Prise en main de Git

Contrôle de version:

Un contrôle de version est une technique de prise en charge des modifications apportées aux fichiers

Système de contrôle de version:

Un système de contrôle de version est un programme qui enregistre l'évolution d’un fichier ou d’un ensemble de fichier au cours du temps de manière à ce qu’on puisse rappeler une version antérieure d’un fichier à tout moment.

Utilité d’un système de contrôle de version:

- Permet de revenir à une version précédente de votre code en cas de problème
- Permet de suivre l’évolution de votre code étape par étape
- Permet un meilleur travail collaboratif

Les outils de contrôle de version:

Il existe plusieurs outils de contrôle de version parmi lesquels nous avons:

Git

CVS

SVN

Etc.

Présentation de Git

Définition

Git est un système de contrôle de version open source et décentralisé qui a été créé par Linus Torvalds. Il se distingue par sa rapidité et sa gestion des branches qui permettent de développer en parallèle de nouvelles fonctionnalités.

Caractéristiques:

- Distribué (chaque développeur dispose d'une version local du code source)
- Approche peer-to-peer (chaque utilisateur est à la fois client et serveur)
- Basé sur des Branches
- Les opérations sont atomique (elles peuvent réussir ou échoué)
- Tout est stocké dans le dossier .git
- Utiliser une **zone de classement** ou **index** (dans l'index les développeurs peuvent formater les modifications et les faire réviser avant de les appliquer réellement)

Dépôt Git :

Un dépôt (ou repository en anglais) est un entrepôt virtuel de votre projet qui vous permet de conserver un historique des versions et des modifications d’un projet.

Il peut être **local** ou **distant**.

Dépôt local:

Un dépôt local stocke les différentes versions de notre code sur la machine locale. Toutes les modifications ne seront accessibles que localement.

Dépôt distant:

Un dépôt distant stocke les différentes versions directement sur Internet ou sur un réseau d’entreprise, on peut donc y accéder à distance.

On peut avoir plusieurs dépôts distant avec des droits différents (lecture seule, écriture, etc.).

Pour pouvoir collaborer avec d'autres personnes, les dépôts sont indispensables.

Parmi les dépôts distant le plus populaire est **Github.**

Notion de branches:

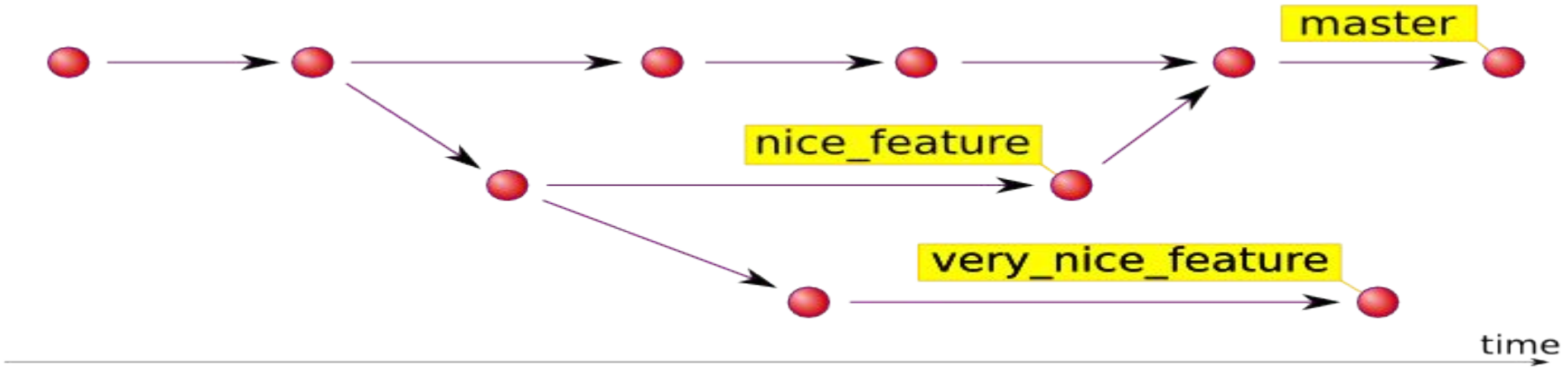
Chaque développeur peut travailler sur différentes parties d’un même projet. Par conséquent, il peut y avoir de nombreuses modifications différentes basées sur la même validation.

Git fournit des outils permettant d’isoler les modifications et de les fusionner ultérieurement. C’est là qu'intervient la notion de **Branches**.

Chaque développeur va donc travailler séparément sur sa branche et une fois le travaille terminé les branches seront fusionnées dans la branche principale.

