# Exercices supplémentaires docker

## Table des matières

I - Exercice : Docker++ : Docker Compose	3
II - Exercice : Docker ++ : Gitlab Cl	4
Solutions des exercices	8

# I Exercice : Docker++ : Docker Compose

#### **Docker Compose**

Déployer des images manuellement c'est très drôle, mais dans la pratique on préfère automatiser tout cela. Pour cela, plusieurs technologies existent (ansible & co) mais on conseillera docker-compose. C'est un outil qui décrit l'architecture docker, quelles images déployer et comment elles s'interconnectent.

Ce tuto<sup>1</sup> est une bonne ressource pour se familiariser avec la philosophie et la syntaxe. Pour plus d'inspiration, les docker compose de Picasoft sont disponibles sur Gitlab<sup>2</sup> et le wiki de Picasoft<sup>3</sup> contient des instructions très complètes.

Le but de cet exercice est de redéployer l'entièreté du serveur par du docker-compose de telle manière à ce que l'on puisse le re-déployer sur un autre serveur en une seule commande.

<sup>1.</sup> https://www.educative.io/blog/docker-compose-tutorial

<sup>2</sup> https://gitlab.utc.fr/picasoft/projets/dockerfiles

<sup>3.</sup> https://wiki.picasoft.net/doku.php?id=technique:docker:start

# II Exercice : Docker ++ : Gitlab CI

#### Gitlab CI

Dans cet exercice, nous allons étudier l'utilisation de docker dans le cadre de l'intégration continue de Gitlab. Le principe de l'intégration continue est d'exécuter une série d'opérations (compilation, tests, documentation, publication) lorsqu'un commit Git est poussé sur Gitlab. Pour cela on va utiliser l'instance Gitlab de l'UTC<sup>4</sup>.

La CI fonctionne grâce à deux aspects :

- Un fichier .gitlab-ci.yml qui décrit la série d'opérations (aussi appelée pipeline) et comment la lancer
- Un *runner*, c'est-à-dire un container sur lequel tournera la suite d'opération

#### Description de la CI

La structure d'un fichier .gitlab-ci.yml est en YAML et ressemble à

```
1 image: debian
 2
 3 build-job:
 4 stage: build
5 tags: [docker]
 6 script:
      - echo "Hello, $GITLAB USER LOGIN!"
 7
 8
9 test-job1:
10 stage: test
11 tags: [docker]
   script:
12
13

    echo "This job tests something"

14
15 test-job2:
16 stage: test
   tags: [docker]
17
18 script:
     - echo "This job tests something, but takes more time than test-jobl."
19
20
      - echo "After the echo commands complete, it runs the sleep command for 20
  seconds"
21
      - echo "which simulates a test that runs 20 seconds longer than test-job1"
22
      - sleep 20
23
24 deploy-prod:
25 stage: deploy
26 tags: [docker]
27
    script:
      - echo "This job deploys something from the $CI COMMIT BRANCH branch."
28
```

Pour aller plus loin sur ce fichier, on peut se référer à la documentation en ligne<sup>5</sup>, et s'inspirer de ce qui est fait à Picasoft<sup>6</sup>.

Il est possible d'exporter des fichiers à chaque étape de la pipeline dans ce qu'on appelle des *artifacts*. Pour cela, il suffit d'énumérer les fichiers/dossiers à exporter dans un champ artifacts de la forme

4. https://gitlab.utc.fr

<sup>5.</sup> https://docs.gitlab.com/ee/ci/yaml/index.html

<sup>6.</sup> https://gitlab.utc.fr/picasoft/apis/init/pr-sentation/-/blob/master/.gitlab-ci.yml

```
1 projet:
2 stage: build
3 tags:
4
     - docker
5
    - latex
6 script:

    cd projet && latexmk -shell-escape -pdf

7
8 artifacts:
9
    paths:
10
        - projet/projet.pdf
      expire_in: 1 hour
11
```

Il existe finalement un espace spécial sur Gitlab nommé *Gitlab Pages*. Il s'agit d'un espace associé à chaque repository capable de servir des pages statiques (html, pdf, etc), qui est concrètement un mini-serveur statique hébergé sur les serveurs de Gitlab (ou de l'UTC dans notre cas). Il est possible d'y mettre des fichiers à l'aide de la CI avec quatre conditions :

- Avoir un job nommé pages dans le fichier .gitlab-ci.yml
- Qu'il soit à l'étape de déploiement
- Qu'il mette lesdits fichiers dans un dossier public
- Qu'il exporte ce dossier public

