Secure Containers for Developers

ToxCon
2019

Mathias Tausig

Who am I?

- MSc in Mathematics (University of Technology Vienna)
- Open Source user since 1997
- Professional experience as a Developer, Sysadmin, Security Officer, Computer retail
- 8 years in PKI business, Security Officer
- 4 years teaching IT-Security at FH Campus Wien
- Soon: Security Consultant as *SBA Research*

History

Any problem in computer science can be solved with another layer of indirection.
– David Wheeler[FDF05]

Virtualization

- In the 1960s, some OS (MIT CTSS) supported *Virtualizaion*: Multiple programs could be ran at the same time while transparentely (for the application) sharing the same hardware
- 1967 IBM's *CP-40/CMS* (System/360) allowed the parallel usage of multiple operating systems
  - Robustness (One crashing OS couldn't take down the other ones)
  - Timesharing
  - Support of legacy application on new systems
- 1972: *IBM VM/370* for System/370 used *Virtual Machines*: All hardware interfaces are virtualized
- Virtualization technology looses traction due to the rise of microprocessors and PCs (and the fall of their costs)
- 1999: VMWare issues the first virtualization software for x86 systems targeted towards a generic audience

[18]

Container I

- Since the 1980s, the *chroot* technology allows parts of the filesystem to separated from each other on *NIX systems
- Parallel to the rise of virtualization 2000, interest in a more lightweight *OS-Virtualization* grew
  - 2000: Virtuozzo (Linux, Windows)
    - 2005 OpenVZ
  - 2000: Jails (BSD)
  - 2001: Linux VServer
  - 2004: Zones (Solaris)
- Around the same time, generic interfaces in the Linux kernel were developed:
  - 1998: AppArmor
  - 2000: SELinux
  - 2002: Namespaces
  - 2007: CGroups

Container II

- 2008 saw the first release of LXC (Linux Containers), a userspace interface to create virtualized environments using these technologies
  - No kernel modification neccesary
- In 2013, the company *dotCloud Inc.* released the first version of its container software: *Docker*
  - Initially based on LXC
  - Switched to the *libcontainer* interface to use the kernel’s capabilities

Containers

VM vs. Container

While a virtual machine is always running a full OS on virtual hardware, a container is part of the current host system sharing its resources (especially the kernel).

VM vs. Container

A container ist more lightweight than a VM.

- Less storage space
- Less memory
- Much faster creation and startup

These performance advantages are offset by a worse isolation. Since all containers share the same kernel, an exploit on the kernel level can comprosie all containers on the host.

System- vs. application-container

- An application container is used to run a single process. If that process is stopped, the corresponding container is terminated as well.
- A system container is able to run multiple processes while keeping a persisten state over a long time.

Privileged containers

A privileged container is one running with root privileges on the host system.[1] An unprivileged container does not have those capabilities.[2]

[1] Default for Docker

[2] Default for LXC

Linux containment features

Namespaces

- Since kernel 3.12
- Used to isolate system ressources and processes
- Provides a certain ressource to a process in an abstracted fashion
- Associated to a process when they are started (`clone()` function)

[AJ17][Ros13]

Namespace types

Currently, the Linux kernel knows 6 types of namespaces:

- **pid**: Container may administer their own process hierarchy while having their own (logical) init process with PID 1
- **user**: Isolation of user- and group IDS (uid and gid) allows a process to run processes as “root” without granting it elevated privileges on the host system
- **net**: Provides separated network devices and configurations as well as routing tables
- **mnt**: Each container can have their own view of the filesystem hierarchy
- **ipc**: Allows the separation of methods for interprocess communication
- **uts**: Short for *Unix Timesharing*. Used to specify an individual hostname

cgroups

To limit negative consequences to the host system by a container, *Control Groups (cgroups)* may be used. They are built out of various *subsystems*, each of which limits a certain resource for a container.

- **blkio**: Limits access to block devices
- **cpu**, **cpuacct**, **cpusets**: Limits CPU access
- **devices**: Access to devices can be granted
- **freezer**: Allows to stop and wakeup tasks
- **hugetlb**, **memory**: Limits available RAM
- **net_cls**, **net_prio**: Used for network priorisation
- **perf_event**: Used for process monitoring

SELinux / AppArmor

The kernel security modules *SELinux* and *AppArmor* greatly extends the usual *Discretionary Access Control (DAC)* model of linux *Access Controll Policies* with a much more advanced and powerful *Mandatory Access Controll (MAC)* system.
The allows i.e to limit which files a certain process may access, or to cut off its network access.

