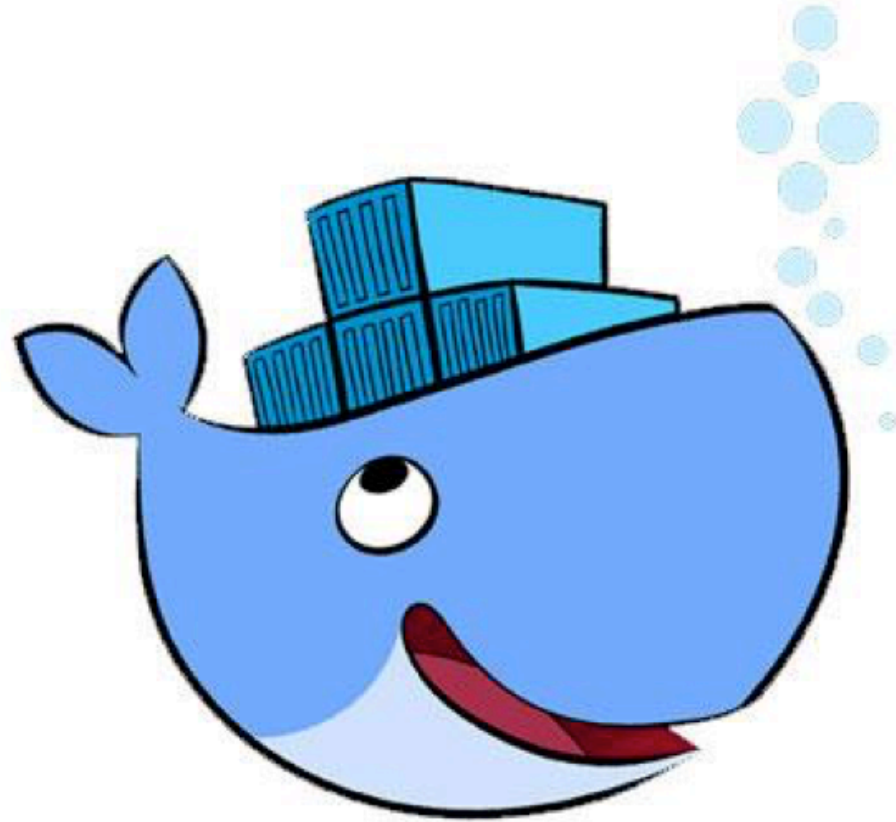


Containers in the Enterprise

Avoiding the Kobayashi Maru

Agenda

- **Containers Bring Change**
- An Approach
 - Required Software
 - Processes
 - Cultural Changes
- Additional Concerns
- Lessons Learned



Why This Talk?

- Containers are great
- You're here
- How do we get it home?
 - Especially in large organizations

Container Adoption is Crazy Fast

- Containers are being adopted at a faster rate than public cloud
 - AWS turned 10 years old this year, with 57% of companies using it
 - Docker turned 3 years old this year, already has 27% penetration
 - Last year it had 13%
- If the migration to cloud was hard for large organizations, how easy will the migration to containers be?

Change is Hard

- Approach varies based on group size
- Old roles and rituals may no longer make sense
- Messengers may get shot



Group Size Affects Approach

	Small Groups / Startups	Enterprise
Roles	People wearing multiple hats	Specific roles established
Change Appetite	Open to change; easy to convince	Many other changes happening; change fatigue
Change Pace	Easy to establish acceptable speed	Likely accepted speed: glacial
Communication	Have a standup	Exercise in herding cats
Fear	Low embarrassment if failure; change or die	Fear of making mistakes can be very high

Containers: Going from Rabbit Ears to Cable

- In traditional model, software choices typically restricted
 - Push to use similar platforms (and versions) across enterprise
 - Ease of operations: easy
 - “I know apache”
- In container model, software choices are vastly increased
 - Developers can have programming language of the month
 - Ease of Operations: difficult
 - “I know apache, nginx, lighttpd, caddy and Hiawatha”



