redhat.

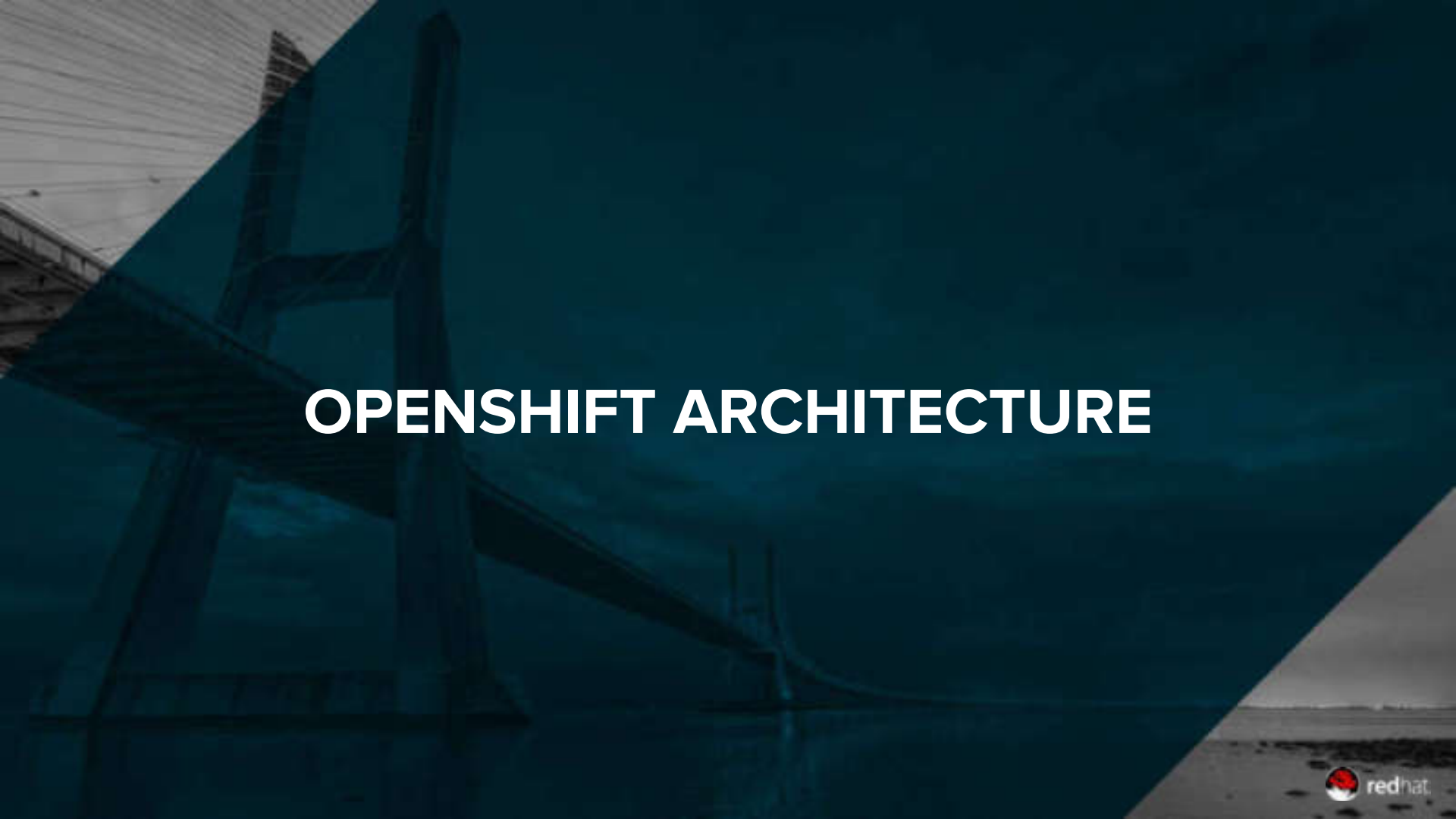
OSEC

FORUM 2017

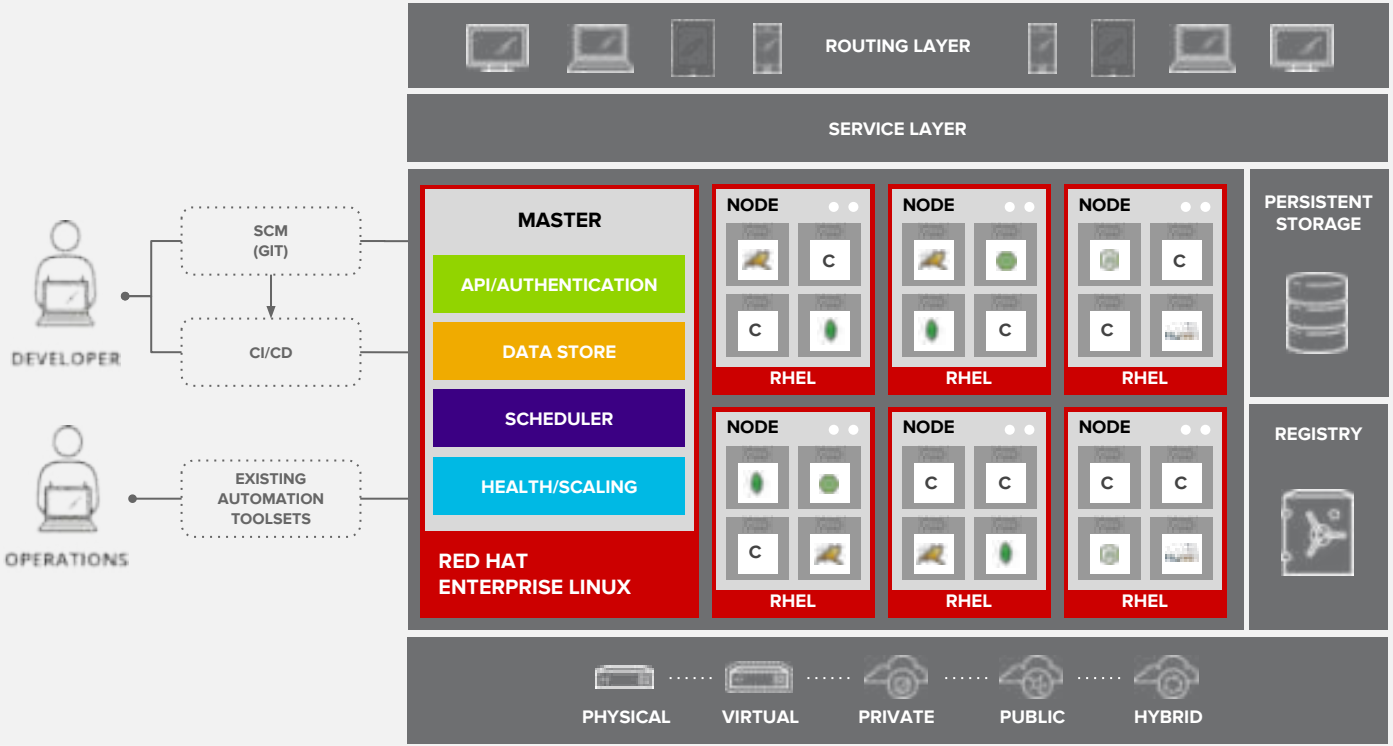
DevOps z Red Hat OpenShift Container Platform czyli jak szybko wdrożyć nową wersję aplikacji

Jaroslaw Stakun
Senior Solution Architect
Red Hat CEE


Self-Service
Multi-language
Automation
Collaboration
Seamless
RED HAT
OPENSHIFT
Standards-based
Web-scale
Open Source
Enterprise Grade
Secure


OPENSHIFT ARCHITECTURE
redhat.

