Least-privilege Microservices

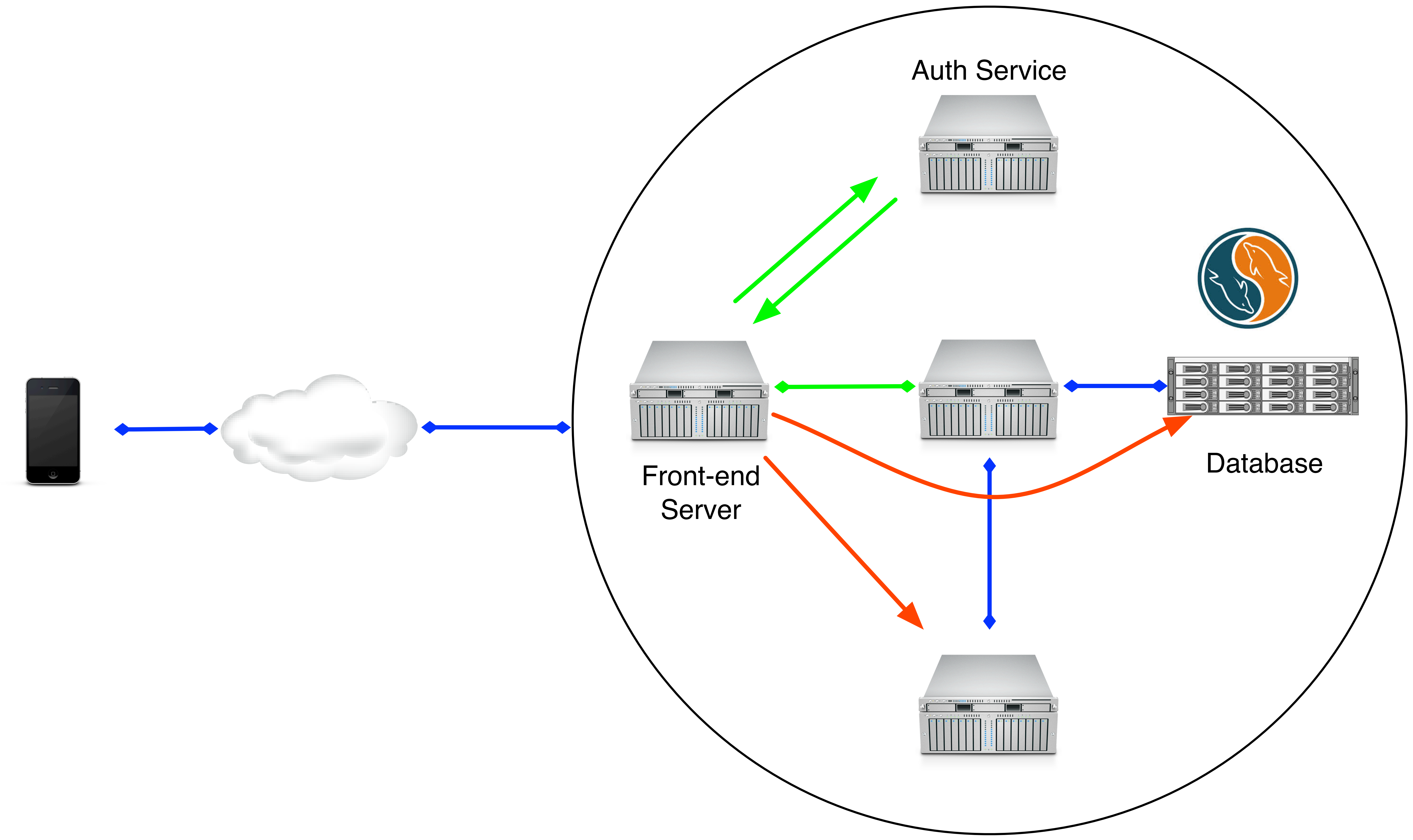
Diogo Mónica
Nathan McCauley

dockercon 15

Agenda

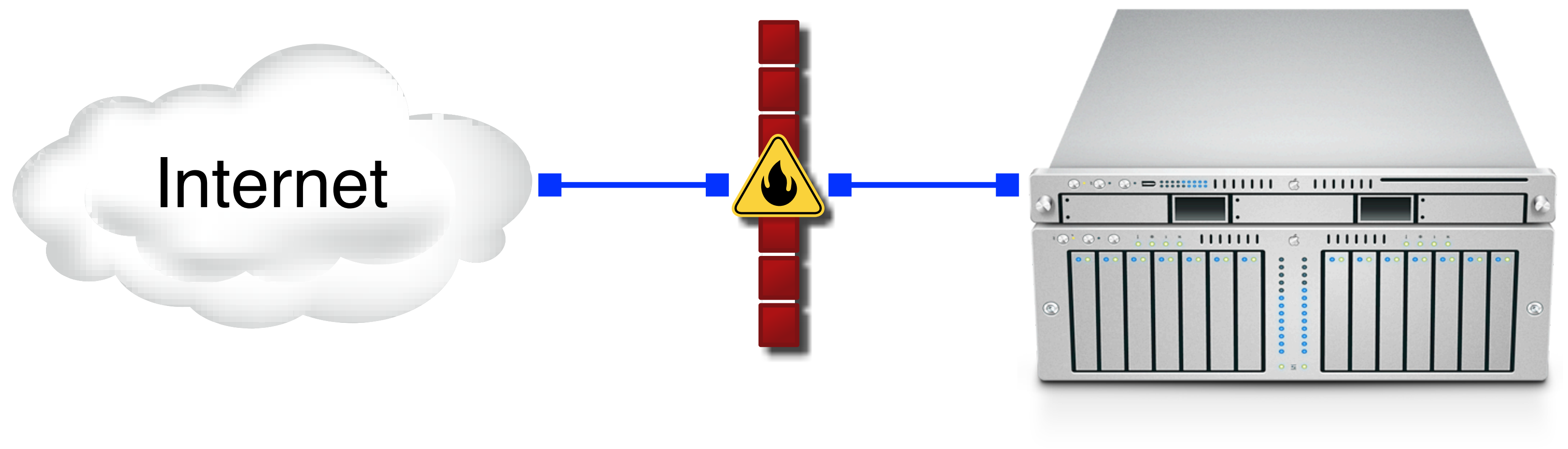
- Why least-privilege
- History of least-privilege
- Least-privilege with Docker
- Ongoing and future work
- Conclusions

“Every process must be able to access only the information and resources that are necessary for its legitimate purpose”


