

Introducing Containers with Docker

WW Developer Advocate Team

IBM Developer

What are containers?

(and how do they differ from Docker?)

Introducing containers and Docker

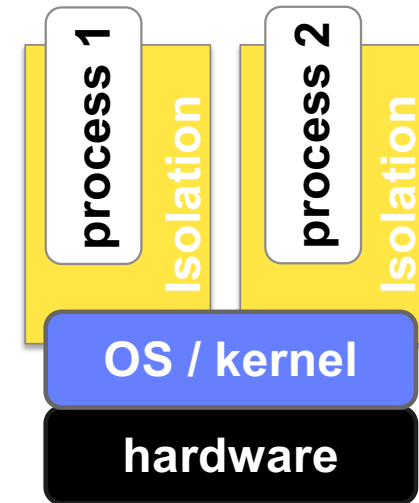
Containers – not a new idea

- chroot ('80s) process spawned in isolated file space
- FreeBSD jails
- OS-level virtualization (user-mode-linux, virtuozzo)
- Solaris Containers
- Linux Containers (LXC)
- Cloud Foundry (Warden, Garden)

More efficient than VMs but less mindshare...

Docker – ecosystem approach transformed perception

- Building application-centric containers
- Mechanism for sharing images (Docker Registry)
- Open-source enabled



What are Containers?

Similar to VMs but managed at the **process level**

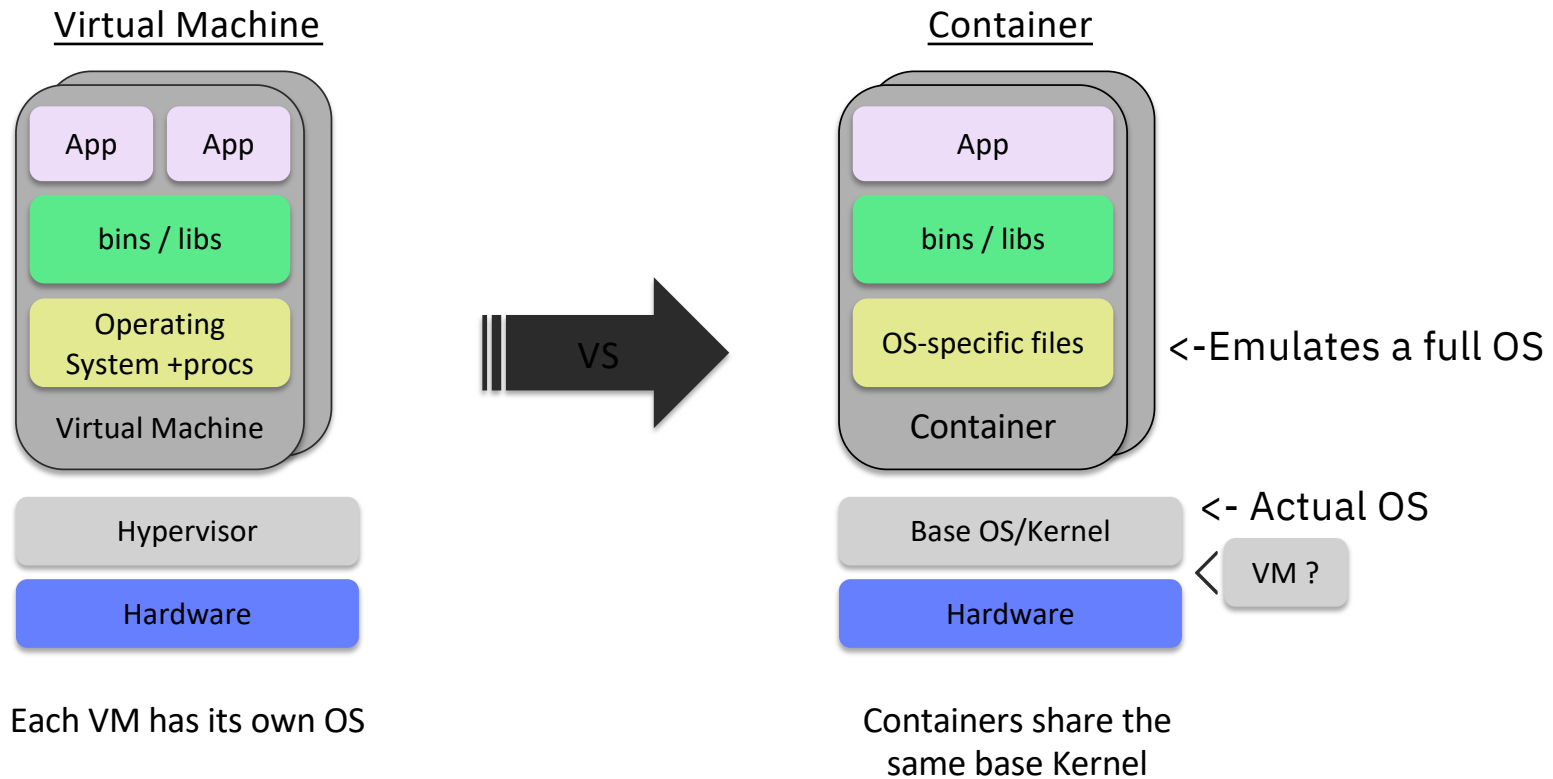
"VM-like" isolated achieved by set of "**namespaces**" (isolated view)

