Utilisation de docker

Objectif

L'objectif de cette documentation est d'apprendre à utiliser les commandes de base de docker ainsi que de créer à partir de celle-ci des images et d'instancier des conteneurs.

Présentation des commandes

Il existe de nombreuse commande sur docker, le liste complète est disponible en utilisant *docker –help* nous allons ici présenter les plus importante pour débuter avec docker.

```
  attach    Entrer dans le conteneur via un shell conteneur
  build     Construire un image depuis un fichier Dockerfile
  commit    Créer une nouvelle image de l'évolution du conteneur
  cp        Copier des fichiers/dossiers entre l'hôte et le conteneur
  create    Créer un nouveau conteneur
  diff      Vérifier les changements sur le conteneur
  events    Obtenir un journal d’événements du service docker
  exec      Exécuter une commande dans le conteneur cible
  history   Voir l'historique d'une image
  images    Lister les images présente sur l'hôte
  logs      Voir les logs du conteneur
  port      Voir la liste des ports redirigés par le proxy docker sur le
conteneur
  ps        Lister les conteneurs démarré (avec l'option -a tous les
conteneurs)
  pull      Télécharger une image depuis un registre
  push      Envoyer une image sur un registre
  restart   Redémarrer un conteneur
  rm        Supprimer un ou plusieurs conteneurs
  rmi       Supprimer une ou plusieurs images
  run       Lancer une commande dans un nouveau conteneur (create + start)
  search    Rechercher une image dans le hub docker
  start     Démarrer un ou plusieurs conteneurs
  stats     Afficher les statistiques d'un ou plusieurs conteneurs
  stop      Arrêter un ou plusieurs conteneurs
  tag       Tager une image dans un registre
  top       Voir les processus en cours d’exécution dans le conteneur
  volume    Gérer les volumes docker
```

Docker build

Avec les différentes commandes que nous avons vue précédemment nous pouvons créer des

Dockerfile qui vont nous permettre de construire une image.

Apache 2 Dockerfile

```
# On definit quel est l'image de base à utiliser
FROM cg44/debian:jessie
# Les personnes qui maintiennent cette image
MAINTAINER Laurent Souchet <hello@viper61.fr>
MAINTAINER Charly Beaupeux

# Les varibles d'environnement
ENV APACHE_RUN_USER=www-data \
    APACHE_RUN_GROUP=www-data \
    APACHE_LOG_DIR=/var/log/apache2 \
    APACHE_LOCK_DIR=/var/lock/apache2 \
    APACHE_PID_FILE=/var/run/apache2.pid

# Les différentes commandes pour installer apache ainsi que les modules
nécessaire
RUN apt-get update -y \
    && apt-get install -y --no-install-recommends apache2 apache2-mpm-worker
apache2-mpm-event libapache2-mod-fcgid \
    && a2dismod mpm_prefork \
    && a2enmod ssl rewrite proxy proxy_fcgi actions alias env headers mime
dir \
# On supprime le cache apt pour alléger l'image
    && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

# On indique à Docker les ports que le conteneur écoute
EXPOSE 80 443

# Dossier de données que l'on rends pérenne
VOLUME ["/etc/apache2/sites-enabled", "/etc/apache2/ssl",
"/var/log/apache2", "/var/www"]

# La commande executer au demarrage du conteneur
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
```

