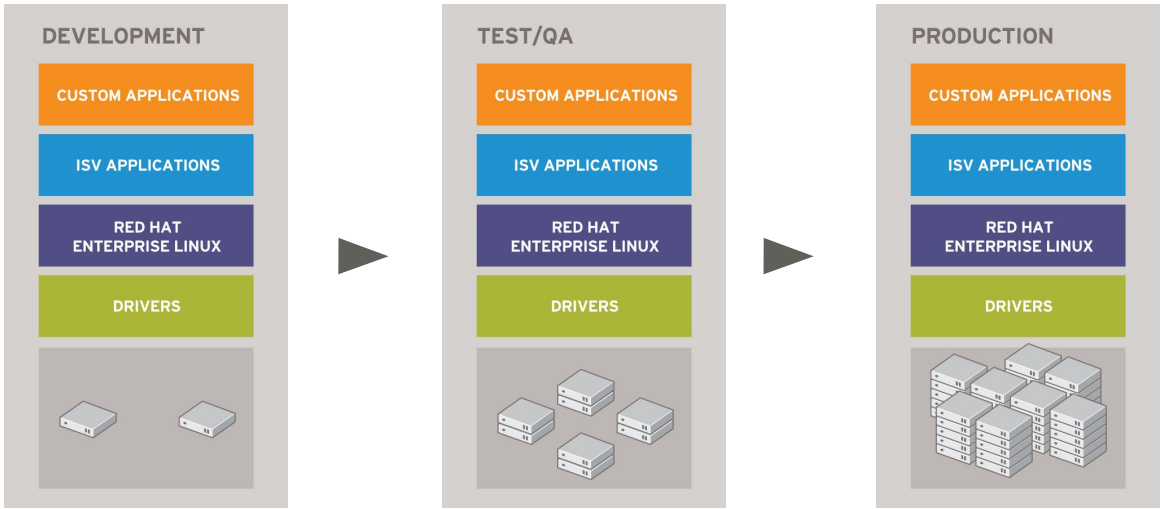
LINUX CONTAINERS 101

Andreas Neeb
Chief Architect Financial Services Vertical

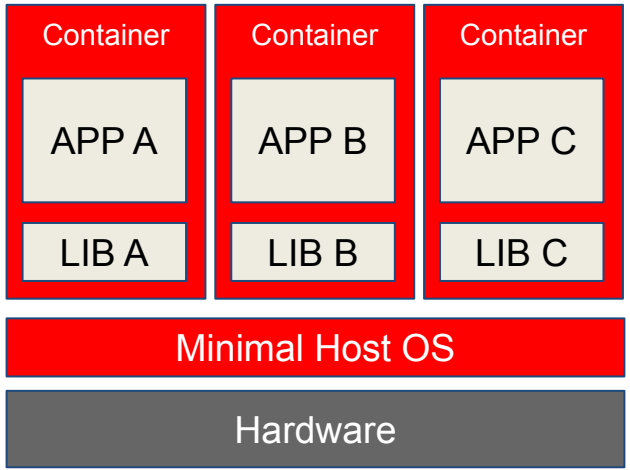
What are containers and why do you need them?

Containers are a solution to the problem of how to get software to run reliably when moved from one computing environment to another.



How do containers try to solve this problem?

Put simply, a container consists of an entire runtime environment: an application, plus all its dependencies, libraries and other binaries, and configuration files needed to run it, bundled into one package.



What are container images?

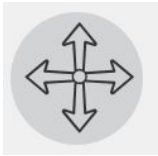
Container images are runnable packages that contain your applications and their dependencies. They are lighter than virtual machine images and can be layered with other Container images to re-use common content.

Isolated

Applications run in containers with isolated memory, file-system, and networking resources for maximum stability and security.

Lightweight

Containers include only the minimal runtime requirements for the application, reducing size and simplifying maintenance.

