Docker, Kubernetes and More

John Zaccone

john.zaccone@ibm.com

IBM

CODE

Introduction to Containers and Docker

What are Containers?

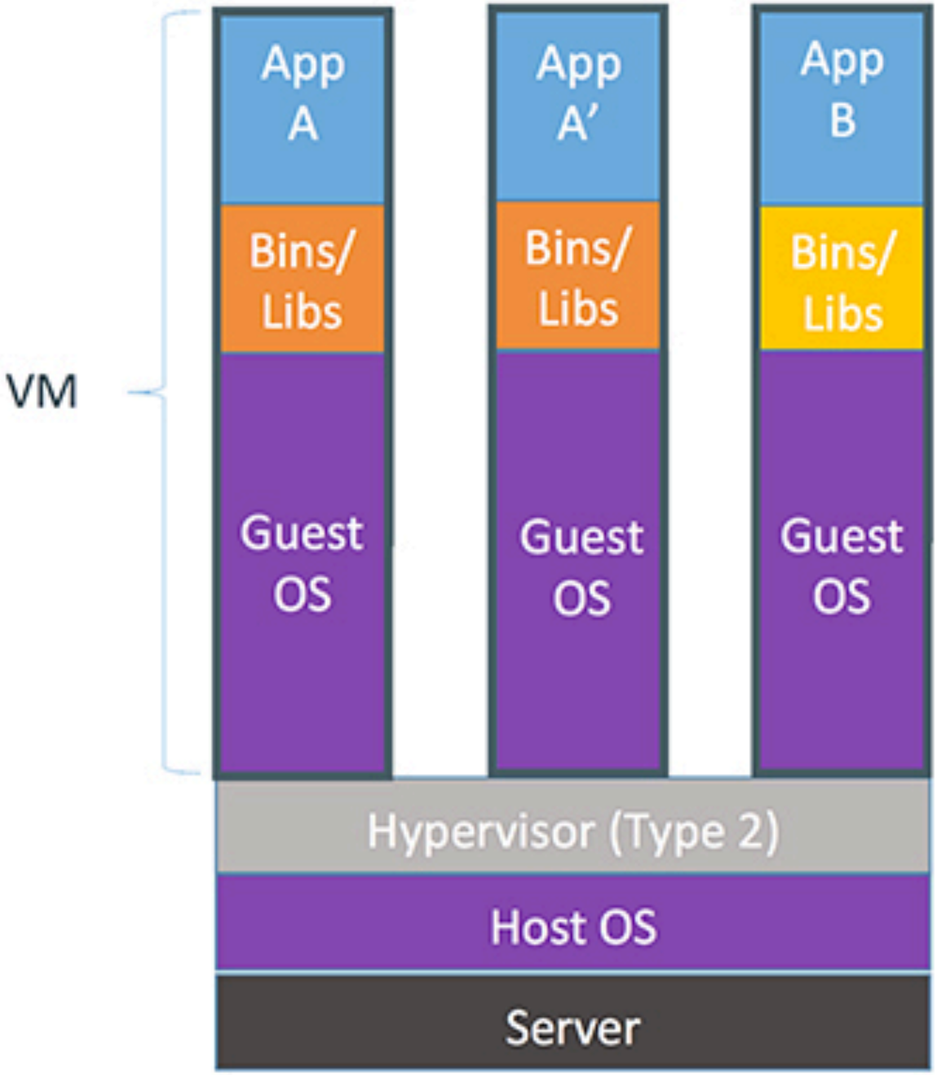
- A group of processes run in isolation
  - Similar to VMs but managed at the process level

- Each container has its own set of "namespaces" (isolated view)
  - **PID** - 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

What is Docker?

