Version : **2022.01**

Dernière mise-à-jour : 2023/12/17 05:36

DOF601 - La Virtualisation par Isolation

Contenu du Module

Présentation de la Virtualisation par Isolation

Un isolateur est un logiciel qui permet d'isoler l'exécution des applications dans des **containers**, des **contextes** ou des **zones d'exécution**.

Historique

- **1979** - chroot - l'isolation par changement de racine,
- **2000** - BSD Jails - l'isolation en espace utilisateur,
- **2004** - Solaris Containers - l'isolation par **zones**,
- **2005** - OpenVZ - l'isolation par **partitionnement du noyau** sous Linux,
- **2008** - LXC - LinuX Containers - l'isolation en utilisant des **namespaces** et des **CGroups** avec **liblxc**,
- **2013** - Docker - l'isolation en utilisant des **namespaces** et des **CGroups** avec **libcontainer**,
- **2014** - LXD - LinuX Container Daemon - l'isolation en utilisant des **namespaces** et des **CGroups** avec **liblxc**.

Présentation des Namespaces

Les espaces de noms permettent de regrouper des processus dans un même espace et d'attribuer des droits sur des ressources par espace. Ceci permet l'exécution de plusieurs init, chacun dans un namespace, afin de recréer un environnement pour les processus qui doivent être isolés.

Présentation des CGroups

LAB #1 - cgroups v1

1.1 - Préparation

Debian 11 utilise cgroups v2 par défault. Pour revenir à l'utilisation de cgroups v1, éditez le fichier **/etc/boot/grub** et ajoutez la directive **systemd.unified_cgroup_hierarchy=0** à la ligne **GRUB_CMDLINE_LINUX_DEFAULT** :

```
root@debian11:~# vi /etc/default/grub
root@debian11:~# cat /etc/default/grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet systemd.unified_cgroup_hierarchy=0"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
```

```
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"

root@debian11:~# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found background image: /usr/share/images/desktop-base/desktop-grub.png
Found linux image: /boot/vmlinuz-5.10.0-13-amd64
Found initrd image: /boot/initrd.img-5.10.0-13-amd64
done
```

Redémarrez ensuite votre VM :

```
root@debian11:~# reboot
```

1.2 - Présentation

Les **Groupes de Contrôles** (*Control Groups*) aussi appelés **CGroups**, sont une façon de contrôler et de limiter des ressources. Les groupes de contrôle

permettent l'allocation de ressources, même d'une manière dynamique pendant que le système fonctionne, telles le temps processeur, la mémoire système, la bande réseau, ou une combinaison de ces ressources parmi des groupes de tâches (processus) définis par l'utilisateur et exécutés sur un système.

Les groupes de contrôle sont organisés de manière hiérarchique, comme des processus. Par contre, la comparaison entre les deux démontre que tandis que les processus se trouvent dans une arborescence unique descandant tous du processus init et héritant de l'environnement de leurs parents, les contrôles groupes peuvent être multiples donnant lieu à des arborescences ou **hiérarchies** multiples qui héritent de certains attributs de leurs groupes de contrôle parents.

Ces hiérarchies multiples et séparés sont necéssaires parce que chaque hiérarchie est attaché à un ou plusieurs **sous-système(s)** aussi appelés des **Contrôleurs de Ressources** ou simplement des **Contrôleurs**. Les contrôleurs disponibles sous Debian 11 sont :

- **blkio** - utilisé pour établir des limites sur l'accès des entrées/sorties à partir et depuis des périphériques blocs,
- **cpu** - utilisé pour fournir aux tâches des groupes de contrôle accès au CPU grâce au planificateur,
- **cpuacct** - utilisé pour produire des rapports automatiques sur les ressources CPU utilisées par les tâches dans un groupe de contrôle,
- **cpuset** - utilisé pour assigner des CPU individuels sur un système multicoeur et des noeuds de mémoire à des tâches dans un groupe de contrôle,
- **devices** - utilisé pour autoriser ou pour refuser l'accès des tâches aux périphériques dans un groupe de contrôle,
- **freezer** - utilisé pour suspendre ou pour réactiver les tâches dans un groupe de contrôle,
- **memory** - utilisé pour établir les limites d'utilisation de la mémoire par les tâches d'un groupe de contrôle et pour génèrer des rapports automatiques sur les ressources rmémoire utilisées par ces tâches,
- **net_cls** - utilisé pour repèrer les paquets réseau avec un identifiant de classe (*classid*) afin de permettre au contrôleur de trafic Linux, **tc**, d'identifier les paquets provenant d'une tâche particulière d'un groupe de contrôle.
- **perf_event** - utilisé pour permettre le monitoring des CGroups avec l'outil perf,
- **hugetlb** - utilisé pour limiter des ressources sur des pages de mémoire virtuelle de grande taille.

Il est à noter que :

- chaque processus du système appartient à un cgroup et seulement à un cgroup à la fois,
- tous les threads d'un processus appartiennent au même cgroup,
- à la création d'un processus, celui-ci est mis dans le même cgroup que son processus parent,
- un processus peut être migré d'un cgroup à un autre cgroup. Par contre, la migration d'un processus n'a pas d'impact sur l'appartenance au cgroup de ses processus fils.

Commencez par installer le paquet **cgroup-tools** :

```
root@debian11:~# apt -y install cgroup-tools
```

Pour visualiser les hiérarchies, il convient d'utiliser la commande **lssubsys** :

```
root@debian11:~# lssubsys -am
cpuset /sys/fs/cgroup/cpuset
cpu,cpuacct /sys/fs/cgroup/cpu,cpuacct
blkio /sys/fs/cgroup/blkio
memory /sys/fs/cgroup/memory
devices /sys/fs/cgroup/devices
freezer /sys/fs/cgroup/freezer
net_cls,net_prio /sys/fs/cgroup/net_cls,net_prio
perf_event /sys/fs/cgroup/perf_event
hugetlb /sys/fs/cgroup/hugetlb
pids /sys/fs/cgroup/pids
rdma /sys/fs/cgroup/rdma
```

Sous Debian 11, **Systemd** organise les processus dans chaque CGroup. Par exemple tous les processus démarrés par le serveur Apache se trouveront dans le même CGroup, y compris les scripts CGI. Ceci implique que la gestion des ressources en utilisant des hiérarchies est couplé avec l'arborescence des unités de Systemd.

En haut de l'arborescence des unités de Systemd se trouve la tranche root - **-.slice**, dont dépend :

- le **system.slice** - l'emplacement des services système,
- le **user.slice** - l'emplacement des sessions des utilisateurs,
- le **machine.slice** - l'emplacement des machines virtuelles et conteneurs.

En dessous des tranches peuvent se trouver :

- des **scopes** - des processus crées par **fork**,
- des **services** - des processus créés par une **Unité**.

Les slices peuvent être visualisés avec la commande suivante :

```
root@debian11:~# systemctl list-units --type=slice
  UNIT                               LOAD   ACTIVE SUB    DESCRIPTION
  -.slice                            loaded active active Root Slice
  system-getty.slice                 loaded active active system-getty.slice
  system-lvm2\x2dpvscan.slice        loaded active active system-lvm2\x2dpvscan.slice
  system-modprobe.slice              loaded active active system-modprobe.slice
  system-systemd\x2dcryptsetup.slice loaded active active Cryptsetup Units Slice
  system.slice                       loaded active active System Slice
  user-1000.slice                    loaded active active User Slice of UID 1000
  user.slice                         loaded active active User and Session Slice

LOAD   = Reflects whether the unit definition was properly loaded.
ACTIVE = The high-level unit activation state, i.e. generalization of SUB.
SUB    = The low-level unit activation state, values depend on unit type.
8 loaded units listed. Pass --all to see loaded but inactive units, too.
To show all installed unit files use 'systemctl list-unit-files'.
```

L'arborescence des unités de Systemd est la suivante :

```
root@debian11:~# systemd-cgls
Control group /:
-.slice
├─user.slice
│ └─user-1000.slice
│   ├─user@1000.service …
│   │ ├─app.slice
│   │ │ ├─pulseaudio.service
│   │ │ │ └─974 /usr/bin/pulseaudio --daemonize=no --log-target=journal
│   │ │ ├─pipewire.service
│   │ │ │ ├─973 /usr/bin/pipewire
│   │ │ │ └─984 /usr/bin/pipewire-media-session
│   │ │ └─dbus.service
│   │ │   └─982 /usr/bin/dbus-daemon --session --address=systemd: --nofork --nopidfile --systemd-activation --
syslog-only
```

```
│    │ └─init.scope
│    │   ├─958 /lib/systemd/systemd --user
│    │   └─959 (sd-pam)
│    ├─session-3.scope
│    │ ├─ 993 sshd: trainee [priv]
│    │ ├─ 999 sshd: trainee@pts/0
│    │ ├─1000 -bash
│    │ ├─1003 su -
│    │ ├─1004 -bash
│    │ ├─1010 systemd-cgls
│    │ └─1011 less
│    └─session-1.scope
│      ├─578 /bin/login -p --
│      ├─975 -bash
│      ├─986 su -
│      └─987 -bash
├─init.scope
│ └─1 /sbin/init
└─system.slice
  ├─apache2.service
  │ ├─595 /usr/sbin/apache2 -k start
  │ ├─597 /usr/sbin/apache2 -k start
  │ └─598 /usr/sbin/apache2 -k start
  ├─systemd-udevd.service
  │ └─317 /lib/systemd/systemd-udevd
  ├─cron.service
  │ └─491 /usr/sbin/cron -f
  ├─polkit.service
  │ └─495 /usr/libexec/polkitd --no-debug
  ├─rtkit-daemon.service
  │ └─979 /usr/libexec/rtkit-daemon
  ├─auditd.service
  │ └─460 /sbin/auditd
  ├─wpa_supplicant.service
```

```
  │ └─498 /sbin/wpa_supplicant -u -s -O /run/wpa_supplicant
  ├─ModemManager.service
  │ └─515 /usr/sbin/ModemManager
  ├─inetd.service
  │ └─694 /usr/sbin/inetd
  ├─systemd-journald.service
  │ └─296 /lib/systemd/systemd-journald
  ├─mdmonitor.service
  │ └─432 /sbin/mdadm --monitor --scan
  ├─ssh.service
  │ └─580 sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups
lines 1-58
[q]
```

