

# Working with Containers

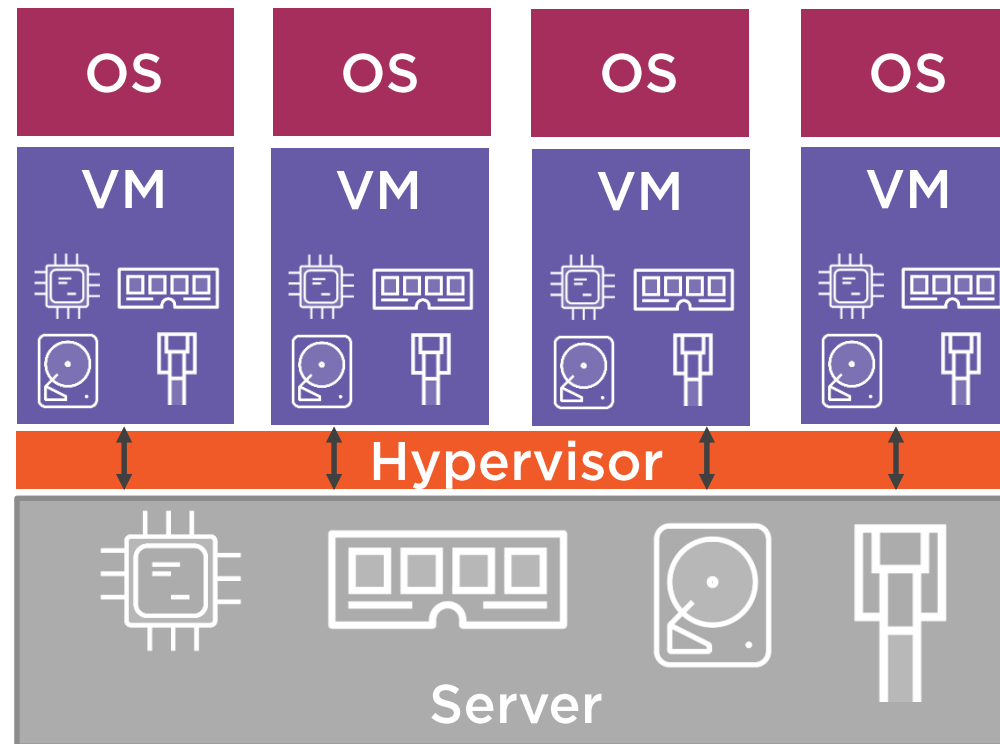
---

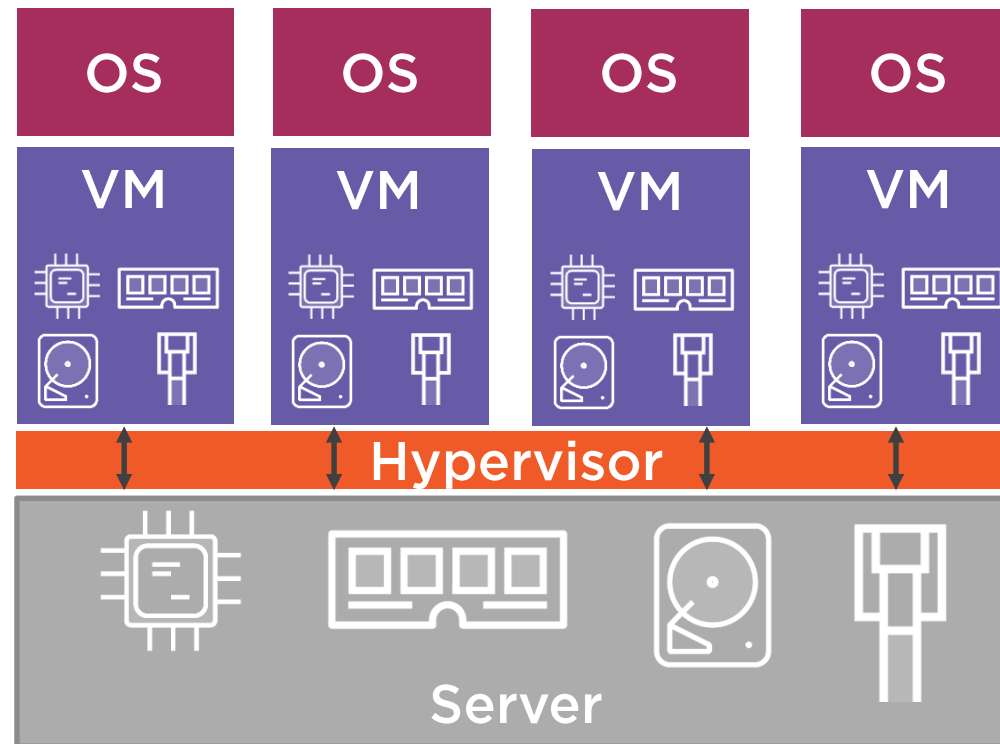


**Nigel Poulton**

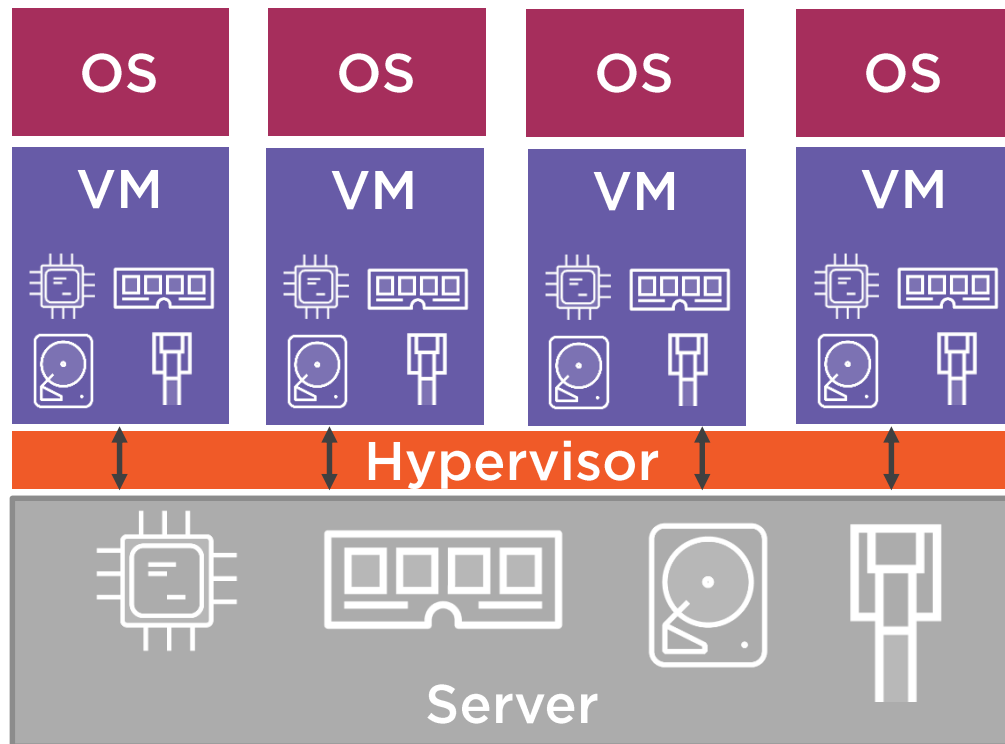
@nigelpoulton [www.nigelpoulton.com](http://www.nigelpoulton.com)



