Red Hat
Enterprise Linux

Red Hat Enterprise Linux 8

What's New
NYRHUG August 2019

Patrick Ladd
Technical Account Manager
pladd@redhat.com
https://people.redhat.com/pladd

Red Hat

What's New in RHEL 8?

RHEL 8 Basics	In Place Upgrades	Cockpit
Release Cycle	Ansible Sys Roles	OCI / UBI
App Streams	Image Builder	VDO
yum v4	Insights	

RHEL 8

At a glance:

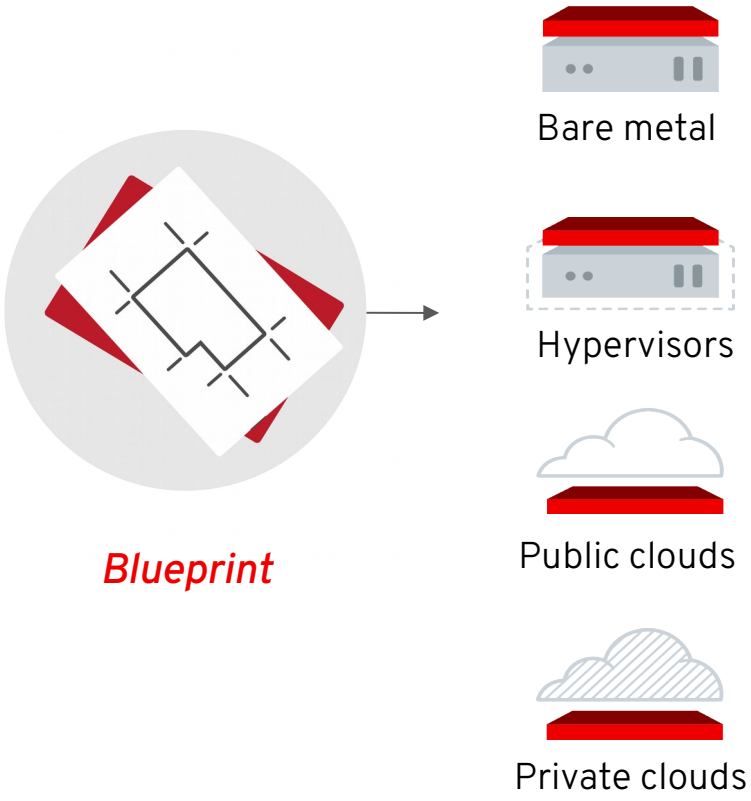
The latest updates from upstream communities, combined with continuity of expected tools

Kernel Version	4.18+
System Compiler	GCC 8.2, LLVM 6.0
Hardware Arhictectures	Intel/AMD 64-bit, IBM Power LE, IBM z Systems, ARM 64-bit
Default File System	XFS
Package Management	Yum v4
Time Synchronization	Chrony
Networking	NetworkManager

Installs and Upgrades

Red Hat

Create images for all your environments with image builder



Single source
Lets you create gold images for any environment from the same blueprint increasing stability and consistency

Any footprint
Supports public cloud, private cloud, enterprise hypervisors, and bare metal

Simple interface
Provides web-based view within the web console for selecting packages and creating blueprints

Red Hat

Image Builder

Image Formats

- Raw disk (.img)
- Live ISO (.iso)
- File system (.img)
- Tarball (.tar.xz)
- VMDK (VMware® vSphere® Hypervisor)
- AMI (Amazon Web Services®)
- VHD (Microsoft® Azure®)
- QCOW2 for KVM/RHV/Satellite/CloudForms
- QCOW2 for OpenStack

Composer Interfaces

- Command Line
- Cockpit Plugin

Installation

- `yum install lorax lorax-composer composer-cli cockpit-composer`
- `systemctl enable --now lorax-composer.socket`
- `systemctl restart cockpit.service`

