LP DAM 22/11/2016

TD Versioning
Prise en main de Git

L'ensemble de ce TD est inspiré des documents de S. Mosser, P. Collet, C. Cecchinel et Simon Urli.

0. Modalités et consignes

Toute consigne non respectée entraînera pénalité!

Le travail est individuel et est à effectuer au cours de la séance.

Le travail de la question I. est à effectuer avant de passer à la suite, et doit être réalisé dans un délai de 15 minutes au maximum, sur papier.

Le travail des questions II à VI doit se faire depuis un terminal de commande et être poussé sur Github (Classroom). Tout commit non pushé ou pushé après la fin de la séance ne sera pas considéré.

Dans chaque message de commit, préciser la question à laquelle il fait référence.

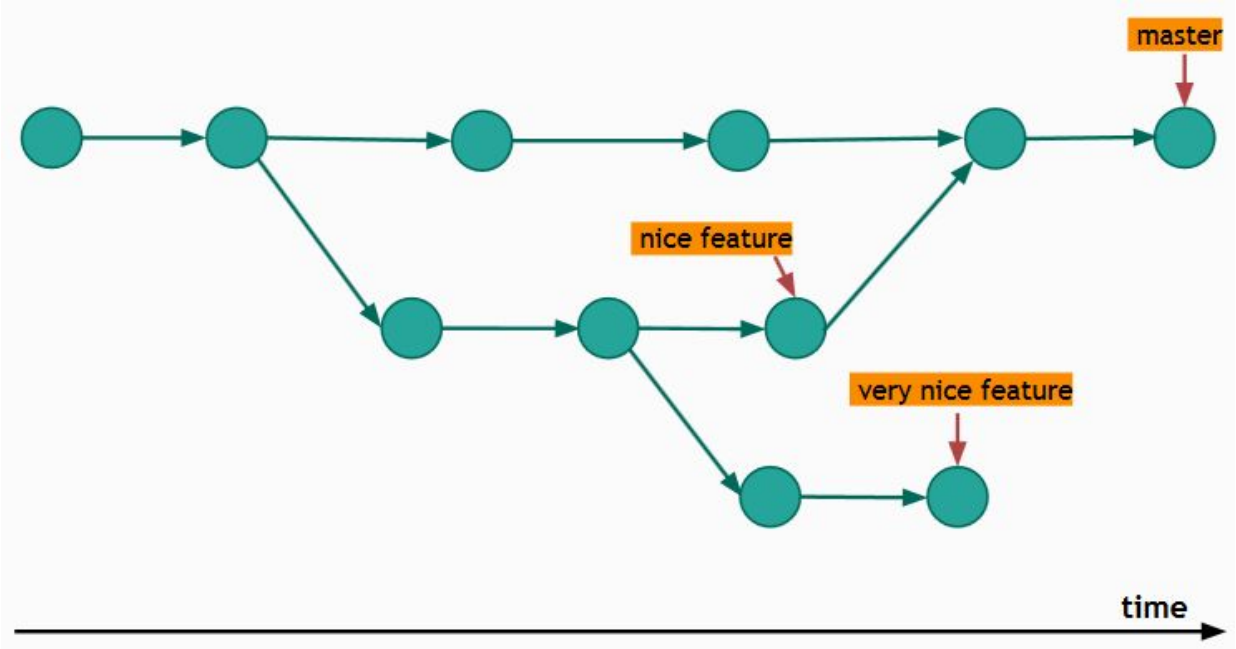
Exemple : “III.5 - mon message de commit”

I. Représentation de l'historique git (15 min max)

Lisez toute la partie I. avant de commencer!

- Lisez les questions III.1 à V.12 de l'énoncé.
- Vous devez représentez le repository tel qu'il devrait être après la question V.12.
- Effectuer le travail sur papier. Inscrivez votre nom et rendez quand vous avez fini.

Utiliser la notation vue en cours:



Représentation des éléments:
- Un commit = un cercle.
- Position d'une branche = nom de la branche + flèche vers le commit.
- Position d'un tag = nom du tag + flèche vers le commit.

Sous chaque commit:
- L'opération effectuée (ajout d'un fichier X, modification d'un fichier Y)
- Si nécessaire, préciser où l'on se trouve lors du commit (sur quelle branche ou dans quel dossier)
- Si c'est un commit de merge, préciser quelle branches sont fusionnées

A la fin de chaque partie (III, IV, V.12)
- indiquer le commit auquel on se trouve. ("fin III" + flèche vers le commit")

II. Mise en place

- Git bash devrait être installé sur les machines, dans le cas contraire il peut être téléchargé ici : https://git-scm.com/downloads

- Lancer 'Git Bash' et configurer Git pour qu'il associes vos commits à votre nom et adresse e-mail:

```
git config --global user.name "Votre Nom"
git config --global user.email supermail@mydomain.com
```

Si un commit ne peut être identifié comme provenant de vous il est considéré comme **anonyme** lors des évaluations ! **=> 0**

- Si ce n'est pas fait, créez-vous un compte Github.
- S'enregistrer pour l'assignement Github Classroom suivant : https://classroom.github.com/assignment-invitations/dfa118be888ef12214f4aa193637d9a8
Il vous faudra vous loguer dans Github classroom, autoriser l'application à accéder à votre compte et enfin indiquer vouloir l'assignement pour que Github vous crée automatiquement un repository.

