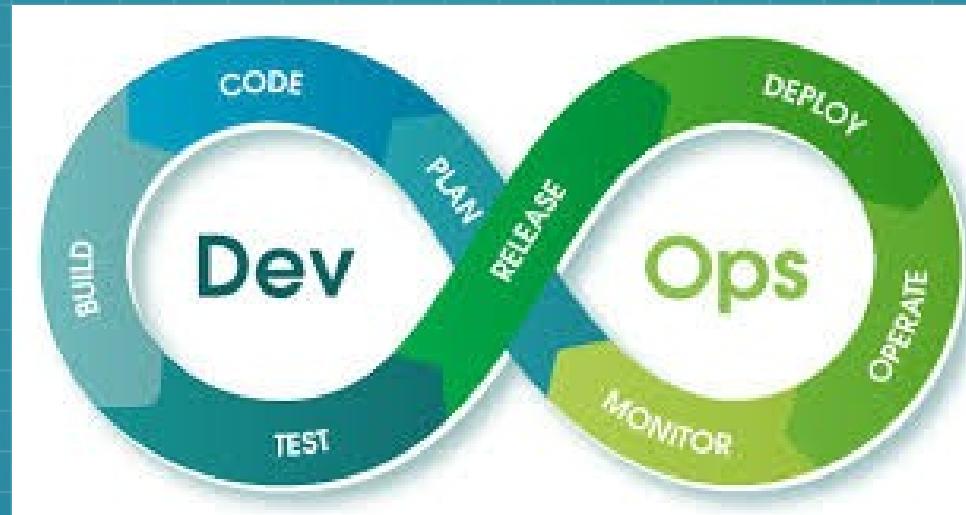
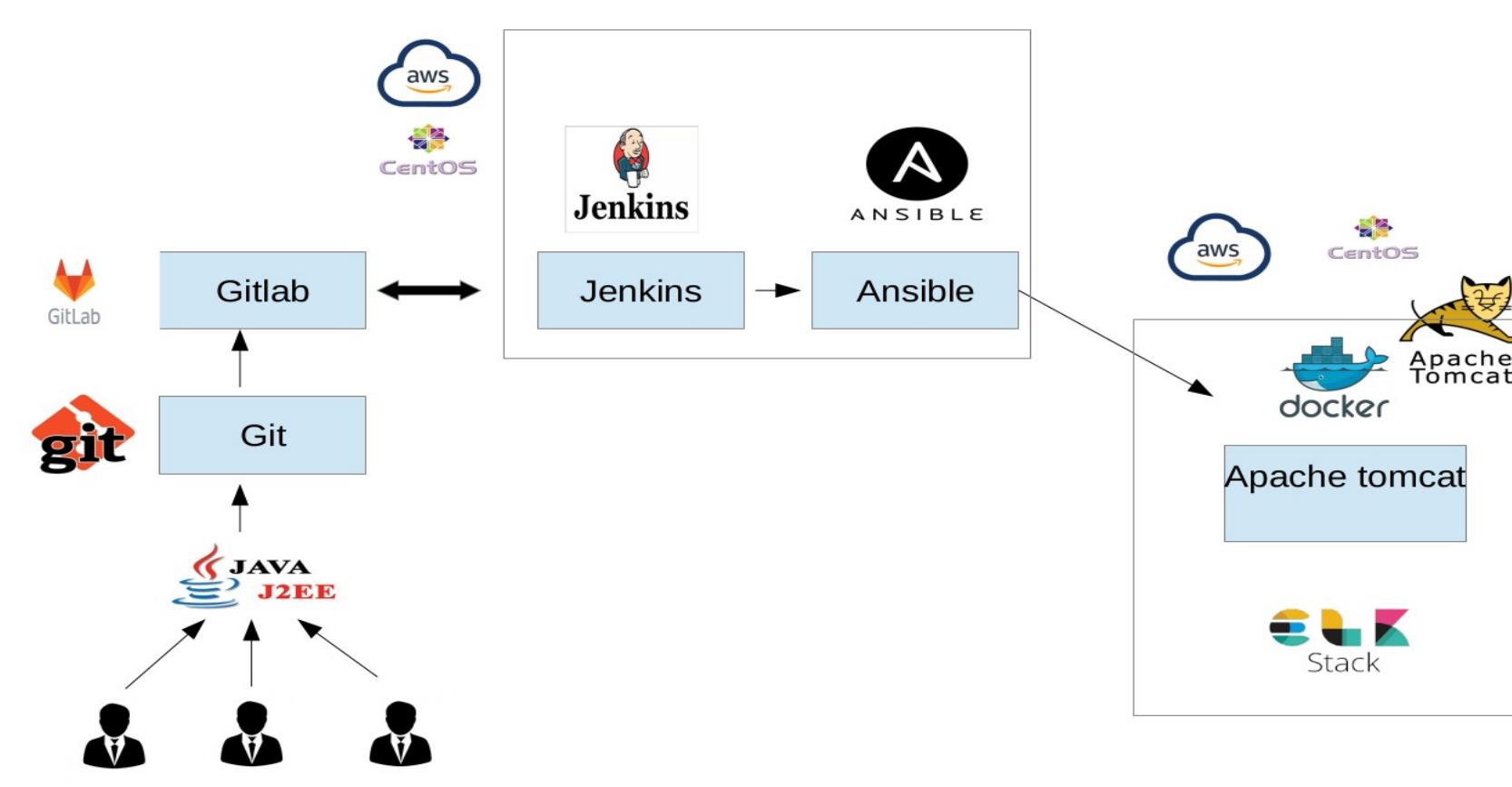


Formation Devops par la pratique



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Objectif : Réaliser une pipeline Git-Gitlab-Jenkins-Ansible-Docker



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Objectif : Réaliser une pipeline Git-Gitlab-Jenkins-Ansible-Docker

Pour atteindre l'objectif, on va décomposer le pipeline en plusieurs projets pour bien comprendre comment ça passe.

Projets :

- 1)- Déploiement Manuelle
- 2)- Git-Gitlab-Jenkins
- 3)- Git-Gitlab-Jenkins-Tomcat
- 4)- Git-Gitlab-Jenkins-Ansible-Tomcat
- 5)- Git-Gitlab-Jenkins-Ansible-Docker
- 6)- ELK Stack
- 7)- kubernetes
- 8)- Prometheus-Graphana



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Parcours de Formation DEVOPS

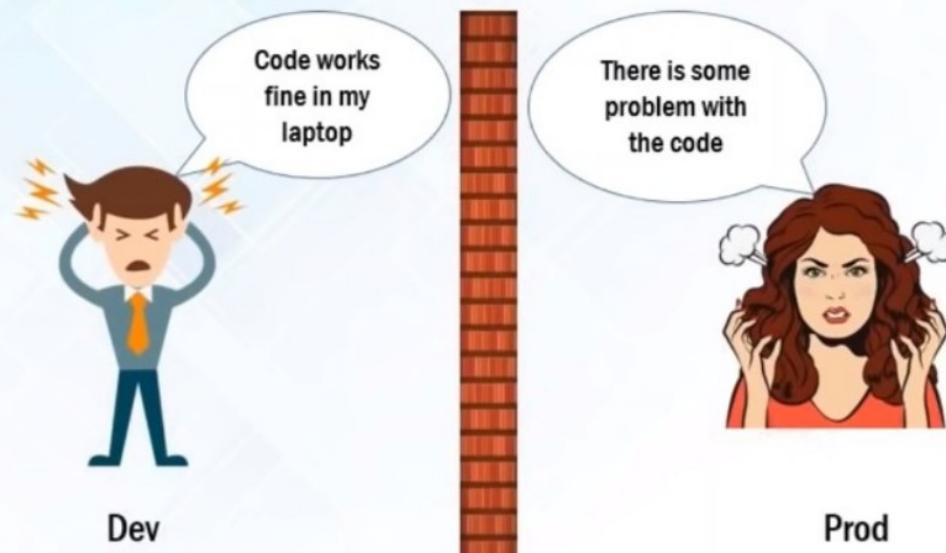
- Révision commande Linux
- Git, Gitlab
- Jenkins
- Ansible
- Docker
- ELK stack
- Kubernetes
- Prometheus-Graphana



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

An application works in developer's laptop but not in testing or production. This is due to difference in computing environment between Dev, Test and Prod.



In Dev there can be a software that is upgraded and in Prod the old version of software might be present



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

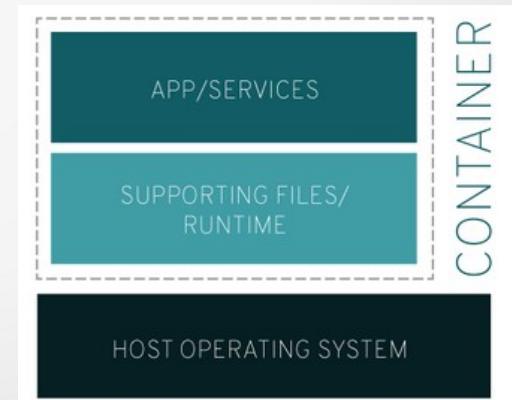
Docker

1- Un conteneur Linux, qu'est-ce que : Est un processus ou un ensemble de processus isolés du reste du système. Tous les fichiers nécessaires à leur exécution sont fournis par une image distincte, ce qui signifie que les conteneurs Linux sont portables et fonctionnent de la même manière dans les environnements de développement, de test et de production. Ainsi, ils sont bien plus rapides que les pipelines de développement qui s'appuient sur la réPLICATION d'environnements de test traditionnels. En raison de leur popularité et de leur facilité d'utilisation.

2- Pourquoi utiliser des conteneurs ?: Imaginons que vous développez une application. Vous travaillez sur un ordinateur portable dont l'environnement présente une configuration spécifique. D'autres développeurs peuvent travailler sur des machines qui présentent des configurations légèrement différentes. L'application que vous développez s'appuie sur cette configuration et utilise des bibliothèques, des dépendances et des fichiers spécifiques. En parallèle, votre entreprise exploite des environnements de développement et de production qui sont standardisés sur la base de configurations qui leur sont propres et de l'ensemble des fichiers associés. Vous souhaitez émuler ces environnements autant que possible localement, mais sans avoir à payer les coûts liés à la recréation des environnements de serveur. Comment faire pour que votre application en développement puisse fonctionner dans ces environnements, passer l'assurance qualité et être déployée sans prise de tête, sans réécriture et sans correctifs ? La réponse est simple : il vous suffit d'utiliser des conteneurs.

Le conteneur qui accueille votre application contient tous les fichiers, bibliothèques et dépendances nécessaires. Vous pouvez ainsi le déplacer jusqu'en production sans aucun effet secondaire.

Les éléments qui composent l'image d'un conteneur sont analogues à ceux de l'installation d'une distribution Linux, car cette image est fournie avec tous ses paquets RPM, fichiers de configuration, etc. Cependant, il est bien plus simple de distribuer des images de conteneurs que d'installer une nouvelle version d'un système d'exploitation. La crise est évitée, le travail peut continuer.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

3- Docker : Le terme « Docker » désigne plusieurs choses : le projet d'une communauté Open Source, les outils issus de ce projet Open Source, l'entreprise Docker Inc. qui constitue le principal soutien de ce projet, ainsi que les outils que l'entreprise prend officiellement en charge. Des technologies et une entreprise qui partagent le même nom, cela peut prêter à confusion.

Voici donc une rapide explication de cet état de fait.

Le logiciel « Docker » est une technologie de conteneurisation qui permet la création et l'utilisation de conteneurs Linux®.

La communauté Open Source Docker travaille à l'amélioration de cette technologie disponible gratuitement pour tout le monde.

L'entreprise Docker Inc. s'appuie sur le travail de la communauté Docker, sécurise sa technologie et partage ses avancées avec tous les utilisateurs. Elle prend ensuite en charge les technologies améliorées et sécurisées pour ses clients professionnels.

Avec la technologie Docker, vous pouvez traiter les conteneurs comme des machines virtuelles très légères et modulaires. En outre, ces conteneurs vous offrent une grande flexibilité : vous pouvez les créer, déployer, copier et déplacer d'un environnement à un autre, ce qui vous permet d'optimiser vos applications pour le cloud.

4- Comment fonctionne la technologie Docker ?: La technologie Docker utilise le noyau Linux et des fonctions de ce noyau, telles que les groupes de contrôle cgroups et les espaces de noms, pour séparer les processus afin qu'ils puissent s'exécuter de façon indépendante. Cette indépendance reflète l'objectif des conteneurs : exécuter plusieurs processus et applications séparément les uns des autres afin d'optimiser l'utilisation de votre infrastructure tout en bénéficiant du même niveau de sécurité que celui des systèmes distincts.

Les outils de conteneurs, y compris Docker, sont associés à un modèle de déploiement basé sur une image. Il est ainsi plus simple de partager une application ou un ensemble de services, avec toutes leurs dépendances, entre plusieurs environnements. Docker permet aussi d'automatiser le déploiement des applications (ou d'ensembles de processus combinés qui forment une application) au sein d'un environnement de conteneurs.

Ces outils conçus sur des conteneurs Linux (d'où leur convivialité et leur singularité) offrent aux utilisateurs un accès sans précédent aux applications, la capacité d'accélérer le déploiement, ainsi qu'un contrôle des versions et de l'attribution des versions.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Docker VS VM: <https://geekflare.com/fr/docker-vs-virtual-machine/>

Site Officiel : <https://www.docker.com/>

Docker-ce VS Docker-ee : <https://boxboat.com/2018/12/07/docker-ce-vs-docker-ee/>



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

5- Installation de docker sur serveur centos

Reference URL : <https://docs.docker.com/install/linux/docker-ce/centos/#install-using-the-repository>

Veuillez suivre les étapes ci-dessous pour installer docker CE sur une instance de serveur CentOS. Pour RedHat uniquement Docker EE disponible

1- Installez les packages requis.

```
$ sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

2- Utilisez la commande suivante pour configurer le référentiel stable.

```
$ sudo yum-config-manager --add-repo https://download.docker.com/linux/centos/docker-ce.repo
```

3. Installez la dernière version de Docker CE.

```
$ sudo yum install docker-ce
```

4. Démarrer et automatiser Docker : Le service Docker doit être configuré pour fonctionner au démarrage. Pour ce faire, saisissez chaque commande suivie de enter:

```
$ sudo systemctl start docker  
$ sudo systemctl enable docker
```

5- ajout de l'utilisateur actuel dans le groupe Docker :

```
$ sudo usermod -aG docker centos
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

6- Installation de docker sur serveur ubuntu & linux mint

```
$ sudo apt install docker.io
```

Démarrer et automatiser Docker : Le service Docker doit être configuré pour fonctionner au démarrage. Pour ce faire, saisissez chaque commande suivie de enter:

```
$ sudo systemctl start docker  
$ sudo systemctl enable docker
```

ajout de l'utilisateur actuel dans le groupe Docker :

```
$ sudo usermod -aG docker ubuntu
```

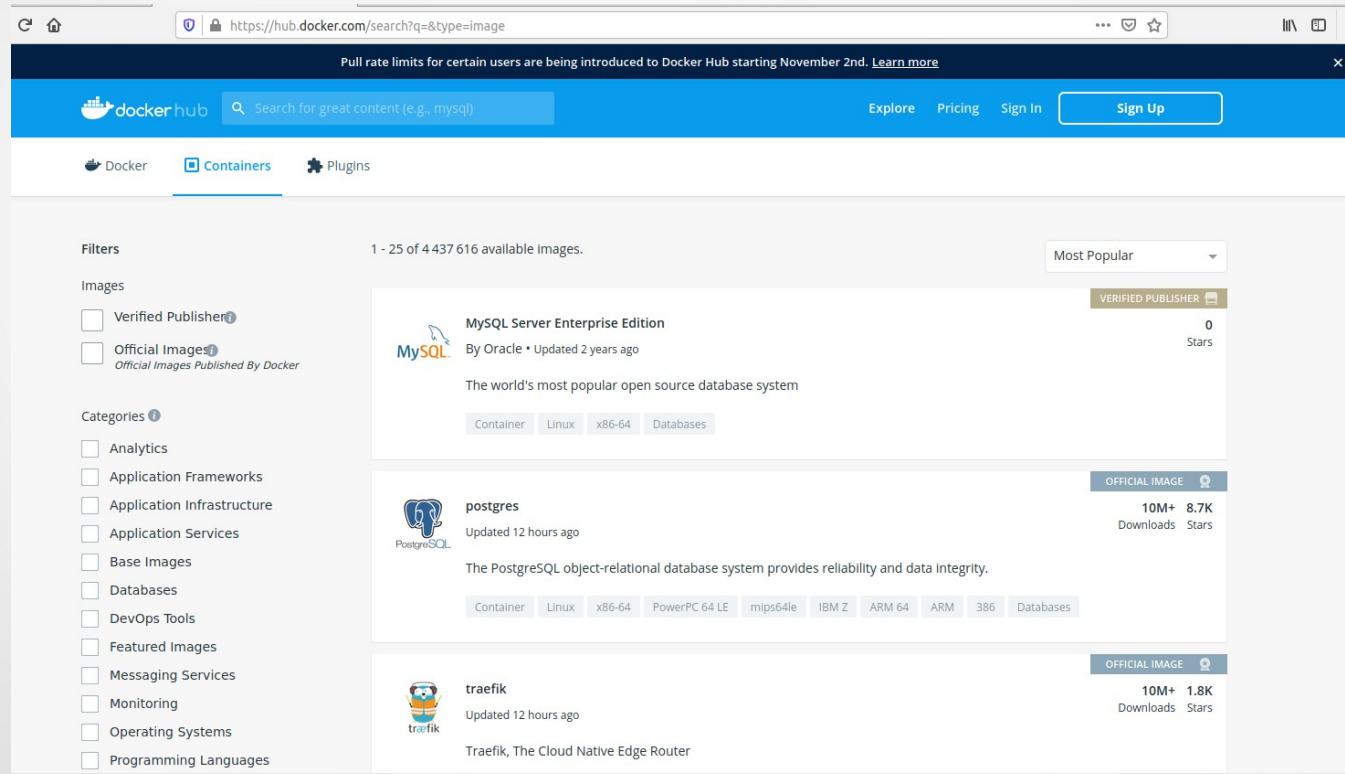


Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

7- Le docker Hub : Le docker Hub est un store où les utilisateurs de docker peuvent partager leurs images. Les images de base ont été créées par l'équipe de docker. Il est accessible ici : <https://hub.docker.com/explore/>

Aucun compte n'est nécessaire pour télécharger une image, mais bien évidemment pour pouvoir envoyer vos images, il faut un compte.



The screenshot shows the Docker Hub search results for "MySQL". There are three main card components displayed:

- MySQL Server Enterprise Edition** by Oracle (Updated 2 years ago): Verified Publisher, 0 stars. Description: "The world's most popular open source database system". Tags: Container, Linux, x86-64, Databases.
- postgres** (Official Image) by PostgreSQL (Updated 12 hours ago): 10M+ downloads, 8.7K stars. Description: "The PostgreSQL object-relational database system provides reliability and data integrity.". Tags: Container, Linux, x86-64, PowerPC 64 LE, mips64le, IBM Z, ARM 64, ARM, 386, Databases.
- traefik** (Official Image) by traefik (Updated 12 hours ago): 10M+ downloads, 1.8K stars. Description: "Traefik, The Cloud Native Edge Router". Tags: Container, Linux, x86-64.

