Containers for Grownups: Migrating Traditional & Existing Applications

Almost all software was designed before containers

Scott McCarty

Container Strategist

Why Migrate

Container Journey

Evaluate Technology

Experiment

Quick Win

Inventory Applications

Determine Technology

Containerize

The Journey

Single Node

Traditional Development

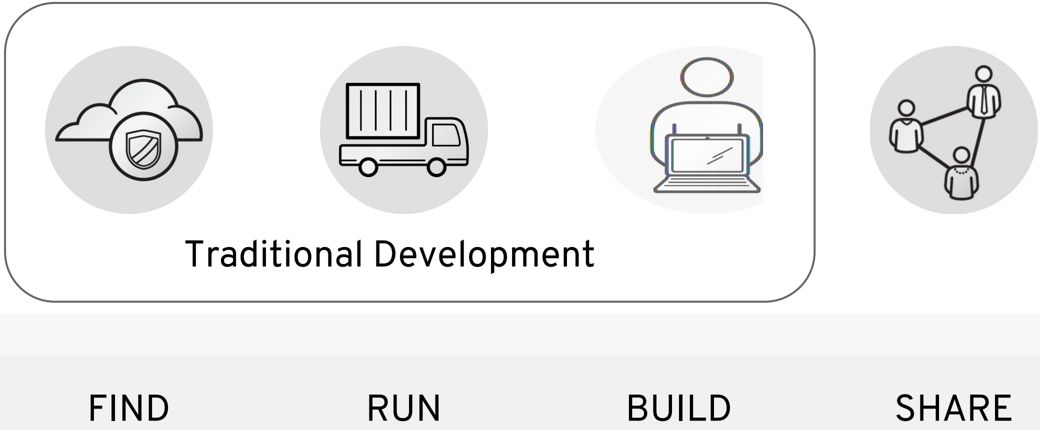
FIND

RUN

BUILD

The Journey

Single Node+



Traditional Development

FIND

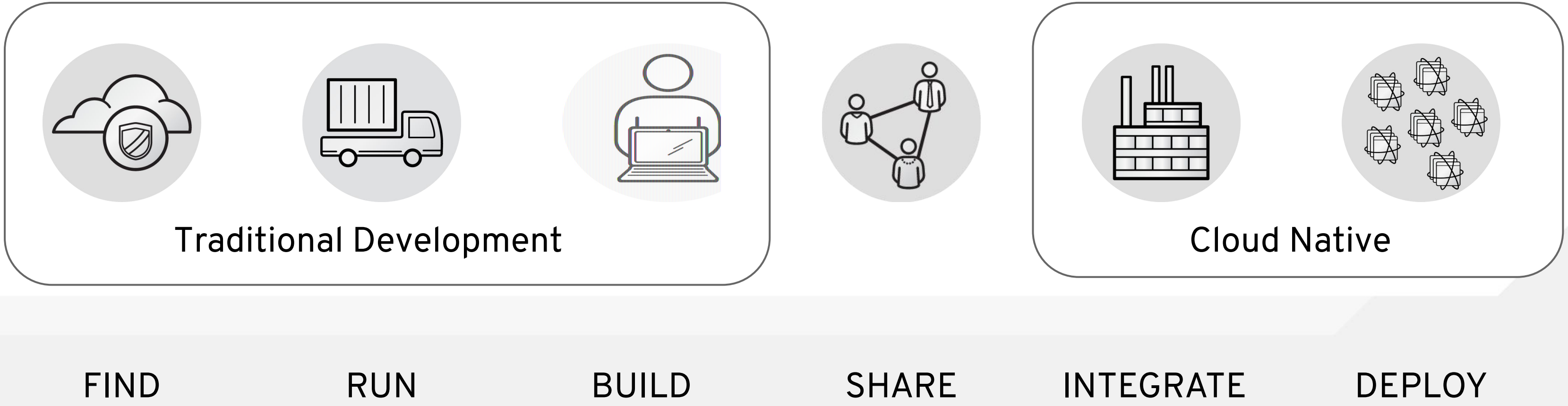
RUN

BUILD

SHARE

The Journey

Multi Node



Traditional Development

Cloud Native

FIND

RUN

BUILD

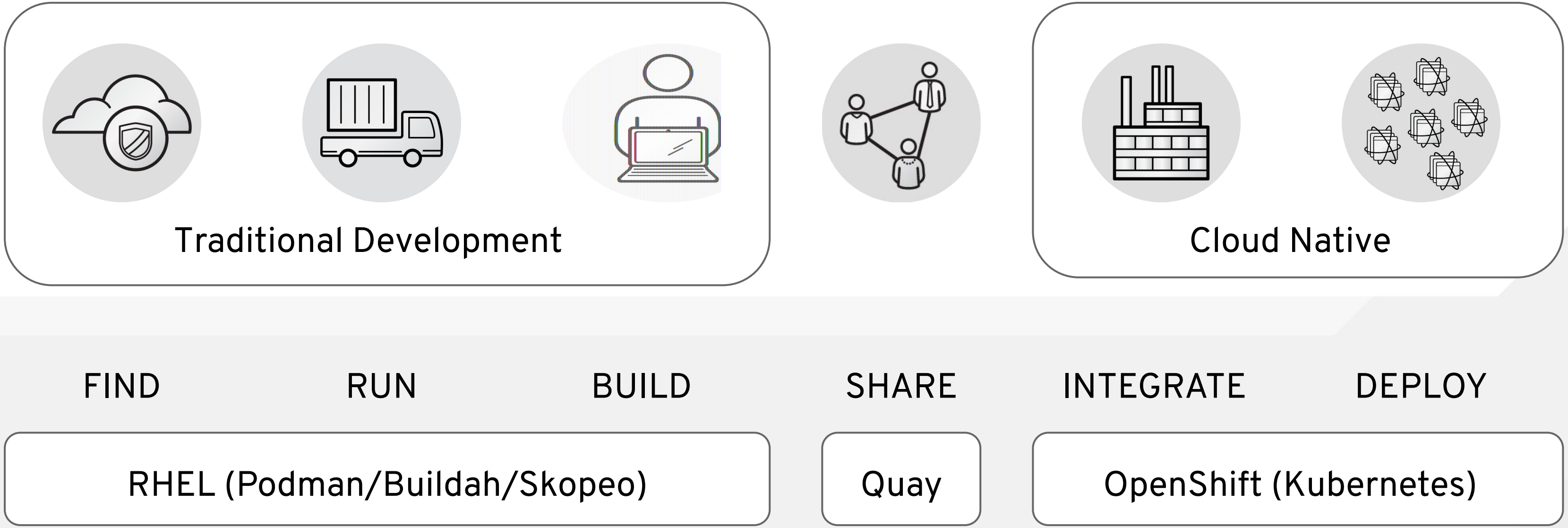
SHARE

INTEGRATE

DEPLOY

The Journey

Can start anywhere



Traditional Development

Cloud Native

FIND

RUN

BUILD

SHARE

INTEGRATE

DEPLOY

RHEL (Podman/Buildah/Skopeo)

Quay

OpenShift (Kubernetes)

Customer Needs

Mapping customer needs to solutions

Capability	Technology	Product
Single Node	Linux & Container Tools	Red Hat Enterprise Linux
Multi Node	Linux & Kubernetes	OpenShift

There are Different Options

Each has a different level of effort

LIFT & SHIFT

Move the application as-is. Do the minimal amount necessary to containerize

REFACTOR

Leave much of the application as-is. Build new strategic portions.

REWRITE

Start from scratch - adopt the latest software development patterns (microservices, etc)

The Problem

Almost all applications, even new ones, are built on technologies that were designed and developed before Linux Containers.

Image: Cargo Designed for a Ship from 1921

Gaining migration skills will help your team make good strategic decisions about what should and should not be containerized.

The solution

Learn to Migrate

redhat.

LISTEN TO YOUR APPLICATION

We have seen this before...

