



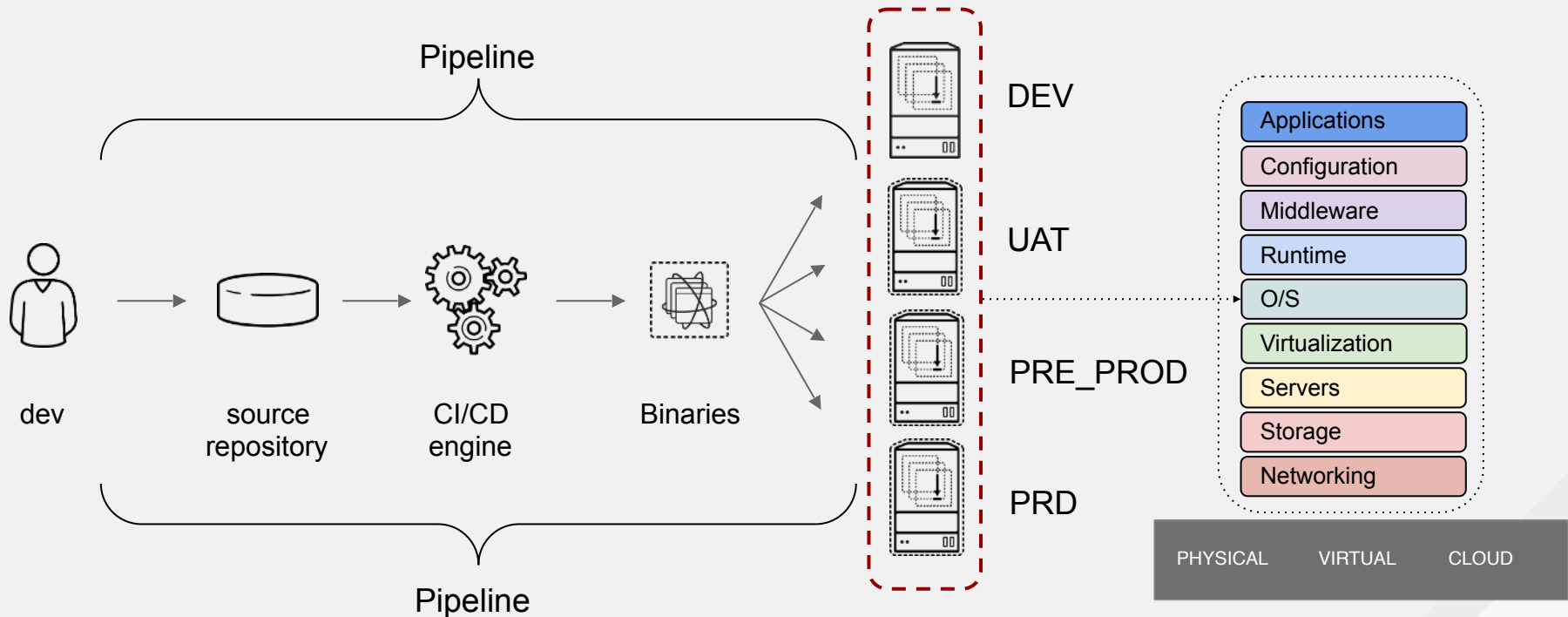
# Red Hat OpenShift

## Overview

Waeil Eldoamiry  
Solutions Architect



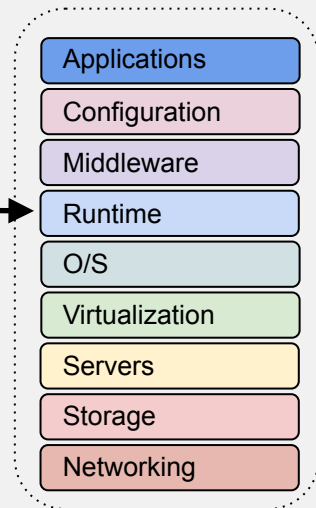
# TRADITIONAL APPLICATION DELIVERY



I know how  
to code



# ZooooM on Runtime



PHYSICAL

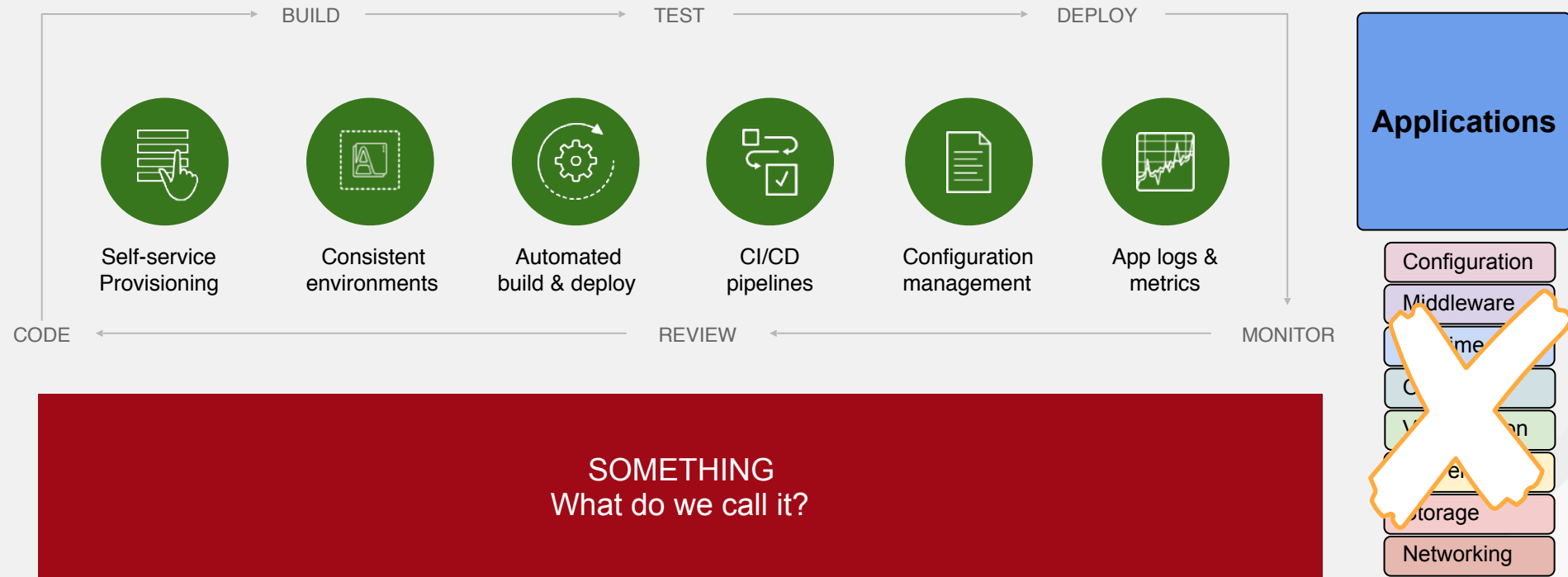
VIRTUAL

CLOUD

## Challenges

- Coding
- Building Artifacts
- Testing (Unit, Integration, Load)
- Quality Review
- Deployment
- Configuration Management
- Traceability
- Observability
- Infrastructure
- High Availability
- Resiliency
- Security
- Scalability
- Logging
- Metrics
- RBAC

# IS THERE A MAGIC SOLUTION



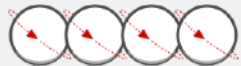
\* coming soon

GENERAL DISTRIBUTION

# KEY TECHNOLOGY TRENDS

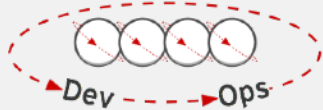
## Development Process

Waterfall



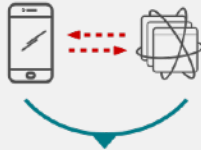
Agile

**DevOps**



## Application Architecture

Monolithic



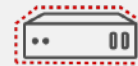
N-Tier

**Microservices**



## Deployment & Packaging

Physical Servers



Virtual Servers

**Containers**



## Application Infrastructure

Datacenter



Hosted



**Cloud**



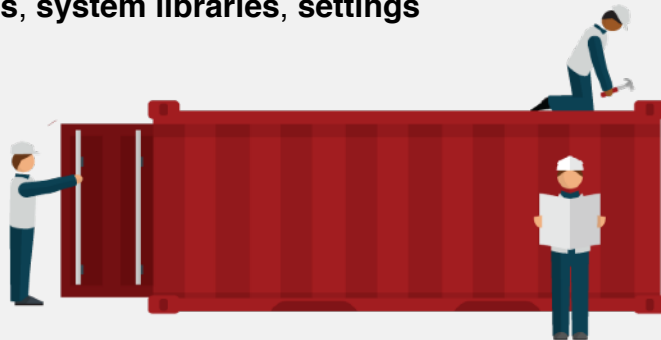
# CONTAINERS

## What is “Container”?

- Easy to deploy and portable across host systems
- Isolates applications on a host operating systems

Adopting a container strategy will allow **applications** to be easily **shared** and **deployed**.

