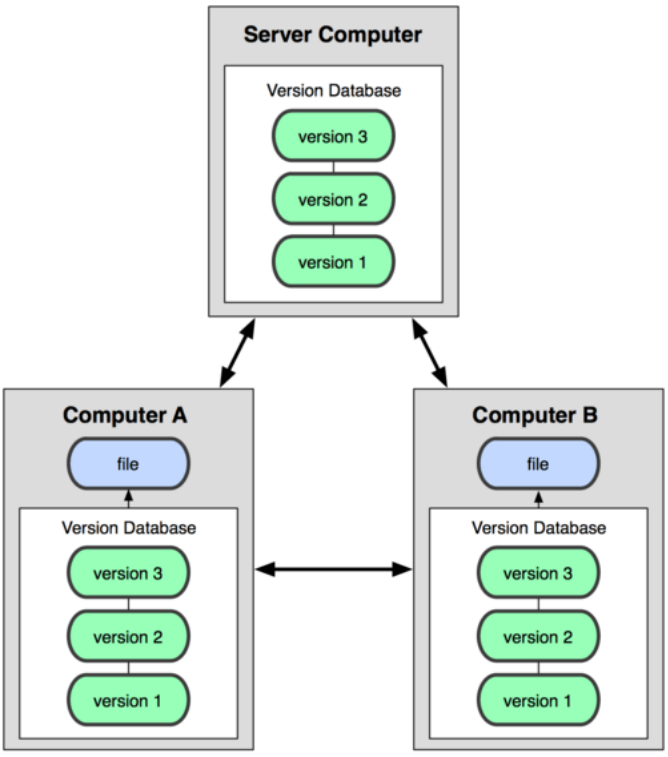
Spatial History

Doing DH Institute

Lincoln Mullen

Don't be afraid of telling lies; be afraid of failing to communicate the truth

Décentralisé



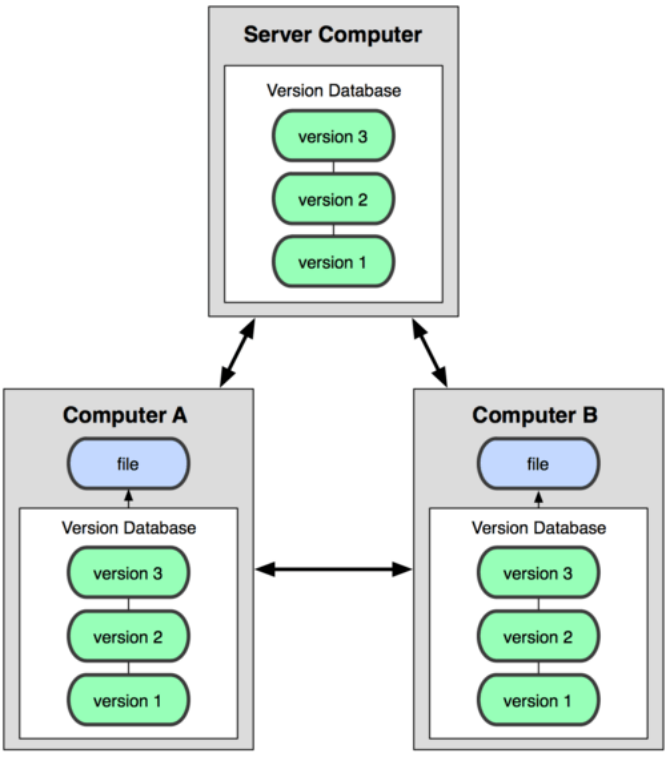
presentation de Git

Concepts de base

- Décentralisé
- Rapide
- Flexible

Note: Inspiré par BitKeeper et créé en 2005 pour le remplacer après que la gratuité de ce dernier soit révoqué.

Décentralisé



Avantages

- Chaque `clone` est un `fork`
- Chaque `clone` est un backup
- Possibilité de versionner hors ligne
- **Très** rapide