Linux Containers

What is LXC?

> *LXC is a userspace interface for the Linux kernel containment features. Through a powerful API and simple tools, it lets Linux users easily create and manage system or application containers.*

– `https://linuxcontainers.org`

Frontends

LXC container can be created and managed using different tools:

- Direct usage of *liblxc* and *lxc-utils*
- Usage of a frontend
  - libvirt
  - ProxMox
  - LXD

LXD

> *LXD is a next generation system container manager. It offers a user experience similar to virtual machines but using Linux containers instead.*

– `https://linuxcontainers.org/lxd/introduction/`

LXD Architecture

LXD is based on a daemon (which in turn is based on *liblxc*) which provides a REST API.
This API is consumed by the command line tool *lxc*[3]

[3] No typo. The tool is really named like this.

Images

New containers are not installedm they get cloned from a base image, which is retrieved from a repository[4].

[4] local or online

Storage

LXC supports multiple storage backends

- Directory
- LVM
- Btrfs
- ZFS
- Ceph

Btrfs and ZFS support the very convenient data deduplication feature.

Use Cases Developer

- Isolated execution of applications
- Development environments with seperated dependencies
- Test environments

LXD Tutorial

Requirements

The following scenarios assume the following:

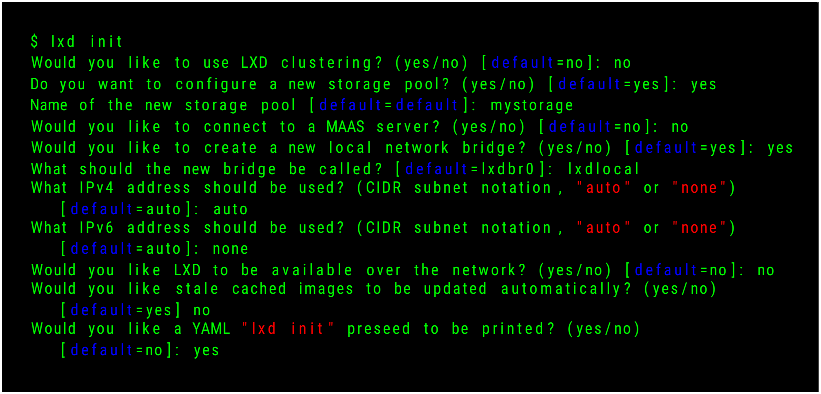
- Ubuntu 18.04 Bionic 64 bit is used
- Packages *lxd, lxdtool* are installedm
- User is part of the group *lxd*[5]

[5] Disclaimer: As with docker, this is equivalent to giving the user root privileges on the system. Take care.

Documentations

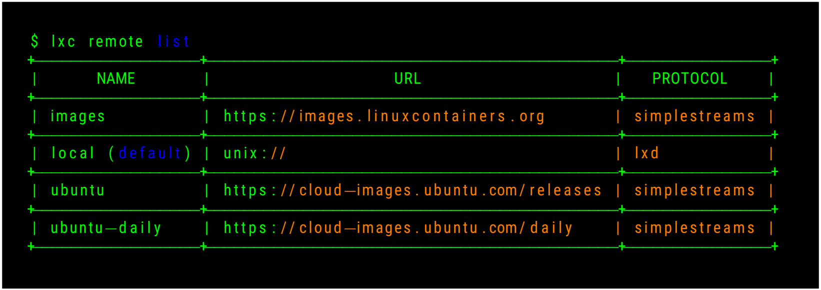
- Official documentation:
  `https://linuxcontainers.org/lxd/docs/master/`
- Blog of Stéphane Graber:
  `https://stgraber.org/category/lxd/`

