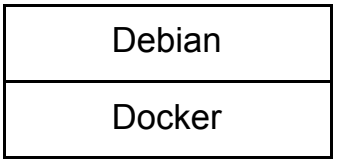
Docker

Docker est un logiciel open source
Alpin est une distribution minimaliste Linux de 5 Mo
nous allons faire 3 Linux avec une installation D'Alpiniste



Installation de Docker :

```
permitted by applicable law.
root@debian:~# apt-get install docker.io_
```

docker fonctionne en arrière-plan

si nous faisons la commande suivante

```
root@debian:~# docker_
```

nous remarquons que docker fonctionne, notons que chaque commande docker commence par "docker"

```
 events      Get real time events from the server
 exec        Run a command in a running container
 export      Export a container's filesystem as a tar archive
 history     Show the history of an image
 images      List images
 import      Import the contents from a tarball to create a filesystem image
 info        Display system-wide information
 inspect     Return low-level information on Docker objects
 kill        Kill one or more running containers
 load        Load an image from a tar archive or STDIN
 login       Log in to a Docker registry
 logout      Log out from a Docker registry
 logs        Fetch the logs of a container
 pause       Pause all processes within one or more containers
 port        List port mappings or a specific mapping for the container
 ps          List containers
 pull        Pull an image or a repository from a registry
 push        Push an image or a repository to a registry
 rename      Rename a container
 restart     Restart one or more containers
 rm          Remove one or more containers
 rmi         Remove one or more images
 run         Run a command in a new container
 save        Save one or more images to a tar archive (streamed to STDOUT by default)
 search      Search the Docker Hub for images
 start       Start one or more stopped containers
 stats       Display a live stream of container(s) resource usage statistics
 stop        Stop one or more running containers
 tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
 top         Display the running processes of a container
 unpause     Unpause all processes within one or more containers
 update      Update configuration of one or more containers
 version     Show the Docker version information
 wait        Block until one or more containers stop, then print their exit codes

un 'docker COMMAND --help' for more information on a command.
oot@debian:~#
```

Docker images permet de voir les images disponible dans docker, ici c’est vide

```
root@debian:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
```

nous installons alpine dans sa dernière version

```
root@debian:~# docker pull alpine:latest_
```

Docker images permet de voir les images disponible dans docker, ici il y à maintenant alpine

```
root@debian:~# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
alpine              latest              28f6e2705743        2 weeks ago         5.61MB
```

```
root@debian:~# docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
```

ps permet d'afficher les conteneurs présents

```
root@debian:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
```

Création du conteneur alpine

```
root@debian:~# docker run alpine
root@debian:~#
```

```
root@debian:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
        PORTS               NAMES
33f8682959d1        alpine              "/bin/sh"           57 seconds ago      Exited (0) 55 second
s ago                       clever_darwin
```

vérification, un conteneur alpine est présent

création du conteneur "alpine1":

```
root@debian:~# docker run -tid --name alpine1 alpine
03d1fadaaed96d2387082431a5f69624dea640a574fcc8f69ecf1a4a8d2b5a0a
root@debian:~# _
```

tid= terminal arrière plan (le "d" veut dire arrière paln)

On a un dock alpine actif et au nom donné (alpine1)

```
root@debian:~# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
      PORTS               NAMES
03d1fadaaed9        alpine              "/bin/sh"           36 seconds ago      Up 35 seconds
                    alpine1
33f8682959d1        alpine              "/bin/sh"           6 minutes ago       Exited (0) 6 minutes
 ago                clever_darwin
root@debian:~# _
```

On reproduit l'opération pour avoir trois docks.

```
root@debian:~# docker run -tid --name alpine2 alpine
2abf1024757f882fcac13aacce937bd29a274073ba327c3548e5548e92ed9e12
root@debian:~# docker run -tid --name alpine3 alpine
850b25e410d35f0e2ce752e392b901953ceb17eb16bccef257de139b721c93d8
^[[A^[[Aroot@debian:~#ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
      PORTS               NAMES
850b25e410d3        alpine              "/bin/sh"           4 seconds ago       Up 3 seconds
                    alpine3
2abf1024757f        alpine              "/bin/sh"           10 seconds ago      Up 9 seconds
                    alpine2
6758b6f52c03        alpine              "/bin/sh"           2 minutes ago       Up 2 minutes
                    alpine1
0b8b517fa002        alpine              "/bin/sh"           8 minutes ago       Exited (0) 8 minutes
 ago                    agitated_merkle
root@debian:~#
```

```
root@debian:~# docker exec -ti alpine1 sh
/ # _
```

