TP Docker Serveur WEB

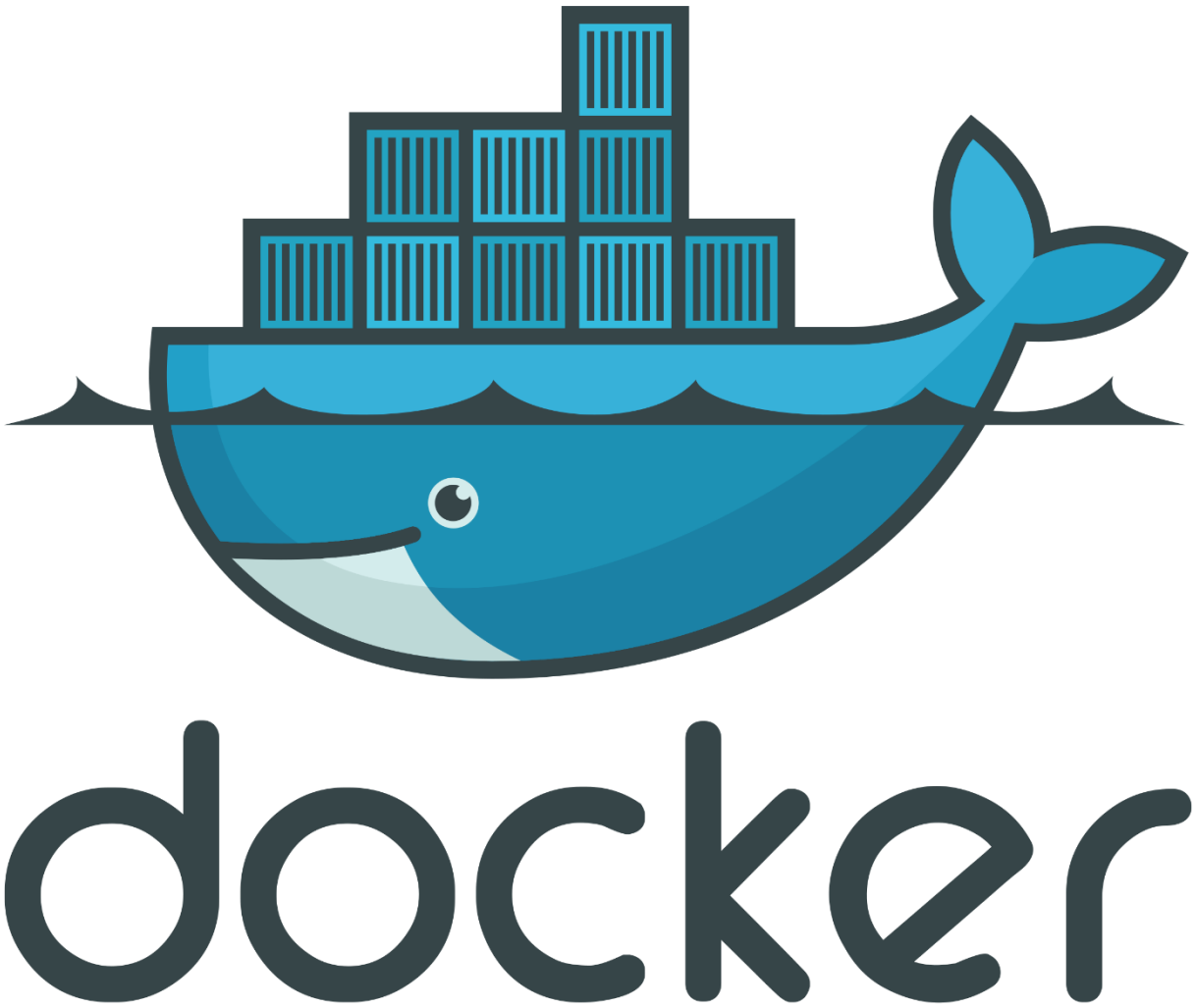


Table des matières :

Prérequis

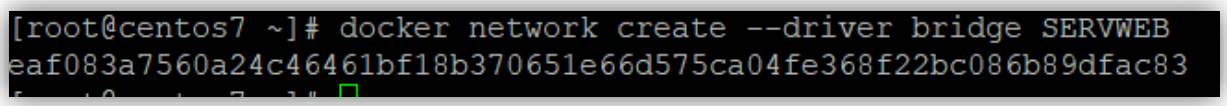
Afin de pouvoir effectuer les manipulation suivantes, il est nécessaire de posséder :

- Une machine virtuelle sous **CentOS 7**, iso disponible ***ici*** (ISO de l'école) ;
- Avoir installé sur notre machine virtuelle docker **version 19.03.13**, voir l'installation de Docker ***ici*** ;
- Installer la commande nano (Vim Editor étant intégré) avec **sudo yum install nano** ;

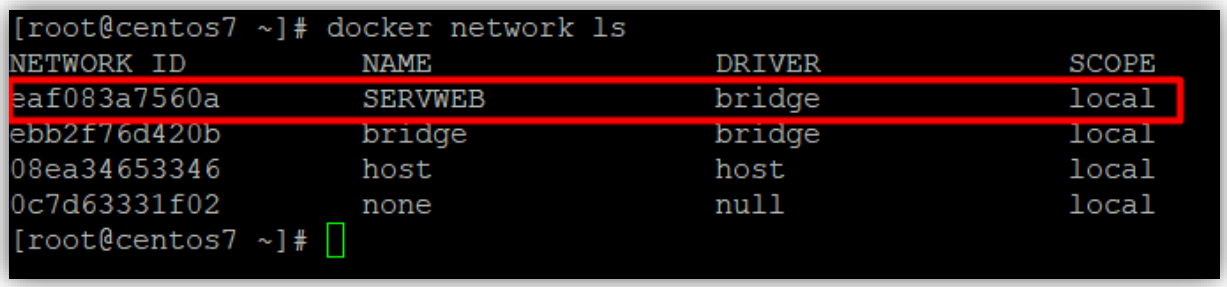
Création du network SERVWEB

Avant de monter nos conteneurs, nous allons les mettre sur un réseau en bridge afin qu'ils puissent communiquer entre eux, pour se faire, on tapera les commandes suivante (sur notre CentOS depuis un invite de commande connecté en SSH à la machine virtuelle) :

