

# SECURING CONTAINERS WITH OPENSHIFT

Alfred Bach

Principal Solution Architect Cloud /Infra

EMEA Partner Enablement

# What makes an effective hybrid cloud platform?

BROAD ECOSYSTEM

BROADEST APPLICATION SUPPORT

DEVELOPER EXPERIENCE & ON-DEMAND

AUTOMATED OPERATIONS

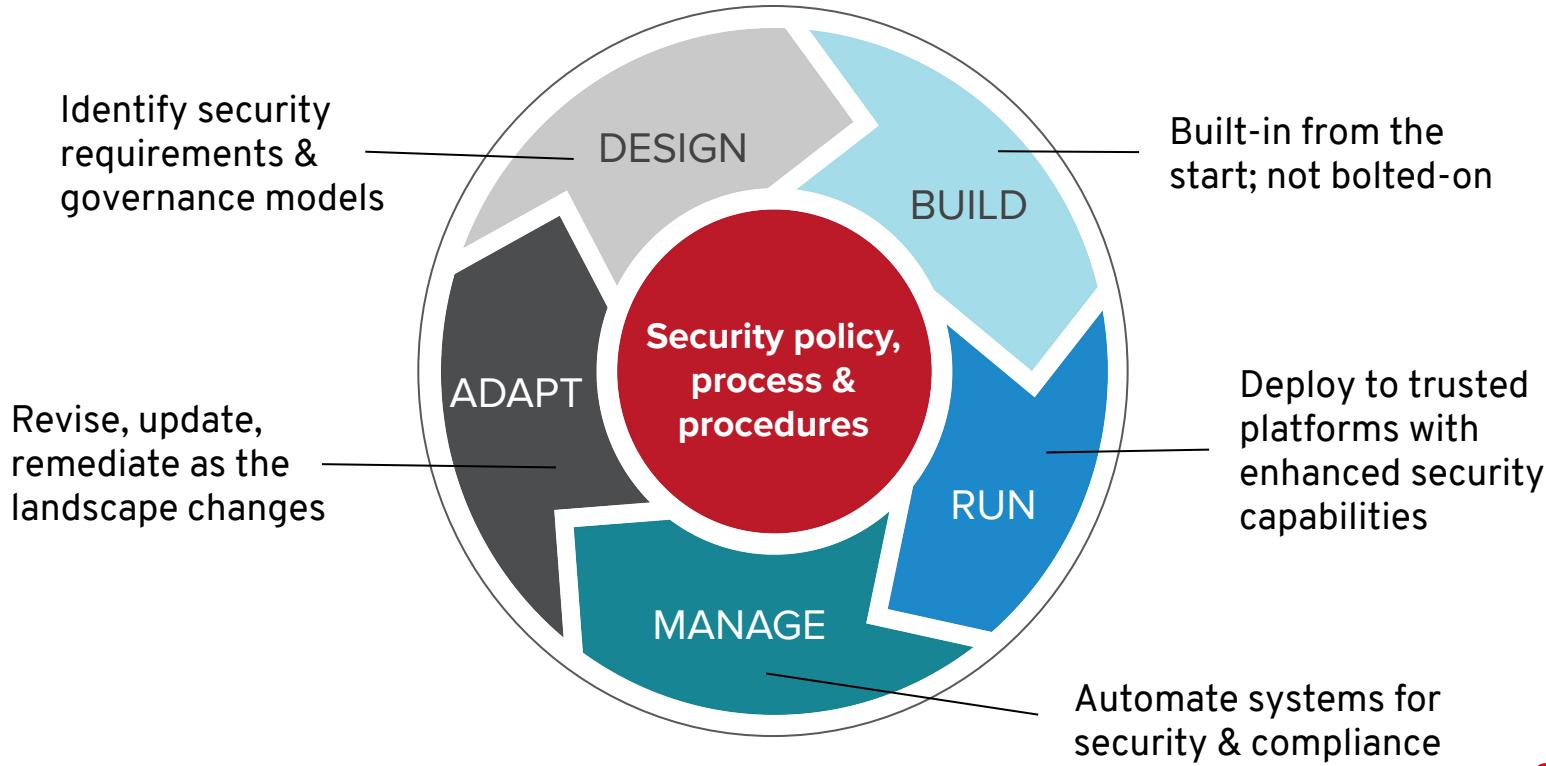
STANDARDS, PORTABILITY & INTEROPERABILITY

SECURITY & COMPLIANCE



# Security must be continuous

## And integrated throughout the IT lifecycle



# A Comprehensive Approach to Securing Containers



## CONTROL

Application  
Security

Container Content

CI/CD Pipeline

Container Registry

Deployment Policies



## DEFEND

Infrastructure

Container Platform

Host Multi-tenancy

Network Isolation

Storage

Audit & Logging

API Management



## EXTEND

Security Ecosystem

# Hardening, applicability guides, certifications

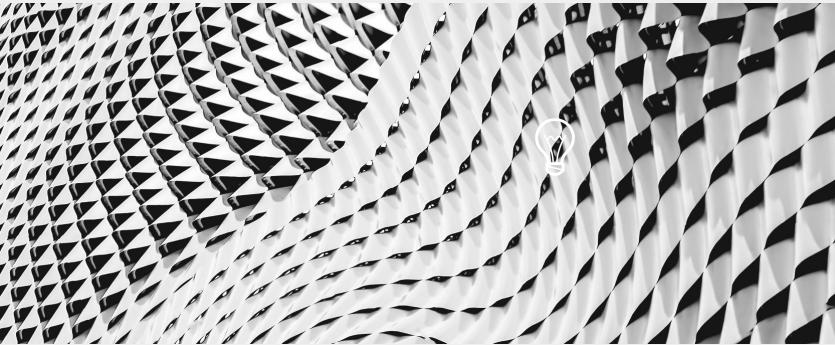
## OpenShift 4

- Available now
  - [HIPAA](#)
  - [ISO 27001](#) (ask RH for a copy)
  - [FISMA](#) (ask RH for a copy)
  - [The OpenShift Security Guide](#)
  - [OpenShift 4 Hardening Guide](#) (ask RH for a copy)
- Target Q1 CY 2021
  - CIS OpenShift Benchmark
  - HITRUST
  - PCI-DSS

## Managed Services certifications

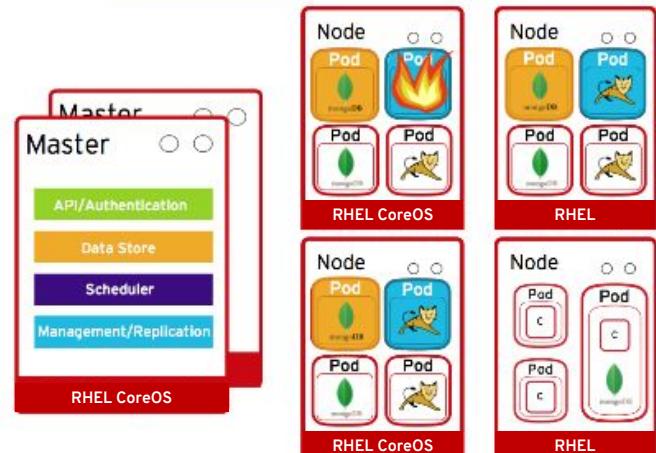
- SOC2-type 1, SOC2-type 2
  - OpenShift Dedicated (OSD) on AWS
  - ARO, IBM ROKS, ROSA
  - In process for OSD on GCP
- ISO-27001
  - OSD on AWS, ARO
- PCI-DSS
  - ARO, IBM ROKS
  - In process for OSD on AWS and GCP, ROSA
- FedRAMP
  - ARO, IBM ROKS
  - In process for OSD on AWS, ROSA
- HIPAA and/or HITRUST
  - ARO, IBM ROKS
  - In discussion for OSD and ROSA

# DEFEND INFRASTRUCTURE



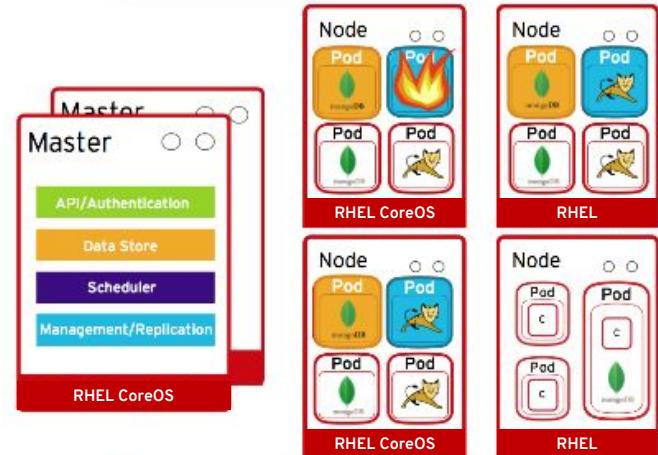
# Securing the container platform

- Configuration and lifecycle management
- Host & runtime security
- Identity and Access Management
- Data at rest, data in transit
- Logging, Monitoring, Metrics
- Audit and Compliance



# Securing the container platform

- **Configuration and lifecycle management**
- Host & runtime security
- Identity and Access Management
- Data at rest, data in transit
- Logging, Monitoring, Metrics
- Audit and Compliance



# Automated Configuration and Lifecycle Management

## Dramatically simplified for the Hybrid Cloud



### Machines

Machines are complex for ops



Make machines easy  
(like containers)



### Configuration

Config change is risky



Make config management  
and config change  
easy and safe



### Lifecycle

Software lifecycle is hard



Automate software  
lifecycle on Kube

# Automated Container Operations

FULLY AUTOMATED DAY-1 AND DAY-2 OPERATIONS

INSTALL

DEPLOY

HARDEN

OPERATE

## AUTOMATED OPERATIONS

Infra provisioning

Full-stack deployment

Secure defaults

Multicloud aware

Embedded OS

On-premises and cloud

Network isolation

Monitoring and alerts

Unified experience

Audit and logs

Full-stack patch & upgrade

Signing and policies

Zero-downtime upgrades

Vulnerability scanning

# The Value Of Kubernetes Operators

No need for operator      Requires custom Operator built with SDK



## Installation

Automated application provisioning and configuration management

## Upgrades

Patch and minor version upgrades supported

## Lifecycle

App lifecycle, storage lifecycle (backup, failure recovery)

## Deep Insights

Metrics, alerts, log processing and workload analysis

## Auto-pilot

Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning...



# Day 1 Installation

## OPENSHIFT CONTAINER PLATFORM

### Full Stack Automation

Simplified opinionated “Best Practices” for cluster provisioning

Fully automated installation and updates including host container OS.



**Red Hat**  
Enterprise Linux  
CoreOS

### Pre-existing Infrastructure

Customer managed resources & infrastructure provisioning

Plug into existing DNS and security boundaries



**Red Hat**  
Enterprise Linux  
CoreOS



**Red Hat**  
Enterprise Linux

## HOSTED OPENSHIFT

### Azure Red Hat OpenShift

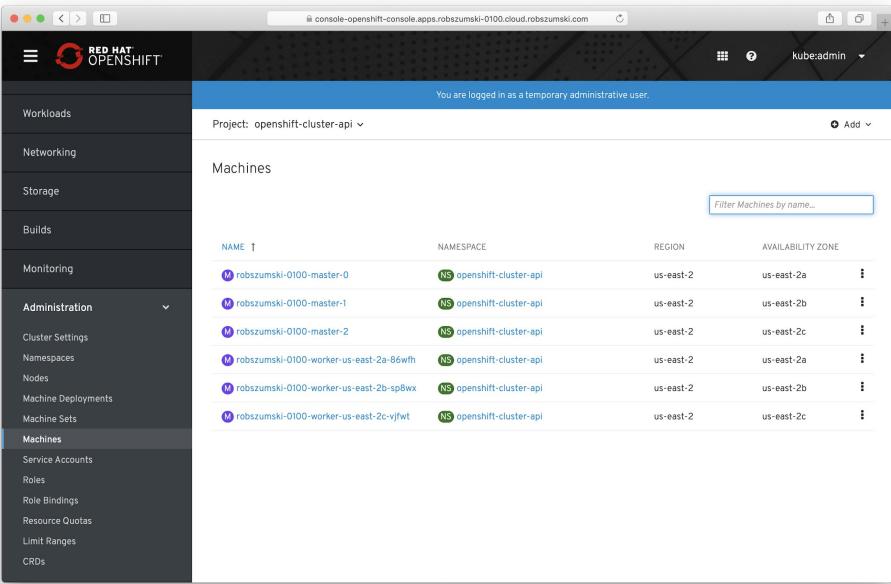
Deploy directly from the Azure console. Jointly managed by Red Hat and Microsoft Azure engineers.

### OpenShift Dedicated

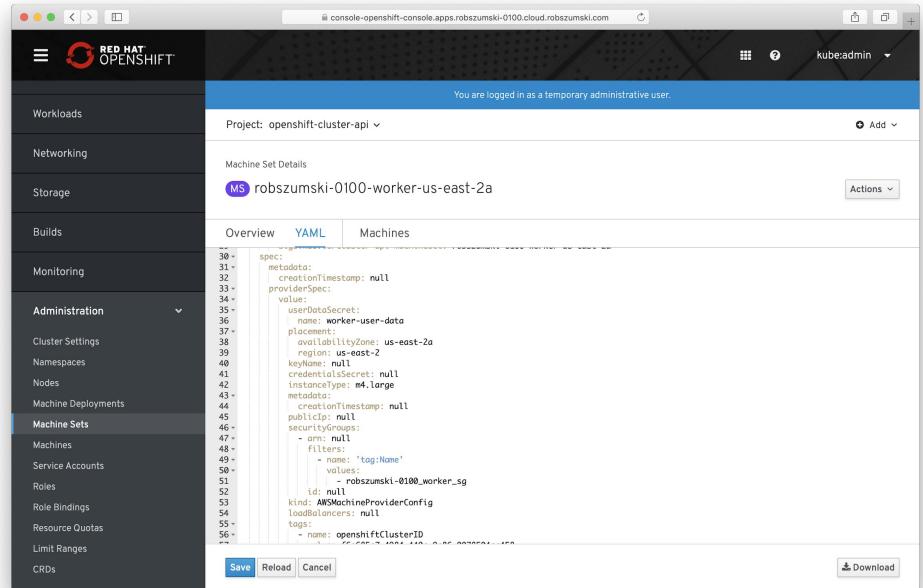
Get a powerful cluster, fully Managed by Red Hat engineers and support.

# Kubernetes Machine API Operator

## Using Kubernetes To Provision And Scale Clusters



The screenshot shows the Red Hat OpenShift web console interface. The left sidebar is a navigation menu with the following items: Workloads, Networking, Storage, Builds, Monitoring, Administration (with sub-options: Cluster Settings, Namespaces, Nodes, Machine Deployments, Machine Sets, and Machines), Service Accounts, Roles, Role Bindings, Resource Quotas, Limit Ranges, and CRDs. The 'Machines' item is currently selected. The main content area has a header 'Project: openshift-cluster-api' and a message 'You are logged in as a temporary administrative user.' Below this is a table titled 'Machines' with columns: NAME, NAMESPACE, REGION, and AVAILABILITY ZONE. The table lists several machine objects, each with a '⋮' icon for more options. A search bar 'Filter Machines by name...' is located at the top of the table.



The screenshot shows the Red Hat OpenShift web console interface. The left sidebar is a navigation menu with the following items: Workloads, Networking, Storage, Builds, Monitoring, Administration (with sub-options: Cluster Settings, Namespaces, Nodes, Machine Deployments, Machine Sets, and Machines), Service Accounts, Roles, Role Bindings, Resource Quotas, Limit Ranges, and CRDs. The 'Machine Sets' item is currently selected. The main content area has a header 'Project: openshift-cluster-api' and a message 'You are logged in as a temporary administrative user.' Below this is a section titled 'Machine Set Details' with a sub-section for 'robiszumski-0100-worker-us-east-2a'. The 'Overview' tab is selected, showing a YAML configuration for the machine set. The configuration includes fields like 'spec', 'metadata', 'creationTimestamp', 'providerSpec', 'value', 'userSecret', 'nodeSelector', 'placement', 'availabilityZone', 'region', 'keyName', 'creationTimestamp', 'instanceType', 'metadata', 'securityGroup', and 'tags'. The 'Machines' tab is also visible. At the bottom, there are 'Save', 'Reload', and 'Cancel' buttons, and a 'Download' link.

# Day 2 Configuration

## Global Configuration

You complete most of the cluster configuration and customization after you deploy your OpenShift Container Platform cluster.