A container image is a **lightweight, stand-alone, executable** package of a piece of software that includes everything needed to run it: **code, runtime, system tools, system libraries, settings**



Package application & dependencies  
Simplify deployment  
Speed delivery



# MICROSERVICES

## CHARACTERISTICS

1

Componentization

Self  
contained

Independently  
Deployable

Independently  
Upgradable

2

Organized around business capabilities

3

Decentralized data management

4

Smart endpoints and dumb pipes

5

Location Transparency

6

Automation

7

API Focused

8

Decentralized Governance

9

Design for failure

# DEVOPS

Everything as code

Application monitoring

Automate everything

Rapid feedback

Continuous Integration/Delivery

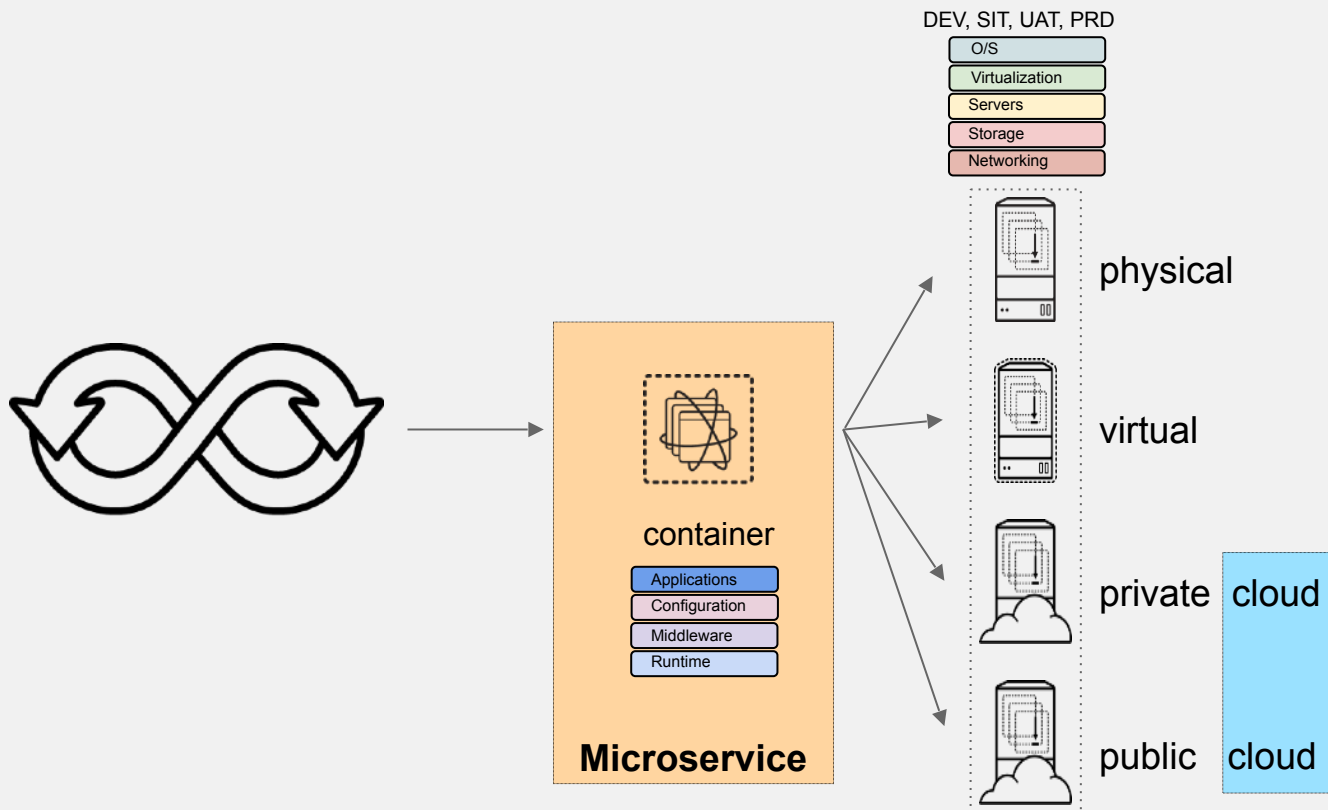
Rebuild vs. Repair

Application is always “releasable”