- **docker network create --driver bridge SERVWEB** :



- On peut vérifier la création de notre réseau en faisant **docker network ls** :



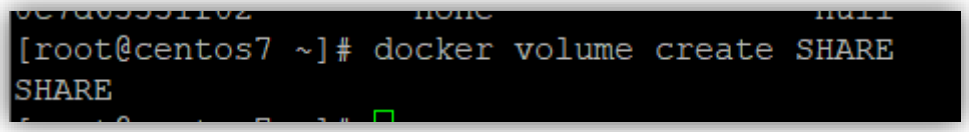
Notre réseau est donc bien créé et opérationnel, avant de créer nos conteneurs, nous allons créer un volume partagé.

Création du volume partagé SHARE

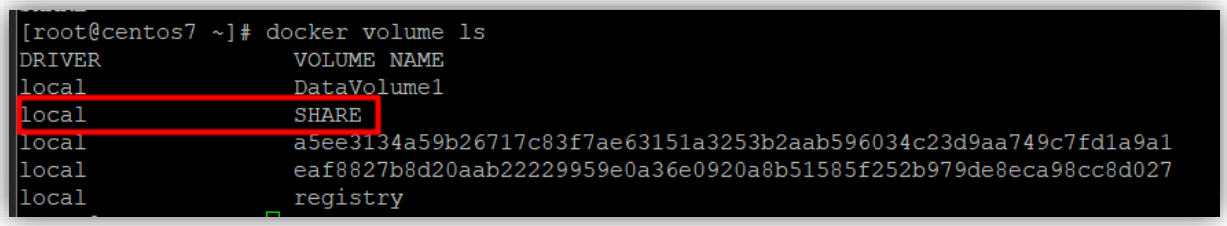
La finalité de la création de ce volume sera que nous mapperons par la suite lors de la création de nos deux conteneurs ce volume afin de pouvoir exécuter ce qui est demander en consigne qui est qu'un container producteur puisse insérer la date périodiquement toutes les heures et que l'autre container puisse l'afficher lorsqu'il est interrogé.

La création du volumes SHARE ce fera avec la commande :

- **docker volume create SHARE** :



- On vérifie que le volume a bien été créé avec la commande **docker volume ls** :



Notre volume est à présent créé, nous allons créer nos conteneurs qui seront sur le réseau SERVWEB créé précédemment avec le volume SHARE mappé à la racine de chaque conteneurs.

Création des conteneurs RUNNEUR et PRODUCTOR

Durant la phase de création de chaque conteneurs, nous allons spécifier les informations suivante qui seront utilisé durant cette phase :

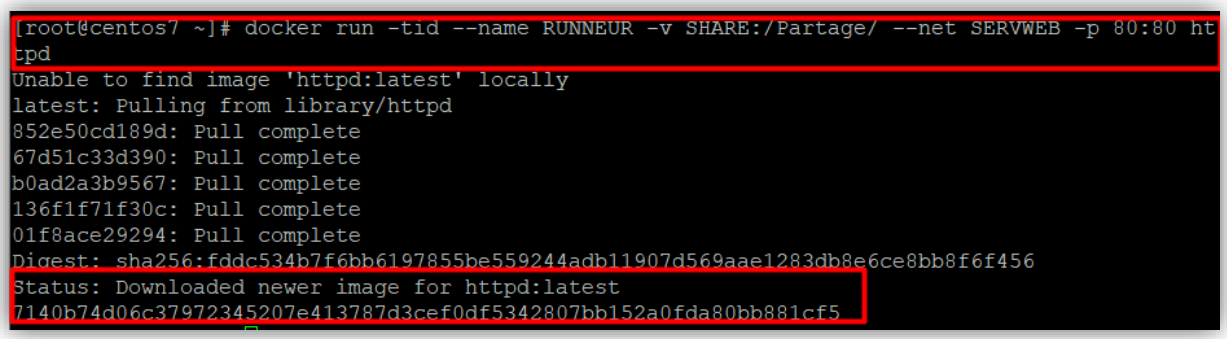
Création du conteneur RUNNEUR

Conteneur WEB avec les options suivantes :

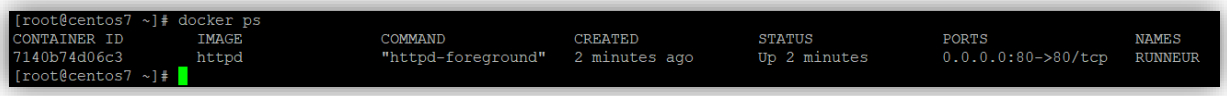
- **-i** pour indiquer à docker de se connecter au **stdin** du conteneur ;
- **-t** qui permet d'obtenir un pseudo-terminal ;
- **-d** qui permet de lancer ce conteneur en mode démon ;
- **--name** qui permet de donner un nom à notre conteneur, dans notre cas, ce sera RUNNEUR ;
- **-v** qui permettra de monter notre volume **SHARE** dans le dossier Partage qui se trouvera à la racine de chaque conteneur ;
- **--net SERVWEB** qui intègrera ce conteneur au réseau SERVWEB créé précédemment ;
- **-p 80:80** pour le port 80 que nous spécifions au conteneur ;
- **Httpd** car il s'agira d'un conteneur sous image apache.

