Annexe 2

Tutoriel GitLab : installation et premiers pas dans GitLab

GitLab est un **système de contrôle de version populaire** employé principalement en développement logiciel. Programmé avec « Ruby on Rails » par Dmitri Saparoschez, le logiciel basé sur Web sorti en 2011 est aujourd'hui devenu incontournable dans les cercles de développeurs.

L'avantage majeur de GitLab est qu'il facilite grandement le **développement logiciel agile et collaboratif**. Plusieurs développeurs peuvent travailler en même temps sur un projet et par exemple éditer des fonctions en parallèle. La **journalisation continue** de toutes les opérations assure que les modifications apportées au code ne sont jamais perdues ni écrasées par mégarde. De plus, les modifications déjà effectuées peuvent être annulées sans peine.

GitLab est basé sur le **logiciel de gestion de versions** très répandu Git. Librement accessible car open source, Git fait partie des systèmes de gestion de versions les plus utilisés. GitLab représente l'une des alternatives à GitHub les plus connues (suite au rachat de GitHub par Microsoft en 2018, beaucoup d'utilisateurs sont passés à GitLab).

Sommaire

Comment fonctionne GitLab ?

GitLab est une **application basée sur le Web** qui possède une interface utilisateur graphique, mais peut aussi être **installé sur son propre serveur**. Les projets, qui renferment le code à éditer dans des archives numériques appelées **repositories** (référentiels), constituent l'élément central de GitLab. Les dossiers projet contiennent tous les contenus et fichiers du projet logiciel, qui peuvent être des fichiers JavaScript, HTML, CSS, PHP...

GitLab fonctionne dans les grandes lignes de la façon suivante : pour commencer, tous les participants à un projet téléchargent sur leur ordinateur une **copie du référentiel central**. Par la suite, les modifications apportées au code sont toujours d'abord effectuées au moyen de ce qu'on appelle des **commits**. Après l'édition du code, les modifications sont ensuite ajoutées au référentiel central.

Une autre fonction importante est le **branching**. L'utilisateur peut créer une « branche » qui bifurque du tronc, c'est-à-dire de la partie principale du code, et dont le code peut être édité séparément. Cette fonction est intéressante pour pouvoir introduire et tester de nouvelles fonctionnalités sans perturber les travaux de développement au niveau du tronc.

Grâce à des outils intégrés de Continuous Delivery et Continuous Integration, GitLab se prête bien à l'utilisation des branches et offre différentes fonctions utiles comme les « merge requests » et la création de « forks ». C'est pour cela qu'il fait partie des outils d'intégration continue les plus populaires.

Caractéristiques de GitLab

Citons parmi les principales caractéristiques de GitLab :

- Interface utilisateur conviviale
- Branches privées et publiques
- Gestion de plusieurs repositories
- Revue de code
- Suivi intégré des bugs et issues
- Intégration/livraison continue (CI/CD) gratuite incluse
- Wikis de projet
- Création facile de snippets (partage de petits bouts de code)

Licence et modèles d'utilisation de GitLab

GitLab possède un code source ouvert et en libre accès. En 2013, une version « Enterprise Edition » a été introduite pour les entreprises, si bien qu'on trouve aujourd'hui deux modèles d'utilisation :

- GitLab CE : **Community Edition (gratuite)**
- GitLab EE : **Enterprise Edition (payante)**

Les deux versions sont sous **licence open source du MIT**. La version entreprise comporte quelques fonctions supplémentaires que la version gratuite n'a pas. GitLab propose trois modèles d'abonnement différents selon l'étendue des fonctions supplémentaires voulues.

La version entreprise peut aussi être utilisée gratuitement, mais seules les fonctions de base issues de la Community Edition sont accessibles. Ce modèle est intéressant quand on prévoit éventuellement d'**installer ultérieurement la version Enterprise**, sachant que le changement ne demande qu'un clic de souris. A contrario, migrer de la version Community à la version Enterprise prend beaucoup plus de temps.

