IT.3503 - Architecture Virtualisée

TP 1: Linux Containers in Practice: a Docker flavor

GUO Xiaofan

YIN Chenghao

CONTENT

1. Environment Setup

1. Linux Namespaces, Cgroups & Docker

1. **What is Docker?**

   Docker is an open platform for developing, shipping, and running applications. Docker enables us to separate our applications from our infrastructure so we can deliver software quickly. It packages applications along with their dependencies into containers, enabling them to run consistently across different environments.

2. **What are the main components of Docker?**

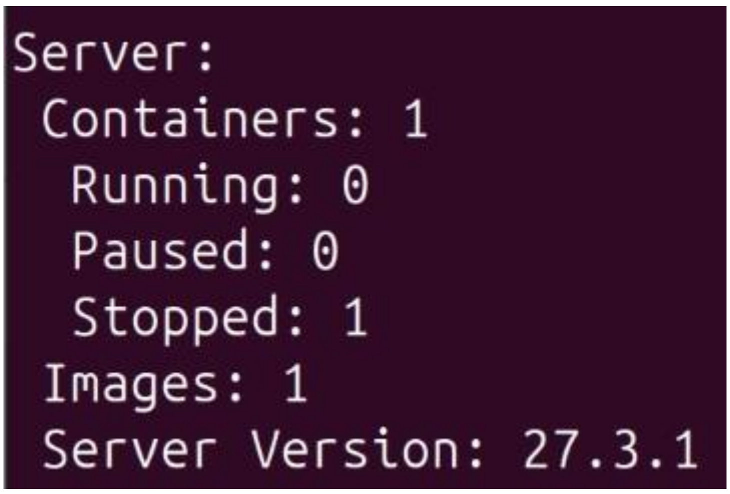
   - Docker Engine: The core engine that runs Docker containers, including the client (CLI) and the server (daemon).
   - Docker CLI: The command line interface for users to interact with Docker.
   - Docker Daemon: A service that runs in the background, responsible for managing containers, images, networks, etc.
   - Docker Images: Read-only templates for containers, containing everything needed to run an application.
   - Docker Containers: Running instances of an image, including a running application and the environment it requires.
   - Docker Registry: A place to store and distribute Docker images, with Docker Hub being the most common public registry.

3. **What are the technologies that Docker uses under the hood?**

- Linux Namespaces: Provide isolation of resources such as processes, networking, and file systems, ensuring that containers are isolated from each other.
- Cgroups (Control Groups): Manage and limit resources (like CPU, memory, disk I/O) for containers, ensuring proper allocation and isolation.
- Union File Systems (e.g., OverlayFS): Support layered filesystems, allowing containers to share base layers, which optimizes storage and performance.
- Container Runtime (e.g., runc): Docker uses a standard container runtime to create and manage containers.

2. Install Docker Engine

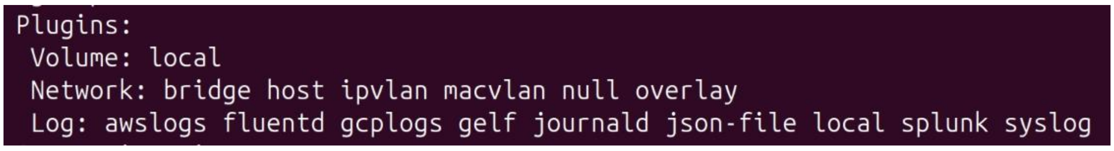
1. **What is the Docker server (daemon) version?**



```
Server:
 Containers: 1
  Running: 0
  Paused: 0
  Stopped: 1
 Images: 1
 Server Version: 27.3.1
```

The version of Docker server is 27.3.1

2. **What are the supported networking plugins?**

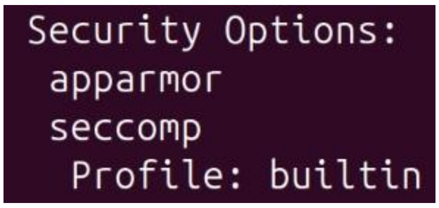


```
Plugins:
 Volume: local
 Network: bridge host ipvlan macvlan null overlay
 Log: awslogs fluentd gcplogs gelf journald json-file local splunk syslog
```

It supports the bridge, host, ipvlan, macvlan, null, overlay plugins.

- <u>bridge:</u> The default networking mode, which creates an internal private network for containers to communicate with each other.
- <u>host:</u> Shares the host's network stack with the container, allowing it to access the same network interfaces.
- <u>ipvlan:</u> Provides Layer 2 (L2) or Layer 3 (L3) network isolation for containers, suitable for more complex network topologies.
- <u>macvlan:</u> Allows containers to have their own MAC addresses, useful for scenarios requiring high network isolation.
- <u>null:</u> Disable container networking
- <u>overlay:</u> Used for cross-host container networks, often used in multi-host cluster environments.

3. **Does Docker use SELinux? If not, what are the supported tools?**



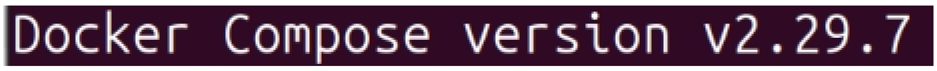
```
Security Options:
 apparmor
 seccomp
  Profile: builtin
```

Docker does not use SELinux.

Instead, it uses <u>apparmor</u> and <u>seccomp</u> as security options.

3. Install Docker Compose

1. What is Docker Compose?



```
Docker Compose version v2.29.7
```

Docker Compose is a tool for defining and running multi-container Docker applications using a YAML file (docker-compose.yml).

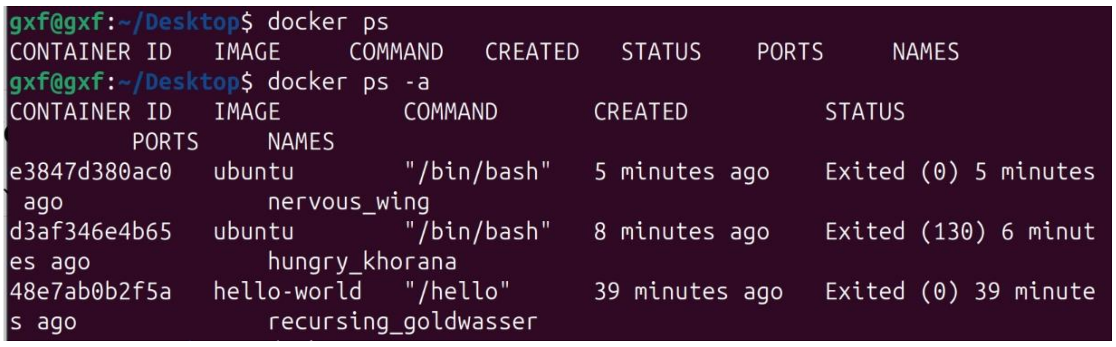
It allows users to describe services, networks, and volumes in a single file, making it easy to orchestrate and manage complex applications with just a few commands.

4. Docker CLI

1. What are the CLI commands that can give you:

a) the list of the running containers

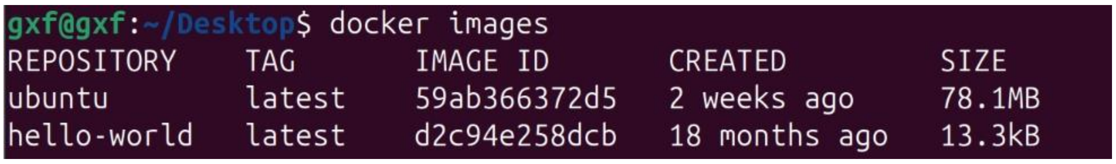
docker



```
gxf@gxf:~/Desktop$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
gxf@gxf:~/Desktop$ docker ps -a
CONTAINER ID   IMAGE         COMMAND       CREATED          STATUS
         PORTS     NAMES
e3847d380ac0   ubuntu        "/bin/bash"   5 minutes ago    Exited (0) 5 minutes
 ago                nervous_wing
d3af346e4b65   ubuntu        "/bin/bash"   8 minutes ago    Exited (130) 6 minut
es ago              hungry_khorana
48e7ab0b2f5a   hello-world   "/hello"      39 minutes ago   Exited (0) 39 minute
s ago               recursing_goldwasser
```

b) the list of available container images

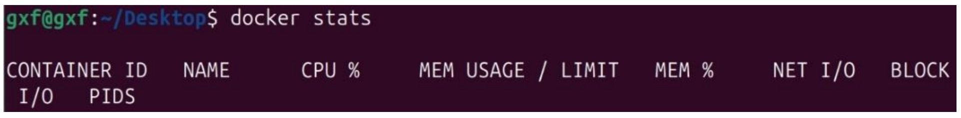
docker images



```
gxf@gxf:~/Desktop$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED         SIZE
ubuntu        latest    59ab366372d5   2 weeks ago     78.1MB
hello-world   latest    d2c94e258dcb   18 months ago   13.3kB
```

c) **some container statistics (CPU, RAM, I/O, etc.)**

docker stats

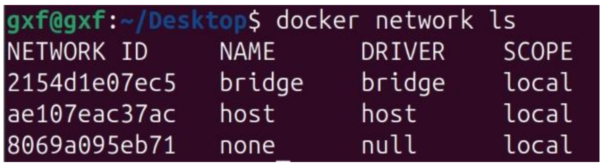


```
gxf@gxf:~/Desktop$ docker stats

CONTAINER ID   NAME      CPU %     MEM USAGE / LIMIT   MEM %     NET I/O   BLOCK
 I/O   PIDS
```

d) **the list of networks created by default**

docker network ls



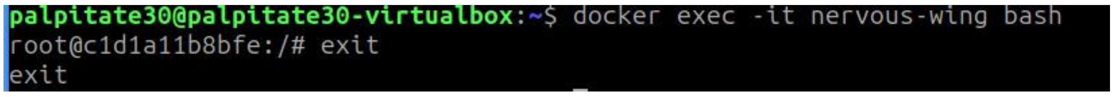
```
gxf@gxf:~/Desktop$ docker network ls
NETWORK ID     NAME      DRIVER    SCOPE
2154d1e07ec5   bridge    bridge    local
ae107eac37ac   host      host      local
8069a095eb71   none      null      local
```

