Introduction à Git

Introduction

Qu’est-ce qu’un gestionnaire de versions (VCS) ?

Introduction

Qu’est-ce qu’un gestionnaire de versions (VCS) ?

➜ Conserver des versions de fichiers et leurs arborescences

➜ Identifier une arborescence de versions de fichiers

➜ Fournir les outils de gestion associés

Introduction

Qu’est-ce que Git ?

Introduction

Qu’est-ce que Git ?

Git est un gestionnaire de versions décentralisé

Introduction

Un gestionnaire de versions décentralisé (DCVS) permet :

➜ Calculer les différences entre deux versions (diff / patch)

➜ Un gestionnaire d’historiques des diff

➜ D’être utilisé offline ou sur de multiples serveurs

Git

Créé par Linus Torvalds pour versionner le noyau Linux

Devenu le VCS le plus populaire

Utilisé par toutes les plateformes (GitHub, Bitbucket, ...)

Fonctionnalités

Présentation

Il existe 3 versions d’un même fichier :

- local
- commit
- remote

Présentation

Git stocke les versions dans le répertoire .git

Un fichier modifié sera commité dans .git et push sur un serveur distant

Il existe donc deux étapes pour partager du code

init

La commande init permet de rendre un répertoire versionnable.

```
git init
```

Créer le répertoire .git dans lequel les fichiers sont versionnés.

clone

La commande clone permet de récupérer l’ensemble d’un dépôt distant

```
git clone <server> <folder>
```

Toutes les modifications sur le serveur <server> sont placées dans le répertoire <folder>

add

Lors de la création d’un nouveau fichier, il faut l’ajouter dans git.

```
git add monfichier.java
```

```
git add -A
```

Dès lors le fichier est dans l’état “stage” et peut être commité.

commit

Une fois le fichier référencé dans Git, un commit permet de stocker l’ensemble des modifications et créer une version unique dans le répertoire .git.

```
git commit -m “Mon message de commit”
```

Le paramètre -m spécifie le message de commit et est obligatoire

push

Afin d’envoyer un ensemble de commit vers un serveur distant, il faut push les commits.

```
git push <server> <branch>
```

```
git push origin master
```

fetch

Récupérer des modifications distants vers .git

```
git fetch <server> <branch>
```

```
git fetch origin master
```

Les commits distants sont récupérés mais ne sont pas appliqués sur le code.

merge

Un merge effectue une mise à jour et une fusion.

```
git merge origin/master
```

```
git merge master
```

Applique l’ensemble des commits à votre code

pull

Afin de faciliter la récupération de commit, la commande pull effectue un fetch / merge

```
git pull origin master
```

=

```
git fetch origin master;git merge origin/master
```

remote

Chaque serveur distant est identifié par un nom unique.

