

Cahier de TP « Git / Gitlab »

Pré-requis :

- Bonne connexion Internet
- Système d'exploitation recommandé : Linux, MacOS, Windows 10
- Option : Utilisation de Docker
- Bon éditeur : .yaml, .xml. Par exemple VSCode

Table des matières

Atelier 1 : Commandes de base Git.....	2
1.1 Installation et configuration de Git.....	2
1.2 Création de dépôt et premier commit.....	2
Atelier 2 : Historique et annulation.....	2
2.1 Visualisation de l'historique dans différents formats.....	2
2.2 Annulation.....	3
Atelier 3 : Branches locales.....	4
3.1 Fusion de branches.....	4
3.2 Rebase.....	4
3.3 Tags.....	4
Atelier 4 : Le serveur Gitlab.....	6
4.1 Groupes de projets, projets et membres.....	6
4.2 Gestion des issues.....	6
Atelier 5 : Workflow de collaboration.....	8
5.1 Dépôts distants.....	8
5.2 Branches distantes.....	8
5.3 Merge request.....	8
Atelier 6 : Quelques outils utiles.....	9
6.1 Picorage.....	9
6.2 Indexation interactive et partielle.....	9
6.3 Mise de côté.....	9
6.4 Modification de l'historique.....	9
Annexe.....	10

Atelier 1 : Commandes de base Git

1.1 Installation et configuration de Git

Après l'installation de Git, le configurer afin de pouvoir committer
`$ git config --global user.name "John Doe"`

`$ git config --global user.email johndoe@example.com`

Vérifier votre configuration et en particulier l'éditeur associé à *git*

1.2 Création de dépôt et premier commit

Récupérer les sources fournies, initialiser un dépôt, vérifier le fichier *.gitignore* et ajouter toutes les sources du projet

1.3 Mise en place des fichiers à ignorer

Mettez en place un fichier *.gitignore* qui spécifie les règles suivantes :

- Ignorer les fichiers se terminant par *~, .bak, .log*
- Ignorer les fichiers *.class* et *.html* dans le répertoire *java*
- Ignorer tous les fichiers d'un répertoire *tmp*
- Ignorer les fichiers se terminant par *.txt* sauf *README.txt*

1.4 Ajout et modifications de fichiers

- Ajouter un fichier de votre choix dans l'arborescence et modifier le fichier *contenu.html*.
- Avant d'indexer les modifications, exécuter la commande *git status* et *git diff*
- Indexer les fichiers et effectuer la commande *git diff --cached*
- Valider vos modifications avec *git commit -v* et éditer le message de commit

1.5 Suppression et déplacement

- Renommer le fichier *README* en *README.txt* et supprimer un des fichiers du dépôt.
- Valider vos modifications

Atelier 2 : Historique et annulation

2.1 Visualisation de l'historique dans différents formats

Utiliser les différentes options de la commande *log* pour :

- Afficher les statistiques pour les fichiers modifiés
- Afficher les différences au niveau mot
- Date relative
- Une seule ligne

Visualisation avec *gitk*

2.2 Annulation

- Modifier le message du dernier commit
- Ajouter un nouveau fichier et l'inclure dans le nouveau commit
- Modifier un fichier et l'ajouter dans l'index ; puis l'enlever de l'index et récupérer la version du dépôt

Atelier 3 : Branches locales

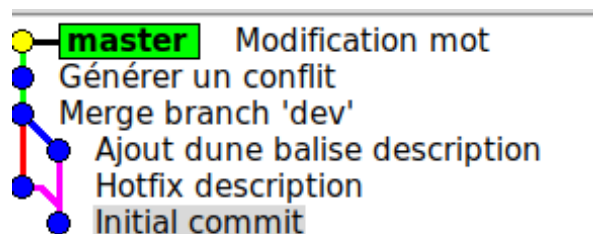
3.1 Fusion de branches

- Basculer vers une nouvelle branche **dev**
Éditer plusieurs fichiers et effectuer au moins 2 commits
- Revenir à la branche **master**
Créer une branche **hotfix** et éditer un des fichier modifiés dans dev (les mêmes lignes, pour générer un conflit)
Committer
- Effectuer un fusion fast-forward de **hotfix** vers **master**
Supprimer **hotfix**
- Intégrer les modifications de **dev** dans **master** avec la commande **merge**, résoudre les conflits
- Visualisez les changements avec *gitk*
- Supprimer la branche **dev**

3.2 Rebase

- Effectuer les mêmes opérations que la section précédente afin que master et de dev soient divergentes et que les patchs concernent les mêmes lignes de code
- Rebaser ensuite **dev** sur **master**
- Faire une fusion rapide de master, (Sur master : *git merge dev*)
- Visualisez les changements avec *gitk*
- Supprimer la branche **dev**

A la fin du TP, l'historique du projet doit ressembler à :



3.3 Tags

- Taggé à posteriori, le commit de merge
- Basculer vers le tag

Dans quel état est le workspace ?

Atelier 4 : Le serveur Gitlab

Il est recommandé pour des soucis de simplicité d'utiliser un compte existant sur un serveur gitlab ou de se créer un compte sur gitlab.com

Pour une installation en local avec docker, voir annexe.