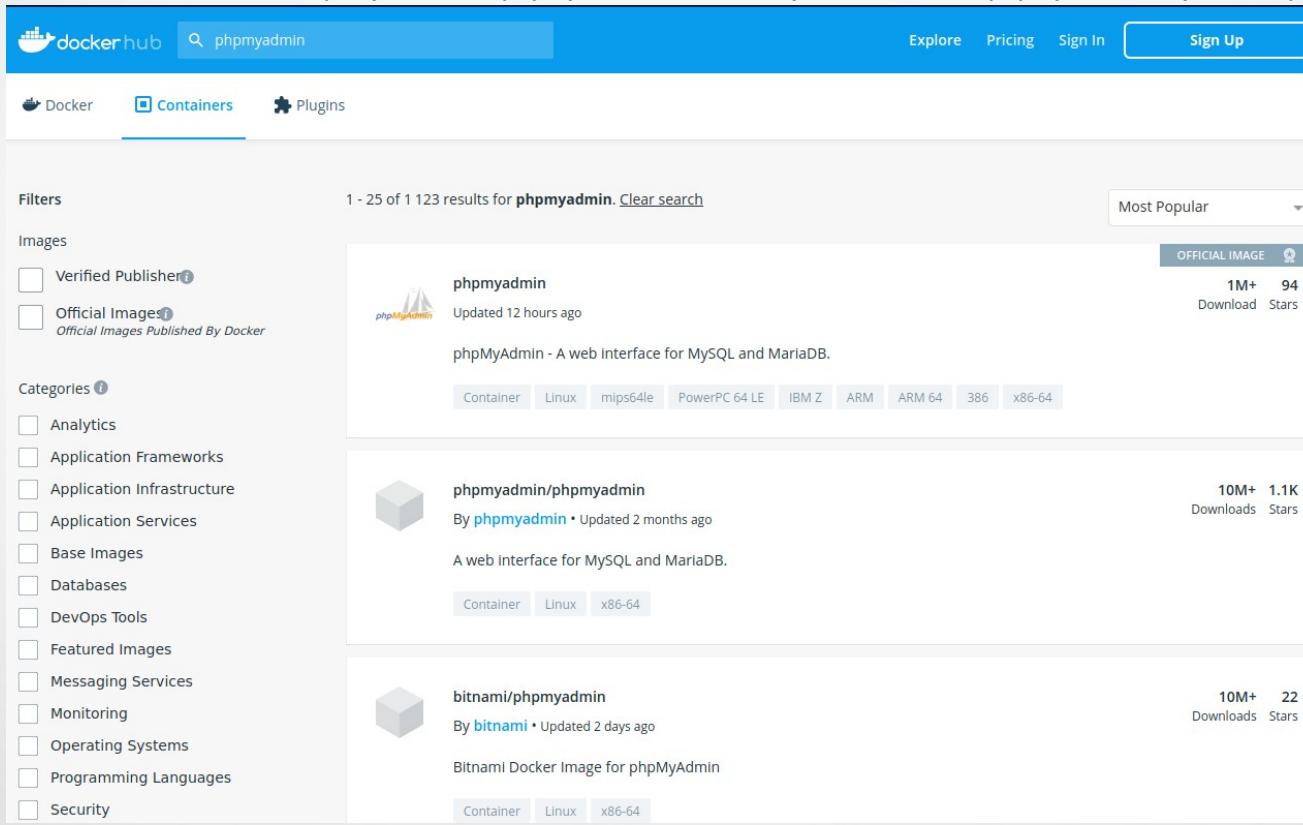


Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

8- Chercher une image : Il existe deux méthodes pour chercher sur le Hub.

- la première par le site web. Gardons mon exemple, je veux un phpmyadmin. Sur le site je cherche donc phpmyadmin, et j'obtiens plusieurs résultats :



The screenshot shows the Docker Hub search interface with the query "phpmyadmin" entered in the search bar. The results are filtered by "Containers". There are three main search results displayed:

- phpmyadmin** (Official Image): Updated 12 hours ago. Description: "phpMyAdmin - A web interface for MySQL and MariaDB." Tags: Container, Linux, mips64le, PowerPC 64 LE, IBM Z, ARM, ARM 64, 386, x86-64. Downloads: 1M+, Stars: 94.
- phpmyadmin/phpmyadmin**: By phpmyadmin • Updated 2 months ago. Description: "A web interface for MySQL and MariaDB." Tags: Container, Linux, x86-64. Downloads: 10M+, Stars: 1.1K.
- bitnami/phpmyadmin**: By bitnami • Updated 2 days ago. Description: "Bitnami Docker Image for phpMyAdmin" Tags: Container, Linux, x86-64. Downloads: 10M+, Stars: 22.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Nous avons donc plusieurs résultats, avec plusieurs informations :

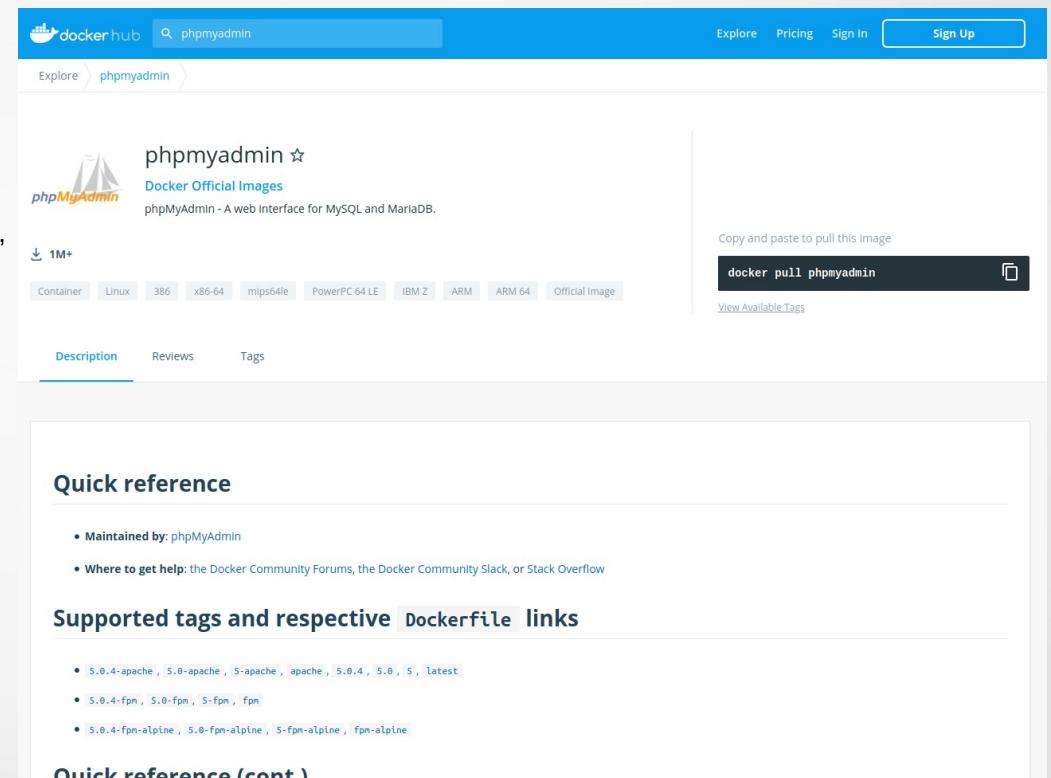
Le nom de l'image => Généralement sous la forme USER/IMAGE_NAME, sauf dans le cas d'une image officielle, où c'est seulement IMAGE_NAME

Le nombre de stars => Le système de notation

Le nombre de pulls => Le nombre de téléchargements

On en choisit l'officielle. Nous allons avoir plusieurs informations :
 description brève de l'image, la commande qui permet de la télécharger,
 plusieurs informations sur l'image, les versions des applications par exemple,
 les commandes/variables pour lancer un conteneur avec cette image...

Les Tags, ce sont les numéros de tags disponibles, souvent apparentés
 au numéro de version de l'application.



The screenshot shows the Docker Hub page for the `phpmyadmin` image. At the top, there's a search bar with `phpmyadmin` and navigation links for `Explore`, `Pricing`, `Sign In`, and `Sign Up`. Below the search bar, the page title is `phpmyadmin` with a star icon. It says "Docker Official Images" and describes it as "phpMyAdmin - A web Interface for MySQL and MariaDB". It shows 1M+ pulls. Below this, there are tabs for `Description`, `Reviews`, and `Tags`. The `Description` tab is active. On the right side, there's a "Copy and paste to pull this image" section with the command `docker pull phpmyadmin` and a "View Available Tags" link. Below this, there's a "Quick reference" section with links to maintainers and help forums, and a "Supported tags and respective Dockerfile links" section listing tags like `5.0.4-apache`, `5.0-apache`, `5-apache`, `apache`, `5.0.4`, `5.0`, `5`, and `latest`.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel,
 il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

- la deuxième méthodes pour trouver une image, en ligne de commande avec docker

```
bilel@bilel-G3-3590:~$ docker search phpmyadmin
NAME                           DESCRIPTION                                     STARS   OFFICIAL   AUTOMATED
phpmyadmin/phpmyadmin          A web interface for MySQL and MariaDB.      1097    [OK]
phpmyadmin                      phpMyAdmin - A web interface for MySQL and M...   94      [OK]
nazarpc/phpmyadmin             phpMyAdmin as Docker container, based on off...  61      [OK]
bitnami/phpmyadmin              Bitnami Docker Image for phpMyAdmin           22      [OK]
jackgruber/phpmyadmin          Raspberry Pi / ARM compatible Docker Image f...   7       [OK]
maxexcloo/phpmyadmin           Application container with phpMyAdmin instal...  5       [OK]
sk278/phpmyadmin-armhf        phpMyAdmin for Arm (like RPi)                   3       [OK]
silintl/phpmyadmin              PHPMyAdmin container for ddev                  1       [OK]
drud/phpmyadmin                phpMyAdmin - A web interface for MySQL and M...   1       [OK]
arm64v8/phpmyadmin              The official phpmyadmin image, but built on ...  1       [OK]
biarms/phpmyadmin               PhpMyAdmin image supporting ALL the PMA vers...  1       [OK]
dnhsoft/phpmyadmin              phpMyAdmin on Alpine                         1       [OK]
splattael/phpmyadmin            phpMyAdmin for OpenShift                     1       [OK]
vshn/phpmyadmin                phpMyAdmin is a free software tool to handle...  0       [OK]
tianon/phpmyadmin               phpMyAdmin                               0       [OK]
computersciencehouse/phpmyadmin phpMyAdmin                               0       [OK]
morozovgroup/phpmyadmin         Openshift Compatible phpmyadmin             0       [OK]
rickw/phpmyadmin                phpMyAdmin - based on PHP 7.0 with Apache       0       [OK]
hauptmedia/phpmyadmin           An ARM image of phpmyadmin using the sourcec...  0       [OK]
pwfraley/phpmyadmin             Pre-configured phpMyAdmin (new theme, settin...  0       [OK]
rkcreation/phpmyadmin           phpMyAdmin Docker image.                      0       [OK]
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

9- Gérer les images : Dans cette partie, nous allons voir comment gérer nos images, c'est-à-dire les télécharger, les lister, et bien sûr les supprimer.

Nous utiliserons ici **docker image [subcommand]** :

```
bilel@bilel-G3-3590:~$ docker image --help

Usage: docker image COMMAND

Manage images

Commands:
build      Build an image from a Dockerfile
history    Show the history of an image
import     Import the contents from a tarball to create a filesystem image
inspect    Display detailed information on one or more images
load       Load an image from a tar archive or STDIN
ls         List images
prune     Remove unused images
pull       Pull an image or a repository from a registry
push       Push an image or a repository to a registry
rm        Remove one or more images
save      Save one or more images to a tar archive (streamed to STDOUT by default)
tag       Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE

Run 'docker image COMMAND --help' for more information on a command.
bilel@bilel-G3-3590:~$ █
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Télécharger des images : Pour télécharger une image, on utilise cette commande

```
$ docker image pull [nom image]:[tag]
```

Ce qui donne pour télécharger notre httpd

```
bilel@bilel-G3-3590:~$ docker pull httpd
Using default tag: latest
latest: Pulling from library/httpd
6ec7b7d162b2: Pull complete
17e233bac21e: Pull complete
130aad5bf43a: Pull complete
81d0a34533d4: Pull complete
da240d12a8a4: Pull complete
Digest: sha256:a3a2886ec250194804974932eaf4a4ba2b77c4e7d551ddb63b01068bf70f4120
Status: Downloaded newer image for httpd:latest
docker.io/library/httpd:latest
bilel@bilel-G3-3590:~$ █
```

Si on ne met pas de tag, il télécharge automatiquement la version latest. Comme nous avons vu dans la partie sur le docker hub, l'image possède plusieurs tags.

Docker pull httpd == docker pull httpd:latest



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Nous pouvons voir qu'il avait déjà des éléments, en fait une image est souvent basée sur une autre image, qui peut être basée sur une autre et ainsi de suite. Ce sont des layers (couches). Vous comprendrez mieux ceci lorsque nous apprendrons à créer des images. Chaque couche possède un id unique, c'est ce qui permet de savoir s'il est déjà présent ou non. Sur certaines images, comme les officielles, plusieurs tags peuvent être associés à une même image pour une même version.

Par exemple on peut voir sur le hub, que latest correspond également à 2.4.46, 2.4, 2

Donc si maintenant je télécharge la version 2.4.46, puisqu'il a déjà toutes les couches, il ne devrait pas les retélécharger :

```
bilel@bilel-G3-3590:~$ docker pull httpd:2.4.46
2.4.46: Pulling from library/httpd
Digest: sha256:a3a2886ec250194804974932eaf4a4ba2b77c4e7d551ddb63b01068bf70f4120
Status: Downloaded newer image for httpd:2.4.46
docker.io/library/httpd:2.4.46
```

je télécharge la version 2 puisqu'il a déjà toutes les couches, il ne devrait pas les retélécharger :

```
bilel@bilel-G3-3590:~$ docker pull httpd:2
2: Pulling from library/httpd
Digest: sha256:a3a2886ec250194804974932eaf4a4ba2b77c4e7d551ddb63b01068bf70f4120
Status: Downloaded newer image for httpd:2
docker.io/library/httpd:2
```

Donc effectivement, tout était déjà présent, donc il n'a rien téléchargé.

TAG	DIGEST	OS/ARCH	COMPRESSED SIZE
latest	47:9f7bd8986	linux/386	52.46 MB
	a938cd7d56e7	linux/amd64	51.85 MB
	b64951473d38	linux/arm/v5	48.58 MB
	582f0c3ae3ea	linux/arm/v7	45.98 MB
	ac0ebdf20650	linux/arm64/v8	50.45 MB
	d1d934d85f34	linux/mips64le	50.19 MB
	033cac019a63	linux/ppc64le	56.44 MB
	a9725ec85f64	linux/s390x	50.12 MB

TAG	DIGEST	OS/ARCH	COMPRESSED SIZE
2.4.46	47:9f7bd8986	linux/386	52.46 MB
	a938cd7d56e7	linux/amd64	51.85 MB
	b64951473d38	linux/arm/v5	48.58 MB
	582f0c3ae3ea	linux/arm/v7	45.98 MB
	ac0ebdf20650	linux/arm64/v8	50.45 MB
	d1d934d85f34	linux/mips64le	50.19 MB
	033cac019a63	linux/ppc64le	56.44 MB
	a9725ec85f64	linux/s390x	50.12 MB

TAG	DIGEST	OS/ARCH	COMPRESSED SIZE
2.4	47:9f7bd8986	linux/386	52.46 MB
	a938cd7d56e7	linux/amd64	51.85 MB
	b64951473d38	linux/arm/v5	48.58 MB
	582f0c3ae3ea	linux/arm/v7	45.98 MB
	ac0ebdf20650	linux/arm64/v8	50.45 MB
	d1d934d85f34	linux/mips64le	50.19 MB
	033cac019a63	linux/ppc64le	56.44 MB
	a9725ec85f64	linux/s390x	50.12 MB

TAG	DIGEST	OS/ARCH	COMPRESSED SIZE
2	47:9f7bd8986	linux/386	52.46 MB
	a938cd7d56e7	linux/amd64	51.85 MB
	b64951473d38	linux/arm/v5	48.58 MB
	582f0c3ae3ea	linux/arm/v7	45.98 MB
	ac0ebdf20650	linux/arm64/v8	50.45 MB
	d1d934d85f34	linux/mips64le	50.19 MB
	033cac019a63	linux/ppc64le	56.44 MB
	a9725ec85f64	linux/s390x	50.12 MB



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Lister les images

Pour lister les images téléchargées, donc disponibles en local, nous utiliserons cette commande : **\$ docker image ls** ou **\$ docker images**

Ce qui donne avec ce que l'on a téléchargé :

```
bilel@bilel-G3-3590:~$ docker image ls
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
httpd               2        dd85cddb9987   11 days ago  138MB
httpd               2.4.46   dd85cddb9987   11 days ago  138MB
httpd               latest    dd85cddb9987   11 days ago  138MB
bilel@bilel-G3-3590:~$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE
httpd               2        dd85cddb9987   11 days ago  138MB
httpd               2.4.46   dd85cddb9987   11 days ago  138MB
httpd               latest    dd85cddb9987   11 days ago  138MB
bilel@bilel-G3-3590:~$ █
```

REPOSITORY : Le nom de l'image

TAG : Version de l'image

IMAGE ID : Identifiant unique de l'image

CREATED : Date de création de l'image

VIRTUAL SIZE : Taille de l'image



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Supprimer les images

Pour supprimer une image, c'est plutôt simple : **\$ docker image rm [ID image ou nom Image :tag]** ou **\$ docker rmi [ID image ou nom Image:tag]**

Voici un exemple :

```
bilel@bilel-G3-3590:~$ docker image rm httpd:2
Untagged: httpd:2
Untagged: httpd@sha256:a3a2886ec250194804974932eaf4a4ba2b77c4e7d551ddb63b01068bf70f4120
Deleted: sha256:dd85cdbb99877b73f0de2053f225af590ab188d79469eebdb23ec2d26d0d10e8
Deleted: sha256:bc2085990715d2d1ac2179131969b293821dbc7f0538eaeb3bc3bb6d5645a13
Deleted: sha256:a11bf2367d443697ca6267194a18b31289b59b677133e6482779851fa33992df
Deleted: sha256:388a801aa9f7953143fc154e7bc16dd5b696d2f3578a054b0b804557a8d89d74
Deleted: sha256:37495edb13f558e001fba79f653b38ac3cad5e29f189c1e20501f3f6d5326fdc
Deleted: sha256:87c8a1d8f54f3aa4e05569e8919397b65056aa71cdf48b7f061432c98475eee9
bilel@bilel-G3-3590:~$ 
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	alpine	5d779ff71c18	5 days ago	55.5MB
httpd	2	dd85cdbb9987	11 days ago	138MB

```
bilel@bilel-G3-3590:~$ docker rmi dd85cdbb9987
Untagged: httpd:2
Untagged: httpd@sha256:a3a2886ec250194804974932eaf4a4ba2b77c4e7d551ddb63b01068bf70f4120
Deleted: sha256:dd85cdbb99877b73f0de2053f225af590ab188d79469eebdb23ec2d26d0d10e8
Deleted: sha256:bc2085990715d2d1ac2179131969b293821dbc7f0538eaeb3bc3bb6d5645a13
Deleted: sha256:a11bf2367d443697ca6267194a18b31289b59b677133e6482779851fa33992df
Deleted: sha256:388a801aa9f7953143fc154e7bc16dd5b696d2f3578a054b0b804557a8d89d74
Deleted: sha256:37495edb13f558e001fba79f653b38ac3cad5e29f189c1e20501f3f6d5326fdc
Deleted: sha256:87c8a1d8f54f3aa4e05569e8919397b65056aa71cdf48b7f061432c98475eee9
bilel@bilel-G3-3590:~$ 
```

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
httpd	alpine	5d779ff71c18	5 days ago	55.5MB



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Nous pouvons bien sûr supprimer plusieurs images : **\$ docker image rm [image] [image] [image]** ou **\$ docker rmi [image] [image] [image]**

Pour supprimer toutes les images inutilisées : **\$ docker image prune -a**

Pour supprimer toutes les images docker on peut utiliser la commande : **\$ docker rmi \$(docker image ls -q)**

Note : On ne peut pas supprimer une image utilisée par un conteneur en cours d'exécution, dans ce cas on arrête le conteneur puis supprimer l'image avec l'option **-f** pour forcer la suppression

```
bilel@bilel-G3-3590:~$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED       SIZE
httpd          alpine        5d779ff71c18    5 days ago   55.5MB
bilel@bilel-G3-3590:~$ docker rmi 5d779ff71c18
Error response from daemon: conflict: unable to delete 5d779ff71c18 (cannot be forced) - image is being used by running container 0cad41ce2f28
bilel@bilel-G3-3590:~$ docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS      PORTS     NAMES
0cad41ce2f28   httpd:alpine   "httpd-foreground" 24 minutes ago Up 24 minutes  80/tcp    hello_devops
bilel@bilel-G3-3590:~$ docker stop 0cad41ce2f28
0cad41ce2f28
bilel@bilel-G3-3590:~$ docker rmi 5d779ff71c18
Error response from daemon: conflict: unable to delete 5d779ff71c18 (must be forced) - image is being used by stopped container 0cad41ce2f28
bilel@bilel-G3-3590:~$ docker rmi -f 5d779ff71c18
Untagged: httpd:alpine
Untagged: httpd@sha256:17e9caf91cbe2388a685d74c3ee2084d3bfbd144f839a5f2245b7f8ef350564
Deleted: sha256:5d779ff71c188aa6da896ffdab929ca0cb1d859dc74650e57b9d8ce2bb6debff
bilel@bilel-G3-3590:~$ docker start 0cad41ce2f28
0cad41ce2f28
bilel@bilel-G3-3590:~$ docker ps
CONTAINER ID   IMAGE          COMMAND       CREATED      STATUS      PORTS     NAMES
0cad41ce2f28   5d779ff71c18   "httpd-foreground" 24 minutes ago Up 4 seconds  80/tcp    hello_devops
bilel@bilel-G3-3590:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Rappel sur les commandes utilisées

\$ docker pull : télécharger une image docker

\$ docker images : lister les images téléchargées

\$ docker rmi <ID_conteneur> : supprimer une image

\$ docker rmi -f <ID_conteneur> : forcé la suppression d'une image

\$ docker image prune -a : supprimer toutes les images inutilisées

\$ docker rmi \$(docker image ls -q) : supprimer toutes les images



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

10-Gérer les conteneurs : Dans cette partie, nous verrons comment créer un conteneur , mais aussi comment le supprimer, comment les gérer, les relancer et plein d'autres choses indispensables.