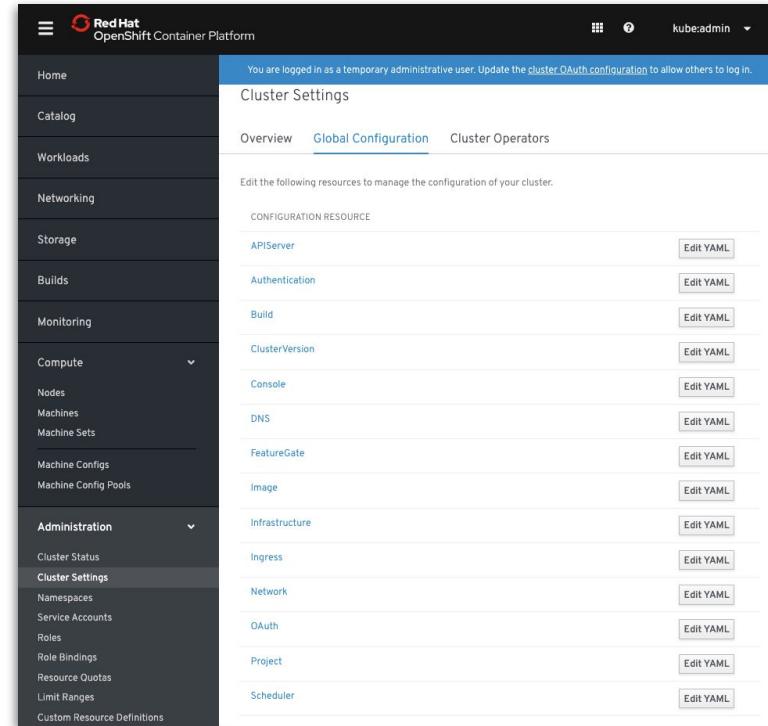
### Change via Cluster Settings screen

Once you have discovered your desired settings (prev. slide), changes can be made via Console or CLI.

### Operators apply these updates

One or more Operators are responsible for propagating these settings through the infrastructure

- Identity Provider
- Ingress Controller
- Logging, Metrics



# Smarter Software Updates

## No downtime for well behaving apps

Applications with multiple replicas, using liveness probes, health checks and taints/tolerations  
Node Pools with more than one worker and slack resources

## Maintenance window for entire cluster

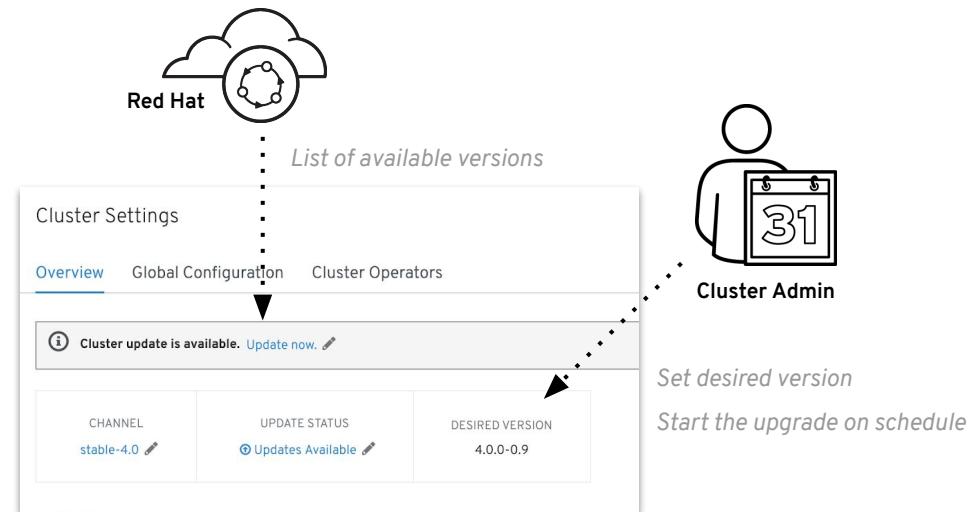
No need for separate windows for each component

## Upgrade runs completely on the cluster

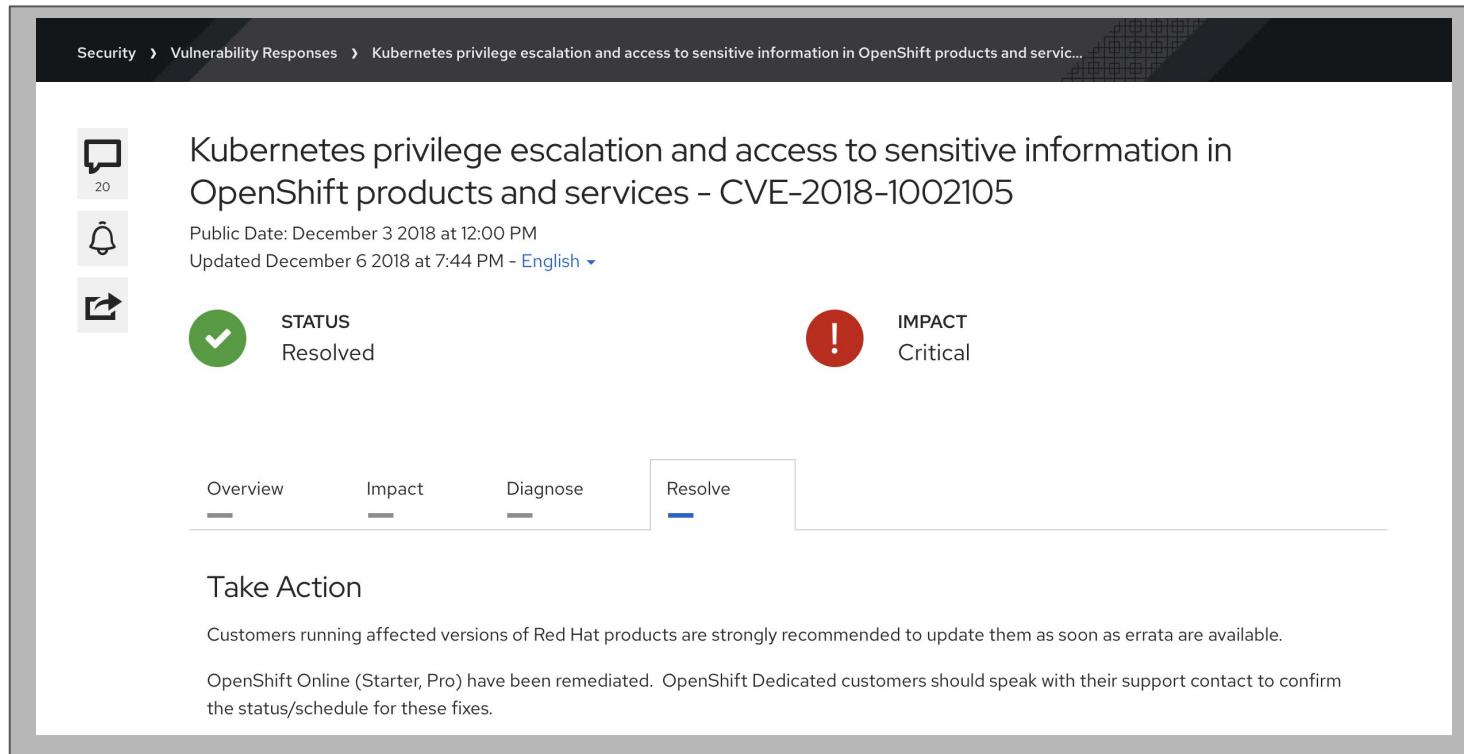
No more long running processes on a workstation

## Constant health checking from each Operator

Operators are constantly looking for incompatibilities and issues that might arise



# Fixes to Kubernetes vulnerabilities



The screenshot shows a Red Hat security vulnerability response page. The URL is <https://access.redhat.com/security/vulnerabilities/3716411>. The page title is "Kubernetes privilege escalation and access to sensitive information in OpenShift products and services - CVE-2018-1002105". The public date is December 3 2018 at 12:00 PM, and it was updated on December 6 2018 at 7:44 PM. The status is "Resolved" (green checkmark icon) and the impact is "Critical" (red exclamation mark icon). The navigation tabs at the bottom are Overview, Impact, Diagnose, and Resolve (which is selected, indicated by a blue underline). The "Take Action" section advises customers to update Red Hat products as soon as errata are available. It notes that OpenShift Online (Starter, Pro) have been remediated, and OpenShift Dedicated customers should speak with their support contact.

Security > Vulnerability Responses > Kubernetes privilege escalation and access to sensitive information in OpenShift products and services - CVE-2018-1002105

20

20

December 3 2018 at 12:00 PM

December 6 2018 at 7:44 PM - English

STATUS

Resolved

IMPACT

Critical

Overview Impact Diagnose Resolve

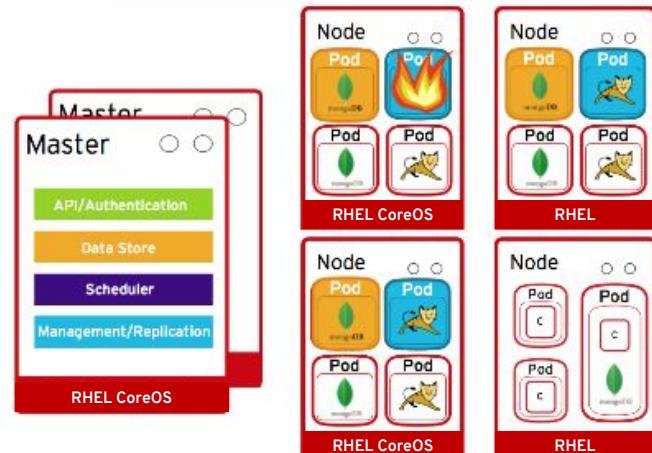
Take Action

Customers running affected versions of Red Hat products are strongly recommended to update them as soon as errata are available.

OpenShift Online (Starter, Pro) have been remediated. OpenShift Dedicated customers should speak with their support contact to confirm the status/schedule for these fixes.

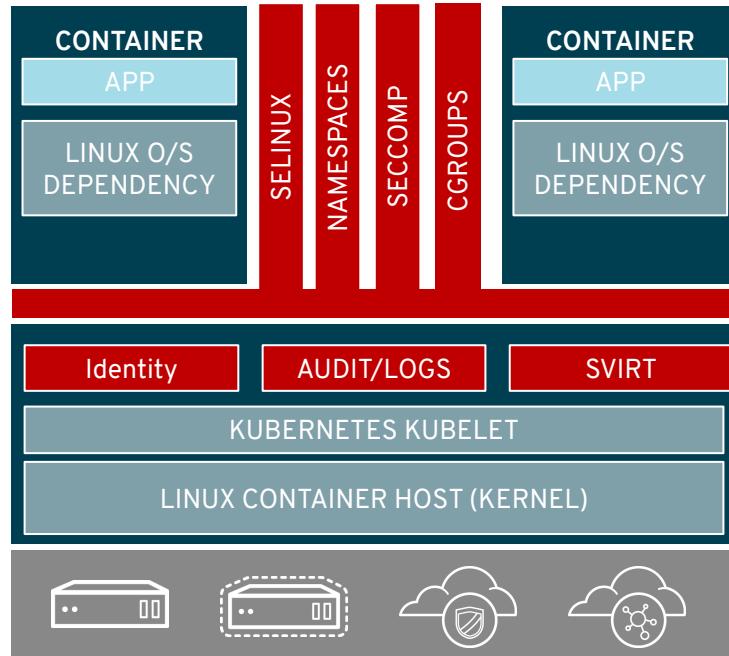
# Securing the container platform

- Configuration and lifecycle management
  - **Host & runtime security**
  - Identity and Access Management
  - Data at rest, data in transit
  - Logging, Monitoring, Metrics
  - Audit and Compliance



# Container security starts with Linux security

- Security in the RHEL host applies to the container
- RHEL enables container multitenancy
- SELinux and Kernel Namespaces are the one-two punch no one can beat
- Protects not only the host, but containers from each other
- RHEL CoreOS provides minimized attack surface



# Red Hat Enterprise Linux CoreOS

The Immutable Container Optimized Operating System

OPENSHIFT 4

OPENSHIFT PLATFORM



OPERATING SYSTEM



RED HAT<sup>®</sup>  
ENTERPRISE  
LINUX CoreOS



## Role in OpenShift Ecosystem

- Versioned and validated for specific OpenShift version
- Required for masters. RHEL option for workers
- User space read-only

## Managed by the OpenShift Cluster

- Considered a member of an OpenShift Deployment
- Configuration managed by the Machine Config Operator
  - Container runtime
  - Kubelet configuration
  - Authorized container registries
  - SSH Configuration



# 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

[CRI - the Container Runtime Interface](#)

[OpenShift 4 defaults to CRI-O](#)

[Red Hat contributes CRI-O to the Cloud Native Computing Foundation](#)

# Runtime security policies

## SCC (Security Context Constraints)

Allow administrators to control permissions for pods

Restricted SCC is granted to all users

By default, no containers can run as root

Admin can grant access to privileged SCC

Custom SCCs can be created

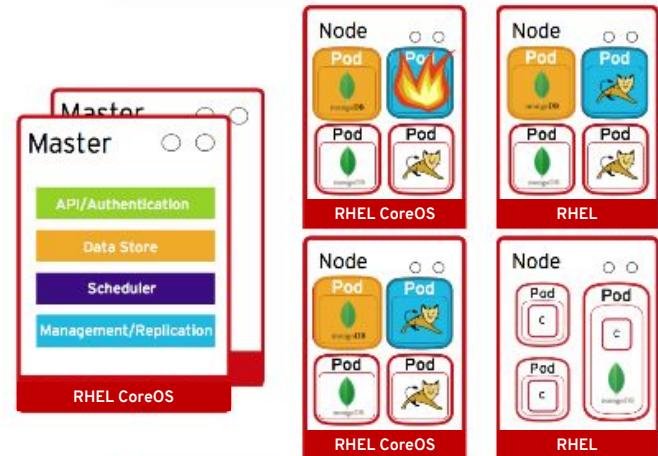
```
$ oc describe scc restricted
Name: restricted
Priority: <none>
Access:
  Users: <none> ①
  Groups: system:authenticated ②
Settings:
  Allow Privileged: false
  Default Add Capabilities: <none>
  Required Drop Capabilities: KILL,MKNOD,SYS_CHROOT,SETUID,SETGID
  Allowed Capabilities: <none>
  Allowed Seccomp Profiles: <none>
  Allowed Volume Types: configMap,downwardAPI,emptyDir,persistentVolumeClaim,projected,
  Allow Host Network: false
  Allow Host Ports: false
  Allow Host PID: false
  Allow Host IPC: false
  Read Only Root Filesystem: false
  Run As User Strategy: MustRunAsRange
    UID: <none>
    UID Range Min: <none>
    UID Range Max: <none>
  SELinux Context Strategy: MustRunAs
    User: <none>
    Role: <none>
    Type: <none>
    Level: <none>
  FSGroup Strategy: MustRunAs
    Ranges: <none>
  Supplemental Groups Strategy: RunAsAny
    Ranges: <none>
```



① Lists which users and service accounts the SCC is applied to.  
② Lists which groups the SCC is applied to.

