Manuel du responsable de version

Créer un dépôt GitLab avec intégration continue

Cette documentation décrit la création et la configuration d'un dépôt commun dans le logiciel de forge GitLab qui vise à organiser le développement d'un projet à plusieurs développeurs. Le projet est supposé programmé en Java, géré par le moteur de production Gradle et inclut l'intégration continue GitLab-CI.

Un exemple de projet intégrant tous ces outils est fourni sur la page de la plateforme pédagogique Moodle comme base de travail.

1. Étape 1 : création d'un dépôt commun sur GitLab

1/ Créez un nouveau dépôt avec l'interface du serveur GitLab de l'école, accessible à l'adresse `https://gitlab.ecole.ensicaen.fr` :

- Créez un projet vide initialisé avec README.

2/ Dans le menu *Project Information::Members*[1], ajoutez les membres de votre équipe ainsi que votre encadrant de TP comme développeurs.

3/ Dans le menu *settings::CI/CD*, ajoutez un *runner* pour réaliser l'intégration continue. Pour cela choisissez l'option *Runner* et sélectionnez *Enable Shared Runner* (voir la section 4 pour plus d'information sur l'intégration continue avec GitLab).

4/ Dans le menu *Settings::General::Badges*, ajoutez trois badges d'information qui vont rendre compte de l'état de l'intégration continue : la compilation, les tests et le numéro de version courant.

- Pour le premier badge *pipeline*, mettez littéralement (ie, sans changer un caractère) :

  Link : https://gitlab.ecole.ensicaen.fr/%{project_path}

  Badge image URL : https://gitlab.ecole.ensicaen.fr/%{project_path}/badges/master/pipeline.svg

- Pour le second badge *coverage*, mettez littéralement :

  Link : https://gitlab.ecole.ensicaen.fr/%{project_path}

  Badge image URL : https://gitlab.ecole.ensicaen.fr/%{project_path}/badges/%{default_branch}/coverage.svg?min_medium=30&min_acceptable=45&min_good=50

- Pour le troisième badge *release*, mettez littéralement :

  Link : https://gitlab.ecole.ensicaen.fr/%{project_path}

  Badge image URL : https://gitlab.ecole.ensicaen.fr/%{project_path}/-/badges/release.svg

Si tout est correct, les trois badges (pipeline, coverage et release) doivent apparaître en haut de la page principale du projet.

1 La langue de configuration du dépôt est supposée ici en anglais, configuration que nous vous incitons à utiliser. L'anglais est la langue de travail de l'informaticien.

5/ Dans le menu *General::Naming*, *tags*, *avatar*, vous pouvez aussi ajouter une petite description du projet ainsi qu'une image pour personnaliser votre projet.

6/ Remplacer le contenu du README.md avec une brève description du projet et la liste des étudiants avec leurs rôles.

2. Étape 2 : création de la première version du projet

Avant toute chose, si ce n'est pas déjà fait :

1. Ajoutez une clé SSH à partir du menu *User settings* de la configuration utilisateur de GitLab. Pour cela, suivez les instructions GitLab données dans la page *SSH Keys* correspondante.

La suite de l'installation utilise l'IDE IntelliJ IDEA sur votre compte personnel :

2. Ouvrez le projet dans IntelliJ avec l'option *Get from VCS*.
   - URL : l'adresse est à récupérer sur GitLab. Elle est du genre : `git@gitlab.ecole.ensicaen.fr:clouard/essai.git`
3. Télécharger l'archive contenant l'exemple de projet `project_template.zip` sur la plateforme pédagogique.
4. Décompressez le contenu de l'archive dans le dossier qui contient le projet importé de GitLab. Vérifiez que le projet est organisé comme suit :
   - src/
   - build.gradle
   - README.md
5. Lancez une exécution en utilisant la tâche *run* du menu *Gradle* à droite de l'interface. S'il le faut, afficher la fenêtre Gradle par *View::ToolWindow*.
6. Lancez les tests à partir de l'ongle *Tasks::verification::check* de la fenêtre Gradle.
7. Modifiez le projet pour vous l'approprier. Pour cela, créez une nouvelle branche puis renommez le projet (cf. le fichier *settings.gradle*) et les paquets pour coller à votre projet.
8. Poussez la branche sur GitLab en utilisant le menu *Git::commit* puis *Git::push* (ou les deux raccourcis CTRL+K puis CTRL+SHIFT+K).
9. Retournez sur GitLab pour créer une « merge-request » dans la branche master puis acceptez-la.
10. Vérifiez que l'intégration continue s'exécute bien, ie, le badge du projet passe au vert et la valeur de la couverture est autour de 20 %.

