OpenShift 4.x Architecture Workshop

OpenShift Container Platform (OCP)
Advanced Architecture

July 2019

Red Hat

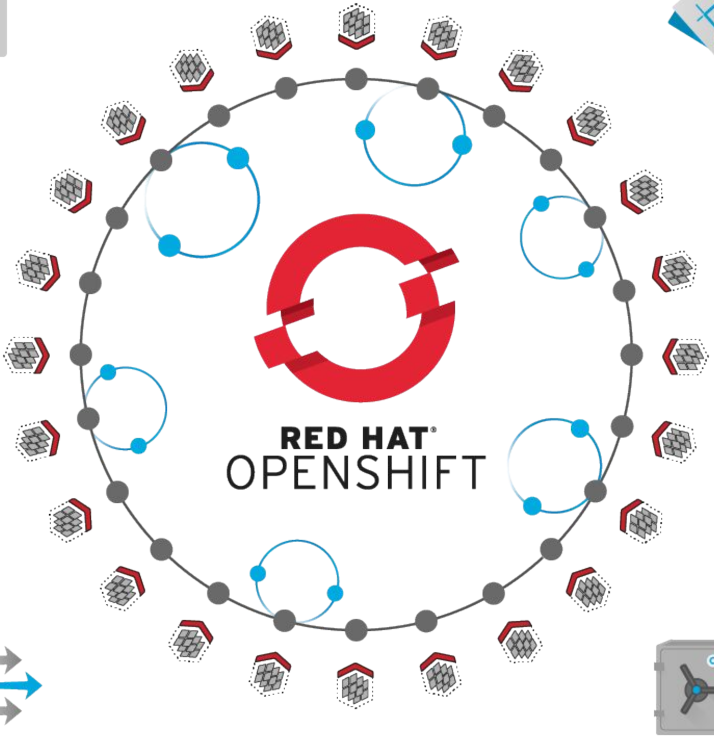
Self-Service

Multi-language

Automation

Collaboration

Multi-tenant



Standards-based

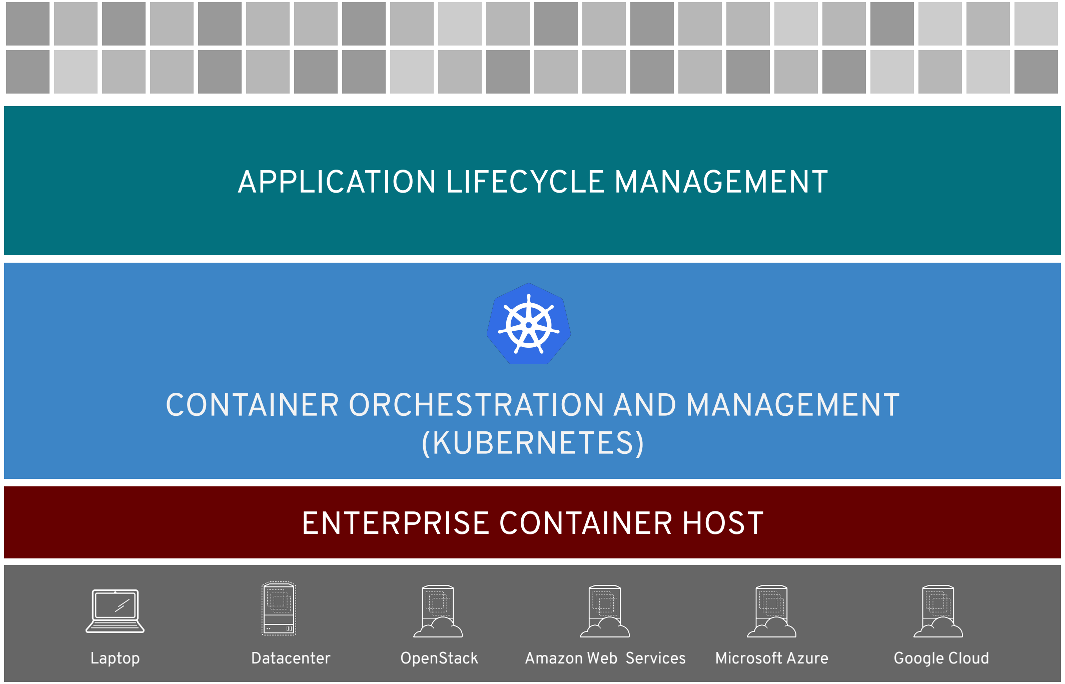
Web-scale

Open Source

Enterprise Grade

Secure

OPENSHIFT CONTAINER PLATFORM



APPLICATION LIFECYCLE MANAGEMENT

CONTAINER ORCHESTRATION AND MANAGEMENT
(KUBERNETES)

ENTERPRISE CONTAINER HOST

Laptop
Datacenter
OpenStack
Amazon Web Services
Microsoft Azure
Google Cloud

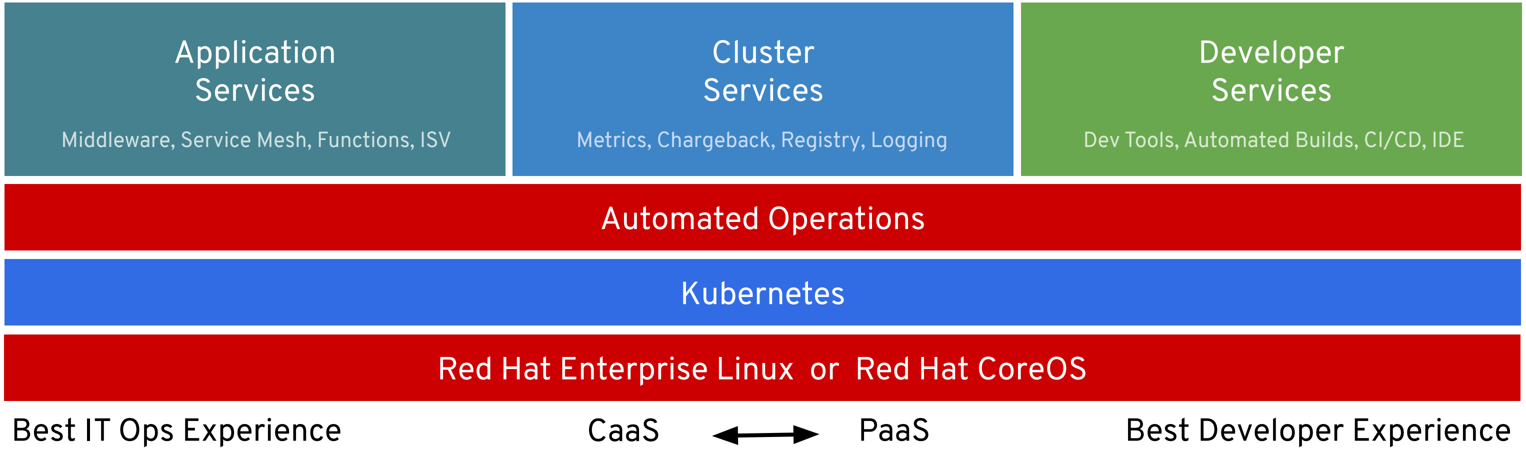
ANY
CONTAINER

RED HAT
OPENSHIFT

ANY
INFRASTRUCTURE

Red Hat

OPENSHIFT CONTAINER PLATFORM



Application Services

Middleware, Service Mesh, Functions, ISV

Cluster Services

Metrics, Chargeback, Registry, Logging

Developer Services

Dev Tools, Automated Builds, CI/CD, IDE

Automated Operations

Kubernetes

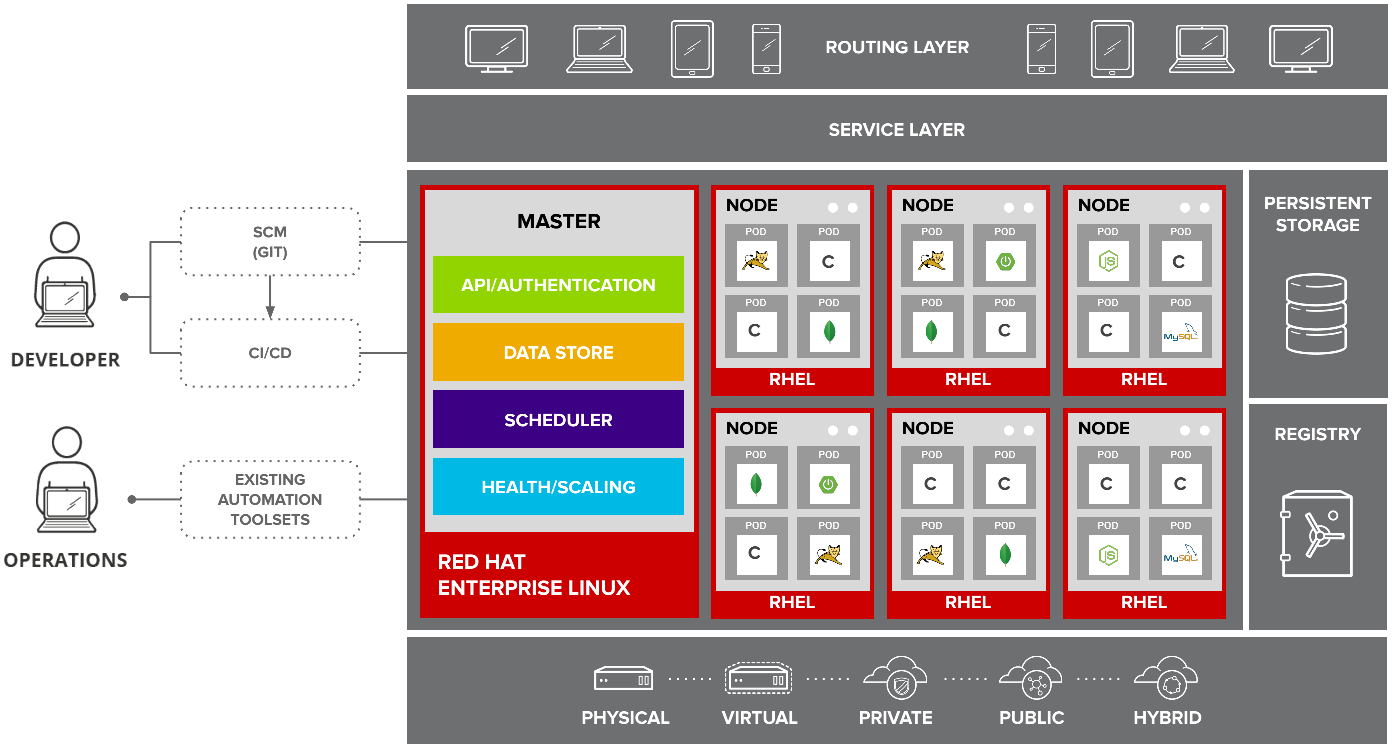
Red Hat Enterprise Linux or Red Hat CoreOS

