



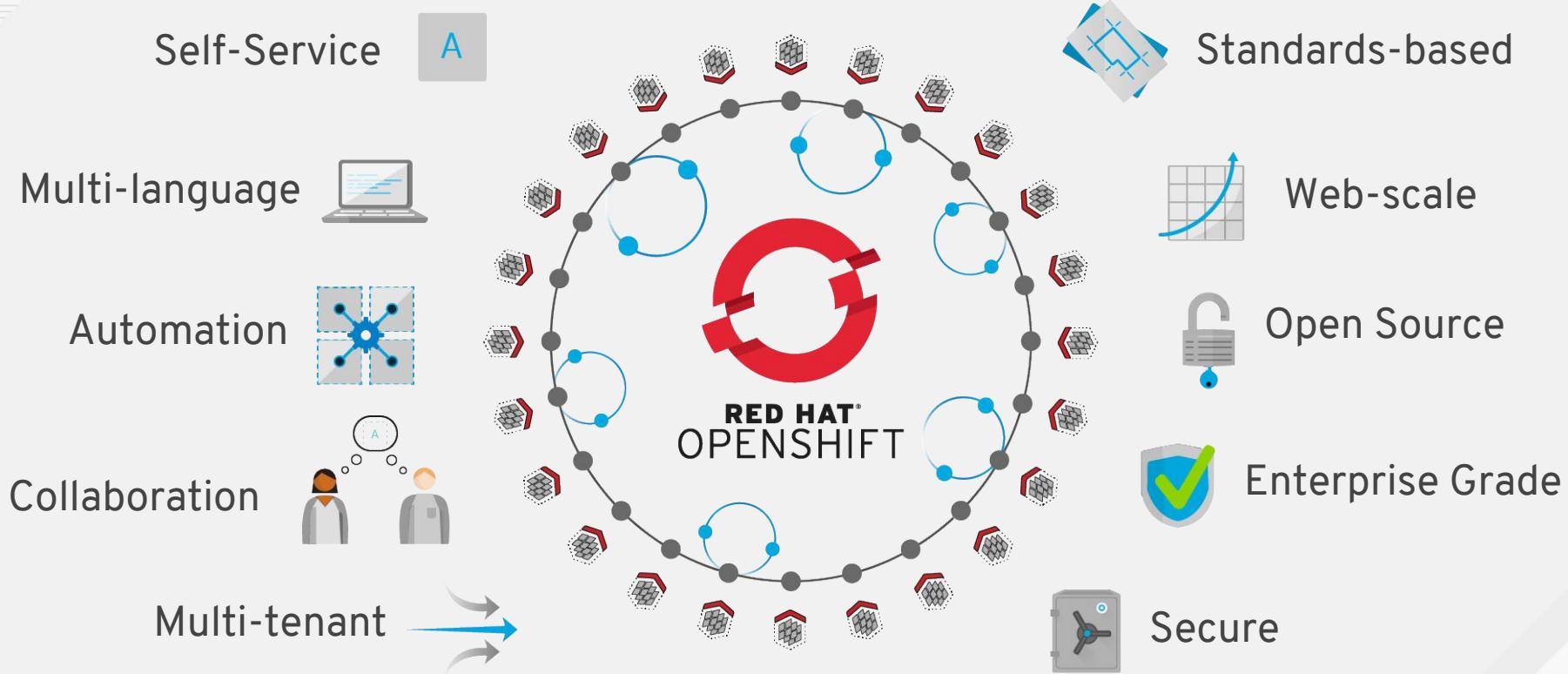
redhat.

OPENSHIFT CONTAINER PLATFORM TECHNICAL OVERVIEW

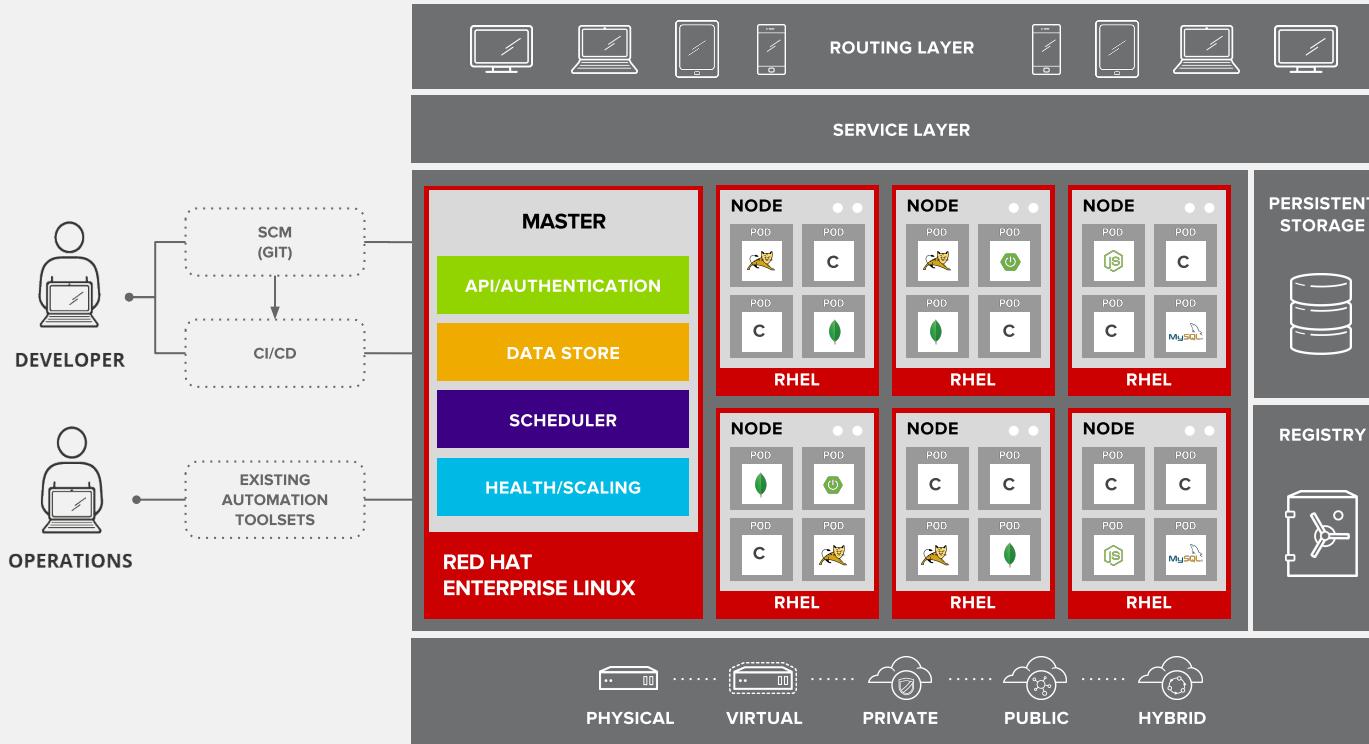
Presenter

Presenter's title

Date



OPENShift ARCHITECTURE





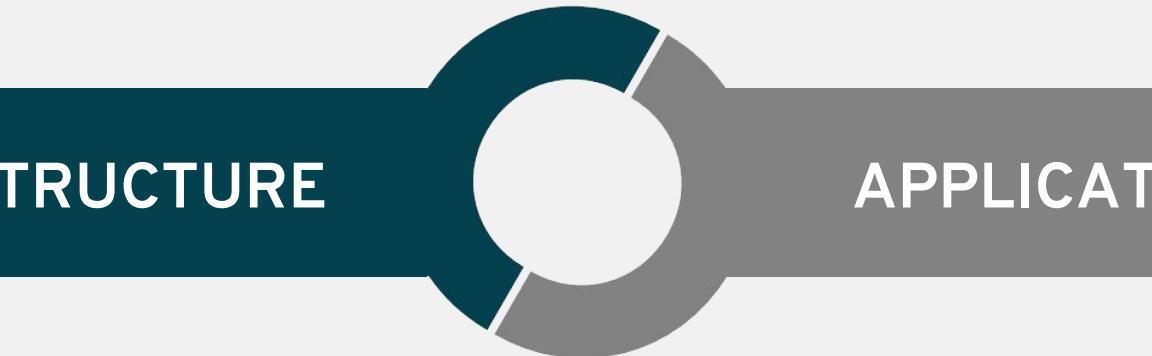
LINUX CONTAINERS

WHAT ARE CONTAINERS?

It Depends Who You Ask

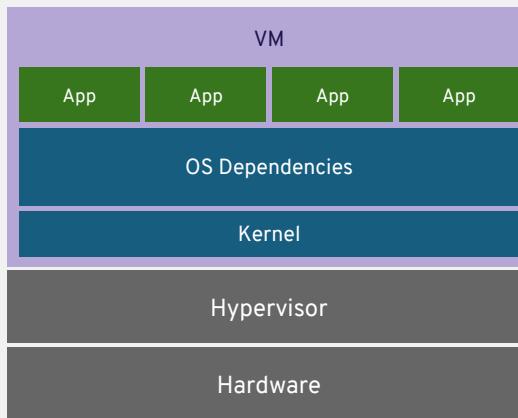
INFRASTRUCTURE

APPLICATIONS

- 
- Application processes on a shared kernel
 - Simpler, lighter, and denser than VMs
 - Portable across different environments
 - Package apps with all dependencies
 - Deploy to any environment in seconds
 - Easily accessed and shared

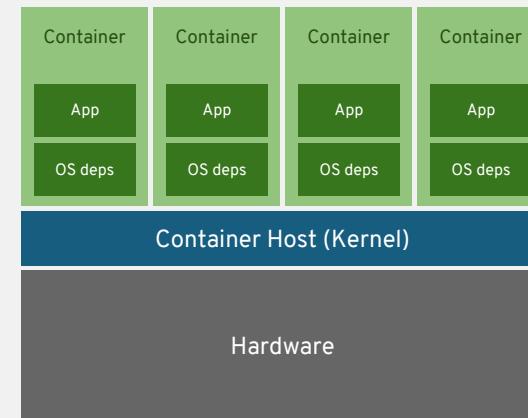
VIRTUAL MACHINES AND CONTAINERS

VIRTUAL MACHINES



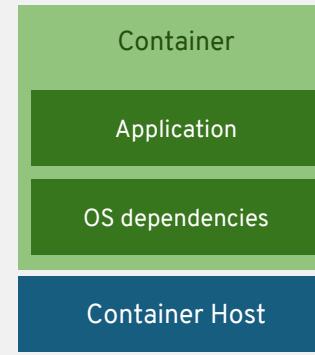
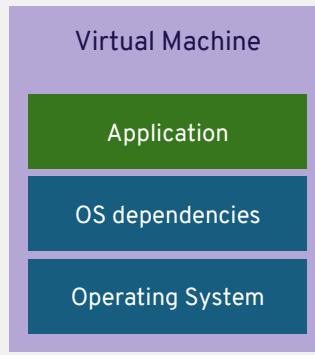
virtual machines are isolated
apps are not

CONTAINERS



containers are isolated
so are the apps

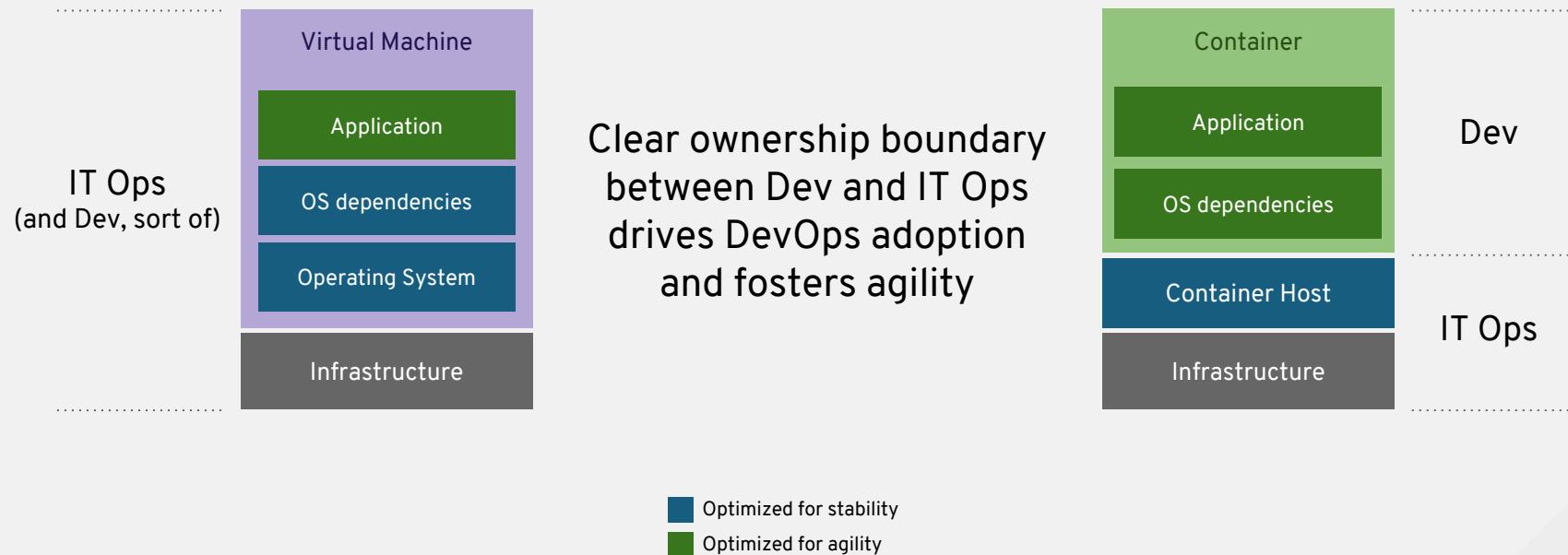
VIRTUAL MACHINES AND CONTAINERS



- + VM Isolation
- Complete OS
- Static Compute
- Static Memory
- High Resource Usage

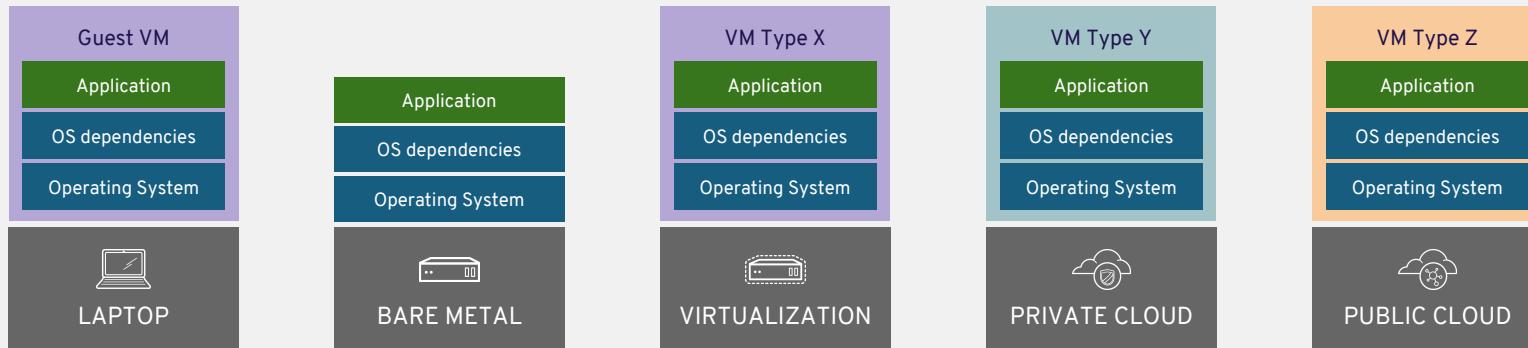
- + Container Isolation
- + Shared Kernel
- + Burstable Compute
- + Burstable Memory
- + Low Resource Usage

VIRTUAL MACHINES AND CONTAINERS



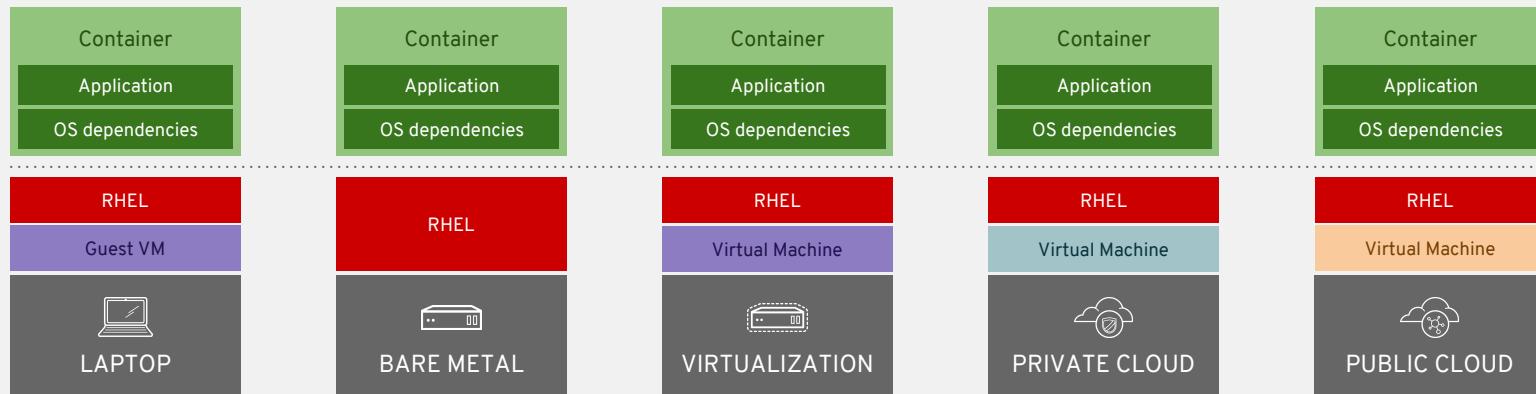
APPLICATION PORTABILITY WITH VM

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

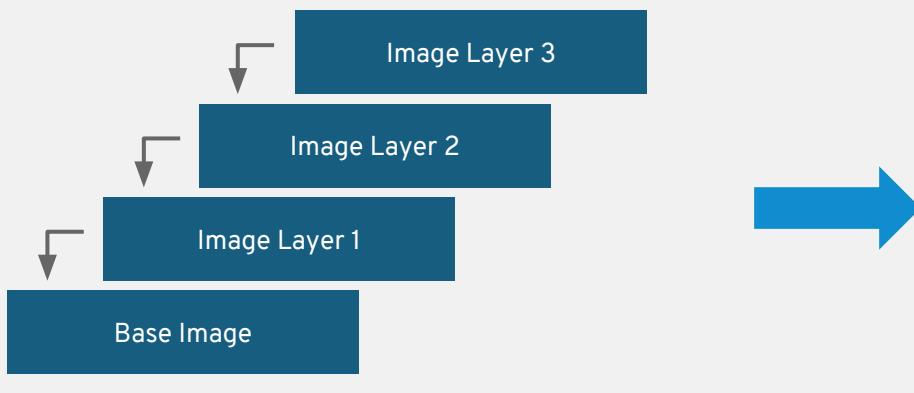


APPLICATION PORTABILITY WITH CONTAINERS

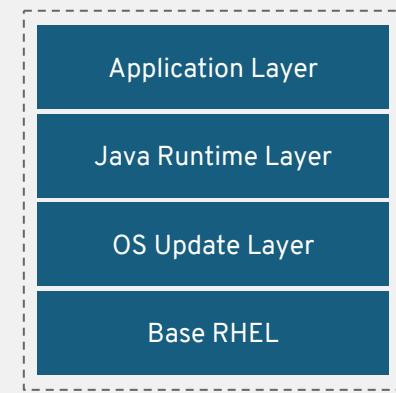
RHEL Containers + RHEL Host = Guaranteed Portability
Across Any Infrastructure



RAPID SECURITY PATCHING USING CONTAINER IMAGE LAYERING



Container Image Layers



Example Container Image



cri-o

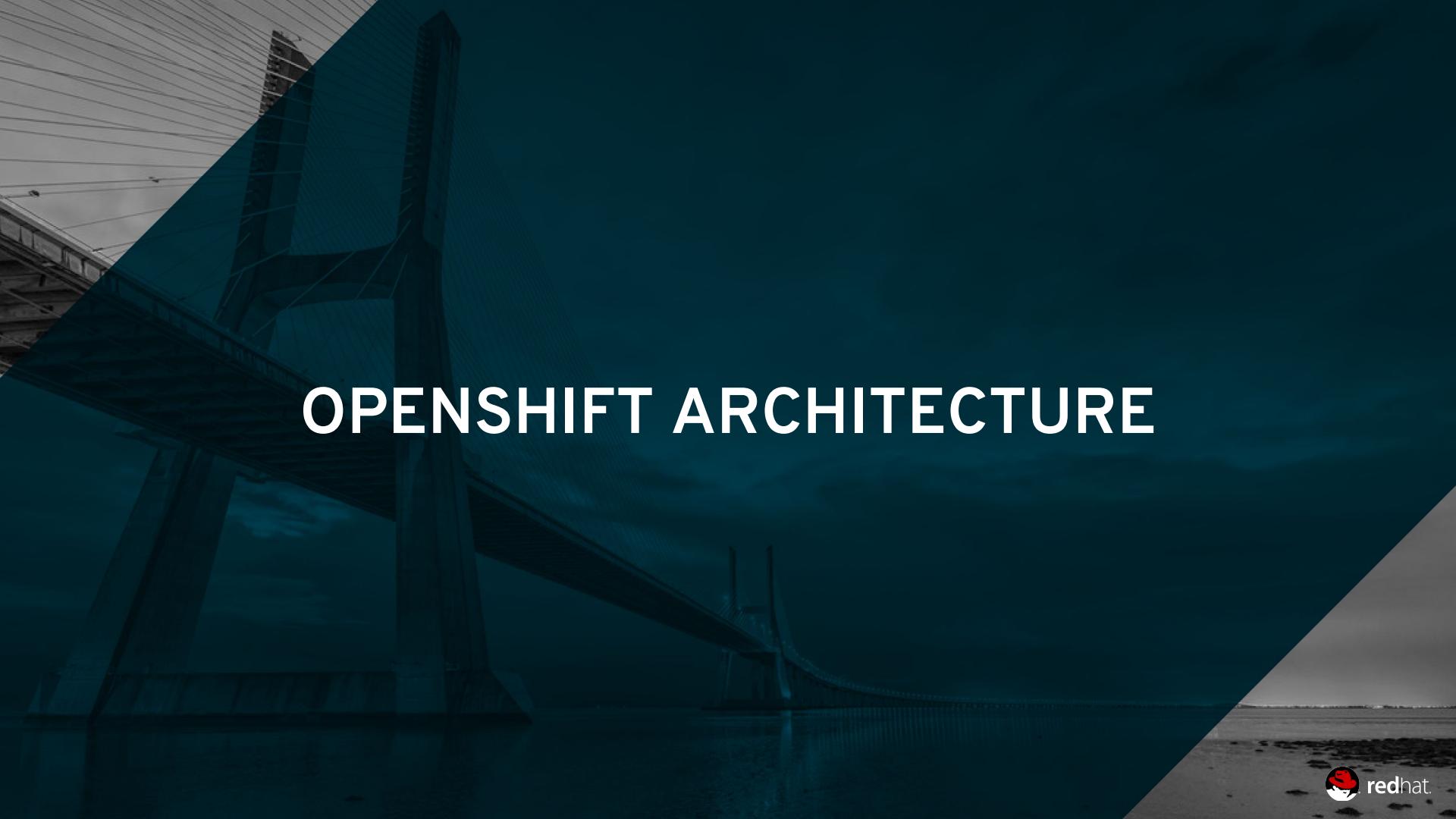
A lightweight, OCI-compliant container runtime

Optimized for
Kubernetes

Any OCI-compliant
container from any
OCI registry
(including docker)

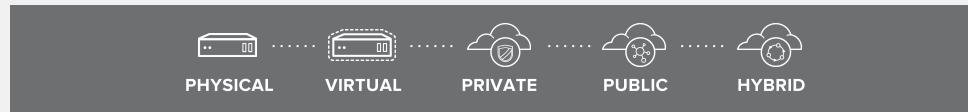
Improve Security and
Performance at scale

