

# AWG Bootcamp

## Red Hat OpenShift and IBM Cloud Paks

—

Luca Floris

IBM Cloud Technical Consultant



# Agenda

## **Day 2 (Architecture and Installation)**

- OpenShift 4.3 Architecture
- OpenShift 4.3 Common Cluster Architectures
- OpenShift Container Security
- OpenShift DNS
- OpenShift Authentication
- OpenShift Container Networking
- OpenShift Storage
- OpenShift 4.3 Installation
- OpenShift Day2

# OpenShift 4.3 Architecture

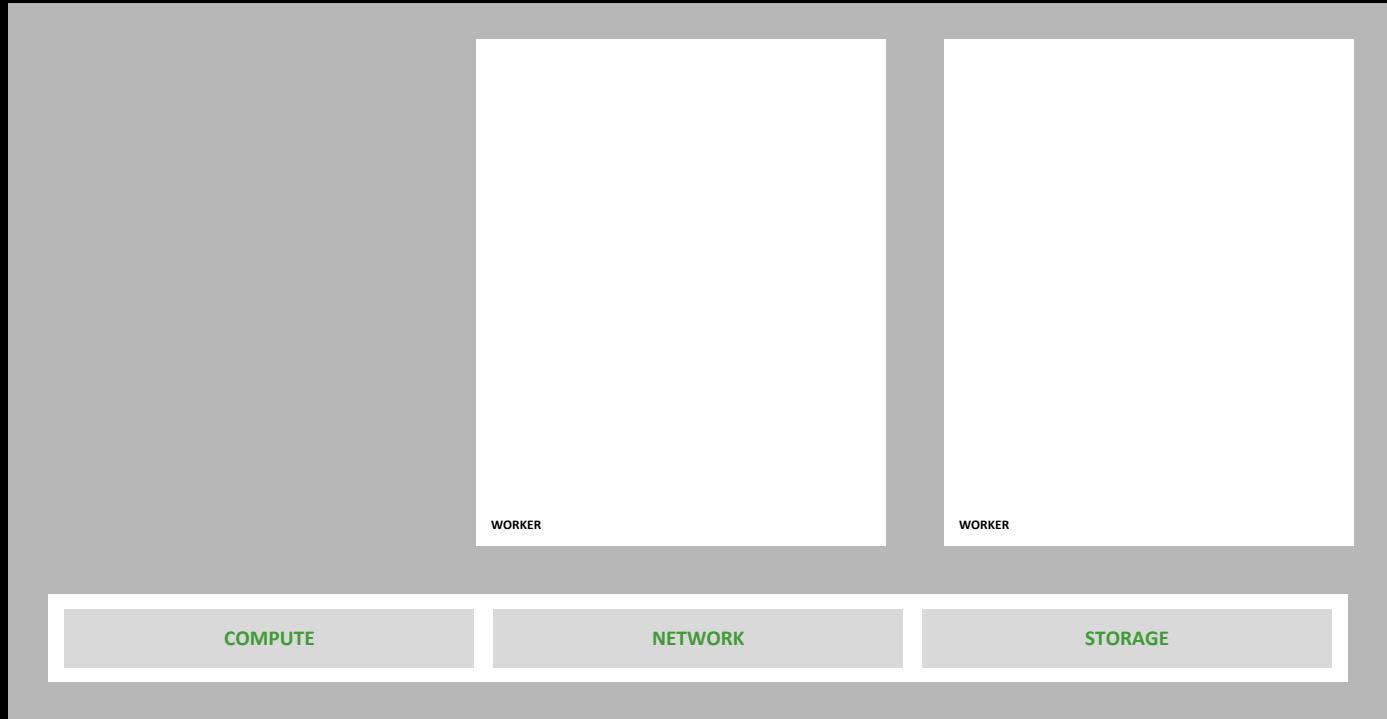
You choose the infrastructure

COMPUTE

NETWORK

STORAGE

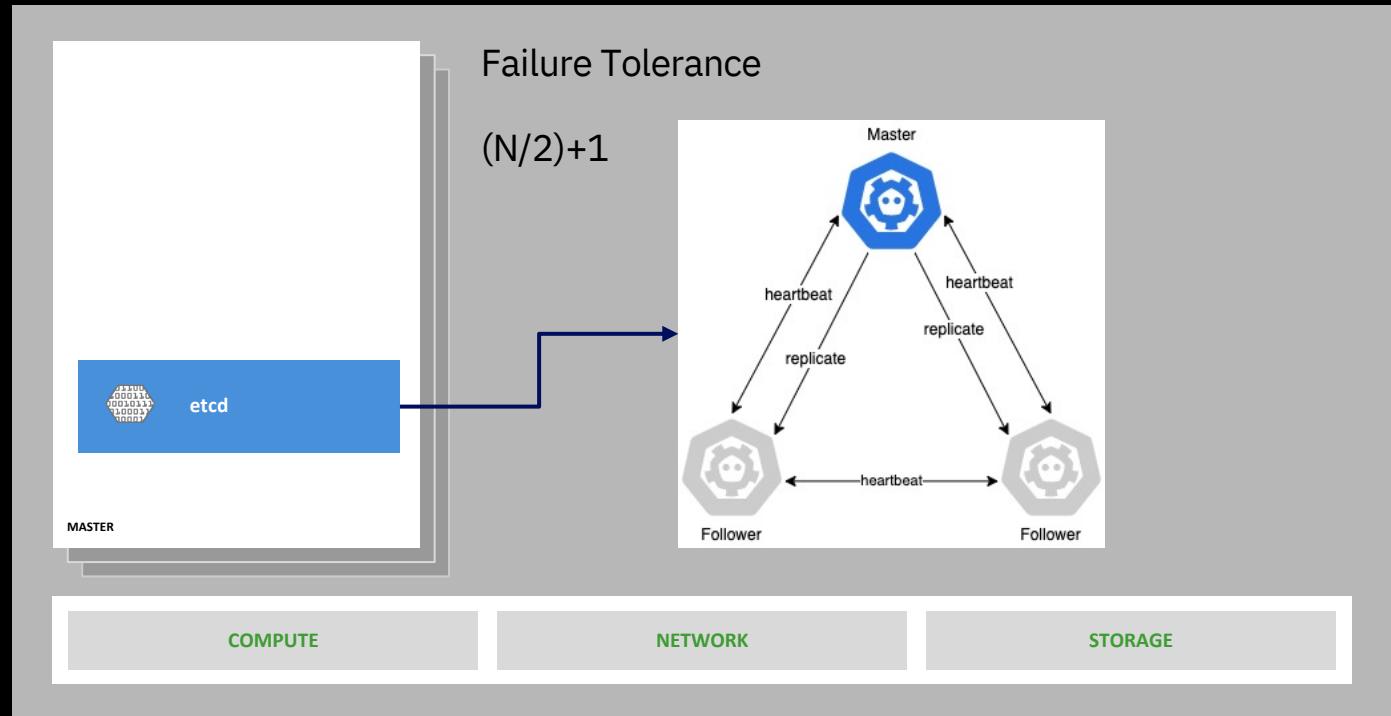
## Workers run workloads



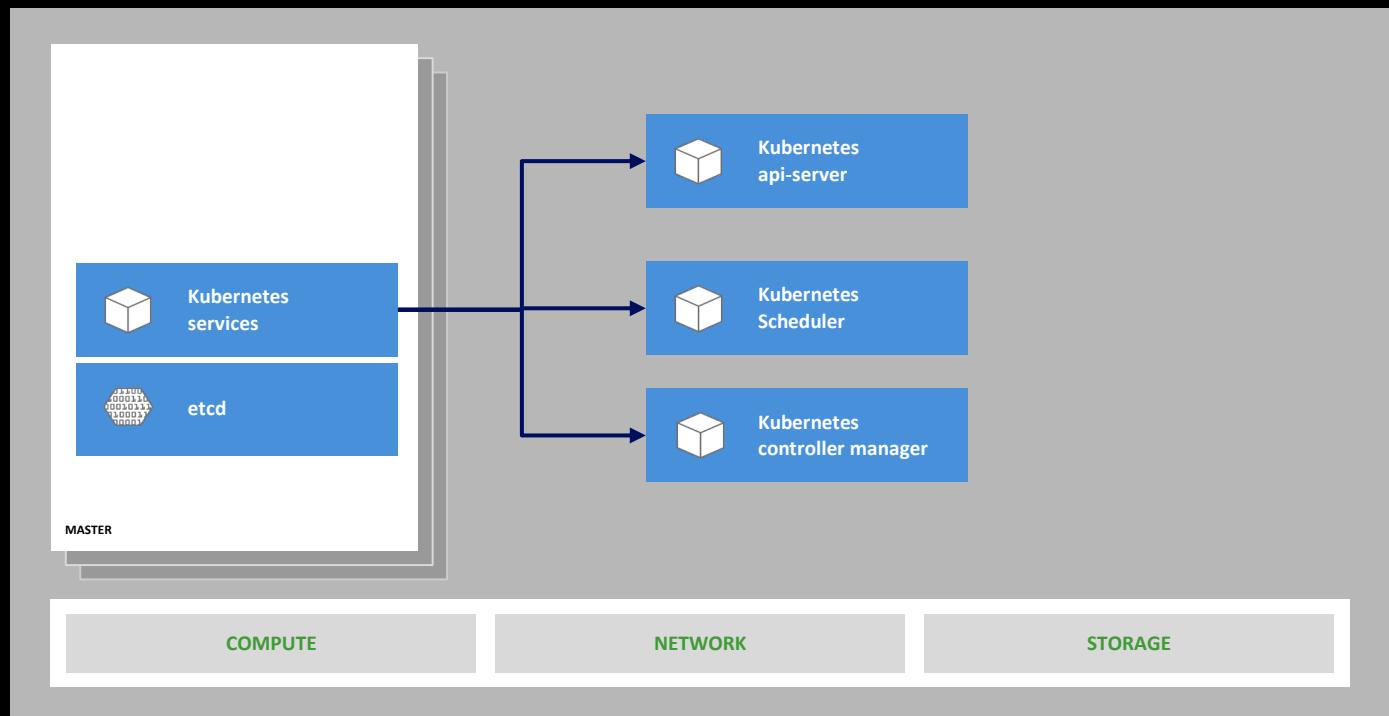
Masters are the control plane



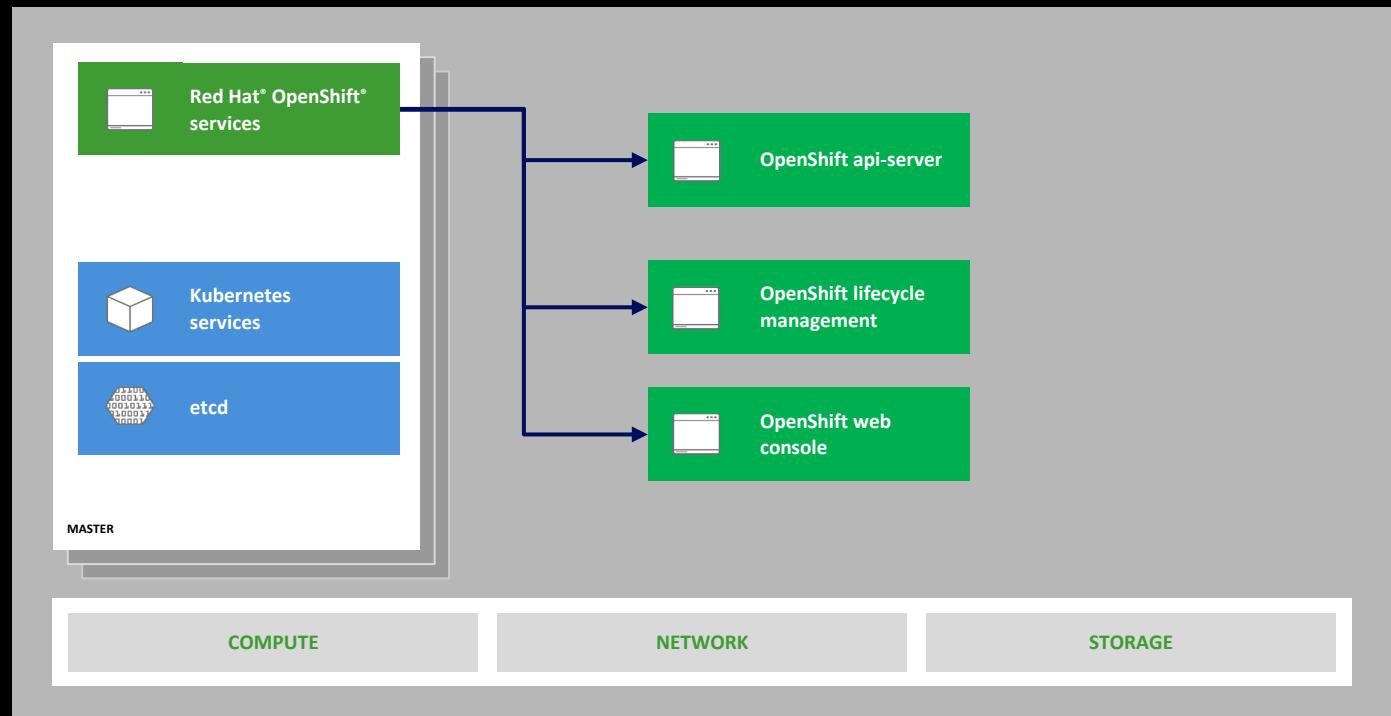
etcd stores the state of everything



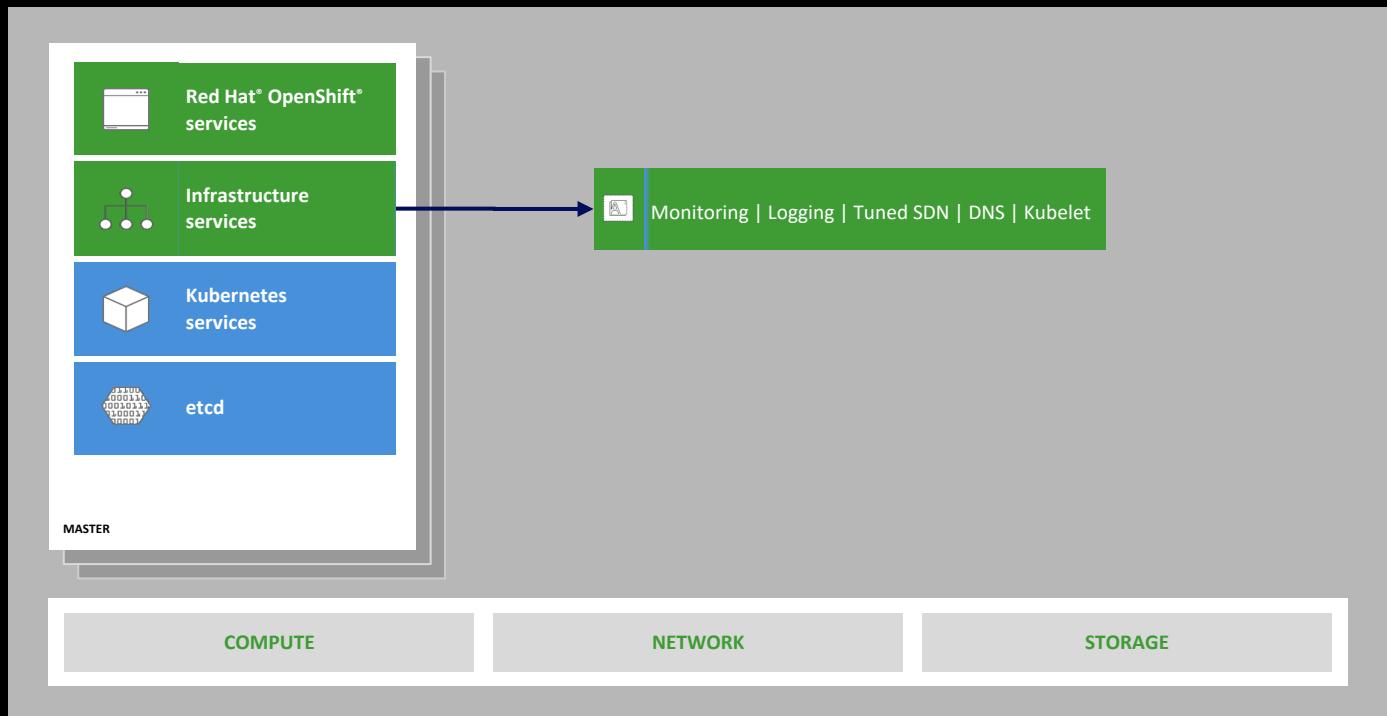
## Core Kubernetes components



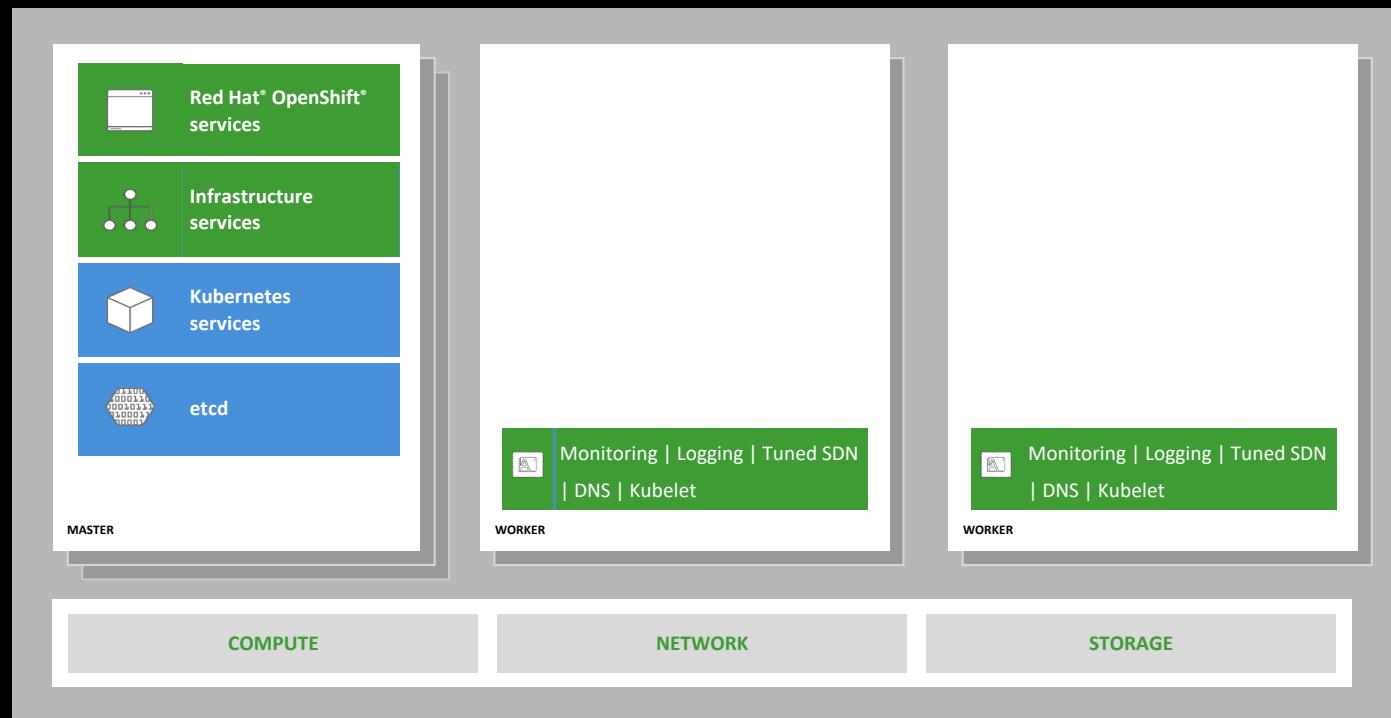
