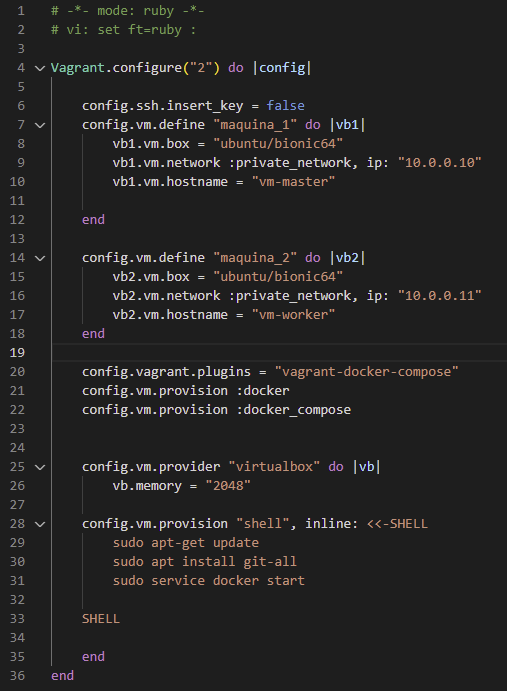
# VAGRANT FILE

First, I created the vm using the following vagrant file:

Texto

Descripción generada automáticamenteFirst service is “maquina\_1” and the second one is “maquina\_2”, both with different ip but in private network. In particular, the private network can be used to make a private network between multiple machines and the host and they will communicate each other.

Texto

Descripción generada automáticamente

The lines 21 and 22 provide docker and docker-compose to both vm, but it didn’t work with git, then, providing a shell terminal the SO is updated, git is installed via apt install, and then docker service is started (lines 28 to 33). Providing docker-compose run an error, so I needed to define the plugins at line 20.

To control multiple machines, the normal commands are used, but you specify which one is gonna follow the order by writing its name at the end. e.g “vagrant ssh maquina\_1”.

By default, Vagrant will share your project directory (the directory with the Vagrantfile) to /vagrant in the VM.

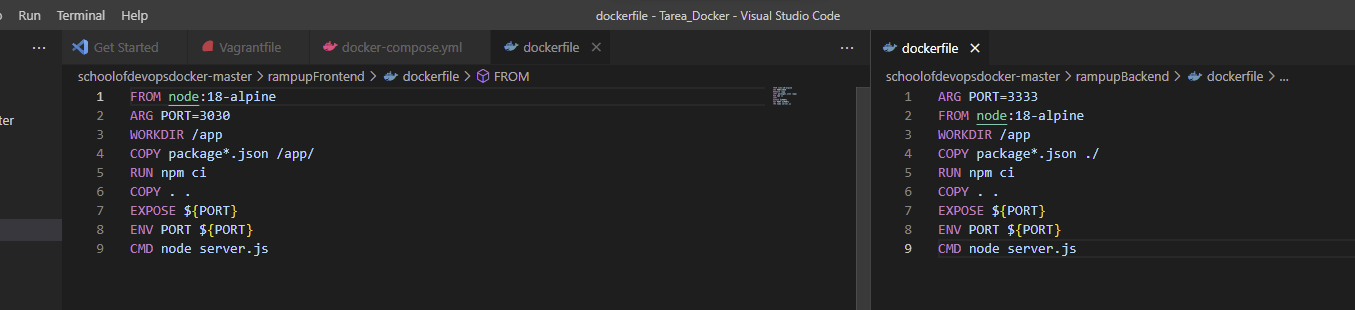
# REPOSITORY

The project repo was forked at: <https://gitlab.com/NavoSan/schoolofdevopsdocker>

And all the files (vagrant, docker-compose, dockerfiles) were pulled into it.

# DOCKERFILES

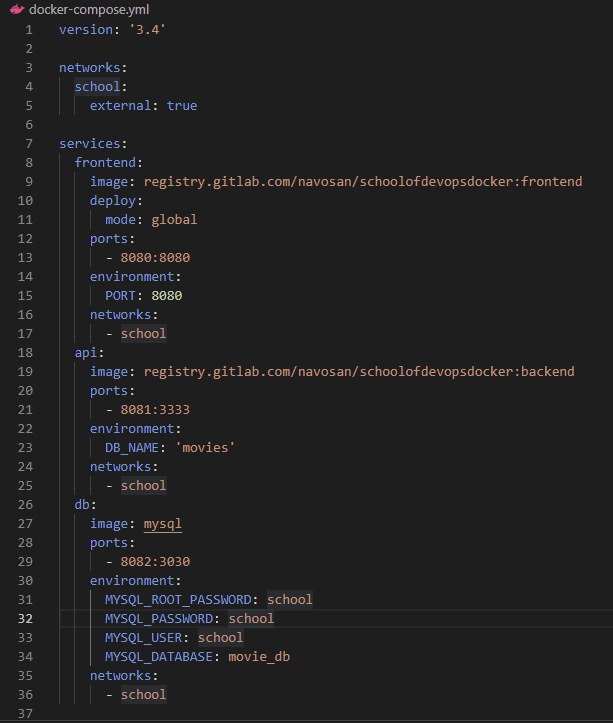
The respective frontend and backend dockerfiles are below:



Both of them pull a public node repository in the release 18-alpine, then create the directory and download the dependencies at package\*.json, with the CMD in line 9 is stated that every time the container is executed it will start node server.js.

