Version : **2024.01**

Last update : 2024/12/17 13:46

DOE601 - Virtualisation by Isolation

Contents

Introduction to Virtualisation using Isolation

An isolator is a piece of software used to isolate the execution of applications in **containers**, **contexts** or **execution zones**.

History

- **1979** - chroot - root change isolation,
- **2000** - BSD Jails - user-space isolation,
- **2004** - Solaris Containers - zone-based isolation**,** * 2005 **- OpenVZ - kernel partitioning** isolation under Linux,
- **2008** - LXC - LinuX Containers - isolation using **namespaces** and **CGroups** with **liblxc**,
- **2013** - Docker - isolation using **namespaces** and **CGroups** with **libcontainer**,
- **2014** - LXD - LinuX Container Daemon - isolation using **namespaces** and **CGroups** with **liblxc**.

Namespaces presentation

Namespaces allow processes to be grouped together in the same space and rights to resources to be allocated per space. This makes it possible to run several inits, each in a namespace, in order to recreate an environment for processes that need to be isolated.

CGroups presentation

LAB #1 - cgroups v1

1.1 - Preparation

Debian 11 uses cgroups v2 by default. To revert to using cgroups v1, edit the **/etc/boot/grub** file and add the **systemd.unified_cgroup_hierarchy=0** directive to the **GRUB_CMDLINE_LINUX_DEFAULT** line:

```
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
```

Then reboot your VM :

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

1.2 - Overview

Control Groups (*Control Groups*) also known as **CGroups**, are a way of controlling and limiting resources. Control groups allow resources to be

allocated, even dynamically while the system is running, such as processor time, system memory, network bandwidth, or a combination of these resources among user-defined groups of tasks (processes) running on a system.

Control groups are organised hierarchically, like processes. However, comparing the two shows that while processes are in a single tree structure all descending from the init process and inheriting the environment from their parents, control groups can be multiple giving rise to multiple trees or **hierarchies** which inherit certain attributes from their parent control groups.

These multiple and separate hierarchies are necessary because each hierarchy is attached to one or more **subsystem(s)** also called **Resource Controllers** or simply **Controllers**. The controllers available under Debian 11 are :

- **blkio** - used to set limits on I/O access to and from block devices,
- **cpu** - used to provide tasks in control groups with CPU access through the scheduler,
- **cpuacct** - used to produce automatic reports on CPU resources used by tasks in a control group,
- **cpuset** - used to assign individual CPUs on a multicore system and memory nodes to tasks in a control group,
- **devices** - used to allow or deny task access to devices in a control group,
- **freezer** - used to suspend or reactivate tasks in a control group,
- **memory** - used to set memory usage limits for tasks in a control group and to generate automatic reports on memory resources used by these tasks,
- **net_cls** - used to track network packets with a class identifier (*classid*) to allow the Linux traffic controller, **tc**, to identify packets originating from a particular control group task.
- **perf_event** - used to allow CGroups to be monitored with the perf tool,
- **hugetlb** - used to limit resources on large virtual memory pages.

Note that :

- each process in the system belongs to a cgroup and only to one cgroup at a time,
- all threads in a process belong to the same cgroup,
- when a process is created, it is placed in the same cgroup as its parent process,
- a process can be migrated from one cgroup to another cgroup. However, migrating a process has no impact on the cgroup membership of its child processes.

Start by installing the **cgroup-tools** package:

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

To view the hierarchies, use the **lssubsys** command:

```
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
```

Under Debian 11, **Systemd** organises processes into each CGroup. For example all processes started by the Apache server will be in the same CGroup, including CGI scripts. This means that resource management using hierarchies is coupled with Systemd's unit tree.

At the top of the Systemd unit tree is the root slice - **-.slice**, on which depends :

- the **system.slice** - the location of system services,
- the **user.slice** - the location of user sessions,
- the **machine.slice** - the location of virtual machines and containers.

Below the slices you can find :

- **scopes** - processes created by **fork**,
- **services** - processes created by a **Unit**.

Slices can be viewed with the following command:

```
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'.
```

The Systemd unit tree is as follows:

```
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]
```

Using Systemd, several resources can be limited:

- **CPUShares** - default 1024,
- **MemoryLimit** - limit expressed in MB or GB. No default value,
- **BlockIOWeight** - value between 10 and 1000. No default value,
- **StartupCPUShares** - like CPUShares but only applied during startup,
- **StartupBlockIOWeight** - like BlockIOWeight but only applied during startup,
- **CPUQuota** - used to limit CPU time, even when the system is not doing anything.

1.3 - Memory limiting

Start by creating the **hello-world.sh** script that will be used to generate a process to work with CGroups:

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

Make the script executable and test it :

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

Now create a CGroup in the **memory** subsystem called **helloworld** :

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

By default, this CGroup will inherit all available memory. To avoid this, now create a **40000000** byte limit for this 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 - Note that the 40,000,000 requested has become 39,997,440 which corresponds to an integer number of kernel memory pages of 4KB. (39 997 440 / 4096 = 9 765).

