



VILLAGE DE L'EMPLOI

ISOSET SA

61 rue Henri Barbusse 92110, CLICHY

Tél. : +33 1 83 62 82 65 (L.G.) / Fax : +33 1 55 48 09 18

info@isaset.fr

SPECIALITE : DEVOPS

Compte Rendu

PROJET DOCKER

Préparé par: JEAN THIBAUT

Formateur: TAFEN

PROJET DOCKER

L'objectif de ce mini-projet docker est de nous permettre d'implémenter tout ce que nous avons appris sur l'écosystème micro-service, conteneurisation, docker-compose.

C'est un projet très intéressant car issu des besoins d'entreprise.

On nous demande dans ce projet de conteneuriser une application écrite en python, et ensuite de la déployer sur docker avec un frontend en PHP et un backend qui est fait en python et qui tourne sur flask.

Dans un premier temps on nous demande de Builder et tester l'appli (en écrivant le DOCKERFILE).

Ensuite après avoir écrit le DOCKERFILE pour conteneuriser l'application, on va tester l'appli avec un Curl.

Ensuite on va mettre en place la partie Infrastructure As Code pour de déploiement facile de notre application et la reproduire ce déploiement sur diverses autres infrastructures. Dans ce cas on va modifier un certain nombre de fichiers.

Par la suite on va créer un docker registry qui va nous permettre d'héberger l'image du client que nous avons produite en occurrence l'image de student list.



VILLAGE DE L'EMPLOI

ISOSET SA

61 rue Henri Barbusse 92110, CLICHY

Tél. : +33 1 83 62 82 65 (L.G.) / Fax : +33 1 55 48 09 18

info@isohet.fr

CLONER LE REPO DISTANT OU SE TROUVE LE PROJET

```
$ git clone https://github.com/diranetafen/student-list
```

```
Jean-Thibaut@LAPTOP-EV0GKLCC MINGW64 ~/OneDrive/Bureau/DEVOPS [42 45]k/DOCKER/DOCKER PROJET/REPO
$ git clone https://github.com/diranetafen/student-list
Cloning into 'student-list'...
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 31 (delta 6), reused 3 (delta 3), pack-reused 17
Receiving objects: 100% (31/31), 10.03 KiB | 855.00 KiB/s, done.
Resolving deltas: 100% (7/7), done.
```

IHM : interface homme-machine

PARTIE 1 : BUILD / TEST.

Tout va partir d'une machine qui contient docker, nous avons une application qui est un script développé en flask et stockée sur un GITHUB.

Du coup on devra télécharger l'application en local, par la suite on pourra créer un dockerfile qui lui va Builder une image pour pouvoir instancier un container de cette application (student-list).

PARTIE 2 : INFRASTRUCTURE AS CODE.

Ici nous allons écrire un docker-compose qui va nous permettre d'automatiser la création de l'application.

A cela rajouter une deuxième appli qui sera l'IHM (interface HOMME-MACHINE) qui va taper le container student-list afin de pouvoir visualiser le rendu via un navigateur.

PARTIE 3 : DEPLOYER UN REGISTRE PRIVE.