Developers Need to Adjust

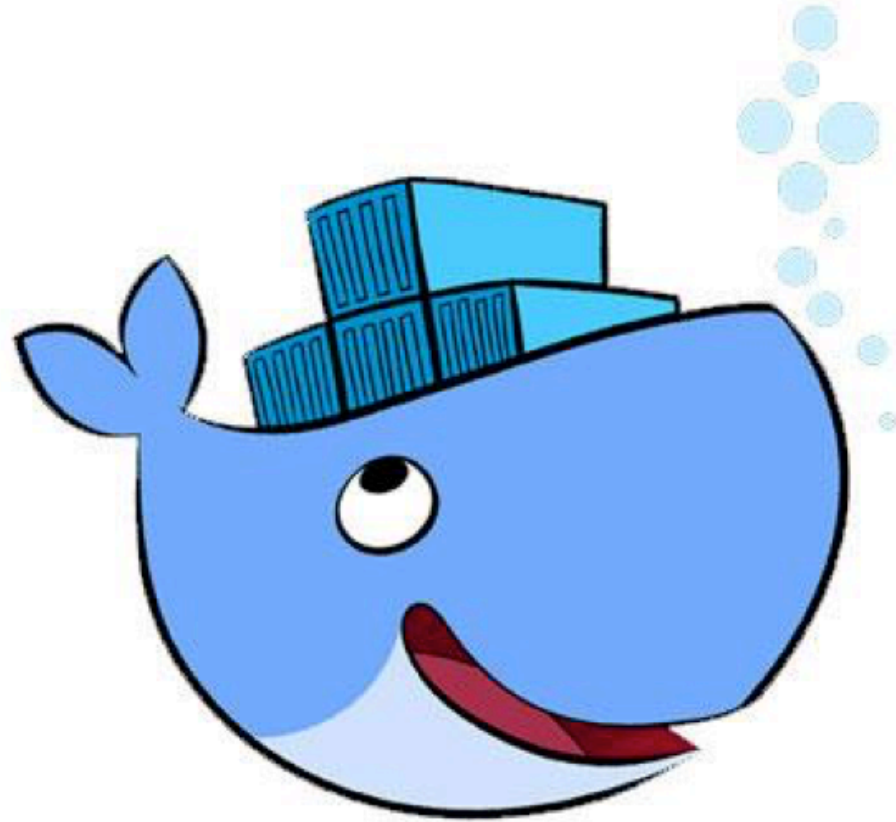
- Developers are being empowered, but need to take on additional responsibility
- Need to know how to build underlying software
 - Even if it is just *FROM nginx*
- Need to either:
 - Know how to document operational routines of their services and train others
 - System Administrators no longer explaining how Apache works!
 - Embrace DevOps culture



You Need a Plan

Agenda

- Containers Bring Change
- An Approach
 - **Required Software**
 - Processes
 - Cultural Changes
- Additional Concerns
- Lessons Learned



What Do You Need For Containers?

Most companies already have most of what is needed:

- Enterprise-Ready Container Registry
- CI/CD Build Environment
- Container Orchestrator
- Version Control System
- Job Ticketing System
- Company Wiki-like system

Enterprise-Ready Container Registry

Can be done with hosted solution, but some enterprises may require on premises solution

- Typical needs:
 - Group Membership
 - User permissions
 - Both Developer and Machine
- Nice to have:
 - Vulnerability Scanning / Notification
 - Auditing

CI/CD Build Environment

Most enterprises have this, but some groups resist embracing it

- Must haves:
 - Docker (or some other container runtime) available
- Good to haves:
 - Integration with version control
 - Triggers for automatic builds
 - Notification integration with chat rooms, etc.
 - Integration with container orchestrator
 - Integration with ticketing system