Best IT Ops Experience

CaaS ⟷ PaaS

Best Developer Experience

OPENSHIFT ARCHITECTURE

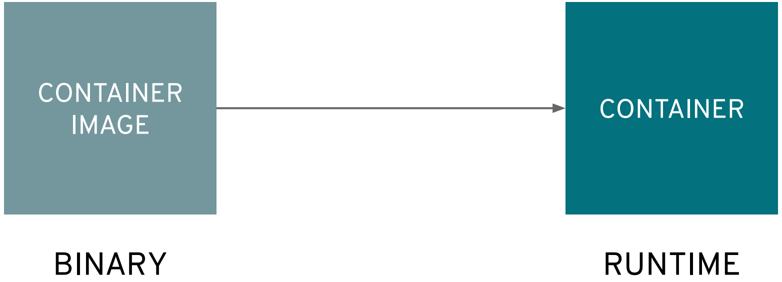


Cotainer Concepts Overview

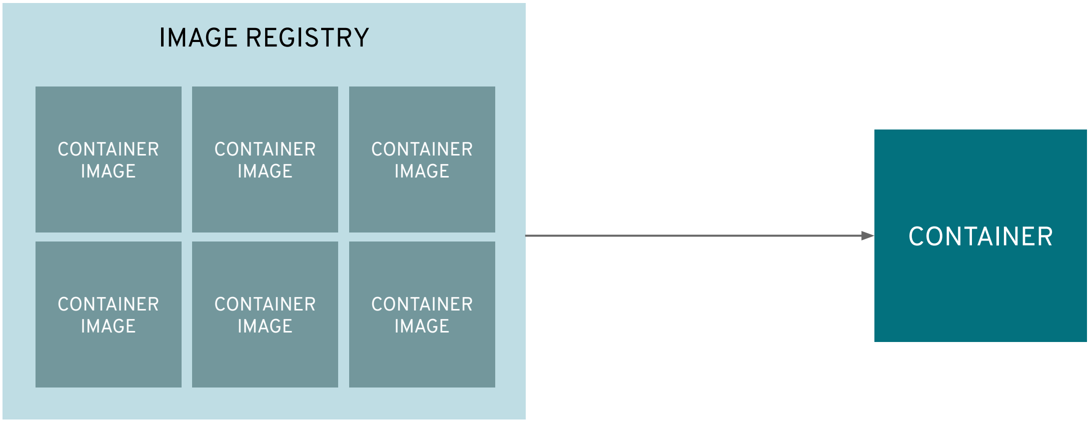
A container is the smallest compute unit

CONTAINER

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

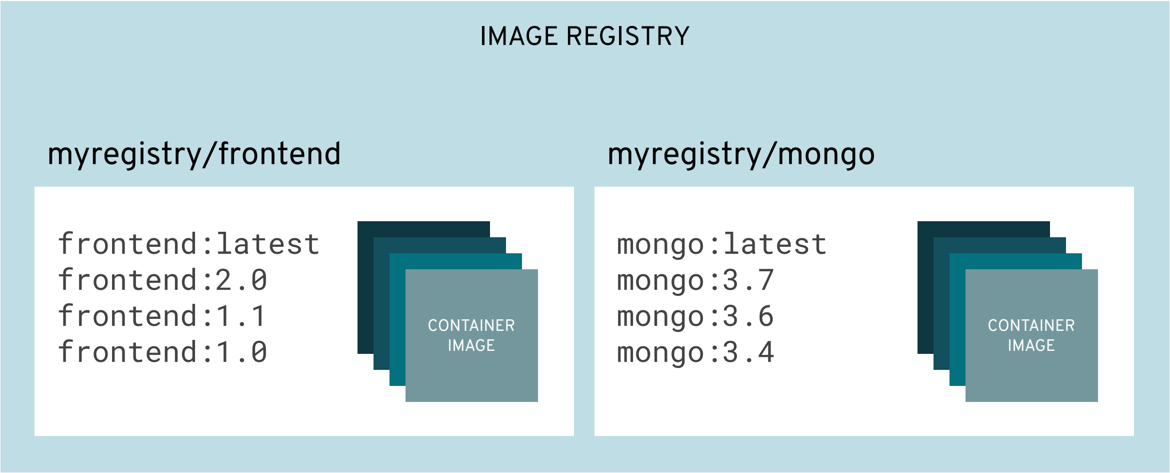


IMAGE REGISTRY

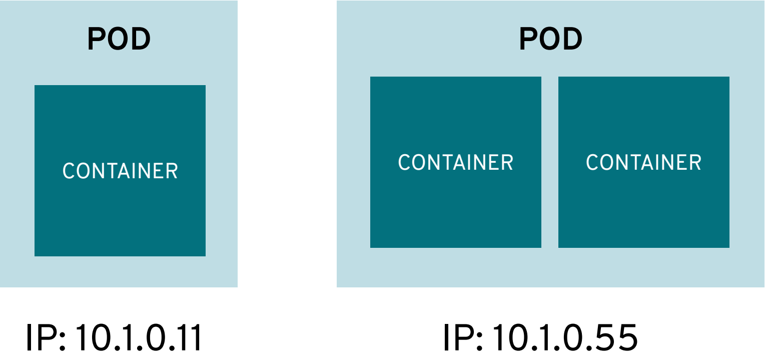
myregistry/frontend

```
frontend:latest
frontend:2.0
frontend:1.1
frontend:1.0
```

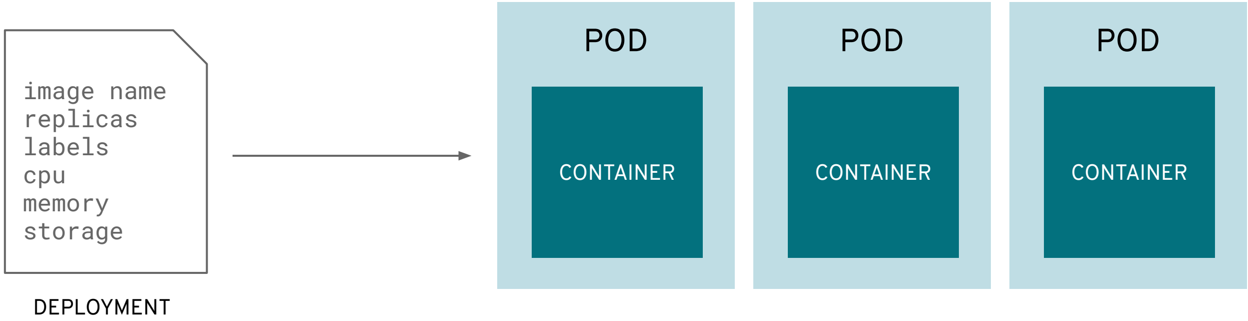
myregistry/mongo

```
mongo:latest
mongo:3.7
mongo:3.6
mongo:3.4
```

