Fedora Docker Layered Image Build Service

PRESENTED BY:
Adam Miller
Fedora Engineering, Red Hat

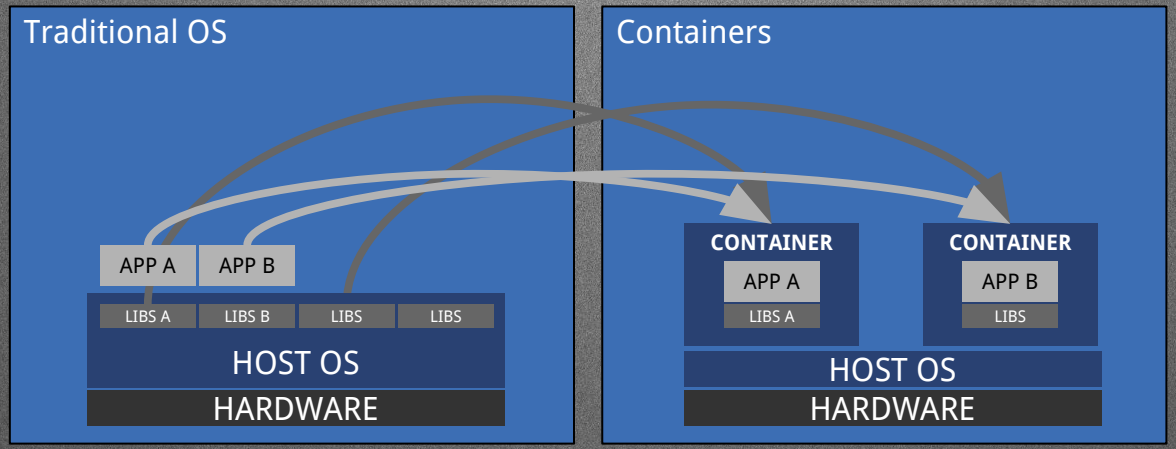
Today's Topics

- Define “containers” in the context of Linux systems
  - Brief History/Background
  - Container Implementations in Linux
- Docker
- Docker Build (Dockerfile)
- Release Engineering
- Docker Layered Image Build Service
  - OpenShift
  - OpenShift Build Service (OSBS)
  - Koji-containerbuild
- Fedora’s Docker Layered Image Build Service
- Q&A

Containers

What are containers?

- Operating-system-level Virtualization
  - We (the greater Linux community) like to call them “containers”
- OK, so what is Operating-system-level Virtualization?
  - The multitenant isolation of multiple user space instances or namespaces.

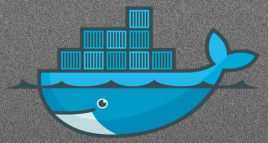


Containers are not new

- The concept of containers is not new
  - chroot was the original “container”, introduced in 1982
    - Unsophisticated in many ways, lacking the following:
      - COW
      - Quotas
      - I/O rate limiting
      - cpu/memory constraint
      - Network Isolation
  - Brief (not exhaustive) history of sophisticated UNIX-like container technology:
    - 2000 - FreeBSD jails
    - 2001 – Linux Vserver
    - 2004 – Solaris Zones
    - 2005 - OpenVZ
    - 2008 – LXC
      - This is where things start to get interesting

The Modern Linux Container is Born

- 2008 - IBM releases LinuX Containers (LXC)
  - Userspace tools to effectively wrap a chroot in kernel namespacing and cgroups
  - Provided sophisticated features the chroot lacked
- 2011 – systemd nspawn containers
  - run a command or OS in a light-weight namespace container. Like chroot, but virtualizes the file system hierarchy, process tree, various IPC subsystems, host and domain name.
- 2013 – DotCloud releases Docker (https://github.com/docker/docker)
  - Originally used LXC as the backend, introduces the Docker daemon, layered images, standard toolset for building images and a distribution method (docker registry). Later makes backend driver pluggable and replaces LXC with libcontainer as default.