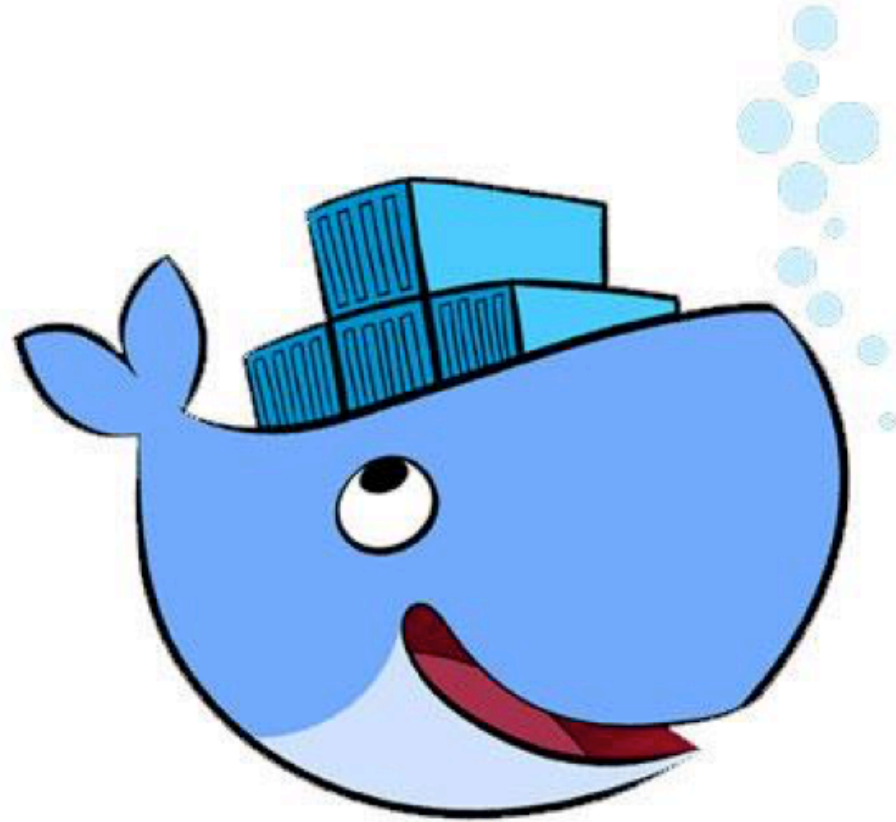
Container Orchestrator

Even if you do not use a container orchestrator for production needs now, get familiar with one

- What its used for:
 - Ability to run / smoke test built container images as part of validation process

Agenda

- Containers Bring Change
- An Approach
 - Required Software
 - **Processes**
 - Cultural Changes
- Additional Concerns
- Lessons Learned



Standardize On Single Container Registry

Operations / Information Security will thank you

- Easier to audit
 - Writes *and* Reads
 - Vulnerable Images
 - Ability to revoke compromised images *and* know who is pulling them
- Avoids comparisons to

```
ruby -e "$(curl -fsSL https://random.server.io/totally/legit/code)"
```

Establish Repositories For Images & Services

- Separate from application code

For Images:

- Dockerfile and related artifacts
- Tests to validate built images
- Information about who maintains image
- Configuration / Parameterization Options
- Build instructions (link to known build job)

📁 cache

📁 deployment

📁 webapp

📄 readme.md

Initial commit

📄 readme.md

threeve.com Deployment

This repository contains all the information needed to build, deploy, and monitor the threeve.com application

Relevant Folders

Folder	Description
deployment	Generic information for deploying the application
cache	The cache service
webapp	The webapp service

Source

 master ▾



threeve.com-deployment / webapp / **php7** /



..

 [docker](#)

 tests

 [readme.md](#)

Initial commit

 **readme.md**

PHP 7 Container Image for threeve.com

This image should extend our standard PHP 7 image, but have the following

- Have the threeve.com code located in /threeve.com

Establish Repositories For Images & Services

For Services:

- Deployment Artifacts
 - Kubernetes pods, ECS tasks, etc.
- Documentation of Interconnectivity
 - Network concerns, File access needs, etc.
- Operational Footprint
 - CPU, Memory constraints
 - Scaling thresholds
- Service Reliability Information
 - How to measure service health

Source

master



threeve.com-deployment / webapp / deployment /



..

k8s

readme.md

Initial commit

readme.md

Deployment information

PHP Engine and Web Engine need to be linked into the same network

Port	Internal / External	Container	Reason
80	External	Apache	Handles web requests
9000	Internal	PHP7	Handles php requests via FastCGI from Apache
9001	External	PHP7	Handles XDebug sessions, if enabled

Source

🔗 master ... | [threeve.com-deployment](#) / [webapp](#) / [deployment](#) / **k8s** /

⬆️ ..