2. **What is the command that can let you execute a command inside a running container?**

   docker exec -it <container_name_or_id> <command>

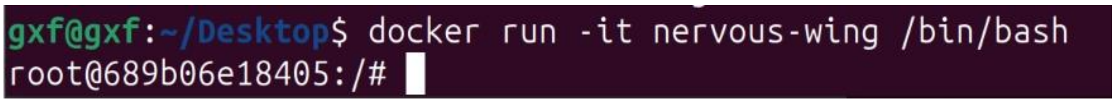
   For example, open an interactive bash shell inside the container named nervous-wing and exit:

   docker exec -it nervous-wing bash



```
palpitate30@palpitate30-virtualbox:~$ docker exec -it nervous-wing bash
root@c1d1a11b8bfe:/# exit
exit
```

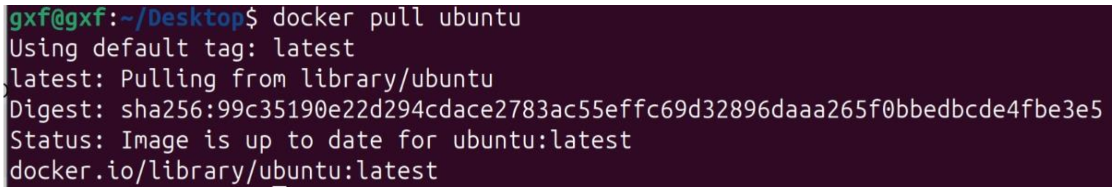
   If can't use *exec* to open, can use *run*:



```
gxf@gxf:~/Desktop$ docker run -it nervous-wing /bin/bash
root@689b06e18405:/#
```

3. **What is the command that can let you download a container image?**

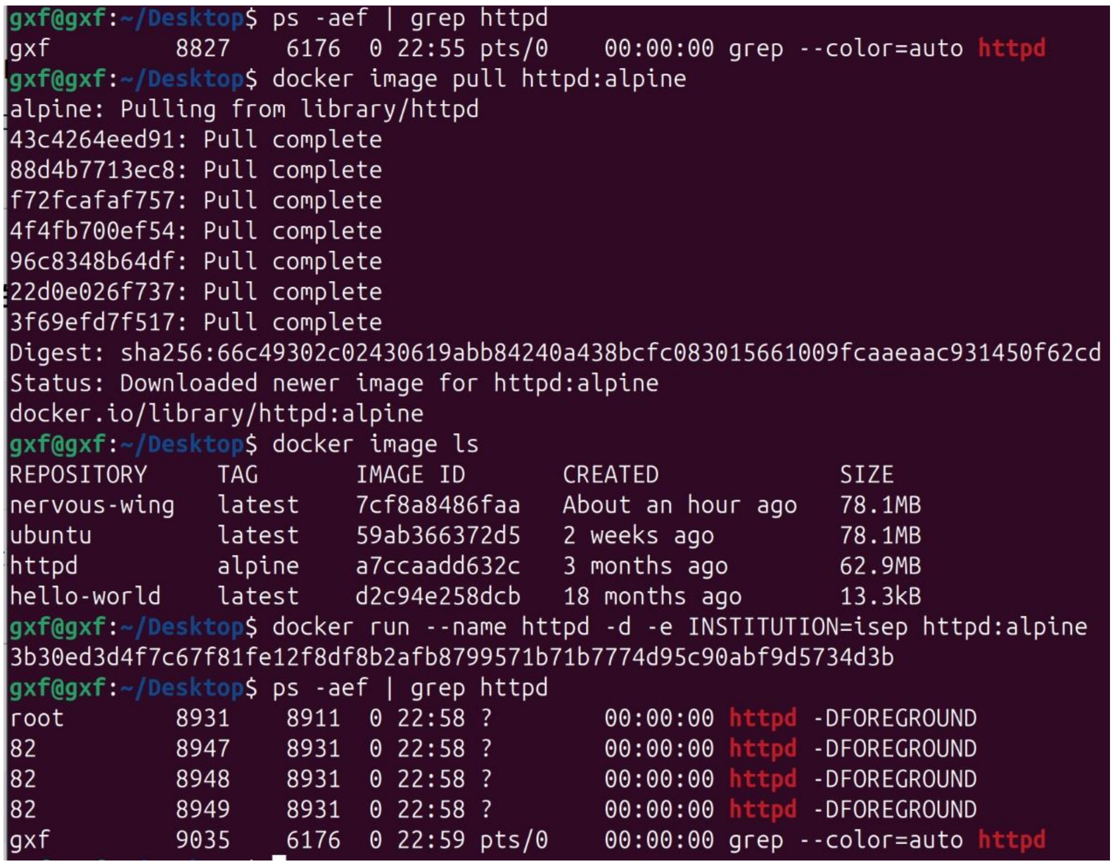
   docker pull <image_name>



```
gxf@gxf:~/Desktop$ docker pull ubuntu
Using default tag: latest
latest: Pulling from library/ubuntu
Digest: sha256:99c35190e22d294cdace2783ac55effc69d32896daaa265f0bbedbcde4fbe3e5
Status: Image is up to date for ubuntu:latest
docker.io/library/ubuntu:latest
```

2. What is a container?

1. Containers & Processes



```
gxf@gxf:~/Desktop$ ps -aef | grep httpd
gxf         8827    6176  0 22:55 pts/0    00:00:00 grep --color=auto httpd
gxf@gxf:~/Desktop$ docker image pull httpd:alpine
alpine: Pulling from library/httpd
43c4264eed91: Pull complete
88d4b7713ec8: Pull complete
f72fcafaf757: Pull complete
4f4fb700ef54: Pull complete
96c8348b64df: Pull complete
22d0e026f737: Pull complete
3f69efd7f517: Pull complete
Digest: sha256:66c49302c02430619abb84240a438bcfc083015661009fcaaeaac931450f62cd
Status: Downloaded newer image for httpd:alpine
docker.io/library/httpd:alpine
gxf@gxf:~/Desktop$ docker image ls
REPOSITORY     TAG       IMAGE ID       CREATED             SIZE
nervous-wing   latest    7cf8a8486faa   About an hour ago   78.1MB
ubuntu         latest    59ab366372d5   2 weeks ago         78.1MB
httpd          alpine    a7ccaadd632c   3 months ago        62.9MB
hello-world    latest    d2c94e258dcb   18 months ago       13.3kB
gxf@gxf:~/Desktop$ docker run --name httpd -d -e INSTITUTION=isep httpd:alpine
3b30ed3d4f7c67f81fe12f8df8b2afb8799571b71b7774d95c90abf9d5734d3b
gxf@gxf:~/Desktop$ ps -aef | grep httpd
root        8931    8911  0 22:58 ?        00:00:00 httpd -DFOREGROUND
82          8947    8931  0 22:58 ?        00:00:00 httpd -DFOREGROUND
82          8948    8931  0 22:58 ?        00:00:00 httpd -DFOREGROUND
82          8949    8931  0 22:58 ?        00:00:00 httpd -DFOREGROUND
gxf         9035    6176  0 22:59 pts/0    00:00:00 grep --color=auto httpd
```

1. What is the result of *ps -aef |grep httpd* now?

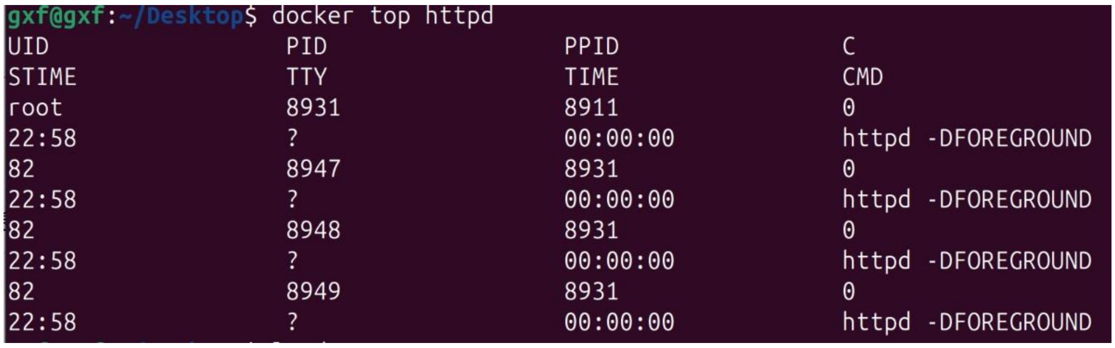
Multiple httpd processes are running:

- PID 8931, 8947, 8949, 8951 are shown as httpd processes.
- Each httpd process is running in the foreground and is shown as -DFOREGROUND.