Lancer, arrêter et lister des conteneurs :

La première commande que nous utiliserons, sera **docker run** qui s'utilise comme ceci : **\$ docker run [OPTIONS] IMAGE [COMMAND]**

Nous allons commencer par un petit conteneur, basé sur Debian, et nous lui dirons d'afficher « Bonjour Ghazela Devops !!! » :

```
bilel@bilel-G3-3590:~$ docker run debian echo "Bonjour Ghazela Devops"
Unable to find image 'debian:latest' locally
latest: Pulling from library/debian
6c33745f49b4: Pull complete
Digest: sha256:22d4552b9f96fd0ea943cb846d58b069d4df297673636055a3d984b3ccac6a28
Status: Downloaded newer image for debian:latest
Bonjour Ghazela Devops
bilel@bilel-G3-3590:~$ █
```

il s'est passé quoi là ? Nous avons créé et exécuté notre conteneur, mais puisqu'il n'a pas trouvé l'image Debian en local, il l'a téléchargée de lui même (sans avoir à utiliser docker pull), Ensuite il a exécuté la commande qu'on lui a passée, à savoir écrire « bonjour ghazela devops ». Et c'est tout, puisque l'echo est terminé, il a éteint le conteneur.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Nous allons vérifier si ce conteneur est démarré ou pas, pour ce faire nous utiliserons la commande **docker ps** :

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
bilel@bilel-G3-3590:~\$						

Nous n'avons aucun conteneur en cours. Mais il doit bien être quelque part ce conteneur !! non ?!

Oui et nous pouvons bien évidemment le voir, il suffit d'ajouter l'option **-a**, qui permet de voir tous les conteneurs :

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
bilel@bilel-G3-3590:~\$						
d1ce5db2e255	debian	"echo 'Bonjour Ghaze...'"	10 minutes ago	Exited (0) 10 minutes ago		nifty_heyrovsky
bilel@bilel-G3-3590:~\$						

Le voici, petite explication de ce tableau.

CONTAINER ID : ID du conteneur, généré de manière à ce qu'il soit unique.

IMAGE : l'image utilisée pour ce conteneur.

COMMAND : la commande exécutée.

CREATED : temps depuis création du conteneur.

STATUS : le statut actuel du conteneur, ici exited avec un code retour 0 (sans erreur) depuis 10 minutes.

PORTS : liste des ports écoutés (nous verrons ceci plus tard).

NAMES : nom du conteneur, ici c'est un nom aléatoire, car nous n'en avons pas défini à notre conteneur.

NB :

- Pour lister les conteneur en cours d'exécution, on utilise la commande **\$ docker ps**

- Pour lister tous les conteneurs, on utilise la commande **\$ docker ps -a**



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Exemple 2 : on va créer un conteneur nommé Hello_Devops à base d'une image docker httpd

```
docker run --name <container_Name> <image_name>:<image_version/tag>
$ docker run --name hello_devops httpd:alpine
```

```
bilel@bilel-G3-3590:~$ docker run --name hello_devops httpd:alpine
Unable to find image 'httpd:alpine' locally
alpine: Pulling from library/httpd
801bfaa63ef2: Pull complete
ac8f86b44b17: Pull complete
078b6c86de97: Pull complete
55f318a9c48a: Pull complete
5da5afdb6ea0: Pull complete
Digest: sha256:17e9cafb91cbe2388a685d74c3ee2084d3bfbd144f839a5f2245b7f8ef350564
Status: Downloaded newer image for httpd:alpine
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
[Tue Dec 22 18:29:01.846316 2020] [mpm_event:notice] [pid 1:tid 139884185820488] AH00489: Apache/2.4.46 (Unix) configured -- resuming normal operations
[Tue Dec 22 18:29:01.846343 2020] [core:notice] [pid 1:tid 139884185820488] AH00094: Command line: 'httpd -D FOREGROUND'
```

Note: comme vous voyez on entre directement dans le conteneur sans avoir la possibilité de taper n'importe quelle commande



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

j'ai fait ctrl+c pour sortir, mais ici le conteneur va être arrêté « exited »

```
bilel@bilel-G3-3590:~$ docker run --name hello_devops httpd:alpine
Unable to find image 'httpd:alpine' locally
alpine: Pulling from library/httpd
801bfaa63ef2: Pull complete
ac8f86b44b17: Pull complete
078b6c86de97: Pull complete
55f318a9c48a: Pull complete
5da5afdb6ea0: Pull complete
Digest: sha256:17e9caf91cbe2388a685d74c3ee2084d3bfbd144f839a5f2245b7f8ef350564
Status: Downloaded newer image for httpd:alpine
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName'
ppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName'
ppress this message
[Tue Dec 22 18:29:01.846316 2020] [mpm_event:notice] [pid 1:tid 139884185820488] AH00489: Apache/2.4.46 (Unix) configured --
s
[Tue Dec 22 18:29:01.846343 2020] [core:notice] [pid 1:tid 139884185820488] AH00094: Command line: 'httpd -D FOREGROUND'
^C[Tue Dec 22 18:30:37.654300 2020] [mpm_event:notice] [pid 1:tid 139884185820488] AH00491: caught SIGTERM, shutting down
bilel@bilel-G3-3590:~$ 
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
3437cb41a5d0	httpd:alpine	"httpd-foreground"	4 minutes ago	Exited (0) 3 minutes ago		hello_devops

Pour démarrer le conteneur de nouveau on utilise la commande **\$ docker start <id_container>**

Pour arrêter un conteneur, on utilise la commande **\$ docker stop <id_container>**



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

```
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
3437cb41a5d0        httpd:alpine       "httpd-foreground"   4 minutes ago     Exited (0) 3 minutes ago
bilel@bilel-G3-3590:~$ docker start 3437cb41a5d0
3437cb41a5d0
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
3437cb41a5d0        httpd:alpine       "httpd-foreground"   6 minutes ago     Up 5 seconds      80/tcp
bilel@bilel-G3-3590:~$ docker stop 3437cb41a5d0
3437cb41a5d0
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
3437cb41a5d0        httpd:alpine       "httpd-foreground"   7 minutes ago     Exited (0) 2 seconds ago
bilel@bilel-G3-3590:~$ 
```

Pour exécuter un conteneur en arrière-plan, on utilise l'argument **-d** lors de la création avec **-i** et **-t**

-t : (Terminal) fournit un terminal au docker ;

-i : (interactive) permet d'écrire dans le conteneur (couplé à **-t**) ;

-d : (detached mode : background mode) exécute le conteneur en arrière-plan ;

```
bilel@bilel-G3-3590:~$ docker run -itd --name hello_devops httpd:alpine
0cad41ce2f2803715323449357db2d92ce5ae365f2bee0722124671094d61ba0
bilel@bilel-G3-3590:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
0cad41ce2f28        httpd:alpine       "httpd-foreground"   7 seconds ago     Up 5 seconds      80/tcp
bilel@bilel-G3-3590:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Pour redémarrer un conteneur on utilise la commande **\$ docker restart <ID_container>**

```
bilel@bilel-G3-3590:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
0cad41ce2f28        5d779ff71c18    "httpd-foreground"   About an hour ago   Up 51 minutes      80/tcp
bilel@bilel-G3-3590:~$ docker restart 0cad41ce2f28
0cad41ce2f28
bilel@bilel-G3-3590:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
0cad41ce2f28        5d779ff71c18    "httpd-foreground"   About an hour ago   Up 2 seconds       80/tcp
bilel@bilel-G3-3590:~$ 
```

Supprimer les conteneurs

Maintenant que nous avons vu comment créer, lister, démarrer, redémarrer et arrêter un conteneur, il ne nous reste plus qu'à... les supprimer. Pour cela, nous allons utiliser la commande : **\$ docker rm <ID_container>** ==> cette commande ne peut supprimer que les conteneurs arrêtés,

```
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS
9ef31244bbb6        httpd:alpine     "httpd-foreground"   2 minutes ago     Exited (0) About a minute ago
0cad41ce2f28        httpd:alpine     "httpd-foreground"   2 hours ago      Up 38 minutes      80/tcp
bilel@bilel-G3-3590:~$ docker rm 9ef31244bbb6
9ef31244bbb6
bilel@bilel-G3-3590:~$ docker rm 0cad41ce2f28
Error response from daemon: You cannot remove a running container 0cad41ce2f2803715323449357db2d92ce5ae365f2bee0722124671094d61ba0. Stop the container before attempting removal or force remove
bilel@bilel-G3-3590:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Vous pouvez également supprimer un conteneur qui tourne, avec l'option -f.

```
bilel@bilel-G3-3590:~$ docker rm 0cad41ce2f28
Error response from daemon: You cannot remove a running container 0cad41ce2f2803715323449357db2d92ce5ae365f2bee0722124671094d61ba0. Stop the container before attempting removal or force remove
bilel@bilel-G3-3590:~$ docker rm -f 0cad41ce2f28
0cad41ce2f28
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS           NAMES
bilel@bilel-G3-3590:~$
```

Pour supprimer tous les conteneurs arrêtés, on utilise la commande **\$ docker container prune**

```
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS           NAMES
9a95dd72fa59        debian              "echo 'Bonjour Ghaze..." 10 seconds ago   Exited (0) 8 seconds ago
c170c5cc6fe7        debian              "echo 'Bonjour Ghaze..." 14 seconds ago   Exited (0) 13 seconds ago
7c39988c9bf3        debian              "echo 'Bonjour Ghaze..." 19 seconds ago   Exited (0) 17 seconds ago
6e17cc700ac4        debian              "echo 'Bonjour Ghaze..." 22 seconds ago   Exited (0) 20 seconds ago
f735c6c9f481        debian              "echo 'Bonjour Ghaze..." 41 seconds ago   Exited (0) 38 seconds ago
a
bilel@bilel-G3-3590:~$ docker container prune
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
Deleted Containers:
9a95dd72fa59aed610719015cd1882bd43c3b814ba179a3ecbc7f75bad80791
c170c5cc6fe777b5bcedd25c6cace59fb24e94a127554ab10a9b96ac5197a12d
7c39988c9bf3b032f1c8737ccead9099ac5b59848410b4c3e2c47428f4f57e1c
6e17cc700ac44c814493704eff6f2323dbcfaa92c21cebfbd2996a02753afeb2
f735c6c9f481359425a5a3f1b720df8a7b9285b2664ed5d5a155fc868d6d3c3

Total reclaimed space: 0B
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS           NAMES
bilel@bilel-G3-3590:~$
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Ou même tous les conteneurs, via la commande `$ docker rm -f $(docker ps -a -q)`

```
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
d5293420b495        httpd:alpine       "httpd-foreground"   6 seconds ago      Up 5 seconds       80/tcp
0b8031a51daa        httpd:alpine       "httpd-foreground"   12 seconds ago     Up 10 seconds      80/tcp
01436da9b5a1        debian             "echo 'Bonjour Ghaze..." 20 seconds ago    Exited (0) 18 seconds ago
2ab92636eaf9        debian             "echo 'Bonjour Ghaze..." 22 seconds ago    Exited (0) 21 seconds ago
ra
bilel@bilel-G3-3590:~$ docker rm -f $(docker ps -a -q)
d5293420b495
0b8031a51daa
01436da9b5a1
2ab92636eaf9
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
bilel@bilel-G3-3590:~$ 
```

Voir les logs des conteneurs

Tout sysadmin/devs doit penser, et même rêver des logs, c'est indispensable. Avec docker c'est assez spécial, les logs d'un conteneur sont en fait ce qui est en output (stdin et stderr) du shell.

C'est plutôt simple, même très simple, nous utiliserons `$ docker logs <ID_container>`



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Exemple :

```
bilel@bilel-G3-3590:~$ docker run -itd --name devops httpd:alpine
aac2e2e63ff77f4faa834f08678390a1dd7b646072471968c01e366d5580e22d
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS           NAMES
aac2e2e63ff7        httpd:alpine       "httpd-foreground"   3 seconds ago     Up 2 seconds      80/tcp          devops
bilel@bilel-G3-3590:~$ docker logs aac2e2e63ff7
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using 172.17.0.2. Set the 'ServerName' directive globally to suppress this message
[Wed Dec 23 12:46:29.382401 2020] [mpm_event:notice] [pid 1:tid 140321870880072] AH00489: Apache/2.4.46 (Unix) configured -- resuming normal operations
[Wed Dec 23 12:46:29.382434 2020] [core:notice] [pid 1:tid 140321870880072] AH00094: Command line: 'httpd -D FOREGROUND'
bilel@bilel-G3-3590:~$
```

Il est possible de faire comme tail :

```
$ docker logs --tail=20 aac2e2e63ff7 # Affiche les 20 dernières lignes
$ docker logs -f aac2e2e63ff7 # Affiche les logs au fur et à mesure
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Docker inspect

La commande « inspect » listera les informations complètes du conteneur. Utilisez l'ID de conteneur répertorié dans la première colonne avec l'option inspect. Vous obtiendrez une sortie assez longue ici.

```
bilel@bilel-G3-3590:~$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS               NAMES
aac2e2e63ff7        httpd:alpine        "httpd-foreground"   About a minute ago   Up About a minute   80/tcp              devops

bilel@bilel-G3-3590:~$ docker inspect aac2e2e63ff7
[
  {
    "Id": "aac2e2e63ff77f4faa834f08678390a1dd7b646072471968c01e366d5580e22d",
    "Created": "2020-12-23T12:46:28.243079277Z",
    "Path": "httpd-foreground",
    "Args": [],
    "State": {
      "Status": "running",
      "Running": true,
      "Paused": false,
      "Restarting": false,
      "OOMKilled": false,
      "Dead": false,
      "Pid": 25317,
      "ExitCode": 0,
      "Error": "",
      "StartedAt": "2020-12-23T12:46:29.347204985Z",
      "FinishedAt": "0001-01-01T00:00:00Z"
    },
    "Image": "sha256:5d779ff71c188aa6da896ffdab929ca0cb1d859dc74650e57b9d8ce2bb6debff",
    "ResolvConfPath": "/var/lib/docker/containers/aac2e2e63ff77f4faa834f08678390a1dd7b646072471968c01e366d5580e22d/resolv.conf",
    "HostnamePath": "/var/lib/docker/containers/aac2e2e63ff77f4faa834f08678390a1dd7b646072471968c01e366d5580e22d/hostname",
    "HostsPath": "/var/lib/docker/containers/aac2e2e63ff77f4faa834f08678390a1dd7b646072471968c01e366d5580e22d/hosts",
    "LogPath": "/var/lib/docker/containers/aac2e2e63ff77f4faa834f08678390a1dd7b646072471968c01e366d5580e22d/aac2e2e63ff77f4faa834f08678390a1dd7b646072471968c01e366d5580e22d-json.log",
    "Name": "/devops",
  }
]
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Rappel sur les commandes utilisées

\$ docker run -itd –name <name> <image:tag> : crée et exécuter un conteneur

\$ docker ps : lister les conteneurs en cours d'exécution

\$ docker ps -a : lister tous les conteneurs

\$ docker stop <ID_container> : arrêter un conteneur

\$ docker start <ID_container> : démarrer un conteneur

\$ docker restart <ID_container> : redémarrer un conteneur

\$ docker rm <ID_container> : supprimer un conteneur arrêté

\$ docker container prune : supprimer tous les conteneurs arrêtés

\$ docker rm -f <ID_container> : forcé la suppression d'un conteneur

\$ docker rm -f \$(docker ps -a -q) : supprimer tous les conteneurs

\$ docker logs <ID_container> : afficher les log d'un conteneur

\$ docker inspect <ID_container> : afficher les informations complètes du conteneur



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Docker Exec

Pour exécuter des commandes sur des conteneurs en cours d'exécution, vous devez exécuter «docker exec» et spécifier le nom (ou l'ID) du conteneur ainsi que la commande à exécuter sur ce conteneur. **\$ docker exec <options> <container> <command>**

À titre d'exemple, disons que vous souhaitez exécuter la commande «ls» sur l'un de vos conteneurs.

```
bilel@bilel-G3-3590:~$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS               NAMES
46cf753d8d6        httpd              "httpd-foreground"   5 minutes ago    Up 5 minutes          80/tcp            devops1
bilel@bilel-G3-3590:~$ docker exec 46cf753d8d6 ls
bin
build
cgi-bin
conf
error
htdocs
icons
include
logs
modules
bilel@bilel-G3-3590:~$ 
```

Autres exemple

```
bilel@bilel-G3-3590:~$ docker exec 46cf753d8d6 pwd
/usr/local/apache2
bilel@bilel-G3-3590:~$ docker exec 46cf753d8d6 whoami
root
bilel@bilel-G3-3590:~$ docker exec 46cf753d8d6 uname -a
Linux 46cf753d8d6 5.4.0-58-generic #64-Ubuntu SMP Wed Dec 9 08:16:25 UTC 2020 x86_64 GNU/Linux
bilel@bilel-G3-3590:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Docker Exec Bash

L'utilisation la plus courante de la commande « docker exec » est de lancer un terminal Bash dans un conteneur.

Afin de démarrer un shell Bash dans un conteneur Docker, exécutez la commande «docker exec» avec l'option «-it» et spécifiez l'ID du conteneur ainsi que le chemin vers le shell bash.

Si le Bash fait partie de votre PATH, vous pouvez simplement taper «bash» et avoir un terminal Bash dans votre conteneur, ou /bin/bash

Exemple :

```
bilel@bilel-G3-3590:~$ docker exec -it 46cf753d8d6 /bin/bash
root@46cf753d8d6:/usr/local/apache2#
root@46cf753d8d6:/usr/local/apache2# uname -a
Linux 46cf753d8d6 5.4.0-58-generic #64-Ubuntu SMP Wed Dec 9 08:16:25 UTC 2020 x86_64 GNU/Linux
root@46cf753d8d6:/usr/local/apache2# pwd
/usr/local/apache2
root@46cf753d8d6:/usr/local/apache2# 
```

Lors de l'exécution de cette commande, vous disposerez d'un terminal Bash interactif où vous pourrez exécuter toutes les commandes que vous souhaitez.

Pour sortir du conteneur et revient vers le terminal de votre hôte, utilise la commande exit

```
root@46cf753d8d6:/usr/local/apache2# pwd
/usr/local/apache2
root@46cf753d8d6:/usr/local/apache2# exit
exit
bilel@bilel-G3-3590:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Comment je peux tester mon application, comment je peux connecter sur mon application déployer sur un conteneur docker depuis l'extérieur

Comme on a vu déjà, la création d'un conteneur se fait avec la commande `$ docker run -itd --name <nom de conteneur> <image de conteneur:tag>`

Exemple : je vais créer un conteneur docker à base d'un image docker NGINX et le nommé Myapplication et essayé de connecter sur la page web de nginx avec l'url :

`http://adresse_ip_serveur_docker:80` =====> si le serveur docker est dans la même plage d'adresse que ma machine (LAN)

`http://adresse_ip_publique_serveur:80` =====> si le serveur est exposé à l'internet, on utilise l'adresse ip publique

