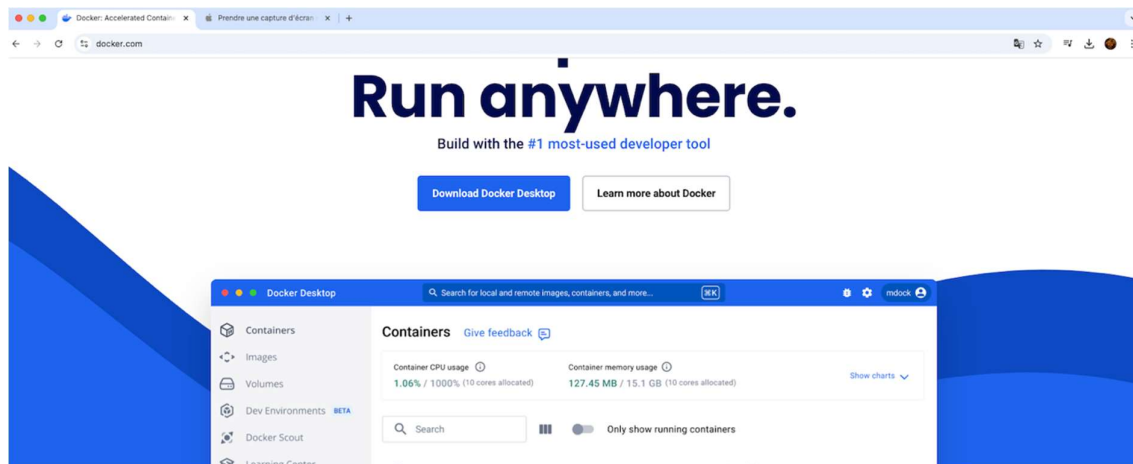


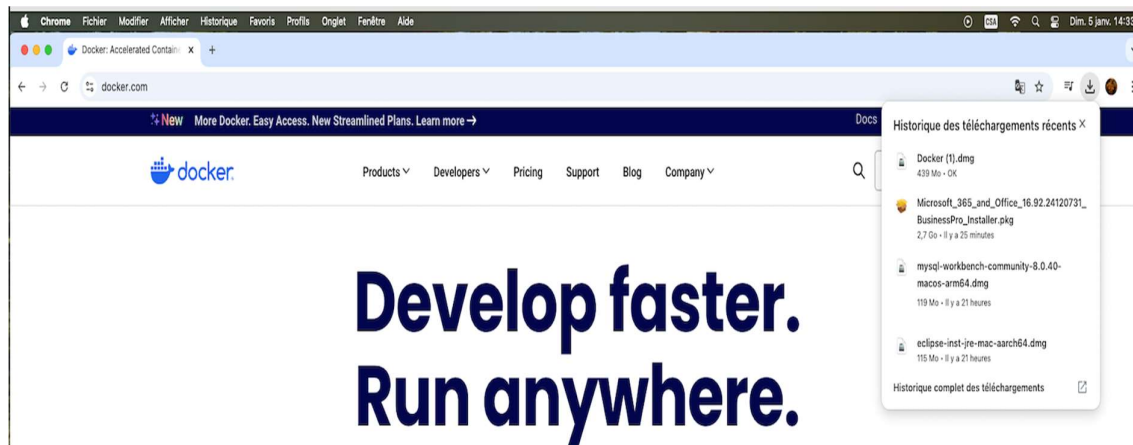
1. Installation de Docker sur MacOS :

● Téléchargement et installation de docker sur Mac :

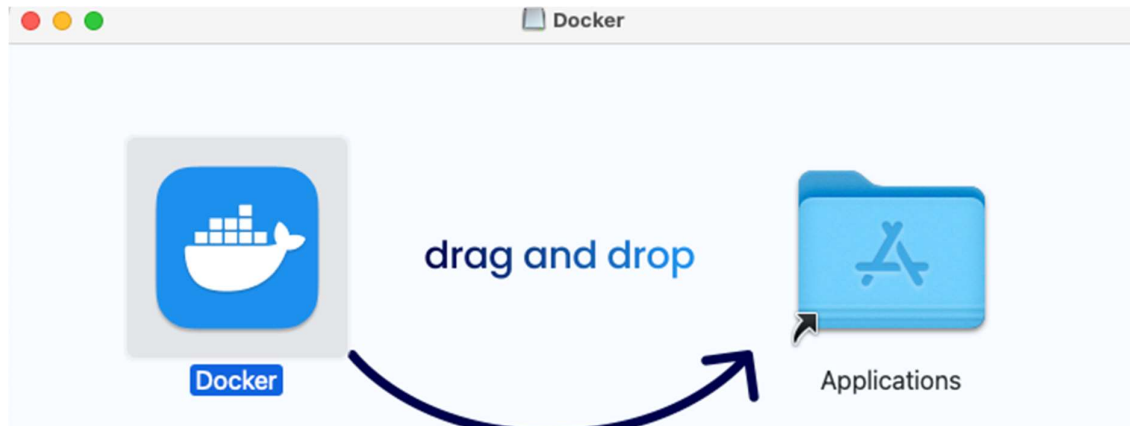
1. 1 Tout d'abord, il faudrait se connecter sur le site docker.com.



