

A complex, monochromatic red illustration on the left side of the slide. It features various icons representing cloud computing and container technology: a cloud with a keyhole, a database cylinder, a server rack, a computer monitor, and several arrows indicating data flow and system architecture. The background is a solid red color.

OPENSIFT CONTAINER PLATFORM

ARCHITECTURAL OVERVIEW



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



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



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



twitter.com/RedHat

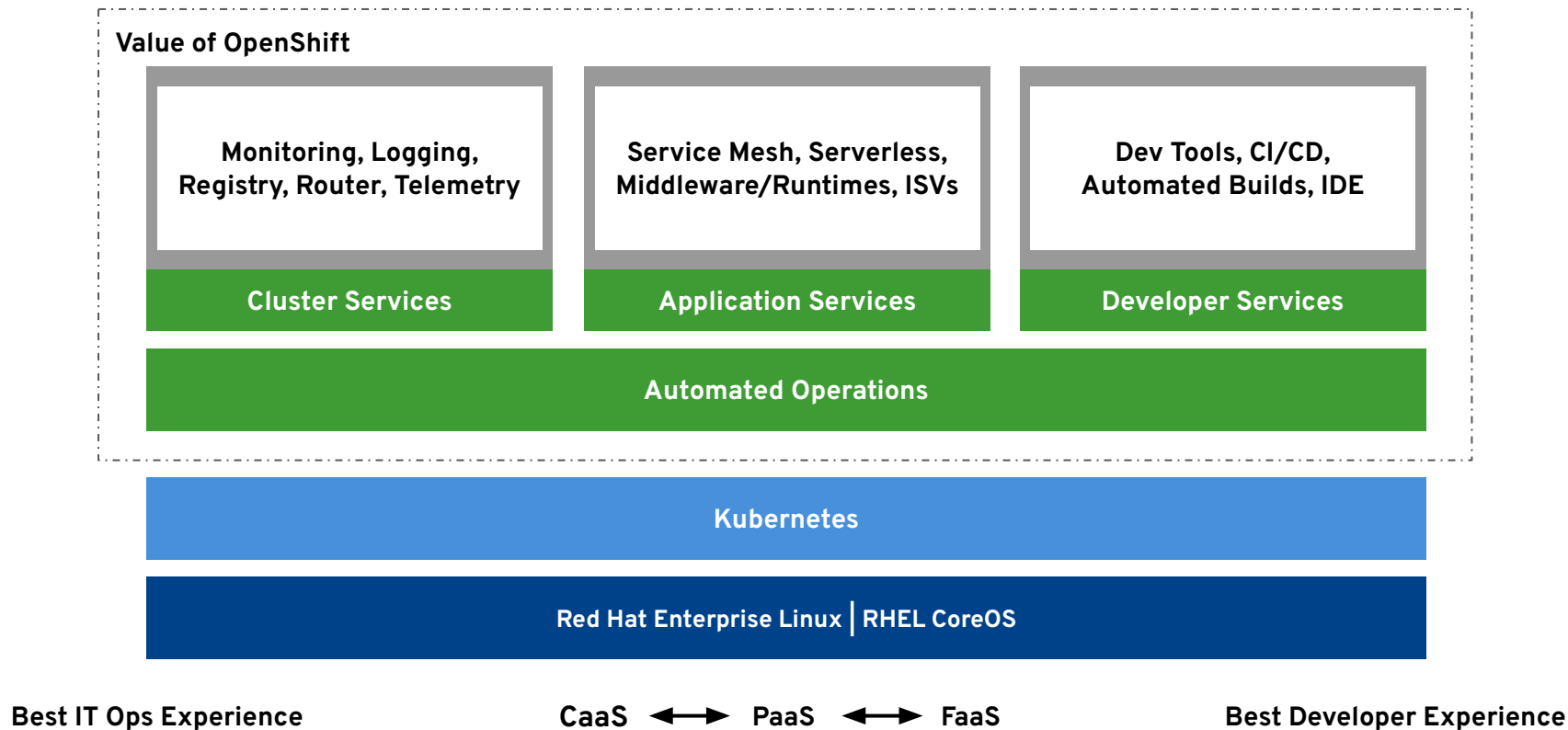
Alfred Bach
Principal Solution Architect
May 2022

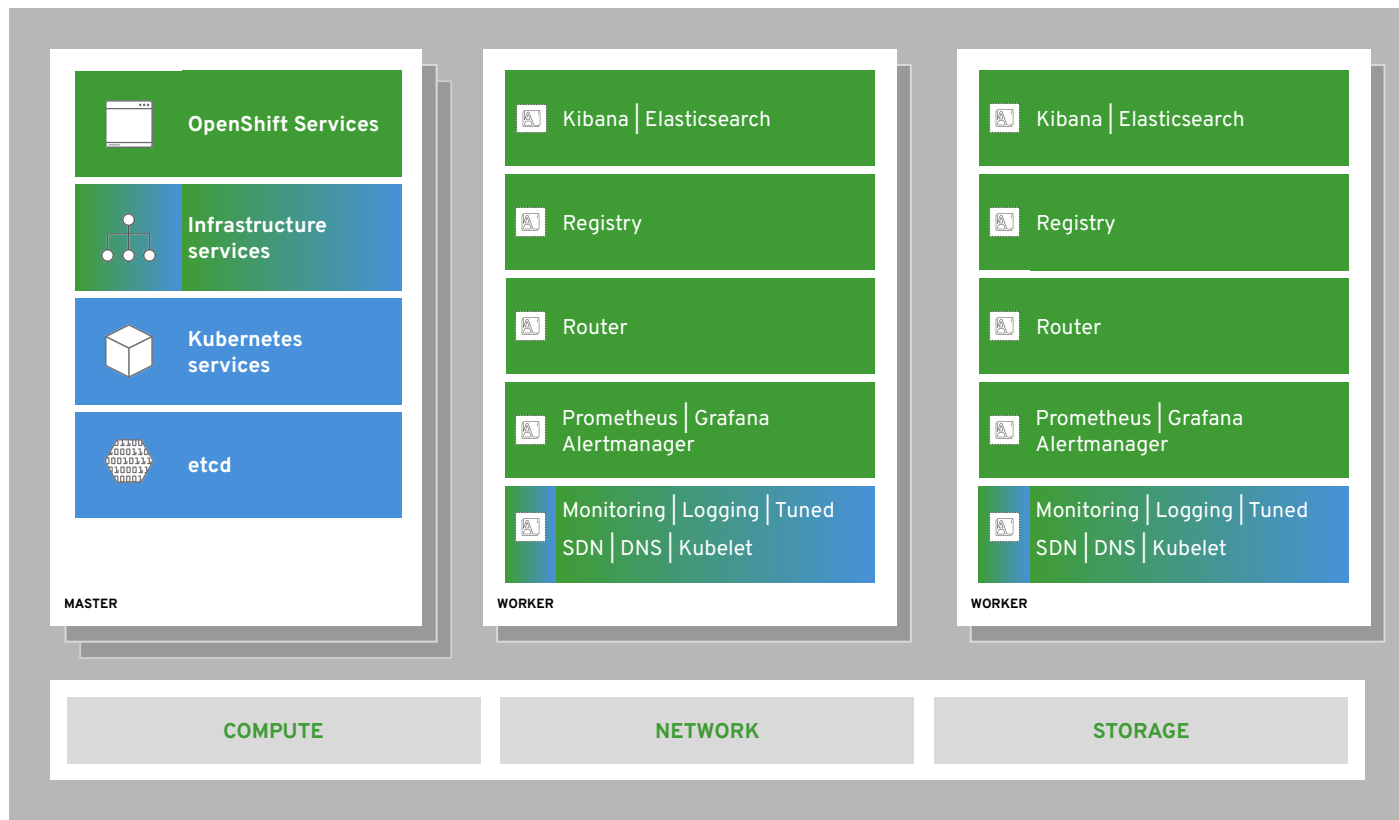
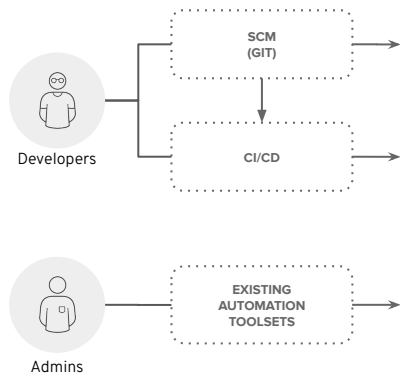




Functional overview







Database

Streaming & Messaging

Application Definition & Image Build

Continuous Integration & Delivery

Platform

Observability and Analysis

App Definition and Development



Orchestration & Management



Cloud-Native Storage

Container Runtime

Cloud-Native Network

Runtime



Automation & Configuration

Container Registry

Security & Compliance

Key Management

Provisioning



Public

Kubernetes Certified Service Provider

Kubernetes Training Partner

Cloud Native Computing Foundation


Cloud Native Landscape

Redpoint Amplify

l.cncf.io

This landscape is intended as a map through the previously uncharted terrain of cloud native technologies. There are many routes to deploying a cloud native application, with CNCF Projects representing a particularly well-traveled path.





OpenShift and Kubernetes core concepts

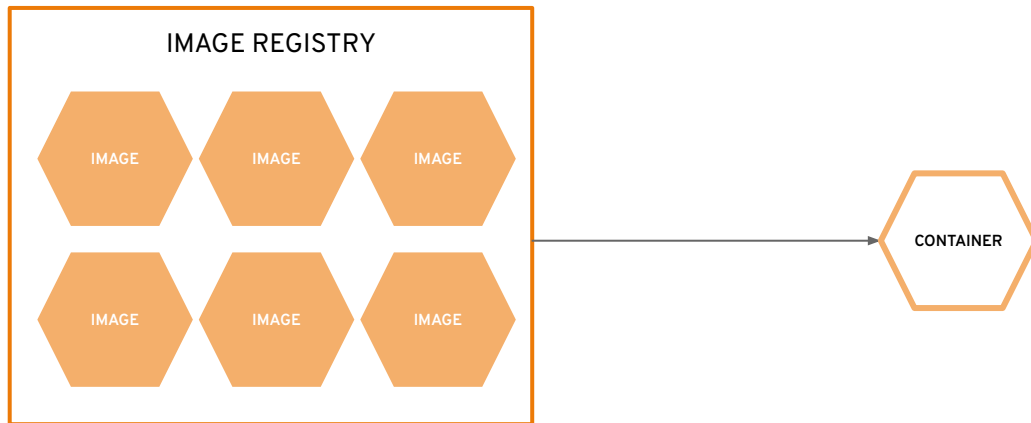
a container is the smallest compute unit



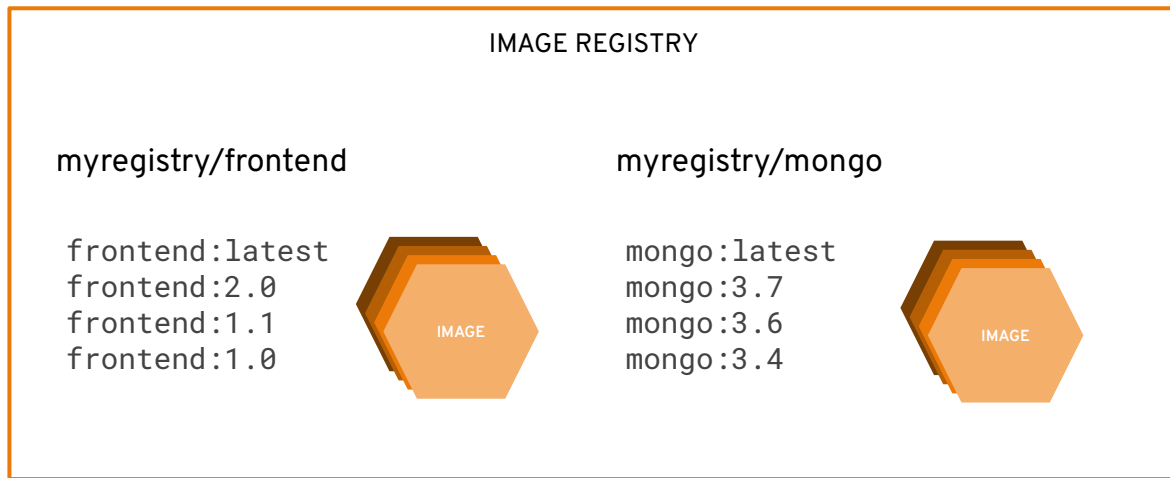
containers are created from container images



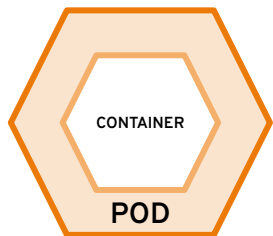
container images are stored in an image registry



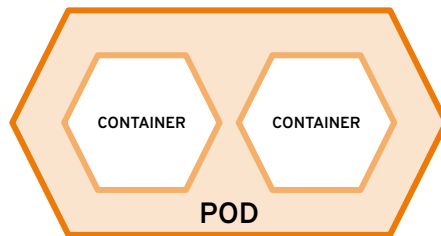
an image repository contains all versions of an image in the image registry



containers are wrapped in pods which are units of deployment and management

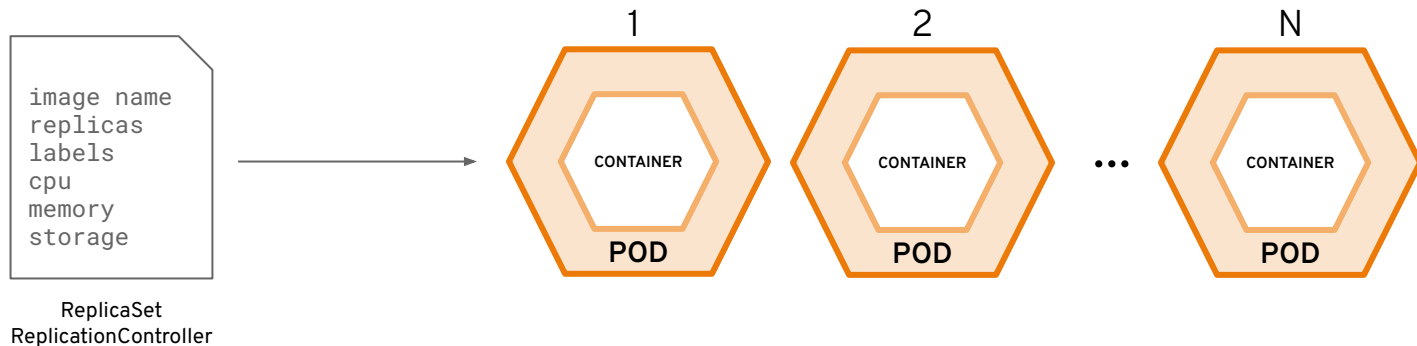


10.140.4.44

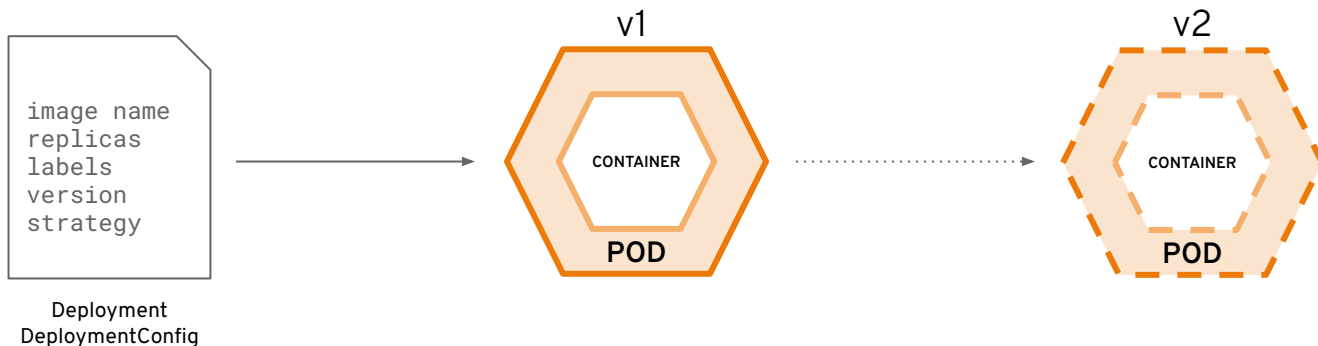


10.15.6.55

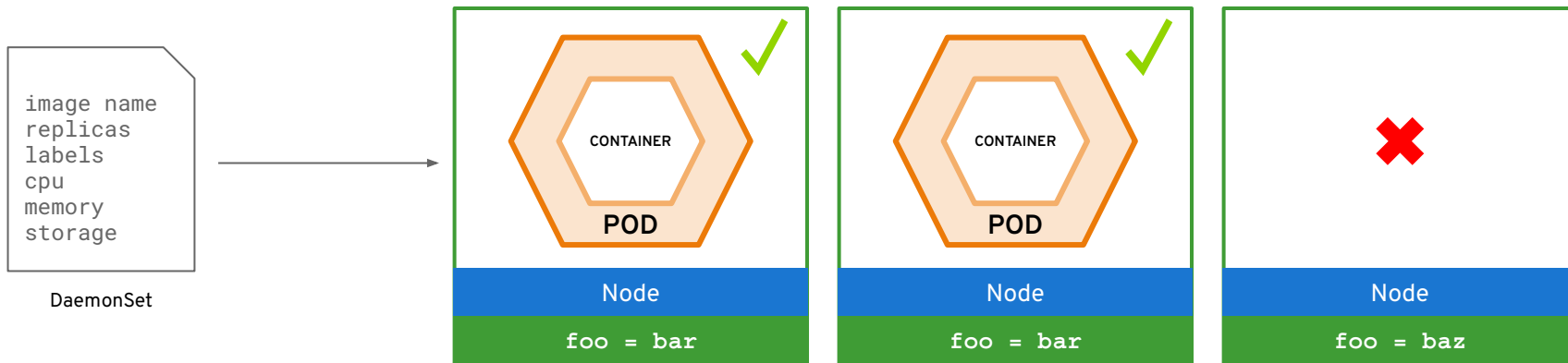
ReplicationControllers & ReplicaSets ensure a specified number of pods are running at any given time



