

SERVEUR WEB

Docker Compose facilite le déploiement et la gestion d'applications multi-conteneurs en utilisant une syntaxe simple et déclarative. J'ai décidé de créer un docker-compose pour déployer le serveur web.

Ce fichier "docker-compose.yml" permet de définir et de gérer trois services interconnectés en une seule commande. Je l'exécute avec **docker-compose up** dans le même répertoire que le fichier "docker-compose.yml" pour lancer l'application et démarrer les conteneurs.

```
haproxy:
  image: haproxy:latest
  ports:
    - "8080:80"
  volumes:
    - ./haproxy.cfg:/usr/local/etc/haproxy/haproxy.cfg
  restart: always

wordpress:
  image: wordpress
  links:
    - mariadb:mysql
  environment:
    - WORDPRESS_DB_PASSWORD=changecomplexpassword777%
    - WORDPRESS_DB_USER=root
  ports:
    - "80:80"
  volumes:
    - ./html:/var/www/html
  restart: always

mariadb:
  image: mariadb
  environment:
    - MYSQL_ROOT_PASSWORD=changecomplexpassword777%
    - MYSQL_DATABASE=wordpress
  volumes:
    - mariadb_data:/var/lib/mysql
  restart: always
```

haproxy : Cette section du fichier définit le service **haproxy** qui utilise l'image Docker **haproxy:latest**. Il mappe le port 8080 de l'hôte sur le port 80 du conteneur HAProxy. Le fichier de configuration **haproxy.cfg** local est monté en tant que volume dans le conteneur.

/etc/haproxy/haproxy.conf*Global*

```
log /dev/log      local0
log /dev/log      local1 notice
chroot /var/lib/haproxy
stats socket /run/haproxy/admin.sock mode 660 level admin expose-fd listeners
stats timeout 30s
user haproxy
group haproxy
daemon

# Default SSL material locations
ca-base /etc/ssl/certs
crt-base /etc/ssl/private

# See:
https://ssl-config.mozilla.org/#server=haproxy&server-version=2.0.3&config=intermediate
ssl-default-bind-ciphers
ECDHE-ECDSA-AES128-GCM-SHA256:ECDHE-RSA-AES128-GCM-SHA256:ECDHE-ECDSA-AES
256-GCM-SHA384:ECDHE-RSA-AES256-GCM-SHA384:ECDHE-ECDSA-CHACHA20-POLY1305:E
CDHE-RSA->
ssl-default-bind-ciphersuites
TLS_AES_128_GCM_SHA256:TLS_AES_256_GCM_SHA384:TLS_CHACHA20_POLY1305_SHA25
6
ssl-default-bind-options ssl-min-ver TLSv1.2 no-tls-tickets
```

defaults

```
log      global
mode     http
option   httplog
option   dontlognull
timeout  connect 5000
timeout  client 50000
timeout  server 50000
errorfile 400 /etc/haproxy/errors/400.http
errorfile 403 /etc/haproxy/errors/403.http
errorfile 408 /etc/haproxy/errors/408.http
errorfile 500 /etc/haproxy/errors/500.http
errorfile 502 /etc/haproxy/errors/502.http
errorfile 503 /etc/haproxy/errors/503.http
errorfile 504 /etc/haproxy/errors/504.http
```

frontend http-in

```
bind 192.168.20.54:8080
default_backend wordpress-backend
```

backend wordpress_server

```
balance roundrobin
server wordpress1 192.168..20.61:80 check
server wordpress1 192.168..20.62:80 check
```

Ces configurations permettent à HAProxy de recevoir les connexions entrantes sur le port 8080 et de les répartir de manière équilibrée vers les serveurs WordPress **wordpress1** et **wordpress2** spécifiés dans le backend.

Lorsque le serveur frontend est down et qu'un des deux serveur backend aussi, je peux quand même joindre webgui.sae.jing.fr.