Available in OpenShift Online (soon)
Tech Preview in OCP 3.7, GA in OCP 3.8

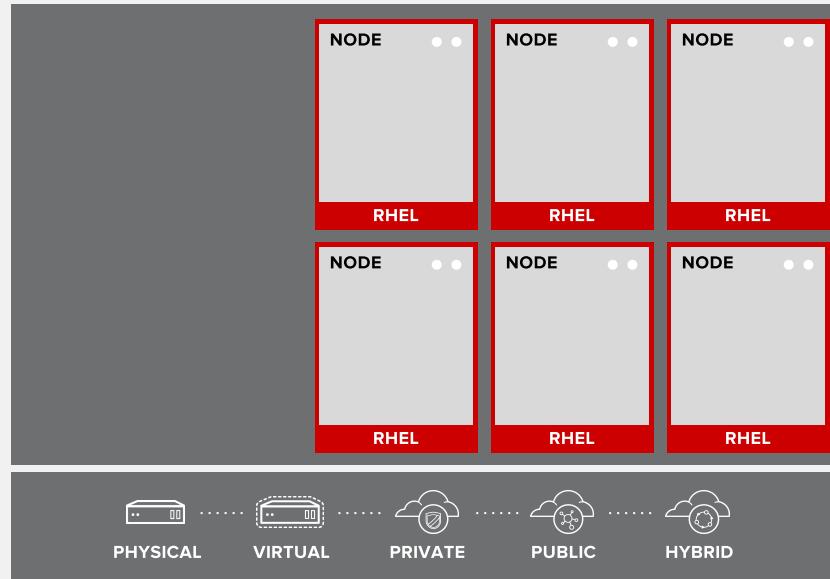


OPENSHIFT ARCHITECTURE

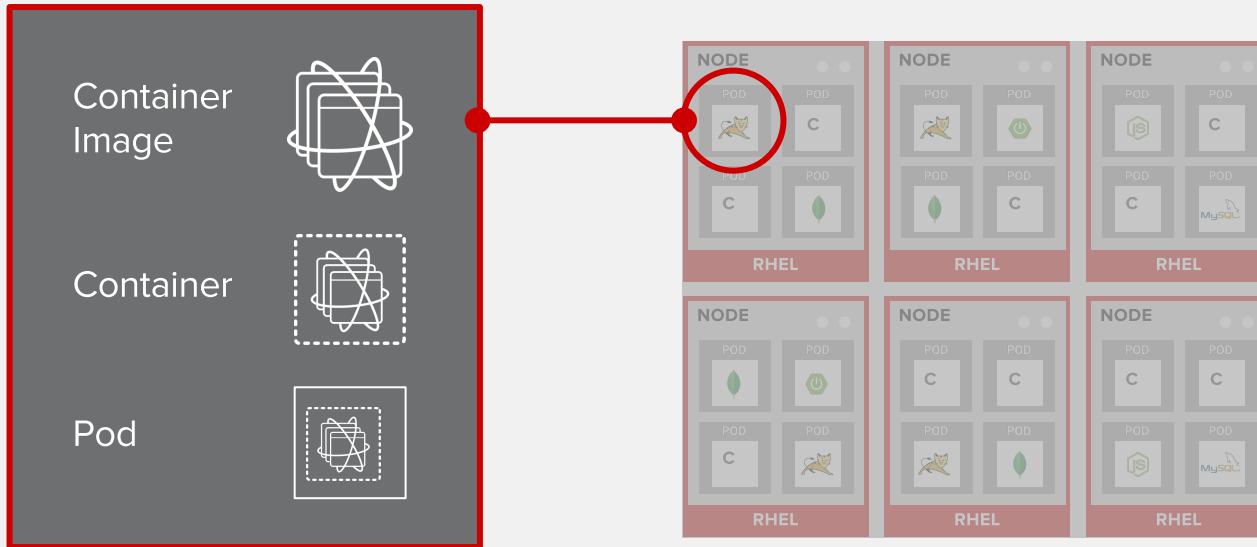
YOUR CHOICE OF INFRASTRUCTURE



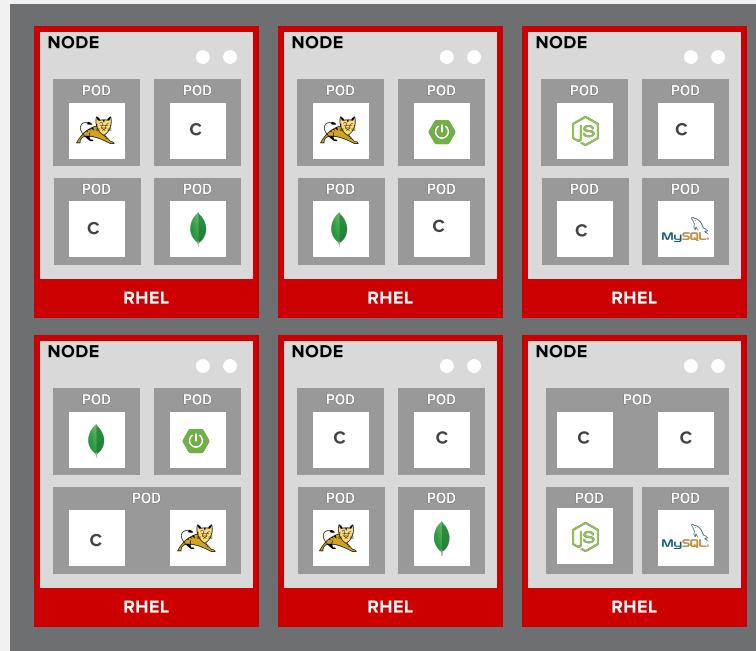
NODES RHEL INSTANCES WHERE APPS RUN



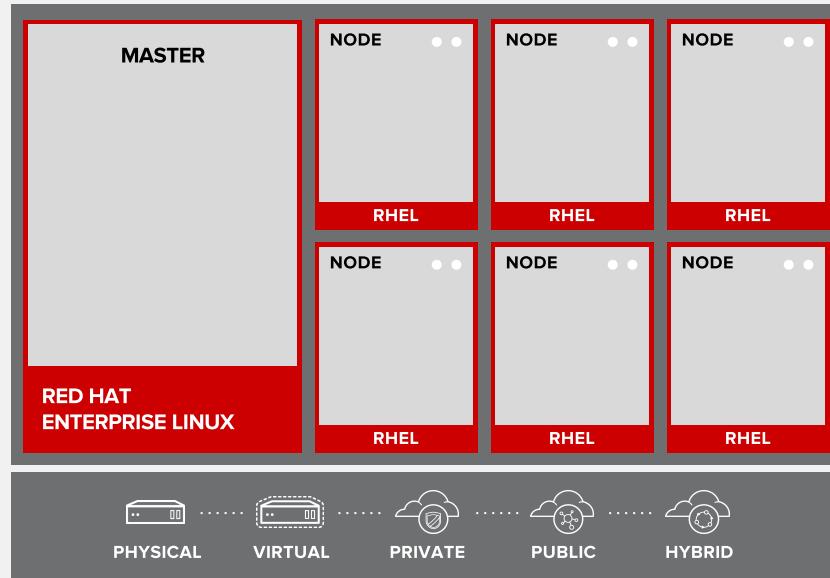
APPS RUN IN CONTAINERS



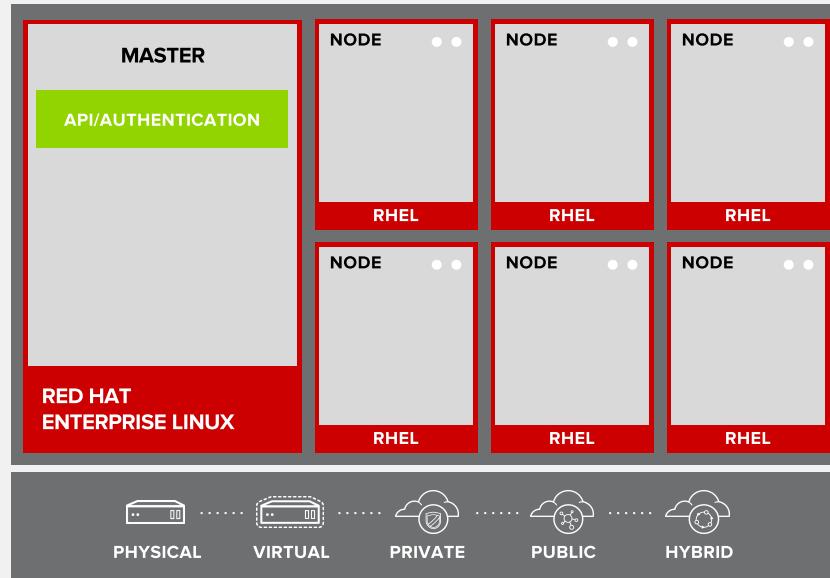
PODS ARE THE UNIT OF ORCHESTRATION



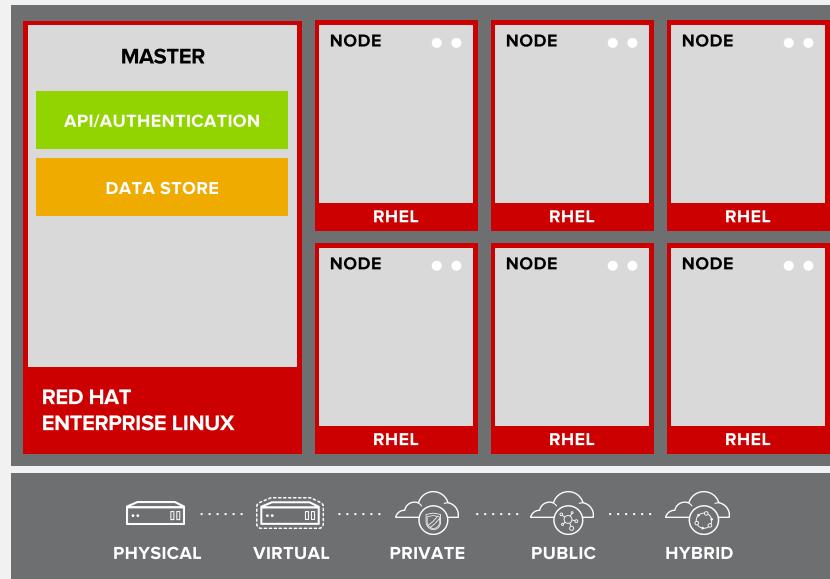
MASTERS ARE THE CONTROL PLANE



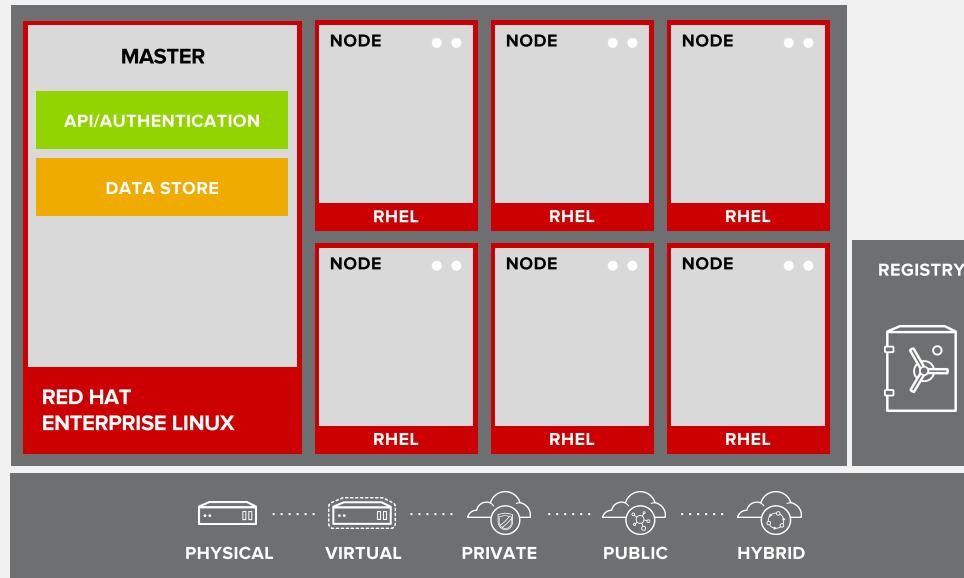
API AND AUTHENTICATION



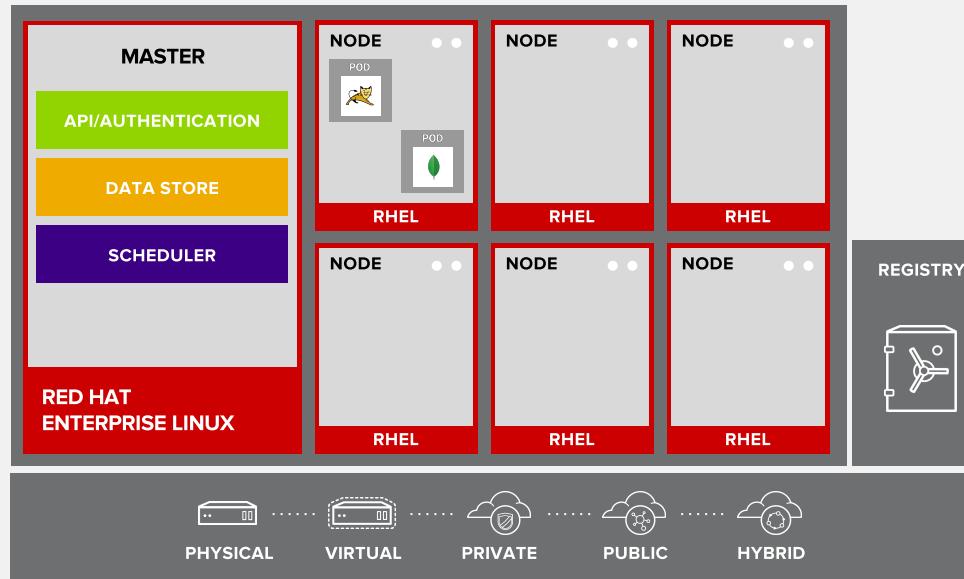
DESIRED AND CURRENT STATE



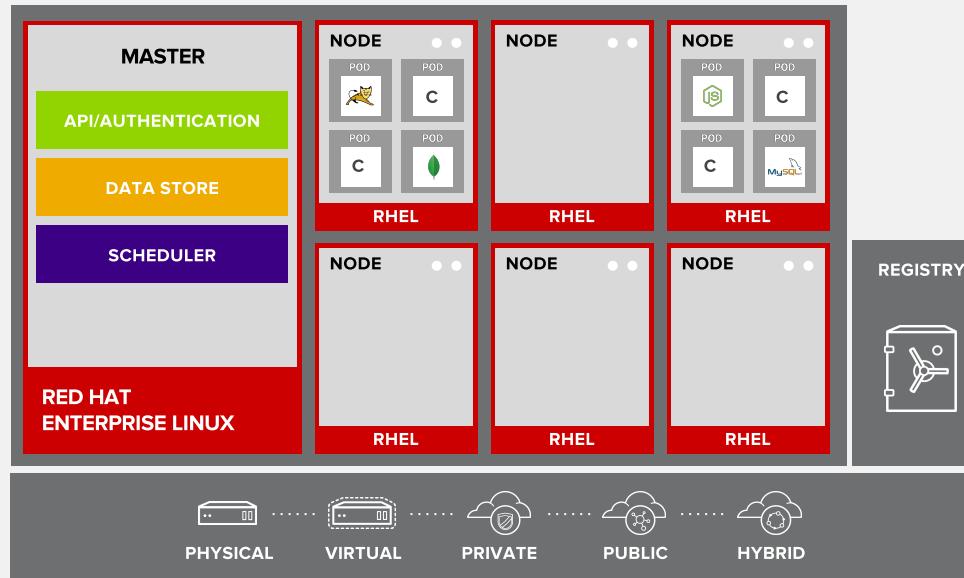
INTEGRATED CONTAINER REGISTRY



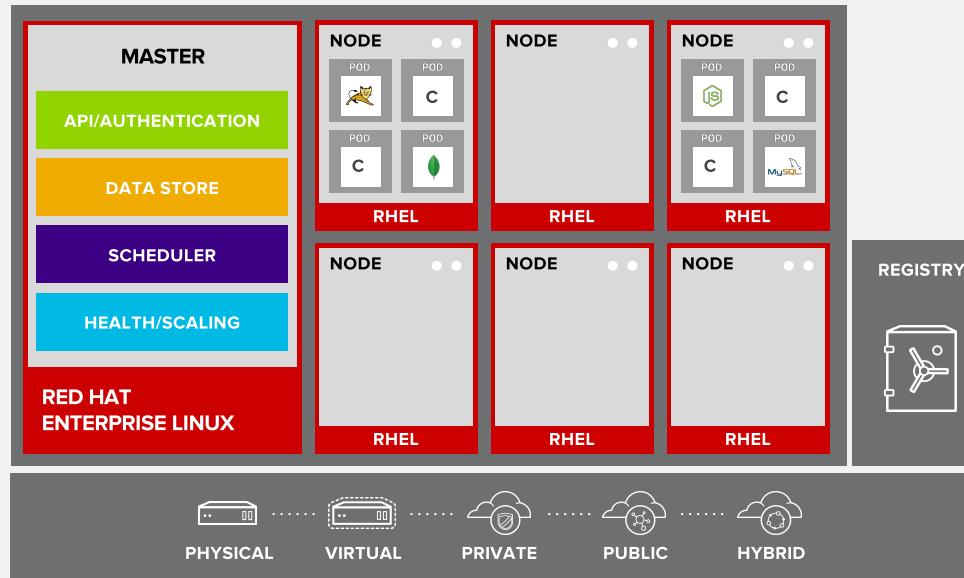
ORCHESTRATION AND SCHEDULING



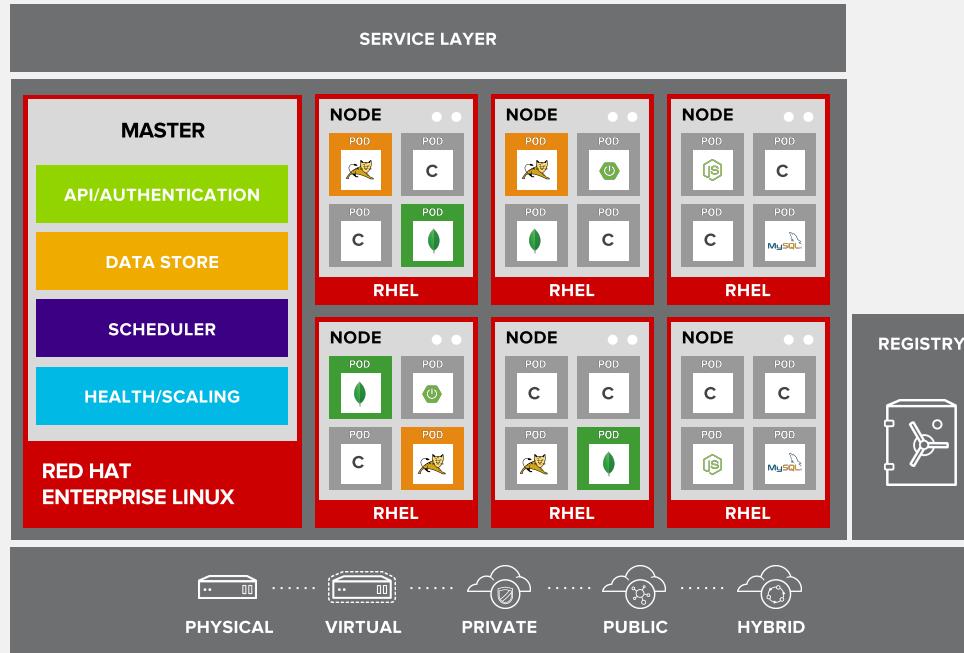
PLACEMENT BY POLICY



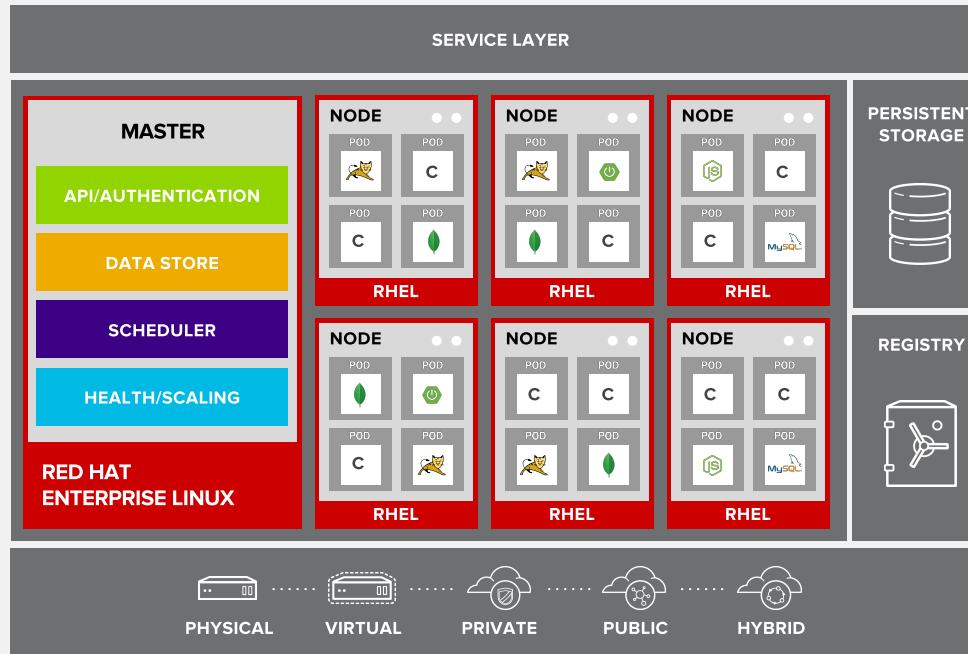
AUTOSCALING PODS



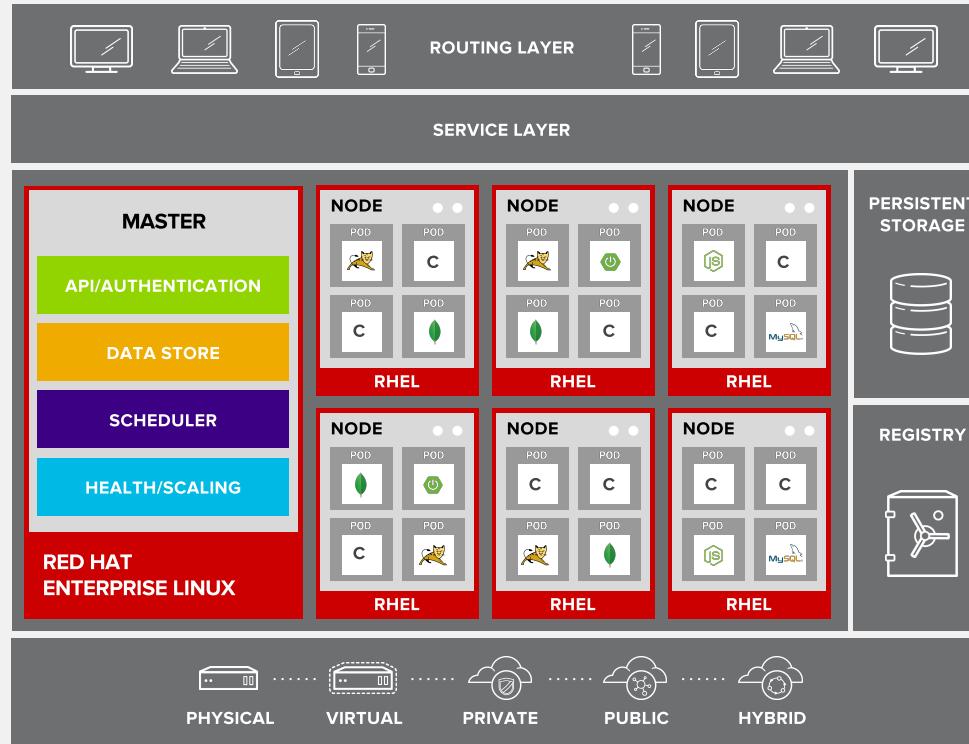
SERVICE DISCOVERY



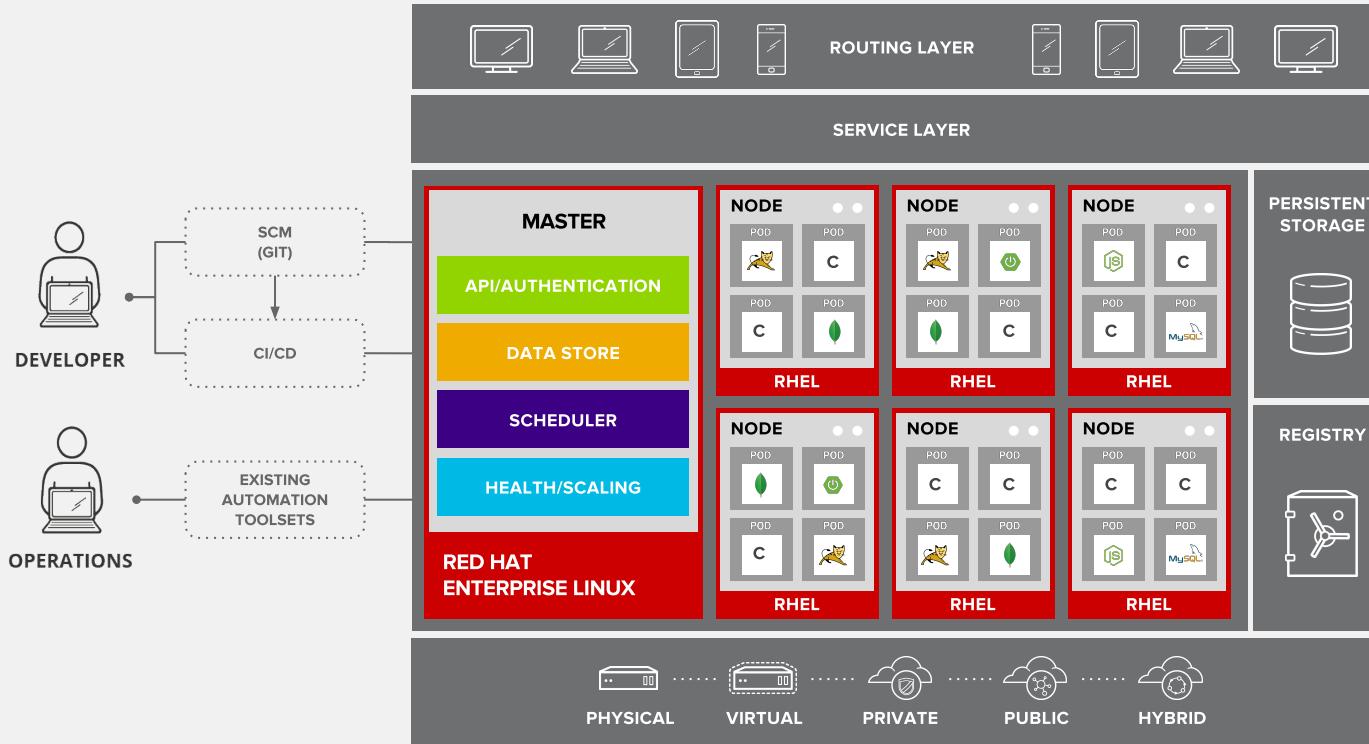
PERSISTENT DATA IN CONTAINERS



ROUTING AND LOAD-BALANCING



ACCESS VIA WEB, CLI, IDE AND API

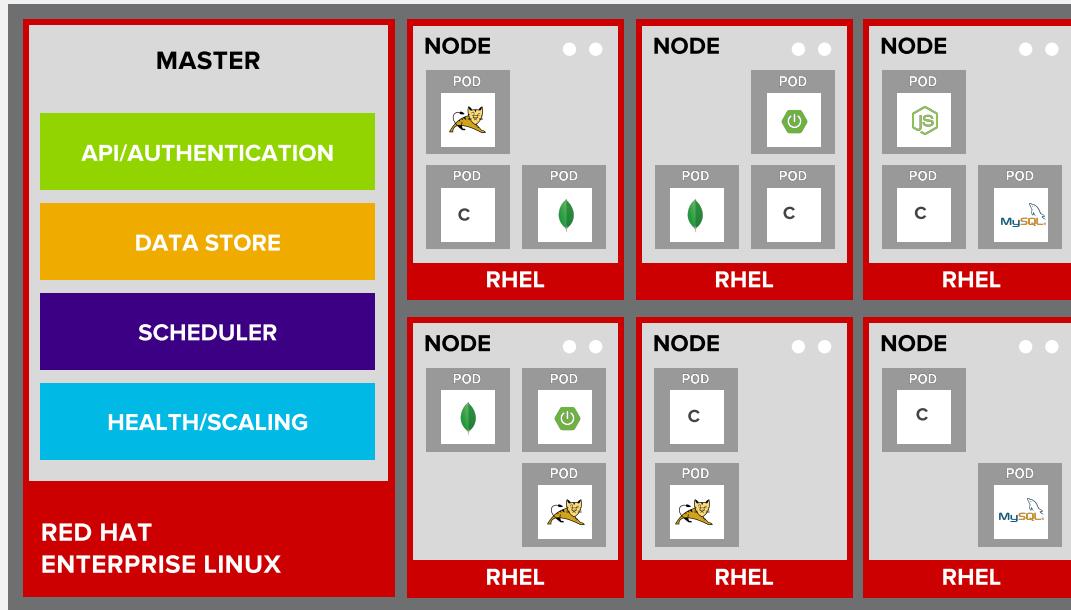


A large cable-stayed bridge with a dark teal overlay and a white diagonal band.

