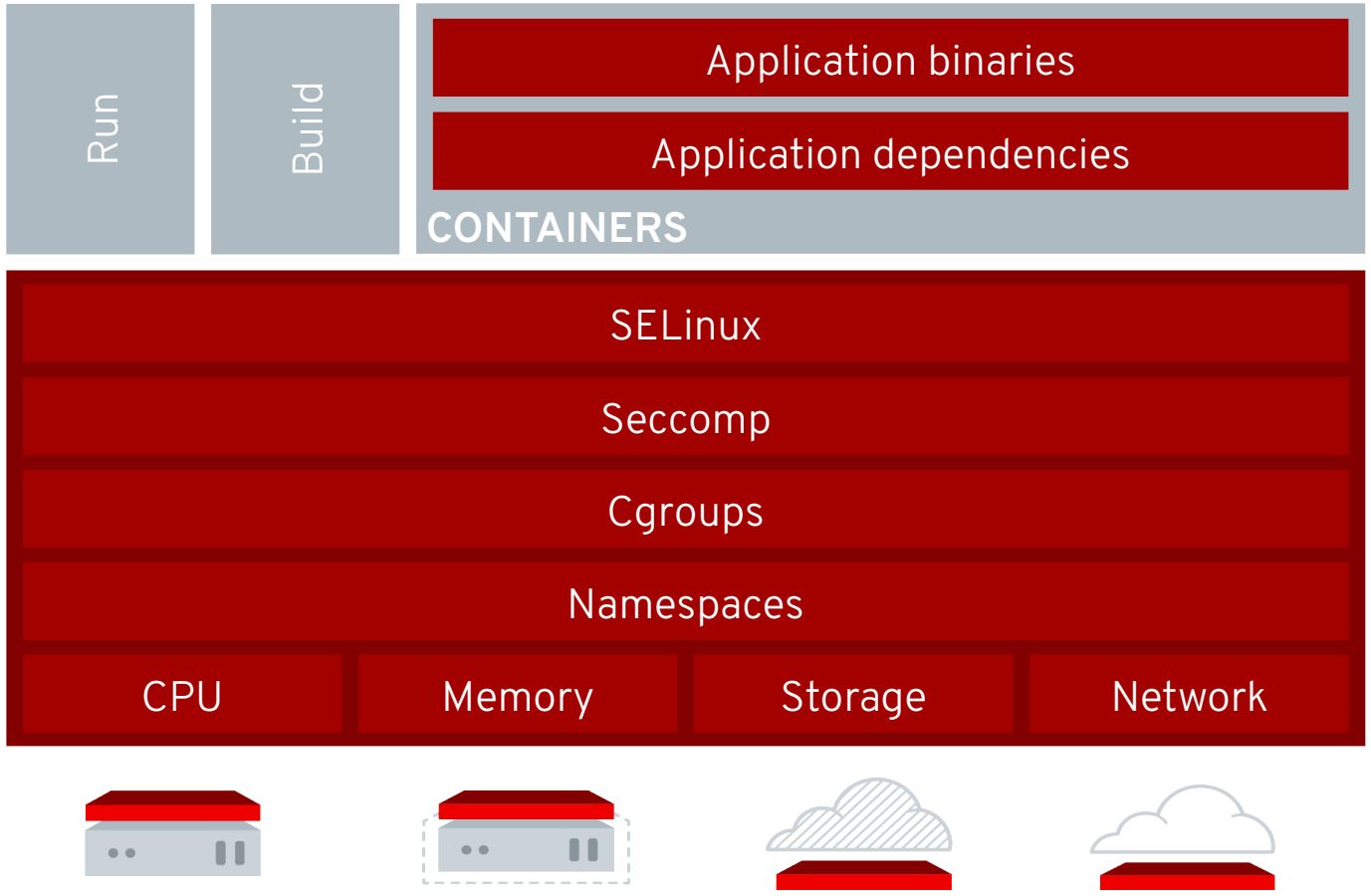
Red Hat
Enterprise Linux

Podifing Your Docker Container

Marc Skinner
Principal Solutions Architect

