Docker and Security

Judith Kahrer

omega point.

Some definitions

What is a container?

A container is a standard unit of software that packages up code and all its dependencies so the application runs quickly and reliably from one computing environment to another.

docker.com

Container […] is a method to package an application so it can be run, with its dependencies, isolated from other processes.

www.techradar.com

omega point.

Containers provide a standard way to package your application's code, configurations, and dependencies into a single object.

Containers share an operating system installed on the server and run as resource-isolated processes, ensuring quick, reliable, and consistent deployments, regardless of environment.

aws.amazon.com

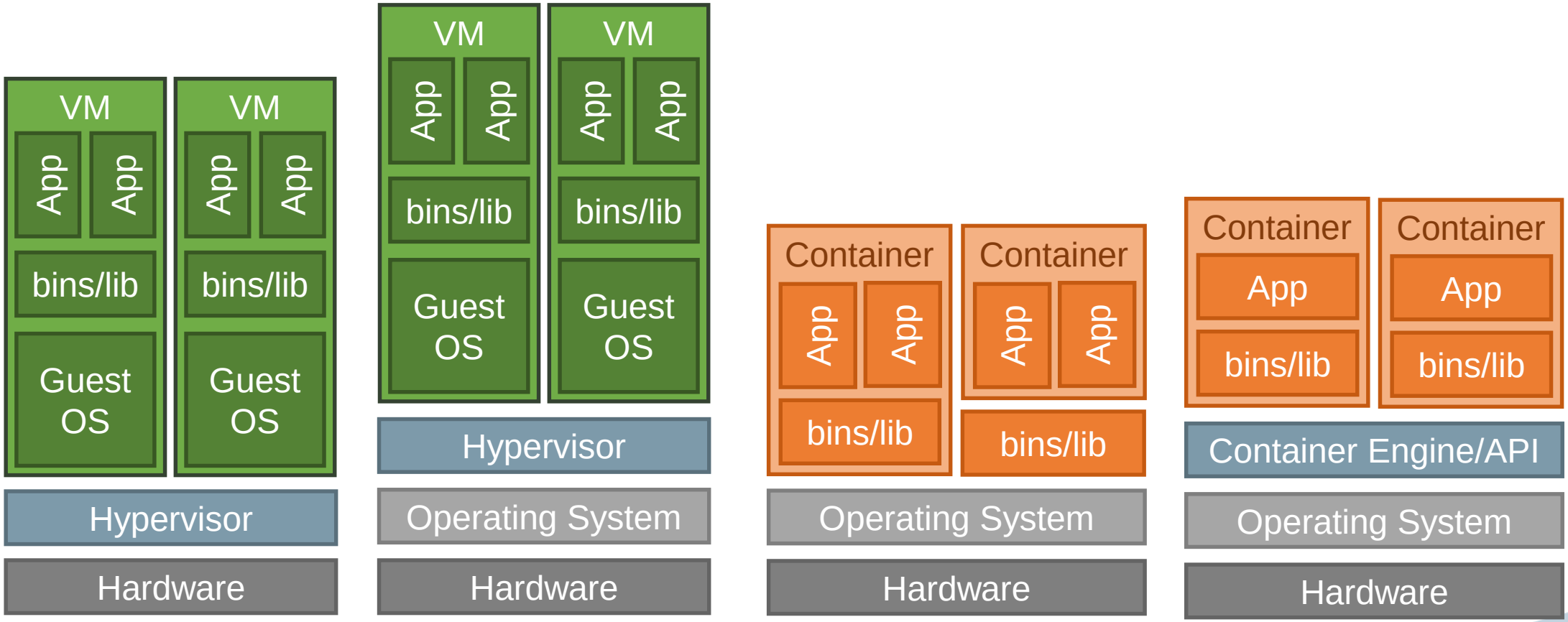
- Self-sufficient packages that enable reliable, consistent deployment of an application regardless of environment.
- Containers run as isolated processes on a target system.

In technical terms

- *It's a lightweight VM – it 'feels' like it!*
- *It's chroot on steroids*

Jérôme Petazzoni, Docker Inc.