Initialize



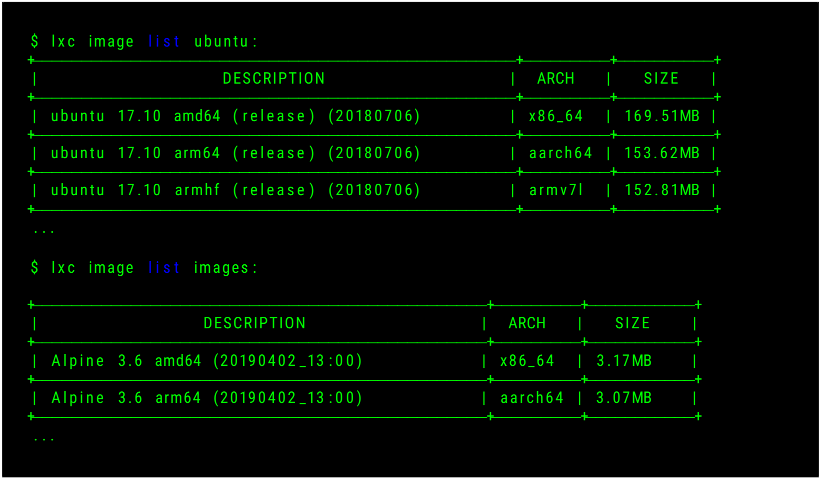
```
$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]: no
Do you want to configure a new storage pool? (yes/no) [default=yes]: yes
Name of the new storage pool [default=default]: mystorage
Would you like to connect to a MAAS server? (yes/no) [default=no]: no
Would you like to create a new local network bridge? (yes/no) [default=yes]: yes
What should the new bridge be called? [default=lxdbr0]: lxdlocal
What IPv4 address should be used? (CIDR subnet notation, "auto" or "none")
   [default=auto]: auto
What IPv6 address should be used? (CIDR subnet notation, "auto" or "none")
   [default=auto]: none
Would you like LXD to be available over the network? (yes/no) [default=no]: no
Would you like stale cached images to be updated automatically? (yes/no)
   [default=yes] no
Would you like a YAML "lxd init" preseed to be printed? (yes/no)
   [default=no]: yes
```

Images



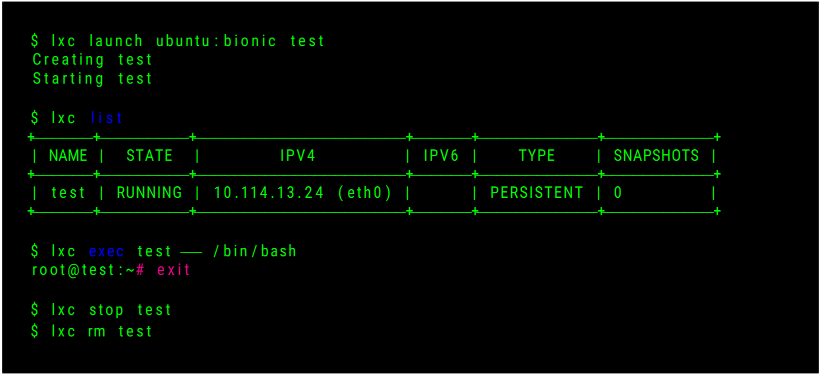
```
$ lxc remote list
+-----------------+------------------------------------------+---------------+
|      NAME       |                   URL                    |   PROTOCOL    |
+-----------------+------------------------------------------+---------------+
| images          | https://images.linuxcontainers.org       | simplestreams |
+-----------------+------------------------------------------+---------------+
| local (default) | unix://                                  | lxd           |
+-----------------+------------------------------------------+---------------+
| ubuntu          | https://cloud-images.ubuntu.com/releases | simplestreams |
+-----------------+------------------------------------------+---------------+
| ubuntu-daily    | https://cloud-images.ubuntu.com/daily    | simplestreams |
+-----------------+------------------------------------------+---------------+
```