## Core OpenShift components



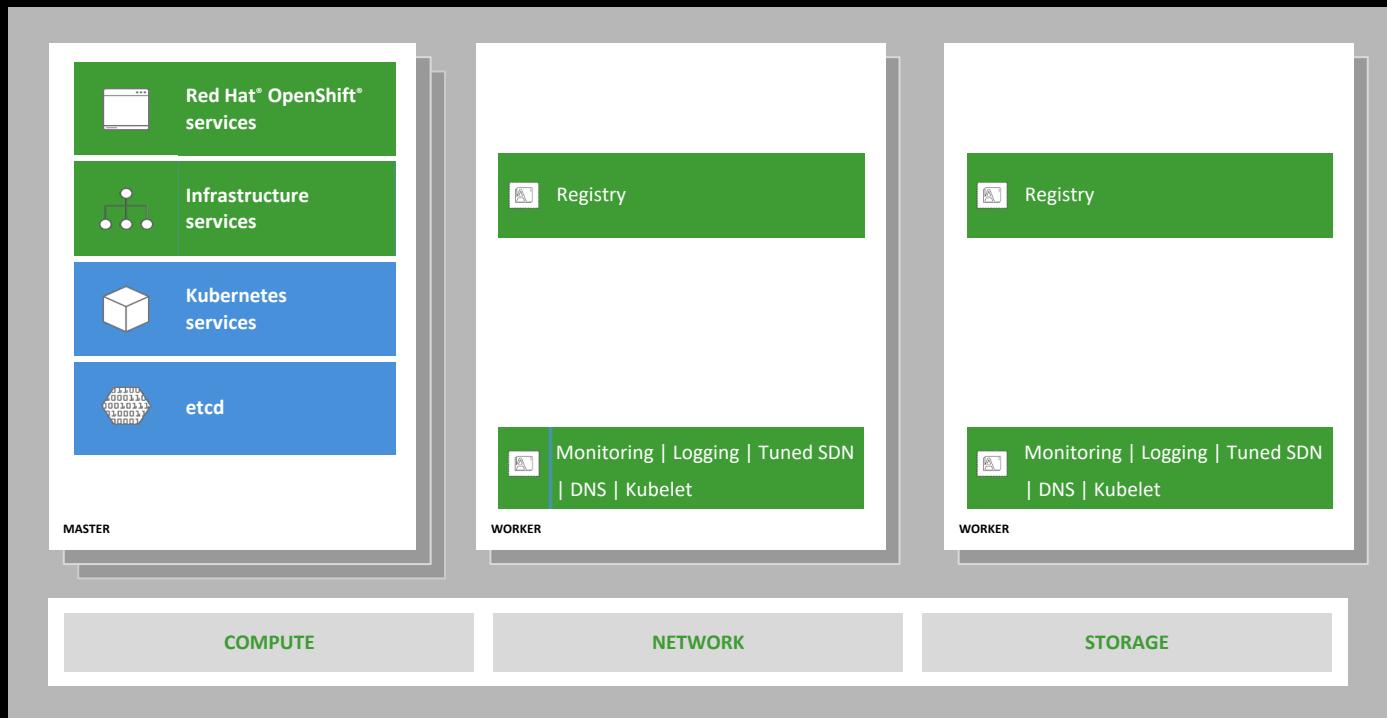
## Internal and support infrastructure services



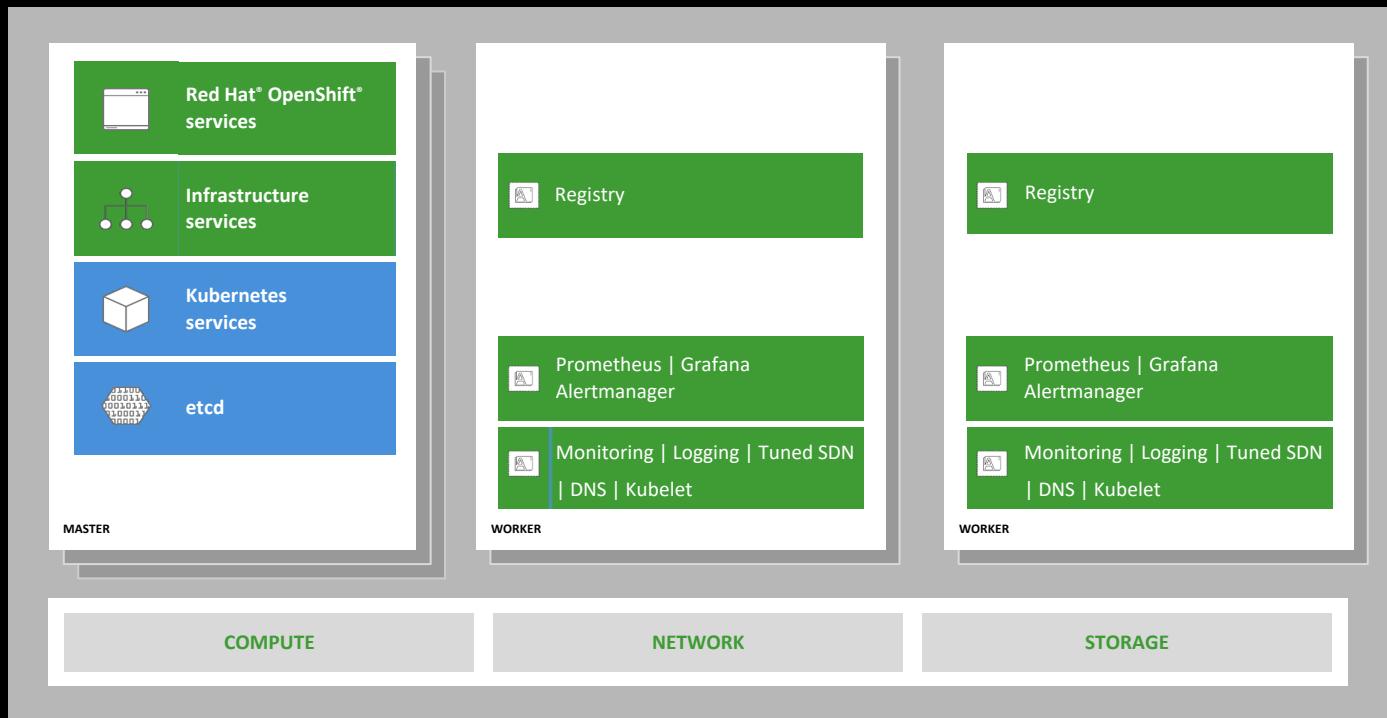
Runs on all hosts



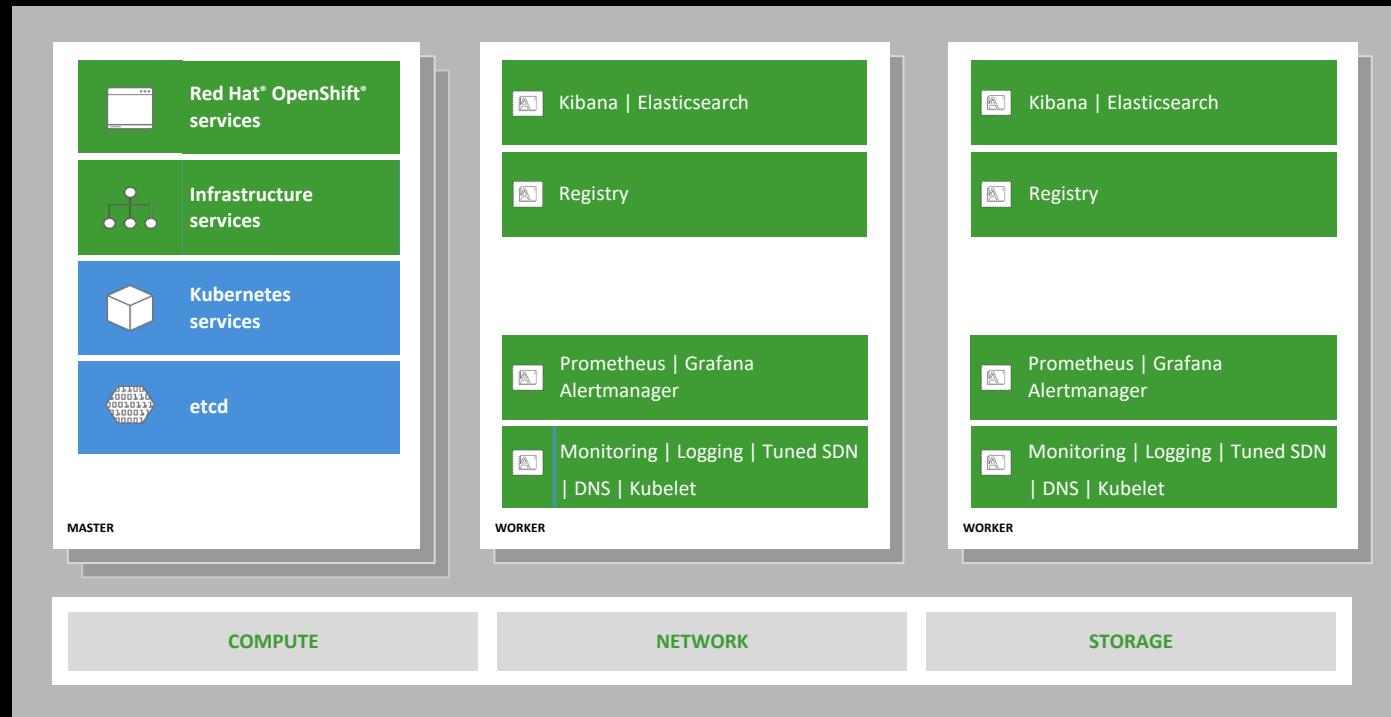
# Integrated image registry



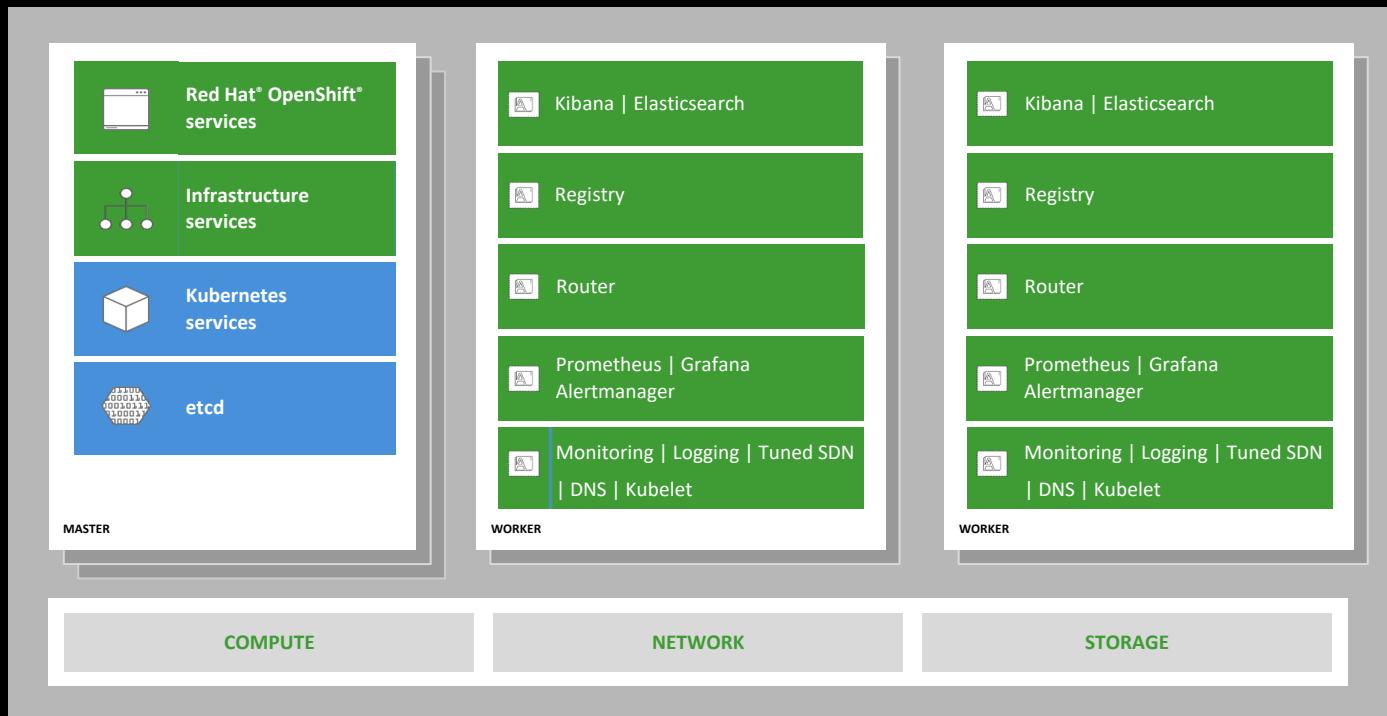
# Cluster Monitoring



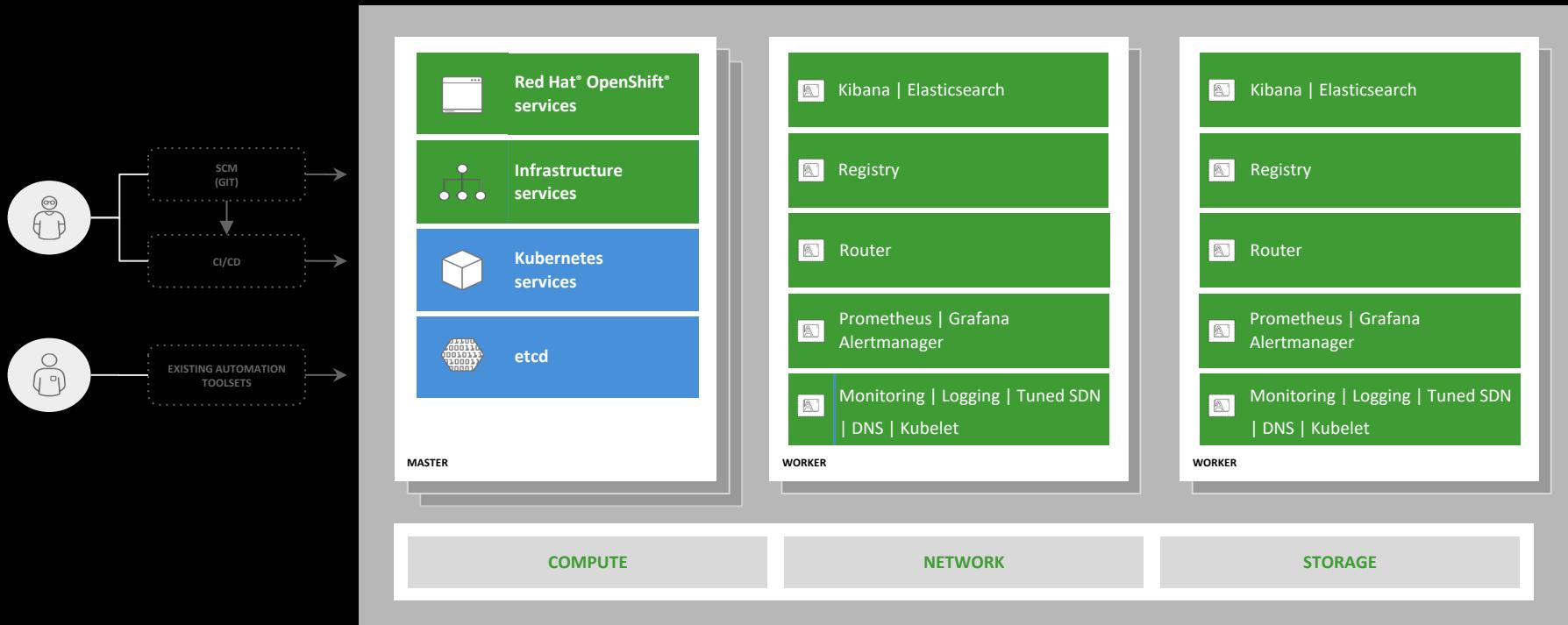
## Log aggregation



# Integrated routing



## Dev and Ops via Web, CLI, API and IDE



# Red Hat CoreOS

**RED HAT<sup>®</sup>**  
ENTERPRISE LINUX<sup>®</sup>

**RED HAT<sup>®</sup>**  
ENTERPRISE LINUX CoreOS

## BENEFITS

- 10+ year enterprise life cycle
- Industry standard security
- High performance on any infrastructure
- Customizable and compatible with wide ecosystem of partner solutions
- Self-managing, over-the-air updates
- Immutable and tightly integrated with OpenShift
- Host isolation is enforced via Containers
- Optimized performance on popular infrastructure