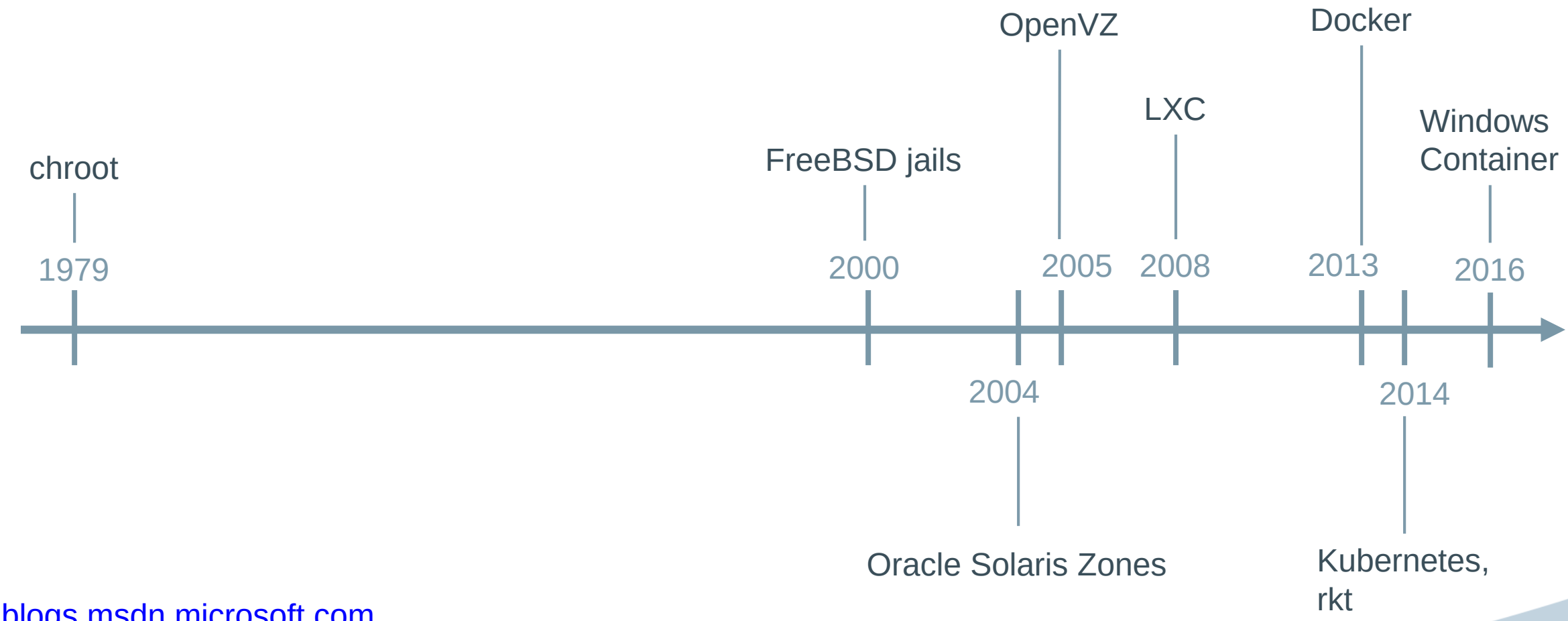
Virtualization



opennebula.org and blog.netapp.com

omega point.

Timeline



blogs.msdn.microsoft.com

omega point.

Getting started

Hello Docker

omega point.

