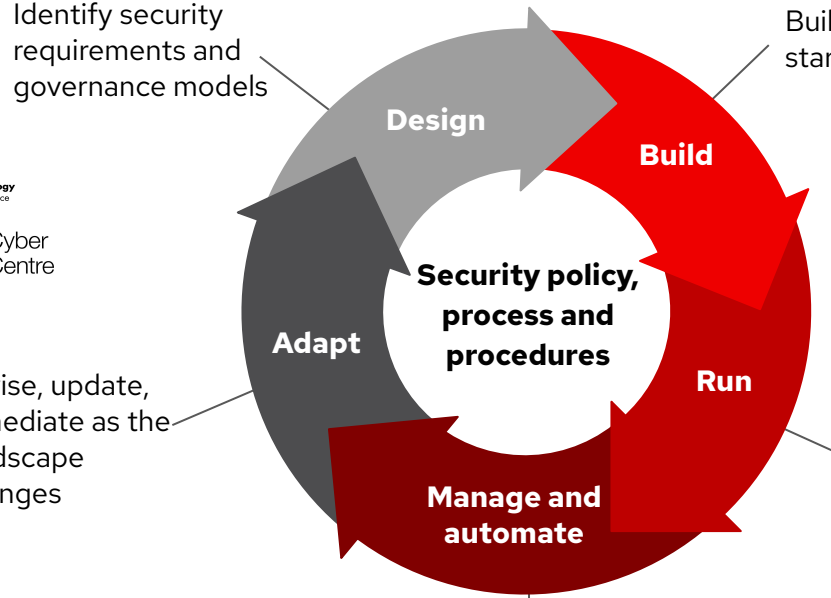
SECURING CONTAINERS WITH OPENSHIFT PLATFORM PLUS

<name>
<title>
<date>

Red Hat

Security must be continuous and holistic



OpenShift 4

Considerations for Securing Containers and Kubernetes

NIST 800-190

"Use container-specific host OSs instead of general-purpose ones to reduce attack surfaces."

CNCF Kube Security Audit

"...the underlying hosts, components, and environment of a Kubernetes cluster must be configured and managed. This management has a direct impact on the capabilities of the cluster..."

Gartner Market Guide for Cloud Workload Protection

"The best way to secure these rapidly changing and short-lived workloads is to start their protection proactively in the development phase ..."

"Replace antivirus (AV)-centric strategies with a "zero-trust execution"/default deny/application control approach to workload protection where possible...."

Sources

NIST Special Publication 800-190 *Application Container Security Guide*

CNCF Cloud Native Security Whitepaper

Kubernetes Security Whitepaper, *Trail of Bits, May 31, 2019*

Gartner: Market Guide for Cloud Workload Protection Platforms, ID G00356240, April 8, 2019

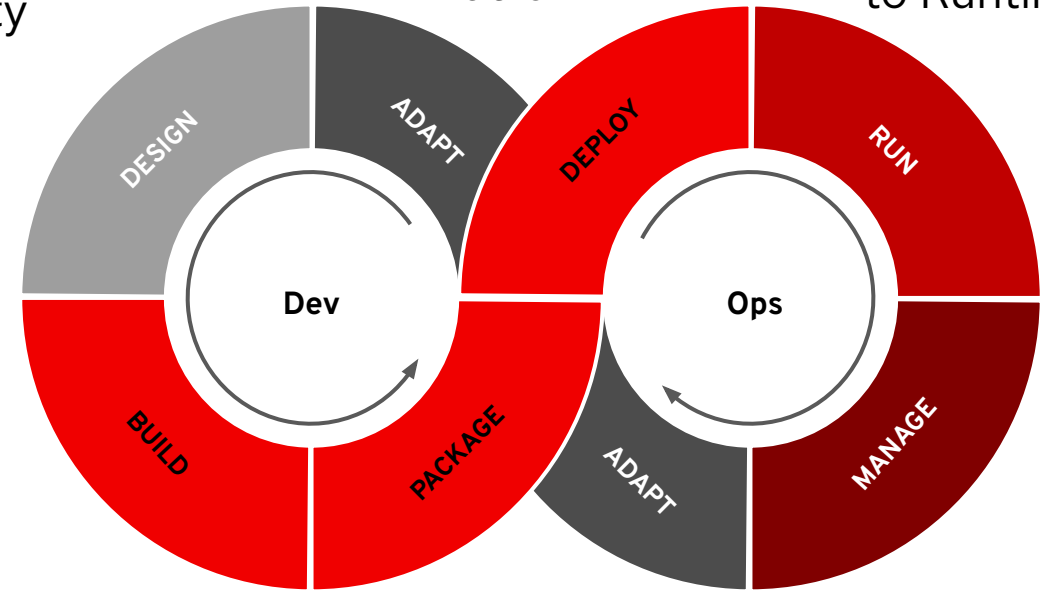
Red Hat

Containers and Kubernetes need DevSecOps

Control Application Security

Protect the Platform

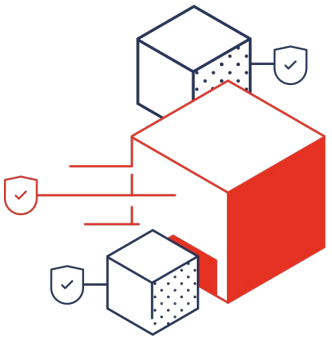
Detect & Respond to Runtime Threats



Red Hat

The OpenShift platform vision:

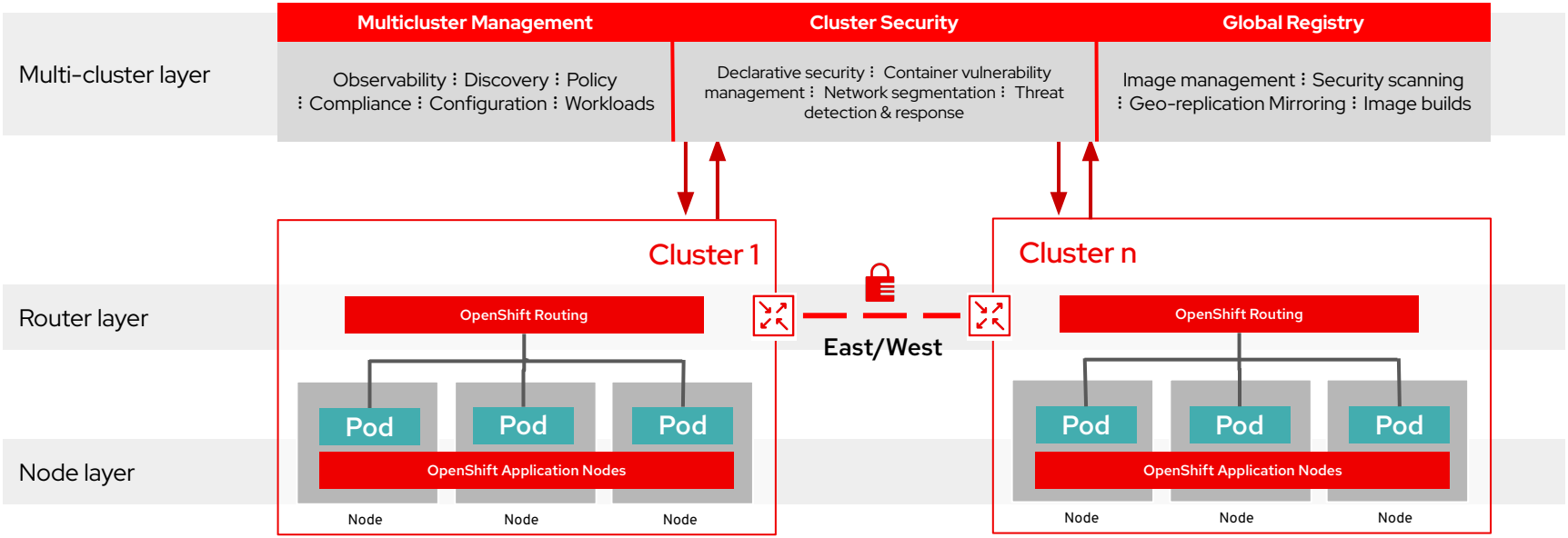
A single hybrid-cloud platform for enterprises to build, deploy, run and manage intelligent applications securely at scale.



Red Hat

Red Hat OpenShift Platform Plus

Enabling Hybrid and Multi-Cloud Deployments



Red Hat

Build: Control application security

Shift Security left

Best practices

Red Hat UBI

- Use trusted sources for external content such as base images

Quay

- Use a trusted private registry to manage supply chain risk

OCP Pipelines

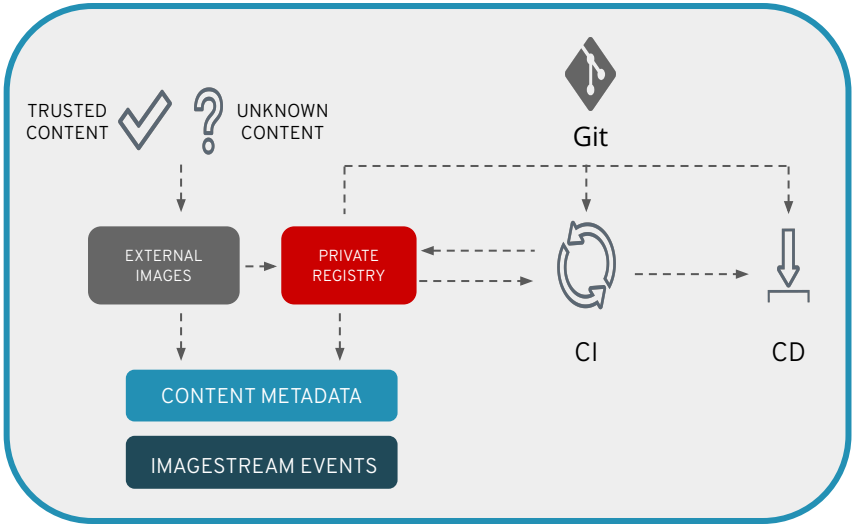
- Automate your CI/CD pipeline to enable rapid updates

Quay scanner (registry)
Code Ready (IDE)
ACS scanner (CI)
KubeLinter (CI)

- Integrate security tools / gates in your pipeline to identify
  - Known vulnerabilities
  - Application misconfigurations

ACM

- Use policy-based deployment tools to manage application placement (e.g. locality)



Red Hat

Deploy: Protect the application platform

Best practices

RHEL CoreOS
- Reduce attack surface with a container optimized operating system

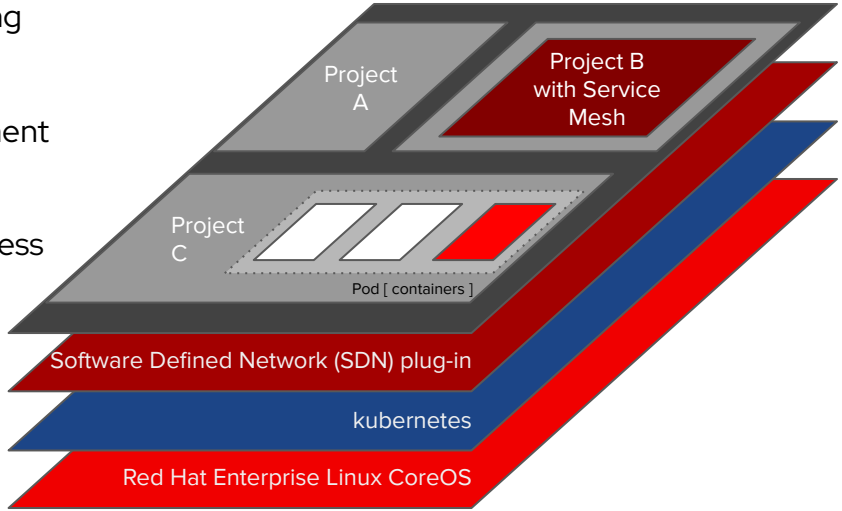
OCP Operators
ACM
- Use automated and policy-driven configuration management across your fleet

OCP RBAC
ACS to monitor
ACM to enforce
- Implement least privilege with fine-grained role based access control (RBAC)

OCP CAs
Service mesh
OCP IPSec
RHCOS NBDE
Encrypt etcd
- Encrypt platform data in transit and at rest

OCP Compliance Operator
ACS
ACM
- Use automated compliance, risk assessment and remediation solutions

OCP Security Context Constraints
ACS
- Reduce deployment risk with admission control policies that
  - Minimize admission of privileged pods, pods with host capabilities
  - Prevent admission of pods with critical vulnerabilities



Red Hat

Run: Securing the container runtime

Best practices

OCP ACS
- Minimize the impact of an attack by isolating running applications with
  - SELinux & Security Context Constraints
  - Kubernetes namespaces (Projects), RBAC
  - Network Policies for microsegmentation

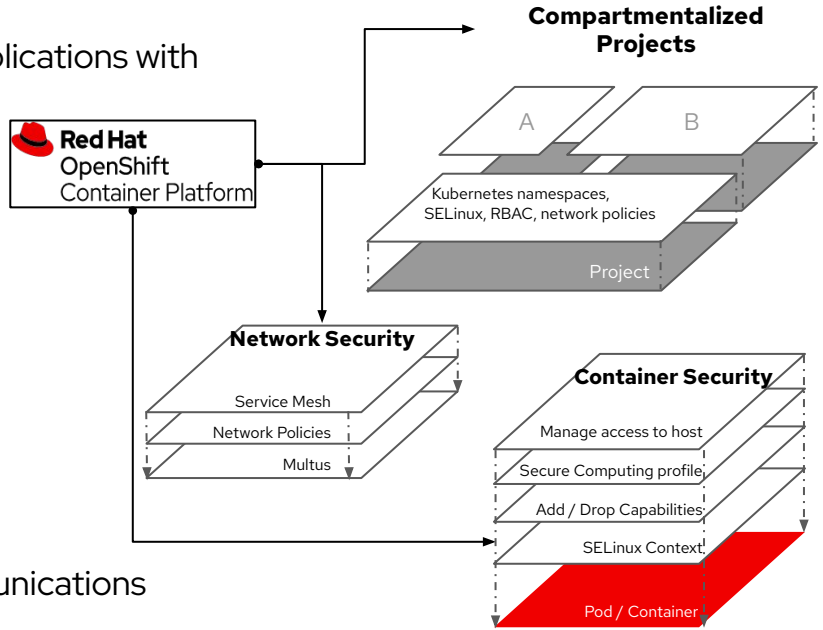
OCP ACM
- Use resource quotas to prevent resource exhaustion

- Manage application access and protect application data

OCP
  - Red Hat Single Sign On for user management
  - Secure routes / ingress, 3Scale API Gateway
  - Service mesh to encrypt pod-to-pod traffic
  - Egress IPs / firewall

OCP ACS ACM
- Monitor application metrics, logging and network communications

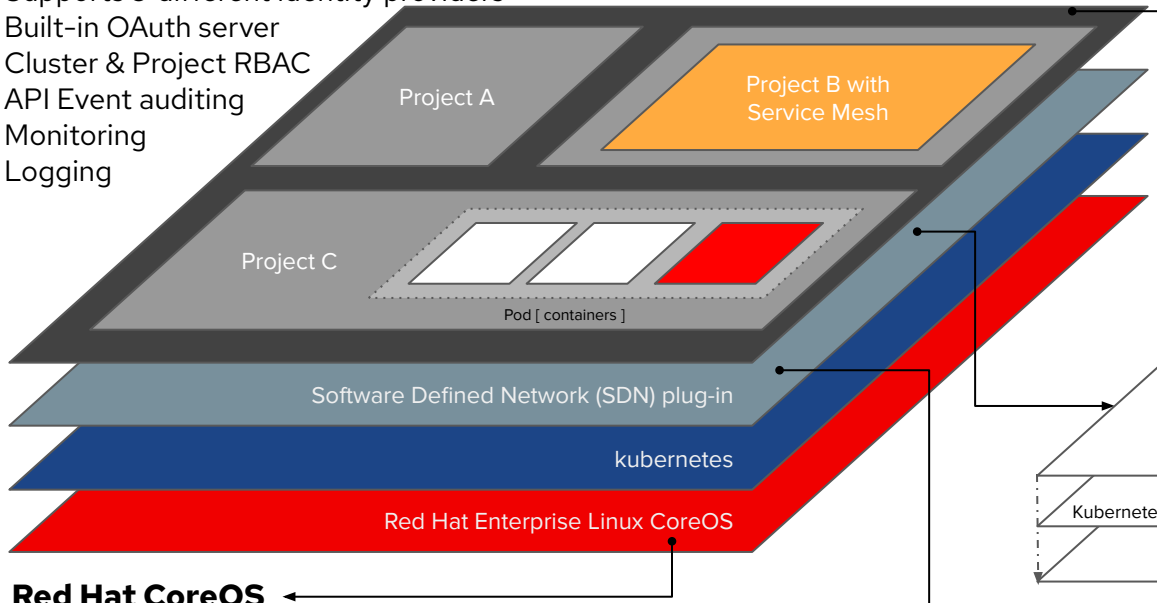
ACS
- Automate threat detection and response
  - Alert or kill pods based on anomalous behavior
  - Detect privilege escalation and risky processes such as cryptomining



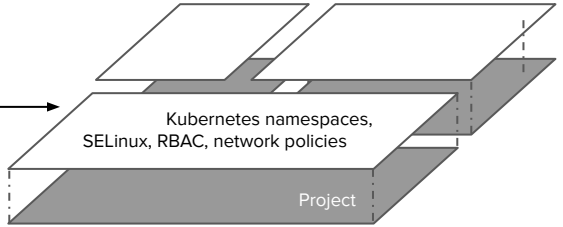
Red Hat

Defense in Depth

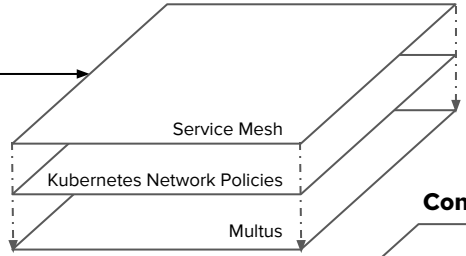
OpenShift Container Platform
- Supports 9 different identity providers
- Built-in OAuth server
- Cluster & Project RBAC
- API Event auditing
- Monitoring
- Logging



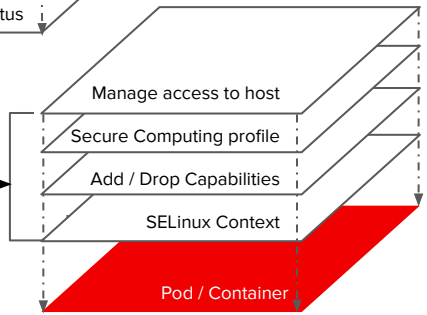
Compartmentalized Projects



Network Security



Container Security



Red Hat CoreOS
- Minimized attack surface
- Controlled Immutability
- SELinux on by default
- Kernel namespaces and Cgroups
- CRI-O container runtime, Kubelet
- Auditd for host-level audit
- FIPS enablement
- RHCOS volume encryption

Security Context Constraints
(Kubernetes Admission Controller)

Red Hat Universal Base Image
containers can leverage RHEL
FIPS capabilities

Red Hat

Hardening, applicability guides, certifications

OpenShift 4 self managed

- HIPAA product applicability guide
- Architecting for HIPAA Security Rule
- ISO 27001 product applicability guide
- FISMA product applicability guide
- PCI-DSS Architecture Design Guide
- The OpenShift Security Guide
- OpenShift 4 Hardening Guide

  (ask RH for a copy)
- CIS OpenShift Benchmark

  (search for Kubernetes then expand)

OpenShift managed services certifications

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

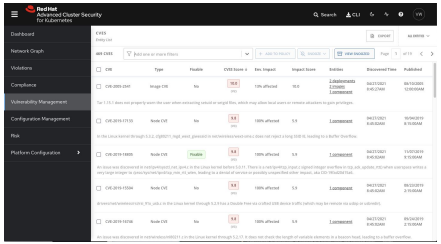
Red Hat Advanced Cluster Security for Kubernetes

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

Shift left

Supply chain security

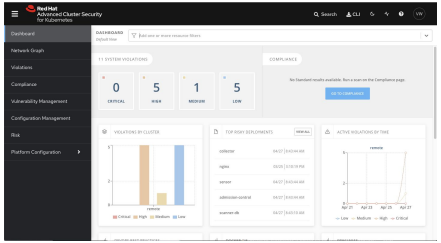
Extend scanning and compliance into development (DevSecOps)



Kubernetes security posture management (KSPM)

Securing infrastructure

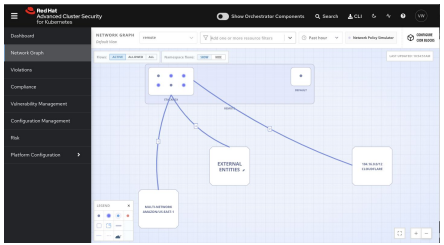
Leverage built-in Kubernetes security posture management to identify and remediate configurations and deployments



Cloud workload protection (CWPP)

Securing workloads

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



Red Hat Advanced Cluster Security: Use Cases

Security across the entire application lifecycle

Vulnerability Management

Protect yourself against known vulnerabilities in images and running containers

Network Segmentation

Apply and manage network isolation and access controls for each application

Configuration Management

Ensure your deployments are configured according to security best practices

Compliance

Meet contractual and regulatory requirements and easily audit against them

Risk Profiling

Gain context to prioritize security issues throughout OpenShift and Kubernetes clusters

Detection and Response

Carry out incident response to address active threats in your environment

Advanced Cluster Management

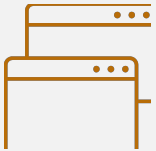
Application-centric Management
Deploy, upgrade, and manage applications with consistency across multiple clouds

Policy-Based Governance
Enforce configuration policies and ensure compliance across clusters, applications and infrastructures

Cluster Lifecycle Management
Centrally, create, update, delete clusters across the enterprise

Multicluster Management

Infrastructure Management



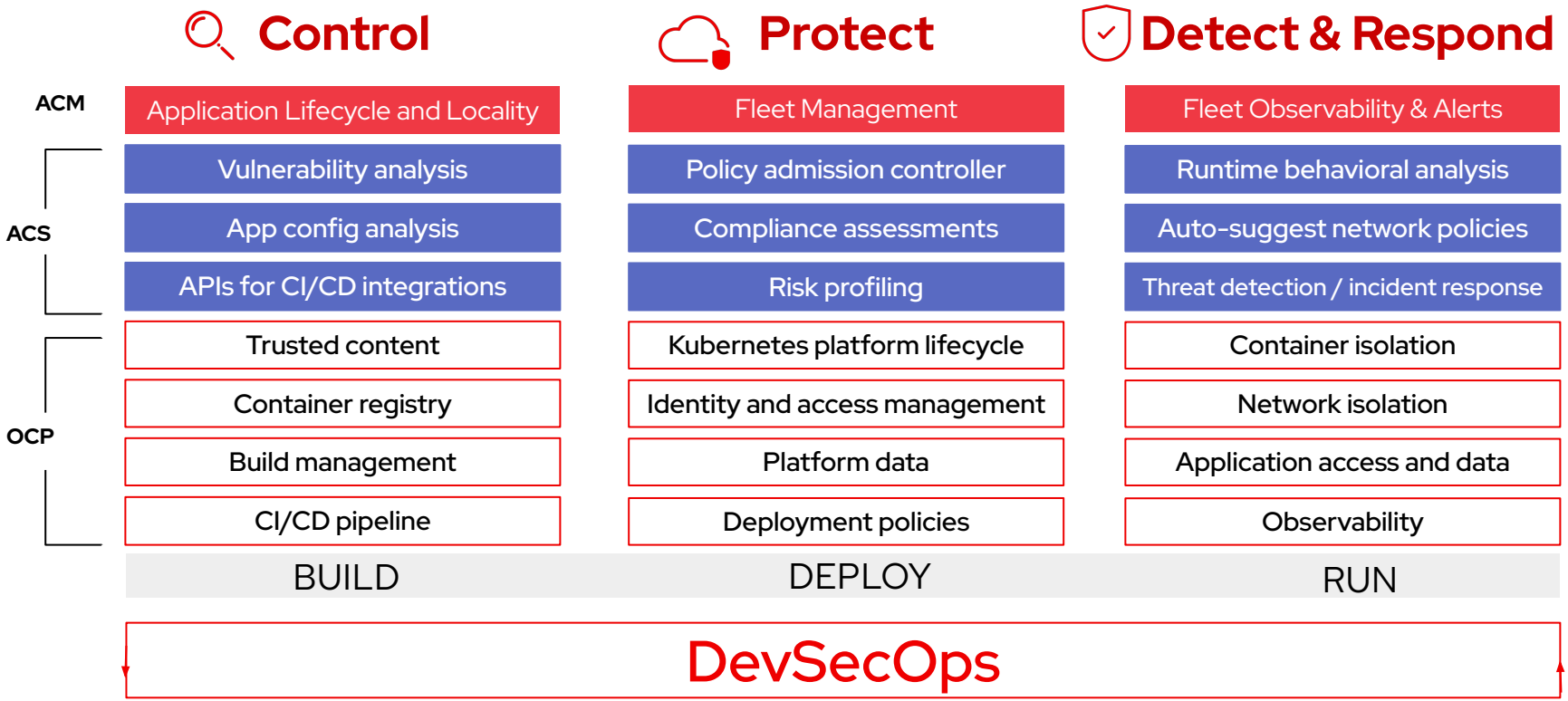
Application Management

Event Management

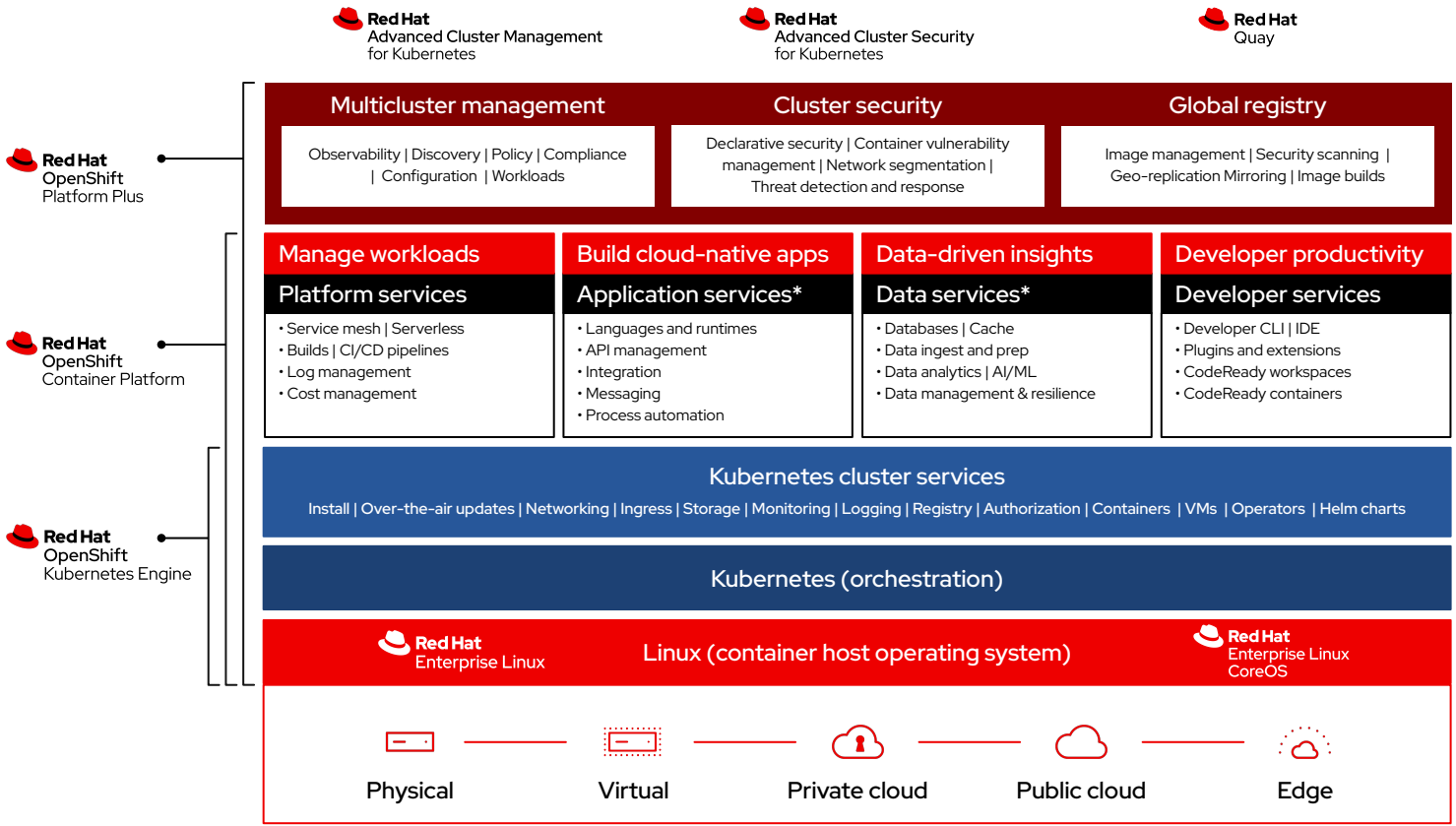
Existing Tools & Processes

Configuration & Compliance Management

OpenShift delivers continuous security



OpenShift Platform Plus

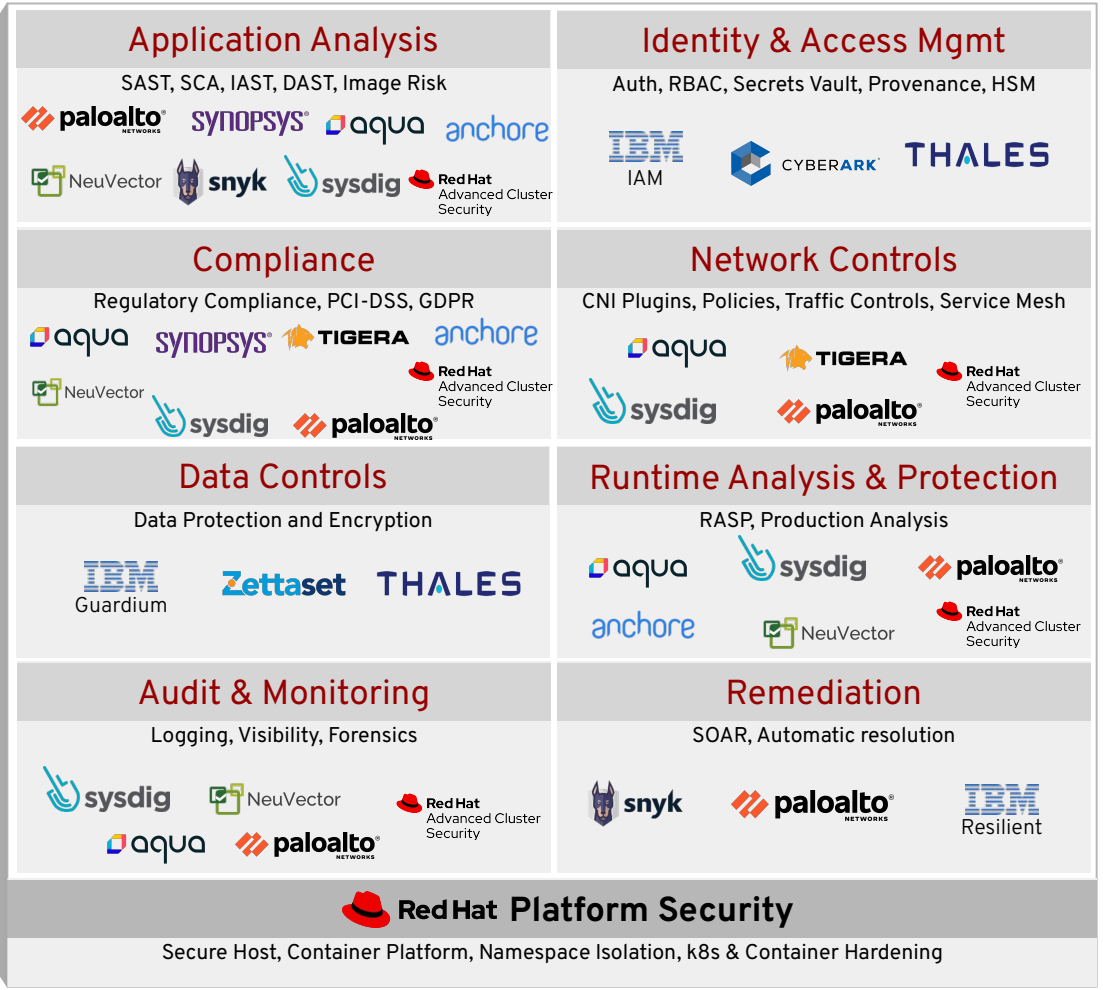


* Red Hat OpenShift® includes supported runtimes for popular languages/frameworks/databases. Additional capabilities listed are from the Red Hat Application and Data Services portfolio.

