GIT

LOGICIEL DE GESTION DE VERSIONS DÉCENTRALISÉ

David DEMONCHY @daviddem

A PROPOS

A quoi peut nous servir un logiciel de versionnement et pourquoi s'y mettre ?

David DEMONCHY @daviddem

Un gestionnaire de version est un système qui enregistre l’évolution d’un fichier, d’un ensemble de fichiers au cours du temps de manière à ce que l’on puisse rappeler une version antérieure d’un fichier à tout moment.

LES SYSTÈMES DE GESTION DE VERSION CENTRALISÉS

Ces systèmes tels que CVS, Subversion, et Perforce mettent en place un serveur central qui contient tous les fichiers sous gestion de version.

Des clients qui peuvent extraire les fichiers de ce dépôt central.

Pendant de nombreuses années, cela a été le standard pour la gestion de version.

LES SYSTÈMES DE GESTION DE VERSION DISTRIBUÉS

C’est à ce moment que les systèmes de gestion de version distribués entrent en jeu (**DVCS** **D**istributed **V**ersion **C**ontrol **S**ystems).

Dans un DVCS (tel que Git, Mercurial, Bazaar ou Darcs), les clients n’extraient plus seulement la dernière version d’un fichier, mais ils dupliquent complètement le dépôt.

Ainsi, si le serveur disparaît et si les systèmes collaboraient via ce serveur, n’importe quel dépôt d’un des clients peut être copié sur le serveur pour le restaurer.

UNE RAPIDE HISTOIRE DE GIT

Comme de nombreuses choses extraordinaires de la vie, Git est né avec une dose de destruction créative et de controverse houleuse.
Le noyau Linux est un projet libre de grande envergure.
Pour la plus grande partie de sa vie (1991–2002), les modifications étaient transmises sous forme de patch et d'archives de fichiers.
En 2002, le projet du noyau Linux commença à utiliser un DVCS propriétaire appelé BitKeeper.

David DEMONCHY @daviddem

En 2005, les relations entre la communauté développant le noyau Linux et la société en charge du développement de BitKeeper furent rompues, et le statut de gratuité de l'outil fut révoqué.

Cela poussa la communauté du développement de Linux et plus particulièrement Linus Torvalds, le créateur de Linux à développer son propre outil en se basant sur les leçons apprises lors de l'utilisation de BitKeeper.

CERTAINS DES OBJECTIFS DU NOUVEAU SYSTÈME ÉTAIENT LES SUIVANTS

vitesse

conception simple

support pour les développements non linéaires
(milliers de branches parallèles)

complètement distribué

capacité à gérer efficacement des projets d'envergure tels que le noyau Linux
(vitesse et compacité des données)

PRESQUE TOUTES LES OPÉRATIONS SONT LOCALES

La plupart des opérations de Git ne nécessitent que des fichiers et ressources locaux

Généralement aucune information venant d’un autre ordinateur du réseau n’est nécessaire.

PRESQUE TOUTES LES OPÉRATIONS SONT LOCALES

Si vous êtes habitué à un CVCS où toutes les opérations sont ralenties par la latence des échanges réseau.

Cet aspect de Git vous fera penser que les dieux de la vitesse ont octroyé leurs pouvoirs à Git.