NGINX : est un serveur Web comme apache et accessible sur le port 80 par défaut

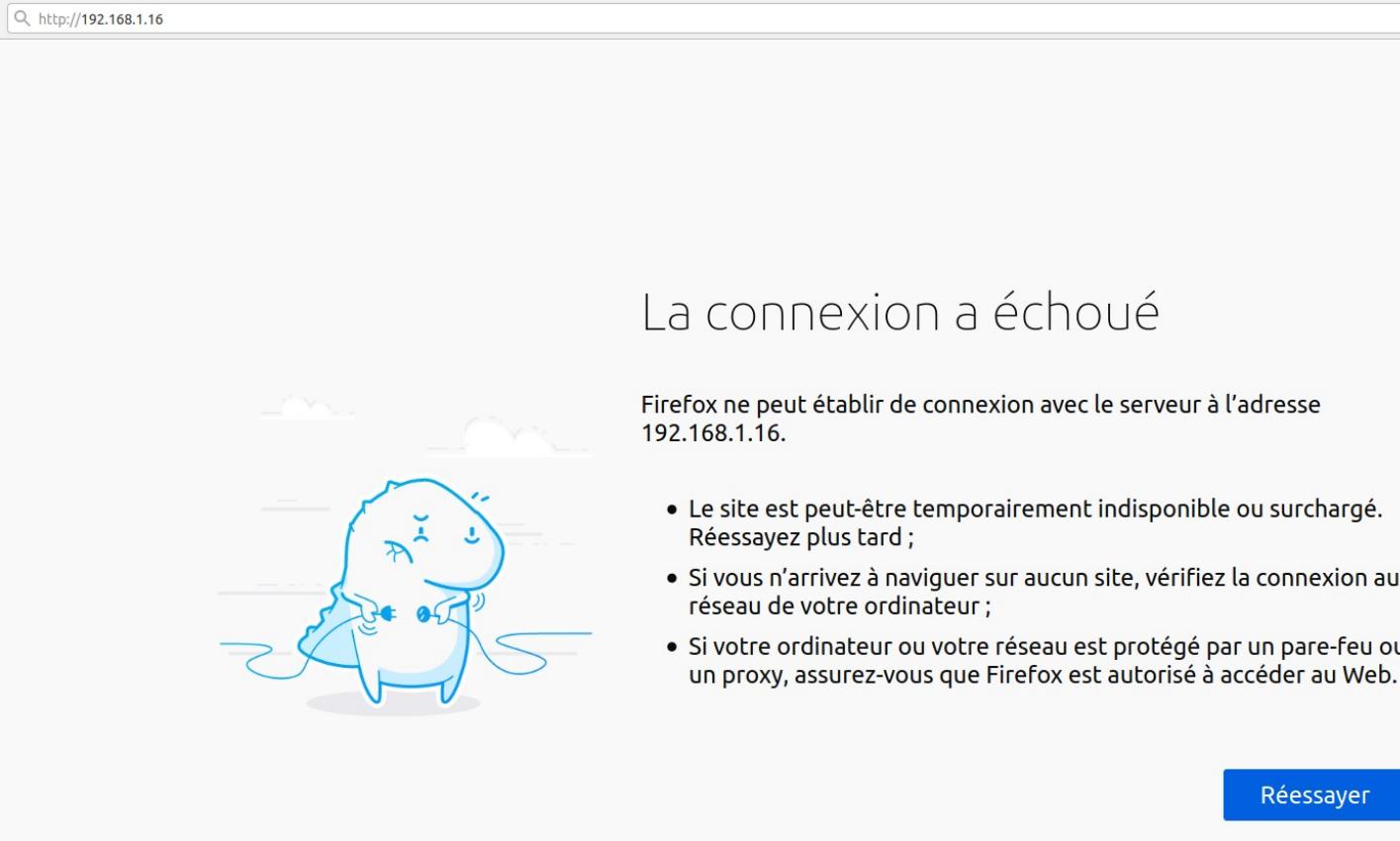
```
bilel@bilel-G3-3590:~$ docker run -itd --name myapplication nginx:alpine
Unable to find image 'nginx:alpine' locally
alpine: Pulling from library/nginx
801bfaa63ef2: Already exists
b1242e25d284: Pull complete
7453d3e6b909: Pull complete
07ce7418c4f8: Pull complete
e295e0624aa3: Pull complete
Digest: sha256:c2ce58e024275728b00a554ac25628af25c54782865b3487b11c21cafb7fabda
Status: Downloaded newer image for nginx:alpine
31edbfff1ef261316abba672766c36fb9fc22b0352c475ff9dfd3130790a4cd1
bilel@bilel-G3-3590:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
31edbfff1ef2        nginx:alpine        "/docker-entrypoint...."   7 seconds ago       Up 5 seconds          80/tcp                myapplication
bilel@bilel-G3-3590:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Mon IP locale est : 192.168.1.16



http://192.168.1.16

La connexion a échoué

Firefox ne peut établir de connexion avec le serveur à l'adresse 192.168.1.16.

- Le site est peut-être temporairement indisponible ou surchargé. Réessayez plus tard ;
- Si vous n'arrivez à naviguer sur aucun site, vérifiez la connexion au réseau de votre ordinateur ;
- Si votre ordinateur ou votre réseau est protégé par un pare-feu ou un proxy, assurez-vous que Firefox est autorisé à accéder au Web.

Réessayer



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

La connexion a échoué, c'est quoi le problème ? et comment je peux le résolut ?!

Ici on ne peut pas accéder de l'extérieur sur le conteneur car le conteneur est tourné sur un réseaux virtuelle dans le serveur. (On va le voir dans le chapitre Docker network).

Docker

La solution est d'ajouter l'argument -p dans la commande docker run :

```
$ docker run -itd --name <nom_container> -p <port_host> :<port_container> <image_docker:tag>
```

-p : permet de binder un port sur le conteneur vers un port sur le hôte

exemple

```
bilel@bilel-G3-3590:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
741ebc7e0751        nginx:alpine      "/docker-entrypoint..."   6 seconds ago       Up 4 seconds          0.0.0.0:80->80/tcp   myapplication
bilel@bilel-G3-3590:~$ 
```

On constate ici que le conteneur est accessible depuis n'importe quelle adresse ip « 0.0.0.0 » sur le port 80 du hôte.

On peut choisir n'importe quel port du host pour le mapper vers le port de conteneur, mais faites attention de ne pas utiliser la même port pour une autre conteneur

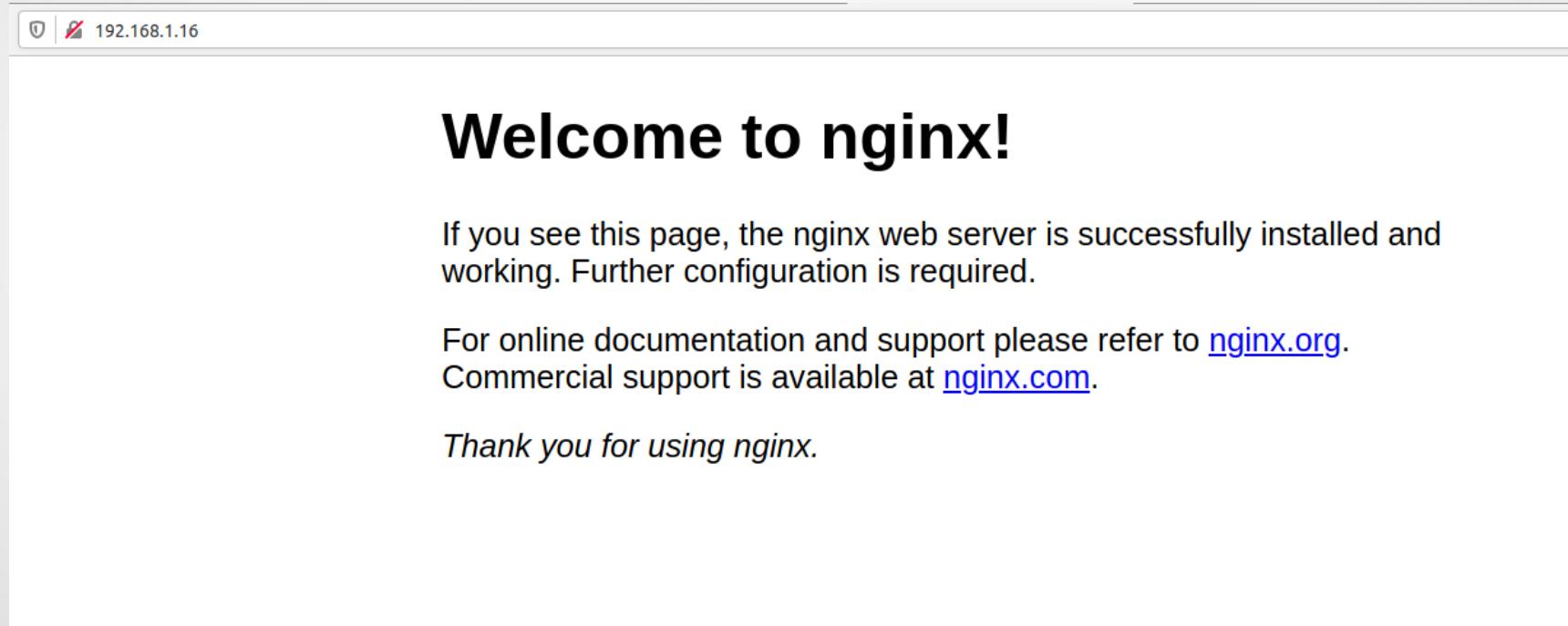
```
bilel@bilel-G3-3590:~$ docker run -itd --name myapplication -p 80:80 nginx:alpine
741ebc7e07513e0055c27b65e0cbe7ef3c06590f8fa522e7e70e1fbb2d71fe33
bilel@bilel-G3-3590:~$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                 NAMES
741ebc7e0751        nginx:alpine      "/docker-entrypoint..."   6 seconds ago       Up 4 seconds          0.0.0.0:80->80/tcp   myapplication
bilel@bilel-G3-3590:~$ docker run -itd --name myapplication1 -p 80:80 nginx:alpine
729fd0934ac0dfac38d890caa503e2db1a9a2e3929c80e83f701b2498eb69f19
docker: Error response from daemon: driver failed programming external connectivity on endpoint myapplication1 (6a9f76f6a27c0a0d52db1f507481837d16cbb8
8dc90d002ad2d6148d2461599c): Bind for 0.0.0.0:80 failed: port is already allocated.
bilel@bilel-G3-3590:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Maintenant notre application est accessible depuis l'extérieur



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

11- Docker Volumes :

Le problème ici que si nous supprimons le conteneur, les données seront supprimées. L'idée est de savoir comment conserver les données, même avec la suppression du conteneur?

Afin de comprendre ce qu'est un **Docker Volume**, nous devons d'abord savoir comment le système de fichiers fonctionne normalement dans Docker. Une image Docker se compose d'un ensemble des couches (Layers) en lecture seule. Lorsque vous lancez un conteneur à partir d'une image, Docker ajoute au sommet de cette pile de layers un nouveau layer en lecture-écriture. Docker appelle cette combinaison de couches un "Union File System".

- Lors d'une modification de fichier, Docker crée une copie depuis les couches en lecture seule vers le layer en lecture-écriture.
- Lors d'une création de fichier, Docker crée le fichier que sur le layer en lecture-écriture, et ne touche pas au layer en lecture seule.
- Lors d'une suppression de fichier, Docker supprime le fichier que sur le layer en lecture-écriture, et si il est déjà existant dans le layer en lecture seule alors il le garde.

Les données dans le layer de base sont en lecture seule, elles sont donc protégées et intactes par toutes modifications, seul le layer en lecture-écriture est impacté lors de modifications de données.

Lorsqu'un conteneur est supprimé, le layer en lecture-écriture est supprimé avec. Cela signifie que toutes les modifications apportées après le lancement du conteneur auront disparus avec.

Afin de pouvoir sauvegarder (conserver) des données et également de partager des données entre conteneurs, Docker a proposé le concept de volumes . Tout simplement, les volumes sont des répertoires (ou fichiers) qui sont en dehors du système de fichiers Union par défaut et qui existent en tant que répertoires et fichiers normaux sur le système de fichiers hôte.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Nous allons voir ici quelques possibilités que nous offre cette commande.

La syntaxe reste dans l'esprit de docker :

```
bilel@bilel-G3-3590:~$ docker volume --help

Usage: docker volume COMMAND

Manage volumes

Commands:
  create      Create a volume
  inspect     Display detailed information on one or more volumes
  ls          List volumes
  prune       Remove all unused local volumes
  rm          Remove one or more volumes

Run 'docker volume COMMAND --help' for more information on a command.
bilel@bilel-G3-3590:~$ █
```

Docker volume create <nom_volume>: crée un volume

Docker volume inspect <nom_volume>: afficher les informations complètes d'un ou plusieurs volumes

Docker volume ls : Lister les volumes docker

Docker volume prune : Supprimer les volumes local non utilisable

Docker volume rm : Supprimer un ou plusieurs volumes



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Méthode 1

Jusque-là, nous utilisions l'option **-v** avec un **docker run**, genre **\$ docker run -itd --name <container_name> -v /path/on/host:/path/on/container image**
Donc on va créer notre premier volume

```
ubuntu@vps-c59c6930:~$ docker volume create --name test
test
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER      VOLUME NAME
local        test
ubuntu@vps-c59c6930:~$ □
```

Que nous utiliserons comme ceci :

```
ubuntu@vps-c59c6930:~$ docker run -itd --name docker_volume -v test:/test alpine
Unable to find image 'alpine:latest' locally
latest: Pulling from library/alpine
801bfaa63ef2: Pull complete
Digest: sha256:3c7497bf0c7af93428242d6176e8f7905f2201d8fc5861f45be7a346b5f23436
Status: Downloaded newer image for alpine:latest
1e256c10de2fee388586bbe27064907c24853ddf7f8be0b2c39cba872481e53f
ubuntu@vps-c59c6930:~$ □
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel,
il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

On a créé un fichier dans le path du conteneur

```
ubuntu@vps-c59c6930:~$ docker exec -ti docker_volume sh
/ # ls
bin   etc   lib   mnt   proc   run   srv   test   usr
dev   home  media  opt   root   sbin  sys   tmp   var
/ # cd test
/test # ls
/test # touch exemple.txt
/test # exit
ubuntu@vps-c59c6930:~$ 
```

Si je crée un autre conteneur, on retrouve notre fichier exemple.txt

```
ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS          PORTS          NAMES
1e256c10de2f        alpine      "/bin/sh"     8 minutes ago   Up 8 minutes           docker_volume
ubuntu@vps-c59c6930:~$ docker run -itd --name docker_volume2 -v test:/test alpine
6f702a4d0db1100232b232b9afbfb957920ac713b632fc47addfc3c6d969ec77
ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS          PORTS          NAMES
6f702a4d0db1        alpine      "/bin/sh"     6 seconds ago   Up 4 seconds           docker_volume2
1e256c10de2f        alpine      "/bin/sh"     8 minutes ago   Up 8 minutes           docker_volume
ubuntu@vps-c59c6930:~$ docker exec -ti docker_volume2 sh
/ # ls test/
exemple.txt
/ # 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

On retrouve bien notre fichier, les données ont donc correctement persisté. mais les fichiers, ils sont où sur l'hôte ?

```
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER      VOLUME NAME
local        test
ubuntu@vps-c59c6930:~$ docker volume inspect test
[
  {
    "CreatedAt": "2021-01-02T11:13:38Z",
    "Driver": "local",
    "Labels": {},
    "Mountpoint": "/var/lib/docker/volumes/test/_data",
    "Name": "test",
    "Options": {},
    "Scope": "local"
  }
]
ubuntu@vps-c59c6930:~$ █
```

Les fichiers se retrouvent dans `/var/lib/docker/volumes/<volumename>/_data`.

```
ubuntu@vps-c59c6930:~$ sudo ls /var/lib/docker/volumes/test/_data
exemple.txt
ubuntu@vps-c59c6930:~$ █
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Je vais supprimé les deux conteneurs et vérifier est ce que je trouve mon fichier sur le hôte

```
ubuntu@vps-c59c6930:~$ sudo ls /var/lib/docker/volumes/test/_data
exemple.txt
ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS          PORTS          NAMES
6f702a4d0db1        alpine      "/bin/sh"     9 minutes ago   Up 9 minutes   docker_volume2
1e256c10de2f        alpine      "/bin/sh"     18 minutes ago  Up 18 minutes   docker_volume
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER    VOLUME NAME
local     test
ubuntu@vps-c59c6930:~$ docker rm -f $(docker ps -a -q)
6f702a4d0db1
1e256c10de2f
ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS          PORTS          NAMES
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER    VOLUME NAME
local     test
ubuntu@vps-c59c6930:~$ sudo ls /var/lib/docker/volumes/test/_data
exemple.txt
ubuntu@vps-c59c6930:~$ █
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Méthode 2

Docker volume peut aussi monter un répertoire local avec un conteneur Docker

```
ubuntu@vps-c59c6930:~$ mkdir Mon_repo
ubuntu@vps-c59c6930:~$ cd Mon_repo/
ubuntu@vps-c59c6930:~/Mon_repo$ touch fichier.sh
ubuntu@vps-c59c6930:~/Mon_repo$ docker volume ls
DRIVER      VOLUME NAME
local       test
ubuntu@vps-c59c6930:~/Mon_repo$ docker run -itd --name docker_mount --mount type=bind,source=/home/ubuntu/Mon_repo,target=/test alpine
0222c26e4f9176404dbdbdc5f7a3d1c19bb2dccb0595bcd1cb11001dd99db9e
ubuntu@vps-c59c6930:~/Mon_repo$ docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
0222c26e4f91   alpine     "/bin/sh"    6 seconds ago  Up 5 seconds          docker_mount
ubuntu@vps-c59c6930:~/Mon_repo$ docker exec -ti 0222c26e4f91 sh
/ # ls test/
fichier.sh
/ # exit
ubuntu@vps-c59c6930:~/Mon_repo$ docker volume ls
DRIVER      VOLUME NAME
local       test
ubuntu@vps-c59c6930:~/Mon_repo$ 
```

Ici : lorsqu'on a créé un fichier sur la machine locale linux, on a trouvé ce fichier sur le container avec l'option mount type=bind



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Exemple :

- 1- dans notre exemple ici je vais créer avec une seule ligne de commande :
- un volume : **Mon_Volume**
- un conteneur Nginx : **MyNginx**
- liée mon volume avec la répertoire **/usr/share/nginx/html** du conteneur
- binder le port 80 du conteneur vers le port 82 du hôte

```
ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER VOLUME NAME
ubuntu@vps-c59c6930:~$ docker run -itd --name MyNginx -v Mon_volume:/usr/share/nginx/html -p 82:80 nginx:alpine
Unable to find image 'nginx:alpine' locally
alpine: Pulling from library/nginx
801bfaa63ef2: Already exists
b1242e25d284: Pull complete
7453d3e6b909: Pull complete
07ce7418c4f8: Pull complete
e295e0624aa3: Pull complete
Digest: sha256:c2ce58e024275728b00a554ac25628af25c54782865b3487b11c21cafb7fabda
Status: Downloaded newer image for nginx:alpine
ee09e8c823bed11930fe4b2be4a2c62be83def96a9ba72706251de2b211877ca
ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES
ee09e8c823be nginx:alpine "/docker-entrypoint...." 5 seconds ago Up 4 seconds 0.0.0.0:82->80/tcp MyNginx
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER VOLUME NAME
local Mon_volume
ubuntu@vps-c59c6930:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Le path **/usr/share/nginx/html** contiens le fichier index de nginx

```
ubuntu@vps-c59c6930:~$ docker exec -ti ee09e8c823be sh
/ # cd /usr/share/nginx/html/
/usr/share/nginx/html # ls
50x.html      index.html
/usr/share/nginx/html # cat index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Ici j'ai vérifier le continue du volume docker

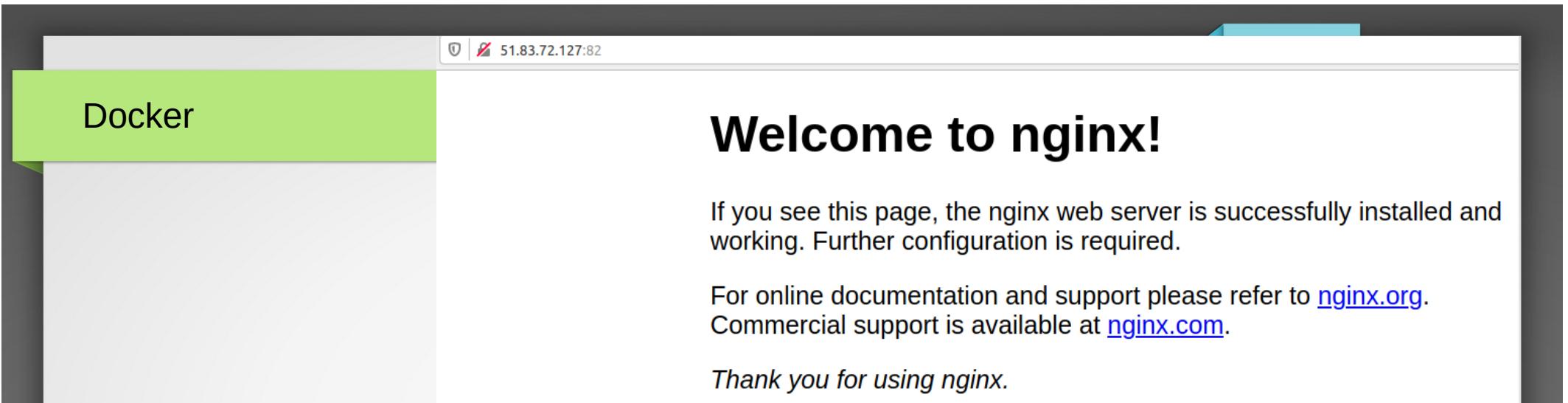
Note : Si vous démarrez un conteneur avec un volume qui n'existe pas encore, Docker le créera pour vous.

```
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER      VOLUME NAME
local       Mon_volume
ubuntu@vps-c59c6930:~$ docker volume inspect Mon_volume
[
    {
        "CreatedAt": "2021-01-02T12:02:23Z",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/Mon_volume/_data",
        "Name": "Mon_volume",
        "Options": null,
        "Scope": "local"
    }
]
ubuntu@vps-c59c6930:~$ sudo ls /var/lib/docker/volumes/Mon_volume/_data
50x.html  index.html
ubuntu@vps-c59c6930:~$ sudo cat /var/lib/docker/volumes/Mon_volume/_data/index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Ici j'ai modifier le fichier index.html situé dans le path de mon hôte de docker volume

