Gestion de projet : Git

Arnaud Giersch

BUT Informatique – S2
2026

Plan

Système de gestion de versions

Définition

Permet de stocker un ensemble de fichiers en conservant l'historique des modifications.
Utilisé en général pour un ensemble de fichiers texte (codes sources en particulier).

Propriétés

- liste des modifications (différences entre les versions)
- auteur
- date
- message associé à chaque modification
- etc.

Intérêts

- Conservation de l'histoire d'un projet (corrections de bugs par ex.)
- Gestion des conflits :
  - développement à plusieurs
  - développement sur plusieurs machines/systèmes
- Obtention aisée de la dernière version
- Récupération d'une version précédente
- Étiquetage des versions
- Gestion de branches : stable, devel, experimental, etc.

Historique

Classification par mode de fonctionnement.

Local

Dépôt local au projet.

- Ex. : SCCS (1972), RCS (1982)

Centralisé

Dépôt sur un serveur central, accessible par le réseau.

- Ex. : CVS (1990), Subversion/SVN (2000)

Décentralisé

Pas nécessairement de serveur central.

- Ex. : Git (2005), Mercurial (2005), Arch (2006, disc.), Bazaar (2005)

Avantages de la gestion décentralisée

- Pas dépendant d'une seule machine comme point de défaillance.
- Possibilité de travailler sans être connecté au gestionnaire de version.
- Participation à un projet sans nécessiter de permissions particulières.
- Opérations locales, généralement sont plus rapides.
- Permet le travail privé (brouillons, . . .).
- Permet toutefois de garder un dépôt de référence central.

Désavantages de la gestion décentralisée

- Cloner un dépôt implique de copier tout l'historique
  - plus long
  - espace disque
- Pas de système de verrou
  - peut poser des problèmes pour des données binaires qui ne se fusionnent pas.
- Peut être plus complexe de fusionner différents développements.

Git

Caractéristiques

- Développé à l'origine par Linus Torvalds (avril 2005)
  - remplacement de Bitkeeper pour la gestion du noyau Linux
- Fonctionne sous Linux, Unix, OS X, Windows
- GPL v2
- Performant
- Communications par les protocoles GIT, HTTP, HTTPS, SSH
- Commits identifiés par une somme SHA-1 (*hash*)

Hébergement

- Services : bitbucket.org, framagit.org, github.com, gitlab.com
- À l'IUT : gitlab.iut-bm.univ-fcomte.fr
- Logiciels : gitolite, GitLab

Références

Ressources locales

- sur `https://cours-info.iut-bm.univ-fcomte.fr` → *S2* → *R2.03 - Qualité du développement*

Ressources globales

- Manuel (man), en particulier gittutorial(7), gittutorial-2(7)
- Site de référence : `https://www.git-scm.com/`
- Livre : Pro Git (Scott Chacon et Ben Straub) `https://www.git-scm.com/book`

Organisation

Travaux pratiques

- Création et utilisation d'un dépôt local
- Création et utilisation d'un dépôt sur un serveur central
- Enregistrer des modifications (commits)
- Échanges entre dépôts
- Exploration de l'historique d'un projet
- Gestion des branches
- etc.

Contrôle de connaissances final

- Contrôle sur feuille
- Note comptant pour la SAÉ S2.05

Pour aller plus loin

Fonctionnalités

- Les hooks (cf. githooks(5))
  permettent de déclencher automatiquement des actions
  (vérification des commits, envoi de mail, etc.)
- Interfaçage avec d'autres systèmes :
  git svn, git cvsimport, ...

Installer son propre serveur

Gestion d'utilisateurs, contrôle d'accès.
Exemples :

- gitolite (`https://gitolite.com/gitolite`)
- GitLab (`https://about.gitlab.com/products/`)

Autres outils

Interface web

- Exemple : gitweb

Revue de code

- Exemple : Gerrit (`https://www.gerritcodereview.com/`)

Intégration continue

- Jenkins (`https://jenkins.io/`)
- GitLab CI (`https://about.gitlab.com/gitlab-ci/`)
- Travis CI (`https://travis-ci.org/`)
- ...