Delivery pipeline



# MODERN APPLICATION DELIVERY



# YOUR JOURNEY

DevOps



Self-Service,  
On-Demand,  
Elastic  
Infrastructure



Automation



CI & CD  
Deployment  
Pipeline



Advanced  
Deployment  
Techniques

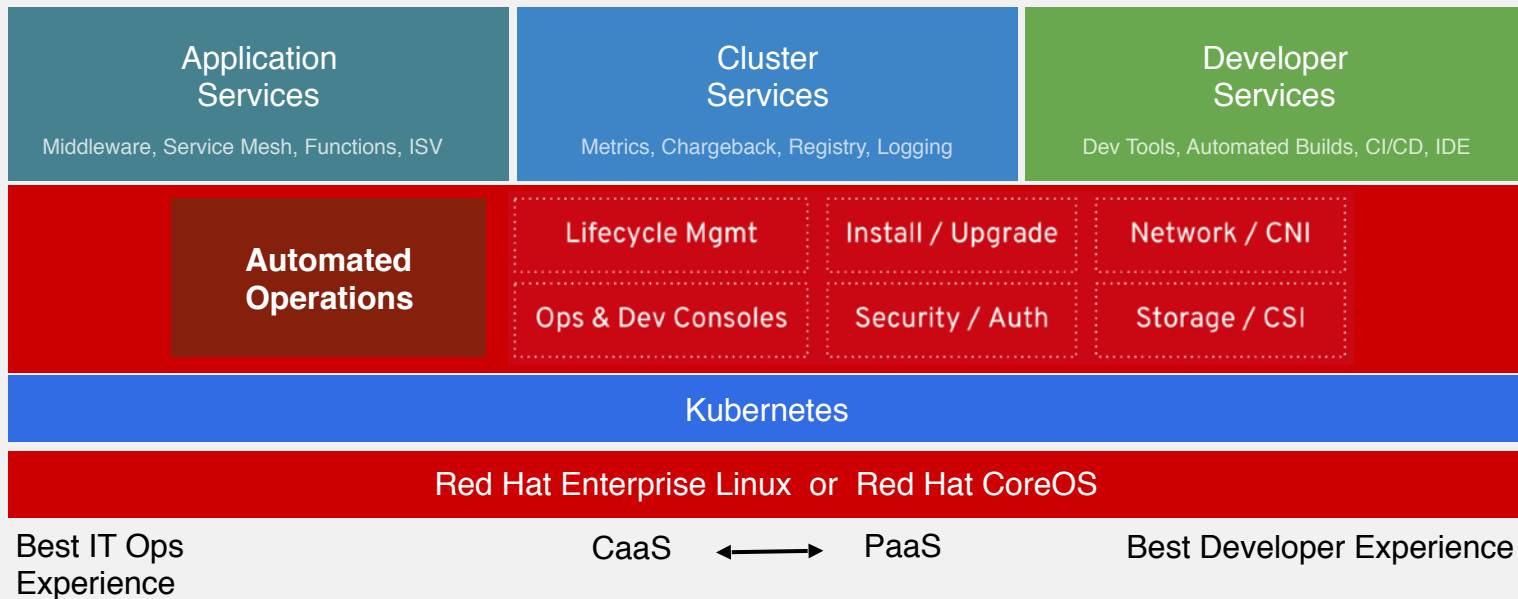


Advanced  
**Microservices**

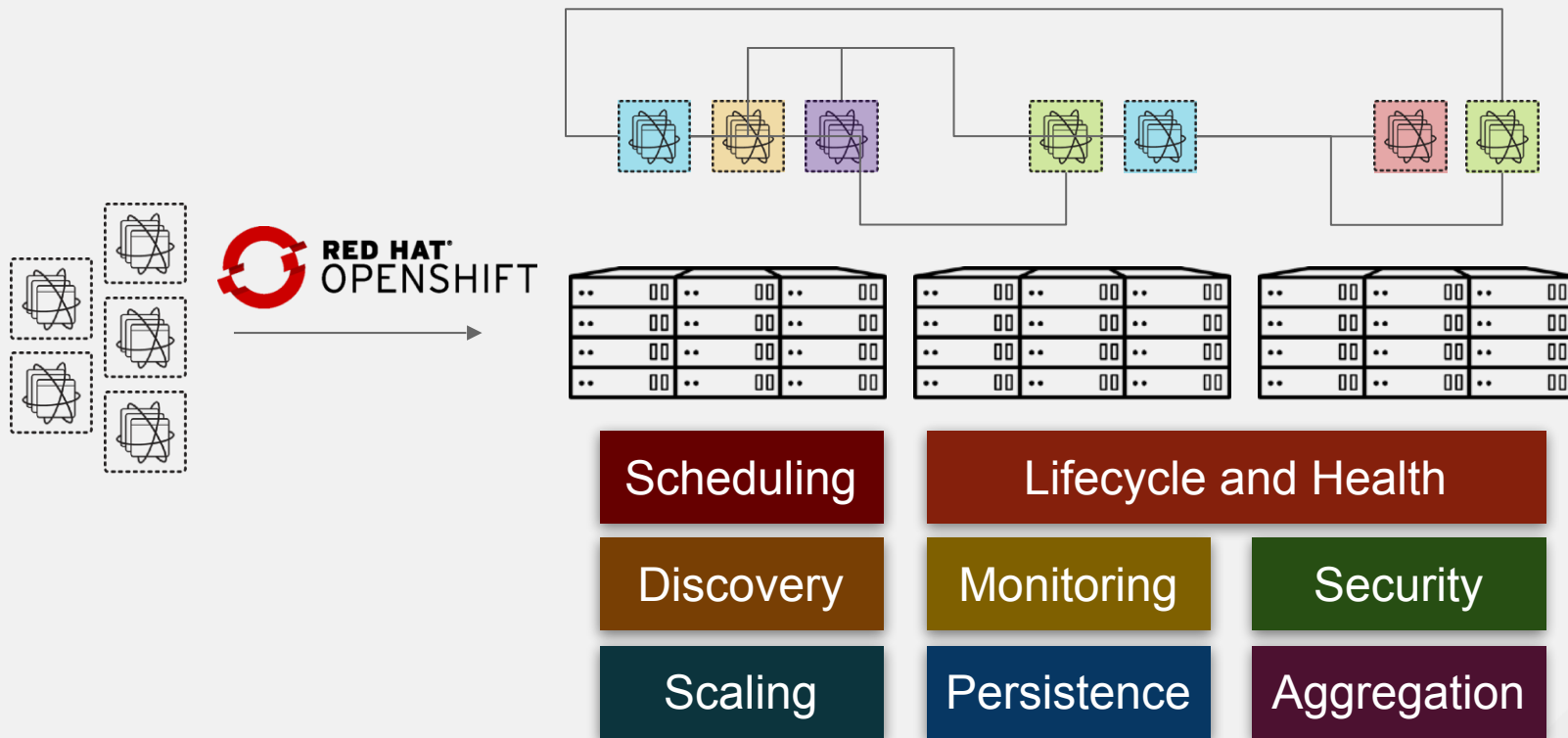
## GET READY FOR THE BIG FIGHT!

# OPENSIFT

# OPENSIFT CONTAINER PLATFORM



# CONTAINER ORCHESTRATION

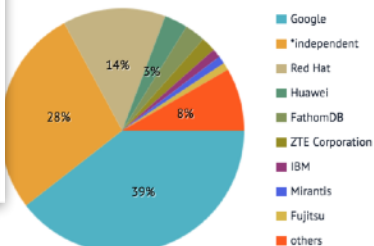


# NOT ENOUGH, THERE IS MORE!

Multi-tenancy	Teams and Collaboration
Routing & Load Balancing	Quota Management
CI/CD Pipelines	Image Build Automation
Role-based Authorization	Container Isolation
Capacity Management	Vulnerability Scanning
Infrastructure Visibility	Chargeback

# OPENSIFT CONTAINER PLATFORM

#	Company	Commits
1	Google	41425
2	*Independent	28979
3	Red Hat	14513
4	Huawei	3238
5	FathomDB	2865
6	ZTE Corporation	2035
7	IBM	1223
8	Mirantis	1082
9	Fujitsu	1034
9	CoreOS	964



FEATURE	KUBERNETES	OPENSIFT ORIGIN	OPENSIFT CONTAINER PLATFORM
Multi-host container scheduling	✓	✓	✓
Self-service provisioning	✓	✓	✓
Service-discovery	✓	✓	✓
Persistent storage	✓	✓	✓
Multi-tenancy	⊖	✓	✓
Collaboration	⊖	✓	✓
Networking	⊖	✓	✓
Image registry	⊖	✓	✓
Monitoring	⊖	✓	✓
Log aggregation	⊖	✓	✓
CI/CD and DevOps	⊖	✓	✓
Application services (databases, runtimes, ...)	⊖	⊖	✓
Middleware services	⊖	⊖	✓
Built-in operational management	⊖	⊖	✓
Enterprise-grade operating system	⊖	⊖	✓
100% Open Source	✓	✓	✓
Community support	✓	✓	✓
Enterprise 24/7 Support	⊖	⊖	✓
Security response team	⊖	⊖	✓
Stable Lifecycle (7 years)	⊖	⊖	✓

APPLICATION LIFECYCLE MANAGEMENT



CONTAINER ORCHESTRATION AND MANAGEMENT  
(KUBERNETES)

ENTERPRISE CONTAINER HOST



Laptop



Datacenter



OpenStack



Amazon Web Services



Microsoft Azure



Google Cloud

ANY  
CONTAINER



RED HAT  
OPENSIFT

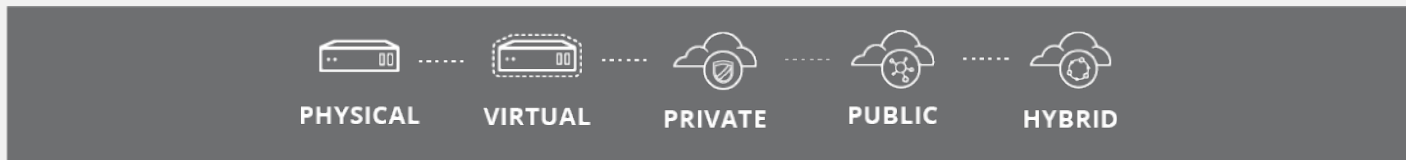
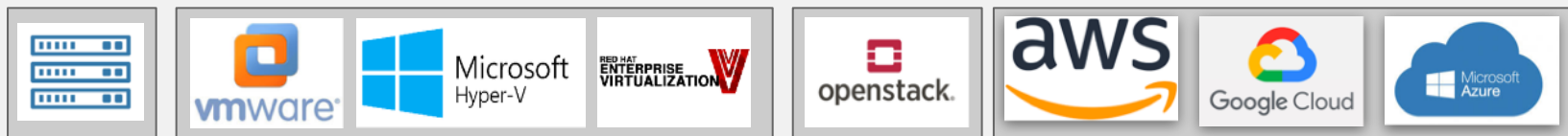
ANY  
INFRASTRUCTURE

# HYBRID CLOUD DEPLOYMENT

## ENTERPRISE WORKLOAD



THE COMMON FABRIC





# Technology Focus

## Developer Productivity

- Cross Technology Consistent Experience
- Service Catalog (Application Services)
- Self-Service Portal
- Automated Build and Deployment (CI/CD)
- Configuration Management
- Service Resiliency
- Team Organization (RBAC)
- Application Frameworks
- Service Mesh

## Operational Efficiency

- Infrastructure transparency
- Built-In HA
- Advanced Deployment Strategies
- Self-Healing
- Auto Scaling
- Application Probs (Readiness and Liveness)
- Central Logging
- Metrics and capacity Management
- Automated Day 2 Ops
- Cloud Adoption

# TRUE POLYGLOT PLATFORM

## LANGUAGES

Java	NodeJS	Python	PHP	Perl	Ruby	.NET Core	Third-party Language Runtimes
------	--------	--------	-----	------	------	-----------	-------------------------------

## DATABASES

MySQL	PostgreSQL	MongoDB	Redis	...and virtually any docker image out there!			Third-party Databases
-------	------------	---------	-------	--	--	--	-----------------------

## WEB SERVERS

Apache HTTP Server	nginx	Varnish	Phusion Passenger	Tomcat			Third-party App Runtimes
--------------------	-------	---------	-------------------	--------	--	--	--------------------------

## MIDDLEWARE

Spring Boot	Wildfly Swarm	Vert.x	JBoss Web Server	JBoss EAP	JBoss A-MQ	JBoss Fuse	Third-party Middleware
3SCALE API mgmt	JBoss BRMS	JBoss BPMS	JBoss Data Virt	JBoss Data Grid	RH Mobile	RH SSO	Third-party Middleware

**CrunchyData**

**GitLab**

**Iron.io**

**Couchbase**

**Sonatype**

**EnterpriseDB**

**NuoDB**

**Fujitsu**

and many more

# DEVELOPER SELF-SERVICE

## ONBOARDING EXPERIENCE

- Entry point for a developer to access all services available to them
- Merges all capabilities from Operators, Service Catalog, Brokers, and S2I

The image shows two overlapping screenshots from the Red Hat OpenShift Developer Catalog. The left screenshot displays the 'Developer Catalog' interface with a sidebar menu containing 'Home', 'Catalog', 'Developer Catalog', 'Installed Operators', 'Provisioned Services', 'Operator Hub', 'Operator Management', 'Broker Management', 'Workloads', 'Networking', and 'Storage'. The main content area shows the 'Developer Catalog' for the 'kube-system' project, listing various items like '.NET Core' and 'Apache HTTPD'. The right screenshot shows the 'Create Source-to-Image Application' form, which includes fields for 'Namespace' (set to 'kube-system'), 'Version' (set to 'nginx:1.12'), and 'Name'. It also features a 'Git Repository' field and a 'Try Sample' button. The form provides detailed instructions for building and deploying the application, including a list of resources that will be created: a build config, an image stream, a deployment config, a service, and an optional route.