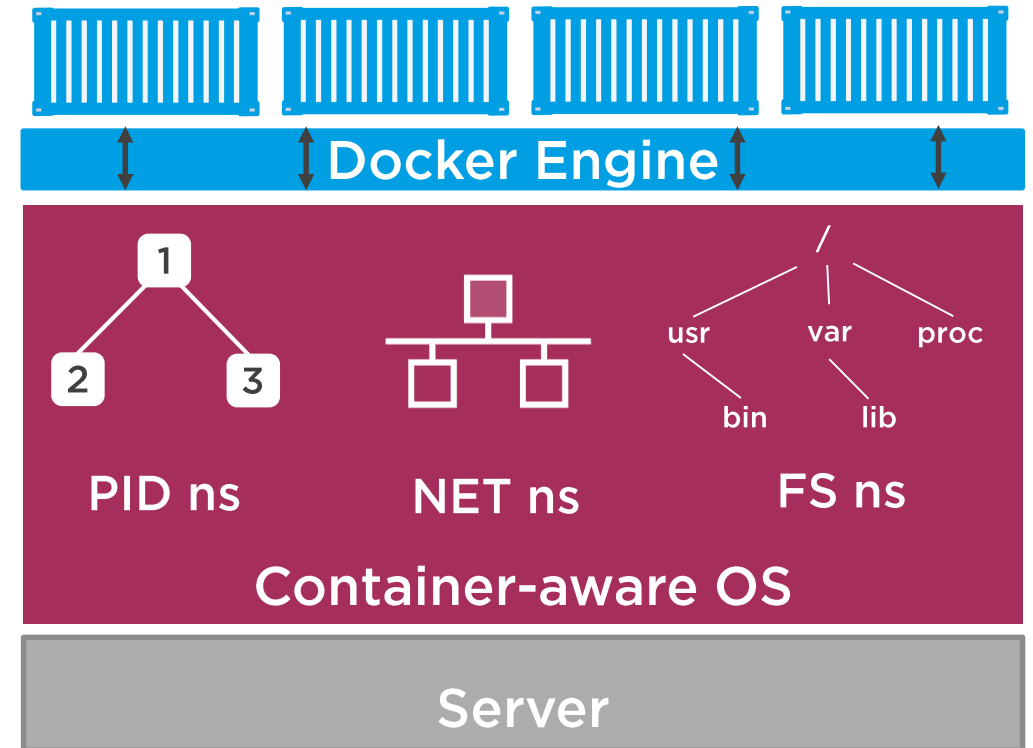




## VM Model



## Container Model



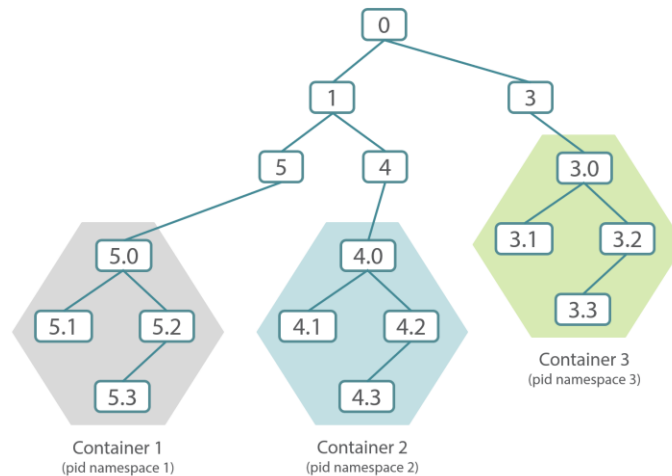
Objects in these diagrams are not drawn to scale



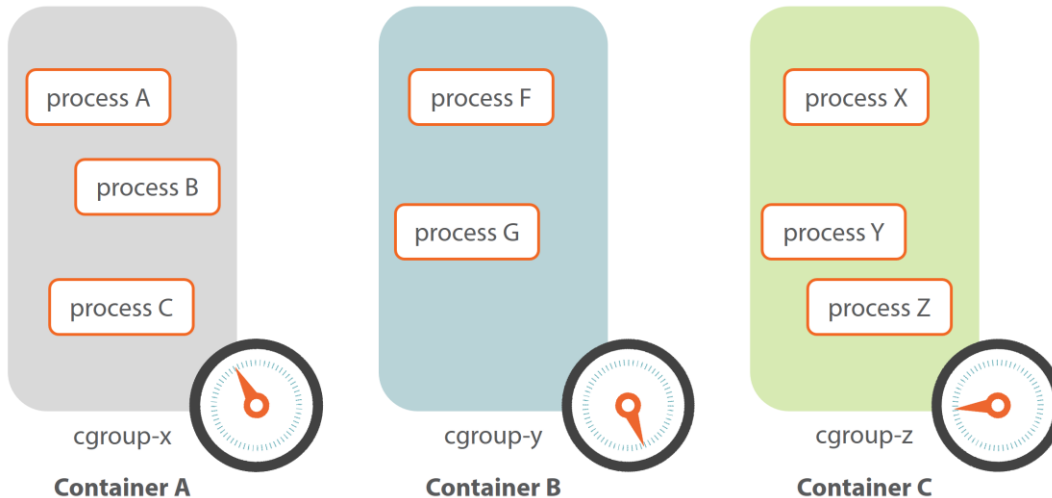
# Docker Deep Dive course for more detail...

## Kernel namespaces

The **pid** Namespace  
The **net** Namespace  
The **mnt** Namespace  
The **user** Namespace



