

Administración

Gonzalo Acosta <gonzalo.acosta@semperti.com>

Agenda

Día 1	Día 2	Día 3	Día 4	Día 5
Contenedores Kubernetes	Producto Infra Providers Arquitectura Datos de Aplicación Network Workflow Build and Deploy	Instalación Autenticación Administración de Grupos. RBAC Service Account Resource Management	Networking Network Policies Machines y Machines Config LimitRange	Horizontal Pod Autoscaling Upgrade Process Cluster Metrics Cluster Logging

Día 1

Contenedores Podman

Semperti

Tecnología de Contenedores

Despliegue tradicional

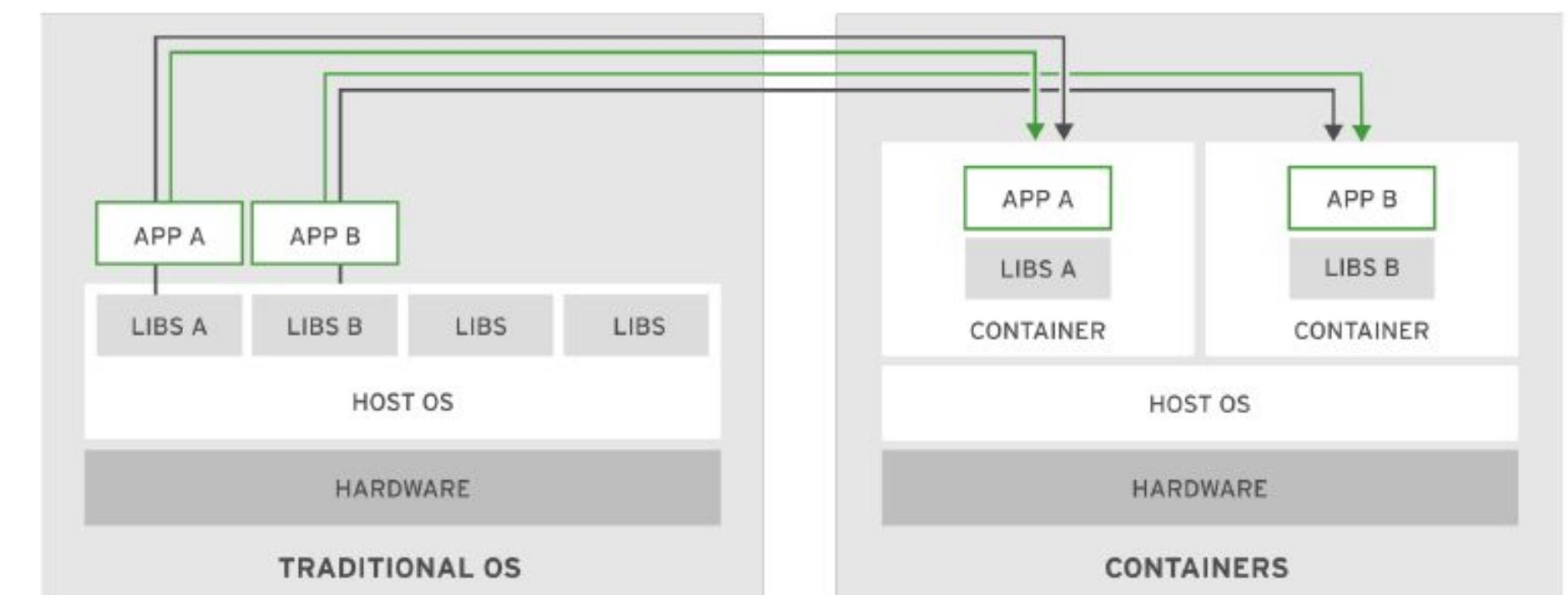
Provisión tradicional

Habitualmente las aplicaciones dependen:

- Librerías
- Archivos de configuración
- Environment runtime.
- Hardware físico o virtual.

