qjG4Ut0r2z3GklpOKx80X+aUu7Wh3vM+arnZd7SocTK43e0vqJAQEXgrXvnN6rUpXLqkWCs2nKbF47w4hZsSuB0= brahim@Training
brahim@Training:~$
```

7. 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**.



8. Dans votre 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.

Cloner un dépôt de projet GitLab sur votre machine locale

9. Sur *gitlab.com*, créez un nouveau groupe privé **orsys9** avec le rôle **DevOps Engineer**.

Your work / Groups / New group / Create group

Create group

Groups allow you to manage and collaborate across multiple projects. Members of a group have access to all of its projects. Groups can also be nested by creating [subgroups](#).

Group name
orsys9
Must start with letter, digit, emoji, or underscore. Can also contain periods, dashes, spaces, and parentheses.

Group URL
https://gitlab.com/ orsys97573649

Visibility level
Who will be able to see this group? [View the documentation](#)

☒ Private
The group and its projects can only be viewed by members.

☐ Public
The group and any public projects can be viewed without any authentication.

Now, personalize your GitLab experience
We'll use this to help surface the right features and information to you.

Role
Devops Engineer

- Créez un nouveau projet blanc **projet1** sous **orsys9**.

orsys9 / New project / Create blank project

Create blank project

Create a blank project to store your files, plan your work, and collaborate on code, among other things.

Project name
My awesome project
Must start with a lowercase or uppercase letter, digit, emoji, or underscore. Can also contain dots, pluses, dashes, or spaces.

Project URL
https://gitlab.com/ orsys97573649

Project slug
my-awesome-project

Want to organize several dependent projects under the same namespace? [Create a group](#).

Project deployment target (optional)
Select the deployment target

Visibility Level
☒ Private
Project access must be granted explicitly to each user. If this project is part of a group, access is granted to members of the group.

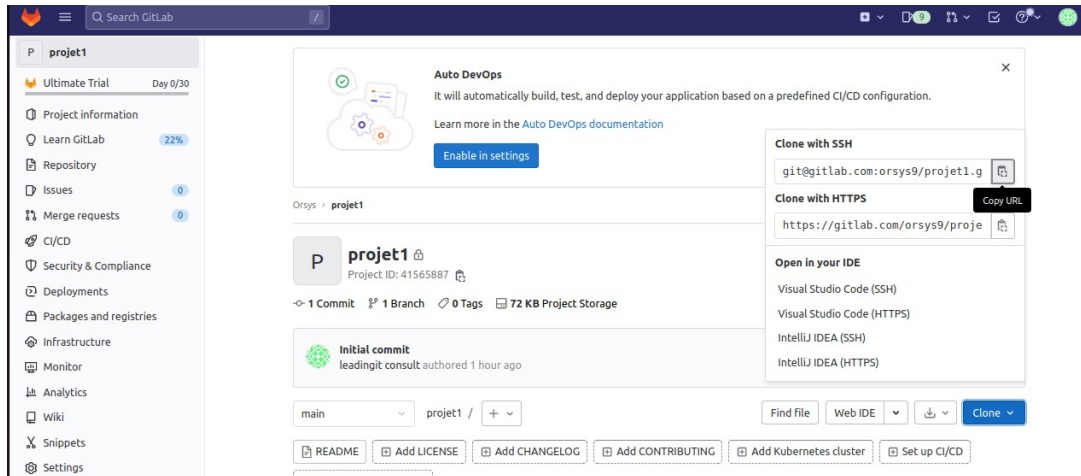
Project Configuration

☒ Initialize repository with a README
Allows you to immediately clone this project's repository. Skip this if you plan to push up an existing repository.

☐ Enable Static Application Security Testing (SAST)
Analyze your source code for known security vulnerabilities. [Learn more](#).

Create project Cancel

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



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

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

11. 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$
```

12. 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
└── HFAN
```

13. 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.

Travailler sur une branche

Jusqu'à maintenant tout le projet est sur une seule branche (main), mais dans la pratique on utilise plusieurs branches par projet. Dans cette partie, nous verrons comment Git gère les branches.

14. 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.

15. 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 *).

16. 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.

17. 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.

Modifier, indexer et valider (commit) un fichier

18. En utilisant votre éditeur de texte préféré, 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.

19. Ajoutez le fichier à la zone de préparation (*index* ou *staging area*). Assurez-vous que `README.md` est maintenant à l'état indexé (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.

20. 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.

Poussez vos modifications vers l'instance GitLab

21. 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.

22. En utilisant l'éditeur de texte de votre ordinateur local, ajoutez cette nouvelle ligne à la fin de votre copie locale de `README.md` et enregistrez le fichier.

Une ligne ajoutée à README.md

- Indexez puis validez la nouvelle modification du fichier `README.md`.

```
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$
```

23. Affichez l'historique des commits (validations) de votre dépôt Git local. Sur quel commit pointe le HEAD ?