Vous ne devez pas créer de repository vous même ! **Mauvais repo => 0**

- Aller dans le répertoire de votre choix sur votre ordinateur et effectuez la commande avec l'url fournie par Github : `git clone https://github.com/LP-IDSE-16-17/{...}.git`
Ceci va cloner le repository distant localement sur votre machine. Allez dans le dossier qui a été créé en clonant : `cd dam1617-td-git-{githubUserName}`

III. Actions basiques

1- Créez un fichier Menu.txt contenant des plats de restaurant, ligne par ligne :

- Steak tartare
- Salade norvégienne
- Filet de dorade
- Bar à la citronnelle

2- Vérifiez le résultat en tapant la commande `git status`
3- Utilisez les commandes `git add` et `git commit` pour faire votre premier commit contenant le fichier Menu.txt, **attention à ne pas oublier le message de commit !**

> Il existe plusieurs manières de committer en laissant un message de commit :
>
> - git commit -m “le message de commit”,
> - git commit sans argument vous ouvre un éditeur (vi par défaut - echap et :wq pour sortir en écrivant le message)
> - et d'autres que vous pourrez retrouver dans la documentation

4- Modifiez le fichier Menu.txt et committez ces modifications.
5- Poussez les commits effectués grâce à la commande `git push -u origin master`
6- Affichez l'historique des modifications et enregistrez les dans un fichier log.txt que vous ajoutez au dépôt.

> Sous un shell linux vous pouvez utiliser ‘>’ pour que la sortie standard soit redirigée sur un fichier. Par exemple ls -l > toto.txt écrit la liste des fichiers du répertoire courant dans le fichier toto.txt.

7- Visualisez les modifications effectuées dans Menu.txt entre le premier et le deuxième commit grâce à `git diff` et enregistrer le résultat dans un fichier `diff.txt` que vous ajoutez au dépôt.

> `git diff` prend en paramètre l’identifiant du commit avec lequel vous souhaitez faire l’opération de diff (par rapport à l’état actuel du repo). L’identifiant du commit est une chaîne de caractère unique pour chaque commit, trouvable en utilisant `git log`.

8- Committez ces deux fichiers
9- Poussez les commits effectués.

IV. Gestion de conflits

1- Sortez de votre répertoire git, créez un nouveau dossier tmp/, allez dans ce dossier et clonez à nouveau le dépôt dans ce nouveau dossier.
2- Faites un commit sur ce nouveau dépôt en ajoutant 2 éléments de menu, et pushez.
3- Retournez sur le premier dépôt. L’autre ne servira plus.
4- Modifiez le menu en ajoutant également deux plats, dont le 1er est le même qu’à la question 2.
5- Committez. Pushez. Que se passe-t-il?
6- Résolvez le conflit en gardant le plat en commun et un seul des deux autres plats.
4- Committez le merge (attention à ajouter le fichier Menu.txt avant).
5- Poussez les changements.

V. Introduction aux branches (sans git flow)

1- Créez une branche develop dans le dépôt : `git branch develop`
2- Basculez sur cette branche : `git checkout develop`
3- Vérifiez que vous êtes sur la bonne branche : `git branch`
4- Ajoutez des desserts à la fin du Menu.txt et committez au fur et à mesure vos modifications
5- Basculez sur la branche principale (master)
6- Observez le fichier Menu.txt et modifiez des plats, puis committez. Ajoutez au moins un dessert en bas du fichier et une entrée au début.
7- Fusionnez la branche que vous aviez créée : `git merge develop`
8- Où y a-t-il des conflits? Résolvez les conflits et committez.
9- Pushez les changements.
10- Créez un tag appelé v1.0 à ce commit : `git tag -a v1.0 -m "version 1.0"`
11- Visualisez le tag : `git show v1.0`
12- Pushez le tag : `git push origin v1.0`
13- Retournez sur develop.
14- Mergez master dans develop afin d'être à jour.
15- Pushez les changements avec '`git push`'.
16- Que se passe-t-il? Faites la bonne commande pour pousser la branche develop.

VI. Introduction à Git flow

1- Initialisez git flow dans votre dépôt : `git flow init`. Conservez les choix par défaut de git flow (appuyer sur entrée pour valider les choix).
2- Commencez le développement d'une nouvelle feature dans le dépôt : `git flow feature start menuSpecial`.
3- Qu'a fait cette commande?
4- Ajoutez des plats dans Menu.txt.
5- Commitez et poussez.
6- Terminez la feature : `git flow feature finish menuSpecial`
7- Qu'a fait cette commande?
8- Faites à nouveau un push.
9- Préparez une nouvelle release : `git flow release start v2.0`
10- Modifiez le fichier Menu.txt pour le mettre en forme (par exemple séparez proprement entrées, plats, desserts, mettez un titre)
11- Terminez la release : `git flow feature finish v2.0`
12- Poussez le tout.
13- Vous pouvez aller vérifier sur Github si tous les tags, commits et branches ont bien été poussés.

Pour aller plus loin…

http://pcottle.github.io/learnGitBranching/ : des exercices interactifs sur le branching avec Git
http://nvie.com/posts/a-successful-git-branching-model/ : article de référence sur Git Flow
http://danielkummer.github.io/git-flow-cheatsheet/ : page très claire présentant l'utilisation de Git Flow

http://git-scm.com/docs : et bien sûr la doc officielle !

Pour éviter la console… (après ce TD)

- Les différents IDE (Visual Studio, IntelliJ, Eclipse) possèdent des plugin permettant d'utiliser git.
- Des outils externes existent, tels que Sourcetree.