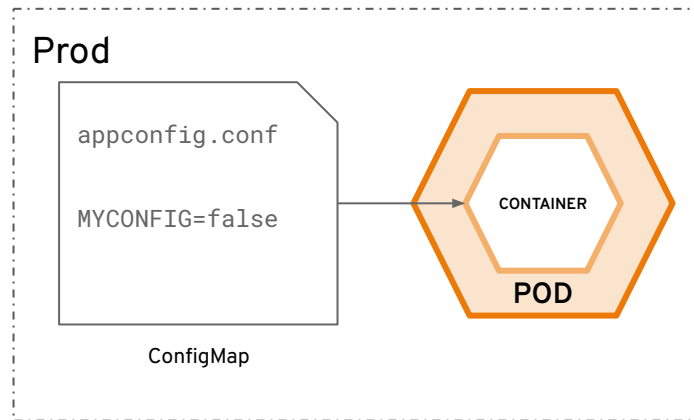
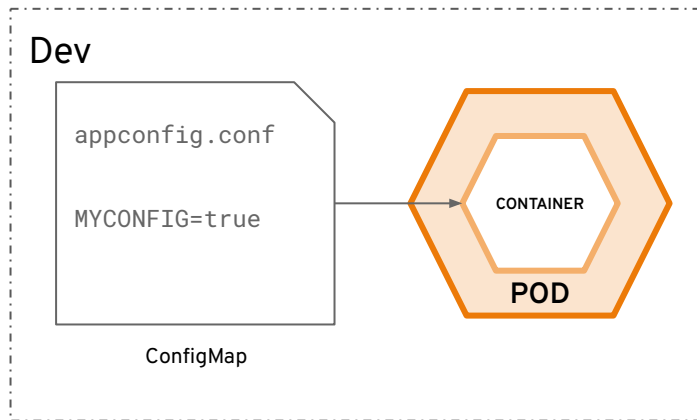
Deployments and DeploymentConfigurations define how to roll out new versions of Pods



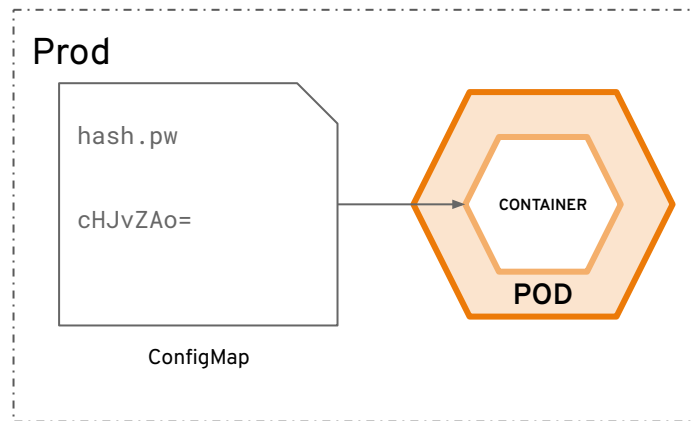
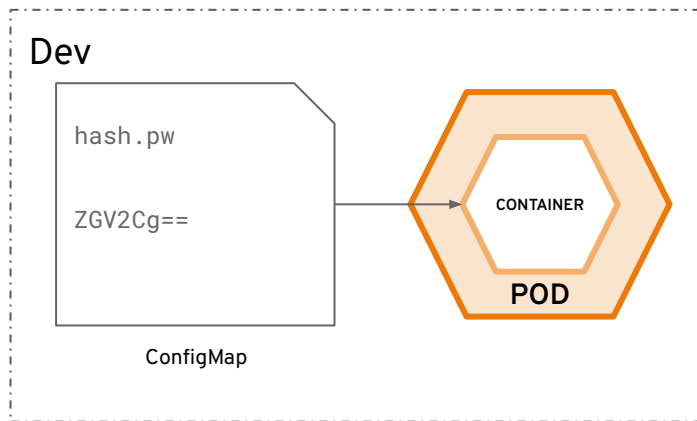
a daemonset ensures that all
(or some) nodes run a copy of a
pod



configmaps allow you to decouple configuration artifacts from image content



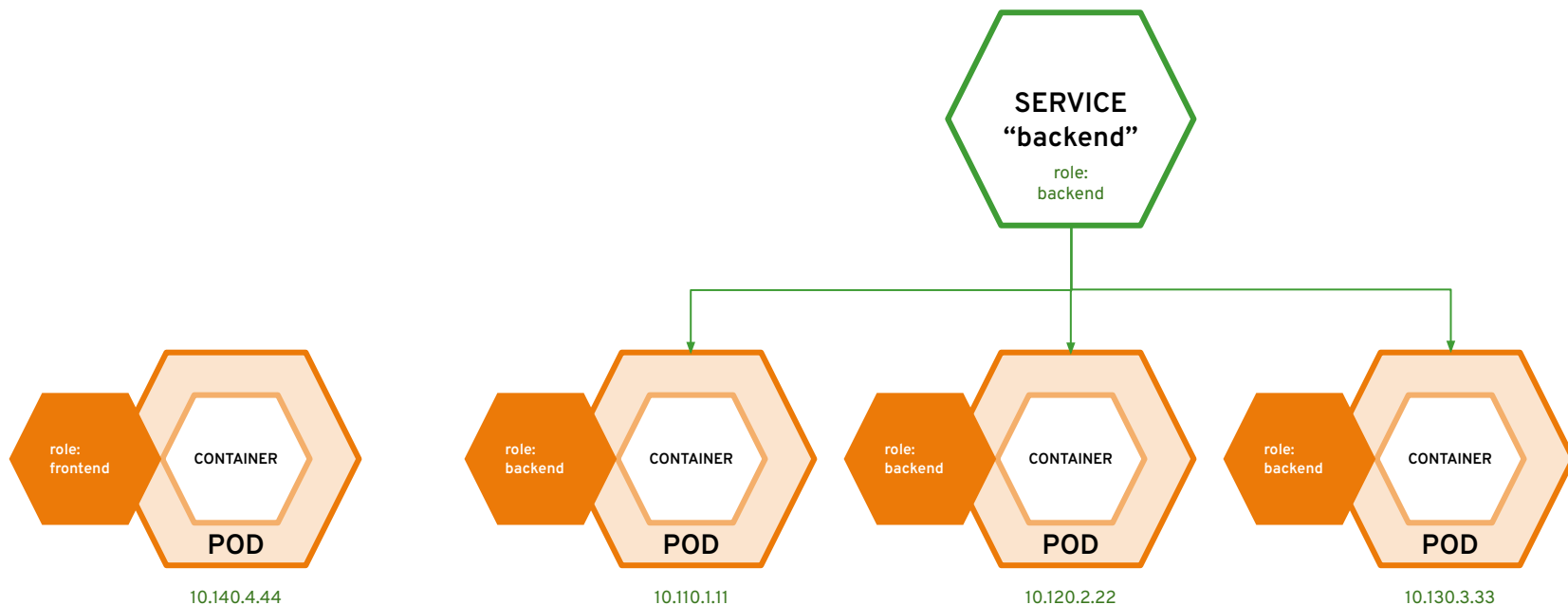
secrets provide a mechanism to hold sensitive information such as passwords



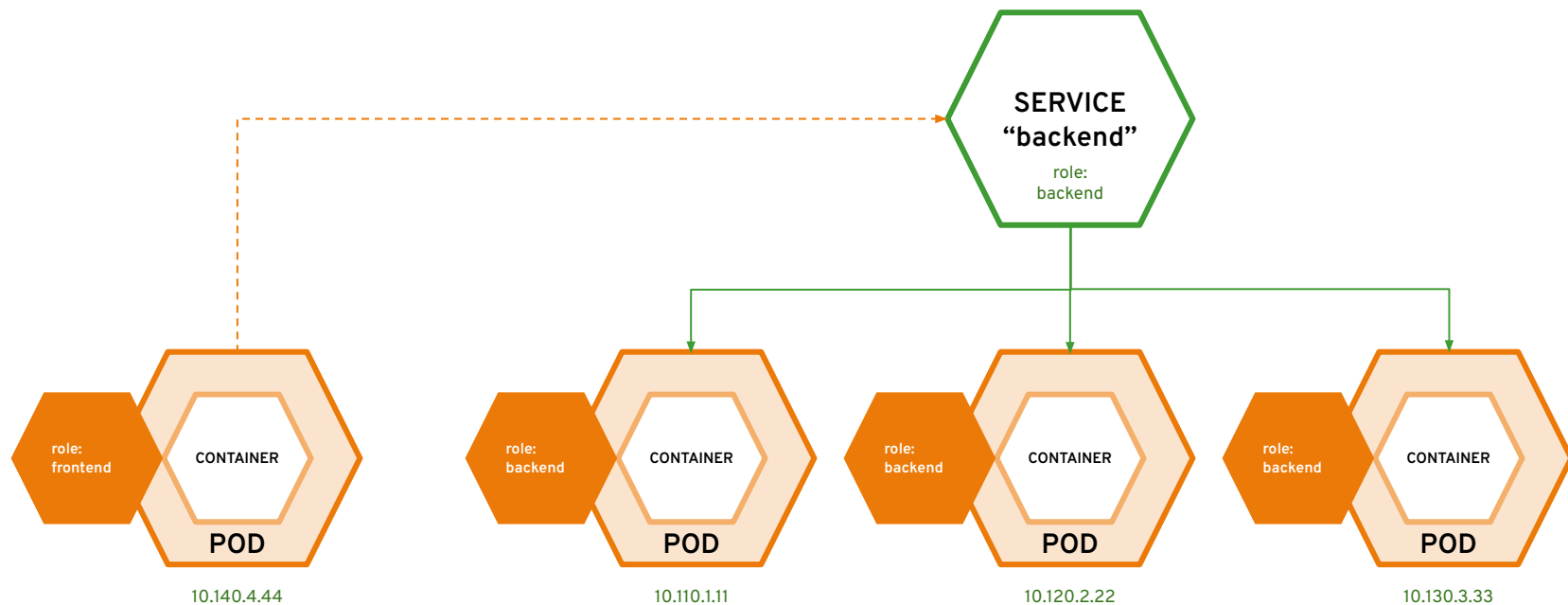
The etcd datastore can be encrypted for additional security

<https://docs.openshift.com/container-platform/4.6/security/encrypting-etcd.html>

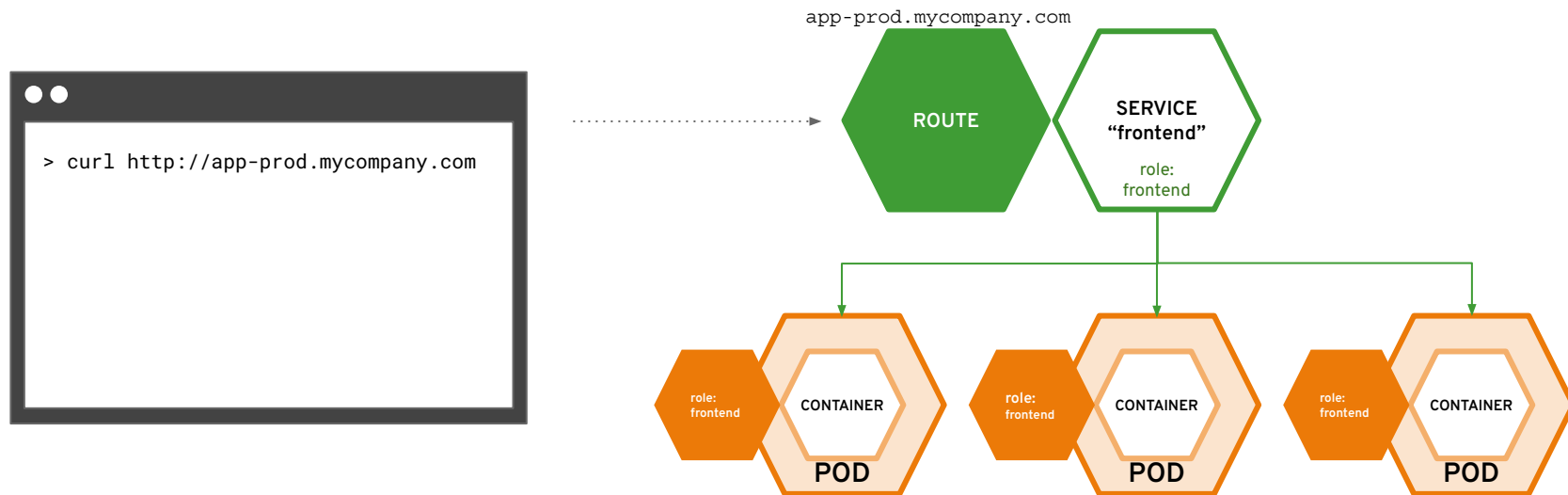
services provide internal load-balancing and service discovery across pods



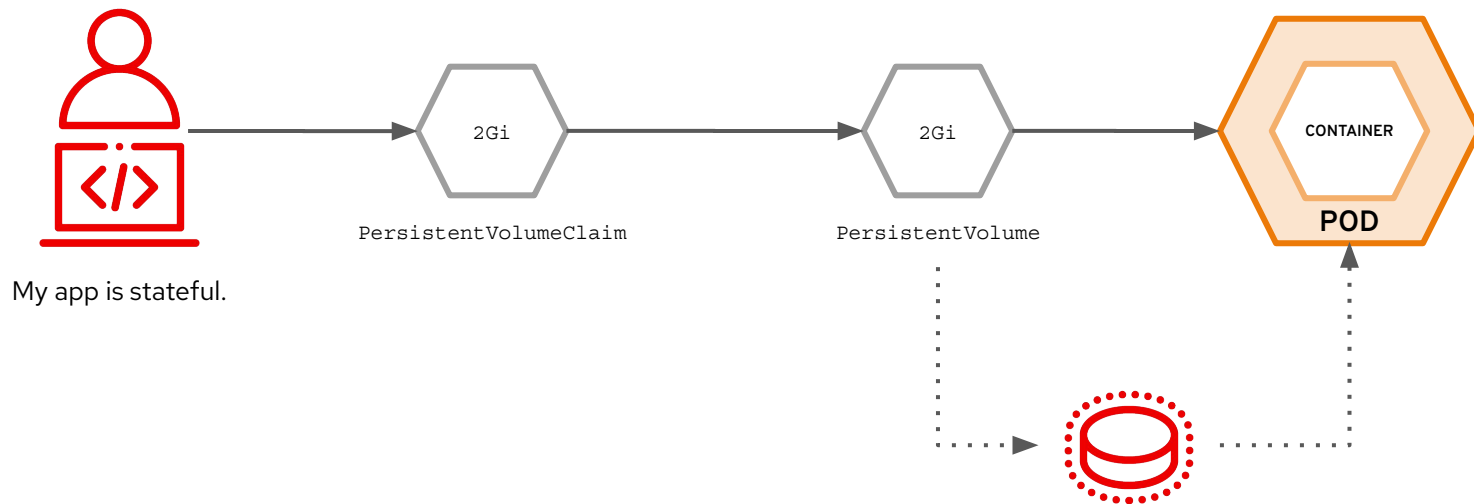
apps can talk to each other via services



