tafritz@outlook.com

 www.linkedin.com/in/tfritz/en

 http://www.slideshare.net/ToddFritz

 https://github.com/todd-fritz

Agenda

1. Why Containers?
2. Business Value
3. Adoption
4. Docker Basics
5. Orchestration Toolkit
6. Decompose the Monolith
7. Microservices
8. API Gateway
9. Service Discovery
10. Security & Networking
11. Putting it Together
12. Questions?

About Me

- Senior Software Architect @ Altisource Labs
  - The opinions contained herein may not represent my employer, but I believe they should.
- Platform Development, Middleware, MoM, EIP, EDA, etc
- Deployment perspective
- Exposed to many environments, technologies, people
- Life-long learner and always curious
- Novice bass player
- Scuba diver - next adventure
  - https://www.bikiniatoll.com/divetour.html

Presentations

DevNexus 2015 (today!)

http://www.slideshare.net/ToddFritz/2015-03-11_Todd_Fritz_Devnexus_2015

Great Wide Open - Atlanta (April 3, 2014)

http://www.slideshare.net/ToddFritz/2014-04-03legacytocloud

AJUG (April 15, 2014)

http://www.slideshare.net/ToddFritz/2014-april-15-atlanta-java-users-group

Video - https://vimeo.com/94556976

Why Containers?

Why Containers?

- *Familiarity with Docker helpful*
- Not Cargo Cult Programming (pun intended)
- Streamline development and testing
- Portable and predictable across environments
- Composable and dynamic
- Ideal for Microservices
  - Deployment
  - Scaling
- Clocker, Apache Brooklyn, RancherOS

Why Containers?

- More disruptive to cloud providers who have defined their own cloud deployment architectures
- Does a heavy O/S get in the way?
- Docker fever! Rapid innovation during past year!
  - Red Hat – early adopters
  - IBM – Q4 2014 - Enterprise Hosting for Docker
  - Microsoft - Q4 2014 - Azure
  - Amazon – Q4 2014
  - Digital Ocean, Joynet
- Need for more standards
  - Blueprints, e.g. Apache Brooklyn

Why Containers?

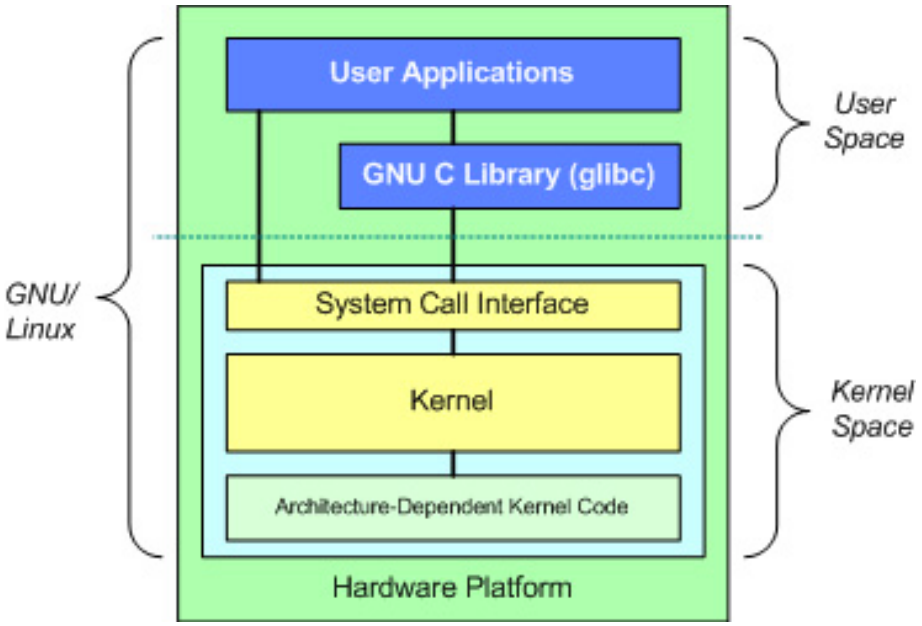
Ancestry?

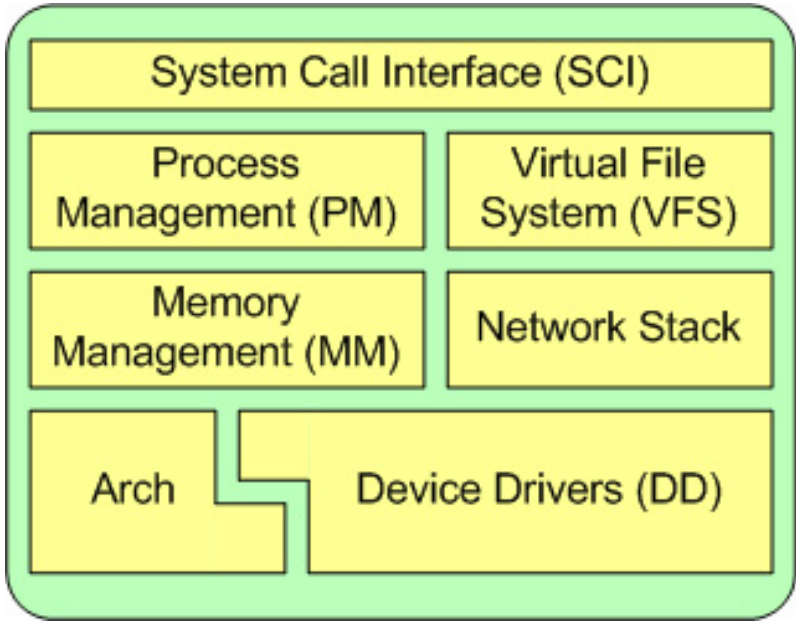
- Jailing / BSD
- Solaris Zones / Containers have been around for a long time

Why Containers?

Thought: a paradigm shift for the O/S?

- Redefines "Kernel Space"? CoreOS, RancherOS
- Redefines "User Space"?
- Better fit for distributed computing



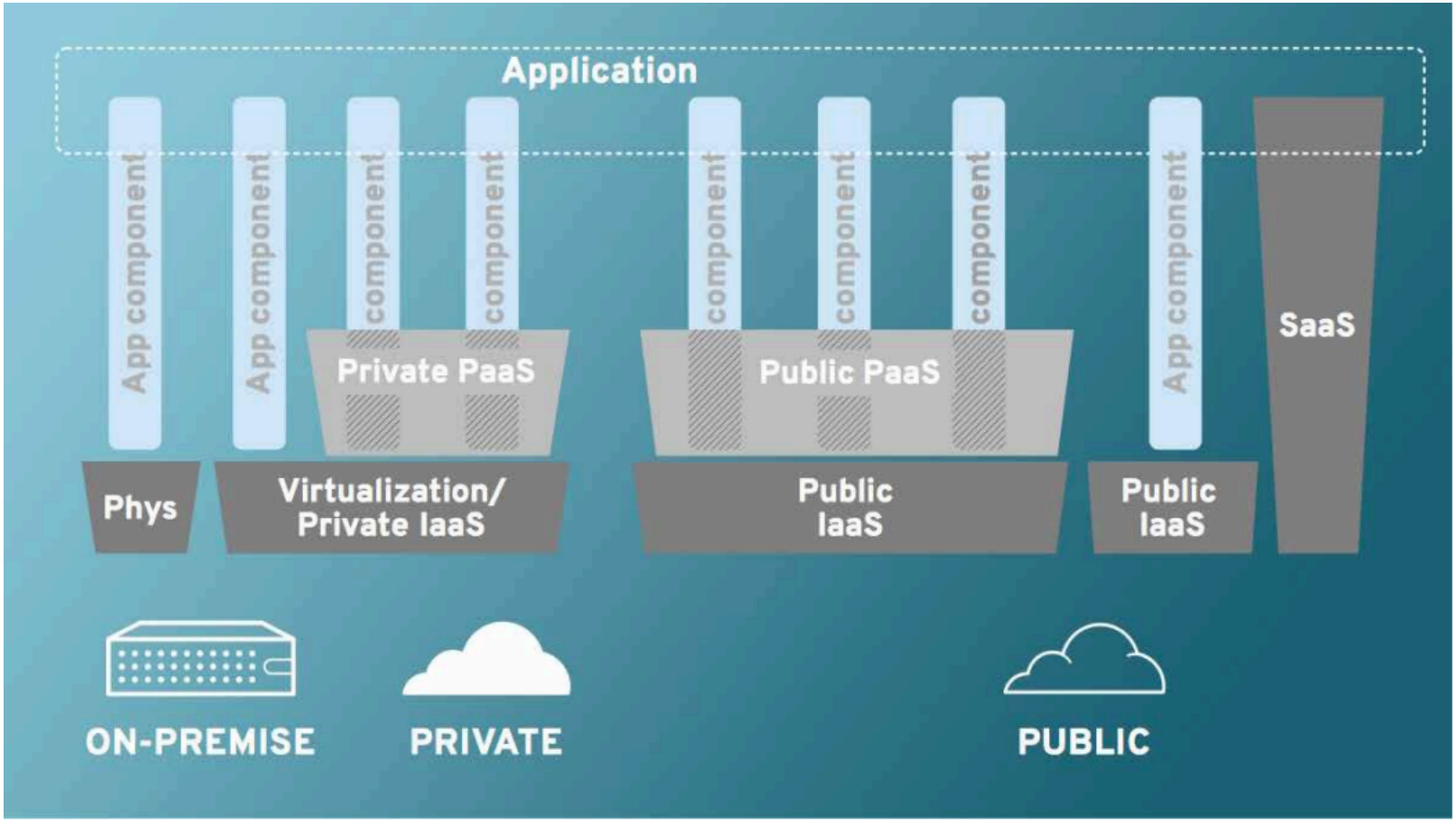


http://www.ibm.com/developerworks/library/l-linux-kernel/

Business Value

Business value

- xPaaS
  - Decoupled, separated concerns, vendor specialization



https://img.en25.com/Web/RedHat/JB_xPaaS_Tech_Overview_11454037_v3_0913cd_web.pdf

Business Value

- Necessary? "If it ain't broke, don't fix it."
- Simplifies deployment, management, administration
- Enhances Architecture, Developers, Dev Ops
- Speed to market with scale-up
- Replaces sandboxing with containers
- Reduces cost
  - Portable and predictable
  - Streamline development, QA
  - Scales via configuration
  - Emerging technologies for autonomic self-management

