redhat.

DO LINUX DISTRIBUTIONS STILL MATTER WITH CONTAINERS?

Balancing the value of innovation & maintenance

Scott McCarty
Principal Offering Product Manager (popm)
02/01/2020

opensource.com

redhat. SUPPORTED BY RED HAT

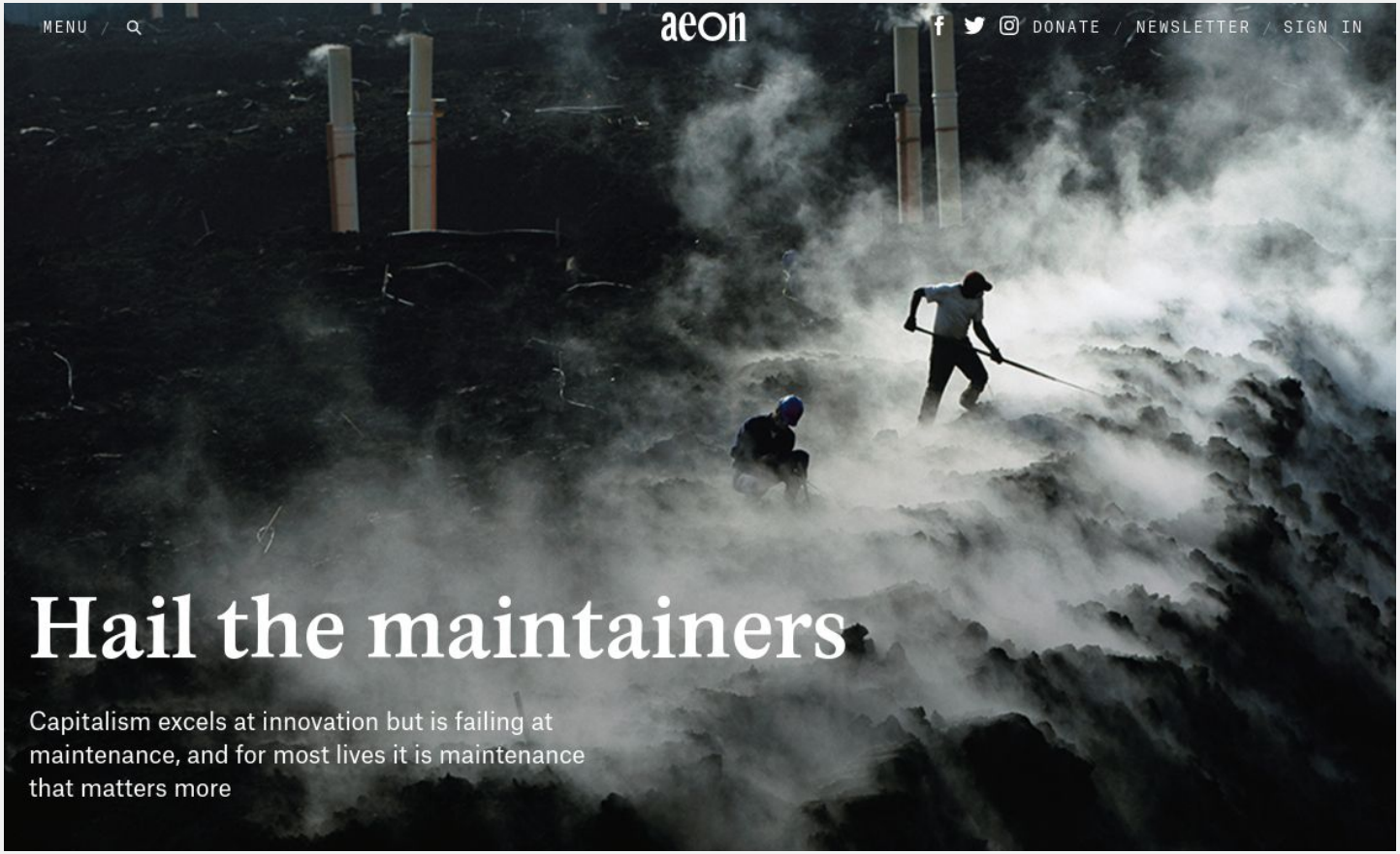
LOG IN SIGN UP

Articles Resources Downloads About The Open Org

Do Linux distributions still matter with containers?

There are two major trends in container builds: using a base image and building from scratch. Each has engineering tradeoffs.

20 Feb 2019 | Scott McCarty (Red Hat) | 37 | 4 comments


MENU
aeon
DONATE / NEWSLETTER / SIGN IN
Hail the maintainers
Capitalism excels at innovation but is failing at maintenance, and for most lives it is maintenance that matters more

“I don’t want to care about the operating system anymore”

Let’s use tires as an analogy…


Safety


Road Performance
RJ66 KYH


Amature Racing


Professional Racing
16

OK, so we do still care. But, what criteria? What context?

UNDERSTANDING THE CRITERIA WITH CONTAINERS

redhat.

