INTRODUCTION AND WORKING FLOW

Nicolas Fazio
@FazioNico

PROGRAMME

Présentation de git
Installer git
Utiliser git
Demo + TP
Les Branches
Utilisation de github

PRÉSENTATION

GIT C'EST QUOI ?

Un système de gestion de version décentralisé !
Créé en 2005 par Linus Torvalds

Logiciel libre et distribué sous licence GPL 3

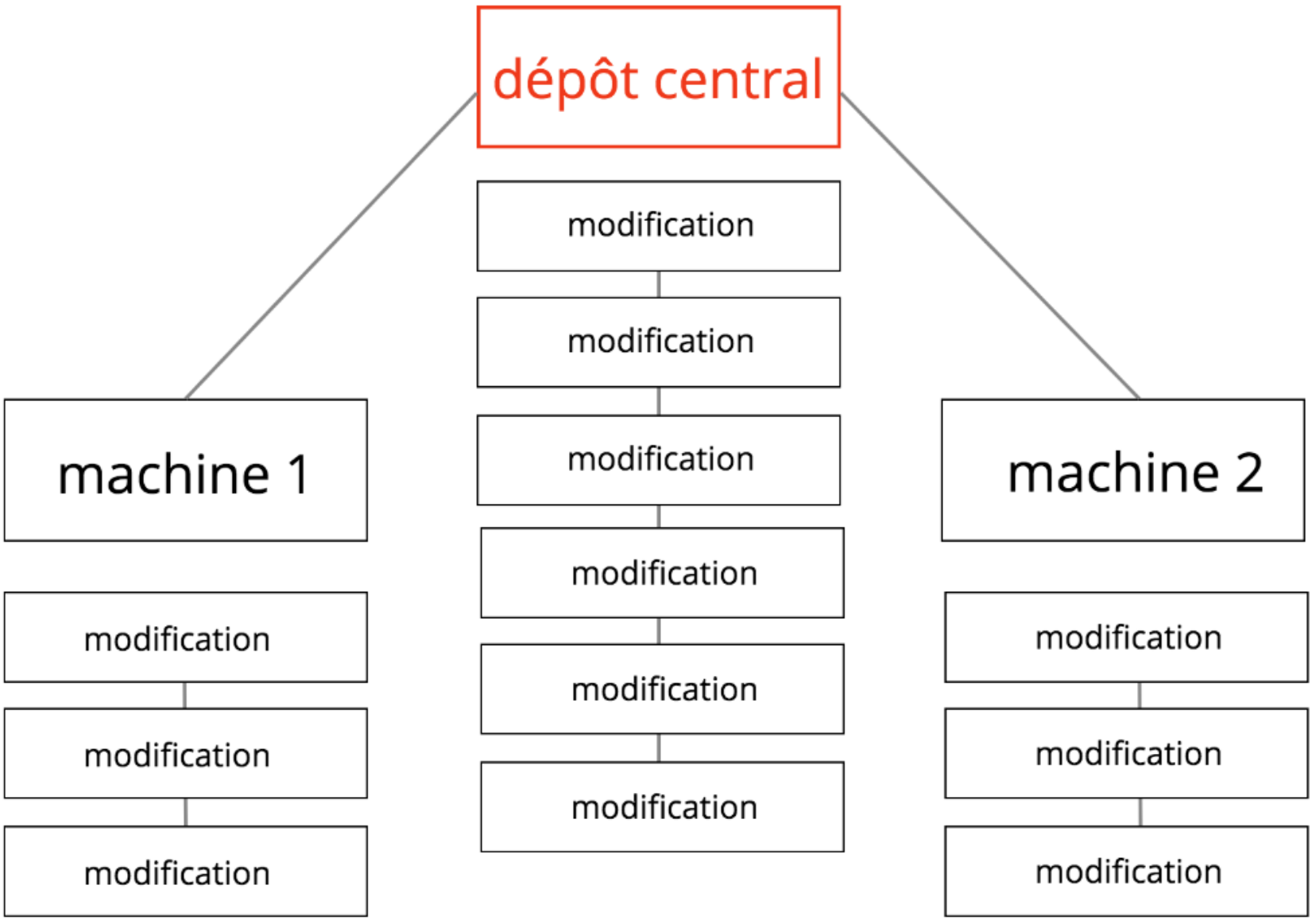
A QUOI SERT GIT ?

