Containers 101

with Podman on Fedora 29

Alessandro Arrichiello
Solution Architect, Red Hat
alezzandro@gmail.com

WHO I AM?

I'm Alessandro Arrichiello, graduated in Computer Engineering at "Federico II" university of Naples.
I’m currently working as a Solution Architect for Red Hat.

I'm a very passionate GNU/Linux fan.
My first Red Hat Linux installation was at age of 14, after that I never left it and kept using Linux in home and worklife.

http://alezzandro.com

WHAT ABOUT ME AND FEDORA?

I use Fedora as primary Operating System for work/personal usage from 5+ years.

I love placing stickers all over my laptop and let my friends and colleagues guess the Open Source project behind the logo.

My favourite Window Manager is GNOME :)

```
[alex@lenny ~]$ cat /etc/fedora-release
Fedora release 29 (Twenty Nine)
```

WHAT ABOUT THE LOGO?

You will find the Red Hat logo in almost every slide because the content comes from Red Hat’s slidedeck

AGENDA

- What are Linux Containers?
- Deep dive in Containers Architecture
- Containers Runtimes
- Pull and Run Containers
- Managing Networking, Logging, Security and Persistent Storage
- System Services in Containers

LINUX CONTAINERS

WHAT ARE CONTAINERS?

It Depends Who You Ask

INFRASTRUCTURE

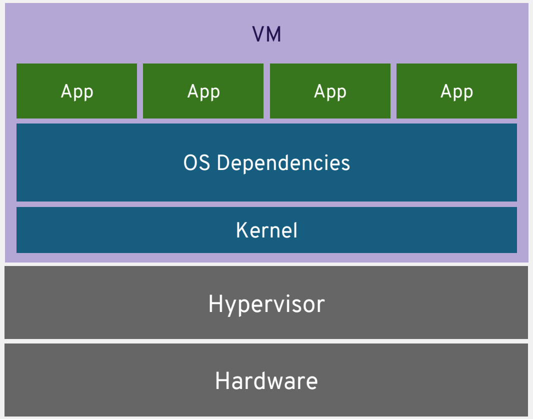
- Application processes on a shared kernel
- Simpler, lighter, and denser than VMs
- Portable across different environments

APPLICATIONS

- Package apps with all dependencies
- Deploy to any environment in seconds
- Easily accessed and shared

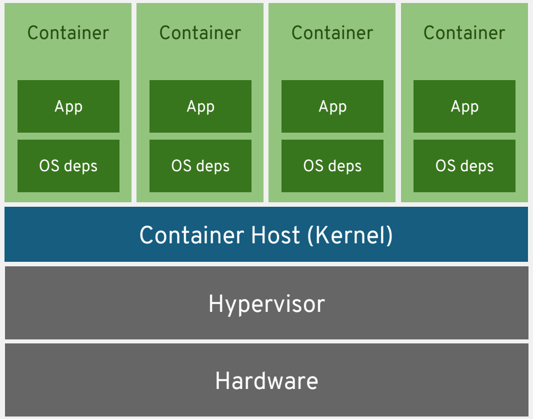
VIRTUAL MACHINES AND CONTAINERS

VIRTUAL MACHINES



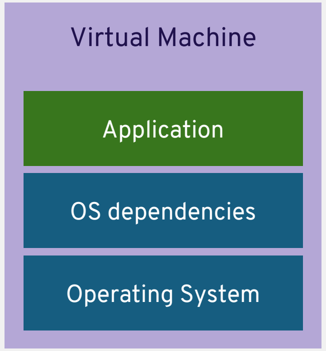
VM isolates the hardware

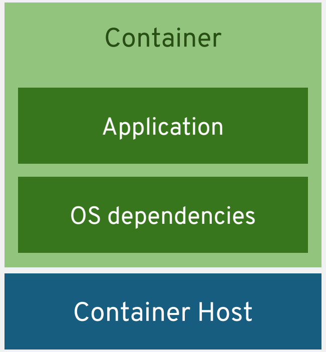
CONTAINERS



Container isolates the process

VIRTUAL MACHINES AND CONTAINERS

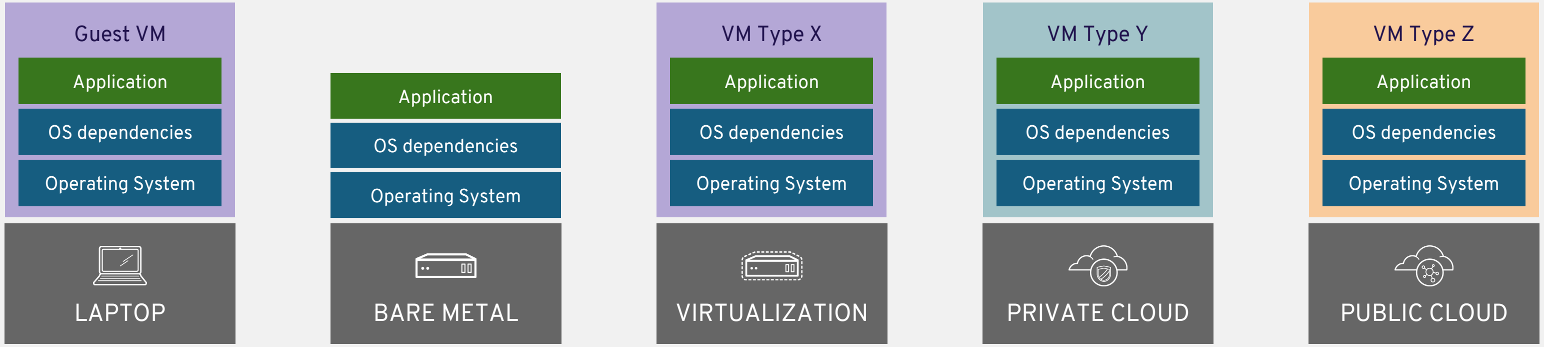




➕ VM Isolation	➕ Container Isolation
➖ Complete OS	➕ Shared Kernel
➖ Static Compute	➕ Burstable Compute
➖ Static Memory	➕ Burstable Memory
➖ High Resource Usage	➕ Low Resource Usage

