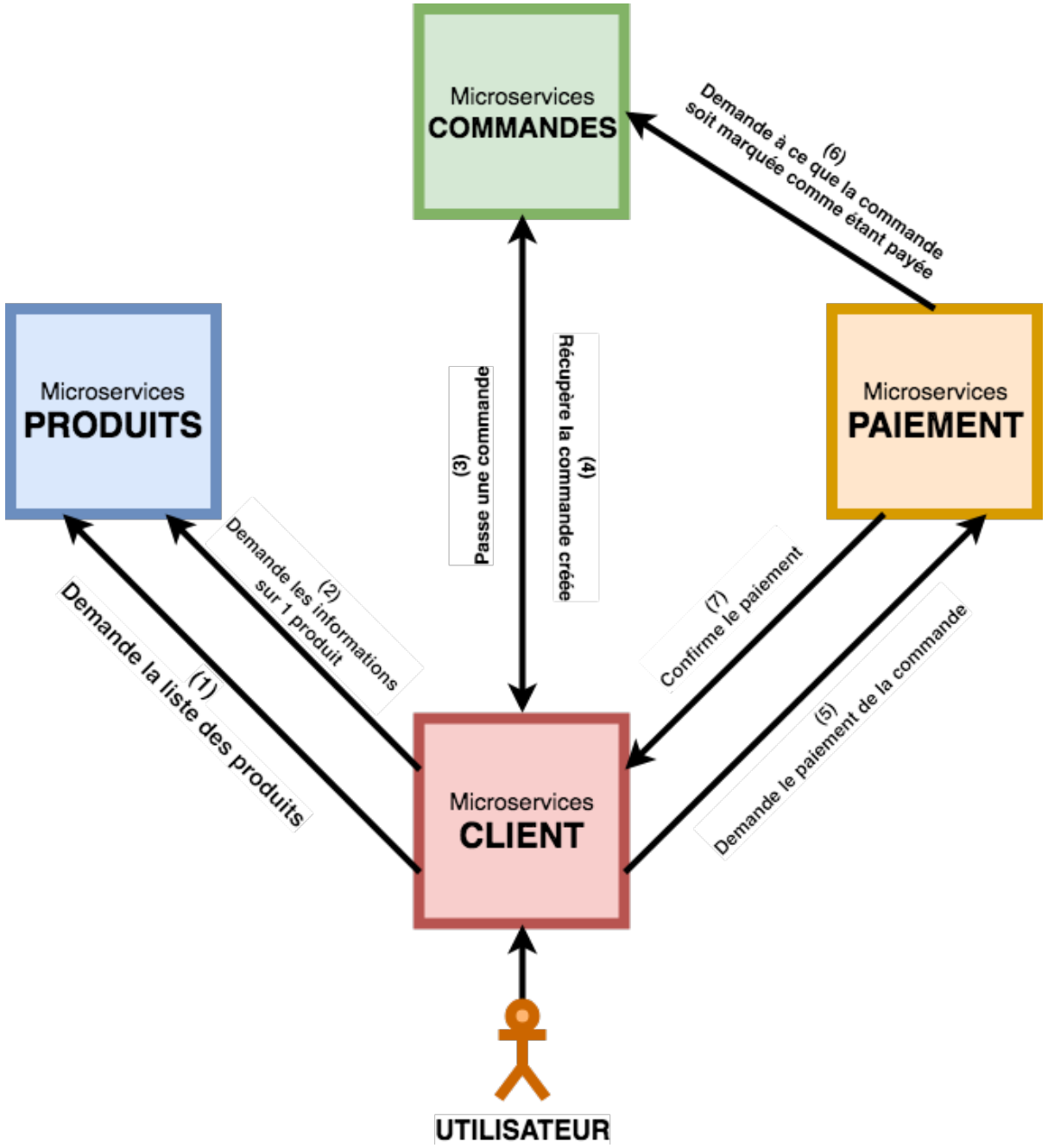
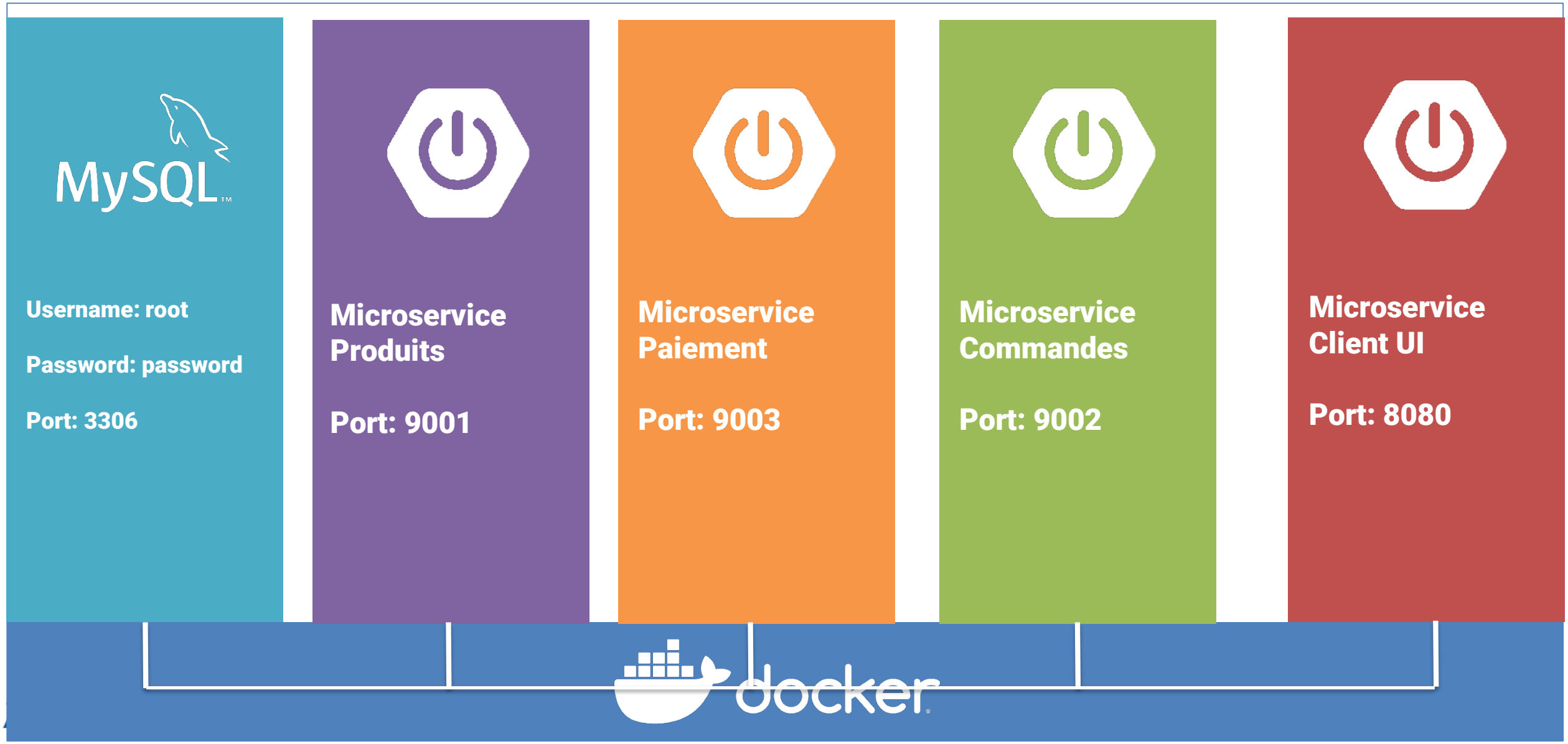
ECOLE NATIONALE
DES SCIENCES
APPLIQUEES
TANGER