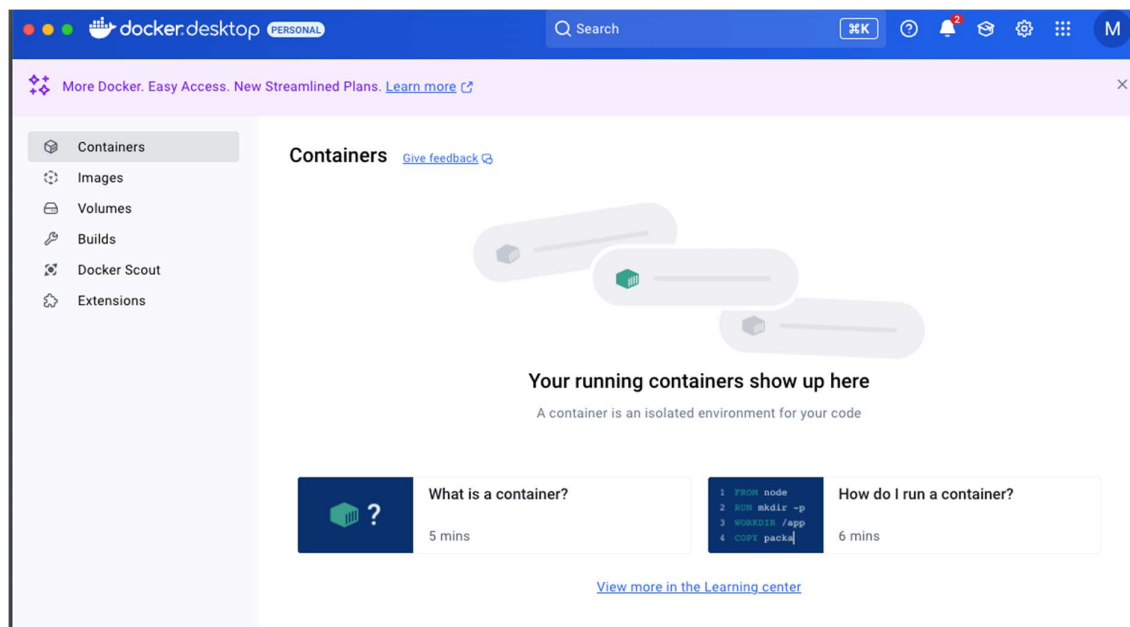
1. 2 Ensuite, il faudra télécharger la version compatible avec Mac (sélectionnez le fichier à télécharger en fonction de la nature du processeur).



1. 3 Après, il faudra lancer Docker et tester son fonctionnement (l'intégrer d'abord à l'application, comme montré ci-dessous).



1. 4 Enfin, lancez l'application Docker Desktop (connectez-vous à votre compte pour vous authentifier).



1. 5 Afin de valider le fonctionnement, une vérification de la version de Docker est nécessaire (version 27.3.1). Comme indiqué ci-dessous, depuis l'invite de commande de Mac, exécutez la commande `docker --version`.

```
ahmedfouadi@Macmini-de-ahmed ~ % docker --version
Docker version 27.3.1, build ce12230
ahmedfouadi@Macmini-de-ahmed ~ %
```

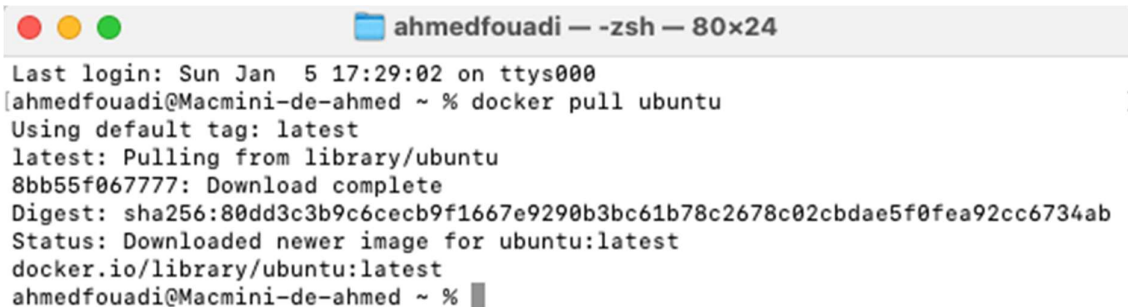
- 2 Installation d'un système exploitation linux sur docker.

2.1 Il faudra ouvrir un terminal sous Mac.



```
ahmedfouadi — -zsh — 80x24
Last login: Sun Jan  5 17:29:02 on ttys000
ahmedfouadi@Macmini-de-ahmed ~ %
```

- 2.2 Afin d'installer un conteneur Linux, il faudra télécharger cette version via la commande `docker pull ubuntu`. Cette commande se connecte au repository des images et télécharge localement le conteneur, qui pourra être utilisé avec Docker Desktop.

