OPENSHIFT 4 & PCI-DSS

Kirsten Newcomer

Director, Cloud and DevSecOps Strategy

Red Hat, Cloud Platforms

December 2020

What makes an effective hybrid cloud platform?

BROAD ECOSYSTEM

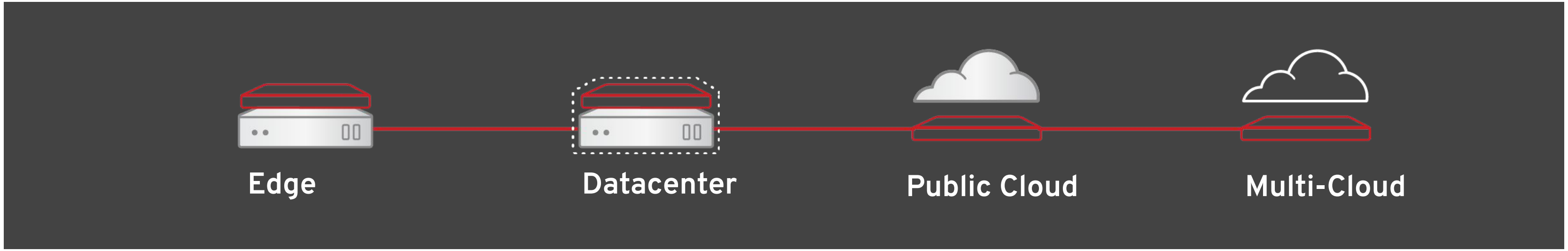
BROADEST APPLICATION SUPPORT

DEVELOPER EXPERIENCE & ON-DEMAND

AUTOMATED OPERATIONS

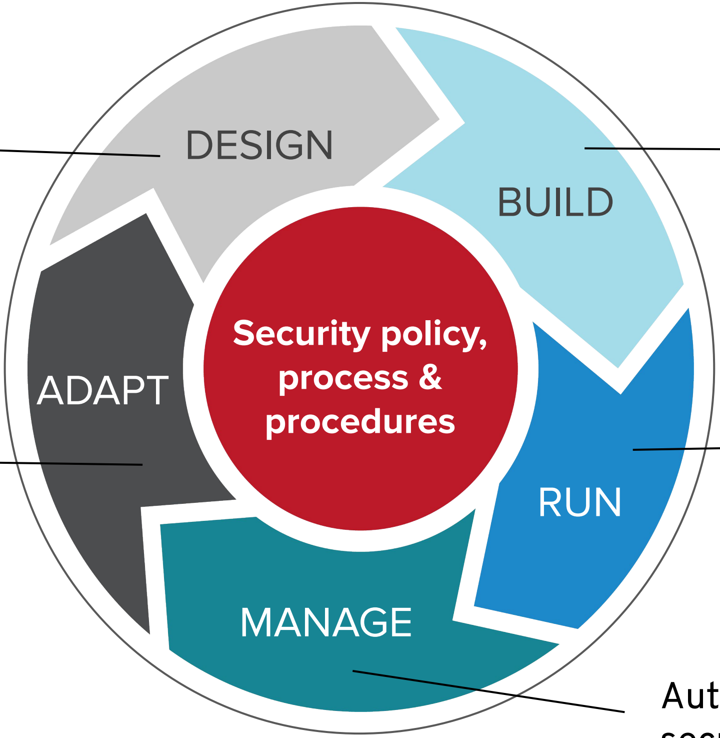
STANDARDS, PORTABILITY & INTEROPERABILITY

SECURITY & COMPLIANCE



Security must be continuous

And integrated throughout the IT lifecycle



Identify security requirements & governance models

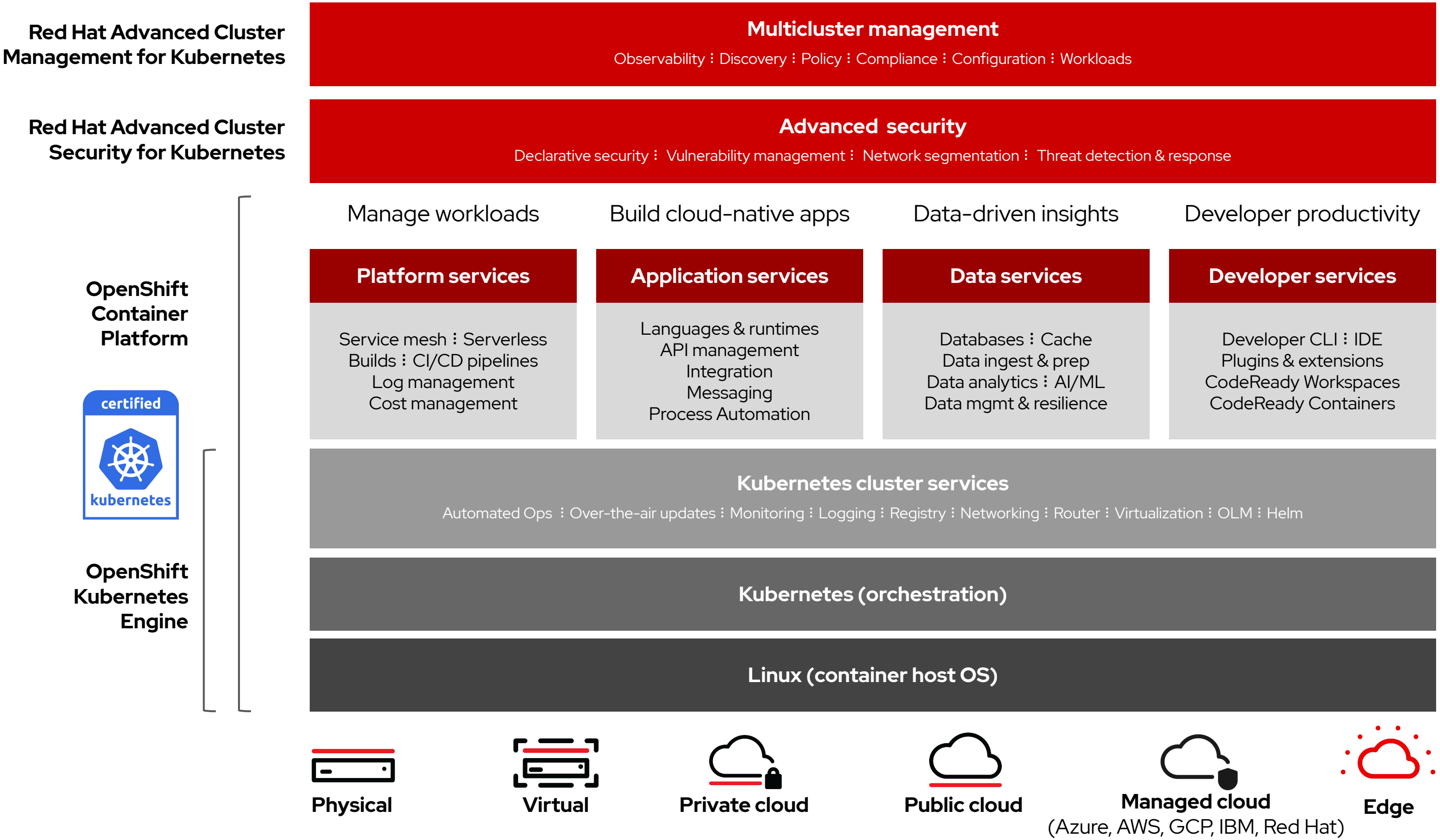
Built-in from the start; not bolted-on

Deploy to trusted platforms with enhanced security capabilities

Automate systems for security & compliance

Revise, update, remediate as the landscape changes

Enterprise Kubernetes from Red Hat



Red Hat Advanced Cluster Management for Kubernetes

Multicluster management

Observability ⁝ Discovery ⁝ Policy ⁝ Compliance ⁝ Configuration ⁝ Workloads

Red Hat Advanced Cluster Security for Kubernetes

Advanced security

Declarative security ⁝ Vulnerability management ⁝ Network segmentation ⁝ Threat detection & response

OpenShift Container Platform

Manage workloads	Build cloud-native apps	Data-driven insights	Developer productivity
Platform services	**Application services**	**Data services**	**Developer services**
Service mesh ⁝ Serverless Builds ⁝ CI/CD pipelines Log management Cost management	Languages & runtimes API management Integration Messaging Process Automation	Databases ⁝ Cache Data ingest & prep Data analytics ⁝ AI/ML Data mgmt & resilience	Developer CLI ⁝ IDE Plugins & extensions CodeReady Workspaces CodeReady Containers

OpenShift Kubernetes Engine

Kubernetes cluster services

Automated Ops ⁝ Over-the-air updates ⁝ Monitoring ⁝ Logging ⁝ Registry ⁝ Networking ⁝ Router ⁝ Virtualization ⁝ OLM ⁝ Helm

Kubernetes (orchestration)

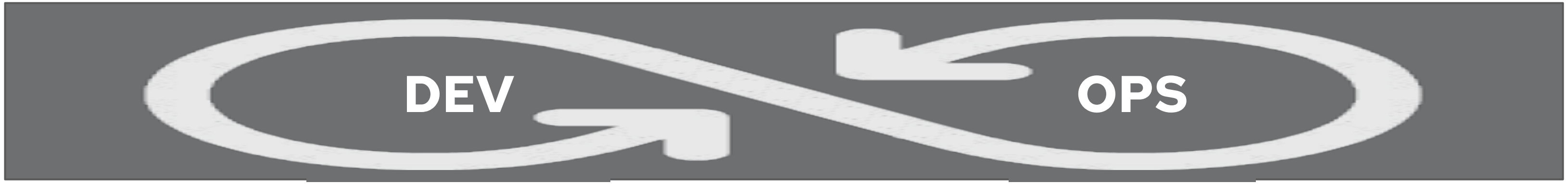
Linux (container host OS)

Physical **Virtual** **Private cloud** **Public cloud** **Managed cloud** (Azure, AWS, GCP, IBM, Red Hat) **Edge**

Red Hat

Red Hat delivers continuous security for containers and Kubernetes

DETECT	PROTECT	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



A layered approach to container and Kubernetes security

Hardening, applicability guides, certifications

OpenShift 4

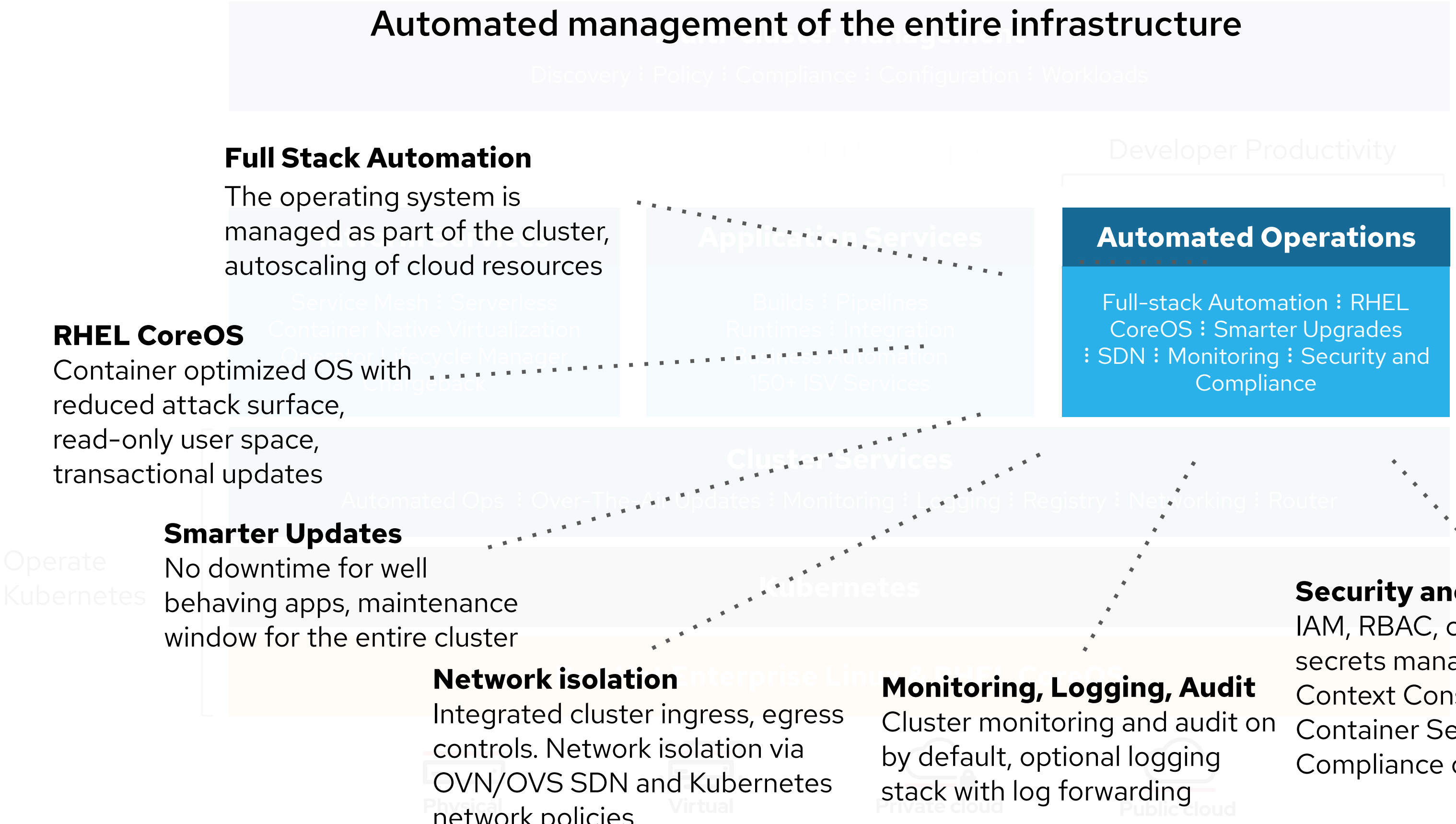
- Available now
  - HIPAA
  - ISO 27001 (ask RH for a copy)
  - FISMA
  - The OpenShift Security Guide
  - OpenShift 4 Hardening Guide (ask RH for a copy)
  - PCI-DSS
- Target Q2 CY 2021
  - CIS OpenShift Benchmark
  - HITRUST

Managed Services certifications

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

OpenShift Container Platform

Automated management of the entire infrastructure



Multicluster Management
Discovery ⁝ Policy ⁝ Compliance ⁝ Configuration ⁝ Workloads
Platform Services
Application Services
Developer Productivity
Service Mesh ⁝ Serverless
Container Native Virtualization
Operator Lifecycle Manager
Chargeback
Builds ⁝ Pipelines
Runtimes ⁝ Integration
Business Automation
150+ ISV Services
Cluster Services
Automated Ops ⁝ Over-The-Air Updates ⁝ Monitoring ⁝ Logging ⁝ Registry ⁝ Networking ⁝ Router
Kubernetes
Enterprise Linux & RHEL CoreOS
Operate Kubernetes
Physical
Virtual
Private cloud
Public cloud

Automated Operations

Full-stack Automation ⁝ RHEL CoreOS ⁝ Smarter Upgrades ⁝ SDN ⁝ Monitoring ⁝ Security and Compliance

Full Stack Automation
The operating system is managed as part of the cluster, autoscaling of cloud resources

RHEL CoreOS
Container optimized OS with reduced attack surface, read-only user space, transactional updates

Smarter Updates
No downtime for well behaving apps, maintenance window for the entire cluster

Network isolation
Integrated cluster ingress, egress controls. Network isolation via OVN/OVS SDN and Kubernetes network policies

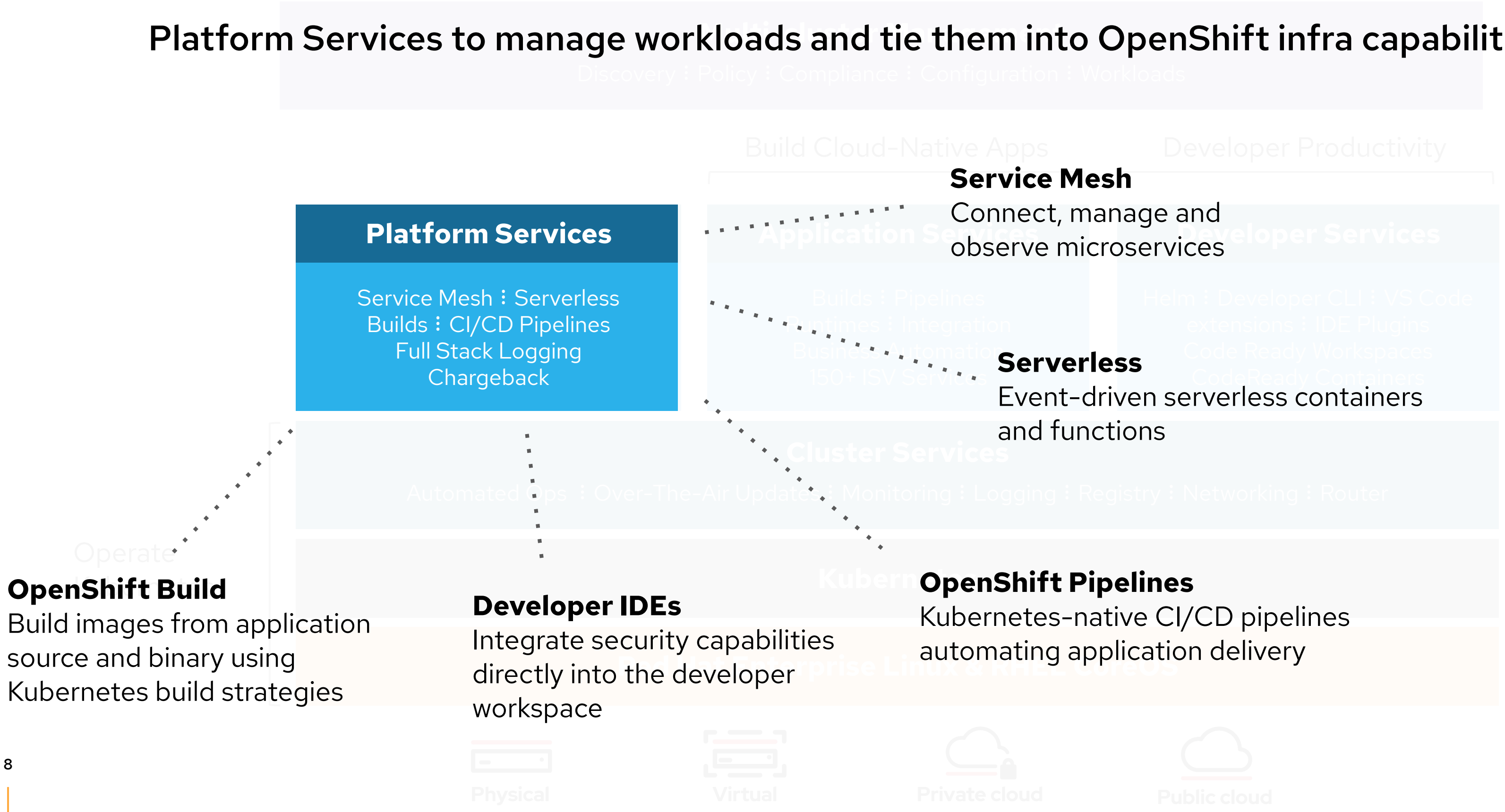
Monitoring, Logging, Audit
Cluster monitoring and audit on by default, optional logging stack with log forwarding

Security and Compliance
IAM, RBAC, certificate and secrets management, Security Context Constraints. Container Security operator, Compliance operator

Red Hat

OpenShift Platform Services

Platform Services to manage workloads and tie them into OpenShift infra capabilities



Service Mesh
Connect, manage and observe microservices

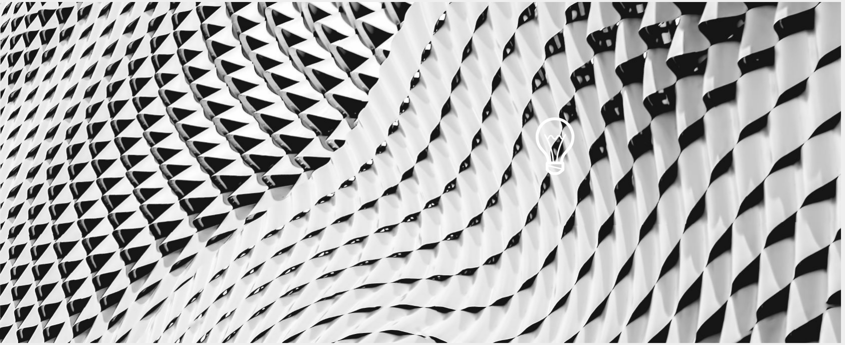
Serverless
Event-driven serverless containers and functions

OpenShift Build
Build images from application source and binary using Kubernetes build strategies

Developer IDEs
Integrate security capabilities directly into the developer workspace

OpenShift Pipelines
Kubernetes-native CI/CD pipelines automating application delivery

PCI-DSS RECOMMENDATIONS



Coalfire Product Applicability Guide & Reference Architecture

- Red Hat contracted with Coalfire to provide a PCI-DSS technical controls product applicability guide (PCI-DSS 3.2) and reference architecture (PCI-DSS 3.2.1) for OpenShift*
- Technical requirements 1, 2, 5, 6, 7, 8, 10, 11 are applicable, discussed

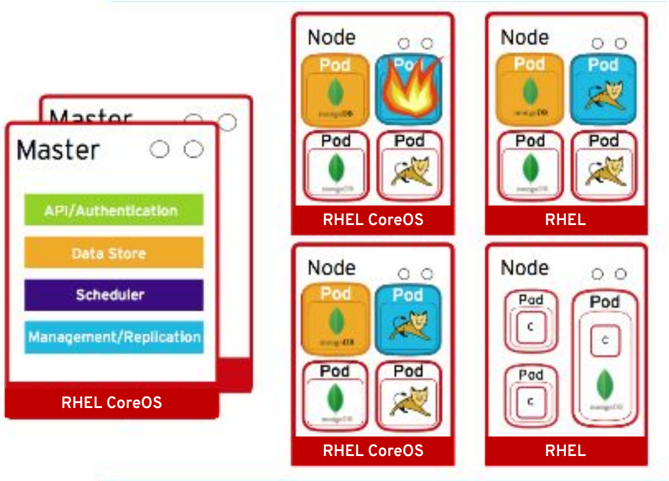
*Coalfire is working on a new version of this guide for OpenShift 4

Coalfire conclusion

“OpenShift hosted on Red Hat Enterprise Linux, as reviewed by Coalfire, can be effective in providing support for the outlined objectives and requirements of PCI DSS v3.2.1. Through proper implementation and integration into the organization’s overall technical infrastructure and information management systems, OpenShift may be useable in a PCI DSS v3.2.1 controlled environment.”