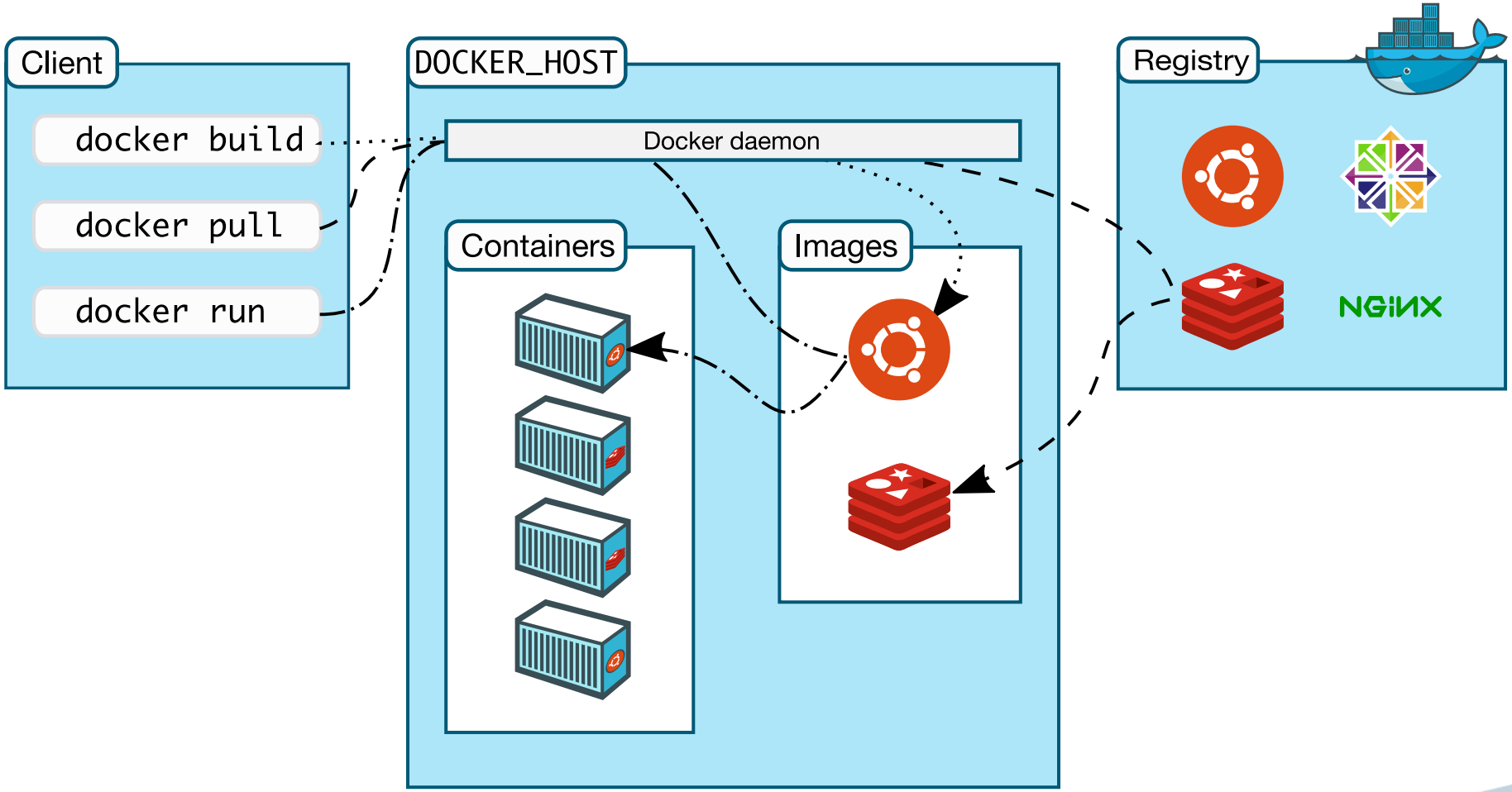
```
$ docker run hello-world
```

omega point.

```
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
d1725b59e92d: Pull complete
Digest: sha256:0add3ace90ecb4adbf7777e9aacf18357296e799f81cabc9fde470971e499788
Status: Downloaded newer image for hello-world:latest

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/get-started/
```

omega point.

Architecture



docker.com

Docker Hub

- Public and private repositories
- Free-to-use registry
- Community Docker images

Malicious Image

- Over 17 malicious Docker images
- Cryptomining containers

- Active for about 1 year
- 5 million pulls
- ~$90000

kromtech.com (june 2018)

Image Security

- Official images
  - https://hub.docker.com/official
- Docker Content Trust

- Docker Trusted Registry and Notary
- Docker Security Scanning

Enabling

Docker Content Trust

omega point.

```
$ export DOCKER_CONTENT_TRUST=1
```

omega point.

```
$ docker pull docker/trusttest
```

omega point.

remote trust data does not exist […]

omega point.

```
$ docker pull busybox
```

Know the Basics

Building Images

omega point.

Automated Builds

- Link Docker Hub with GitHub/Bitbucket
- Traceability between
  - Dockerfile
  - Version of the image

Dockerfile

- Text file with instructions to build an image

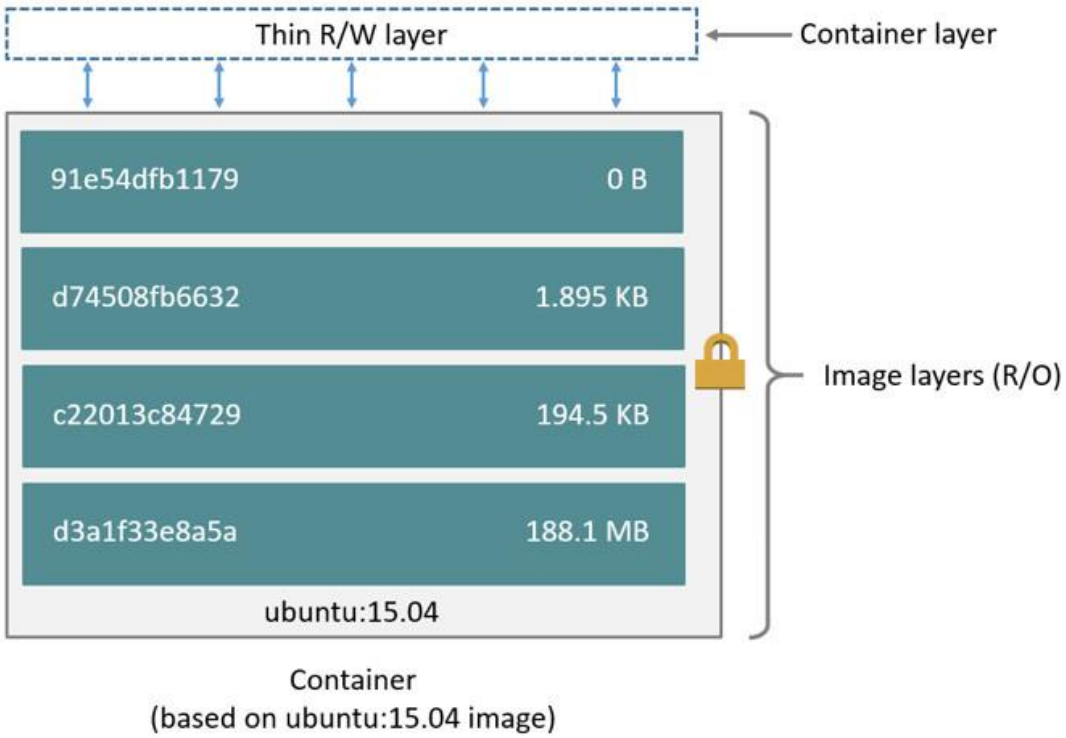
- May contain i.a.
  - Base image
  - Environment variables
  - Files to include
  - Entrypoint and commands to run

```
FROM busybox
ENV variable=visible
COPY somelocalfile /greetings/fromMe.txt
CMD ["/bin/cat", "/greetings/fromMe.txt"]
```

Layers

- Image Layers
  - Instructions from Dockerfile as read-only layers
  - Differences from previous layer
- Container Layer
  - Writeable layer on top during runtime

Layers



docs.docker.com

omega point.

```
$ docker history IMAGE
```

```
IMAGE          [...]  CREATED BY
11d8f9e12e42          [...] CMD ["/bin/cat" "/greetings/fromMe.txt"]
63dc92745629          [...] COPY file:3bff68af9383544cc9f1cd820647b69…
09cdd4f30470          [...] ENV variable=visible
788edf1f3e            [...] CMD ["sh"]
<missing>             [...] ADD file:63eebd629a5f7558c361be0305df5f16b…
```

omega point.