Demo

- Install Image Builder
- Create blueprint
- Customize blueprint
- Create image

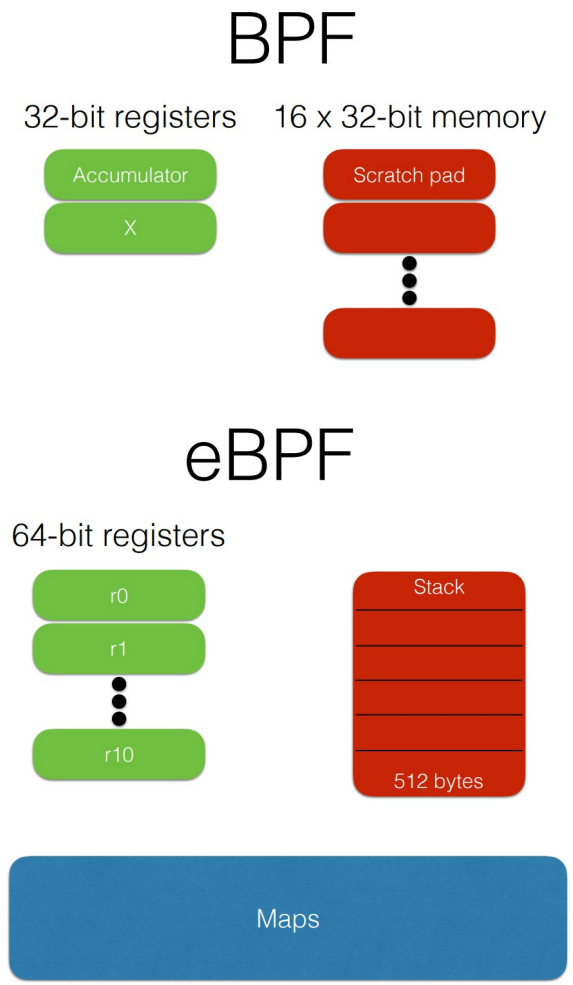
Red Hat

eBPF

Red Hat

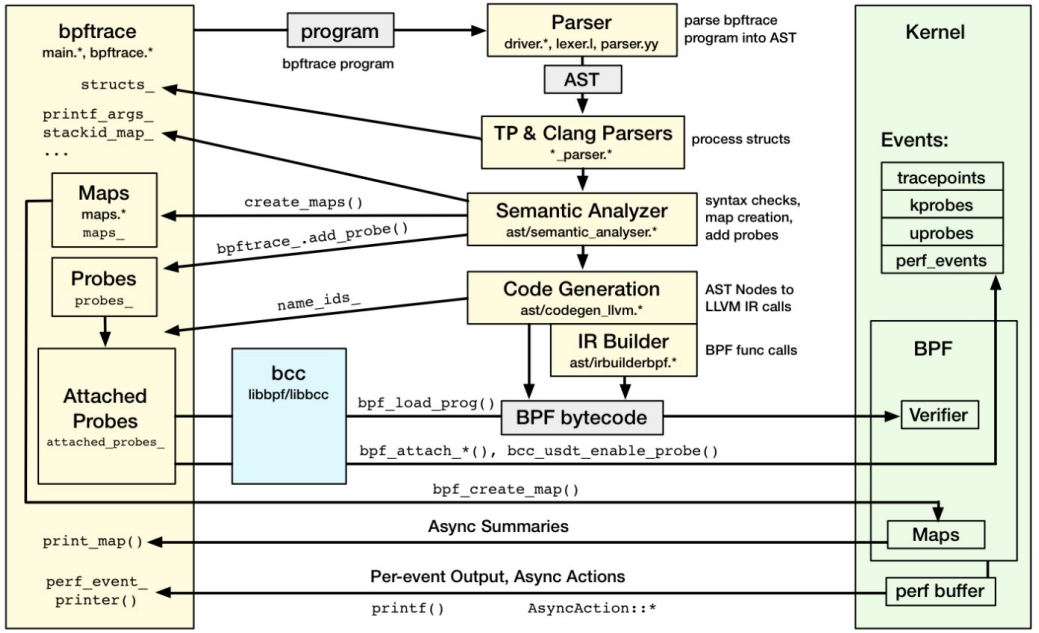
eBPF – What is it?

- BPF: Berkely Packet Filter
  - Network packet filtering circa 1993
  - Programs run on register-based virtual machine
    - 2x 32 bit registers
    - 16x 32 bit memory
- eBPF: enhanced BPF
  - 10x 64-bit registers
  - Memory Stack
  - Maps
  - Multiprocessor instructions
  - Kernel calls



eBPF – Security through simplicity

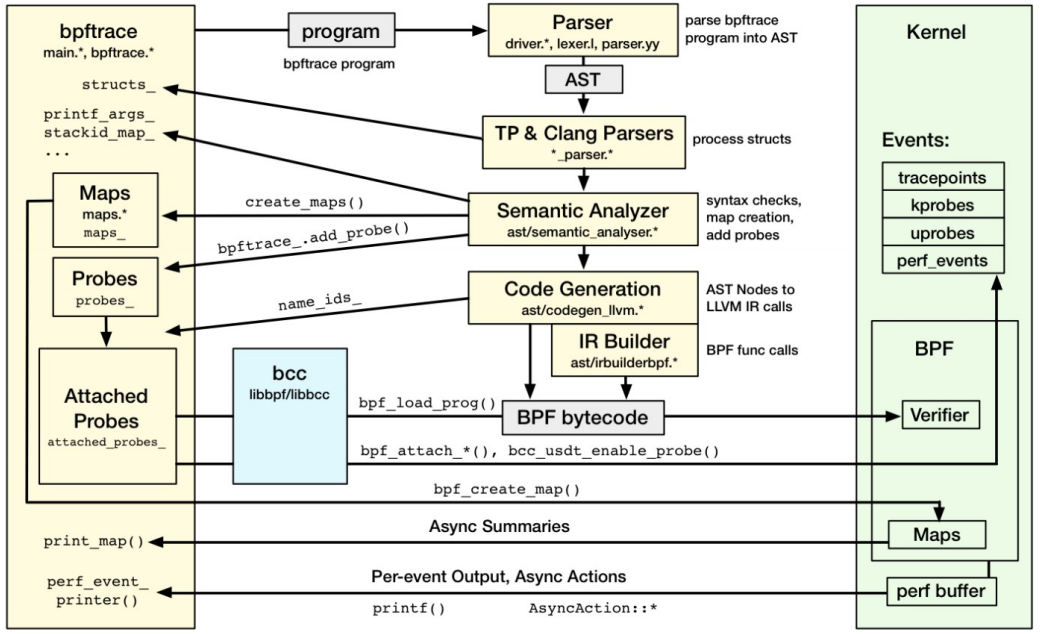
- Sanity Checking via in-kernel verifier
  - Program terminates
  - No loops
  - No unreachable instructions
  - Valid register & stack state
  - No out-of-bounds jumps / data access
  - Secure mode – no pointer arithmetic
  - No uninitialized register/stack reads
  - No writing to read-only locations
  - Access to restricted set of kernel functions / data structures



Red Hat

eBPF – Speed

- Simple → direct mapping to machine code via JIT
- Attached directly to kernel functions to serve a specific purpose
- Triggered automatically when those functions are called



Red Hat

eBPF – Functions

- `strace` replacement
- Network packet filtering
- Security (`seccomp`)
- Kernel monitoring / debug

eBPF – Functions

- **BPF_PROG_TYPE_SOCKET_FILTER**: a network packet filter
- **BPF_PROG_TYPE_KPROBE**: determine whether a kprobe should fire or not
- **BPF_PROG_TYPE_SCHED_CLS**: a network traffic-control classifier
- **BPF_PROG_TYPE_SCHED_ACT**: a network traffic-control action
- **BPF_PROG_TYPE_TRACEPOINT**: determine whether a tracepoint should fire or not
- **BPF_PROG_TYPE_XDP**: a network packet filter run from the device-driver receive path
- **BPF_PROG_TYPE_PERF_EVENT**: determine whether a perf event handler should fire or not
- **BPF_PROG_TYPE_CGROUP_SKB**: a network packet filter for control groups
- **BPF_PROG_TYPE_CGROUP_SOCK**: a network packet filter for control groups that is allowed to modify socket options
- **BPF_PROG_TYPE_LWT_***: a network packet filter for lightweight tunnels
- **BPF_PROG_TYPE_SOCK_OPS**: a program for setting socket parameters
- **BPF_PROG_TYPE_SK_SKB**: a network packet filter for forwarding packets between sockets
- **BPF_PROG_CGROUP_DEVICE**: determine if a device operation should be permitted or not

eBPF – Installing

- Install:
  RHEL8:
  - **`yum install bcc bcc-tools kernel-devel python3-bcc`**

  RHEL7:
  - **`yum install bcc bcc-tools kernel-devel python-bcc`**

- Note: ***eBPF tooling is currently tech preview***
- Tools located in **`/usr/share/bcc/tools`**
- Not in default **`$PATH`**