Ce qui nous donne la commande suivante :

- **docker run -tid --name RUNNEUR -v SHARE:/Partage/ --net SERVWEB -p 80:80 httpd**



Nous pouvons vérifier que notre conteneur est bien lancé avec la commande **docker ps**



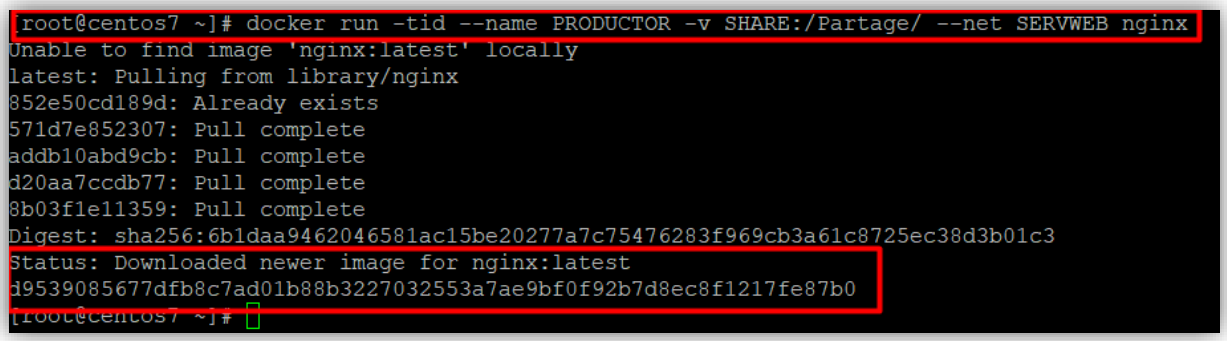
Création du conteneur PRODUCTOR

Conteneur PRODUCTOR avec les options suivantes :

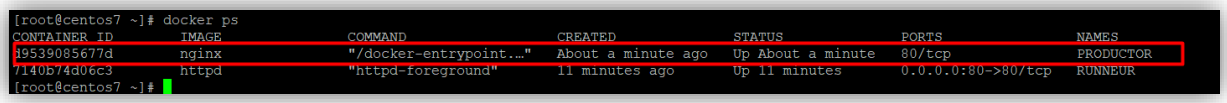
- **-i** pour indiquer à docker de se connecter au **stdin** du conteneur ;
- **-t** qui permet d'obtenir un pseudo-terminal ;
- **-d** qui permet de lancer ce conteneur en mode démon ;
- **--name** qui permet de donner un nom à notre conteneur, dans notre cas, ce sera PRODUCTOR ;
- **-v** qui permettra de monter notre volume **SHARE** dans le dossier Partage qui se trouvera à la racine de chaque conteneur ;
- **--net SERVWEB** qui intègrera ce conteneur au réseau SERVWEB créé précédemment ;
- **nginx** car il s'agira d'un conteneur sous image nginx.

Ce qui nous donne la commande suivante :

- **docker run -tid --name PRODUCTOR -v SHARE:/Partage/ --net SERVWEB nginx**



On vérifie à présent si le conteneur est lancé avec un **docker ps** :

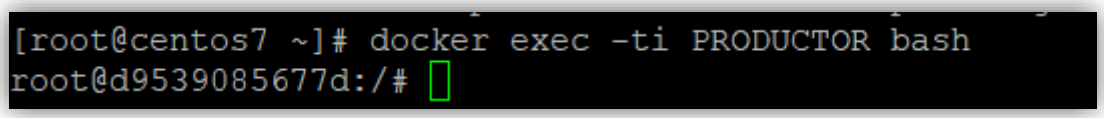


On peut voir que notre conteneur est créé et à présent nous avons nos conteneurs RUNNEUR et PRODUCTOR, nous allons à présent nous connecter sur chaque conteneurs et les mettre à jour pour la suite des opérations.

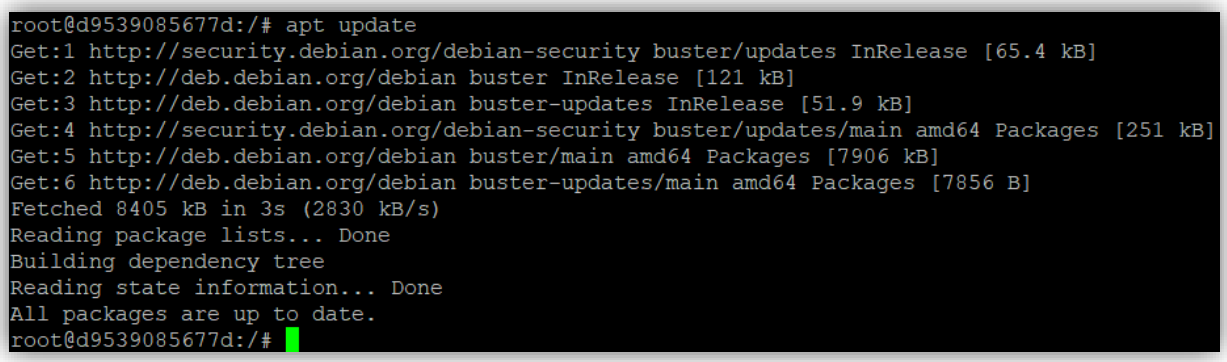
Mise à jour des conteneurs + installation de nano et ping

