

#### R5.A.09 Virtualisation avancée



# Part 2 GitLab CI

Matthieu Le Lain, Xavier Roirand

BUT3 INFO, 2023-2024

IUT de Vannes, Université Bretagne Sud

matthieu.le-lain@univ-ubs.fr

# **Planning**

> Runners

➤ Shared Runners

Création de conteneurs et push DockerHub

# **Gitlab Runners**

### **Gitlab Runners**

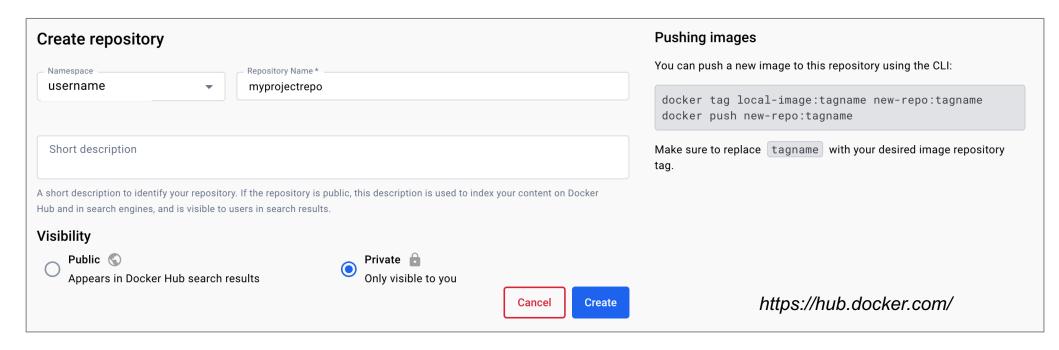
#### Tour d'horizon

Automatiser ses déploiements | GitLab CI CD | Pipelines

https://www.youtube.com/watch?v=8aYNrBMekA4

## **Shared Runners**

# **Repository Docker Hub**



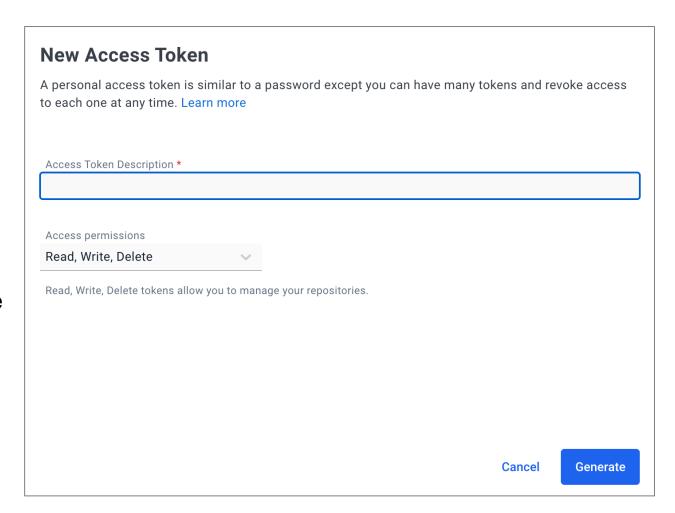
- Création d'un nouveau repository sur Docker
- Le nom du dépôt (path) sera utilisé en tant que variable dans gitlab
- Le Docker Hub centralise toutes nos images Docker, et leurs versions

#### **Token Docker Hub**

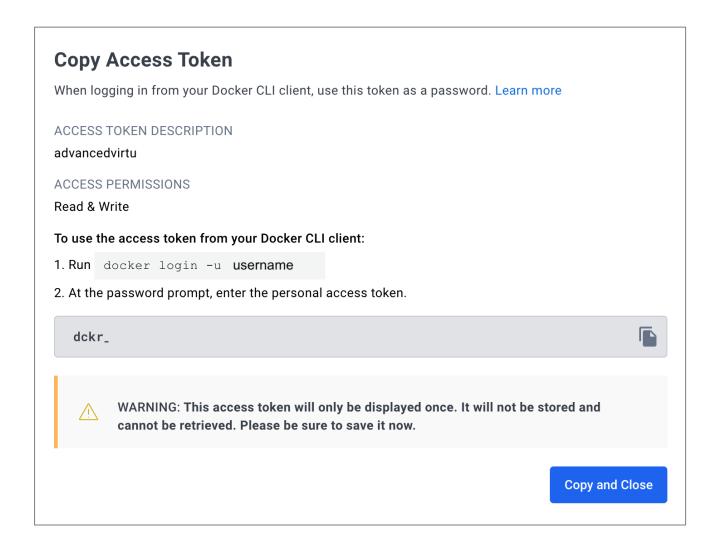
**Token d'accès:** Afin de pouvoir pousser notre Image directement sur le Docker Hub à partir de Gitlab, nous devons créer un Token (Read and Write).