Posibles problemas:

- Update Sistema Operativo?
- Cambio de hardware o máquina?
- Eliminación de dependencias.
- Stop > Update > Start = Corte de servicio



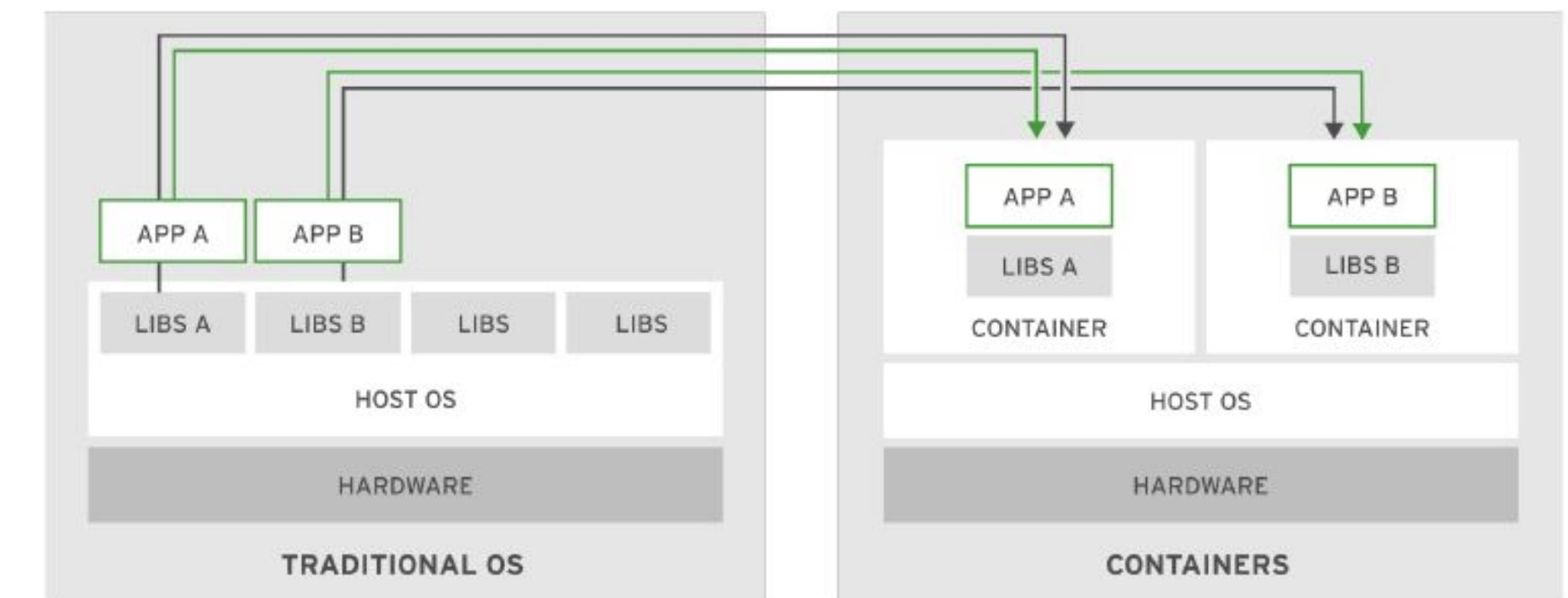
Tecnología de Contenedores

Contenedores

Provisión en contenedores

Un **contenedor** es un set de uno o más procesos de manera aislados del resto del sistema.

- Mismo beneficio que una VM, seguridad, storage, networking aislado.
- Requieren menos recursos de hardware.
- Menor tiempo de start/stop
- Pueden aislarse a nivel recursos (CPU, Memoria)



Tecnología de Contenedores

OCI Specification

Open Container Initiative (OCI)

Proporciona un conjunto de estándares de la industria que definen dos aspectos:

- **image-spec:** Las especificaciones de imágenes definen cómo será el formato del bundle de archivos y metadata que tendrá la imagen de un contenedor.
- **runtime-spec:** Algunos de los runtime disponibles son containerd, CRI-O, Firecracker, lxd, runc, docker, podman.



