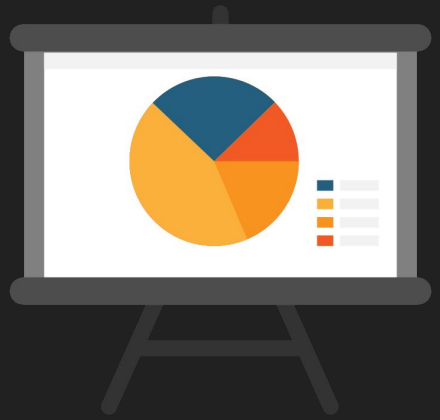
GIT

Rachid EDJEKOUANE (edjek@hotmail.fr)

Prérequis

- Connaître le fonctionnement du terminal
- Explorateur de fichiers
- Savoir créer des dossiers
- Se situer dans une arborescence de fichiers
- Ponctualité

Introduction



GIT, quézako?

Git est un système de contrôle de version populaire qui permet de suivre les modifications apportées aux fichiers d'un projet au fil du temps.

Il enregistre chaque modification dans un historique et permet de revenir à des versions antérieures si nécessaire.

Les avantages?

Git facilite la collaboration entre les développeurs en permettant le partage des modifications et la fusion harmonieuse des branches de développement.

Il offre une gestion efficace des conflits lors de la fusion et permet de travailler à la fois localement et avec des référentiels distants.

Git est largement utilisé dans le développement de logiciels pour assurer la traçabilité, la collaboration et la sauvegarde des projets.

Encore plus d’avantages…

Léger et rapide : ce qui lui permet de gérer de grands projets avec efficacité.

Dépôt distant : ce qui permet aux équipes de partager leur code et de travailler ensemble même à distance. Ils facilitent le processus de sauvegarde et de récupération en cas de perte de données.

Écosystème riche : Git bénéficie d'un large écosystème d'outils, de services d'hébergement de dépôts et de communautés actives. Il est largement adopté par l'industrie et dispose d'une documentation complète et de nombreuses ressources disponibles.

Un peu d’histoire : Git

Git a été créé par Linus Torvalds en 2005 pour répondre aux limitations et aux problèmes de performance rencontrés avec les systèmes de contrôle de version existants.

Torvalds avait besoin d'un outil de contrôle de version distribué et rapide pour gérer le développement du noyau Linux.

Il a conçu Git avec un modèle de stockage efficace basé sur des instantanés (snapshots) plutôt que sur des différences de fichiers.

Popularité

Git a été largement adopté par la communauté open source et est devenu le système de contrôle de version le plus populaire.

Il est connu pour sa vitesse, sa flexibilité et sa capacité à gérer efficacement des projets de toutes tailles.

Des services d'hébergement tels que GitHub et GitLab ont contribué à la popularité de Git en offrant des fonctionnalités supplémentaires pour la collaboration et la gestion des dépôts. Aujourd'hui, Git est utilisé dans divers domaines du développement de logiciels et reste un outil essentiel pour la gestion des versions.

Découverte

Git : comment ça marche ?

Pour utiliser Git il suffit de télécharger sa version sur le site officiel :

https://git-scm.com/.

Git : configuration

Ouvrir le terminal : ctrl + ù

```
1  git config --global user.name "Votre nom"
2  git config --global user.email "votre@email.com"
3  git config --global core.editor "code --wait"
4  git config color.ui true
```

En pratique

Git : repository

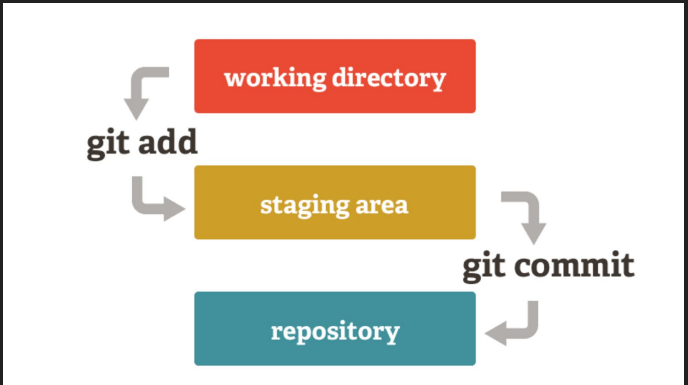
Repository (Dépôt) :