**Developer Catalog Interface:**

- Project: kube-system
- Developer Catalog
- Filter by keyword...
- TYPE: Service Class (0), Source-to-Image (9), Installed Operators (8)
- Items: .NET Core, Apache HTTPD

**Create Source-to-Image Application Form:**

- Namespace: kube-system
- Version: nginx:1.12
- Name:
- Git Repository:
- Try Sample
- Create route (checkbox)
- Create button

**Build and Deployment Details:**

- Builder: NGINX
- Build: Nginx HTTP server and a reverse proxy 1.12
- Build Description: Build and serve static content via Nginx HTTP Server and a reverse proxy on CentOS 7.
- Sample repository: <https://github.com/sclorg/nginx-ex.git>
- Resources to be created:
  - A build config to build source from a Git repository.
  - An image stream to track built images.
  - A deployment config to rollout new revisions when the image changes.
  - A service to expose your workload inside the cluster.
  - An optional route to expose your workload outside the cluster.

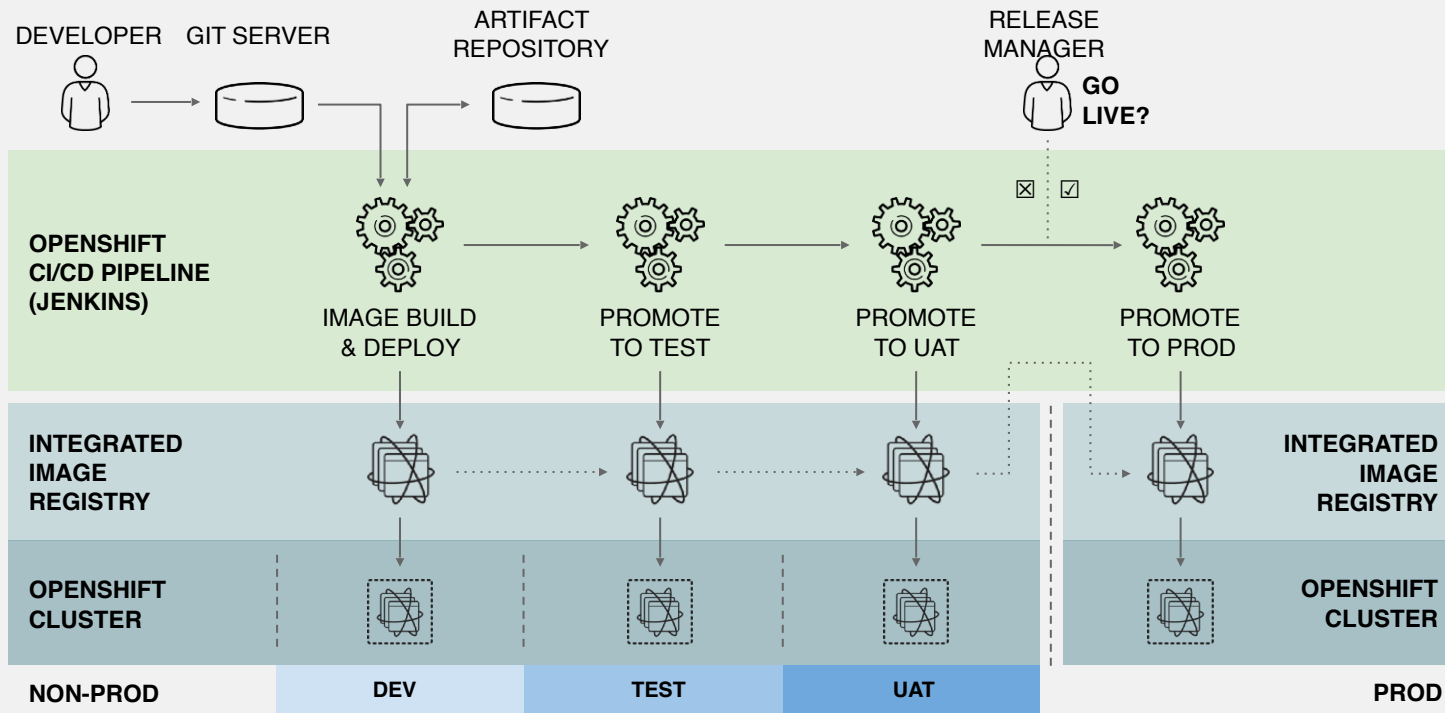
\* coming soon

# CROSS TECHNOLOGY DEVELOPMENT

## CONSISTENT EXPERIENCE

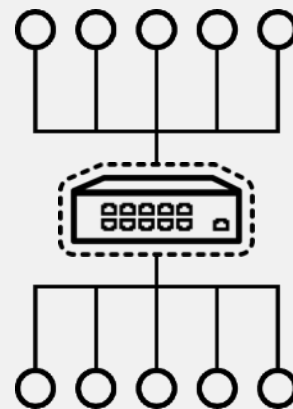
- Consistent provisioning process (**Service Catalog**)
- Consistent Build and Deployment process (**S2I**)
- Dependencies hassle free application delivery (**Container Image**)
- Consistent Configuration Management (**IS, DC, ConfigMap, Secret**)
- Cross technology services resiliency (**self-healing, Autoscaling**)
- Cross technology health check (**Readiness and Liveness Probs**)
- Cross technology deployment strategies (**Rolling, Canary, ..etc**)
- Cross technology Microservices platform (**Istio**)
- Cross technology Application Logs (**EFK**)
- Cross technology Web Based IDE (**Code Ready Workspaces**)
- Commonly used runtimes and Frameworks (**Spring Boot, Thorntail, Wildfly Swarm, Vert.x, Open Liberty, ...etc**)

# CONTINUOUS DELIVERY PIPELINE



# OPENSIFT NETWORKING

- Software Defined Networking (SDN) for a unified cluster network to enable pod-to-pod communication
- Built-in internal DNS to reach services by name
- OpenShift follows the Kubernetes Container Networking Interface (CNI) plug-in model



# OPENSIFT SDN

## FLAT NETWORK (Default)

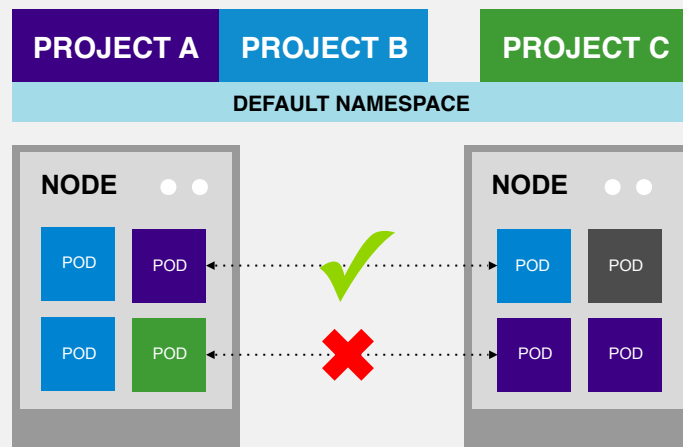
- All pods can communicate with each other across projects