Get into the Internals

Kernel Features

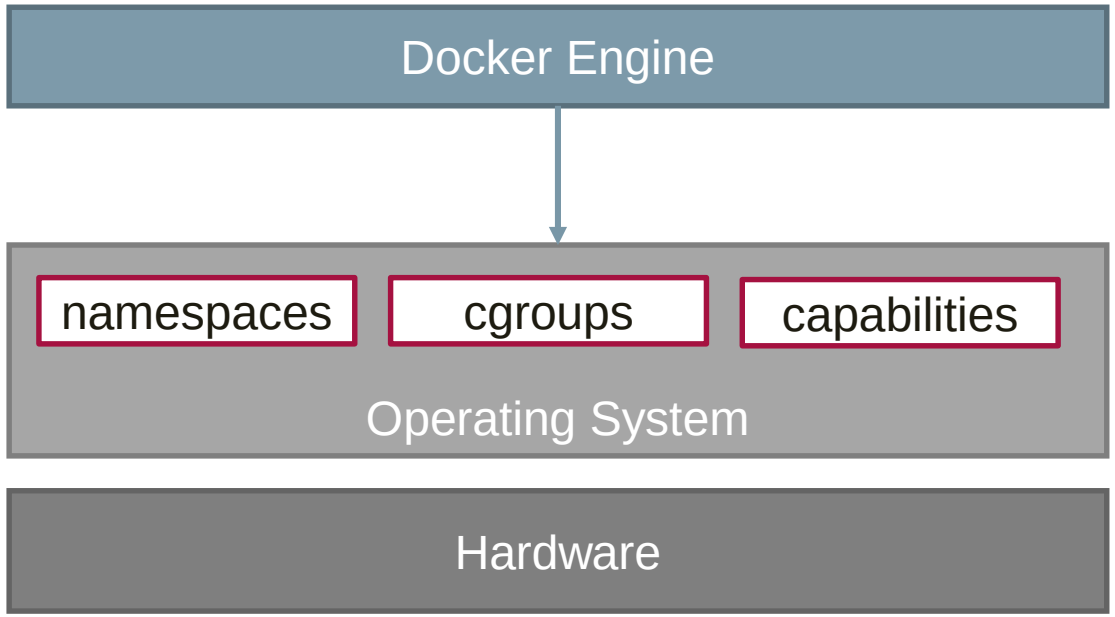
Inside Docker

- Docker containers create an isolated complete execution environment

Building Blocks

- Isolation
  - by namespaces
- Resource utilization
  - with the help of control groups (cgroups)
- Security
  - in form of capabilities