THERE ARE A LOT OF DIFFERENT OPTIONS

Figuring out which container base image to use can be difficult

Traditional Options

- Red Hat Enterprise Linux
- Fedora
- CentOS
- Debian
- Ubuntu
- Windows

Minimal Options

- Distroless
- Scratch
- RHEL Minimal
- Alpine

There is no cloud!

Just someone else's computer

There is no distroless!

Just another dependency ***you*** manage

HOW TO SELECT THE RIGHT IMAGE

There is some standard criteria that can help

Architecture

- C Library
- Core Utilities
- Size
- Life Cycle
- Compatibility
- Troubleshooting
- Technical Support
- ISV Support
- Distributability

Security

- Updates
- Tracking
- Security Response Team

Performance

- Automated
- Performance Engineering

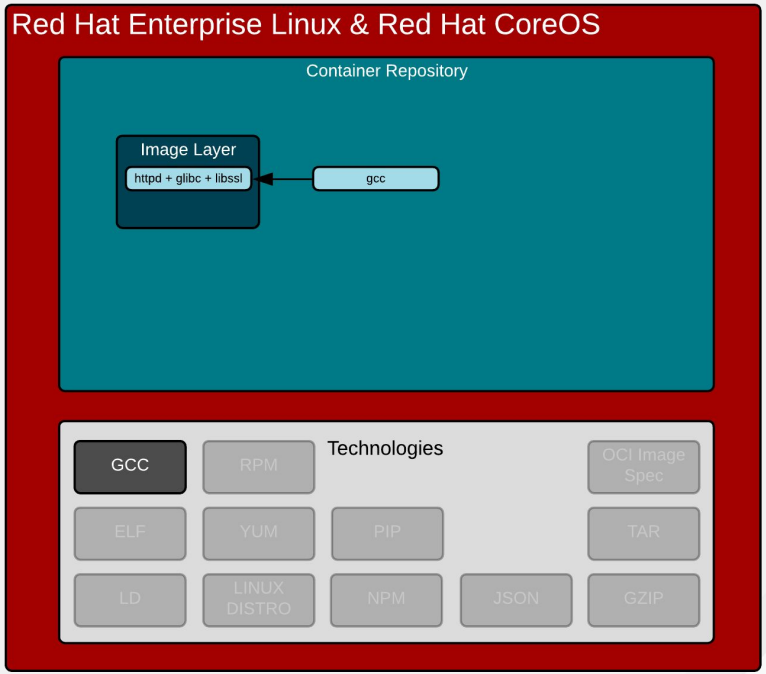
HOW DO THINGS WORK?

IT ALL STARTS WITH COMPILING

Statically linking everything into the binary

Starting with the basics:

- Programs rely on libraries
- Especially things like SSL - difficult to reimplement in for example PHP
- Math libraries are also common
- Libraries can be compiled into binaries - called static linking
- Example: C code + glibc + gcc = program

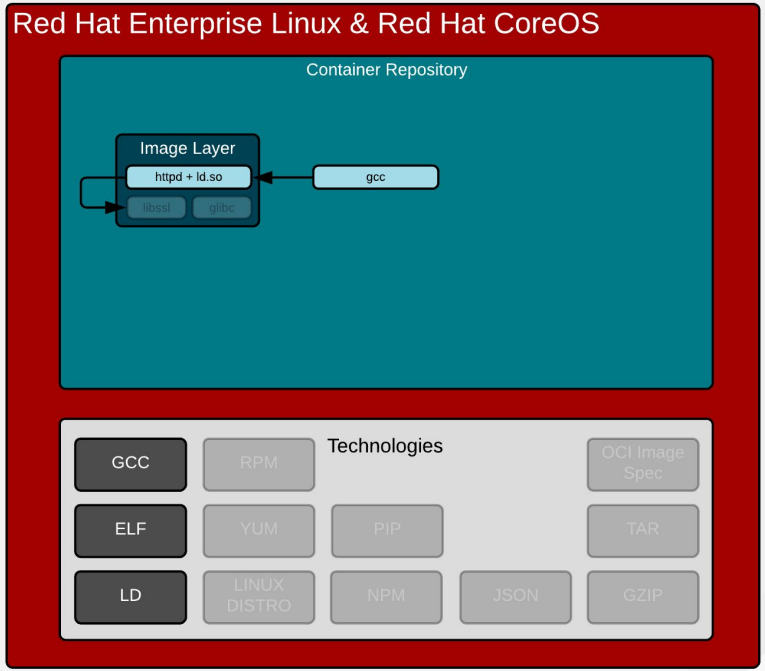


LEADS TO DEPENDENCIES

Dynamically linking libraries into the binary

Getting more advanced:

- This is convenient because programs can now share libraries
- Requires a dynamic linker
- Requires the kernel to understand where to find this linker at runtime
- Not terribly different than interpreters (hence the operating system is called an interpretive layer)