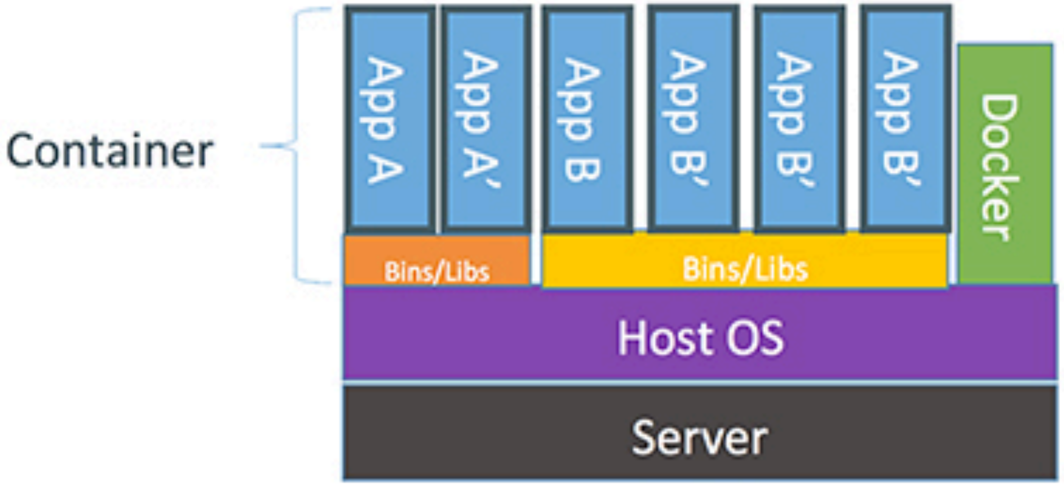
- Containers is the technology, Docker is the tooling around containers
- Without Docker, containers would be unusable (for most people)
- Docker simplified container technology to enable it for the masses
- Added value: Lifecycle support, setup file system, etc

- For extra confusion: Docker is also a company, which is different then Docker the technology…



Containers are isolated, but share OS and, where appropriate, bins/libraries

…result is significantly faster deployment, much less overhead, easier migration, faster restart



Playground

- http://play-with-docker.com

Running our first container

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

- What happened?
  - Docker pulled the “ubuntu” image from Dockerhub
  - 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

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

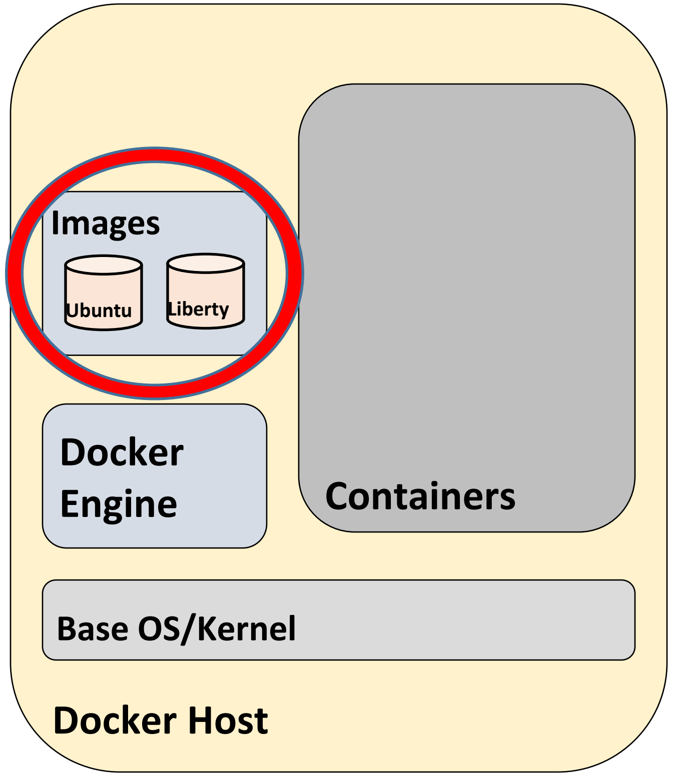
ssh-ing into a container - fake it…

```
$ docker run -ti ubuntu bash
root@62deec4411da:/# pwd
/
root@62deec4411da:/# exit
$
```

- bash is just a process that we run in container namespaces
- No need for ssh server
- Can “enter” namespaces retroactively with `docker exec`