Les branches sont donc comme des espaces de travail qui sont isolés les uns des autres.

De façon technique, une branche dans Git est simplement un pointer léger et déplaçable vers un de ces commits. La branche par défaut dans Git s'appelle **master**. Au fur et à mesure des validations, la branche **master** pointe vers le dernier des commits réalisés. A chaque validation, le pointeur de la branche master avance automatiquement.



Commandes essentiels:

- **git config:**

  Permet de voir et modifier les variables de configuration qui contrôlent tous les aspects de l'apparence et du comportement de Git.

  Exemple: `git config --global user.email sam@google.com`

- **git init:**

  Créer un nouveau dépôt local

  Exemple: `git init`

- **git add:**

  Ajouter un ou des fichiers à l'index.

  Exemple: `git add temp.txt`

- **git clone:**

  Clone un dépôt dans un nouveau repertoire

  Exemple: `git clone /chemin/vers/dépôt`

- **git commit:**

  Enregistrer les modifications dans le dépôt.

  Exemple: `git commit –m "Description du commit"`

- **git status:**

  Affiche la liste des fichiers modifiés ainsi que les fichiers qui doivent être commités.

  Exemple: `git status`

- **git push:**

  Envoie les modifications locales apportées à une branche dans le dépôt distant.

  Exemple: `git push origin main`

- **git branch:**

  Liste, crée, ou supprime des branches.

  Exemple: `git branch`

- **git checkout:**

  Bascule sur une autre branche

  Exemple: `git checkout <nom-branche>`

- **git remote:**

  Permet de se connecter à un dépôt distant.

  Exemple: `git remote add origin <chemin-depot-distant>`

- **git pull:**

  Fusionner toutes les modifications présentes sur le dépôt distant dans le dépôt local.

  Exemple: `git pull <depot-distant>s`

- **git merge:**

  Fusionner une branche dans la branche active

  Exemple: `git merge <nom-branche>`

- **git rm:**

  Supprimer des fichiers de l'index et du répertoire de travail.

  Exemple: `git rm nomfichier.py`

- **git reset:**

  Réinitialiser l'index et le répertoire de travail à l'état spécifié.

  Exemple: `git reset --hard HEAD`

- **gitk:**

  Affiche l'interface graphique du dépôt local

  Exemple: `gitk`

- **git rebase:**

  Réappliquer des commits sur une autre branche.

  Exemple: `git rebase master`

- **git fetch:**

  Télécharger tous les fichiers du dépôt distant qui ne sont pas actuellement dans le répertoire de travail local.

  Exemple: `git fetch origin`

- **git stash:**

  Enregistrer les changements qui ne doivent pas être commit immédiatement.

  Exemple: `git stash`

GitHub Cheat Sheet

Versionner son travail

Versionner en local

git init	initialise le dépôt (se mettre sur le bon dossier), mieux à faire depuis Github.com
git add .	ajoute toutes les modifications (le . symbolise tout)
git commit -m "explication"	créer un nouveau commit. **git add** pousse les fichiers en zone d'index, **git commit** les sauvegarde réellement dans un nouveau commit

Gérer les commits

git log	liste des commits
git log -n2	affiche les 2 derniers commits
git show sha-1	voir commit spécifique (cliquer molette souris pour coller)
git checkout sha-1	remettre la version du sha-1
git checkout main	remettre version la plus récente

Versionner sur un dépôt distant

git clone lien-github.com	récupérer travail depuis dépôt distant
git push -u origin main	pousse les modifications vers serveur
git push -f origin main	pousse de force des modifications (à manipuler avec précaution)

Naviguer dans Git Bash

pwd	savoir dans quel dossier je suis
mkdir "dossier"	créer un dossier (Make Directory)
touch fichier.txt	créer fichier
ls	liste le dossier courant
ls -la	liste tout plus précisément que ls
cd dossier	aller dans le dossier (Change Directory)
cd ..	Remonter d'un dossier

Initialisation de Git

git config --global user.name	"Mon Nom"
git config --global user.email	mon@mail.com
git config --global --list	Affiche nom et mail

Autres commandes

git status	état du fichier
git diff	affiche les modifs avant commit

Commit son projet sur Github

```
git add .
git commit -m "message"
git push -u origin main
```

Workflows

Un workflow Git est une recette ou une recommandation expliquant comment utiliser Git pour accomplir une tâche de façon cohérente et productive. Les workflows Git encouragent les développeurs et les équipes DevOps à exploiter Git de façon efficace et cohérente.

Installation

Télécharger le setup Git sur https://git-scm.com/downloads.

Une fois le téléchargement terminé on double-clique sur le setup et on suit les instructions pour finaliser l’installation.

Demo