Inside Docker

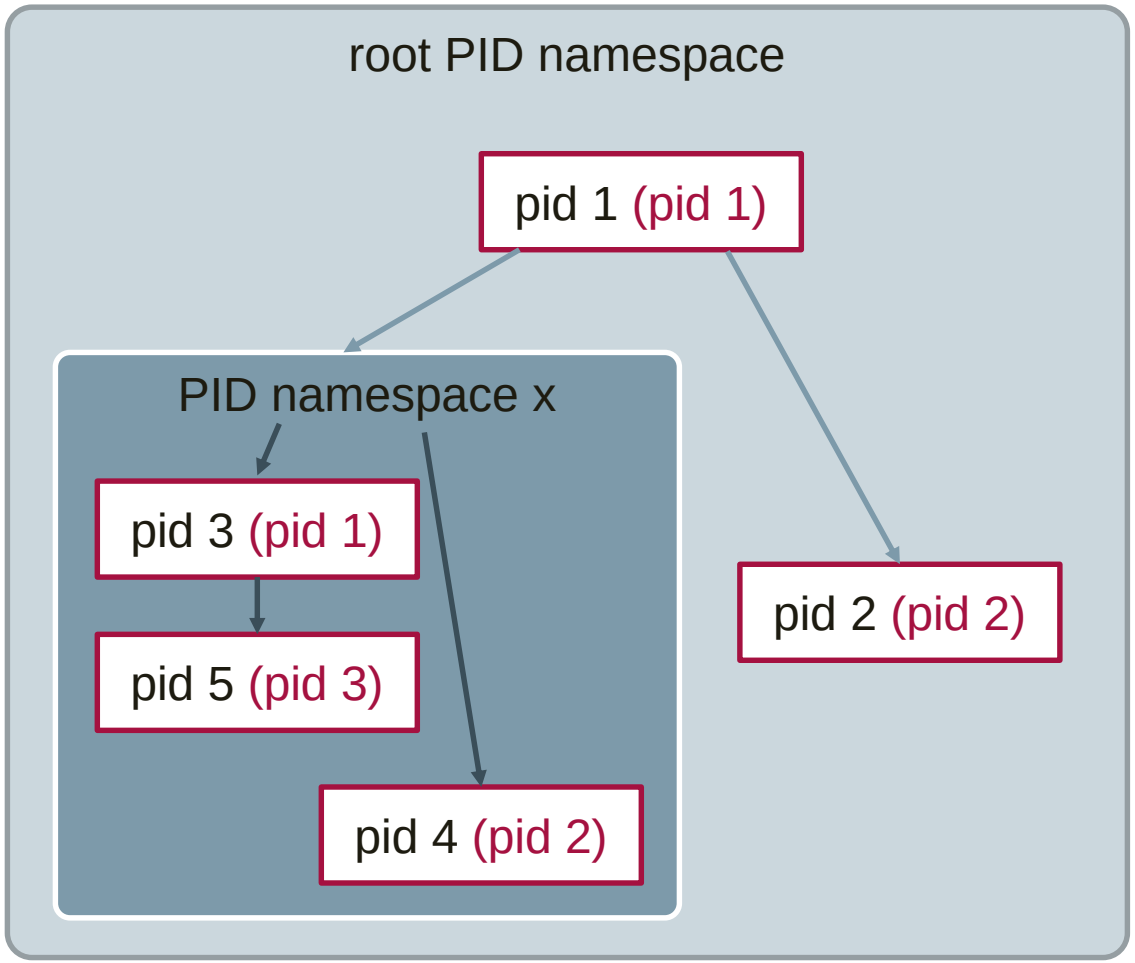


medium.com

Namespaces

- Provide processes with their own view of the system
  - mnt (mount points, filesystems)
  - pid (processes)
  - uts (hostname)
  - ipc (interprocess communication)
  - net (network stack)
  - user (UIDs)

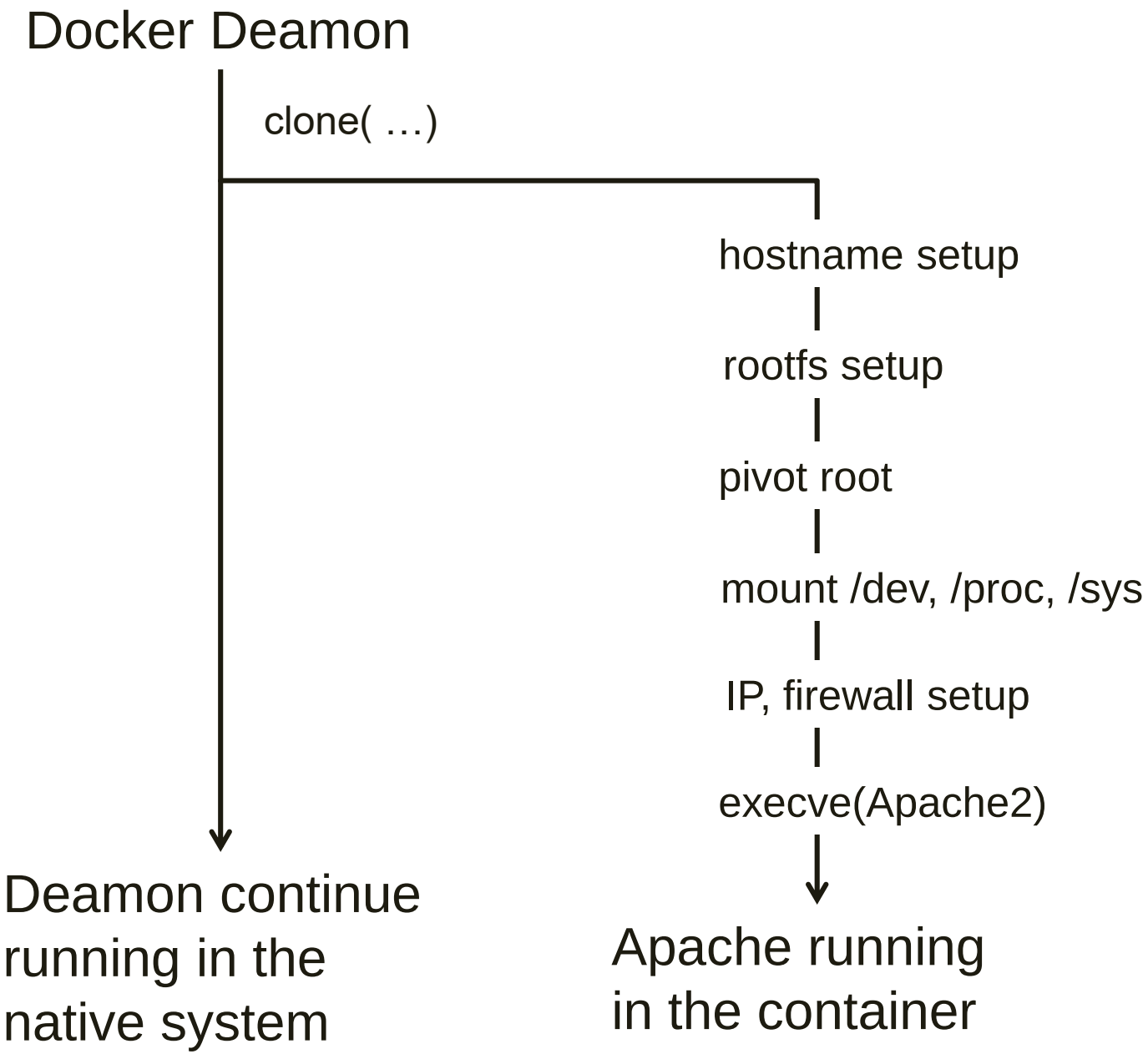
PID namespace



black: real PID
red: PID in namespace

medium.com

omega point.



