Introduction à Git et GitHub

Scott Goldstein et Kristen Howard

8 août 2022

Bonjour !

- Cette présentation débutera avec un peu de contexte et puis utilisera un exemple interactif simple.
- Elle met l'accent sur des opérations clefs dans Git. En tant que nouveau·lle utilisarice·eur, vous n'avez pas à tenter de saisir tout ce qui est possible avec Git.
- Nous allons plutôt montrer des exemples de projets spécifiques au monde bibliothéconomique et des trucs et astuces qui ne sont pas révélés dans des démonstrations standards (mais nous vous donnerons quand même une liste de ressources à la fin !)

Qu'est ce que le contrôle de versions ?

- Tout système qui gère les changements à des fichiers à travers le temps.
- Les bénéfices d'un système de contrôle de versions, entre autres :
  - Collaboration
  - Historique des versions

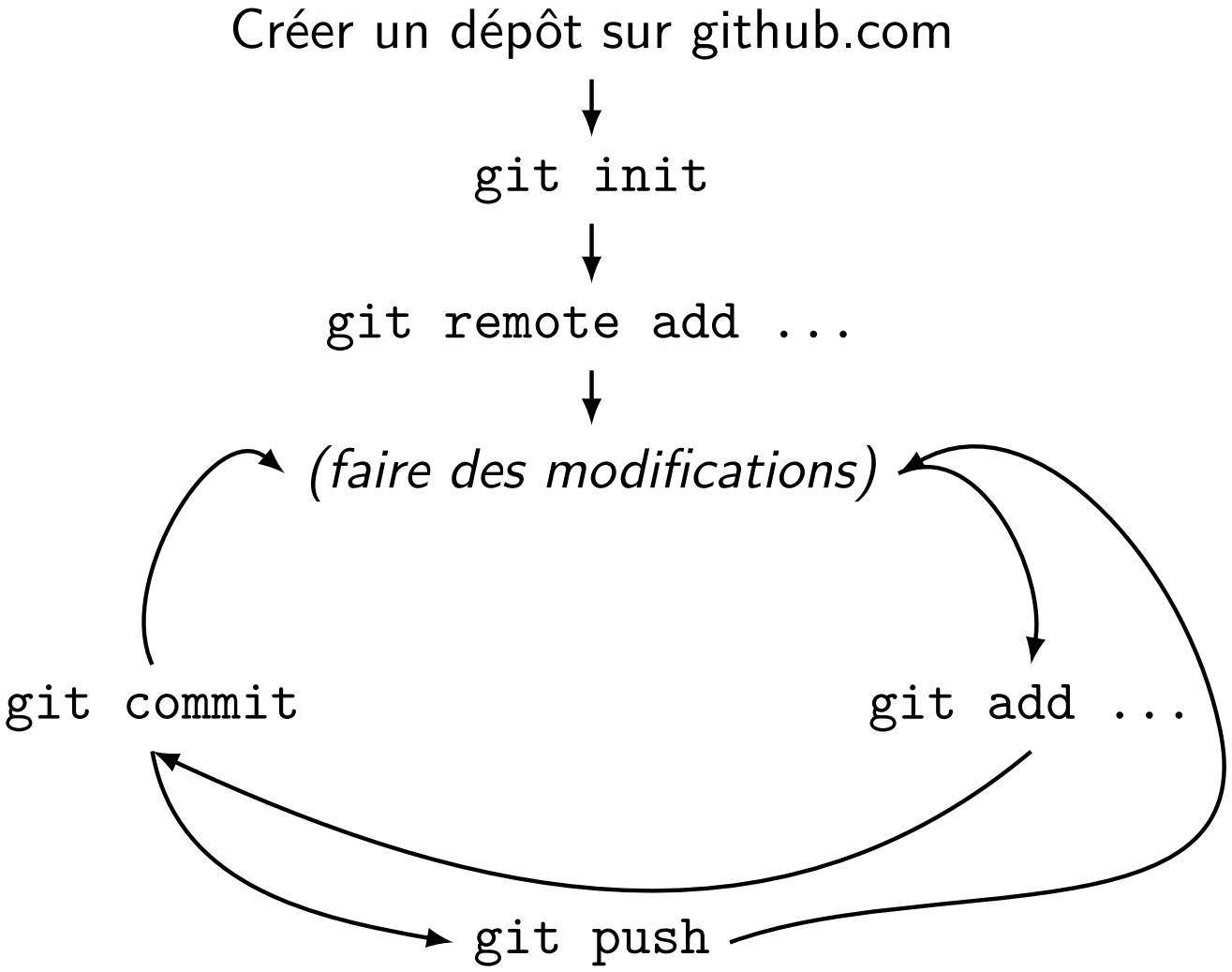
Une courte histoire du contrôle des versions