Auth Service
Front-end Server
Database

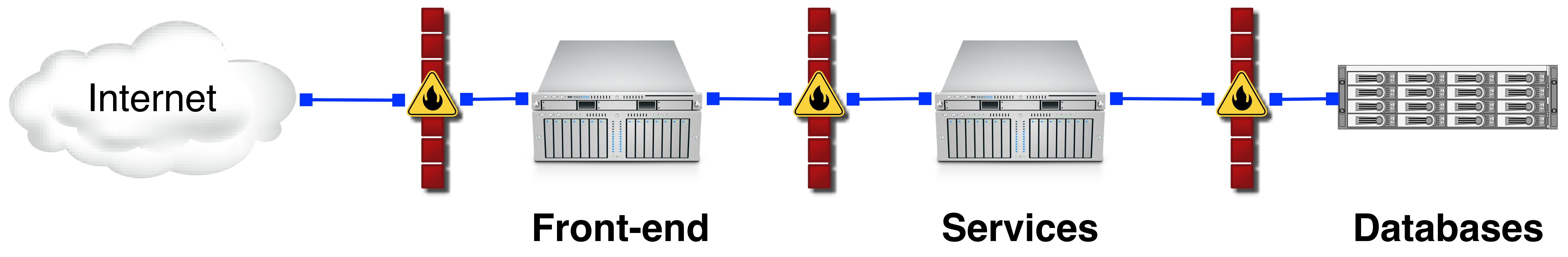
RESTRICTED AREA
AUTHORIZED PERSONNEL ONLY