APPLICATION PORTABILITY WITH VM

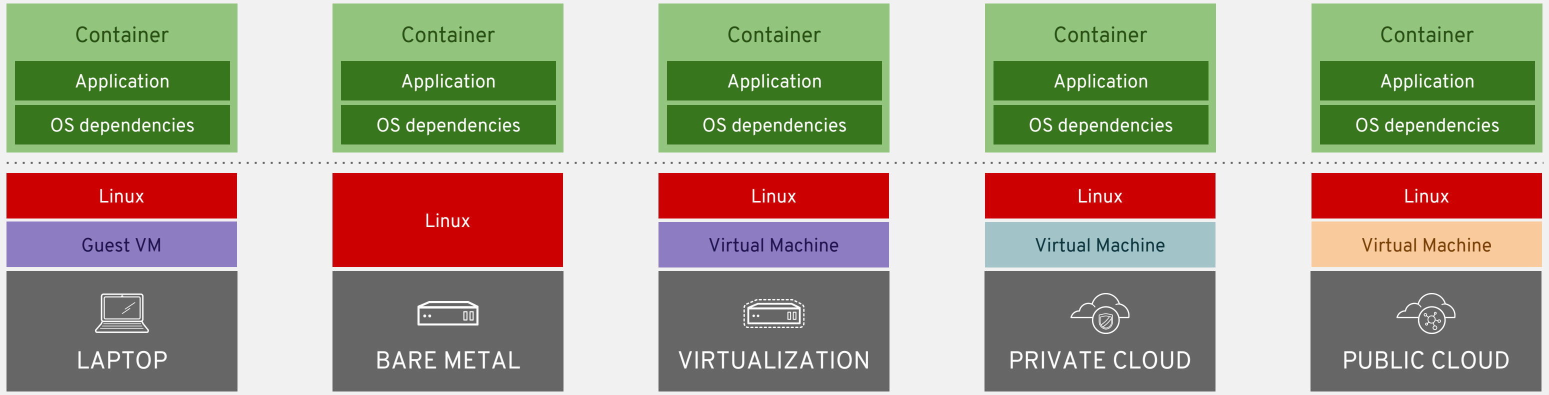
Virtual machines are NOT portable across hypervisor and do NOT provide portable packaging for applications



redhat.

APPLICATION PORTABILITY WITH CONTAINERS

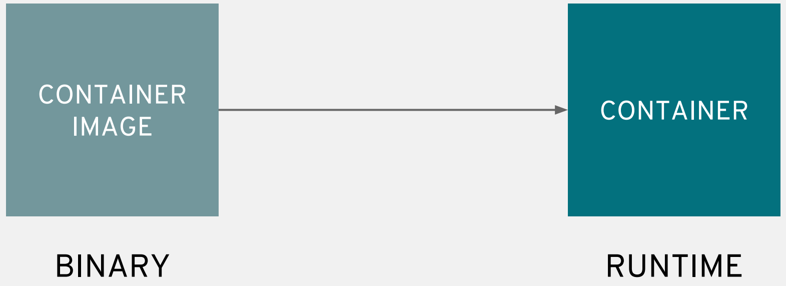
Linux Containers + Linux Host = Guaranteed Portability
Across Any Infrastructure



CONTAINERS DEEP DIVE

A container is the smallest compute unit

containers are created from container images

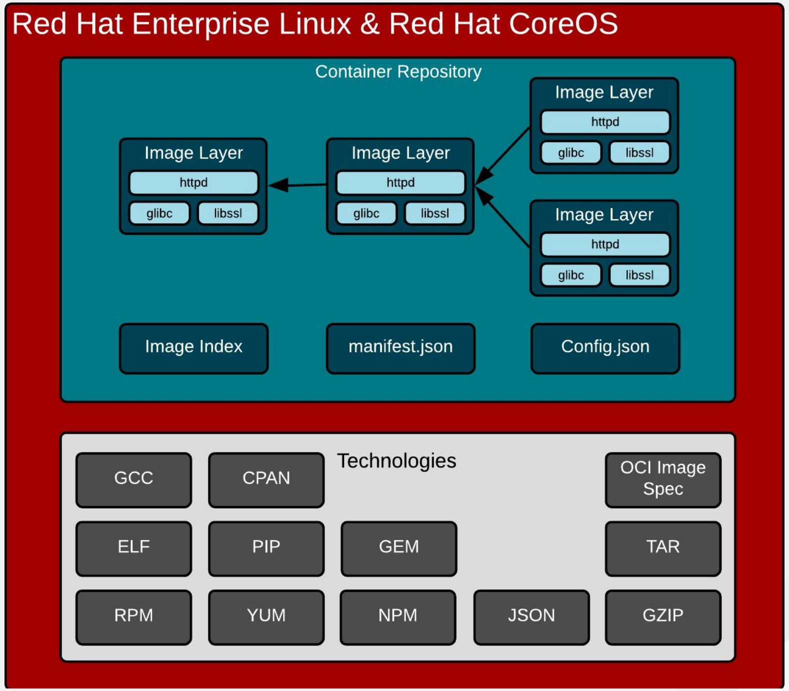


CONTAINER IMAGE

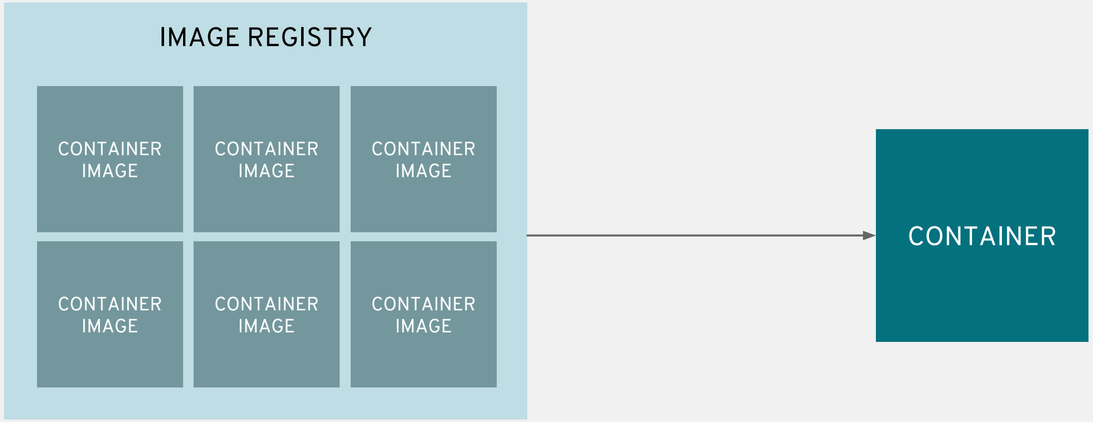
Open source code/libraries, in a Linux distribution, in a tarball

Even base images are made up of layers:

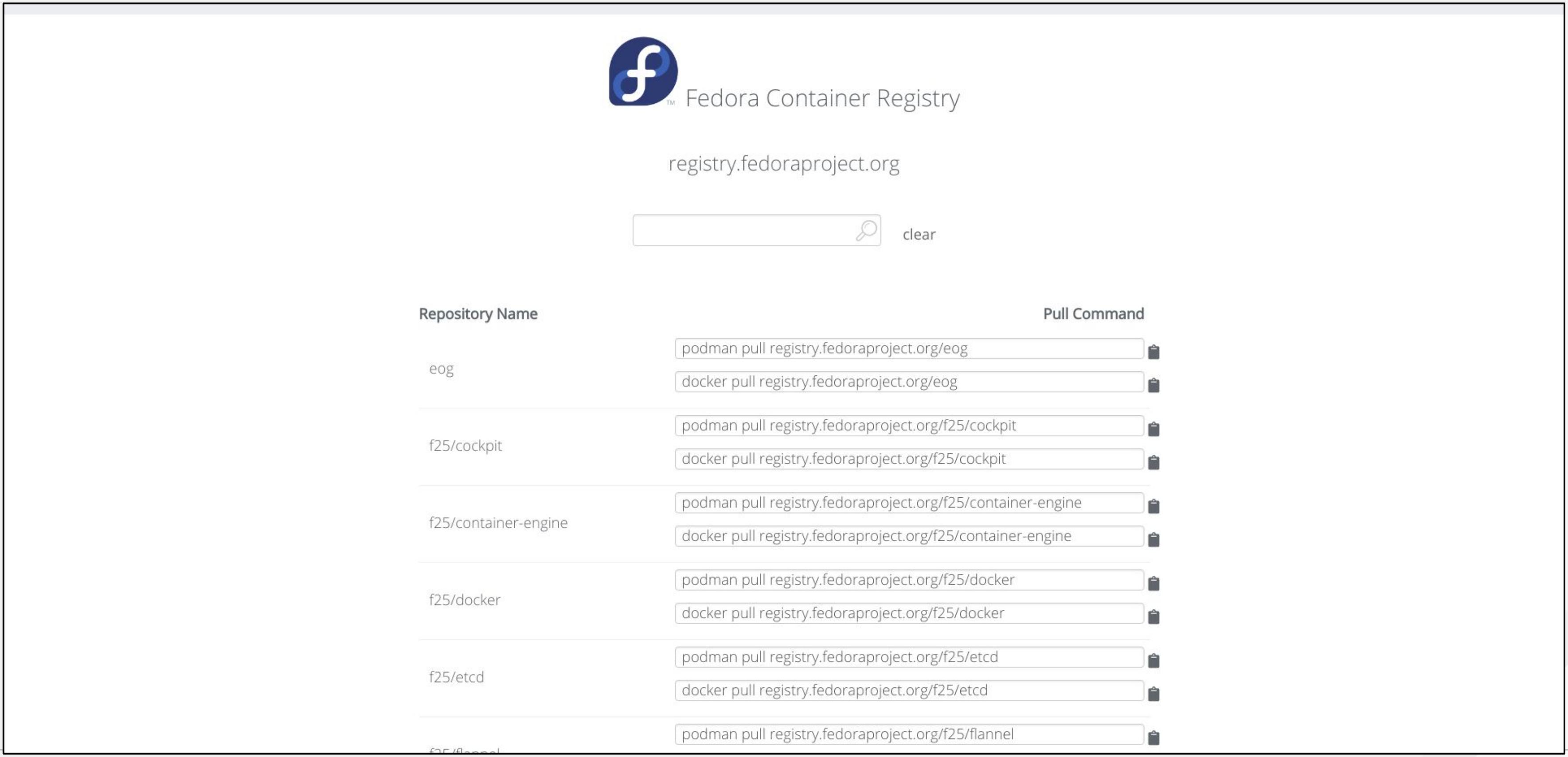
- Libraries (glibc, libssl)
- Binaries (httpd)
- Packages (rpms)
- Dependency Management (yum)
- Repositories (rhel7)
- Image Layer & Tags (rhel7:7.5-404)
- At scale, across teams of developers and CI/CD systems, consider all of the necessary technology



container images are stored in an image registry



Fedora Containers' Images Registry



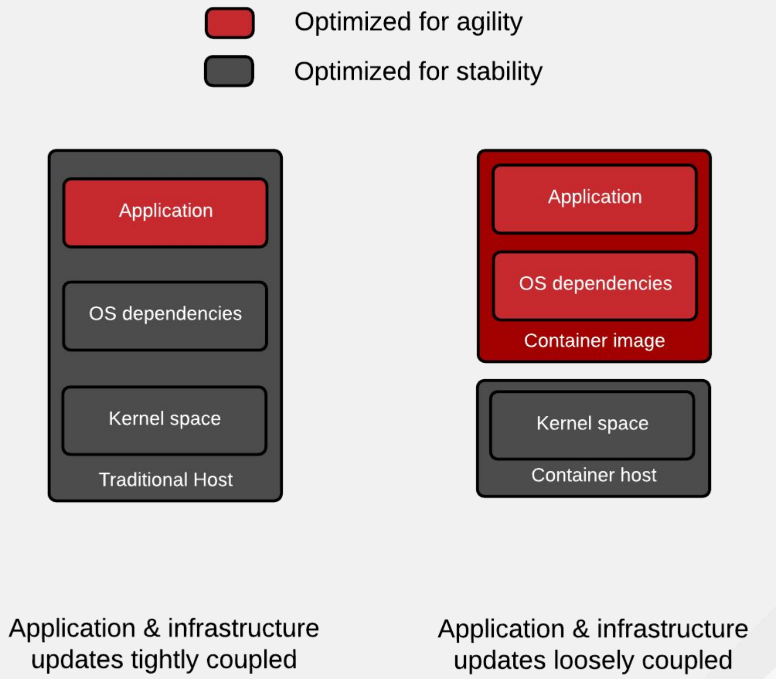
Repository Name	Pull Command
eog	podman pull registry.fedoraproject.org/eog
	docker pull registry.fedoraproject.org/eog
f25/cockpit	podman pull registry.fedoraproject.org/f25/cockpit
	docker pull registry.fedoraproject.org/f25/cockpit
f25/container-engine	podman pull registry.fedoraproject.org/f25/container-engine
	docker pull registry.fedoraproject.org/f25/container-engine
f25/docker	podman pull registry.fedoraproject.org/f25/docker
	docker pull registry.fedoraproject.org/f25/docker
f25/etcd	podman pull registry.fedoraproject.org/f25/etcd
	docker pull registry.fedoraproject.org/f25/etcd
	podman pull registry.fedoraproject.org/f25/flannel

REGISTRY SERVERS

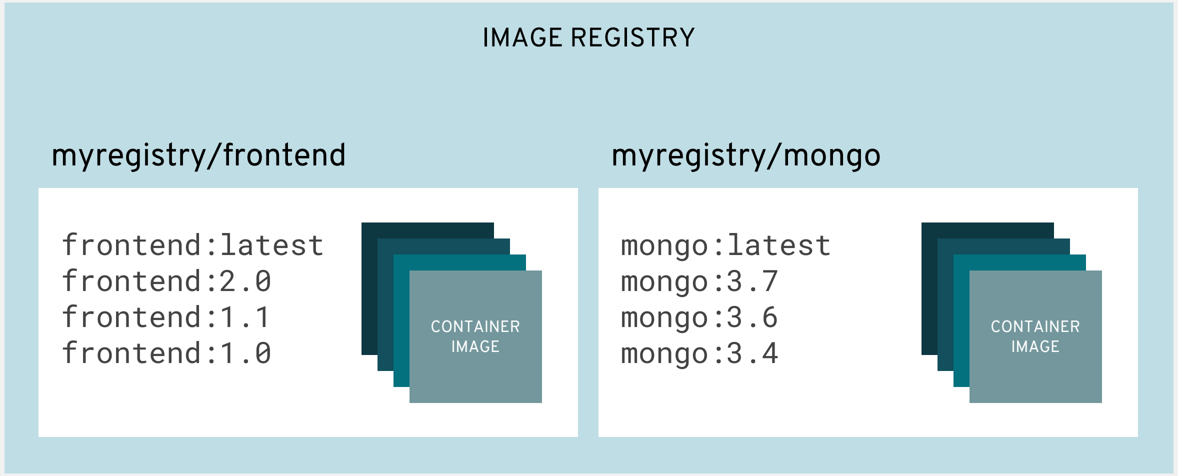
Better than virtual appliance market places :-)

