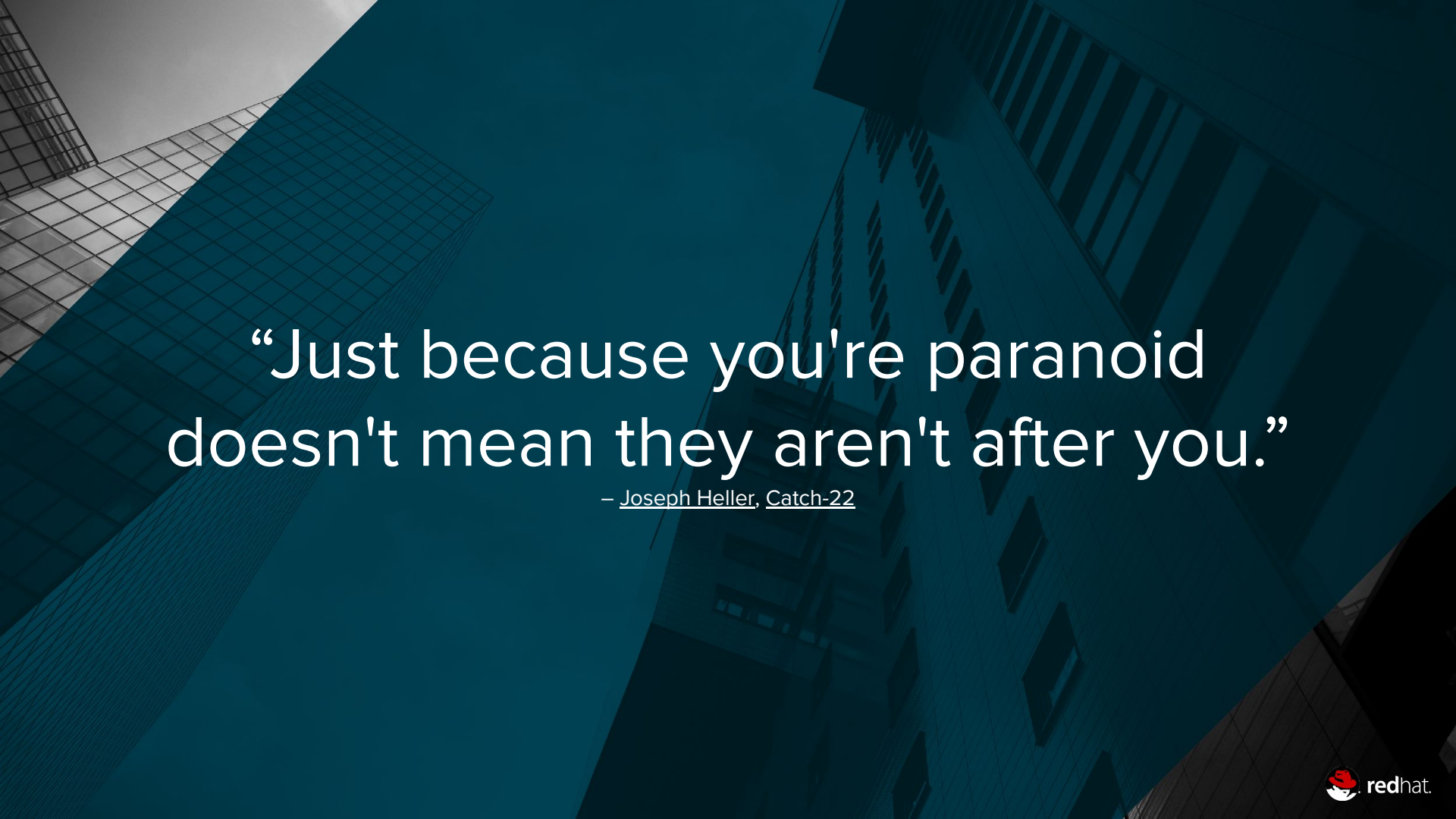
redhat.