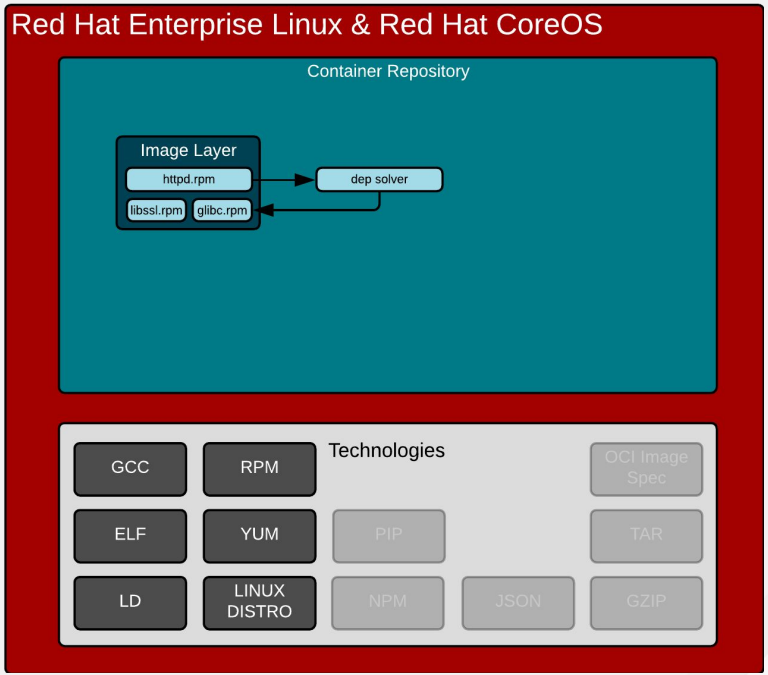


PACKAGING & DEPENDENCIES

RPM and Yum were invented a long time ago

Dependencies need resolvers:

- Humans have to create the dependency tree when packaging
- Computers have to resolve the dependency tree at install time (container image build)
- This is essentially what a Linux distribution does sans the installer (container image)

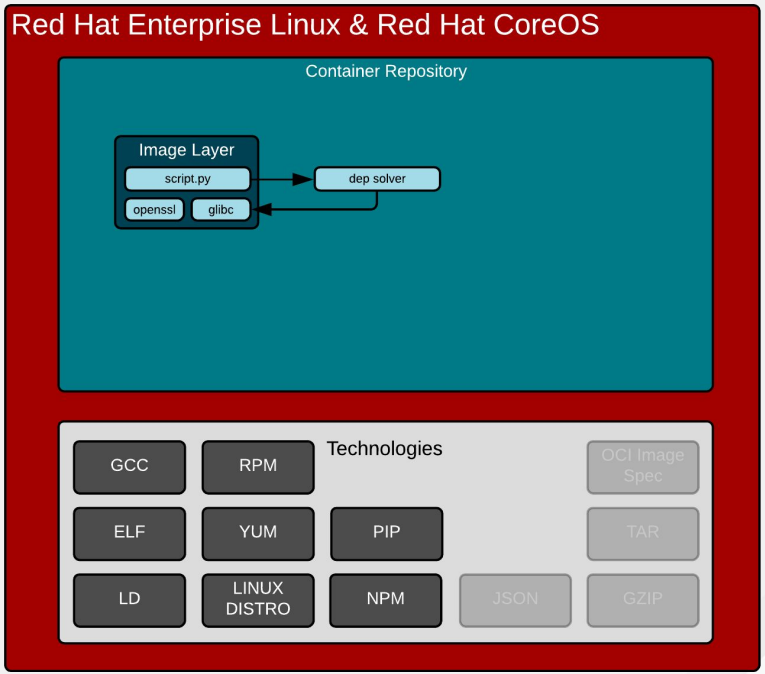


PACKAGING & DEPENDENCIES

Interpreters have to handle the same problems

Dependencies need resolvers:

- Humans have to create the dependency tree when packaging
- Computers have to resolve the dependency tree at install time (container image build)
- Python, Ruby, Node.js, and most other interpreted languages rely on C libraries for difficult tasks (ex. SSL)

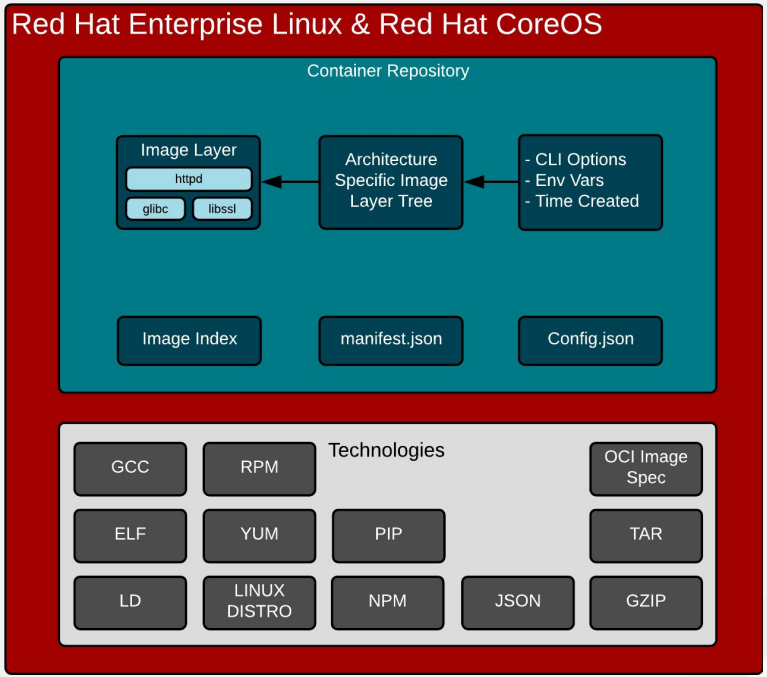


CONTAINER IMAGE PARTS

Governed by the OCI image specification standard

Lots of payload media types:

- Image Index/Manifest.json - provide index of image layers
- Image layers provide change sets - adds/deletes of files
- Config.json provides command line options, environment variables, time created, and much more
- Not actually single images, really repositories of image layers

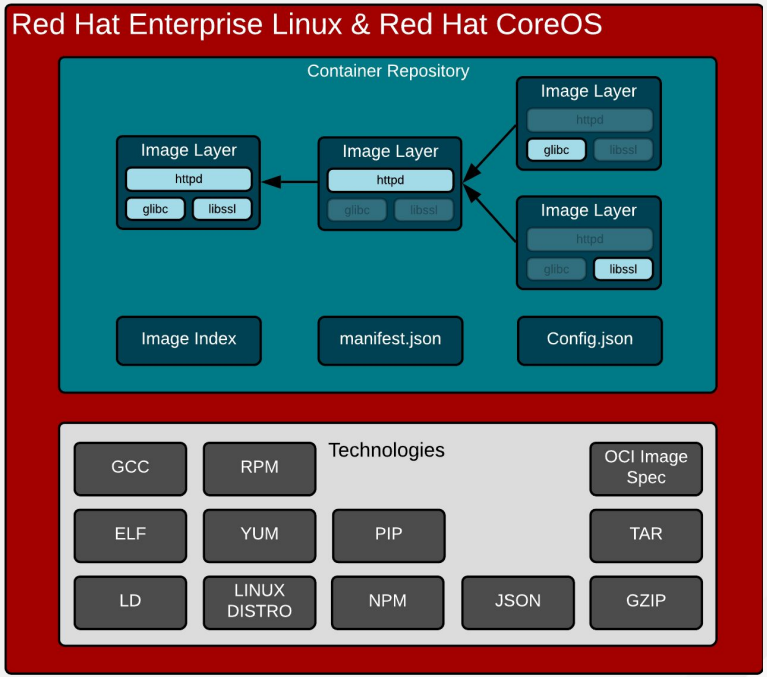


LAYERS ARE CHANGE SETS

Each layer has adds/deletes

Each image layer is a permutation in time:

- Different files can be added, updated or deleted with each change set
- Still relies on package management for dependency resolution
- Still relies on dynamic linking at runtime

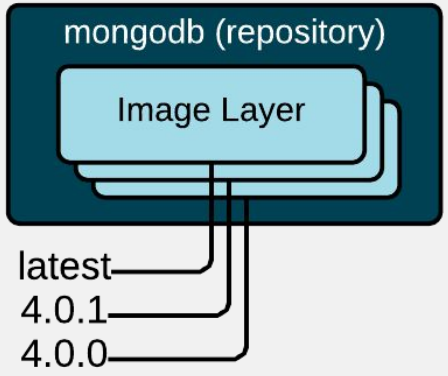


LAYERS ARE CHANGE SETS

Some layers are given a human readable name

Each image layer is a permutation in time:

- Different files can be added, updated or deleted with each change set
- Still relies on package management for dependency resolution
- Still relies on dynamic linking at runtime



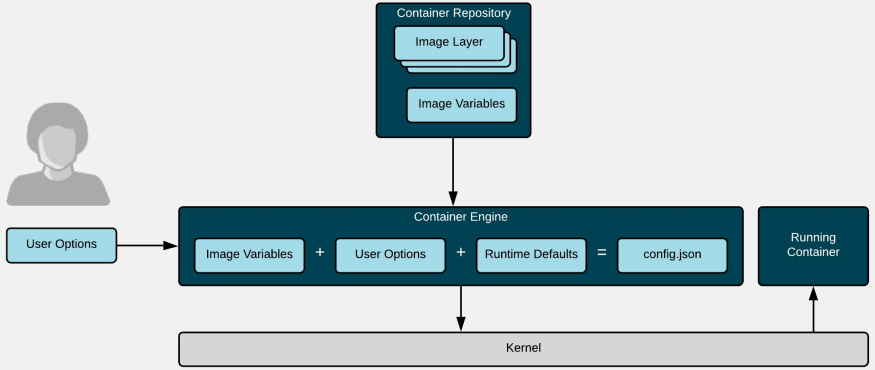
Layers and Tags

CONTAINER IMAGES & USER OPTIONS

Come with default binaries to start, environment variables, etc

Each image layer is a permutation in time:

- Different files can be added, updated or deleted with each change set
- Still relies on package management for dependency resolution
- Still relies on dynamic linking at runtime