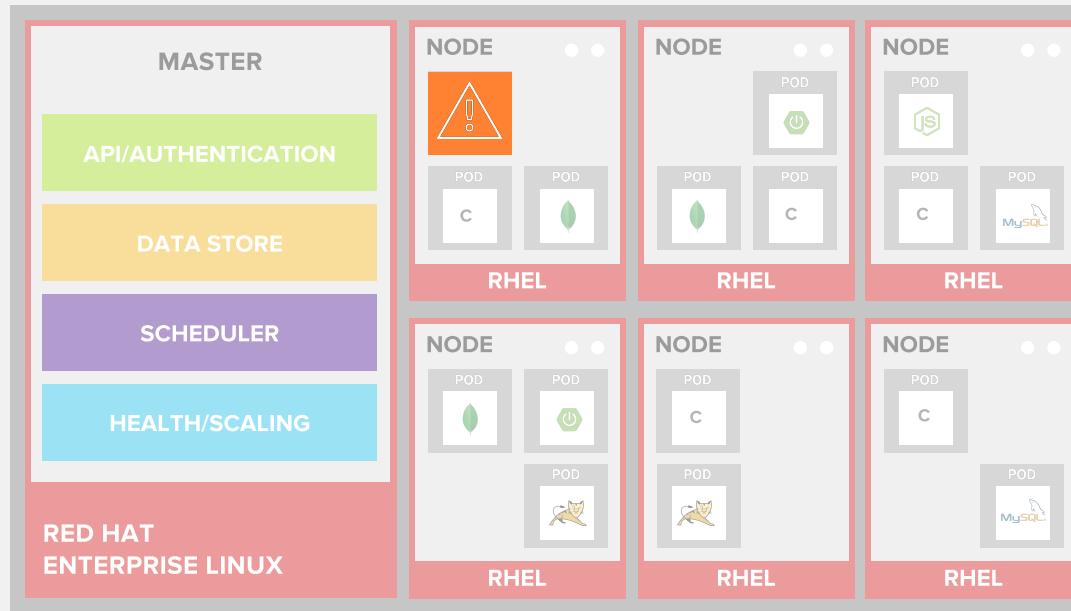
TECHNICAL DEEP DIVE

MONITORING APPLICATION HEALTH

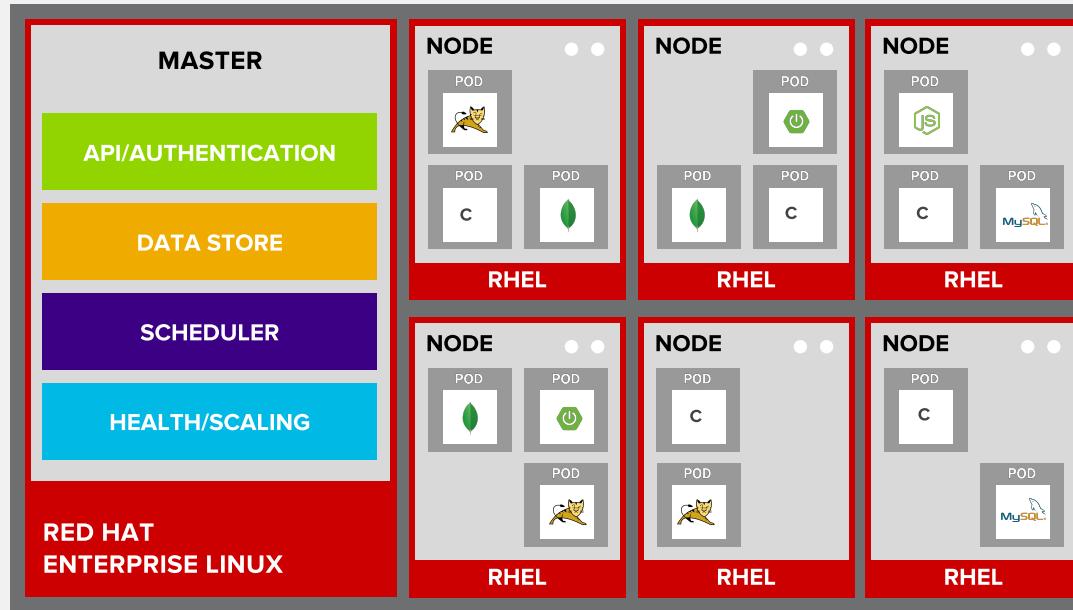
AUTO-HEALING FAILED PODS



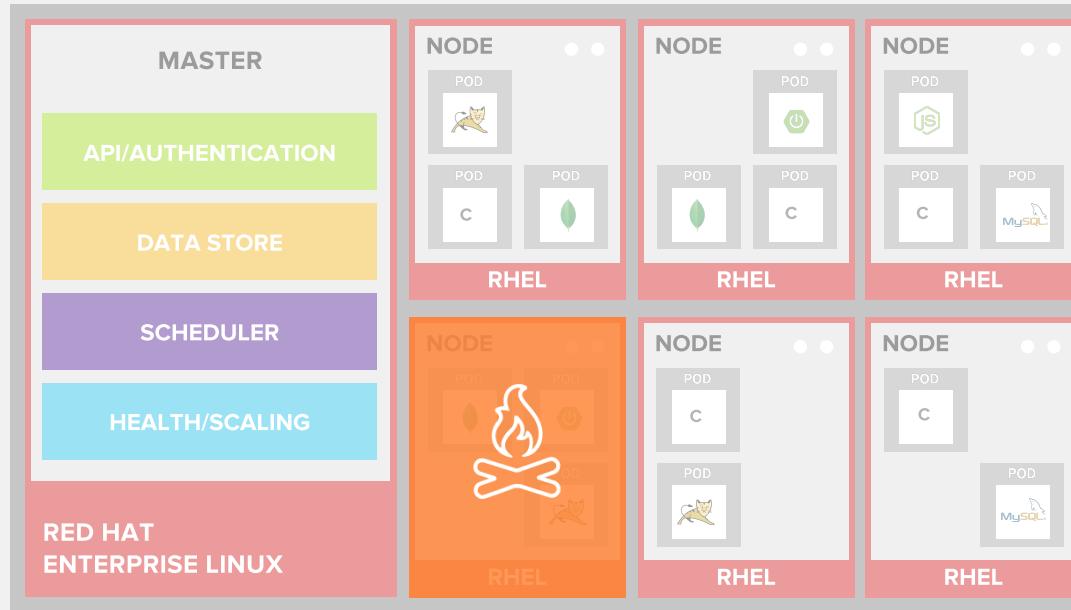
AUTO-HEALING FAILED CONTAINERS



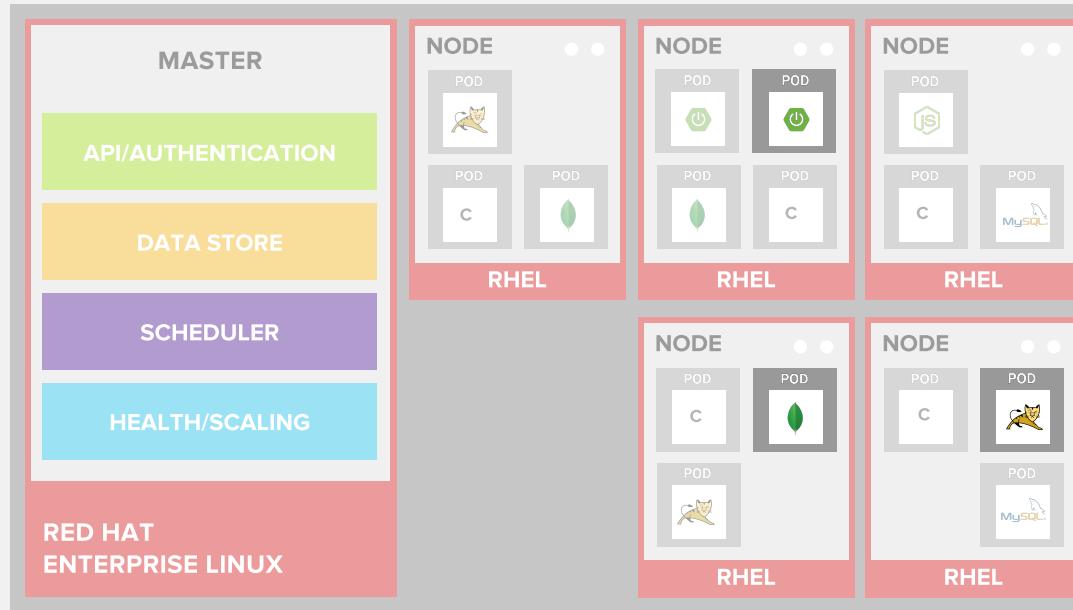
AUTO-HEALING FAILED CONTAINERS



AUTO-HEALING FAILED CONTAINERS

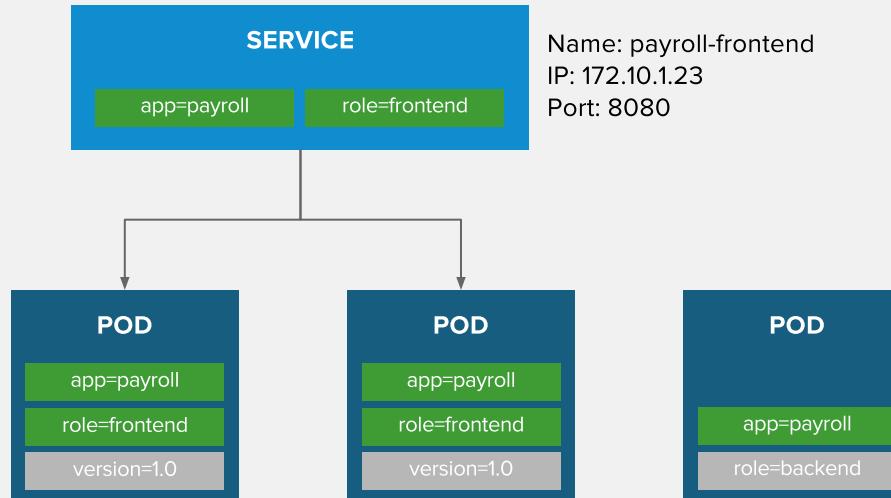


AUTO-HEALING FAILED CONTAINERS

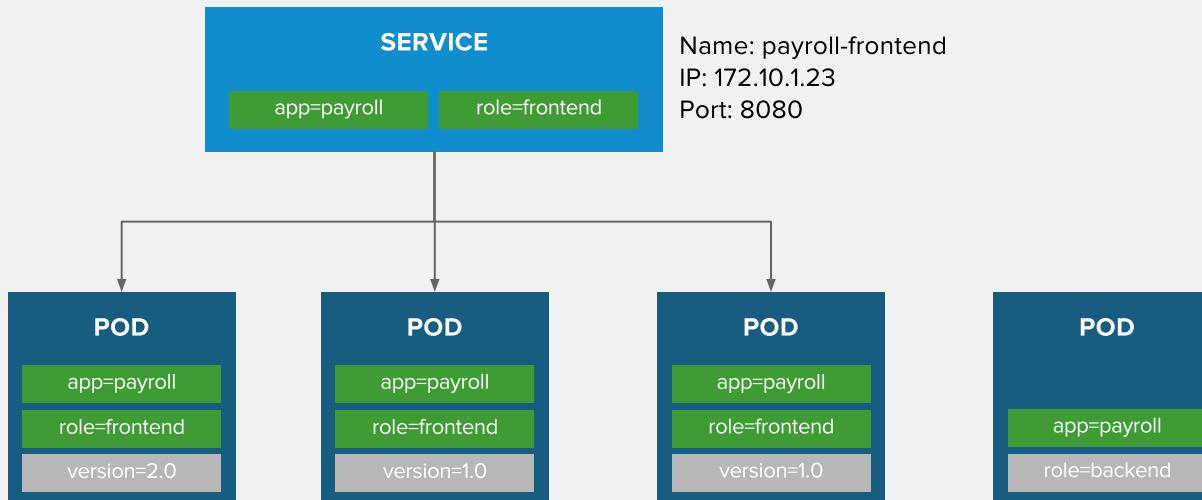


NETWORKING

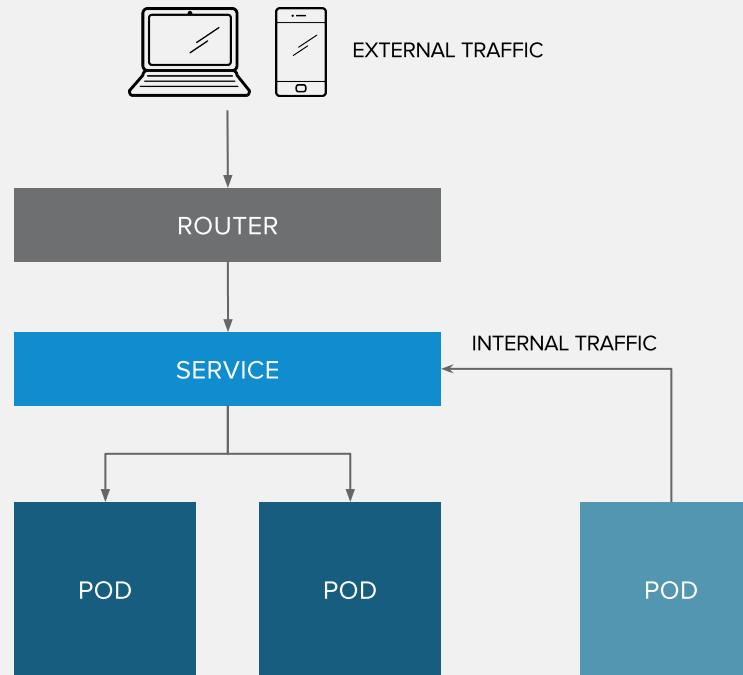
BUILT-IN SERVICE DISCOVERY INTERNAL LOAD-BALANCING



BUILT-IN SERVICE DISCOVERY INTERNAL LOAD-BALANCING

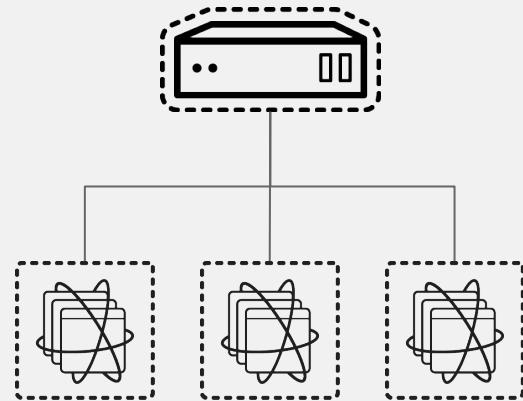


ROUTE EXPOSES SERVICES EXTERNALLY



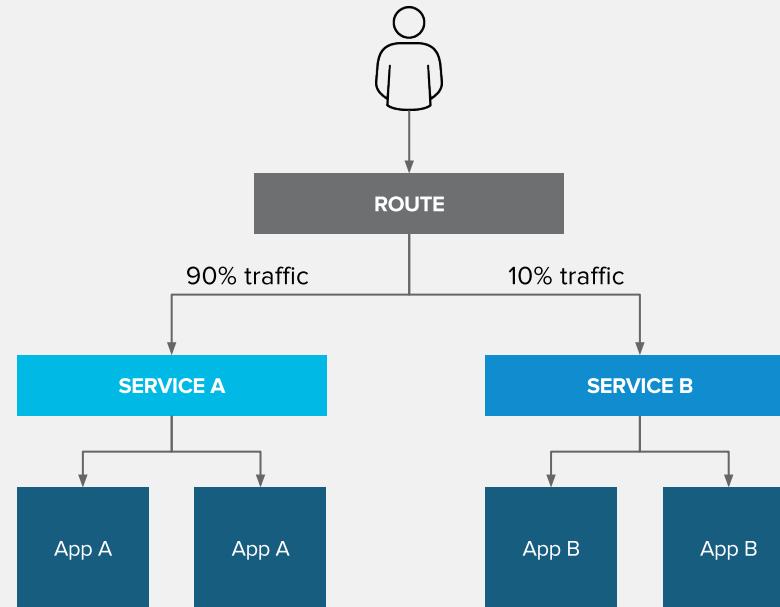
ROUTING AND EXTERNAL LOAD-BALANCING

- Pluggable routing architecture
 - HAProxy Router
 - F5 Router
- Multiple-routers with traffic sharding
- Router supported protocols
 - HTTP/HTTPS
 - WebSockets
 - TLS with SNI
- Non-standard ports via cloud load-balancers, external IP, and NodePort



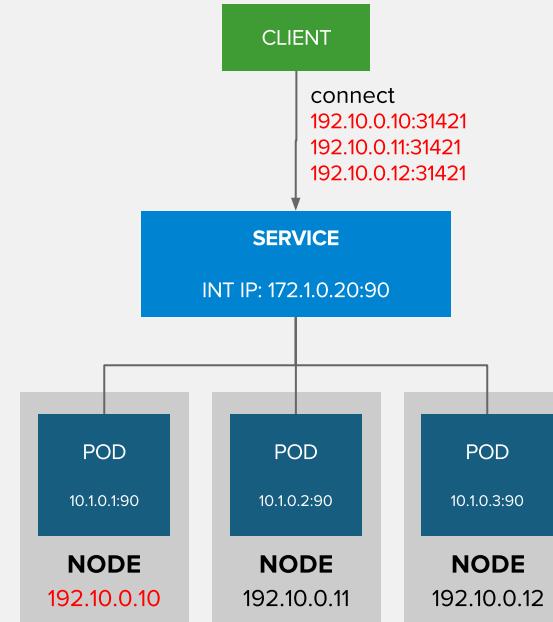
ROUTE SPLIT TRAFFIC

Split Traffic Between
Multiple Services For A/B
Testing, Blue/Green and
Canary Deployments



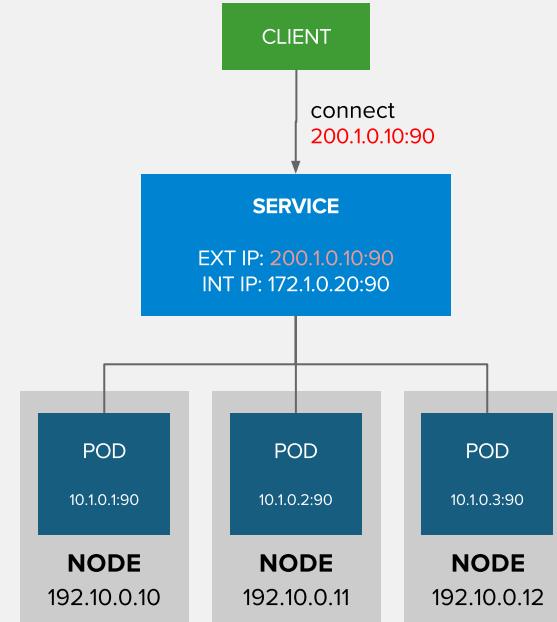
EXTERNAL TRAFFIC TO A SERVICE ON A RANDOM PORT WITH NODEPORT

- NodePort binds a service to a unique port on all the nodes
- Traffic received on any node redirects to a node with the running service
- Ports in 30K-60K range which usually differs from the service
- Firewall rules must allow traffic to all nodes on the specific port

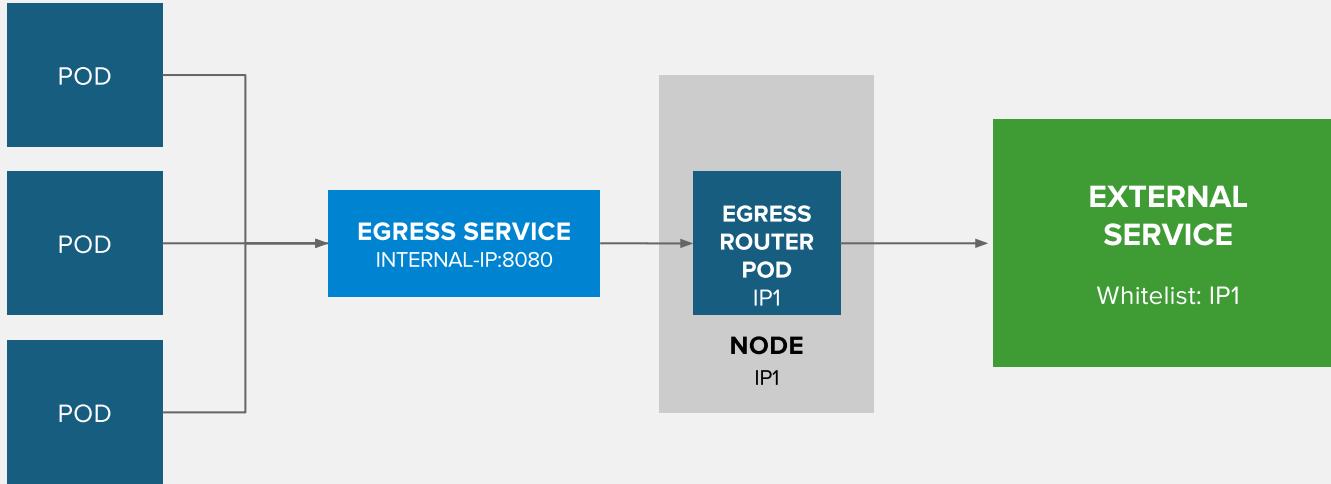


EXTERNAL TRAFFIC TO A SERVICE ON ANY PORT WITH INGRESS

- Access a service with an external IP on any TCP/UDP port, such as
 - Databases
 - Message Brokers
- Automatic IP allocation from a predefined pool using Ingress IP Self-Service
- IP failover pods provide high availability for the IP pool

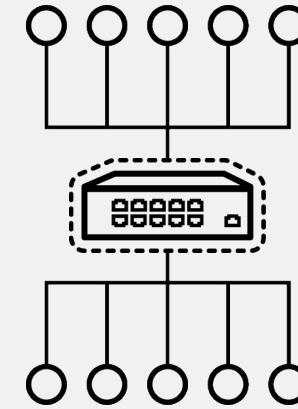


CONTROL OUTGOING TRAFFIC SOURCE IP WITH EGRESS ROUTER

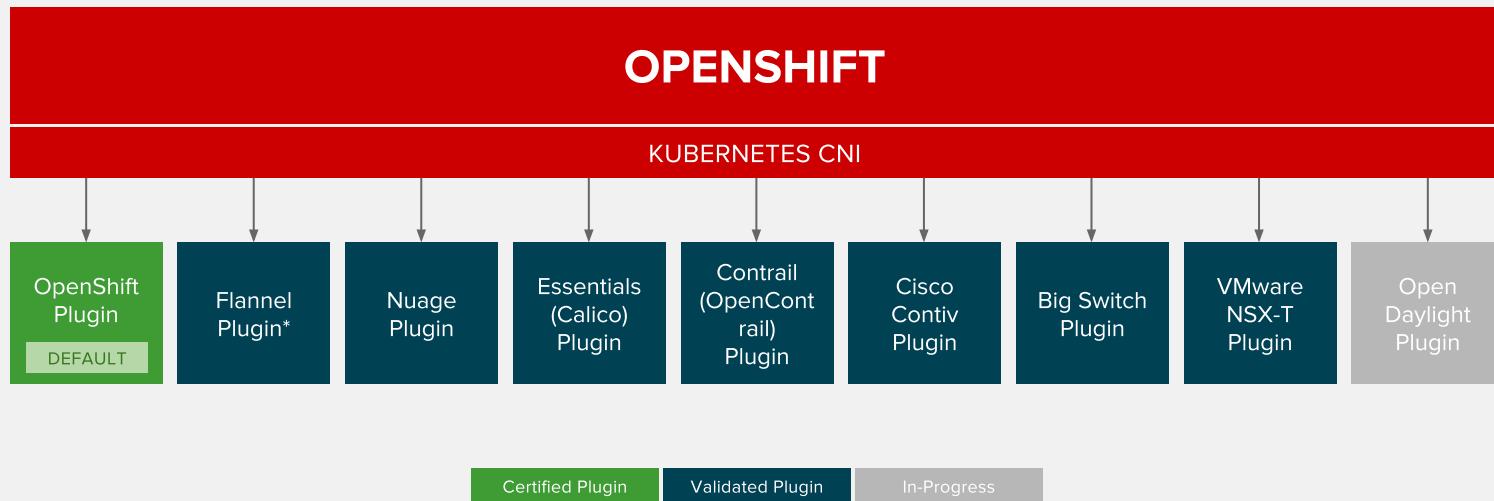


OPENShift NETWORKING

- Built-in internal DNS to reach services by name
- Split DNS is supported via SkyDNS
 - Master answers DNS queries for internal services
 - Other nameservers serve the rest of the queries
- Software Defined Networking (SDN) for a unified cluster network to enable pod-to-pod communication
- OpenShift follows the Kubernetes Container Networking Interface (CNI) plug-in model

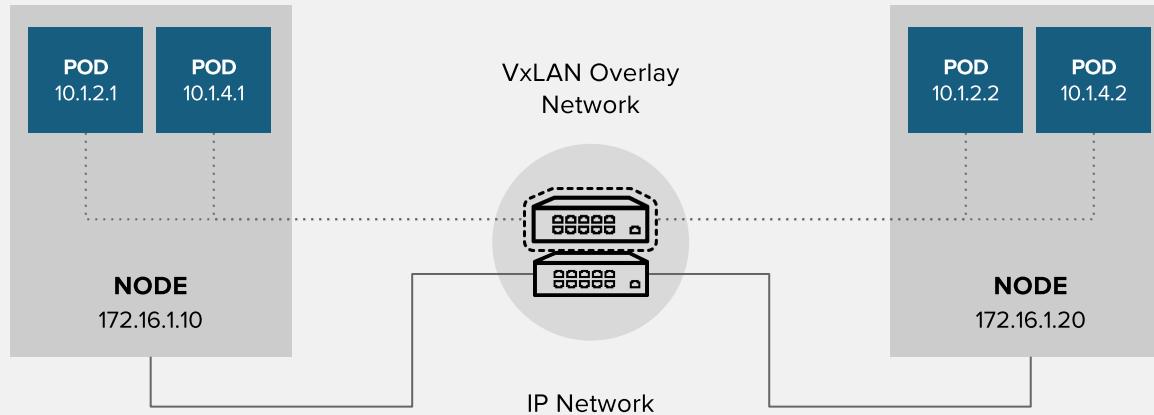


OPENShift NETWORK PLUGINS



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

OPENShift NETWORKING



OPENShift SDN

FLAT NETWORK (Default)

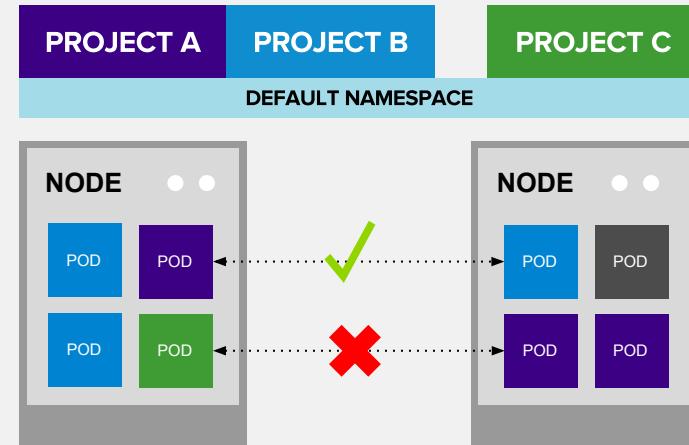
- All pods can communicate with each other across projects

MULTI-TENANT NETWORK

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

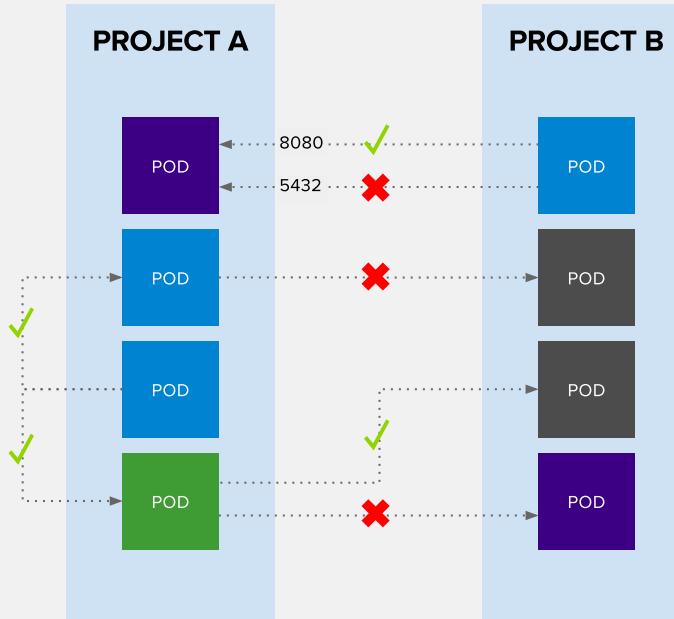
NETWORK POLICY (Tech Preview)

- Granular policy-based isolation



Multi-Tenant Network

OPENShift SDN - NETWORK POLICY



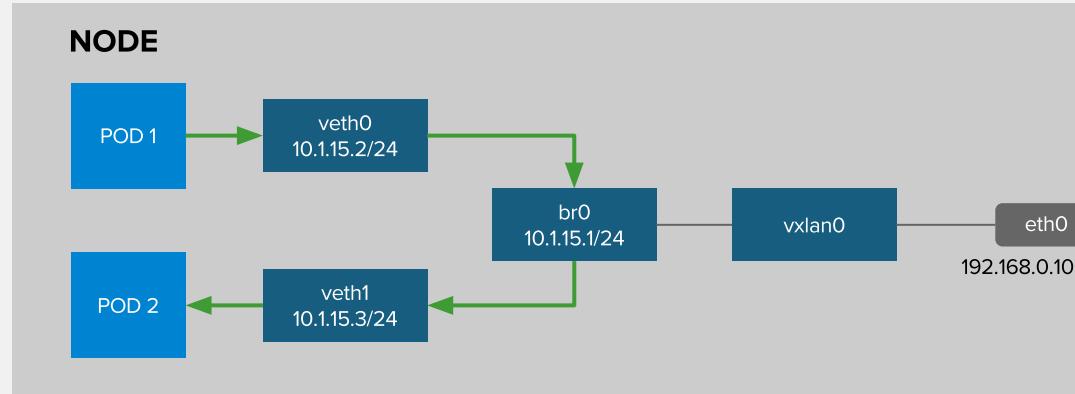
Example Policies

- Allow all traffic inside the project
- Allow traffic from green to gray
- Allow traffic to purple on 8080

```
apiVersion: extensions/v1beta1
kind: NetworkPolicy
metadata:
  name: allow-to-purple-on-8080
spec:
  podSelector:
    matchLabels:
      color: purple
  ingress:
    - ports:
        - protocol: tcp
          port: 8080
```