## WHEN TO USE

When customization and integration with additional solutions is required

When cloud-native, hands-free operations are a top priority

# 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.

## RHEL CoreOS admins are responsible for:

Nothing. 😊 🙌

## 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



v4.3.5



v4.3.5

# Common Architectures

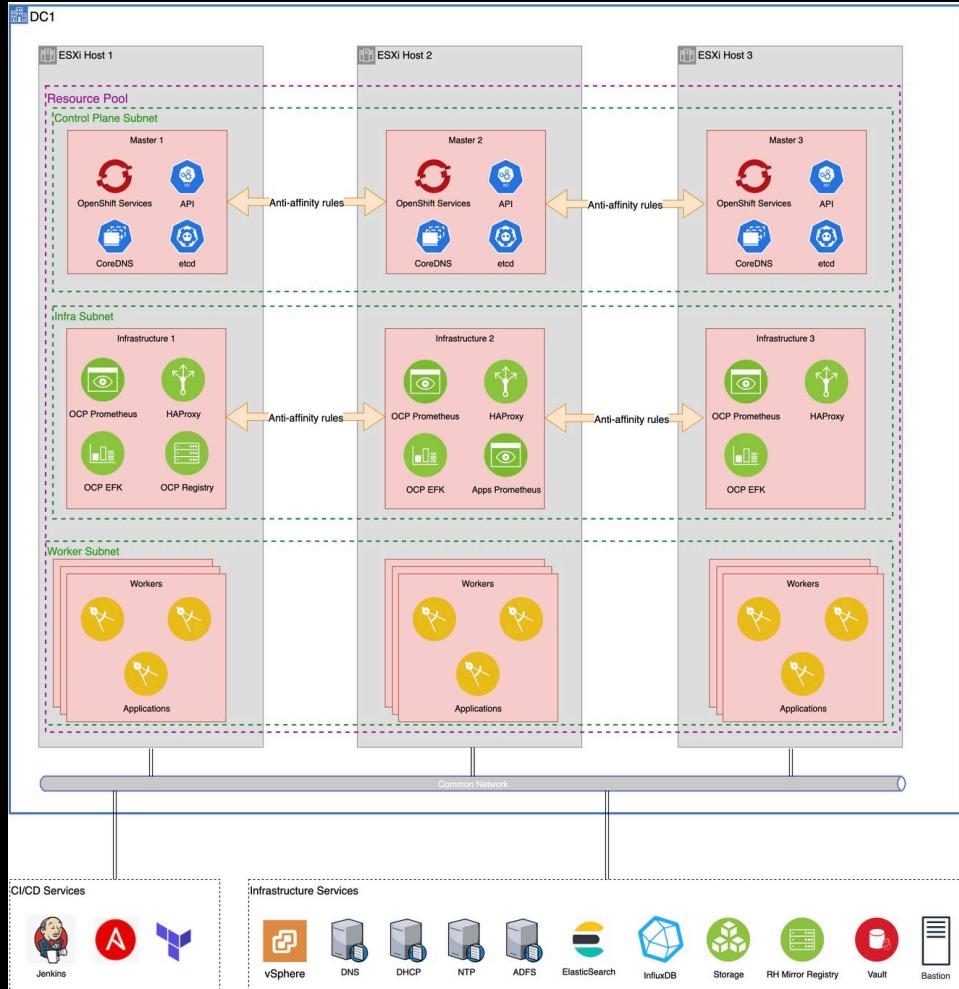
# Common Architectures

## Single cluster, single datacenter

A single cluster is distributed within one datacenter

Control plane is distributed across ESXi hosts/availability zones

Implements HA/DR measures to protect against host failure



# Common Architectures

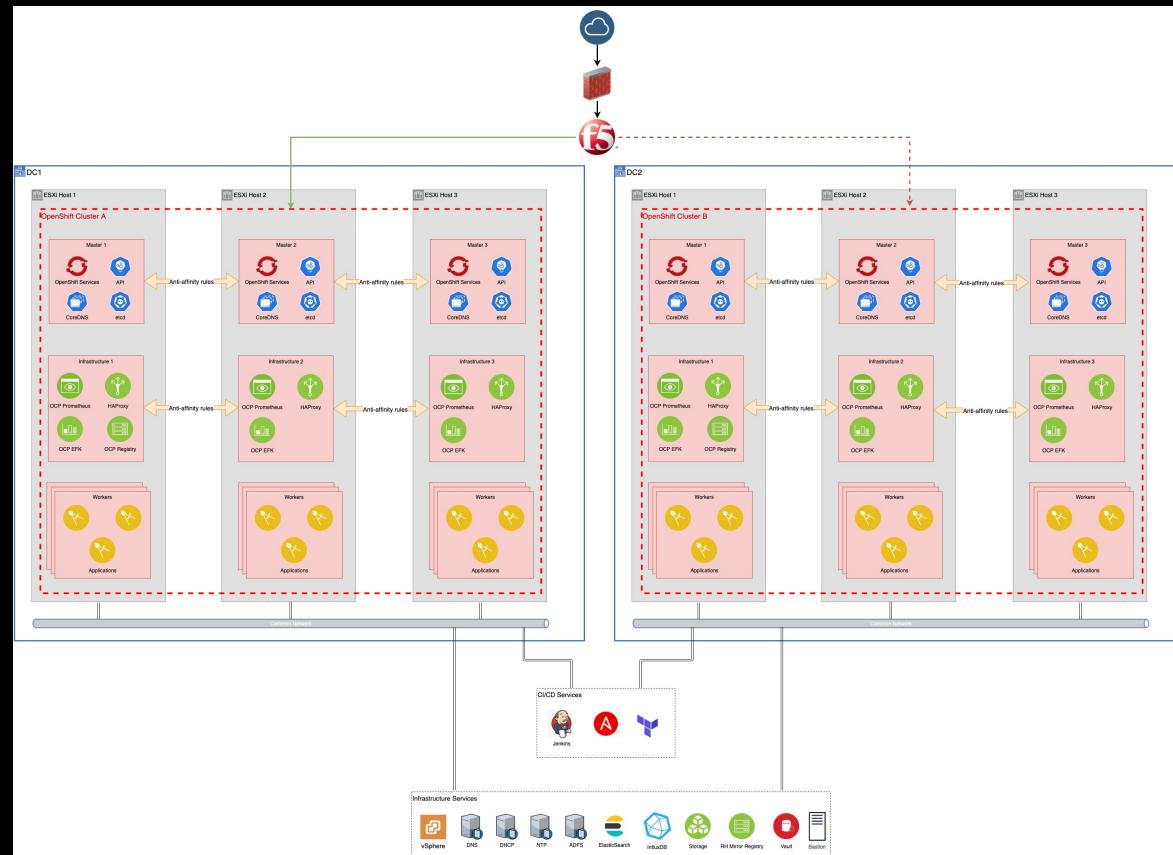
## Two clusters, multiple datacentres

Multiple clusters are deployed to different datacenters in an active/active or active/passive approach

Traffic is routed to the active cluster via a global load balancer

Implements the same principles as the single cluster, single datacenter strategy

Implements HA/DR measures to protect against site failure



# Common Architectures

## Single cluster, multiple datacentres (stretched)

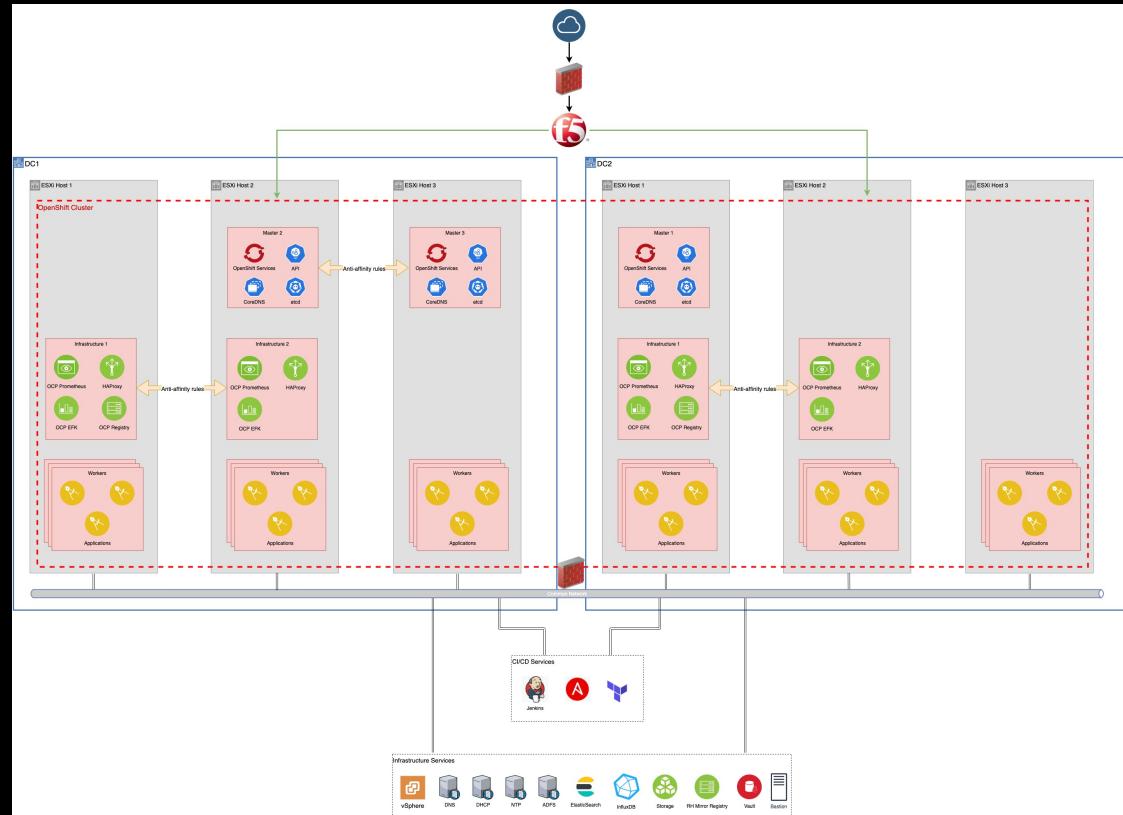
Single cluster deployed across two datacenters with stretched networking approach

Requires low latency

Intra-datacenter implements the same principles as the single cluster, single datacenter strategy

Implements HA/DR measures to protect against site failure

Datacenter hosting both masters is a weak point



# Some Considerations for Designing OpenShift Clusters

## Workload and Application considerations

What availability is needed for the application?

RPO and RTO for the application?

What are your workload requirements?

## Platform Considerations

Are we understanding the capacity needs? Do we need N+1 or N+2?

What tooling is foreseen to replicate the data between prod and DR?

What are the external or both? nal integrations required?  
Logging, monitoring, storage?

## Container considerations

How is the traffic sprayed over the HA components?

Do we allow DR for individual applications or only for the cluster?

## Infrastructure considerations

How is the HA foreseen for the infrastructure?

Do we have a dark DR site or an active one?

Does it need to scale?

Single cluster, or multiple clusters?

## Organization considerations

Who is managing and monitoring the platform for availability?

Do we apply regular switches between prod and DR?

What are the High Availability and Disaster Recovery strategies?

# Some Considerations for Designing OpenShift Clusters

## Workload Type

- For heritage workloads, some customers may have separate VMs for each application instance (AI), where the # of AIs / VMs will equal the number of pods.
- In these scenarios, be sure to ask about VM specs (cores per VM)
- For microservices workloads be sure to understand the dynamic behaviour

## Platform Type

- When evaluating how many servers or VMs will be in use, be sure to confirm # of cores per machine.
- Understand the needs of your workload to determine the type of worker nodes - elasticsearch needs large nodes with high memory
- Also verify whether your customer expects this environment to grow within the next years.

## Cluster Requirements

- Look at the isolation and segmentation as this can increase the node numbers
- You should also be sure to verify with your customer whether they need HA and DR
- Check whether the current environment size already accounts for redundancy / HA.
- How many environments are needed and how are they placed over the different DCs.

# OpenShift Security

# OpenShift Security



## 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

# OpenShift Security

**To make the correct security design decisions we need to understand:**



The type of workloads the client is going to run

The type of external interactions we need for the applications

The way of interacting with the platform and which interfaces they are going to use.

The different integrations for IAM and other security components

The isolation and segmentation is needed for the different groups/teams

The data protection needs for the different workloads

How containers need to be secured, monitored for mutations

Etc.

# OpenShift Security

## - Certificates and Certificate Management

OpenShift provides its own internal CA

Certificates are used to provide secure connections to

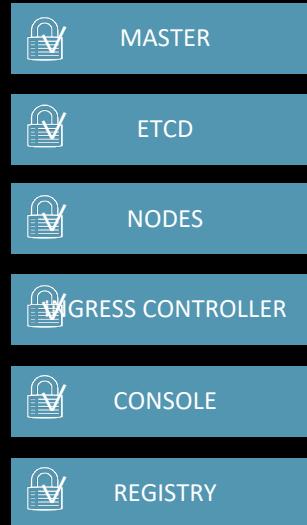
master (APIs) and nodes

Ingress controller and registry

etcd

Certificate rotation is automated

Optionally configure external endpoints to use custom certificates



# OpenShift Security

## - Role Based Access Control (RBAC)

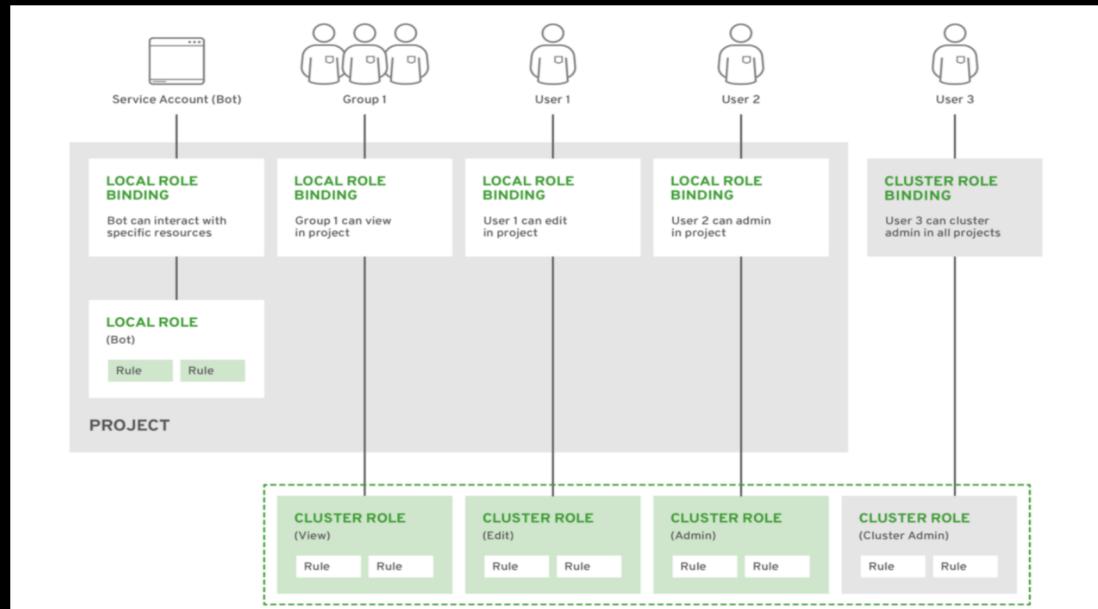
Project scope & cluster scope available

Matches request attributes (verb,object,etc)

If no roles match, request is denied ( deny by default )

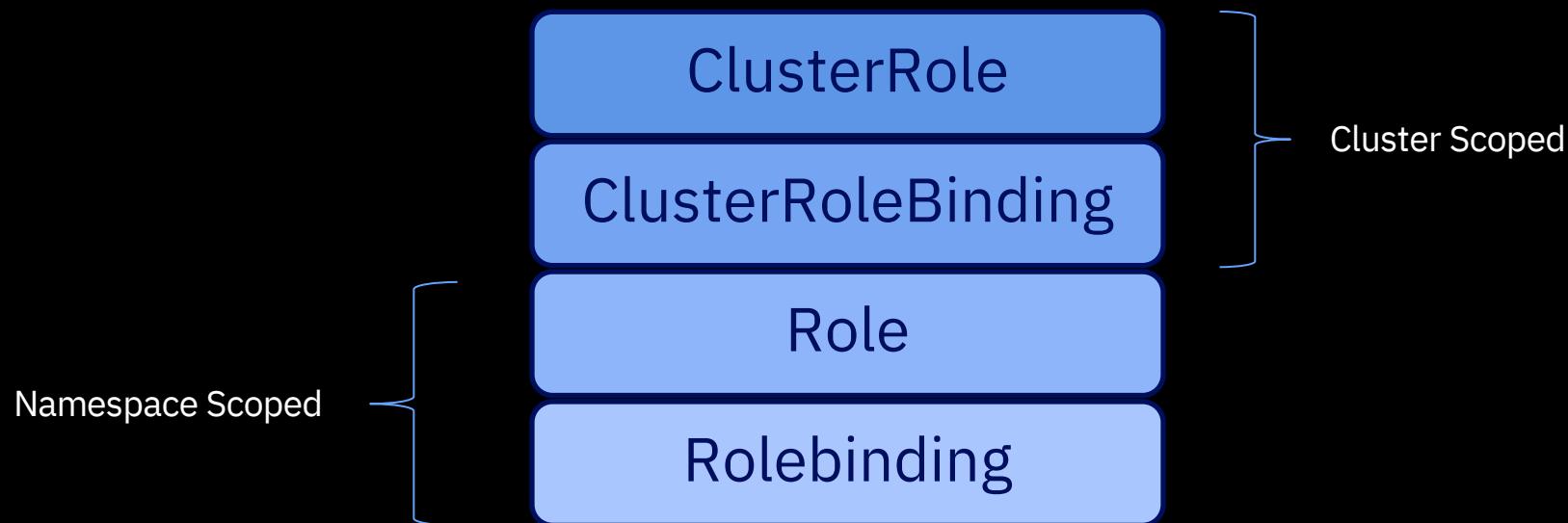
Operator- and user-level roles are defined by default

Custom roles are supported



# OpenShift Security

## - Role Based Access Control (RBAC)



# OpenShift Security

## - Role Based Access Control (RBAC)

Default Cluster Role	Description
admin	A project manager. If used in a local binding, an admin has rights to view any resource in the project and modify any resource in the project except for quota.
basic-user	A user that can get basic information about projects and users.
cluster-admin	A super-user that can perform any action in any project. When bound to a user with a local binding, they have full control over quota and every action on every resource in the project.
cluster-status	A user that can get basic cluster status information.
edit	A user that can modify most objects in a project but does not have the power to view or modify roles or bindings.
self-provisioner	A user that can create their own projects.
view	A user who cannot make any modifications, but can see most objects in a project. They cannot view or modify roles or bindings.