<https://opencontainers.org>

<https://landscape.cncf.io/category=container-runtime>

Tecnología de Contenedores

Arquitectura de Contenedores

El kernel de linux posee una serie de features que permiten aislar procesos:

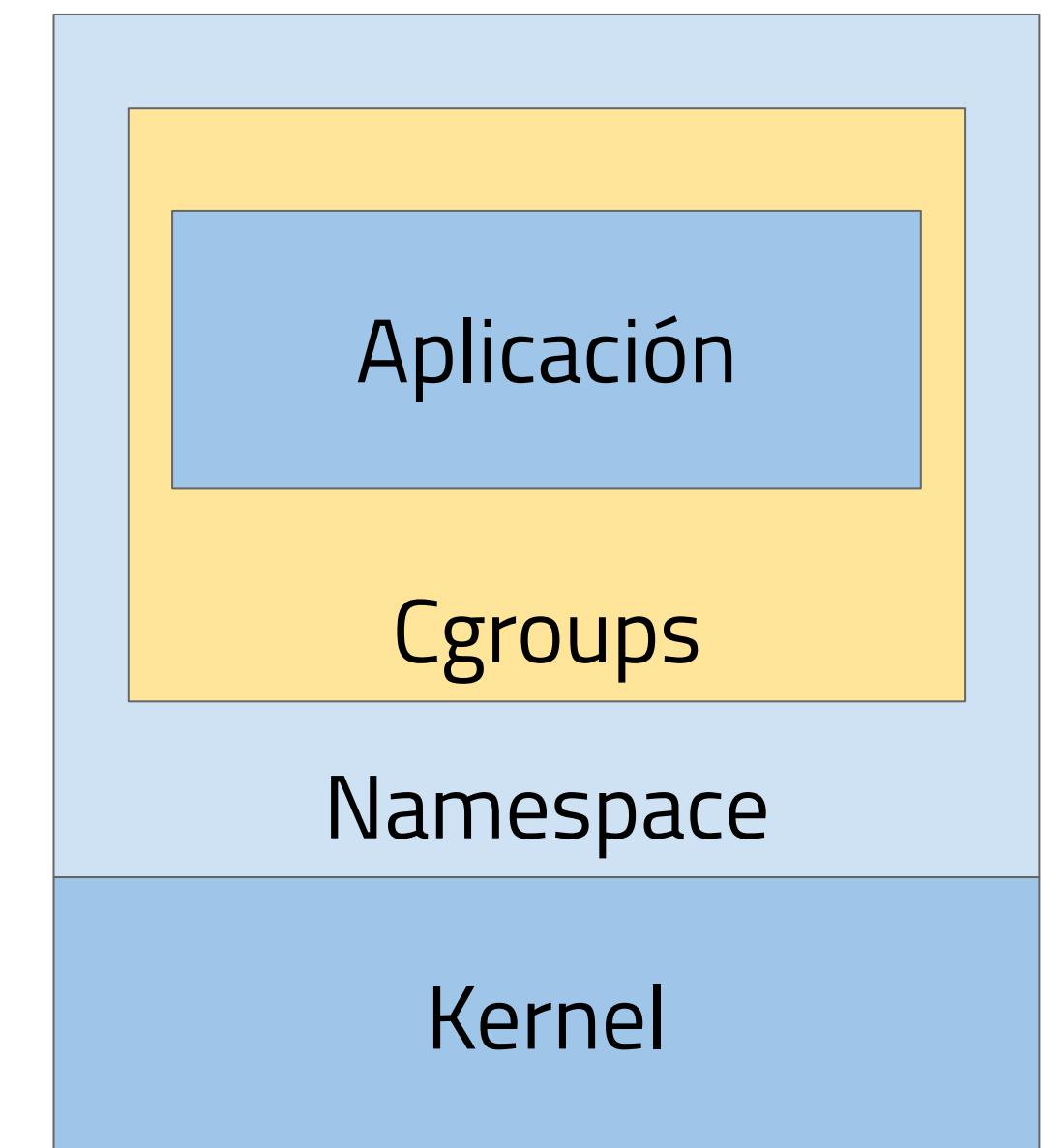
- **Namespaces:**

El kernel aislar recursos específicos de sistemas, usualmente visibles por todos los procesos y colocarlos dentro de un namespace. Solo los procesos que son miembros pueden ver los recursos.

Pueden incluir *red, interfaces, process id, puntos de montaje, recursos de IPC y información del host de sistema.*

- **Control Groups (cgroups):**

Los *cgroups* imponen *restricciones sobre la cantidad de recursos* del sistema que los procesos podrían usar. Protegen los recursos del sistema.



