**Docker**

# **III- Volumes / Environments / Docker-compose :**

## **Volumes :**

## **Mount bind:**

We use volumes to synchronize the local application with the container.

We use the -v for volumes

**docker run –name node-app-container -v /$(pwd):/app -d -p 1000:4000 node-app**

*$(pwd)* : get the local path to the application folder

*/app :* the folder initialized in the Dockerfile (WORKDIR)

Mount bind: synchronize the application with the container (two-way binding)

**Problem :**

If you change something in the local application it will affect the container, and the same thing if we change in the container.

Example :

If we delete a file or a folder in the local application it will be deleted in the container, the same thing if we delete it in the container.

## **Read-only:**

*how can we do a one-way binding?*

It means when we change in the local it will affect the container, but when we change in the container it will not affect the application.

**docker run --name node-app-container -v /$(pwd):/app:ro -d -p 1000:4000 node-app**

We add :ro after /app (ro stand for read-only).

## **Anonymous volume:**

If we want to protect a file or a folder, we add it to another volume.

For example, if we delete the node\_modules folder from the local application and we don’t want it to be deleted in the container, we add the node\_modules in another volume.

**docker run --name node-app-container -v /$(pwd):/app:ro -v /app/node\_modules -d -p 1000:4000 node-app**

If we check in the docker application we will find 2 volumes : the one that contains the application and another one that contains the node\_modules folder.

*How can we bind just the src folder:*

**docker run --name node-app-container -v /$(pwd)/src:/app/src:ro -d -p 1000:4000 node-app**

if we change for example in node\_modules or another file or folder in the local that it isn’t in src folder it won’t be affected in the container.

# **Docker compose:**

It's in utility that comes with docker but not included in docker itself.

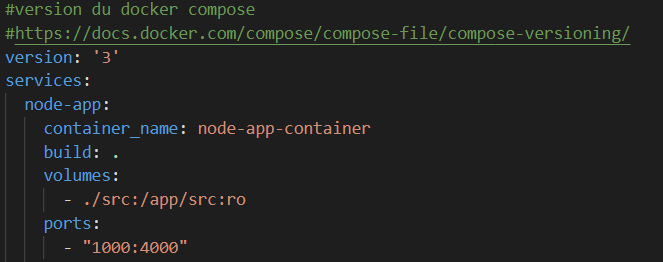
It helps to manage the container with better features.

To use this utility, we add a file called : ***docker-compose.yml***

To build a docker compose : **compose-docker up**

(We can add -d to not show the logs)

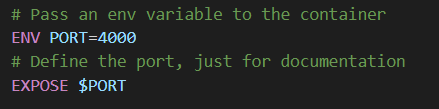
To stop a docker compose : **compose-docker down**



# **Environment variables:**

To passe a variable to the container we use :

* *Dockerfile:*

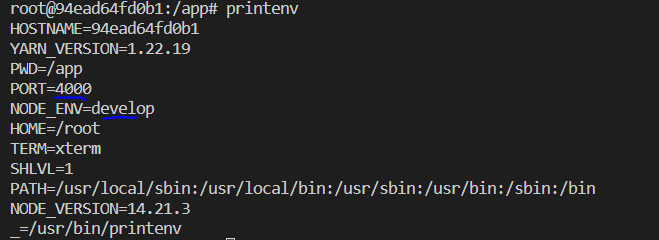


* *Run command: (we use –env)*

**docker run --name node-app-container -v ${pwd}/src:/app/src:ro --env PORT=4000 --env NODE\_ENV=develop -d -p 1000:4000 node-app**

Here we pass 2 env variables : PORT and NODE\_ENV

To know the env variables existing, we enter with exec command to the container then we execute **printenv**



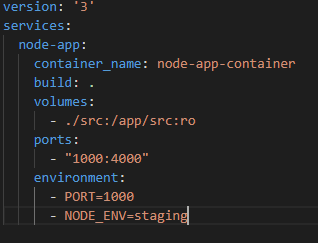
If we have multiple env, we can add all the environment variables into an .env file



And execute the command :

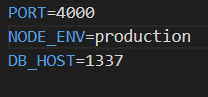
**docker run --name node-app-container --env-file ./.env -d -p 1000:4000 node-app**

* *use docker-compose*



to add it in the .env file

.env:



docker-compose:

