Gestion de versions

Patrick MARCEL, Université d'Orléans

L2 Outils du développeur — S3

Une petite histoire de programmeur...

- Depuis 6 mois, Pierre travaille sur un logiciel de statistiques
- La semaine dernière dernière, il a modifié le code d'une fonction complexe
- Il vient de s'apercevoir que c'était une erreur car elle ne fonctionne plus
- il aimerait revenir en arrière...

Une deuxième histoire de programmeur...

- Pierre ne s'y laissera plus prendre
- Chaque soir il fait un zip de son projet
- Etienne passe le voir :
  - *Tu te souviens que tu as changé l'interface d'affichage des probabilités ?*
  - Heu... oui.. ça remonte à quelques semaines...
  - *Tu pourrais me retrouver la dernière version avant l'ajout des couleurs ?*
  - Je vais chercher...
- et c'est parti pour une longue séance de fouilles archéologiques...

Une troisième histoire de programmeur...

- Pierre et Etienne ne se sont pas vu depuis deux semaines
- Ce soir, Etienne vient voir Pierre avec une clé USB
  - *J'ai trouvé un bug dans la lecture de fichier de données, je l'ai corrigé, voilà le nouveau code*
  - Maiiiiis... j'ai ajouté la gestion des fichiers CSV entre temps...
  - *Ah... mais t'as pas modifié le fichier readMyFile.java ?*
  - Ben si, justement
  - *Bon. Fais voir ce que tu as changé*
- et c'est reparti pour une longue soirée !

Une variante...

- La semaine dernière, Pierre a demandé à Etienne de travailler sur le calcul de la distribution de Poisson
- Pierre, lui, n'a pas touché au code depuis
- Ce matin, il souhaite reprendre le développement, mais... il ne reconnait plus le code !
  - *Etienne, y a eu une panne ?*
  - Non, je ne crois pas, j'ai même pu travailler sur le code...
  - *Mais... tu as aussi changé les classes sur lesquelles je travaillais ? !*
  - Ha ben oui, sinon, ma distribution n'était pas bonne :-D
  - *Le problème, c'est que maintenant je ne retrouve plus le bug sur lequel je travaillais.*
  - Ah, heu peut-être est-il corrigé ?...
  - *Croisons les doigts...*

Une histoire courte de programmeur...

- Enfer et damnation, mon dur a crashé !
- C'est pas grave, tu as plein de zips de ton code, tu en fais un chaque soir.
- Oui, mais ils étaient tous sur mon disque dur :-o...

Conclusion ?

Le début d'une autre histoire...

- Voilà, vous allez faire un nouveau projet à 25. A vous de vous organiser...

Problématique

Les **systèmes de gestion de versions** visent à :

- permettre un partage efficace du code source entre **plusieurs développeurs** ;
- **conserver l'historique** des changements apportés au code tout en permettant de les **annuler** ;
- **résoudre les conflits** de mise en commun des modifications.

L'utilité de ces outils dépasse le simple cadre du développement collectif de logiciels de grosse taille.

Au début était diff. . .

La commande `diff` compare deux fichiers ligne à ligne.

```
$ diff hello.c bye.c
0a1,2
> #include <stdio.h>
>
3c5
<     printf("Hello world!\n");
---
>     printf("Goodbye love!\n");
```

Elle existe en plusieurs parfums : unifié (`-u`), contextuel (`-c`).

Au début était diff. . .

La commande `diff` compare deux fichiers ligne à ligne.

```
$ diff -c hello.c bye.c
*** hello.c 2012-09-27 13:21:34.000000000 +0200
--- bye.c   2012-09-27 16:28:11.000000000 +0200
***************
*** 1,4 ****
  void main()
  {
!     printf("Hello world!\n");
  }
--- 1,6 ----
+ #include <stdio.h>
+
  void main()
  {
!     printf("Goodbye love!\n");
  }
```

Elle existe en plusieurs parfums : unifié (`-u`), contextuel (`-c`).

Au début était diff. . .