- 📁 [production](#)
- 📁 [testing](#)
- 📄 [readme.md](#) Initial commit

📄 [readme.md](#)

Kubernetes Deployment

Stages

Stage	Purpose
Production	Receives actual threeve.com traffic
Testing	Used for CI/CD testing, torn down often

What files are needed

Service	Purpose
threeve-webapp-image-pull-secret.yaml	Image pull secret for threeve.com webapp container images
threeve-webapp-replication-controller.yaml	Specifies the replication controller for the webapp

Fooville / threeve.com-deployment

Source



master ▾



threeve.com-deployment / webapp / **monitoring** /



..



sysdig



readme.md

Initial commit



readme.md

Automated Builds

For every service and image

- Triggered by changes in:
 - Image/Service repository
 - Linked application code
 - Upstream builds
 - Derivative of image X? Rebuild when X changes!
- Capable of automated deployment
 - Continuous Delivery is container image publishing
 - Continuous Deployment is integration with container orchestrator

Builds Should Embrace CI/CD & Dependencies

Bamboo

My Bamboo

Build ▾

Deploy ▾

Reports ▾

Create ▾

Search

?

⚙

▾

Build projects

Docker


Project wallboard

⋮

Plan	Build	Completed	Tests	Reason	
csslint	✓ #3	4 minutes ago	No tests found	Child of DOC-NOD-2	▶ ✎ ☆
doxygen	✓ #1	1 hour ago	No tests found	Manual run by Michael Venezia	▶ ✎ ☆
nodejs	✓ #2	4 minutes ago	No tests found	Changes by Michael Venezia <mvenezia@gmail.com>	▶ ✎ ☆
PHP-FPM	✓ #7	2 minutes ago	No tests found	Changes by Michael Venezia <mvenezia@gmail.com>	▶ ✎ ☆

Continuous integration powered by Atlassian Bamboo version 5.12.2 build 51212 - 31 May 16

[Report a problem](#) · [Request a feature](#) · [Contact Atlassian](#) · [Contact Administrators](#)



Automated Builds

- Make it easy for someone *else* to build and deploy your service
 - Security Vulnerability
 - Hit by bus
- Runs tests to validate result
 - Run tests on container image before pushing to registry
 - Push to registry
 - Deploy to test environment
 - Perform end to end tests
 - Promote to production

Automated Builds

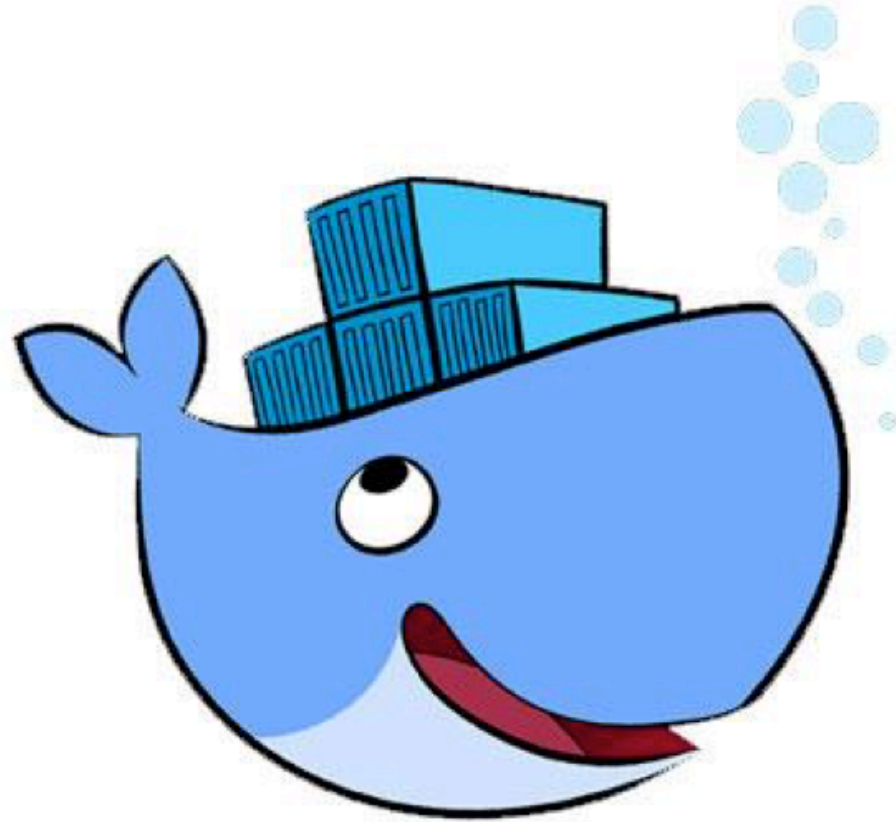
- Integrated ticketing system provides complete visibility
 - What tickets were included in build
 - What tickets are closed with deployment
 - What tickets are reopened with rollback
- Need Prune Policy
 - If automated builds are happening, container images will increase without bound
 - Need eviction policy so only N amount of images are kept around

