Sécurité des containers & Openshift

Workshop 1 - 28 Novembre 2019

Red Hat

Red Hat TEAM pour EDF

Wafa ANTAR

Key Account Manager
wantar@redhat.com
Mob.: +33 6 23 72 44 16

Laurent-Xavier MURGIER

Sales Business Manager - Red Hat Storage & Open HCI
lmurgier@redhat.com
Mob.: +33 6 33 25 40 85

Yuliya YEROKHINA

Account Executive - ENERGY
yyerokhi@redhat.com
Mob.: +33 6 47 46 82 33

Luc PIORO

Account Solutions Architect EDF
lpioro@redhat.com
Mob.: +33 6 32 18 32 67

Jaafar CHRAIBI

Principal Solutions Architect AppDev
jchraibi@redhat.com
Mob.: +33 7 76 88 95 16

Jérôme BRUSQ

EMEA Solutions Architect Storage
jbrusq@redhat.com
Mob.: +33 6 77 34 31 81

David CLAUVEL

Solution Architect Ansible Automation
dclauvel@redhat.com

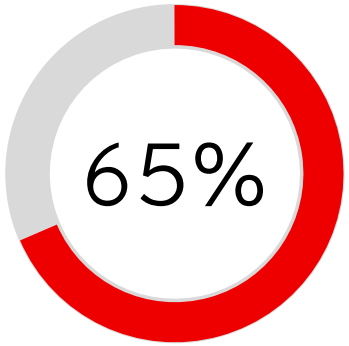
Ordre du jour - Workshop 1 - Echanges sur les solutions de sécurité proposée par Redhat

- *Introduction aux principes de sécurité d'une plateforme de containers*
- *Mécanismes sécurité Linux utilisées (cgroups, règles SCC, SELinux, etc.)*
- *Présentation de Red Hat CoreOS - socle "immutable" - Réduction de la surface d'attaque*
- *Gestion des projets (Namespace) - Multi-tenancy - RBAC (Rôles utilisateurs)*
- *Gestion des identités - introduction RH SSO pour les cas d'usage SAML et Synchro AD*
- *Gestion des certificats*
- *Gestion des secrets (intégration possible Vault)*
- *Gestion des Logs - Intégration SIEM*
- *Sécurisation de la Registry + Images certifiées + Scan Images + gestion des droits d'accès (push/pull image)*
- *Sécurisation du Control Plane - APIs*
- *Software Defined Network - SDN - Isolation Réseau - Egress - Micro-Segmentation*
- *Architecture DMZ - Isolated Zones*
- *Stockage persistent des containeurs - sécurisation des accès*
- *Information sur les certifications (FISMA ISO27000 PCI-DSS) - Hardening*

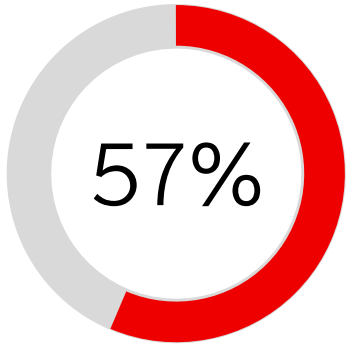
Ordre du jour - Workshop 2 - Echanges sur les solutions de sécurité proposée par Redhat

- *Présentation de Red Hat CoreOS - socle "immutable" - Réduction de la surface d'attaque*
- *Installation de la plateforme Openshift en mode déconnecté*
- *CI/CD - Pipeline & Sécurité (Jenkins, Git, etc.)*
- *Suivi de la conformité (OpenSCAP)*
- *Mécanisme AUDIT*
- *Alerting proactif sur les risques par rapport aux vulnérabilités (CVE)*
- *Application MicroServices - Service Mesh (ISTIO) - Gestion du trafic et Policies*
- *Architecture DMZ - Isolated Zones*
- *Lien avec API Management*
- *Openshift & Public Cloud - RETEX architecture*
- *Ecosystème Sécurité K8S*

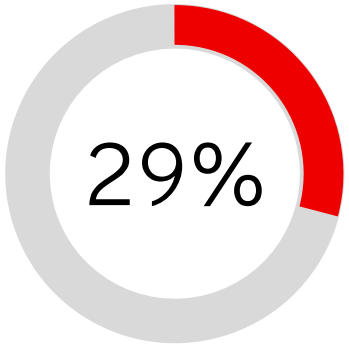
The Cyber Security Challenge is not Getting Easier



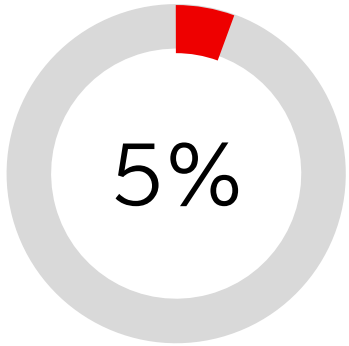
Reported increased Severity of attacks



Said the time to resolve an incident has grown



Have their ideal security-skilled staffing level, making it the #2 barrier to Cyber resilience

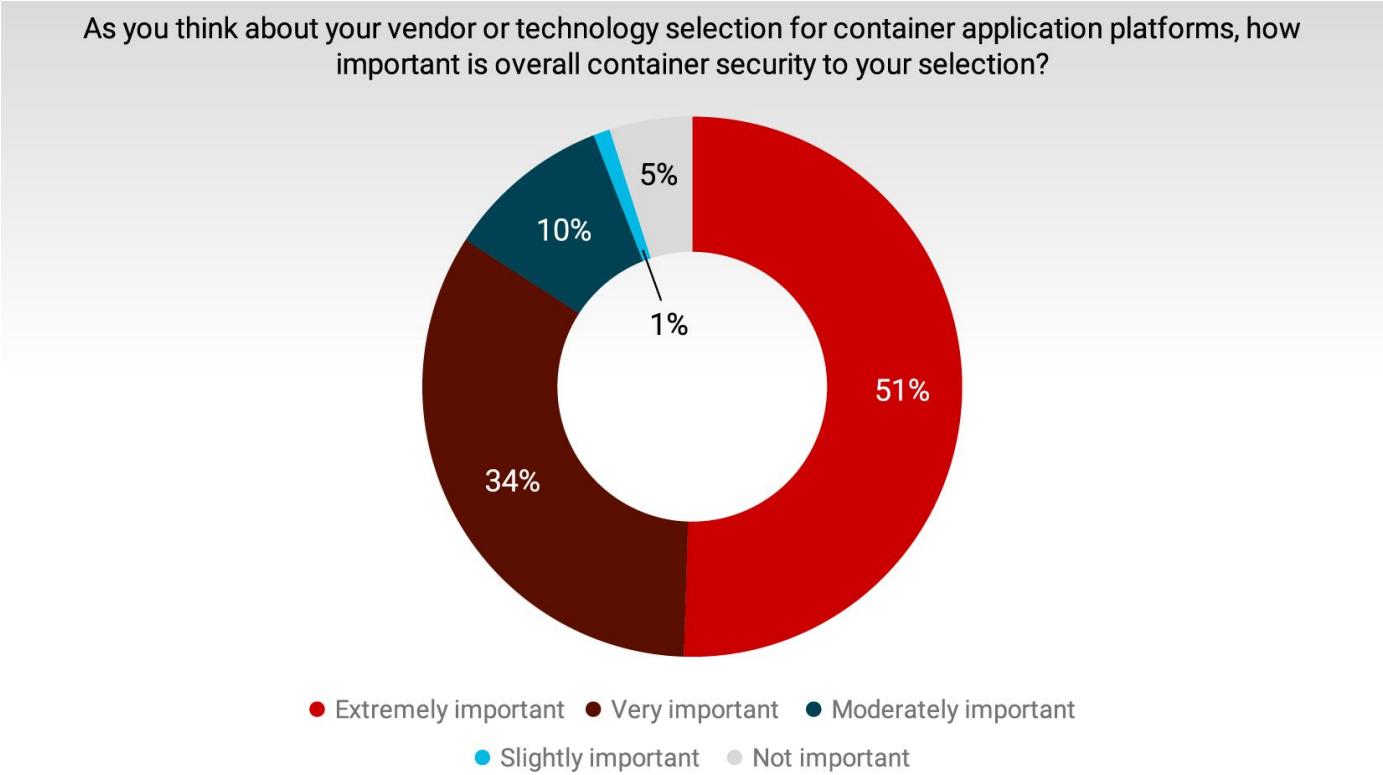


Portio of alerts coming in that the average security team examines every day

[1] The Third Annual Study on the Cyber Resilient Organization - Ponemon Institute, 2018 (Sponsored by IBM)
[2] https://venturebeat.com/2017/12/16/the-lesson-behind-2017s-biggest-enterprise-security-story/

Red Hat

Choosing a Container Platform Vendor? Security is Critical



Source: Global IT Trends & Priorities Research, November 2018, Qualtrics and Red Hat

Red Hat

Hackers exploit Jenkins servers, make $3 million by mining bitcoin

Hackers exploiting Jenkins servers made $3 million in one of the biggest malicious cryptocurrency mining operations ever.

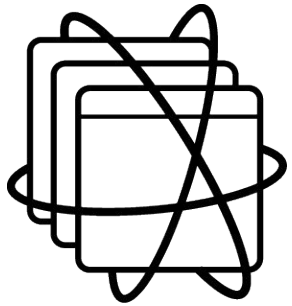
A hacker group has made over $3 million by breaking into Jenkins servers and installing malware that mines the Monero cryptocurrency.

Hackers are targeted Jenkins, a continuous integration/deployment web application built in Java that allows dev teams to run automated tests and execute various operations based on test results, including deploying new code to production servers. Because of this, Jenkins servers are extremely popular with both freelance web developers, but also with large enterprises.

Well over 100,000 installs worldwide……..

Common OpenShift Security Related Customer Concerns

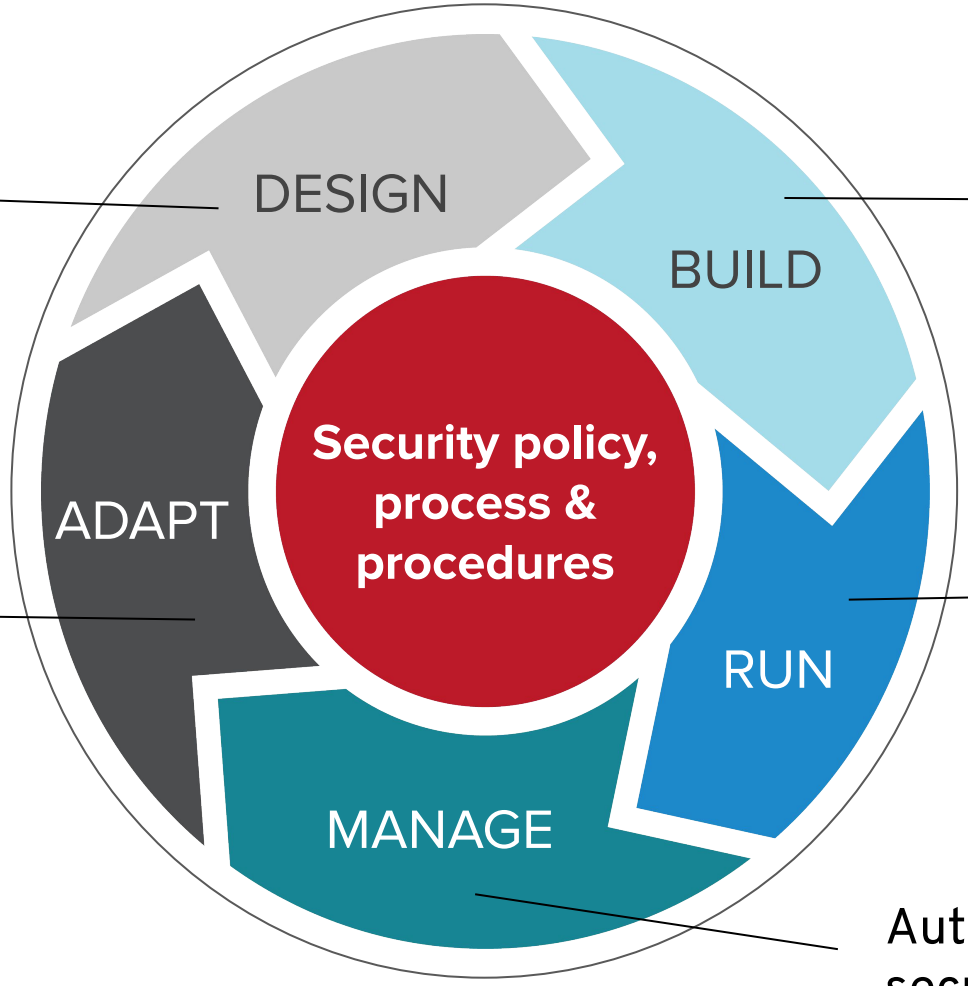
What are the typical questions we hear from our customers?



- Are containers secure?
- Host security
- Audit and Logging
- Network security
- Certificates
- Application secrets management
- Image scanning and signing
- When to use 3rd party OpenShift security vendors and who to pick
- OpenShift compliance to security standards
- Other OpenShift security concerns

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

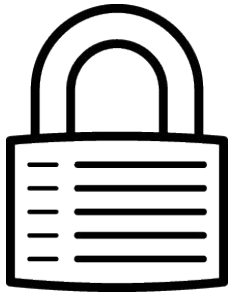
Automate systems for security & compliance

Revise, update, remediate as the landscape changes

Red Hat

COMPREHENSIVE CONTAINER SECURITY

OPENSHIFT



CONTROL
Application Security

- Container Content
- CI/CD Pipeline
- Container Registry
- Deployment Policies

DEFEND
Infrastructure

- Container Platform
- Container Host Multi-tenancy
- Network Isolation
- Storage
- Audit & Logging
- API Management

EXTEND

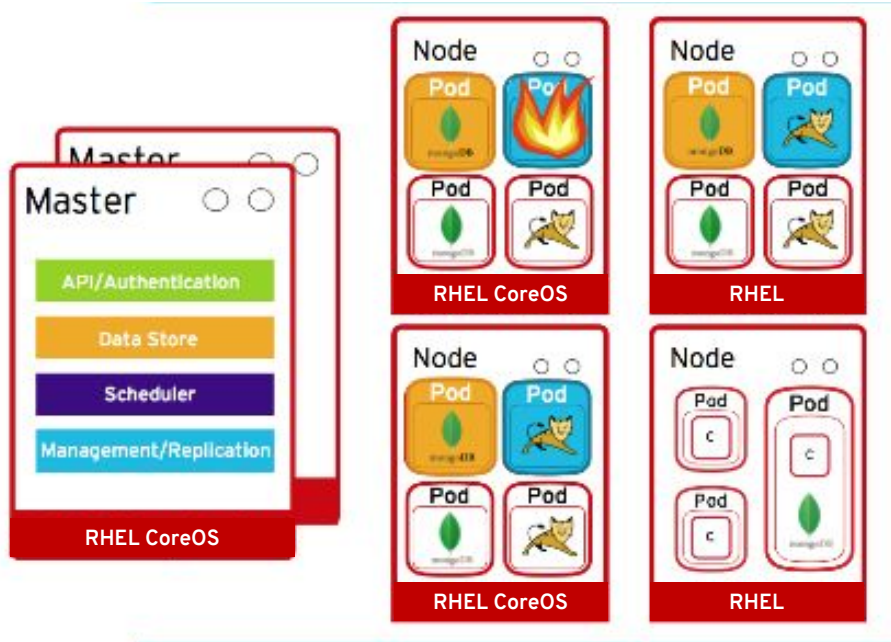
- Security Ecosystem

Red Hat

SECURING THE CONTAINER PLATFORM

Security Features Include

- Host & Runtime security
- Identity and Access Management
- Role-based Access Controls
- Project namespaces
- Integrated SDN - Network Policies is default
- Integrated & extensible secrets management
- Logging, Monitoring, Metrics



COMPREHENSIVE SECURITY FEATURES

DEFENSE IN DEPTH

OPENSHIFT

Linux Host Security
- SELinux+
- Common Criteria Certification
- FIPS mode available

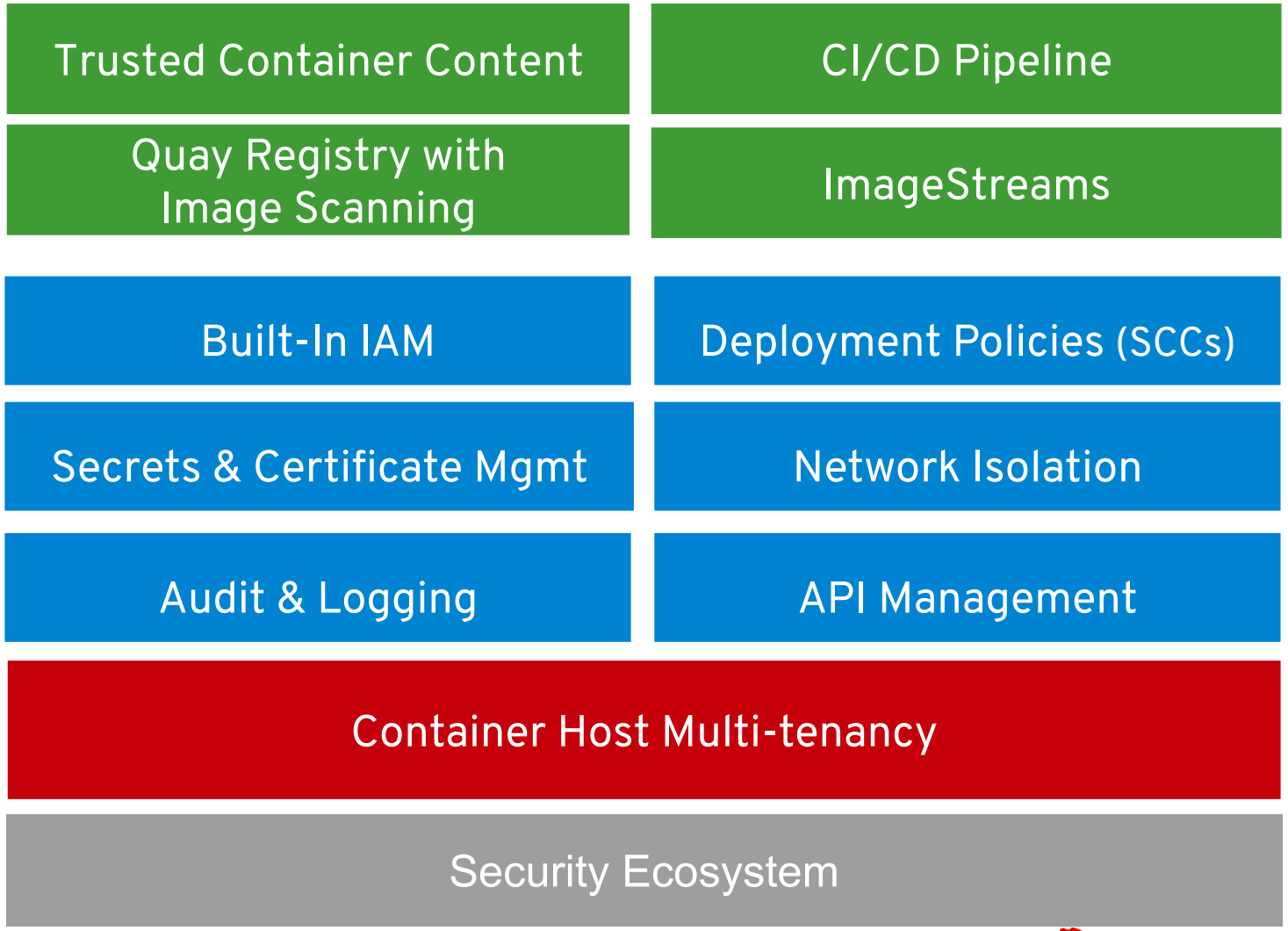
Authentication & Authorization
- Embedded OAuth Server
- Supports 9 Identity Providers including AD/LDAP
- Multi-Level Access Control (Users and Groups)
- Secrets and certificate management

Image Security
- ImageStreams
- Scanning
- Deployment policies

Integrated Audit, Logging, Monitoring

Security Policies
- SCC (Security Context Controls)
- Non-Root Containers
- Controlled Access to Resources

Networking Isolation
- Ingress / Egress control
- Network microsegmentation



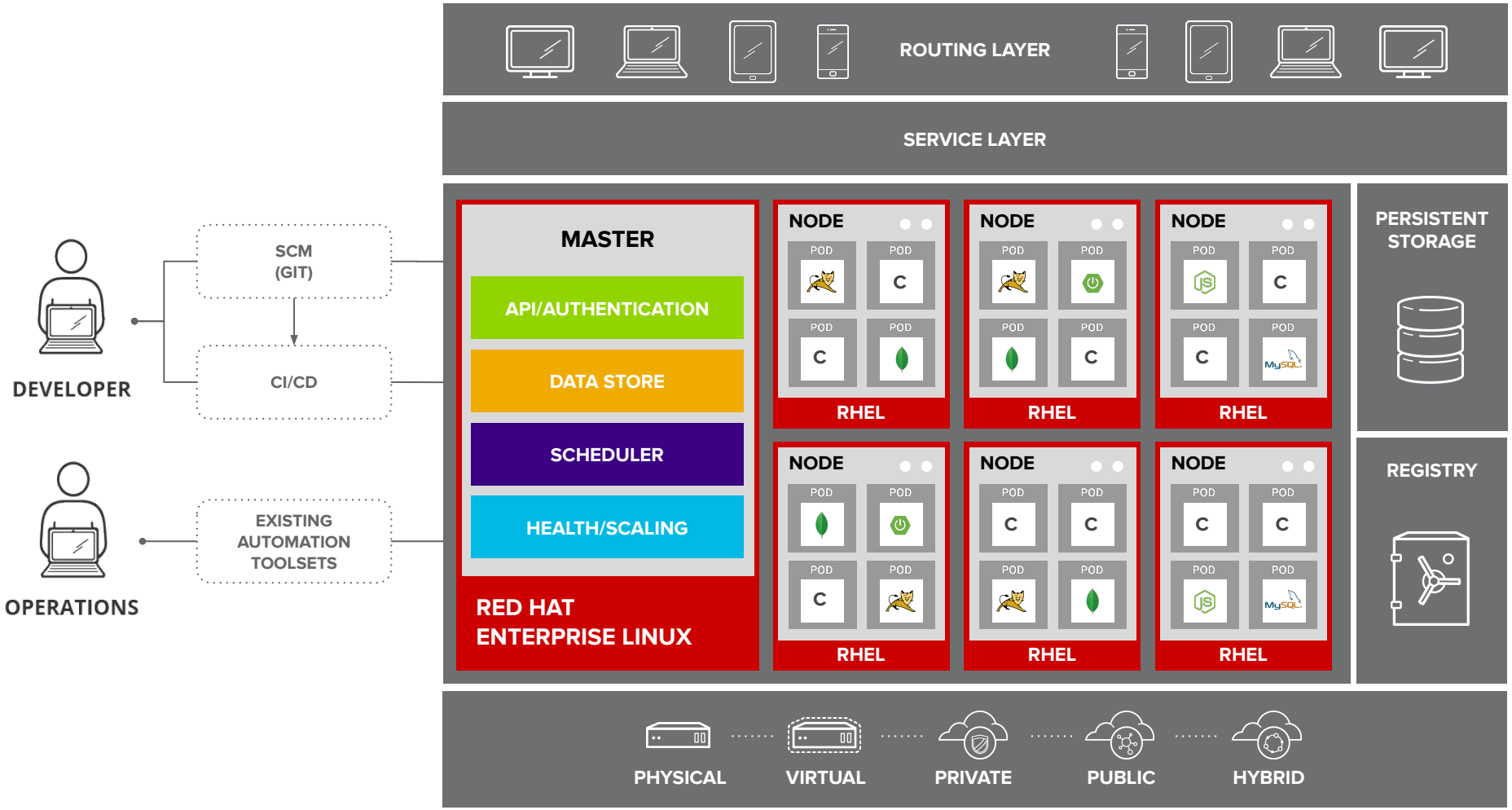
Ten Layers of Container Security

Red Hat

Rappel : Openshift Architecture
“High level”

OPENSHIFT ARCHITECTURE (Overview)

OPENSHIFT



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]

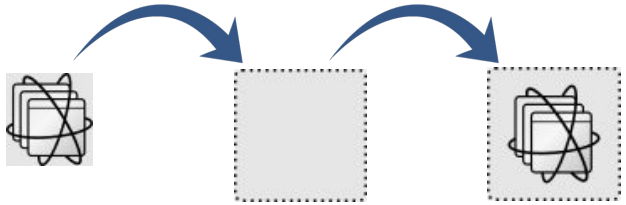
NEXT-GEN CONTAINER TOOLS

Providing stability, flexibility and performance with containers and images

Container-tools - OCI tooling to create, run, and manage, Linux Containers with an enterprise life cycle.

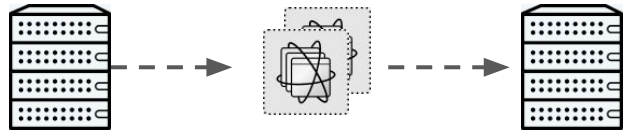
- Conform to the OCI image and runtime specifications
- Daemon-less, OS-native container tooling
- Separation of concerns

buildah



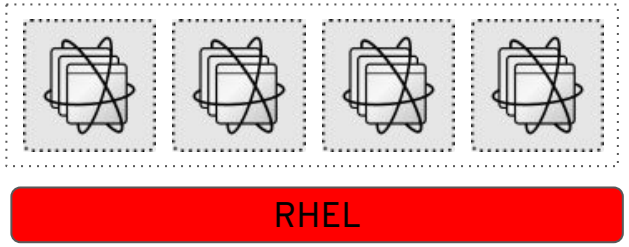
Build OCI/docker Images

skopeo



Inspect, copy, & sign Images

podman



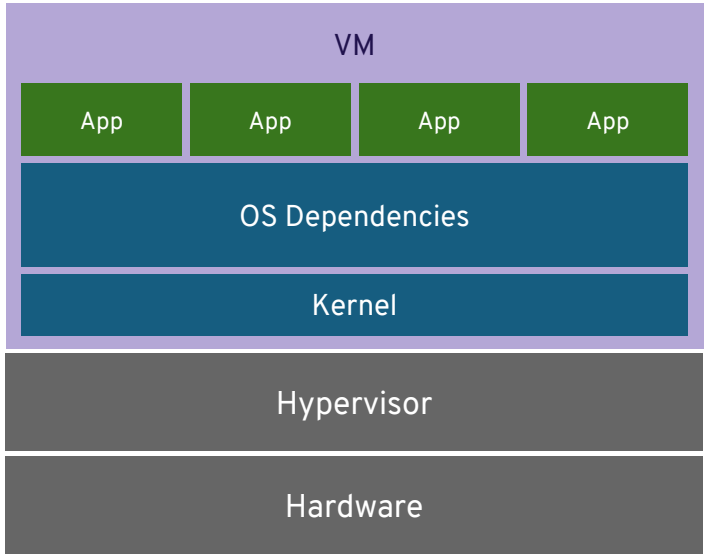
run, manage, debug containers

Red Hat

Mécanismes sécurité Linux utilisées (cgroups, règles SCC, SELinux, etc.)

