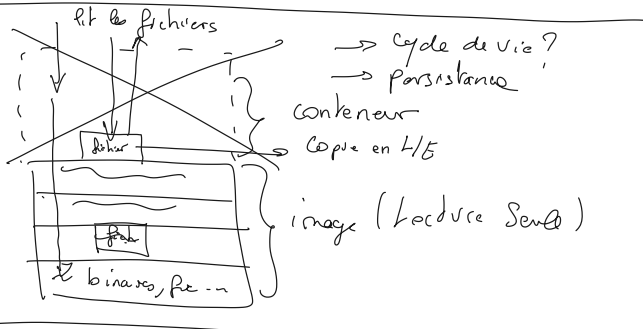
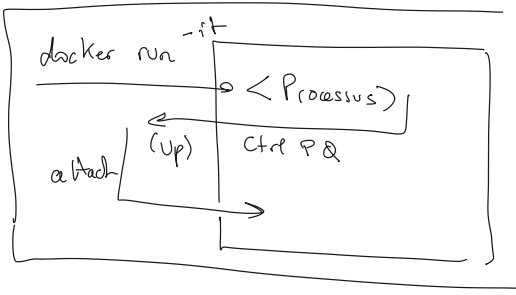
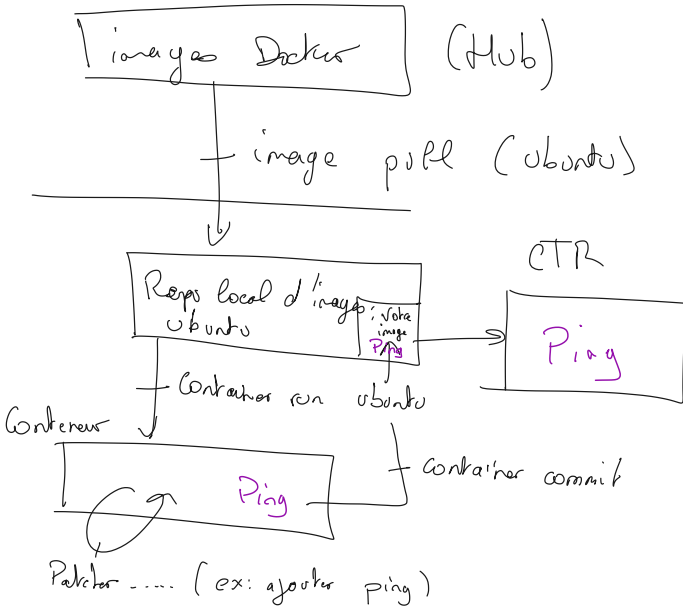
Bonjour tout le monde








```
docker container run -it --name myu20 ubuntu
apt update && apt -y install iputils-ping
docker container commit myu20 myubuntu
docker container run -it --rm myubuntu
```

```
docker container run -d --name nginx nginx
docker container exec -it nginx /bin/bash
apt update
apt install procps
ps -faux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root          45  0.1  0.0   3872  3300 pts/0    Ss   07:12   0:00 /bin/bash
root         373  0.0  0.0   7644  2792 pts/0    R+   07:12   0:00 \_ ps -faux
root           1  0.0  0.0  10664  6024 ?        Ss   07:11   0:00 nginx: master process nginx -g daemon off;
nginx         24  0.0  0.0  11100  2676 ?        S    07:11   0:00 nginx: worker process
nginx         25  0.0  0.0  11100  2676 ?        S    07:11   0:00 nginx: worker process
nginx         26  0.0  0.0  11100  2676 ?        S    07:11   0:00 nginx: worker process
nginx         27  0.0  0.0  11100  2676 ?        S    07:11   0:00 nginx: worker process
nginx         28  0.0  0.0  11100  2676 ?        S    07:11   0:00 nginx: worker process
nginx         29  0.0  0.0  11100  2676 ?        S    07:11   0:00 nginx: worker process
nginx         30  0.0  0.0  11100  2676 ?        S    07:11   0:00 nginx: worker process
nginx         31  0.0  0.0  11100  2676 ?        S    07:11   0:00 nginx: worker process
```

lancer un PG :
docker container run --name pg1 -e POSTGRES_PASSWORD=password -d postgresdocker container exec -it pg1 psql -U postgres

Lancer un mysql :
docker container run --name ms1 -e MYSQL_ROOT_PASSWORD=password -d mysqldocker container exec -it ms1 mysql --host=localhost -u root --protocol=tcp -p

Reference Dockerfile
https://docs.docker.com/engine/reference/builder/

```
FROM ubuntu

RUN apt update && apt -y upgrade && apt -y install iputils-ping procps

CMD ["ping", "www.google.fr"]
```

Exercice :
Fabriquez un petit script "Hello World" en Python, ou Php, ou...

hello.py :
print('Hello World')

python hello.py

Créez une image sur une base ubuntu ou centos (pas avec Python déjà installé)
Installez-y python
définissez votre script comme démarrage par défaut
testez votre image

root@u20:/home/student/HelloWorldPython# cat 00-build.sh
docker image build -t nboost/hellopython .
root@u20:/home/student/HelloWorldPython# cat 01-test.sh
docker container run -it --rm nboost/hellopython
root@u20:/home/student/HelloWorldPython# cat 00-build.sh
docker image build -t nboost/hellopython .