Modern Linux Container

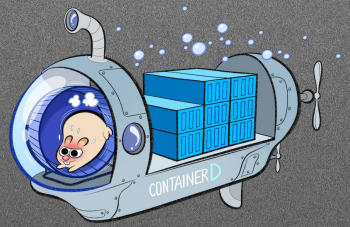
- 2014 – CoreOS releases rkt (https://github.com/coreos/rkt)
  - rkt is an implementation of App Container(appc) specification and App Container Image(ACI) specification, built on top of systemd-nspawn.
  - ACI and appc aimed to be a cross-container specification to be a common ground between container implementations.

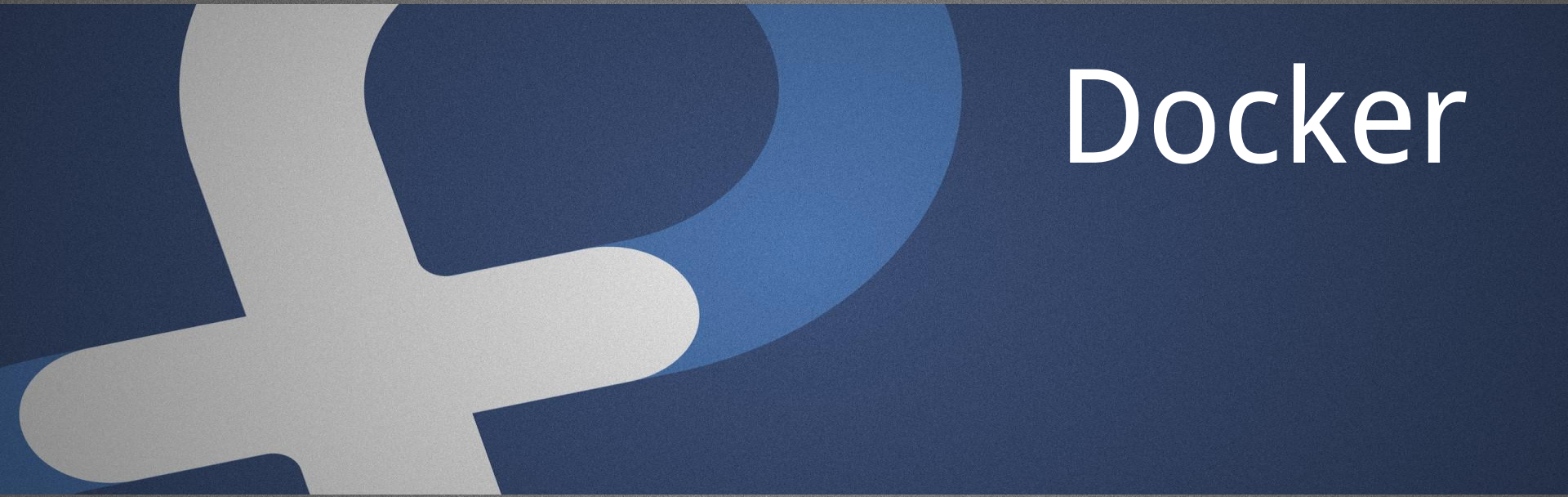
- 2015 – Open Container Project (http://opencontainers.org/)
  - “The Open Container Initiative is a lightweight, open governance structure, to be formed under the auspices of the Linux Foundation, for the express purpose of creating open industry standards around container formats and runtime.” - http://opencontainers.org/
  - Initiative Sponsors: Apcera, AT&T, AWS, Cisco, ClusterHQ, CoreOS, Datera, Docker, EMC, Fujitsu, Google, Goldman Sachs, HP, Huawei, IBM, Intel, Joyent, Kismatic, Kyup, the Linux Foundation, Mesosphere, Microsoft, Midokura, Nutanix, Oracle, Pivotal, Polyverse, Rancher, Red Hat, Resin.io, Suse, Sysdig, Twitter, Verizon, VMWare

Modern Linux Container

- 2015 – runC (http://runc.io/)
  - Stand-alone command line tool for spawning containers as per the OCP specification.
  - Containers are child processes of runC, no system daemon, can be embedded.
  - Shares technology lineage with Docker (libcontainer and others).
  - Compatible with Docker images.
  - Docker Engine v1.11+