1990

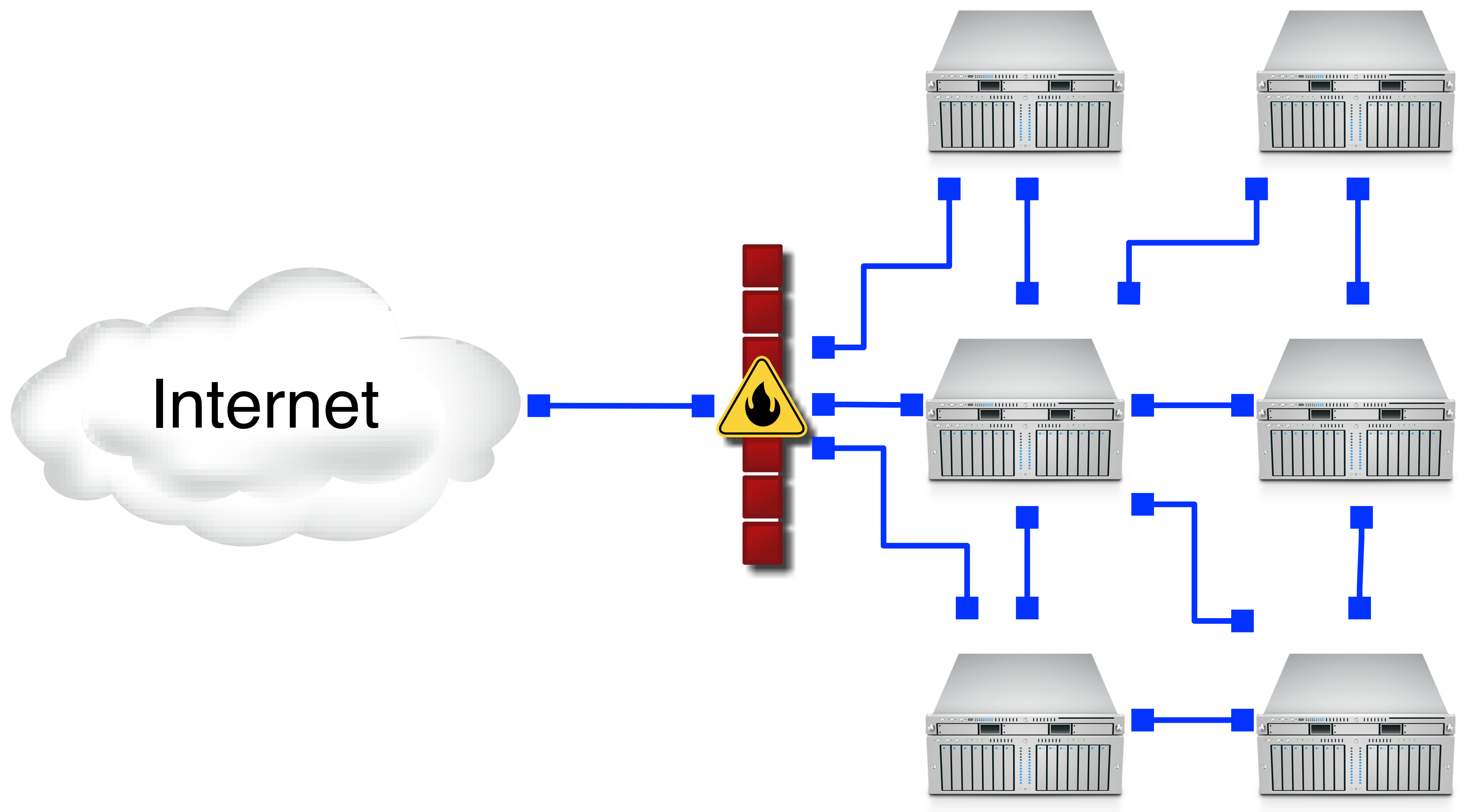


All-in-one

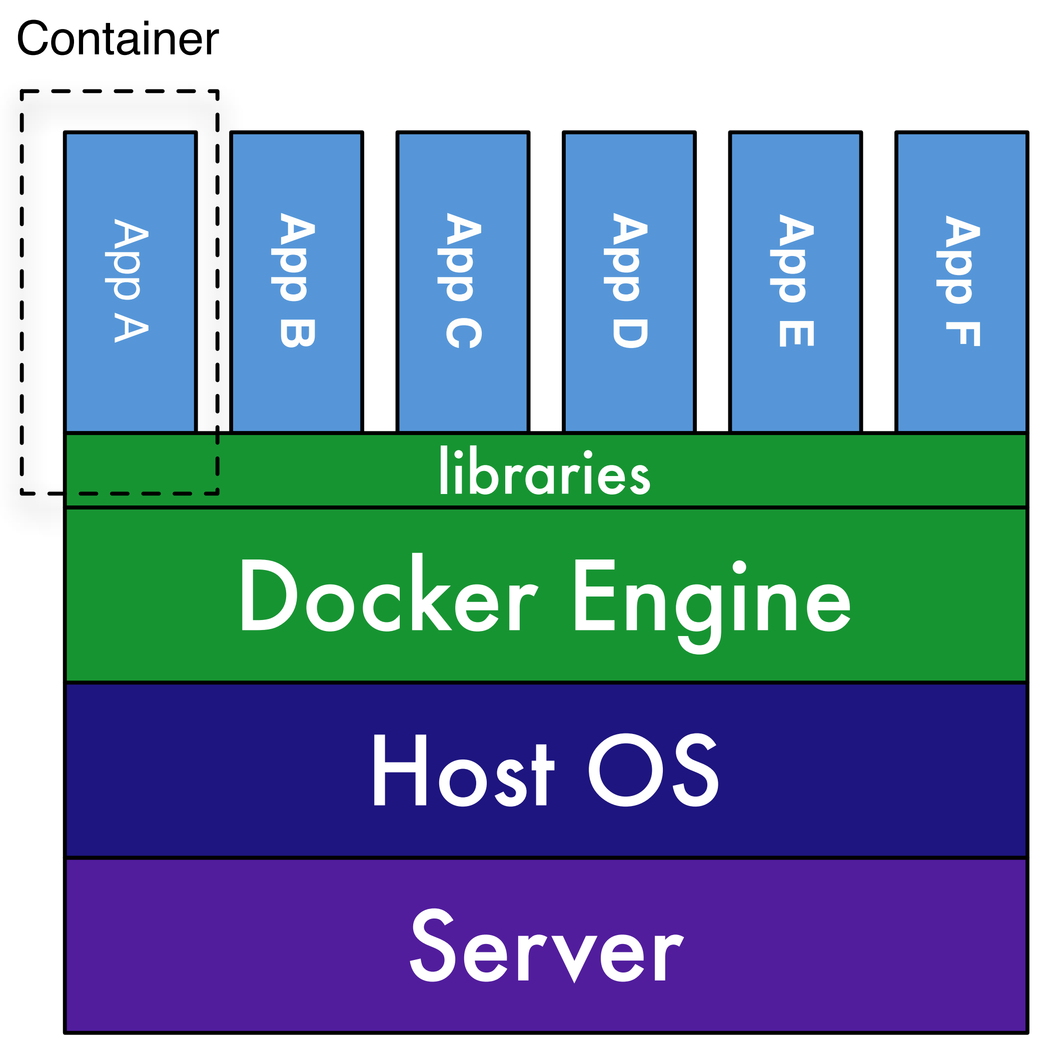
2000



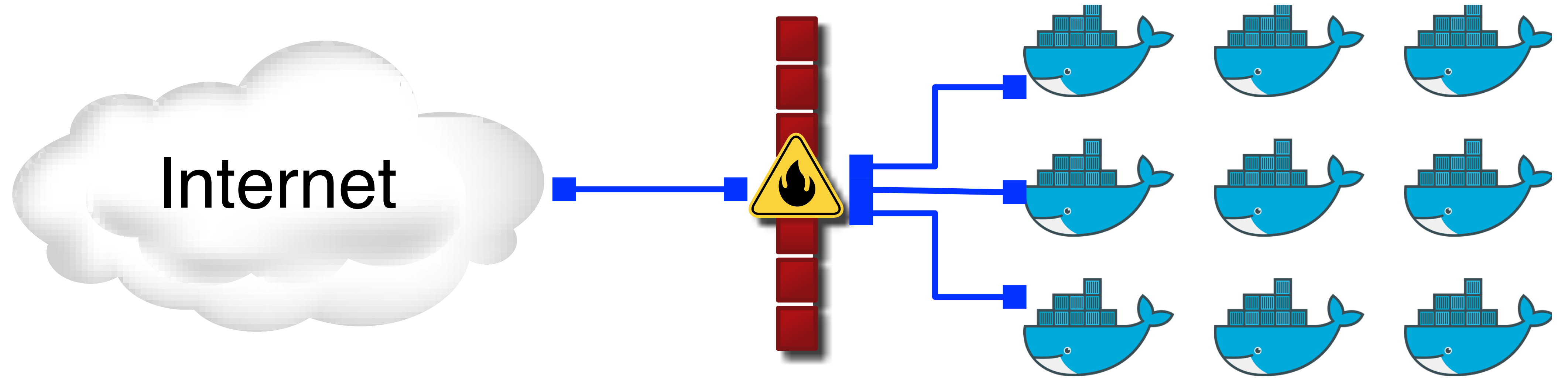
2010



One Process

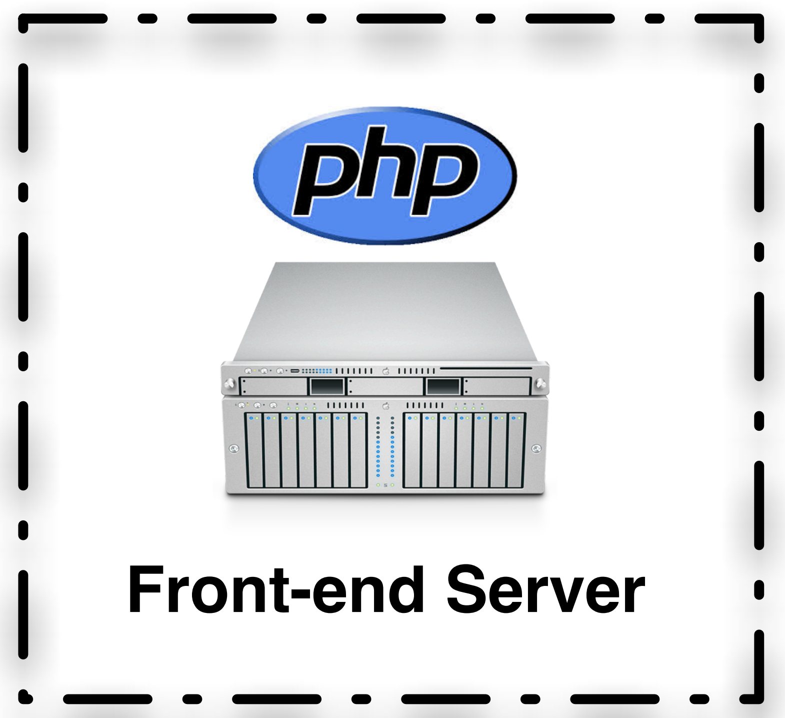

Container
App A
App B
App C
App D
App E
App F
libraries
Docker Engine
Host OS
Server

