Usage de GitLab

La création d'un depôt git sur le GitLab de l'UFR permettra à votre groupe de disposer d'un dépôt commun de fichiers sur ce serveur. Chaque membre du groupe pourra ensuite disposer d'une copie locale de ces fichiers sur sa machine, les faire évoluer, puis sauvegarder les changements jugés intéressants et les synchroniser sur le serveur. Si vous ne vous êtes pas déjà servi de GitLab dans d'autres matières, ce fichier décrit son usage le plus élémentaire. Pour plus d'informations sur git et GitLab, il existe de multiples tutoriels en ligne. Nous contacter en cas de problèmes.

Accès au serveur et configuration personnelle.

Se connecter via l'interface web https://gaufre.informatique.univ-paris-diderot.fr. Utilisez pour cela les mêmes nom et mot de passe que sur les machines de l'UFR, et pas vos compte "ENT" de paris diderot ou u-paris. Cliquer ensuite sur l'icône en haut à droite, puis sur "Settings". À droite, aller ensuite dans la section "SSH Keys", et ajouter ici la partie publique de votre clé ssh (ou de vos clés si vous en avez plusieurs). Cela facilitera grandement les accès ultérieurs à votre dépôt git, et vous évitera de taper votre mot de passe à chaque action.

Si vous n'avez pas encore de clé ssh, s'en générer une sur sa machine. L'usage de ssh n'est pas spécifique à git et GitLab, et permet des connections "shell" à des machines distantes. Si vous n'utilisez pas encore ssh et les clés publiques/privées ssh, il est temps de s'y mettre ! Plus d'information sur ssh à l'UFR.

Création du dépôt

Pour ce projet, nous vous fournissons quelques fichiers initiaux. Votre dépôt git sera donc un dérivé (ou "fork") du dépôt public du cours. En pratique:

- L'un des membres de votre groupe se rend sur le dépôt git du projet https://gaufre.informatique.univ-paris-diderot.fr/pf5-profs/pf5-projet-2022 puis s'identifie si ce n'est pas déjà fait, et appuie sur le bouton "fork" (vers le haut, entre "Star" et "Clone"). Attention, un seul "fork" par groupe suffit.
- Ensuite, aller dans la section "Settings" en bas à gauche, défiler un peu et cliquer sur "Visibility", et selectionner "Private" comme "projet visibility", puis "Save changes" un peu plus bas. Vérifier qu'un cadenas apparaît maintenant à droite de `pf5-projet-2022` quand vous cliquez sur "Projet" en haut à gauche.
- Toujours dans "Settings" en bas à gauche, mais sous-section "Members" maintenant. "Invitez" votre collègue de projet en choissant "Maintainer" comme rôle, ainsi également que l'utilisateur `letouzey`, également comme "Maintainer".
- Voilà, votre dépôt sur le GitLab est prêt !

Création et synchronisation de vos copies locales de travail

Chaque membre du projet "clone" le dépôt du projet sur sa propre machine, c'est-à-dire en télécharge une copie locale : `git clone ...` suivi de l'addresse du projet tel qu'il apparaît dans l'onglet "Clone" sur la

page du projet, champ en "SSH". Pour cela, il faut avoir installé `git` et `ssh` et configuré au moins une clé ssh dans GitLab.

Une fois le dépôt créé et cloné et en se plaçant dans le répertoire du dépôt, chaque membre pourra à tout moment :

- télécharger en local la version la plus récente du dépôt distant sur Gitlab via `git pull`
- téleverser sa copie locale modifiée sur GitLab : `git push`

Avant toute synchronisation, il est demandé d'avoir une copie locale "propre" (où toutes les modifications sont enregistrées dans des "commits").

Modifications du dépôt : les commits

Un dépôt Git est un répertoire dont on peut sauvegarder l'historique des modifications. Chaque action de sauvegarde est appelée une *révision* ou "commit".

L'*index* du dépôt est l'ensemble des modifications qui seront sauvegardées à la prochaine révision. La commande `git add` suivit du nom d'un ou plusieurs fichiers permet d'ajouter à l'index toutes les modifications faites sur ces fichiers. Si l'un d'eux vient d'être créé, on ajoute dans ce cas à l'index l'opération d'ajout de ce fichier au dépôt. La même commande suivie d'un nom de répertoire ajoute à l'index l'opération d'ajout du répertoire et de son contenu au dépôt.

