Centre de Calcul de l'Institut National de Physique Nucléaire et de Physique des Particules

GitLab CI

11 juin 2018

Jean-René ROUET

La fonctionnalité GitLab-CI

- Tuyauterie d'intégration et livraison continue pour construire, tester, déployer et monitorer votre code
  - Intégrée à GitLab
  - Apprentissage facile
  - Scalable et Rapide

Fonctionnalités (quelques unes)

- Fonctionnalité intégrée à GitLab
- Déclenchement d’actions et création de build sur des événements du dépôt
- Multi plateforme: Unix, Windows, OSX, a juste besoin de Go.
- Multi language
- Stable, les runners sont différents de la plate-forme GitLab
- Parallélisme
- Compte rendu de build
- Gitlab Runner supporte docker
- Variables d’environnement
- Processus DevOps
- Production d’artefacts

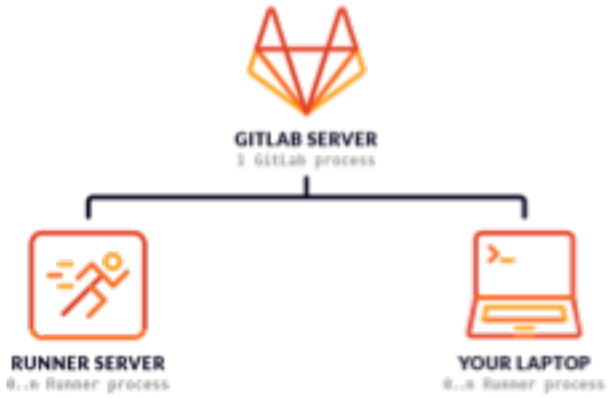
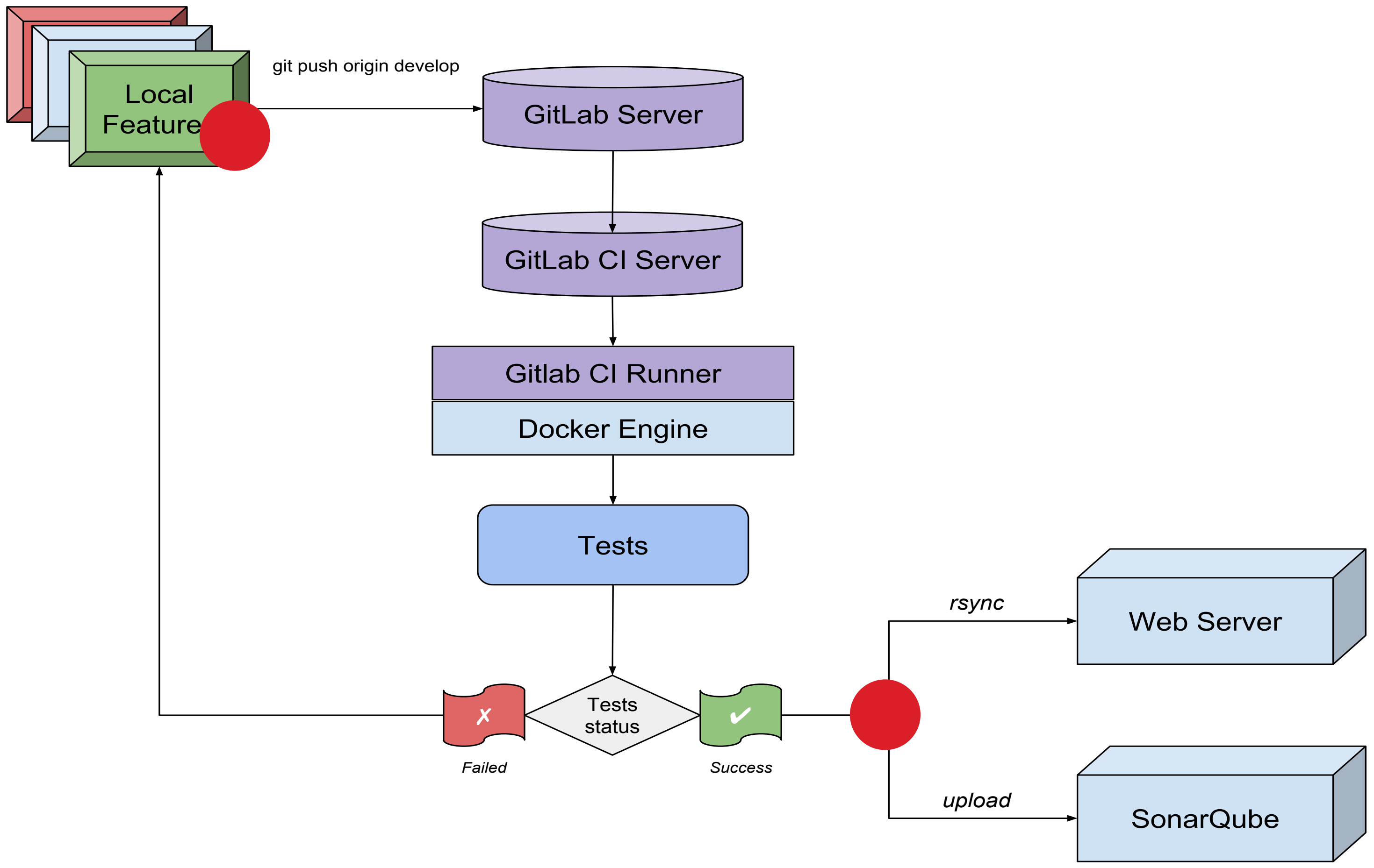