# OpenShift Security

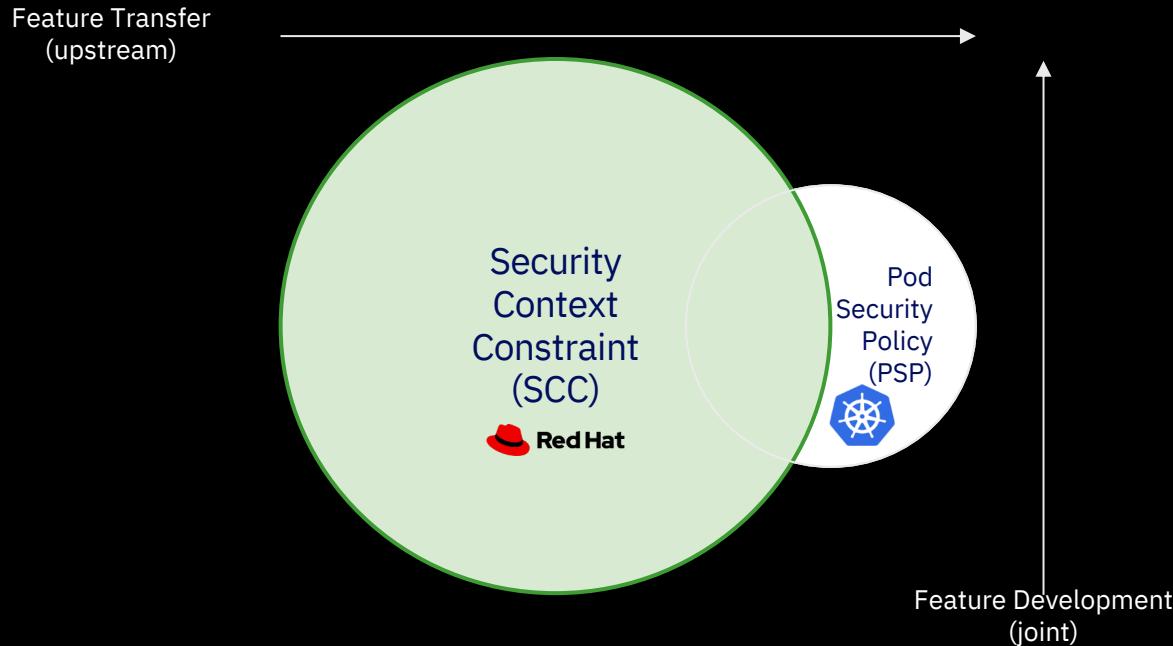
## - Security Context Constraints

SCCs allow an administrator to control:

- Whether a pod can run privileged containers.
- The capabilities that a container can request.
- The use of host directories as volumes.
- The SELinux context of the container.
- The container user ID.
- The use of host namespaces and networking.
- The allocation of an FSGroup that owns the pod's volumes.
- The configuration of allowable supplemental groups.
- Whether a container requires the use of a read only root file system.
- The usage of volume types.
- The configuration of allowable seccomp profiles.

# OpenShift Security

## - Security Context Constraints



# OpenShift Security

## - etcd Encryption

When you enable etcd encryption, encryption keys are created.

These keys are rotated on a weekly basis. You **must** have these keys in order to restore from an etcd backup.

By default, etcd data is not encrypted in OpenShift Container Platform.

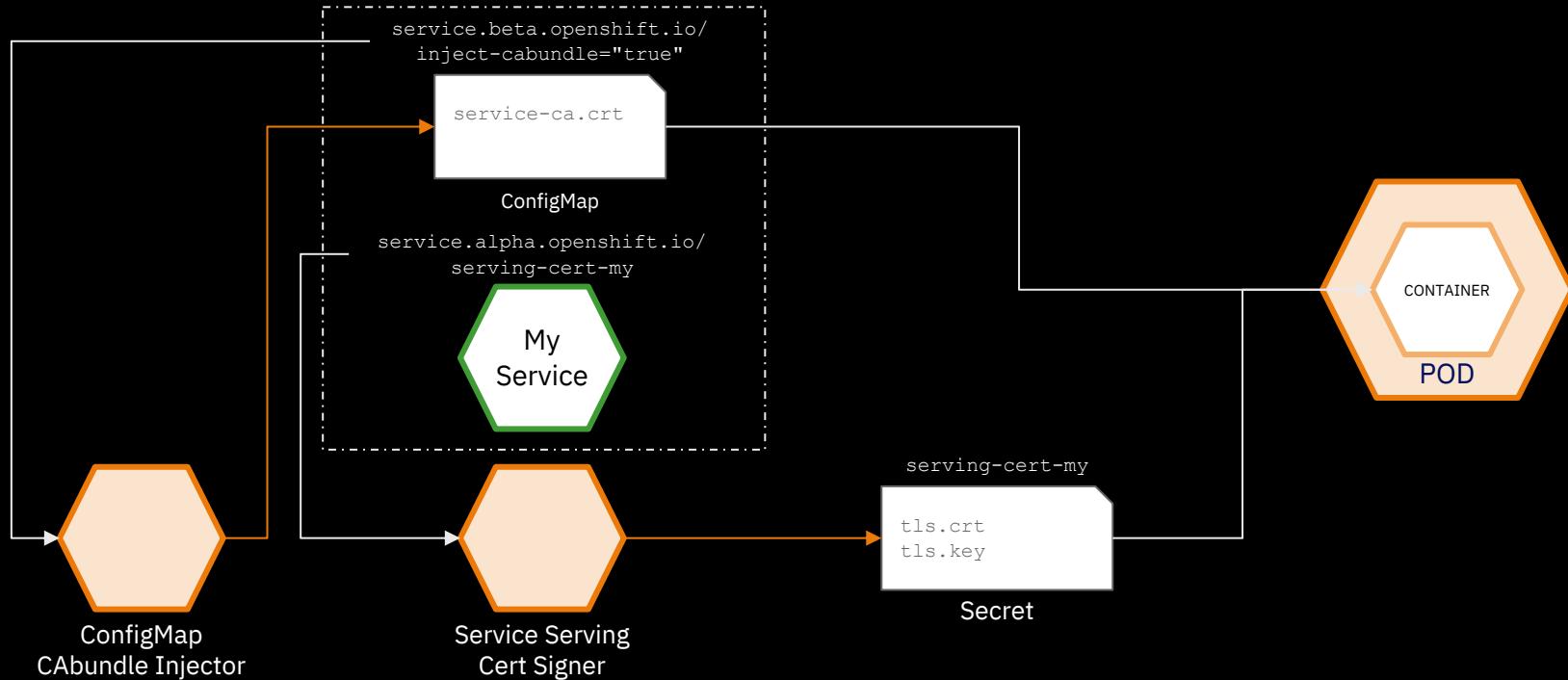
When you enable etcd encryption, the following OpenShift API server and Kubernetes API server resources are encrypted:

- Secrets
- ConfigMaps
- Routes
- OAuth access tokens
- OAuth authentication tokens



# OpenShift Security

## - Service Certificates





# FIPS 140-2 SECURITY CERTIFICATION



PRESS RELEASE

## Red Hat Continues Drive for More Secure Enterprise IT, Re-Certifies Red Hat Enterprise Linux for FIPS 140-2

Red Hat Enterprise Linux 7.5 and many layered technologies meet stringent software security criteria for sensitive public sector deployments

<https://www.redhat.com/en/about/press-releases/red-hat-continues-drive-more-secure-enterprise-it-re-certifies-red-hat-enterprise-linux-fips-140-2>

Validates Secure Design & Implementation of cryptographic modules

- Periodic re-certifications
- Removed SSL3.0
- Disabled protocols with known issues (TLS1.0, TLS1.1)
- Require secure key sizes for existing crypto systems (*RSA, DH: 2048+ bit keys*)
- Provide only secure protocols (TLS1.2, TLS1.3, SSH2, IKEv2)

# OpenShift DNS

Built-in internal DNS to reach services by name

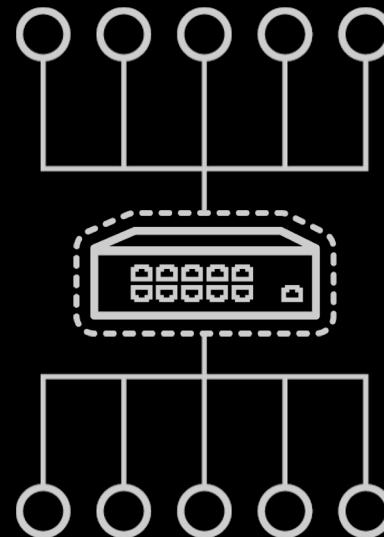
Split DNS is supported via DNSmasq

- Master answers DNS queries for internal services
- Other name servers serve the rest of the queries

You can use DNS forwarding to override the forwarding configuration identified in `etc/resolv.conf` on a per-zone basis by specifying which name server should be used for a given zone.

`my-svc.my-namespace.svc.cluster.local`

The DNS Operator deploys and manages CoreDNS to provide a name resolution service to pods, enabling DNS-based Kubernetes Service discovery in OpenShift.



# OpenShift Security

## - Network Security

A network policy is a specification of how groups of pods are allowed to communicate with each other and other network endpoints.

A default installation of OpenShift allows all pods to communicate with all other pods in the system - there is no pre-defined network segregation unless you select the multitenant mode

```
kind: NetworkPolicy
apiVersion: networking.k8s.io/v1
metadata:
  name: front-allow
  namespace: development
spec:
  podSelector:
    matchLabels:
      role: backend
      app: my-app
  policyTypes:
  - Ingress
  ingress:
  - from:
    - podSelector:
        matchLabels:
          role: webfront
          app: my-app
    ports:
    - protocol: TCP
      port: 3306
```

# OpenShift Security

## - Network Security

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
        - ipBlock:
            cidr: 172.17.0.0/16
          except:
            - 172.17.1.0/24
        - namespaceSelector:
            matchLabels:
              project: myproject
        - podSelector:
            matchLabels:
              role: frontend
  ports:
    - protocol: TCP
      port: 6379
  egress:
    - to:
        - ipBlock:
            cidr: 10.0.0.0/24
  ports:
    - protocol: TCP
      port: 5978
```

**podSelector:** Each NetworkPolicy includes a podSelector which selects the grouping of pods to which the policy applies. The example policy selects pods with the label “role=db”. An empty podSelector selects all pods in the namespace.

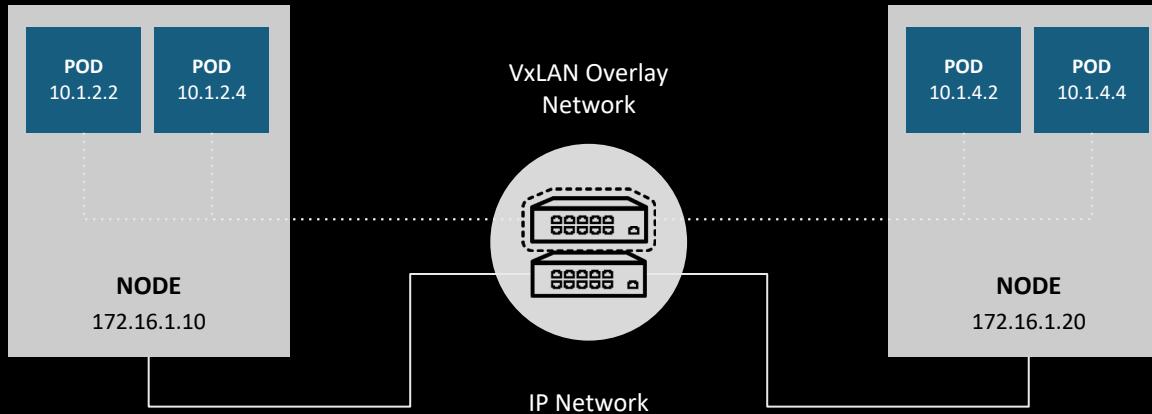
**policyTypes:** Each NetworkPolicy includes a policyTypes list which may include either Ingress, Egress, or both. The policyTypes field indicates whether or not the given policy applies to ingress traffic to selected pod, egress traffic from selected pods, or both. If no policyTypes are specified on a NetworkPolicy then by default Ingress will always be set and Egress will be set if the NetworkPolicy has any egress rules.

**ingress:** Each NetworkPolicy may include a list of whitelist ingress rules. Each rule allows traffic which matches both the from and ports sections. The example policy contains a single rule, which matches traffic on a single port, from one of three sources, the first specified via an ipBlock, the second via a namespaceSelector and the third via a podSelector.

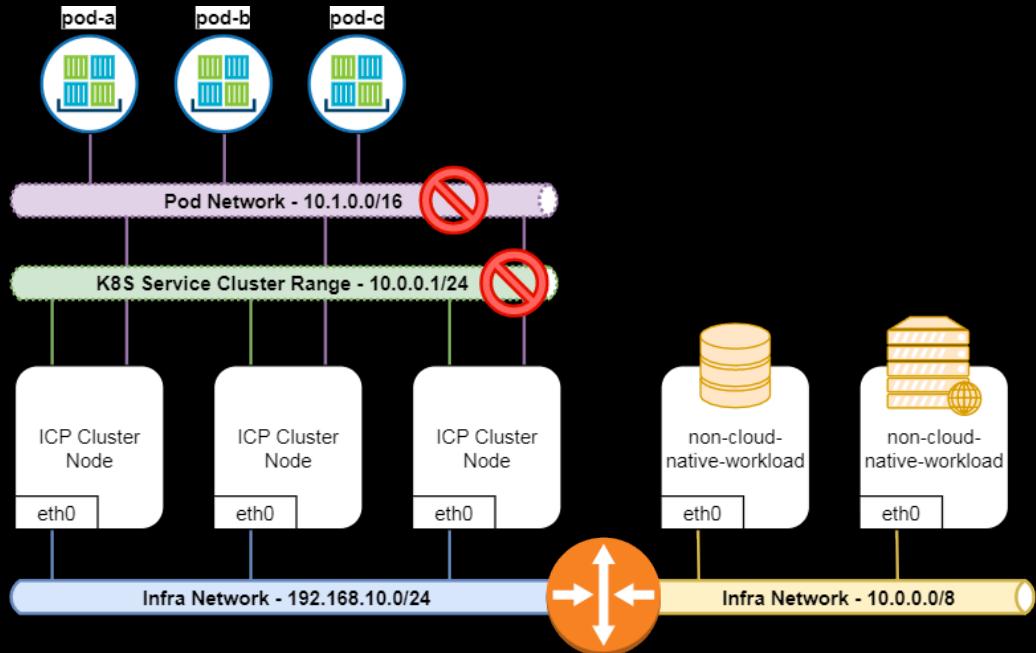
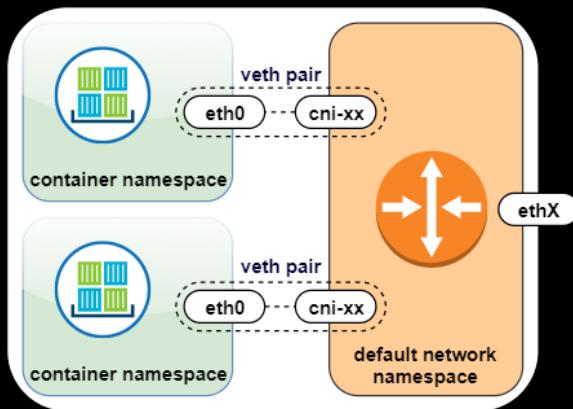
**egress:** Each NetworkPolicy may include a list of whitelist egress rules. Each rule allows traffic which matches both the to and ports sections. The example policy contains a single rule, which matches traffic on a single port to any destination in 10.0.0.0/24.

# OpenShift Networking

# OpenShift Networking



# OpenShift Networking



# OpenShift Networking

## - OpenShift SDN

OpenShift uses software-defined-networking approach for a unified cluster network

Enables pod-to-pod communication

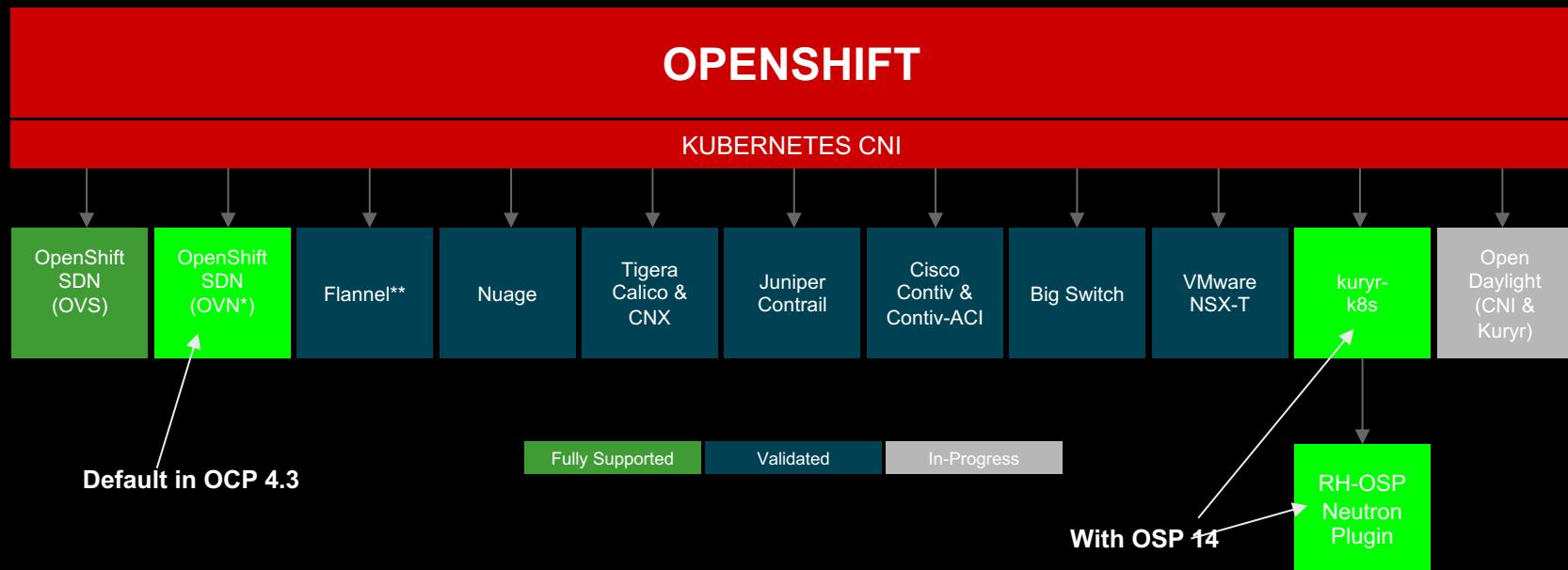
OpenShift SDN is the default

3 modes:

- NetworkPolicy
- Multitenant
- Subnet

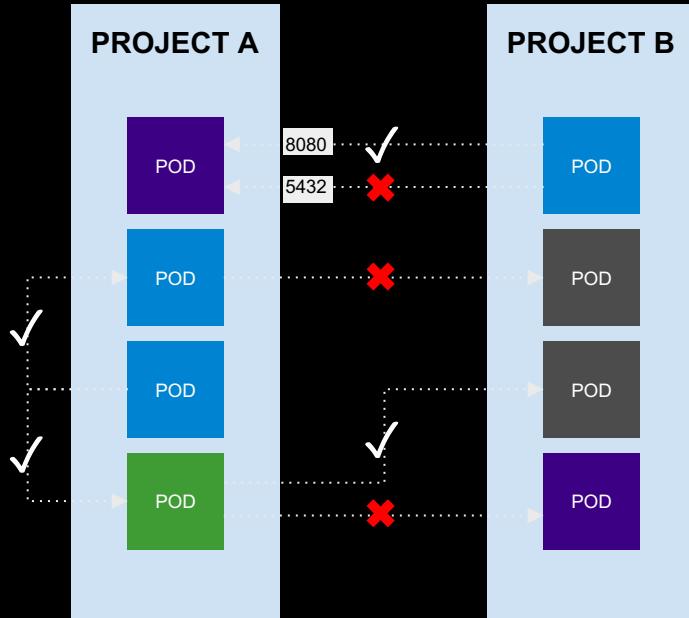
Governed by the Cluster Network Operator (CNO)

# OpenShift Networking - Network Plugins



# OpenShift Networking

## - Network Policy



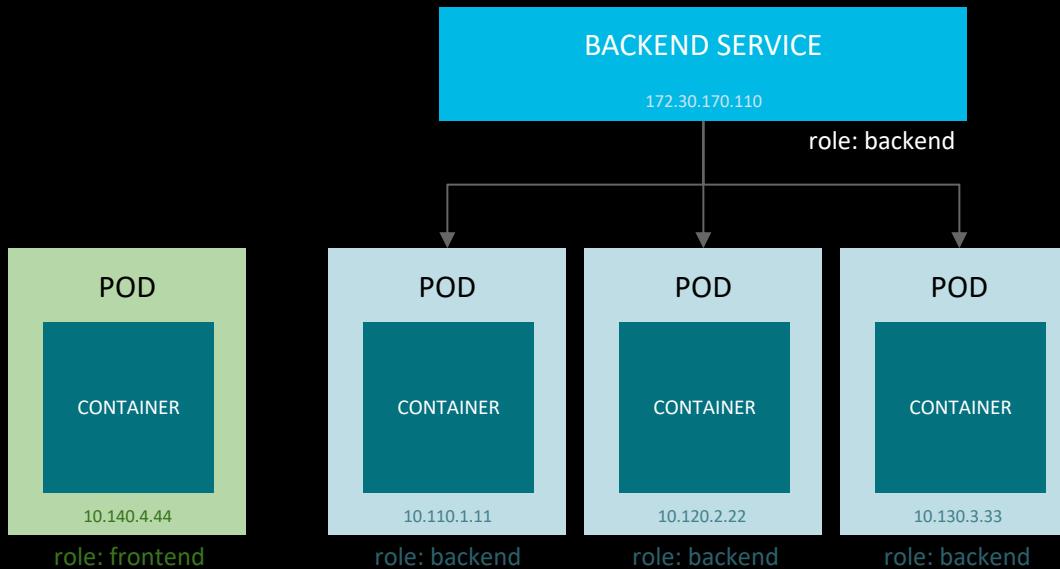
### Example Policies

- 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
```