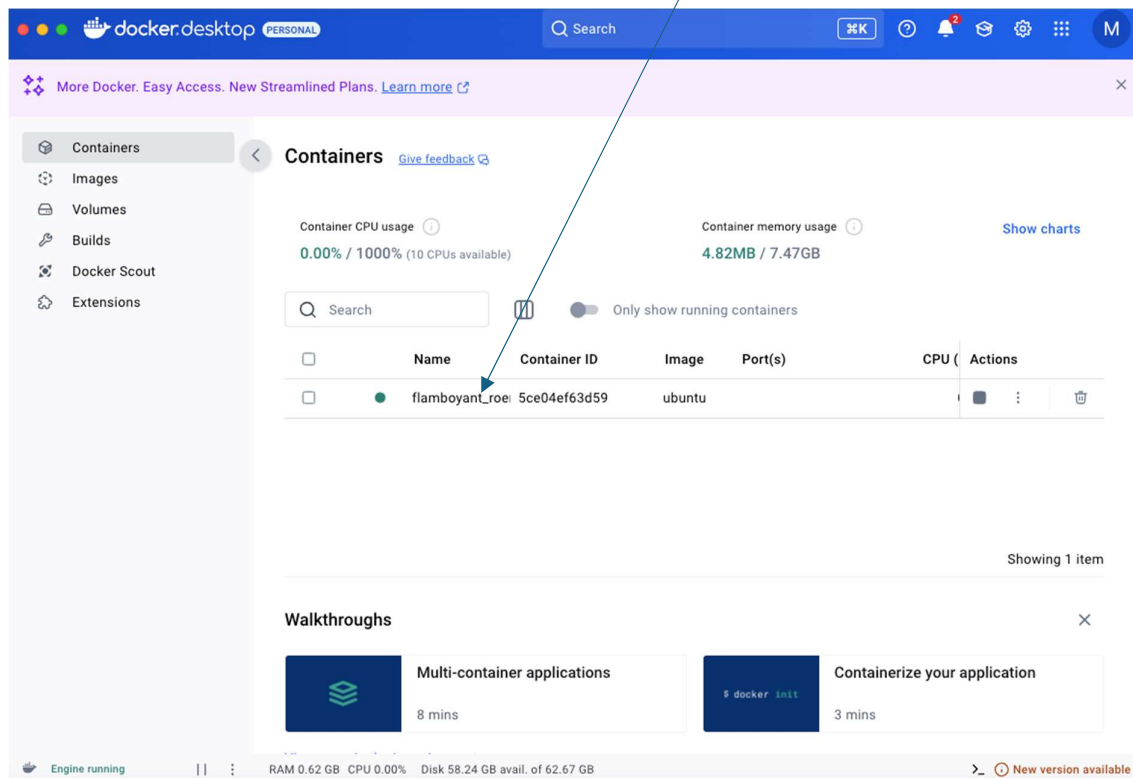


```
ahmedfouadi — -zsh — 80x24
Last login: Sun Jan  5 17:29:02 on ttys000
ahmedfouadi@Macmini-de-ahmed ~ % docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
8bb55f067777: Download complete
Digest: sha256:80dd3c3b9c6cecb9f1667e9290b3bc61b78c2678c02cbdae5f0fea92cc6734ab
Status: Downloaded newer image for ubuntu:latest
docker.io/library/ubuntu:latest
ahmedfouadi@Macmini-de-ahmed ~ %
```

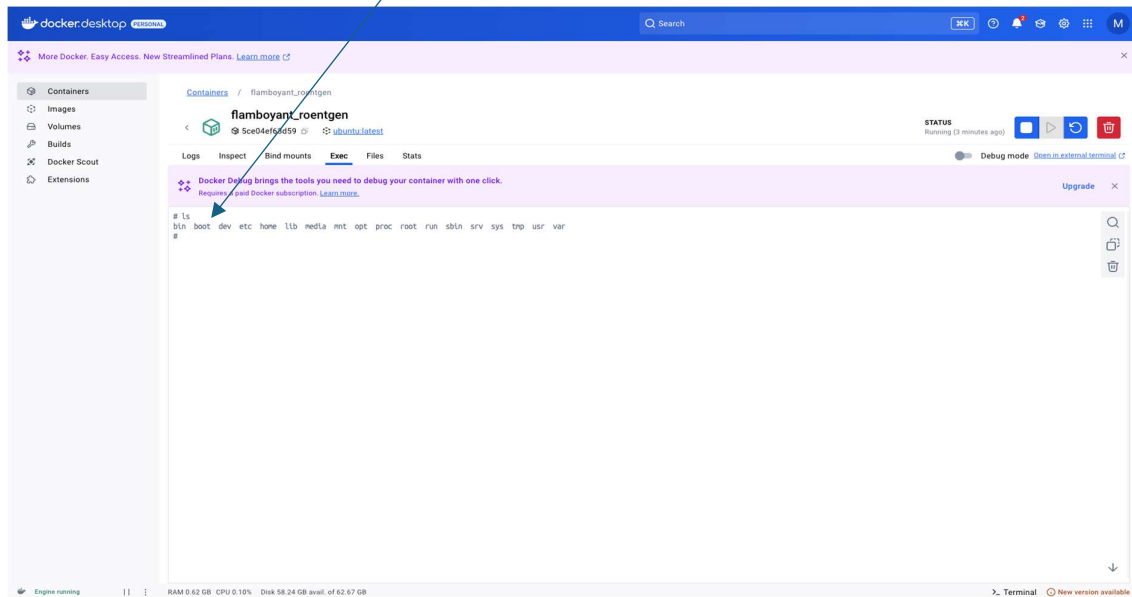
2.3 Après le téléchargement, il faudra créer et exécuter le conteneur Docker.

```
ahmedfouadi@Macmini-de-ahmed ~ % docker run -it ubuntu
root@5ce04ef63d59:/#
```

