Guide Git

Le guide rapide présente les commandes Git les plus courantes.

Pour chaque commande, consultez les sections correspondants pour plus d'information si une erreur se produit.

Configuration

1. Vérifier l'installation de Git

```
git version
```

Git affiche la version installée, par exemple `git version 2.43.0`.

2. Tester l'accès SSH

```
ssh -T git@git.esiee.fr
```

Le message de bienvenue indique votre nom d'utilisateur GitLab.

3. Créer et cloner un dépôt Git

Créez d'abord un dépôt Git **ici**, de nom `e3-algo-2026` et visibilité Interne.

```
git clone git@git.esiee.fr:<username>/e3-algo-2026.git
```

Git crée une copie local du dépôt dans le dossier `e3-algo-2026`.

4. Vérifier l'identité des commits

```
git config -l | grep user
```

Deux lignes indiquent vos coordonnées, `user.name=...` et `user.email=...`.

Utilisation

5. Voir les changements du projet

```
git status
```

Affiche les changements apportés au projet.

6. Indexer les changements et créer un commit

```
git add .
git commit -m "Corrige le tri par insertion"
```

Prépare les changements du dossier courant, puis les enregistre dans un commit local.

7. Recevoir les commits du serveur

```
git pull -r
```

Récupère les commits du serveur et les intègre aux commits locaux.

8. Envoyer ses commits sur GitLab

```
git push
```

Envoie les commits locales sur GitLab ESIEE.

Configuration · 1 Retour à la fiche réflexe

Vérifier l'installation de Git

Dans un terminal, entrez la commande

```
git version
```

Si Git est installé et accessible depuis le terminal, il affiche son numéro de version, par exemple `git version 2.43.0`.

Si le terminal ne reconnaît pas la commande, installez Git pour votre système.

Windows. Téléchargez Git pour Windows, lancez le programme d'installation, puis ouvrez **Git Bash**.

macOS. Dans Terminal, tapez `xcode-select -install`. Une fenêtre s'ouvre pour installer les outils de développement qui contiennent Git. Cliquez sur **Installer**.

Linux. Sous Debian ou Ubuntu, tapez

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

Rouvrez le terminal et relancez `git version`.

Configuration · 2 Retour à la fiche réflexe

Tester l'accès SSH

Avant de cloner le projet, testez la connexion au serveur GitLab de l'ESIEE.

```
ssh -T git@git.esiee.fr
```

Le message de bienvenue doit indiquer votre identifiant GitLab.

The authenticity of host can't be established. Cette erreur n'apparait qu'à la première connexion. Tapez `yes` et appuyez sur Entrée pour continuer.

Erreur de connexion. Si SSH affiche `Could not resolve hostname`, vérifiez que vous avez tapé `git.esiee.fr`. Si le nom est correct, ou si vous obtenez `Connection timed out` ou `Network is unreachable`, vérifiez votre connexion réseau. Sur le campus, connectez-vous au Wi-Fi **eduroam** (utilisateur `utilisateur@esiee.fr`), puis relancez le test SSH.

Permission denied (publickey). GitLab ne reconnaît aucune clé présentée par cette machine. Si vous y avez déjà une clé SSH, passez directement à l'affichage de sa clé publique. Sinon, créez une clé :

```
ssh-keygen -t ed25519
```

Appuyez sur Entrée pour accepter le chemin proposé, puis saisissez et confirmez une phrase de passe. Si une clé existe déjà à cet emplacement, répondez `n` pour la conserver.

Affichez la clé publique. Pour une clé Ed25519 au chemin par défaut, utilisez :

```
cat ~/.ssh/id_ed25519.pub
```

ou ouvrez le fichier `.ssh/id_ed25519.pub` dans votre dossier personnel.

Copiez la ligne entière. Sur GitLab ESIEE, ouvrez les paramètres (**Preferences**) de votre profil, puis **SSH keys** sous **Access**.

Ajoutez une clé, collez la ligne, donnez-lui le nom de la machine et enregistrez-la. La clé privée reste sur votre ordinateur. Relancez `ssh -T git@git.esiee.fr`.

Voire instructions SSH de GitLab pour plus de détails.

Configuration · 3 Retour à la fiche réflexe

Créer et cloner un dépôt Git

Sur GitLab ESIEE :

1. Cliquez sur le signe **+** en haut à droite.
2. Sélectionnez **New project/repository** (Nouveau projet / dépôt).
3. Cliquez sur **Create blank project** (Créer un projet vide).
4. Renseignez les paramètres suivants :
   - **Project name** (Nom du projet) : `e3-algo-2026`
   - **Project URL** (URL du projet) : cliquez sur le champ vide et sélectionnez votre nom d'utilisateur.
   - **Visibility** (Visibilité) : sélectionnez **Internal**.
5. Cliquez sur **Create project** (Créer le projet).

Un message de confirmation devrait apparaître, suivi de la page de votre dépôt.

Sur la page du projet, ouvrez le menu **Code** et copiez l'URL de clonage **SSH**. Dans le terminal, placez-vous dans le dossier qui doit accueillir le projet. Remplacez l'adresse ci-dessous par celle que vous venez de copier.

```
git clone git@git.esiee.fr:<username>/e3-algo-2026.git
```

Git affiche `Cloning into 'e3-algo-2026'...`, puis indique la progression du clonage.