- PID –isolated view of process IDs
- USER- user and group IDs
- UTS - hostname and domain name
- NS - mount points
- NET - Network devices, stacks, ports
- IPC - inter-process communications, message queues

cgroups - controls limits and monitoring of resources

The key statement: **A container is a process(es) running in isolation**

VM vs Container



What is Docker?

Containers is the technology,

Docker is the **tooling** around containers

Without Docker, containers would be **unusable** (for most people)

Docker **simplified** container technology to enable it for the masses

Added value: Lifecycle support, setup file system, etc

For extra confusion: Docker is also a company, as well as Docker the technology...

Follow Along...

<https://labs.play-with-docker.com>

Our First Container

```
$ docker run ubuntu echo Hello World  
Hello World
```

What happened?

- Docker created a directory with a "ubuntu" filesystem (image)
- Docker created a new set of namespaces
- Ran a new process: echo Hello World
 - Using those namespaces to isolate it from other processes
 - Using that new directory as the "root" of the filesystem (chroot)
- That's it!
 - Notice as a user I never installed "ubuntu"
- Run it again - notice how quickly it ran

ssh-ing into a container - fake it...

```
$ docker run -ti ubuntu bash
```

```
root@62deec4411da:/# pwd
```

```
/
```

```
root@62deec4411da:/# exit
```

```
$
```

Now the process is "bash" instead of "echo"

But its still just a process

Look around, mess around, its totally isolated

- rm /etc/passwd – no worries!
- MAKE SURE YOU'RE IN A CONTAINER!

A look under the covers

```
$ docker run ubuntu ps -ef
```

| UID | PID | PPID | C | STIME | TTY | TIME | CMD |
|------|-----|------|---|-------|-----|----------|--------|
| root | 1 | 0 | 0 | 14:33 | ? | 00:00:00 | ps -ef |

Things to notice with these examples

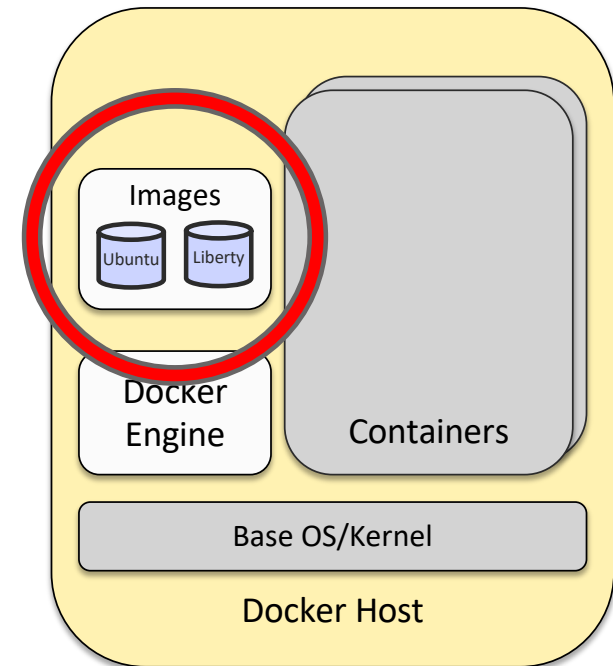
- Each container only sees its own process(es)
- Each container only sees its own filesystem
- Running as "root"
- Running as PID 1