```
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

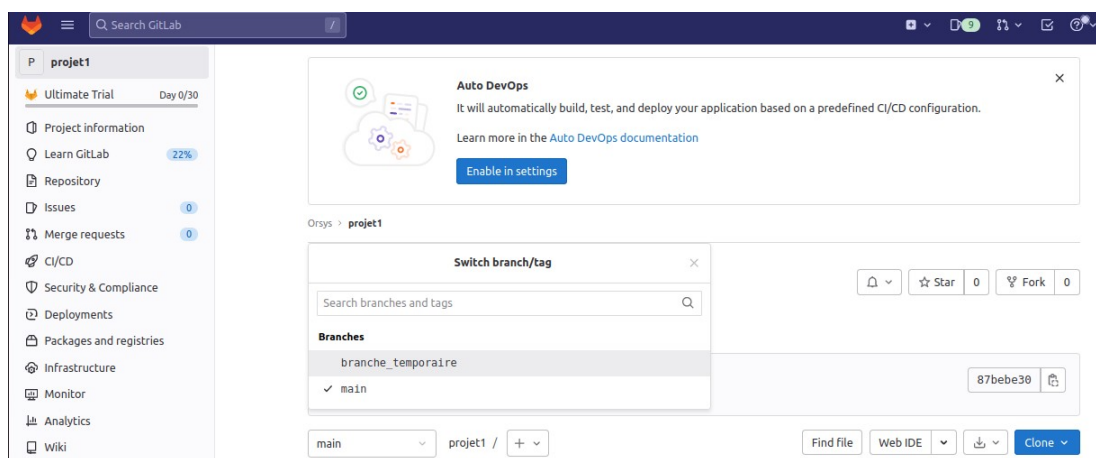
    Initial commit
brahim@Training:~/formation/projet1$
```

24. Poussez de nouveau vos mises à jour vers le référentiel distant **projet1** sur le serveur GitLab.

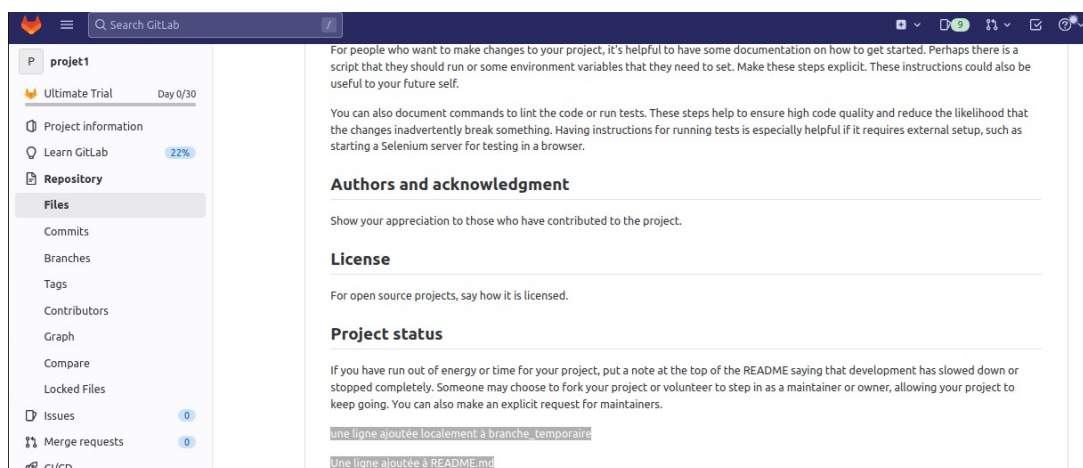
```
brahim@Training:~/formation/projet1$ git push
Énumération des objets: 5, fait.
Décompte des objets: 100% (5/5), fait.
Compression par delta en utilisant jusqu'à 8 fils d'exécution
Compression des objets: 100% (2/2), fait.
Écriture des objets: 100% (3/3), 320 octets | 320.00 Kio/s, fait.
Total 3 (delta 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.