2. What is the PID and PPID of the parent httpd process?

- PID: 8931
- PPID: 8911

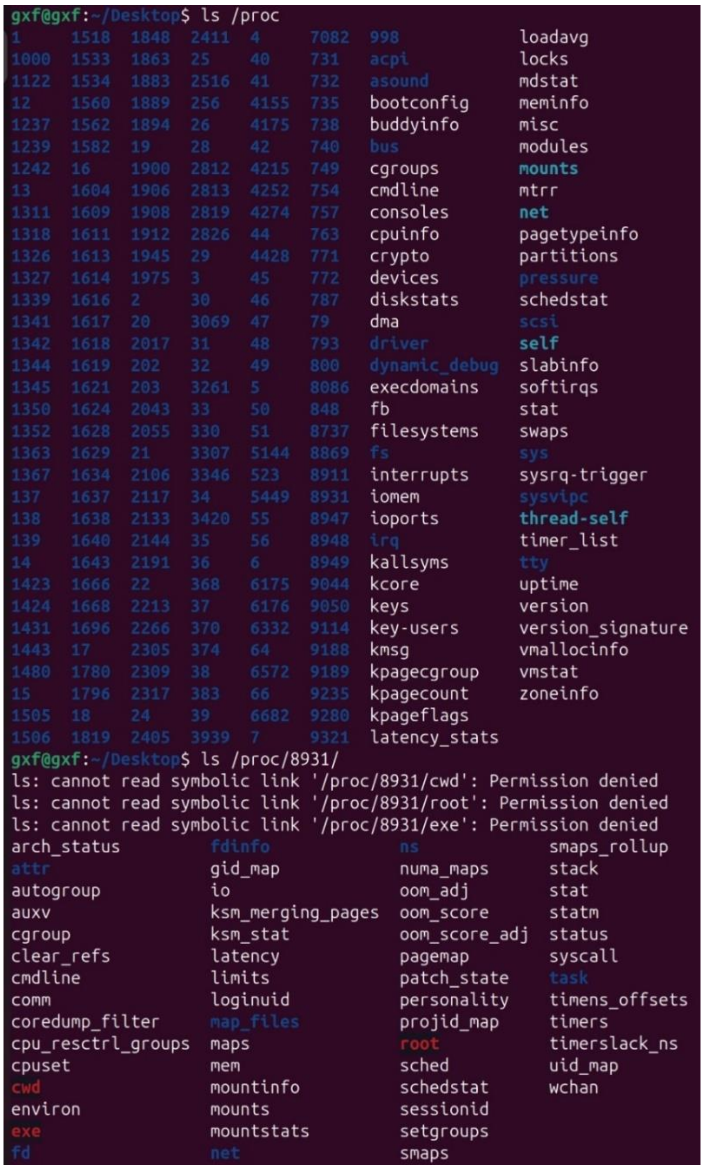
3. What can you notice about both outputs?



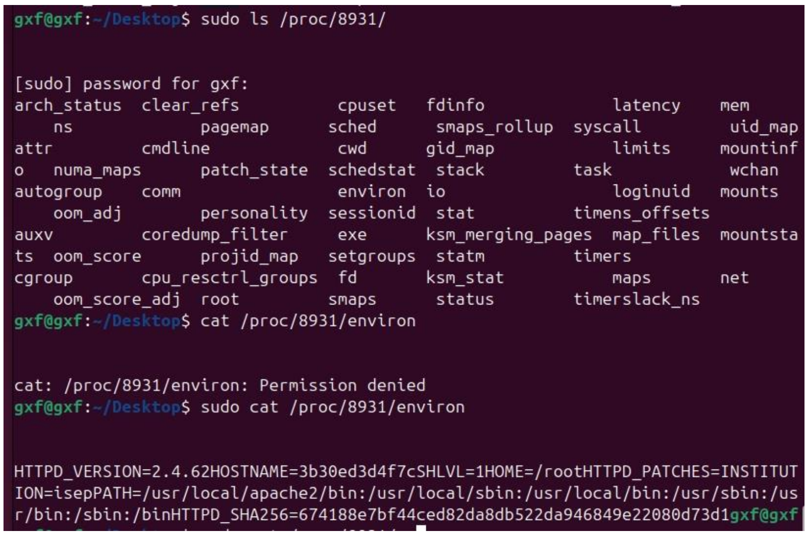
```
gxf@gxf:~/Desktop$ docker top httpd
UID                 PID                 PPID                C
STIME               TTY                 TIME                CMD
root                8931                8911                0
22:58               ?                   00:00:00            httpd -DFOREGROUND
82                  8947                8931                0
22:58               ?                   00:00:00            httpd -DFOREGROUND
82                  8948                8931                0
22:58               ?                   00:00:00            httpd -DFOREGROUND
82                  8949                8931                0
22:58               ?                   00:00:00            httpd -DFOREGROUND
```

- *ps -aef* shows all processes running on the host, including processes in containers. The PIDs of these processes are at the host system level.
- *docker top* shows the processes inside the container, which have different PIDs and PPIDs. Inside the container, the PID of the httpd process is 8931, which corresponds to the process ID inside the container.
- The processes inside the container shown by the *docker top* command are consistent with the processes shown by *ps -aef*, and the PID of the main process and the hierarchical structure of the child processes are the same.
- The processes inside the container have different PID and parent process relationships on the host, because Docker containers use independent PID namespaces.

4. What do you notice?



```
gxf@gxf:~/Desktop$ ls /proc
1     1518  1848  2411  4     7082  998             loadavg
1000  1533  1863  25    40    731   acpi            locks
1122  1534  1883  2516  41    732   asound          mdstat
12    1560  1889  256   4155  735   bootconfig      meminfo
1237  1562  1894  26    4175  738   buddyinfo       misc
1239  1582  19    28    42    740   bus             modules
1242  16    1900  2812  4215  749   cgroups         mounts
13    1604  1906  2813  4252  754   cmdline         mtrr
1311  1609  1908  2819  4274  757   consoles        net
1318  1611  1912  2826  44    763   cpuinfo         pagetypeinfo
1326  1613  1945  29    4428  771   crypto          partitions
1327  1614  1975  3     45    772   devices         pressure
1339  1616  2     30    46    787   diskstats       schedstat
1341  1617  20    3069  47    79    dma             scsi
1342  1618  2017  31    48    793   driver          self
1344  1619  202   32    49    800   dynamic_debug   slabinfo
1345  1621  203   3261  5     8086  execdomains     softirqs
1350  1624  2043  33    50    848   fb              stat
1352  1628  2055  330   51    8737  filesystems     swaps
1363  1629  21    3307  5144  8869  fs              sys
1367  1634  2106  3346  523   8911  interrupts      sysrq-trigger
137   1637  2117  34    5449  8931  iomem           sysvipc
138   1638  2133  3420  55    8947  ioports         thread-self
139   1640  2144  35    56    8948  irq             timer_list
14    1643  2191  36    6     8949  kallsyms        tty
1423  1666  22    368   6175  9044  kcore           uptime
1424  1668  2213  37    6176  9050  keys            version
1431  1696  2266  370   6332  9114  key-users       version_signature
1443  17    2305  374   64    9188  kmsg            vmallocinfo
1480  1780  2309  38    6572  9189  kpagecgroup     vmstat
15    1796  2317  383   66    9235  kpagecount      zoneinfo
1505  18    24    39    6682  9280  kpageflags
1506  1819  2405  3939  7     9321  latency_stats
gxf@gxf:~/Desktop$ ls /proc/8931/
ls: cannot read symbolic link '/proc/8931/cwd': Permission denied
ls: cannot read symbolic link '/proc/8931/root': Permission denied
ls: cannot read symbolic link '/proc/8931/exe': Permission denied
arch_status         fdinfo             ns              smaps_rollup
attr                gid_map            numa_maps       stack
autogroup           io                 oom_adj         stat
auxv                ksm_merging_pages  oom_score       statm
cgroup              ksm_stat           oom_score_adj   status
clear_refs          latency            pagemap         syscall
cmdline             limits             patch_state     task
comm                loginuid           personality     timens_offsets
coredump_filter     map_files          projid_map      timers
cpu_resctrl_groups  maps               root            timerslack_ns
cpuset              mem                sched           uid_map
cwd                 mountinfo          schedstat       wchan
environ             mounts             sessionid
exe                 mountstats         setgroups
fd                  net                smaps
```



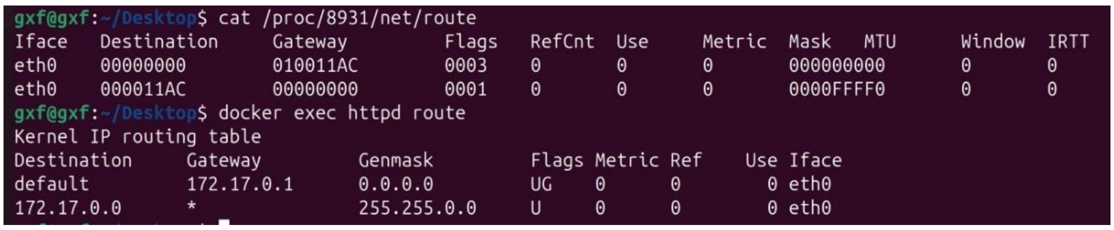
```
gxf@gxf:~/Desktop$ sudo ls /proc/8931/


[sudo] password for gxf:
arch_status  clear_refs          cpuset    fdinfo             latency    mem
    ns          pagemap      sched      smaps_rollup  syscall         uid_map
attr         cmdline             cwd       gid_map            limits     mountinf
o   numa_maps   patch_state  schedstat  stack         task            wchan
autogroup    comm                environ   io                 loginuid   mounts
    oom_adj        personality  sessionid  stat          timens_offsets
auxv         coredump_filter     exe       ksm_merging_pages  map_files  mountsta
ts  oom_score      projid_map   setgroups  statm         timers
cgroup       cpu_resctrl_groups  fd        ksm_stat           maps       net
    oom_score_adj  root         smaps      status        timerslack_ns
gxf@gxf:~/Desktop$ cat /proc/8931/environ


cat: /proc/8931/environ: Permission denied
gxf@gxf:~/Desktop$ sudo cat /proc/8931/environ


HTTPD_VERSION=2.4.62HOSTNAME=3b30ed3d4f7cSHLVL=1HOME=/rootHTTPD_PATCHES=INSTITUT
ION=isepPATH=/usr/local/apache2/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/us
r/bin:/sbin:/binHTTPD_SHA256=674188e7bf44ced82da8db522da946849e22080d73d1gxf@gxf
```

