Gitlab Workflow : (From Idea to Production) or (Code, Integration, Deployement, Delivery)

Benoît Bayol, BIOMATHEMATICS, MICS Laboratory, CentraleSupélec

30-31 May 2017

Contents

1 What is this document ? 1

2 Which data are we going to use during this session ? 2
2.1 creating data with django (conda or pip, ...) . . . 2
2.2 creating data with pseudocode . . . 2
2.3 creating data with previous projects of the practical session . . . 2
2.4 creating data with your own projects . . . 2

3 What are the basic tasks with gitlab ? 2

4 Where can I find information about continuous integration with Gitlab ? 3
4.1 Reference documentation . . . 3
4.2 Quick start (.gitlab-ci.yml and runners) . . . 3
4.2.1 Installation of runner (M, VM, Cloud, HPC) . . . 3
4.2.2 Code . . . 3
4.3 YAML Reference . . . 3
4.3.1 global keywords . . . 3
4.3.2 jobs keywords . . . 4
4.4 Creating a pipeline (a collection of jobs with different stages) . . . 4
4.5 Knowing about variables . . . 4
4.6 Using docker images . . . 4
4.7 Building docker images and pushing to registry . . . 4

5 Examples of .gitlab-ci.yml scripts 4
5.1 Using a "shell" runner . . . 4
5.1.1 hello world . . . 4
5.1.2 run on shell . . . 4
5.1.3 reformat for adding test stage on shell . . . 5
5.1.4 test and deploy on shell . . . 5
5.1.5 build, test and deploy a docker image on shell . . . 5
5.1.6 build gitlab pages website . . . 6
5.1.7 build, test, deploy, release and publish pages . . . 6
5.2 Using a "docker" runner . . . 7
5.2.1 test on docker . . . 7
5.2.2 build, test, deploy with docker in docker . . . 8

1 What is this document ?

This document is a notes version of a talk given during a workshop about continous integration for scientific software at Institut Henri Poincaré in Paris, France. This workshop has been organized by the french group : calcul@listes.math.cnrs.fr and was about git, docker, jenkins, gitlab and travis.

We have seen major features of gitlab with an emphasis on continuous integration.

2 Which data are we going to use during this session ?

For the session, we worked on a template code based on django but you could used any other code or pseudocode you wanted.

2.1 creating data with django (conda or pip, . . .)

```
conda create -y -n django python
source activate django
conda install -y django
django-admin startproject myproject
cd myproject
./manage.py startapp website
echo "class MyTests(TestCase):
    def test_1(self):
        self.assertEqual(1,1)" > website/tests.py
./manage.py test
```

2.2 creating data with pseudocode

```
function add(argv):
  return argv[0] + argv[1]
```

2.3 creating data with previous projects of the practical session

2.4 creating data with your own projects

3 What are the basic tasks with gitlab ?

We have seen the tasks below :

- create a project
- information
- add code
- add branch
- add members
- specific case of master
- report issue ; board ; milestone
- merge request
- fork
- manage large files

```
git lfs install
git lfs track "*.hdf5"
git add .gitattributes
git add data.hdf5
git commit -m "Add data file"
git push origin master
#...
git pull origin master
git lfs checkout
```

- create group
- administration
- `https://mattermost.math.unistra.fr/`
- `https://bayol.pages.math.unistra.fr/tp-gitlab/`
- `https://registry.math.unistra.fr/`

4 Where can I find information about continuous integration with Gitlab ?

4.1 Reference documentation

`https://docs.gitlab.com/ce/ci/`

4.2 Quick start (.gitlab-ci.yml and runners)

`https://docs.gitlab.com/ce/ci/quick_start/`

4.2.1 Installation of runner (M, VM, Cloud, HPC)

`https://docs.gitlab.com/runner/install/`

4.2.2 Code

Here is the output of the installation and registration of a runner. You need to find the coordinator URL and token registration on your gitlab instances in your project/settings or in the global administration panel.

```
sudo apt-get install gitlab-ci-multi-runner
sudo gitlab-ci-multi-runner register
  Please enter the gitlab-ci coordinator URL (e.g. https://gitlab.com )
  https://gitlab.com
  Please enter the gitlab-ci token for this runner
  xxx
  Please enter the gitlab-ci description for this runner
  my-runner
  INFO[0034] fcf5c619 Registering runner... succeeded
  Please enter the executor: shell, docker, docker-ssh, ssh?
  docker
  Please enter the Docker image (eg. ruby:2.1):
  ruby:2.1
  INFO[0037] Runner registered successfully. Feel free to start it, but if it's
  running already the config should be automatically reloaded!
```