eBPF – Kernel Tools

- Many useful example tools in **`/usr/share/bcc/tools`**
  - Some useful (and some *terrifying*)
  - *snoop*
    - **`biosnoop`** -block I/O snoop
    - **`dcsnoop`** – dcache snoop
    - **`execsnoop`** – exec() calls
    - **`mountsnoop`** – mounts
    - **`opensnoop`** – open() calls
    - **`statsnoop`** – stat() calls
    - **`syncsnoop`** – sync() calls
    - **`ttysnoop`** – watch all activity on tty :0

eBPF – Kernel Tools

- *latency*
  - **`biolatency`** -block I/O call latency
  - **`funclatency`** -syscall latency
  - **`gethostlatency`** – getaddrinfo() / gethostbyname() latency
- *top*
  - **`biotop`** -block I/O call top calls
  - **`cachetop`** – cache hits
  - **`filetop`** – file I/O
  - **`slabratetop`** – Summarize kmem_cache_alloc() calls
  - **`tcptop`** – TCP

eBPF – Kernel Tools

- *slow*
  - **`dbslower / mysqld_qslower`** – database calls
  - **`nfsslower / ext4slower / xfsslower`** – filesystem ops
  - **`fileslower`** – file ops
  - **`funcslower`** – function calls
  - **`runqslower`** – Kernel scheduler
- Language specific
  - **`java* / perl* / php* / python* / ruby* / tcl*`**
- *count* *hist*
  - Histograms / counts of func calls

eBPF – Kernel Tools

- tcp*
  - **`tcpaccept / tcpdrop / tcpconnect / tcpretrans`** – trace tcp events
  - **`tcplife`** – Summarize lifecycles of TCP connections
  - **`tcptracer / tcpstates`** – show TCP stream info
- Other
  - **`argdist`** – Distribution of args to func call
  - **`bashreadline`** – everything typed on a bash command line
  - **`oomkill`** – Watch oomkill events
  - **`profile`** – Profile CPU usage by sampling stack traces at a timed interval

eBPF – Kernel Tools

- SystemTap (stap) with eBPF
  - **`stap ---runtime=bpf sample.stp`**
- Write your own?
  - https://access.redhat.com/articles/3550581
  - CLANG compiler collection can now compile C code into eBPF
    - Add `-march=bpf`

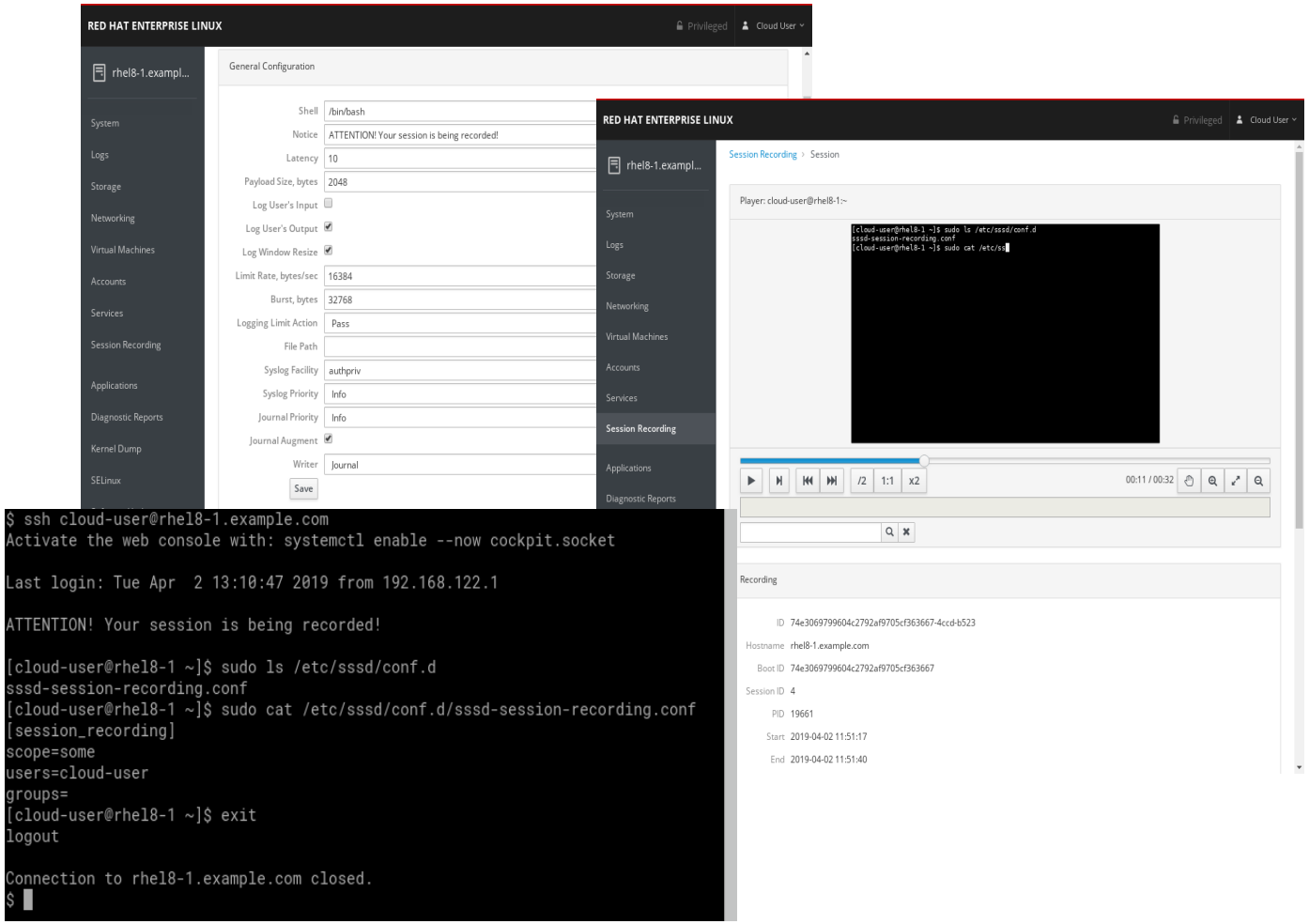
eBPF – Network Packet Filtering

- nftables
  - Designated successor to the iptables, ip6tables, arptables, and ebtables utilities
  - Deserves it’s own talk (coming soon ;))
- Custom eBPF
  - https://access.redhat.com/solutions/3939151
  - Capable of near-linespeed packet filtering
  - Some NICs support embedded eBPF