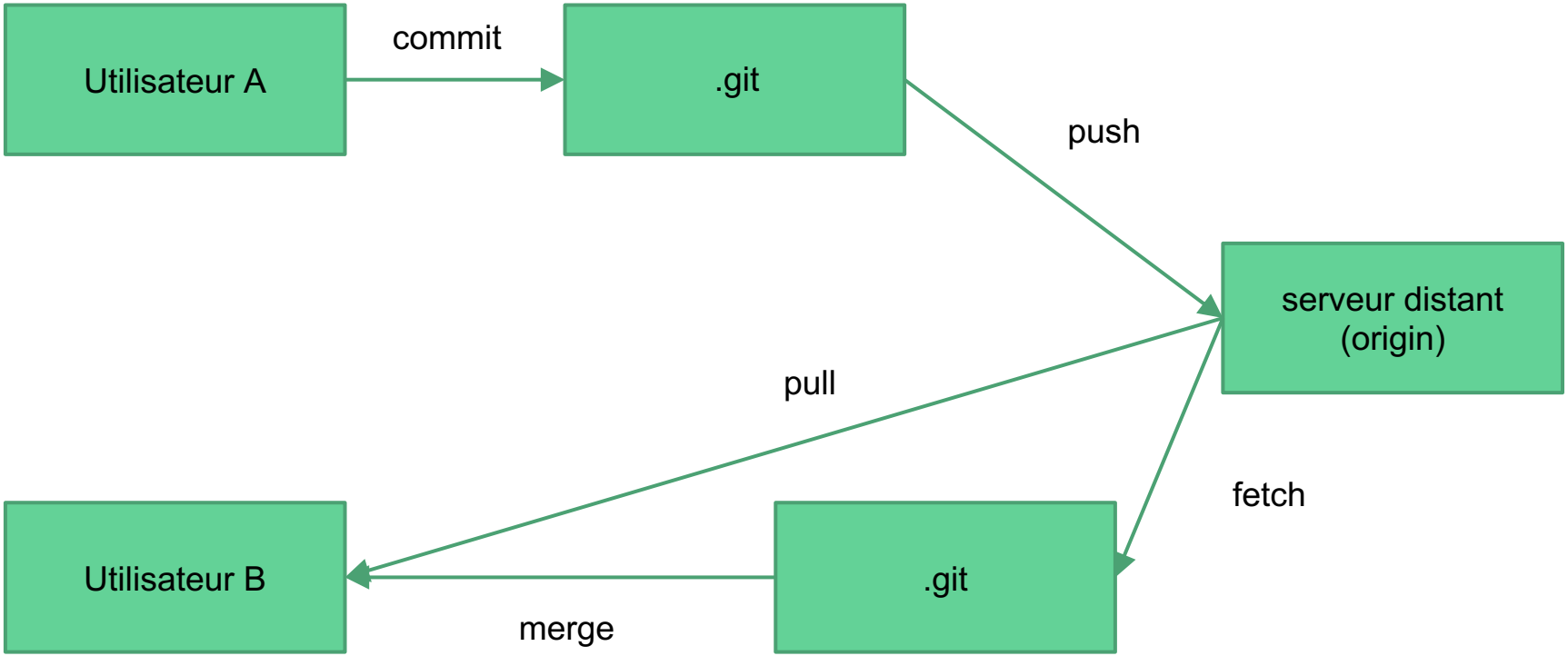
```
git remote add <name> <server>
```

```
git remote add github https://github.com/vert-x3/vertx-web
```

Le serveur par défaut lors d’un clone est nommé origin

Lors d’un push vous choisissez le serveur sur lequel vous voulez push

Vue globale


Utilisateur A
commit
.git
push
serveur distant
(origin)
pull
fetch
Utilisateur B
merge
.git

Branches

Une branche est une sous-partie du graphe des commits.

Un développement précis = une branche

Les branches peuvent être fusionnées entre elles

Branches

La branche par défaut s’appelle master.

`git checkout -b <branch>` ➜ Créer une nouvelle branche

`git checkout <branch>` ➜ Se déplacer vers une branche

Branches

Fusionner des branches se résument à donner le nom de la branche à merger.

```
git merge sprint25
```

Applique les commits de sprint25 dans la branche courante.

Workflow

Créer son environnement

```
echo “bonjour” > README.md

git init

git add -A

git commit -m “Add README.md”

git remote add origin http://github.com/olivier-pitton/dant.git

git push origin master
```

Développement standard

```
git pull origin master

git add -A

git commit -m “Mes modifications”

git fetch origin master

git merge origin/master

git commit -m “Merge”

git push origin master
```

Fusion de branches

```
git pull origin master

git checkout -b “dev”

git add -A;git commit -m “Mes modifications”

git fetch origin master;git merge origin/master

git commit -m “Merge”

git push origin dev
```

Commandes avancées

Conserver les modifications en cours

Lorsque vous avez des modifications en cours et que vous devez changer de branche.

Plutôt que de faire un commit, que pouvez faire ?

Squash

La commande squash permet de conserver dans une pile les modifications actuelles, pour les retrouver plus tard

```
git squash
```