Un dépôt Git est un espace où les fichiers du projet sont stockés, et où l'historique des modifications est enregistré. Il peut être situé localement sur votre machine ou sur un serveur distant.

Git : zone de transit (staging area)

La zone de staging est un concept clé de Git qui joue un rôle important dans la préparation des modifications avant de les valider (commit).

La zone de staging permet de sélectionner les fichiers et les modifications spécifiques à inclure dans le prochain commit.



Git : sauvegarder ses changements

Un commit représente une sauvegarde cohérente des modifications apportées à votre dépôt à un instant donné :

```
git commit -m "Ajouter une nouvelle fonctionnalité"
```

Un commit en enregistre les modifications ajoutées à la zone de staging.

Les commits permettent de suivre l'historique des modifications, de revenir à des états antérieurs et de collaborer efficacement avec d'autres développeurs.

Git : historique des commits

git log ou git log --oneline :

Une commande utile pour visualiser l'historique des commits dans votre référentiel.

Elle permet de voir les modifications apportées, les auteurs et les dates des commits, ce qui facilite le suivi des modifications et la compréhension de l'évolution du code.

Git : se déplacer parmi les commits

La commande "git checkout" est utilisée pour basculer entre les commits spécifiques ou les étiquettes (tags) dans Git. Toutefois, l'utilisation de "git checkout" avec un identifiant n'est pas une pratique courante, car les identifiants (hash) des commits sont généralement difficiles à mémoriser :

```
git checkout [id-du-commit]
```

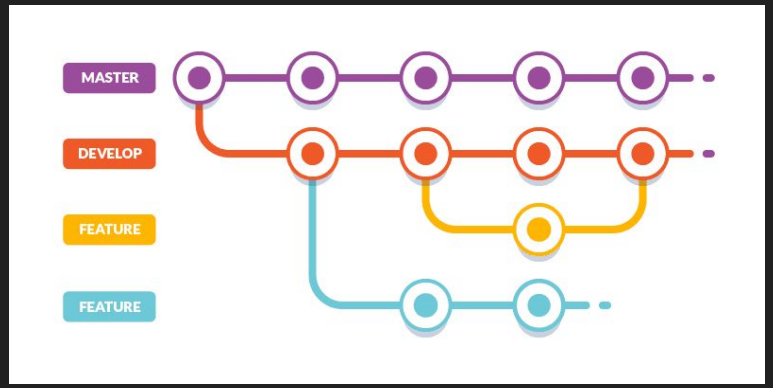
Pour retourner au dernier commit :

```
git checkout [nom-de-branche]
```

Branch

Git : workflow

Le flux de travail Git, également connu sous le nom de "git workflow" en anglais, se réfère à la façon dont les développeurs utilisent Git pour gérer leurs projets de manière collaborative.



Git : branch

Une branche (ou branch en anglais) dans Git est une référence vers une version spécifique de votre code source.

Elle permet de travailler sur différentes versions du projet de manière isolée, sans affecter la branche principale (généralement appelée "master" ou "main").

Git : branch en pratique

Lister toutes les branches :

```
git branch
```

Créer une nouvelle branche :

```
git branch [nom-de-branche]
```

Changer le nom de la branche :

```
git branch -m [nom-de-branche]
```

Basculer sur une branche spécifique :

```
git switch [nom-de-branche]
```

Créer et basculer sur une branche:

```
git switch -c [nom-de-branche]
```

Git : fusion de branche

La commande : `git merge`

Utilisée pour fusionner les modifications de différentes branches dans Git.

