MoodleMoot UK & Ireland 2017 - London 10th - 12th April

Containers vs Virtualization

Presented by **Paul Greidanus**
Site Reliability Engineer

Who Am I?

Paul Greidanus
Email: paul@moodle.com
Twitter: @paulgreidanus - But there's nothing there

DevOps for MoodleCloud and Moodle ICT

Built the current Kubernetes based Moodle Infrastructure

Brief history of computers

Mainframes: Batch processing, time sharing and virtual machines
Really big computer

Servers: Racks of smaller servers doing tasks

Virtualization with Vmware/KVM/Xen/Hyper-V:
Less servers, bigger servers again

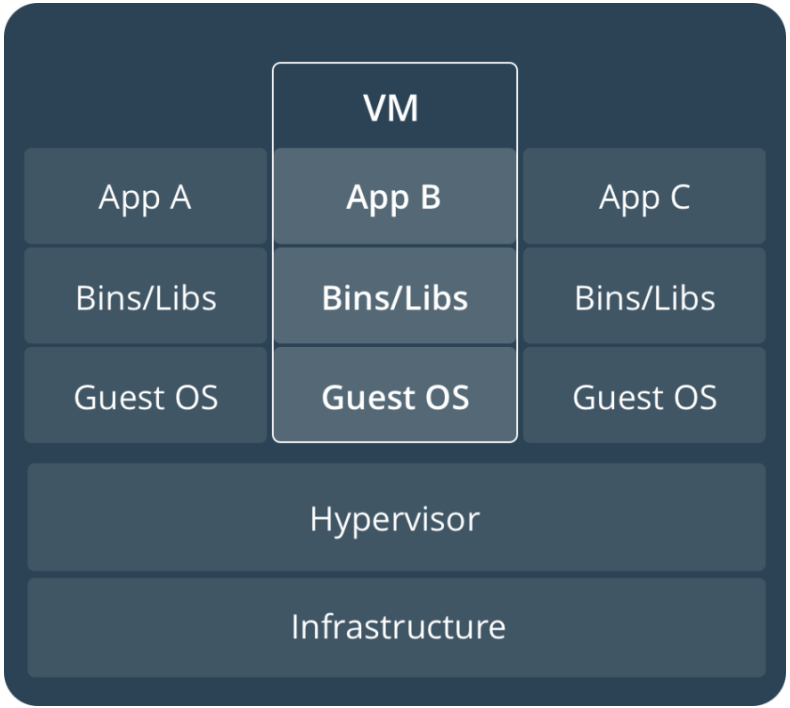
Containers:
Shared kernel with security and resource control between containers
Single application per container and microservice architectures

Why Virtualize?

Computers are powerful, too powerful for many applications to completely utilize

Virtualization allows higher system utilization by allowing sharing of server resources between applications

Virtualization allows for separation of security by keeping applications isolated.



https://www.docker.com/what-container

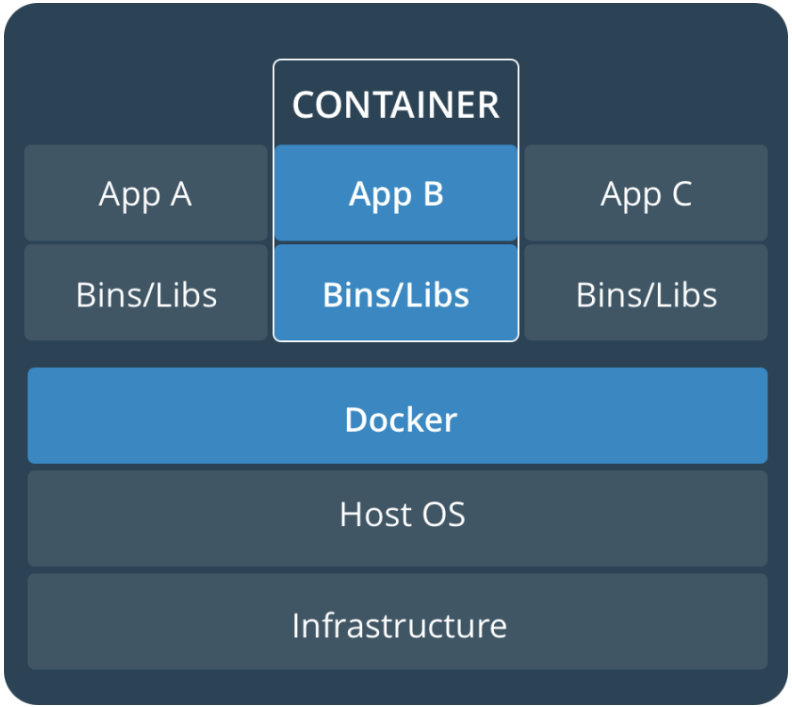
Why Containers?

