Linux Containers and Security

Plan

- Introduction
  - Virtual Machines
  - Containers
  - Container examples / projects
- Cloud native ecosystem
- Linux containers modern application deployment
- Container images
- Linux file permissions and capabilities
- Linux system calls
- Container isolation
- Containers in practice
- Containers advanced security protections
  - Seccomp
  - Linux security modules
  - Linux eBPF
- Cloud deployment
  - Orchestrators
- Conclusion

Information

Djalal Harouni Medium - Github - Twitter Website: https://djalal.opendz.org/

Presentation at: https://github.com/OpenDZ/courses/blob/master/tixxdz/linuxdz/linux-containers.md

Email for corrections here: tixxdz+linuxdz@gmail.com - (sorry if I do not reply to all emails.)

Date: 06-04-2021 Last Modified: 10 Apr 2021

Note to convert to pdf:

```
pandoc --variable urlcolor=blue \
    linuxdz/linux-containers.md \
    -o linuxdz/linux-containers.pdf
```

1. Introduction

Usually you develop your application on your own workstation that has:

- Your system configurations
- Your own OS or Linux distribution
- Specific libraries and development environment

The problems:

- Standardized environment for business production ?
- Collaboration with other team members ?
- Move the application from one cloud / server hosting provider to another one without redoing all the work ?
- Perform CI/CD and pass quality assurance tests

The answer: containers.

The container is like a package or an image of your “backend” application(s) with all the dependencies included. You can move it through production environments easily.

The point of Linux containers is to develop and ship faster.

1.1 Virtual Machines

Virtualization uses a hypervisor to emulate hardware. Multiple operating systems can run side by side, this is heavy compared to containers.

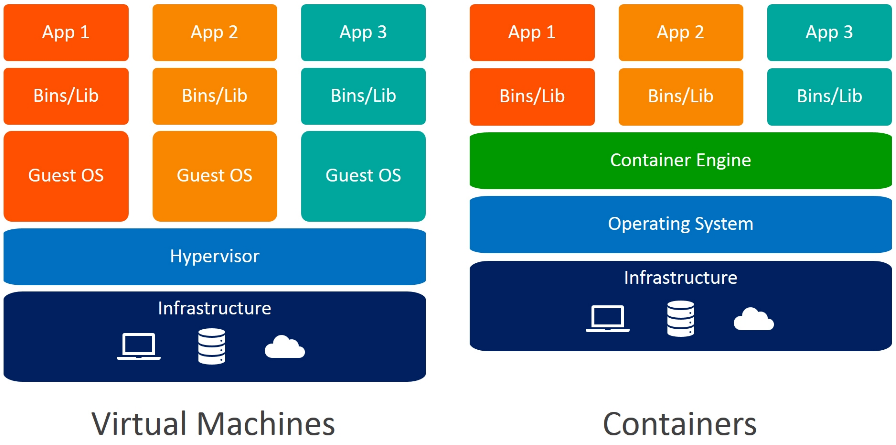

Figure 1: Virtual machines vs Containers

Image source: https://www.weave.works/blog/a-practical-guide-to-choosing-between-docker-containers-and-vms