Now run the **helloworld.sh** script:

```
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
```

Note that there is no memory limit, which implies inheritance by default:

```
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
```

Insert the PID of our script in the **helloworld** CGroup :

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

Now note the inheritance of the memory limitation - **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
```

Then note the actual memory consumption :

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

Kill the **hello-world.sh** script:

```
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
```

Create a second, much more restrictive CGroup:

```
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
```

Relaunch the **hello-world.sh** script and insert it into the new CGroup:

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

root@debian11:~# hello world
[Entrée]

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

Wait for the next **hello world** output on the STDOut and then notice that the script stops:

```
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
```

Note the contents of the **/var/log/messages** file:

```
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 - The cgcreate command

This command is used to create a 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
```

However, there is no command to assign a memory limit:

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

1.5 - The cgexec Command

This command inserts the limit into the CGroup **and** runs the script in a single line:

```
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 - The cgdelete Command

Once the script has completed, this command deletes the 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 - The /etc/cgconfig.conf file

In order to make them persistent, the **/etc/cgconfig.conf** file needs to be edited:

```
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 - Note the creation of **two** limits, one of 40,000 bytes of memory and the other of **100 cpu.shares**. The latter is a value expressed over 1,024, where 1,024 represents 100% of CPU time. The limit set is therefore equivalent to 9.77% of CPU time.

Now create the two CGroups required:

```
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 - The cgconfigparser command

Apply the contents of the **/etc/cgconfig.conf** file using the **cgconfigparser** command:

```
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 - Preparation

To revert to using cgroups v2, edit the **/etc/boot/grub** file and change the **systemd.unified_cgroup_hierarchy=0** directive to **systemd.unified_cgroup_hierarchy=1** in the **GRUB_CMDLINE_LINUX_DEFAULT** line:

```
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
```

Then reboot your VM :

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

2.2 - Presentation

Unlike cgroup v1, cgroup v2 has only one tree or hierarchy and therefore only one mount point. All v2-compatible controllers that are not linked to a v1 hierarchy are automatically linked to the v2 hierarchy. An inactive controller in the v2 hierarchy can be linked to another hierarchy. Migration of a controller from one hierarchy to another is only possible if the controller is deactivated and no longer referenced in the original hierarchy.

To check that cgroups v2 is being used, the mount point should be viewed:

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

and view the contents of this mount point:

```
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
```

In version 2 of cgroups, some names have changed from those used in version 1:

Version 1	Version 2
CPUShares	CPUWeight
StartupCPUShares	StartupCPUWeight
MemoryLimit	MemoryMax

Start by creating the child cgroup **pids** in the root cgroup:

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

Place the PID of the current terminal in the **cgroup.procs** file of the child cgroup:

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

Now check the contents of the cgroup.procs file as well as the number of PIDs in the **pids** cgroup:

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

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

Important - Note that the cgroup.procs file contains **two** PIDs. The first is from the Shell while the second is from the cat command.

Now inject the value of **5** into the **pids.max** file of the cgroup **pids** :

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

Run the following command to create 6 pids in the 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 - Note that when attempting to create the 6th process, an error is returned. The system then tries 4 more times and finally gives up with the error message **-bash: fork: Resource temporarily unavailable**.

Lastly, try deleting the **pids** cgroup:

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

Important - Note that it is not possible to remove a cgroup as long as it contains a process.

Move the current terminal process into the root cgroup:

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

It is now possible to delete the **pids** cgroup:

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

2.3 - Limiting CPU Resources

There are two ways to limit CPU resources:

- **CPU bandwidth**,
  - a limiting system based on a percentage of CPU for one or more processes,
- **CPU weight**,
  - a limiting system based on the prioritisassion of one or more processes relative to other processes.

In the following example, you are going to set up a **CPU bandwidth** type limit.

Start by creating a service called **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
```

Start and enable the 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.
```

Use the **ps** command to see the percentage of CPU used by this service :

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

```
    997 sha1sum         00:01:33  100
```

Now create another service called **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
```

Start and enable the 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.
```

Use the **ps** command to see the percentage of CPU used by this service :

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

Now check for the presence of the **cpuset** and **cpu** controllers in the root cgroup tree, which is mounted at **/sys/fs/cgroup/** :

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

Now enable the two controllers **cpuset** and **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
```

Create the **child** cgroup called **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
```

Enable the **cpuset** and **cpu** controllers for the **FooBar** cgroup:

```
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** - Note that it is not possible to enable controllers for a child cgroup if those same controllers are not already enabled for the parent cgroup. Also note that in the **FooBar** cgroup, the **memory** and **pids** controllers are **not** enabled.