Securing the container platform

- Configuration and lifecycle management
- Host & runtime security
- Identity and Access Management
  - Project namespaces
  - Role Based Access Controls
- Data at rest, data in transit
  - Ingress & egress controls
  - Encryption
- Logging, Monitoring, Metrics
- Audit and Compliance



Automated Configuration and Lifecycle Management
Dramatically simplified for the Hybrid Cloud

Machines

Machines are complex for ops

Make machines easy
(like containers)

Configuration

Config change is risky

Make config management
and config change
easy and safe

Lifecycle

Software lifecycle is hard

Automate software
lifecycle on Kube

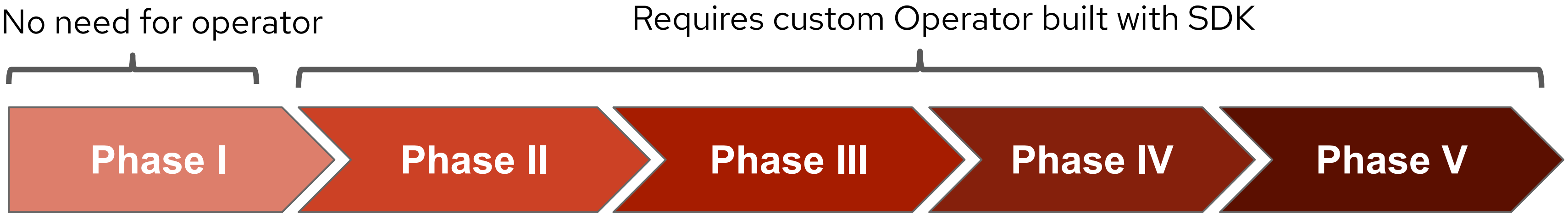
Automated Container Operations

FULLY AUTOMATED DAY-1 AND DAY-2 OPERATIONS

AUTOMATED OPERATIONS

INSTALL	DEPLOY	HARDEN	OPERATE
Infra provisioning	Full-stack deployment	Secure defaults	Multicluster aware
Embedded OS	On-premises and cloud	Network isolation	Monitoring and alerts
	Unified experience	Audit and logs	Full-stack patch & upgrade
		Signing and policies	Zero-downtime upgrades
			Vulnerability scanning

The Value Of Kubernetes Operators



No need for operator

Requires custom Operator built with SDK

Phase I

Phase II

Phase III

Phase IV

Phase V

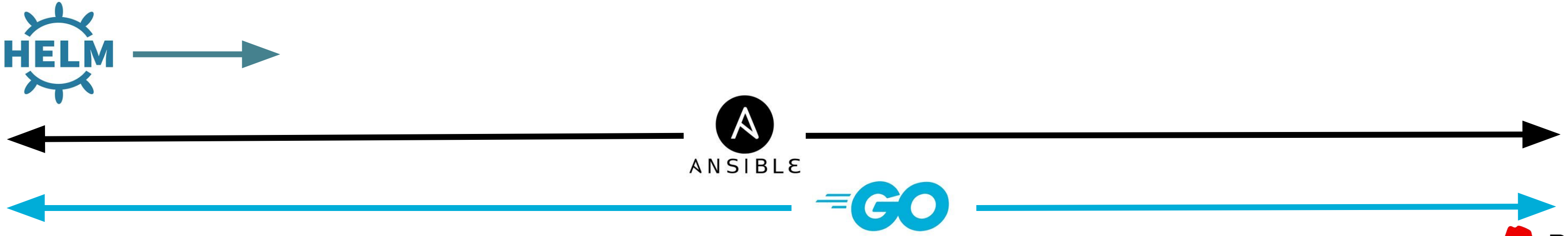
Installation
Automated application provisioning and configuration management

Upgrades
Patch and minor version upgrades supported

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

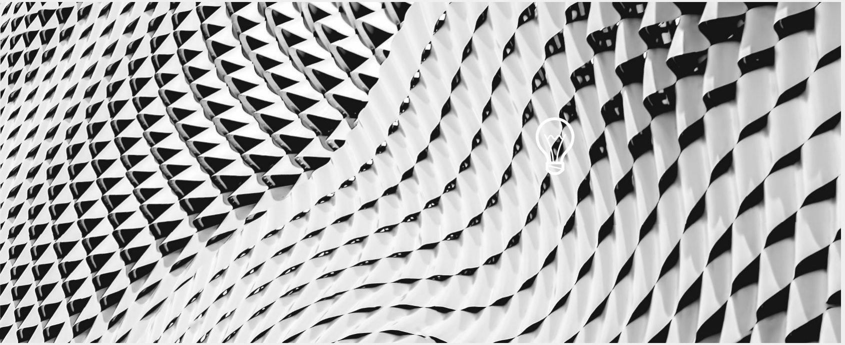
Deep Insights
Metrics, alerts, log processing and workload analysis

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



REQUIREMENT 1

Install & maintain a firewall configuration



Requirement 1: Install & maintain a firewall configuration to protect cardholder data

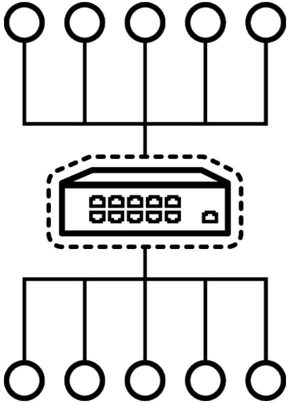
“Requirement 1 is primarily concerned with traditional edge protections between the Internet or “untrusted networks” and internal networks. It is recommended that the OpenShift environment be placed in an internal controlled network that is protected with traditional edge protections provided by third-party solutions. ***As such, Coalfire determined that most, if not all, of these requirements were not pertinent to OpenShift’s capabilities. However, assessors often look at implementation of firewalls and routers on internal networks used to isolate or segment workloads as a method of reducing assessment scope. With this in mind, many of the requirements may apply to the internal network elements that perform segmentation, including the SDN elements provided by OpenShift***."

Requirement 1 Applicability

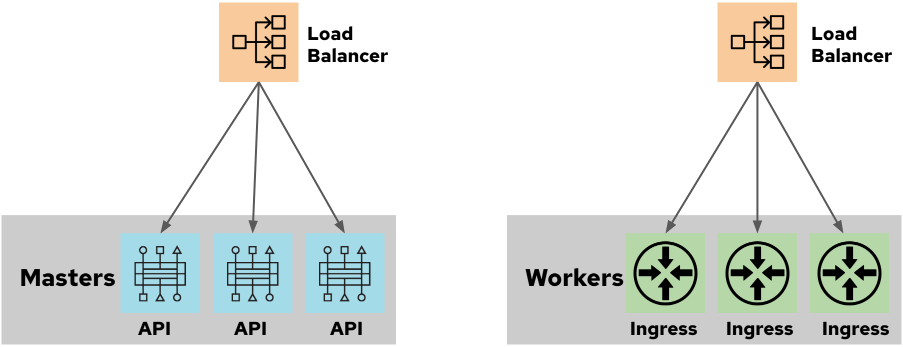
- Protect with “traditional edge protections provided by third party solutions.”
- Use the available SDN network policies to provide micro-segmentation and isolation of workloads
- Use ingress and network policy objects to restrict inbound traffic
- OpenShift provides the ability to separate workloads onto different servers (nodes)
- Similarly, infrastructure pods (ingress and egress router) can be hosted on separate nodes from the master
- Use egress to restrict outbound traffic

OpenShift networking

- Built-in internal DNS to reach services by name
- Software Defined Networking (SDN) for a unified cluster network to enable pod-to-pod communication
- OpenShift follows the Kubernetes Container Networking Interface (CNI) plug-in model
- Isolate applications from other applications within a cluster
- Isolate environments (Dev / Test / Prod) from other environments within a cluster



External Access to Cluster Resources

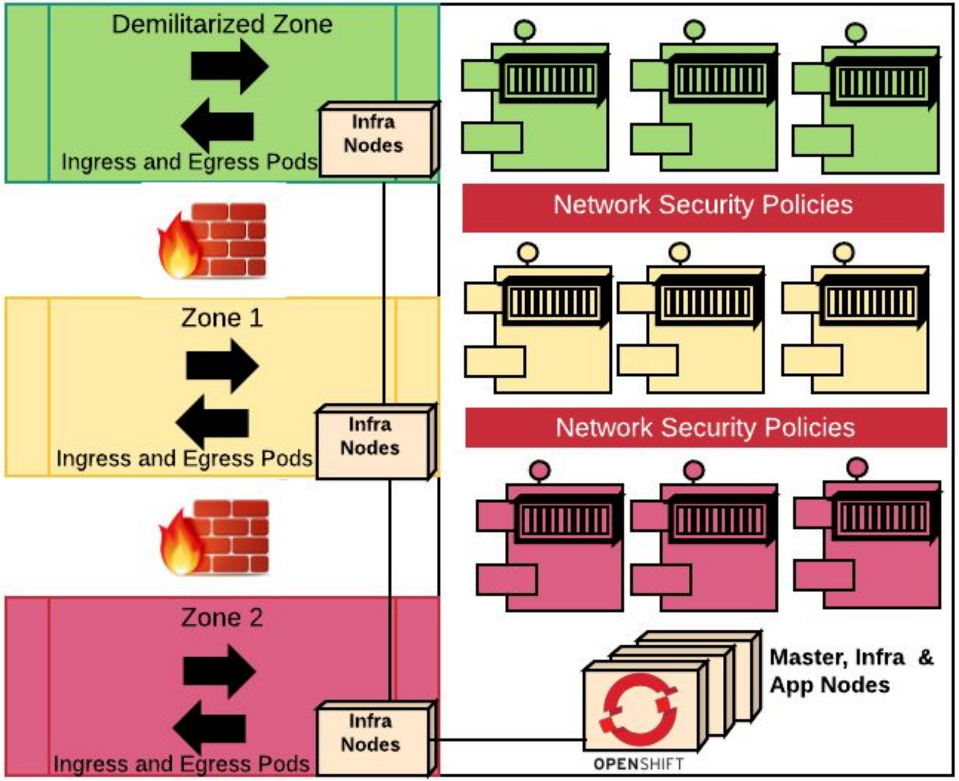


Ingress Traffic

- 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

OpenShift cluster with multiple zones

Using multiple ingress controllers, network policies, multiple egress pods



Application pods run on one OpenShift Cluster.

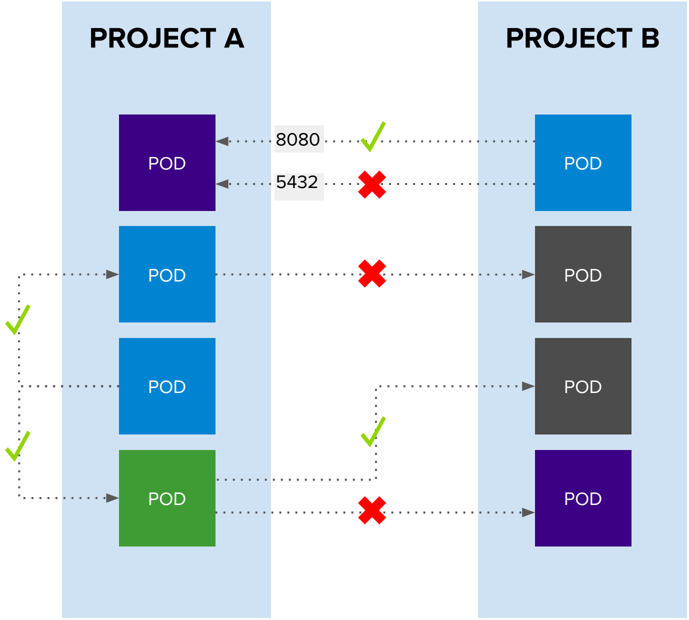
Microsegmented with Network Security policies.

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.

OpenShift SDN

Network policy enabled by default



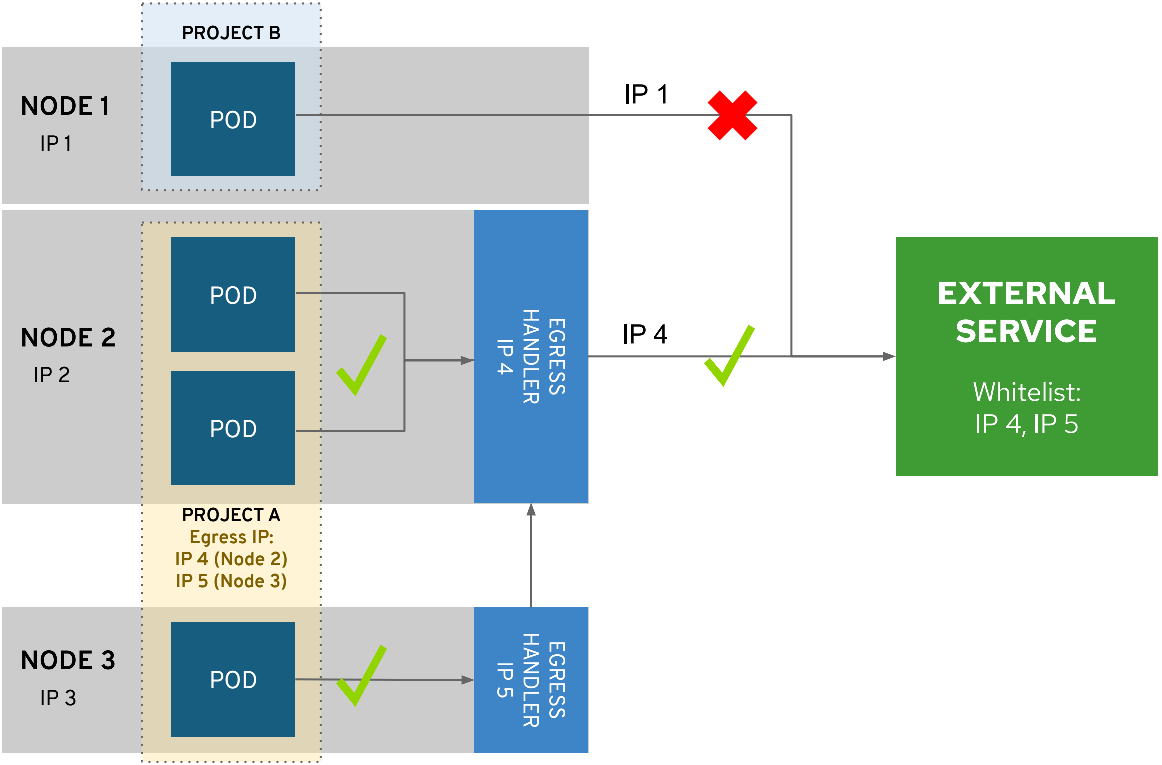
Example Policies

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

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

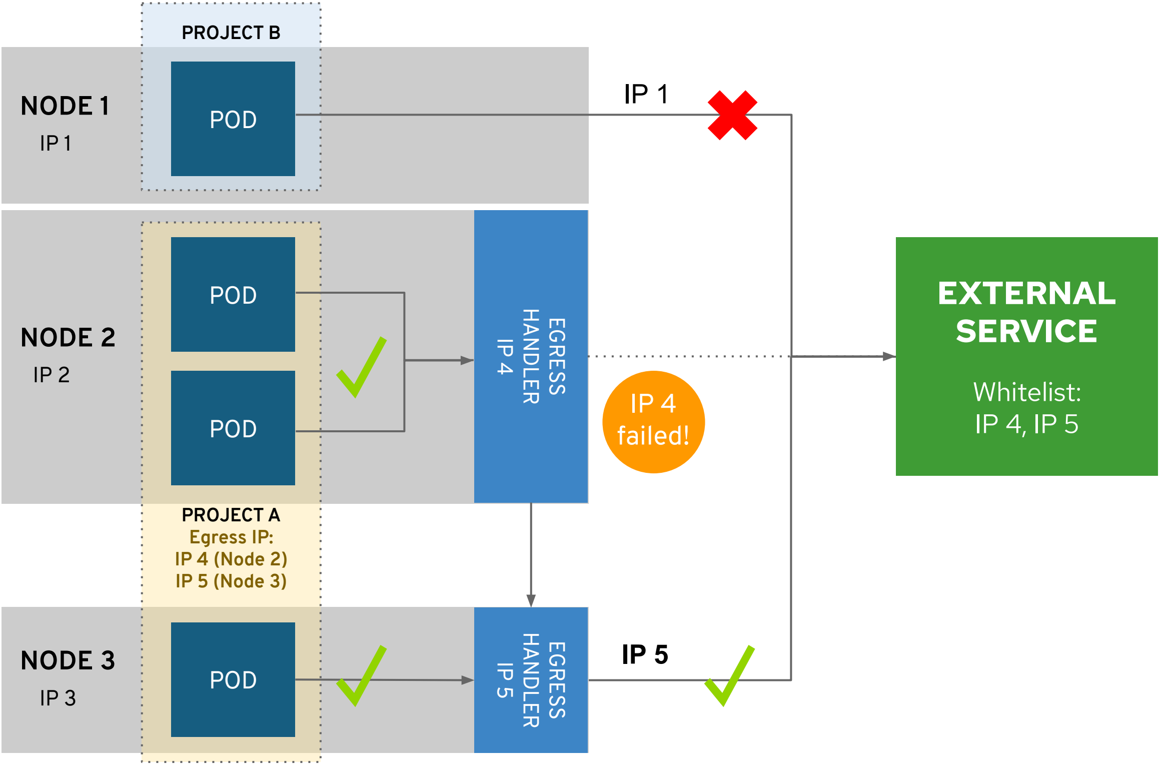
Controlling Egress Traffic

Egress IP high availability (multiple IPs)



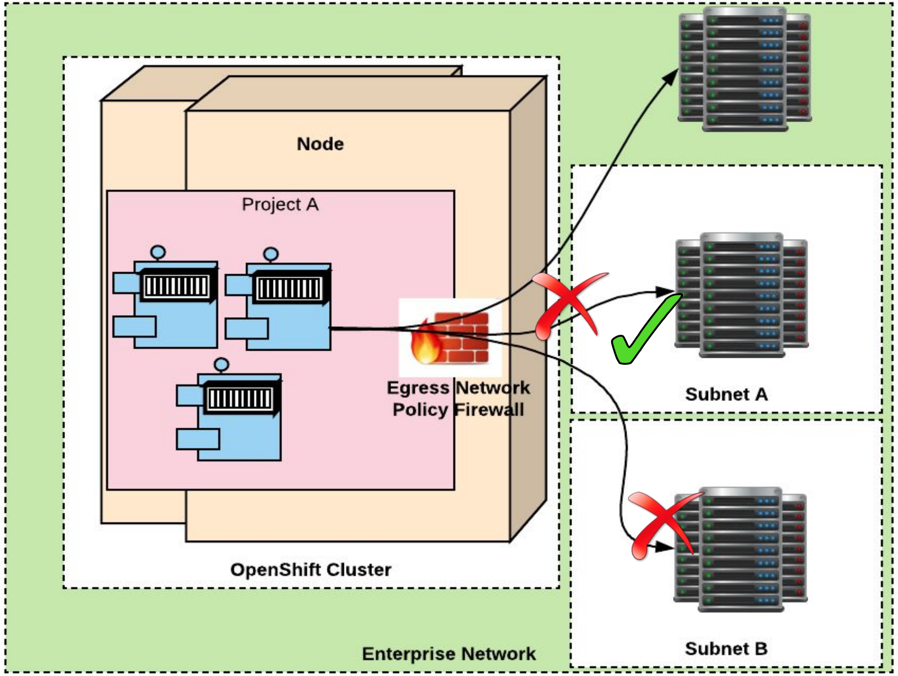
Controlling Egress Traffic

Egress IP high availability (multiple IPs)