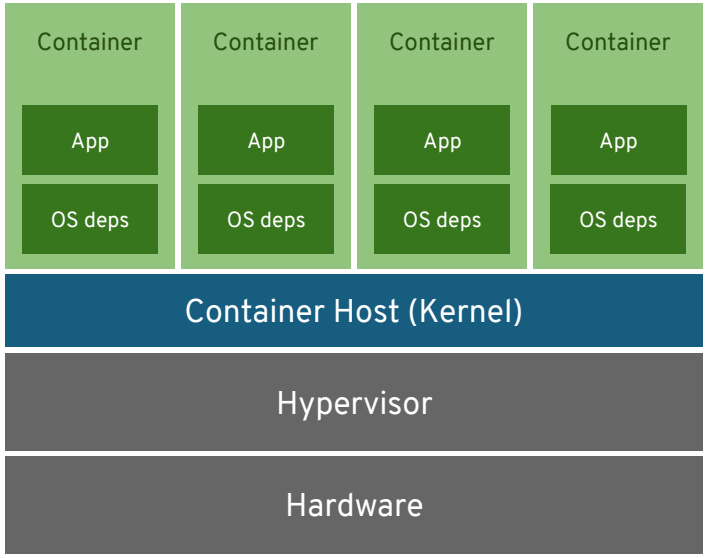
VIRTUAL MACHINES vs CONTAINERS

VIRTUAL MACHINES



VM isolates the hardware

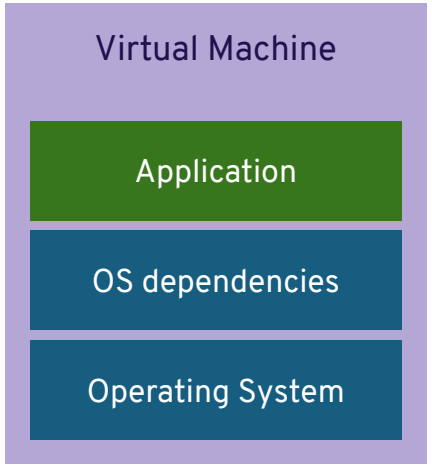
CONTAINERS

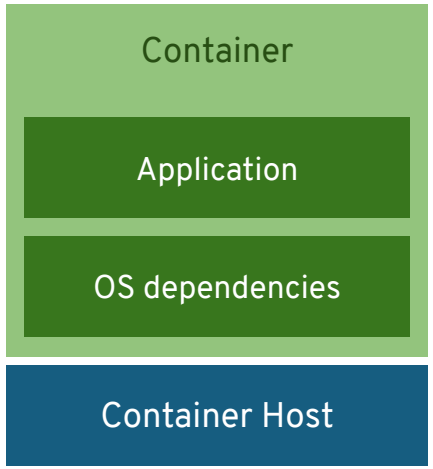


Container isolates the process

Red Hat

VIRTUAL MACHINES vs CONTAINERS





- ✚ VM Isolation
- ▬ Complete OS
- ▬ Static Compute
- ▬ Static Memory
- ▬ High Resource Usage

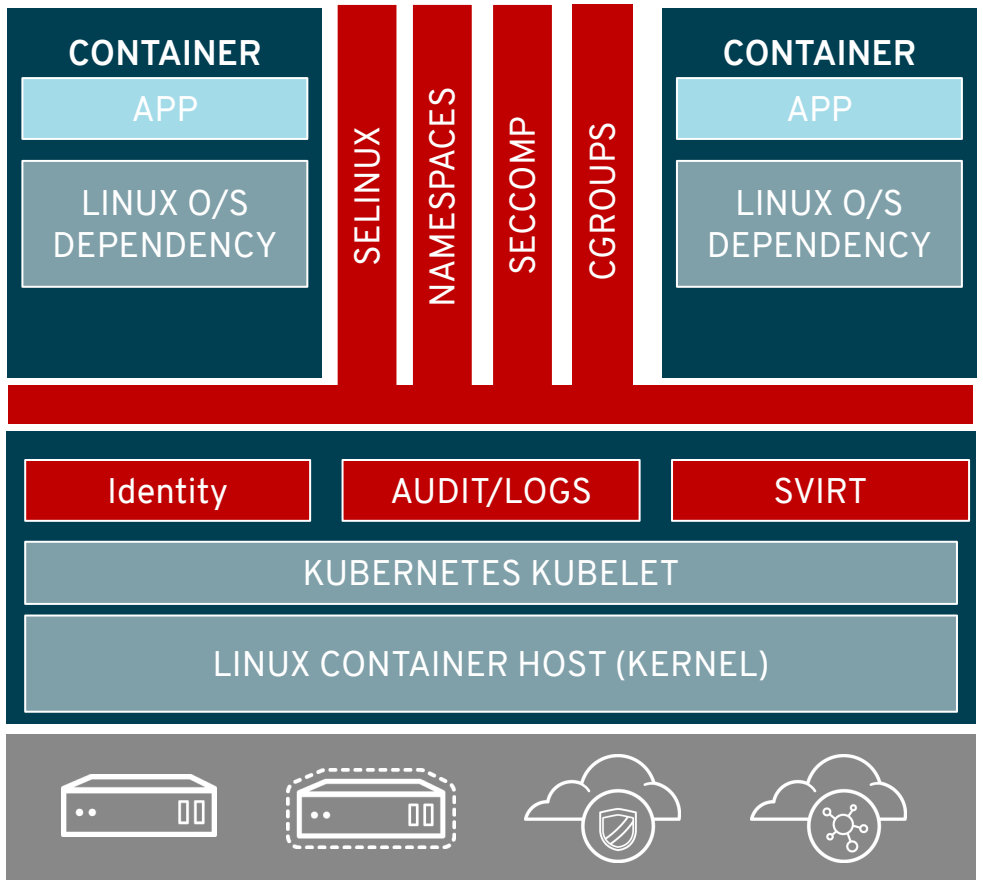
- ✚ Container Isolation
- ✚ Shared Kernel
- ✚ Burstable Compute
- ✚ Burstable Memory
- ✚ Low Resource Usage

Red Hat

HOST OS CONTAINER MULTI-TENANCY

Container Security starts with Linux Security

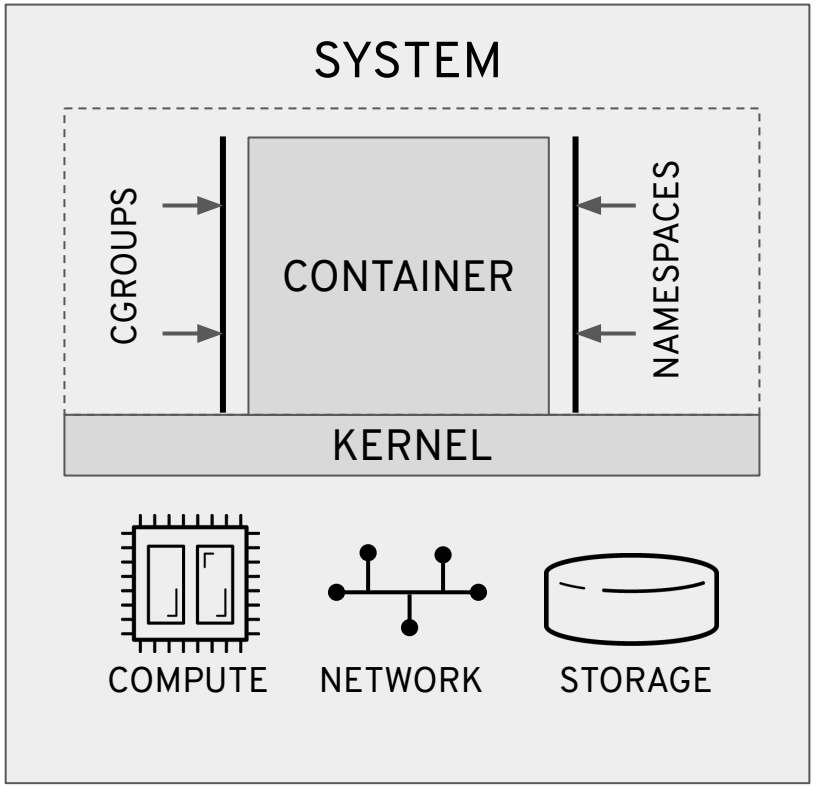
- Security in the RHEL host applies to the container
- SELINUX and Kernel Namespaces are the one-two punch no one can beat
- Protects not only the host, but containers from each other
- RHEL CoreOS provides minimized attack surface
- Common Criteria certification



Red Hat

WHAT IS A CONTAINER?

- Isolation on OS level
- Technically speaking, a **process**, running with the privileges of its parent process (user process), but i**solated by kernel features** (Cgroups, Namespaces[1])
- Therefore, a container is - surprisingly - already defined as a **security artefact** on its own



[1] Applies to Linux and Docker containers only, Windows containers use different mechanisms

Red Hat

FOCUS: CGROUPS

- Kernel Control Groups (cgroups), allow sysadmins to have **fine-grain control of the resources** available to an application, user or group.

- You can control access to **system memory**, **CPU time**, **network access**, and **block devices**.

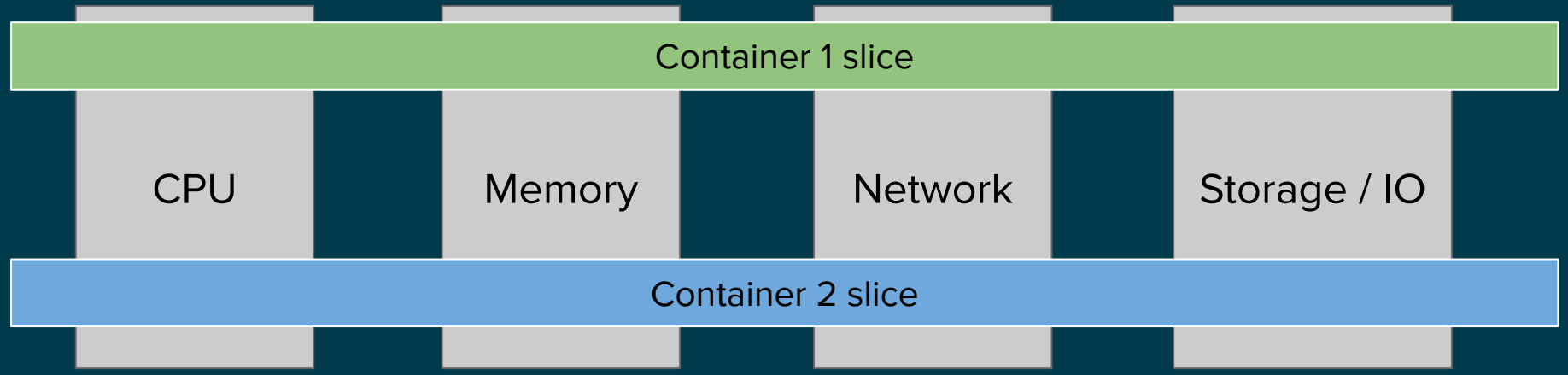
- Since this feature controls which device nodes can be created within a namespace and allows processes to configure the kernel it needs to be properly secured in order to maintain container isolation.

To accomplish this aspect OpenShift will only allow access to the following locations by default:

/dev/console /dev/zero /dev/null /dev/fuse /dev/full /dev/tty /dev/urandom /dev/random

Without this constraint an exploit might allow an attacker to gain control of the host storage layer.

CGROUPS - Resource Isolation



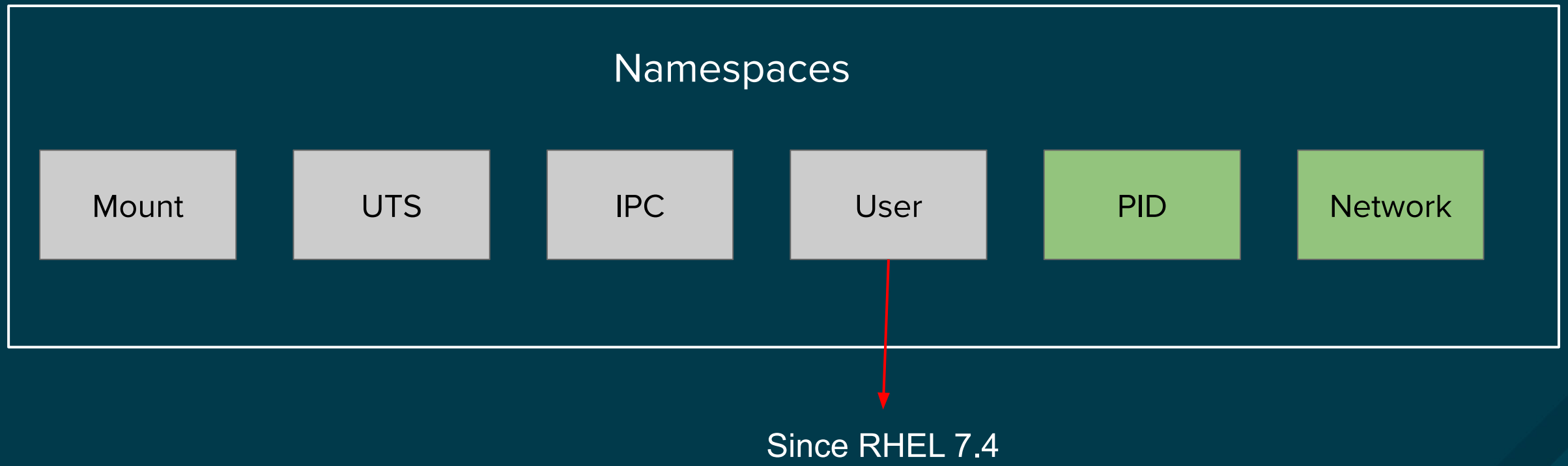
FOCUS: NAMESPACES

Namespaces provide most **“isolation”** to an application running in a container.

We are concerned with 6 namespaces:

- **Mount Namespace** - get your own mount point
- **UTS Namespace** - get own domain name and hostname
- **Network Namespace** - have your very own network
- **IPC Namespace** - how about memory isolation?
- **PID Namespace** - count your own IDs
- **User Namespaces** - separate IDs in containers from those in the host system

NAMESPACES
Process Isolation



FOCUS: LINUX CAPABILITIES

Beginning with kernel 2.2, Linux divides the privileges traditionally associated with superuser into 32 distinct units, known as capabilities, which can be independently enabled and disabled.

Since capabilities are a per-thread attribute you can dictate to your containers what is allowable and what should be restricted.

For instance the ping function only needs to have socket access capability and an Apache server may only need have bind access to ports < 1024, but these functions do not require all of root's
many capabilities.

By limiting these capabilities you can restrict the containers access to the corresponding host.

CAPABILITIES - DROPPING PRIVILEGES

CAP_SETPCAP	Modify process capabilities
CAP_SYS_MODULE	Insert/Remove kernel modules
CAP_SYS_RAWIO	Modify Kernel Memory
CAP_SYS_PACCT	Configure process accounting
CAP_SYS_NICE	Modify Priority of processes
CAP_SYS_RESOURCE	Override Resource Limits
CAP_SYS_TIME	Modify the system clock
CAP_SYS_TTY_CONFIG	Configure tty devices
CAP_AUDIT_WRITE	Write the audit log
CAP_AUDIT_CONTROL	Configure Audit Subsystem
CAP_MAC_OVERRIDE	Ignore Kernel MAC Policy
CAP_MAC_ADMIN	Configure MAC Configuration
CAP_SYSLOG	Modify Kernel printk behaviour
CAP_NET_ADMIN	Configure the network: - Setting the hostname/domainname - mount(),unmount() - nfsservctl - ....
CAP_SYS_ADMIN	

FOCUS: LINUX CAPABILITIES

OpenShift Container Platform have ***removed all of the above capabilities***
from root users running inside a container.

⇒ dramatically lowers the attack surface for containers and their applications.

Another example is that OpenShift takes the precaution of delivering giving read only mount points to the following locations (+ CAP_SYS_ADMIN capability remove)
/sys
/proc/sys
/proc/sysrg-trigger
/proc/irq
/proc/bus

CAPABILITIES - DROPPING PRIVILEGES

A root user inside a container running in OpenShift has none of the previous capabilities available!

```
{
        "defaultAction": "SCMP_ACT_ERRNO",
        "archMap": [
                {
                        "architecture": "SCMP_ARCH_X86_64",
                        "subArchitectures": [
                                "SCMP_ARCH_X86",
                                "SCMP_ARCH_X32"
                        ]
                },
                {
                        "architecture": "SCMP_ARCH_AARCH64",
                        "subArchitectures": [
                                "SCMP_ARCH_ARM"
                        ]
                },
                {
                        "architecture": "SCMP_ARCH_S390X",
                        "subArchitectures": [
                                "SCMP_ARCH_S390"
                        ]
                }
        ],
        "syscalls": [
                {
                        "names": [
                                "accept",
                                "accept4",
                                "access",
                                "alarm",
                                "alarm",
                                "bind",
```

FOCUS: SCC - SECURITY CONTEXT CONSTRAINTS

- In addition to RBAC ressources that control what a user can do, OpenShift provides security context constraints (SCC) that **control the actions that a pod can perform** and **what it has the ability to access**
- Prohibiting any container to get **access to unnecessary capabilities** or to **run in an insecure way** (e.g. privileged or as root)
- The OpenShift Container Application Platform provides a set of **predefined Security Context Constraints** (e.g. privileged or restricted) that can be used, modified or extended by any administrator
- By default, the execution of any container will be granted the **restricted** SCC and only the capabilities defined by that SCC

```
$ oc get scc
NAME               PRIV      CAPS      SELINUX     RUNASUSER          FSGROUP     SUPGROUP    PRIORITY
anyuid             false     []        MustRunAs   RunAsAny           RunAsAny    RunAsAny    10
hostaccess         false     []        MustRunAs   MustRunAsRange     MustRunAs   RunAsAny    <none>
hostmount-anyuid   false     []        MustRunAs   RunAsAny           RunAsAny    RunAsAny    <none>
hostnetwork        false     []        MustRunAs   MustRunAsRange     MustRunAs   MustRunAs   <none>
nonroot            false     []        MustRunAs   MustRunAsNonRoot   RunAsAny    RunAsAny    <none>
privileged         true      [*]       RunAsAny    RunAsAny           RunAsAny    RunAsAny    <none>
restricted         false     []        MustRunAs   MustRunAsRange     MustRunAs   RunAsAny    <none>
```

Red Hat

FOCUS: SECURITY CONTEXT CONSTRAINTS

SCCs are objects that define a set of conditions that a pod must run with in order to be accepted into the system.

They allow an administrator to control the following:

1. Running of privileged containers.
2. Capabilities a container can request to be added.
3. Use of host directories as volumes.
4. The SELinux context of the container.
5. The user ID.
6. The use of host namespaces and networking.
7. Allocating an **FSGroup** that owns the pod's volumes
8. Configuring allowable supplemental groups
9. Requiring the use of a read only root file system
10. Controlling the usage of volume types
11. Configuring allowable seccomp profiles