Tecnología de Contenedores

Arquitectura de Contenedores

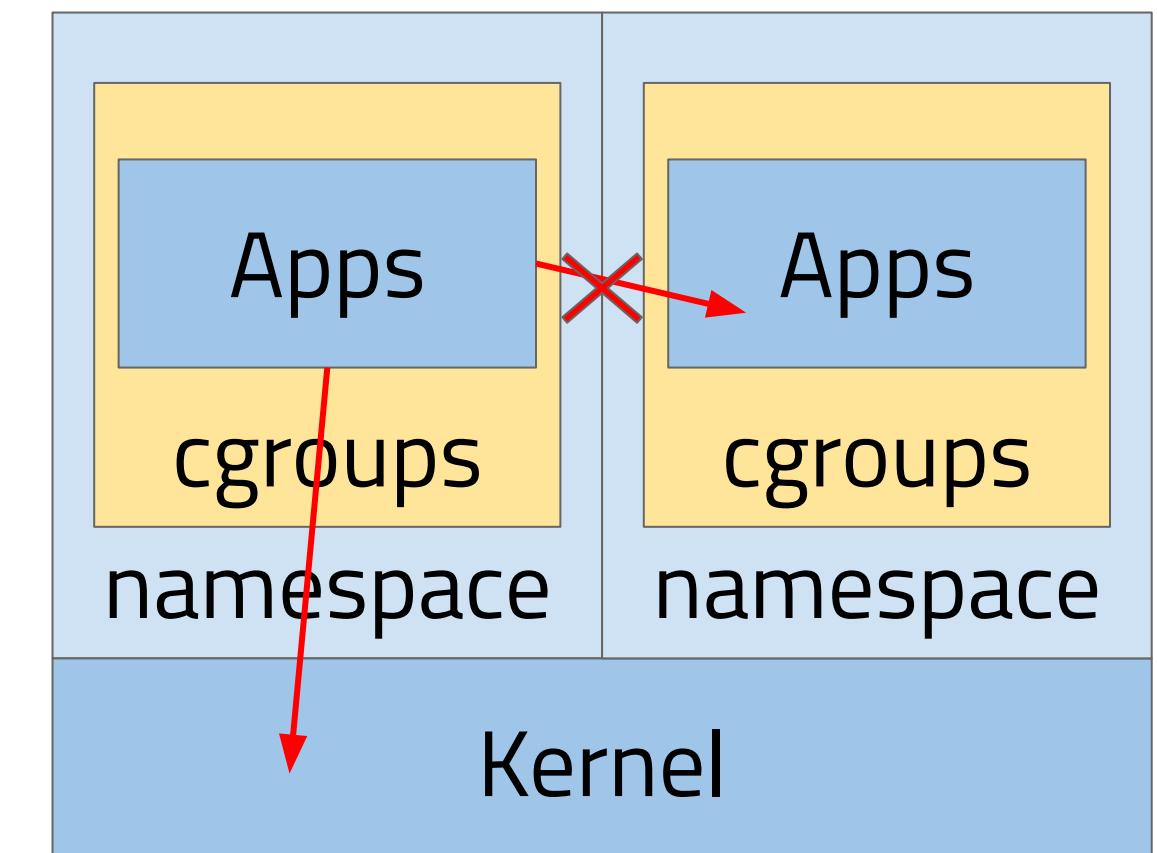
El kernel de linux posee una serie de features que permiten aislar procesos:

- **SECCOMP:**

Limita *cómo pueden los procesos consumir las system calls*. Define un *security profile* para procesos. Whitelisting de syscalls, params y que file descriptors que son permitidos para usar.

- **SELinux:**

SELinux (Security-Enhanced Linux) definen el sistema de control de acceso para procesos. El kernel de linux utiliza SELinux para proteger procesos de otros procesos y también limitar el acceso a archivos de sistema del host.