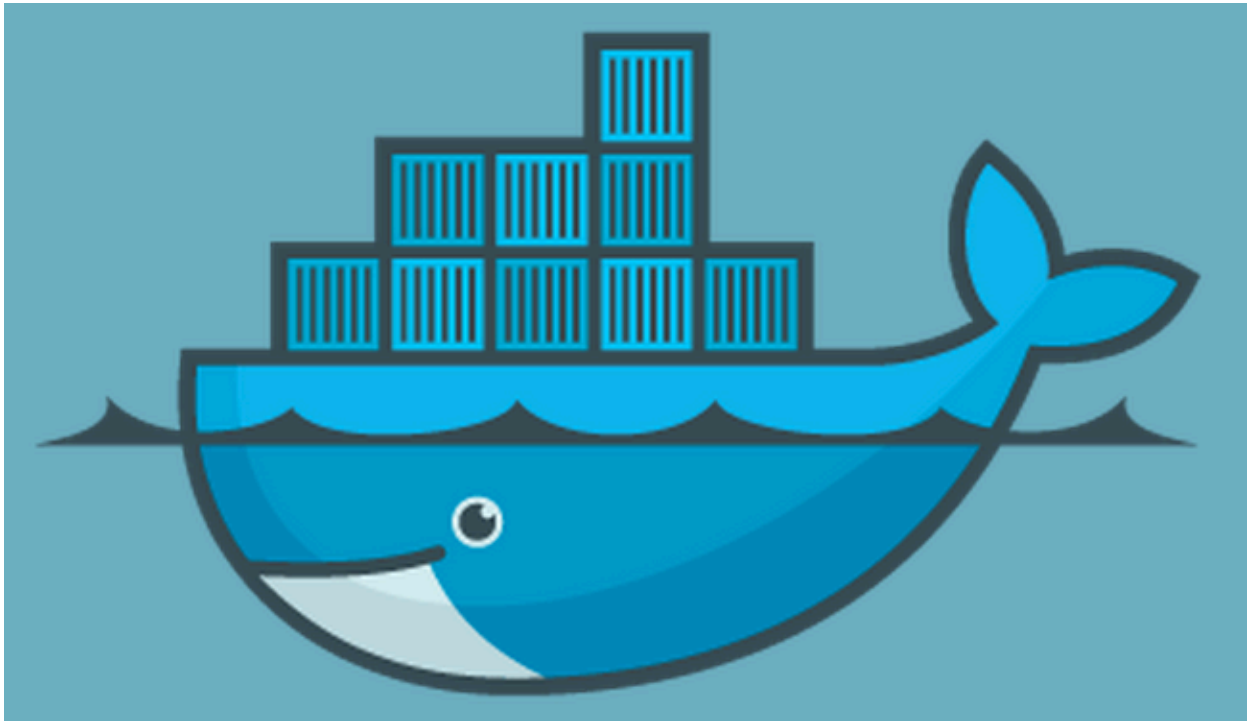
Business Value

Containers enable the "Intelligent Cloud"?

- Container is basic unit
- Self-scaling
- Self-managing; platform intelligence reduces TCO
- Self-healing
- Containers/apps expose metrics and events (security)
- Container manager collects metrics
- Machine learning interprets metrics to identify stimuli
- Platform reacts to interpreted stimuli (OPS, security)
  - Programmed or self-organized

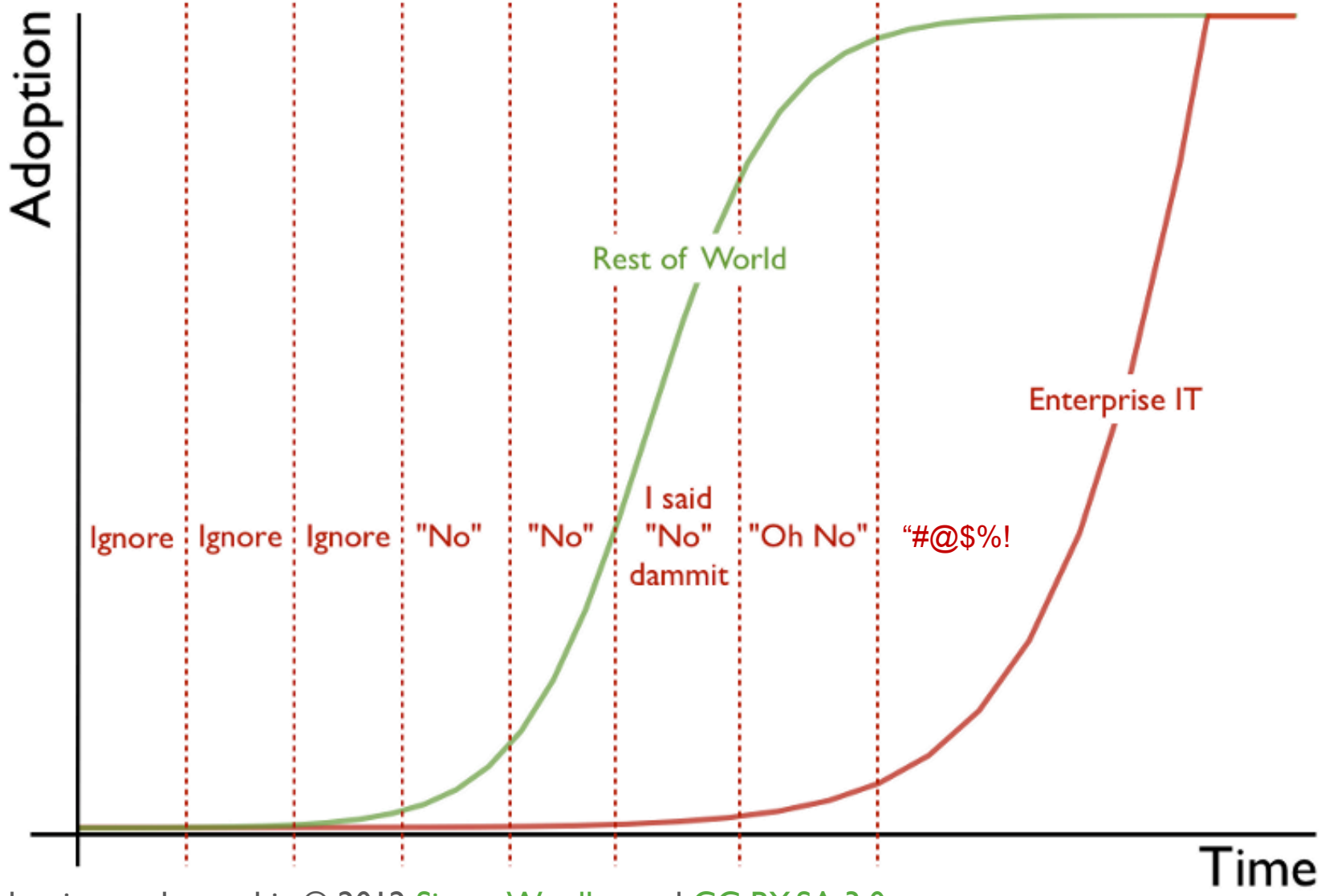
Adoption

Adoption



Whale you be my container?

Adoption



Adoption - Obstacles

Management

- Need right skills and people to pull it off
- Heard it is complex with perceived security flaws
- Lack of in-house technical expertise, schedule, time
- Learning curve, fear of change, unable to quantify risk
- How to justify the expense?

Developers, QA, DevOps

- Learning curve
- Need for direction and guidance
- Know-it-alls or superficial, tangential decision making

Architects and Technical Leadership

- Learning curve
- Executive sponsorship

Adoption - Overcoming Obstacles

Architects

- Begins with a knowledgeable evangelist
- Sell to developers and articulate technical / business benefits
- Involve DevOps – "Not a Chef replacement"
- Mentor growth of in-house expertise

Developers / QA

- Evangelize benefits to team, generate excitement, spike it
- Obtain buy in, get team to sell up the chain
- Try it out, be curious do a PoC!

Management

- Evangelize to forward-thinkers who "get it"
- Sell ROI, speed to market, benefits of good architecture
- Reduces TCO of engineering and operations

Docker Basics

Docker Basics

- Open-source
- Easier scalability
- Lightweight, portable, insulated containers
- Reuse from dev ("local cloud") through production
- Can run at scale on VMs, bare metal, cloud
- Encapsulate any payload - application
- Run consistently including hybrid environments
- PaaS enabler
- Replaces sandboxing

Docker Basics

- What is Docker?
  - "(…) an open platform (…) to build, ship, and run distributed applications."
- Docker Engine
  - "Portable, lightweight, runtime and packaging tool."
- Docker Hub Registry
  - "Cloud service for sharing applications and automation workflows."

https://www.docker.com/whatisdocker/

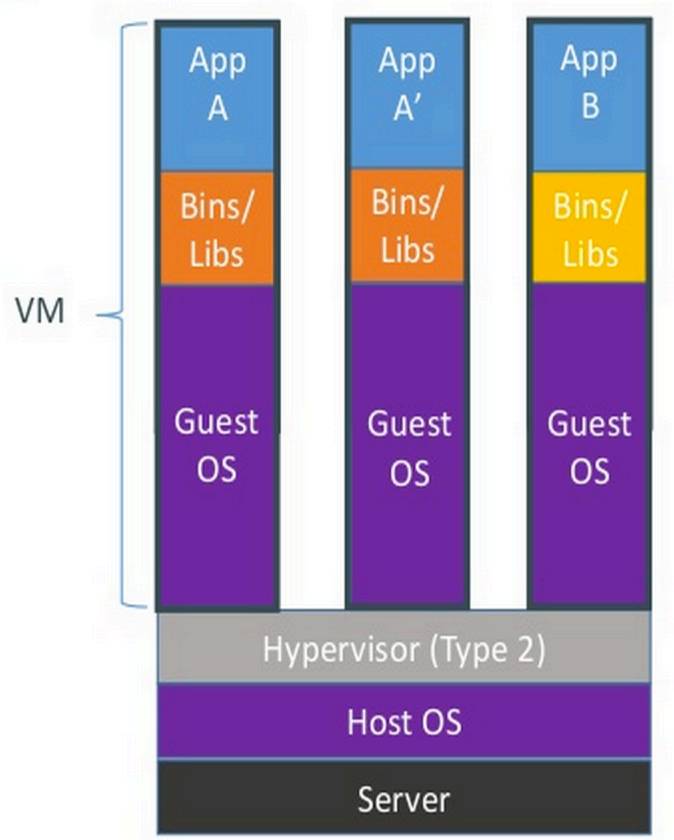
Docker Basics

- Docker Image
  - Configured Snapshot of server or service.
  - Can be extended
- Docker Container
  - Docker images run within the container
- DockerFile
  - Instructions/commands that create image