Publish Services and Images to Catalog

- Image discovery still a bit problematic
 - Search works, but so what?
- Provide centralized list, presumably in a Wiki
 - Images and Services available
 - Which team is maintaining them
 - Build link
 - Repository Link
 - Example(s) of Image or Service being used

Agenda

- Containers Bring Change
- An Approach
 - Required Software
 - Processes
 - **Cultural Changes**
- Additional Concerns
- Lessons Learned



Images and Services Can Be Reused

- All too often teams do not think of reusability of image or service
 - Yet vast majority of images extend from an existing image

Container Images: Analogous to OOP

- Think of images as classes in OOP
 - Configuration variables
 - Serve a specific purpose
 - Can either be:
 - Delegated to, like a sidecar container
 - Enhanced, like a ruby image with application code burned in
- Use judgement to decide seams
 - Ask how could you sell this image to someone else

Embrace Internal Open Source

- Side effect of Service / Image repositories: Forking
- Allow other groups to fork / pull request updates
- Reduces container variants while also reducing maintenance
 - Easier for Information Security to audit
 - One group has to maintain container image, all groups benefit
- Allows for transfer of control
 - Group may shift away from python, but other groups may want to continue it

Have Open Source Strategy

Have a great container image? Shouldn't it be on Docker Hub too?

- Good Karma
- Increases visibility / free advertising!
- Maybe even reduced maintenance

Embrace CI/CD

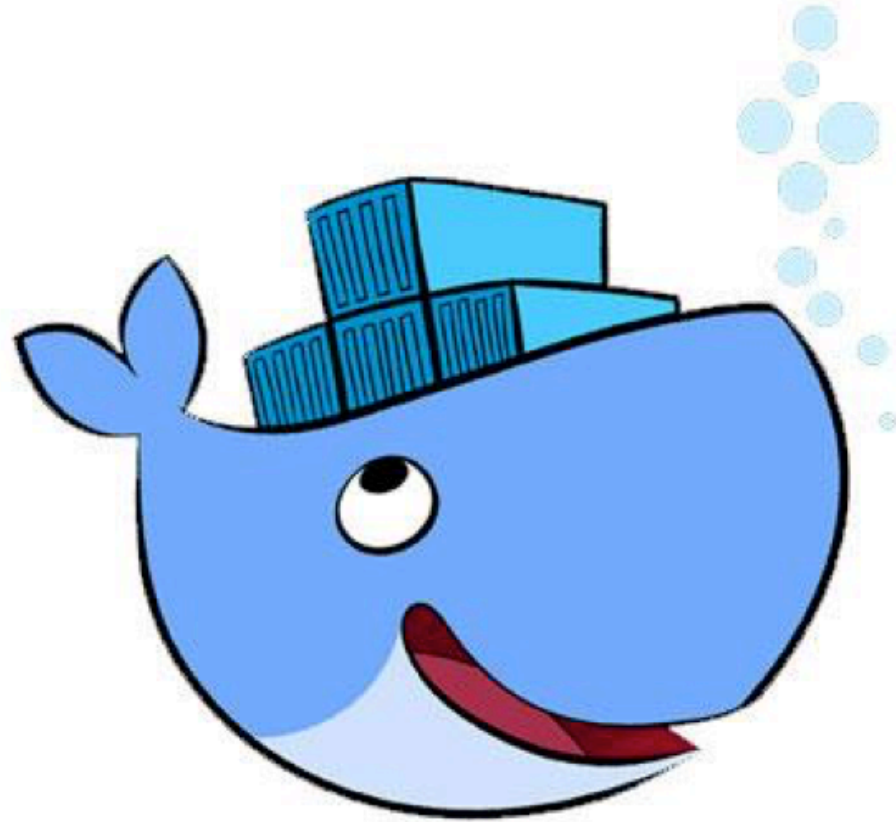
- Containers lend well to CI/CD
 - Immutable environments
 - Repeatable events
- May be the bait needed to convince groups to embrace CI/CD