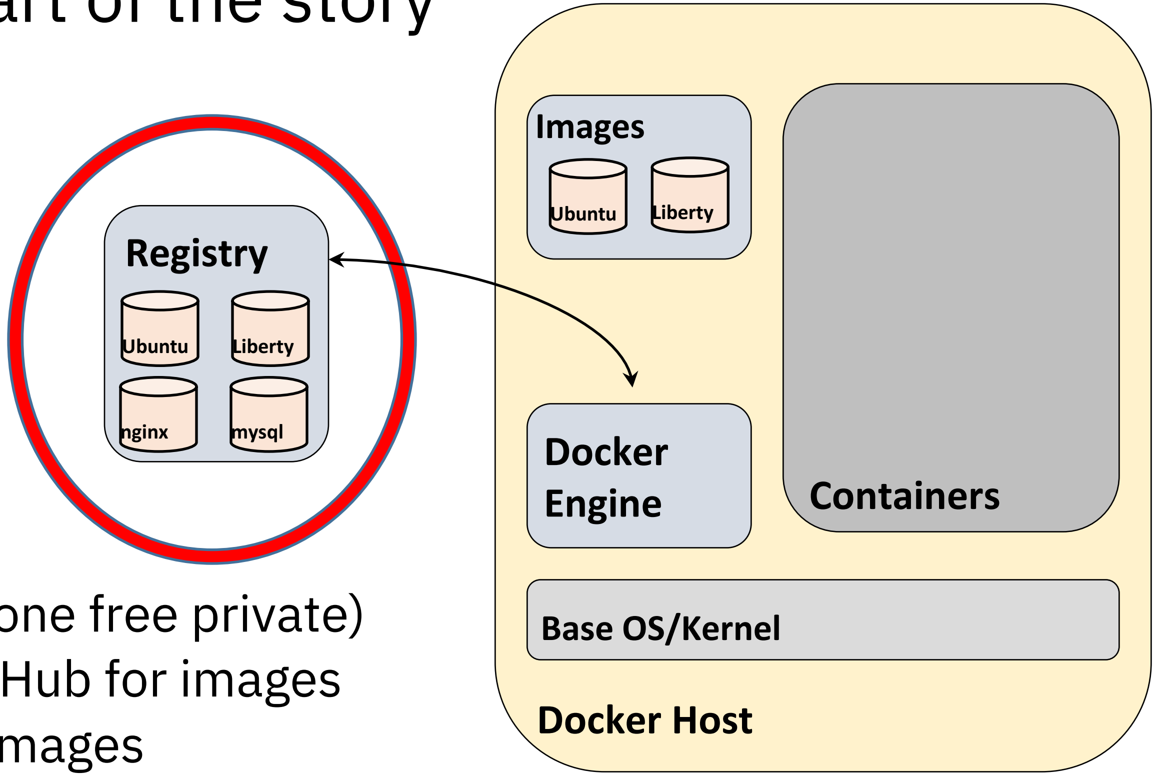
What is a Docker Image?

- Tar file containing a container's filesystem + metadata
- For sharing and redistribution
  - Global/public registry for sharing: DockerHub



Docker Registry

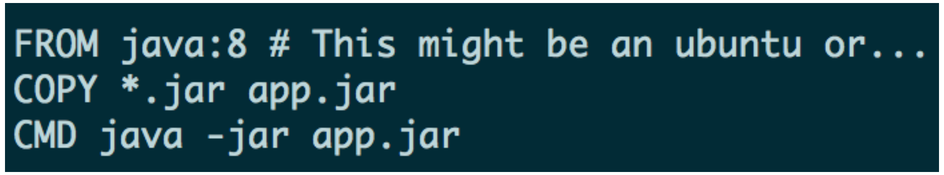
- Creating and using images is only part of the story
- Sharing them is the other



- DockerHub - http://hub.docker.com
  - Public registry of Docker Images
  - Hosted by Docker Inc.
  - Free for public images, pay for private ones (one free private)
  - By default docker engines will look in DockerHub for images
  - Web interface for searching, descriptions of images

Build your own image!

- 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) $$$$$$

Lab

- https://github.com/IBM/intro-to-docker-lab
- Go to “Lab 2: Adding Value with Custom Docker Images”