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

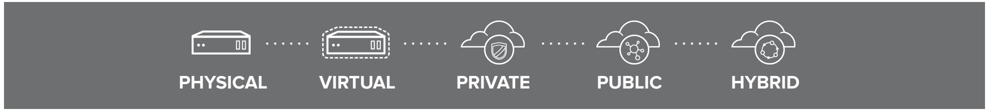


pods configuration is defined in a `deployment`

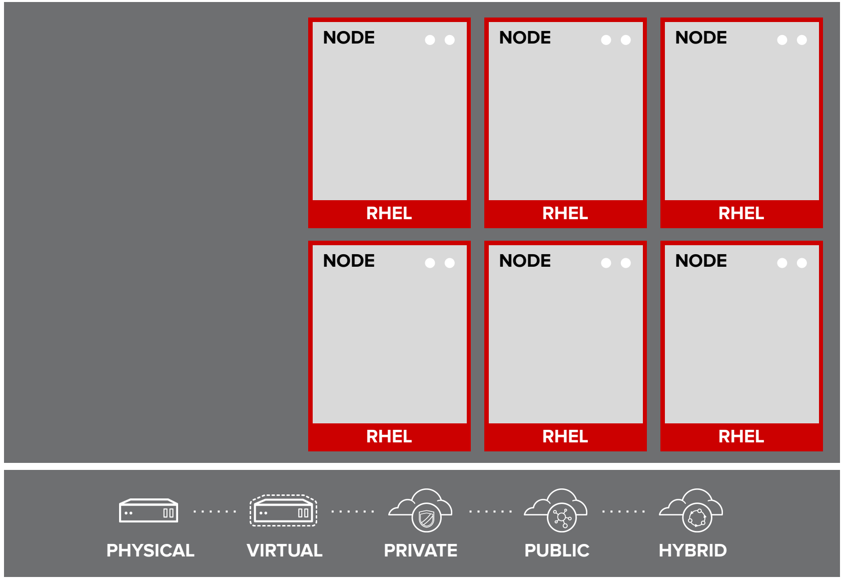


OpenShift Architecture

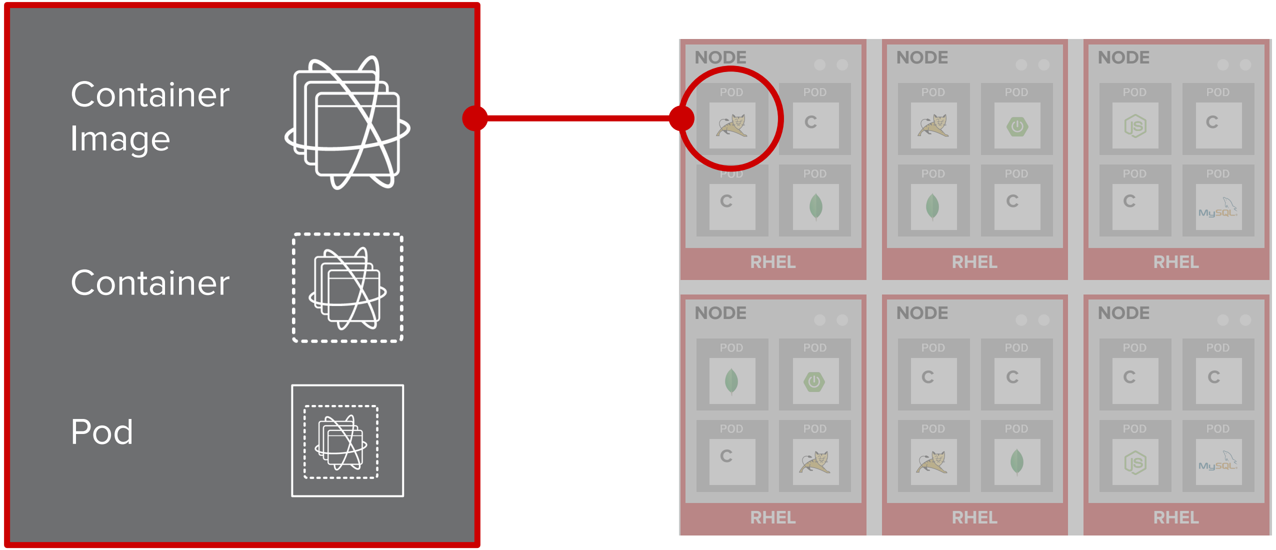
YOUR CHOICE OF INFRASTRUCTURE



NODES RHEL INSTANCES WHERE APPS RUN



APPS RUN IN CONTAINERS

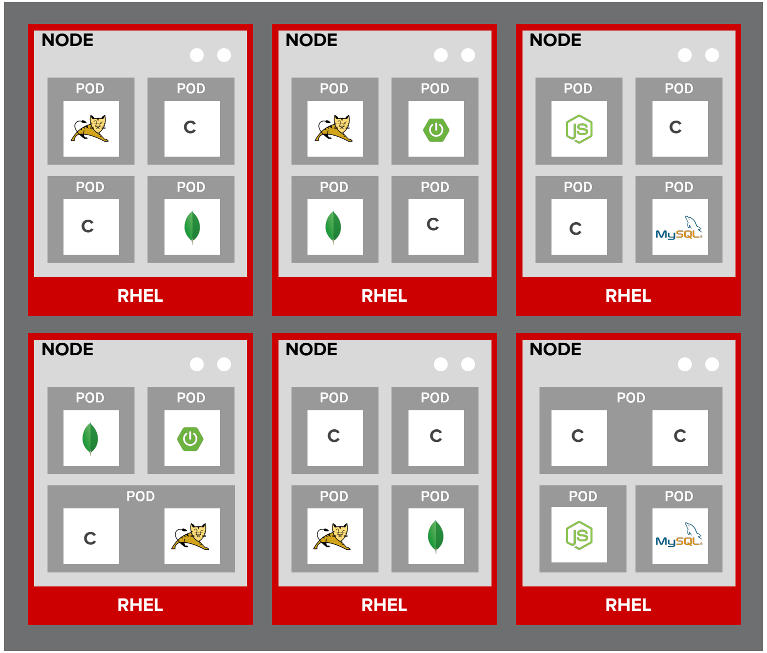


Container Image

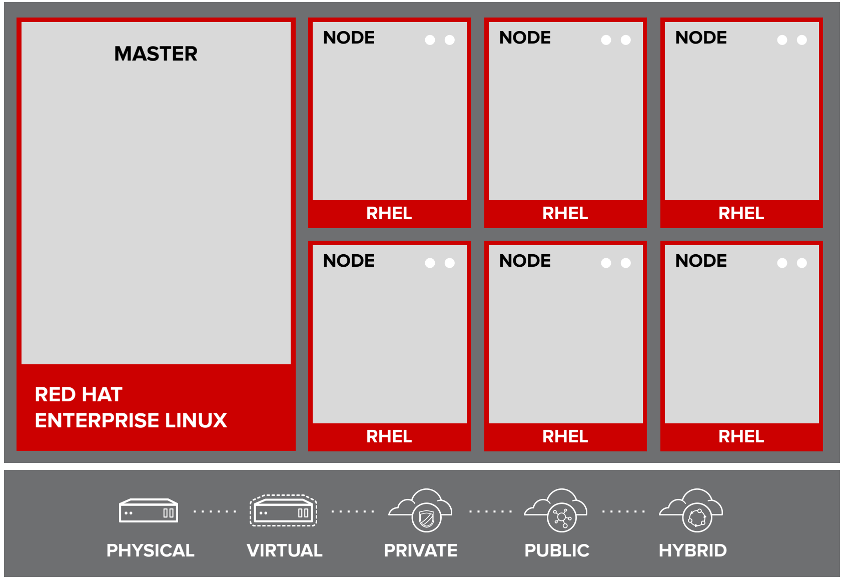
Container

Pod

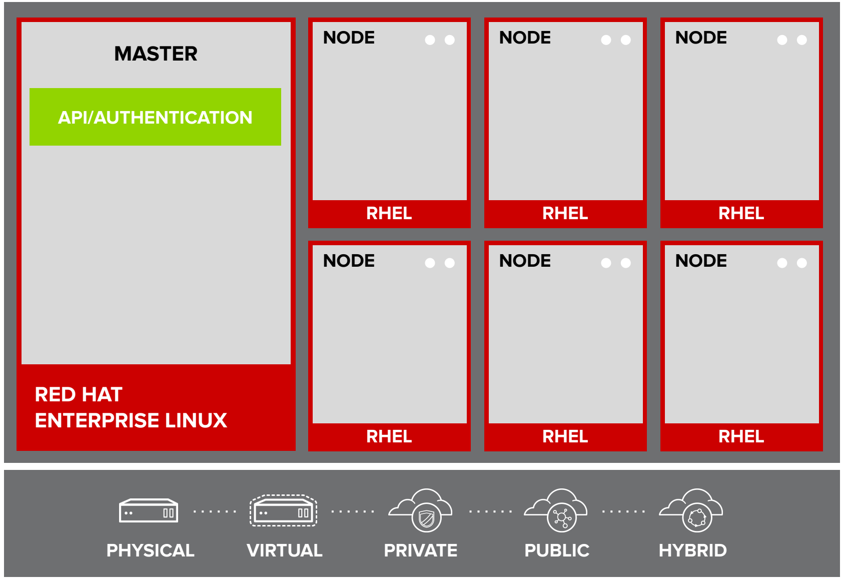
PODS ARE THE UNIT OF ORCHESTRATION



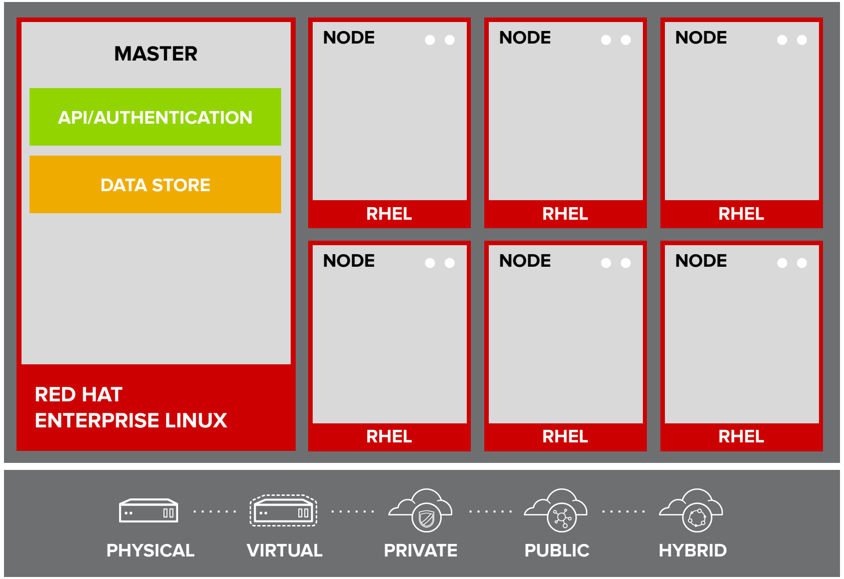
MASTERS ARE THE CONTROL PLANE



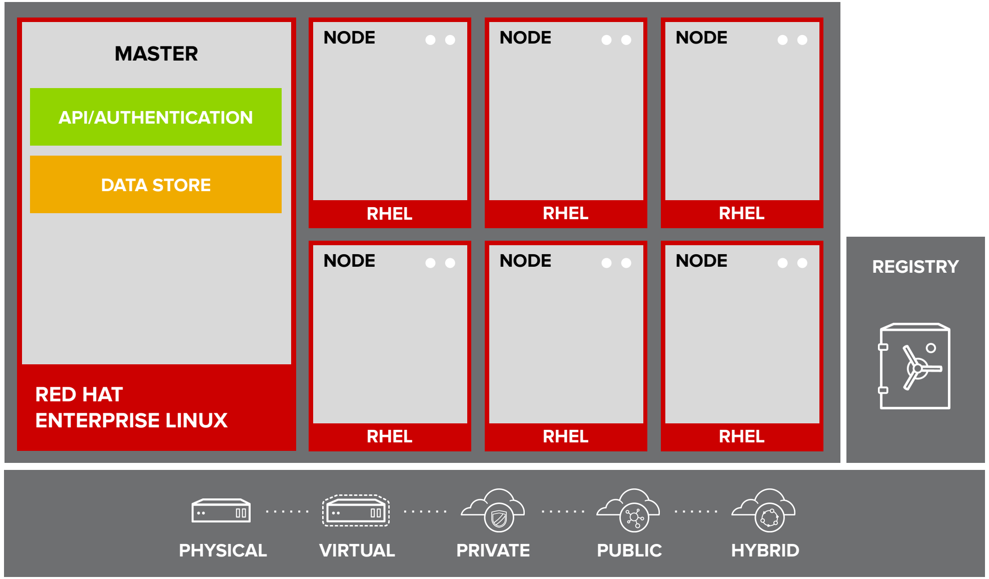
API AND AUTHENTICATION



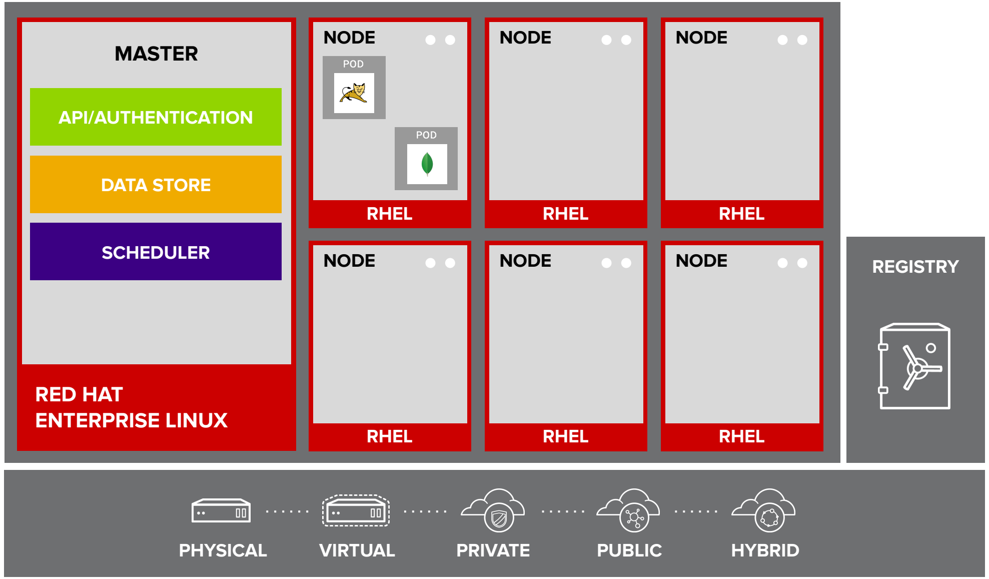
DESIRED AND CURRENT STATE



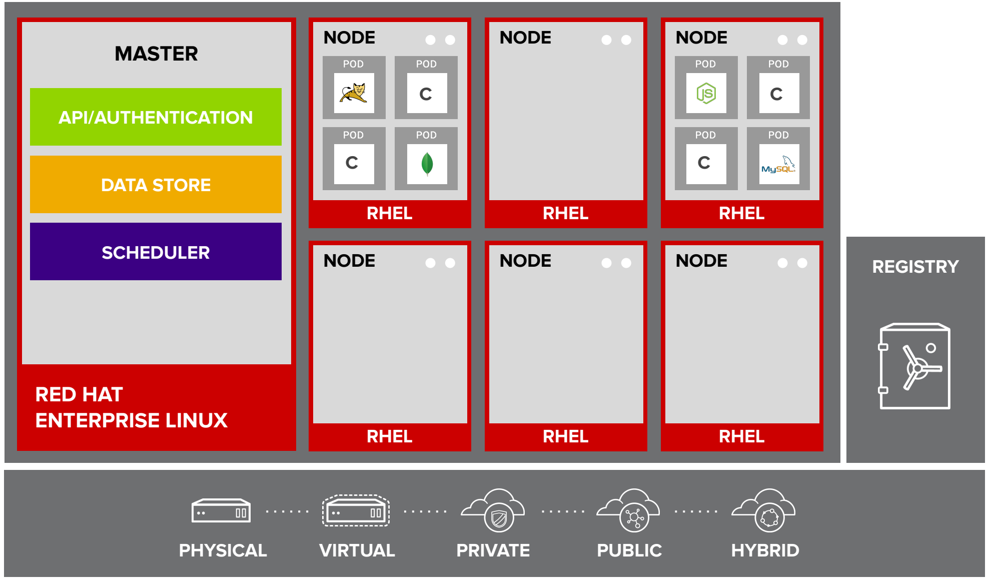
INTEGRATED CONTAINER REGISTRY



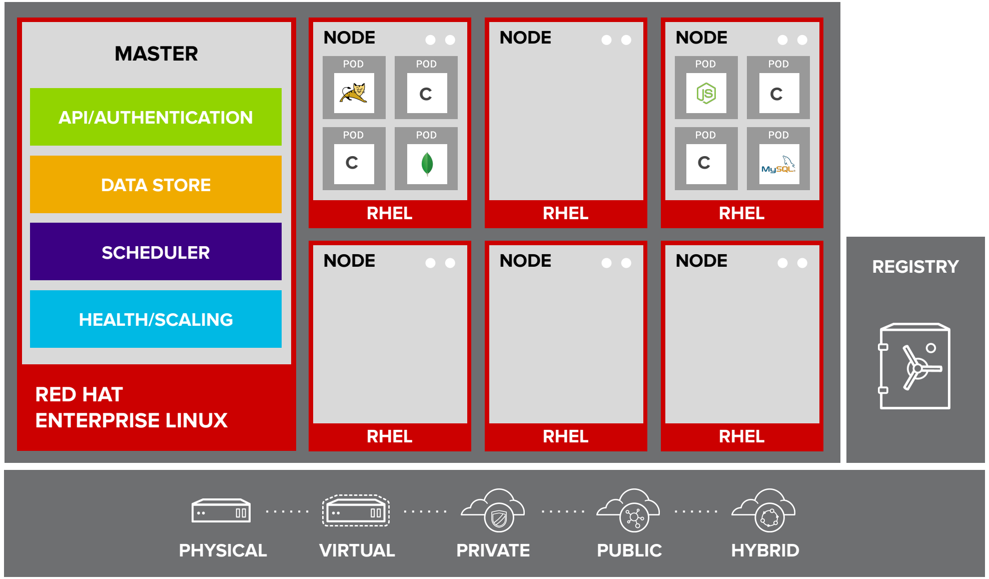
ORCHESTRATION AND SCHEDULING