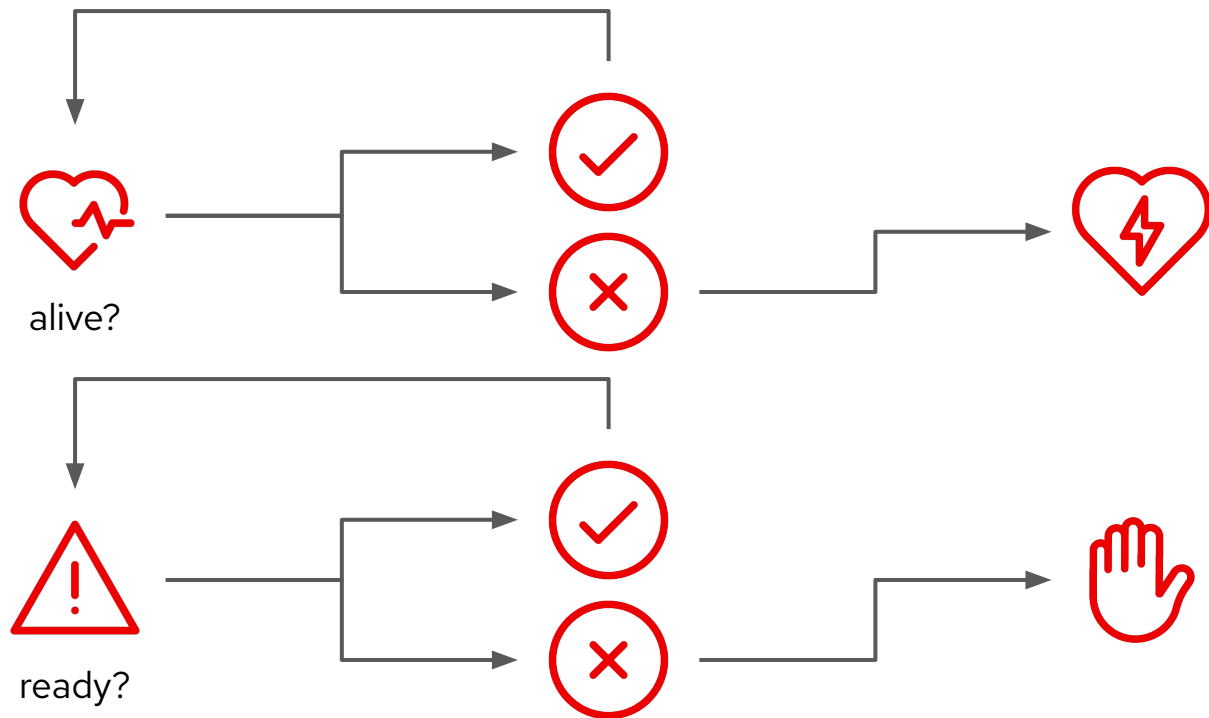
routes make services accessible to clients outside the environment via real-world urls