4.1 Groupes de projets, projets et membres

4.1.1 ; Création de 3 comptes (Si installation autonome)

Avec un compte admin, créer

- Un utilisateur avec votre identité
- Un login developer
- Un login reporter

4.1.2 Mise en place clé ssh

Se connecter ensuite avec votre compte

Configurer une clé ssh en suivant les instructions de : <https://docs.gitlab.com/ee/user/ssh.html>

4.1.3 Création de groupe de projet

Créer un groupe de projet nommé **formation**

Interne : Visibilité privé

Répartir les rôles sur les groupes de projet avec les autres stagiaires ou les comptes précédemment créés.

4.1.4 Création de projet

Créer le projet **html-project** appartenant à ce groupe, le repository de ce projet ne sera pas initialisé pour l'instant

4.2 Gestion des issues

Avec le compte Owner,

- Créer plusieurs milestone sur le projet

Avec le compte reporter

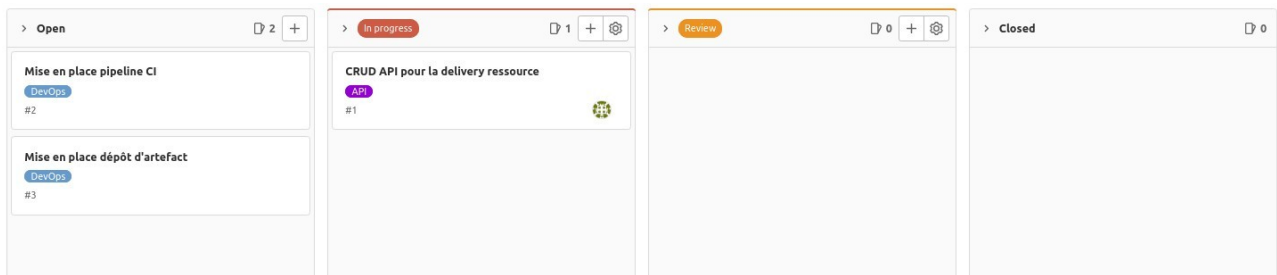
- Saisir plusieurs issues
- Discussion sur une issue entre Reporter/Mainteneur projet :
- Saisir quelques commentaires

Avec le compte owner :

- mise au planning et affectation des issues

- Création de 4 labels :
 - En cours
 - En recette
 - Front-end
 - Back-end

Utiliser les 2 premiers labels pour créer un tableau de bord



Affecter des labels à vos issues

Avec le compte **developer**, accès au tableau de bord et déplacement d'une issue de *Open* → *En cours*

Atelier 5 : Workflow de collaboration

Travailler à 2 stagiaires sur la même instance de *gitlab* avec des rôles différents (Mainteneur et Developer par exemple)

Au besoin, utiliser *gitlab.com*

5.1 Dépôts distants

A partir du dépôt local, déclarer le dépôt distant de gitlab, y pousser toutes les branches et tous les tags

Normalement, les commandes nécessaires sont visibles sur la page d'accueil du projet *Gitlab*

5.2 Branches distantes

- Créer une branche sur gitlab à partir de master en utilisant la souris
- Récupérer la branche sur votre dépôt local.
- Y faire des *push*

5.3 Merge request

Avec votre compte, vérifier la configuration des MRs, en particulier si toutes les discussions doivent être résolues.

Se logger avec le compte *developer*

- Créer une Merge Request à partir d'une issue
- Commencer à collaborer et ouvrir une discussion avant de pousser des modifications.
- Faire un commit directement par le site web
- Faire un push à partir de votre repository local
- Changer le statut de la MR

Se logger avec votre compte (*maintainer*)

- Faire une revue de code
- Répondre à la discussion ouverte et résoudre la discussion
- Effectuer le Merge et vérifier la suppression de la branche

Atelier 6 : Quelques outils utiles

6.1 Picorage

Essayer de picorer un ensemble de commits consécutifs d'une branche à une autre

6.2 Indexation interactive et partielle

- Ajouter 2 nouveaux fichiers et modifier plusieurs lignes sur un fichier particulier (A des endroits différents de préférence)
- Lancer une indexation interactive et ne prendre qu'un fichier ajouté et qu'une ligne modifiée dans une première indexation
- Committer
- Indexer le 2 ème ensemble de modifications et committer

6.3 Mise de côté

- Faire des modifications sur un fichier
- Indexer les modifications
- Mettre les modifications de côté avec la commande stash
- Faire d'autres modifications
- Les mettre également de côté
- Réappliquer le premier stash en conservant l'indexation associée (option *-index*)

6.4 Modification de l'historique

Retravailler les 3 derniers commits de l'historique

- Modifier le message du commit le plus ancien
- Fusionner les 2 commits suivants en un seul

6.4 Hooks

Renommer le hook ***pre-commit.sample*** en ***pre-commit***. Ce hook vérifie le nom des fichiers faisant partie du commit et interdit le commit si le nom du fichier contient des caractères non-ascii

Faire ensuite une modification de nom de fichier interdite.

Ecrire un hook post-commit vérifiant que le message de commit est supérieur à 10 caractères

Annexe

Installation Gitlab avec Docker

Via docker :

```
sudo docker run --detach \  
  --hostname gitlab.formation.org \  
  --publish 443:443 --publish 80:80 --publish 22:22 \  
  --name gitlab \  
  --volume /srv/gitlab/config:/etc/gitlab \  
  --volume /srv/gitlab/logs:/var/log/gitlab \  
  --volume /srv/gitlab/data:/var/opt/gitlab \  
  gitlab/gitlab-ce:latest
```

Configurer également votre */etc/hosts* pour faire pointer *gitlab.formation.org* vers *localhost*

Se connecter sur <http://gitlab.formation.com> avec un navigateur