## MULTI-TENANT NETWORK

- Project-level network isolation
- Multicast support
- Egress network policies

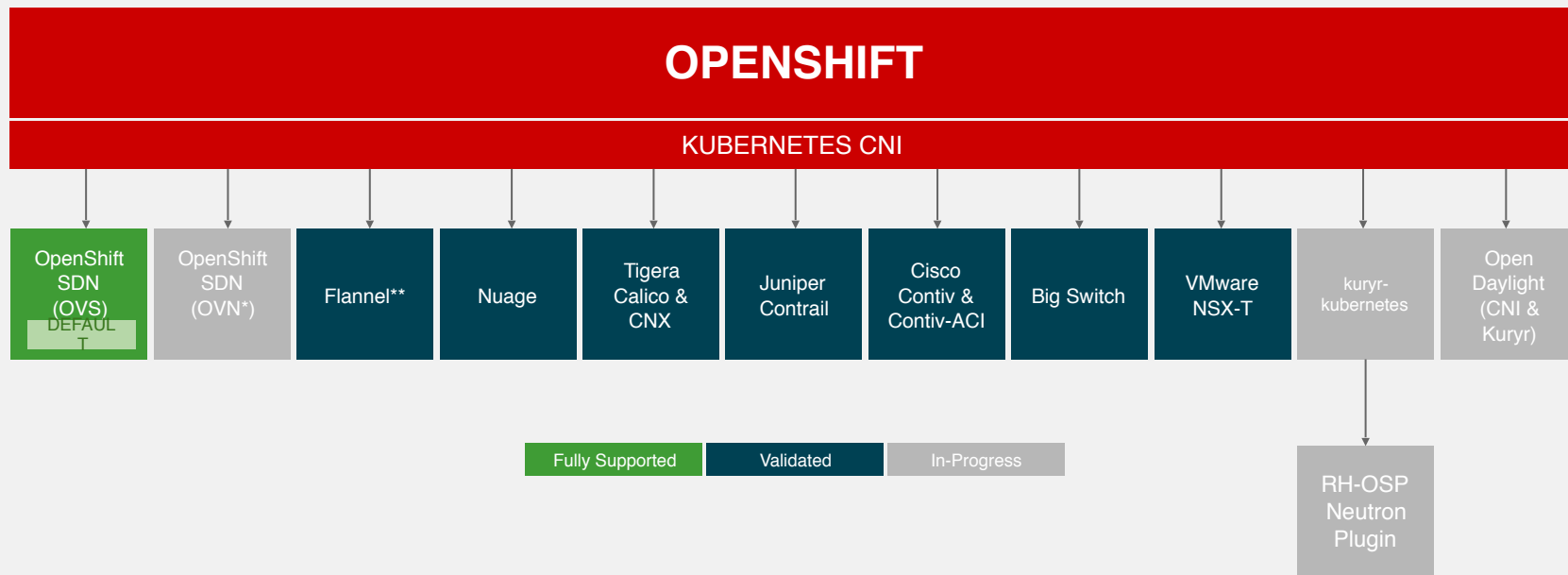
## NETWORK POLICY

- Granular policy-based isolation



Multi-Tenant Network

# OPENSHIFT NETWORK PLUGINS



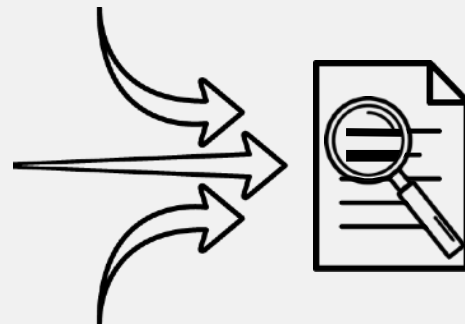
\* Coming as default in OCP 4.1

\*\* Flannel is minimally verified and is supported only and exactly as deployed in the OpenShift on OpenStack reference architecture

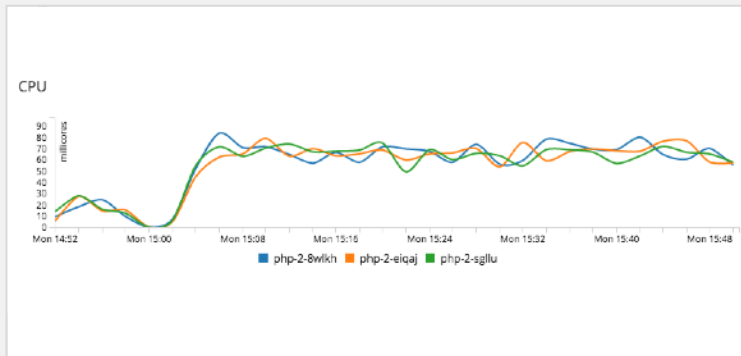
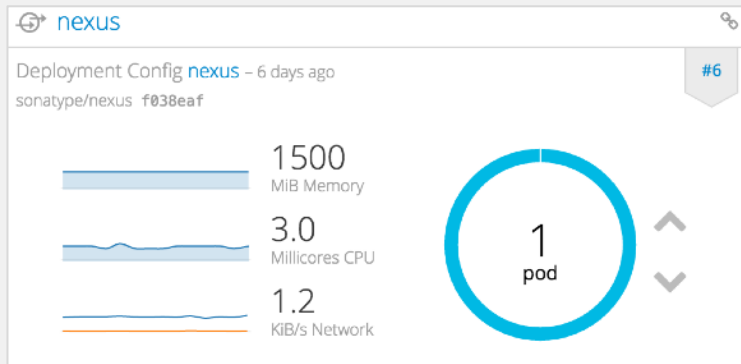


# CENTRAL LOG MANAGEMENT WITH EFK

- EFK stack to aggregate logs for hosts and applications
  - **Elasticsearch:** a search and analytics engine to store logs
  - **Fluentd:** gathers logs and sends to Elasticsearch.
  - **Kibana:** A web UI for Elasticsearch.
- Access control
  - Cluster administrators can view all logs
  - Users can only view logs for their projects
- Ability to send logs elsewhere
  - External elasticsearch, Splunk, etc



# CONTAINER METRICS

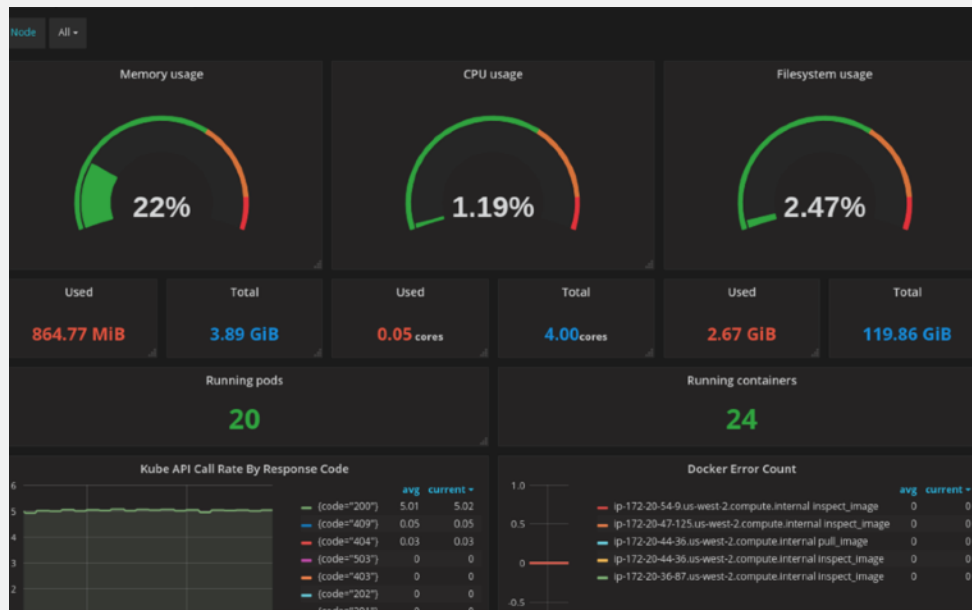


HAWKULAR

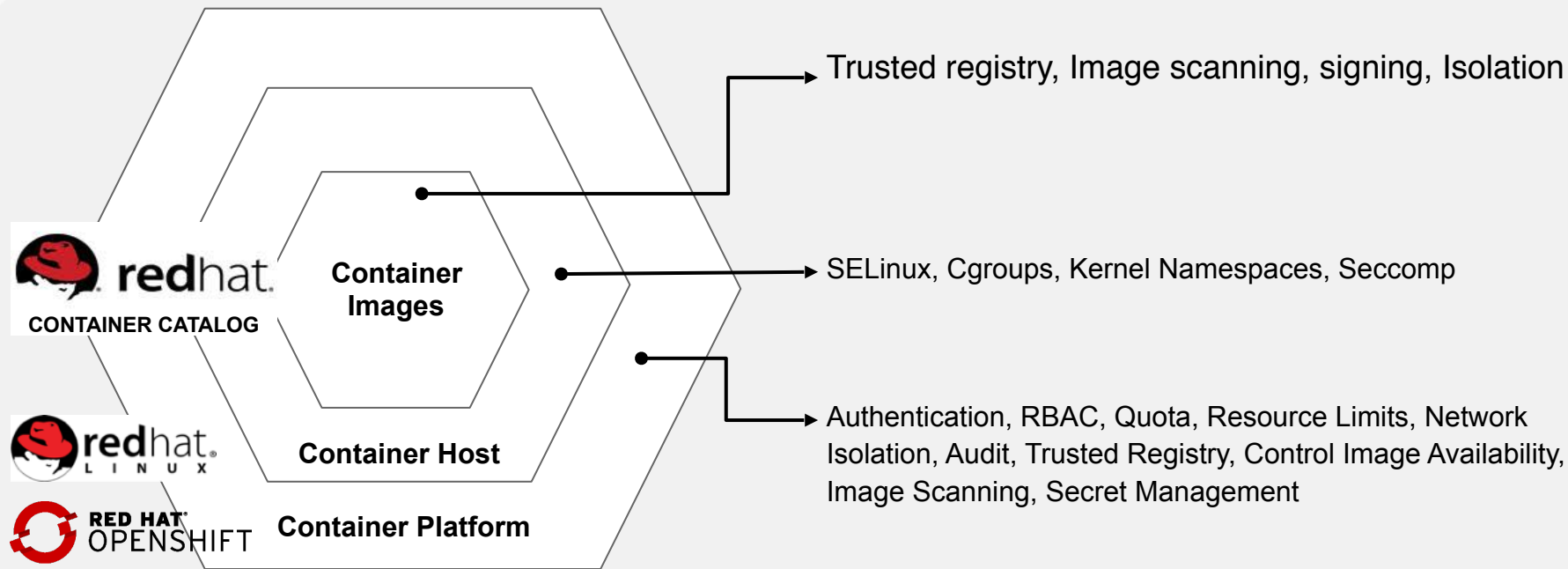
HEAPSTER

PROMETHEUS

Grafana



# SECURITY



**PCI DSS Product applicability Guide** to help customers understand how the Payment Card Industry Data Security Standard (PCI DSS) apply to the Red Hat OpenShift Container Platform.