The security implications of running software in containers

Taming Container Fears

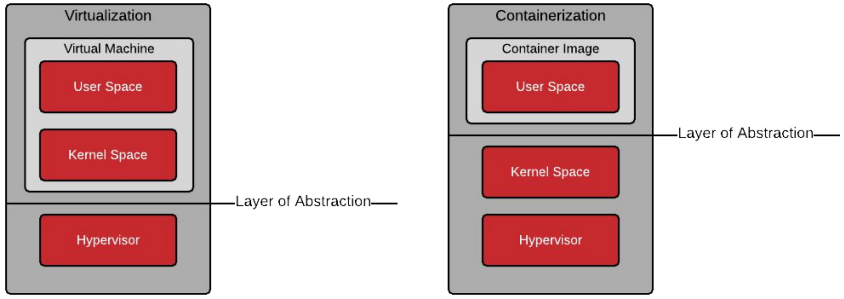
Scott McCarty
Principal Product Manager, Containers - RHEL & OpenShift
05/14/2019


“Just because you're paranoid doesn't mean they aren't after you.”
– Joseph Heller, Catch-22
redhat.

THE PROBLEMS

CONTAINERS DON'T CONTAIN

Dan Walsh (my shirt is dedicated to you)



Move the kernel around or move the user space around

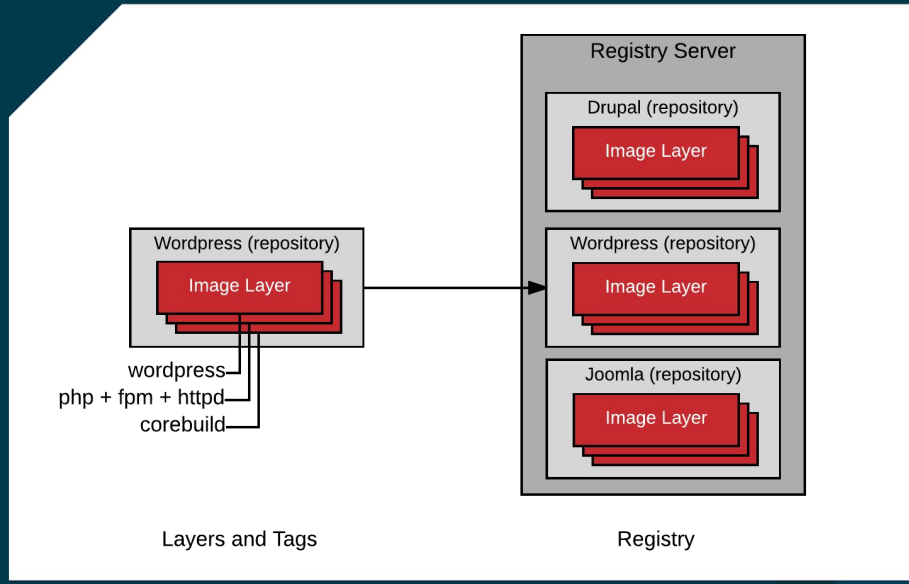
- Fancy processes
- Breaking the OS in two pieces
- All containers share a kernel
- Root only exploits can be ba'a'a'a'ad

CONTAINER IMAGES

Currency for collaboration

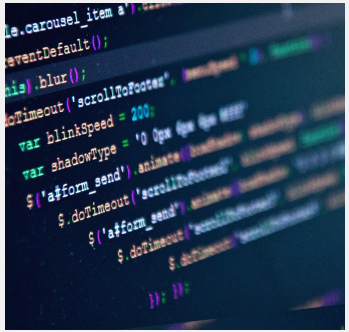
Developers, operations, middleware, performance, and security specialists all have a role to play

- Fancy files
- Who controls what?
- Who is responsible for what?
- What about bad content?



Hard Work

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




NEW CONCEPTS

CIA

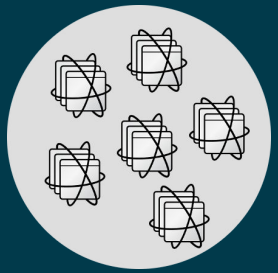
Not them, but yeah, they might be after you too....

CONFIDENTIALITY

Has data leaked from the container platform?

