git : gestion de version

git : gestion de version

Présentation

Naissance de GIT : 2005

Contexte : Organiser et gérer les différentes mise à jour du noyau Linux

Créateur : Linus Torvalds

Objectifs :

VITESSE SIMPLICITE MULTI-VERSION EFFICACITE

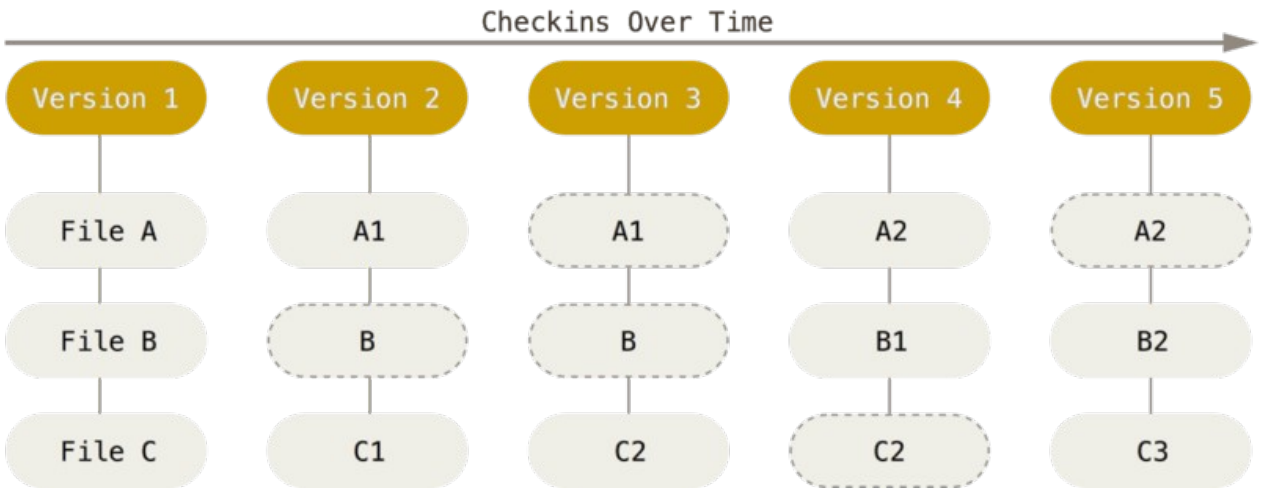
Utilisation sur Linux, Windows, Mac, etc.

git : gestion de version

Principe des versions

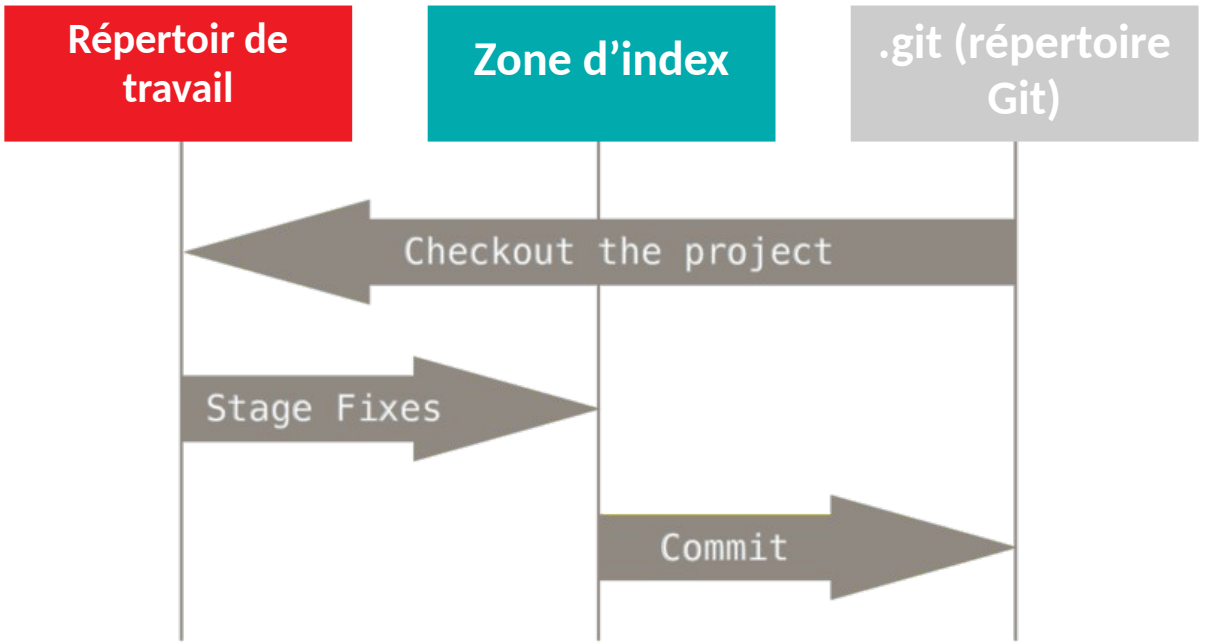
GIT est un logiciel de gestion de versions **décentralisée** :

