TP Git

Thomas Clavier

Présentation

Git est un logiciel de gestion de versions décentralisé. C'est le plus utilisé, d'après openhub.net il représente à lui seul plus de 89% des projets utilisant un gestionnaire de versions décentralisé.

Des objets

Un dépôt Git peut être vu comme une collection d'objets liés entre eux. Chaque objet est identifié par une chaîne de 40 caractères hexadécimaux correspondant à la somme de contrôle de son contenu. Il y a 3 types d'objets :

- blob : les données
- tree : les arborescences
- commit : une version du répertoire de travail

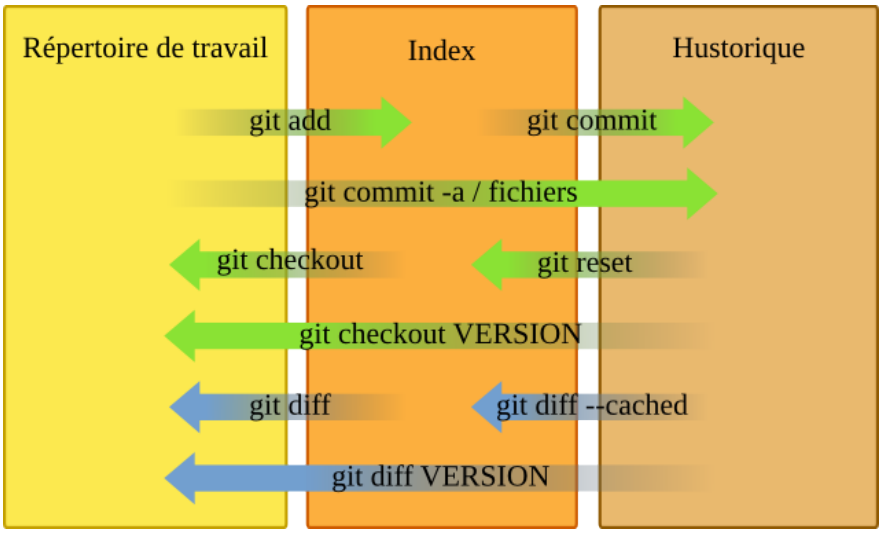
Un objet de type «arborescence» peut contenir des objets «données» et «arborescence». Le répertoire de travail est lui-même un objet de type «arborescences».
L'historique d'un dépôt Git c'est l'ensemble des versions du répertoire de travail. Pour identifier une version, Git s'appuie sur un objet de type «commit». L'objet de type «commit» associe de nombreuses informations comme l'auteur, un message, une version de l'objet répertoire de travail mais aussi les «commits> parents.

Les espaces

Git utilise 3 espaces différents pour manipuler toutes ces données.

- le répertoire de travail
- l'index
- l'historique

Le répertoire de travail présente dans un dossier de la machine, l'ensemble des fichiers du dépôt. C'est le point d'entrée de l'utilisateur.
L'index contient les données en préparation pour le commit
La tête (HEAD) de l'historique contient le dernier commit.



configuration

Configuration de git :

```
git config --global user.email jean.dupont@example.com
git config --global user.name 'Jean Dupont'
```

L'espace de travail

À partir de là nous allons construire un blog de promotion.

Initialisation

Création d'un dépôt local un blog en ligne de tous les élèves de la promotion.

```
mkdir /tmp/blog
cd /tmp/blog
git init .
```

Historique de version

Création d'un premier fichier à votre nom, il contiendra votre CV au format Markdown (`http://daringfireball.net/projects/markdown/syntax`).
Pour ce premier commit nous allons le créer avec juste votre nom dedans :

```
echo "#Prénom Nom" > nom-prenom.mkd
```

L'ajouter dans l'index comme fichier à suivre

```
git add nom-prenom.mkd
```

Validez l'index pour créer un commit

```
git commit -m "première version du cv de Nom Prénom"
```

Question 1:
Observez l'historique des changements, quelles sont les informations disponibles ? À quoi correspond chaque champ ?

```
git log
```

Question 2:
Pour avoir de l'aide sur git, il est possible de lancer "man git". Pour avoir de l'aide sur la commande "git log" il est possible de lancer "man git-log".
Comment avoir l'affichage suivant :

```
commit 062f166cbbd729a90a30a31578f10953d7a05dc5
Author: Nom Prénom <prenom.nom@example.com>
Date:   Sun Mar 1 11:49:45 2034 +0100

    première version du cv de Nom Prénom

 nom-prenom.mkd | 5 ++++
 1 file changed, 1 insertion(+)
```

Question 3:
Que représentent les "+" ?
Ajoutez quelques informations dans votre CV.

```
gedit nom-prenom.mkd
git commit -m "Une version plus à jour du CV de mon CV" -a
```

Question 4:
Que font les commandes suivantes :

```
echo "Une nouvelle ligne" >> nom-prenom.mkd
git blame nom-prenom.mkd
git status
git diff
git commit -m "Commit uniquement pédagogique" -a
```

Question 5:
Comment obtenir le diff entre les 2 derniers commits ?
Créez le CV de Satoshi Tajiri

```
gedit satoshi-tajiri.mkd
git add satoshi-tajiri.mkd
git commit -m "Ajout du Cv de Satoshi Tajiri"
```