Egress firewall to limit access

to external addresses accessed by some or all pods from within the cluster



Examples:

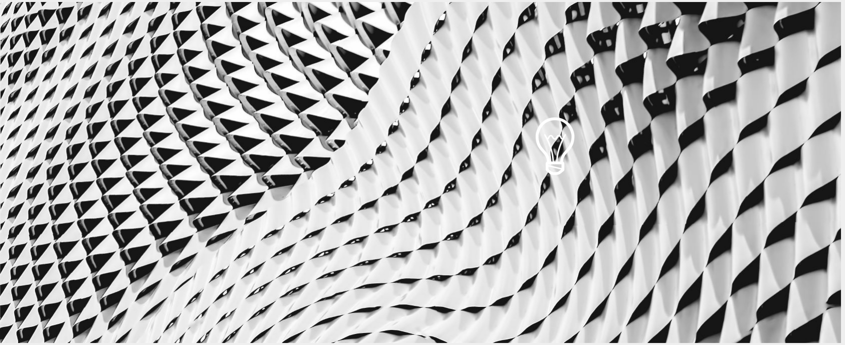
A pod can talk to hosts (outside OpenShift cluster) but cannot connect to public internet

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

A pod cannot reach specific subnets/hosts

REQUIREMENT 2

Vendor Standards and Configs



Requirement 2
Vendor defaults & configuration standards

2.1 Always change vendor-supplied defaults and remove or disable unnecessary default accounts

2.2 Develop config standards for all components

- 2.2.2 Enable only necessary services, protocols, daemons, etc., as required for the function of the system
- 2.2.3 Implement additional security features as required. For example, encryption
- 2.2.4 Configure system security parameters to prevent misuse
- 2.2.5 Remove all unnecessary functionality

2.3 Encrypt all non-console administrative access

Requirement 2 Applicability

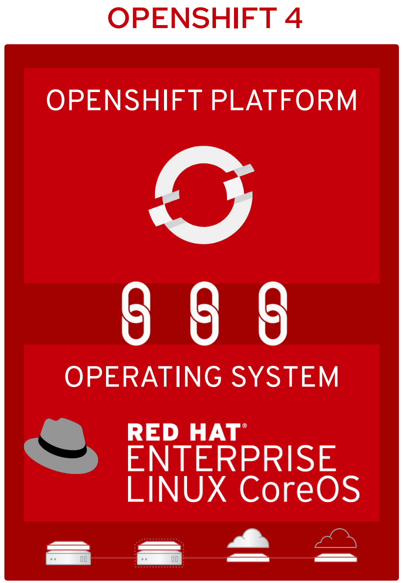
- No vendor provided default passwords are in use
- Configure an external identity provider, create your own cluster admin, remove kubeadmin user
- By default, RHEL CoreOS is a container-optimized OS and includes only the components needed to run OpenShift 4
- Encrypt RHEL CoreOS volumes
- Encrypt the etcd datastore
- By default, all communication between the control plane and the data plane is encrypted
- Install the Compliance operator (requires 4.6)

Container host vision

An Ideal Container Host would be	RHEL CoreOS
Minimal	Only what’s needed to run containers
Secure	Read-only & locked down
Immutable	Immutable image-based deployments & updates
Always up-to-date	OS updates are automated and transparent
Updates never break my apps	Isolates all applications as containers
Updates never break my cluster	OS components are compatible with the cluster
Supported on my infra of choice	Inherits majority of the RHEL ecosystem
Simple to configure	Installer generated configuration
Effortless to manage	Managed by Kubernetes Operators

Red Hat Enterprise Linux CoreOS

The Immutable Container Optimized Operating System



Role in OpenShift Ecosystem

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

Managed by the OpenShift Cluster

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

cri-o

A lightweight, OCI-compliant container runtime

Optimized for Kubernetes

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

Improve Security and Performance at scale

CRI - the Container Runtime Interface
OpenShift 4 defaults to CRI-O
Red Hat contributes CRI-O to the Cloud Native Computing Foundation

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.

RHEL CoreOS management

- **Regular management** of the underlying RHCOS cluster nodes is designed to be performed via the OpenShift API itself.
- **The only users** that exist on an RHCOS OpenShift node are *root* and *core*.
  - A user named core is created, with your ssh key assigned to that user. This allows you to log in to the cluster with that user name and your credentials.
  - The core user has permission to run privileged commands.
  - Adding additional users at the node level is highly discouraged.
  - **Updates are managed through the OpenShift MachineConfigOperator**
  - OS upgrades are delivered as an atomic unit.
  - The new OS deployment is staged during upgrades and goes into effect on the next reboot.
  - RHCOS upgrades in OpenShift Container Platform are performed during cluster updates.

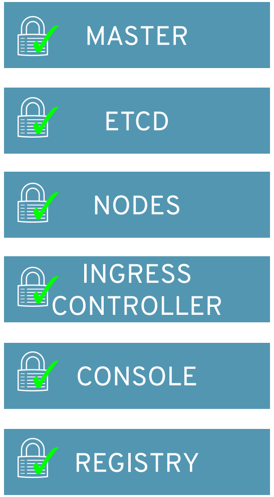
OpenShift Cluster Management

- OpenShift Container Platform **creates the kubeadmin user** after the installation process completes.
- The **kubeadmin user has the cluster-admin role automatically applied** and is treated as the root user for the cluster. The password is dynamically generated and unique to your OpenShift Container Platform environment. After installation completes the password is provided in the installation program’s output.
- **After you define an identity provider and create a new cluster-admin user, you can remove the kubeadmin to improve cluster security**.
- Cluster configuration changes are managed through cluster operators.

For more information see:
https://docs.openshift.com/container-platform/4.5/authentication/understanding-identity-provider.html

Encrypted control plane communication

- 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]()



MASTER

ETCD

NODES

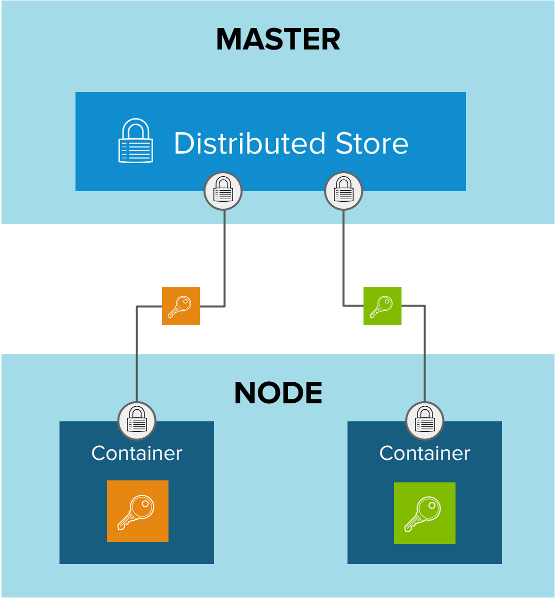
INGRESS CONTROLLER

CONSOLE

REGISTRY

Encrypt secrets in transit and at rest

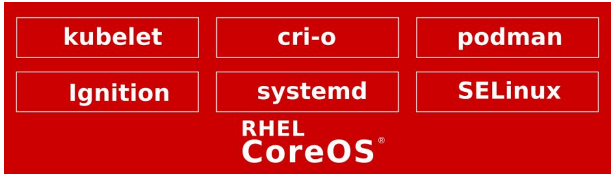
- Secure mechanism for holding sensitive data, such as
  - Passwords and credentials
  - SSH Keys
  - Certificates
- Secrets are made available as
  - Environment variables
  - Volume mounts
  - Interaction with external systems (e.g. vaults)
- Encrypted in transit and at rest
  - Encrypt the etcd datastore
  - Encrypt RHCOS volumes
- Never rest on the nodes



Volume Encryption

Network Bound Disk Encryption

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



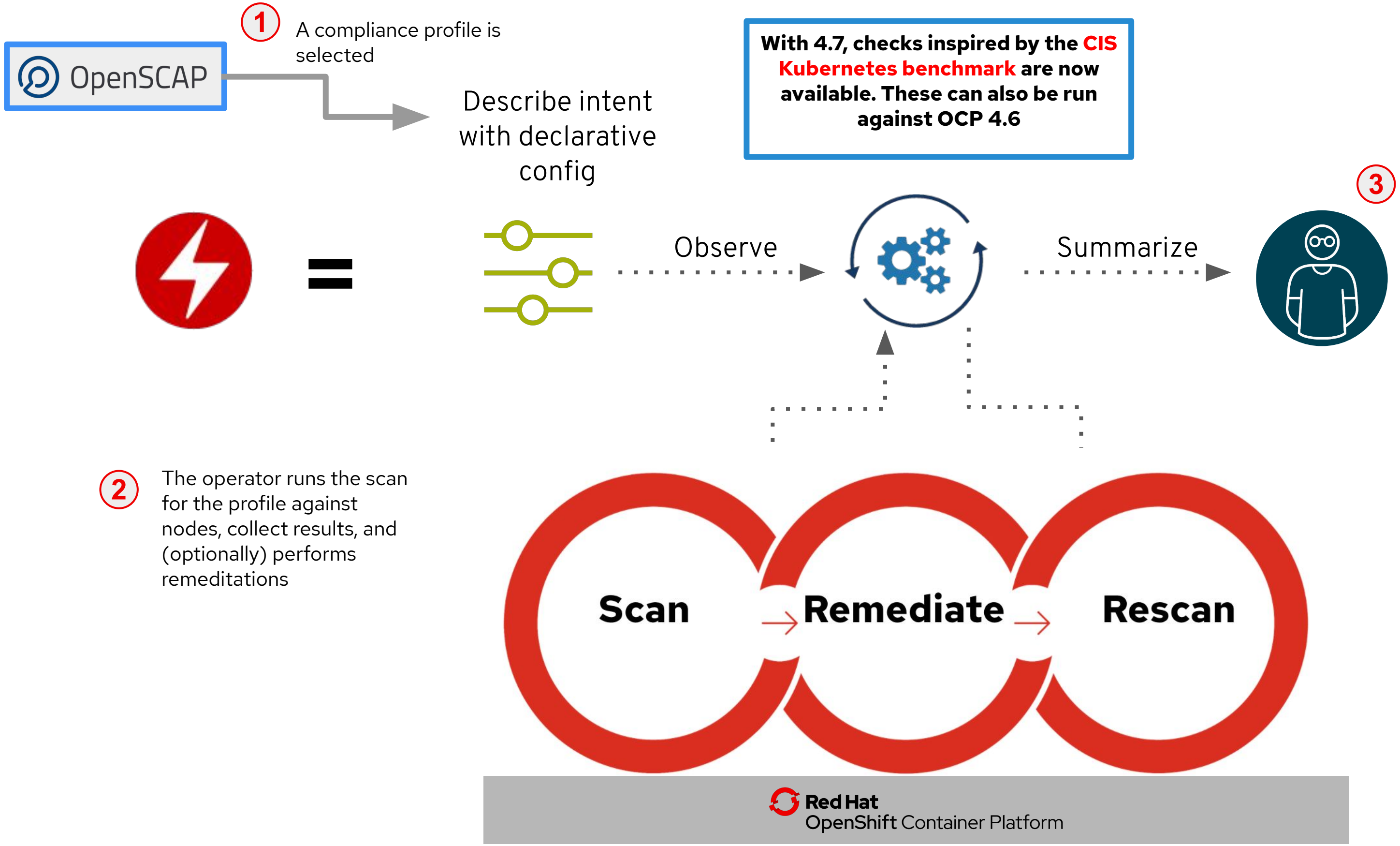
kubelet cri-o podman

Ignition systemd SELinux

RHEL CoreOS®

OCP 4.6

Openshift Compliance Operator: Declarative Security Compliance


OpenSCAP
Describe intent with declarative config
=
Observe
Summarize
Scan
Remediate
Rescan
Red Hat OpenShift Container Platform

1 A compliance profile is selected

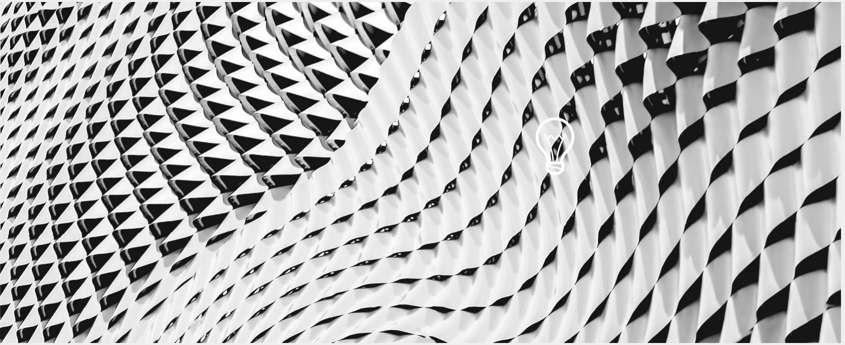
With 4.7, checks inspired by the CIS Kubernetes benchmark are now available. These can also be run against OCP 4.6

3 Accreditors or Auditors can examine the scan results for compliance status, After review, if desired, remediations can be manually applied by the cluster-admin.

2 The operator runs the scan for the profile against nodes, collect results, and (optionally) performs remeditations

RECOMMENDATION 2

Protect against malware



Requirement 5

Protect all systems against malware and regularly update anti-virus software or programs

Requirement 5: Considerations & Recommendations

- Look for a malware and/or anti-virus solution that is designed to work with containers or container images.
- Deploy the container security opertator
- Use Security Context Constraints
  - Containers cannot run as root by default
- Monitor for deprecated nodes
- Deploy the file integrity operator
- Container runtime security solutions (behavioral analysis) are available from Red Hat partners such as
  Aqua Security, Neuvector, Palo Alto (Twistlock), Sysdig

RHCOS & anti-virus scanners

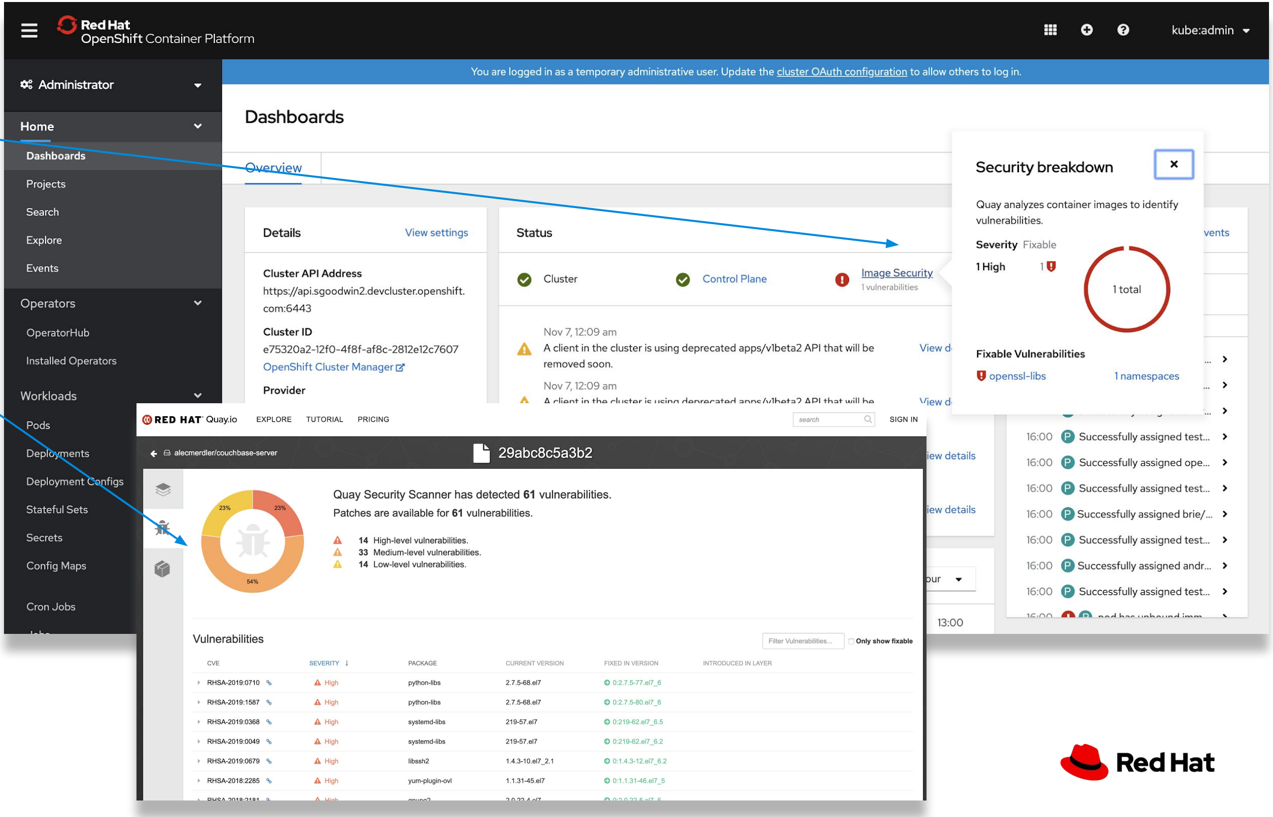
- Solutions, such as anti-virus scanners, can be deployed as daemonsets or container images. However, few, if any anti-virus vendors are delivering their software in this form.
- We recommend talking with your anti-virus vendor and asking what solutions they have available for a container-optimized OS.
- It's still RHEL
  - To see the **packages installed**, run the same commands as on a RHEL system: `rpm -qa |sort`
  - To see which **ports are in use**: `ss -tulpn`
- However, in a 2019 paper from Gartner on Cloud Workload Protection Platforms, recommends that clients "Replace antivirus (AV)-centric strategies with a "zero-trust execution"/default deny/application control approach to workload protection where possible, even if used only in detection mode." [1]

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

View Security Vulnerabilities with the Quay Operator

See all your Container Vulnerabilities right from the Console Dashboard

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



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Cluster API Address
https://api.sgoodwin2.devcluster.openshift.com:6443

Cluster ID
e75320a2-12f0-4f8f-af8c-2812e12c7607
OpenShift Cluster Manager

Provider

Nov 7, 12:09 am
A client in the cluster is using deprecated apps/v1beta2 API that will be removed soon.

Nov 7, 12:09 am
A client in the cluster is using deprecated apps/v1beta2 API that will be

Security breakdown

Quay analyzes container images to identify vulnerabilities.

Severity	Fixable
1 High	1

1 total

Fixable Vulnerabilities

openssl-libs	1 namespaces

16:00 Successfully assigned test...
16:00 Successfully assigned ope...
16:00 Successfully assigned test...
16:00 Successfully assigned brie/...
16:00 Successfully assigned test...
16:00 Successfully assigned andr...
16:00 Successfully assigned test...
16:00 red has unbound imm...

alecmerdler/couchbase-server

29abc8c5a3b2

Quay Security Scanner has detected **61** vulnerabilities.
Patches are available for **61** vulnerabilities.

- **14** High-level vulnerabilities.
- **33** Medium-level vulnerabilities.
- **14** Low-level vulnerabilities.