Solution de l'exercice :

```
root@u20:/home/student/HelloWorldPython# cat 00-build.sh
docker image build -t nboost/hellopython .
root@u20:/home/student/HelloWorldPython# cat 01-test.sh
docker container run -it --rm nboost/hellopython
root@u20:/home/student/HelloWorldPython# cat Dockerfile
FROM ubuntu

RUN apt update && apt -y upgrade && apt install -y python3
WORKDIR /
COPY files/* /

#CMD ["python3", "hello.py"]    -> lance le hello world
CMD [ "/bin/bash", "/start.sh" ]    -> démonstration pour lancer hello et rester dans le conteneur.
root@u20:/home/student/HelloWorldPython# cat files/hello.py
print("Hello World!")
root@u20:/home/student/HelloWorldPython# cat files/start.sh
#/bin/bash

python3 /hello.py
/bin/bash


ENTRYPOINT :
root@u20:/home/student/HelloWorldPython# docker container run -it --rm nboost/hellopython
/hello.py /hello.py
Hello World!
root@u20:/home/student/HelloWorldPython# docker container run -it --rm nboost/hellopython
/hello.py --help
Hello World!
root@u20:/home/student/HelloWorldPython# docker container run -it --rm nboost/hellopython
/hello.py param1 param2 ....
Hello World!
root@u20:/home/student/HelloWorldPython# docker container run -it --rm --
entrypoint=/bin/bash nboost/hellopython
root@231df8cca03f:/# python3 /hello.py
Hello World!
root@231df8cca03f:/# exit
exit
root@u20:/home/student/HelloWorldPython# cat Dockerfile
FROM ubuntu

RUN apt update && apt -y upgrade && apt install -y python3
WORKDIR /
COPY files/* /

ENTRYPOINT ["python3"]
CMD [ "/hello.py" ]
```

Envoi de votre image dans le hub Docker :
root@u20:/home/student/HelloWorldPython# **docker image push nboost/hellopython**
Using default tag: latest
The push refers to repository [docker.io/nboost/hellopython]
71cc80fb48ae: Preparing
09cdf5941a16: Preparing
9f54eef41275: Preparing
denied: requested access to the resource is denied
root@u20:/home/student/HelloWorldPython# docker login
Login with your Docker ID to push and pull images from Docker Hub. If you don't have a Docker ID, head over to https://hub.docker.com to create one.
Username: nboost
Password:
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
root@u20:/home/student/HelloWorldPython# **docker image push nboost/hellopython**
Using default tag: latest
The push refers to repository [docker.io/nboost/hellopython]
71cc80fb48ae: Pushed
09cdf5941a16: Pushed
9f54eef41275: Mounted from library/ubuntu
latest: digest: sha256:f7a9cd1a1d0d9c93ee10e5fe7134874d2b10799c85b1ff9998fbca1d06ebe596 size: 948

root@u20:/home/student/hellopython2# cat ../HelloWorldPython/Dockerfile
FROM ubuntu

RUN apt update && apt -y upgrade && apt install -y python3
WORKDIR /
COPY files/* /

ONBUILD RUN touch /imagereconstruite

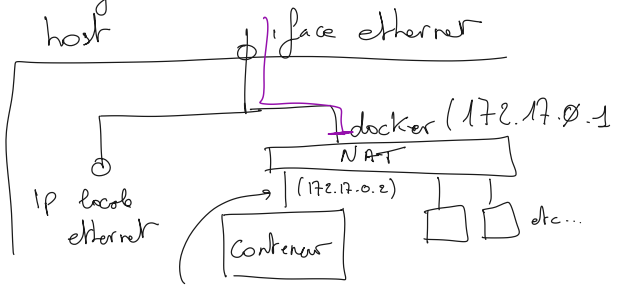
ENTRYPOINT ["python3"]
CMD ["/hello.py"]
root@u20:/home/student/hellopython2# cat Dockerfile
FROM nboost/hellopython

