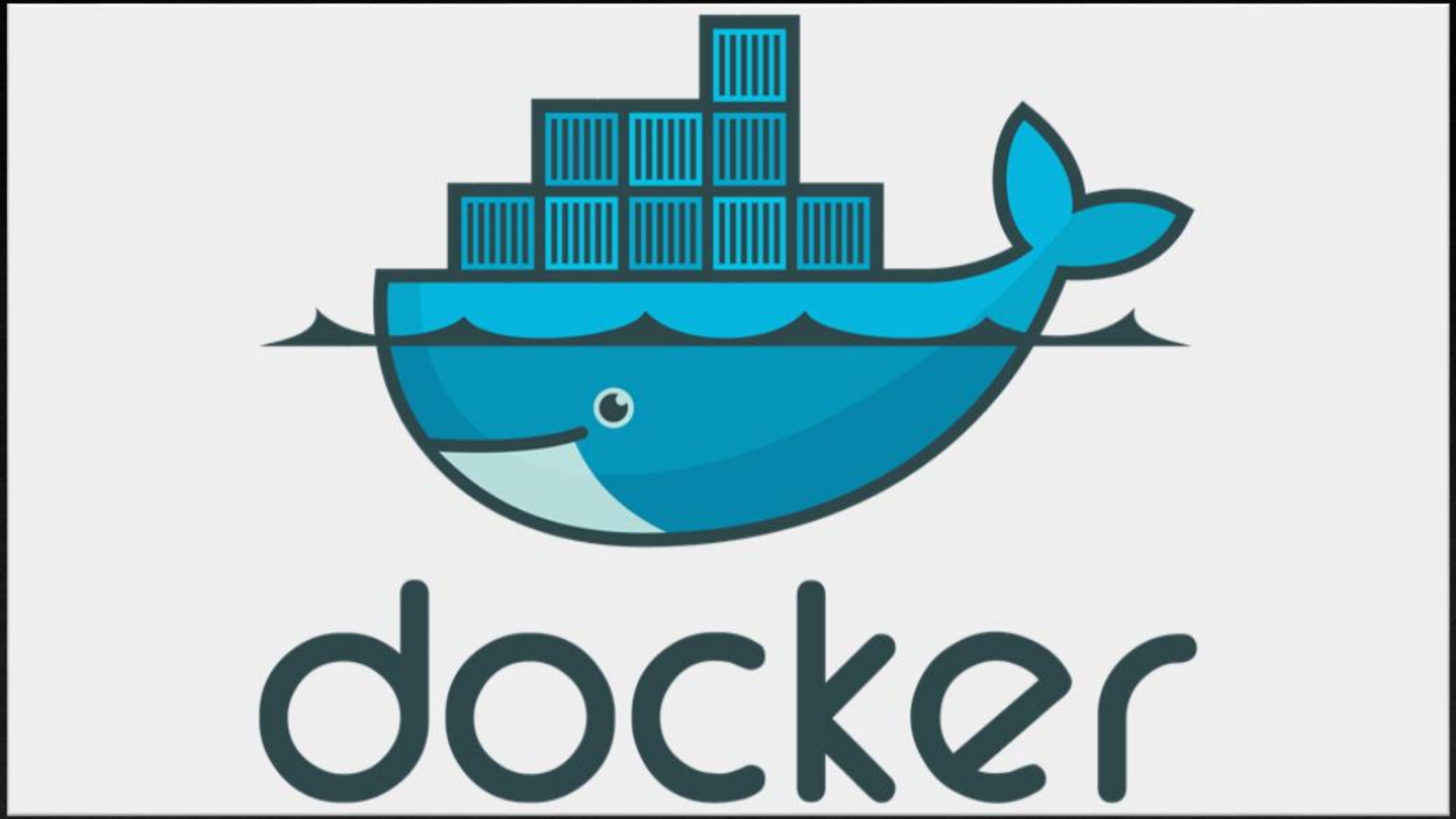
PS6 - Temps plein Docker compose

Nassim Bounouas – Juin 2023

POLYTECH NICE SOPHIA
UNIVERSITÉ CÔTE D'AZUR


docker

Rappels du cours de système

UNIVERSITÉ CÔTE D'AZUR

Virtualisation vs Conteneurisation

✓ Machine Virtuelle

VM

Apps	Apps	Apps
Bins / Libs	Bins / Libs	Bins / Libs
Système d'exploitation invité Windows 10	Système d'exploitation invité Ubuntu	Système d'exploitation invité android

Hyperviseur Type 2 VirtualBox

Système d'Exploitation Hôte Windows 11

Infrastructure / Matériel (*Hardware*)