Compte DockerSecurityNew Token

ATTENTION: Le token ne sera visible qu'à la création. Ensuite il ne sera plus consultable, il faudra en refaire un autre si vous ne le notez pas!



#### **Token Docker Hub**



#### **GITLAB CI**

#### **Gitlab Projet**

Paramètres=> CI/CD=> Variables

#### **Variables**

Les variables stockent des informations, telles que des mots de passe et des clés secrètes, que vous pouvez utiliser dans les scripts de jobs. Chaque project peut définir un maximum de 8000 variables. En savoir plus.

Les variables peuvent avoir plusieurs attributs. En savoir plus.

- Protégées : Exposées uniquement aux branches ou étiquettes protégées.
- Masqué : Masqué dans les journaux des logs. Doit correspondre aux exigences de masquage.
- Affichée : les variables contenant \$ seront traitées comme le début d'une référence vers une autre variable.

Variables CI/CD  4		Révéler les valeurs Ajouter	une variable
↑ Clé	Valeur	Environnements	Actions
HUB 🛱 Protégée Développée	***** [2]	Tous (par défaut) ਹਿ	
HUB_IMAGE [c] Protégée Développée	***** [0]	Tous (par défaut) ਹਿ	
TOKEN C	***** [ ]	Tous (par défaut) ្រឹ	
USERNAME CONTROL Protégée Développée	***** [1]	Tous (par défaut) ្រឹ	

# **Exemple de Variables**

**HUB**: docker.io

**HUB\_IMAGE**: index.docker.io/username/dockerproject

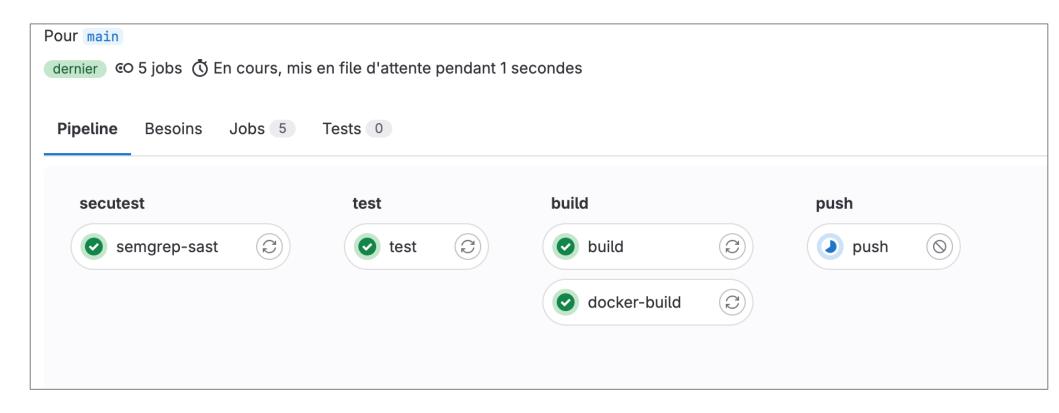
Ex: index.docker.io/username/ myprojectrepo