La commande `diff` compare deux fichiers ligne à ligne.

```
$ diff -u hello.c bye.c
--- hello.c     2012-09-27 13:21:34.000000000 +0200
+++ bye.c       2012-09-27 16:28:11.000000000 +0200
@@ -1,4 +1,6 @@
+#include <stdio.h>
+
 void main()
 {
-    printf("Hello world!\n");
+    printf("Goodbye love!\n");
 }
```

Elle existe en plusieurs parfums : unifié (`-u`), contextuel (`-c`).

. . . et son inséparable patch

La commande `patch` applique une collection engendrée par `diff`.

Devine seule le type de patch présenté (contextuel, unifié, etc).

Robuste aux divergences dans les numéros de ligne.

L'option `-p` permet de supprimer des répertoires en préfixe.

Pour les cas difficile il y avait diff3 !

La commande `diff3` permet de fusionner deux fichiers dont on connaît un ancêtre commun.

```
% diff3 -m helloA.c helloO.c helloB.c
<<<<<<< helloA.c
#include <stdio.h>

void main()
||||||| helloO.c
void main()
=======
int main(int argc, char *argv[])
>>>>>>> helloB.c
{
<<<<<<< helloA.c
    printf("Goodbye love!\n");
||||||| helloO.c
    printf("Hello world!\n");
=======
    printf("Hello world!\n");
    return 0;
>>>>>>> helloB.c
```

Système de gestion de versions

Un **système de gestion de versions** est constitué d'une base de donnée, appelée **dépôt**, qui stocke les versions successives de chacun des fichiers sous son contrôle.

Chaque utilisateur dispose d'une **copie locale** d'une des versions de l'ensemble des fichiers sur laquelle il peut travailler.

L'utilisateur peut **soumettre** ses modifications pour qu'elles soient ajoutées au dépôt en tant que nouvelle version.

Le système apporte des mécanisme pour prendre en compte les **conflits de modification**.

Aux temps anciens

Les premiers systèmes de gestion de versions (SCCS, RCS) permettaient uniquement une gestion de version sur le système de fichier local.

Le dépôt était simplement constitué de la version originelle de chaque fichier et de la suite des patchs correspondant aux versions successives.

Systèmes centralisés

Le dépôt est situé sur un **serveur central commun**.

L'utilisateur ne dispose essentiellement que de sa copie de travail.

CVS en est le représentant historique.

Subversion est une solution moderne très populaire.

Systèmes distribués

L'incarnation moderne des systèmes de gestion de versions est **distribuée**.

Chaque utilisateur dispose de sa copie complète du dépôt.

N'importe quels utilisateurs consentants peuvent échanger les modifications et nouvelles versions de leurs dépôts.

Git développé par Linus Torvalds pour le noyau Linux est très populaire.

Mercurial est une alternative libre aussi très répandue.

Git

Site officiel : `http://git-scm.com/`

Documentation **Git Book** : `http://git-scm.com/book/fr`

Documentation **A Visual Git Reference** :

`http://marklodato.github.io/visual-git-guide/`

*Les figures suivantes sont de Mark Lodato, extraites de **A Visual Git Reference**, sous licence Creative Commons BY-NC-SA-3.0.*

Que souhaitons-nous faire ?

- avoir un endroit (dépôt) pour déposer les versions de notre code
  $\rightarrow$ créer un nouveau dépôt, ou
  $\rightarrow$ se connecter à un dépôt existant
- sauver des versions du code
- retrouver d'ancienne versions
- travailler à plusieurs versions sans qu'elles interfèrent
- les fusionner le moment venu
- fusionner le travail de plusieurs personnes