Image copyright GitLab.com

Un exemple de processus de travail au jour le jour



.gitlab-ci.yml

```yaml
image: ccin2p3/php-xdebug-sonar:latest

cache:
  paths:
    - vendor/

before_script:

  # run ssh-agent
  - eval $(ssh-agent -s)

  # add ssh key stored in SSH_PRIVATE_KEY variable to the agent store
  - ssh-add <(echo "$SSH_PRIVATE_KEY")

  […]

  - php composer.phar install

stages:
  - test
  - deploy
```

[illegible]

Mise en cache de fichiers

Script shell exécuté par le runner

Ordonnancement de l'exécution et regroupement des jobs

.gitlab-ci.yml

```
job_phpunit_sonar:
  stage: test
  only:
    - develop
  tags:
    - docker
  script:
      […]
    - php -dzend_extension=xdebug.so phpunit.phar --confi
    - /sonar-scanner-2.5/bin/sonar-runner -Dsonar.host.url=${                                $
{SONAR_JDBC_URL} -Dsonar.jdbc.username=${SONAR_JDBC_USERNAME}
{SONAR_JDBC_PASSWORD} -Dsonar.projectVersion=${CI_BUILD_REF}

job_phpunit:
  stage: test
  only:
    - tags
  tags:
    - docker
  script:
    - php app/console doctrine:database:drop --env=test --forc
    […]
    - php phpunit.phar --configuration app/phpunit.xml.dist

job_deploy_prod:
  stage: deploy
  only:
    - tags
  script:
    - echo "${PARAMETERS}" > app/config/parameters.yml
    - rsync -az --delete --exclude=web/media --exclude=/web/cache -e "ssh" .
webcast2@webcast2.in2p3.fr:/www/htdocs
```