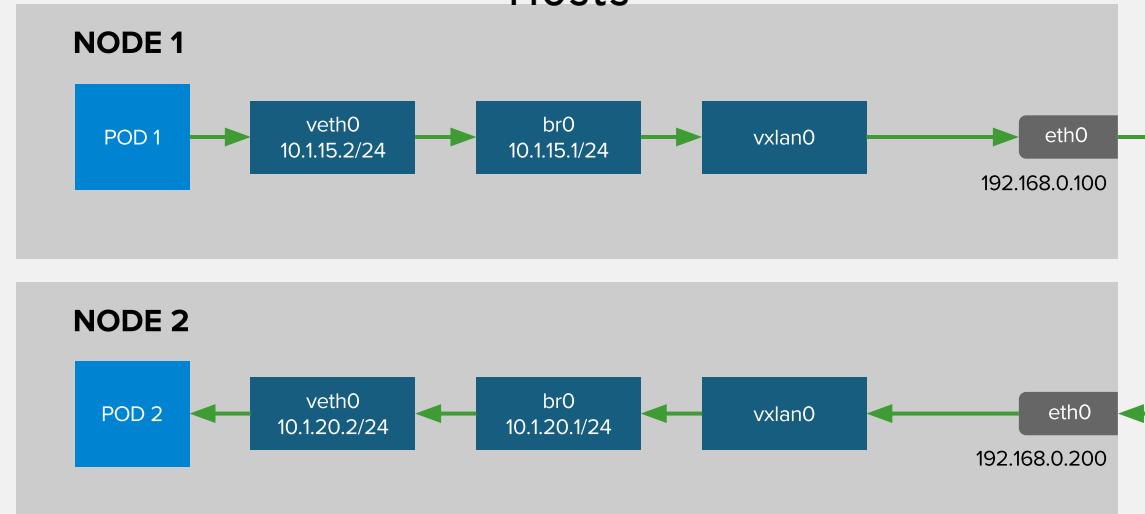
OPENShift SDN - OVS PACKET FLOW

Container to Container on the Same Host



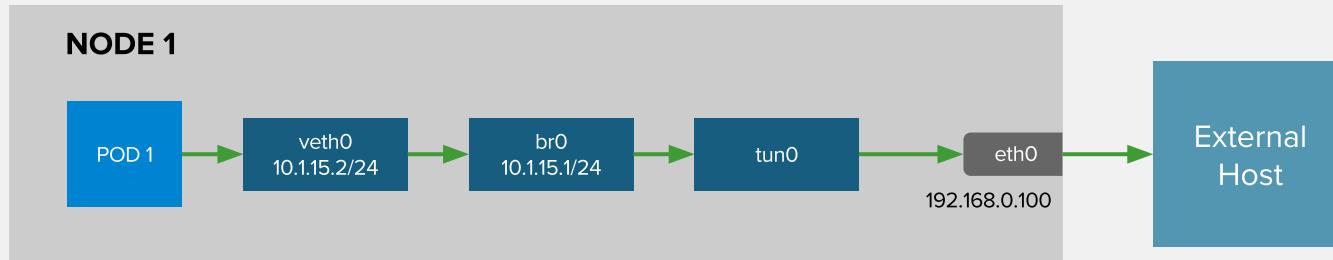
OPENShift SDN - OVS PACKET FLOW

Container to Container on the Different Hosts

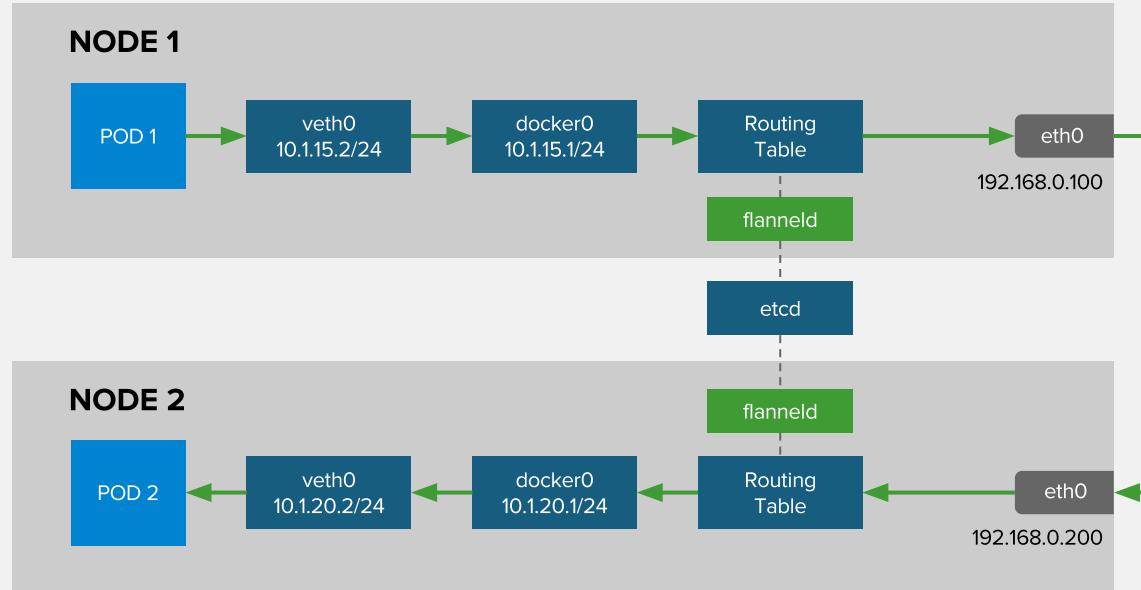


OPENShift SDN - OVS PACKET FLOW

Container Connects to External Host



OPENShift SDN WITH FLANNEL FOR OPENSTACK

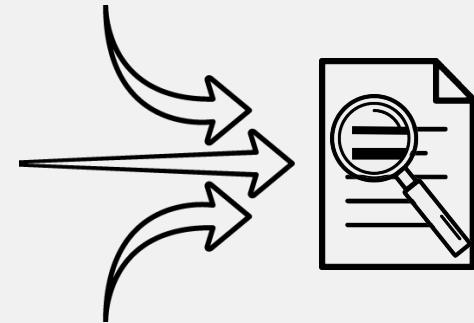


Flannel is minimally verified and is supported only and exactly as deployed in the OpenShift on OpenStack reference architecture <https://access.redhat.com/articles/2743631>

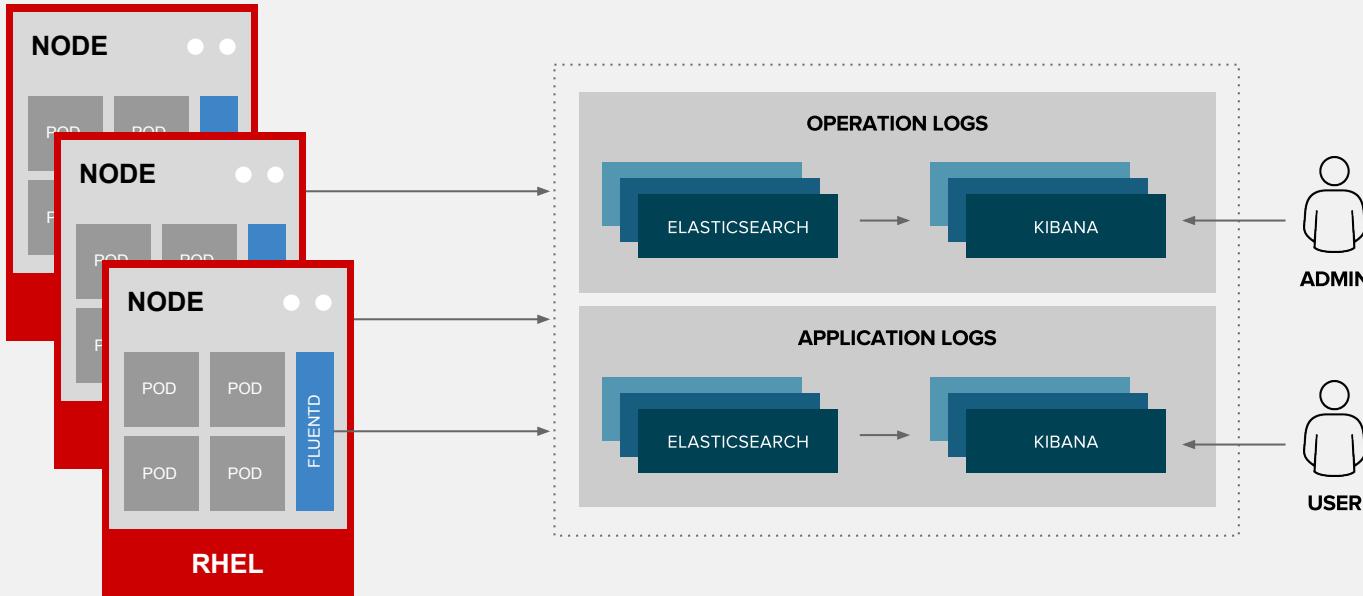
LOGGING & METRICS

CENTRAL LOG MANAGEMENT WITH EFK

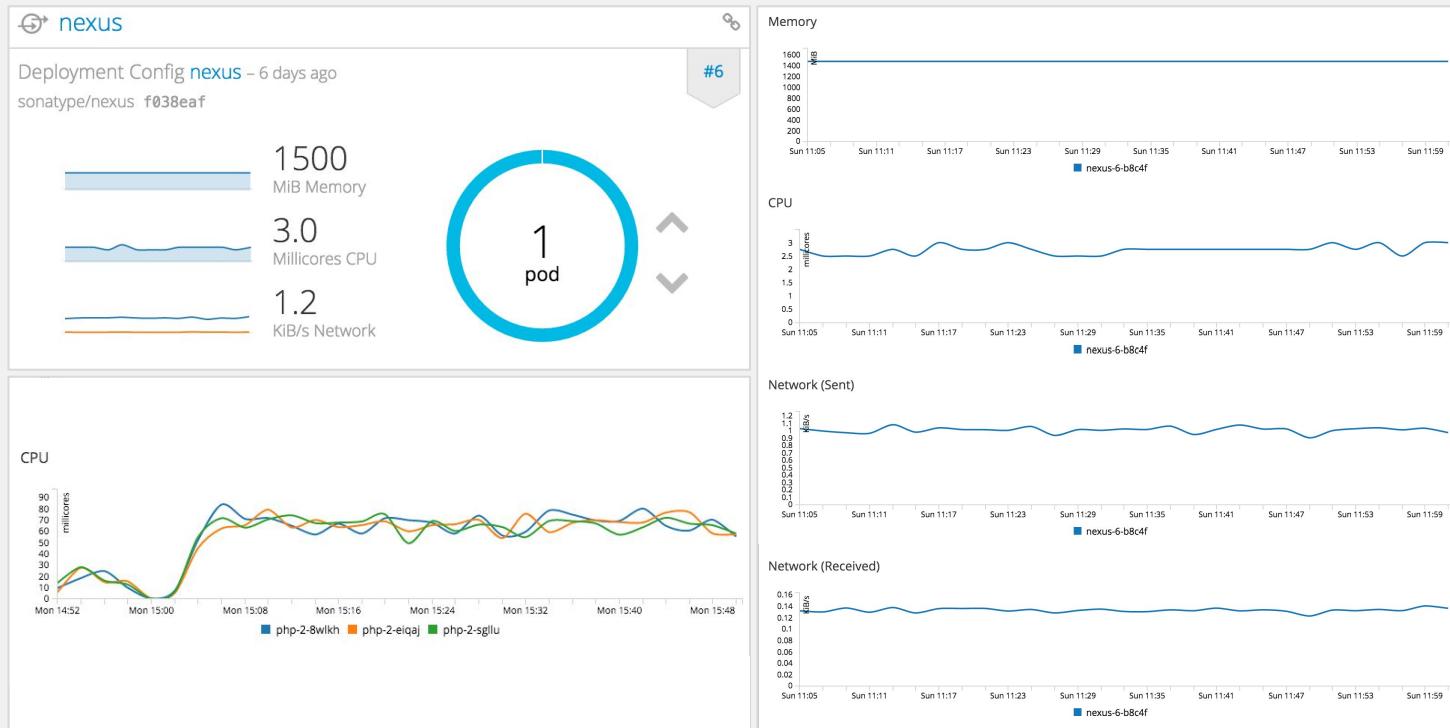
- EFK stack to aggregate logs for hosts and applications
 - **Elasticsearch:** an object store to store all logs
 - **Fluentd:** gathers logs and sends to Elasticsearch.
 - **Kibana:** A web UI for Elasticsearch.
- Access control
 - Cluster administrators can view all logs
 - Users can only view logs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



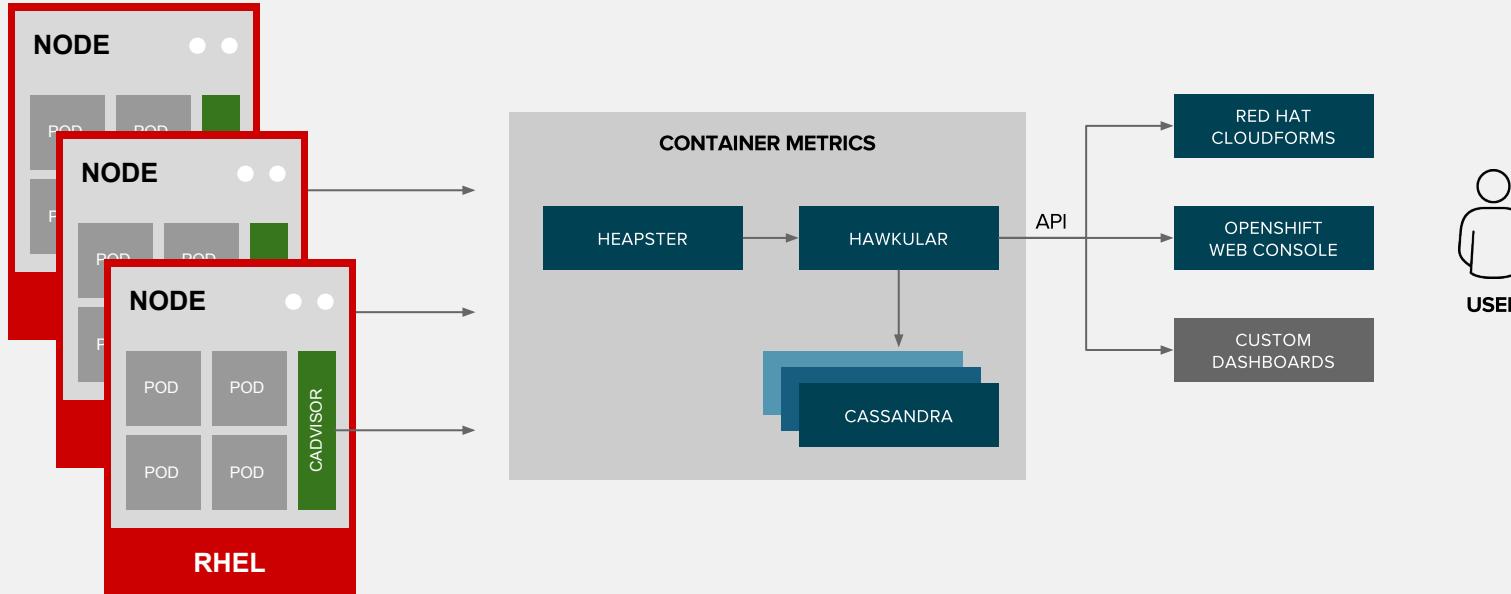
CENTRAL LOG MANAGEMENT WITH EFK



CONTAINER METRICS



CONTAINER METRICS



SECURITY

TEN LAYERS OF CONTAINER SECURITY

Container Host & Multi-tenancy

Federated Clusters

Container Platform

API Management

Network Isolation

Deploying Container

Container Registry

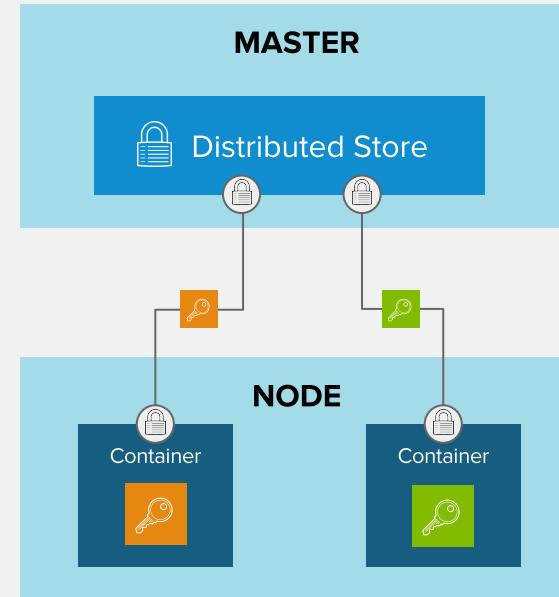
Container Content

Storage

Building Containers

SECRET MANAGEMENT

- Secure mechanism for holding sensitive data e.g.
 - Passwords and credentials
 - SSH Keys
 - Certificates
- Secrets are made available as
 - Environment variables
 - Volume mounts
 - Interaction with external systems
- Encrypted in transit
- Never rest on the nodes



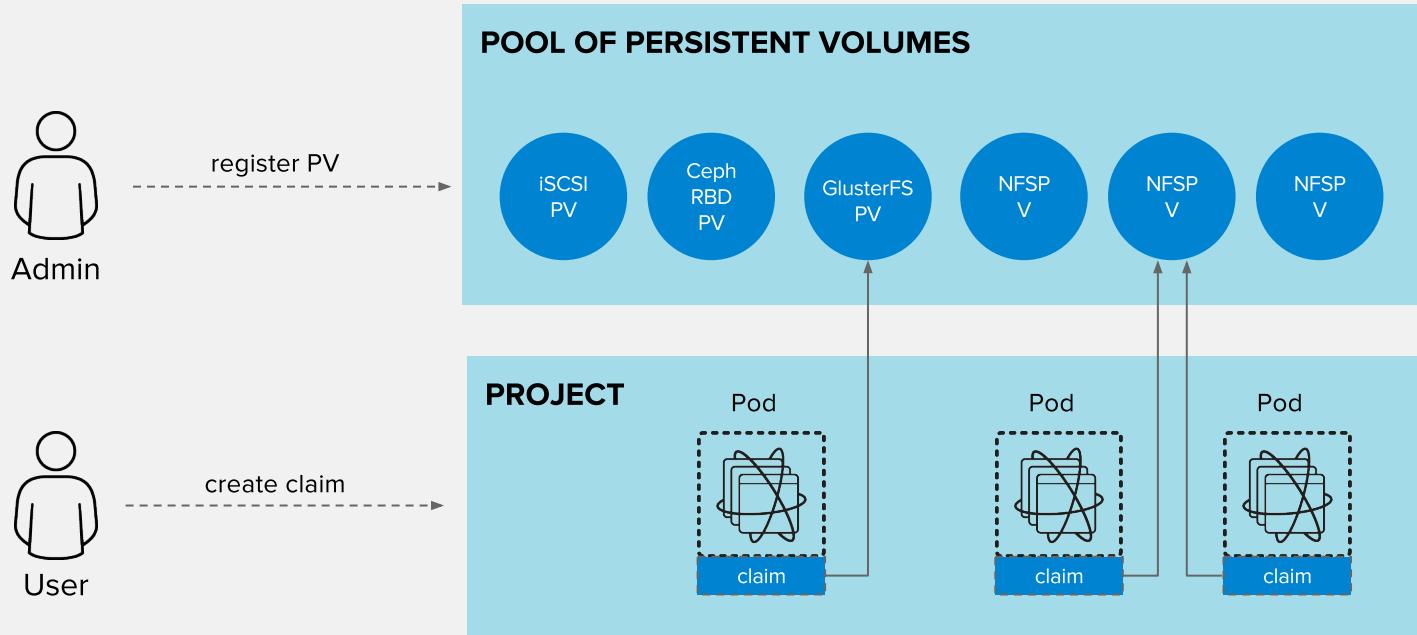
PERSISTENT STORAGE

PERSISTENT STORAGE

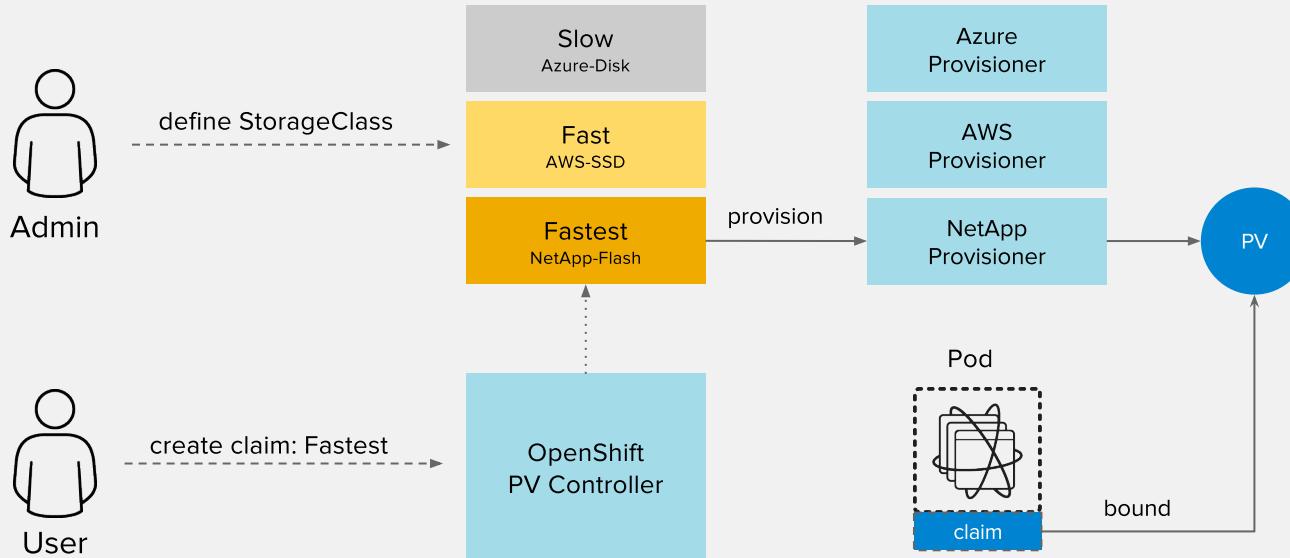
- Persistent Volume (PV) is tied to a piece of network storage
- Provisioned by an administrator (static or dynamically)
- Allows admins to describe storage and users to request storage
- Assigned to pods based on the requested size, access mode, labels and type

NFS	OpenStack Cinder	iSCSI	Azure Disk	AWS EBS	FlexVolume
GlusterFS	Ceph RBD	Fiber Channel	Azure File	GCE Persistent Disk	VMWare vSphere VMDK

PERSISTENT STORAGE

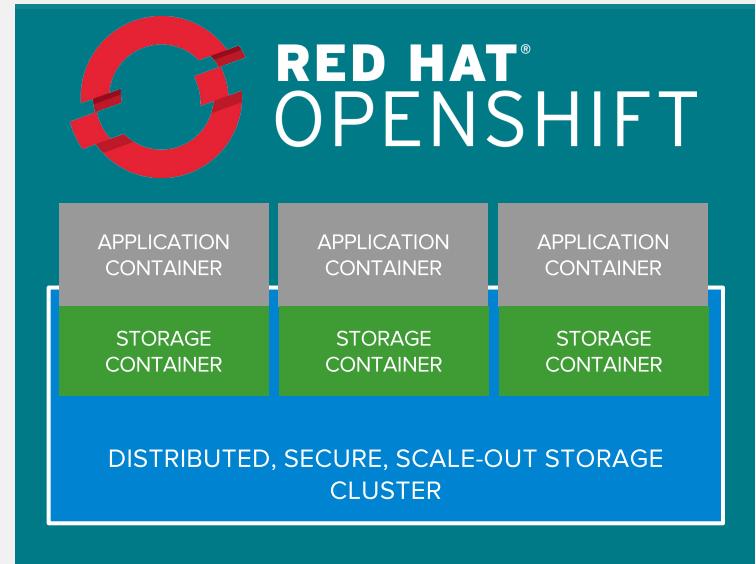


DYNAMIC VOLUME PROVISIONING

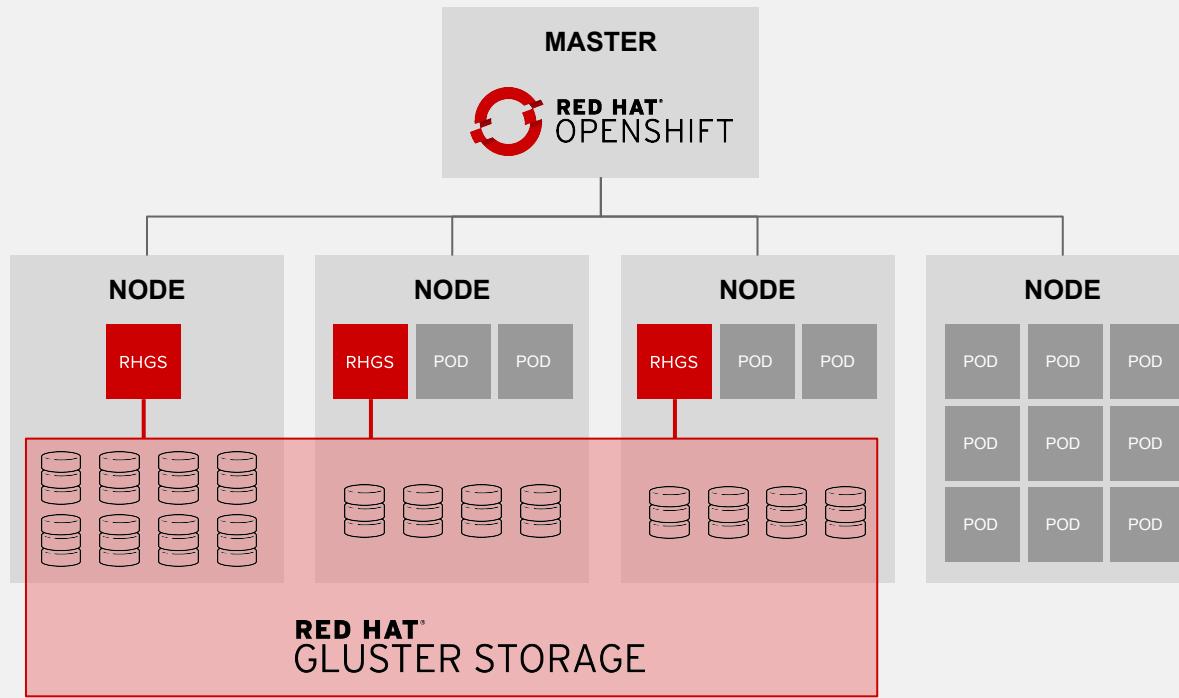


CONTAINER-NATIVE STORAGE

- 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

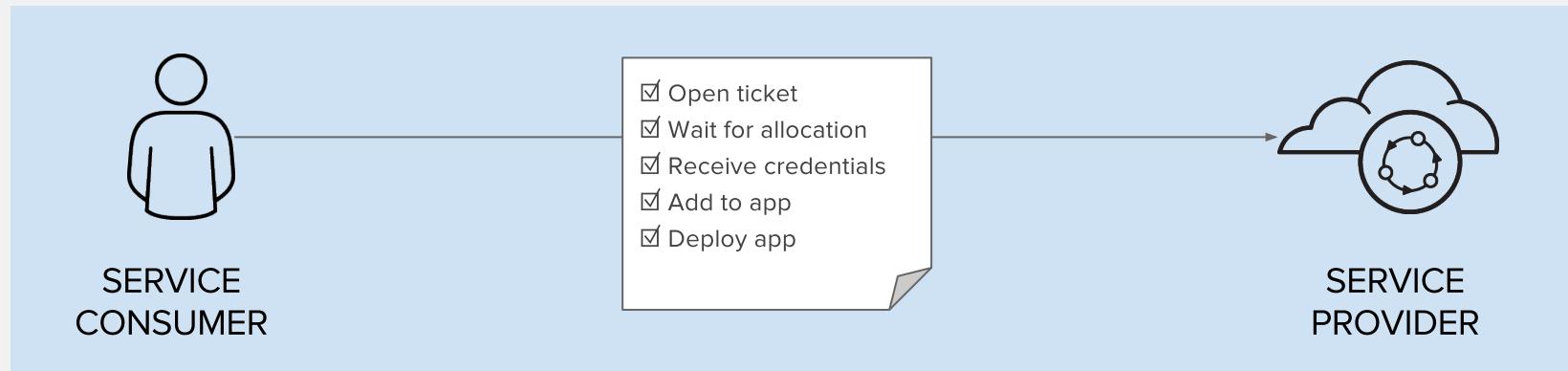


CONTAINER-NATIVE STORAGE



SERVICE BROKER

WHY A SERVICE BROKER?