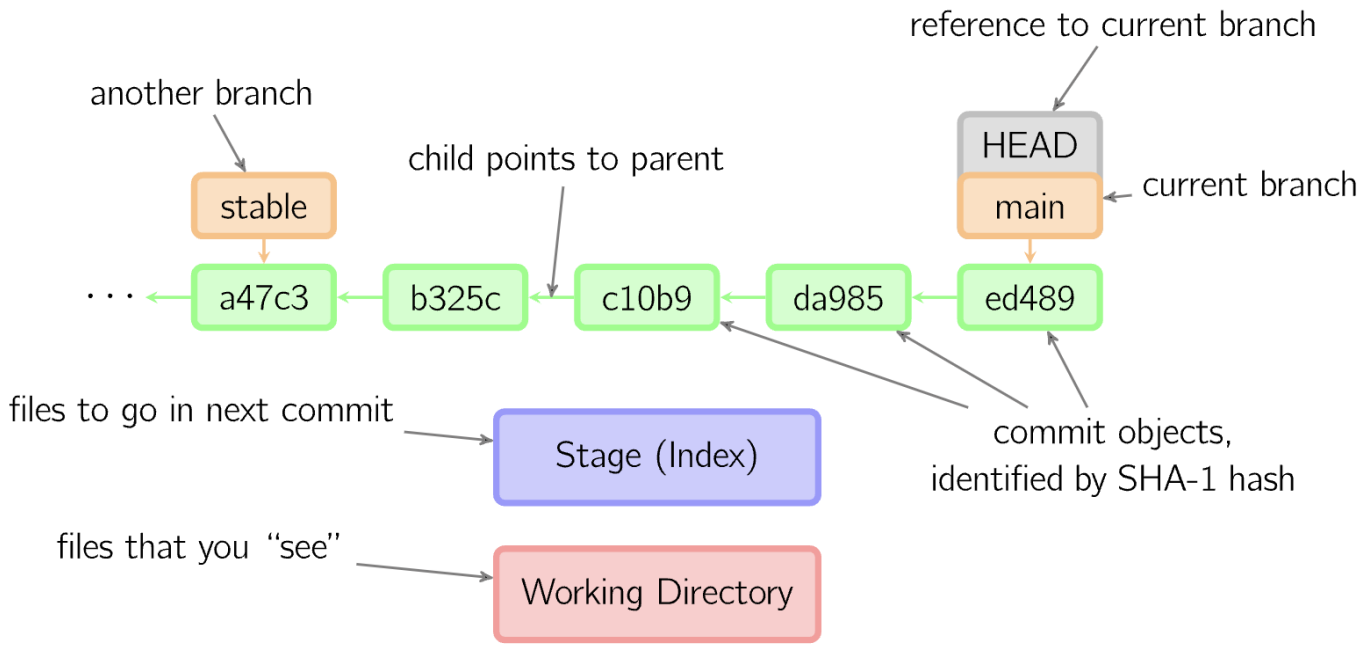
Création d'un dépôt

Un dépôt vide se crée dans un répertoire avec `git init`

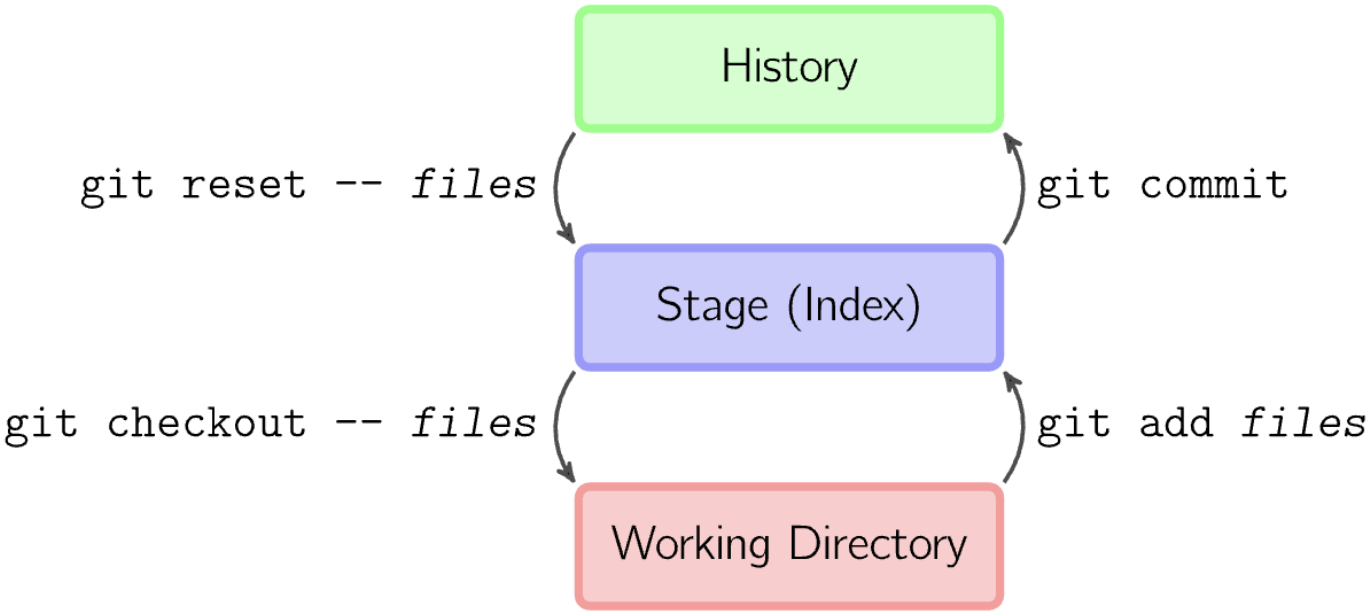
Un dépôt se clone avec la commande `git clone` :

```
$ git clone https://github.com/nopid/atelier.git
```