Images



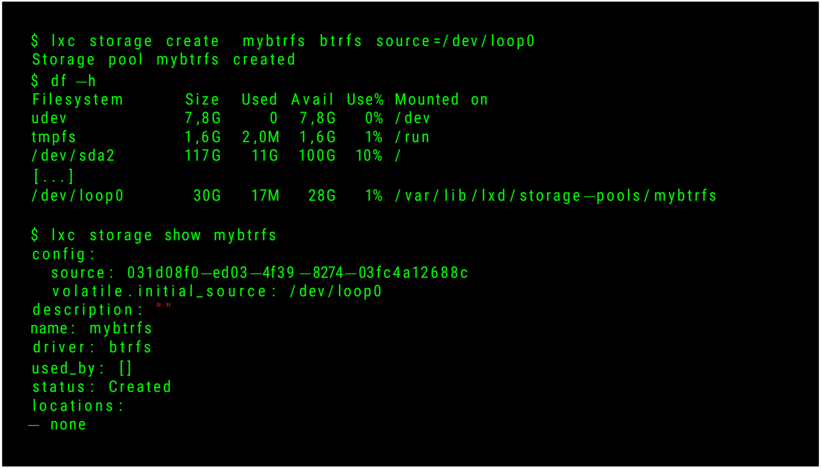
```
$ lxc image list ubuntu:
+------------------------------------------------+---------+----------+
|                  DESCRIPTION                   |  ARCH   |   SIZE   |
+------------------------------------------------+---------+----------+
| ubuntu 17.10 amd64 (release) (20180706)        | x86_64  | 169.51MB |
+------------------------------------------------+---------+----------+
| ubuntu 17.10 arm64 (release) (20180706)        | aarch64 | 153.62MB |
+------------------------------------------------+---------+----------+
| ubuntu 17.10 armhf (release) (20180706)        | armv7l  | 152.81MB |
+------------------------------------------------+---------+----------+
...

$ lxc image list images:

+---------------------------------------------+---------+----------+
|                 DESCRIPTION                 |  ARCH   |   SIZE   |
+---------------------------------------------+---------+----------+
| Alpine 3.6 amd64 (20190402_13:00)           | x86_64  | 3.17MB   |
+---------------------------------------------+---------+----------+
| Alpine 3.6 arm64 (20190402_13:00)           | aarch64 | 3.07MB   |
+---------------------------------------------+---------+----------+
...
```

Container lifecycle



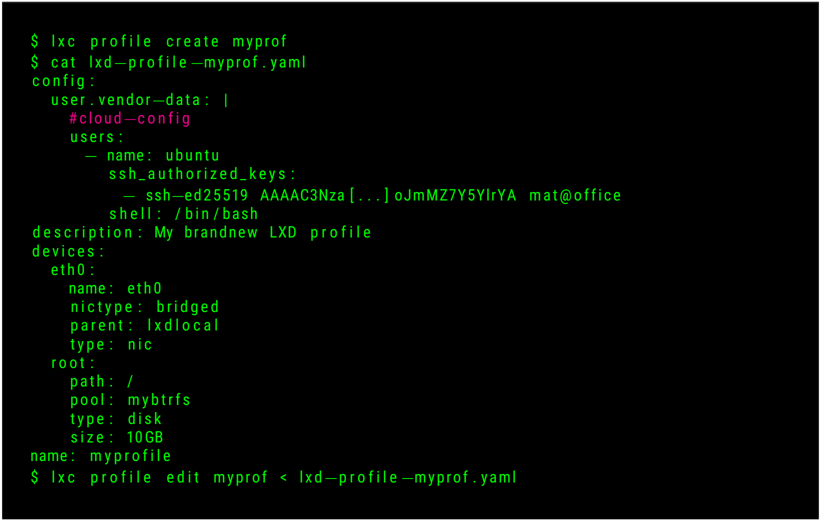
```
$ lxc launch ubuntu:bionic test
Creating test
Starting test

$ lxc list
+------+---------+---------------------+------+------------+-----------+
| NAME |  STATE  |        IPV4         | IPV6 |    TYPE    | SNAPSHOTS |
+------+---------+---------------------+------+------------+-----------+
| test | RUNNING | 10.114.13.24 (eth0) |      | PERSISTENT | 0         |
+------+---------+---------------------+------+------------+-----------+

$ lxc exec test -- /bin/bash
root@test:~# exit

$ lxc stop test
$ lxc rm test
```