sh pour le shell et ti pour “terminal”

```
root@debian:~# docker exec -ti alpine1 sh
/ # ls
bin    etc    lib    mnt    proc   run    srv    tmp    var
dev    home   media  opt    root   sbin   sys    usr
/ #
```

nous somme dans alpine, la couleur bleu désigne les dossiers

```
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
6: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.2/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
```

Nous sommes

Docker → Hôte
Docker0 → 172.17.0.1

docker possède un mode pont interne et envoie des ip à tout les autres conteneur,

`/ # exit` pour sortir du conteneur

```
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:11:00:03 brd ff:ff:ff:ff:ff:ff
    inet 172.17.0.3/16 brd 172.17.255.255 scope global eth0
       valid_lft forever preferred_lft forever
/ #
```

nous nous connectons au conteneur alpine2, on en profite pour ping alpine1.

```
/ # ping 172.17.0.1
PING 172.17.0.1 (172.17.0.1): 56 data bytes
64 bytes from 172.17.0.1: seq=0 ttl=64 time=0.103 ms
64 bytes from 172.17.0.1: seq=1 ttl=64 time=0.108 ms
64 bytes from 172.17.0.1: seq=2 ttl=64 time=0.116 ms
64 bytes from 172.17.0.1: seq=3 ttl=64 time=0.115 ms
64 bytes from 172.17.0.1: seq=4 ttl=64 time=0.107 ms
64 bytes from 172.17.0.1: seq=5 ttl=64 time=0.110 ms
64 bytes from 172.17.0.1: seq=6 ttl=64 time=0.109 ms
64 bytes from 172.17.0.1: seq=7 ttl=64 time=0.109 ms
^C
--- 172.17.0.1 ping statistics ---
8 packets transmitted, 8 packets received, 0% packet loss
round-trip min/avg/max = 0.103/0.109/0.116 ms
/ # ping 172.17.0.2
PING 172.17.0.2 (172.17.0.2): 56 data bytes
64 bytes from 172.17.0.2: seq=0 ttl=64 time=0.111 ms
64 bytes from 172.17.0.2: seq=1 ttl=64 time=0.126 ms
^C
--- 172.17.0.2 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 0.111/0.118/0.126 ms
/ # ping 172.17.0.4
PING 172.17.0.4 (172.17.0.4): 56 data bytes
64 bytes from 172.17.0.4: seq=0 ttl=64 time=0.111 ms
64 bytes from 172.17.0.4: seq=1 ttl=64 time=0.128 ms
64 bytes from 172.17.0.4: seq=2 ttl=64 time=0.130 ms
64 bytes from 172.17.0.4: seq=3 ttl=64 time=0.128 ms
64 bytes from 172.17.0.4: seq=4 ttl=64 time=0.127 ms
64 bytes from 172.17.0.4: seq=5 ttl=64 time=0.130 ms
64 bytes from 172.17.0.4: seq=6 ttl=64 time=0.127 ms
^C
```

```
/ # mkdir test_alpine2
/ # ls
bin           home          mnt           root          srv           tmp
dev           lib           opt           run           sys           usr
etc           media         proc          sbin          test_alpine2  var
/ # exit
root@debian:~# ls
python  unix
root@debian:~# _
```

Un dossier créé sur un dock alpine reste dans son dock respectif.

À noter : si on arrête le conteneur, on perd le dossier.