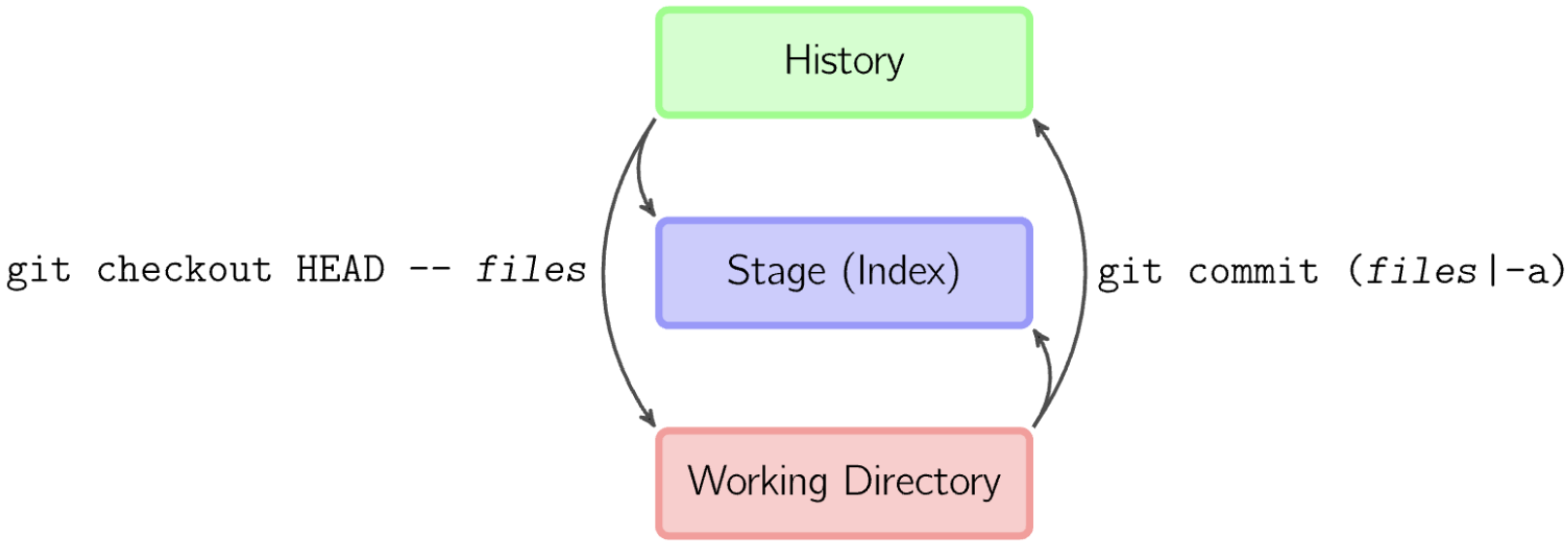
Conventions