PHP-fpm Dockerfile

```
# On definit quel est l'image de base à utiliser
FROM cg44/debian:jessie
# Les personnes qui maintiennent cette image
MAINTAINER Laurent Souchet <hello@viper61.fr>
MAINTAINER Charly Beaupeux

# Les varibles d'environnement
RUN apt-get update -y && \
```

```
    apt-get install -y --no-install-recommends php5-fpm php5-curl php5-ldap
php5-memcache php5-memcached \
    php5-mssql php5-mysqlnd php5-pgsql php5-sqlite php5-xdebug php5-gd && \
    rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

# Les différentes commandes pour installer php5 ainsi que les modification à
effectuer dans le fichier de configuration
RUN sed -i "s/;date.timezone =.*/date.timezone = Europe\/Paris/"
/etc/php5/fpm/php.ini && \
    sed -i "s/;cgi.fix_pathinfo=1/cgi.fix_pathinfo=1/" /etc/php5/fpm/php.ini
&& \
    sed -i "s/display_errors = Off/display_errors = stderr/"
/etc/php5/fpm/php.ini && \
    sed -i "s/upload_max_filesize = 2M/upload_max_filesize = 1G/"
/etc/php5/fpm/php.ini && \
    sed -i "s/;opcache.enable=0/opcache.enable=0/" /etc/php5/fpm/php.ini &&
\
    sed -i "s/;daemonize\s*=\s*yes/daemonize = no/g" /etc/php5/fpm/php-
fpm.conf && \
    sed -i "/^listen = /clisten = 9000" /etc/php5/fpm/pool.d/www.conf && \
    sed -i "/^listen.allowed_clients/c;listen.allowed_clients ="
/etc/php5/fpm/pool.d/www.conf && \
    sed -i "/^;catch_workers_output/ccatch_workers_output = yes"
/etc/php5/fpm/pool.d/www.conf

# On indique à Docker les ports que le conteneur écoute
EXPOSE 9000

# Dossier de données que l'on rends pérenne
VOLUME ["/var/www"]

# La commande executer au demarrage du conteneur
ENTRYPOINT ["/usr/sbin/php5-fpm", "-F"]
```

Postgres Dockerfile

```
# On definit quel est l'image de base à utiliser
FROM cg44/debian:jessie
# Les personnes qui maintiennent cette image
MAINTAINER Laurent Souchet <hello@viper61.fr>
MAINTAINER Charly Beaupeux

# Les varibles d'environnement
ENV PG_MAJOR=9.4
ENV PG_VERSION=9.4.5-0+deb8u1
ENV PGDATA=/var/lib/postgresql/data
ENV PGDIR=/etc/postgresql/$PG_MAJOR/main
ENV DBUSER=unusual
ENV DBPWD=5uP3r?4s5w02D
ENV DBBASE=anydb
```

```
ENV PATH=/usr/lib/postgresql/$PG_MAJOR/bin:$PATH

# Les différentes commandes pour installer postgres ainsi que les dossier
nécessaire
RUN apt-get update -y \
    && apt-get install -y --no-install-recommends postgresql-
$PG_MAJOR=$PG_VERSION \
    && mkdir -p /var/run/postgresql \
    && chown -R postgres /var/run/postgresql \
# On supprime le cache apt pour alléger l'image
    && rm -rf /var/lib/apt/lists/* /tmp/* /var/tmp/*

# On copie le script dans le conteneur
COPY init.sh /

# On indique à Docker les ports que le conteneur écoute
EXPOSE 5432

# Dossier de données que l'on rends pérenne
VOLUME ["/var/lib/postgresql/data"]

# On change d'utilisateur
USER postgres

# On change le dossier de travail
WORKDIR /var/lib/postgresql

# La commande executer au demarrage du conteneur
ENTRYPOINT ["/init.sh"]
```

Docker run

Apache 2 run.sh

```
#!/bin/bash

docker run -dti -p 80:80 -p 443:443 --link php --name apache -v
/data/apache2/etc/apache2/sites-enabled/:/etc/apache2/sites-enabled/ -v
/data/apache2/etc/apache2/ssl:/etc/apache2/ssl -v
/data/apache2/var/log/apache2:/var/log/apache2 -v
/data/apache2/var/www:/var/www -h apache
debiandockerdev2:5000/cg44/apache2:latest
```

PHP-fpm run.sh

```
#!/bin/bash
```

```
docker run -dtiv /data/apache2/var/www:/var/www --link postgres --name php -
h php debiandockerdev2:5000/cg44/php:latest
```

