ACAT 2021
November 29, 2021 to December 3, 2021, Virtual and IBS Science Culture Center, Daejeon, South Korea

OpenTop, towards a lightweight scaleable containerization using the Linux Kernel

Ludwig Jaffé, Alexander Adler, Udo Kebschull
luja@openhardware.de, adler@cern.ch, kebschull@rz.uni-frankfurt.de
Infrastructure and Computer Systems in Data Processing (IRI), Goethe University Frankfurt, Germany

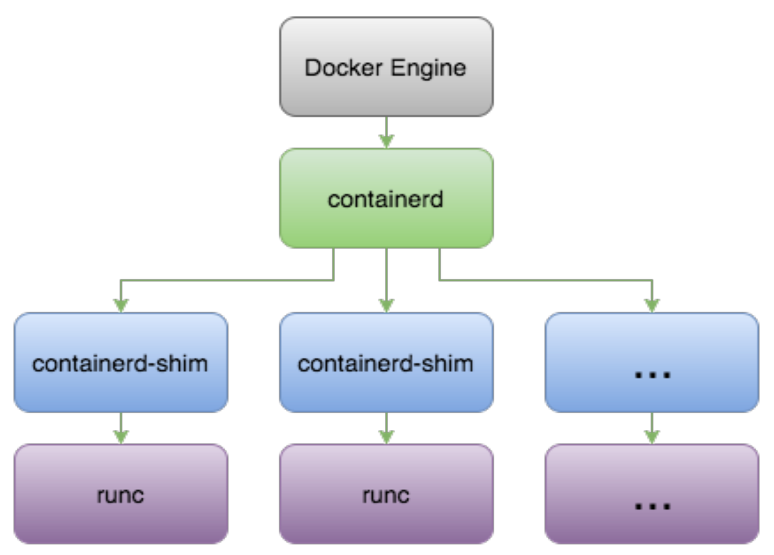
Abstract

Containerisation is an indispensible tool for sharing IT resources:
It is more light-weight than full virtualisation, but offers comparable isolation. We argue that for many use-cases which are typically approached with standard containerisation tools, **less than full isolation is sufficient**: Sometimes, only networking or only storage or both need to be different from their native state.
We present OpenTop, an ongoing effort towards composable and **configurable containerisation**. It was decided to focus primarily on configurable networking; later research will add the remaining aspects. A **domainspecific language** is proposed to describe the network **topology of "containers**", including their network devices, routes, firewalls/packetfilters etc. All the typical means of network administration (if applicable) should be available. Only elementary techniques are used,
e.g., **Linux namespaces, controlgroups**, iptables/nftables and virtual extensible LANs with veth pairs. For simplicity of use, these tools are applied **directly without involving any frameworks**, libraries (besides the libc) or init systems. This approach is deemed to be both more **portable** to different (embedded, also lfs) Linux distributions and more easily debugged/audited (both for developers and users of opentop). Also as isolation consumes computing time
OpenTop **minimizes overhead** caused by isolation to the minimum, as **only the specified isolation features** are used.

Motivation:

Well known "hipster" containerization tools for Linux are complex

If people think about containerization, most think about Docker, Kubernetes or LXC. But what are they made of ?
As there is no containerization (cf. BSD jails, Solaris zones) in the Linux Kernel, all container frameworks combine different isolation features (e.g. namespaces, cgroups, SELinux, AppArmor) of the Kernel to emulate containers. In the end these well known tools are very complex, especially Docker is an example of a bloated architecture as runc alone can run containers. Do they need to be that big?
In the end the Linux Kernel provides the isolation not the container tool. The container tools hide those means away from the user to present the concept of a container to the user.
OpenTop also uses those Kernel features but, based on recipes, it just does what is needed for the use-case specified by the user. There will be no demons in OpenTop (footprint, attack surface, reliability), just generated executeables to serve the use-case like create, exec, destroy. OpenTop does not need to provide full isolation, and allows the user to specify only the aspects of isolation needed for her use-case. So OpenTop gets closer to the metal. Like an Open Top shipping container OpenTop is more flexible in the isolation provided and the payload, so computing power is saved, which is important for High Performance Computing.



Docker architecture as an example for bloat [1]: BTW, runc alone can run containers

Findings: all Container tools for Linux build on Kernel isolation features:

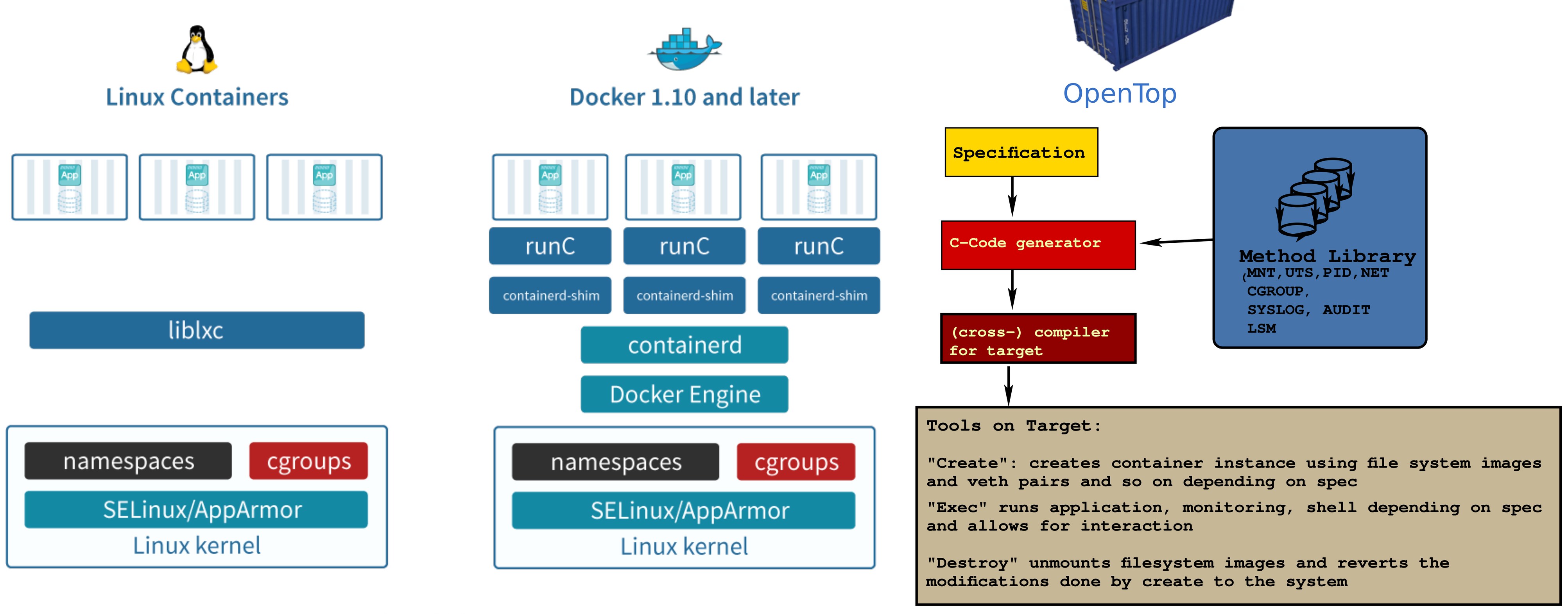
runc
used by docker,

liblxc
used by LXC.
Kernel provides:

namespaces
`MNT, UTS, IPC, PID, NET, USER, CGROUP, (SYSLOG, AUDIT)`

CGroups
CPU, memory, block I/O, network I/O, process limits
Linux Security Modules (LSM)
AppArmor, SELinux, Seccomp

Architecture of LXC and Docker containerization on Linux compared with the OpenTop approach



Linux Containers, LXC vs Docker [2] vs OpenTop

Both use namespaces, cgroups, LSM and the Kernel
we can use these tools too and do it using command line tools.

Example: Use shell to create the container aspect of network name spaces

```
ip link add br10 type bridge
ip addr add 10.10.10.1/255.255.255.0 dev br10
ip link set br10 up

iptables -t nat -A POSTROUTING -s 10.10.10.0/255.255.255.0 \
     -o eth0 -j MASQUERADE >/dev/null 2>&1

ip netns add thenamespace
ip link add dev veth0_42 type veth peer name veth1_42
ip link set veth0_42 up
ip link set veth0_42 master br10
ip link set veth1_42 netns thenamespace
ip netns exec thenamespace ip link set dev lo up
ip netns exec thenamespace ip link set veth1_42 address "00:80:41:00:00:01"
ip netns exec thenamespace ip addr add 10.10.10.23/255.255.255.0 dev veth1_42
ip netns exec thenamespace ip link set dev veth1_42 up
```

We want a new net
10.10.10.0/24
in the net name space
and run a server in it

```
#this is what the network looks like
ip netns exec thenamespace ip addr
ip addr
```

We now see that
the server in the name space
can be reached only by a client
in the same name space
so the new net is isolated

#now we run a server in the namespace thenamespace and try to connect it from outside and from inside

TERMINAL 1
```
ip netns exec thenamespace nc -l 2000
Hello, World!
```

#now we connect from outside without success

TERMINAL 2
```
telnet 10.10.10.23  2000
```

#now we enter the namespace and connect to the server sucessfully, and type the magic words.

TERMINAL 3
```
ip netns exec thenamespace telnet 10.10.10.23 2000

Trying 10.10.10.23...
Connected to 10.10.10.23.
Escape character is '^]'.
Hello, World!
```

If the code generator would produce linux shelll (c source would fill a book) this shell upto "#now we run a server" would be part of the generated code. IP addresses and the like would be hard coded into the code as it is just for the use-case.

Outlook: OpenTop will be a lightweight scaleable tool for Linux containerization

No Daemons, but simple customized executables created from a specification file exactly to the need.
Create
Create a container from a specification

Exec
Run applications and or shell in container

Destroy
Destroy container and restore previous state

quoted material:
[1] https://raw.githubusercontent.com/ippontech/blog-usa/master/images/2016/06/dockerrunc-3.png
[2] https://belajarlinux.id/pengenalan-lxc-atau-lxd/