En utilisant Systemd, plusieurs ressources peuvent être limitées :

- **CPUShares** - par défault 1024,
- **MemoryLimit** - limite exprimée en Mo ou en Go. Pas de valeur par défaut,
- **BlockIOWeight** - valeur entre 10 et 1000. Pas de valeur par défaut,
- **StartupCPUShares** - comme CPUShares mais uniquement appliqué pendant le démarrage,
- **StartupBlockIOWeight** - comme BlockIOWeight mais uniquement appliqué pendant le démarrage,
- **CPUQuota** - utilisé pour limiter le temps CPU, même quand le système ne fait rien.

1.3 - Limitation de la Mémoire

Commencez par créer le script **hello-world.sh** qui servira à générer un processus pour travailler avec les CGroups :

```
root@debian11:~# vi hello-world.sh
root@debian11:~# cat hello-world.sh
#!/bin/bash
while [ 1 ]; do
        echo "hello world"
        sleep 360
```

```
done
```

Rendez le script exécutable et testez-le :

```
root@debian11:~# chmod u+x hello-world.sh
root@debian11:~# ./hello-world.sh
hello world
^C
```

Créez maintenant un CGroup dans le sous-système **memory** appelé **helloworld** :

```
root@debian11:~# mkdir /sys/fs/cgroup/memory/helloworld
```

Par défaut, ce CGroup héritera de l'ensemble de la mémoire disponible. Pour éviter cela, créez maintenant une limite de **40000000** octets pour ce CGroup :

```
root@debian11:~# echo 40000000 > /sys/fs/cgroup/memory/helloworld/memory.limit_in_bytes
root@debian11:~# cat /sys/fs/cgroup/memory/helloworld/memory.limit_in_bytes
39997440
```

Important - Notez que les 40 000 000 demandés sont devenus 39 997 440 ce qui correspond à un nombre entier de pages mémoire du noyau de 4Ko. (39 997 440 / 4096 = 9 765).

Lancez maintenant le script **helloworld.sh** :

```
root@debian11:~# ./hello-world.sh &
[1] 1073
root@debian11:~# hello world
[Entrée]
```

```
root@debian11:~# ps aux | grep hello-world
root        1073  0.0  0.0   6756  3100 pts/0    S    06:33   0:00 /bin/bash ./hello-world.sh
root        1077  0.0  0.0   6180   712 pts/0    R+   06:34   0:00 grep hello-world
```

Notez qu'il n'y a pas de limite de la mémoire, ce qui implique l'héritage par défaut :

```
root@debian11:~# ps -ww -o cgroup 1073
CGROUP
8:devices:/user.slice,7:pids:/user.slice/user-1000.slice/session-3.scope,5:memory:/user.slice/user-1000.slice/ses
sion-3.scope,1:name=systemd:/user.slice/user-1000.slice/session-3.scope,0::/user.slice/user-1000.slice/session-3.
scope
```

Insérer le PID de notre script dans le CGroup **helloworld** :

```
root@debian11:~# echo 1073 > /sys/fs/cgroup/memory/helloworld/cgroup.procs
```

Notez maintenant l'héritage de la limitation de la mémoire - **5:memory:/helloworld** :

```
root@debian11:~# ps -ww -o cgroup 1073
CGROUP
8:devices:/user.slice,7:pids:/user.slice/user-1000.slice/session-3.scope,5:memory:/helloworld,1:name=systemd:/use
r.slice/user-1000.slice/session-3.scope,0::/user.slice/user-1000.slice/session-3.scope
```

Constatez ensuite l'occupation mémoire réelle :

```
root@debian11:~# cat /sys/fs/cgroup/memory/helloworld/memory.usage_in_bytes
274432
```

Tuez le script **hello-world.sh** :

```
root@debian11:~# kill 1073
root@debian11:~# ps aux | grep hello-world
root        1086  0.0  0.0   6180   716 pts/0    S+   06:37   0:00 grep hello-world
```

```
[1]+  Terminated              ./hello-world.sh
```

Créez un second CGroup beaucoup plus restrictif :

```
root@debian11:~# mkdir /sys/fs/cgroup/memory/helloworld1
root@debian11:~# echo 6000 > /sys/fs/cgroup/memory/helloworld1/memory.limit_in_bytes
root@debian11:~# cat /sys/fs/cgroup/memory/helloworld1/memory.limit_in_bytes
4096
```

Relancez le script **hello-world.sh** et insérez-le dans le nouveau CGroup :

```
root@debian11:~# ./hello-world.sh &
[1] 1089

root@debian11:~# hello world
[Entrée]

root@debian11:~# echo 1089 > /sys/fs/cgroup/memory/helloworld1/cgroup.procs
```

Attendez la prochaine sortie de **hello world** sur le canal standard puis constatez que le script s'arrête :

```
root@debian11:~# ps aux | grep hello-world
root        1100  0.0  0.0   6180   720 pts/0    S+   06:45   0:00 grep hello-world
[1]+  Killed                  ./hello-world.sh
```

Notez la trace dans le fichier **/var/log/messages** :

```
root@debian11:~# tail /var/log/messages
May  4 06:44:43 debian11 kernel: [  994.012423] workingset_nodereclaim 0
May  4 06:44:43 debian11 kernel: [  994.012423] pgfault 0
May  4 06:44:43 debian11 kernel: [  994.012423] pgmajfault 0
May  4 06:44:43 debian11 kernel: [  994.012423] pgrefill 0
May  4 06:44:43 debian11 kernel: [  994.012423] pgscan 0
May  4 06:44:43 debian11 kernel: [  994.012423] pgsteal 0
```

```
May  4 06:44:43 debian11 kernel: [  994.012425] Tasks state (memory values in pages):
May  4 06:44:43 debian11 kernel: [  994.012426] [  pid  ]   uid  tgid total_vm      rss pgtables_bytes swapents
oom_score_adj name
May  4 06:44:43 debian11 kernel: [  994.012428] [   1089]     0  1089     1689      780    53248        0
0 hello-world.sh
May  4 06:44:43 debian11 kernel: [  994.012430] oom-
kill:constraint=CONSTRAINT_MEMCG,nodemask=(null),cpuset=/,mems_allowed=0,oom_memcg=/helloworld1,task_memcg=/hello
world1,task=hello-world.sh,pid=1089,uid=0
```

1.4 - La Commande cgcreate

Cette commande permet la création d'un CGroup :

```
root@debian11:~# cgcreate -g memory:helloworld2

root@debian11:~# ls -l /sys/fs/cgroup/memory/helloworld2/
total 0
-rw-r--r-- 1 root root 0 May  4 06:47 cgroup.clone_children
--w--w--w- 1 root root 0 May  4 06:47 cgroup.event_control
-rw-r--r-- 1 root root 0 May  4 06:47 cgroup.procs
-rw-r--r-- 1 root root 0 May  4 06:47 memory.failcnt
--w------- 1 root root 0 May  4 06:47 memory.force_empty
-rw-r--r-- 1 root root 0 May  4 06:47 memory.kmem.failcnt
-rw-r--r-- 1 root root 0 May  4 06:47 memory.kmem.limit_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:47 memory.kmem.max_usage_in_bytes
-r--r--r-- 1 root root 0 May  4 06:47 memory.kmem.slabinfo
-rw-r--r-- 1 root root 0 May  4 06:47 memory.kmem.tcp.failcnt
-rw-r--r-- 1 root root 0 May  4 06:47 memory.kmem.tcp.limit_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:47 memory.kmem.tcp.max_usage_in_bytes
-r--r--r-- 1 root root 0 May  4 06:47 memory.kmem.tcp.usage_in_bytes
-r--r--r-- 1 root root 0 May  4 06:47 memory.kmem.usage_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:47 memory.limit_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:47 memory.max_usage_in_bytes
```

```
-rw-r--r-- 1 root root 0 May  4 06:47 memory.memsw.failcnt
-rw-r--r-- 1 root root 0 May  4 06:47 memory.memsw.limit_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:47 memory.memsw.max_usage_in_bytes
-r--r--r-- 1 root root 0 May  4 06:47 memory.memsw.usage_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:47 memory.move_charge_at_immigrate
-r--r--r-- 1 root root 0 May  4 06:47 memory.numa_stat
-rw-r--r-- 1 root root 0 May  4 06:47 memory.oom_control
---------- 1 root root 0 May  4 06:47 memory.pressure_level
-rw-r--r-- 1 root root 0 May  4 06:47 memory.soft_limit_in_bytes
-r--r--r-- 1 root root 0 May  4 06:47 memory.stat
-rw-r--r-- 1 root root 0 May  4 06:47 memory.swappiness
-r--r--r-- 1 root root 0 May  4 06:47 memory.usage_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:47 memory.use_hierarchy
-rw-r--r-- 1 root root 0 May  4 06:47 notify_on_release
-rw-r--r-- 1 root root 0 May  4 06:47 tasks
```