# OpenShift Networking - Services

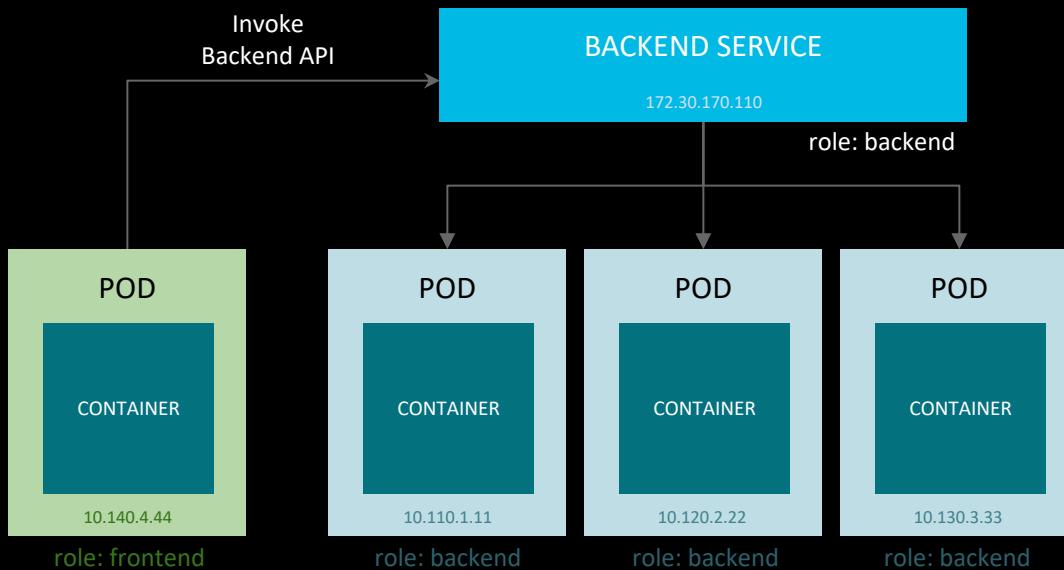
services provide internal load-balancing and service discovery across pods



# OpenShift Networking - Services

47

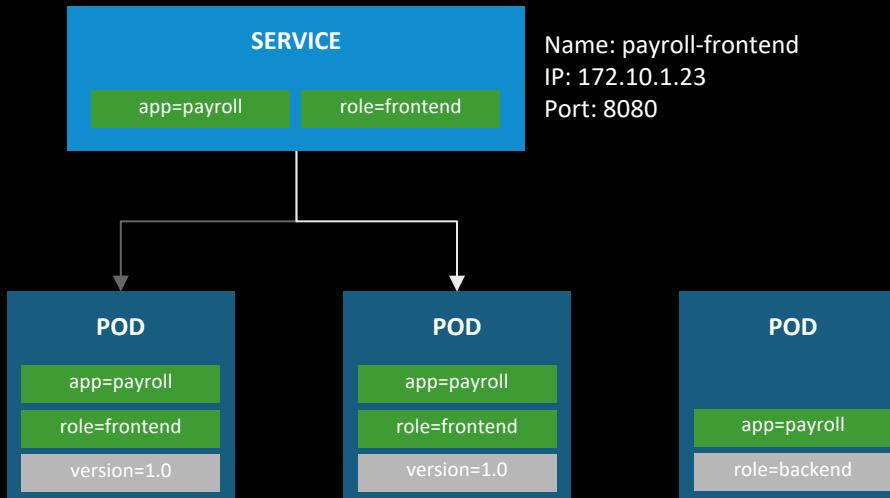
apps can talk to each other via services



# OpenShift Networking

## - Service Discovery

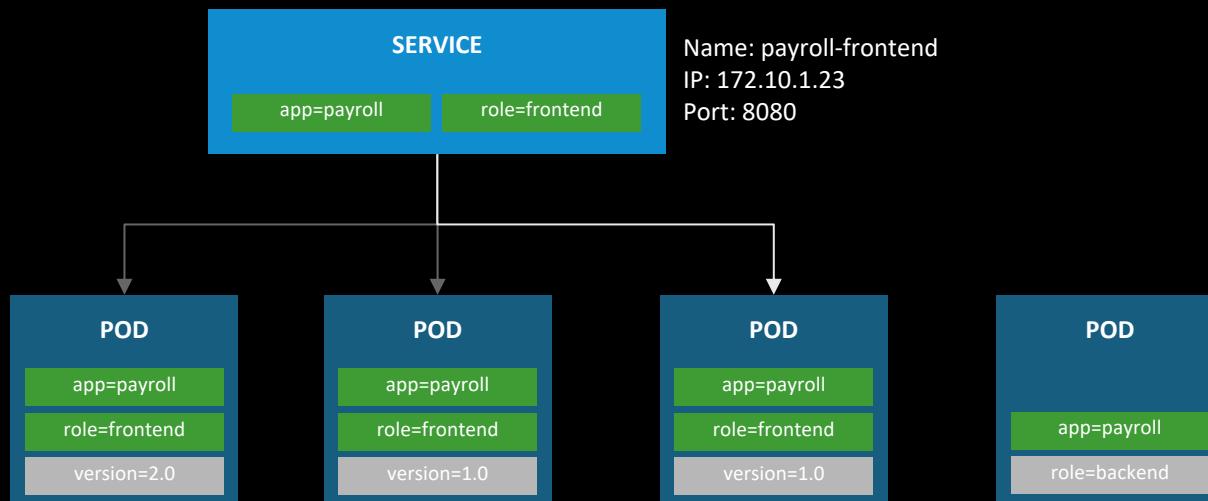
BUILT-IN SERVICE DISCOVERY  
INTERNAL LOAD-BALANCING



# OpenShift Networking

## - Service Discovery

### BUILT-IN SERVICE DISCOVERY INTERNAL LOAD-BALANCING



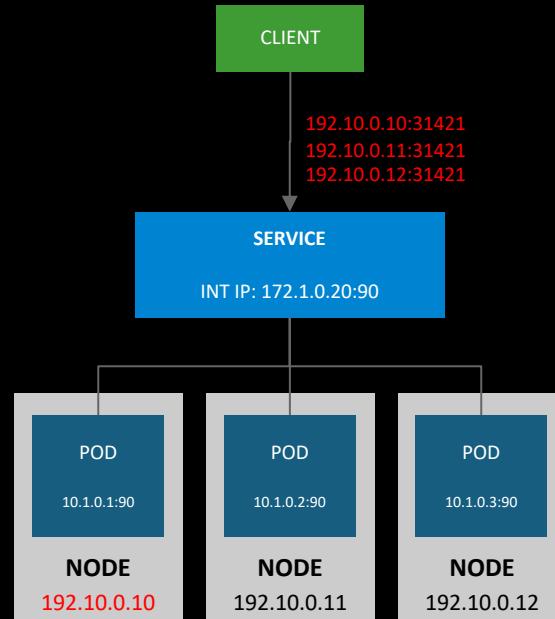
# OpenShift Networking

## - NodePort

50

### 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

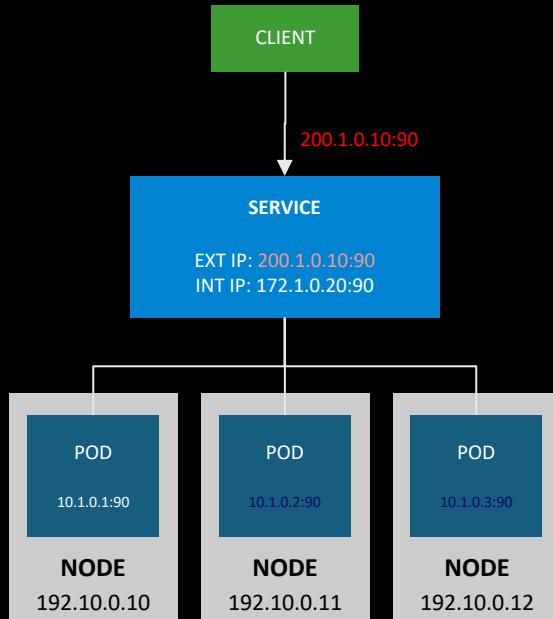


# OpenShift Networking

## - External Traffic

### EXTERNAL TRAFFIC TO A SERVICE ON ANY PORT WITH INGRESS

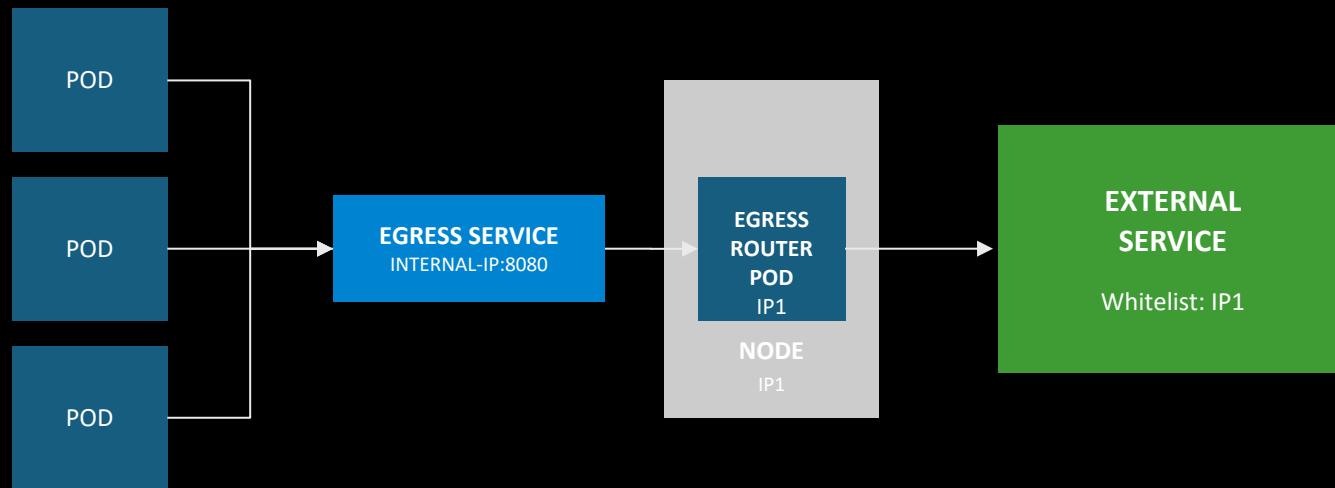
- Access a service with an external IP on any TCP/UDP port, such as
  - Databases
  - Message Brokers
- Automatic IP allocation from a predefined pool using Ingress IP Self-Service
- IP failover pods provide high availability for the IP pool



# OpenShift Networking

## - Egress Traffic

CONTROL OUTGOING TRAFFIC  
SOURCE IP WITH EGRESS ROUTER



# OpenShift Networking

## - Routes

53

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

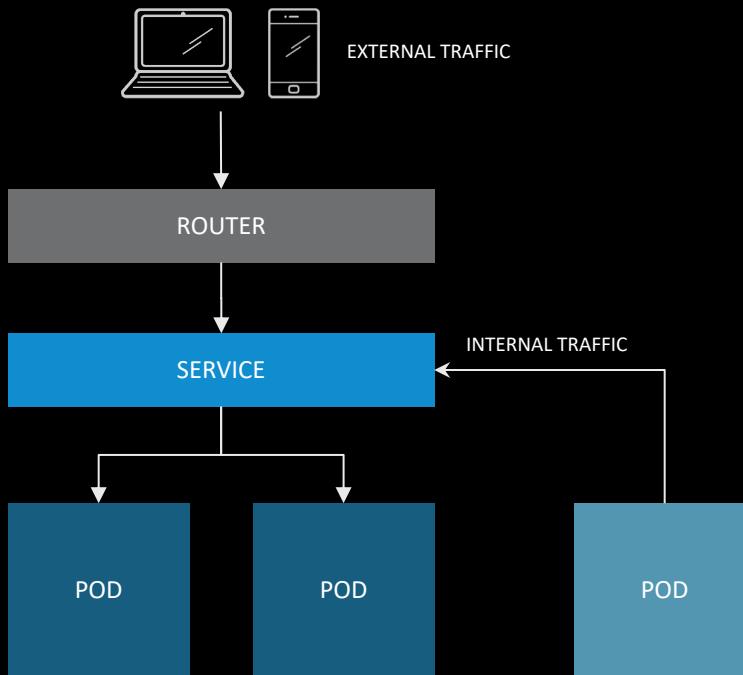


# OpenShift Networking

## - Routes

54

ROUTE EXPOSES SERVICES  
EXTERNALLY

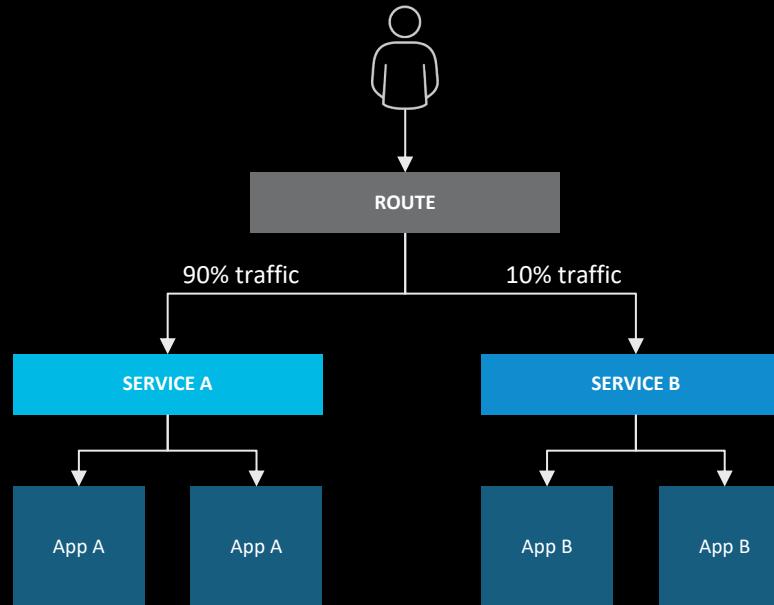


# OpenShift Networking

## - Route Traffic Splitting

### ROUTE SPLIT TRAFFIC

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



# OpenShift Networking - Multiple Pod Networks

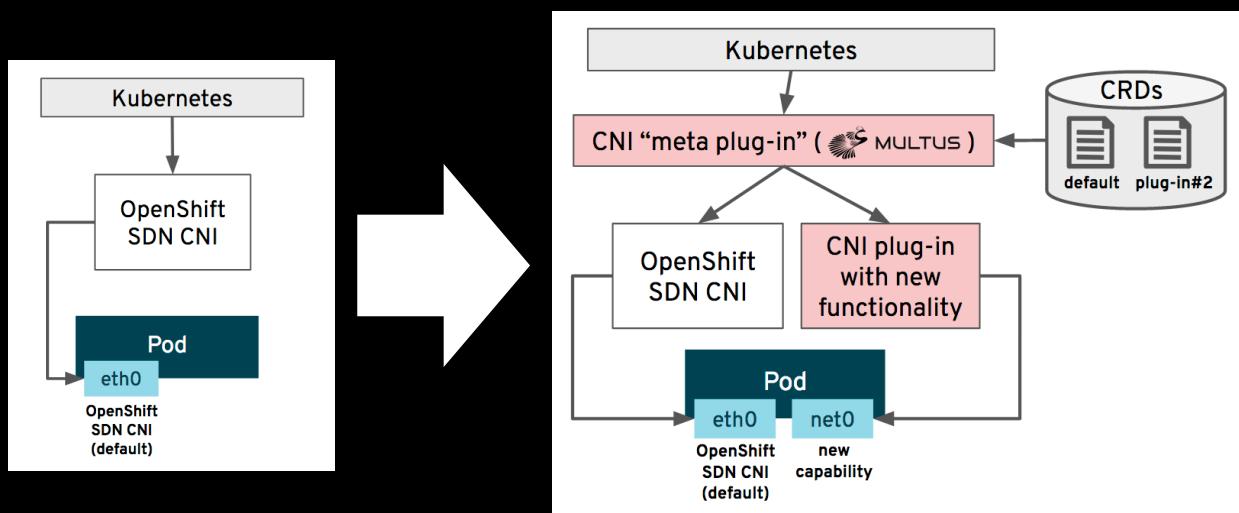
Can add multiple networks to a pod using Multus