Nom des jobs

Script shell exécuté par le runner

Étape dans laquelle va être exécuté le job

```
stages:
  - test
  - deploy
```

.gitlab-ci.yml

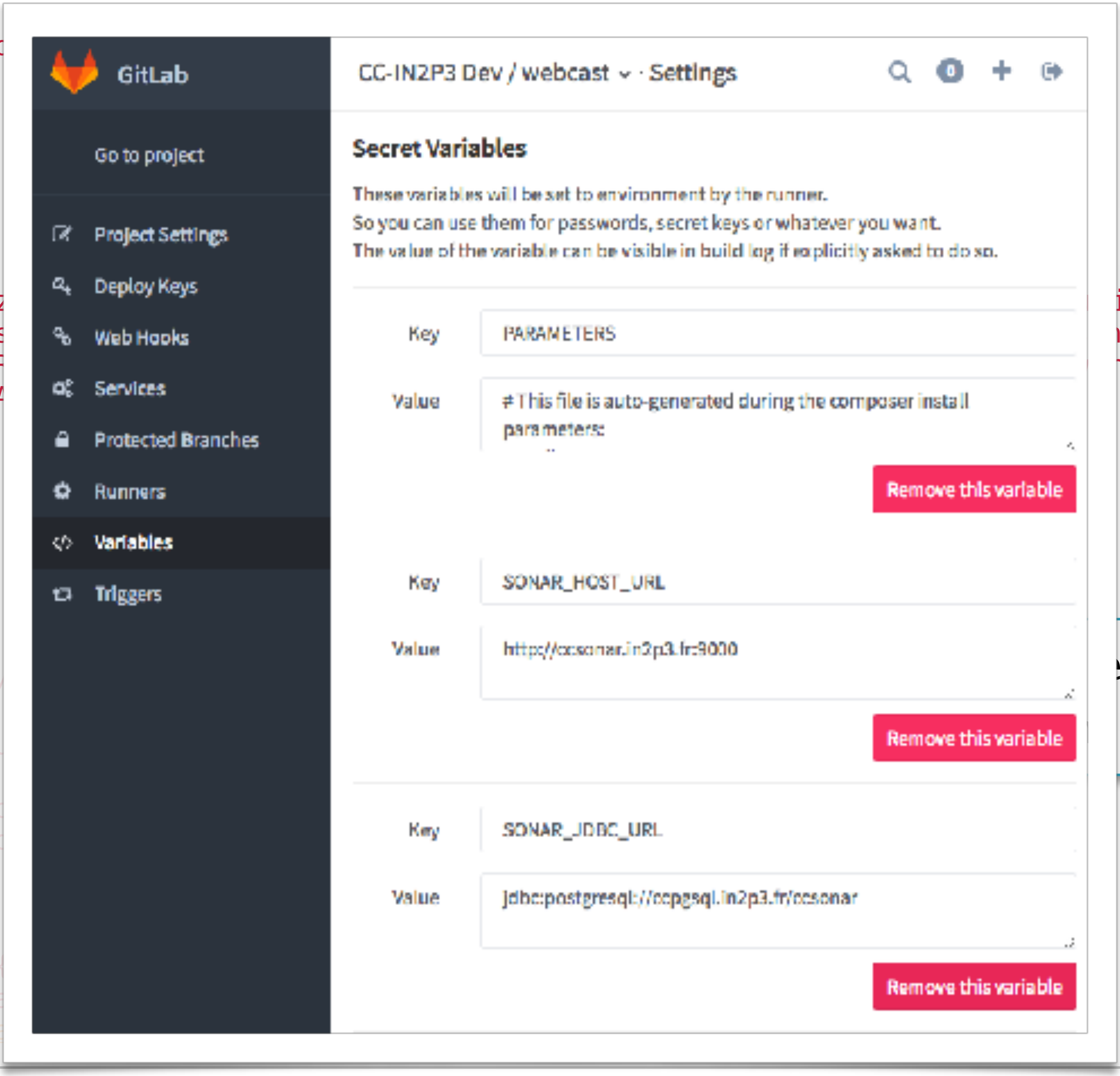
```
job_phpunit_sc
  stage: test
  only:
    - develop
  tags:
    - docker
  script:
      [...]
     - php -dz                                                it.xml.dist
    - /sonar-s                                                ar.jdbc.url=$
{SONAR_JDBC_UF                                                d=$
{SONAR_JDBC_PA

job_phpunit:
  stage: test
  only:
    - tags
  tags:
    - docker
  script:
    - php app/
    [...]
    - php phpu

job_deploy_pro
  stage: deplo
  only:
    - tags
  script:
    - echo "$
    - rsync -a
webcast2@webca
```

s

ment



Secret Variables

These variables will be set to environment by the runner.
So you can use them for passwords, secret keys or whatever you want.
The value of the variable can be visible in build log if explicitly asked to do so.

Key	PARAMETERS
Value	# This file is auto-generated during the composer install parameters:
Key	SONAR_HOST_URL
Value	http://ccsonar.in2p3.fr:9000
Key	SONAR_JDBC_URL
Value	jdbc:postgresql://ccpgsql.in2p3.fr/ccsonar

Démarrage

- Ajouter un fichier .gitlab-ci.yml à la racine de son projet

```yaml
# This file is a template, and might need editing before it works on your project.
# use the official gcc image, based on debian
# can use verions as well, like gcc:5.2
# see https://hub.docker.com/_/gcc/
image: gcc

stages:
  - build
  - test

build:
  stage: build
  # instead of calling g++ directly you can also use some build toolkit like make
  # install the necessary build tools when needed
  # before_script:
  #   - apt update && apt -y install make autoconf
  script:
    - g++ helloworld.cpp -o mybinary
  artifacts:
    paths:
      - mybinary
  # depending on your build setup it's most likely a good idea to cache outputs to reduce the build time
  # cache:
  #   paths:
  #     - "*.o"

# run tests using the binary built before
test:
  stage: test
  script:
    - ./runmytests.sh
```