Red Hat

Security Partners by Use Case

Partners extend and enhance Red Hat functionality



Topics

Control the application supply chain

Protect the container application platform

Detect & Respond to issues discovered in running workloads

Red Hat Advanced Cluster Security (StackRox)

Securing the network: a deeper dive

Resource management

Cipher suites and encrypting etcd

The security ecosystem

The future of security

Control: shift security left

Trusted Content

Container Registry

Build Management

CI/CD Pipeline

Red Hat

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

DevNation Federal - The Journey to DevSecOps

Build: Control application security

Shift Security left

Best practices

Red Hat UBI

- Use trusted sources for external content such as base images

Quay

- Use a trusted private registry to manage supply chain risk

OCP Pipelines

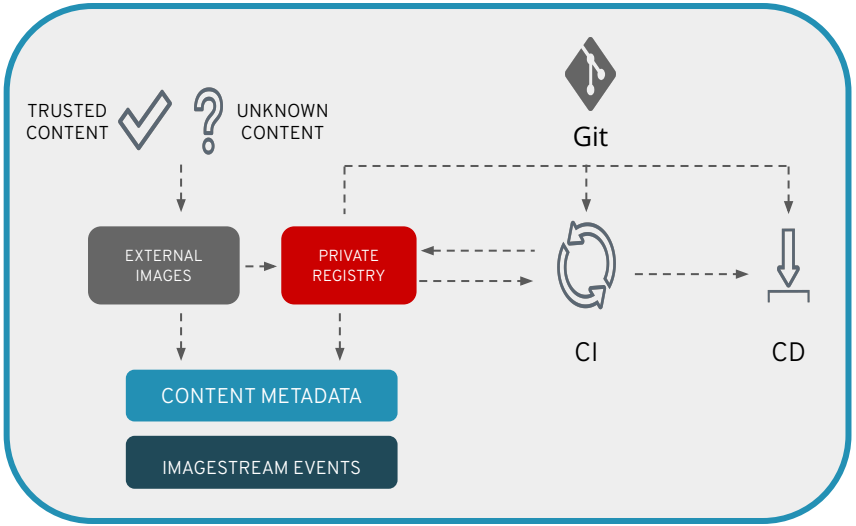
- Automate your CI/CD pipeline to enable rapid updates

Quay scanner (registry)
Code Ready (IDE)
ACS scanner (CI)
KubeLinter (CI)

- Integrate security tools / gates in your pipeline to identify
  - Known vulnerabilities
  - Application misconfigurations

ACM

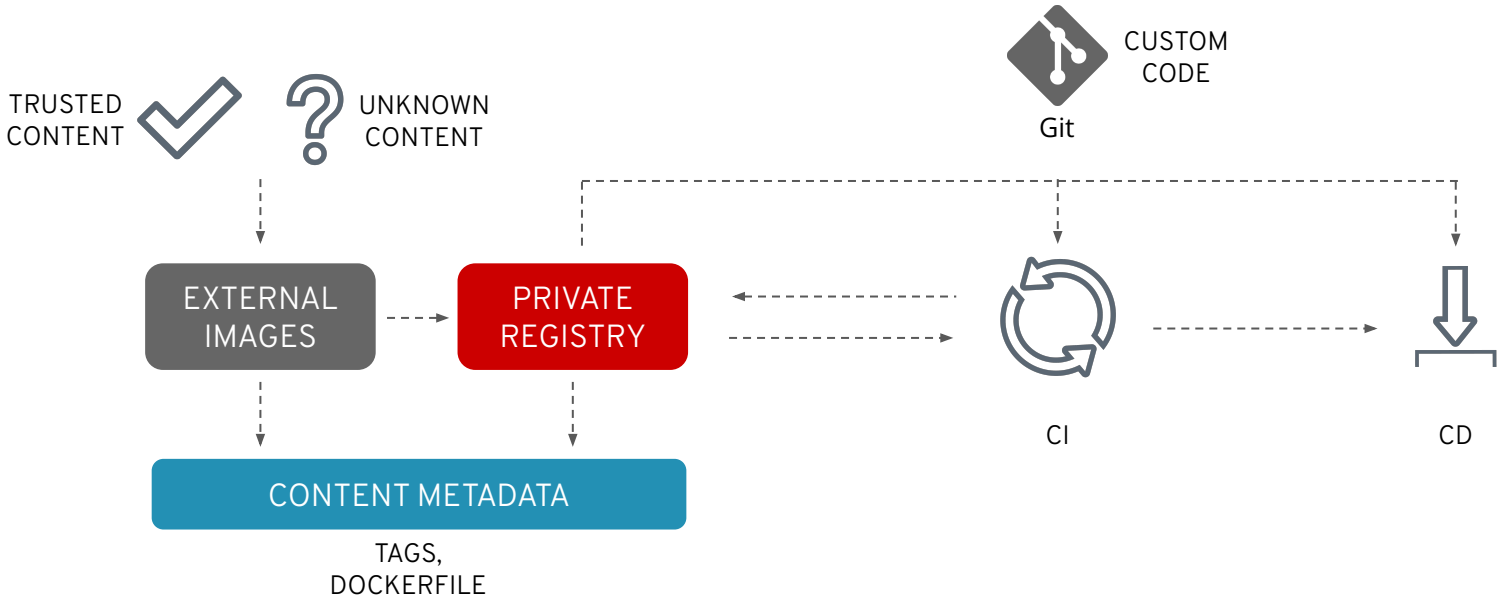
- Use policy-based deployment tools to manage application placement (e.g. locality)



Red Hat

Secure & Automate The Content Lifecycle

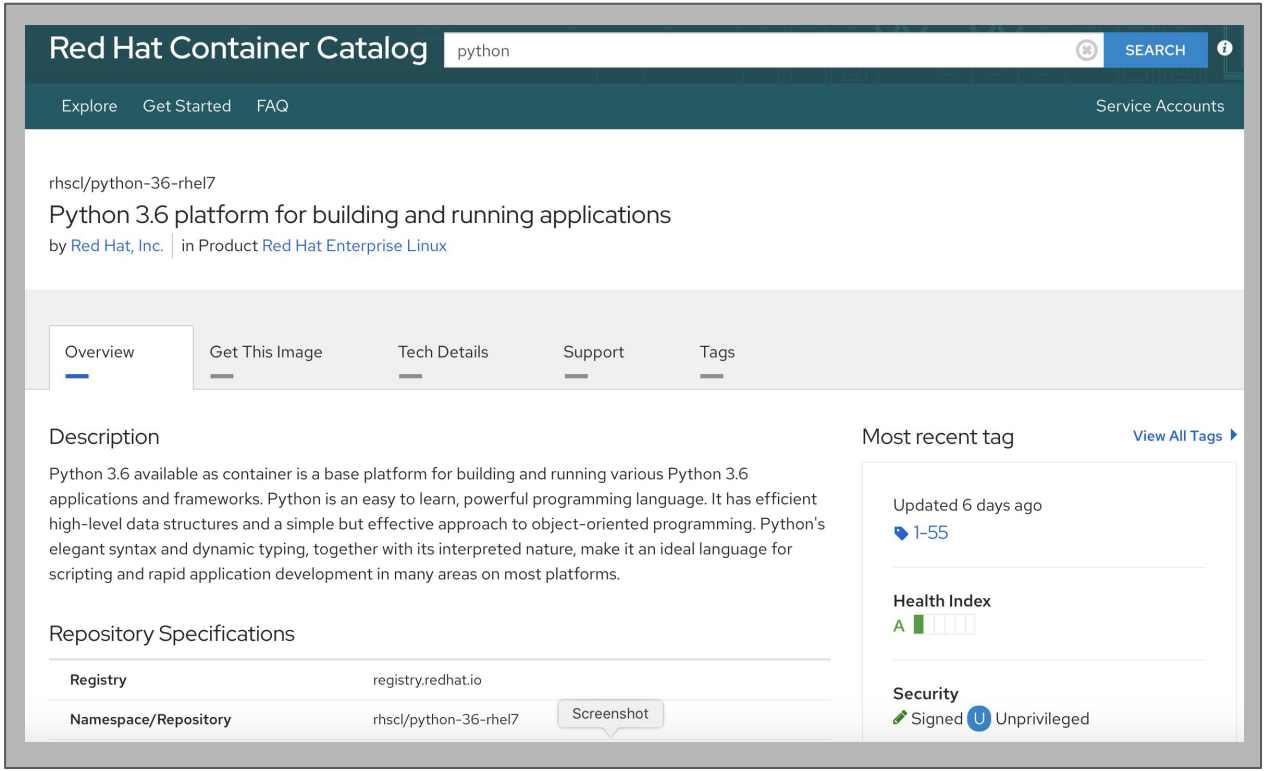
Elements of the container image supply chain



External Content: Use Trusted Sources

Red Hat Container Images

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

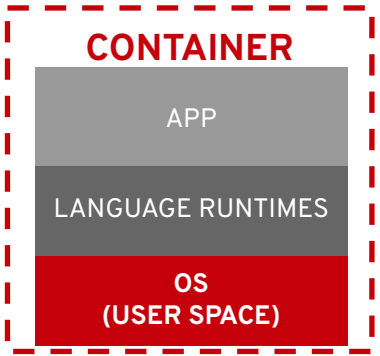


*More about the Health Index

Red Hat

The Red Hat Universal Base Image

The base image for all your needs -- enterprise architecture, security and performance



The Red Hat Universal Base Image is based on RHEL and made available at no charge by a new end user license agreement.

Development

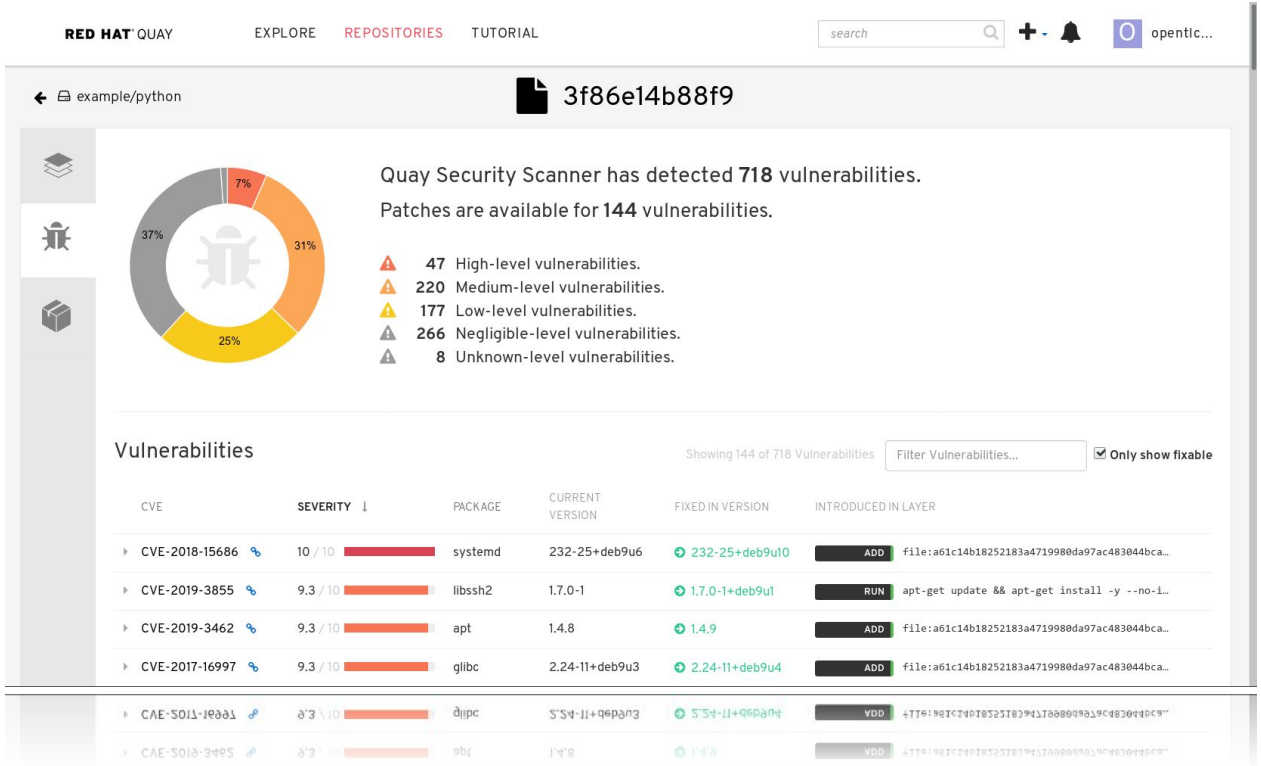
- Minimal footprint (~90 to ~200MB)
- Programming languages (Modularity & AppStreams)
- Enables a single CI/CD chain

Production

- Supported as RHEL when running on RHEL
- Same Performance, Security & Life cycle as RHEL
- Can attach RHEL support subscriptions as RHEL

Red Hat Quay
Enterprise Container Registry

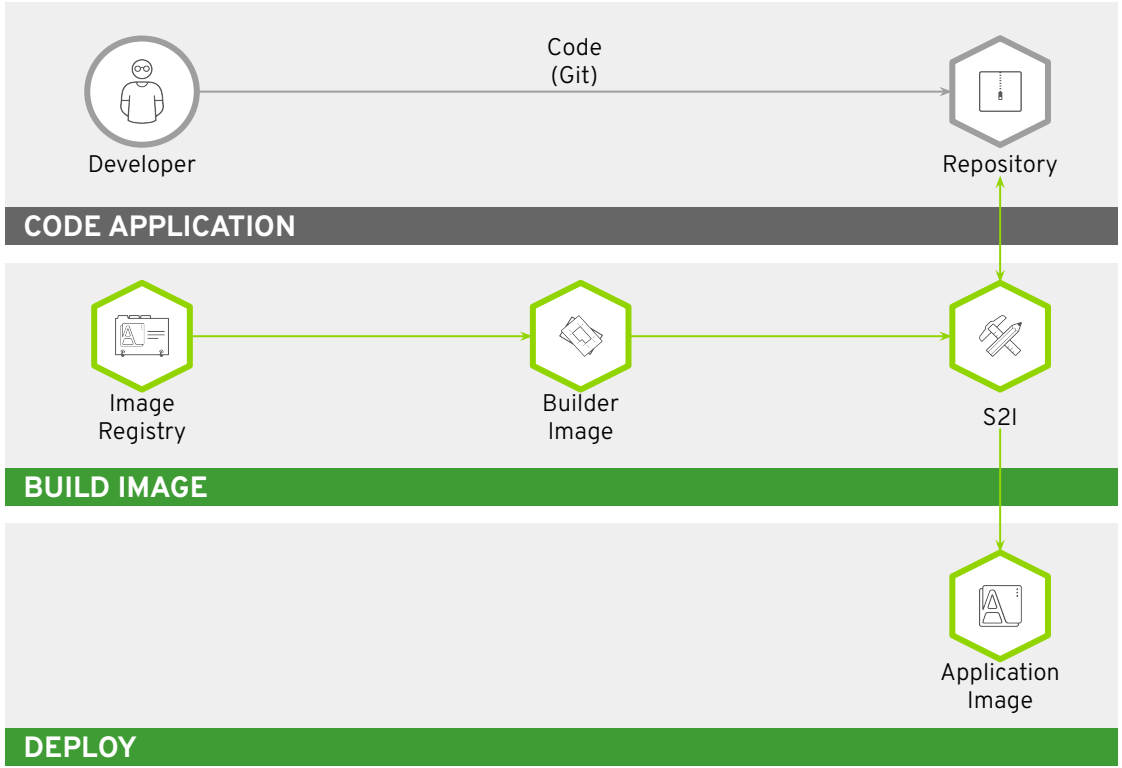
- Runs on and integrates with OpenShift
- Vulnerability Scanning (Clair)
- Geographic Replication
- Build Image Triggers
- Image Rollback with Time Machine



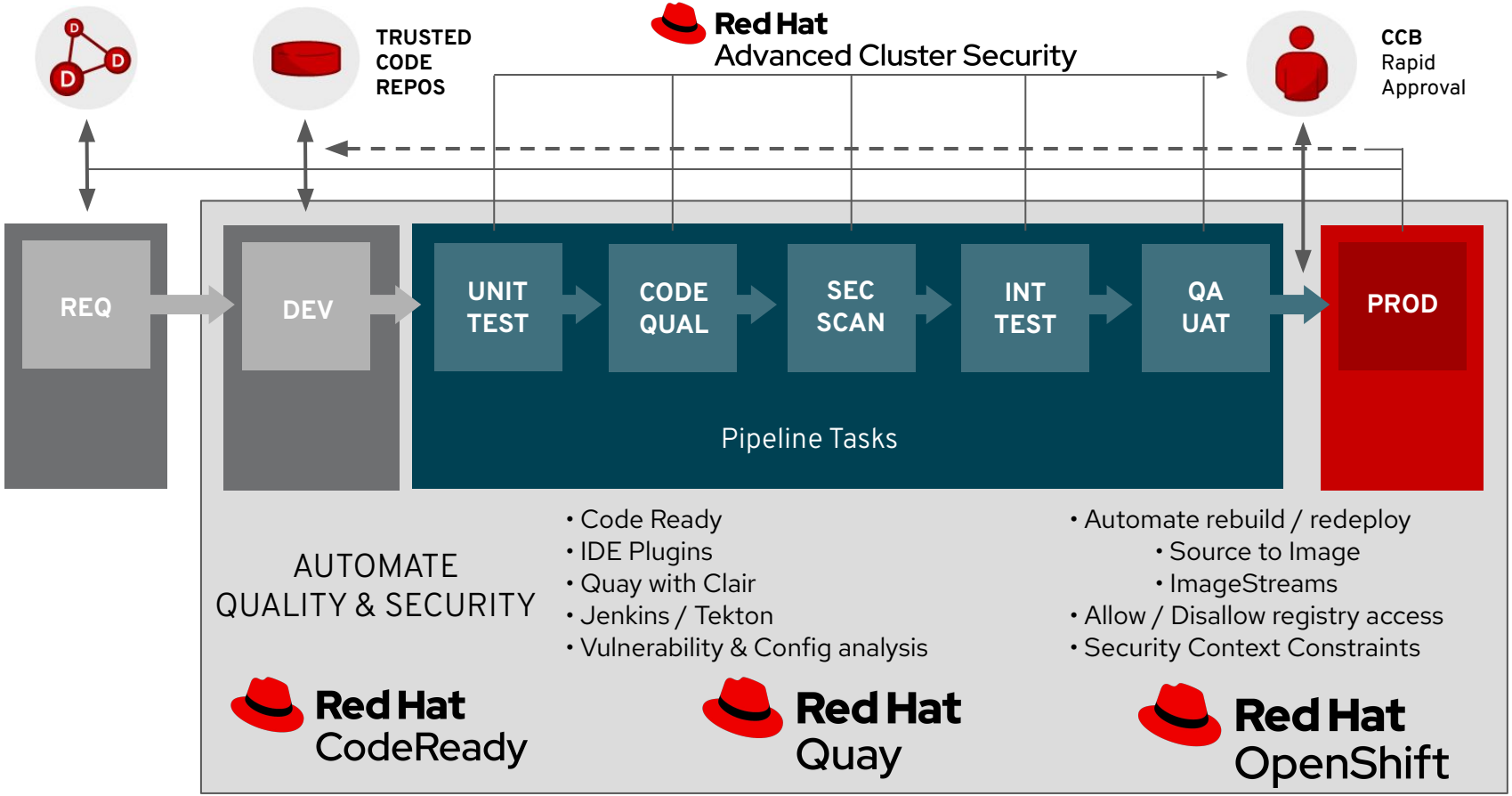
Red Hat

OpenShift Builds

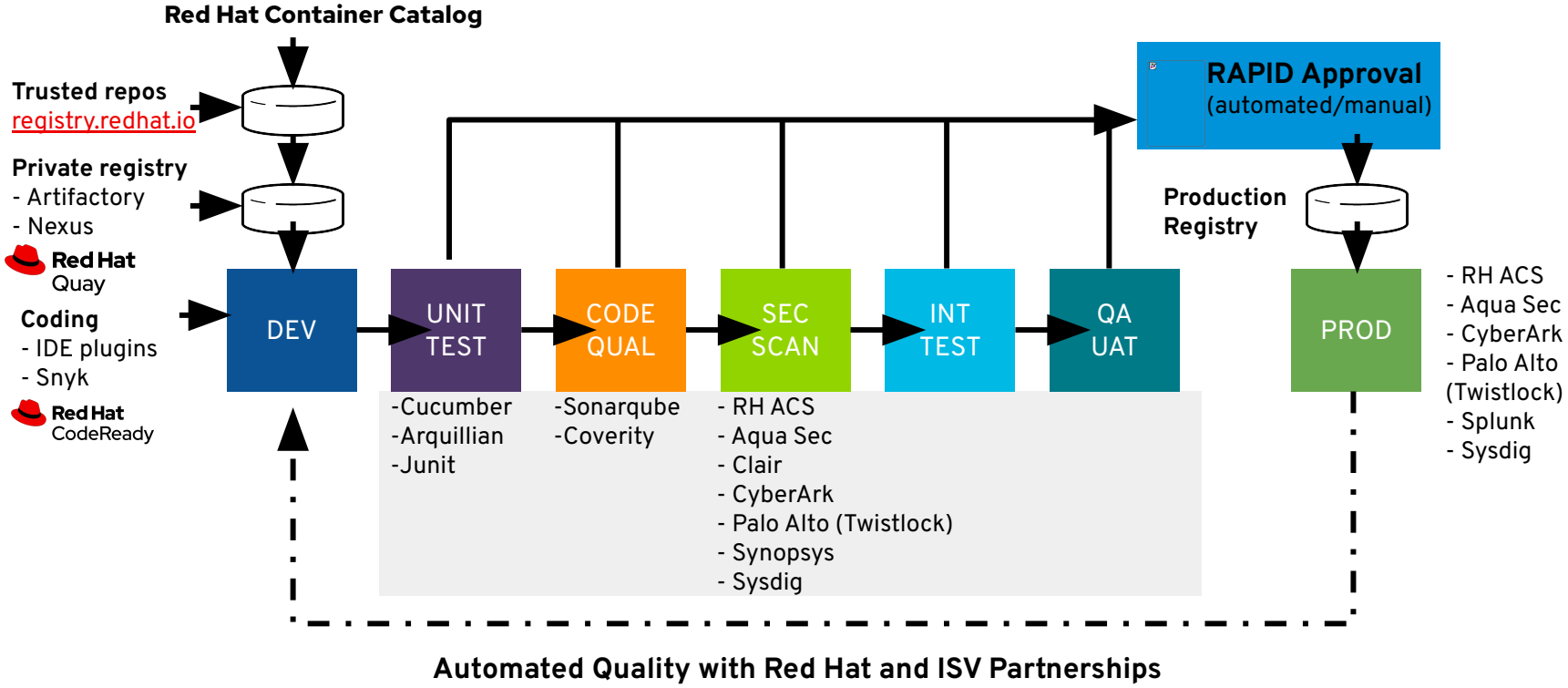
- Build lean images from application source code and binary using Kubernetes tools on OpenShift
- Use Kubernetes build tools (e.g. buildah, S2I, CNB1, Kaniko, etc)
- Build slim runtime images without the build dependencies
- Extensible and customizable
- Portable builds to any Kubernetes platform



The CI/CD pipeline for containers needs automation



Securing your CI/CD Pipeline with Red Hat



Red Hat Container Catalog: Java Applications
More about the Container Health Index

Red Hat

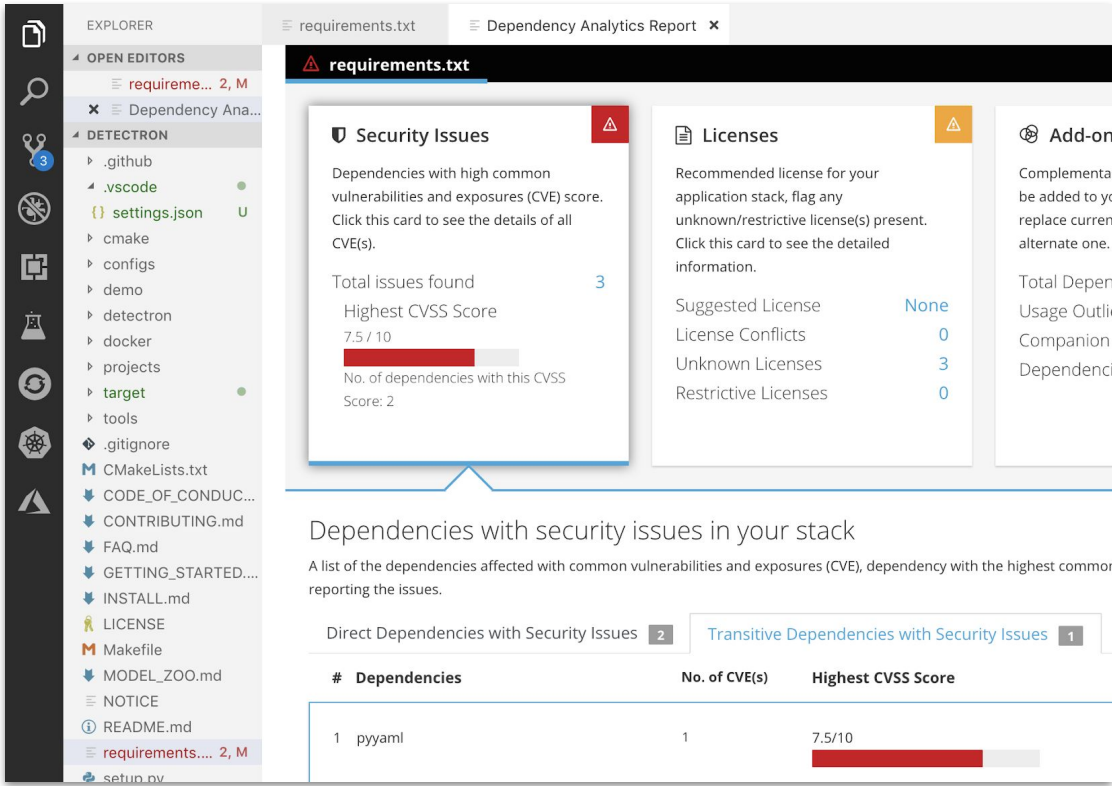
Plugins available for VSCode and Eclipse IDEs