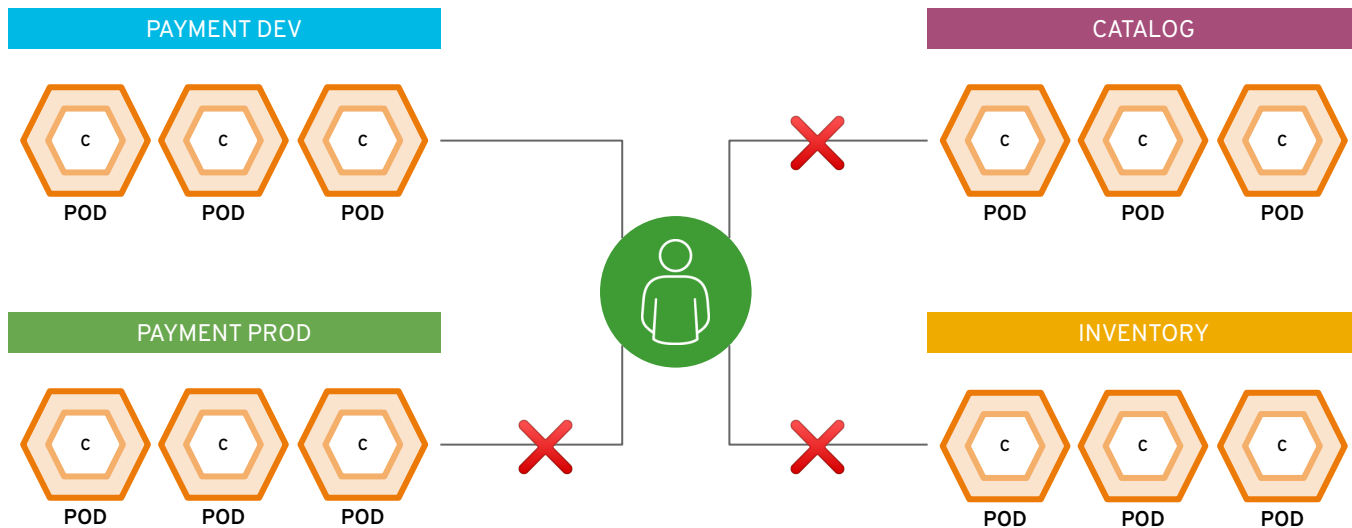
Persistent Volume and Claims



Liveness and Readiness



projects isolate apps across environments, teams, groups and departments





OpenShift 4 Architecture

your choice of infrastructure

COMPUTE

NETWORK

STORAGE

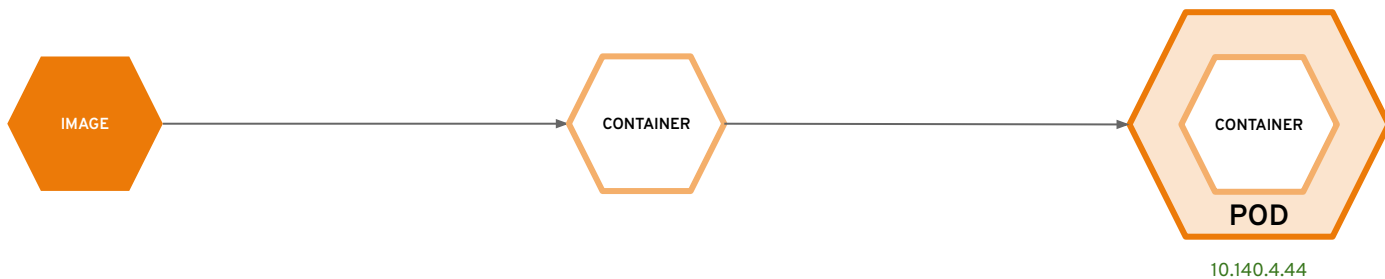
workers run workloads



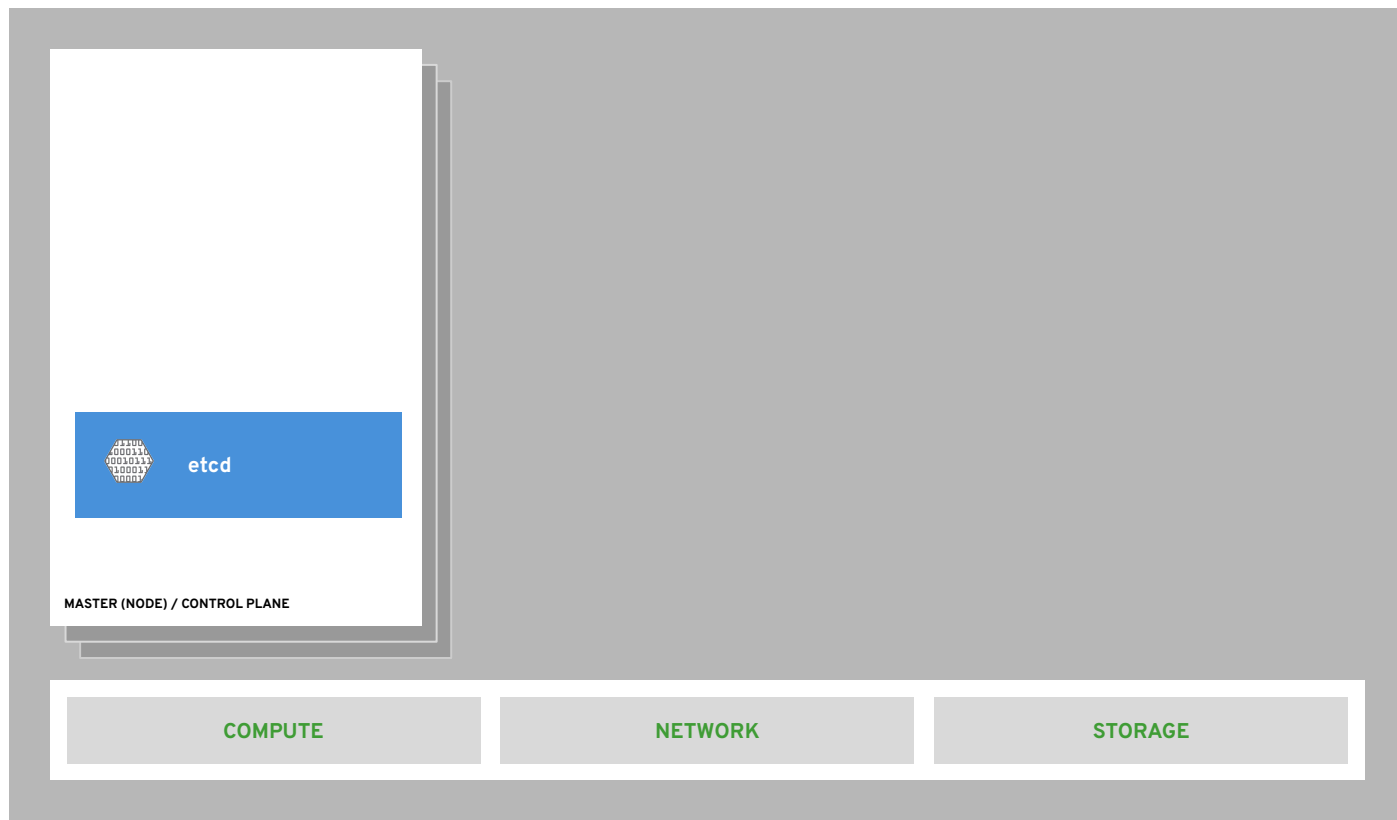
masters are the control plane



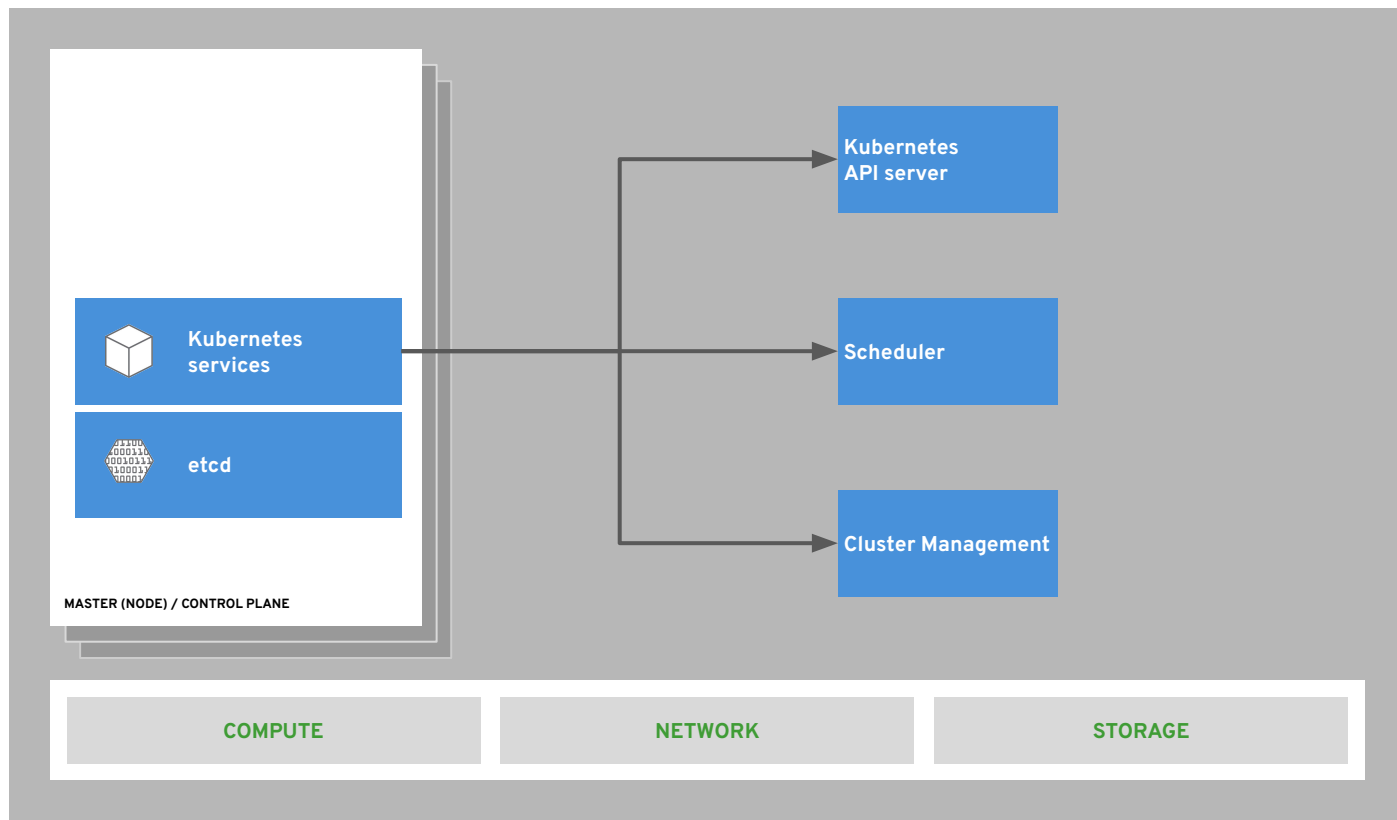
everything runs in pods



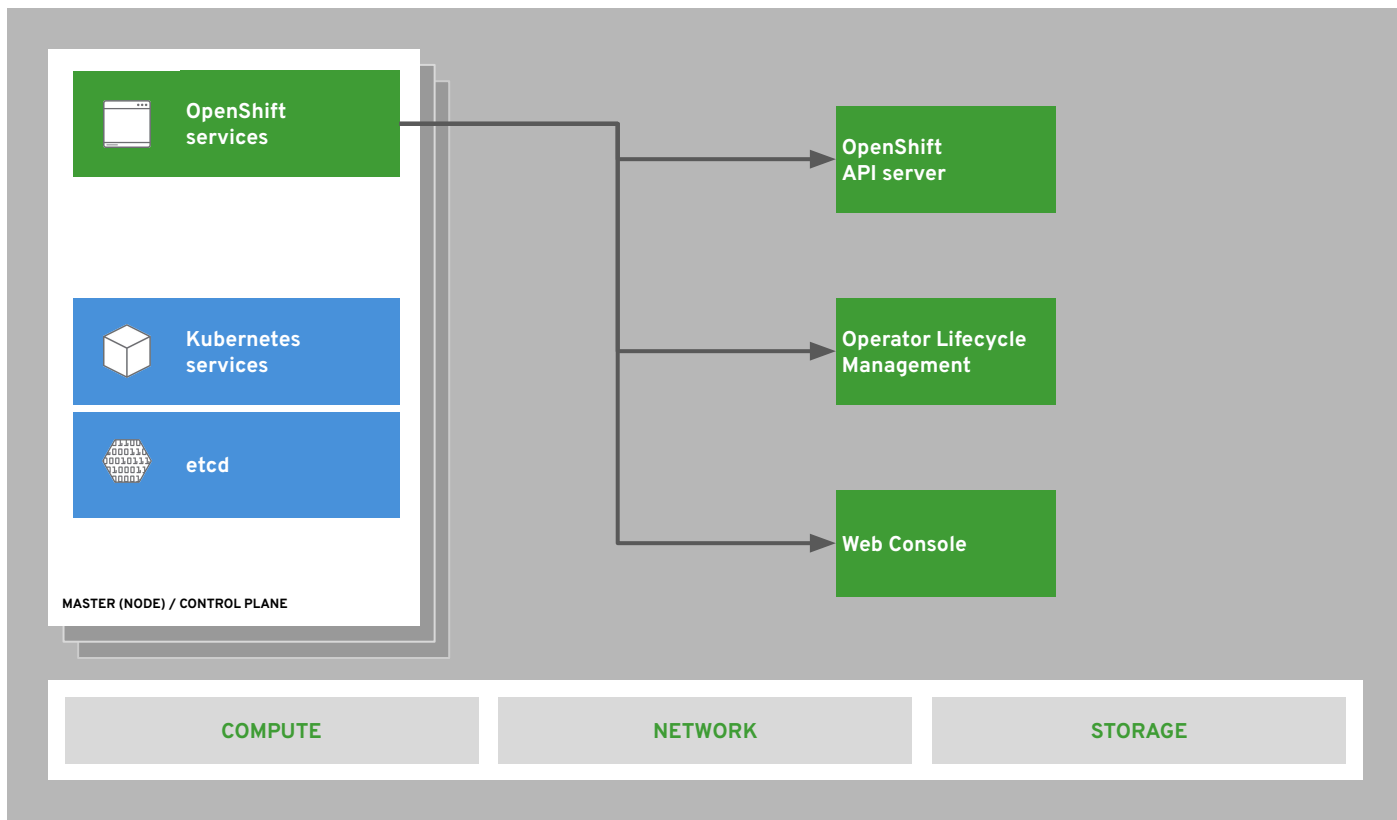
state of everything



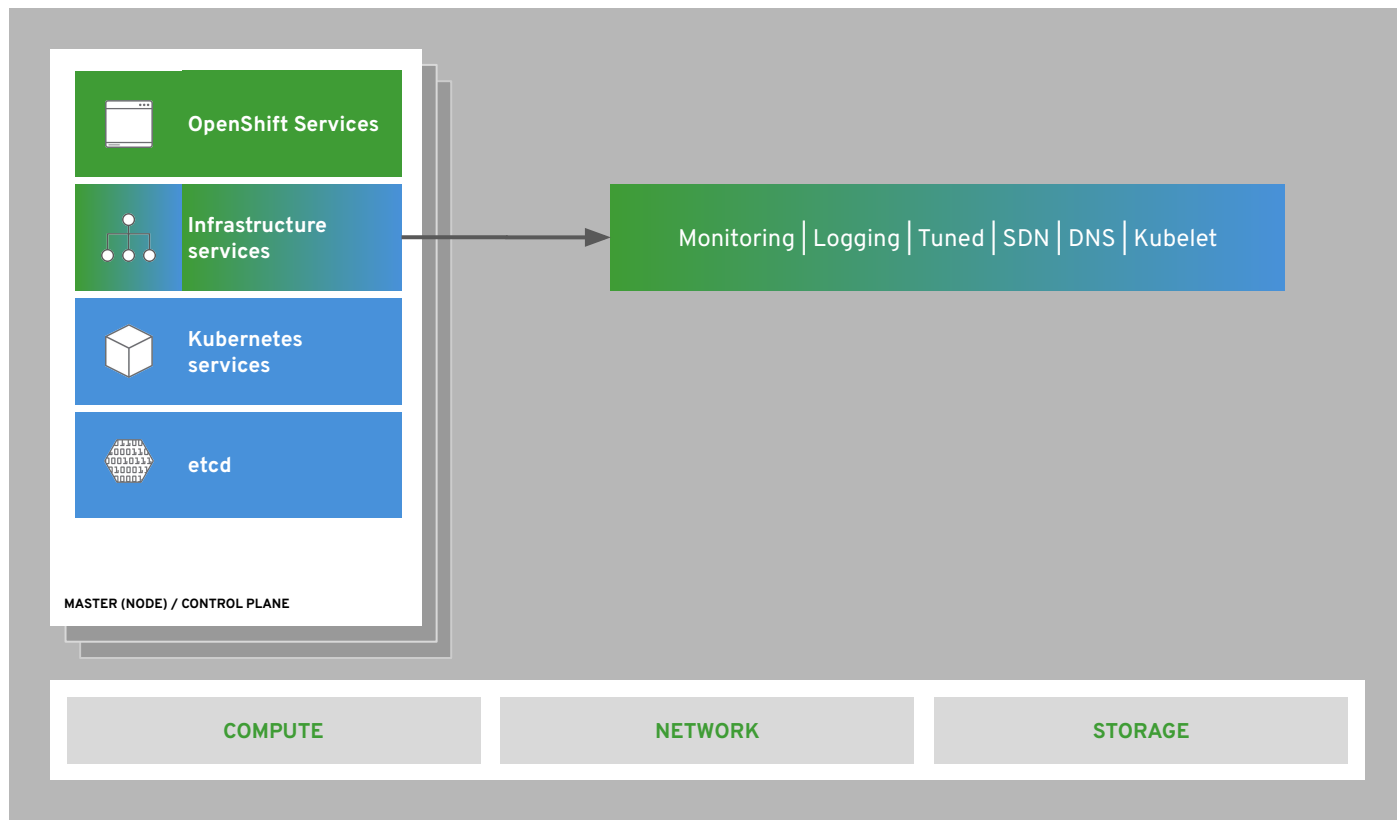
core kubernetes components



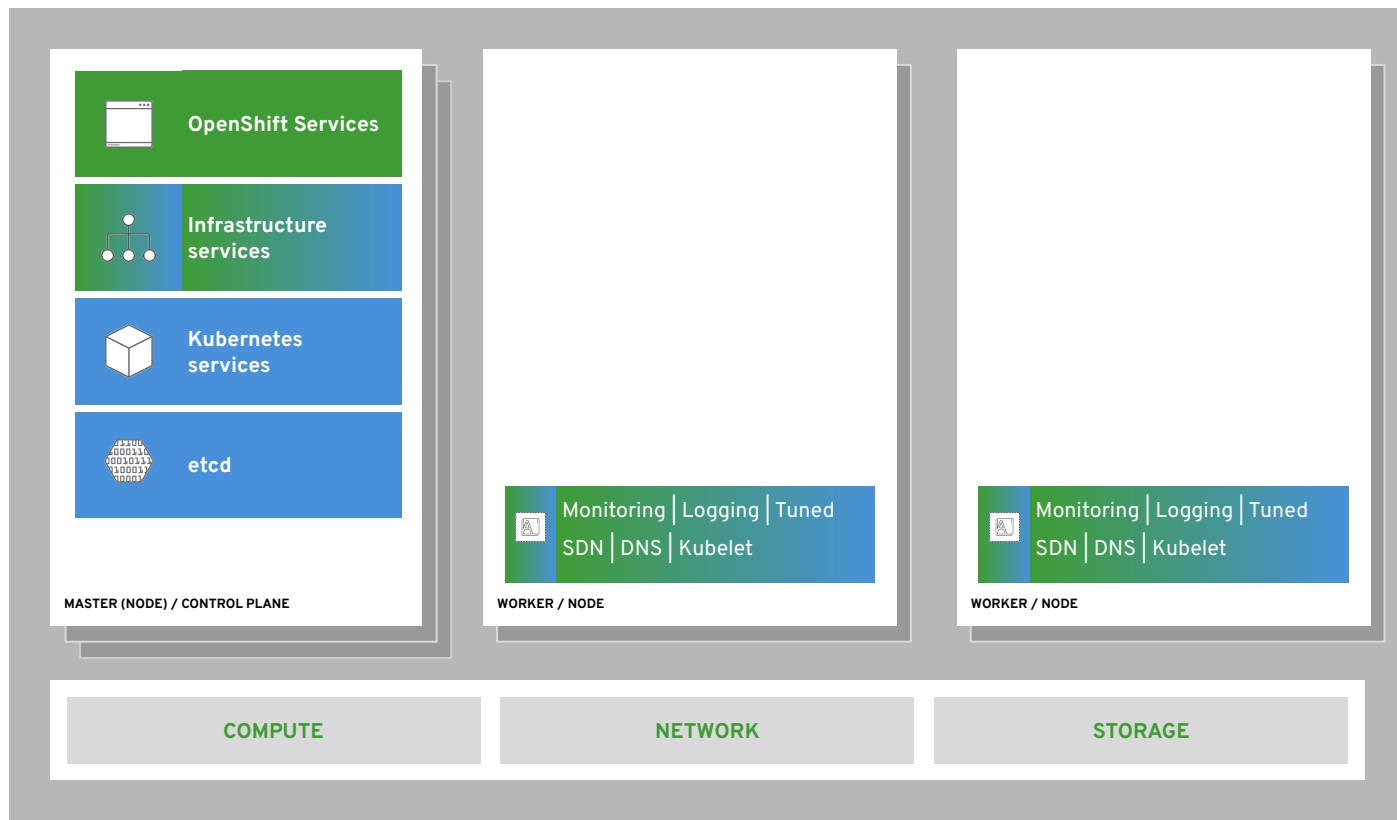
core OpenShift components



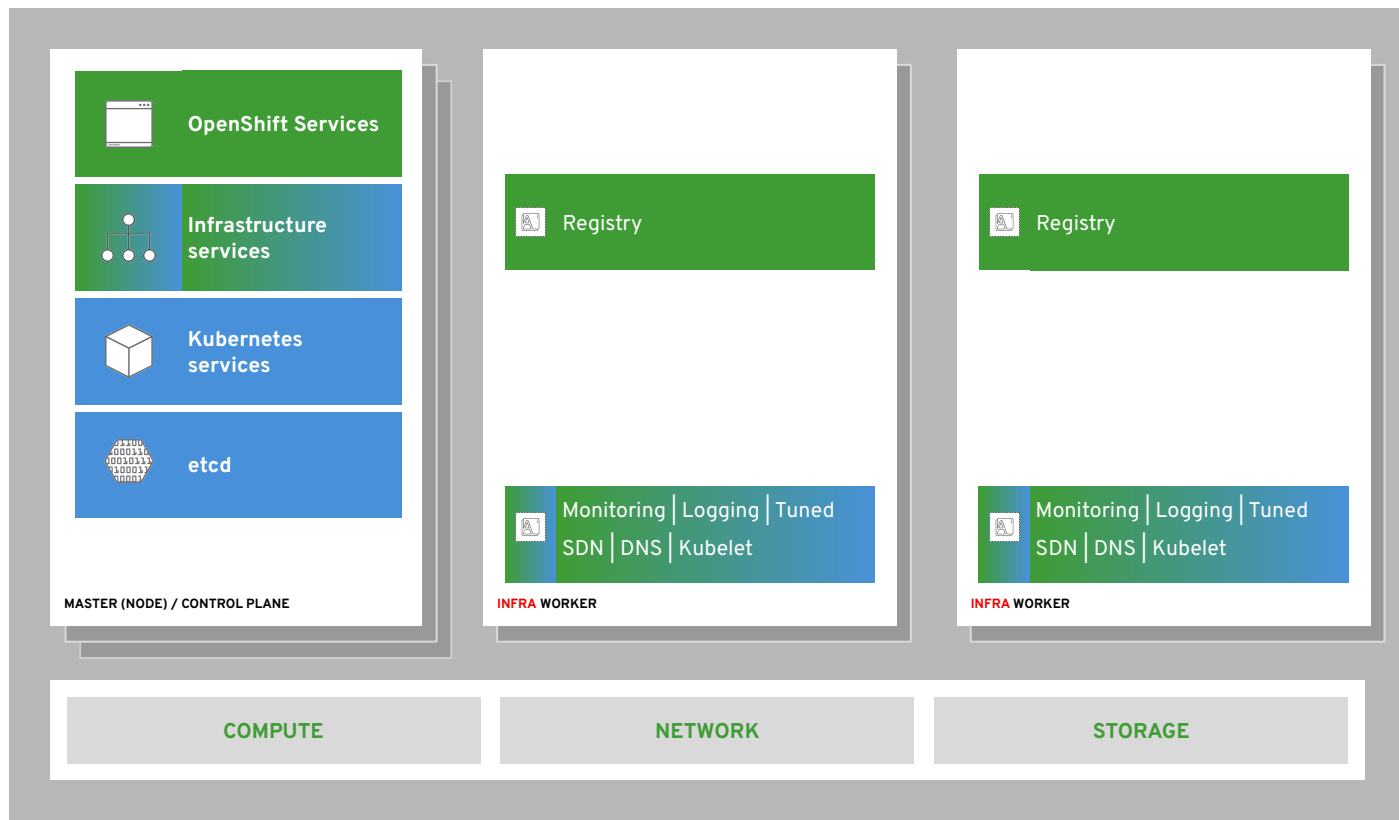
internal and support infrastructure services