Cependant en tapant l'adresse webgui.sae.jing.fr, je n'ai aucune réponse

```
root@test:/home/test# curl -I webgui.sae.jing.fr:8080
HTTP/1.0 503 Service Unavailable
cache-control: no-cache
content-type: text/html

root@test:/home/test# curl -I webgui.sae.jing.fr:8080
HTTP/1.1 301 Moved Permanently
date: Wed, 14 Jun 2023 14:17:50 GMT
server: Apache/2.4.56 (Debian)
x-powered-by: PHP/8.0.28
x-redirect-by: WordPress
location: http://webgui.sae.jing.fr/
content-type: text/html; charset=UTF-8
```

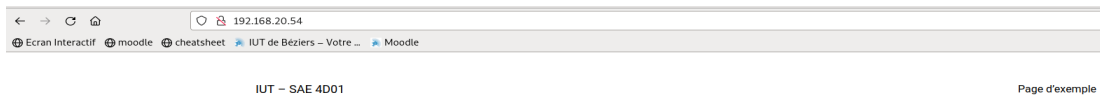
wordpress : Cette section du fichier définit le service **wordpress** qui utilise l'image Docker **wordpress**. Il utilise également le service mariadb en tant que dépendance et se connecte à celui-ci en utilisant le lien **mariadb:mysql**. Il configure les variables d'environnement pour définir le mot de passe de la base de données (**WORDPRESS_DB_PASSWORD**) et l'utilisateur de la base de données (**WORDPRESS_DB_USER**). Le port 80 du conteneur WordPress est mappé sur le port 80 de l'hôte, et le répertoire **html** local est monté en tant que volume pour stocker les fichiers HTML du site WordPress.

mariadb : Cette section du fichier définit le service **mariadb** qui utilise l'image Docker **mariadb**. Il configure les variables d'environnement pour définir le mot de passe root de la base de données (**MYSQL_ROOT_PASSWORD**) et le nom de la base de données (**MYSQL_DATABASE**). Le volume **mariadb_data** est monté pour persister les données de la base de données MariaDB.

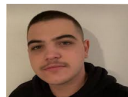
restart: always : Cette option spécifie que tous les services doivent être redémarrés automatiquement en cas d'arrêt.

docker-compose up

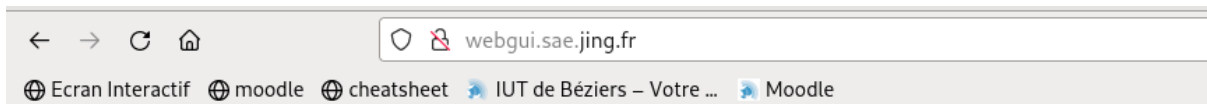
```
haproxy_1 | -x <unix_socket> get listening sockets from a unix socket
haproxy_1 | -S <bind>[,<bind options>...] new master CLI
haproxy_1 |
wordpress_1 | 192.168.20.22 - - [14/Jun/2023:12:14:10 +0000] "GET / HTTP/1.1" 2
00 10029 "-" "Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/1
02.0"
```



IUT – SAE 4D01



Après la résolution DNS, j'ai accès au site avec l'adresse webgui.sae.jing.fr.



IUT – SAE 4D01

IUT – SAE 4D01

SERVEUR GITLAB

Sur la VM qui contiendra le container pour le serveur gitlab, j'installe docker puis je télécharge l'image Docker de la dernière version de GitLab Community Edition avec la commande docker suivante:

docker pull gitlab/gitlab-ce:latest

J'exécute le container

***docker run --detach \
--publish 443:443 --publish 80:80 --publish 2222:22 \
--name gitlab \
--restart always \
--volume \$GITLAB_HOME/config:/etc/gitlab \
--volume \$GITLAB_HOME/logs:/var/log/gitlab \
--volume \$GITLAB_HOME/data:/var/opt/gitlab \
--shm-size 256m \
gitlab/gitlab-ce:latest***

Le container redémarre automatiquement s'il est arrêté pour n'importe quelle raison grâce à **--restart always**


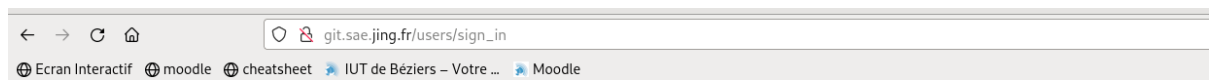
--volume \$GITLAB_HOME/config:/etc/gitlab: Montage d'un volume Docker pour stocker la configuration de GitLab. **\$GITLAB_HOME** est la variable d'environnement qui représente le répertoire de destination sur l'hôte.

--volume \$GITLAB_HOME/logs:/var/log/gitlab: Montage d'un volume Docker pour stocker les journaux de GitLab.