Afin de mettre à jour les conteneurs, nous allons nous connecter dessus en faisant la commande suivante :

- **docker exec -ti PRODUCTOR bash**



On peut voir que nous sommes connectés à notre conteneur PRODUCTOR, nous allons à présent le mettre à jour avec la commande **apt update** :



Nous allons à présent installer nano sur le conteneur avec la commande **apt install nano** :

```
root@d9539085677d:/# apt install nano
Reading package lists... Done
Building dependency tree
Reading state information... Done
Suggested packages:
  spell
The following NEW packages will be installed:
  nano
0 upgraded, 1 newly installed, 0 to remove and 0 not upgraded.
Need to get 544 kB of archives.
After this operation, 2269 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian buster/main amd64 nano amd64 3.2-3 [544 kB]
Fetched 544 kB in 0s (3855 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package nano.
(Reading database ... 7631 files and directories currently installed.)
Preparing to unpack .../archives/nano_3.2-3_amd64.deb ...
Unpacking nano (3.2-3) ...
Setting up nano (3.2-3) ...
update-alternatives: using /bin/nano to provide /usr/bin/editor (editor) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/editor.1.gz because associ
ated file /usr/share/man/man1/nano.1.gz (of link group editor) doesn't exist
update-alternatives: using /bin/nano to provide /usr/bin/pico (pico) in auto mode
update-alternatives: warning: skip creation of /usr/share/man/man1/pico.1.gz because associat
ed file /usr/share/man/man1/nano.1.gz (of link group pico) doesn't exist
root@d9539085677d:/#
```

Nous allons installer la commande ping avec la commande suivante :