Si ces conditions sont réunies, un site de la forme <username>.gitlab.utc.fr/<projet> sera créé. La première fois qu'un *Gitlab Page* est crée, cela peut prendre jusqu'à 30 minutes.

```
1 pages:
 2 stage: deploy
 3 tags:
 4
    - linux
 5 script:
 6 - mkdir -p public
7
      - cp projet/projet.pdf public/
8 artifacts:
9
    paths:
10
      - public
11
    expire_in: 1 year
```

#### Utilisation de Runners

Un runner est un programme qui met à disposition la puissance de calcul d'une machine réelle (un VPS, votre machine personnelle, n'importe quel ordinateur) à Gitlab. Installer un runner sur une machine Linux revient simplement à exécuter la bonne image docker en suivant ce tuto<sup>7</sup>. Il existe d'autres façons de faire mais celle ci est de loin la plus simple.



Exercice : Docker ++ : Gitlab CI

Pour éviter que tout le monde ai à déployer son propre VPS, l'UTC propose un Shared Runner, un runner accessible à n'importe qui à l'UTC mais qui peut être un peu lent car partagé entre tous les utilisateurs.

On peut également spécifier son propre runner, en théorie bien rapide et disponible juste à côté. Pour cela, on peut suivre le tut récupérer le token spécifique au projet dans la configuration repository.

#### Jobs

Lorsqu'un commit est poussé sur Gitlab et que le repository contient un .gitlab-ci.yml, un pipeline va automatique se lancer. On peut retrouver les pipelines courants et passés dans le menu CI/CD.

#### **Question 1**

Créer un repository sur gitlab.utc.fr dans lequel vous ajouterez un fichier .gitlab-ci.yml gui reprend l'exemple précédent. Ajouter dans les paramètres du repository le shared runner afin que la CI puisse se dérouler dessus.

**Question 2** 

**Question 3** 

Déployer un runner sur votre VPS, le sélectionner dans les paramètres du repository et vérifier que la CI se déroule bien dessus.

Utilisez les Gitlab Pages pour mettre en ligne une page html ou un pdf de votre choix.

Djøbe for Gitl, ab. How do i configure runnend er nunners as separafe users, on separate se	wars, and on your local machine. Runners are either:		6
	Alternative second		
	These runners These runners are shared asnoss this Gill, ab instance. The same shared runner executes code from multiple project	ts, unless you configure autoscaling with MaxBuilds set	
ore. De é neve one. picarisons foit, instael Gité, ab Runnee:	to 1 pekiek k kon Gitak con). Enable where frames for this project Con Available shared runners: 1 er (32a/x02)		
	Croup runners These runners are shared at mass projects in this project. Once yours runners in the managed with the human Alfe, Dealed group comess		
	Available group runners: 1		
plus to <sup>8</sup> et	Set up a specific runner mar 1. Install GitLab Runner a 2. Register the runner with https://gitlab.utc And this registration to VMEM562DP2MPH6spax Reset registration token Show Runner installation in	ually and ensure it's running. h this URL: fr/ 다 다 ken: q4 다 structions	
n du	Other available runne	ers	
	#158 (z4M7GB4s)	Enable for this proje	ct
	test_velodyne_utils docker linux sivalab_deploy		
	<b>a</b> #16 (c81ea865)	Enable for this proje	oct
	- · ·	chable for this proje	
	laver		

[solution n°2 p. 8]

[solution n°1 p. 8]

[solution n°3 p. 8]

#### **Question 4**

Il est de bonne pratique de ne pas mettre en clair ses identifiants et mots de passe dans les scripts. Pour cela Gitlab propose un système de variables. En vous aidant de la documentation<sup>9</sup>, remplacer dans le fichier .gitlab-ci.yml précédent les identifiants, hôte et mot de passes par des variables que vous aurez préalablement créé dans les paramètres du repository.

1 11 111	des					
faciables	store information, like passwords and secret keys, that you can use in	job scripts. Learn more.				
fariables	can be:					
· Po Ha	etected i. Only exposed to protected branches or tags. skell – Hidden in job logs. Must match masking requirements. Learn n	sore.				
Туре	7 Key	Value	Protected	Masked	Environments	
foriable	180H_PTR		×	~	All (default)	
foriable	SFTP_PASS	****************	×	~	All (default)	
foriable foriable	SFTP_PA8S SFTP_PORT		×	×	All (default) All (default)	

## Solutions des exercices

Solution n°1	[exercice p. 6]
Solution n°2	[exercice p. 6]
Solution n°3	[exercice p. 6]
Solution n°4	[exercice p. 7]

Solutions des exercices