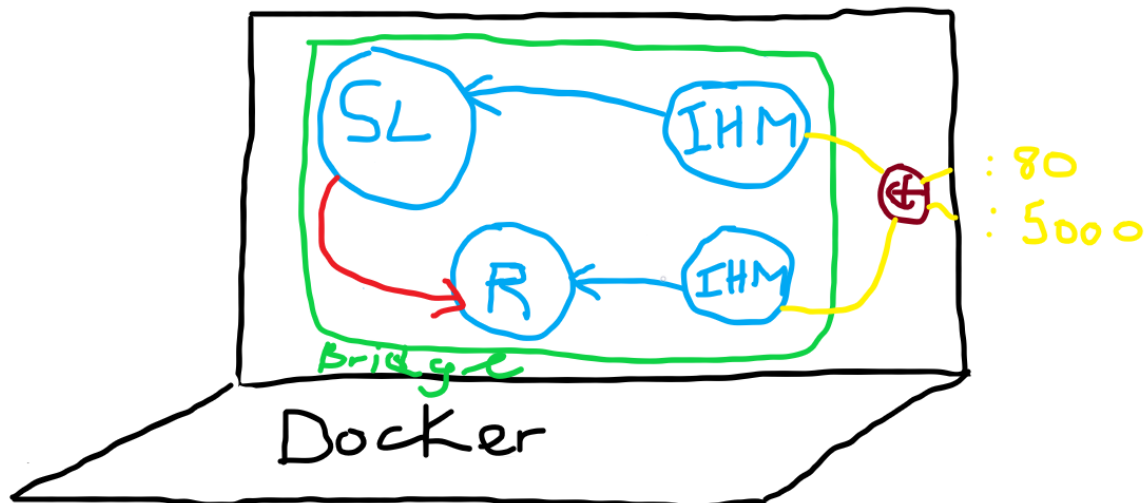
Déployer un 3^{ème} container qui aura pour rôle qui aura pour rôle d'être un registre privé pour notre image que nous aurons Buildé.

Pareillement pour pouvoir tester cela on nous demande de rajouter une IHM pour le registre qui elle aura pour rôle de

venir taper le registre afin qu'on le consulter via un navigateur.

Logiquement pour que tous ces containers puissent interagir il faudrait qu'ils soient tous dans un même réseau docker. Nous on va partir sur un réseau de type BRIDGE.

Pour exposer nos applications vers l'extérieur, il faudra mettre des règles d'expositions. Nos applications doivent être accessibles via l'IHM du coup nous allons exposer que nos containers IHM




1ère étape : Nous allons déployer notre machine qui contient docker.

Nous allons travailler sur VirtualBox et utiliser vagrant pour le déploiement.



`cursus-devops / vagrant / docker /`

 `ulrichmonji` fix zsh

This branch is up to date with `diranetafen/cursus-devops:master`.

Name	Last commit
..	
 Vagrantfile	add zsh on
 install_docker.sh	fix zsh

Par la suite nous allons copier ces deux fichiers avec les extensions qui vont avec dans le répertoire de notre projet

Mini Projet Docker				
Nom	Modifié le	Type	Taille	
 install_docker.sh	10/05/2023 00:58	sh_auto_file	2 Ko	
 Vagrantfile	10/05/2023 00:55	Fichier	1 Ko	

Et par la suite nous allons l'ouvrir avec Visual studio code

Une fois dans visual studio code on fait :

Vagrant up --provision : cette commande va nous permettre de créer notre machine docker dans VirtualBox.

Une fois la machine créée on fait : **uptime** pour avoir les infos sur la machine créée.

Ensuite pour pouvoir se connecter on fait un **Vagrant ssh**

```
docker:
docker: =====
docker:
docker: Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service to /
docker:  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
docker:                                Dload  Upload  Total  Spent  Left  Speed
0      0     0      0      0      0      0      0  --:--:-- --:--:-- --:--:--     0
100 12.1M 100 12.1M    0      0  846k      0  0:00:14 0:00:14 --:--:--  847k
docker: The zsh is not installed on this server
docker: For this Stack, you will use 192.168.56.3 IP Address
PS C:\Users\ndjin\OneDrive\Bureau\Mini Projet Docker> vagrant ssh
[vagrant@docker ~]$
```

A cette étape nous allons télécharger le repo GIT du projet dans notre machine DOCKER déployée. Mais avant il faut installer GIT.