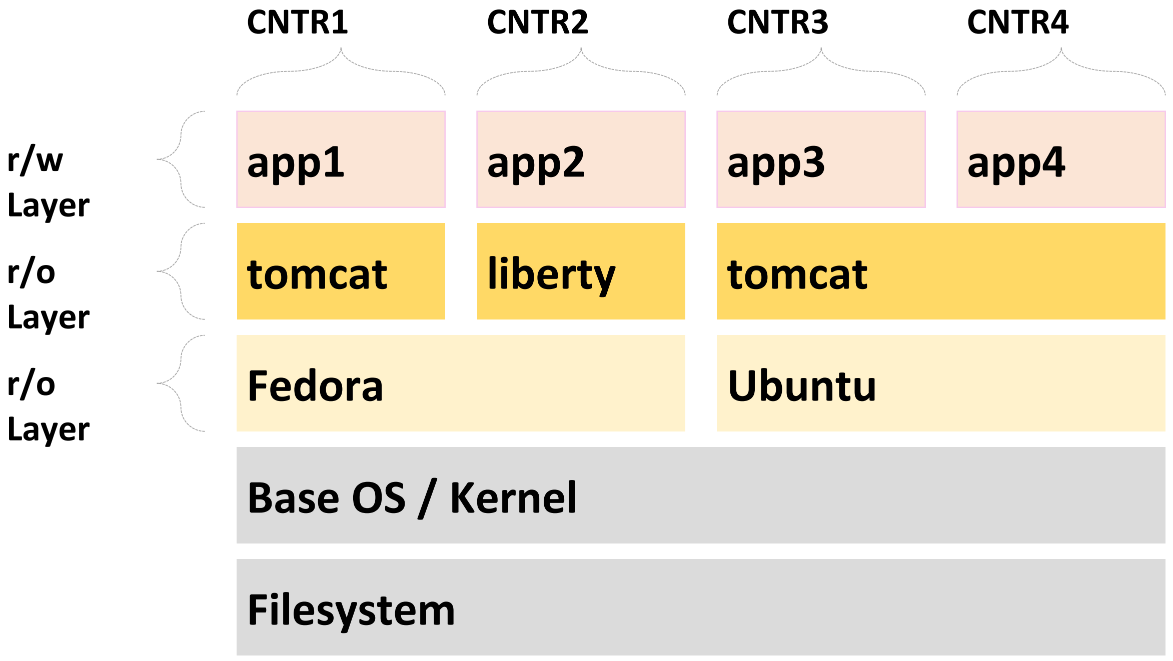
Dockerfile Instructions

- What are some of the other instructions?
  - RUN
  - HEALTHCHECK
  - COPY/ADD
  - CMD & ENTRYPOINT
  - LABEL
  - ENV/ARG
  - VOLUME
  - USER
  - WORKDIR

Docker Layers

- Docker uses a **copy-on-write** (union) filesystem
- Containers copy files from lower layers to top r/w layer for writes
- All lower layers are **read-only**

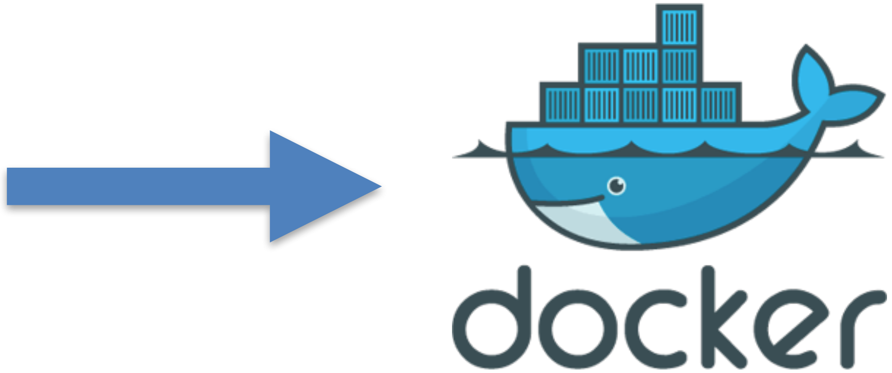
- Read-only layers allow for reuse
  - Less storage on host
  - Faster container startup
  - Fast pushes/pulls