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
```

Red Hat

FOCUS: SELINUX

In a container world SELinux is important because it **separates containers from each other** - even when they are processes launched from the very same binary!

Security-Enhanced Linux (SELinux) is a Linux kernel security module that provides a mechanism for supporting access control security policies.

As a conceptual overview it is primarily a labeling system that assigns a label (name) to every process and system object. SELinux Context or Labels uses the format of *user::role::type::level*.

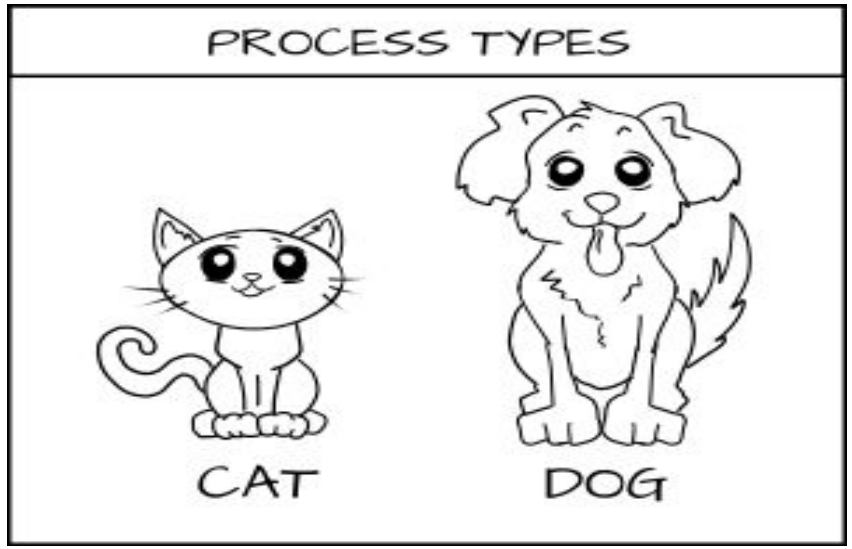
This allows every aspect of kernel operations to be first labeled, second classified, and then ultimately enforced by a set of rules that the provider maintains.

Policy rules control access between labeled processes and labeled objects.

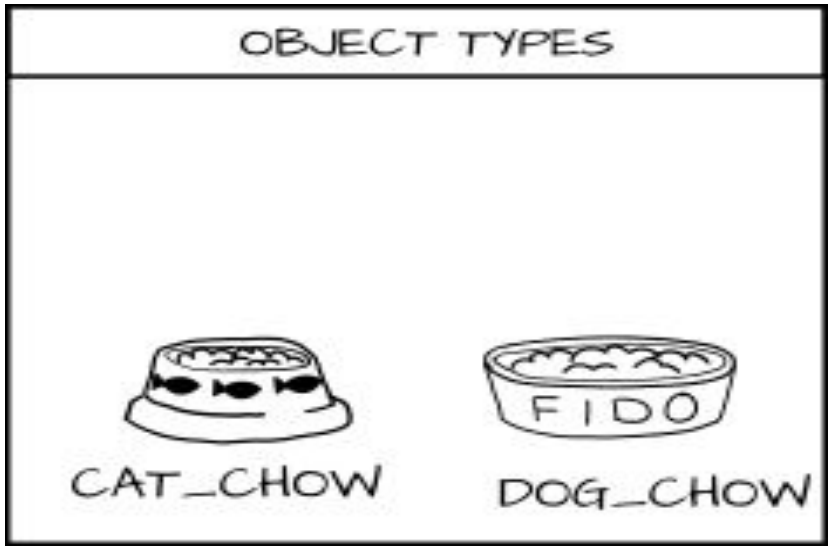
The Kernel then enforces the rules that the user has defined in the policies. By default any rules not allowed are automatically denied, similar to how a firewall will deny any access not explicitly permitted.

FOCUS: SELINUX

Imagine a system where we define types on objects like cats and dogs. A cat and dog are process types:



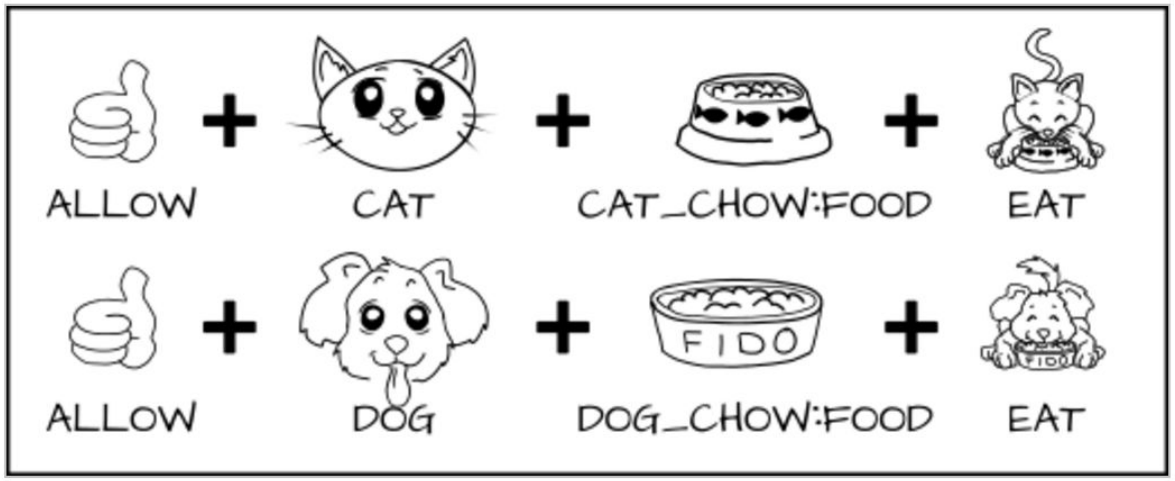
We have a class of objects that they want to interact with which we call food. And I want to add types to the food, *cat_food* and *dog_food*.



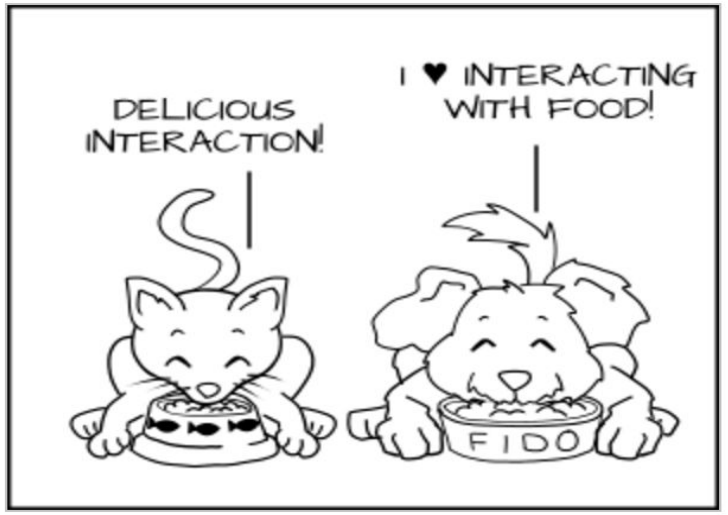
Red Hat

FOCUS: SELINUX

As a policy writer, we would define that a dog has permission to eat *dog_chow* food and a cat has permission to eat *cat_chow* food.
In SELinux we would write this rule in policy.
allow cat cat_chow:food eat;
allow dog dog_chow:food eat;



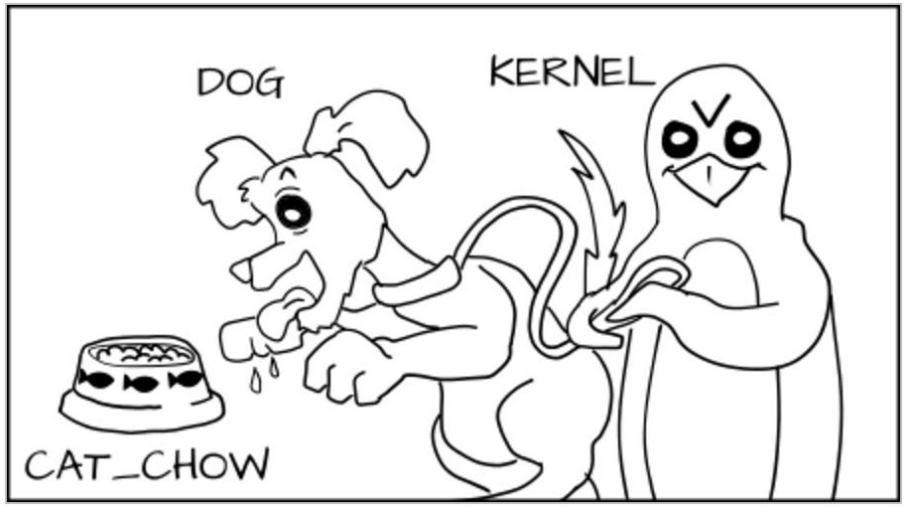
With these rules the kernel would allow the cat process to eat food labeled *cat_chow* and the dog to eat food labeled *dog_chow*.



Red Hat

FOCUS: SELINUX

But in a SELinux system everything is denied by default. This means that if the dog process tried to eat the *cat_chow*, the kernel would prevent it.



This is called Type Enforcement and is essential in protecting the host system from container processes.

The container processes can only read/execute /usr files and the processes can only write to the container files.

FOCUS: SELINUX

How OpenShift uses SELinux?

OpenShift is using SELinux with the “security constraints” feature as documented here :

https://docs.openshift.com/container-platform/3.11/architecture/additional_concepts/authorization.html#security-context-constraints

OpenShift provides 2 modes of operation with SELinux -

- RunAsAny (privileged)
- MustRunAs (restricted)

Generally, the containers of the applications will be created in the restricted mode, which will have SELinux Context specific changes applied on the container resources - process, files, directories, ports.

SELINUX TO THE RESCUE

On-entry container attack - CVE-2016-9962

On Red Hat systems with SELinux enabled, the dangers of even privileged containers are mitigated. SELinux prevents container processes from accessing host content even if those container processes manage to gain access to the actual file descriptors.

FOCUS: SECCOMP

- Finally, a **secure computing mode (seccomp) profile** can be associated with a container to restrict available system calls.
- Attach a **preconfigured seccomp profile** to every container during launch
- Can **immediate terminate** a container process with SIGKILL or SIGSYS in case the rules in the profile are violated

Présentation de Red Hat CoreOS
socle OS "immutable"
Réduction de la surface d'attaque

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

IMMUTABLE OPERATING SYSTEM

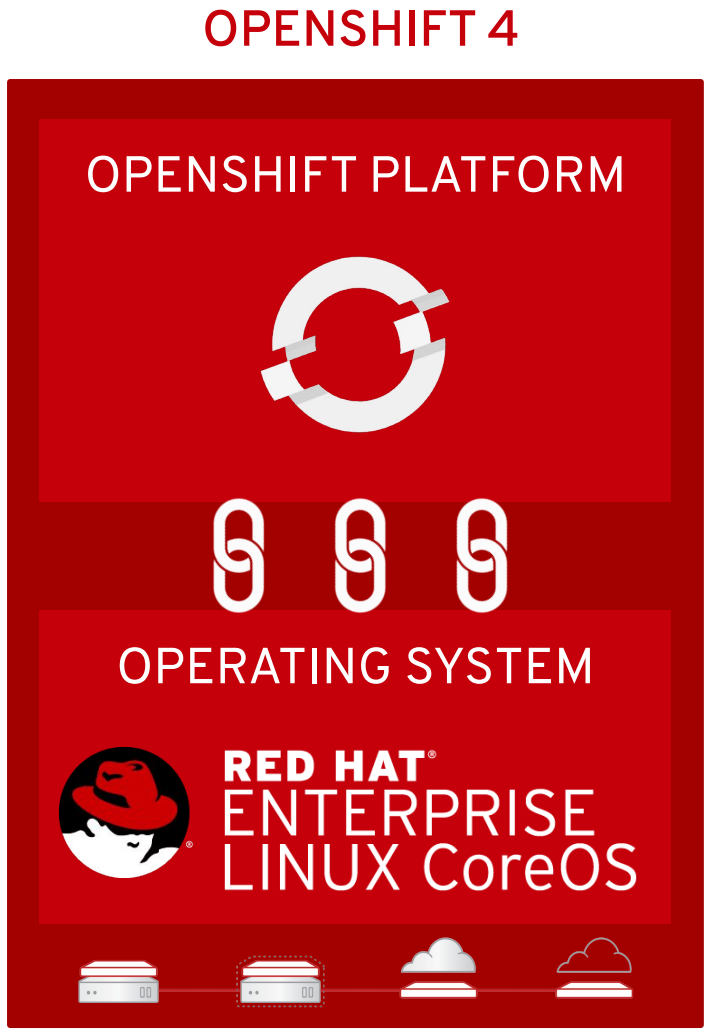
Red Hat Enterprise Linux CoreOS is versioned with OpenShift

CoreOS is tested and shipped in conjunction with the platform. Red Hat runs thousands of tests against these configurations.

Red Hat Enterprise Linux CoreOS is managed by the cluster

The Operating system is operated as part of the cluster, with the config for components managed by Machine Config Operator:

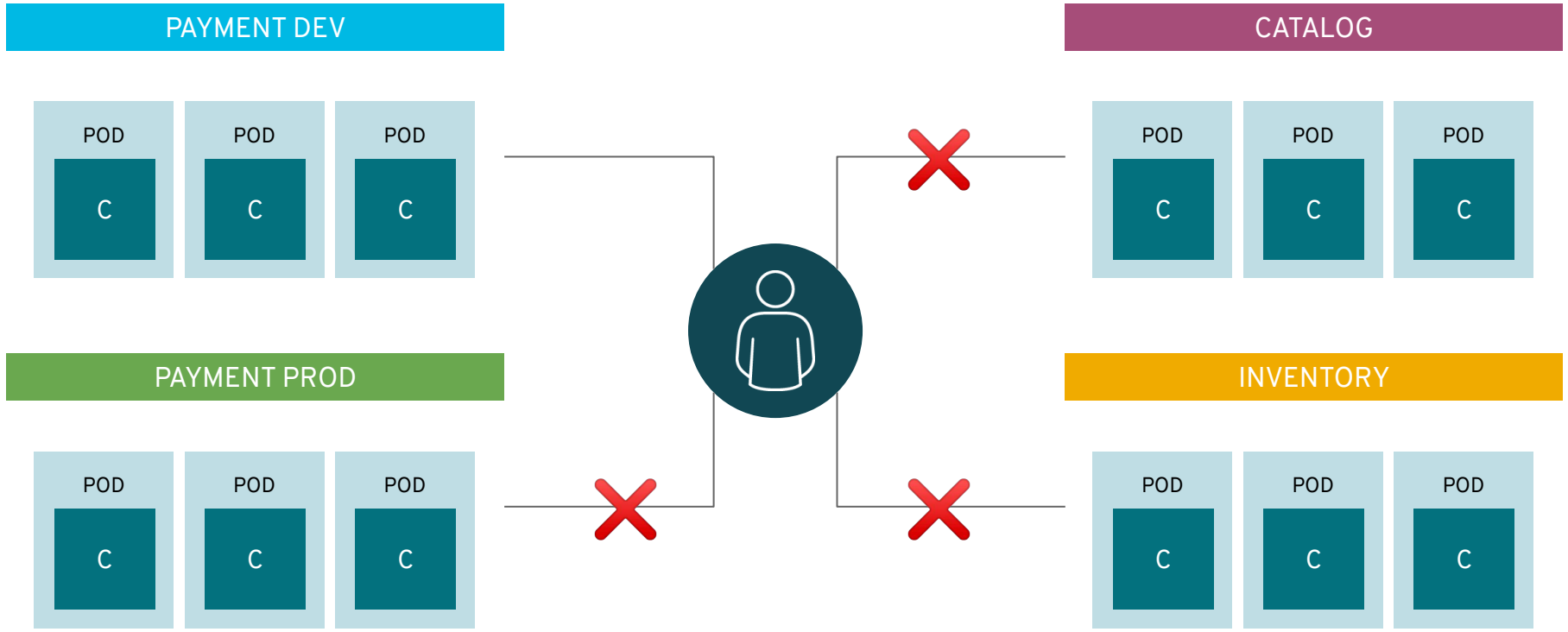
- CRI-O config
- Kubelet config
- Authorized registries
- SSH config



Red Hat

Gestion des projets (Namespace)
RBAC
Rôles utilisateurs

PROJECTS ISOLATE APPLICATIONS
across teams, groups and departments



Red Hat

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

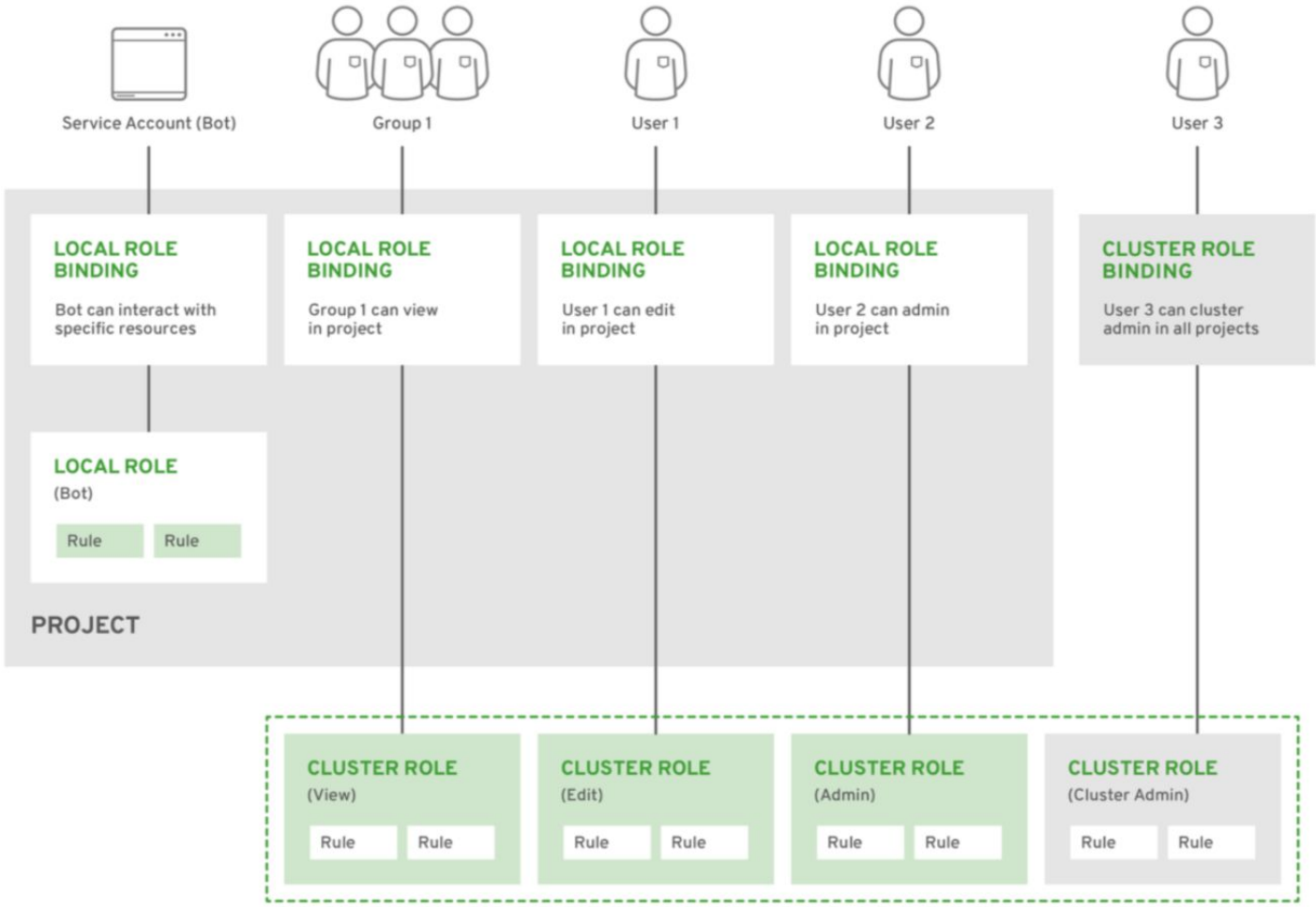


Figure 12 - Authorization Relationships

For more information see: [Managing RBAC in OpenShift]

Red Hat

Gestion des identités

introduction RH SSO pour les cas d'usage SAML et Synchro AD

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

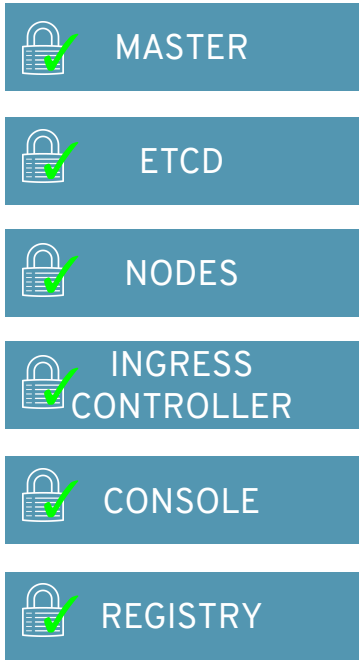
Supported Identity Providers include

- 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)

Gestion des Certificats

CERTIFICATE MANAGEMENT

- Certificates are used to provide secure connections to
  - master and nodes
  - Ingress controller and registry
  - etcd
- Certificate rotation is automated
- Optionally configure external endpoints to use custom certificates
- For example:
  Requesting and Installing Let's Encrypt Certificates for OpenShift 4



Red Hat

CERTIFICATE EXPIRY REPORT

OCP Certificate Expiry Report

Redeploying Certificates | Expiry Role Documenta

m01.example.com

Checked 14 total certificates. Expired/Warning/OK: 0/0/14. Warning window: 30 days

- **Expirations checked at:** 2017-03-15 16:02:53.563634
- **Warn after date:** 2017-04-14 16:02:53.563634

ocp_certs

	Certificate Common/Alt Name(s)	Serial	Health	Days Remaining	Expiration Date	Path
✔	CN:172.30.0.1, DNS:kubernetes, DNS:kubernetes.default, DNS:kubernetes.default.svc, DNS:kubernetes.default.svc.cluster.local, DNS:m01.example.com, DNS:openshift, DNS:openshift.default, DNS:openshift.default.svc, DNS:openshift.default.svc.cluster.local, DNS:172.30.0.1, DNS:192.168.122.241, IP Address:172.30.0.1, IP Address:192.168.122.241	int(4)/hex(0x4)	ok	694	2019-02-07 18:12:40	/etc/origin/master/master.server.crt
✔	CN:192.168.122.241, DNS:m01.example.com, DNS:192.168.122.241, IP Address:192.168.122.241	int(14)/hex(0xe)	ok	694	2019-02-07 18:19:36	/etc/origin/node/server.crt
✔	CN:openshift-signer@1486491158	int(1)/hex(0x1)	ok	1789	2022-02-06 18:12:38	/etc/origin/master/ca-bundle.crt
✔	CN:openshift-signer@1486491158	int(1)/hex(0x1)	ok	1789	2022-02-06 18:12:38	/etc/origin/node/ca.crt

etcd

	Certificate Common/Alt Name(s)	Serial	Health	Days Remaining	Expiration Date	Path
✔	CN:etcd-signer@1486490714	int(13699604374018404644)/hex(0xbe1ec59834c13124L)	ok	1789	2022-02-06 18:06:00	/etc/etcd/ca.crt
✔	CN:m01.example.com, IP Address:192.168.122.241	int(1)/hex(0x1)	ok	1789	2022-02-06 18:06:16	/etc/etcd/server.crt
✔	CN:m01.example.com, IP Address:192.168.122.241	int(2)/hex(0x2)	ok	1789	2022-02-06 18:06:16	/etc/etcd/peer.crt

kubeconfigs

	Certificate Common/Alt Name(s)	Serial	Health	Days	Expiration	Path

CERTIFICATE CHECKS

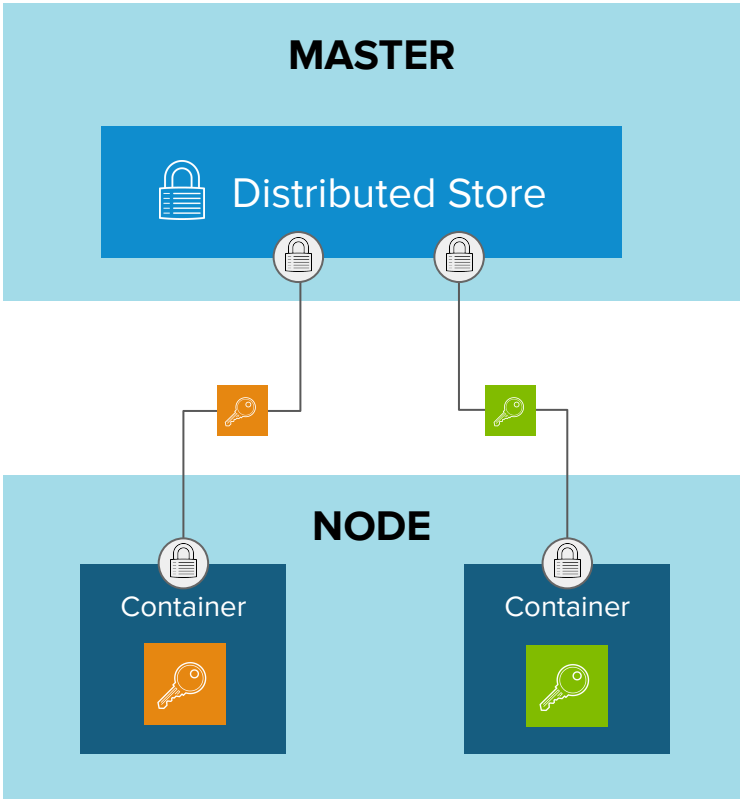
- master and nodes
- router and registry service certificates from etcd secrets
- master, node, router, registry, and kubeconfig files for cluster-admin users
- etcd certificates

Red Hat

Gestion des secrets
Intégration Vault externe

SECRETS MANAGEMENT

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

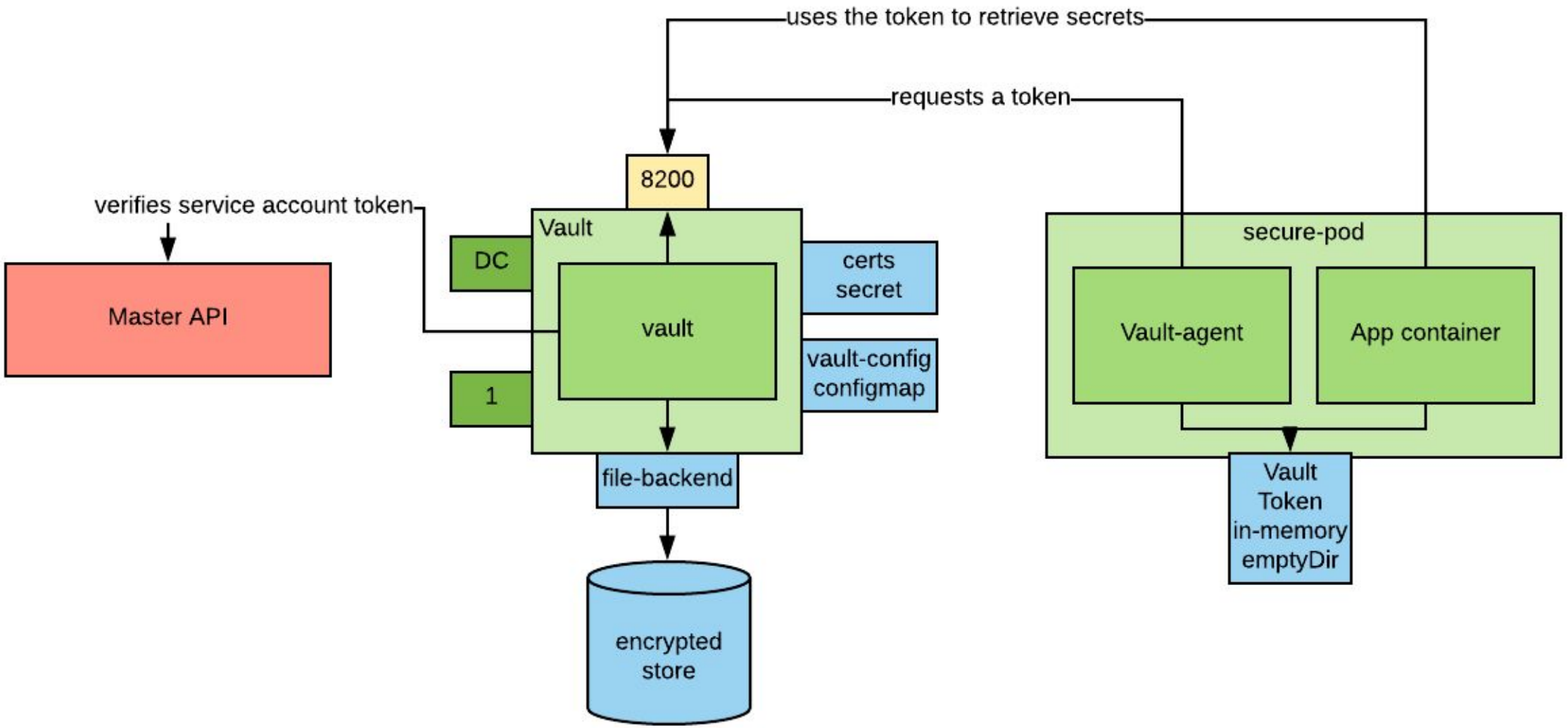


* Full disk encryption is supported for AWS, VSphere. Encrypting the etcd datastore is not available in OCP 4.1. Targeted for OCP 4.3.

About HashiCorp Vault

- Secures, stores and controls access to tokens, passwords, certificates, encryption keys for protecting secrets and other sensitive data
- Provides UI, CLI, or HTTP API.

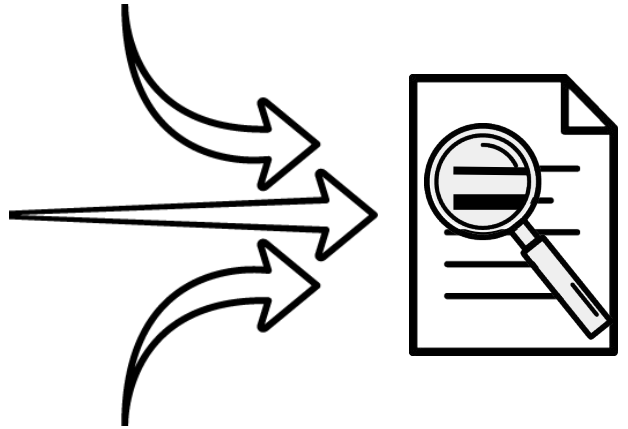
Vault Agent Architecture



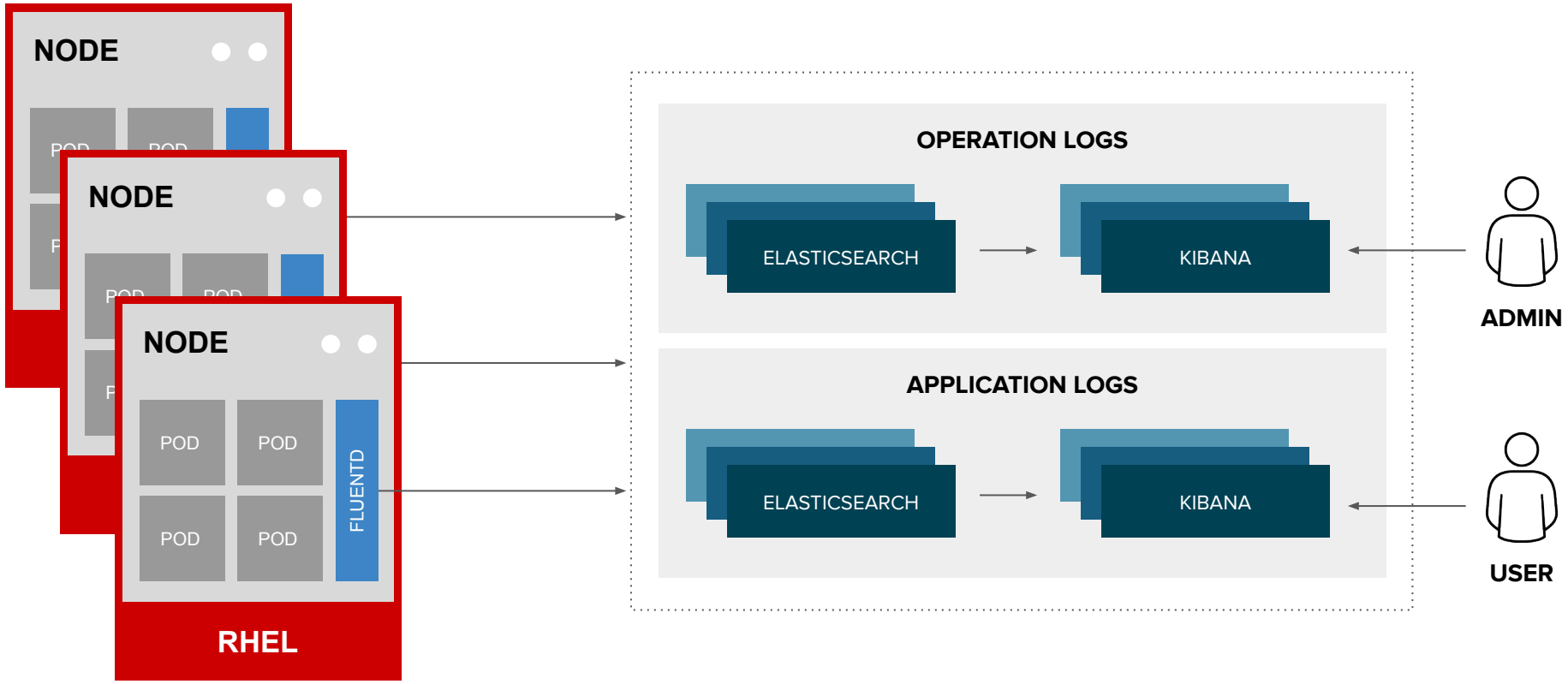
Gestion des Logs
“Forward” vers service Log externe

CENTRAL LOG MANAGEMENT WITH EFK

- EFK stack to aggregate 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
- Ability to send logs elsewhere
  - External elasticsearch, Splunk, etc



CENTRAL LOG MANAGEMENT WITH EFK

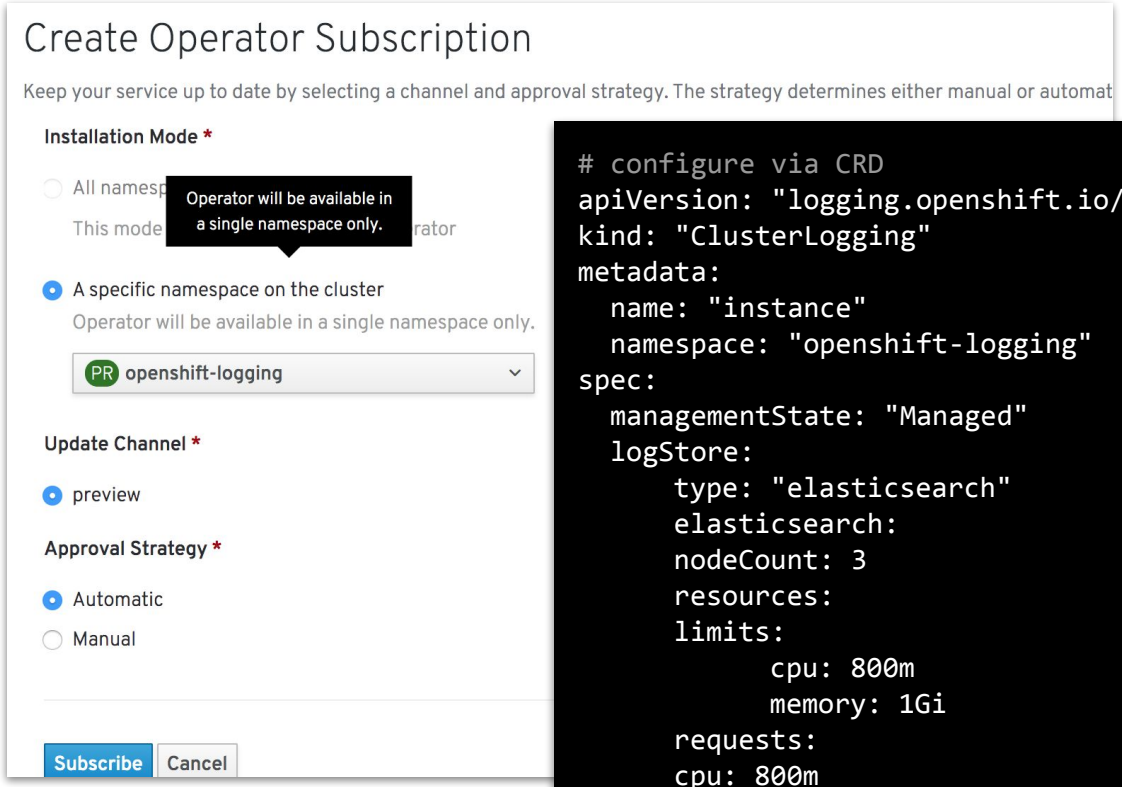


Red Hat

CLUSTER LOG MANAGEMENT

Install the Elasticsearch and Cluster Logging Operators from OperatorHub

- 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
- Ability to send logs elsewhere
  - External elasticsearch, Splunk, etc

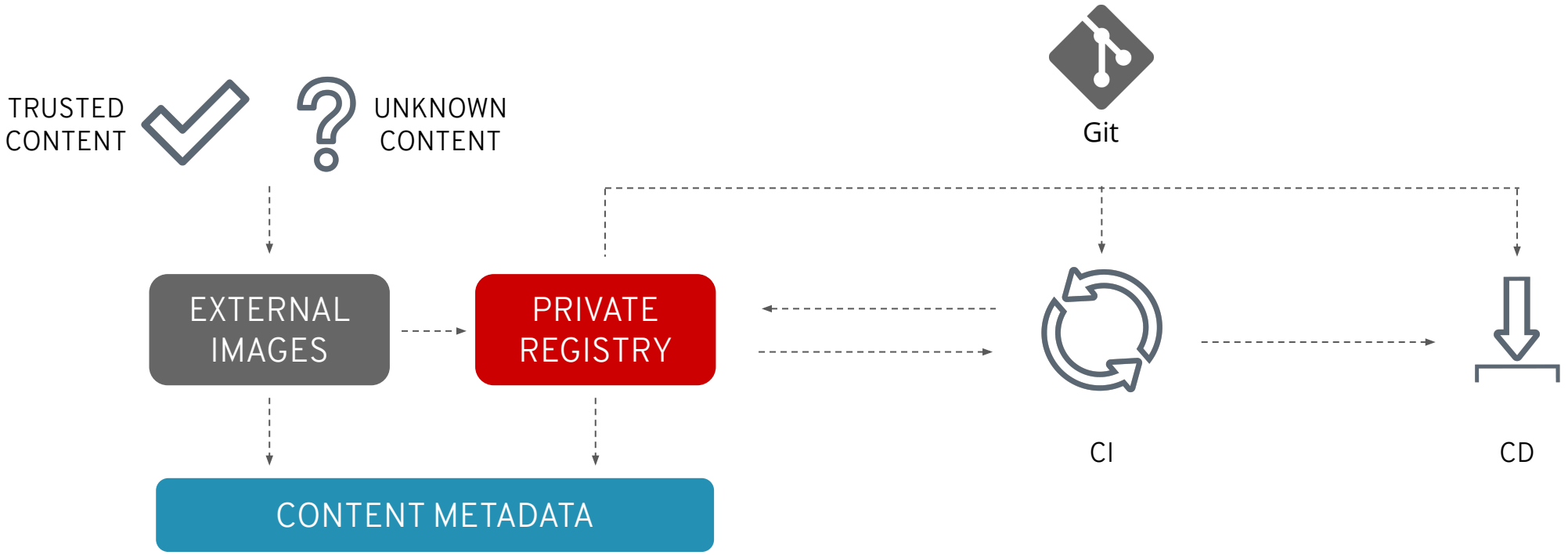