RUN apt-get clean

ENTRYPOINT ["/bin/bash"]
root@u20:/home/student/hellopython2# docker container run -it --rm nboost/hellopython2
root@555f2fb32d9e:/# ls /
bin boot dev etc hello.py home imagereconstruite lib lib32 lib64 libx32 media mnt opt proc root run sbin srv start.sh sys tmp usr var

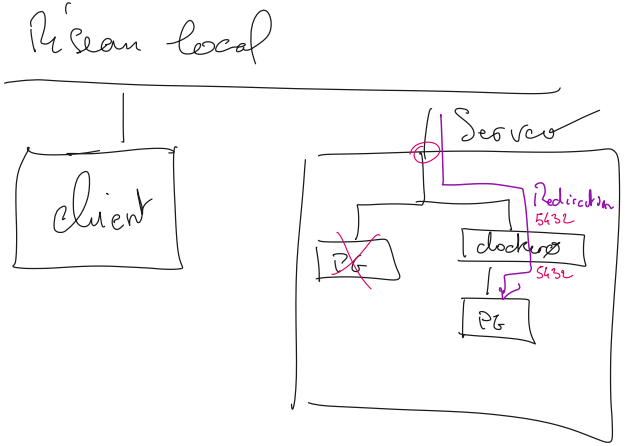
Install Docker →
Créé une iface "docker0"
(172.17.0.0/16)

Par défaut :



une IP de 172.17.0.0/16

NAT = Box internet
→ Rendre un Service accessible
avec une redirection de Port



docker container run -d --name pg1 -e POSTGRES_PASSWORD=password -p 5432:5432 postgres

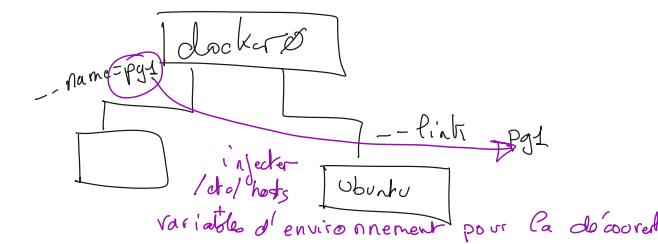
-p <port sur le host>:<port du conteneur>

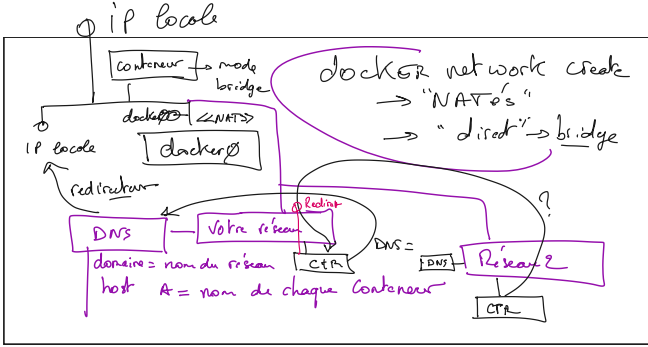
apt install postgresql-client*

psql --user=postgres --host=<ip locale>

Contrôle :

```
root@u20:/home/student# docker container ls
CONTAINER ID   IMAGE      COMMAND                  CREATED         STATUS         PORTS                                       NAMES
32bfc1eb76f5   postgres   "docker-entrypoint.s..."   8 minutes ago   Up 8 minutes   0.0.0.0:5432->5432/tcp, :::5432->5432/tcp   pg1
```





- docker network create vnet0
- docker container run -d --name pg1 -e POSTGRES_PASSWORD=password -p 5432:5432 **--network=vnet0** postgres
- docker container run --rm --name demo -it --network=vnet0 postgres /bin/bash
  - -> permet de lancer psql et se connecter dans le host <pg1>

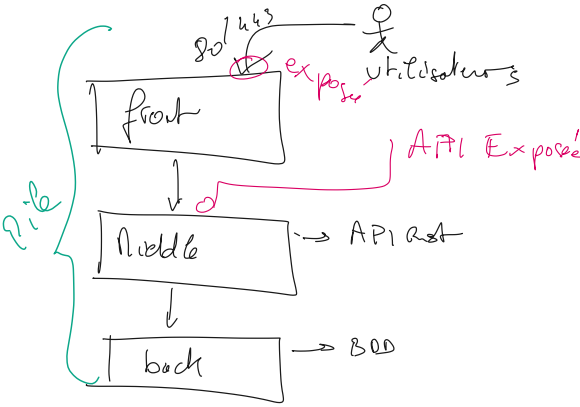
Résumé : Par défaut, pas de DNS avec docker0 mais utiliser --link

Recommandé : creez un réseau par projet/pile, n'utilisez pas --link
→ résolutions via DNS renseigné automatiquement