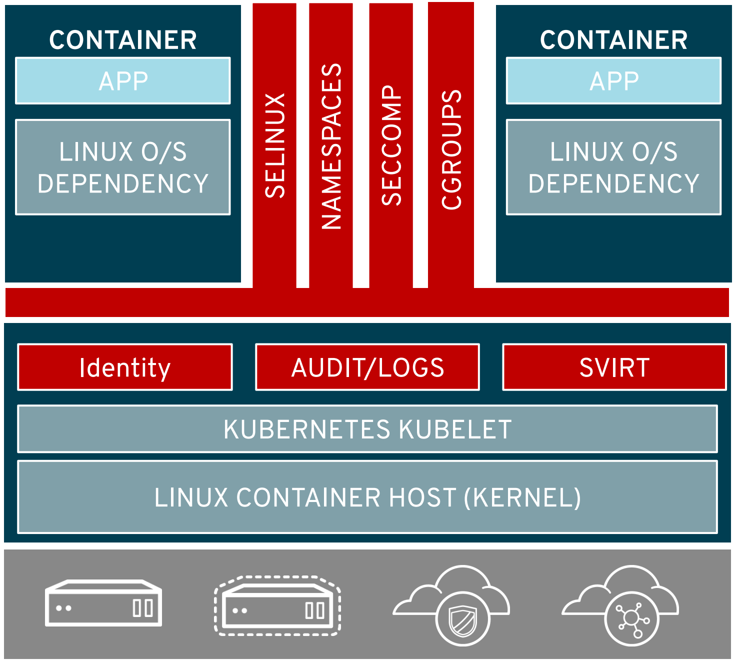
Vulnerabilities

Filter Vulnerabilities... □ Only show fixable

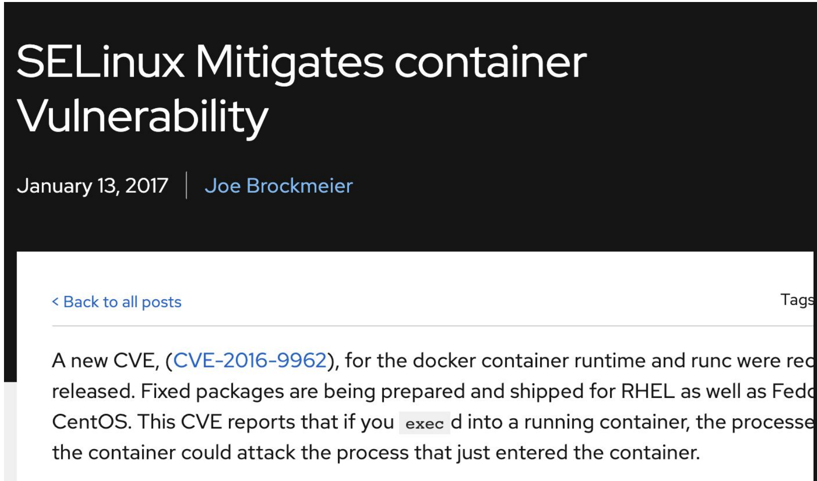
CVE	SEVERITY ↓	PACKAGE	CURRENT VERSION	FIXED IN VERSION	INTRODUCED IN LAYER
RHSA-2019:0710	High	python-libs	2.7.5-68.el7	0:2.7.5-77.el7_6	
RHSA-2019:1587	High	python-libs	2.7.5-68.el7	0:2.7.5-80.el7_6	
RHSA-2019:0368	High	systemd-libs	219-57.el7	0:219-62.el7_6.5	
RHSA-2019:0049	High	systemd-libs	219-57.el7	0:219-62.el7_6.2	
RHSA-2019:0679	High	libssh2	1.4.3-10.el7_2.1	0:1.4.3-12.el7_6.2	
RHSA-2018:2285	High	yum-plugin-ovl	1.1.31-45.el7	0:1.1.31-46.el7_5	
RHSA-2018:2181	High	gnupg2	2.0.22-4.el7	0:2.0.22-5.el7_5	

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

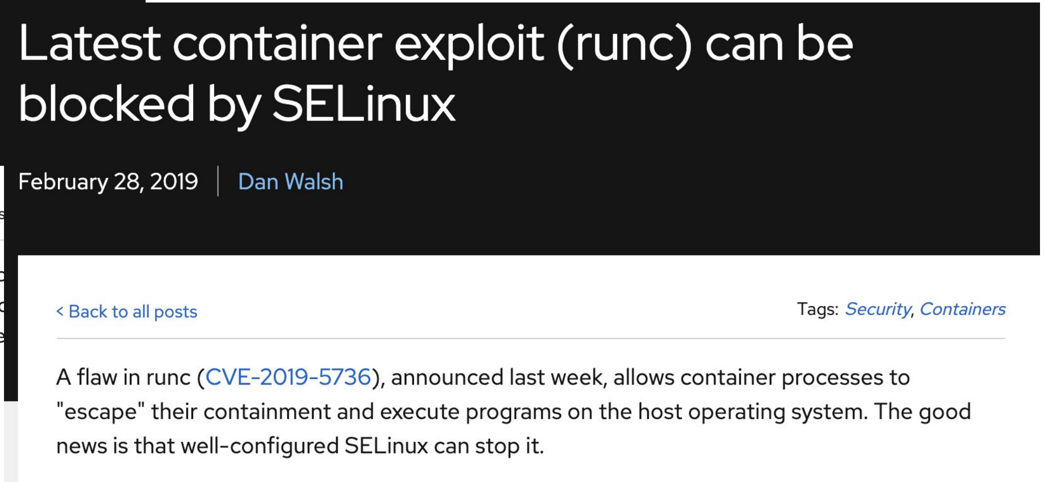
January 13, 2017 | Joe Brockmeier

< 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 Fedc 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

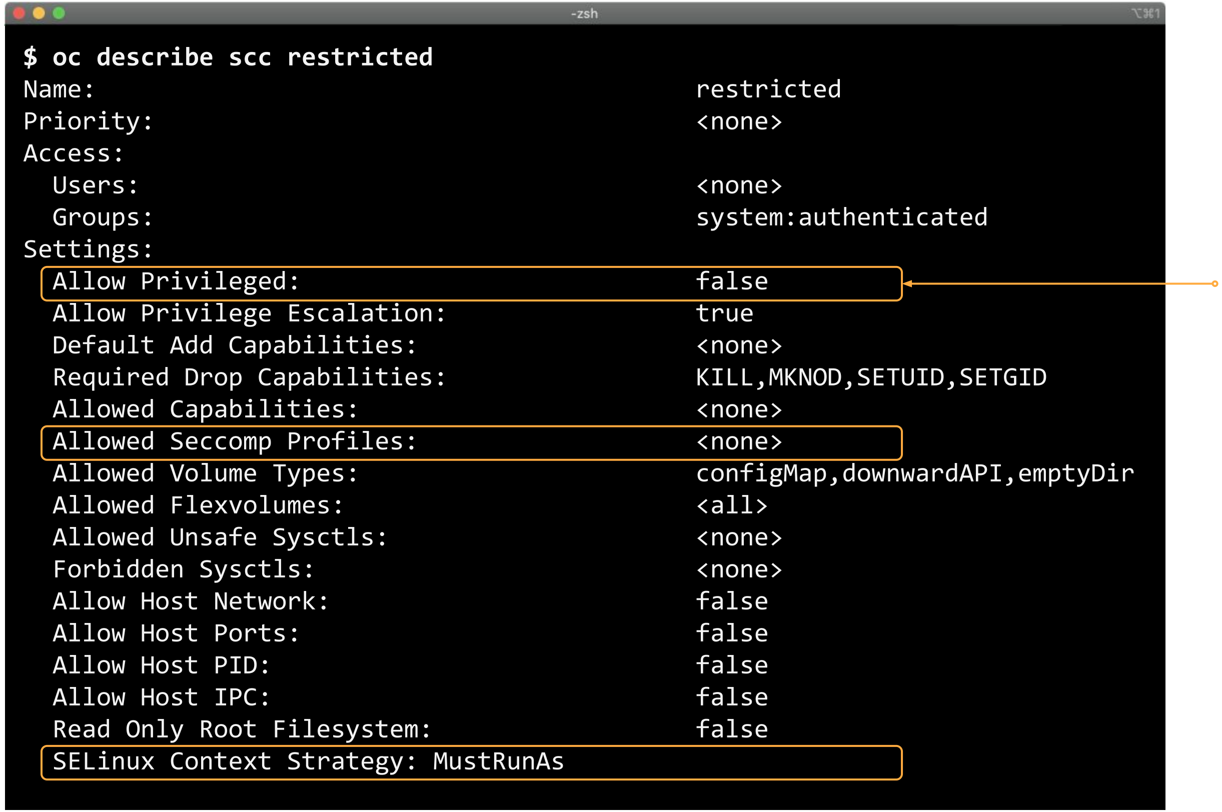
Tags: *Security*, *Containers*

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

Runtime security policies

(Pod Security Policies / Security Context Constraints)



```
$ oc describe scc restricted
Name:                                          restricted
Priority:                                      <none>
Access:
  Users:                                       <none>
  Groups:                                      system:authenticated
Settings:
  Allow Privileged:                            false
  Allow Privilege Escalation:                  true
  Default Add Capabilities:                    <none>
  Required Drop Capabilities:                  KILL,MKNOD,SETUID,SETGID
  Allowed Capabilities:                        <none>
  Allowed Seccomp Profiles:                    <none>
  Allowed Volume Types:                        configMap,downwardAPI,emptyDir
  Allowed Flexvolumes:                         <all>
  Allowed Unsafe Sysctls:                      <none>
  Forbidden Sysctls:                           <none>
  Allow Host Network:                          false
  Allow Host Ports:                            false
  Allow Host PID:                              false
  Allow Host IPC:                              false
  Read Only Root Filesystem:                   false
  SELinux Context Strategy: MustRunAs
```

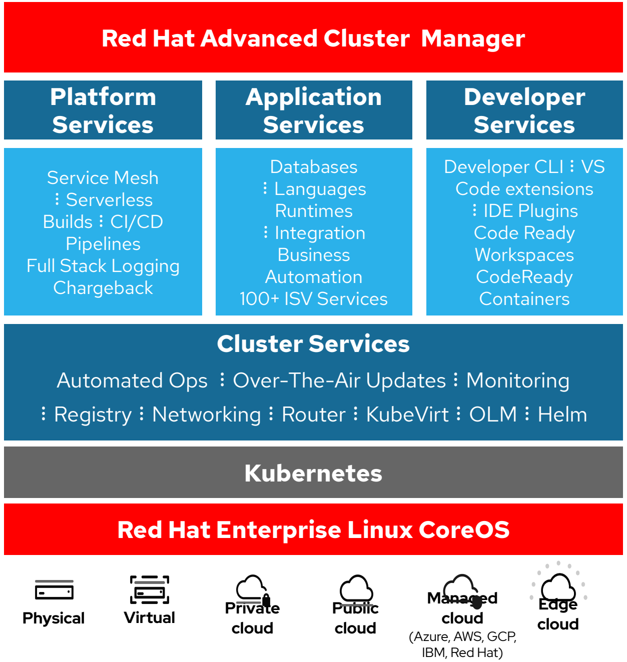
- Allow administrators to control permissions for pods
- By default, ensure no containers can run as root
- Admin can grant access to privileged PSP / SCC
- 8 included. Custom SCCs can be created

File Integrity

- Secure Boot - provides guarantee that a trusted, unmodified Kernel is loaded
- File integrity monitoring
  - /usr is read only
  - Machine Config Operator marks nodes with wrongly configured files as degraded
- Optional OpenShift File Integrity Operator using AIDE
  - Advanced Intrusion Detection Environment is a utility that creates a database of files on the system, and then uses that database to ensure file integrity and detect system intrusions

Red Hat OpenShift certified operators - Security

aqua
paloalto NETWORKS
sysdig
CYBERARK
NeuVector
JETSTACK
StackR
SYNOPSYS
snyk
portshift
Zettaset
IBM
Lacework
CYBER ARMOR
THALES
TREMOLO SECURITY
TIGERA
anchore
tufin


Red Hat Advanced Cluster Manager

Platform Services

Service Mesh ⁝ Serverless Builds ⁝ CI/CD Pipelines Full Stack Logging Chargeback

Application Services

Databases ⁝ Languages Runtimes ⁝ Integration Business Automation 100+ ISV Services

Developer Services

Developer CLI ⁝ VS Code extensions ⁝ IDE Plugins Code Ready Workspaces CodeReady Containers

Cluster Services

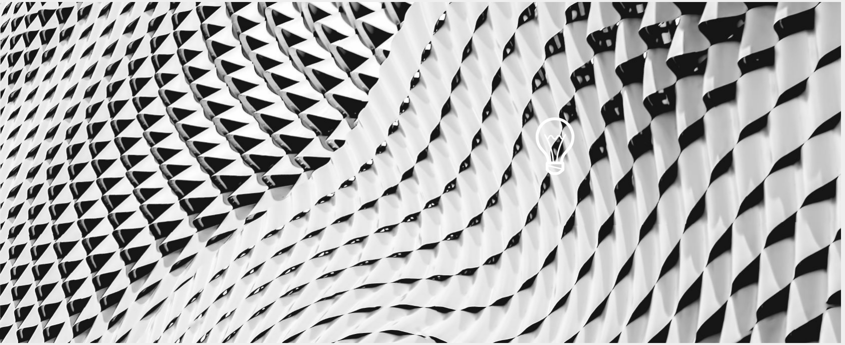
Automated Ops ⁝ Over-The-Air Updates ⁝ Monitoring ⁝ Registry ⁝ Networking ⁝ Router ⁝ KubeVirt ⁝ OLM ⁝ Helm

Kubernetes

Red Hat Enterprise Linux CoreOS

REQUIREMENT 6

Develop and maintain secure applications



Requirement 6: Develop & Maintain Secure Applications

6.1 Establish a process to identify security vulnerabilities

6.2 Ensure that all system components and software are protected by applying applicable vendor-supplied patches

6.4 Follow change control policies for all changes to system components (GitOps)

Requirement 6 Applicability

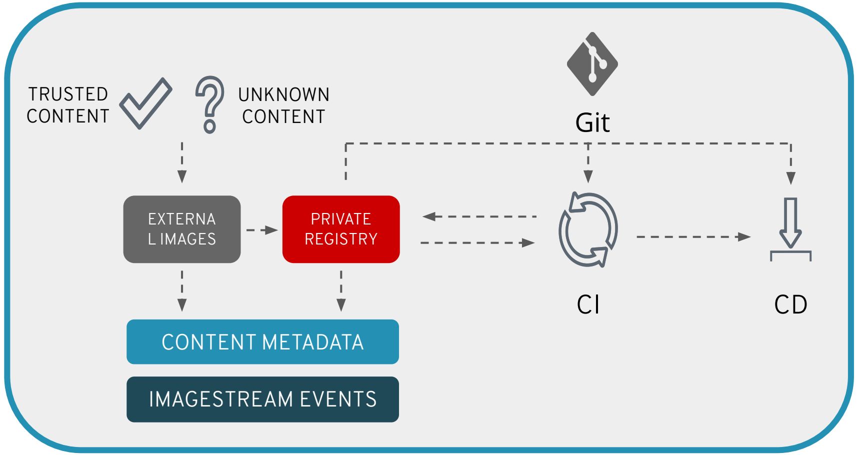
- “Immutable containers are containers that will never be changed while running.”
- The best practice for patching containerized applications is to rebuild the container image and redeploy from the image.
- Separate development, test and production deployments with OpenShift projects and deploy to separate hosts if appropriate (node-selector)
- Deploy OpenShift Service Mesh to encrypt service to service communication and add protection for application ingress and egress
- Keep up with OpenShift z stream releases

Securing Containerized Applications

An opportunity to shift security left

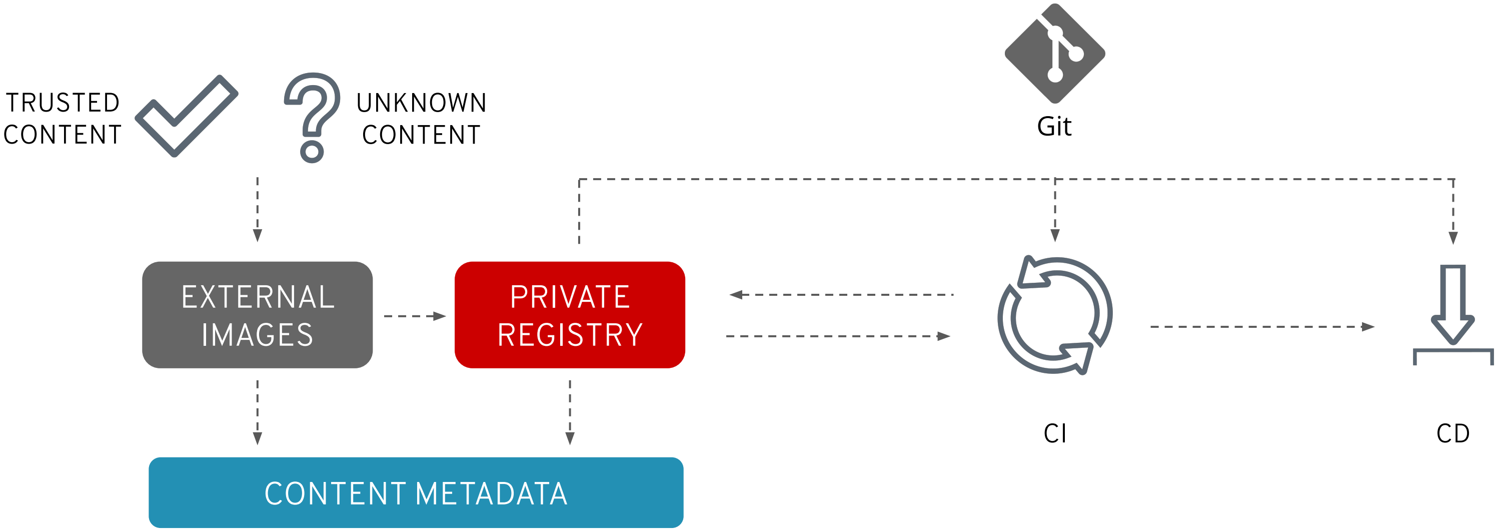
Best practices

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

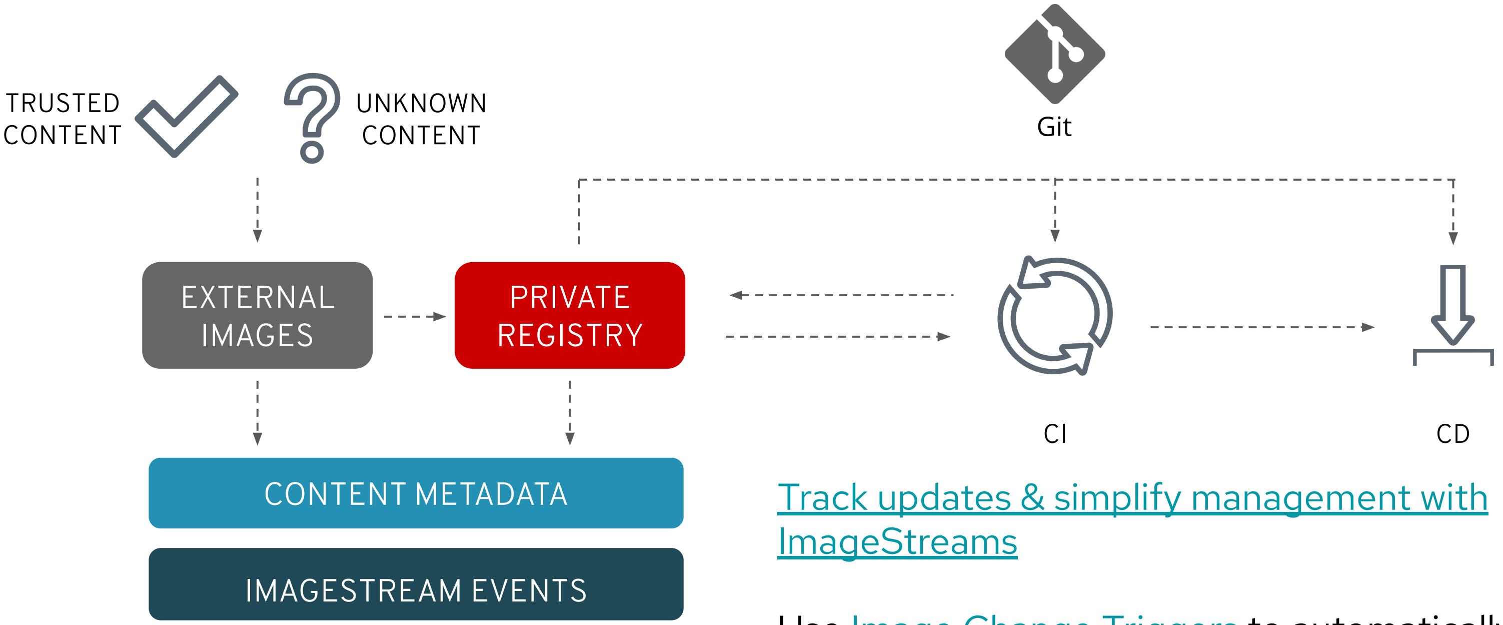


Secure & Automate The Content Lifecycle

Elements of the Openshift container pipeline



Trust is temporal: rebuild and redeploy as needed



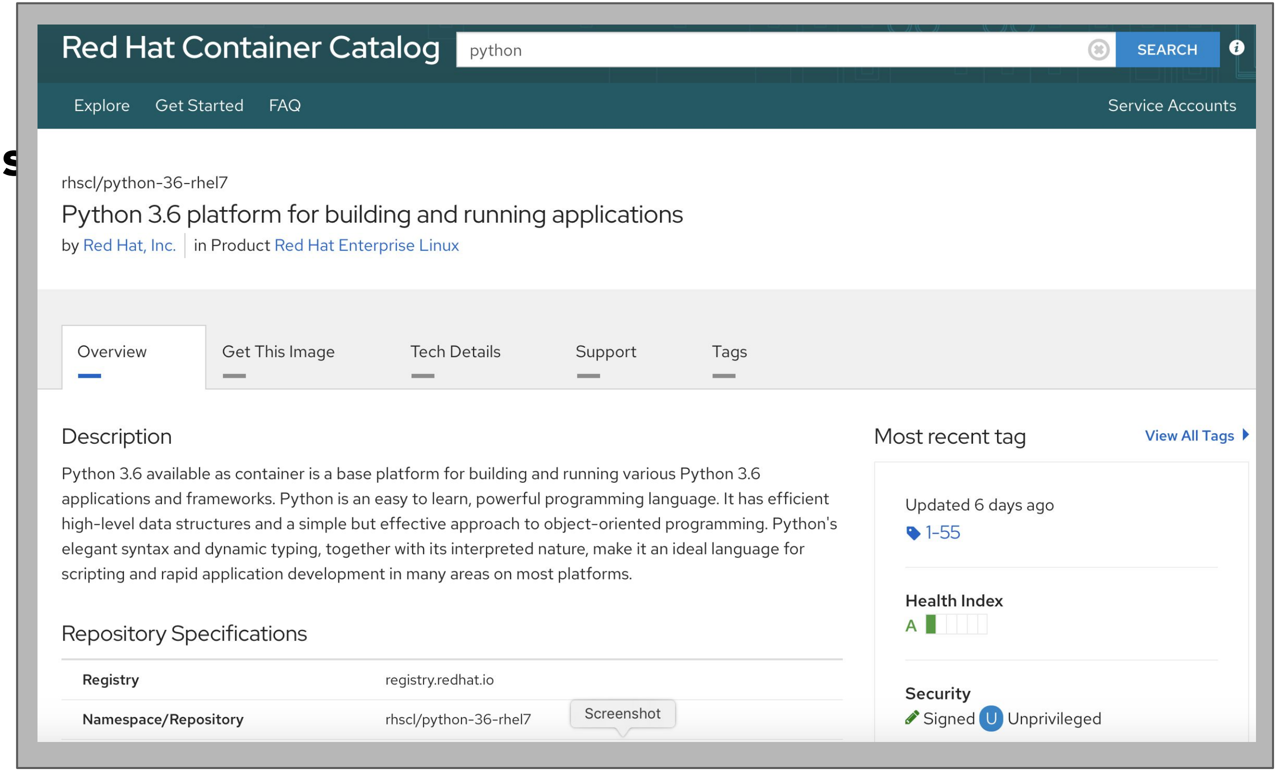
Track updates & simplify management with ImageStreams