# Securing the container platform

- Configuration and lifecycle management
- Host & runtime security
- **Identity and Access Management**
- Data at rest, data in transit
- Logging, Monitoring, Metrics
- Audit and Compliance



# Identity and access management

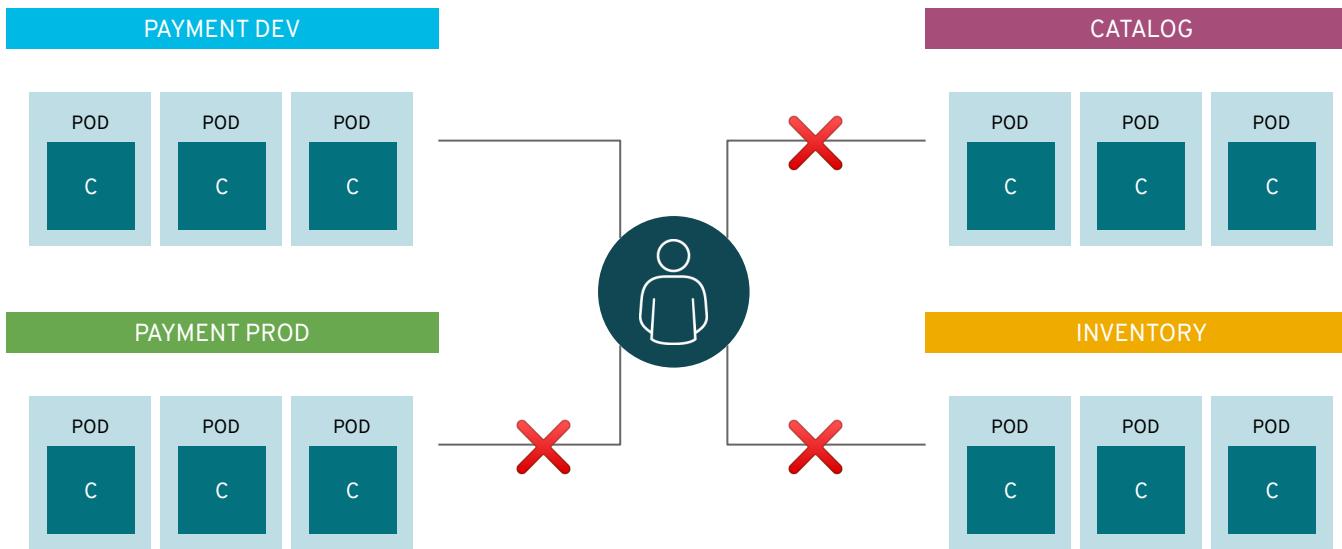
OpenShift includes an OAuth server, which does three things:

- Identifies the person requesting a token, using a configured identity provider
  - Determines a mapping from that identity to an OpenShift user
  - Issues an OAuth access token which authenticates that user to the API
- [Managing Users and Groups in OpenShift](#)  
[Configuring Identity Providers](#)

Supported Identity Providers include

- Keystone
- LDAP
- GitHub
- GitLab
- GitHub Enterprise (new with 3.11)
- Google
- OpenID Connect
- Security Support Provider Interface (SSPI) to support SSO flows on Windows (Kerberos)

# Projects isolate applications across teams, groups and departments



# Restrict access by need to know

## Role based authorization

- Project scope & cluster scope available
- Matches request attributes (verb,object,etc)
- If no roles match, request is denied ( deny by default )
- Operator- and user-level roles are defined by default
- Custom roles are supported

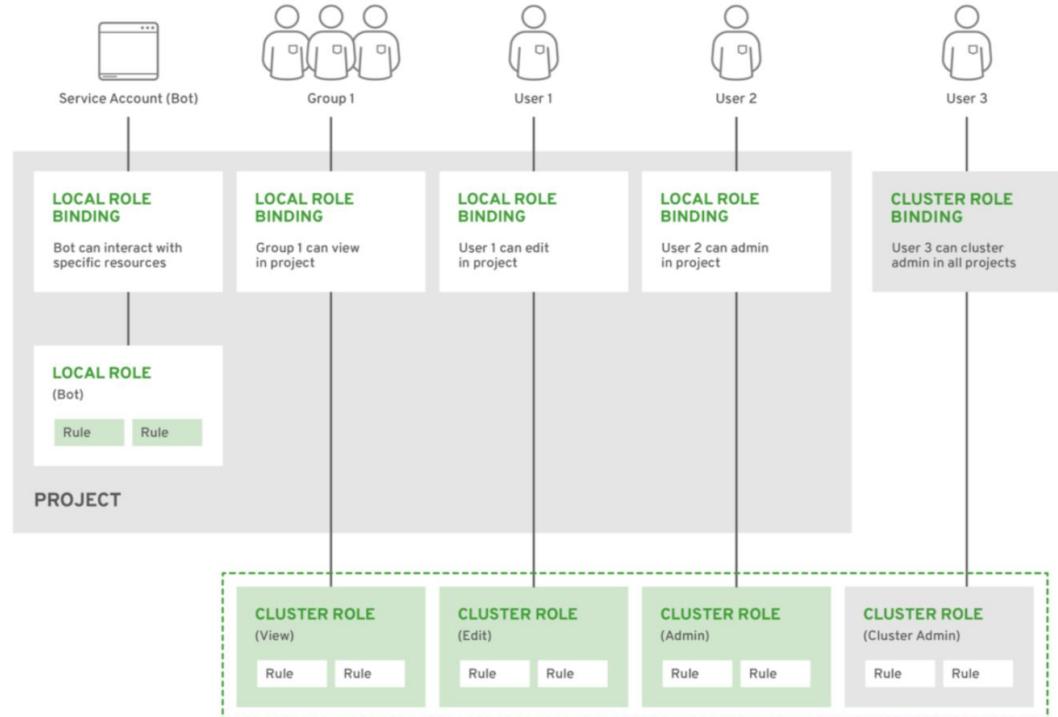
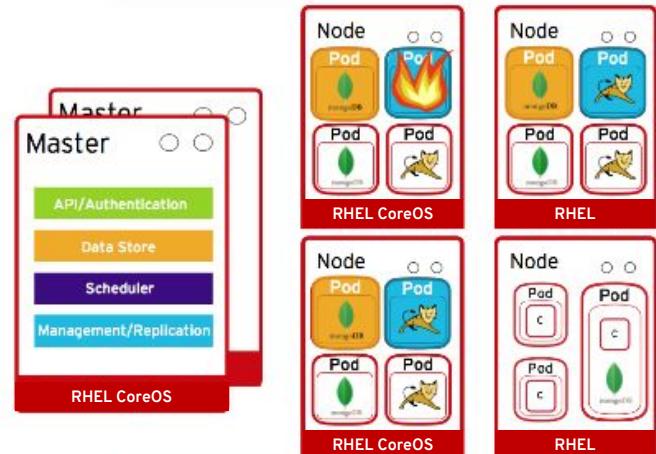


Figure 12 - Authorization Relationships

# Securing the container platform

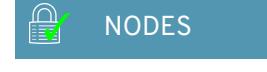
- Configuration and lifecycle management
- Host & runtime security
- Identity and Access Management
- **Data at rest, data in transit**
- Logging, Monitoring, Metrics
- Audit and Compliance



# Certificate management

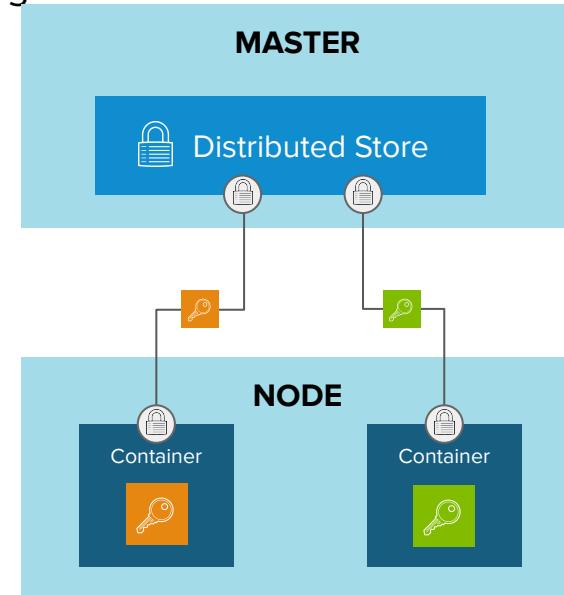
- Certificates are used to provide secure connections to
  - master and nodes
  - Ingress controller and registry
  - etcd
- Certificate rotation is automated
- Optionally configure external endpoints to use custom certificates
- For example:

[Requesting and Installing Let's Encrypt Certificates for OpenShift 4](#)



# Secrets 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 (e.g. vaults)
- Encrypted in transit and at rest
  - Encrypt the etcd datastore
  - Encrypt RHCOS volumes
- Never rest on the nodes



# Volume Encryption

## Network Bound Disk Encryption

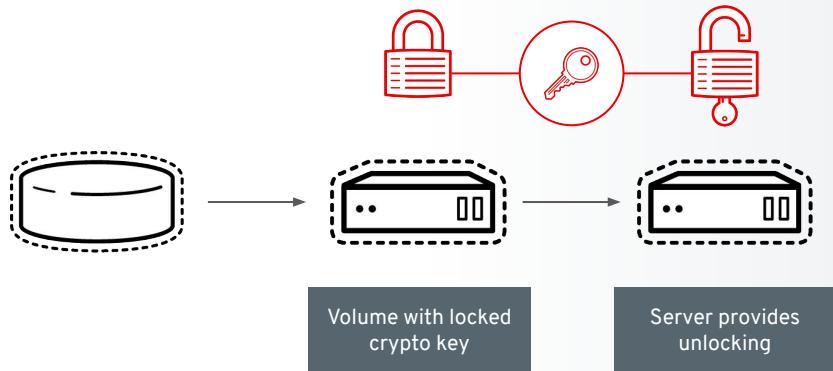
- Provides encryption for local storage
- Addresses disk/image theft
- Platform/cloud agnostic implementation
- TPM/vTPM (v2) and Tang endpoints for automatic decryption



# Attached storage

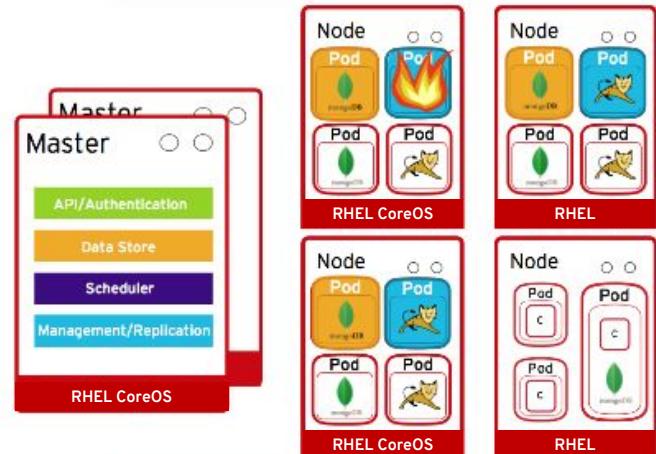
Secure storage by using

- SELinux access controls
- Secure mounts
- Supplemental group IDs for shared storage
- Network bound disk encryption



# Securing the container platform

- Configuration and lifecycle management
- Host & runtime security
- Identity and Access Management
- Data at rest, data in transit
- **Logging, Monitoring, Metrics**
- Audit and Compliance



# Cluster log and audit management

## Install the Elasticsearch and Cluster Logging Operators

- EFK stack aggregates 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
  - Central Audit policy configuration
- API server events are automatically audited
- Logging pipelines collect API server and host audit logs as well as cluster and application logs for forwarding to the SIEM of your choice

### Create Operator Subscription

Keep your service up to date by selecting a channel and approval strategy. The strategy determines either manual or automatic updates.

#### Installation Mode \*

All namespaces  
This mode  
Operator will be available in a single namespace only.

A specific namespace on the cluster  
Operator will be available in a single namespace only.

openshift-logging

#### Update Channel \*

preview

#### Approval Strategy \*

Automatic

Manual

Subscribe

Cancel

```
# configure via CRD
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
    type: "elasticsearch"
    elasticsearch:
      nodeCount: 3
      resources:
        limits:
          cpu: 800m
          memory: 1Gi
        requests:
          cpu: 800m
          memory: 1Gi
      storage:
        storageClassName: gp2
        size: 100G
        redundancyPolicy: "SingleRedundant"
    visualization:
      type: "kibana"
      kibana:
        replicas: 1
    curation:
      type: "curator"
```

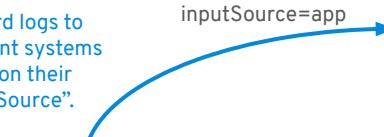
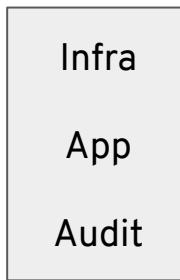
# Introduce new log forwarding API

**Abstract Fluentd configuration by introduce new log forwarding API to improve support and experience for customers.**

- Introduce a new, cluster-wide *ClusterLogForwarder* CRD (API) that replaces needs to configure log forwarding via Fluentd ConfigMap.
- The API helps to reduce probability to misconfigure Fluentd and helps bringing in more stability into the Logging stack.
- Route logs based on their source type (infra, app or audit logs) and filter them further by namespaces.
- Collecting and forwarding audit logs
- With the API, we also introduce the following endpoint improvements to bring more value on to the table:
  - Improved syslog support adding TLS for secure communication + support for the newest standard (RFC5424).
  - Kafka support.



Forward logs to different systems based on their “inputSource”.



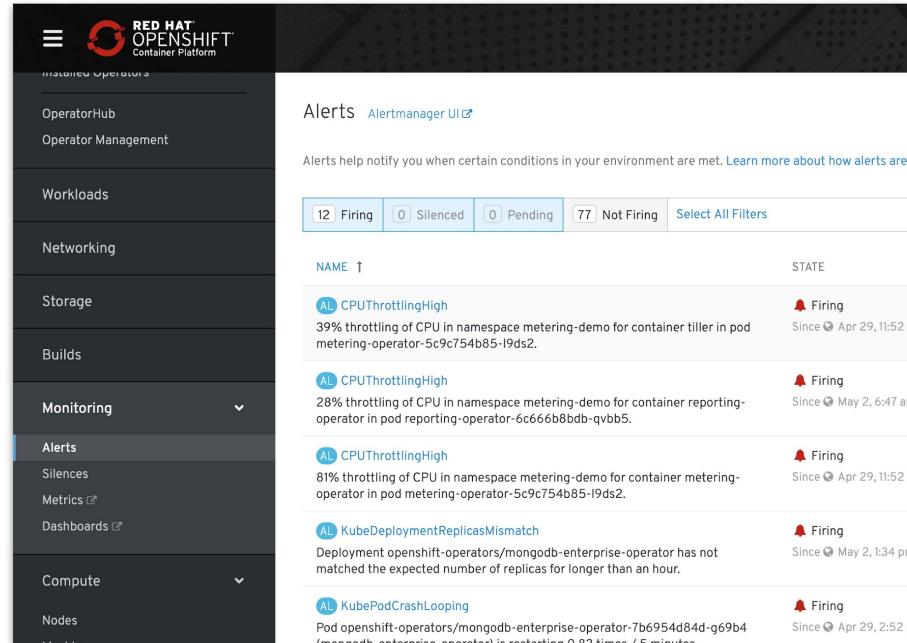
```
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogForwarder"
spec:
  outputs:
    - name: MyLogs
      type: Syslog
      syslog:
        Facility: Local0
        url: localstore.example.com:9200
  pipelines:
    - inputs: [Infrastructure, Application, Audit]
      outputs: [MyLogs]
```



# Cluster monitoring

## Cluster monitoring is installed by default

- Exposes resource metrics for Horizontal Pod Autoscaling (HPA) by default
  - HPA based on custom metric is tech preview
- No manual etcd monitoring configuration anymore
- New screens for managing Alerts & Silences
- More metrics available for troubleshooting purposes (e.g. HAProxy)
- Configuration via ConfigMaps and Secrets

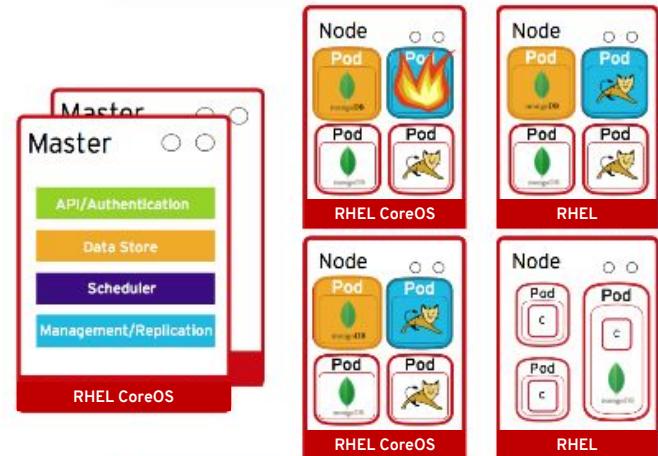


The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar has a dark theme with the Red Hat logo and 'RED HAT OPENSHIFT Container Platform'. It includes sections for 'Installed Operators', 'OperatorHub', 'Operator Management', 'Workloads', 'Networking', 'Storage', 'Builds', 'Monitoring' (which is currently selected), 'Alerts', 'Silences', 'Metrics', 'Dashboards', 'Compute', and 'Nodes'. The 'Monitoring' section is expanded, showing 'Alerts', 'Silences', 'Metrics', and 'Dashboards'. The 'Alerts' section is further expanded, showing a list of alerts with columns for 'NAME', 'STATE', and 'LAST FIRED'. The alerts listed are:

NAME	STATE	LAST FIRED
AL CPUThrottlingHigh	Firing	Since 2023-04-29 11:52:00
AL CPUThrottlingHigh	Firing	Since 2023-05-02 06:47:00
AL CPUThrottlingHigh	Firing	Since 2023-05-02 06:47:00
AL KubeDeploymentReplicasMismatch	Firing	Since 2023-05-02 01:34:00
AL KubePodCrashLooping	Firing	Since 2023-04-29 20:52:00

# Securing the container platform

- Configuration and lifecycle management
- Host & runtime security
- Identity and Access Management
- Data at rest, data in transit
- Logging, Monitoring, Metrics
- **Audit and Compliance**



# View Security Vulnerabilities with the Quay Operator

See all your Container Vulnerabilities right from the Console Dashboard

- Link out to **Red Hat Quay** for more in depth information
- The Quay Operator supports both **On-premise and External Quay** Registries
- Currently uses **Clair for Security Scan**; Planning to expand to other Vendors( TwistLock, Aqua, e.g. )
- Only works for images managed by Quay

The screenshot shows the Red Hat OpenShift Container Platform dashboard. A blue arrow points from the text "See all your Container Vulnerabilities right from the Console Dashboard" to the "Dashboards" section of the sidebar. Another blue arrow points from the text "Link out to Red Hat Quay for more in depth information" to the Quay image detail view.