Features

Red Hat

Recording user terminal sessions



Audit activities

Create a record of actions taken for review against security policies

Create visual guides

Build run books and training materials with demonstrations

Record and play back

Logged via standard channels with multiple playback options

Red Hat

Install - `yum install tlog cockpit-session-recording`
Enable & Check
Playback -
Export - `yum install systemd-journal-remote`
(whole file: `journalctl -o export | /usr/lib/systemd/systemd-journal-remote -o /tmp/example.journal -`)
Session only:

```
journalctl -o verbose | grep -i \"rec\"
id tlog
journalctl -xe -o json-pretty _UID=<User ID of tlog>
journalctl -o export TLOG_REC=<Session ID> |
/usr/lib/systemd/systemd-journal-remote -o
/tmp/example.journal -
```

Playback:

```
tlog-play -r journal --file-path=/tmp/example.journal -M
TLOG_REC=<Session ID>
```

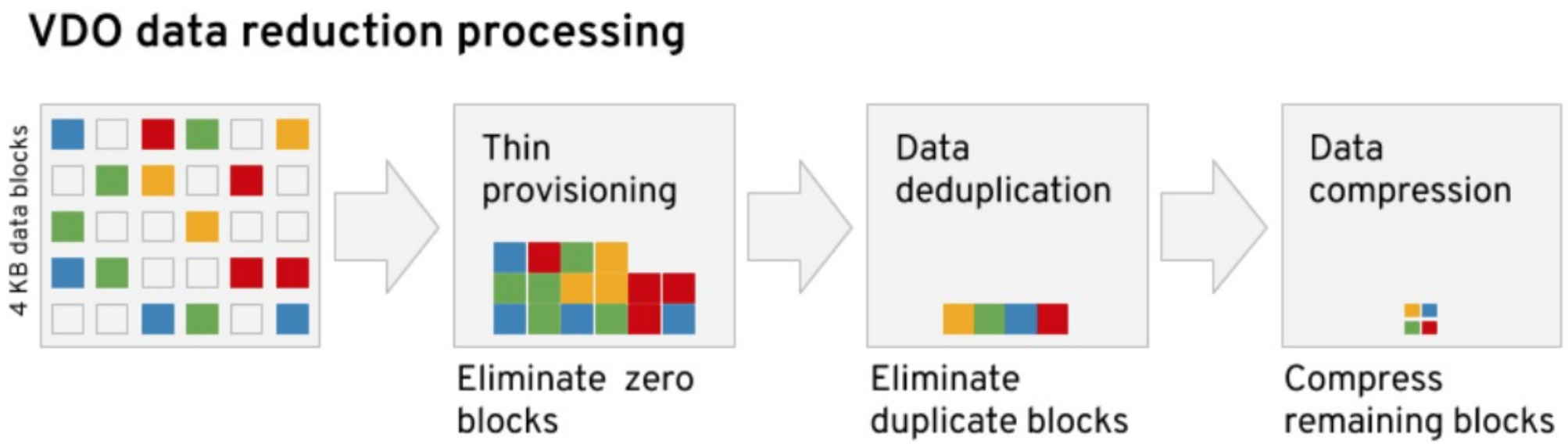
Session Recording Demo

Session Recording

- Documentation:
  - https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/recording_sessions/index
- Red Hat portal
  - https://access.redhat.com/solutions/3902881
  - https://access.redhat.com/solutions/4068941

Virtual Data Optimizer (VDO)

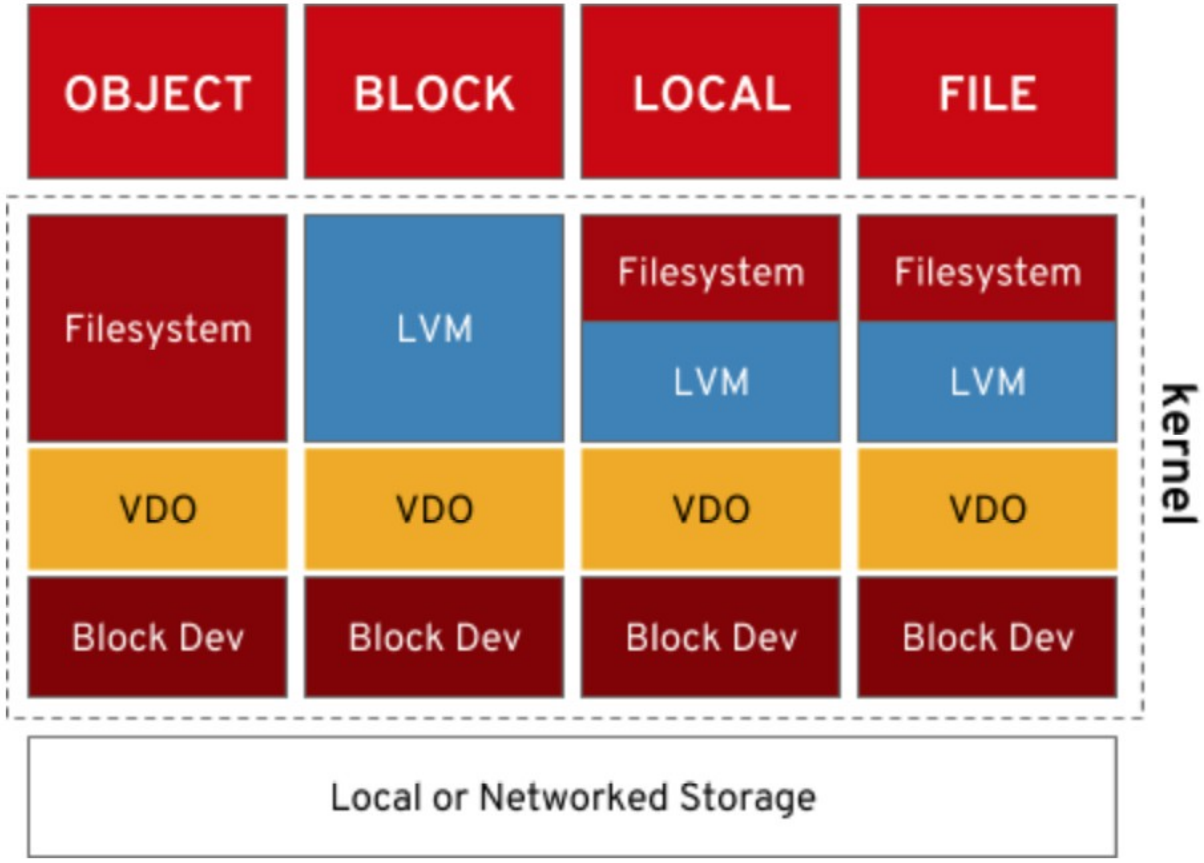
Disk De-duplication and Compression



Red Hat

Virtual Data Optimizer (VDO)

Where It Fits



VDO Demo

Create VDO device
Format and mount it
Copy some data
Observe usage

Virtual Data Optimizer (VDO)

Resources

- Documentation:
  - https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/deduplicating_and_compressing_storage/index
- Red Hat blog:
  - https://www.redhat.com/en/blog/look-vdo-new-linux-compression-layer
  - https://www.redhat.com/en/blog/understanding-concepts-behind-virtual-data-optimizer-vdo-rhel-75-beta
  - https://www.redhat.com/en/blog/determining-space-savings-virtual-data-optimizer-vdo-rhel-75-beta
  - https://www.redhat.com/en/blog/how-set-new-virtual-data-optimizer-device-using-cockpit-web-admin-console

Even More Stuff!