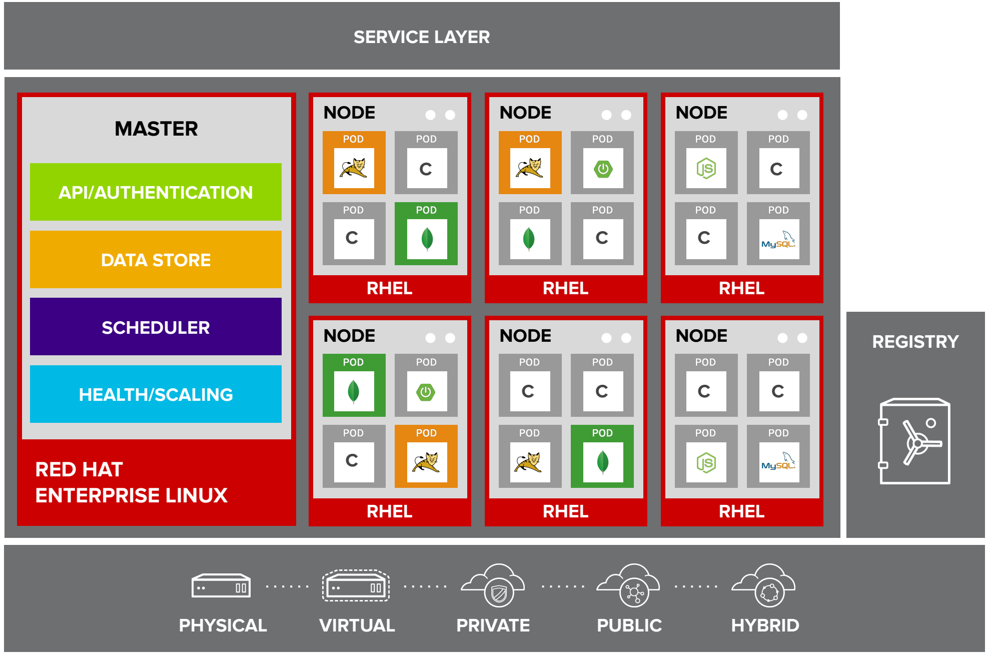
PLACEMENT BY POLICY



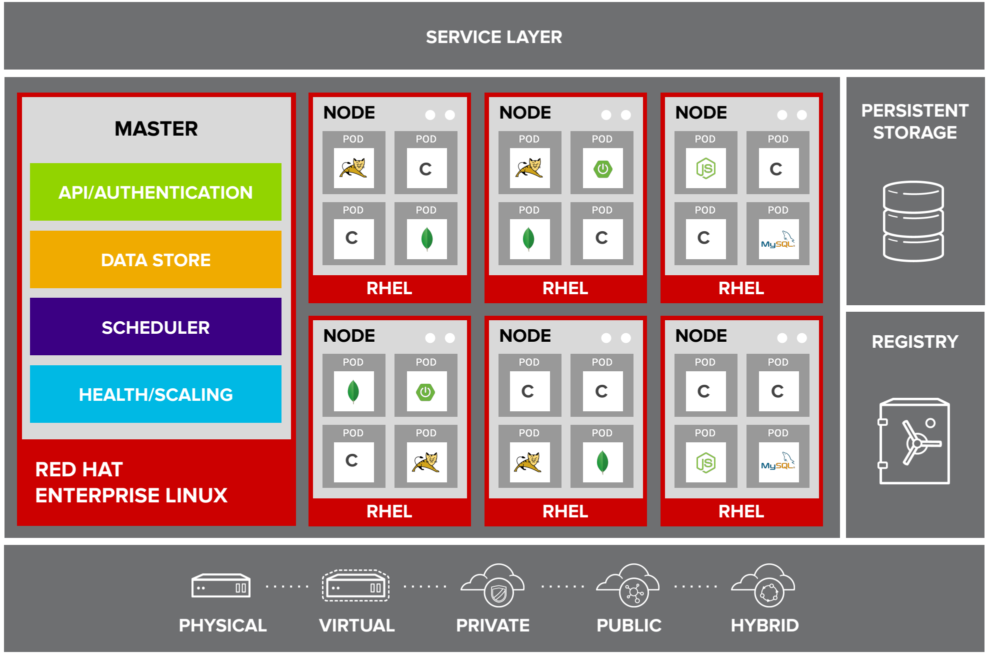
AUTOSCALING PODS



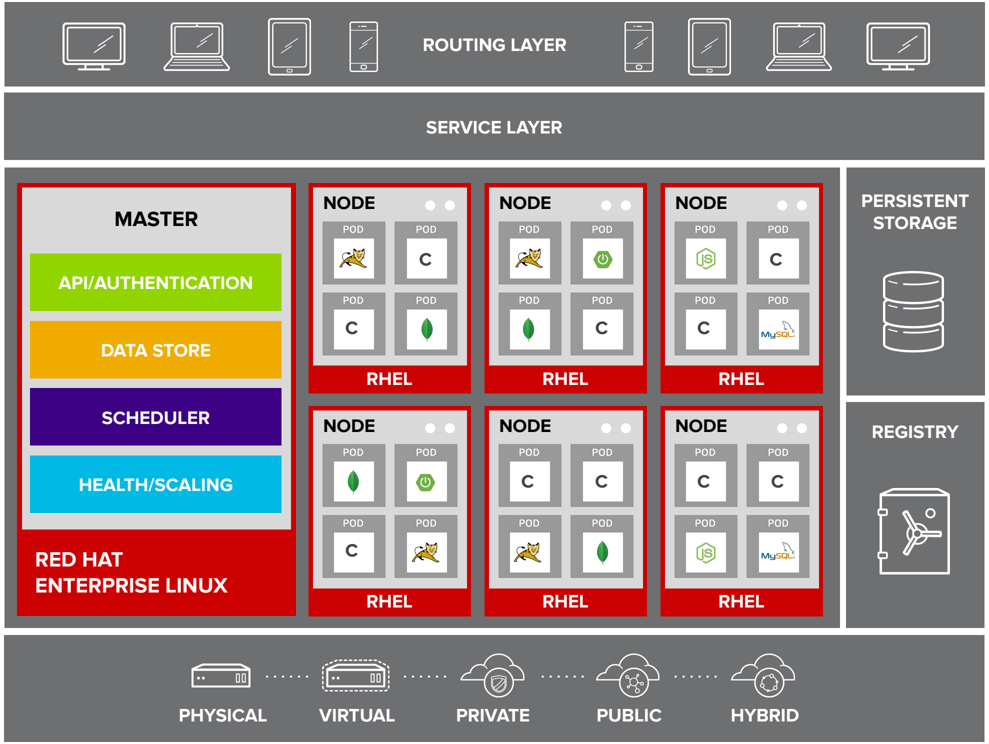
SERVICE DISCOVERY



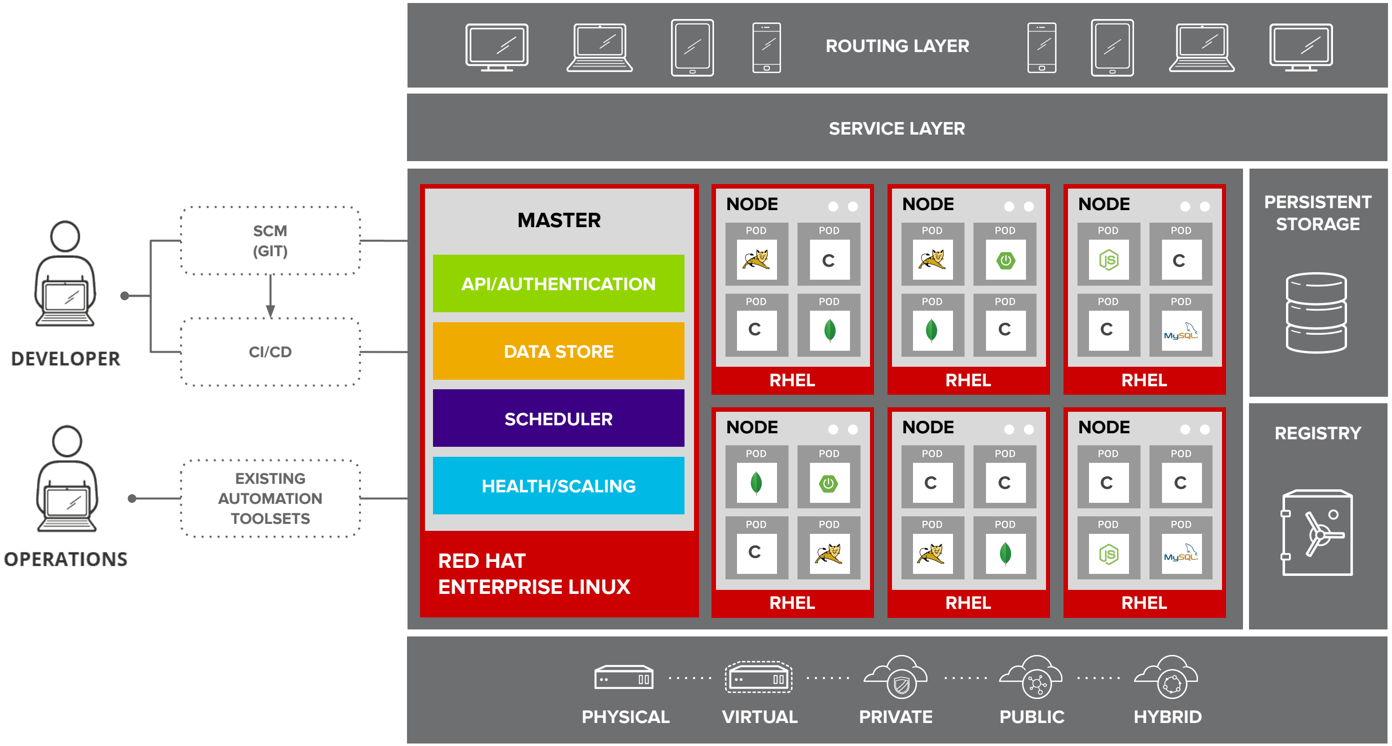
PERSISTENT DATA IN CONTAINERS



ROUTING AND LOAD-BALANCING



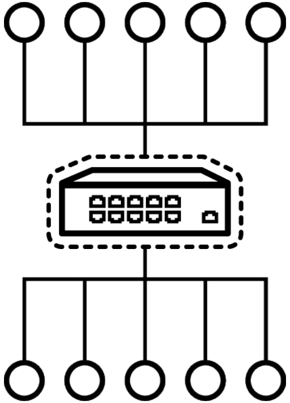
ACCESS VIA WEB, CLI, IDE AND API



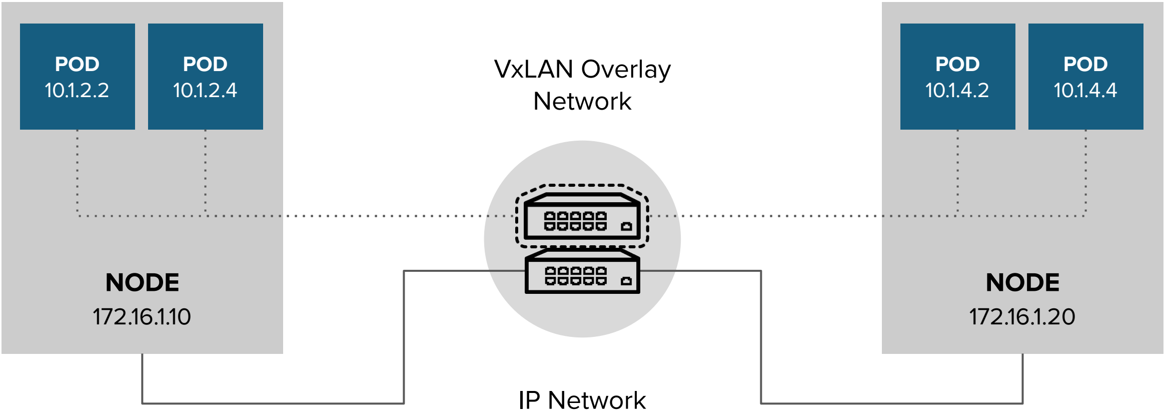
Networking

OPENSHIFT NETWORKING

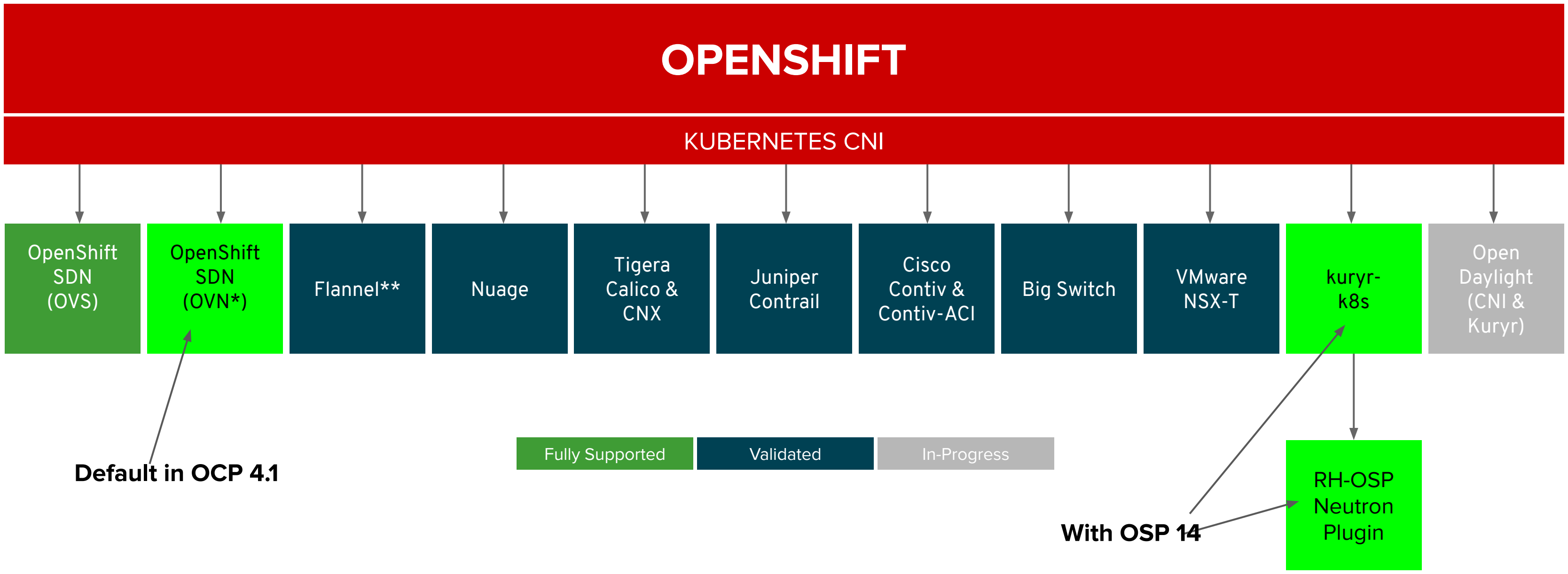
- Built-in internal DNS to reach services by name
- Split DNS is supported via DNSmasq
  - Master answers DNS queries for internal services
  - Other name servers serve the rest of the queries
- Software Defined Networking (SDN) for a unified cluster network to enable pod-to-pod communication
- OpenShift follows the Kubernetes Container Networking Interface (CNI) plug-in model



OPENSHIFT NETWORKING



OPENSHIFT NETWORK PLUGINS