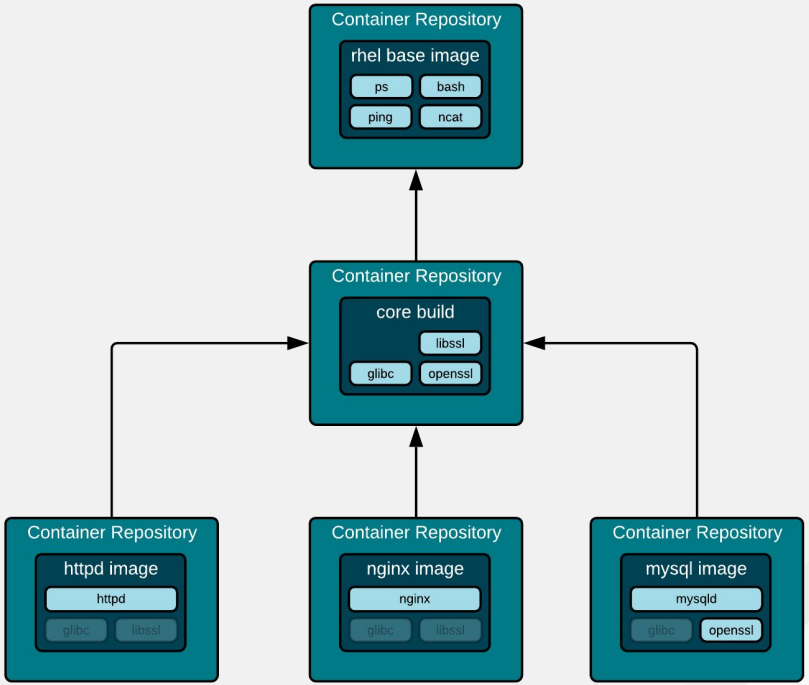


INTER REPOSITORY DEPENDENCIES

Think through this problem as well

You have to build this dependency tree yourself:

- DRY - Do not repeat yourself. Very similar to functions and coding
- OpenShift BuildConfigs and DeploymentConfigs can help
- Letting every development team embed their own libraries takes you back to the 90s

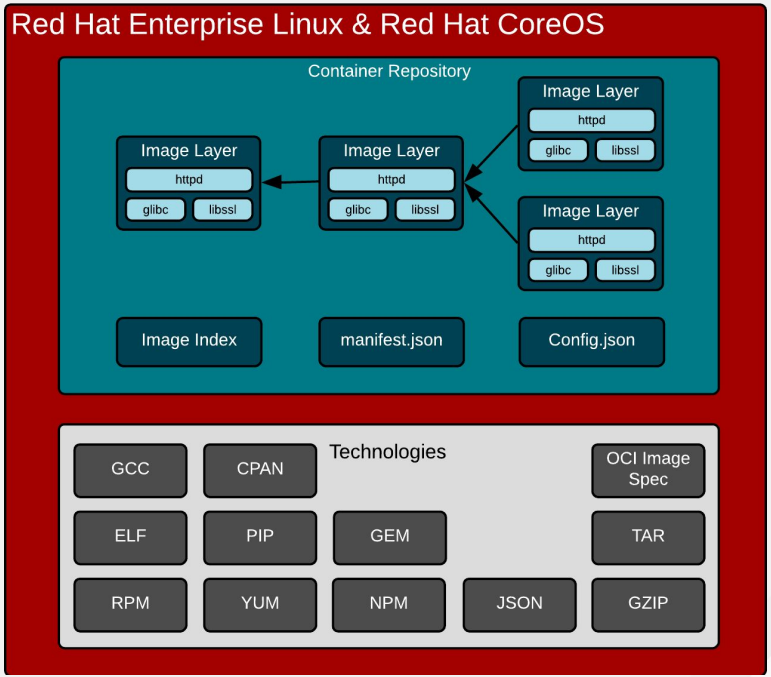


CONTAINER IMAGE

Open source code/libraries, in a Linux distribution, in a tarball

Even base images are made up of layers:

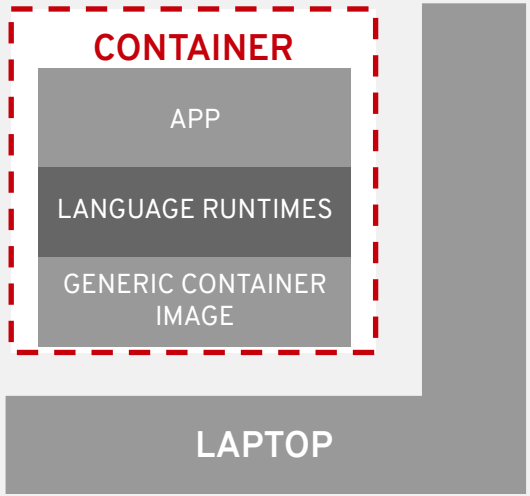
- Libraries (glibc, libssl)
- Binaries (httpd)
- Packages (rpms)
- Dependency Management (yum)
- Repositories (rhel7)
- Image Layer & Tags (rhel7:7.5-404)
- At scale, across teams of developers and CI/CD systems, consider all of the necessary technology