## Use Cases

- Performance
- Security

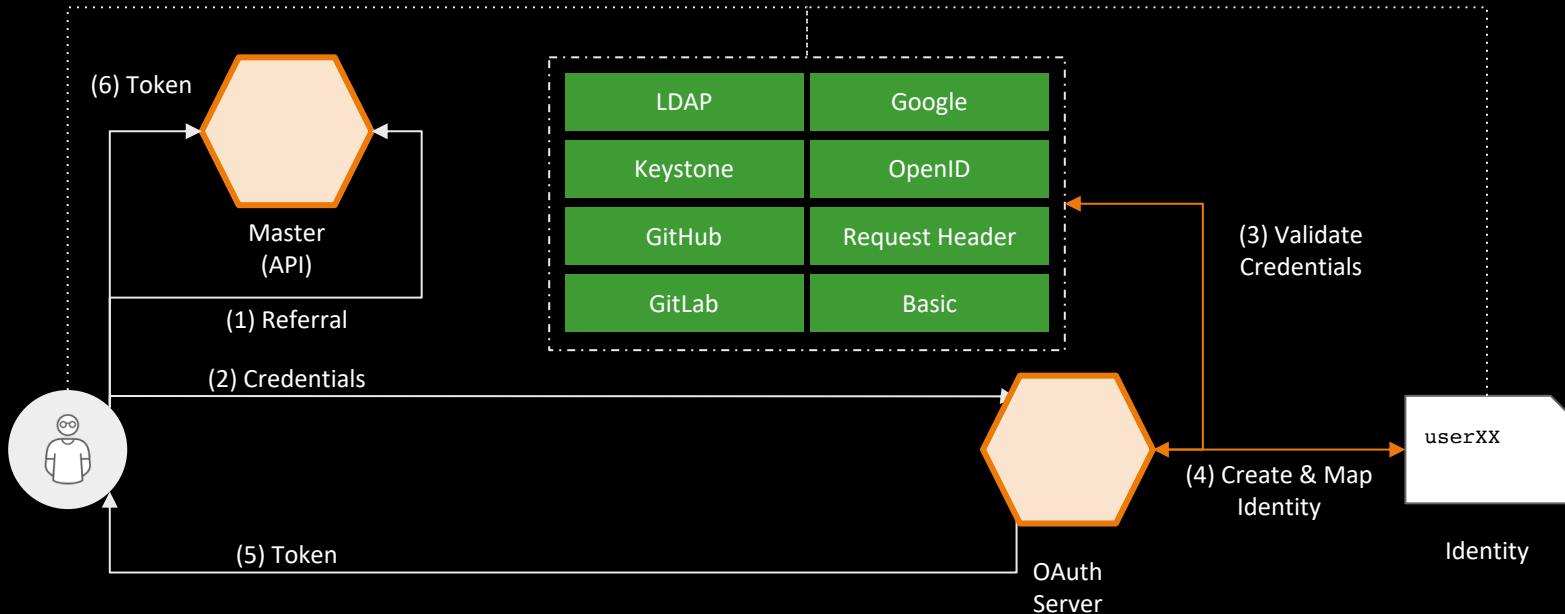
## Additional Networks

- bridge
- host-device
- macvlan
- ipvlan
- SR-IOV



# OpenShift Authentication

# OpenShift Authentication



# OpenShift Authentication - Identity Providers

Many supported providers

Only kubeadmin exists by default, which should be removed

oAuth configuration is done by creating an oAuth Custom Resource

Identity provider	Description
<a href="#">HTPasswd</a>	Configure the htpasswd identity provider to validate user names and passwords against a flat file generated using <a href="#">htpasswd</a> .
<a href="#">Keystone</a>	Configure the keystone identity provider to integrate your OpenShift Container Platform cluster with Keystone to enable shared authentication with an OpenStack Keystone v3 server configured to store users in an internal database.
<a href="#">LDAP</a>	Configure the ldap identity provider to validate user names and passwords against an LDAPv3 server, using simple bind authentication.
<a href="#">Basic authentication</a>	Configure a basic-authentication identity provider for users to log in to OpenShift Container Platform with credentials validated against a remote identity provider. Basic authentication is a generic backend integration mechanism.
<a href="#">Request header</a>	Configure a request-header identity provider to identify users from request header values, such as X-Remote-User. It is typically used in combination with an authenticating proxy, which sets the request header value.
<a href="#">GitHub or GitHub Enterprise</a>	Configure a github identity provider to validate user names and passwords against GitHub or GitHub Enterprise's OAuth authentication server.
<a href="#">GitLab</a>	Configure a gitlab identity provider to use <a href="#">GitLab.com</a> or any other GitLab instance as an identity provider.
<a href="#">Google</a>	Configure a google identity provider using <a href="#">Google's OpenID Connect integration</a> .
<a href="#">OpenID Connect</a>	Configure an oidc identity provider to integrate with an OpenID Connect identity provider using an <a href="#">Authorization Code Flow</a> .

# OpenShift Storage

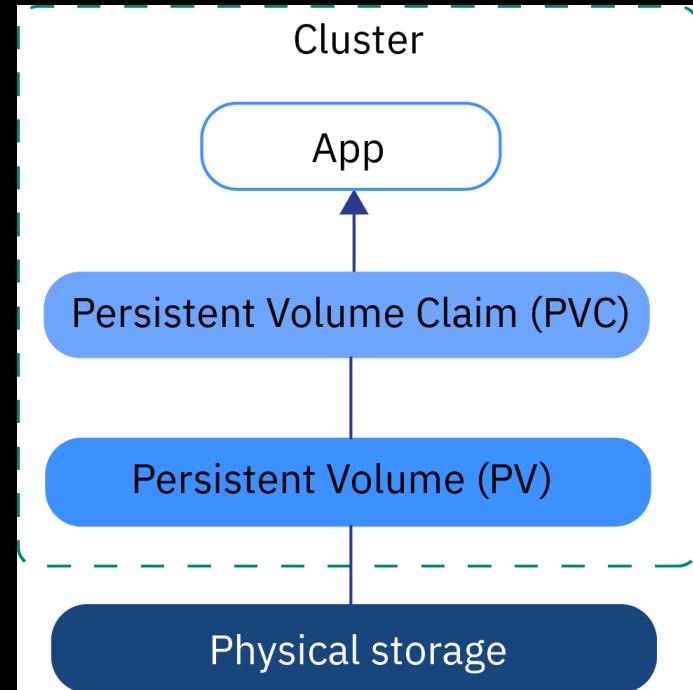
# OpenShift Storage

OpenShift leverages Kubernetes Persistent Volumes

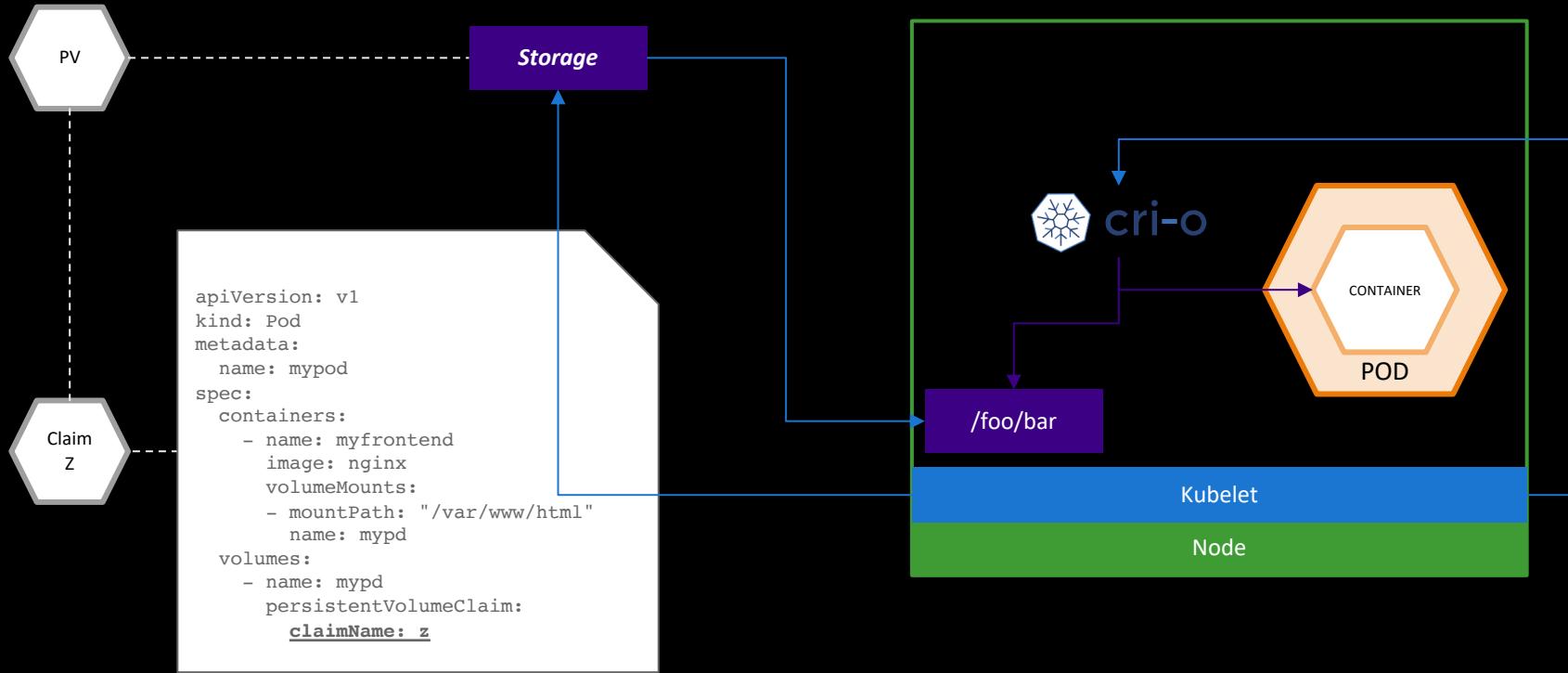
Persistent Volume (PV) is a piece of storage, provisioned by an administrator or dynamically provisioned using [Storage Classes](#)

Persistent Volume Claim (PVC) is a claim for that storage by a user

Storage Classes (SC) allow allocating storage technologies and dynamic provisioning

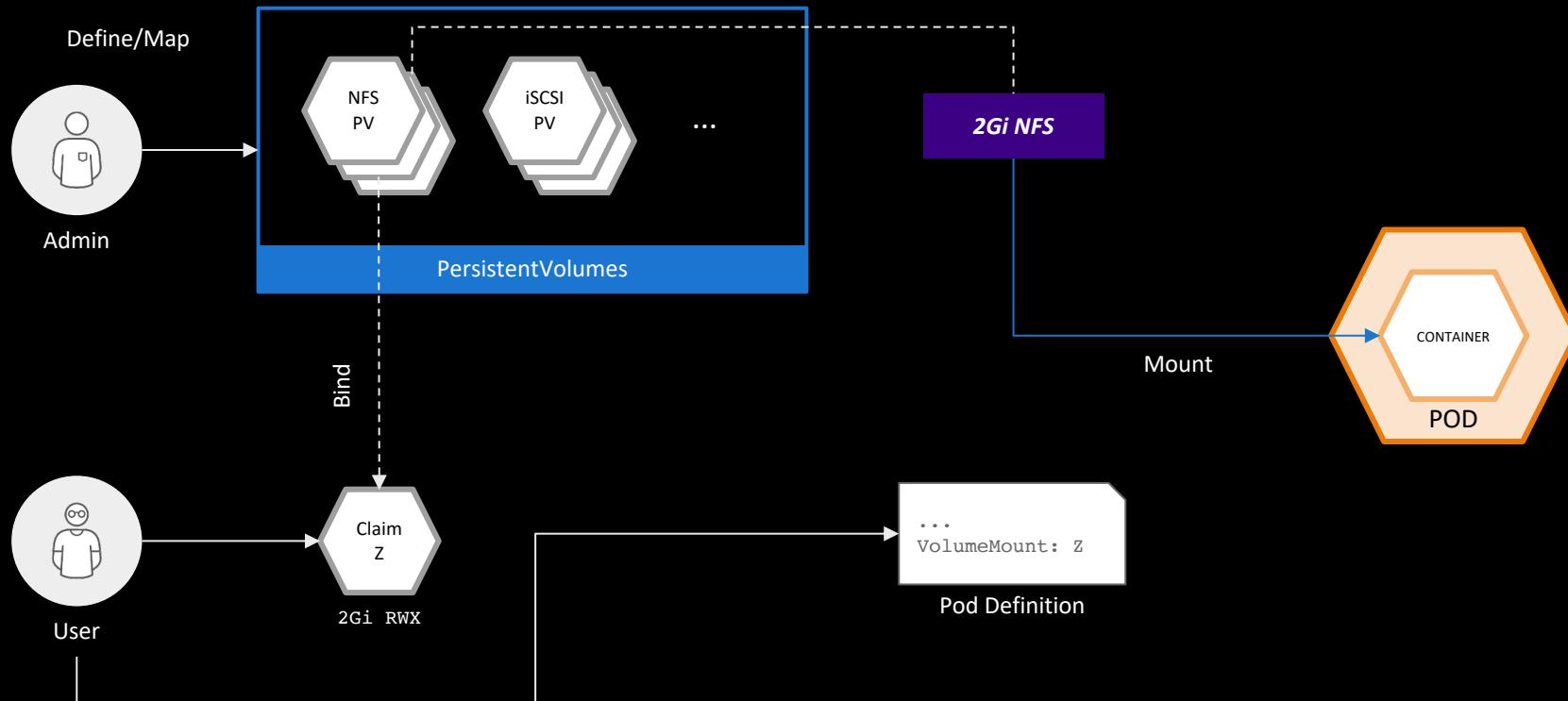


# OpenShift Storage - PV Consumption

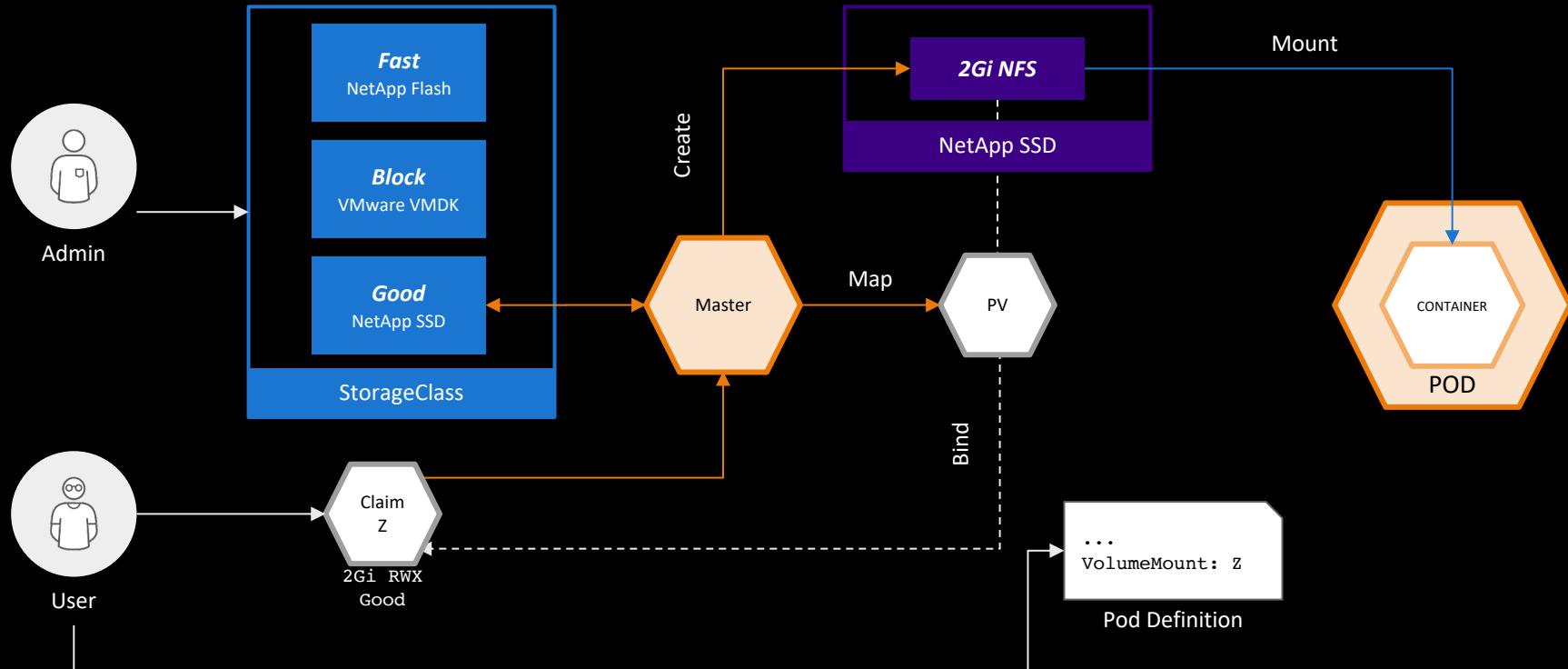


# OpenShift Storage

## - Static Volume Provisioning



# OpenShift Storage - Dynamic Provisioning



# OpenShift Storage

## - Supported Providers

Volume Plug-in	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
AWS EBS	✓	-	-
Azure File	✓	✓	✓
Azure Disk	✓	-	-
Cinder	✓	-	-
Fibre Channel	✓	✓	-
GCE Persistent Disk	✓	-	-
HostPath	✓	-	-
iSCSI	✓	✓	-
Local volume	✓	-	-
NFS	✓	✓	✓
Red Hat OpenShift Container Storage	✓	-	✓
VMware vSphere	✓	-	-

# Machine Config Operator

Provides node lifecycle management

Orchestrate updates

Consists of three main components

- **machine-config-controller** - coordinates machine upgrades from the control plane
- **machine-config-daemon** - runs on each node in the cluster and updates a machine to configuration defined by MachineConfig
- **machine-config-server** - provides the Ignition config files to master nodes as they join the cluster.

# Machines and MachineSets

The Machine API is a combination of primary resources that are based on the upstream Cluster API project and custom OpenShift Container Platform resources.

## Machines

A fundamental unit that describes the host for a Node. A machine has a providerSpec, which describes the types of compute nodes that are offered for different cloud platforms.

## MachineSets

Groups of machines. MachineSets are to machines as ReplicaSets are to Pods. If you need more machines or must scale them down, you change the **replicas** field on the MachineSet to meet your compute need.