- **apt install iputils-ping** et nous mettrons **y** pour valider l'installation :

```
root@d9539085677d:/# apt install iputils-ping
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcap2 libcap2-bin libpam-cap
The following NEW packages will be installed:
  iputils-ping libcap2 libcap2-bin libpam-cap
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 104 kB of archives.
After this operation, 319 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

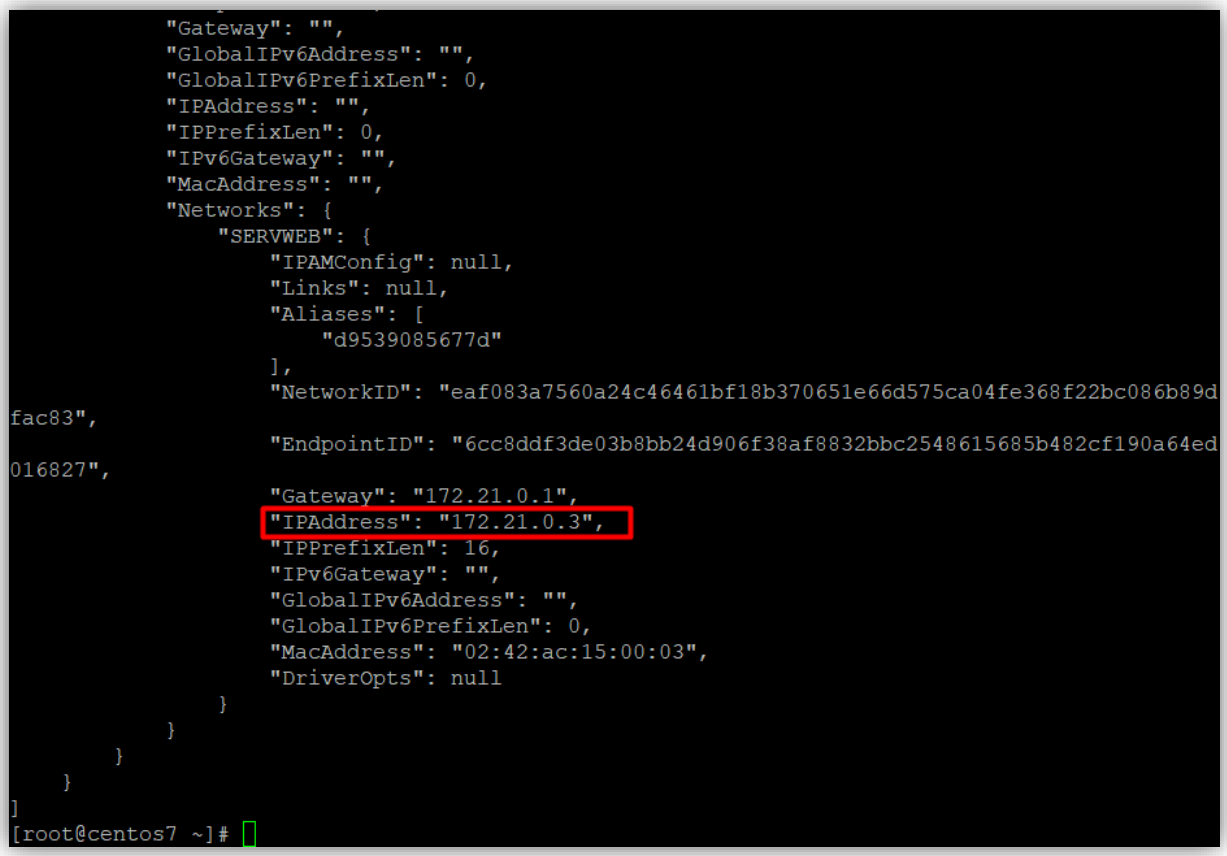
```
Do you want to continue? [Y/n] y
Get:1 http://deb.debian.org/debian buster/main amd64 libcap2 amd64 1:2.25-2 [17.6 kB]
Get:2 http://deb.debian.org/debian buster/main amd64 iputils-ping amd64 3:20180629-2+deb10u1
[43.3 kB]
Get:3 http://deb.debian.org/debian buster/main amd64 libcap2-bin amd64 1:2.25-2 [28.8 kB]
Get:4 http://deb.debian.org/debian buster/main amd64 libpam-cap amd64 1:2.25-2 [14.3 kB]
Fetched 104 kB in 0s (1309 kB/s)
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package libcap2:amd64.
(Reading database ... 7733 files and directories currently installed.)
Preparing to unpack .../libcap2_1%3a2.25-2_amd64.deb ...
Unpacking libcap2:amd64 (1:2.25-2) ...
Selecting previously unselected package iputils-ping.
Preparing to unpack .../iputils-ping_3%3a20180629-2+deb10u1_amd64.deb ...
Unpacking iputils-ping (3:20180629-2+deb10u1) ...
Selecting previously unselected package libcap2-bin.
Preparing to unpack .../libcap2-bin_1%3a2.25-2_amd64.deb ...
Unpacking libcap2-bin (1:2.25-2) ...
Selecting previously unselected package libpam-cap:amd64.
Preparing to unpack .../libpam-cap_1%3a2.25-2_amd64.deb ...
Unpacking libpam-cap:amd64 (1:2.25-2) ...
Setting up libcap2:amd64 (1:2.25-2) ...
Setting up libcap2-bin (1:2.25-2) ...
Setting up libpam-cap:amd64 (1:2.25-2) ...
debconf: unable to initialize frontend: Dialog
debconf: (No usable dialog-like program is installed, so the dialog based frontend cannot be
used. at /usr/share/perl5/Debconf/FrontEnd/Dialog.pm line 76.)
debconf: falling back to frontend: Readline
debconf: unable to initialize frontend: Readline
debconf: (Can't locate Term/ReadLine.pm in @INC (you may need to install the Term::ReadLine m
odule) (@INC contains: /etc/perl /usr/local/lib/x86_64-linux-gnu/perl/5.28.1 /usr/local/share
/perl/5.28.1 /usr/lib/x86_64-linux-gnu/perl5/5.28 /usr/share/perl5 /usr/lib/x86_64-linux-gnu/