```
ubuntu@vps-c59c6930:~$ sudo nano /var/lib/docker/volumes/Mon_volume/_data/index.html
ubuntu@vps-c59c6930:~$ sudo cat /var/lib/docker/volumes/Mon_volume/_data/index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to to nginx</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to Devops !! Docker_Volume</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Actualiser la page pour
Vérifier le changement

Welcome to Devops !! Docker_Volume

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

Ici j'ai supprimer le conteneur et vérifier la présence du fichier index.html dans le hote

```
ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID        IMAGE               COMMAND
ee09e8c823be      nginx:alpine      "/docker-entrypoint..."   4 hours ago    Up 4 hours          0.0.0.0:82->80/tcp    MyNginx
ubuntu@vps-c59c6930:~$ docker rm -f ee09e8c823be
ee09e8c823be
ubuntu@vps-c59c6930:~$ sudo cat /var/lib/docker/volumes/Mon_volume/_data/index.html
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to Devops !! Docker_Volume</h1>
```



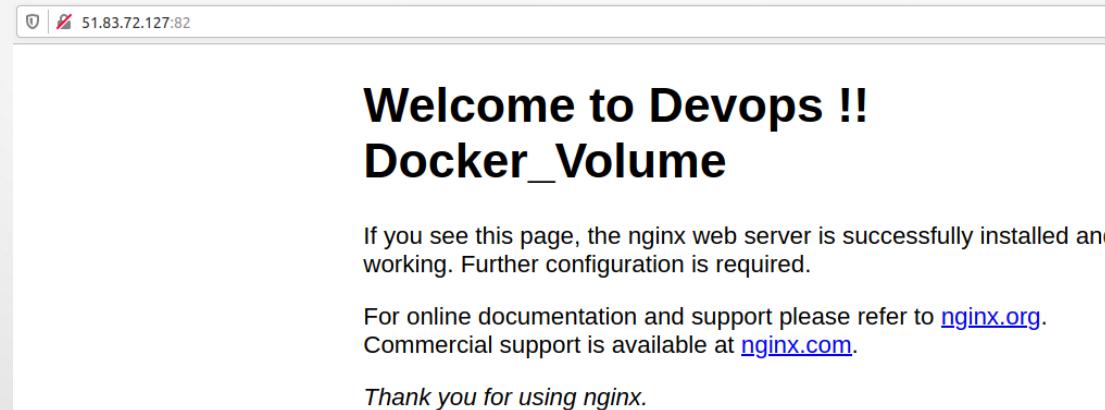
Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Ici j'ai crée une autre conteneur Nginx et l'attacher a la docker volume Mon_Volume pour réutilisé mon fichier index

```
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER      VOLUME NAME
local      Mon_volume
local      test

ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
ubuntu@vps-c59c6930:~$ docker run -itd --name test2 -v Mon_volume:/usr/share/nginx/html -p 82:80 nginx:alpine
3c9ec1cd9b7f94ae45f7595be158d7f46be9e8f70bb09db8798309b2ce7bfbbf
ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
3c9ec1cd9b7f   nginx:alpine    "/docker-entrypoint...."   9 seconds ago   Up 8 seconds   0.0.0.0:82->80/tcp   test2
ubuntu@vps-c59c6930:~$
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Lister les volumes docker dans mon hôte

```
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER      VOLUME NAME
local      Mon_volume
local      test
```

Supprimer tous les volumes docker

```
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER      VOLUME NAME
local      Mon_volume
local      test1
local      test2
local      test3
ubuntu@vps-c59c6930:~$ docker volume rm $(docker volume ls -q)
Mon_volume
test1
test2
test3
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER      VOLUME NAME
ubuntu@vps-c59c6930:~$
```

Supprimer le volume test

```
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER      VOLUME NAME
local      Mon_volume
local      test
ubuntu@vps-c59c6930:~$ docker volume rm test
test
ubuntu@vps-c59c6930:~$ docker volume ls
DRIVER      VOLUME NAME
local      Mon_volume
ubuntu@vps-c59c6930:~$
```

Note : on ne peut pas supprimer un volume attaché à un conteneur en cours d'exécution



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

11- Docker Network :

Lors de l'installation de Docker, trois réseaux sont créés automatiquement. On peut voir ces réseaux avec la commande **docker network ls**. Un réseau de type bridge est créé : Par défaut, nous avons déjà un réseau bridge, un réseau host et un réseau none. Nous ne pouvons pas créer de réseau host ou none supplémentaire. Ce chapitre expliquera donc l'utilisation des réseaux bridge

```
ubuntu@vps-c59c6930:~$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
a7791825e989    bridge    bridge      local
2283d90bf12e    host      host      local
4f0c4ec0018e    none      null      local
ubuntu@vps-c59c6930:~$ 
```

Le réseau Bridge est présent sur tous les hôtes Docker. Lors de la création d'un conteneur, si l'on ne spécifie pas un réseau particulier, les conteneurs sont connectés au Bridge **docker0**.

La commande **ifconfig** ou **ip a** fournit les informations sur le réseau bridge.

```
ubuntu@vps-c59c6930:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
      valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
      valid_lft forever preferred_lft forever
2: ens3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default
  link/ether fa:16:3e:25:0d:b9 brd ff:ff:ff:ff:ff:ff
    inet 51.83.72.127/32 scope global dynamic ens3
      valid_lft 47327sec preferred_lft 47327sec
    inet6 fe80::f816:3eff:fe25:db9%64 scope link
      valid_lft forever preferred_lft forever
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN group default
  link/ether 02:42:99:9c:8e:48 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.1/16 brd 172.17.255.255 scope global docker0
      valid_lft forever preferred_lft forever
    inet6 fe80::42:99ff:fe9c:8e48%64 scope link
      valid_lft forever preferred_lft forever
ubuntu@vps-c59c6930:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

La commande **docker network inspect bridge**, retourne les informations concernant ce réseau :

Subnet : 172.17.0.0 est l'adresse de sous réseaux défini par défaut sur tous Hôtes docker

Gateway : 172.17.0.1 est l'adresse de l'interface Docker0.
 L'interface docker0 joue le rôle d'un passerelle et d'un serveur DHCP pour Les conteneurs.

```
ubuntu@vps-c59c6930:~$ docker network inspect bridge
[
    {
        "Name": "bridge",
        "Id": "a7791825e989b8d768da3c643a3dc63bdf49c24f5d2cb72f4711541168fe0779",
        "Created": "2020-12-31T12:54:25.75406708Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.17.0.0/16",
                    "Gateway": "172.17.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {
            "com.docker.network.bridge.default_bridge": "true",
            "com.docker.network.bridge.enable_icc": "true",
            "com.docker.network.bridge.enable_ip_masquerade": "true",
            "com.docker.network.bridge.host_binding_ipv4": "0.0.0.0",
            "com.docker.network.bridge.name": "docker0",
            "com.docker.network.driver.mtu": "1500"
        },
        "Labels": {}
    }
]
ubuntu@vps-c59c6930:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Créons deux containers avec l'image alpine.

```
ubuntu@vps-c59c6930:~$ docker run -tid --name container1 alpine
8ddf3f03f9edf3464c204b20a7473ccfc1f335036c7e0e0a545493bc1c17b9f9
ubuntu@vps-c59c6930:~$ docker run -tid --name container2 alpine
e616072f3247a7fb8effd8ac41a072ab1f6228819854e5e339ed1429c34218bf
ubuntu@vps-c59c6930:~$
```

Et visualisons les informations du réseau avec docker network inspect :

Les conteneurs sont connectés au Bridge par défaut docker0 et peuvent communiquer entre eux par adresse IP, les conteneurs se trouvent alors, sur le même réseau.

Subnet : 172.17.0.0

Gateway : 172.17.0.1

Container1 : 172.17.0.2

Container2 : 172.17.0.3

```
{
  "Name": "bridge",
  "Id": "a7791825e989b8d768da3c643a3dc63bd49c24f5d2cb72f4711541168fe0779",
  "Created": "2020-12-31T12:54:25.75406708Z",
  "Scope": "local",
  "Driver": "bridge",
  "EnableIPv6": false,
  "IPAM": {
    "Driver": "default",
    "Options": null,
    "Config": [
      {
        "Subnet": "172.17.0.0/16",
        "Gateway": "172.17.0.1"
      }
    ],
    "Internal": false,
    "Attachable": false,
    "Ingress": false,
    "ConfigFrom": {
      "Network": ""
    },
    "ConfigOnly": false,
    "Containers": {
      "8ddf3f03f9edf3464c204b20a7473ccfc1f335036c7e0e0a545493bc1c17b9f9": {
        "Name": "container1",
        "EndpointID": "0de450822e5334fd09e0957933dc804e6814f4e0964efd1b7d29eb91ea9704ce",
        "MacAddress": "02:42:ac:11:00:02",
        "IPv4Address": "172.17.0.2/16",
        "IPv6Address": ""
      },
      "e616072f3247a7fb8effd8ac41a072ab1f6228819854e5e339ed1429c34218bf": {
        "Name": "container2",
        "EndpointID": "3b2ef75dcc907582c182e9f1a59de3945368f3e99f4a4bc649be94595649b821",
        "MacAddress": "02:42:ac:11:00:03",
        "IPv4Address": "172.17.0.3/16",
        "IPv6Address": ""
      }
    }
  }
}
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Exemple : Dans cet exemple nous allons créer un réseau de type bridge nommé mon-bridge :

```
ubuntu@vps-c59c6930:~$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
a7791825e989    bridge    bridge      local
2283d90bf12e    host      host       local
4f0c4ec0018e    none      null       local
ubuntu@vps-c59c6930:~$ docker network create --driver bridge mon-bridge
deaf27230db000058f947f0fb1b79ebd1ed83e2b4c45f4a3404cb1e25fa83385
ubuntu@vps-c59c6930:~$ █
```

On va ensuite lister les réseaux docker avec la commande suivante :

```
ubuntu@vps-c59c6930:~$ docker network create --driver bridge mon-bridge
deaf27230db000058f947f0fb1b79ebd1ed83e2b4c45f4a3404cb1e25fa83385
ubuntu@vps-c59c6930:~$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
a7791825e989    bridge    bridge      local
2283d90bf12e    host      host       local
deaf27230db0    mon-bridge  bridge      local
4f0c4ec0018e    none      null       local
ubuntu@vps-c59c6930:~$ █
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Il est possible de récolter des informations sur le réseau docker, comme par exemple la config réseau, en tapant la commande suivante :

```
ubuntu@vps-c59c6930:~$ docker network inspect mon-bridge
[
    {
        "Name": "mon-bridge",
        "Id": "deaf27230db000058f947f0fb1b79ebd1ed83e2b4c45f4a3404cb1e25fa83385",
        "Created": "2021-01-05T11:54:15.845542208Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": {},
            "Config": [
                {
                    "Subnet": "172.18.0.0/16",
                    "Gateway": "172.18.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": false,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {},
        "Options": {},
        "Labels": {}
    }
]
ubuntu@vps-c59c6930:~$ █
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Pour cet exemple, nous allons connecter deux conteneurs à notre réseau bridge créé précédemment :

```
ubuntu@vps-c59c6930:~$ docker run -dit --name alpine1 --network mon-bridge alpine
3cfef53b40be9157259f10d39d8f3d521bb9315aa88d93bdःfa4ea35748487e20
ubuntu@vps-c59c6930:~$ docker run -dit --name alpine2 --network mon-bridge alpine
a4a48b135f7e2fcea8224e74d1c9d88823d1648773ba439299450cd2fdb3015e
ubuntu@vps-c59c6930:~$ █
```

Si on inspecte une nouvelle fois notre réseau mon-bridge, on verra nos deux nouveaux conteneurs dans les informations retournées :

```
{
  "Containers": {
    "3cfef53b40be9157259f10d39d8f3d521bb9315aa88d93bdःfa4ea35748487e20": {
      "Name": "alpine1",
      "EndpointID": "72964c14f9d3969c5a74fe8fe3965353f8d8619d7573ae094f9b846fee636572",
      "MacAddress": "02:42:ac:12:00:02",
      "IPv4Address": "172.18.0.2/16",
      "IPv6Address": ""
    },
    "a4a48b135f7e2fcea8224e74d1c9d88823d1648773ba439299450cd2fdb3015e": {
      "Name": "alpine2",
      "EndpointID": "b78a3877ed9a71cle720712e3f0bec61e6a8192420332d21e28c83cda66e1133",
      "MacAddress": "02:42:ac:12:00:03",
      "IPv4Address": "172.18.0.3/16",
      "IPv6Address": ""
    }
}
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

D'après le résultat, on peut s'apercevoir que notre conteneur alpine1 possède l'adresse IP 172.18.0.2, et notre conteneur alpine2 possède l'adresse IP 172.18.0.3. Tentons de les faire communiquer ensemble à l'aide de la commande ping :

```
ubuntu@vps-c59c6930:~$ docker exec alpine1 ping -c 1 172.18.0.3
PING 172.18.0.3 (172.18.0.3): 56 data bytes
64 bytes from 172.18.0.3: seq=0 ttl=64 time=0.278 ms

--- 172.18.0.3 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.278/0.278/0.278 ms
ubuntu@vps-c59c6930:~$ docker exec alpine1 ping -c 1 172.18.0.2
PING 172.18.0.2 (172.18.0.2): 56 data bytes
64 bytes from 172.18.0.2: seq=0 ttl=64 time=0.134 ms

--- 172.18.0.2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.134/0.134/0.134 ms
ubuntu@vps-c59c6930:~$ █
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Supprimer, et connecter un réseau Docker : Avant de supprimer votre réseau docker, il est nécessaire au préalable de supprimer tout conteneur connecté à votre réseau docker, ou sinon il suffit juste de déconnecter votre conteneur de votre réseau docker sans forcément le supprimer.

Nous allons choisir la méthode 2, en déconnectant tous les conteneurs utilisant le réseau docker mon-bridge :

```
ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID   IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
a4a48b135f7e   alpine     "/bin/sh"    14 minutes ago   Up 14 minutes
3cefef53b40be   alpine     "/bin/sh"    14 minutes ago   Up 14 minutes
ubuntu@vps-c59c6930:~$ docker network disconnect mon-bridge alpinel
ubuntu@vps-c59c6930:~$ docker network disconnect mon-bridge alpine2
ubuntu@vps-c59c6930:~$ █
```

Maintenant, si vous vérifiez les interfaces réseaux de vos conteneurs basées sur l'image alpine, vous ne verrez que l'interface loopback comme pour le driver none :

```
ubuntu@vps-c59c6930:~$ docker exec alpinel ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 scope host lo
            valid_lft forever preferred_lft forever
ubuntu@vps-c59c6930:~$ █
```

Une fois que vous avez déconnecté tous vos conteneurs du réseau docker mon-bridge, vous pouvez alors le supprimer :

```
ubuntu@vps-c59c6930:~$ docker network rm mon-bridge
mon-bridge
ubuntu@vps-c59c6930:~$ docker network ls
NETWORK ID      NAME      DRIVER      SCOPE
a7791825e989    bridge    bridge      local
2283d90bf12e    host      host      local
4f0c4ec0018e    none      null      local
ubuntu@vps-c59c6930:~$ █
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Cependant vos conteneurs se retrouvent maintenant sans interface réseau bridge, il faut donc reconnecter vos conteneurs au réseau bridge par défaut pour qu'ils puissent de nouveau communiquer entre eux :

```
ubuntu@vps-c59c6930:~$ docker ps
CONTAINER ID        IMAGE       COMMAND       CREATED          STATUS          PORTS          NAMES
a4a48b135f7e        alpine      "/bin/sh"     19 minutes ago   Up 19 minutes
3cfef53b40be        alpine      "/bin/sh"     19 minutes ago   Up 19 minutes
ubuntu@vps-c59c6930:~$ docker network connect bridge alpine1
ubuntu@vps-c59c6930:~$ docker network connect bridge alpine2
ubuntu@vps-c59c6930:~$ █
```

Vérifiez ensuite si vos conteneurs ont bien reçu la bonne Ip :

```
ubuntu@vps-c59c6930:~$ docker network connect bridge alpine1
ubuntu@vps-c59c6930:~$ docker network connect bridge alpine2
ubuntu@vps-c59c6930:~$ docker inspect -f '{{.Name}} - {{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps -aq)
/alpine2 - 172.17.0.3
/alpine1 - 172.17.0.2
ubuntu@vps-c59c6930:~$ █
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Comme d'habitude, voici l'aide-mémoire de ce cours :

- Créer un réseau docker

```
docker network create --driver <DRIVER TYPE> <NETWORK NAME>
```

- Lister les réseaux docker

```
docker network ls
```

- Supprimer un ou plusieurs réseau(x) docker

```
docker network rm <NETWORK NAME>
```

- Récolter des informations sur un réseau docker

```
docker network inspect <NETWORK NAME>
```

-v ou --verbose : mode verbose pour un meilleur diagnostique

- Supprimer tous les réseaux docker non inutilisés

```
docker network prune
```

-f ou --force : forcer la suppression

- Connecter un conteneur à un réseau docker

```
docker network connect <NETWORK NAME> <CONTAINER NAME>
```

- Déconnecter un conteneur à réseau docker

```
docker network disconnect <NETWORK NAME> <CONTAINER NAME>
```