✓ Conteneurisation

– Légèreté et isolation pour l'exécution de processus

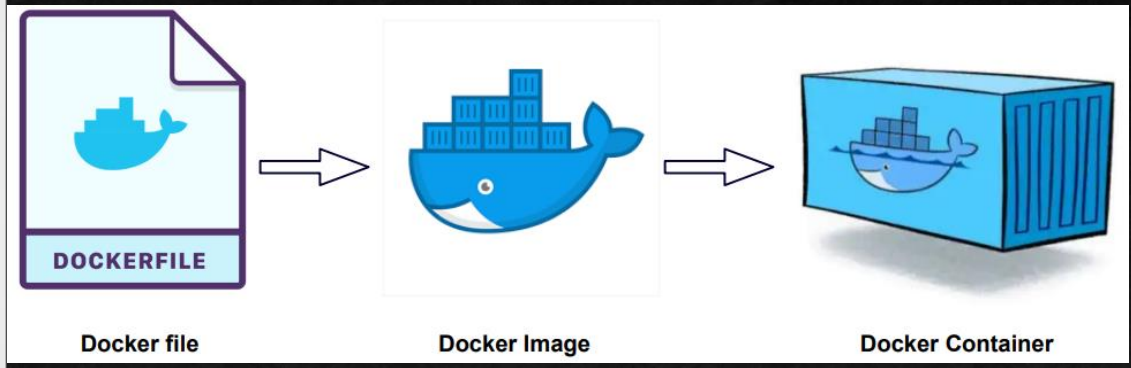
Conteneur

Apps	Apps	Moteur de Conteneur docker
Bins / Libs	Bins / Libs	

Système d'Exploitation Hôte Linux

Infrastructure / Matériel (*Hardware*)

Présentation: S. Lavirotte – Auteurs : ... et al* 5

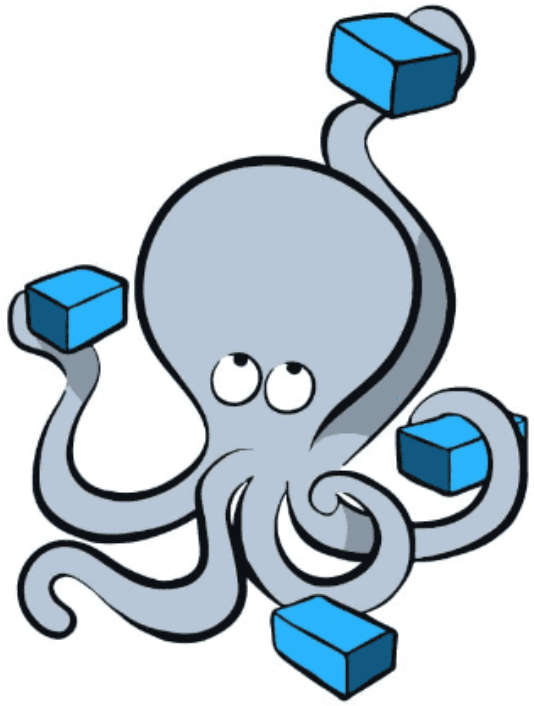


Dockerfile : Rappel

- FROM
- ENV
- ARG
- WORKDIR
- RUN
- ADD / COPY
- CMD / ENTRYPOINT
- USER
- EXPOSE
- VOLUME
- HEALTHCHECK

Dockerfile : Rappel

- FROM : Spécifie l'image de base à partir de laquelle construire le conteneur.
- ENV : Définit une variable d'environnement pour une utilisation dans le conteneur.
- ARG : Définit une variable pouvant être passée à l'image lors de sa construction.
- WORKDIR : Définit le répertoire de travail à utiliser dans le conteneur.
- RUN : Exécute une commande lors de la construction de l'image.
- ADD / COPY : Ajoute ou copie des fichiers et des répertoires dans le conteneur.
- CMD / ENTRYPOINT : Définit la commande par défaut à exécuter lorsque le conteneur est démarré.
- USER : Spécifie l'utilisateur courant
- EXPOSE : Indique les ports sur lesquels le conteneur écoutera les connexions.
- VOLUME : Crée un point de montage marqué comme externe
- HEALTHCHECK: Exécute une vérification pour déterminer l'état de santé du conteneur.



docker Compose

Qu'est-ce que docker compose ?

- Outil d'orchestration de conteneurs
- Permet de gérer "une application" composée de plusieurs conteneurs
- Permet d'aggréger les paramètres, configurations et dépendances nécessaires au lancement des conteneurs
- Gain de temps et de reproductibilité

Les concepts de compose

- Images
- Conteneurs
- Réseaux
- Volumes
- Variables d'environnement

- Services
- Fichier de composition
- Commandes propres à docker compose

Services

Un service représente une partie d'une application, généralement un conteneur, mais peut également être un ensemble de conteneurs liés entre eux.

Les services peuvent avoir des dépendances entre eux. Ils permettent de gérer et de coordonner plusieurs conteneurs travaillant ensemble pour fournir une fonctionnalité complète.