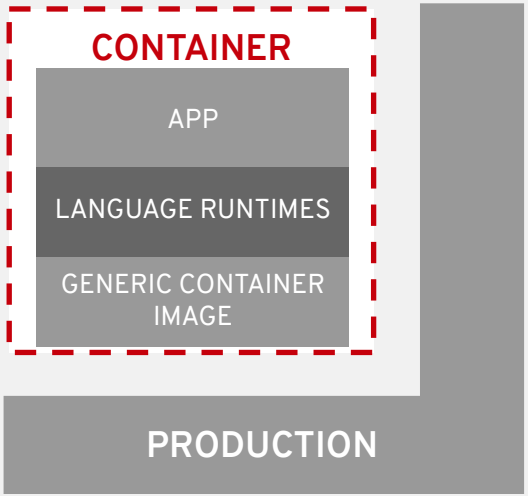
UNDERSTANDING THE CONTEXT WITH CONTAINERS

IT WORKS ON MY LAPTOP, AND...

From an architecture perspective



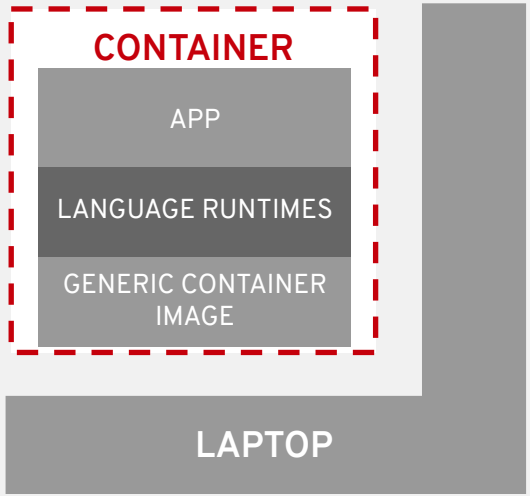
Works on my laptop



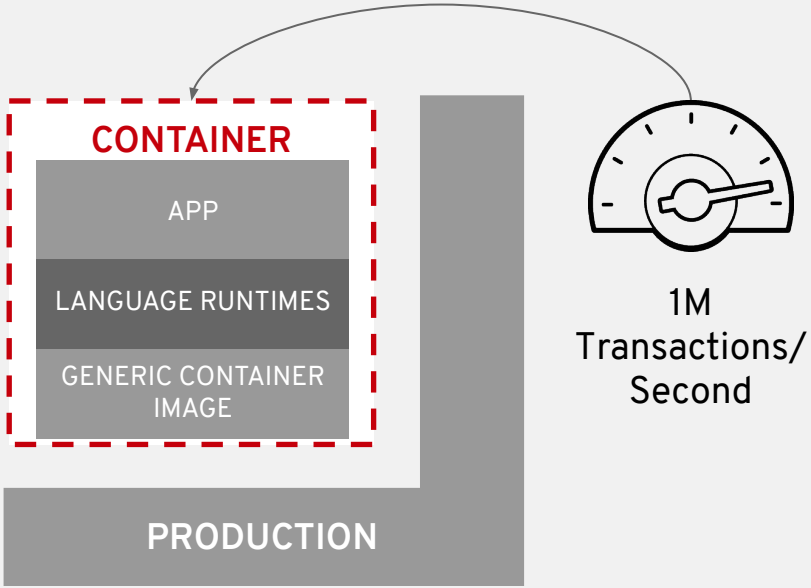
The service starts in production

IT WORKS ON MY LAPTOP, BUT...

What about performance?