Marche à suivre pour installer GitLab

Pour utiliser GitLab, il est conseillé d'avoir un **environnement Linux**. À l'instar de Git, GitLab est taillé par nature pour Linux. Son installation et son utilisation sous Windows se heurtent à des limitations. Malgré cela, une solution existe pour remédier au problème : utiliser une machine virtuelle capable de simuler l'environnement Linux sur l'ordinateur Windows. L'installation du

runner GitLab, qui est aussi nécessaire pour utiliser l'intégration continue, ne pose aucun problème.

Héberger GitLab soi-même ou dans le Cloud ?

L'installation de GitLab sur son propre serveur ne pose pas de difficultés majeures à l'utilisateur qui connaît déjà Linux, mais est relativement chronophage. Outre l'installation à proprement parler, il faut prévoir du temps pour la configuration et pour les opérations de maintenance régulières.

Si vous souhaitez vous épargner cela, vous avez aussi la possibilité d'utiliser GitLab comme Software as a Service (SaaS), autrement dit de **l'installer et de l'utiliser sur un serveur Cloud** (différents fournisseurs en proposent).

Dans ce cas, le logiciel est configuré et prêt à être utilisé en très peu de temps, sans installation fastidieuse. Normalement, le runner GitLab est installé en même temps, ce qui permet de commencer immédiatement.

L'avantage d'une installation manuelle sur son **propre environnement serveur** est la **flexibilité supérieure**. L'utilisateur est entièrement libre de choisir les sauvegardes, mises à jour et autres ressources, ainsi que d'installer précisément ce dont il a besoin pour son application concrète. Néanmoins, la solution Cloud est intéressante lorsque l'administrateur a déjà un emploi du temps bien chargé.

Installer GitLab sur un serveur Linux

Pour installer GitLab sur un serveur Linux, il faut au préalable installer le **logiciel Git**. Pour savoir comment procéder, vous pouvez consulter notre tutoriel Git. Une fois fait, vous devez télécharger le **package GitLab Omnibus** sur le site Web officiel de GitLab. Ce package contient tous les fichiers nécessaires et est conseillé pour installer GitLab sur Linux.

Actualiser le repository

Connectez-vous au serveur en tant qu'utilisateur racine (« root ») puis actualisez le repository (ici il s'agit d'Ubuntu) afin de recevoir tous les packages nécessaires pour GitLab. Pour ce faire, utilisez les commandes suivantes :

```
sudo ssh root@GitLabServer
sudo apt-get update
```

Ensuite, installez les packages comme suit :

```
sudo apt install curl openssh-server ca-certificates postfix
```

Pendant l'installation de Postfix, un écran de configuration apparaît. Sélectionnez alors l'option « **Site internet** » puis tapez le **nom du domaine du serveur** que vous utilisez pour envoyer et recevoir des Emails.

Installer GitLab

L'étape suivante consiste à installer le package Omnibus de GitLab. Pour cela, vous devez d'abord ajouter le « GitLab Package Repository » avec cette commande :

```
curl
https://packages.GitLab.com/install/repositories/GitLab/GitLab-ee/script.deb.
sh | sudo bash
```

Ensuite, installez GitLab avec la commande *apt*. Dans ce code d'illustration, GitLab est installé en version Community Edition (CE) :

```
sudo apt install GitLab-ce
```

Après la saisie, le serveur télécharge tout seul le package GitLab et l'installe. Après confirmation de l'installation, vous devez encore **configurer l'URL principale** avec laquelle vous accédez au serveur GitLab.

Remplacez l'URL *GitLab.example.com* par l'URL de votre choix. Pour cela, allez dans le dossier */etc/GitLab*, dans lequel se trouve la configuration, et éditez le fichier de configuration *GitLab.rb* avec l'éditeur de texte standard vim.