Composition

```python
import time

import redis
from flask import Flask

app = Flask(__name__)
cache = redis.Redis(host='redis', port=6379)

def get_hit_count():
    retries = 5
    while True:
        try:
            return cache.incr('hits')
        except redis.exceptions.ConnectionError as exc:
            if retries == 0:
                raise exc
            retries -= 1
            time.sleep(0.5)

@app.route('/')
def hello():
    count = get_hit_count()
    return 'Hello World! I have been seen {} times.\n'.format(count)
```

```
# syntax=docker/dockerfile:1
FROM python:3.7-alpine
WORKDIR /code
ENV FLASK_APP=app.py
ENV FLASK_RUN_HOST=0.0.0.0
RUN apk add --no-cache gcc musl-dev linux-headers
COPY requirements.txt requirements.txt
RUN pip install -r requirements.txt
EXPOSE 5000
COPY . .
CMD ["flask", "run"]
```

Source: https://docs.docker.com/compose/gettingstarted/

Composition

```yaml
services:
  web:
    build: .
    ports:
      - "8000:5000"
  redis:
    image: "redis:alpine"
```

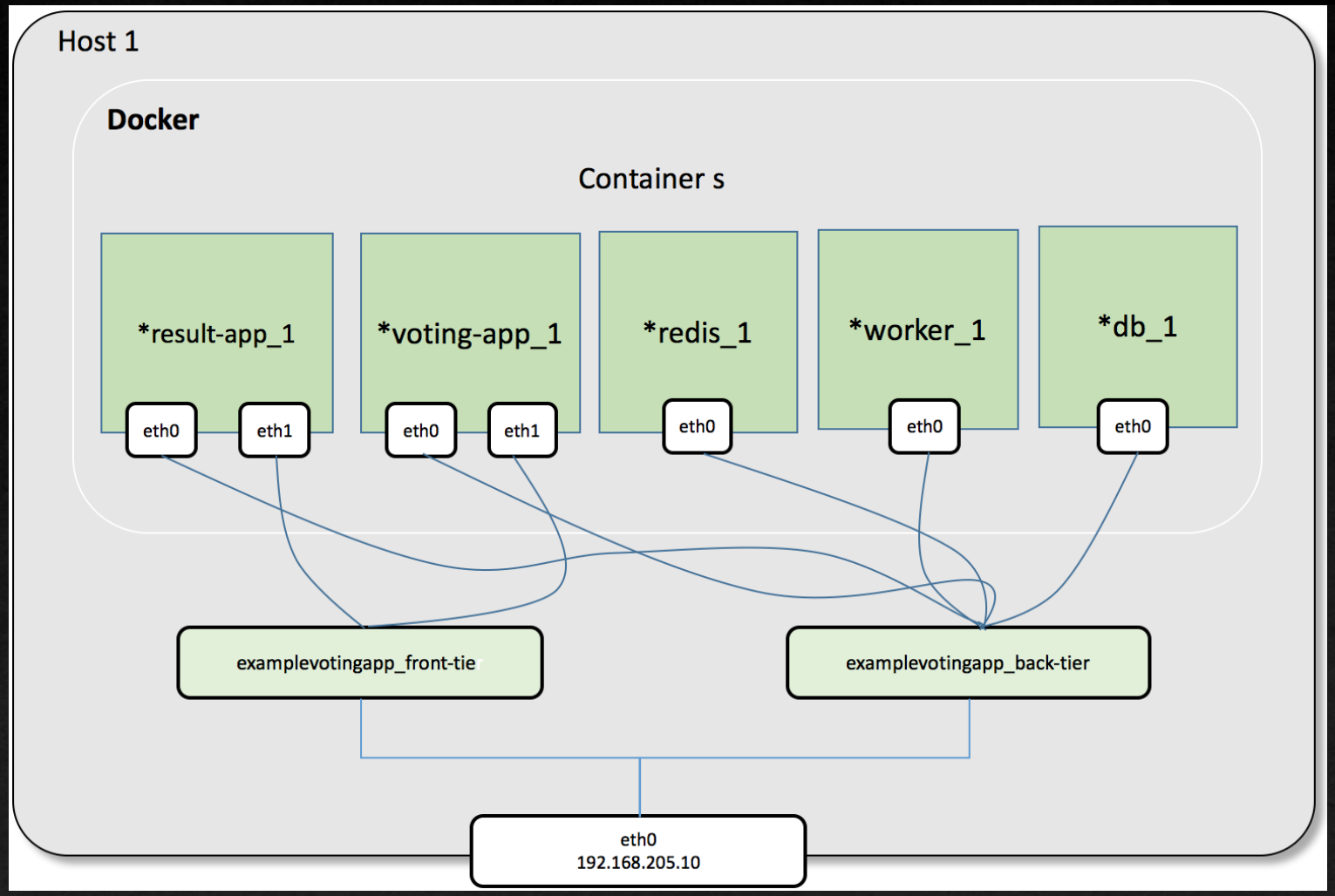
```
$ docker compose up

Creating network "composetest_default" with the default driver
Creating composetest_web_1 ...
Creating composetest_redis_1 ...
Creating composetest_web_1
Creating composetest_redis_1 ... done
Attaching to composetest_web_1, composetest_redis_1
web_1    |  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
redis_1  | 1:C 17 Aug 22:11:10.480 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
redis_1  | 1:C 17 Aug 22:11:10.480 # Redis version=4.0.1, bits=64, commit=00000000, modified=0, pid=1, just started
redis_1  | 1:C 17 Aug 22:11:10.480 # Warning: no config file specified, using the default config. In order to specif
web_1    |  * Restarting with stat
redis_1  | 1:M 17 Aug 22:11:10.483 * Running mode=standalone, port=6379.
redis_1  | 1:M 17 Aug 22:11:10.483 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/ne
web_1    |  * Debugger is active!
redis_1  | 1:M 17 Aug 22:11:10.483 # Server initialized
redis_1  | 1:M 17 Aug 22:11:10.483 # WARNING you have Transparent Huge Pages (THP) support enabled in your kernel. T
web_1    |  * Debugger PIN: 330-787-903
redis_1  | 1:M 17 Aug 22:11:10.483 * Ready to accept connections
```

```
$ docker compose down --volumes
```