Manual, Time-consuming and Inconsistent



OPEN SERVICE BROKER API™

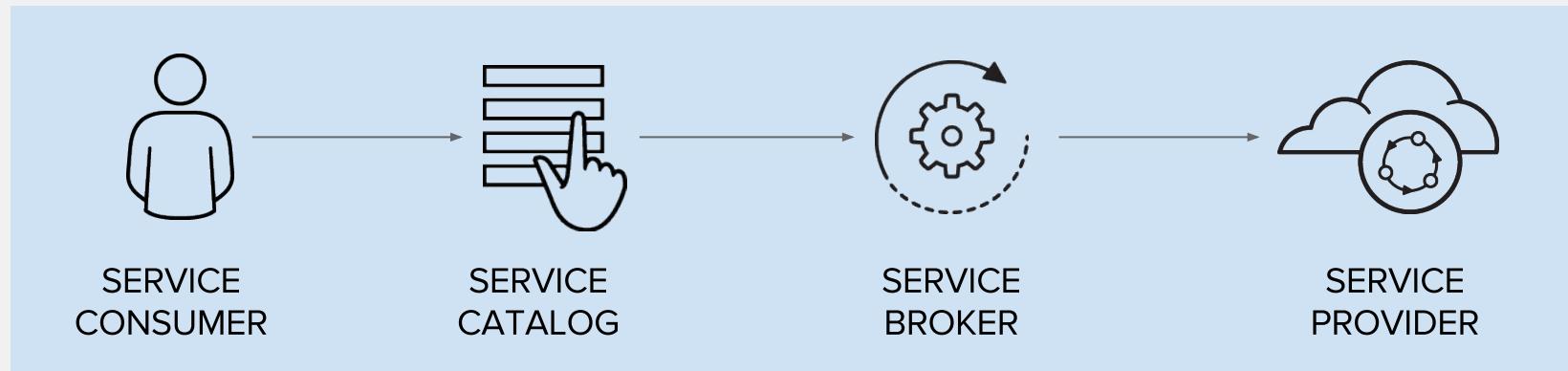
A multi-vendor project to standardize how services are consumed on cloud-native platforms across service providers

FUJITSU Pivotal

IBM redhat

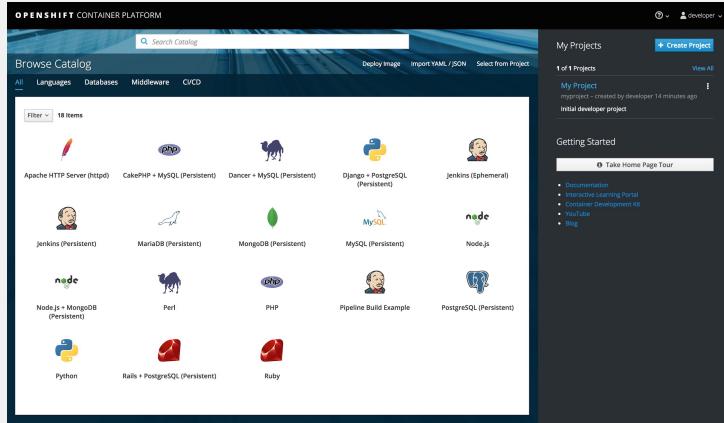
Google SAP

WHAT IS A SERVICE BROKER?

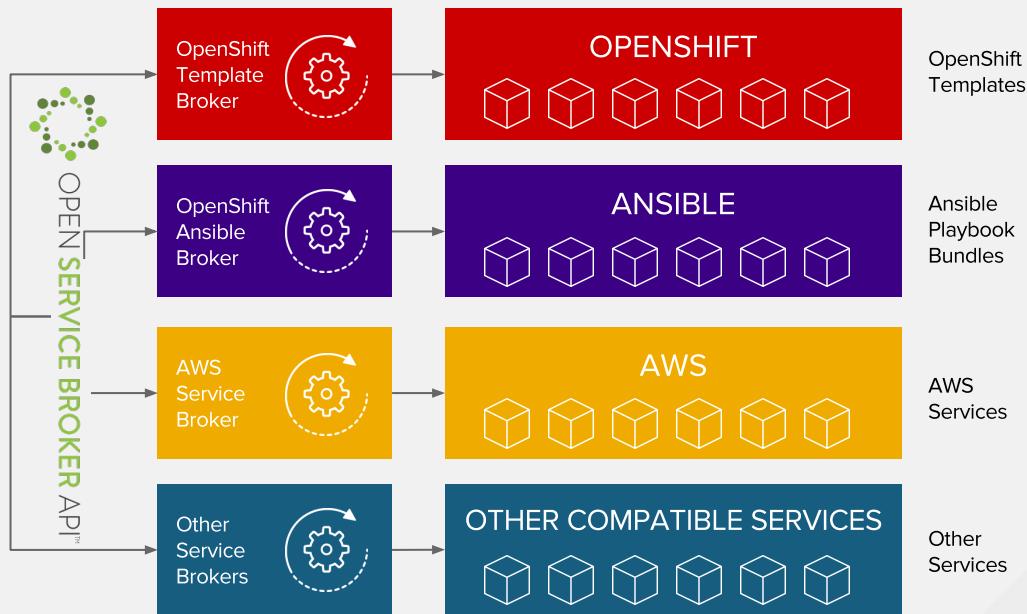


Automated, Standard and Consistent

OPENShift SERVICE CATALOG



OPENShift SERVICE CATALOG



SERVICE BROKER CONCEPTS

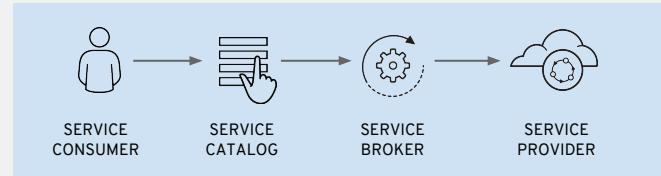
SERVICE: an offering that can be used by an app e.g. database

PLAN: a specific flavor of a service e.g. Gold Tier

SERVICE INSTANCE: an instance of the offering

PROVISION: creating a service instance

BIND: associate a service instance and its credentials to an app



HOW TO ADD A SERVICE BROKER

- Deploy service broker on or off OpenShift
- Register the broker referring to the deployed broker

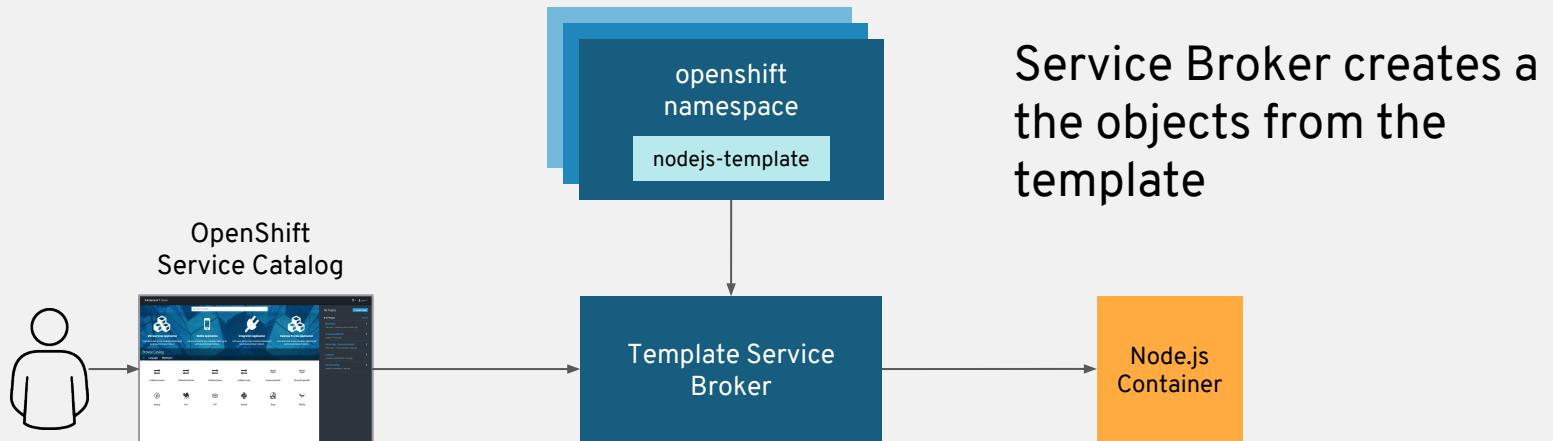
```
apiVersion: servicecatalog.k8s.io/v1alpha1
kind: Broker
metadata:
  name: asb-broker
spec:
  url: https://asb-1338-ansible-service-broker.10.2.2.15.nip.io
```

- Register the broker services by creating ServiceClass resources
(the service broker might automatically perform this step)

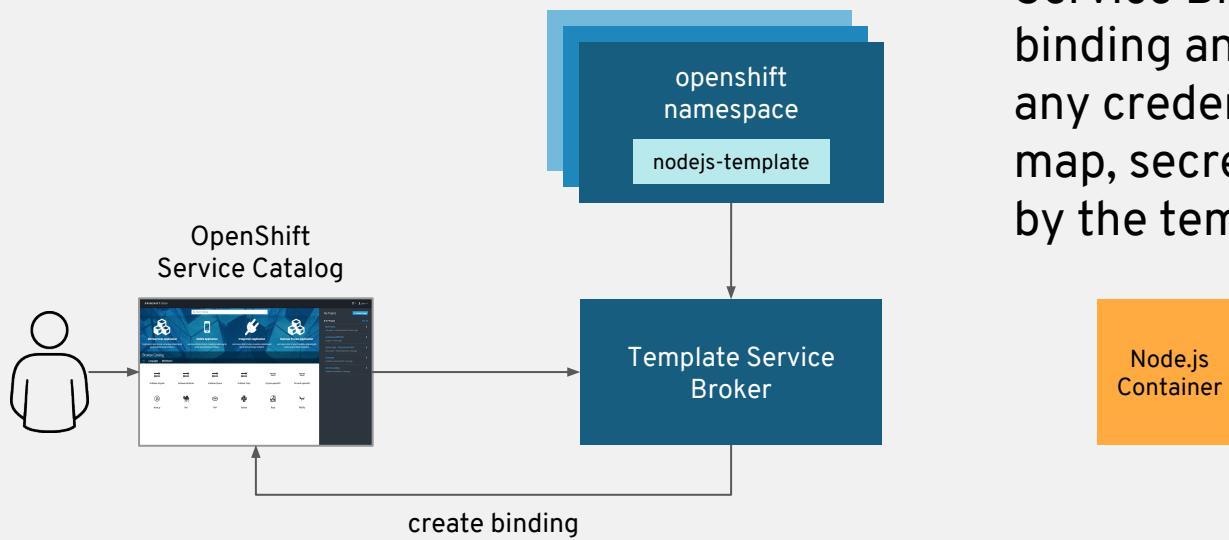
TEMPLATE SERVICE BROKER

- Exposes Templates and Instant Apps in the Service Catalog
- Pulled from openshift namespace by default
- Multiple namespaces can be configured for template discovery

TEMPLATE SERVER BROKER PROVISIONING



TEMPLATE SERVICE BROKER BINDING



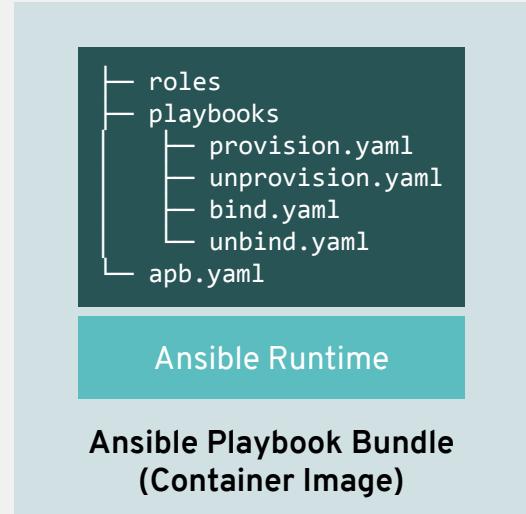
Service Broker creates a binding and secret for any credentials (config map, secret, etc) created by the template

OPENShift ANSIBLE BROKER

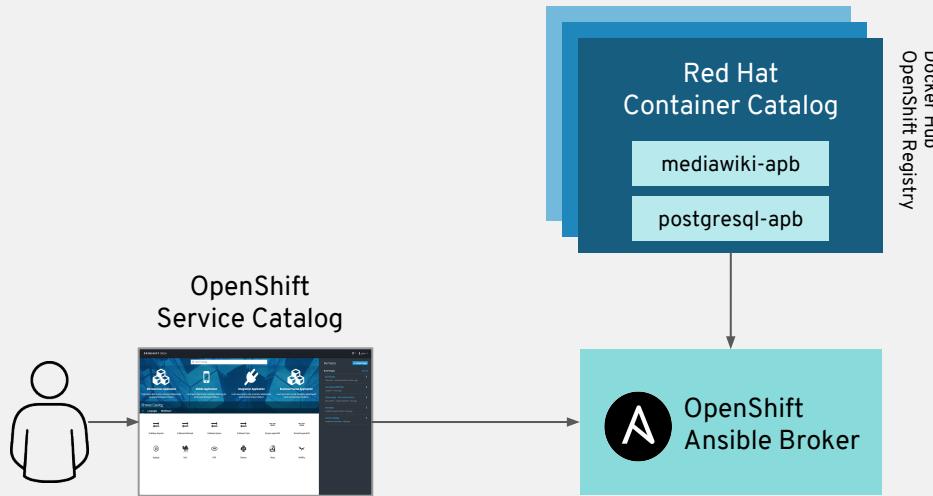
- Use Ansible on OpenShift
 - Deploy containerized applications
 - Provision external services (e.g. Oracle database)
 - Provision cloud services (e.g. AWS RDS)
 - Orchestrate multi-service solutions
 - Conditional logic for control on deployments (e.g. database is initialized)
- Leverage existing Ansible playbooks
- Anything you can do with Ansible, you can do with OAB

ANSIBLE PLAYBOOK BUNDLES (APB)

- Lightweight application definition
- Packaged as a container image
- Embedded Ansible runtime
- Metadata for parameters
- Named playbooks for actions
- Leverage existing Ansible playbooks
- Registry is queried to discover APBs

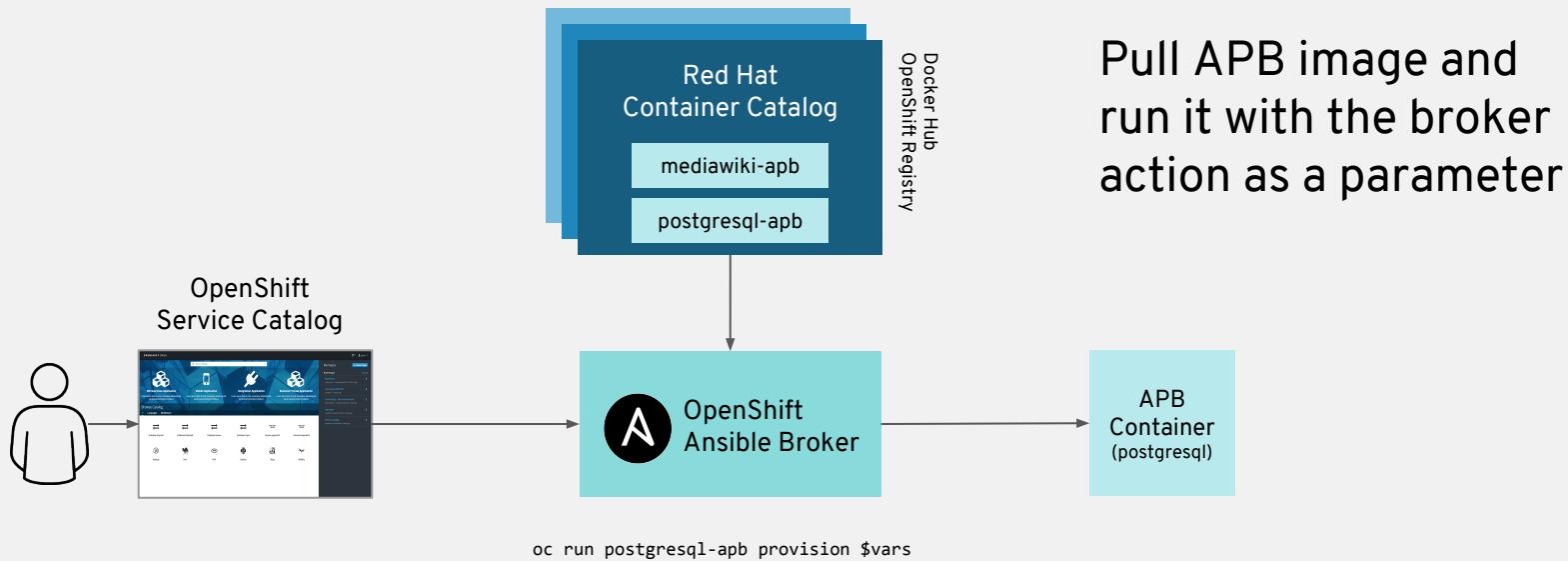


OPENShift ANSIBLE BROKER PROVISIONING



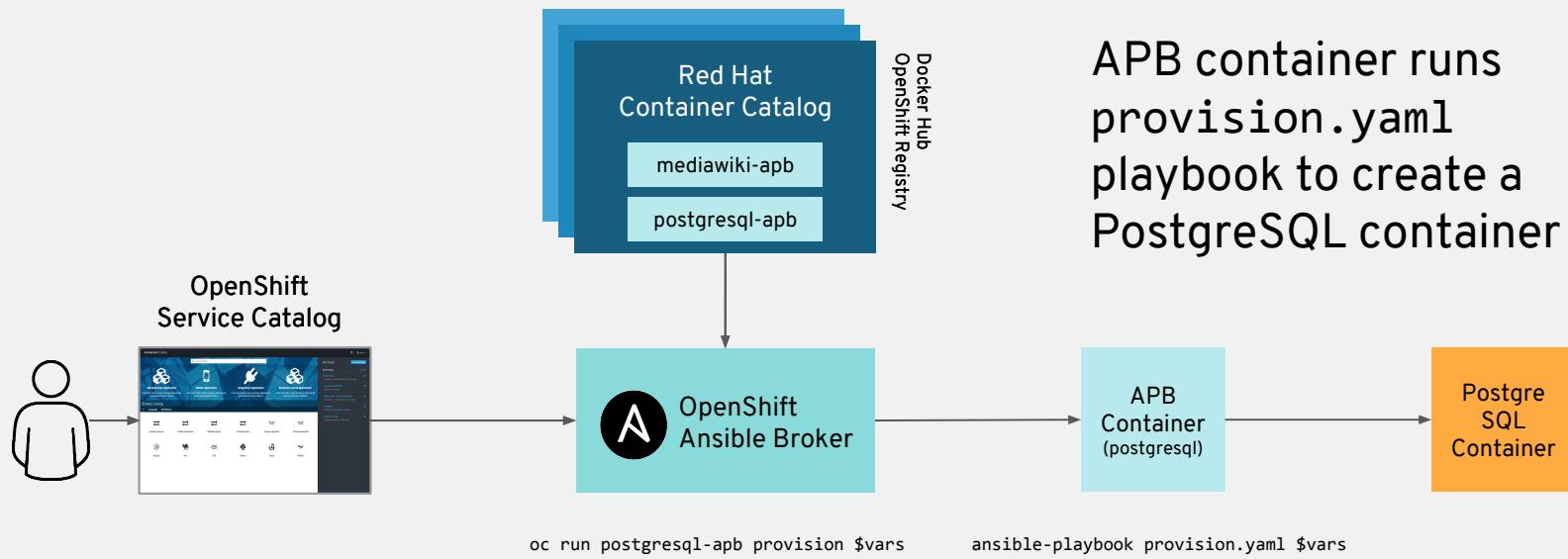
Discover and list APBs from the configured image registries

