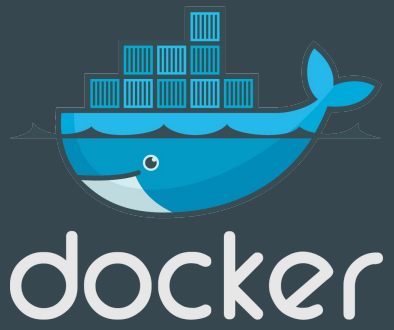
Intro to Containers

...

(and a practical sample)



Krumware

Colin Griffin

Chief Engineer
Developer (fullstack?)

Helping companies build better software, and software developers.

Forward

- Thanks!
- Ask questions!
- I don’t know everything
- https://www.github.com/KrumIO/docker-pwa-sample

Enterprise Applications - Distributed Systems

- Monolith vs “Microservices”
- Lots of parts and pieces that work together
  - Many APIs, many “Apps”
- Scalability
  - Scale up (supply/demand), Scale down (save $$$!)
- SaaS all the things?
  - Not Always, we still need to ship
- Portability
  - Avoid platform and vendor lock
  - Ship anywhere, get the best price

Background

- LXC - Linux Containers
- dotCloud project -> Docker
- Wrote their execution environment
- Open-Sourced in 2013
- Experienced rapid support and adoption
- Open Container Initiative - standardize containers

Container Basics

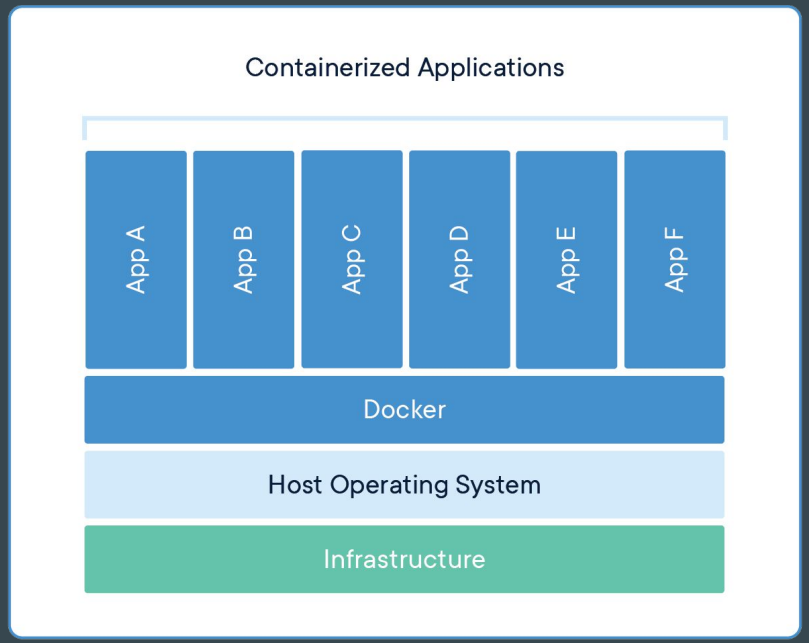
- What is a container
- Why not a VM?
- Benefits
- Drawbacks
- Container Concepts
- Using Containers
- Shipping Container Images

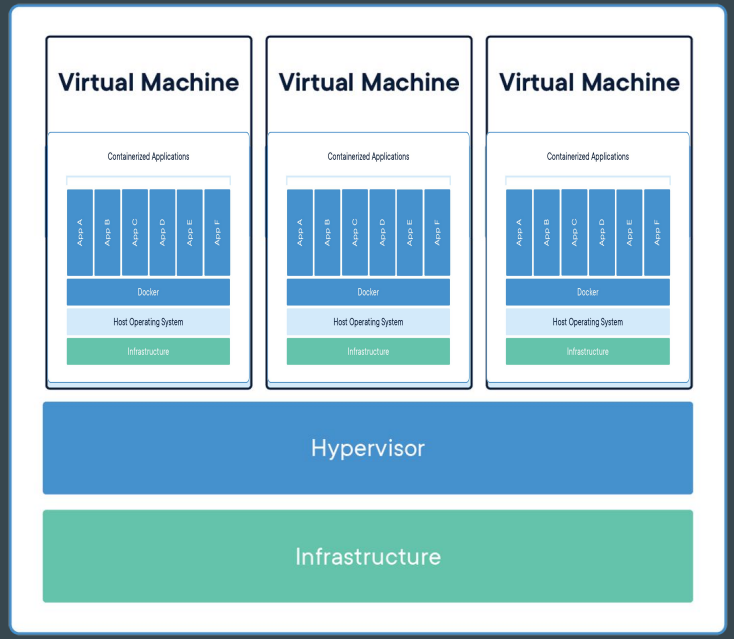
What is a Container

- Container Image vs Container
- “Container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings”
- A “Container” is that container image at run-time
- So, an EXE?
  - Maybe if the exe packaged all of the dependencies
  - An installer? No manipulation of the host system.
- Why not a VM?