-f ou --force : forcer la déconnexion

- Démarrer un conteneur et le connecter à un réseau docker

```
docker run --network <NETWORK NAME> <IMAGE NAME>
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

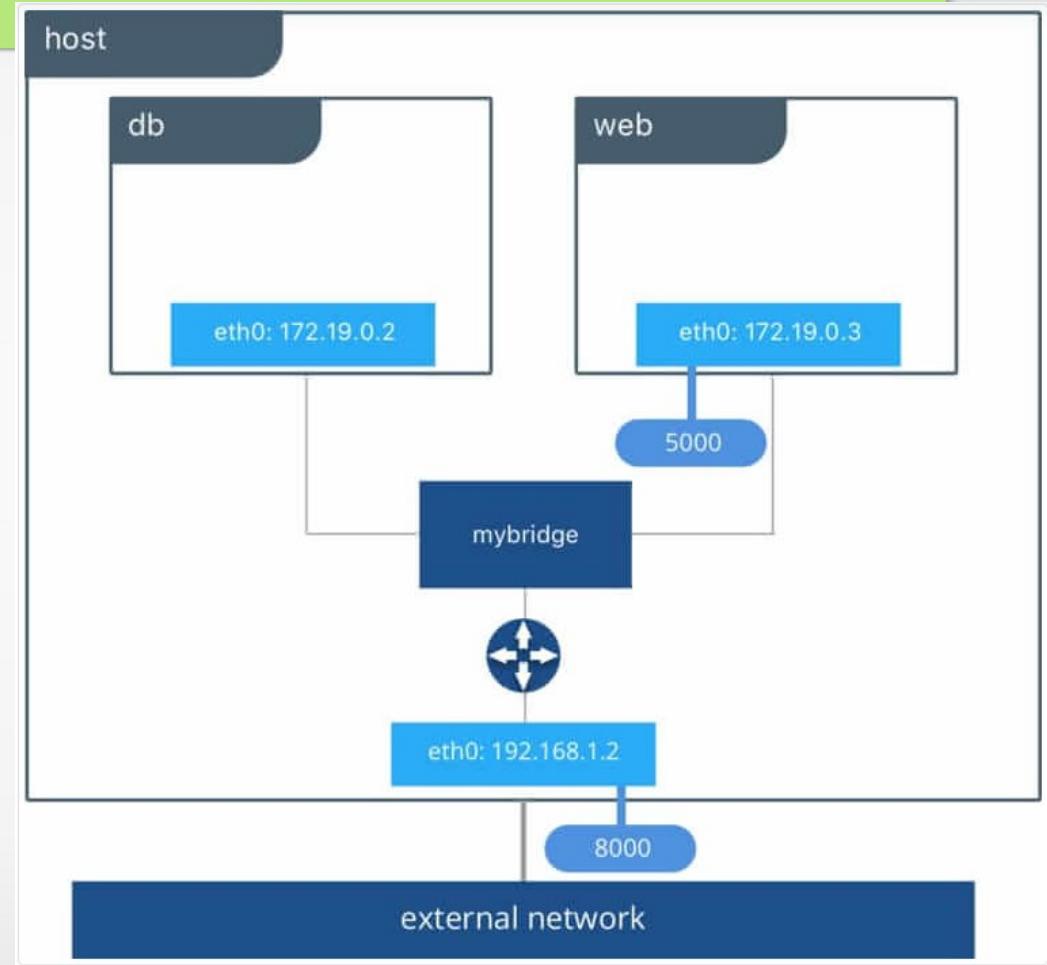
Network Drive :

1- Bridge :

Le réseau bridge est le type de réseau le plus couramment utilisé. Il est limité aux conteneurs d'un hôte unique exécutant le moteur Docker. Les conteneurs qui utilisent ce driver, ne peuvent communiquer qu'entre eux, cependant ils ne sont pas accessibles depuis l'extérieur.

Pour que les conteneurs sur le réseau bridge puissent communiquer ou être accessibles du monde extérieur, vous devez configurer le mappage de port.

Exp : **Docker run -name web -p 8000:5000 httpd**



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

None : C'est le type de réseau idéal, si vous souhaitez interdire toute communication interne et externe avec votre conteneur, car votre conteneur sera dépourvu de toute interface réseau (sauf l'interface loopback).

Host : Ce type de réseau permet aux conteneurs d'utiliser la même interface que l'hôte. Il supprime donc l'isolation réseau entre les conteneurs et seront par défaut accessibles de l'extérieur. De ce fait, il prendra la même IP que votre machine hôte. L'hôte n'est disponible que pour les services swarm sur Docker 17.06 et supérieur.

Overlay : Si vous souhaitez une mise en réseau multi-hôte native, vous devez utiliser un driver overlay. Il crée un réseau distribué entre plusieurs hôtes possédant le moteur Docker et permettent aux services Swarm de communiquer entre eux qui gère de manière transparente le routage de chaque paquet vers et depuis le bon hôte et le bon conteneur.

Macvlan : L'utilisation du driver macvlan est parfois le meilleur choix lorsque vous utilisez des applications qui s'attendent à être directement connectées au réseau physique, car le driver Macvlan vous permet d'attribuer une adresse MAC à un conteneur, le faisant apparaître comme un périphérique physique sur votre réseau. Le moteur Docker route le trafic vers les conteneurs en fonction de leurs adresses MAC.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

12- Dockerfile : Si nous voulons créer un conteneur spécifique avec une configuration spécifique, dans ce cas, nous devrons créer.

Docker vous donne également la possibilité de créer vos propres images Docker, et cela peut être fait à l'aide de Docker Files. Le Dockerfile (toujours avec une majuscule) est un fichier qui contient toutes les instructions pour créer une image, comme des métadonnées (Mainteneur, label, etc.), ou même les commandes à exécuter pour installer un logiciel.

Voici la liste des instructions d'un Dockerfile les plus communément utilisées

FROM : Définit l'image de base qui sera utilisée par les instructions suivantes.

LABEL : Ajoute des métadonnées à l'image avec un système de clés-valeurs, permet par exemple d'indiquer à l'utilisateur l'auteur du Dockerfile.

ARG : Variables temporaires qu'on peut utiliser dans un Dockerfile.

ENV : Variables d'environnements utilisables dans votre Dockerfile et conteneur.

RUN : Exécute des commandes Linux ou Windows lors de la création de l'image. Chaque instruction RUN va créer une couche en cache qui sera réutilisée dans le cas de modification ultérieure du Dockerfile.

COPY : Permet de copier des fichiers depuis notre machine locale vers le conteneur Docker.

ADD : Même chose que COPY mais prend en charge des liens ou des archives (si le format est reconnu, alors il sera décompressé à la volée).

ENTRYPOINT : comme son nom l'indique, c'est le point d'entrée de votre conteneur, en d'autres termes, c'est la commande qui sera toujours exécutée au démarrage du conteneur. Il prend la forme de tableau JSON (ex : CMD ["cmd1","cmd1"]) ou de texte.

CMD : Spécifie les arguments qui seront envoyés au ENTRYPOINT, (on peut aussi l'utiliser pour lancer des commandes par défaut lors du démarrage d'un conteneur). Si il est utilisé pour fournir des arguments par défaut pour l'instruction ENTRYPOINT, alors les instructions CMD et ENTRYPOINT doivent être spécifiées au format de tableau JSON.

WORKDIR : Définit le répertoire de travail qui sera utilisé pour le lancement des commandes CMD et/ou ENTRYPOINT et ça sera aussi le dossier courant lors du démarrage du conteneur.

EXPOSE : Expose un port.

VOLUMES : Crée un point de montage qui permettra de persister les données.

USER : Désigne quel est l'utilisateur qui lancera les prochaines instructions RUN, CMD ou ENTRYPOINT (par défaut c'est l'utilisateur root).



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Pour plus d'informations, je vous conseille de consulter la [documentation officielle](#)

Et la commande pour construire l'image : **\$ docker image build -t [imagename]:[tag] [dockerfile folder]**

Si nous voulons créer un conteneur spécifique avec une configuration spécifique, dans ce cas, nous allons utiliser une image de base avec dans nous allons ajouter une instruction spécifique afin que nous puissions avoir une image spécifique.

Exemple :

1- On commence par créer le répertoire de notre projet :

```
ubuntu@vps-c59c6930:~$ mkdir projet
ubuntu@vps-c59c6930:~$ cd projet/
[...]
```

2- Ensuite dans la racine du dossier que vous venez de créer, créez un fichier et nommez le Dockerfile, puis rajoutez le contenu

```
ubuntu@vps-c59c6930:~$ mkdir projet
ubuntu@vps-c59c6930:~$ cd projet/
ubuntu@vps-c59c6930:~/projet$ vi Dockerfile
ubuntu@vps-c59c6930:~/projet$
```

```
ubuntu@vps-c59c6930:~/projet
Fichier Édition Affichage Rechercher Terminal Aide
From nginx

WORKDIR /usr/share/nginx/html
COPY index.html /usr/share/nginx/html
~
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

3- créez un fichier et nommez le index.html, puis rajoutez le contenu suivant :

```
ubuntu@vps-c59c6930:~$ mkdir projet
ubuntu@vps-c59c6930:~$ cd projet/
ubuntu@vps-c59c6930:~/projet$ vi Dockerfile
ubuntu@vps-c59c6930:~/projet$ vi index.html
ubuntu@vps-c59c6930:~/projet$ █
```

ubuntu@vps-c59c6930:~/projet

Fichier Édition Affichage Rechercher Terminal Aide

Hi DevOps Team !!!!█

~

Voici l'architecture que vous êtes censé avoir :

```
ubuntu@vps-c59c6930:~/projet$ tree
.
└── Dockerfile
    └── index.html

0 directories, 2 files
ubuntu@vps-c59c6930:~/projet$ █
```

4- Voici la commande pour qui nous permet de construire une image docker depuis un Dockerfile :

```
ubuntu@vps-c59c6930:~/projets$ docker build -t my_nginx_image .
Sending build context to Docker daemon 3.072kB
Step 1/3 : From nginx
latest: Pulling from library/nginx
6ec7b7d162b2: Pull complete
cb420a90068e: Pull complete
2766c0bf2b07: Pull complete
e05167b6a99d: Pull complete
70ac9d795e79: Pull complete
Digest: sha256:4cf620a5c81390ee209398ecc18e5fb9dd0f5155cd82adcbae532fec94006fb9
Status: Downloaded newer image for nginx:latest
--> ae2feff98a0c
Step 2/3 : WORKDIR /usr/share/nginx/html
--> Running in c1d39b9d5fba
Removing intermediate container c1d39b9d5fba
--> 63136332bfb4
Step 3/3 : COPY index.html /usr/share/nginx/html
--> 96486e5d45f0
Successfully built 96486e5d45f0
Successfully tagged my_nginx_image:latest
ubuntu@vps-c59c6930:~/projets$ █
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Vérifier l'image

```
ubuntu@vps-c59c6930:~/projets$ docker images
REPOSITORY      TAG      IMAGE ID      CREATED      SIZE
my_nginx_image  latest   96486e5d45f0  3 minutes ago  133MB
alpine          latest   389fef711851  2 weeks ago   5.58MB
nginx           latest   ae2feff98a0c  3 weeks ago   133MB
ubuntu@vps-c59c6930:~/projets$
```

On peut la tester :

- crée un conteneur Nginx1 on utilisons nos propre image docker
- crée un conteneur Nginx2 on utilisons l'image Docker de base

- Mon image docker « **my_nginx_image** » crée depuis 3 minutes qui contienne mon propre index
- Image docker de base « nginx » contienne l'index nginx de base

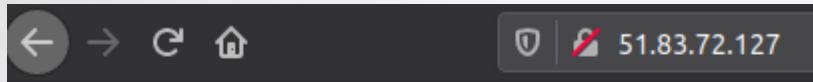
```
ubuntu@vps-c59c6930:~/projets$ docker run -tid --name Nginx1 -p 80:80 my_nginx_image
f9e8ca25a73b2b43a56d46e1e4d67bf2e18bab04dad7670b430c9d6771dc0ebc
ubuntu@vps-c59c6930:~/projets$ docker run -tid --name Nginx2 -p 82:80 nginx
b7319cb5849b7fadf53a8f4421d64ec99afc079c027195d1868079ac38b16735
ubuntu@vps-c59c6930:~/projets$ docker ps
CONTAINER ID      IMAGE      COMMAND      CREATED      STATUS      PORTS      NAMES
b7319cb5849b      nginx      "/docker-entrypoint...."  7 seconds ago  Up 6 seconds  0.0.0.0:82->80/tcp  Nginx2
f9e8ca25a73b      my_nginx_image      "/docker-entrypoint...."  29 seconds ago  Up 28 seconds  0.0.0.0:80->80/tcp  Nginx1
ubuntu@vps-c59c6930:~/projets$
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

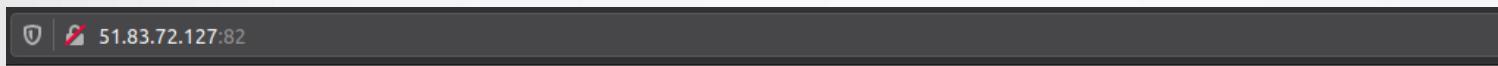
Visitez ensuite les pages index des deux conteneur Nginx, et vous obtiendrez le résultat suivant :



http://adresse_ip:80 : index de conteneur Nginx1

Hi DevOps Team !!!!

http://adresse_ip:82 : index de conteneur Nginx2



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Exemple2 : Petit rappel, une image est un modèle composé de plusieurs couches, ces couches contiennent notre application ainsi que les fichiers binaires et les bibliothèques requises.

Pour s'exercer, nous allons créer notre propre stack LAMP (Linux Apache MySQL PHP) au moyen de Docker. Voici les différentes couches de cette image :

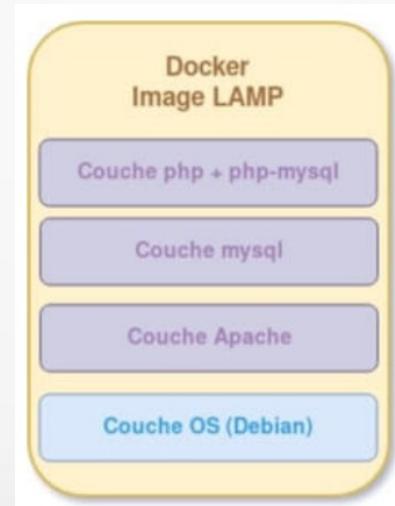
Une couche OS pour exécuter notre Apache, MySQL et PHP, je vais me baser sur la distribution Debian.

Une couche Apache pour démarrer notre serveur web.

Une couche PHP qui contiendra un interpréteur PHP mais aussi les bibliothèques qui vont avec.

Une couche Mysql qui contiendra notre système de gestion de bases de données.

Voici le schéma de notre image :



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Création de notre image

Normalement pour faire les choses dans les règles de l'art, il faut séparer l'image de l'application web par rapport à l'image de la base de données. Mais pour ce cours je vais faire une exception et je vais mettre toute notre stack dans une seule image pour bien comprendre la notion des couches.

1- Commencez par créer un dossier et téléchargez les sources de l'image : https://gitlab.com/bileli/projet_dockerfile.git

Voici l'architecture que vous êtes censé avoir :

```
ubuntu@vps-c59c6930:~/projet_dockerfile$ tree
.
├── Dockerfile
├── app
│   ├── db-config.php
│   └── index.php
└── db
    └── articles.sql

2 directories, 4 files
ubuntu@vps-c59c6930:~/projet_dockerfile$ █
```

db : contient un fichier `articles.sql`, qui renferme toute l'architecture de la base de données.

app : comporte les sources php de notre l'application web.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Explication du Dockerfile

```
# ----- DÉBUT COUCHE OS -----
FROM debian:stable-slim
# ----- FIN COUCHE OS -----
```

Pour créer ma couche OS, je me suis basée sur l'image debian-slim. Vous pouvez, choisir une autre image si vous le souhaitez (il existe par exemple une image avec une couche OS nommée alpine, qui ne pèse que 5 MB !), sachez juste qu'il faut adapter les autres instructions si jamais vous choisissez une autre image de base.

```
# MÉTADONNÉES DE L'IMAGE
LABEL version="1.0" maintainer="Issaoui Bilel <issaouuiios2015@hotmail.com>"
```

Ensuite, j'ai rajouté les métadonnées de mon image. Comme ça, si un jour je décide de partager mon image avec d'autres personnes, alors ils pourront facilement récolter des métadonnées sur l'image (ex: l'auteur de l'image) depuis la commande `docker inspect <IMAGE_NAME>`

```
# VARIABLES TEMPORAIRES
ARG APT_FLAGS="-q -y"
ARG DOCUMENTROOT="/var/www/html"
```

Ici, j'ai créé deux variables temporaires qui ne me serviront qu'au sein de mon Dockerfile, d'où l'utilisation de l'instruction `ARG`. La première variable me sert comme arguments pour la commande `apt`, et la seconde est le répertoire de travail de mon apache.

```
# ----- DÉBUT COUCHE APACHE -----
RUN apt-get update -y && \
    apt-get install ${APT_FLAGS} apache2
# ----- FIN COUCHE APACHE -----
```

Par la suite, j'ai construit ma couche Apache. Pour cela j'ai d'abord commencé par récupérer la liste de paquets et ensuite j'ai installé mon Apache.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

```
# ----- DÉBUT COUCHE MYSQL -----
RUN apt-get install ${APT_FLAGS} mariadb-server
COPY db/articles.sql /
# ----- FIN COUCHE MYSQL -----
```

Ici, je commence d'abord par télécharger le service mysql et ensuite je rajoute mon fichier articles.sql pour mon futur nouveau conteneur.

```
# ----- DÉBUT COUCHE PHP -----
RUN apt-get install ${APT_FLAGS} \
  php-mysql \
  php && \
  rm -f ${DOCUMENTROOT}/index.html && \
  apt-get autoclean -y

COPY app ${DOCUMENTROOT}
# ----- FIN COUCHE PHP -----
```

Ici j'installe l'interpréteur php ainsi que le module php-mysql. j'ai ensuite vidé le cache d'apt-get afin de gagner en espace de stockage. J'ai aussi supprimé le fichier index.html du DocumentRoot d'Apache (par défaut /var/www/html), car je vais le remplacer par mes propres sources.

# OUVERTURE DU PORT HTTP	# RÉPERTOIRE DE TRAVAIL
EXPOSE 80	WORKDIR \${DOCUMENTROOT}

J'ouvre le port HTTP et j'ai mis le dossier /var/www/html en tant que répertoire de travail, comme ça, quand je démarrerai mon conteneur, alors je serai directement sur ce dossier.

```
# DÉMARRAGE DES SERVICES LORS DE L'EXÉCUTION DE L'IMAGE
ENTRYPOINT service mysql start && mysql < /articles.sql && apache2ctl -D FOREGROUND
```

Ici, lors du lancement de mon conteneur, le service mysql démarrera et construira l'architecture de la base de données grâce à mon fichier articles.sql . Maintenant, il faut savoir qu'un conteneur se ferme automatiquement à la fin de son processus principal. Il faut donc un processus qui tourne en premier plan pour que le conteneur soit toujours à l'état running, d'où le lancement du service Apache en premier plan à l'aide de la commande apache2 -D FOREGROUND.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Construction et Exécution de notre image

Voici la commande pour qui nous permet de construire une image docker depuis un Dockerfile :

\$ docker build -t <IMAGE_NAME> .

Ce qui nous donnera :

\$ docker build -t my_lamp .

```
ubuntu@vps-c59c6930:~/projet_dockerfile$ docker build -t my_lamp .
Sending build context to Docker daemon 67.58kB
Step 1/12 : FROM debian:stable-slim
--> 2bb074a4c289
Step 2/12 : LABEL version="1.0" maintainer="Issaoui Bilel <issaouuiios2015@hotmail.com>"
--> Running in 85a1178baceb
Removing intermediate container 85a1178baceb
--> 2c94de28eed8
Step 3/12 : ARG APT_FLAGS="-q -y"
--> Running in 3394f6528250
Removing intermediate container 3394f6528250
--> bef1523a736a
Step 4/12 : ARG DOCUMENTROOT="/var/www/html"
--> Running in 2f5a5b49a321
Removing intermediate container 2f5a5b49a321
--> 2013ec68318b
Step 5/12 : RUN apt-get update -y && apt-get install ${APT_FLAGS} apache2
--> Running in 5afd27c3e033

```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Ensuite, exécutez votre image personnalisée : **\$ docker run -d --name my_app -p 8080:80 my_lamp**

```

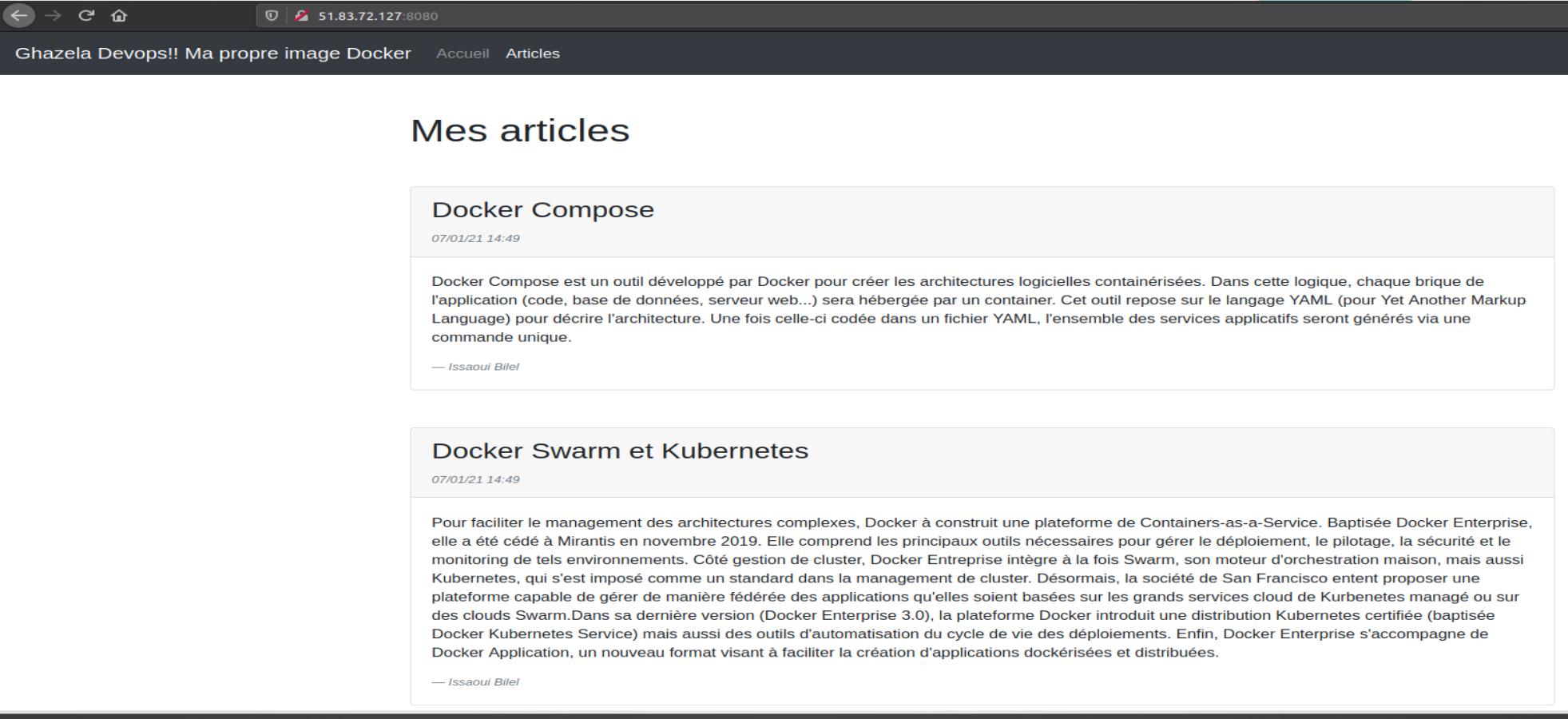
Step 9/12 : COPY app ${DOCUMENTROOT}
--> 7cf0b147857e
Step 10/12 : EXPOSE 80
--> Running in 8d803ab3b706
Removing intermediate container 8d803ab3b706
--> 07e2b8032161
Step 11/12 : WORKDIR ${DOCUMENTROOT}
--> Running in 3222c162fafafa
Removing intermediate container 3222c162fafafa
--> e560f887ebb8
Step 12/12 : ENTRYPOINT service mysql start && mysql < /articles.sql && apache2ctl -D FOREGROUND
--> Running in effa2445979a
Removing intermediate container effa2445979a
--> della21a3af5
Successfully built della21a3af5
Successfully tagged my_lamp:latest
ubuntu@vps-c59c6930:~/projet_dockerfile$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED        SIZE
my_lamp         latest       della21a3af5   4 minutes ago  501MB
alpine          latest       389fef711851   3 weeks ago   5.58MB
nginx           latest       ae2feff98a0c   3 weeks ago   133MB
debian          stable-slim  2bb074a4c289   3 weeks ago   69.2MB
ubuntu@vps-c59c6930:~/projet_dockerfile$ docker run -d --name my_app -p 8080:80 my_lamp
4a9123849ddb854a7c1e479001d90e543c4a3138bb491a7169dd6ef11d42364
ubuntu@vps-c59c6930:~/projet_dockerfile$ docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS     NAMES
4a9123849ddb   my_lamp    "/bin/sh -c 'service..."   4 seconds ago   Up 3 seconds   0.0.0.0:8080->80/tcp   my_app
ubuntu@vps-c59c6930:~/projet_dockerfile$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Visitez ensuite la page suivante http://@ip_serveur:8080 et vous obtiendrez le résultat suivant :



The screenshot shows a web browser window with a dark header bar. The address bar displays the URL <http://51.83.72.127:8080>. Below the address bar, the header includes the text "Ghazela Devops!! Ma propre image Docker" and navigation links for "Accueil" and "Articles". The main content area is titled "Mes articles". It lists two articles:

- Docker Compose** (published on 07/01/21 at 14:49). The text describes Docker Compose as a tool developed by Docker to create software architectures contained within containers. It uses YAML files to define services, which are then orchestrated into a single command. The author is listed as "Issaoui Bilel".
- Docker Swarm et Kubernetes** (published on 07/01/21 at 14:49). The text discusses Docker's transition to Docker Enterprise, which includes a platform for Container-as-a-Service. It highlights the integration of Docker Swarm and Kubernetes, mentioning Docker's role in the development of Kubernetes and its own Docker Kubernetes Service. The author is listed as "Issaoui Bilel".



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Publier son image dans Docker Hub :

Pour commencer, il faut se créer un compte sur le Docker Hub : rendez-vous sur <https://hub.docker.com/>

L'étape suivante est de se connecter au Docker Hub à partir de la ligne de commande **\$ docker login**. Il va vous demander, votre nom d'utilisateur et votre mot de passe, et si tout se passe bien vous devez avoir le message suivant : **Login Succeeded**

