

# TP Git

Thomas Clavier

## 1 configuration

Configurez votre nom (user.name) votre email (user.email) et le serveur mandataire (http.proxy) dans git. L'URL du serveur mandataire de l'université est <http://cache.univ-lille1.fr:3128>

## 2 L'espace de travail

À partir de là nous allons construire un blog pour le groupe de TP.

### 2.1 Initialisation

Créez un répertoire de travail et initialisez le comme un dépôt git local.

### 2.2 Historique de version

Dans ce blog, tous les articles sont rédigé en utilisant le format Markdown (<http://daringfireball.net/projects/markdown/syntax>).

#### 2.2.1 Premier article

Dans l'espace de travail que vous avez initialisé comme dépôt local, créez un sous répertoire «content».

Avec votre voisin choisissez un sujet, puis créez le même article «content/votre-sujet.md» chacun dans votre dépôt local avec le contenu suivant :

```
+++  
date = "AAAA-MM-DDT12:00:00+01:00"  
draft = true  
title = "Mes réponses au TP Git"  
+++
```

# TP Git

Attention, «AAAA-MM-DD» est la date du jour au format Année-mois-jour.

Vous pouvez par exemple faire un article avec les réponses à ce TP.

L'ajouter dans l'index comme fichier à suivre, puis validez le changement et enfin observez l'historique des changements, quelles sont les informations disponibles ? À quoi correspond chaque champ ?

### 2.2.2 Second changement

Ajoutez quelques informations dans votre article puis validez un nouveau commit.

Par exemple :

```
+++
date = "AAAA-MM-DDT12:00:00+01:00"
draft = true
title = "Mes réponses au TP Git"
+++

# TP Git
## Configuration

git config --global user.name "Pierre Dupond"
git config --global user.email "pierre.dupond@univ-lille1.fr"
git config --global http.proxy http://cache.univ-lille1.fr:3128
```

Comment obtenir le diff entre les 2 derniers commits ?

### 2.2.3 Suppression

- Créez un article «content/pokemon.md» puis validez un nouveau commit.
- Supprimez l'article «content/pokemon.md» puis validez un nouveau commit.

Que voit-on dans l'historique ? Comment faire pour retrouver le fichier en question ?

### 2.2.4 Retour en arrière

joutez quelques informations dans votre article sans valider un nouveau commit.

Comment faire pour annuler le travail et revenir à la dernière version «committée» du fichier ?

- Créez un article «content/satoshi-tajiri.md» puis validez un nouveau commit.

Comment faire pour annuler ce commit et revenir avec une copie de travail sans ce dernier fichier en HEAD ?

## 3 Partager

### 3.1 Via un dépôt partagé

Indiquons à Git que nous souhaitons échanger des données avec le dépôt distant suivant : <http://git.iut.azae.net/git/tp-git.git> Nous appelons ce dépôt «origin».

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

Que se passe-t-il ?

Visitez <http://git.iut.azae.net/> pour admirer votre travail.

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
```

Quelles sont les formes d'URL possibles ?

### 3.1.1 Ignorer

Lançons la génération des pages html :

```
./bin/hugo --theme=hyde --buildDrafts
```

Que donne la commande suivante ? Pourquoi ?

```
git status
```

Est-il normale d'enregistrer du code généré ? Comment faire pour ignorer les nouveaux fichiers présent dans le répertoire «public» ?

## 4 Gérer le code

Pour cette partie du TP, il est préférable de ne plus faire de pull ou de push. 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.

### 4.1 Des branches

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

- Ajoutez une branche "nom-prenom-merge",
- dans cette branche modifiez votre article (en fin de fichier), puis validez le commit (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-merge" avec master.

Observez le résultat dans l'arbre des versions.

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

Expliquer ce qui c'est passé.

- Ajoutez une branche "nom-prenom-rebase",
- dans cette branche modifiez votre article (en fin de fichier), puis validez votre commit (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».

Observez le résultat dans l'arbre des versions.

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

Expliquez ce qui c'est passé.

## 4.2 Stash

Il est possible de mettre son travail de coté pour lancer des opérations plus urgente, par exemple une fusion.

- modifiez votre article, puis validez le commit,
- modifiez votre article sans validez le commit,
- faire un «git stash»
- observer le contenu de votre article
- modifiez votre article sans validez le commit,
- faire un «git stash»
- observer le contenu de votre article
- faire un «git stash pop»
- observer le contenu de votre article

Expliquez ce qui c'est passé.

## 4.3 Squash

Il est possible de réécrire l'hitoire, par exemple pour fusionner plusieurs commits.

- Créez une nouvelle branche «nom-prenom-squash» et basculer dedans,
- modifiez votre article, puis validez votre commit (faire au moins 4 modifications et 4 commits).
- à l'aide de «git rebase -i » fusionner les commits 2 par 2

Expliquez ce qui c'est passé.

## 5 Plus loin

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 bisect
git stage
git unstage
git blame
git format-patch
git send-email
git archive
```