Containers vs VMs

- Open Container Initiative - limited proprietary software
  - Run-time spec, image spec
- No Guest OS
- Ship only what is necessary, focus on the files not the OS
- Size, containers are dramatically smaller
- Share resources, do more with less
- Intermediate Images (layers), caching!
- ...Pictures!


Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure


Virtual Machine
Virtual Machine
Virtual Machine
Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure
Hypervisor
Infrastructure

What's the Diff: VMs vs. Containers

VMs	Containers
Heavyweight	Lightweight
Limited performance	Native performance
Each VM runs in its own OS	All containers share the host OS
Hardware-level virtualization	OS virtualization
Startup time in minutes	Startup time in milliseconds
Allocates required memory	Requires less memory space
Fully isolated and hence more secure	Process-level isolation, possibly less secure

https://www.backblaze.com/blog/vm-vs-containers/

Benefits

- Predictability, “What you ship is what you get”
- Focus: Less infrastructure, more code
- Why not just deploy something using AWS Lambda or other cool cloud services?
- Agnostic: Limited vendor lock-in, Open Container Initiative
- Portability: Ship anywhere, share with other developers
- Similarity to Git Branch workflow

Drawbacks / Watch out for

- Containers are not for everyone
- “A container for everything” - NO
- Creates complexity in some systems
- Sometimes applications need more control over the host hardware
- Lots of images, lots of data

Applications

- Continuous Integration
  - Automated testing
  - Feature branch testing via image tagging
  - Build once run many
- Delivery
  - Ship more images, write less scripts
- Sharing
  - Use in development
  - Learning
  - Store pre-requisites in containers

Prerequisites

- Docker Desktop - https://www.docker.com/products/docker-desktop
- Your first image: https://hub.docker.com/_/hello-world/
  - `docker pull hello-world`
  - `docker run hello-world`

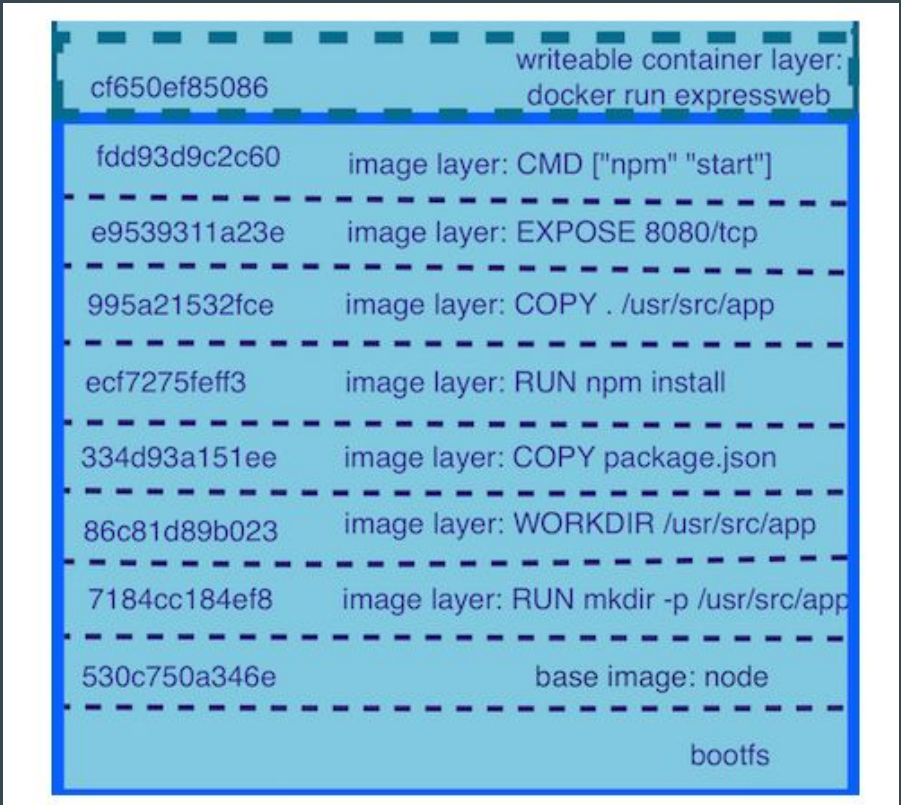
Images / Composure

- “Building” a container image
  - Scripted file
- Commands
  - COPY - copy files *into* the build
  - RUN - run shell commands
  - WORKDIR - “sticky” `cd`
  - EXPOSE - provide network access
  - ENTRYPOINT/CMD - Need at least one

https://medium.com/@jessgreb01/digging-into-docker-layers-c22f948ed612

Images / Composure

- “Building” a container image
  - Scripted file
- Layers
  - Each step generates a Layer
  - Caching
  - Base Images
  - New commands: FROM



https://medium.com/@jessgreb01/digging-into-docker-layers-c22f948ed612

Volumes

- Storage Drivers - specify storage drivers
- Provides Control (read-only, write)
- Provides access to host system storage
  - My development folder?
- Persistance
- Portability, share information between containers