BTRFS storage



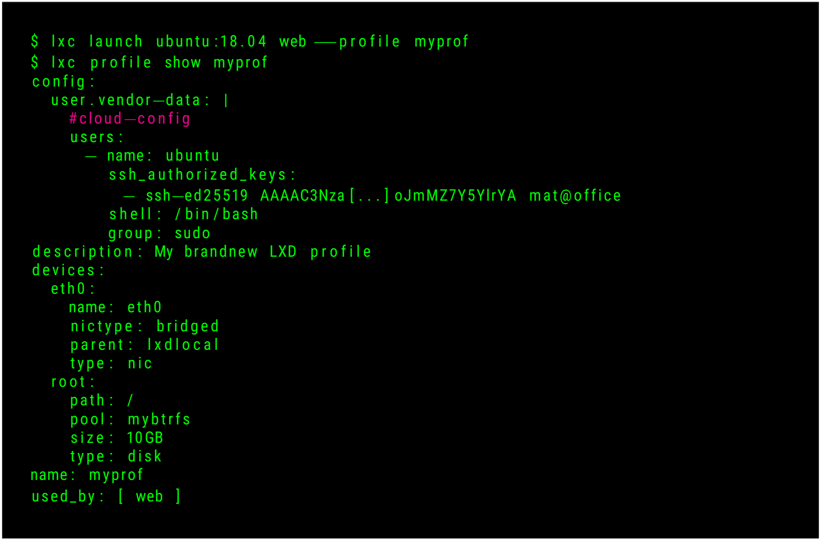
```
$ lxc storage create  mybtrfs btrfs source=/dev/loop0
Storage pool mybtrfs created
$ df -h
Filesystem       Size  Used  Avail  Use% Mounted on
udev             7,8G     0   7,8G    0% /dev
tmpfs            1,6G  2,0M   1,6G    1% /run
/dev/sda2        117G   11G   100G   10% /
[...]
/dev/loop0        30G   17M    28G    1% /var/lib/lxd/storage-pools/mybtrfs

$ lxc storage show mybtrfs
config:
  source: 031d08f0-ed03-4f39-8274-03fc4a12688c
  volatile.initial_source: /dev/loop0
description: ""
name: mybtrfs
driver: btrfs
used_by: []
status: Created
locations:
- none
```

Profile



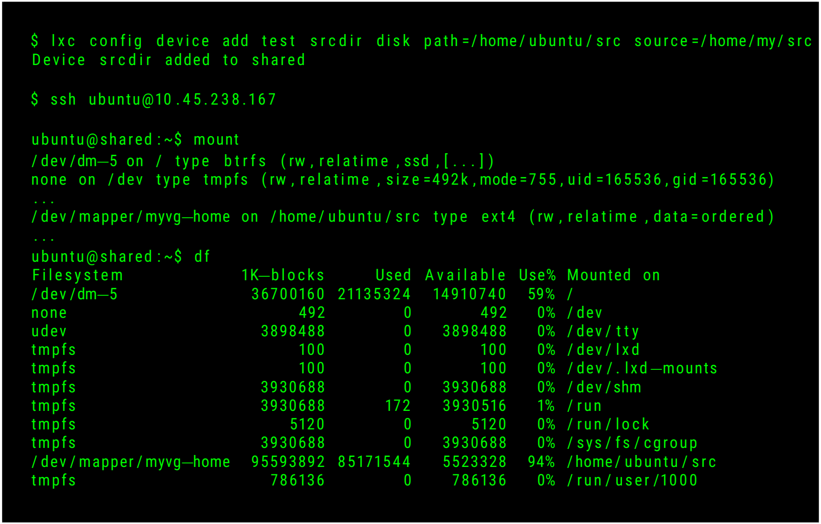
```
$ lxc profile create myprof
$ cat lxd-profile-myprof.yaml
config:
  user.vendor-data: |
    #cloud-config
    users:
      - name: ubuntu
        ssh_authorized_keys:
          - ssh-ed25519 AAAAC3Nza[...]oJmMZ7Y5YlrYA mat@office
        shell: /bin/bash
description: My brandnew LXD profile
devices:
  eth0:
    name: eth0
    nictype: bridged
    parent: lxdlocal
    type: nic
  root:
    path: /
    pool: mybtrfs
    type: disk
    size: 10GB
name: myprofile
$ lxc profile edit myprof < lxd-profile-myprof.yaml
```

