SCM (Source Code Management) - Git

Les Fondations

C'est quoi un SCM ? (Le Concept)

Le **SCM** signifie **Source Code Management** (gestionnaire de code source) est une composante essentielle de la **Gestion de Configuration Logicielle** (GCL). Il ne s'agit pas d'un simple outil d'historique, mais d'un système normé garantissant le cycle de vie du code.

Les trois piliers du SCM :

1. **Versioning (Gestion de versions) :** Capacité à maintenir plusieurs états d'un même projet simultanément (ex: Production v1.0, Développement v1.1, Correctif de sécurité v1.0.1) sans duplication de fichiers physique.
2. **Traçabilité & Imputabilité :** Identification non-répudiable de l'auteur de chaque modification (Qui ?), de la date (Quand ?) et de la raison (Pourquoi ? - via le message de commit).
3. **Collaboration & Concurrence :** Gestion des accès simultanés sur les mêmes fichiers par plusieurs développeurs, avec des mécanismes de **résolution de conflits** et de **fusion (Merge)**.

> **Note Technique :** Dans un contexte DevOps/CI-CD, le SCM est la "Source de Vérité" unique. Il déclenche les pipelines d'intégration continue.

C'est quoi Git ? (L'Outil)

Git est un logiciel de SCM **Décentralisé** (DVCS). Créé par Linus Torvalds (le père de Linux), c'est aujourd'hui le standard absolu de l'industrie.

Contrairement à un système centralisé où l'historique réside uniquement sur un serveur maître, Git est un **DVCS (Distributed Version Control System)**.

- Chaque poste client (votre laptop) possède une copie **intégrale** et autonome du dépôt (Repository), incluant tout l'historique depuis la création.
- **Avantage :** Résilience totale (si le serveur distant tombe, n'importe quel client peut le restaurer) et capacité à travailler hors-ligne (commit local).

- **Git vs GitHub/GitLab :** Attention à la confusion courante.
  - **Git :** C'est le moteur sous le capot (installé sur votre laptop).
  - **GitHub/GitLab :** C'est le garage dans le Cloud où on gare le code pour le partager.

Intégrité des Données (SHA-1)

Git ne stocke pas des fichiers par leur nom, mais par leur contenu.

- Chaque objet (commit, fichier, dossier) est identifié par une **empreinte cryptographique unique** (Hash SHA-1) de 40 caractères hexadécimaux.
- **Sécurité :** Il est mathématiquement impossible de modifier un fichier ou un message de commit dans l'historique sans changer l'ID du commit et briser la chaîne de confiance. L'intégrité est native.

Snapshots vs Deltas

- **SCM Classique :** Stocke les différences entre les fichiers (Delta storage).
- **Git :** Stocke un **instantané (Snapshot)** complet du système de fichiers à chaque commit. Pour optimiser l'espace, si un fichier n'a pas changé, Git stocke simplement un lien symbolique vers la version précédente.

Installation & Configuration (L'Identité)

Avant de pouvoir dire "I know Kung Fu", il faut installer le dojo.

L'Installation

- **Windows :** Télécharge "Git for Windows" (Git Bash). C'est le standard. Lors de l'installation, laisse les options par défaut, sauf si tu sais exactement ce que tu fais (ce dont je doute fort pour l'instant).
- **Linux (Debian/Ubuntu) :** sudo apt install git (Simple, efficace, basique).
- **Mac :** git --version dans le terminal (S'il n'est pas là, macOS te proposera de l'installer).

La Configuration (Qui es-tu ?)

Git a besoin de savoir qui blâmer quand le code plantera. C'est l'équivalent de donner tes papiers à l'entrée du club.

Ouvre ton terminal (ou Git Bash) et tape ceci :

```
# Définir ton nom (Visible dans l'historique)
git config --global user.name "TonNom"

# Définir ton email (Doit correspondre à ton futur compte GitHub/GitLab)
git config --global user.email "ton.email@exemple.com"

# Vérifier que tu n'as pas fait de faute de frappe
git config --list
```

> **Note Culturelle :** "Identity theft is not a joke, Jim!" (*The Office*). Si tu ne configures pas ça, tes commits seront anonymes et sales.

Commandes & Syntaxe (Le Cycle de Vie)

Voici les commandes vitales. Considère-les comme tes fonctions primaires.

Commande	Explication
git init	Crée un dépôt Git dans le dossier actuel. La naissance de l'univers.
git status	L'état des lieux. Qu'est-ce qui a changé ? Qu'est-ce qui est cassé ?
git add .	Prépare TOUS les fichiers modifiés pour la sauvegarde (Staging Area).
git commit -m "msg"	Valide les changements. Crée un point de sauvegarde permanent.
git log	Affiche l'historique. La preuve de tes crimes passés.

Exemple de flux de travail (Workflow)

- Tu vas dans ton dossier de projet :

```
cd mon_super_site_wiki
```

- Tu inities la surveillance :

```
git init
```

- Tu codes, tu modifies ton HTML/CSS.
- Tu vérifies ce que tu as fait :

```
git status
```

(Tu verras tes fichiers en rouge : ils ne sont pas suivis).

- Tu les ajoutes à la zone de transit (Staging) :

```
git add .
```

- Tu graves ça dans le marbre :

```
git commit -m "Création de la structure HTML de base"
```

(Sois précis dans tes messages. Si je vois un commit nommé "fix truc", je formate ton disque).

Best Practices & Sécurité (Le .gitignore)

C'est ici que tu appliques la **Matrice Culturelle : L'ORDRE & LA SÉCURITÉ**.

Il y a des choses qu'on ne versionne **JAMAIS** : les fichiers temporaires, les dossiers de build (node_modules), et surtout ... les mots de passe ou clés API.

Pour cela, on crée un fichier spécial nommé .gitignore à la racine du projet.

Contenu exemple d'un .gitignore :

```
# Fichiers systèmes (Le bazar de l'OS)
.DS_Store
Thumbs.db

# Logs et bases de données locales
*.log
*.sqlite

# Dépendances (C'est lourd et inutile à stocker)
node_modules/
vendor/

# SECRETS (NE PAS TOUCHER, C'EST POUR MR. WOLF)
.env
config_secret.json
```

> **"Keep it secret. Keep it safe."** (*Le Seigneur des Anneaux*). Si tu envoies un mot de passe sur Git, considère qu'il est public. Gandalf ne pourra rien pour toi.

Audit & Debug (Quand tu as tout cassé)

Tu as fait un git add d'un fichier qu'il ne fallait pas ?

```
git restore --staged <fichier>
```

C'est le "Ctrl+Z" de la vraie vie.

Tu veux voir qui a écrit cette ligne de code horrible ?

```
git blame <fichier>
```

C'est souvent toi, il y a deux semaines. L'ironie est savoureuse.