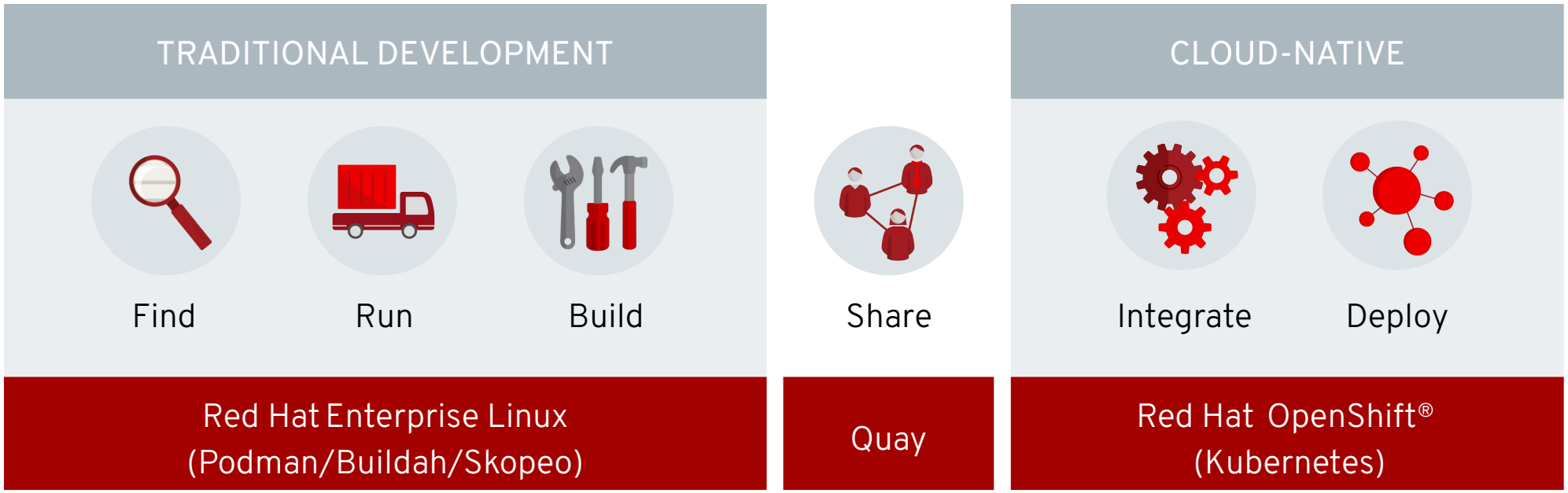
Containers

Containers are Linux



Red Hat

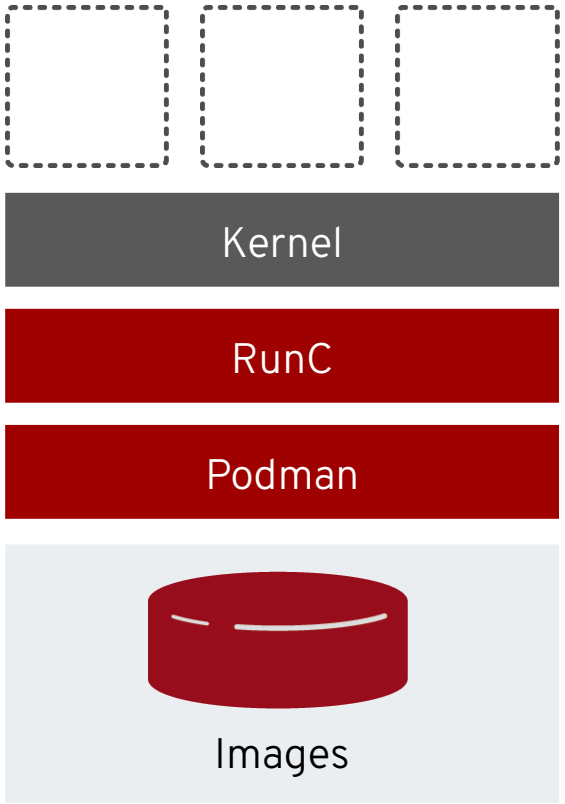
Powering the adoption of containerized workloads