Nous construisons l'blog de la promo ...

```
git rm satoshi-tajiri.mkd
git commit -m "suppession du CV de Satoshi Tajiri"
```

Question 6:
Que voit-on dans l'historique (git log) ?
Tous les CV sont au format Markdown mais nous allons publier du html. Nous allons déplacer les CV dans un sous répertoire cv pour bien séparer le code html des cv.

```
mkdir cv
git mv nom-prenom.mkd cv/nom-prenom.mkd
git commit -m "Création du répertoire cv"
```

Question 7:
Quel aurait été le résultat si nous avions remplacé le git mv par les 3 commandes suivantes :

```
mv nom-prenom.mkd cv/nom-prenom.mkd
git add cv/nom-prenom.mkd
git rm nom-prenom.mkd
```

Partager

Via un dépôt partagé

Ajouter votre clé ssh sur le serveur gitlab : https://git.iut-info.univ-lille1.fr/profile/keys/new
Dans le champ "Key" coller le contenu du fichier «~/.ssh/id_rsa.pub». Si ce fichier n'existe pas, le créer avec la commande ssh-keygen.
Dans le champ Title donner un nom à votre clé, par exemple «tp iut».
Indiquons à Git que nous souhaitons échanger des données avec le dépôt distant suivant : git@git.iut-info.univ-lille1.fr:thomas.clavier/2014tpgits2.git Nous appelons ce dépôt «origin».

```
git remote add origin \
git@git.iut-info.univ-lille1.fr:thomas.clavier/2014tpgits2.git
```

Envoyons l'ensemble des modifications de la branche courante (master) vers le dépôt «origin»

```
git push origin master
```

Question 8:
Que se passe-t-il ?
Question 9:
Pour récupérer le travail fait par le reste du groupe avant d'envoyer ses propres modifications il suffit de lancer :

```
git pull
```

Avant de renvoyer le tout sur le serveur :

```
git push origin master
```

Expliquez le résultat de la commande suivante :

```
git log --oneline --graph --decorate
```

Si l'on souhaite obtenir une copie de travail d'un dépôt distant existant il est possible d'utiliser la commande :

```
git clone url
```

Question 10:
Quelles sont les formes d'URL possibles ?
Lançons la génération des pages html :

```
make html
```

Question 11:
Que donne la commande suivante ? Pourquoi ?

```
git status
```

Question 12:
Est-il normale d'enregistrer du code généré ?
Pour ignorer tout le répertoire de génération modifions le fichier .gitignore

```
gedit .gitignore
```

Lancez la commande suivante :

```
make pok
```

Question 13:
Expliquez ce qui s'est passé. Vous pouvez utiliser la commande "git status" pour vous aider.
Question 14:
Que fait la commande suivante ?

```
git clean -f
```

Modifiez le fichier changelog pour indiquer que la nouvelle version de l'blog comporte votre nom

```
gedit changelog
```

Question 15:
Enregistrez et partagez vos modifications, que se passe-t-il ?
Question 16:
À quoi correspondent les lignes suivantes :

```
<<<<<<< HEAD
DD MM YYYY : Monsieur Patate <monsieur@patate.org>
- Monsieur Patate est dans l'blog
=======
DD MM YYYY : Madame Patate <madame@patate.org>
- Madame Patate est dans l'blog
>>>>>>> branch-a
```

Question 17:
Comment résoudre ce problème ?

Par mail

```
git format-patch
git send-email
git archive
```

Gérer le code

Pour identifier un commit avec un nom facile à retenir, il est possible de poser un tag. C'est par exemple utilisé pour marquer une version donnée.

```
git tag 1.0.0
```

Des branches

Pour gérer plusieurs versions de l'ensemble du code en parallèle, Git propose de faire des «branches».

Question 18:

- Ajoutez une branche "nom-prenom-merge",
- dans cette branche modifiez votre CV (en fin de fichier), puis commitez (faire au moins 2 modifications et 2 commits).
- Revenez sur la branche master,
- faites 2 modifications en début de fichiers (avec 2 commits)
- puis fusionnez la branche "nom-prenom" avec master.

```
git merge nom-prenom-merge
```

Observez le résultat dans l'arbre des versions.

```
git log --graph --oneline --all
```

Expliquer ce qui c'est passé.

Question 19:

- Ajoutez une branche "nom-prenom-rebase",
- dans cette branche modifiez votre CV, puis commitez (faire au moins 2 modifications et 2 commits).
- Revenez sur la branche master,
- et faites à nouveau 2 modifications en début de fichier (avec 2 commits)
- puis «rebasez».

```
git rebase nom-prenom-rebase
```

Observez le résultat dans l'arbre des versions.

```
git log --graph --oneline --all
```

Expliquer ce qui c'est passé.

Question 20:

Au moment de faire un pull pour rapatrier le travail du reste de l'équipe, il est possible de faire un pull rebase.

```
git pull -r
```

Expliquer ce que fait cette commande.

Plus loin

Question 21:
Expliquez chacune des commandes suivantes :

```
git log --graph --pretty=format:\
'%Cred%h%Creset -%C(yellow)%d%Creset \
%s %C(bold blue)<%an>%Creset%n' --abbrev-commit --all
git stash
git stash pop
git bisect
git staged
git unstage
git fix
```