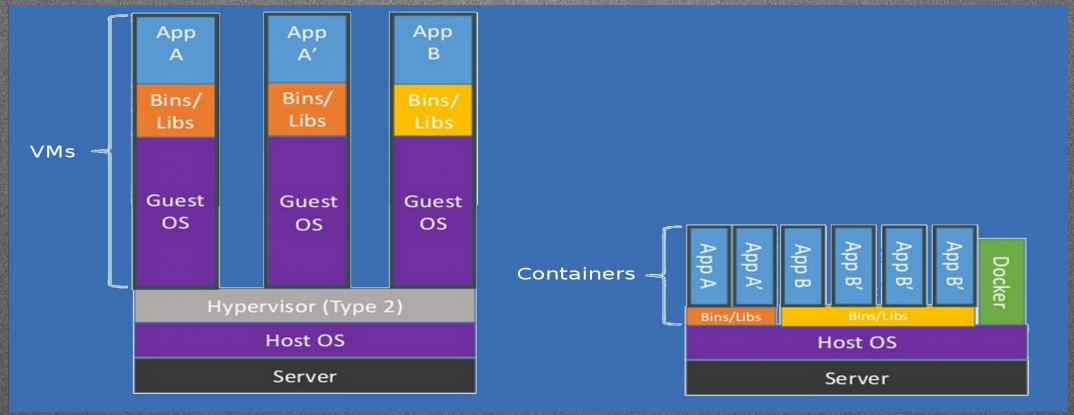
- 2016 – containerd (http://containerd.tools/)
  - Containerd is a daemon with an API and a command line client, to manage containers on one machine. It uses runC to run containers according to the OCI specification.
  - Docker Engine v1.11+



