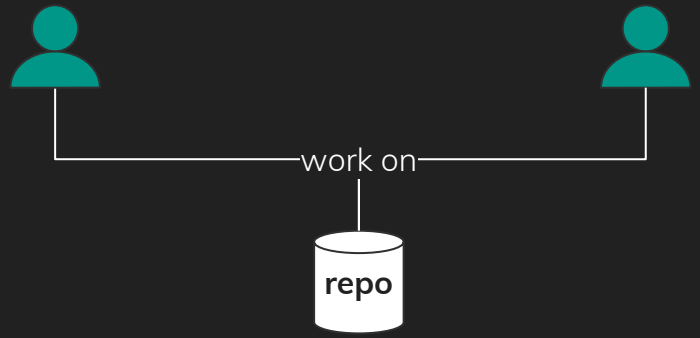
Mise en production

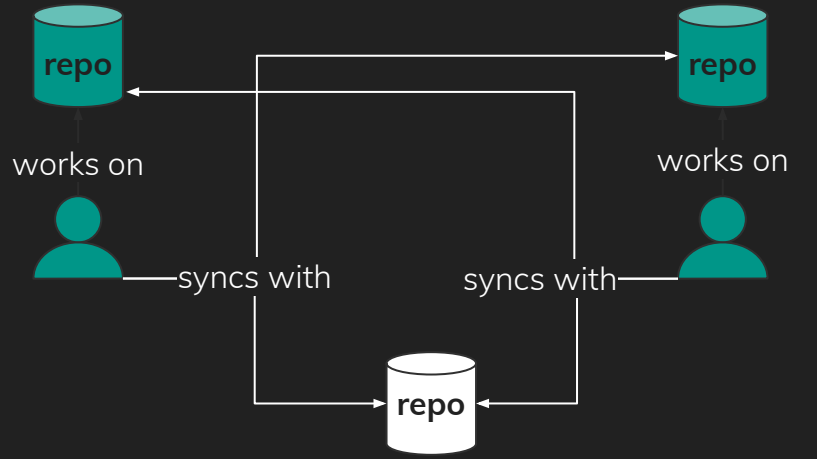
SCM
Source Code Management

Concepts

- Centralisé: Les utilisateurs interagissent directement avec un repository central sur le reseau.



- Décentralisé: Les utilisateurs travaillent en local, sur un repository complément séparé, et peuvent se synchroniser sur d'autres repositories sur le réseau. Il peut y avoir d'autres utilisateurs. Il peut y avoir d'autres repositories appartenant à des utilisateurs et un central.



git

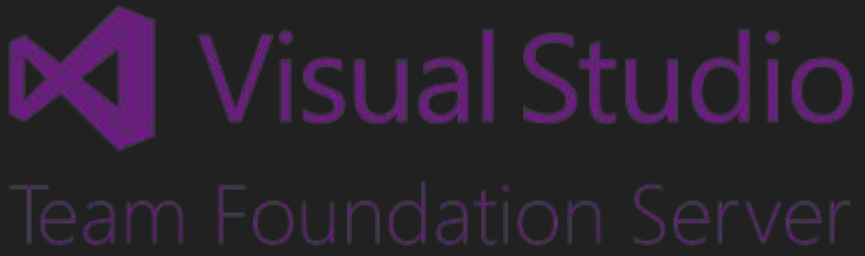
- SCM le plus utilisé
- Modèle: décentralisé
- Concurrence: patch
- License: GPL-2 (Free Software)

Subversion

- Encore beaucoup utilisé
- Modèle: décentralisé
- Concurrence: file
- Licence: Apache 2.0 (Free Software)

Team Foundation Version Control (TFVC)

- Environnement Microsoft & Azure DevOps
- Modèle: Distribué ou centralisé
- Concurrence: file
- Licence: propriétaire



Mercurial

- Créé pour les mêmes raisons que git mais n’a pas gagné
- Modèle: Distribué
- Concurrence: patch
- License: GPL-2 or later (Free Software)

Focus sur Git

Histoire Courte

- Créé en 2005 par Linus Torvalds pour les besoins du Linux Kernel quand BitKeeper est devenu payant
- Maintenu principalement par Junio Hamano depuis 2005
- SCM le plus utilisé

Principal caractéristiques et forces

- Modèle: Décentralisé
- Concurrence: patch
- C’est un filesystem (.git)
- Les commits sont immutable
- Historique peut être réécrit