OPENSHIFT

KUBERNETES CNI

OpenShift SDN (OVS)

OpenShift SDN (OVN*)

Flannel**

Nuage

Tigera Calico & CNX

Juniper Contrail

Cisco Contiv & Contiv-ACI

Big Switch

VMware NSX-T

kuryr-k8s

Open Daylight (CNI & Kuryr)

RH-OSP Neutron Plugin

Fully Supported | Validated | In-Progress

Default in OCP 4.1

With OSP 14

OPENSHIFT SDN

FLAT NETWORK

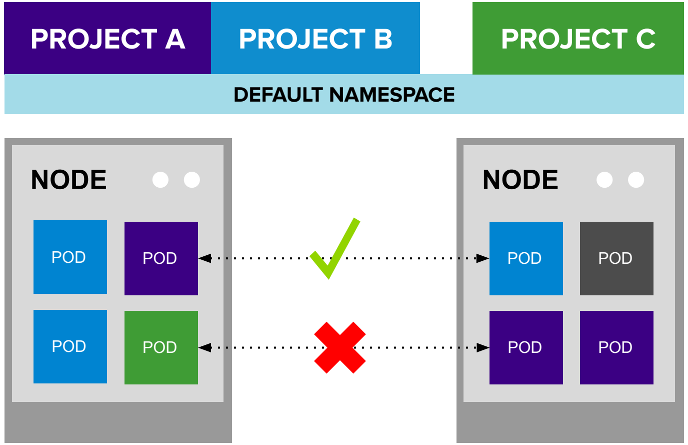
- All pods can communicate with each other across projects

MULTI-TENANT NETWORK

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

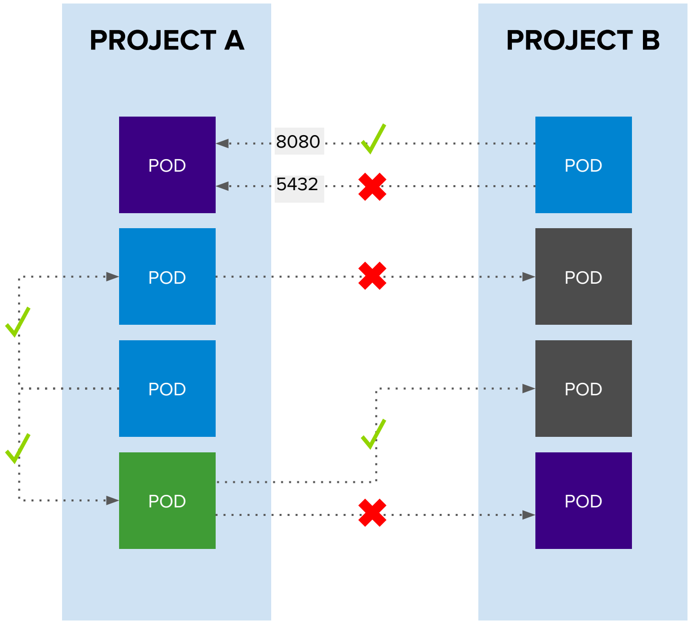
NETWORK POLICY (Default)

- Granular policy-based isolation



Multi-Tenant Network

OPENSHIFT SDN - NETWORK POLICY

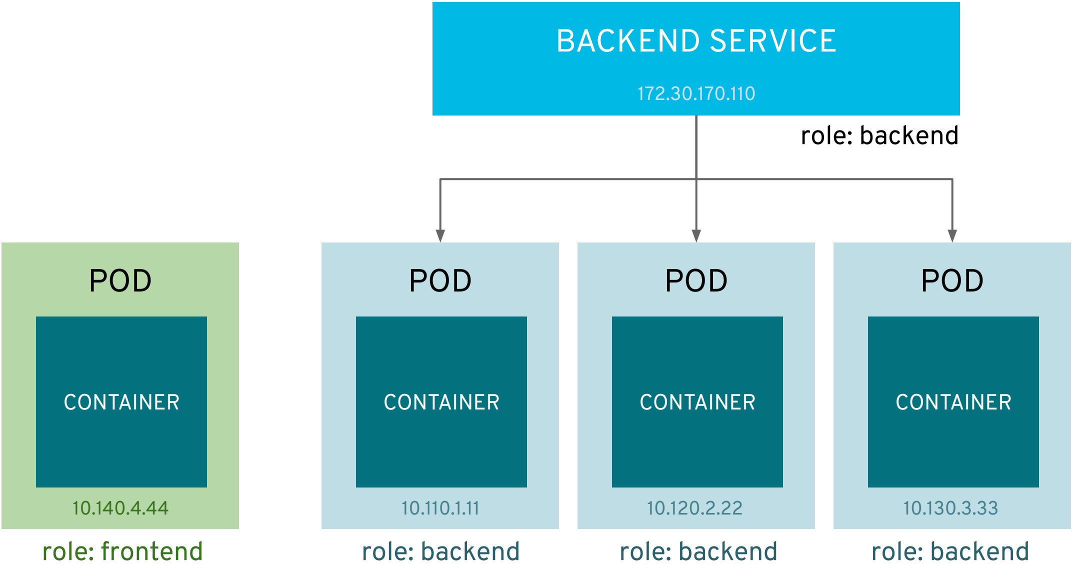


Example Policies

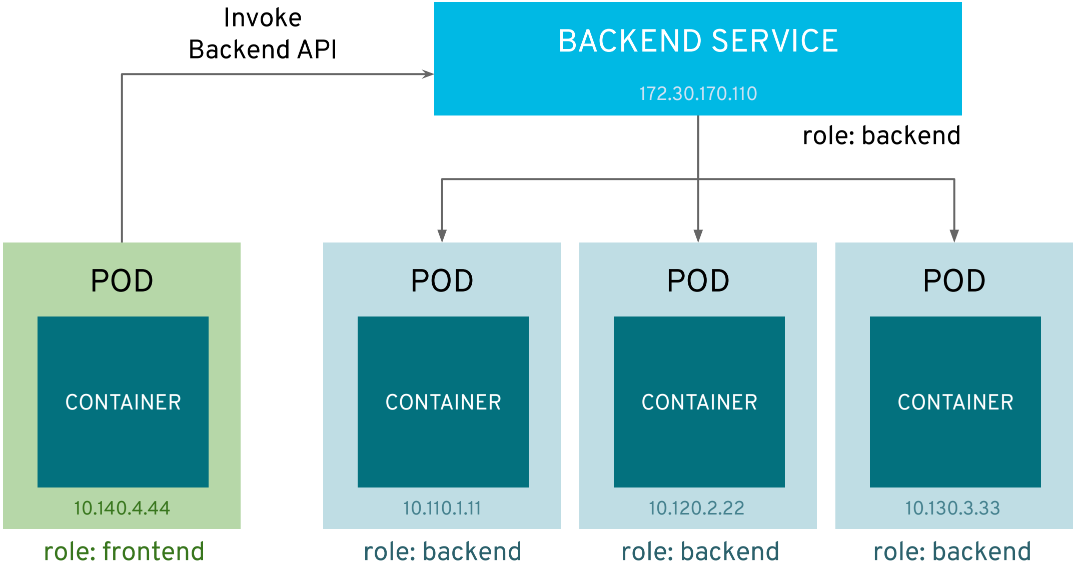
- Allow all traffic inside the project
- Allow traffic from green to gray
- Allow traffic to purple on 8080

