Cloud Computing Systems session 1: Using Docker Compose

Novembre 06,2017

Instructions:

1. Make teams of two students to solve the TP
2. Listen the brief introduction
3. Answer the questions, write a report and code. ☺
4. Send me **the report and the project of the Docker Compose part** by **email** at the end of the TP. My address is z.li@catie.fr
5. The TP lasts 4 hours. If you finish before, let me know.
6. You are allowed to ask questions if you don't understand the examples.
7. Get the code source from the url: https://gitlab.com/Aurore0319/TP1-DockerCompose

Before Docker Compose:

Basic Concepts:

We will just review several basic concepts of docker. Can you answer these questions? (Try not to look up on the internet).

a) What can you use Docker for?
b) What is Docker Image?
c) Can you explain the differences between a VM and a Container?
d) What is the role of Docker daemon?

Even smaller Docker Image Size:

It's important to keep the size of a docker image small. Some builds have prerequisites that take up a lot of space the need to be later clean up. With Docker 17.06, a new feature "multi-stage builds" is introduced.

Get the code from the url: https://gitlab.com/Aurore0319/Docker-multi-stage-build.

This is a very very simple application written in Golang.

a) Launch Docker build and note the size of the image
b) Replace Dockerfile with the following content:

```
FROM golang as builder

WORKDIR /go/src/github.com/user/app
COPY . .
RUN set -x && \
   go get -d -v . && \
   CGO_ENABLED=0 GOOS=linux go build -a -installsuffix cgo -o app .
```

```
FROM scratch

WORKDIR /root/
COPY --from=builder /go/src/github.com/user/app .
CMD ["./app"]
```

Launch Docker build and look the size of the image. And Try to explain this approach.

Docker Compose:

Check docker compose:

Please launch the provided Virtual Machine with Virtual Box. Docker and Docker Compose are already installed. Check that everything works with:

```
$ docker-compose -f hello-world-composition.yml up
```

Dockerizing a simple application

You have a simple web application written in nodeJS to dockerize. The code is in the **webapp** folder.

a) What base image can we use?
b) What tools do you use to install the dependencies in the file **package.json?**
c) Launch the image with "docker run". Get the log and explain the error. Use *curl* to access the application and give the result.
d) When you work locally with nodejs, you can generate the files such as npm-debug.log or the modules already installed. How can we prevent to copy these content into the image? If we don't do this, it can take a long time before Docker starts "sending the build context to the demon".

Hints:

In the file **package.json,** you can find this: "scripts: {"start": "node app.js"}". So, you can start your application with command "npm start"

Integrating MongoDB

Now we will integrate a MongoDB database.

a) What base image can we use?
b) Create a *compose* file and define the services mongodb and the application dockerized in the previous step.
c) Launch the composed application with **docker-compose up.** You will see that the application can't connect to the database. Why? How can we do to make this work?
d) Use *curl* to access the application and give the result.
e) If we want to ensure that the web application starts after the service database. What can we do to achieve this? The option "depends_on" is really a good solution? If not, why?

Hints: You can write a shell script naming "wait-for-mongo.sh" for example. In this script, you can use "***nc -z $DATABASE_HOST $DATABASE_PORT***" to check if MongoDB is started completely. And finally, you should integrate this script into the service of the web application in the ***composition*** file. If you don't implement this, you will still have the error to connect to the database later. Don't forget to give the execution permission to your script.

Initialization of the database

By default, the mode of authentication of MongoDB isn't enabled. But in the real life, we should create a user admin who have the super rights on all databases and the other users to read/write in the specific database. In this step, we should:

a) **Enable the authentication mode in MongoDB**. When you define the environment variables **MONGO_INITDB_ROOT_USERNAME** and **MONGO_INITDB_ROOT_PASSWORD,** the base image of MongoDB on the Docker Hub will enable this mode for you. What can we do to achieve this securely? (You can't write the password with the clear text in the *compose* file or in the Dockerfile.) Explain your choice.

b) **Create another user and another database in MongoDB at the startup of MongoDB.** We need to create a specific user to read/write our data from/into the database "**myapp**". How can we create this user ? And the same question: how can we pass the name and the **password** of this new user to create the image? What modifications we should do in the ***compose*** file?

   In this step, you should create a Dockerfile (if you didn't do that yet) for the MongoDB service because you should add some customized modifications in addition to the based image.

   Hint:

   1) You can copy the *setupUser.sh* into **/docker-entrypoint-initdb.d/** in the Docker image.
   2) You have an incomplete *setupUser.sh* in the source code. The command to create a new user with the right permissions is already there. You can do the rest.
   3) To handle the sensitive information in Docker, you can use Docker Secret. In the current version, secrets can be created in the context of a Swarm. Begin by doing "docker swarm init" to create a manager node. For the rest, you should read this link: https://docs.docker.com/engine/swarm/secrets/#use-secrets-in-compose

When you finish all these steps, launch **docker-compose up.**

a) How to check everything works fine?
b) What messages in the log show that all the users are created successfully in MongDB?

c) Use curl to check if the web application works.

Load-balance the service:

We want to integrate a load balancer to our composed application. You can use the base image **dockercloud/haproxy** directly. Read the documentation (https://hub.docker.com/r/dockercloud/haproxy/) to see how to use it.

a) What modifications you should do in compose file?
b) Launch **docker-compose up,** how to check if everything is ok?
c) Explain why we don't need to update the configuration of the proxy manually if a new container is added?
d) How to Scale the number of containers of web application to 3? Propose two solutions.

Hints: With one solution, we don't need to modify the compose file. But with another solution, we need modify the compose file. What are the required modifications?

Docker Networking:

Now we have a database, a web application and a proxy. We don't want the database to be accessed by the proxy.

a) How can we isolate the services between them?
b) How can we test the new configuration?