2.4 Vérification du conteneur depuis l'interface de Docker Desktop.

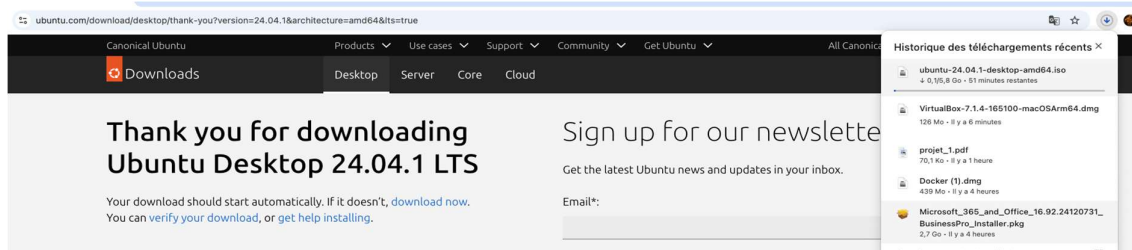


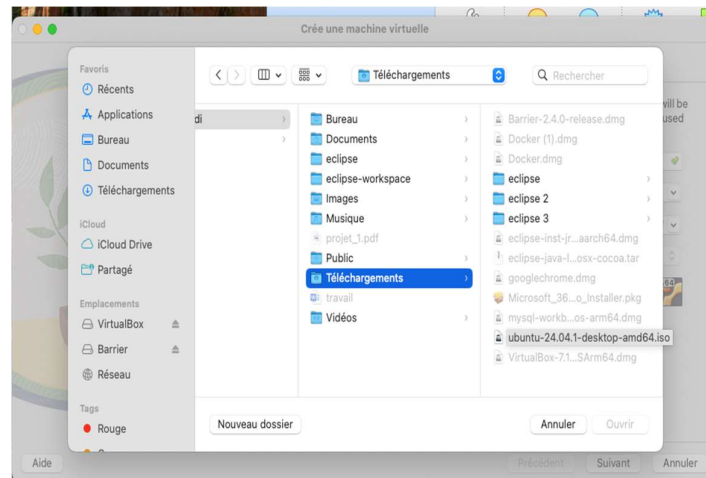
2.4 Connexion en mode commande depuis l'interface graphique de GitHub Desktop. Effectuez un ls pour vérifier la connectivité et la disponibilité de notre conteneur.



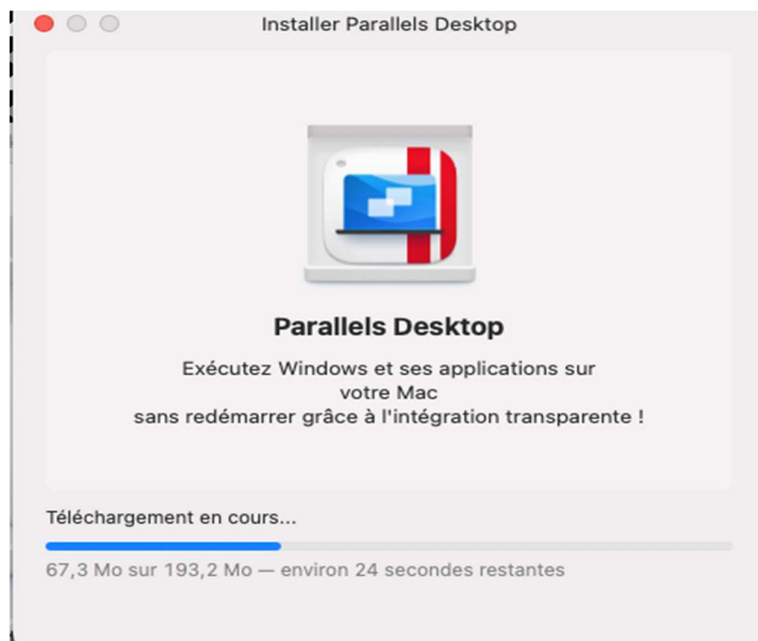
3. Installation d'un hyperviseur et d'une machine virtuelle :