UNIX TO LINUX

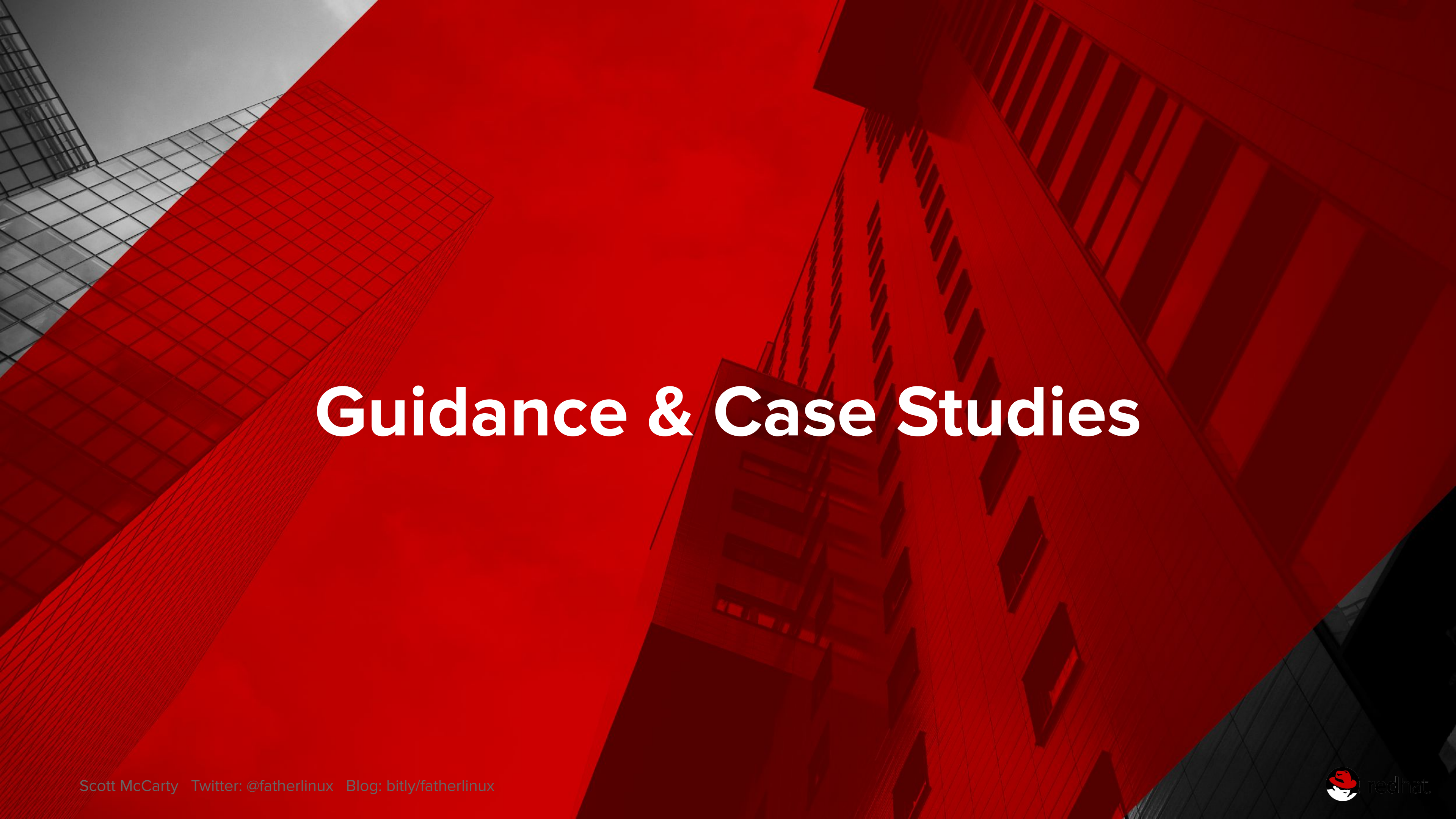
Will my app run?

PHYSICAL TO VIRTUAL

Will my app run?

APPS TO CONTAINERIZED APPS

Will my app run?



Guidance & Case Studies

APPLICATION CHARACTERISTICS

These are the things you need to think about..