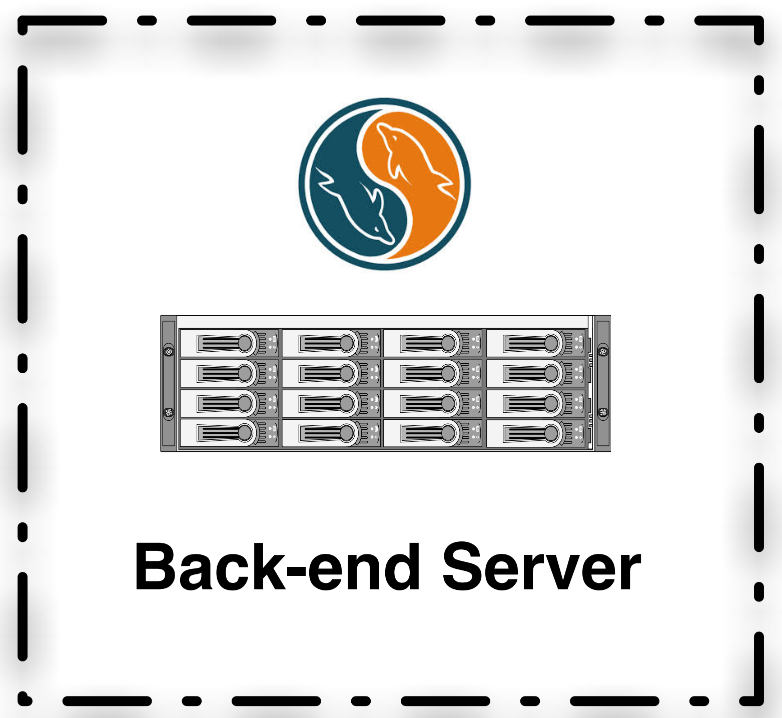
Today

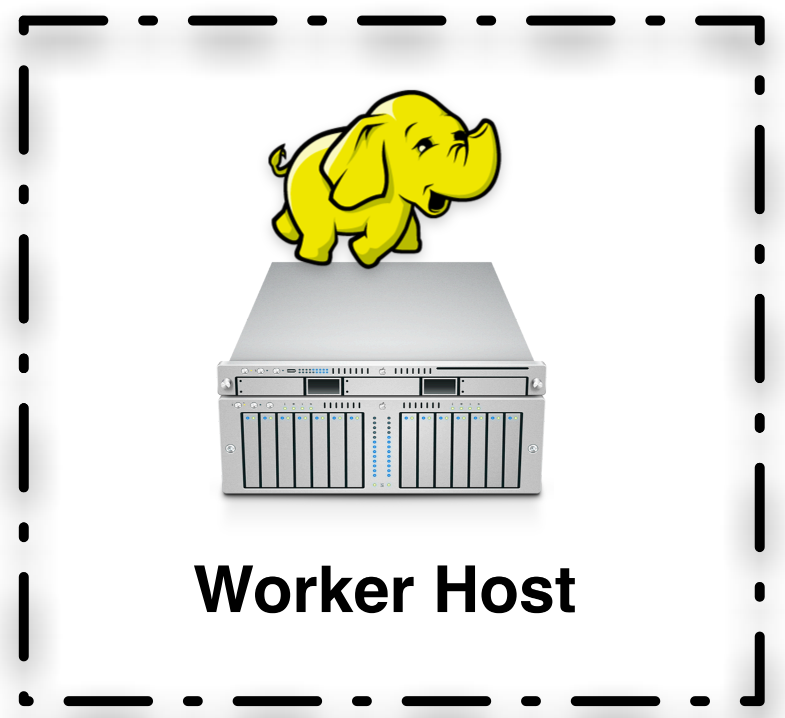


Profiles

- A FE server has a very different security profile than a database or a worker host
- Imagine that each container only has access exactly to the resources and APIs it needs. No more, no less.







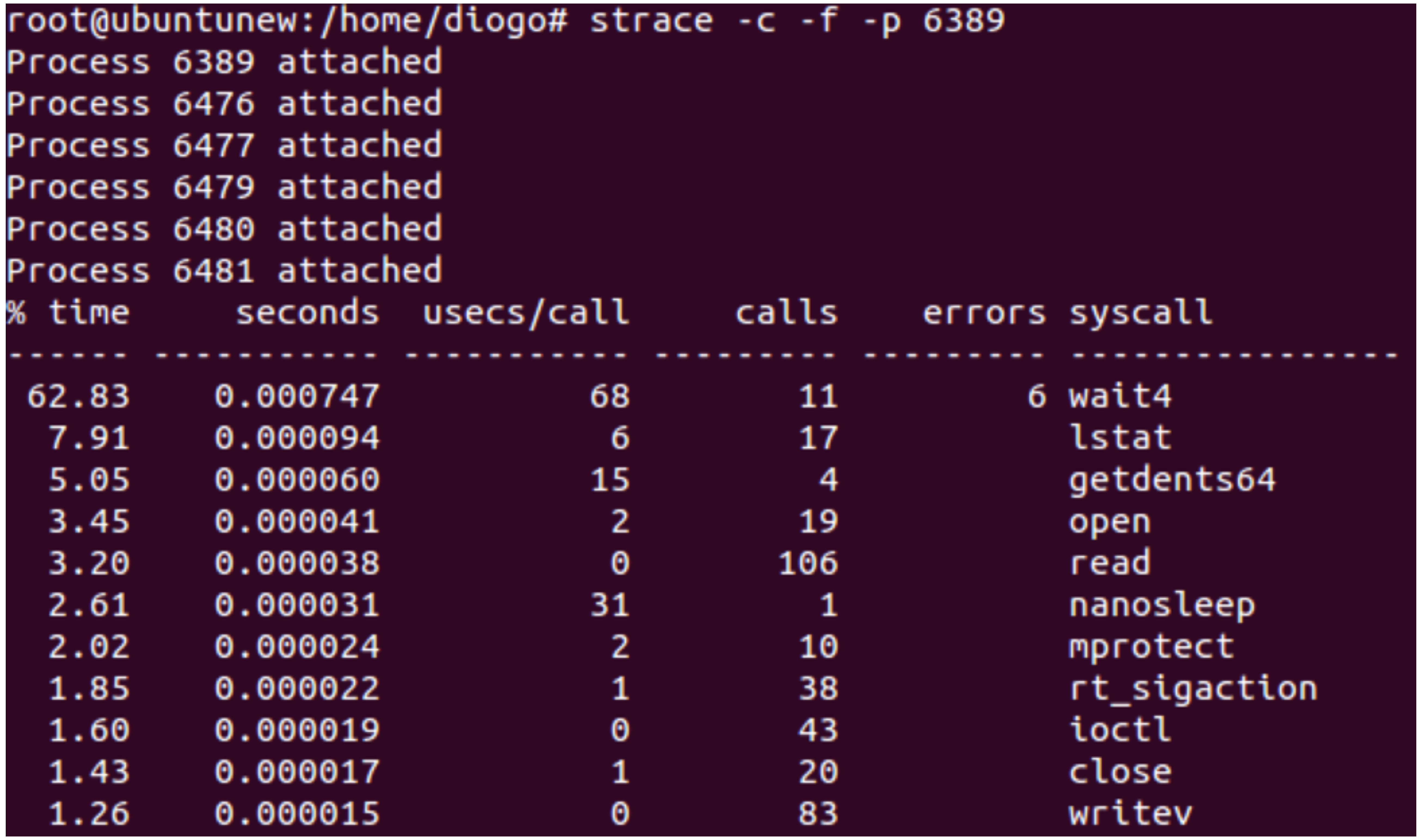
- Access to a lot of downstream services
- Most exposed

- I/O intensive
- Limited network access

- CPU Intensive
- Wide range of workloads

Process Monitoring

- A container is a process. Let's find out what syscalls it needs.