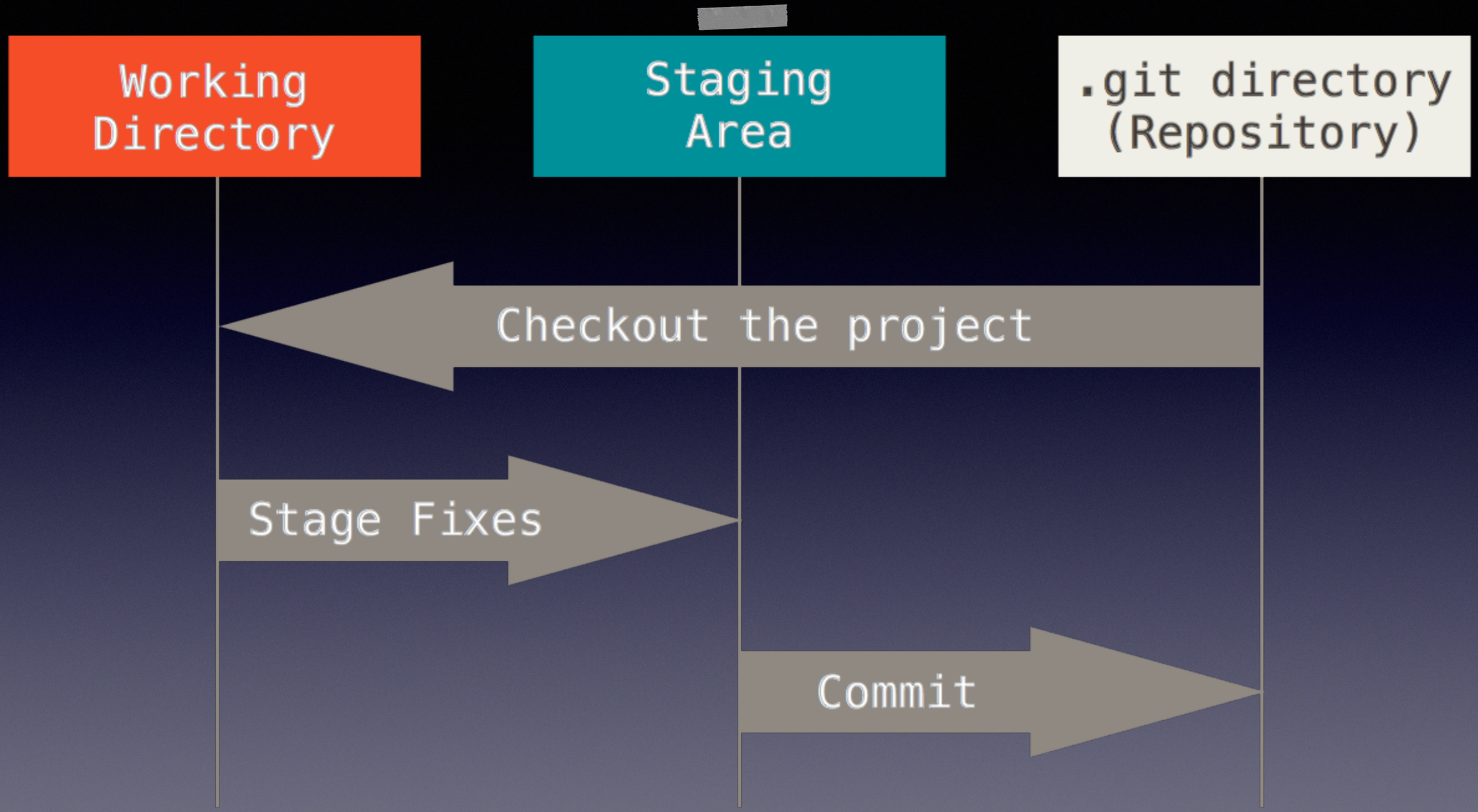
PRESQUE TOUTES LES OPÉRATIONS SONT LOCALES

