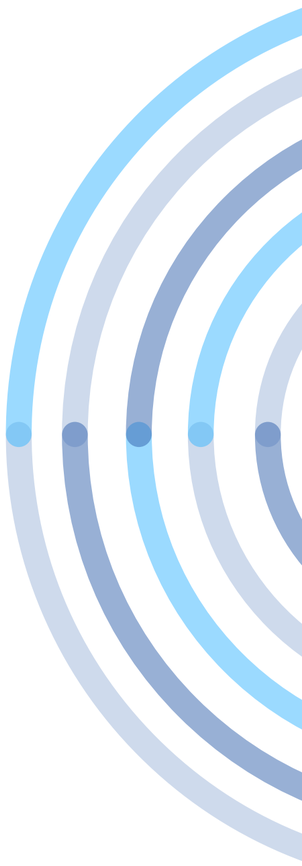
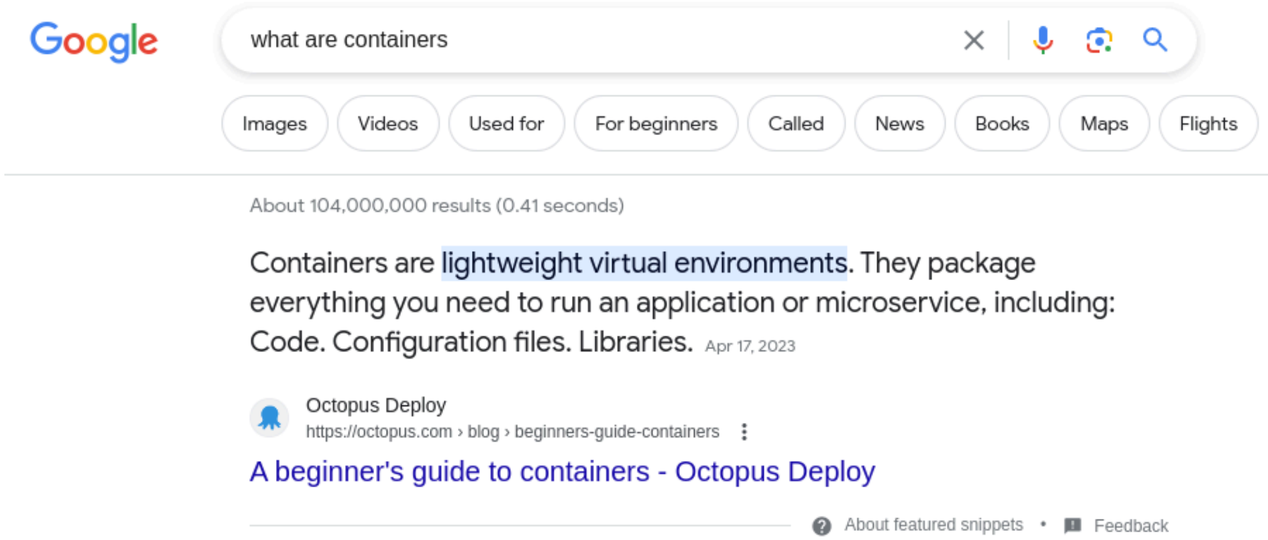




Unveiling containers

Alberto Pimpo


IT WORKS ON MY MACHINE
THEN WE'LL SHIP YOUR MACHINE
AND THAT IS HOW DOCKER WAS BORN



For a long time, companies have been using container technologies to address the weak points of virtual machines. We can think of containers as **more lightweight versions of VMs**. The important difference between containers and VMs is that containers don't need their own operating system. All containers on a host share that host's operating system, which frees up a lot of system resources.

“Containers are not a thing, it's just giving a name to the use of namespaces and cgroups.”

- Some passive aggressive dude on Reddit

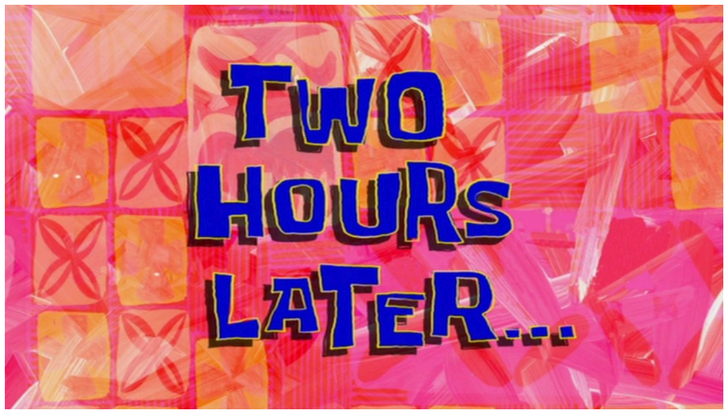
How to approach such a technology?