Security Namespace: Making Linux Security Frameworks Available to Containers, Sun et al. (2018)

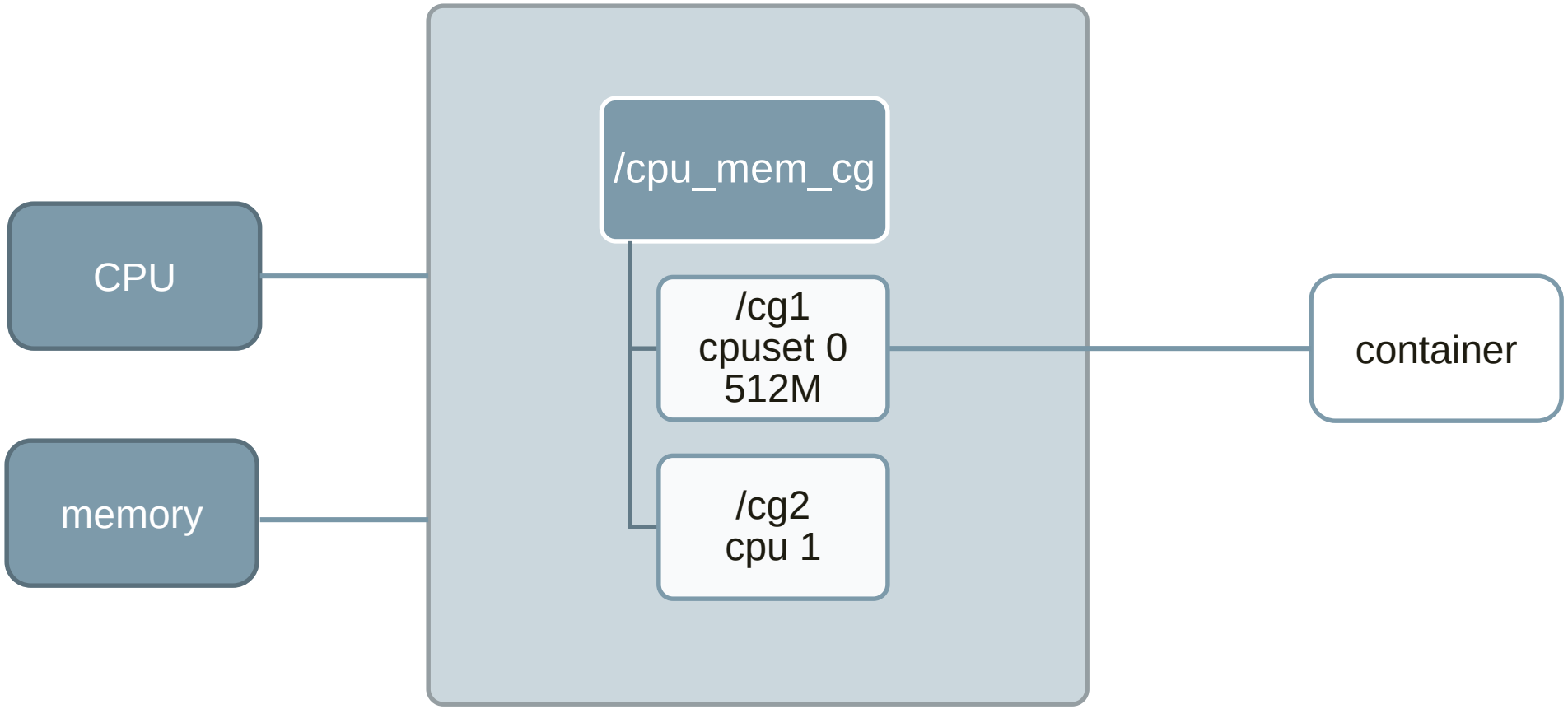
cgroups

- Hierarchical groups of processes
- Resource accounting and limiting
  - memory
  - CPU time
  - disk I/O
  - network

Limit Resources

- --memory=<value>
- --memory-swap=<value>
- --oom-kill-disable
- --cpus=<value>
- --cpuset-cpus=<value>
- …

cgroup example



access.redhat.com

```
$ docker run --cpuset-cpus 0 -m 512MB <id>
```

omega point.

```
$ docker stats --no-stream
```

omega point.

Capabilities

- Docker container runs as root (default)
  - with a subset of capabilities
- Divide the power of superuser

Default capabilities

- CHOWN
- DAC_OVERRIDE
- FOWNER
- NET_RAW
- …

```
$ docker run --cap-drop ALL --user <uid>:<gid> <id>
```

omega point.