Encourage Ownership

- Containers allow for any developer to pick any technology
- Processes enforce developer to make technology sustainable
 - Documentation of deployment
 - Build jobs
 - Tests
- Fosters innovation within organization

Agenda

- Containers Bring Change
- An Approach
 - Required Software
 - Processes
 - Cultural Changes
- **Additional Concerns**
- Lessons Learned



Additional Security: Signed Images

- Available through Docker and rkt
- Allows centralized authorization
- Not very prevalent
 - Use is increasing
 - Can help convince some groups regarding security or fidelity

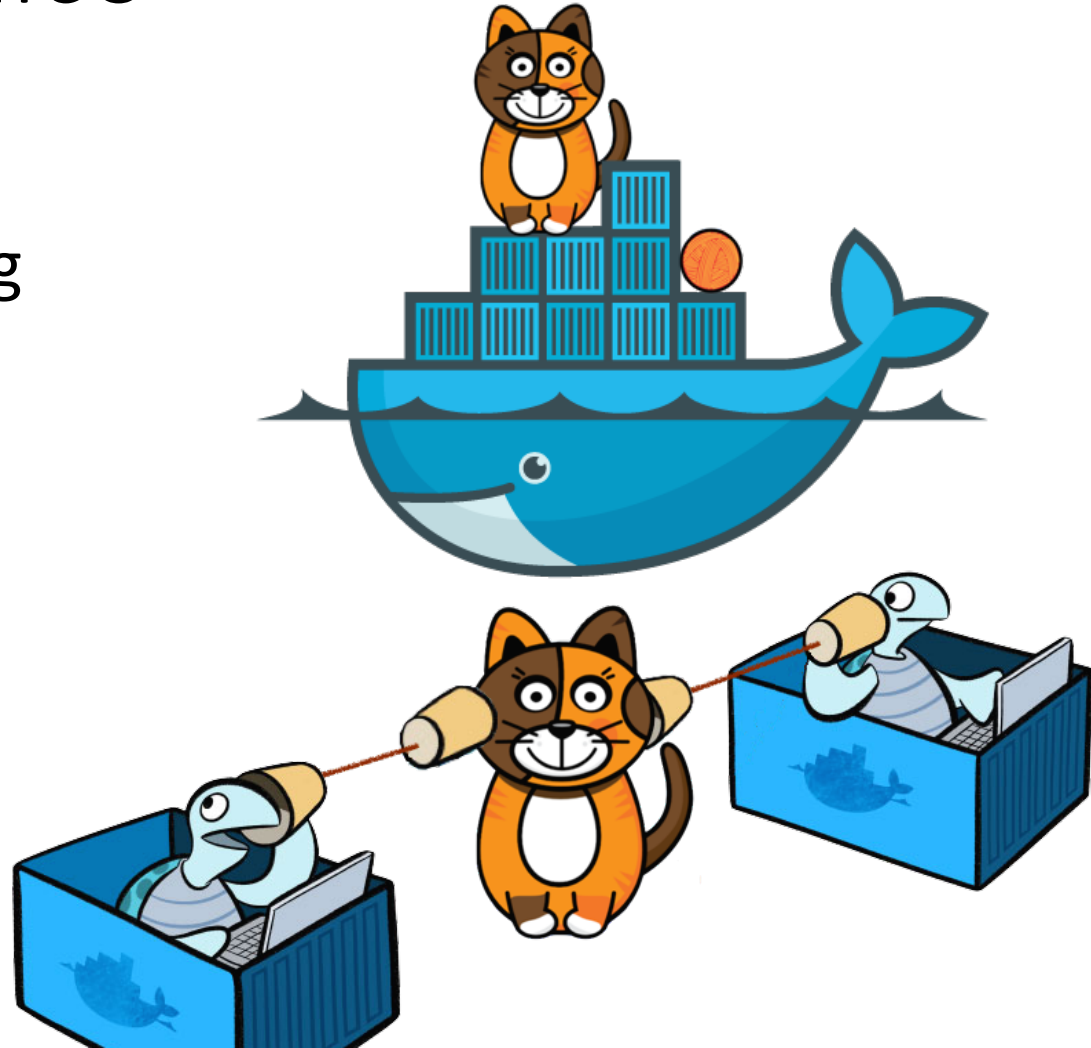


Software-Defined Firewalls

- Desire:
 - Firewalls similar to traditional cloud-provided firewalls
 - Have auditability
- Problem:
 - Container Orchestrators tend to schedule containers in a rather fluid environment
 - Some container environments hide the underlying cloud almost completely
 - Kubernetes with network overlay