**HUB\_IMAGE**: index.docker.io/username/dockerproject

**TOKEN**: le\_token\_docker

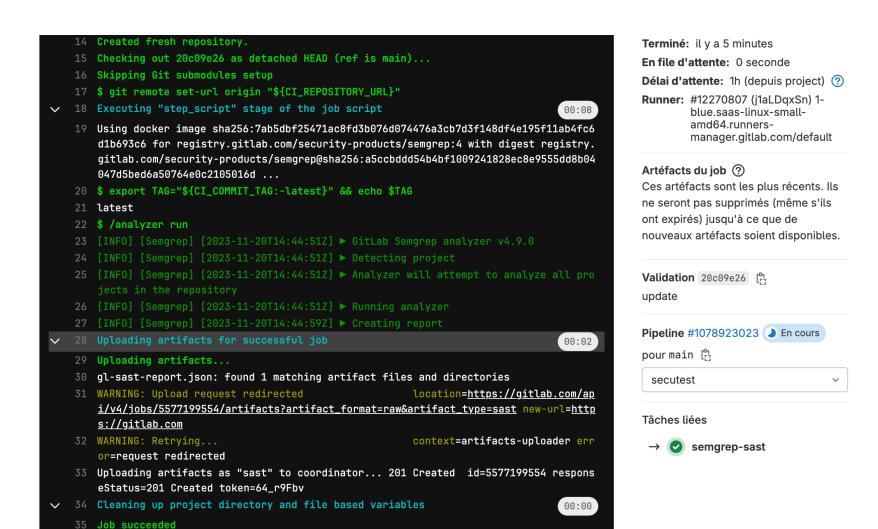
**USERNAME**: votre\_username

# **Pipeline**



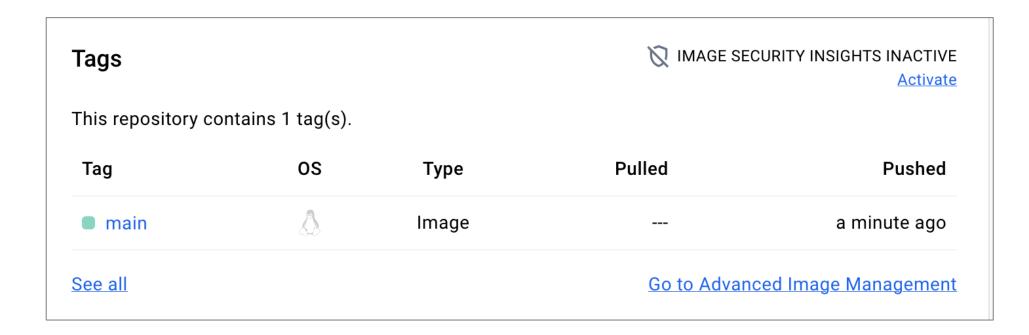


## **Jobs**



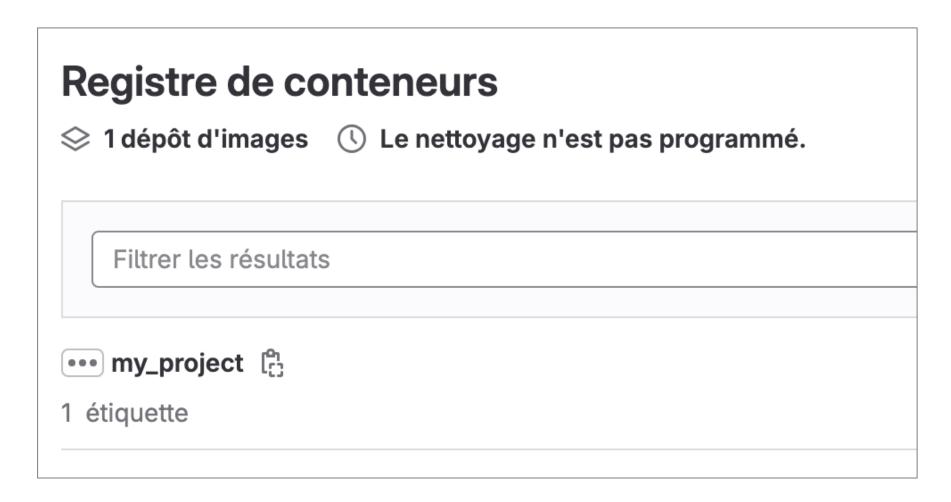
## Résultat Hub Docker

Hub: L'image est automatiquement poussé sur le Hub



# Résultat Gitlab Registry

Ainsi que sur le registre Gitlab



# **Exemple 1 : Stages de Test**

gitlab-ci.yml

```
stages: # étapes pour les jobs
      - test9
      - test10
 5
    #### -----####
    test_one:
        image: python:3.9
 9
        stage: "test9"
        before_script:
10
            - echo "Create Virtual Environment venv"
11
            - python -V # Print out python version for debugging
12
            - pip install virtualenv
13
14
            - virtualenv venv
15
            - echo "Activate venv"
            - source venv/bin/activate
16
17
18
        script:
            - echo "Install Fastapi & dependency"
19
20
            - cd app
21
            - pip install --upgrade pip
            - pip install -r requirements.txt
22
            - cd ..
23
            - sleep 10
24
           #- python main.py &
25
26
            #- sleep 10
27
            #- kill $!
```

Si besoin, activer Runners partagés dans Paramètres => CI/CD

# **Exemple 1 : Stages de Test**

```
31
    test_two:
32
        image: python:3.10
33
        stage: "test10"
34
        before_script:
35
             - echo "Create Virtual Environment venv"
36
             - python -V # Print out python version for debugging
37
             - pip install virtualenv
             - virtualenv venv
38
39
             - echo "Activate venv"
40
             - source venv/bin/activate
41
42
        script:
43
             - echo "Install Fastapi & dependency"
44
             - cd app
45
             - pip install --upgrade pip
46
             - pip install -r requirements.txt
47
             - cd ..
48
            - sleep 10
49
            #- python main.py &
50
            #- sleep 10
            #- kill $!
51
```