OPENShift ANSIBLE BROKER PROVISIONING

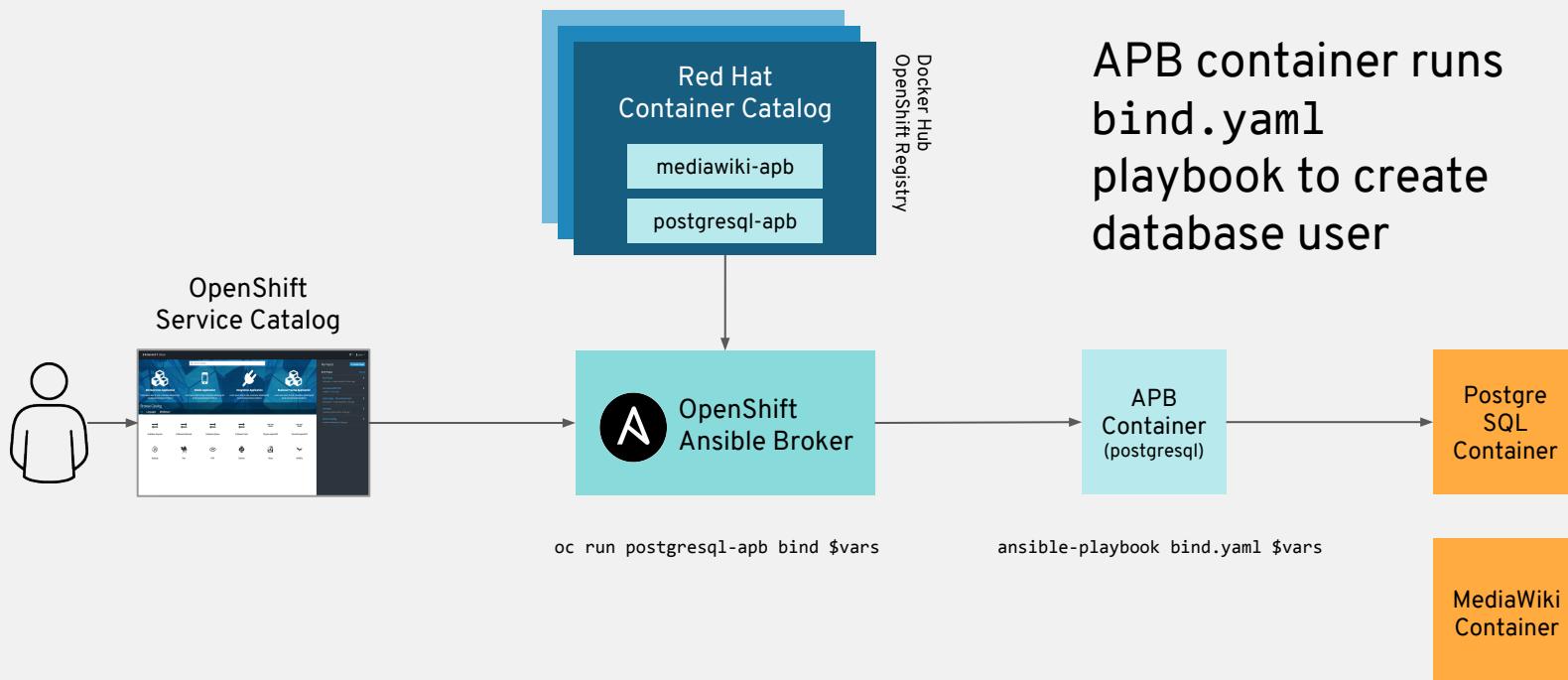


Pull APB image and run it with the broker action as a parameter

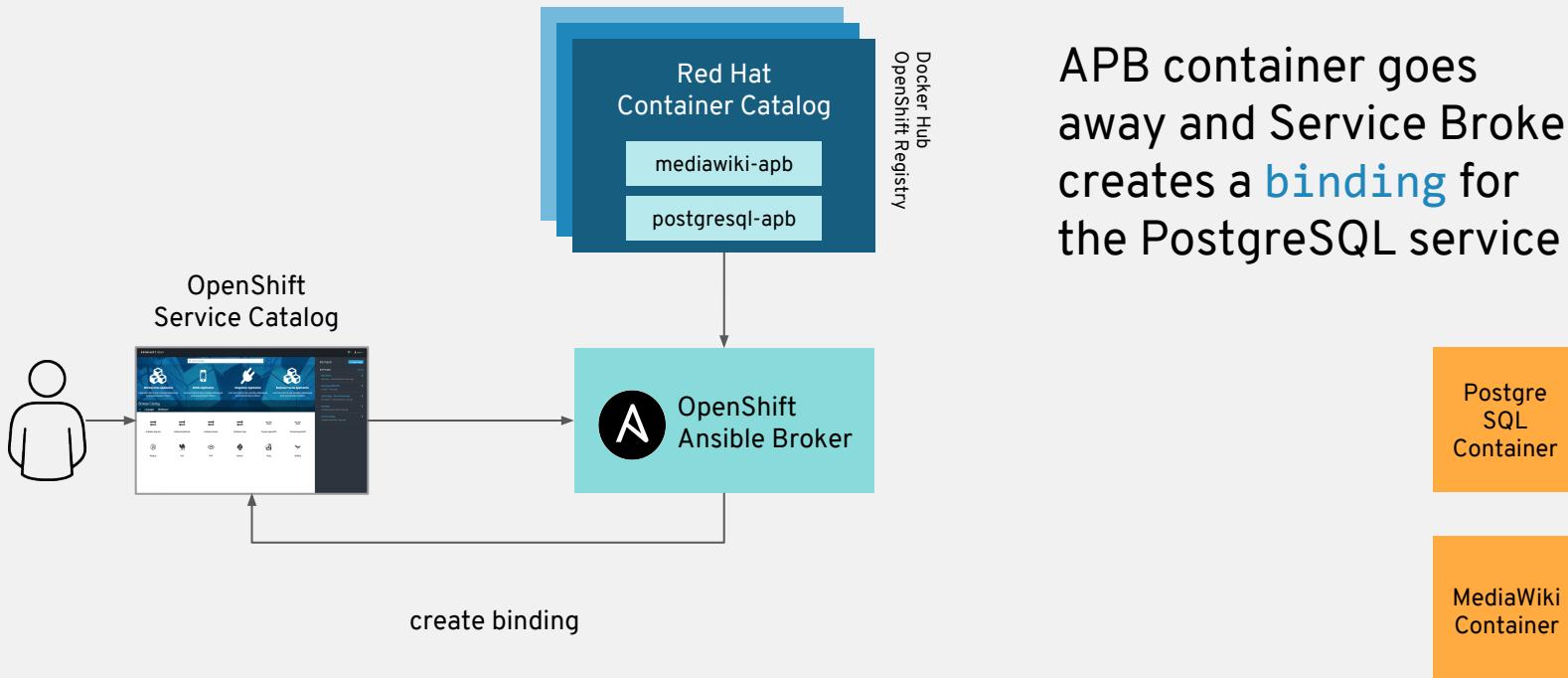
OPENShift ANSIBLE BROKER PROVISIONING



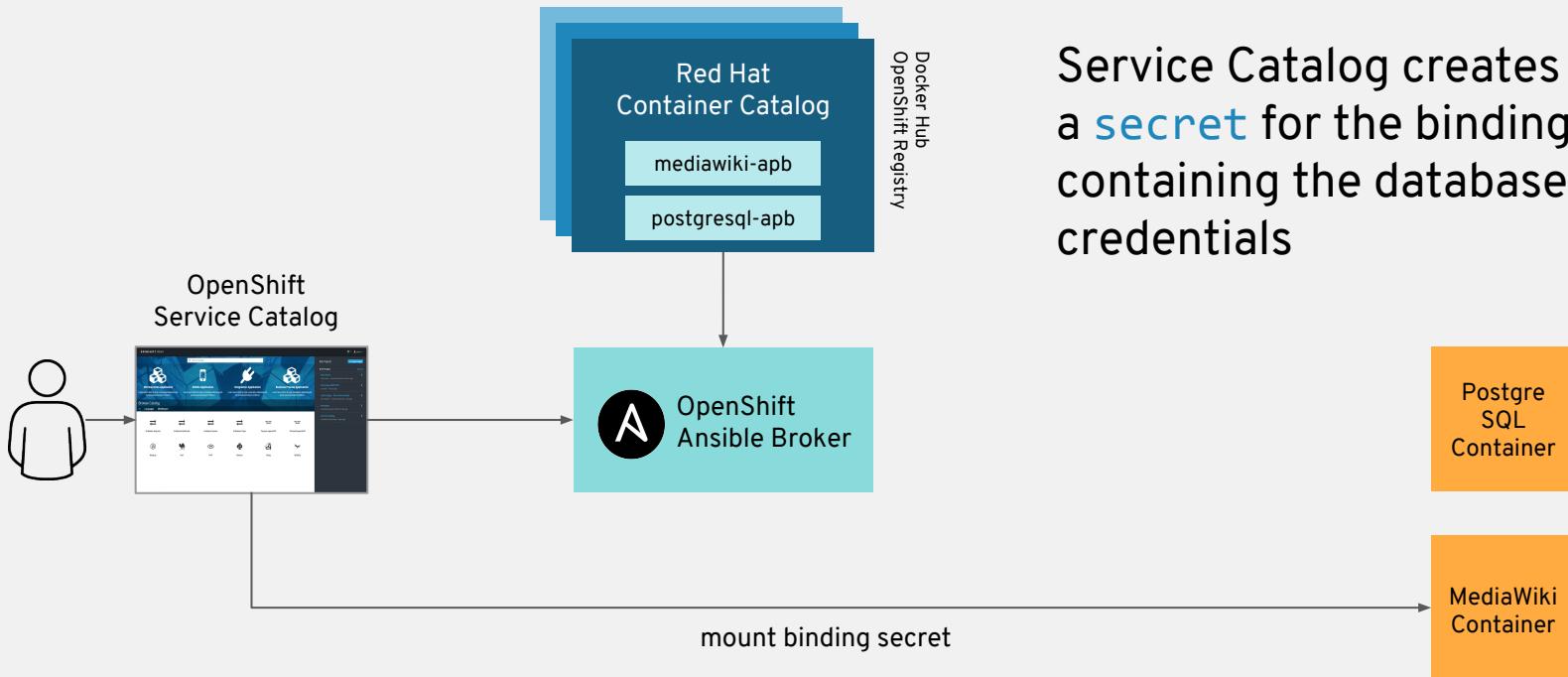
OPENShift ANSIBLE BROKER BINDING



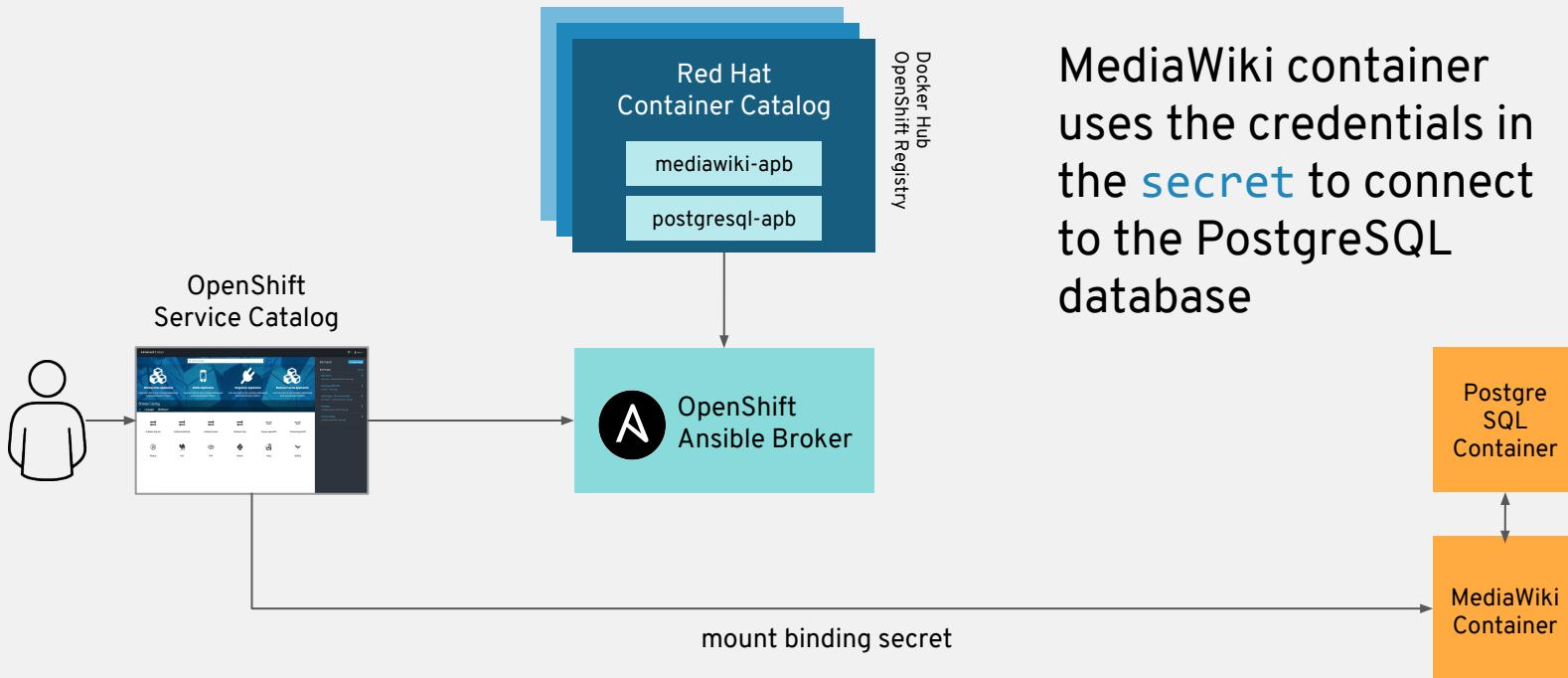
OPENShift ANSIBLE BROKER BINDING



OPENShift ANSIBLE BROKER BINDING



OPENShift ANSIBLE BROKER BINDING

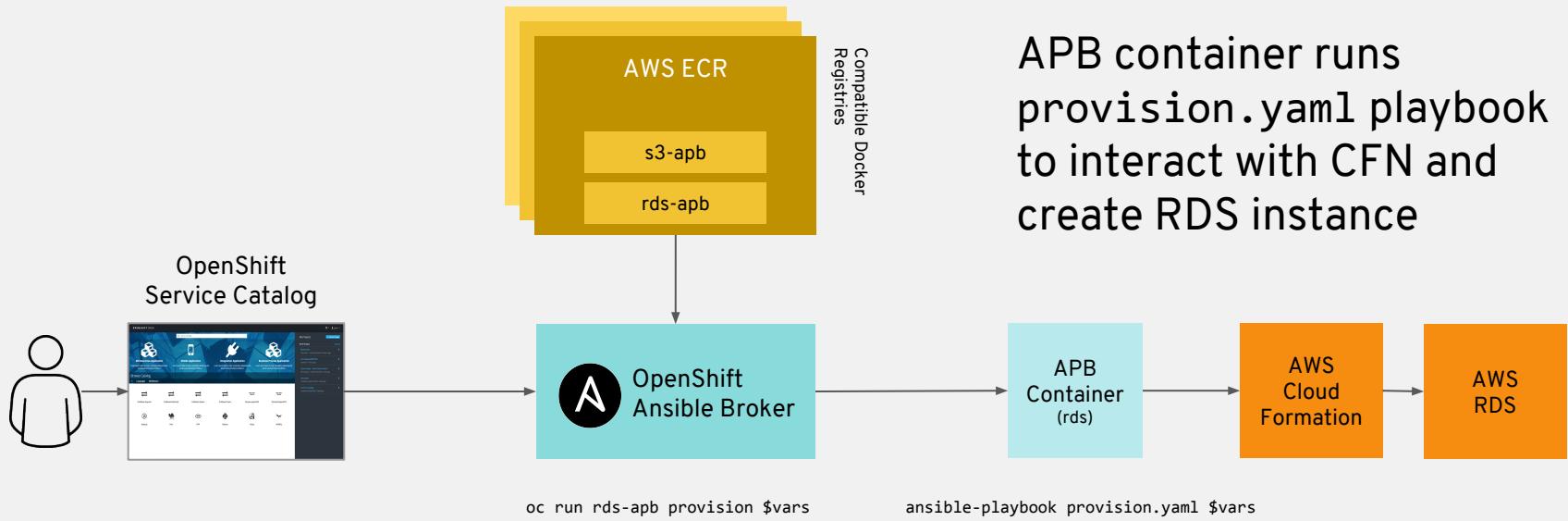


AWS SERVICE BROKER

- Targets Top 10 AWS Services
- Uses Ansible Playbook Bundles
- Available in OpenShift 3.7

SQS	SNS	DynamoDB	Redshift SES	S3
RDS	EMR	AWS Batch	ElastiCache	Route 53

AWS PROVISIONING



OPERATIONAL MANAGEMENT

TOP CHALLENGES OF RUNNING CONTAINERS AT SCALE



OPERATIONAL
EFFICIENCY



SERVICE
HEALTH



SECURITY
& COMPLIANCE



FINANCIAL
MANAGEMENT



RED HAT[®] CLOUDFORMS

Operational Management Across the Stack

- Real-time discovery
- Visualize relationships
- Monitoring and alerts
- Vulnerability scanning
- Security compliance
- Workflow and policy
- Automation
- Chargeback

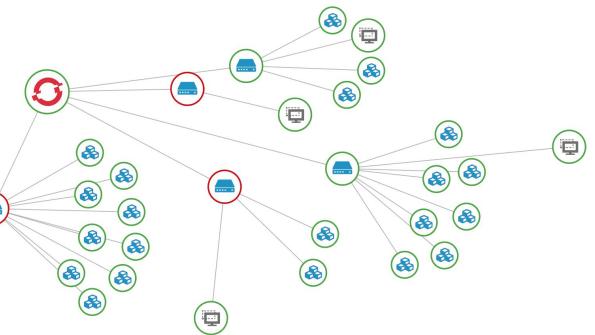
OPERATIONAL EFFICIENCY

- CloudForms continuously discovers your infrastructure in near real time.
- CloudForms discovers and visualizes relationships between infra components
- CloudForms cross references inventory across technologies.
- CloudForms offers custom automation via control policy or UI extensions



OPERATIONAL EFFICIENCY

Replicators Pods Containers Services Routes Nodes VMs Hosts



Projects by Number of Pods Widget

Project Name	Number of Pods
demo-project	12
demo-project	7
default	3
openshift-infra	3
management-infra	3
default	2
cloudforms	1
openshift-infra	0
openshift	0
management-infra	0

Relationships

Containers Provider	OpenShift Container Platform
Project	cicd
Container Services	1
Replicator	gogs-1
Containers	1
Node	ocp-node-2.lab.example.com
Underlying Instance	ocp-node-2.lab.example.com
Container Images	1

1 Providers 6 Nodes 26 Containers 3 Registries 14 Projects

0 1 26 346 14

Aggregated Node Utilization

CPU 83 Available of 84 Cores Memory 95 Available of 152 GB

1 Cores Used 57 GB Used

1301 Kbps

Network Utilization Trend

New Image Usage Trend

Pod Creation and Deletion Trends

Last 30 Days Last 30 Days Last 30 Days Last 30 Days

Images

Node Utilization

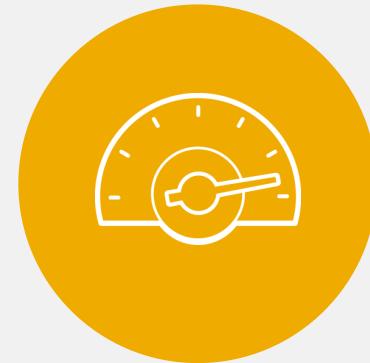
CPU Memory

> 90% 80-90% 70-80% < 70%

Created Deleted

SERVICE HEALTH

- CloudForms monitors resource consumption and shows trends
- CloudForms alerts on performance thresholds or other events
- CloudForms offers right-sizing recommendations
- CloudForms enforces configuration and tracks it over time.



SERVICE HEALTH



Normal Operating Ranges (up to 30 days' data)

	Max	High	Average	Low
CPU	745.90 MHz	705.74 MHz	663.99 MHz	622.23 MHz
CPU Usage	100.00%	15.36%	14.10%	12.84%
Memory	7.7 GB	7.57 GB	7.37 GB	7.18 GB
Memory Usage	65.00%	63.46%	61.78%	60.11%

Right-Sizing (Conservative - derived from Absolute Maximum)

	Current	Recommended	% Savings	Savings
Processors	4	5	-25.0%	-1
Memory	12288 MB	7988 MB	35.0%	4300 MB

Right-Sizing (Moderate - derived from High NORM)

	Current	Recommended	% Savings	Savings
Processors	4	1	75.0%	3
Memory	12288 MB	7800 MB	36.5%	4488 MB

Right-Sizing (Aggressive - derived from Average NORM)

	Current	Recommended	% Savings	Savings
Processors	4	1	75.0%	3
Memory	12288 MB	7596 MB	38.2%	4692 MB

SECURITY & COMPLIANCE

- CloudForms finds and marks nodes non-compliant with policy.
- CloudForms allows reporting on container provenance.
- CloudForms scans container images using OpenSCAP.
- CloudForms tracks genealogy between images and containers.



SECURITY & COMPLIANCE

Compliance		
Status	 Non-Compliant as of 5 Days Ago	
History	 Available	
Configuration		
Packages	 528	
OpenSCAP Results	 431	
OpenSCAP HTML	Available	
Last scan	Tue, 28 Mar 2017 11:05:54 +0000	
OpenSCAP Failed Rules Summary		
Medium	1	
High	3	
 Name		
Result		
Severity		
 xccdf_com.redhat.rhsa_rule_oval_com.redhat.rhsa-def-20170386	 Fail	 High
 xccdf_com.redhat.rhsa_rule_oval_com.redhat.rhsa-def-20170372	 Fail	 High
 xccdf_com.redhat.rhsa_rule_oval_com.redhat.rhsa-def-20170294	 Fail	 High
 xccdf_com.redhat.rhsa_rule_oval_com.redhat.rhsa-def-20170286	 Fail	 Medium
 xccdf_com.redhat.rhsa_rule_oval_com.redhat.rhsa-def-20140685	 Pass	 High
 xccdf_com.redhat.rhsa_rule_oval_com.redhat.rhsa-def-20140686	 Pass	 High
 xccdf_com.redhat.rhsa_rule_oval_com.redhat.rhsa-def-20140679	 Pass	 High
 xccdf_com.redhat.rhsa_rule_oval_com.redhat.rhsa-def-20140703	 Pass	 Medium
 xccdf_com.redhat.rhsa_rule_oval_com.redhat.rhsa-def-20140684	 Pass	 High
 xccdf_com.redhat.rhsa_rule_oval_com.redhat.rhsa-def-20140704	 Pass	 Medium

Compliance and Scoring

The target system did not satisfy the conditions of 4 rules! Please review rule results and consider applying remediation.

Rule results

427 passed

4

Severity of failed rules

1 medium 3 high

Score

Scoring system	Score	Maximum	Percent
urn:xccdf:scoring:default	99.071922	100.000000	99.07%

Rule Overview

<input type="checkbox"/> pass	<input checked="" type="checkbox"/> fail	<input type="checkbox"/> notchecked
<input type="checkbox"/> fixed	<input type="checkbox"/> error	<input type="checkbox"/> notselected
<input type="checkbox"/> informational	<input type="checkbox"/> unknown	<input type="checkbox"/> notapplicable

Search through XCCDF rules

Search

Group rules by: Default

Title	Severity	Result
▼ Automatically generated XCCDF from OVAL file: com.redhat.rhsa-RHEL7.xml 4x fail		
RHSA-2017:0286: openssl security update (Moderate)	medium	 fail
RHSA-2017:0294: kernel security update (Important)	high	 fail
RHSA-2017:0372: kernel-aarch64 security and bug fix update (Important)	high	 fail
RHSA-2017:0386: kernel security, bug fix, and enhancement update (Important)	high	 fail

FINANCIAL MANAGEMENT

- Define cost models for infrastructure and understand your cost.
- Rate schedules per platform and per tenant with multi-tiered and multi-currency support
- CloudForms shows top users for CPU, memory, as well as cost.
- Chargeback/showback to projects based on container utilization.



FINANCIAL MANAGEMENT

Top CPU Consumers (Last Hour)			
VM Name	CPU Usage	Allocated vCPUs	VM Vendor
overcloud1-telus	21.4%	8	vmware
manageiq-euwe-2	18.3%	4	redhat
manageiq-euwe-3	14.0%	4	redhat
Lenovo XClarity Administrator - Do not delete	9.1%	2	vmware
vcenter6	8.0%	4	vmware

Updated December 21, 2016 20:17 | Next January 11, 2017 23:45

Top Memory Consumers (last hour)		
VM Name	Memory Usage	Allocated Memory
CF41_DB	100.0%	16 GB
CF42_UI2	97.7%	8 GB
CF42_UI1	97.7%	8 GB
manageiq-euwe-3	97.6%	8 GB
CF42_google1	97.0%	8 GB

Updated January 11, 2017 23:49 | Next January 12, 2017 00:05

Saved Report "ChargeBack by Project - Tue, 18 Apr 2017 17:59:28 +0000"

Date Range	Project Name	Project Uid	Cpu Cores Used Cost	Memory Used Cost	Total Cost
04/17/2017	cicd	b8f35aeee974-11e6-89d9-fa163ec3f31d	\$24.00	\$30.33	\$66.34
04/17/2017	default	4c767b2b-df4d-11e6-8850-fa163ec3f31d	\$24.00	\$4.90	\$40.90
04/17/2017	ifixied	acc6113d-ed77-11e6-8c6a-fa163ec3f31d	\$24.00	\$28.77	\$64.77
04/17/2017	jritenour-demo	47ee9d2a-efae-11e6-8c6a-fa163ec3f31d	\$24.00	\$28.80	\$64.80
04/17/2017	mlbparks	4666e252-e296-11e6-8a49-fa163ec3f31d	\$24.00	\$406.96	\$442.96
04/17/2017	openshift-infra	4e37af93-df4d-11e6-8850-fa163ec3f31d	\$24.06	\$992.75	\$1,290.78
04/17/2017	stage	b771432a-e974-11e6-89d9-fa163ec3f31d	\$24.00	\$491.89	\$527.89
Totals:					
			\$168.07	\$1,984.40	\$2,498.43
All Rows					
			\$168.07	\$1,984.40	\$2,498.43

REFERENCE ARCHITECTURES

REFERENCE ARCHITECTURES

[OpenShift on VMware vCenter](#)

[OpenShift on Red Hat OpenStack Platform](#)

[OpenShift on Amazon Web Services](#)

[OpenShift on Google Cloud Platform](#)

[OpenShift on Microsoft Azure](#)

[OpenShift on Red Hat Virtualization](#)

[OpenShift on HPE Servers with Ansible Tower](#)

[OpenShift on VMware vCenter 6 with Gluster](#)

[Deploying an OpenShift Distributed Architecture](#)

[OpenShift Architecture and Deployment Guide](#)

[OpenShift Scaling, Performance, and Capacity Planning](#)

[Application Release Strategies with OpenShift](#)

[Building Polyglot Microservices on OpenShift](#)

[Building JBoss EAP 6 Microservices on OpenShift](#)

[Building JBoss EAP 7 Microservices on OpenShift](#)

[Business Process Management with JBoss BPM Server on OpenShift](#)

[Build and Deployment of Java Applications on OpenShift](#)

[Building Microservices on OpenShift with Fuse Integration...](#)

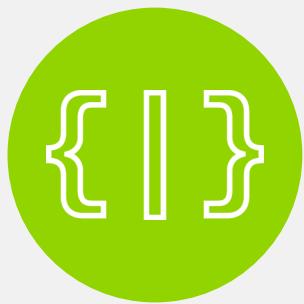
[JFrog Artifactory on OpenShift Container Platform](#)

[Spring Boot Microservices on Red Hat OpenShift](#)

[API Management with Red Hat 3scale on OpenShift](#)

BUILD AND DEPLOY CONTAINER IMAGES

BUILD AND DEPLOY CONTAINER IMAGES



DEPLOY YOUR
SOURCE CODE

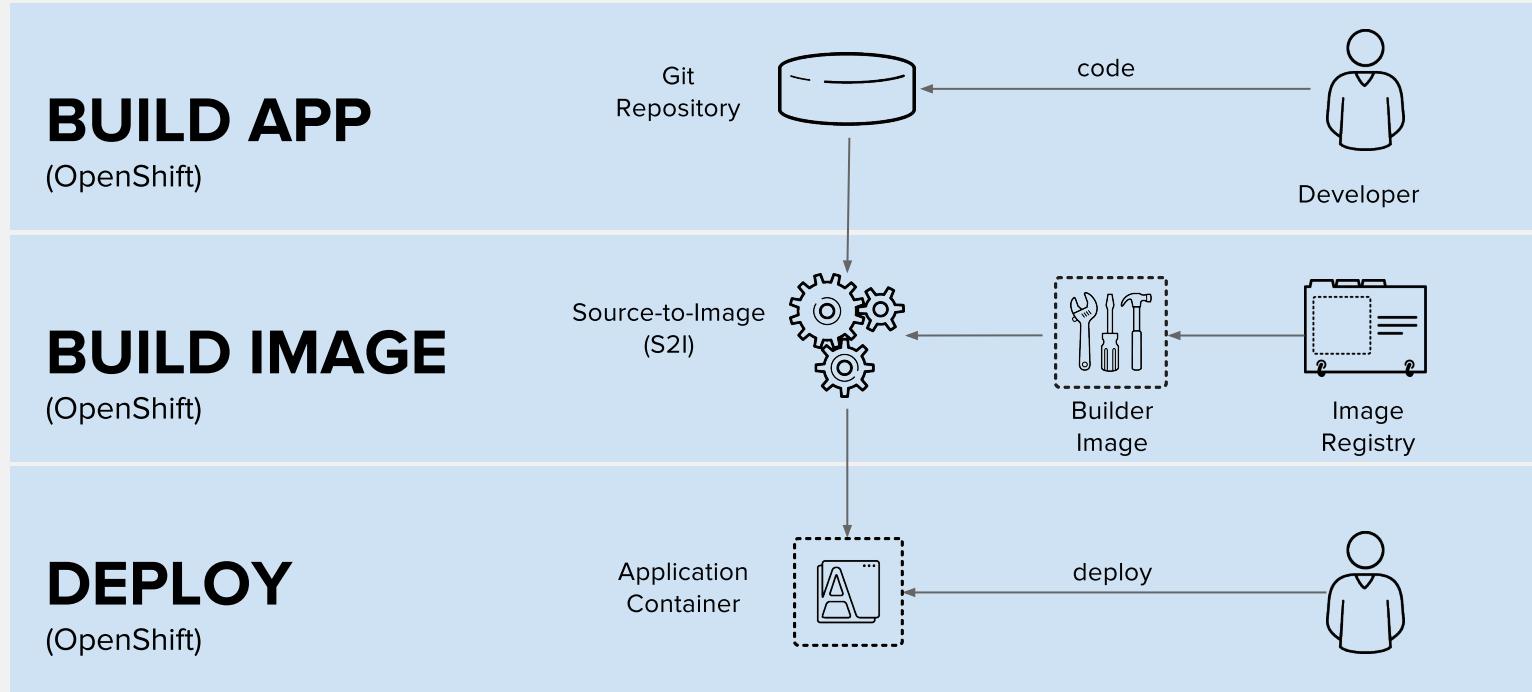


DEPLOY YOUR
APP BINARY

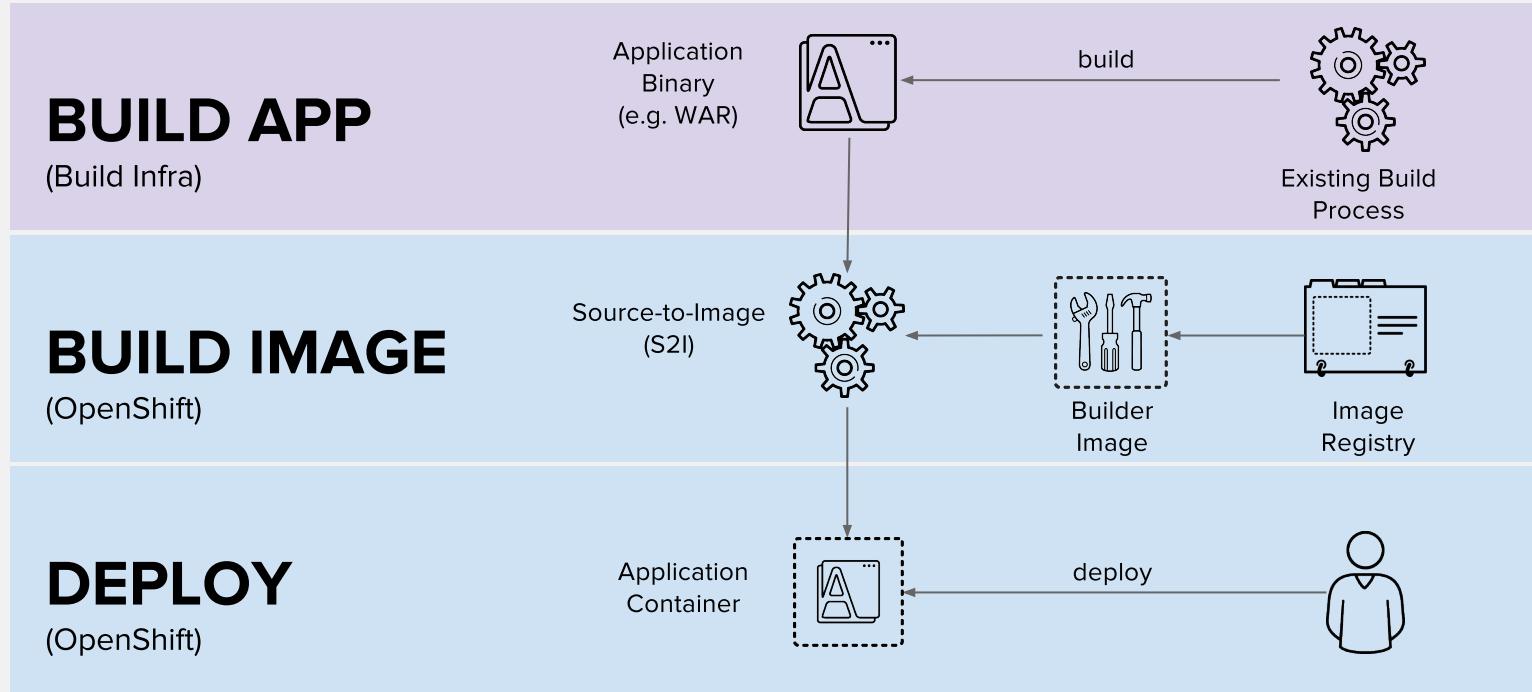


DEPLOY YOUR
CONTAINER IMAGE

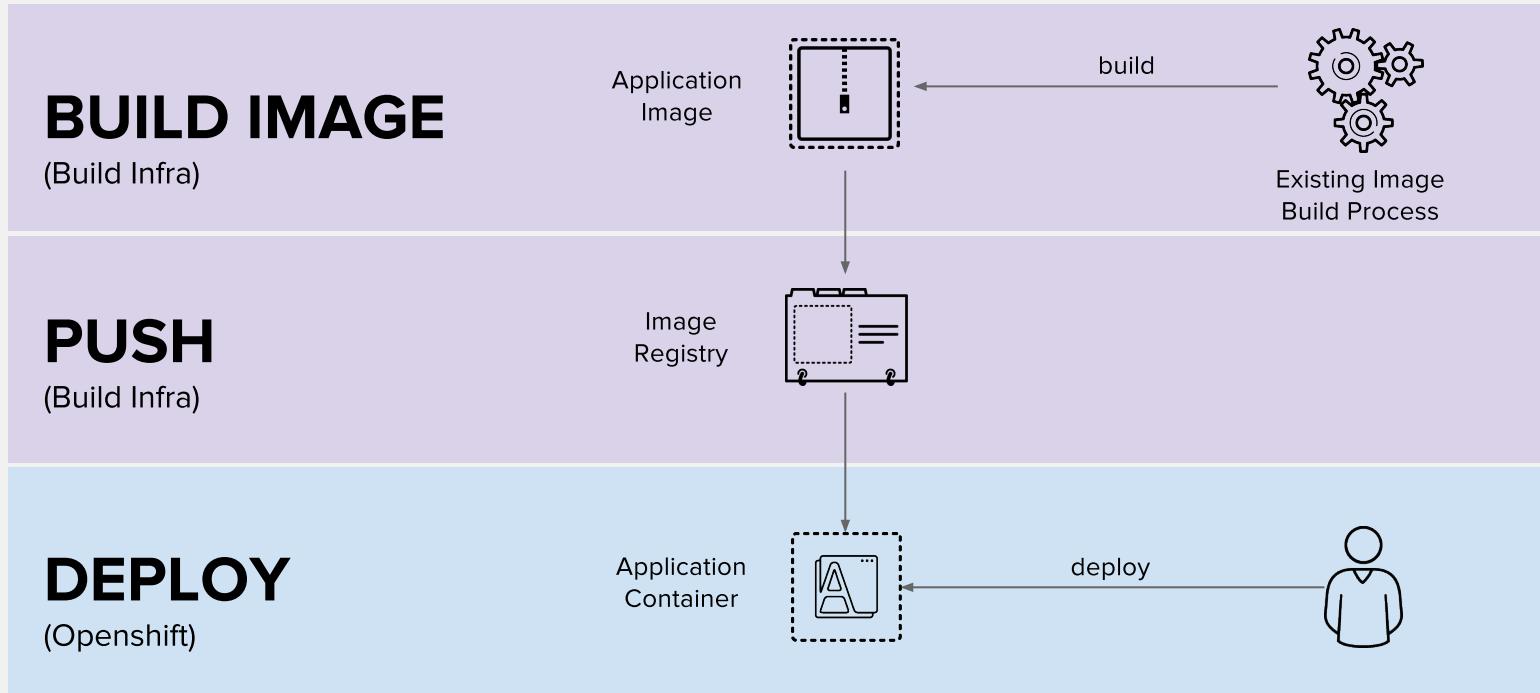
DEPLOY SOURCE CODE WITH SOURCE-TO-IMAGE (S2I)