Réseaux

- Permettre la communication entre les différents conteneurs d'une application
- Par défaut privés et isolés
- Isolation : seuls les conteneurs appartenant au même réseau peuvent communiquer entre eux
- Simplification : les conteneurs peuvent se référencer par leur nom de service au sein du réseau, sans avoir besoin de connaître les adresses IP spécifiques.
- Configuration : Docker Compose permet de créer des réseaux personnalisés avec des configurations spécifiques.

Réseaux




```yaml
version: "2"

services:
  voting-app:
    build: ./voting-app/.
    volumes:
     - ./voting-app:/app
    ports:
      - "5000:80"
    links:
      - redis
    networks:
      - front-tier
      - back-tier

  result-app:
    build: ./result-app/.
    volumes:
      - ./result-app:/app
    ports:
      - "5001:80"
    links:
      - db
    networks:
      - front-tier
      - back-tier

  worker:
    build: ./worker
    links:
      - db
      - redis
    networks:
      - back-tier

  redis:
    image: redis
    ports: ["6379"]
    networks:
      - back-tier

  db:
    image: postgres:9.4
    volumes:
      - "db-data:/var/lib/postgresql/data"
    networks:
      - back-tier

volumes:
  db-data:

networks:
  front-tier:
  back-tier:
```

Source: https://docker-k8s-lab.readthedocs.io/en/latest/docker/docker-compose.html

Installation

Important

Compose V1 no longer receives updates and will not be available in new releases of Docker Desktop after June 2023.

Compose V2 is included with all currently supported versions of Docker Desktop. For more information, see Migrate to Compose V2.

Draw me a Dockerfile / docker-compose.yml

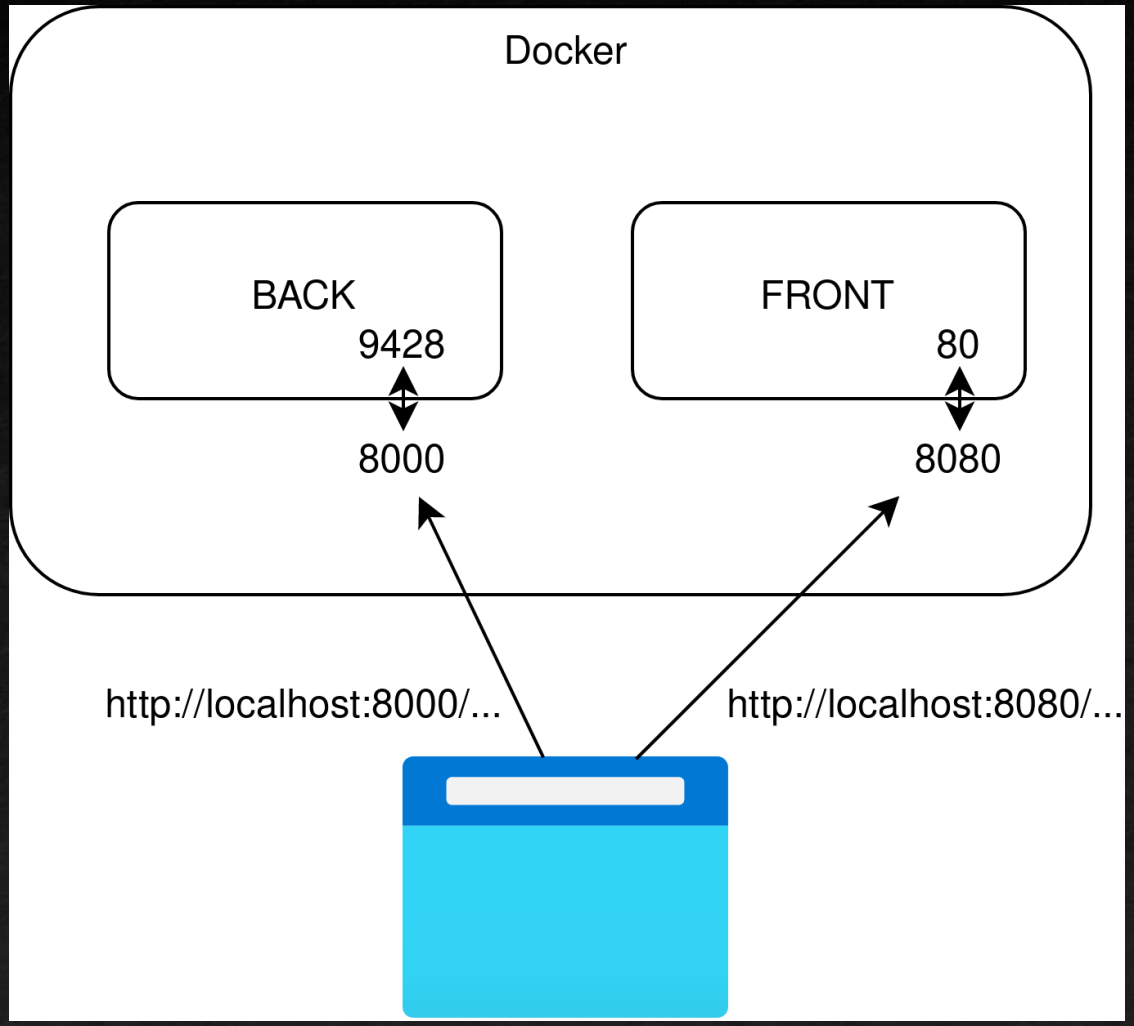
- Version(s)
- Image de base
- Packages manager
- Port
- LLM approach
- ....