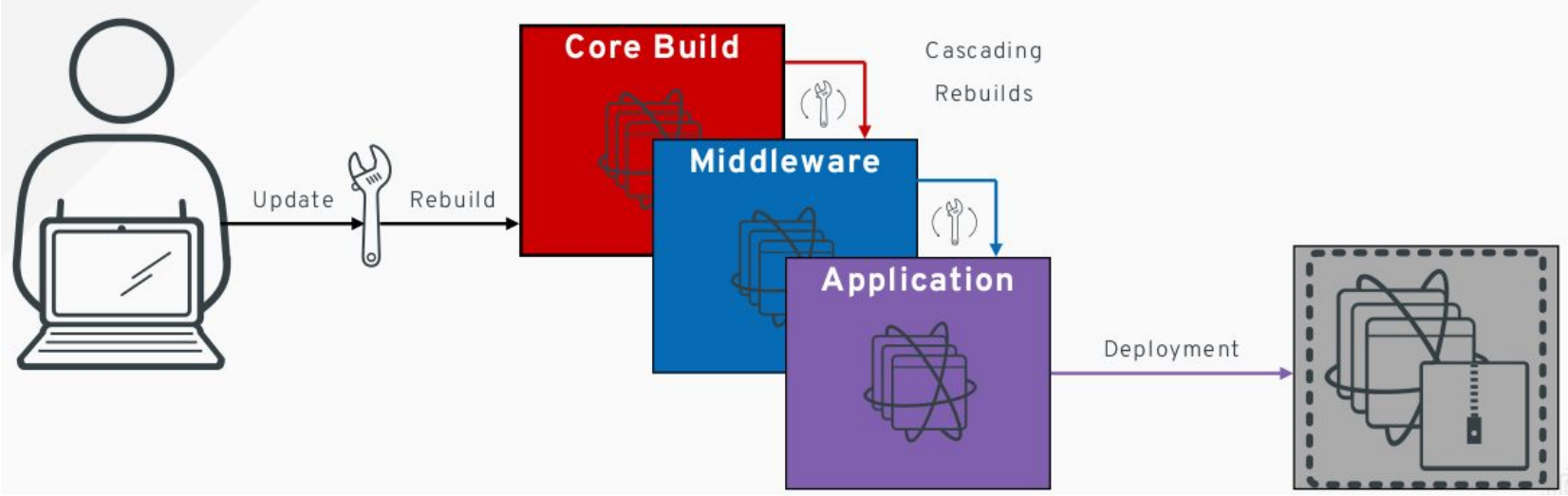


Portable

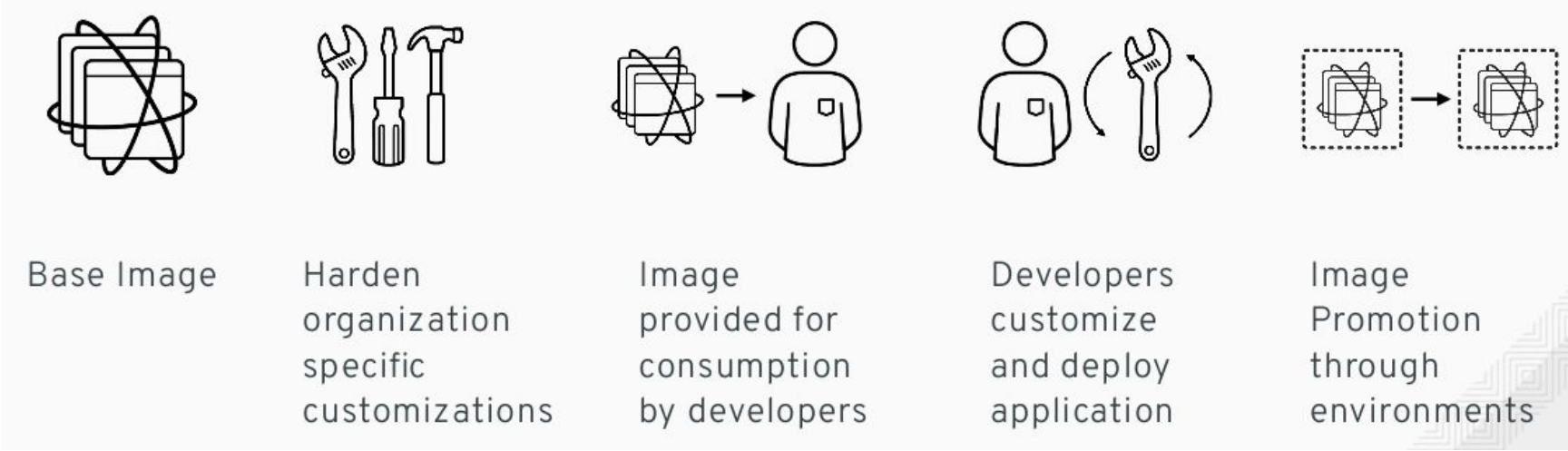
Move applications and all of their runtime requirements across systems.