Red Hat

Container Tooling Overview

Manage containers with Podman

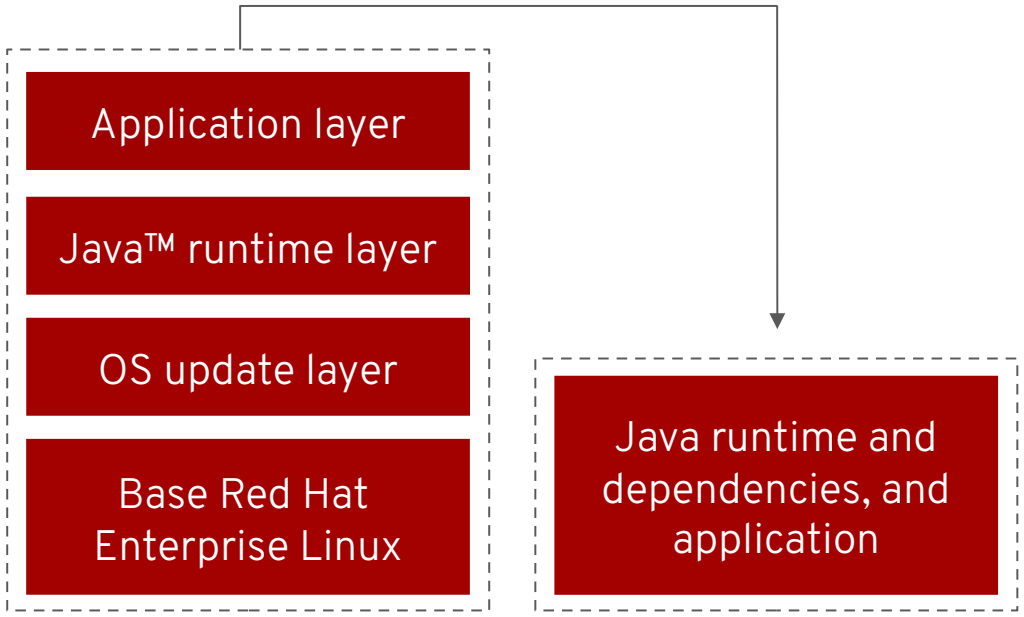


Fast and lightweight
No daemons required

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



From base, multilayer

From scratch, single layer

More control

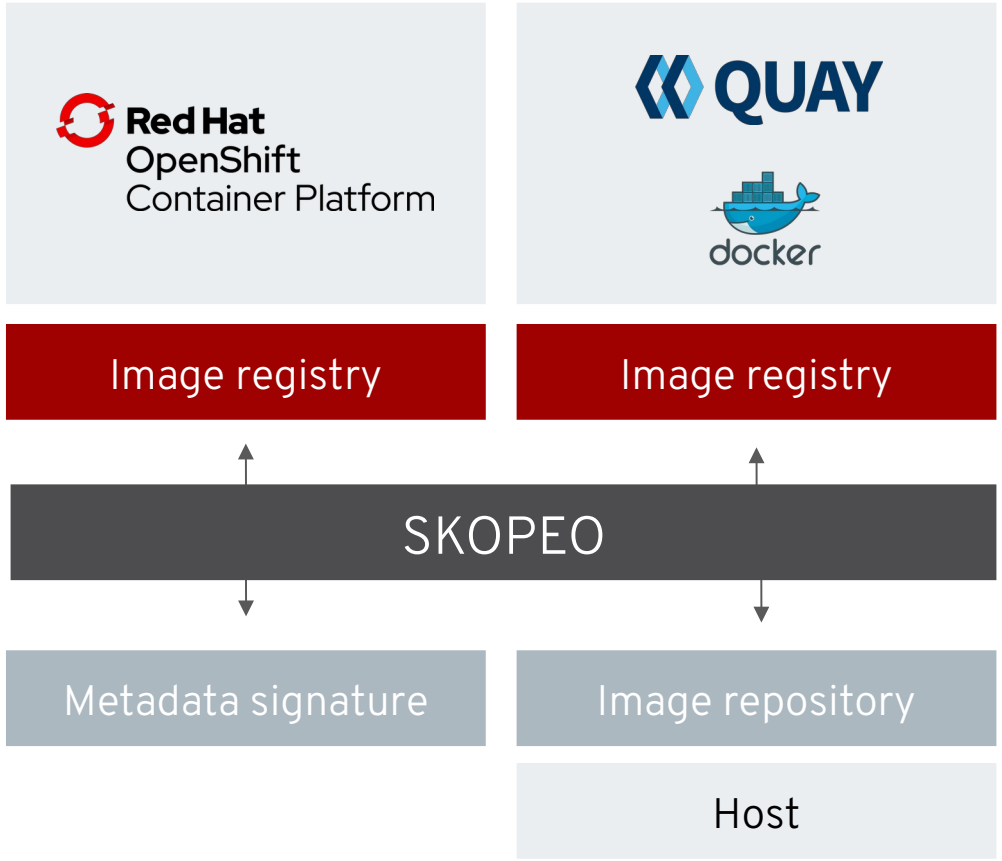
Scriptable tooling for fine-grained image control, and maximum control starting from base or scratch images

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

Container Tooling Details

Wait, no Docker?

- Docker and the Docker daemon are not needed
- Podman is a replacement
- We move from a centralized Docker daemon to a process runtime
- Client/Server vs Fork/Exec model

- No single point of failure
- No more orphaned processes if daemon fails
- Ability to run / build containers as unprivileged users

Are you sure it will work?

- Docker commands work with Podman

  # systemctl stop docker

  # alias docker=podman

- Docker images are compatible with Podman
- Podman is available on RHEL 7.6 or later

- Docker stores images in:

  “/var/lib/docker”

- Podman stores images in :

  “/var/lib/containers”

  “~/.local/share/containers”

What else should I know about Podman?

- Podman follows the Open Containers Initiative (OCI) standards
- Podman can create Kubernetes PODs

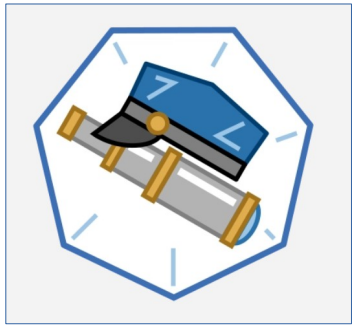