INTEGRITY

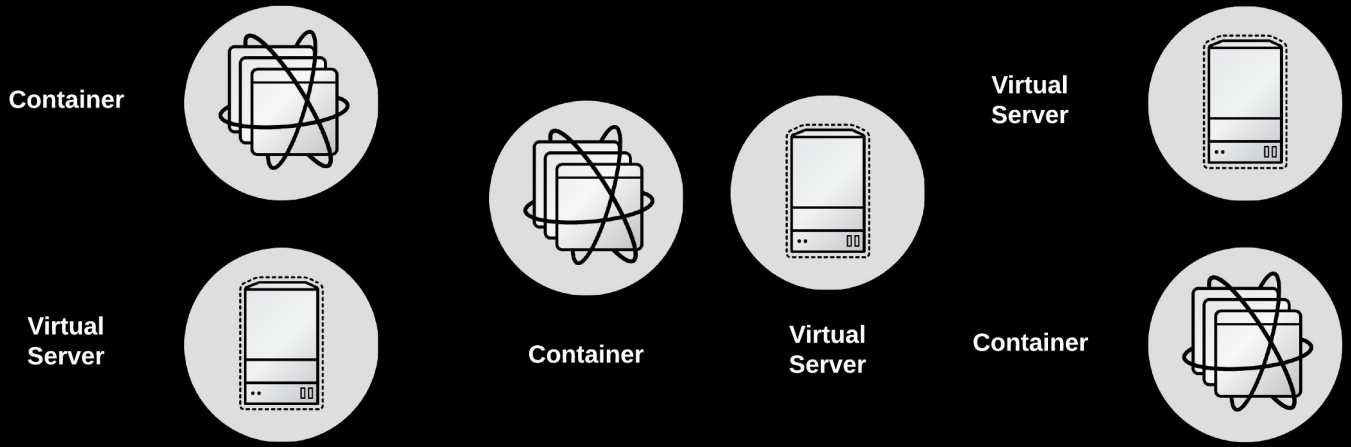
Has somebody tampered with the container?



AVAILABILITY

Is the container up and running?

