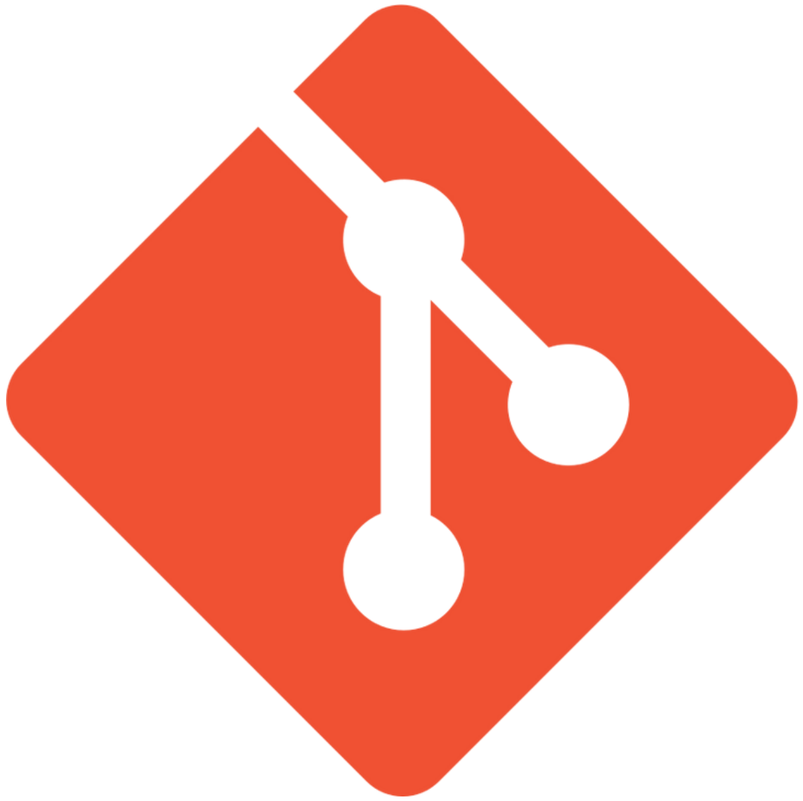
Git: démarrage
20231123

vendredi 23 novembre



git

vendredi 23 novembre
GIT: démarrage

- principes
- premiers pas
- github

Autres besoins

UNE SOURCE D'INFO

https://git-scm.com/book/fr/v2

Git: démarrage
20231123

Systèmes de gestion de version

VCS

“ Suivi de l’évolution d’un ensemble de fichiers”

- **voir l’historique**
- **revenir en arrière: pourquoi ?**
- **backup**
- **travail collaboratif: quels ex. ?**

Git: pourquoi

- **dev / Linus Thorvald en 2005 pour le dev du noyau linux**
- **plus léger et rapide: pourquoi ?**
- **décentralisé : c-a-d ?**
- **branching facile: heu ... ?**

Git pour le noyau linux

https://fr.wikipedia.org/wiki/Noyau_Linux

Entre mars et avril 2005: lignes code x 2 (de **4,4 millions à 8,8 millions)**.
Début 2009: la V 2.6.30 > **11,5 millions** de lignes de code **28 000** fichiers
Entre Noël 2008 et janvier 2010: **2,8 millions** de lignes ont été ajoutées
Entre 2005 et mi-2009: **+5 000** développeurs et **+500** entreprises