```
ubuntu@vps-c59c6930:~$ docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't
have a Docker ID, head over to https://hub.docker.com to create one.
Username: issaouib
Password:
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config
.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@vps-c59c6930:~$ 
```

Récupérer ensuite l'id ou le nom de votre image :

```
ubuntu@vps-c59c6930:~$ docker images
REPOSITORY      TAG          IMAGE ID       CREATED        SIZE
my_lamp         latest        della21a3af5   19 hours ago   501MB
alpine          latest        389fef711851   3 weeks ago    5.58MB
nginx           latest        ae2feff98a0c   3 weeks ago    133MB
debian          stable-slim  2bb074a4c289   4 weeks ago    69.2MB
ubuntu@vps-c59c6930:~$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Ensuite il faut rajouter un tag à l'id ou le nom de l'image récupérée. il faut que votre image soit nommée sous cette forme : **username/imagename:tag**

Il existe une commande pour ça, **\$ docker tag <IMAGENAME OU ID> <HUB-USER>/<REPONAME>[:<TAG>]**

Si vous relancez la commande docker images, vous verrez alors votre image avec le bon tag.

```
ubuntu@vps-c59c6930:~$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED       SIZE
my_lamp          latest        della21a3af5  19 hours ago  501MB
alpine           latest        389fef711851  3 weeks ago   5.58MB
nginx            latest        ae2feff98a0c  3 weeks ago   133MB
debian           stable-slim  2bb074a4c289  4 weeks ago   69.2MB
ubuntu@vps-c59c6930:~$ docker tag my_lamp issaouib/lamp:latest
ubuntu@vps-c59c6930:~$ docker images
REPOSITORY      TAG          IMAGE ID      CREATED       SIZE
issaouib/lamp   latest        della21a3af5  19 hours ago  501MB
my_lamp          latest        della21a3af5  19 hours ago  501MB
alpine           latest        389fef711851  3 weeks ago   5.58MB
nginx            latest        ae2feff98a0c  3 weeks ago   133MB
debian           stable-slim  2bb074a4c289  4 weeks ago   69.2MB
ubuntu@vps-c59c6930:~$ █
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Maintenant envoyez votre image vers Docker Hub grâce à la commande suivante : **\$ docker push <HUB-USER>/<REPONAME>[:<TAG>]**

```
ubuntu@vps-c59c6930:~$ docker images
REPOSITORY      TAG          IMAGE ID   CREATED        SIZE
issaouib/lamp  latest       de11a21a3af5  19 hours ago  501MB
my_lamp         latest       de11a21a3af5  19 hours ago  501MB
alpine          latest       389fef711851  3 weeks ago   5.58MB
nginx           latest       ae2feff98a0c  3 weeks ago   133MB
debian          stable-slim  2bb074a4c289  4 weeks ago   69.2MB
ubuntu@vps-c59c6930:~$ docker push issaouib/lamp:latest
The push refers to repository [docker.io/issaouib/lamp]
c2cd960934ee: Pushed
f295e1d8eab3: Pushed
a895509f783a: Pushed
9ce4784f287e: Pushed
8f8e2ea7b7f9: Pushed
daa624bd433e: Mounted from library/debian
latest: digest: sha256:d1940e9984edfd286246b7dcadced987def8ac78d45cc77e0e9620d63
9a59078 size: 1580
ubuntu@vps-c59c6930:~$ □
```

Et voilà, notre image est dans les nuages. Vous pouvez vous rendre sur votre Hub, et vous trouverez un nouveau repository.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

 docker hub Search for great content (e.g., mysql)

Explore Repositories Organizations Get Help issaouib ▾ 

Repositories issaouib / lamp Using 0 of 1 private repositories. [Get more](#)

General Tags Builds Timeline Collaborators Webhooks Settings

 issaouib / lamp Docker commands [Public View](#)

This repository does not have a description 

Last pushed: 4 minutes ago To push a new tag to this repository,

`docker push issaouib/lamp:tagname`

Tags and Scans  VULNERABILITY SCANNING - DISABLED [Enable](#)

This repository contains 1 tag(s).

TAG	OS	PULLED	PUSHED
latest		4 minutes ago	4 minutes ago

[See all](#)

Recent builds
Link a source provider and run a build to see build results here.

Readme  

Repository description is empty. Click [here](#) to edit.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Exemple3 : Amélioration de notre image LAMP

Dans l précédent, nous avions créé une image avec stack LAMP, malheureusement, c'est pas top niveau persistance de données car, lors d'un redémarrage du conteneur, nous allons rencontrer les deux problèmes suivants :

- Les données de notre base de données ne sont pas sauvegardées.
- Les modifications des sources de notre application ne seront pas appliquées.

Pour résoudre ce problème, nous allons utiliser les volumes !

- Commencez par télécharger le projet , https://gitlab.com/bileli/projet_dockerfile_volume.git. Dans ce projet, j'ai gardé le même Dockerfile, mais j'ai juste changé les fichiers sources en rajoutant un formulaire.
- buildez votre image : **\$ docker build -t my_lamp .**
- Concernant la base de données, nous allons créer un volume nommé "mysqldata" qui sera par la suite basé sur le dossier /var/lib/mysql du conteneur:
\$ docker volume create --name mysqldata
- Pour les sources de mon application, je vais faire les choses différemment. Je vais juste changer le dossier source du volume (les volumes nous donne cette possibilité). Lançons donc notre conteneur :
\$ docker run -d --name my_app -v \$PWD/app:/var/www/html -v mysqldata:/var/lib/mysql -p 8080:80 my_lamp

La commande \$PWD prendra automatiquement le chemin absolu du dossier courant, donc faites bien attention à exécuter votre image depuis le dossier du projet où mettez le chemin complet si vous souhaitez lancer votre commande depuis n'importe quelle autre chemin.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

51.83.72.127:8080/#

Ghazela Devops !! Ma propre image Docker Accueil Articles

Articles

Nouveau article

Titre *

Nom de l'auteur *

Contenu *

Envoyer

Liste d'articles

© Copyright: [MonAppDocker](#)



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Vous pouvez dès à présent modifier vos sources, depuis votre conteneur ou votre machine local et vos changements seront permanents et immédiatement traités par l'interpréteur php. Les données de votre base de données seront aussi automatiquement sauvegardées dans le volume mysqldata.

Avec les volumes, nous avons pu créer une image assez stable et exploitable dans un environnement de production. Vous l'avez sûrement remarqué mais notre commande docker run commence à devenir vraiment longue et nous n'avons pas encore résolu le problème qui est de séparer notre conteneur d'application web de notre conteneur de base de données. Et c'est pour ces raisons que dans le prochain chapitre, nous verrons le docker-compose.yml, c'est un fichier qui va nous permettre de définir le comportement de nos conteneurs et d'exécuter des applications Docker à conteneurs multiples.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

13- Docker-compose

Docker Compose est un outil permettant de définir le comportement de vos conteneurs et d'exécuter des applications Docker à conteneurs multiples. La config se fait à partir d'un fichier YAML, et ensuite, avec une seule commande, vous créez et démarrez tous vos conteneurs de votre configuration.

I- Installation du docker-compose

Docker Compose n'est pas installé par défaut et s'appuie sur le moteur Docker pour fonctionner. Voici la procédure à suivre pour télécharger Docker Compose sous un environnement Linux :

Pour être sûrs d'avoir la version stable la plus récente de Docker Compose, nous téléchargerons ce logiciel à partir de son dépôt officiel Github.
<https://github.com/docker/compose>

Tout d'abord, confirmez la dernière version disponible dans leur page des versions. Au moment de la rédaction du présent document, la version stable la plus récente est 1.27.4

La commande suivante permet de télécharger la version 1.27.4 et d'enregistrer le fichier exécutable à /usr/local/bin/docker-compose qui rendra ce logiciel accessible dans le monde entier sous la forme d'un docker-compose:

```
$ sudo curl -L "https://github.com/docker/compose/releases/download/1.27.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

Ensuite, définissez les autorisations correctes de sorte que le docker-compose soit exécutable :

```
$ sudo chmod +x /usr/local/bin/docker-compose
```

Pour vérifier que l'installation a réussi, vous pouvez lancer :

```
$ docker-compose --version
```

Vous verrez une sortie semblable à celle-ci : **docker-compose version 1.27.4, build 40524192**



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

II- Définition des besoins du Docker Compose et amélioration du Dockerfile

Le but de cet tuto est d'améliorer notre ancienne application LAMP. Par la suite nous allons séparer le conteneur de notre application web par rapport au conteneur de notre base de données.

Au préalable, commencez par télécharger les sources du projet https://gitlab.com/bileli/projet_dockecompose.git

II-1 : Amélioration du Dockerfile

Profitons de cet exemple pour améliorer le Dockerfile de notre stack LAMP en réduisant son nombre d'instructions. Pour cela, on se basera sur une nouvelle image.

Conseil : Si vous souhaitez conteneuriser une application assez connue, alors je vous conseille de toujours fouiller dans le Hub Docker, afin de savoir si une image officielle de l'application existe déjà.

En cherchant dans le Hub Docker, j'ai pu dénicher les images adéquates, notamment :

- Une image officielle php avec le tag 7-apache : https://hub.docker.com/_/php
- Une image officielle mysql : https://hub.docker.com/_/mysql

Une fois que j'ai trouvé les bonnes images, je peux alors m'attaquer à la modification du Dockerfile.

Pour le moment, nous utiliserons ce Dockerfile seulement pour construire une image avec une couche OS, Apache et Php sans implémenter aucun service de base de données. Cette image se basera sur l'image officielle php avec le tag 7-apache qui vient déjà avec un OS (distribution Debian). Concernant l'image mysql nous l'utiliserons plus tard dans notre docker-compose.yml.

Dans le même dossier que vous avez téléchargé, créez un fichier Dockerfile et mettez dedans le contenu suivant :



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

```
FROM php:7-apache
LABEL version="1.0" maintainer="Issaoui Bilel <issaouios2015@hotmail.com>"
# Activation des modules php
RUN docker-php-ext-install pdo pdo_mysql
WORKDIR /var/www/html
```

Buildez ensuite votre image avec la commande suivante : **\$ docker build -t myapp .**

II-2 : Les besoins pour notre Docker Compose

Avant de créer notre fichier docker-compose.yml, il faut auparavant définir les comportements de nos conteneurs.

a- Nos besoins pour le conteneur de la base de données

On va débuter par la récolte des besoins du conteneur de la base de données. Pour celle-ci, il nous faudra :

- Un fichier sql pour créer l'architecture de notre base de données.
- Un volume pour stocker les données.

Avant de foncer tête baissée dans la création/modification de notre fichier sql, il toujours important de vérifier avant ce que nous propose la page Docker Hub de l'image mysql. En lisant sa description, les informations qui m'ont le plus captivé sont ses variables d'environnements qu'on peut surcharger, notamment :

MYSQL_ROOT_PASSWORD: spécifie le mot de passe qui sera défini pour le compte MySQL root (c'est une variable obligatoire).

MYSQL_DATABASE: spécifie le nom de la base de données à créer au démarrage de l'image.

MYSQL_USER et MYSQL_PASSWORD : utilisées conjointement pour créer un nouvel utilisateur avec son mot de passe. Cet utilisateur se verra accorder des autorisations de super-utilisateur pour la base de données **MYSQL_DATABASE**

Ces variables d'environnements vont nous aider à créer une partie de l'architecture de notre base de données.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Dans la description de l'image mysql, il existe une autre information très utile. Lorsqu'un conteneur mysql est démarré, il exécutera des fichiers avec des extensions .sh, .sql et .sql.gz qui se trouvent dans /docker-entrypoint-initdb.d. Nous allons profiter de cette information pour déposer le fichier articles.sql (disponible dans les sources téléchargées) dans le dossier /docker-entrypoint-initdb.d afin de créer automatiquement notre table SQL.

b- Nos besoins pour le conteneur de l'application web

Concernant le conteneur de l'application web, nous aurons besoin de :

Une communication avec le conteneur de la base de données.

Un volume pour stocker les sources de l'application web.

Me concernant la seule information utile dans la description de la page Docker Hub de l'image php, est qu'il est possible d'installer et d'activer les modules php dans le conteneur php avec la commande docker-php-ext-install (C'est la commande utilisée dans notre Dockerfile afin d'activer le module pdo et pdo_mysql).

c- Lancer les conteneurs sans le docker-compose

Histoire de vous donner une idée sur la longueur de la commande **docker run** sans utiliser le fichier docker-compose.yml. Je vais alors l'utiliser pour démarrer les différents conteneurs de notre application.

Premièrement je vais vous dévoiler, deux nouvelles options de la commande **docker run** :

-e : définit/surcharge des variables d'environnement

--link : ajoute un lien à un autre conteneur afin de les faire communiquer entre eux.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Voici à quoi va ressembler la commande docker run pour la création du conteneur de la base de données :

```
docker run -d -e MYSQL_ROOT_PASSWORD='test' \
-e MYSQL_DATABASE='test' \
-e MYSQL_USER='test' \
-e MYSQL_PASSWORD='test' \
--volume db-volume:/var/lib/mysql \
--volume $PWD/articles.sql:/docker-entrypoint-initdb.d/articles.sql \
--name mysql_c mysql:5.7
```

Voici à quoi va ressembler la commande docker run pour la création du conteneur de l'application web :

```
docker run -d --volume $PWD/app:/var/www/html -p 8080:80 --link mysql_c --name myapp_c myapp
```

Dans cet exemple, on peut vite remarquer que les commandes **docker run** sont assez longues et par conséquent elles ne sont pas assez lisibles. De plus, vous aurez à lancer cette commande pour chaque nouveau démarrage de l'application. Mais vous aurez aussi à gérer vos différents conteneurs séparément. C'est pour ces raisons, que nous utiliserons le fichier `docker-compose.yml` afin de centraliser la gestion de nos multiples conteneurs d'une application Docker depuis un seul fichier. Dans notre cas il va nous permettre d'exécuter et définir les services, les volumes et la mise en relation des différents conteneurs de notre application.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

II-3 : Création du docker-compose

a- Contenu du docker-compose

Commencez d'abord par créer un fichier et nommez le docker-compose.yml, ensuite copiez collez le contenu ci-dessous. Par la suite, plus bas dans l'article, je vais vous fournir les explications des différentes lignes de ce fichier :

```
version: '3.8'
services:
  db:
    image: mysql:5.7
    container_name: mysql_c
    restart: always
    volumes:
      - db-volume:/var/lib/mysql
      - ./articles.sql:/docker-entrypoint-initdb.d/articles.sql
    environment:
      MYSQL_ROOT_PASSWORD: test
      MYSQL_DATABASE: test
      MYSQL_USER: test
      MYSQL_PASSWORD: test
  app:
    image: myapp
    container_name: myapp_c
    restart: always
    volumes:
      - ./app:/var/www/html
    ports:
      - 8080:80
    depends_on:
      - db
  volumes:
    db-volume:
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

b- Explication du fichier docker-compose.yml

```
version: '3.8'
```

Il existe plusieurs versions rétrocompatibles pour le format du fichier Compose (voici la liste des versions de Docker Compose selon la version moteur Docker <https://docs.docker.com/compose/compose-file/>). Dans mon cas je suis sous la version 20.10.1 du moteur Docker, donc j'utilise la version 3.8

```
services:
```

Dans une application Docker distribuée, différentes parties de l'application sont appelées services. Les services ne sont en réalité que des conteneurs. Dans notre cas nous aurons besoin d'un service pour notre base de données et un autre pour notre application web.

```
db:
  image: mysql:5.7
  container_name: mysql_c
  restart: always
  volumes:
    - db-volume:/var/lib/mysql
    - ./articles.sql:/docker-entrypoint-initdb.d/articles.sql
  environment:
    MYSQL_ROOT_PASSWORD: test
    MYSQL_DATABASE: test
    MYSQL_USER: test
    MYSQL_PASSWORD: test
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Dans cette partie, on crée un service nommé db. Ce service indique au moteur Docker de procéder comme suit :

- Se baser sur l'image **mysql:5.7**
- Nommer le conteneur **mysql_c**
- Le **restart: always** démarrera automatiquement le conteneur en cas de redémarrage du serveur
- Définir les volumes à créer et utiliser (un volume pour exécuter automatiquement notre fichier sql et un autre pour sauvegarder les données de la base de données)
- Surcharger les variables d'environnements à utiliser



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

```
app:  
  image: myapp  
  container_name: myapp_c  
  restart: always  
  volumes:  
    - ./app:/var/www/html  
  ports:  
    - 8080:80  
  depends_on:  
    - db
```

Ici, on crée un service nommé app. Ce service indique au moteur Docker de procéder comme suit :

- Se baser sur l'image nommée **myapp** qu'on avait construit depuis notre Dockerfile
- Nommer le conteneur **myapp_c**
- Le **restart: always** démarrera automatiquement le conteneur en cas de redémarrage du serveur
- Définir les volumes à créer et à utiliser pour sauvegarder les sources de notre application
- Mapper le port 8080 sur le port 80
- Le **depends_on** indique les dépendances du service **app**. Ces dépendances vont provoquer les comportements suivants :
 - * Les services démarrent en ordre de dépendance. Dans notre cas, le service db est démarré avant le service app
 - * Les services s'arrêtent selon l'ordre de dépendance. Dans notre cas, le service app est arrêté avant le service db



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

```
volumes:  
  db-volume:
```

Enfin, je demande au moteur Docker de me créer un volume nommé **db-volume**, c'est le volume pour stocker les données de notre base de données.

c- Lancer l'application depuis docker-compose.yml

Pour être sur le même pied d'estale, voici à quoi doit ressembler votre arborescence :

```
.  
└── Dockerfile  
└── app  
    ├── db-config.php  
    ├── index.php  
    └── validation.php  
└── articles.sql  
└── docker-compose.yaml  
  
1 directory, 6 files
```

Placez vous au niveau du dossier qui contient le fichier docker-compose.yml. Ensuite lancez la commande suivante pour exécuter les services du docker-compose.yml : `docker-compose up -d`

Ici l'option **-d** permet d'exécuter les conteneurs du Docker compose en arrière-plan.