Le clonage crée le dossier `e3-algo-2026`, y copie les fichiers et l'historique, puis mémorise l'adresse du serveur sous le nom `origin`. Utilisez ensuite cette copie pour les séances suivantes.

Si le dossier de destination existe déjà, entrez-y et vérifiez `git remote -v`. Si l'adresse correspond au projet, la copie est déjà prête. Si le clonage échoue avec `Permission denied (publickey)`, suivez la section sur SSH, puis relancez le clonage.

Configuration · 4 Retour à la fiche réflexe

Vérifier l'identité des commits

Depuis le dossier du projet, affichez les réglages et repérez les lignes `user.name` et `user.email`. Chaque commit conserve le nom et l'adresse e-mail de son auteur.

```
git config -l | grep user
```

Pour Camille Martin, ces lignes affichent par exemple

```
user.name=Camille Martin
user.email=camille.martin@edu.esiee.fr
```

Si une valeur manque ou ne correspond pas à vos coordonnées, configurez votre identité dans cette copie du dépôt. Remplacez les valeurs de l'exemple par votre nom et votre adresse e-mail :

```
git config user.name "Camille Martin"
git config user.email "camille@example.org"
```

Git enregistre ces réglages sans afficher de message. Relancez la commande de vérification.

Ces valeurs figurent dans les prochains commits.

Utilisation · 5 Retour à la fiche réflexe

Voir les changements du projet

Après avoir enregistré un fichier dans votre éditeur, vous pouvez examiner l'état du projet.

```
git status
```

Git compare trois états du projet. La **copie de travail** contient les fichiers que vous éditez. L'**index** contient la version préparée pour le prochain commit. Le **dernier commit** est la dernière version enregistrée dans l'historique de votre branche.

Changes to be committed : L'index diffère du dernier commit. Ces changements seront inclus si vous tapez maintenant `git commit -m "Mon message"`. Pour les lire ligne par ligne, utilisez `git diff -staged`.

Changes not staged for commit : Un fichier suivi a changé depuis sa dernière indexation. `git diff` montre ces différences entre le dossier de travail et l'index.

Untracked files : Les fichiers non suivis sont ceux que Git voit pour la première fois et que vous n'avez pas encore ajoutés.

Pourquoi le même fichier apparaît-il deux fois ? Vous avez indexé un fichier `exercice.c`, puis y ajouté une nouvelle fonction. La première version (sans la nouvelle fonction) est prête pour le commit. La nouvelle fonction reste dans la copie de travail jusqu'au prochain `git add`.

Utilisation · 6 Retour à la fiche réflexe

Préparer et enregistrer un commit

À la racine du projet, repérez les changements à enregistrer ensemble.

```
git status
```

Si vous avez modifié le fichier `example.c`, la ligne `modified: example.c` apparaît sous `Changes not staged for commit:`.

Pour indexer les modifications, tapez

```
git add example.c fichier.c
```

pour ajouter au index les fichiers nommés dans la commande, ou tapez

```
git add .
```

pout ajouter tous les fichiers dans le dossier courant et ses sous-dossiers.

Git indexe les changements sans afficher de message. Vous pouvez les vérifier à l'aide de `git status`, ou dans plus de détail avec la commande :

```
git diff -staged
```

Pour chaque fichier, Git affiche `diff -git a/fichier b/fichier`, puis les différences indexées.

Enregistrez les changements indexés avec un message qui décrit la modification.

```
git commit -m "Ajoute les exercices TP01"
```

Git confirme le commit en affichant par exemple `[main 7cd8901] Ajoute les exercices TP01`. Ce commit se trouve sur votre ordinateur. La publication se fait avec `git push`.

Si le commit semble incomplet , vérifiez si vous avez retouché un fichier après `git add`. Git enregistre la version présente dans l'index au moment du commit. Indexez les retouches pour les inclure dans le prochain commit.

Utilisation · 7 Retour à la fiche réflexe

Recevoir les commits du serveur

Utilisation · 8 Retour à la fiche réflexe

Envoyer ses commits sur GitLab

Après un commit, une copie de travail « propre » indique que tous les changements suivis sont enregistrés localement.

Pour envoyer les commits realisés sur le GitLab ESIEE, tapez

```
git push
```

Git indique la destination et la mise à jour du depôt, par exemple

```
To git@git.esiee.fr:<username>/e3-algo-2026.git
 a4a51d1..dd447cb main -> main
```

Pour vérifier l'envoi, ouvrez le projet sur GitLab ESIEE, et retrouvez votre message dans l'historique des commits (**<N> Commits** à droite).

fatal : The current branch main has no upstream branch. Le suivi du dêpot distant n'est pas configuré. Pour le faire, tapez

```
git push -u origin main
```

Push refusé avec `non-fast-forward`. Si le `git push` est refusé avec une erreur

```
error: failed to push some refs to ...
To prevent you from losing history, non-fast-forward updates were rejected
```

la branche distante contient des commits absents de votre branche locale.

Exécutez `git pull -r`, puis relancez `git push`.