perl/5.28 /usr/share/perl/5.28 /usr/local/lib/site_perl /usr/lib/x86_64-linux-gnu/perl-base)
at /usr/share/perl5/Debconf/FrontEnd/Readline.pm line 7.)
debconf: falling back to frontend: Teletype
Setting up iputils-ping (3:20180629-2+deb10u1) ...
Processing triggers for libc-bin (2.28-10) ...
root@d9539085677d:/#
```

Nous ferons les mêmes manipulations avec le conteneur RUNNEUR.

Une fois ces prérequis effectué, nous allons tester la connectivité entre nos deux conteneurs pour voir s'ils communiquent entre eux.

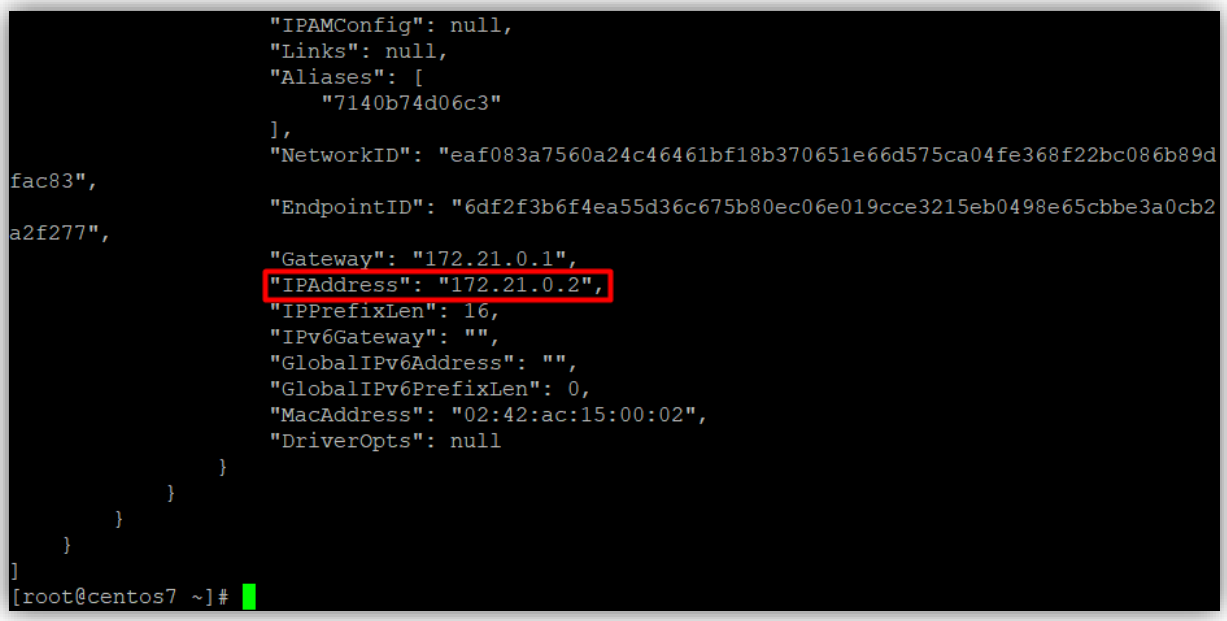
Vérification connectivité entre nos deux conteneurs

Nous allons obtenir l'adresse IP de chacun de nos conteneurs avec la commande **docker inspect PRODUCTOR** ce qui nous retournera un message de ce type :



Notre conteneur PRODUCTOR a comme adresse IP **172.21.0.3**.

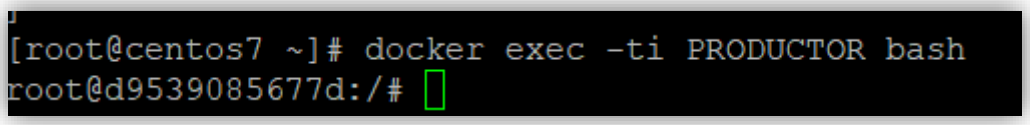
On fera la même opération sur le conteneur RUNNEUR, ce qui nous retournera le message suivant :



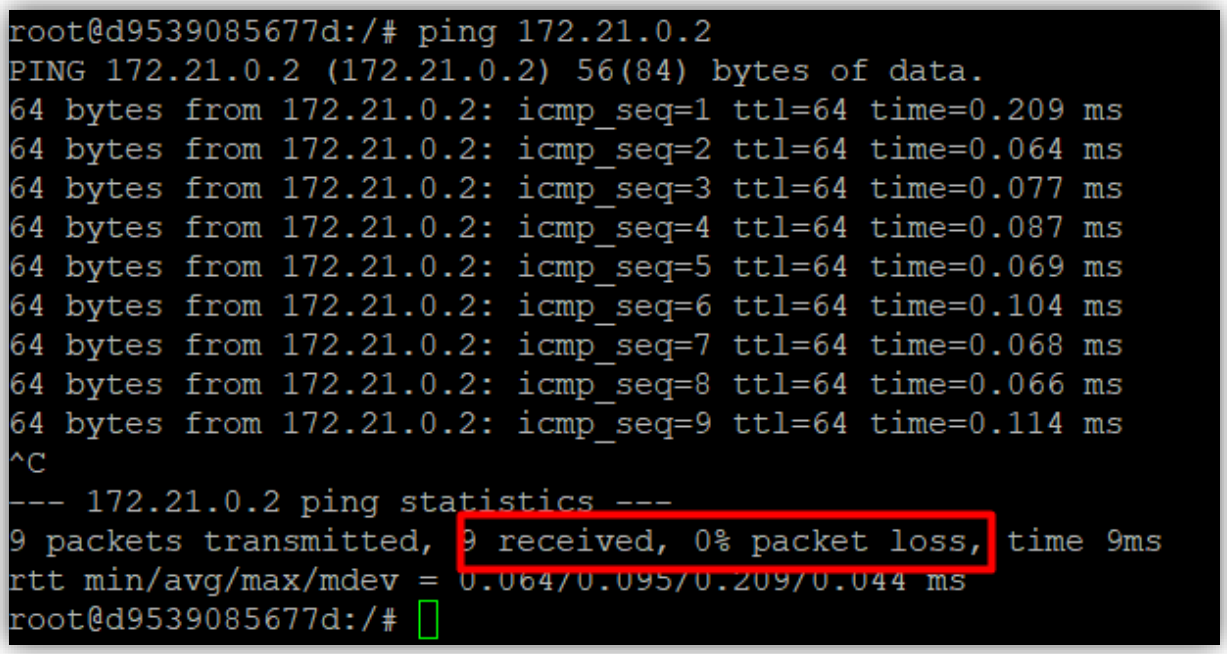
Notre conteneur RUNNEUR a comme adresse **172.21.0.2**.

On se connecte à notre conteneur PRODUCTOR avec la commande suivante :

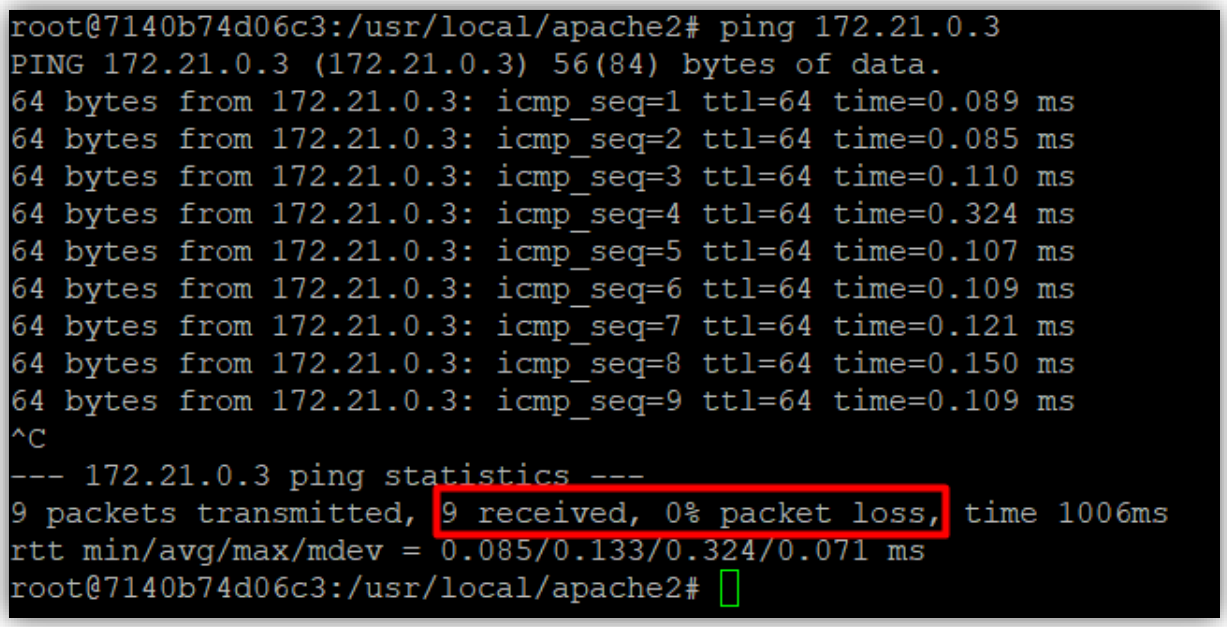
- **docker exec -ti PRODUCTOR bash**



Nous allons vérifier que notre conteneur PRODUCTOR détecte notre conteneur RUNNEUR avec la commande **ping 172.21.0.2**



Nous voyons que notre requête ping communique bien avec notre conteneur RUNNEUR, nous ferons les mêmes opération sur le conteneur RUNNEUR avec **ping 172.21.0.3** :

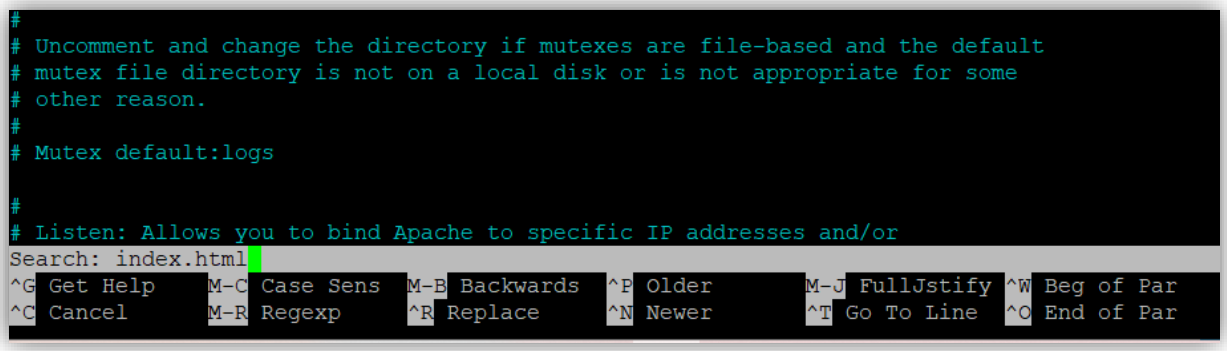


Nos conteneurs communiquent bien entre eux à présent, nous allons procéder au paramétrage de redirection vers le fichier index.html sur notre conteneur apache.

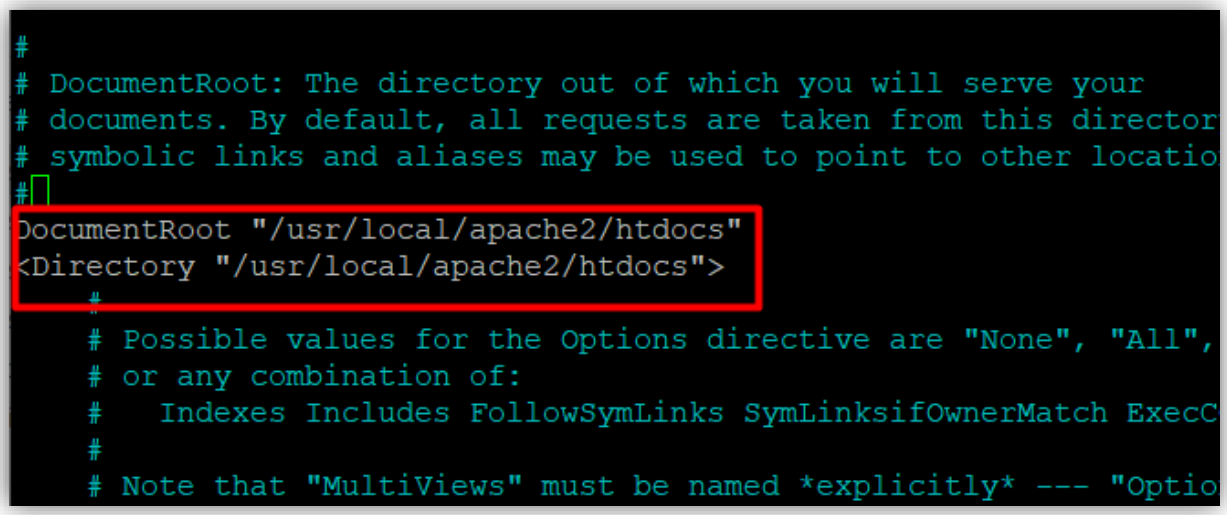
Configuration de l'index.html

Nous nous rendons sur notre conteneur RUNNEUR à l'aide de **docker exec -it RUNNEUR bash**.

Nous allons éditer le fichier **httpd.conf** en faisant **nano /usr/local/apache2/conf/httpd.conf** et on fera un **CTRL + W** et on recherchera les lignes contenant **index.html** puis nous appuierons sur entré :



Ce que l'on modifiera sera les champs **DocumentRoot et <Directory […]>** :



On y mettra notre chemin vers notre volume partagé ce qui revient à faire comme sur la capture ci-dessous :

- **DocumentRoot "/Partage"**
- **<Directory "/Partage">**