Exemple plus complexe

```
# This file is a template, and might need editing before it works on your project.
# Read more about this script on this blog post https://about.gitlab.com/2016/11/30/setting-up-gitlab-ci-for-android-projects/, by Greyson Parrelli
image: openjdk:8-jdk

variables:
  ANDROID_COMPILE_SDK: "25"
  ANDROID_BUILD_TOOLS: "24.0.0"
  ANDROID_SDK_TOOLS: "24.4.1"

before_script:
  - apt-get --quiet update --yes
  - apt-get --quiet install --yes wget tar unzip lib32stdc++6 lib32z1
  - wget --quiet --output-document=android-sdk.tgz https://dl.google.com/android/android-sdk_r${ANDROID_SDK_TOOLS}-linux.tgz
  - tar --extract --gzip --file=android-sdk.tgz
  - echo y | android-sdk-linux/tools/android --silent update sdk --no-ui --all --filter android-${ANDROID_COMPILE_SDK}
  - echo y | android-sdk-linux/tools/android --silent update sdk --no-ui --all --filter platform-tools
  - echo y | android-sdk-linux/tools/android --silent update sdk --no-ui --all --filter build-tools-${ANDROID_BUILD_TOOLS}
  - echo y | android-sdk-linux/tools/android --silent update sdk --no-ui --all --filter extra-android-m2repository
  - echo y | android-sdk-linux/tools/android --silent update sdk --no-ui --all --filter extra-google-google_play_services
  - echo y | android-sdk-linux/tools/android --silent update sdk --no-ui --all --filter extra-google-m2repository
  - export ANDROID_HOME=$PWD/android-sdk-linux
  - export PATH=$PATH:$PWD/android-sdk-linux/platform-tools/
  - chmod +x ./gradlew

stages:
  - build
  - test
```

Exemple plus complexe

```
build:
  stage: build
  script:
    - ./gradlew assembleDebug
  artifacts:
    paths:
    - app/build/outputs/


unitTests:
  stage: test
  script:
    - ./gradlew test


functionalTests:
  stage: test
  script:
    - wget --quiet --output-document=android-wait-for-emulator https://raw.githubusercontent.com/travis-ci/travis-
cookbooks/0f497eb71291b52a703143c5cd63a217c8766dc9/community-cookbooks/android-sdk/files/default/android-wait-for-
emulator
    - chmod +x android-wait-for-emulator
    - echo y | android-sdk-linux/tools/android --silent update sdk --no-ui --all --filter sys-img-x86-google_apis-
${ANDROID_COMPILE_SDK}
    - echo no | android-sdk-linux/tools/android create avd -n test -t android-${ANDROID_COMPILE_SDK} --abi
google_apis/x86
    - android-sdk-linux/tools/emulator64-x86 -avd test -no-window -no-audio &
    - ./android-wait-for-emulator
    - adb shell input keyevent 82
    - ./gradlew cAT
```

Troisième exemple

```
# This file is a template, and might need editing before it
works on your project.
# Full project: https://gitlab.com/pages/plain-html
pages:
  stage: deploy
  script:
  - mkdir .public
  - cp -r * .public
  - mv .public public
  artifacts:
    paths:
    - public
  only:
  - master
```

GitLab-CI toutes les options (enfin presque)

- Jobs
- image; services
- before_script; after_script
- stages
- script
- only; except
- tags
- when
- environment
- cache
- artifacts
- dependencies
- job templates
- Le reste sur : https://docs.gitlab.com/ce/ci/yaml/README.html

GitLab-CI exemples de la vie réelle

Construction d'image

```
# This file is a template, and might need editing before it works on your project.
# Official docker image.
image: docker:latest

services:
  - docker:dind

before_script:
  - docker login -u "$CI_REGISTRY_USER" -p "$CI_REGISTRY_PASSWORD" $CI_REGISTRY

build-master:
  stage: build
  script:
    - docker build --pull -t "$CI_REGISTRY_IMAGE" .
    - docker push "$CI_REGISTRY_IMAGE"
  only:
    - master

build:
  stage: build
  script:
    - docker build --pull -t "$CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG" .
    - docker push "$CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG"
  except:
    - master
```