```
# configure via CRD
apiVersion: "logging.openshift.io/v1"
kind: "ClusterLogging"
metadata:
  name: "instance"
  namespace: "openshift-logging"
spec:
  managementState: "Managed"
  logStore:
      type: "elasticsearch"
      elasticsearch:
      nodeCount: 3
      resources:
      limits:
       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

Sécurisation de la Registry
+ Images certifiées
+ Scan Images
+ gestion des droits d'accès (push/pull image)

SECURE & AUTOMATE THE CONTENT LIFECYCLE

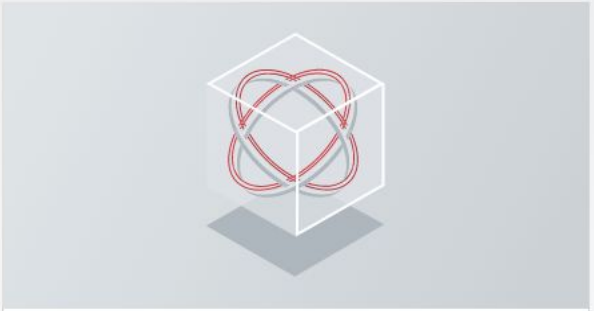
Elements of the Openshift container pipeline



Red Hat

Software

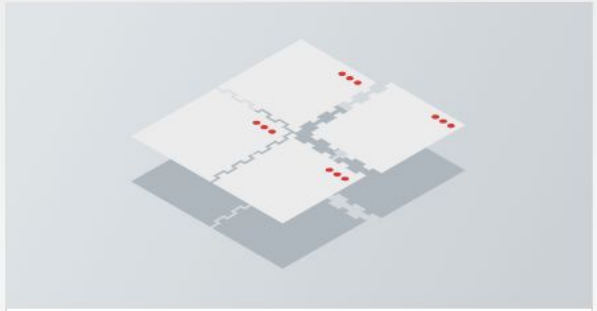
Containers, operators, and standalone applications tested specifically on the Red Hat platform.



OpenShift Operators

Operators enable easy orchestration of containers into OpenShift and other Kubernetes-compatible environments.

Explore operators



OpenStack Infrastructure

Applications and plugins certified on the Red Hat platform.

Explore OpenStack

Applications

Non-containerized, third party applications tested specifically on the Red Hat platform.

Explore applications

Container Images

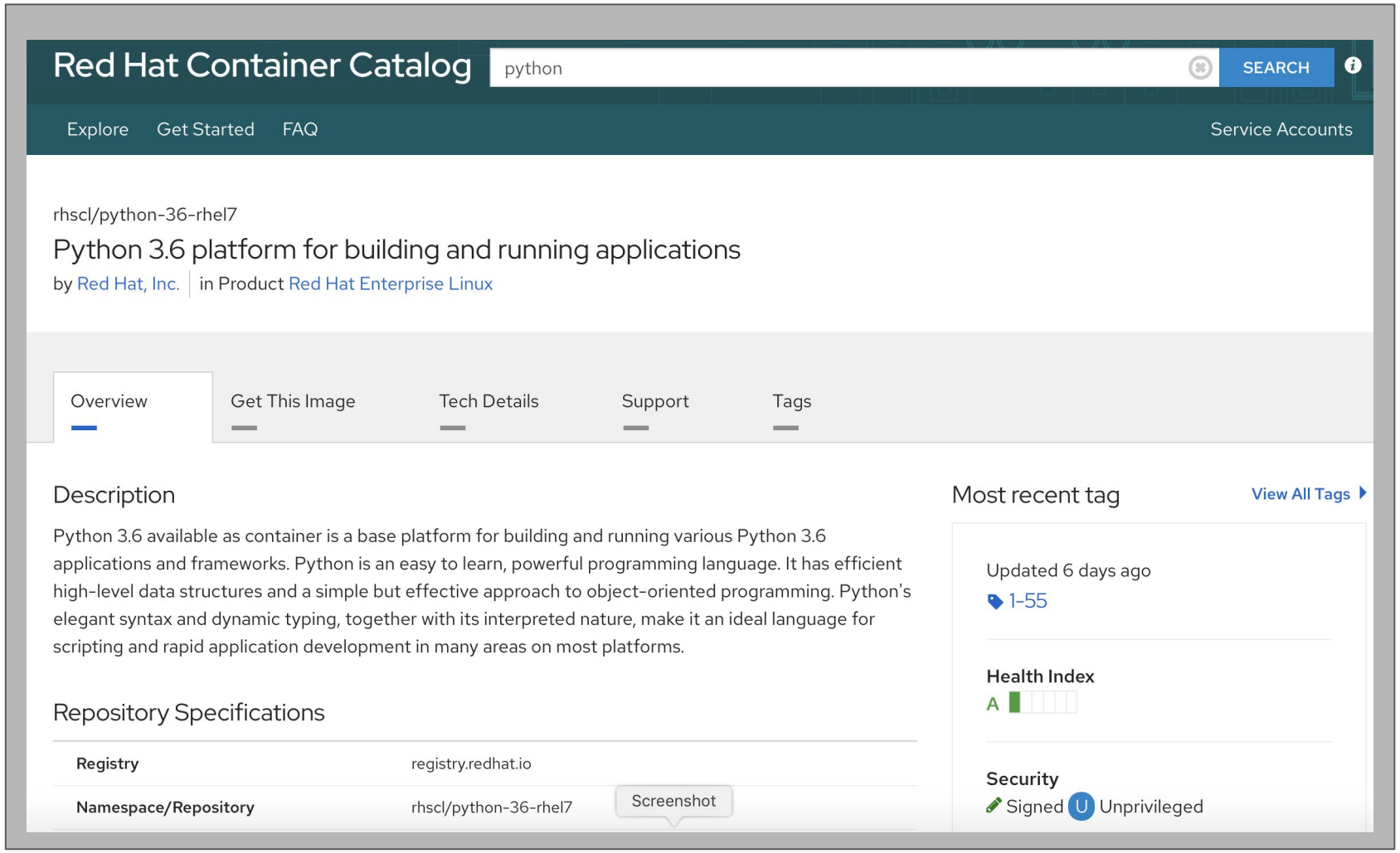
Container images offer lightweight and self-contained software to enable deployment at scale.

Explore container images

EXTERNAL CONTENT: USE TRUSTED SOURCES

Red Hat Container Images

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



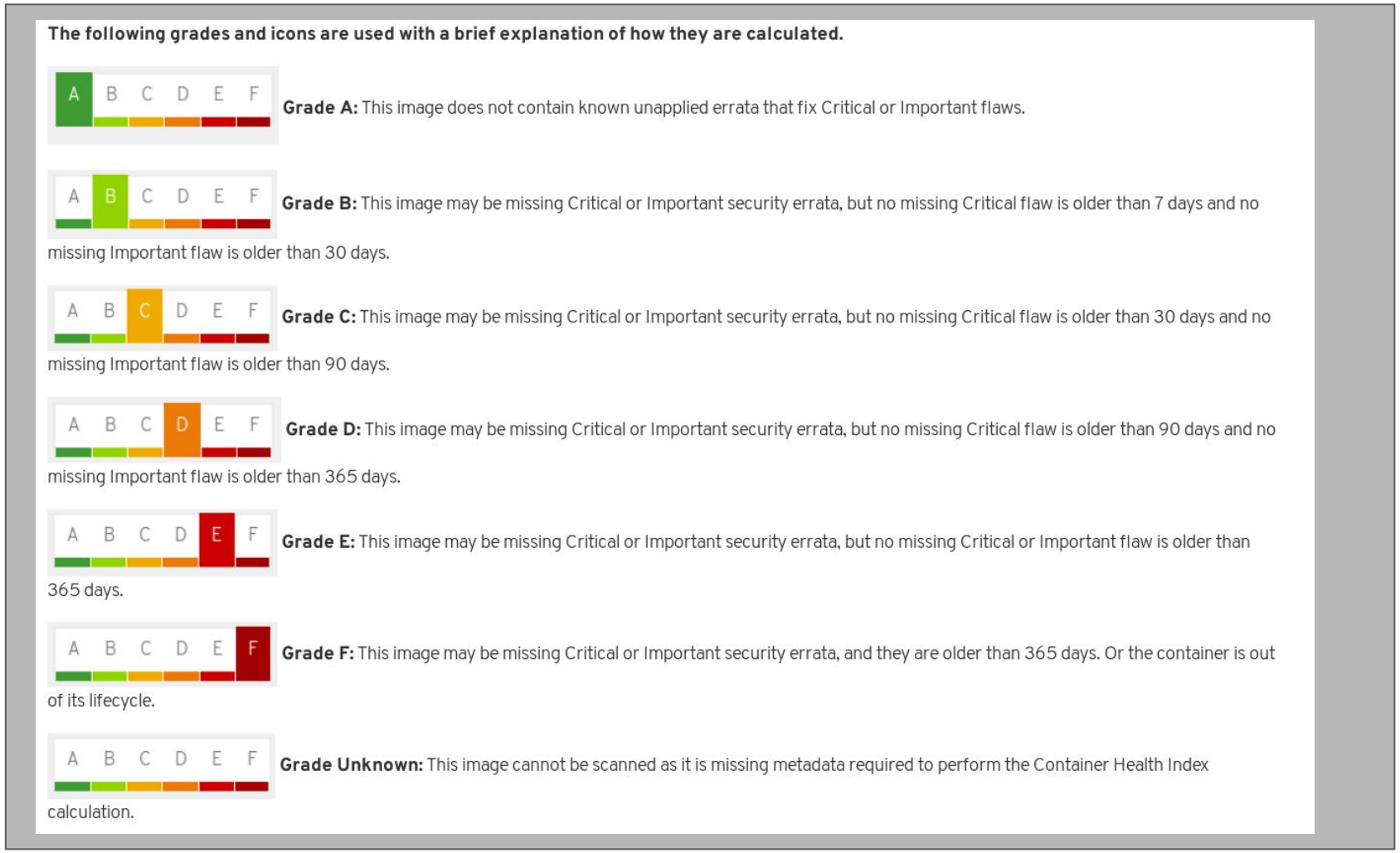
*More about the Health Index

Red Hat

CONTENT : CONTAINER HEALTH INDEX

Red Hat Container Images

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



*More about the Health Index

Red Hat

RED HAT QUAY

ENTERPRISE CONTAINER REGISTRY

Quay v3 Key Features

- Full support for Docker Registry API v2_s2
- Multi-arch (manifest) and Windows images
- Rebranded and rebased (RHEL)
- New config UI to save & upload Quay config
- Parallel upgrade (2x 5min downtime only)

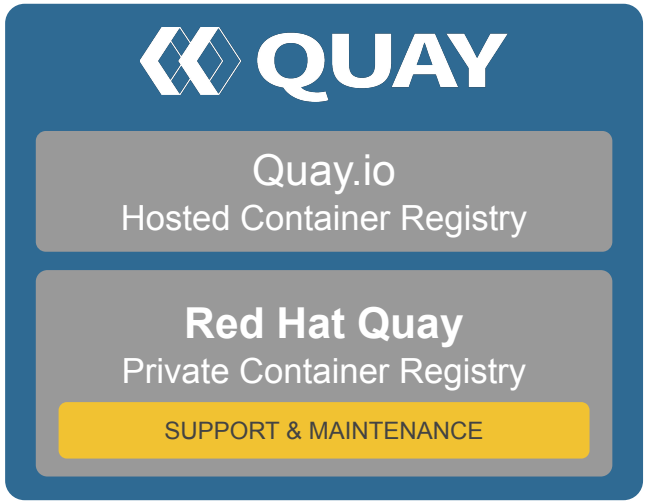
Red Hat Quay and OpenShift

Red Hat Quay and OpenShift work well together. This includes both running Quay **on** and using Quay **with** OpenShift.

Introducing Red Hat Quay 3

Red Hat

REGISTRY FEATURES

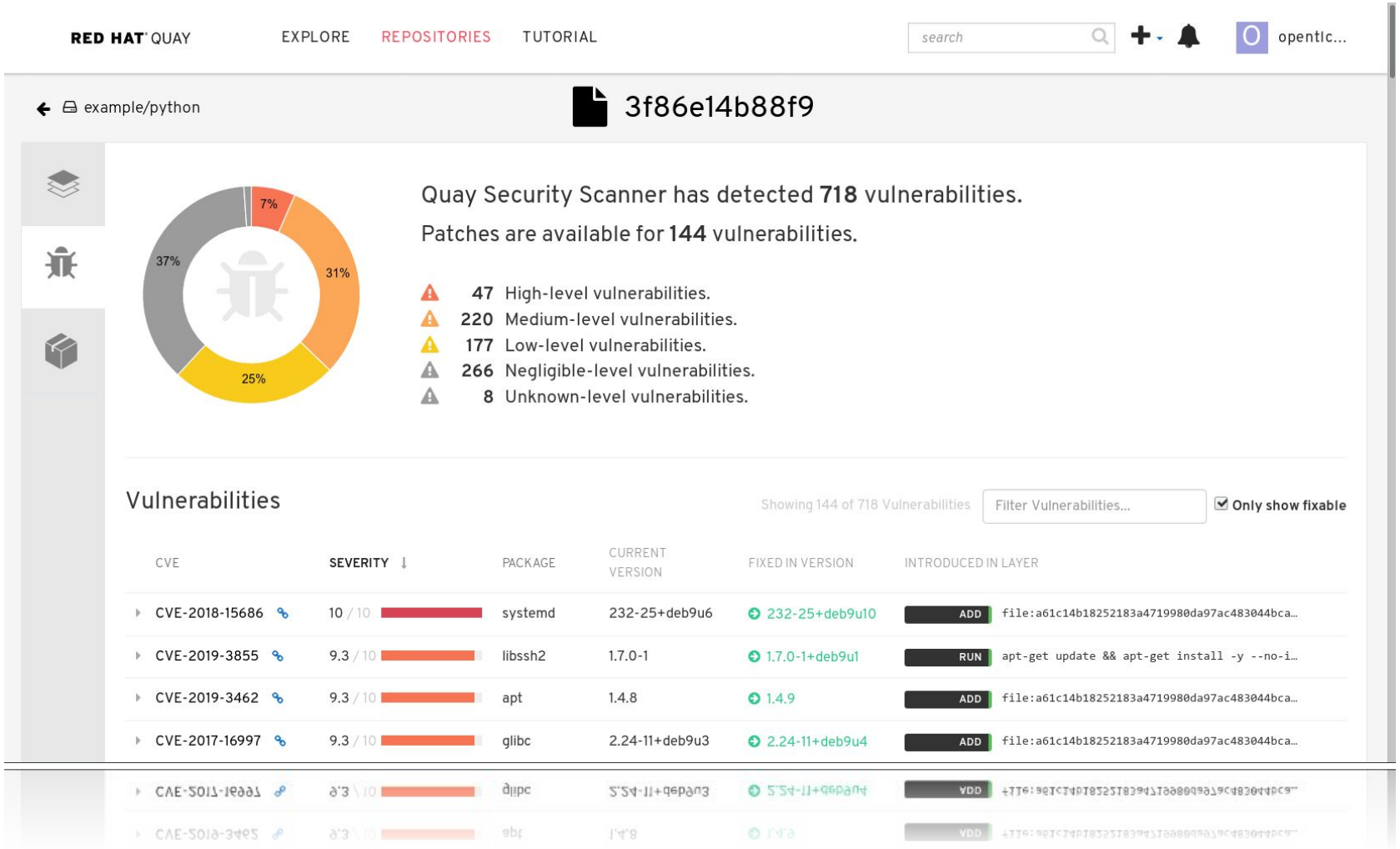


- **Vulnerability Scanning (powered by Clair)**
  Continually scan your containers for vulnerabilities, giving you complete visibility into known issues and how to fix them
- **Geographic Replication**
  Reliably store, build and deploy a single set of container images across multiple geographies
- **Automated software deployments**
  Streamline your continuous integration/continuous delivery (CI/CD) pipeline with build triggers, git hooks, and robot accounts
- **Advanced Access Control Management**
  Fine-grain access control of the registry with multiple identity and authentication providers as well as support for teams and organization mapping

Red Hat

VULNERABILITY SCANNING - CLAIR

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

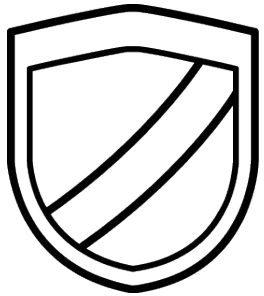


Red Hat

CONTAINER IMAGE MANAGEMENT

Image Policy

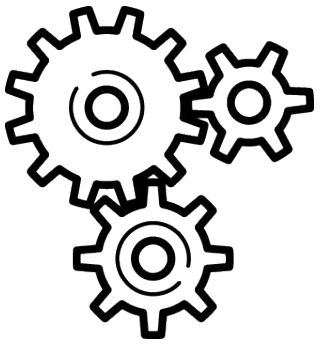
OpenShift ecosystem supports controlling the images that can be run on the platform



- Container runtime level
  - Block vulnerable image registries (DockerHub)
- OpenShift
  - ImagePolicy Admission Plugin
    - Block images from running
      - Based image source, label or annotation
        - Annotation: `images.openshift.io/deny-execution`

Red Hat

Image Scanning



Reduce threat vector by assessing image contents

- OpenSCAP scanning integrated into OpenShift Ecosystem
  - Supports only RPM analysis. Most organizations want deeper inspection
  - Image-inspector image available for standalone use or integration in CFME
- Red Hat Solutions
  - Atomic scan, Clair (Quay ecosystem)
  - Red Hat Container Catalog (RHCC) provides health index
- Multiple 3rd Party vendor solutions
  - Twistlock, BlackDuck, Sysdig, Aqua

Many customers evaluating 3rd party integrations - Trust but verify

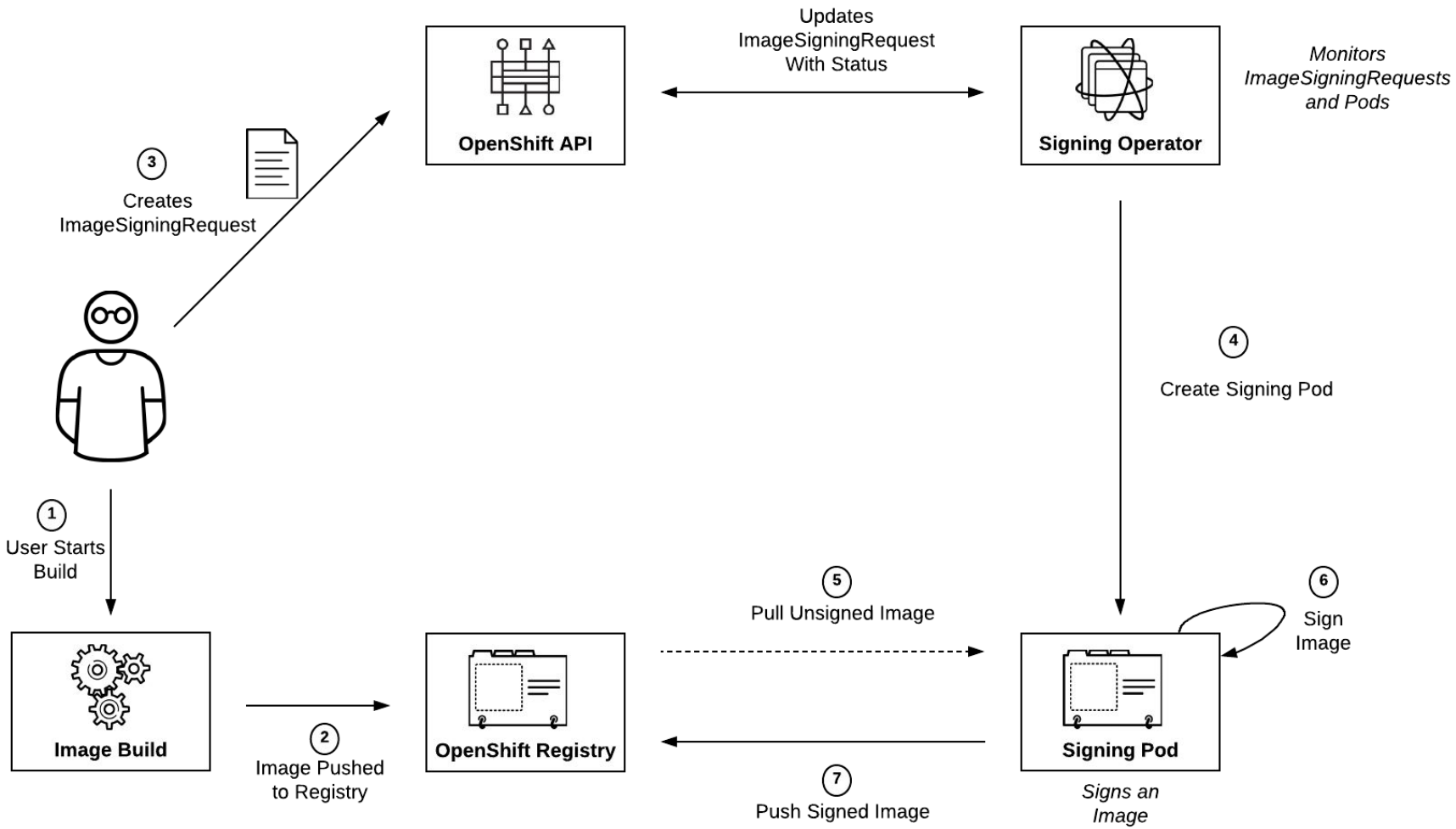
Image Signing

Digital signature applied to container images to aid in verifying trust

- Organizations want to leverage signing to certify images at various stages of SDLC
- `atomic sign` is the current supported method
  - GPG key based
  - Each node must have GPG key used to sign image
- Container runtime can also be configured to enforce image signatures
- Docker Notary one of the most popular implementations
- Future:
  - Clair as part of Quay to do scanning, signing and policy enforcement in OpenShift (act on it). Targeted for 4.1.

Several projects of interest in the Kubernetes community including Grafeas and Kritis.
Progress in these communities is slow.