Docker Basics – Dockerfiles

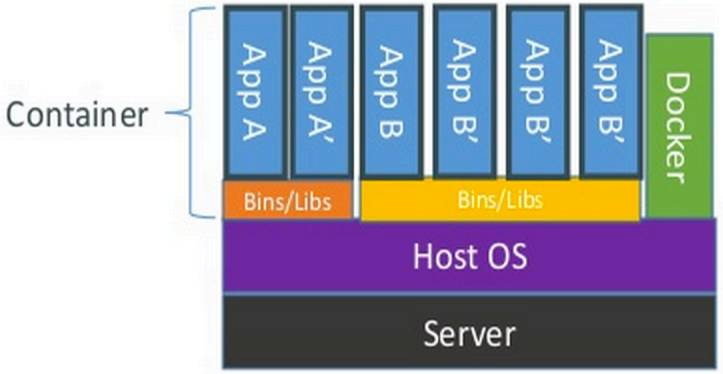
- Used to build containers by scripting creation actions of image layers
- Less powerful than traditional configuration management software
- Example
  - In discrete steps, instruct Docker to build an image by taking a default Ubuntu distro, apt-get install dependencies, and then add the application.
  - Each Dockerfile command creates a new image layer, and clever structuring of the commands enables reuse

Docker Basics – Virtual Environment vs. VM



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

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



http://www.slideshare.net/dotCloud/docker-intro-november

Docker Basics – How, What, Why

- Automates app deployment within containers
- Originally based on Linux Containers (LXC) via an API to expose lightweight virtualization that:
  - Process isolation
  - Utilizes LXC, cgroups, and the kernel
  - Does not include a separate operating system
  - Kernel provides resource isolation
    - CPU, memory, block I/O, network
- Now have libcontainer (native Go) – replaces use of LXC
  - Interface between user space and kernel
- Works well with Chef, Puppet, Vagrant
- Dynamic provisioning – ideal for PaaS

Docker Basics

- Container is the basic building block
- Analogous to a slice of a VM image, containing application code or binaries, as well as the execution environment and dependencies
- Images may be built atop each other
- An image contains only the incremental changes required to transform its base image to the state required by the image
- May contain metadata, such as how to run what is inside the image, expose ports, etc
- Acts as an instantiable container template

Docker Basics

WHY

- Run everywhere
  - Regardless of kernel version (2.6.32+)
  - Regardless of host distro
  - Physical or virtual, cloud or not
  - Container and host architecture must match*
- Run anything
  - If it can run on the host, it can run in the container
  - i.e. if it can run on a Linux kernel, it can run

WHAT

- High Level—It's a lightweight VM
  - Own process space
  - Own network interface
  - Can run stuff as root
  - Can have its own /sbin/init (different from host)
  - <<machine container>>
- Low Level—It's chroot on steroids
  - Can also *not* have its own /sbin/init
  - Container=isolated processes
  - Share kernel with host
  - No device emulation (neither HVM nor PV) from host)
  - <<application container>>

http://www.slideshare.net/dotCloud/docker-intro-november

Docker Basics

- Intra-container communication via sockets (Docker links). Containers can talk to each other.
- Familiar lifecycle: started, stopped, or kill
- Can create images from previously run containers, to persist changes made while container was executing

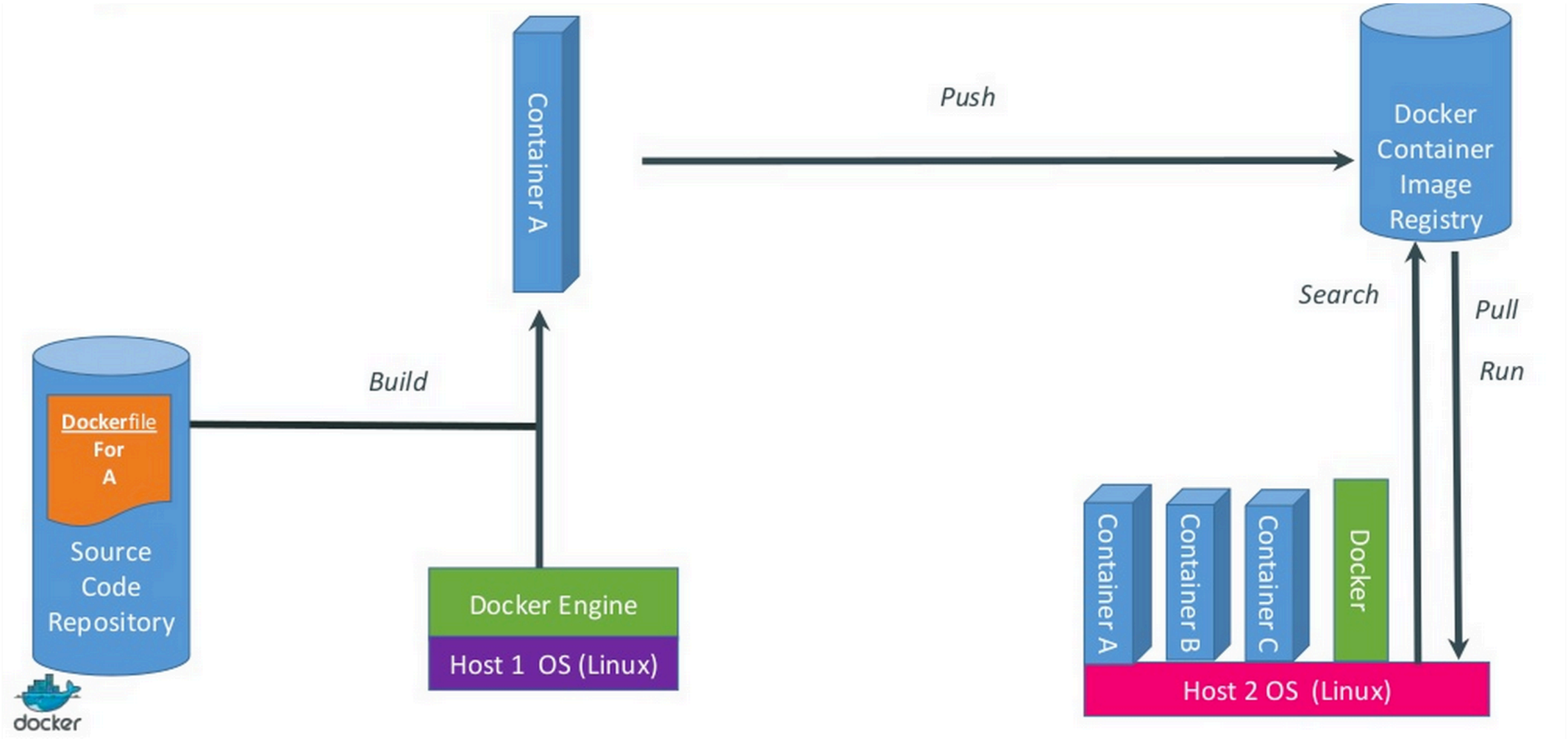
Docker Basics

- Docker has three parts
  - **docker daemon** runs as root to manage containers
  - **docker containers** spawn from images, which are tiny and can be versioned
  - **docker repository** allows images to be exchanged and versioned like code (public or private)
- Each container has a unique IP address
- Port and pipework to expose containers outside host
- Share volumes, multi home, integrate containers into host network (and more)
- Continuous integration can generate versioned docker images, web hooks, repo notifications

Docker Basics

- Common use cases
  - Automate application packaging and deployment
  - Lightweight PaaS environments
  - Automate testing, continuous integration, and deployment
  - Deploy and scale web apps, databases, backend services

Docker Basics



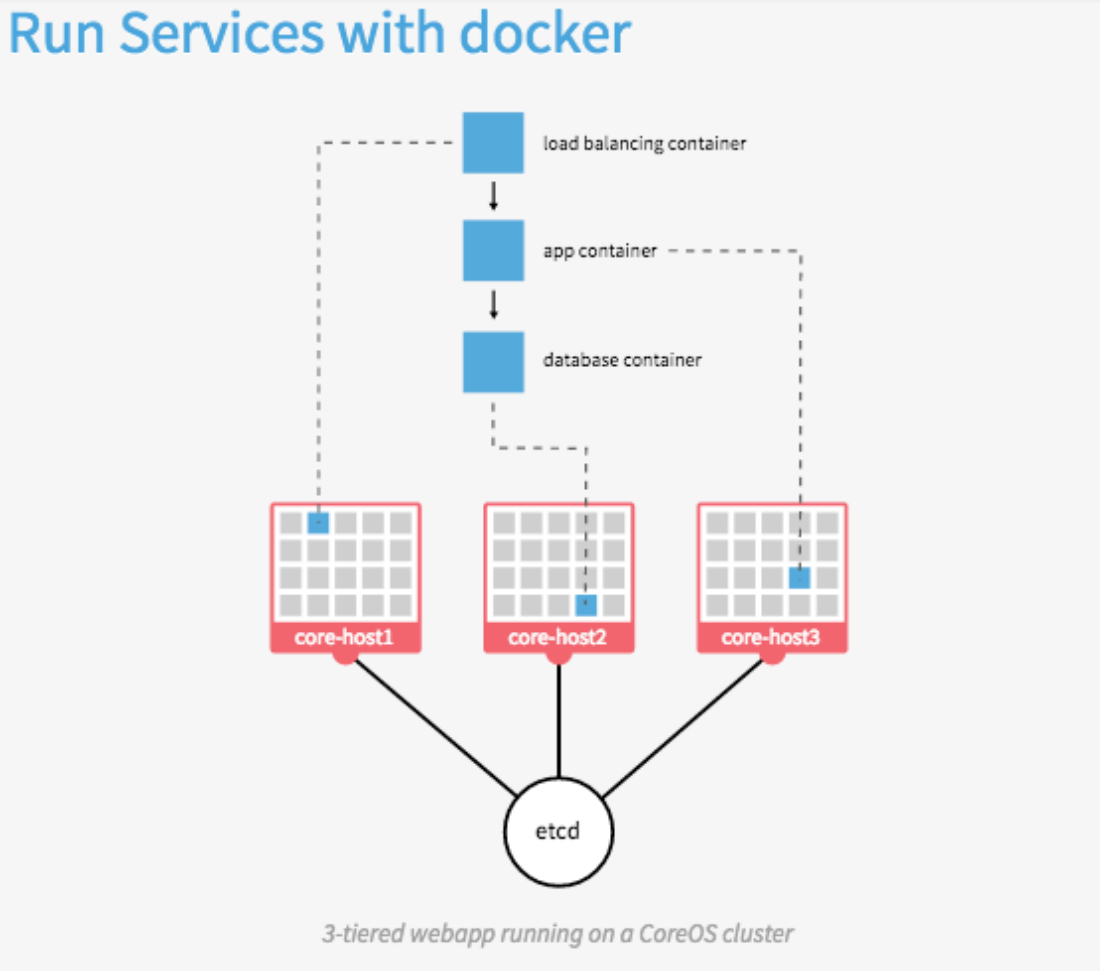
http://www.slideshare.net/dotCloud/docker-intro-november