Inconvénients

- Pas très performant pour les gros fichiers binaires

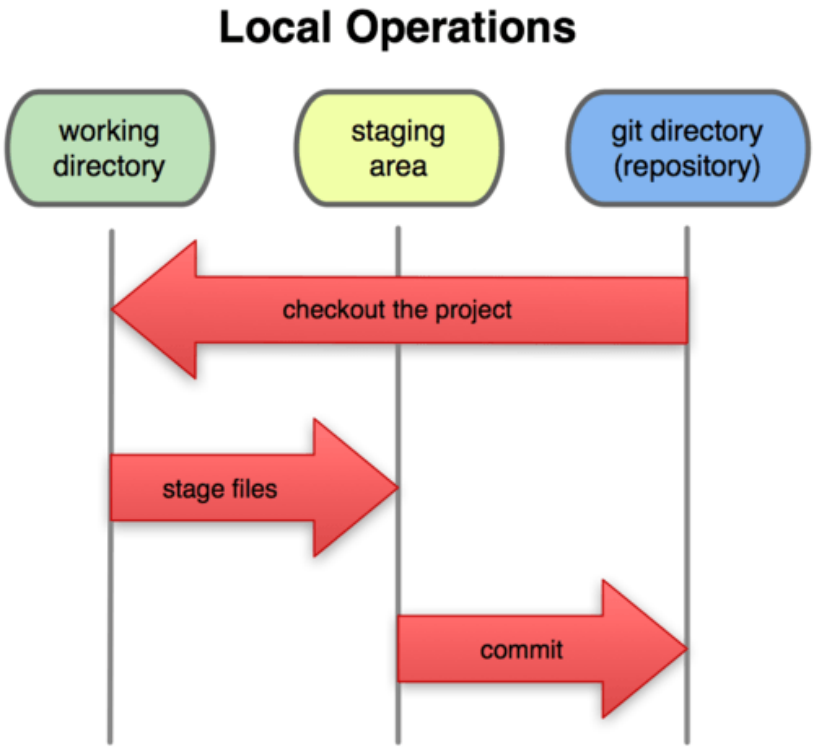
Note: Chaque clone est un backup : a condition de maintenir a jour toutes les branches.

Objectif de git

- Rapidité
- Simplicité
- Prise en charge de milliers de branches parallèles
- Architecture distribuée
- Capable de gerer efficacement de gros projet *(ex:Le noyau Linux)*
- Garantir l'intégrité

Note: + Git est rapide car presque toutes les opérations sont locale. + Git gère l'intégrité

Les trois états



- Validé signifie que les données sont stockées en sécurité dans votre base de données locale.
- Modifié signifie que vous avez modifié le fichier mais qu'il n'a pas encore été validé en base.
- Indexé signifie que vous avez marqué un fichier modifié dans sa version actuelle pour qu'il fasse partie du prochain instantané du projet.

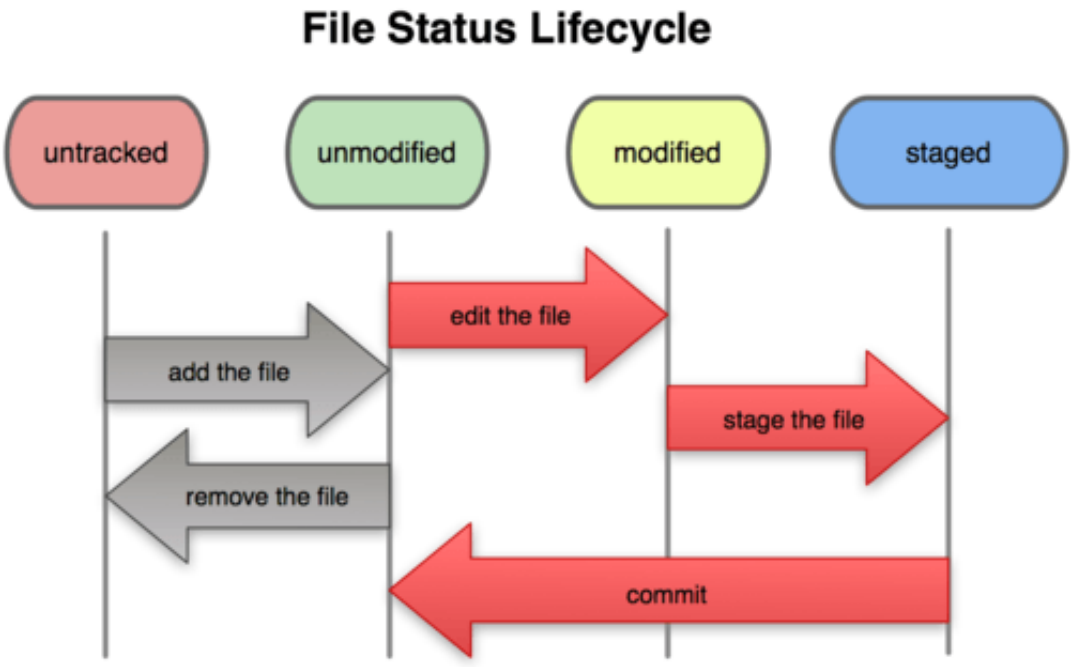
Ceci nous mène aux trois sections principales d'un projet Git : le répertoire Git, le

Le dossier git

Note: Le répertoire Git est l'endroit où Git stocke les méta-données et la base de données des objets de votre projet. C'est la partie la plus importante de Git, et c'est ce qui est copié lorsque vous clonez un dépôt depuis un autre ordinateur.