Defines a standard way to:

- Find images
- Run images
- Build new images
- Share images
- Pull images
- Introspect images
- Shell into running container
- Etc, etc, etc



an image repository contains all versions of an image in the image registry

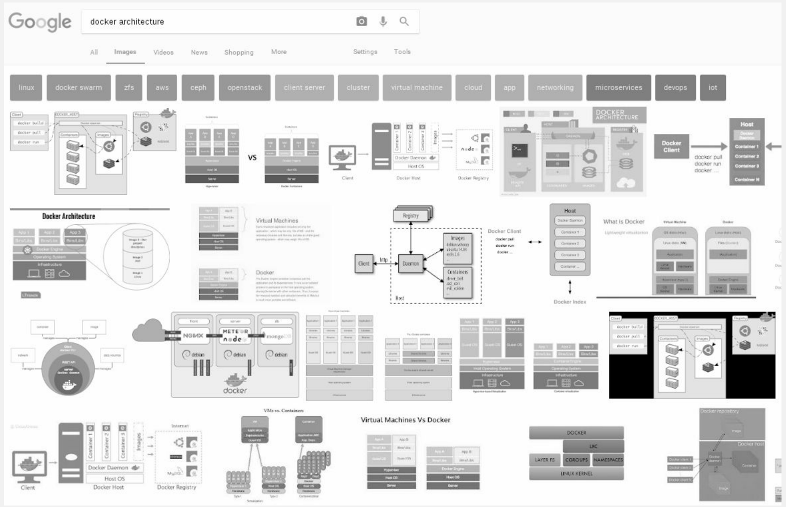


CONTAINERS DON’T RUN ON DOCKER

The Internet is WRONG :-)

Important corrections

- Containers do not run ON docker. Containers are processes - they run on a container host. Containers are Linux.
- The docker daemon is one of the many user space tools/libraries that talks to the kernel to set up containers

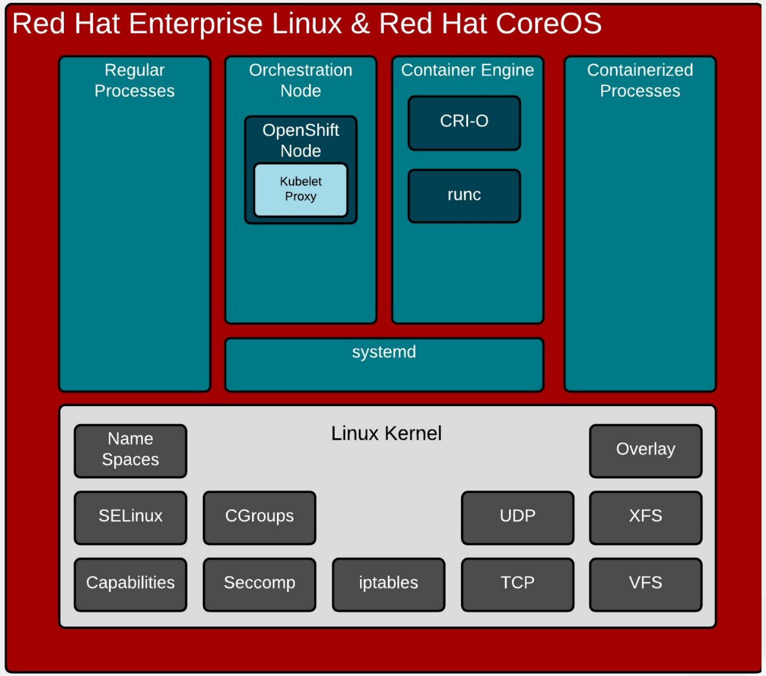


CONTAINER HOST

Regular processes, daemons, and containers all run side by side

Tightly coupled communication through the kernel - all or nothing feature support:

- Operating System (kernel)
- Container Runtime (runc)
- And other tools for orchestrate container in orchestrators like Kubernetes (CRI-O, Kubelet..)

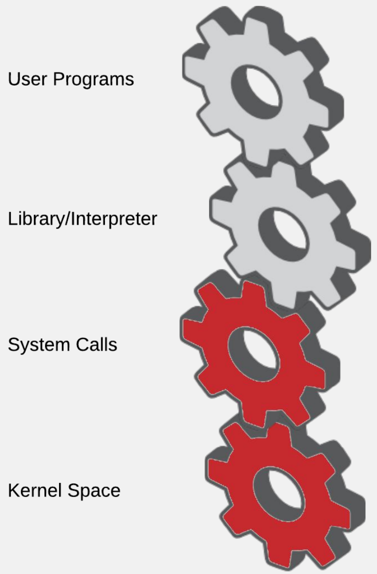


KERNEL

User space vs. kernel

The kernel is the gatekeeper for all access to resources and data structures:

- System calls
- Memory
- CPU
- Devices
- Drivers
- Filesystems

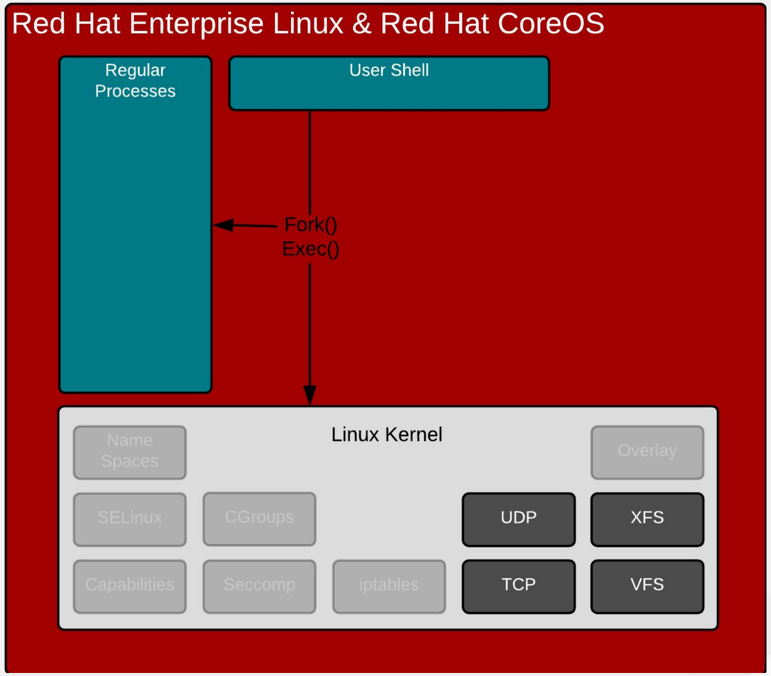


KERNEL

Creating regular Linux processes

Normal processes are created, destroyed, and managed with system calls:

