Développer localement et introduction au versionning avec Git

Atelier : Développer localement et introduction au versionning avec Git (30 min)

Objectifs de l'atelier :

- Comprendre le workflow de développement local.
- Apprendre à utiliser les commandes de base de Git : init, add, commit, push.
- Mettre en pratique ces concepts en ajoutant un fichier CSS à un projet et en le versionnant avec Git.

Étapes de l'atelier :

1. Introduction au workflow de développement local

Le développement local fait référence à la création et la gestion de projets directement sur votre ordinateur sans la nécessité d'un serveur en ligne. Cela vous permet de tester votre code et d'apporter des modifications avant de les publier.

Voici les étapes classiques d'un workflow de développement local :

1. **Écrire du code** : Vous modifiez ou ajoutez des fichiers dans votre projet.
2. **Tester** : Vous vérifiez si les modifications apportées au code fonctionnent comme prévu en testant localement.
3. **Versionner avec Git** : Une fois satisfait de vos modifications, vous enregistrez ces changements dans l'historique de votre projet avec Git, ce qui permet de garder une trace de chaque étape du développement.
4. **Pousser sur un dépôt distant** : Une fois que le code est stable et fonctionnel, vous poussez vos modifications vers un dépôt distant (par exemple, sur GitHub) pour partager ou déployer le projet.

2. Commandes de base de Git

Les commandes de Git que vous allez utiliser dans cet atelier sont :

- **git init** : Initialise un nouveau dépôt Git local dans le dossier de votre projet.
- **git add** : Ajoute les fichiers modifiés ou nouveaux au suivi de Git.
- **git commit** : Enregistre les modifications dans l'historique du projet avec un message de commit.

- **git push** : Envoie les commits locaux vers un dépôt distant (par exemple, GitHub).

Exercice pratique : Ajouter un fichier styles.css et faire un commit

Étape 1 : Ajouter un fichier styles.css au projet

1. **Créez le fichier styles.css**
   Dans votre éditeur VS Code, créez un nouveau fichier nommé styles.css. Ce fichier contiendra le style de base pour votre projet. Voici un exemple de contenu pour le fichier styles.css :

```
body {
  font-family: Arial, sans-serif;
  background-color: #f4f4f4;
  margin: 0;
  padding: 0;
}
h1 {
  color: #333;
  text-align: center;
}
```

2. **Lier le fichier CSS à votre index.html**
   Ouvrez votre fichier index.html et ajoutez une balise <link> pour lier le fichier styles.css à la page HTML. Placez cette balise dans la section <head> de votre fichier HTML :

```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Mon Projet Web</title>
  <link rel="stylesheet" href="styles.css">
</head>
```

3. **Sauvegarder les fichiers**
   Sauvegardez les fichiers index.html et styles.css après avoir apporté ces modifications.

Étape 2 : Versionner les fichiers avec Git

1. **Vérifier l'état du projet avec Git**
   Ouvrez le terminal dans VS Code et tapez la commande suivante pour voir les fichiers modifiés ou non suivis par Git :

git status

2. **Ajouter les fichiers modifiés au suivi de Git**
   Une fois que vous avez modifié ou ajouté de nouveaux fichiers, vous devez les ajouter au suivi de Git avec la commande git add. Dans ce cas, vous ajoutez les fichiers index.html et styles.css :

git add index.html styles.css

3. **Faire un commit avec un message explicatif**
   Après avoir ajouté les fichiers, vous pouvez enregistrer ces modifications dans Git avec la commande git commit. Assurez-vous d'ajouter un message expliquant ce que vous avez fait dans ce commit :

git commit -m "Ajout du fichier styles.css et liaison avec index.html"

4. **Vérifier les commits**
   Pour vérifier les commits effectués, vous pouvez utiliser la commande :

git log

Cela vous affichera l'historique des commits avec les messages associés.