Enhancing Secure Application Development and DevSecOps

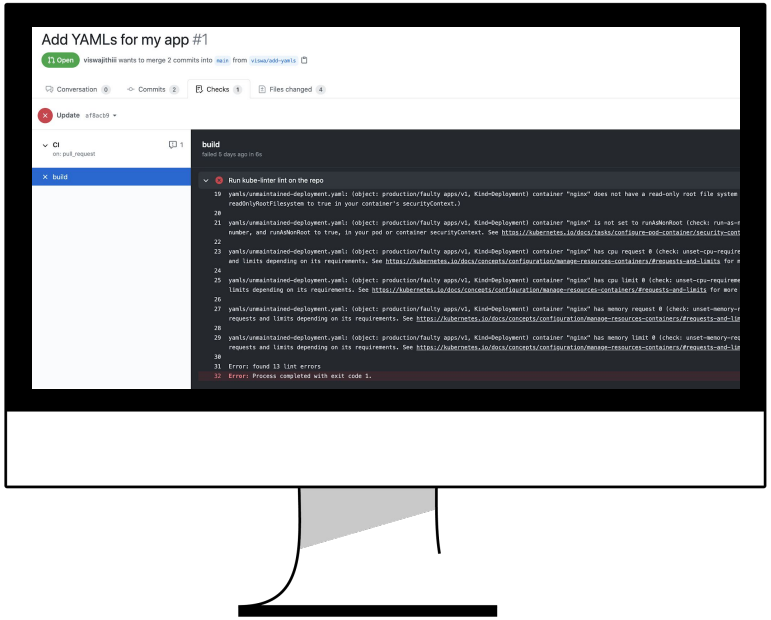
"Shift Left" - find CVEs and license issues during development

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



KubeLinter: Enforce Kubernetes security best practices



KubeLinter as a GitHub action

- Checks Kubernetes YAML files and Helm charts
- 16 default checks
- Extensible with custom checks
- Integrates with any CI tool

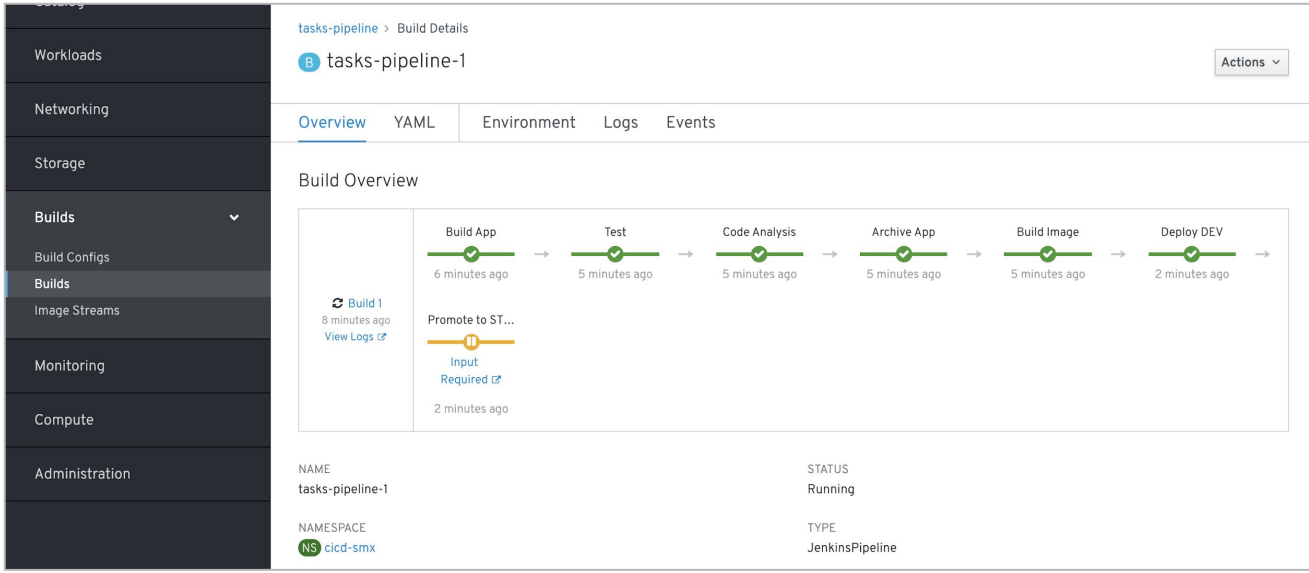
https://github.com/stackrox/kube-linter

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.

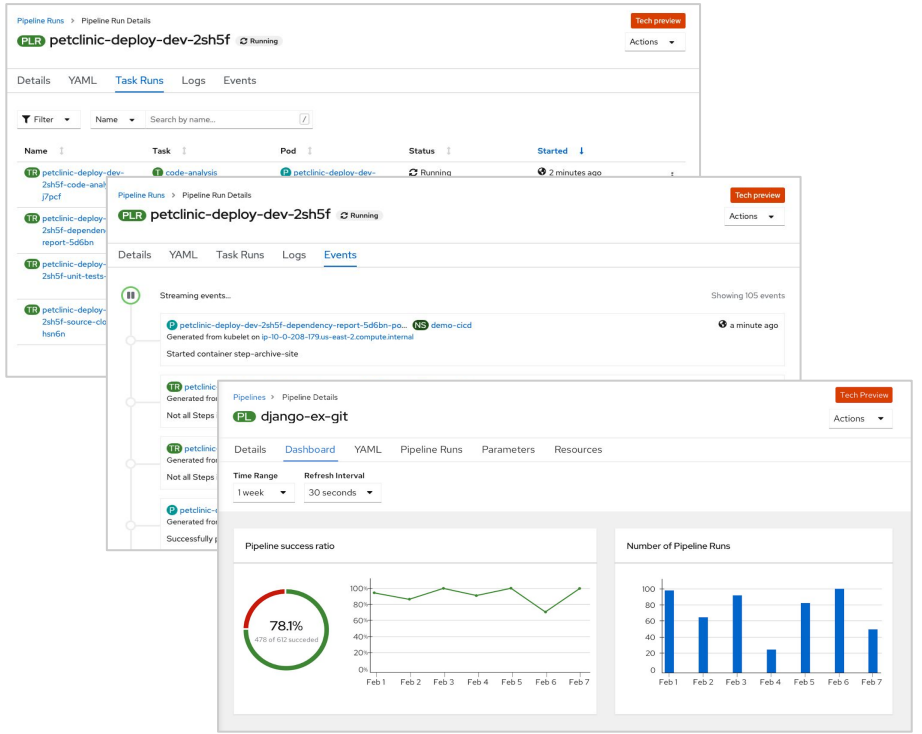
The full Jenkins UI is also available.



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

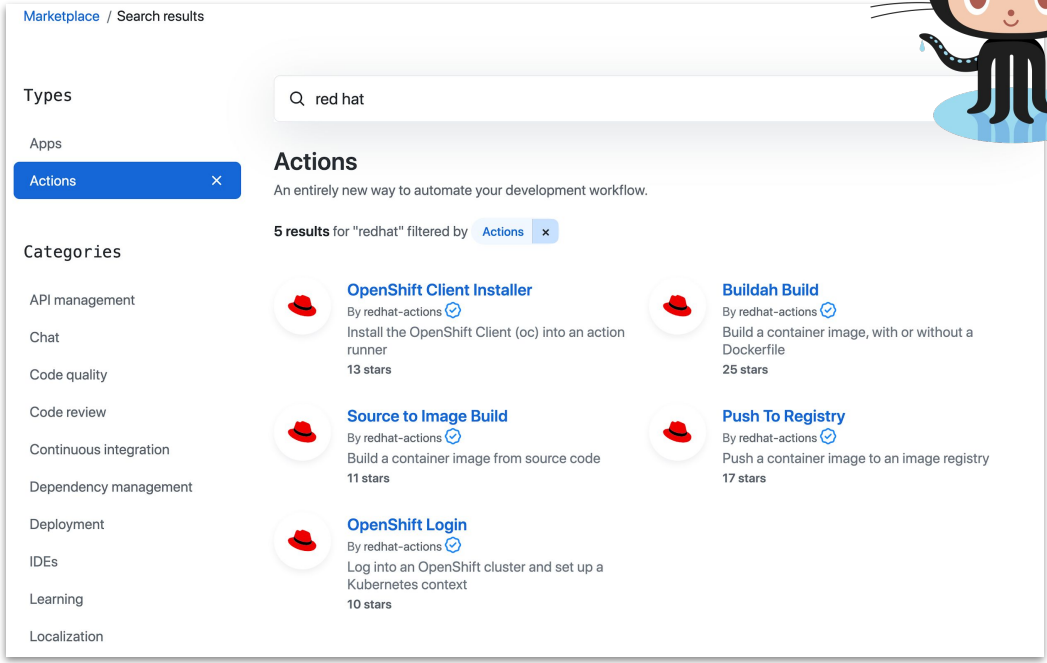
OpenShift Pipelines

- 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.
- Reduced pipeline privileges (controllers as nonroot, pipelines as anyuid
- Cluster-wide proxy configs passed to TaskRuns pods
- HTTPS support for webhooks (TLS in EventListeners)
- EventListener can be shared across multiple namespaces to reduce resource consumption
- Image digest published as result in buildah and S2I tasks



Red Hat GitHub Actions

- GitHub partnerships (press release)
- Interact with OpenShift from GitHub workflows
- Verified OpenShift actions on GitHub Marketplace
  - OpenShift client (oc)
  - OpenShift login
  - S2I build
  - Buildah builds
  - Push image to registry
- More actions and GitHub Runner to come...



Blog: Deploying to OpenShift using GitHub Actions | Demo

Red Hat

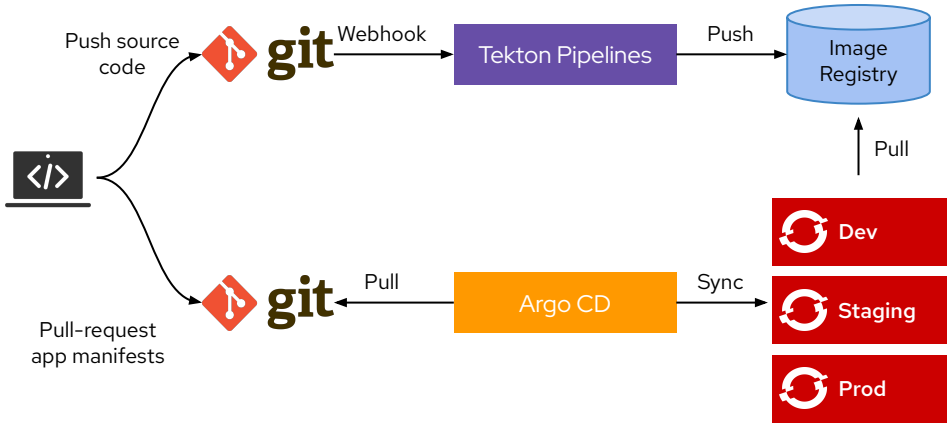
ROADMAP

GitOps Application Manager

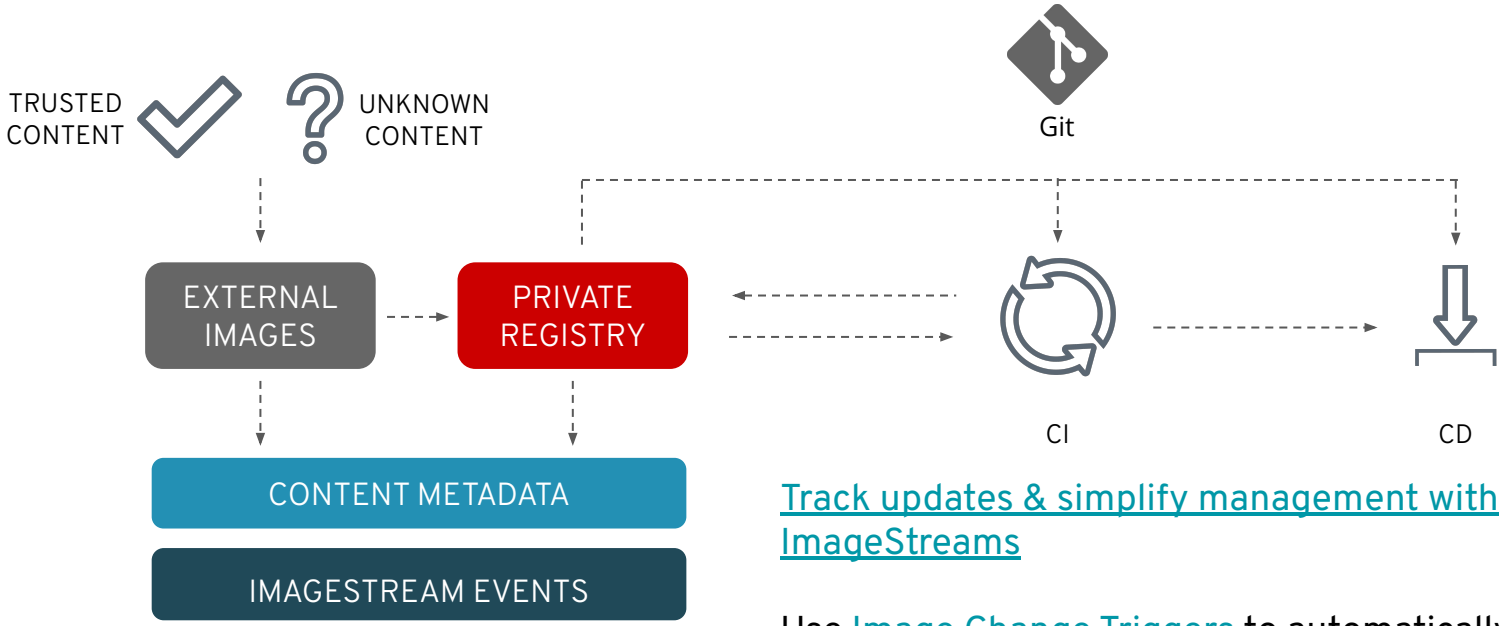
An opinionated continuous delivery process with GitOps principles

```
$ kam bootstrap
```

- Automated, integrated and opinionated
- Bootstraps Git repos and CI/CD
- Deployment environments for apps
- Generated Tekton Pipelines for CI
- Argo CD for multi-cluster CD
- Secret management integration
- Kustomize and Helm



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

Red Hat

Protect: secure the platform

Platform Lifecycle

Identity and Access Management

Platform Data

Deployment Policies

Red Hat

Deploy: Protect the application platform

Best practices

RHEL CoreOS
- Reduce attack surface with a container optimized operating system

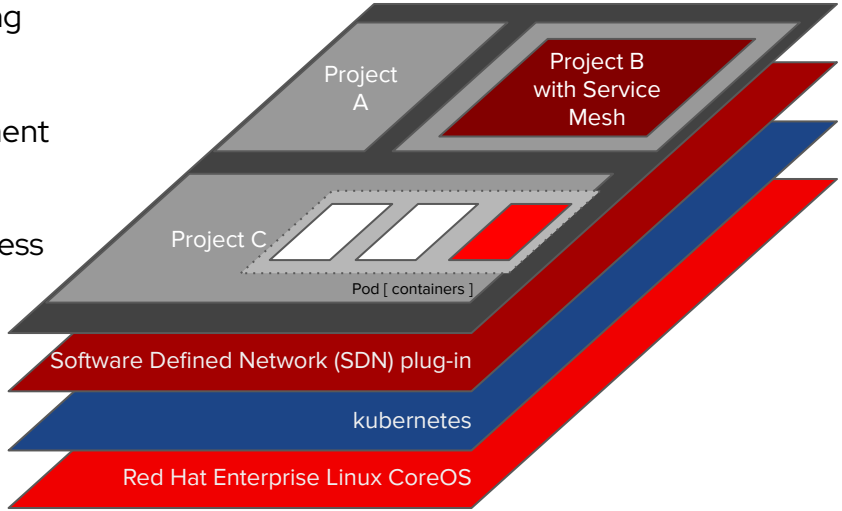
OCP Operators
ACM
- Use automated and policy-driven configuration management across your fleet

OCP RBAC
ACS to monitor
ACM to enforce
- Implement least privilege with fine-grained role based access control (RBAC)

OCP CAs
Service mesh
OCP IPSec
RHCOS NBDE
Encrypt etcd
- Encrypt platform data in transit and at rest

OCP Compliance Operator
ACS
ACM
- Use automated compliance, risk assessment and remediation solutions

OCP Security Context Constraints
ACS
- Reduce deployment risk with admission control policies that
  - Minimize admission of privileged pods, pods with host capabilities
  - Prevent admission of pods with critical vulnerabilities



Red Hat

OpenShift 4 Configuration, Lifecycle management and Security

Dramatically simplified for The Hybrid Cloud

Machines

Machines are complex for ops

Make machines easy
(like containers)

Configuration

Config change is risky

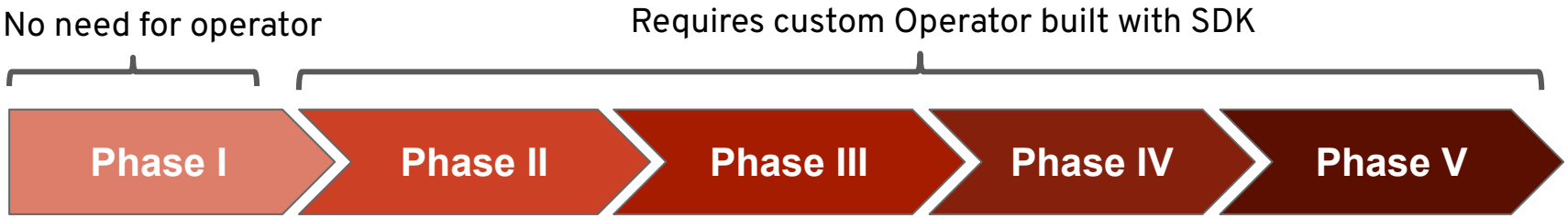
Make config management
and config change
easy and safe

Lifecycle

Software lifecycle is hard

Automate software
lifecycle on Kube

The Value Of Kubernetes Operators



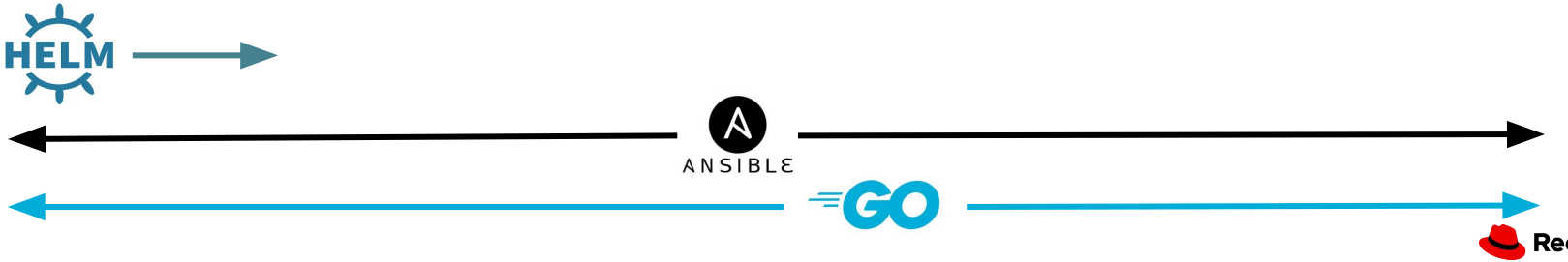
Installation
Automated application provisioning and configuration management

Upgrades
Patch and minor version upgrades supported

Lifecycle
App lifecycle, storage lifecycle (backup, failure recovery)

Deep Insights
Metrics, alerts, log processing and workload analysis

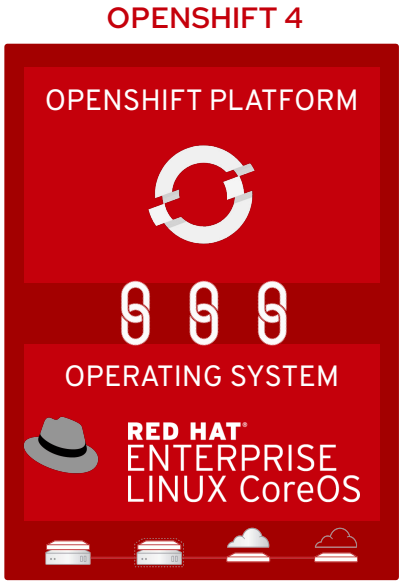
Auto-pilot
Horizontal/vertical scaling, auto config tuning, abnormal detection, scheduling tuning...



Red Hat

Red Hat Enterprise Linux CoreOS

Controlled Immutability / Container Optimized



Role in OpenShift Ecosystem

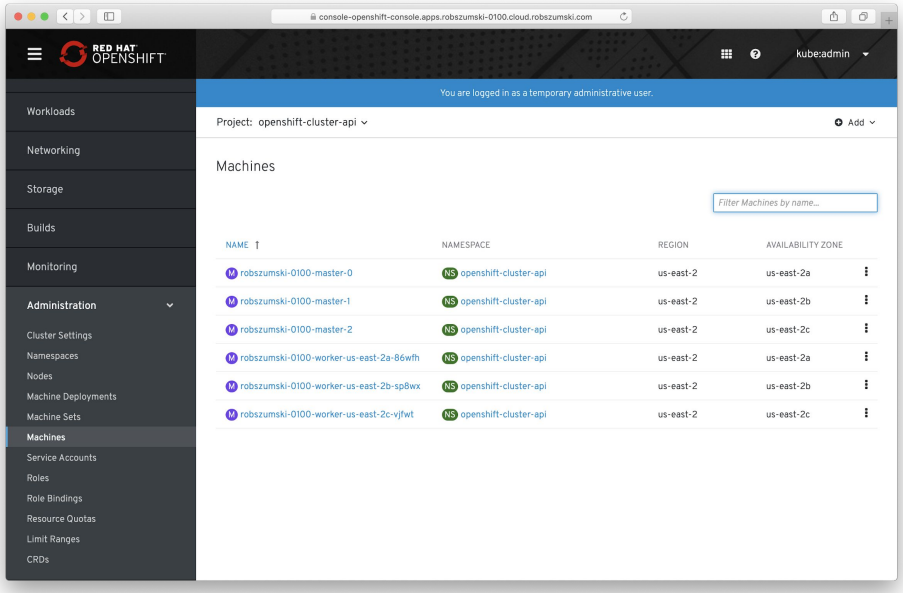
- Versioned and validated for specific OpenShift version
- User space read-only

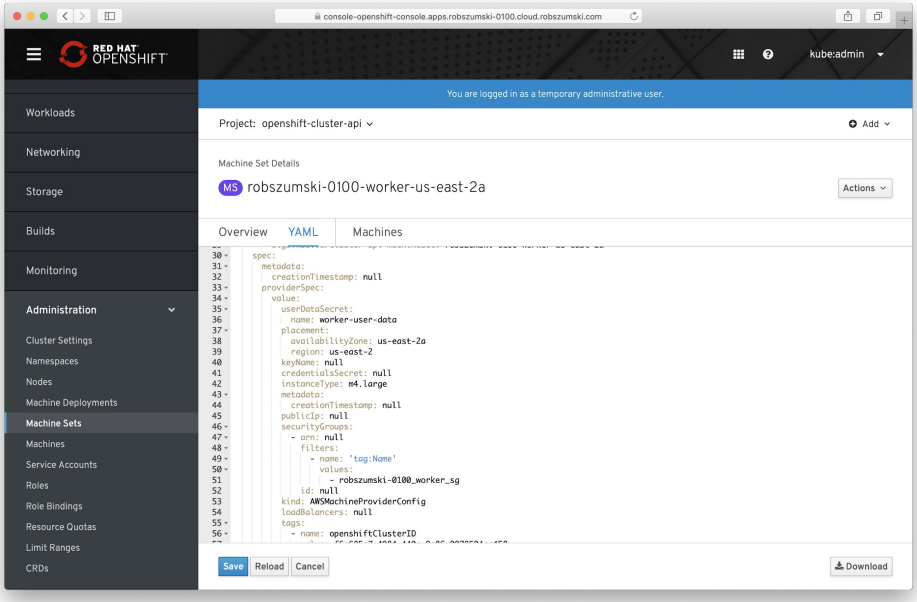
Managed by the OpenShift Cluster

- Considered a member of an OpenShift Deployment
- Configuration managed by the Machine Config Operator
  - Container runtime
  - Kubelet configuration
  - Authorized container registries
  - SSH Configuration
  - Multiple machine pools can be created
- Continuously monitoring for configuration drift
- Deploy the File Integrity operator to monitor for changes to files

Kubernetes Machine Api Operator

Using Kubernetes To Provision And Scale Clusters

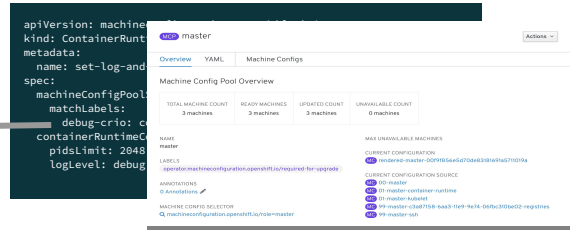




Openshift Machine Config Operator: Monitor for Configuration Drift

1 A user requests a new cluster

Red Hat OpenShift Container Platform



2 Red Hat curates MachineConfigs to meet security best practices

Red Hat OpenShift Container Platform

Red Hat Enterprise Linux CoreOS

Describe intent with declarative config

=

Observe

Maintain

4 Metrics are sent to Red Hat Insights for analysis via secured HTTPS.

3 The Machine Config Operator delivers the secure machine config you need

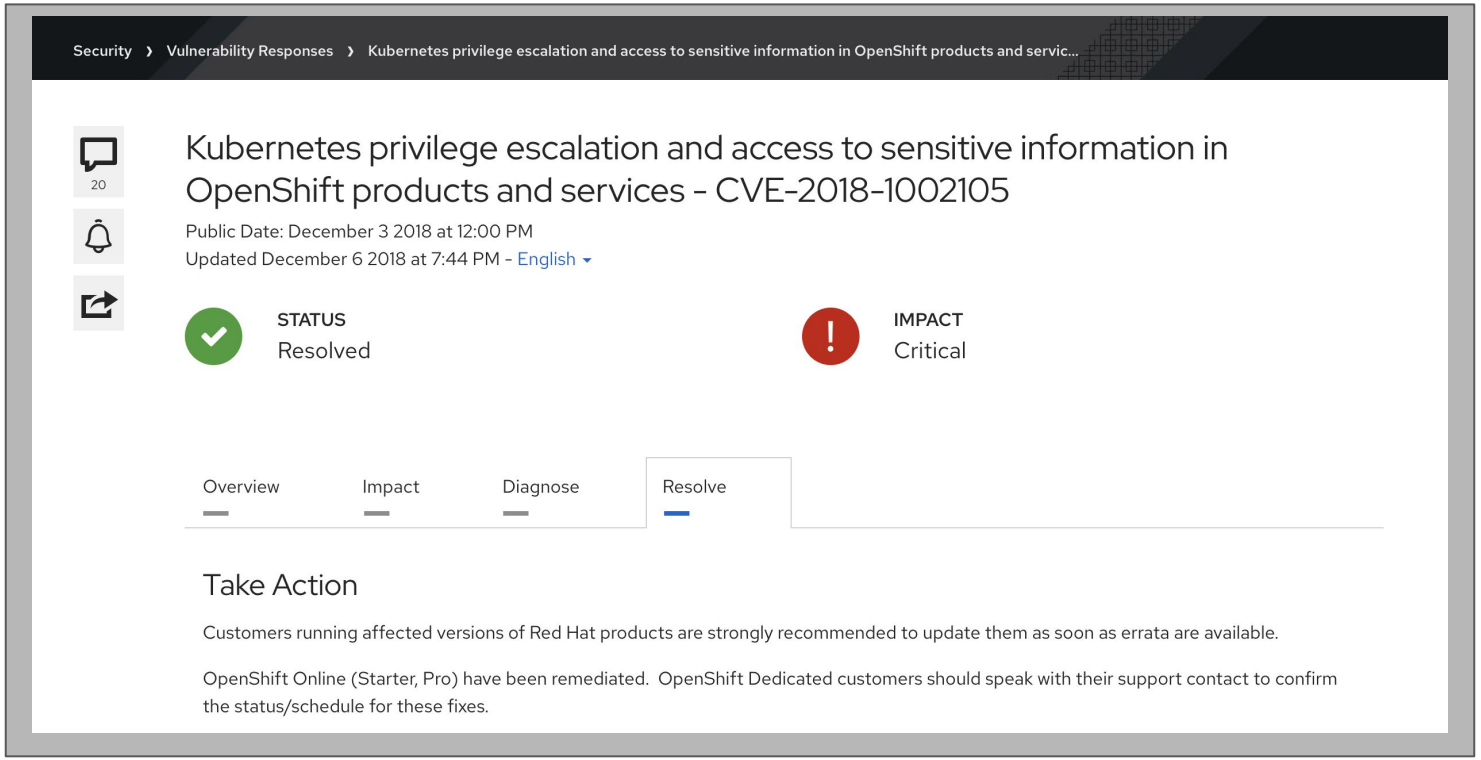
Monitor, scale, troubleshoot, backup

Install, upgrade, reconcile, config

Red Hat OpenShift Container Platform

Red Hat

Integrated stack enables faster fixes



https://access.redhat.com/security/vulnerabilities/3716411

Fix backported all the way back to OpenShift 3.2 (Kubernetes 1.2)