Stats Git du noyau linux en 2020

27,8 millions de lignes

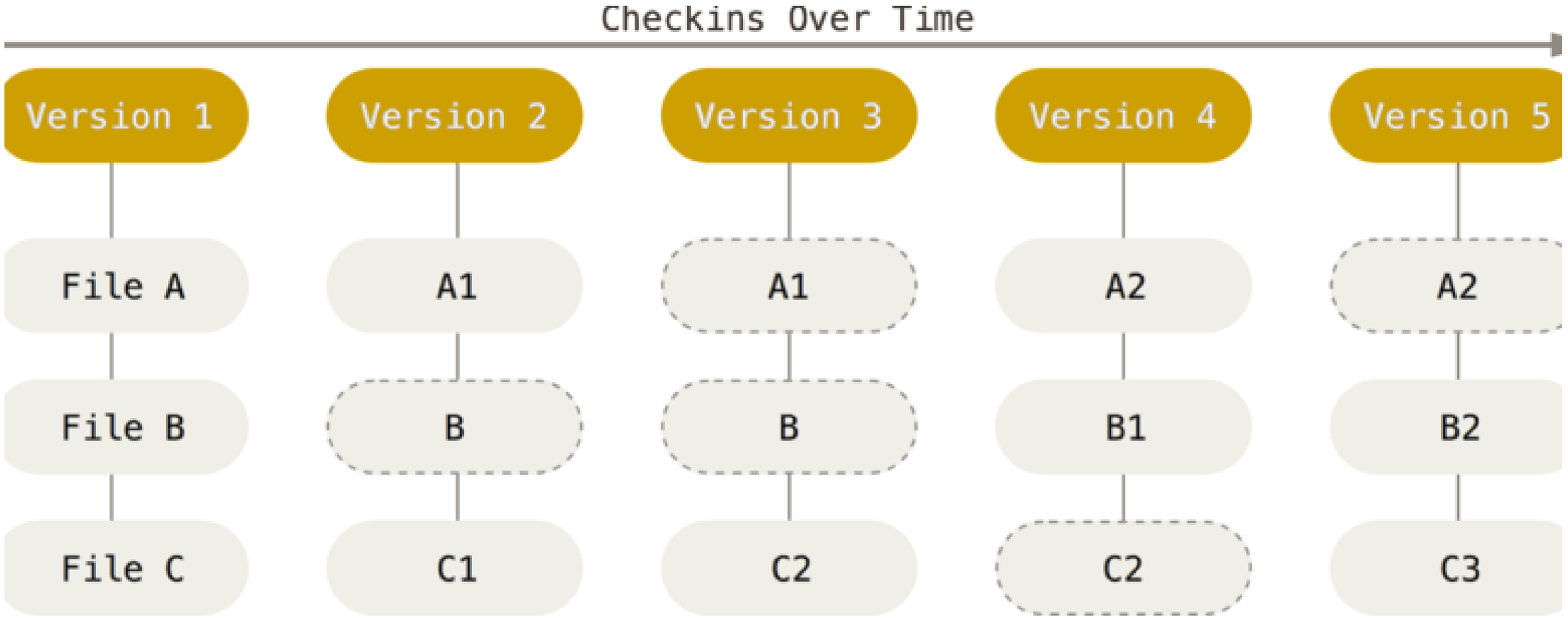
66 492 fichiers,

887 925 commits

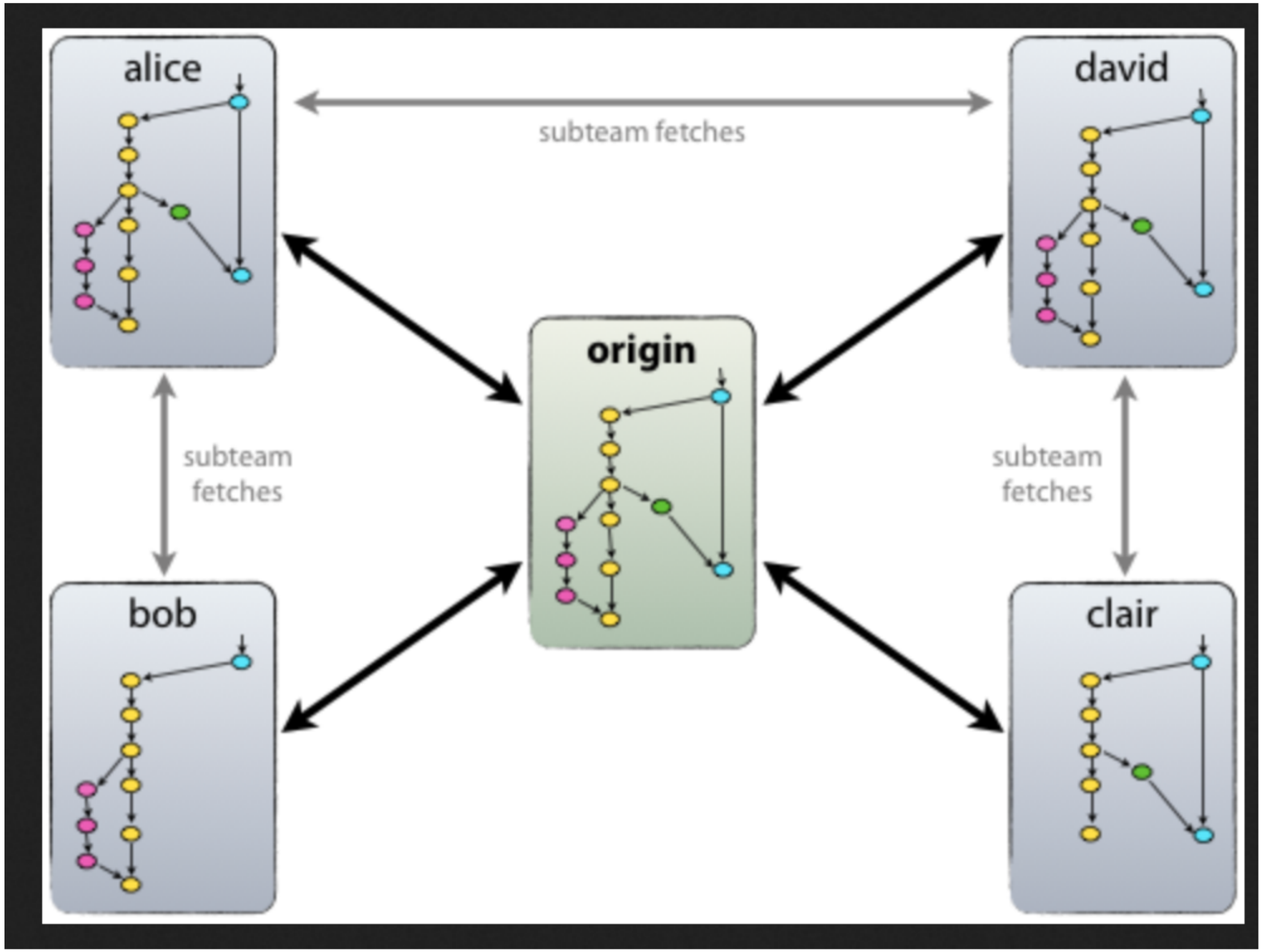
plus de 21 000 auteurs

Source: le monde informatique

Git: plus léger (snapshots du tree + stockage local)