Docker Basics – PaaS Frameworks

- Docker sold DotCloud PaaS to cloudControl
- Not opinionated
  - Deis, Flynn
  - (Both provide scaffolding to support distributed, containerized, service implementations)
- Opinionated
  - CloudFoundry, OpenShift, Apcera Continuum
  - Integrates Docker into their systems
- CoreOS, RancherOS

Docker Basics - CoreOS

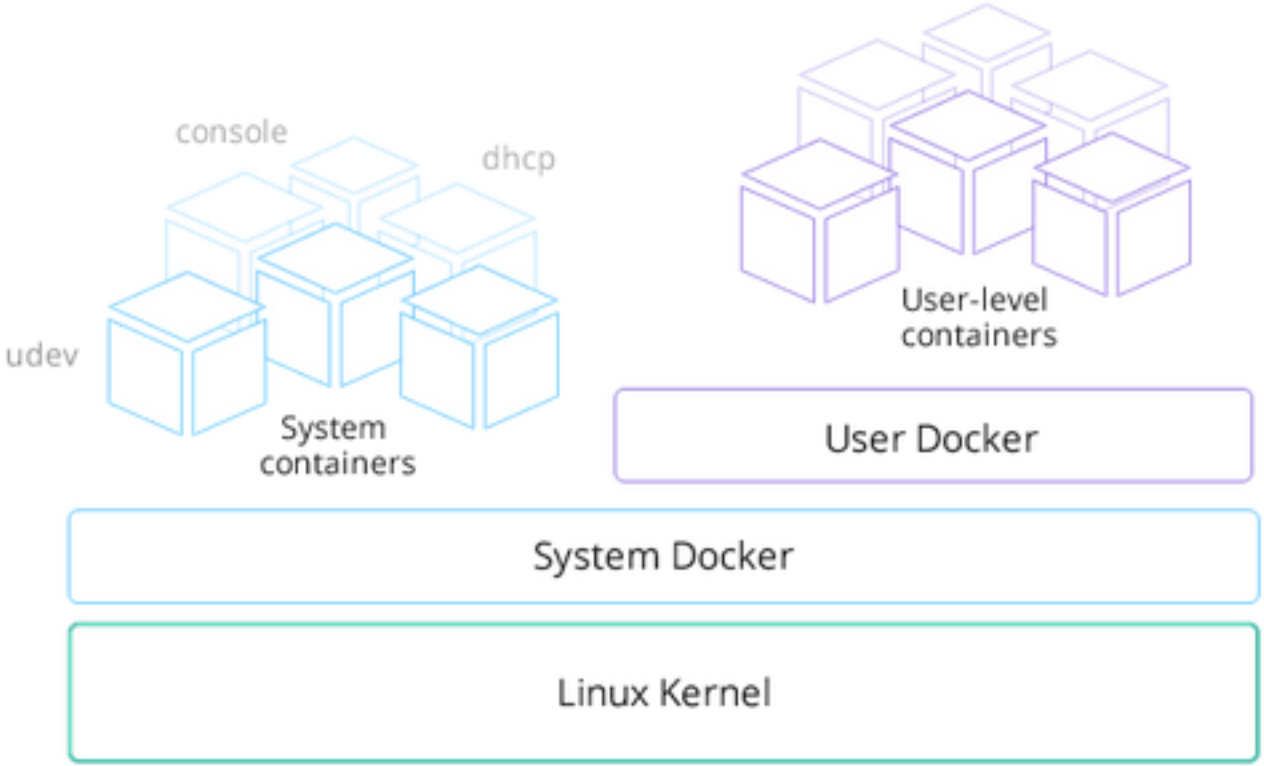
- CoreOS uses fleet to run containers with systemd



https://coreos.com/using-coreos/

Docker Basics - RancherOS

- A "System Docker" as PID1 manages "User Dockers"
- Separate Docker daemon runs in User Docker container



http://rancher.com/rancher-os/

Orchestration Toolkit

Orchestration Toolkit

- Portability across environments, providers
- Composable
- New technologies (beta), not production ready
  1. Docker Machine
  2. Docker Swarm
  3. Docker Compose (based on fig)
- http://www.infoq.com/news/2015/03/docker-machine-swarm-compose
- Micro Virtualization to follow?

http://thenewstack.io/orchestration-toolkit-release-aims-prove-dockers-commitment-flexibility-community-ecosystem/

Orchestration Toolkit – Docker Machine

Docker Machine

- (Separate project from Docker Engine)
- Abstraction used to provision to different environments
  - Support for 12 environment providers
  - Virtualbox (local)
  - Digital Ocean, AWS, Azure, Vmware, and more
- Hybrid environments
- https://github.com/docker/machine

```
% machine create -d [infrastructure provider] [provider options]
[machine name]
```

http://blog.docker.com/2014/12/announcing-docker-machine-swarm-and-compose-for-orchestrating-distributed-apps/

Orchestration Toolkit – Docker Swarm

- Native clustering
- A pool of Docker hosts exposed as a single, virtual host
  - Schedule containers to run atop
  - Automatic workload management
- Uses standard Docker API
  - Any tool using docker daemon can Swarm transparently
  - Dokku, Compose, Krane, Flynn, Desi, DockerUI, Shipyard, Jenkins
  - Docker client
- H/A and Failover
  - Swarm performs health checks with auto-rebalancing
- Policy-based scheduling: standard and custom constraints

https://github.com/docker/swarm/

Orchestration Toolkit – Docker Swarm

- Scheduling algorithm
- Standard filters
  - Constraint (storage=ssd, storage=disk, etc)
    - storagedriver, executiondriver, kernelversion, operatingsystem
  - Affinity
    - Certain containers may perform better on same host
    - "Locality of Reference"
  - Port
  - Health
- Even Azure cloud computing platform supports it
  - http://www.eweek.com/cloud/docker-swarms-onto-microsofts-cloud.htm

Orchestration Toolkit – Docker Swarm

- Undergoing development
- Adding feature to reschedule container on failed host

Orchestration Toolkit – Docker Swarm

```
# create a cluster
$ docker run --rm swarm create
6856663cdefdec325839a4b7e1de38e8 # <- this is your unique <cluster_id>

# on each of your nodes, start the swarm agent
#  <node_ip> doesn't have to be public (e.g. 192.168.0.X),
#  as long as the swarm manager can access it.
$ docker run -d swarm join --addr=<node_ip:2375> token://<cluster_id>

# start the manager on any machine or your laptop
$ docker run -d -p <swarm_port>:2375 swarm manage token://<cluster_id>

# use the regular docker cli
$ docker -H tcp://<swarm_ip:swarm_port> info
$ docker -H tcp://<swarm_ip:swarm_port> run ...
$ docker -H tcp://<swarm_ip:swarm_port> ps
$ docker -H tcp://<swarm_ip:swarm_port> logs ...
...

# list nodes in your cluster
$ docker run --rm swarm list token://<cluster_id>
<node_ip:2375>
```

Orchestration Toolkit – Docker Compose

- Final piece of orchestration puzzle
  - After provisioning with Machine
  - After defining clustering with Swarm
- Based on Fig
  - Define application components within one file
  - Containers, container configuration, links, volumes
- Assemble multi-container distributed apps to run atop Swarm-managed clusters
- Simple YAML configuration
- https://github.com/docker/docker/issues/9694