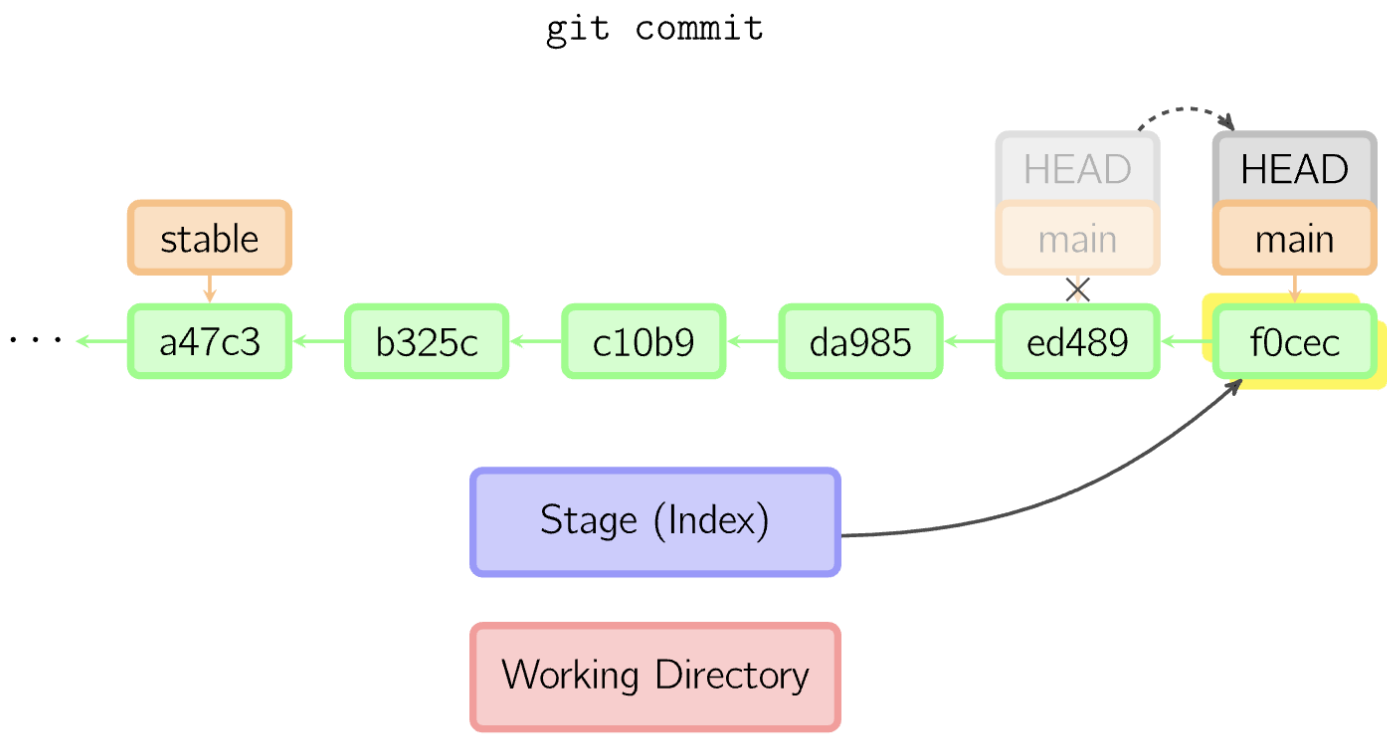
Premier contact (1/2)