## cgroups



## Capabilities

CAP\_AUDIT\_CONTROL



CAP\_CHOWN



CAP\_DAC\_OVERRIDE



CAP\_KILL



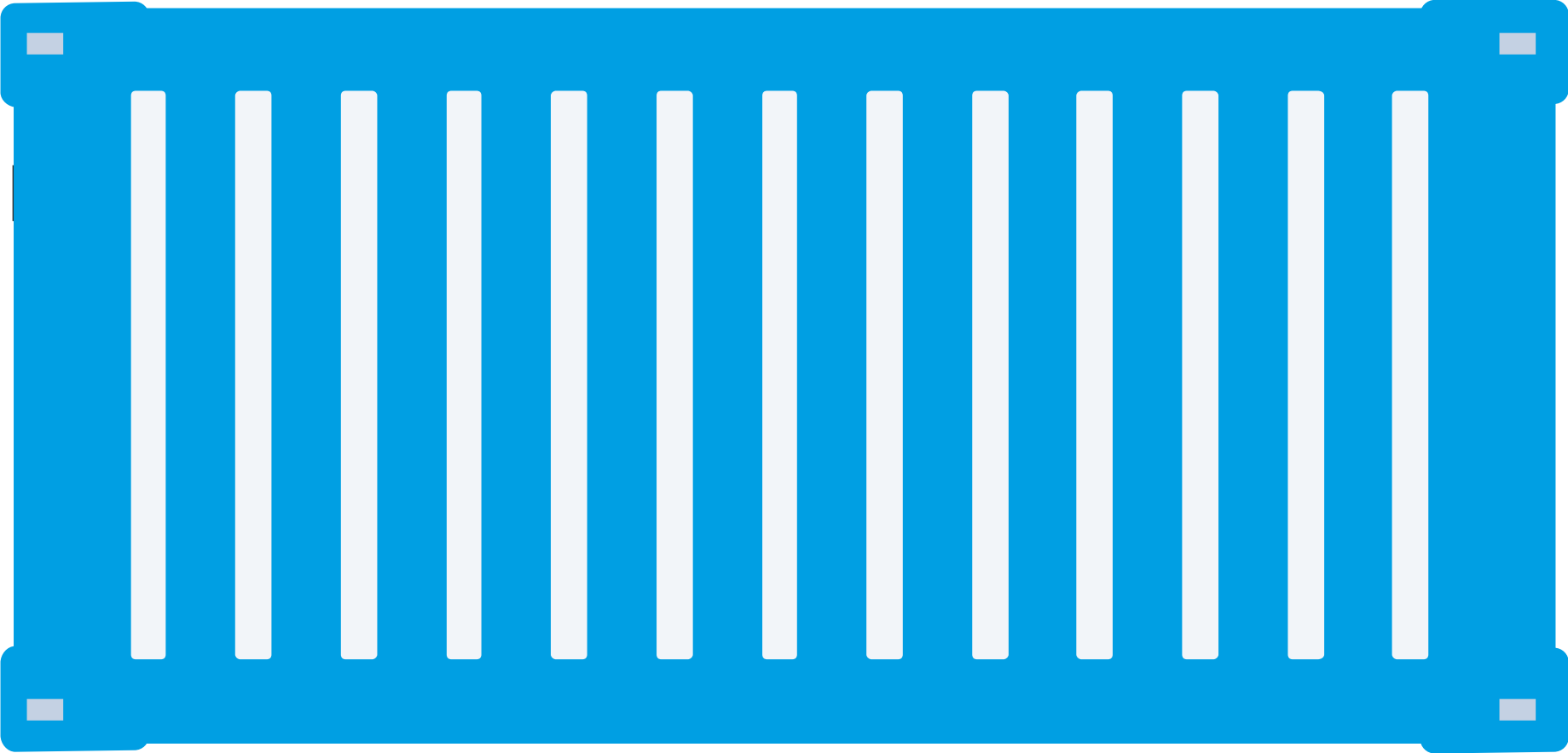
CAP\_NET\_BIND\_SERVICE



CAP\_SETUID



Col



Quick recap...