<https://www.redhat.com/en/resources/openshift-pci-product-applicability-guide-datasheet>

# PCI DSS Product applicability Guide

## Greater security and compliance

- **PCI DSS product applicability guide** to help customers understand how the Payment Card Industry Data Security Standard (PCI DSS) apply to the Red Hat OpenShift Container Platform
- Red Hat engaged [Coalfire Systems, Inc.](#), a respected Payment Card Industry Qualified Security Assessor (QSA) company, to conduct an independent technical assessment of OpenShift Container Platform running on RHEL and/or Atomic Host. The applicability guide examines the PCI DSS through the eye of a QSA and identifies where the various requirements apply, or do not apply, to the overall solution. Overall, Red Hat and Coalfire came to the conclusion that OpenShift could be configured and deployed in a way that would satisfy the PCI DSS, and we produced a Product Applicability Guide (PAG) to help you understand these opinions.
- Find the guide on <https://www.redhat.com/en/resources/openshift-pci-product-applicability-guide-datasheet>

# PERSISTENT STORAGE

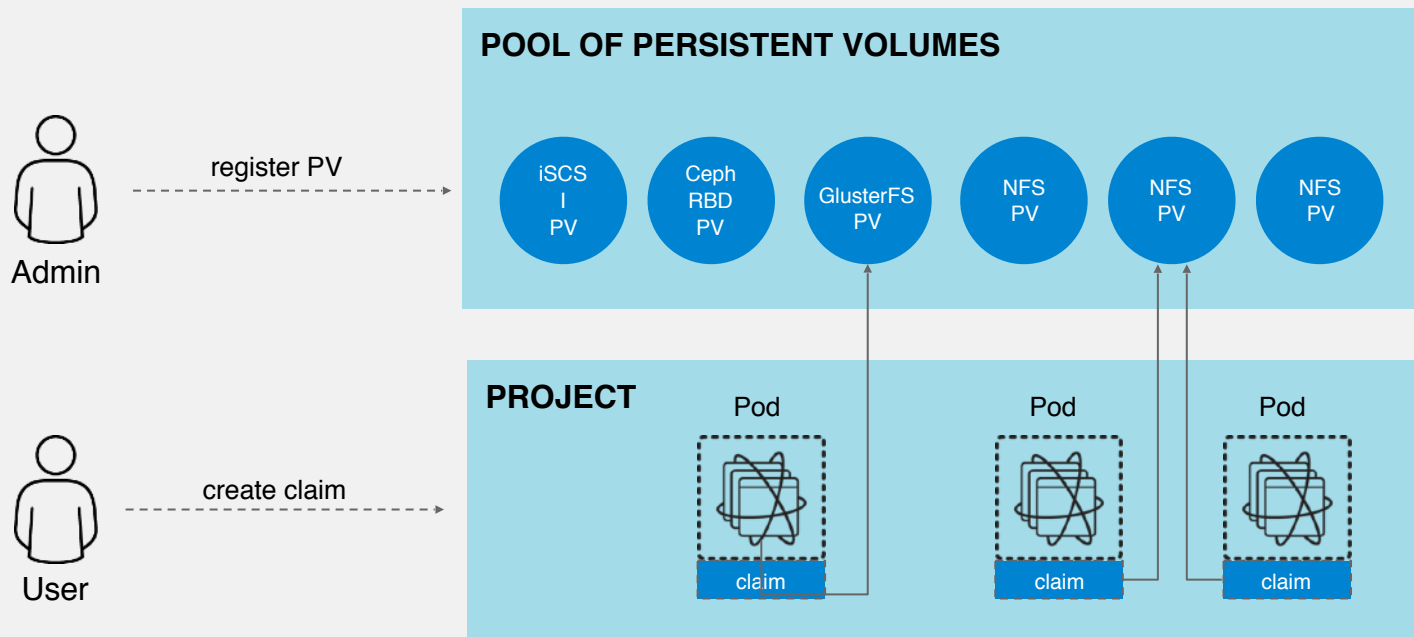
- Persistent Volume (PV) is tied to a piece of network storage
- Provisioned by an administrator (static or dynamically)
- Allows admins to describe storage and users to request storage
- Assigned to pods based on the requested size, access mode, labels and type

NFS	OpenStack Cinder	iSCSI	Azure Disk	AWS EBS	FlexVolume
GlusterFS	Ceph RBD	Fiber Channel	Azure File	GCE Persistent Disk	VMWare vSphere VMDK
		NetApp Trident*	Container Storage Interface (CSI)**		

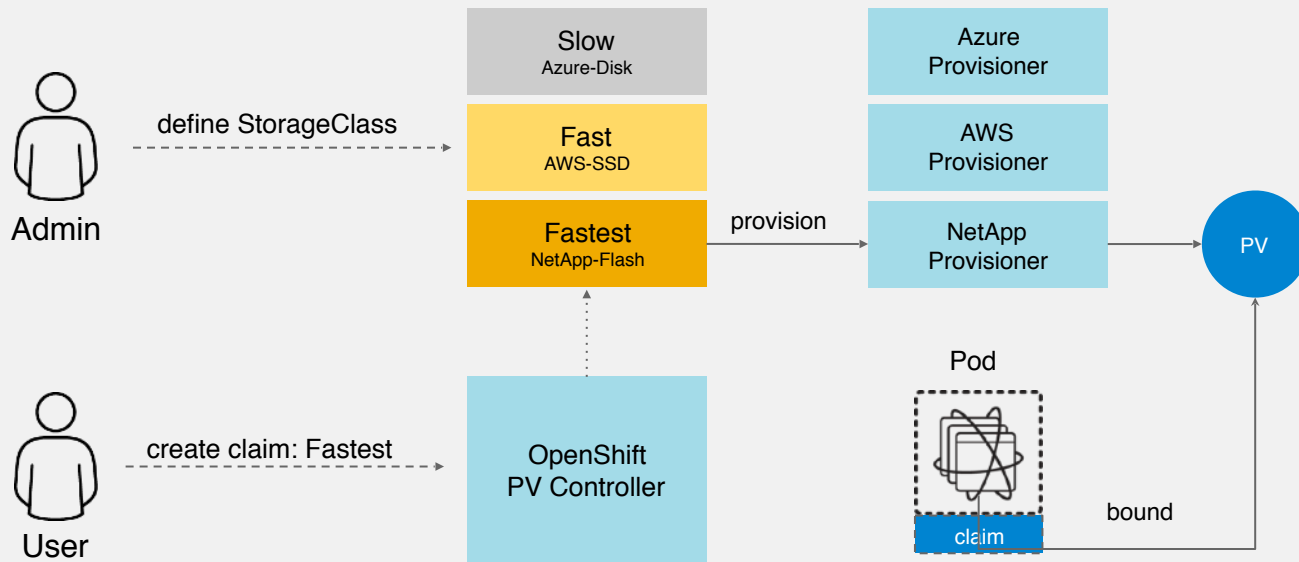
\* Shipped and supported by NetApp via TSANet