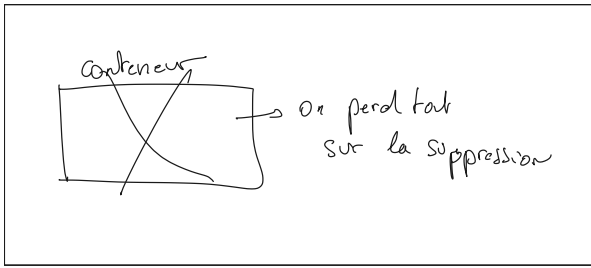
- j'ai une pile wordpress à déployer
le front doit être accessible avec l'IP publique

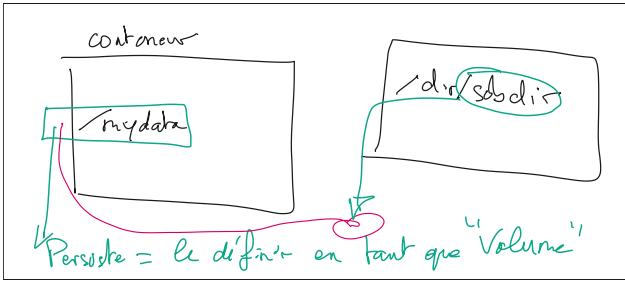
— créer un réseau ? Recommandé
— lancer le front avec -p ? OUI
- lancer le back avec -p ? → NON

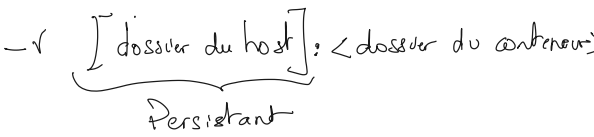
Pile / stack (architecture n-tier)



Volumes







Lancez deux conteneurs, qui, via un dossier sur le host bien connu (celui que vous voulez), peuvent s'échanger des fichiers entre eux (il est recommandé de passer par deux fenêtres shell pour cet exercice)

```
docker container run -d --name pg1 -e POSTGRES_PASSWORD=password -p 5432:5432 --network=vnet0 -v /root/myvolume/pg1:/var/lib/postgresql/data postgres:10
```

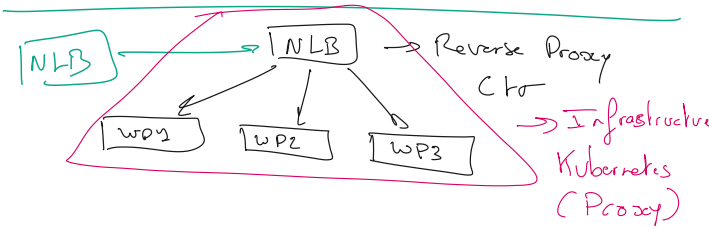
```
sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose
docker-compose --version
```

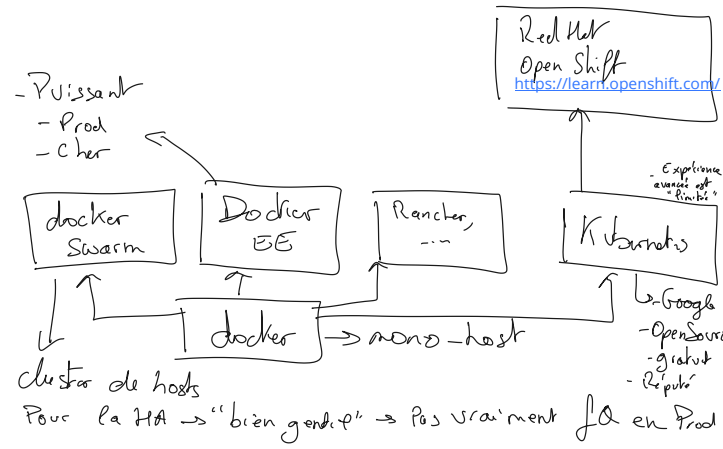
```
version: "3.9"

services:
  db:
    image: mysql:5.7
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: somewordpress
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    depends_on:
      - db
    image: wordpress:latest
    volumes:
      - wordpress_data:/var/www/html
    ports:
      - "8000:80"
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
      WORDPRESS_DB_NAME: wordpress
volumes:
  db_data:
    driver: local
    driver_opts:
      o: bind
      type: none
      device: /root/wordpress/db
  wordpress_data: {}
```

REPLICAS:




Red Hat
Open Shift
https://learn.openshift.com/
- Puissant
- Prod
- Cher
- Expérience avancée est "limité"
docker Swarm
Docker EE
Rancher, ...
Kubernetes
-Google
-OpenSource
-gratuit
- Répulé
docker
-> mono-host
cluster de hosts
Pour la HA -> "bien gerdi" -> Pas vraiment fiable en Prod