Containers give you many of the advantages of Virtualization without the costs

Easy to move between platforms

Containers give you the security and isolation of VMs, without paying the performance penalties

Standardized images, “DevOps” workflows

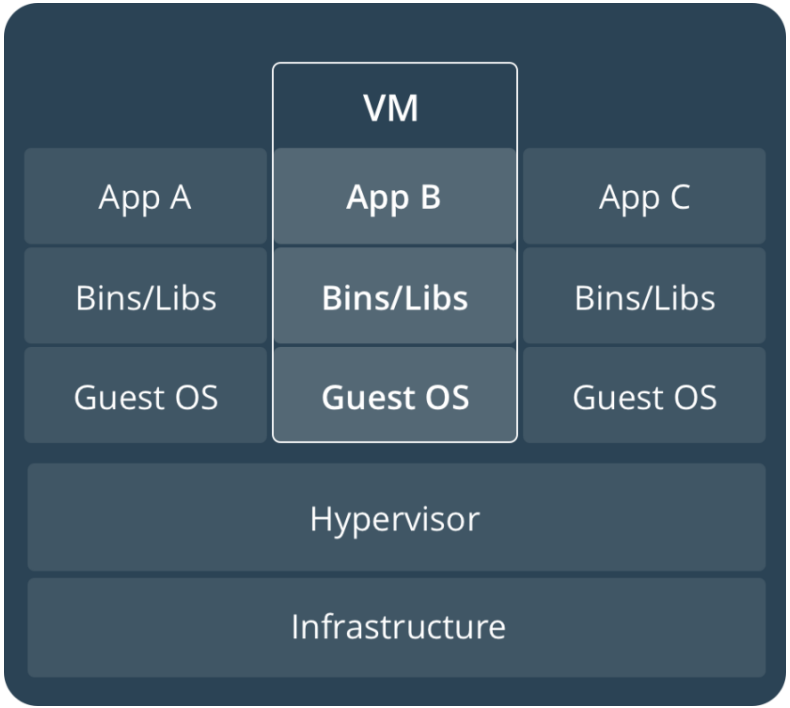


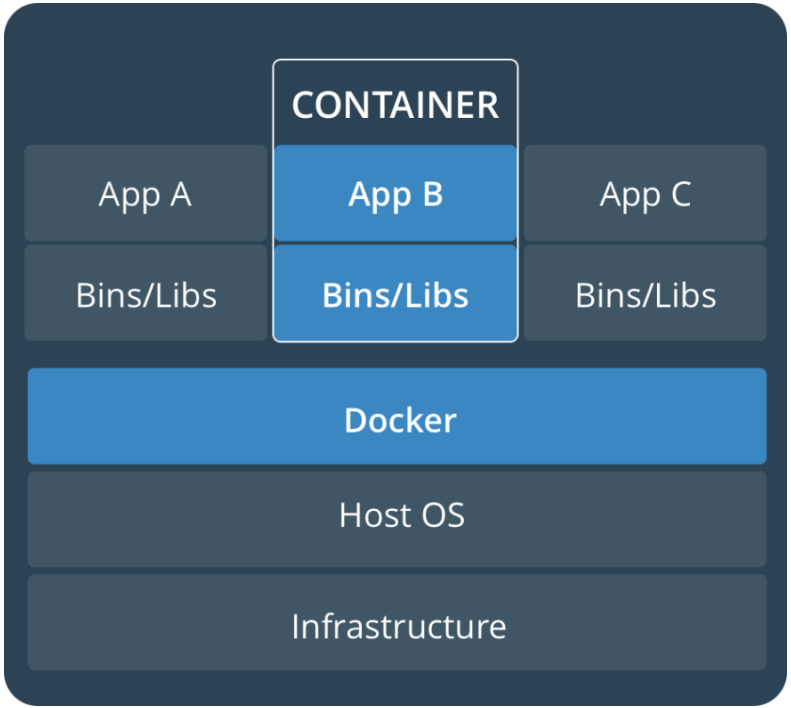
https://www.docker.com/what-container

The Why Nots

Containers	Virtualization
Security is not as good as Virtual or Bare Metal	Hypervisor and Kernel overhead costs performance
Complex configuration	Management of individual VMs
Ephemeral (Cows)	Managed System (Pets)