**Dashboard Overview:** The main dashboard shows a summary of cluster status: Cluster (green), Control Plane (green), and Image Security (red, 1 vulnerability). It also displays a timeline of recent events, including a warning about deprecated APIs.

**Quay Image Detail View:** This view shows a donut chart of vulnerabilities (61 total) across three severity levels: High (14), Medium (33), and Low (14). Below the chart, a table lists specific vulnerabilities with details like CVE ID, severity, package, and introduced in layer.

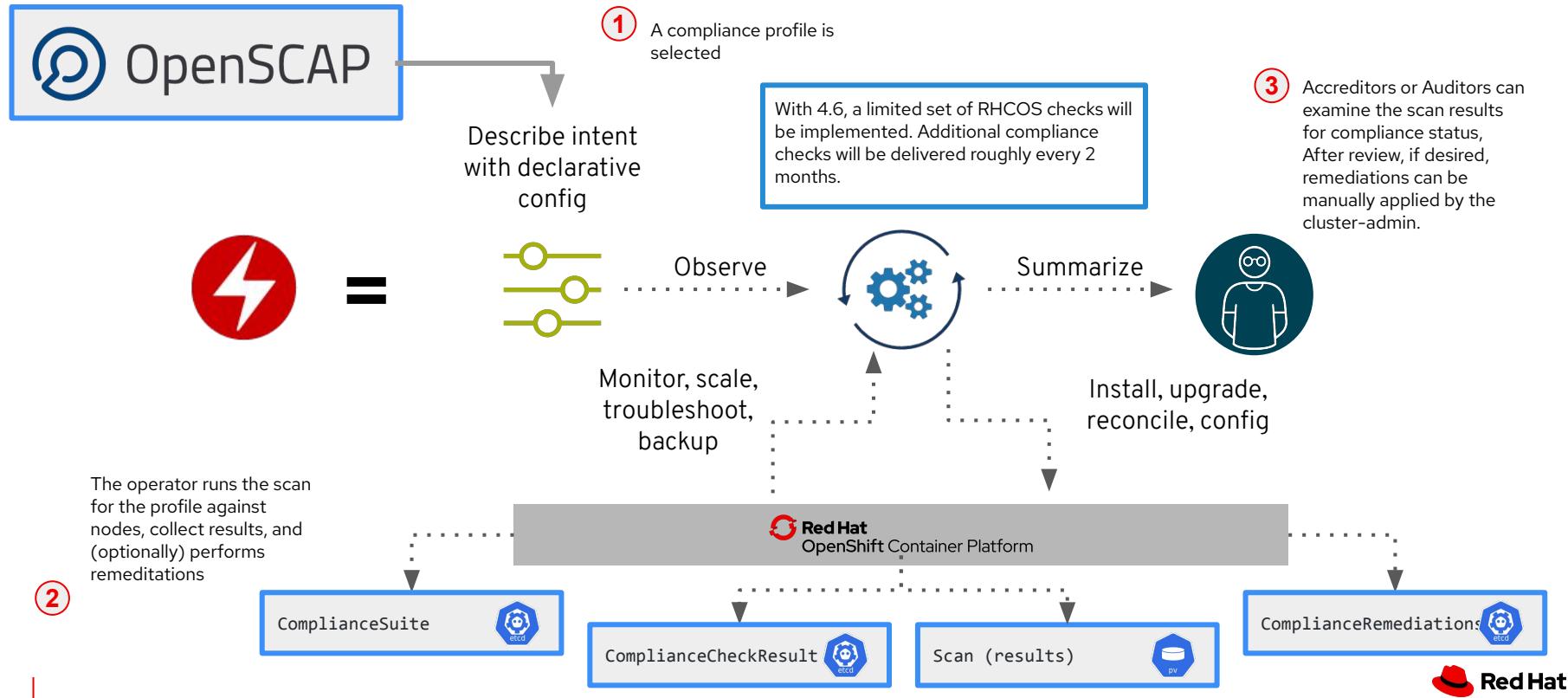
CVE	SEVERITY	PACKAGE	CURRENT VERSION	FIXED IN VERSION	INTRODUCED IN LAYER
RHSA-2019-0710	High	python-lbs	2.7.5-6.el7	0.2.2-5.7-7.el7_8	
RHSA-2019-1587	High	python-lbs	2.7.5-6.el7	0.2.2-5.8-0.el7_8	
RHSA-2019-0368	High	systemd-lbs	219-57.el7	0.219-62.el7_8.5	
RHSA-2019-0049	High	systemd-lbs	219-57.el7	0.219-62.el7_8.2	
RHSA-2019-0679	High	libssh2	1.4.3-10.el7_2.1	0.1.4.3-12.el7_8.2	
RHSA-2018-2285	High	yunplug-novl	1.1.31-6.el7	0.1.1.31-46.el7_5	
RHSA-2018-5184	High	curl-lbs	7.6.2-1.el7	0.7.6.2-3.el7	

**Security Breakdown:** A circular chart shows the distribution of vulnerabilities by severity: 22% High, 23% Medium, and 54% Low.

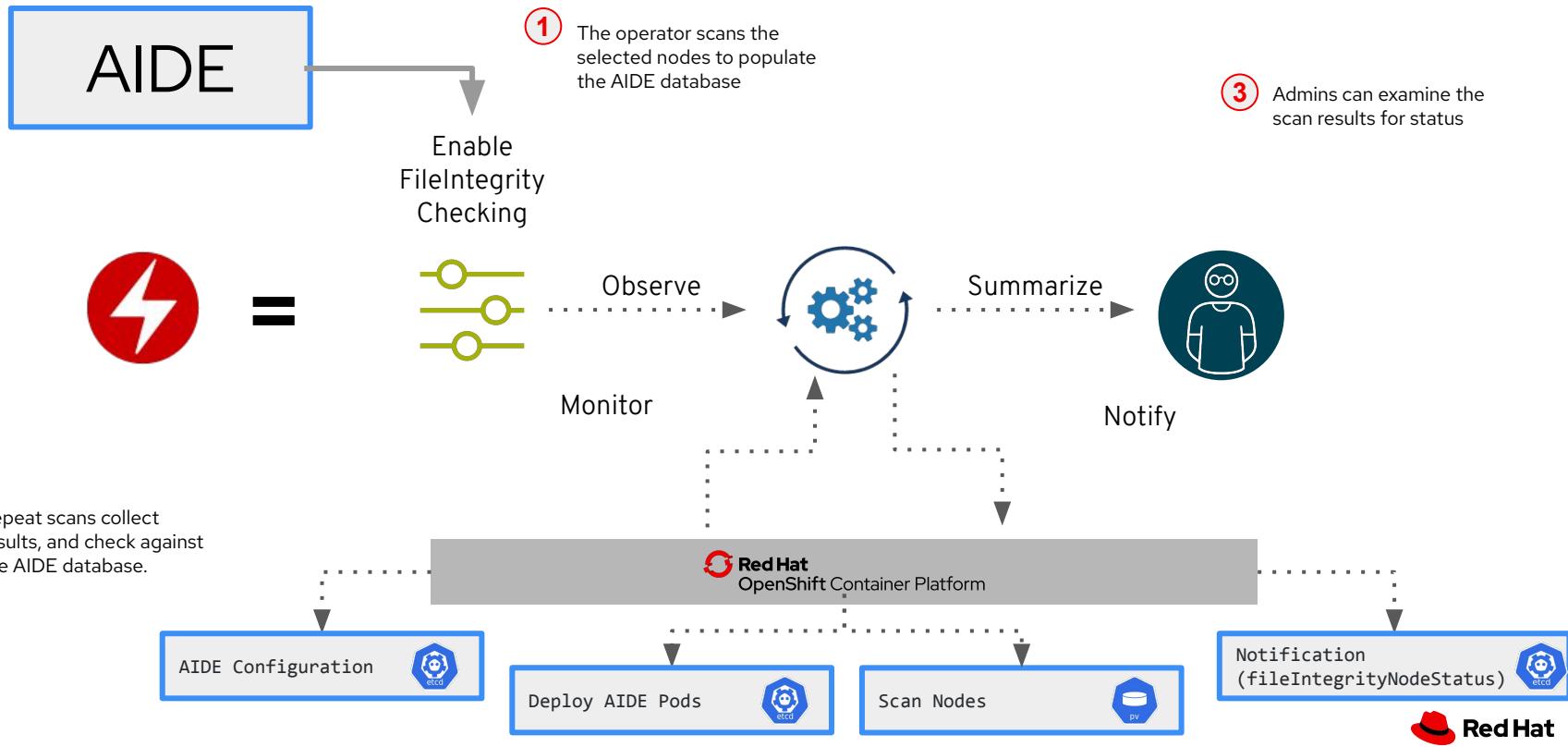
**Fixable Vulnerabilities:** A list of 16 fixable vulnerabilities, all marked as "Successfully assigned" and dated 16:00.

**Log:** A log window showing recent events, all marked as "Successfully assigned".

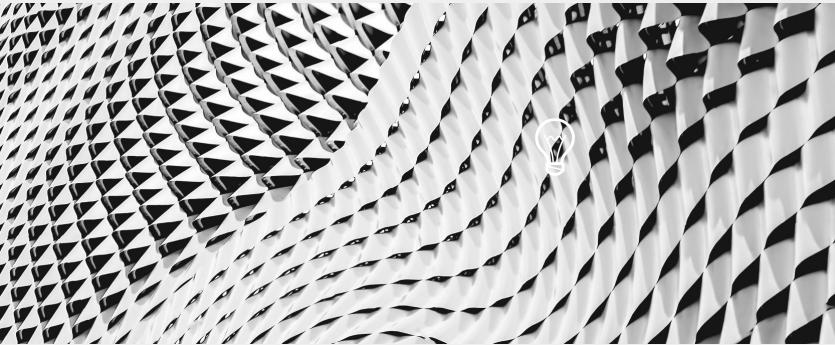
# Openshift Compliance Operator: Declarative Security Compliance



# Openshift File Integrity Operator

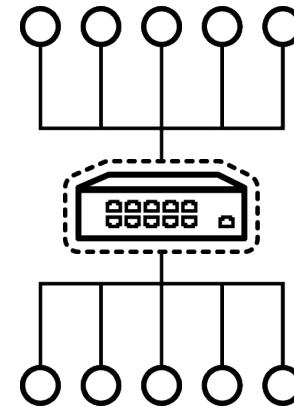


# DEFEND THE NETWORK

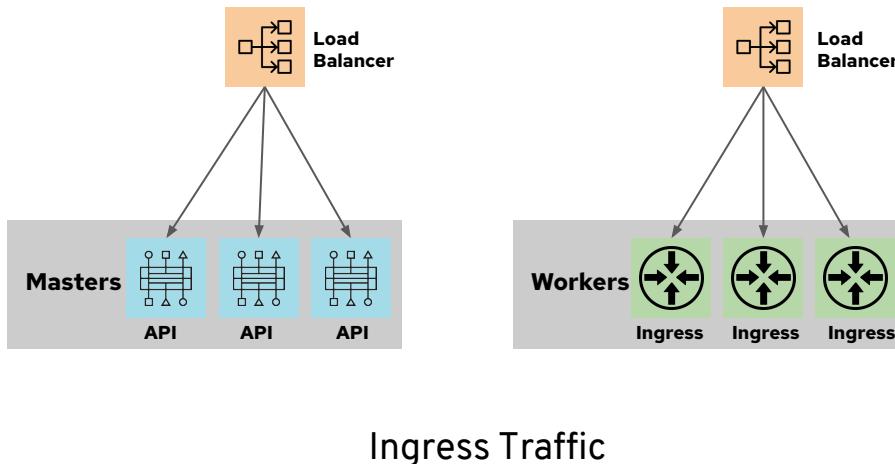


# OpenShift networking

- Built-in internal DNS to reach services by name
- OpenShift follows the Kubernetes Container Networking Interface (CNI) plug-in model
- Software Defined Networking (SDN) for a unified cluster network to enable pod-to-pod communication
- Network Policies to isolate applications from other applications within a cluster
- Service Mesh provides advanced capabilities for managing service to service communication
- Multiple options for ingress and egress control



# External Access to Cluster Resources



- Two primary entry points into OpenShift
  - API
  - Ingress/Router
- Proper DNS entries must be configured
- Additional ingress types available
  - NodePort (requires additional port resources)
  - LoadBalancer

# Ingress NodePortService

Some admins don't want OpenShift to manage a cloud load balancer and DNS for their IngressControllers.

They may want IngressControllers to be exposed through node ports to enable custom integration with a front-end load balancing solution.

The NodePortService endpoint publishing strategy publishes the Ingress Controller using a [Kubernetes NodePort service](#), which enables:

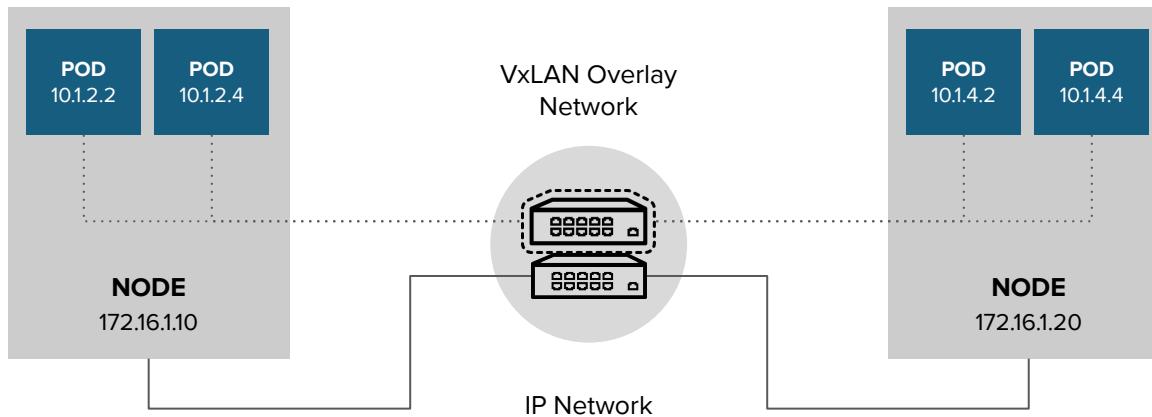
- alternate forms of ingress for better support of bare metal deployments
- use cases requiring more than one instance of the HAProxy router per node

```
apiVersion: v1
kind: Service
metadata:
  name: router-default
  namespace: openshift-ingress
  annotations:
    operator.openshift.io/node-port-service-for: default
spec:
  type: NodePort
  externalTrafficPolicy: Local
  ports:
  - name: http
    port: 80
    protocol: TCP
    targetPort: http
  - name: https
    port: 443
    protocol: TCP
    targetPort: https
  selector:
    ingresscontroller.operator.openshift.io/deployment-ingresscontroller: default
```

# Routes vs Ingress

Feature	Ingress	Route
Standard Kubernetes object	X	
External access to services	X	X
Persistent (sticky) sessions	X	X
Load-balancing strategies (e.g. round robin)	X	X
Rate-limit and throttling	X	X
IP whitelisting	X	X
TLS edge termination	X	X
TLS re-encryption	X	X
TLS passthrough	X	X
Multiple weighted backends (split traffic)		X
Generated pattern-based hostnames		X
Wildcard domains		X

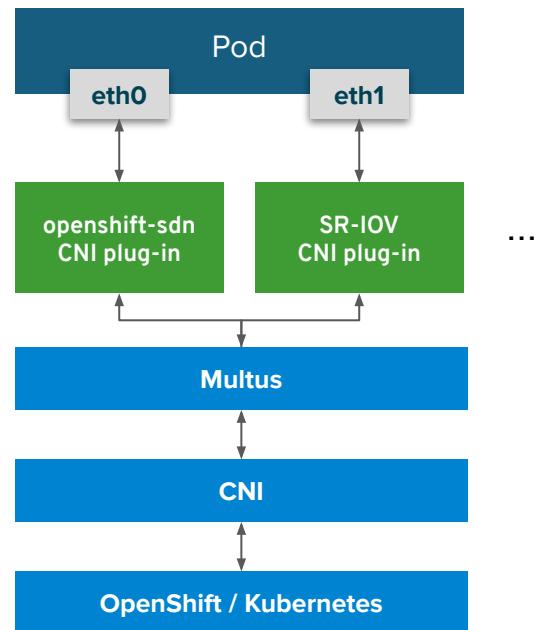
# OpenShift pod to pod networking



# Multus

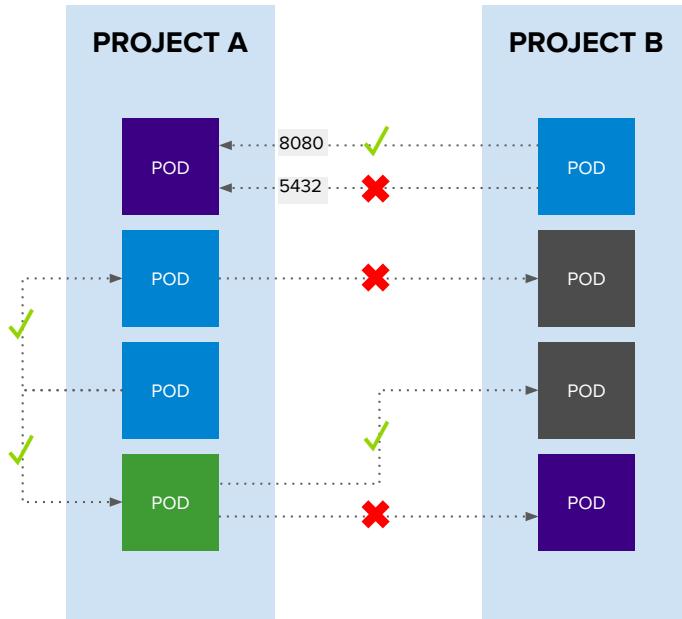
- Problem: Kubernetes only supports one network interface, "eth0", but we need:
  - Functional separation of control/data planes
  - Link aggregation for network redundancy
  - Different network protocol stacks, capabilities, SLAs
  - Traffic isolation / Network segregation and security
  - QoS
- Solution: Multus "meta plug-in" for Kubernetes CNI
- Enables multiple network interfaces per pod, each assigned a different CNI plug-in defined in pod spec
  - Each with its configuration defined in CRD objects
- SR-IOV enablement