Red Hat

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

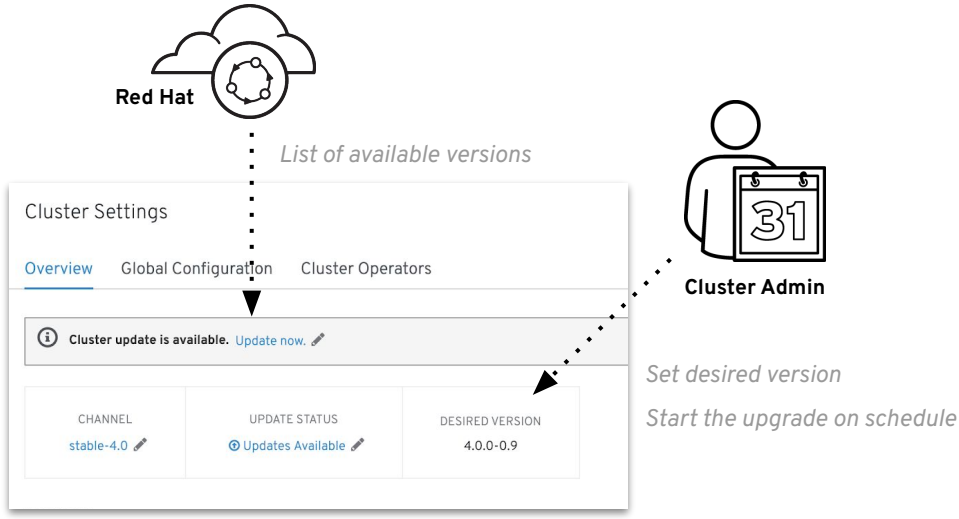
No need for separate windows for each component

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



[NEW: OpenShift Update Service for restricted networks]

Red Hat

Day 2 Configuration

Global Configuration

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

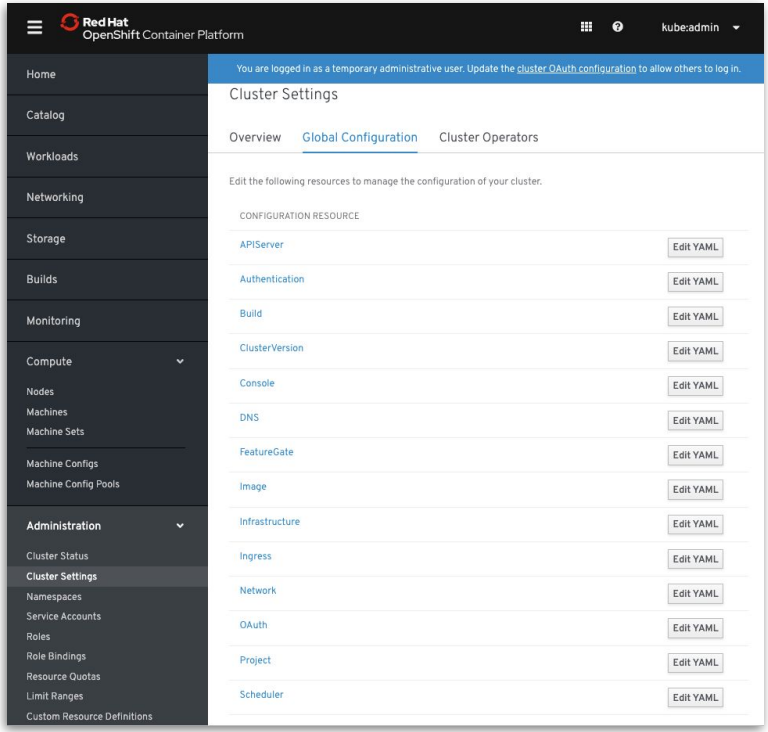
Change via Cluster Settings screen

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

Operators apply these updates

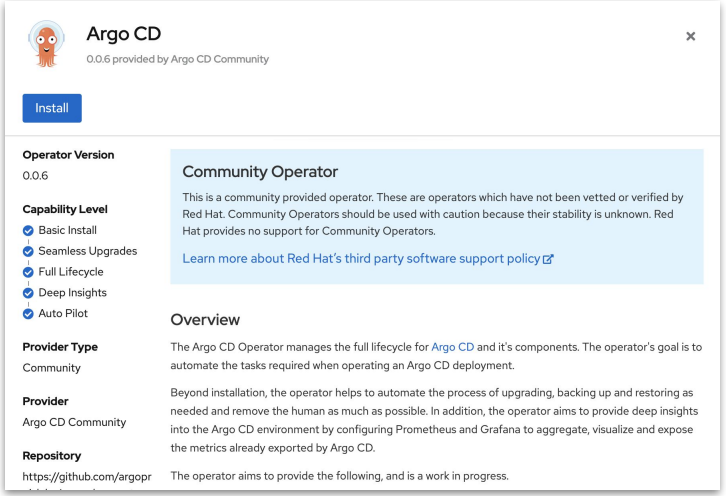
One or more Operators are responsible for propagating these settings through the infrastructure

- Identity Provider
- Ingress Controller
- Logging, Metrics



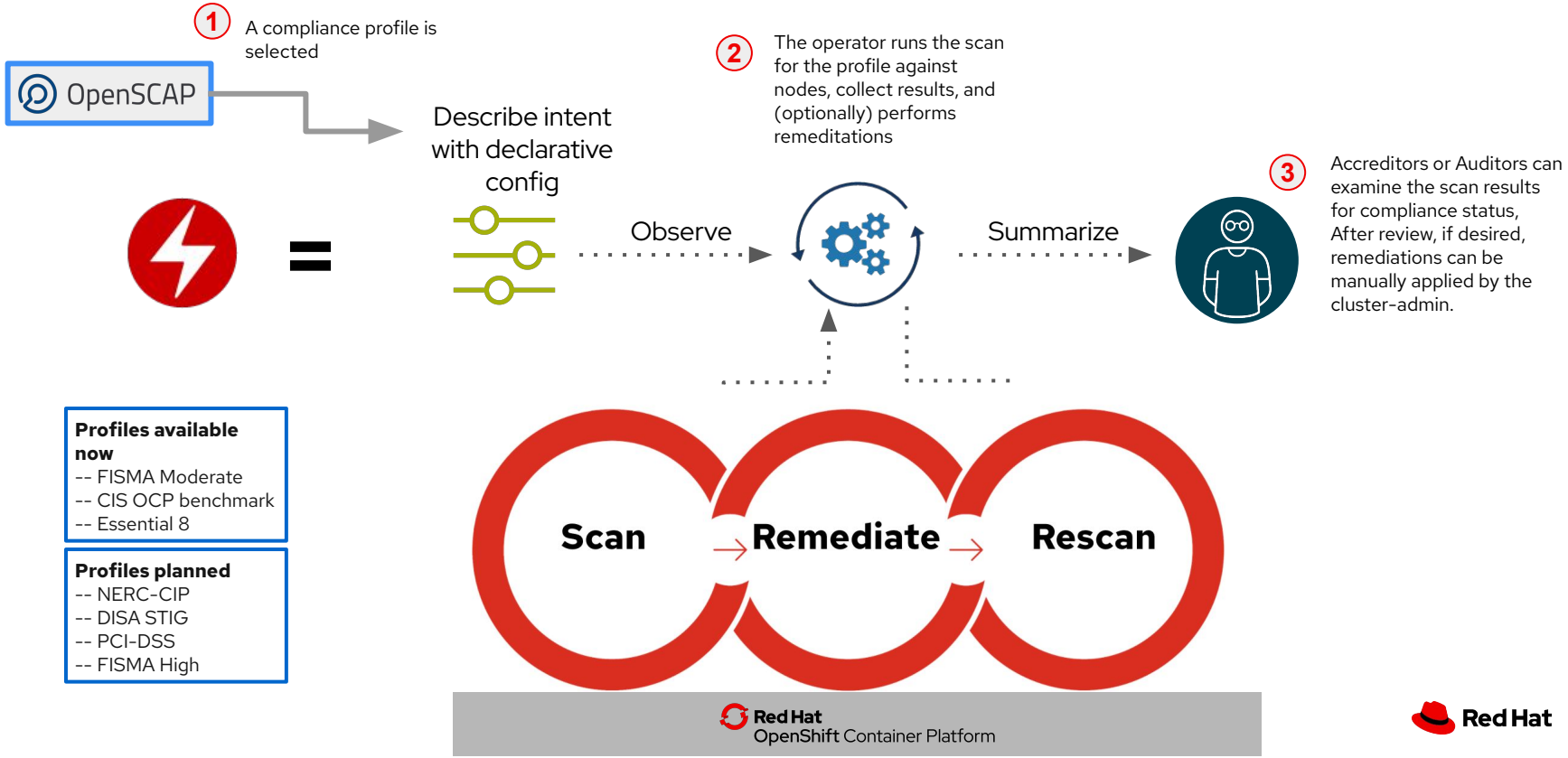
GitOps with ArgoCD Guide

- Guide published to GitHub
  github.com/openshift/openshift-gitops-examples
- Topics
  - Install and configuration of ArgoCD
  - Cluster configs with ArgoCD
  - Operator installation
  - Multi-cluster configs



Red Hat

Openshift Compliance Operator for Continuous Compliance



Red Hat

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)

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

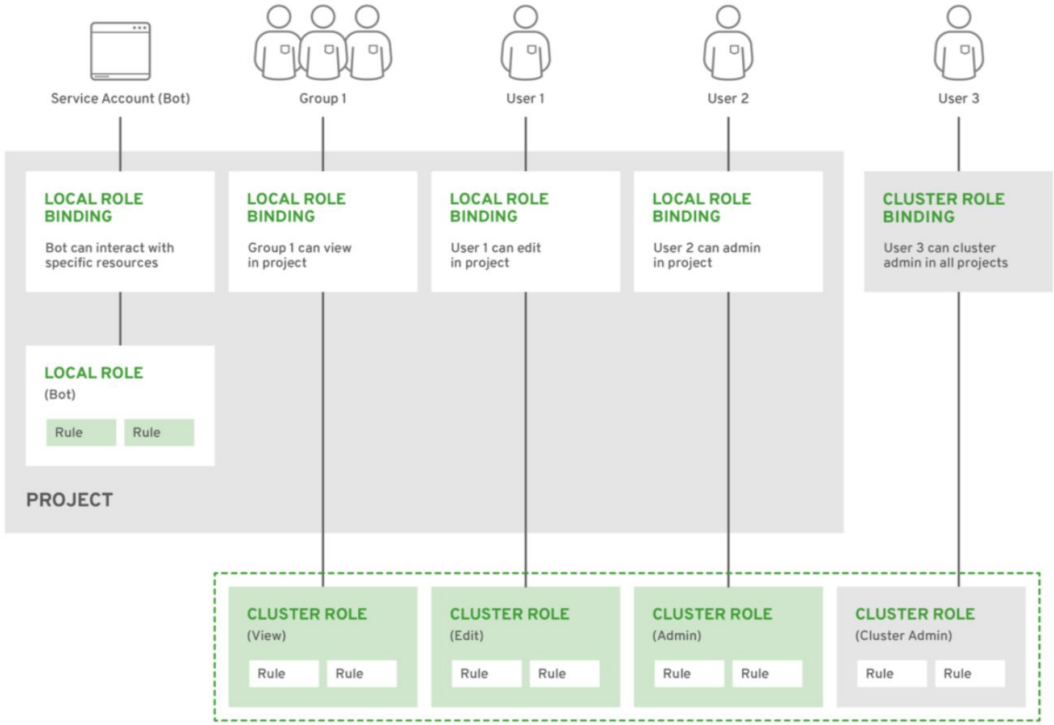
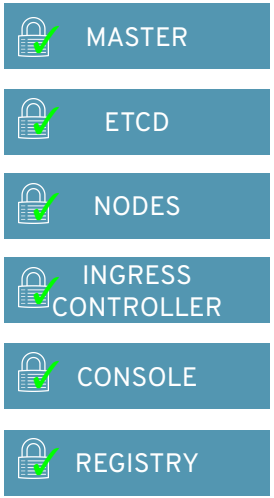


Figure 12 - Authorization Relationships

For more information see: [Managing RBAC in OpenShift]

Protecting data in motion

- All traffic between control plane and worker nodes is encrypted
- 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



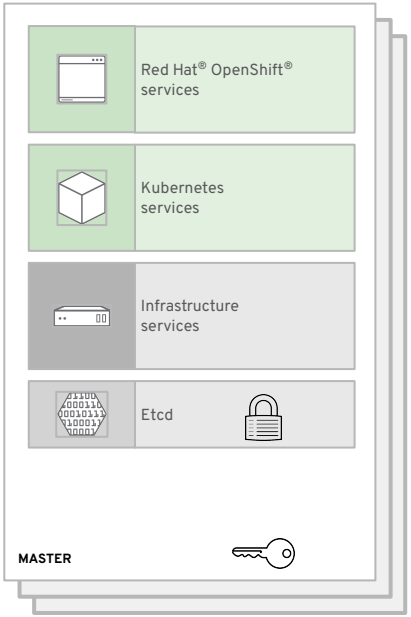
Protecting data at rest

Encrypt secrets, config maps by encrypting etcd

- Encryption of the etcd datastore is optional. Once enabled, encryption cannot be disabled.
- The aes-cbc cipher is used.
- Backup: The etcd data store should be backed up separately from the file system with the key.
- Disaster recovery: a backup of both the encrypted etcd data and encryption keys must be available.

Encrypt volumes with 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



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

Red Hat OpenShift 4 and FIPS 140-2

FIPS-ready services

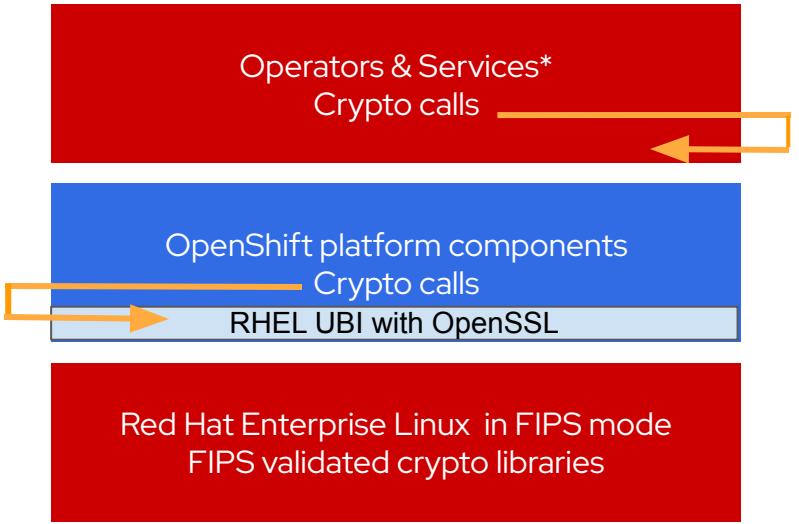
- When built with Red Hat Enterprise Linux 7 base image, the operating system is FIPS ready.

Red Hat OpenShift calls FIPS validated crypto

- When running on RHEL in FIPS mode, OpenShift components bypass go cryptographic routines and call into a RHEL FIPS 140-2 validated cryptographic library.
- This feature is specific to binaries built with the RHEL go compiler and running on RHEL.

Red Hat Enterprise Linux CoreOS FIPS mode

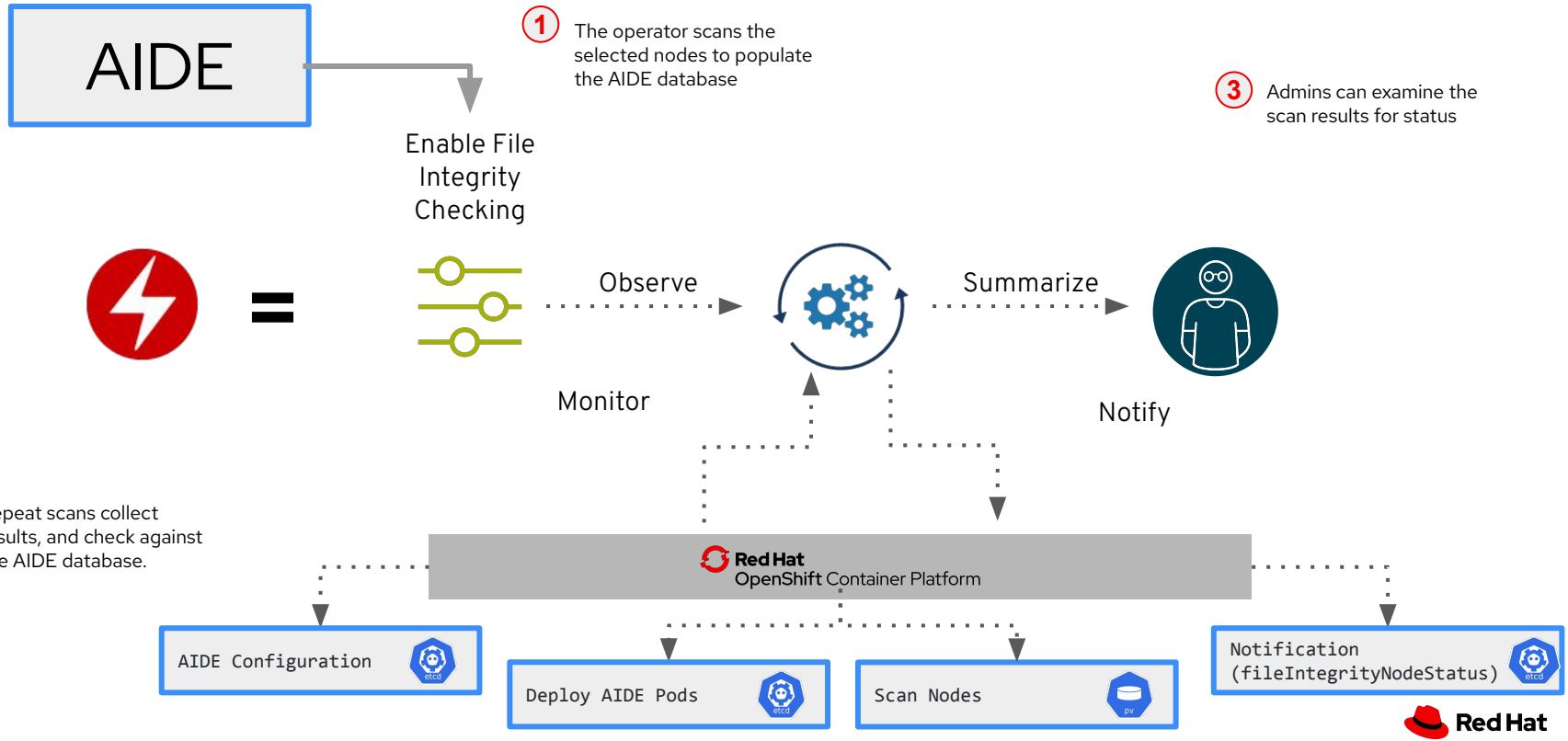
- Configure at install to enforce use of FIPS Implementation Under Test* modules.



*When built with Red Hat Enterprise Linux base images

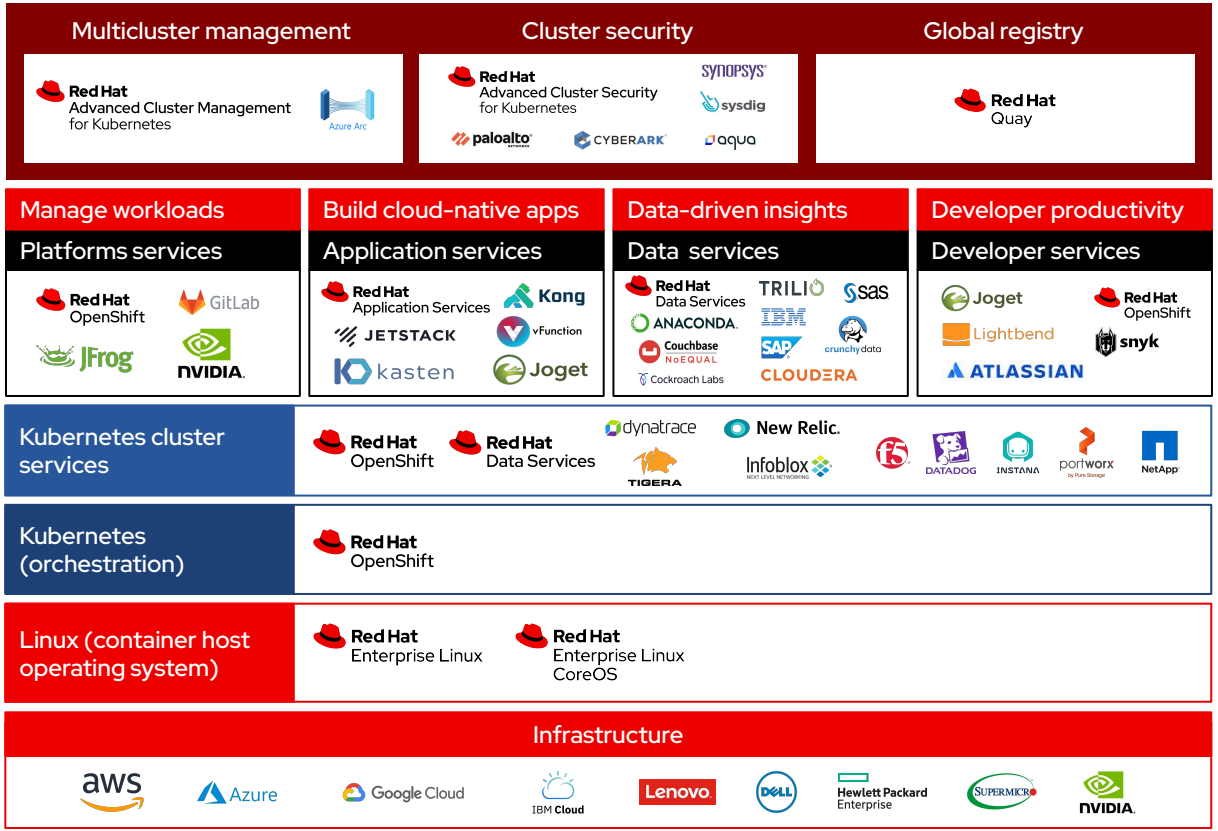
More about RHEL go and FIPS 140-2

Openshift File Integrity Operator



Red Hat open hybrid cloud platform with ISV ecosystem

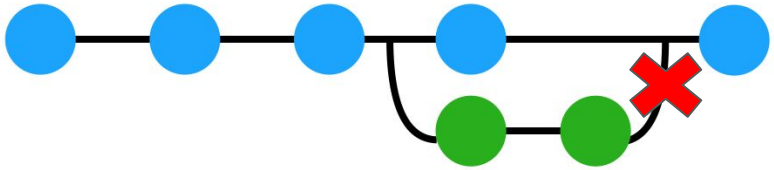
Simplify the trial, procurement, and deployment of ISV software on Red Hat OpenShift, anywhere with Red Hat Marketplace



Red Hat

Policy-based deployment

- Allow list / block list to ensure pods are only deployed from approved registries
- Validate image signatures
- Automate principle of least privilege with Security Context Constraints
  - Automate allowed permissions for pods; if requested permissions are not allowed, the pod is not deployed
  - With the restricted SCC, pods cannot run as privileged, mount host directory volumes, or access the host network.
  - Admin can grant access to privileges when necessary.



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>
  Groups:                               system:authenticated
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.

(Security Context Constraints)

Detect & Respond: secure running applications and services

Container Isolation

Network Isolation

Application Access and Data

Observability

Red Hat

Run: Securing the container runtime

Best practices

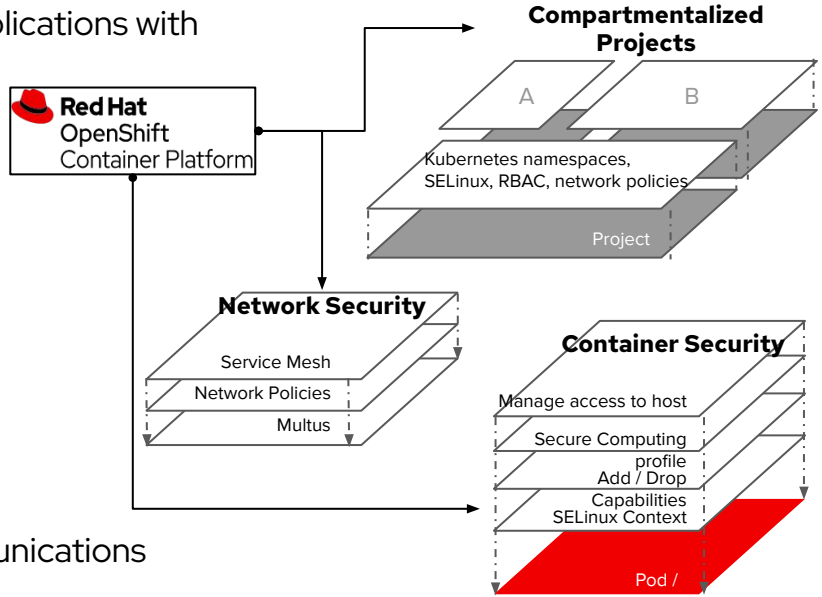
OCP ACS
- Minimize the impact of an attack by isolating running applications with
  - SELinux & Security Context Constraints
  - Kubernetes namespaces (Projects), RBAC
  - Network Policies for microsegmentation

OCP ACM
- Use resource quotas to prevent resource exhaustion

OCP
- Manage application access and protect application data
  - Red Hat Single Sign On for user management
  - Secure routes / ingress, 3Scale API Gateway
  - Service mesh to encrypt pod-to-pod traffic
  - Egress IPs / firewall

OCP ACS ACM
- Monitor application metrics, logging and network communications

ACS
- Automate threat detection and response
  - Alert or kill pods based on anomalous behavior
  - Detect privilege escalation and risky processes such as cryptomining



Red Hat

cri-o

A lightweight, OCI-compliant container runtime

Optimized for Kubernetes

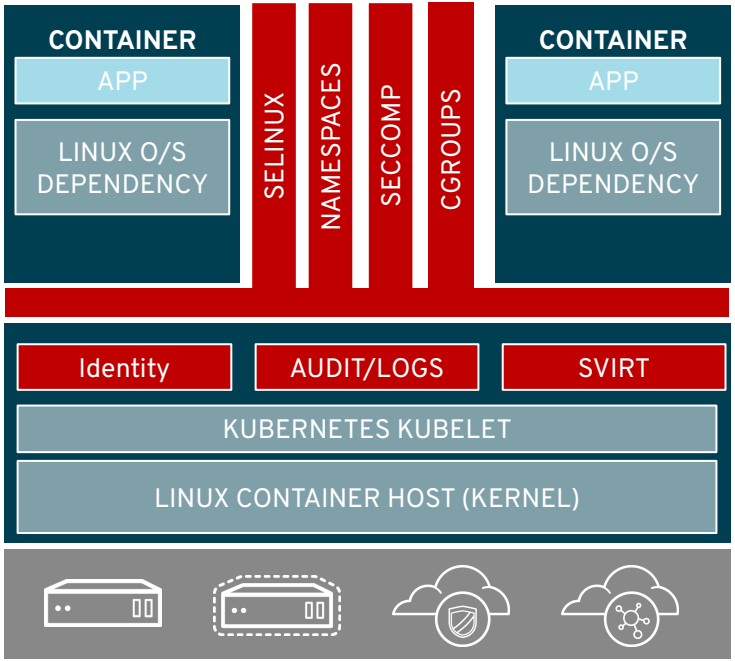
Any OCI-compliant container from any OCI registry (including docker)

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]

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



SELinux mitigates container runtime vulnerabilities

SELinux Mitigates container Vulnerability

January 13, 2017 | Joe Brockmeier

< Back to all posts Tags

A new CVE, (CVE-2016-9962), for the docker container runtime and runc were rec
released. Fixed packages are being prepared and shipped for RHEL as well as Fedo
CentOS. This CVE reports that if you `exec` d into a running container, the processe
the container could attack the process that just entered the container.