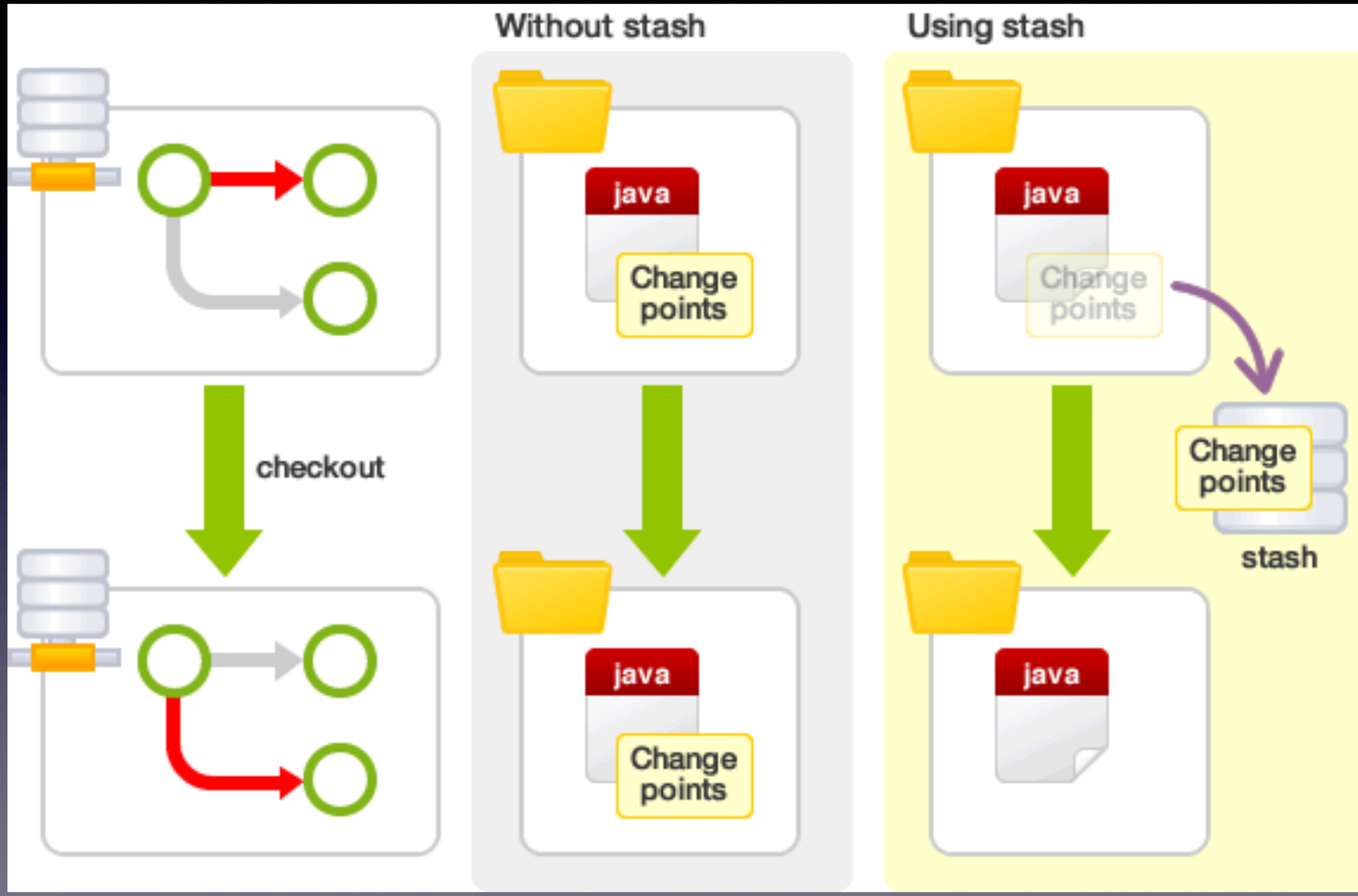
Comme vous disposez de l’historique complet du projet localement sur votre disque dur, la plupart des opérations semblent instantanées.

COMMANDES GIT

NIVEAU « UTILISATEUR »
± 50 commandes avec complétion

NIVEAU « CORE »
± 60 commandes sans complétion

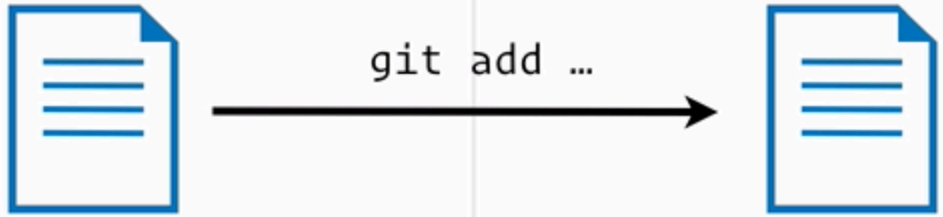

Working Directory
Staging Area
.git directory (Repository)
Checkout the project
Stage Fixes
Commit


Without stash
Using stash
java
Change points
checkout
java
Change points
java
Change points
Change points
stash
java