**Pod with Multus**



# OpenShift SDN

## Network policy enabled by default



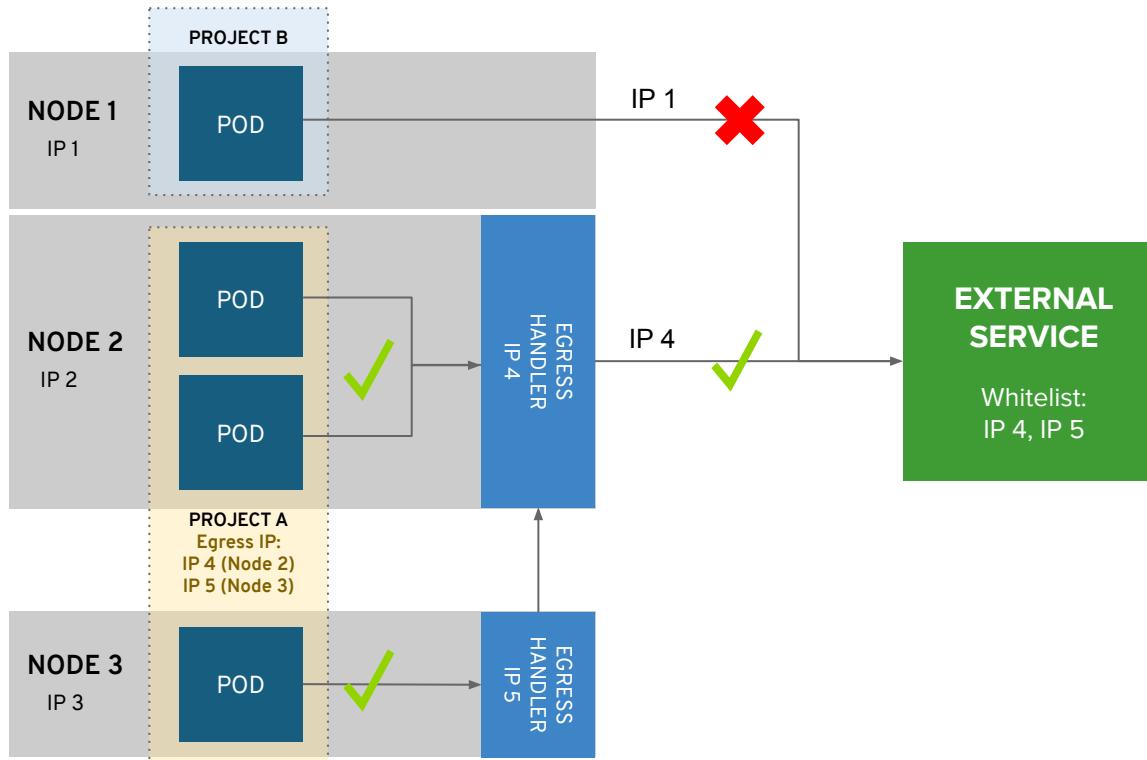
### Example Policies

- 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
```

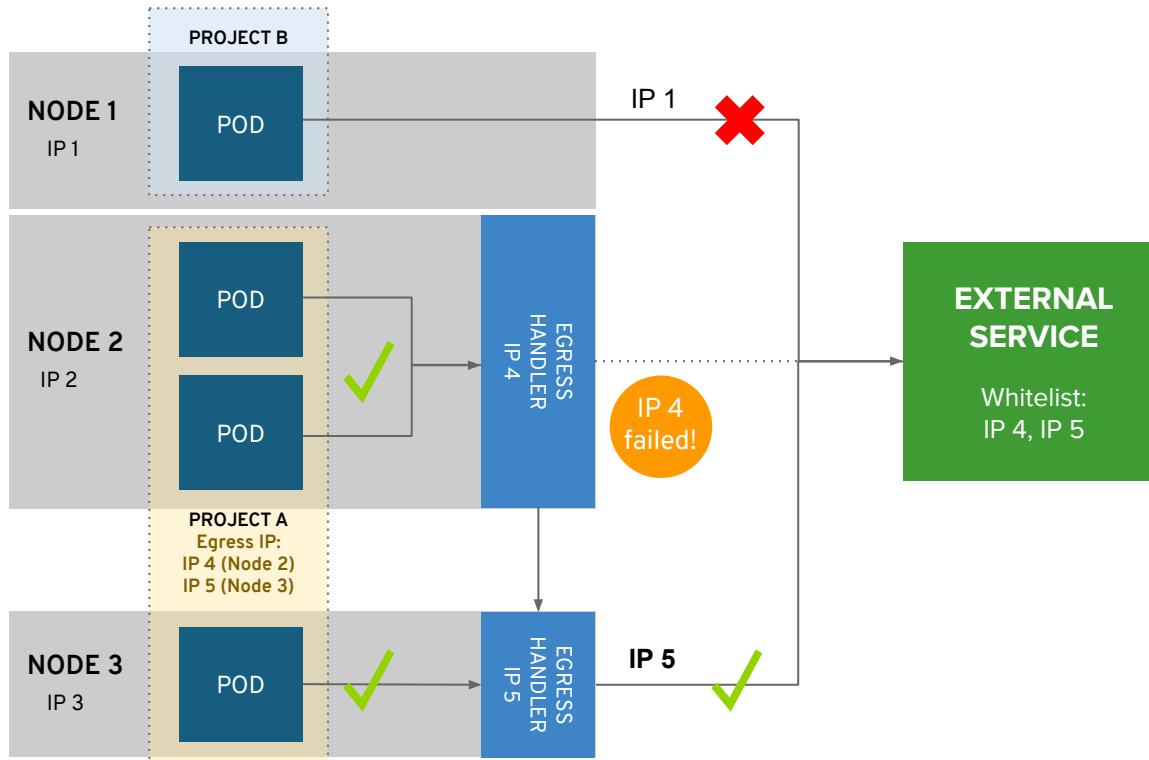
# Controlling Egress Traffic

Egress IP high availability (multiple IPs)

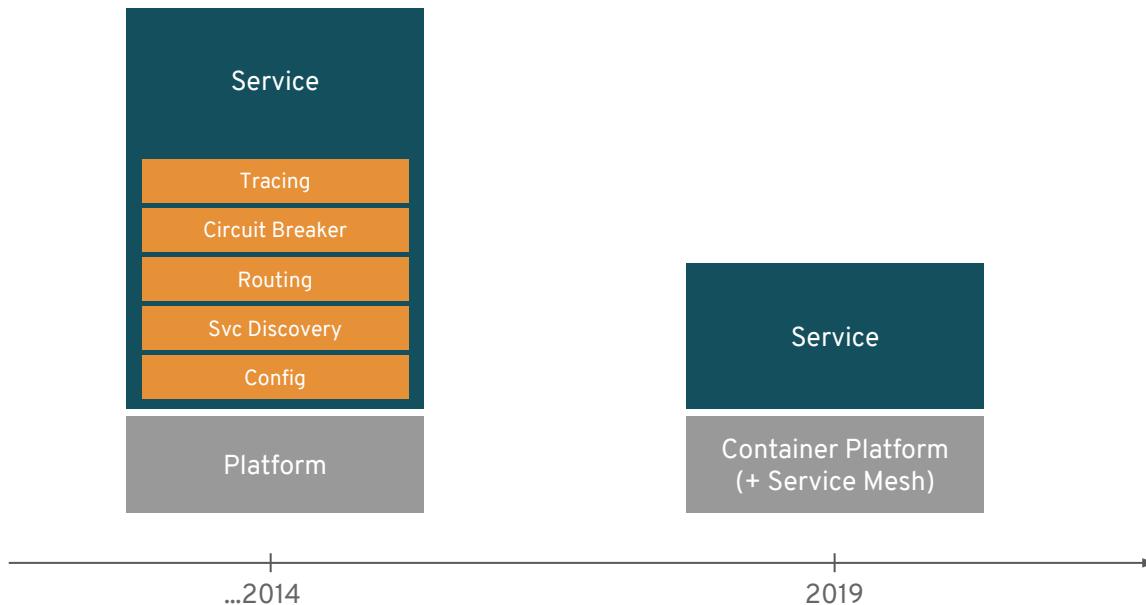


# Controlling Egress Traffic

Egress IP high availability (multiple IPs)



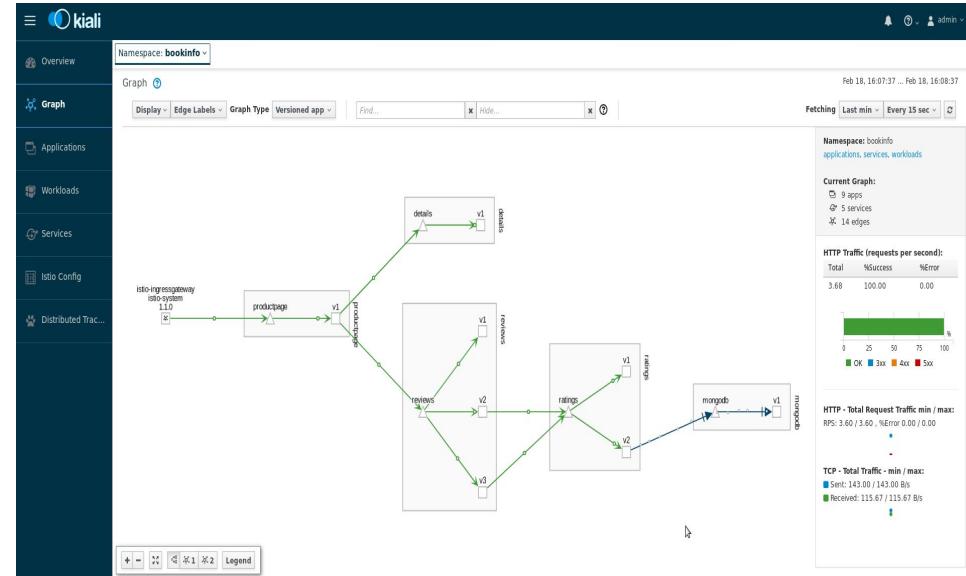
# Microservices evolution



# Secure microservices with Service Mesh

## Key Features

- A dedicated network for service to service communications
- Observability and distributed tracing
- Policy-driven security
- Routing rules & chaos engineering
- Powerful visualization & monitoring
- Will be available via OperatorHub



# Observability with Kiali

Namespace: **bookinfo**

Graph ?

Display Edge Labels Graph Type Versioned app Find... Hide... ?

Fetching Last min Every 15 sec ?

Feb 18, 16:07:37 ... Feb 18, 16:08:37

Namespace: bookinfo applications, services, workloads

Current Graph:

- 9 apps
- 5 services
- 14 edges

HTTP Traffic (requests per second):

Total	%Success	%Error
3.68	100.00	0.00

OK 3xx 4xx 5xx

HTTP - Total Request Traffic min / max: RPS: 3.60 / 3.60, %Error 0.00 / 0.00

TCP - Total Traffic - min / max:

- Sent: 143.00 / 143.00 B/s
- Received: 115.67 / 115.67 B/s

istio-ingressgateway istio-system 1.1.0

productpage

v1

reviews

v1

v2

v3

ratings

v1

v2

mongodb

details

v1

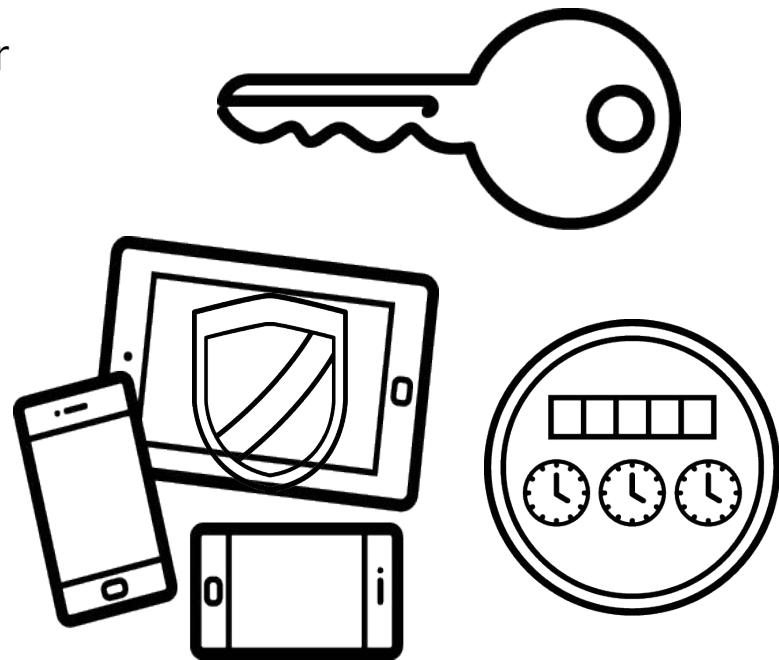
details

+ - X X 1 X 2 Legend

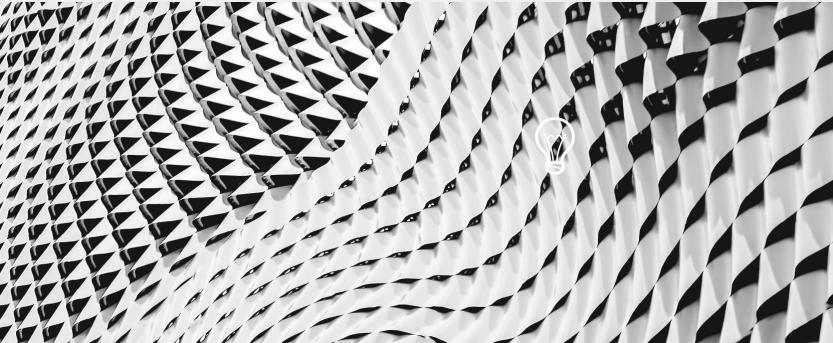
# Application API management

Consider configuring an API gateway for container platform & application APIs

- Authentication and authorization
- LDAP integration
- End-point access controls
- Rate limiting



# CONTROL APPLICATION SECURITY



# DEVSECOPS

## THROUGH THE ADOPTION OF CONTAINERS

We created Dev and Ops and Security user stories and tackled them together.



DEVELOPER

I can break builds if security and compliance rules aren't followed...



SECURITY

We're empowering the developers and ideally empowering them straight to production.



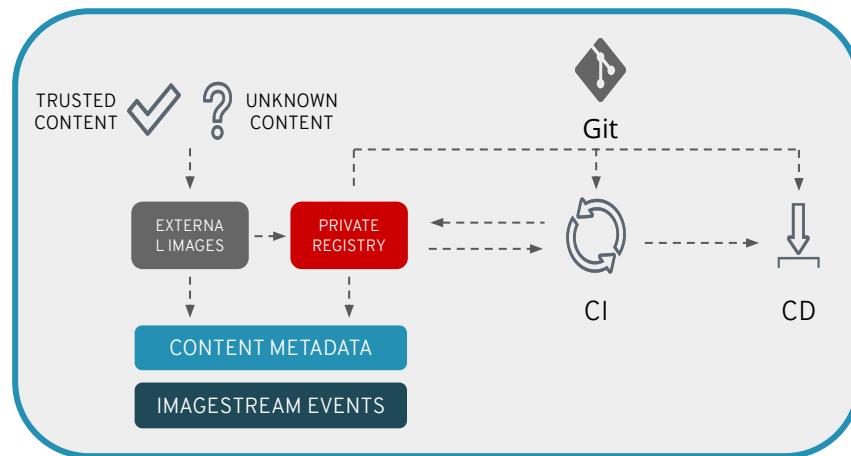
OPERATIONS

# Securing Containerized Applications

## An opportunity to shift security left

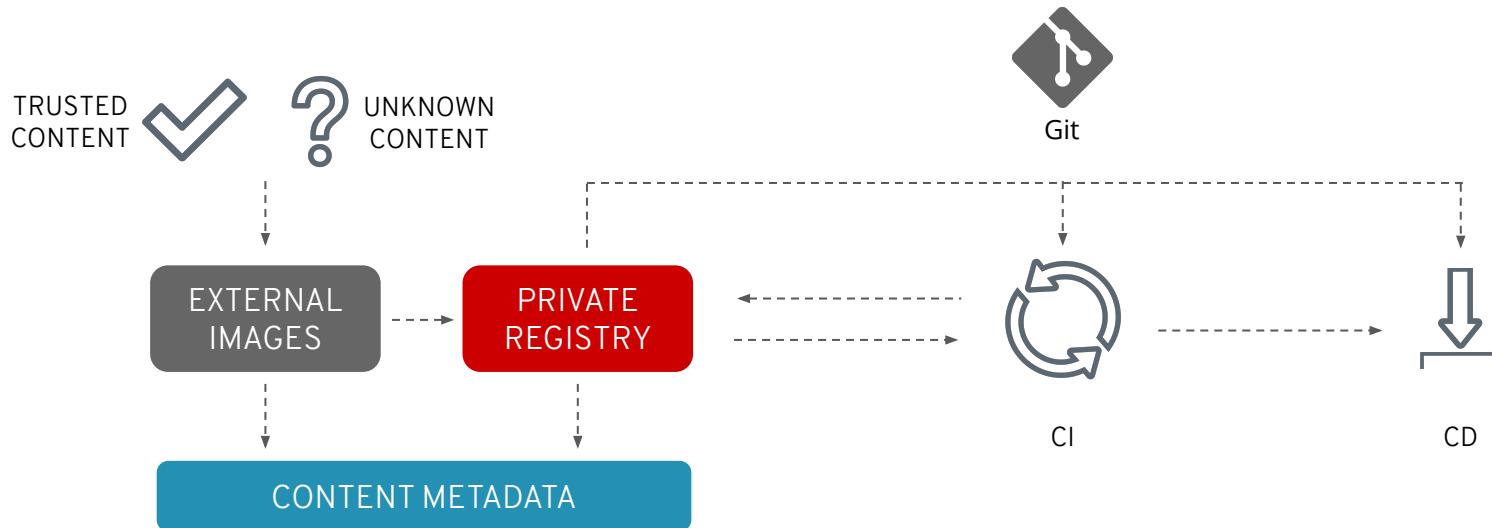
### Best practices

- Use trusted sources for external content
- Use a private registry to manage images
- CI/CD must have security gates
- Application secrets management
- Apply runtime security policies
- Rebuild and redeploy - never patch a running container
- Ensure application logging, monitoring

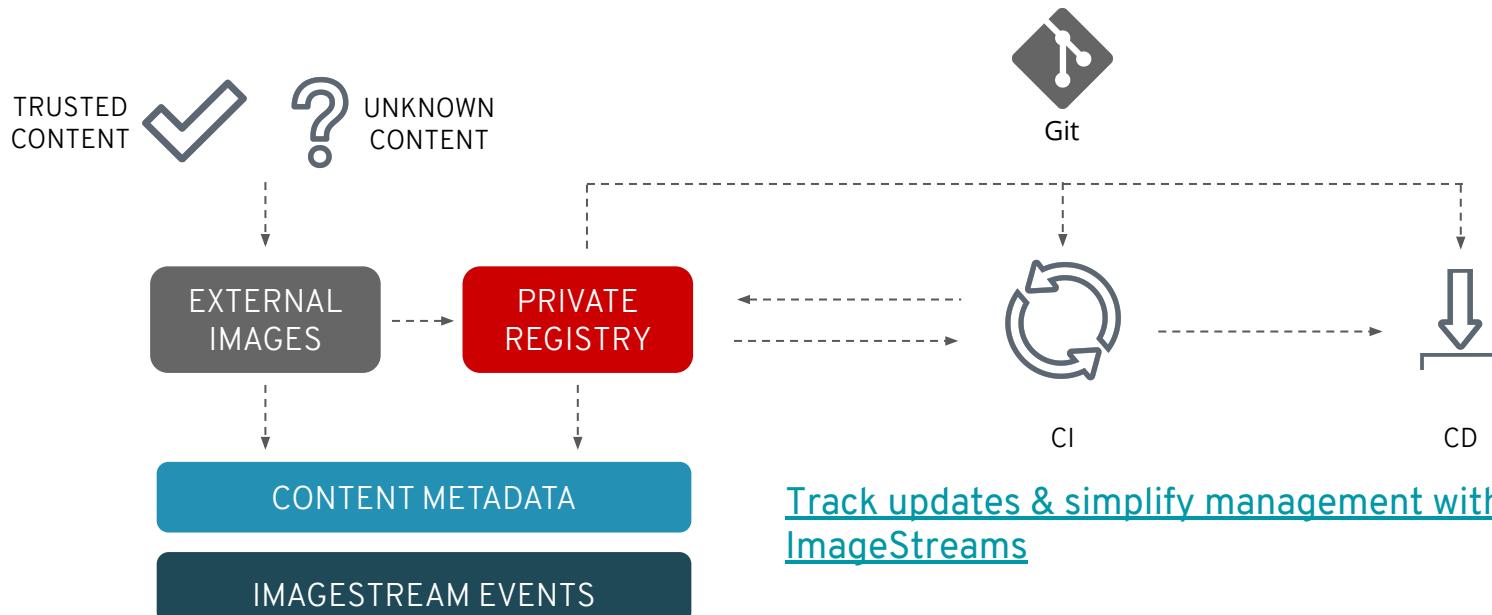


# Secure & Automate The Content Lifecycle

Elements of the Openshift container pipeline



# Trust is temporal: rebuild and redeploy as needed



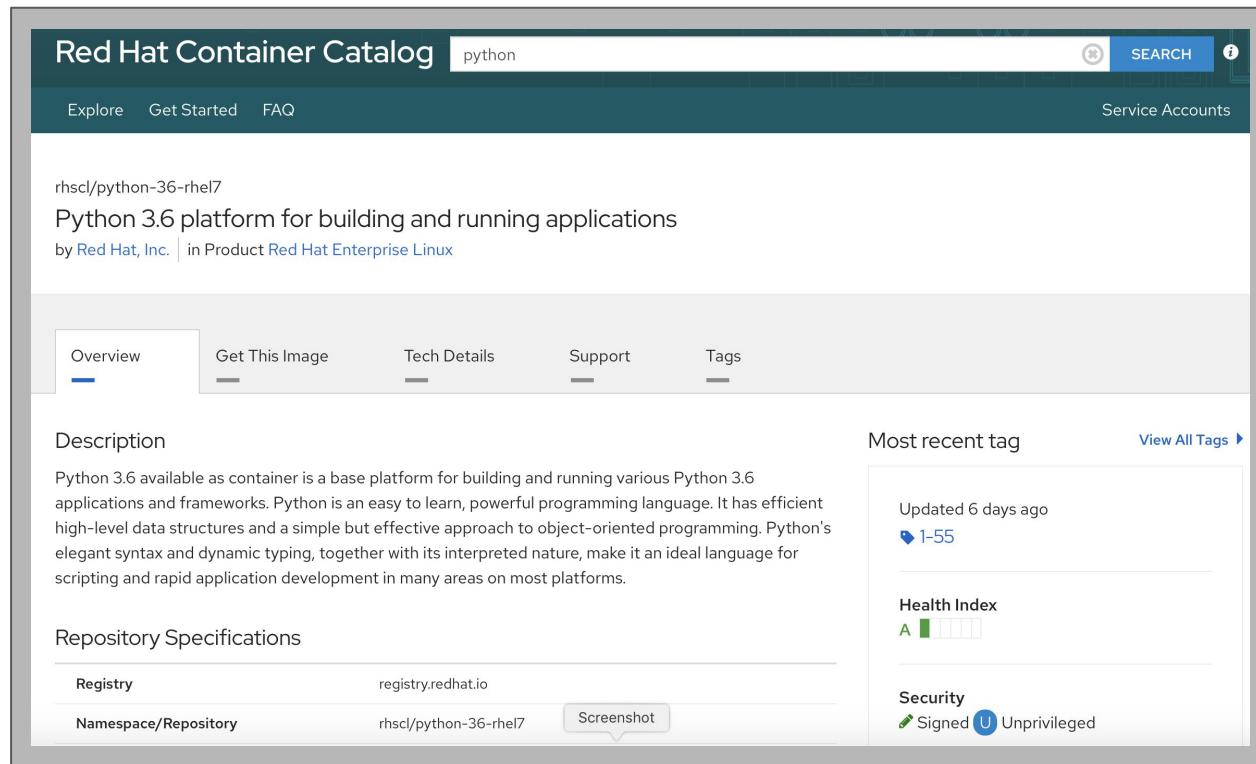
[Track updates & simplify management with ImageStreams](#)

Use [Image Change Triggers](#) to automatically rebuild custom images with updated (patched) external images

# External Content: Use Trusted Sources

## Red Hat Container Images

- Signed Images
- Health Index (A to F grade)\*
- Security advisories & errata (patches)



The screenshot shows the Red Hat Container Catalog interface. The search bar at the top contains the text "python". The main content area displays the "rhscl/python-36-rhel7" image. The description states: "Python 3.6 platform for building and running applications" by Red Hat, Inc. in Product Red Hat Enterprise Linux. Below the description are tabs for "Overview" (selected), "Get This Image", "Tech Details", "Support", and "Tags". The "Description" section contains a detailed paragraph about Python 3.6. The "Repository Specifications" section shows the registry as "registry.redhat.io" and the namespace/repository as "rhscl/python-36-rhel7". A "Screenshot" button is also present. To the right, a sidebar shows the "Most recent tag" as "1-55" (updated 6 days ago) and a "Health Index" of "A" (green). Below that is a "Security" section indicating the image is "Signed" and "Unprivileged".