Solution: Project Calico

- Allows engineering team to define firewalls within existing artifacts
- Easily Auditable
- Performs well

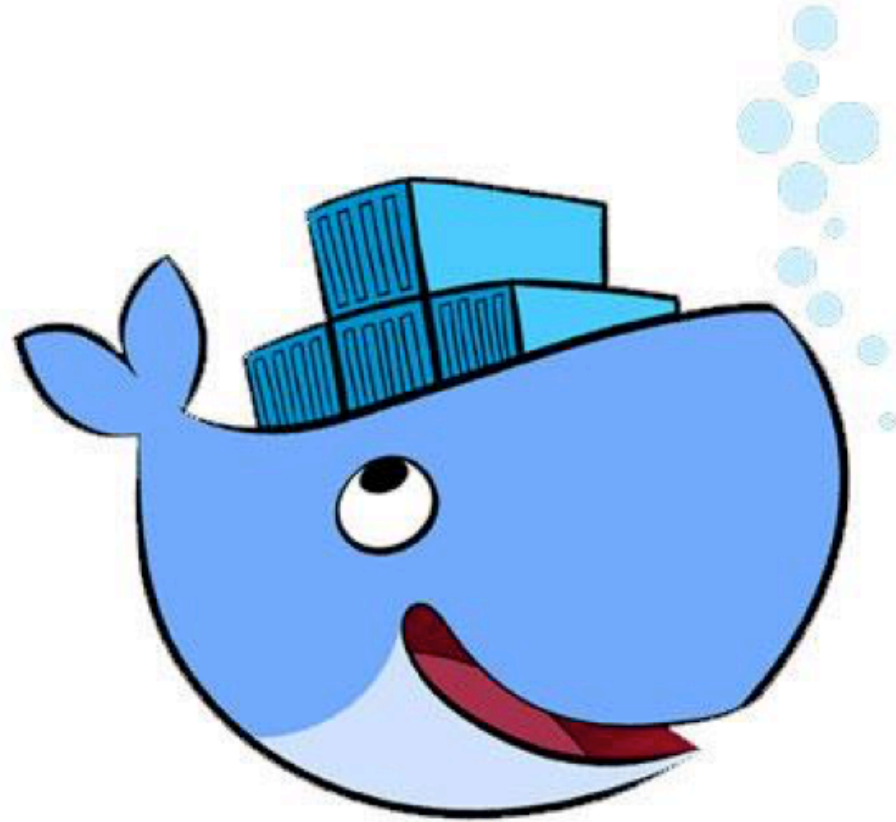


Embrace Container-Native Monitoring

- Traditional monitoring changes when moving to container orchestration
- Good time to re-evaluate approach and desired outcomes from monitoring
 - Is service and VM monitoring one in the same currently?
 - Are you already using projects like statsd or Prometheus?
- Providing a good monitoring tool can help ease transition into cloud-native computing

Agenda

- Containers Bring Change
- An Approach
 - Required Software
 - Processes
 - Cultural Changes
- Additional Concerns
- **Lessons Learned**



Not Everyone Gets It, Or You

- Be prepared for differences of opinion on what things are
- Who's right?
 - Single container and service image of MySQL, Apache, PHP, Wordpress
 - Two services
 - MySQL
 - Web Service
 - Apache
 - PHP + Wordpress

Adjustments Are Always Needed

- What works for one organization may not work for another
- Be open to changes if organization desires it
 - Better for a subpar agreement than no agreement