Il n'existe cependant pas de commande pour affecter une limitation de la mémoire :

```
root@debian11:~# echo 40000000 > /sys/fs/cgroup/memory/helloworld2/memory.limit_in_bytes
```

1.5 - La Commande cgexec

Cette commande permet d'insérer la limitation dans le CGroup **et** de lancer le script en une seule ligne :

```
root@debian11:~# cgexec -g memory:helloworld2 ./hello-world.sh &
[1] 1106

root@debian11:~# hello world
[Entrée]

root@debian11:~# cat /sys/fs/cgroup/memory/helloworld2/cgroup.procs
1106
```

```
1107
root@debian11:~# ps aux | grep 110
root        1106  0.0  0.0   6756  3060 pts/0    S    06:48   0:00 /bin/bash ./hello-world.sh
root        1107  0.0  0.0   5304   508 pts/0    S    06:48   0:00 sleep 360
root        1108  0.0  0.0      0     0 ?        I    06:49   0:00 [kworker/1:0-events_freezable]
root        1113  0.0  0.0   6180   652 pts/0    S+   06:50   0:00 grep 110
```

1.6 - La Commande cgdelete

Une fois le script terminé, cette commande permet de supprimer le cgroup :

```
root@debian11:~# kill 1106
root@debian11:~# ps aux | grep 110
root        1107  0.0  0.0   5304   508 pts/0    S    06:48   0:00 sleep 360
root        1108  0.0  0.0      0     0 ?        I    06:49   0:00 [kworker/1:0-mm_percpu_wq]
root        1115  0.0  0.0   6180   716 pts/0    R+   06:51   0:00 grep 110
[1]+  Terminated              cgexec -g memory:helloworld2 ./hello-world.sh

root@debian11:~# cgdelete memory:helloworld2

root@debian11:~# ls -l /sys/fs/cgroup/memory/helloworld2/
ls: cannot access '/sys/fs/cgroup/memory/helloworld2/': No such file or directory
```

1.7 - Le Fichier /etc/cgconfig.conf

Afin de les rendre persistants, il convient d'éditer le fichier **/etc/cgconfig.conf** :

```
root@debian11:~# vi /etc/cgconfig.conf
root@debian11:~# cat /etc/cgconfig.conf
group helloworld2 {
        cpu {
```

```
                cpu.shares = 100;
        }
        memory {
                memory.limit_in_bytes = 40000;
        }
}
```

Important - Notez la création de **deux** limitations, une de 40 000 octets de mémoire et l'autre de **100 cpu.shares**. Cette dernière est une valeur exprimée sur 1 024, où 1 024 représente 100% du temps CPU. La limite fixée est donc équivalente à 9,77% du temps CPU.

Créez donc les deux CGroups concernés :

```
root@debian11:~# cgcreate -g memory:helloworld2

root@debian11:~# ls -l /sys/fs/cgroup/memory/helloworld2/
total 0
-rw-r--r-- 1 root root 0 May  4 06:53 cgroup.clone_children
--w--w--w- 1 root root 0 May  4 06:53 cgroup.event_control
-rw-r--r-- 1 root root 0 May  4 06:53 cgroup.procs
-rw-r--r-- 1 root root 0 May  4 06:53 memory.failcnt
--w------- 1 root root 0 May  4 06:53 memory.force_empty
-rw-r--r-- 1 root root 0 May  4 06:53 memory.kmem.failcnt
-rw-r--r-- 1 root root 0 May  4 06:53 memory.kmem.limit_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:53 memory.kmem.max_usage_in_bytes
-r--r--r-- 1 root root 0 May  4 06:53 memory.kmem.slabinfo
-rw-r--r-- 1 root root 0 May  4 06:53 memory.kmem.tcp.failcnt
-rw-r--r-- 1 root root 0 May  4 06:53 memory.kmem.tcp.limit_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:53 memory.kmem.tcp.max_usage_in_bytes
-r--r--r-- 1 root root 0 May  4 06:53 memory.kmem.tcp.usage_in_bytes
```

```
-r--r--r-- 1 root root 0 May  4 06:53 memory.kmem.usage_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:53 memory.limit_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:53 memory.max_usage_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:53 memory.memsw.failcnt
-rw-r--r-- 1 root root 0 May  4 06:53 memory.memsw.limit_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:53 memory.memsw.max_usage_in_bytes
-r--r--r-- 1 root root 0 May  4 06:53 memory.memsw.usage_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:53 memory.move_charge_at_immigrate
-r--r--r-- 1 root root 0 May  4 06:53 memory.numa_stat
-rw-r--r-- 1 root root 0 May  4 06:53 memory.oom_control
---------- 1 root root 0 May  4 06:53 memory.pressure_level
-rw-r--r-- 1 root root 0 May  4 06:53 memory.soft_limit_in_bytes
-r--r--r-- 1 root root 0 May  4 06:53 memory.stat
-rw-r--r-- 1 root root 0 May  4 06:53 memory.swappiness
-r--r--r-- 1 root root 0 May  4 06:53 memory.usage_in_bytes
-rw-r--r-- 1 root root 0 May  4 06:53 memory.use_hierarchy
-rw-r--r-- 1 root root 0 May  4 06:53 notify_on_release
-rw-r--r-- 1 root root 0 May  4 06:53 tasks
```

```
root@debian11:~# cgcreate -g cpu:helloworld2

root@debian11:~# ls -l /sys/fs/cgroup/cpu/helloworld2/
total 0
-rw-r--r-- 1 root root 0 May  4 06:54 cgroup.clone_children
-rw-r--r-- 1 root root 0 May  4 06:54 cgroup.procs
-r--r--r-- 1 root root 0 May  4 06:54 cpuacct.stat
-rw-r--r-- 1 root root 0 May  4 06:54 cpuacct.usage
-r--r--r-- 1 root root 0 May  4 06:54 cpuacct.usage_all
-r--r--r-- 1 root root 0 May  4 06:54 cpuacct.usage_percpu
-r--r--r-- 1 root root 0 May  4 06:54 cpuacct.usage_percpu_sys
-r--r--r-- 1 root root 0 May  4 06:54 cpuacct.usage_percpu_user
-r--r--r-- 1 root root 0 May  4 06:54 cpuacct.usage_sys
-r--r--r-- 1 root root 0 May  4 06:54 cpuacct.usage_user
-rw-r--r-- 1 root root 0 May  4 06:54 cpu.cfs_period_us
```

```
-rw-r--r-- 1 root root 0 May  4 06:54 cpu.cfs_quota_us
-rw-r--r-- 1 root root 0 May  4 06:54 cpu.shares
-r--r--r-- 1 root root 0 May  4 06:54 cpu.stat
-rw-r--r-- 1 root root 0 May  4 06:54 notify_on_release
-rw-r--r-- 1 root root 0 May  4 06:54 tasks
```

1.8 - La Commande cgconfigparser

Appliquez le contenu du fichier **/etc/cgconfig.conf** grâce à l'utilisation de la commande **cgconfigparser** :

```
root@debian11:~# cgconfigparser -l /etc/cgconfig.conf

root@debian11:~# cat /sys/fs/cgroup/memory/helloworld2/memory.limit_in_bytes
36864

root@debian11:~# cat /sys/fs/cgroup/cpu/helloworld2/cpu.shares
100
```