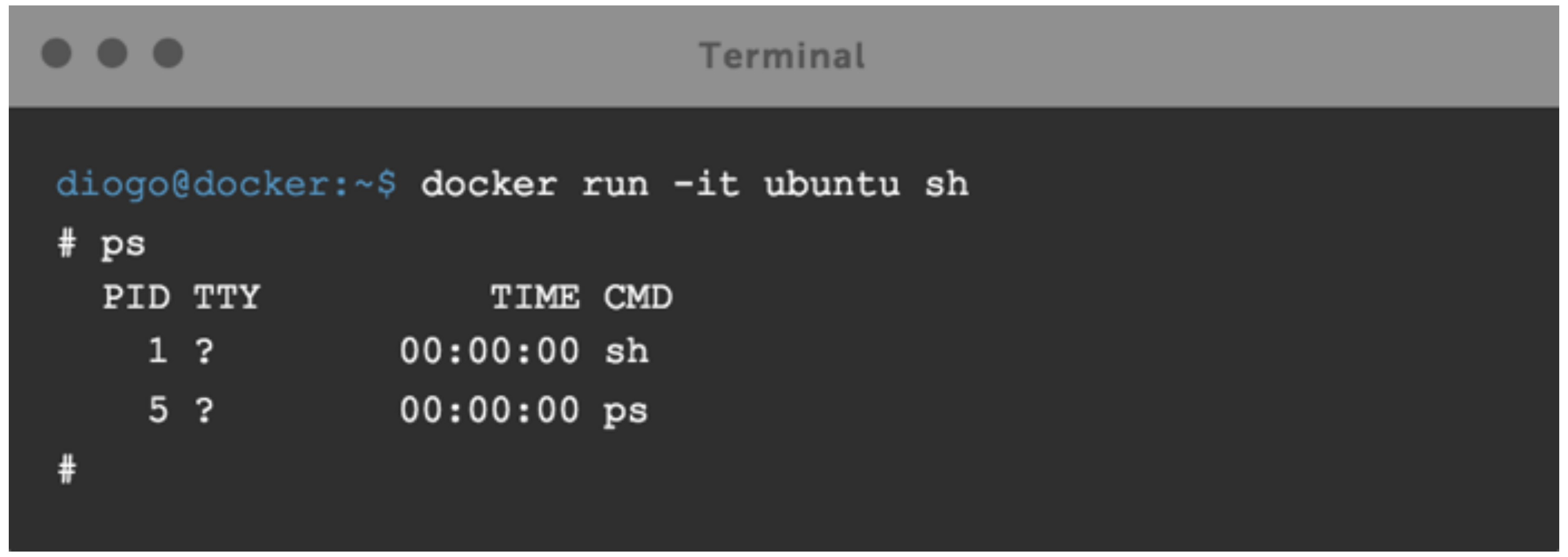


```
root@ubuntunew:/home/diogo# strace -c -f -p 6389
Process 6389 attached
Process 6476 attached
Process 6477 attached
Process 6479 attached
Process 6480 attached
Process 6481 attached
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- --------- ----------------
 62.83    0.000747          68        11         6 wait4
  7.91    0.000094           6        17           lstat
  5.05    0.000060          15         4           getdents64
  3.45    0.000041           2        19           open
  3.20    0.000038           0       106           read
  2.61    0.000031          31         1           nanosleep
  2.02    0.000024           2        10           mprotect
  1.85    0.000022           1        38           rt_sigaction
  1.60    0.000019           0        43           ioctl
  1.43    0.000017           1        20           close
  1.26    0.000015           0        83           writev
```

dockercon 15

Fine-grained controls

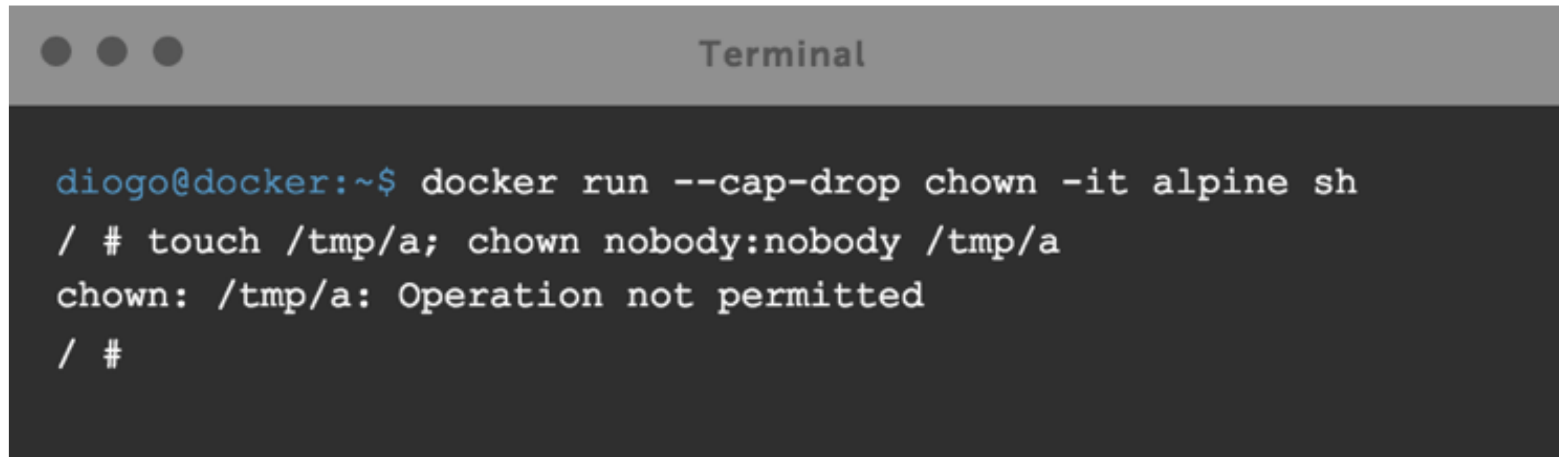
- Namespaces provide an isolated view of the system (Network, PID, etc)
- Cgroups limit and isolate the resource usage of a collection of processes
- Linux Security Modules give us a MAC (AppArmor, SELinux)



```
diogo@docker:~$ docker run -it ubuntu sh
# ps
  PID TTY          TIME CMD
    1 ?        00:00:00 sh
    5 ?        00:00:00 ps
#
```

Fine-grained controls

- Capabilities divides the privileges of root into distinct units (bind, chown, etc)
- Per-container ulimit (since 1.6)
- User-namespaces: root inside is not root outside (remapped root for 1.8)
- Seccomp: Individual syscall filtering (working on my laptop)