Docker Images

Tar file containing a container's filesystem + metadata

For sharing and redistribution

– Global/public registry for sharing: DockerHub



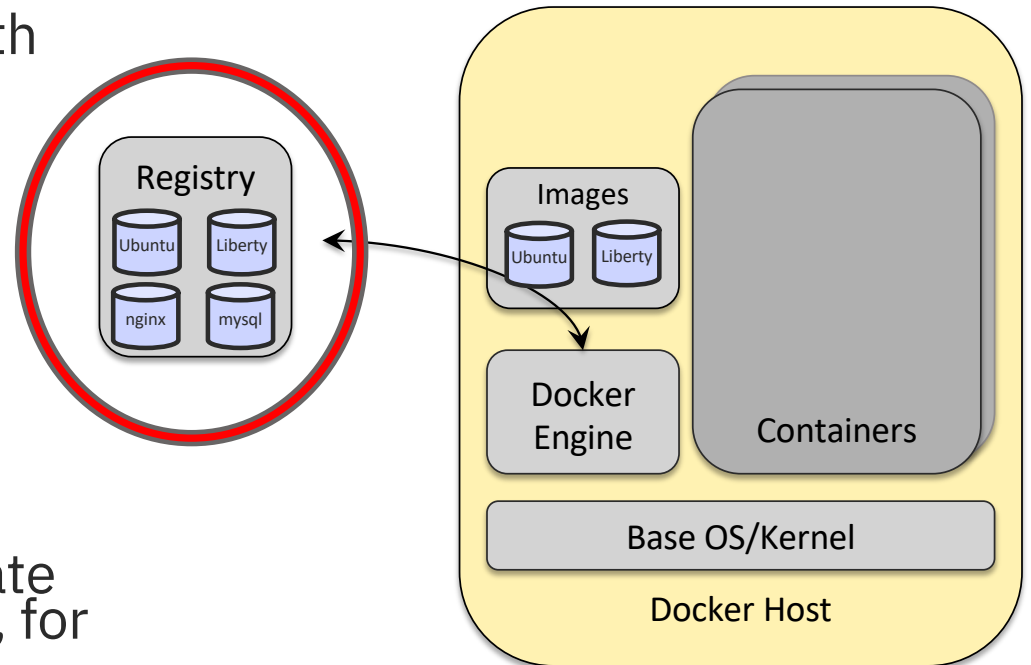
Docker Registry

The central place to share images with friends! (or coworkers)

DockerHub - <http://hub.docker.com>

- Public registry of Docker Images
- Also useful to find prebuilt images for web servers, databases, etc

Enterprises will want to find a private registry to use for internal projects, for consistency and security



Build your own image with a Dockerfile

Step 1) Create Dockerfile to script how you want the image to be built

```
FROM java:8 # This might be an ubuntu or...  
COPY *.jar app.jar  
CMD java -jar app.jar
```