Le répertoire de travail est une extraction unique d'une version du projet. Ces fichiers sont extraits depuis la base de données compressée dans le répertoire Git et placés sur

Utilisation standard



Installer git

Figure 5: one does not simply install git on windows

Officiel

http://git-scm.com/downloads

Note: Binaire, installeur et source pour Windows, OS X, Linux, Solaris

Windows

http://msysgit.github.io/

LINUX

- Debian: `aptitude install git`
- Ubuntu: `sudo apt-get install git`
- ArchLinux: `pacman -S git`

Interface graphique ?

http://git-scm.com/downloads/guis

Note: Des interface graphique existe... Vous pouvez les trouver la. À vous de choisir celle qui vous convient. Recommandé :

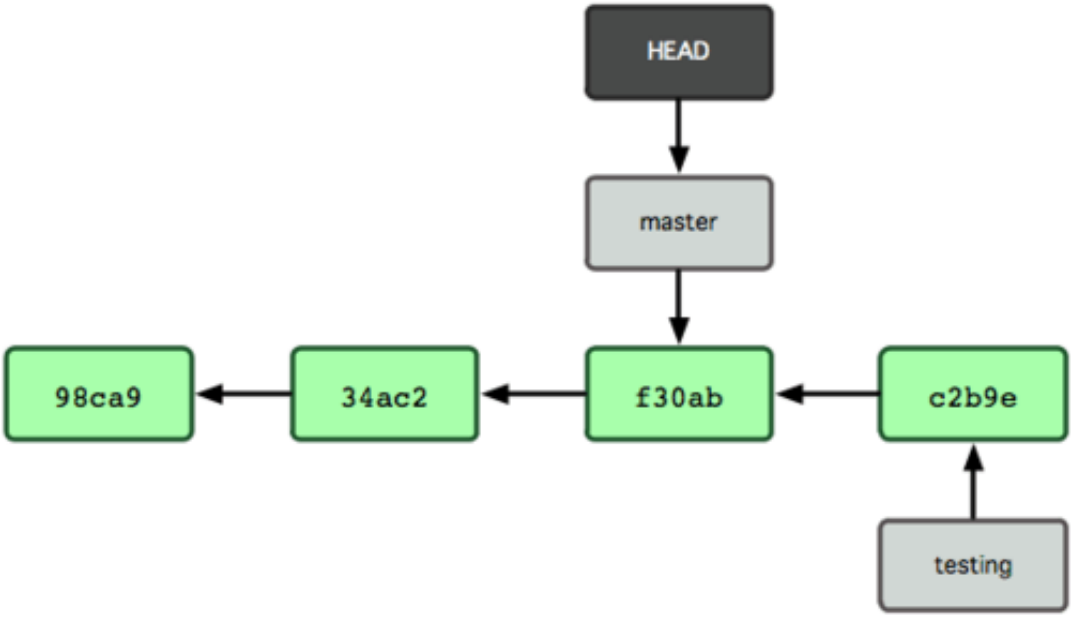
- SmartGit (crossplatform, gratuit pour usage personnel)
- SourceTree (sous mac, le plus populaire)
- GitHub for Mac / Windows (attention : spécifique à GitHub)

Configuration

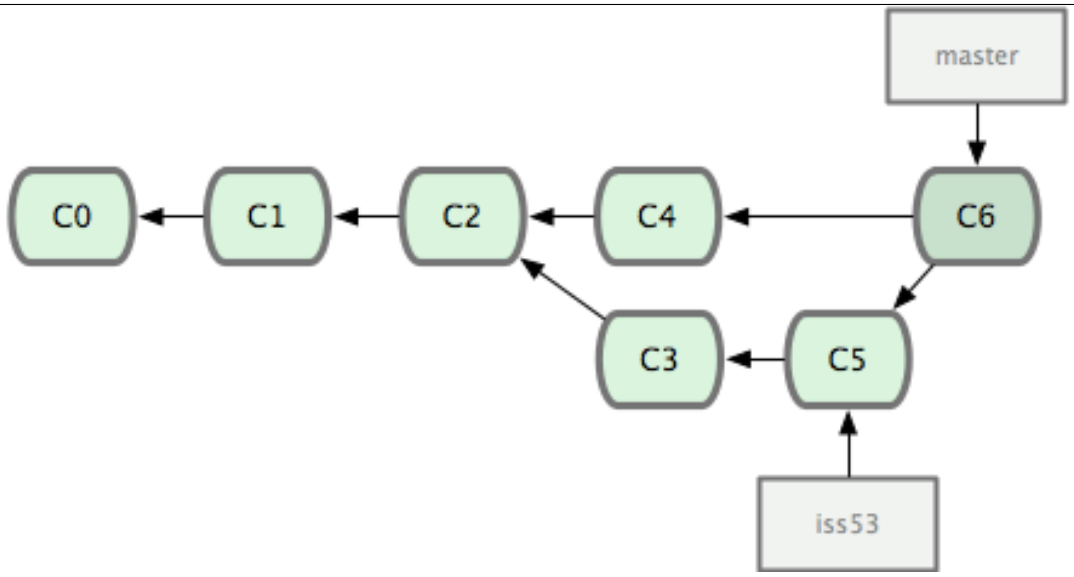
http://git-scm.com/book/fr/Personnalisation-de-Git-Configuration-de-Git

Branches

Les branches sont des “pointeurs” vers des commits.