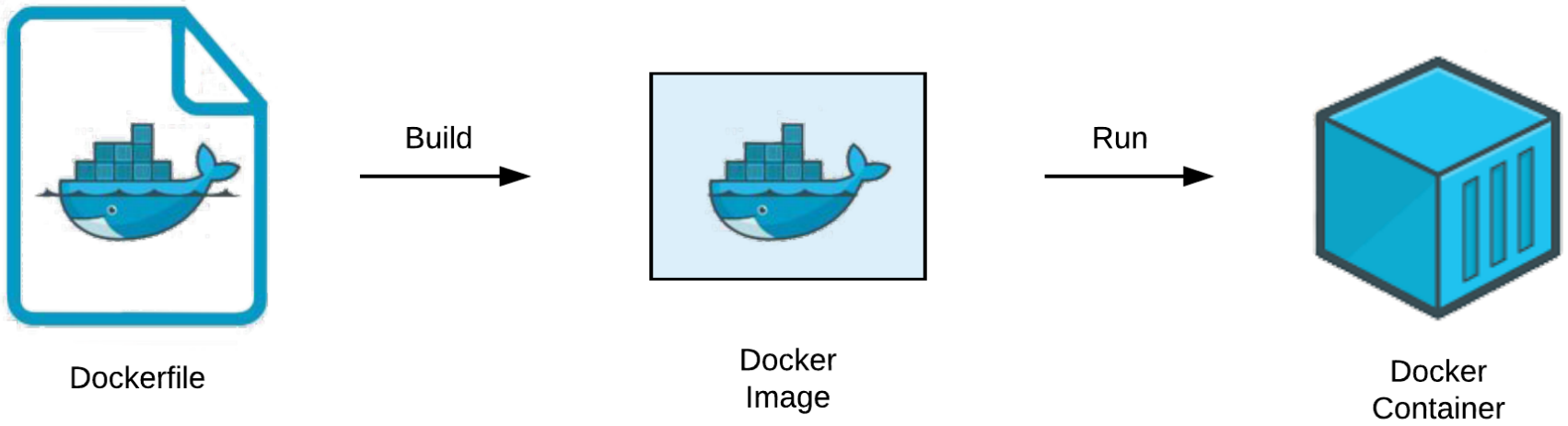
MICROSERVICES
&
DOCKER


Microservices
COMMANDES
Microservices
PRODUITS
Microservices
PAIEMENT
Microservices
CLIENT
UTILISATEUR
(1)
Demande la liste des produits
(2)
Demande les informations
sur 1 produit
(3)
Passe une commande
(4)
Récupère la commande créée
(5)
Demande le paiement de la commande
(6)
Demande à ce que la commande
soit marquée comme étant payée
(7)
Confirme le paiement


MySQL
Username: root
Password: password
Port: 3306
Microservice Produits
Port: 9001
Microservice Paiement
Port: 9003
Microservice Commandes
Port: 9002
Microservice Client UI
Port: 8080
docker

$ docker run -p 3306:3306 --net=my-network --name=mydatabase -e MYSQL_ROOT_PASSWORD=password -d mysql:5.7

DOCKERFILE



DOCKERFILE

```
FROM openjdk:8-jdk-alpine

VOLUME /tmp

ARG JAR_FILE=target/*.jar

COPY ${JAR_FILE} app.jar

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]
```

```
$ mvn package -DskipTests

# create a  network
$ docker network create my-network
```

Pour créer l'image, vous pouvez utiliser la ligne de commande Docker. Par exemple:

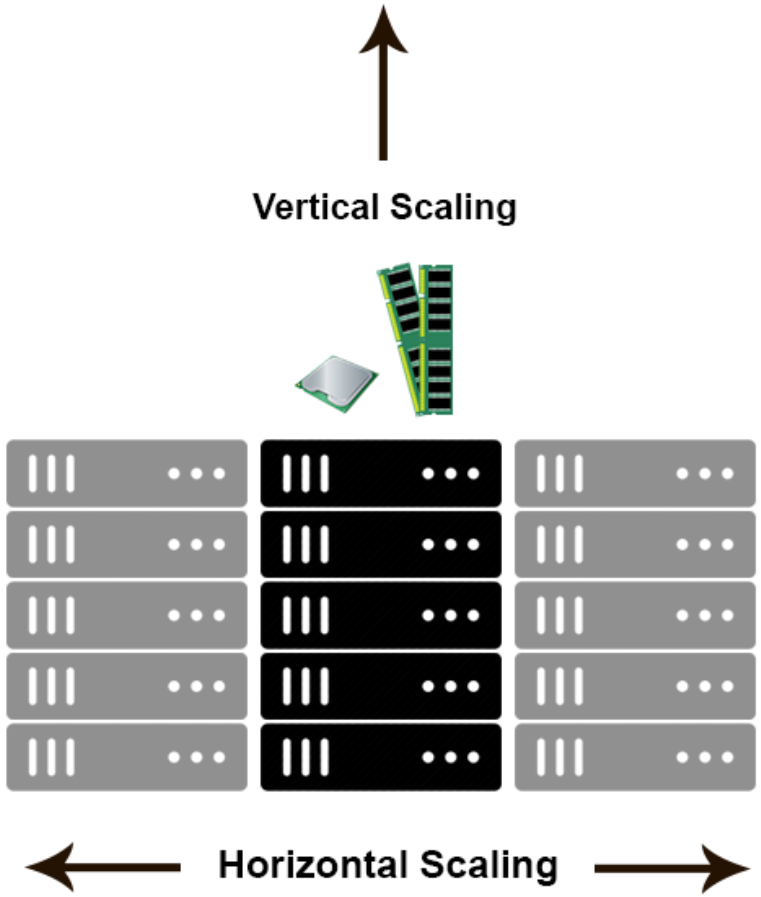
```
$ docker build -t ecommerce/produits .
```

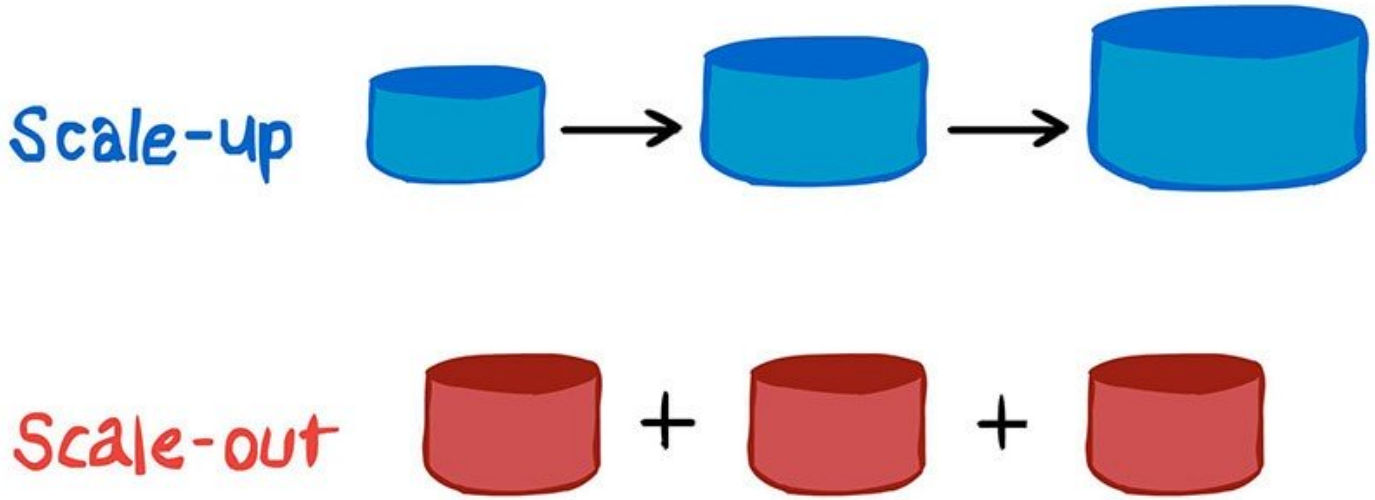
pour exécuter une image Docker qui a été créée localement, vous pouvez l'exécuter comme ceci:

```
$ docker run -d --net=my-network --name=produits -p 9001:9001 -t ecommerce/produits
```