LET'S
DO THIS

1. Dockeriser le front et le back

- 2 Dockerfiles
  - Back : Node
  - Front : Node (build) + Nginx (hosting : site statique contenant l'url du back)
    - Configuration nginx non variable (localhost)
- Ne pas être root
- Localhost -> 2 ports différents
- Conteneurs indépendants (pas de compose)

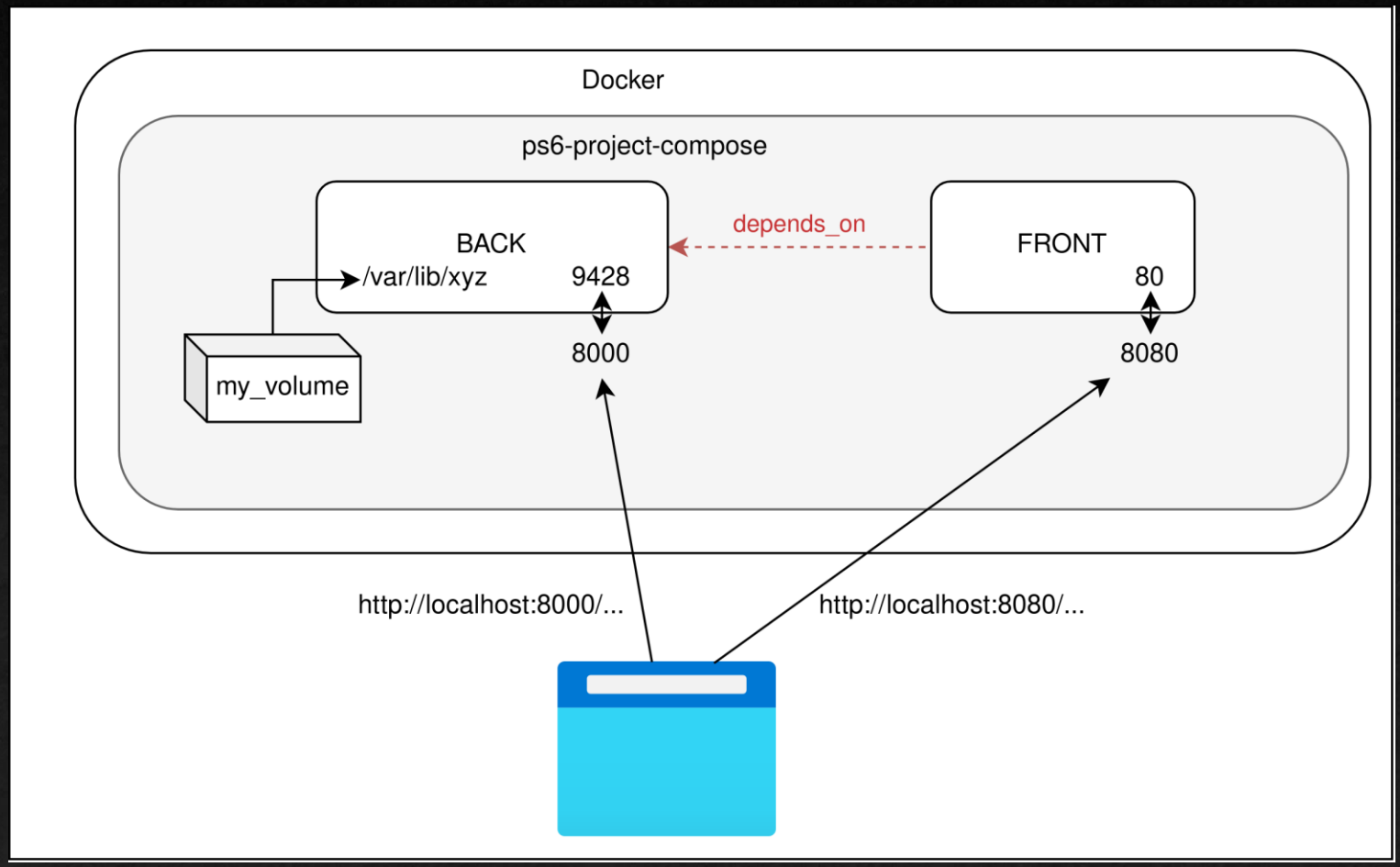
1. Dockeriser le front et le back



2. Orchestrer le front et le back

- Monter les 2 conteneurs dans un compose
- Variabiliser le build (localhost or custom host)
- Mise en place d'un ordonnancement + healthcheck
  - 1 - Backend
  - 2 - Frontend
- Configurer les volumes