OPENSHIFT ARCHITECTURE

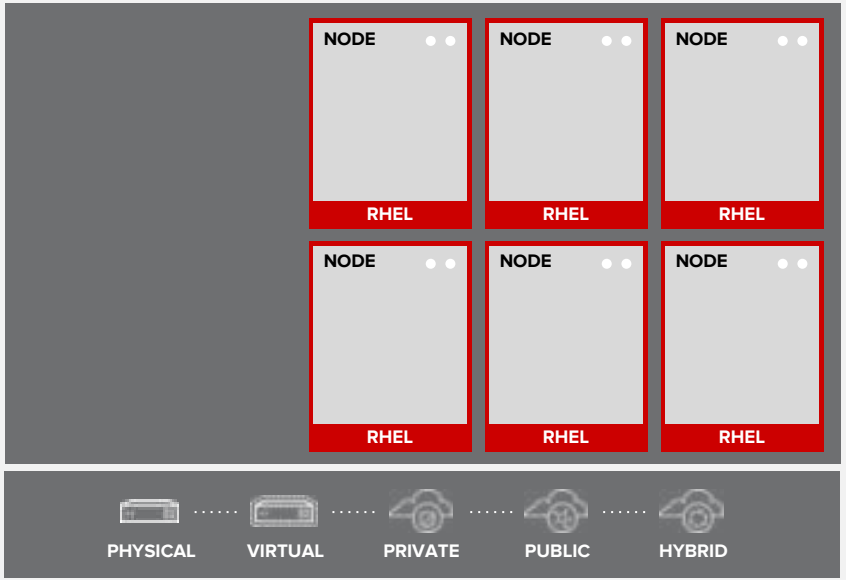


redhat

YOUR CHOICE OF INFRASTRUCTURE

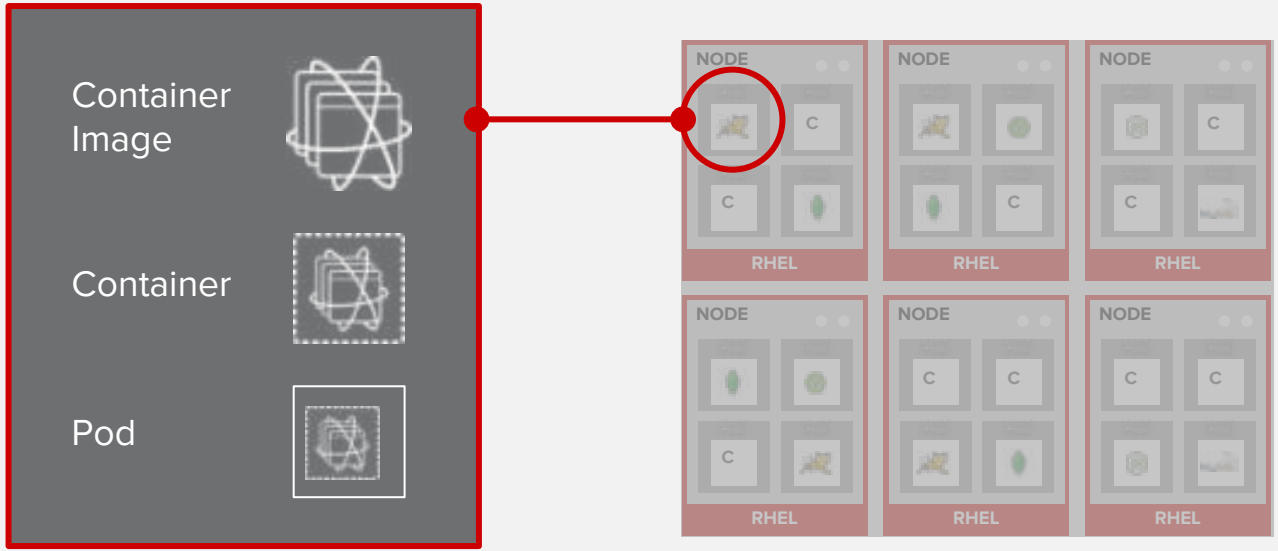
redhat

NODES RHEL INSTANCES WHERE APPS RUN

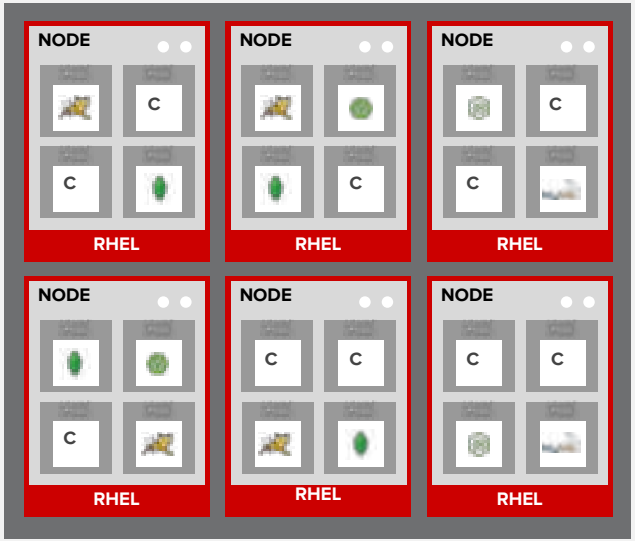


redhat

APPS RUN IN CONTAINERS

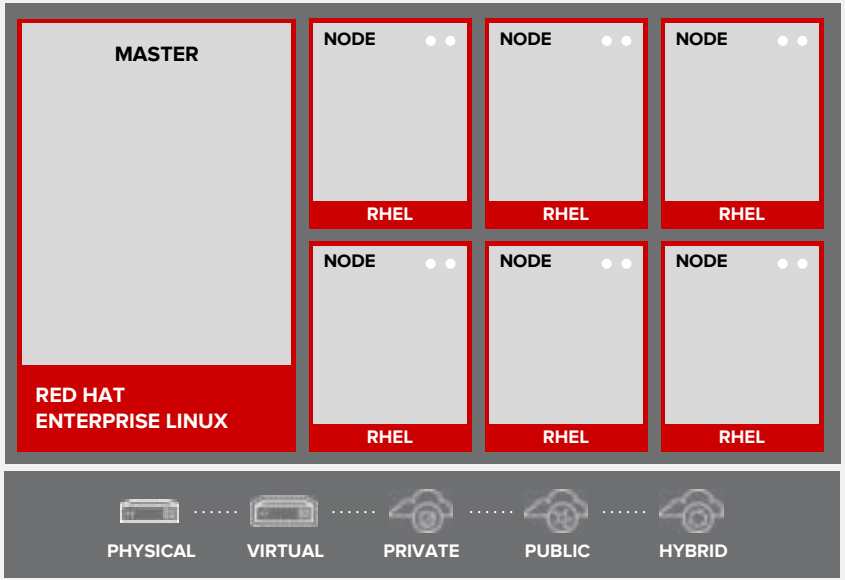


PODS ARE THE UNIT OF ORCHESTRATION



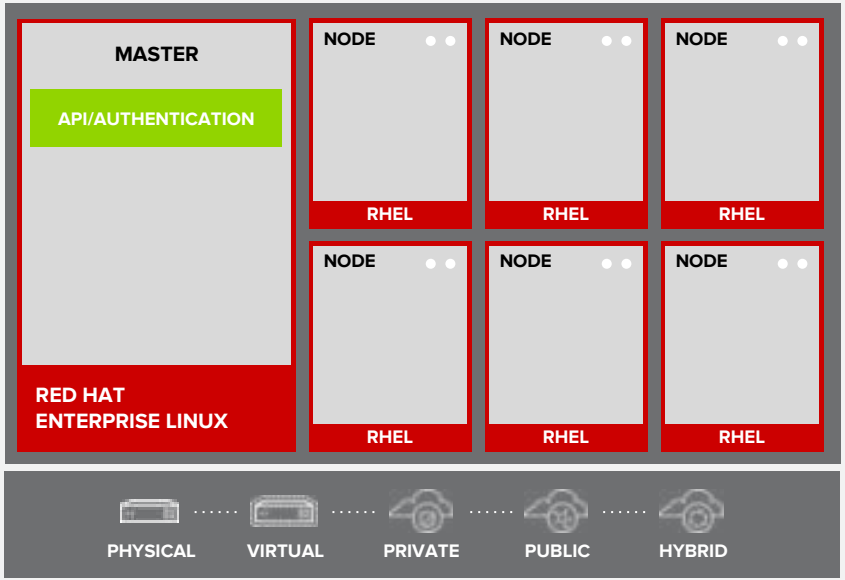
redhat

MASTERS ARE THE CONTROL PLANE



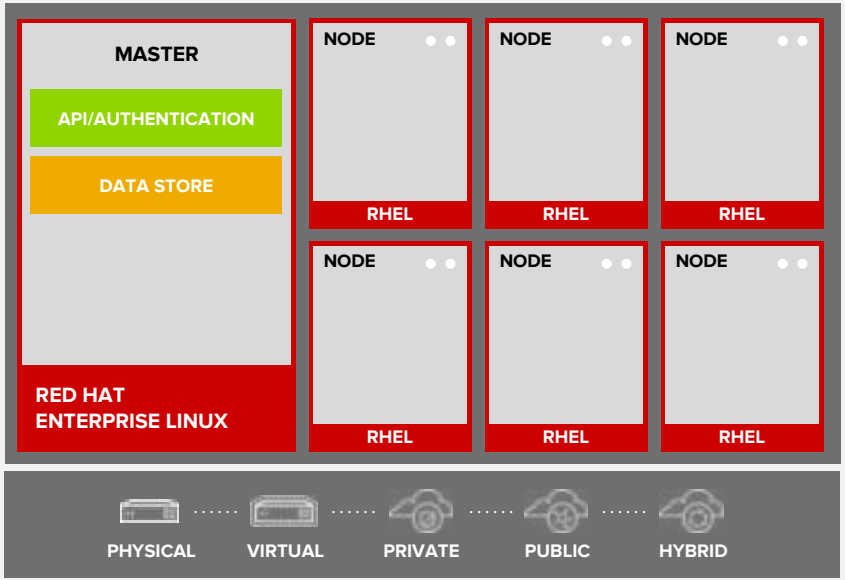
redhat

API AND AUTHENTICATION



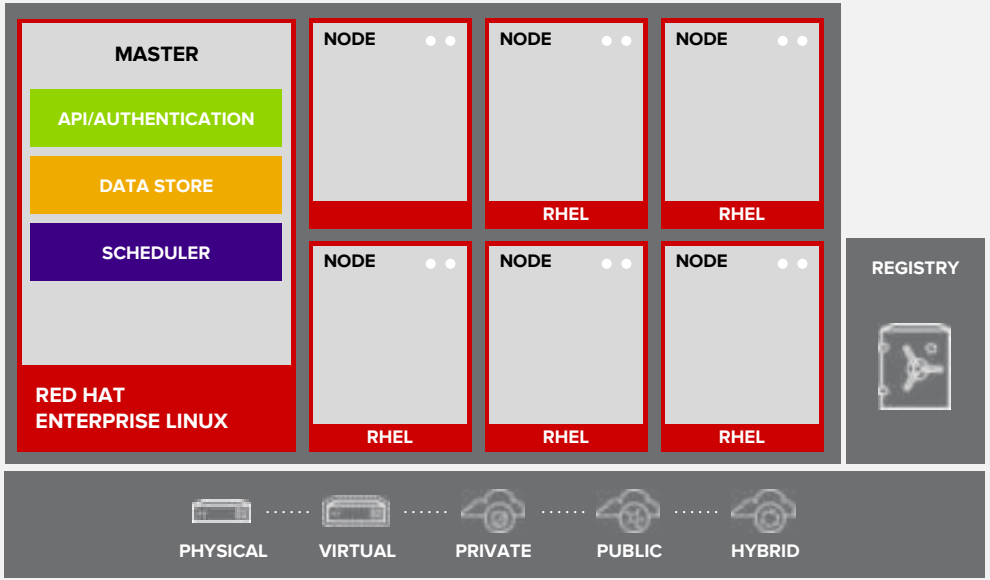
redhat

DESIRED AND CURRENT STATE

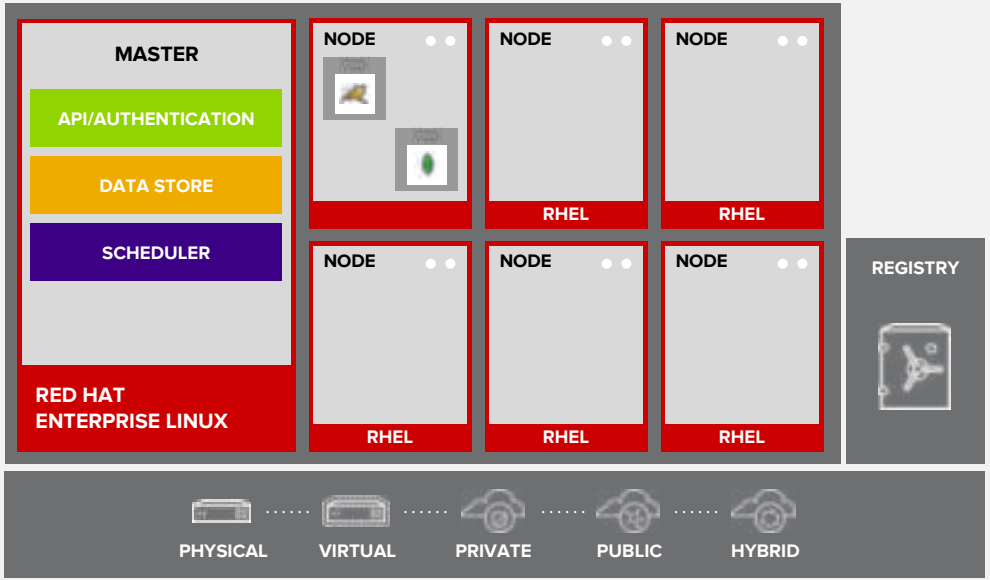


redhat

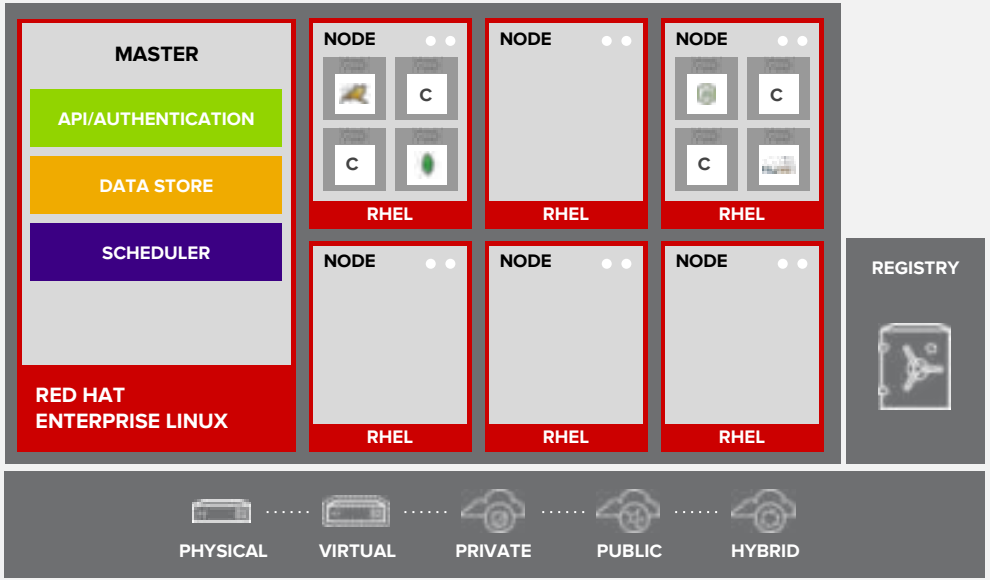
INTEGRATED CONTAINER REGISTRY



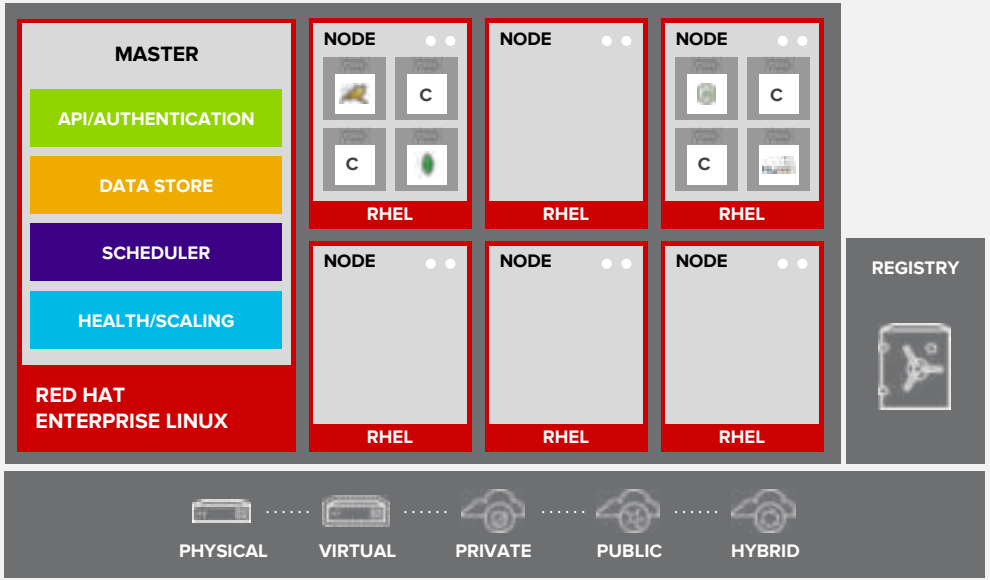
ORCHESTRATION AND SCHEDULING



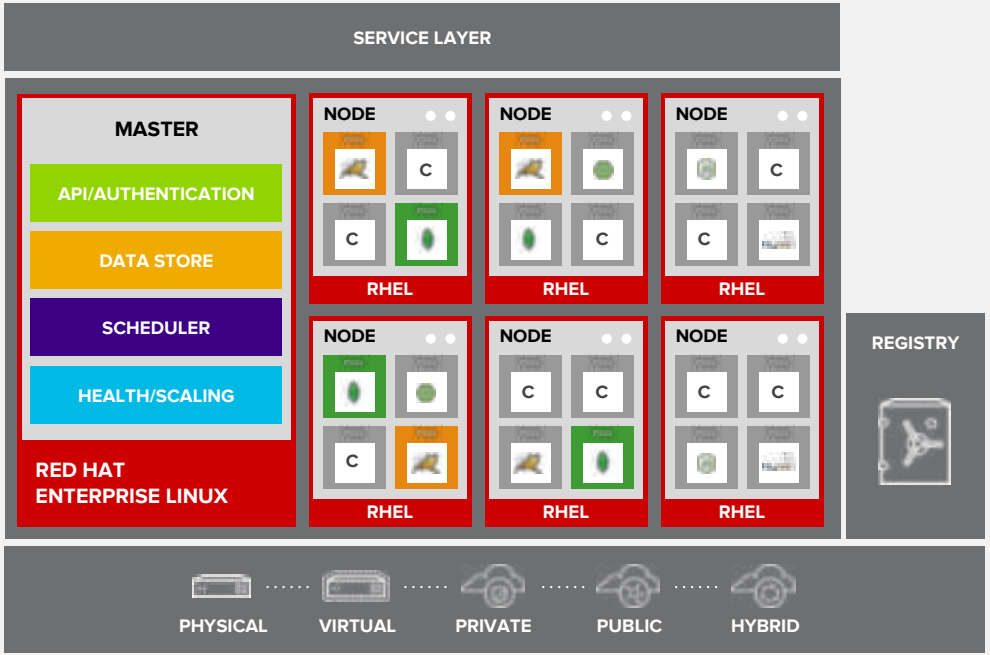
PLACEMENT BY POLICY



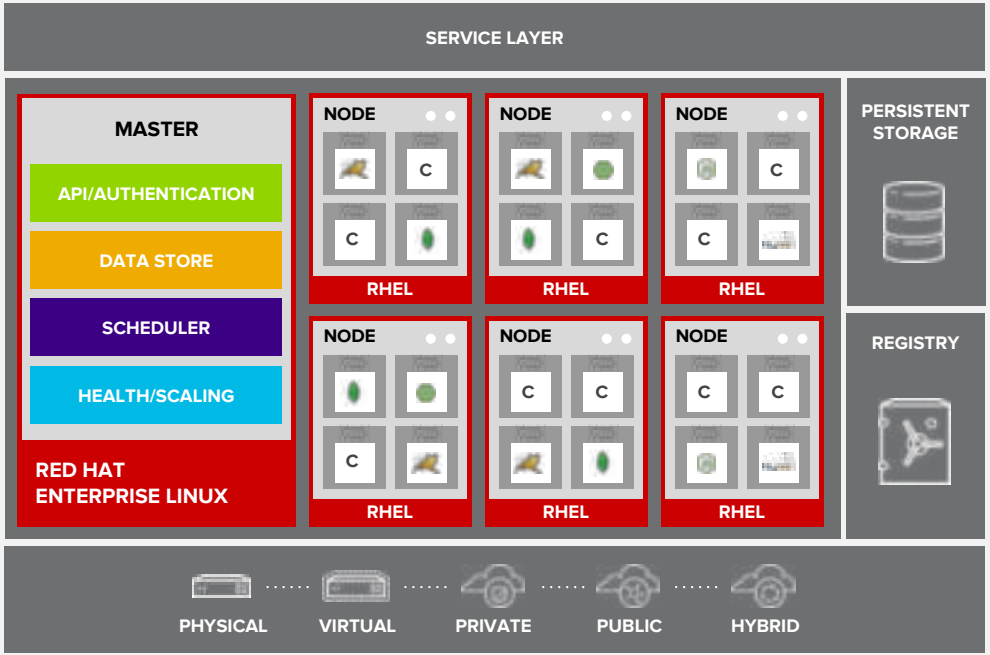
AUTOSCALING PODS



SERVICE DISCOVERY

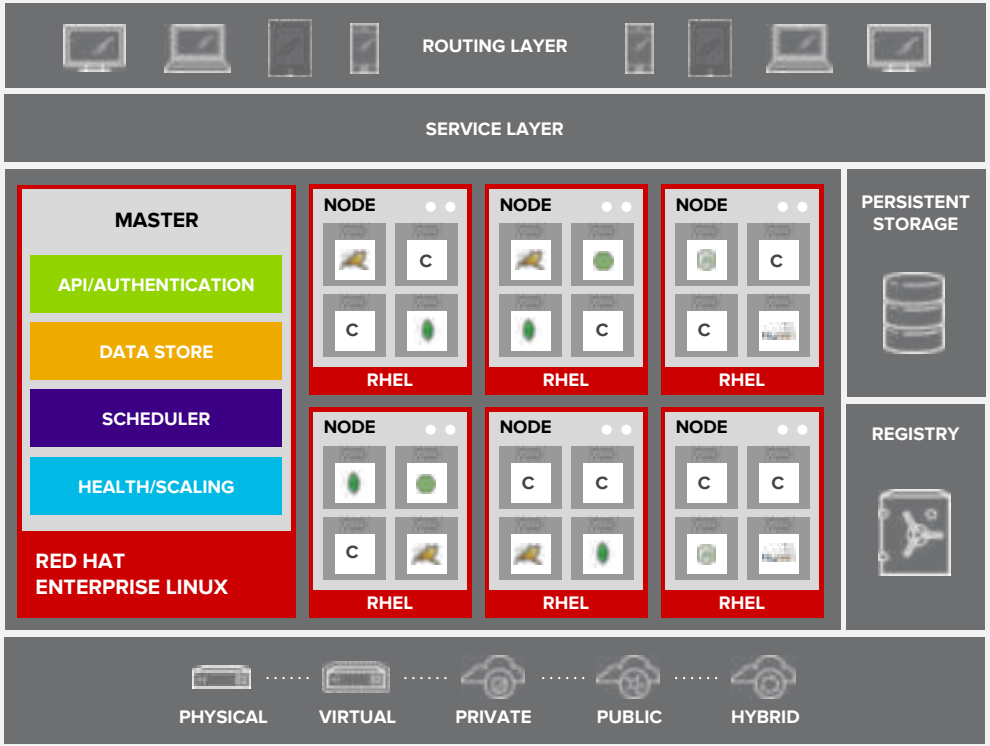


PERSISTENT DATA IN CONTAINERS



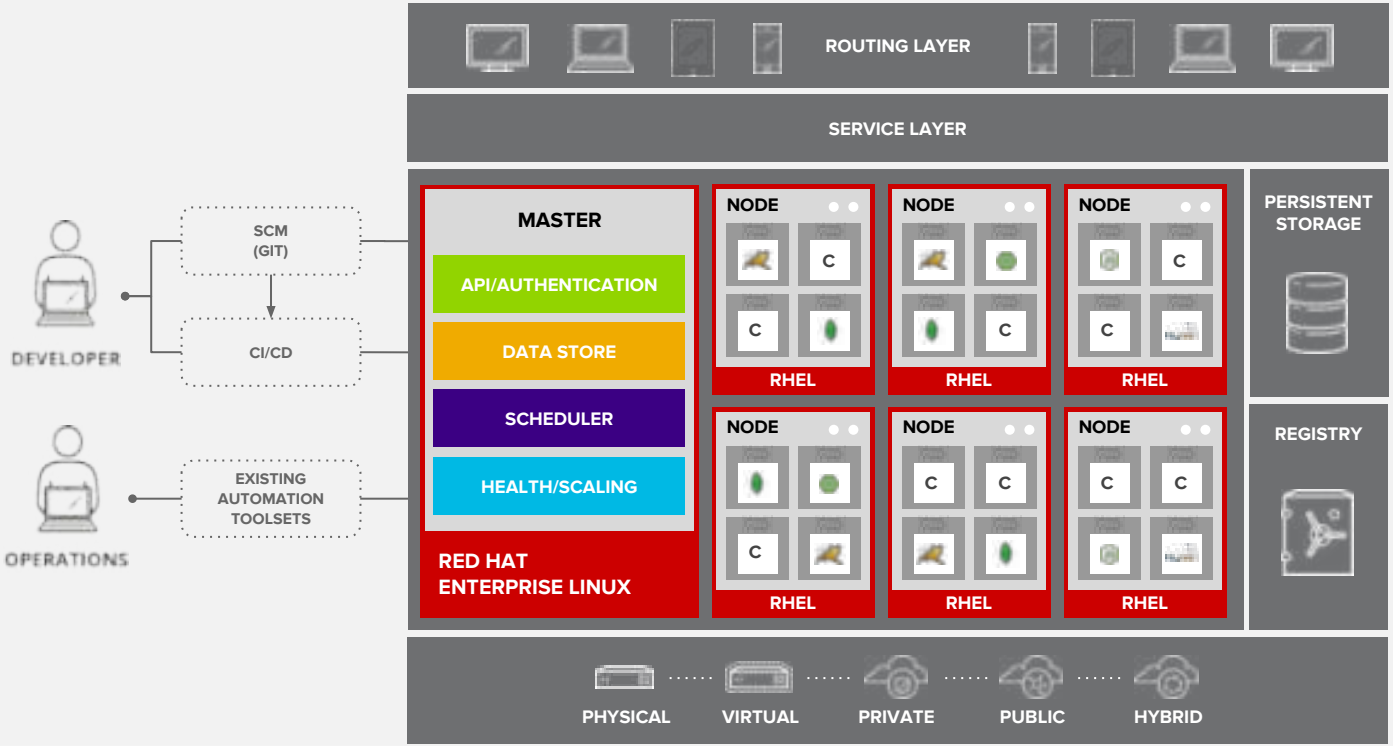
redhat

ROUTING AND LOAD-BALANCING



redhat

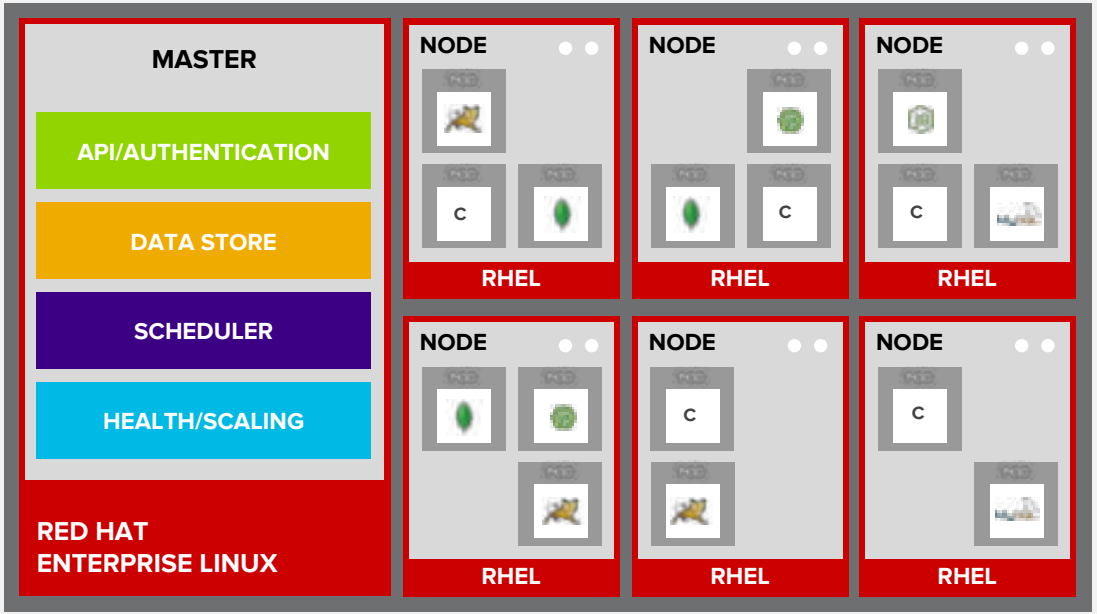
ACCESS VIA WEB, CLI, IDE AND API



redhat

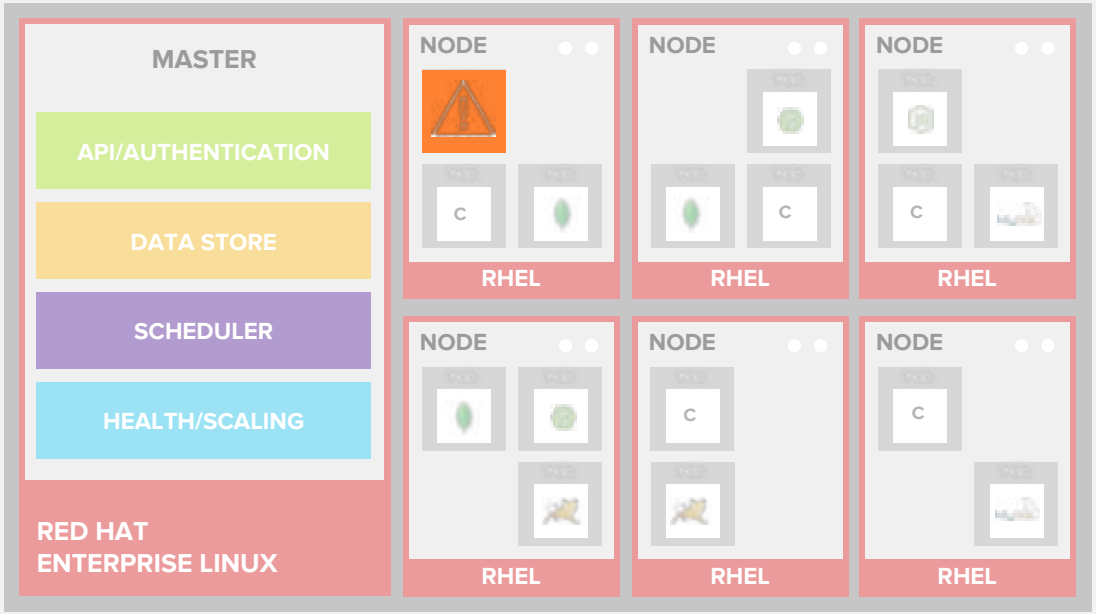

TECHNICAL DEEP DIVE
redhat.