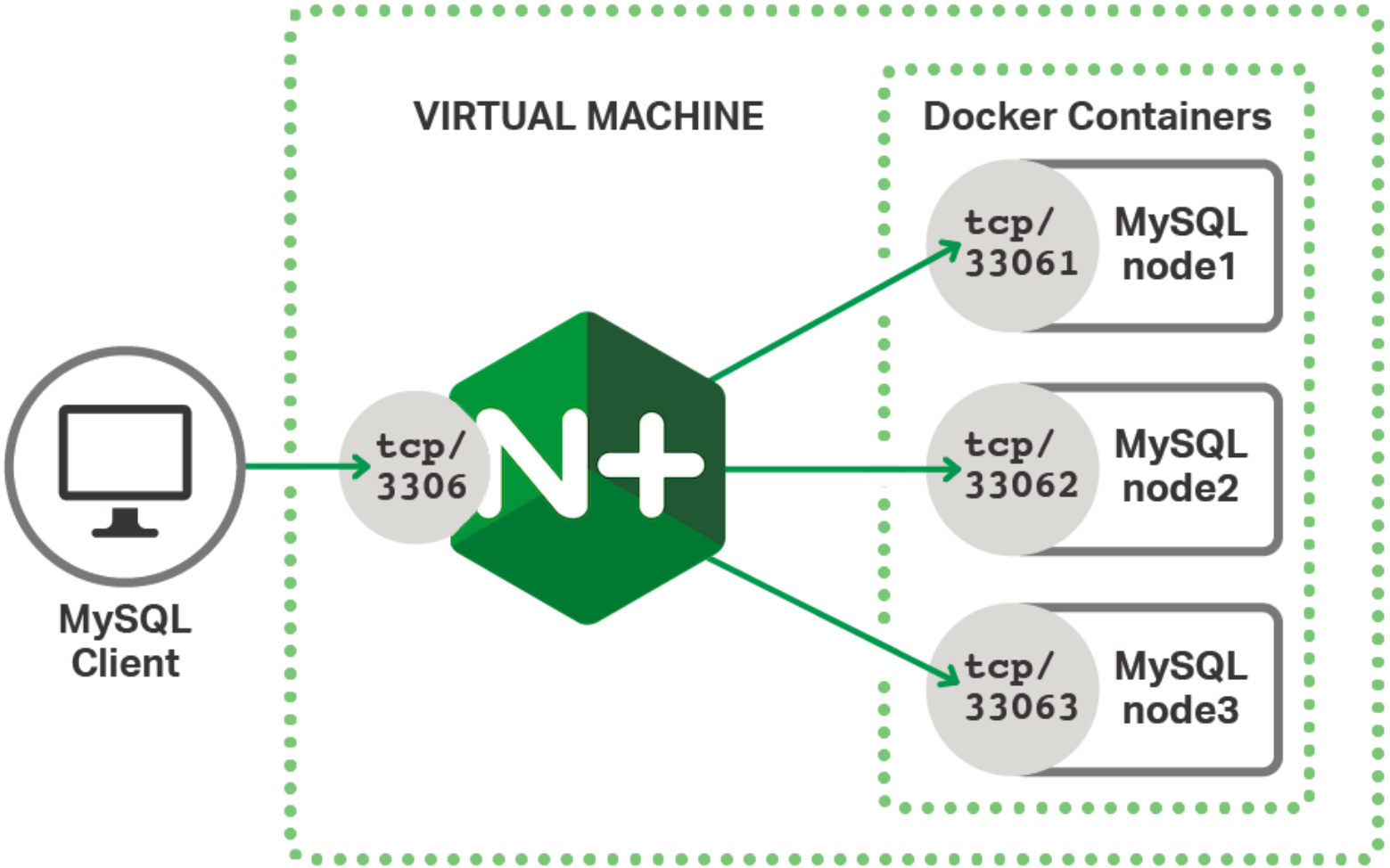
L'application est ensuite disponible sur http://localhost:9001