- Through the above instructions, many detailed information related to the httpd process can be seen, but some information cannot be accessed due to permission issues.
- Permission Denied: "*cat /porc/8931/environ*" displays a "Permission Denied" error. These directories represent the current working directory, root directory, and executable file path of the process. These symbolic links cannot be accessed as normal access rights. If you need to access this information, you can use sudo to elevate permissions to execute the command.



```
gxf@gxf:~/Desktop$ cat /proc/8931/net/route
Iface   Destination     Gateway         Flags   RefCnt  Use     Metric  Mask    MTU     Window  IRTT
eth0    00000000        010011AC        0003    0       0       0       000000000       0       0
eth0    000011AC        00000000        0001    0       0       0       0000FFFF0       0       0
gxf@gxf:~/Desktop$ docker exec httpd route
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         172.17.0.1      0.0.0.0         UG    0      0        0 eth0
172.17.0.0      *               255.255.0.0     U     0      0        0 eth0
```

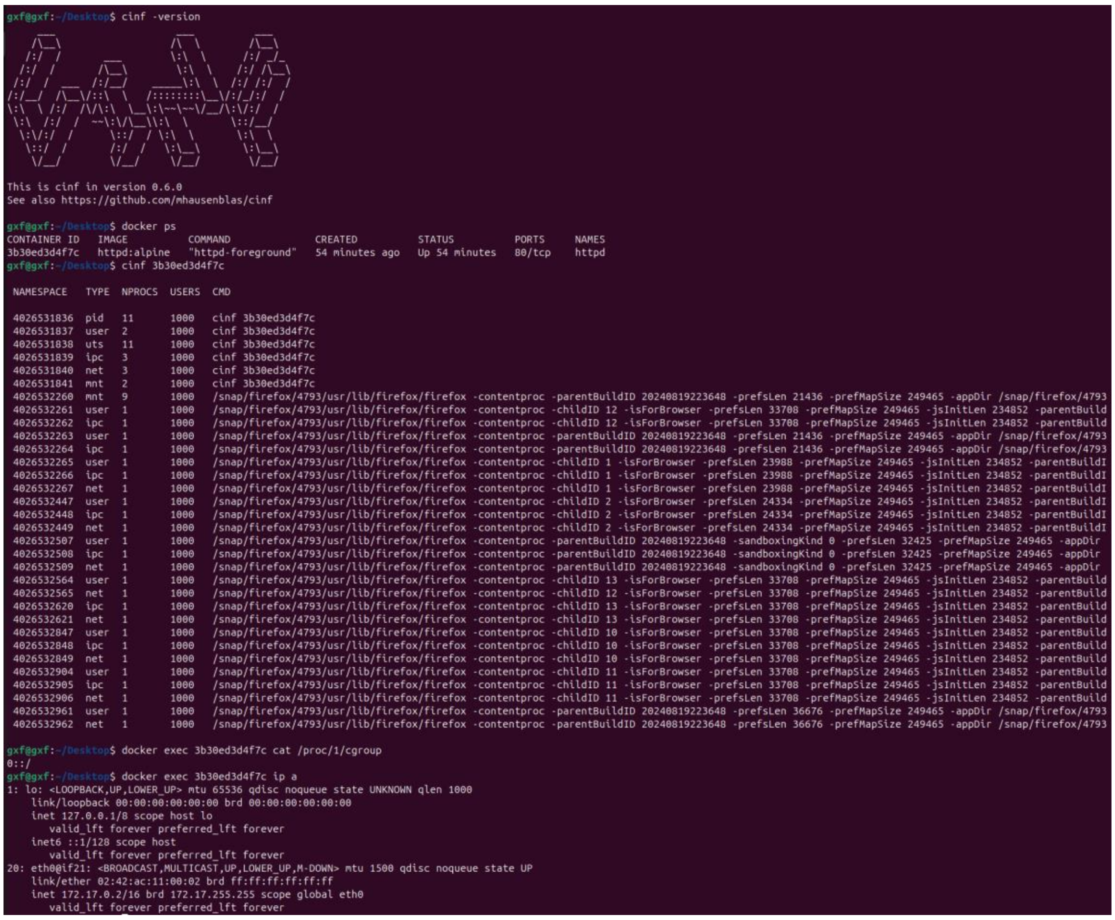
2. Containers & Namespaces

1. What cinf is used for?

cinf is a tool used to inspect the namespaces and other isolation features of running containers.

It allows to easily check which namespaces (such as PID, network, and mount) are being used by a container.

2. What <u>namespaces</u> are used by <u>httpd</u> container? How many?



```
gxf@gxf:~/Desktop$ cinf -version
      ___                       ___           ___
     /\__\                     /\  \         /\__\
    /:/  /         ___         \:\  \       /:/ _/_
   /:/  /         /\__\         \:\  \     /:/ /\__\
  /:/  /  ___    /:/__/     _____\:\  \   /:/ /:/  /
 /:/__/  /\__\/::\  \    /::::::::\__\/:/_/:/  /
 \:\  \ /:/  /\/\:\  \__\:\~~\~~\/__/\:\/:/  /
  \:\  /:/  /  ~~\:\/\__\\:\  \       \::/__/
   \:\/:/  /      \::/  /  \:\  \       \:\  \
    \::/  /       /:/  /    \:\__\       \:\__\
     \/__/        \/__/      \/__/        \/__/

This is cinf in version 0.6.0
See also https://github.com/mhausenblas/cinf

gxf@gxf:~/Desktop$ docker ps
CONTAINER ID   IMAGE          COMMAND              CREATED          STATUS          PORTS     NAMES
3b30ed3d4f7c   httpd:alpine   "httpd-foreground"   54 minutes ago   Up 54 minutes   80/tcp    httpd
gxf@gxf:~/Desktop$ cinf 3b30ed3d4f7c

 NAMESPACE   TYPE  NPROCS  USERS  CMD

 4026531836  pid   11      1000   cinf 3b30ed3d4f7c
 4026531837  user  2       1000   cinf 3b30ed3d4f7c
 4026531838  uts   11      1000   cinf 3b30ed3d4f7c
 4026531839  ipc   3       1000   cinf 3b30ed3d4f7c
 4026531840  net   3       1000   cinf 3b30ed3d4f7c
 4026531841  mnt   2       1000   cinf 3b30ed3d4f7c
 4026532260  mnt   9       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -parentBuildID 20240819223648 -prefsLen 21436 -prefMapSize 249465 -appDir /snap/firefox/4793
 4026532261  user  1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 12 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532262  ipc   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 12 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532263  user  1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -parentBuildID 20240819223648 -prefsLen 21436 -prefMapSize 249465 -appDir /snap/firefox/4793
 4026532264  ipc   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -parentBuildID 20240819223648 -prefsLen 21436 -prefMapSize 249465 -appDir /snap/firefox/4793
 4026532265  user  1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefsLen 23988 -prefMapSize 249465 -jsInitLen 234852 -parentBuildI
 4026532266  ipc   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefsLen 23988 -prefMapSize 249465 -jsInitLen 234852 -parentBuildI
 4026532267  net   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 1 -isForBrowser -prefsLen 23988 -prefMapSize 249465 -jsInitLen 234852 -parentBuildI
 4026532447  user  1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 2 -isForBrowser -prefsLen 24334 -prefMapSize 249465 -jsInitLen 234852 -parentBuildI
 4026532448  ipc   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 2 -isForBrowser -prefsLen 24334 -prefMapSize 249465 -jsInitLen 234852 -parentBuildI
 4026532449  net   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 2 -isForBrowser -prefsLen 24334 -prefMapSize 249465 -jsInitLen 234852 -parentBuildI
 4026532507  user  1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -parentBuildID 20240819223648 -sandboxingKind 0 -prefsLen 32425 -prefMapSize 249465 -appDir
 4026532508  ipc   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -parentBuildID 20240819223648 -sandboxingKind 0 -prefsLen 32425 -prefMapSize 249465 -appDir
 4026532509  net   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -parentBuildID 20240819223648 -sandboxingKind 0 -prefsLen 32425 -prefMapSize 249465 -appDir
 4026532564  user  1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 13 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532565  net   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 12 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532620  ipc   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 13 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532621  net   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 13 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532847  user  1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 10 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532848  ipc   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 10 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532849  net   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 10 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532904  user  1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 11 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532905  ipc   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 11 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532906  net   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -childID 11 -isForBrowser -prefsLen 33708 -prefMapSize 249465 -jsInitLen 234852 -parentBuild
 4026532961  user  1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -parentBuildID 20240819223648 -prefsLen 36676 -prefMapSize 249465 -appDir /snap/firefox/4793
 4026532962  net   1       1000   /snap/firefox/4793/usr/lib/firefox/firefox -contentproc -parentBuildID 20240819223648 -prefsLen 36676 -prefMapSize 249465 -appDir /snap/firefox/4793

gxf@gxf:~/Desktop$ docker exec 3b30ed3d4f7c cat /proc/1/cgroup
0::/
gxf@gxf:~/Desktop$ docker exec 3b30ed3d4f7c 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
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
20: eth0@if21: <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
```

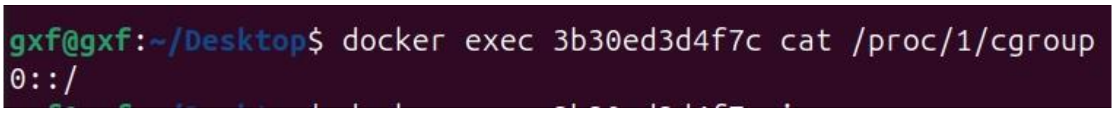
A total of 6 namespaces are used:

- <u>pid</u> (process namespace): PID namespace is used to isolate process IDs, so that the process ID inside the container is different from that of the host.
- <u>user</u> (user namespace): User namespace provides independent user and group IDs for different containers.
- <u>mnt </u>(mount namespace): used to isolate file system mount points, so that each container has an independent mount point view.
- <u>net</u> (network namespace): isolates network interfaces, so that the

network inside the container is isolated from the host.

- ipc (inter-process communication namespace): used to isolate inter-process communication resources of containers, such as semaphores.
- uts (host name and domain namespace): isolates host names and domain names.