```
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "/Partage"
<Directory "/Partage">
    #
    # Possible values for the Options directive are "None", "All",
    # or any combination of:
    #   Indexes Includes FollowSymLinks SymLinksifOwnerMatch ExecCGI MultiViews
    #
    # Note that "MultiViews" must be named *explicitly* --- "Options All"
    # doesn't give it to you.
    #
    # The Options directive is both complicated and important.  Please see
```

On fera ensuite un **CTRL + X → Y → Entrée** pour que les modifications soient sauvegardées.

Pour que les modifications soient appliquées, on sortira du conteneur en faisant un **exit** et on fera un **docker stop RUNNEUR** puis un **docker start RUNNEUR**.

Notre apache est maintenant configuré pour récupéré le fichier index.html qui sera alimenté par notre conteneur PRODUCTOR et qui se trouvera au chemin **/Partage** qui correspond à notre volume créé précédemment.

Nous allons désormais créer notre script d'ajout de date dans le fichier index.html qui se situera sur **/Partage** et qui sera joué toute les heures.

Initialisation du fichier date.sh

Une fois sur le conteneur PRODUCTOR, nous nous rendons dans le dossier /opt qui contiendra notre script avec la commande **cd /opt**

```
root@d9539085677d:/# cd /opt
root@d9539085677d:/opt#
```

On fera un **nano date.sh** et on y mettra le texte suivant :

#!/bin/bash
echo "" > /Partage/index.html
date >> /Partage/index.html