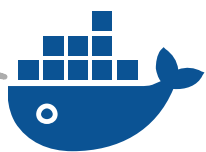




geekchell/hello-world

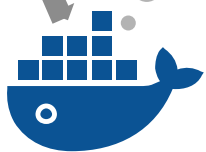


```
$ docker run hello-world
```

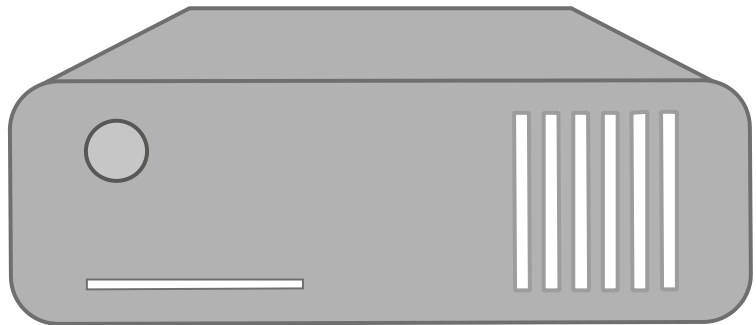


Client


API calls



Daemon



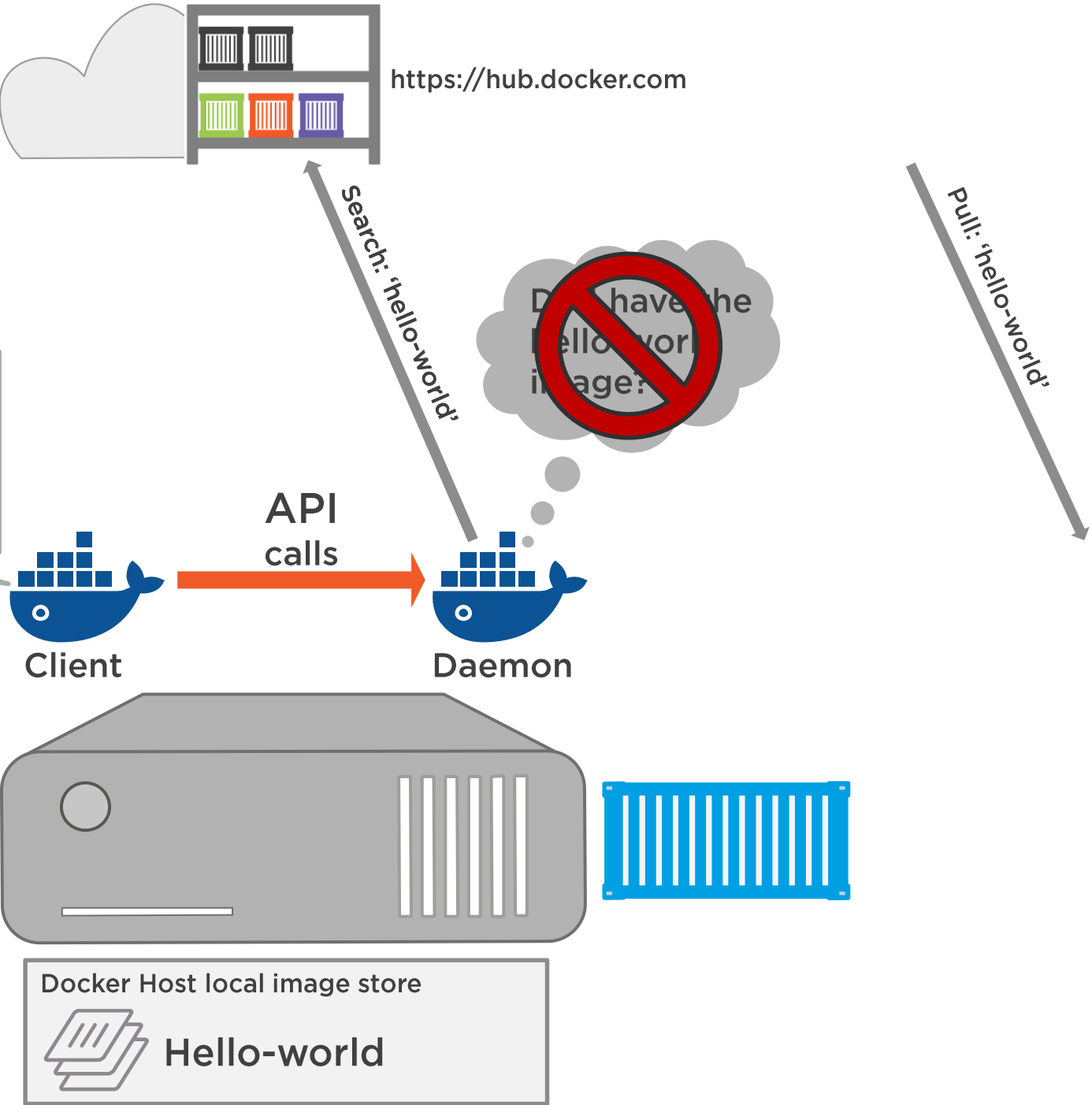