Step 2) **docker build** to build an image

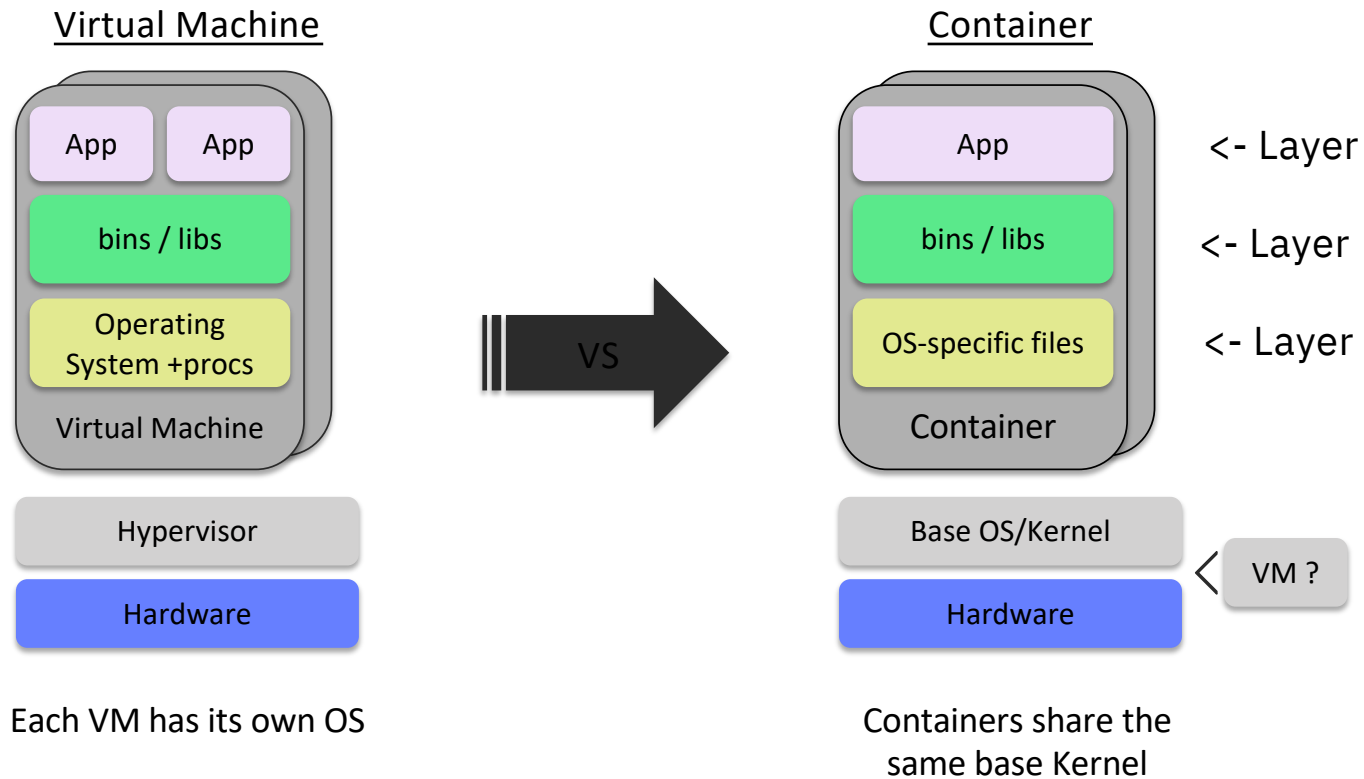
Step 3) **docker push** to push to registry

Step 4) From another location, **docker pull** to download an image

Docker special sauce: Layers

But first, let's compare VMs and Containers one more time...

VM vs Container: Notice the layers!

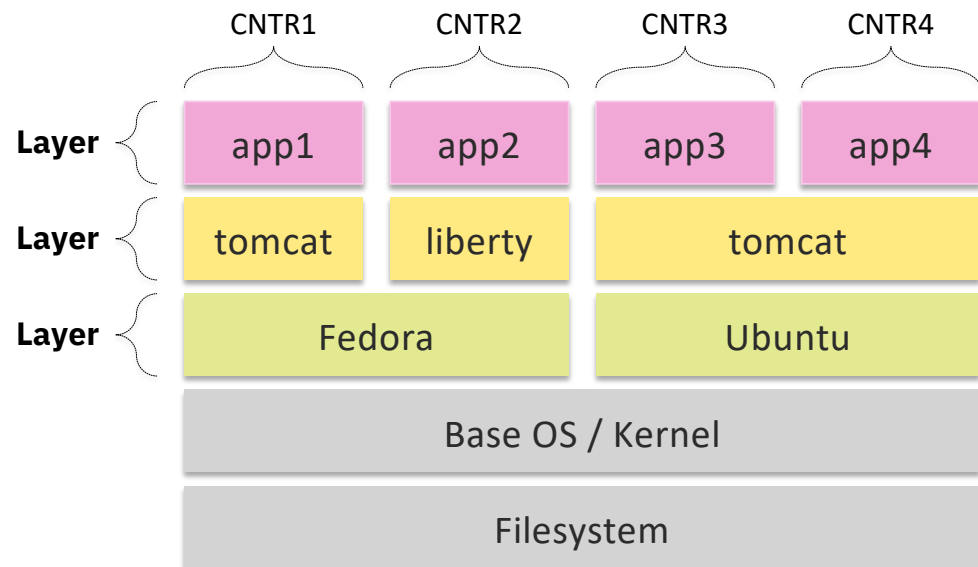


Shared / Layered / Union Filesystems

- Docker re-uses common layers between containers and images
- A single writeable layer is added on the top every time a new layer is created
- Layers are “smushed” with **union file system** (think transparencies on a projector)
- Files are copied up when writes need to be made (copy-on-write)

Bottom Line

- More containers per host
- Faster downloads and uploads
- Faster container startups



Summary

Docker is just a tool to manage containers

- Key concepts: Containers, Engine, Images, Registry

Docker value-add:

- An excellent User Experience
- Image Layers
- Easily shared images - DockerHub

Why? When compared to VMs:

- Better resource utilization - CPU, Memory, Disk
- Faster start-up times
- Easier tooling/scripting

Quiz!