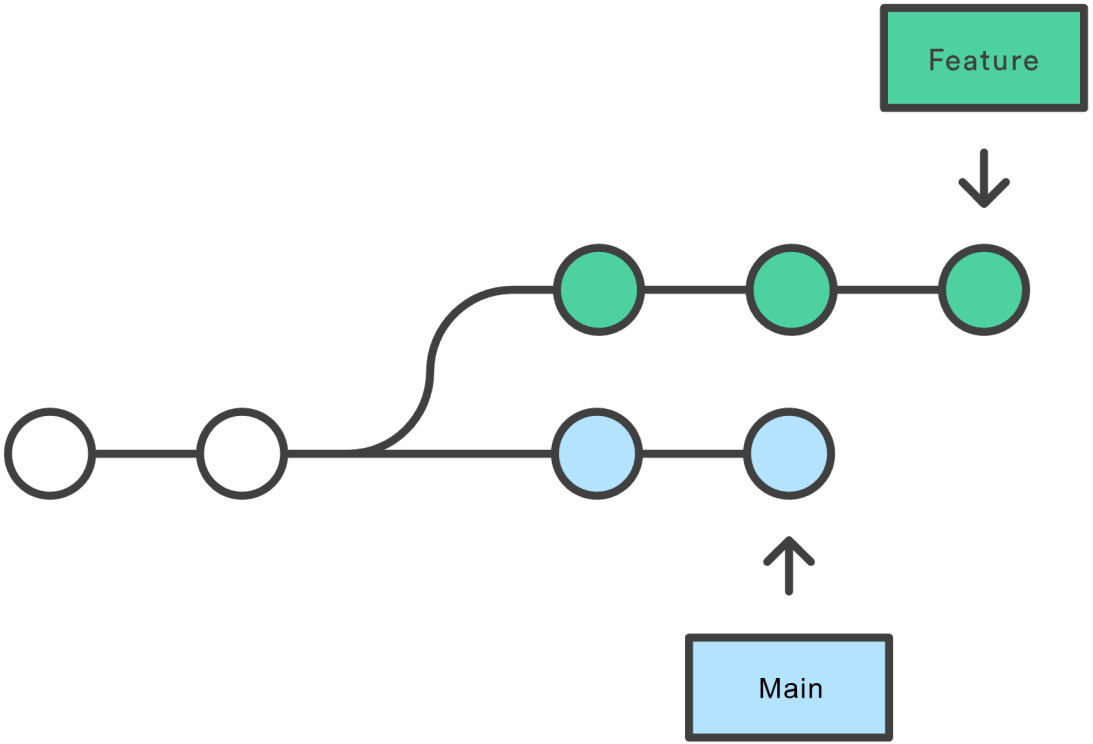
```
git squash apply
```

Différents types de fusion

Il existe 3 manières de fusionner son code :