Docker Host local image store

 Hello-world





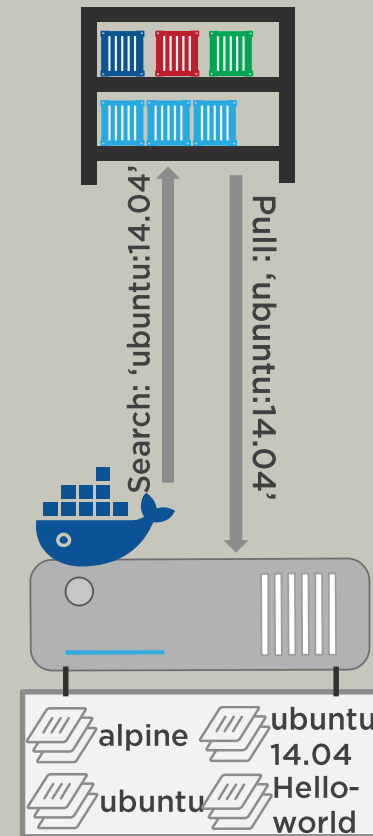
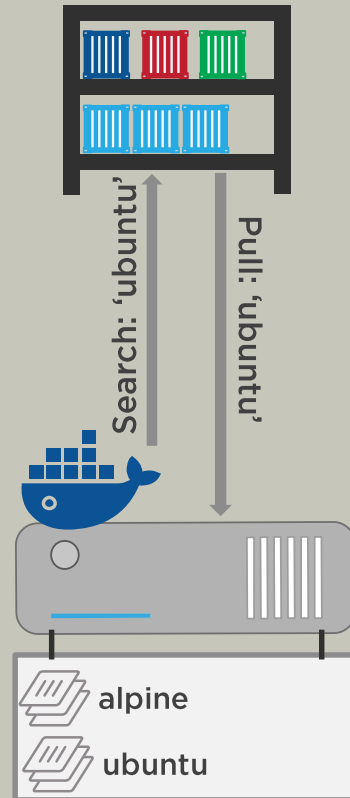
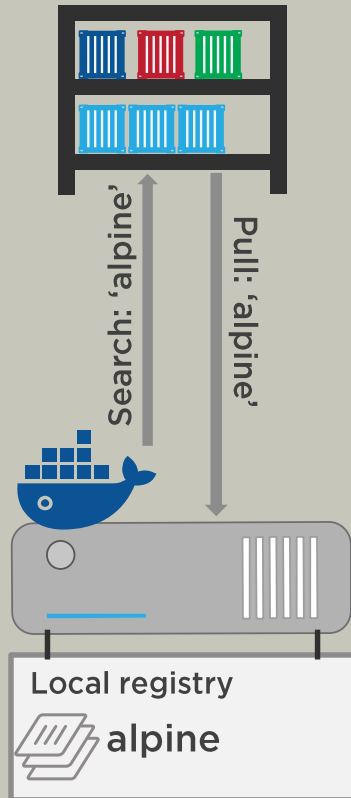
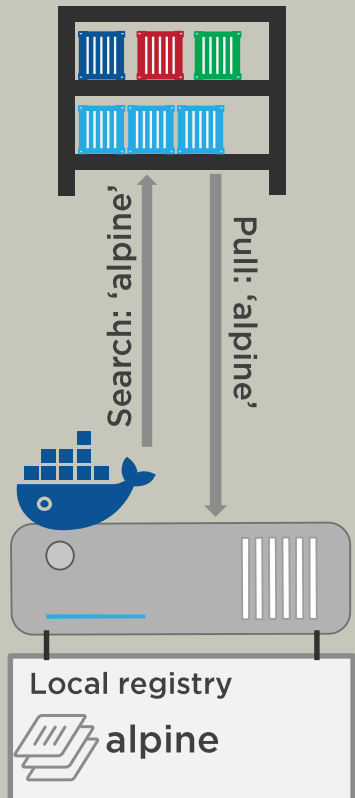
```
$ docker run hello-world
```