1.2 Containers

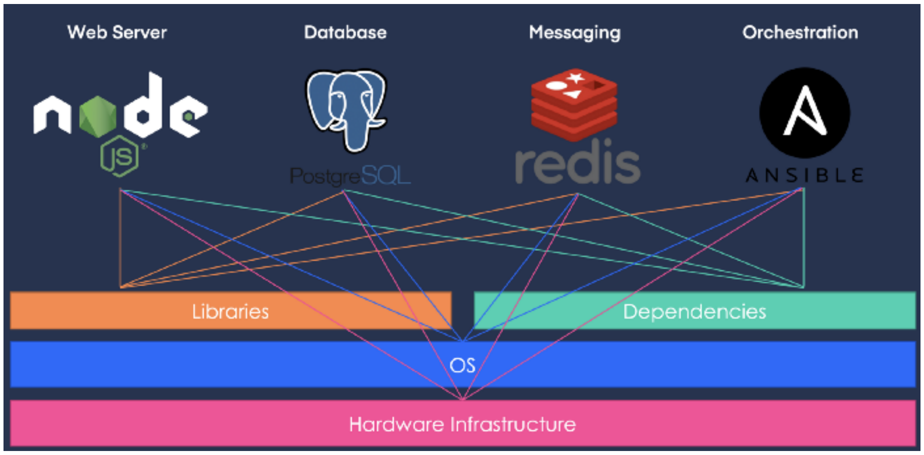

Figure 2: Life without container

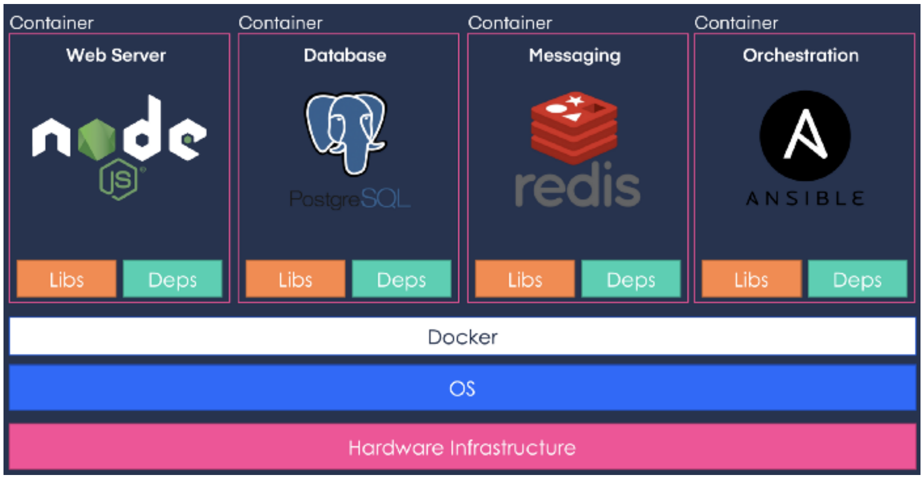

Figure 3: Life with container

Reference: * https://towardsdatascience.com/a-concise-guide-to-docker-f6b6d5fb56f4

History of containers: * https://dzone.com/articles/evolution-of-linux-containers-future

Docker is an engine that runs containers. These days internally it uses containerd, and Docker is more of an engine on its own.

Install docker: * https://docs.docker.com/engine/install/ubuntu/

1.3 Container examples / projects

Hello world:

```
sudo docker run hello-world
```

Exercise: what the following command mean ?

```
sudo usermod -aG docker $USER
```

Nginx server:

```
sudo docker run --name nginx -p 80:80 -d nginx
```

Exercises: - Check if nginx is running - Change listening port to 8080 - What the -d parameter means ? - Run nginx server with your index.html: https://hub.docker.com/_/nginx - What the -v parameter means ? - Explain `/some/content:/usr/share/nginx/html:ro` ? - List running containers ?

Advanced exercises: Dockerizing a Node.js web app: https://nodejs.org/en/docs/guides/nodejs-docker-webapp/

Python Development environment: https://hub.docker.com/_/python

More reading: dockerfile and docker compose * https://www.techrepublic.com/article/what-is-the-difference-between-dockerfile-and-docker-compose-yml-files/

2. Cloud native ecosystem

"Cloud native computing is an approach in software development that utilizes cloud computing to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Technologies such as containers, microservices, serverless functions and immutable infrastructure, deployed via declarative code are common elements of this architectural style". Wikipedia.

Cloud native landscape: https://landscape.cncf.io/

Containers: https://landscape.cncf.io/card-mode?category=container-runtime&grouping=category

3. Linux containers modern application deployment

Linux container images provide portability and version control, it helps ensuring that applications will work on different production environment. It powers the : develop, test and deploy workflow.

See workshop Test-Driven Clean Architecture by Mohamed Cherif Bouchelaghem.