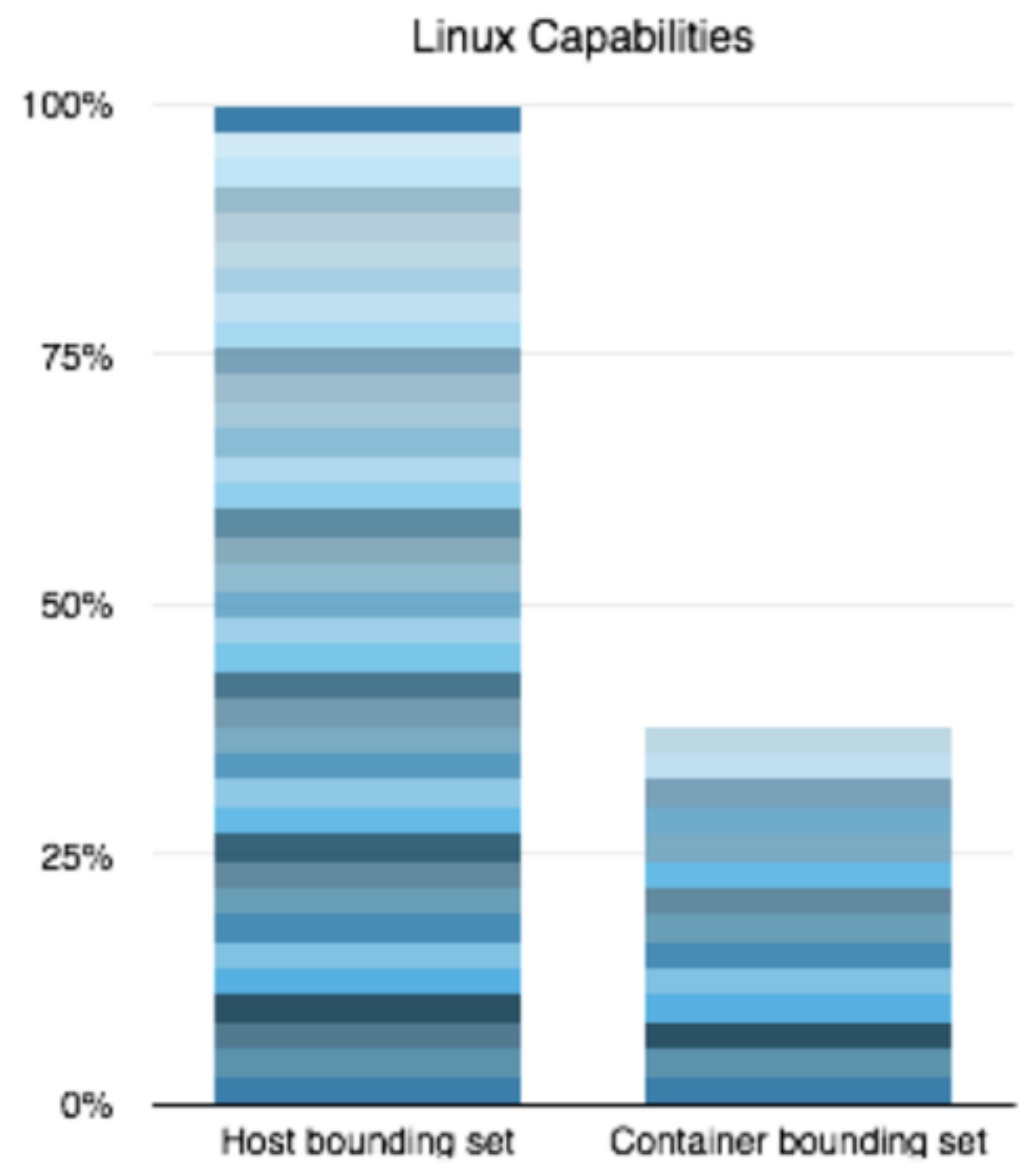


```
diogo@docker:~$ docker run --cap-drop chown -it alpine sh
/ # touch /tmp/a; chown nobody:nobody /tmp/a
chown: /tmp/a: Operation not permitted
/ #
```

dockercon 15

Safer by default

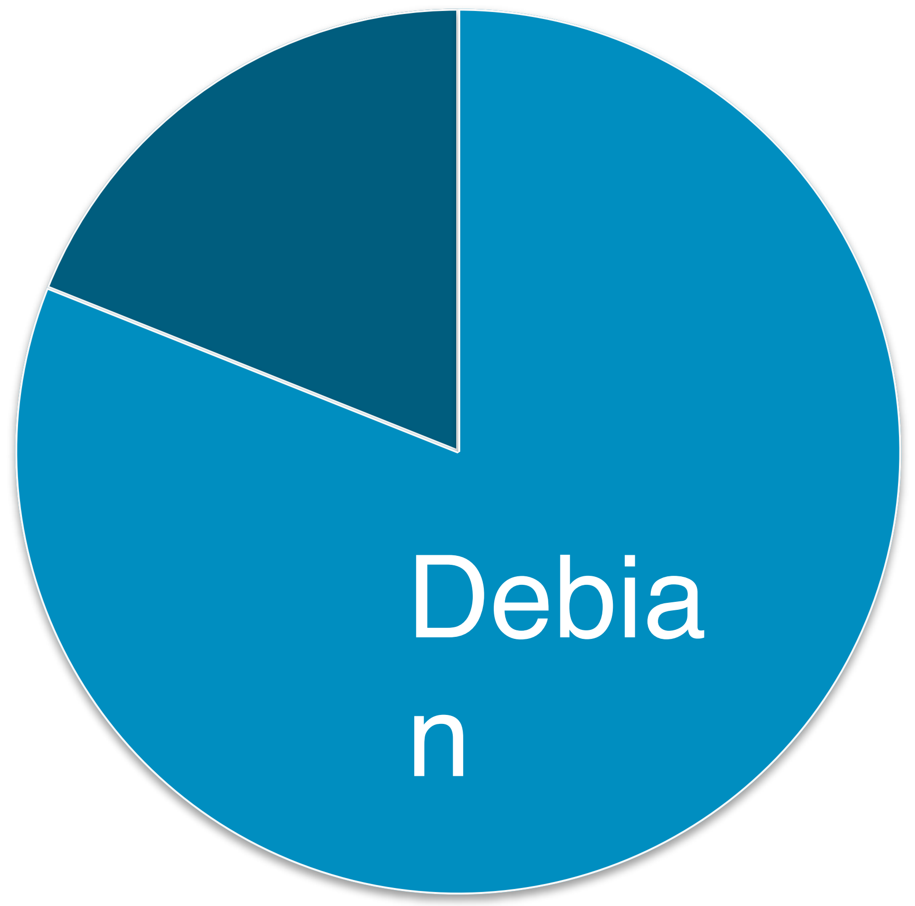
- Less than half the Linux capabilities by default
- Copy-on-write ensures immutability
- No device access by default
- Default AppArmor and SELinux profiles for an increasing number of containers