- Fork() - Think Apache
- Exec() - Think ps
- Exit()
- Kill()
- Open()
- Close()
- System()

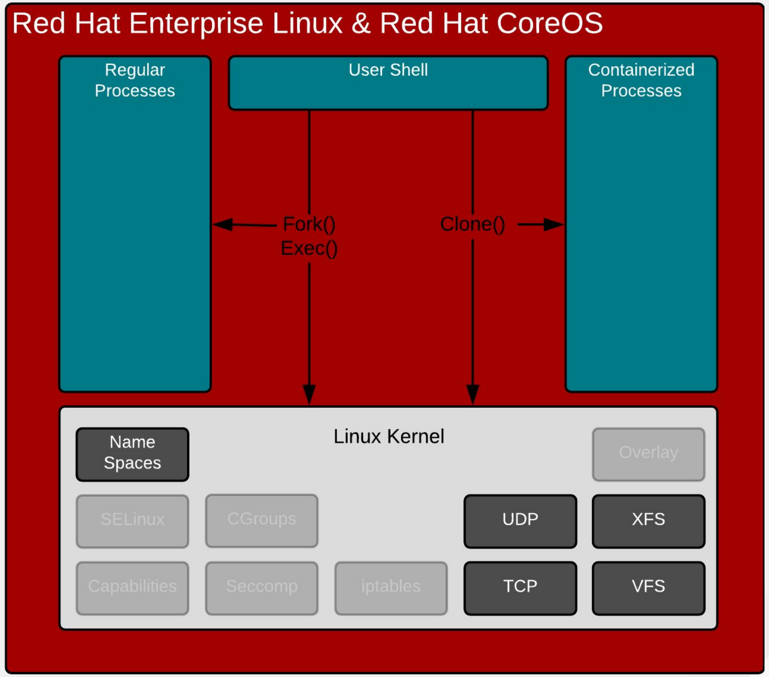


KERNEL

Creating “containerized” Linux processes

What is a container anyway?

- No kernel definition for what a container is - only processes
- Clone() - closest we have
- Creates namespaces for kernel resources
  - Mount, UTC, IPC, PID, Network, User
- Essentially, virtualized data structures

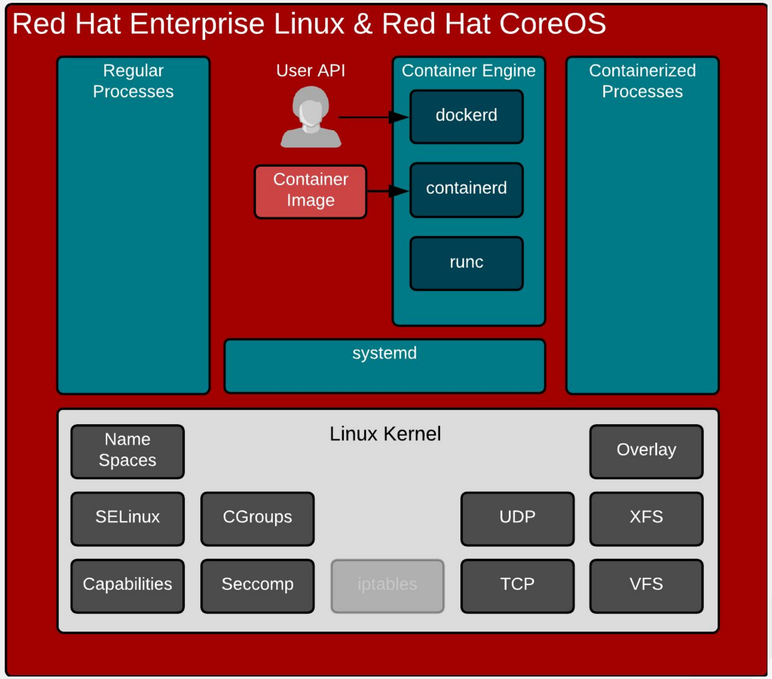


CONTAINER ENGINE

Provides an API prepares data & metadata for runc

Two major jobs:

- Provide an API - can be consumed by users or robots
- Prepares data & metadata from container image
  - Creates a manifest.json file
  - Graph driver decodes the container images layers - maps to filesystem (overlay or device mapper)
  - Prepares a directory to be mounted

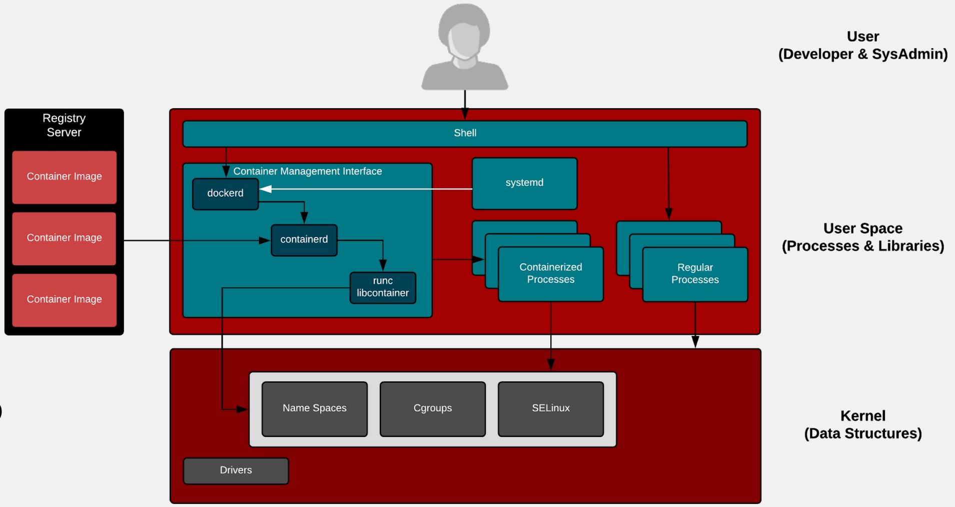


CONTAINER ENGINE

Regular processes, daemons, and containers all run side by side

In action:

- Takes command line options
- Creates and prepares manifest.json
- Pulls image
- Decodes image on disk (graph drivers)
- Hands directory and manifest.json to container runtime (runc)



CONTAINER RUNTIMES

EARLY CONCERNS WITH DOCKER

Enterprise Build and Runtime concerns

Since the early days users had concerns:

- Build requires a daemon
- Build requires a running container
- Build has secret handling issues
- Root/privileged concerns at runtime
- Regression for integration with container platforms (Kubernetes)

OPEN CONTAINER INITIATIVE

- Docker, Red Hat et al. June 2015
- Two Specifications
  - Runtime
    - How to run a “filesystem bundle” that is unpacked on disk
  - Image Format
    - How to create an OCI Image that contains sufficient information to launch the application on the target platform

OCI - RUNTIME

- Version 1.0 Released July 19th 2017
  - https://github.com/opencontainers/runtime-spec/releases/tag/v1.0.0
- runc - default implementation
- support other runtimes as they develop
- Docker 1.11 uses runc as the default back end
- **ocitool** https://github.com/opencontainers/ocitools
  - OCI runtime specification tools

