TD sur le contrôle de version
par Cyrille Ceccinel

(adapté par MBF pour Fabron, septembre 2015)

Objectifs du TD :

- Découvrir les concepts du contrôle de version
- Développer de manière collaborative
- Utiliser Git comme outil de contrôle de version

Un logiciel de contrôle de version permet de conserver un historique des modifications apportées sur des fichiers tout en offrant la possibilité de revenir à une version antérieure. Il existe des logiciels de contrôle de version centralisés et décentralisés.

Dans le cadre des cours de Conception et de Méthodologie de la production d'applications, nous allons utiliser Git.

Git comme historique des modifications

1) Dans un répertoire quelconque, initialisez un nouveau dépôt
   a. *git init myRepo*
   b. créez-y un fichier `Menu.txt` contenant les plats de votre restaurant favori, ligne par ligne.

```
Salade norvégienne
Œufs fauchés
Steak tartare
Filet de dorade
Profiteroles
```

2) Comment interprétez-vous le résultat de la commande *git status* ?
3) Ajoutez le fichier Menu.txt au gestionnaire de version *(git add Menu.txt)* et réinterprétez le résultat de la commande *git status*.
4) Le fichier Menu.txt est maintenant prêt à être versionné. Editez un message de commit et commitez. Que donne la commande git status ?
   a. Soit vous « commitez » directement *git commit –m « message expliquant le commit »*
   b. Soit vous « commitez » sans message : *git commit* et l'éditeur est lancé pour saisir le message de commit (vi sur les machines IUT (esc. :wq pour sortir).
   c. Soit vous « commitez » sans message : *git commit -F -* et vous saisissez le message de « commit » en ligne.

d. Soit vous écrivez le message lié au commit dans un fichier : git commit -F nomDuFichier

Evidemment vous associez toujours un message à un commit !

5) Apportez quelques modifications au fichier `Menu.txt`. Essayez de commiter ces modifications. Que se passe-t-il ? En vous aidant de la documentation accessible en tapant *git help commit*, versionnez ces modifications (avec un message décrivant le changement).
6) Affichez l'historique des modifications du dépôt.
7) En utilisant *git diff* , visualisez les modifications effectuées entre le premier commit et le second commit.
8) Répétez trois fois la question 5) afin d'avoir des nouvelles versions du menu.

Branches de développement

Sauf indication contraire, vous ajouterez les plats séquentiellement dans le menu (les uns après les autres)

Vous êtes chargé(e) d'introduire des plats végétariens dans le menu du restaurant. Ces plats n'ayant pas encore été validés par la cuisine et la direction du restaurant, vous souhaitez travailler sur la carte sans casser la carte existante. La notion de branche permet de passer instantanément d'une version « stable » (branche « master » créée par défaut) du projet à une « version en cours de développement » (n'importe quelle autre branche que « master »)

9) Créez une branche « vegetarien » dans votre dépôt Git.
    a. *git checkout -b vegetarien*
    b. Véfifiez que vous êtes bien dans la branche « vegetarien » à l'aide de la commande *git branch*.
10) Ajoutez deux/trois plats végétariens au menu et commitez au fur et à mesure les modifications.
11) Observez l'historique des modifications du dépôt, que remarquez-vous ? Vérifiez que vous n'avez pas oublié de commiter les changements.
12) Revenez à la carte principale (branche « master »).
    a. Observez le contenu du fichier Menu.txt
    b. Observez l'historique des modifications, que remarquez-vous ?
13) Ajoutez un plat non-végétarien au menu de la branche master et commitez la modification.
14) Le cuisinier et le directeur du restaurant sont satisfaits de vos propositions de plat et souhaitent maintenant les ajouter au menu principal. Fusionnez (« merge ») la branche « vegetarien » à la branche « master ».
    a. *git merge vegetarien*
    b. Que se passe-t-il ? Ouvrez le fichier Menu.txt et résolvez le conflit de manière à ajouter tous les plats situés entre les balises de conflit (<<<<< et >>>>>).
    c. Commitez le changement (et donc la fusion) en tapant « git commit –a »
15) Créez une branche « japonais » et ajoutez EN TETE DE FICHIER deux/trois plats japonais en commitant au fur et à mesure les modifications.

16) Revenez sur la branche « master » et fusionnez la branche « japonais ». Regardez le contenu du fichier `Menu.txt` et l'historique de modifications du dépôt. Y a-t-il eu un conflit ? Pourquoi ?
17) Supprimez les branches « vegetarien » et « japonais ».
18) Le dernier plat ajouté ne vous plait finalement pas. Il existe deux manières de revenir à une version antérieure : de manière temporaire ou définitive.
    a. Exécuter git log et récupérez le hash (HASH) du commit où vous souhaitez revenir en arrière.
    b. Pour revenir en arrière de manière temporaire, exécutez *git checkout HASH*. Vérifiez que votre fichier Menu.txt est dans son état antérieur. Revenez au dernier commit (HEAD) en exécutant *git checkout master*.
    c. Pour revenir en arrière de manière définitive, et donc supprimer tout ce que vous avez fait depuis ce moment : *git reset --hard HASH*. Dans git log, vérifiez que tout ce que vous aviez effectué depuis ce commit a été effacé.

Synchronisation de votre répertoire

Jusqu'à présent, vous avez pu expérimenter Git pour gérer localement vos versions. Nous allons maintenant nous intéresser au développement collaboratif de fichier sources. Pour cette partie, mettez vous avec votre groupe de méthodologie.

19) (TOUS) En suivant les instructions de la Forge IUT, clonez le dépôt git associé à votre sous-groupe.
    a. Récupérer l'identifiant de votre sous-groupe dans la forge sous Configuration -> Informations -> Identifiant
    b. git clone https://LOGIN@git-iutinfo.unice.fr/2015-groupe-X-Y

20) (Personne A) Créez un fichier index.html reprenant le code ci-dessous :

```
<html>
<head>
<title>Demo</demo>
</head>
<body>
</body>
</html>
```

Ajoutez le fichier au dépôt git et commitez le fichier. Envoyez ensuite le commit vers le dépôt distant : *git push origin master ou git push*

21) (Tous sauf A) Synchroniser votre dépôt git avec la commande *git pull.*
22) (Un développeur autre que A) Modifiez le fichier index.html afin d'ajouter du texte entre les balises *body*. Commitez les modifications et envoyez-les vers le dépôt distant.
23) (TOUS) Synchroniser votre dépôt git avec la commande *git pull*.
24) (Personne A) Modifiez le titre de la page Web et commitez/envoyez les modifications.

25)(Personne B) (SANS SYNCHRONISER LE DEPOT) Modifiez le titre de la page Web et commitez/envoyez les modifications. Que se passe-t-il ? Remédiez au conflit.

Ressources :

Pro Git, Scoot Chacon and Ben Straub https://git-scm.com/book/fr/v1
Prise en main de Git : http://www-igm.univ-mlv.fr/~dr/XPOSE2008/git/
Learning Git in 15 minutes : https://try.github.io/levels/1/challenges/1
A successful git branching model : http://nvie.com/posts/a-successful-git-branching-model/