



Module 2

Contrôle de Version avec Azure Repos



Pourquoi le Contrôle de Version ?

Le contrôle de version est essentiel pour suivre les modifications du code, collaborer efficacement en équipe et gérer les différentes versions d'un projet logiciel.



Revenir à des versions
antérieures



Collaboration en équipe



Fusionner le travail



Éviter les conflits

Git : Les Fondamentaux



Dépôts (Repositories) : Centralisés vs Distribués



Working Directory, Staging Area, Local Repository



Commits : Enregistrement des modifications










Branches : Développement parallèle



Merge et Rebase : Intégration des branches

Flux de Travail Git Typique

- 1  Cloner le dépôt
- 2  Créer une nouvelle branche pour les fonctionnalités
- 3  Effectuer des modifications et commiter
- 4  Pousser la branche vers le dépôt distant
- 5  Créer une Pull Request
- 6  Revue de code et approbation
- 7  Fusionner la Pull Request

Stratégies de Branches



GitFlow

Branches principales (master, develop) et branches de support (feature, release, hotfix)



GitHub Flow

Simple, basé sur une branche principale et des branches de fonctionnalités



GitLab Flow

Similaire à GitHub Flow, avec des branches d'environnement



Choisir la stratégie adaptée à votre équipe et projet

Pull Requests



Proposer des modifications de code



Faciliter la revue de code par les pairs



Assurer la qualité et la conformité du
code



Fusionner les modifications dans la
branche principale

Revue de Code



Détecter les erreurs et les bugs tôt



Partager les connaissances et les
bonnes pratiques



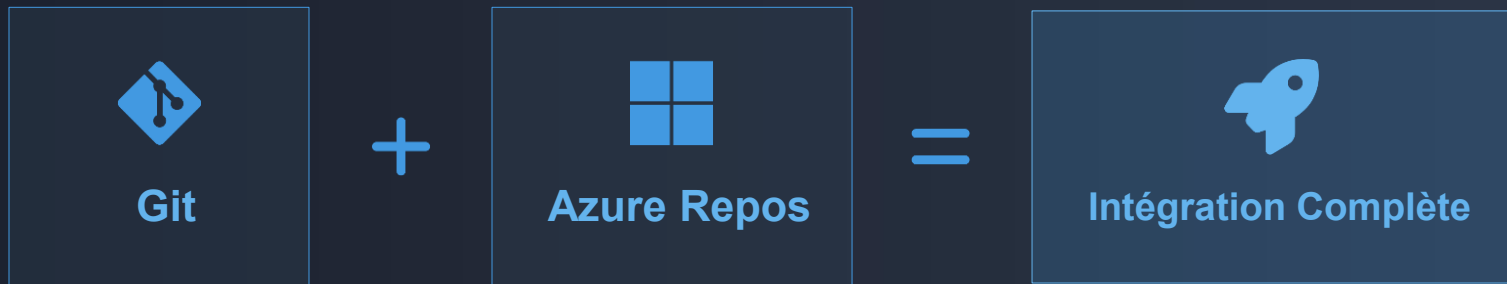
Améliorer la qualité et la
maintenabilité du code



Favoriser la collaboration et le
mentorat

Intégration Git dans Azure Repos

Azure Repos offre une intégration native et complète avec Git, permettant aux équipes de tirer parti de toutes les fonctionnalités de Git tout en bénéficiant des capacités de gestion de projet d'Azure DevOps.



Azure Repos : Fonctionnalités



Dépôts Git

Hébergement de dépôts Git privés et publics



Pull Requests

Processus de revue de code et de fusion sécurisé



Politiques de Branches

Application de règles pour garantir la qualité du code



Intégration

Intégration transparente avec Azure Pipelines et Boards

Sécurité des Dépôts



Contrôle d'accès basé sur les rôles (RBAC)



Politiques de branches pour la protection du code



Gestion des identités et des secrets



Analyse des vulnérabilités

Migration vers Azure Repos



Migration de TFVC vers Git



Migration depuis d'autres systèmes Git (GitHub, GitLab)



Outils et stratégies de migration



Bonnes pratiques post-migration

Git Avancé



Rebase interactif



Git Reflog



Tags et Releases



Submodules et Subtrees

Activité Pratique 1 : Dépôt Git

Objectif :

Créer et gérer un dépôt Git dans Azure Repos.

Étapes :

- 1 Créer un nouveau dépôt Git dans Azure Repos
- 2 Cloner le dépôt localement
- 3 Effectuer des modifications, commiter et pousser le code

Activité Pratique 2 : Politiques de Branches

Objectif :

Mettre en œuvre des politiques de branches pour garantir la qualité du code.

Étapes :

- 1 Configurer une politique de branche pour exiger une revue de code
- 2 Exiger la réussite des builds avant la fusion
- 3 Tester la politique en tentant de fusionner sans respecter les règles

Récapitulatif du Module 2



Principes du contrôle de version avec Git



Fonctionnalités et intégration d'Azure Repos



Stratégies de branches et Pull Requests



Sécurité des dépôts et revue de code



Questions & Réponses

N'hésitez pas à poser vos questions !