  # podman pod
- Podman can generate Kubernetes YAML files

  # podman generate kube PODID

- Podman has a REST API
- Podman supports rootless containers
  - Supports non-root users
- Ansible Collection: containers.podman

What is this Buildha?

- Build OCI containers without requiring a full container runtime or daemon
- Build from scratch
- Build from an existing container image
- Build from a Dockerfile
- Build as non-root



Skopeo – the Greek word for inspect

- Developed to easily inspect a remote container image – JSON output
- Evolved into a great tool for moving images between different types of container storage
  - Docker.io
  - Quay.io
  - Local filesystem

- Great for CI/CD
- One step versus three steps

```
# skopeo copy REGSRC/IMAGE REGDST/IMAGE
-----------------------------
# podman pull OLDIMAGE
# podman tag OLDIMAGE NEWIMAGE
# podman push NEWIMAGE
```

Skopeo JSON metadata

- Inspect a container image without downloading

  # skopeo inspect --config docker://registry.access.redhat.com/ubi8/ubi

  # skopeo inspect --config dir:/home/developer/ubi


```
{
    "created": "2021-06-02T19:27:22.256235Z",
    "architecture": "amd64",
    "os": "linux",
    "config": {
  "maintainer": "Red Hat, Inc.",
  "name": "ubi8",
  "release": "203.1622660121",
  "summary": "Provides the latest release of Red Hat Universal Base Image 8.",
```