AUTO-HEALING FAILED CONTAINERS

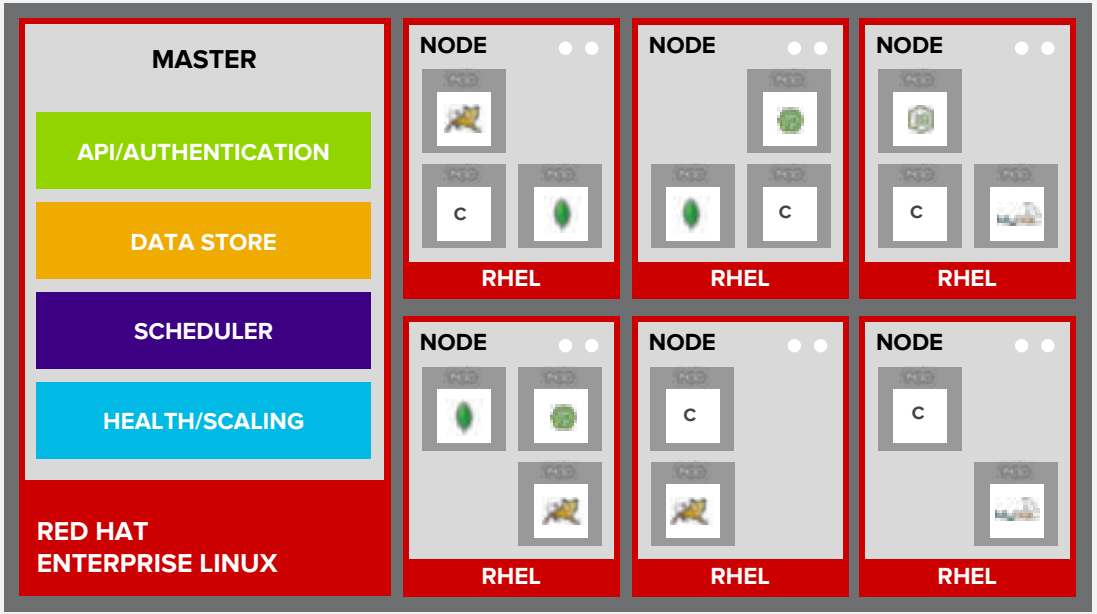


redhat

AUTO-HEALING FAILED CONTAINERS

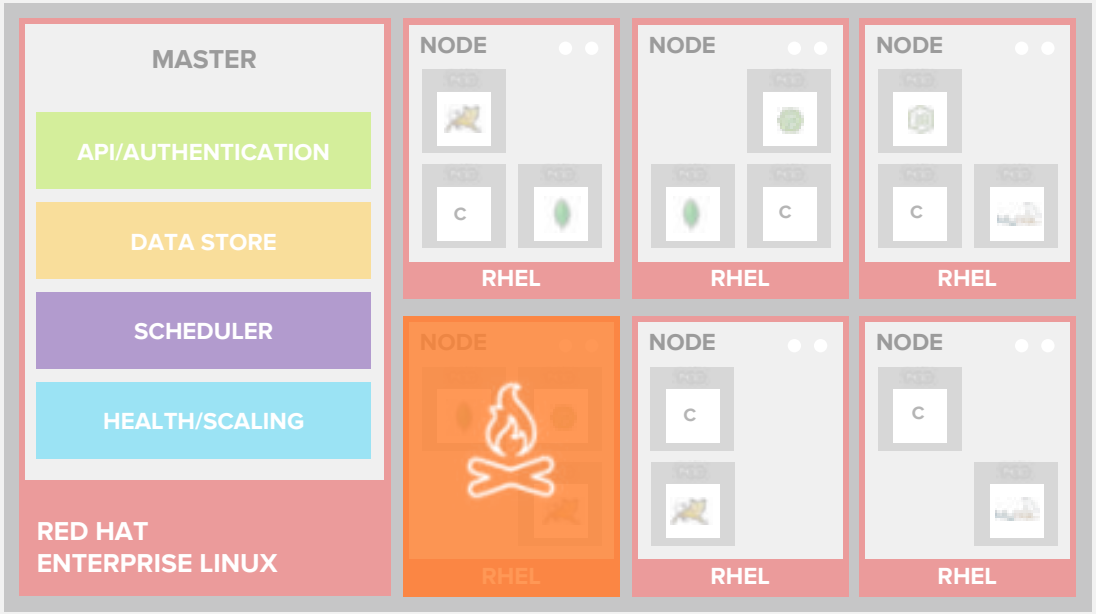


AUTO-HEALING FAILED CONTAINERS

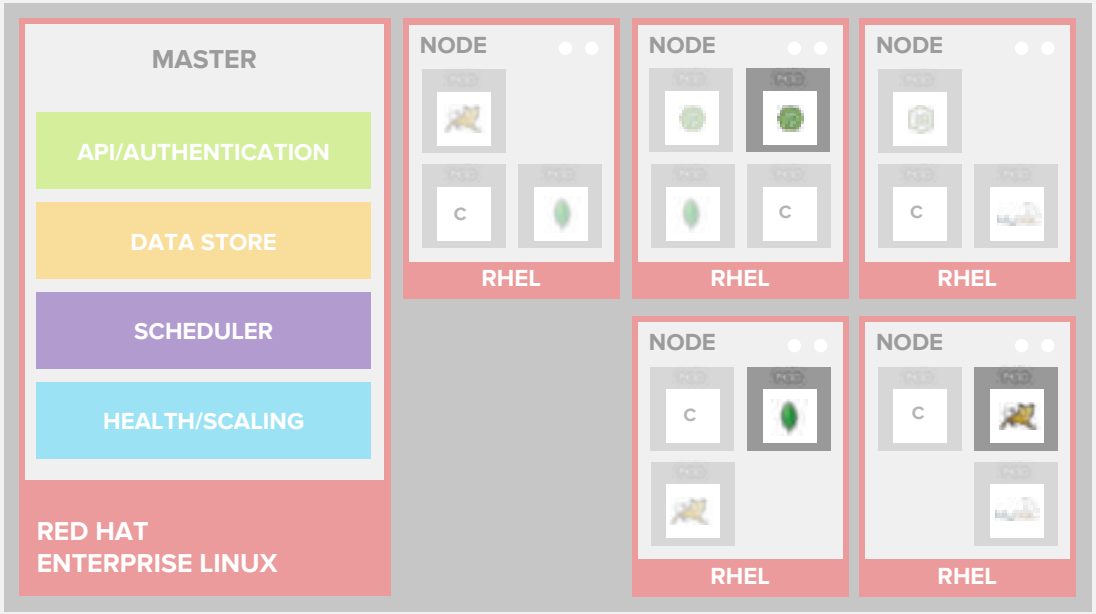


redhat

AUTO-HEALING FAILED CONTAINERS

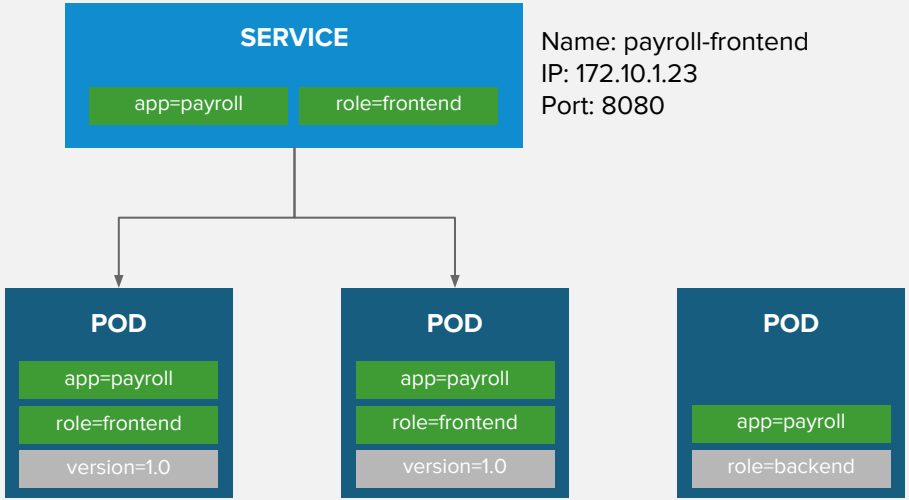


AUTO-HEALING FAILED CONTAINERS

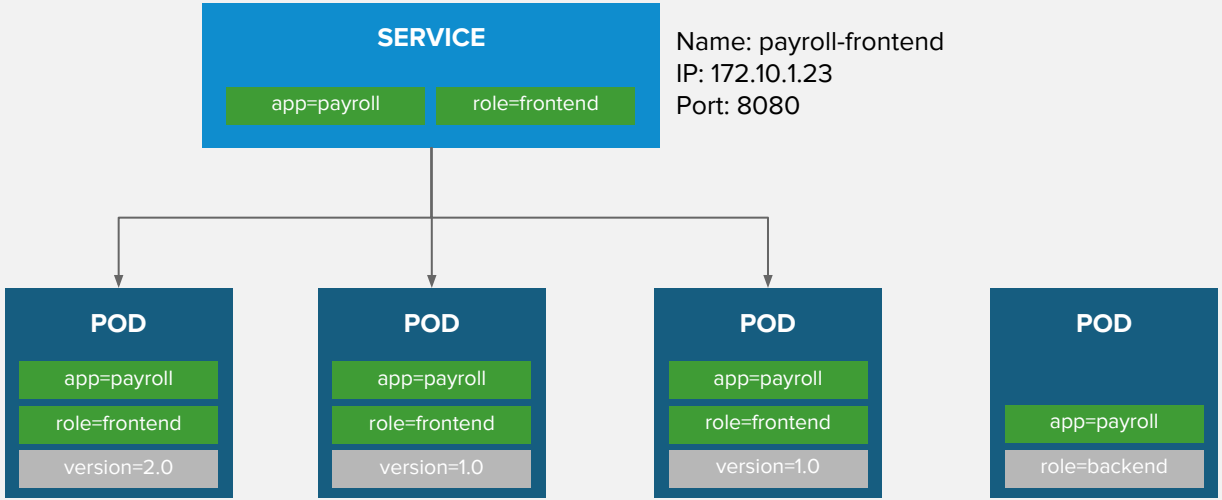


NETWORKING

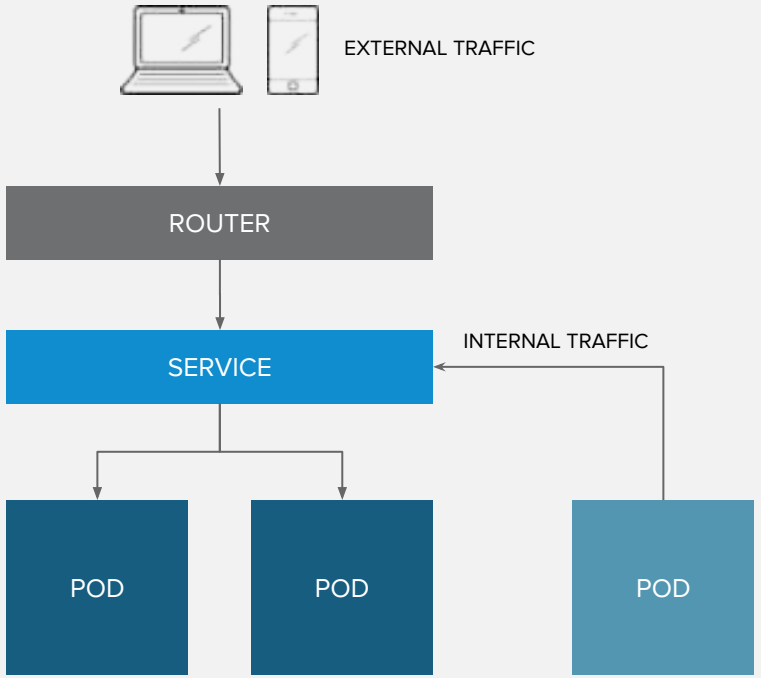
BUILT-IN SERVICE DISCOVERY
INTERNAL LOAD-BALANCING



BUILT-IN SERVICE DISCOVERY
INTERNAL LOAD-BALANCING

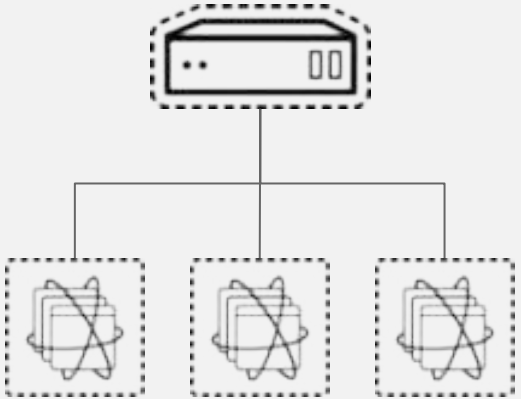


ROUTE EXPOSES SERVICES EXTERNALLY



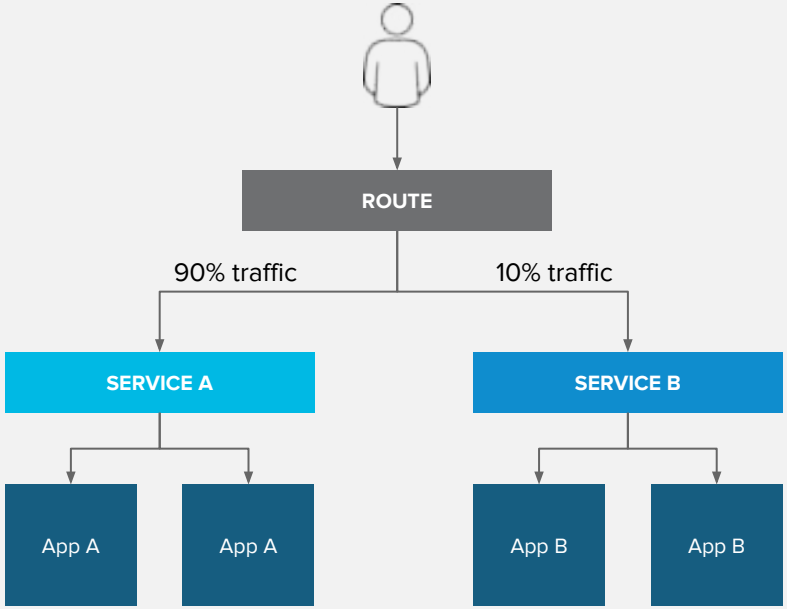
ROUTING AND EXTERNAL LOAD-BALANCING

- 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



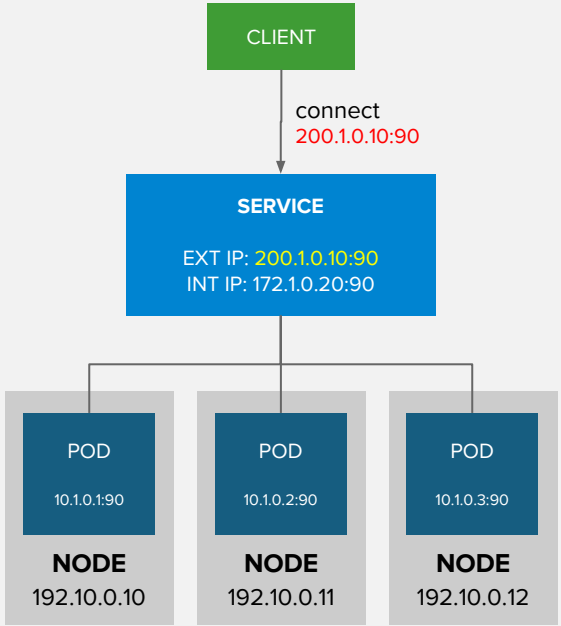
ROUTE SPLIT TRAFFIC

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



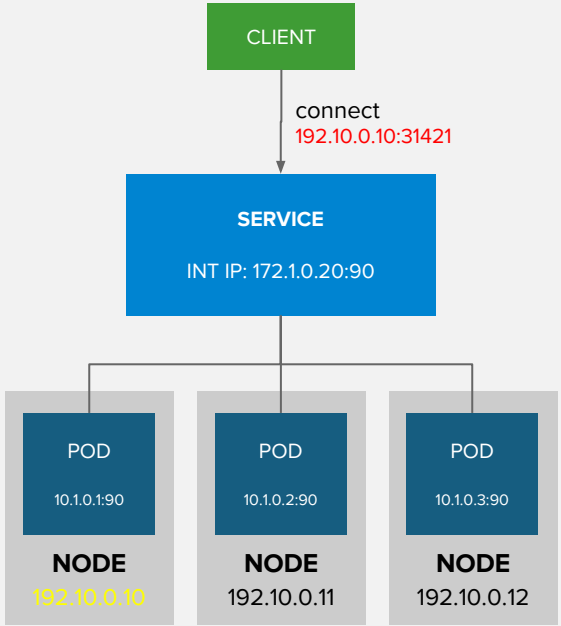
ASSIGN EXTERNAL IP TO SERVICES

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

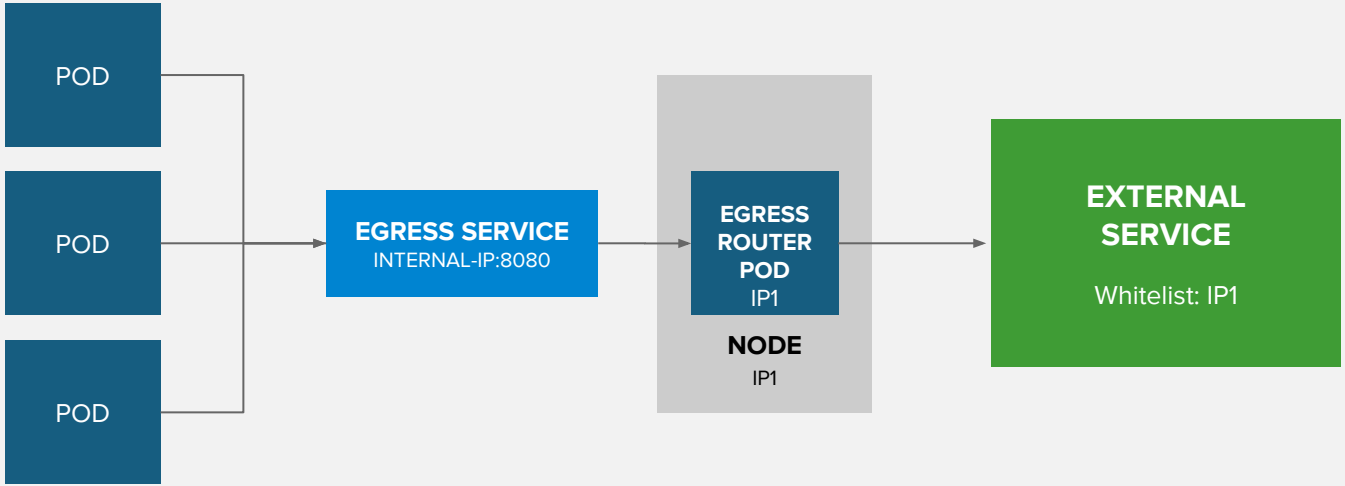


EXPOSE A SERVICE ON ALL NODE IPS

- 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

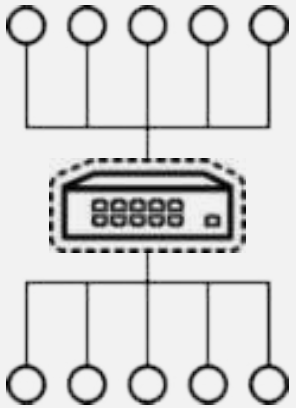


CONTROL OUTGOING TRAFFIC SOURCE IP WITH EGRESS ROUTER

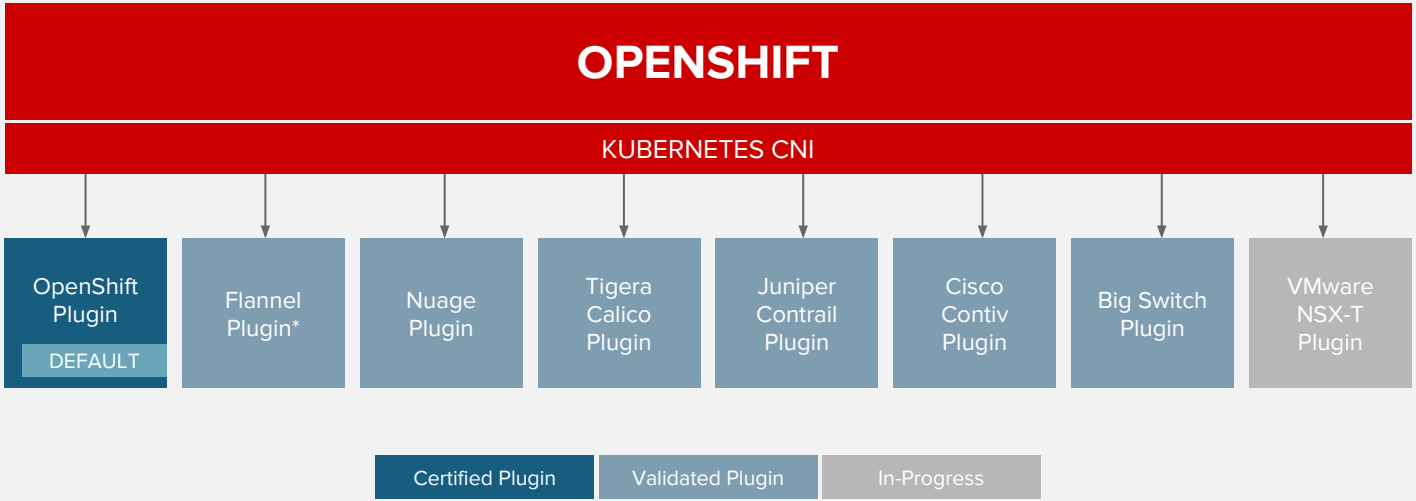


OPENSHIFT NETWORKING

- 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

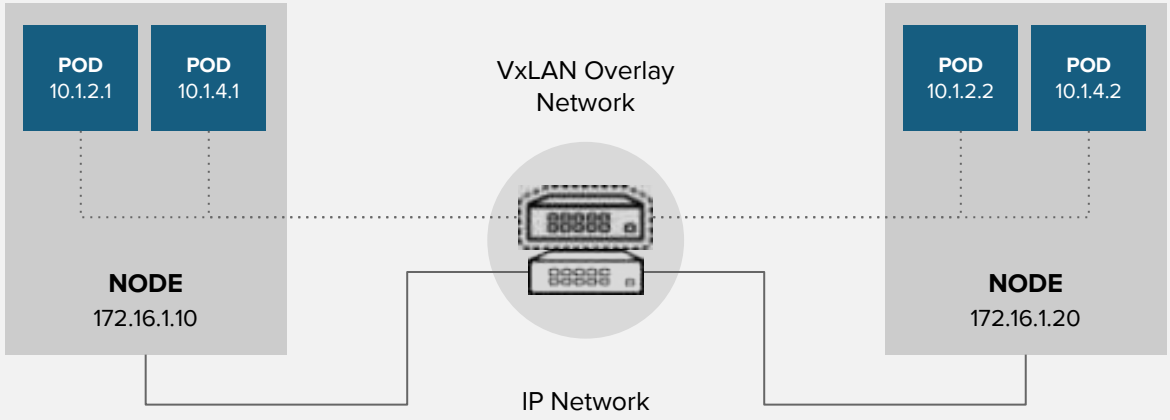


For a Complete List of Certified Plugins refer to OpenShift Third-Party SDN FAQ

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

redhat

OPENSHIFT NETWORKING



OPENSHIFT SDN

FLAT NETWORK (Default)

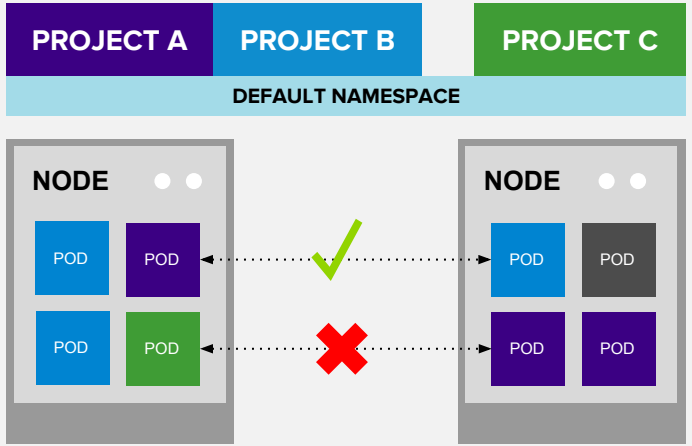
- All pods can communicate with each other across projects

MULTI-TENANT NETWORK

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

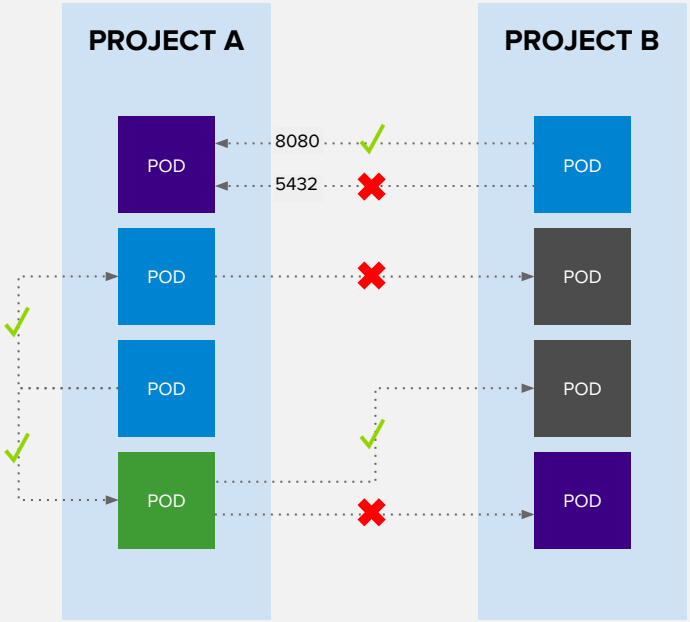
NETWORK POLICY (Tech Preview)