Secure Computing Mode

- seccomp
- Restrict system calls from the container
- More fine-grained control than capabilities
- Docker provides default whitelist

Docker Awareness in Java

- CPU and memory config directly from the underlying host
- Effects thread pool size
- OutOfMemoryError

- Mitigated in Java 8u131
- Solved in Java 10

efekahraman.github.io

DockerTest.java

```
public class DockerTest {
  public static void main(String[] args) throws InterruptedException {
    Runtime runtime = Runtime.getRuntime();
    int  cpus = runtime.availableProcessors();
    long mmax = runtime.maxMemory() / 1024 / 1024;

    System.out.println("System properties");
    System.out.println("Cores        : " + cpus);
    System.out.println("Memory (Max): " + mmax);
    while (true) Thread.sleep(1000);
  }
}
```

omega point.

```
$ docker run <id>
System properties Cores : 2
Memory (Max): 241
```

omega point.

```
$ docker run --cpu-shares 512 --memory 512MB
<id>
System properties Cores : 2
Memory (Max): 241
```

omega point.

```
$ docker run \
--cpuset-cpus 0 \
--memory 512MB \
--env JAVA_OPT="-XX:+UnlockExperimentalVMOptions \
               -XX:+UseCGroupMemoryLimitForHeap" \
<id>

System properties
Cores        : 1
Memory (Max): 123
```

omega point.

```
$ docker run --cpu-shares 512 --memory 512MB
<id>
System properties Cores : 1
Memory (Max): 123
```

omega point.

```
$ docker run \
--cpu-shares 512 \
--memory 512MB \
--env JAVA_OPT=-XX:-UseContainerSupport \
<id>

System properties
Cores : 2
Memory (Max): 241
```

omega point.