3.1 Accédez au site d'Ubuntu pour télécharger une version d'Ubuntu Linux.





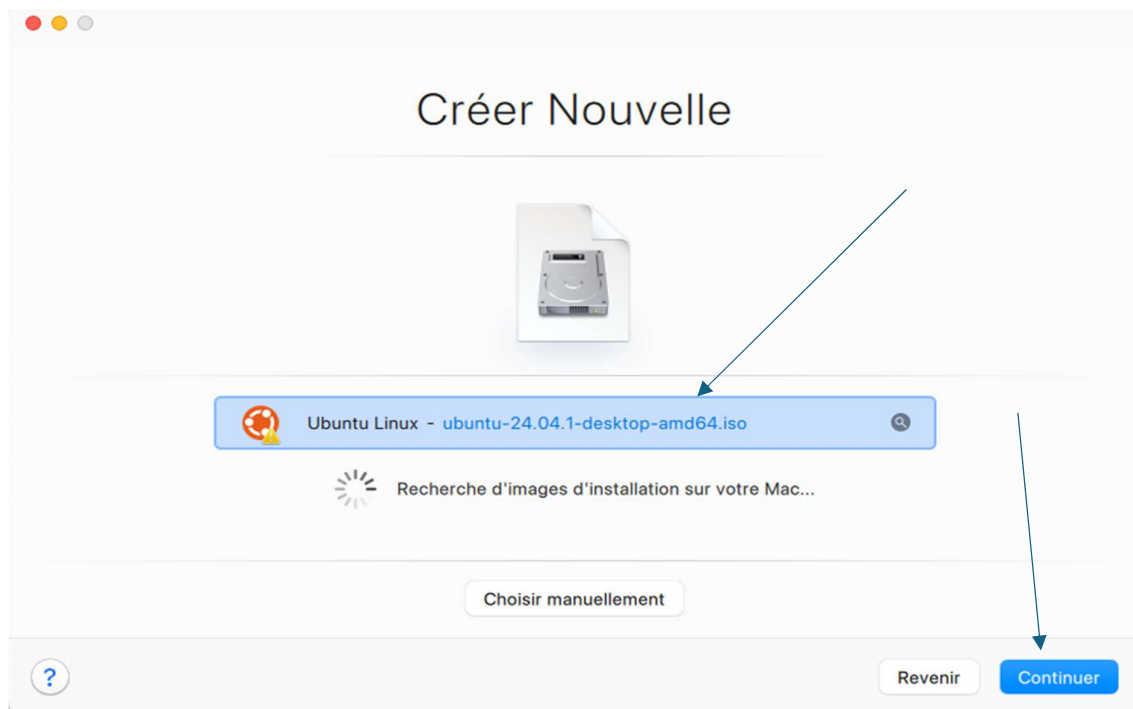
3.2 Téléchargez Parallels Desktop et procédez à l'installation (rôle d'hyperviseur).



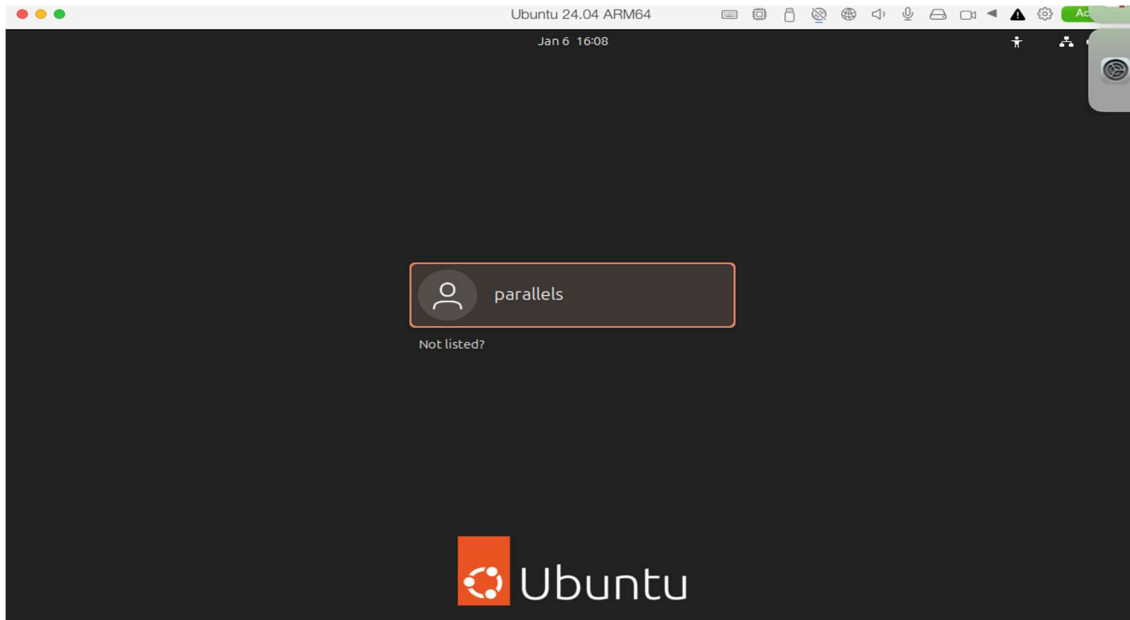
3.3 Parallels propose déjà des versions à télécharger. Étant donné que nous allons utiliser une version Ubuntu sur le conteneur Docker, il faudra, à cette étape, installer Ubuntu (en cliquant sur 'Télécharger Debian').