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

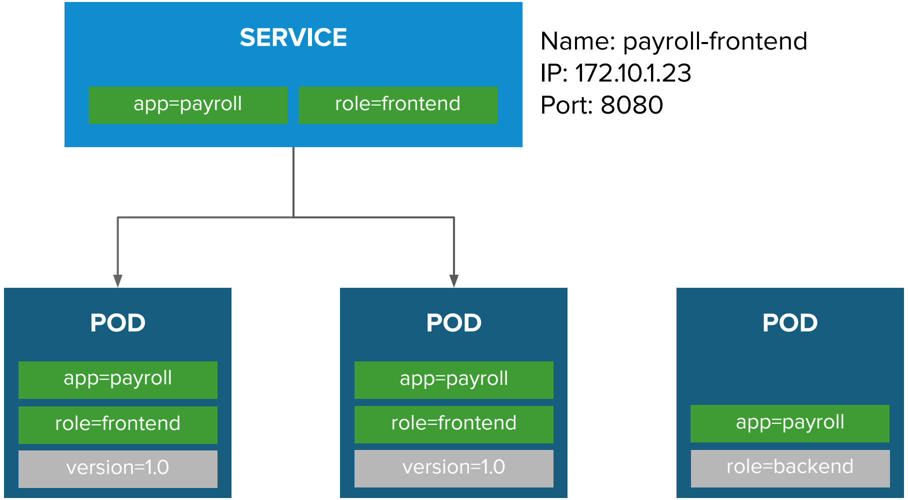
services provide internal load-balancing and service discovery across pods



apps can talk to each other via services

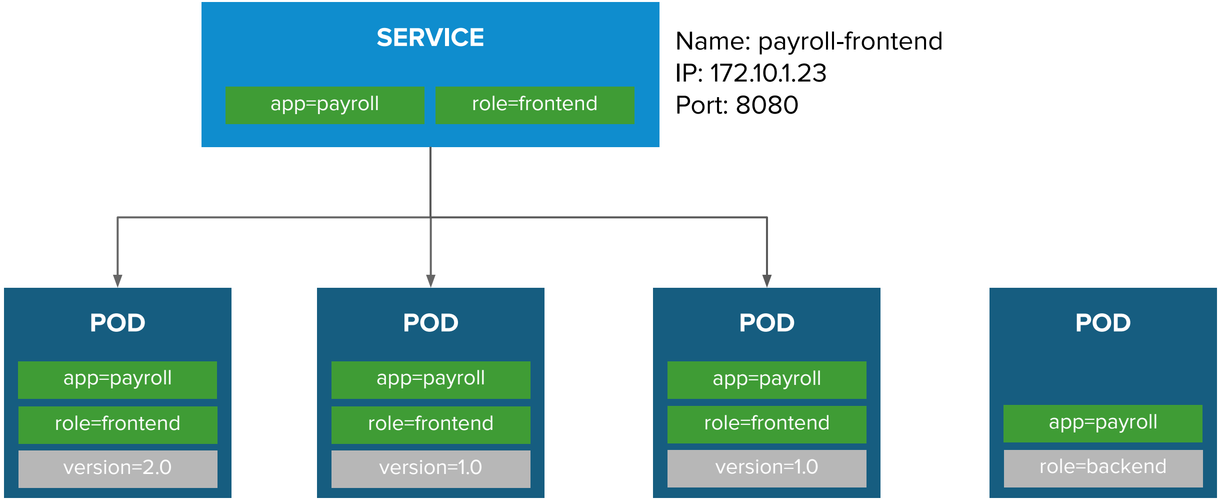


BUILT-IN SERVICE DISCOVERY INTERNAL LOAD-BALANCING



Name: payroll-frontend
IP: 172.10.1.23
Port: 8080

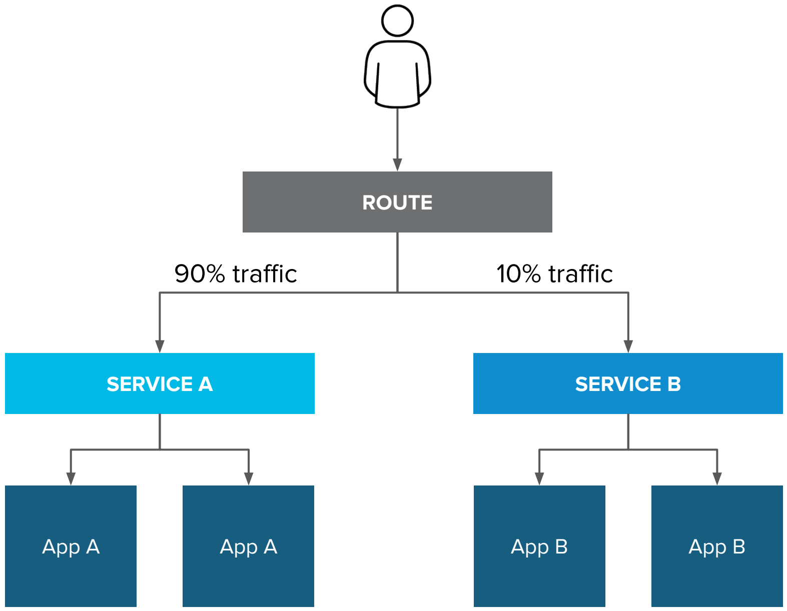
BUILT-IN SERVICE DISCOVERY
INTERNAL LOAD-BALANCING



Name: payroll-frontend
IP: 172.10.1.23
Port: 8080

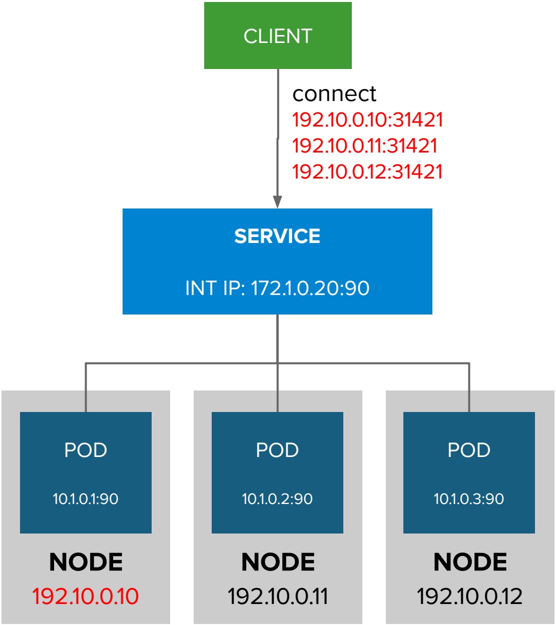
ROUTE SPLIT TRAFFIC

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



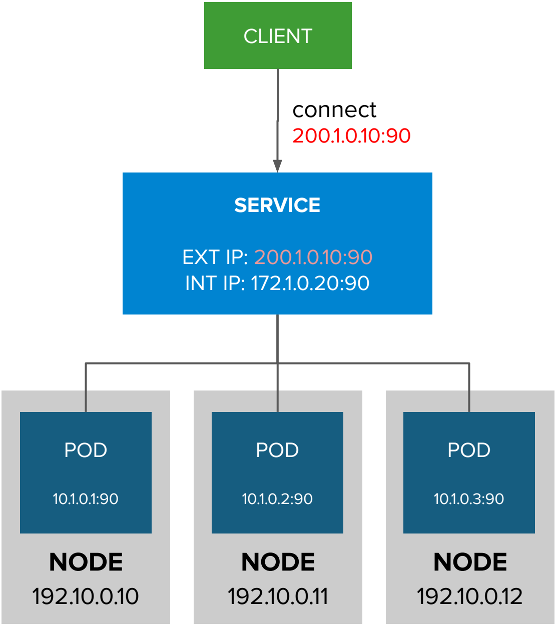
EXTERNAL TRAFFIC TO A SERVICE ON A RANDOM PORT WITH NODEPORT

- 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

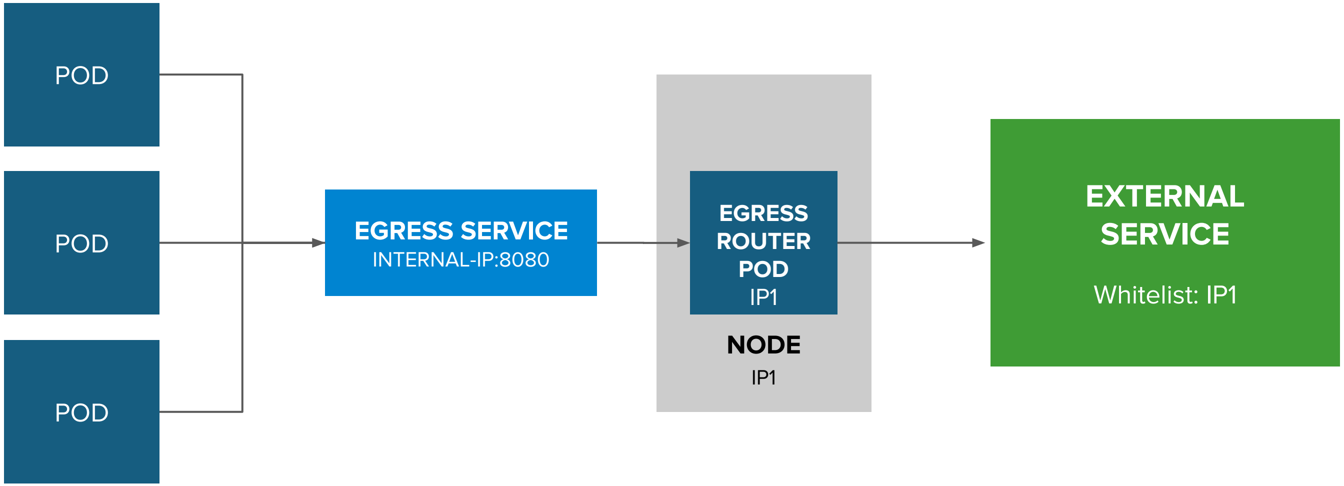


EXTERNAL TRAFFIC TO A SERVICE ON ANY PORT WITH INGRESS

- 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

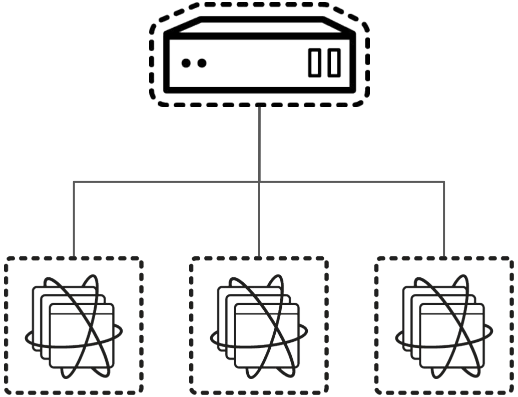


CONTROL OUTGOING TRAFFIC SOURCE IP WITH EGRESS ROUTER

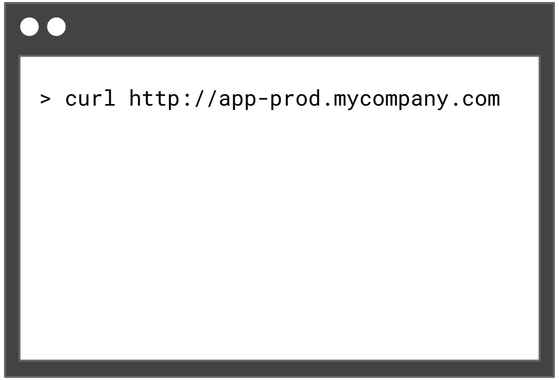


ROUTING AND EXTERNAL LOAD-BALANCING

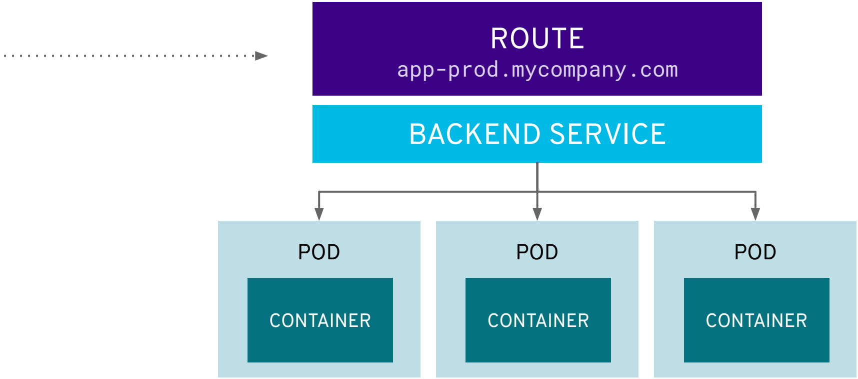
- Pluggable routing architecture
  - HAProxy Router
  - F5 Router
- Multiple-routers with traffic sharding
- Router supported protocols
  - HTTP/HTTPS
  - WebSockets
  - TLS with SNI
- Non-standard ports via cloud load-balancers, external IP, and NodePort