Git:décentralisé : c-a-d ?



Git: démarrage
20231123

évolutions versions

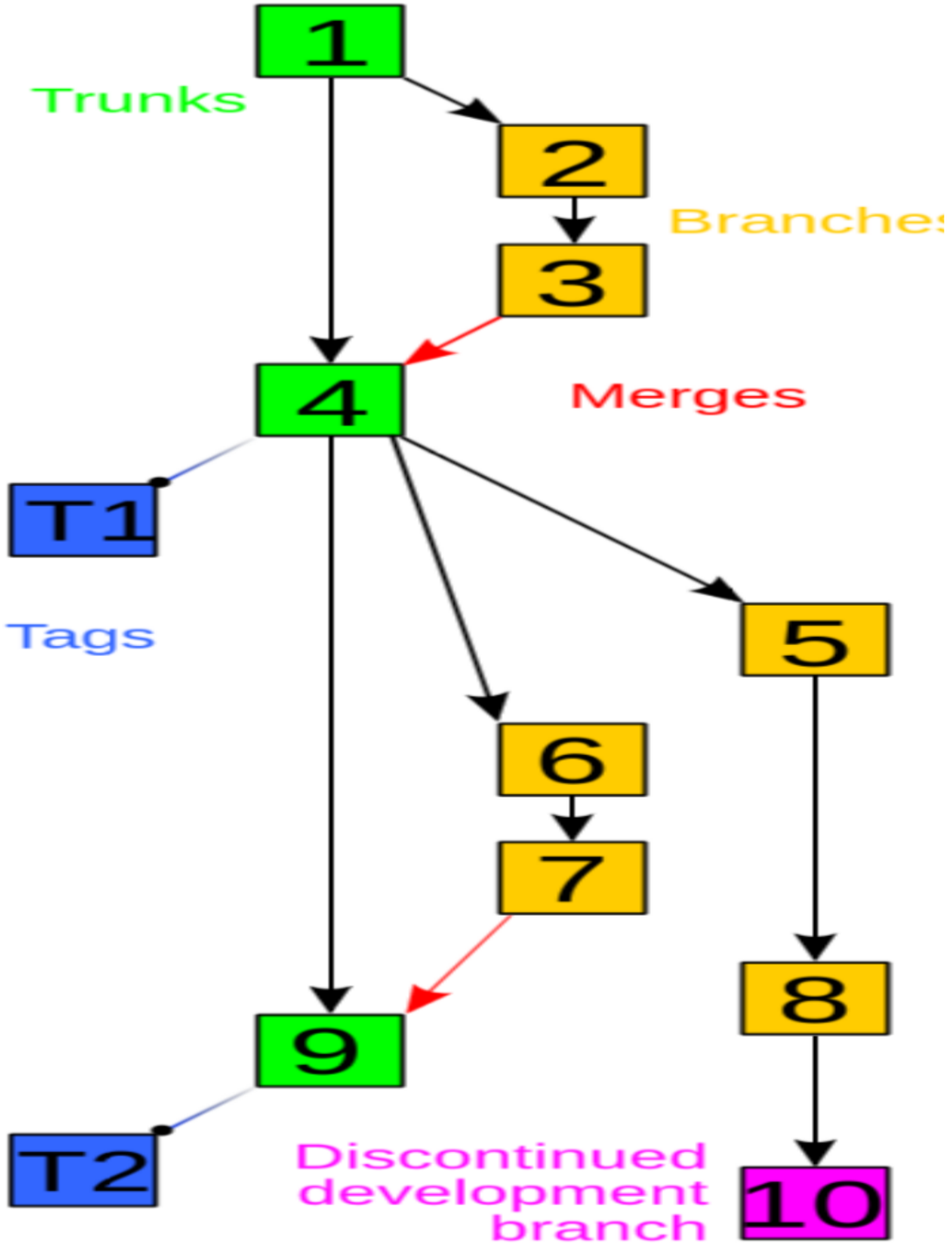
Trunk: tronc, flux principal

Branch: bifurcation dans le dev

Merge: fusion de branches

Tags: marquage, repère

V-13.4.22



Git: plateformes et dépôts

“repository”