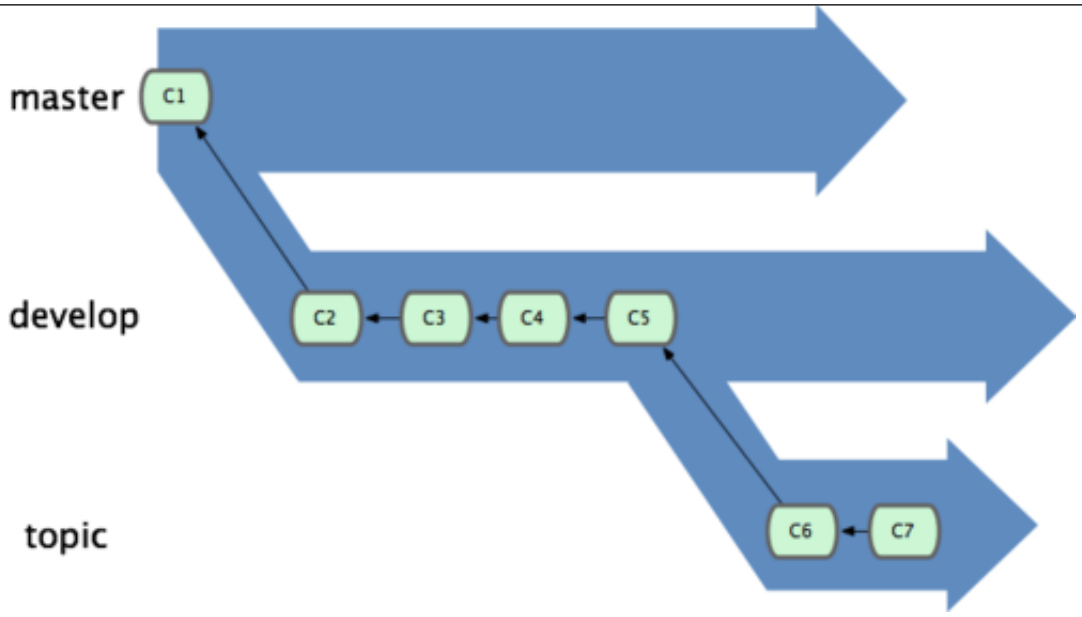


Fusionner des branches



En général ça marche sans problème

Workflow courrant



Et maintenant ?

Resources

Très (très) inspirée de
`https://github.com/p-j/isep-git/blob/master/data/slides.md`

- "Pro Git" Book
- Git reference
- Github git challenges
- Cheat sheet
- Git Visual Cheat sheet

Questions ?

Master comme branche stable

- On ne commit dans master que du code stable
- On commit le code instable dans une branche de développement
- On déploie des patchs sur master
- On merge les patchs dans la branche de développement

Master comme branche de développement

- On commit le code instable dans master
- On créer des tags pour marquer les étapes stables
- On créer une branche à partir d'un tag pour les patchs
- On merge les patchs dans master

Compléments

Commandes de base

L'aide dans git

```
$ git --help
$ git add --help
$ git <sub-command> --help
```

Créer un dépôt local

```
$ git init
```

Cloner un dépôt existant

To be continued...

.gitignore

```
$ cat .gitignore
.DS_Store
.svn
log/*.log
tmp/**
node_modules/
```

- Les lignes vides ou démarrant par un # sont ignorées
- Il est possible d'utiliser des jokers

Remotes

- Autre clones du même dépôt
- Peut être local (un autre `checkout`) ou distant (collègue, serveur central)
- On peut configurer une valeur par défaut pour `push` et `pull`

```
$ git remote -v
origin  git@github.com:p-j/isep-fsnp.g
origin  git@github.com:p-j/isep-fsnp.g
```

Push to remote

Stashing

```
$ git stash
$ git stash pop
$ git stash list
$ git stash apply
$ git stash drop
$ git stash clear
```

Note: Utilisez git stash quand vous voulez enregistrer l'état actuel du répertoire de travail et de l'index, et revenir à un répertoire de travail propre. La commande enregistre

Couleur

```
$ git config --global color.ui true
```