Integrity

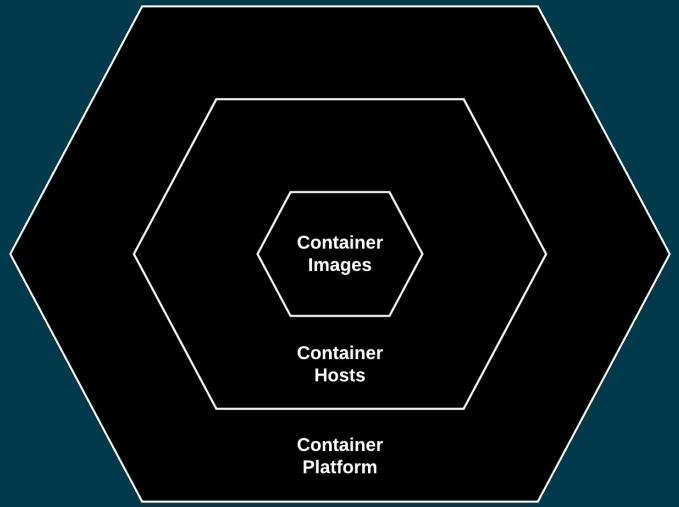

Container
Virtual Server
Container
Virtual Server
Virtual Server
Container

Defense in Depth

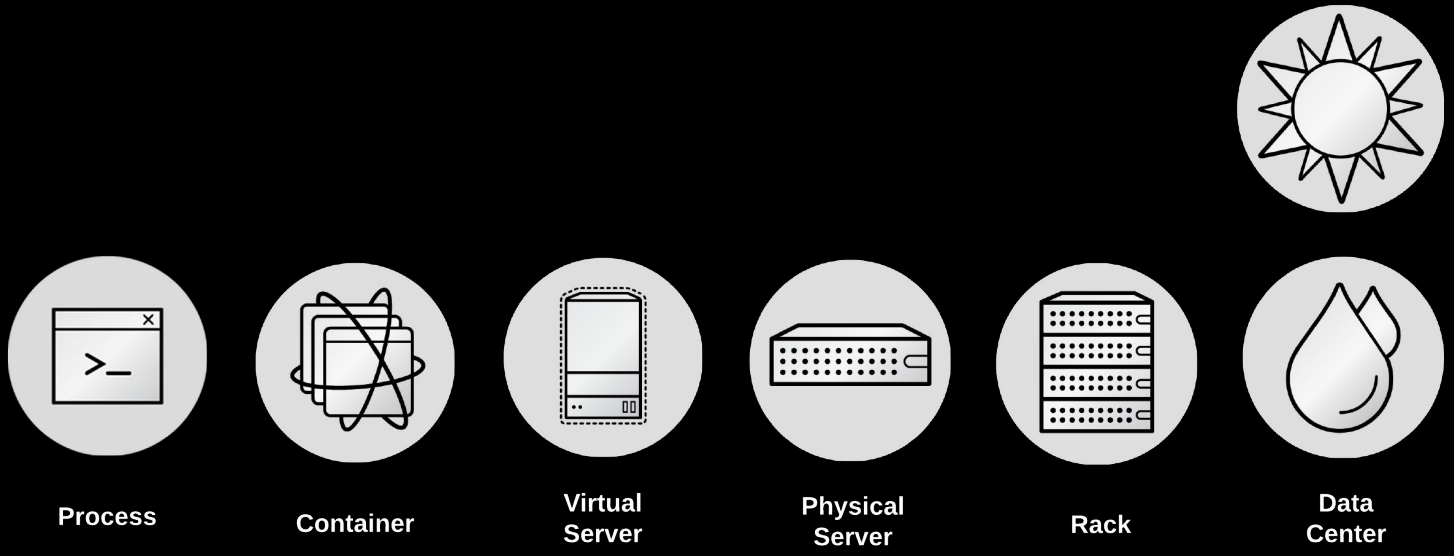
the practice of arranging defensive lines or fortifications so that they can defend each other, especially in case of an enemy incursion.

Can we harden each layer?

- Image scanning, signing, and blueprinting
- Container host hardening
- Platform delegation practices



The Tenancy Scale



Security Controls

SELinux

Who you can talk to. Which objects in the kernel can communicate with other objects.

SECCOMP

What you can say. Limiting system calls is like limiting what words can be said


NEW TECHNICAL CONTROLS

CONTAINER IMAGES

Our current operating model controls:

- Trusted Content (What's in the container matters. Don't install from hackme.com.)
- Content Provenance (Track who changed what.)
- Security Scans
- Remediation/Patching
- Bill of Materials
- CVE Databases
- Security Response Teams
- Limit Root Access (Don't oversell User Namespaces.)
- Limit User Access (Who controls content.)

Containers add the ability to easily apply techniques such as:

- Bill of Materials
- Signing
- Read-only Containers (Read-only servers were popular in the late 90s.)
- Podman diff to see what changed in a container

CONTAINER HOST

Many of these techniques, we apply today.

- Kernel Quality
- Capabilities
- Read Only Images
- Limiting ssh access (root access and users)
- Well understood/controlled configuration (cloud-init, Ansible)
- Tenancy

Since containers are just fancy processes with a well-controlled user space, it's easier to apply techniques like:

- SECCOMP
- sVirt
- Hardening: NO_NEW_PRIVS, Read Only Images, –cap-drop=ALL, –user=user

CONTAINER PLATFORM

This layer exists in the world of physical and virtual servers but is typically an administrator only tool, such as vCenter or HPSA. In the world of containers, it's much more common to delegate some access to developers, architects, and application owners.

- Role-Based Authorization
- Authentication (LDAP, network level access/restriction to the platform)
- Environment Isolation (development, testing, production)
- User Demarcation (kubectl exec)
- Network Separation
- Key Management

STANDARD WEB APPLICATION

Many security controls are inconvenient

Benefits

- Network firewall (possibly layer 7)
- Host based firewall
- Kernel quality
- CVE database
- Well understood tenancy
- Understood remediation/patching
- Security scanning

Limitations

- Tripwire, SELinux, SECCOMP usually disabled
- Mutable user space
- No temporal understanding
- No spatial understanding (code, configuration, data)
- No platform delegation granularity
- Not patched often

CONTAINERIZED WEB APPLICATION

Many security controls are essentially free

Benefits

- All tools from standard web application
- Read only containers
- Signing
- Platform delegation
- Spatial and temporal understanding of containers and application
- Updates practiced more

Limitations

- Tenancy not well understood
- Shared kernel
- Applications hard to break up into code/configuration/data
- More infrastructure (platform and management)
- Need better understanding of applications

Questions?

redhat.

redhat.

