Configurer son git comme un pro

Lors de ce TP, nous verrons comment tirer partie de git à son maximum.

Git, tout en étant puissant est en effet vraiment customisable.

Git config ou `.git/config` ?

Nous avions lors d'un précédent TP comment configurer son nom et son email, pour que chaque commit soit correctement identifié.

Nous avions utilisé alors la commande `git config` pour configurer l'email et le nom du commiter.

Que s'est-il réellement passé au final?

- Sur votre ordinateur, créer un dossier "git-configuration".
- Dans le dossier "git-configuration", initialiser le dépot Git: `git init` .
- Re-définir son email et son nom d'utilisateur à l'aide de la commande `git config` .
- Ouvrir le fichier ".git/config" (le "." est important).
- Exécuter la commande "git config user.name".

Gestion des alias

Il est possible - comme en bash - de définir des alias de commande git. De nombreux templates et resources existent, nous allons cependant apprendre à en créer un.

- A l'aide du terminal, accéder au dossier "git-configuration".
- Exécuter la commande `git status` suivi de `git --version` .
- Exécuter la commande `git config --global alias.st status` .
- Exécuter la commande `git st` .
- Exécuter la commande `git config --global alias.v --version` .
- Exécuter la commande `git v` : que s'est-il passé?

Afficher la branche git dans son prompt

> *Utilisateurs de zsh ce script n'est pas compatible avec votre Shell. Si ce n'est pas déjà fait, installez "oh my zsh et activez le plugin git".*

Que vous soyez sur Windows, Mac OSX ou GNU/Linux un terminal bash est à votre disposition. Il est assez facile de customiser le “prompt” pour afficher quelques informations supplémentaires.

- Sur GNU/Linux, s’il n’existe pas créer le fichier “.bashrc” dans “~”.
- Pour MacOS ou Windows, créer le fichier “.bash_profile” dans votre dossier utilisateur.
- Ajouter dans le fichier “.bashrc” ou “.bash_profile” le contenu suivant:

```
parse_git_branch() {
     git branch 2> /dev/null | sed -e '/^[^*]/d' -e 's/* \(.*\)/ (\1)/'
}

export PS1="\u@\h \[\033[32m\]\w\[\033[33m\]\$(parse_git_branch)\[\033[00m\] $
 "
```

- Ré-ouvrir Git Bash sur Windows ou un terminal sur Mac OSX ou GNU/LInux et accéder à un dépôt git.
- Effectuer une capture d’écran de votre terminal.

Appliquer une configuration globalement

Jusque là nous avons configuré le comportement d’un dépôt git, mais dans le cas des alias ou d’autres options, ça a plus de sens d’appliquer une configuration globale.

- Ouvrir le fichier “.gitconfig” non pas au niveau d’un dépot git mais au niveau du dossier utilisateur:
  - Dans “~” pour GNU/Linux et Mac OSX.
  - Dans `C:\Documents and Settings\$USER` ou `C:\Users\$USER` selon la version de Windows.
- Dans ce fichier - s’il n’existe pas créez-le - ajouter le contenu suivant:

```
[user]
    name            = Prenom Nom
    email           = mail@mail.fr

[alias]
    st              = status
    co              = checkout
    br              = branch -v
    lg              = log --graph --pretty=tformat:'%Cred%h%Creset -%C(cyan)%d
```

```
 %Creset%s %Cgreen(%an %cr)%Creset' --abbrev-commit
    lga             = log --graph --pretty=tformat:'%Cred%h%Creset -%C(cyan)%d
 %Creset%s %Cgreen(%an %cr)%Creset' --abbrev-commit --all
    di              = diff -M
    sdi             = diff --cached -M
    dis             = diff --ignore-all-space -M
    logfull         = log --pretty=fuller --graph --stat -p
    unstage         = reset HEAD
    uncommit        = update-ref HEAD HEAD^
    uncommithard    = reset --hard HEAD^
    oups            = commit --amend -C HEAD
    tagg            = ! git fetch origin -t && git tag -f -a -m 'created tag :
 `cat VERSION`' `cat VERSION` && git push origin --tags
    tagc            = ! TAG=`git describe --tags $(git rev-list --tags --max-co
unt=1) | sed 's/v//' | awk -F . '{ printf "v%d.%d.%d", $1, $2, $3 + 1 }'` && ec
ho $TAG > VERSION && git ci -m "Bumped to version : $TAG" VERSION && git tag -a
 -m "Created tag $TAG" $TAG
    rebc            = rebase --continue
    rebt            = rebase --abort
    rebs            = rebase --skip
    prev            = checkout HEAD^1
    next            = "!sh -c 'git log --reverse --pretty=%H master | awk \"/
$(git rev-parse HEAD)/{getline;print}\" | xargs git checkout'"

[color]
    ui              = true

[core]
    editor          = vim
    #excludesfile    = ~/.gitglobalexclude

[push]
    default         = tracking
```

- Fermer et ré-ouvrir le terminal.
- Retourner sur un dépot Git et tester quelques commandes.