CI/CD

CI
Continuous Intégration

Les concepts de la Continuous Integration

Git comme outil de SCM pour partager et collaborer sur du code

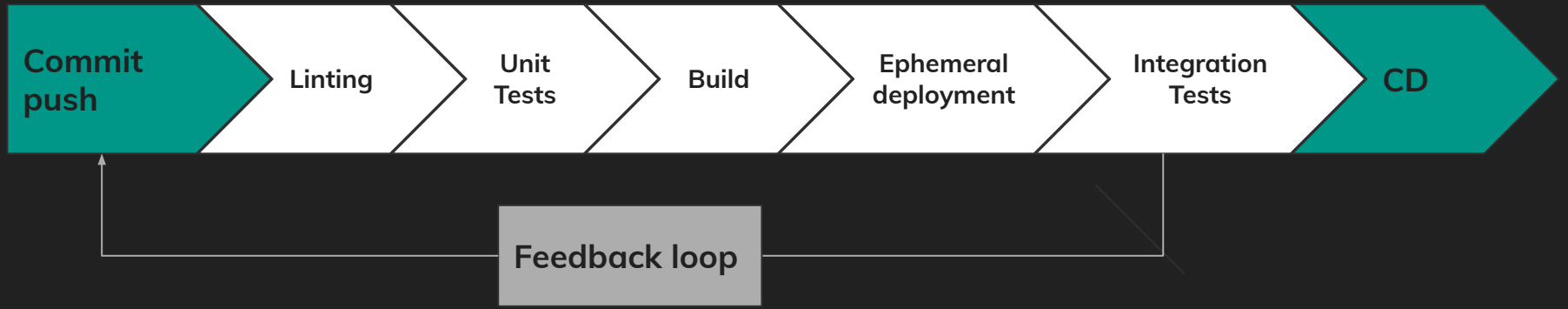
Comment garantir la qualité du code produit par les contributeurs ?

Comment déterminer si un ensemble de contributions est prêt à être déployé ?

Les concepts de la Continuous Integration

CI permet de:

- Mettre toutes les contributions sur le code, évaluer le contenu, et préparer la livraison.
- Réduire la durée de la boucle de feedback grâce à l'automatisation !



CD
Continuous Delivery / Deployment

Les concepts du Continuous Delivery et du Continuous Deployment

Que voulons-nous déployer ?

Quand voulez nous le déployer ?

Comment allons-nous le déployer ?

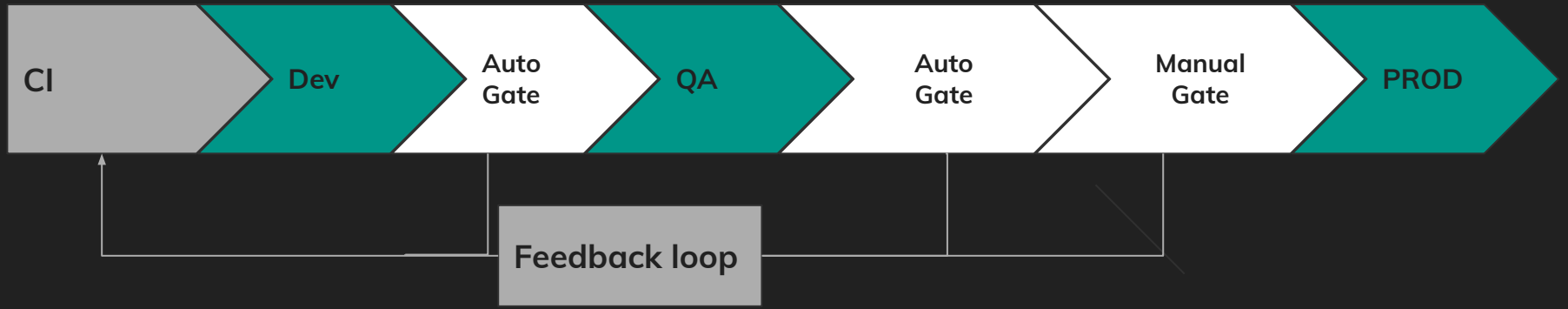
Dans quels environnements ?

En production ?

Les concepts du Continuous Delivery et du Continuous Deployment

CD permet de:

- Déployer notre livrable dans différents environnements
- Évaluer les gates pour déployer vers l'environnement suivant (promotion)
- Réduire les boucles de feedbacks avec l'automatisation pour corriger un maximum de bugs avant la mise en prod (MEP)



Circle CI

Drone.io

Gitlab CI

Jenkins

Github Actions