U-B-I = Universal Base Image

- OCI-compliant container image build on RHEL 7 or 8
- No subscription required
- Inherits support subscription of running container host
- Red Hat patches, curates, publishes base images

- Four base image options:
  - ubi8/ubi-micro
  - ubi8/ubi-minimal
  - ubi8/ubi
  - ubi8/ubi-init

How small are micro and minimal? Can we get under 100MB?

ubi8/ubi-micro

- Smallest possible UBI image
- No package manager
- Must use buildha or external mechanism to update/add content

ubi8/ubi-minimal

- Designed for applications that contain their own dependencies (python, node.js, .NET, etc.)
- Minimized pre-installed content
- microdnf - Minimal package manager (install, update, and remove)
- No suid binaries

What about ubi vs init?

ubi8/ubi

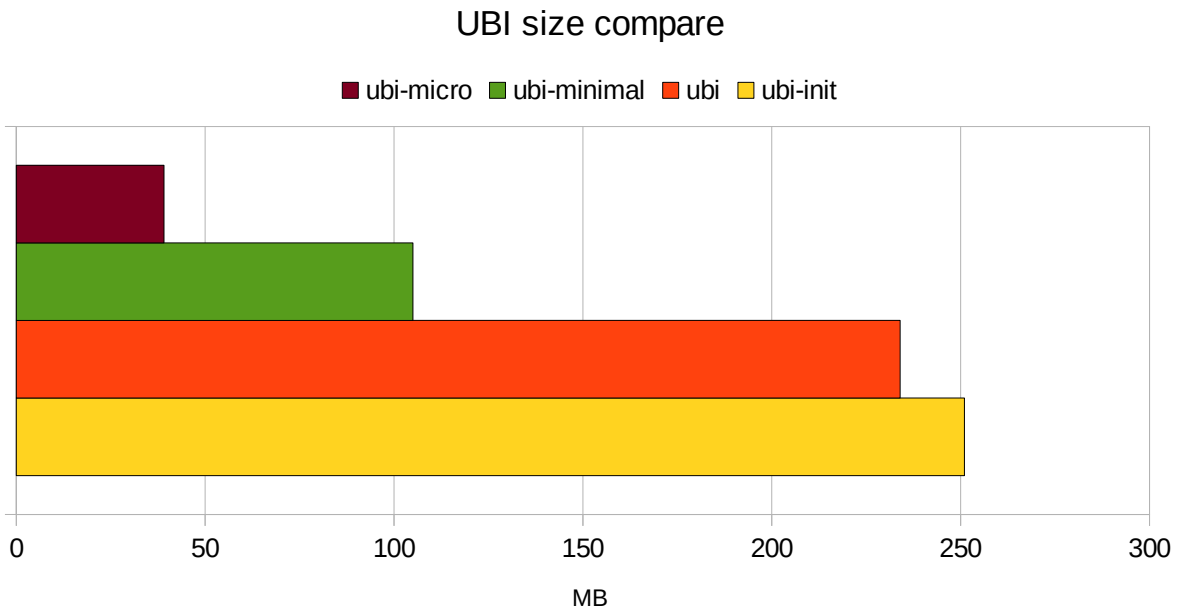
- Also called Platform
- Use for any application that runs on RHEL
- Unified, OpenSSL crypto stack
- Full YUM stack
- Includes useful basic OS tooling (tar, gzip, vi, etc.)

ubi8/ubi-init

- Known as Multi-Service
- Eases the ability to run multiple services from a single container
- Based on the ubi8/ubi image
- Configured to run systemd on start
- Allows you to enable services at build time

Let's compare the actual sizes

- ubi8/ubi-micro : 39.1 MB
- ubi8/ubi-minimal : 105 MB
- ubi8/ubi : 234 MB
- ubi8/ubi-init : 251 MB