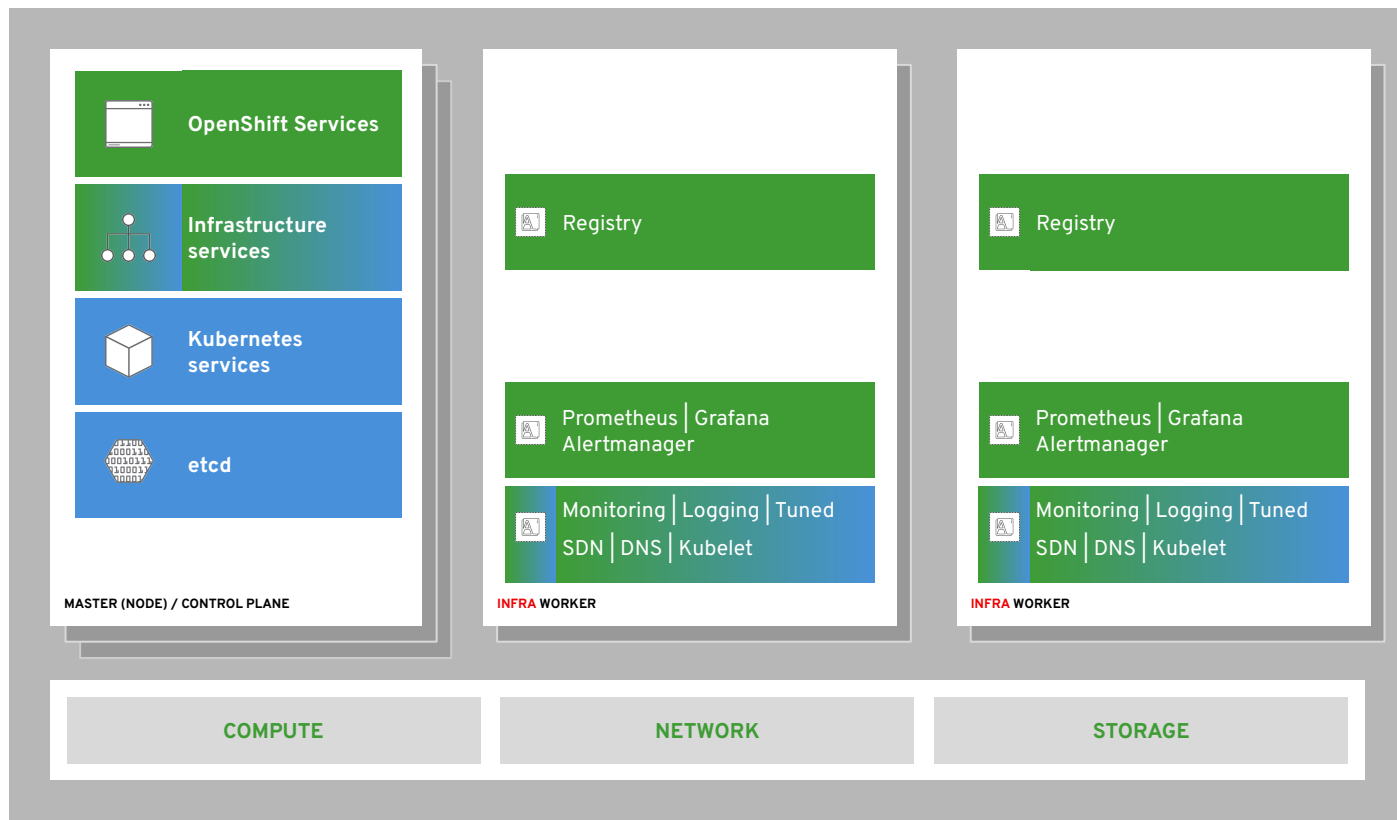
run on all hosts



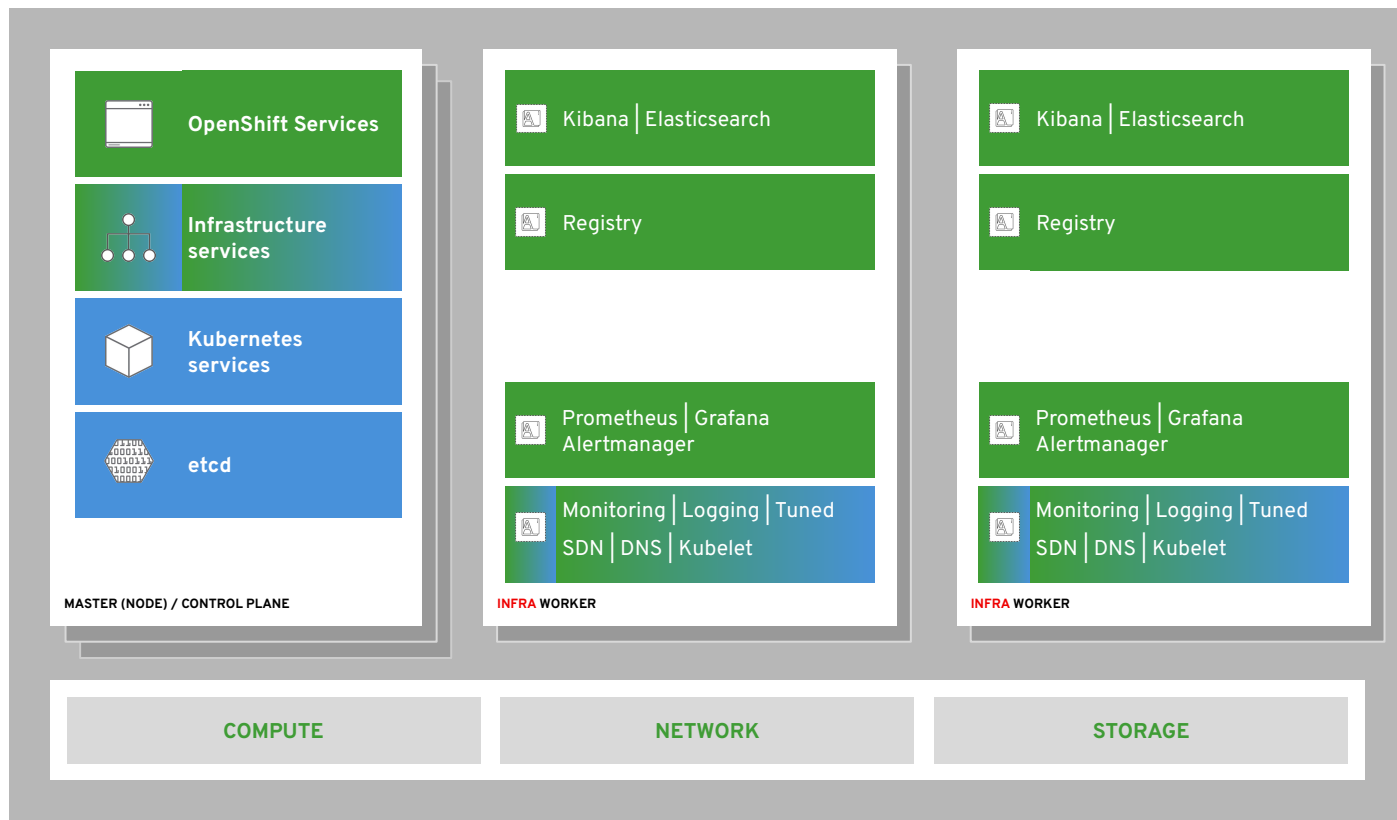
integrated image registry



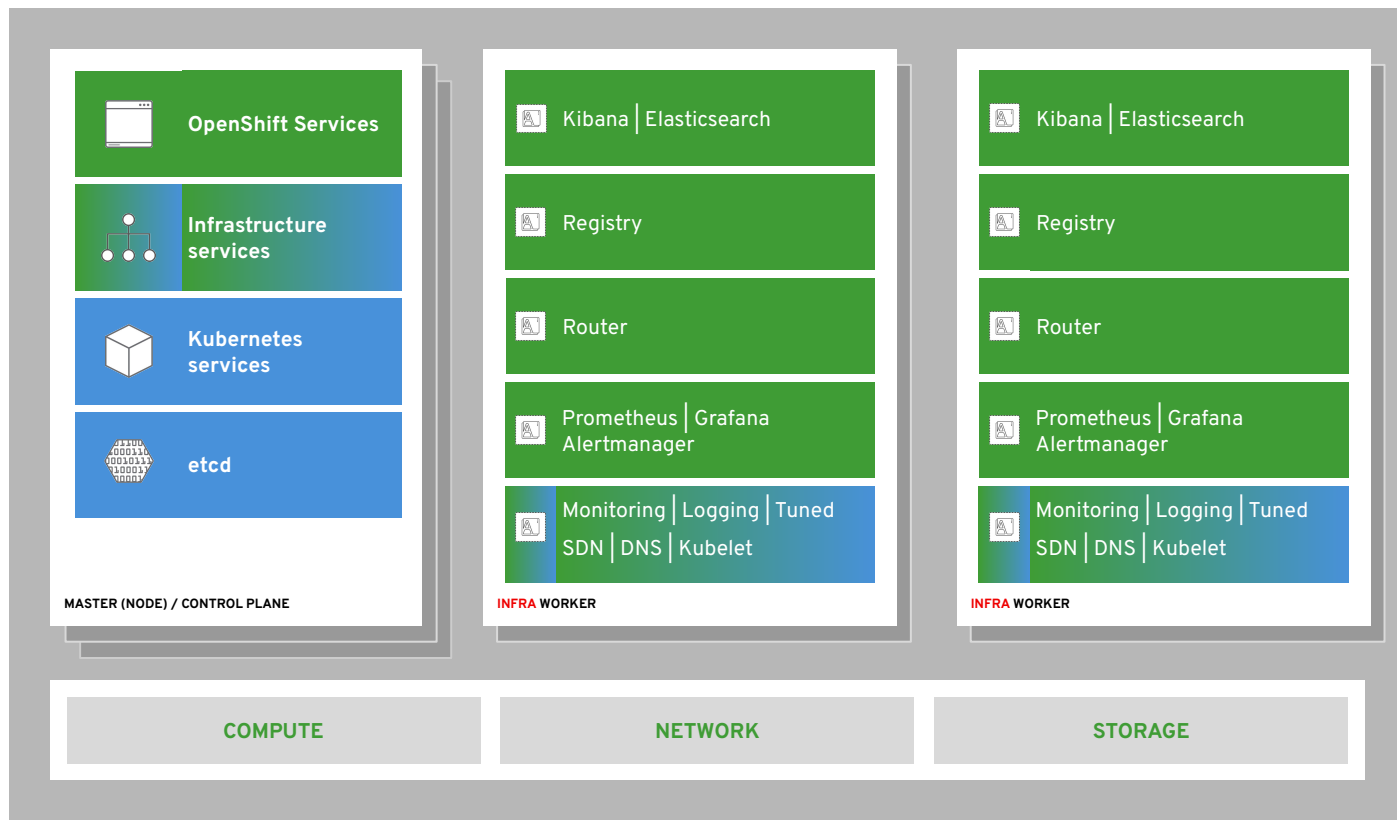
cluster monitoring



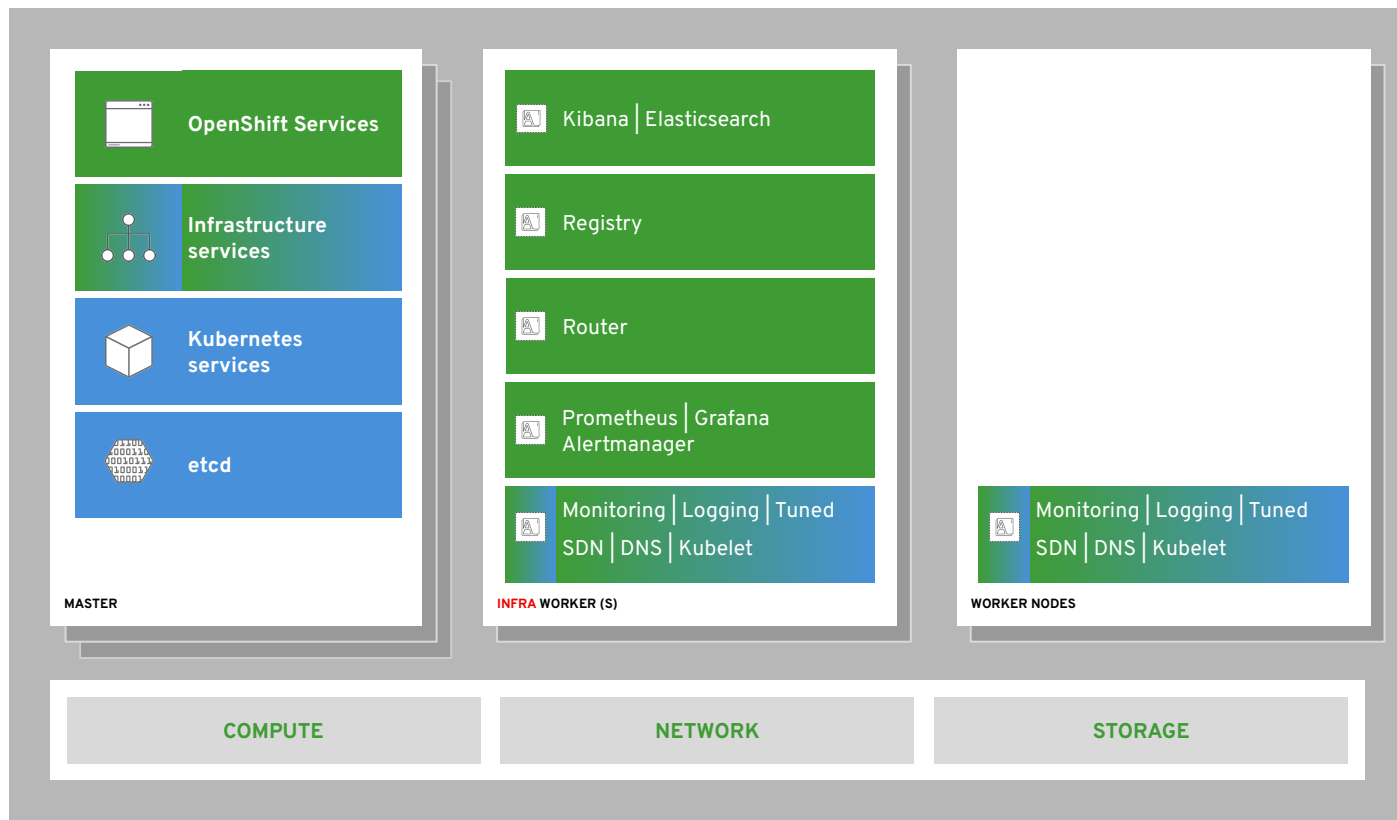
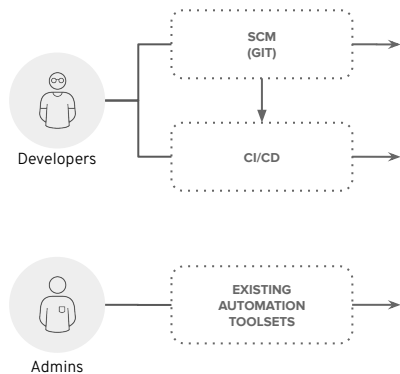
log aggregation



integrated routing



dev and ops via web, cli, API, and IDE



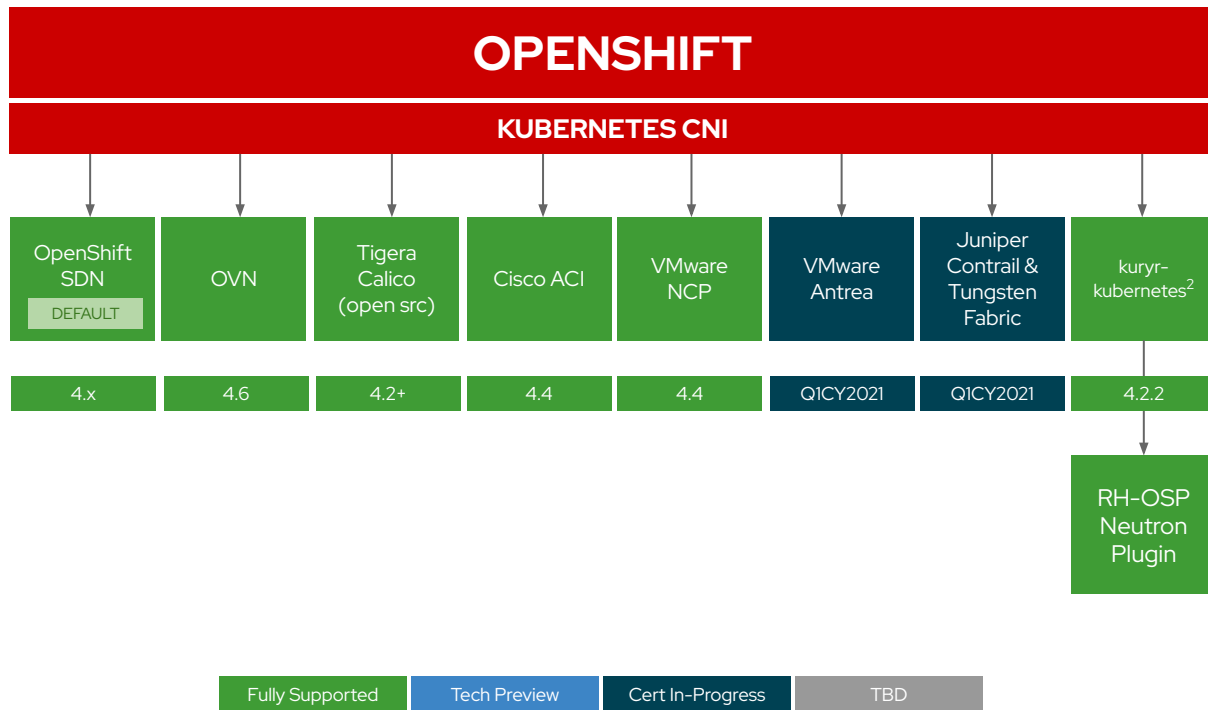
Networking

A pluggable model for
network interface
controls in kubernetes

OpenShift Networking Plug-ins

3rd-party Kubernetes CNI plug-in certification primarily consists of:

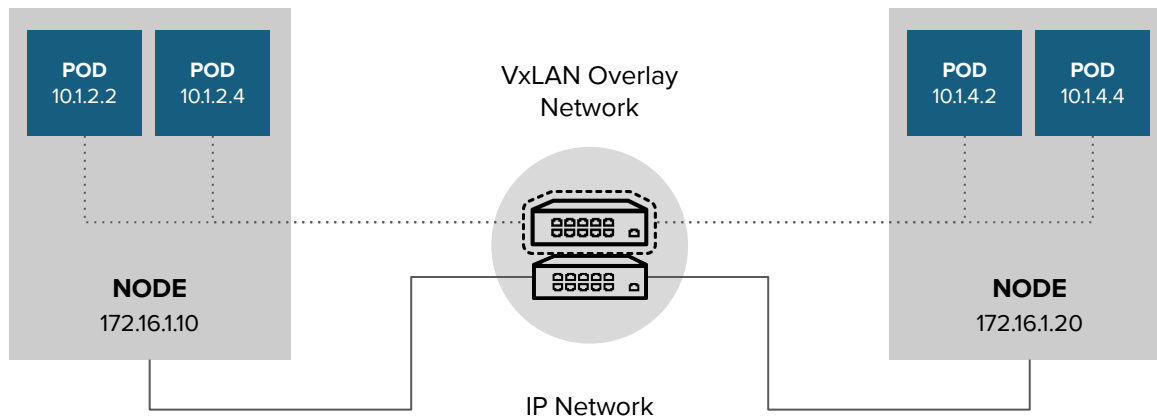
1. Formalizing the partnership
2. Certifying the container(s)
3. Certifying the Operator
4. Successfully passing the same Kubernetes networking conformance tests that OpenShift uses to validate its own SDN



OpenShift SDN

An Open
vSwitch-based
Software Defined
Network for
kubernetes

OpenShift SDN high-level architecture



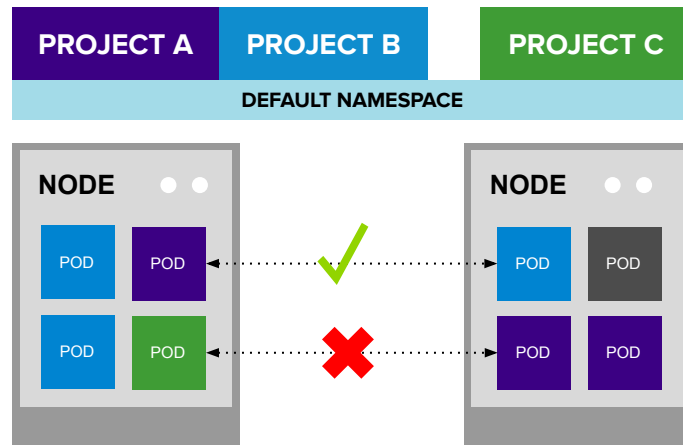
OpenShift SDN “flavors”

OPEN NETWORK (Default)