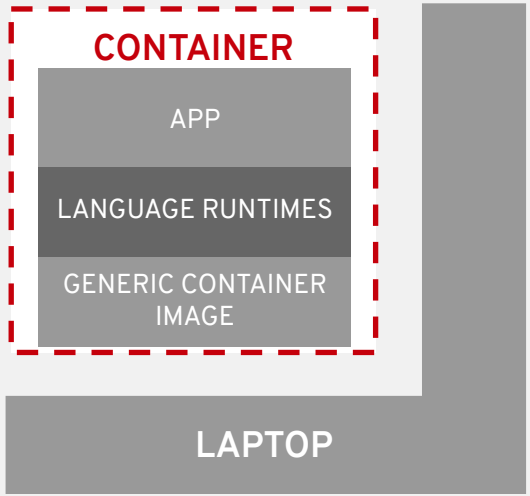
Works on my laptop



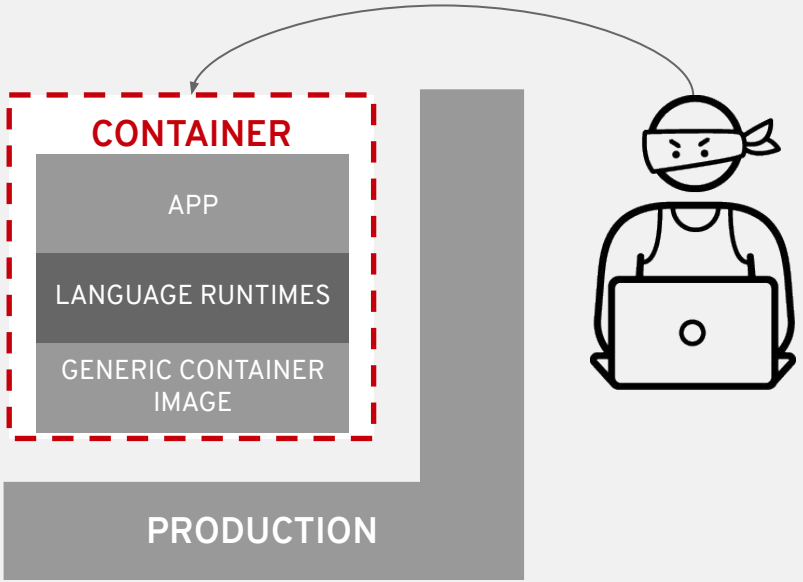
But, what about at 1M TPS

IT WORKS ON MY LAPTOP, BUT...

What about security?



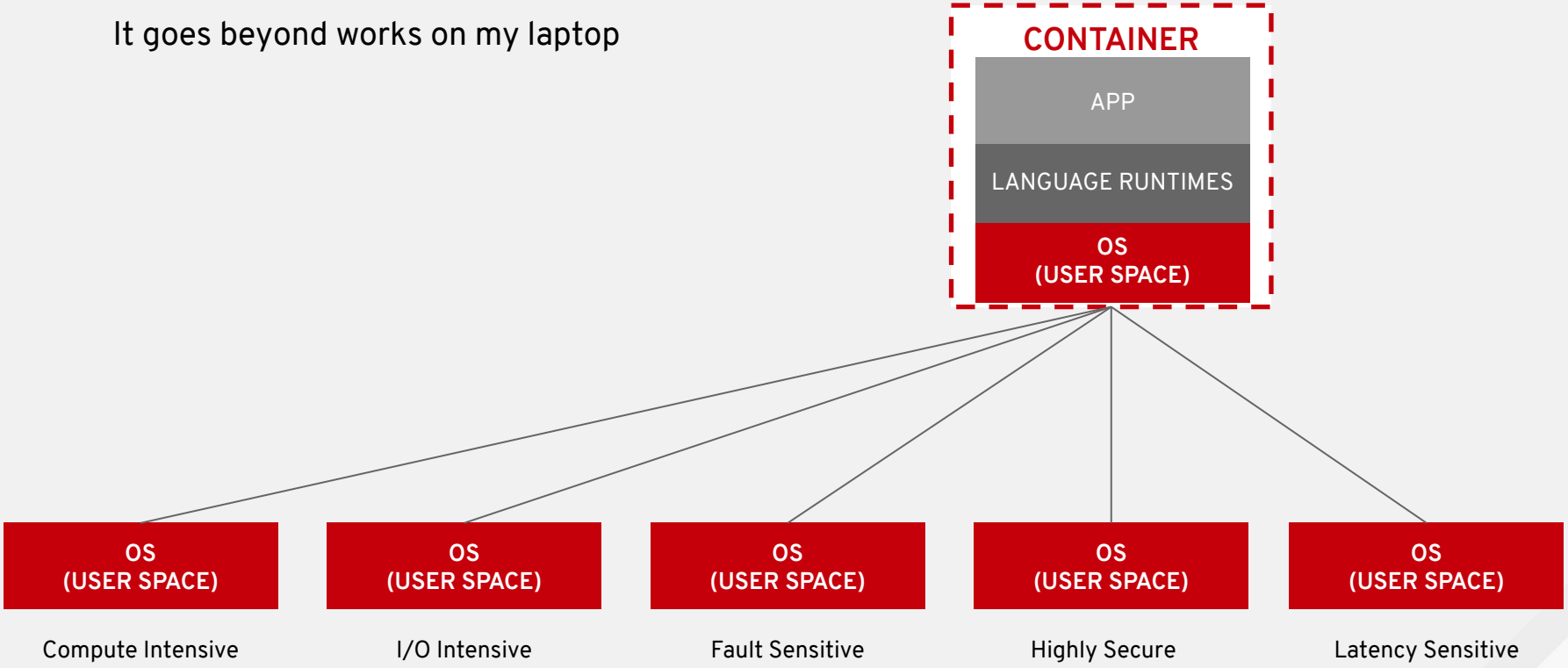
Works on my laptop



What about hackers?

THE QUALITY OF THE BITS MATTERS

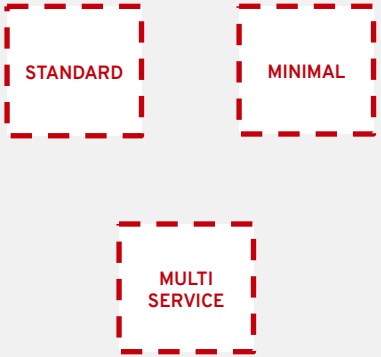
It goes beyond works on my laptop



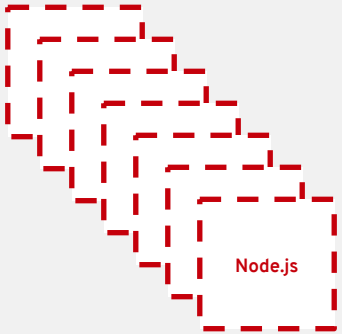
redhat.