Image Signing Example



Red Hat

Sécurisation
Du Control Plane
APIs

AUTHENTICATION & AUTHORIZATION (Master)

OAUTH API AUTHENTICATION

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
  - Allows multiple identities to map to the same OpenShift user
  - Allows deconflicting between identity provider roles
- Issues an OAuth access token which authenticates that user to the API

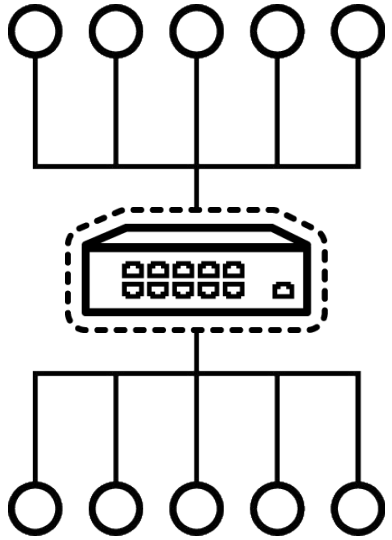
API ROLE-BASED AUTHORIZATION

- 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

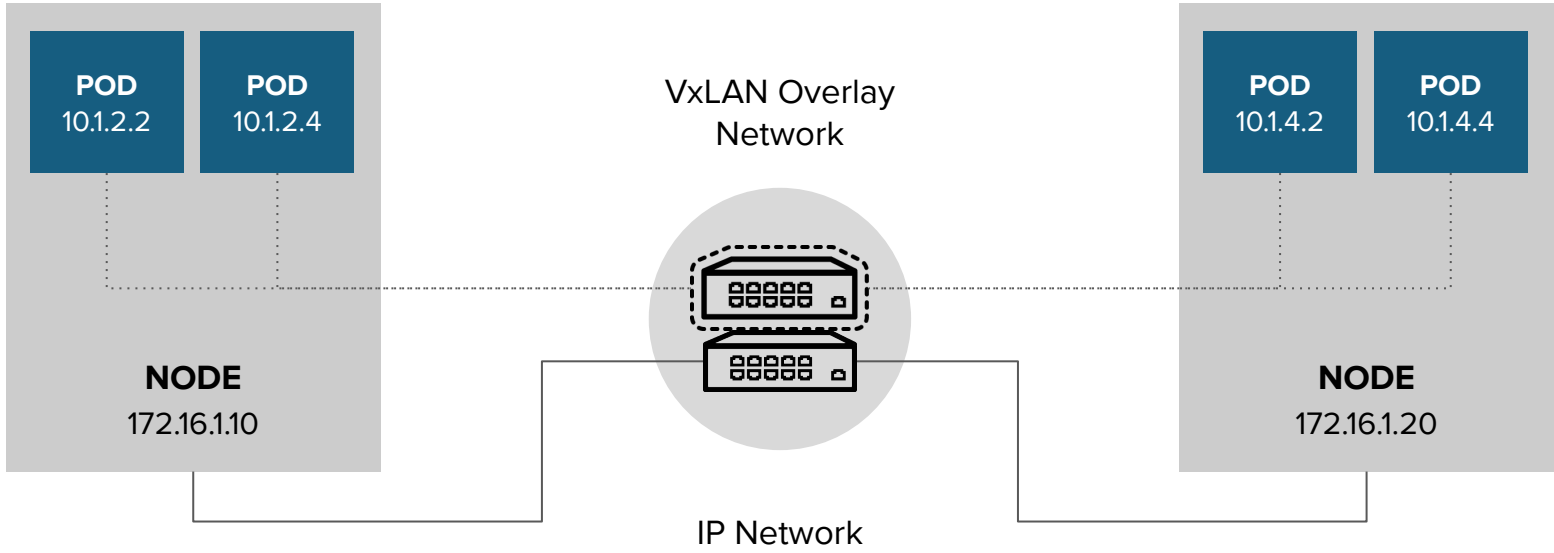
Software Defined Network
SDN
Isolation Réseaux
EGRESS
Micro-Segmentation

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

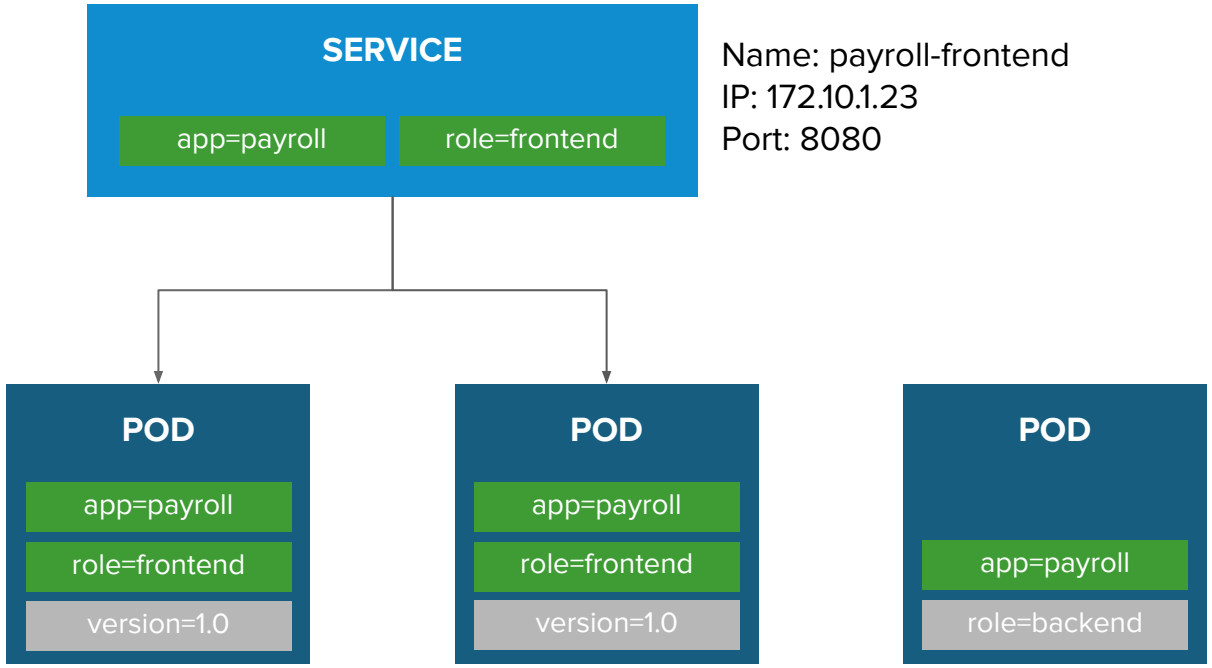


OPENSHIFT NETWORKING



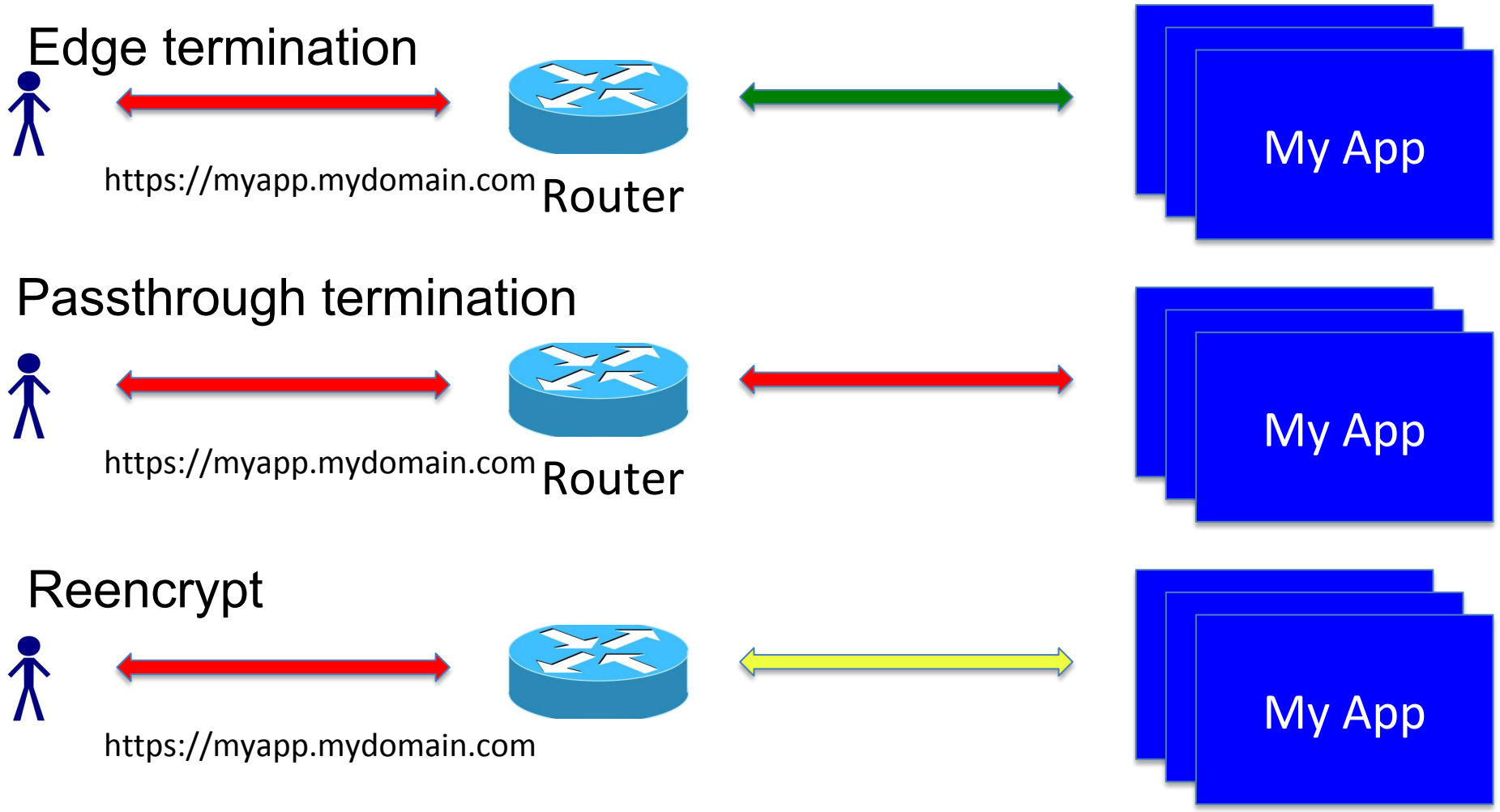
Red Hat

BUILT-IN SERVICE DISCOVERY
INTERNAL LOAD-BALANCING



Red Hat

SSL AT INGRESS (WITH OPENSHIFT ROUTES)

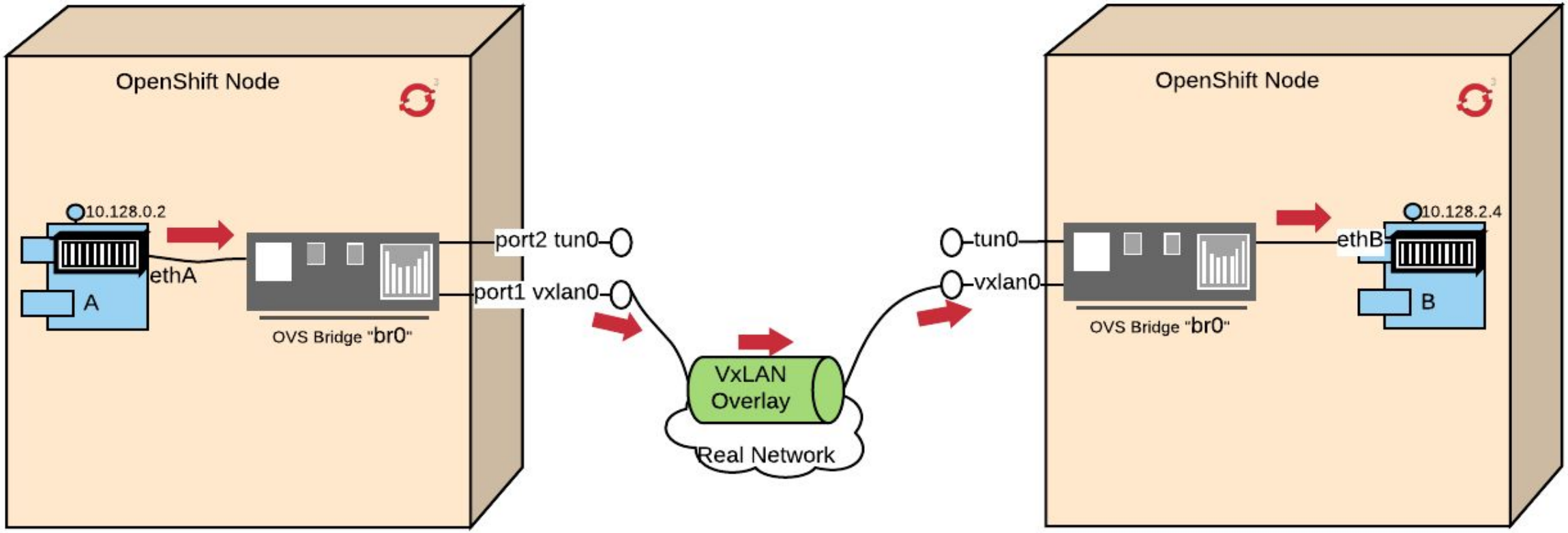


Red Hat

POD TO POD TRAFFIC - PODS ON DIFFERENT NODES

Flow of traffic

eth0(in A's netns) - vethA - br0 - vxlan0 - network - vxlan0 - br0- vethB - eth0(in B's netns)



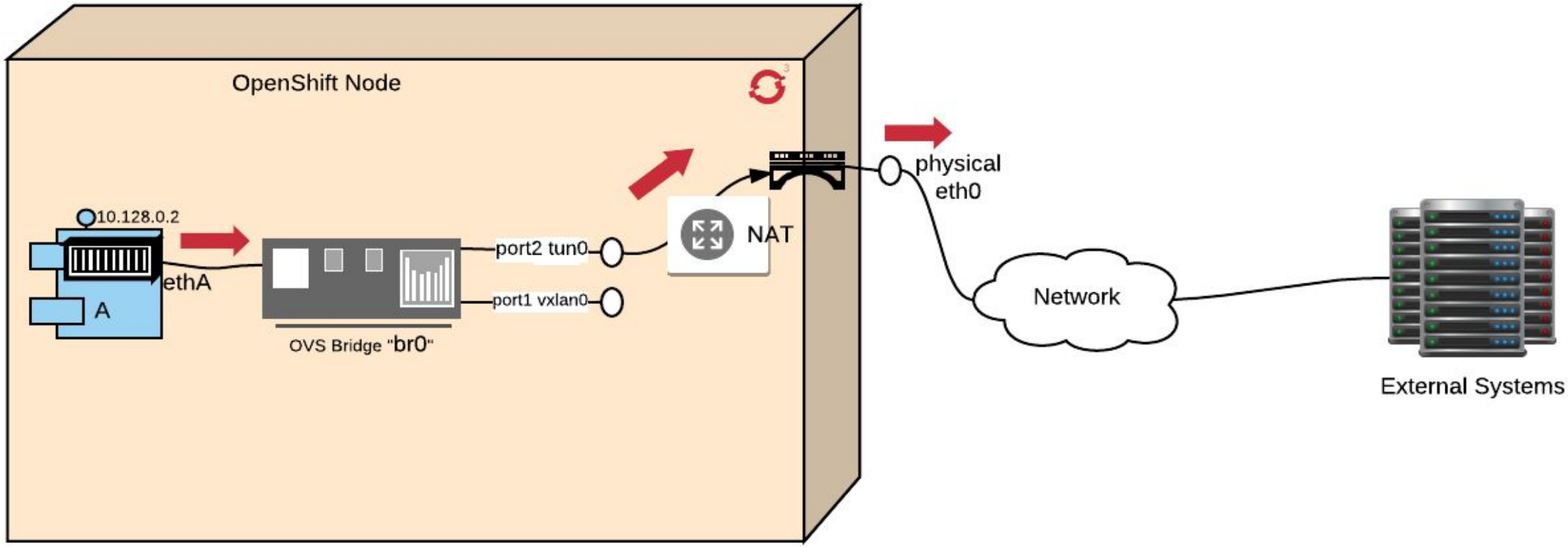
* Peer vEthernet device for container A is named ethA and for container B is named ethB

Red Hat

POD TO EXTERNAL SYSTEM OUTSIDE OPENSHIFT

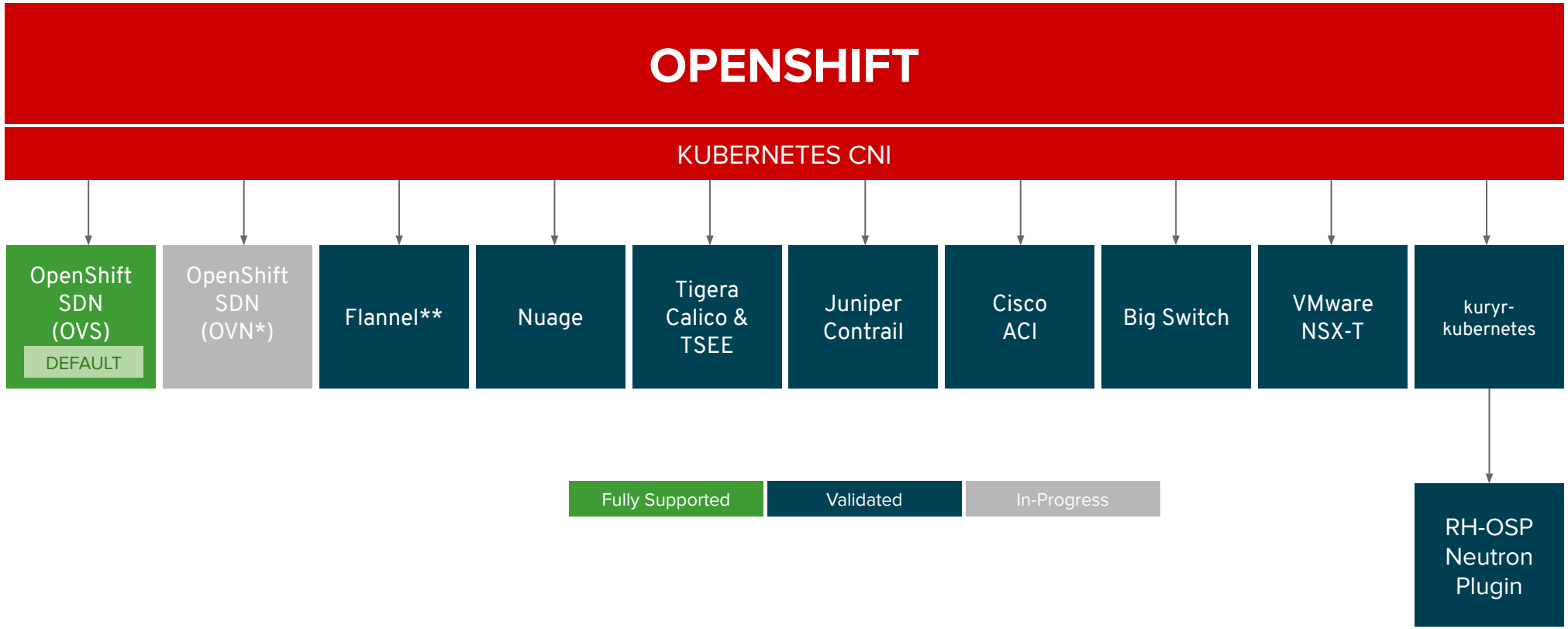
Flow of traffic

eth0(in A's netns) - vethA - br0 - tun0 - (NAT) - eth0(physical device) - Internet



Red Hat

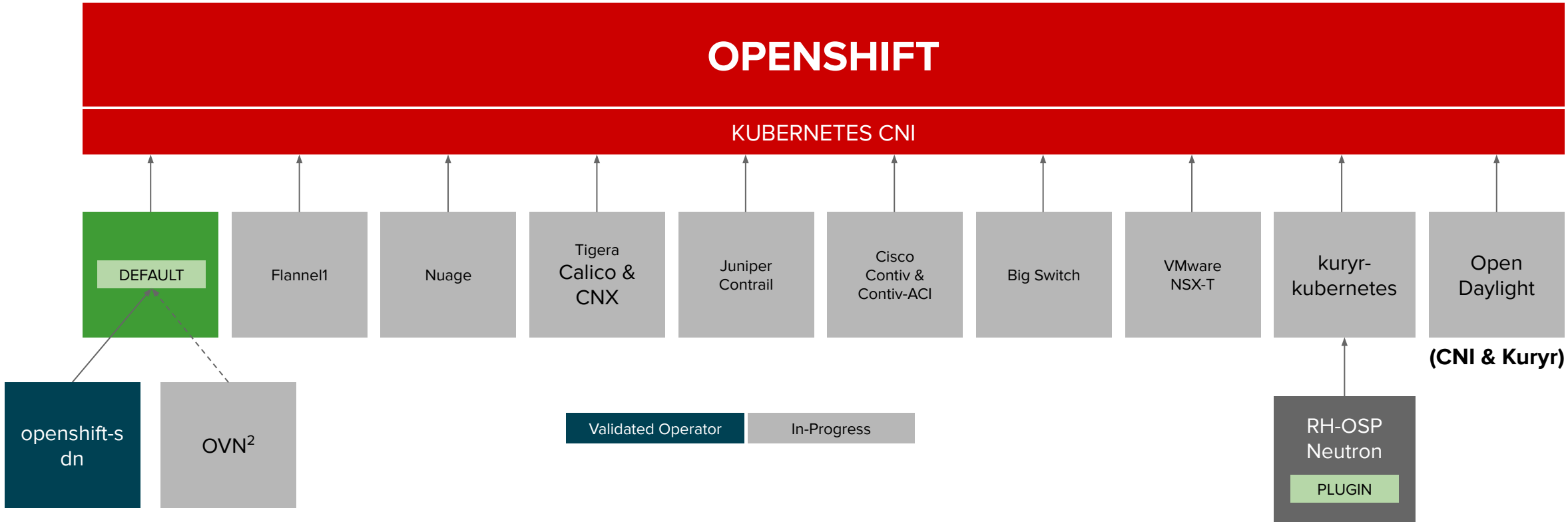
OPENSHIFT v3 NETWORK PLUGINS



* Coming as default in OCP 4.3
** Flannel is minimally verified and is supported only and exactly as deployed in the OpenShift on OpenStack reference architecture

Red Hat

OPENSHIFT v4 NETWORK PLUGINS



[1] Flannel is minimally verified and is supported only and exactly as deployed in the OpenShift on OpenStack reference architecture
[2] Targeting OCP 4.2 GA

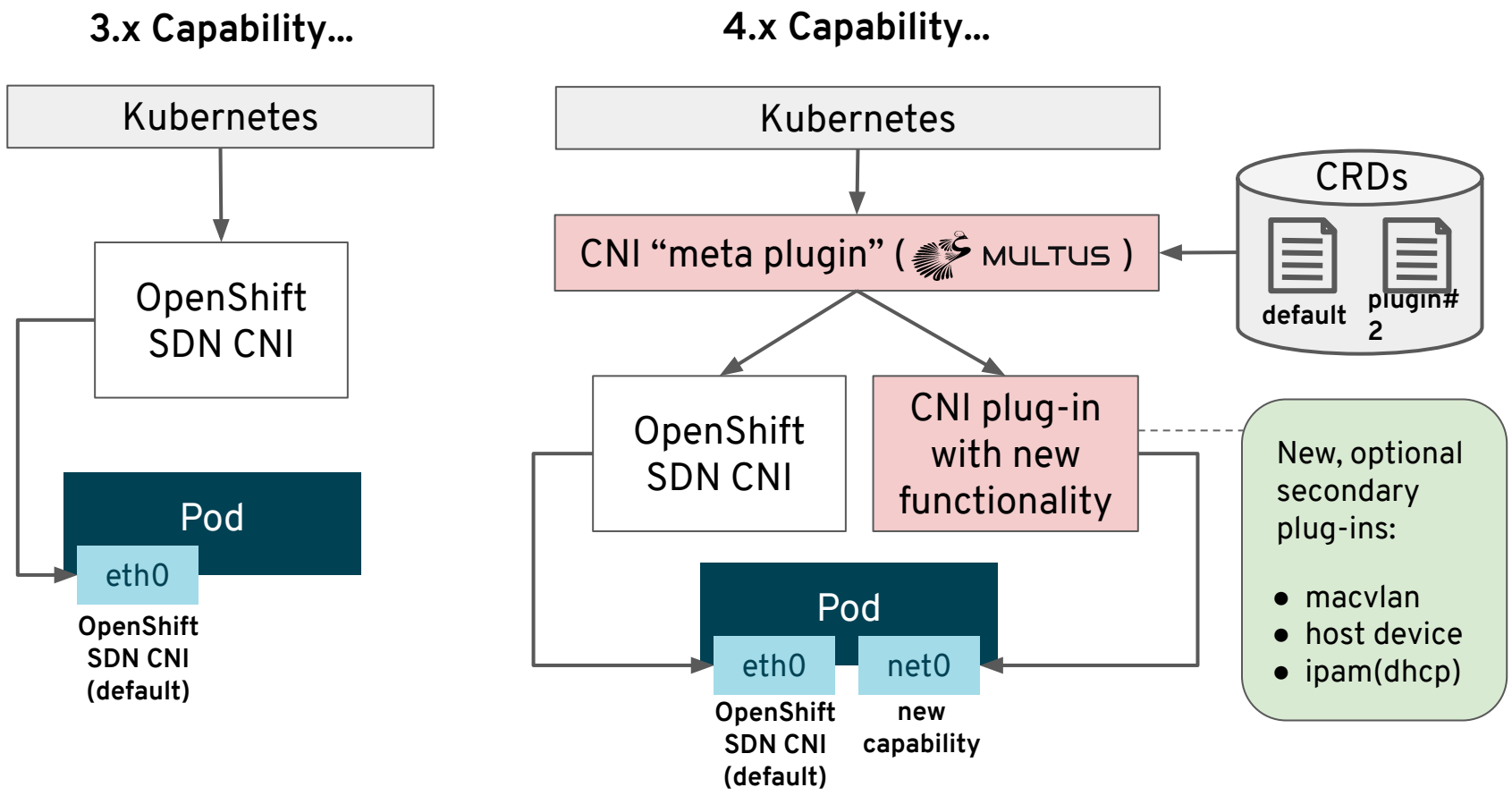
Red Hat

OPENSHIFT MULTUS

Multus Enables Multiple Networks & New Functionality to Existing Networking

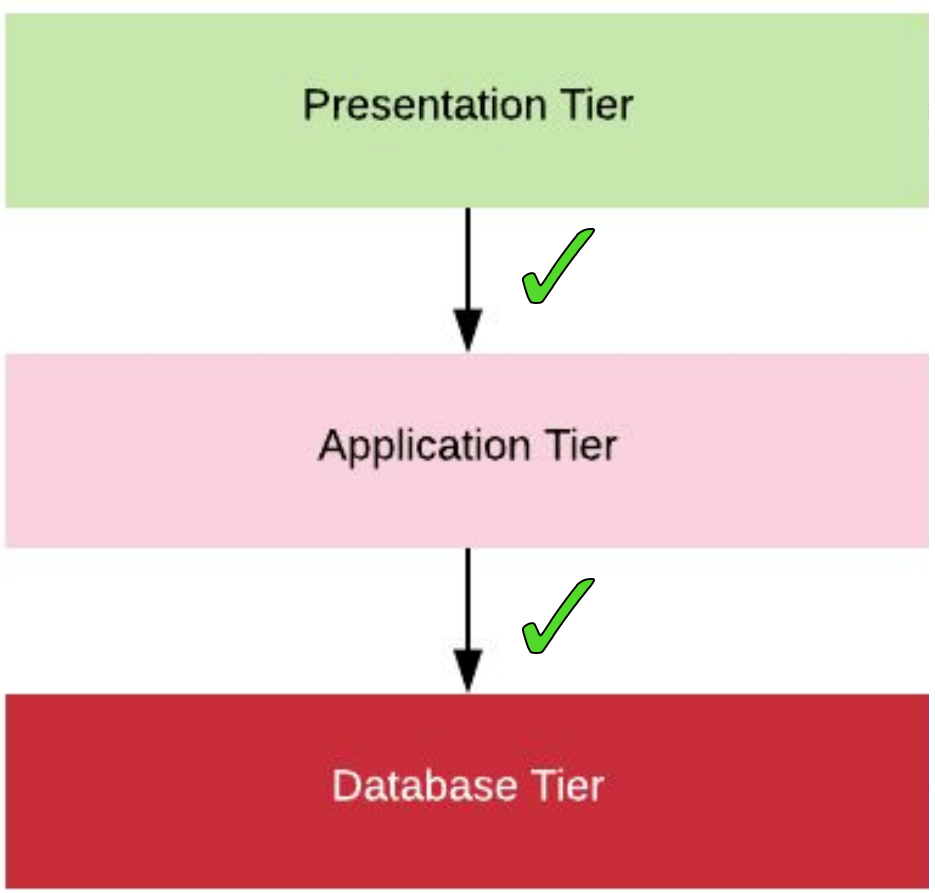
The Multus CNI "meta plugin" for Kubernetes enables one to **create multiple network interfaces per pod,** and assign a CNI plugin to each interface created.