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

External Content: Use Trusted Sources

Red Hat Container Images

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



rhscl/python-36-rhel7

Python 3.6 platform for building and running applications

by Red Hat, Inc. | in Product Red Hat Enterprise Linux

Description

Python 3.6 available as container is a base platform for building and running various Python 3.6 applications and frameworks. Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

Repository Specifications

Registry	registry.redhat.io
Namespace/Repository	rhscl/python-36-rhel7

Most recent tag

View All Tags

Updated 6 days ago

1-55

Health Index

A

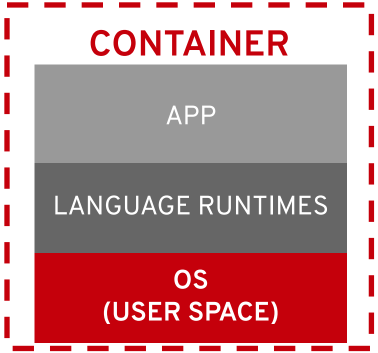
Security

Signed U Unprivileged

*More about the Health Index

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

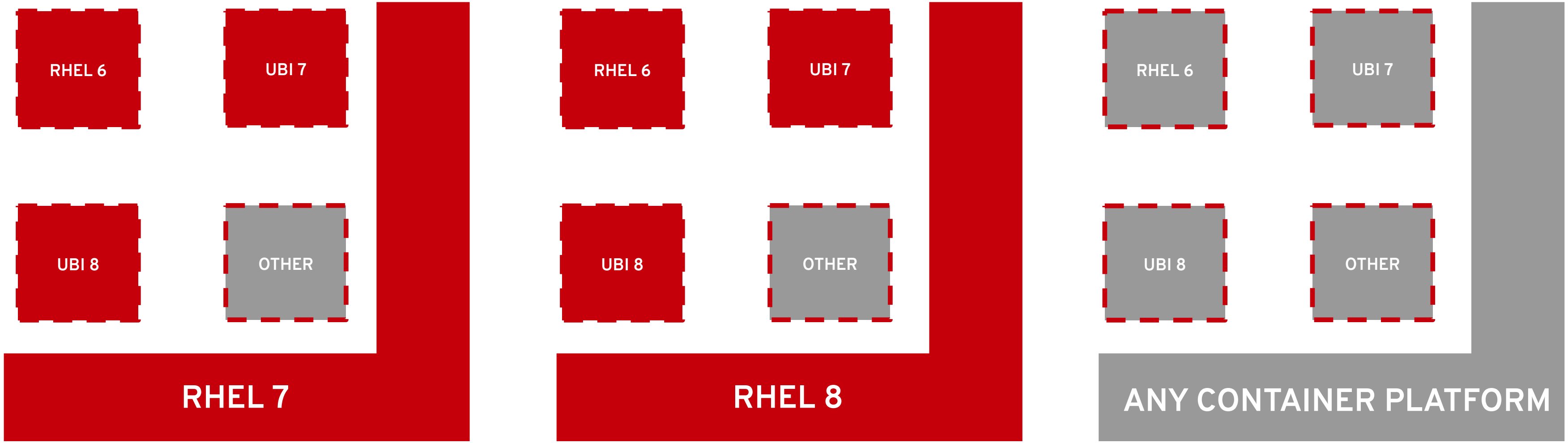
- Supported as RHEL when running on RHEL
- Same Performance, Security & Life cycle as RHEL
- Can attach RHEL support subscriptions as RHEL

Supportability matrix

Red Hat Support and Community Support

🟥 RED HAT SUPPORT

▨ COMMUNITY SUPPORT



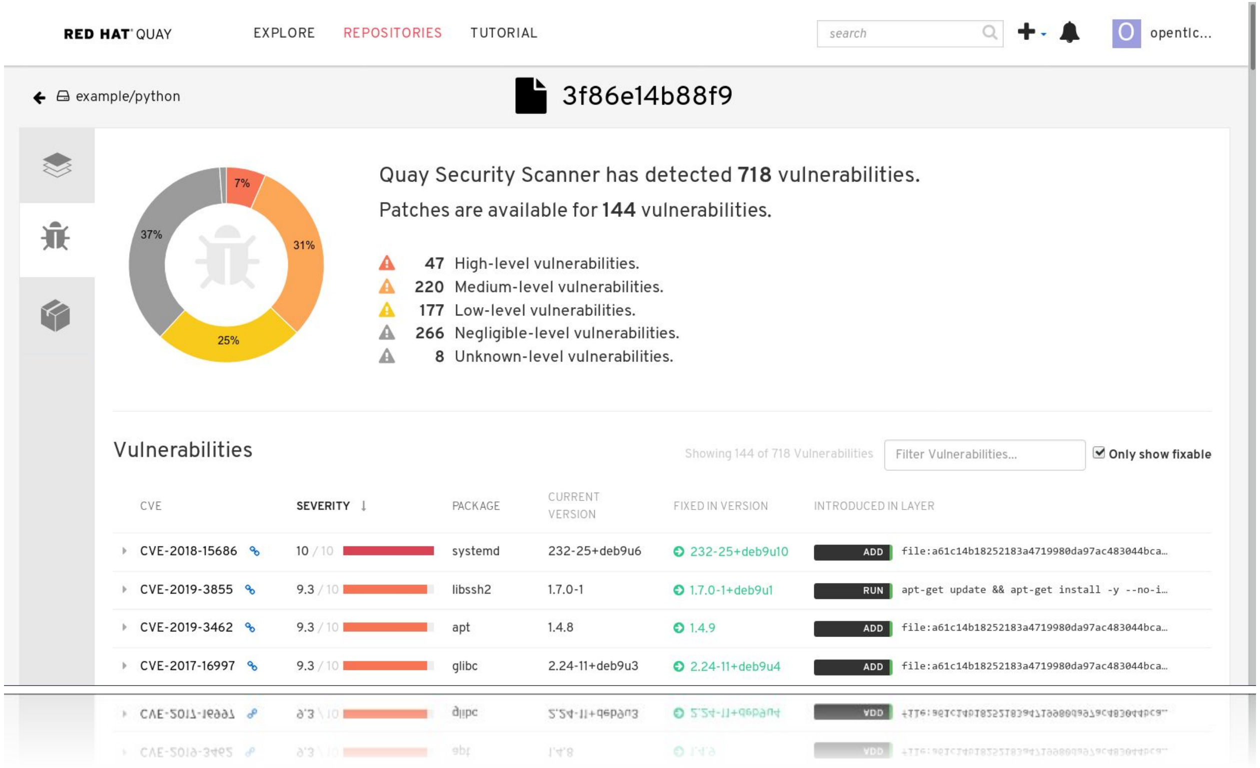
Red Hat Enterprise Linux 7

Red Hat Enterprise Linux 8

Like any upstream project

Red Hat Quay
Enterprise Container Registry

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



3f86e14b88f9

Quay Security Scanner has detected **718** vulnerabilities.

Patches are available for **144** vulnerabilities.

47	High-level vulnerabilities.
220	Medium-level vulnerabilities.
177	Low-level vulnerabilities.
266	Negligible-level vulnerabilities.
8	Unknown-level vulnerabilities.

Vulnerabilities

Showing 144 of 718 Vulnerabilities

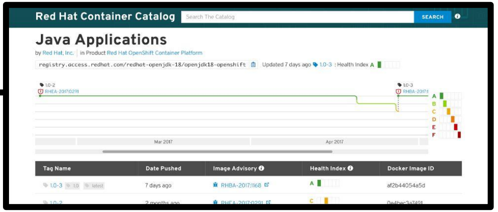
Filter Vulnerabilities...

☑ **Only show fixable**

CVE	SEVERITY ↓	PACKAGE	CURRENT VERSION	FIXED IN VERSION	INTRODUCED IN LAYER
CVE-2018-15686	10 / 10	systemd	232-25+deb9u6	232-25+deb9u10	ADD file:a61c14b18252183a4719980da97ac483044bca...
CVE-2019-3855	9.3 / 10	libssh2	1.7.0-1	1.7.0-1+deb9u1	RUN apt-get update && apt-get install -y --no-i...
CVE-2019-3462	9.3 / 10	apt	1.4.8	1.4.9	ADD file:a61c14b18252183a4719980da97ac483044bca...
CVE-2017-16997	9.3 / 10	glibc	2.24-11+deb9u3	2.24-11+deb9u4	ADD file:a61c14b18252183a4719980da97ac483044bca...

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

by Red Hat, Inc. | in Product Red Hat OpenShift Container Platform

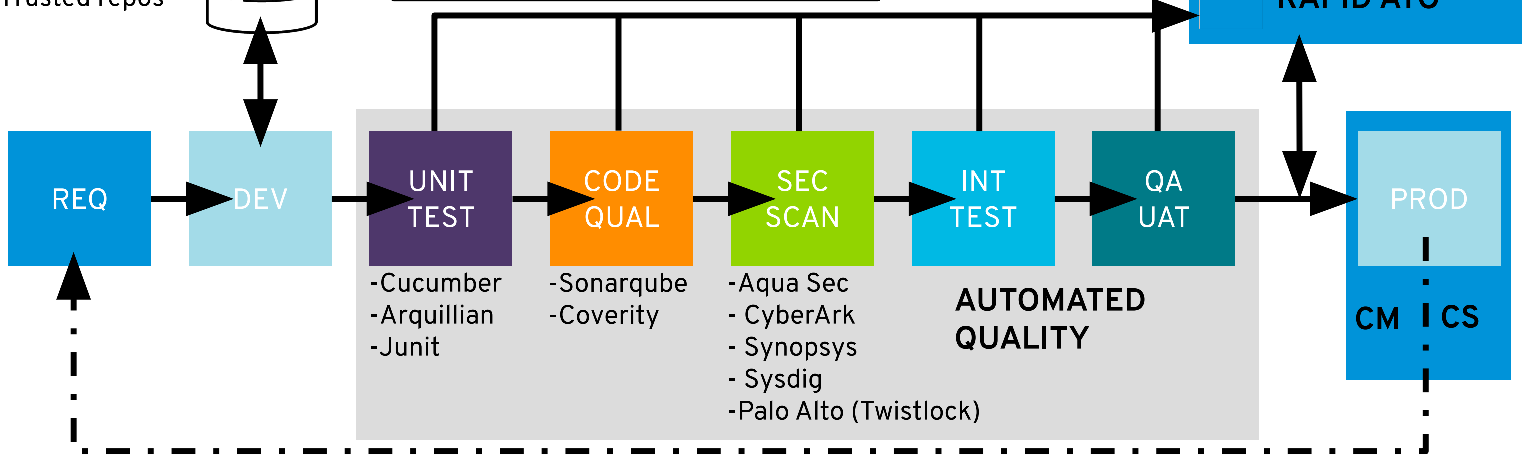
registry.access.redhat.com/redhat-openjdk-18/openjdk18-openshift Updated 7 days ago 1.0-3 : Health Index A

Tag Name	Date Pushed	Image Advisory	Health Index	Docker Image ID
1.0-3 1.0 latest	7 days ago	RHBA-2017:1168	A	af2b44054a5d
1.0-2	2 months ago	RHEA-2017:0291	C	0e4bec3a7491

Secure Supply Chain of Quality Parts

- Quality Assurance
- Certifications
- Signing & Secure
- Distribution

Automatically prohibit untrusted containers via Openshift policy



-Cucumber
-Arquillian
-Junit

-Sonarqube
-Coverity

-Aqua Sec
- CyberArk
- Synopsys
- Sysdig
-Palo Alto (Twistlock)

AUTOMATED QUALITY

-Aqua Sec
- CyberArk
- Neuvector
-Palo Alto (Twistlock)
- Sysdig

OPENSHIFT SOFTWARE FACTORY

Red Hat Container Catalog: Java Applications

More about the Container Health Index

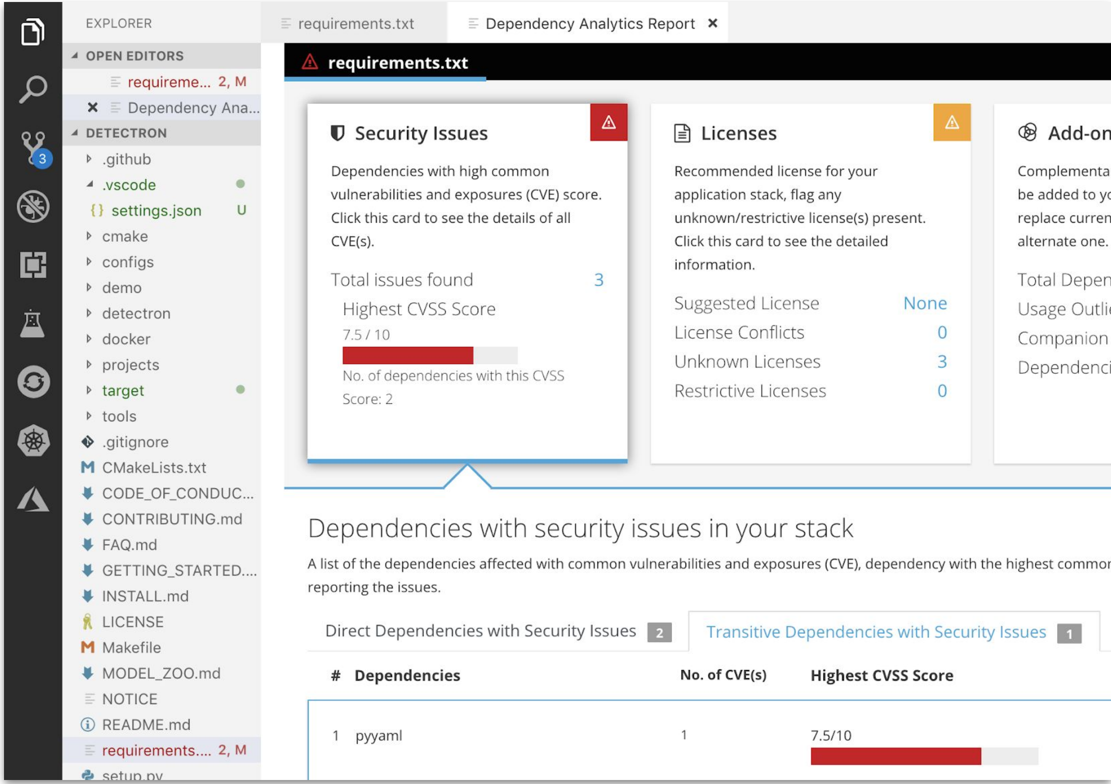
Plugins available for VSCode and Eclipse IDEs

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



requirements.txt

Security Issues

Dependencies with high common vulnerabilities and exposures (CVE) score. Click this card to see the details of all CVE(s).

Total issues found 3

Highest CVSS Score

7.5 / 10

No. of dependencies with this CVSS Score: 2

Licenses

Recommended license for your application stack, flag any unknown/restrictive license(s) present. Click this card to see the detailed information.

Suggested License None

License Conflicts 0

Unknown Licenses 3

Restrictive Licenses 0

Add-on

Complementa be added to yo replace curren alternate one.

Total Depen

Usage Outli

Companion

Dependenci

Dependencies with security issues in your stack

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

Direct Dependencies with Security Issues 2 | Transitive Dependencies with Security Issues 1

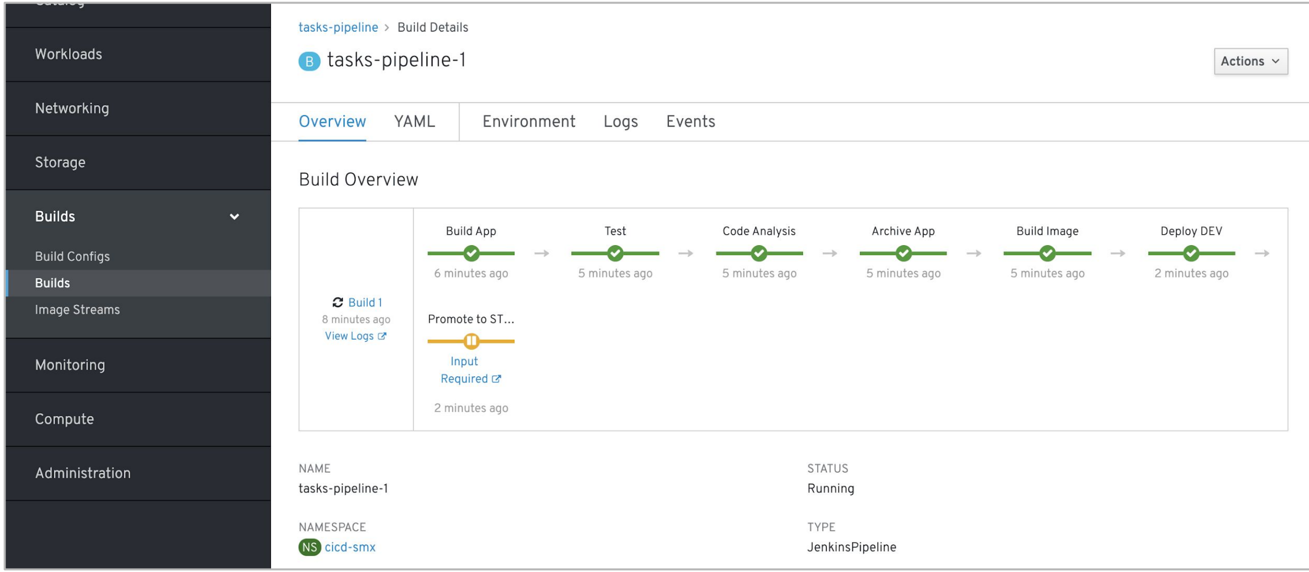
#	Dependencies	No. of CVE(s)	Highest CVSS Score
1	pyyaml	1	7.5/10

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

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

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

The full Jenkins UI is also available.



tasks-pipeline > Build Details

B tasks-pipeline-1

Build Overview

Build 1
8 minutes ago
View Logs

Build App	Test	Code Analysis	Archive App	Build Image	Deploy DEV
6 minutes ago	5 minutes ago	5 minutes ago	5 minutes ago	5 minutes ago	2 minutes ago

Promote to ST...
Input Required
2 minutes ago

NAME
tasks-pipeline-1

STATUS
Running

NAMESPACE
NS cicd-smx

TYPE
JenkinsPipeline

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

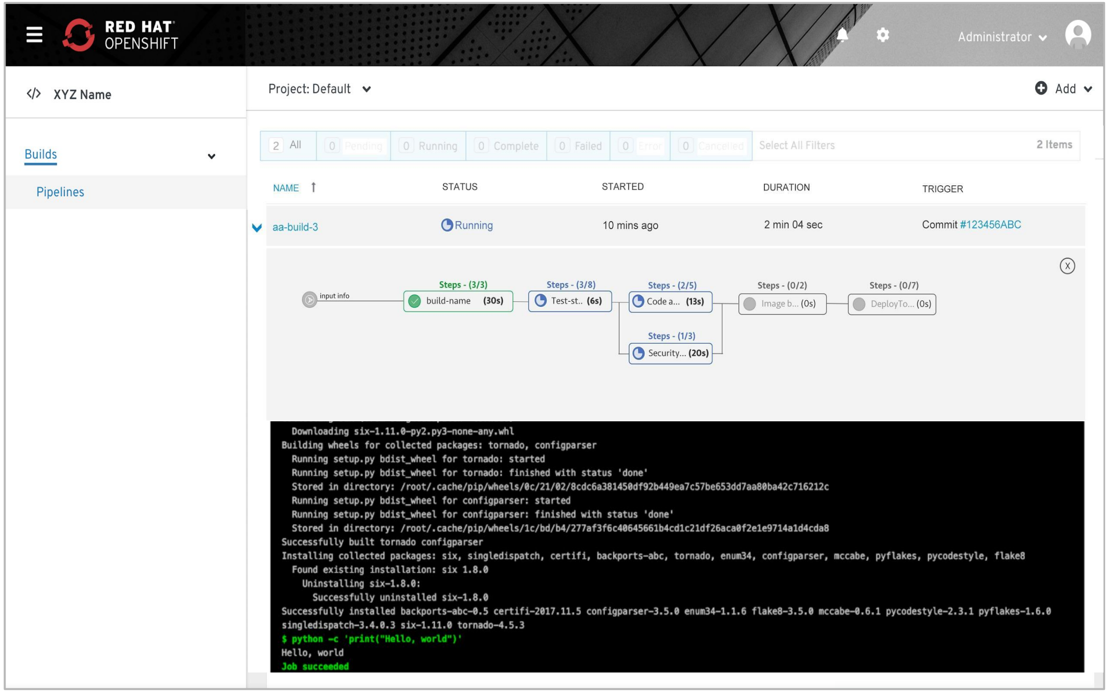
Tech Preview

OpenShift Pipelines: A Kubernetes-native CI/CD platform

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

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

Target general availability in OpenShift 4.7.