- All pods can communicate with each other across projects

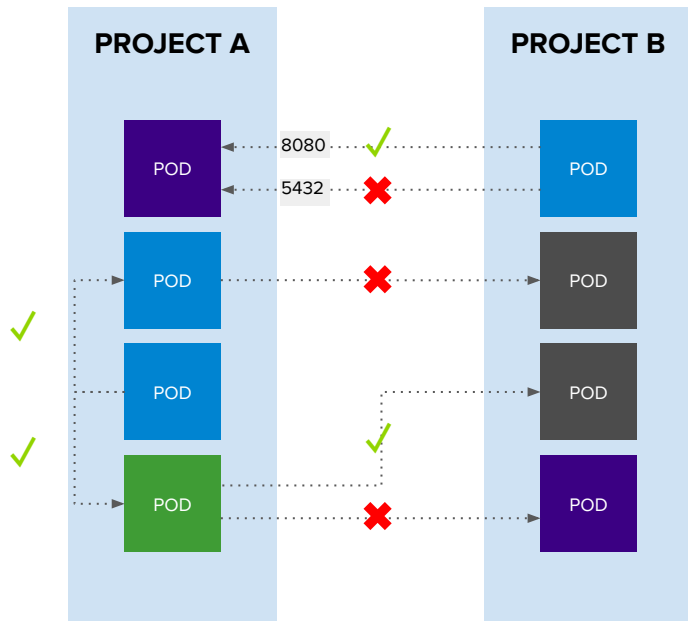
MULTI-TENANT NETWORK

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



Multi-Tenant Network

NetworkPolicy

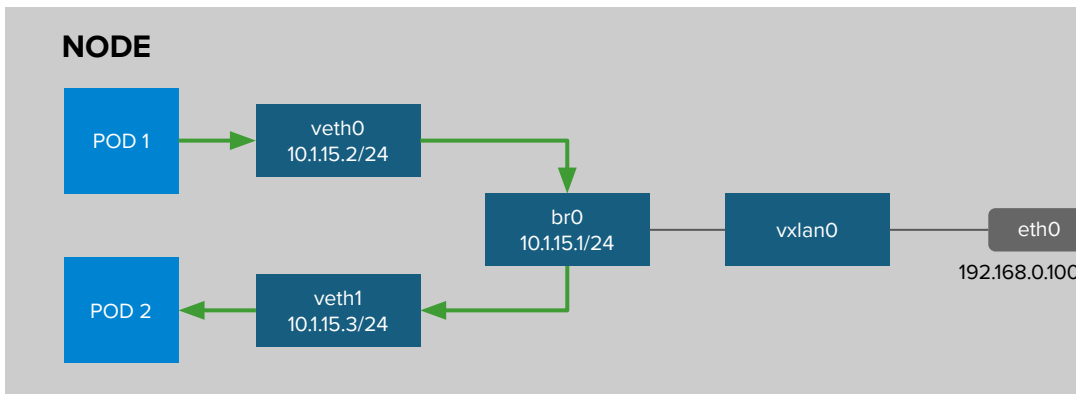


Example Policies

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

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

OpenShift SDN packet flows container-container on same host

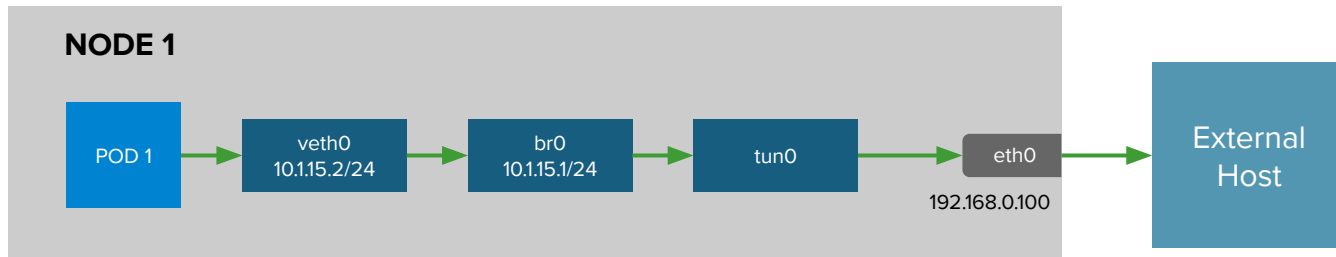


OpenShift SDN packet flows

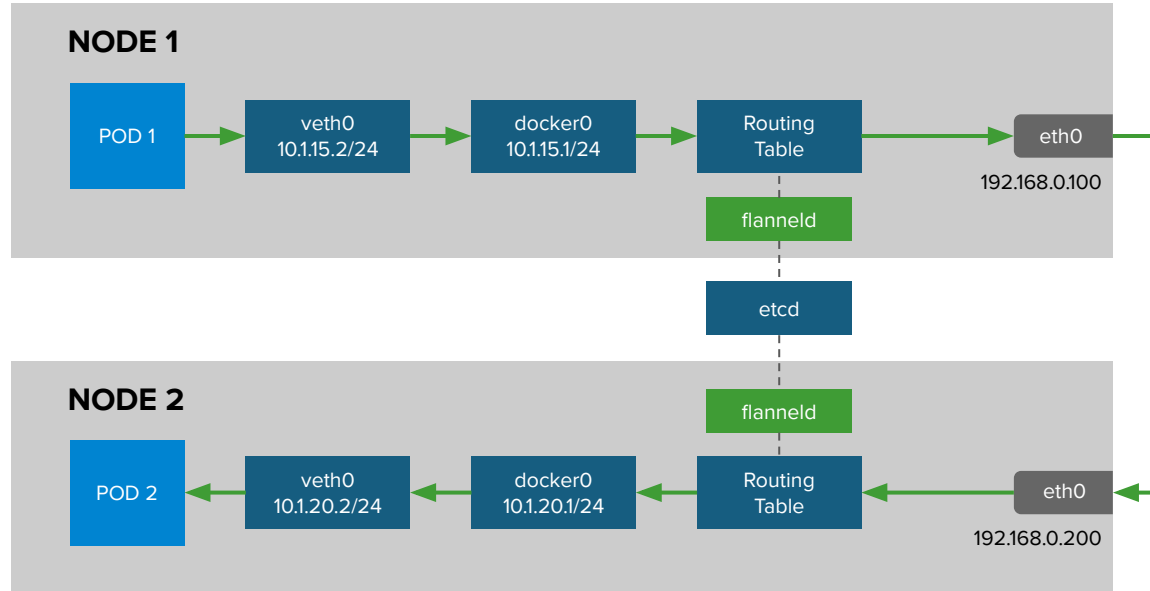
container-container across hosts



OpenShift SDN packet flows container leaving the host



Kuryr and OpenStack



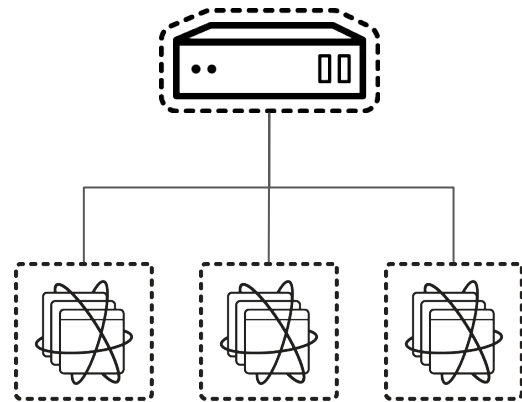
Flannel is minimally verified and is supported only and exactly as deployed in the OpenShift on OpenStack reference architecture <https://access.redhat.com/articles/2743631>

routes and ingress

How traffic enters the
cluster

Routing and Load Balancing

- Pluggable routing architecture
 - HAProxy Router
 - F5 Router
- Multiple-routers with traffic sharding
- Router supported protocols
 - HTTP/HTTPS
 - WebSockets
 - TLS with SNI
- Non-standard ports via cloud load-balancers, external IP, and NodePort

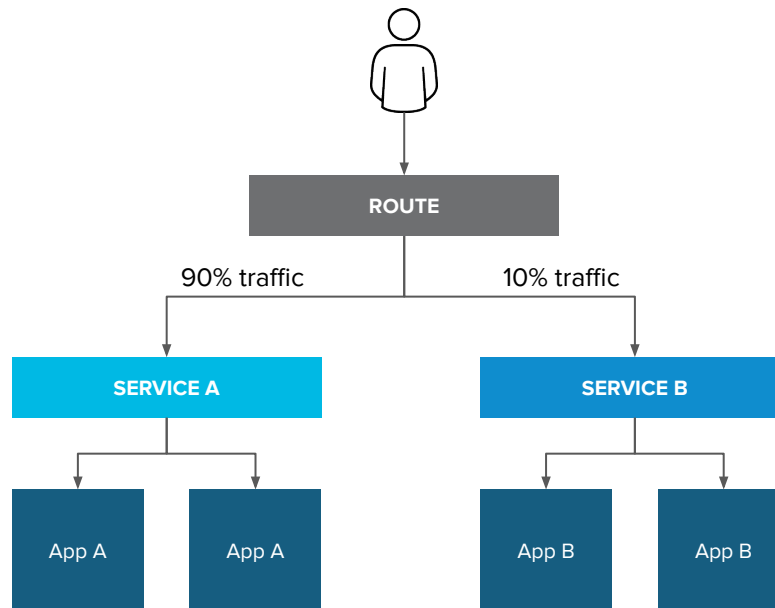


Routes vs Ingress

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

Router-based deployment methodologies

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

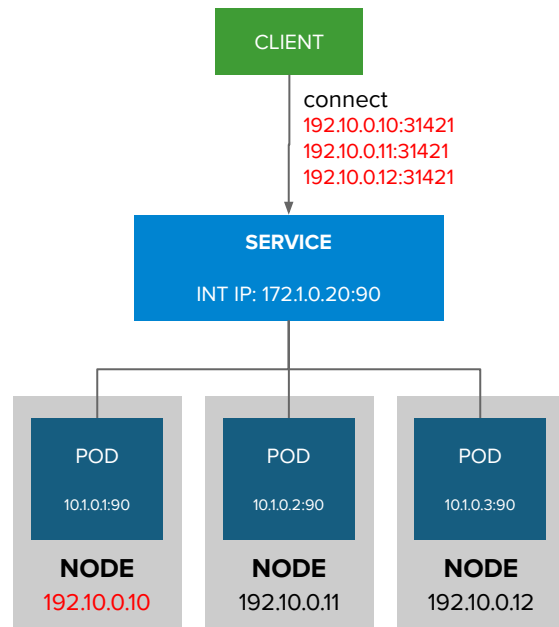


Alternative methods for ingress

Different ways that
traffic can enter the
cluster without the
router

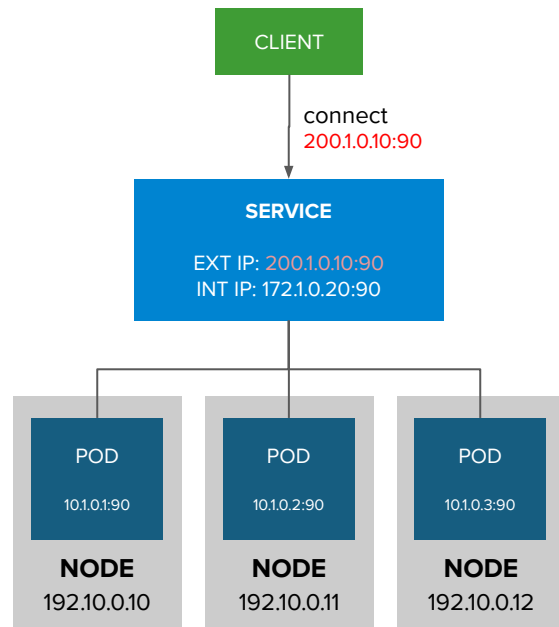
Entering the cluster on a random port with service nodeports

- 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



External traffic to a service on any port with external IP

- 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 (fully supported in 4.8)



Cluster DNS

An automated system
for providing hostname
resolution within
kubernetes

CoreDNS

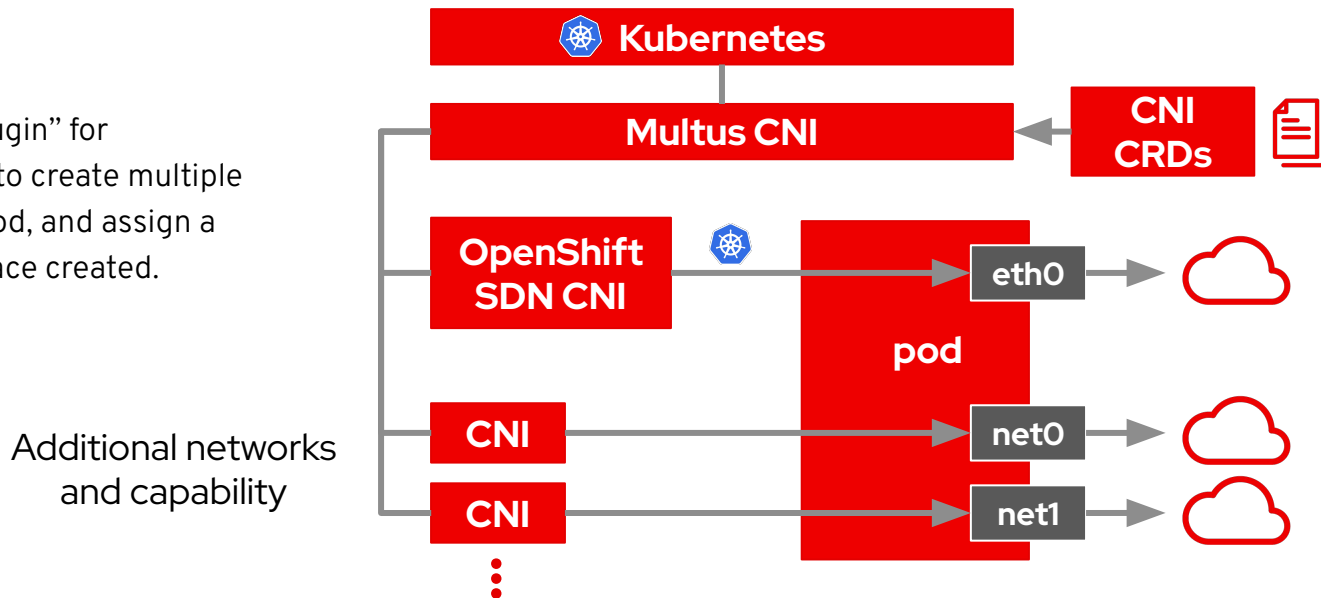
- Built-in internal DNS to reach services by a (fully qualified) hostname
- Split DNS is used with CoreDNS
 - CoreDNS answers DNS queries for internal/cluster services
 - Other defined “upstream” name servers serve the rest of the queries

Multus

A CNI plugin that
provides multiple
network interfaces for
pods

Multinetwork with Multus

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.