LAB #2 - cgroups v2

2.1 - Préparation

Pour revenir à l'utilisation de cgroups v2, éditez le fichier **/etc/boot/grub** et modifiez la directive **systemd.unified_cgroup_hierarchy=0** à **systemd.unified_cgroup_hierarchy=1** dans la ligne **GRUB_CMDLINE_LINUX_DEFAULT** :

```
root@debian11:~# vi /etc/default/grub
root@debian11:~# cat /etc/default/grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'
```

```
GRUB_DEFAULT=0
GRUB_TIMEOUT=5
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet systemd.unified_cgroup_hierarchy=1"
GRUB_CMDLINE_LINUX=""

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"

# Uncomment to disable graphical terminal (grub-pc only)
#GRUB_TERMINAL=console

# The resolution used on graphical terminal
# note that you can use only modes which your graphic card supports via VBE
# you can see them in real GRUB with the command `vbeinfo'
#GRUB_GFXMODE=640x480

# Uncomment if you don't want GRUB to pass "root=UUID=xxx" parameter to Linux
#GRUB_DISABLE_LINUX_UUID=true

# Uncomment to disable generation of recovery mode menu entries
#GRUB_DISABLE_RECOVERY="true"

# Uncomment to get a beep at grub start
#GRUB_INIT_TUNE="480 440 1"

root@debian11:~# grub-mkconfig -o /boot/grub/grub.cfg
Generating grub configuration file ...
Found background image: /usr/share/images/desktop-base/desktop-grub.png
Found linux image: /boot/vmlinuz-5.10.0-13-amd64
Found initrd image: /boot/initrd.img-5.10.0-13-amd64
done
```

Redémarrez ensuite votre VM :

```
root@debian11:~# reboot
```

2.2 - Présentation

A l'opposé des cgroups v1, cgroup v2 n'a qu'une seule arborescence ou hiérarchie et donc un seul point de montage. Tous les contrôleurs compatibles v2 qui ne sont pas liés à une hiérarchie v1 sont automatiquement liés à la hiérarchie v2. Un contrôleur inactif dans la hiérarchie v2 peut être lié à un autre hiérarchie. La migration d'un contrôleur d'une hiérarchie à une autre hiérarchie n'est possible que dans le cas où le contrôleur est désactivé et n'est plus référencé dans la hiérarchie d'origine.

Pour vérifier l'utilisation de cgroups v2, il convient de visualiser le point de montage :

```
root@debian11:~# mount -l | grep cgroup
cgroup2 on /sys/fs/cgroup type cgroup2 (rw,nosuid,nodev,noexec,relatime,nsdelegate,memory_recursiveprot)
```

et de consulter le contenu de ce point de montage :

```
root@debian11:~# ls -l /sys/fs/cgroup/
total 0
-r--r--r--  1 root root 0 Jul  6 10:58 cgroup.controllers
-rw-r--r--  1 root root 0 Jul  6 11:32 cgroup.max.depth
-rw-r--r--  1 root root 0 Jul  6 11:32 cgroup.max.descendants
-rw-r--r--  1 root root 0 Jul  6 10:58 cgroup.procs
-r--r--r--  1 root root 0 Jul  6 11:32 cgroup.stat
-rw-r--r--  1 root root 0 Jul  6 10:58 cgroup.subtree_control
-rw-r--r--  1 root root 0 Jul  6 11:32 cgroup.threads
-rw-r--r--  1 root root 0 Jul  6 11:32 cpu.pressure
-r--r--r--  1 root root 0 Jul  6 11:32 cpuset.cpus.effective
-r--r--r--  1 root root 0 Jul  6 11:32 cpuset.mems.effective
-r--r--r--  1 root root 0 Jul  6 11:32 cpu.stat
drwxr-xr-x  2 root root 0 Jul  6 10:58 dev-hugepages.mount
drwxr-xr-x  2 root root 0 Jul  6 10:58 dev-mqueue.mount
```

```
drwxr-xr-x  2 root root 0 Jul  6 10:58 init.scope
-rw-r--r--  1 root root 0 Jul  6 11:32 io.cost.model
-rw-r--r--  1 root root 0 Jul  6 11:32 io.cost.qos
-rw-r--r--  1 root root 0 Jul  6 11:32 io.pressure
-r--r--r--  1 root root 0 Jul  6 11:32 io.stat
-r--r--r--  1 root root 0 Jul  6 11:32 memory.numa_stat
-rw-r--r--  1 root root 0 Jul  6 11:32 memory.pressure
-r--r--r--  1 root root 0 Jul  6 11:32 memory.stat
drwxr-xr-x  2 root root 0 Jul  6 10:58 sys-fs-fuse-connections.mount
drwxr-xr-x  2 root root 0 Jul  6 10:58 sys-kernel-config.mount
drwxr-xr-x  2 root root 0 Jul  6 10:58 sys-kernel-debug.mount
drwxr-xr-x  2 root root 0 Jul  6 10:58 sys-kernel-tracing.mount
drwxr-xr-x 23 root root 0 Jul  6 11:26 system.slice
drwxr-xr-x  4 root root 0 Jul  6 11:30 user.slice
```

Dans la version 2 de cgroup, certains noms ont changé par rapport à ceux utilisés dans la version 1 :

Version 1	Version 2
CPUShares	CPUWeight
StartupCPUShares	StartupCPUWeight
MemoryLimit	MemoryMax

Commencez par créer le cgroup enfant **pids** dans le cgroup racine :

```
root@debian11:~# mkdir /sys/fs/cgroup/pids
```

Placez le PID du terminal courant dans le fichier **cgroup.procs** du cgroup enfant :

```
root@debian11:~# echo $$
1230
root@debian11:~# echo $$ > /sys/fs/cgroup/pids/cgroup.procs
```

Contrôlez maintenant le contenu du fichier cgroup.procs ainsi que le nombre de PIDs dans le cgroup **pids** :

```
root@debian11:~# cat /sys/fs/cgroup/pids/cgroup.procs
1230
1281

root@debian11:~# cat /sys/fs/cgroup/pids/pids.current
2
```

Important - Notez que le fichier cgroup.procs contient **deux** PIDs. Le premier est celui du Shell tandis que le deuxième est celui de la commande cat.

Injectez maintenant la valeur de **5** dans le fichier **pids.max** du cgroup **pids** :

```
root@debian11:~# echo 5 > /sys/fs/cgroup/pids/pids.max
```

Lancez la commande suivante pour créer 6 pids dans le cgroup :

```
root@debian11:~# for a in $(seq 1 5); do sleep 60 & done
[1] 1290
[2] 1291
[3] 1292
[4] 1293
-bash: fork: retry: Resource temporarily unavailable
-bash: fork: retry: Resource temporarily unavailable
-bash: fork: retry: Resource temporarily unavailable
-bash: fork: retry: Resource temporarily unavailable
-bash: fork: Resource temporarily unavailable
```

Important - Notez qu'à la tentative de création du 6ème processus, une

erreur est retournée. Le système tente ensuite 4 fois de plus puis renonce finalement avec le message d'erreur **-bash: fork: Resource temporarily unavailable**.

Dernièrement, essayez de supprimer le cgroup **pids** :

```
root@debian11:~# rmdir /sys/fs/cgroup/pids
rmdir: failed to remove '/sys/fs/cgroup/pids': Device or resource busy
```

Important - Notez qu'il n'est pas possible de supprimer un cgroup tant que celui-ci contient un processus.

Déplacez le processus du terminal courant dans le cgroup racine :

```
root@debian11:~# echo $$ > /sys/fs/cgroup/cgroup.procs
```

Il est maintenant possible de supprimer le cgroup **pids** :

```
root@debian11:~# rmdir /sys/fs/cgroup/pids
root@debian11:~#
```

2.3 - Limitation de la CPU

Il existe deux façons de limiter les ressources de la CPU :

- **CPU bandwidth**,
  - un système de limitation basé sur un pourcentage de CPU pour un ou plusieurs processus,
- **CPU weight**,

- un système de limitation basé sur la prioritisassion d'un ou de plusieurs processus par rapports aux autres processus.

Dans l'exemple suivant, vous allez mettre en place une limite de type **CPU bandwidth**.

Commencez par créer un service appelé **foo** :

```
root@debian11:~# vi /lib/systemd/system/foo.service
root@debian11:~# cat /lib/systemd/system/foo.service
[Unit]
Description=The foo service that does nothing useful
After=remote-fs.target nss-lookup.target

[Service]
ExecStart=/usr/bin/sha1sum /dev/zero
ExecStop=/bin/kill -WINCH ${MAINPID}

[Install]
WantedBy=multi-user.target
```

Démarrez et activez le service :

```
root@debian11:~# systemctl start foo.service
root@debian11:~# systemctl enable foo.service
Created symlink /etc/systemd/system/multi-user.target.wants/foo.service → /lib/systemd/system/foo.service.
root@debian11:~# systemctl status foo.service
● foo.service - The foo service that does nothing useful
     Loaded: loaded (/lib/systemd/system/foo.service; enabled; vendor preset: enabled)
     Active: active (running) since Wed 2022-07-06 11:41:18 CEST; 19s ago
   Main PID: 997 (sha1sum)
      Tasks: 1 (limit: 19155)
     Memory: 296.0K
        CPU: 19.114s
     CGroup: /system.slice/foo.service
             └─997 /usr/bin/sha1sum /dev/zero
```

```
Jul 06 11:41:18 debian11 systemd[1]: Started The foo service that does nothing useful.
```

Utilisez la commande **ps** pour voir le pourcentage de la CPU utilisé par ce service :

```
root@debian11:~# ps -p 997 -o pid,comm,cputime,%cpu
    PID COMMAND             TIME %CPU
    997 sha1sum         00:01:33  100
```