Container are immutable

Never change any part of your system once it is deployed. If you need to change it, rebuild and deploy a new system.



Typical image workflow



Containers bring Big Wins for Developers

- Lightweight, Encapsulated OS abstraction - carry your OS with you
- No more waiting 3+ weeks for a VM to be provisioned by Ops just so you can run a series of tests
- Getting Started instantly (docker run -it rhel/eap7)
- Dev Environments that more closely match Prod Environments
- Dev Environments that match OTHER Dev Environments

Containers bring Big Wins for Operations

- Lightweight footprint and minimal overhead, fast provision, start and decommission
- High density, especially when mixing different type of workloads
- Application as the unit of management
- Portability across machines and environments
- Infrastructure as Code/Text → Version Control, Automation

For a Java Developer

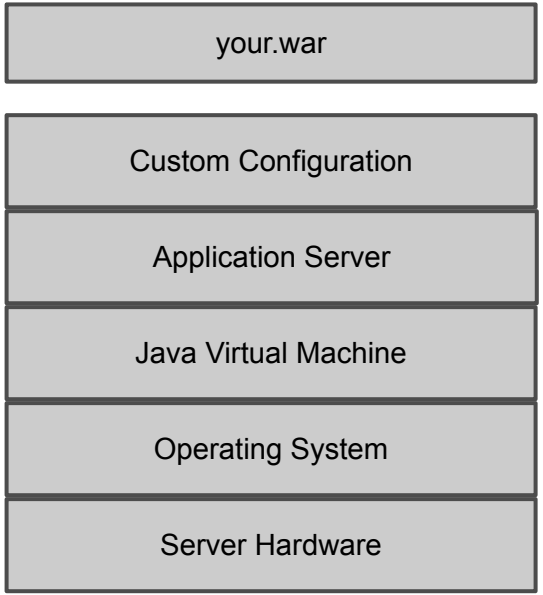
Have you ever had “/” vs “\” break your app? (Unix to Windows)

Or perhaps your app needed a unique version of a JDBC driver?

Or had a datasource with a slightly misspelled JNDI name?

Or received a patch for the JVM or app server that broke your code?

Your Stack Matters



JDBC driver, datasource, JMS queue, users

Weblogic 10.x.y, Tomcat 6.x.y, JBoss EAP 6.x.y

Java 1.7.0_67 or Java 1.8.0_65

Linux Kernel Version & Distribution

Your Stack Matters

MyApp.war has been tested with the following

- On my Windows 7 desktop
  - JDK 1.8.43
  - Jboss WildFly
- Configuration:
  - Datasource: MySQLDS
  - Tested with: mysql-connector-java-5.1.31-bin.jar

Your Stack Matters

MyApp.war has been tested with the following

- On my Windows 7 desktop
  - JDK 1.8.43
  - Jboss WildFly
- Configuration:
  - Datasource: MySQLDS
  - Tested with: mysql-con