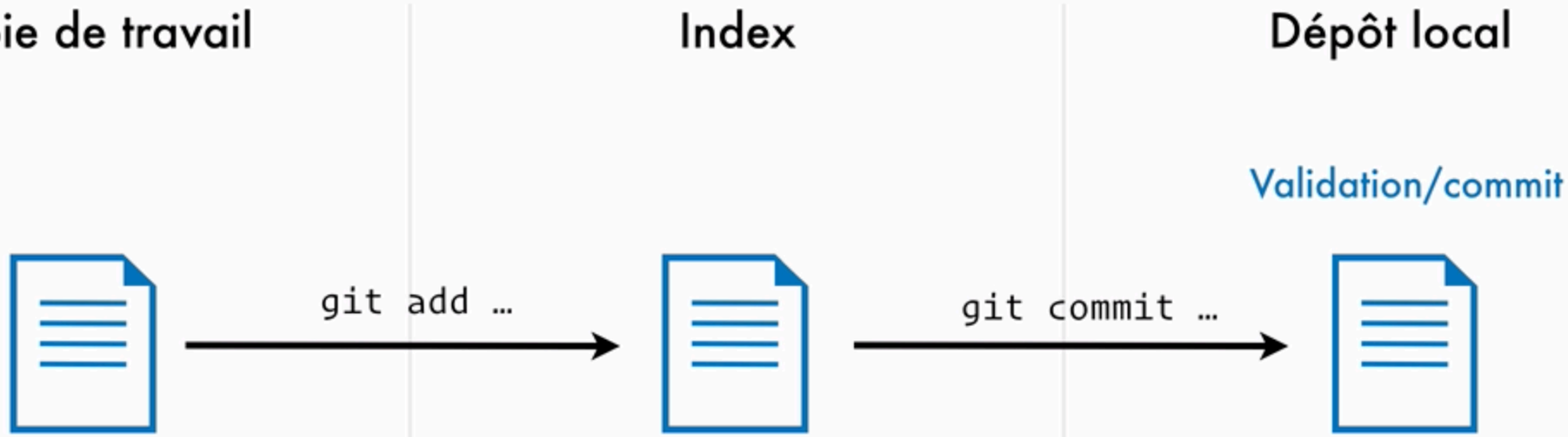
Copie de travail	Index	Dépôt local
Création d'un fichier 	État : non suivi	