3. What is the version of cgroups used by this container? Justify whether it's v1 or v2.



```
gxf@gxf:~/Desktop$ docker exec 3b30ed3d4f7c cat /proc/1/cgroup
0::/
```

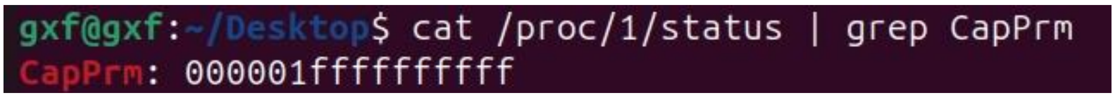
The container is using cgroups v1, confirmed by the output of /proc/1/cgroup.

3. Containers & Linux Capabilities

1. How process capabilities can be listed?

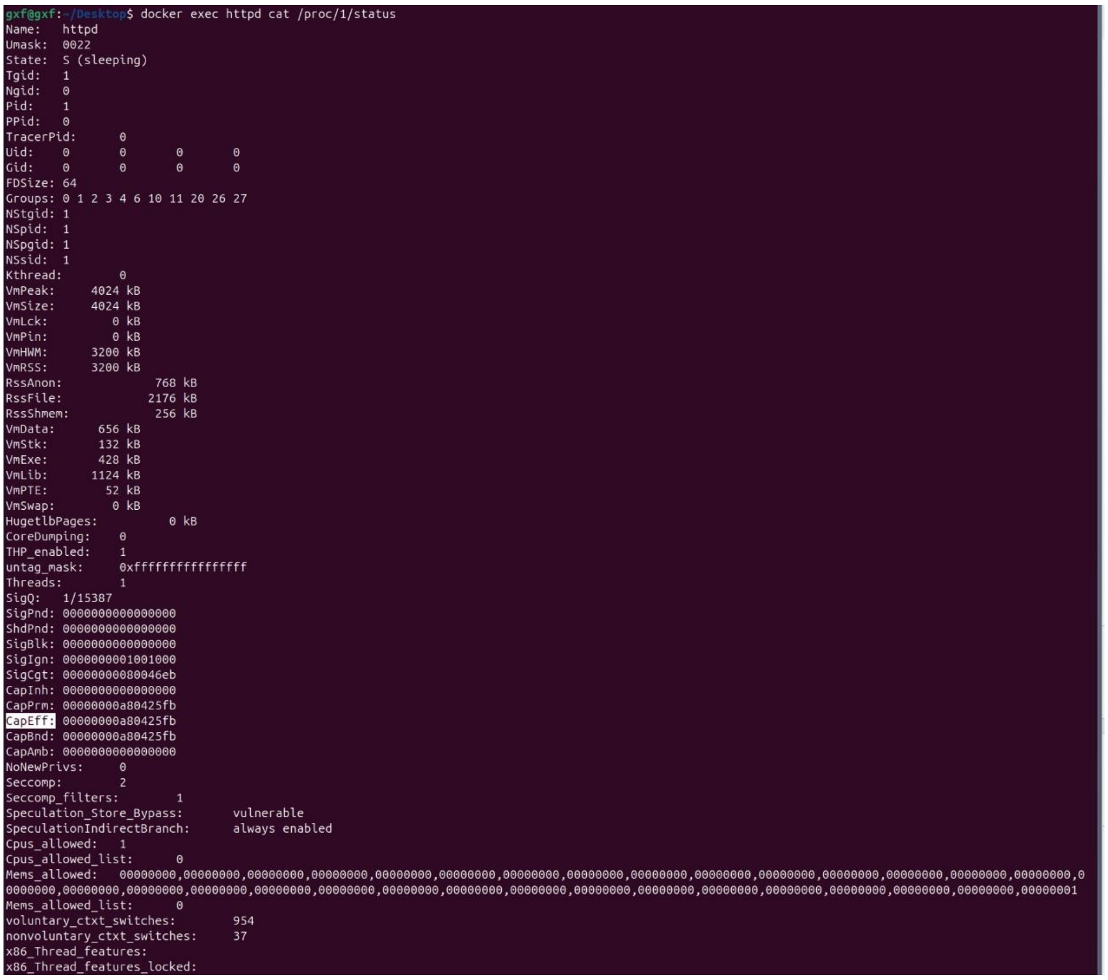
The capabilities of a process can be listed by reading the */proc/<PID>/status* file.

For example:

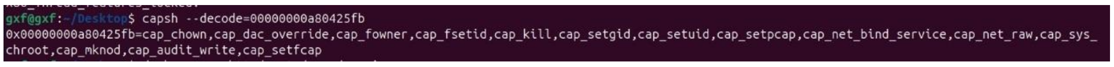


```
gxf@gxf:~/Desktop$ cat /proc/1/status | grep CapPrm
CapPrm: 000001ffffffffff
```

2. What are the permitted capabilities of the httpd container?



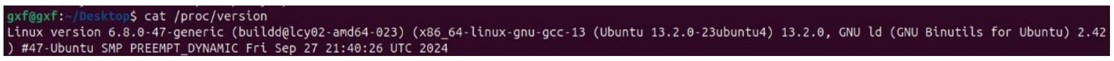
```
gxf@gxf:~/Desktop$ docker exec httpd cat /proc/1/status
Name:   httpd
Umask:  0022
State:  S (sleeping)
Tgid:   1
Ngid:   0
Pid:    1
PPid:   0
TracerPid:      0
Uid:    0       0       0       0
Gid:    0       0       0       0
FDSize: 64
Groups: 0 1 2 3 4 6 10 11 20 26 27
NStgid: 1
NSpid:  1
NSpgid: 1
NSsid:  1
Kthread:        0
VmPeak:     4024 kB
VmSize:     4024 kB
VmLck:         0 kB
VmPin:         0 kB
VmHWM:      3200 kB
VmRSS:      3200 kB
RssAnon:             768 kB
RssFile:            2176 kB
RssShmem:            256 kB
VmData:      656 kB
VmStk:       132 kB
VmExe:       428 kB
VmLib:      1124 kB
VmPTE:        52 kB
VmSwap:        0 kB
HugetlbPages:          0 kB
CoreDumping:    0
THP_enabled:    1
untag_mask:     0xffffffffffffffff
Threads:        1
SigQ:   1/15387
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000001001000
SigCgt: 00000000080046eb
CapInh: 0000000000000000
CapPrm: 00000000a80425fb
CapEff: 00000000a80425fb
CapBnd: 00000000a80425fb
CapAmb: 0000000000000000
NoNewPrivs:     0
Seccomp:        2
Seccomp_filters:        1
Speculation_Store_Bypass:       vulnerable
SpeculationIndirectBranch:      always enabled
Cpus_allowed:   1
Cpus_allowed_list:      0
Mems_allowed:   00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,0
0000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000001
Mems_allowed_list:      0
voluntary_ctxt_switches:        954
nonvoluntary_ctxt_switches:     37
x86_Thread_features:
x86_Thread_features_locked:
```



```
gxf@gxf:~/Desktop$ capsh --decode=00000000a80425fb
0x00000000a80425fb=cap_chown,cap_dac_override,cap_fowner,cap_fsetid,cap_kill,cap_setgid,cap_setuid,cap_setpcap,cap_net_bind_service,cap_net_raw,cap_sys_
chroot,cap_mknod,cap_audit_write,cap_setfcap
```

4. Linux Kernel

1. What is the Linux kernel's version of the httpd container?



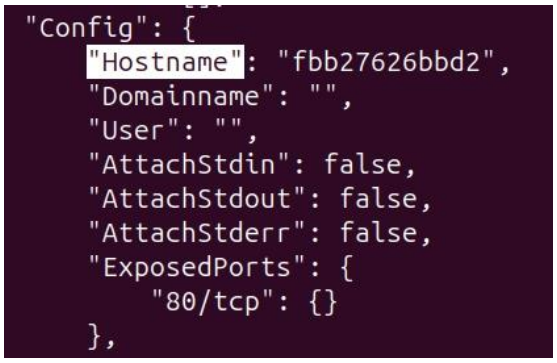
```
gxf@gxf:~/Desktop$ cat /proc/version
Linux version 6.8.0-47-generic (buildd@lcy02-amd64-023) (x86_64-linux-gnu-gcc-13 (Ubuntu 13.2.0-23ubuntu4) 13.2.0, GNU ld (GNU Binutils for Ubuntu) 2.42
) #47-Ubuntu SMP PREEMPT_DYNAMIC Fri Sep 27 21:40:26 UTC 2024
```

2. What can you say about it?

This is a relatively new Linux kernel version, especially used on Ubuntu 24.04, which brings better hardware compatibility and security enhancements.