# Red Hat Quay

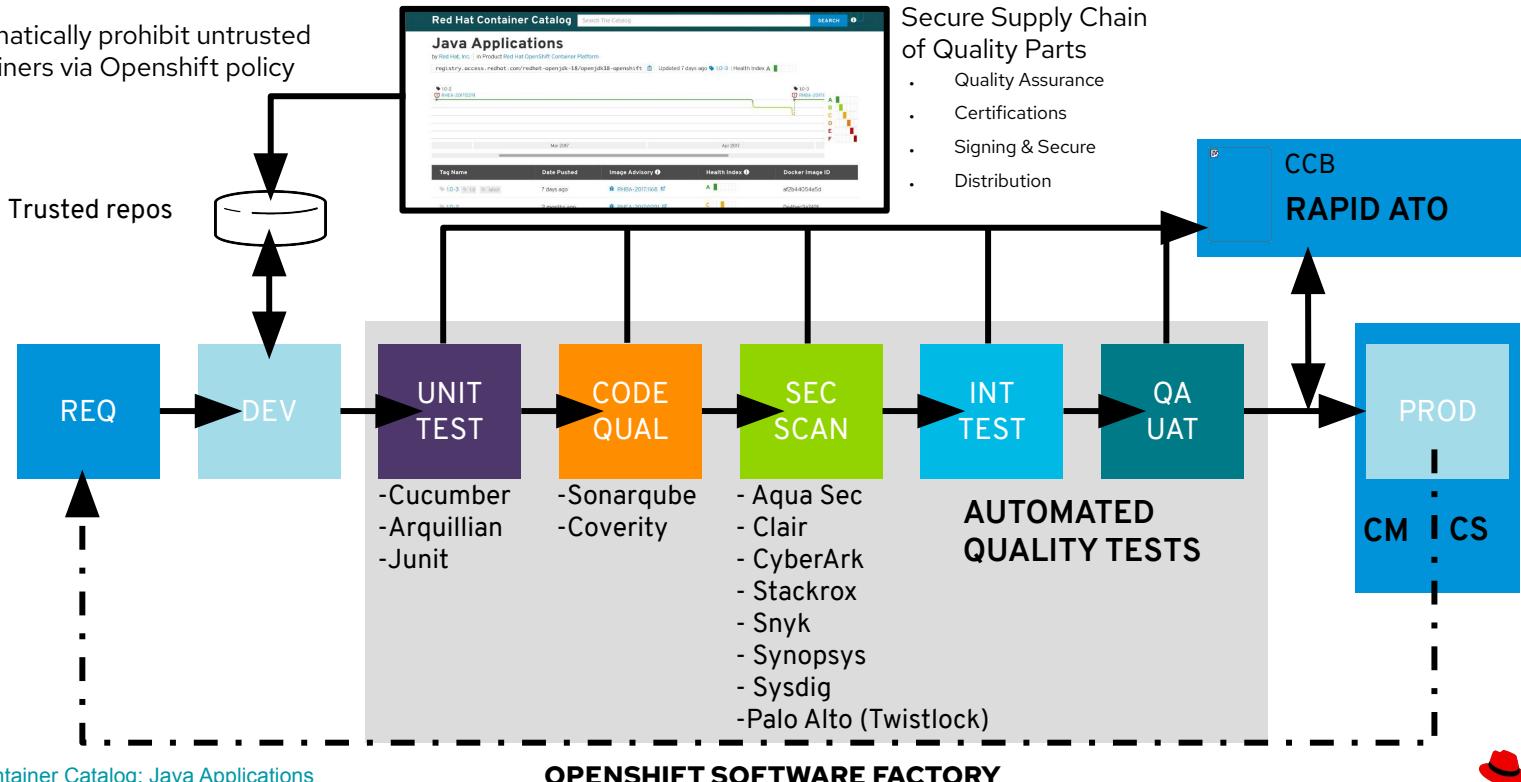
## Enterprise Container Registry

- Offered as self-managed and as-a-service
  - Vulnerability Scanning (Clair)
  - Geographic Replication
  - Build Image Triggers
  - Image Rollback with Time Machine

# Integrate Security in your CI/CD Pipeline

**Automated quality and security: because you can't inspect quality into a product**

Automatically prohibit untrusted containers via Openshift policy

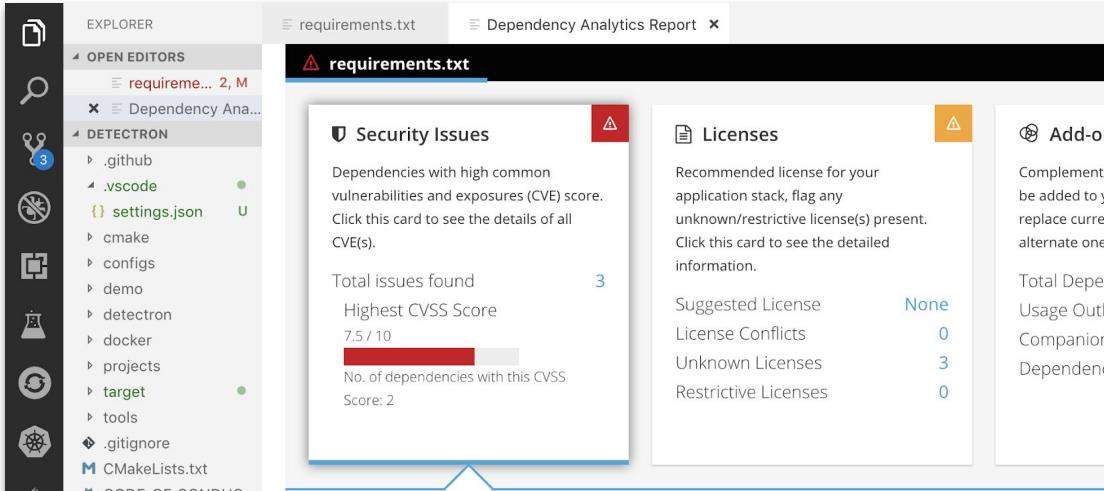


# Enhancing Secure Application Development and DevSecOps

“Shift Left” - find CVEs and license issues during development

Red Hat Dependency Analytics IDE plugins provide security and license warnings for any project dependency:

- Be notified of CVEs in any package or sub-package
- Remediation advice (upgrade / downgrade)
- Uses open source and Snyk CVE databases
- Supported for Java, Node, Python



The screenshot shows the Red Hat Dependency Analytics IDE interface. The left sidebar displays project files and dependencies. The main area shows a 'requirements.txt' file with a red warning icon. A central card displays 'Security Issues' with a total of 3 issues found, the highest CVSS Score of 7.5/10, and 2 dependencies affected. To the right, a 'Licenses' card shows recommended licenses and conflicts. Below these are tabs for 'Direct Dependencies with Security Issues' (2) and 'Transitive Dependencies with Security Issues' (1). A table at the bottom lists dependencies with their CVE counts and highest CVSS scores.

**Dependencies with security issues in your stack**

A list of the dependencies affected with common vulnerabilities and exposures (CVE), dependency with the highest common reporting the issues.

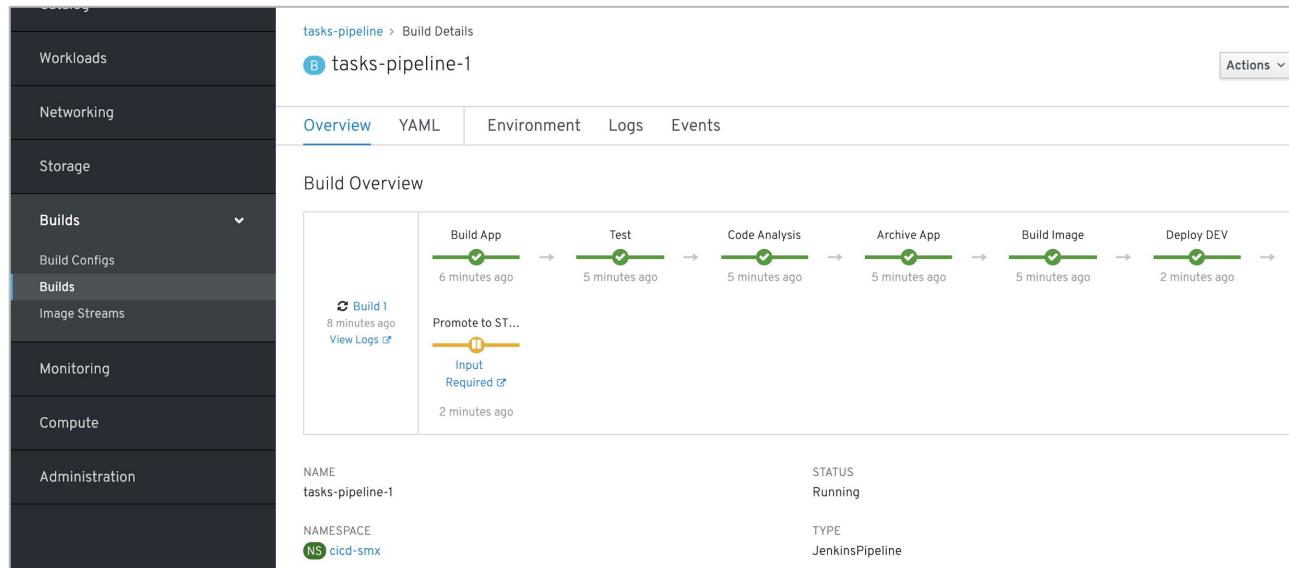
#	Dependencies	No. of CVE(s)	Highest CVSS Score
1	pyyaml	1	7.5/10

# Jenkins CI/CD, run in OpenShift and deploy to OpenShift

Jenkins is still the most used CI/CD platform in enterprises and can be used from inside OpenShift.

An intuitive pipeline visualization makes it simple for users to see how builds are progressing.

The full Jenkins UI is also available.



The screenshot shows the OpenShift web interface. On the left, a sidebar menu is open under the 'Builds' section, showing options like 'Builds', 'Build Configs', 'Image Streams', 'Monitoring', 'Compute', and 'Administration'. The main content area is titled 'tasks-pipeline > Build Details' for 'tasks-pipeline-1'. It shows a 'Build Overview' with a horizontal timeline of pipeline stages: 'Build App', 'Test', 'Code Analysis', 'Archive App', 'Build Image', and 'Deploy DEV'. Each stage is represented by a green circle with a checkmark, indicating success. The 'Build App' stage was 6 minutes ago, 'Test' was 5 minutes ago, 'Code Analysis' was 5 minutes ago, 'Archive App' was 5 minutes ago, 'Build Image' was 5 minutes ago, and 'Deploy DEV' was 2 minutes ago. Below the timeline, detailed information is provided: NAME: tasks-pipeline-1, STATUS: Running; and NAMESPACE: NS cicd-smx, TYPE: JenkinsPipeline.

**Why?** Build in, or for, OpenShift from your enterprise CI/CD system.

# OpenShift Pipelines: A Kubernetes-native CI/CD platform

Provides a next-gen Kubernetes CI/CD pipeline that works for containers (including serverless).

Based on the Tekton project (which was spun out of the Knative Pipelines project) started by Google, Red Hat and others.

Target general availability in OpenShift 4.7.