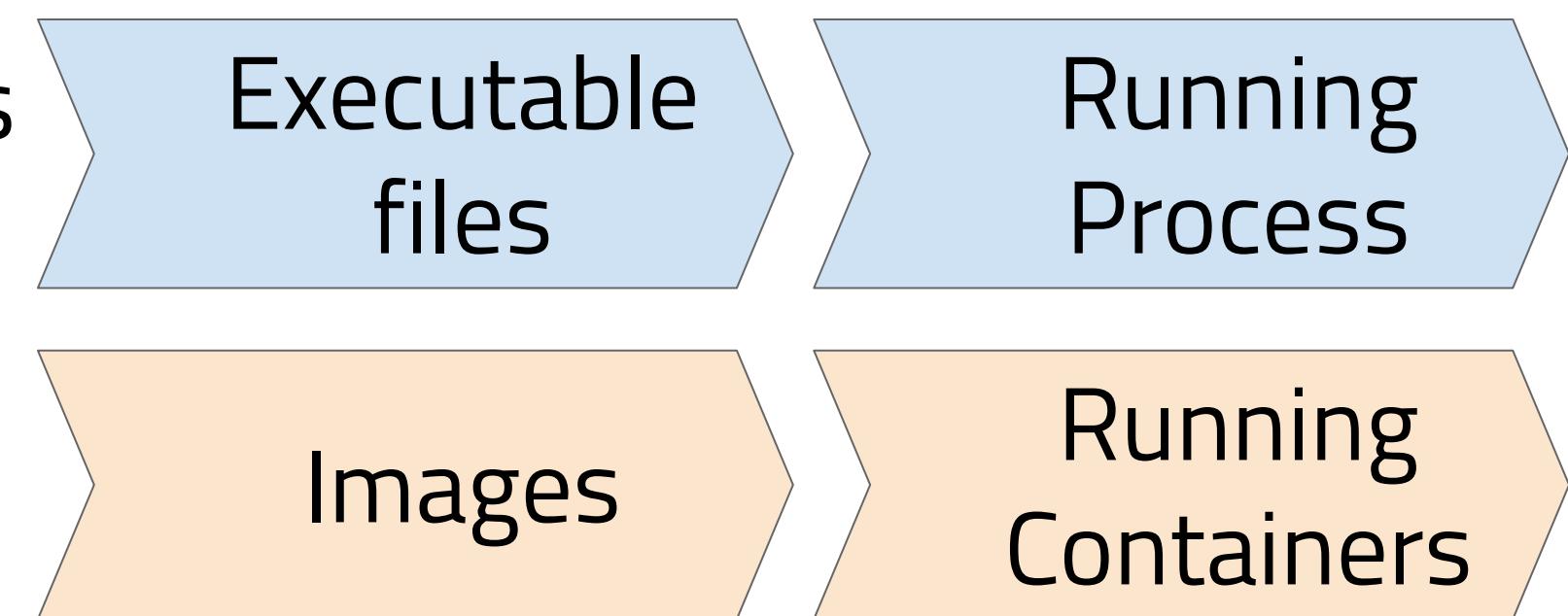


Tecnología de Contenedores

Arquitectura de Contenedores

Desde la perspectiva del kernel *un contenedor es un proceso con restricciones. En lugar de correr un binario, corre una **imagen**.*

- Brinda la característica de **inmutabilidad** y **reutilización**, permite que los contenedores sean ejecutados en múltiples sistemas.
- Bundle de archivos que pueden ser administrados por un sistema de control de versión (.tar)
- Necesitan ser alojadas de manera local, /var/lib/containers
- Son consumidas de un lugar centralizado público o privado, **image registry**
 - quay.io
 - DockerHub
 - Google Container Registry
 - Amazon Elastic Container Registry