Elle permet d'intégrer les modifications d'une branche source (généralement une branche de fonctionnalité ou de correction) dans une branche cible.

On se déplace d’abord sur la branche qu’on veut fusionner avec une autre.

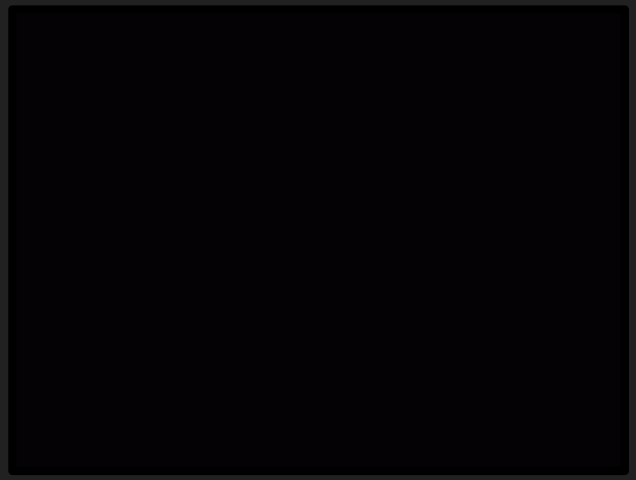
Git : supprimer une branch

Supprimer une branche dans Git :

```
git branch -d [nom-de-branche]
```

Git vérifiera si la branche que vous souhaitez supprimer a été fusionnée avec succès dans la branche actuelle.
Si la branche n'a pas été fusionnée, Git affichera un avertissement et ne supprimera pas la branche, à moins que vous n'utilisiez l'option -D (majuscule) pour forcer la suppression.

Tags



Git : taguer un commit

Les tags sont des marqueurs stables et immuables qui sont associés à un commit spécifique dans l'historique du projet. Ils sont souvent utilisés pour marquer des versions stables, des versions de publication ou des points de repère importants dans le développement du logiciel.

Lister les tags :

```
git tag
```

Créer une tag :

```
git tag [nom-de-tag] [commit-associé]
```

Afficher les détails d’un tag :

```
git show[nom-du-tag]
```

Dépôt

Git : cloner un dépôt distant

On peut créer une copie locale complète d'un dépôt distant Git.

Cela permet de récupérer l'intégralité de l'historique des commits, des branches et des fichiers du dépôt distant sur votre machine locale.

```
git clone [url-depot]
```

Git : connexion à un dépôt distant

Un dépôt distant dans Git fait référence à une copie du référentiel Git située sur un autre emplacement, généralement sur un serveur ou un autre ordinateur.

Les dépôts distants permettent de collaborer avec d'autres développeurs, de partager des modifications et de synchroniser le code source.

```
git remote [nom-depot] [url-depot]
```

Git : mettre à jour son dépôt local

La commande "git pull" est utilisée pour récupérer les dernières modifications d'un dépôt distant et les fusionner avec votre branche locale.

Elle combine les étapes de récupération (fetch) et de fusion (merge) en une seule commande.

La première fois : `git pull origin [nom-de-branche]`

Les fois suivantes : `git pull`

Git : envoyer ses modifications

La commande "git push" est utilisée pour envoyer vos modifications locales vers un dépôt distant.

Elle permet de mettre à jour le dépôt distant avec les derniers commits de votre branche locale.

La première fois : `git push -u origin main`

Les fois suivantes : `git push`

Sass : Lectures complémentaires

Consultez certains des liens pour mieux comprendre le fonctionnement de Git :

Git : Site officiel

Comprendre git en 7 minutes : Rappel de qq fonctions prédéfinies

Apprendre Git : Tuto vidéo de Grafikart

50 commandes Git : Tuto vidéo de Grafikart

git commit –m 'merci'