

Git et Gitlab au quotidien

Commandes Principales et Workflows

Workflow - Gitlab

Benoît Bayol

June 9, 2015

Outline

Licence

Workflow

Gitlab

Topic

Licence

Workflow

Gitlab

- ▶ Les graphiques présents dans les slides sont extraits du livre "Pro Git 2" qui est aussi présent dans l'archive.
- ▶ La licence de ce document est :
<https://creativecommons.org/licenses/by-nc-sa/3.0/>

Topic

Licence

Workflow

Gitlab

- ▶ Un workflow ou flux de travail est le processus décrivant les interactions entre les différentes entités permettant la mise au point d'un code et son déploiement.
- ▶ Un workflow est un modèle de travail idéal. La plupart des activités de développement tombent dans le cadre d'un workflow.
- ▶ Workflow types
 - ▶ développeur isolé
 - ▶ équipe de développeurs avec un seul dépôt
 - ▶ équipe de développeurs avec plusieurs dépôts
 - ▶ équipe de développeurs avec plusieurs dépôts et un dépôt officiel d'intégration
- ▶ Savoir reconnaître un workflow est très important MAIS savoir reconnaître quand ne pas l'appliquer ou le transformer est encore plus important.
 - ▶ Faire la distinction entre faire la chose correcte et faire correctement la chose.
- ▶ Il y a une notion de masse critique dans l'utilisation d'un workflow.
- ▶ Il faut aussi connaître les gens avec qui vous travaillerez avant de l'appliquer lorsque vous êtes sur des petites tailles (compétences, inertie et résistance au changement, ...)

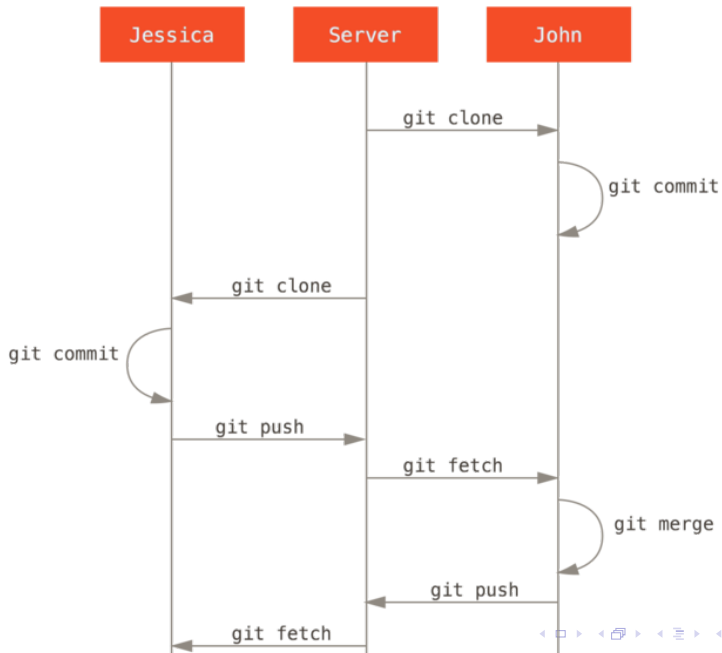
Vision globale

- ▶ Workflow Idéal :
 - ▶ Sélection des nouveaux éléments à développer (Scrum // Kanban)
 - ▶ Écriture des tests d'acceptation en groupe avec le responsable architecture (Boost.Test)
 - ▶ Écriture du code avec en parallèle écriture de tests unitaire et commits (C++ / Boost.Test / Git)
 - ▶ Publication sur un dépôt distant (Git // Gitlab)
 - ▶ Récupération du code par automate (Gitlab CI // Jenkins)
 - ▶ Compilation et passage des tests
 - ▶ Si ok déploiement sinon refus (Ansible // Docker)
- ▶ Dans l'idéal le flux de travail va de l'écriture du code à sa livraison sous forme de programme testé et validé.
- ▶ On distingue différents outils pour arriver à cette fin :
 - ▶ Gestion de version
 - ▶ Git
 - ▶ Intégration continue
 - ▶ Gitlab CI
 - ▶ Jenkins
 - ▶ Déploiement Continu
 - ▶ Script
 - ▶ Docker / Vagrant
 - ▶ Ansible / Chef / Puppet