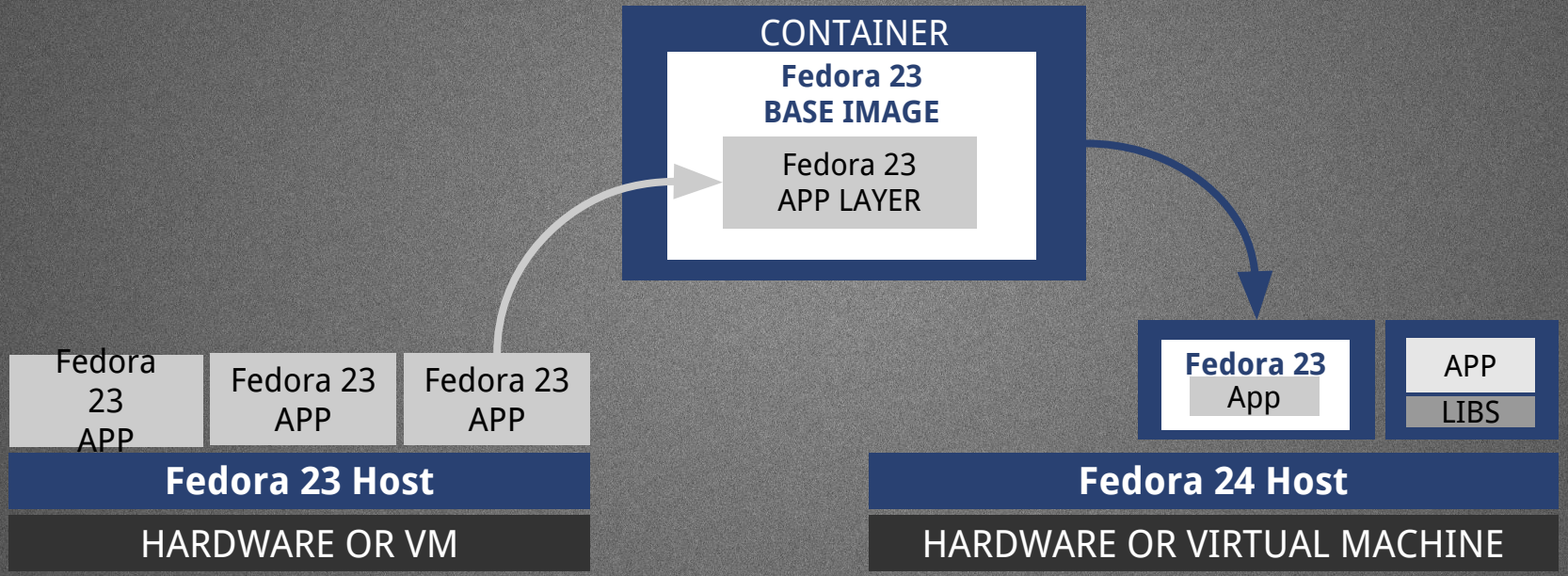

Docker

Docker

- Docker Engine (daemon) is the single point of entry, has language bindings for other clients and tooling. (Image verification)
- **Containers** are instances of **images**.
- Images are built in a standard way using Dockerfile
- SELinux support upstream in Docker.
- Pluggable backends for isolation mechanism, storage, networking, etc.



Docker Base vs Layered Images

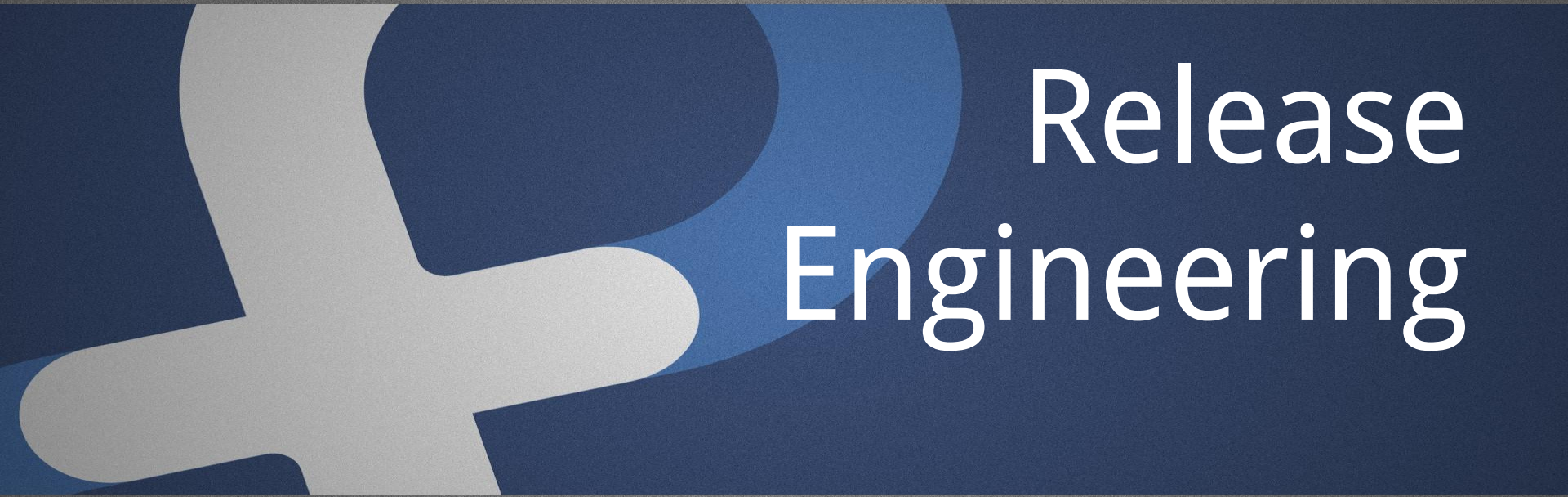


Dockerfile

```
FROM fedora
MAINTAINER http://fedoraproject.org/wiki/Cloud

RUN dnf -y update && dnf clean all
RUN dnf -y install httpd && dnf clean all
RUN echo "HTTPD" >> /var/www/html/index.html

EXPOSE 80

# Simple startup script
ADD run-httpd.sh /run-httpd.sh
RUN chmod -v +x /run-httpd.sh

CMD ["/run-httpd.sh"]
```

Docker Build

```
$ docker build -t fedora-httpd .
Sending build context to Docker daemon 24.06 kB
Step 1 : FROM docker.io/fedora
 ---> f9873d530588
Step 2 : MAINTAINER http://fedoraproject.org/wiki/Cloud
 ---> Running in d7c01855128e
 ---> 819fb0ed13b0
Removing intermediate container d7c01855128e
Step 3 : LABEL RUN 'docker run -d -p 80:80 $IMAGE'
 ---> Running in 4288ff446166
 ---> 5f2b85cdbd73
Removing intermediate container 4288ff446166
Step 4 : RUN dnf -y update && dnf -y install httpd && dnf clean all
 ---> Running in df63942c3979
… OUTPUT OMITTED FOR BREVITY …
Successfully built 63bc543a1868
```


