GIT

Par Martin Dubois, ing.

Plan de la présentation

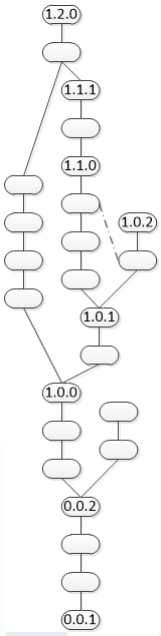
- Présentation du formateur et des participants
- Qu'est-ce qu'un logiciel de contrôle de version
- L'évolution des logiciels de contrôle de version
- Comment fonctionne GIT?
- Comment utiliser GIT?
- Les fonctionnalités avancées
- Autres formations offertes

Partie 1

QU'EST-CE QU'UN LOGICIEL DE CONTRÔLE DE VERSION

Le problème de départ

- Conserver toutes les versions des sources d'un logiciel
- Pourquoi?
  - Permettre de revenir en arrière si une erreur est commise
  - Permettre de corriger un « bug » dans une version antérieure et produire une mise à jour introduisant un minimum de changement
  - Garder l'historique!

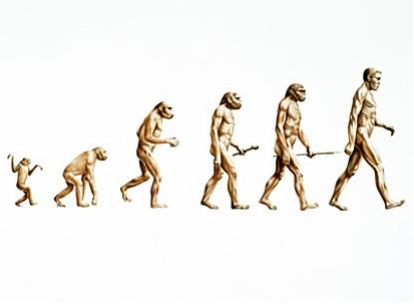


L'arbre de sources

- Après la version 0.0.2, un développeur à une idée géniale, mais il n'est pas certain de terminer à temps pour l'inclure dans la version 1.0.0. Il crée donc une branche de travail. Finalement, l'idée ne s'avère pas si géniale.
- Après la version 1.0.0, une partie de l'équipe s'attaque à une fonctionnalité importante et ne veut pas déranger la partie de l'équipe s'occupant de la maintenant. Ils créent donc une branche de travail.
- Un client découvre un mauvais fonctionnement dans la version 1.0.1 et ne désire pas installer une version plus récente et exige une correction « immédiate ». Un développeur crée une branche de maintenance fixe l'anomalie et publie une version 1.0.2. Il transporte aussi sa modification sur la branche principale pour éviter que la prochaine version souffre du même problème.
- La fonctionnalité importante prend du temps et des clients impatients demandent des améliorations mineures. La version 1.1.0 et 1.1.1 répond à ces requêtes.
- La fonctionnalité importante est enfin prête, ce qui permet de publier la version 1.2.0.

Les règles de base

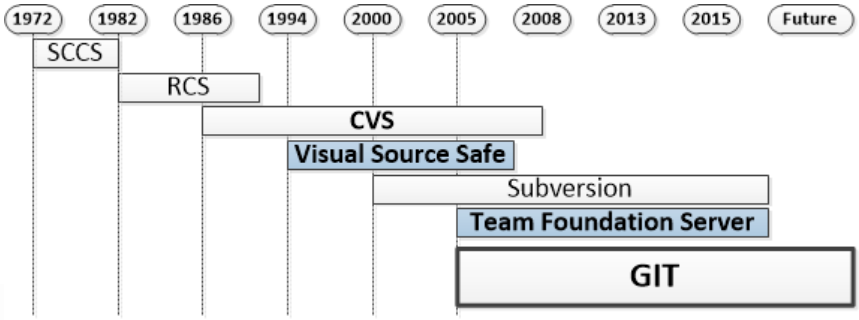
- Ne **jamais** placer des **outils de développement** dans un système de contrôle de version.
- Ne **jamais** placer des **fichiers temporaires** dans un système de contrôle de version.
- Ne **jamais** placer des **exécutables**, ou d'autres **fichiers générés** à partir des sources, dans un système de contrôle de version.
- **Éviter** de placer des **fichiers binaires** dans un système de contrôle des sources.
- **Attendre qu'une branche soit nécessaire avant de la créer**.



Partie 2

ÉVOLUTION DES LOGICIELS DE CONTRÔLE DE VERSION

Évolution des logiciels de contrôle de version



RCS

- 1982 au milieu des années 1990
- Travail sur des **fichiers**
- Sauvegarde **locale** des différentes versions
- Un **utilisateur unique** sur un **ordinateur isolé**

- L'arrivée des **réseaux** et des **équipes de développement** cause sa perte!

CVS

- 1986 au milieu des années 2000
- Travaille aussi sur des fichiers
- Architecture « Client – Serveur »
- L'équivalent « Microsoft » est « Visual Source Safe » (2000 au milieu des années 2000)

- C'est la complexité des projets logiciels qui a causé sa perte! Travailler sur des fichiers est devenue limitatif.

Subversion

- 2000 à maintenant
- Travail sur un projet logiciel
- Architecture « Client – Serveur »
- L'équivalent Microsoft est « Team Foundation Server » (2005 à maintenant)

- C'est l'« Open Source » qui a causé sa perte. Maintenant, plusieurs équipes participent à un même projet. Un serveur central est devenu limitatif.

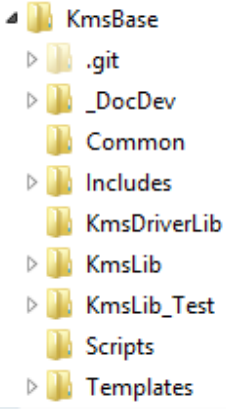
GIT

- Depuis 2005
- Travail sur un projet logiciel
- Architecture « distribuée »
- Microsoft supporte l'intégration de GIT avec Visual Studio à partir de 2013

Partie 3

COMMENT FONCTIONNE GIT?

Comment fonctionne GIT?



- Le répertoire « .git » contient le **dépôt** (repository)
- Le dépôt contient
  - des différences entre les versions de fichiers
  - des pointeurs (branche, tag)
  - des fichiers de configuration

Nous pourrions nous croire de retour à l'époque de RCS!

Mais non!

GIT est un système distribué!