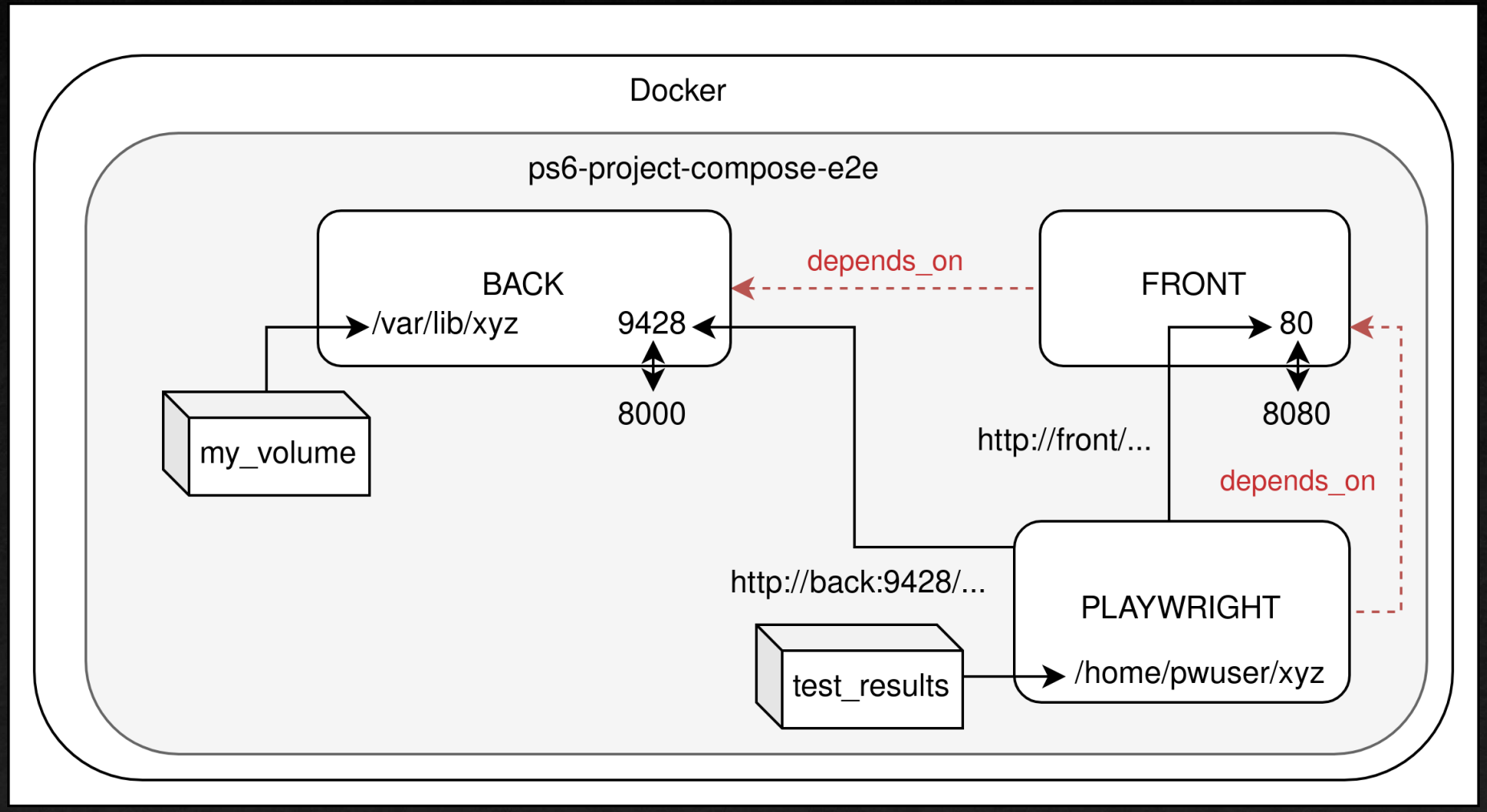
2. Orchestrer le front et le back



3. Dockeriser l'exécution des tests !

- Dupliquer le setup précédent
- Dockeriser l'exécution des tests
- Ajouter le conteneur au setup dupliqué
- Configurer le backend en mode test
- Configurer le front pour qu'il appelle le backend au travers du network compose
- Charger le front dans le conteneur de test, runner les tests et stocker le(s) résultat(s) dans un volume