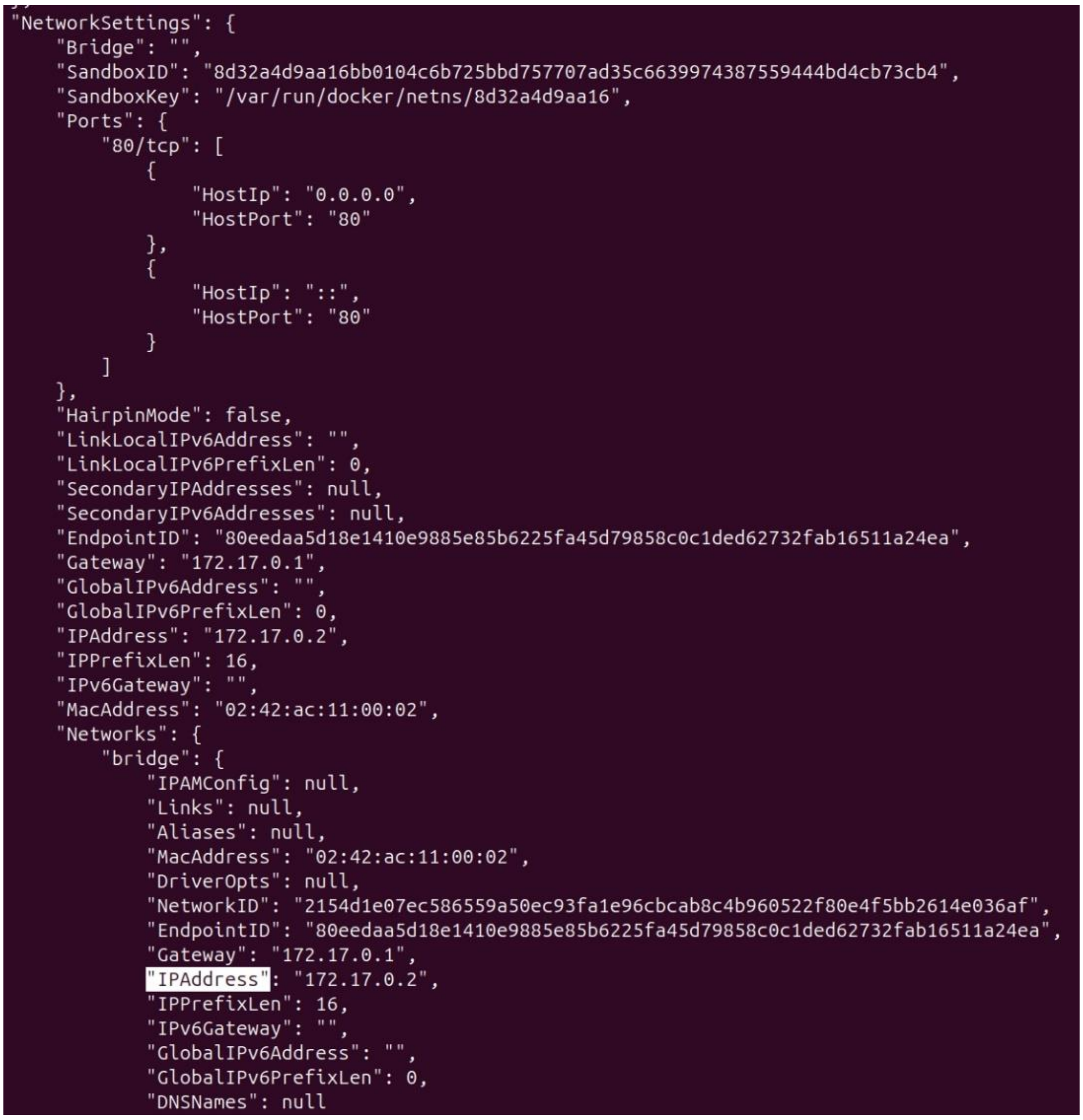
5. Inspecting a container

1. What is the Hostname of the container?



```
"Config": {
    "Hostname": "fbb27626bbd2",
    "Domainname": "",
    "User": "",
    "AttachStdin": false,
    "AttachStdout": false,
    "AttachStderr": false,
    "ExposedPorts": {
        "80/tcp": {}
    },
```

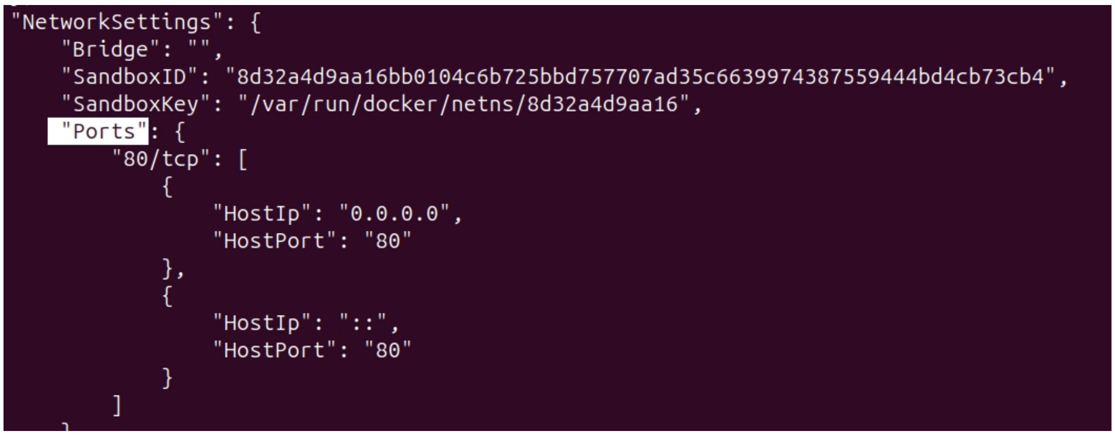
2. What is the IP address of the container?



```
"NetworkSettings": {
    "Bridge": "",
    "SandboxID": "8d32a4d9aa16bb0104c6b725bbd757707ad35c6639974387559444bd4cb73cb4",
    "SandboxKey": "/var/run/docker/netns/8d32a4d9aa16",
    "Ports": {
        "80/tcp": [
            {
                "HostIp": "0.0.0.0",
                "HostPort": "80"
            },
            {
                "HostIp": "::",
                "HostPort": "80"
            }
        ]
    },
    "HairpinMode": false,
    "LinkLocalIPv6Address": "",
    "LinkLocalIPv6PrefixLen": 0,
    "SecondaryIPAddresses": null,
    "SecondaryIPv6Addresses": null,
    "EndpointID": "80eedaa5d18e1410e9885e85b6225fa45d79858c0c1ded62732fab16511a24ea",
    "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,
            "MacAddress": "02:42:ac:11:00:02",
            "DriverOpts": null,
            "NetworkID": "2154d1e07ec586559a50ec93fa1e96cbcab8c4b960522f80e4f5bb2614e036af",
            "EndpointID": "80eedaa5d18e1410e9885e85b6225fa45d79858c0c1ded62732fab16511a24ea",
            "Gateway": "172.17.0.1",
            "IPAddress": "172.17.0.2",
            "IPPrefixLen": 16,
            "IPv6Gateway": "",
            "GlobalIPv6Address": "",
            "GlobalIPv6PrefixLen": 0,
            "DNSNames": null
```

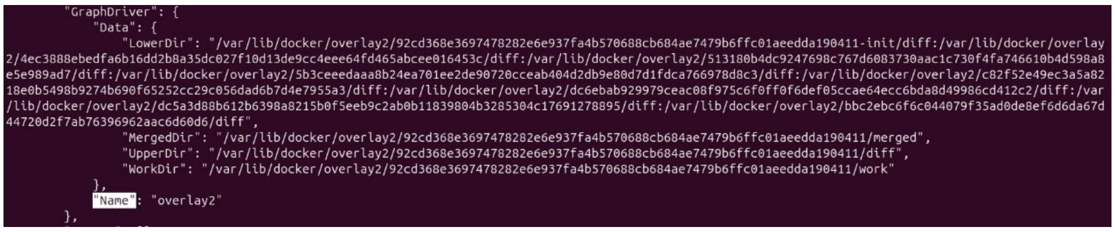
3. Does the container open any ports? If yes, which ones?

Yes, the open port is 80/TCP.



```
"NetworkSettings": {
    "Bridge": "",
    "SandboxID": "8d32a4d9aa16bb0104c6b725bbd757707ad35c6639974387559444bd4cb73cb4",
    "SandboxKey": "/var/run/docker/netns/8d32a4d9aa16",
    "Ports": {
        "80/tcp": [
            {
                "HostIp": "0.0.0.0",
                "HostPort": "80"
            },
            {
                "HostIp": "::",
                "HostPort": "80"
            }
        ]
```

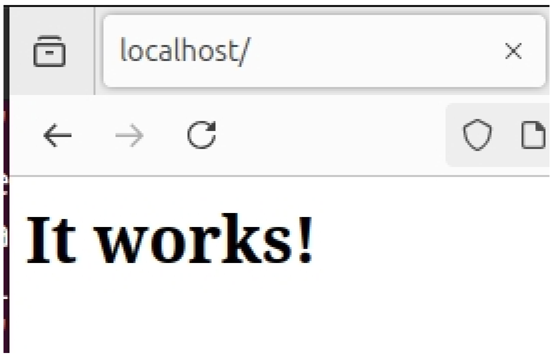
4. What storage driver the containers uses?



```
        "GraphDriver": {
            "Data": {
                "LowerDir": "/var/lib/docker/overlay2/92cd368e3697478282e6e937fa4b570688cb684ae7479b6ffc01aeedda190411-init/diff:/var/lib/docker/overlay
2/4ec3888ebedfa6b16dd2b8a35dc027f10d13de9cc4eee64fd465abcee016453c/diff:/var/lib/docker/overlay2/513180b4dc9247698c767d6083730aac1c730f4fa746610b4d598a8
e5e989ad7/diff:/var/lib/docker/overlay2/5b3ceeedaaa8b24ea701ee2de90720cceab404d2db9e80d7d1fdca766978d8c3/diff:/var/lib/docker/overlay2/c82f52e49ec3a5a82
18e0b5498b9274b690f65252cc29c056dad6b7d4e7955a3/diff:/var/lib/docker/overlay2/dc6ebab929979ceac08f975c6f0ff0f6def05ccae64ecc6bda8d49986cd412c2/diff:/var
/lib/docker/overlay2/dc5a3d88b612b6398a8215b0f5eeb9c2ab0b11839804b3285304c17691278895/diff:/var/lib/docker/overlay2/bbc2ebc6f6c044079f35ad0de8ef6d6da67d
44720d2f7ab76396962aac6d60d6/diff",
                "MergedDir": "/var/lib/docker/overlay2/92cd368e3697478282e6e937fa4b570688cb684ae7479b6ffc01aeedda190411/merged",
                "UpperDir": "/var/lib/docker/overlay2/92cd368e3697478282e6e937fa4b570688cb684ae7479b6ffc01aeedda190411/diff",
                "WorkDir": "/var/lib/docker/overlay2/92cd368e3697478282e6e937fa4b570688cb684ae7479b6ffc01aeedda190411/work"
            },
            "Name": "overlay2"
        },
```

6. Publishing ports

1. What is the result of the test?



It works!

3. Docker Images

1. Dockerfile

1. **What is the role of the FROM instruction?**

   The *FROM* instruction specifies the base image for creating a new Docker image. It serves as the foundation upon which subsequent layers and instructions will build.