1. Create pod annotation(s) to call out a list of intended network attachments…
2. …each pointing to CNI network configurations packed inside CRD objects



Red Hat

TRAFFIC RESTRICTIONS ACROSS APPLICATION TIERS



Allowed connections

Disallowed connections

In the world of OpenShift, how can we restrict traffic across Application Tiers?

Red Hat

NETWORK POLICY OBJECTS - INTRODUCTION

Enables **Microsegmentation**

Allows configuring individual policies at the Pod Level

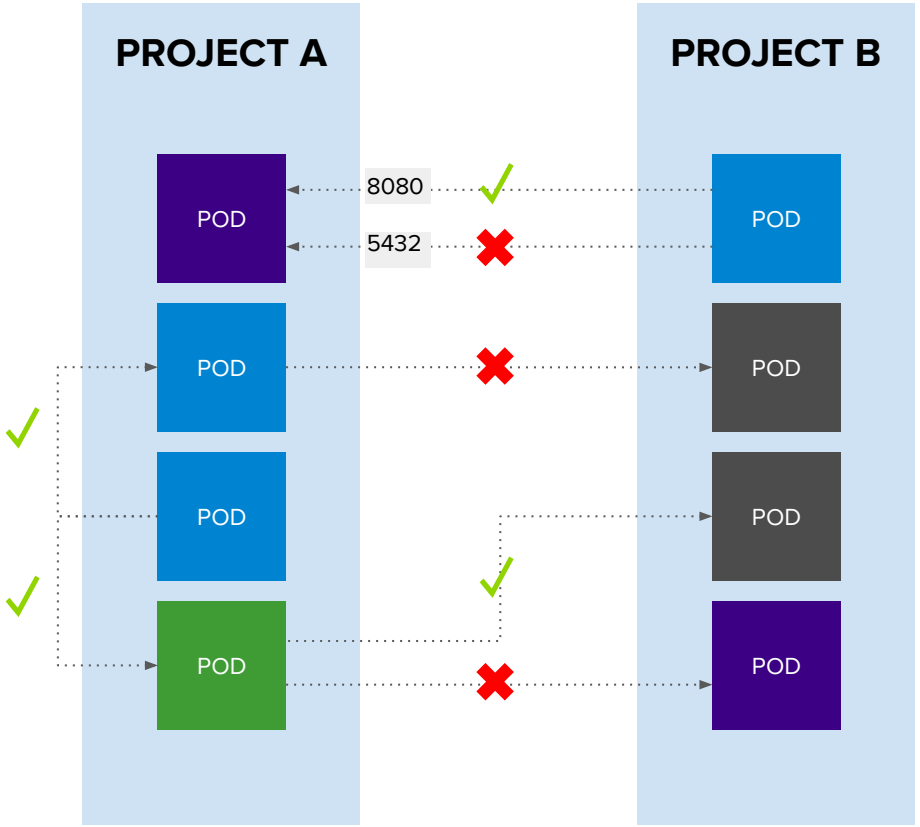
Apply to ingress traffic for pods and services

Allows restricting traffic between the pods within a project/namespace

Allows traffic to specific pods from other projects/namespaces

OPENSHIFT SDN

Network Policy enabled by default in OpenShift 4



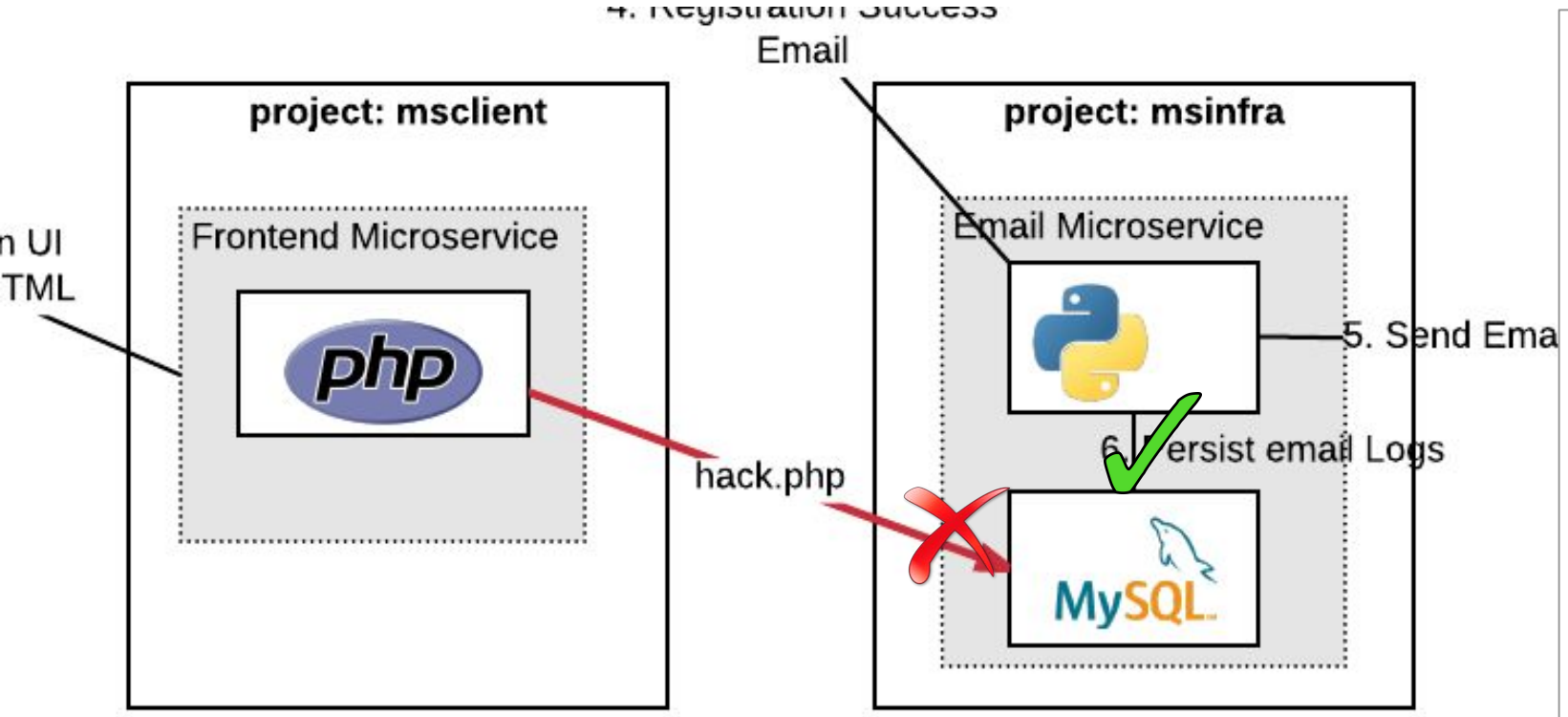
Example Policies

- 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

NETWORK POLICY OBJECTS TO RESCUE



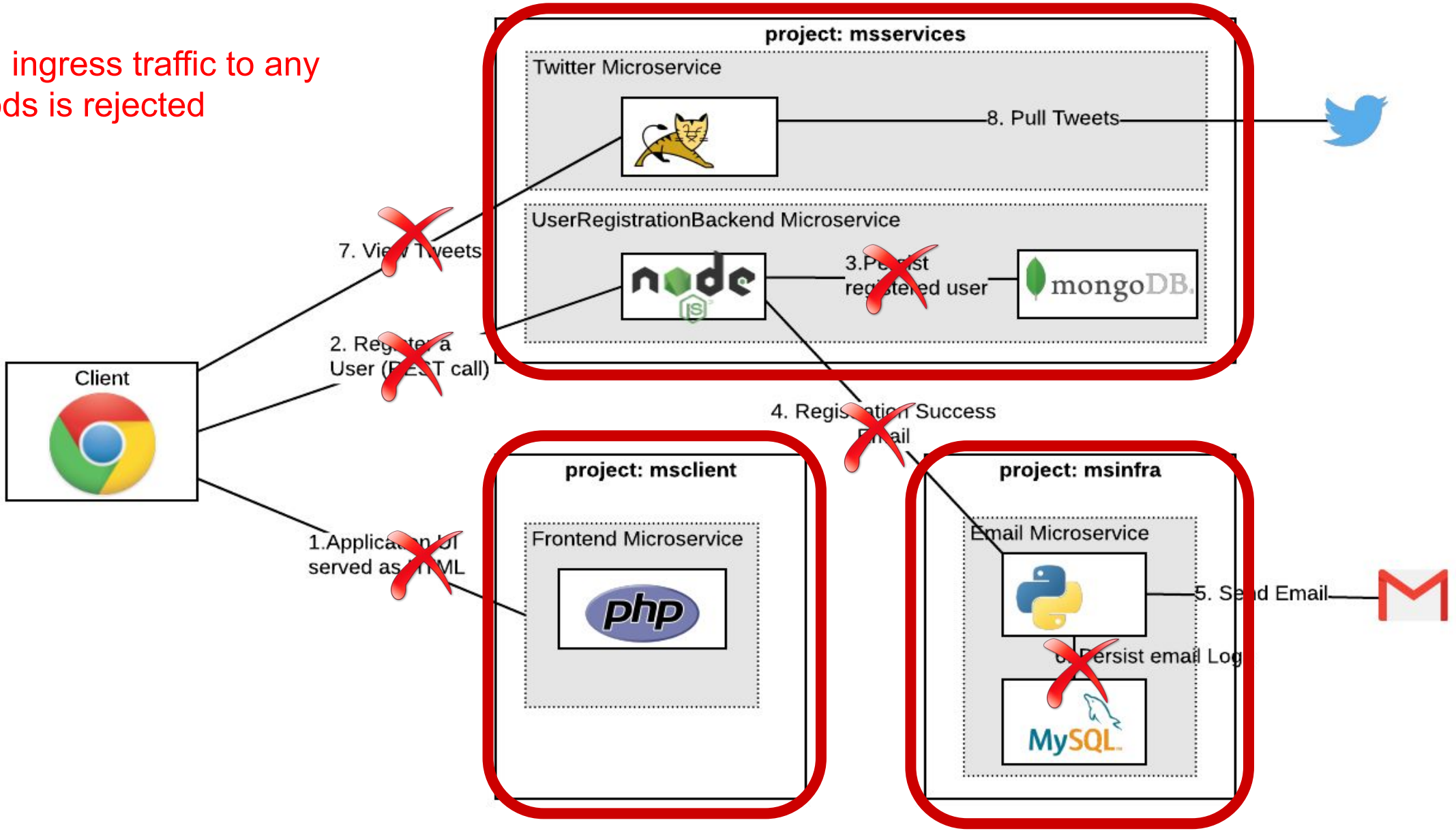
Allow MySQLDB connection from Email Service

```
kind: NetworkPolicy
apiVersion: extensions/v1beta1
metadata:
  name: allow-3306
spec:
 podSelector:
   matchLabels:
     app: mysql
 ingress:
 - from:
   - podSelector:
       matchLabels:
         app: emailsvc
   ports:
   - protocol: TCP
     port: 3306
```

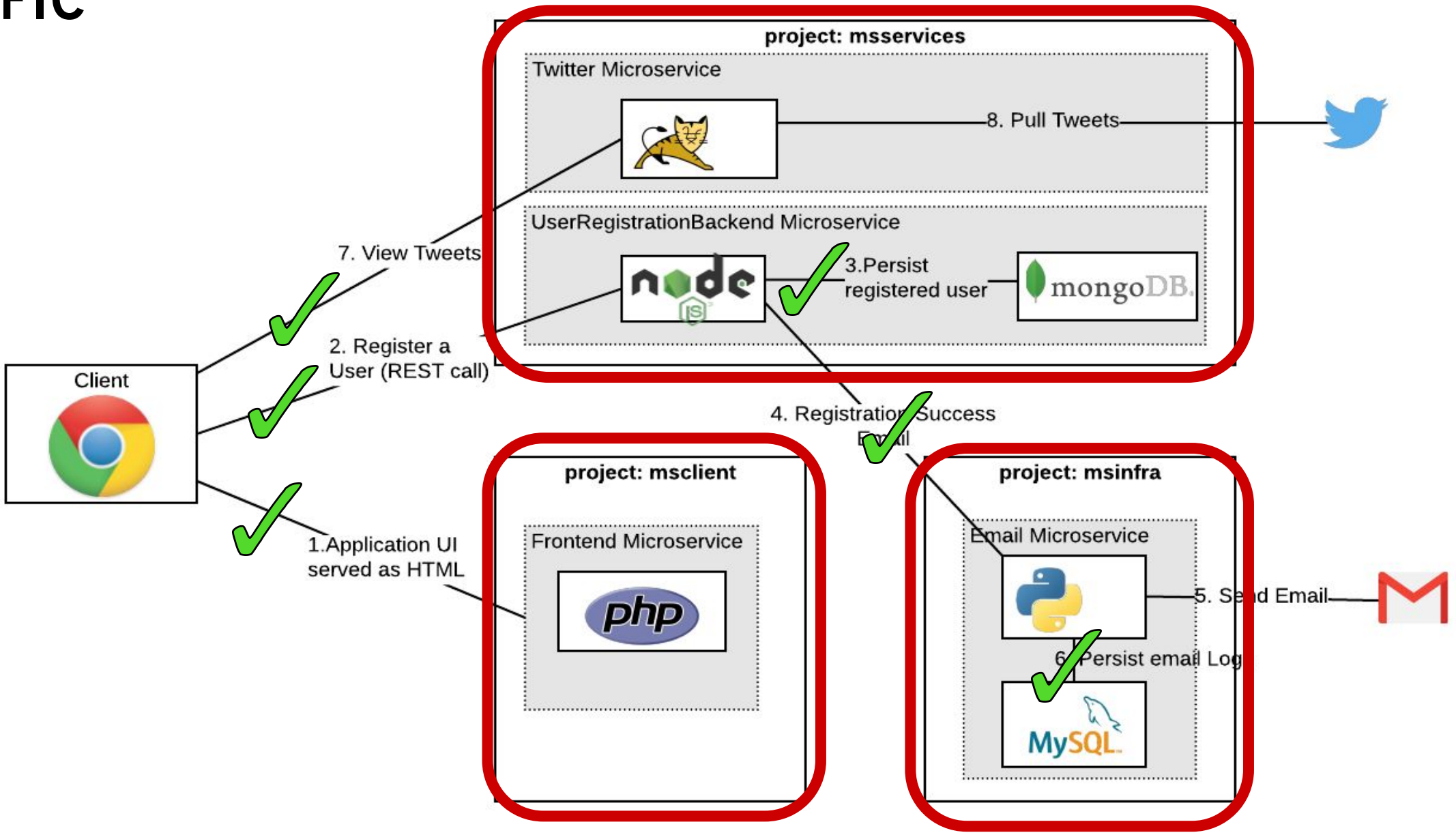
Red Hat

START WITH DEFAULT DENY

All ingress traffic to any pods is rejected

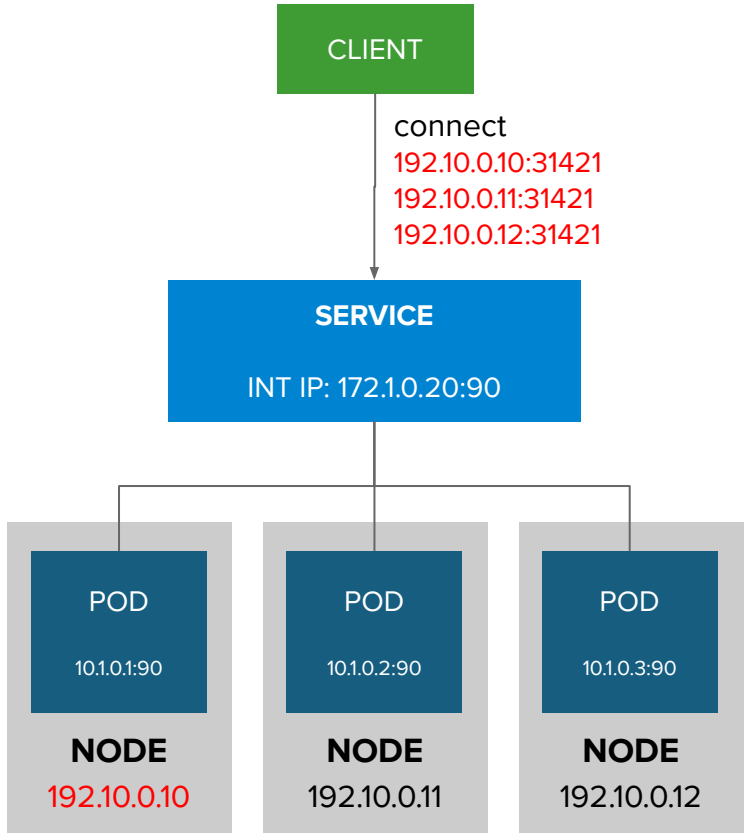


ADD NETWORK POLICIES TO ALLOW SPECIFIC INCOMING TRAFFIC



EXTERNAL TRAFFIC TO A SERVICE ON A RANDOM PORT WITH NODEPORT

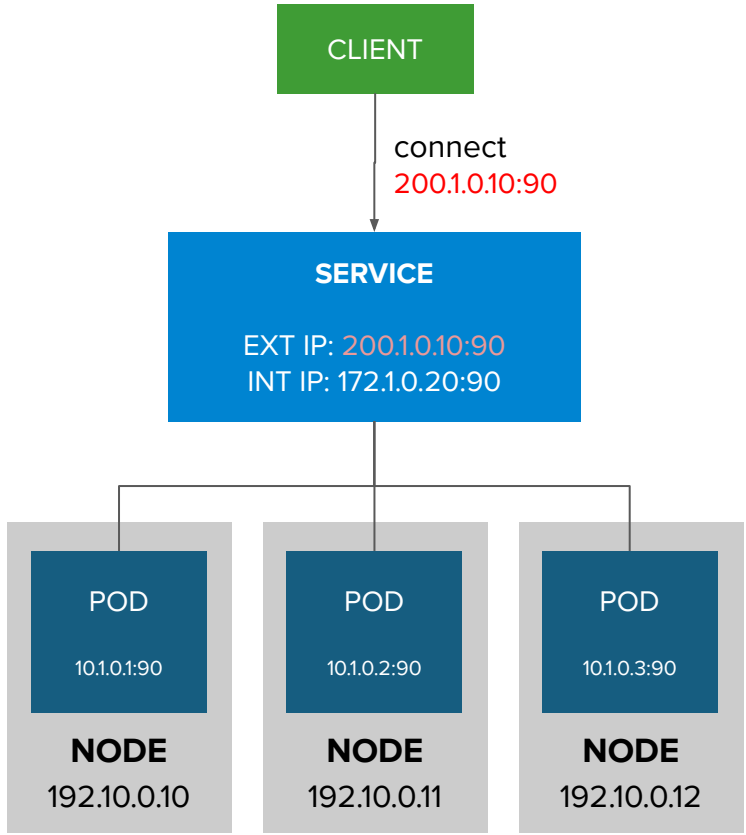
- NodePort binds a service to a unique port on all the nodes
- Traffic received on any node redirects to a node with the running service
- Ports in 30K-60K range which usually differs from the service
- Firewall rules must allow traffic to all nodes on the specific port



Red Hat

EXTERNAL TRAFFIC TO A SERVICE ON ANY PORT WITH INGRESS

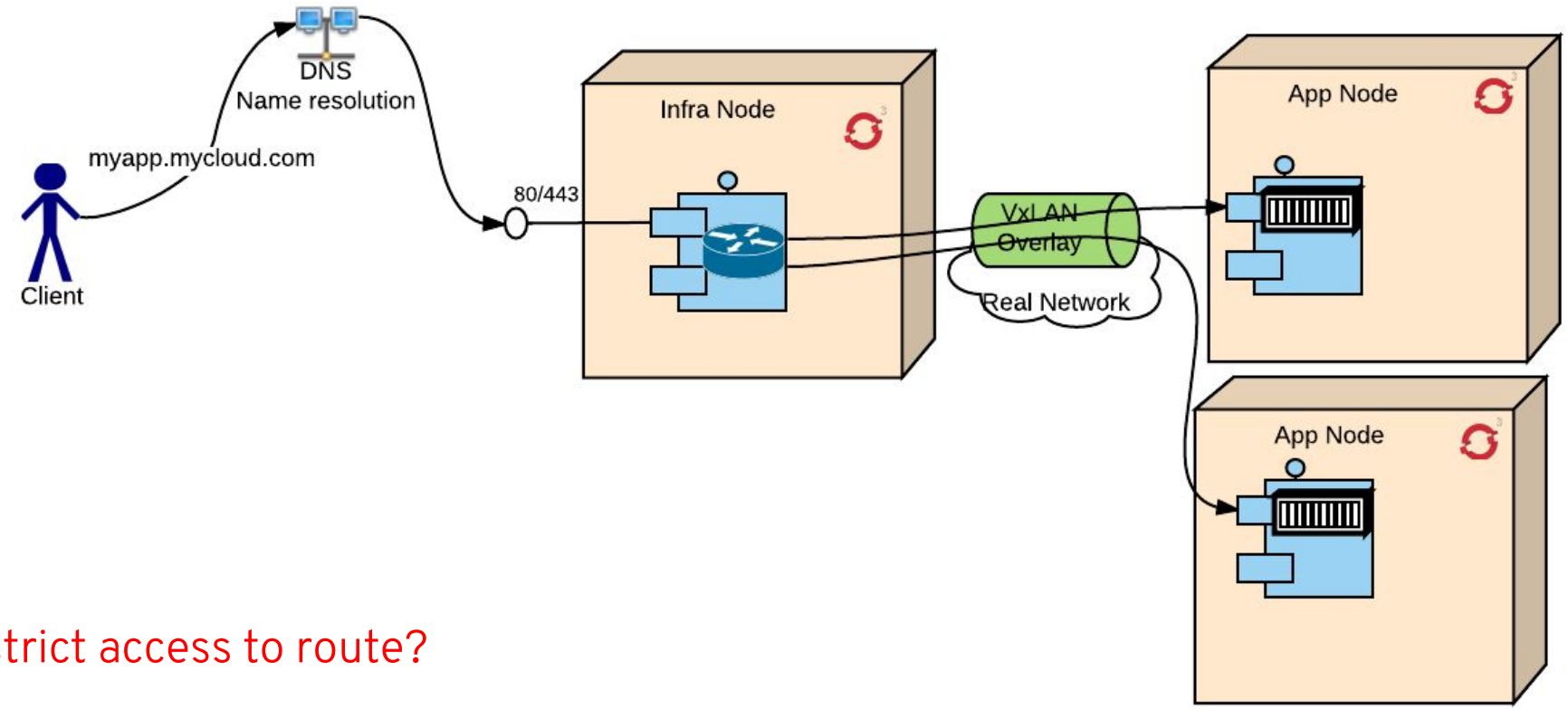
- Access a service with an external IP on any TCP/UDP port, such as
  - Databases
  - Message Brokers
- Automatic IP allocation from a predefined pool using Ingress IP Self-Service
- IP failover pods provide high availability for the IP pool



Red Hat

Securing INGRESS

OPENSHIFT ROUTER AS INGRESS



Can I restrict access to route?

Red Hat

ROUTE SPECIFIC IP WHITELISTS

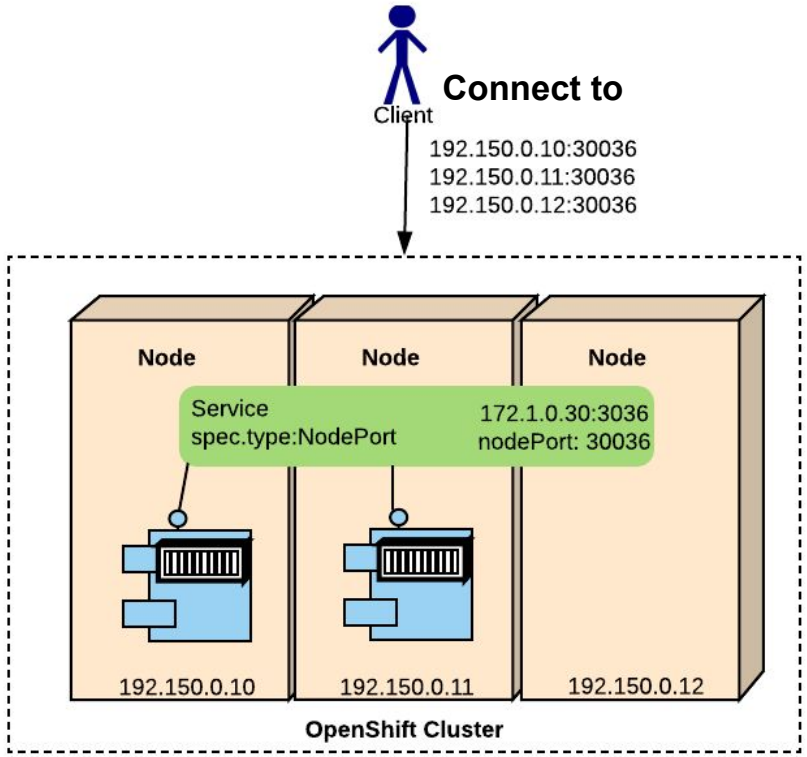
- Restrict access to a route to a select IP address(es)
- Annotate the route with the whitelisted/allowed IP addresses
- Connections from any other IPs are blocked

```
metadata:
  annotations:
    haproxy.router.openshift.io/ip_whitelist: 192.168.1.10 192.168.1.11
```

What about ingress traffic on ports that are not 80 or 443?