OCI - IMAGE FORMAT

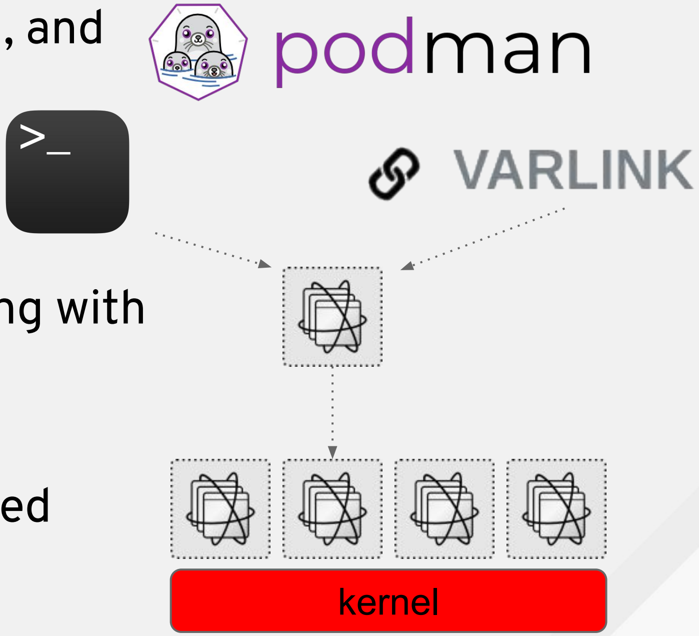
- Version 1.0 Released July 19th 2017
    - https://github.com/opencontainers/image-spec/releases/tag/v1.0.0
- Performed by a build system
- *Output includes* :
    - Image manifest
    - Filesystem serialization
    - Image configuration which includes information such as application arguments, environments, etc.

PODMAN

Podman is included in latest Fedora 29.

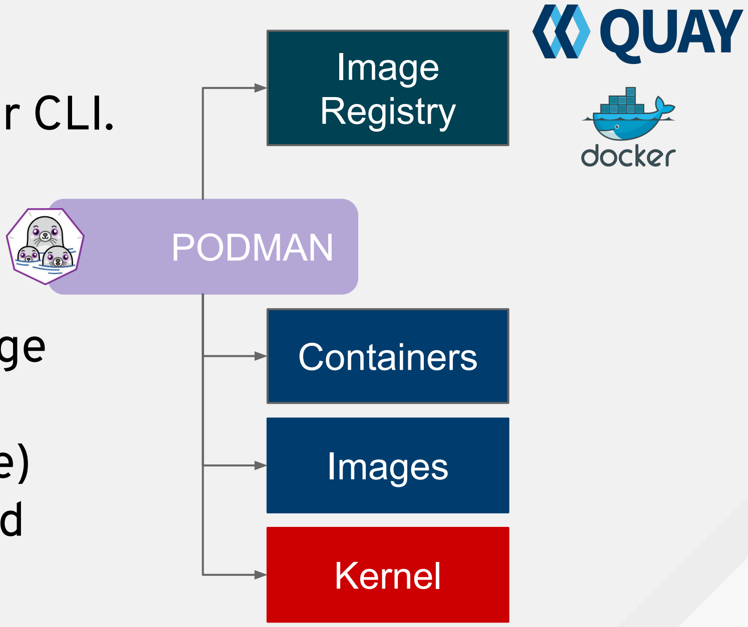
A daemon-less CLI/API for running, managing, and debugging OCI containers and pods

- Fast and lightweight
- Leverages runC
- Provides a “docker-like” syntax for working with containers
- Remote management API via Varlink
- Provides systemd integration and advanced namespace isolation



CONTAINER CLI

- Part of ProjectAtomic project on GitHub
- Client only tool that is based on the Docker CLI. (same+)
- Does not require a running daemon
- Utilizes :
  - containers/image & containers/storage
  - oci-runtime-tool/generate
  - runc (or other OCI compatible runtime)
- Shares state with other tool like: CRI-O and Buildah



EXAMPLE:
PULLING & RUNNING A CONTAINER

EXAMPLE: Pulling a container image

We start pulling down a container image and inspecting it for showing details.

```
[root@lenny ~]# podman pull registry.fedoraproject.org/f29/httpd
Trying to pull registry.fedoraproject.org/f29/httpd...Getting image source signatures
Copying blob sha256:281a37f51f750824a0addded649118d193f071a12bea2bca046a89e564df2da1
 85.68 MB / 85.68 MB [=====================================================] 16s
Copying blob sha256:ab0d48faadd2893c7cb2693ba352fafe60d7faf1fb5cf005164de77ecc340c66
 4.64 MB / 4.64 MB [========================================================] 1s
Copying blob sha256:e1bf69dce18d7eb90fdc5ee14f2170f274c5ae221589d323c7f780e0d507e9b7
 49.77 MB / 49.77 MB [======================================================] 6s
Copying config sha256:532763348c4e0991ea7eb439676675b0b9e5518db9a9f21b9862abfb533e6e83
 6.66 KB / 6.66 KB [========================================================] 0s
Writing manifest to image destination
Storing signatures
532763348c4e0991ea7eb439676675b0b9e5518db9a9f21b9862abfb533e6e83
[root@lenny ~]#
[root@lenny ~]# podman images
REPOSITORY                             TAG      IMAGE ID       CREATED       SIZE
registry.fedoraproject.org/f29/httpd   latest   532763348c4e   4 weeks ago   462MB
[root@lenny ~]#
[root@lenny ~]# podman inspect httpd
[
    {
        "Id": "532763348c4e0991ea7eb439676675b0b9e5518db9a9f21b9862abfb533e6e83",
...
```

EXAMPLE: Running a container image 1/3

We're now ready for starting it.. and seeing what will happen!

```
[root@lenny ~]# podman run httpd
=> sourcing 10-set-mpm.sh ...
=> sourcing 20-copy-config.sh ...
=> sourcing 40-ssl-certs.sh ...
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using
10.0.2.100. Set the 'ServerName' directive globally to suppress this message
[Tue Dec 04 19:21:29.834230 2018] [ssl:warn] [pid 1:tid 140546178489728] AH01909: 10.0.2.100:8443:0
server certificate does NOT include an ID which matches the server name
AH00558: httpd: Could not reliably determine the server's fully qualified domain name, using
10.0.2.100. Set the 'ServerName' directive globally to suppress this message
[Tue Dec 04 19:21:29.958497 2018] [ssl:warn] [pid 1:tid 140546178489728] AH01909: 10.0.2.100:8443:0
server certificate does NOT include an ID which matches the server name
[Tue Dec 04 19:21:29.959009 2018] [lbmethod_heartbeat:notice] [pid 1:tid 140546178489728] AH02282: No
slotmem from mod_heartmonitor
[Tue Dec 04 19:21:29.966405 2018] [mpm_event:notice] [pid 1:tid 140546178489728] AH00489:
Apache/2.4.37 (Fedora) OpenSSL/1.1.1-pre9 configured -- resuming normal operations
[Tue Dec 04 19:21:29.966464 2018] [core:notice] [pid 1:tid 140546178489728] AH00094: Command line:
'httpd -D FOREGROUND'
```

