

Documentation Serveur Base de données

1-L'Installation de Docker

D'abord on met à jour les paquets avec : `sudo apt-get update`

Puis on installe les paquets nécessaires pour utiliser le dépôt APT via HTTPS avec :

`sudo apt-get install apt-transport-https ca-certificates curl software-properties-common`

```
root@debian:/home/joel#  
root@debian:/home/joel# sudo apt-get update  
Atteint :1 http://security.debian.org/debian-security bookworm-security InRelease  
Atteint :2 http://deb.debian.org/debian bookworm InRelease  
Atteint :3 http://deb.debian.org/debian bookworm-updates InRelease  
Lecture des listes de paquets... Fait  
root@debian:/home/joel#  
root@debian:/home/joel# sudo apt-get install apt-transport-https ca-certificates curl gnupg2 software-properties-common  
Lecture des listes de paquets... Fait  
Construction de l'arbre des dépendances... Fait  
Lecture des informations d'état... Fait  
ca-certificates est déjà la version la plus récente (20230311).  
curl est déjà la version la plus récente (7.88.1-10+deb12u5).  
software-properties-common est déjà la version la plus récente (0.99.30-4).  
software-properties-common passé en « installé manuellement ».  
Les NOUVEAUX paquets suivants seront installés :  
  apt-transport-https gnupg2
```

Enfin on installe Docker avec : `sudo apt-get install docker-ce docker-ce-cli containerd.io`

```
root@debian:/home/joel#  
root@debian:/home/joel# sudo apt-get install docker-ce docker-ce-cli containerd.io  
Lecture des listes de paquets... Fait  
Construction de l'arbre des dépendances... Fait  
Lecture des informations d'état... Fait  
Les paquets supplémentaires suivants seront installés :  
  docker-buildx-plugin docker-ce-rootless-extras docker-compose-plugin git git-man iptables liberror-perl libipset2 libslirp patch pigz slurp4netns  
Paquets suggérés :  
  aufs-tools cgroupfs-mount | cgroup-lite git-daemon-run | git-daemon-sysvinit git-doc git-email git-gui gitk gitweb git-cvs git-mediawiki git-svn firewallld ed diffutils-doc  
Les NOUVEAUX paquets suivants seront installés :  
  containerd.io docker-buildx-plugin docker-ce docker-ce-cli docker-ce-rootless-extras docker-compose-plugin git git-man iptables liberror-perl libipset2 libslirp patch pigz slurp4netns  
0 mis à jour, 15 nouvellement installés, 0 à enlever et 0 non mis à jour.  
Il est nécessaire de prendre 131 Mo dans les archives.  
Après cette opération, 485 Mo d'espace disque supplémentaires seront utilisés.
```

Pour vérifier que Docker est bien présent sur notre serveur on va utiliser :

`Docker run hello-world` ça nous retourne `Hello from Docker!` Donc tout va bien.

```
root@debian:/home/joel#  
root@debian:/home/joel# docker run hello-world  
Unable to find image 'hello-world:latest' locally  
latest: Pulling from library/hello-world  
c1ec1eb5944: Pull complete  
Digest: sha256:d18eb08a1f859111bdf2b3ed698489c41046cb9dd6d1743e37ef8d9f3dda0ef  
Status: Downloaded newer image for hello-world:latest  
  
Hello from Docker!  
This message shows that your installation appears to be working correctly.  
  
To generate this message, Docker took the following steps:  
1. The Docker client contacted the Docker daemon.  
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
3. The Docker daemon created a new container from that image which runs the  
4. The Docker daemon streamed that output to the Docker client, which sent it  
  
To try something more ambitious, you can run an Ubuntu container with:  
$ docker run -it ubuntu bash  
  
Share images, automate workflows, and more with a free Docker ID:  
https://hub.docker.com/  
  
For more examples and ideas, visit:  
https://docs.docker.com/get-started/  
root@debian:/home/joel#
```

On va programmer le démarrage automatique de Docker chaque fois le serveur se rallume :

```
Paramétrage de docker-ce (5.26.1-4-1-debian.12-bookworm) ...
Created symlink /etc/systemd/system/multi-user.target.wants/docker.service -> /lib/systemd/system/docker.service.
Created symlink /etc/systemd/system/sockets.target.wants/docker.socket -> /lib/systemd/system/docker.socket.
Paramétrage de git (1:2.39.2-1.1) ...
Traitement des actions différées (« triggers ») pour man-db (2.11.2-2) ...
Traitement des actions différées (« triggers ») pour libc-bin (2.36-9+deb12u7) ...
root@debian:/home/joel#
root@debian:/home/joel#
root@debian:/home/joel# sudo systemctl enable docker
Synchronizing state of docker.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable docker
root@debian:/home/joel# sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Thu 2024-06-13 20:36:52 CEST; 1min 21s ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
     Main PID: 4958 (dockerd)
        Tasks: 8
       Memory: 41.8M
          CPU: 933ms
      CGroup: /system.slice/docker.service
              └─4958 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

juin 13 20:36:50 debian systemd[1]: Starting docker.service - Docker Application Container Engine...
juin 13 20:36:50 debian dockerd[4958]: time="2024-06-13T20:36:50.7114201+02:00" level=info msg="Starting up"
juin 13 20:36:51 debian dockerd[4958]: time="2024-06-13T20:36:51.11713997+02:00" level=info msg="Loading containers: start."
juin 13 20:36:52 debian dockerd[4958]: time="2024-06-13T20:36:52.155492348+02:00" level=info msg="Loading containers: done."
juin 13 20:36:52 debian dockerd[4958]: time="2024-06-13T20:36:52.313976413+02:00" level=info msg="Docker daemon" commit=de9c4f containerd-snapshotter=false storage-driver=overlay2 version=26.1.4
juin 13 20:36:52 debian dockerd[4958]: time="2024-06-13T20:36:52.314566219+02:00" level=info msg="Daemon has completed initialization"
juin 13 20:36:52 debian dockerd[4958]: time="2024-06-13T20:36:52.423593357+02:00" level=info msg="API listen on /run/docker.sock"
juin 13 20:36:52 debian systemd[1]: Started docker.service - Docker Application Container Engine.
root@debian:/home/joel#
```