Les commandes sont :

```
cd /etc/GitLab
vim GitLab.rb
```

Dans le fichier *GitLab.rb*, recherchez la ligne 9 (« *external_url* ») et saisissez l'URL voulue dedans. GitLab démarrera et configurera l'installation sous cette URL.

Au premier démarrage s'ouvre un écran qui vous invite à **réinitialiser le mot de passe**. Définissez le mot de passe pour l'administrateur. Après cela, vous devriez être redirigé vers l'écran de connexion. Pour vous connecter, vous pouvez d'abord utiliser l'utilisateur standard « root ». Vous pourrez modifier les paramètres plus tard dans les paramètres de profil.

Installer GitLab sous Windows

GitLab lui-même **ne s'installe pas sur un serveur Windows**. Toutefois, à l'aide du **runner GitLab**, il est possible d'accéder depuis Windows à une installation GitLab déjà configurée sur un serveur Linux. Ce logiciel est installé sous Windows et est compatible avec la fonction d'intégration continue (CI/CD) de GitLab. Le runner peut donc envoyer des demandes et des ordres à GitLab.

Télécharger Git pour Windows et les données binaires du runner GitLab

Pour installer GitLab sur un serveur Windows, il faut au préalable **Git pour Windows**. Vous pouvez le télécharger sur le site Web officiel de Git. Par ailleurs, il est recommandé de définir un mot de passe personnel pour le compte utilisateur dès l'instant où le compte système standard n'est pas utilisé.

Il faut aussi en plus ce qu'on appelle un token pour que le runner ait accès à l'instance GitLab. Pour obtenir une clé d'accès de ce genre, rendez-vous dans « **Settings** » -> « **CI/CD** ».

L'étape suivante consiste à **télécharger le fichier binaire** (*x86* ou *amd64*) pour le **runner GitLab pour Windows** et à créer un dossier à l'emplacement de votre choix dans votre système, par exemple *C:\GitLab-runner*.

Collez le fichier dans ce dossier et renommez-le en *GitLab-runner.exe*. Ensuite, ouvrez PowerShell (ou l'invite de commandes) dans Windows en appliquant les droits d'administrateur.

Enregistrer le runner GitLab dans Windows

Pour enregistrer le runner GitLab, tapez maintenant la commande suivante :

```
./GitLab-runner.exe register
```

Ensuite, saisissez l'URL de l'installation GitLab (il s'agit ici d'un exemple) :

```
https://GitLab.com
```

Dans la fenêtre suivante, **saisissez le token** afin d'associer le runner à l'installation GitLab. Vous pouvez ensuite **ajouter unedescription pour le runner**. Cette option peut être modifiée plus tard dans l'interface de GitLab. La fenêtre qui suit donne la possibilité de **définir des tags**. Ces tags servent surtout lorsqu'un runner doit traiter plusieurs projets en même temps, et permettent de savoir exactement quels projets sont attribués.

Pour finir, il vous reste à définir l'*executor*, c'est-à-dire l'environnement dans lequel s'exécute le runner, qui peut être par exemple une instance VirtualBox ou un environnement Shell. Shell est l'exécuteur le plus simple à configurer et l'option par défaut lorsqu'on enregistre un runner GitLab sous Windows pour la première fois.

```
Please enter the executor: ssh, docker+machine, docker-ssh+machine,
kubernetes, docker, parallels, virtualbox, docker-ssh, shell:
shell
```

Installer le runner GitLab sous Windows et le démarrer

Pour installer le runner GitLab, vous pouvez au choix utiliser le compte système intégré ou bien votre propre compte utilisateur. Dans PowerShell ou l'invite de commandes, rendez-vous dans le **dossier que vous avez créé ci-dessus** puis tapez les commandes suivantes successivement :

```
cd C:\GitLab-Runner
.\GitLab-runner.exe install
.\GitLab-runner.exe start
```

Tutoriel GitLab : premiers pas dans GitLab

Une fois GitLab installé, vous pouvez appeler l'**interface utilisateur graphique** au moyen de l'URL précédemment définie. Pour cela, vous pouvez utiliser le navigateur de votre choix et vous connecter ensuite en tant qu'administrateur. Vous pourrez modifier le nom d'utilisateur et le mot de passe associé dans la section Admin par la suite.