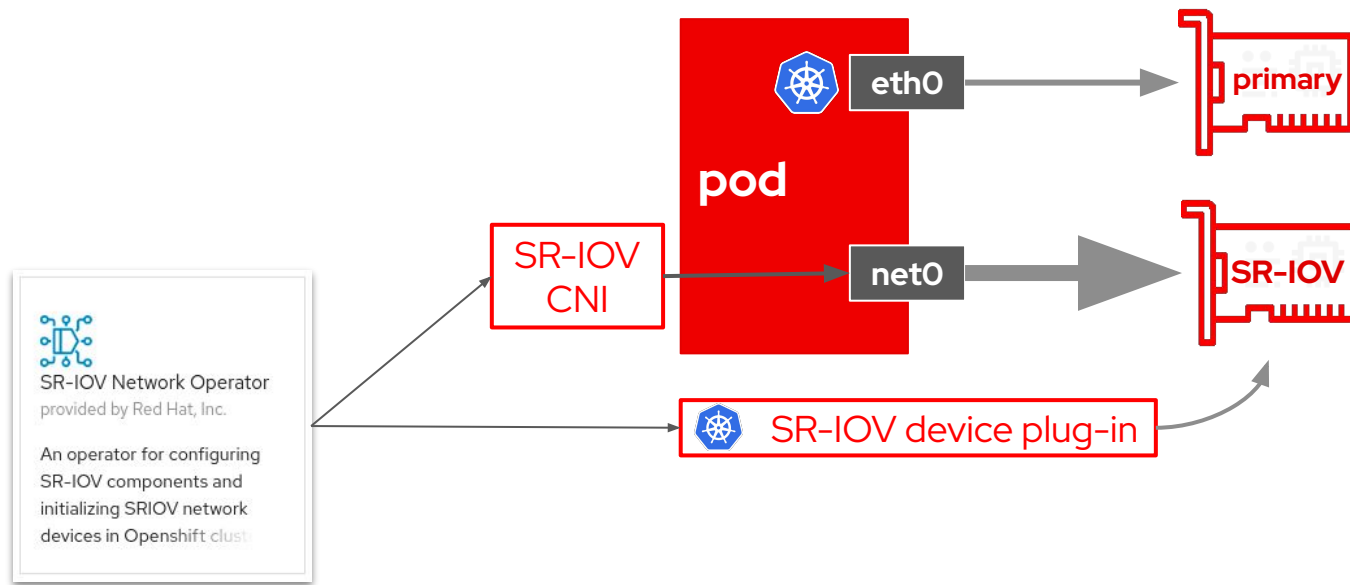


Additional OpenShift-Supported Secondary CNI Plug-Ins

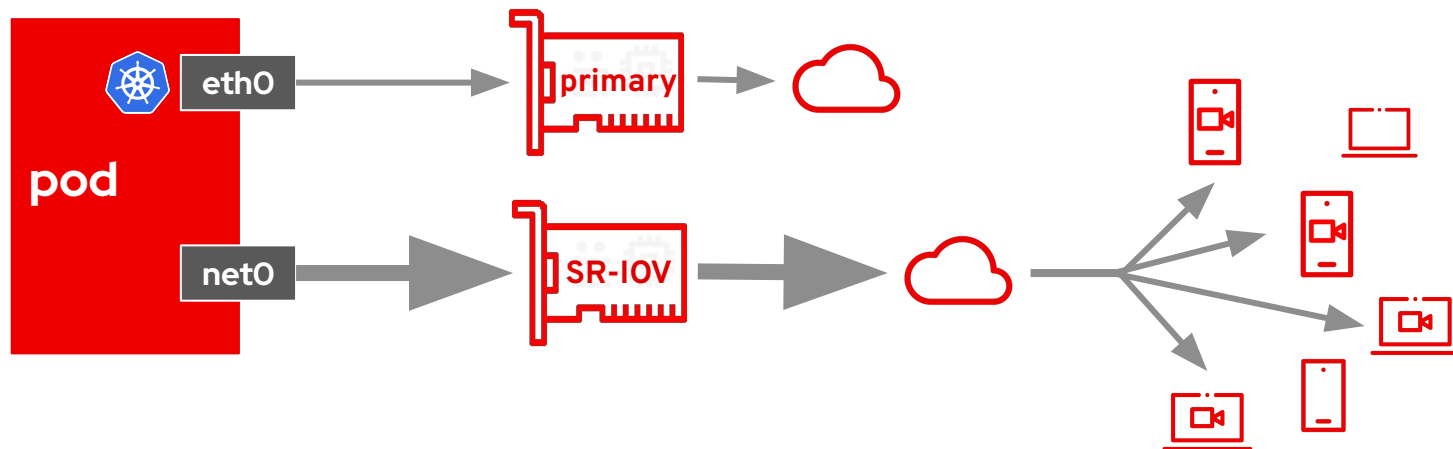
OpenShift 4.x Tested Integrations: [Network Components and Plugins](#)

- host device
- IPAM(dhcp)
- MACVLAN
- IPVLAN
- Bridge with VLAN
- Static IPAM
- DHCP IPAM
- Route Override
- whereabouts
- SR-IOV
- ...

SR-IOV



High-performance multicast



OpenShift Monitoring

An integrated cluster
monitoring and alerting
stack

OpenShift Cluster Monitoring



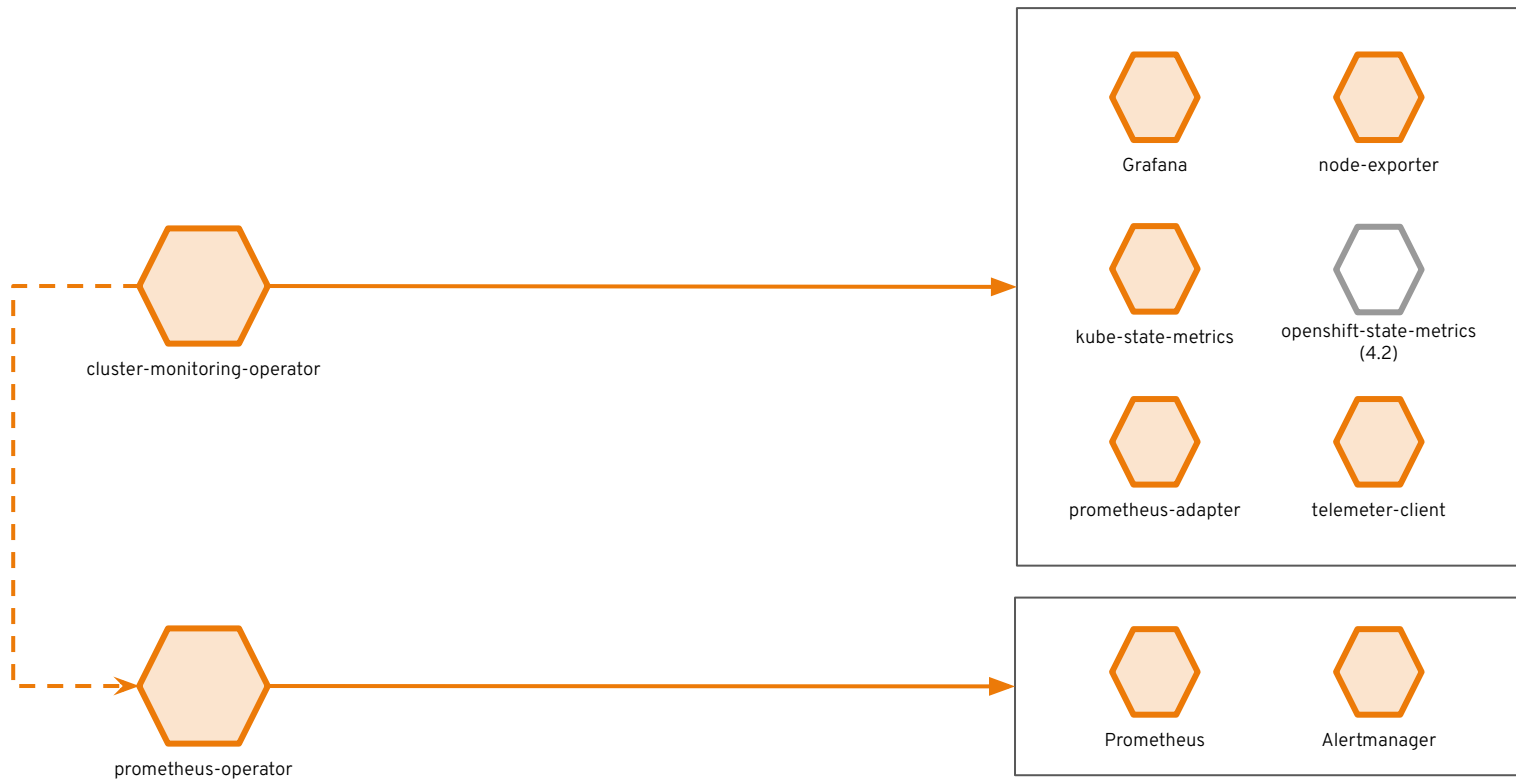
Metrics collection and storage
via Prometheus, an
open-source monitoring system
time series database.

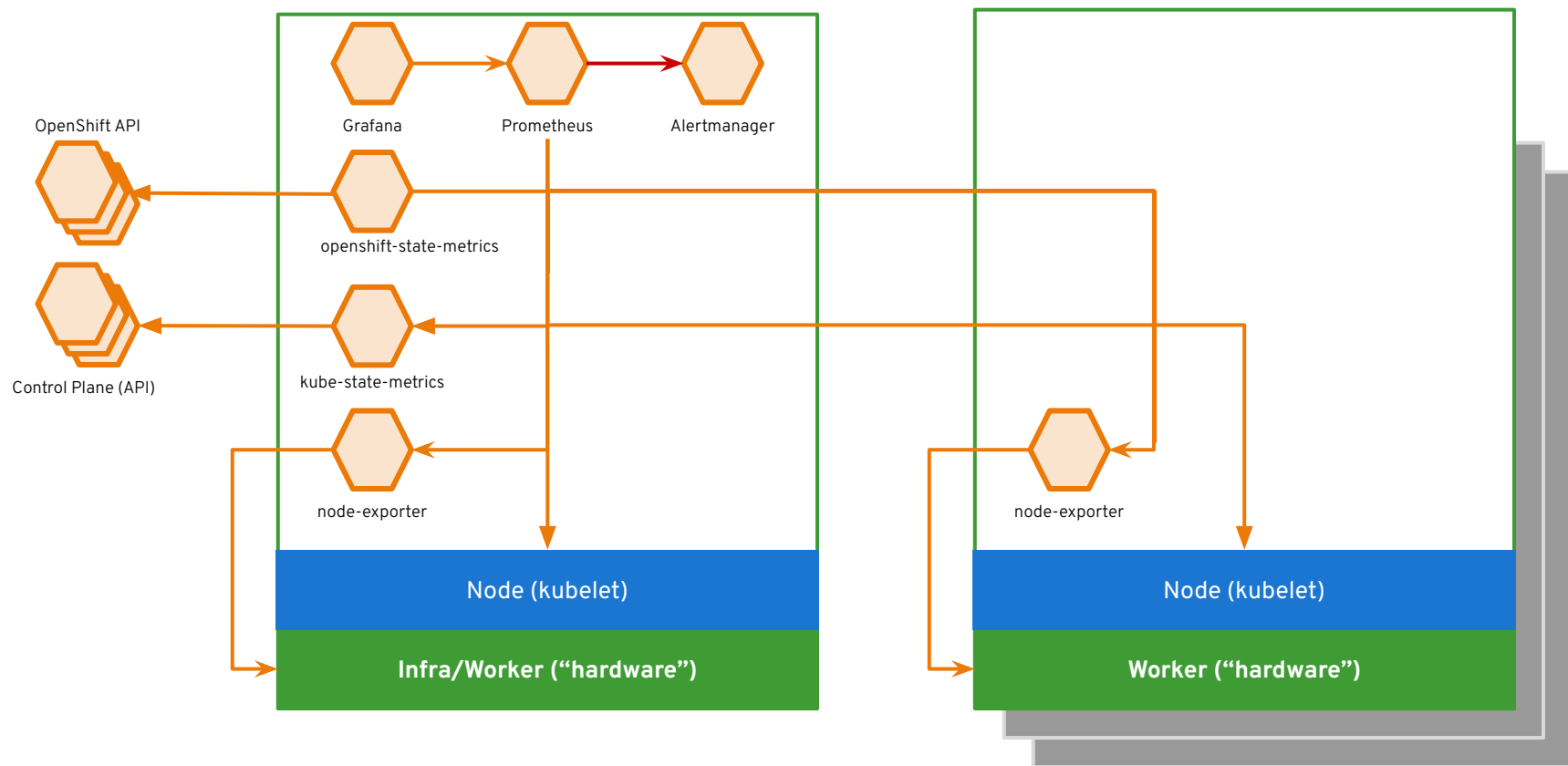


Alerting/notification via
Prometheus' Alertmanager, an
open-source tool that handles
alerts send by Prometheus.



Metrics visualization via
Grafana, the leading metrics
visualization technology.





OpenShift Logging

An integrated solution
for exploring and
corroborating
application logs

Observability via log exploration and corroboration with EFK

Components

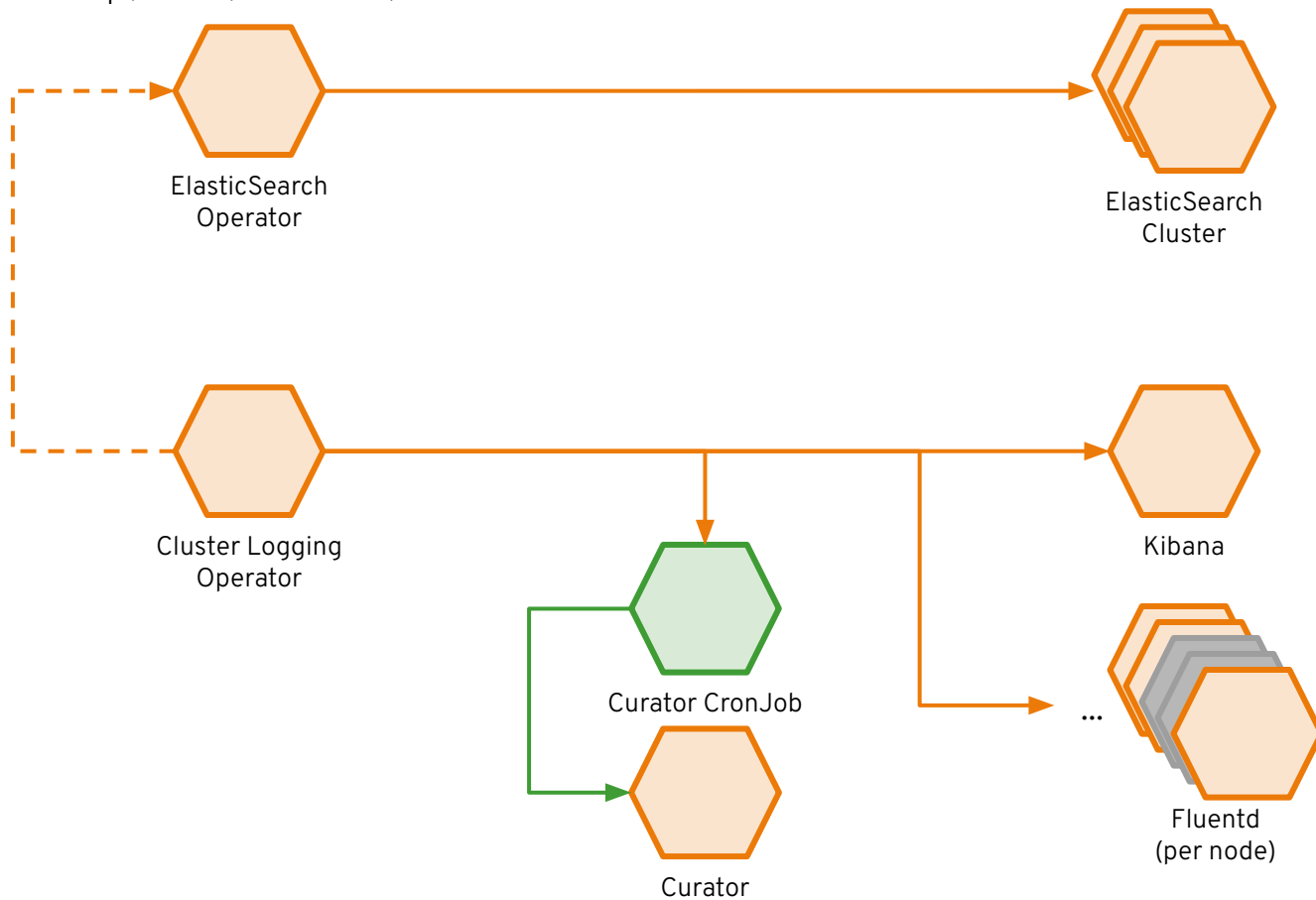
- **Elasticsearch:** a search and analytics engine to store logs
- **Fluentd:** gathers logs and sends to Elasticsearch.
- **Kibana:** A web UI for Elasticsearch.