- **gitlab.com**
- **github.com**
- **dépôts distants locaux / perso**

Git: démarrage
20231123

Git: on va faire quoi nous

déc

- **suivi local d’historique du code**
- **utilisation des dépôts distants**
- **échanger avec d’autres dev**
- **fonction backup**
- **Usages** **Github**
  (gh-actions, pull request)

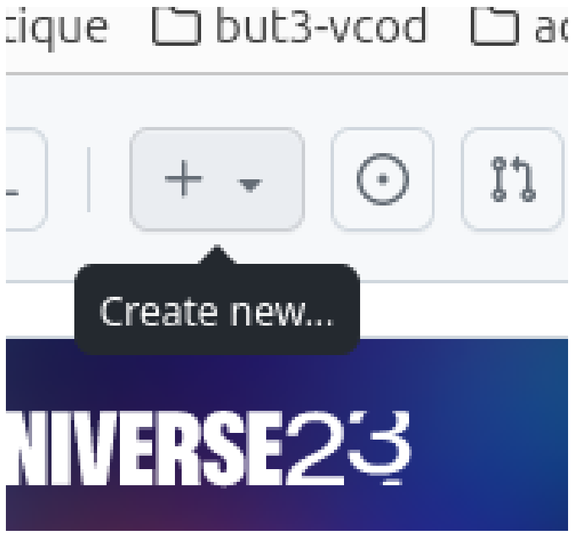
Les Outils pour la pratique

- **shell Windows: vérifier git --version**
- **compte github pour chacun**

function Backup -> Retrouver l’environnemnet à chaque cours quelque soit le lieu, ou l’ordi.

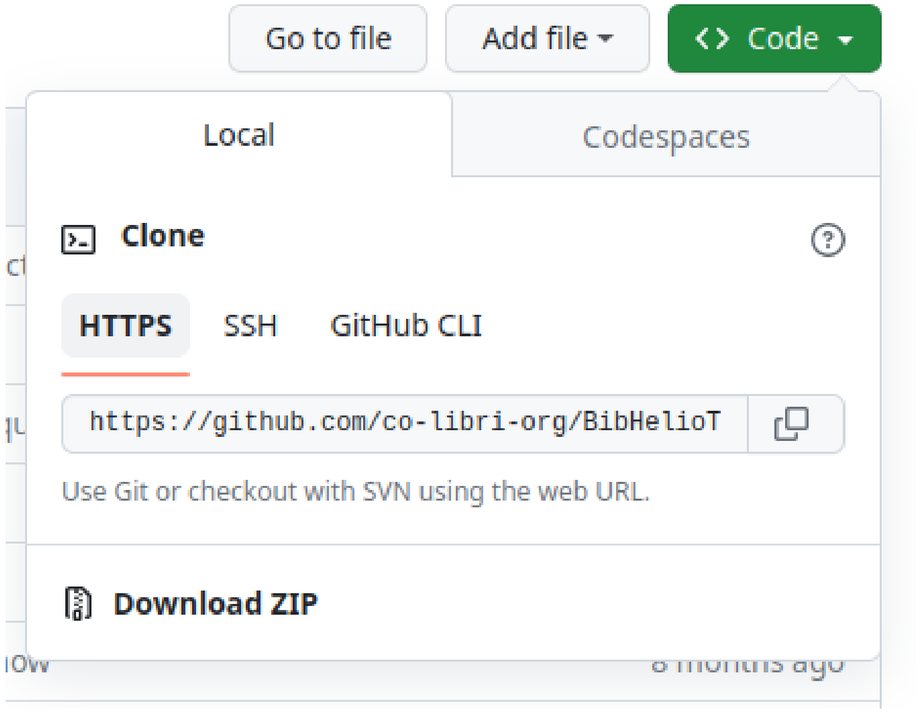
chacun son projet

1- creer son compte sur
https://github.com



2- creer un projet avec README.md

Récupérer en local



1- identifier l’url

2- récupérer en local:

```
$> git clone https://github.com/mon_compte/mon_projet
```

Ya quoi dans mon répertoire

```
$> cd mon_projet/ && ls
    -> .git/
    ->  README.md
```

```
$> cd .git/ && ls
    -> ./config
    -> ./HEAD
    -> ./refs/
```

Git: démarrage
20231123

ZONES ou ESPACES

Le Répertoire de Travail

(project_directory/)

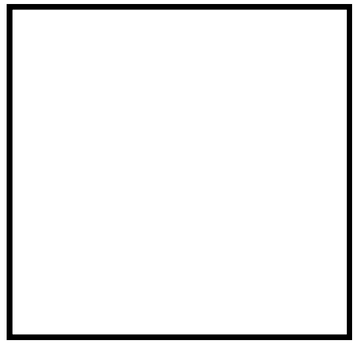
L'Index : "staging area"