NOT CURRENTLY USED IN VMWARE

# Cluster Autoscaler

Adjusts the size of an OpenShift Container Platform cluster to meet its current deployment needs

It uses declarative, Kubernetes-style arguments to provide infrastructure management

ClusterAutoscaler **increases** the size of the cluster when there are Pods that failed to schedule on any of the current nodes due to insufficient resources or when another node is necessary to meet deployment needs

ClusterAutoscaler **decreases** the size of the cluster when some nodes are consistently not needed for a significant period, such as when it has low resource use and all of its important Pods can fit on other nodes.

# OpenShift Installation

# OpenShift Installation

## OPENSHIFT CONTAINER PLATFORM

### Full Stack Automated

Simplified opinionated “Best Practices” for cluster provisioning

Fully automated installation and updates including host container OS.



**Red Hat**  
Enterprise Linux  
CoreOS

### Pre-existing Infrastructure

Customer managed resources & infrastructure provisioning

Plug into existing DNS and security boundaries



**Red Hat**  
Enterprise Linux  
CoreOS



**Red Hat**  
Enterprise  
Linux

## HOSTED OPENSHIFT

### Azure Red Hat OpenShift

Deploy directly from the Azure console. Jointly managed by Red Hat and Microsoft Azure engineers.

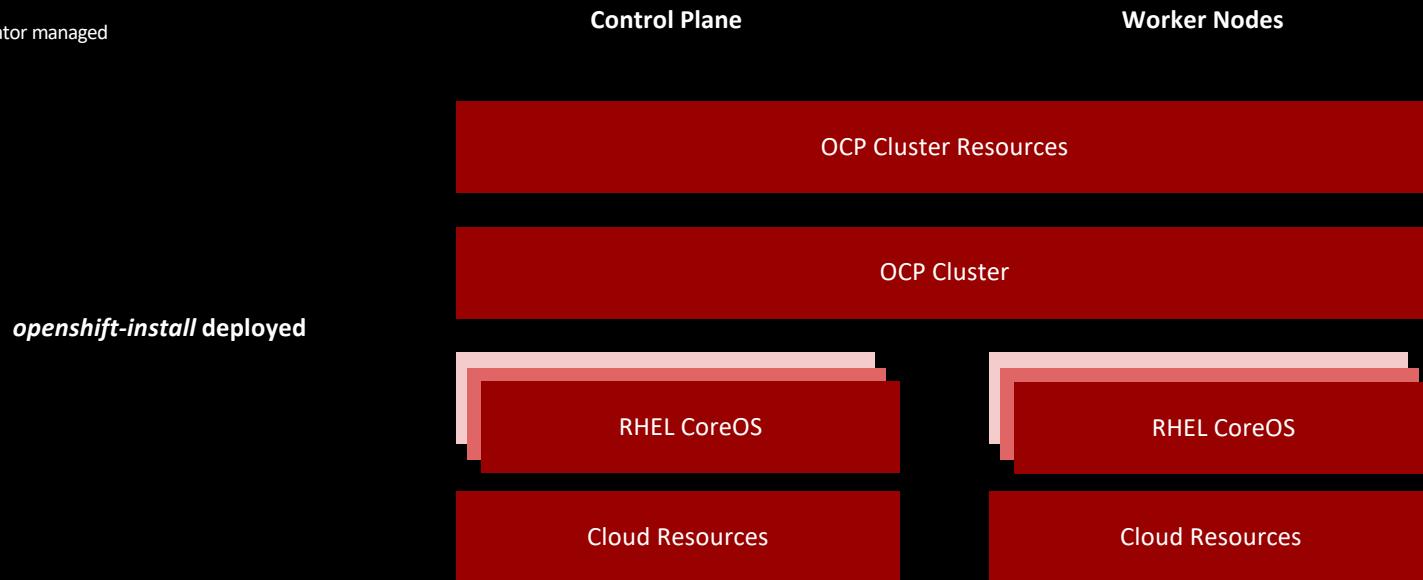
### OpenShift Dedicated

Get a powerful cluster, fully Managed by Red Hat engineers and support.

# OpenShift Installation - Installation Methods

## Full-stack Automated Installation

- User managed
- Operator managed

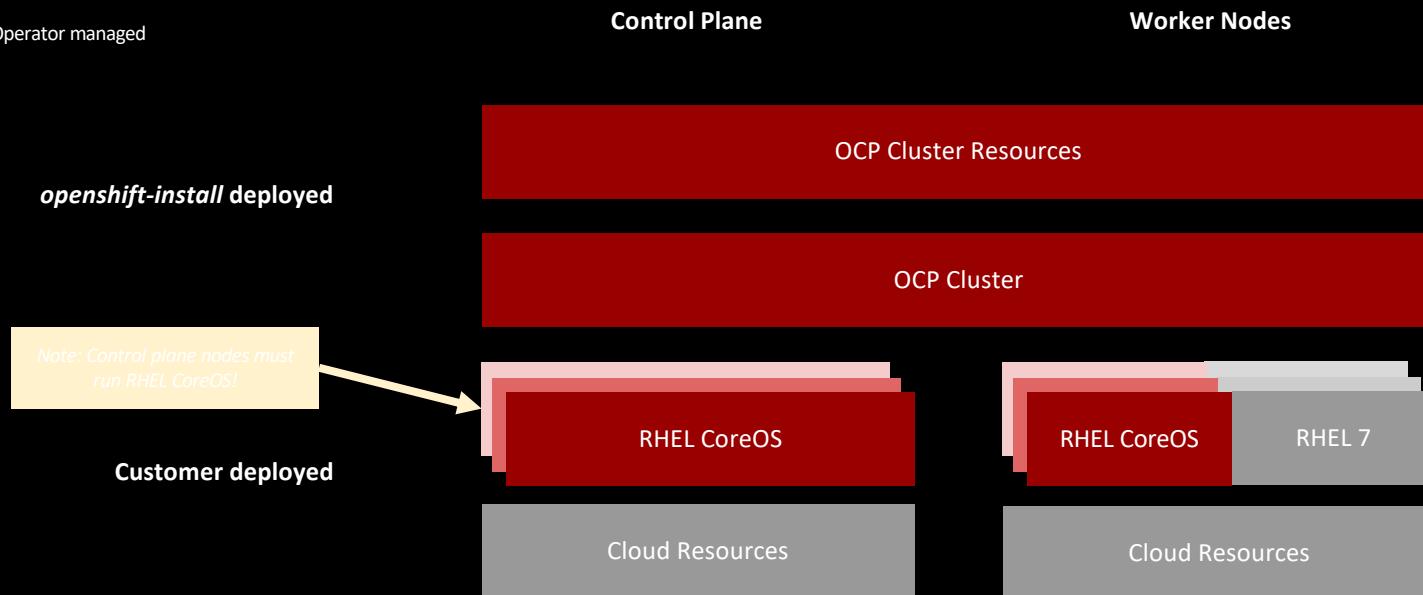


# OpenShift Installation

## - Installation Methods

### Pre-existing Infrastructure Installation

- User managed
- Operator managed



# OpenShift Installation

## - Comparison of Paradigms

	Full Stack Automation	Pre-existing Infrastructure
Build Network	Installer	User
Setup Load Balancers	Installer	User
Configure DNS	Installer	User
Hardware/VM Provisioning	Installer	User
OS Installation	Installer	User
Generate Ignition Configs	Installer	Installer
OS Support	Installer: RHEL CoreOS	User: RHEL CoreOS + RHEL 7
Node Provisioning / Autoscaling	Yes	Only for providers with OpenShift Machine API support

# OpenShift Installation

## - Resource Requirements

Machine	Operating System	vCPU	RAM	Storage
Bootstrap	RHCOS	4	16 GB	120 GB
Control plane	RHCOS	4	16 GB	120 GB
Compute	RHCOS or RHEL 7.6	2	8 GB	120 GB

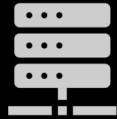
Not sized for additional management services!

# OpenShift Installation

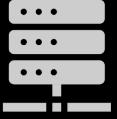
## - Pre-requisites



Ignition  
Webserver



DNS



DHCP



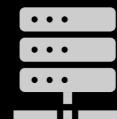
Load Balancer



Internet



Docs



Infrastructure

# OpenShift Installation

## OpenShift Bootstrap Process: Self-Managed Kubernetes

### **How to boot a self-managed cluster:**

OpenShift 4 is unique in that management extends all the way down to the operating system

Every machine boots with a configuration that references resources hosted in the cluster it joins enabling cluster to manage itself

Downside is that every machine looking to join the cluster is waiting on the cluster to be created

Dependency loop is broken using a bootstrap machine, which acts as a temporary control plane whose sole purpose is bringing up the permanent control plane nodes

Permanent control plane nodes get booted and join the cluster leveraging the control plane on the bootstrap machine

Once the pivot to the permanent control plane takes place, the remaining worker nodes can be booted and join the cluster

### **Bootstrapping process step by step:**

Bootstrap machine boots and starts hosting the remote resources required for master machines to boot.

Master machines fetch the remote resources from the bootstrap machine and finish booting.

Master machines use the bootstrap node to form an etcd cluster.

Bootstrap node starts a temporary Kubernetes control plane using the newly-created etcd cluster.

Temporary control plane schedules the production control plane to the master machines.

Temporary control plane shuts down, yielding to the production control plane.

Bootstrap node injects OpenShift-specific components into the newly formed control plane.

Installer then tears down the bootstrap node or if user-provisioned, this needs to be performed by the administrator.

# OpenShift Installation

## 1. Configure DHCP

All the Red Hat Enterprise Linux CoreOS (RHCOS) machines require network in initramfs during boot to fetch Ignition config from the Machine Config Server.

During the initial boot, the machines require a DHCP server in order to establish a network connection, which allows them to download their Ignition config files.

What if I use static/pre-defined IPs?

Still need DHCP.

# OpenShift Installation

## 2. Configure DNS

Component	Record	Description
Kubernetes API	api.<cluster_name>.<base_domain>	This DNS record must point to the load balancer for the control plane machines. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	api-int.<cluster_name>.<base_domain>	This DNS record must point to the load balancer for the control plane machines. This record must be resolvable from all the nodes within the cluster. The API server must be able to resolve the worker nodes by the host names that are recorded in Kubernetes. If it cannot resolve the node names, proxied API calls can fail, and you cannot retrieve logs from Pods.
Routes	*.apps.<cluster_name>.<base_domain>	A wildcard DNS record that points to the load balancer that targets the machines that run the Ingress router pods, which are the worker nodes by default. This record must be resolvable by both clients external to the cluster and from all the nodes within the cluster.
	etcd-<index>.<cluster_name>.<base_domain>	OpenShift Container Platform requires DNS records for each etcd instance to point to the control plane machines that host the instances. The etcd instances are differentiated by <index> values, which start with 0 and end with n-1, where n is the number of control plane machines in the cluster. The DNS record must resolve to an unicast IPv4 address for the control plane machine, and the records must be resolvable from all the nodes in the cluster. For each control plane machine, OpenShift Container Platform also requires a SRV DNS record for etcd server on that machine with priority 0, weight 10 and port 2380. A cluster that uses three control plane machines requires the following records:
etcd	_etcd-server-ssl._tcp.<cluster_name>.<base_domain>	# _service._proto.name. TTL class SRV priority weight port target. _etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN SRV 0 10 2380 etcd-0.<cluster_name>.<base_domain>. _etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN SRV 0 10 2380 etcd-1.<cluster_name>.<base_domain>. _etcd-server-ssl._tcp.<cluster_name>.<base_domain> 86400 IN SRV 0 10 2380 etcd-2.<cluster_name>.<base_domain>.

# OpenShift Installation

## 3. Configure Load Balancer

Port	Machines	Internal	External	Description
<b>6443</b>	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	x	x	Kubernetes API server
<b>22623</b>	Bootstrap and control plane. You remove the bootstrap machine from the load balancer after the bootstrap machine initializes the cluster control plane.	x		Machine Config server
<b>443</b>	The machines that run the Ingress router pods, compute, or worker, by default.	x	x	HTTPS traffic
<b>80</b>	The machines that run the Ingress router pods, compute, or worker by default.	x	x	HTTP traffic

# OpenShift Installation

## 4. Generate SSH Keys

1. If you do not have an SSH key that is configured for password-less authentication on your computer, create one. For example, on a computer that uses a Linux operating system, run the following command:

```
$ ssh-keygen -t rsa -b 4096 -N '' \  
-f <path>/<file_name> ①
```

2. Start the **ssh-agent** process as a background task:

```
$ eval "$(ssh-agent -s)"  
Agent pid 31874
```

3. Add your SSH private key to the **ssh-agent**:

```
$ ssh-add <path>/<file_name> ①  
Identity added: /home/<you>/<path>/<file_name> (<computer_name>)
```

# OpenShift Installation

## 5. Get the openshift-install tool

<https://cloud.redhat.com/openshift/install>

# OpenShift Installation

## 6. Edit the install-config.yaml

```
apiVersion: v1
baseDomain: openshift.cloud.lab.com
compute:
- hyperthreading: Enabled
  name: worker
  replicas: 0
controlPlane:
  hyperthreading: Enabled
  name: master
  replicas: 3
metadata:
  name: ocp4
platform:
  vsphere:
    vcenter: 9.180.210.21
    username: Administrator@isstlab.org
    password: Passw0rd!
    datacenter: clod-fondly
    defaultDatastore: cluster02
pullSecret: '{"auths":{"mirror-registry.openshift.cloud.lab.com:5001":{"auth": "YWRtaW47YWMjAxOQ==", "email": "admin@isstlab.com"}}}'
sshKey: 'ssh-rsa ...dK8Fpb8i1wdpQ== root@lf-ocp-boot'
```

[https://docs.openshift.com/container-platform/4.2/installing/installing\\_vsphere/installing-vsphere.html#installation-initializing-manual\\_installing-vsphere](https://docs.openshift.com/container-platform/4.2/installing/installing_vsphere/installing-vsphere.html#installation-initializing-manual_installing-vsphere)

# OpenShift Installation

## 7. Create manifests and ignition files

```
$ ./openshift-install create manifests --dir=<installation_directory>
```

Replace the masterSchedulable flag from true to false in cluster-scheduler-02-config.yml

```
$ sed -i "s/mastersSchedulable: true/mastersSchedulable: false/g manifests/cluster-scheduler-02-config.yml
```

```
$ ./openshift-install create ignition-configs --dir=<installation_directory>
```

```
├── auth
│   ├── kubeadmin-password
│   └── kubeconfig
├── bootstrap.ign
├── id_rsa
├── install-config.yaml
├── master.ign
├── worker.ign
└── metadata.json
```

# OpenShift Installation

## 8. Create append-bootstrap ignition file

Upload the bootstrap Ignition config file, which is named <installation\_directory>/bootstrap.ign, that the installation program created to your HTTP server. Note the URL of this file.

```
{  
  "ignition": {  
    "config": {  
      "append": [  
        {  
          "source": "<bootstrap_ignition_config_url>",  
          "verification": {}  
        }  
      ]  
    },  
    "timeouts": {},  
    "version": "2.1.0"  
  },  
  "networkd": {},  
  "passwd": {},  
  "storage": {},  
  "systemd": {}  
}
```

# OpenShift Installation

## 9. Encrypt ignition files

```
$ base64 -w0 <installation_directory>/master.ign > <installation_directory>/master.64  
$ base64 -w0 <installation_directory>/worker.ign > <installation_directory>/worker.64  
$ base64 -w0 <installation_directory>/append-bootstrap.ign > <installation_directory>/append-bootstrap.64
```

These will end up as the encoded ignition configuration for the RHCOS VM

# OpenShift Installation

## 10. Upload RHCOS OVA to vSphere client

## 11. Create infrastructure from RHCOS template

## 12. Wait for bootstrap to complete

```
$ ./openshift-install --dir=<installation_directory> wait-for bootstrap-complete --log-level=info
```

```
INFO Waiting up to 30m0s for the Kubernetes API at https://api.test.example.com:6443...
INFO API v1.14.6+c4799753c up
INFO Waiting up to 30m0s for the bootstrap-complete event...
```

After bootstrap process is complete, remove the bootstrap machine from the load balancer.

## 13. Check Operator status

```
$ export KUBECONFIG=<installation_directory>/auth/kubeconfig
```

```
$ watch -n5 oc get clusteroperators
```

NAME	VERSION	AVAILABLE	PROGRESSING	DEGRADED	SINCE
authentication	4.2.0	True	False	False	69s
cloud-credential	4.2.0	True	False	False	12m
cluster-autoscaler	4.2.0	True	False	False	11m
console	4.2.0	True	False	False	46s
dns	4.2.0	True	False	False	11m

# OpenShift Installation

## 14. Configure Image Registry Storage

### Production

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: registry
provisioner: no-provisioning
parameters:

apiVersion: v1
kind: PersistentVolume
metadata:
  name: registry
spec:
  capacity:
    storage: 100Gi
  accessModes:
    - ReadWriteMany
  nfs:
    path: /nfs/ocp4/registry
    server: nfs.ocp4.openshift.cloud.lab.com
  persistentVolumeReclaimPolicy: Retain
  storageClassName: registry

apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  finalizers:
    - kubernetes.io/pvc-protection
  name: image-registry-storage
  namespace: openshift-image-registry
  annotations:
    imageregistry.openshift.io: "true"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: registry
```

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"claim":"image-registry-storage"}}}'
```

### Non-production (default)

```
$ oc patch configs.imageregistry.operator.openshift.io cluster --type merge --patch '{"spec":{"storage":{"emptyDir":{}}}}'
```

# OpenShift Installation

## 15. Wait for cluster to install

```
$ ./openshift-install --dir=<installation_directory> wait-for install-complete  
INFO Waiting up to 30m0s for the cluster to initialize...
```

The command succeeds when the Cluster Version Operator finishes deploying the OpenShift Container Platform cluster from Kubernetes API server.

```
$ oc get pods --all-namespaces  
  
NAMESPACE          NAME                           READY   STATUS    RESTARTS   AGE  
openshift-apiserver-operator  openshift-apiserver-operator-85cb746d55-zqhs8  1/1     Running   1          9m  
openshift-apiserver        api-server-67b9g                      1/1     Running   0          3m  
openshift-apiserver        apiserver-ljcmx                     1/1     Running   0          1m  
openshift-apiserver        apiserver-z25h4                     1/1     Running   0          2m  
openshift-authentication-operator authentication-operator-69d5d8bf84-vh2n8  1/1     Running   0          5m  
  
...
```

# OpenShift Installation

[https://github.com/ibm-cloud-architecture/refarch-privatecloud/blob/master/Install\\_OCP\\_4.x.md](https://github.com/ibm-cloud-architecture/refarch-privatecloud/blob/master/Install_OCP_4.x.md)

# OpenShift Installation Troubleshooting

## Installation

- If the bootstrap is not working properly
  - ignition config could be wrong
  - DHCPDISCOVERY not working
  - DNS not working properly
  - Check firewalls
- If using a mirror-registry (air-gapped) make sure you've extracted the correct openshift-install tool
- If control-plane/compute nodes are not working
  - If nodes cannot pull ignition config?
    - bootstrap is probably not working
    - Check LB is working and routing traffic correctly
  - If nodes cannot resolve DNS names... check DNS entries are correct

# Day 2 Operations

# Day 2 Operations

- Cluster Updates
- OpenShift Cluster Logging
- OpenShift Cluster Monitoring
- Adding/Removing nodes
- Node maintenance

# Day 2 Operations - Cluster updates

The screenshot displays the Red Hat OpenShift Container Platform interface. On the left, a sidebar menu includes 'Administrator', 'Home', 'Dashboards', 'Projects', 'Search', 'Explore', 'Events', 'Operators', and 'Workloads'. The main content area shows 'Cluster Settings' with tabs for 'Overview' (selected), 'Cluster Operators', and 'Global Configuration'. The 'Overview' tab displays a table with columns 'Channel' (stable-4.2) and 'Last Completed Version' (4.2.10). A blue button labeled 'Update to another version' is visible. Below the table, a note says 'View this cluster and manage subscription settings in OpenShift Cluster Manager'. A modal window titled 'Update Cluster' is open, showing the current version as 4.2.10 and a dropdown menu for 'Select New Version' with options 4.2.14, 4.2.13, and 4.2.12. At the bottom of the modal, there are 'Cancel' and 'Update' buttons. In the bottom right corner of the main screen, there is a small status message: 'Working towards 4.2.14: 13% complete' with a 'View details' link.