Containers ~ Stopped containers  
Images ~ Stopped containers

Containers ~ Running Images





# Docker Deep Dive

By Nigel Poulton

Learn everything you need to know to get yourself up and running

## Course info

Level Intermediate

Rating ★★★★★ (1377)

Duration 5h 38m

Released 28 Jan 2015



# Container Lifecycle



“To persist or not to persist.”

~~William Shakespeare~~

tosh

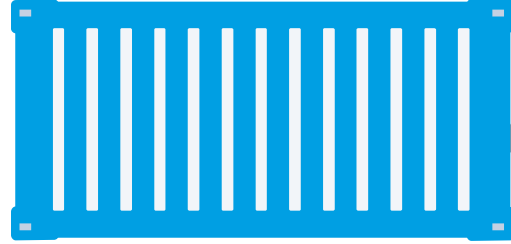
/tɒʃ/

*noun* **BRITISH** *informal*

rubbish; nonsense



RUNNING (UP)



`docker start <container>`

`docker stop <container>`

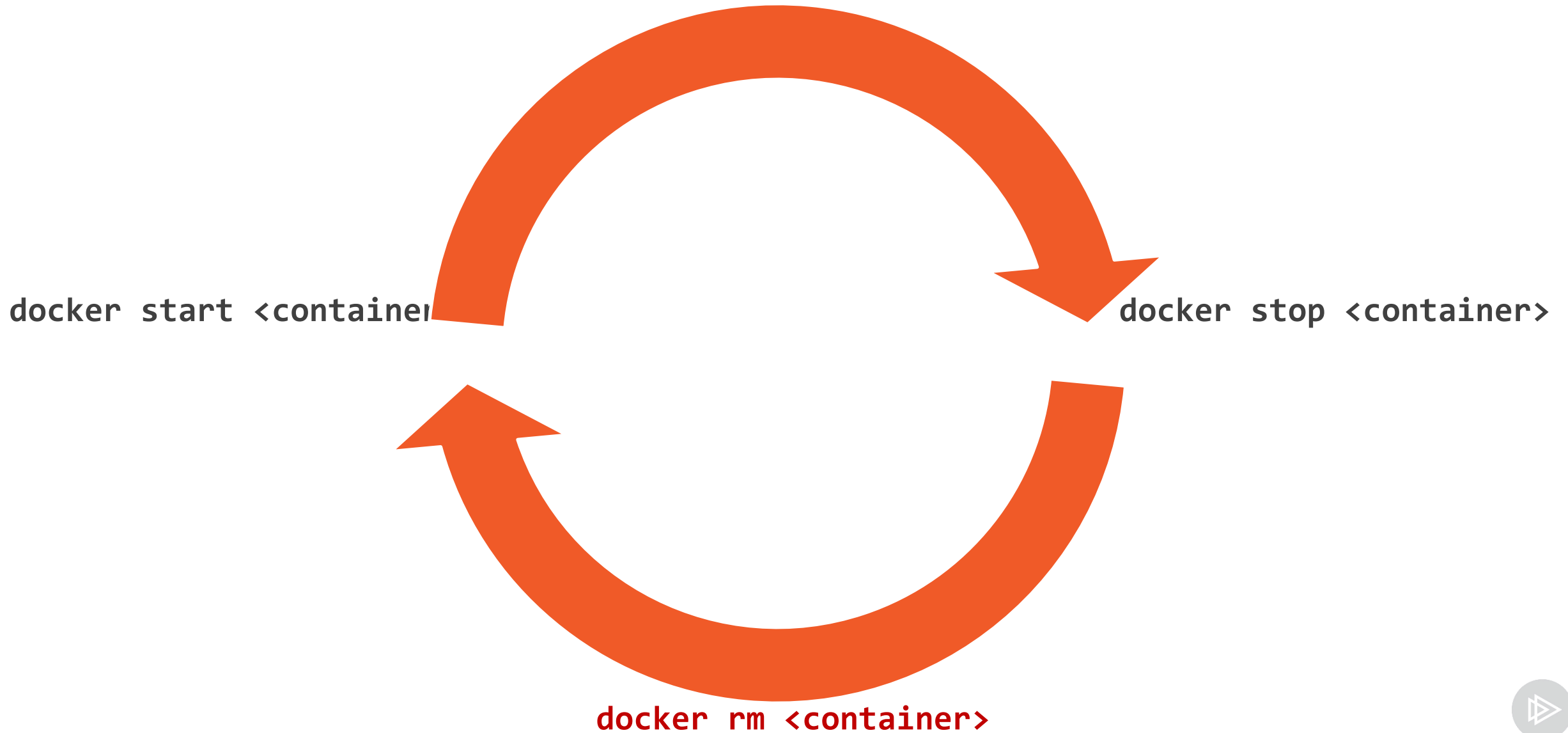


STOPPED  
(EXITED)

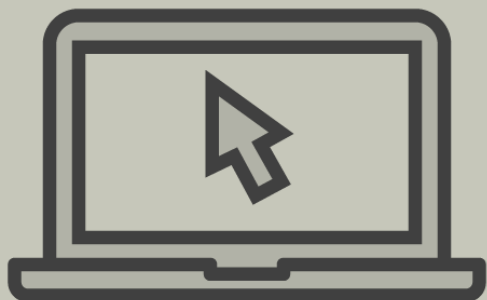
`docker rm <container>`