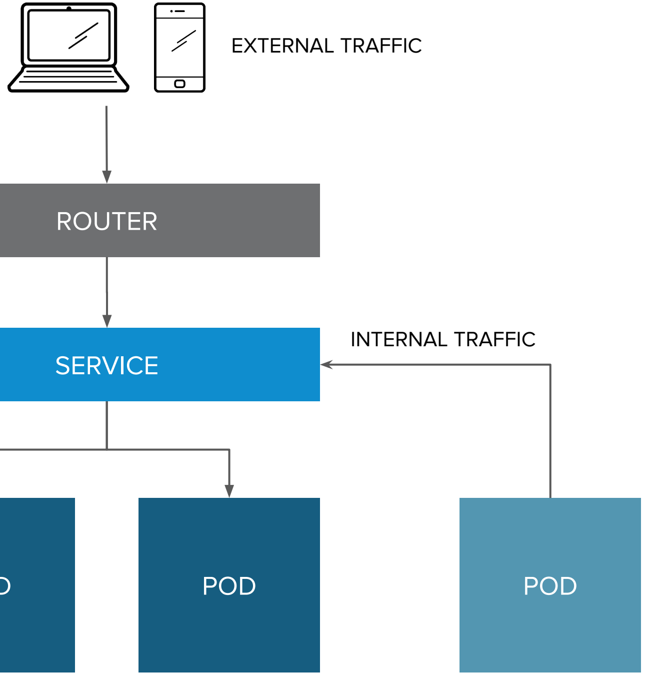
routes add services to the external load-balancer and provide readable urls for the app



```
> curl http://app-prod.mycompany.com
```