Créez maintenant un autre service dénommé **bar** :

```
root@debian11:~# vi /lib/systemd/system/bar.service
root@debian11:~# cat /lib/systemd/system/bar.service
[Unit]
Description=The bar service that does nothing useful
After=remote-fs.target nss-lookup.target

[Service]
ExecStart=/usr/bin/md5sum /dev/zero
ExecStop=/bin/kill -WINCH ${MAINPID}

[Install]
WantedBy=multi-user.target
```

Démarrez et activez le service :

```
root@debian11:~# systemctl start bar.service

root@debian11:~# systemctl enable bar.service

Created symlink /etc/systemd/system/multi-user.target.wants/bar.service → /lib/systemd/system/bar.service.

root@debian11:~# systemctl status bar.service
● bar.service - The bar service that does nothing useful
     Loaded: loaded (/lib/systemd/system/bar.service; enabled; vendor preset: enabled)
```

```
     Active: active (running) since Wed 2022-07-06 11:45:24 CEST; 15s ago
   Main PID: 1020 (md5sum)
      Tasks: 1 (limit: 19155)
     Memory: 236.0K
        CPU: 15.079s
     CGroup: /system.slice/bar.service
             └─1020 /usr/bin/md5sum /dev/zero

Jul 06 11:45:24 debian11 systemd[1]: Started The bar service that does nothing useful.
```

Utilisez la commande **ps** pour voir le pourcentage de la CPU utilisé par ce service :

```
root@debian11:~# ps -p 1020 -o pid,comm,cputime,%cpu
    PID COMMAND             TIME %CPU
   1020 md5sum          00:01:03 99.4
```

Vérifiez maintenant la présence des contrôleurs **cpuset** et **cpu** dans l'arborescence du cgroup racine qui est monté à **/sys/fs/cgroup/** :

```
root@debian11:~# cat /sys/fs/cgroup/cgroup.controllers
cpuset cpu io memory hugetlb pids rdma
```

Activez maintenant les deux contrôleurs **cpuset** et **cpu** :

```
root@debian11:~# cat /sys/fs/cgroup/cgroup.subtree_control
memory pids

root@debian11:~# echo "+cpu" >> /sys/fs/cgroup/cgroup.subtree_control

root@debian11:~# echo "+cpuset" >> /sys/fs/cgroup/cgroup.subtree_control

root@debian11:~# cat /sys/fs/cgroup/cgroup.subtree_control
cpuset cpu memory pids
```

Créez le cgroup **enfant** appelé **FooBar** :

```
root@debian11:~# mkdir /sys/fs/cgroup/FooBar/

root@debian11:~# ls -l /sys/fs/cgroup/FooBar/
total 0
-r--r--r-- 1 root root 0 Jul  6 12:18 cgroup.controllers
-r--r--r-- 1 root root 0 Jul  6 12:18 cgroup.events
-rw-r--r-- 1 root root 0 Jul  6 12:18 cgroup.freeze
-rw-r--r-- 1 root root 0 Jul  6 12:18 cgroup.max.depth
-rw-r--r-- 1 root root 0 Jul  6 12:18 cgroup.max.descendants
-rw-r--r-- 1 root root 0 Jul  6 12:18 cgroup.procs
-r--r--r-- 1 root root 0 Jul  6 12:18 cgroup.stat
-rw-r--r-- 1 root root 0 Jul  6 12:18 cgroup.subtree_control
-rw-r--r-- 1 root root 0 Jul  6 12:18 cgroup.threads
-rw-r--r-- 1 root root 0 Jul  6 12:18 cgroup.type
-rw-r--r-- 1 root root 0 Jul  6 12:18 cpu.max
-rw-r--r-- 1 root root 0 Jul  6 12:18 cpu.pressure
-rw-r--r-- 1 root root 0 Jul  6 12:18 cpuset.cpus
-r--r--r-- 1 root root 0 Jul  6 12:18 cpuset.cpus.effective
-rw-r--r-- 1 root root 0 Jul  6 12:18 cpuset.cpus.partition
-rw-r--r-- 1 root root 0 Jul  6 12:18 cpuset.mems
-r--r--r-- 1 root root 0 Jul  6 12:18 cpuset.mems.effective
-r--r--r-- 1 root root 0 Jul  6 12:18 cpu.stat
-rw-r--r-- 1 root root 0 Jul  6 12:18 cpu.weight
-rw-r--r-- 1 root root 0 Jul  6 12:18 cpu.weight.nice
-rw-r--r-- 1 root root 0 Jul  6 12:18 io.pressure
-r--r--r-- 1 root root 0 Jul  6 12:18 memory.current
-r--r--r-- 1 root root 0 Jul  6 12:18 memory.events
-r--r--r-- 1 root root 0 Jul  6 12:18 memory.events.local
-rw-r--r-- 1 root root 0 Jul  6 12:18 memory.high
-rw-r--r-- 1 root root 0 Jul  6 12:18 memory.low
-rw-r--r-- 1 root root 0 Jul  6 12:18 memory.max
-rw-r--r-- 1 root root 0 Jul  6 12:18 memory.min
-r--r--r-- 1 root root 0 Jul  6 12:18 memory.numa_stat
-rw-r--r-- 1 root root 0 Jul  6 12:18 memory.oom.group
```

```
-rw-r--r-- 1 root root 0 Jul  6 12:18 memory.pressure
-r--r--r-- 1 root root 0 Jul  6 12:18 memory.stat
-r--r--r-- 1 root root 0 Jul  6 12:18 memory.swap.current
-r--r--r-- 1 root root 0 Jul  6 12:18 memory.swap.events
-rw-r--r-- 1 root root 0 Jul  6 12:18 memory.swap.high
-rw-r--r-- 1 root root 0 Jul  6 12:18 memory.swap.max
-r--r--r-- 1 root root 0 Jul  6 12:18 pids.current
-r--r--r-- 1 root root 0 Jul  6 12:18 pids.events
-rw-r--r-- 1 root root 0 Jul  6 12:18 pids.max
```

Activez les contrôleurs **cpuset** et **cpu** pour le cgroup **FooBar** :

```
root@debian11:~# echo "+cpu" >> /sys/fs/cgroup/FooBar/cgroup.subtree_control

root@debian11:~# echo "+cpuset" >> /sys/fs/cgroup/FooBar/cgroup.subtree_control

root@debian11:~# cat /sys/fs/cgroup/cgroup.subtree_control /sys/fs/cgroup/FooBar/cgroup.subtree_control
cpuset cpu memory pids
cpuset cpu
```

Important - Notez qu'il n'est pas possible d'activer les contrôleurs pour un cgroup enfant si ces mêmes contrôleurs ne sont pas déjà activés pour le cgroup parent. Notez aussi que dans le cgroup **FooBar**, les contrôleurs **memory** et **pids** ne sont **pas** activés.

Créez maintenant le répertoire **/sys/fs/cgroup/FooBar/tasks** :

```
root@debian11:~# mkdir /sys/fs/cgroup/FooBar/tasks
root@debian11:~# ls -l /sys/fs/cgroup/FooBar/tasks
total 0
-r--r--r-- 1 root root 0 Jul  6 12:20 cgroup.controllers
```

```
-r--r--r-- 1 root root 0 Jul  6 12:20 cgroup.events
-rw-r--r-- 1 root root 0 Jul  6 12:20 cgroup.freeze
-rw-r--r-- 1 root root 0 Jul  6 12:20 cgroup.max.depth
-rw-r--r-- 1 root root 0 Jul  6 12:20 cgroup.max.descendants
-rw-r--r-- 1 root root 0 Jul  6 12:20 cgroup.procs
-r--r--r-- 1 root root 0 Jul  6 12:20 cgroup.stat
-rw-r--r-- 1 root root 0 Jul  6 12:20 cgroup.subtree_control
-rw-r--r-- 1 root root 0 Jul  6 12:20 cgroup.threads
-rw-r--r-- 1 root root 0 Jul  6 12:20 cgroup.type
-rw-r--r-- 1 root root 0 Jul  6 12:20 cpu.max
-rw-r--r-- 1 root root 0 Jul  6 12:20 cpu.pressure
-rw-r--r-- 1 root root 0 Jul  6 12:20 cpuset.cpus
-r--r--r-- 1 root root 0 Jul  6 12:20 cpuset.cpus.effective
-rw-r--r-- 1 root root 0 Jul  6 12:20 cpuset.cpus.partition
-rw-r--r-- 1 root root 0 Jul  6 12:20 cpuset.mems
-r--r--r-- 1 root root 0 Jul  6 12:20 cpuset.mems.effective
-r--r--r-- 1 root root 0 Jul  6 12:20 cpu.stat
-rw-r--r-- 1 root root 0 Jul  6 12:20 cpu.weight
-rw-r--r-- 1 root root 0 Jul  6 12:20 cpu.weight.nice
-rw-r--r-- 1 root root 0 Jul  6 12:20 io.pressure
-rw-r--r-- 1 root root 0 Jul  6 12:20 memory.pressure
```

Important - Le répertoire **/sys/fs/cgroup/FooBar/tasks** définit un groupe *enfant* du cgroup FooBar qui ne concerne que les contrôleurs **cpuset** et **cpu**.