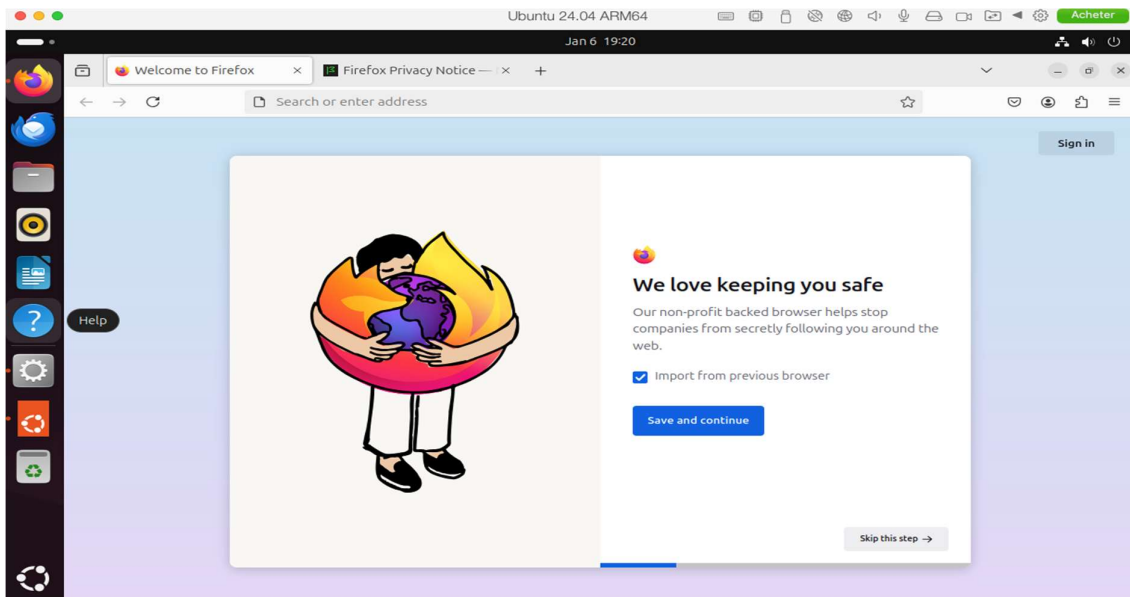
3.4 Parallels recherche la version en ISO d'Ubuntu et la télécharge.



3.5 Une fois l'installation terminée, Parallels vous propose de changer le mot de passe et d'accéder à la machine sous le système Linux.



3.6 Finalement, l'installation se termine et vous verrez un affichage comme ci-dessous.



4. Comparaison des systèmes d'exploitation (Docker vs Hyperviseur) :

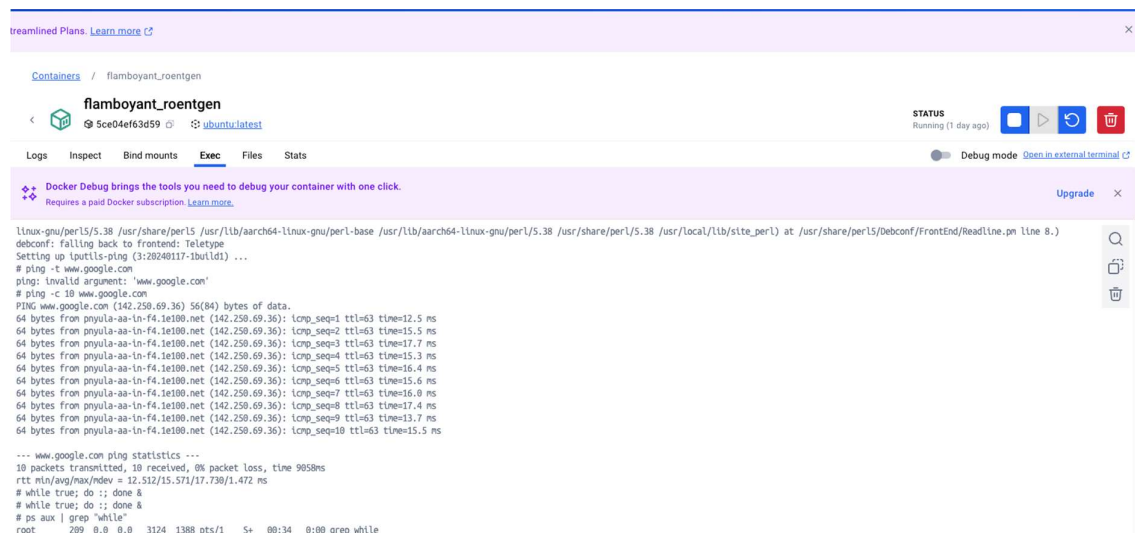
Pour comparer les deux systèmes, il nous faut une tâche répétitive et la capacité de voir l'utilisation du CPU, de la mémoire et des écritures sur le disque.

Les deux tâches que nous pouvons effectuer pour cette comparaison sont :

1. Exécuter un ping -c 10 www.google.com et
2. Lancer une boucle infinie avec while true; do ;; done &.

Note : Le ping effectuera 10 tests de connectivité vers le site www.google.com (Il faudra s'assurer d'avoir installé au préalable le package ping sur le conteneur Ubuntu). La boucle infinie est un processus qui restera en mémoire."

