Cours GitHub (en 3 pages)

Introduction à Git et GitHub

1. Qu'est-ce que Git ?

Git est un système de contrôle de version décentralisé permettant aux développeurs de suivre les modifications apportées à un fichier ou à un projet de code source au fil du temps. Contrairement aux systèmes de contrôle de version centralisés, Git permet à chaque développeur d'avoir une copie complète de l'historique du projet sur sa propre machine. Cela facilite le travail collaboratif en permettant à chacun de travailler indépendamment et de fusionner ensuite ses modifications avec celles des autres.

Fonctionnement de Git :

- **Commits** : Chaque changement enregistré dans un projet Git est appelé un commit. Chaque commit contient un message qui décrit les changements effectués.
- **Branches** : Git permet de créer des branches pour expérimenter, créer de nouvelles fonctionnalités ou corriger des bugs sans affecter le code principal.
- **Fusion (merge)** : Lorsque le travail sur une branche est terminé, il peut être fusionné avec la branche principale (souvent appelée main ou master).

2. Qu'est-ce que GitHub ?

GitHub est une plateforme en ligne qui repose sur Git et permet de stocker, gérer et partager des projets de développement. GitHub offre également des fonctionnalités supplémentaires telles que la gestion de projet, le suivi des issues (problèmes), les pull requests et la collaboration avec d'autres développeurs à travers le monde.

Principales fonctionnalités de GitHub :

- **Repos (Dépôts)** : Un dépôt GitHub est un projet ou un ensemble de fichiers stockés et gérés via Git. Il peut être public ou privé.
- **Issues** : Outil pour suivre les bugs, les tâches et les améliorations.
- **Pull Requests (PR)** : Permet aux utilisateurs de proposer des modifications au code d'un autre dépôt. Les mainteneurs du dépôt peuvent réviser, discuter et fusionner les modifications.
- **Actions GitHub** : Automatisation des workflows pour tester, déployer ou effectuer d'autres tâches à chaque modification du code.

---

3. Installation de Git et création d'un compte GitHub

3.1 Installation de Git

1. **Télécharger Git** : Allez sur le site git-scm.com et téléchargez la version de Git correspondant à votre système d'exploitation (Windows, macOS, Linux).
2. **Installation** : Suivez les instructions d'installation pour votre OS.
3. **Vérification** : Ouvrez un terminal et tapez git --version pour vérifier que Git est installé correctement.

3.2 Créer un compte GitHub

1. Rendez-vous sur github.com.
2. Cliquez sur "Sign Up" pour créer un compte.
3. Suivez les étapes pour créer un compte avec un email et un mot de passe sécurisé.
4. Vous pouvez également configurer une clé SSH pour une connexion plus sécurisée entre votre machine locale et GitHub.

---

4. Premiers pas avec GitHub

4.1 Création d'un dépôt GitHub

1. Connectez-vous à votre compte GitHub.
2. Cliquez sur le bouton "New repository" sur votre page d'accueil ou depuis la barre latérale.
3. Nommez votre dépôt (par exemple, "mon-projet").
4. Choisissez si vous voulez que le dépôt soit public ou privé.
5. Cliquez sur "Create repository".

GitHub vous fournira alors les commandes à exécuter pour initialiser votre projet local et le lier à GitHub.

4.2 Cloner un dépôt GitHub

Si vous souhaitez travailler sur un dépôt GitHub existant, vous devez le cloner. Cela crée une copie locale du dépôt sur votre machine.

1. Accédez au dépôt GitHub que vous souhaitez cloner.
2. Cliquez sur le bouton "Code" puis copiez l'URL du dépôt (HTTPS ou SSH).
3. Ouvrez un terminal et tapez la commande suivante :

bash

Copier

git clone <url-du-dépôt>

Cela crée une copie locale du dépôt sur votre machine.

4.3 Ajouter et modifier des fichiers dans un dépôt local

1. **Ajouter un fichier** : Créez ou modifiez des fichiers dans le dossier du projet cloné.
2. **Ajouter les changements à Git** : Après avoir modifié ou ajouté des fichiers, vous devez les ajouter à l'index de Git.

bash

Copier

git add <nom-du-fichier>

Pour ajouter tous les fichiers modifiés, vous pouvez utiliser git add ..

3. **Faire un commit** : Un commit permet de sauvegarder les modifications.

bash

Copier

git commit -m "Message décrivant les changements"

4.4 Pousser les modifications sur GitHub

1. Une fois que vos changements sont prêts, vous pouvez les envoyer sur GitHub.

bash

Copier

git push origin main

La commande git push envoie vos commits locaux vers le dépôt distant sur GitHub. origin désigne l'URL du dépôt distant et main est la branche par défaut.

---

5. Collaborer sur GitHub

5.1 Travailler avec des branches

Les branches permettent de travailler sur différentes fonctionnalités ou corrections sans affecter la branche principale.

1. **Créer une branche** :

bash

Copier

git checkout -b nom-de-la-branche

2. **Passer d'une branche à une autre** :

bash

Copier

git checkout main

Pour revenir à la branche principale ou à une autre branche, utilisez git checkout suivi du nom de la branche.

3. **Fusionner une branche** : Une fois votre travail terminé sur une branche, vous pouvez la fusionner dans la branche principale :

bash

Copier

git checkout main

git merge nom-de-la-branche

5.2 Pull Request (PR)

Lorsque vous travaillez sur un projet avec plusieurs personnes, vous pouvez proposer vos modifications à l'aide d'une Pull Request. C'est un mécanisme sur GitHub qui permet de demander la révision et la fusion de vos changements avec la branche principale du dépôt.

1. **Créer une Pull Request** :
    - Allez sur la page de votre dépôt sur GitHub.
    - Cliquez sur "Pull Requests" puis sur "New Pull Request".
    - Sélectionnez la branche que vous souhaitez fusionner avec la branche principale.
    - Ajoutez une description et cliquez sur "Create Pull Request".
2. **Revue et fusion** : Le propriétaire du dépôt ou un collaborateur pourra réviser votre PR, laisser des commentaires, et, une fois validée, fusionner vos modifications.

---

6. Bonnes pratiques et sécurité

6.1 Bonnes pratiques

- **Commits fréquents et clairs** : Faites des commits fréquents avec des messages explicites pour expliquer les changements.
- **Branchez pour chaque fonctionnalité** : Créez une nouvelle branche pour chaque fonctionnalité ou correction afin de garder une bonne organisation.
- **Revue de code** : Utilisez les Pull Requests pour effectuer une revue de code et améliorer la qualité du code avant de le fusionner.

6.2 Sécurité sur GitHub

- **Authentification à deux facteurs (2FA)** : Activez la 2FA sur votre compte GitHub pour ajouter une couche de sécurité.
- **Clés SSH** : Utilisez des clés SSH pour vous connecter à GitHub de manière plus sécurisée que via un mot de passe.
- **Secrets et variables d'environnement** : Ne stockez jamais d'informations sensibles comme des mots de passe ou des clés API dans votre dépôt. Utilisez plutôt les fonctionnalités de GitHub pour gérer les secrets de manière sécurisée.

Conclusion

Git et GitHub sont des outils incontournables pour les développeurs travaillant en équipe. Git permet de gérer les versions de manière locale, tandis que GitHub facilite la collaboration, le partage et la gestion des projets de manière centralisée. En suivant les bonnes pratiques et en vous familiarisant

avec les commandes essentielles, vous pourrez travailler efficacement sur vos projets et collaborer avec d'autres développeurs à travers le monde.