IBM CODE

Container = Code + Dependencies

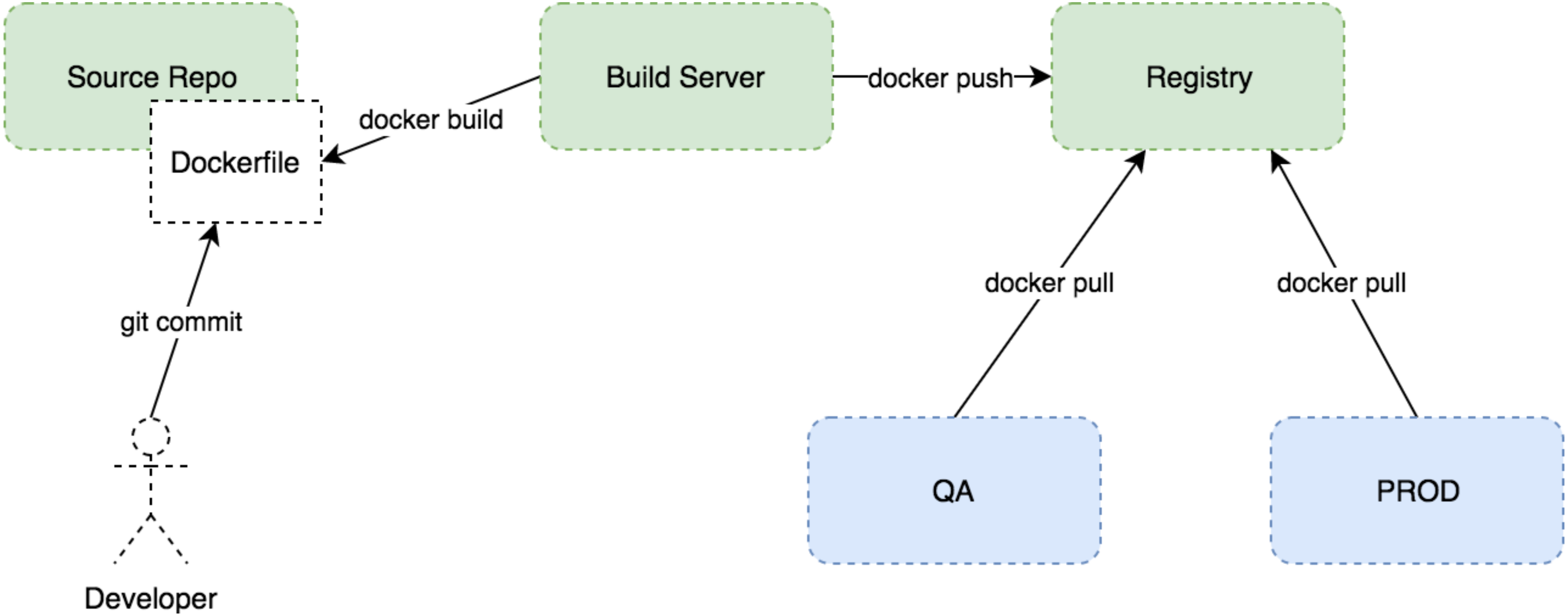
- Code (packages archive)
- App server
- Runtime versions
- System libraries and versions



What are you testing?

Are you testing these on ever commit?

- Code (packages archive) ✓
- App server ✗
- Runtime versions ✗
- System libraries and versions ✗


Source Repo
Dockerfile
docker build
Build Server
docker push
Registry
git commit
docker pull
docker pull
Developer
QA
PROD

Docker for Operations

Kubernetes

Why Containers are Appealing to Users

Lightweight & Fast
Faster startup/showdown. Gives services near instant scaling capabilities.

Faster Time to Market
Apps & dependencies are bundled into a single image. Host, OS, distro and deployment are independent allowing for workload portability.

Version Tracking
User easily rolls between versions

Simplified Isolation
Each container has its own network stack with controls over ports and permissions.

Enhanced Security
Containers allow for finer-grained control over data and software installed. Reduces the attack surface area/vulnerabilities of the apps.

Easier to Manage
Enables frequent patch of applications while reducing the effort of validating compatibility between apps/environment.

Simpler to Maintain
Install, run, maintain and upgrade applications and their envs quickly, consistently and more efficiently than VMs.