   Every Dockerfile starts with a *FROM* statement, which tells Docker which base image to use (e.g., alpine:latest, ubuntu, etc.).

2. **What is an image layer?**

   An image layer is a read-only file system that adds to the base image each time a Dockerfile instruction (such as RUN, COPY, ADD) is executed. Layers are stacked on top of each other, with each new instruction adding a new layer. These layers make up the final Docker image.

3. **What is the difference between a container layer and an image layer?**

   - Image layer: These are read-only layers that make up a Docker image. Each layer corresponds to an instruction in the Dockerfile and forms part of the immutable file system.
   - Container layer: When a container is created from an image, a

writable layer is added on top of the image layers. This writable container layer allows changes (e.g., creating files, modifying configurations) during the container's runtime, but these changes are lost when the container is destroyed unless saved explicitly.

4. **Is there any alternatives for Docker doemon to build a Docker image?**

   Yes.

   - Podman allows building Docker images without requiring a Docker daemon.
   - Buildah can build OCI (Open Container Initiative) images without needing a running Docker daemon.

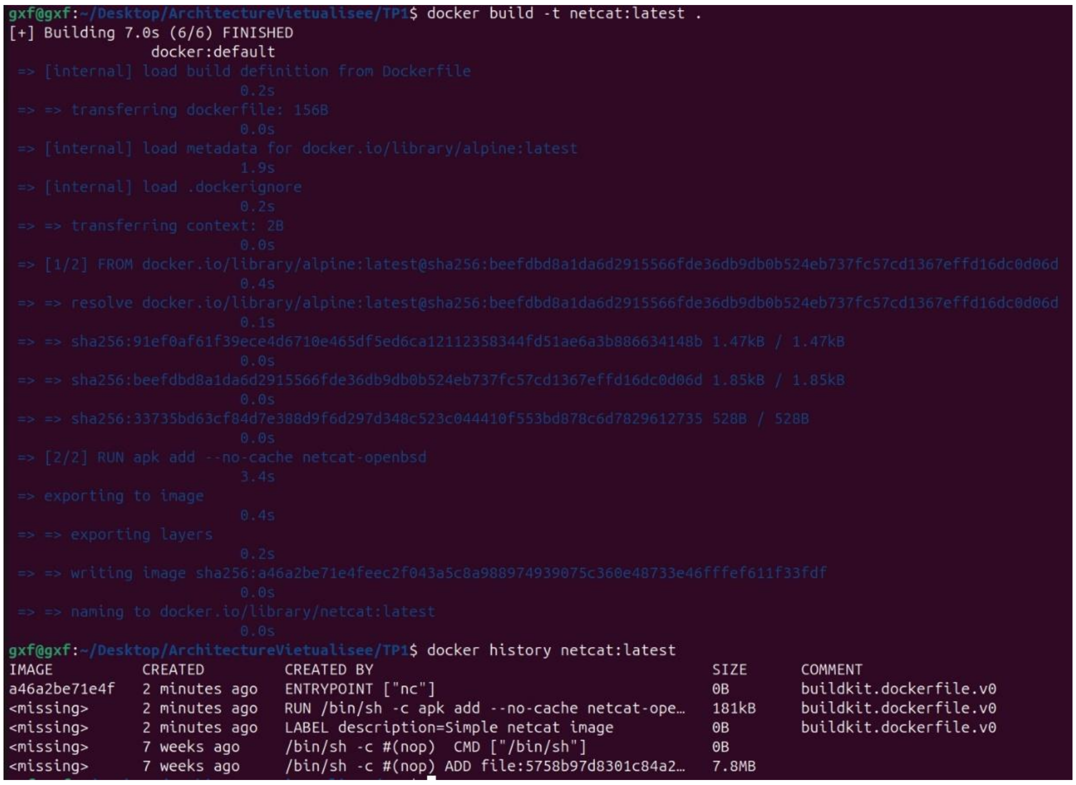
5. **What ENTRYPOINT is used for?**

   The *ENTRYPOINT* instruction defines the main command that will run when a container starts. *ENTRYPOINT* cannot be overridden during the container's runtime, making it more suitable for defining the main application that should always run in the container.

   In this case, the *ENTRYPOINT ["nc"]* ensures that *netcat* is always executed when the container starts.

2. Build the image

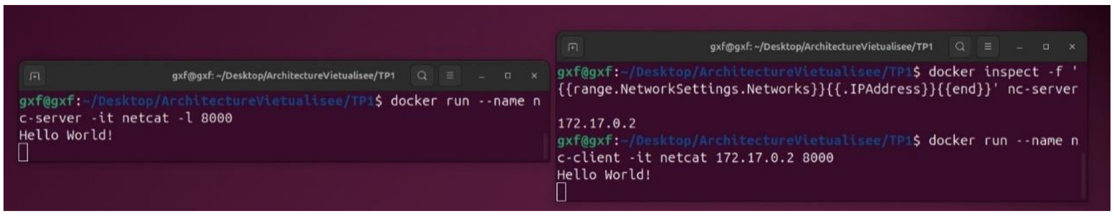
1. **How many layers your <u>netcat:latest</u> image contains ? Explain why?**



```
gxf@gxf:~/Desktop/ArchitectureVietualisee/TP1$ docker build -t netcat:latest .
[+] Building 7.0s (6/6) FINISHED
                docker:default
 => [internal] load build definition from Dockerfile
                         0.2s
 => => transferring dockerfile: 156B
                         0.0s
 => [internal] load metadata for docker.io/library/alpine:latest
                         1.9s
 => [internal] load .dockerignore
                         0.2s
 => => transferring context: 2B
                         0.0s
 => [1/2] FROM docker.io/library/alpine:latest@sha256:beefdbd8a1da6d2915566fde36db9db0b524eb737fc57cd1367effd16dc0d06d
                         0.4s
 => => resolve docker.io/library/alpine:latest@sha256:beefdbd8a1da6d2915566fde36db9db0b524eb737fc57cd1367effd16dc0d06d
                         0.1s
 => => sha256:91ef0af61f39ece4d6710e465df5ed6ca12112358344fd51ae6a3b886634148b 1.47kB / 1.47kB
                         0.0s
 => => sha256:beefdbd8a1da6d2915566fde36db9db0b524eb737fc57cd1367effd16dc0d06d 1.85kB / 1.85kB
                         0.0s
 => => sha256:33735bd63cf84d7e388d9f6d297d348c523c044410f553bd878c6d7829612735 528B / 528B
                         0.0s
 => [2/2] RUN apk add --no-cache netcat-openbsd
                         3.4s
 => exporting to image
                         0.4s
 => => exporting layers
                         0.2s
 => => writing image sha256:a46a2be71e4feec2f043a5c8a988974939075c360e48733e46fffef611f33fdf
                         0.0s
 => => naming to docker.io/library/netcat:latest
                         0.0s
gxf@gxf:~/Desktop/ArchitectureVietualisee/TP1$ docker history netcat:latest
IMAGE          CREATED         CREATED BY                                      SIZE      COMMENT
a46a2be71e4f   2 minutes ago   ENTRYPOINT ["nc"]                               0B        buildkit.dockerfile.v0
<missing>      2 minutes ago   RUN /bin/sh -c apk add --no-cache netcat-ope…   181kB     buildkit.dockerfile.v0
<missing>      2 minutes ago   LABEL description=Simple netcat image           0B        buildkit.dockerfile.v0
<missing>      7 weeks ago     /bin/sh -c #(nop)  CMD ["/bin/sh"]              0B
<missing>      7 weeks ago     /bin/sh -c #(nop) ADD file:5758b97d8301c84a2…   7.8MB
```

- The <u>*netcat:latest*</u> image has 4 layers in total.
- Each layer is generated based on the instructions in the <u>*Dockerfile*</u>. In the Dockerfile, instructions such as <u>*FROM, RUN, LABEL, ENTRYPOINT*</u> will generate new image layers.

2. **Why nc-client was able to connect to nc-server?**


gxf@gxf: ~/Desktop/ArchitectureVietualisee/TP1
gxf@gxf: ~/Desktop/ArchitectureVietualisee/TP1

```
gxf@gxf:~/Desktop/ArchitectureVietualisee/TP1$ docker run --name n
c-server -it netcat -l 8000
Hello World!
```