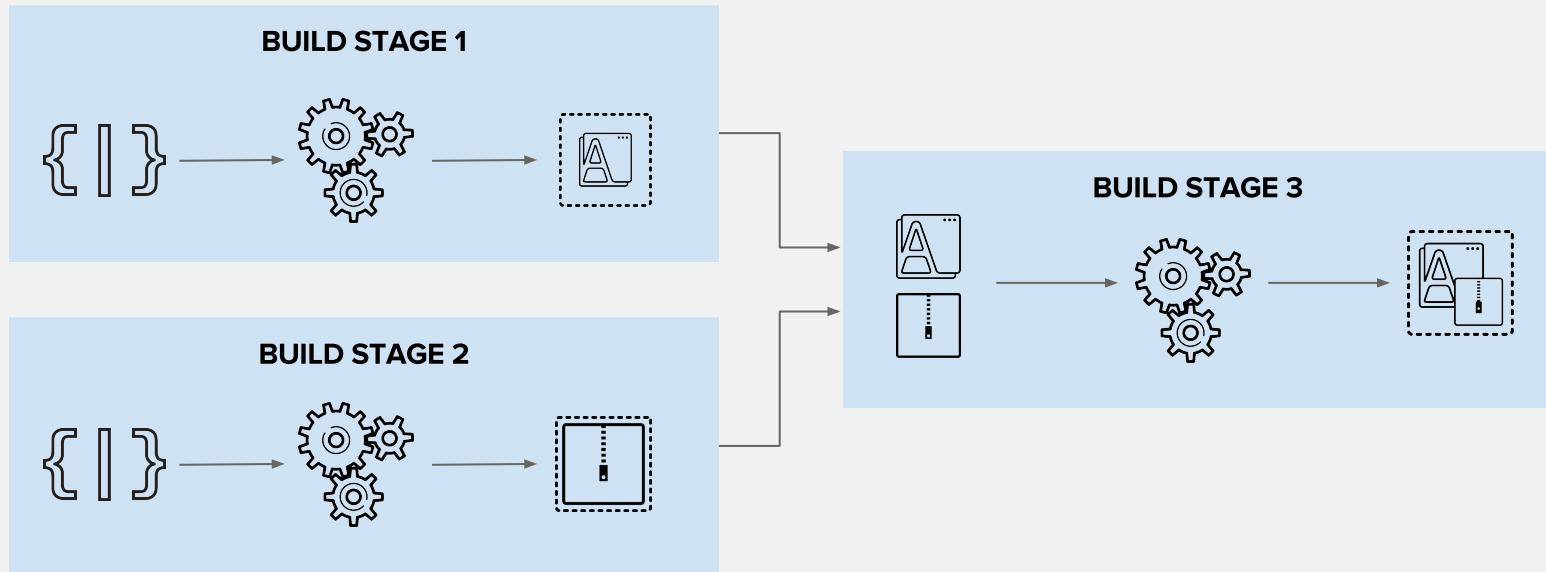
DEPLOY APP BINARY WITH SOURCE-TO-IMAGE (S2I)



DEPLOY DOCKER IMAGE

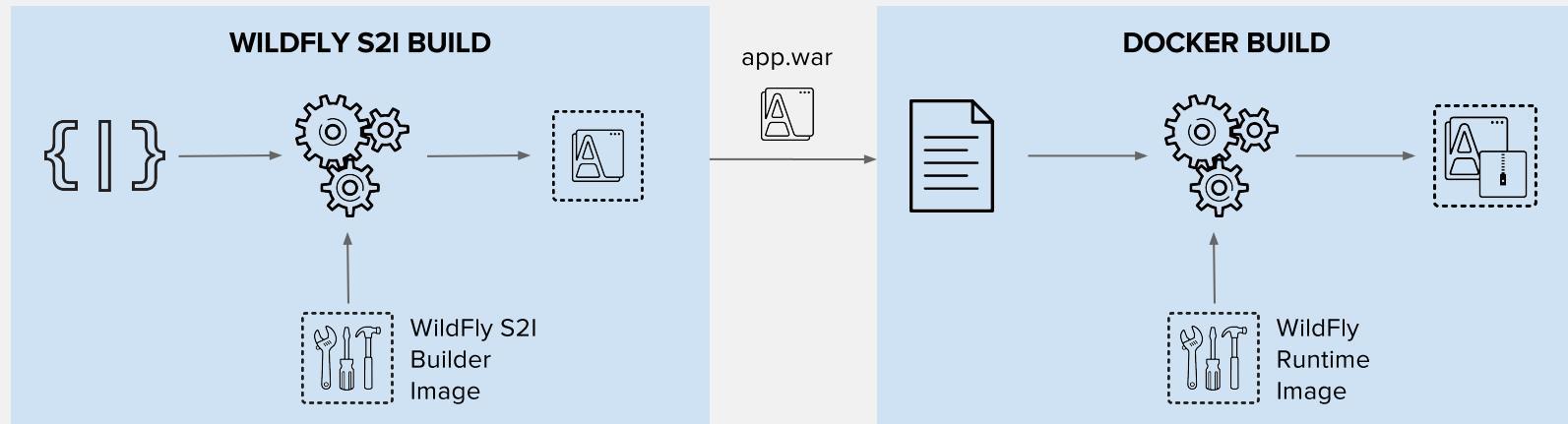


BUILD IMAGES IN MULTIPLE STAGES



EXAMPLE: USE ANY RUNTIME IMAGE WITH SOURCE-TO-IMAGE BUILDS

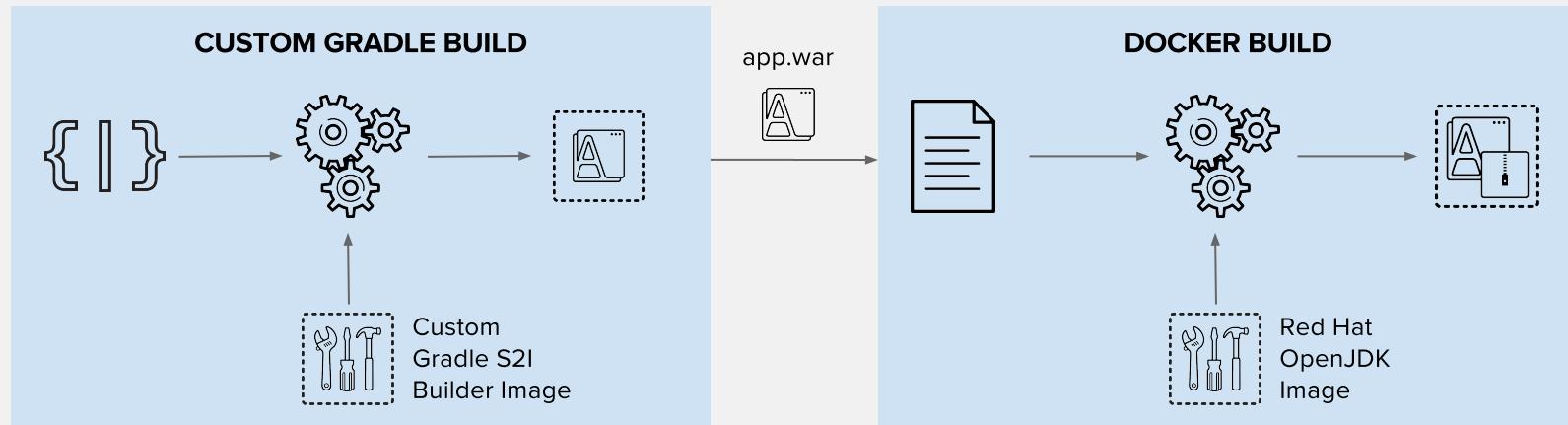
Use Source-to-Image to build app binaries and deploy on lean vanilla runtimes



read more on <https://blog.openshift.com/chaining-builds/>

EXAMPLE: USE ANY BUILD TOOL WITH OFFICIAL RUNTIME IMAGES

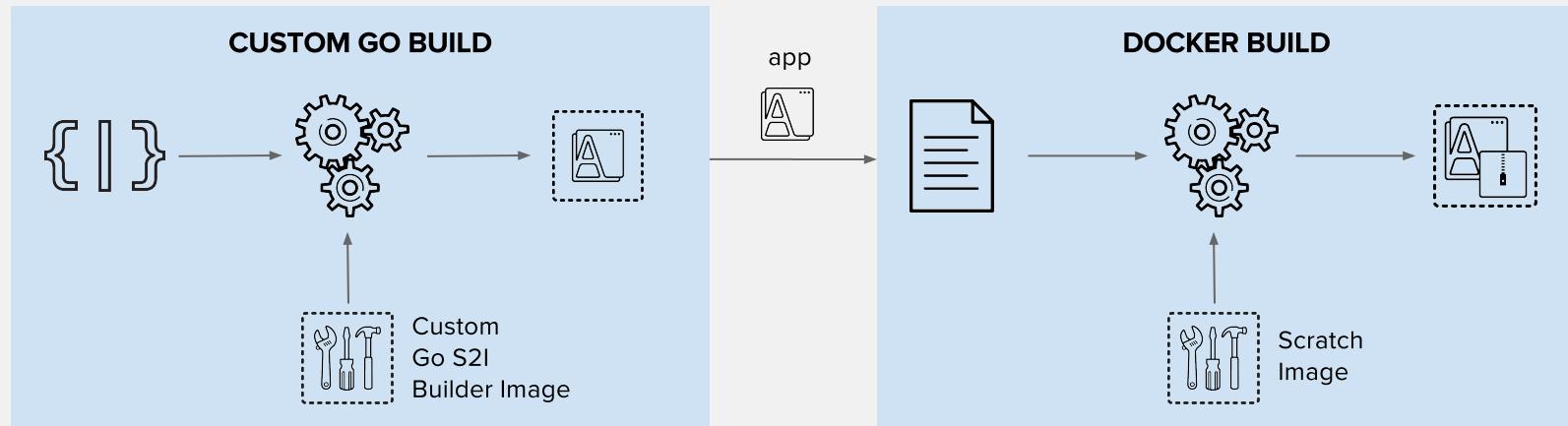
Use your choice of build tool like Gradle and deploy to official images like the JDK image



read more on <https://blog.openshift.com/chaining-builds/>

EXAMPLE: SMALL LEAN RUNTIMES

Build the app binary and deploy on small scratch images



read more on <https://blog.openshift.com/chaining-builds/>

CONTINUOUS INTEGRATION (CI) CONTINUOUS DELIVERY (CD)

CI/CD WITH BUILD AND DEPLOYMENTS

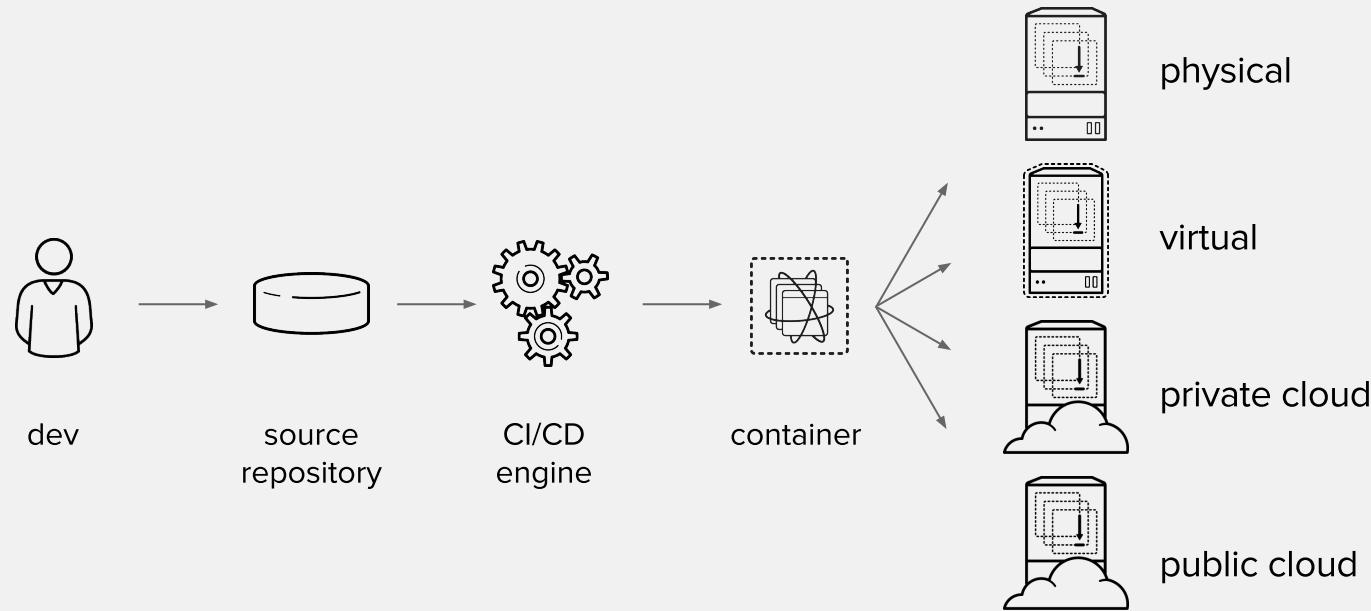
BUILDS

- Webhook triggers: build the app image whenever the code changes
- Image trigger: build the app image whenever the base language or app runtime changes
- Build hooks: test the app image before pushing it to an image registry

DEPLOYMENTS

- Deployment triggers: redeploy app containers whenever configuration changes or the image changes in the OpenShift integrated registry or upstream registries

CONTINUOUS DELIVERY WITH CONTAINERS



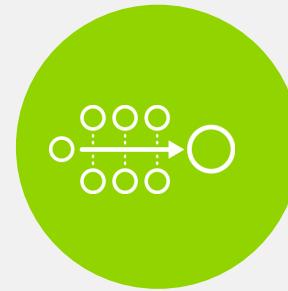
OPENSHIFT LOVES CI/CD



JENKINS-AS-A SERVICE
ON OPENSHIFT



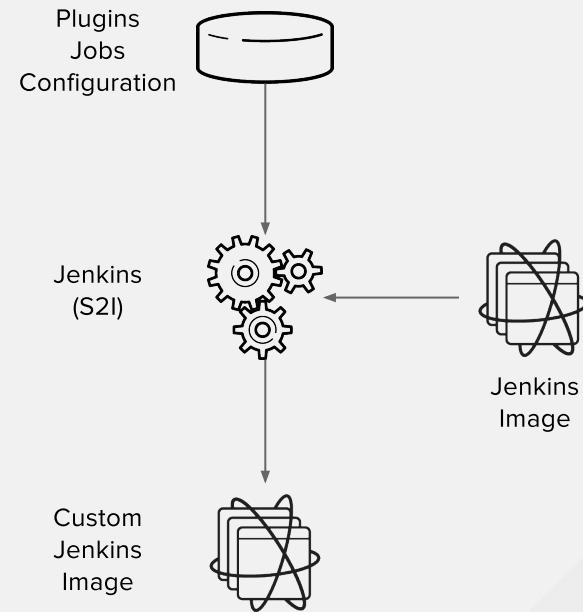
HYBRID JENKINS INFRA
WITH OPENSHIFT



EXISTING CI/CD
DEPLOY TO OPENSHIFT

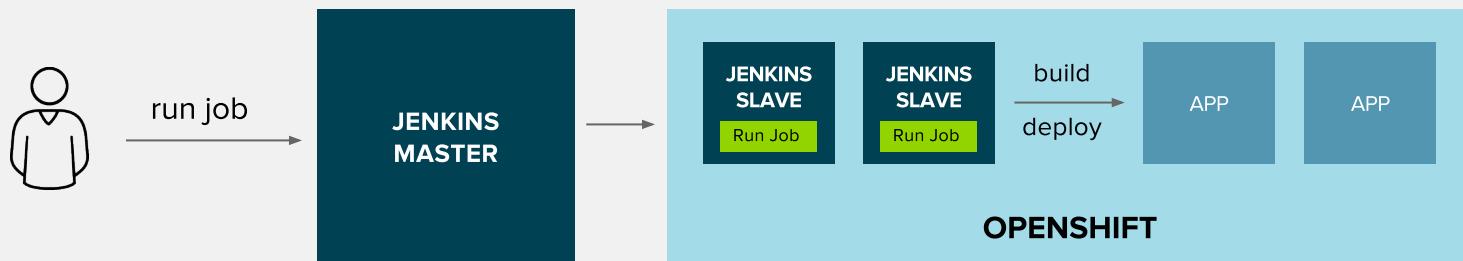
JENKINS-AS-A-SERVICE ON OPENSIFT

- Certified Jenkins images with pre-configured plugins
 - Provided out-of-the-box
 - Follows Jenkins 1.x and 2.x LTS versions
- Jenkins S2I Builder for customizing the image
 - Install Plugins
 - Configure Jenkins
 - Configure Build Jobs
- OpenShift plugins to integrate authentication with OpenShift and also CI/CD pipelines
- Dynamically deploys Jenkins slave containers



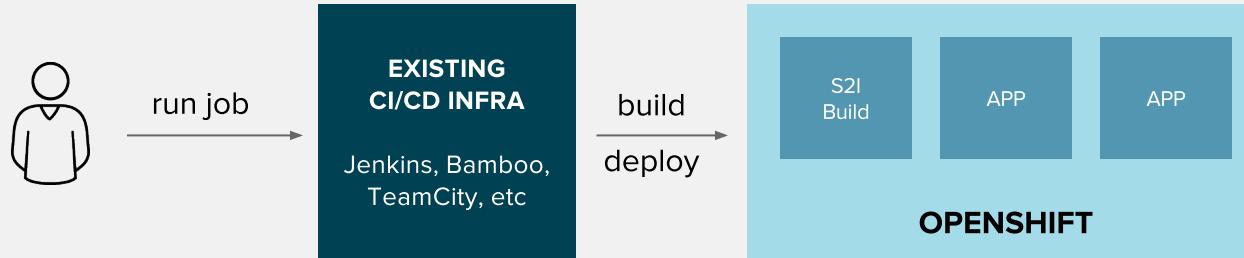
HYBRID JENKINS INFRA WITH OPENSHIFT

- Scale existing Jenkins infrastructure by dynamically provisioning Jenkins slaves on OpenShift
- Use Kubernetes plug-in on existing Jenkins servers



EXISTING CI/CD DEPLOY TO OPENSHIFT

- Existing CI/CD infrastructure outside OpenShift performs operations against OpenShift
 - OpenShift Pipeline Jenkins Plugin for Jenkins
 - OpenShift CLI for integrating other CI Engines with OpenShift
- Without disrupting existing processes, can be combined with previous alternative



OPENShift PIPELINES

- OpenShift Pipelines allow defining a CI/CD workflow via a Jenkins pipeline which can be started, monitored, and managed similar to other builds
- Dynamic provisioning of Jenkins slaves
- Auto-provisioning of Jenkins server
- OpenShift Pipeline strategies
 - Embedded Jenkinsfile
 - Jenkinsfile from a Git repository

```
apiVersion: v1
kind: BuildConfig
metadata:
  name: app-pipeline
spec:
  strategy:
    type: JenkinsPipeline
    jenkinsPipelineStrategy:
      jenkinsfile: |-
```

Provision a Jenkins slave for running Maven

```
      node('maven') { ←.....  
        stage('build app') {  
          git url: 'https://git/app.git'  
          sh "mvn package"  
        }  
        stage('build image') {  
          sh "oc start-build app --from-file=target/app.jar"  
        }  
        stage('deploy') {  
          openshiftDeploy deploymentConfig: 'app'  
        }  
      }
```

OpenShift Pipelines in Web Console

app-pipeline created 32 minutes ago

[Start Build](#) [Actions](#)

[Summary](#) Configuration

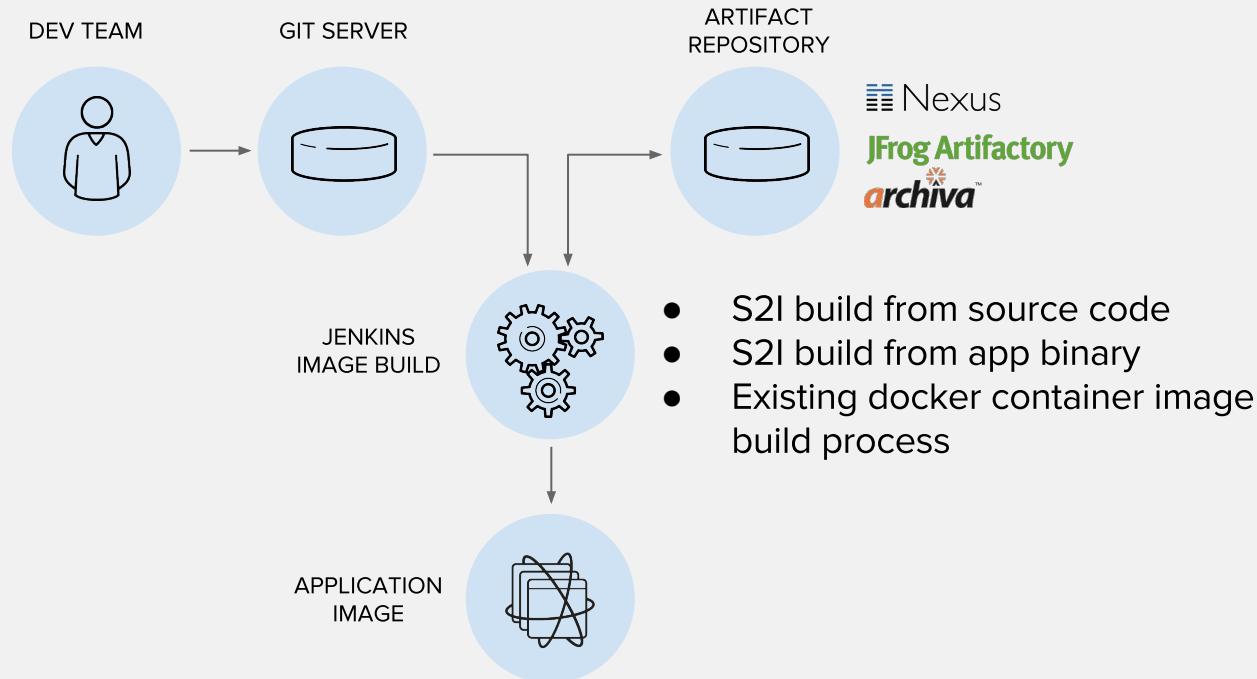
✓ Latest build #11 complete. [View Log](#)
started 16 minutes ago

Build Number	Status	Duration
#2	Failed	2m 20s
#3	Failed	2m 0s
#8	Complete	1m 40s
#9	Complete	1m 20s
#10	Complete	1m 0s
#11	Complete	40s

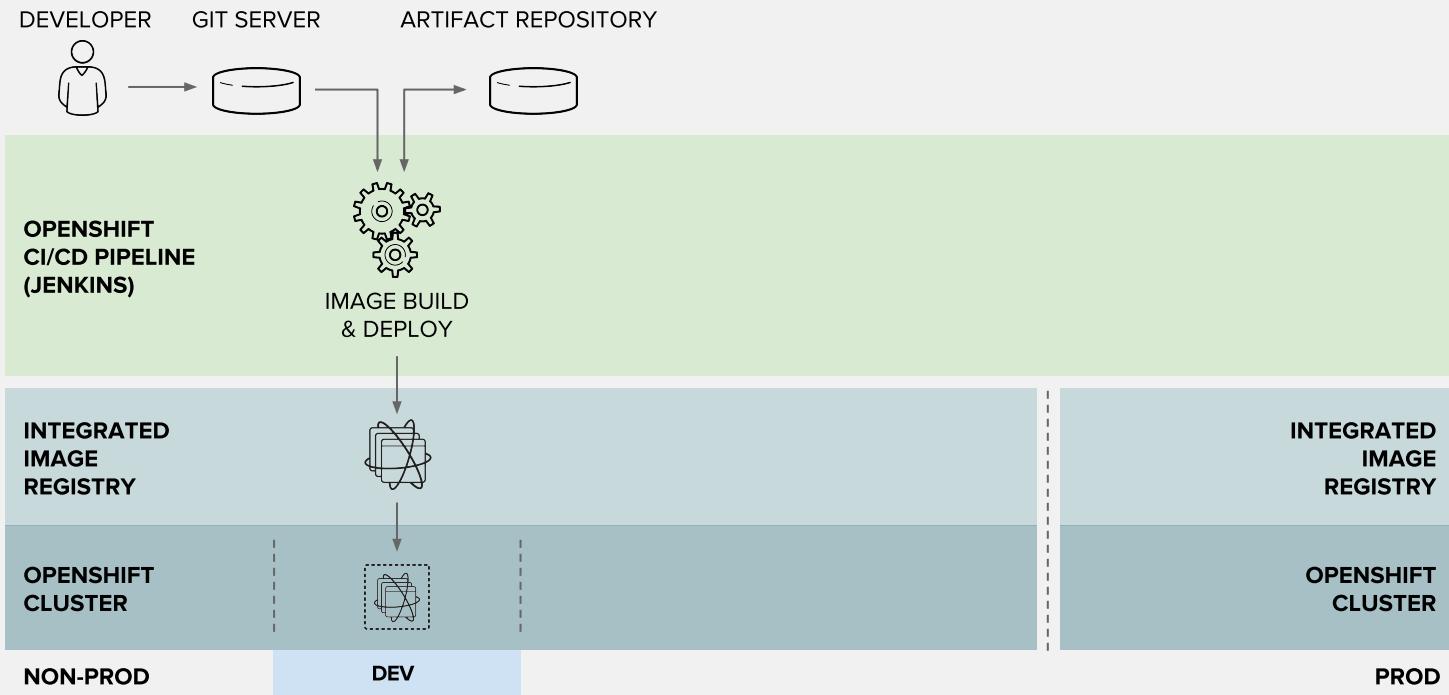
— Average: 1m 55s

Filter by label		Add	
Build #11	19 minutes ago	build app	25s
View Log		build image	16s
		deploy	45s
Build #10	19 minutes ago	build app	26s
View Log		build image	16s
		deploy	47s