Copie de travail | Index | Dépôt local

Ajout du fichier à l'index



État : indexé

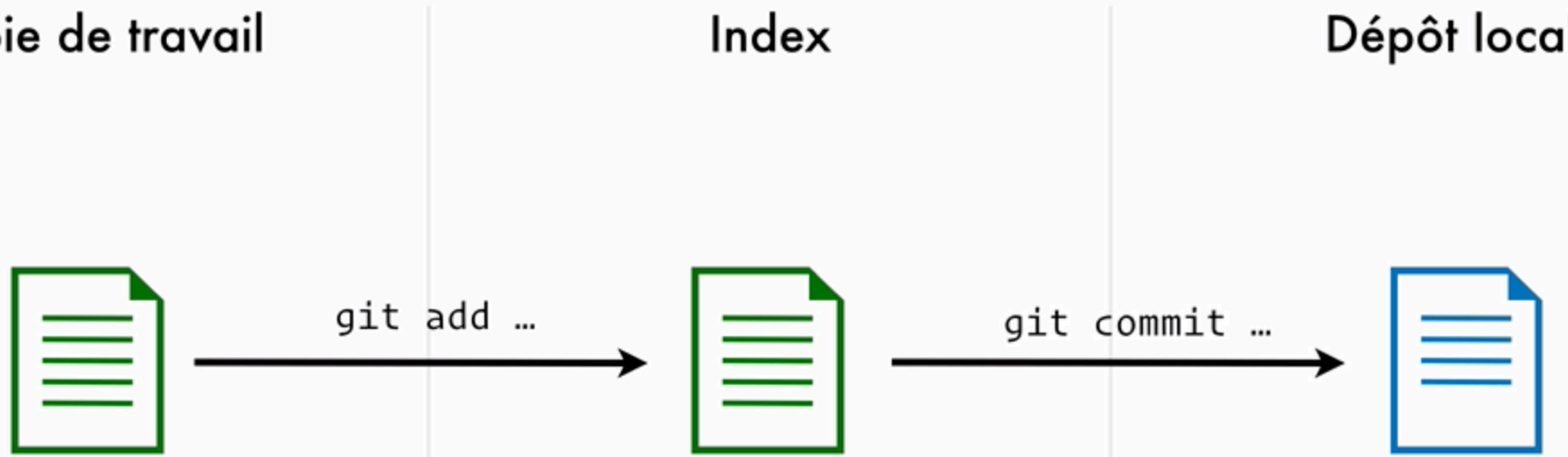


État : non modifié

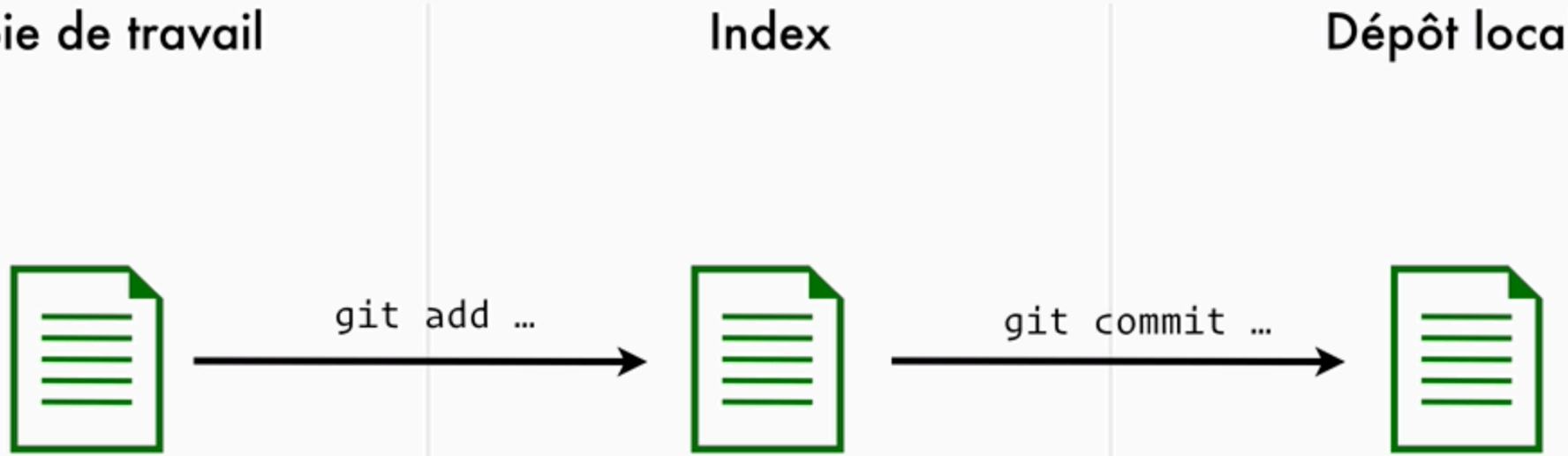
Copie de travail | Index | Dépôt local

Modification d'un fichier

État : modifié



État : indexé



État : non modifié

Copie de travail | Index | Dépôt local

Suppression d'un fichier

État : supprimé

COMMANDES DE BASE

config

init

OBTENIR LA CONFIGURATION LOCAL OU GLOBAL DE GIT

```
ma@machine:/~/depot_git$ git config --list --local

$ git config --list —global
```

*LOCAL NÉCESSITE D'ÊTRE DANS UN DÉPÔT LOCAL

David DEMONCHY @daviddem

GIT CONFIG

```
$ git config --local user.name "David Demonchy"
```

Configure mon dépôt git actuel pour utiliser **David Demonchy** comme nom d'utilisateur (lors des commits)

ou

```
$ git config --global user.email "dd@pop.eu.com"
```

Utilisera cette adresse email sur tous mes dépôts y compris celui sur lequel je me trouve.

GIT INIT

Initialise le contenu d'un répertoire pour le maintenir sous git.

```
$ git init
```

```
$ git init .
```

```
$ git init /le/chemin/vers/mon/depot/a/initialiser
```

C'EST PARTI

David DEMONCHY @daviddem

```
$ mkdir -p ~/git/premier_pas/
```

```
$ cd ~/git/premier_pas/
```

```
$ git init .
```

Notre répertoire est maintenant versionné.
Nous allons pouvoir manipuler des fichiers
à l'intérieur de celui-ci.

Créer un fichier index.html
et y mettre la syntaxe de base d'un fichier HTML5

```html
<!doctype html>
<html lang="fr">
        <head>
                <meta charset="utf-8">
                <title>Titre de la page</title>
                <link rel="stylesheet" href="style.css">
                <script src="script.js"></script>
        </head>
        <body>
                <!-- ... ... -->
        </body>
</html>
```

index.html hosted with ❤ by GitHub view raw

David DEMONCHY @daviddem

```
$ git status
```

```
$ git add index.html
```

```
$ git status
```

```
$ git commit -m 'index html de base’
```

```
$ git status
```

```
$ git log
```

ET MAINTENANT ?

NOUS ALLONS OFFRIR NOTRE CODE AU MONDE

Il y a plusieurs façon pour faire cela,
mais nous allons nous intéresser au plus courant.
Utiliser une application de dépôts en ligne

FRAMAGIT GITHUB BITBUCKET…

David DEMONCHY @daviddem