1 À propos

cf. https://fr.wikipedia.org/wiki/Git.

> Git est un logiciel de gestion de versions **décentralisé**. C'est un logiciel libre et gratuit (GNU GPL2). Depuis les années 2010, il s'agit du logiciel de gestion de versions le plus populaire dans le développement logiciel et web.

La *forge* git que nous utilisons pour le projet (et pour d'autres cours & projets de votre cursus) est la forge proposée par la communauté d'enseignement supérieur et de recherche, hébergée à l'adresse :

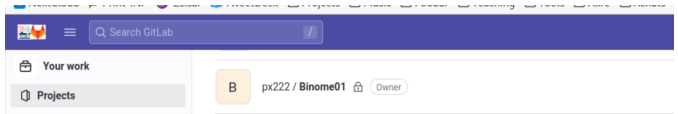
https://gricad-gitlab.univ-grenoble-alpes.fr/

à laquelle nous pouvons nous connecter via login/mot de passe INP (*agalan*).

> Git est un excellent outil de partage de code. Néanmoins, son utilisation doit respecter les règles des enseignements de l'école : la copie de code entre binômes reste interdite. De plus, il est de votre responsabilité de laisser votre dépôt privé.

2 Pour démarrer

Nous vous avons créé un dépôt git par binôme. Vous pouvez voir ce dépôt dans votre liste des projets sous gricad-gitlab :



Dans le menu bleu "clone", vous trouverez l'url pour récupérer une *copie locale du dépôt une première fois* :

```
git clone git@gricad-gitlab.univ-grenoble-alpes.fr:px222/binomeXX.git
OU
git clone https://gricad-gitlab.univ-grenoble-alpes.fr/px222/binomeXX.git
```

Remarques

- Une fois ce clone réalisé sur votre machine, tout est prêt pour travailler avec git à partir de cette machine (plus de clone nécessaire).
- Pour utiliser git avec ssh, vous devez déposer une clé publique dans les paramètres de votre compte (à partir de votre navigateur). Sinon, vous devrez utiliser votre login/mdp agalan.
- Nous vous montrons ici l'usage standard de git adapté à un petit projet, mais suffisant. Il est demandé de rester dans ce cadre (pas de branche, pas d'usage avancé).
- La suite logicielle `gitlab` fournit d'autres services autour de git, comme la notion *dissues*. Si vous êtes curieux·se, il n'est pas interdit de regarder et de s'en servir.

3 Versions des fichiers

Cf Figure 1.

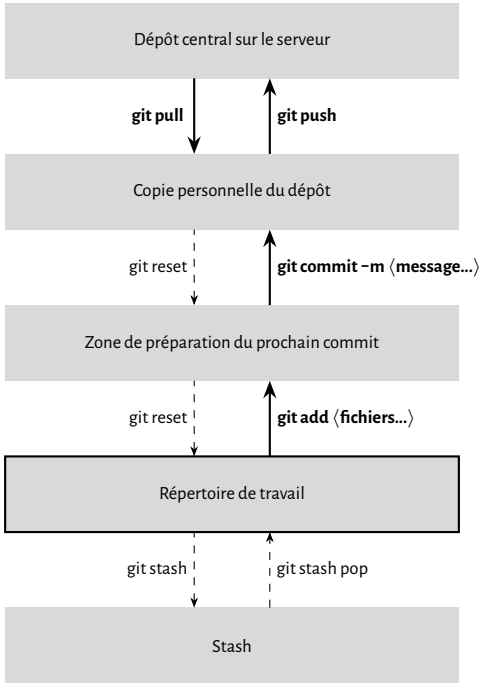


FIGURE 1 – Vues logiques des fichiers dans git

Répertoire de travail

Vos fichiers sont stockés dans un dossier comme d'habitude, sauf que Git surveille ce que vous modifiez. Périodiquement vous sauvegardez vos modifications dans le dépôt en les mettant dans un *commit*.

- **git status** donne l'état du dossier de travail : fichiers modifiés, fichiers ajoutés au prochain commit, et nouveau fichiers.
- **git add ⟨fichiers...⟩** ajoute les fichiers indiqués à la zone de préparation du prochain commit.
- **git diff** liste tous les changements faits sur les fichiers qui n'ont *pas* été ajoutés au prochain commit avec git add. **git diff --staged** liste les changements qui ont été ajoutés avec git add.

Copie personnelle du dépôt

Il s'agit d'une séquence de *commits* stockée en local sur votre ordinateur, qui doit régulièrement être synchronisée avec le dépôt central sur le serveur.

- **git commit -m "⟨message⟩"** crée un nouveau commit contenant les changements ajoutés par git add et le message indiqué (qui décrit ce qui a été changé dans le projet).
- **git log** montre la séquence de commits ainsi que les messages associés.

Le dépôt central sur le serveur

Le dépôt central est aussi une séquence de *commits* et c'est le code officiel.

- **git push** envoie l'état de votre dépôt local vers le serveur, une fois que des nouveaux commits ont été créés.
- **git pull** télécharge les commits qui ont été envoyés sur le serveur par des collègues et que vous n'avez pas encore.

4 Flot de travail classique avec git

- Début de ma séance de travail, je synchronise mes fichiers locaux avec ce qui a été fait dans le dépôt : **git pull** .
- Je crée un nouveau fichier non vide (ou je modifie un fichier existant), mettons `README.md`.
- J'ajoute ce fichier à ma copie locale : **git add README.md** .
- Je *commite* mes changements : **git commit -m "ajout d'un readme"**
- Je pousse mes changements sur le serveur : **git push** .

5 Gestion de conflits - exemple de scénario

A et B travaillent tous les deux sur le fichier `foo.c` :

- A commite et pousse son travail sur le serveur.
- B commite, elle ne peut pousser, elle lance la commande **git pull**
- Dans le cas favorable, une fusion (*merge* peut être automatiquement réalisée, et la commande précédente réussit, B peut pousser cette fusion.
- Dans le cas défavorable, la fusion échoue, et le fichier `foo.c` contient *à chaque endroit de conflit non résolu* des marqueurs comme ceci :

```
<<<<<<< HEAD
modifications apportées localement par B
=======
modifs de la branche distantes (probablement A)
>>>>>>>
```

- B choisit ce qu'elle garde, enlève les chevrons et la/les barre(s) de séparation.
- B propage ses modifications : **git add foo.c** , puis commit, puis push.

Remarque : certains IDE ou autres outils plus ou moins graphiques peuvent réaliser cette fusion.