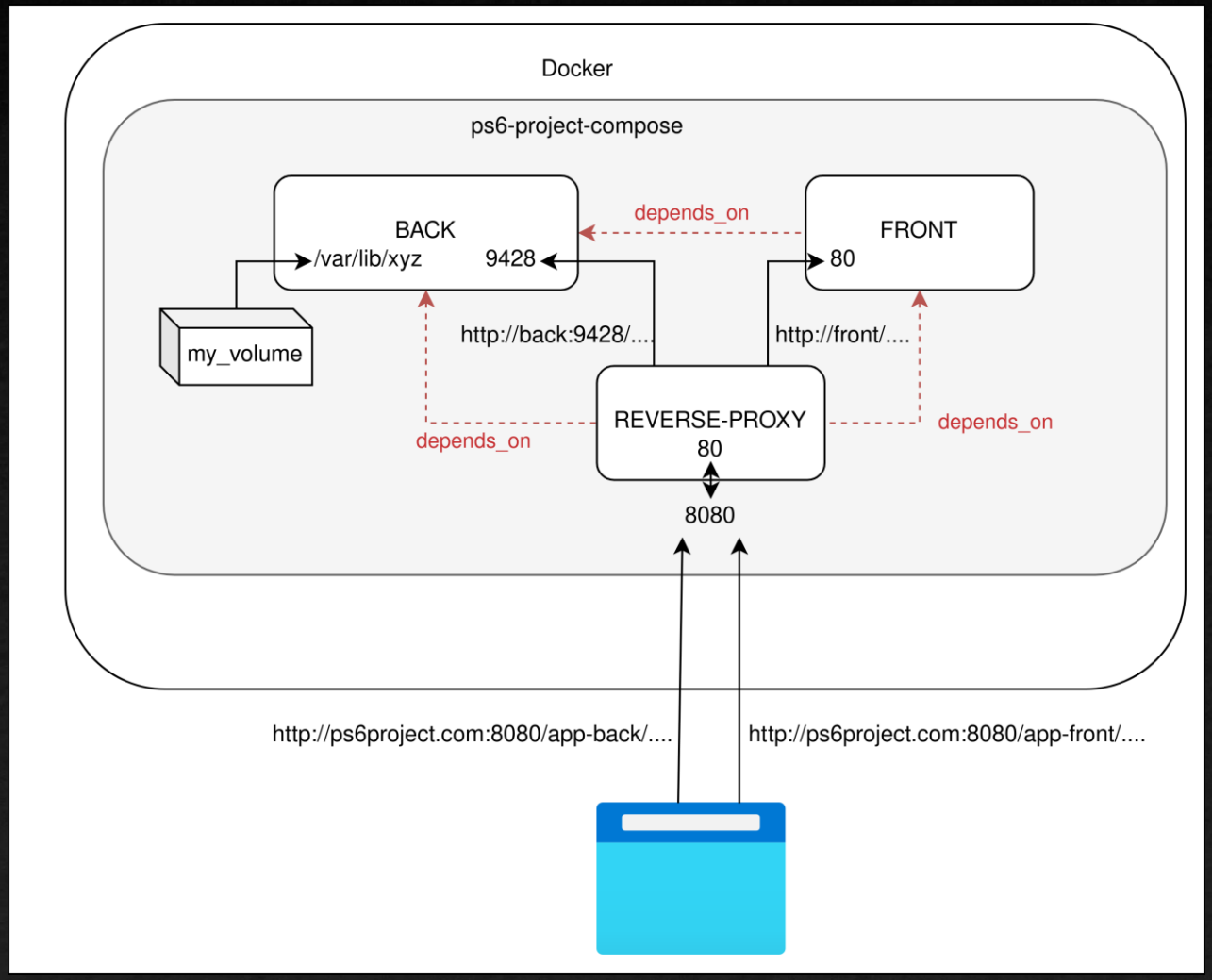
3. Dockeriser l'exécution des tests !



4. Proxifier l'ensemble des appels

- Reprendre le setup 2.
- Ajouter un conteneur de proxyfication (apache)
- Lui fournir deux règles de proxyfication sur un domaine commun (voir configuration du fichier hosts en fonction de l'OS) et des sous chemins
- Adapter le front pour qu'il prenne son sous-chemin comme url de base
- Configurer l'URL du back proxyfié dans le front
- Adapter si nécessaire pour la récupération des ressources