Check this out too!

- nftables
- Network bound disk encryption
- Fast file copy with XFS shared data extents
- Kernel EBPF tracing
- eBPF XDP (Xpress DataPath) and TC (Traffic Control)
- IPSec crpyto offloading
- TCP BBR for Flaky Mobile Networks
- 5 level page tables
- Stratis storage manager
- LUKS2 disk encryption

https://access.redhat.com/articles/4079441

Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc

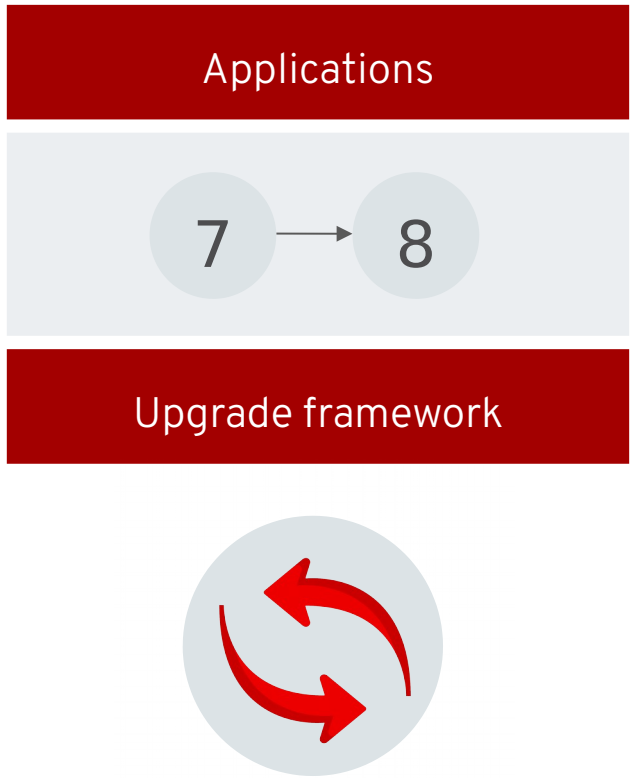

twitter.com/RedHat

Red Hat

In-place Upgrade

Red Hat

In-place upgrades for your systems



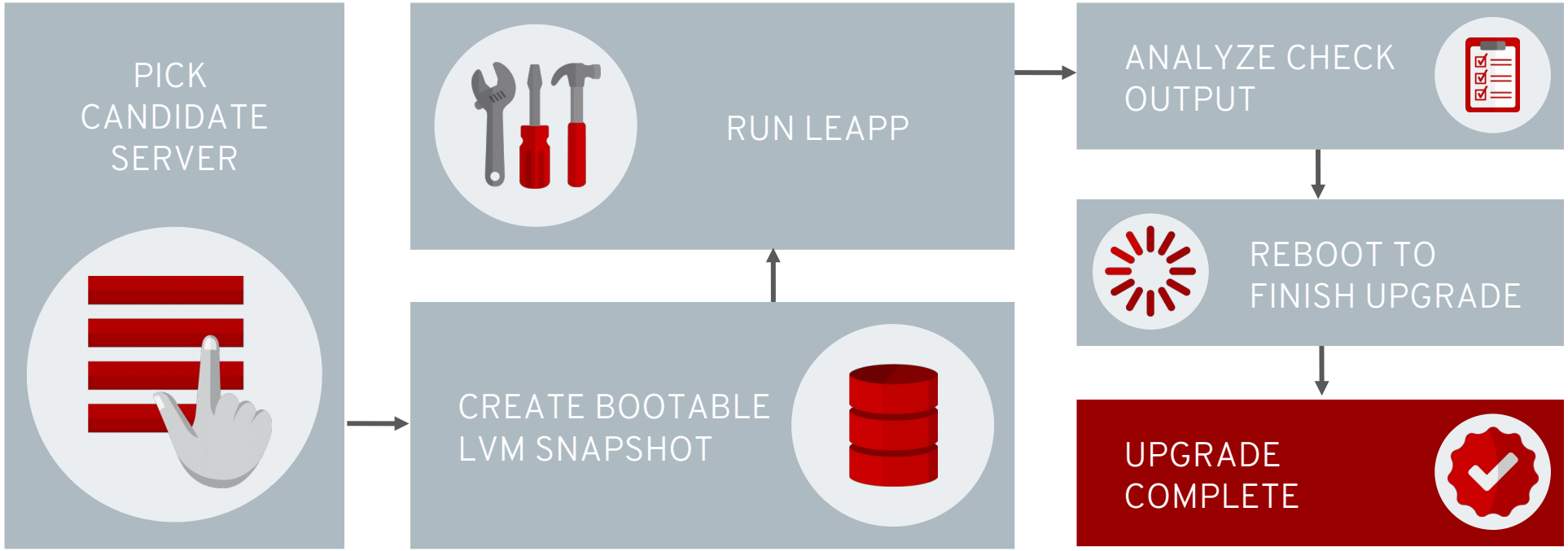
Reduced migrations
Analyze systems to determine if upgrading in place can avoid a costly migration

Easy rollback options
Combine with bootable LVM snapshots for safety

Improved framework
Get better analysis and a simplified process with a more extensible framework

Red Hat

Can I upgrade this host?