2-Création et installation du conteneur PostgreSQL :

Avec **sudo docker pull postgres** on va télécharger l'image de postgresql dans docker

```
root@debian:/home/joel#
root@debian:/home/joel#
root@debian:/home/joel# sudo docker pull postgres
Using default tag: latest
latest: Pulling from library/postgres
09f37deb190: Pull complete
119215dfb3e3: Pull complete
e02bbcc80292: Extracting [> ] 65.54kB/4.534MB
0e1f31083c55: Download complete
acc44983f49a: Download complete
2016f4f0e69a: Download complete
088e651d7f69: Download complete
ed155773e5e0: Download complete
f7ebb35d2984: Downloading [=====] 80.55MB/109MB
293f6bec643a: Download complete
1655a257a5b5: Download complete
4ddba450499d: Download complete
90e48ae83559: Download complete
822c1a513e0a: Download complete

```

Puis on va créer et lancer un conteneur tout en créant un utilisateur en lui assignant un mot de passe :

```
root@debian:/home/joel#
root@debian:/home/joel# sudo docker run --name yplanningDB -e POSTGRES_USER=joel -e POSTGRES_PASSWORD=parynov -d postgres
46d9b9510367b53dfab421e85d9c9be638a0bf8144f6bdac3387348f0d4b
root@debian:/home/joel# sudo docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED        STATUS        PORTS          NAMES
46d9b951036   postgres "docker-entrypoint.sh"   30 seconds ago Up 20 seconds 5432/tcp       yplanningDB
1e8d3208a0e   hello-world "/hello"                58 minutes ago Exited (0) 58 minutes ago          relaxed_rhodes
root@debian:/home/joel#
```

Dans notre cas l'utilisateur est 'joel' et le mot de passe est 'parynov' et le conteneur est 'yplanningDB'

On va aussi programmer le démarrage automatique de notre conteneur :

```
root@debian:/home/joel#
root@debian:/home/joel# docker update --restart always yplanningDB
yplanningDB
root@debian:/home/joel#
root@debian:/home/joel#
```

3- Création de la base de données :

On se connecte à notre conteneur : **sudo docker exec -it yplanningDB bash**

Puis on se connecte à Docker : **psql -U joel -d postgres**

Une fois connecté à Docker on crée notre base de données :

```
joel=# Create DATABASE yplanning;
CREATE DATABASE
joel=#
joel=#
```

Sur notre bureau nous avons pris le soin de créer notre script de base de données nous allons copier le script dans notre conteneur :

```
root@debian:/home/joel# sudo docker cp Bureau/script.sql yplanning08:/script.sql
Successfully copied 3.58kB to yplanning08:/script.sql
root@debian:/home/joel#
```

Nous allons nous connecter et exécuter notre script

La sortie 'CREATE TABLE' nous prouve que notre script existe bel et bien ;

Puis à la fin on se deconnecte

```
root@debian:/home/joel#
root@debian:/home/joel# docker exec -it yplanning08 bash
root@646d9b951036:/# psql -U joel -d yplanning
psql (16.3 (Debian 16.3-1.pgdg120+1))
Type "help" for help.

yplanning=# \i /script.sql
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
CREATE TABLE
yplanning=# \q
root@646d9b951036:/# exit
exit
root@debian:/home/joel#
```

4 : Mise en place des sauvegardes automatisée :

Nous allons mettre en place les sauvegardes automatisées :

D'abord on fait notre script de sauvegarde

```
#!/bin/bash

# Variables
BACKUP_DIR="/backups"
BACKUP_FILE="yplan_backup_${date +%Y/%m/%d/%H/%M/%S}.sql"
CONTAINER_NAME="yplanning08"
DB_USER="joel"
DB_NAME="yplanning"

# Créer le répertoire de sauvegarde s'il n'existe pas
mkdir -p ${BACKUP_DIR}

# Exécuter la commande pg_dump dans le conteneur Docker
sudo docker exec ${CONTAINER_NAME} pg_dump -U ${DB_USER} ${DB_NAME} > ${BACKUP_DIR}/${BACKUP_FILE}

# Optionnel : supprimer les sauvegardes anciennes (plus de 7 jours)
find ${BACKUP_DIR} -type f -name "*.sql" -mtime +7 -exec rm {} \;

echo "Sauvegarde effectuée: ${BACKUP_DIR}/${BACKUP_FILE}"
```

Puis on lui rajoute les droits d'exécution au script :

```
root@debian:/home/joel# chmod +x Bureau/backup.sh
root@debian:/home/joel#
```

Avec Crontab on va automatiser l'exécution du script de sauvegarde en utilisant la commande :

sudo crontab -e

ensuite on va rajouter une entrée cron pour exécuter le script tous les jours à 1h du matin :

```
# 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 (dme), 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.tar.gz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
0 1 * * * /Buresu/backup.sh
```