Difference between scaling horizontally and vertically





Scale-Out vs Scale-Up Storage


VIRTUAL MACHINE
Docker Containers
tcp/
33061
MySQL
node1
tcp/
3306
tcp/
33062
MySQL
node2
MySQL
Client
tcp/
33063
MySQL
node3

limit cpu

```
$ docker container run -it --cpus="0.2" -p 9001:9001 --name testcpu ecommerce/produits
```

limit I/O

```
$ docker container run -it --device-write-bps /dev/sda:1mb -p 9001:9001 --name testio ecommerce/produits
```

limit Memory

```
$ docker container run -d --memory=180m --memory-swap=180m -p 9001:9001 --name testram  springio/gs-spring-boot-docker
```

https://docs.docker.com/v17.09/engine/admin/resource_constraints/

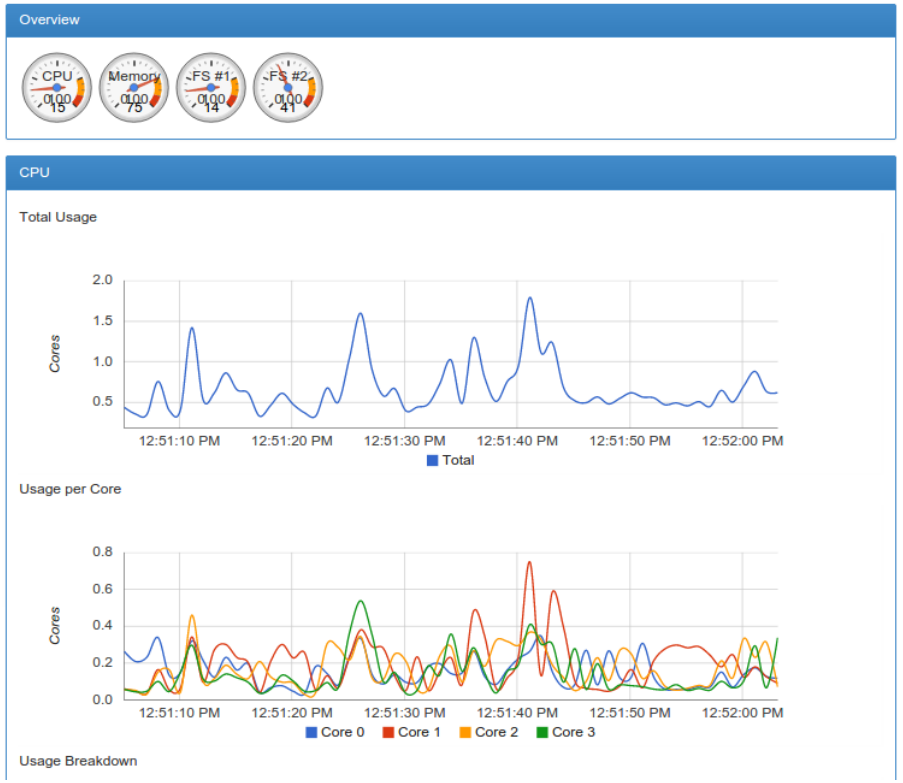
cAdvisor (Container Advisor) permet aux utilisateurs de conteneurs de comprendre l'utilisation des ressources et les caractéristiques de performance de leurs conteneurs en cours d'exécution. Il s'agit d'un démon en cours d'exécution qui collecte, agrège, traite et exporte des informations sur l'exécution des conteneurs

cAdvisor

```
$ docker run -d --name=cadvisor -p 8888:8888 --volume=/var/run:/var/run:rw --volume=/sys:/sys:ro --volume=/var/lib/docker/:/var/lib/docker:ro google/cadvisor:latest
```

Or vous pouvez utilisez:

```
$ docker stats
```



ECOLE NATIONALE DES SCIENCES APPLIQUEES TANGER

“From Gmail to YouTube to Search, everything at Google runs in containers. Containerization allows our development teams to move fast, deploy software efficiently, and operate at an unprecedented scale. Each week, we start over several billion containers” https://cloud.google.com/containers/

C'est une autre histoire!

D’un système open source qui vise à fournir une « plate-forme permettant d'automatiser le déploiement, la montée en charge et la mise en œuvre de conteneurs d'application sur des clusters de serveurs

MERCI POUR VOTRE ATTENTION

ENSA Tanger
ECOLE NATIONALE
DES SCIENCES
APPLIQUEES
TANGER