https://www.redhat.com/en/blog/selinux-mitigates-container-vulnerability

Latest container exploit (runc) can be blocked by SELinux

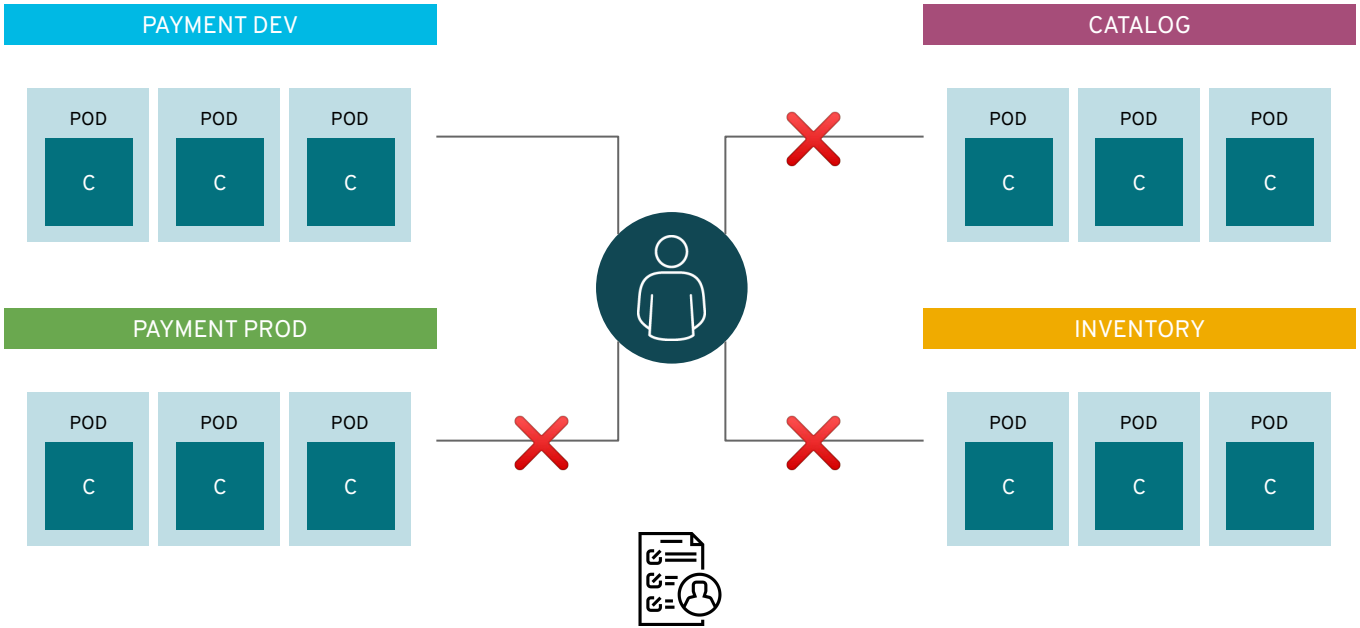
February 28, 2019 | Dan Walsh

< Back to all posts Tags: *Security*, *Containers*

A flaw in runc (CVE-2019-5736), announced last week, allows container processes to "escape" their containment and execute programs on the host operating system. The good news is that well-configured SELinux can stop it.

https://www.redhat.com/en/blog/latest-container-exploit-runc-can-be-blocked-selinux

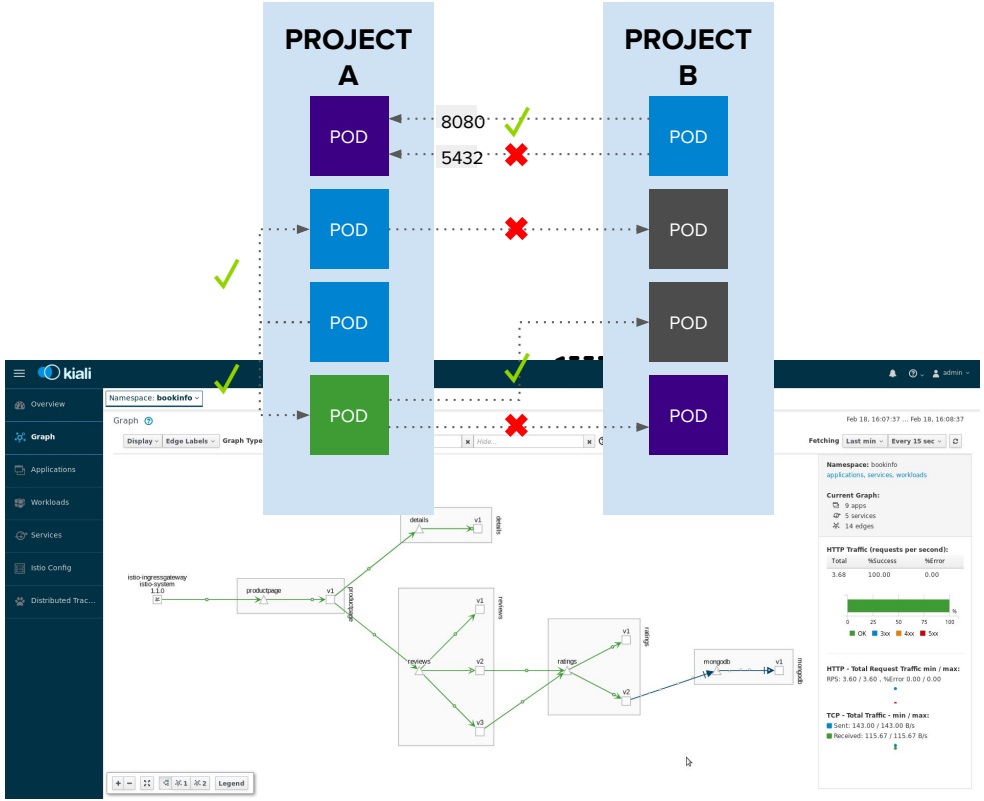
Projects isolate applications across teams, groups and departments



Enforced with IAM, RBAC, SELinux

OpenShift SDN and Service Mesh

- Integrated 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, including encryption and traceability
- Multiple options for ingress and egress control



Red Hat

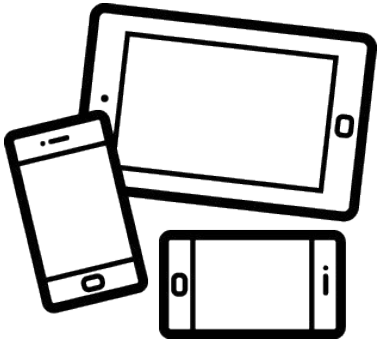
Secrets management & Protecting Application APIs

Secrets can be

- Made available as
  - Environment variables
  - Volume mounts
  - Or through 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
  - When best practices are followed

Consider configuring an API gateway for container platform & application APIs

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

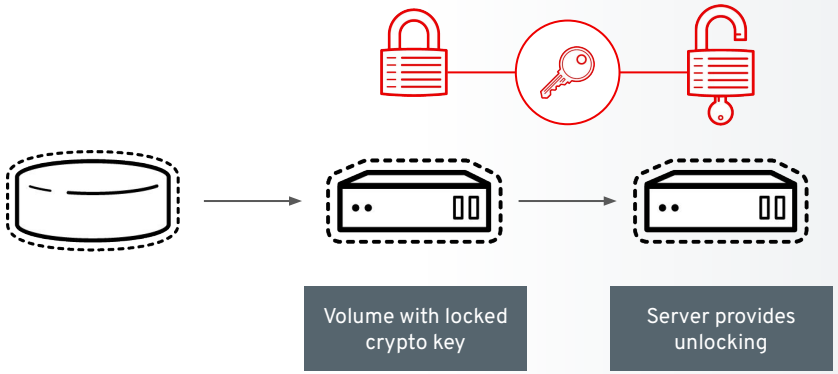


Red Hat

Attached storage

Secure storage by using

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



Observability:
Cluster and application monitoring, audit, and logging

Cluster monitoring is installed by default

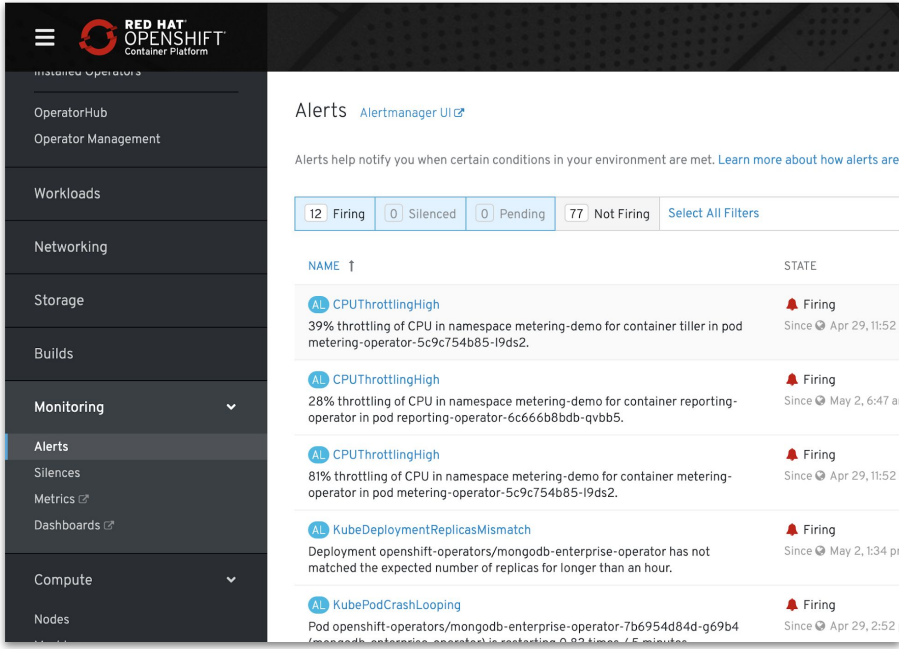
- No manual etcd monitoring configuration anymore
- Screens for managing Alerts & Silences
- Configuration via ConfigMaps and Secrets

Host and API server events are automatically audited

- Central audit policy configuration

Optional 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
- Logging pipelines aggregate logs for forwarding to the SIEM of your choice

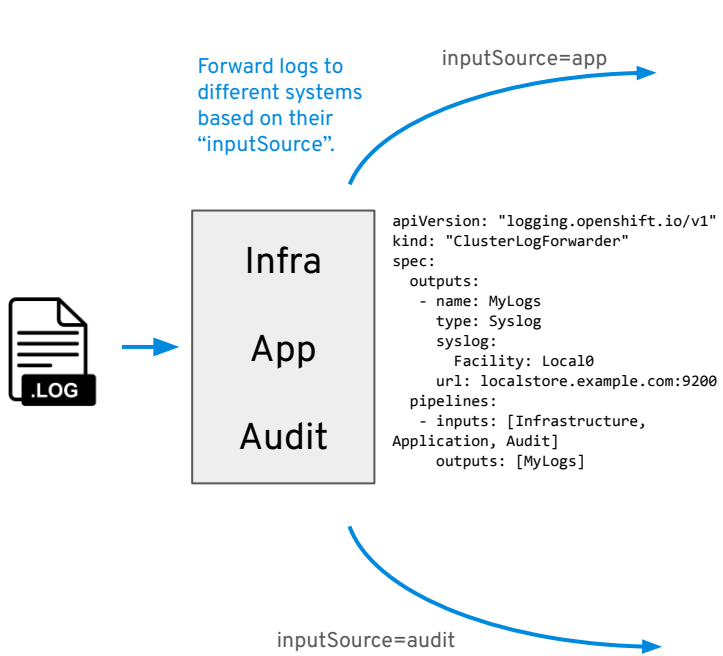


Red Hat

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.

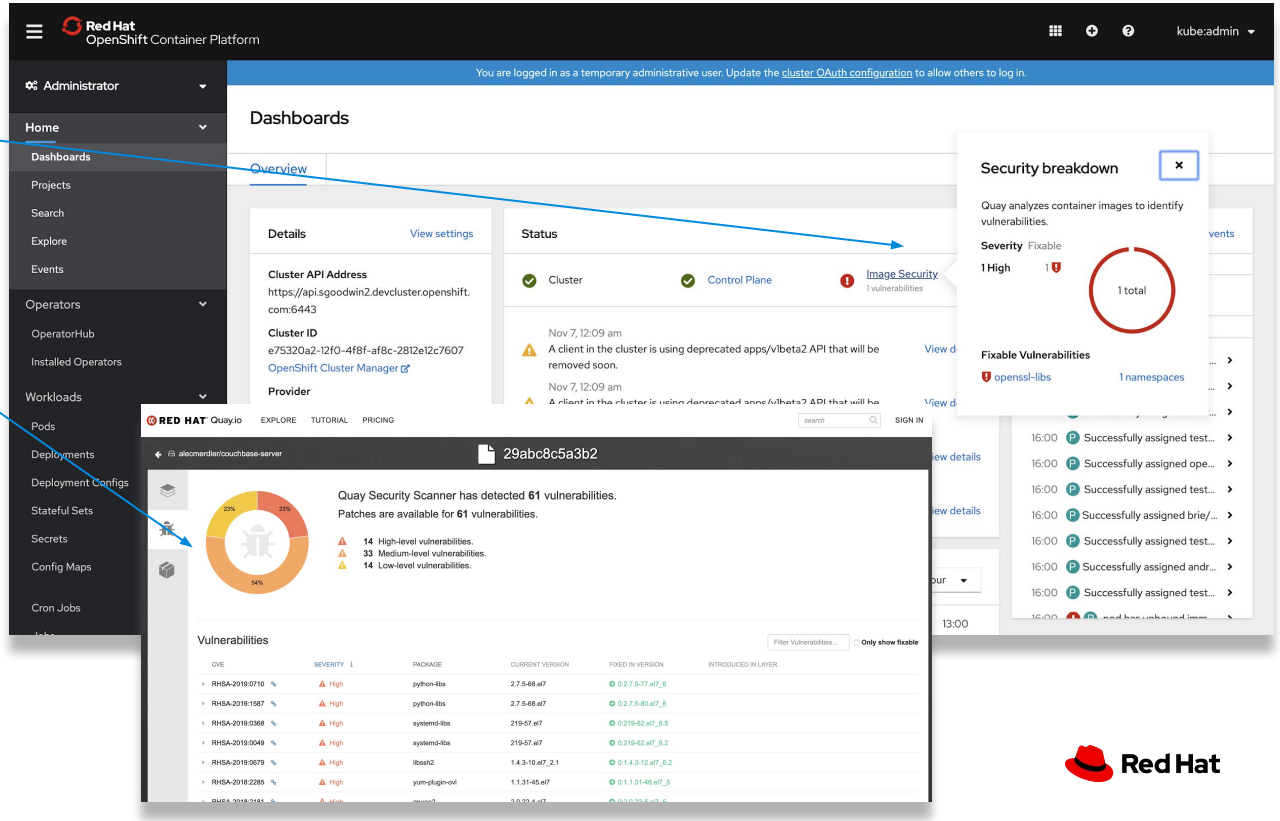


Red Hat

View Security Vulnerabilities with the Quay Operator

See vulnerability data for images managed by Quay in 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 Scans**; Planning to expand to other Vendors(TwistLock, Aqua, e.g.)
- *Only works for images managed by Quay*



Red Hat

A comprehensive approach to securing containers and Kubernetes

Detect

Trusted Content
- RH supply chain (backport fixes)
- RH Trusted Content with Health Index
- Universal Base Images
- Runtime images

Private Registry
- Integrated registry
- Quay with Clair for image scanning

Build Management
- Source2Image
- ImageStreams track changes to external images

Pipelines & developer tools
- IDE plugins for dependency analysis
- Code Ready Workspaces
- Jenkins / Tekton Pipelines

Protect

Configuration & Lifecycle Management
- OpenShift operators manage drift
- OLM manages operator privileges
- One maintenance window for the full stack
- Upgrades with zero application downtime
- Automate Compliance

Identity and Access Management
- Built-in token based authentication
- Supports 9 Identity Providers including AD/LDAP
- RBAC with Multi-Level Access Control

Platform Data Protection
- Encrypt secrets at rest (etcd datastore)
- All traffic to master nodes is encrypted by default; x.509 certificates for authentication
- Configure cipher suites

Deployment Policies
- SCC (Security Context Constraints)
- No privileged containers by default

Respond

Container isolation
- RHCOS Immutable user space
- SELinux+
- Secure boot
- LUKS volume encryption / FIPS mode
- Non-root containers

Network Isolation
- Ingress / Egress control
- Multus CNI plugin
- Network microsegmentation

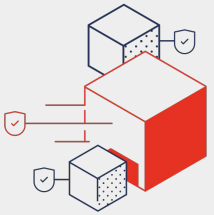
Application access and data
- Projects with SELinux annotations control Access to Resources
- Encrypt east / west traffic (Service Mesh)

Observability
- Host and K8s event audit on by default
- Monitoring on by default
- Applications can use cluster monitoring
- Service Mesh traceability
- Container Security Operator

A layered approach to container and Kubernetes security

Red Hat

Red Hat | StackRox



Red Hat Advanced Cluster Security

Powered by StackRox

Red Hat

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

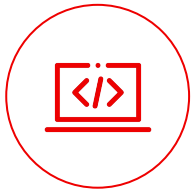
Benefits of a Kubernetes-native approach to security

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

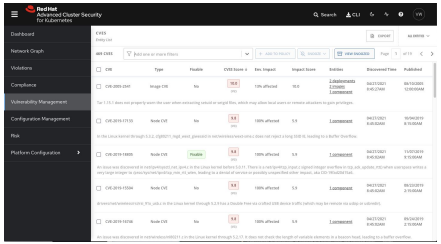
Red Hat Advanced Cluster Security for Kubernetes

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

Shift left

Secure supply chain

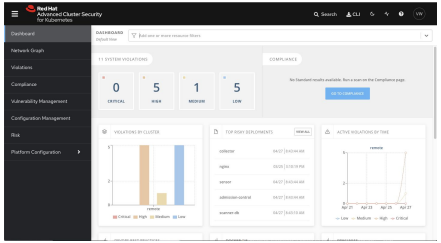
Extend scanning and compliance into development (DevSecOps)



Kubernetes security posture management (KSPM)

Secure infrastructure

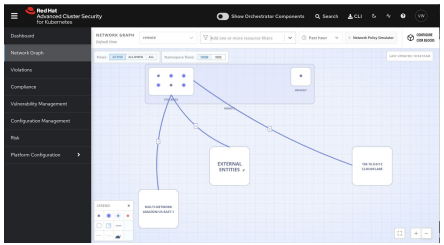
Leverage built-in Kubernetes security posture management to identify and remediate configurations and deployments



Cloud workload protection (CWPP)

Secure workloads

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



Red Hat Advanced Cluster Security: Use Cases

Security across the entire application lifecycle

Vulnerability Management

Protect yourself against known vulnerabilities in images and running containers

Network Segmentation

Apply and manage network isolation and access controls for each application

Configuration Management

Ensure your deployments are configured according to security best practices

Compliance

Meet contractual and regulatory requirements and easily audit against them

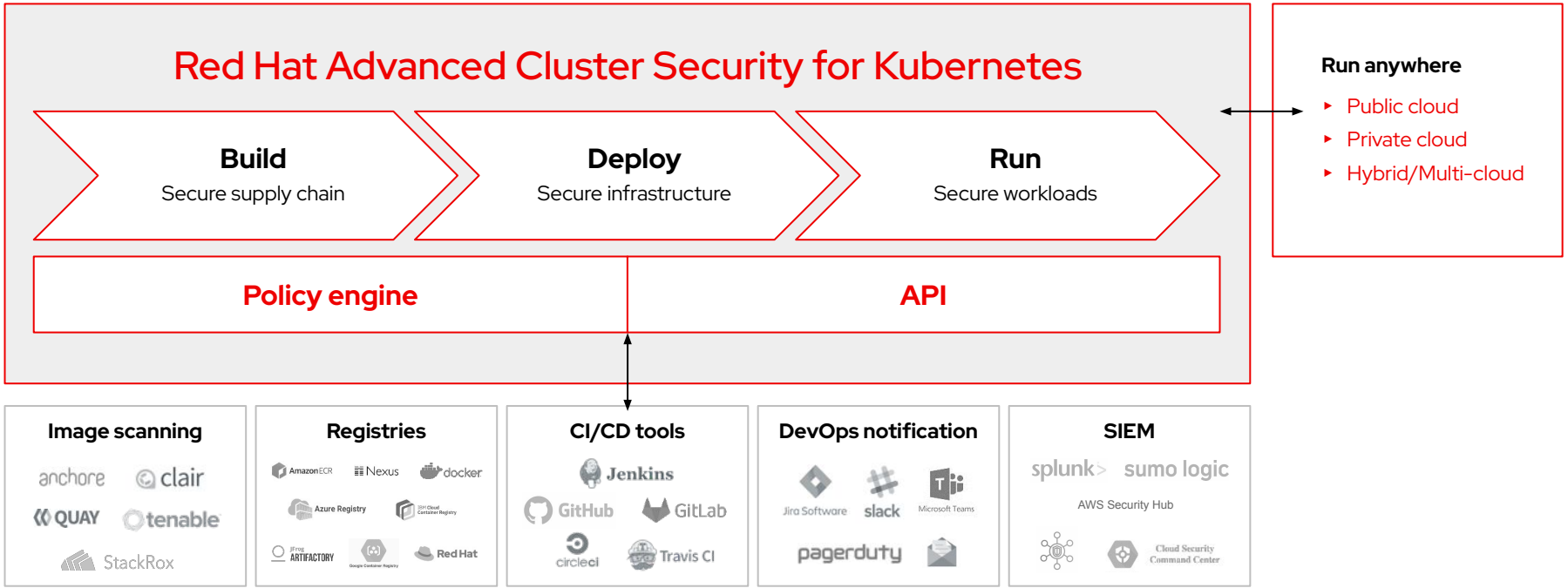
Risk Profiling

Gain context to prioritize security issues throughout OpenShift and Kubernetes clusters

Detection and Response

Carry out incident response to address active threats in your environment

The first Kubernetes-native security platform



Red Hat

Red Hat Advanced Cluster Security

Control	Protect	Detect & Respond
Vulnerability analysis	Image assurance and policy admission controller	Runtime behavioral analysis
App config analysis	Compliance assessments	Auto-suggest network policies
APIs for CI/CD integrations	Risk profiling	Threat detection / incident response
Trusted content	Kubernetes platform lifecycle	Container isolation
Container registry	Identity and access management	Network isolation
Build management	Platform data	Application access and data
CI/CD pipeline	Deployment policies	Observability
BUILD	DEPLOY	RUN

DevSecOps

Resource Management

Red Hat

Case study

sumo logic

"...(StackRox) natively deploys in Kubernetes, it fits right into our architecture."

George Gerchow,
Chief Security Officer

SaaS / Technology

Use Cases

- Vulnerability management
- Security compliance
- Security configuration management

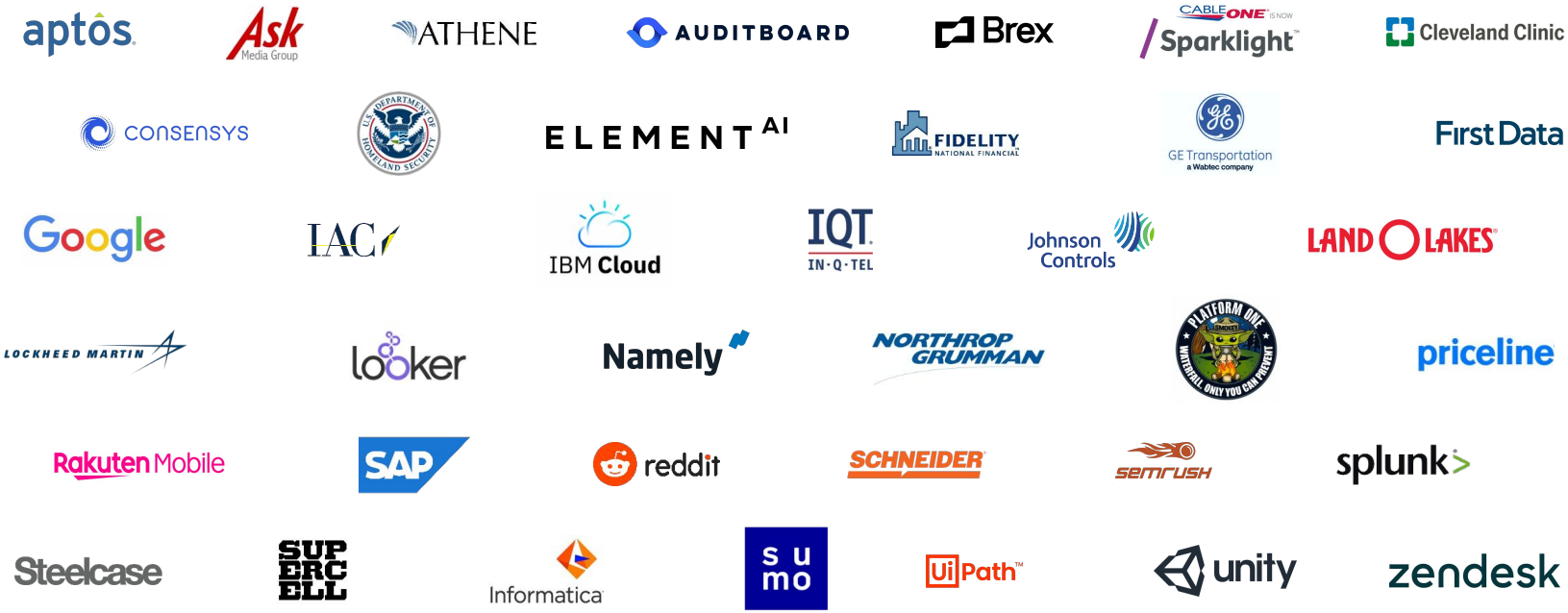
Challenges

- Moving workloads to Kubernetes
- Implementing security while building app dev environments
- Securing sensitive data

Results

- Common set of policies enforced across teams and the entire build-deploy-run lifecycle
- Able to maintain and demonstrate compliance with standards
- Improved app risk profiling for DevOps