25. Sur l'interface web de votre projet Gitlab, 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.

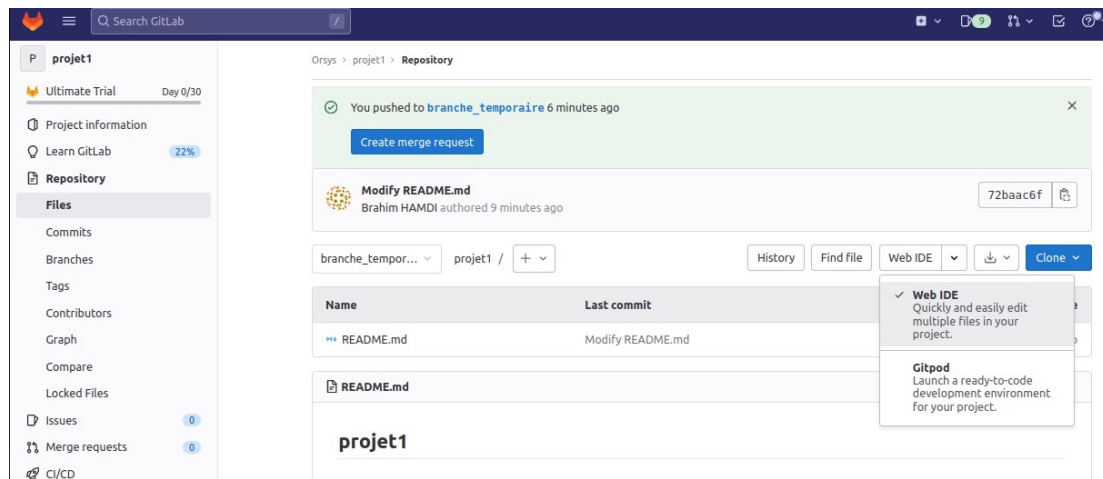


Modifier une branche distante

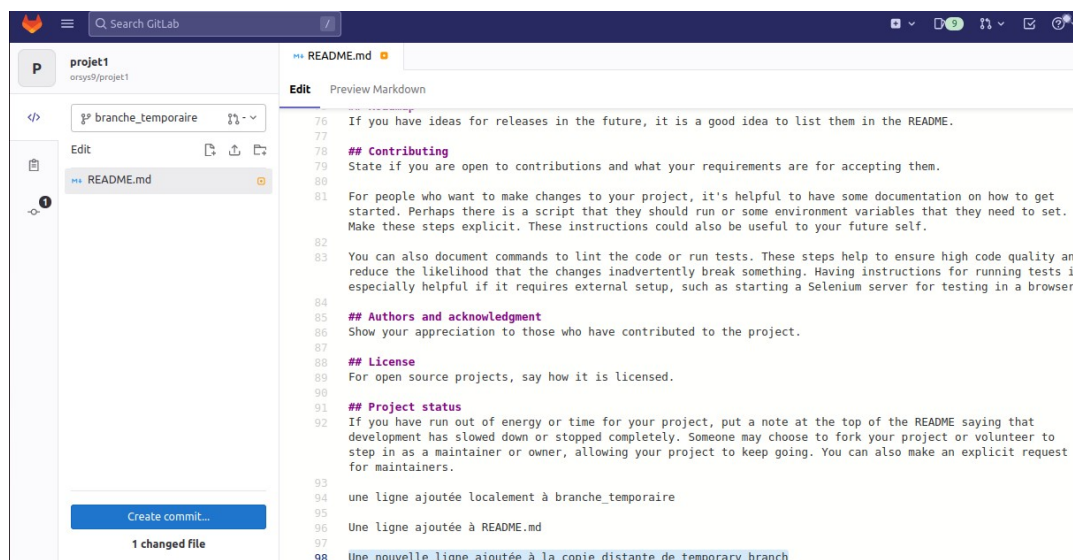
Supposant qu'un autre utilisateur a apporté une modification à la **branche temporaire** qui se trouve dans le référentiel distant de GitLab. Après cette modification, les versions distante et locale de cette branche seront différentes. Dans la section suivante, nous verrons comment concilier cette différence.

26. Sur GitLab.com, 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.



27. 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.

Dans les bonnes pratiques de Gitlab, 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).

Récupérer les modifications depuis l'instance Gitlab

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.

La commande `git pull` a le même comportement que `git fetch`, sauf qu'elle fusionne les mises à jour avec une branche locale : `git pull = git fetch + git merge`

28. Toujours sur votre dépôt local, 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)

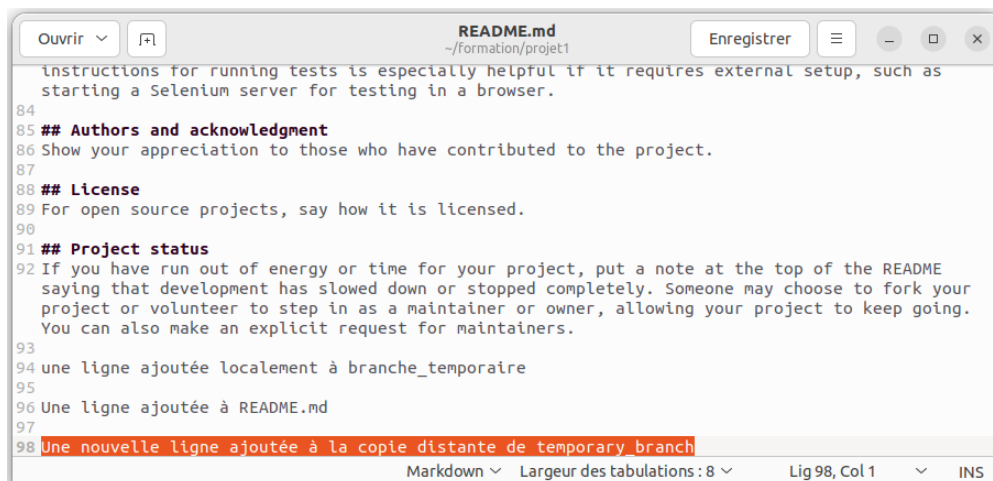
rien à valider, la copie de travail est propre
brahim@Training:~/formation/projet1$
```

29. Refaire la même chose mais en fusionnant les mises à jour avec la branche locale.

```
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.

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



30. 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**.

- 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
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$
```