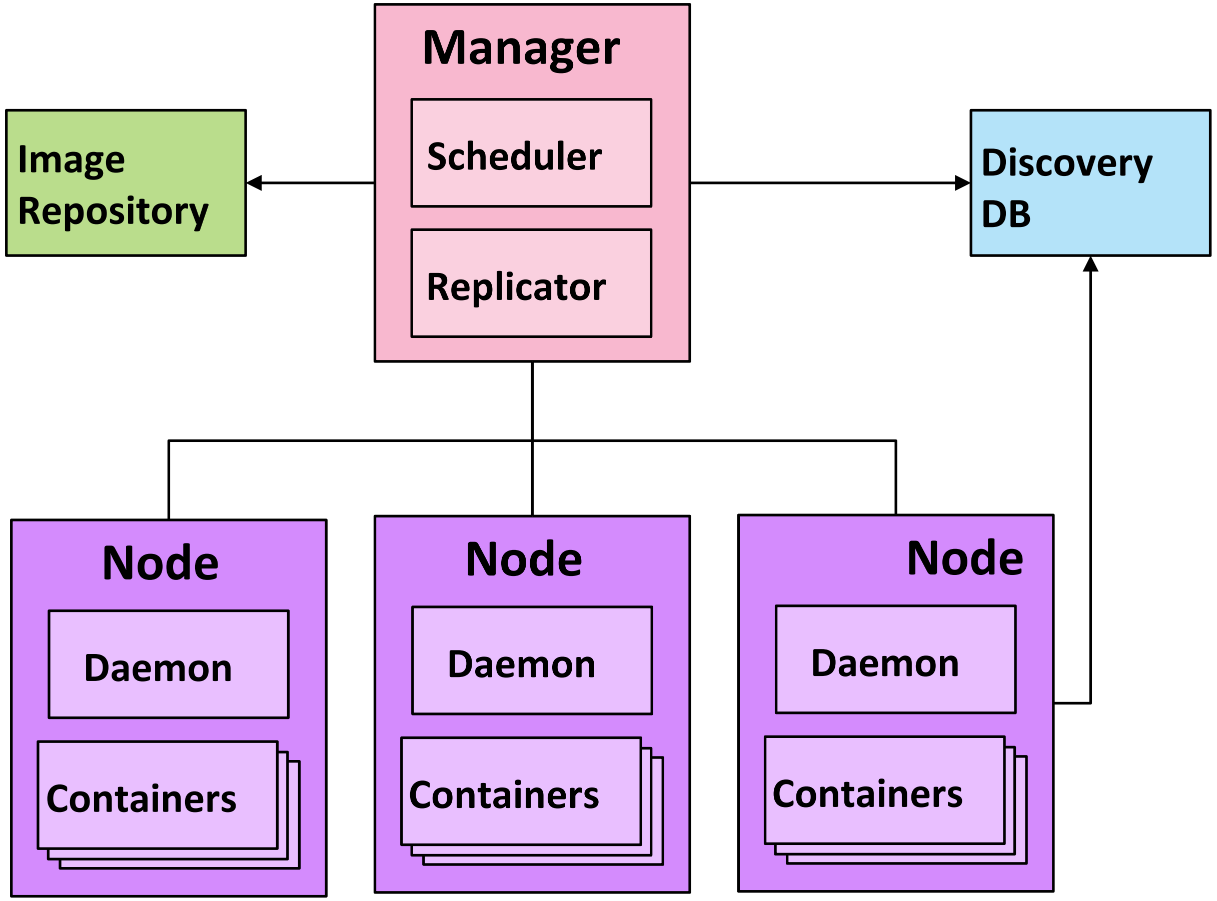
Resource Friendly
Can host more containers then corresponding VMs.

But Wait? What About Production?

- Automated scheduling and scaling
- Zero downtime deployments
- High availability and fault tolerance
- A/B deployments

What is container orchestration?

- Container orchestration
- Cluster management
- Scheduling
- Service discovery
- Replication
- Health management



IBM CODE

What is Kubernetes?

- Kubernetes - K8s
  - "Helmsman" in ancient Greek
- Container Orchestration
  - Provisions apps, services, deployments, vols, nets, etc... with a desired state
  - Kubernetes then tries to align the system to that desired state

  - Similar to Docker's SwarmKit
  - But was there first and does so much more
- FYI: Kubernetes == K8s == Kube

Kubernetes - Background

- http://kubernetes.io

- Started by Google
  - Initial release June 2014
  - Up to v1.6 now
  - Now part of the CNCF
    - Moving K8s to an open governance model was the driving force behind CNCF

- Large/wide community
  - While still heavily controlled by Google they're trying to shift that