Cette commande permet de lister les cartes réseaux de docker.

```
root@debian:/# docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
b2c1b9ba6810        bridge              bridge              local
12442abd59e2        host                host                local
7d2b621e9231        none                null                local
```


```
            "HairpinMode": false,
            "LinkLocalIPv6Address": "",
            "LinkLocalIPv6PrefixLen": 0,
            "Ports": {},
            "SandboxKey": "/var/run/docker/netns/ce468ed456a1",
            "SecondaryIPAddresses": null,
            "SecondaryIPv6Addresses": null,
            "EndpointID": "1ade56a33d6ab56e3996a8cabff326fc62f936392104f9a28b2e0833dc0791c9",
            "Gateway": "172.17.0.1",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "MacAddress": "02:42:ac:11:00:02",
            "Networks": {
                "bridge": {
                    "IPAMConfig": null,
                    "Links": null,
                    "Aliases": null,
                    "NetworkID": "b2c1b9ba68108c1eed73f664cd1eb8d30c3ba727754e9588123abd27ed21860a",
                    "EndpointID": "1ade56a33d6ab56e3996a8cabff326fc62f936392104f9a28b2e0833dc0791c9"
,
                    "Gateway": "172.17.0.1",
                    "IPAddress": "172.17.0.2",
                    "IPPrefixLen": 16,
                    "IPv6Gateway": "",
                    "GlobalIPv6Address": "",
                    "GlobalIPv6PrefixLen": 0,
                    "MacAddress": "02:42:ac:11:00:02",
                    "DriverOpts": null
                }
            }
        }
    }
]
root@debian:/# docker inspect alpine1
```

Nous créons un nouveau réseaux dans docker du nom de pont3

```
root@debian:/# docker network create --driver bridge --subnet 172.20.0.0/24 pont3
8e24ae8676110f5e5a16c4d981fb9f99d6858e13add3117e57a11d7e125d7084
root@debian:/#
```

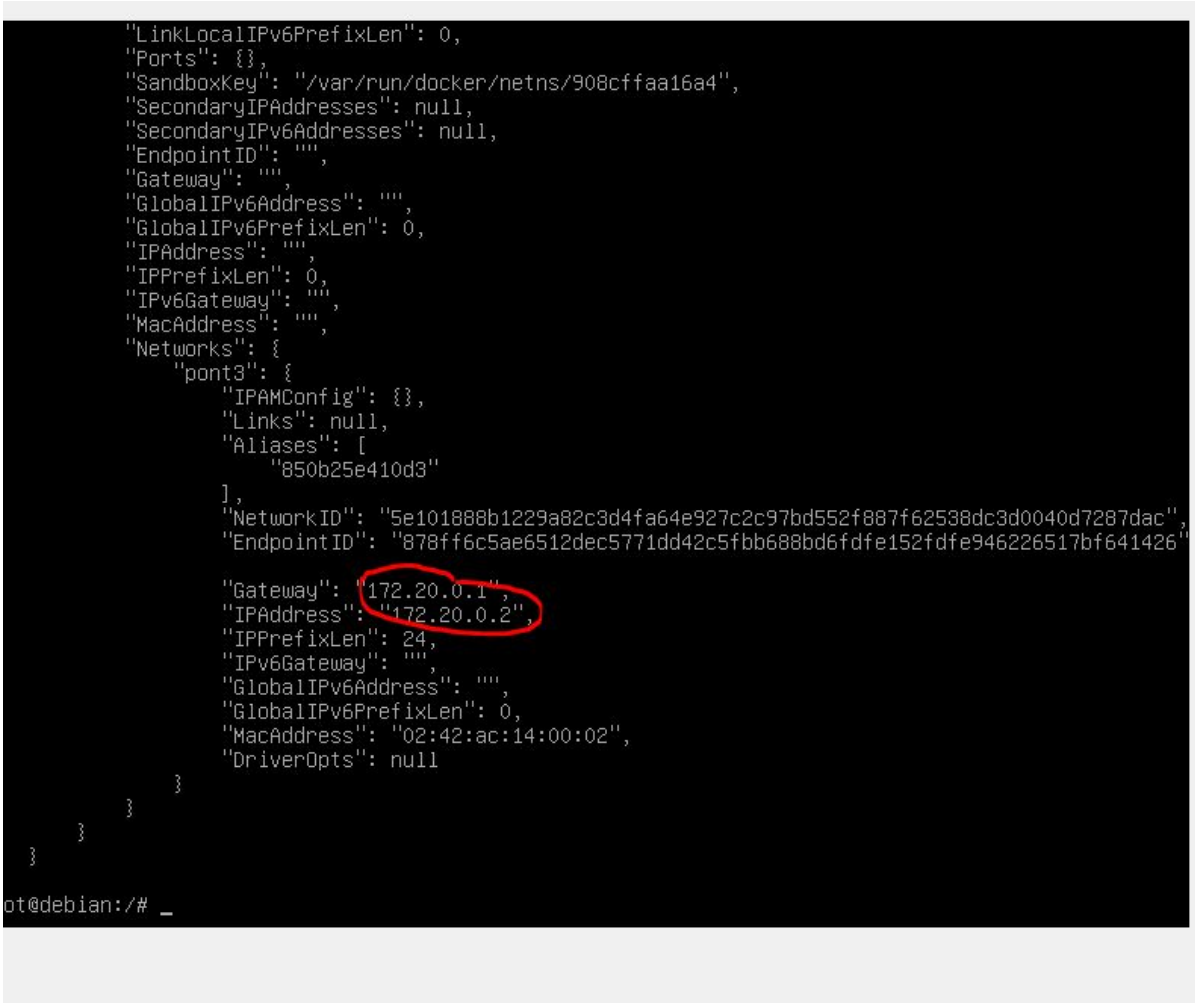
puis on connecte alpine3 a pont3

```
root@debian:/# docker network connect pont3 alpine3_
```