Orchestration Toolkit – Docker Compose

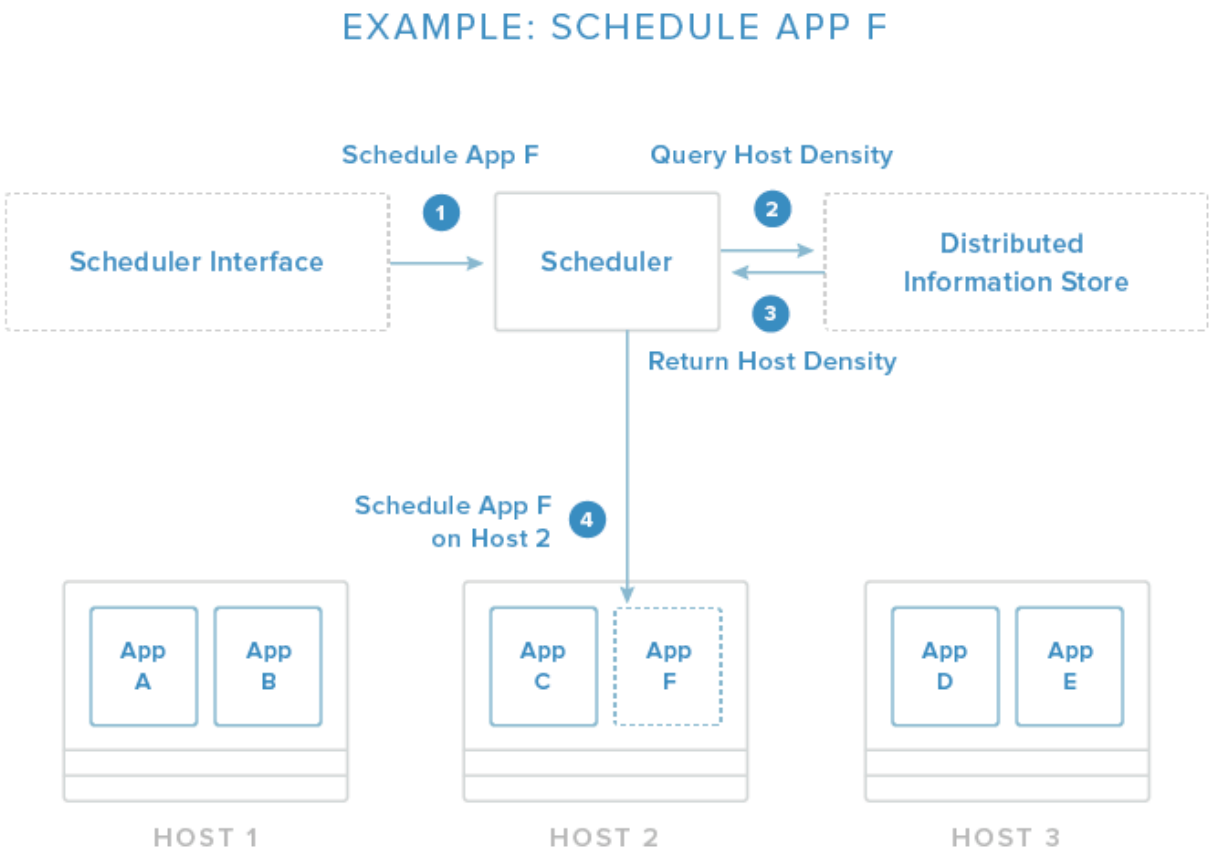
- Commands to manage lifecycle: start, stop, rebuild
- Obtain status of running services or view logs
- On Roadmap:
  - Support for test, staging, production environments
  - Integration with Swarm
  - Support for multi-team and multi-registry development
  - Improved container build automation (e.g. auto C.I.)
  - https://github.com/docker/compose/blob/master/ROADMAP.md#applications-spanning-multiple-teams

Orchestration Toolkit – Docker Compose

```
containers:
  web:
     build: .
     command: python app.py
     ports:
     - "5000:5000"
     volumes:
     - .:/code
     links:
     - redis
     environment:
     - PYTHONUNBUFFERED=1
  redis:
     image: redis:latest
     command: redis-server --appendonly yes
```

Orchestration Toolkit - Schedulers

Schedulers are responsible for starting containers on available hosts.



https://www.digitalocean.com/community/tutorials/the-docker-ecosystem-an-introduction-to-common-components

Orchestration Toolkit - Schedulers

Options

- Fleet
- Marathon
- Swarm
- Kubernetes
- Compose

Decompose the Monolith

Decompose the Monolith

- So, you have a monolithic app that needs to be modernized… For whatever reason.
  - How to rebuild it? (insert: Six Million Dollar Man ref.)

Decompose the Monolith

- No smoking in the Emergency Room!
- System Analysis
- Use an iterative methodology
  - Leverage existing knowledge and literature
  - Some analysis before
  - Decompose and modularize (scale cube)
  - Separation of concerns
  - Proof of concepts are your friend

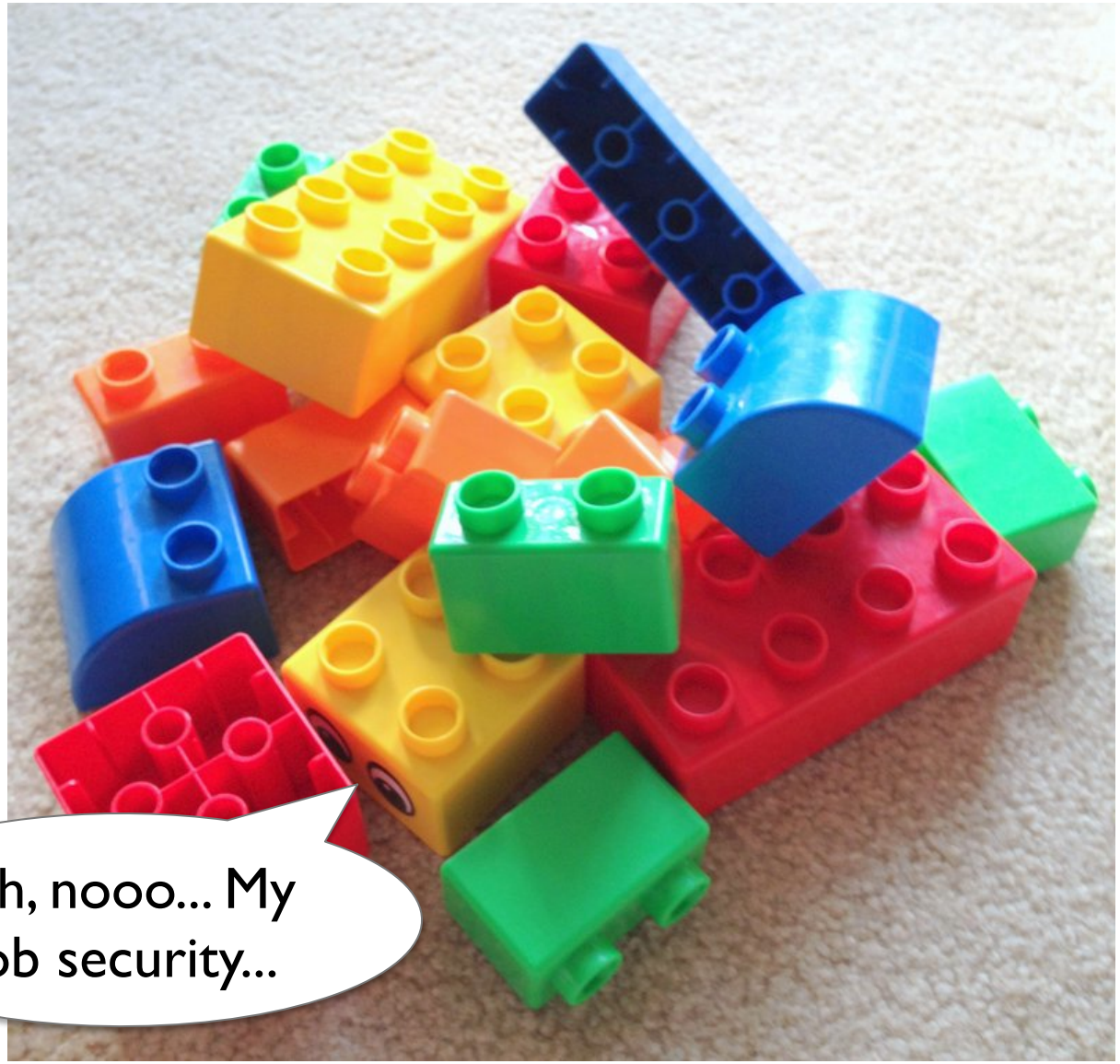
Decompose the Monolith

A simple metaphor to follow

(Please do not leave)

Decompose the Monolith

Decompose the Monolith



Decompose the Monolith

Decompose the Monolith - Technique

"Bounded Context" from Domain-Driven Design

- Identify discrete business capabilities ("Functions")
- Functions govern part of enterprise data model
- A Microservice implements a Function
- Microservices encapsulate data access via API (REST)
- Data store per microservice (no sharing)
- Overlap managed by higher-order microservices or hypermedia
  - Functions may share – e.g. "Addresses"

Also consider the "Scale Cube"

- http://microservices.io/articles/scalecube.html

Microservices

Microservices

Why?

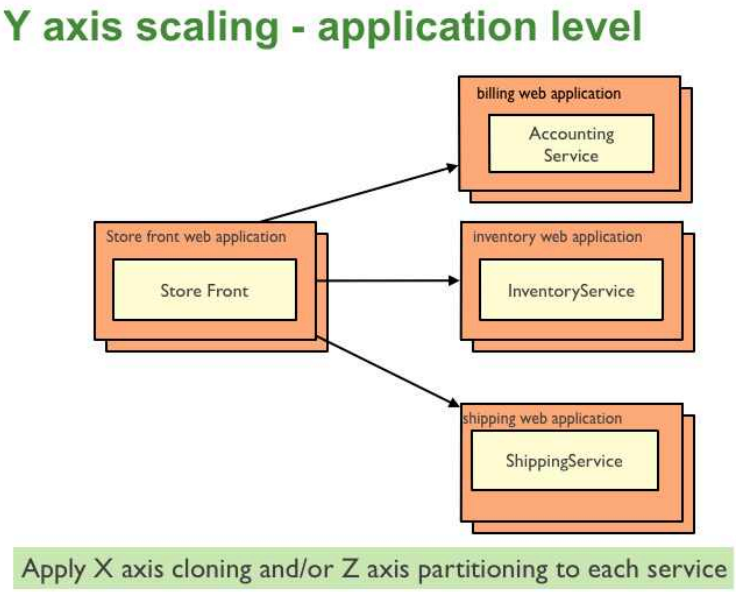
- Good fit for containers
- Ideal for distributed computing
- Correct implementation is hard
- A warning
  - http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html

The ESB from yesterday does not fit

- https://www.voxxed.com/blog/2015/01/good-microservices-architectures-death-enterprise-service-bus-part-one/
- Once upon a time, Enterprise Application Integration (EAI)
- SOA emerges, vendors rebrand EAI offerings as ESBs
- Old ESBs were centralized → new ESBs decentralized

Microservices

- Various definitions (sometimes a "Holy War")
- Why Microservices?
  - Because building monoliths is so 2005…
  - What runs in the container, matters
  - http://microservices.io/patterns/microservices.html
- Architecture for delivery of systems as a set of services
  - Small
  - Collaborating
  - Independent



Microservices – tenets – SOLID

Single Responsibility Principle (SRP)

- "One reason to change"

Open-Closed Principle (OCP)

- Open for Extension, closed for modification
- Swappable implementations behind API, Polyglot persistence

Liskov Substitution Principle (LSP)

- Implementation honors API contract

Interface Segregation Principle (ISP)

- Clients should not be forced to implement unused methods
- "Fat Interfaces"

Dependency Inversion Principle (DIP)

- Abstractions should not depend upon details. API GATEWAY.