4.1 Lancement des tests (ping et boucle).

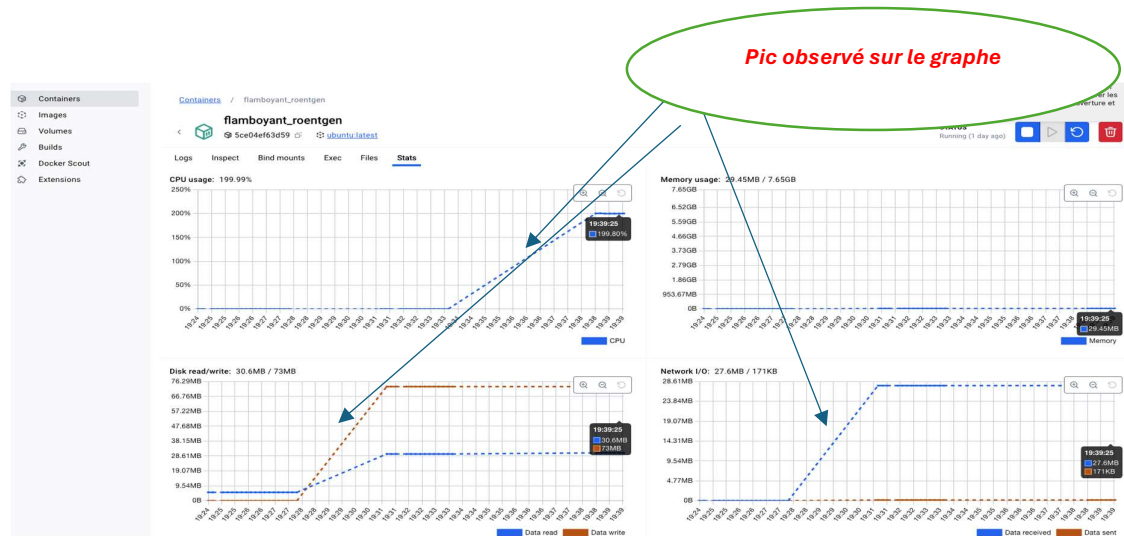


The screenshot shows the Docker Desktop interface for a container named 'flamboyant_roentgen'. The 'Exec' tab is active, displaying a terminal window. The terminal output shows the execution of a ping command and an infinite loop. The ping command results show 10 successful pings to www.google.com with varying times. The infinite loop is running in the background.

```
Linux-gnu/perl/5.38 /usr/share/perl5 /usr/lib/aarch64-linux-gnu/perl-base /usr/lib/aarch64-linux-gnu/perl/5.38 /usr/share/perl/5.38 /usr/local/lib/site_perl at /usr/share/perl5/Debianconf/FrontEnd/Readline.pm line 8.)
debconf: falling back to frontend: Teletype
Setting up lptutils-ping (3:20240117-1build1) ...
# ping -t www.google.com
ping: Unvalid argument: 'www.google.com'
# ping -c 10 www.google.com
PING www.google.com (142.250.69.36) 56(84) bytes of data.
64 bytes from pnyula-aa-in-f4.1e100.net (142.250.69.36): icmp_seq=1 ttl=63 time=12.5 ms
64 bytes from pnyula-aa-in-f4.1e100.net (142.250.69.36): icmp_seq=2 ttl=63 time=15.5 ms
64 bytes from pnyula-aa-in-f4.1e100.net (142.250.69.36): icmp_seq=3 ttl=63 time=17.7 ms
64 bytes from pnyula-aa-in-f4.1e100.net (142.250.69.36): icmp_seq=4 ttl=63 time=15.3 ms
64 bytes from pnyula-aa-in-f4.1e100.net (142.250.69.36): icmp_seq=5 ttl=63 time=16.4 ms
64 bytes from pnyula-aa-in-f4.1e100.net (142.250.69.36): icmp_seq=6 ttl=63 time=15.6 ms
64 bytes from pnyula-aa-in-f4.1e100.net (142.250.69.36): icmp_seq=7 ttl=63 time=16.0 ms
64 bytes from pnyula-aa-in-f4.1e100.net (142.250.69.36): icmp_seq=8 ttl=63 time=17.4 ms
64 bytes from pnyula-aa-in-f4.1e100.net (142.250.69.36): icmp_seq=9 ttl=63 time=13.7 ms
64 bytes from pnyula-aa-in-f4.1e100.net (142.250.69.36): icmp_seq=10 ttl=63 time=15.5 ms

--- www.google.com ping statistics ---
10 packets transmitted, 10 received, 0% packet loss, time 9058ms
rtt min/avg/max/mdev = 12.512/15.571/17.730/1.472 ms
# while true; do ;; done &
# ps aux | grep "while"
root      209  0.0  0.0   3124 1388 pts/1    S+   00:34   0:00 grep while
```

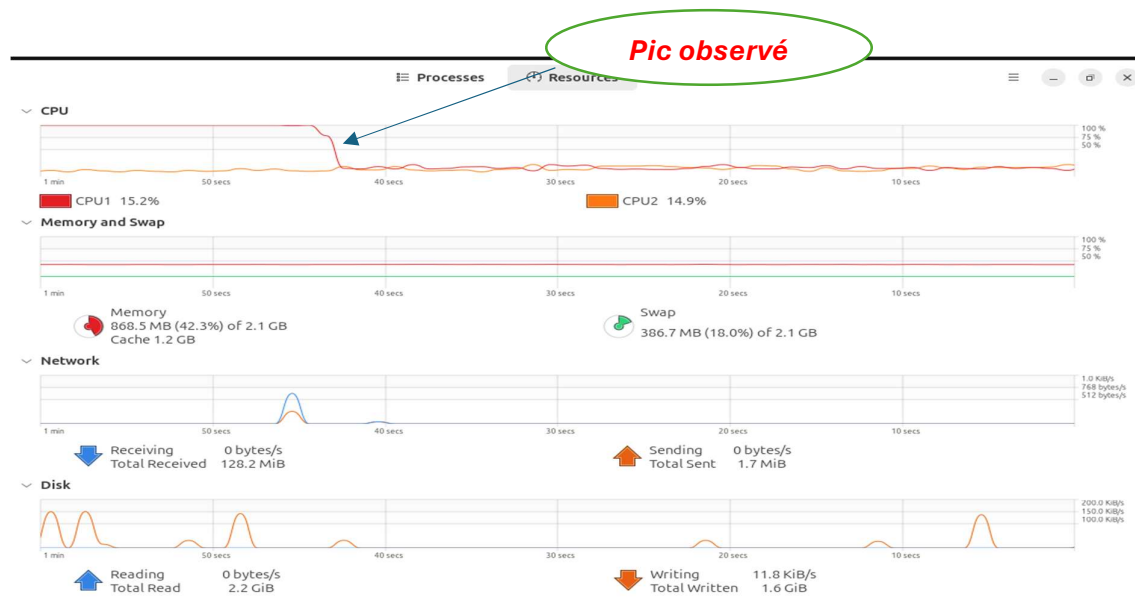
4.2 Après avoir lancé ces tests, Docker nous offre la possibilité de voir en temps réel la surcharge du CPU, des entrées/sorties (IO) et du réseau.



