📖

Sommaire : Cours sur Git

N.B : + ajout personnes sur projet + interfaces + gitignore + pb de merge

1. Introduction à Git

1.1 Qu'est-ce que Git ?

1.2 L'histoire de Git et ses objectifs (gestion de version, collaboration, etc.)

1.3 Git vs autres systèmes de gestion de version (Subversion, Mercurial, etc.)

1.4 Installation de Git (Windows, macOS, Linux)

1.5 Configurer Git pour la première fois (nom, email, alias)

- Commandes :

```
git config --global user.name "Votre Nom"
git config --global user.email "votre.email@example.com"
```

---

2. Les bases de Git

2.1 Créer un dépôt Git local

- Commandes : `git init` 2.2 Cloner un dépôt distant
- Commandes : `git clone URL` 2.3 Les trois zones de Git :
- **Working Directory** : fichiers modifiés.
- **Staging Area** : fichiers ajoutés avec `git add`.
- **Commit History** : fichiers validés avec `git commit` .2.4 Les commandes de base :
- Vérifier l'état : `git status`
- Ajouter des fichiers : `git add fichier`
- Faire un commit : `git commit -m "Message"`
- Historique des commits : `git log`

---

3. Travailler avec des fichiers

3.1 Suivre et ignorer des fichiers

- `.gitignore` : principes et exemples
- Exclure des fichiers temporairement avec `.git/info/exclude` 3.2 Renommer et déplacer des fichiers
- Commande : `git mv ancien_nom nouveau_nom` 3.3 Supprimer des fichiers du dépôt
- Commande : `git rm fichier`

---

4. Les branches dans Git

4.1 Qu'est-ce qu'une branche ?

4.2 Commandes essentielles :

- Créer une branche : `git branch nom`
- Basculer entre les branches : `git checkout` ou `git switch`
- Supprimer une branche : `git branch -d` 4.3 Fusionner des branches :
- Commandes : `git merge`
- Résolution des conflits4.4 Rebaser une branche : `git rebase` 4.5 Stratégies de branches : Git Flow, Trunk-Based Development

---

5. Collaborer avec Git

5.1 Dépôts distants (GitHub, GitLab, Bitbucket)

5.2 Ajouter et gérer des remotes :

- Ajouter un remote : `git remote add origin URL`
- Lister les remotes : `git remote -v`
- Supprimer ou renommer un remote5.3 Pousser et tirer des changements :
- `git push` : envoyer des commits
- `git pull` : récupérer les modifications5.4 Suivre une branche distante :
- `git checkout --track origin/branche`

---

6. Résolution des conflits

6.1 Pourquoi les conflits arrivent-ils ?

6.2 Identifier les conflits : `git status`

6.3 Résoudre les conflits manuellement (outils intégrés ou IDE)

6.4 Valider après résolution :

```
git add fichier_conflit
git commit
```

---

7. Gestion de l'historique

7.1 Afficher l'historique :

- `git log`, `git log --oneline`, `git log --graph` 7.2 Annuler des changements :
- Annuler un commit local : `git reset` ou `git revert`
- Revenir à une version précédente : `git checkout SHA` 7.3 Nettoyer le dépôt :
- Supprimer les fichiers inutilisés : `git clean`

---

8. Commandes avancées et bonnes pratiques

8.1 Réécriture de l'historique avec `git rebase`

8.2 Cherry-picking : appliquer un commit spécifique sur une autre branche (`git cherry-pick`)

8.3 Stash : sauvegarder des changements temporaires (`git stash`)

8.4 Tags : marquer des commits importants

- Créer un tag : `git tag -a v1.0 -m "Version 1.0"`
- Pousser un tag : `git push origin --tags`

---

9. Git et outils tiers

9.1 Utilisation de Git avec un IDE (VS Code, IntelliJ, etc.)

9.2 Intégration avec GitHub/GitLab :

- Pull Requests / Merge Requests
- CI/CD (déploiement continu)9.3 Gestion des secrets avec `.env` et outils comme `git-secrets`9.4 Générer un fichier `.gitignore` via gitignore.io

---

10. Bonnes pratiques et astuces

10.1 Structurer les messages de commit (exemple avec convention Angular)

10.2 Utiliser les branches courtes et les merges réguliers

10.3 Protéger la branche principale avec des règles (GitHub/GitLab)

10.4 Automatiser les workflows avec des hooks Git

---

11. Dépannage et débogage

11.1 Comprendre les erreurs courantes (`detached HEAD`, conflits de merge, etc.)

11.2 Récupérer des commits perdus avec `git reflog`

11.3 Déboguer avec `git bisect`

---

12. Cas pratiques

12.1 Travailler en équipe sur un projet

12.2 Gérer un hotfix en urgence

12.3 Rebaser une branche pour garder un historique propre

12.4 Restaurer un fichier supprimé par erreur