Cahier de TP « Git / Gitlab »

Pré-requis :

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

TP1 : 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 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

1.4 Suppression et déplacement

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

1.5 Mise en place des fichiers à ignorer

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

- Ignorer les fichiers se terminant par ~, .bak, .log
- Ignorer les fichier .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

TP2 : 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
- Un 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

TP3 : 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***

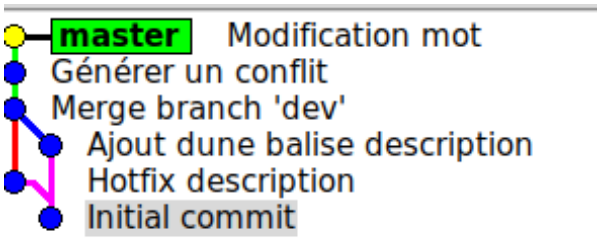
3.2 Rebase

- 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 ?

TP4 : Le serveur Gitlab

4.1 Installation

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

4.2 Groupes de projets, projets et membres

Avec un compte admin, créer

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

Se connecter avec votre compte

Créer 1 groupe de projet

- ***Interne*** : Visibilité privé

Répartir les rôles sur les groupes de projet avec les utilisateurs précédents

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

Pour votre utilisateur, créer une paire de clés *ssh* et la configurer sur le serveur

4.3 Gestion des issues

Avec le compte Owner,

- Créer plusieurs milestone sur le projet

Avec le compte reporter

- Saisir plusieurs issues dont une s'appelant : « CRUD pour delivery-service »
- Discussion sur une issue entre Reporter/Mainteneur projet :
- Saisir quelques commentaires

Avec le compte owner :

- mise au planning et affectation
- Création de 3 labels : BUG, RFC, Refactoring

Avec le compte ***developer***, accès au tableau de bord et déplacement du post-it To Do -> Doing

TP5 : Workflow de collaboration

Travailler à 2 stagiaires sur la même instance de gitlab avec des rôles différentes (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

5.2 Branches distantes

- Créer une branche sur gitlab à partir de master.
- Récupérer la branches 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