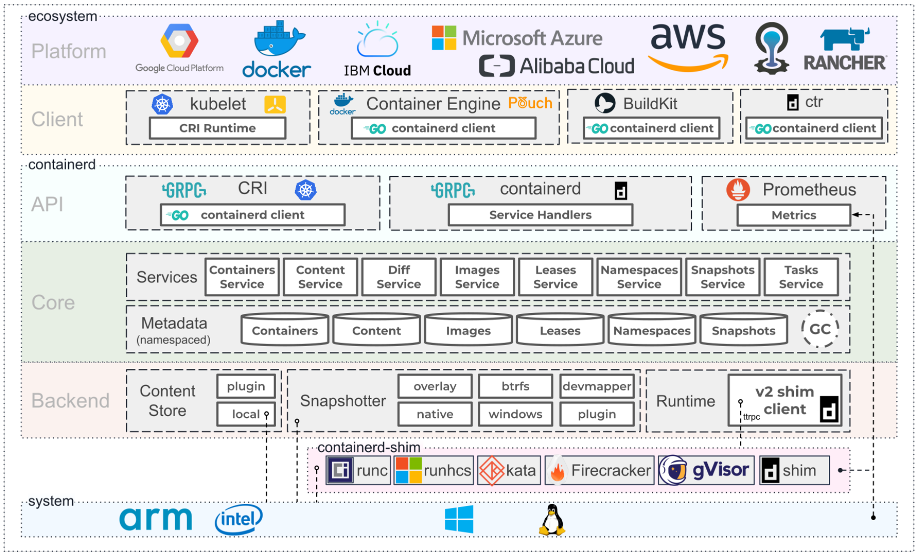

Figure 4: Linux cloud ecosystem

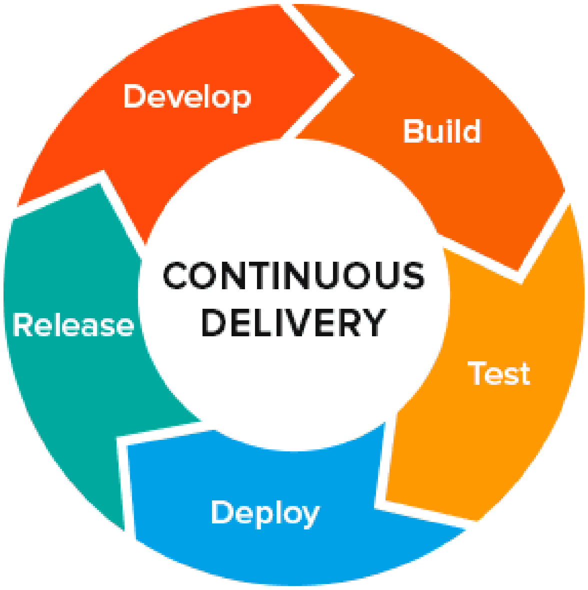

Figure 5: Continues Delivery

4. Container images

Container image is composed of:

- The root filesystem
- Some configuration
- Application

https://hub.docker.com/search?q=&type=image

Exercises:

- Install Alpine Linux image
- Inspect docker images "inspect, run, exec commands"

  ```
  docker run -it --rm alpine /bin/sh
  ```

- List processes inside and outside container

  ```
  ps
  ```

- Inspect filesystem

More exercises: - Deploy Redis key-value store and use it to store your data. - Deploy Wordpress + MYSQL: https://hub.docker.com/_/wordpress

4. Linux file permissions and capabilities

Linux Discretionary Access Controls (DAC): https://en.wikipedia.org/wiki/Discretionary_access_control

Command:

```
ls -lha ~/.bashrc
```

Output:

```
Permissions  Owner  Group
-rw-r--r-- 1 tixxdz tixxdz 3.5K Sep  3  2020 .bashrc
```

setuid and setgid: When you execute a program, the process inherits your user ID. If the file has the setuid bit set, the process will receive the user ID of the file's owner.

```
cp /usr/bin/sleep ./
sudo chown root.tixxdz sleep
sudo chmod u+s sleep
ls -lha sleep
-rwsr-xr-x 1 root tixxdz 39K Apr 10 06:43 sleep
```

Terminal 1:

```
./sleep 100
```

Terminal 2:

```
$ ps aux | grep sleep -
root      1647  0.0  0.0   5260   744 pts/14   S+   06:43   0:00 ./sleep 100
tixxdz    1652  0.0  0.0   6076   884 pts/16   S+   06:44   0:00 grep sleep -
```

Posix Access Control List (ACL): https://en.wikipedia.org/wiki/Access-control_list

Linux kernel Mandatory access control (MAC): https://en.wikipedia.org/wiki/Mandatory_access_control

Linux capabilities:

```
man capabilities
```

Commands `getcaps`, `setcaps`, `getpcaps` , etc

Privilege Escalation: When attackers get access to your web application or any other application, usually their injected code will run with the same privileges of that application. With capabilities and containers you can reduce privilege escalations which prevents extending the set of permitted operations.

In this case attackers will need to chain another exploit to elevate their privileges.

```
docker run -d --cap-drop=all --cap-add="cap_net_bind_service" -p 80:80 httpd
```

Exercise: - What does –cap-drop=all mean ? - What capabilities the httpd container is running with ?