SUDO YUM INSTALL GIT -Y

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
[vagrant@docker ~]$ sudo yum install git -y
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
epel/x86_64/metalink
* base: mirror.in2p3.fr
* epel: mirror.in2p3.fr
* extras: mirror.in2p3.fr
* updates: mirror.in2p3.fr
base
docker-ce-stable
epel
extras
updates
(1/2): epel/x86_64/updateinfo
(2/2): epel/x86_64/primary_db
Package git-1.8.3.1-24.el7_9.x86_64 already installed and latest version
Nothing to do
[vagrant@docker ~]$
```

Une fois GIT installé on peut faire un **GIT CLONE (link_repo)**

```
Nothing to do
[vagrant@docker ~]$ git clone https://github.com/jean-thibaut/student-list.git
Cloning into 'student-list' ...
remote: Enumerating objects: 31, done.
remote: Counting objects: 100% (14/14), done.
remote: Compressing objects: 100% (10/10), done.
remote: Total 31 (delta 6), reused 3 (delta 3), pack-reused 17
Unpacking objects: 100% (31/31), done.
[vagrant@docker ~]$
```


PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[vagrant@docker ~]$ ll
total 20
-rw-r--r--. 1 root    root    18565 May 10 00:08 get-docker.sh
drwxrwxr-x. 5 vagrant vagrant  94 May 10 12:38 student-list
[vagrant@docker ~]$
```

On va se déplacer dans notre répertoire de travail student-list

```
[vagrant@docker ~]$ ll
total 20
-rw-r--r--. 1 root    root    18565 May 10 00:08 get-docker.sh
drwxrwxr-x. 5 vagrant vagrant  94 May 10 12:38 student-list
[vagrant@docker ~]$ cd student-list/
[vagrant@docker student-list]$ ls
docker-compose.yml  README.md  simple_api  website
[vagrant@docker student-list]$ ll
total 8
-rw-rw-r--. 1 vagrant vagrant   0 May 10 12:38 docker-compose.yml
-rw-rw-r--. 1 vagrant vagrant 7133 May 10 12:38 README.md
drwxrwxr-x. 2 vagrant vagrant  70 May 10 12:38 simple_api
drwxrwxr-x. 2 vagrant vagrant  23 May 10 12:38 website
[vagrant@docker student-list]$
```

Nous avons à cette étape mis en place notre environnement de travail en déployant notre machine docker avec Vagrant, une fois cela fait nous nous sommes connectés à notre machine docker (vagrant ssh). Une fois dans notre machine docker, nous avons installé GIT (sudo yum Install GIT -y) pour par la suite faire un :

Git clone (<https://github.com/diranetafen/student-list>) pour récupérer le repo du projet.



VILLAGE DE L'EMPLOI

ISOSET SA

61 rue Henri Barbusse 92110, CLICHY

Tél. : +33 1 83 62 82 65 (L.G.) / Fax : +33 1 55 48 09 18

info@isohet.fr

A ce niveau nous allons donc commencer la construction du Dockerfile.

En suivant les instructions données dans l'énoncé nous avons construit le dockerfile ci-dessous.

YUM est le gestionnaire de package de CentOS

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL
GNU nano 2.3.1                                     File: Dockerfile

FROM python:2.7-buster
MAINTAINER JEAN THIBAUT PIOBLI
ADD student_age.py /
RUN apt-get update -y && apt-get install python-dev python3-dev libsasl2-dev python-dev libldap2-dev libssl-dev -y
RUN pip install flask==1.1.2 flask_httpauth==4.1.0 flask_simpleldap python-dotenv==0.14.0
VOLUME [ "/data" ]
EXPOSE 5000
CMD [ "python", "./student_age.py" ]
```

```
Code  Blame  9 lines (8 loc) · 358 Bytes  Raw  Copy  Download  Edit  More

1  FROM python:2.7-buster
2  LABEL maintainer="Ulrich MONJI" email="toto@admin.fr"
3  RUN apt update -y && apt install python-dev python3-dev libsasl2-dev python-dev libldap2-dev libssl-dev -y
4  RUN pip install flask==1.1.2 flask_httpauth==4.1.0 flask_simpleldap python-dotenv==0.14.0
5  COPY student_age.py /
6
7  VOLUME /data
8  EXPOSE 5000
9  CMD [ "python", "./student_age.py" ]
```

Et donc après avoir construit le dockerfile on fait un

DOCKER BUILD -T API-POZOS:1 .