http://www.mattstine.com/2014/06/30/microservices-are-solid

Microservices – Advantages

- Familiar concept? SOA without the ESB?
- Simple services, focused on doing one thing (well)
- Loose coupling
- Facilitates parallel development; multiple teams
- Easier to test
- Enables C.I. / C.D.
- Decouple technology and tooling from service
  - Use what you want to implement (polyglot)

Microservices – Disadvantages

- Complexity of distributed system
- Operationalization overhead
  - Each service needs load balancing, monitoring,
- Testing distributed systems is hard
- Deployment complexity; DevOps skill
- Implicit Interfaces between collaborating components
  - Backwards compatibility quicksand
- Some duplication of effort

http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html

Microservices – Best Practices

- Separate data store per microservice
- Similar level of code maturity
- Separate build, CI per microservice
- Containerize!
- Stateless!

http://nginx.com/blog/microservices-at-netflix-architectural-best-practices/

Microservices

Google gRPC

- http://www.infoq.com/news/2015/02/grpc

API Gateway

API Gateway

Two peas in a pod

- API Gateway & Microservice Architecture

API Gateway acts as single point of entry into Microservice architecture for a client

- Serve different types of clients
- Protocol Translation, etc (EIP – Camel)
- http://techblog.netflix.com/2013/01/optimizing-netflix-api.html

Handles requests in two ways

- Proxy/routing to service
- Fan-out to multiple services

API Gateway

Benefits

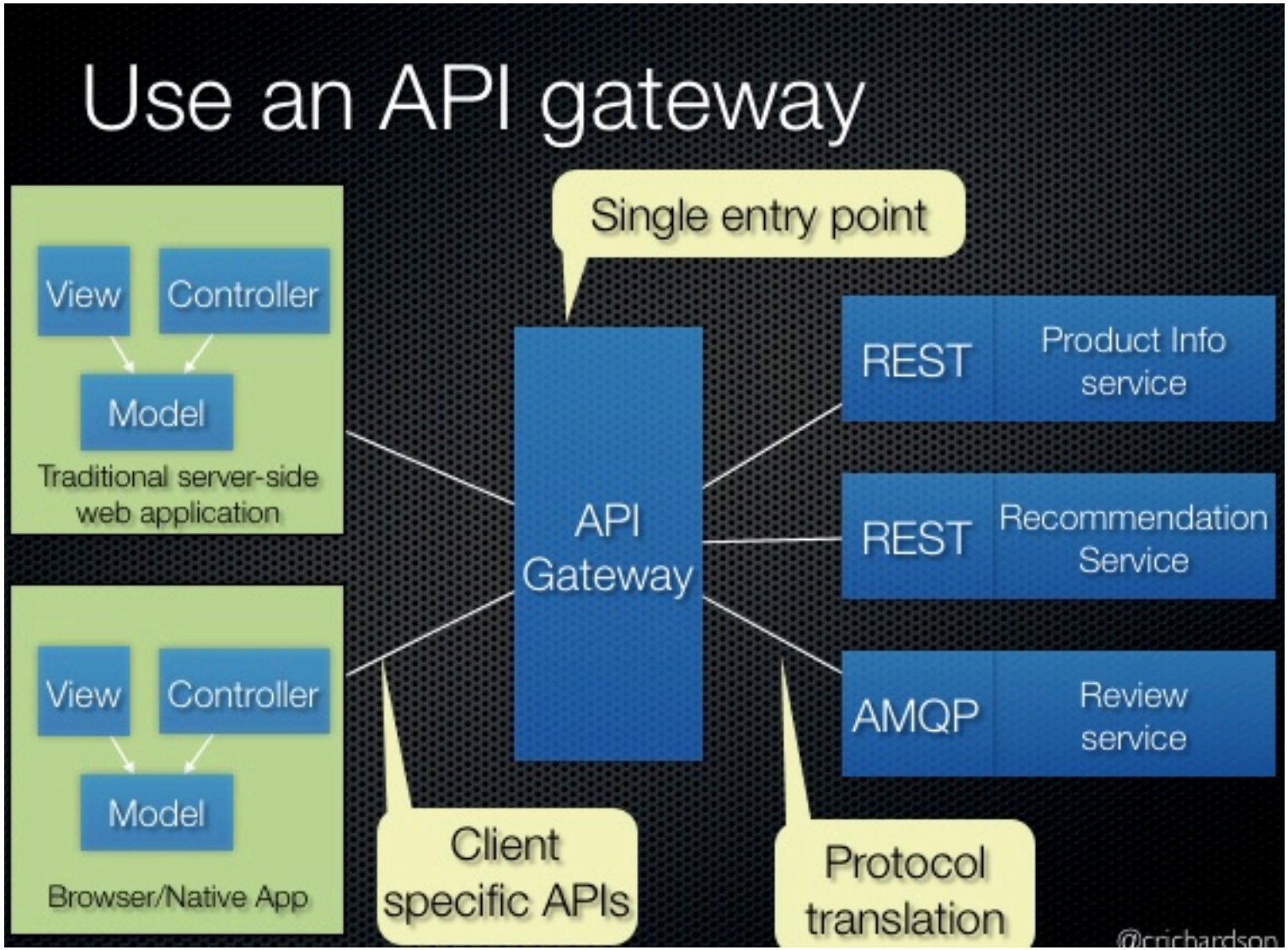
- Decouples clients from services
- Improves client's API experience
- Abstract logic for orchestrated services into gateway
- Reduces request/response round trips

Disadvantages

- More complex
- Possible increase in latency (network hop through gateway)

http://microservices.io/patterns/apigateway.html

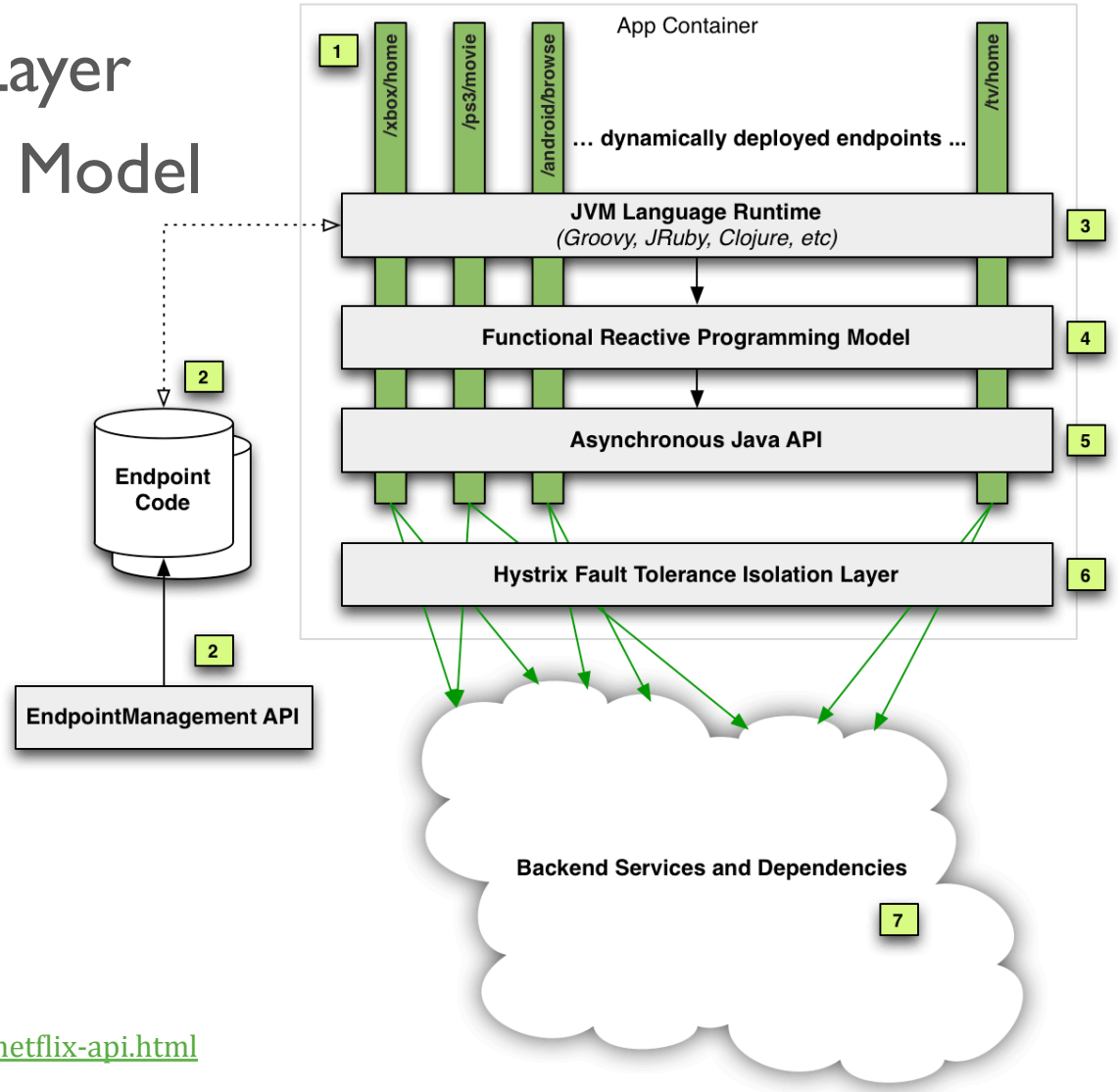
API Gateway



http://microservices.io/patterns/apigateway.html

API Gateway – At Netflix

- Dynamic Polyglot Runtime
- Fully Synch Service Layer
- Reactive Programing Model