Postgres run.sh

```
#!/bin/bash

docker run -dtiv
/data/postgres/var/lib/postgresql/data:/var/lib/postgresql/data --name
postgres -h postgres debiandockerdev2:5000/cg44/postgres:latest
```

Ainsi que le script d'initialisation

```
#!/bin/bash

if [[ -z "$(ls -A "$PGDATA")" || "$1" = "master" || "$1" = "slave" ]]
then
  if [ -z "$(ls -A "$PGDATA")" ]
  then
    chmod 700 $PGDATA
    chown -R postgres $PGDATA
    initdb

    sed -i "s,#listen_addresses = 'registry.cg44.fr',listen_addresses =
'*',g" $PGDATA/postgresql.conf
    sed -i "s,#data_directory = 'ConfigDir',data_directory = '$PGDATA',g"
$PGDATA/postgresql.conf
    sed -i "s,#hba_file = 'ConfigDir/pg_hba.conf',hba_file =
'$PGDATA/pg_hba.conf',g" $PGDATA/postgresql.conf
    echo "host    all             all             172.17.0.0/16
md5" >> $PGDATA/pg_hba.conf

    pg_ctl -D $PGDATA -o "-c listen_addresses='*'" -w start

    psql -U postgres << EOF
      create user $DBUSER createdb;
      alter user $DBUSER encrypted password '$DBPWD';
      create database $DBBASE template template0 encoding 'unicode';
      alter database $DBBASE owner to $DBUSER;
      grant all privileges on database $DBBASE to $DBUSER;
EOF

    echo "L'utilisateur et la base de données ont bien été créer et sont
préte !"
  fi

  if [ "$1" = "master" ]
  then
    echo "master"
```

```
    if [ ! $(pgrep postgres) ]
    then
      pg_ctl -D $PGDATA -o "-c listen_addresses='*'" -w start
    fi
    echo "création $REPUSER"
    psql -U postgres << EOF
      create user $REPUSER replication;
      alter user $REPUSER encrypted password '$REPPWD';
EOF

    sed -i "s,#wal_level = minimal,wal_level = hot_standby,g"
$PGDATA/postgresql.conf
    sed -i "s,#max_wal_senders = 0,max_wal_senders = 1,g"
$PGDATA/postgresql.conf

  elif [ "$1" = "slave" ]
  then
    sed -i "s,#hot_standby = off,hot_standby = on,g" $PGDATA/postgresql.conf
    echo "standby_mode = 'on'" > $PGDATA/recovery.conf
    echo "primary_conninfo = 'host=$REPMASTERIP port=5432 user=$REPUSER
password=$REPPWD'" >> $PGDATA/recovery.conf
  fi

else
  postgres -D $PGDATA
fi
```

Scripts

Nous avons réalisé trois scripts permettant de facilité certaines opérations.

Le premier consiste à effacer l'ensemble des images sans tag, lister sous le tag <*none*> :

```
#!/bin/bash
docker rmi $(docker images | grep "^<none>" | awk '{print $3}')
```

Le second nous permet de déclencher l'arrêt de l'ensemble des conteneurs actif sur notre machine :

```
#!/bin/bash
docker stop $(docker ps -a -q)
```

Enfin, le dernier nous permet d'arrêter, détruire puis reconstruire de nouveaux conteneurs pour notre mode “brick” Postegre / PHP / Apache :

```
#! /bin/sh
# Fonction d'arret et suppression
clean() {
  process=$(docker ps -a)
```

```
  echo "Arret du conteneur $1 + suppresion"

  if ( echo $process | grep "$1" > /dev/null ); then
    docker stop $1
    docker rm $1
  fi
}

# Fonction de lancement
start() {
  echo "Start conteneur $1"

  /docker/$1/run.sh

  sleep 5
}

# Nettoyage des conteneurs
clean "postgres"
clean "php"
clean "apache"

# Demarrage des conteneurs
start "postgres"
start "php"
start "apache"
```