- SCCS (Source Code Control System) (1972)
- RCS (Revision Control System) (1982)
- CVS (Concurrent Versions System) (1990)
- Subversion (2000)
- Git, Mercurial, Bazaar (2005)

Quel usage de Git en bibliothèque ?

- GitHub Pages – un serveur web gratuit pour des sites web statiques simples
  - Sites professionnels, portfolios
  - Sites divers pour la bibliothèques (conférences, projets, etc.)
  - Prototypes/modèles
- Partage de documents ou d'autres objets (polices, images, etc.)
- Partage de code
  - Script en Python, R, etc.
  - Modules Drupal
  - Etc...

Un exemple très simple



Quelques termes

Un **dépôt (repository ou repo)** désigne l'ensemble des fichiers qui forment votre projet.

- Local ou externe

Une **validation (commit)** est comme une photo de votre projet à un moment donné. Au fil de votre travail, vous « prendrez des photos ». La séquence en ordre de ces commits s'appelle une **branche**. (Un dépôt peut avoir plus d'une branche, mais aujourd'hui nous travaillerons avec une seule branche, appelée la principale, « main ».)

Quelques termes

D'une certaine manière, Git est comme l'historique d'un document au format Microsoft Word ou Google Doc. Ce qui rend les choses un peu plus compliquées est le fait qu'un projet consiste de plusieurs fichiers et ces fichiers ne doivent pas tous être mis ensemble dans les mêmes « commits ». C'est à vous de signifier à Git quels fichiers changent à quel moment.

Arbre de travail (working tree) : les fichiers que vous éditez ; les fichiers que vous voyez devant vous.
L'index de staging (staging area) : les fichiers que vous identifiez comme nécessitant une mise à jour dans le prochain « commit ». Les fichiers qui ne sont pas sur l'index sont ramenés « carried over » par le « commit » précédent.