Red Hat

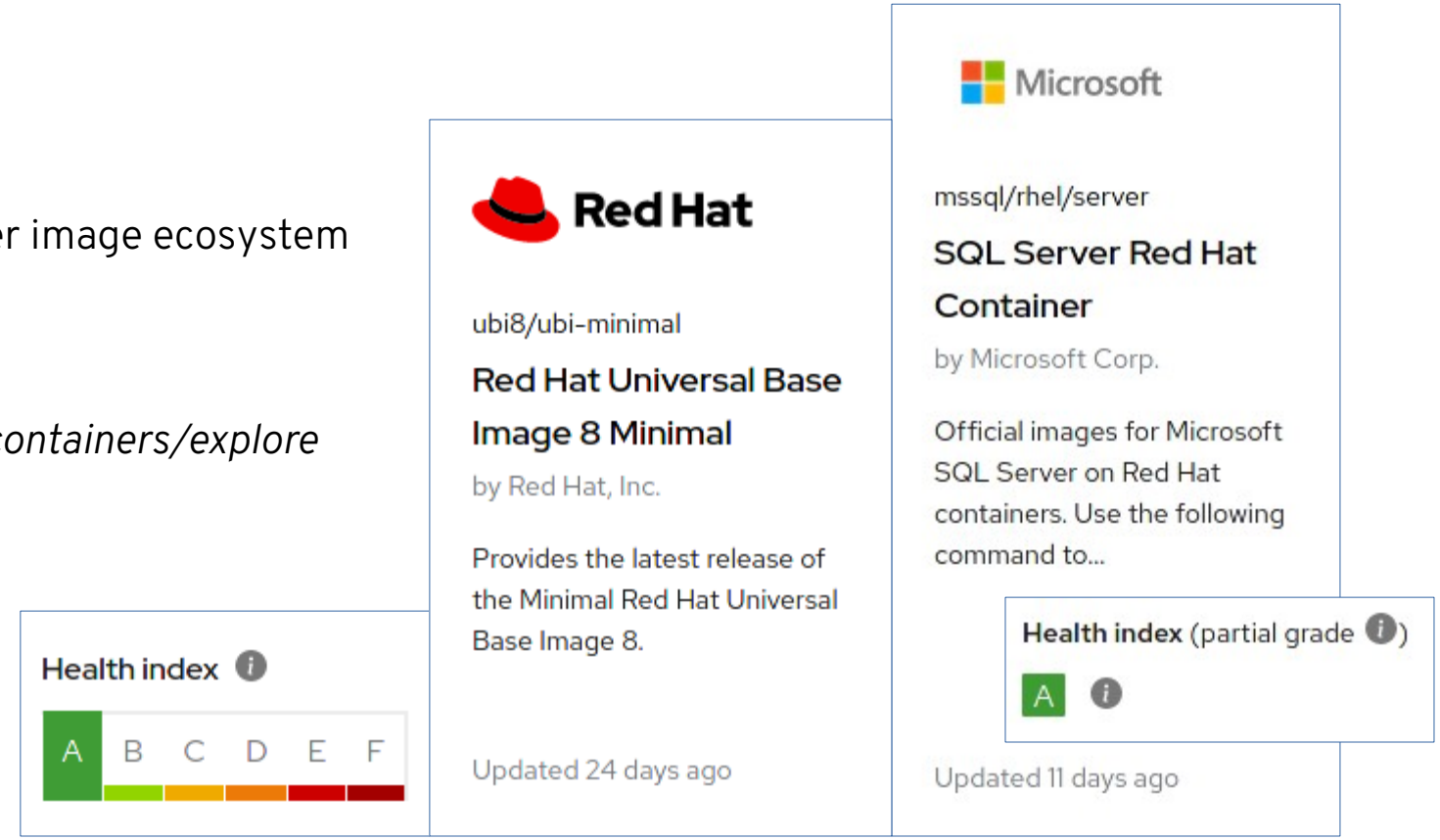
Wait … I'm still on RHEL 7

- RHEL 7 only has 3 base images – no micro image
  - ubi7/ubi-minimal
  - ubi7/ubi
  - ubi7/ubi-init

Where do I find trusted containers?