Créer un utilisateur dans GitLab

Dans la **section Admin**, vous pouvez créer des utilisateurs en choisissant « **New User** » pour qu'ils travaillent ensemble sur des projets GitLab. À cet effet, vous devez définir une adresse Email ainsi que des données de connexion individuelles pour l'utilisateur afin de lui attribuer le projet voulu.

Au même endroit, vous pouvez aussi modifier les **droits d'utilisateur** avec « Edit » et bloquer voire supprimer des utilisateurs. À noter que le blocage d'un utilisateur empêche celui-ci de se

connecter, mais toutes ses données (commits par exemple) sont conservées. A contrario, la **suppression efface les informations associées à l'utilisateur**, raison pour laquelle cette option doit être utilisée avec prudence.

Créer un nouveau projet

Une action essentielle dans GitLab est la création d'un nouveau projet. Pour ce faire, cliquez sur le bouton « **New Project** ». Vous accédez alors à la fenêtre du projet à créer. Dans le champ « **Project Name** », renseignez le nom du projet (qui ne doit pas contenir de caractères spéciaux ni d'espaces). Dans « **Visibility** », vous réglez la visibilité, c'est-à-dire le niveau d'accès au projet. Les niveaux sont les suivants :

- **Privé** : vous êtes le seul à avoir accès au projet.
- **Interne** : tout utilisateur connecté a accès au projet.
- **Public** : n'importe quel utilisateur a accès au projet sans authentification préalable.

Une fois le paramétrage terminé, cliquez sur « **Create Project** ». Après cela, vous pouvez le lier directement à un référentiel Git local. Pour ce faire, sélectionnez l'option « HTTPS » dans la vue de projet, sous le nom du projet, puis copiez les commandes affichées dans l'interface en ligne de commande.

Si vous n'avez pas encore de **copie locale du repository** sur le serveur, vous pouvez vous rattraper ici en tapant la commande suivante :

```
$ git clone https://server/namespace/project.git
```

Après l'**initialisation du repository par le premier push**, toutes les informations sur le repository sont consultables côté projet. Vous y trouverez également les dernières opérations et, grâce au journal des commits, pourrez savoir qui a modifié le code et quand.

Travailler en équipe avec GitLab

Le moyen le plus simple de collaborer avec d'autres utilisateurs sur un projet GitLab est de leur octroyer un **accès push direct au repository**. Pour cela, ajoutez les utilisateurs au projet comme décrit plus haut, et attribuez-leur les droits d'utilisateur correspondants.

Les utilisateurs disposant de la **permission « Developer »** ou supérieure peuvent placer leurs commits et branches dans le repository sans restrictions. Il est aussi possible d'utiliser ce qu'on appelle des **merge requests**, qui fournissent un contrôle plus strict des accès, puisque la « master branch » ne peut pas être éditée directement. Autrement, les utilisateurs peuvent **créer une branche**, saisir leurs commits, puis effectuer une merge request pour relier la branche à une autre branche ou à la master branch.

Les **utilisateurs sans droits d'accès** peuvent aussi **créer des forks**, autrement dit éditer leur copie du projet avec des push commits. Ensuite, ils peuvent envoyer une merge request afin de réintégrer la fork au projet principal. Cette fonction donne au propriétaire du projet un contrôle total sur ce qui est déposé dans le repository et lui laisse en même temps la possibilité d'accepter les contributions d'utilisateurs inconnus.

SOURCE DU TUTO :
https://www.ionos.fr/digitalguide/sites-internet/developpement-web/tutoriel-gitlab/

FIN DU TUTO DE GITLAB