Platform of choice for cloud-native innovators

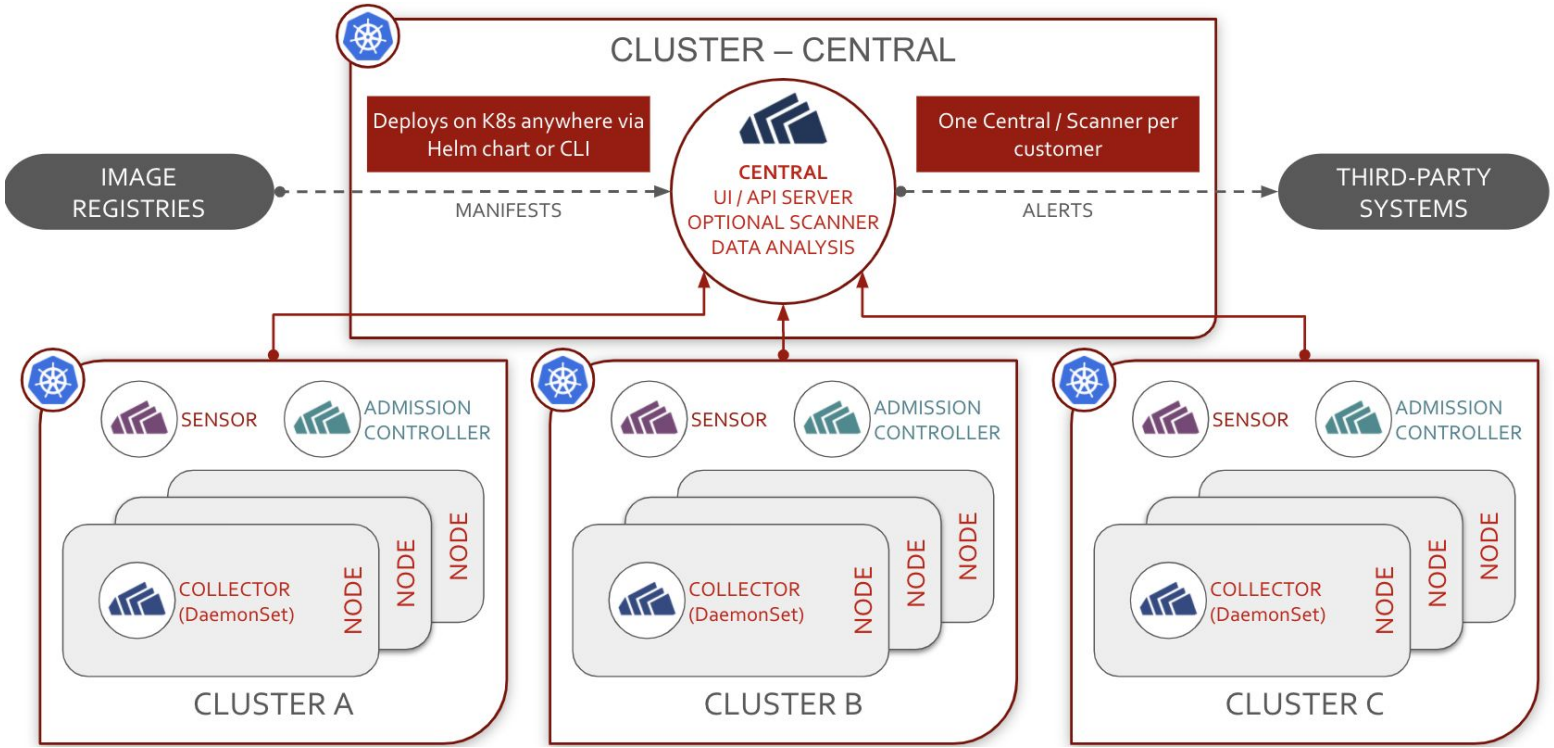


Red Hat

Startups trust us

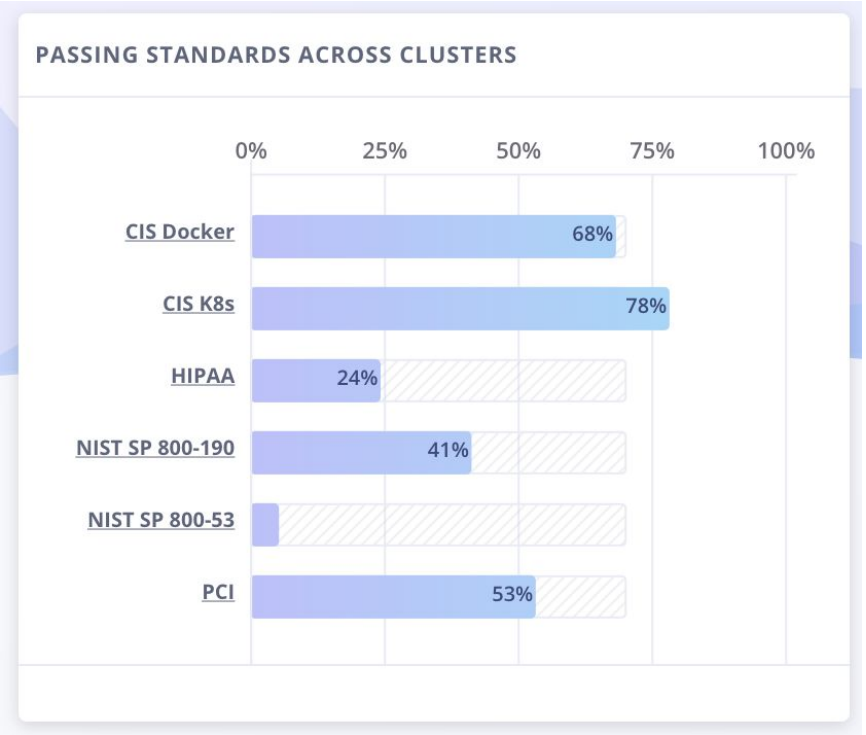
Red Hat

Architecture



Red Hat

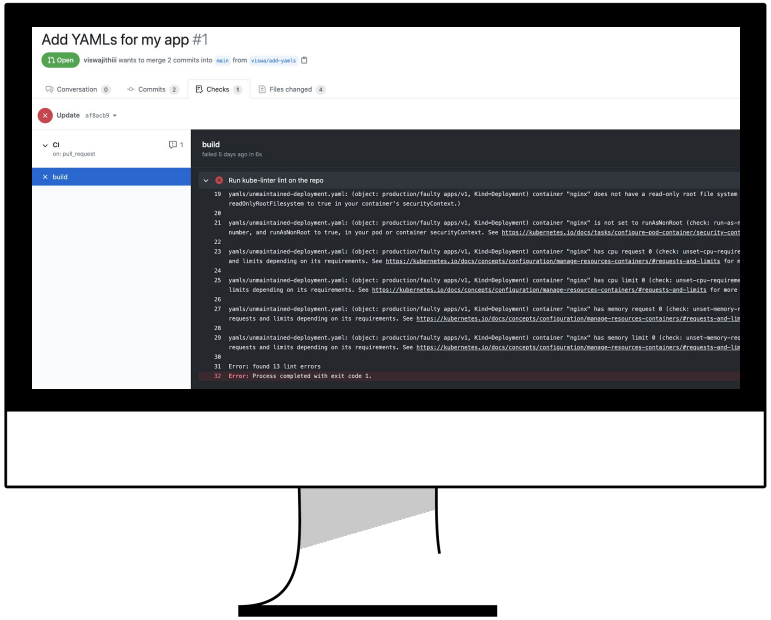
ACS Dashboard: Compliance & Vulnerabilities



RH Vulnerability Scanner Certification announcement

Red Hat

KubeLinter: Enforce Kubernetes security best practices



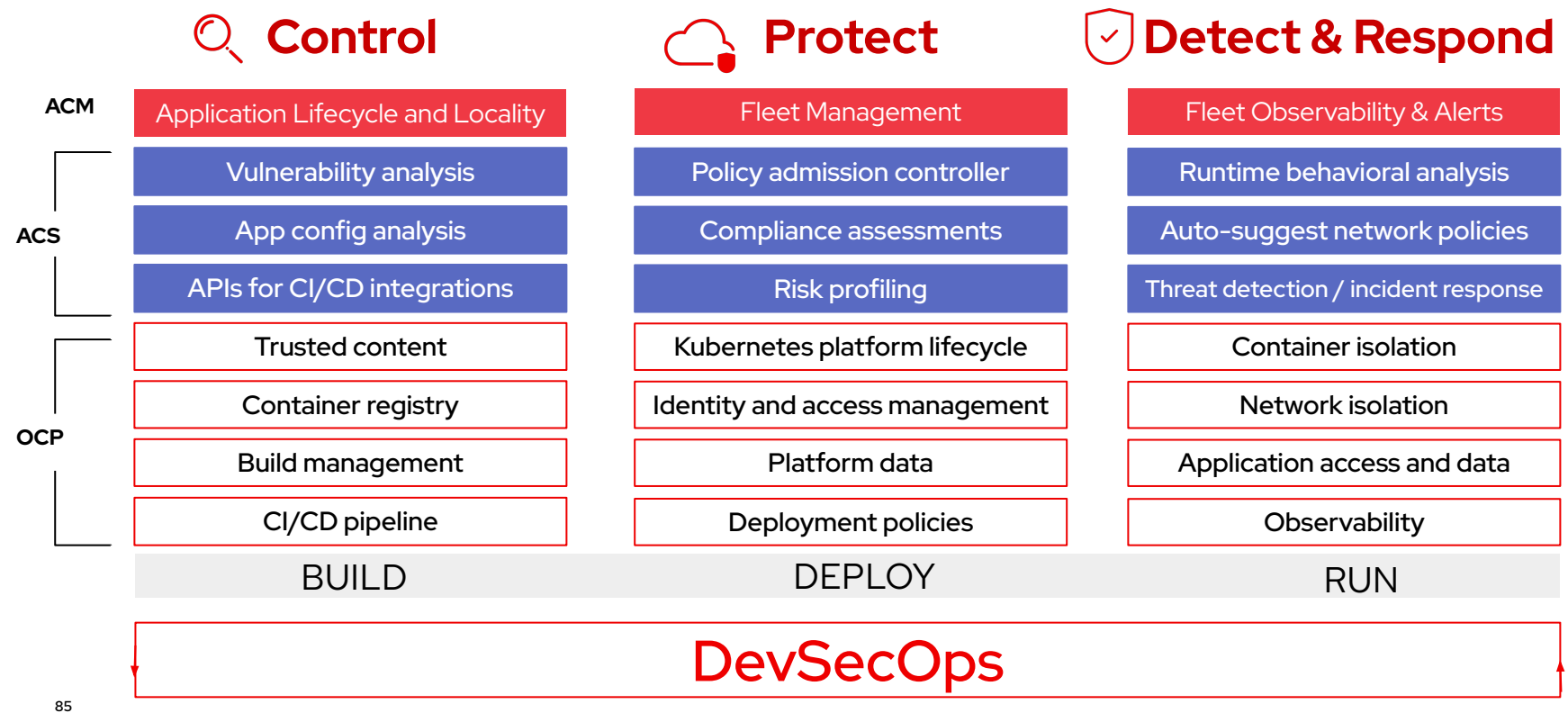
KubeLinter as a GitHub action

- Checks Kubernetes YAML files and Helm charts
- 16 default checks
- Extensible with custom checks
- Integrates with any CI tool

https://github.com/stackrox/kube-linter

Red Hat

OpenShift delivers continuous security



Red Hat

A comprehensive approach to securing containers and Kubernetes

Control

Trusted Content
- RH supply chain (backport fixes)
- RH Trusted Content with Health Index
- Universal Base Images
- Runtime images

Private Registry
- Integrated registry
- Quay with Clair for image scanning

Build Management
- Source2Image
- ImageStreams track changes to images

Pipelines & developer tools
- IDE plugins for dependency analysis
- Code Ready Workspaces
- Jenkins / Tekton Pipelines
- Integrate vulnerability scans and application configuration analysis
- Integrate Kube Linter

Protect

Configuration & Lifecycle Management
- OpenShift operators manage drift
- OLM manages operator privileges
- One maintenance window for the full stack
- Upgrades with zero application downtime
- Automate Compliance

Identity and Access Management
- Built-in token based authentication
- Supports 9 Identity Providers including AD/LDAP
- RBAC with Multi-Level Access Control

Platform Data Protection
- Encrypt secrets at rest (etcd datastore)
- All traffic to master nodes is encrypted by default; x.509 certificates for authentication
- Configure cipher suites

Deployment Policies
- SCC (Security Context Constraints)
- No privileged containers by default
- Policy-based admission control

Detect & Respond

Container isolation
- RHCOS Immutable user space
- SELinux+
- Secure boot
- LUKS volume encryption / FIPS mode
- Non-root containers

Network Isolation
- Ingress / Egress control
- Multus CNI plugin
- Network microsegmentation
- Autogenerate network policies

Application access and data
- SELinux annotations control Access to Resources
- Encrypt east / west traffic (Service Mesh)
- Baseline application behavior and alert on changes

Observability
- Report on images with known vulnerabilities
- Host and K8s event audit on by default
- Monitoring on by default
- Applications can use cluster monitoring
- Network policy graph
- Service Mesh traceability

A layered approach to container and Kubernetes security

Roadmap*: Identity, Integrity, Observability

Control

Trusted Application Identity

Improve supply chain security with solutions to verify identity of users, images, deployments, config data

Keyless signatures
Tekton CD chains
Encrypted containers
Rootless builds

Protect

Platform Integrity

Deliver platform integrity with attestation and verification as a service; Mitigate risk by expanding isolation capabilities

Keylime / IMA
Kube support for user namespaces
Externally managed control planes
Intel SGX support

Detect & Respond

Observe, Analyze, Remediate

Active recommendations to automate remediation based on deep data collection and analysis

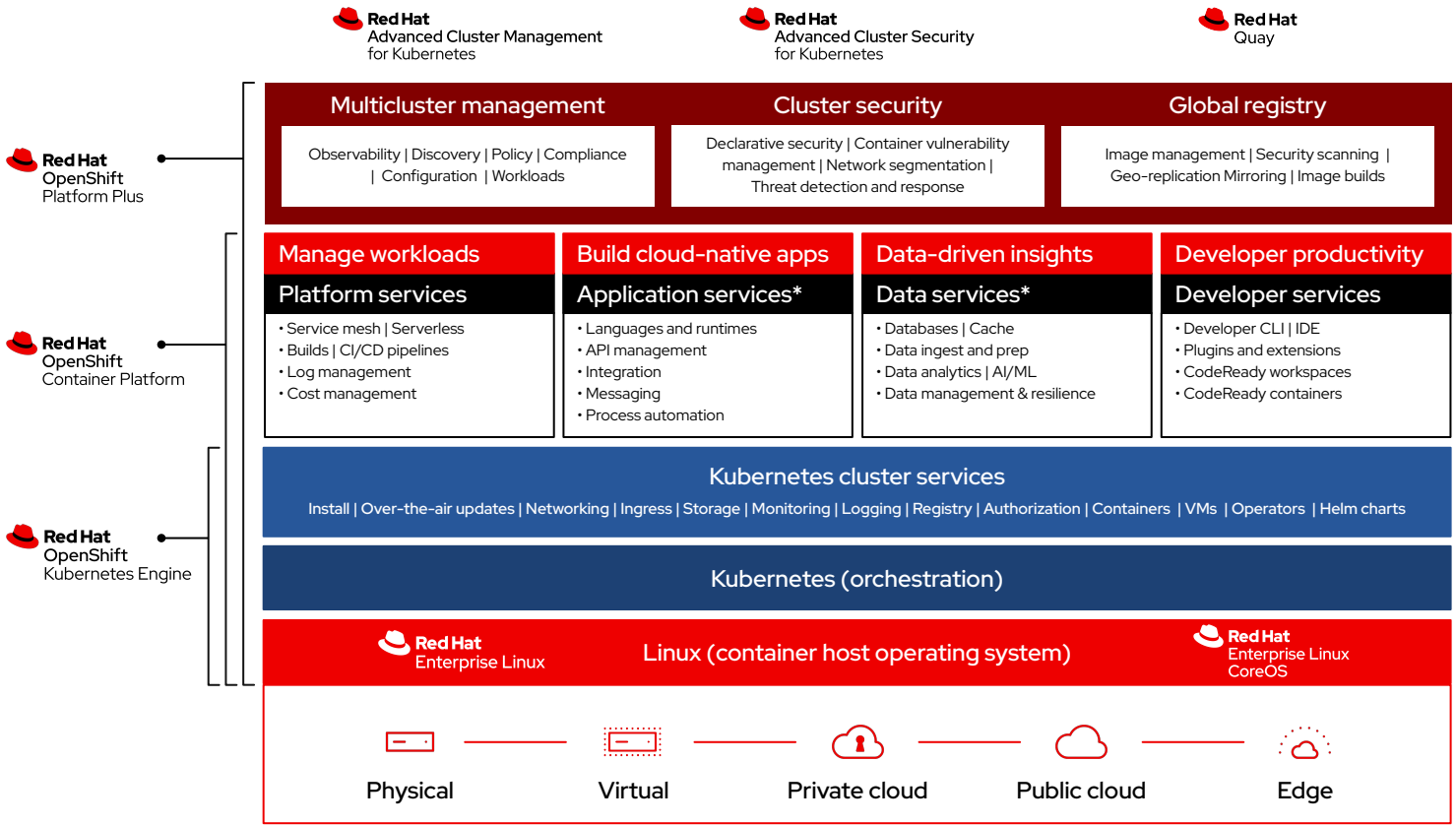
Security Profile operator
Deep network observability
Service Mesh recommendations

BUILD | DEPLOY | RUN

DevSecOps

*Subject to change without notice

OpenShift Platform Plus



* Red Hat OpenShift® includes supported runtimes for popular languages/frameworks/databases. Additional capabilities listed are from the Red Hat Application and Data Services portfolio.

Red Hat

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


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat

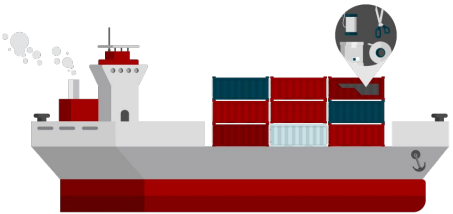
Containers and Kubernetes enable hybrid cloud

Containers make it easier to deliver applications faster

- Packaging dependencies with application
- Simpler, lighter, greater utilization then VMs
- Portable across environments
- Shorter development cycle
- Quickly deploy to any environment
- And much more

Kubernetes makes it easier to manage containerized applications

- Scaling
- Resiliency
- Networking and Routing
- Persistent data storage
- Platform HA
- Application HA



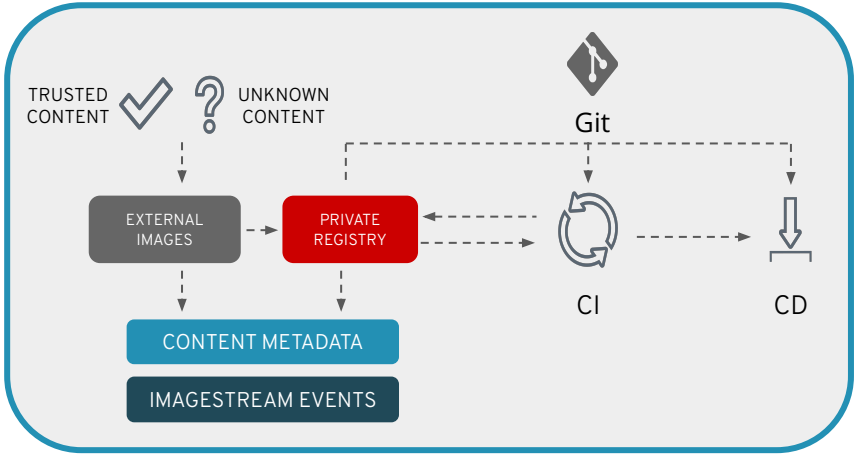
Build: Control Application Security

Shift Security left

Best practices

- Use trusted sources for external content
- Use a private registry with integrated vulnerability scanning to manage images
- Automate your container pipeline with Jenkins or Tekton
- Integrate security into the pipeline
  - Image vulnerability analysis
  - App configuration analysis

Automate multi-cluster application deployment / locality



Red Hat Quay

Red Hat OpenShift

Red Hat CodeReady

Red Hat Advanced Cluster Security

Red Hat Advanced Cluster Management

Red Hat

Deploy: Protect the platform

- Configuration and lifecycle management
  - Container optimized OS for controlled immutability
  - Policy-based fleet configuration management

Red Hat Enterprise Linux CoreOS
Red Hat Advanced Cluster Management

- Identity and Access Management

Red Hat OpenShift

- Protect platform data at rest and in motion

Red Hat OpenShift

- Automated compliance management

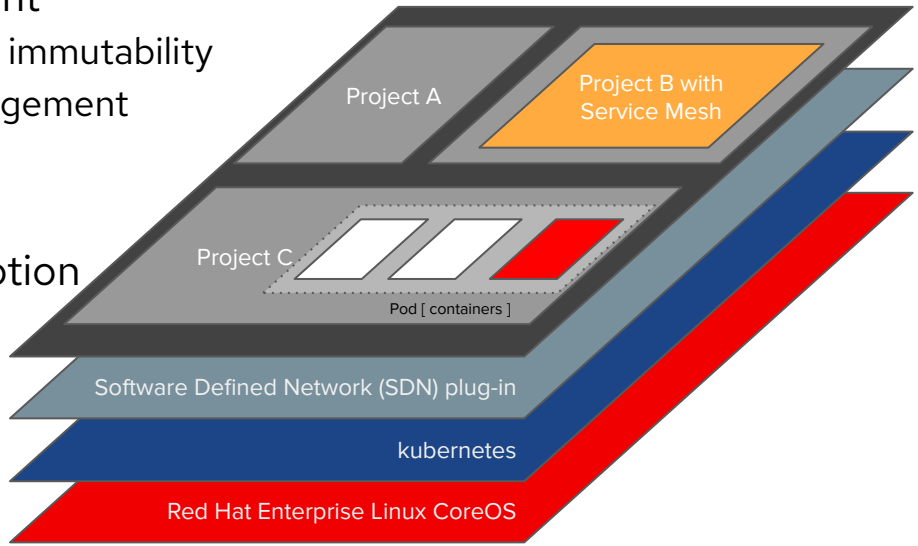
Red Hat OpenShift

- Automated risk assessment

Red Hat Advanced Cluster Security

- Policy based deployment
  - Prevent admission of privileged pods
  - Prevent admission of pods with critical vulnerabilities

Red Hat OpenShift
Red Hat Advanced Cluster Security



Red Hat

Run: Detect and respond to runtime issues

Red Hat Enterprise Linux CoreOS

- Container isolation & resource management

Red Hat OpenShift

Red Hat Advanced Cluster Security

- Application and network isolation
  - Integrated RBAC for fine grained control
  - Network policies for SDN microsegmentation
  - Auto-suggest network policies
  - Service Mesh encrypts pod-to-pod traffic

Red Hat OpenShift

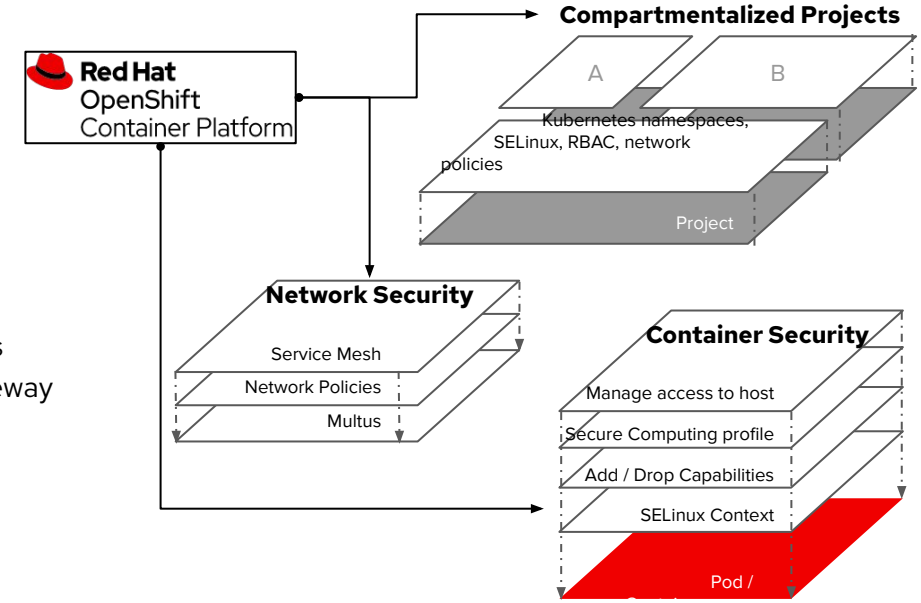
- Securing application access
  - Multiple options for managing ingress and egress
  - RH SSO for user management / 3Scale API Gateway

Red Hat OpenShift

- Observability
  - Application monitoring & logging
  - Network policy and service mesh visualization

Red Hat Advanced Cluster Security

- Threat detection and response
  - Alert or kill pods based on anomalous behavior
  - Detect privilege escalation, cryptomining



Red Hat

The OpenShift Security Guide is Available

- OpenShift Security Guide is released on Amazon (Kindle format)
- Also available to our customers via the customer portal - here
- We are working on a page not requiring Red Hat login for download
- Amazon Print On Demand option coming soon

Red Hat

Customers in Every Industry and Geography

FINANCIAL SERVICES

INSURANCE

AUTOMOTIVE

RETAIL

LOGISTICS

TRAVEL

HOSPITALITY

MEDIA | TELCO

Red Hat

Support for a broad set of Applications and Use-Cases

MODERNIZE APPS

WEB APPS

CLOUD NATIVE DEV

MULTI-CLOUD

MOBILE

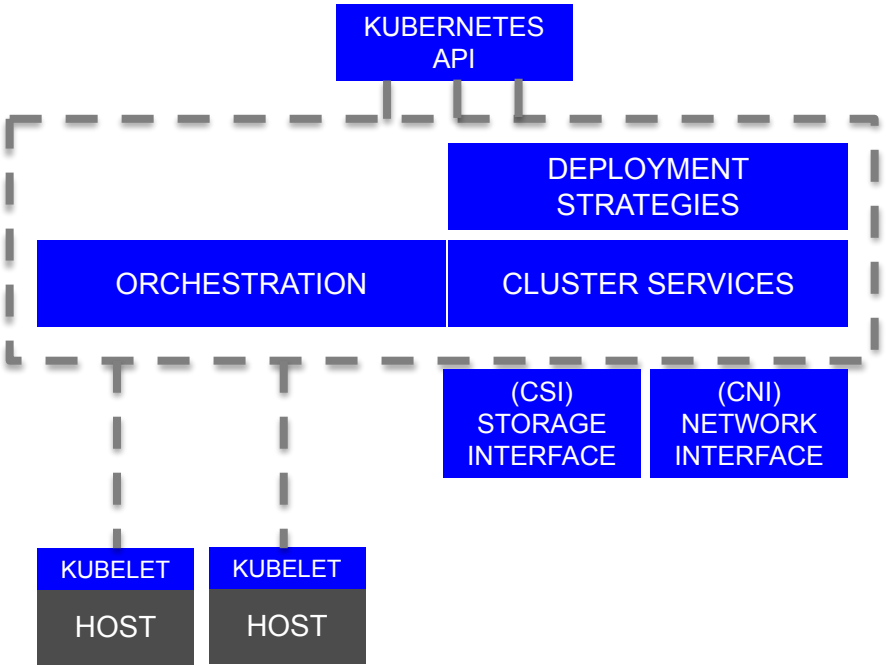
BIG DATA | ANALYTICS

AI | ML

IOT

Red Hat

Kubernetes vs. Application Platforms



Platform Services	Application Services	Developer Services
Service Mesh Serverless Builds ⁝ CI/CD Pipelines Logging ⁝ Chargeback	Databases Languages Runtimes ⁝ Integration Business Automation 100+ ISV Services	Developer CLI ⁝ VS Code extensions ⁝ IDE Plugins Code Ready

AUTOMATED OPERATIONS

KUBERNETES

Linux (or Windows) OS

Red Hat

ACS and ACM

Advanced Cluster Security

Implement and enforce security policies at build, deploy and runtime

- Respond to security incidents with runtime threat detection & behavioral analysis
- Create trusted application deployments with security and operational readiness best practice assurance policies
- Enable teams to visualize, audit and recommend network policy improvements
- Reduce known attack surface with automated vulnerability & compliance analysis and workflows

Advanced Cluster Management

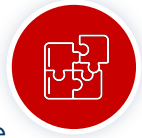
Automate policy-based cluster and application governance, across the fleet

- Operations teams need a centralized tool for hybrid cloud cluster lifecycle management
- SecOps can leverage policy-based compliance to audit/enforce desired state cluster configuration
- Manage cluster and application configuration as code through GitOps deployments
- Capitalize on existing investments using open source integration like OPA Gatekeeper
- Optimize compute resources and gain insights on operational health with multi-cluster observability

ACS and ACM complement each other

Compliance

- ACM is focused on providing visibility to operations teams & helping teams to ensure repeatable controls across clusters
- ACS is focused on providing evidence to auditors and deep insight to security professionals.
- Both solutions can provide different levels of compliance controls, both of which are valid use cases.



Policy

- Policy is a generic term. While both solutions enable policy there are different levels to policy.
- ACM is focused on repeatable policy that can be used at the cluster level
- ACS is focused on repeatable policy that can be used at the workload level.
- Each solution generally enforces different types of policy.
- ACM is used to distribute all policy types.

Vulnerability Scanning

- ACM provides visibility & ACS provides deep details
- Use ACS to triage and fix issues with developers and security teams
- Use ACM to create broad visibility for administrators and operators.
- Use ACM with third parties and our ecosystem. (Sysdig, BlackDuck)

Observability

- ACM allows you to obtain visibility into cluster operational health across multiple clusters
- ACS allows you to obtain visibility into security risk and to observe network traffic flow
- Use both to provide a broad and deep view of fleet health and compliance.

More on RHEL CoreOS

Red Hat

Key characteristics of RHEL CoreOS

- **Transactional updates** - RHCOS is distributed as an image and each operating system update is versioned and distributed as containers. Major releases (and some z stream releases) provide new boot images. The OS always boots into a known-good version; this is similar in principle to how container images are managed and deployed.
- **Immutable management** - RHCOS is built to be managed in an immutable fashion by the Machine Config Operator and Kubernetes API. While certain parts of the OS are truly immutable, others are not. Immutable management enables us to spawn new nodes and ensure that the cluster is the single source of truth for provisioning configurations, OS versions, and run-time configuration. Apart from consistency, this also enables elastic clusters to spawn and destroy nodes.

Key characteristics of RHEL CoreOS (cont'd)

- **Applications need to run in containers** - Installing RPMs on RHCOS is not supported. The OS is built to run all processes outside the OS as a container. This allows us to guarantee successful upgrades and automation beyond what a traditional operating system can deliver.
- **rpm-ostree** - This is the technology used to assemble the operating system. RHEL RPMs are used to create the OS images, and versions can easily be queried using the rpm command.
  - **/usr** is where the operating system binaries and libraries are stored and is read-only.
  - **/etc**, **/boot**, **/var** are writable on the system but only intended to be altered by the Machine Config Operator.
  - **/var/lib/containers** is the graph storage location for storing container images.