# DOCKER-COMPOSE

The docker-compose.yml file is shown below:

So, I set 3 services, the “backend” for backend, the “frontend” for frontend and the “db” for the database.

Back and front containers are deployed by the images created with the dockerfiles and db is deployed with last mysql version image. Frontend is exposed in port 8080.

All containers are linked to the network “school” and all the environment variables were stablished as it were indicated in the class session.

Network “school” was determined as an external network to be able to communicate containers between nodes.

Note in frontend service declaration in lines 10-11 how deploy mode is global, this is because is asked this service to be deploy in each node.

Now, to generate the cluster with docker swarm I first declare the VM “maquina\_1” as the master. To do that I first enter to this VM to control it with the command: $ vagrant ssh maquina\_1

And there I initialize swarm mode:

Texto

Descripción generada automáticamente

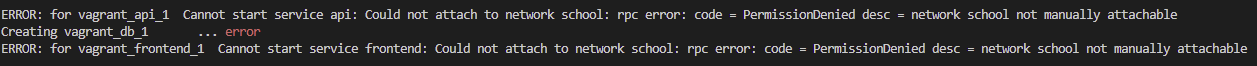
Then add the other VM as a worker node using the generated token. So I have two nodes:

Texto

Descripción generada automáticamente

Now, in order to create the network, I used: $ docker network create -d overlay school –attachable

That was because when I tried to deploy the application stack to the swarm in a default network driver the next error was sent: “network "school" is declared as external, but it is not in the right scope: "local" instead of "swarm" ”. The flag -attachable is necessary because of next error:



Then, with $ docker-compose up, the images are pulled and the containers in the master are created

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Then, to create the registry service I used:

* $ docker service create --name registro --publish published=5000,target=5000 registry:2

A generic image to create registries was used, and port 5000:5000 just to avoid default creation of ports. Then the services for Frontend and Backend are created from the docker-compose file with:

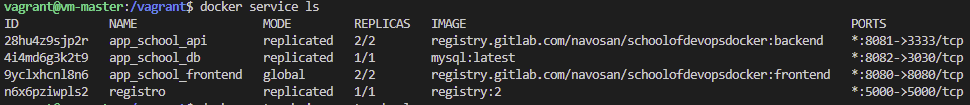
* $ docker stack deploy --compose-file docker-compose.yml app\_school

Now I have one replica for each service (at this point, one replica of frontend service at each node). But I need two replicas of “backend” and I suppose it doesn’t matter the node, then I let the docker swarm to load them by itself and just one replica of “db”.

To create the 2 replicas for backend the used command was:

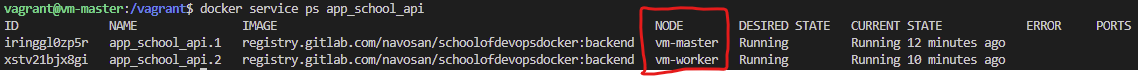
* $ docker service scale app\_school\_api=2

Created services are shown:



And we can confirm how they were distributed into the nodes by using:

* $ docker service ps app\_school\_api



# PROBLEMS

1. When I tried to create the replicas of the services in the different nodes, I was not able to run them in the worker node, the message error was “Not such an image”. That was because in the docker-compose file I first established de image like this

Texto

Descripción generada automáticamente

So, it searches the image in each node, but since they weren’t pulled manually in the other VM they were not found. To solve this I decided to build the images in the container registry in the repository in GitLab, and when the docker-compose try to build the containers it searches the image in the container registry link and not at the local.

1. To look at the frontend webapp, the port 8080 was exposed, but like I’m using a virtualbox VM as the host, I need to link the port 8080 in the VM with some in my computer (also 8080), that is possible from the configuration int the Oracle VM VritualBox Administrator.
2. I haven’t been able to execute the seed.js, a problem with MySQL for authentication is happening:

Texto

Descripción generada automáticamente

I’ll try to solve it next days. Frontend is working normally. Conecction with the seed.js is needed but also is needed to create the data base with the schema in rampupBackend directory. Not enough time to do that much things ☹

Interfaz de usuario gráfica, Aplicación, Sitio web

Descripción generada automáticamente