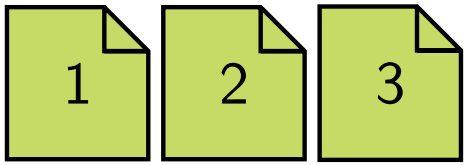
Représentation visuelle

WORKING TREE STAGE REPOSITORY

TERMINAL STATUS

Représentation visuelle

WORKING TREE | STAGE | REPOSITORY

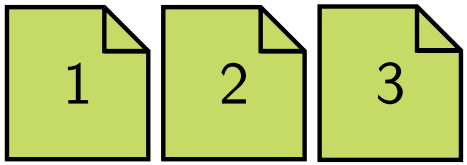


TERMINAL | STATUS

3 fichiers non-suivis

Représentation visuelle

WORKING TREE | STAGE | REPOSITORY



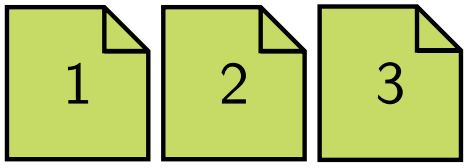
TERMINAL | STATUS

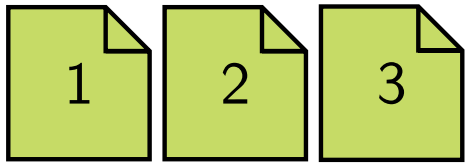
```
git add 1 2 3
```

3 fichiers non-suivis

Représentation visuelle

WORKING TREE | STAGE | REPOSITORY





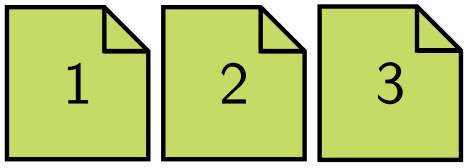
TERMINAL | STATUS

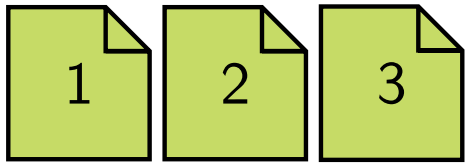
```
git add 1 2 3
```

3 fichiers non-suivis ; 3 fichiers indexés

Représentation visuelle

WORKING TREE | STAGE | REPOSITORY





TERMINAL | STATUS

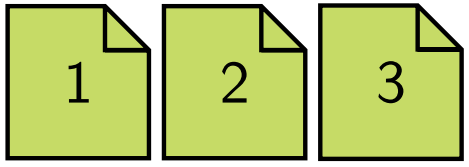
`git commit` | 3 fichiers non-suivis ; 3 fichiers indexés

Représentation visuelle

WORKING TREE

STAGE

REPOSITORY



TERMINAL

```
git commit
```

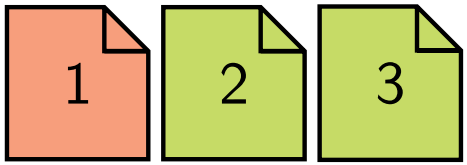
STATUS

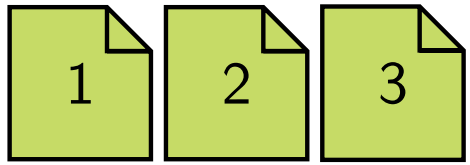
Représentation visuelle

WORKING TREE

STAGE

REPOSITORY





TERMINAL

STATUS

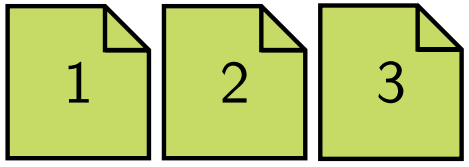
1 fichier modifié

Représentation visuelle

WORKING TREE

STAGE

REPOSITORY



TERMINAL

```
git add 1
```

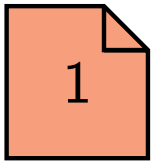
STATUS

1 fichier modifié

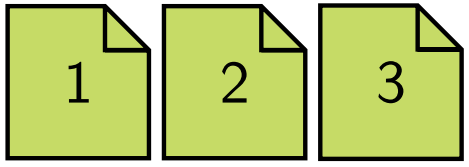
Représentation visuelle

WORKING TREE

STAGE



REPOSITORY



TERMINAL

STATUS

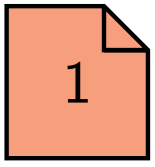
```
git add 1
```

1 fichier modifié ; 1 fichier indexé

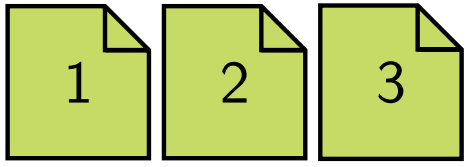
Représentation visuelle

WORKING TREE

STAGE



REPOSITORY



TERMINAL

STATUS

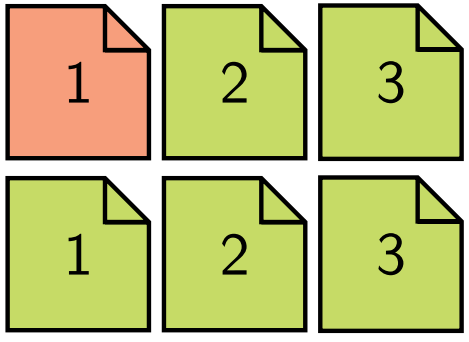
`git commit` 1 fichier modifié ; 1 fichier indexé