EXAMPLE: Running a container image 2/3

What about running it in background and check if webserver is working?

```
[root@lenny ~]# podman run --name myhttpservice -d httpd
f2ef0980dd67dd45e1ee0a0d05f5084a69a9be00bfa9ca3f9facd07ad57d84b7
[root@lenny ~]# podman ps
CONTAINER ID   IMAGE                                        COMMAND                     CREATED
STATUS               PORTS    NAMES
f2ef0980dd67   registry.fedoraproject.org/f29/httpd:latest   container-entrypoin...   2 minutes ago
Up 2 minutes ago            fervent_murdock
[root@lenny ~]#
[root@lenny ~]# podman inspect myhttpservice | grep -i ipaddr
            "SecondaryIPAddresses": null,
            "IPAddress": "10.88.0.4",

[root@lenny ~]# podman inspect myhttpservice    # search for exposed ports

[root@lenny ~]# curl 10.88.0.4:8080
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
      <head>
            <title>Test Page for the Apache HTTP Server on Fedora</title>
...
```

EXAMPLE: Running a container image 3/3

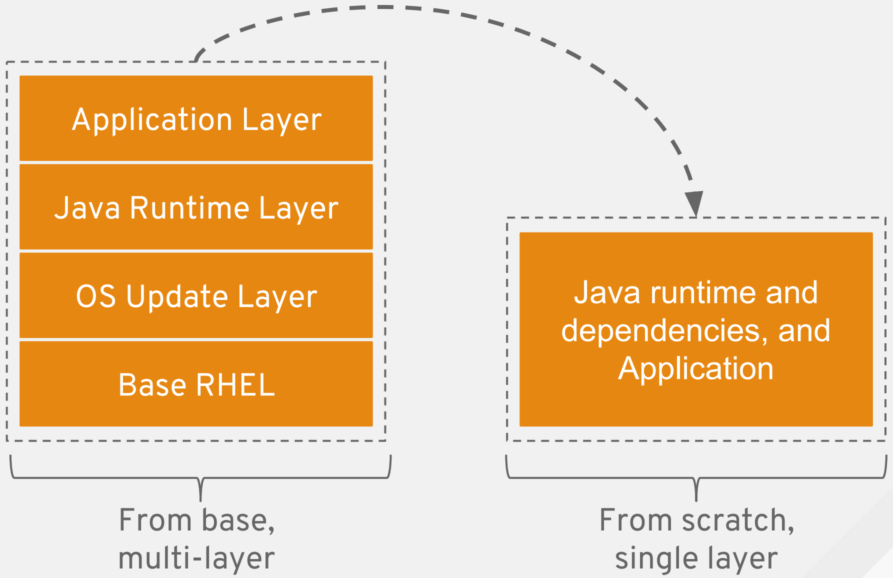
Make some edits and try to restart our container.. Is it really immutable? Yes it is!!

```
[root@lenny ~]# podman run -d httpd
[root@lenny ~]# podman help exec
[root@lenny ~]# podman exec -ti f2ef0980dd67 /bin/bash
bash-4.4$ echo "MySecretData" > my.data
bash-4.4$ cat my.data
MySecretData
bash-4.4$
bash-4.4$ exit
[root@lenny ~]# podman kill f2ef0980dd67
8df6cb35a24660aa797c3d326b2e90348238429a9a3478aba4e0e6beb8eec5ec
[root@lenny ~]# podman ps
[root@lenny ~]# podman run -d httpd
1aeac2d0951323754c0d9b27fe6a23fb4b8a98f466a3e1543933fb07a433e03c
[root@lenny ~]# podman ps
CONTAINER ID   IMAGE                                         COMMAND                  CREATED
STATUS             PORTS   NAMES
1aeac2d09513   registry.fedoraproject.org/f29/httpd:latest   container-entrypoin...   4 seconds ago
Up 4 seconds ago           wonderful_snyder
[root@lenny ~]# podman exec -ti 1aeac2d09513 /bin/bash
bash-4.4$ ls
bash-4.4$ id
uid=1001(default) gid=0(root) groups=0(root)
```

BUILD NEW CONTAINERS

WHY USE BUILDAH ?

- Builds OCI compliant images
- No daemon - no docker socket
- Does not require a running container
- Can use the hosts subscriptions and other secrets.
- Fine-grained control over the commands and content of layer(s)
- Single layer, from scratch images are made easy and it ensures limited manifest.
- If needed you can still maintain Dockerfile based workflow



SO WHAT DOES BUILDAH DO?

- *buildah from*
  - Build up a container root filesystem from an image or *scratch*.
- *buildah config*
  - Adjust defaults in the image's configuration blob.
- *buildah run*
  - Run a command in the container's filesystem using runc.
  - NOT like docker run. Like Dockerfile RUN.
- *buildah mount*
  - Mount the container's root filesystem on the host.
- *buildah commit*
  - Use the container's changes wrt its image to build a new image.

Want to know more?

Keep watching the next Fedora Classrooms for an upcoming session fully dedicated to Buildah!

https://fedoraproject.org/wiki/Classroom/F29

ISOLATION

EXAMPLE: Isolation matters!

Security relies on various operating system feature

```
[root@lenny ~]# podman exec -u root -ti 068f43308502 /bin/bash
bash-4.4# id
uid=0(root) gid=0(root) groups=0(root)
bash-4.4# dnf install --repo fedora -y iputils procps-ng
...
bash-4.4# ping google.com
bash-4.4# vi /etc/resolv.conf
bash-4.4# ping google.com
bash-4.4# date
...
bash-4.4# mv /etc/localtime /etc/localtime.bak
bash-4.4# ln -s /usr/share/zoneinfo/America/New_York /etc/localtime
bash-4.4# date
...
bash-4.4# ps aux
...
```

NETWORKING

EXAMPLE: Expose your services!

Containers isolation is really useful but we may want our services to be reachable by world!

```
[root@lenny ~]# podman run -d httpd
40255ac12df9ed7bb6f41cda697418021efed910d6d2c2ee85885fa0a60ec3db
[root@lenny ~]# podman ps
CONTAINER ID   IMAGE                                      COMMAND                  CREATED          STATUS
PORTS   NAMES
40255ac12df9   registry.fedoraproject.org/f29/httpd:latest   container-entrypoin...   2 seconds ago
Up 2 seconds ago           reverent_curran
[root@lenny ~]# podman inspect 40255ac12df9 | grep -i ipaddr
            "SecondaryIPAddresses": null,
            "IPAddress": "10.88.0.26",

[root@lenny ~]# curl 10.88.0.26:8080
...

[root@lenny ~]# podman kill 40255ac12df9 && podman rm 40255ac12df9
40255ac12df9ed7bb6f41cda697418021efed910d6d2c2ee85885fa0a60ec3db
40255ac12df9ed7bb6f41cda697418021efed910d6d2c2ee85885fa0a60ec3db

[root@lenny ~]# podman run -d -p 8080:8080 httpd
ef5941e6d20ba5fbd8a4fe0a35547c61bf000a2e3f1b7d9c1c4ad34f5a8e3502
[root@lenny ~]# curl localhost:8080
...
```

LOGGING

EXAMPLE: What about troubleshooting?