Release
Engineering

Release Engineering

- What is Release Engineering?
  - Making a software production pipeline that is Reproducible, Auditable, Definable, and Deliverable
    - It should also be able to be automated
- Definition (or the closest there really is)

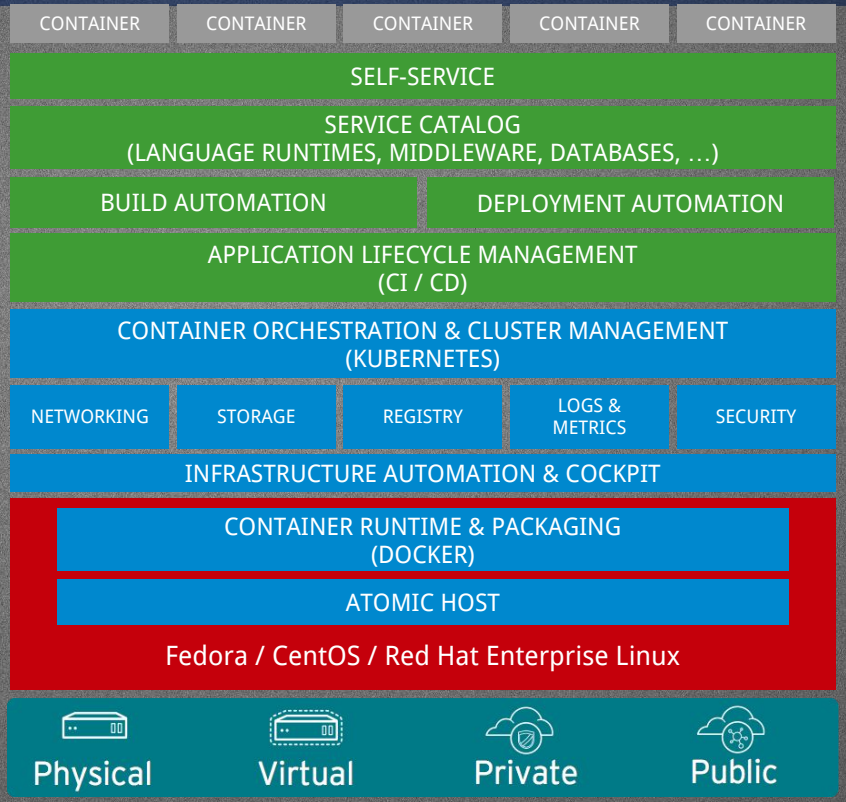
> “Release engineering is the difference between manufacturing software in small teams or startups and manufacturing software in an industrial way that is repeatable, gives predictable results, and scales well. These industrial style practices not only contribute to the growth of a company but also are key factors in enabling growth.”
>
> - Boris Debic of Google Inc