4.3 Lancement des tests sur la machine virtuelle Ubuntu.

```
parallels@ubuntu-linux-2404: ~  
64 bytes from pnyula-ab-in-f4.1e100.net (142.250.69.100): icmp_seq=2 ttl=128 time=13.8 ms  
64 bytes from pnyula-ab-in-f4.1e100.net (142.250.69.100): icmp_seq=3 ttl=128 time=16.2 ms  
64 bytes from pnyula-ab-in-f4.1e100.net (142.250.69.100): icmp_seq=4 ttl=128 time=17.4 ms  
64 bytes from pnyula-ab-in-f4.1e100.net (142.250.69.100): icmp_seq=5 ttl=128 time=15.6 ms  
64 bytes from pnyula-ab-in-f4.1e100.net (142.250.69.100): icmp_seq=6 ttl=128 time=14.8 ms  
64 bytes from pnyula-ab-in-f4.1e100.net (142.250.69.100): icmp_seq=7 ttl=128 time=19.3 ms  
64 bytes from pnyula-ab-in-f4.1e100.net (142.250.69.100): icmp_seq=8 ttl=128 time=14.0 ms  
64 bytes from pnyula-ab-in-f4.1e100.net (142.250.69.100): icmp_seq=9 ttl=128 time=16.1 ms  
64 bytes from pnyula-ab-in-f4.1e100.net (142.250.69.100): icmp_seq=10 ttl=128 time=13.9 ms  
  
--- www.google.com ping statistics ---  
10 packets transmitted, 10 received, 0% packet loss, time 9018ms  
rtt min/avg/max/mdev = 11.124/15.215/19.299/2.135 ms  
parallels@ubuntu-linux-2404:~$ while true; do ; done
```

4.4 Vérification des performances à travers le menu Ubuntu sur le système.



4.5 Liste des observations sur Docker avec le container Ubuntu :

Observation 1 : Nous constatons une surcharge rapide sur le conteneur, notamment au niveau du CPU, du réseau, ainsi que des opérations de lecture et d'écriture.

Observation 2 : Le conteneur est limité en termes d'espace disque, de CPU et de RAM. La taille allouée est très minimale (voir capture d'écran précédente).

4.5 Liste des observations sur hyperviseur avec parallèles et VM Ubuntu :

Observation : Nous constatons une surcharge rapide sur le CPU et aucun changement au niveau de la RAM et quelque pic minime sur le network et le disque.

4.6 Résumé des observations :

Lors de l'utilisation de Docker et d'une machine virtuelle, plusieurs différences notables ont été observées. Docker offre une gestion plus légère des ressources, car il fonctionne directement sur le noyau de l'hôte, ce qui lui permet de démarrer rapidement et de consommer moins de ressources système. Cependant, lorsqu'il est soumis à une charge importante (comme un usage intensif du CPU et du réseau, testé

lors des précédents tests avec la boucle infinie et le ping répétitif), il montre rapidement des signes de surcharge, notamment au niveau des opérations de lecture/écriture et de la gestion du réseau.

En comparaison, la machine virtuelle, qui dispose de son propre système d'exploitation et fonctionne sous un hyperviseur, offre un environnement plus complet mais aussi plus lourd. Elle consomme davantage de ressources système, notamment en termes de CPU, de mémoire et d'espace disque. Le démarrage est plus lent, mais elle offre un meilleur contrôle et une isolation plus forte des ressources, ce qui peut être essentiel pour certaines applications.

En résumé, selon les besoins du client et le contexte, chaque solution présente des avantages et des inconvénients. Pour des applications de grande envergure nécessitant beaucoup de ressources, comme dans le cas des applications de production, l'option de la machine virtuelle reste préférable. En revanche, pour des applications légères et dans le cadre de tests, la conteneurisation est un concept très judicieux.

5. Utilisation de commandes de base Linux :

5.1 La commande `ls` permet de lister le contenu d'un répertoire, et la commande `cd` permet de naviguer entre les répertoires.

Cas sur le container Ubuntu.

```
# ls
bin boot dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
# ls
bin boot dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
# ls
bin boot dev etc home lib media mnt opt proc root run sbin srv sys tmp usr var
# cd boot
#
```

5. 2 Cas sur la machine virtuelle sous Parallels : utilisation de la commande pwd pour connaître la racine sur laquelle on pointe, création et suppression de répertoires.