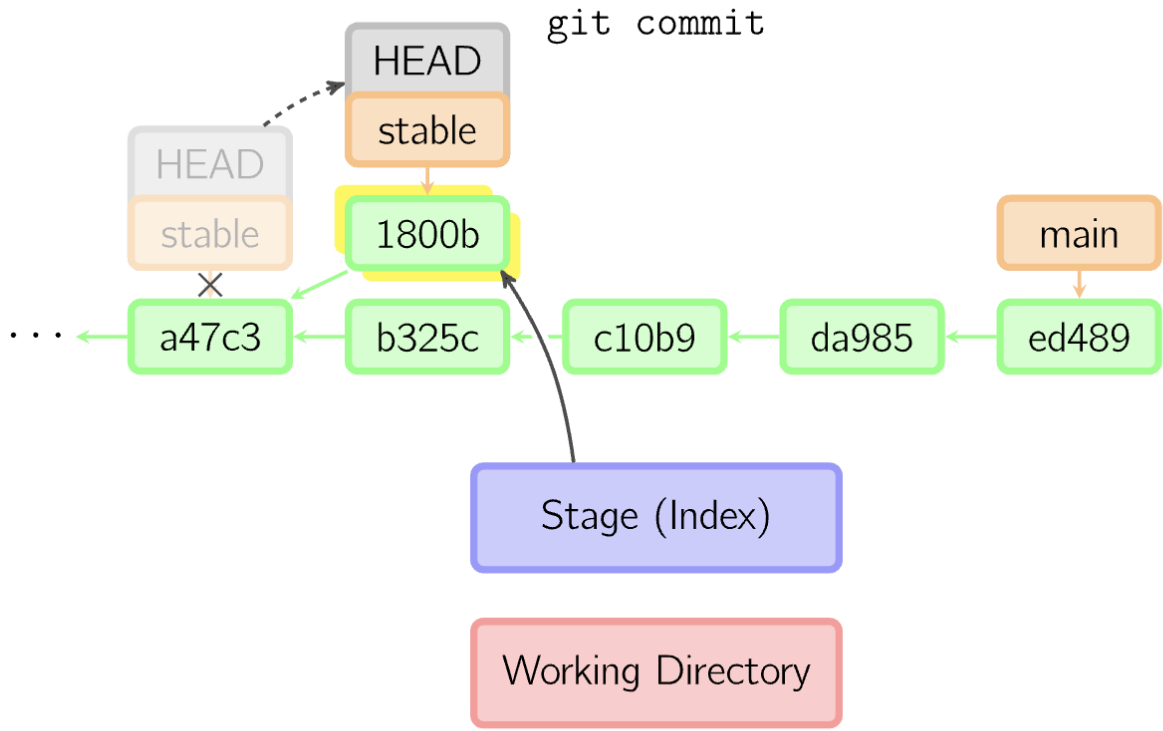
Premier contact (2/2)