Red Hat

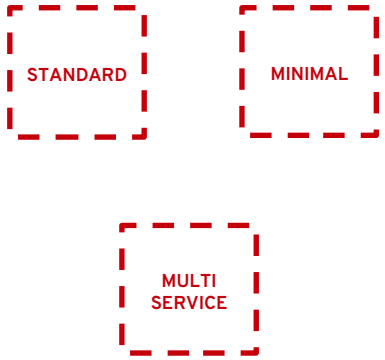
Demo

- Create LVM bootable snapshot
- Run LEAPP
- Analyze output
- Reboot to finish

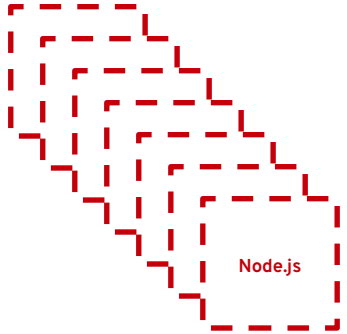
Containers are Linux

Red Hat Universal Base Image (UBI)

"To be the highest quality and most flexible base container image available"



Base Images



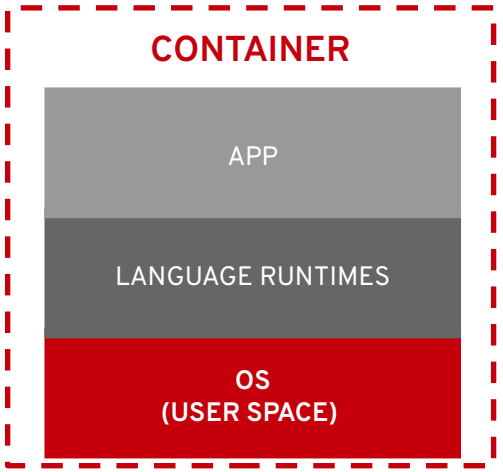
Pre-Built Language Images

Package Subset

Red Hat

Red Hat Universal Base Image (UBI) Licensing

Run anywhere at no charge



The Red Hat Universal Base Image is based on RHEL and made available at no charge by a new end user license agreement.

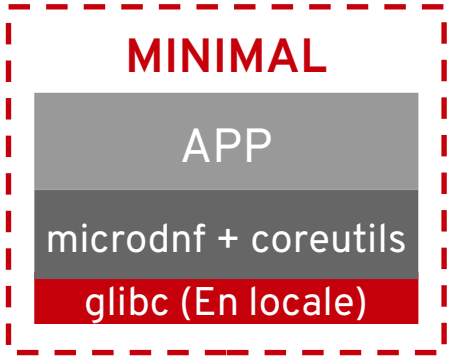
- Supported as RHEL when running on RHEL
- Same Performance, Security & Life cycle as RHEL
- Can attach RHEL support subscriptions as RHEL

https://www.redhat.com/en/about/red-hat-end-user-license-agreements#UBI
http://crunchtools.com/ubi-licensing/

Red Hat

Red Hat Universal Base Image (UBI)

Standard Image Types



ubi8/ubi-minimal

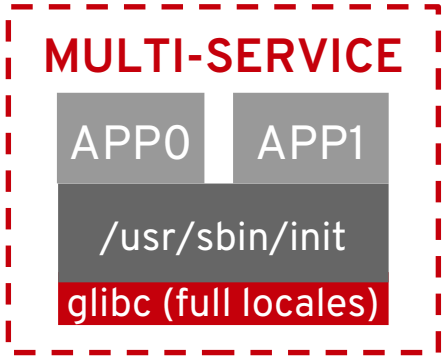
Designed for applications that contain all dependencies (Golang, dotnet, etc)

- Minimized content set
- No suid binaries
- Minimal package manager (install, update, remove)

ubi8/ubi

For any application that runs on RHEL

- Unified, openssl crypto stack
- Full YUM stack
- Includes useful basic OS tools (tar, gzip, vi, etc)



ubi8/ubi-init

Eases running multiple services in a single container

- Configured to run systemd on start
- Simply enable the services at build time

Red Hat

Red Hat Universal Base Image (UBI)

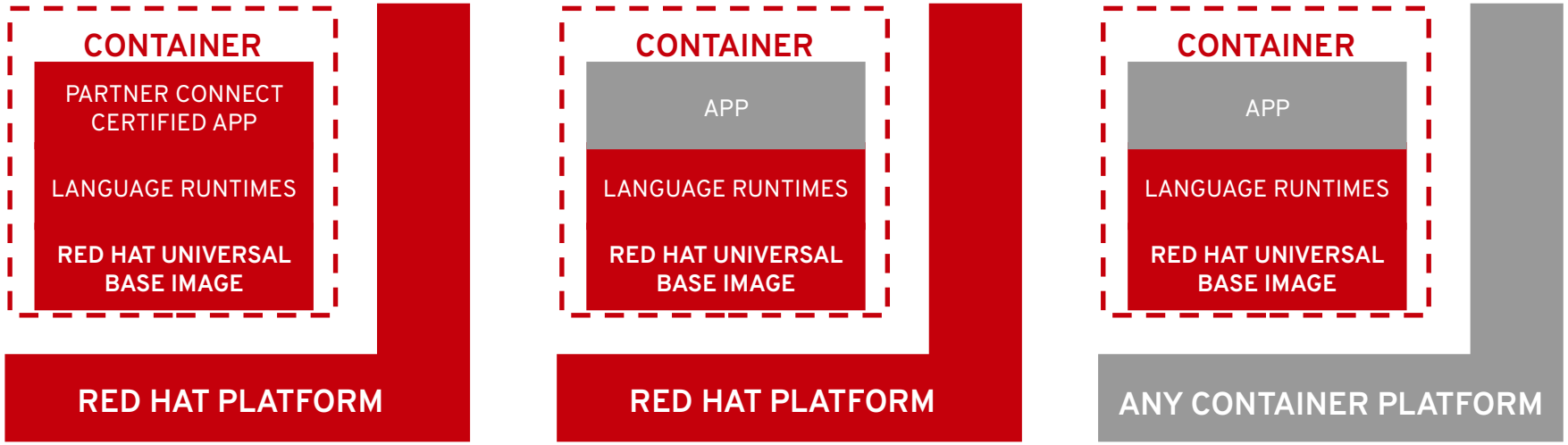
Standard Runtimes

- DotNet
- Perl
- PHP
- NodeJS
- Python
- Ruby
- s2i

Detailed list at registry.redhat.io

Red Hat Universal Base Image (UBI)