Architectural

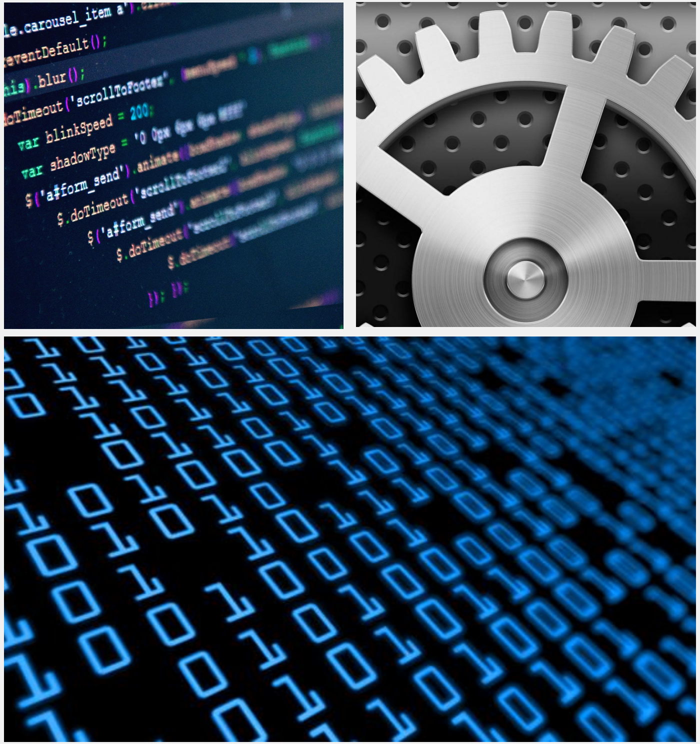
Security

Performance

Migration from process isolation to containers

Architecture

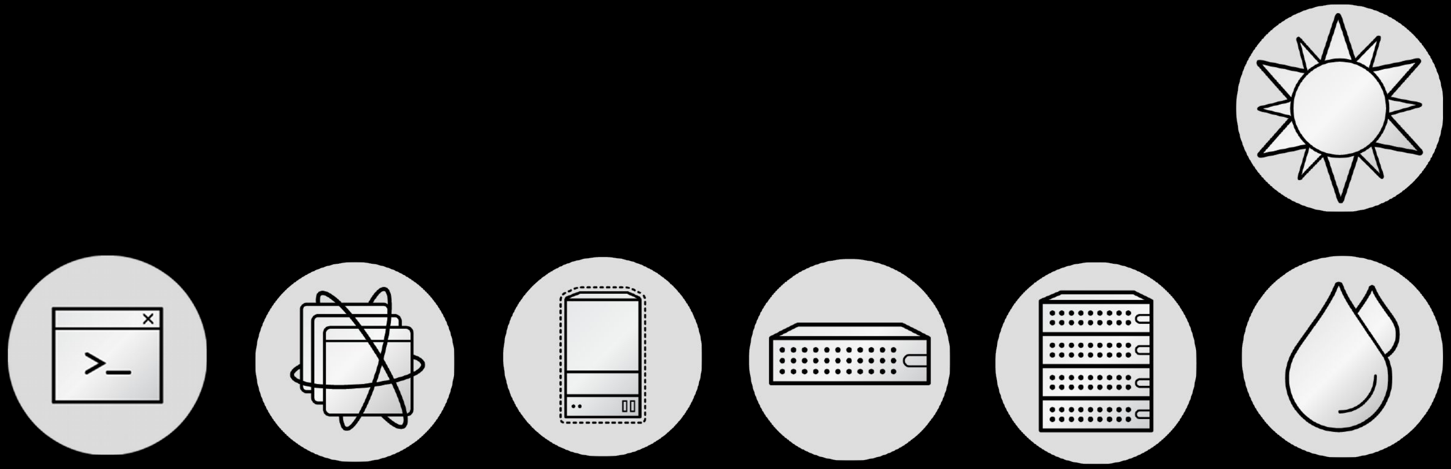
1. Code: mysqld
2. Configuration: /etc/my.cnf
3. Data: /var/lib/mysql
4. Other stuff :-)