De façon à ce que les deux processus issus des services **foo** et **bar** se font concurrence sur la même CPU, injectez la valeur de **1** dans le fichier **/sys/fs/cgroup/FooBar/tasks/cpuset.cpus** :

```
root@debian11:~# echo "1" > /sys/fs/cgroup/FooBar/tasks/cpuset.cpus
```

```
root@debian11:~# cat /sys/fs/cgroup/FooBar/tasks/cpuset.cpus
1
```

Important - Notez que dans les faits, le contrôleur **cpu** n'est activé **que** dans le cas où le cgroup contient au moins 2 processus qui se font concurrence sur la même CPU.

Mettez en place une limitation des ressources de la CPU avec la commande suivante :

```
root@debian11:~# echo "200000 1000000" > /sys/fs/cgroup/FooBar/tasks/cpu.max
```

Important - Dans la commande ci-dessus, le premier nombre est un quota en microsecondes pendant lequel les processus dans le cgroup peuvent s'exécuter dans une **période** de temps donnée. Le deuxième nombre, également exprimé en microsecondes, et la **période**. Autrement dit, les processus dans le cgroup seront limités à une exécution de 200 000 / 1 000 000 = 0.2 secondes pendant chaque seconde.

Ajoutez maintenant les processus des services **foo** et **bar** au cgroup **FooBar** :

```
echo "997" > /sys/fs/cgroup/FooBar/tasks/cgroup.procs

echo "1020" > /sys/fs/cgroup/FooBar/tasks/cgroup.procs
```

Vérifiez la prise en compte par le système de la commande précédente :

```
root@debian11:~# cat /proc/997/cgroup /proc/1020/cgroup
0::/FooBar/tasks
```

```
0::/FooBar/tasks
```

Dernièrement, utilisez la commande **top** pour constater que la consommation de la CPU et limitée à 20% sur l'ensemble des processus du cgroup **FooBar et** que ces 20% sont répartis en parts égales sur les deux processus **foo** et **bar** :

```
top - 12:36:33 up  1:37,  2 users,  load average: 0.01, 0.70, 1.39
Tasks: 154 total,   3 running, 151 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.5 us,  0.0 sy,  0.0 ni, 97.5 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :  16007.9 total,  15503.7 free,    203.6 used,    300.6 buff/cache
MiB Swap:    975.0 total,    975.0 free,      0.0 used.  15536.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
    997 root      20   0    5312    572    508 R  10.0   0.0  50:12.26 sha1sum
   1020 root      20   0    5308    508    444 R  10.0   0.0  47:00.56 md5sum
```

2.4 - La Commande systemctl set-property

Comme déjà vu, systemd organise les processus dans des **slices**, par exemple les utilisateurs sont regroupés dans **/sys/fs/cgroup/user.slice** :

```
root@debian11:~# ls -l /sys/fs/cgroup/user.slice
total 0
-r--r--r-- 1 root root 0 Jul  6 16:13 cgroup.controllers
-r--r--r-- 1 root root 0 Jul  6 10:58 cgroup.events
-rw-r--r-- 1 root root 0 Jul  6 16:13 cgroup.freeze
-rw-r--r-- 1 root root 0 Jul  6 16:13 cgroup.max.depth
-rw-r--r-- 1 root root 0 Jul  6 16:13 cgroup.max.descendants
-rw-r--r-- 1 root root 0 Jul  6 16:13 cgroup.procs
-r--r--r-- 1 root root 0 Jul  6 16:13 cgroup.stat
-rw-r--r-- 1 root root 0 Jul  6 15:05 cgroup.subtree_control
-rw-r--r-- 1 root root 0 Jul  6 16:13 cgroup.threads
-rw-r--r-- 1 root root 0 Jul  6 16:13 cgroup.type
-rw-r--r-- 1 root root 0 Jul  6 16:13 cpu.max
```

```
-rw-r--r-- 1 root root 0 Jul  6 16:13 cpu.pressure
-rw-r--r-- 1 root root 0 Jul  6 16:13 cpuset.cpus
-r--r--r-- 1 root root 0 Jul  6 16:13 cpuset.cpus.effective
-rw-r--r-- 1 root root 0 Jul  6 16:13 cpuset.cpus.partition
-rw-r--r-- 1 root root 0 Jul  6 16:13 cpuset.mems
-r--r--r-- 1 root root 0 Jul  6 16:13 cpuset.mems.effective
-r--r--r-- 1 root root 0 Jul  6 10:58 cpu.stat
-rw-r--r-- 1 root root 0 Jul  6 16:13 cpu.weight
-rw-r--r-- 1 root root 0 Jul  6 16:13 cpu.weight.nice
-rw-r--r-- 1 root root 0 Jul  6 16:13 io.pressure
-r--r--r-- 1 root root 0 Jul  6 16:13 memory.current
-r--r--r-- 1 root root 0 Jul  6 16:13 memory.events
-r--r--r-- 1 root root 0 Jul  6 16:13 memory.events.local
-rw-r--r-- 1 root root 0 Jul  6 10:58 memory.high
-rw-r--r-- 1 root root 0 Jul  6 10:58 memory.low
-rw-r--r-- 1 root root 0 Jul  6 10:58 memory.max
-rw-r--r-- 1 root root 0 Jul  6 10:58 memory.min
-r--r--r-- 1 root root 0 Jul  6 16:13 memory.numa_stat
-rw-r--r-- 1 root root 0 Jul  6 10:58 memory.oom.group
-rw-r--r-- 1 root root 0 Jul  6 16:13 memory.pressure
-r--r--r-- 1 root root 0 Jul  6 16:13 memory.stat
-r--r--r-- 1 root root 0 Jul  6 16:13 memory.swap.current
-r--r--r-- 1 root root 0 Jul  6 16:13 memory.swap.events
-rw-r--r-- 1 root root 0 Jul  6 16:13 memory.swap.high
-rw-r--r-- 1 root root 0 Jul  6 10:58 memory.swap.max
-r--r--r-- 1 root root 0 Jul  6 16:13 pids.current
-r--r--r-- 1 root root 0 Jul  6 16:13 pids.events
-rw-r--r-- 1 root root 0 Jul  6 10:58 pids.max
drwxr-xr-x 8 root root 0 Jul  6 15:22 user-1000.slice
drwxr-xr-x 5 root root 0 Jul  6 11:41 user-113.slice
```

et les processus d'un utilisateur spécifique dans un slice dénommé **user-UID.slice** :

```
root@debian11:~# ls -l /sys/fs/cgroup/user.slice/user-1000.slice
```

```
total 0
-r--r--r-- 1 root    root    0 Jul  6 16:14 cgroup.controllers
-r--r--r-- 1 root    root    0 Jul  6 11:30 cgroup.events
-rw-r--r-- 1 root    root    0 Jul  6 16:14 cgroup.freeze
-rw-r--r-- 1 root    root    0 Jul  6 16:14 cgroup.max.depth
-rw-r--r-- 1 root    root    0 Jul  6 16:14 cgroup.max.descendants
-rw-r--r-- 1 root    root    0 Jul  6 16:14 cgroup.procs
-r--r--r-- 1 root    root    0 Jul  6 16:14 cgroup.stat
-rw-r--r-- 1 root    root    0 Jul  6 15:05 cgroup.subtree_control
-rw-r--r-- 1 root    root    0 Jul  6 16:14 cgroup.threads
-rw-r--r-- 1 root    root    0 Jul  6 16:14 cgroup.type
-rw-r--r-- 1 root    root    0 Jul  6 16:14 cpu.pressure
-r--r--r-- 1 root    root    0 Jul  6 11:30 cpu.stat
-rw-r--r-- 1 root    root    0 Jul  6 16:14 io.pressure
-r--r--r-- 1 root    root    0 Jul  6 16:14 memory.current
-r--r--r-- 1 root    root    0 Jul  6 16:14 memory.events
-r--r--r-- 1 root    root    0 Jul  6 16:14 memory.events.local
-rw-r--r-- 1 root    root    0 Jul  6 11:30 memory.high
-rw-r--r-- 1 root    root    0 Jul  6 11:30 memory.low
-rw-r--r-- 1 root    root    0 Jul  6 11:30 memory.max
-rw-r--r-- 1 root    root    0 Jul  6 11:30 memory.min
-r--r--r-- 1 root    root    0 Jul  6 16:14 memory.numa_stat
-rw-r--r-- 1 root    root    0 Jul  6 11:30 memory.oom.group
-rw-r--r-- 1 root    root    0 Jul  6 16:14 memory.pressure
-r--r--r-- 1 root    root    0 Jul  6 16:14 memory.stat
-r--r--r-- 1 root    root    0 Jul  6 16:14 memory.swap.current
-r--r--r-- 1 root    root    0 Jul  6 16:14 memory.swap.events
-rw-r--r-- 1 root    root    0 Jul  6 16:14 memory.swap.high
-rw-r--r-- 1 root    root    0 Jul  6 11:30 memory.swap.max
-r--r--r-- 1 root    root    0 Jul  6 16:14 pids.current
-r--r--r-- 1 root    root    0 Jul  6 16:14 pids.events
-rw-r--r-- 1 root    root    0 Jul  6 11:30 pids.max
drwxr-xr-x 2 root    root    0 Jul  6 14:56 session-13.scope
drwxr-xr-x 2 root    root    0 Jul  6 15:22 session-15.scope
```

```
drwxr-xr-x 2 root    root    0 Jul  6 11:30 session-4.scope
drwxr-xr-x 2 root    root    0 Jul  6 12:12 session-6.scope
drwxr-xr-x 4 trainee trainee 0 Jul  6 11:30 user@1000.service
drwxr-xr-x 2 root    root    0 Jul  6 11:41 user-runtime-dir@1000.service
```

De ce fait, il est possible d'utiliser systemd pour la mise en place des limitations des ressources en utilisant la commande **systemd set-property** :

CPU

```
root@debian11:~# systemctl set-property user-1000.slice CPUQuota=40%
root@debian11:~# cat /sys/fs/cgroup/user.slice/user-1000.slice/cpu.max
40000 100000
```

Mémoire

```
root@debian11:~# systemctl set-property user-1000.slice MemoryMax=1G
root@debian11:~# cat /sys/fs/cgroup/user.slice/user-1000.slice/memory.max
1073741824
```

Important - Notez que l'utilisation de **MemoryMax** met en place un **hard limit**. Il est aussi possible de mettre en place un **soft limit** en utilisant **MemoryHigh**.