(Salle d'attente)

Le Dépot: "repository"

(local .git/, distant github)

2 Possibilités pour un fichier



Non suivi

Suivi

2 Possibilités pour un fichier

```
$> git status
Sur la branche main
rien à valider, la copie de travail est propre
```

```
$> echo “hello world” >> HelloWorld.txt && git status
Sur la branche main
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
 HelloWorld.txt
aucune modification ajoutée à la validation mais des fichiers non
suivis sont présents (utilisez "git add" pour les suivre)
```

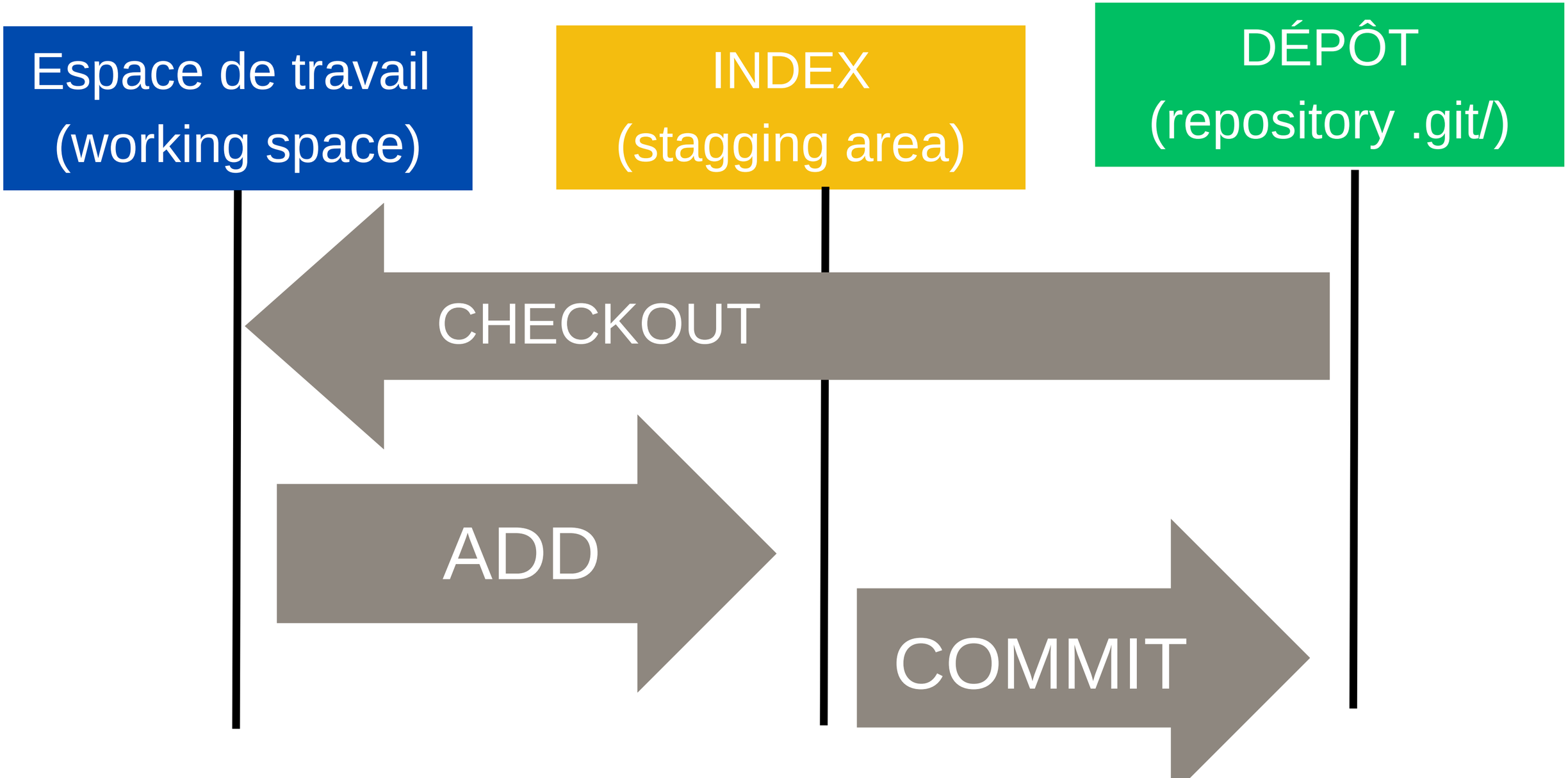
2 Possibilités pour un fichier

```
$> git commit -m “Nouveau fichier de salutation”
[main 3ca453d] “Nouveau fichier de salutation”
 1 file changed, 1 insertion(+)
 create mode 100644 HelloWorld.txt

$> git status
 Sur la branche main
rien à valider, la copie de travail est propre
```