Level of Effort

	EASY	MODERATE	DIFFICULT
Code	Completely Isolated (single process)	Somewhat Isolated (multiple processes)	Self Modifying (e.g. Actor Model)
Configuration	One File	Several Files	Anywhere in Filesystem
Data	Saved in Single Place	Saved in Several Places	Anywhere in Filesystem
Secrets	Static Files	Network	Dynamic Generation of Certificates
Network	HTTP, HTTPS	TCP, UDP	IPSEC, Highly Isolated
Installation	Packages, Source	Installer and Understood Configuration	Installers (install.sh)
Licensing	Open Source	Proprietary	Restrictive & Proprietary
Recoverability	Easy to Restart	Fails Sometimes	Fails Often

The Tenancy Scale



Process

Container

Virtual Server

Physical Server

Rack

Data Center

Performance Characteristics

	Bare Metal	+Containers	+Virtualization
CPU Intensive	Fast	Fast	Fast
Memory Intensive	Fast	Fast	Fast
Disk I/O Latency	Fast	Fast	Medium
Disk I/O Throughput	Fast	Fast	Fast
Network Latency	Fast	Fast	Medium
Network Throughput	Fast	Fast	Fast
Deployment Speed	Slow	Fast	Medium
Uptime (Live Migration)	No	No	Yes
Alternative OS	Yes	Some	Yes

Use Cases & Demos

Inventory: Large Scale Web Applications

PROBLEM

- WordPress websites needed at large scale
- Individual server doesn't scale with load
- Multi-server states hard to synchronize
- Development environment difficult to replicate

SOLUTION

- Containerized service
- Site code distributed in images
- Single-service containers are easy to swap
- Development & production built from images

THE RESULT

- Enterprise-level - ~8,800 sites and ~29,000 users
- Quickly scalable, cloudburst-able
- Interchangeable front-end containers = automated sites with custom DNS and HTTPS
- Developers have taken ownership of deployment, can update, roll-back, clone w/o sysadmins

Inventory: Red Hat IDM in a Container

PROBLEM

- Dozens of daemons, their own data volumes
- Libraries share config files
- Installer logic expects single machine
- Knowledge embedded in init/systemd
- Extensive initial setup needed

SOLUTION

- Build IDM (aka FreeIPA) in single container
- Simplified install, upgrade, rollback
- Minimize data volumes
- Template based data volume population

THE RESULT

- Better software delivery
- Better technical demarcation between vendor and customer
- Extremely portable
- Environments can be setup, tested, torn down in minutes
- Same pattern for other products and services: OpenShift, OpenStack (yes, OSP), sssd, etc.

Inventory: Tools

PROBLEM

- Can’t install software on prod servers
- Can’t introduce risk on prod servers
- Can’t access tools quickly

SOLUTION

- Build tools containers
- Stage tools containers in registry server
- Completely self contained

THE RESULT

- Use tools and delete them easily (rhel-tools)
- Scan network
- Instrument kernel
- Use any tool you like

Remember, Containers Are Just Child Processes

Questions?

Further Reading & Citations

- Do Linux distributions still matter with containers? https://goo.gl/t4LLWw
- A Practical Introduction to Docker Terminology: http://red.ht/2beXHDD
- Container Tidbits: When Should I Break My Application into Multiple Containers? http://red.ht/22xKw9i
- Architecting Containers Part 4: Workload Characteristics and Candidates for Containerization: http://red.ht/1SBw9ql
- Supply Chain Demo on GitHub: http://bit.ly/2aY1WEO
- The New Stack: Container Defense in Depth: http://bit.ly/2buXflB
- Architecting Containers Series: http://red.ht/2aXjVJF
- Red Hat Connect for Technology Partners: https://connect.redhat.com/

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos

FUTURE OPPORTUNITIES

- Do Linux distributions still matter with containers? https://goo.gl/t4LLWw
- Let Red Hat help you analyze your application portfolio http://red.ht/2ic4TX3
- Check out Architecting Containers Series http://red.ht/1SBw9ql
- Learn more about Red Hat Summit at http://redhat.com/summit