Lab

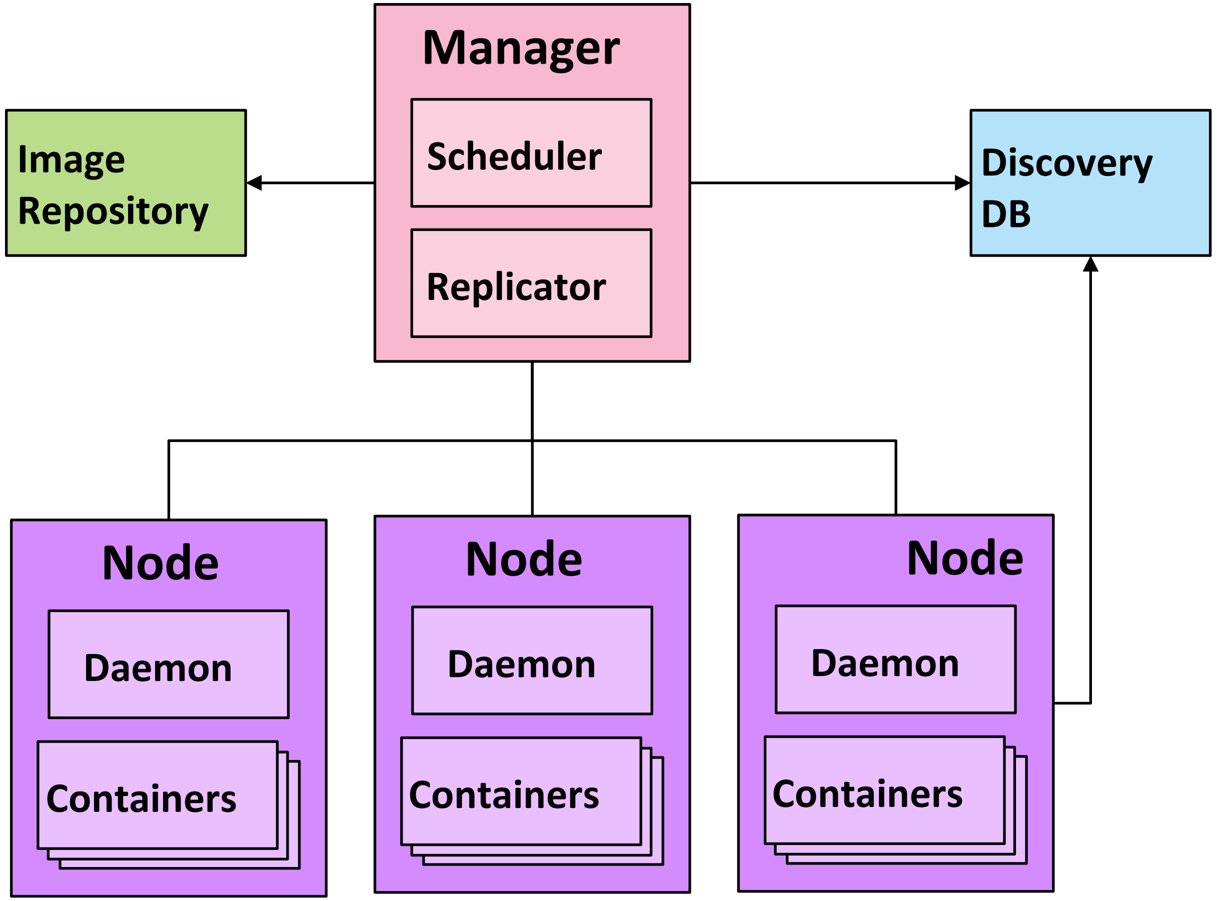
- https://github.com/IBM/kube101
- Go to “workshops”, then “Lab 1”

- Follow the instruction verbatim, or…
- Replace “guestbook” with “helloworld”
- Replace “ibmcom/guestbook:v1” with “[dockerhub id]/python-hello-world”

IBM Cloud Kubernetes Service

Benefits of Container Orchestration

- Automated scheduling and scaling
- Zero downtime deployments
- High availability and fault tolerance
- A/B deployments



But Wait? What About Production?

- Kubernetes by itself is not enterprise-ready
- Deploying your own Kubernetes is challenging
  - Updating nodes
  - Setting up networking correctly
  - Managing security
  - Installing Kubernetes with high-availability (multiple masters)

IBM Cloud Container Service

- “Push-button” clusters
- Dedicated team at IBM to handle deploying K8s with enterprise-ready configuration
- Powerful tools
- Intuitive user experience
- Built-in security and isolation
- Leverages IBM Cloud Services such as Watson

- https://www.ibm.com/cloud/container-service

Bonus Lab

- https://developer.ibm.com/code/patterns/deploy-spring-boot-microservices-on-kubernetes/

Next Steps

- Intro Labs
  - https://github.com/IBM/intro-to-docker-lab
- Full set of labs on IBM Cloud Kubernetes Service
  - https://github.com/IBM/kube101
- IBM Code
  - Collection of patterns, how-tos, blog posts, tech talks
  - https://developer.ibm.com/code/

Questions?

john.zaccone@ibm.com