\*\* Tech Preview

# PERSISTENT STORAGE

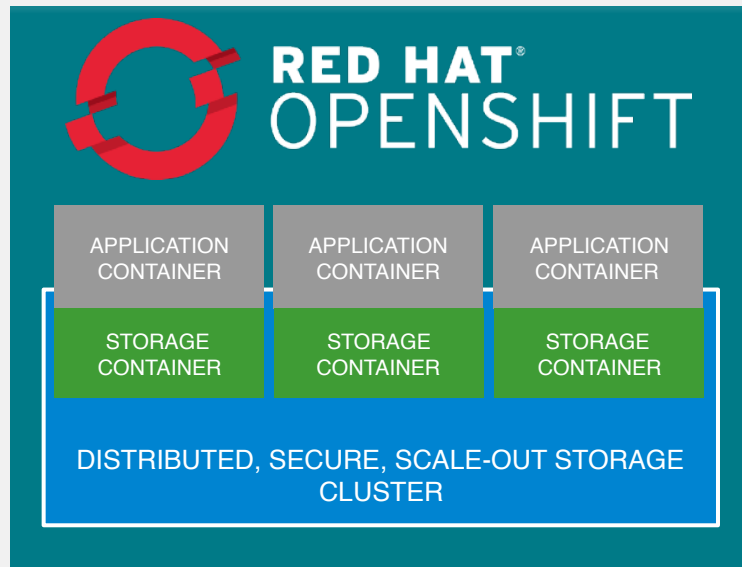


# DYNAMIC VOLUME PROVISIONING



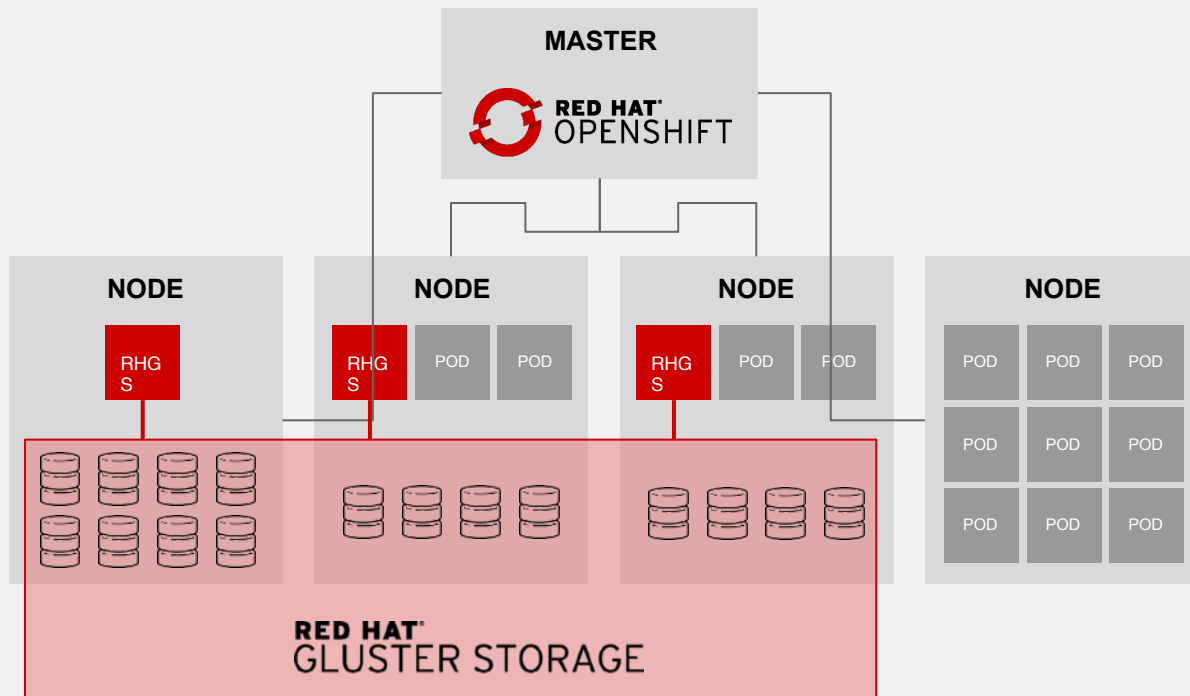
# OPENSIFT CONTAINER STORAGE

- Containerized Red Hat Gluster Storage
- Native integration with OpenShift
- Unified Orchestration using Kubernetes for applications and storage
- Greater control & ease of use for developers
- Lower TCO through convergence
- Single vendor Support