```

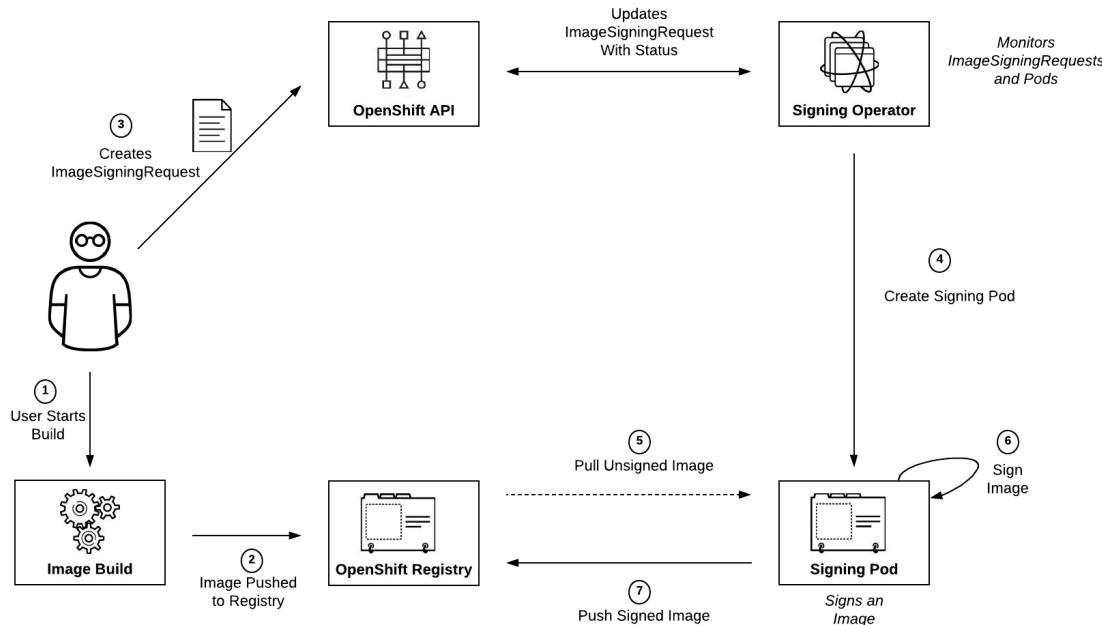
  Downloading six-1.11.0-py2.py3-none-any.whl
  Building wheels for collected packages: tornado, configparser
  Running setup.py bdist_wheel for tornado: started
  Running setup.py bdist_wheel for tornado: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/0c21/02/8cd6a381458df92b449ea7c57be653dd7aa80ba42c716212c
  Running setup.py bdist_wheel for configparser: started
  Running setup.py bdist_wheel for configparser: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/1c/bd/b4/277af3f6c40845651b4cd1c21df26aca0f2e1e9714a1d4cd8
  Successfully built tornado configparser
  Installing collected packages: six, singledispatch, certifi, backports-abc, tornado, enum34, configparser, mccabe, pyflakes, pycodestyle, flake8
  Found existing installation: six 1.8.0
  Uninstalling six-1.8.0:
  Successfully uninstalled six-1.8.0
  Successfully installed backports-abc-0.5 certifi-2017.11.5 configparser-3.5.0 enum34-1.1.6 flake8-3.5.0 mccabe-0.6.1 pycodestyle-2.3.1 pyflakes-1.6.0
  singledispatch-3.4.0.3 six-1.11.0 tornado-4.5.3
  $ python -c "print('Hello, world')"
  Hello, world
  Job succeeded

```

**Why?** A faster, less resource-intensive CI/CD platform that's Kubernetes-native.

# Container Signing

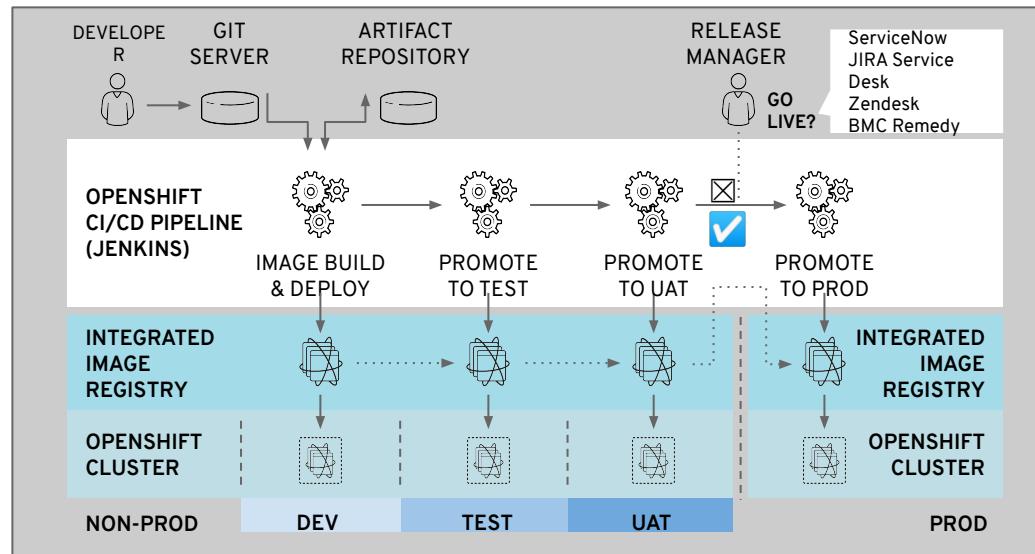
A simplified and automated approach to signing container images



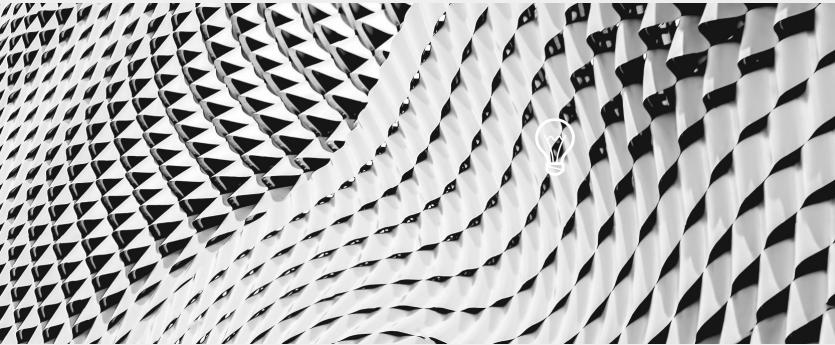
# Summary

## Managing container deployment

- Deployments: Containerized App Configuration as Code
- Whitelist / Blacklist external repos
- Apply runtime security policies
- Validate image signatures
- Monitor for new vulnerabilities
- Trust is temporal:  
rebuild & redeploy as needed



# EXTEND SECURITY



# The Security Ecosystem

For enhanced security, or to meet existing policies, you may choose to integrate with enterprise security tools, such as

- Identity and Access management / Privileged Access Management
- External Certificate Authorities
- External Vaults / Key Management solutions
- Filesystem encryption tools
- Container content scanners & vulnerability management tools
- Container runtime analysis tools
- Security Information and Event Monitoring (SIEM)

# Red Hat OpenShift certified operators - Security



Red Hat Advanced Cluster Manager		
Platform Services	Application Services	Developer Services
Service Mesh : Serverless Builds : CI/CD Pipelines Full Stack Logging Chargeback	Databases : Languages Runtimes : Integration Business Automation 100+ ISV Services	Developer CLI : VS Code extensions : IDE Plugins Code Ready Workspaces CodeReady Containers
Cluster Services		
Automated Ops : Over-The-Air Updates : Monitoring : Registry : Networking : Router : KubeVirt : OLM : Helm		
Kubernetes		
Red Hat Enterprise Linux CoreOS		
 Physical	 Virtual	 Private cloud
 Public cloud	 Managed cloud (Azure, AWS, GCP, IBM, Red Hat)	 Edge cloud

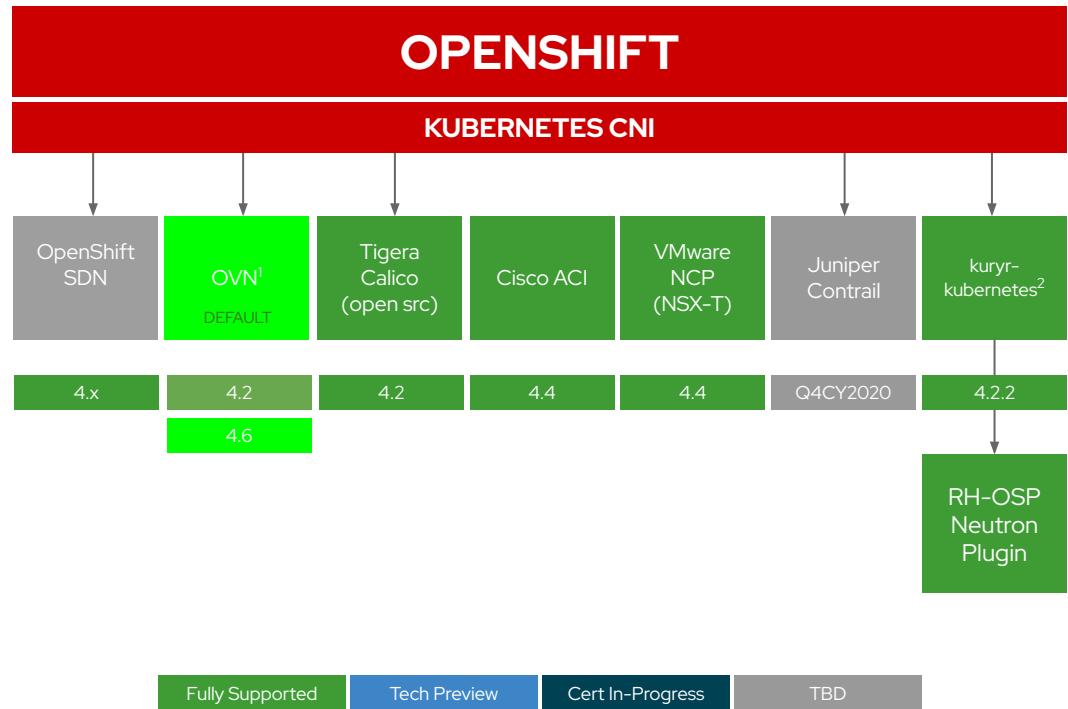
# 3rd-Party CNI Plug-in Certification Update

The following 3rd-party Kubernetes CNI plug-ins have begun the OpenShift certification process and are at varying stages of progress:

-  **ACI**
-  **NSX-T**

The certification process primarily consists of:

1. Formalizing the partnership
2. Certifying the container(s)
3. Certifying the Operator