4.3 YAML Reference

`https://docs.gitlab.com/ce/ci/yaml/`

4.3.1 global keywords

image	no	Use docker image, covered in Use Docker
services	no	Use docker services, covered in Use Docker
stages	no	Define build stages
types	no	Alias for stages (deprecated)
before_script	no	Define commands that run before each job's script
after_script	no	Define commands that run after each job's script
variables	no	Define build variables
cache	no	Define list of files that should be cached between subsequent runs

4.3.2 jobs keywords

script	yes	Defines a shell script which is executed by Runner
image	no	Use docker image, covered in Using Docker Images
services	no	Use docker services, covered in Using Docker Images
stage	no	Defines a job stage (default: test)
type	no	Alias for stage
variables	no	Define job variables on a job level
only	no	Defines a list of git refs for which job is created
except	no	Defines a list of git refs for which job is not created
tags	no	Defines a list of tags which are used to select Runner
allow_failure	no	Allow job to fail. Failed job doesn't contribute to commit status
when	no	Define when to run job. Can be on_success, on_failure, always or manual
dependencies	no	Define other jobs that a job depends on so that you can pass artifacts between them
artifacts	no	Define list of job artifacts
cache	no	Define list of files that should be cached between subsequent runs
before_script	no	Override a set of commands that are executed before job
after_script	no	Override a set of commands that are executed after job
environment	no	Defines a name of environment to which deployment is done by this job
coverage	no	Define code coverage settings for a given job

4.4 Creating a pipeline (a collection of jobs with different stages)

`https://docs.gitlab.com/ce/ci/pipelines.html`

4.5 Knowing about variables

`https://docs.gitlab.com/ce/ci/variables`

4.6 Using docker images

`https://docs.gitlab.com/ce/ci/docker/using_docker_images.html`

4.7 Building docker images and pushing to registry

`https://docs.gitlab.com/ce/ci/docker/using_docker_build.html`

5 Examples of .gitlab-ci.yml scripts

5.1 Using a "shell" runner

5.1.1 hello world

```
hello_world:
  #script is the only mandatory keyword
  script:
    - echo "Hello World"
  #tags will help to assign a job to a particular runner or a set of runners
  tags:
    - shell
```

5.1.2 run on shell

In this script I install dependencies and execute tests.

```
run:
  script:
    - wget https://bootstrap.pypa.io/get-pip.py
    - python get-pip.py --user
    - /home/gitlab-runner/.local/bin/pip install --user django
    - ./manage.py test
  tags:
    - shell
```

5.1.3 reformat for adding test stage on shell

In this script I add stages for creating a pipeline of jobs with only test being used.

```
stages:
  - test
  - deploy

run_test:
  stage: test
  script:
    - wget https://bootstrap.pypa.io/get-pip.py
    - python get-pip.py --user
    - /home/gitlab-runner/.local/bin/pip install --user django
    - ./manage.py test
  tags:
    - shell
```

5.1.4 test and deploy on shell

In this script I do use test and deploy stages. deploy stage is using the registry for pushing a new docker image. You can find information on the registry in the registry section of your project.

```
stages:
  - test
  - deploy

run_test:
  stage: test
  script:
    - wget https://bootstrap.pypa.io/get-pip.py
    - python get-pip.py --user
    - /home/gitlab-runner/.local/bin/pip install --user django
    - ./manage.py test
  tags:
    - shell

run_deploy:
  stage: deploy
  script:
  #Here I use a login mechanism given by gitlab. The user is gitlab-ci-token and the password is given
    - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN registry.math.unistra.fr
    - docker build -t registry.math.unistra.fr/bayol/tp-gitlab/image:latest .
    - docker push registry.math.unistra.fr/bayol/tp-gitlab/image:latest
  tags:
    - shell
```