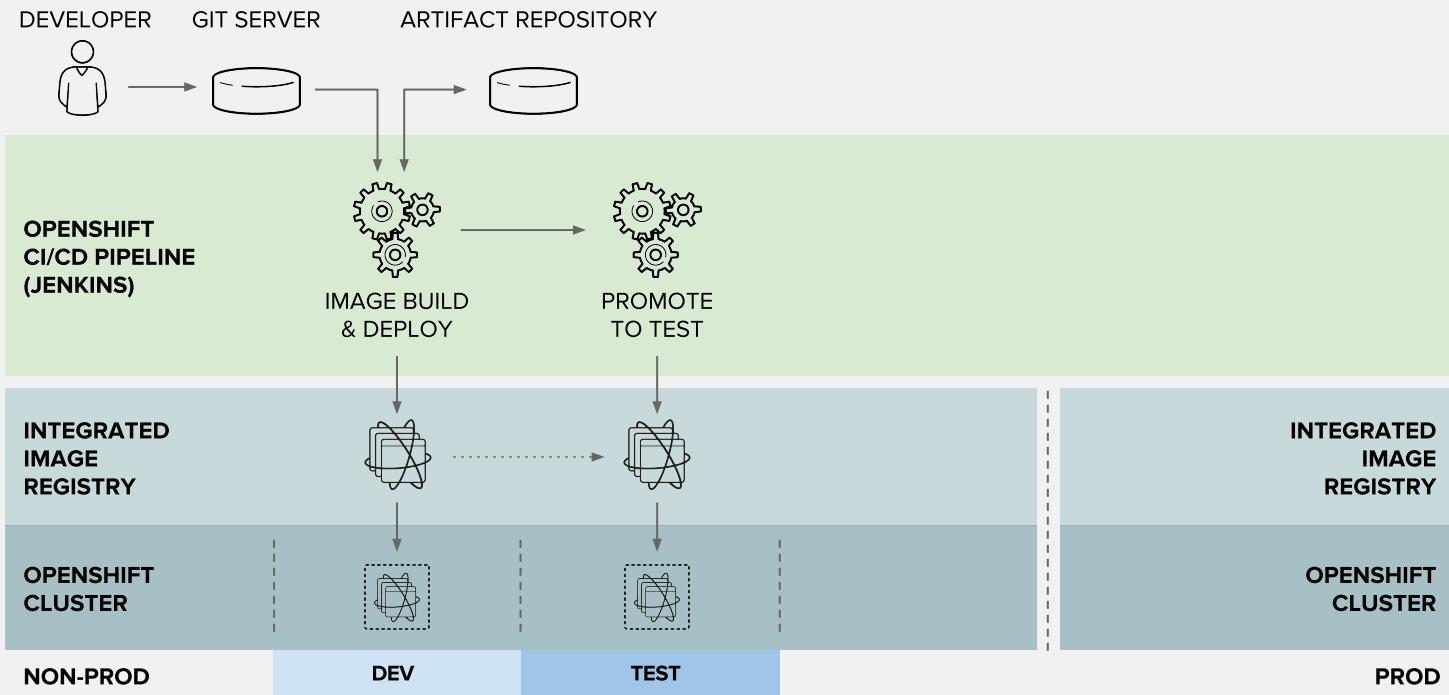
CONTINUOUS DELIVERY PIPELINE



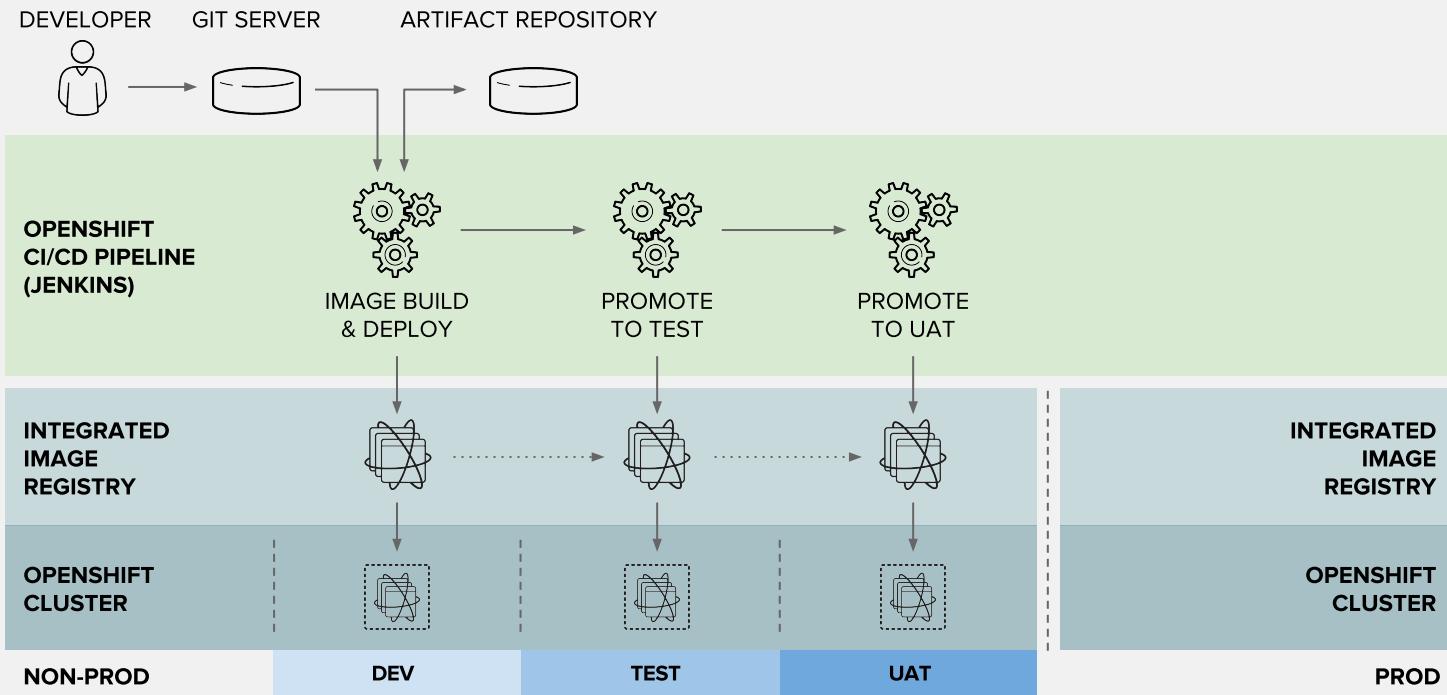
CONTINUOUS DELIVERY PIPELINE



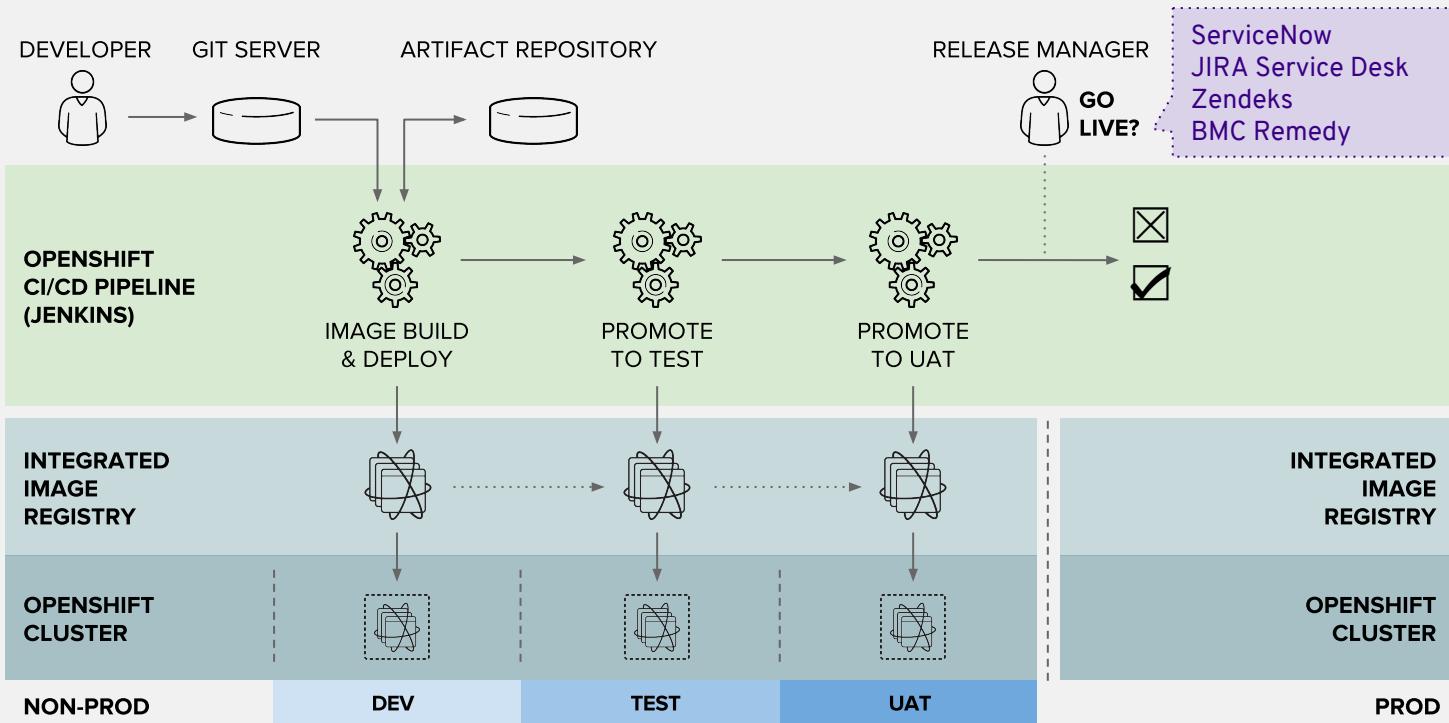
CONTINUOUS DELIVERY PIPELINE



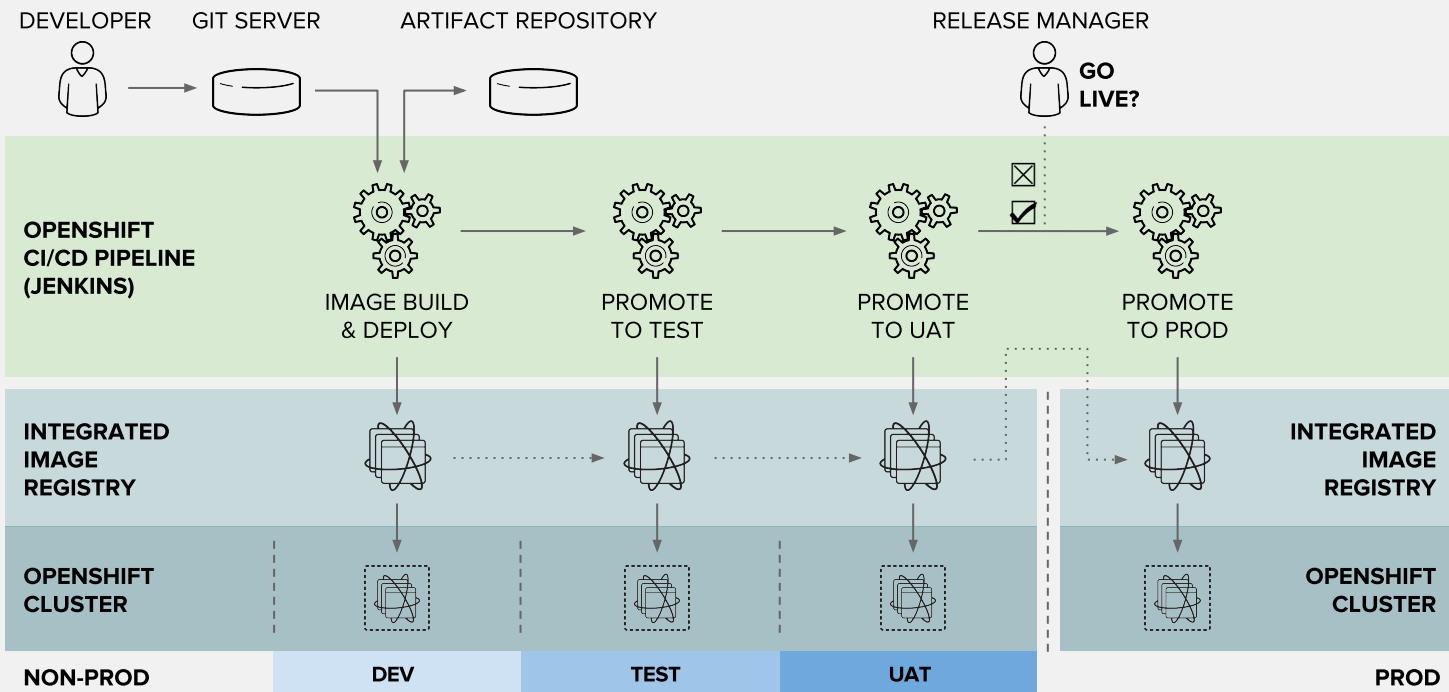
CONTINUOUS DELIVERY PIPELINE



CONTINUOUS DELIVERY PIPELINE

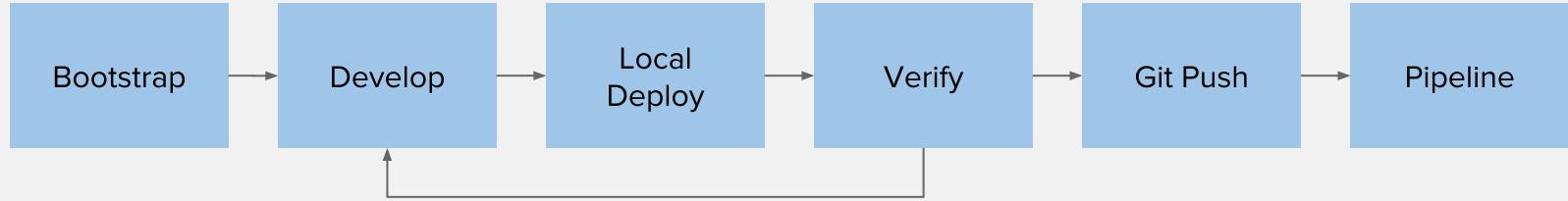


CONTINUOUS DELIVERY PIPELINE

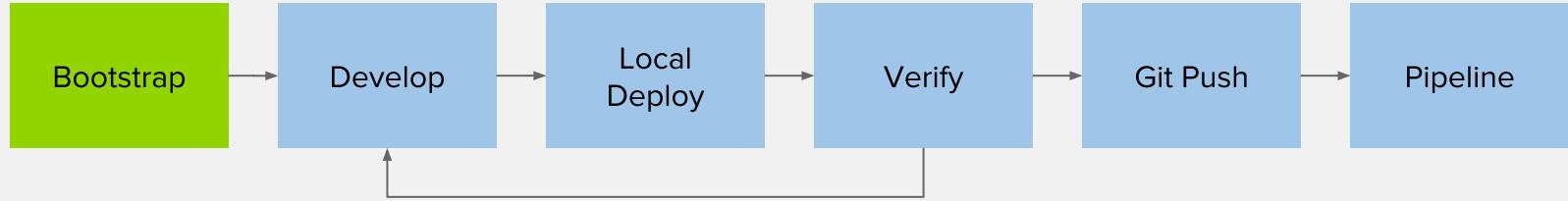


DEVELOPER WORKFLOW

LOCAL DEVELOPMENT WORKFLOW



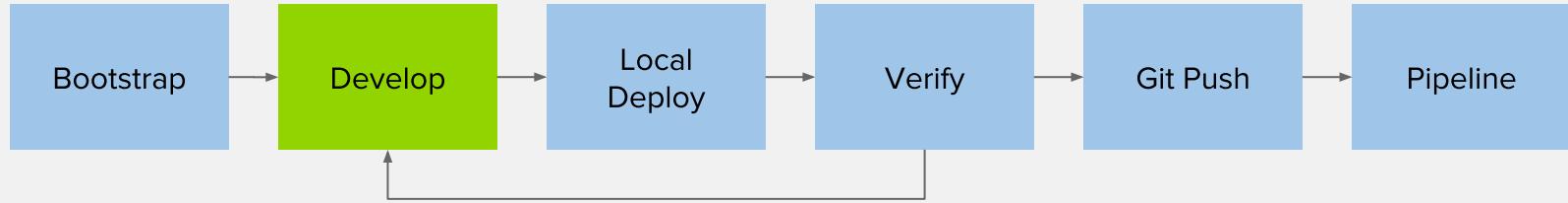
LOCAL DEVELOPMENT WORKFLOW



BOOTSTRAP

- Pick your programming language and application runtime of choice
- Create the project skeleton from scratch or use a generator such as
 - Maven archetypes
 - Quickstarts and Templates
 - OpenShift Generator
 - Spring Initializr

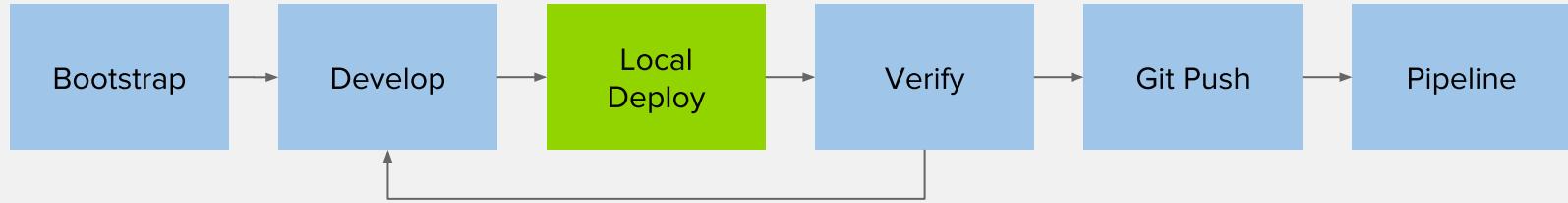
LOCAL DEVELOPMENT WORKFLOW



DEVELOP

- Pick your framework of choice such as Java EE, Spring, Ruby on Rails, Django, Express, ...
- Develop your application code using your editor or IDE of choice
- Build and test your application code locally using your build tools
- Create or generate OpenShift templates or Kubernetes objects

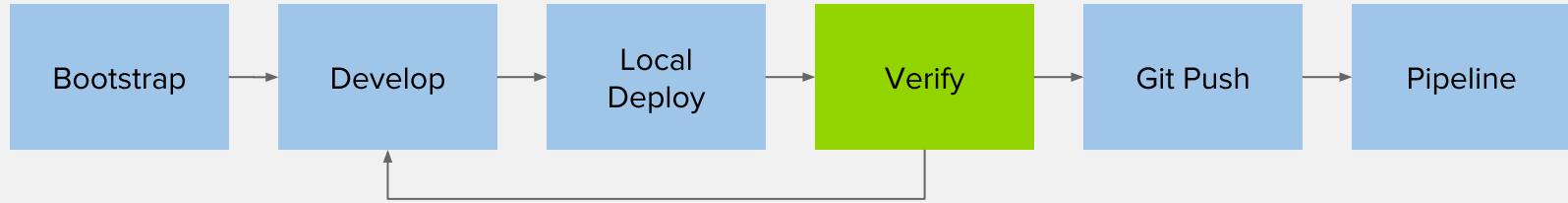
LOCAL DEVELOPMENT WORKFLOW



LOCAL DEPLOY

- Deploy your code on a local OpenShift cluster
 - Red Hat Container Development Kit (CDK), minishift and oc cluster
- Red Hat CDK provides a standard RHEL-based development environment
- Use binary deploy, maven or CLI rsync to push code or app binary directly into containers

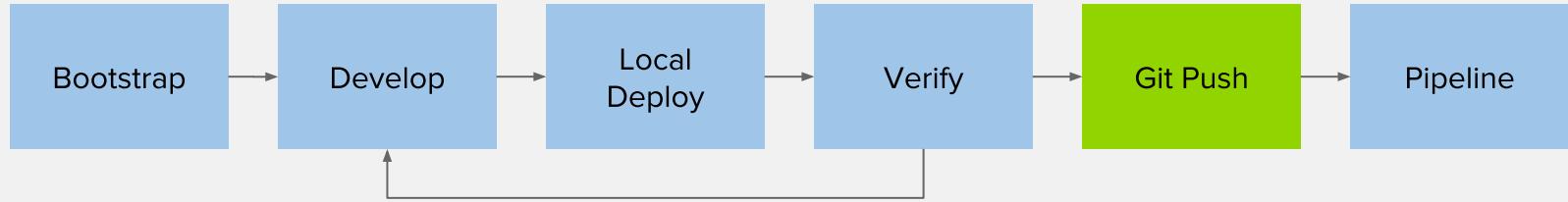
LOCAL DEVELOPMENT WORKFLOW



VERIFY

- Verify your code is working as expected
- Run any type of tests that are required with or without other components (database, etc)
- Based on the test results, change code, deploy, verify and repeat

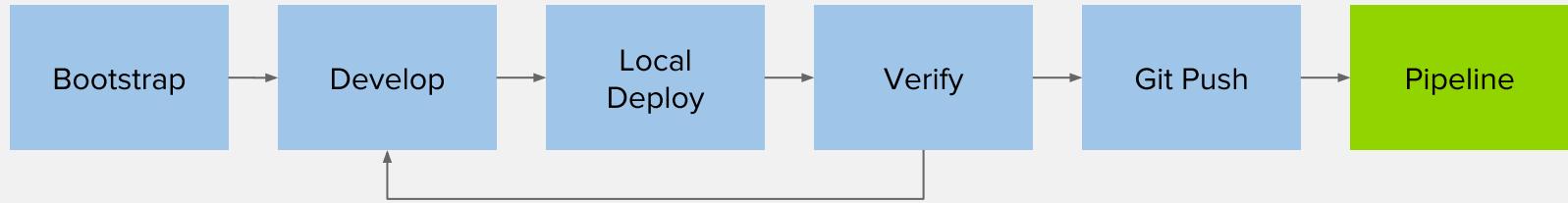
LOCAL DEVELOPMENT WORKFLOW



GIT PUSH

- Push the code and configuration to the Git repository
- If using Fork & Pull Request workflow, create a Pull Request
- If using code review workflow, participate in code review discussions

LOCAL DEVELOPMENT WORKFLOW

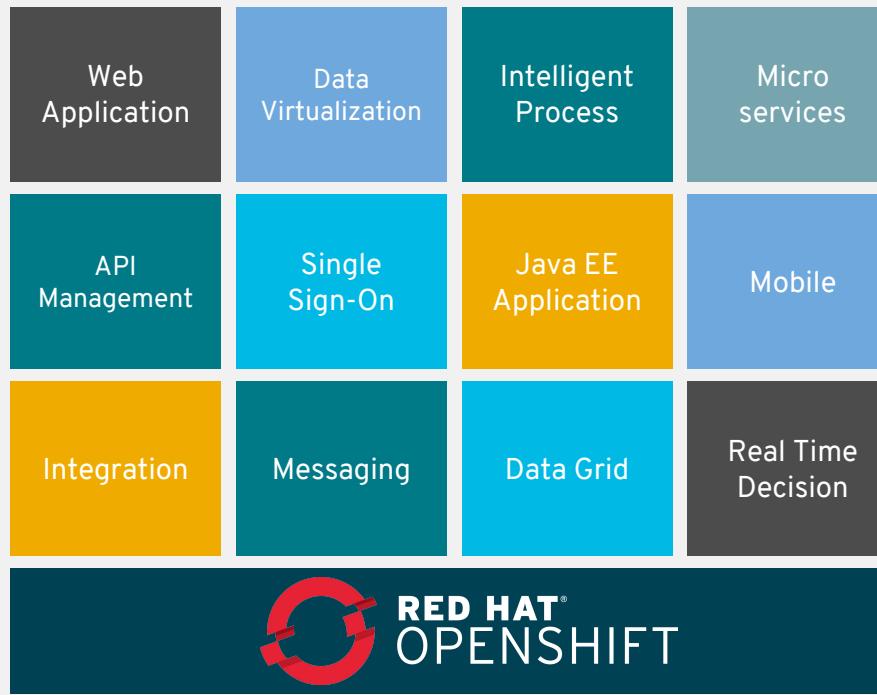


PIPELINE

- Pushing code to the Git repository triggers one or multiple deployment pipelines
- Design your pipelines based on your development workflow e.g. test the pull request
- Failure in the pipeline? Go back to the code and start again

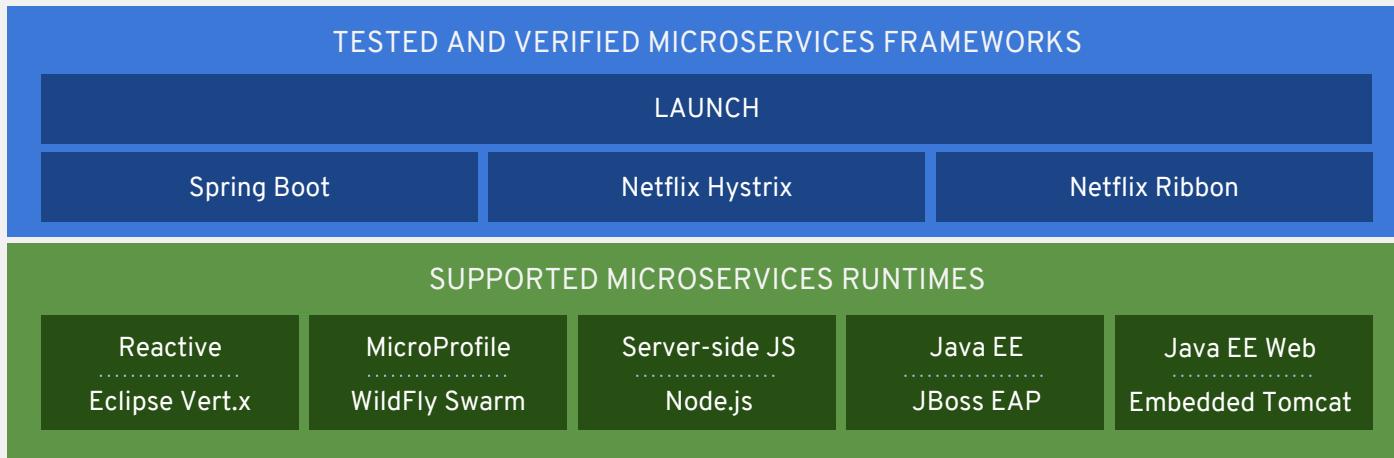
APPLICATION SERVICES

A PLATFORM THAT GROWS WITH YOUR BUSINESS



TRUE POLYGLOT PLATFORM

LANGUAGES	Java	NodeJS	Python	PHP	Perl	Ruby	.NET Core	Third-party Language Runtimes	
DATABASES	MySQL	PostgreSQL	MongoDB	Redis	<p>...and virtually any docker image out there!</p>				CrunchyData GitLab Iron.io Couchbase Sonatype EnterpriseDB NuoDB Fujitsu and many more
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	



Modern, Cloud-Native Application Runtimes and
an Opinionated Developer Experience

MICROSERVICES INFRASTRUCTURE: ISTIO SERVICE MESH

WHAT YOU NEED FOR MICROSERVICES?

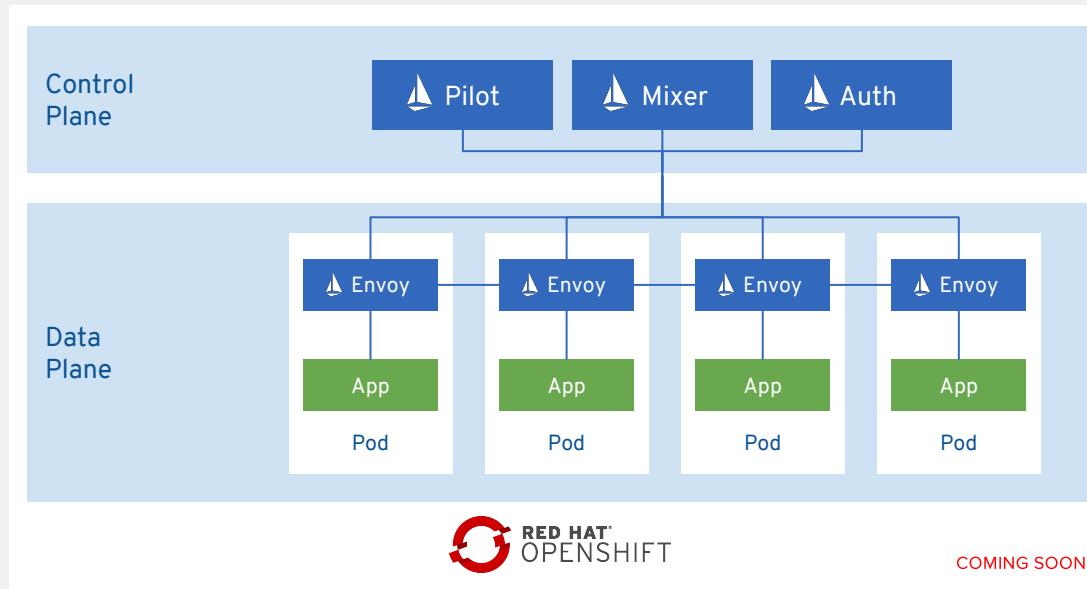


WHAT YOU NEED FOR MICROSERVICES?



WHAT IS ISTIO?

a service mesh to connect, manage, and secure microservices





redhat.

THANK YOU



plus.google.com/+RedHat



linkedin.com/company/red-hat



youtube.com/user/RedHatVideos



facebook.com/redhatinc



twitter.com/RedHatNews