“Our results show that containers result in equal or better performance than VM in almost all cases.”
An Updated Performance Comparison of Virtual Machines and Linux Containers - IBM Research





https://www.docker.com/what-container

Docker 101

https://docs.docker.com/engine/installation/

Dockerfile for simple webserver

```
FROM httpd:alpine
COPY index.html /usr/local/apache2/htdocs/index.html
 EXPOSE 80
```

Docker 101

```
-> docker build .
Sending build context to Docker daemon 3.072 kB
Step 1/3 : FROM httpd:alpine
 ---> 3a7965c78d17
Step 2/3 : COPY index.html /usr/local/apache2/htdocs/index.html
 ---> c5cf8779cf32
Removing intermediate container 8a04781aa49e
Step 3/3 : EXPOSE 80
 ---> Running in fffccc41af4d
 ---> e5d7085ab6b2
Removing intermediate container fffccc41af4d
Successfully built e5d7085ab6b2
```

Docker 101

```
-> docker run --publish 8080:80 --detach e5d7085ab6b2

-> docker ps

CONTAINER ID        IMAGE               COMMAND
94eb38b79529        e5d7085ab6b2        "httpd-foreground"
CREATED             STATUS              PORTS
24 seconds ago      Up 23 seconds       0.0.0.0:8080->80/tcp
NAMES
affectionate_mclean
```

Orchestration Frameworks

Now that you have built your containers, what next?

Production Containers require orchestration

Orchestration frameworks are varied in what they provide, but generally:

- Maintain running containers
- Provide networking layer
- Scale containers
- Ingress network

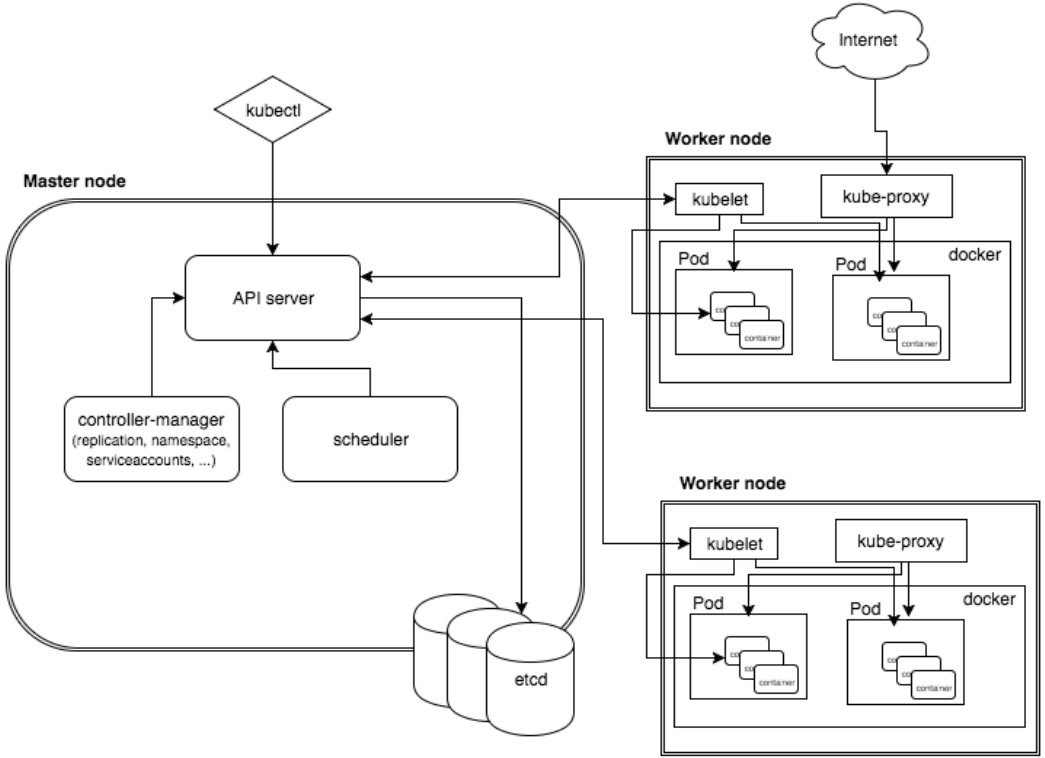
Docker Swarm

Kubernetes / Google Container Engine (GKE)

Mesosphere Marathon

Amazon ECS

Azure Container Services



https://x-team.com/blog/introduction-kubernetes-architecture/

For more information contact:

paul@moodle.com
moodle.com