nous sommes connectés au DEUX ponts.

Donc nous déconnectons alpine3 de bridge

```
root@debian:/# docker network disconnect bridge alpine3
root@debian:/#
```



On vérifie en allant pingué un autre dock

```
root@debian:/# docker exec -ti alpine3 sh
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
13: eth1@if14: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue state UP
    link/ether 02:42:ac:14:00:02 brd ff:ff:ff:ff:ff:ff
    inet 172.20.0.2/24 brd 172.20.0.255 scope global eth1
       valid_lft forever preferred_lft forever
/ # ping 172.17.0.3
PING 172.17.0.3 (172.17.0.3): 56 data bytes
_
```

Cela ne marche pas.

Aparté: apt-get ne marche pas sur alpine, c'est apk update

```
/ # apt-get update
sh: apt-get: not found
```

```
/ # apk update
fetch https://dl-cdn.alpinelinux.org/alpine/v3.13/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.13/community/x86_64/APKINDEX.tar.gz
v3.13.2-66-g3837863e5e [https://dl-cdn.alpinelinux.org/alpine/v3.13/main]
v3.13.2-65-g71dd266990 [https://dl-cdn.alpinelinux.org/alpine/v3.13/community]
OK: 13877 distinct packages available
```

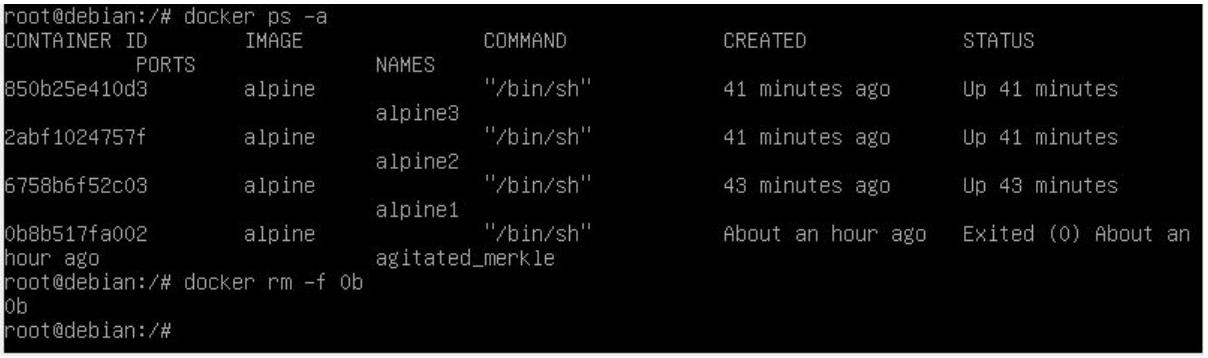
```
/ # mkdir testalpine3
/ # cd testalpine3
/testalpine3 # touch fichier
/testalpine3 # ls
fichier
/testalpine3 # nano fichier
sh: nano: not found
/testalpine3 # _
```

nano n'est pas installé, et le apt get n'existe pas, la commande est donc "apk add"

```
/ # apk add nano
(1/4) Installing libmagic (5.39-r0)
(2/4) Installing ncurses-terminfo-base (6.2_p20210109-r0)
(3/4) Installing ncurses-libs (6.2_p20210109-r0)
(4/4) Installing nano (5.4-r1)
Executing busybox-1.32.1-r3.trigger
OK: 13 MiB in 18 packages
/ # _
```