First of all we should inspect logs..

```
[root@lenny ~]# podman run -d -p 8080:8080 registry.fedoraproject.org/f29/httpd
147826570cf67595c71a96beab92956940e53cf8bc9914225023d33272255903
[root@lenny ~]# curl localhost:8080
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN" "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
...
[root@lenny ~]# podman ps
CONTAINER ID   IMAGE                                         COMMAND                   CREATED
STATUS              PORTS                         NAMES
147826570cf6   registry.fedoraproject.org/f29/httpd:latest   container-entrypoin...    33 seconds ago
Up 33 seconds ago   0.0.0.0:8080->8080/tcp        nostalgic_williams
[root@lenny ~]# podman logs 147826570cf6
...
[Sun Dec 09 11:06:09.733485 2018] [core:notice] [pid 1:tid 140709119864192] AH00094: Command line:
'httpd -D FOREGROUND'
[Sun Dec 09 11:06:27.767573 2018] [autoindex:error] [pid 30:tid 140708497323776] [client
10.88.0.1:42902] AH01276: Cannot serve directory /var/www/html/: No matching DirectoryIndex
(index.html) found, and server-generated directory index forbidden by Options directive
10.88.0.1 - - [09/Dec/2018:11:06:27 +0000] "GET / HTTP/1.1" 403 4650 "-" "curl/7.61.1"
[Sun Dec 09 11:06:30.870588 2018] [autoindex:error] [pid 29:tid 140708776216320] [client
10.88.0.1:42910] AH01276: Cannot serve directory /var/www/html/: No matching DirectoryIndex
(index.html) found, and server-generated directory index forbidden by Options directive
10.88.0.1 - - [09/Dec/2018:11:06:30 +0000] "GET / HTTP/1.1" 403 4650 "-" "curl/7.61.1"
```

PERSISTENT STORAGE

EXAMPLE: Containers are ephemeral..

So lets attach a persistent storage!

```
[root@lenny ~]# podman inspect httpd | grep -i user
            "User": "1001",
        "User": "1001"

[root@lenny ~]# podman run -ti registry.fedoraproject.org/f29/httpd /bin/bash
bash-4.4$ cat /etc/httpd/conf/httpd.conf | grep DocumentRoot
DocumentRoot "/var/www/html"

[root@lenny ~]# mkdir -p /opt/var/www/html
[root@lenny ~]# cd /opt/var/www/html
[root@lenny html]# wget --page-requisites --convert-links https://registry.fedoraproject.org/
[root@lenny html]# cd
[root@lenny ~]# chown 1001 -R /opt/var

[root@lenny ~]# podman run -d --name myhttpservice -p 8080:8080 -v
/opt/var/www/html:/var/www/html:Z registry.fedoraproject.org/f29/httpd

[root@lenny ~]# podman logs -f myhttpservice
...

### Open http://localhost:8080/registry.fedoraproject.org in your Web Browser
```

CONTAINERIZED SYSTEM SERVICES

EXAMPLE: System Services in Containers

Containers are portable and ready-to-use unit: Why not use them as system services?

```
[root@lenny ~]# vi /etc/systemd/system/myhttpservice.service
[Unit]
Description=Just and http service with Podman Container

[Service]
Type=simple
TimeoutStartSec=30s
ExecStartPre=-/usr/bin/podman rm "myhttpservice"

ExecStart=/usr/bin/podman run --name myhttpservice -p 8080:8080 -v /opt/var/www/html:/var/www/html:Z
registry.fedoraproject.org/f29/httpd

ExecReload=-/usr/bin/podman stop "myhttpservice"
ExecReload=-/usr/bin/podman rm "myhttpservice"
ExecStop=-/usr/bin/podman stop "myhttpservice"
Restart=always
RestartSec=30

[Install]
WantedBy=multi-user.target

[root@lenny ~]# systemctl daemon-reload
```

https://developers.redhat.com/blog/2018/11/29/managing-containerized-system-services-with-podman

EXAMPLE: System Services in Containers

Containers are portable and ready-to-use unit: Why not use them as system services?

```
[root@lenny system]# systemctl status myhttpservice.service
● myhttpservice.service - Just and http service with Podman Container
   Loaded: loaded (/etc/systemd/system/myhttpservice.service; disabled; vendor preset: disabled)
   Active: inactive (dead)
[root@lenny system]# systemctl start myhttpservice.service
[root@lenny system]# systemctl status myhttpservice.service
● myhttpservice.service - Just and http service with Podman Container
   Loaded: loaded (/etc/systemd/system/myhttpservice.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2018-12-11 18:05:55 CET; 5s ago
  Process: 16248 ExecStartPre=/usr/bin/podman rm myhttpservice (code=exited, status=125)
 Main PID: 16269 (podman)
    Tasks: 12 (limit: 4915)
   Memory: 9.8M
   CGroup: /system.slice/myhttpservice.service
           └─16269 /usr/bin/podman run --name myhttpservice -p 8080:8080 -v
/opt/var/www/html:/var/www/html:Z registry.fedoraproject.org/f29/httpd
...
Dec 11 18:05:56 lenny podman[16269]: [Tue Dec 11 17:05:56.412372 2018] [core:notice] [pid 1:tid
140083483450752] AH00094: Command line: 'httpd -D FOREGROUND'
...

### Open http://localhost:8080/registry.fedoraproject.org in your Web Browser
```

https://developers.redhat.com/blog/2018/11/29/managing-containerized-system-services-with-podman

EXAMPLE: System Services in Containers

Containers are portable and ready-to-use unit: Why not use them as system services?

Want to know more?
Take a look at the article I wrote on RH Dev Blog:

https://developers.redhat.com/blog/2018/11/29/managing-containerized-system-services-with-podman

LINKS & DOCS

Links & Documentation

- Intro to Podman (Red Hat Enterprise Linux 7.6 Beta)
  https://developers.redhat.com/blog/2018/08/29/intro-to-podman/
- Containers without daemons: Podman and Buildah available in RHEL 7.6 and RHEL 8 Beta
  https://developers.redhat.com/blog/2018/11/20/buildah-podman-containers-without-daemons/
- Managing containerized system services with Podman
  https://developers.redhat.com/blog/2018/11/29/managing-containerized-system-services-with-podman
- Fedora Registry: https://registry.fedoraproject.org/
- Podman main page: https://podman.io/
- Podman Github project: https://github.com/containers/libpod

Keep watching project page! Podman is continuously evolving!
We're now at version 0.12.1

CREDITS

Thanks to..

I want really thanks all the people that help me out filling this slidedeck:

- OpenShift BU team
- Scott McCarty
- Thomas Cameron and William Henry
- Fedora’s team!

Thank You!

Contact:
alezzandro@gmail.com