- Réalisation d'instantanés et non de sauvegardes
- Gestion des versions en local sur chaque poste
- Intégrité des fichiers présevée



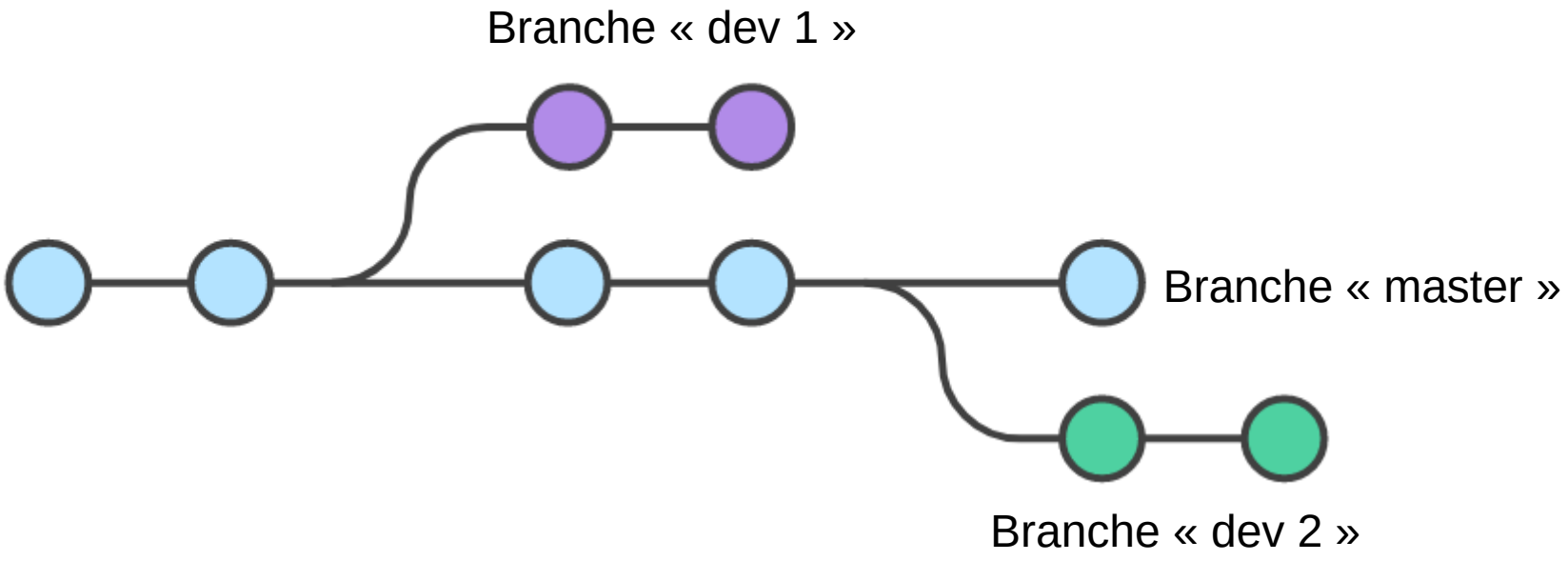
git : gestion de version

Fonctionnement de base



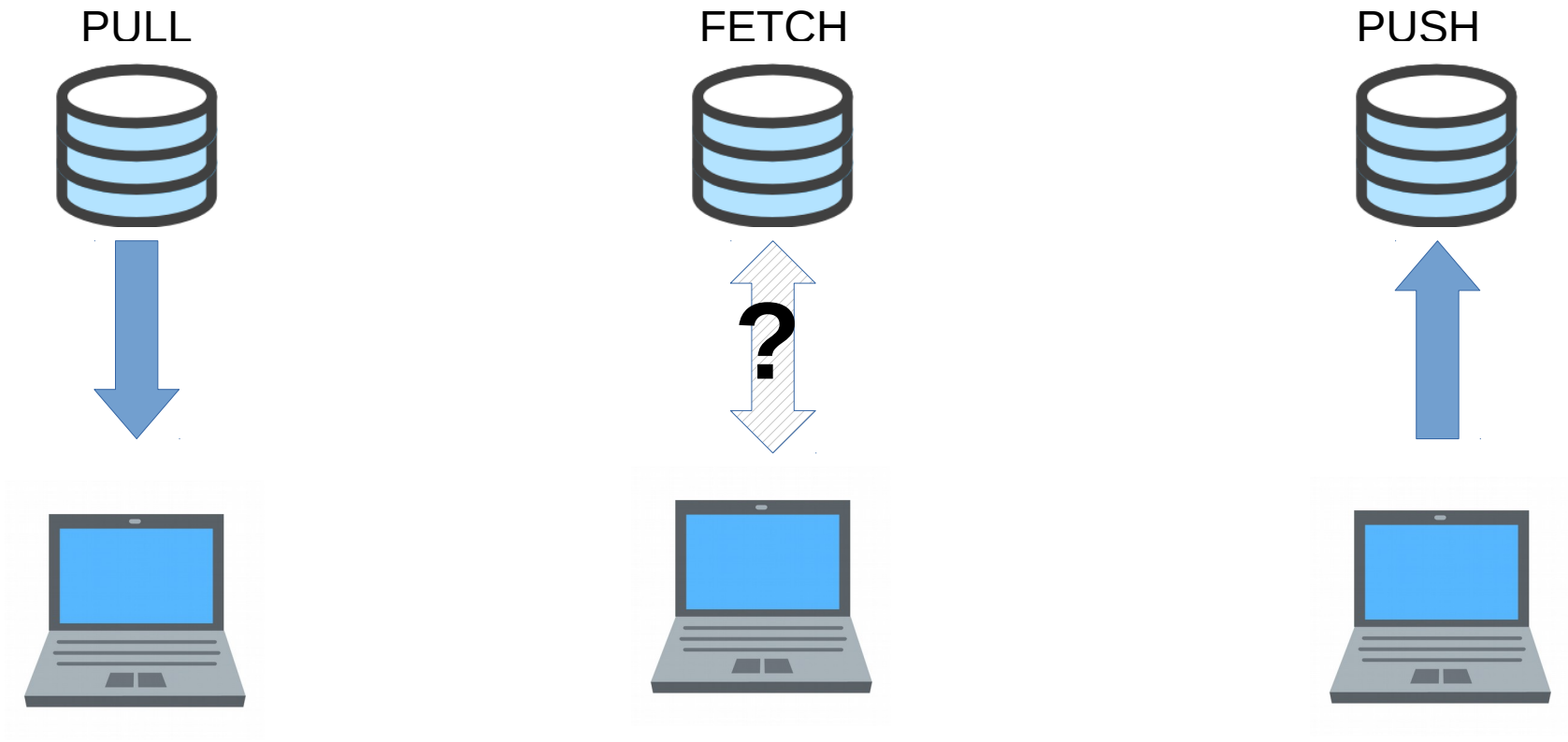
git : gestion de version

Travailler avec les branches



git : gestion de version

Travailler avec des dépôts distants



git : gestion de version

Déroulement d'un projet avec GIT

1) On définit les informations concernant notre profil GIT (attention ! Sur un dépôt distant, le mail et le nom sont obligatoires afin de vous identifier comme auteur).