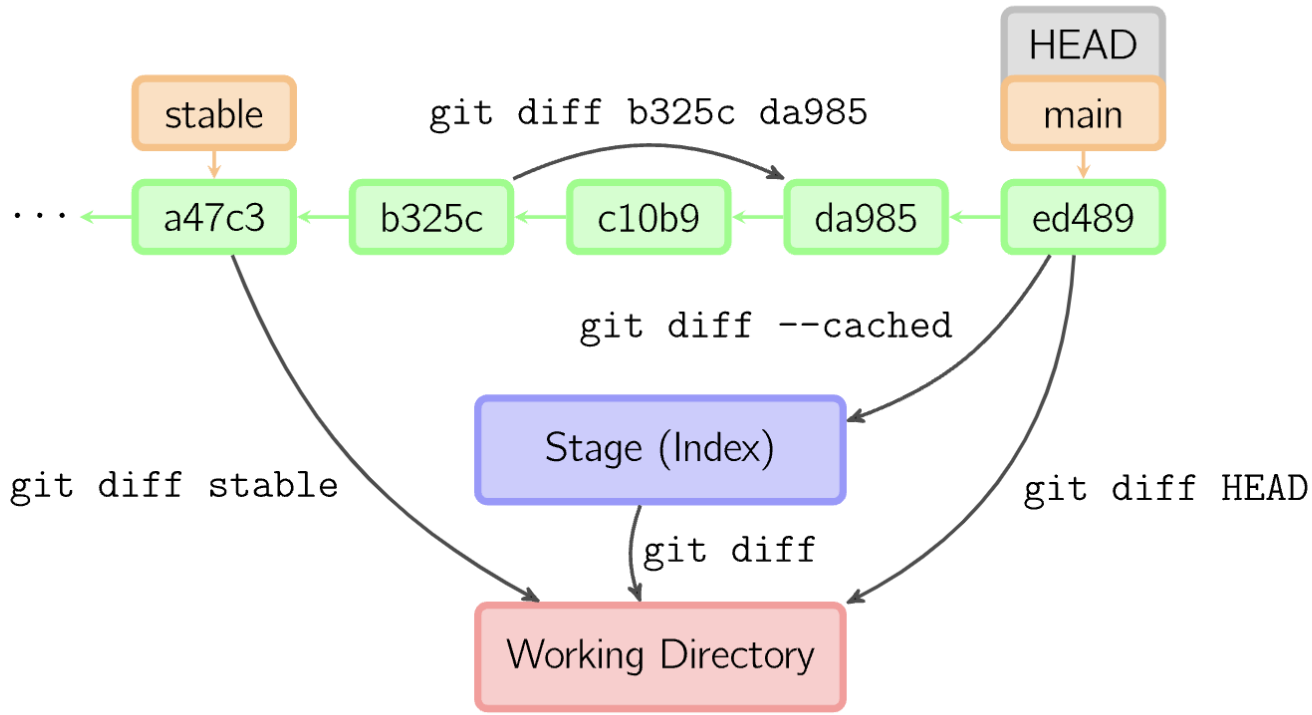
Commit simple



Commit d'une branche

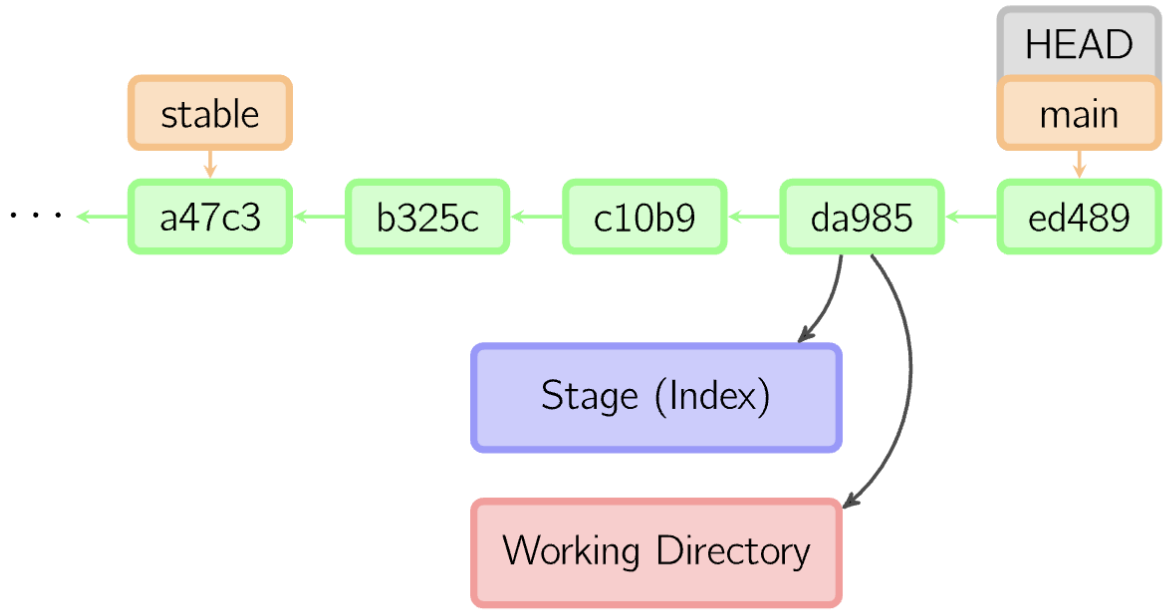


Diff

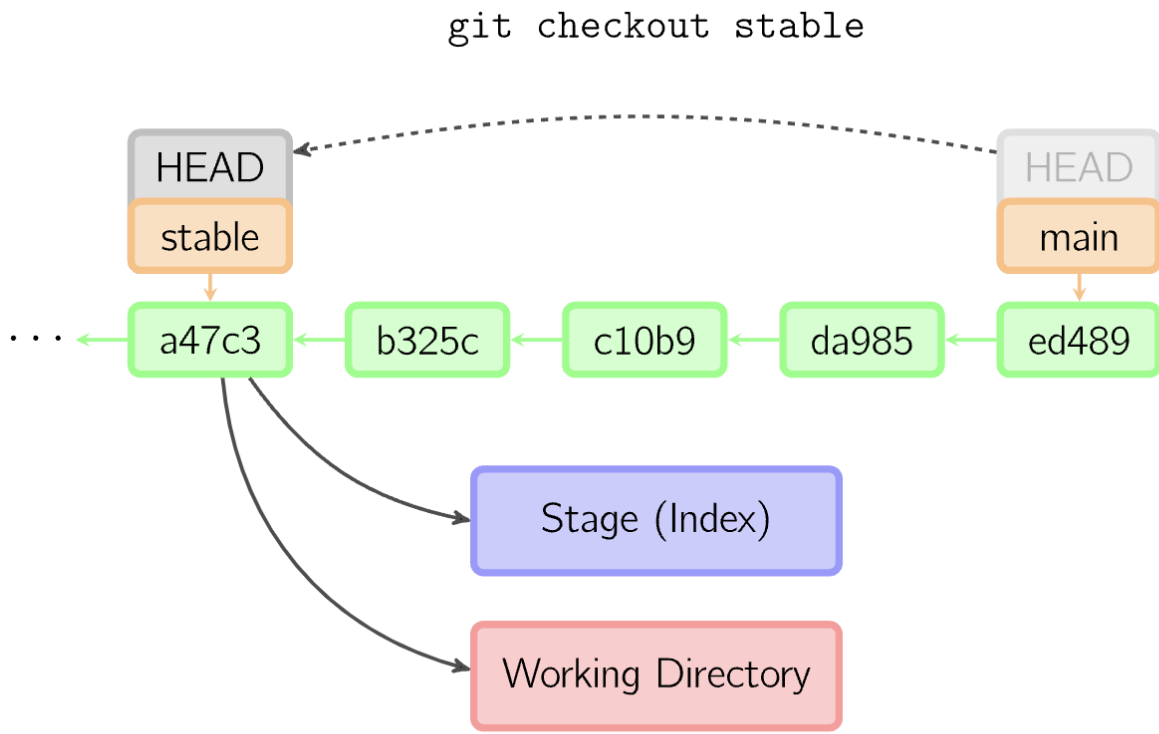

HEAD
stable
main
git diff b325c da985
...
a47c3
b325c
c10b9
da985
ed489
git diff --cached
Stage (Index)
git diff stable
git diff HEAD
git diff
Working Directory

Checkout HEAD

`git checkout HEAD~ files`



Checkout d'une branche



Pense-bête

`git init`	création de dépôt
`git help`	aide
`git clone /chemin/dépôt`	copie un dépôt
`git pull`	mise à jour du dépôt
`git add mon_fichier`	ajout de fichiers
`git rm mon_fichier`	suppression de fichiers
`git commit -m message`	soumettre une version
`git branch ma_branche`	créer une branche
`git merge autre_branche`	fusionner des branches
`git status`	état courant de la copie
`git log`	visualiser le journal
`git diff`	comparer des versions