

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

Create repository


Namespace
username


Repository Name *
myprojectrepo

Short description

A short description to identify your repository. If the repository is public, this description is used to index your content on Docker Hub and in search engines, and is visible to users in search results.

Visibility

☐ Public 
Appears in Docker Hub search results

☒ Private 
Only visible to you

Cancel

Create

Pushing images

You can push a new image to this repository using the CLI:

```
docker tag local-image:tagname new-repo:tagname  
docker push new-repo:tagname
```

Make sure to replace `tagname` with your desired image repository tag.

<https://hub.docker.com/>

- **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 Docker
=> Security
=> New 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 !

New Access Token

A personal access token is similar to a password except you can have many tokens and revoke access to each one at any time. [Learn more](#)

Access Token Description *

Access permissions

Read, Write, Delete ▼

Read, Write, Delete tokens allow you to manage your repositories.

Cancel Generate

Token Docker Hub

Copy Access Token

When logging in from your Docker CLI client, use this token as a password. [Learn more](#)

ACCESS TOKEN DESCRIPTION

advancedvirtu

ACCESS PERMISSIONS

Read & Write

To use the access token from your Docker CLI client:

1. Run `docker login -u username`
2. At the password prompt, enter the personal access token.

dckr_



WARNING: This access token will only be displayed once. It will not be stored and cannot be retrieved. Please be sure to save it now.

Copy and Close

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	***** 	Tous (par défaut) 	 
HUB_IMAGE  Protégée Développée	***** 	Tous (par défaut) 	 
TOKEN  Protégée Développée	***** 	Tous (par défaut) 	 
USERNAME  Protégée Développée	***** 	Tous (par défaut) 	 

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

Pour `main`

dernier 5 jobs ⌚ En cours, mis en file d'attente pendant 1 secondes

Pipeline

Besoins

Jobs 5

Tests 0

secutest



semgrep-sast



test



test



build



build



docker-build



push



push



✓ Réussi

⌚ 00:21:28

📅 il y a 3 minutes

commit

#1078961550

🔗 main

🔗 42037a37



plus récent



Jobs

```
14 Created fresh repository.
15 Checking out 20c09e26 as detached HEAD (ref is main)...
16 Skipping Git submodules setup
17 $ git remote set-url origin "${CI_REPOSITORY_URL}"
18 Executing "step_script" stage of the job script 00:08
19 Using docker image sha256:7ab5dbf25471ac8fd3b076d074476a3cb7d3f148df4e195f11ab4fc6
d1b693c6 for registry.gitlab.com/security-products/semgrep:4 with digest registry.
gitlab.com/security-products/semgrep@sha256:a5ccbddd54b4bf1009241828ec8e9555dd8b04
047d5bed6a50764e0c2105016d ...
20 $ export TAG="${CI_COMMIT_TAG:-latest}" && echo $TAG
21 latest
22 $ /analyzer run
23 [INFO] [Semgrep] [2023-11-20T14:44:51Z] ► GitLab Semgrep analyzer v4.9.0
24 [INFO] [Semgrep] [2023-11-20T14:44:51Z] ► Detecting project
25 [INFO] [Semgrep] [2023-11-20T14:44:51Z] ► Analyzer will attempt to analyze all pro
jects in the repository
26 [INFO] [Semgrep] [2023-11-20T14:44:51Z] ► Running analyzer
27 [INFO] [Semgrep] [2023-11-20T14:44:59Z] ► Creating report
28 Uploading artifacts for successful job 00:02
29 Uploading artifacts...
30 gl-sast-report.json: found 1 matching artifact files and directories
31 WARNING: Upload request redirected location=https://gitlab.com/ap
i/v4/jobs/5577199554/artifacts?artifact_format=raw&artifact_type=sast new-url=http
s://gitlab.com
32 WARNING: Retrying... context=artifacts-uploader err
or=request redirected
33 Uploading artifacts as "sast" to coordinator... 201 Created id=5577199554 respons
eStatus=201 Created token=64_r9Fbv
34 Cleaning up project directory and file based variables 00:00
35 Job succeeded
```

Terminé: il y a 5 minutes


En file d'attente: 0 seconde

Délai d'attente: 1h (depuis project) ?