Now create the **/sys/fs/cgroup/FooBar/tasks** directory:

```
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** - The **/sys/fs/cgroup/FooBar/tasks** directory defines a *child* group of the FooBar cgroup that only affects the **cpuset** and **cpu** controllers.

So that the two processes from the **foo** and **bar** services compete on the same CPU, inject the value of **1** into the **/sys/fs/cgroup/FooBar/tasks/cpuset.cpus** file:

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

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

Set up CPU resource limit with the following command:

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

> **Important** - In the above command, the first number is a quota in microseconds for which processes in the cgroup can run in a given **period** of time. The second number, also in microseconds, is the **period**. In other words, processes in the cgroup will be limited to running 200,000 / 1,000,000 = 0.2 seconds during each second.

Now add the **foo** and **bar** service processes to the **FooBar** cgroup:

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

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

Check that the previous command has been taken into account by the system:

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

Lastly, use the **top** command to see that CPU consumption and limited to 20% on all processes in the **FooBar** cgroup **and** that this 20% is divided equally between the two **foo** and **bar** processes :

```
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 - The systemctl set-property Command

As already seen, systemd organizes processes into **slices**, for example users are grouped into **/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
```

and the processes of a specific user in a slice named **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
```

Because of this, it is possible to use systemd to set resource limits using the **systemd set-property** command:

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
```

Memory

```
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** - Note that using **MemoryMax** sets up a **hard limit**. It is also possible to set up a **soft limit** by using **MemoryHigh**.

Introduction to Linux Containers

LAB #3 - Working with LXC

3.1 - Installation

The essential tools for using Linux Containers under Debian are included in the **lxc** package:

```
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
```

Installing this package will create the **/usr/share/lxc/config** directory containing the template configuration files and the **/usr/share/lxc/templates** directory containing template files for creating containers :

```
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 - Creating a Simple Container

Create a simple container using the following command:

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

> **Important** - Note the use of the **-n** option which allows a name to be associated with the container as well as the **-t** option which indicates the template to be used. Note also that the template is referenced by the name of the file in the **/usr/share/lxc/templates** directory without its **lxc-** prefix.

The **backingstore** (*storage method*) used by default is **dir** which implies that the **rootfs** of the container is located on disk in the **/var/lib/lxc/** directory :

```
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
```

Note that LXC can also use backingstores of the following types:

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

3.3 - Starting a Simple Container

To start the container, the **lxc-start** command should be used:

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

3.4 - Attaching to a Simple Container

To attach to the started container, the **lxc-attach** command should be used:

```
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 - Note the absence of the **passwd** command in the container, which explains the error when creating it.

To exit the container, use the **exit** command or the **<Ctrl+d>** key combination:

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

Exiting the container does not stop it, as can be seen by using the **lxc-ls** command:

```
~ # 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 - Basic LXC commands

The lxc-console Command

To launch a console attached to a TTY in the container, the **lxc-console** command should be used:

```
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:
```

To exit the console, use the key combination **<Ctrl+a> <q>** :

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

The lxc-stop Command

To stop the container, use the **lxc-stop** command :

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

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

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

root@debian11:~#
```

The lxc-execute command

The **lxc-execute** command starts a container (which must be created but stopped), executes the command passed as an argument using the **–** characters and then stops the container :

```
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:~#
```

The lxc-info command

This command gives information about a container :

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

The lxc-freeze Command

The **lxc-freeze** command pauses all processes in the container :

```
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:~#
```

The lxc-unfreeze Command

The **lxc-unfreeze** command cancels the effect of a previous **lxc-freeze** command :

```
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
```

Other commands

The other commands you need to know about are :

Command	Description
lxc-destroy	Allows you to completely destroy a container
lxc-autostart	Allows you to reboot, kill or stop containers whose flag **lxc.start.auto** is set in the **/var/lib/<container_name>/config** file
lxc-cgroup	Enables hot manipulation of CGroups for a given container
lxc-device	Enables hot addition of devices to a container
lxc-usernsexec	Allows you to execute commands as root in a non-privileged container
lxc-wait	Allows you to wait until a container has reached a certain state before continuing

3.6 - Creating an Ephemeral Container

By default, LXC containers are permanent. It is possible to create an ephemeral container, i.e. one in which all data is destroyed when the container is shut down, using the **lxc-copy** command and the **–epheremal** or **-e** option to this command.

The lxc-copy command

Note that the original container must be stopped when using the **lxc-copy** command:

```
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:~#
```

So stop the **lxc-bb** container and then create the copy:

```
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
```

Attach to the **lxc-bb-eph** container:

```
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.

~ #
```

Create a control file called **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

~ #
```

Disconnect from the container and then re-attach:

```
~ # 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 - Note that the **testdata** file is still present.

Log out again and shut down the container:

```
~ # 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 - Note that the **lxc-bb-eph** container has been destroyed.

3.7 - Saving Containers

An LXC container can be backed up in three different ways:

- use the **tar** or **cpio** command to create an archive of the **rootfs** directory and **config** file associated with the container,
- use the **lxc-copy** command without the **-e** option,
- use the **lxc-snapshot** command.

The lxc-snapshot command

This command is used to manage container snapshots. Note that containers must be stopped before taking a snapshot:

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

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

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

Snapshots are stored in the **snaps** subdirectory of the **/var/lib/lxc/<container_name>/** directory. The first one is called **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
```

The snapshot creation timestamp is stored in the **ts** file:

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

Comparing the size of the **rootfs** of the original container as well as its snapshot, we can see that both are identical:

```
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/
```

To restore a container identical to the original, use the **lxc-snapshot** command again:

```
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:~#
```