OpenShift

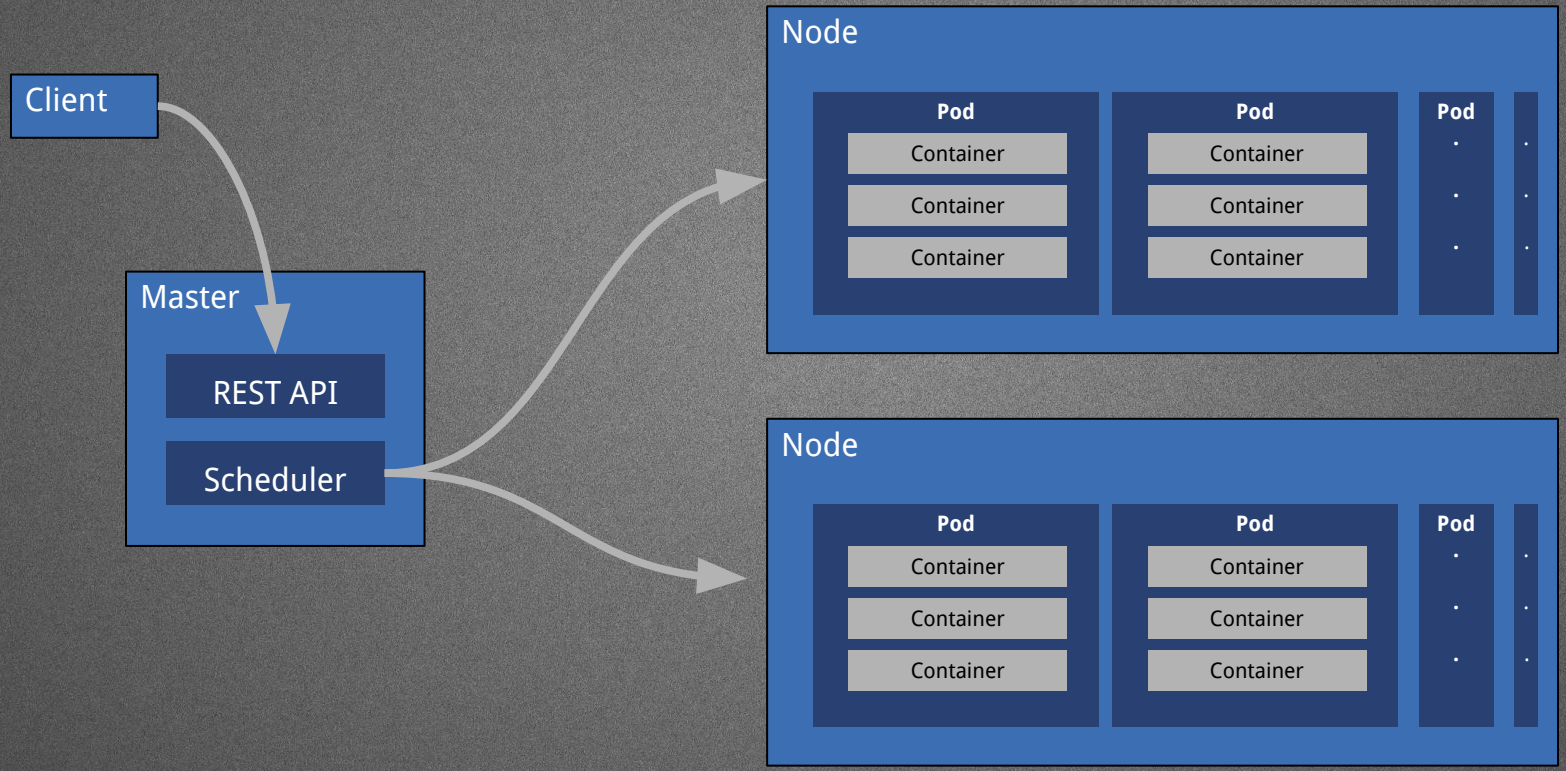
OpenShift

OPENSHIFT origin

OPENSHIFT by Red Hat

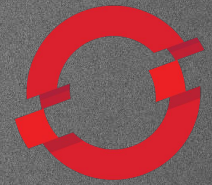


OpenShift/Kubernetes Overview



OpenShift

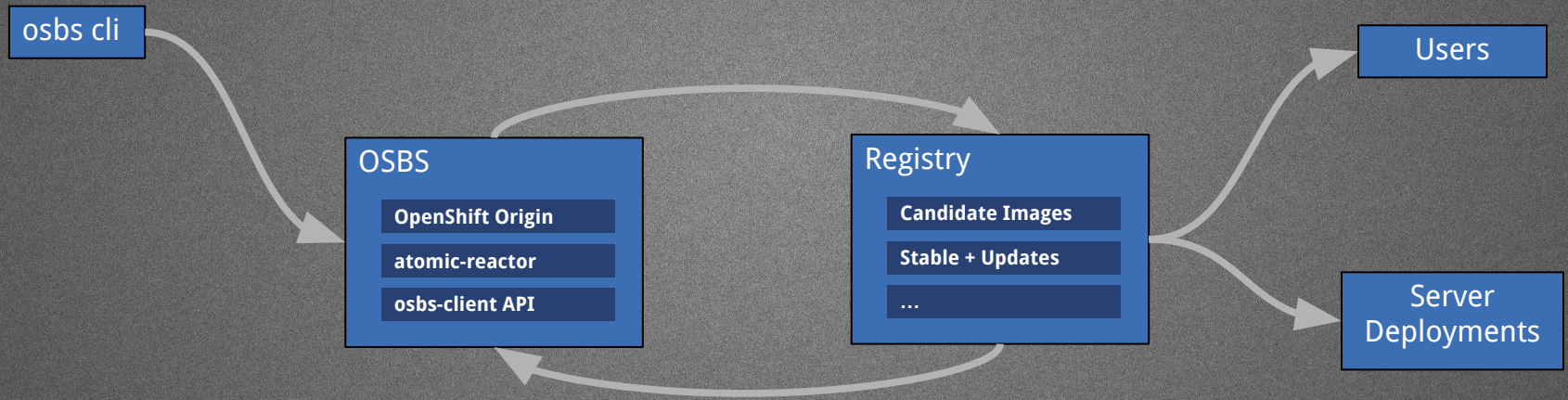
- OpenShift
  - Container Platform built on top of Kubernetes
  - Advanced Features
    - Build Pipelines
    - Image Streams
    - Application Lifecycle Management
    - CI/CD Integrations
    - Binary Deployment
    - Triggers (Event, Change, Image, Web, etc)
  - REST API, Command line interface, IDE Integrations
  - Web UI and Admin dashboard