Network Security: A deep dive

Ingress

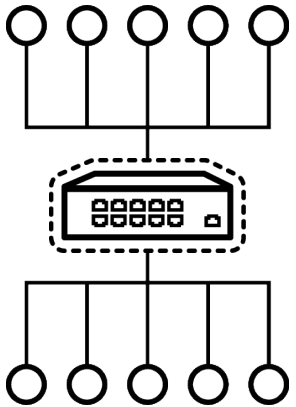
CNI: Multus & OVN

Egress

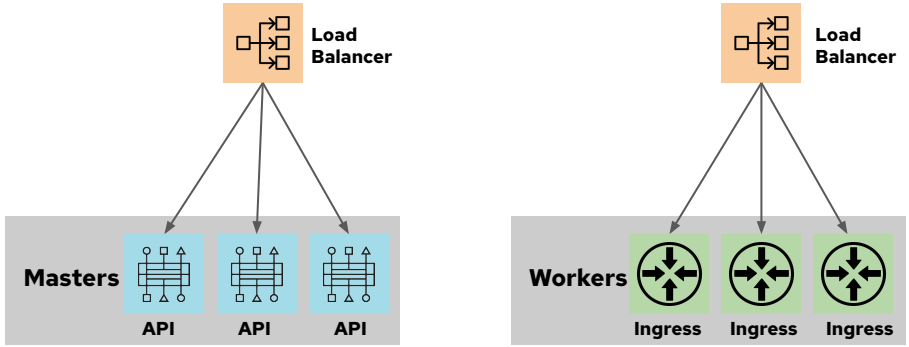
Service Mesh

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



Ingress Traffic

- 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

```yaml
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

Ingress Access Logging

- There is a new API field on the IngressController resource to configure it:
  - Ability to enable access logs
  - Choice of logging to a pod container or to a Syslog server
  - Options to configure HTTP log format and Syslog facility
  - *Limitation:* Syslog endpoint must be UDP
- Log the hostname of a node from which the log message originated (send-log-hostname) enabled

Log to Sidecar Container

```
$ oc -n openshift-ingress-operator patch ingresscontroller/default --type=merge
--patch='{"spec":{"logging":{"access":{"destination":{"type":"Container"}}}}}'
```

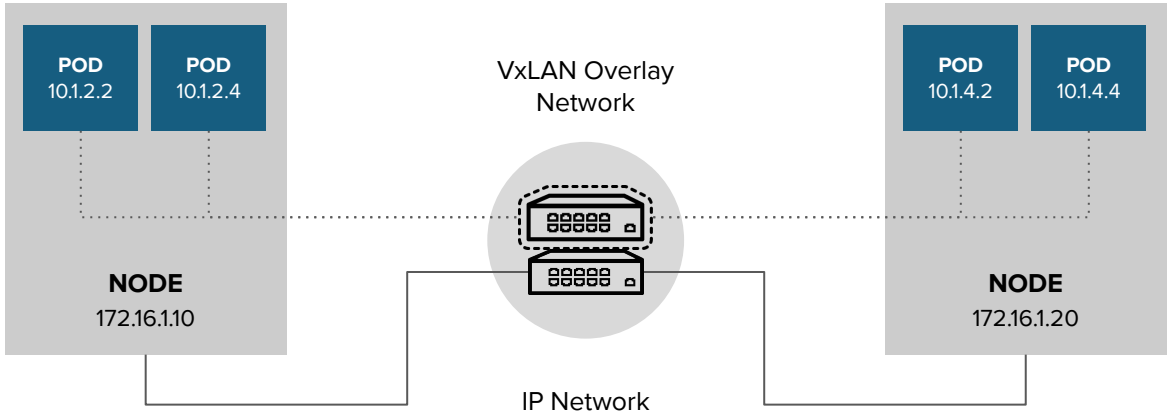
Log to a "facility" on a Syslog server

```
$ oc -n openshift-ingress-operator patch ingresscontroller/default --type=merge
--patch='{"spec":{"logging":{"access":{"destination":{"type":"Syslog","syslog":{"addr
ess":"1.2.3.4","port":10514,"facility":"audit"}}}}}}'
```

View the Logs

```
$ oc -n openshift-ingress logs deploy/router-default -c logs --tail=10 --follow
```

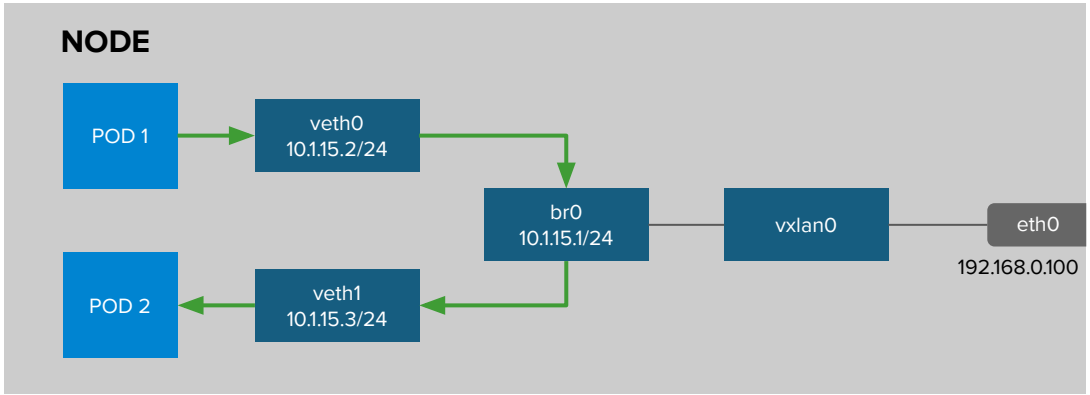
OpenShift pod to pod networking



Red Hat

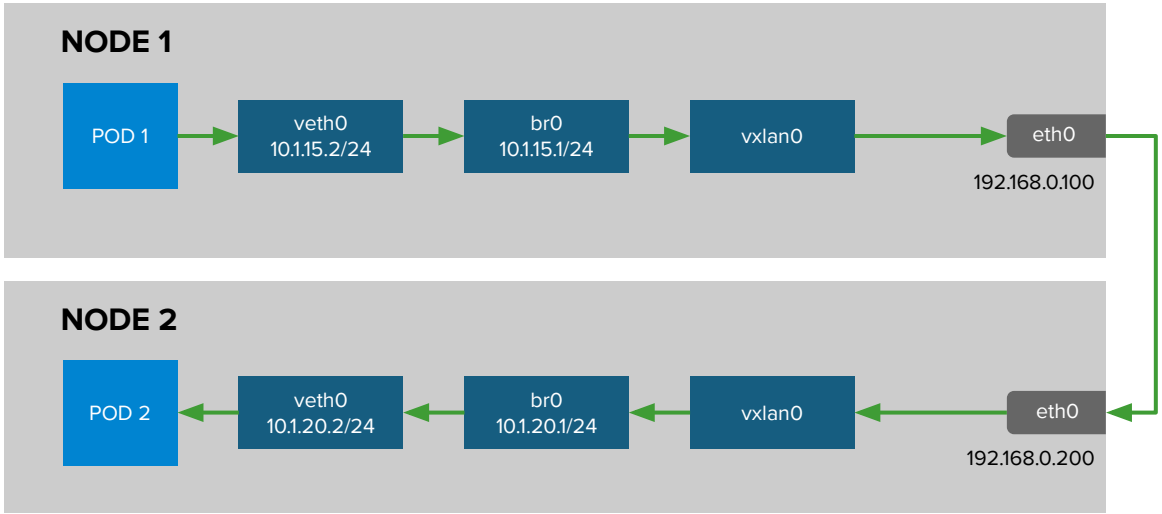
OPENSHIFT SDN - OVS PACKET FLOW

Container to Container on the Same Host



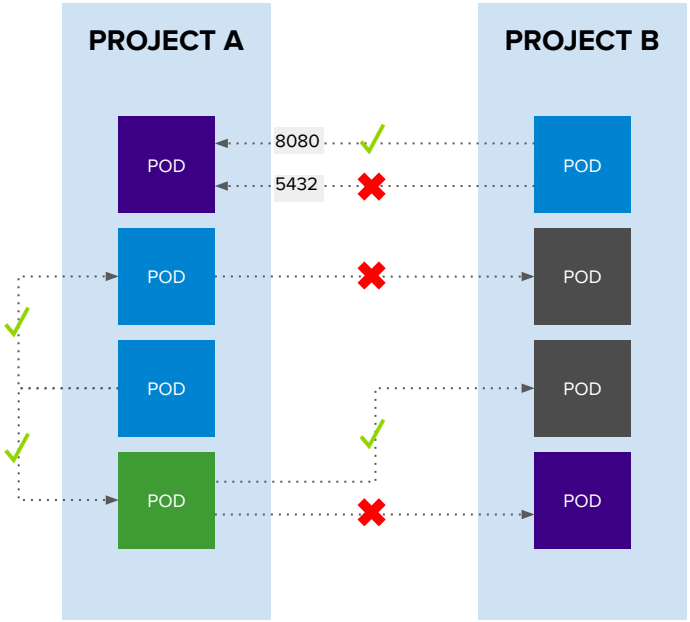
OPENSHIFT SDN - OVS PACKET FLOW

Container to Container on the Different Hosts



OpenShift SDN

Network policy enabled by default



Example Policies

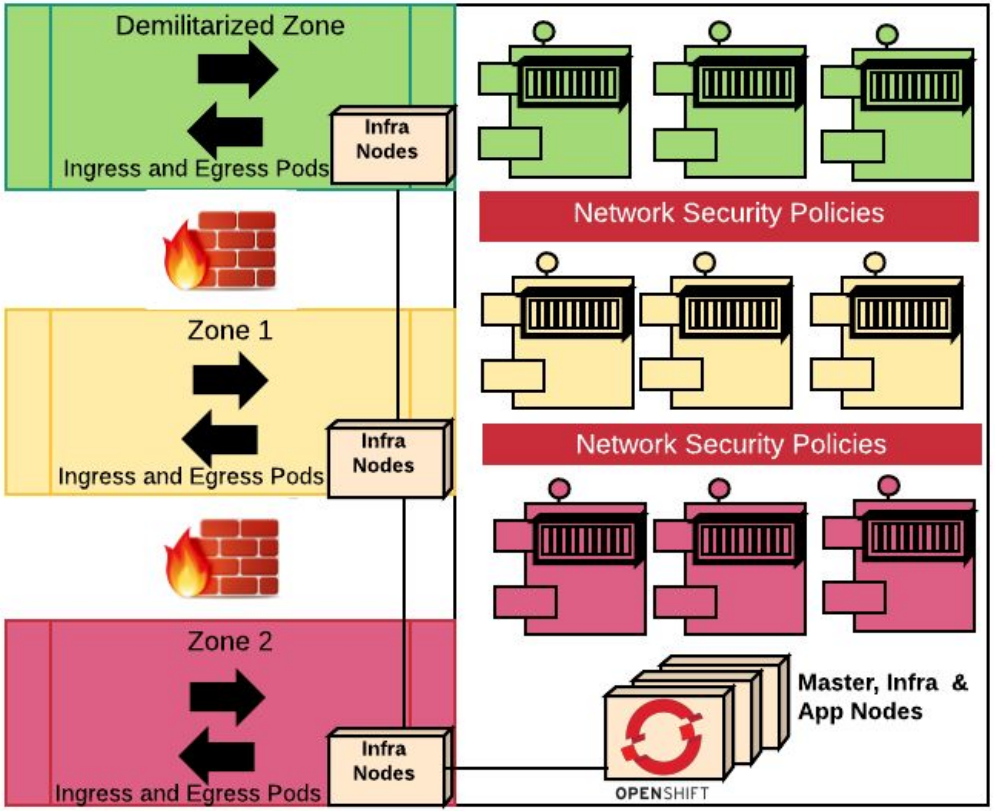
- Allow all traffic inside the project
- Allow traffic from green to gray
- Allow traffic to purple on 8080

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


Red Hat

OpenShift cluster with multiple zones

Using multiple ingress controllers, network policies, multiple egress pods



Application pods run on one OpenShift Cluster.

Microsegmented with Network Security policies.

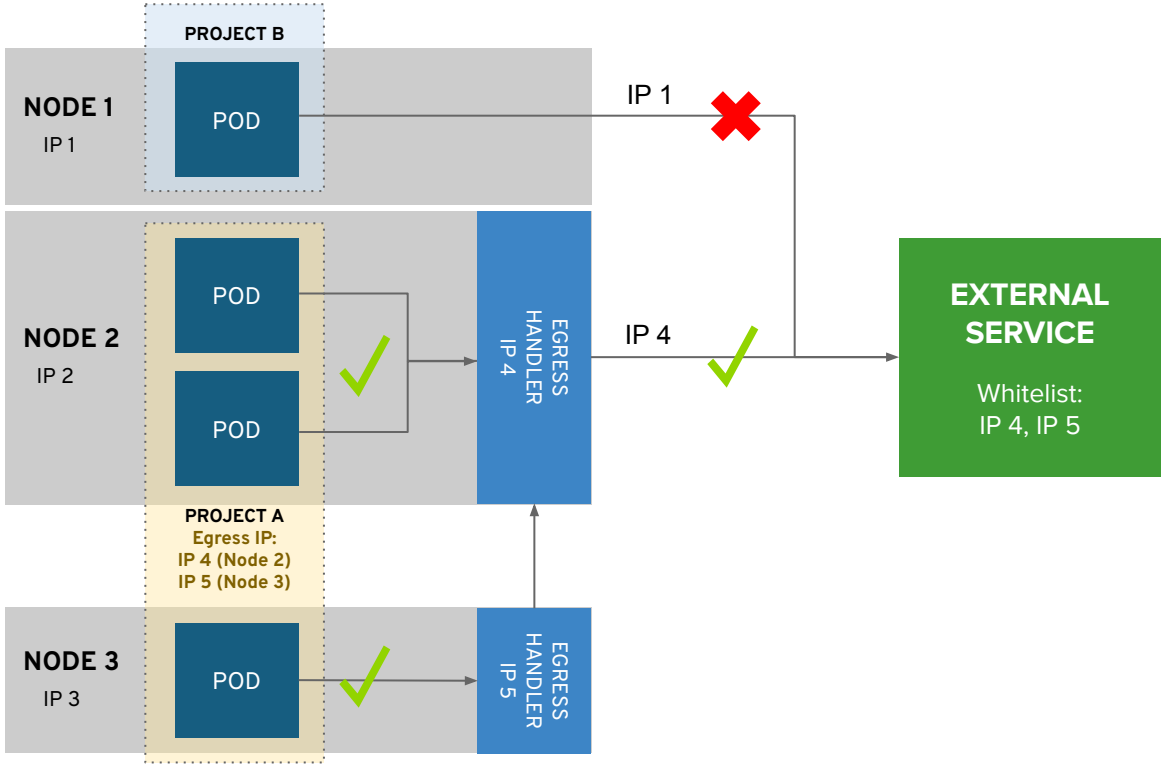
Infra Nodes in each zone run Ingress and Egress pods for specific zones.

If required, physical isolation of pods to specific nodes is possible with node-selectors. But that can reduce worker node density.

Red Hat

Controlling Egress Traffic

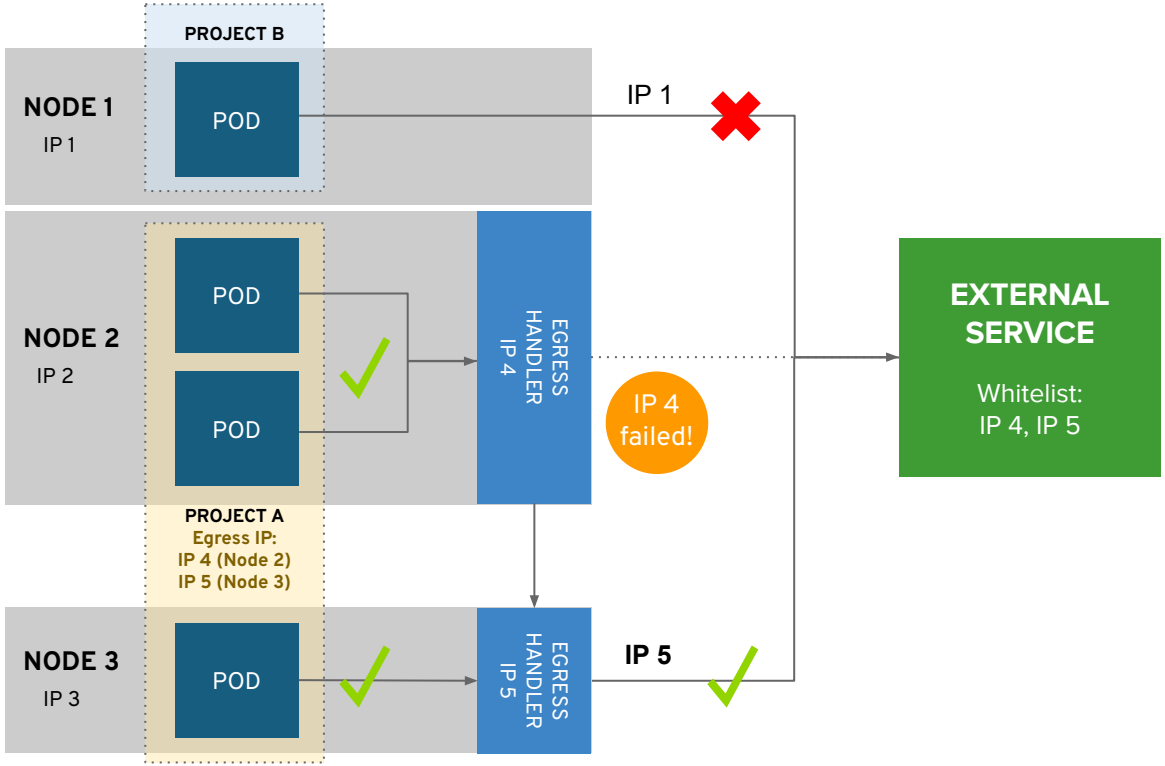
Egress IP high availability (multiple IPs)



Red Hat

Controlling Egress Traffic

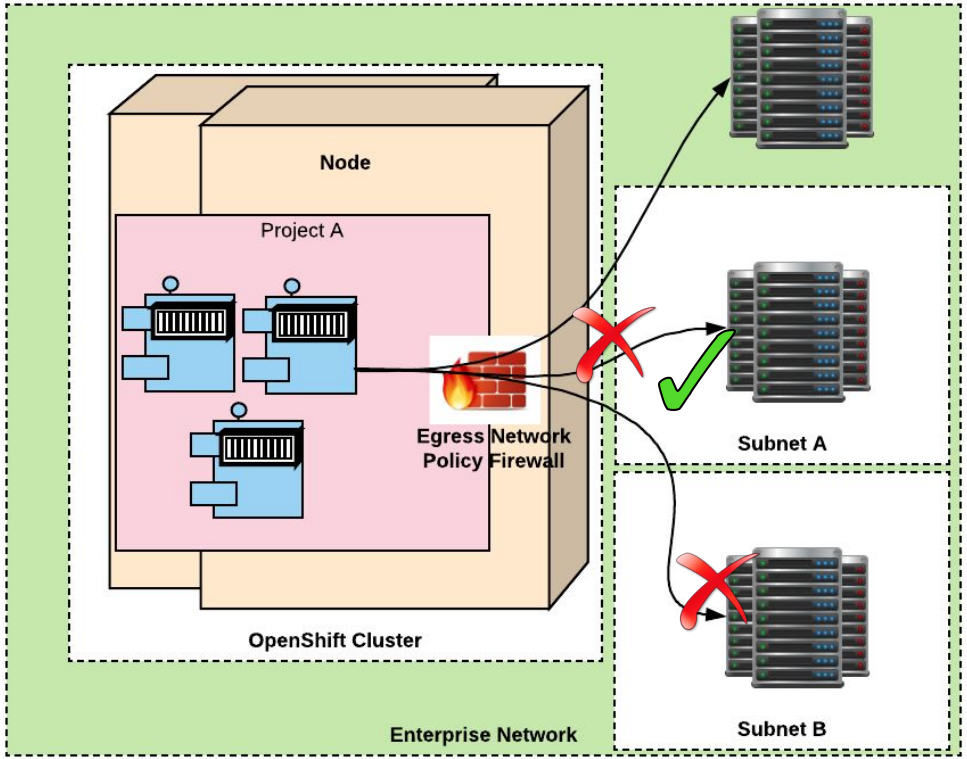
Egress IP high availability (multiple IPs)



Red Hat

Egress firewall to limit access

Cluster admin can limit the external addresses accessed by some or all pods



Examples:

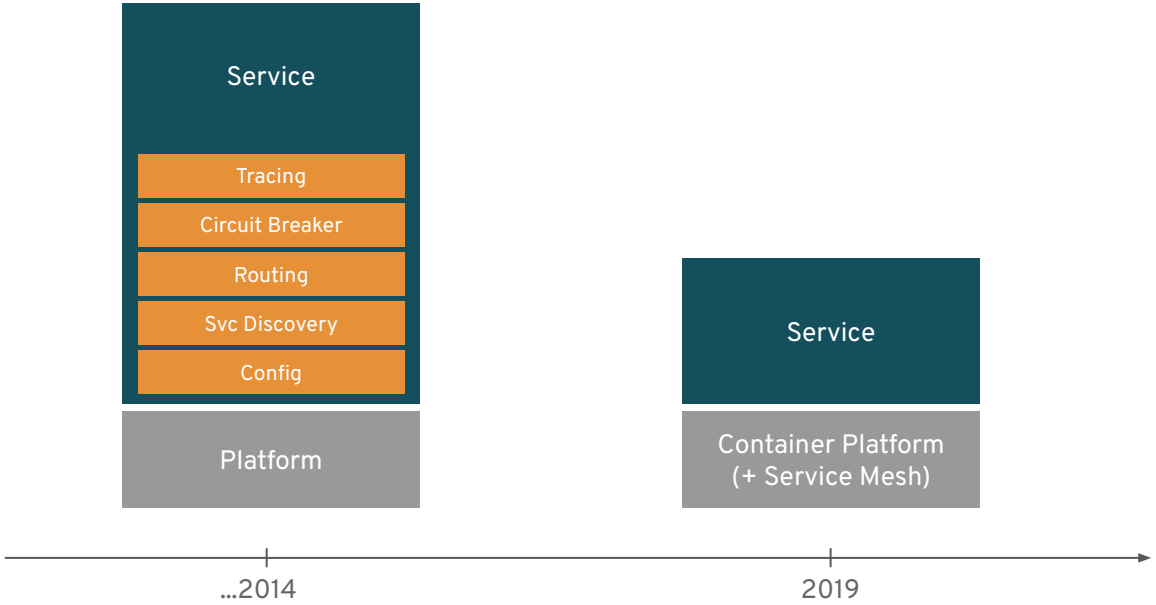
A pod can talk to hosts (outside OpenShift cluster) but cannot connect to public internet

A pod can talk to public internet, but cannot connect to hosts (outside OpenShift cluster)

Public Internet

A pod cannot reach specific subnets/hosts

Microservices evolution

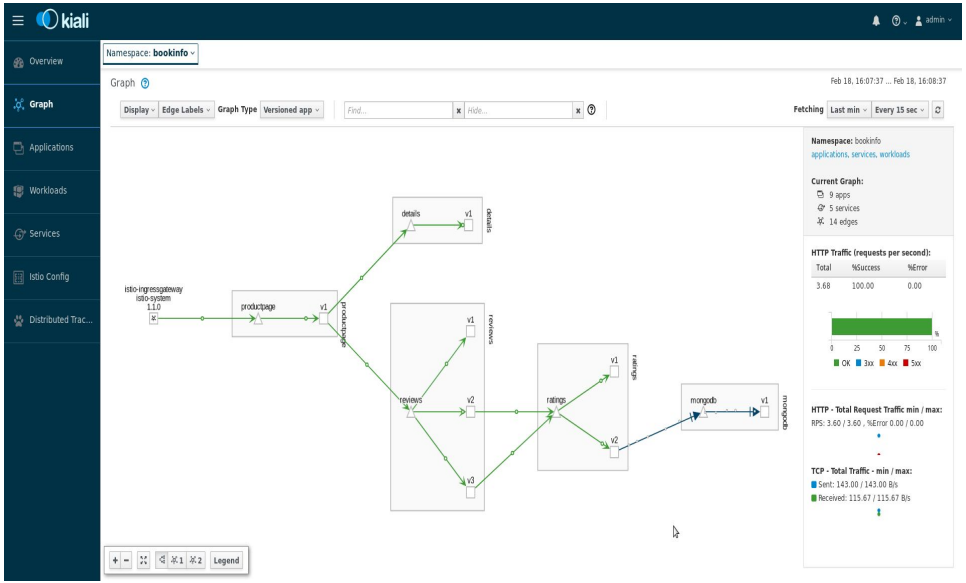


Red Hat

Secure microservices with Service Mesh

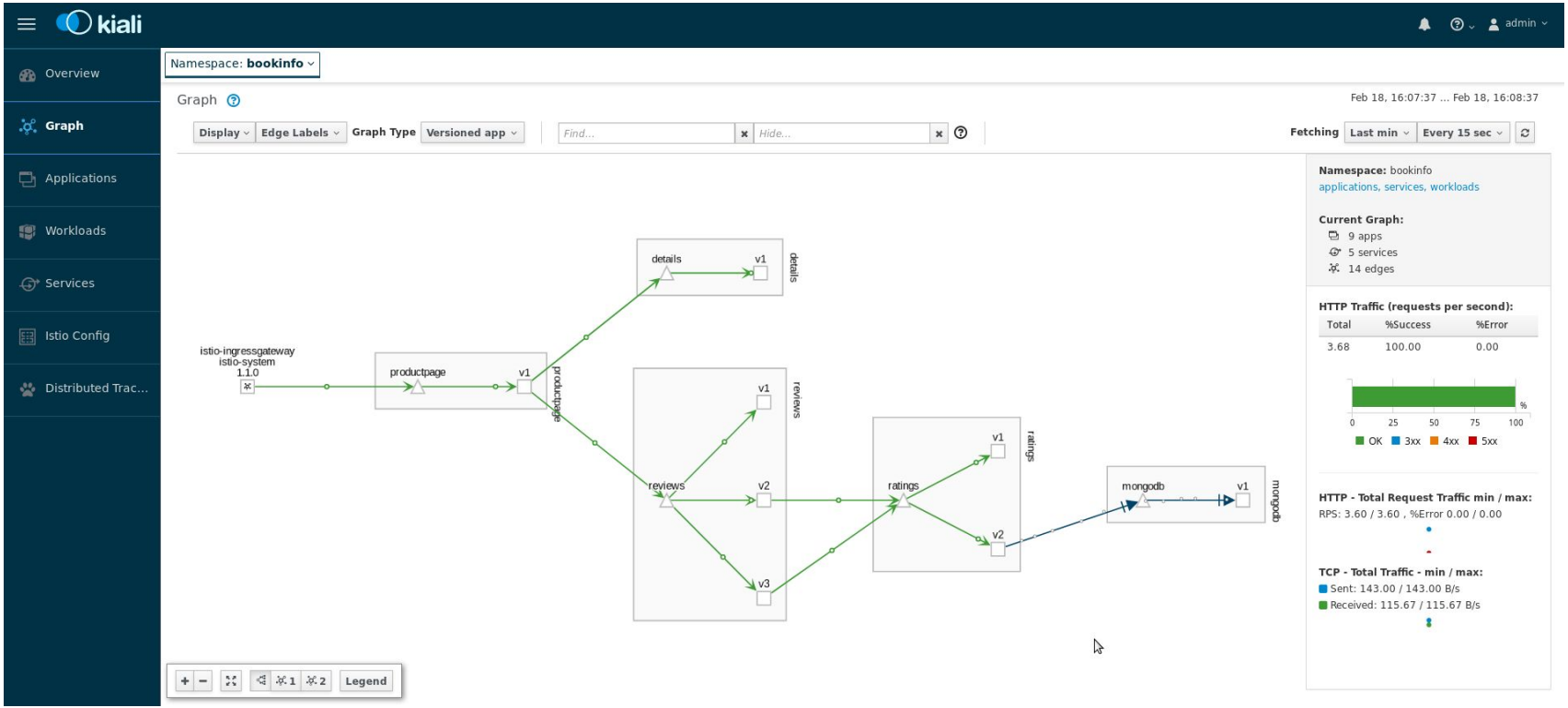
Key Features

- 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



Red Hat

Observability with Kiali



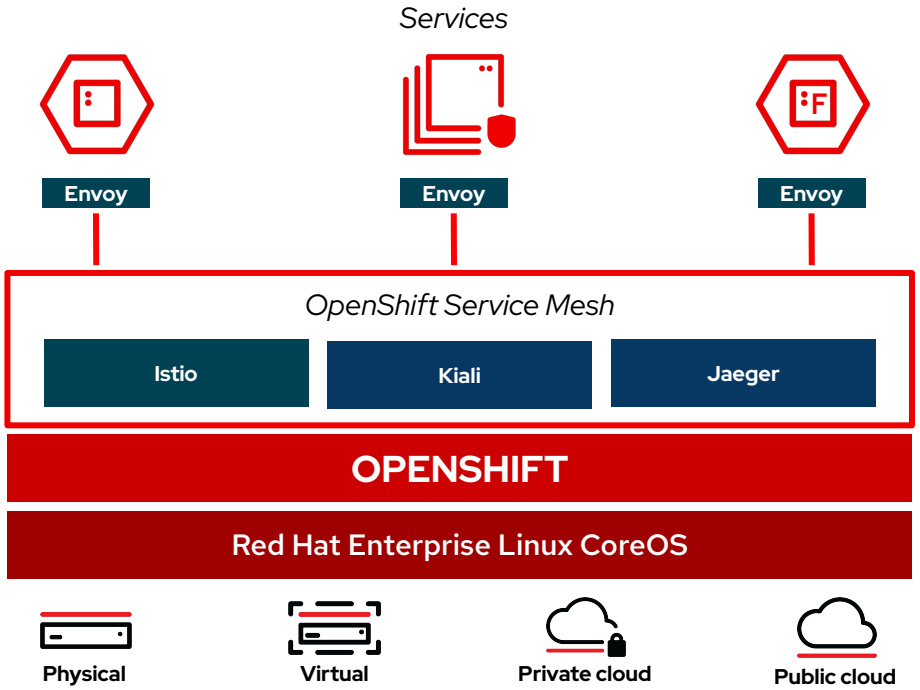
Red Hat

OpenShift Service Mesh

Connect, Secure, Control and Observe Services on OpenShift

- Connect services securely with zero-trust network policies.
- Automatically secure your services with managed authentication, authorization and encryption.
- Control traffic to safely manage deployments, A/B testing, chaos engineering and more.
- See what's happening with out of the box distributed tracing, metrics and logging.
- Manage OpenShift Service Mesh with the **Kiali** web console.

[Product briefing deck]



Product Manager: Jamie Longmuir and Mauricio "Maltron" Leal

* Eventing is currently in Technology Preview

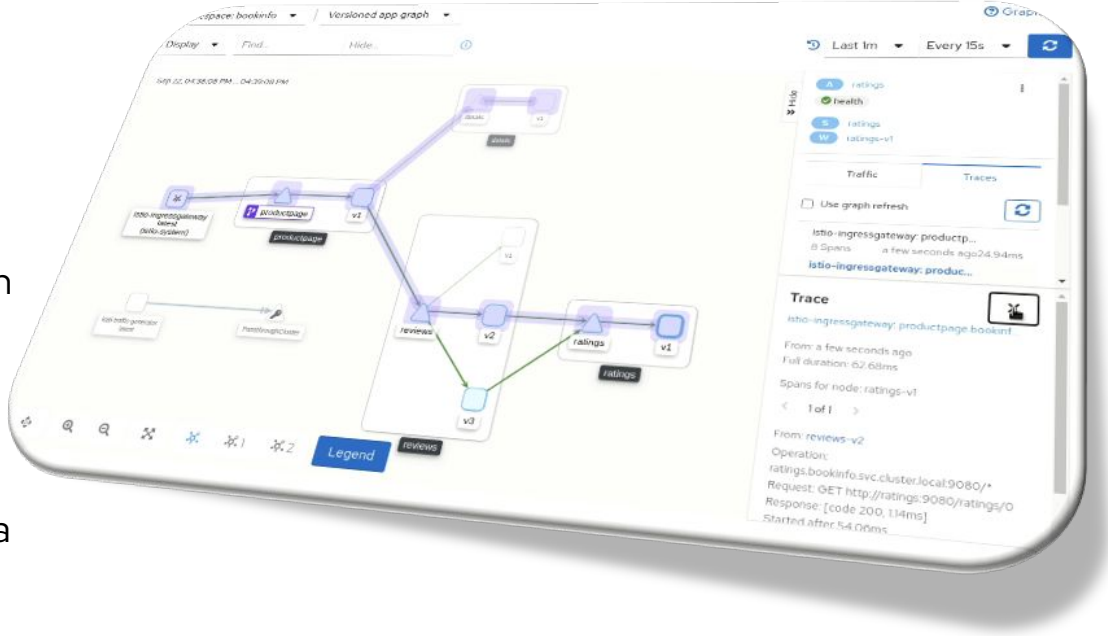
** Functions are currently a work in progress initiative

Red Hat

OpenShift Service Mesh 2.0

Key Features & Updates

- Version 2.0 to GA in November 2020
- Upgrades Istio to version 1.6
- Simplifies architecture based on a single Istio daemon (“Istiod”)
- Improves key and certificate rotation with Secret Discovery Service
- Improves metrics collection with Telemetry V2 architecture.
- Introduces WebAssembly extensions as a “Tech Preview” feature.



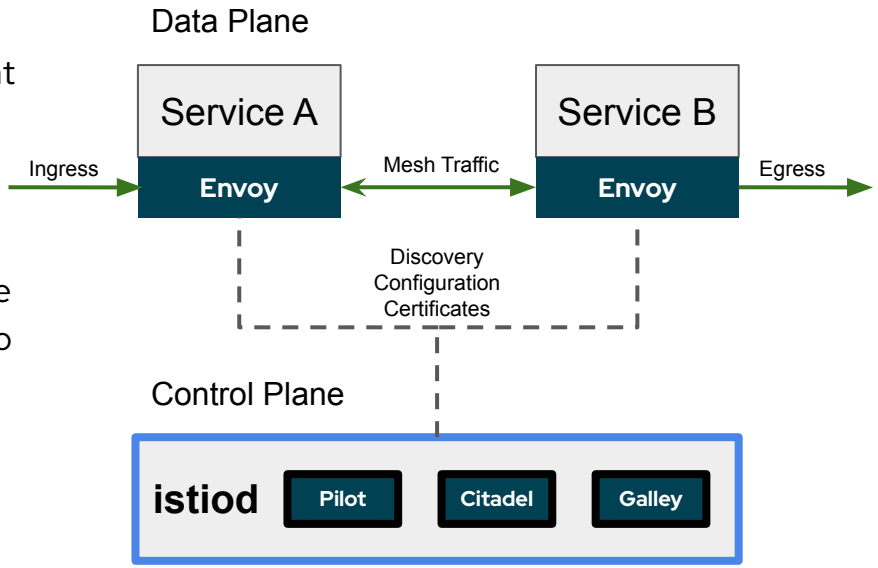
Product Manager: Jamie Longmuir and Mauricio "Maltron" Leal

Red Hat

OpenShift Service Mesh 2.0

Istio 1.6 - Architectural Changes

- Consolidates the Istio control plane components (Pilot, Galley, Citadel) into a single binary known as **istiod**.
  - Simplifies installation, upgrades and management of the Control Plane.
  - Reduces the Control Plane's resource usage, startup time and improves performance.
- Secret Discovery Service (**SDS**) provides a more secure and performant mechanism for delivering certificates to Envoy side car proxies.
  - Removes the use of Kubernetes Secrets.
  - Enables 3rd party cert manager integrations.
- New **Telemetry V2** architecture substantially reduces metrics collection latency.



Product Manager: Jamie Longmuir and Mauricio "Maltron" Leal

OpenShift Service Mesh 2.0

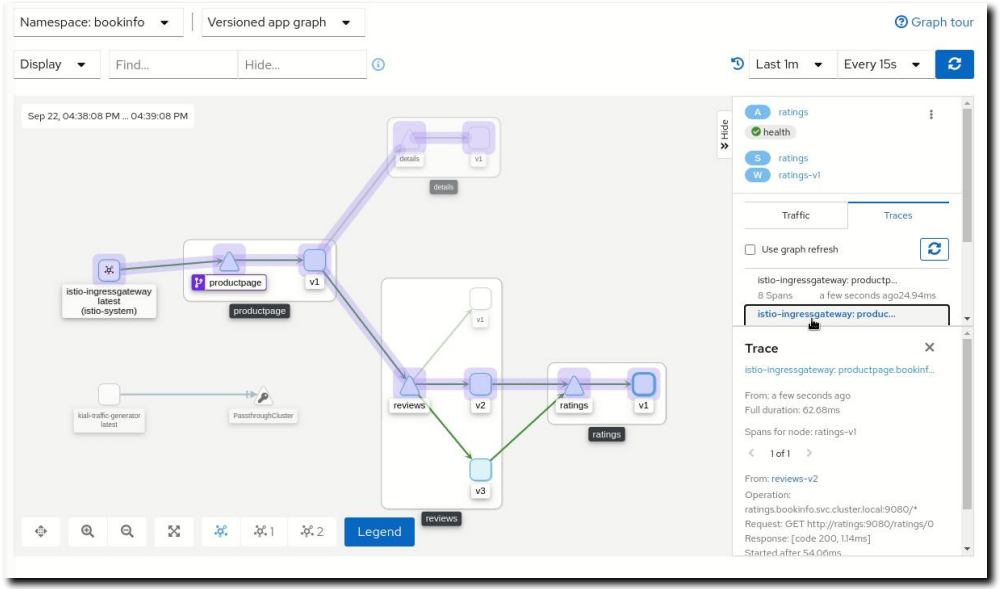
Secret Discovery Service

- Secret Discovery Service (**SDS**) is a more secure and performant mechanism for delivering secrets to Envoy side car proxies.
  - Removes the need to use Kubernetes Secrets, which have well known security risks.
  - Improves performance during certificate rotation, as proxies no longer require a restart.

OpenShift Service Mesh 2.0

User Experience Enhancements

- New **ServiceMeshControlPlane** resource (v2) to simplify configuration.
- **Kiali**:
  - Distributed traces are visualized and accessible in the service graph.
  - New wizards make it easier to configure timeouts, retries and fault injection scenarios.
- **Jaeger**:
  - Support for external ElasticSearch clusters.
  - OpenTelemetry collector in Tech Preview enabling vendor-neutral instrumentation.



Product Manager: Jamie Longmuir and Mauricio "Maltron" Leal

Security Ecosystem

Red Hat

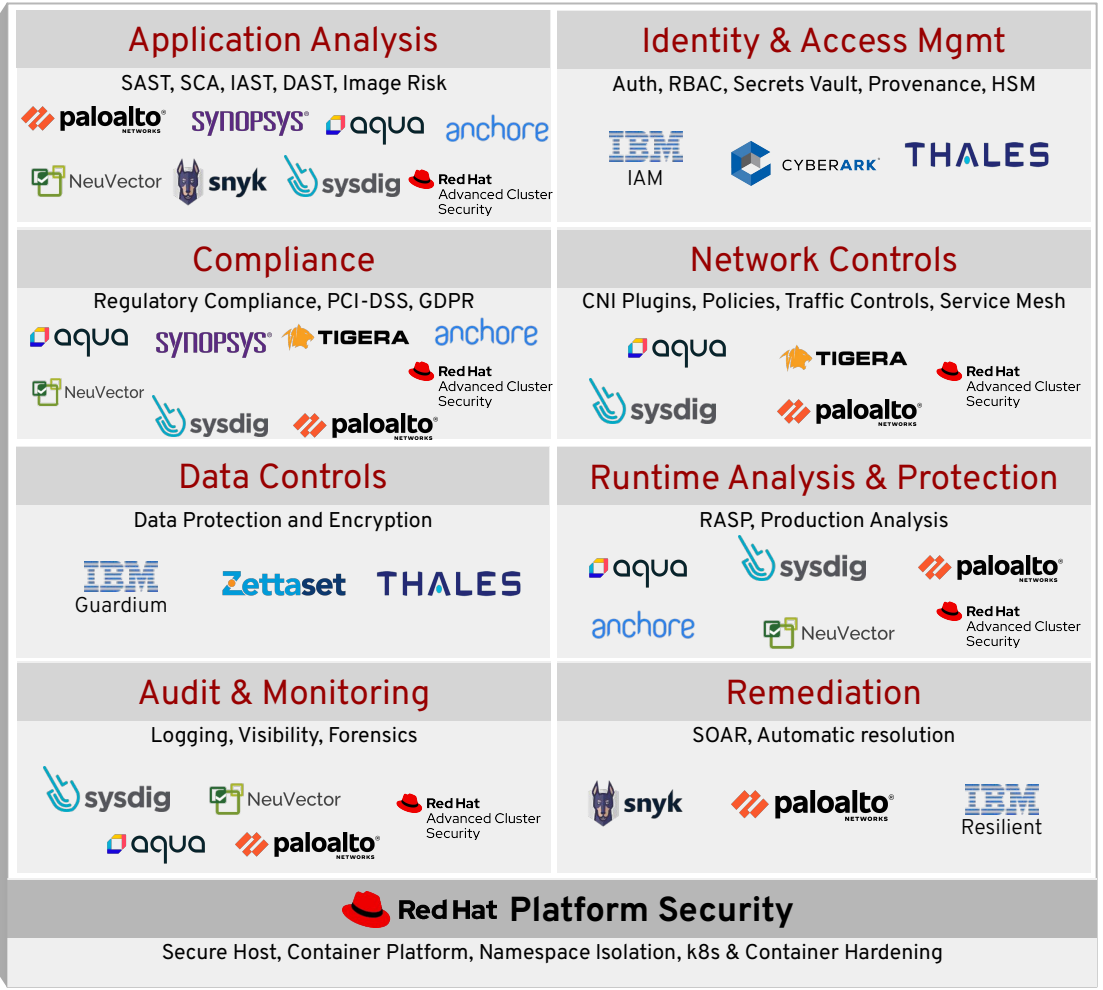
The Security Ecosystem

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)

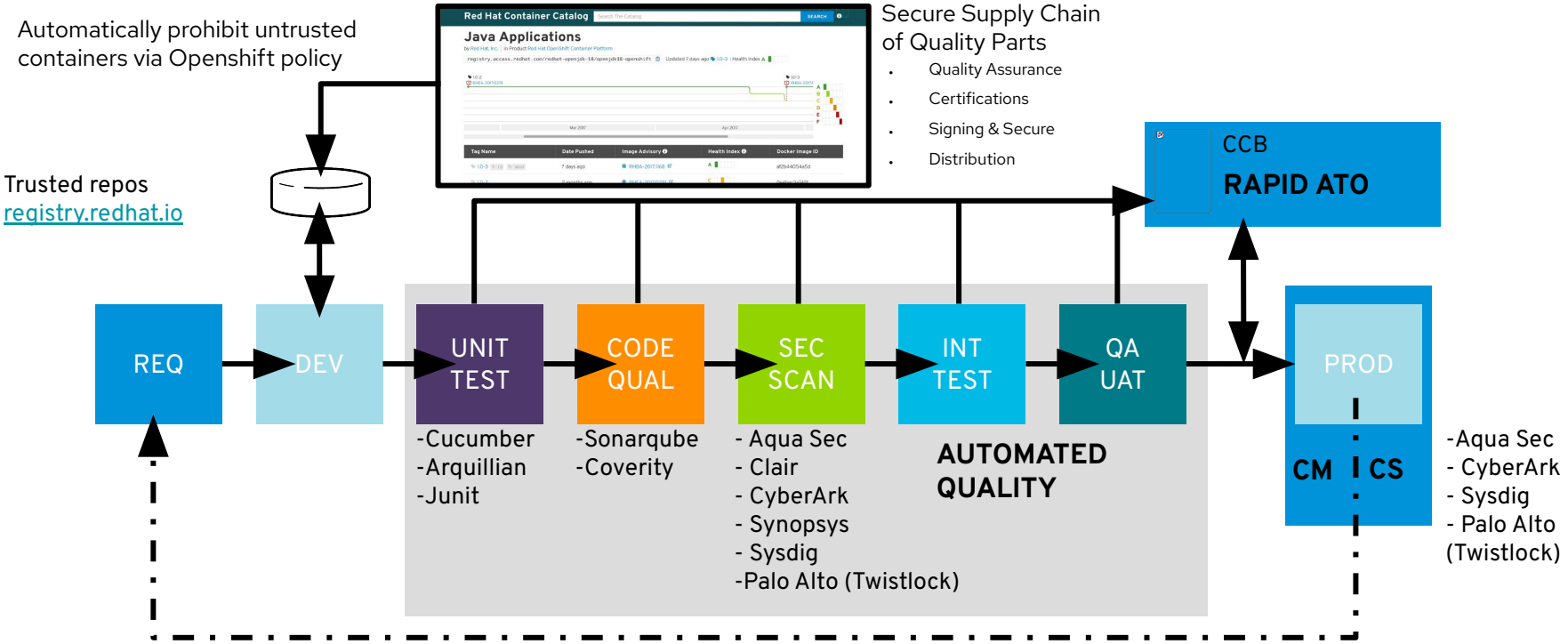
Security Partners by Use Case

Partners extend and enhance Red Hat functionality



Integrate Security in your CI/CD Pipeline

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

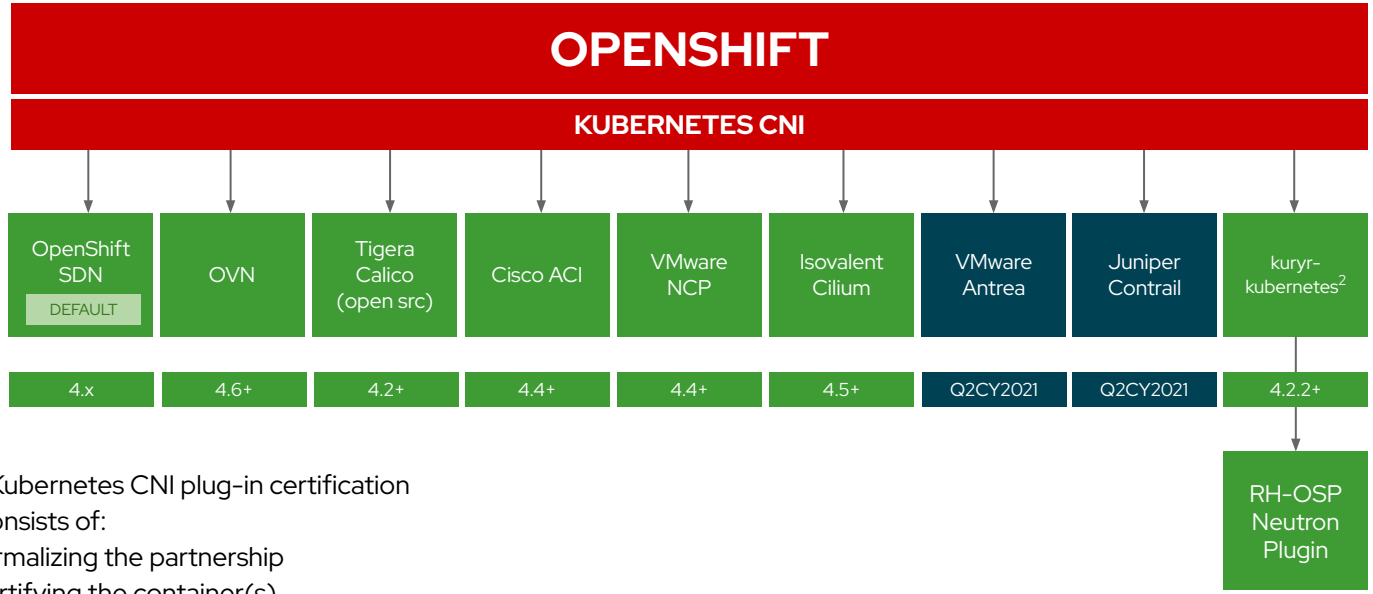


Red Hat Container Catalog: Java Applications
More about the Container Health Index

OPENSHIFT SOFTWARE FACTORY

Red Hat

OpenShift Networking Plug-ins



3rd-party Kubernetes CNI plug-in certification 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

Fully Supported | Tech Preview | Cert In-Progress

Resource & Cluster Capacity Management

- Manage compute resources, object counts, storage resources
  - Resource quotas per project
  - Resource quotas across multiple projects
- OpenShift Cluster Capacity Tool
  - Simulate a sequence of scheduling decisions to determine how many instances of an input pod can be scheduled on the cluster before it is exhausted of resources

Table 1. Compute resources managed by quota

Resource Name	Description
`cpu`	The sum of CPU requests across all pods in a non-terminal state cannot exceed this value. `cpu` and `requests.cpu` are the same value and can be used interchangeably.
`memory`	The sum of memory requests across all pods in a non-terminal state cannot exceed this value. `memory` and `requests.memory` are the same value and can be used interchangeably.
`ephemeral-storage`	The sum of local ephemeral storage requests across all pods in a non-terminal state cannot exceed this value. `ephemeral-storage` and `requests.ephemeral-storage` are the same value and can be used interchangeably. This resource is available only if you enabled the ephemeral storage technology preview. This feature is disabled by default.
`requests.cpu`	The sum of CPU requests across all pods in a non-terminal state cannot exceed this value. `cpu` and `requests.cpu` are the same value and can be used interchangeably.
`requests.memory`	The sum of memory requests across all pods in a non-terminal state cannot exceed this value. `memory` and `requests.memory` are the same value and can be used interchangeably.
`requests.ephemeral-storage`	The sum of ephemeral storage requests across all pods in a non-terminal state cannot exceed this value. `ephemeral-storage` and `requests.ephemeral-storage` are the same value and can be used interchangeably. This resource is available only if you enabled the ephemeral storage technology preview. This feature is disabled by default.
`limits.cpu`	The sum of CPU limits across all pods in a non-terminal state cannot exceed this value.
`limits.memory`	The sum of memory limits across all pods in a non-terminal state cannot exceed this value.

Tech Preview

Descheduler

Evict a running Pod so that the Pod can be **rescheduled** onto a more suitable node.

Situations when to use:

Nodes are underutilized or overutilized

Pod and node affinity requirements, such as taints or labels, have changed and the original scheduling decisions are no longer appropriate for certain nodes.

Node failure requires Pods to be moved.

New nodes are added to clusters.

https://docs.openshift.com/container-platform/4.4/nodes/scheduling/nodes-descheduler.html#nodes-descheduler

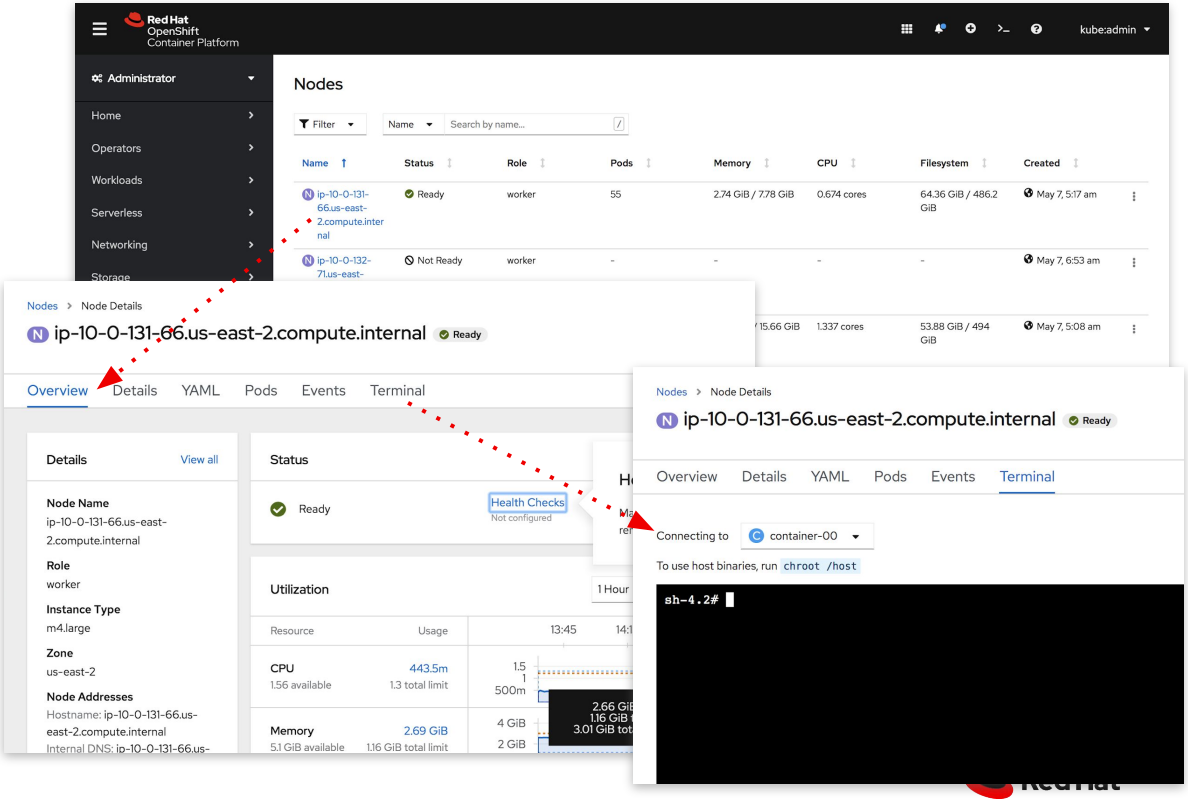
Install the descheduler via OperatorHub to the openshift-kube-descheduler-operator namespace and Create a descheduler instance