# Container lifecycle ~ VM lifecycle







`http://public-dns-of-docker-host (:80)`



`:80`



`:8080`



Docker Host



## Top Level Images \*Official\*

(stored in the root of Hub)

- nginx
- busybox
- ubuntu
- redis
- alpine
- ...

## Second Level Images

(stored in their own namespace)

- nigelpoulton/pluralsight-docker-ci
- dockercloud/haproxy
- phusion/baseimage
- mesoscloud/mesos-master
- cockpit/ws



None of the above images and repos are endorsed or recommended by this course. They are just shown as examples of second-level repos



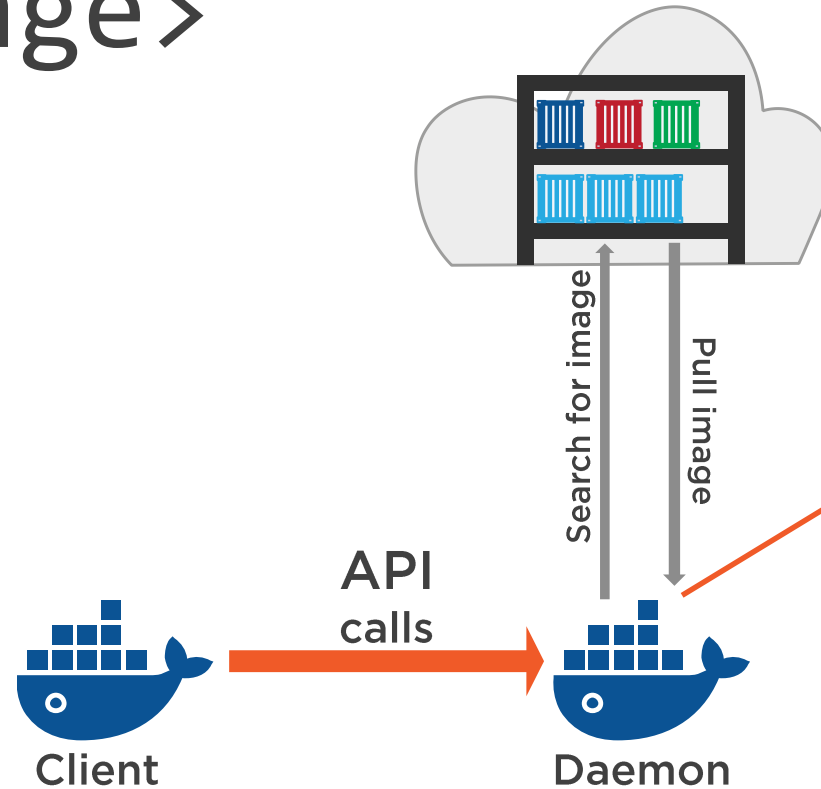
The other processes seen (`ps` and `top`) are temporary processes that Bash created (forked) when I ran those respective commands

# Recap



~stopped container (template)