Profile



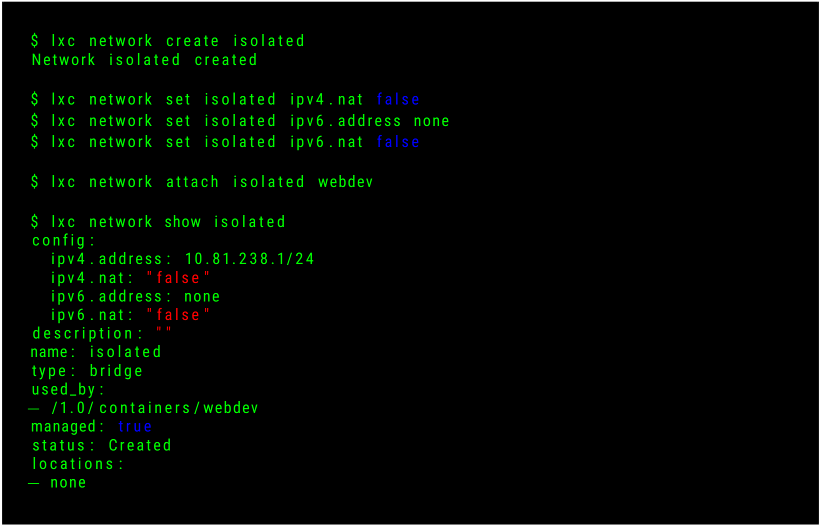
```
$ lxc launch ubuntu:18.04 web --profile myprof
$ lxc profile show myprof
config:
  user.vendor-data: |
    #cloud-config
    users:
      - name: ubuntu
        ssh_authorized_keys:
          - ssh-ed25519 AAAAC3Nza[...]oJmMZ7Y5YlrYA mat@office
        shell: /bin/bash
        group: sudo
description: My brandnew LXD profile
devices:
  eth0:
    name: eth0
    nictype: bridged
    parent: lxdlocal
    type: nic
  root:
    path: /
    pool: mybtrfs
    size: 10GB
    type: disk
name: myprof
used_by: [ web ]
```

Shared Disk



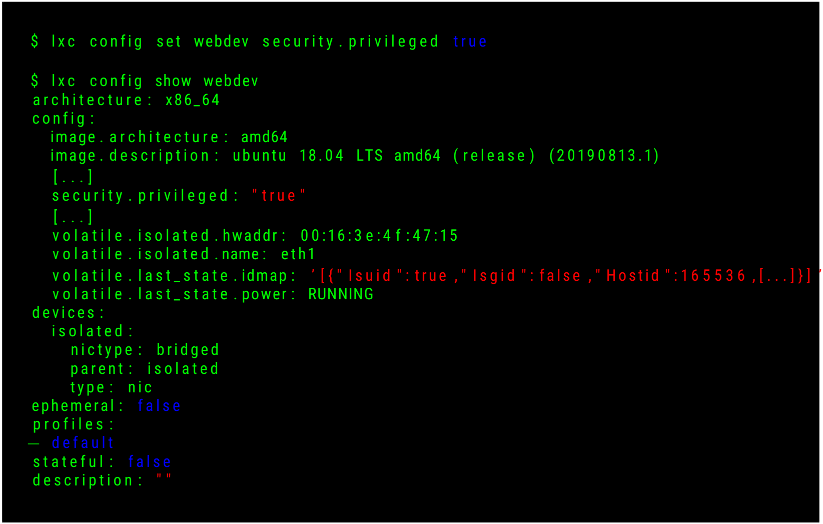
```
$ lxc config device add test srcdir disk path=/home/ubuntu/src source=/home/my/src
Device srcdir added to shared

$ ssh ubuntu@10.45.238.167

ubuntu@shared:~$ mount
/dev/dm-5 on / type btrfs (rw,relatime,ssd,[...])
none on /dev type tmpfs (rw,relatime,size=492k,mode=755,uid=165536,gid=165536)
...
/dev/mapper/myvg-home on /home/ubuntu/src type ext4 (rw,relatime,data=ordered)
...
ubuntu@shared:~$ df
Filesystem             1K-blocks     Used Available Use% Mounted on
/dev/dm-5               36700160 21135324  14910740  59% /
none                         492        0       492   0% /dev
udev                     3898488        0   3898488   0% /dev/tty
tmpfs                        100        0       100   0% /dev/lxd
tmpfs                        100        0       100   0% /dev/.lxd-mounts
tmpfs                    3930688        0   3930688   0% /dev/shm
tmpfs                    3930688      172   3930516   1% /run
tmpfs                       5120        0      5120   0% /run/lock
tmpfs                    3930688        0   3930688   0% /sys/fs/cgroup
/dev/mapper/myvg-home   95593892 85171544   5523328  94% /home/ubuntu/src
tmpfs                     786136        0    786136   0% /run/user/1000
```