OPENSHIFT

Docker Layered Image Build Service

Build System



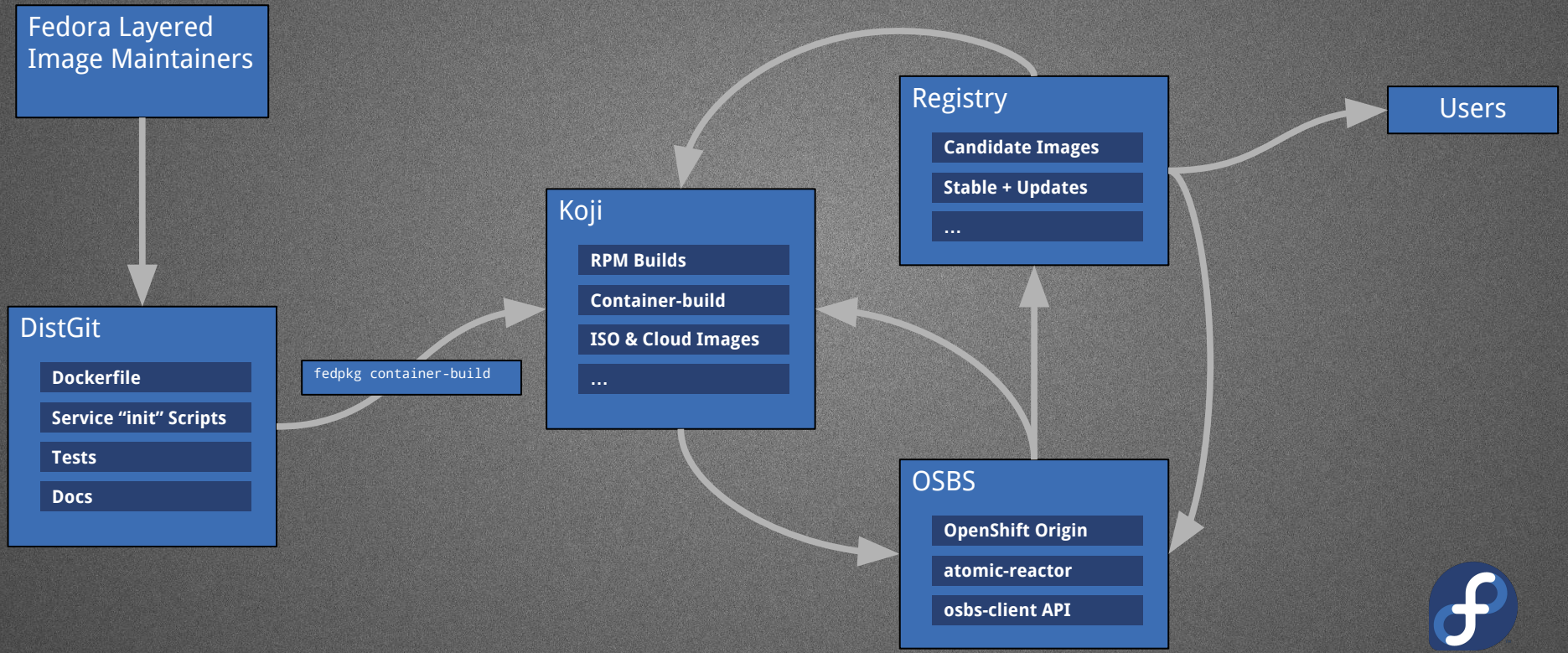
OSBS

- OpenShift Build Service
  - Takes advantage of OpenShift's built in Build primitive with a "Custom Strategy" and BuildConfig
    - This defines what can be the inputs to a build
  - Relies on OpenShift for scheduling of build tasks throughout the cluster
  - Presents this defined component to developers/builders as CLI and Python API
  - osbs enforces that the inputs come from auditable sources.
    - Git repo for source Dockerfile, git commits and builds centrally logged
  - BuildRoot - limited docker runtime
    - Firewall constrained docker bridge interface
    - Unprivileged container runtime with SELinux Enforcing
    - Inputs are sanitized before reaching to build phase
      - Unknown or unvetted sources are disallowed by the system
  - Uses OpenShift ImageStreams as input sources to BuildRoot
  - Utilizes OpenShift Triggers to spawn rebuild actions based on parent image changes
    - How often are your images rebuilt?

OSBS - Continued

- atomic-reactor
  - Single-pass Docker build tool used inside constrained buildroot in OSBS
  - Automates tasks via plugins, such as:
    - pushing images to a registry when successfully built
    - injecting yum/dnf repositories inside Dockerfile (change source of your packages for input sanitization/gating)
    - change base image (FROM) in your Dockerfile to
    - match that of the registry available inside the isolated buildroot, run simple
    - tests after image is built
- Gating of updates
  - Automated tests can be tied to the output of OSBS
  - RelEng is able to then "promote" images to a "production" or "stable" registry/tag/repository

Fedora's Implementation



Fedora's Implementation

- DistGit ("Distro Git")
  - Each Branch = Fedora Release
  - master branch is Devel (codename "Rawhide")
- fedpkg
  - Fedora Package Maintainer helper tool
  - Manages distgit branches
  - Initiate builds (local and remote, mock integration)
  - Much more …
- Koji
  - Fedora's authoritative build system
  - Everything for Fedora is built here or it's build is integrated here
    - Live USB images, DVD ISOs, IaaS Cloud Images, RPMs, Docker
    - This defines what can be the inputs to a build
- Koji-containerbuild
  - Plugin to orchestrate builds between Koji and OSBS
- Registry
  - Upload/download destination, point of distribution

Questions?

CONTACT:
maxamillion@fedoraproject.org
@TheMaxamillion

References

- https://en.wikipedia.org/wiki/Operating-system-level_virtualization
- https://coreos.com/blog/rocket
- https://coreos.com/blog/appc-gains-new-support
- https://www.docker.com
- https://github.com/docker/distribution
- http://www.redhat.com/en/insights/containers
- http://www.projectatomic.io
- http://www.openshift.org
- https://www.openshift.com
- http://www.redhat.com/en/about/blog/red-hat-and-google-collaborate-kubernetes-manage-docker-containers-scale
- http://rhelblog.redhat.com/2014/04/15/rhel-7-rc-and-atomic-host
- http://opencontainers.org
- http://runc.io
- http://queue.acm.org/detail.cfm?id=2884038
- http://containerd.tools
- https://drive.google.com/file/d/0B_JI94nModqdSFVseUotQVB1Rnc/view?usp=sharing
- http://valleyproofs.debic.net/2009/03/behind-scenes-production-pushes.html
- https://github.com/release-engineering/koji-containerbuild
- https://github.com/projectatomic/atomic-reactor
- https://github.com/projectatomic/osbs-client
- https://pagure.io/koji
- https://fedorahosted.org/koji/wiki