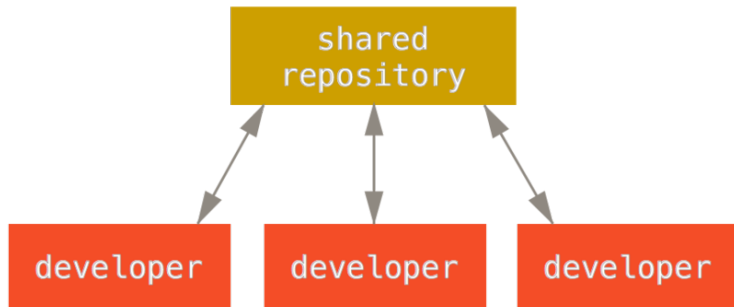
Exemple :

- ▶ <http://decrypt.ysance.com/2014/07/retour-experience-integration-continue-avec-docker-gitlab-jenkins/>

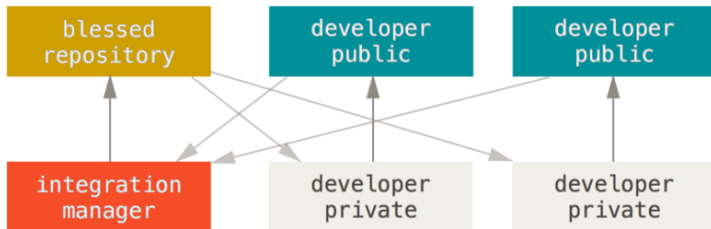
Cycle général

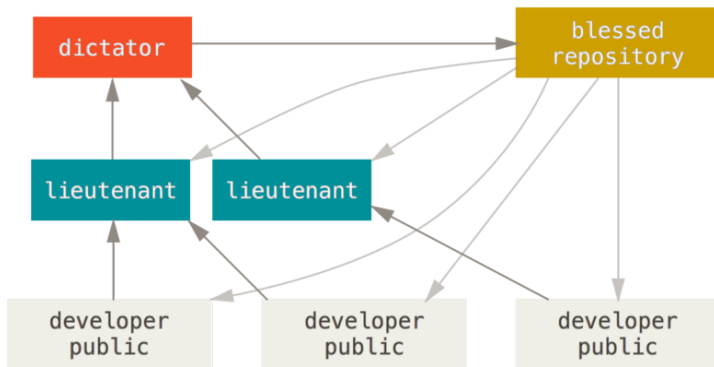


Centralisé



Moyen Projet





Topic

Licence

Workflow

Gitlab

Présentation Générale

- ▶ Gitlab est un logiciel orienté web qui permet de stocker et manipuler des dépôts git sur un serveur
- ▶ Il y a en plus des outils de communication comme un wiki, la possibilité de mettre des snippets, de le connecter à d'autres services.

Utilisation

- ▶ se logger
- ▶ créer un dépôt
- ▶ README
- ▶ cloner le dépôt
- ▶ pusher
- ▶ forker
- ▶ merge request
- ▶ issue

Comparaison à Github

- ▶ Pourquoi choisir Gitlab plutôt que Github, ou l'inverse ?
 - ▶ Github est un logiciel de type SaaS qui est gratuit pour du code publié à tous et payant pour avoir des dépôts privés. L'ensemble des données est hébergé chez Rackspace. On peut avoir un github au sein d'un réseau mais c'est relativement cher.
 - ▶ Gitlab est logiciel libre de type SaaS qui est gratuit du moment que l'on installe sur une instance en privé L'ensemble des données est hébergé sur vos serveurs.
 - ▶ Je pense que pour un projet libre et afin de garantir sa diffusion github est une très bonne plateforme. Néanmoins si l'on travaille sur des codes "internes" ou du moins dont on ne connaît pas encore le type de diffusion ou qui fait intervenir des données privés ou sensibles, Il est important d'héberger l'ensemble sur une plateforme permettant de rester "maître" de ce qui se passe. Bien sur cela est à mettre en regard de la qualité de service du réseau interne en terme d'infrastructure (aide aux utilisateurs, sauvegarde automatique, ...)