What's the difference between a container and an image?

Answer:

- An image is a tar of a filesystem
- A container is a filesystem + a set of processing running in isolation

In a Traditional Deployment...

Are you testing these on every commit?

Code (packages archive) ✓

App server ✗

Runtime versions ✗

System libraries and versions ✗

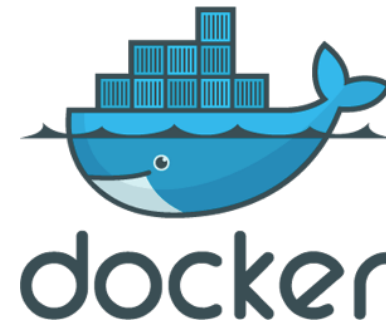
Container = Code + Dependencies

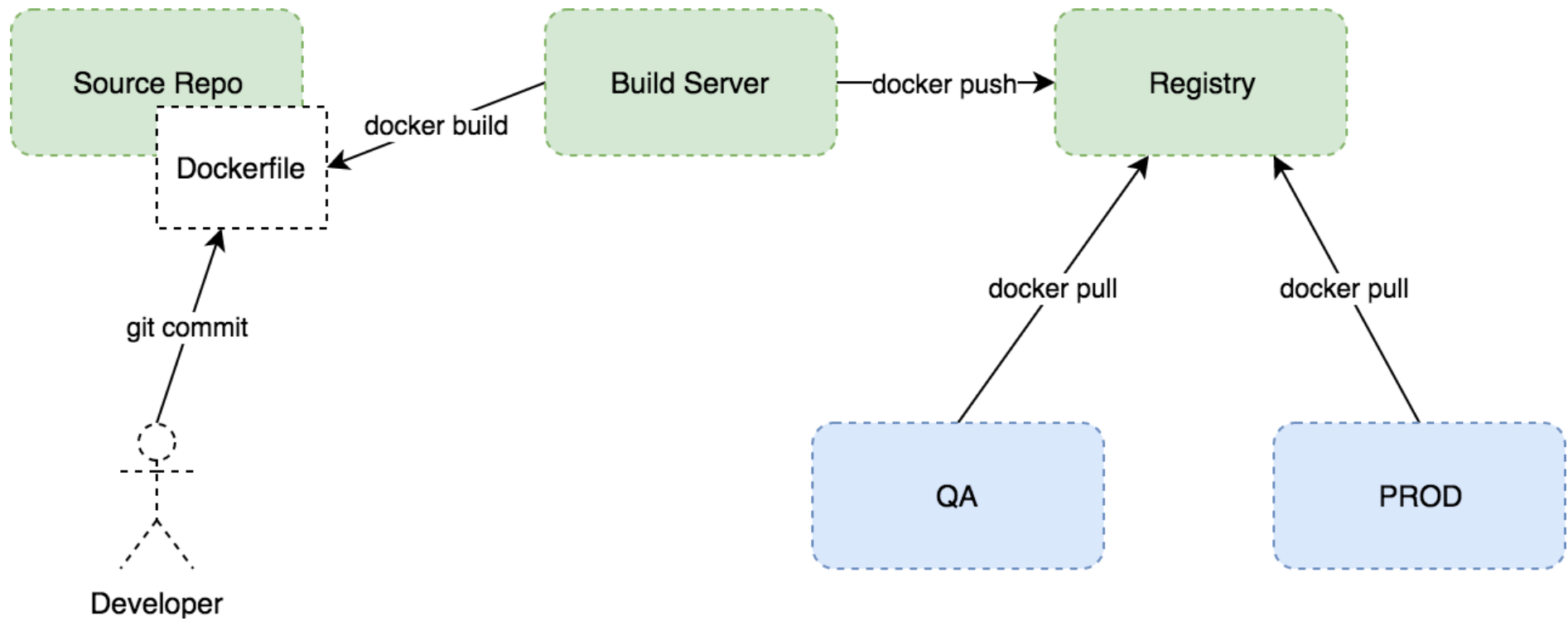
Code (packages archive)

App server

Runtime versions

System libraries and versions





Docker for Operations



Lab- build your own image

<https://github.com/IBM/docker101>

Questions?