```
apiVersion: operator.openshift.io/v1beta1
kind: KubeDescheduler
metadata:
  name: cluster
  namespace: openshift-kube-descheduler-operator
spec:
  deschedulingIntervalSeconds: 3600
  strategies:
    - name: "LowNodeUtilization"
      params:
       - name: "cputhreshold"
         value: "10"
       - name: "memorythreshold"
         value: "20"
       - name: "podsthreshold"
         value: "30"
       - name: "memorytargetthreshold"
         value: "40"
       - name: "cputargetthreshold"
         value: "50"
       - name: "podstargetthreshold"
         value: "60"
       - name: "nodes"
         value: "3"
            - name: "RemoveDuplicates"
```

Nodes Resource views

At-a-glance views for your nodes right from the OpenShift Console.

- Key node data surfaced in the **List view**
- Offers a **new Overview** to provide **insights into critical data** back to you
  - Role/Type/Zone/Address
  - Status/Health Checks
  - Resource Utilizations
    - CPU/Memory/Filesystem
- Directly access to your node with a new **Terminal** view right in the console
  - Act as **root** on the node
  - Access Node Logs (journalctl)



Vertical Pod Autoscaler

Tech Preview

The **VPA** can determine the the **right size for pods** and **frees** the user from having to set **pod resource requests and limits.**

Three controllers:

Recommender - Recommends values for cpu and memory requests based on past consumption

Updater - Kills pods where VPA recommendations do not match the current settings so that they can be recreated by their controllers with the updated requests.

Admission Plugin - Sets the correct resource requests on new pods (due to Updater's activity).

The Vertical Pod Autoscaler Operator is managed by the Cluster Version Operator (CVO) and creates the openshift-vertical-pod-autoscaler namespace

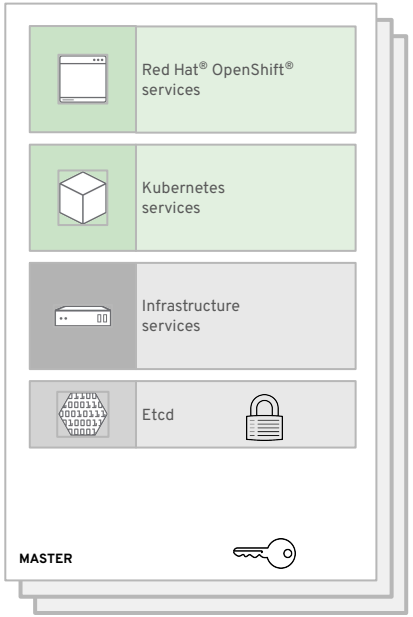
```
apiVersion: autoscaling.openshift.io/v1
kind: VerticalPodAutoscalerController
metadata:
 name: default
spec:
 safetyMarginFraction: 0.15
 podMinCPUMillicores: 25
 podMinMemoryMb: 250
```

Cipher Suites & Encrypting etcd

OpenShift 4 etcd Encryption

Encrypt secrets, config maps...

- Encryption of the etcd datastore is optional. Once enabled, encryption cannot be disabled.
- The aes-cbc cipher is used.
- Keys are created and automatically rotated by an operator and stored on the master node's file system.
- Keys are available as a secret via the kube API to a cluster admin.
- Assuming a healthy cluster: after enabling encryption, within a day, all relevant items in etcd are encrypted
- Backup: The etcd data store should be backed up separately from the file system with the key.
- Disaster recovery: a backup of both the encrypted etcd data and encryption keys must be available.



Red Hat

Ingress & API Cipher Suite Configuration

- Allow customers to meet policies requiring them to use specific cipher suites and/or to ensure that disallowed ciphers are not available.
- The TLSSecurityProfile defines the schema for a TLS security profile that will be used by Ingress and the API server.
- Type is one of Old, Intermediate, or Custom. The Modern profile is currently not supported because it is not yet well adopted by common software libraries.

```
// custom is a user-defined TLS security profile. Be extremely careful using a custom
// profile as invalid configurations can be catastrophic. An example custom profile
// looks like this:
//
//   ciphers:
//     - ECDHE-ECDSA-CHACHA20-POLY1305
//     - ECDHE-RSA-CHACHA20-POLY1305
//     - ECDHE-RSA-AES128-GCM-SHA256
//     - ECDHE-ECDSA-AES128-GCM-SHA256
//   minTLSVersion: TLSv1.1
//
// +optional
// +nullable
Custom *CustomTLSProfile `json:"custom,omitempty"`
```

OpenShift Container Platform

Control	Protect	Detect & Respond
Trusted Content	Kubernetes Platform Lifecycle	Container Isolation
Container Registry	Identity and Access Management	Network Isolation
Build Management	Platform Data	Application Access & Data
CI/CD Pipeline	Deployment Policies	Observability
BUILD	DEPLOY	RUN

DevSecOps

A layered approach to container and Kubernetes security

Red Hat Advanced Cluster Management

Control

Application Lifecycle and Locality

- Vulnerability analysis
- App config analysis
- APIs for CI/CD integrations
- Trusted content
- Container registry
- Build management
- CI/CD pipeline

BUILD

Protect

Fleet Configuration Management

- Policy admission controller
- Compliance assessments
- Risk profiling
- Kubernetes platform lifecycle
- Identity and access management
- Platform data
- Deployment policies

DEPLOY

Detect & Respond

Fleet Observability & Alerts

- Runtime behavioral analysis
- Auto-suggest network policies
- Threat detection / incident response
- Container isolation
- Network isolation
- Application access and data
- Observability

RUN

DevSecOps