Runner: #12270807 (j1aLDqxSn) 1-
blue.saas-linux-small-
amd64.runners-
manager.gitlab.com/default

Artéfacts du job ?

Ces artéfacts sont les plus récents. Ils ne seront pas supprimés (même s'ils ont expirés) jusqu'à ce que de nouveaux artéfacts soient disponibles.

Validation 20c09e26 
update

Pipeline #1078923023  En cours

pour main 

secutest

Tâches liées

→  semgrep-sast



Résultat Hub Docker

Hub : L'image est automatiquement poussé sur le Hub

Tags

IMAGE SECURITY INSIGHTS INACTIVE [Activate](#)

This repository contains 1 tag(s).

Tag	OS	Type	Pulled	Pushed
 main		Image	---	a minute ago

[See all](#)[Go to Advanced Image Management](#)

Résultat Gitlab Registry

Ainsi que sur le registre Gitlab

Registre de conteneurs

 1 dépôt d'images  Le nettoyage n'est pas programmé.

Filtrer les résultats

 my_project 

1 étiquette

Exemple 1 : Stages de Test

gitlab-ci.yml

```
1 stages: # étapes pour les jobs
2   - test9
3   - test10
4
5
6 ##### ----- Test -----#####
7 test_one:
8   image: python:3.9
9   stage: "test9"
10  before_script:
11    - echo "Create Virtual Environment venv"
12    - python -V # Print out python version for debugging
13    - pip install virtualenv
14    - virtualenv venv
15    - echo "Activate venv"
16    - source venv/bin/activate
17
18  script:
19    - echo "Install Fastapi & dependency"
20    - cd app
21    - pip install --upgrade pip
22    - pip install -r requirements.txt
23    - cd ..
24    - sleep 10
25    #- python main.py &
26    #- sleep 10
27    #- kill $!
```

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

Exemple 1 : Stages de Test

gitlab-ci.yml

```
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
38         - virtualenv venv
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
51         #- kill $!
```


Exemple 2 : Test, Build, Push (buildah & Docker)

gitlab-ci.yml

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

Exemple 2 : Test, Build, Push (buildah & Docker)

gitlab-ci.yml

```
35 ##### ----- Test -----#####
```

```
36 test:
```

```
37   image: python:3.10
```

```
38   stage: "test"
```

```
39   before_script:
```

```
40     - echo "Create Virtual Environment venv"
41     - python -V # Print out python version for debugging
42     - pip install virtualenv
43     - virtualenv venv
44     - echo "Activate venv"
45     - source venv/bin/activate
```

```
46
47   script:
```

```
48     - echo "Install Fastapi & dependency"
49     - cd app
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 ##### ----- Build -----#####
```

```
60 build:
```

```
61   stage: "build" # étape de build du pipeline
```

```
62   image: # image officielle de Buildah (latest)
```

```
63     name: "quay.io/buildah/stable:latest"
```

```
64   script: # builder et exporter l'image
```

```
65     - "buildah bud -f Dockerfile -t ${CI_REGISTRY_IMAGE}:${TAG} ."
66     - "mkdir ${ARTIFACT_DIR}"
67     - "buildah push ${CI_REGISTRY_IMAGE}:${TAG} docker-archive:${ARTIFACT_DIR}/${ARTIFACT_NAME}:${TAG}"
```

```
68
69   artifacts: # Gestion automatique de la récupération et du stockage des fichiers spécifiés par gitlab.com
```

```
70     name: "archive:${ARTIFACT_NAME}:${CI_JOB_ID}" # nom
```

```
71     when: "on_success" # si pas d'erreur
```

```
72     expire_in: "6h" # suppression de l'artefact après 6h
```

```
73     paths: # le chemin, relatif au dossier de build
```

```
74     - "${ARTIFACT_DIR}/"
```

Variable interne Gitlab

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

buildah push.

Exemple 2 : Test, Build, Push (buildah & Docker)

gitlab-ci.yml

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

Exemple 2 : Test, Build, Push (buildah & Docker)

gitlab-ci.yml

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

Variable ajoutée dans « Gitlab Variables »