Reference: https://dreamlab.net/en/blog/post/kernel-capabilities-in-docker-a-fine-grained-access-control-system/

5. Linux system calls

Linux applications run in userspace, when an application wants to do something like access a file it asks the kernel (Operating system) to perform it.

There are some +300 different system calls in Linux.

```
read read data from a file
write write data to a file
open open a file for subsequent reading or writing
execve run an executable program
clone create a new process
```

Manual for Linux system calls:

```
man syscalls
```

https://man7.org/linux/man-pages/man2/syscalls.2.html

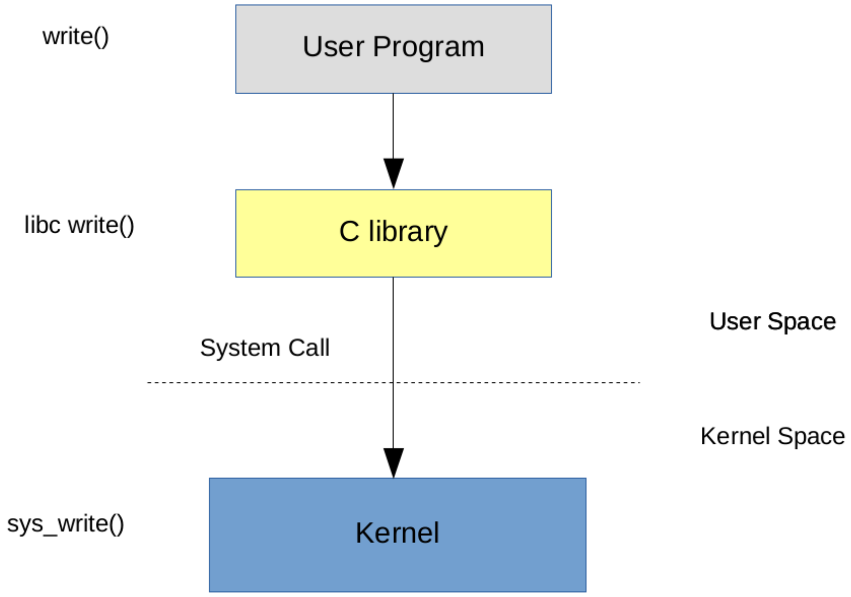

Figure 6: Linux system calls

6. Container isolation

6.1 Linux Namespaces

Linux namespaces are a set of primitives that controls what a process can use or see.

“Namespaces are a feature of the Linux kernel that partitions kernel resources such that one set of processes sees one set of resources while another set of processes sees a different set of resources. The feature works by having the same namespace for a set of resources and processes, but those namespaces refer to distinct resources. Resources may exist in multiple spaces. Examples of such resources are process IDs, hostnames, user IDs, file names, and some names associated with network access, and interprocess communication.” https://en.wikipedia.org/wiki/Linux_namespaces

https://man7.org/linux/man-pages/man7/namespaces.7.html

On startup there is a single namespace of each type.

To list the namespaces, use the command:

```
lsns
```

Output:

```
        NS TYPE   NPROCS   PID USER   COMMAND
4026531835 cgroup     17  1394 tixxdz /lib/systemd/systemd --user
4026531836 pid        17  1394 tixxdz /lib/systemd/systemd --user
4026531837 user       17  1394 tixxdz /lib/systemd/systemd --user
4026531838 uts        17  1394 tixxdz /lib/systemd/systemd --user
4026531839 ipc        17  1394 tixxdz /lib/systemd/systemd --user
4026531840 mnt        17  1394 tixxdz /lib/systemd/systemd --user
4026531992 net        17  1394 tixxdz /lib/systemd/systemd --user
```