Access control

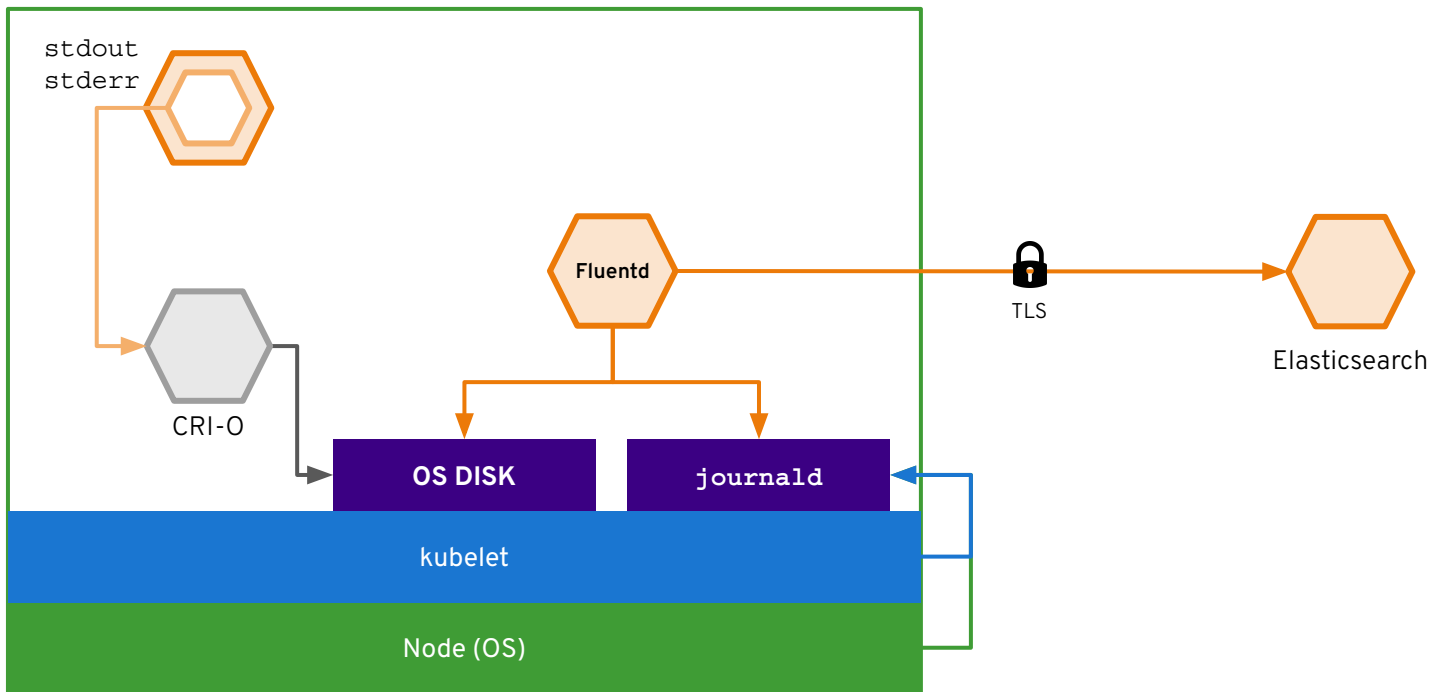
- Cluster administrators can view all logs
- Users can only view logs for their projects

Ability to forward logs elsewhere

- External elasticsearch, Splunk, etc



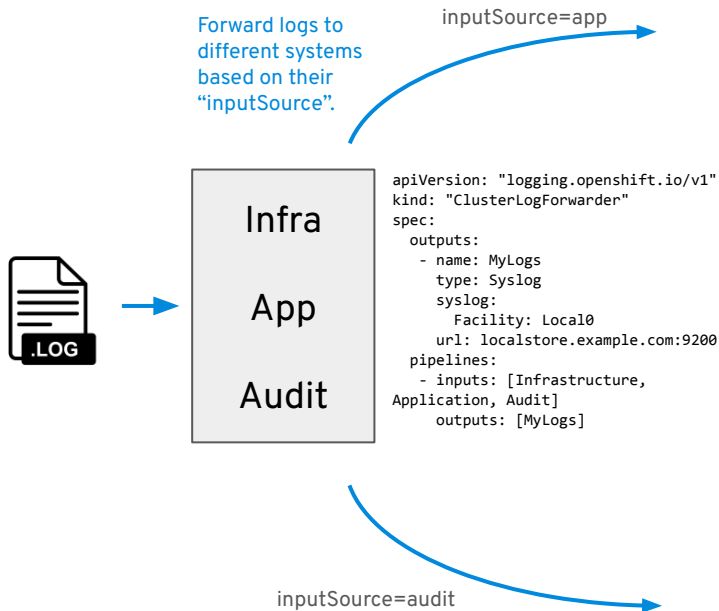
Log data flow in OpenShift



New log forwarding API (since 4.6)

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

- Introducing a new, cluster-wide *ClusterLogForwarder* CRD (API) that replaces needs to configure log forwarding via Fluentd ConfigMap.
- The API helps to reduce probability to misconfigure Fluentd and helps bringing in more stability into the Logging stack.
- Features include: Audit log collection and forwarding, Kafka support, namespace- and source-based routing, tagging, as well as improvements to the existing log forwarding features (e.g. syslog RFC5424 support).



Secure Log Forwarding to 3rd party

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

"ClusterLogForwarder"
Custom Resource

Cluster Logging
Operator

watches

creates

elasticsearch

fluentd

kafka

SYS
LOG

External Logging system

Node

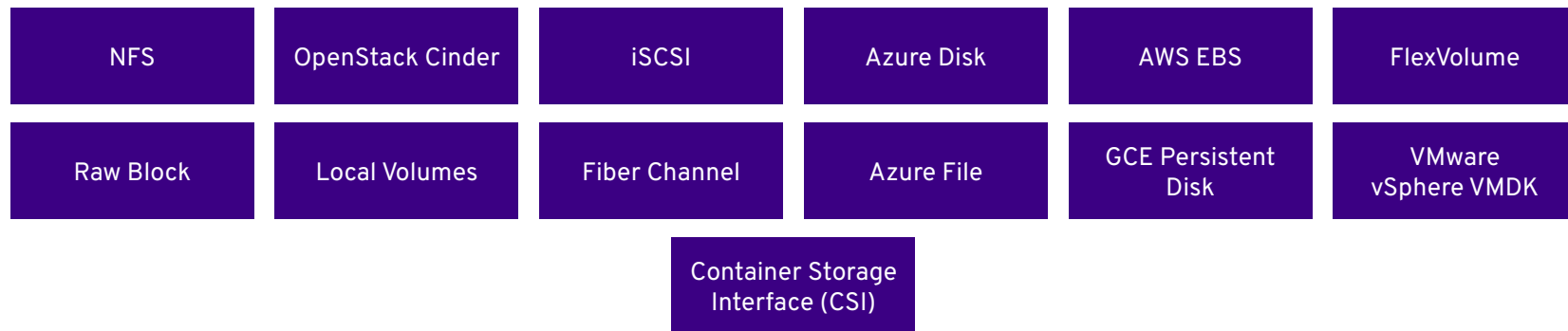
Fluentd
daemonset

Fluentd
forwarder

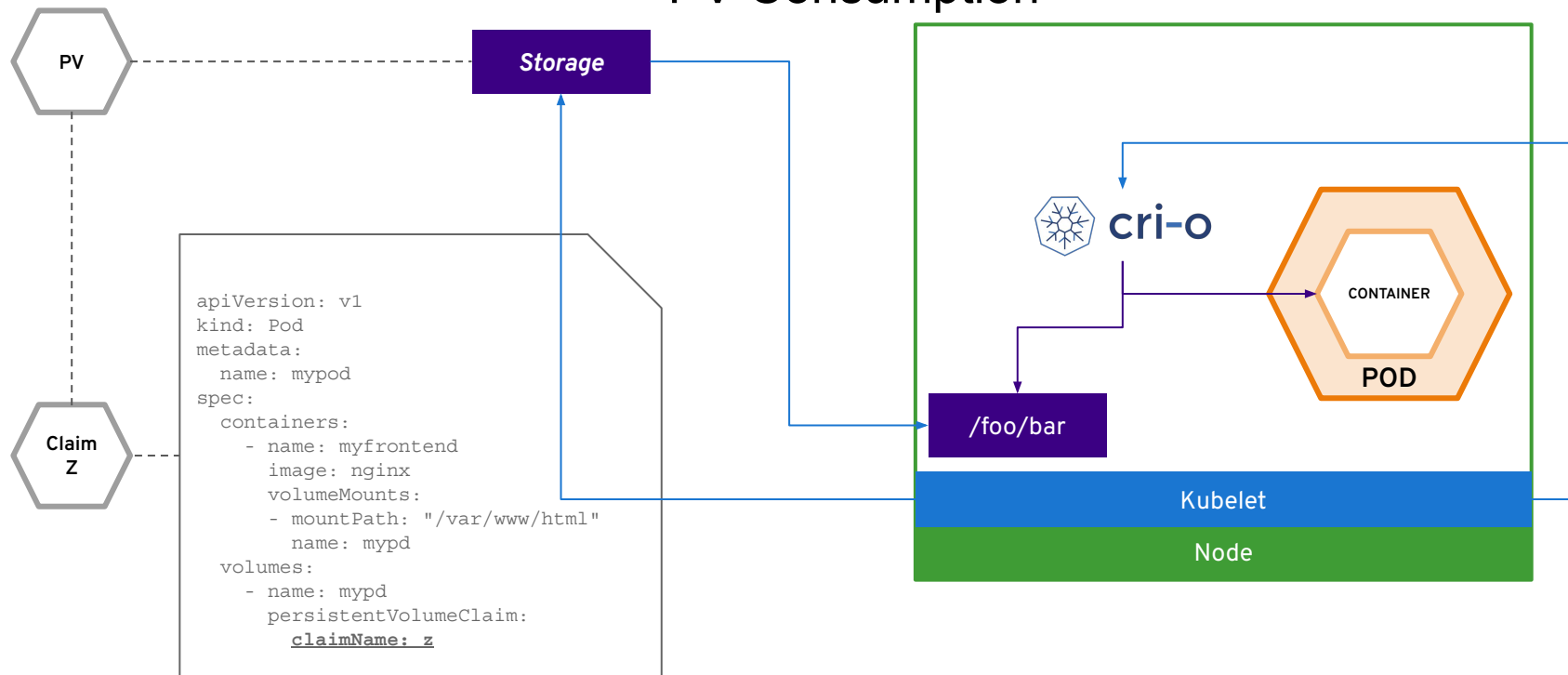
Persistent Storage

Connecting real-world
storage to your
containers to enable
stateful applications

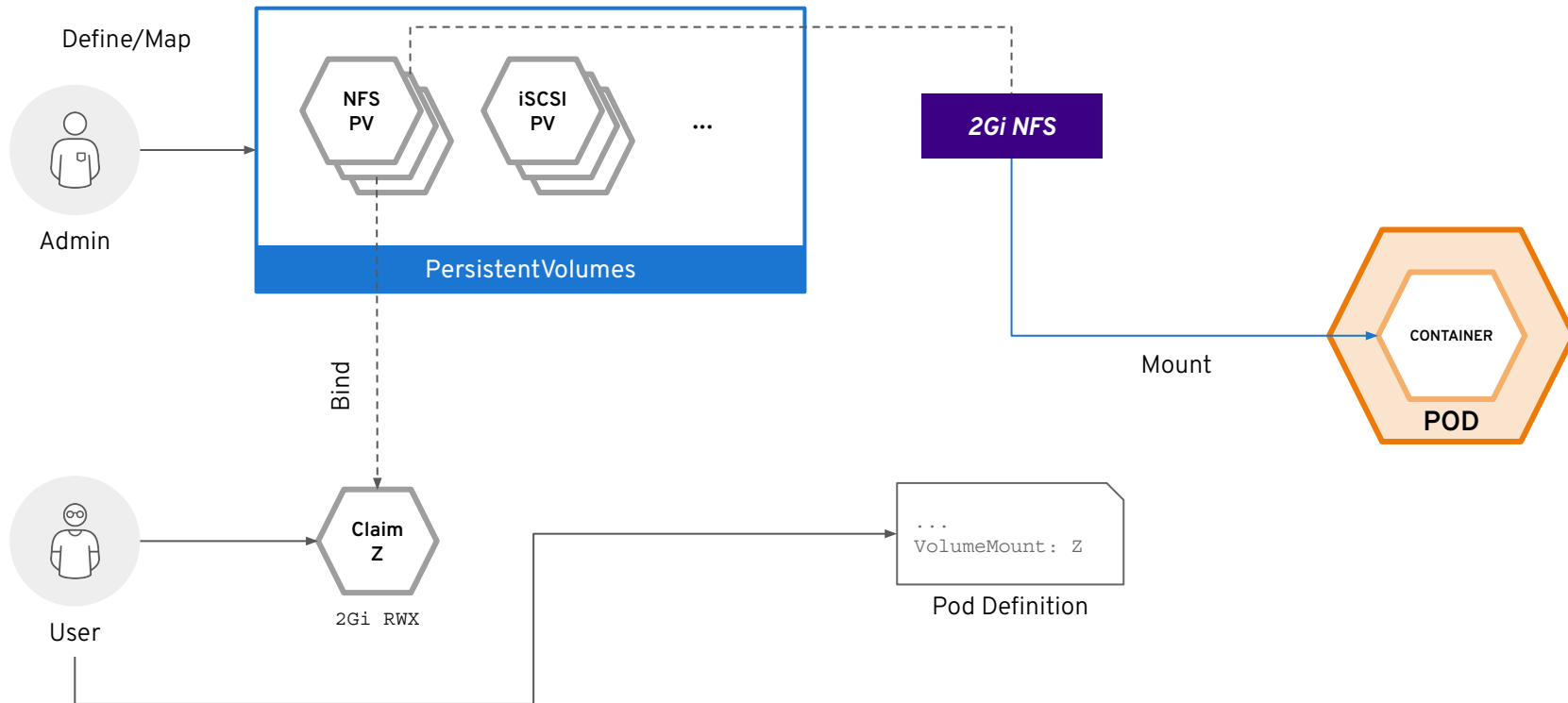
A broad spectrum of static and dynamic storage endpoints



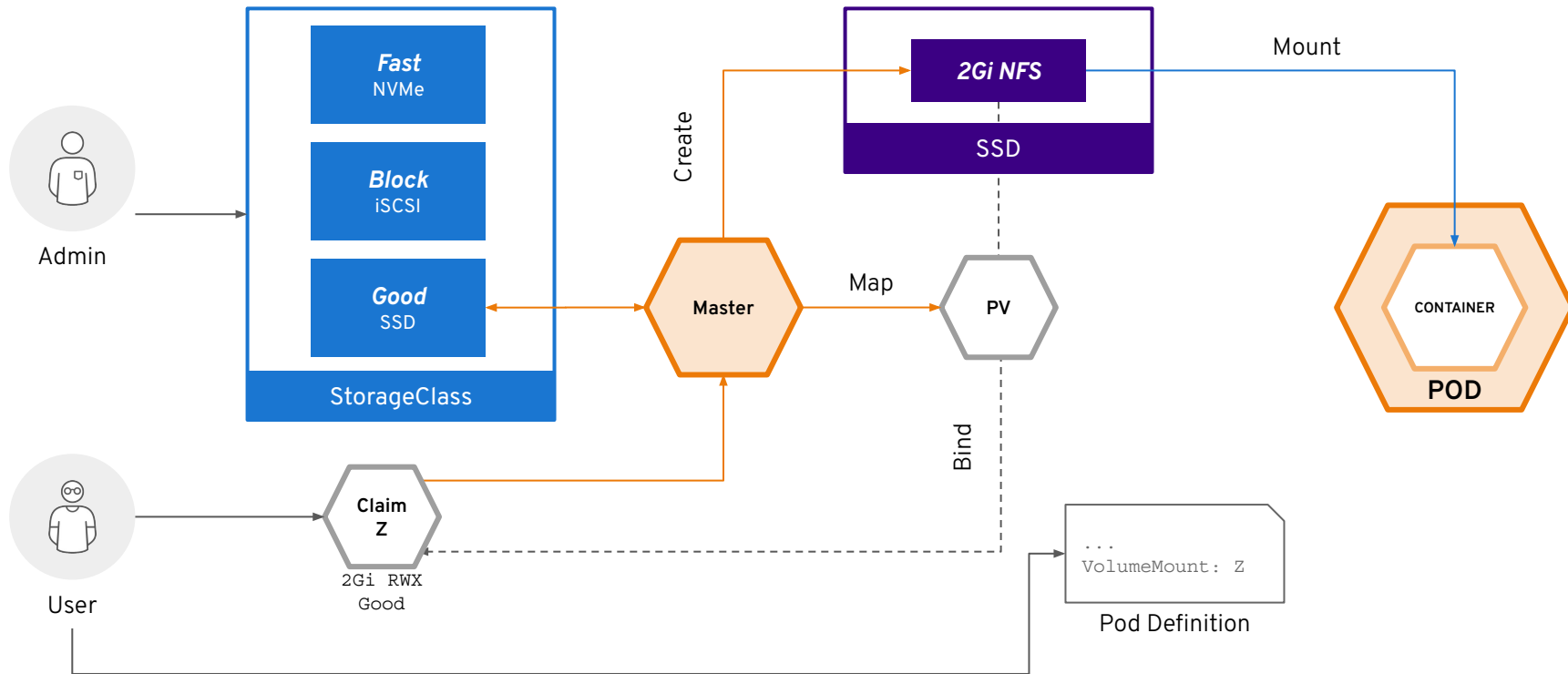
PV Consumption



Static Storage Provisioning



Dynamic Storage Provisioning





linkedin.com/company/red-hat



youtube.com/user/RedHatVideos



facebook.com/redhatinc



twitter.com/RedHat