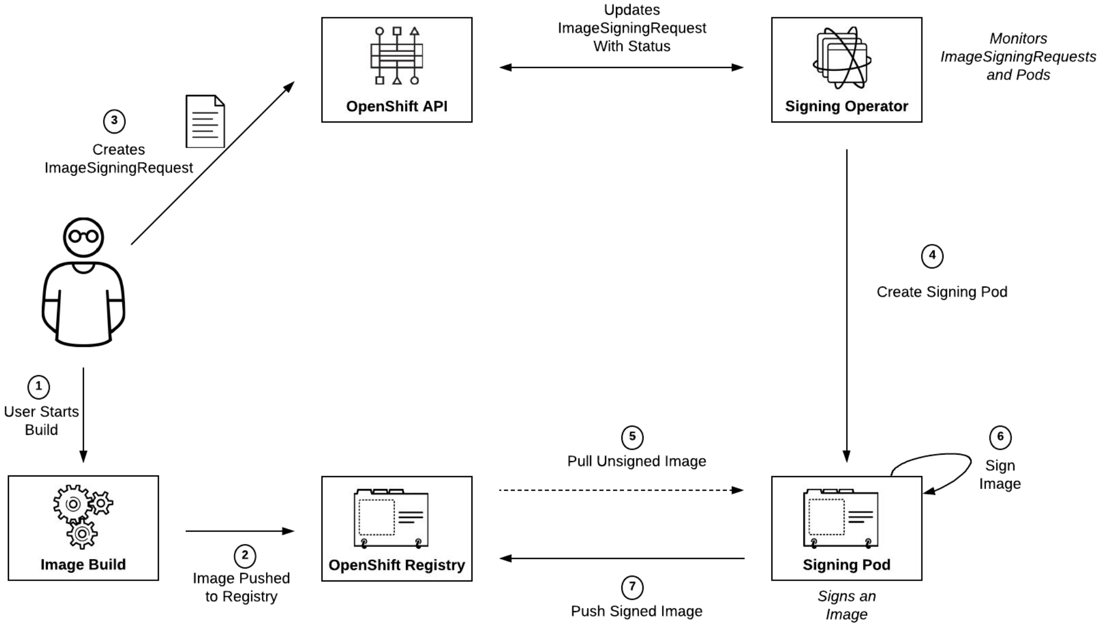
NAME	STATUS	STARTED	DURATION	TRIGGER
aa-build-3	Running	10 mins ago	2 min 04 sec	Commit #123456ABC

```
  Downloading six-1.11.0-py2.py3-none-any.whl
Building wheels for collected packages: tornado, configparser
  Running setup.py bdist_wheel for tornado: started
  Running setup.py bdist_wheel for tornado: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/0c/21/02/8cdc6a381450df92b449ea7c57be653dd7aa80ba42c716212c
  Running setup.py bdist_wheel for configparser: started
  Running setup.py bdist_wheel for configparser: finished with status 'done'
  Stored in directory: /root/.cache/pip/wheels/1c/bd/b4/277af3f6c40645661b4cd1c21df26aca0f2e1e9714a1d4cda8
Successfully built tornado configparser
Installing collected packages: six, singledispatch, certifi, backports-abc, tornado, enum34, configparser, mccabe, pyflakes, pycodestyle, flake8
  Found existing installation: six 1.8.0
    Uninstalling six-1.8.0:
      Successfully uninstalled six-1.8.0
Successfully installed backports-abc-0.5 certifi-2017.11.5 configparser-3.5.0 enum34-1.1.6 flake8-3.5.0 mccabe-0.6.1 pycodestyle-2.3.1 pyflakes-1.6.0
singledispatch-3.4.0.3 six-1.11.0 tornado-4.5.3
$ python -c 'print("Hello, world")'
Hello, world
Job succeeded
```

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

Container Signing

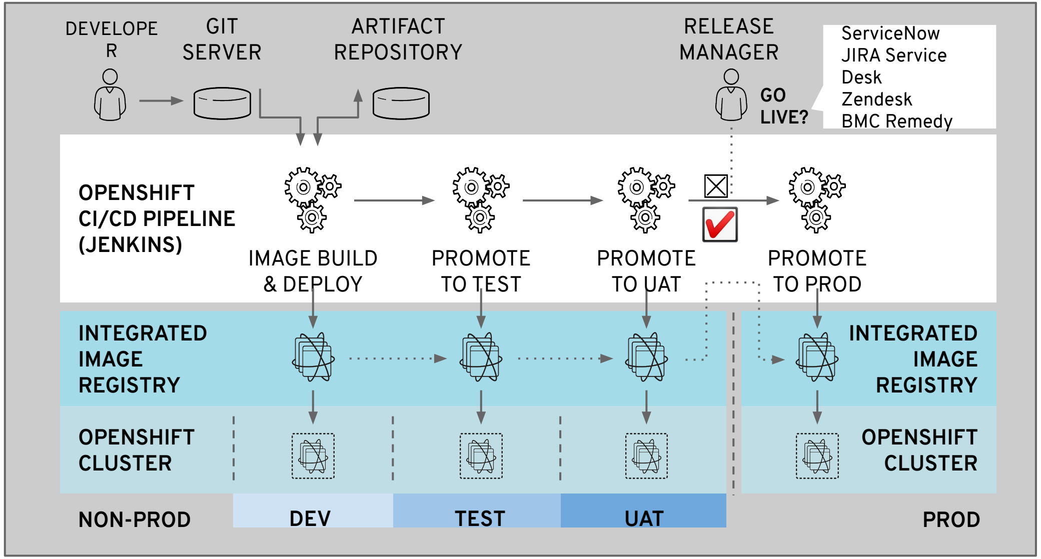
A simplified and automated approach to signing container images



Red Hat

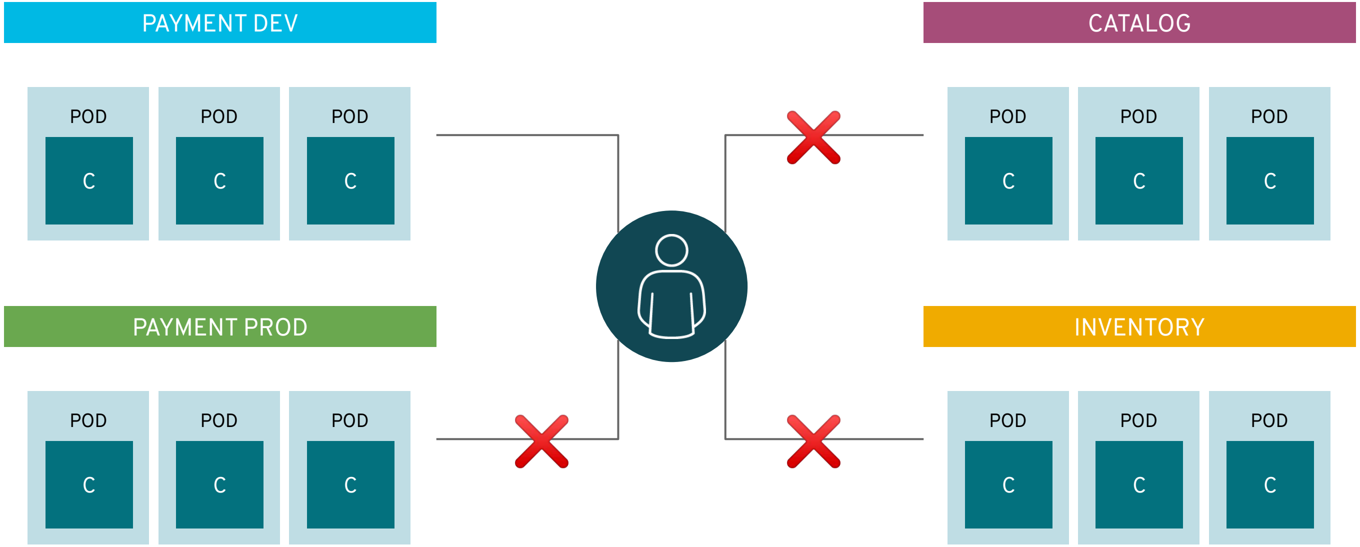
Managing Container Deployment

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



Red Hat

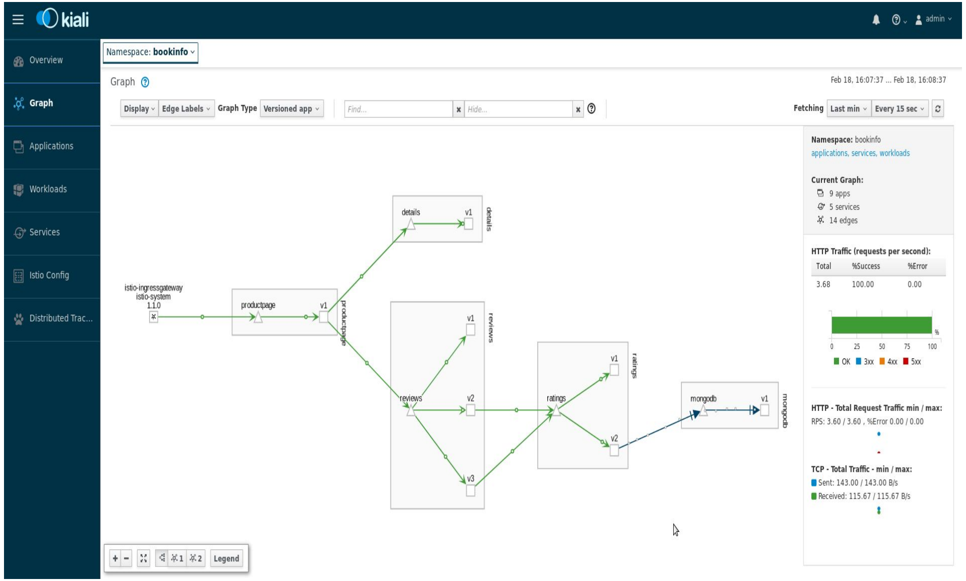
Projects isolate applications across teams, groups and departments



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



Namespace: **bookinfo**

Graph

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

Namespace: bookinfo
applications, services, workloads

Current Graph:
9 apps
5 services
14 edges

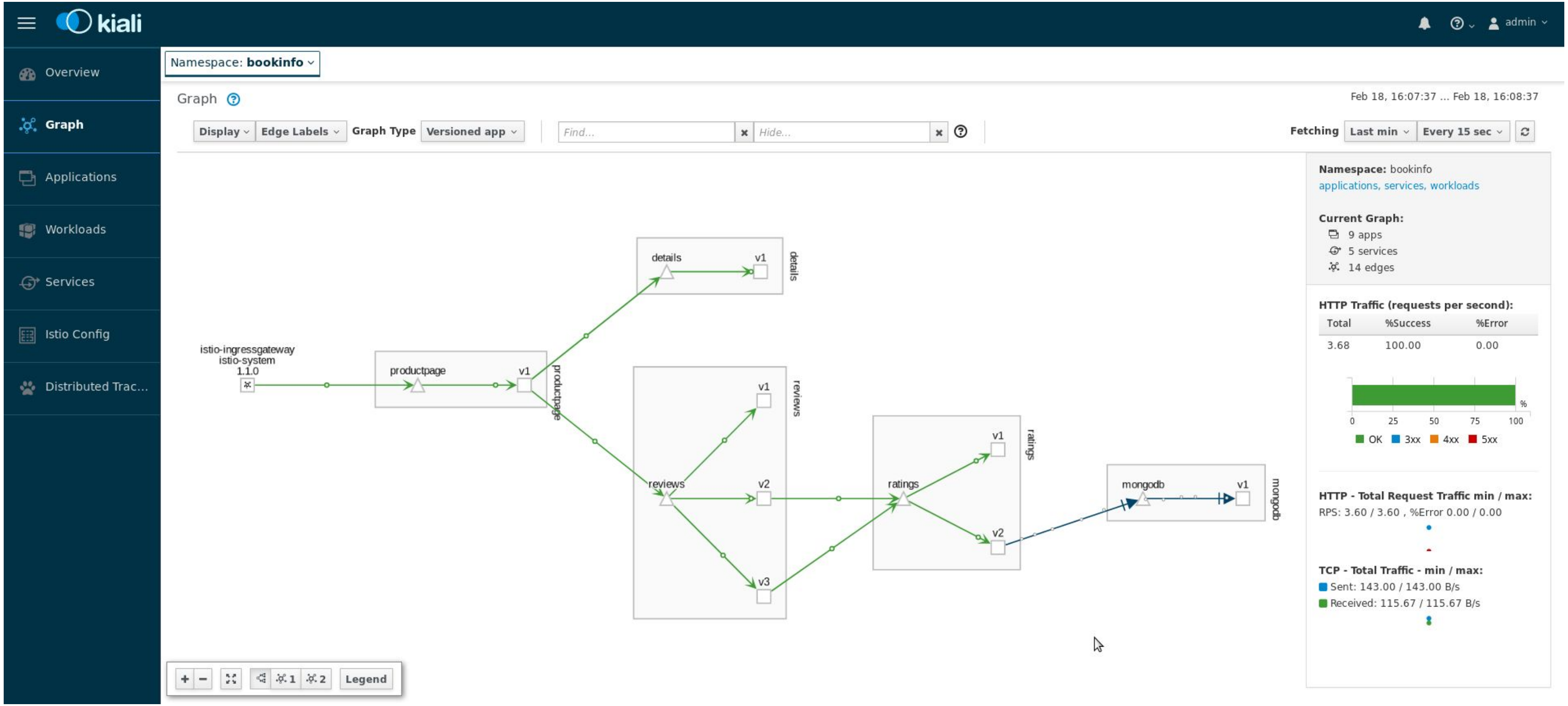
HTTP Traffic (requests per second):

Total	%Success	%Error
3.68	100.00	0.00

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

TCP - Total Traffic - min / max:
🟦 Sent: 143.00 / 143.00 B/s
🟩 Received: 115.67 / 115.67 B/s

Observability with Kiali



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

Namespace: bookinfo
applications, services, workloads

Current Graph:
9 apps
5 services
14 edges

HTTP Traffic (requests per second):

Total	%Success	%Error
3.68	100.00	0.00

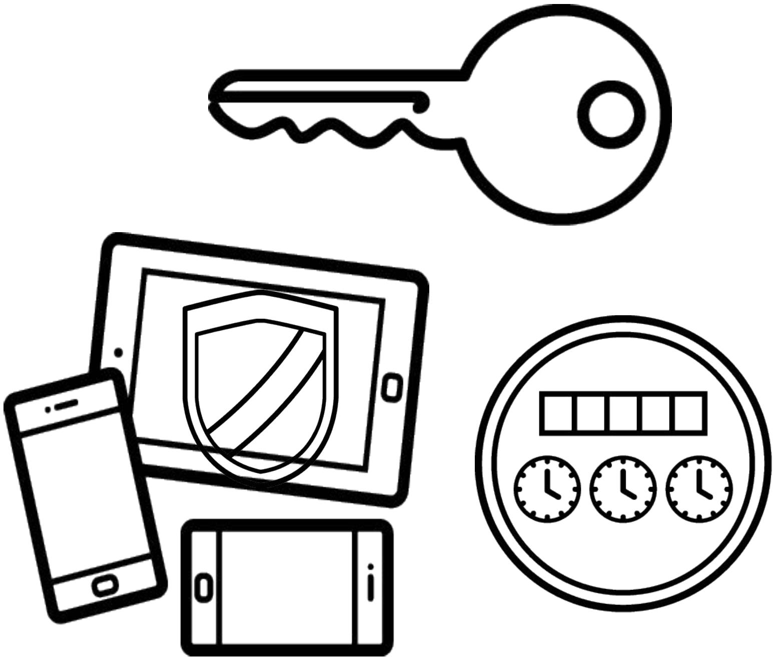
HTTP - Total Request Traffic min / max:
RPS: 3.60 / 3.60 , %Error 0.00 / 0.00

TCP - Total Traffic - min / max:
Sent: 143.00 / 143.00 B/s
Received: 115.67 / 115.67 B/s

Application API management

Consider configuring an API gateway for container platform & application APIs

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



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

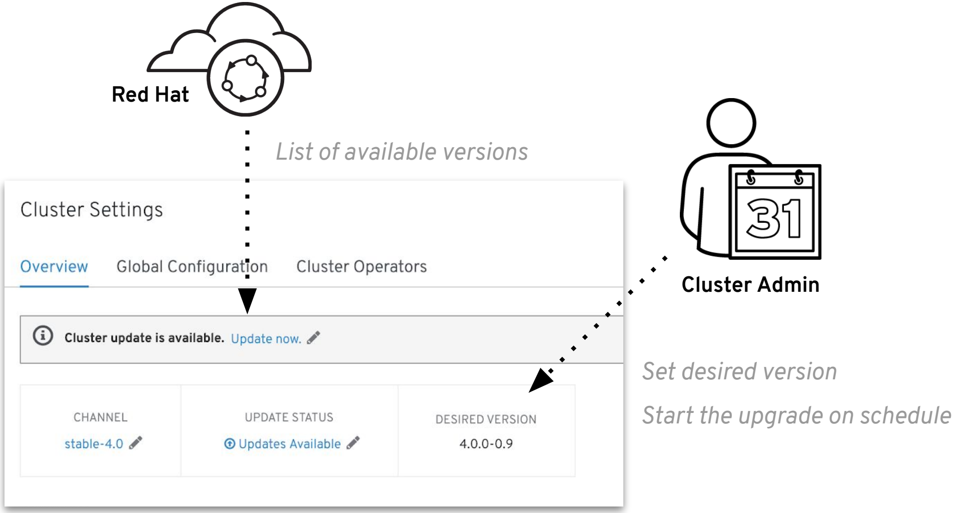
No need for separate windows for each component

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

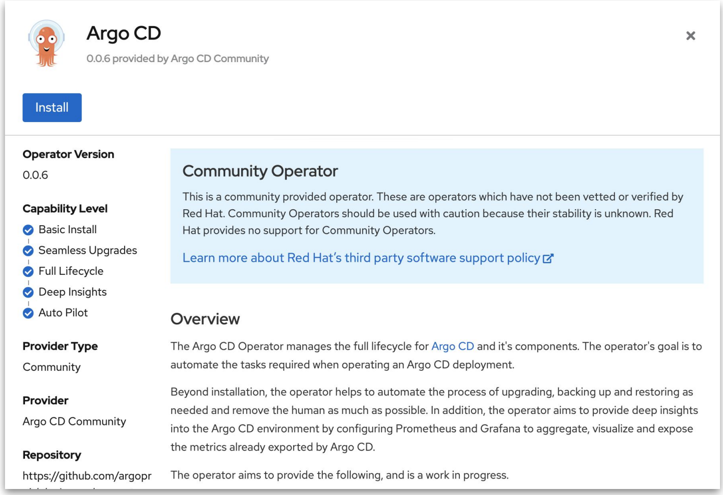


ⓘ Cluster update is available. Update now.