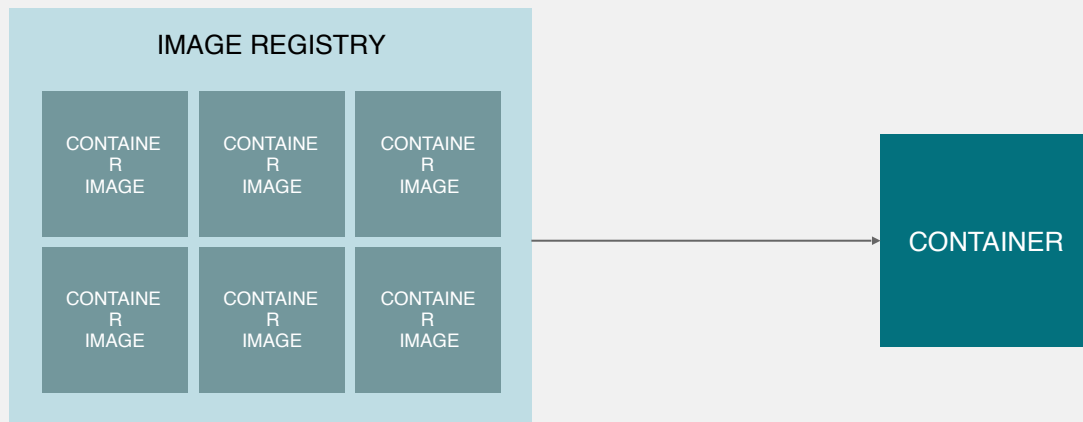




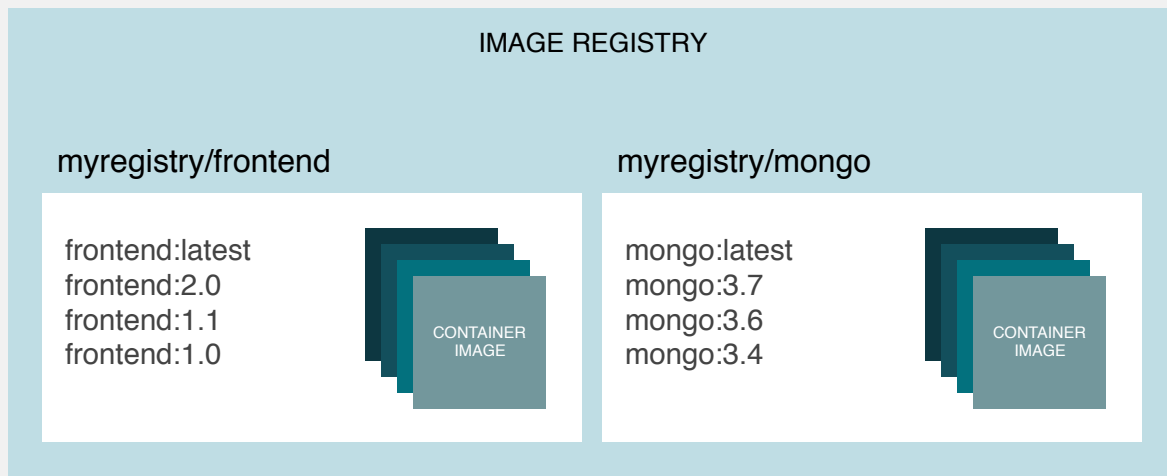
# OPENSIFT CONTAINER STORAGE



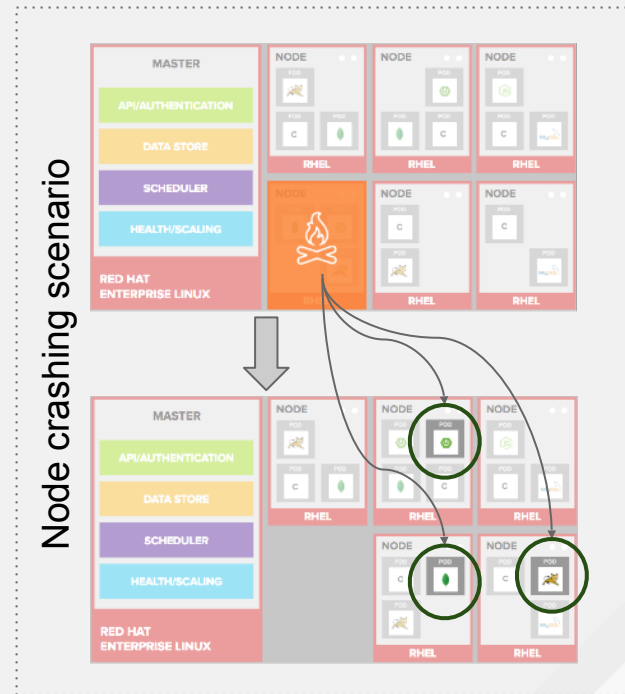
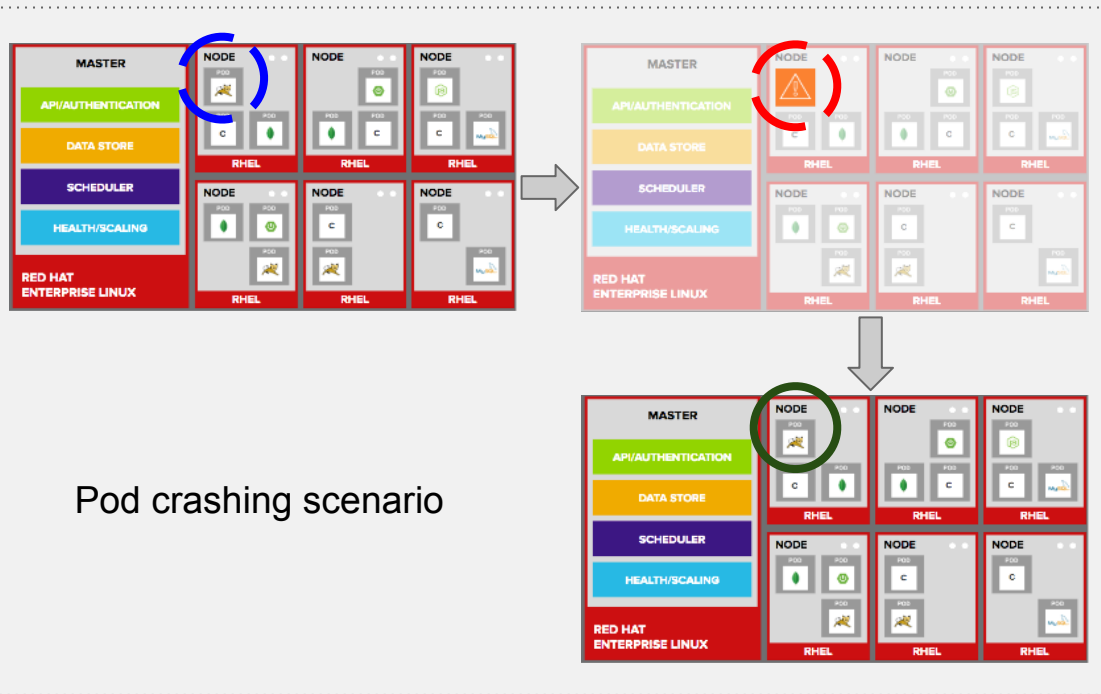
# OPENSIFT CONTAINER REGISTRY OCR



# VERSIONING and RBAC



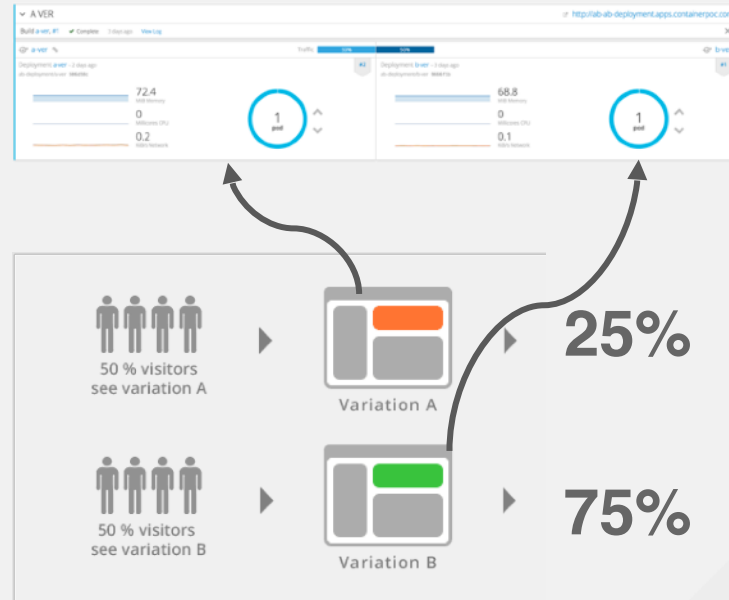
# SELF-HEALING PODS AND NODES



# DEPLOYMENT STRATEGIES

Rolling, Recreate, Blue/Green, A/B

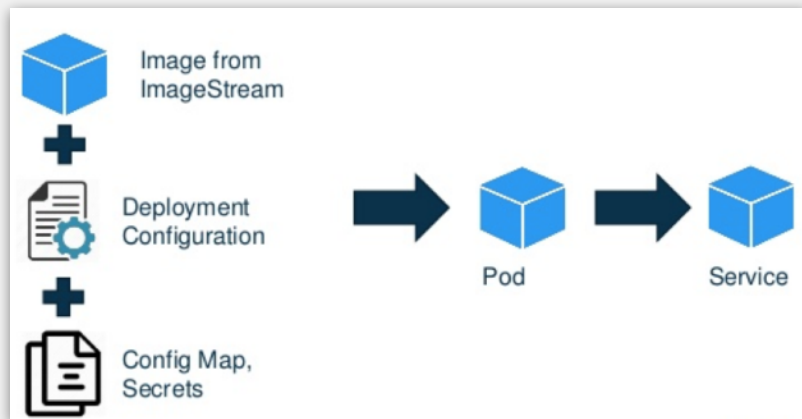
- Support for zero downtime deployment
- Canary deployment
  - A/B
  - Blue/Green
- Custom deployment strategy
  - Pre deployment hooks
  - Post deployment hooks
- Deployment triggers



# CONFIGURATION MANAGEMENT

ConfigMap, Secret, Environment Parameters, Image Promotion

- The ConfigMap object provides mechanisms to inject containers with configuration data.
- The ConfigMap API object holds key-value pairs of configuration data that can be consumed in pods.
- Configuration data can be consumed in pods in a variety of ways. A ConfigMap can be used to:
  - Populate the value of environment variables.
  - Set command-line arguments in a container.
  - Populate configuration files in a volume.
- Secrets provides data encryption for data at rest and in motion



# OPENSHIFT CLUSTER CONSOLE

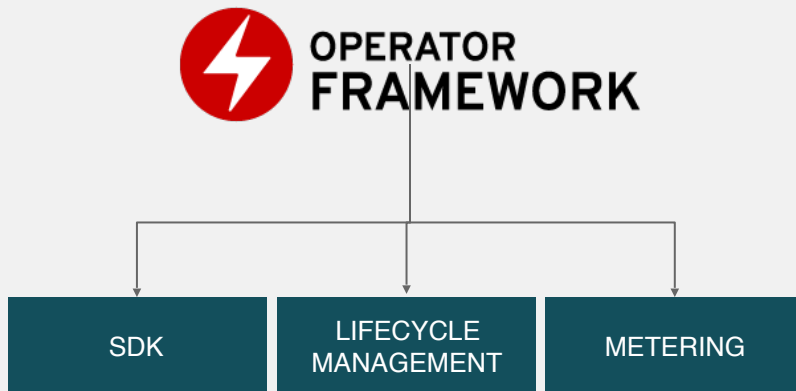
## Cluster Management

- Overall Health
- Utilization
- Cluster Nodes
- Storage
- Networking
- Projects
- Workloads
- Built-in Dashboards
- Custom Dashboards
- RBAC
- More...

The screenshot displays the OpenShift Cluster Console interface. The top navigation bar shows 'OPENSHIFT CONTAINER PLATFORM' and 'Cluster Console'. A sidebar on the left contains a menu with options: Home, Status, Search, Events, Operators, Workloads, Networking, Storage, Builds, Monitoring, and Administration (with sub-items: Projects, Namespaces, Nodes, Service Accounts, Roles, Role Bindings, Resource Quotas, and CRDs). The main content area is titled 'Project: default' and shows the 'Status of default' section. This section includes a 'Health' overview with four metrics: Kubernetes API (UP, All good), OpenShift Console (UP, All good), Alerts Firing (0 Alerts), and Crashlooping Pods (0 Pods). Below this is an 'Events' section with a 'View All' link. It features filters for 'All Types' and 'All Categories', and a search bar 'Filter Events by name or message...'. The events list shows two events from 'template-service-broker' and 'ansible-service-broker', both generated from the 'service-catalog-controller-manager' and successfully fetching catalog entries. Each event is timestamped '8 minutes ago' and '10 minutes ago' respectively, with a frequency of '289 times in the last 2 days'. A timeline on the left indicates 'Streaming events...' and 'There are no events before 10 minutes ago'.

# OPERATOR FRAMEWORK

Operators codify operational knowledge and workflows to automate lifecycle management of containerized applications with Kubernetes



Operators **are only** targeting the **platform**, But also all **workloads** running on top of it!



# OPERATOR HUB COMMUNITY CERTIFIED & SUPPORTED OPENSOURCE OPERATORS

- Accessible to admins only
- Discovery/install of all optional components and apps
- Upstream and downstream content
- ISV partners will support their Operators

TYPES OF  
OPERATORS  
Red Hat Products  
ISV Partners  
Community