# Day 2 Operations

## - Cluster Logging



Log data storage  
via **ElasticSearch**



Log collection by **Fluentd**



Log visualisation by **Kibana**

# Day 2 Operations

## - Cluster Logging

### 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

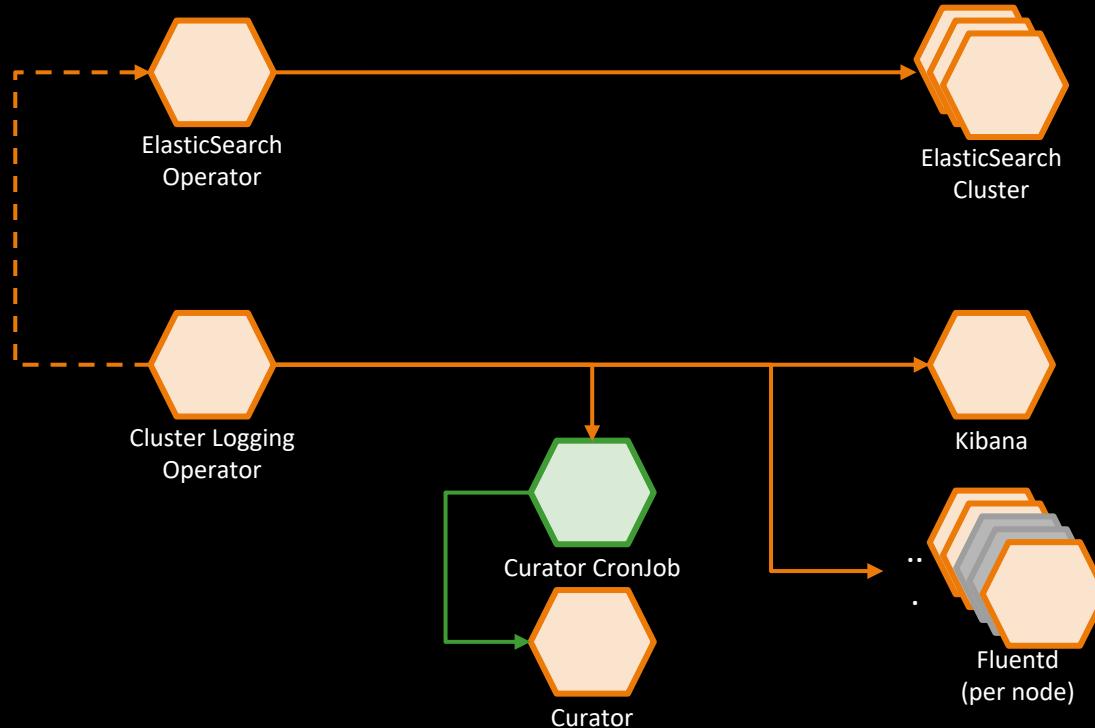
- Cluster administrators can view all logs
- Users can only view logs for their projects

#### Ability to forward logs elsewhere

- External elasticsearch, Splunk, etc

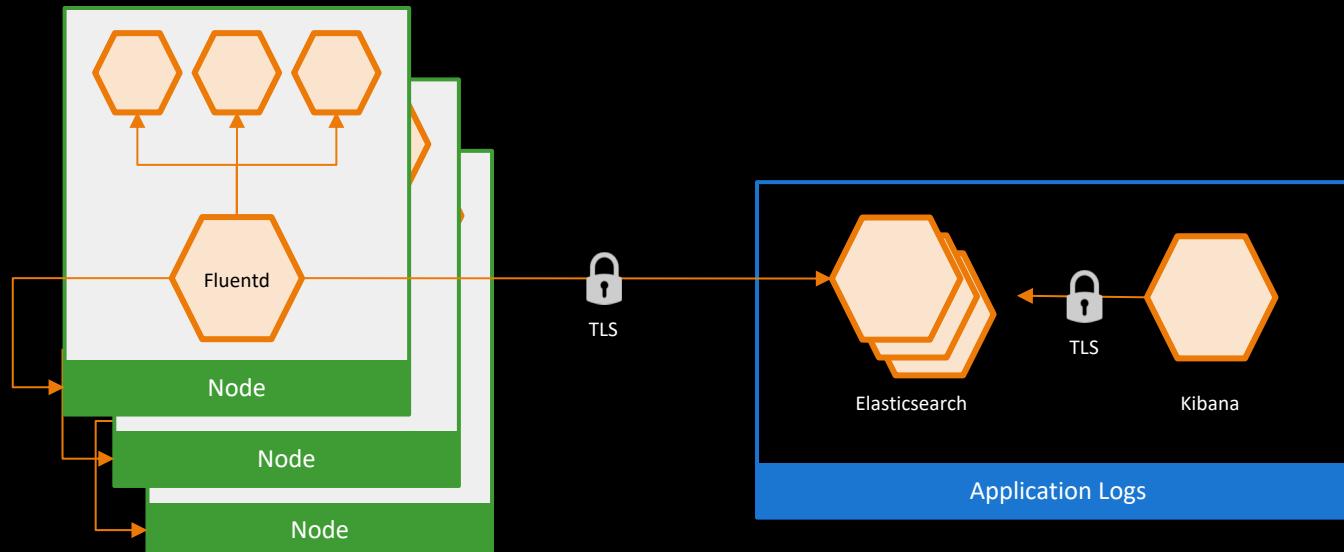
# Day 2 Operations

## - Cluster Logging



# Day 2 Operations

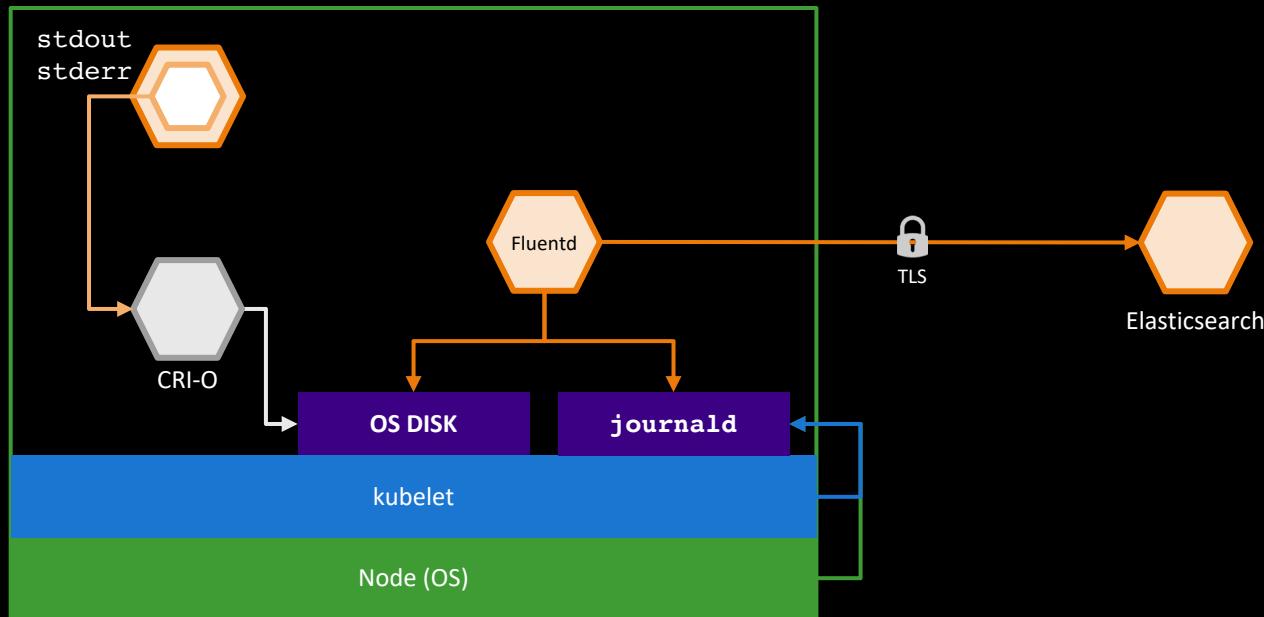
## - Cluster Logging



# Day 2 Operations

## - Cluster Logging

Log data flow



# Day 2 Operations

## - Cluster Monitoring



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



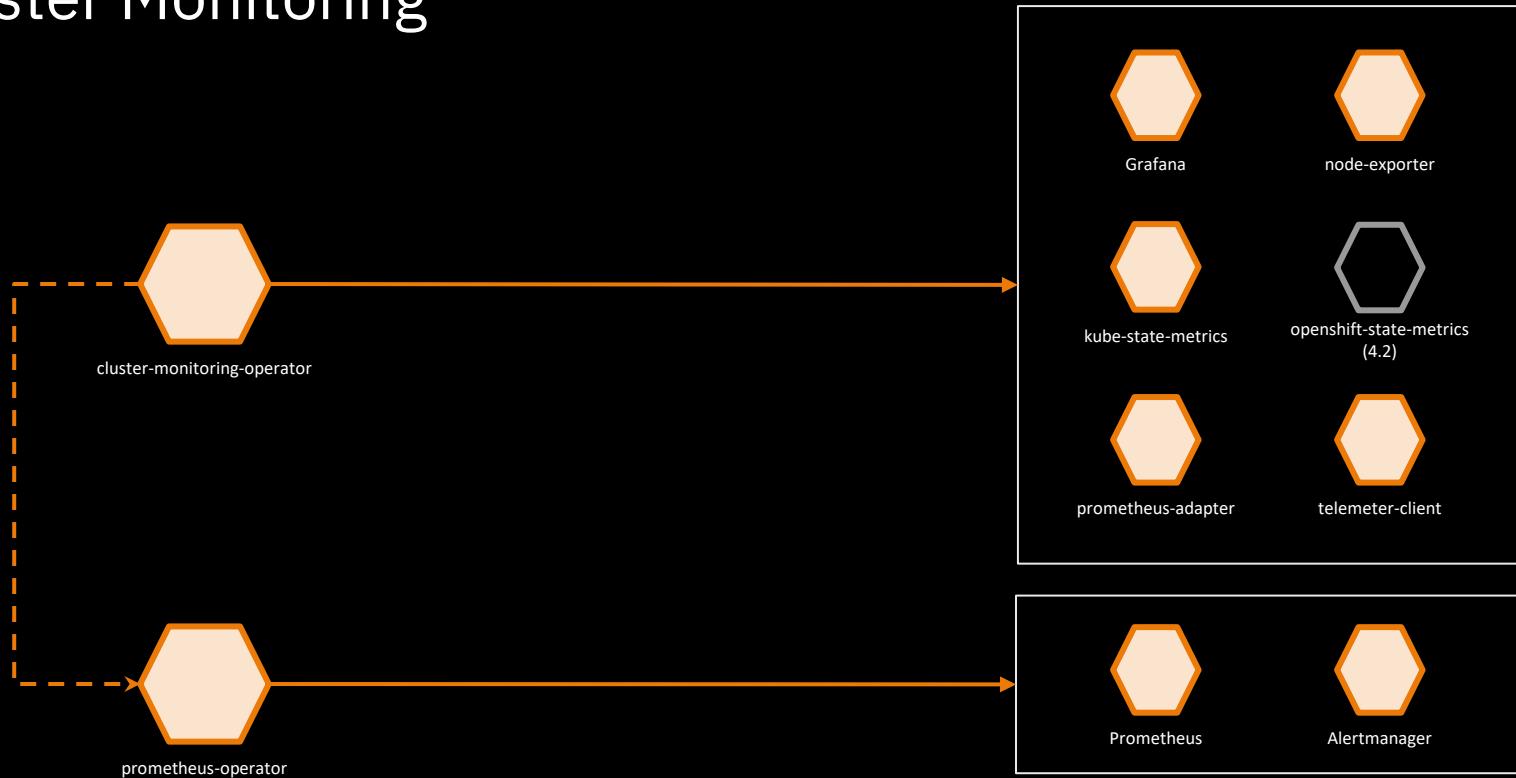
**Alerting/notification** via Prometheus' Alertmanager, an open-source tool that handles alerts sent by Prometheus.



**Metrics visualization** via Grafana, the leading metrics visualization technology.

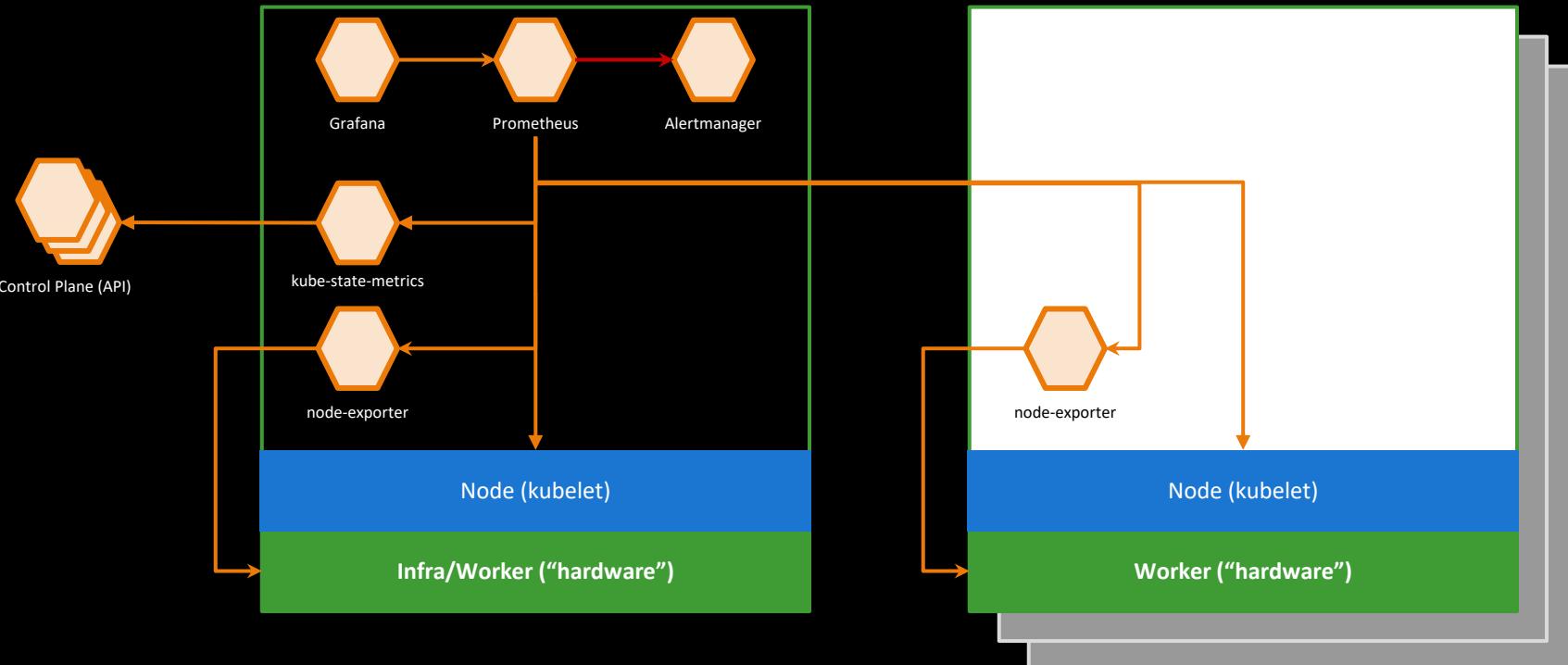
# Day 2 Operations

## - Cluster Monitoring



# Day 2 Operations

## - Cluster Monitoring



# Day 2 Operations

## - Adding/Removing nodes

### Add Node

1. Deploy new worker node using worker ignition config
2. Accept the CSRs in OpenShift (`oc adm certificate approve <csr-name>`)
3. Node joins the cluster

### Remove Node

1. Get the machine  
`$ oc get machine -n openshift-machine-api`
2. Delete the machine  
`$ oc delete <machine> -n openshift-machine-api`
3. Delete the VM

# Day 2 Operations

## - Node maintenance

What node maintenance?



# Day 2 Operations

## - Debugging nodes

As an Administrator, you have access to commands to debug CoreOS nodes

```
[root@ocp4-inf ~]# oc debug node/worker4.ocp4.os.fyre.ibm.com
Starting pod/worker4ocp4osfyreibmcom-debug ...
To use host binaries, run `chroot /host`
Pod IP: 10.16.18.130
If you don't see a command prompt, try pressing enter.
sh-4.2# chroot /host
sh-4.4# whoami
root
sh-4.4# systemctl status crio
● crio.service - Open Container Initiative Daemon
  Loaded: loaded (/usr/lib/systemd/system/crio.service; disabled; vendor preset: disabled)
  Drop-In: /etc/systemd/system/crio.service.d
            └─10-default-env.conf
    Active: active (running) since Tue 2020-06-23 18:36:21 UTC; 2 days ago
      Docs: https://github.com/cri-o/cri-o
   Main PID: 1327 (crio)
     Tasks: 30
    Memory: 1.6G
      CPU: 7h 1min 9.720s
     CGroup: /system.slice/crio.service
              └─1327 /usr/bin/crio --enable-metrics=true --metrics-port=9537
```