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

2) Dans notre dossier de développement, on initilialise GIT

```
$ git init
```

git : gestion de version

Déroulement d'un projet avec GIT

3.1) Si le projet existe, on récupère le projet distant depuis un dépôt

```
$ git clone https://github.com/libgit2/libgit2 master
```

3.2) OU / ET on lance les fonctions pour effectuer notre premier instantané

```
$ git status
(permet de voir l'état des fichiers : modifier ou ajouter)

$ git add . OU  git add index.php
(ajouter les fichiers pour le prochain commit)

$ git commit -m « init commit »
Réaliser le commit permet de créer une version
```

git : gestion de version

Déroulement d'un projet avec GIT

4) Je travaille sur la partie back-office de mon projet, en fin de journée

```
$ git status
(permet de voir l'état des fichiers : modifier ou ajouter)

$ git add . OU  git add ./backoffice
(ajouter les fichiers pour le prochain commit)

$ git commit m « modif 1 backoffice »
On réalise un second instantané de notre projet
```

git : gestion de version

Déroulement d'un projet avec GIT

4) Je travaille sur la partie front-office de mon projet, en fin de journée

```
$ git status
(permet de voir l'état des fichiers : modifier ou ajouter)

$ git add . OU  git add ./frontoffice
(ajouter les fichiers pour le prochain commit)

$ git commit m « modif 1 front-office »
On réalise un troisième instantané de notre projet
```

git : gestion de version

Déroulement d'un projet avec GIT

4) Je suis satisfait de ma version dev, je pousse sur le serveur

```
$ git remote add pb https://github.com/paulboone/ticgit
Je défini un alias pour mon url distante

$ git fetch pb
On regarde les différences entre le dépôt local et le dépôt distant

$ git push pb master
On pousse la branche principal sur le serveur
```

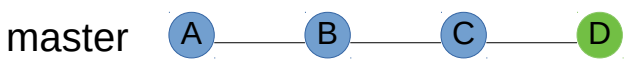
git : gestion de version

Déroulement d'un projet avec GIT

5) Bob rejoins l'aventure, et je sais qu'il travaille aussi sur la branche principale de son dépôt local

```
$ git pull pb

Je charge les dernières mises à jour sur le serveur
```



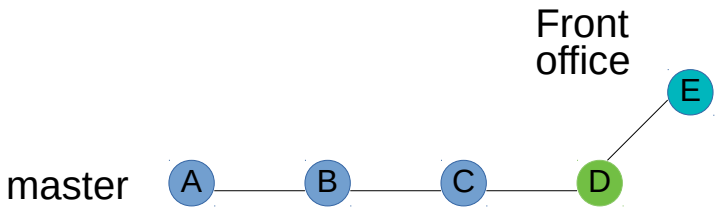
git : gestion de version

Déroulement d'un projet avec GIT

5) Bob travaille sur le back, je travaille sur le front. Cependant je veux quand même garder la branche principale (au cas où Bob ferai une mise à jour sur le serveur).

```
$ git checkout-b front_office master

Je créé la branche front_office, issu de la branche master (donc avec
le même contenu) et me place dedans
```

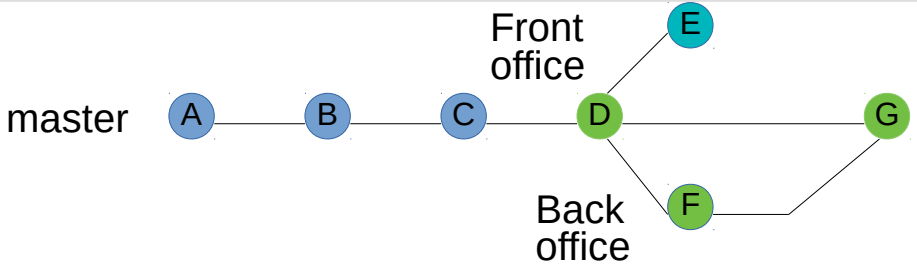


git : gestion de version

Déroulement d'un projet avec GIT

5) Bob a fait une modification et a effectuer un merge sur la branche master. Je souhaite mettre à jour la branche master sans perdre mon travail courant.

```
$ git stash
Je créé une sauvegarde de mon travail afin de changer de branche
$ git checkout master
Je me place dans la branche master
$ git pull pv
Je récupère le travail de bob sur la branche principale
$ git checkout front-office
$ git stash pop
Je récupère les données enregistrées précédemment et les réaplique à E
```