Safer by default

- Smaller footprint
  - Remove all unneeded packages
  - Remove all unneeded users
  - Remove all suid binaries

…



Security Profiles

- Producers of containers should be responsible for creating adequate profiles
- Profile gets shipped with the container
- Aggregates all of the different isolation mechanisms into one single profile

Securing the Ecosystem

IBM

User-namespaces

HUAWEI

Seccomp

intel

Provenance

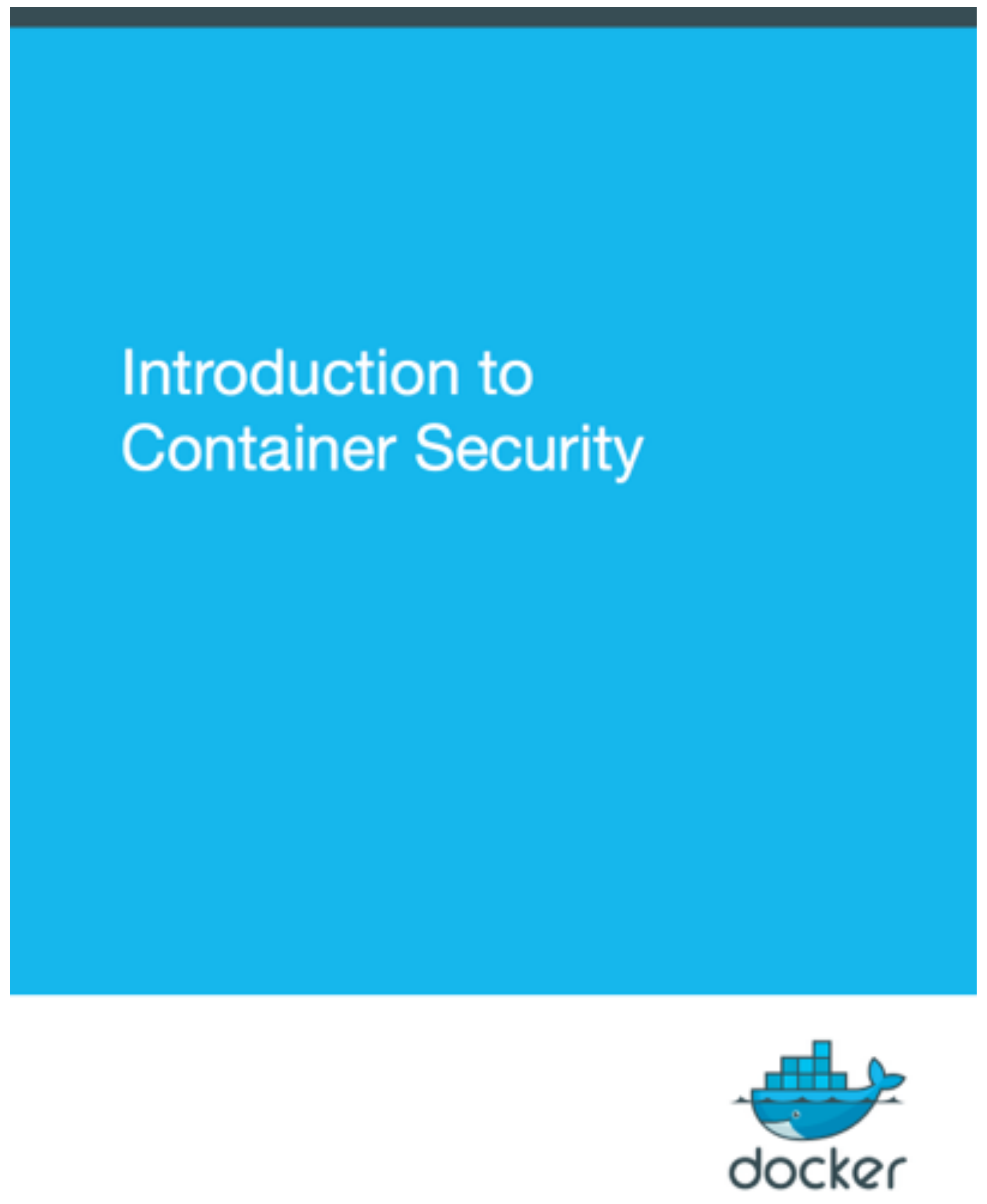
Selinux

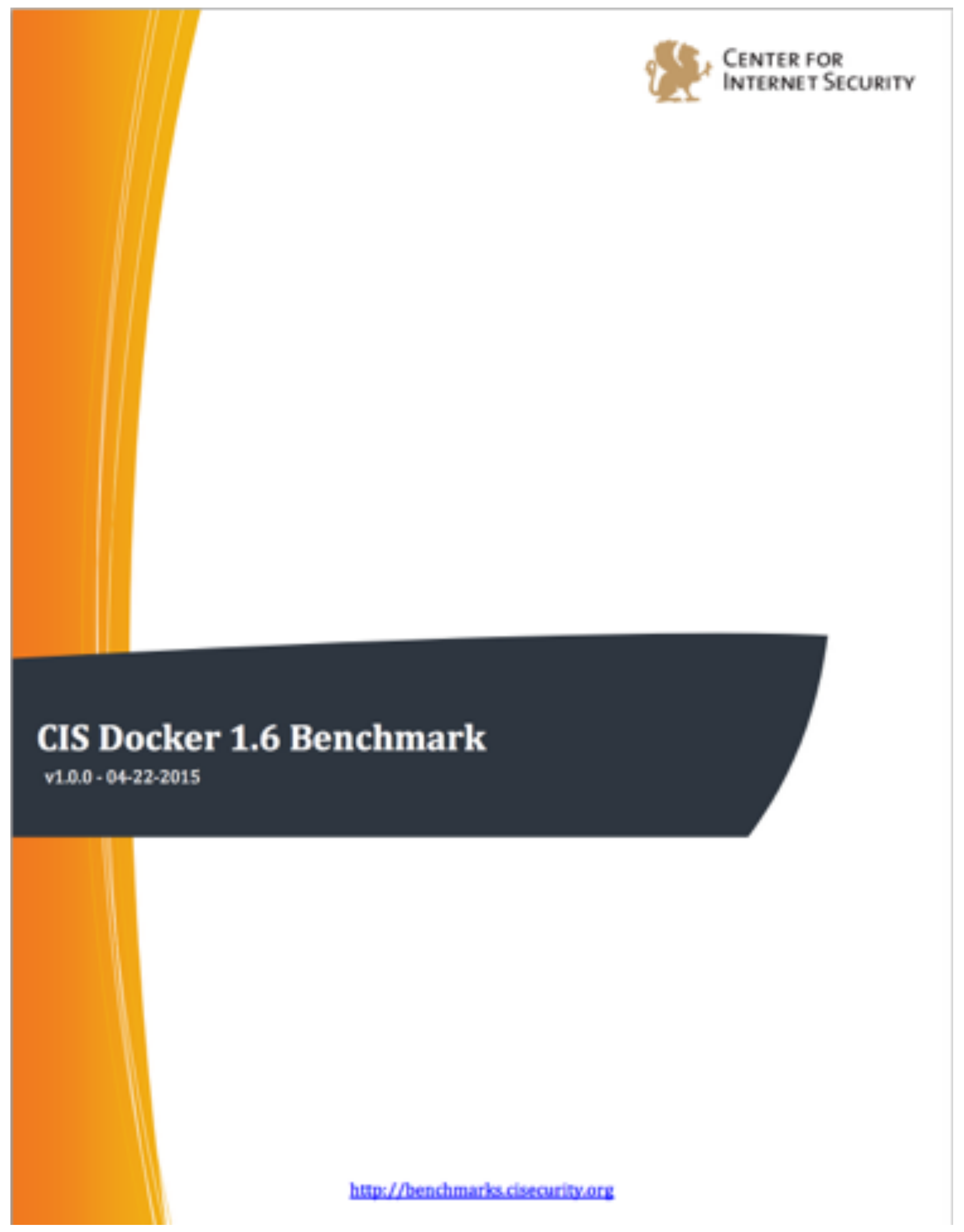
redhat

Kerberos

Twistlock

Intro to Container Security


Introduction to
Container Security

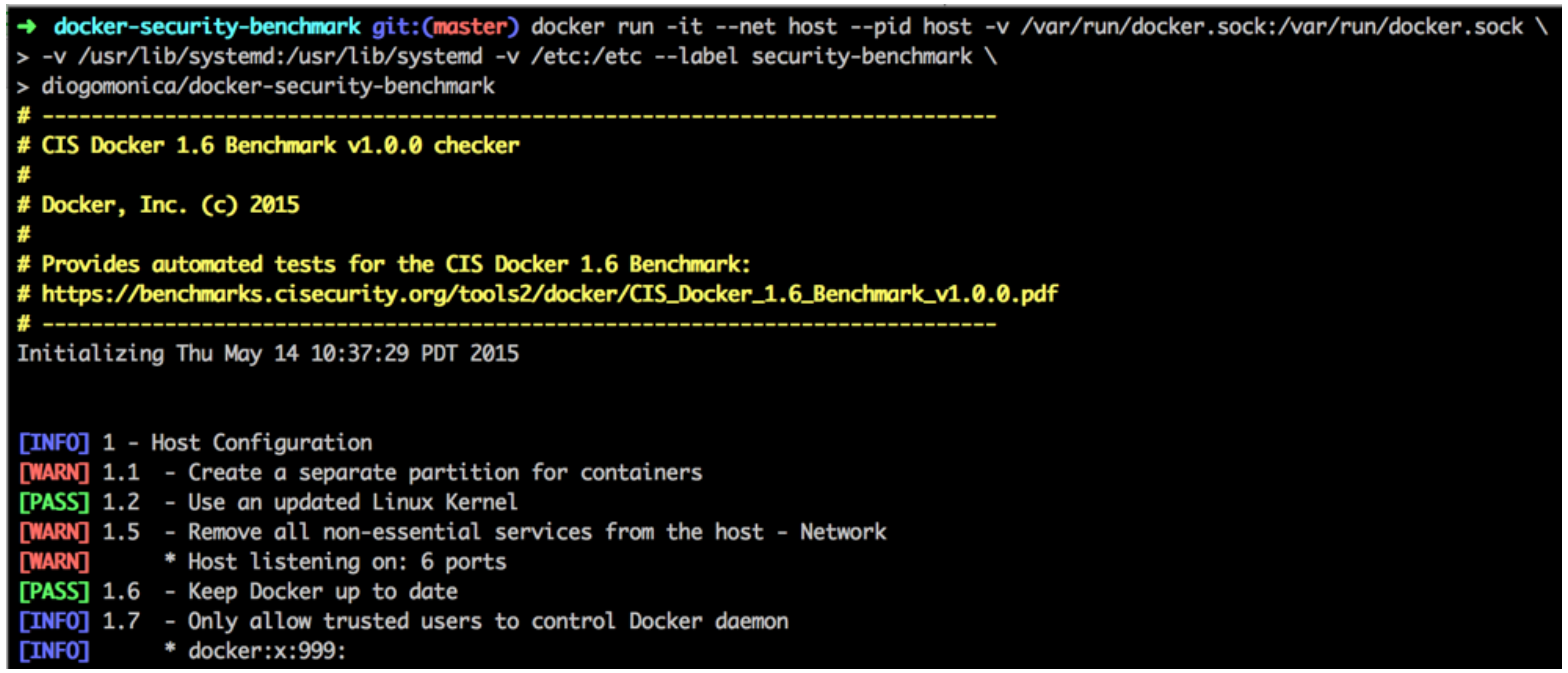


CIS Docker 1.6 Benchmark
v1.0.0 - 04-22-2015
http://benchmarks.cisecurity.org

http://bit.ly/1M4O9XE

Docker Bench

- Fully automated
- Shipped as a container that tests containers



```
➜  docker-security-benchmark git:(master) docker run -it --net host --pid host -v /var/run/docker.sock:/var/run/docker.sock \
> -v /usr/lib/systemd:/usr/lib/systemd -v /etc:/etc --label security-benchmark \
> diogomonica/docker-security-benchmark
# ------------------------------------------------------------------------------
# CIS Docker 1.6 Benchmark v1.0.0 checker
#
# Docker, Inc. (c) 2015
#
# Provides automated tests for the CIS Docker 1.6 Benchmark:
# https://benchmarks.cisecurity.org/tools2/docker/CIS_Docker_1.6_Benchmark_v1.0.0.pdf
# ------------------------------------------------------------------------------
Initializing Thu May 14 10:37:29 PDT 2015


[INFO] 1 - Host Configuration
[WARN] 1.1  - Create a separate partition for containers
[PASS] 1.2  - Use an updated Linux Kernel
[WARN] 1.5  - Remove all non-essential services from the host - Network
[WARN]      * Host listening on: 6 ports
[PASS] 1.6  - Keep Docker up to date
[INFO] 1.7  - Only allow trusted users to control Docker daemon
[INFO]      * docker:x:999:
```

https://dockerbench.com/

Conclusion

- Docker is on the path to support least-privilege microservices, since it allows fine-grained control over what access each container should have.
- We will need easier tooling to define per-container security profiles
  - You can help!

#docker-security on Freenode

Thank you

diogo@docker.com

nathan.mccauley@docker.com

dockercon 15