Git: démarrage
20231123

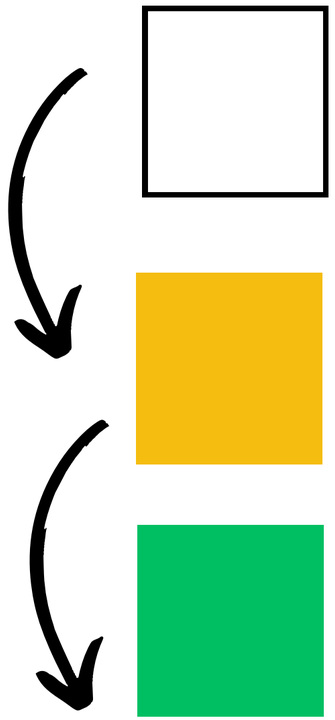
Workflow ZONES



Git: démarrage
20231123

Workflow

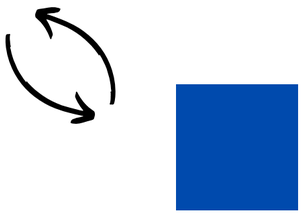
De “untracked” à “tracked”



Non suivi (“untracked”)

En attente (“stagged”)

Validé (“saved in db”)



Non Modifié

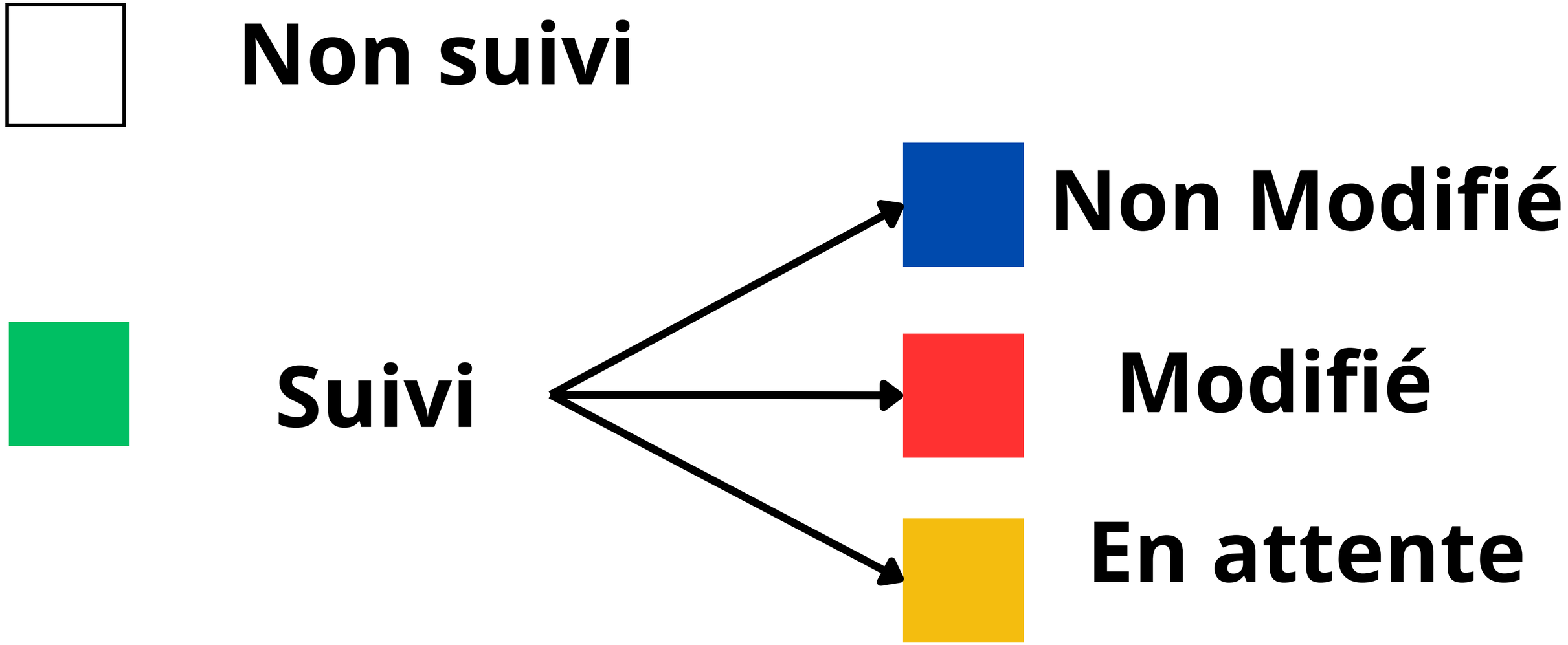
2 Possibilités pour un fichier

```
$> git status
Sur la branche main
rien à valider, la copie de travail est propre
```

```
$> echo “hello world” >> HelloWorld.txt && git status
Sur la branche main
Fichiers non suivis:
  (utilisez "git add <fichier>..." pour inclure dans ce qui sera validé)
 HelloWorld.txt
aucune modification ajoutée à la validation mais des fichiers non
suivis sont présents (utilisez "git add" pour les suivre)
```