CHANNEL	UPDATE STATUS	DESIRED VERSION
stable-4.0	Updates Available	4.0.0-0.9

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



Argo CD

0.0.6 provided by Argo CD Community

Operator Version

0.0.6

Capability Level

- ✓ Basic Install
- ✓ Seamless Upgrades
- ✓ Full Lifecycle
- ✓ Deep Insights
- ✓ Auto Pilot

Provider Type

Community

Provider

Argo CD Community

Repository

https://github.com/argopr

Community Operator

This is a community provided operator. These are operators which have not been vetted or verified by Red Hat. Community Operators should be used with caution because their stability is unknown. Red Hat provides no support for Community Operators.

Learn more about Red Hat's third party software support policy

Overview

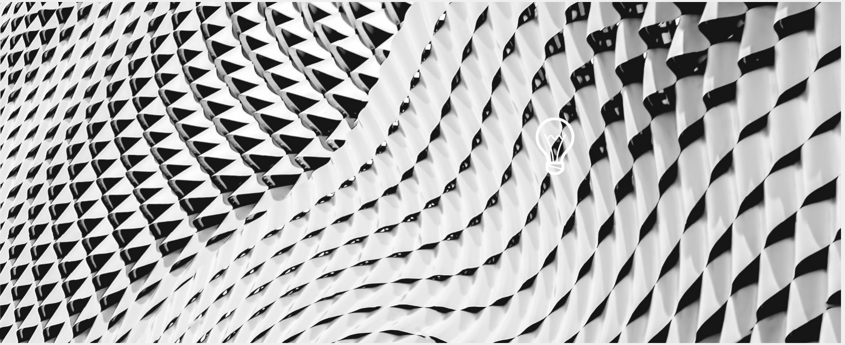
The Argo CD Operator manages the full lifecycle for Argo CD and it's components. The operator's goal is to automate the tasks required when operating an Argo CD deployment.

Beyond installation, the operator helps to automate the process of upgrading, backing up and restoring as needed and remove the human as much as possible. In addition, the operator aims to provide deep insights into the Argo CD environment by configuring Prometheus and Grafana to aggregate, visualize and expose the metrics already exported by Argo CD.

The operator aims to provide the following, and is a work in progress.

REQUIREMENT 7 & 8

Restrict access by need to know
Identity and access management



Requirement 7 & 8: Restrict access by need to know, IAM

7. Restrict access to cardholder data by business need to know

   7.1 Limit access to system components and cardholder data to only those who need to know

   7.2 Establish an access control system for system components; default to deny all

   7.3 Ensure that security policies and procedures are documented and in use (Git Ops, Compliance operator)

8. Identify and authenticate access to system components

Requirement 7 & 8 Applicability

- Use OpenShifts built-in RBAC, projects and network policies to isolate users, teams and applications from each other

- Use a 3rd party, external identity provider (integrate via LDAP or OAuth)

- Manage account lockouts, session timeouts, via external IdP (can manage session timeouts in OpenShift)

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

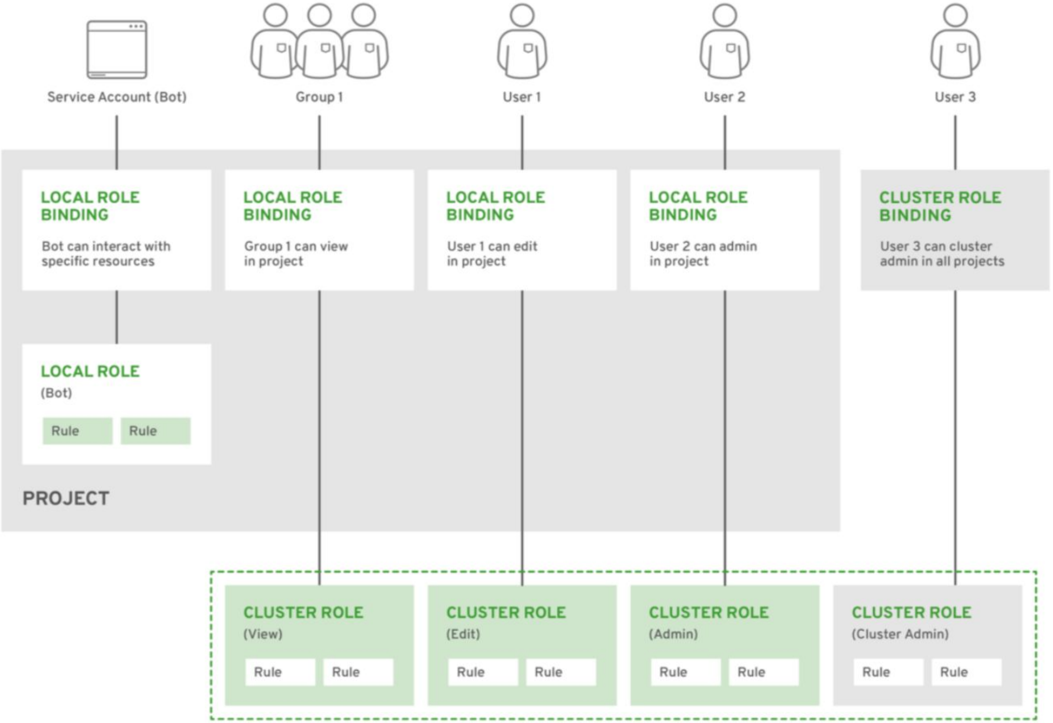
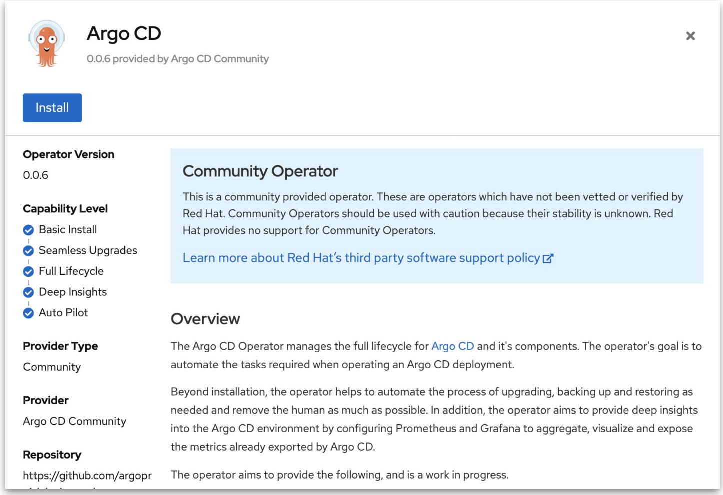


Figure 12 - Authorization Relationships

For more information see: [Managing RBAC in OpenShift]

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



Argo CD

0.0.6 provided by Argo CD Community

Operator Version

0.0.6

Capability Level

- ✓ Basic Install
- ✓ Seamless Upgrades
- ✓ Full Lifecycle
- ✓ Deep Insights
- ✓ Auto Pilot

Provider Type

Community

Provider

Argo CD Community

Repository

https://github.com/argopr

Community Operator

This is a community provided operator. These are operators which have not been vetted or verified by Red Hat. Community Operators should be used with caution because their stability is unknown. Red Hat provides no support for Community Operators.

Learn more about Red Hat's third party software support policy

Overview

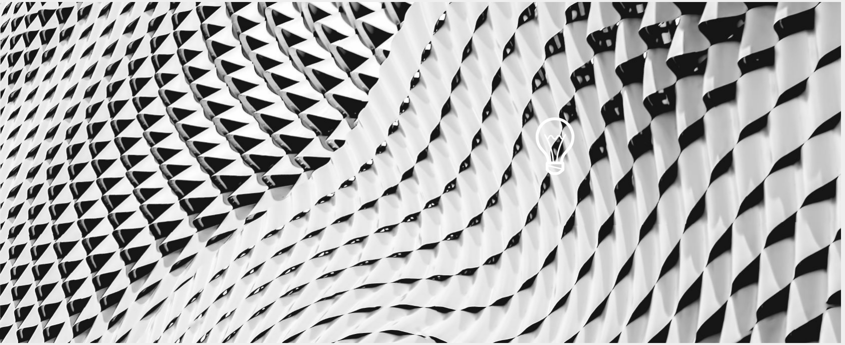
The Argo CD Operator manages the full lifecycle for Argo CD and it's components. The operator's goal is to automate the tasks required when operating an Argo CD deployment.

Beyond installation, the operator helps to automate the process of upgrading, backing up and restoring as needed and remove the human as much as possible. In addition, the operator aims to provide deep insights into the Argo CD environment by configuring Prometheus and Grafana to aggregate, visualize and expose the metrics already exported by Argo CD.

The operator aims to provide the following, and is a work in progress.

REQUIREMENT 10

Track and monitor all access



Requirement 10: Track and Monitor all access

10.1 Implement audit trails to link system actions to users

10.2 Implement automated audit trails to monito

- 10.2.1 All individual user access to card data
- 10.2.2 All actions taken by privileged users
- 10.2.3 Access to audit trails
- 10.2.4 Invalid logical access attempts
- 10.2.5 Use of and changes to identity and authentication methods
- 10.2.6 Initializing, stopping, pausing of audit logs
- 10.2.7 Creation and deletion of system level objects

10.3 Record user identity, type of event, date and time of event, success or failure, origin of event, identity or name of target of event

10.4 Use time synchronization technology

10.5 Secure audit trails so they cannot be altered

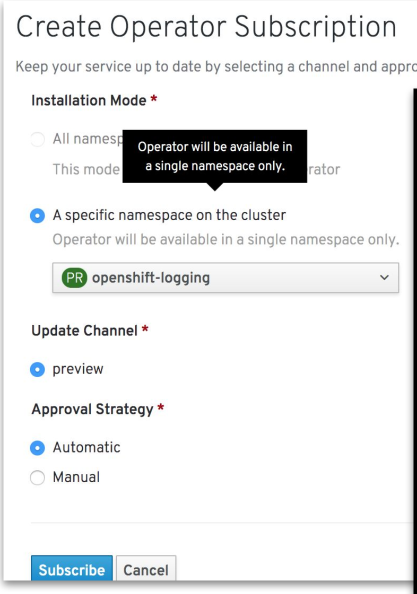
Requirement 10 Applicability

- Host level and API server audit is on by default
- Evaluate default audit levels
- Forward all logs, including audit logs, to SIEM

Cluster log and audit management

Install the Elasticsearch and Cluster Logging Operators

- EFK stack aggregates logs for hosts and applications
  - Elasticsearch: a search and analytics engine to store logs
  - Fluentd: gathers logs and sends to Elasticsearch.
  - Kibana: A web UI for Elasticsearch.
- Access control
  - Cluster administrators can view all logs
  - Users can only view logs for their projects
  - Central Audit policy configuration
- API server events are automatically audited
- Logging pipelines collect API server and host audit logs as well as cluster and application logs for forwarding to the SIEM of your choice



Create Operator Subscription

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

Installation Mode *

○ All namesp

This mode rator

Operator will be available in a single namespace only.

● A specific namespace on the cluster

Operator will be available in a single namespace only.

PR openshift-logging

Update Channel *

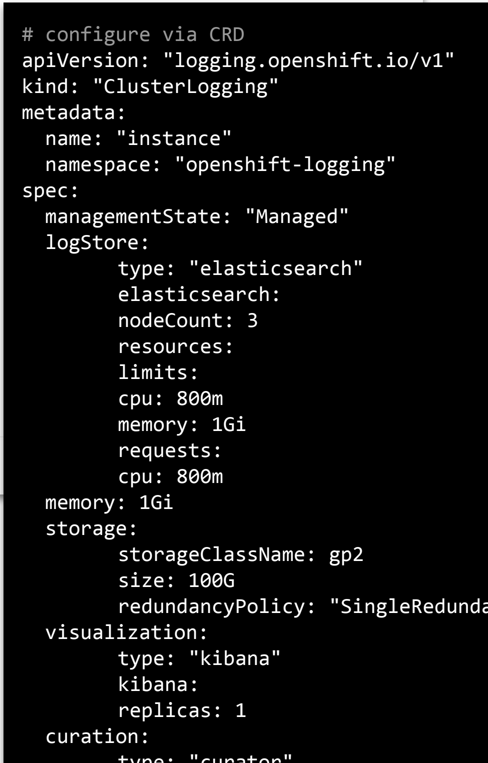
● preview

Approval Strategy *

● Automatic

○ Manual

Subscribe
Cancel

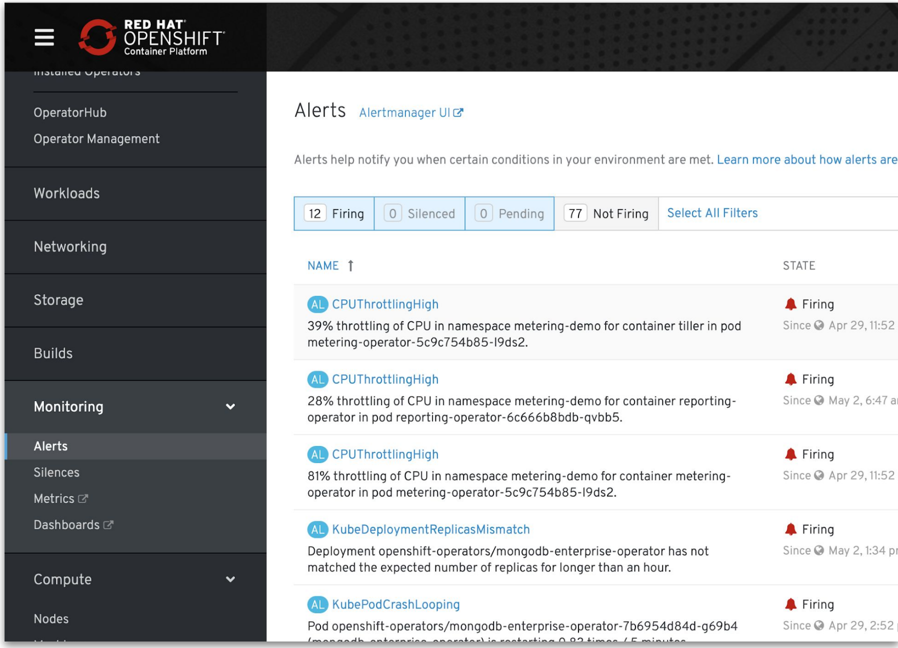


```
# configure via CRD
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
        type: "elasticsearch"
        elasticsearch:
        nodeCount: 3
        resources:
        limits: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

Cluster monitoring

Cluster monitoring is installed by default

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



Alerts Alertmanager UI

Alerts help notify you when certain conditions in your environment are met. Learn more about how alerts are

12 Firing | 0 Silenced | 0 Pending | 77 Not Firing | Select All Filters

NAME	STATE
AL CPUThrottlingHigh 39% throttling of CPU in namespace metering-demo for container tiller in pod metering-operator-5c9c754b85-l9ds2.	Firing Since Apr 29, 11:52
AL CPUThrottlingHigh 28% throttling of CPU in namespace metering-demo for container reporting-operator in pod reporting-operator-6c666b8bdb-qvbb5.	Firing Since May 2, 6:47 a
AL CPUThrottlingHigh 81% throttling of CPU in namespace metering-demo for container metering-operator in pod metering-operator-5c9c754b85-l9ds2.	Firing Since Apr 29, 11:52
AL KubeDeploymentReplicasMismatch Deployment openshift-operators/mongodb-enterprise-operator has not matched the expected number of replicas for longer than an hour.	Firing Since May 2, 1:34 p
AL KubePodCrashLooping Pod openshift-operators/mongodb-enterprise-operator-7b6954d84d-g69b4	Firing Since Apr 29, 2:52

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

HAPROXY 2.0

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"}}}}}}'
```

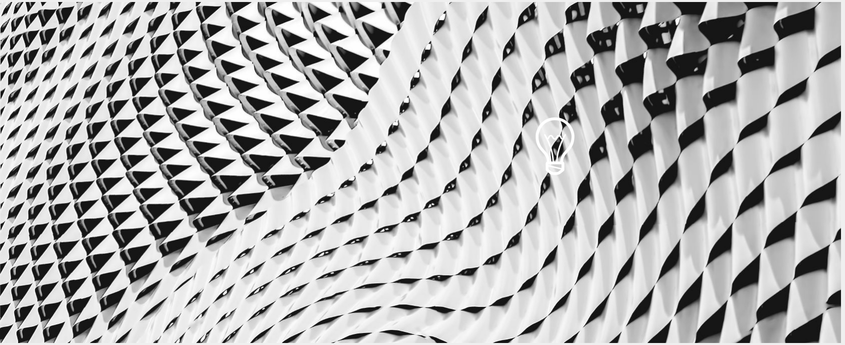
View the Logs

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

Auditd

- Low level system wide auditing system
- Integrated in Kernel and userspace - no security event escapes!
- Very detailed feed that meets all existing compliance standards
- Actively used by customers that need to adhere to tight security practices
- Auditd is included in RHEL CoreOS
- Host level audit logs are collected for forwarding by the OpenShift Logging Pipelines feature

SUMMARY



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

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.

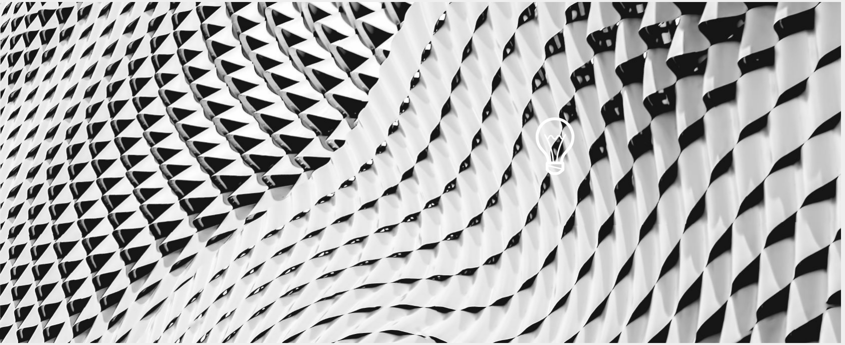
in linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

f facebook.com/redhatinc

twitter.com/RedHat

DEFEND INFRASTRUCTURE



Day 2 Configuration

Global Configuration

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

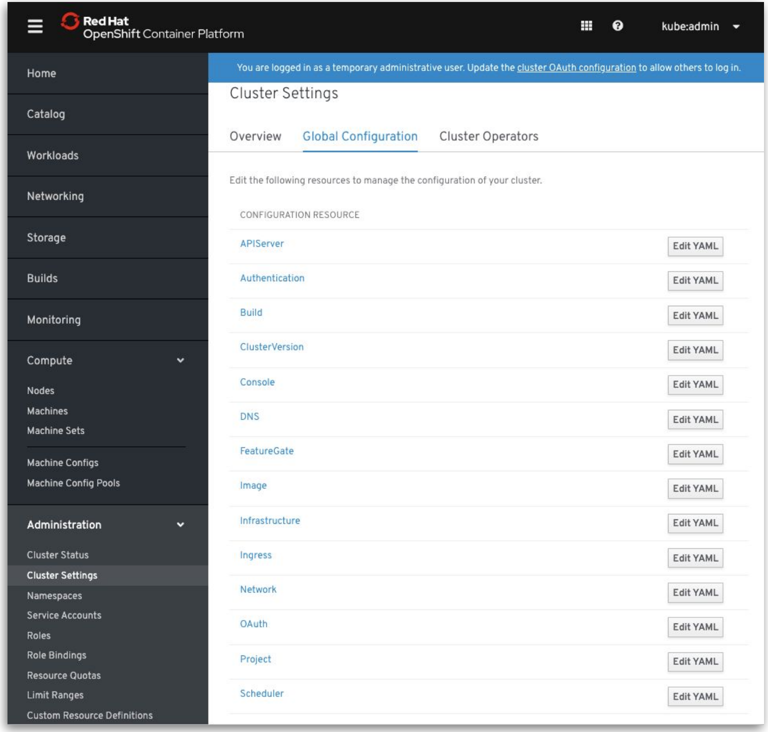
Change via Cluster Settings screen

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



You are logged in as a temporary administrative user. Update the cluster OAuth configuration to allow others to log in.

Cluster Settings

Edit the following resources to manage the configuration of your cluster.