Permet de versioner son code afin de garder un historique des modification effectuées.

Travailler à plusieurs sur le même code.

COMMENT ÇA MARCHE ?

- Un dépôt principal permet de centraliser l'information.
- Des clones de ce dépôt sont créés sur les machines des développeurs.
- On envoie régulièrement les modifications locales sur le dépôt.
- Git gère les conflits sur les modifications du code.


dépôt central
modification
modification
modification
modification
modification
modification
machine 1
modification
modification
modification
machine 2
modification
modification
modification

INSTALLATION

COMMENT INSTALLER GIT

Mac OSX

http://sourceforge.net/projects/git-osx-installer/

Window

http://msysgit.github.io

CONFIGURER GIT

- **/etc/gitconfig :** Contient les valeurs pour tous les utilisateurs et tous les dépôts du système. Si vous passez l'option --system à git config, il lit et écrit ce fichier spécifiquement.
- **~/.gitconfig :** Spécifique à votre utilisateur. Vous pouvez forcer Git à lire et écrire ce fichier en passant l'option --global.
- **.git/config :** (du dépôt en cours d'utilisation) : spécifique au seul dépôt en cours. Chaque niveau surcharge le niveau précédent, donc les valeurs dans .git/config surchargent celles de /etc/gitconfig.

CONFIGURER GIT [SUITE]

$ git config --global user.name "John Doe"
$ git config --global user.email johndoe@example.com

Nécessaire **qu'une fois si vous passez l'option --global**, parce que Git utilisera toujours cette information pour tout ce que votre utilisateur fera sur ce système.

Si vous souhaitez surcharger ces valeurs avec un nom ou une adresse e-mail différents pour **un projet spécifique, vous pouvez lancer ces commandes sans option --global lorsque vous êtes dans ce projet.**

Verifier la configuration:
$ git config --list
ou $ git config user.name

UTILISATION

CORE DEFINITION

- **Commit:** ensemble de modifications
- **Working copy:** les modification en cour
- **Staging Area:** liste de modifications effectuée dans le working copy qui sera ajouter au repository lors du prochain commit
- **Repository:** ensemble des commits du projet

INTERFACE UTILISATEUR

- On utilise git depuis le CLI de votre machine (terminal, MS DOS).
- Plugins pour IED (éditeur de texte)
- Logiciel comme GitHub pour Mac et Windows, SourceTree, GitKraken, Fork etc...

Atom PlatformIO IDE Terminal: https://atom.io/packages/platformio-ide-terminal
Atom git-plus: https://atom.io/packages/git-plus

INITIALISER GIT POUR UN PROJET

```
$ mkdir project-name
$ cd project-name
$ touch README.md
$ git inti
$ git add .
$ git commit -m "first commit"
```

AJOUTER DES FICHIER & VALIDER

Pour **effectuer un changement** (ajout à l'index):
$ git add mon-fichier
$ git add *

Pour **lister les changements** (voir l'index) avant validation:
$ git status

Pour **valider ces changements**:
$ git commit -m "Message de validation"

DÉMO

TP

(installer git sur votre machine)
(installer terminal-plus & PlatformIO IDE Terminal)
créer un repo local 'git-training'
faire votre 'first commit'
ajouter un fichier 'index.html' dans un dossier 'app/'
sauvegarder les modifications