- Merge commit
- Merge squash
- Rebase

A forked commit history



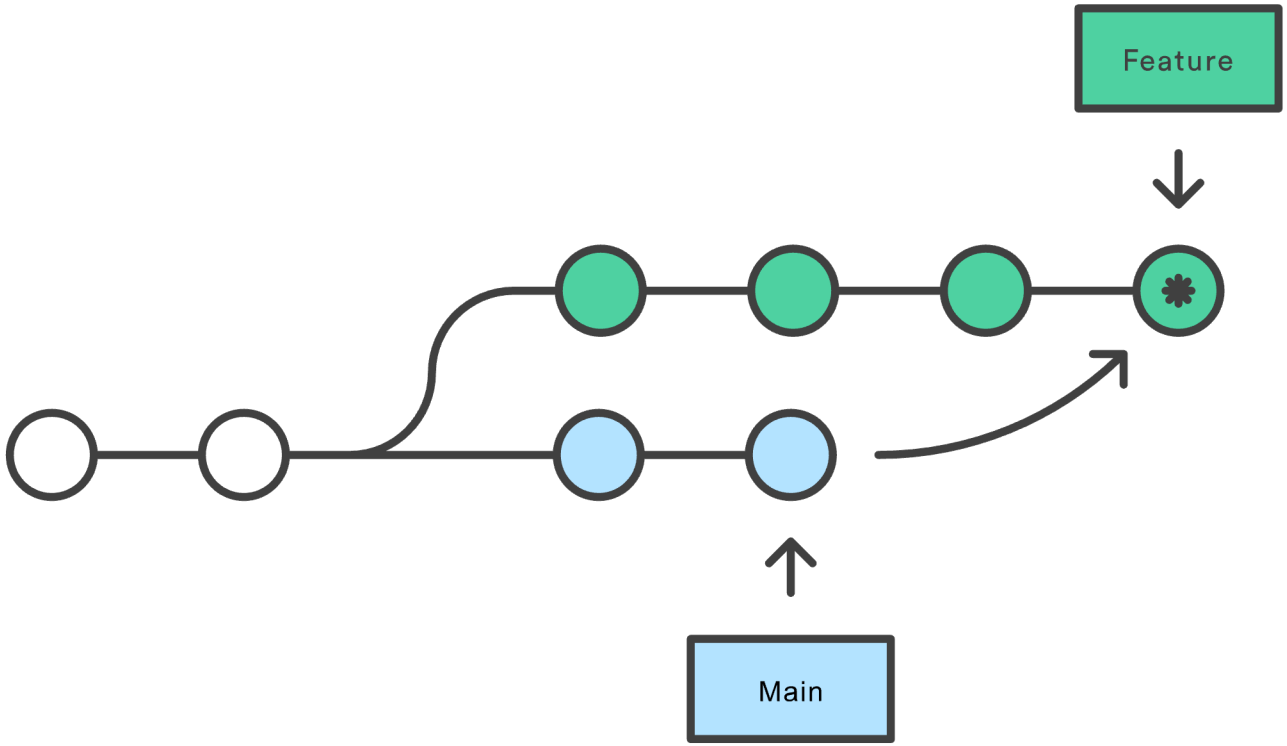
Merge / commit

Un nouveau commit est créé entre les deux branches

Tous les commits intermédiaires sont conservés

Option par défaut dans Github / Bitbucket

Merging main into the feature branch



Merge Commit

Merge / squash

Un nouveau commit est créé entre les deux branches

Tous les commits intermédiaires sont fusionnés en 1 seul commit

Pratique pour enlever tous les commits poubelles (fix, fix test ...) et définir un commit final propre

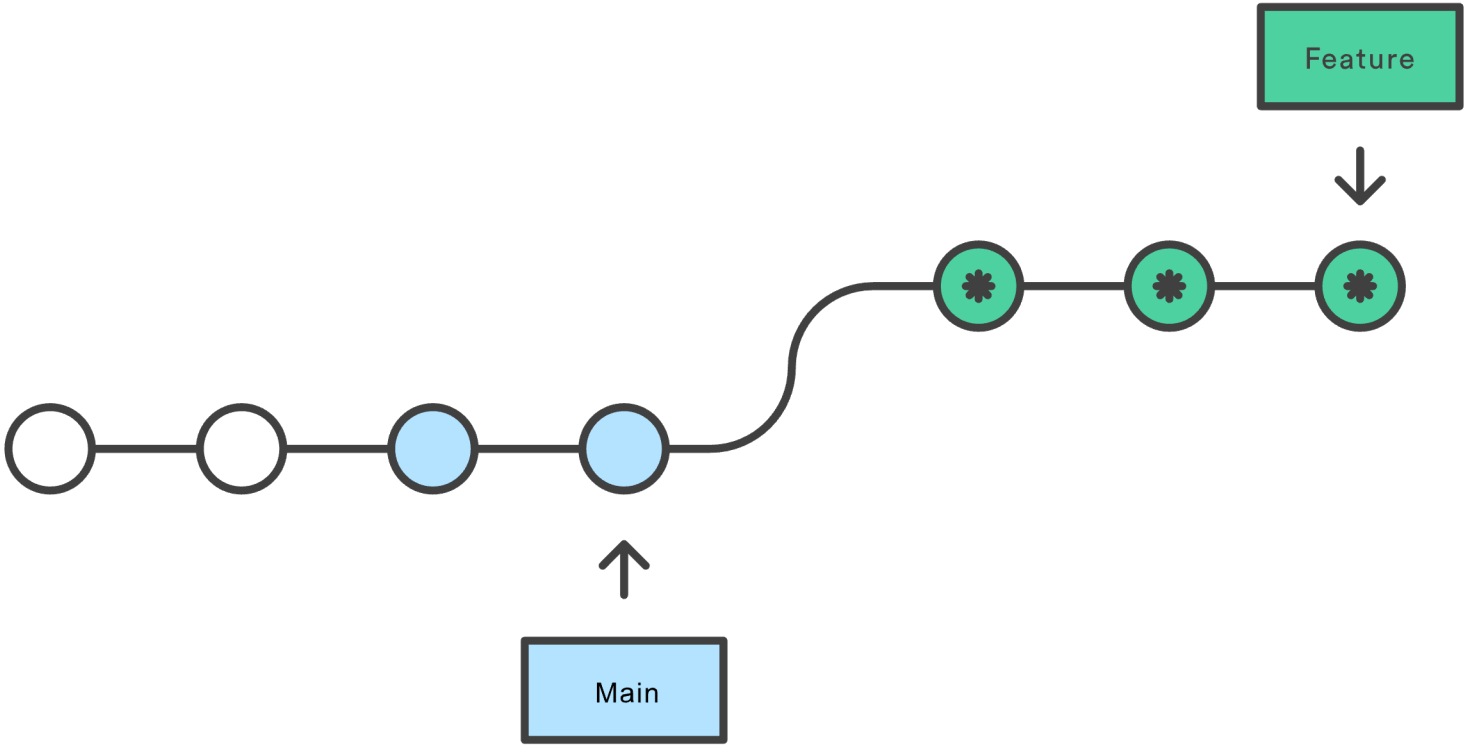
Rebase

Toutes les modifications sont intégrées dans la branche

Tous les commits intermédiaires sont conservés (par défaut)