- Red Hat maintains a curated container image ecosystem catalog of more than 5,000

  https://catalog.redhat.com/software/containers/explore



Red Hat

How do I download the UBI?

- Use the Red Hat Container Catalog via a web browser:

  https://catalog.redhat.com/software/containers/explore
- Use podman to pull directly to your local registry:

  # podman pull registry.access.redhat.com/ubi7/ubi:latest

  # podman pull registry.access.redhat.com/ubi8/ubi:latest
- Use skopeo to copy directly to your local filesystem:

  # skopeo copy docker://registry.access.redhat.com/ubi8/ubi dir:/home/developer/ubi8
- Use skopeo to copy directly to you local registry:

  # skopeo copy docker://registry.access.redhat.com/ubi8/ubi docker://localhost:5000/ubi8/ubi

Let's run a Docker image on Podman, please

Download Percona Monitoring and Management 2

NOTE: Percona Monitoring and Management (PMM) employs a client/server model. You must download and install both the client and server applications. The directions for doing this are in the documentation.

Version:

Percona Monitoring and Management 2 2.18.0

Software:

Server - Docker Image

pmm-client-2.18.0.docker	115.5 MB
pmm-client-2.18.0.sha256sum	91 bytes
pmm-server-2.18.0.docker	395.5 MB
pmm-server-2.18.0.sha256sum	91 bytes

:s/docker/podman/

- Download docker image from docker.io

  # podman pull percona/pmm-server:2

- Create persistent storage mount point

  # podman create -v /srv --name pmm-data percona/pmm-server:2 /bin/true

- Start container

  # podman run -d -p 80:80 -p 443:443 --volumes-from pmm-data --name pmm-server --restart always percona/pmm-server:2

1. Pull the image.

```
docker pull percona/pmm-server:2
```

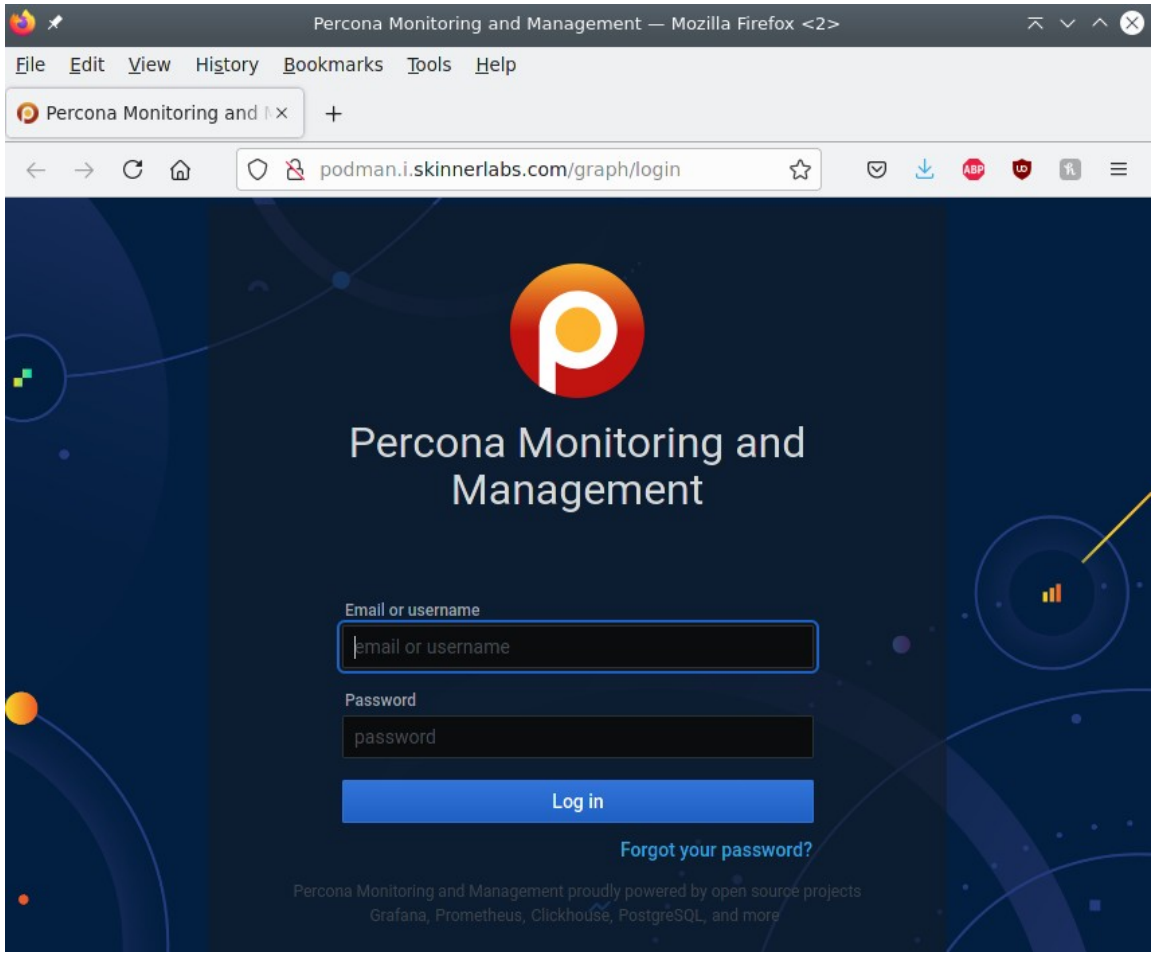
2. Create a persistent data container.

```
docker create --volume /srv \
--name pmm-data \
percona/pmm-server:2 /bin/true
```