Control methods

Environment variables

- Accessible inside container
- Secrets
- Internal Options
- Flexibility

Labels

- Inspect information about other containers
- Discovery, options
- Information for Orchestrators and Schedulers via engine APIs

Putting Containers to Work

- Docker run
- docker-compose
- Orchestrators/Schedulers
  - Swarm
  - Cattle/Rancher
  - Kubernetes
  - OpenShift
- PaaS
  - AWS EKS
  - Azure AKS
  - Rancher

Docker Basics

- docker pull hello-world
- docker run hello-world
- docker ps
- docker image ls
- docker volume ls
- docker system prune --all --volumes

Registries

- Public vs Private
- hub.docker.com
- Image Namespacing - Tag
  - {registry}/{organization}/{image}:{version}

  - `docker tag 0e5574283393 myregistryhost:5000/fedora/httpd:version1.0`
- Security
  - `docker login`

Lets create some stuff

- https://www.github.com/KrumIO/docker-pwa-sample
-

Putting it together

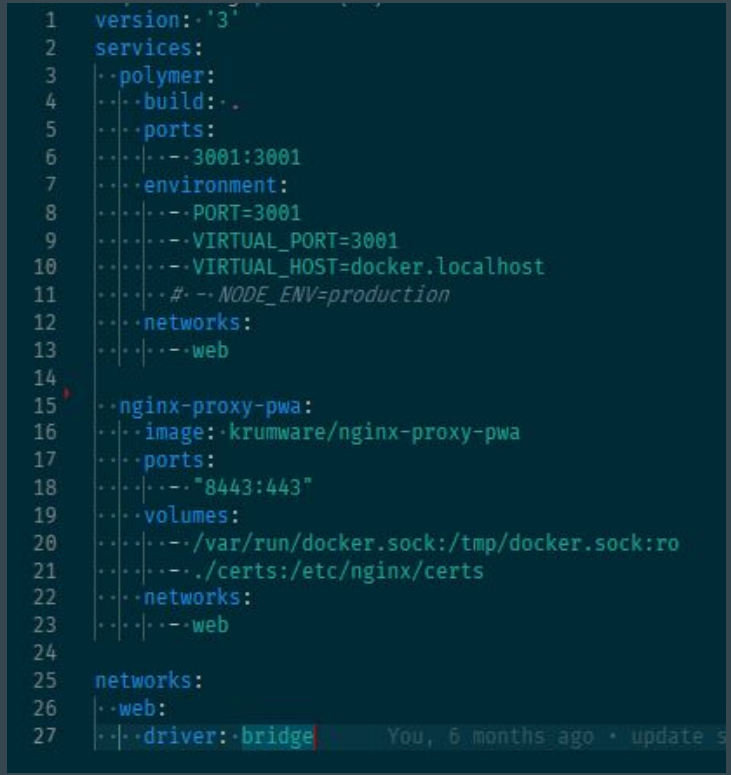
- `docker build . -t krumware/docker-pwa-sample:latest`
- `docker push krumware/docker-pwa-sample:latest`
- `docker run krumware/docker-pwa-sample:latest`
- `docker run krumware/docker-pwa-sample:latest`

For others/clients/devs/etc:

`docker pull krumware/docker-pwa-sample:latest`

Services - Running Multiple Containers

- Lightweight scheduling for containers
- docker-compose.yml
- Services - a collection of common containers
- Orchestrate Multiple Services
- Networking
  - Ports ‘external:internal`
  - Links
  - Drivers:
    - Bridge
    - Overlay
    - MACVLAN
    - Custom
      - Weavenet



Advanced docker-compose

- Images
- Build paths
- Overrides
  - Docker-compose.override.yml
- “Kubernetes Mode”
- Local Demo

Orchestrators / Schedulers + Shipping Containers

- Docker, Docker-compose,
- More Advanced, multi-node support:
  - Kubernetes, Docker Swarm, Rancher Cattle (kind-of)
- PaaS
  - Openshift, Rancher, Amazon EKS, Azure AKS
- Demo: Docker-Swarm + Jenkins
- Demo: PaaS Tour: Rancher 1.6

Thanks!

- Learning Recap:
  - Difference between VMs and Containers
  - Basic container concepts
  - Containers in practice
  - Services in practice

Resources

- https://www.docker.com
- https://www.docker.com/resources/what-container
- https://www.backblaze.com/blog/vm-vs-containers/
- https://medium.com/@jessgreb01/digging-into-docker-layers-c22f948ed612
- https://kubernetes.io/
- https://rancher.com/
- https://www.okd.io/
- https://github.com/openshift/origin