Red Hat

USING NODEPORT AS INGRESS TO SERVICE



Binds service to a unique port on every node in the cluster

Port randomly assigned or optionally picked from port range 30000-32767

All nodes act as ingress point at the port assigned

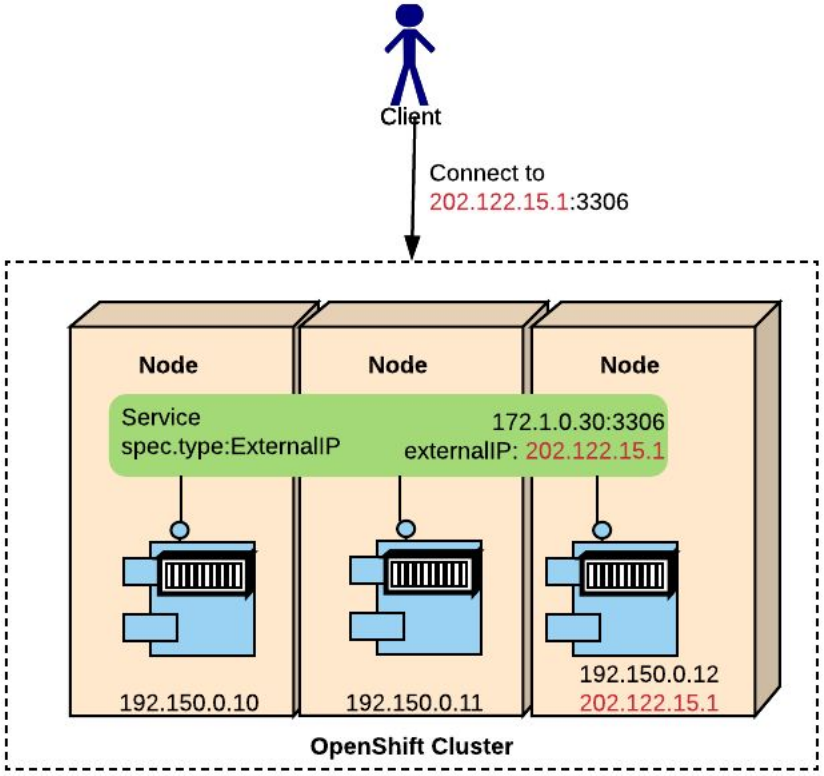
Every node in the cluster redirects traffic to service service endpoints even if a corresponding pod is not running on that node

Firewall rules should not prevent nodes listening on these ports

Every exposed service uses up a port on all the nodes in a cluster. Are there alternatives?

Red Hat

ASSIGNING EXTERNAL IP TO A SERVICE WITH INGRESS



Admin defines External-IP address range. Assigns these extra IPs to nodes.

OpenShift assigns both internal IP and external IP to a service. Or a specific External IP can be chosen.

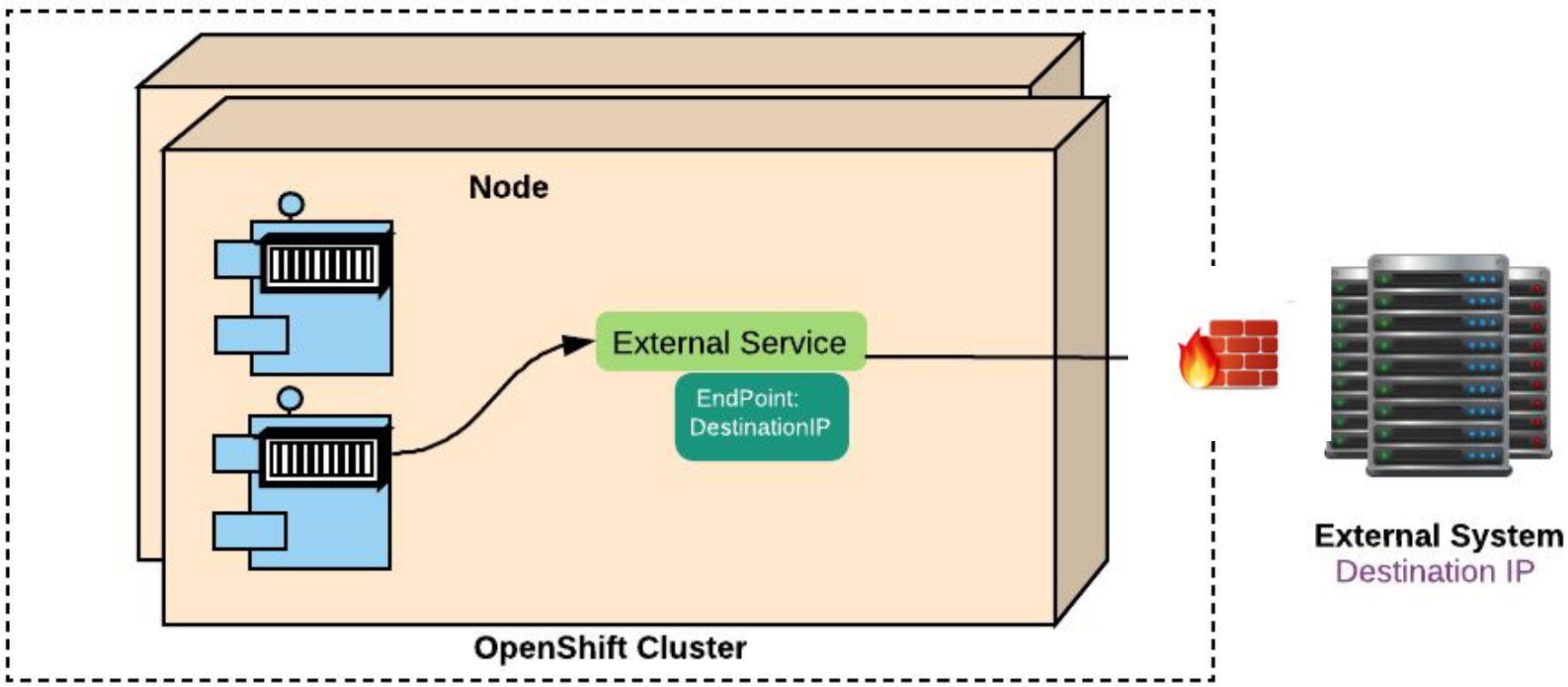
Node to which External-IP is assigned acts as the ingress point to the service.

External-IP can be a VIP. You can set up ipfailover to reassign VIP to other nodes. Ipfailover runs as a privileged pod and handles VIP assignment.

Red Hat

Securing EGRESS

CONNECTING VIA EXTERNAL SERVICE

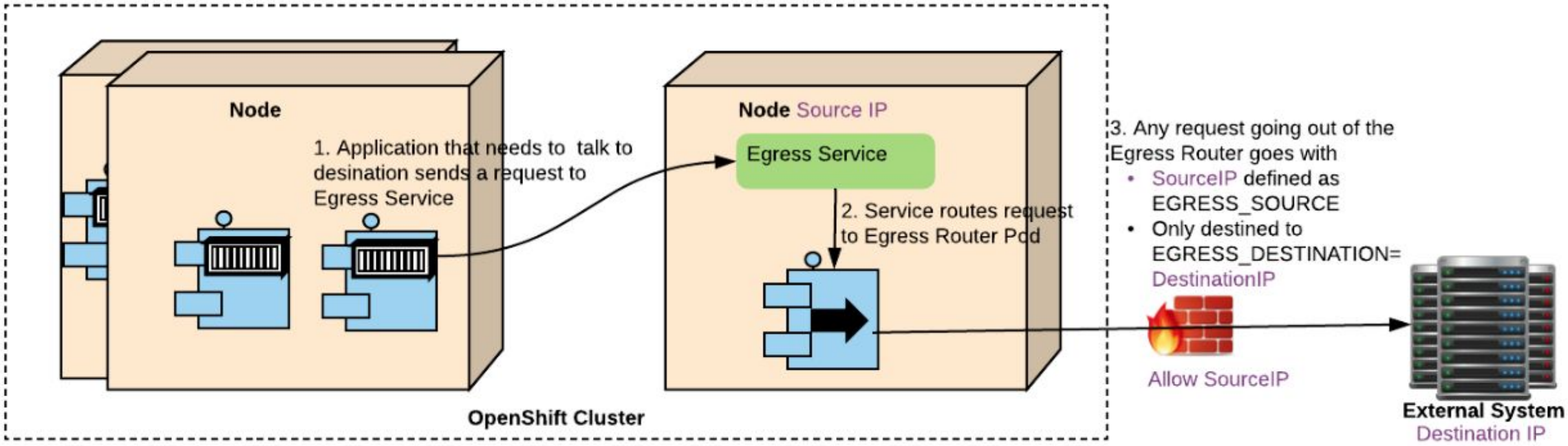


Application connecting to External System talks to an External Service whose Endpoint is set as Destination IP & Port (or FQDN of the external system and port)

But, what if we have a firewall in front of the External System that allows only Specific IPs?

Red Hat

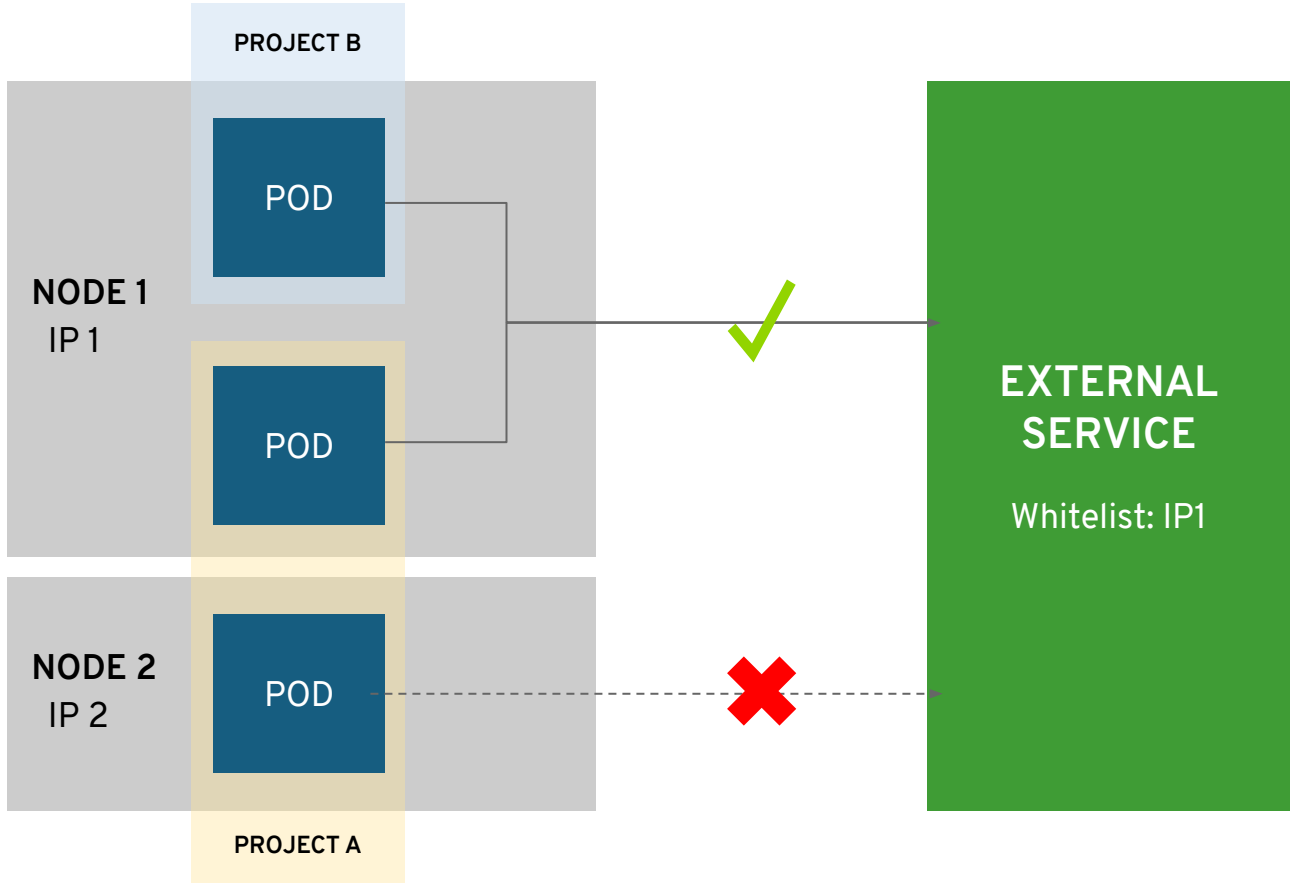
CONNECTING VIA EGRESS ROUTER



Red Hat

CONTROLLING EGRESS TRAFFIC

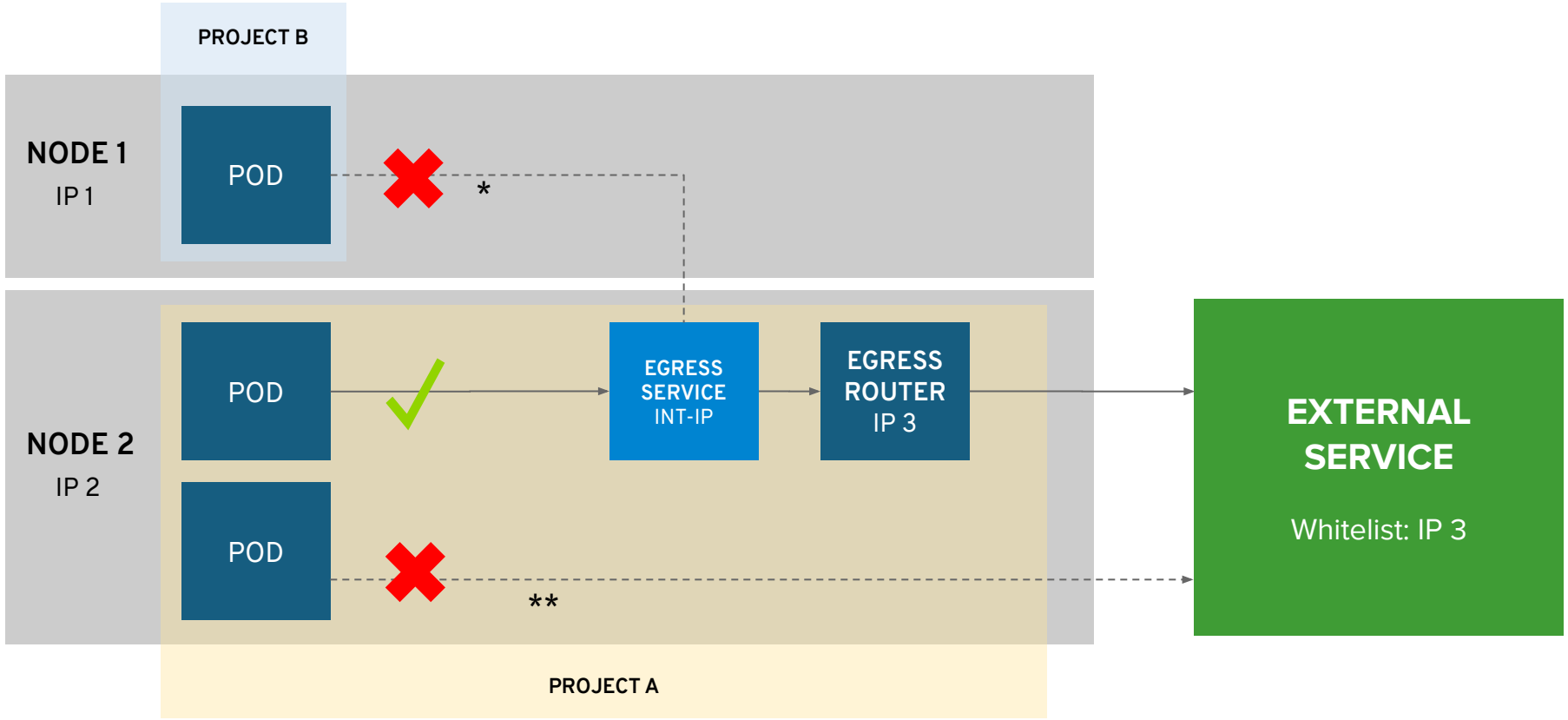
Default Kubernetes Behaviour



Red Hat

CONTROLLING EGRESS TRAFFIC

Egress Router

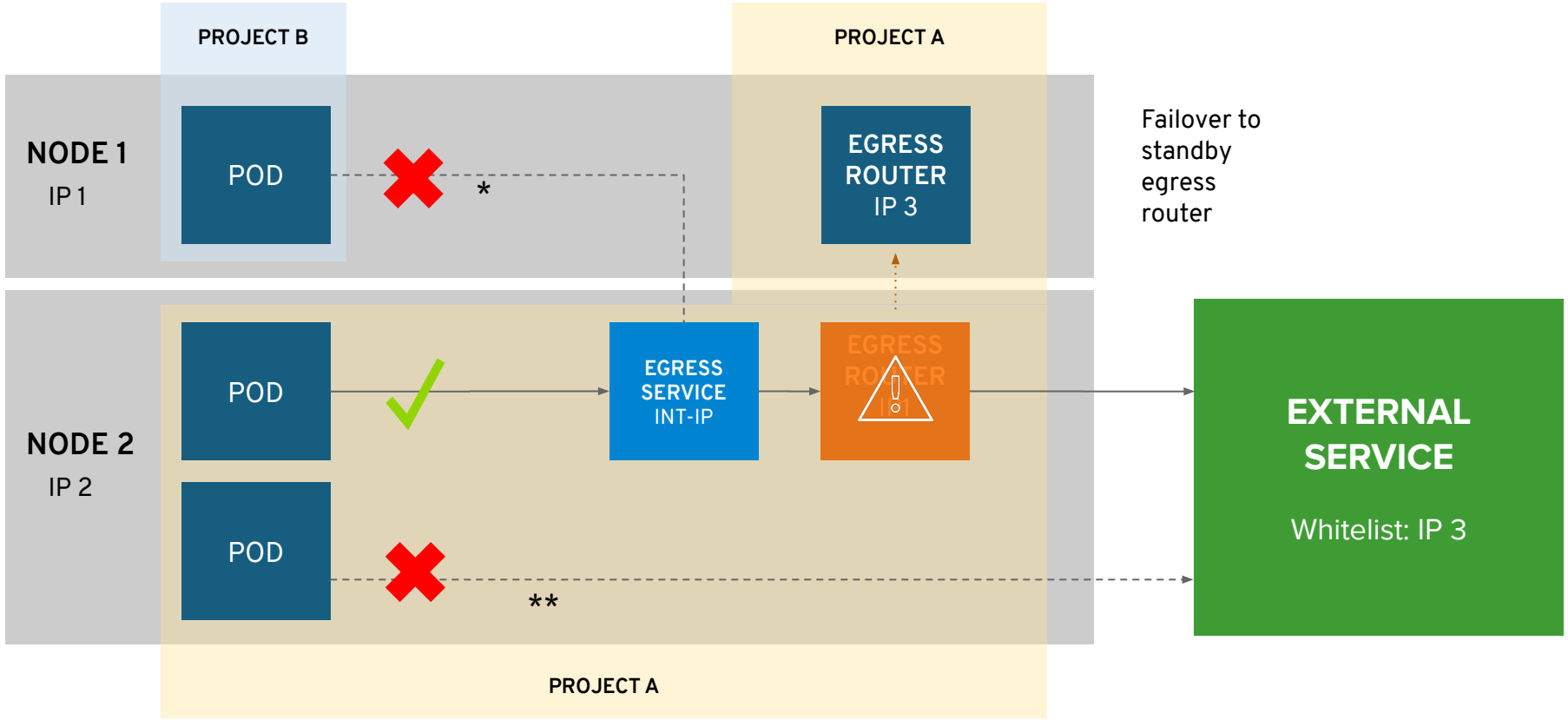


* Blocked by multi-tenant network plugin
** Blocked by external service

Red Hat

CONTROLLING EGRESS TRAFFIC

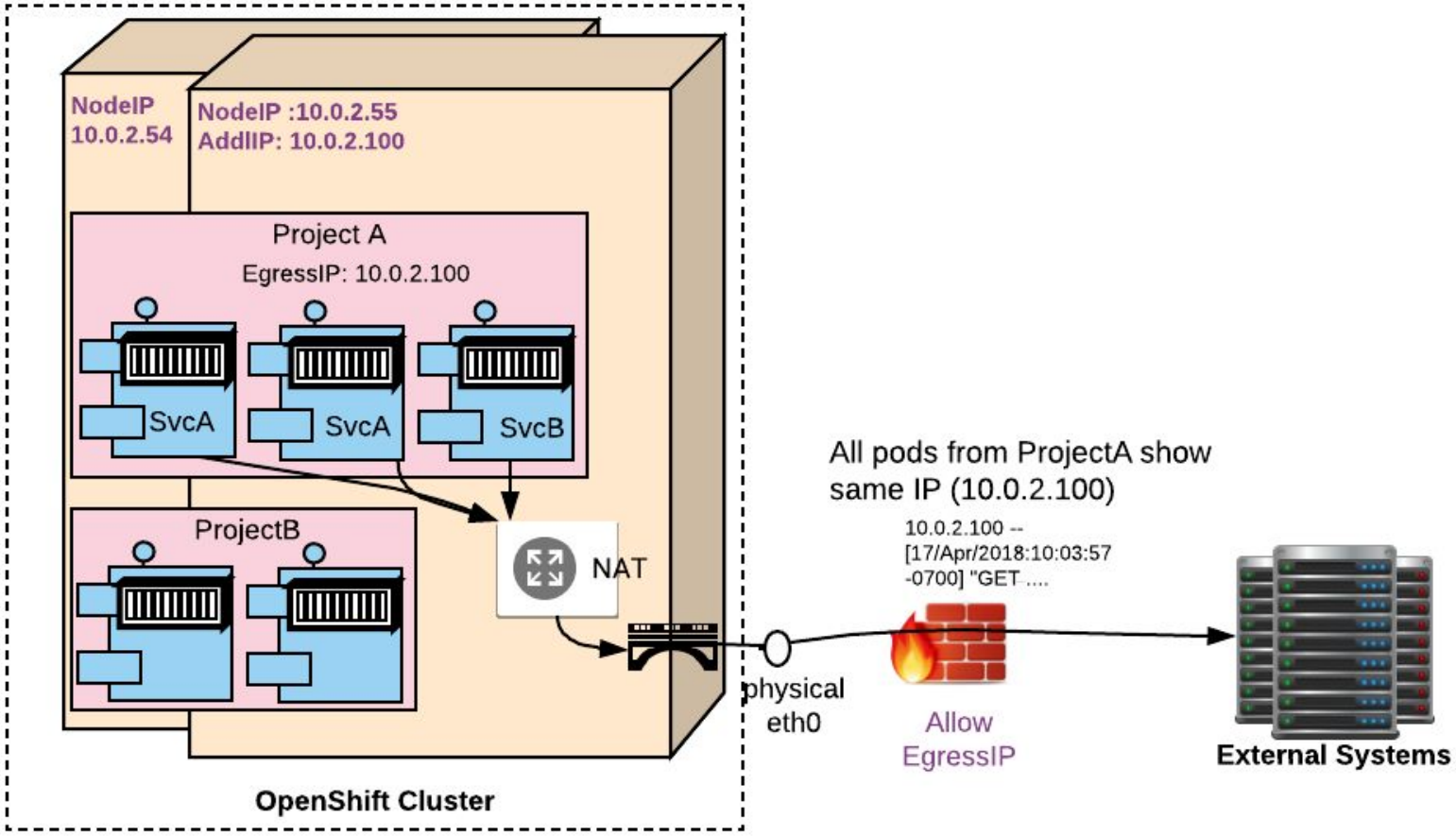
Egress Router



* Blocked by multi-tenant network plugin
** Blocked by external service

Red Hat

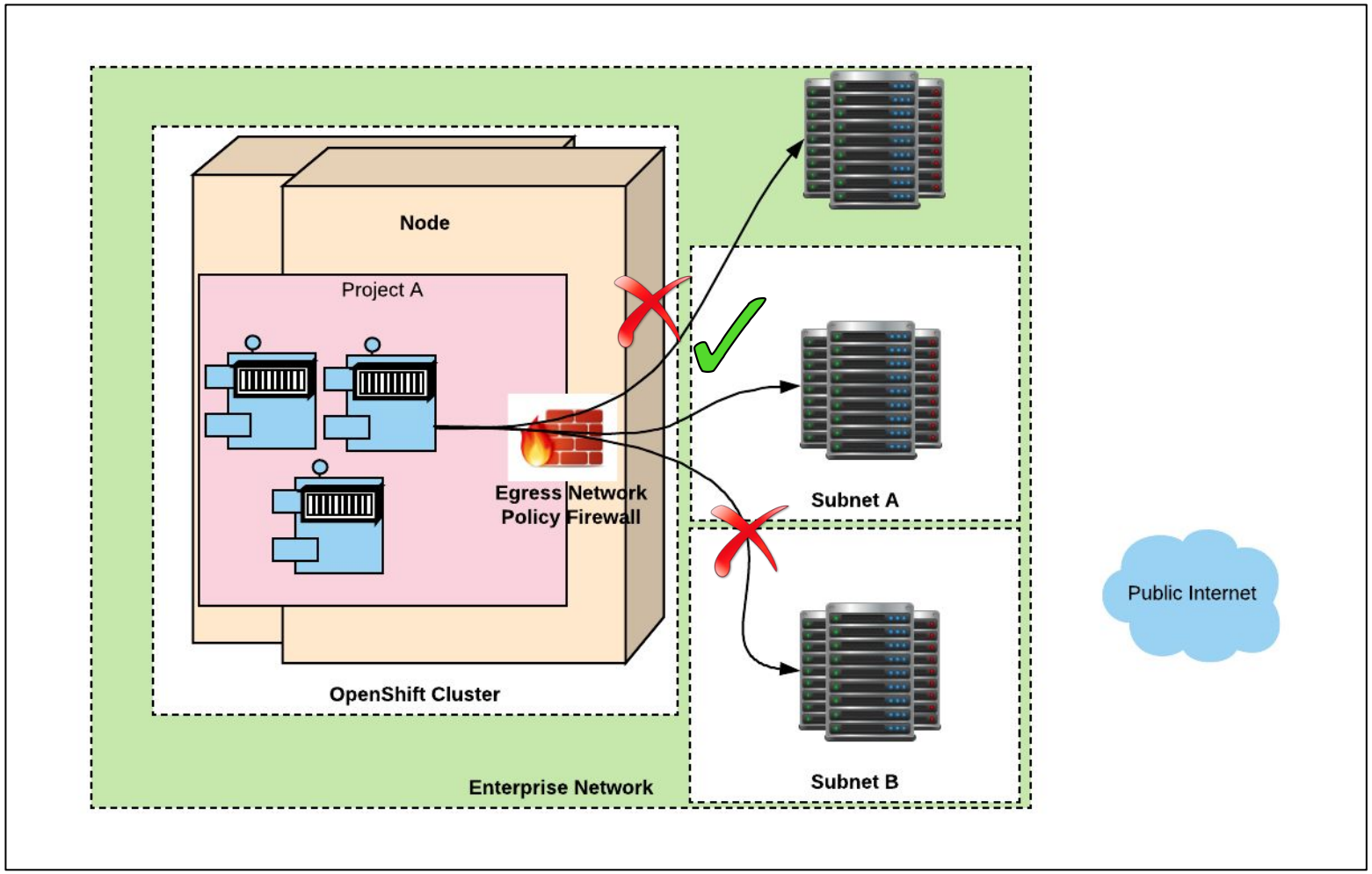
STATIC IP FOR ALL TRAFFIC FROM A PROJECT