git : gestion de version

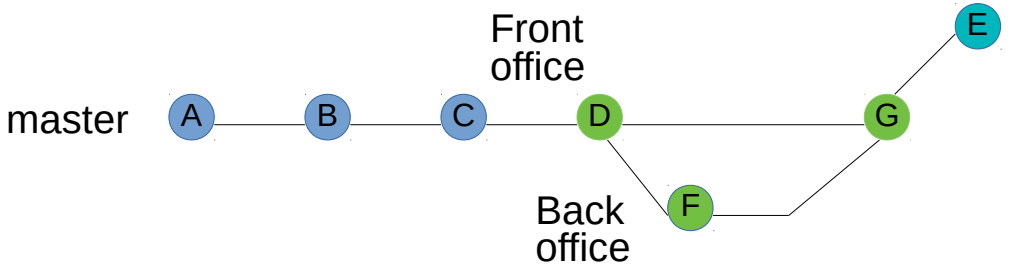
Déroulement d'un projet avec GIT

6) J'ai effectué la mise à jour de la branche principale… Mais la mienne n'est plus à jour. Je repositionne mon travail tout en mettant à jour la branche de dev.

```
$ git stash
Je créé une sauvegarde de mon travail afin de changer de branche

$ git rebase front-office
Je repositionne ma branche sur la dernière mise à jour de la branche
principale

$ git stash pop
Je récupère les données enregistrées précédemment et les réaplique à E
```

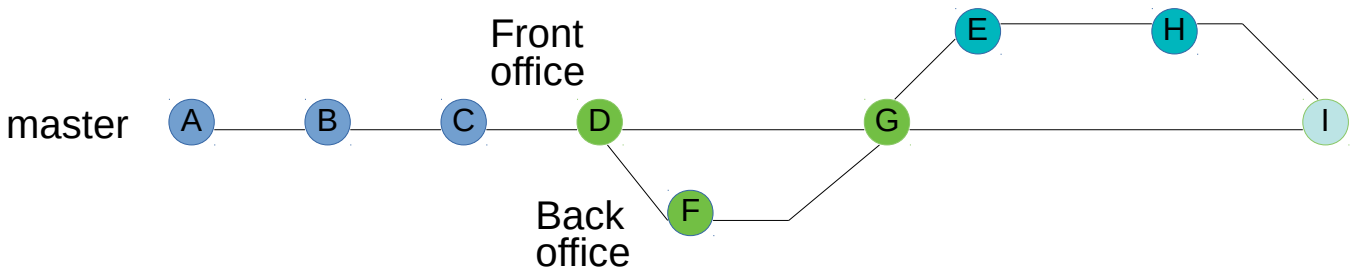


git : gestion de version

Déroulement d'un projet avec GIT

7) J'ai fini mon travail. Afin que bob ai mes mises à jour, je réalise un merge avec la branche master. Puis je pousse sur le serveur distant.

```
$ git commit -m « front office, mise à jour 1 »
Je fais le commit de mes modifications
$ git checkout master
Je me place dans master
$ git merge front_office
Je fusionne le contenu de master avec le contenu de la branche front
$ git push pb
```



git : gestion de version

Déroulement d'un projet avec GIT

8) Bob fait une modification directement sur master sans créer de branche…Mais il fait tout planter ! Vous décidez de sauver le site.

```
$ git pull pv
On récupère le travail de bob, afin de garder une trace

$ git revert J
La commande revert va effectuer le commit inverse (donc « annuler » le dernier commit).
```