Production Environment

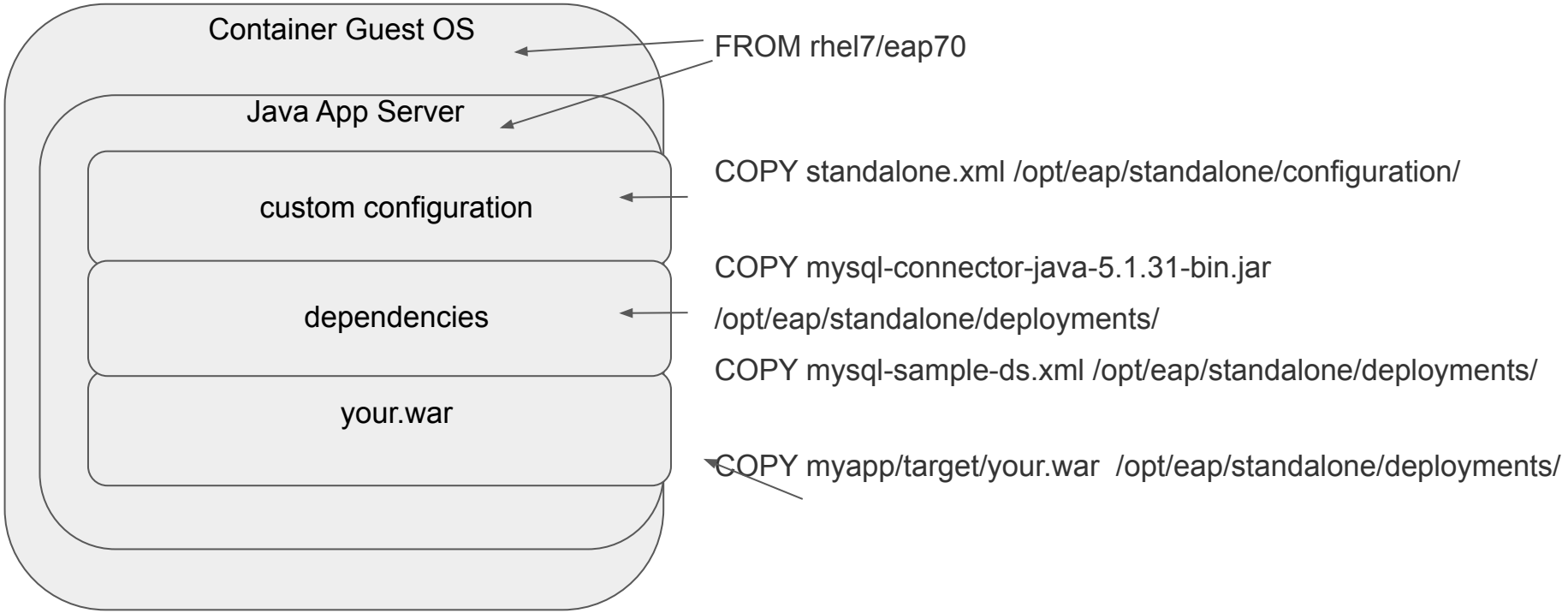
Red Hat Enterprise Linux 6.2

JRE 1.7.3

Jboss EAP 6.41

Oracle 9

Container Magic



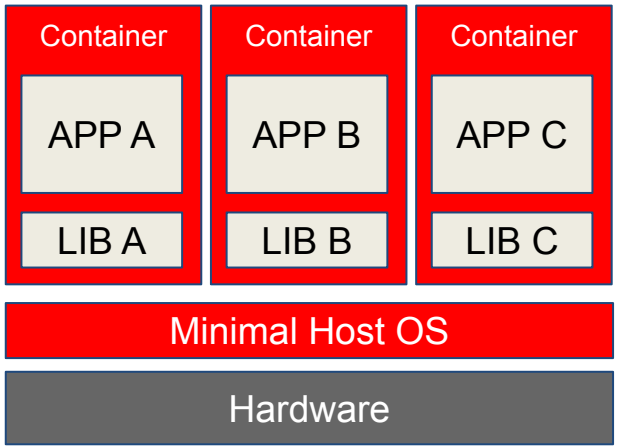
Containers package applications with dependencies and isolate run-time

Software packaging concept that typically includes an application and all of its runtime dependencies.

- Easy to deploy and portable across host systems
- Isolates applications on a host operating system.

In RHEL, this is done through:

- Control Groups (cgroups)
- Kernel Namespaces
- SELinux, sVirt, iptables
- Docker



This sounds like virtualization. What's the difference?

With virtualization, the package that can be passed around is a virtual machine and it includes an entire operating system as well as the application.