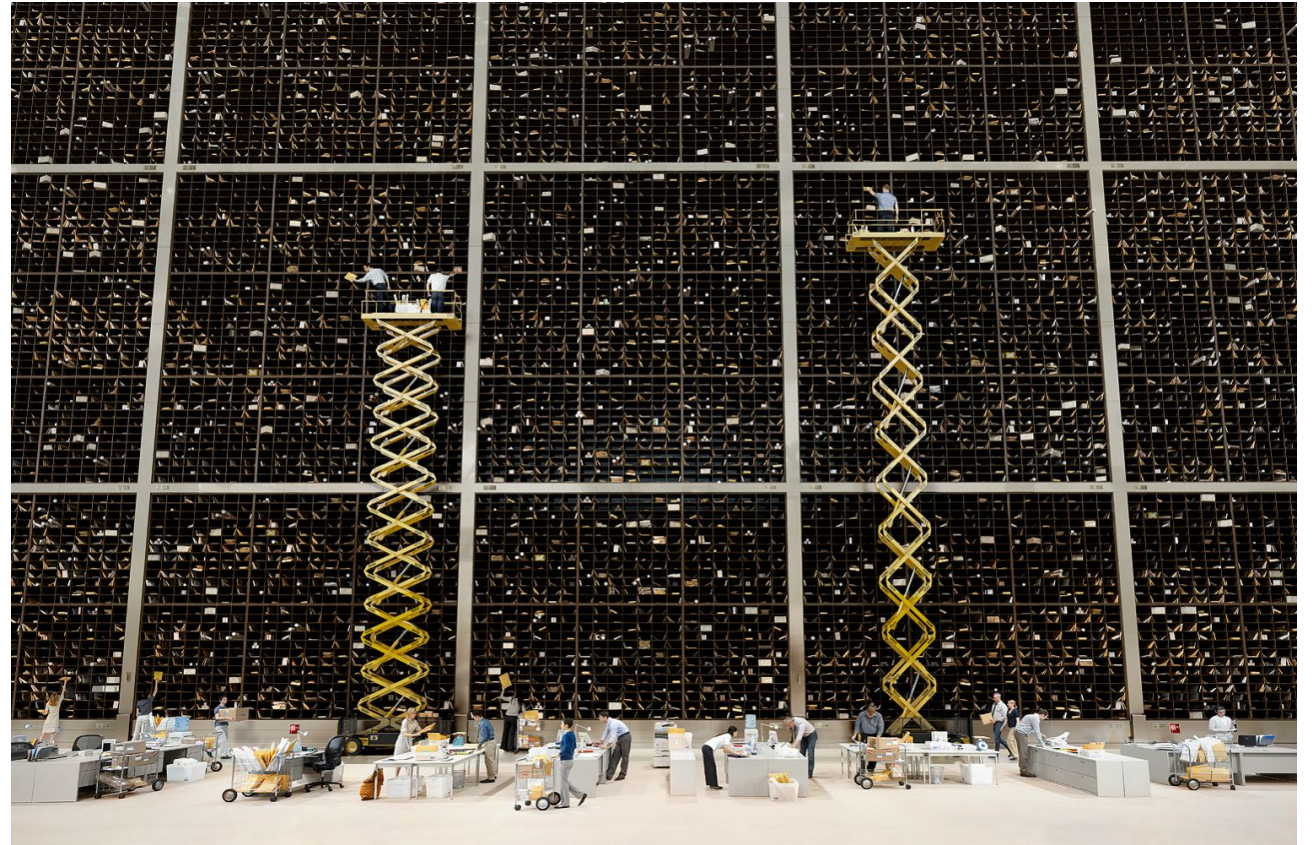
Not Everyone Will Share, Nor Look

- People may not create repositories, build jobs, publish services to central list
 - May not see value
- Others may never look to see if existing container images exist



Too Much Work

- Creating repositories and documentation may seem like too much work
- Containers are supposed to be fun, this seems like a drag



Don't Give Up

- Change is hard
- May take multiple attempts to gain traction
- See what works, what does not, and adjust



Thanks!

- Any Questions?

- mvenezia@gmail.com
- <https://github.com/venezia>

Example of outdated role: Repo Maintainer

- In traditional VM world, large enterprises often standardize on:
 - Single Linux distribution
 - Single private package repository
 - Often with limited versions of any given package
- In container-native environment:
 - No single Linux distribution
 - Each individual/group creates packages
 - No need for private packages
 - But now new people need to learn similar practices



Consequences of No Central Repo Maintainer

- Packages everywhere!
 - Every group creates own nginx variant
 - Who's monitoring for vulnerabilities?
- Nightmare for Compliance / Information Security
 - Typically easier when single authority of creating packages
 - Horse has left the stable, need to adjust thinking