```
[vagrant@docker simple_api]$ docker build -t api-pozos:1 .
[+] Building 536.1s (9/9) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 447B
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load metadata for docker.io/library/python:2.7-buster
=> [internal] load build context
=> => transferring context: 1.77kB
```


DOCKER SYSTEM DF

```
[vagrant@docker simple_api]$ docker system df
```

TYPE	TOTAL	ACTIVE	SIZE	RECLAIMABLE
Images	1	0	1.119GB	1.119GB (100%)
Containers	0	0	0B	0B
Local Volumes	0	0	0B	0B
Build Cache	15	0	2.016kB	2.016kB

```
[vagrant@docker simple_api]$
```

On peut remarquer dans la capture que notre Build n'a pas pris en compte le VOLUME que nous avons déclaré dans le dockerfile.

Il nous était demandé de créer un fichier data à la racine de notre machine docker et la considérer comme volume car c'est là que vont être stockés les données de notre application (student_age.json)

Face à cette situation, nous avons donc de nous même créé ce volume dans la racine.

DOCKER VOLUME CREATE DATA

```
[vagrant@docker /]$ docker volume create data
data
[vagrant@docker /]$ ls
bin boot dev etc home lib lib64 media mnt opt proc root run
[vagrant@docker /]$ docker volume ls
DRIVER      VOLUME NAME
local       data
[vagrant@docker /]$ docker volume inspect data
[
  {
    "CreatedAt": "2023-05-12T00:07:08Z",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/data/_data",
    "Name": "data",
    "Options": null,
    "Scope": "local"
  }
]
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

[vagrant@docker simple_api]$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
api-pozos     1         e073ededbbb5  13 hours ago  1.12GB
[vagrant@docker simple_api]$
```

Maintenant que nous avons notre image il faut la tester mais avant il faut faire un run qui lancer un container.

Pour tester nous allons nous créer un réseau, nous allons déjà commencer par travailler dans un réseau. Créons un réseau **pozos**.

```
[vagrant@docker simple_api]$ docker network create pozos
6e7c437d102d650d9b6adb5d9207289b8b790e9129fc7428656bbe2b12be262
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$
[vagrant@docker simple_api]$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
9bc4f6782005    bridge    bridge       local
a43ea093183c    host      host         local
1b2f1d956e24    none      null         local
6e7c437d102d    pozos     bridge       local
```

Avec la commande ci-dessous nous allons pouvoir déployer le container.

```
Docker run -d --network pozos --name test-api-pozos
-v ${PWD}/student-age.json:/data/student-age.json
-p 4000:5000 api-pozos:1
```

```
[vagrant@docker simple_api]$ docker run -d --network pozos --name test-api-pozos -v ${PWD}/student_age.json:/data/student_age.json -p 4000:5000 api-pozos:1
bd096d1935462be79c7780820b6a76b9a81210aba96552893530802f1828d6ad
[vagrant@docker simple_api]$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
bd096d193546   api-pozos:1 "python ./student_ag..." 21 seconds ago Up 18 seconds 0.0.0.0:4000->5000/tcp, :::4000->5000/tcp test-api-pozos
```

Une fois le container qui tourne, il faut le tester.

```
Curl -u toto:python -X GET http://<host IP>:<API exposed
port>/pozos/api/v1.0/get_student_ages
```

```
[vagrant@docker simple_api]$ curl -u toto:python -X GET http://192.168.56.3:4000/pozos/api/v1.0/get_student_ages
{
  "student_ages": {
    "alice": "12",
    "bob": "13"
  }
}
[vagrant@docker simple_api]$
```

On a notre application qui nous répond, le script student-age.json

PARTIE 2 : INFRASTRUCTURE AS CODE

Il est question dans cette partie de pouvoir mettre en place un docker-compose qui va nous permettre de pouvoir automatiser le déploiement de notre infrastructure.