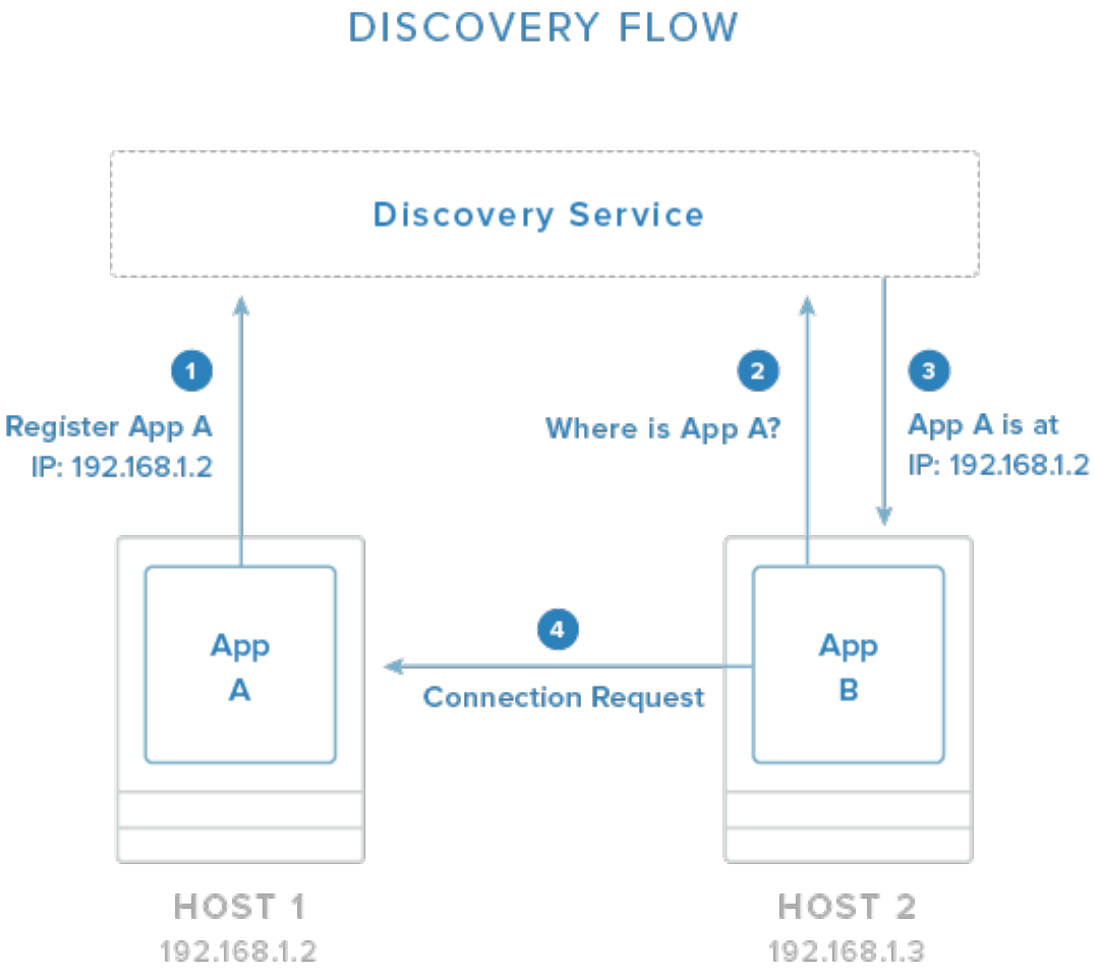
http://techblog.netflix.com/2013/01/optimizing-netflix-api.html

Service Discovery

Service Discovery

- Service Discovery requires a Service Registry
- Apps need need to register exposed services
- Clients lookup, bind, invoke
- Applications then query discovery service for how to connect to an application
- Responsibilities
  1. Provide clients with metadata to connect to service
  2. Allow applications to register to provide #1
  3. Expose global location to start arbitrary configuration
  4. Store metadata about cluster members

Service Discovery



https://www.digitalocean.com/community/tutorials/the-docker-ecosystem-an-introduction-to-common-components

Service Discovery

Popular service discovery tools

- Etcd
  - Created by CoreOS. HTTP API and a CLI.
- Consul
  - "Advanced features": health checks, ACL, HAProxy config
- Zookeeper
  - Older and lacks new features, but more stable and mature

Service Discovery

Value adds

- Crypt
  - Components can use keys to protect information
  - Clients can read if they have decryption key
- Confd
  - Dynamic reconfiguration of apps
- Vulcand
  - Load balancer. “Etcd aware”
- Marathon
  - Scheduler and basic HAProxy config management
- Nerve
  - Used with Synapse for health checks and auto cluster mgmt

Service Discovery – Fibers > Asynch?

Alternatives to Asynch?

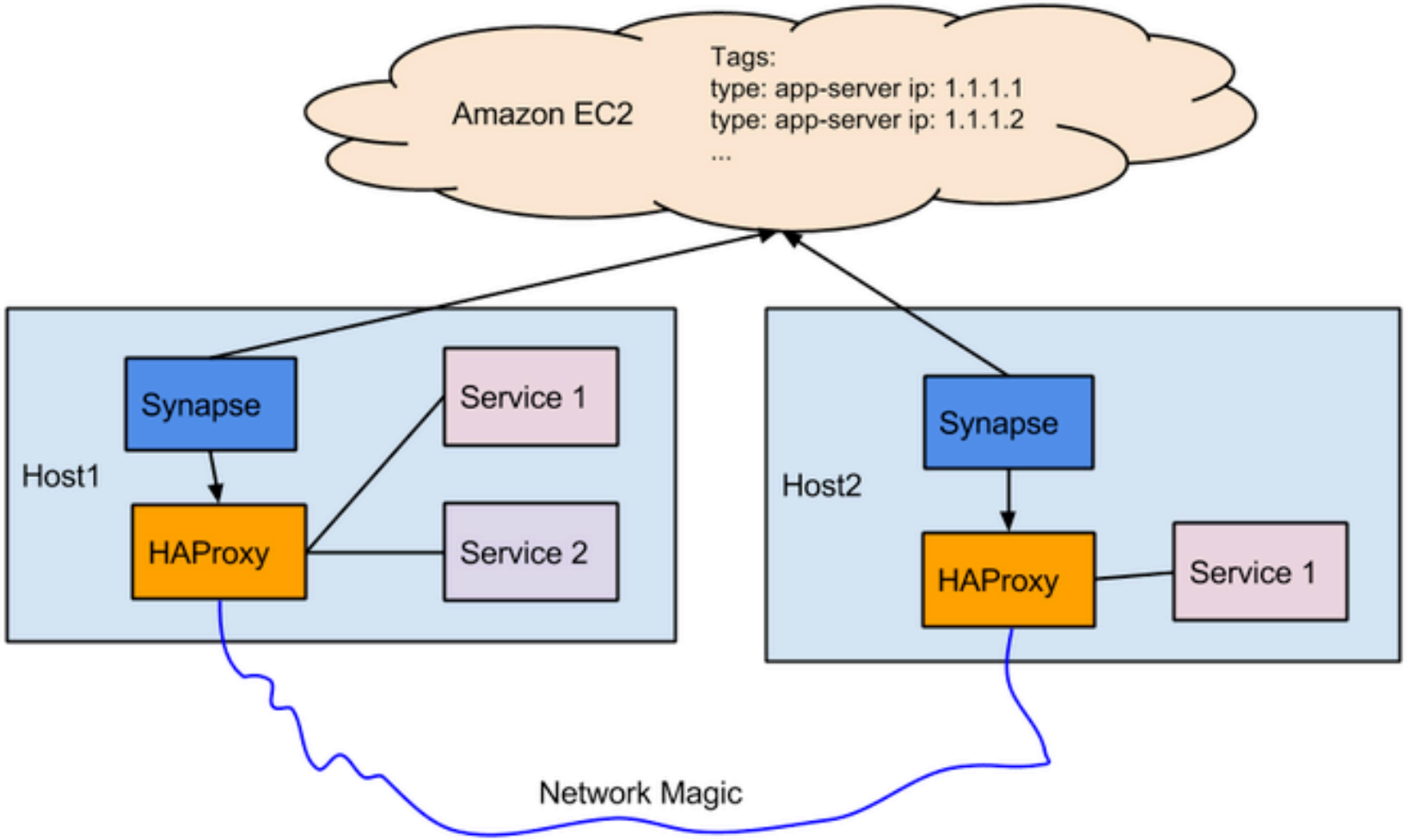
- Why deal with the heaviness?

Lightweight threads and Actors for JVM

- Quasar
  - https://github.com/puniverse/quasar
  - Lightweight threads, Go-like channels, Erlang like actors, etc
- COMSAT
  - https://github.com/puniverse/comsat
  - Scalable, concurrent web-apps
  - Not Web Framework
  - Implementations to popular (and standard) APIs, including JAX-RS, JDBC, Servlet, etc – that are called with Quasar fibers

https://www.voxxed.com/blog/2015/02/farewell-to-asynchronous-code/

Service Discovery – Clay.io example



http://zolmeister.com/2015/02/10x-service-discovery-at-clay-io.html
http://zolmeister.com/2014/12/10x-docker-at-clay-io.html

Security and Networking

Security and Networking

- Very hot topic
- Ongoing investment and innovation
- Docker adds security experts to team
  - Announced last week
- Docker acquires SocketPlane
  - Announced last week
  - 3 months from startup to acquisition? A record?
  - Native to Docker
  - Software defined network services
  - For distributed applications that span hybrid clouds
  - http://www.programmableweb.com/news/docker-extends-api-reach-socketplane-acquisition/2015/03/05

Security and Networking

- Swarm
  - TLS AuthN between CLI and Swarm
  - TLS between Swarm and Docker nodes
- Containers weaker than VM for isolation
  - VM's can use ring-1 hardware isolation
  - Prevents VM's from "breaking out"
  - Containers lack hardware isolation
  - "Shocker" attack (for pre 1.0 Docker containers)
  - Running atop hypervisor enhances resource isolation but does not necessarily make this more secure
- Container in VM reduces surface area for attacks
- Don't do stupid stuff in container, follow best practices

Security and Networking

"When we feel comfortable saying that Docker out-of-the-box can safely contain untrusted uid0 programs, we will say so clearly"

- Docker CTO Solomon Hykes

Security and Networking

- Networking capabilities somewhat limited
- libchan is officially sponsored approach
  - Go like channels over network
- Flocker
  - Proxy based approach
  - Host-portable services (including underlying storage)
- Weave
- Powerstrip
  - Use Weave and Flocker together
  - Run a Crate cluster using Weave's container networking and portable storage provided by Flocker

Security and Networking

- OpenVswitch
  - https://github.com/openvswitch/ovs
  - Multilayer software switch (VM environments)
  - Standard control and visibility interfaces
  - Designed for distribution across multiple servers
- Security companies not yet offering container endpoint protection
- Lack of tools for encryption at container level
- Lacks live migration tools

Putting it Together

Putting it Together

"Without Paas, Docker is just a bunch of containers."