Red Hat

EGRESS FIREWALL TO LIMIT ACCESS

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



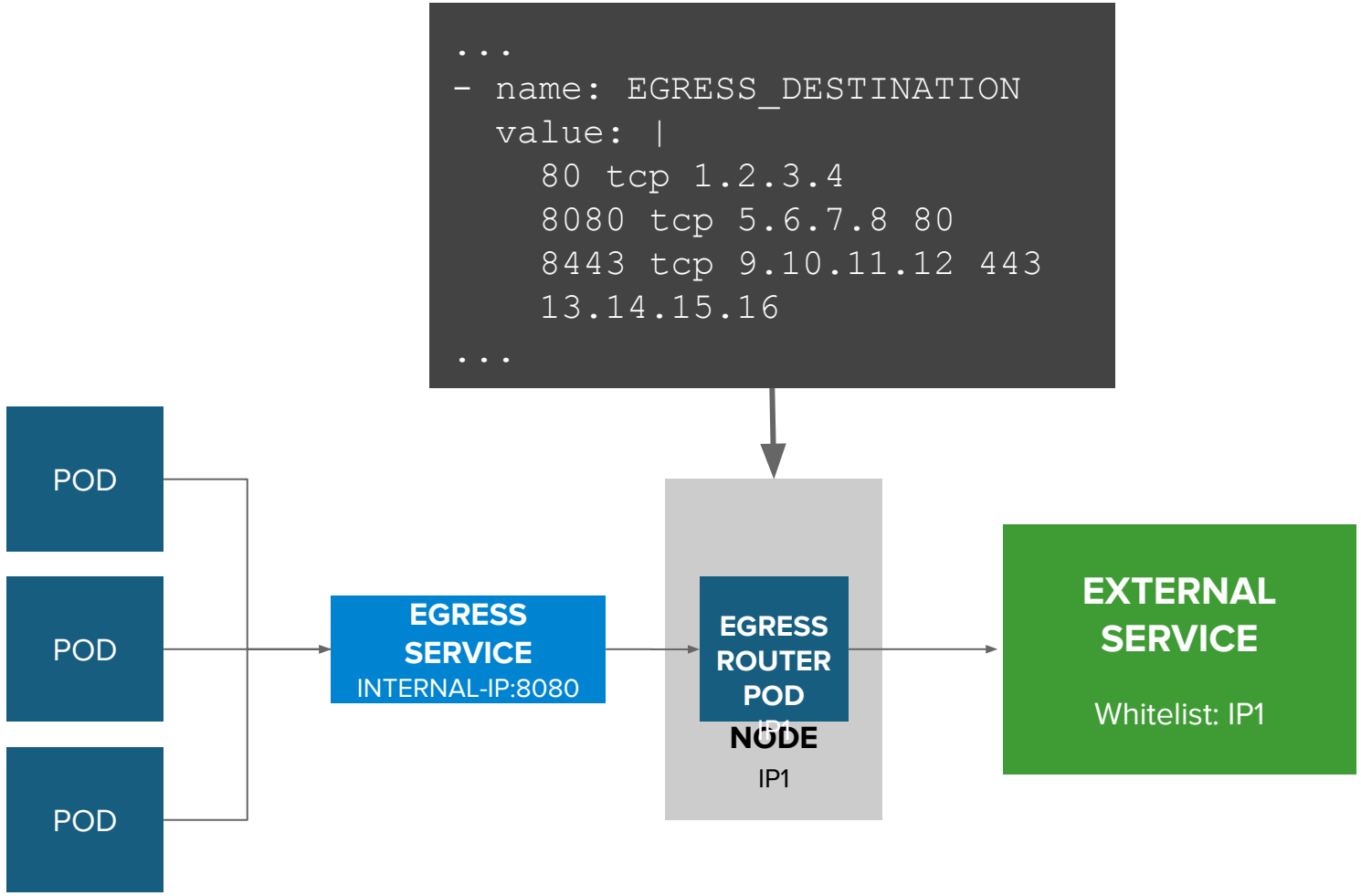
Examples:

- 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

Red Hat

NETWORK DEFENSE: EGRESS ROUTER

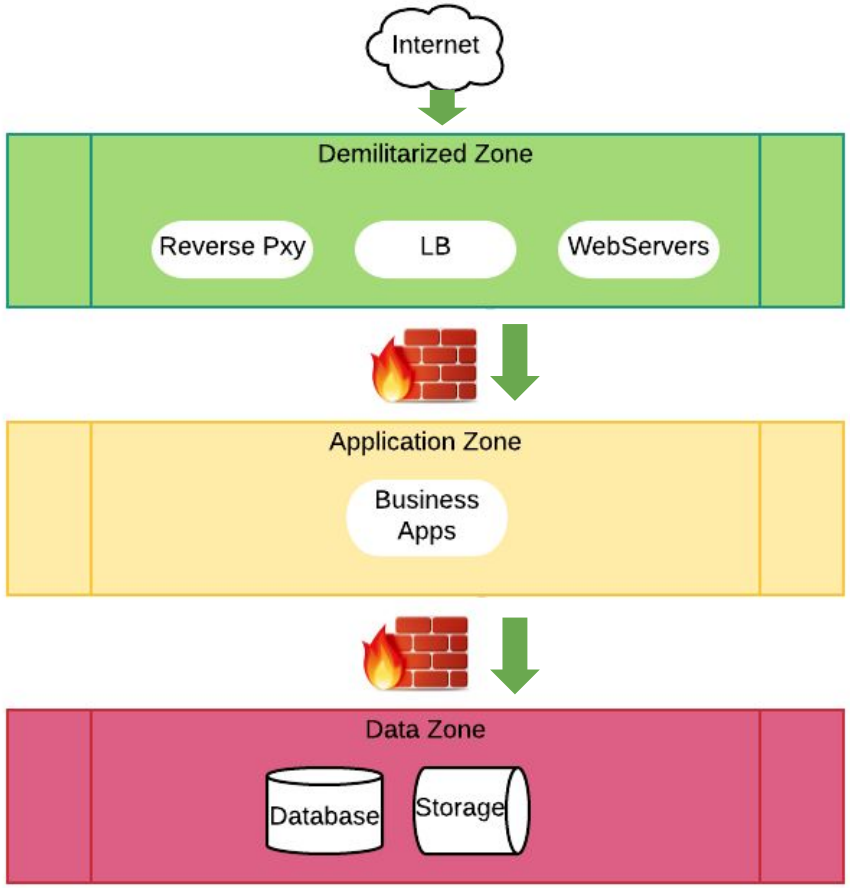
The OpenShift egress router runs a service that redirects egress pod traffic to one or more specified remote servers, using a pre-defined source IP address that can be whitelisted on the remote server. The egress router can also be run as a proxy.



Red Hat

Architecture DMZ
Isolated Zones

NETWORK ZONES SEPARATED BY FIREWALLS



External traffic allowed to touch DMZ

Holes punched in firewalls to allow specific traffic from

- DMZ to Application Zone

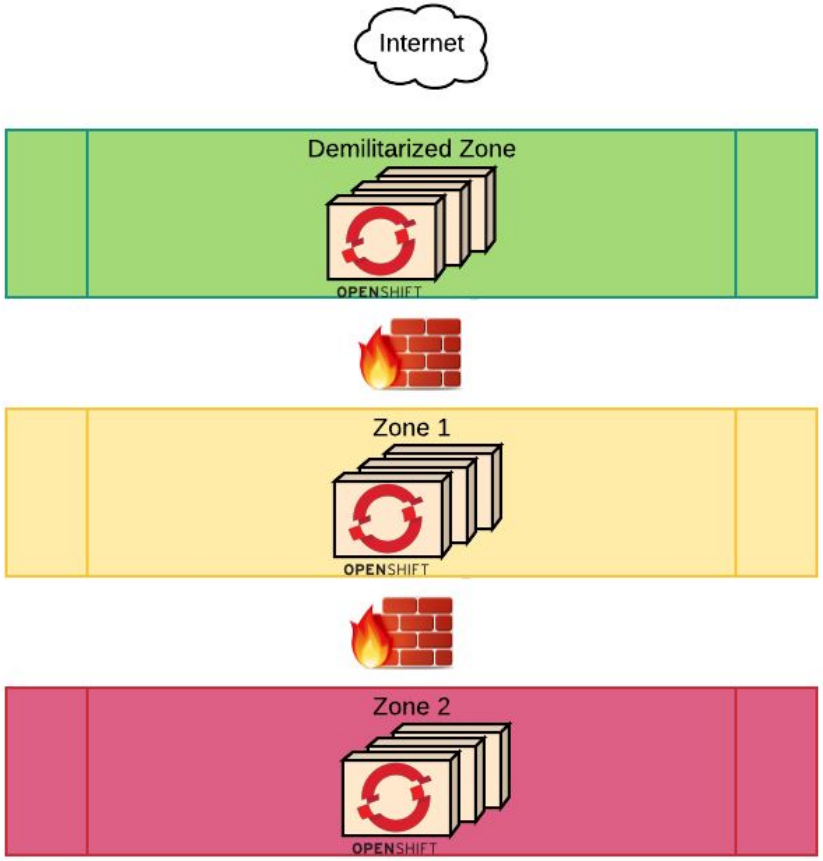
and from

- Application Zone to Data Zone

How do I setup OpenShift here?

Red Hat

OPTION 1: OPENSHIFT CLUSTER PER ZONE



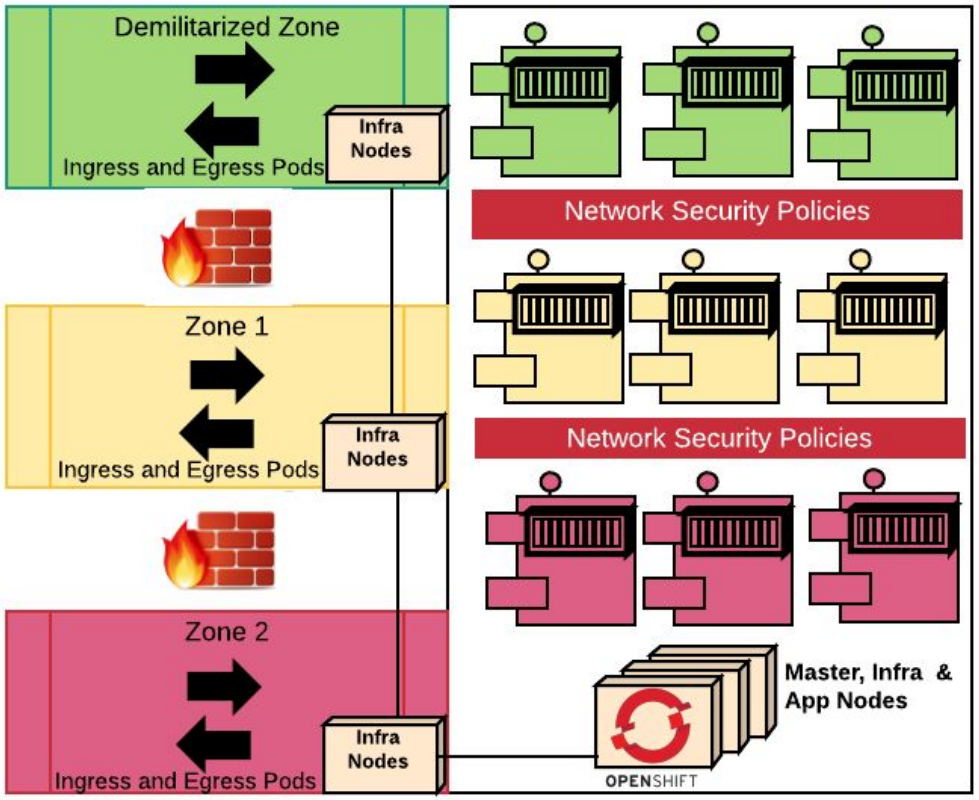
Useful to demonstrate compliance with Security Standards and Regulations

Additional actions needed to protect Master APIs, and other URLs in DMZ that are not supposed to be exposed to Internet

Cost of maintenance is high

Red Hat

OPTION 2 :
OPENSHIFT CLUSTER COVERING MULTIPLE ZONES



Application pods run on one OpenShift Cluster. Micro-segmented 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 defeats the purpose of a shared cluster. Micro-segmentation with SDN is the way to go.

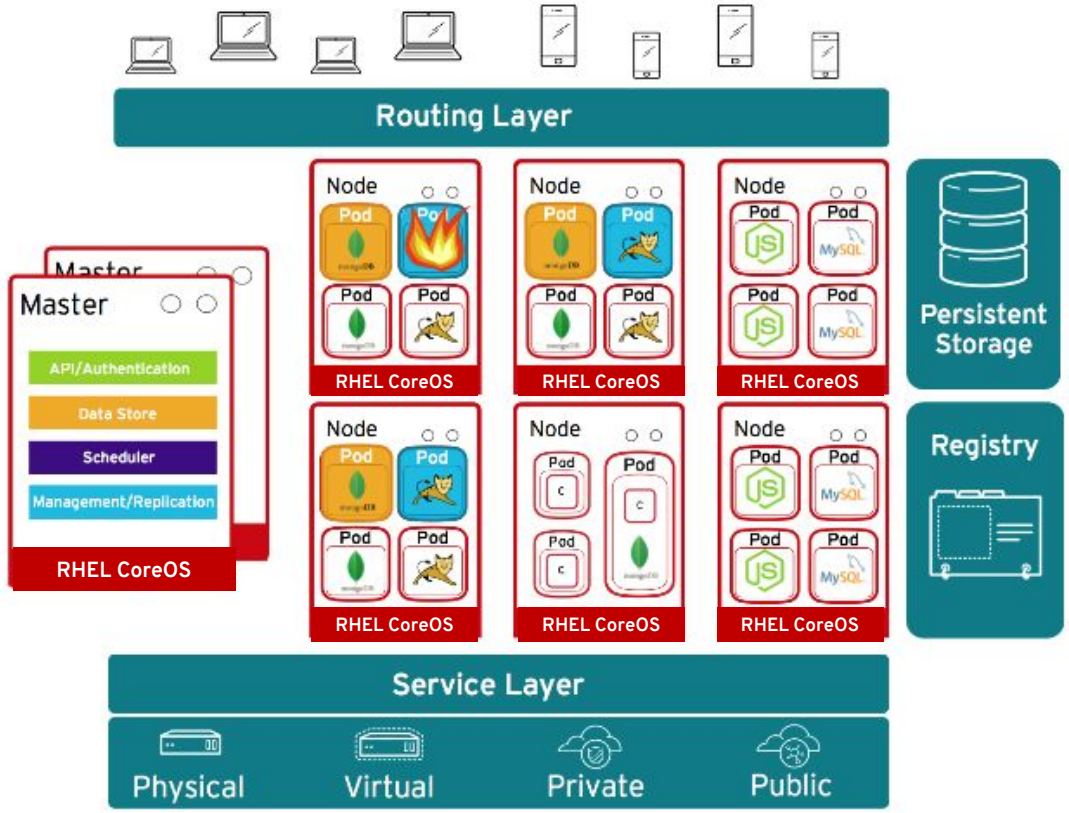
Red Hat

Stockage persistent des containeurs sécurisation des accès

ATTACHED STORAGE

Secure storage by using

- SELinux access controls
- Secure mounts
- Supplemental group IDs for shared storage



Red Hat

ATTACHED STORAGE

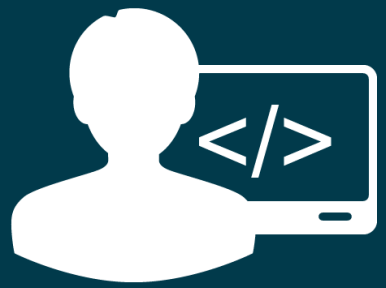
SCCs are also very useful for managing access to persistent storage.

Controlling Volumes

The usage of specific volume types can be controlled by setting the `volumes` field of the SCC.

- azureFile
- azureDisk
- flocker
- flexVolume
- hostPath
- emptyDir
- gcePersistentDisk
- awsElasticBlockStore
- gitRepo
- secret
- nfs
- iscsi
- glusterfs
- persistentVolumeClaim
- rbd
- cinder
- cephFS
- downwardAPI
- fc
- configMap
- vsphereVolume
- quobyte
- photonPersistentDisk
- projected
- portworxVolume
- scaleIO
- storageos

STORAGE ISOLATION



Create app with storage

SCC access Layer

supplementalGroups	fsGroup	seLinuxOption	runAsUser

Check for UID/GIDfor access to shared storage?

Is the pod's "file system group" ID correct for the block storage?

Is the seLinuxContext user, role,type set and is this user allowed to mount it?

What is the RunAsUser or MustRunAsRange?

STORAGE ISOLATION

Admin provisions storage

```
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv0001
spec:
  capacity:
    storage: 10
  persistentDisk:
    pdName: "abc123"
    fsType: "ext4"
```

User requests storage

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: myclaim-1
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3
```

Claim usage

```
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - image: nginx
      name: myfrontend
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      source:
        persistentVolumeClaim:
         accessMode: ReadWriteOnce
         claimRef:
           name: myclaim-1
```

Information sur les certifications FISMA ISO27000 PCI-DSS Hardening Présentation de Red Hat CoreOS - socle "immutable" qui réduit la surface d'attaque

HARDENING TOOLS & APPLICABILITY GUIDES

Product Applicability Guides

- FISMA Moderate & FISMA (NIST)
- ISO 27001
- PCI-DSS
- PCI-DSS Reference Architecture

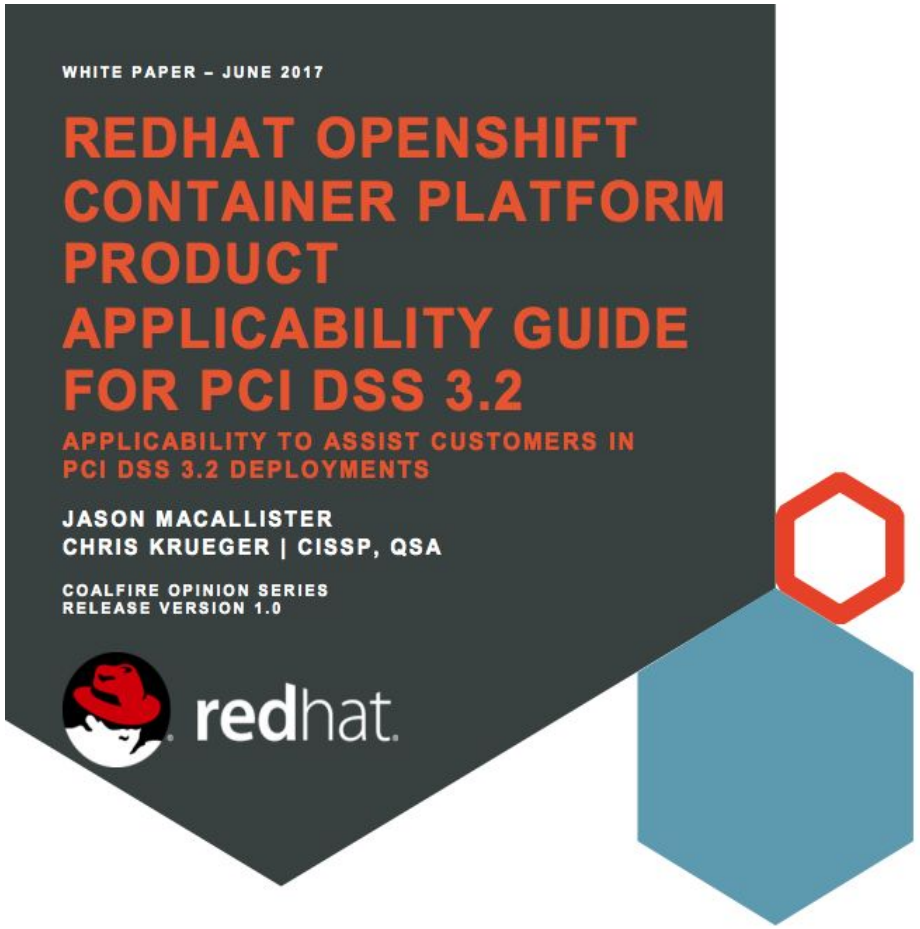
OpenShift Hardening Guide for 3.10 & 3.11 - OpenShift 4 Guide planned for Fall 2019

Upstream, 3rd party tools

- docker-bench supports versions 1.13 and 17.06
- kube-bench → WIP, support for OCP 3.9 & 3.10

https://orocktech.com/company/news/2019-07-23/

PCI-DSS Product Applicability Guide for Red Hat Products



NIST National Checklist for Red Hat OpenShift Container Platform 3.x content v0.1.44

Checklist Details (Checklist Revisions)

SCAP 1.3 Content:

- Download SCAP 1.3 Content - NIST National Checklist Collection for Red Hat Products with SCAP 1.3
  - Author: Red Hat

SCAP 1.2 Content:

- Download SCAP 1.2 Content - NIST National Checklist Collection for Red Hat Products with SCAP 1.2
  - Author: Red Hat

Supporting Resources:

- Download Security Template - NIST 800-53/FISMA Applicability Guide for OpenShift 3.x
  - Red Hat
- Download Prose - OpenShift Security Configuration Guide (HTML)
  - Red Hat
- Download Machine-Readable Format - OpenControl-formatted NIST 800-53/FISMA Applicability Guide for OpenShift 3.x
  - Red Hat

Target:

Target	CPE Name

CHECKLIST HIGHLIGHTS

Checklist Name: NIST National Checklist for Red Hat OpenShift Container Platform 3.x
Checklist ID: 866
Version: content v0.1.44
Type: Compliance
Review Status: Final
Authority: Software Vendor: Red Hat
Original Publication Date: 05/03/2019

Center for Internet Security (CIS)

https://www.cisecurity.org/

The Center for Internet Security (CIS) is a 501(c)(3) nonprofit organization, formed in October, 2000. Its mission is to "identify, develop, validate, promote, and sustain best practice solutions for cyber defense and build and lead communities to enable an environment of trust in cyberspace".

The organization is headquartered in East Greenbush, New York, with members including large corporations, government agencies, and academic institutions.[1]

[1] https://en.wikipedia.org/wiki/Center_for_Internet_Security

CIS benchmarks

https://www.cisecurity.org/cis-benchmarks/

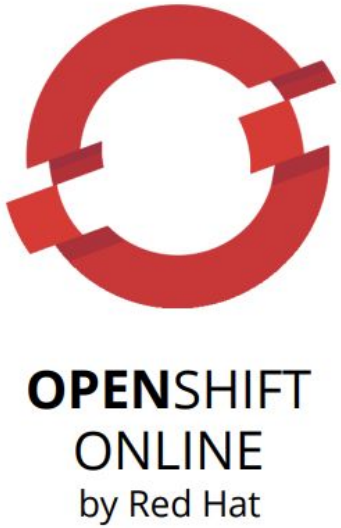
CIS provides benchmarks:

- Configuration guidelines
- Different technology groups:
  - Servers
  - Network devices
  - ....

Mapping CIS Benchmark Criteria to Openshift

Recommendations	Relevance	OpenShift Compliance	Comments	
1.1 Create a separate partition for containers	Scored	**Yes**	OpenShitft uses a separate logical volume for all containers data and metadata	
1.2 Use the updated Linux Kernel	Scored	**Yes**	OpenShift runs on Red Hat Enterprise Linux 7 with linux kernel version 3.10 or higher	
1.3 Harden the container host	Not Scored	**Yes**	OpenShift runs on Red Hat Enterprise Linux 7 and leverages SELinux for isolating containers. Red Hat Enterprise Linux 7 documentation includes a comprehensive security guide including instructions for hardening the operating system: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/ Furthermore, OpenShift is certified on Red Hat Enterprise Linux Atomic Host which is purpose built, lightweight, and minimal-footprint operating system optimized to run Linux containers.	
1.4 Remove all non-essential services from the host	Not Scored	**Yes**	OpenShift is certified on Red Hat Enterprise Linux Atomic Host which is purpose built, lightweight, and minimal-footprint operating system optimized to run Linux containers.	
1.5 Keep Docker up to date	Not Scored	**Yes**	OpenShift includes Red Hat Enterprise Linux, which packages a stable and fully supported version of Docker Engine. Red Hat maintains the Docker runtime for OpenShift and Red Hat Enterprise Linux users as part of their subscription and this includes relevant bugfixes and security updates.	
1.6 Only allow trusted users to control Docker daemon	Scored	**Yes**	OpenShift does not allow users to have direct access to Docker on the nodes (hosts). Users obtain access to run containers via the role-based access control system in OpenShift without directly accessing the Docker daemon.	
1.7 Audit docker daemon	Scored	**Yes**	OpenShift does not configure Red Hat Enterprise Linux auditing, but the platform admin can turn on auditing as described in Red Hat Enterprise Linux Security Guide: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/	
1.8 Audit Docker files and directories - /var/lib/docker	Scored	**Yes**	OpenShift does not configure Red Hat Enterprise Linux auditing, but the platform admin can turn on auditing as described in Red Hat Enterprise Linux Security Guide: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/html/Security_Guide/	

Annual External Penetration Testing of Openshift Online



Findings are applicable to Openshift Container Platform

External Penetration Testing Completed August 2016

2018 Penetration Testing results for OpenShift completed by Product Security currently in Red Hat Legal's hands

In August 2016, Red Hat and the OpenShift team commissioned an external penetration test of the Developer Preview environment and a typical OpenShift Dedicated cluster. The testing has completed and the Red Hat team is reviewing the test results. The test results are confidential to Red Hat and there is no plan to publish a comprehensive report, however, a summary of the testing, results and current state are documented here.

Red Hat

Ecosystème Sécurité K8S

RED HAT CERTIFIED OPERATORS

DEVOPS	dynatrace
APM	APPDYNAMICS, INSTANA, New Relic., Sysdig
DATA SERVICES	GIGASPACES, hazelcast, PlanetScale
DATABASE	MEMSQL, Couchbase, mongoDB., NUODB, PingCAP
SECURITY	aqua, anchore, BLACKDUCK BY SYNOPSYS, TREMOLO SECURITY, tufin
STORAGE	ROBIN, STORAGEOS

When do I have to use a 3rd party vendor solution and who do I pick?


NGINX

Networking

Code Scanning


Black Duck

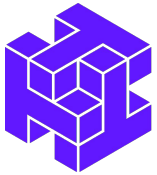
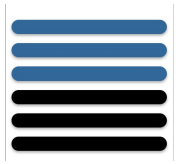

Twistlock

Image Scanning


Sonatype

JFrog

Repositories


Cyberark

Secrets Management


SysDig


Dynatrace

Monitoring

Merci

Red Hat est le leader mondial des solutions logicielles open source d'entreprise.
Un support reconnu, ainsi que des offres complètes de formation et de conseil font de Red Hat une référence pour un grand nombre de sociétés.


linkedin.com/company/red-hat


youtube.com/user/RedHatVideos


facebook.com/redhatinc


twitter.com/RedHat

Red Hat