- Granular policy-based isolation



Multi-Tenant Network

OPENSHIFT SDN - NETWORK POLICY



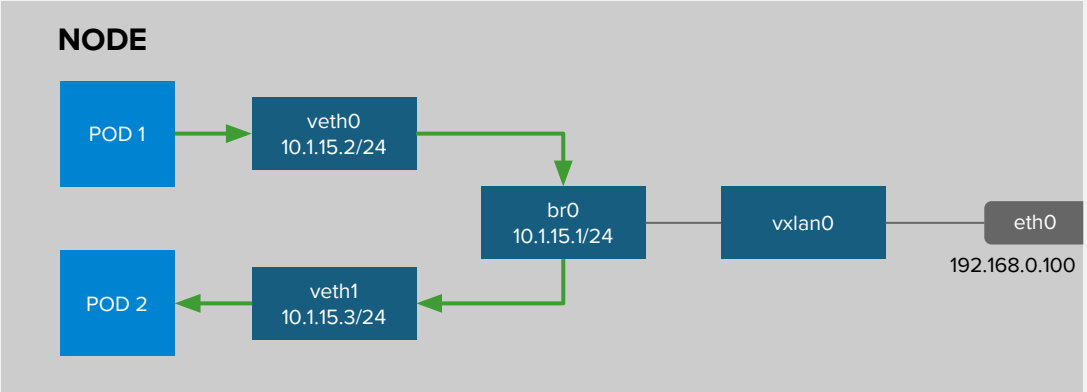
Example Policies

- 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
```

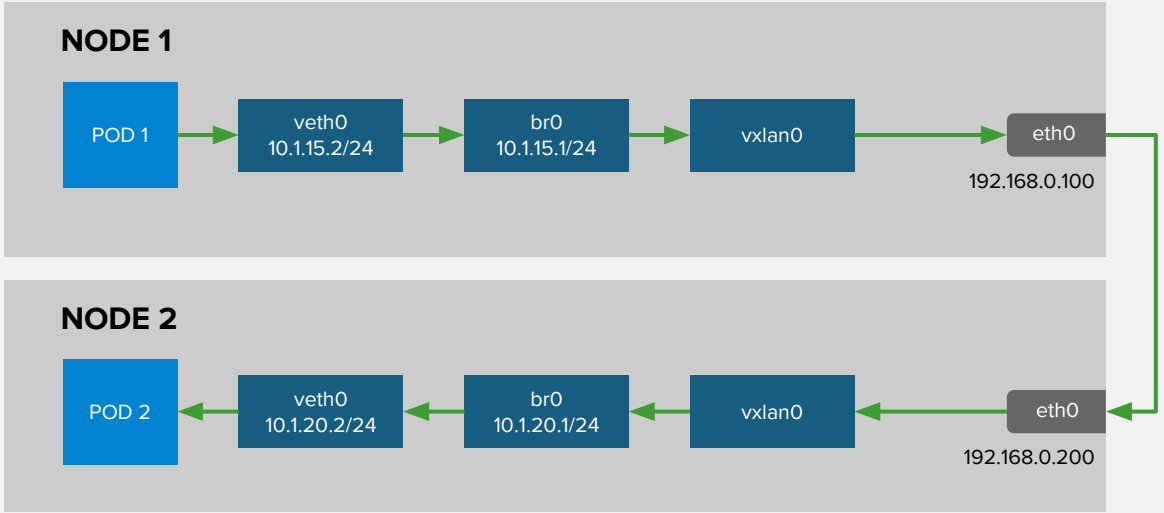
OPENSHIFT SDN - OVS PACKET FLOW

Container to Container on the Same Host



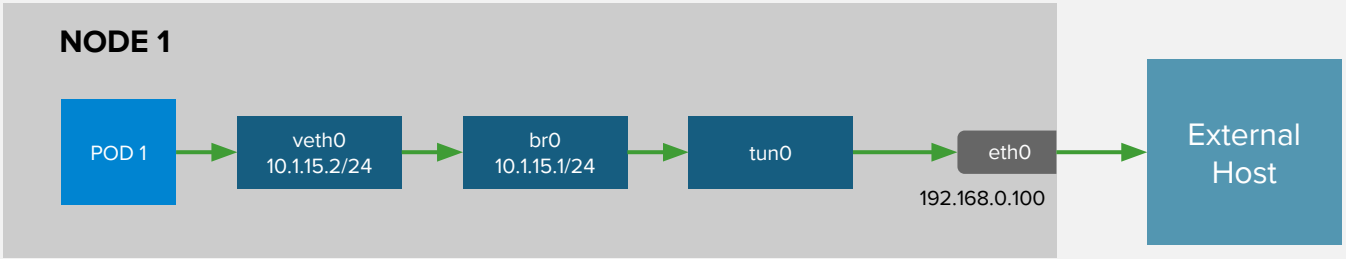
OPENSHIFT SDN - OVS PACKET FLOW

Container to Container on the Different Hosts

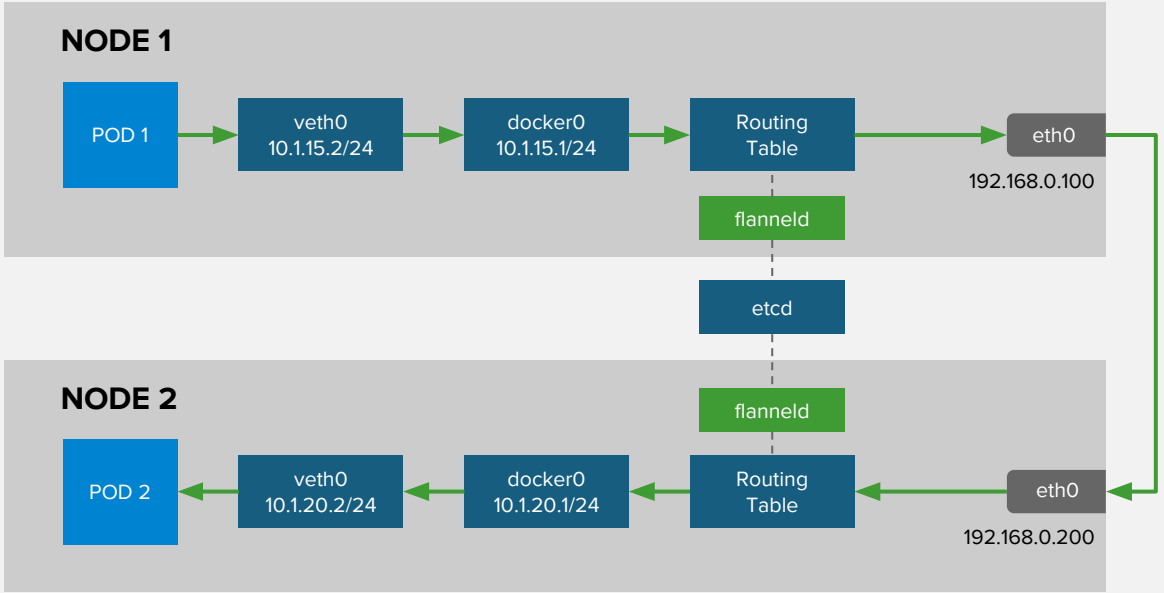


OPENSHIFT SDN - OVS PACKET FLOW

Container Connects to External Host



OPENSHIFT SDN WITH FLANNEL FOR OPENSTACK

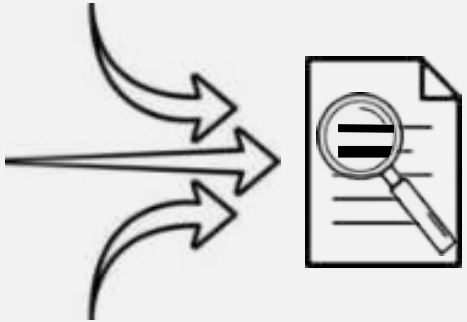


Flannel is minimally verified and is supported only and exactly as deployed in the OpenShift on OpenStack reference architecture https://access.redhat.com/articles/2743631

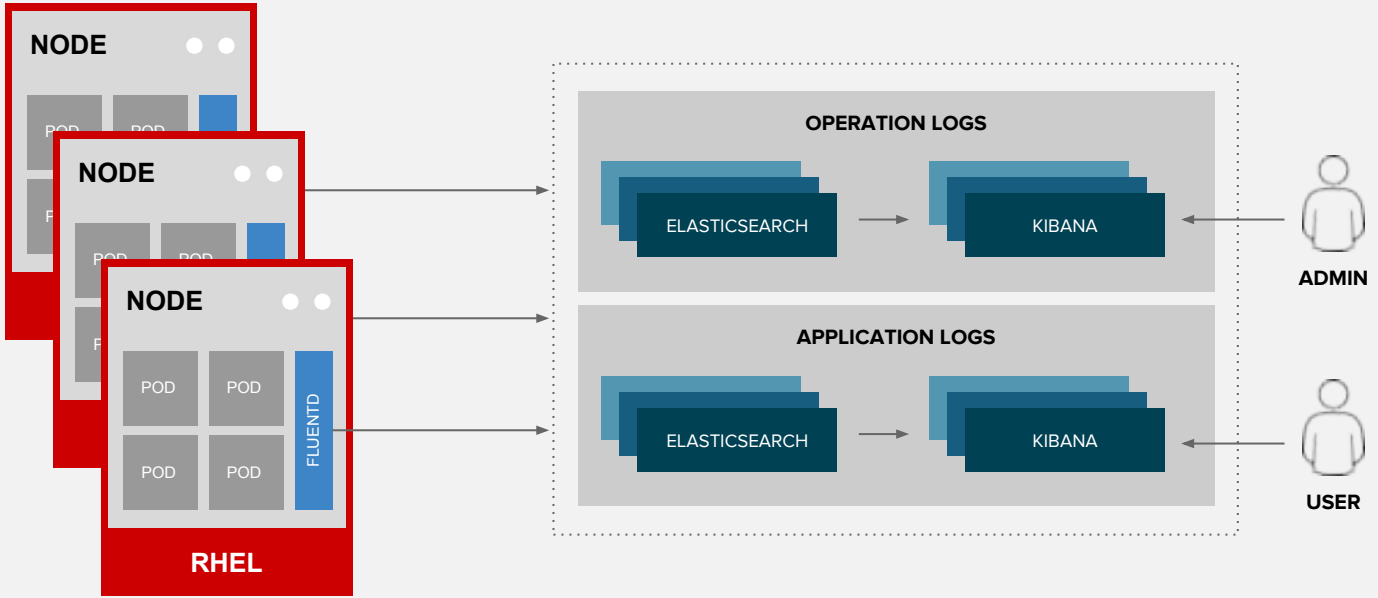
LOGGING & METRICS

CENTRAL LOG MANAGEMENT WITH EFK

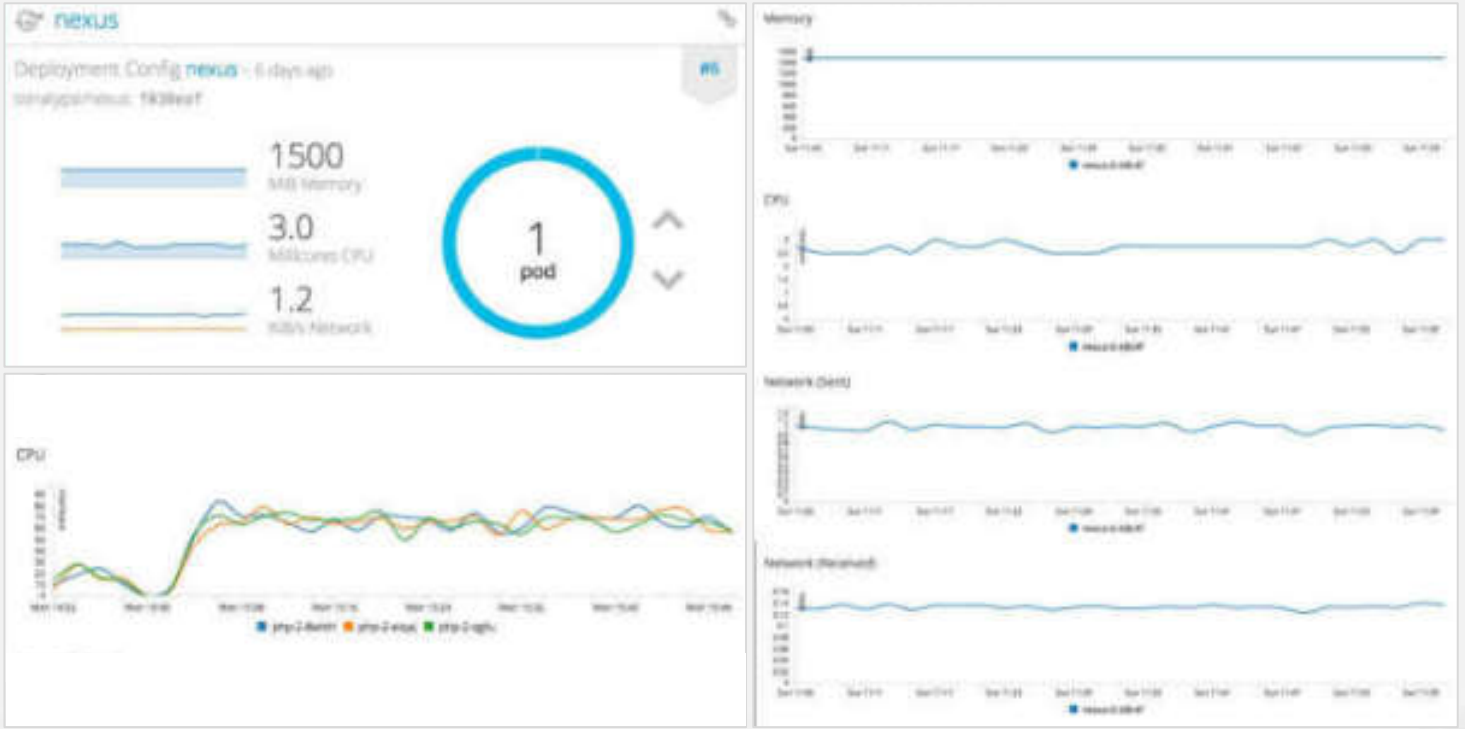
- EFK stack to aggregate logs for hosts and applications
  - **Elasticsearch:** an object store to store all logs
  - **Fluentd:** gathers logs and sends to Elasticsearch.
  - **Kibana:** A web UI for Elasticsearch.
- Access control
  - Cluster administrators can view all logs
  - Users can only view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



CENTRAL LOG MANAGEMENT WITH EFK

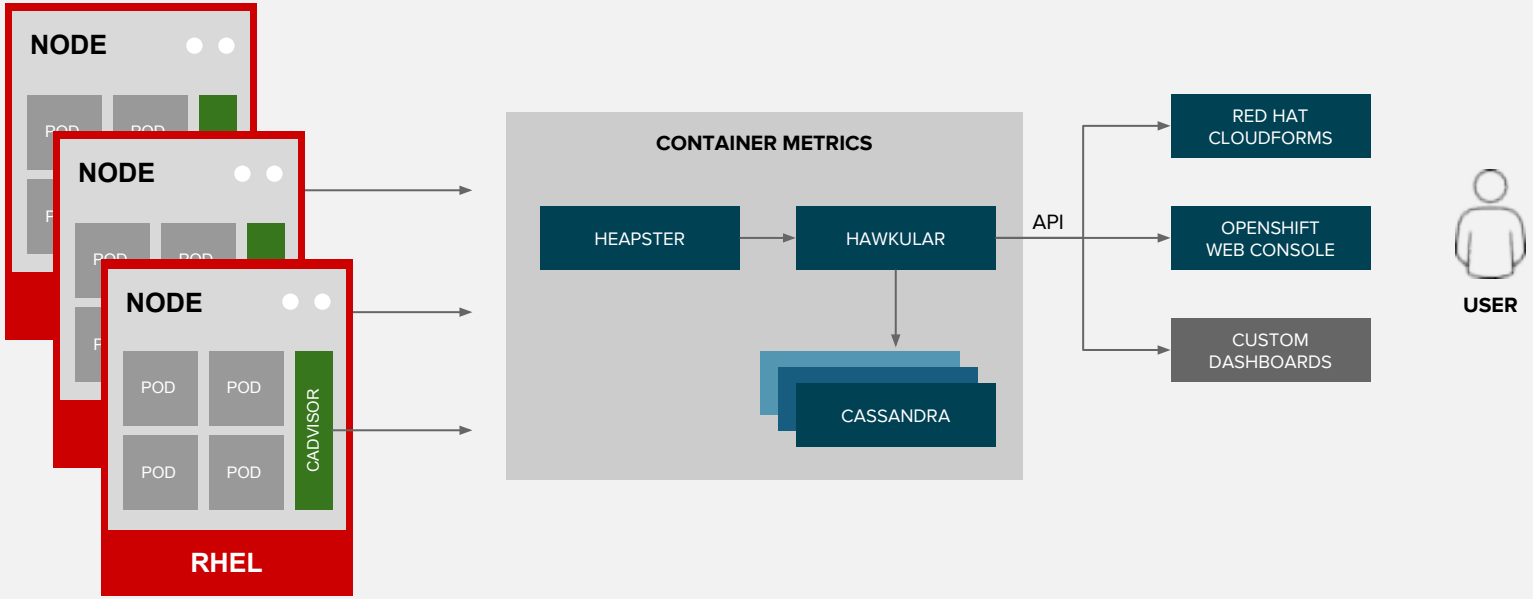


CONTAINER METRICS



redhat

CONTAINER METRICS



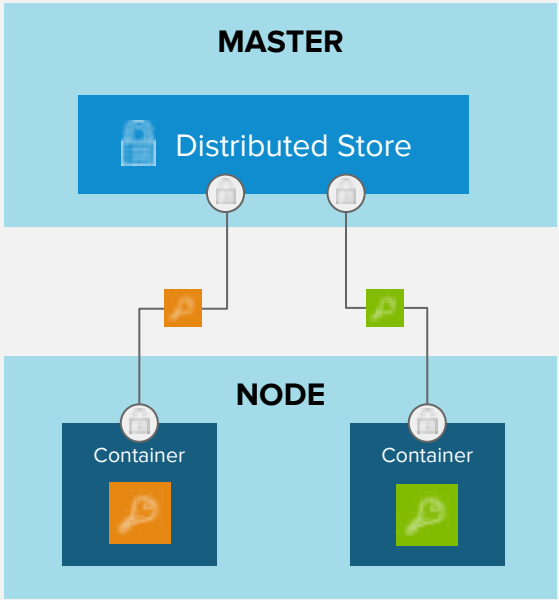
SECURITY

TEN LAYERS OF CONTAINER SECURITY

- Container Host & Multi-tenancy
- Container Platform
- Network Isolation
- Container Registry
- Storage
- Federated Clusters
- API Management
- Deploying Container
- Container Content
- 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

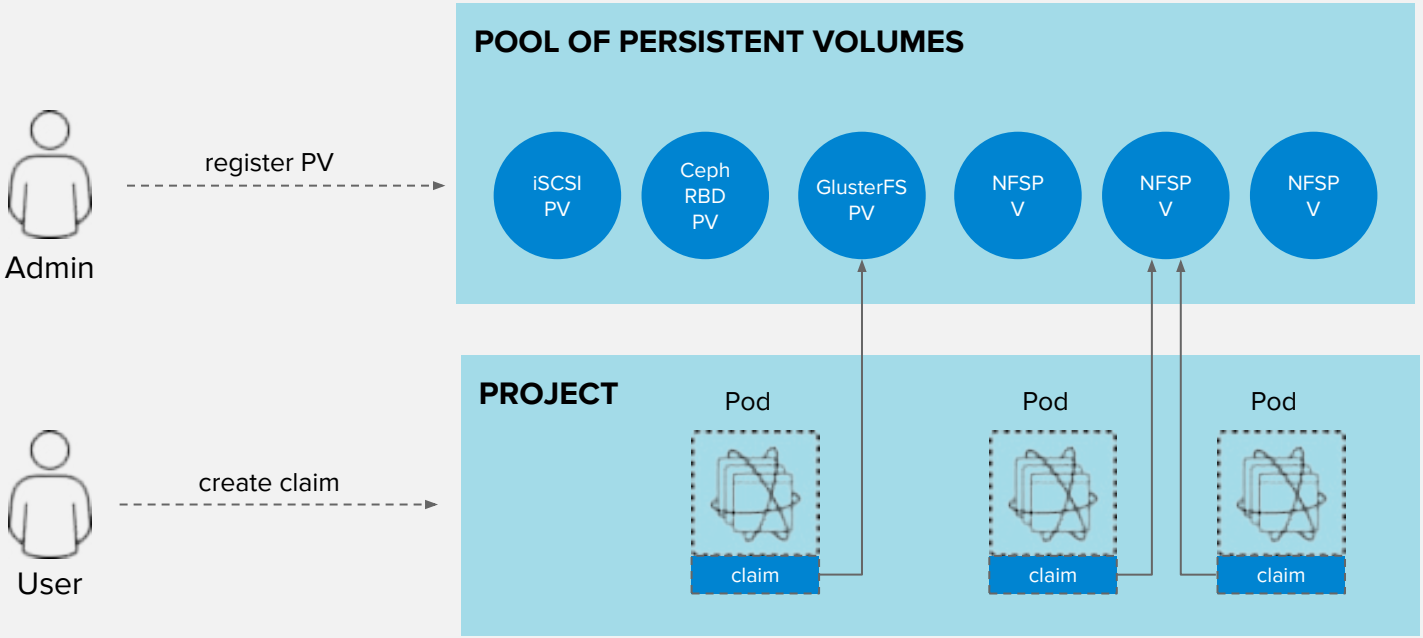


PERSISTENT STORAGE

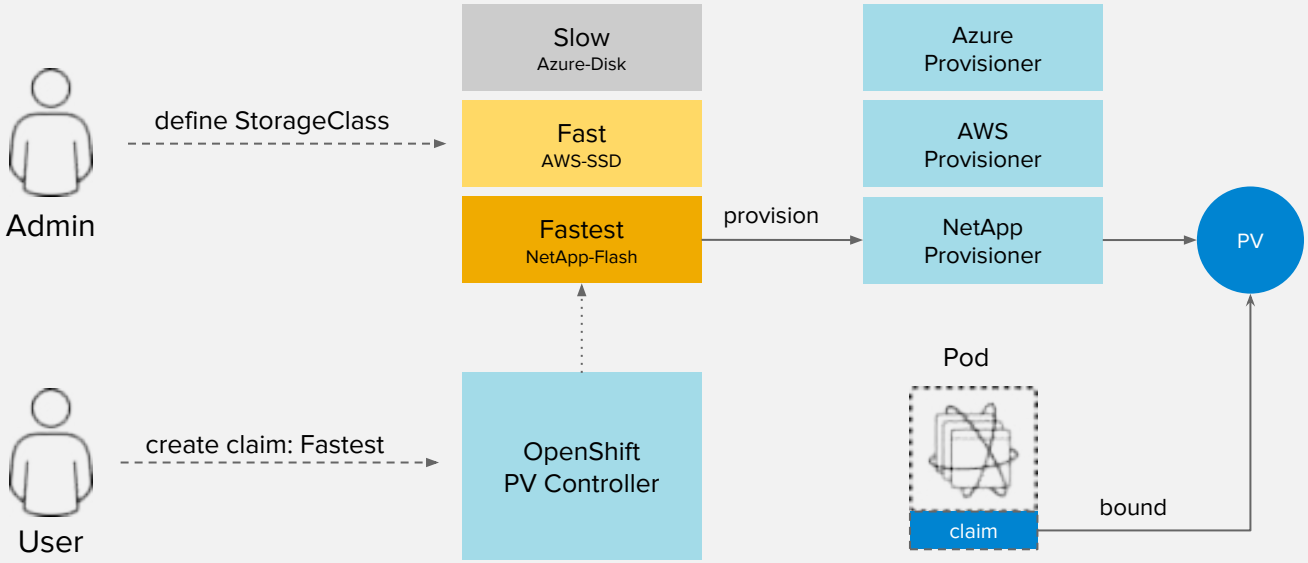
PERSISTENT STORAGE

- 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

PERSISTENT STORAGE

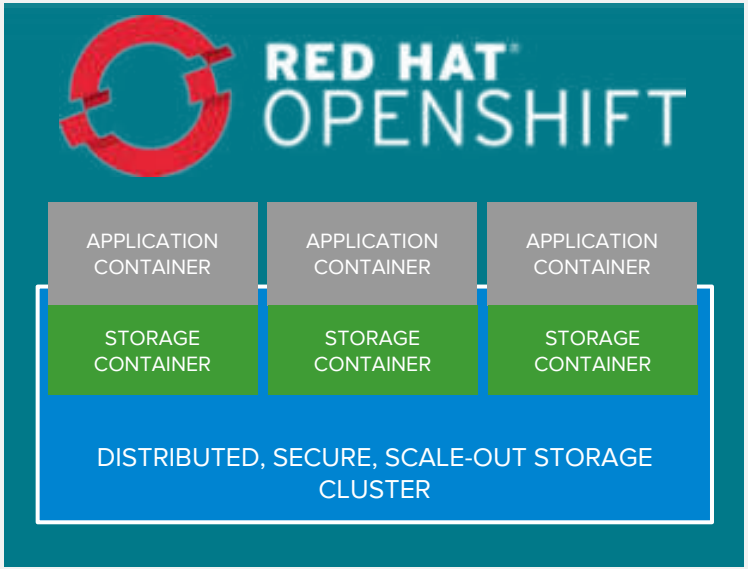


DYNAMIC VOLUME PROVISIONING

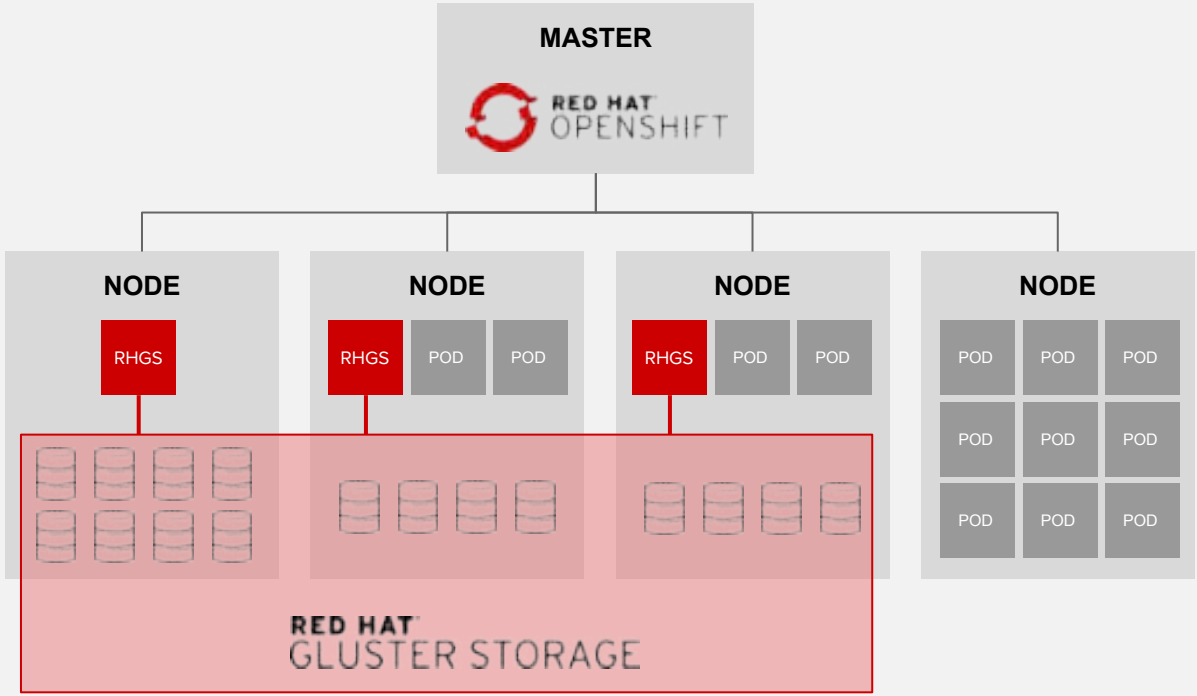


CONTAINER-NATIVE STORAGE

- 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



redhat

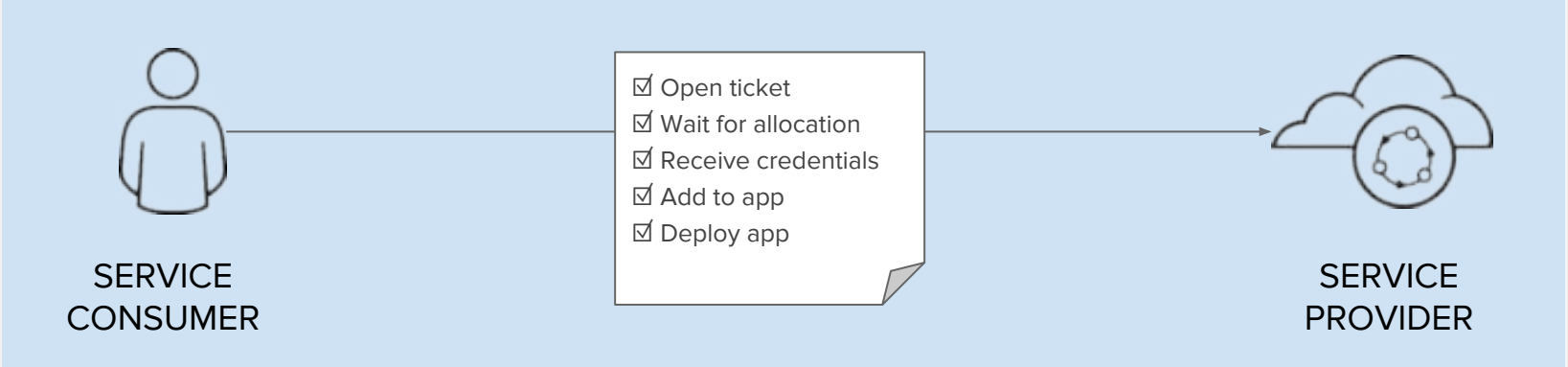
SERVICE BROKER

OPEN SERVICE BROKER API

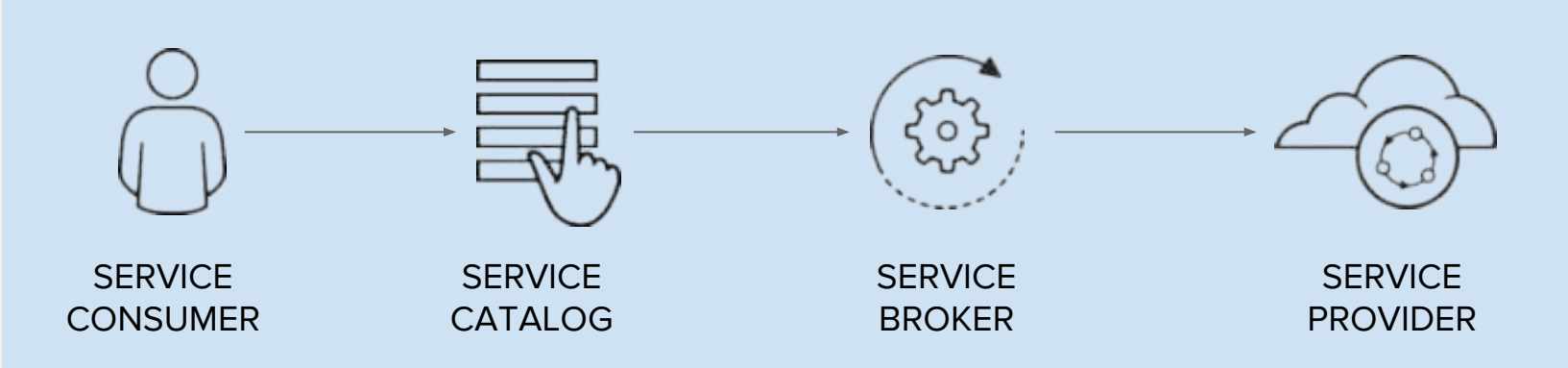
- Born out of Cloud Foundry Foundation
- Standard way to deliver services to apps running on OpenShift, Kubernetes, etc
- A collaboration between multiple vendors
- Integrated with OpenShift and Kubernetes
- Release Timeline
  - OCP 3.6 Tech Preview
  - OCP 3.7 GA

RED HAT
PIVOTAL
IBM
SAP
GOOGLE
FUJITSU

CONSUMING SERVICES

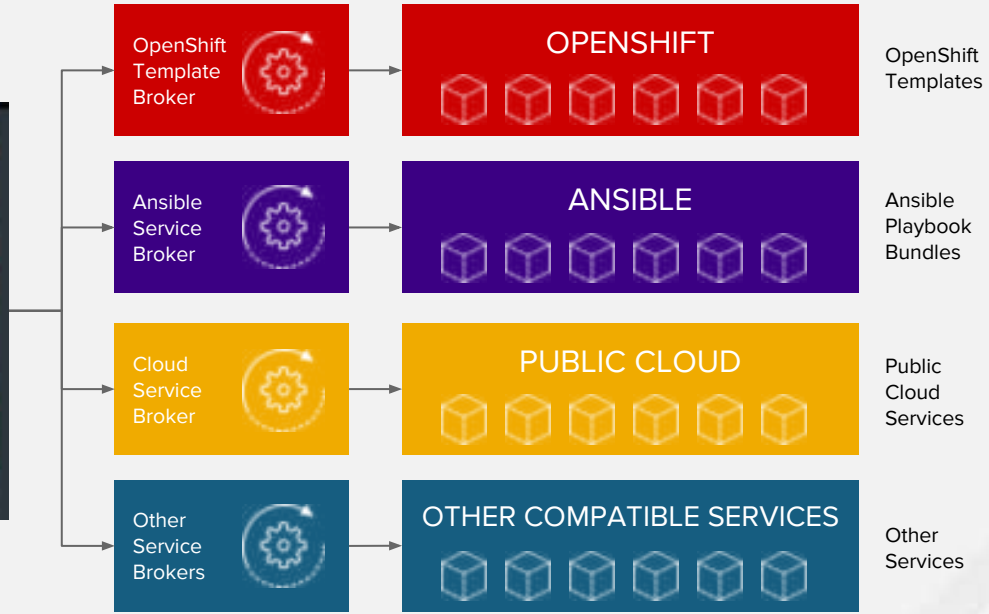


CONSUMING SERVICES WITH OPEN SERVICE BROKER API



OPENSHIFT SERVICE CATALOG

OPENSHIFT SERVICE CATALOG
(TECH PREVIEW)



redhat

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

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

Time Stamp	Type	Name	Event Type	Severity	Message