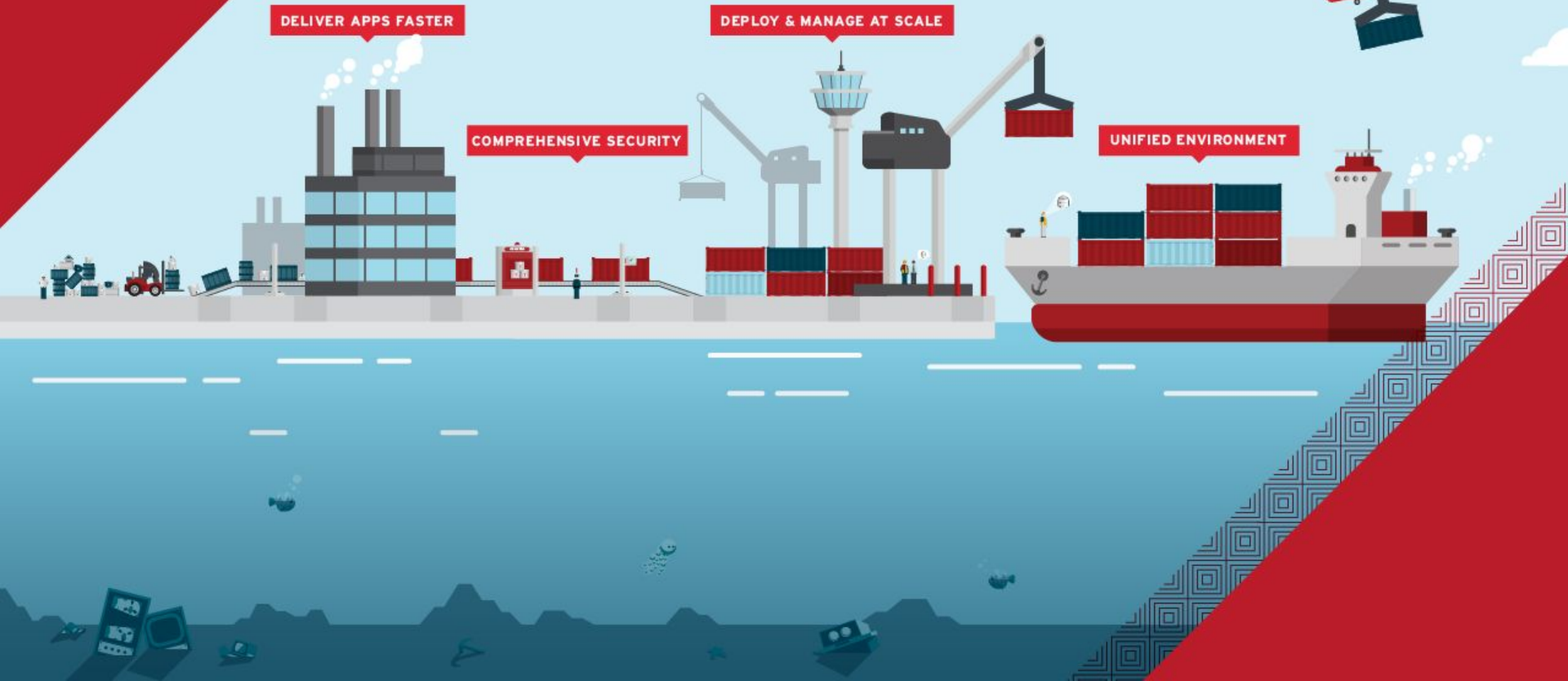
Citations

- 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
- A Practical Introduction to Docker Terminology: http://red.ht/2beXHDD
- Confidentiality, Integrity, Availability: http://bit.ly/2bcStO9

By Scott McCarty @fatherlinux

redhat.