Network



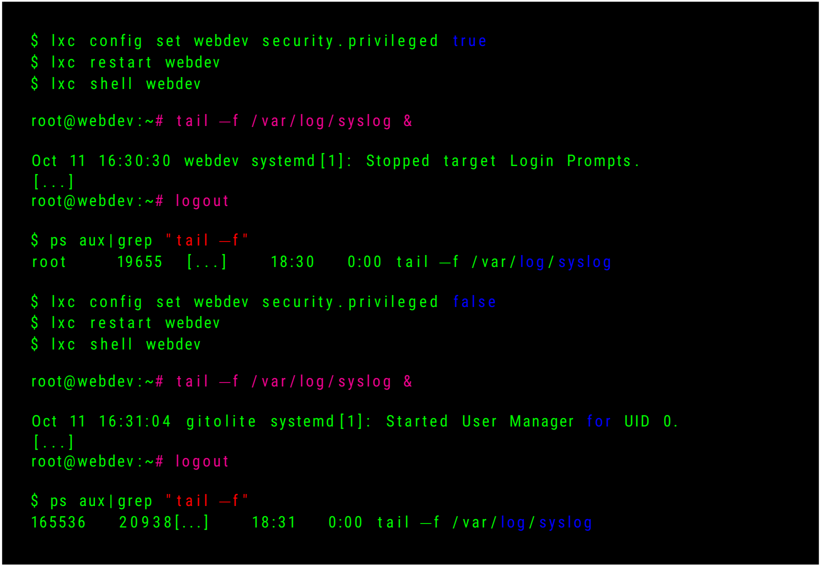
```
$ lxc network create isolated
Network isolated created

$ lxc network set isolated ipv4.nat false
$ lxc network set isolated ipv6.address none
$ lxc network set isolated ipv6.nat false

$ lxc network attach isolated webdev

$ lxc network show isolated
config:
  ipv4.address: 10.81.238.1/24
  ipv4.nat: "false"
  ipv6.address: none
  ipv6.nat: "false"
description: ""
name: isolated
type: bridge
used_by:
- /1.0/containers/webdev
managed: true
status: Created
locations:
- none
```

Privileged containers



```
$ lxc config set webdev security.privileged true

$ lxc config show webdev
architecture: x86_64
config:
  image.architecture: amd64
  image.description: ubuntu 18.04 LTS amd64 (release) (20190813.1)
  [...]
  security.privileged: "true"
  [...]
  volatile.isolated.hwaddr: 00:16:3e:4f:47:15
  volatile.isolated.name: eth1
  volatile.last_state.idmap: '[{"Isuid":true,"Isgid":false,"Hostid":165536,[...]}]'
  volatile.last_state.power: RUNNING
devices:
  isolated:
    nictype: bridged
    parent: isolated
    type: nic
ephemeral: false
profiles:
– default
stateful: false
description: ""
```

Privileged containers



```
$ lxc config set webdev security.privileged true
$ lxc restart webdev
$ lxc shell webdev

root@webdev:~# tail —f /var/log/syslog &

Oct 11 16:30:30 webdev systemd[1]: Stopped target Login Prompts.
[...]
root@webdev:~# logout

$ ps aux|grep "tail —f"
root      19655   [...]     18:30    0:00  tail —f /var/log/syslog

$ lxc config set webdev security.privileged false
$ lxc restart webdev
$ lxc shell webdev

root@webdev:~# tail —f /var/log/syslog &

Oct 11 16:31:04 gitolite systemd[1]: Started User Manager for UID 0.
[...]
root@webdev:~# logout

$ ps aux|grep "tail —f"
165536    20938[...]     18:31    0:00  tail —f /var/log/syslog
```

The End!

References I

[18] Hypervisor. In: Wikipedia. Page Version ID: 182551293. 7. Nov. 2018. URL: https://de.wikipedia.org/w/index.php?title=Hypervisor&oldid=182551293.

[AJ17] Manuel Atug und Daniel Jedecke. “Gut gepflegt - Linux Container sicher betreiben”. In: iX Kompakt Container und Virtualisierung (Herbst 2017 2017), S. 144–147.

[FDF05] R. Figueiredo, P.A. Dinda und J. Fortes. “Guest Editors’ Introduction: Resource Virtualization Renaissance”. In: Computer 38.5 (Mai 2005), S. 28–31. DOI: 10.1109/MC.2005.159. URL: https://www.computer.org/csdl/mags/co/2005/05/r5028.html.

[Ros13] Rami Rosen. “Resource Management: Linux Kernel Namespaces and Cgroups”. Mai 2013. URL: https://www.cs.ucsb.edu/~rich/class/cs293b-cloud/papers/lxc-namespace.pdf.