Git: démarrage
20231123

3 États d'un fichier suivi



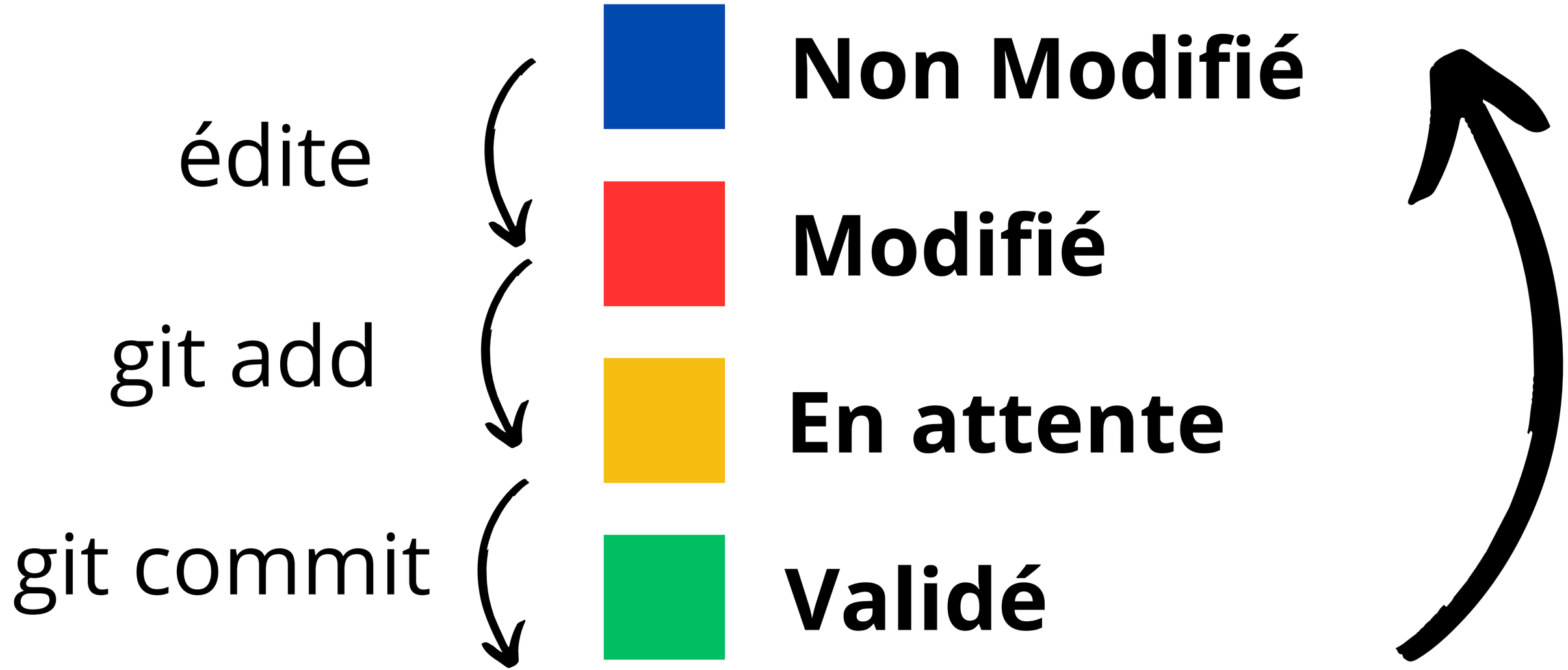
3 États d’un fichier suivi

```
$> echo “## Sous Titre” >> README.md
$> git status
$> git add
$> git status
$> git commit -m “Ajout d’un sous titre”
$> git status
```

Git: démarrage
20231123

Workflow

suivi -> édité -> validé



Git: démarrage
20231123

Au Final, un fichier a 5 ÉTATS

Non suivi (“untracked”)

Non Modifié (“unmodified”)