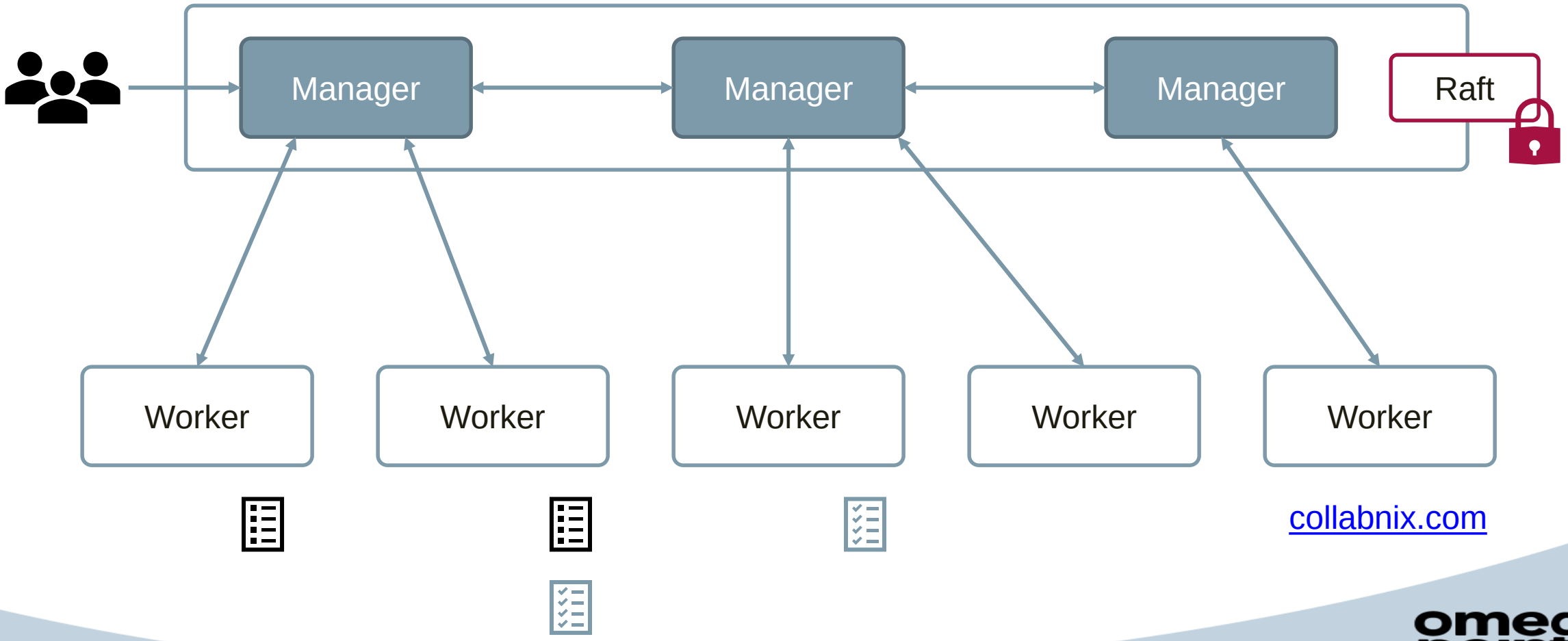
Orchestration

Swarm mode

Key Concepts

- Nodes
  - Manager
  - Worker
- Services
- Stack

Cluster


Manager
Manager
Manager
Raft
Worker
Worker
Worker
Worker
Worker
collabnix.com

Security Concepts

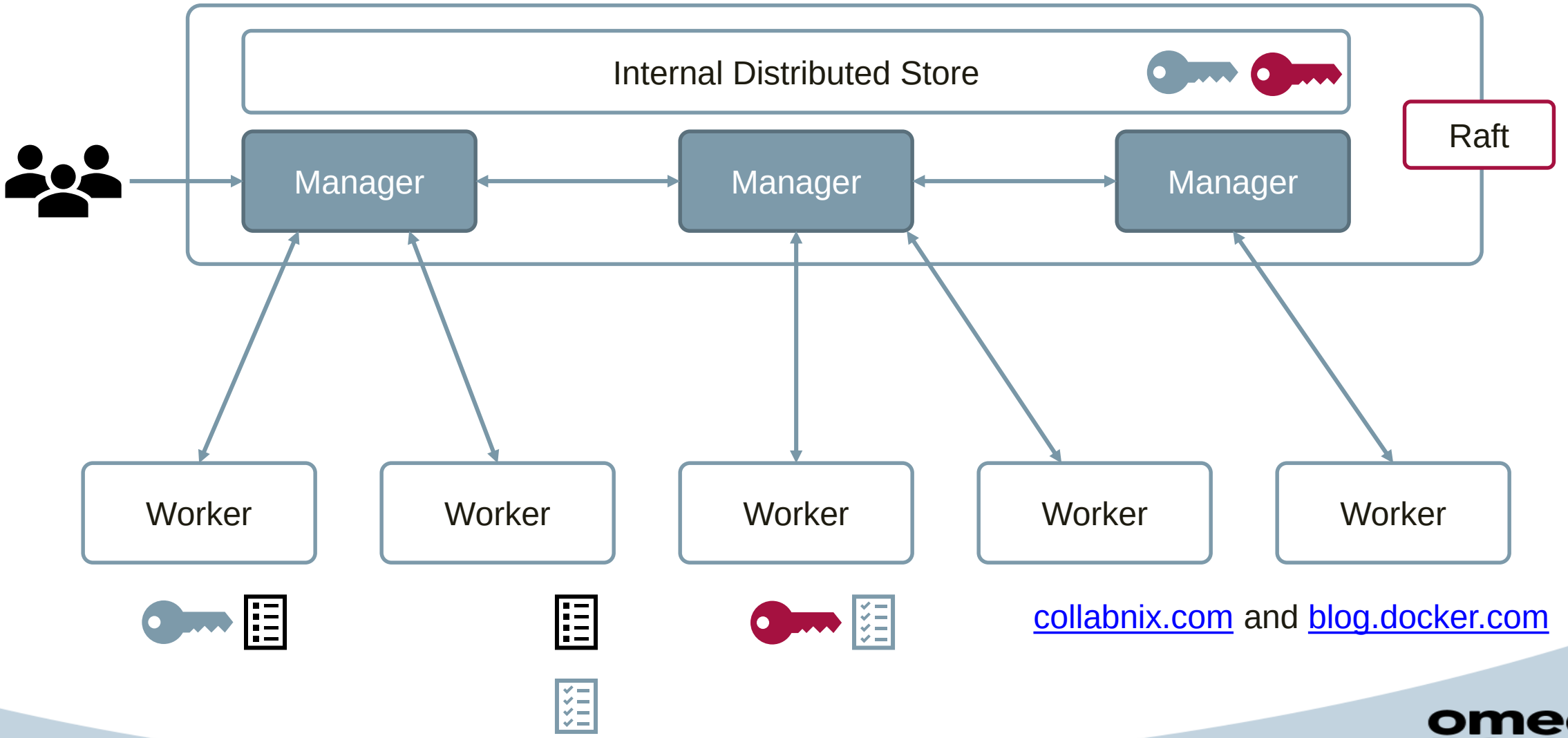
- Encrypted Raft logs
- Encrypted control and management plane traffic with mutual TLS
- HA Scheduling

- (optional) Encryption for application data plane traffic

Docker Secrets

- Encrypted
  - In transit
  - At rest
- Centrally managed
- Mounted in an in-memory filesystem
  - /run/secrets/<secret_name>

Cluster



collabnix.com and blog.docker.com

omega point.

```
$ docker secret create mySecret secretInFile
```

omega point.

```
$ docker service create --secret mySecret …
```

omega point.

```
$ docker create secrets site.key site.key
$ docker create secrets site.crt site.crt
```

omega point.

```
server {
    listen                  443 ssl;
    server_name             localhost;
    ssl_certificate         /run/secrets/site.crt;
    ssl_certificate_key     /run/secrets/site.key;

    location / {
        root   /usr/share/nginx/html;
        index  index.html index.htm;
    }
}
```

omega point.

```
$ docker service create \
      --name nginx \
      --secret site.key \
      --secret site.crt \
       --config source=site.conf, \
                target=/etc/nginx/conf.d/site.conf,\
                mode=0440 \
      --publish published=3000,target=443 \
      nginx:latest
```

omega point.

Resources

- Documentation: https://docs.docker.com
- Docker on Youtube: https://www.youtube.com/channel/UC76AVf2JkrwjxNKMuPpscHQ
- Hands-on Learning: https://training.play-with-docker.com