WHAT IS THE RED HAT UNIVERSAL BASE IMAGE?

Three base images, language runtime images, and software packages



Base Images



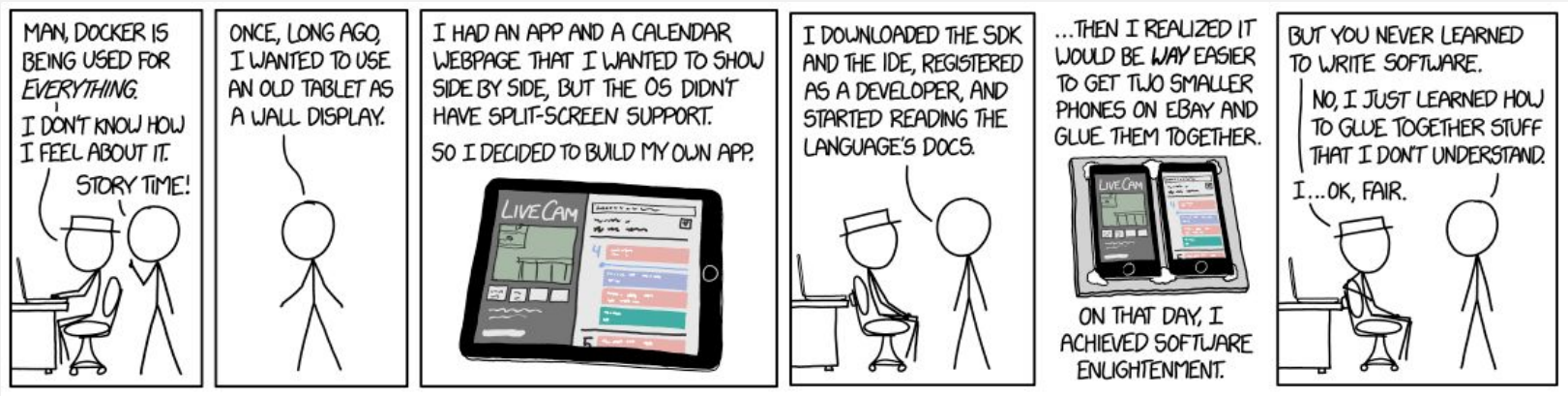
Pre-Built Language Images

Package Subset

RECOMMENDATIONS

PEOPLE DON'T UNDERSTAND THE VALUE

This is the fundamental problem

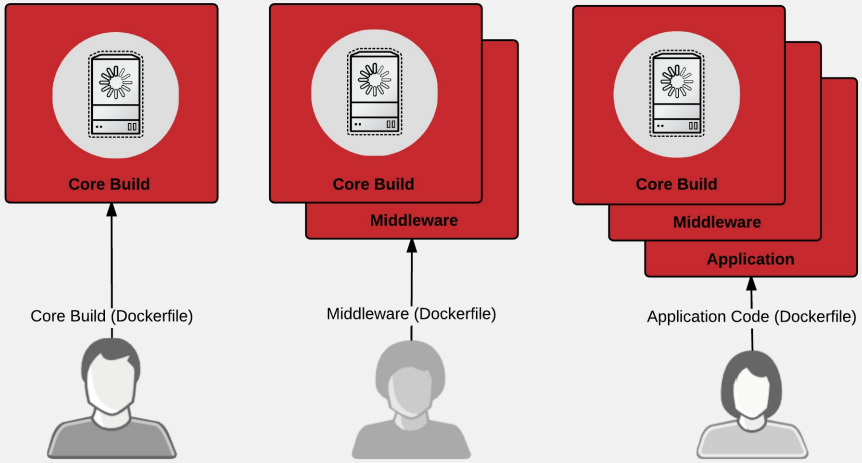


CALL TO ACTION

All Linux distros need to think about market problems

Recommendations:

- Educate people on current value: dependency tree which provides a catalog of software and libraries
- Create new value: smaller images, environment variables to configure software inside, sane defaults, new optimized security, optimized tooling, meta-data



QUESTIONS?

Citations

- Twitter: @fatherlinux
- https://opensource.com/article/19/2/linux-distributions-still-matter-containers
- https://www.redhat.com/en/blog/introducing-red-hat-universal-base-image
- https://developers.redhat.com/blog/2018/02/22/container-terminology-practical-introduction/
- https://www.techrepublic.com/article/6-critical-factors-to-consider-when-deploying-containers/
- https://www.techrepublic.com/article/what-does-linux-have-to-do-with-containers-everything/

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos

SPECIFICALLY CONTAINER IMAGES

This is the fundamental problem