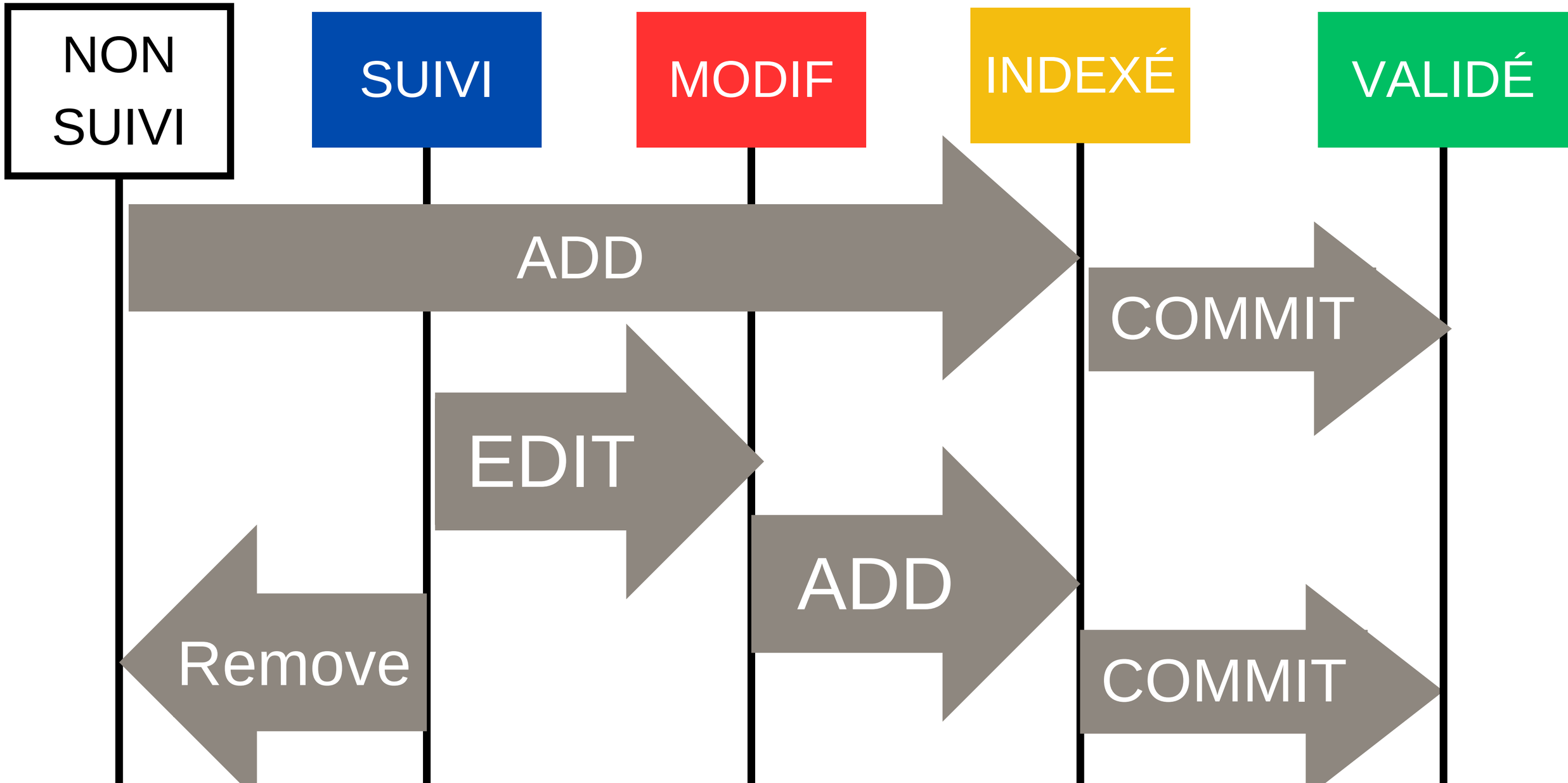
Modifié (“modified”)

En attente (“stagged”)

Validé (“saved in db”)

Git: démarrage
20231123

Workflow ÉTATS



Git pratique: configurer

```
git config --global user.name "Richard Hitier"
git config --global user.email rhitier@gmail.com
```

voir le fichier ~/.gitconfig

Git pratique:
voir ce qui est là

```
git branch
git status
git status -s -uno
```

```
git log
git log --pretty --one-line
```

Git pratique:
pousser sur le dépôt distant

```
git remote
git branch
git push origin main
```

vérifier sur

Qu’est ce qu’un commit

- **sauvegarde d’un état de l’espace de travail**
- **lié au précédent**
- **avec un message**
- **et un N° SHA**

Git: démarrage
20231123

Untracked
Added
Commited

Dépôt Local

./.git/

!=

Dépôt Distant

Dépôt Distant

git remote

Git pratique: init

mkdir ./mon_rep

git init => que se passe t il ?

tree .git

./config

./HEAD

./refs/

Git pratique: premier ajout

echo hello >> README.md

git add README.md

git commit -m “premier commit”

Git pratique:
log pretty

echo hello >> README.md && git commit -am “mmesage”
echo hello >> README.md && git commit -am “mmesage”
echo hello >> README.md && git commit -am “mmesage”
echo hello >> README.md && git commit -am “mmesage

git log

git log --pretty --one-line

Git pratique: advanced

git branch

git merge

git rebase

Git pratique: github

création de compte

2FA

ssh keys

Git pratique: github

git clone

git push

git pull