```
Version: '3'
services:
  web-pozos:
    image: php:apache
    depends_on :
      - api-pozos
    ports:
      - "8082:80"
    volumes:
      - ./website:/var/www/html
    environment:
      - USERNAME=toto
      - PASSWORD=python
    networks:
      - api-pozos

  api-pozos:
    image: api-pozos:1
    ports:
      - "4000:5000"
    volumes:
      - ./simple_api/student_age.json:/data/student_age.json
    networks:
      - api-pozos

networks:
  api-pozos:
```

DOCKER-COMPOSE UP -D

```
Go Run Terminal Help install_docker.sh - Mini Projet Docker - Visual Studio Code
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
[vagrant@docker student-list]$ echo "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)"
https://github.com/docker/compose/releases/download/1.29.2/docker-compose-Linux-x86_64
[vagrant@docker student-list]$
```



```
[vagrant@docker student-list]$ docker-compose up -d
Creating network "student-list_api-pozos" with the default driver
Pulling web-pozos (php:apache)...
apache: Pulling from library/php
9e3ea8720c6d: Pull complete
07353b772b5e: Pull complete
5908153120ba: Pull complete
8681ad2eeea6: Pull complete
92711ce78973: Pull complete
bf1c5be6427e: Pull complete
1d02a81768ed: Pull complete
e06f9186afb2: Pull complete
e8efb75efb64: Pull complete
a79bde90684a: Pull complete
6cca2fbecbc2: Pull complete
42b682eaf8ed: Pull complete
ddec5f413e8f: Pull complete
Digest: sha256:6ddd79964fb6085e44183d4ca8e56e944ba1dec94deb054bd20e3fce04ef9772
Status: Downloaded newer image for php:apache
Creating student-list_api-pozos_1 ... done
Creating student-list_web-pozos_1 ... done
```

```
[vagrant@docker student-list]$ docker container ls
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS                               NAMES
fcfcb9f6dcfd   php:apache    "docker-php-entrypoi..." 8 minutes ago  Up 7 minutes  0.0.0.0:8082->80/tcp, :::8082->80/tcp  student-list_web-pozos_1
6d53aa5d97af   api-pozos:1   "python ./student_ag..." 8 minutes ago  Up 7 minutes  0.0.0.0:4000->5000/tcp, :::4000->5000/tcp  student-list_api-pozos_1
[vagrant@docker student-list]$
```



← → ↻ **http://192.168.56.3:8082**

Gmail YouTube Maps FUN Maîtriser le shell Ba... | Udemv Justificati

Student Checking App

List Student

IP IHM

Notre IHM écoute sur le port :8082 de l'extérieur redirigé vers le port 80 de notre container.



VILLAGE DE L'EMPLOI

ISOSET SA

61 rue Henri Barbusse 92110, CLICHY

Tél. : +33 1 83 62 82 65 (L.G.) / Fax : +33 1 55 48 09 18

info@isohet.fr

Warning: file_get_contents(http://<api_ip_or_name:port>/pozos/api/v1.0/get_student_ages): Failed to open stream: operation failed in /var/www/html/index.php on line 30

This is the list of the student with age

Warning: Trying to access array offset on value of type null in /var/www/html/index.php on line 32

Warning: foreach() argument must be of type array/object, null given in /var/www/html/index.php on line 32

NANO WEBSITE/INDEX.PHP

```
$url = 'http://<api_ip_or_name:port>/pozos/api/v1.0/get_student_ages';  
$list = json_decode(file_get_contents($url, false, $context), true);  
echo "<p style='color:red;; font-size: 20px;'>This is the list of the student with age</p>";  
foreach($list["student_ages"] as $key => $value) {  
    echo "- $key is $value years old <br>";  
}  
}
```

```
));  
$url = 'http://student-list_api-pozos_1[:5000]/pozos/api/v1.0/get_student_ages';  
$list = json_decode(file_get_contents($url, false, $context), true);  
echo "<p style='color:red;; font-size: 20px;'>This is the list of the student with age</p>";  
foreach($list["student_ages"] as $key => $value) {
```

← → ↻ ⚠ Non sécurisé | 192.168.56.3:8082

Gmail YouTube Maps FUN Maîtriser le shell Ba... | Udemy

Student Checking App

List Student

This is the list of the student with age

- alice is 12 years old
- bob is 13 years old

PARTIE 3 : DOCKER REGISTRY

Commençons par déployer le registre