- http://www.zdnet.com/article/paas-and-docker/
- Needs supporting technologies

Gaps

- Underserved: networking, storage, granular versioning

Apache Mesos for simpler clustering?

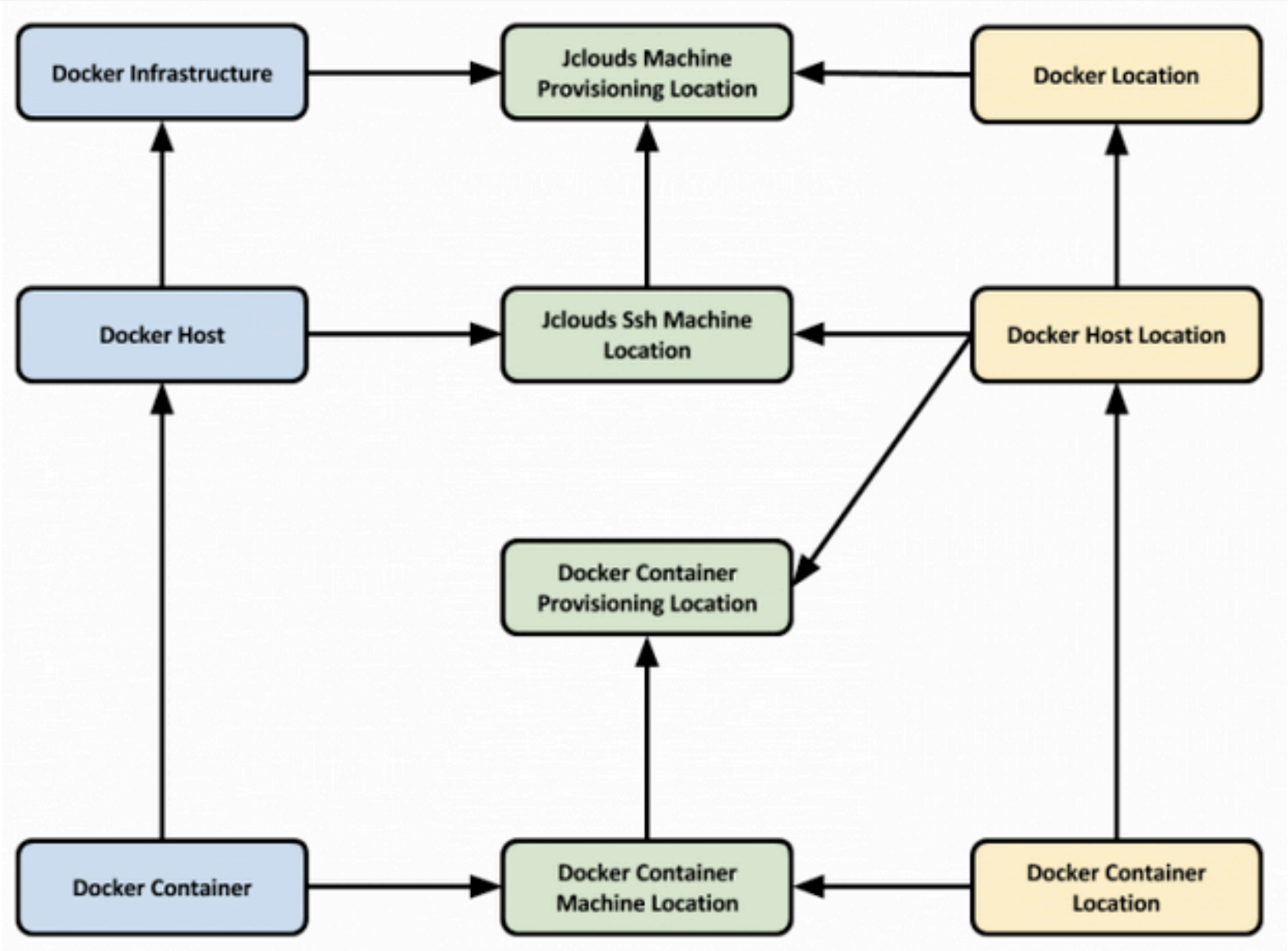
- http://www.zdnet.com/article/why-amazons-docker-service-is-linking-into-apache-mesos-for-simpler-clustering/

Putting it all Together

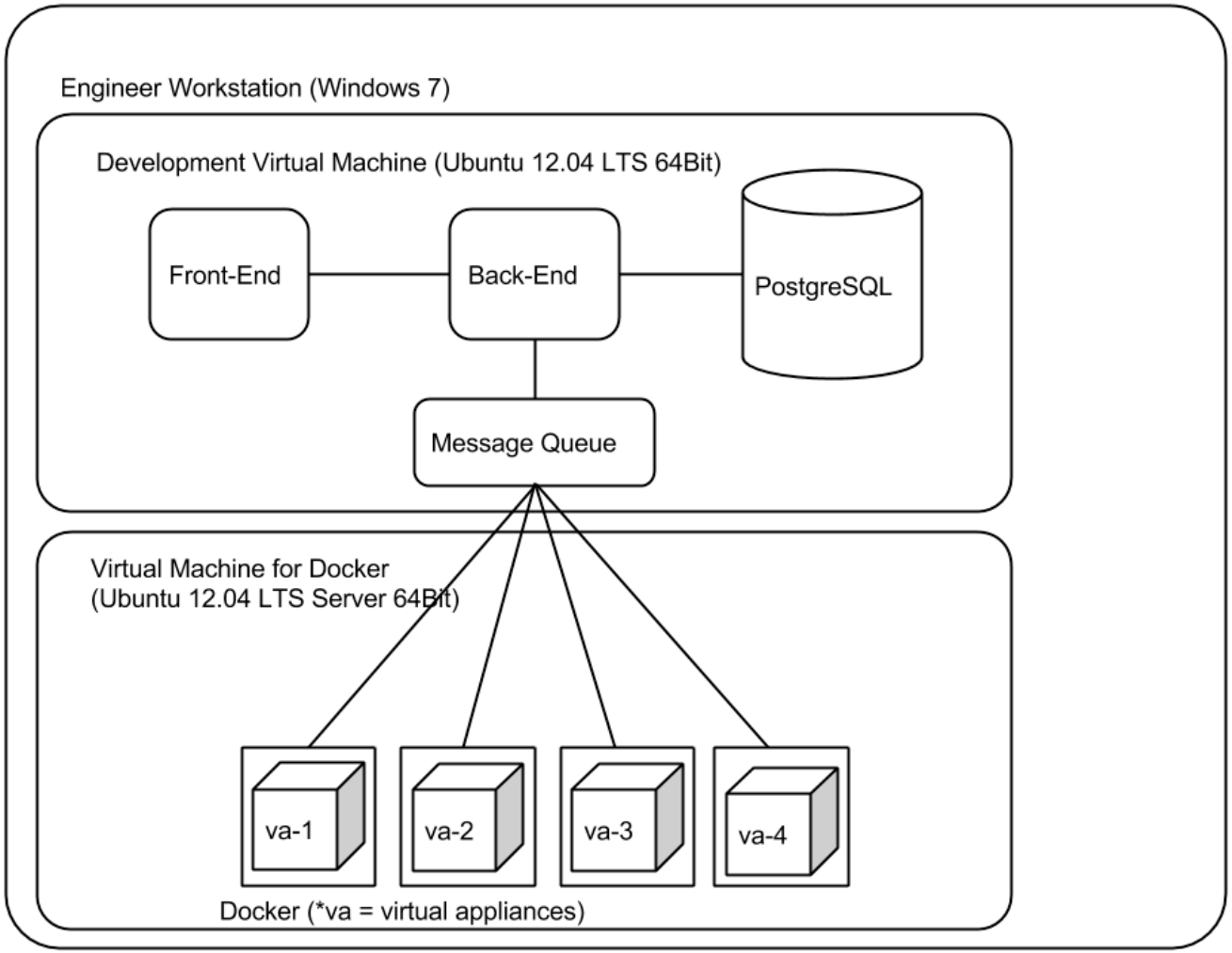
- How to put a solution together?
- Clocker (from Cloudsoft)
  - Create Docker clouds orchestrated by Apache Brooklyn
  - Single to multi host
  - Intelligent container placement: resilience, scaling, fault tolerance, resource optimization
  - Existing Brooklyn/CAMP blueprints port without change
- Apache Brooklyn
  - jClouds API (cloud API agnostic) for provision
  - Monitors and manages containers
  - Configuration blueprints that use docker infrastructure
  - Uses Dockerfiles

http://thenewstack.io/an-open-source-story-clocker-does-what-docker-cant-do-alone/

Putting it Together



Putting it Together



http://www.appneta.com/blog/automated-testing-with-docker/

Putting it Together

- Not part of this presentation
- Worth reading
- Apache Samza

  http://blog.confluent.io/2015/03/04/turning-the-database-inside-out-with-apache-samza/

Appendix

Trends & Innovation

Hardware

- "Quantum" processors – Qbits
- Input, Output, Memory, CPU, Network

Software

- Machine Learning, Artificial Intelligence
- Autonomous systems
  - Self-Tuning -> Self-Improving -> Self-Coding

Manufacturing

- 3D printing
- A.I. managed hardware provisioning and "data center" construction". On-premise construction?

Within 50 years, the Singularity?

Questions?

Resources

- http://highscalability.com/blog/2014/4/8/microservices-not-a-free-lunch.html
- https://medium.com/aws-activate-startup-blog/using-containers-to-build-a-microservices-architecture-6e1b8bacb7d1
- http://codesamplez.com/web-server/dockerize-an-application-from-scratch
- http://blog.docker.com/2014/12/announcing-docker-machine-swarm-and-compose-for-orchestrating-distributed-apps/
- http://www.infoq.com/articles/docker-future
- https://crate.io/blog/crate-flocker-weave-powerstrip-snowsprint/
- https://coreos.com/using-coreos/
- http://rancher.com/announcing-rancher-io-portable-infrastructure-services-for-docker/
- http://nginx.com/blog/microservices-at-netflix-architectural-best-practices/
- http://zolmeister.com/2015/02/10x-service-discovery-at-clay-io.html
- http://zolmeister.com/2014/10/10x-architecture-at-clay-io.html
- http://zolmeister.com/2014/10/10x-logging-at-clay-io.html