Supportability



Certification provides the highest level of support

Enterprise support when run on Red Hat platforms

Trusted base for any environment

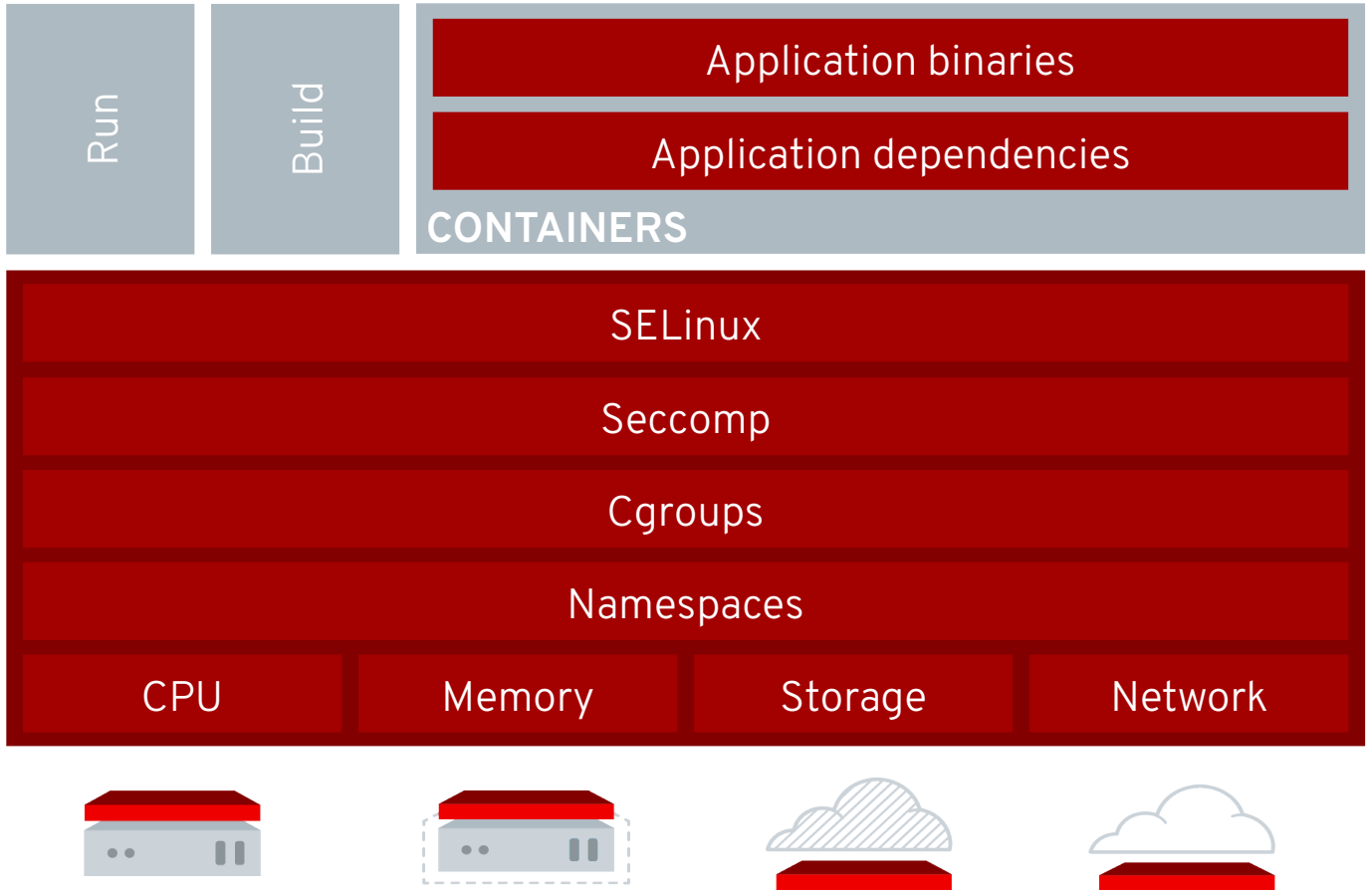
Red Hat

Red Hat Universal Base Image (UBI)