Étape 3: configurer le git local pour se connecter sur votre repository a distance

Supprimer les informations d'identification globales (nom d'utilisateur et email) :
Ces commandes suppriment les informations de connexion que Git utilise pour identifier l'utilisateur.

git config --global --unset user.name

git config --global --unset user.email

Supprimer les informations de connexion enregistrées (par exemple, GitHub) : Si vous avez utilisé un gestionnaire de mots de passe pour stocker vos informations de connexion (par exemple, git-credential-manager ou credential-cache), vous pouvez vider ces informations :

Pour vider le cache des informations d'identification :

git credential-cache exit

Pour se connecter à Git (en configurant un utilisateur global) :

Configurer le nom d'utilisateur et l'email : Ces commandes définissent le nom d'utilisateur et l'email que Git utilisera pour toutes les actions futures :

git config --global user.name "Votre Nom"

git config --global user.email "votre.email@example.com"

Afficher les informations de configuration globale :

Cette commande vous permettra de voir le nom d'utilisateur et l'email globalement configurés pour Git :

git config --global --list

Cela affichera quelque chose comme :

user.name=Votre Nom

user.email=votre.email@example.com

2.Afficher les informations de configuration locale (pour un projet spécifique) :

Si vous voulez voir les informations de configuration Git pour un projet spécifique (et non globalement), exécutez cette commande dans le répertoire de votre projet :

git config --list

Étape 3 : Pousser les modifications sur un dépôt distant

Si vous avez configuré un dépôt distant (par exemple, sur GitHub), vous pouvez pousser vos commits locaux vers ce dépôt pour partager vos modifications. Voici comment procéder :

1. Ajouter un dépôt distant (si ce n'est pas déjà fait)

Avant d'ajouter un dépôt distant à votre projet local, vous devez d'abord créer un dépôt sur GitHub.

1. **Allez sur GitHub** et connectez-vous à votre compte.
2. **Créez un nouveau dépôt** :
    - Cliquez sur le bouton **"New"** en haut à droite de votre tableau de bord.
    - Donnez un nom à votre dépôt (par exemple, mon-projet-web).
    - Choisissez si vous souhaitez qu'il soit public ou privé.
    - N'ajoutez pas de fichier README ni de licence pour l'instant (vous les ajouterez plus tard si nécessaire).
    - Cliquez sur **"Create repository"**.

Vous serez redirigé vers la page de votre nouveau dépôt, qui contient l'URL pour le dépôt distant. Exemple : **https://github.com/kbma/mon-projet-web.git**

```
...or create a new repository on the command line
echo "# mon-projet-web" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/kbma/mon-projet-web.git
git push -u origin main

...or push an existing repository from the command line
git remote add origin https://github.com/kbma/mon-projet-web.git
git branch -M main
git push -u origin main
```

explication:

git remote add origin <URL>

2. **Pousser les modifications vers le dépôt distant**
   Pour envoyer vos commits locaux sur le dépôt distant, utilisez la commande git push :

git push origin main

Cette commande pousse votre branche locale (main) vers le dépôt distant. Vous serez invité à entrer vos informations d'identification GitHub si c'est la première fois.

Exercices supplémentaires :

1. **Exercice 1 : Modifier le fichier CSS**
   Ajoutez un nouveau style dans le fichier styles.css. Par exemple, ajoutez une règle pour centrer un texte dans le body :

```
body {
  text-align: center;
}
```

Ensuite, faites un commit pour versionner cette modification.

git add .

git commit -m "Nodification dans le fichier styles.css"

git push

2. **Exercice 2 : Ajouter une nouvelle section HTML**
   Ajoutez une nouvelle section dans le fichier index.html (par exemple, une section avec un sous-titre). Par exemple :

```
<section>
  <h2>Présentation</h2>
  <p>Bienvenue sur mon projet !</p>
</section>
```

Après avoir ajouté cette section, faites un commit pour versionner cette modification.

git add .

git commit -m "Ajouter la section présentation"

git push

Pour visualiser votre site web (uniquement la partie statique)

Activer GitHub Pages :

- Va dans les paramètres de ton dépôt GitHub (Settings).
- Dans la section "**Pages**", sous "Source", sélectionne la branche **`main`** ou `master`, puis choisis le dossier **`root`** ou `docs` si tu veux un répertoire spécifique.
- Clique sur "Save".

Accéder à ton site :

- Après quelques minutes, GitHub Pages va générer ton site. L'URL de ton site sera généralement du format
  `https://<ton_nom_utilisateur>.github.io/<nom_de_repositoire>/`
- Tu peux accéder à ton site en utilisant cette URL. Exemple :
  https://kbma.github.io/mon-projet-web/

Conclusion :

Cet atelier vous a permis de comprendre comment développer localement en utilisant Git pour versionner vos projets. Vous avez appris à ajouter des fichiers à votre projet, à les versionner à l'aide des commandes git add et git commit, et à partager vos modifications en les poussant vers un dépôt distant avec git push. Ces pratiques sont essentielles pour travailler efficacement sur vos projets web, que ce soit de manière indépendante ou en collaboration avec d'autres développeurs.