Présentation de Linux Containers

LAB #3 - Travailler avec LXC

3.1 - Installation

Les outils indispensables à l'utilisation des Linux Containers sous Debian sont inclus dans le paquet **lxc** :

```
root@debian11:~# apt install lxc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libopengl0 linux-headers-5.10.0-15-amd64 linux-headers-5.10.0-15-common
Use 'apt autoremove' to remove them.
The following additional packages will be installed:
  arch-test bridge-utils busybox-static cloud-image-utils debootstrap distro-info
  fakechroot genisoimage libaio1 libdistro-info-perl libfakechroot liblxc1
  libpam-cgfs lxc-templates lxcfs mmdebstrap qemu-utils rsync uidmap uuid-runtime
Suggested packages:
  ubuntu-archive-keyring squid-deb-proxy-client shunit2 wodim cdrkit-doc btrfs-progs
  lvm2 python3-lxc qemu-user-static apt-transport-tor binfmt-support perl-doc proot
  qemu-user squashfs-tools-ng qemu-block-extra
The following packages will be REMOVED:
  busybox
The following NEW packages will be installed:
  arch-test bridge-utils busybox-static cloud-image-utils debootstrap distro-info
  fakechroot genisoimage libaio1 libdistro-info-perl libfakechroot liblxc1
  libpam-cgfs lxc lxc-templates lxcfs mmdebstrap qemu-utils rsync uidmap
  uuid-runtime
0 upgraded, 21 newly installed, 1 to remove and 5 not upgraded.
Need to get 6,127 kB of archives.
After this operation, 33.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

L'installation de ce paquet va créer les répertoires **/usr/share/lxc/config** contenant les fichiers de configurations des gabarits ainsi que le répertoire **/usr/share/lxc/templates** contenant fichiers de gabarits pour la création des conteneurs :

```
root@debian11:~# ls /usr/share/lxc
config  hooks  lxc.functions  lxc-patch.py  selinux  templates

root@debian11:~# ls /usr/share/lxc/config
alpine.common.conf     gentoo.moresecure.conf  slackware.userns.conf
alpine.userns.conf     gentoo.userns.conf      sparclinux.common.conf
archlinux.common.conf  nesting.conf            sparclinux.userns.conf
archlinux.userns.conf  oci.common.conf         ubuntu-cloud.common.conf
centos.common.conf     opensuse.common.conf    ubuntu-cloud.lucid.conf
centos.userns.conf     opensuse.userns.conf    ubuntu-cloud.userns.conf
common.conf            openwrt.common.conf     ubuntu.common.conf
common.conf.d          oracle.common.conf      ubuntu.lucid.conf
common.seccomp         oracle.userns.conf      ubuntu.userns.conf
debian.common.conf     plamo.common.conf       userns.conf
debian.userns.conf     plamo.userns.conf       voidlinux.common.conf
fedora.common.conf     sabayon.common.conf     voidlinux.userns.conf
fedora.userns.conf     sabayon.userns.conf
gentoo.common.conf     slackware.common.conf

root@debian11:~# ls /usr/share/lxc/templates
lxc-alpine     lxc-cirros         lxc-gentoo        lxc-oracle     lxc-sparclinux
lxc-altlinux   lxc-debian         lxc-local         lxc-plamo      lxc-sshd
lxc-archlinux  lxc-download       lxc-oci           lxc-pld        lxc-ubuntu
lxc-busybox    lxc-fedora         lxc-openmandriva  lxc-sabayon    lxc-ubuntu-cloud
lxc-centos     lxc-fedora-legacy  lxc-opensuse      lxc-slackware  lxc-voidlinux
```

3.2 - Création d'un Conteneur Simple

Créez un conteneur simple en utilisant la commande suivante :

```
root@debian11:~# lxc-create -n lxc-bb -t busybox
```

Important - Notez l'utilisation de l'option **-n** qui permet d'associer un nom au conteneur ainsi que l'option **-t** qui indique le gabarit à utiliser. Notez aussi que le gabarit est référencé par le nom du fichier dans le répertoire **/usr/share/lxc/templates** sans son préfix **lxc-**.

Le **backingstore** (*méthode de stockage*) utilisé par défaut est **dir** ce qui implique que le **rootfs** du conteneur se trouve sur disque dans le répertoire **/var/lib/lxc/** :

```
root@debian11:~# ls /var/lib/lxc/
lxc-bb

root@debian11:~# ls /var/lib/lxc/lxc-bb/
config  rootfs

root@debian11:~# ls /var/lib/lxc/lxc-bb/rootfs
bin  dev  etc  home  lib  lib64  mnt  proc  root  sbin  selinux  sys  tmp  usr  var
```

Il est à noter que LXC peut également utiliser des backingstores de type :

- ZFS
- Brtfs
- LVM
- Loop
- rbd (CephFS)

3.3 - Démarrage d'un Conteneur Simple

Pour démarrer le conteneur, il convient d'utiliser la commande **lxc-start** :

```
root@debian9:~# lxc-start --name lxc-bb
```

3.4 - S'attacher à un Conteneur Simple

Pour s'attacher au conteneur démarré, il convient d'utiliser la commande **lxc-attach** :

```
root@debian11:~# lxc-start --name lxc-bb

root@debian11:~# lxc-attach --name lxc-bb
lxc-attach: lxc-bb: terminal.c: lxc_terminal_create_native: 924 No space left on device - Failed to open terminal
multiplexer device


BusyBox v1.30.1 (Debian 1:1.30.1-6+b3) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ # which passwd
/bin/passwd
~ #
```

Important - Notez l'absence de la commande **passwd** dans le conteneur, ce qui explique l'erreur lors de la création de celui-ci.

Pour sortir du conteneur, il convient d'utiliser la commande **exit** ou bien la combinaison de touches **<Ctrl+d>** :

```
~ # [Ctrl+d]
~ # root@debian11:~#
```

Le fait de sortir du conteneur ne l'arrête pas pour autant, comme il peut être constaté par l'utilisation de la commande **lxc-ls** :

```
~ # root@debian11:~# [Enter]

root@debian11:~# lxc-ls --running
lxc-bb

root@debian11:~# lxc-ls -f --running
NAME   STATE   AUTOSTART GROUPS IPV4      IPV6 UNPRIVILEGED
lxc-bb RUNNING 0         -      10.0.3.48 -    false          -       -    -
```

3.5 - Commandes LXC de Base

La Commande lxc-console

Pour lancer une console attachée à un TTY dans le conteneur, il convient d'utiliser la commande **lxc-console** :

```
root@debian11:~# lxc-console --name lxc-bb

Connected to tty 1
Type <Ctrl+a q> to exit the console, <Ctrl+a Ctrl+a> to enter Ctrl+a itself

lxc-bb login:
```

Pour sortir de la console, il faut utiliser la combinaison de touches **<Ctrl+a> <q>** :

```
lxc-bb login: [Ctrl+a] [q] root@debian11:~#
```

La Commande lxc-stop

Pour arrêter le conteneur, utilisez la commande **lxc-stop** :

```
root@debian11:~# lxc-ls --running
lxc-bb

root@debian11:~# lxc-stop --name lxc-bb

root@debian11:~# lxc-ls --running

root@debian11:~#
```

La Commande lxc-execute

La commande **lxc-execute** démarre un conteneur (qui doit être créé mais arrêté), exécute la commande passée en argument grâce aux caractères **–** puis arrête le conteneur :

```
root@debian11:~# lxc-execute -n lxc-bb -- uname -a
Linux lxc-bb 5.10.0-24-amd64 #1 SMP Debian 5.10.179-5 (2023-08-08) x86_64 GNU/Linux

root@debian11:~# lxc-ls --running

root@debian11:~#
```

La Commande lxc-info

Cette commande donne des informations sur un conteneur :