5.1.5 build, test and deploy a docker image on shell

Here we add some variables by using some global variables that are set by the gitlab platform like $CI_COMMIT_REF_NAM

```
stages:
- build
- test
- release

variables:
  CONTAINER_TEST_IMAGE: registry.math.unistra.fr/bayol/tp-gitlab/image:$CI_COMMIT_REF_NAME
  CONTAINER_RELEASE_IMAGE: registry.math.unistra.fr/bayol/tp-gitlab/image:latest

before_script:
  - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN registry.math.unistra.fr
```

```
build:
  stage: build
  script:
    - docker build --pull -t $CONTAINER_TEST_IMAGE .
    - docker push $CONTAINER_TEST_IMAGE
  tags:
    - shell

test1:
  stage: test
  script:
    - docker pull $CONTAINER_TEST_IMAGE
    - docker run $CONTAINER_TEST_IMAGE /manage.py test
  tags:
    - shell

release-image:
  stage: release
  script:
    - docker pull $CONTAINER_TEST_IMAGE
    - docker tag $CONTAINER_TEST_IMAGE $CONTAINER_RELEASE_IMAGE
    - docker push $CONTAINER_RELEASE_IMAGE
  only:
    - master
  tags:
    - shell
```

5.1.6 build gitlab pages website

Here we use the pages mechanism for publishing the index.html page that is available in the repository.

```
pages:
  stage: deploy
  script:
  - mkdir .public
  - cp index.html .public
  - mv .public public
  artifacts:
    paths:
    - public
  only:
  - master
  tags:
    - shell
```

5.1.7 build, test, deploy, release and publish pages

```
stages:
- build
- test
- release
- deploy

variables:
  CONTAINER_TEST_IMAGE: registry.math.unistra.fr/bayol/tp-gitlab/image:$CI_COMMIT_REF_NAME
  CONTAINER_RELEASE_IMAGE: registry.math.unistra.fr/bayol/tp-gitlab/image:latest

before_script:
  - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN registry.math.unistra.fr

build:
  stage: build
  script:
    - docker build --pull -t $CONTAINER_TEST_IMAGE .
    - docker push $CONTAINER_TEST_IMAGE
  tags:
    - shell

test1:
  stage: test
  script:
    - docker pull $CONTAINER_TEST_IMAGE
    - docker run $CONTAINER_TEST_IMAGE /manage.py test
  tags:
    - shell

release-image:
  stage: release
  script:
    - docker pull $CONTAINER_TEST_IMAGE
    - docker tag $CONTAINER_TEST_IMAGE $CONTAINER_RELEASE_IMAGE
    - docker push $CONTAINER_RELEASE_IMAGE
  only:
    - master
  tags:
    - shell

# with also another deploy script for giving download to users
# deploy:
#   stage: deploy
#   script:
#     - ./deploy.sh
#   only:
#     - master

pages:
  stage: deploy
  script:
  - mkdir .public
  - cp index.html .public
  - mv .public public
  artifacts:
    paths:
    - public
  only:
  - master
  tags:
    - shell
```

5.2 Using a "docker" runner

5.2.1 test on docker

Here we use a "docker" runner for using an image of ananconda directly.

```
image: continuumio/anaconda:4.3.1

stages:
  - test

run_test:
  stage: test
  script:
    - conda create -y -n django
    - source activate django
    - conda install -y django
    - ./manage.py test
  tags:
    - docker
```

5.2.2 build, test, deploy with docker in docker

Warning : docker in docker is "hype" but might not be suitable for you. Simpler is better. See https://jpetazzo.github.io/2015/09/03/do-not-use-docker-in-docker-for-ci/

```
image: docker:latest
services:
- docker:dind

stages:
- build
- test
- release


variables:
  CONTAINER_TEST_IMAGE: registry.math.unistra.fr/bayol/tp-gitlab/image:$CI_COMMIT_REF_NAME
  CONTAINER_RELEASE_IMAGE: registry.math.unistra.fr/bayol/tp-gitlab/image:latest

before_script:
  - docker login -u gitlab-ci-token -p $CI_JOB_TOKEN registry.math.unistra.fr

build:
  stage: build
  script:
    - docker build --pull -t $CONTAINER_TEST_IMAGE .
    - docker push $CONTAINER_TEST_IMAGE

test1:
  stage: test
  script:
    - docker pull $CONTAINER_TEST_IMAGE
    - docker run $CONTAINER_TEST_IMAGE /manage.py test

release-image:
  stage: release
  script:
    - docker pull $CONTAINER_TEST_IMAGE
    - docker tag $CONTAINER_TEST_IMAGE $CONTAINER_RELEASE_IMAGE
    - docker push $CONTAINER_RELEASE_IMAGE
  only:
    - master
```