--volume \$GITLAB_HOME/data:/var/opt/gitlab: Montage d'un volume Docker pour stocker les données de GitLab.

```
Unable to find image 'gitlab/gitlab-ee:latest' locally
latest: Pulling from gitlab/gitlab-ee
d1669123f281: Pull complete
071f9bcad8ed: Extracting 18.64MB
e79eb07f7f66: Download complete
81a6135ab188: Download complete
alde0a9e3554: Download complete
4ba08eee1cd2: Download complete
3c575f695ab3: Download complete
7dc5cb0e44af: Downloading 47.51MB/1.436GB
```

J'y ai d'abord accès depuis l'adresse ip de la VM puis avec la résolution DNS avec le lien créé.



GitLab Édition Communautaire

Nom d'utilisateur ou adresse de courriel

Mot de passe

☐ Se souvenir de moi [Mot de passe oublié ?](#)

Connexion

[Vous n'avez pas encore de compte ? Inscrivez-vous maintenant](#)

Une fois un compte créé, ce message apparaît

 You have signed up successfully. However, we could not sign you in because your account is awaiting approval from your GitLab administrator. 

Le compte se crée mais impossible de se connecter. L'administrateur bloque les nouvelles connexions. Cependant je ne connais pas les identifiants de l'administrateur donc je les modifie directement dans le container:

gitlab-rake "gitlab:password:reset"

```
root@192:/# gitlab-rake "gitlab:password:reset"
Enter username: root
Enter password:
Confirm password:
Password successfully updated for user with username root.
```

Je me connecte en administrateur



GitLab Édition Communautaire

Nom d'utilisateur ou adresse de courriel

root

Mot de passe

●●●●●●●●●●●●●●●●

☐ Se souvenir de moi

[Mot de passe oublié ?](#)

Connexion

Vous n'avez pas encore de compte ?

[Inscrivez-vous maintenant](#)

Puis dans les settings de décoche la case “Require admin approval for new sign-ups”

Sign-up restrictions

Collapse

Configure the way a user creates a new account.

☒ Sign-up enabled

Any user that visits http://192.168.20.55/users/sign_in can create an account.

☐ Require admin approval for new sign-ups

Any user that visits http://192.168.20.55/users/sign_in and creates an account must be explicitly approved by an administrator before they can sign in. Only effective if sign-ups are enabled.

Je vois les utilisateurs et les utilisateurs peuvent maintenant se connecter.

Admin Area > Users

Users

Cohorts

Active 4

Admins 1

2FA Enabled 0

2FA Disabled 5

External 0

Blocked 0

Banned 0

Pending approval 0

Deactivated 0





Without >

New user

Q Search by name, email, or username

Sort by

Last created v

Name	Projects	Groups	Created on	Last activity	
 Ilker Onay ilker.onay@etu.umontpellier.fr	0	0	Jun 14, 2023	Never	<div>Edit</div> <div></div>
 Julien Alleaume julien.alleaume@etu.umontpellier.fr	0	0	Jun 14, 2023	Jun 14, 2023	<div>Edit</div> <div></div>
 Guilhem Mas guilhem.mas@etu.umontpellier.fr	0	0	Jun 12, 2023	Jun 14, 2023	<div>Edit</div> <div></div>
 Administrator Admin It's you! admin@example.com	0	0	Jun 12, 2023	Jun 14, 2023	<div>Edit</div> <div></div>

SERVEUR SCODOC

- J'ajoute le dépôt scodoc. Je crée le fichier scodoc.list dans /etc/apt/sources.list.d/ contenant juste cette ligne:

deb http://scodoc.org/repo bullseye main

- J'installe la clé: en root sur le serveur

apt-get -y install gnupg

wget -O - https://scodoc.org/misc/scodoc-repo.gpg.key | apt-key add -

- J'installe le logiciel:

apt-get update

apt-get install nginx

apt-get install scodoc9

- J'attribue un mot de passe à l'utilisateur scodoc:

passwd scodoc

- Je lance le script suivant en tant que root sur votre serveur nouvellement installé:

/opt/scodoc/tools/configure-scodoc9.sh

- Ensuite je lance scodoc

systemctl restart nginx

systemctl restart scodoc9

- L'administration se fait dans un terminal connecté au serveur (en général via ssh), en tant qu'utilisateur scodoc

su scodoc

- Je crée le département RT

flask create-dept DEPT

- Je crée un utilisateur

flask user-create LOGIN ROLE DEPT

J'ai le département qui est créé, mais la création de l'utilisateur ne fonctionne pas.



Grâce à la résolution DNS le nom de domaine pour le serveur scodoc est scodoc.sae.jing.fr