```
root@debian11:~# lxc-info -n lxc-bb
Name:           lxc-bb
State:          STOPPED
```

La Commande lxc-freeze

La commande **lxc-freeze** met en pause tous les processus du conteneur :

```
root@debian11:~# lxc-start -n lxc-bb

root@debian11:~# lxc-ls --running
lxc-bb

root@debian11:~# lxc-info -n lxc-bb
Name:           lxc-bb
State:          RUNNING
PID:            28581
IP:             10.0.3.65
Link:           vethcJlTVk
 TX bytes:      1.22 KiB
 RX bytes:      3.88 KiB
 Total bytes:   5.10 KiB

root@debian11:~# lxc-freeze -n lxc-bb

root@debian11:~# lxc-info -n lxc-bb
Name:           lxc-bb
State:          FROZEN
PID:            28581
IP:             10.0.3.65
Link:           vethcJlTVk
 TX bytes:      1.22 KiB
 RX bytes:      4.06 KiB
 Total bytes:   5.28 KiB

root@debian11:~#
```

La Commande lxc-unfreeze

La commande **lxc-unfreeze** annule l'effet d'une commande **lxc-freeze** précédente :

```
root@debian11:~# lxc-unfreeze -n lxc-bb

root@debian11:~# lxc-info -n lxc-bb
Name:           lxc-bb
State:          RUNNING
PID:            28581
IP:             10.0.3.65
Link:           vethcJlTVk
 TX bytes:      1.22 KiB
 RX bytes:      4.47 KiB
 Total bytes:   5.69 KiB
```

Autres Commandes

Les autres commandes dont il faut avoir une connaissance sont :

Commande	Description
lxc-destroy	Permet de détruire complètement un conteneur
lxc-autostart	Permet de rebooter, tuer ou arrêter les conteneurs dont le drapeau **lxc.start.auto** est fixé dans le fichier **/var/lib/<nom_conteneur>/config**
lxc-cgroup	Permet de manipuler à chaud les CGroups pour un conteneur donné
lxc-device	Permet de rajouter à chaud les devices à un conteneur
lxc-usernsexec	Permet d'exécuter des commandes en tant que root dans un conteneur non-privilégié
lxc-wait	Permet d'attendre à ce qu'un conteneur ait atteint un certain état avant de continuer

3.6 - Création d'un Conteneur Éphémère

Par défaut les conteneurs LXC sont permanents. Il est possible de créer un conteneur éphémère, c'est-à-dire un conteneur où toutes les données sont détruites à l'arrêt de celui-ci, en utilisant la commande **lxc-copy** ainsi que l'option de cette commande **–epheremal** ou **-e**.

La Commande lxc-copy

Notez que le conteneur d'origine doit être arrêté lors de l'utilisation de la commande **lxc-copy** :

```
root@debian11:~# lxc-ls -f --running
NAME   STATE   AUTOSTART GROUPS IPV4      IPV6 UNPRIVILEGED
lxc-bb RUNNING 0         -      10.0.3.65 -    false
root@debian11:~# lxc-copy -e -N lxc-bb-eph -n lxc-bb

root@debian11:~# lxc-ls -f --running

root@debian11:~#
```

Arrêtez donc le conteneur **lxc-bb** puis créez la copie :

```
root@debian11:~# lxc-stop -n lxc-bb

root@debian11:~# lxc-ls -f --running

root@debian11:~# lxc-copy -e -N lxc-bb-eph -n lxc-bb
Created lxc-bb-eph as clone of lxc-bb

root@debian11:~# lxc-ls -f --running
NAME       STATE   AUTOSTART GROUPS IPV4      IPV6 UNPRIVILEGED
lxc-bb-eph RUNNING 0         -      10.0.3.21 -    false
```

Attachez-vous au conteneur **lxc-bb-eph** :

```
root@debian11:~# lxc-ls -f --running
```

```
NAME       STATE   AUTOSTART GROUPS IPV4      IPV6 UNPRIVILEGED
lxc-bb-eph RUNNING 0         -      10.0.3.21 -    false
root@debian11:~# lxc-attach lxc-bb-eph
lxc-attach: lxc-bb-eph: terminal.c: lxc_terminal_create_native: 924 No space left on device - Failed to open
terminal multiplexer device


BusyBox v1.30.1 (Debian 1:1.30.1-6+b3) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ #
```

Créez un fichier de contrôle appelé **testdata** :

```
~ # ls -l
total 0

~ # pwd
/root

~ # echo "test" > testdata

~ # ls -l
total 4
-rw-r--r--    1 root     root             5 Aug 20 09:10 testdata

~ #
```

Déconnectez-vous du conteneur puis attachez-vous de nouveau :

```
~ # exit

root@debian11:~# lxc-attach -n lxc-bb-eph
lxc-attach: lxc-bb-eph: terminal.c: lxc_terminal_create_native: 924 No space left on device - Failed to open
```

```
terminal multiplexer device


BusyBox v1.30.1 (Debian 1:1.30.1-6+b3) built-in shell (ash)
Enter 'help' for a list of built-in commands.

~ # ls -l
total 4
-rw-r--r--    1 root     root             5 Aug 20 09:10 testdata

~ #
```

Important - Notez que le fichier **testdata** est toujours présent.

Déconnectez-vous de nouveau et arrêtez le conteneur :

```
~ # exit

root@debian11:~# lxc-stop -n lxc-bb-eph

root@debian11:~# lxc-ls
lxc-bb

root@debian11:~# lxc-start -n lxc-bb-eph
lxc-start: lxc-bb-eph: tools/lxc_start.c: main: 268 No container config specified

root@debian11:~#
```

Important - Notez que le conteneur **lxc-bb-eph** a été détruit.

3.7 - Sauvegarde des Conteneurs

Un conteneur LXC peut être sauvegardé de trois façons différentes :

- utiliser la commande **tar** ou **cpio** pour créer un archive du répertoire **rootfs** et du fichier **config** associés au conteneur
- utiliser la commande **lxc-copy** sans l'option **-e**
- utiliser la commande **lxc-snapshot**

La Commande lxc-snapshot

Cette commande permet de gérer des instantanées des conteneurs. A noter que les conteneurs doivent être arrêtés avant de prendre une instantanée :

```
root@debian11:~# lxc-ls -f --running

root@debian11:~# lxc-snapshot -n lxc-bb

root@debian11:~#
```

Les snapshots sont stockés dans le sous-répertoire **snaps** du répertoire **/var/lib/lxc/<nom_conteneur>/**. Le premier s'appelle **snap0** :

```
root@debian11:~# ls -l /var/lib/lxc/lxc-bb
total 12
-rw-r-----  1 root root 1276 Aug 20 10:01 config
drwxr-xr-x 17 root root 4096 Aug 20 10:38 rootfs
drwxr-xr-x  3 root root 4096 Aug 20 12:35 snaps

root@debian11:~# ls -l /var/lib/lxc/lxc-bb/snaps/
total 4
drwxrwx--- 3 root root 4096 Aug 20 12:35 snap0

root@debian11:~# ls -l /var/lib/lxc/lxc-bb/snaps/snap0/
```

```
total 12
-rw-r-----  1 root root 1284 Aug 20 12:35 config
drwxr-xr-x 17 root root 4096 Aug 20 10:38 rootfs
-rw-r--r--  1 root root   19 Aug 20 12:35 ts
```

L'horodatage de la création du snapshot est stocké dans le fichier **ts** :

```
root@debian11:~# cat /var/lib/lxc/lxc-bb/snaps/snap0/ts
2023:08:20 12:35:35root@debian11:~#
```

En comparant la taille du **rootfs** du conteneur d'origine ainsi que de son snapshot, on peut constater que les deux sont identiques :

```
root@debian11:~# du -sh /var/lib/lxc/lxc-bb/rootfs/
2.1M    /var/lib/lxc/lxc-bb/rootfs/

root@debian11:~# du -sh /var/lib/lxc/lxc-bb/snaps/snap0/rootfs/
2.1M    /var/lib/lxc/lxc-bb/snaps/snap0/rootfs/
```

Pour restaurer un conteneur identique à l'original, il convient d'utiliser de nouveau la commande **lxc-snapshot** :

```
root@debian11:~# lxc-snapshot -r snap0 -n lxc-bb -N lxc-bb-snap0

root@debian11:~# lxc-ls
lxc-bb       lxc-bb-snap0

root@debian11:~# lxc-start -n lxc-bb-snap0

root@debian11:~# lxc-attach -n lxc-bb-snap0
lxc-attach: lxc-bb-snap0: terminal.c: lxc_terminal_create_native: 924 No space left on device - Failed to open
terminal multiplexer device


BusyBox v1.30.1 (Debian 1:1.30.1-6+b3) built-in shell (ash)
Enter 'help' for a list of built-in commands.
```

```
~ # exit

root@debian11:~#
```

Copyright © 2023 Hugh Norris.

From:
https://ittraining.team/ - **www.ittraining.team**

Permanent link:
https://ittraining.team/doku.php?id=elearning:workbooks:docker3:drf08

Last update: **2023/12/17 05:36**