Si vous le souhaitez, vous pouvez vérifier le démarrage des conteneurs issus du docker-compose.yml :

```
docker ps
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

```
Creating mysql_c ... done
Creating myapp_c ... done
ubuntu@vps-c59c6930:~/projet_dockecompose$ docker ps
CONTAINER ID   IMAGE       COMMAND           CREATED          STATUS          PORTS          NAMES
37008a29ae44   myapp      "docker-php-entrypoi..."   About a minute ago   Up About a minute   0.0.0.0:8080->80/tcp   myapp_c
b6fe93e878b7   mysql:5.7  "docker-entrypoint.s..."   About a minute ago   Up About a minute   3306/tcp, 33060/tcp   mysql_c
ubuntu@vps-c59c6930:~/projet_dockecompose$
```

Pour seulement lister les conteneurs du docker-compose.yml, il suffit d'exécuter la commande suivante :

```
ubuntu@vps-c59c6930:~/projet_dockecompose$ docker-compose ps
      Name            Command           State          Ports
-----
myapp_c      docker-php-entrypoint apac ...    Up      0.0.0.0:8080->80/tcp
mysql_c      docker-entrypoint.sh mysqld      Up      3306/tcp, 33060/tcp
ubuntu@vps-c59c6930:~/projet_dockecompose$
```

Si jamais vos conteneurs ne sont pas dans l'état UP, alors vérifiez les logs des services de votre Docker Compose en tapant la commande suivante :

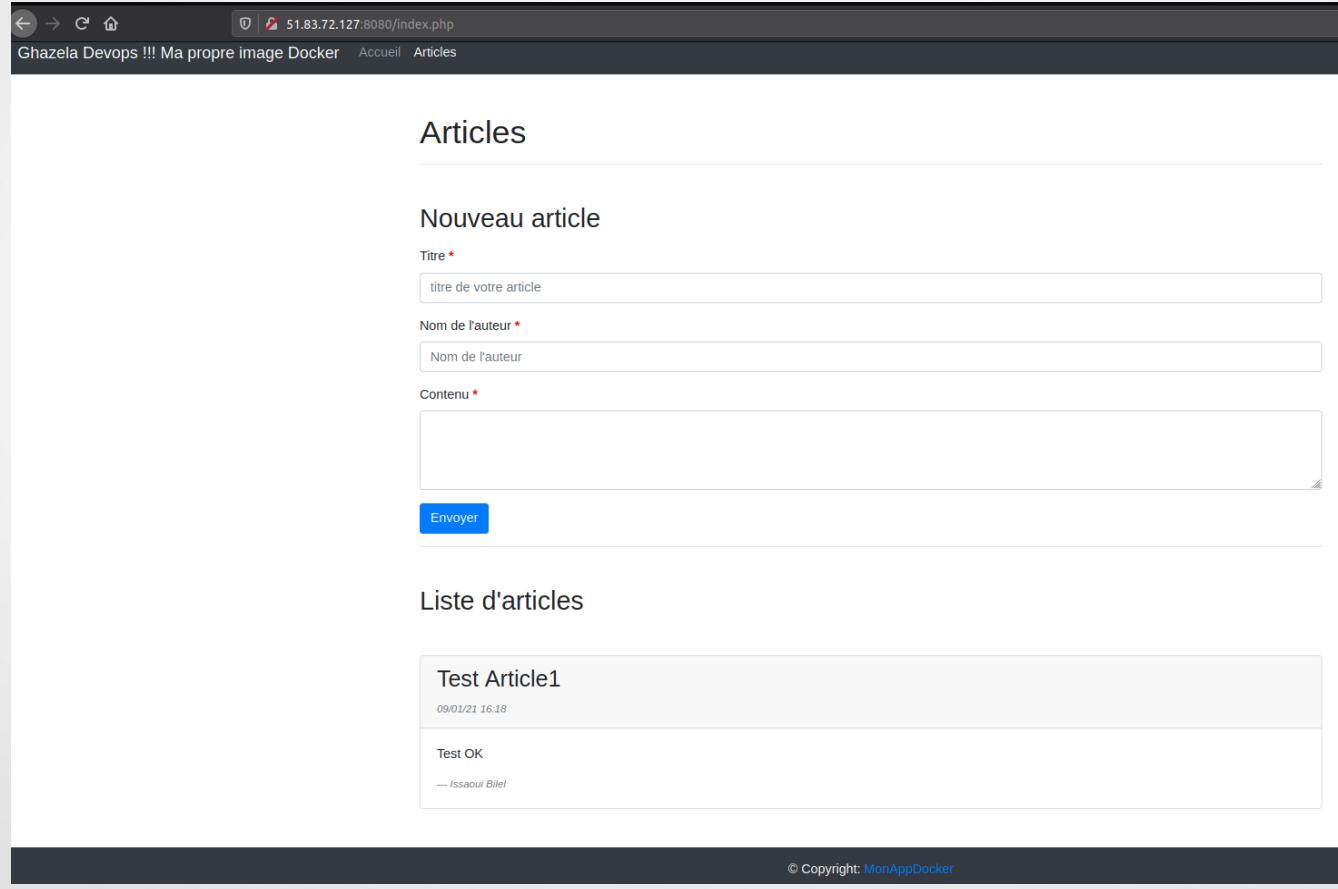
```
docker-compose logs
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Si tout c'est bien passé, alors visitez la page suivante http://@ip_serveur:8080/, et vous obtiendrez le résultat suivant :



The screenshot shows a web application running on a Docker container. The URL in the address bar is `51.83.72.127:8080/index.php`. The page title is "Ghazela Devops !!! Ma propre image Docker". The main content area has a green header titled "Articles". Below it, there's a form titled "Nouveau article" with fields for "Titre" (with placeholder "titre de votre article"), "Nom de l'auteur" (placeholder "Nom de l'auteur"), and "Contenu" (a large text area). A blue "Envoyer" button is at the bottom of the form. Below the form, there's a section titled "Liste d'articles" containing a single item: "Test Article1" from "09/01/21 16:18" by "Test OK" (signature "— Issaoui Bilel"). At the bottom of the page, a dark footer bar contains the copyright notice "© Copyright: MonAppDocker".



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Remplissez le formulaire de l'application, et tuez les conteneurs du docker-compose.yml, avec la commande suivante : `docker-compose kill`

Relancez ensuite vos services, et vous verrez que vos données sont bel et bien sauvegardées.

```
ubuntu@vps-c59c6930:~/projet_dockecompose$ docker-compose kill
Killing myapp_c ... done
Killing mysql_c ... done
ubuntu@vps-c59c6930:~/projet_dockecompose$ docker-compose ps
      Name          Command         State    Ports
----- 
myapp_c    docker-php-entrypoint apac ...   Exit 137
mysql_c    docker-entrypoint.sh mysqld       Exit 137
ubuntu@vps-c59c6930:~/projet_dockecompose$ docker-compose up -d
Starting mysql_c ... done
Starting myapp_c ... done
ubuntu@vps-c59c6930:~/projet_dockecompose$ docker-compose ps
      Name          Command         State    Ports
----- 
myapp_c    docker-php-entrypoint apac ...   Up        0.0.0.0:8080->80/tcp
mysql_c    docker-entrypoint.sh mysqld       Up        3306/tcp, 33060/tcp
ubuntu@vps-c59c6930:~/projet_dockecompose$
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

d- Détails de la communication inter-conteneurs dans les sources de l'application

Si vous êtes dans un répertoire qui s'appelle `projet_dockecompose`, docker compose va créer un réseau qui s'appelle `projet_dockecompose` de type **Bridge**. Et puis il va créer les containers indiqués dans le fichier, et les ajouter à ce réseau,

Vous pouvez visualiser les réseaux docker en faisant : `$ docker network ls`

```
ubuntu@vps-c59c6930:~/projet_dockecompose$ ls
Dockerfile  app  articles.sql  docker-compose.yaml
ubuntu@vps-c59c6930:~/projet_dockecompose$ docker-compose ps
      Name            Command           State    Ports
-----
myapp_c    docker-php-entrypoint apac ...    Up      0.0.0.0:8080->80/tcp
mysql_c    docker-entrypoint.sh mysqld    Up      3306/tcp, 33060/tcp
ubuntu@vps-c59c6930:~/projet_dockecompose$ docker network ls
NETWORK ID     NAME          DRIVER      SCOPE
a7791825e989   bridge        bridge      local
2283d90bf12e   host          host       local
4f0c4ec0018e   none          null       local
717533602b66   projet_dockecompose_default bridge      local
ubuntu@vps-c59c6930:~/projet_dockecompose$ 
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Donc les conteneurs possèdent par défaut une adresse ip. Vous pouvez récolter cette information grâce à la commande suivante :

```
ubuntu@vps-c59c6930:~/projet_dockecompose$ docker network inspect projet_dockecompose_default
[
    {
        "Name": "projet_dockecompose_default",
        "Id": "717533602b66f29dcc717f1f8a8c69e46ef97b78347fa3878c1e403d9e369983",
        "Created": "2021-01-09T16:11:10.880744967Z",
        "Scope": "local",
        "Driver": "bridge",
        "EnableIPv6": false,
        "IPAM": {
            "Driver": "default",
            "Options": null,
            "Config": [
                {
                    "Subnet": "172.19.0.0/16",
                    "Gateway": "172.19.0.1"
                }
            ]
        },
        "Internal": false,
        "Attachable": true,
        "Ingress": false,
        "ConfigFrom": {
            "Network": ""
        },
        "ConfigOnly": false,
        "Containers": {
            "37008a29ae44cb61c6ee2d9df0497b8d6d2900bb8d93c347cecea7a1f29d9903": {
                "Name": "myapp_c",
                "EndpointID": "9f6159194cd3cf6d6241354e255b7b41b4f34eb0afb05a27c983af98810f97f",
                "MacAddress": "02:42:ac:13:00:03",
                "IPv4Address": "172.19.0.3/16",
                "IPv6Address": ""
            },
            "b6fe93e878b79fd528a92c8cdc08599ab2bd3d64a6041d910f1bdbf904c58050": {
                "Name": "mysql_c",
                "EndpointID": "ecb92d3b427920613f28bc6b9600466247c1f310da2817447f16c8f5e7e4d20f",
                "MacAddress": "02:42:ac:13:00:02",
                "IPv4Address": "172.19.0.2/16",
                "IPv6Address": ""
            }
        }
    }
]
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Ou par la commande :

```
docker inspect -f '{{.Name}} - {{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps -aq)
```

```
ubuntu@vps-c59c6930:~/projet_dockecompose$ docker inspect -f '{{.Name}} - {{range .NetworkSettings.Networks}}{{.IPAddress}}{{end}}' $(docker ps -aq)
/myapp_c - 172.19.0.3
/mysql_c - 172.19.0.2
ubuntu@vps-c59c6930:~/projet_dockecompose$
```

Pour faire communiquer notre application web avec la base de données, on peut utiliser dans le conteneur de l'app web soit l'ip, le nom du service (ici db) ou le nom du conteneur (ici mysql_c) de la base de données.

Si vous ouvrez le fichier db-config.php dans le dossier app, alors vous verrez la ligne suivante : `const DB_DSN = 'mysql:host=mysql_c;dbname=test';`

Dans ces cas, j'ai utilisé le nom du conteneur de la base de données pour communiquer avec ce dernier.

e- Conclusion

Je pense que vous l'aurez compris, le Docker Compose est un outil permettant de faciliter la gestion des applications Docker à conteneurs multiples, comme :

- Démarrer, arrêter et reconstruire des services
- Afficher le statut des services en cours d'exécution
- Diffuser la sortie des logs des services en cours d'exécution
- Exécuter une commande unique sur un service
- etc ...



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

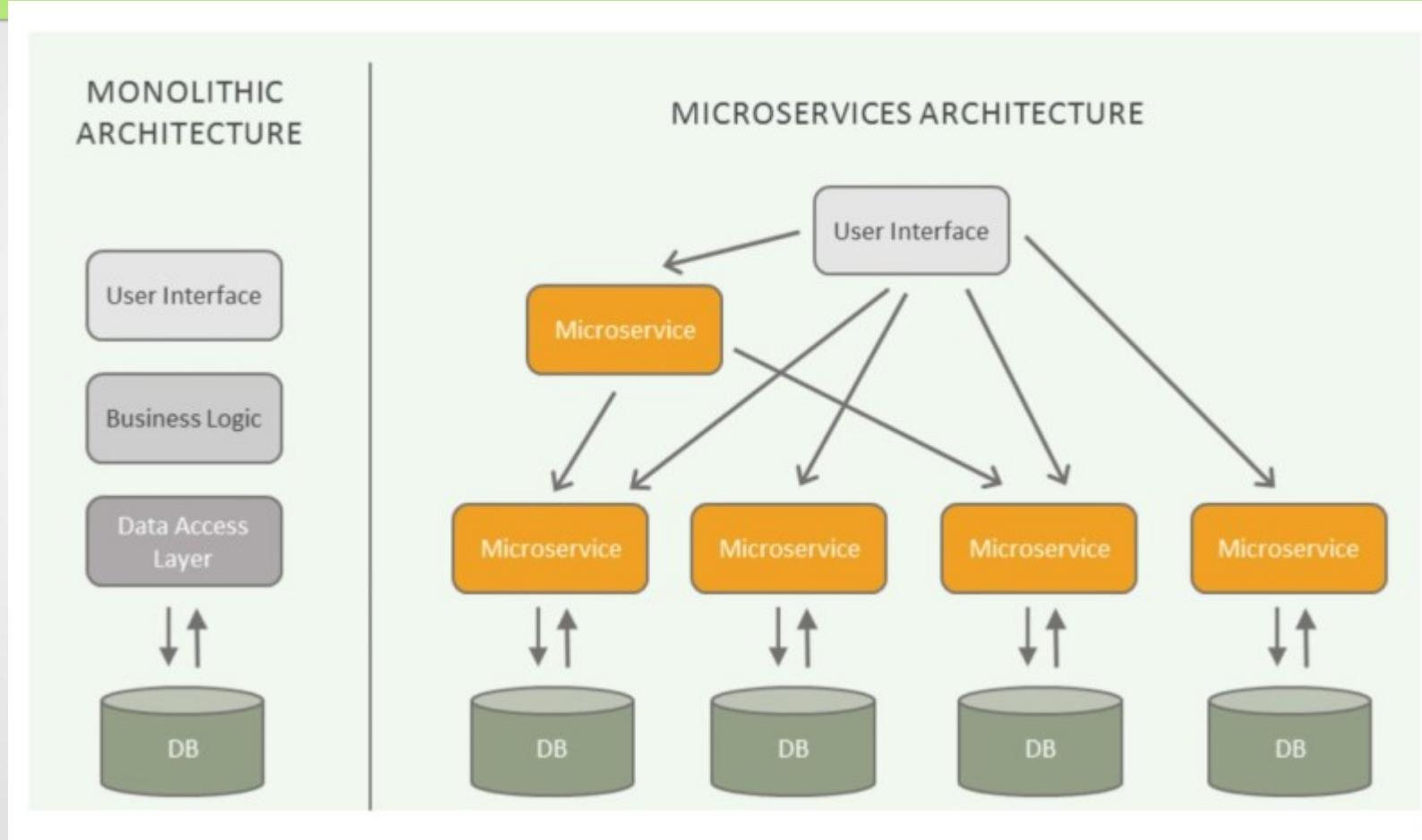
Comme pour chaque fin de chapitre, je vous liste ci-dessous un récapitulatif de quelques commandes intéressantes du Docker Compose:

```
## Exécuter les services du docker-compose.yml
docker-compose up
    -d : Exécuter les conteneurs en arrière-plan
## Lister des conteneurs du Docker Compose
docker-compose ls
    -a ou --all : afficher aussi les conteneurs stoppés
## Sorties/erreurs des conteneurs du Docker Compose
docker-compose logs
    -f : suivre en permanence les logs du conteneur
    -t : afficher la date et l'heure de la réception de la ligne de log
    --tail=<NOMBRE DE LIGNE> = nombre de lignes à afficher à partir de la fin pour chaque conteneur.
## Tuer les conteneurs du Docker Compose
docker-compose kill
## Stopper les conteneurs du Docker Compose
docker-compose stop
    -t ou --timeout : spécifier un timeout en seconde avant le stop (par défaut : 10s)
## Démarrer les conteneurs du Docker Compose
docker-compose start
## Arrêtez les conteneurs et supprimer les conteneurs, réseaux, volumes, et les images
docker-compose down
    -t ou --timeout : spécifier un timeout en seconde avant la suppression (par défaut : 10s)
## Supprimer des conteneurs stoppés du Docker Compose
docker-compose rm
    -f ou --force : forcer la suppression
## Lister les images utilisées dans le docker-compose.yml
docker-compose images
```



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Points forts de l'architecture microservice

Composants indépendants. Premièrement, tous les services peuvent être déployés et mis à jour indépendamment, ce qui donne plus de flexibilité. Deuxièmement, un bogue dans un microservice a un impact uniquement sur un service particulier et n'influence pas l'ensemble de l'application. En outre, il est beaucoup plus facile d'ajouter de nouvelles fonctionnalités à une application de microservice qu'à une application monolithique.

Compréhension plus facile. Divisée en composants plus petits et plus simples, une application de microservice est plus facile à comprendre et à gérer. Vous vous concentrez simplement sur un service spécifique lié à un objectif commercial que vous avez.

Meilleure évolutivité. Un autre avantage de l'approche des microservices est que chaque élément peut être mis à l'échelle indépendamment. Ainsi, l'ensemble du processus est plus rentable et plus rapide qu'avec les monolithes lorsque l'ensemble de l'application doit être mis à l'échelle même si cela n'est pas nécessaire. De plus, chaque monolithe a des limites en termes d'évolutivité, donc plus vous acquérez d'utilisateurs, plus vous rencontrez de problèmes avec votre monolithe. Par conséquent, de nombreuses entreprises finissent par reconstruire leurs architectures monolithiques.

Flexibilité dans le choix de la technologie. Les équipes d'ingénieurs ne sont pas limitées par la technologie choisie dès le départ. Ils sont libres d'appliquer diverses technologies et cadres pour chaque microservice.

Le plus haut niveau d'agilité. Toute panne dans une application de microservices affecte uniquement un service particulier et non la solution entière. Ainsi, tous les changements et expériences sont mis en œuvre avec moins de risques et moins d'erreurs.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker

Faiblesses de l'architecture des microservices

Complexité supplémentaire. Étant donné qu'une architecture de microservices est un système distribué, vous devez choisir et configurer les connexions entre tous les modules et bases de données. Aussi, tant qu'une telle application comprend des services indépendants, tous doivent être déployés indépendamment.

Distribution du système. Une architecture de microservices est un système complexe de plusieurs modules et bases de données, toutes les connexions doivent donc être traitées avec soin.

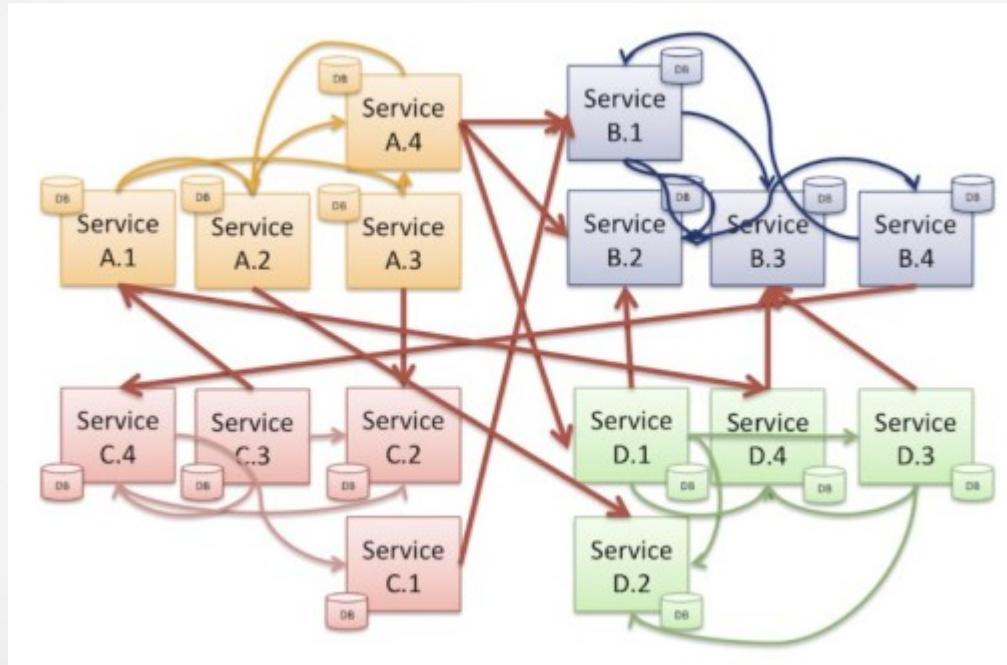
Préoccupations transversales. Lors de la création d'une application de microservices, vous devrez faire face à un certain nombre de préoccupations transversales. Ils incluent la configuration externalisée, la journalisation, les métriques, les contrôles de santé et autres.

Teste. Une multitude de composants déployables indépendamment rend le test d'une solution basée sur des microservices beaucoup plus difficile.



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155

Docker



Mr. BILEL Issaoui, Formateur DevOps chez Ghazela Technology Academy, www.ghazelatc.com. Tél. +21654260000. Le support est à usage personnel, il est propriété intellectuel de l'académie, il n'est pas à usage commercial, contact@ghazelatc.com, +21671866142, +21654828018, +21627862155