CONTAINERS FOR THE ENTERPRISE



redhat.

THANK YOU

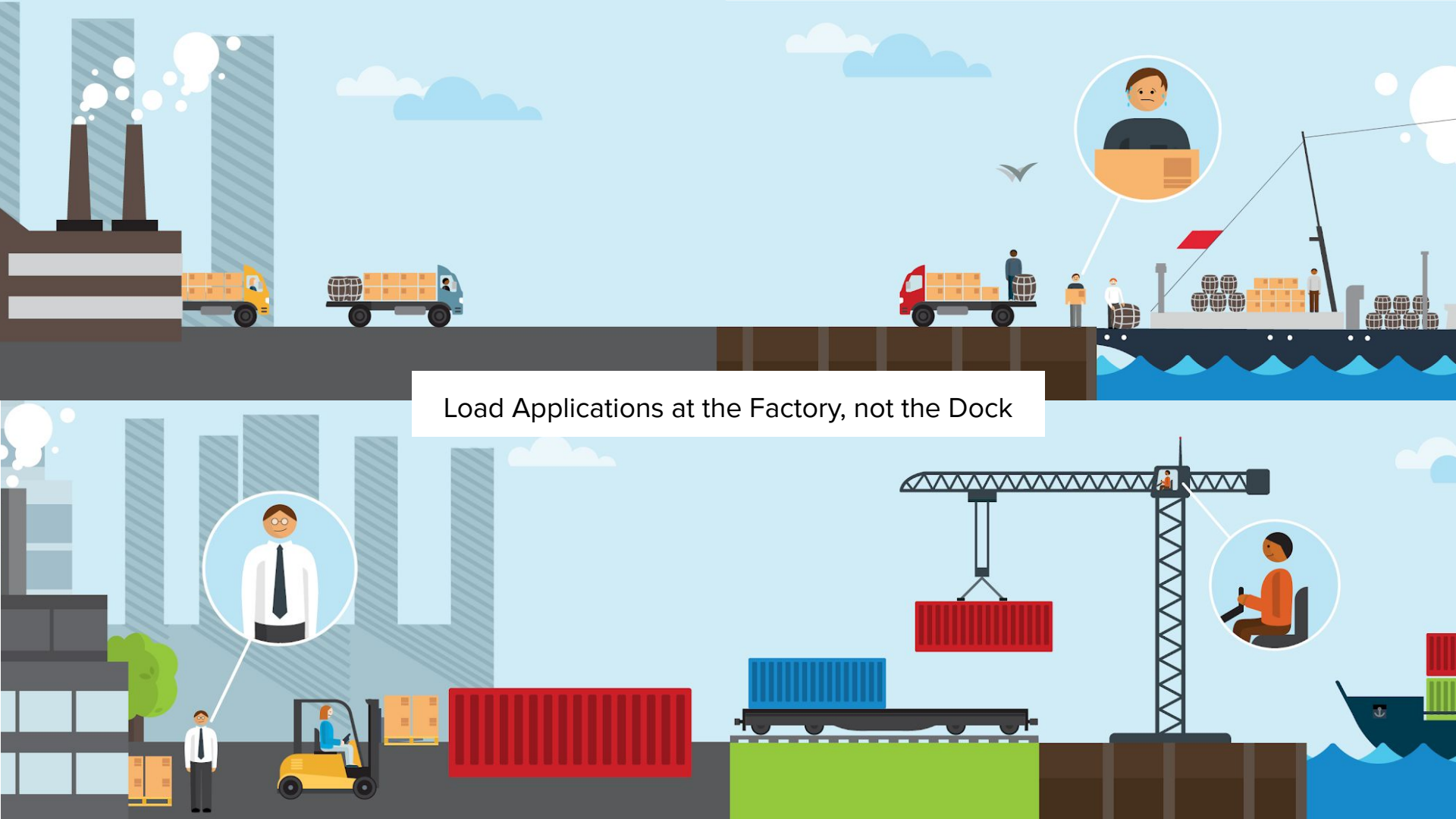
plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos


Load Applications at the Factory, not the Dock