Pratique pour garder un historique propre en une seule ‘ligne’

Rebasing the feature branch onto main



Brand New Commit

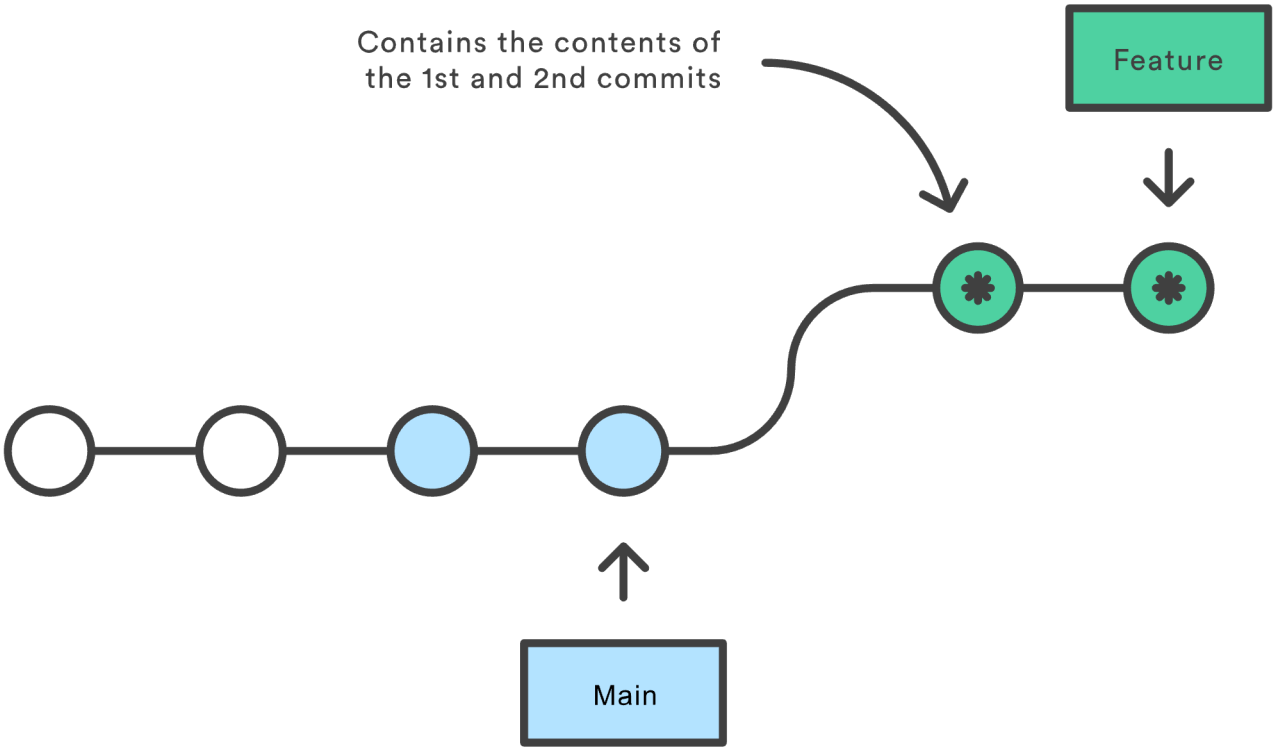
Rebase interactif

Le rebase interactif permet de définir quoi faire de chaque commit

```
git rebase -i HEAD~3
```

Pratique pour préparer un historique propre avant de rebase dans la branche principale

Squashing a commit with an interactive rebase



Brand New Commit

Github

Github

Plateforme sociale de dépôts Git

Gratuit

Fait la promotion des bonnes pratiques de développement

La pull request

Qu’est ce qu’une pull request ?

La pull request

Qu’est ce qu’une pull request ?

Permet de partager et intégrer du code

Valider une PR revient à intégrer le code dans la branche master

La pull request

Comment faire des pull request ?

La pull request

```
git pull origin master
```

```
git checkout -b “dev”
```

```
git add -A
```

```
git commit -m “Mes modifications”
```

```
git push origin dev
```

Dans Github, créer une PR. from : dev - base : master

Conclusion

Git est utilisé dans la majorité des (nouvelles) entreprises

Facilite le développement logiciel (contrairement à SVN)

Obligatoire à utiliser, via GitHub, pour le projet !