3. Étape 3 : installation du dépôt local pour les développeurs

Le dépôt commun est maintenant en place. La procédure suivante consiste à créer le dépôt local sur chaque compte de développeur. Elle utilise l'IDE IntelliJ IDEA :

1. Fermez tous les projets dans IntelliJ et revenez à la fenêtre d'accueil.
2. Cliquez sur *Checkout from Version Control*.
3. Mettez l'URL du projet, quelque chose du genre : `git@gitlab.ecole.ensicaen.fr:clouard/essai.git`
4. Installez la clé SSH dans les *User settings* de la configuration utilisateur.
5. Dans la fenêtre *Event log*, choisissez *Import Gradle Project*.
6. Faire un essai de compilation et d'exécution des tests.
7. Chaque développeur peut maintenant contribuer au projet en créant une branche par fonctionnalité puis en la poussant sur GitLab.

4. Intégration continue avec GitLab

Sur GitLab, l'intégration continue se fait avec un fichier de commandes nommé `.gitlab-ci.yml`, dont un exemple est donné ici :

```
image: gradle:jdk16

variables:
  GRADLE_OPTS: "-Dorg.gradle.daemon=false"
  http_proxy: http://193.49.200.22:3128
  https_proxy: http://193.49.200.22:3128
  HTTP_PROXY: http://193.49.200.22:3128
  HTTPS_PROXY: http://193.49.200.22:3128

before_script:
  - chmod +x gradlew
  - mv gradle.properties.ci gradle.properties
  - apt-get install findutils

stages:
  - build
  - test

build:
  tags :
    - gitlab-runner-ensicaen
  stage: build
  script:
    - ./gradlew -Pci --console=plain --build-cache assemble
  only:
    - master

unit_test:
  tags :
    - gitlab-runner-ensicaen
  stage: test
  coverage: '/Coverage Total: ([0-9]{1,3})%/'
```

```
33    only:
34      - master
35    script:
36      - ./gradlew -Pci --console=plain test jacocoTestReport
37      - cat build/jacocoHtml/index.html | grep -o 'Total[^%]*%' | sed
    's/<.*>/ /; s/Total/Coverage Total:/'
```

Ce fichier décrit deux tâches d'intégration. La première « build » permet de vérifier que la compilation fonctionne bien et que l'on obtient un exécutable en sortie. La seconde « test » permet de vérifier que tous les tests passent bien sur cette version.

Ce fichier est exécuté par un « runner » à chaque fois que la branche master est modifiée. Un runner utilise la technologie Docker dont le fonctionnement est proche de celui d'une machine virtuelle Linux en beaucoup plus léger. Il est possible d'installer un « runner » sur sa machine (voir la documentation GitLab).

Si l'intégration est satisfaite, le badge « build » passe au vert et le badge de couverture contient le pourcentage de couverture des tests. Si l'intégration n'est pas satisfaite, alors la badge passe au rouge. Dans ce cas, l'équipe se mobilise toute affaire cessante pour résoudre le problème et refaire passer la badge au vert.

5. Création de la première « release »

Afin de renseigner le badge *release*, créez la première *release*.

1/ Dans GitLab, menu *Deployments::Release*, créez la *release* 0.0.1 à partir du contenu de la branche `master`.

6. La revue de code avec GitLab

Le responsable de version est aussi responsable de la qualité du code du logiciel. Pour cela, avant chaque acceptation d'une « merge request », il doit vérifier la qualité du code par une revue de code.

6.1 La revue de code sur IntelliJ

La revue de code peut se faire en rapatriant la branche dans IntelliJ puis en y ajoutant des remarques, précédées du tag FIXME. Il suffit de pousser cette version corrigée sur le GitLab pour que le développeur voit les remarques que vous avez apportées. Il peut alors rapatrier la version sur sa machine et appliquer les corrections avant de repousser cette version sur le GitLab pour une nouvelle revue de code.

6.2 La revue de code directement sur GitLab

Mais, la revue de code peut aussi se faire directement sur le serveur GitLab. Pour cela, sur la page de la « merge request », il suffit d'aller dans l'onglet « changes » ou « commits » et de relire les changements pour y ajouter des commentaires ou des propositions de changements (toujours précédé du tag FIXME). Le développeur fait

ses corrections sur sa machine, mais il peut aussi les faire directement sur le GitLab. GitLab propose un IDE, sommaire, mais qui peut être suffisant dans bien des cas.

Si tout est correct, vous validez la « merge request » et vous obtenez une nouvelle version du logiciel sur la branche master que tous les développeurs sont invités à fusionner (ou rebaser) à leur version courante.