4. Successfully passing the same Kubernetes networking conformance tests that OpenShift uses to validate its own SDN





**Red Hat**

Advanced Cluster Security  
for Kubernetes

StackRox | Red Hat ACS

Kubernetes is the standard  
for application innovation...



- ▶ Microservices architecture
- ▶ Declarative definition
- ▶ Immutable infrastructure

...and Kubernetes-native  
security is increasingly critical



- ▶ Secure supply chain
- ▶ Secure infrastructure
- ▶ Secure workloads

DevOps

DevSecOps

Security

# Benefits of a Kubernetes-native approach to security



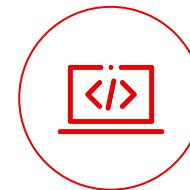
## Lower operational cost

DevOps and Security teams can use a common language and source of truth



## Reduce operational risk

Ensure alignment between security and infrastructure to reduce application downtime



## Increase developer productivity

Leverage Kubernetes to seamlessly provide guardrails supporting developer velocity

# Red Hat Advanced Cluster Security for Kubernetes

A cloud workload protection platform and cloud security posture management to enable you to “shift left”

## Shift left

Secure supply chain

Extend scanning and compliance into development (DevSecOps)

## Cloud security posture management (CSPM)

Secure infrastructure

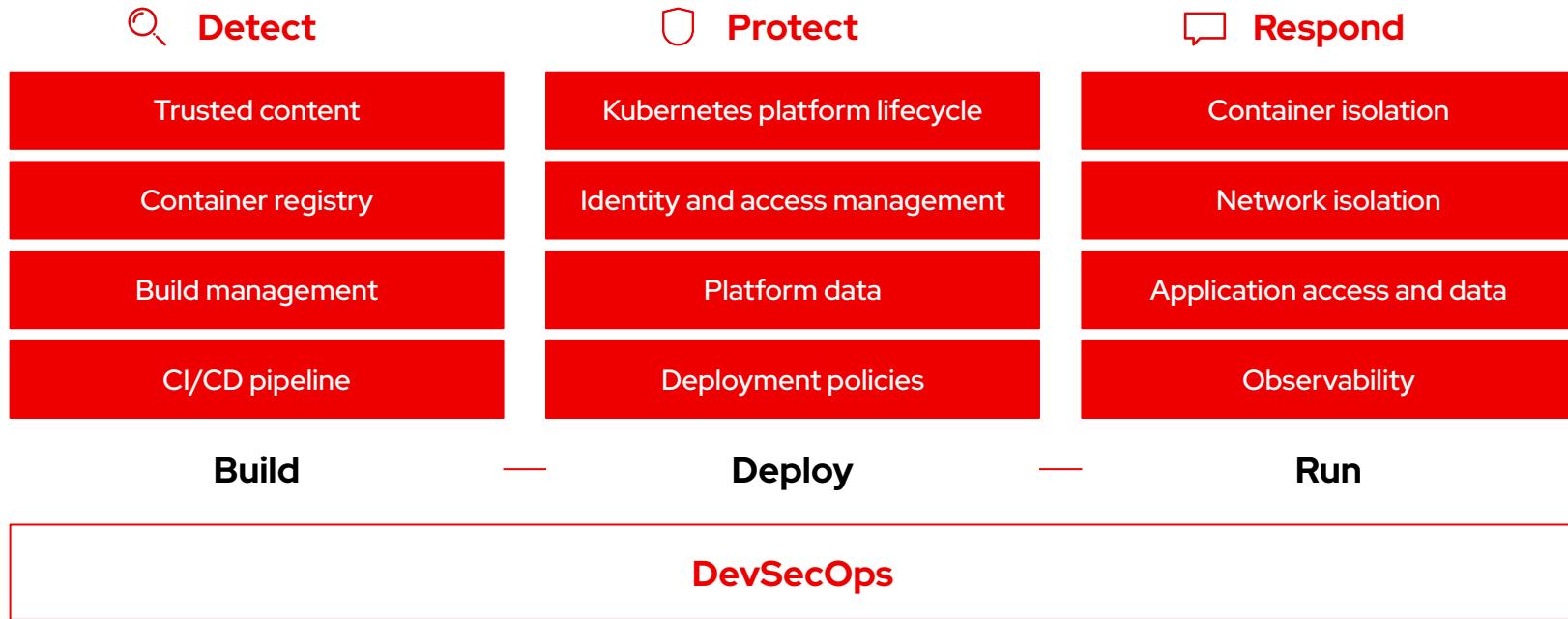
Leverage built-in Kubernetes CSPM to identify and remediate risky configurations

## Cloud workload protection (CWPP)

Secure workloads

Maintain and enforce a “zero-trust execution” approach to workload protection

# Red Hat OpenShift provides a secure foundation



# RHACS delivers security depth to entire application lifecycle

## Detect

Trusted content
Container registry
Build management
CI/CD pipeline

## Protect

Kubernetes platform lifecycle
Identity and access management
Platform data
Deployment policies

## Respond

Container isolation
Network isolation
Application access and data
Observability



Vulnerability analysis

App config analysis

APIs for CI/CD integrations

Image assurance and policy admission controller

Compliance assessments

Risk profiling

Runtime behavioral analysis

Auto-suggest network policies

Threat detection / incident response

Build

Deploy

Run

DevSecOps

# RHACS

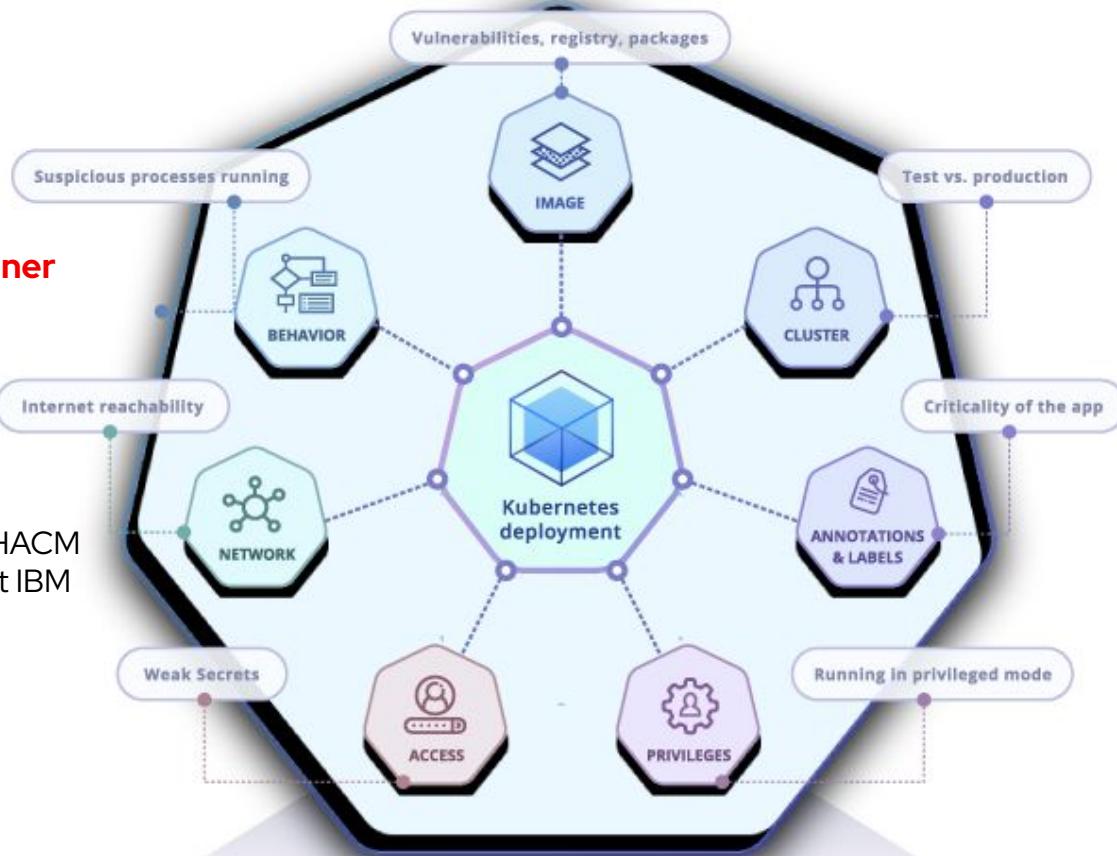
## Securing Kubernetes Deployments

### It's all about the Application in the container

- plus a Registry Scanner.

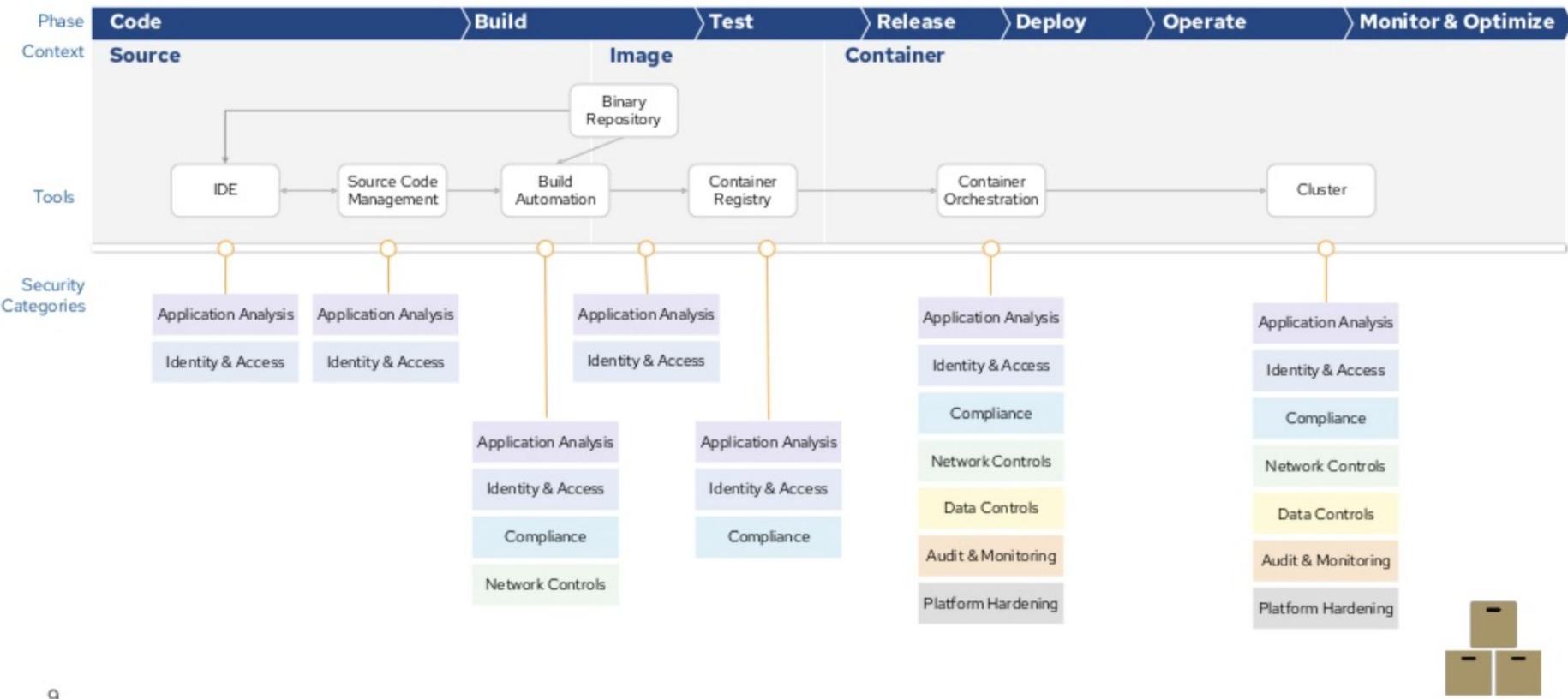
### It's not:

- End 2 End Monitoring -> Dynatrace
- Infrastructure Monitoring - RHACM
- Infrastructure Compliance Monitoring - RHACM
- Access Control / Audit to and in OpenShift IBM QRadar or CyberARC
- SIEM Solution -> Splunk
- Certificate Management - Cert Manager
- API Management - 3scale
- Application Performance Management
- Registry - QUAY
- Service MESH

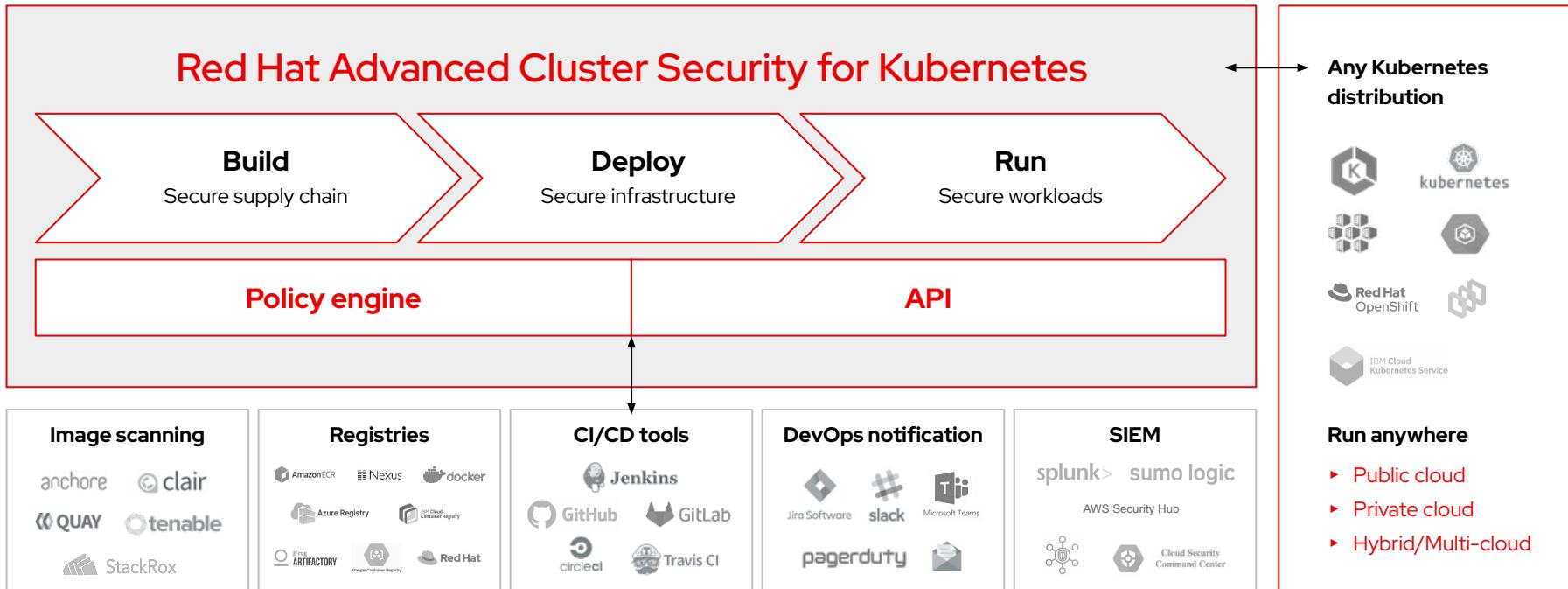


# RHACS

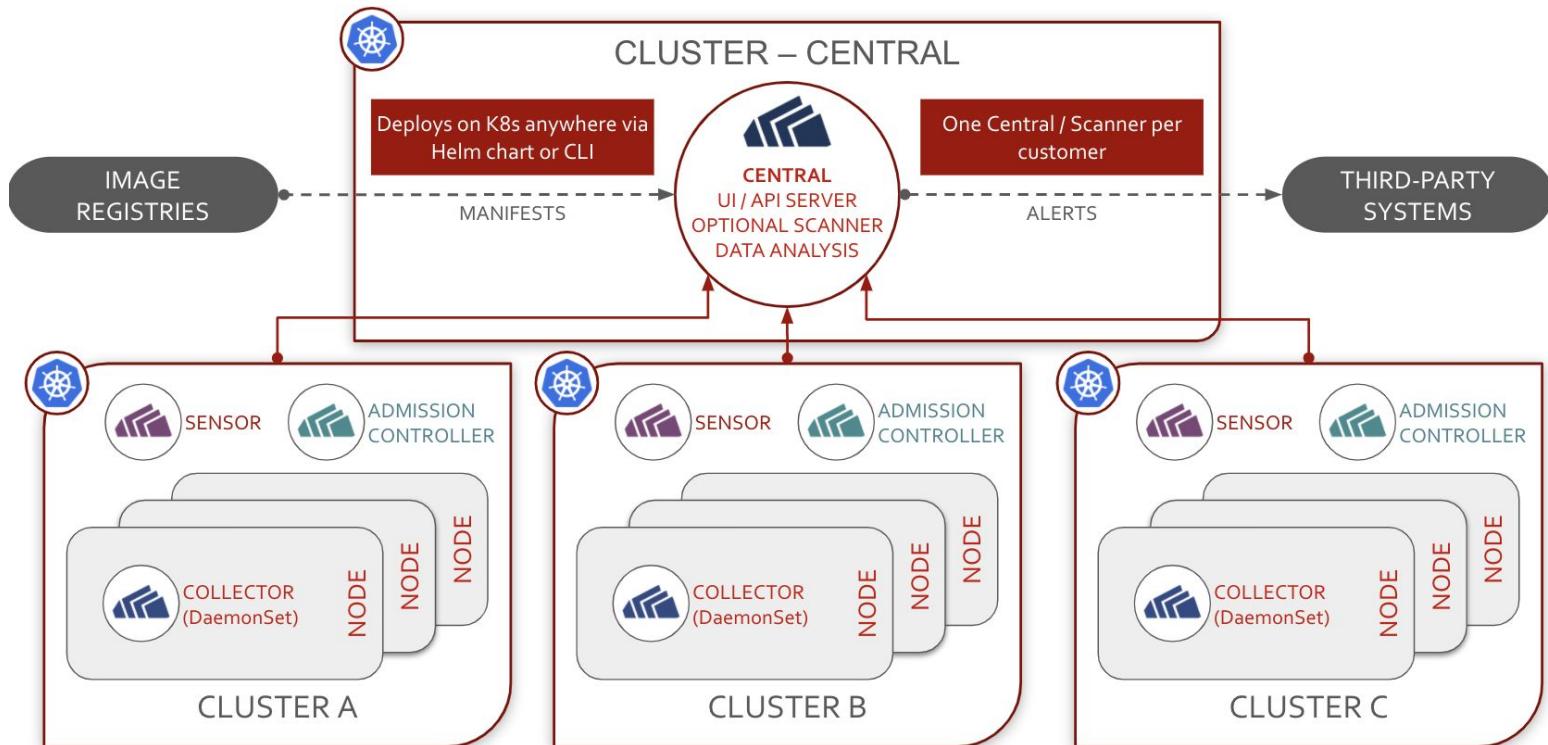
adding security to dev ops for your kubernetes native applications



# The first Kubernetes-native security platform



# Architecture



# Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.



[linkedin.com/company/red-hat](https://www.linkedin.com/company/red-hat)



[youtube.com/user/RedHatVideos](https://www.youtube.com/user/RedHatVideos)



[facebook.com/redhatinc](https://www.facebook.com/redhatinc)



[twitter.com/RedHat](https://twitter.com/RedHat)