Représentation visuelle

WORKING TREE

STAGE

REPOSITORY



TERMINAL

STATUS

```
git commit
```

Représentation visuelle

WORKING TREE

STAGE

REPOSITORY

TERMINAL

STATUS

```
git commit
```

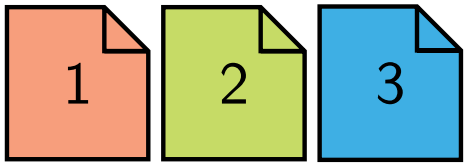
Représentation visuelle

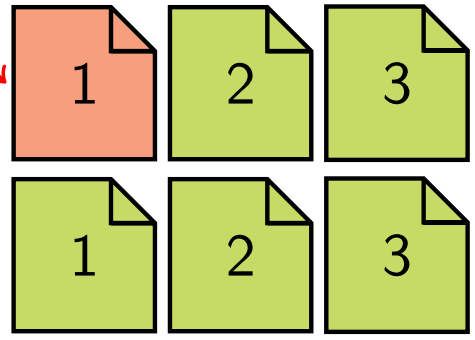
WORKING TREE

STAGE

REPOSITORY

HEAD





TERMINAL

STATUS

1 fichier modifié

Représentation visuelle

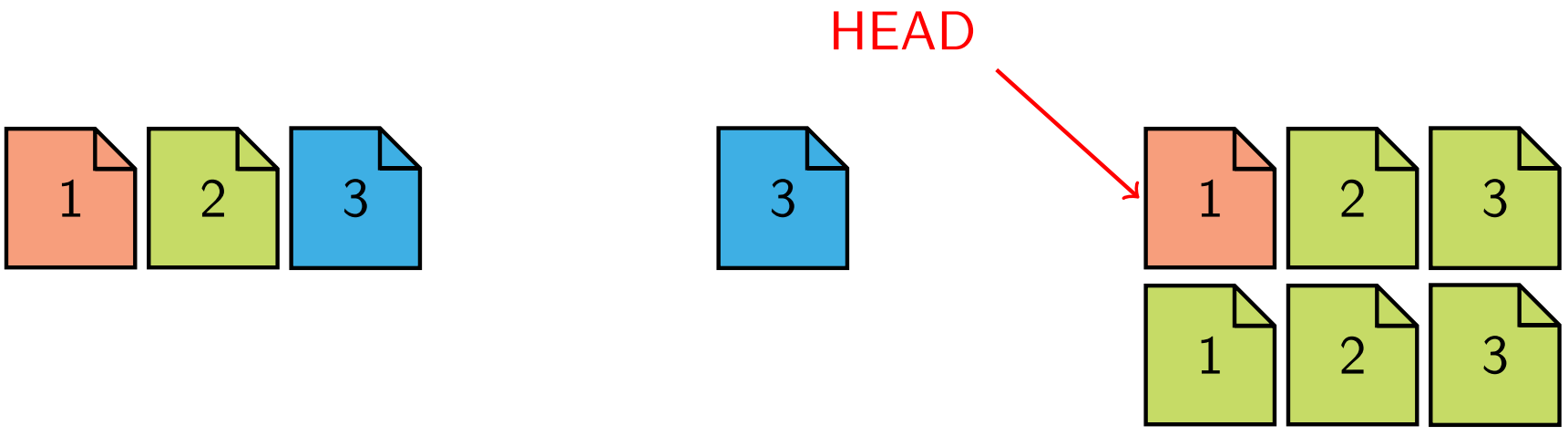
WORKING TREE | STAGE | REPOSITORY

TERMINAL | STATUS

`git add 3` | 1 fichier modifié

Représentation visuelle

WORKING TREE | STAGE | REPOSITORY

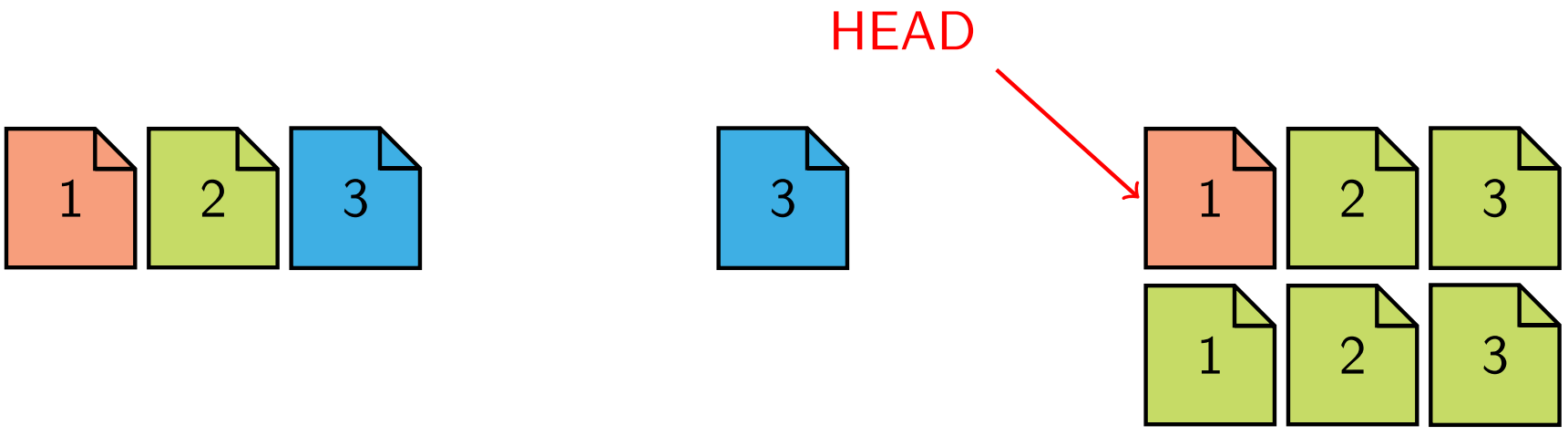


TERMINAL | STATUS

`git add 3` 1 fichier modifié ; 1 fichier indexé

Représentation visuelle

WORKING TREE | STAGE | REPOSITORY



TERMINAL | STATUS

`git commit` 1 fichier modifié ; 1 fichier indexé

Représentation visuelle

WORKING TREE

STAGE

REPOSITORY

TERMINAL

STATUS

```
git commit
```

Exercice pratique

Voyons l'exemple d'un profil GitHub README.

Jeton d'accès personnel

New personal access token

Personal access tokens function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to authenticate to the API over Basic Authentication.

Note

SG

What's this token for?

Expiration *

Custom... 2022-12-31

Select scopes

Scopes define the access for personal tokens. Read more about OAuth scopes.

repo	Full control of private repositories
repo:status	Access commit status
repo_deployment	Access deployment status
public_repo	Access public repositories
repo:invite	Access repository invitations
security_events	Read and write security events

Ce que j'ai appris « sur le tas »

- N'oubliez pas de configurer votre adresse de courriel sur votre poste de manière à ce qu'elle soit la même que celle que vous utilisez sur github.com
- N'en mettez pas trop
  - `git add --all --dry-run` est très utile !
  - Pensez à utiliser un fichier .gitignore
- Ne faites pas trop de commits
  - Habituez-vous à utiliser des **commits atomiques**
- Utilisez fréquemment `git status`, puis lisez le résultat et posez-vous la question : est-ce que cela a du sens ? Cela vous évitera de rater des erreurs que vous devrez corriger ensuite.
- Il vaut toujours mieux utiliser les meilleures pratiques du type de projet que vous êtes en train de faire (meilleures pratiques Python dans Python, meilleures pratiques Jekyll pour Jekyll, pareil pour Drupal, etc.)

Ressources

Library Carpentry: Introduction to Git

git - the simple guide

git/github guide

Oh Shit, Git!?!