La révision effective du dépôt se fait par la commande `git commit -m "message"` où le message entre guillements double permet de décrire le contenu du commit.

Invocable à tout instant, la commande `git status` permet d'afficher l'état courant du dépôt depuis sa dernière révision : quels fichiers ont été modifiés, renommés, effacés, créés, etc., et lesquelles de ces modifications sont dans l'index. Elle indique également comment rétablir l'état d'un fichier à celui de la dernière révision, ce qui est utile en cas de fausse manoeuvre.

Les commandes `git mv` et `git rm` se comportent comme `mv` et `rm`, mais ajoutent immédiatement les modifications associées du répertoire à l'index.

Il est conseillé d'installer et d'utiliser les interfaces graphiques `gitk` (visualisation de l'arbre des commits) et `git gui` (aide à la création de commits).

Une dernière chose : git est là pour vous aider à organiser et archiver vos divers fichiers sources. Par contre il vaut mieux ne **pas** y enregistrer les fichiers issues de compilations (binaires, répertoire temporaire tels que `_build` pour `dune`, fichiers objets OCaml `*.cm{o,x,a}`, etc.

Les fusions (merge) et les conflits

Si vous êtes plusieurs à modifier vos dépôts locaux chacun de votre côté, celui qui se synchronisera en second avec votre dépôt GitLab commun aura une manoeuvre nommé "merge" à effectuer. Tant que vos modifications respectives concernent des fichiers ou des zones de code différentes, ce "merge" est aisé, il suffit d'accepter ce que git propose, en personnalisant éventuellement le message de merge. Si par contre les modifications se chevauchent et sont incompatibles, il y a alors un conflit, et git vous demande d'aller décider quelle version est à garder. Divers outils peuvent aider lors de cette opération, mais au plus basique il s'agit d'aller éditer les zones entre `<<<<<` et `>>>>>` puis faire `git add` et `git commit` de nouveau.

Intégrer les modifications venant du dépôt du cours

Si le dépôt du cours reçoit ultérieurement des correctifs ou des évolution des fichiers fournis pour le projet, ces modifications peuvent être intégrées à vos dépôts.

La première fois, aller dans votre répertoire de travail sur votre machine, et taper:

```
git remote add prof git@gaufre.informatique.univ-paris-diderot.fr:pf5-profs/pf5-projet-2022.git
```

Ensuite, à chaque fois que vous souhaiterez récupérer des commits du dépôt enseignant : `git pull prof master` .

Selon les modifications récupérées et les votres entre-temps, cela peut occasionner une opération de "merge" comme décrite auparavant.

Enfin, ces modifications sont maintenant intégrées à votre copie locale de travail, il ne reste plus qu'à les transmettre également à votre dépôt sur GitLab via `git push` .

Les branches

Il est parfois pratique de pouvoir essayer différentes choses, même incompatibles. Pour cela, Git permet de travailler sur plusieurs exemplaires d'un même dépôt, des *branches*. Un dépôt contient toujours une branche principale, la branche `master`, dont le rôle est en principe de contenir sa dernière version stable. Les autres branches peuvent servir à développer des variantes de la branche master, par exemple pour tenter de corriger un bug sans altérer cette version de référence.

La création d'une nouvelle branche, copie conforme de la branche courante (qui est au départ `master`) dans son état courant, se fait par `git branch nom_de_la_nouvelle_branche` .

Sans arguments, `git branch` indique la liste des branches existantes, ainsi que celle dans laquelle se trouve l'utilisateur.

Le passage à une branche se fait par `git checkout nom_de_branche` .

Pour ajouter au dépôt distant une branche qui n'est pas encore sur celui-ci, après s'être placé dans la branche, faire `git push --set-upstream origin nom_de_branche` .

Un `git push` depuis une branche déjà sur le serveur se fait de la manière habituelle.

Enfin, on peut "réunifier" deux branches avec `git merge`, voir la documentation pour plus de détails.

Noter que GitLab propose également un mécanisme de "Merge Request" : il permet de proposer des modifications, soit à son propre projet, soit au projet qui a été "forké" à l'origine, les membres du projet en question pouvant alors accepter ou non ces suggestions après discussion.