```
#!/bin/bash
echo "" > /Partage/index.html
date >> /Partage/index.html
```

Puis on enregistre notre fichier avec **CTRL + X → Y → Entrée** pour que les modifications soient prises en compte.

On peut vérifier le contenu de notre fichier **date.sh** en faisant un **cat date.sh** :

```
root@d9539085677d:/opt# cat date.sh
#!/bin/bash
echo "" > /Partage/index.html
date >> /Partage/index.html
root@d9539085677d:/opt#
```

On donnera des droits spécifique à notre **date.sh** avec la commande **chmod a+x date.sh**.

Notre script **date.sh** est à présent initialisé, nous allons désormais installer **Cron** sur notre conteneur PRODUCTOR.

Installation et configuration de CRON sur PRODUCTOR

On se connecte sur PRODUCTOR avec **docker exec -it PRODUCTOR bash** puis on fait la commande suivante pour installer cron :

- **apt install cron** et on tape **Y** puis **entrée** afin de valider l'installation

```
root@d9539085677d:/# apt install cron
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  bzip2 exim4-base exim4-config exim4-daemon-light file guile-2.2-libs libevent-2.1-6
  libfribidi0 libgc1c2 libgnutls-dane0 libgpm2 libgsasl7 libidn11 libkyotocabinet16v5
  libltdl7 liblzo2-2 libmagic-mgc libmagic1 libmailutils5 libmariadb3 libncurses6 libntlm0
  libpython2.7 libpython2.7-minimal libpython2.7-stdlib libsqlite3-0 libunbound8 libwrap0
  mailutils mailutils-common mariadb-common mime-support mysql-common netbase psmisc
  xz-utils
Suggested packages:
  bzip2-doc anacron logrotate checksecurity exim4-doc-html | exim4-doc-info eximon4
  spf-tools-perl swaks dns-root-data gpm mailutils-mh mailutils-doc
The following NEW packages will be installed:
  bzip2 cron exim4-base exim4-config exim4-daemon-light file guile-2.2-libs libevent-2.1-6
  libfribidi0 libgc1c2 libgnutls-dane0 libgpm2 libgsasl7 libidn11 libkyotocabinet16v5
  libltdl7 liblzo2-2 libmagic-mgc libmagic1 libmailutils5 libmariadb3 libncurses6 libntlm0
  libpython2.7 libpython2.7-minimal libpython2.7-stdlib libsqlite3-0 libunbound8 libwrap0
  mailutils mailutils-common mariadb-common mime-support mysql-common netbase psmisc
  xz-utils
0 upgraded, 37 newly installed, 0 to remove and 0 not upgraded.
Need to get 16.8 MB of archives.
After this operation, 85.0 MB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Nous allons paramétrer cron avec la commande **crontab -e** puis nous mettrons à la fin du fichier les paramètres pour que le job d'exécution du script **date.sh** puisse s'effectuer toute les heures :

- **0 */1 * * * /bin/sh /opt/date.sh**

```
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').
#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h  dom mon dow   command
0 */1 * * * /bin/sh /opt/date.sh
```

On enregistre nos modifications avec **CTRL + X → Y → Entrée**.

Afin que le job soit pris en compte et actif, nous allons démarrer le service **cron** avec la commande **service cron restart**

```
root@d9539085677d:/Partage# service cron restart
[ ok ] Restarting periodic command scheduler: cron[....] Stopping periodic command scheduler: cron.
[ ok ] Starting periodic command scheduler: cron.
```

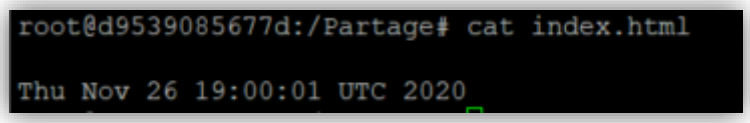
Nous allons à présent vérifier que le script s'exécute bien depuis notre conteneur RUNNEUR.

Vérification du bon fonctionnement du script

Nous allons avoir 2 méthodes pour vérifier que notre script se joue bien et que l'index.html est bien actualisé à la date du jour.

Méthode 1 :

Se connecter sur le conteneur RUNNEUR et se rendre dans le dossier /Partage avec la commande **cd /Partage** et on fera un **cat index.html** :



Méthode 2 :

Se connecter sur le conteneur RUNNEUR et faire un **curl 172.21.0.2** :