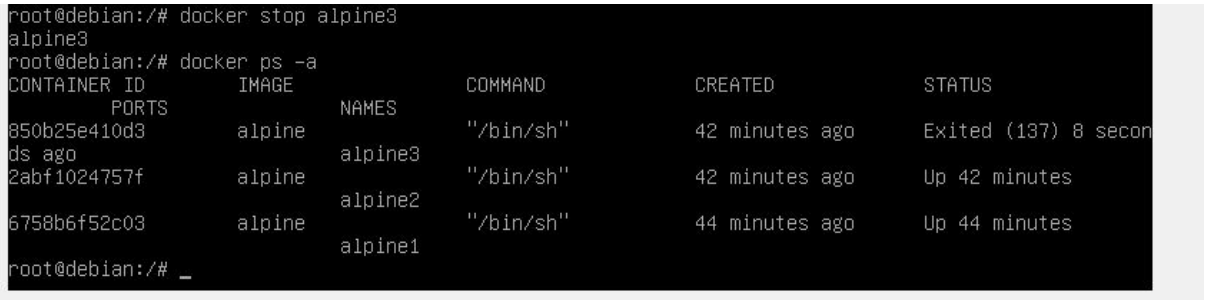
Aparté: plutôt qu'installer Nano, il y a Vi sur alpine (plus dur mais pur linux)

À noter : docker est très performant, tout simplement, car il n'y à pas d'os à installer, l'os est le Debian de base qui fait tourner docker.
On supprime le dock inactif, désormais inutile:



"rm"=suppression, "-f"= force (même si le dock est actif ça le supprime quand même)

On arrête le dock 3:



On le relance:

```
root@debian:/# docker start alpine3
alpine3
root@debian:/# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
850b25e410d3        alpine              "/bin/sh"           43 minutes ago      Up 1 second
                    alpine3
2abf1024757f        alpine              "/bin/sh"           43 minutes ago      Up 43 minutes
                    alpine2
6758b6f52c03        alpine              "/bin/sh"           45 minutes ago      Up 45 minutes
                    alpine1
root@debian:/# _
```

Suppression de alpine3 par son nom :

```
root@debian:/# docker rm -f alpine3
alpine3
root@debian:/# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
9076ac6a7cfe        alpine              "/bin/sh"           44 minutes ago      Up 44 minutes
                    alpine2
03d1fadaaed9        alpine              "/bin/sh"           About an hour ago   Up About an hour
                    alpine1
root@debian:/#
```

On supprime le reste:

```
root@debian:/# docker rm -f alpine1 - alpine2
alpine1
alpine2
Error: No such container: -
root@debian:/# docker ps -a
CONTAINER ID        IMAGE               COMMAND             CREATED             STATUS
PORTS               NAMES
root@debian:/#
```

Suppression de l'image alpine :

```
root@debian:/# docker rmi id 28
Untagged: alpine:latest
Untagged: alpine@sha256:a75afd8b57e7f34e4dad8d65e2c7ba2e1975c795ce1ee22fa34f8cf46f96a3be
Deleted: sha256:28f6e27057430ed2a40dbdd50d2736a3f0a295924016e294938110eeb8439818
Deleted: sha256:cb381a32b2296e4eb5af3f84092a2e6685e88adbc54ee0768a1a1010ce6376c7
Error: No such image: id
root@debian:/# docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
root@debian:/# _
```

Installation du dock d'apache "httpd" :

```
root@debian:/# docker pull httpd:latest
latest: Pulling from library/httpd
45b42c59be33: Pull complete
83ac8490fcc3: Pull complete
bdb2d204d86d: Pull complete
243acf75a504: Pull complete
8fc1ad93a9b1: Pull complete
Digest: sha256:3c252c919ef2445a6a41dde913a56202754821af87c049c4312bf81bdbc6df4b
Status: Downloaded newer image for httpd:latest
root@debian:/# docker run -tid --name apache httpd
b2a3c64aaa0b40ffff1d83ee2a9cbde1834dc3d671429a947c147ef18873ee96
root@debian:/#
```