01/02/18 07:23:19 UTC	Cluster / Deployment Role	Raleigh	Memory Usage	1	Memory - Peak Aggregate Used for Child VMs for Collected Intervals (MB) is projected to reach 765.6 GB (100% of Memory Max Total)
11/01/17 06:18:52 UTC	Cluster / Deployment Role	Raleigh	Memory Usage	1	Memory - Peak Aggregate Used for Child VMs for Collected Intervals (MB) is projected to reach 689 GB (90% of Memory Max Total)
07/31/17 04:42:25 UTC	Cluster / Deployment Role	Raleigh	Memory Usage	2	Memory - Peak Aggregate Used for Child VMs for Collected Intervals (MB) is projected to reach 574.2 GB (75% of Memory Max Total)
02/28/17 02:01:39 UTC	Cluster / Deployment Role	Raleigh	Memory Usage	3	Memory - Peak Aggregate Used for Child VMs for Collected Intervals (MB) is projected to reach 382.8 GB (50% of Memory Max Total)

Normal Operating Ranges (up to 30 days' data)

	Max	High	Average	Low
CPU	345.30 MHz	105.74 MHz	663.99 MHz	522.22 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	83.00%	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	7908 MB	35.5%	4380 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)

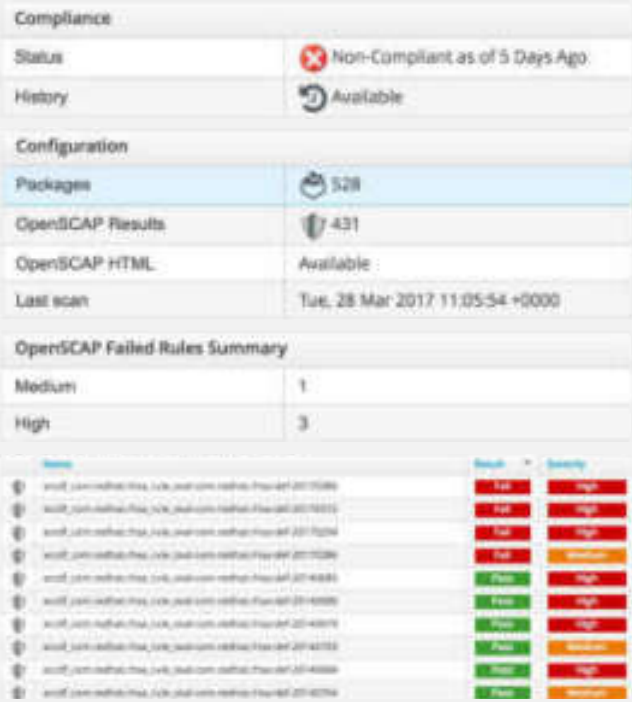
	Current	Recommended	% Savings	Savings
Processors	4	1	75.0%	3
Memory	12288 MB	7656 MB	38.2%	4632 MB

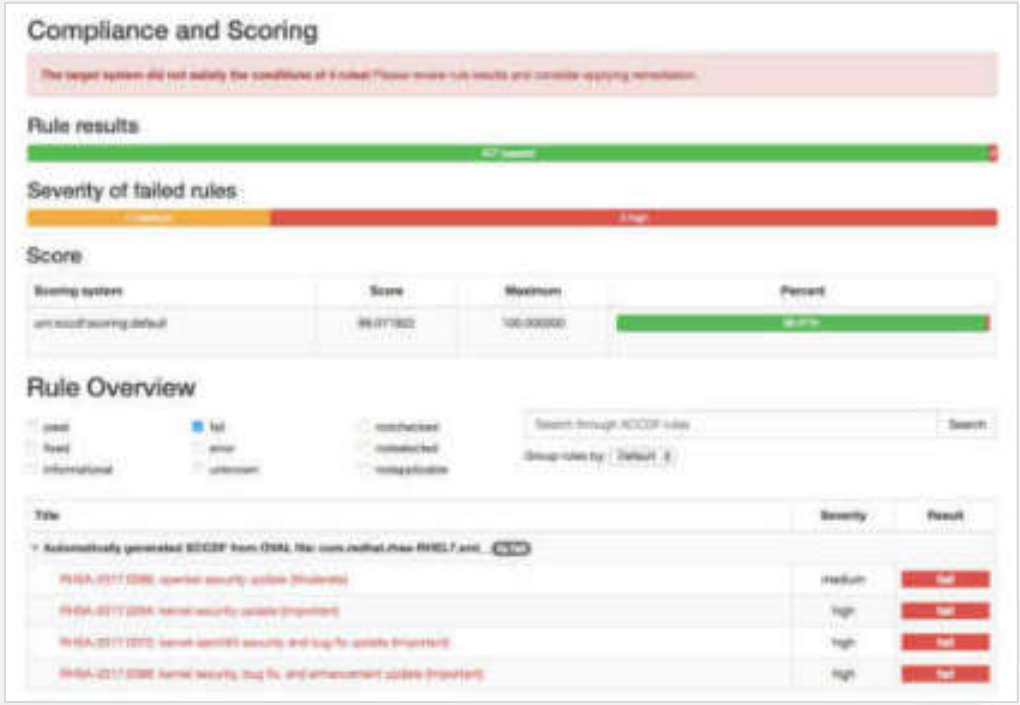
redhat

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





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

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	b8f35aee-e974-11e6-89d9-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	ifixed	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
04/17/2017					
Totals:			$168.07	$1,984.40	$2,498.43
All Rows					
Totals:			$168.07	$1,984.40	$2,498.43

REFERENCE ARCHITECTURES

REFERENCE ARCHITECTURES

OpenShift on VMware vCenter
https://access.redhat.com/articles/2745171

OpenShift on Red Hat OpenStack Platform
https://access.redhat.com/articles/2743631

OpenShift on Amazon Web Services
https://access.redhat.com/articles/2623521

OpenShift on Google Cloud Platform
https://access.redhat.com/articles/2751521

OpenShift on Microsoft Azure
https://access.redhat.com/articles/3030691

Deploying an OpenShift Distributed Architecture
https://access.redhat.com/articles/1609803

OpenShift Architecture and Deployment Guide
https://access.redhat.com/articles/1755133

OpenShift Scaling, Performance, and Capacity Planning
https://access.redhat.com/articles/2191731

Application Release Strategies with OpenShift
https://access.redhat.com/articles/2897391

Building Polyglot Microservices on OpenShift
https://access.redhat.com/articles/2893381

Building JBoss EAP 6 Microservices on OpenShift
https://access.redhat.com/articles/2094731

Building JBoss EAP 7 Microservices on OpenShift
https://access.redhat.com/articles/2407801

Business Process Management with JBoss BPMS on OpenShift
https://access.redhat.com/articles/2893421

Build and Deployment of Java Applications on OpenShift
https://access.redhat.com/articles/3016691

JFrog Artifactory on OpenShift Container Platform
https://access.redhat.com/articles/3049611