Now privileged:

```
sudo lsns
```

Output:

```
        NS TYPE   NPROCS   PID USER             COMMAND
4026531835 cgroup    198     1 root             /lib/systemd/systemd --system --deserialize
4026531836 pid       198     1 root             /lib/systemd/systemd --system --deserialize
4026531837 user      197     1 root             /lib/systemd/systemd --system --deserialize
4026531838 uts       195     1 root             /lib/systemd/systemd --system --deserialize
4026531839 ipc       198     1 root             /lib/systemd/systemd --system --deserialize
4026531840 mnt       187     1 root             /lib/systemd/systemd --system --deserialize
4026531860 mnt         1    35 root             kdevtmpfs
4026531992 net       197     1 root             /lib/systemd/systemd --system --deserialize
4026532148 mnt         1 19867 root             /lib/systemd/systemd-udevd
4026532149 uts         1 19867 root             /lib/systemd/systemd-udevd
```

```
4026532150 mnt         1 20036 systemd-resolve  /lib/systemd/systemd-resolved
4026532151 mnt         1 19992 systemd-timesync /lib/systemd/systemd-timesyncd
4026532152 uts         1 19992 systemd-timesync /lib/systemd/systemd-timesyncd
4026532171 mnt         1   656 root             /usr/sbin/NetworkManager --no-daemon
4026532172 mnt         1   805 root             /usr/sbin/ModemManager --filter-policy=stric
4026532282 mnt         1   674 root             /usr/sbin/irqbalance --foreground
4026532283 mnt         1   698 root             /usr/libexec/switcheroo-control
4026532284 mnt         1   699 root             /lib/systemd/systemd-logind
4026532286 uts         1   699 root             /lib/systemd/systemd-logind
4026532289 net         1   961 rtkit            /usr/libexec/rtkit-daemon
4026532342 mnt         1  1091 root             /usr/lib/upower/upowerd
4026532343 user        1  1091 root             /usr/lib/upower/upowerd
4026532456 mnt         1  1327 colord           /usr/libexec/colord
```

Isolating the Hostname: https://medium.com/@teddyking/linux-namespaces-850489d3ccf

Isolating Process IDs: https://opensource.com/article/19/10/namespaces-and-containers-linux

Mount namespaces: https://lwn.net/Articles/689856/

Other namespaces https://man7.org/linux/man-pages/man7/namespaces.7.html

Control Groups https://www.kernel.org/doc/html/latest/admin-guide/cgroup-v2.html

7. Containers in practice

Docker Tutorial for Beginners - A Full DevOps Course on How to Run Applications in Containers:

https://www.youtube.com/watch?v=fqMOX6JJhGo

Dockerfile Best Practices: https://www.youtube.com/watch?v=JofsaZ3H1qM

8. Containers advanced security protections

8.1 Seccomp

Seccomp BPF (SECure COMPuting with filters): https://www.kernel.org/doc/html/v4.16/userspace-api/seccomp_filter.html

https://blog.selectel.com/containers-security-seccomp/

Linux security modules https://www.kernel.org/doc/html/latest/admin-guide/LSM/index.html

Linux eBPF https://ebpf.io/

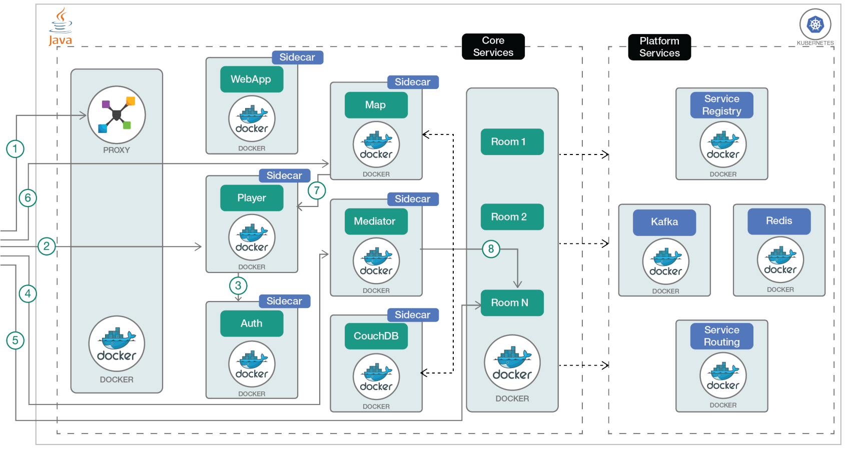

Figure 7: Linux system calls

9. Cloud deployment

https://kubernetes.io/

See workshop Discovering Kubernetes by Mr. Djelloul Bouida.

10. Conclusion

Modern continues deployment workflow dictates fast delivery.

In this workshop we saw Linux containers, introduced to docker tools.

We saw briefly Linux containers security and other Linux security mechanisms, attendees interested into the subject should continue and

References

- https://containerlabs.kubedaily.com/LXC/
- https://github.com/Fewbytes/rubber-docker
- https://en.wikipedia.org/wiki/Linux_namespaces
- https://towardsdatascience.com/a-concise-guide-to-docker-f6b6d5fb56f4
- Book “Container Security Fundamental Technology Concepts that Protect Containerized Applications” by Liz Rice.