Updates

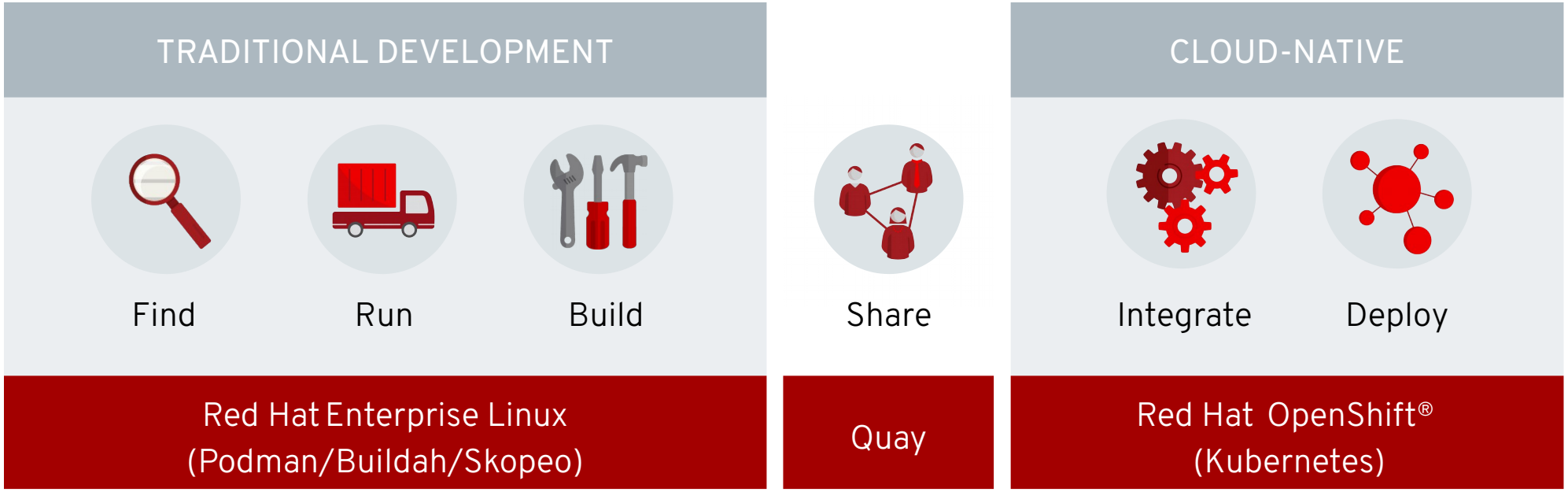
Red Hat

Containers are Linux



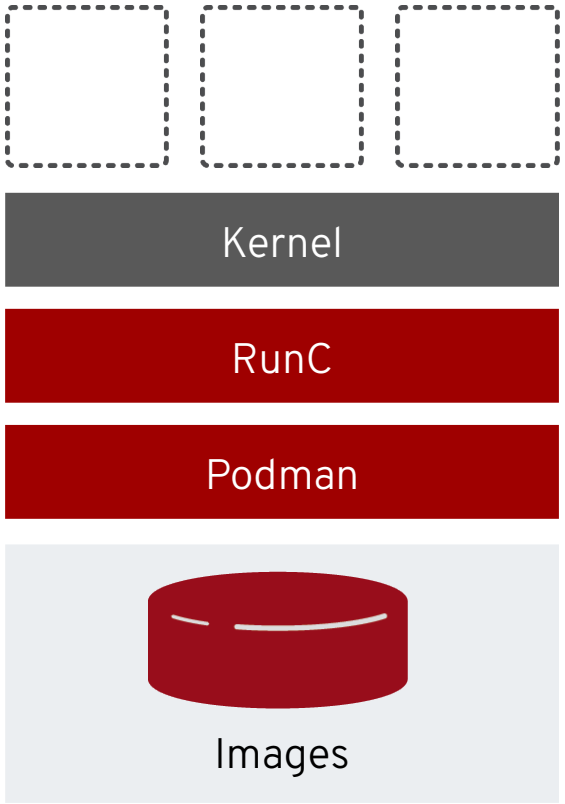
Red Hat

Powering the adoption of containerized workloads



Red Hat

Manage containers with Podman

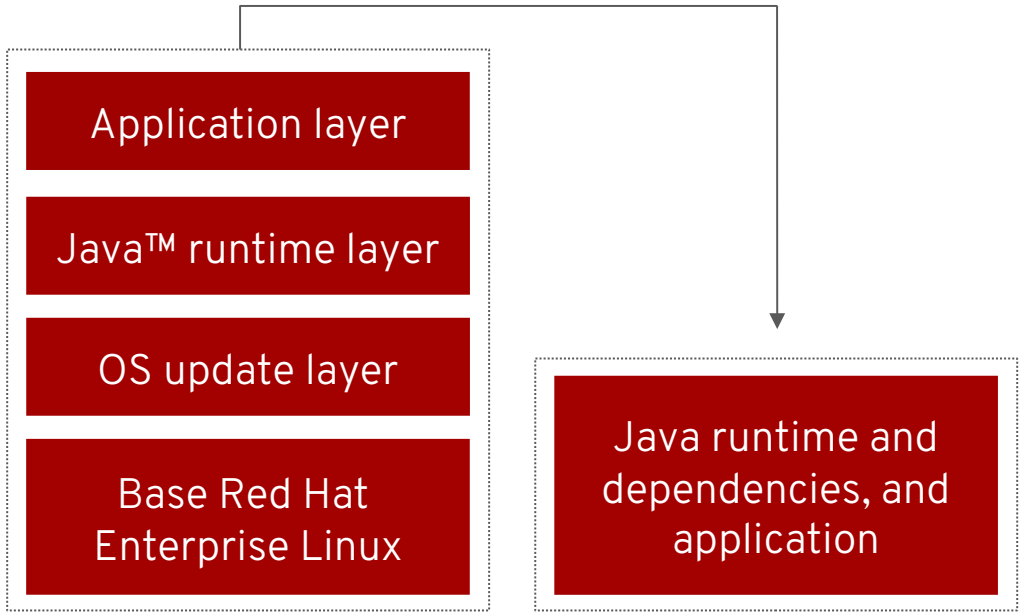


Fast and lightweight
No daemons required

Advanced namespace isolation
Rootless operations for container run and build

Open standards compliant
Creates and maintains any standard Open Containers Initiative (OCI) - compliant containers and pods

Create images with Buildah

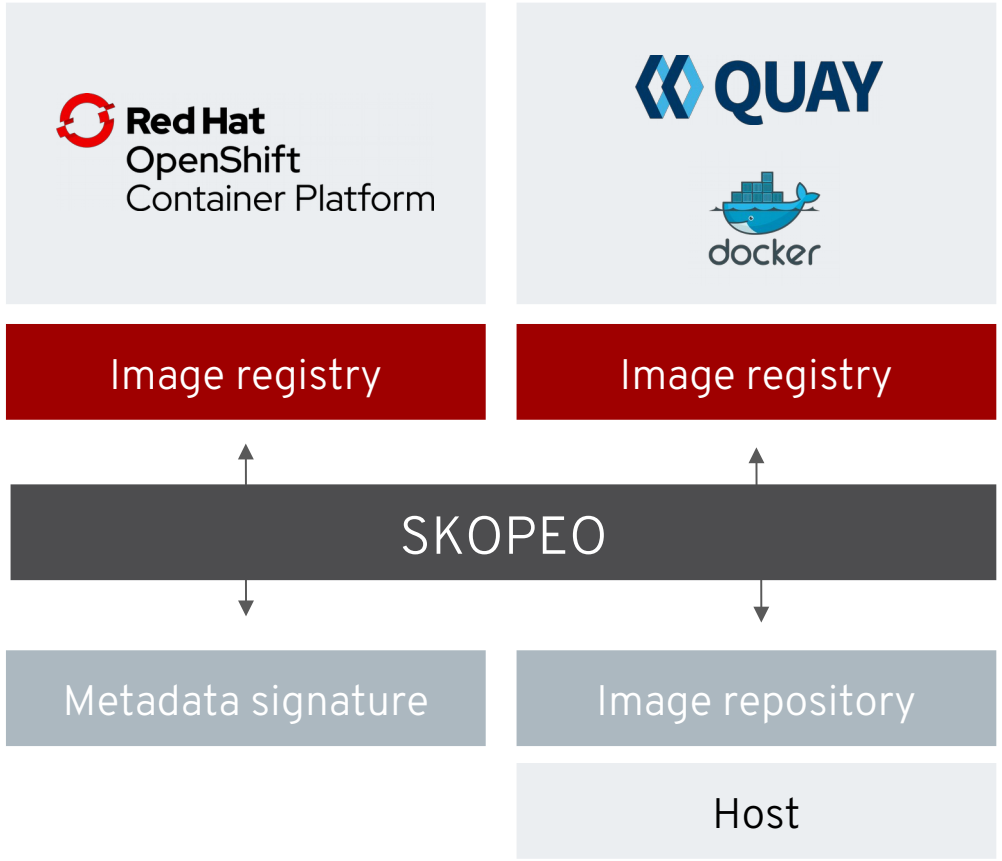


More control
Scriptable tooling for fine-grained image control, and maximum control starting from base or scratch images

Minimization of images
Elimination of unneeded dependencies by using host-based tools

Red Hat

Inspect and transport images with Skopeo



Inspect images remotely
Examine image metadata without needing to download

Publish and transfer images
Copy images from registries to hosts or directly between registries

Sign and verify images
Supports GPG key signing on publish

Red Hat

OCI Demo

```
yum install docker
```

Management

Red Hat