BUILD AND DEPLOY CONTAINER IMAGES

BUILD AND DEPLOY CONTAINER IMAGES

DEPLOY YOUR SOURCE CODE

DEPLOY YOUR APP BINARY

DEPLOY YOUR CONTAINER IMAGE

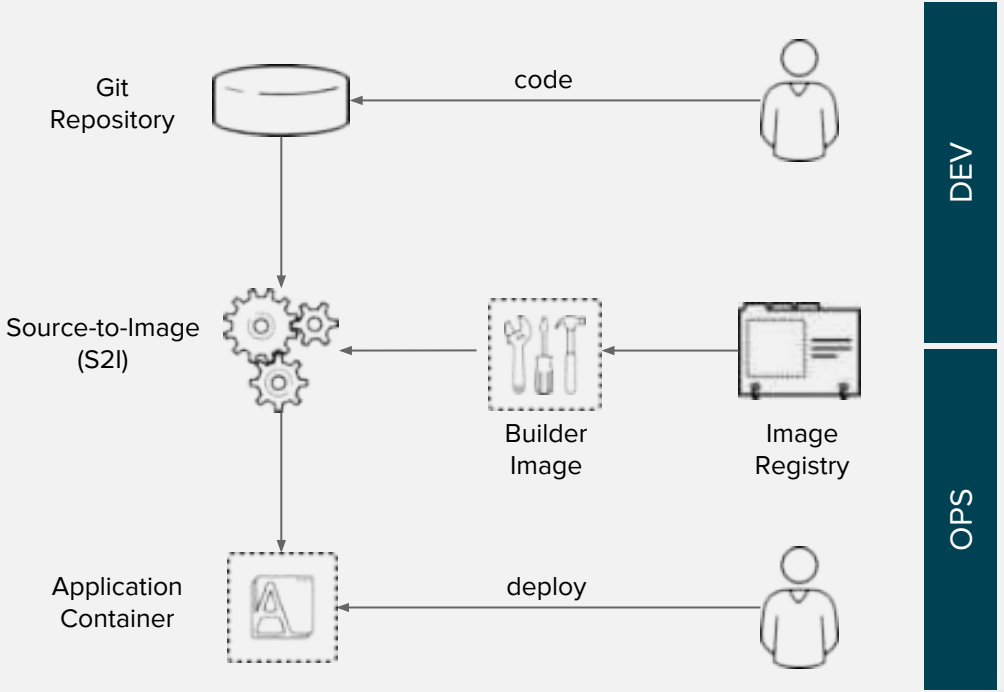
DEPLOY YOUR SOURCE CODE

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

CODE

BUILD

DEPLOY

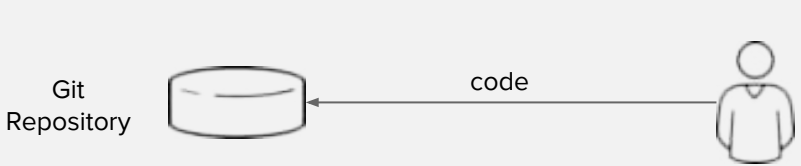


redhat

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

CODE

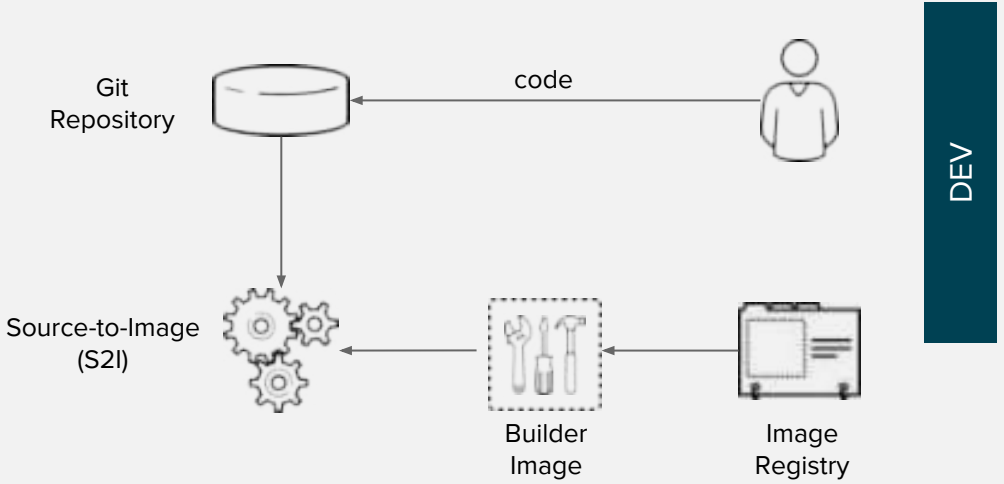
Developers write code using existing development tools such as Maven, NPM, Bower, PIP, Dockerfile and Git and then access the OpenShift Web, CLI or IDE to create an app from the code



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

BUILD

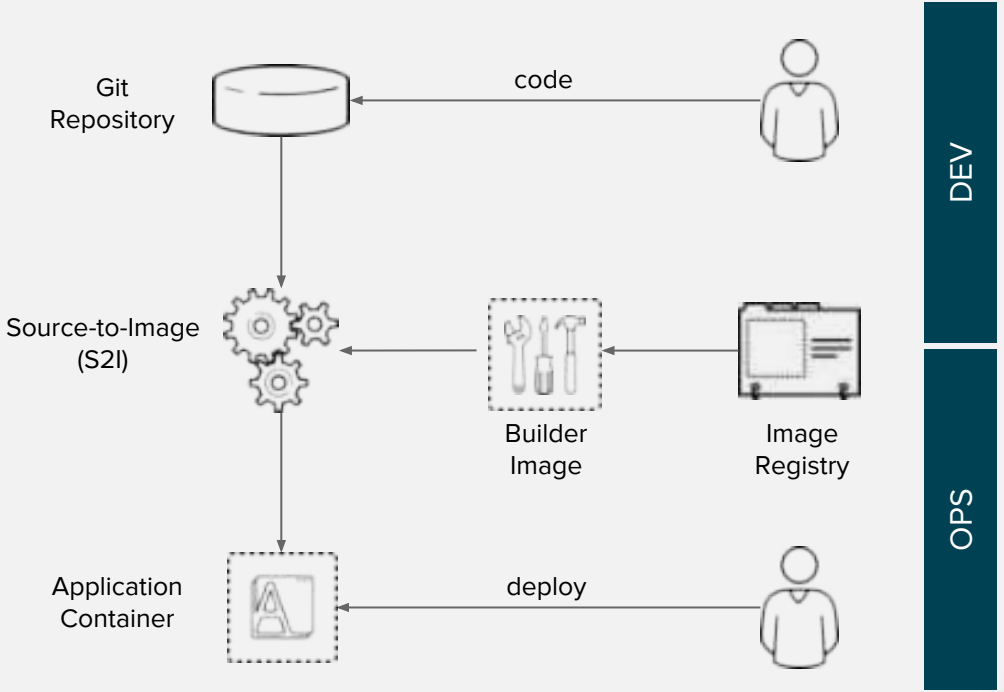
S2I combines source code with a builder image (language and application runtimes) and stores the resulting application image in the image registry



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

DEPLOY

OpenShift automates the deployment of application containers across multiple hosts via the Kubernetes. Users can trigger deployments, rollback, configure A/B or other custom deployments



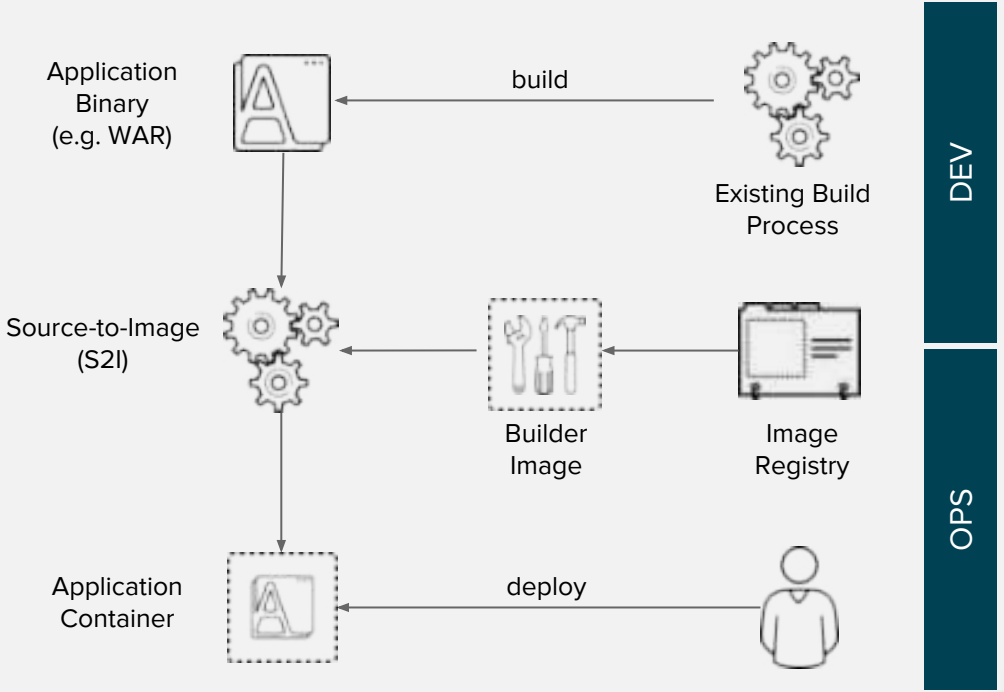
DEPLOY YOUR APP BINARY

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

BUILD APP

BUILD IMAGE

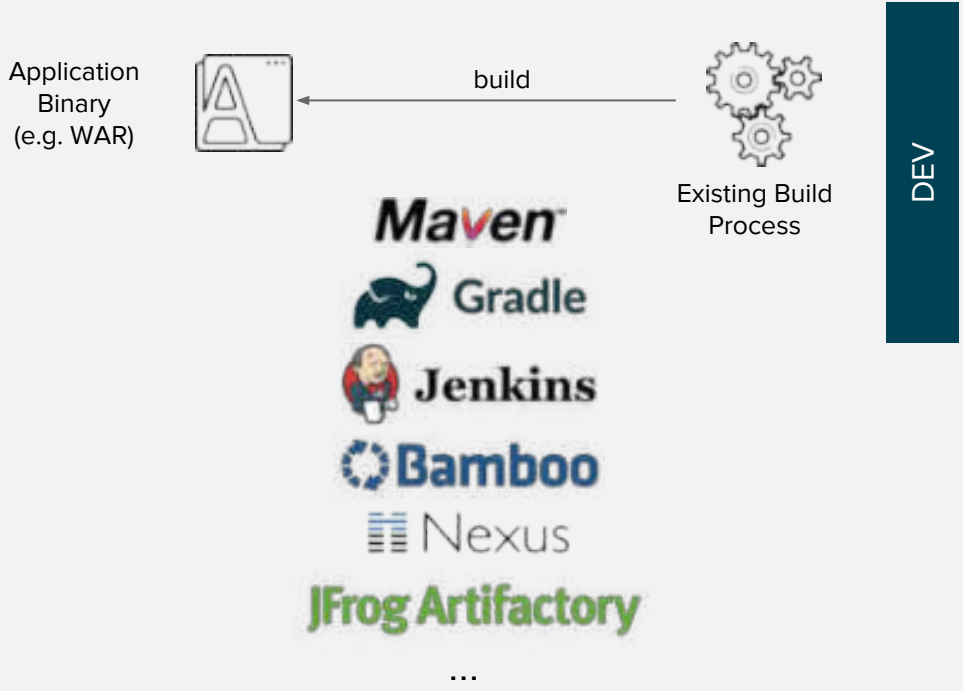
DEPLOY



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

BUILD APP

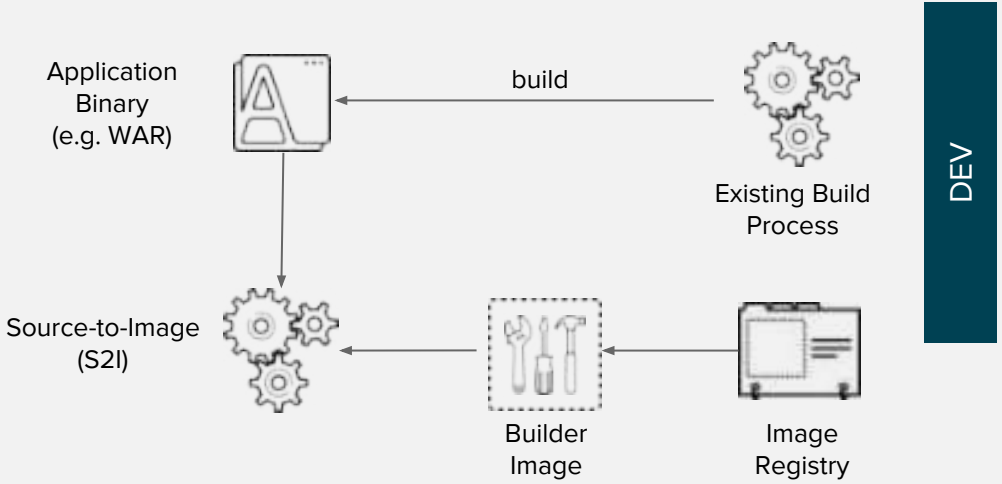
Developers use the existing build process and tools (e.g. Maven, Gradle, Jenkins, Nexus) to build the app binaries (e.g. JAR, WAR, EAR) and use OpenShift CLI to create an app from the app binaries



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

BUILD IMAGE

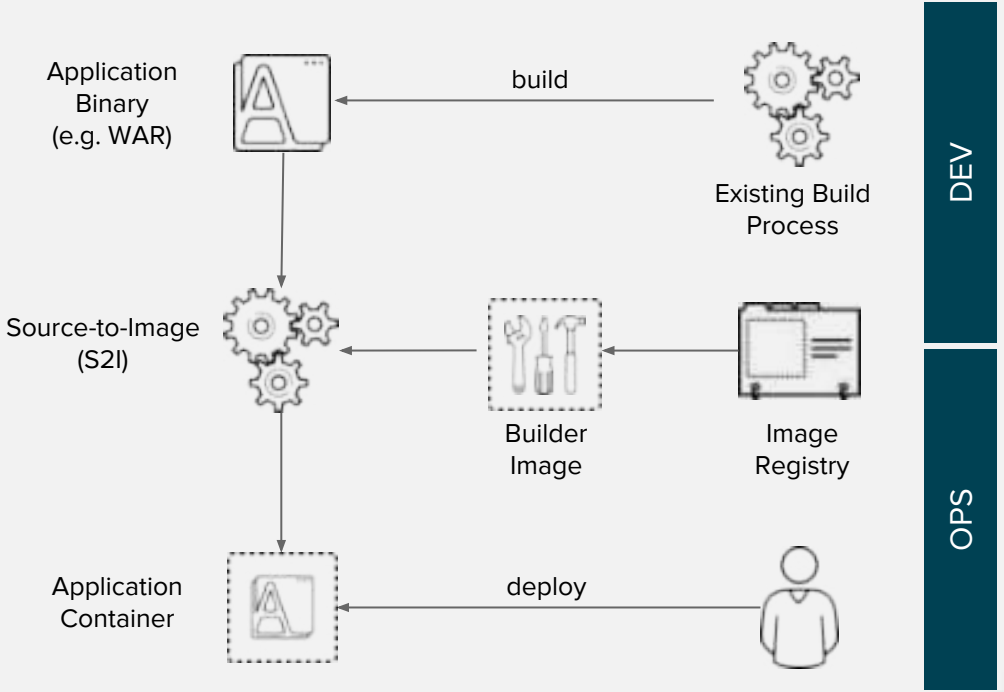
S2I combines app binaries (e.g. JAR, WAR, EAR) with a builder image (language and application runtimes) and stores the resulting application image in the image registry



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

DEPLOY

OpenShift automates the deployment of application containers across multiple hosts via the Kubernetes. Users can trigger deployments, rollback, configure A/B or other custom deployments



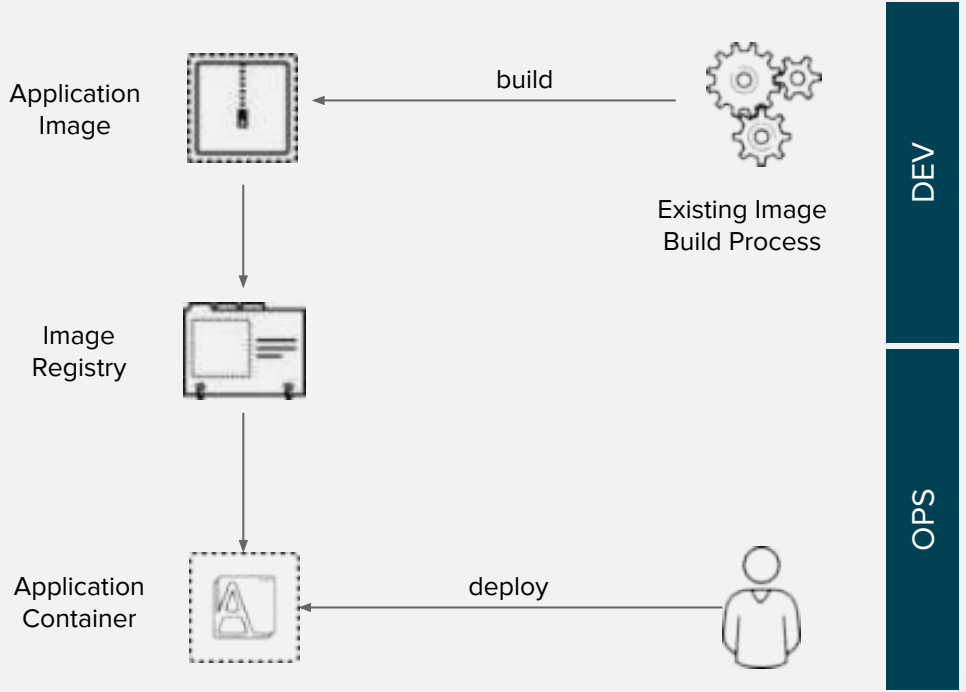
DEPLOY YOUR CONTAINER IMAGE

DEPLOY DOCKER IMAGE

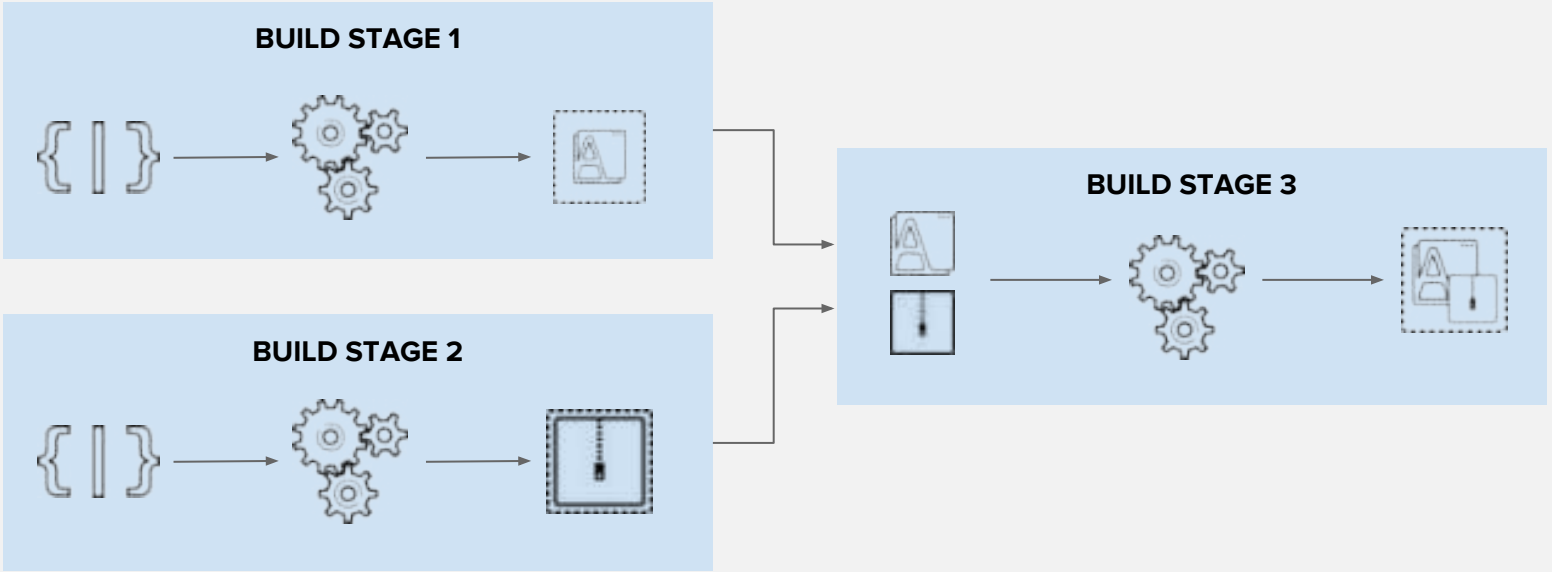
BUILD

App images are built using an existing image build process. OpenShift automates the deployment of app containers across multiple hosts via the Kubernetes. Users can trigger deployments, rollback, configure A/B, etc

DEPLOY

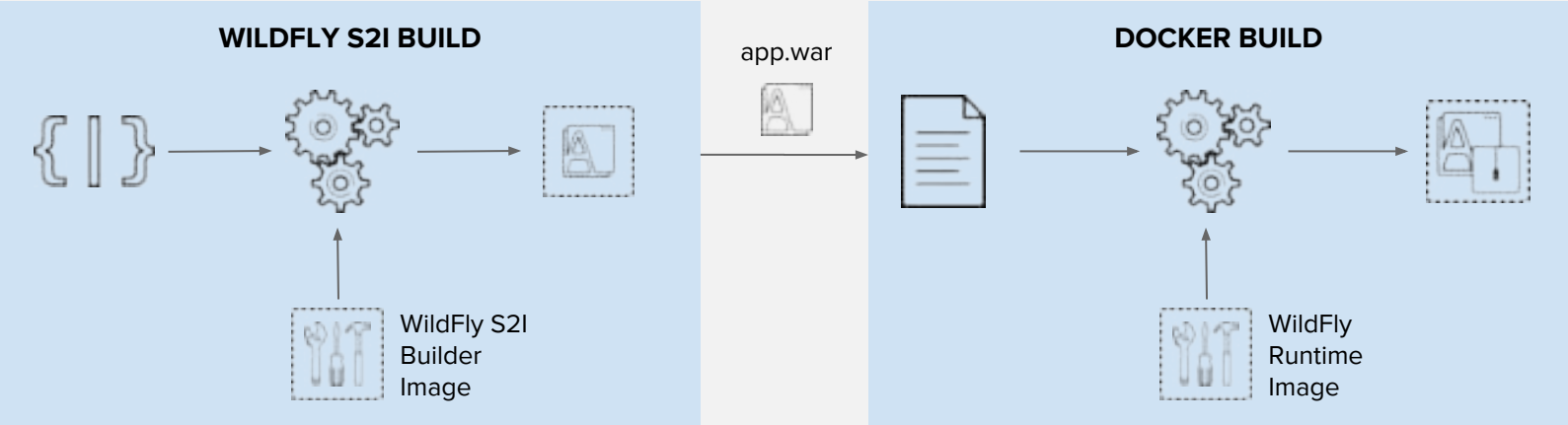


BUILD IMAGES IN MULTIPLE STAGES



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

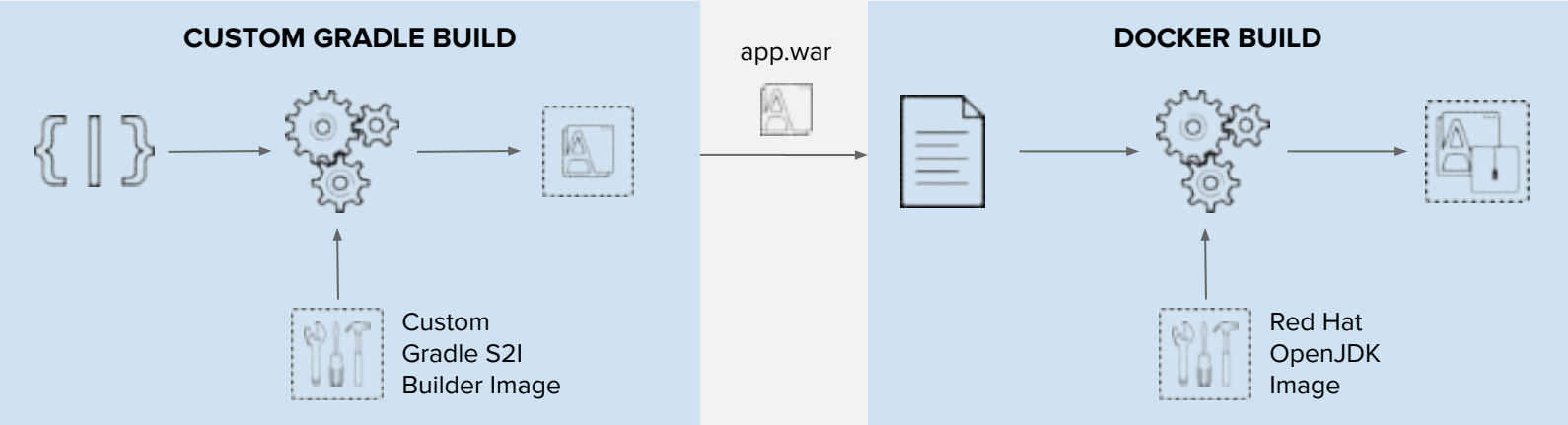
Use Source-to-Image to build app binaries and deploy on lean vanilla runtimes



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

EXAMPLE: USE ANY BUILD TOOL WITH OFFICIAL RUNTIME IMAGES

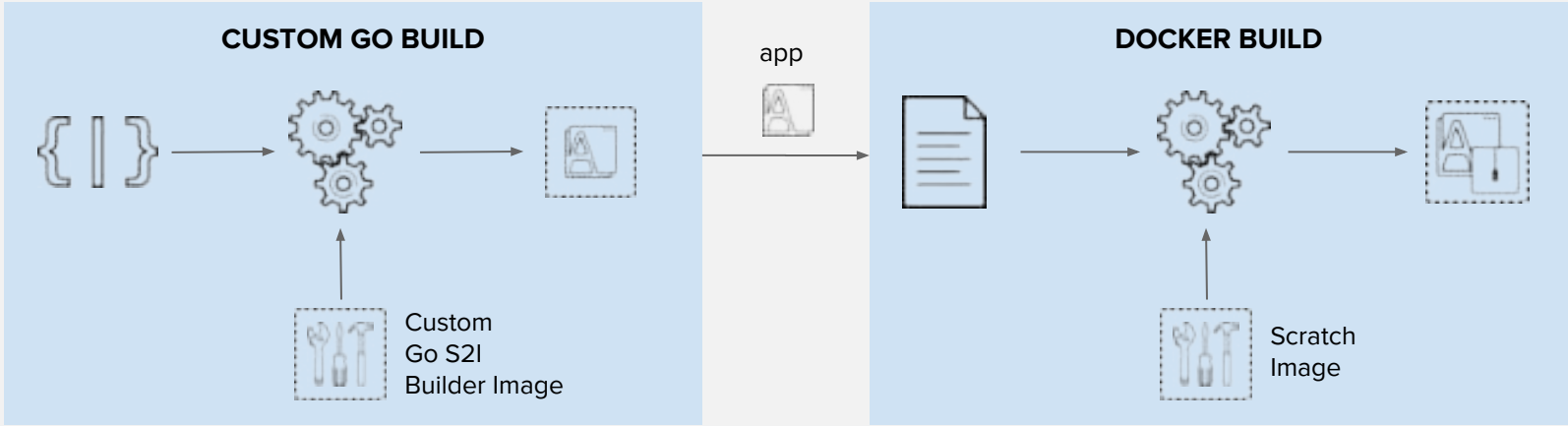