4. Proxifier l'ensemble des appels

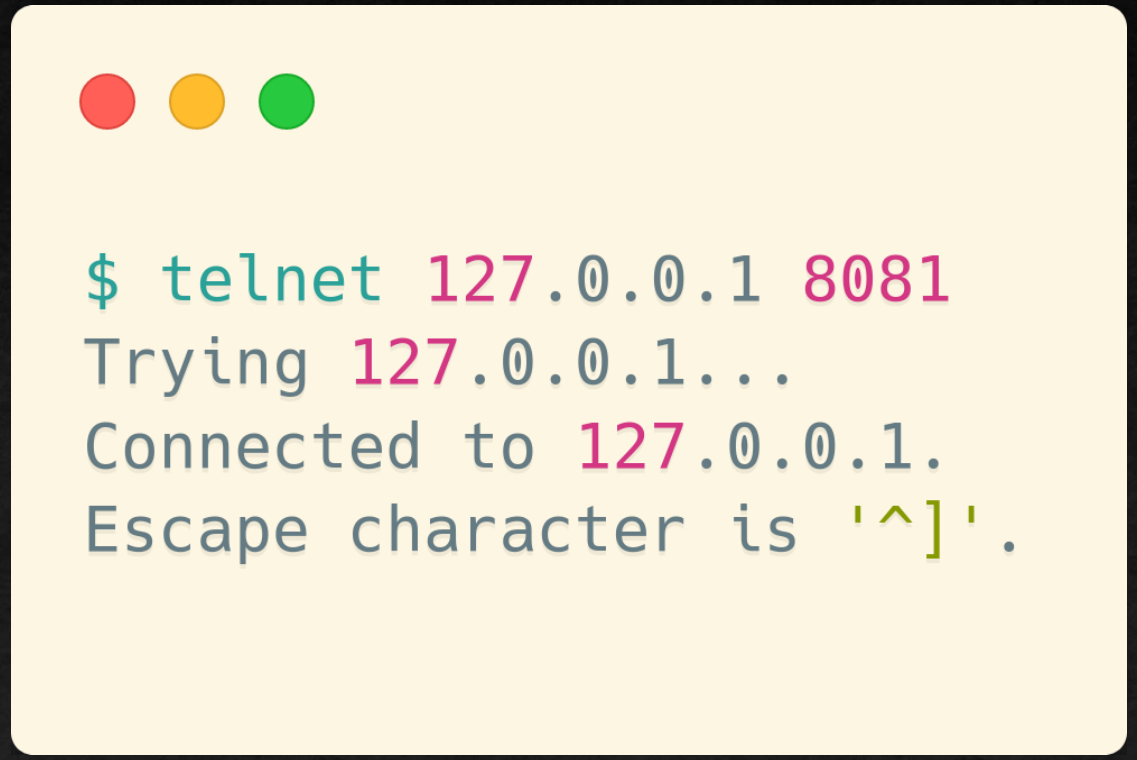


Livrable OPS

- Dans un dossier "ops" à la racine du repo taggué "FINAL"
  - docker-compose.yml : Setup de run (pas de tests playwright)
  - run.sh : script lançant le compose de run
  - docker-compose-e2e.yml : Setup de run des tests
  - run-e2e.sh : script lançant le compose de run de test
  - Healthcheck back
  - Healthcheck front
  - readme.md: Explications et statut du livrable, des healthchecks

La boite à problèmes

Tester l'accès réseau



Suivre les logs d'un conteneur détaché

```
$ docker logs --follow ps6-correction-td1-td2-v2-back-1

> backend-ps6-starter-quiz@1.0.0 start
> node app/index.js

[2023-06-13T22:26:16.034Z] Server is listening on port 9428
[2023-06-13T22:26:24.941Z] GET /api/status 200 4.967 ms - 4
[2023-06-13T22:26:35.037Z] GET /api/status 200 1.916 ms - 4
[2023-06-13T22:26:45.232Z] GET /api/status 200 9.522 ms - 4
[2023-06-13T22:26:55.376Z] GET /api/status 200 1.035 ms - 4
[2023-06-13T22:27:05.534Z] GET /api/status 200 0.634 ms - 4
```

Obtenir des logs verbeux lors du up

```
$ BUILDKIT_PROGRESS=plain docker compose up
[+] Running 2/2
#1 [back internal] load .dockerignore
#1 transferring context: 2B done
#1 DONE 0.0s

#2 [back internal] load build definition from Dockerfile
#2 transferring dockerfile: 379B done
#2 DONE 0.0s

#3 [reverse-proxy internal] load build definition from Dockerfile
#3 transferring dockerfile: 496B done
#3 DONE 0.1s
```

Obtenir des logs du healthcheck

```
$ docker inspect --format "{{json .State.Health }}" ps6-correction-td1-td2-v2-back-1  | jq
{
  "Status": "healthy",
  "FailingStreak": 0,
  "Log": [
    {
      "Start": "2023-06-14T00:30:17.895581255+02:00",
      "End": "2023-06-14T00:30:18.002144538+02:00",
      "ExitCode": 0,
      "Output": "HEALTHCHECK\n  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current\n                                 Dload  Upload   Total   Spent    Left  Speed\n\r  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0\r100     4  100     4    0     0   2544      0 --:--:-- --:--:-- --:--:--
4000\n\"ok\"\n"
    },
    {
      "Start": "2023-06-14T00:30:28.016486444+02:00",
      "End": "2023-06-14T00:30:28.139315608+02:00",
      "ExitCode": 0,
      "Output": "HEALTHCHECK\n  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current\n                                 Dload  Upload   Total   Spent    Left  Speed\n\r  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0\r100     4  100     4    0     0   1249      0 --:--:-- --:--:-- --:--:--
1333\n\"ok\"\n"
    }
  ]
}
```

Variables d'environnement

```
page.goto(`${testUrl}`);
```

```
page.goto("http://localhost:4200");
```

Cache-Cache

```
$ docker image prune -a
WARNING! This will remove all images without at least one container associated to them.
Are you sure you want to continue? [y/N]

$ docker system prune -a
WARNING! This will remove:
  - all stopped containers
  - all networks not used by at least one container
  - all images without at least one container associated to them
  - all build cache

Are you sure you want to continue? [y/N]
```


How to debug ?

#ps6-22-23


Aye, You could do that !