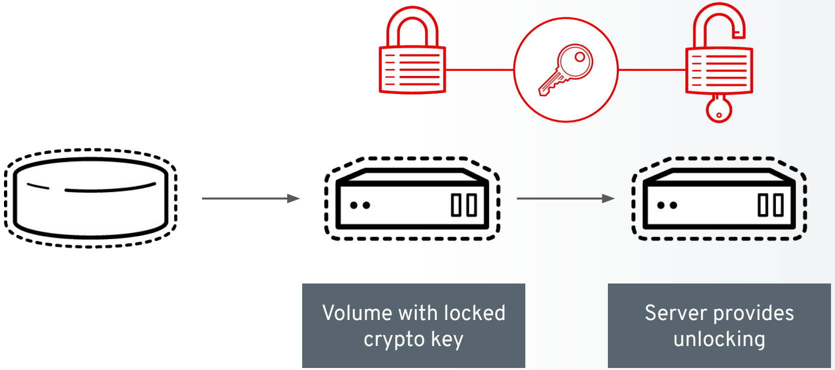
CONFIGURATION RESOURCE	
APIServer	Edit YAML
Authentication	Edit YAML
Build	Edit YAML
ClusterVersion	Edit YAML
Console	Edit YAML
DNS	Edit YAML
FeatureGate	Edit YAML
Image	Edit YAML
Infrastructure	Edit YAML
Ingress	Edit YAML
Network	Edit YAML
OAuth	Edit YAML
Project	Edit YAML
Scheduler	Edit YAML

Generally Available

Attached storage

Secure storage by using

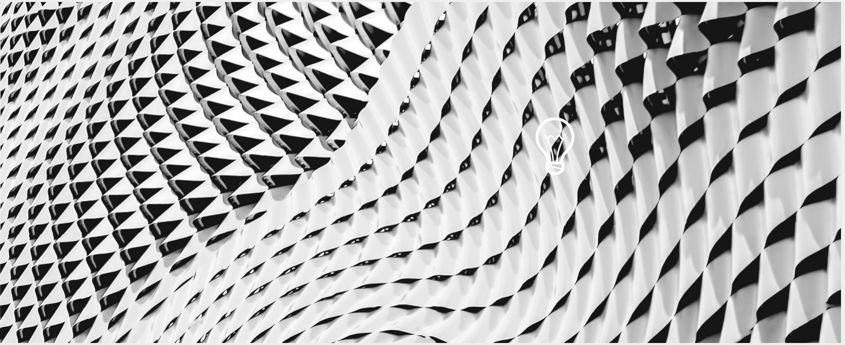
- SELinux access controls
- Secure mounts
- Supplemental group IDs for shared storage
- Network bound disk encryption



Volume with locked crypto key

Server provides unlocking

EXTEND SECURITY



The Security Ecosystem

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

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

Red Hat Advanced Cluster Management

Red Hat Advanced Cluster Management for Kubernetes

Robust, proven, award-winning

Multicluster life-cycle management

Policy-driven governance, risk, and compliance

Advanced application life-cycle management

Advanced Cluster Management

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

Policy-Based Governance
Enforce 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

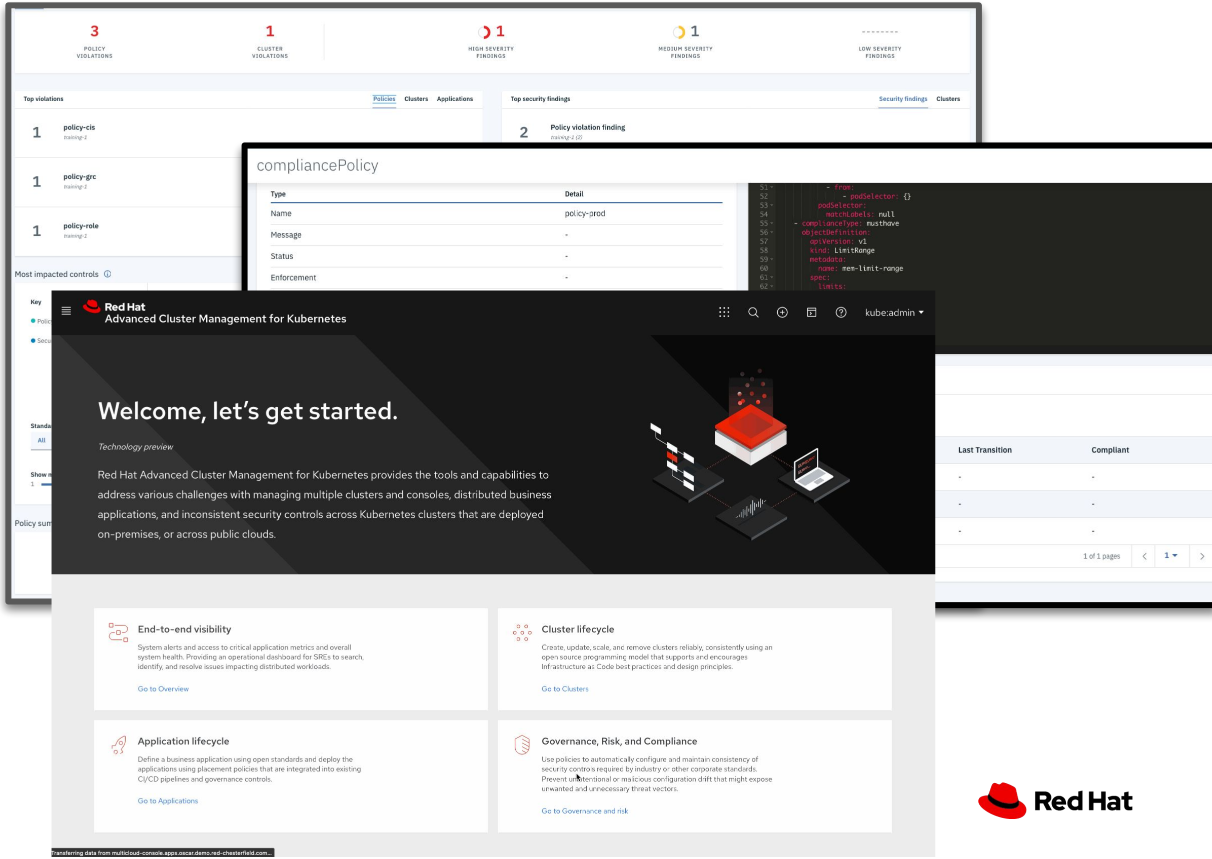
Event Management

Existing Tools & Processes

Security & Compliance Management

Multi-Cluster Management and Security with Red Hat Advanced Cluster Management for Kubernetes

- Centrally set & enforce policies for security, applications, & infrastructure
- Quickly visualize detailed auditing on configuration of apps and clusters
- Built-in CIS compliance policies and audit checks
- Immediate visibility into your compliance posture based on your defined standards



1	policy-cis *training-1*
1	policy-grc *training-1*
1	policy-role *training-1*

2	Policy violation finding *training-1 (2)*

compliancePolicy

Type	Detail
Name	policy-prod
Message	-
Status	-
Enforcement	-

```
              - from:
                  - podSelector: {}
                podSelector:
                  matchLabels: null
        - complianceType: musthave
          objectDefinition:
            apiVersion: v1
            kind: LimitRange
            metadata:
              name: mem-limit-range
            spec:
              limits:
```

Welcome, let’s get started.

Technology preview

Red Hat Advanced Cluster Management for Kubernetes provides the tools and capabilities to address various challenges with managing multiple clusters and consoles, distributed business applications, and inconsistent security controls across Kubernetes clusters that are deployed on-premises, or across public clouds.

End-to-end visibility

System alerts and access to critical application metrics and overall system health. Providing an operational dashboard for SREs to search, identify, and resolve issues impacting distributed workloads.

Go to Overview

Cluster lifecycle

Create, update, scale, and remove clusters reliably, consistently using an open source programming model that supports and encourages Infrastructure as Code best practices and design principles.

Go to Clusters

Application lifecycle

Define a business application using open standards and deploy the applications using placement policies that are integrated into existing CI/CD pipelines and governance controls.

Go to Applications

Governance, Risk, and Compliance

Use policies to automatically configure and maintain consistency of security controls required by industry or other corporate standards. Prevent unintentional or malicious configuration drift that might expose unwanted and unnecessary threat vectors.

Go to Governance and risk

Transferring data from multicloud-console.apps.oscar.demo.red-chesterfield.com...

Appendix: External hybrid cloud security guidance

Securing Kubernetes

Guidance from the CNCF Kubernetes Security Audit

“ While Kubernetes facilitates high-availability workload deployments, 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, and affects the behavior of an operator’s composed objects.

With this in mind, the options available for configuring components of Kubernetes often fluctuate significantly in supported versions, and vary in their approach to default settings. This leads to a non-trivial amount of configuration required by an administrator to stand-up a functional cluster for a given workload.

More effort must then be spent maintaining the cluster to abide by these settings, especially when planning and executing upgrades of Kubernetes components.”

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

Securing the container host

Guidance from NIST

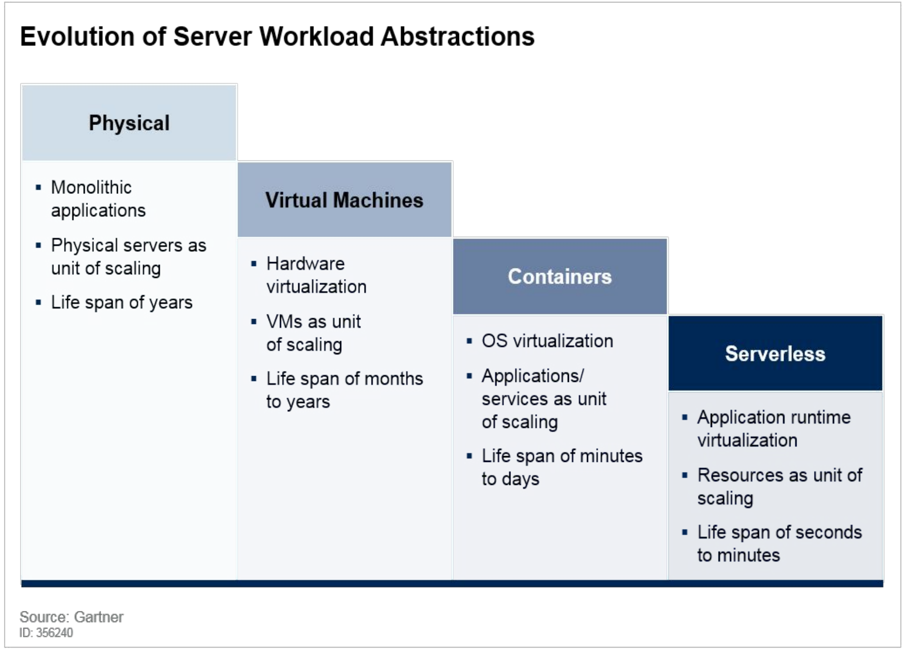
Use container-specific host OSs instead of general-purpose ones to reduce attack surfaces.

A container-specific host OS is a minimalist OS explicitly designed to only run containers, with all other services and functionality disabled, and with read-only file systems and other hardening practices employed. When using a container-specific host OS, attack surfaces are typically much smaller than they would be with a general-purpose host OS, so there are fewer opportunities to attack and compromise a container-specific host OS. Accordingly, whenever possible, organizations should use container-specific host OSs to reduce their risk.

NIST Special Publication 800-190
Application Container Security Guide

Securing cloud native workloads

Guidance from Gartner



Evolution of Server Workload Abstractions

Physical

- Monolithic applications
- Physical servers as unit of scaling
- Life span of years

Virtual Machines

- Hardware virtualization
- VMs as unit of scaling
- Life span of months to years

Containers

- OS virtualization
- Applications/ services as unit of scaling
- Life span of minutes to days

Serverless

- Application runtime virtualization
- Resources as unit of scaling
- Life span of seconds to minutes

Source: Gartner
ID: 356240

"The best way to secure these rapidly changing and short-lived workloads is to start their protection proactively in the development phase … so that when a workload is instantiated in production, it is "born" protected."

"Replace antivirus (AV)-centric strategies with a "zero-trust execution"/default deny/application control approach to workload protection where possible…." [1]

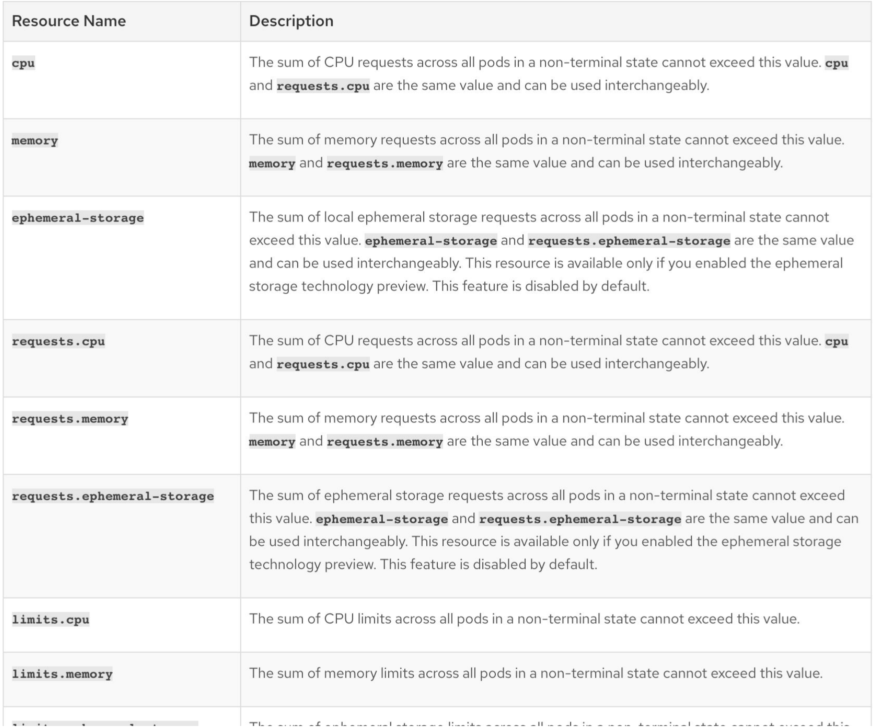
[1] Gartner: Market Guide for Cloud Workload Protection Platforms, ID G00356240, April 8, 2019

Appendix: Resource management

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.
`limits.ephemeral-storage`	The sum of ephemeral storage limits across all pods in a non-terminal state cannot exceed this

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

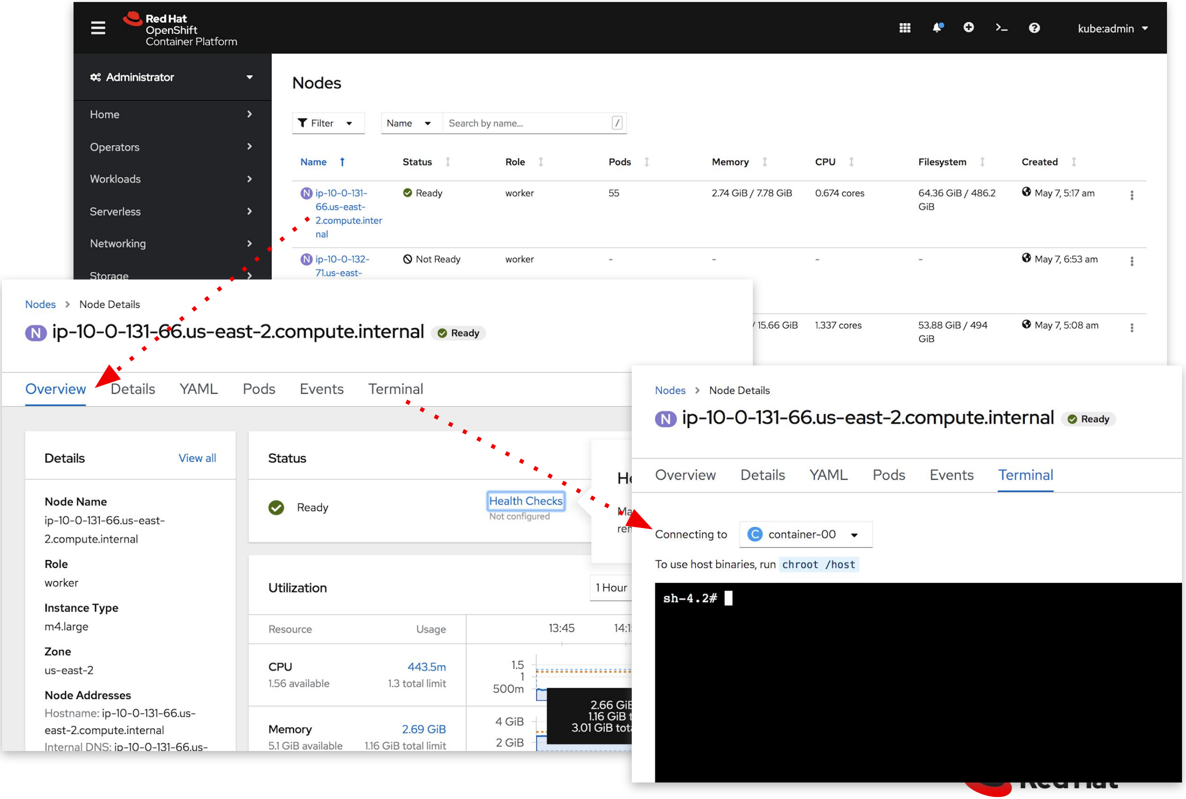
Install the descheduler via OperatorHub to the openshift-kube-descheduler-operator namespace and Create a descheduler instance

```
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)



Name	Status	Role	Pods	Memory	CPU	Filesystem	Created
ip-10-0-131-66.us-east-2.compute.internal	Ready	worker	55	2.74 GiB / 7.78 GiB	0.674 cores	64.36 GiB / 486.2 GiB	May 7, 5:17 am
ip-10-0-132-71.us-east-	Not Ready	worker	-	-	-	-	May 7, 6:53 am
				/ 15.66 GiB	1.337 cores	53.88 GiB / 494 GiB	May 7, 5:08 am

Node Name
ip-10-0-131-66.us-east-2.compute.internal

Role
worker

Instance Type
m4.large

Zone
us-east-2

Node Addresses
Hostname: ip-10-0-131-66.us-east-2.compute.internal
Internal DNS: ip-10-0-131-66.us-

Status
Ready
Health Checks
Not configured

Utilization
1 Hour

Resource	Usage
CPU 1.56 available	443.5m 1.3 total limit
Memory 5.1 GiB available	2.69 GiB 1.16 GiB total limit

Connecting to container-00

To use host binaries, run `chroot /host`

```
sh-4.2#
```

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

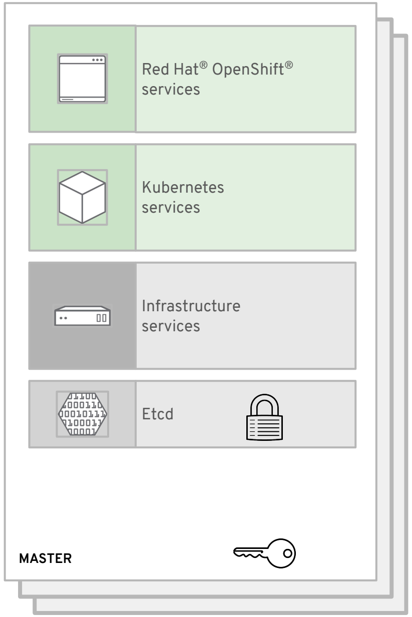
```
apiVersion: autoscaling.openshift.io/v1
kind: VerticalPodAutoscalerController
metadata:
 name: default
spec:
 safetyMarginFraction: 0.15
 podMinCPUMillicores: 25
 podMinMemoryMb: 250
```

Appendix: Etcd encryption & cipher suites

OpenShift 4 etcd Encryption

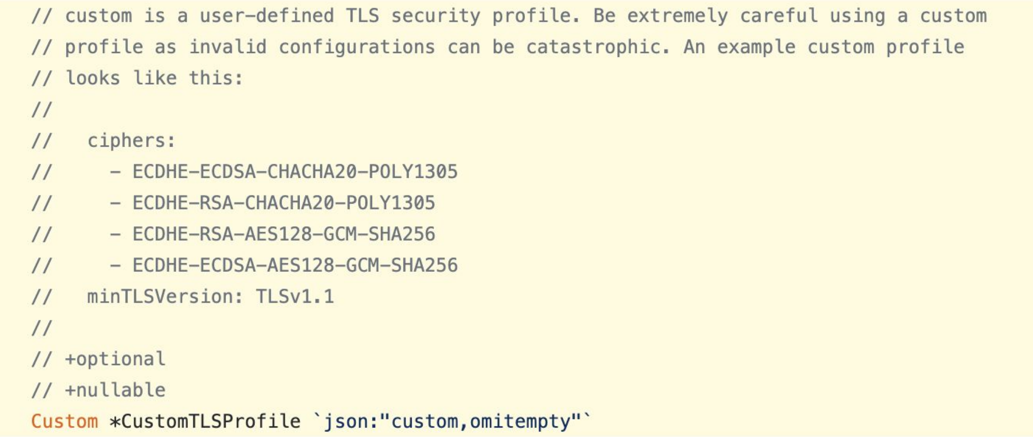
Encrypt secrets, config maps...

- 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.



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"`
```