gitlab-ci.yml

gitlab-ci.yml

```
9 # --- GITLAB Artefact - Test / Build / Push Gitlab registry -----
10 variables: # variables de configuration budah
      PIP_CACHE_DIR: "$CI_PROJECT_DIR/.cache/pip"
11
      ARTIFACT_DIR: "./archive" # dossier des artifact - doit être relatif au dossier de build
12
13
      ARTIFACT_NAME: "build-example.tar"
      STORAGE_DRIVER: "vfs"
14
      BUILDAH_FORMAT: "docker"
15
16
    stages: # étapes pour les jobs
17
18
      - secutest
19
      - test
20
      - build
21
      - push
22
    default: # l'export de la variable TAG qui sera exécuté avant chaque job
23
24
      before_script:
        - export TAG="${CI_COMMIT_TAG:-latest}" && echo $TAG
25
26
27
    #### --- SecuTest ----####
28
29
    sast:
30
      stage: secutest
31
    include:
    - template: Security/SAST.gitlab-ci.yml
32
33
```

```
#### -----####
                                                                                                                                     gitlab-ci.yml
36
    test:
37
        image: python:3.10
38
        stage: "test"
        before_script:
39
40
           - echo "Create Virtual Environment venv"
           - python -V # Print out python version for debugging
41
42
           - pip install virtualenv
           - virtualenv venv
43
            - echo "Activate venv"
44
45
            - source venv/bin/activate
46
47
        script:
48
            - echo "Install Fastapi & dependency"
49
50
            - pip install --upgrade pip
51
            - pip install -r requirements.txt
52
            - cd ..
53
           - sleep 10
54
           #- python main.py &
55
           #- sleep 10
56
           #- kill $!
57
58
    #### -----####
59
                                                                     Variable interne Gitlab
    build:
60
      stage: "build" # étape de build du pipeline
61
62
      image: # image officielle de Buildah (latest)
63
       name: "quay.io/buildah/stable:latest"
      script: # builder et exporter l'image
64

    buildah bud (Build Using Dockerfile) pour construire l'image.

        - "buildah bud -f Dockerfile -t ${CI_REGISTRY_IMAGE}:${TAG} ."
65
        - "mkdir ${ARTIFACT_DIR}"
66
        - "buildah push ${CI_REGISTRY_IMAGE}:${TAG} docker-archive:${ARTIFACT_DIR}/${ARTIFACT_NAME}:${TAG}" ◀
                                                                                                                   buildah push.
67
68
      artifacts: # Gestion automatique de la récupération et du stockage des fichiers spécifiés par gitlab.com
69
        name: "archive:${ARTIFACT_NAME}:${CI_JOB_ID}" # nom
70
71
        when: "on_success" # si pas derreur
72
        expire_in: "6h" # suppression de l'artifact après 6h
73
        paths: # le chemin, relatif au dossier de build
          - "${ARTIFACT_DIR}/"
74
```

gitlab-ci.yml

```
#### -----####
79
80
    push:
      stage: "push" # étape de push du pipeline
81
      image: # utilise l'image officielle de Buildah
82
        name: "quay.io/buildah/stable:latest"
83
      only: # ne s'éxécute que sur la branche principale ou à la création d'un tag
84
        - "tags"
85
        - "main"
86
      script: # les commandes pour importer l'image depuis l'artifact et la publier
87
        - "buildah pull docker-archive:${ARTIFACT_DIR}/${ARTIFACT_NAME}"
88
        - "echo $CI_REGISTRY_PASSWORD | buildah login -u $CI_REGISTRY_USER --password-stdin $CI_REGISTRY"
89
        - "buildah push ${CI_REGISTRY_IMAGE}:${TAG}"
90
91
      dependencies:
        - "build" # explicite la dépendance avec le job précédent, et donc la récupération de l'artifact
92
93
```

```
gitlab-ci.yml
     #----- Generate Docker Image with Docker - Push to Docker hub -----#
97
     docker-build-master:
       # Official docker image.
98
99
       image: docker:latest
       stage: build
100
       services:
101
102
       - docker:dind
103
       before_script:
         - docker login -u "$USERNAME" -p "$TOKEN" $HUB
104
       script:
105
         - docker build --pull -t "$HUB_IMAGE" .
106
         docker push "$HUB_IMAGE"
107
108
       only:
109
         - master
110
111
     docker-build:
112
       # Official docker image.
       image: docker:latest
113
114
       stage: build
                                                Variable ajoutée dans « Gitlab Variables »
115
       services:
116
         - docker:dind
117
       before_script:
         - docker login -u "$USERNAME" -p "$TOKEN" $HUB
118
119
       script:
120
         - docker build --pull -t "$HUB_IMAGE:$CI_COMMIT_REF_SLUG" .
         - docker push "$HUB_IMAGE:$CI_COMMIT_REF_SLUG"
121
122
       except:
123
         - master
```