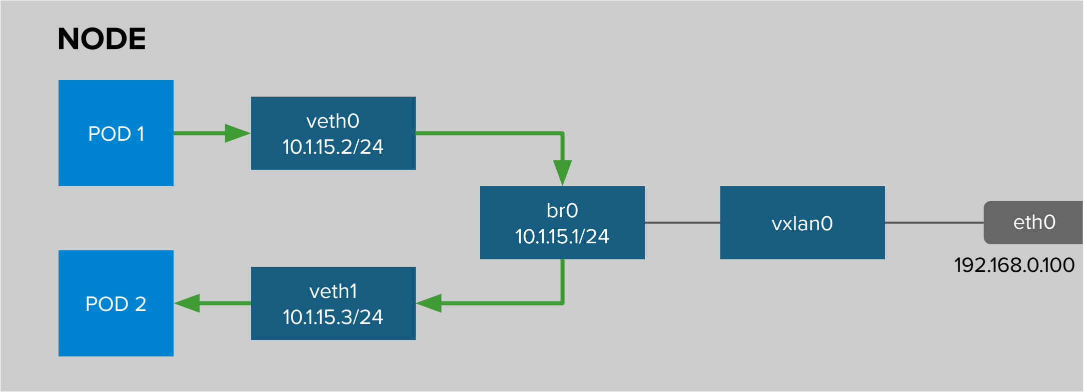


ROUTE EXPOSES SERVICES EXTERNALLY



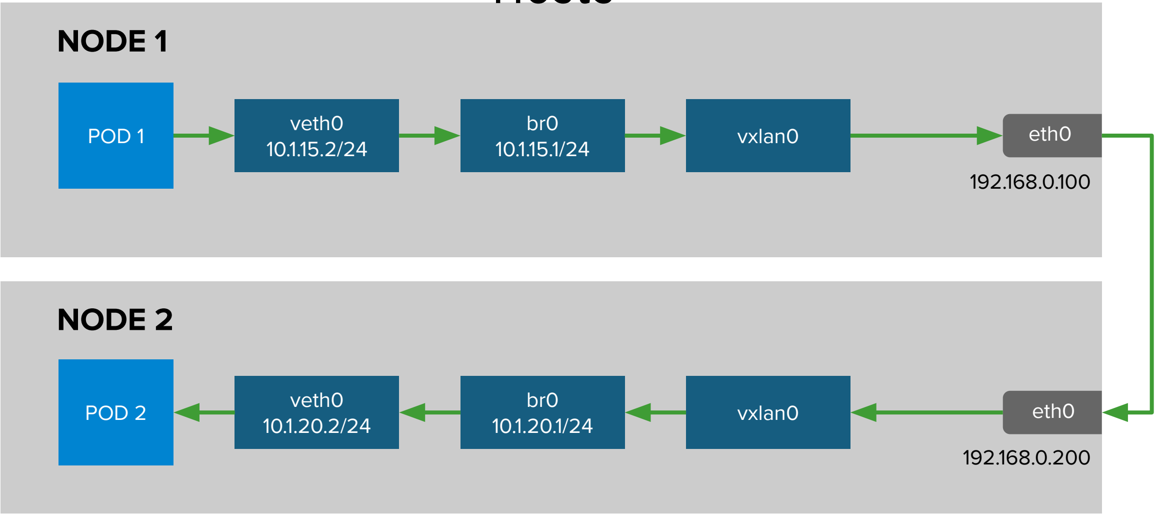
OPENSHIFT SDN - OVS PACKET FLOW

Container to Container on the Same Host



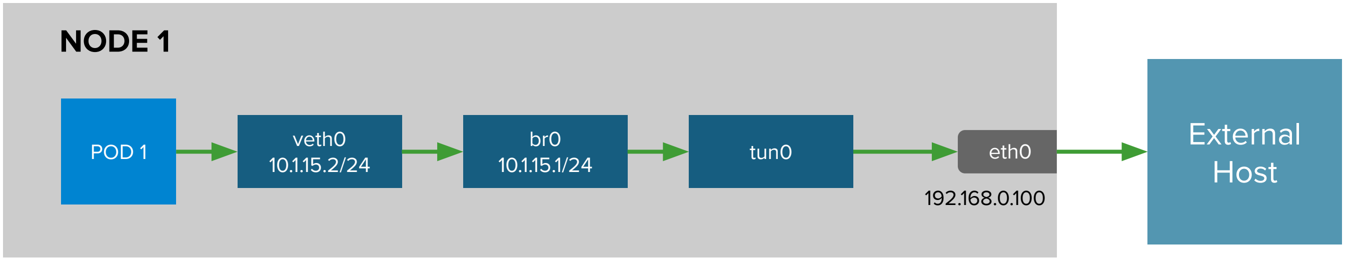
OPENSHIFT SDN - OVS PACKET FLOW

Container to Container on the Different Hosts



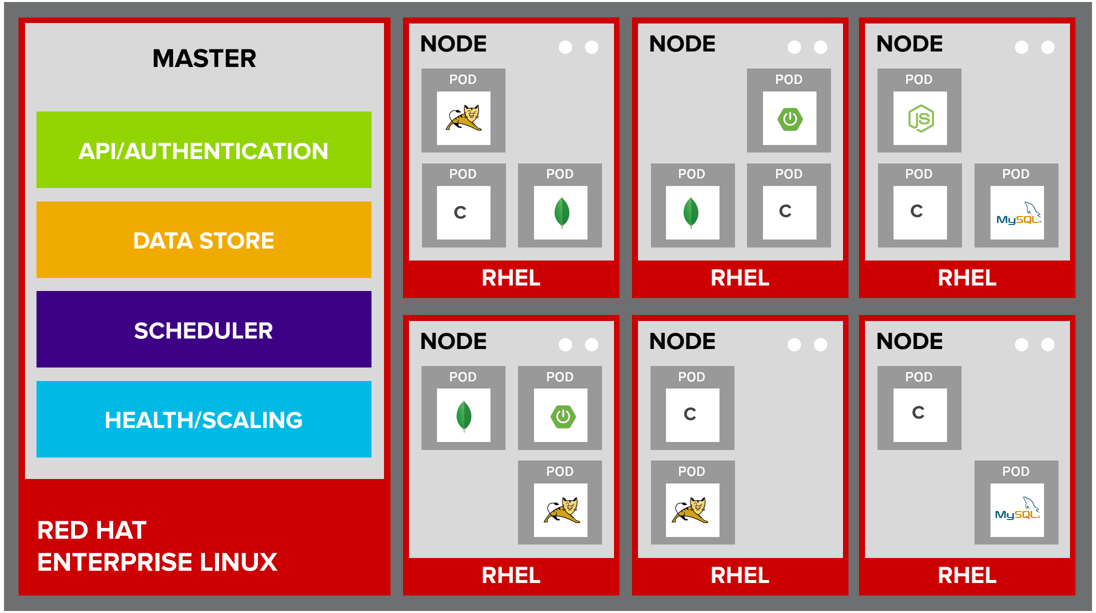
OPENSHIFT SDN - OVS PACKET FLOW

Container Connects to External Host



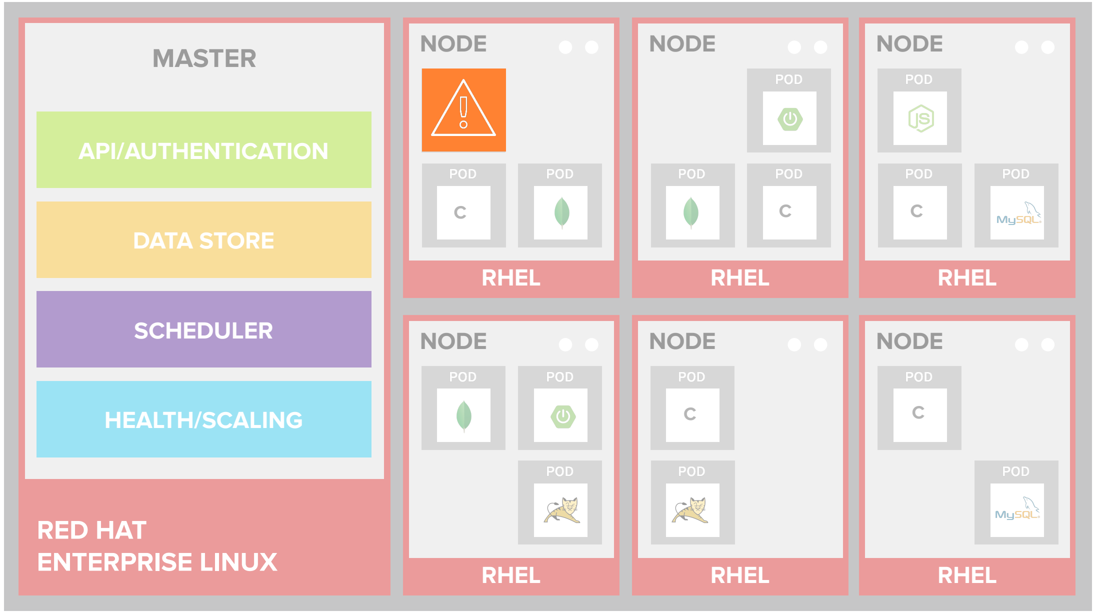
OpenShift Monitoring / Clustering

AUTO-HEALING FAILED PODS

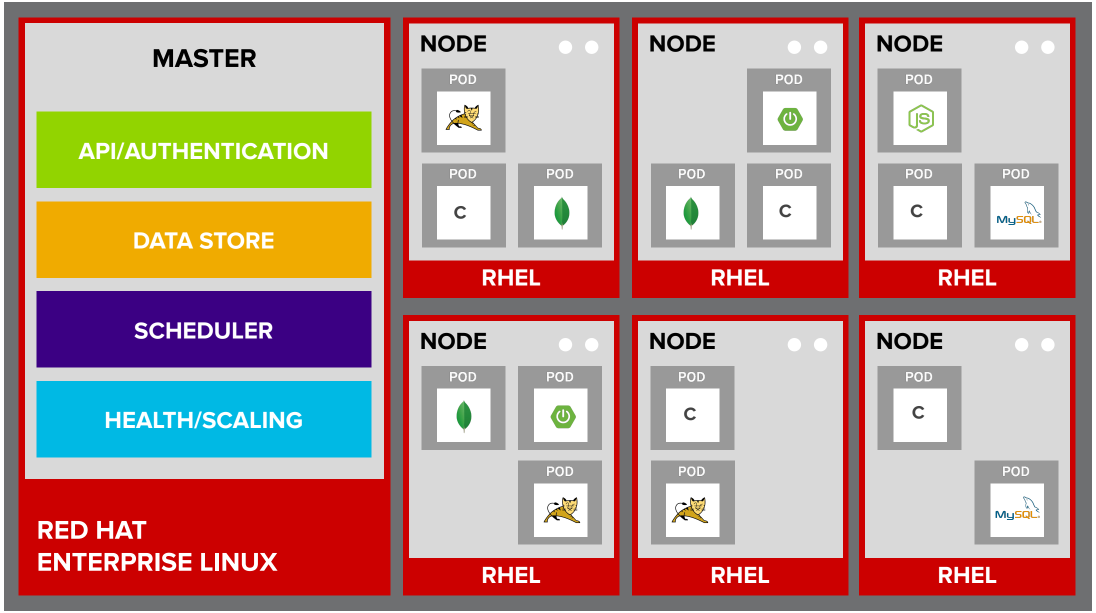


Red Hat

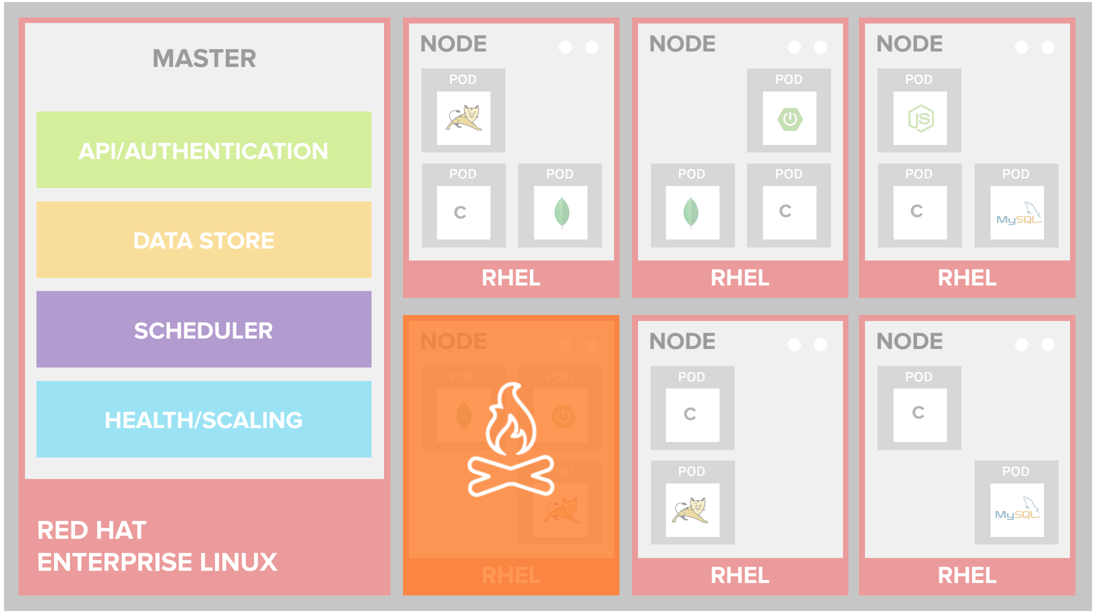
AUTO-HEALING FAILED CONTAINERS



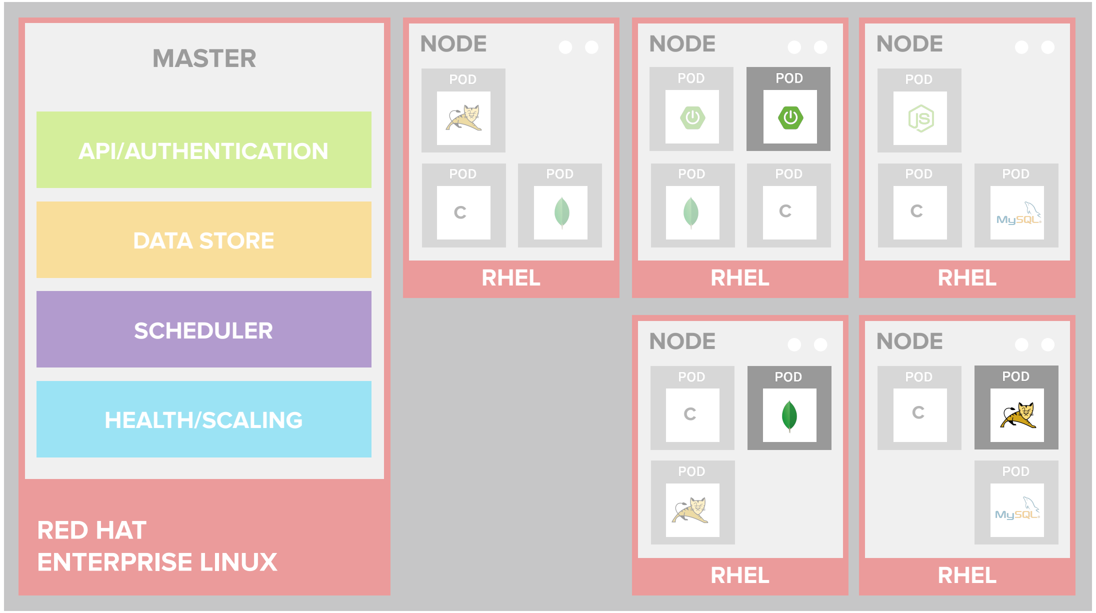
AUTO-HEALING FAILED CONTAINERS



AUTO-HEALING FAILED CONTAINERS



AUTO-HEALING FAILED CONTAINERS



OpenShift persistent Storage

PERSISTENT STORAGE

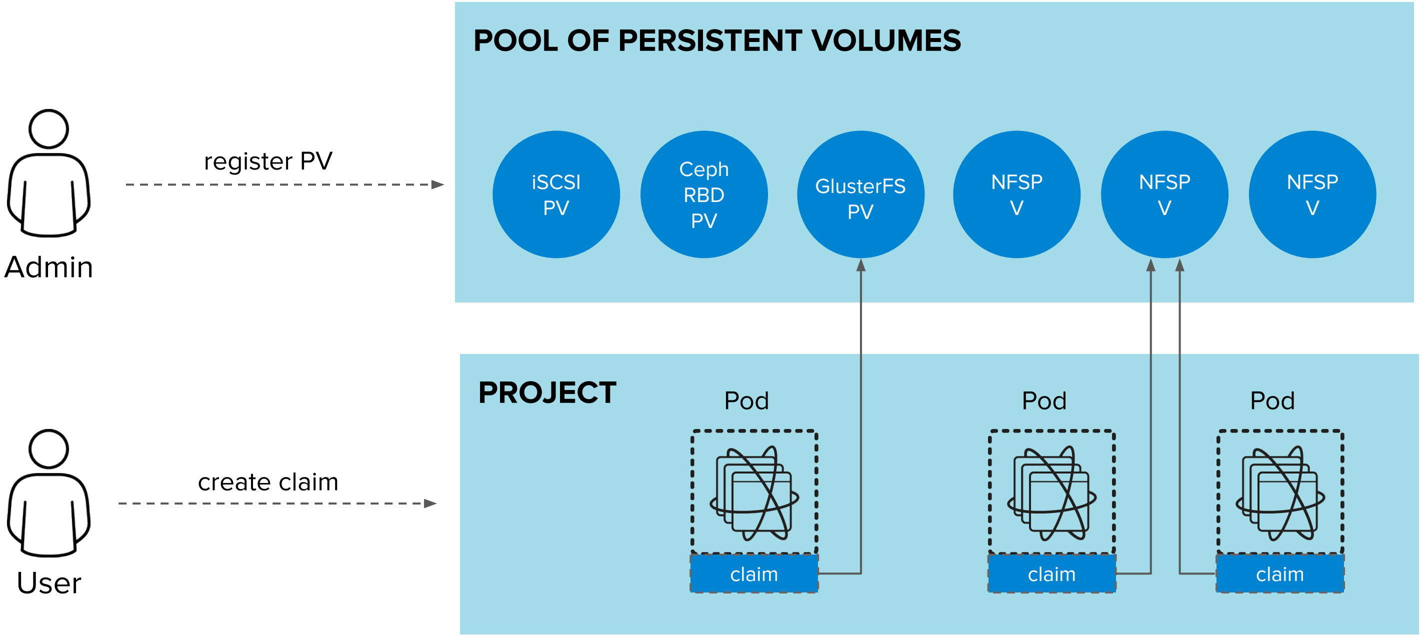
- Persistent Volume (PV) is tied to a piece of network storage
- Provisioned by an administrator (static or dynamically)
- Allows admins to describe storage and users to request storage
- Assigned to pods based on the requested size, access mode, labels and type

NFS	OpenStack Cinder	iSCSI	Azure Disk	AWS EBS	FlexVolume
GlusterFS	Ceph RBD	Fiber Channel	Azure File	GCE Persistent Disk	VMWare vSphere VMDK
		NetApp Trident*	Container Storage Interface (CSI)**		

* Shipped and supported by NetApp via TSANet

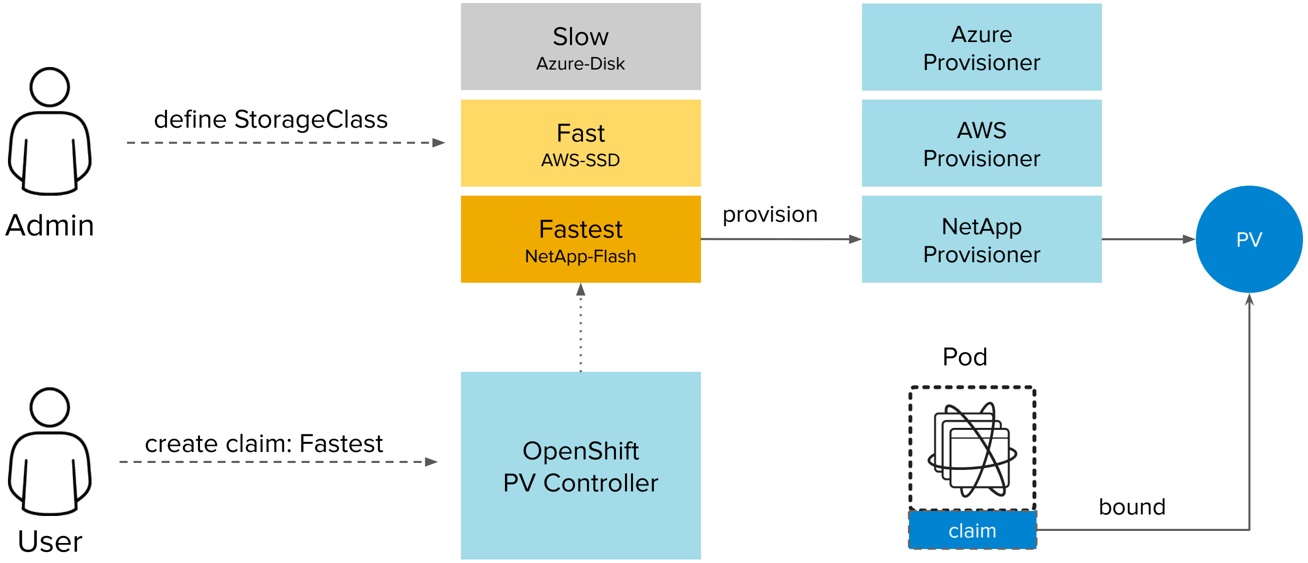
** Tech Preview

PERSISTENT STORAGE



Red Hat

DYNAMIC VOLUME PROVISIONING



Thank you !