Bottom-up

- Learning golang
- Studying and became proficient with the codebase
- Understanding how the codebase interacts with the kernel

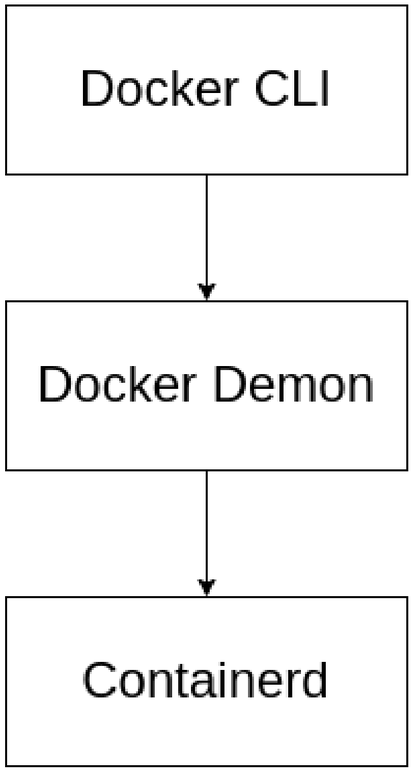
Top-down

- Starting to understand how external layer works and then going deeper



What we installed?

```
sudo yum install docker-ce docker-ce-cli containerd.io
```



The client that calls the Docker APIs

The demon that implements the Docker APIs

The “container runtime” compliant to industry standards

```
[ber@docker-demo ~]$ sudo strace -f -p `pidof containerd` -o strace_log
```

```
27243 execve("/usr/bin/runc", ["runc", "--root", "/var/run/docker/runtime-runc/mob"..., "--log"
, "/run/containerd/io.containerd.ru"..., "--log-format", "json", "--systemd-cgroup", "create",
"--bundle", "/run/containerd/io.containerd.ru"..., "--pid-file", "/run/containerd/io.containerd
.ru"..., "--console-socket", "/tmp/pty4243692610/pty.sock", "ca6b56de0f9336818bd1d3a1144f91a2".
..], 0xc0001280a0 /* 8 vars */ <unfinished ...>
```

```
27253 prctl(PR_SET_NAME, "runc:[1:CHILD]") = 0
27253 unshare(CLONE_NEWNS|CLONE_NEWUTS|CLONE_NEWIPC|CLONE_NEWPID|CLONE_NEWNET) = 0
```

Host

```
[ber@docker-demo ~]$ ps -ax
    PID TTY      STAT   TIME COMMAND
      1 ?        Ss     0:05 /usr/lib/systemd/systemd --switched-root --system --deserialize 31
      2 ?        S      0:00 [kthreadd]
      3 ?        I<     0:00 [rcu_gp]
```

```
  24642 ?        I      0:00 [kworker/u4:2-events_unbound]
  24644 ?        I      0:00 [kworker/u4:0-events_unbound]
  24651 ?        I      0:00 [kworker/1:0]
  24689 pts/0    S+     0:00 watch ps -ax
  24750 pts/1    R+     0:00 ps -ax
```

```
ubuntu@85e2ca00001e:~$ watch ps -ax
```

```
Every 2.0s: ps -ax

    PID TTY      STAT   TIME COMMAND
      1 pts/0    Ss     0:00 /bin/bash
    586 pts/0    S+     0:00 watch ps -ax
    599 pts/0    S+     0:00 watch ps -ax
    600 pts/0    S+     0:00 sh -c ps -ax
    601 pts/0    R+     0:00 ps -ax
```

Container

Takeaway concepts

- Never use memes as source of information
- Even if a lot of articles says something, it is not necessary true
- Sometimes complex technologies are easy to experiment, always worth to try
- Containers are not VM, processes runs directly on the host machines

Any question?

Host

```
[ber@docker-demo shared_folder]$ id
uid=1000(ber) gid=1000(ber) groups=1000(ber),10(wheel),991(docker) context=uncon
fined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[ber@docker-demo shared_folder]$ ls -al
total 0
drwxr-xr-x. 2 ber ber  18 Aug 24 14:42 .
drwx------. 4 ber ber 175 Aug 24 14:55 ..
-rw-r--r--. 1 ber ber   0 Aug 24 14:42 file
```

```
ubuntu@502e844e7438:~/shared_folder$ id
uid=1001(ubuntu) gid=0(root) groups=0(root),27(sudo)
ubuntu@502e844e7438:~/shared_folder$ ls -al
total 0
drwxr-xr-x. 2    1000 1000 18 Aug 24 12:42 .
drwxr-x---. 1 ubuntu root 27 Aug 24 12:58 ..
-rw-r--r--. 1    1000 1000  0 Aug 24 12:42 file
```

Container