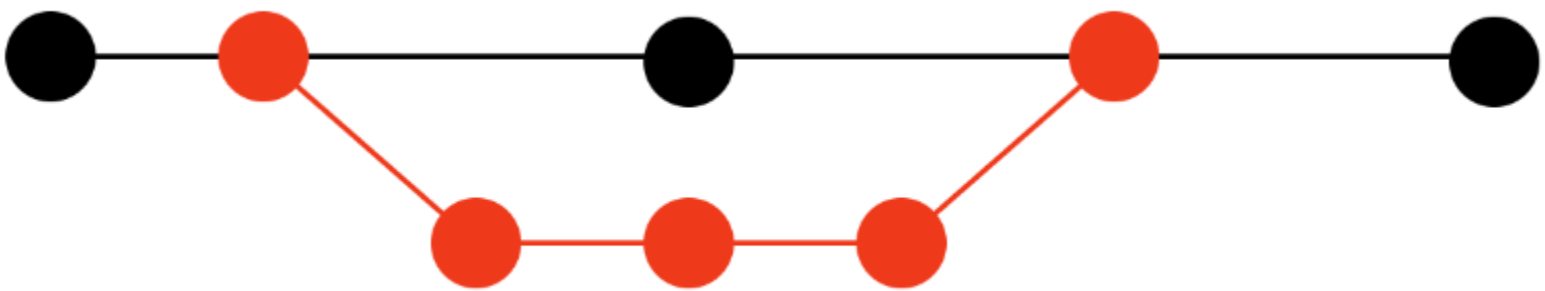
LES BRANCHES

PRESENTATION DES BRANCHES

Une branche est une version parallèle du projet.
C’est comme si vous aviez une copie du code de votre projet.

Cela permet de tester de nouvelles idées et de vérifier si elles fonctionnent avant de les intégrer au véritable code du projet.

EXEMPLE DE BRANCHES



master
branch feature

GESTION DES BRANCHES

Lister les branches existante:
$ git branch

Créer une branch et s'y déplacer
$ git branch nom-branch
$ git checkout nom-branch

Supprimer une branch local
$ git checkout master
$ git branch -d nom-branch

MERGE DES BRANCHES

Pour merger une branch feature sur la branch master:

- se déplacer sur la branch de destination
$ git checkout master
$ git merge feature
ou
$ git merge feature --no-ff
$ git branch -d feature (supprimer la branch)

Supprimer une branch local
$ git checkout master
$ git branch -d nom-branch

MERGE CONFLITS

Lorsque deux développeurs modifient le même fichier en même temps, ces modifications rentrent en conflit.

Si certains conflits sont réglés automatiquement, d'autres doivent l'être manuellement.

RESOLUTION DES CONFLITS

Ici, on gardera la deuxième ligne si on veux afficher "tata", la quatrième si on veux afficher "titi".

```
<<<<<<< HEAD
        let title = "tata";
=======
        let title = "titi";
>>>>>>> 05ad020ca4d641797f42cb107967662c4d9e7e1f
```

DEMO

TP

- créer une branch 'feature.test'
- modifier le fichier existant
- ajouter un nouveau fichier test.js
- merger la branch 'feature.test' sur master
- supprimer la branch 'feature.test'

GIT ET GITHUB

PRÉSENTATION DE GITHUB

Github permet de publier son repo sur un serveur en ligne afin de pouvoir travailler à plusieur sur le même code.

Le fonctionnement est identique à son fonctionnement en local. Il y a uniquement quelque commande de plus qui permettent de publier son code sur github.

ENVOYER SON CODE SUR GITHUB

Créer un compte sur github.com

Relier votre repo local à github:

- cliquer sur 'create new repository' et choisit l'option 'public' sans readme.md
- choisir 'existing repo', copier le code et le coller dans ton CLI à la racine de ton projet.

Commande pour envoyer branch sur github:
$ git checkout nom-branch
$ git push -u origin nom-branch

IGNORER DES FICHIERS

On peut ignorer certains fichiers ou documents que l'on ne veut pas envoyer sur github.
Exemple: fichiers de configuration BDD, nodes modules...

Créer un fichier nommé 'gitignore' à la racine de votre projet.

```
# liste des fichiers
# ignoré par git

node_modules/
www/build/
platforms/
plugins/
*.swp
.htaccess
.DS_Store
Thumbs.db

config.js
```

AFFICHER L'HISORIQUE GIT

On peut afficher la liste des commit avec la commande
$ git log

RESOURCES

doc officiel git:
https://git-scm.com/book/fr/v1/

liste commande base:
https://github.com/quinnliu/gitCommands/blob/master/README.md

Model git-flow:
https://www.synbioz.com/blog/git-adopter-un-modele-de-versionnement-efficace

git... no deep shit:
http://rogerdudler.github.io/git-guide/index.fr.html

FIN ;-)