Use your choice of build tool like Gradle and deploy to official images like the JDK image



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

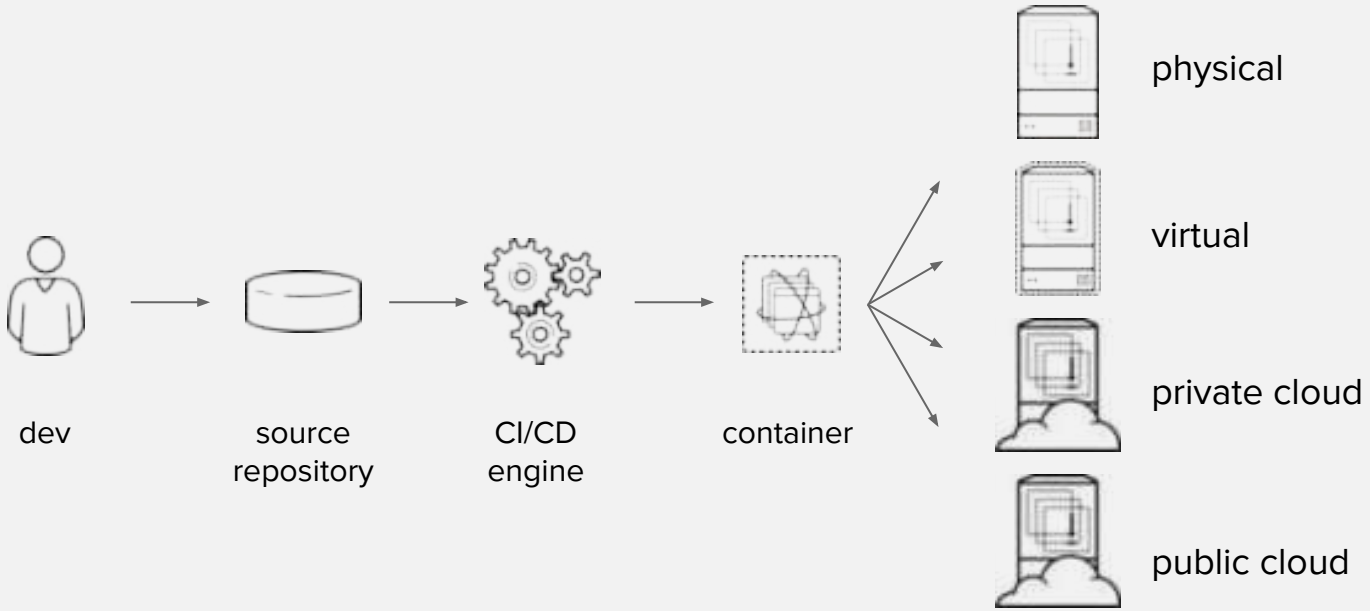
BUILDS

- 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

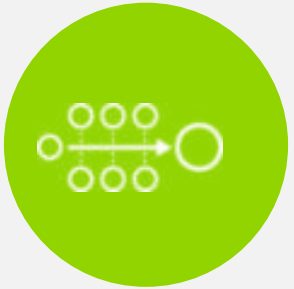


DevOps Excellence Awards 2017

OPENSHIFT LOVES CI/CD

JENKINS-AS-A SERVICE ON OPENSHIFT

HYBRID JENKINS INFRA WITH OPENSHIFT

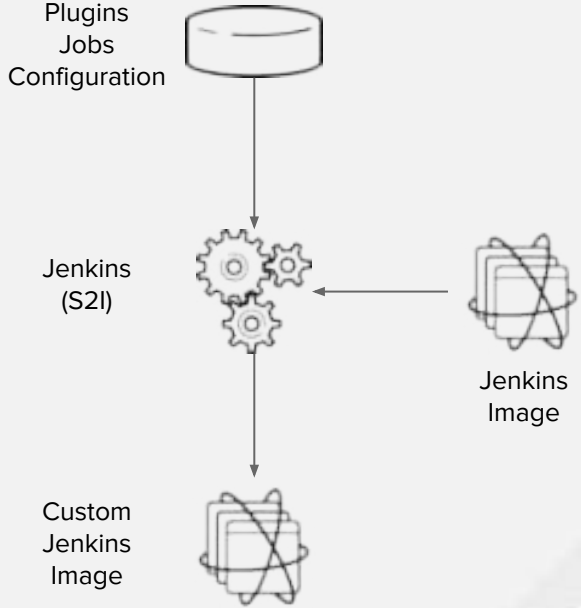


EXISTING CI/CD DEPLOY TO OPENSHIFT

redhat

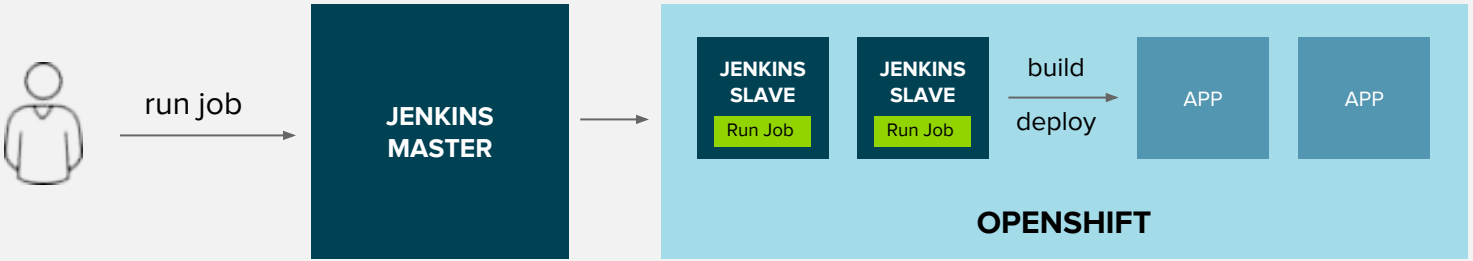
JENKINS-AS-A-SERVICE ON OPENSHIFT

- 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



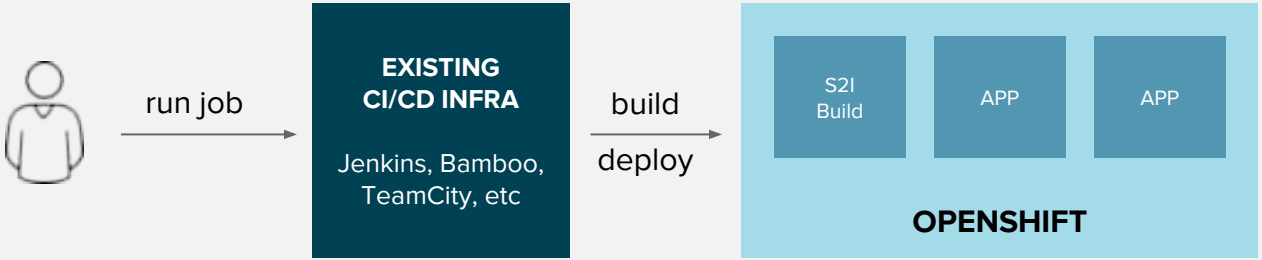
HYBRID JENKINS INFRA WITH OPENSHIFT

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

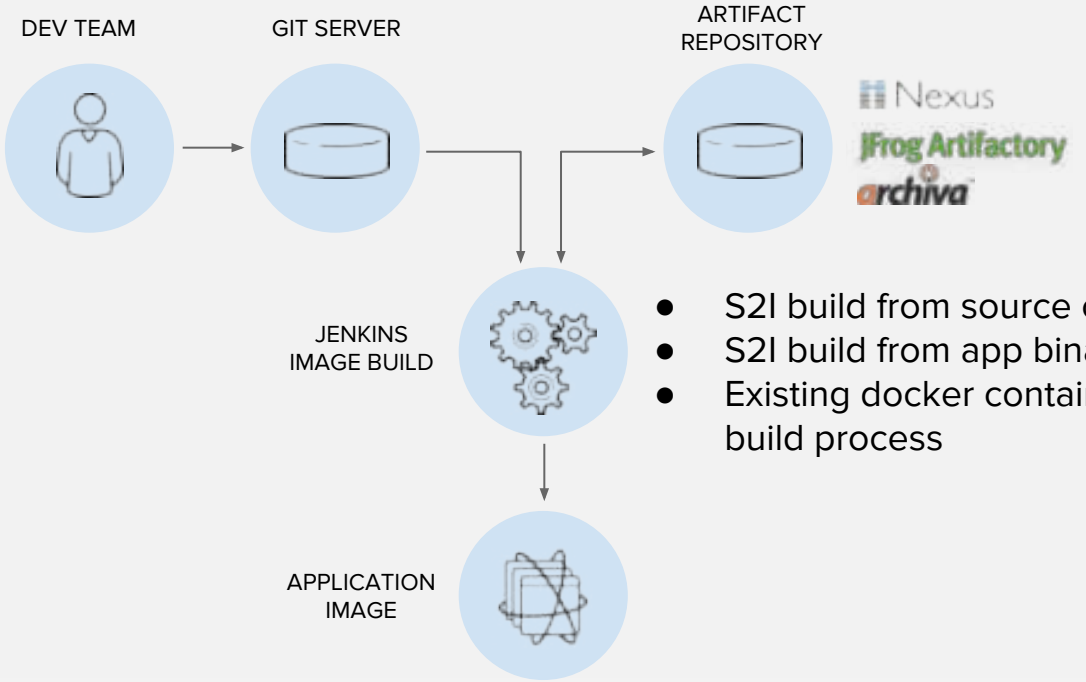
```
apiVersion: v1
kind: BuildConfig
metadata:
  name: app-pipeline
spec:
  strategy:
    type: JenkinsPipeline
    jenkinsPipelineStrategy:
      jenkinsfile: |-
        node('maven') {
          stage('build app') {
            git url: 'https://git/app.git'
            sh "mvn package"
          }
          stage('build image') {
            sh "oc start-build app --from-file=target/app.jar
          }
          stage('deploy') {
            openshiftDeploy deploymentConfig: 'app'
          }
        }
```

Provision a Jenkins slave for running Maven

OpenShift Pipelines in Web Console

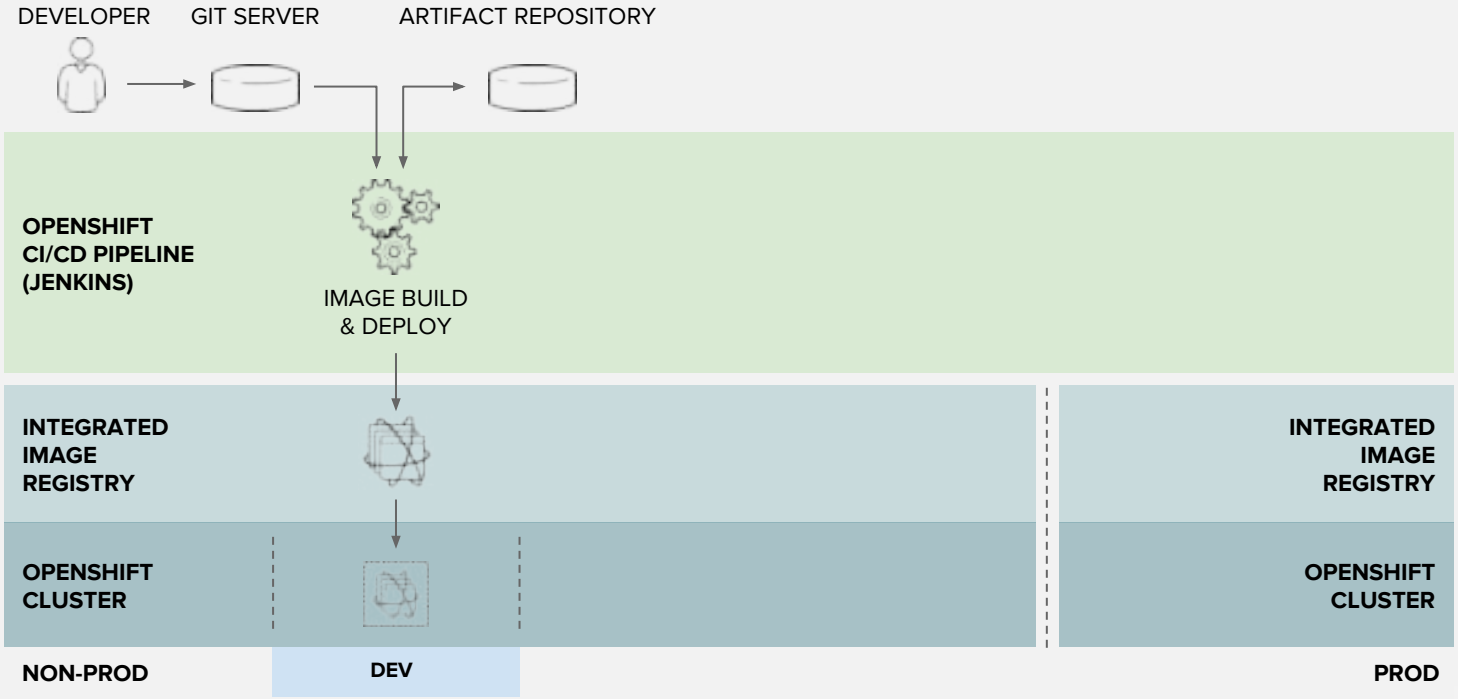
redhat

CONTINUOUS DELIVERY PIPELINE



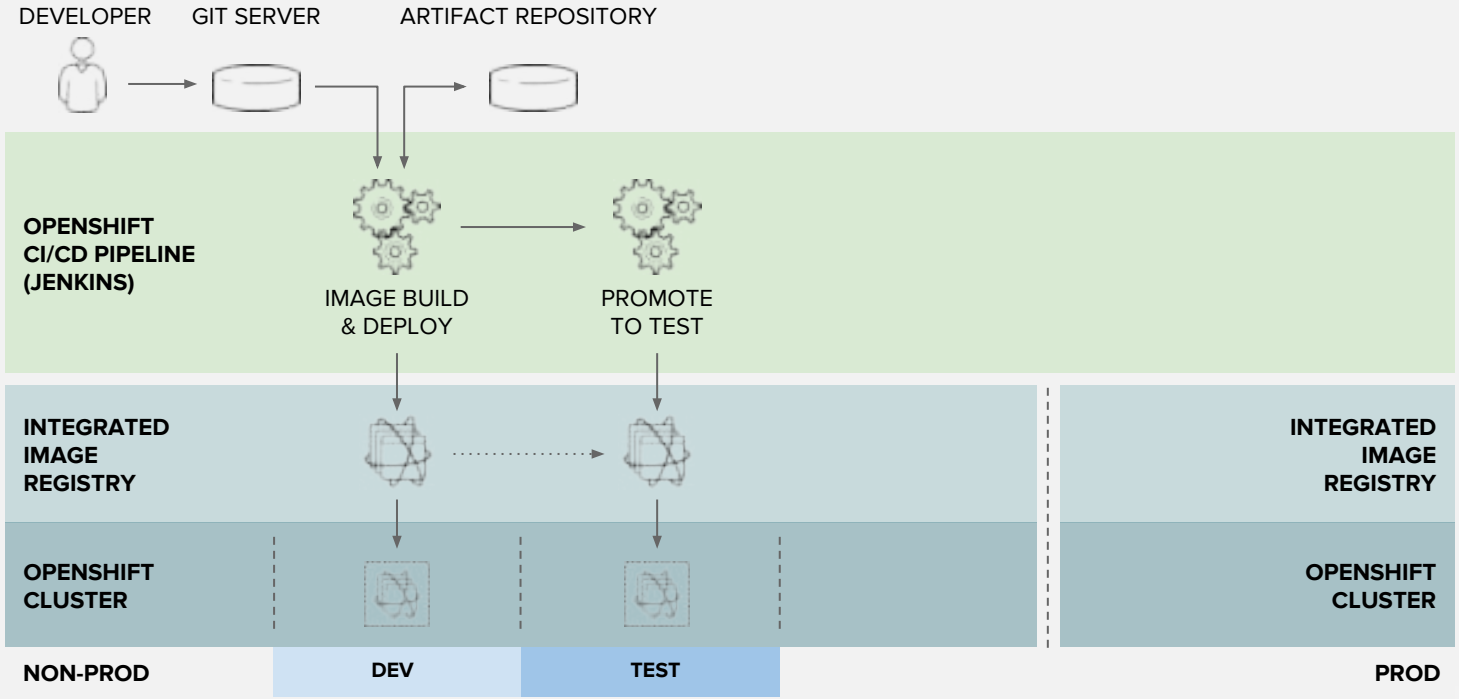
- S2I build from source code
- S2I build from app binary
- Existing docker container image build process

CONTINUOUS DELIVERY PIPELINE

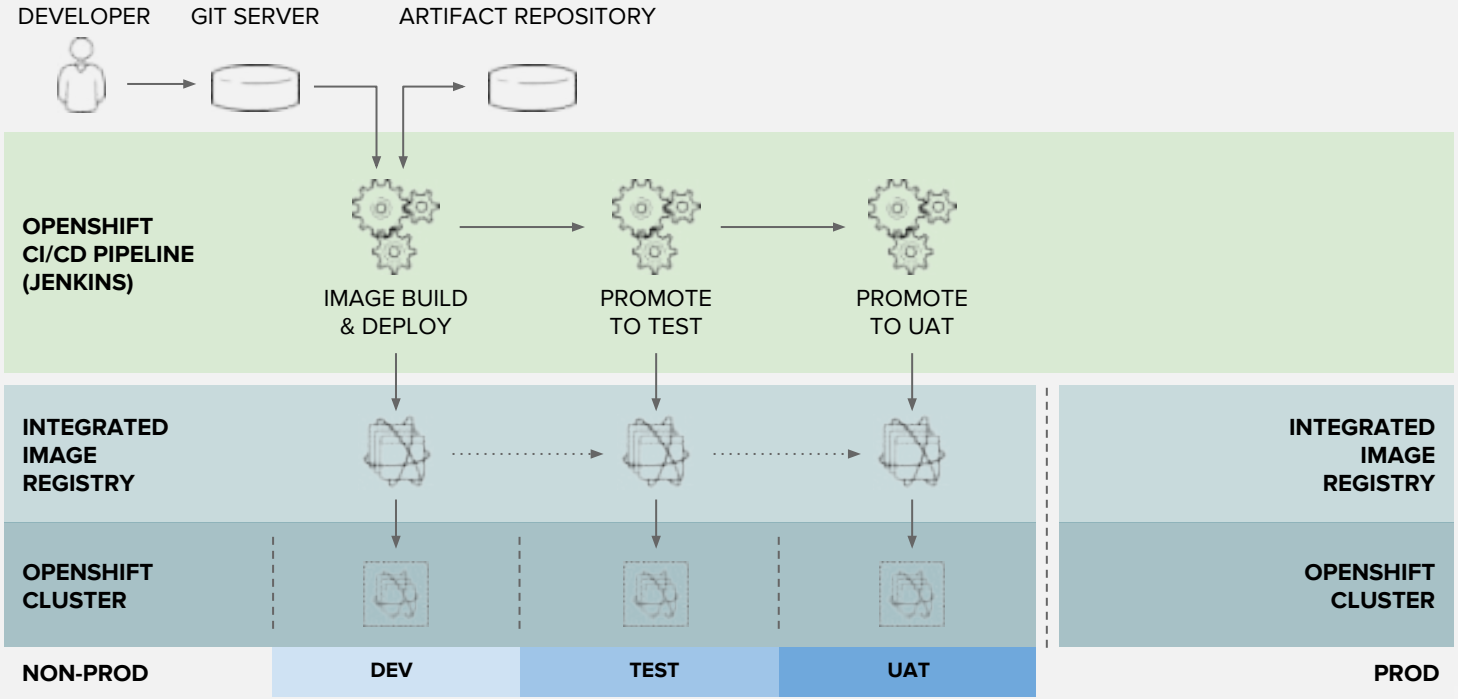


redhat

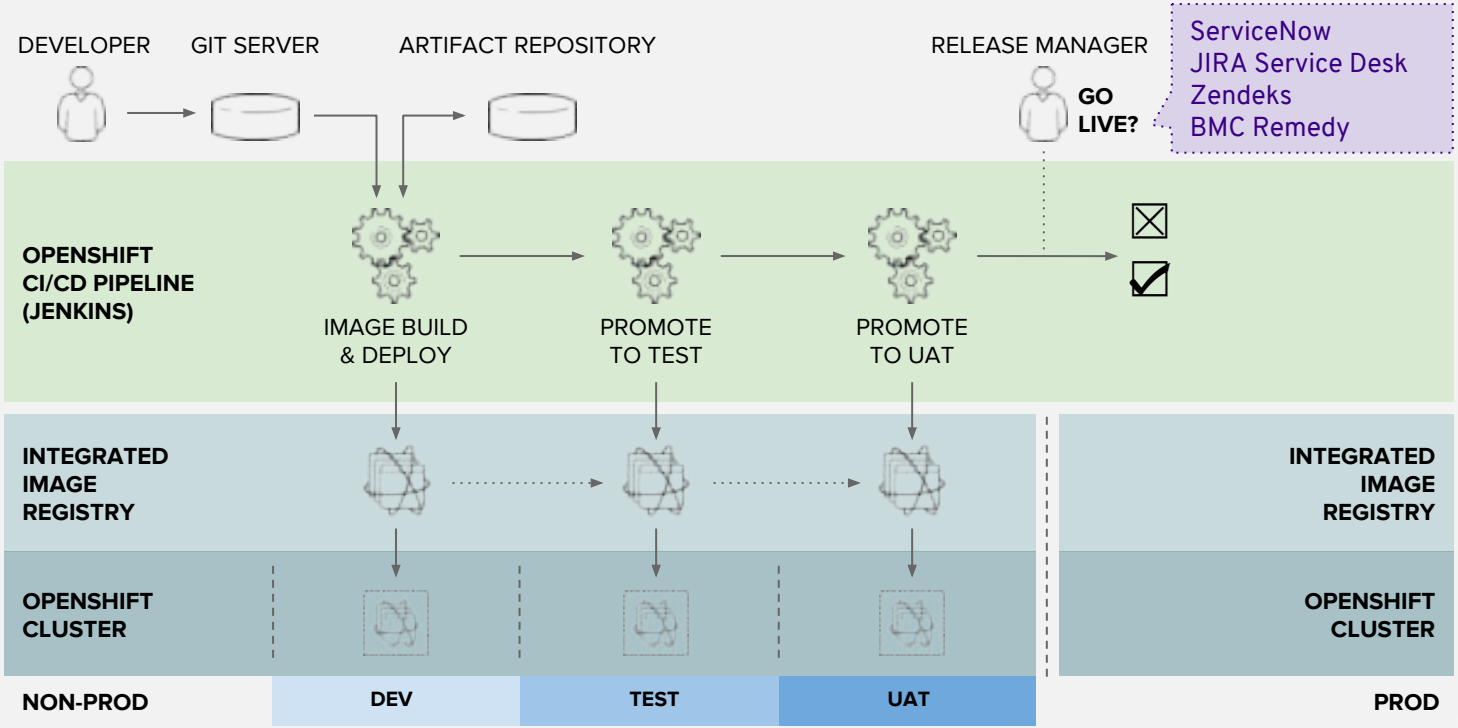
CONTINUOUS DELIVERY PIPELINE



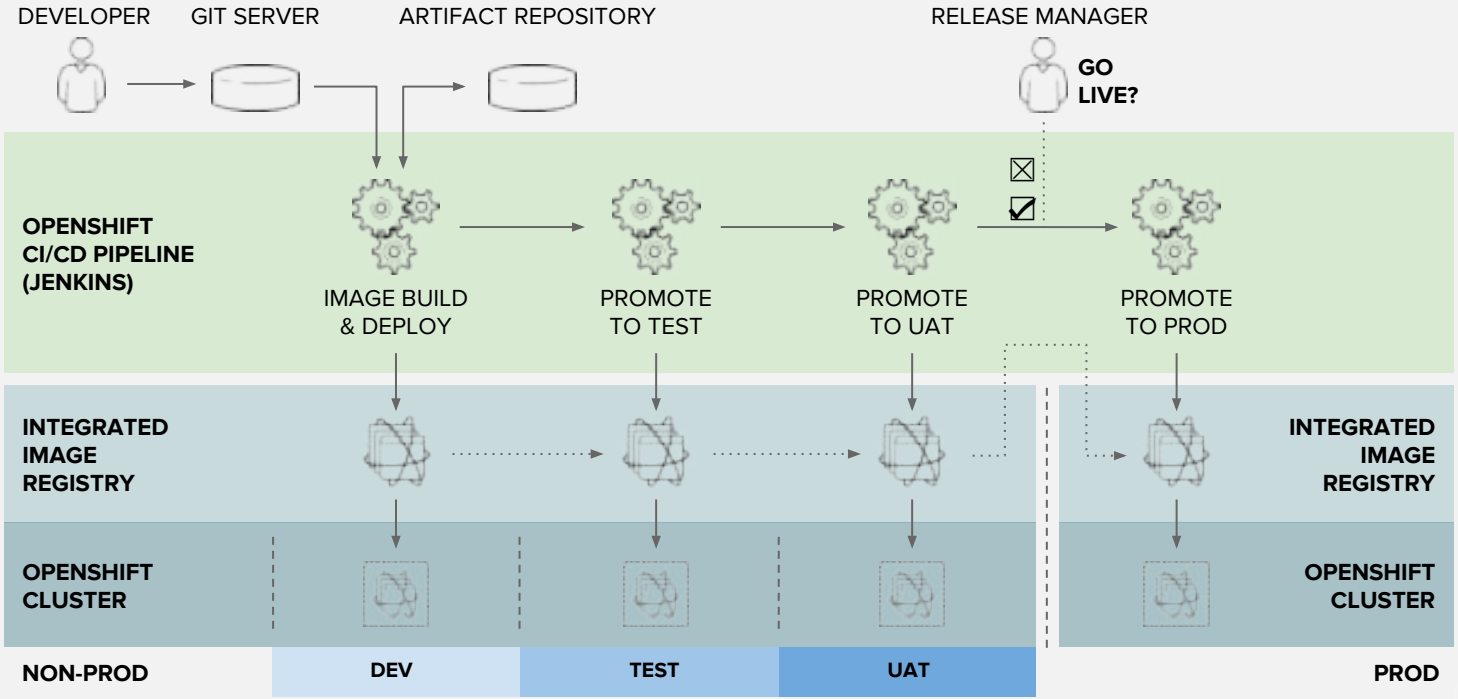
CONTINUOUS DELIVERY PIPELINE



CONTINUOUS DELIVERY PIPELINE



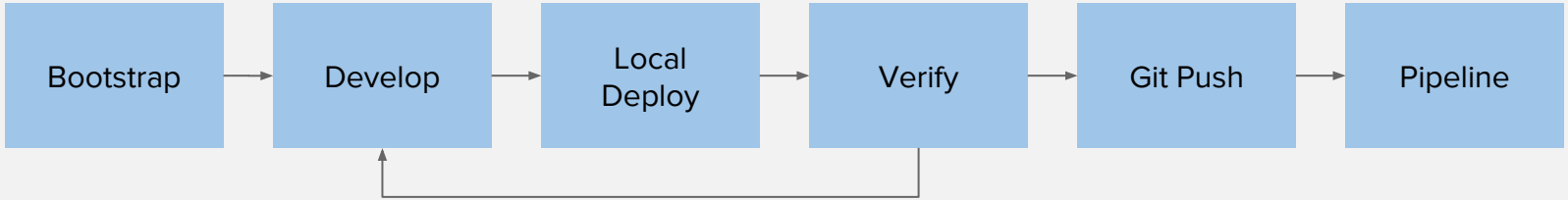
CONTINUOUS DELIVERY PIPELINE



redhat

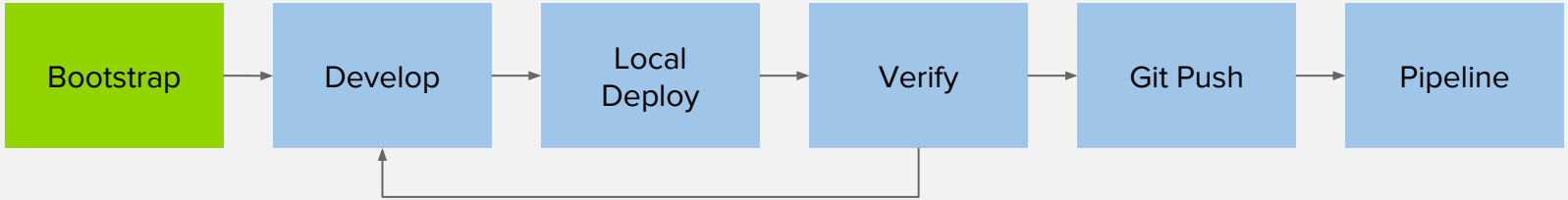
DEVELOPER WORKFLOW

LOCAL DEVELOPMENT WORKFLOW



RED HAT® OPENSHIFT Container Platform

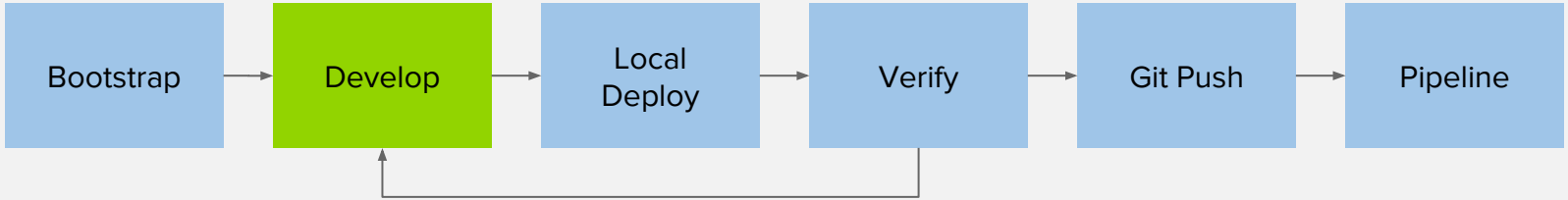
LOCAL DEVELOPMENT WORKFLOW



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

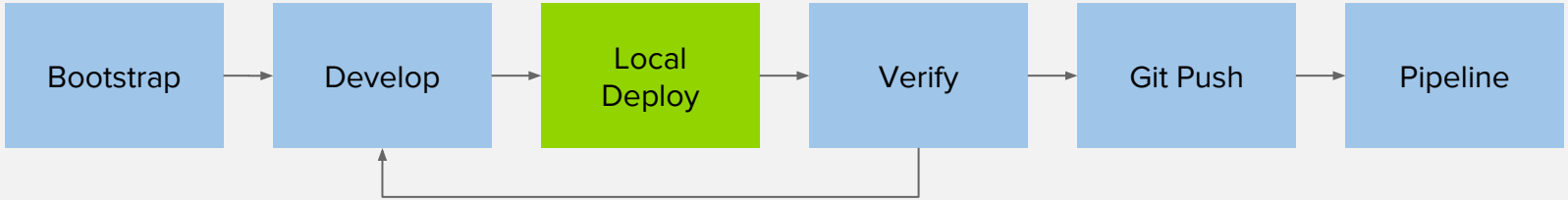
LOCAL DEVELOPMENT WORKFLOW



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

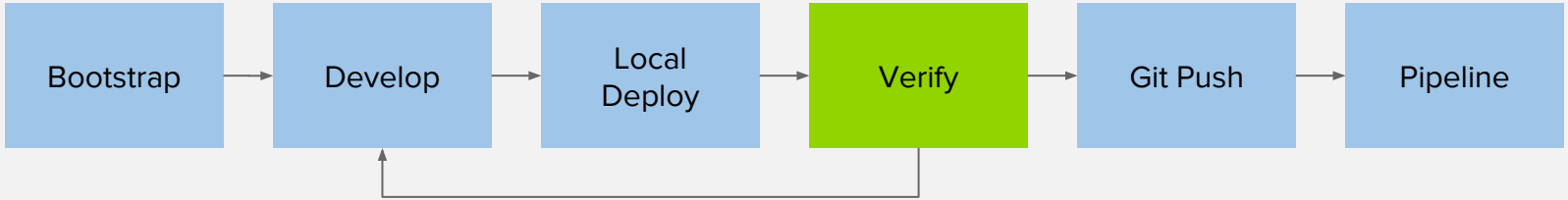
LOCAL DEVELOPMENT WORKFLOW



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

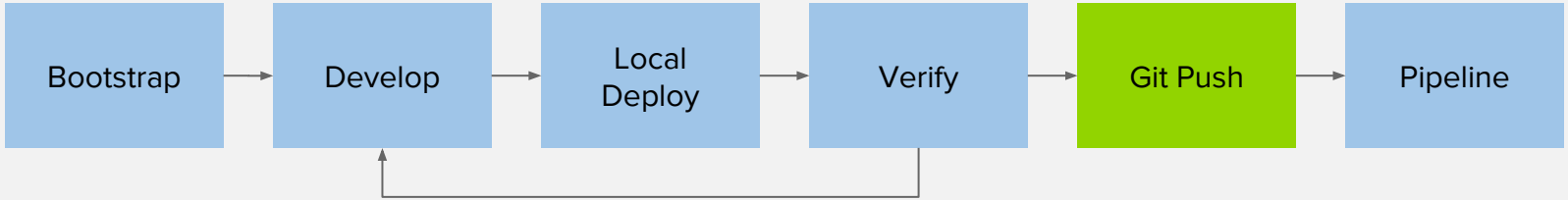
LOCAL DEVELOPMENT WORKFLOW



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

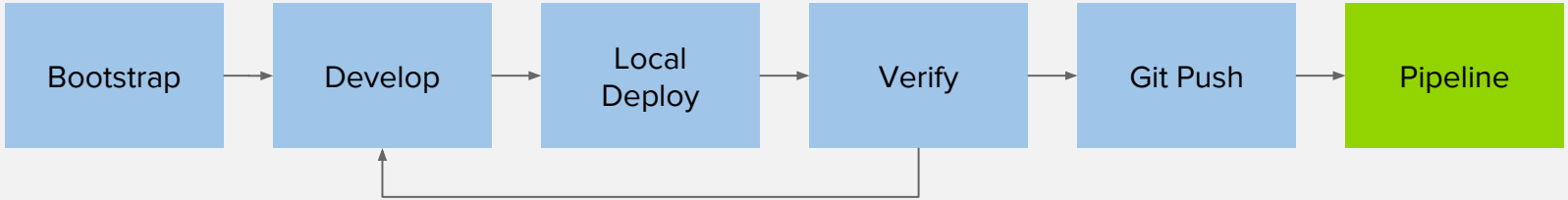
LOCAL DEVELOPMENT WORKFLOW



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

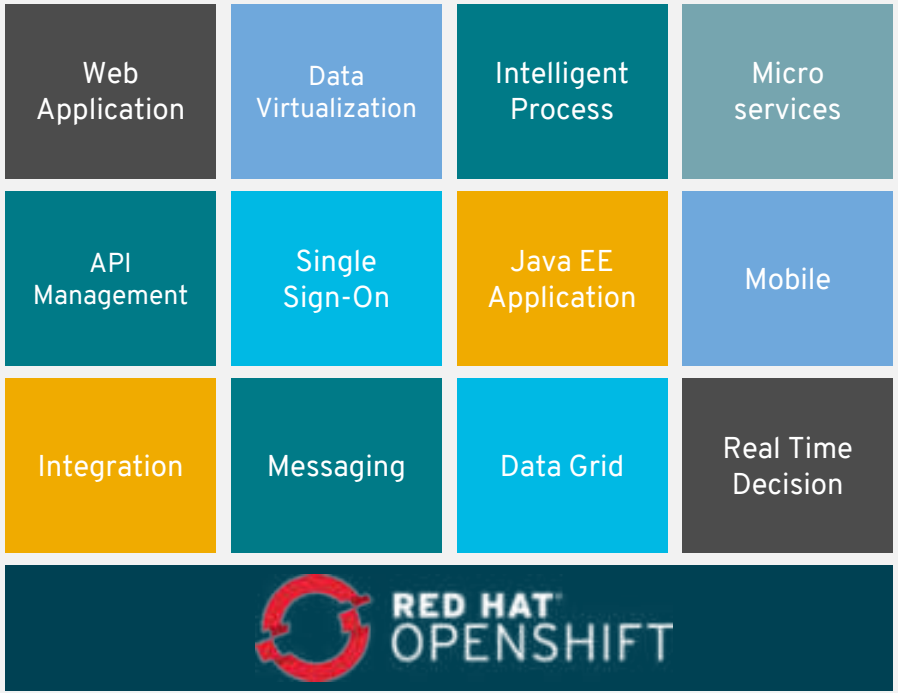


PIPELINE

- 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

CrunchyData

GitLab

Iron.io

Couchbase

Sonatype

EnterpriseDB

NuoDB

Fujitsu

and many more

redhat.

THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHatNews

youtube.com/user/RedHatVideos