`docker run <image>`

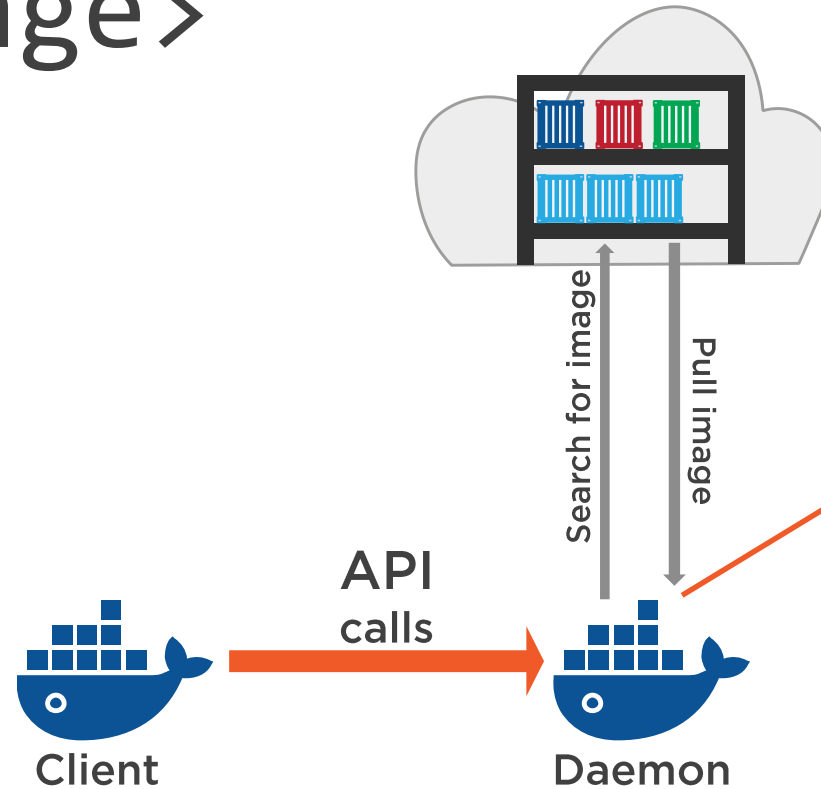


Implements the  
**Docker Remote API**  
[https://docs.docker.com/engine/reference/api/docker\\_remote\\_api/](https://docs.docker.com/engine/reference/api/docker_remote_api/)



~stopped container (template)

`docker run -d <image>`



Implements the  
**Docker Remote API**  
[https://docs.docker.com/engine/reference/api/docker\\_remote\\_api/](https://docs.docker.com/engine/reference/api/docker_remote_api/)

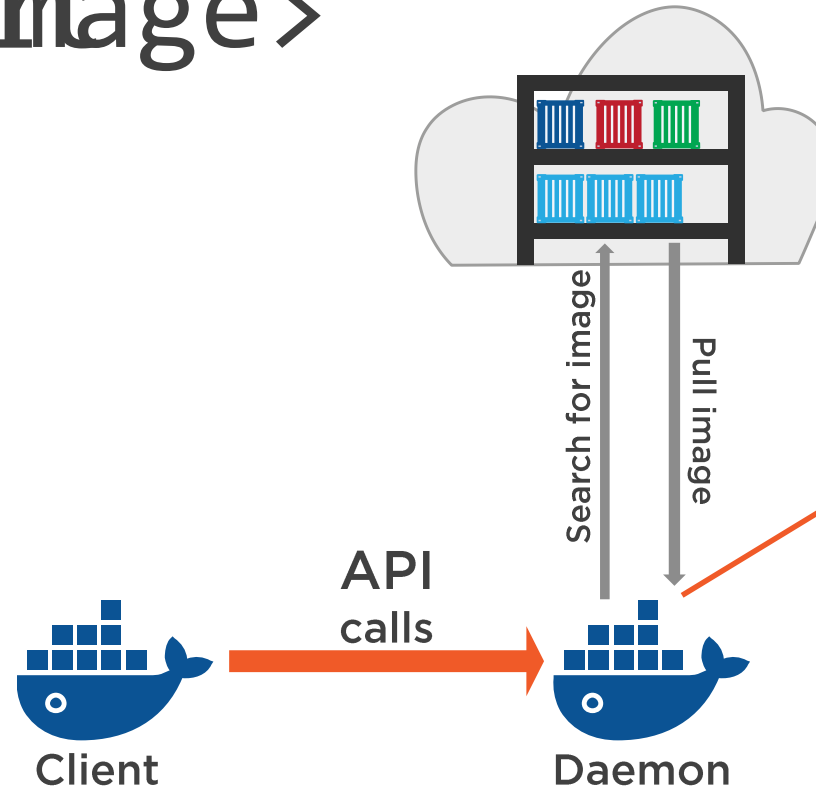


```
docker run -d<image>
```

detached

interactive

~stopped container (template)



Implements the  
**Docker Remote API**  
[https://docs.docker.com/engine/reference/api/docker\\_remote\\_api/](https://docs.docker.com/engine/reference/api/docker_remote_api/)



```
docker pull
docker run
docker images
docker rmi
```

Starts a new container

Copies images to the Docker Host

Lists images on the Docker Host

Removes images from the Docker Host

```
docker ps
docker stop
docker rm
```

Lists running containers

Stops running containers

Removes (deletes) stopped containers





Coming up next....

Native Docker clustering and orchestration