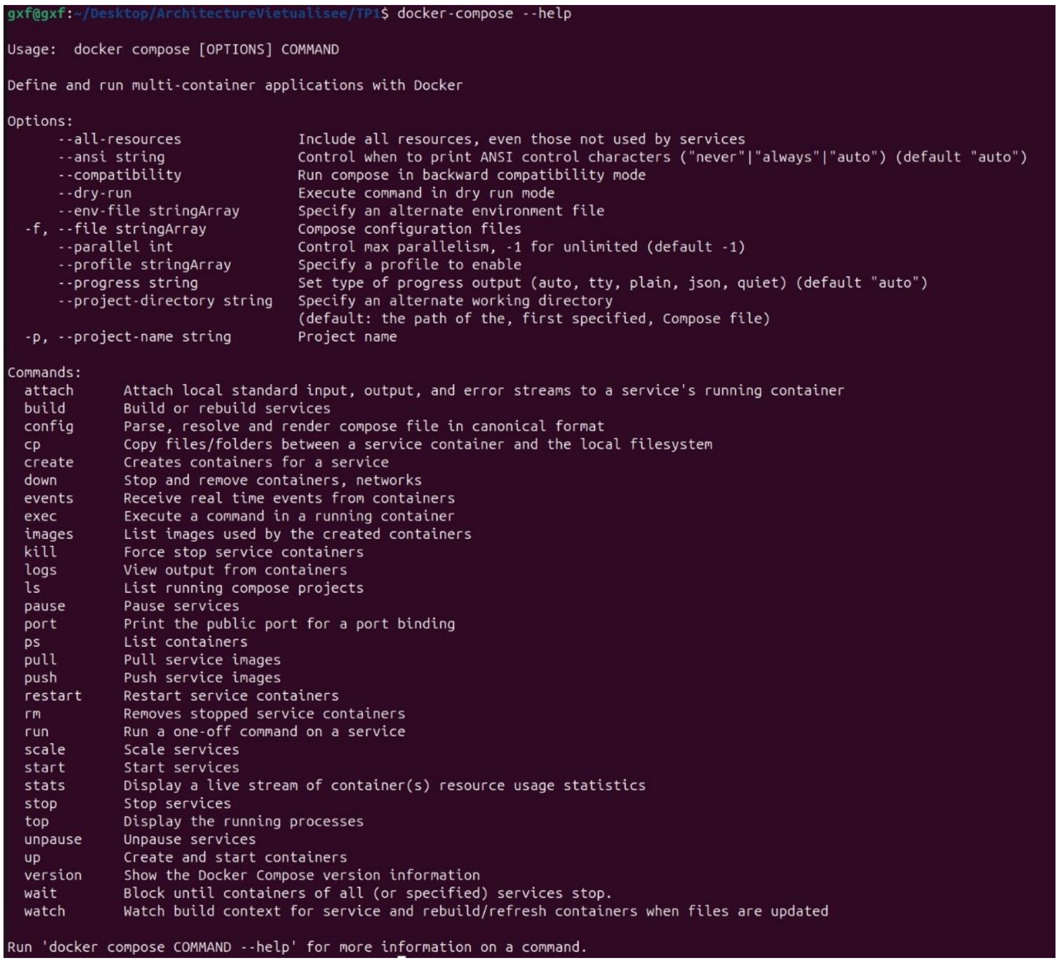
```
gxf@gxf:~/Desktop/ArchitectureVietualisee/TP1$ docker inspect -f '
{{range.NetworkSettings.Networks}}{{.IPAddress}}{{end}}' nc-server

172.17.0.2
gxf@gxf:~/Desktop/ArchitectureVietualisee/TP1$ docker run --name n
c-client -it netcat 172.17.0.2 8000
Hello World!
```

- The nc-client was able to connect to nc-server because both containers are part of the same Docker network.

- By default, Docker creates a bridge network that allows containers within that network to communicate with each other using their internal IP addresses.

4. Docker Compose



```
gxf@gxf:~/Desktop/ArchitectureVietualisee/TP1$ docker-compose --help

Usage:  docker compose [OPTIONS] COMMAND

Define and run multi-container applications with Docker

Options:
      --all-resources              Include all resources, even those not used by services
      --ansi string                Control when to print ANSI control characters ("never"|"always"|"auto") (default "auto")
      --compatibility              Run compose in backward compatibility mode
      --dry-run                    Execute command in dry run mode
      --env-file stringArray       Specify an alternate environment file
  -f, --file stringArray           Compose configuration files
      --parallel int               Control max parallelism, -1 for unlimited (default -1)
      --profile stringArray        Specify a profile to enable
      --progress string            Set type of progress output (auto, tty, plain, json, quiet) (default "auto")
      --project-directory string   Specify an alternate working directory
                                   (default: the path of the, first specified, Compose file)
  -p, --project-name string        Project name

Commands:
  attach      Attach local standard input, output, and error streams to a service's running container
  build       Build or rebuild services
  config      Parse, resolve and render compose file in canonical format
  cp          Copy files/folders between a service container and the local filesystem
  create      Creates containers for a service
  down        Stop and remove containers, networks
  events      Receive real time events from containers
  exec        Execute a command in a running container
  images      List images used by the created containers
  kill        Force stop service containers
  logs        View output from containers
  ls          List running compose projects
  pause       Pause services
  port        Print the public port for a port binding
  ps          List containers
  pull        Pull service images
  push        Push service images
  restart     Restart service containers
  rm          Removes stopped service containers
  run         Run a one-off command on a service
  scale       Scale services
  start       Start services
  stats       Display a live stream of container(s) resource usage statistics
  stop        Stop services
  top         Display the running processes
  unpause     Unpause services
  up          Create and start containers
  version     Show the Docker Compose version information
  wait        Block until containers of all (or specified) services stop.
  watch       Watch build context for service and rebuild/refresh containers when files are updated

Run 'docker compose COMMAND --help' for more information on a command.
```

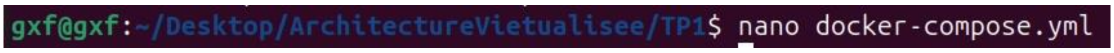
1. Which command can be used to run a service?

docker-compose up

2. Which command can be used to teardown a service?

docker-compose down

3. What does this file contain?



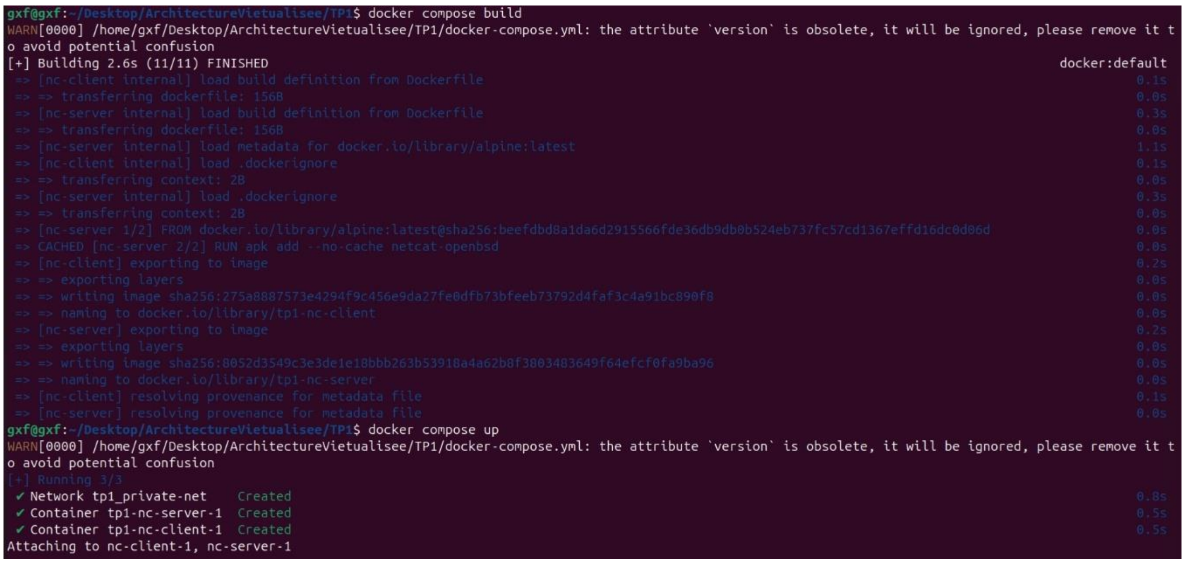
```
gxf@gxf:~/Desktop/ArchitectureVietualisee/TP1$ nano docker-compose.yml
```

It defines two services (*nc-server* and *nc-client*) that will be part of a private network called *private-net.* Both services use a custom build with an entrypoint of *sleep 60*, meaning the containers will pause for 60 seconds upon starting.

4. What part of the default image is overriden ?

The *entrypoint* is overridden. Instead of running the default command in the image, the containers are instructed to run *sleep 60*, which temporarily pauses their execution for 60 seconds.

5. What are the containers that are created by this compose file in the running containers' list?



```
gxf@gxf:~/Desktop/ArchitectureVietualisee/TP1$ docker compose build
WARN[0000] /home/gxf/Desktop/ArchitectureVietualisee/TP1/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it t
o avoid potential confusion
[+] Building 2.6s (11/11) FINISHED                                                                                              docker:default
 => [nc-client internal] load build definition from Dockerfile                                                                            0.1s
 => => transferring dockerfile: 156B                                                                                                      0.0s
 => [nc-server internal] load build definition from Dockerfile                                                                            0.3s
 => => transferring dockerfile: 156B                                                                                                      0.0s
 => [nc-server internal] load metadata for docker.io/library/alpine:latest                                                                1.1s
 => [nc-client internal] load .dockerignore                                                                                               0.1s
 => => transferring context: 2B                                                                                                           0.0s
 => [nc-server internal] load .dockerignore                                                                                               0.3s
 => => transferring context: 2B                                                                                                           0.0s
 => [nc-server 1/2] FROM docker.io/library/alpine:latest@sha256:beefdbd8a1da6d2915566fde36db9db0b524eb737fc57cd1367effd16dc0d06d           0.0s
 => CACHED [nc-server 2/2] RUN apk add --no-cache netcat-openbsd                                                                          0.0s
 => [nc-client] exporting to image                                                                                                        0.2s
 => => exporting layers                                                                                                                   0.0s
 => => writing image sha256:275a8887573e4294f9c456e9da27fe0dfb73bfeeb73792d4faf3c4a91bc890f8                                              0.0s
 => => naming to docker.io/library/tp1-nc-client                                                                                          0.0s
 => [nc-server] exporting to image                                                                                                        0.2s
 => => exporting layers                                                                                                                   0.0s
 => => writing image sha256:8052d3549c3e3de1e18bbb263b53918a4a62b8f3803483649f64efcf0fa9ba96                                              0.0s
 => => naming to docker.io/library/tp1-nc-server                                                                                          0.0s
 => [nc-client] resolving provenance for metadata file                                                                                    0.1s
 => [nc-server] resolving provenance for metadata file                                                                                    0.0s
gxf@gxf:~/Desktop/ArchitectureVietualisee/TP1$ docker compose up
WARN[0000] /home/gxf/Desktop/ArchitectureVietualisee/TP1/docker-compose.yml: the attribute `version` is obsolete, it will be ignored, please remove it t
o avoid potential confusion
[+] Running 3/3
 ✔ Network tp1_private-net    Created                                                                                                     0.8s
 ✔ Container tp1-nc-server-1  Created                                                                                                     0.5s
 ✔ Container tp1-nc-client-1  Created                                                                                                     0.5s
Attaching to nc-client-1, nc-server-1
```

- *nc-server* & *nc-client*