3. Run the image.

```
docker run --detach --restart always \
--publish 443:443 \
--volumes-from pmm-data \
--name pmm-server \
percona/pmm-server:2
```

Test your container application



Red Hat

Create a systemd unit file to autostart your containerized application

- Grab the container ID of the running container

  # podman ps
- Generate systemd service file

  # podman generate systemd 4e860abd9941 > pmm-server.service

  # cp pmm-server.service /etc/systemd/system/

  # systemctl enable pmm-server
- Check out what is running in the container with podman top!

  # podman top pmm-server

Show me some rootless!

- Update “user.max_user_namespaces” via sysctl
- Create login for a normal user
- Remember: Podman local storage will be at ~/.local/share/containers/storage/
- Make sure rootless is working; check the namespace map

```
# podman unshare cat /proc/self/uid_map
0        1001    0
1  1000000    65536
```

- Check your UID

```
# id
uid=1001(test) gid=1001(test) ...
```

I want the Buildah steps

```
# buildah from ubi8
# buildah run ubi8-working-container yum install httpd -y
# echo "Hello from Red Hat" > index.html
# buildah copy ubi8-working-container index.html /var/www/html/index.html
# buildah config --entrypoint "/usr/sbin/httpd -DFOREGROUND" ubi8-working-container
# buildah commit ubi8-working-container rhug-demo-website
# podman run --name rhug-demo -dt -p 8080:80/tcp localhost/rhug-demo-website
# curl http://podman.i.skinnerlabs.com:8080
Hello from Red Hat
```

Running Information

- View container information

  # podman ps

  # podman top -l (-l = latest)

  # podman top [NAME] or [CONTAINERID]

  # podman inspect [NAME] or [CONTAINERID]

Passing environmental variables

- Pass variables

```
# podman run -e 'ACCEPT_EULA=Y' -e 'SA_PASSWORD=yourStrong(!)Password' -p 1433:1433 -d
mcr.microsoft.com/mssql/rhel/server:2019-latest
```

Clean up

```
# podman volume ls
# podman volume rm VOLUME_ID
# podman system prune --all --force
# podman stop --all
# podman rmi --all --force
```

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