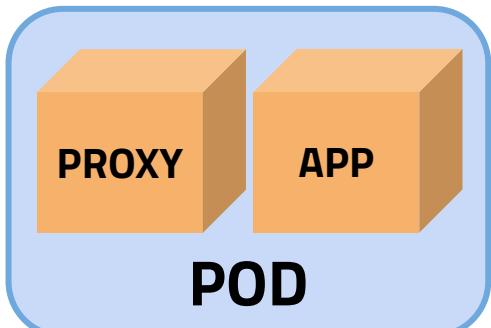


Tecnología de Contenedores

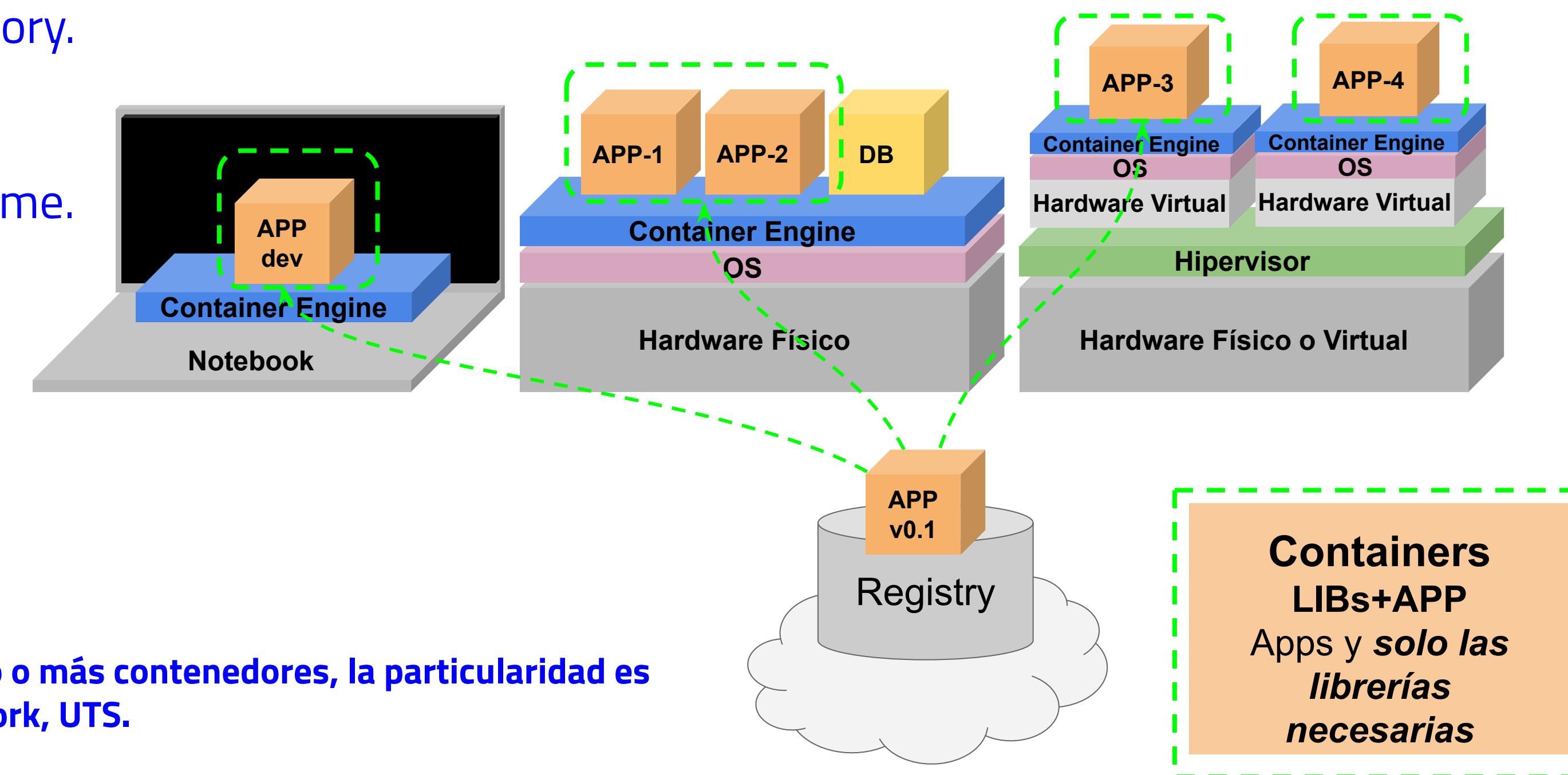
Arquitectura de Contenedores

7 Niveles de aislación

1. **IPC**, Communicate over share memory.
2. **Network**, Network device, stacks, ports.
3. **UTS**, Hostname and NIS domain name.
4. **Cgroups**, Resource protection.
5. **Mount** - Mount points, mapeo de directorios
6. **PID**, Process IDs
7. **User**, User and groups IDs.



Un pod puede contener uno o más contenedores, la particularidad es que comparten IPC, Network, UTS.



Tecnología de Contenedores

Podman

Podman es una herramienta open source para administrar contenedores, imágenes de contenedores e interactuar con imágenes del registro.



podman

Ventajas del uso de podman.

- OCI Compliance. Standard, community-driven, non-proprietary image format.
- Almacena de manera local las imágenes de los contenedores. Evitando arquitecturas cliente/servidor y demonios locales.
- Sigue las mismas especificaciones que Docker CLI.
- Es compatible con kubernetes y kubernetes puede usar podman para manejar contenedores.
- Librería libpod.

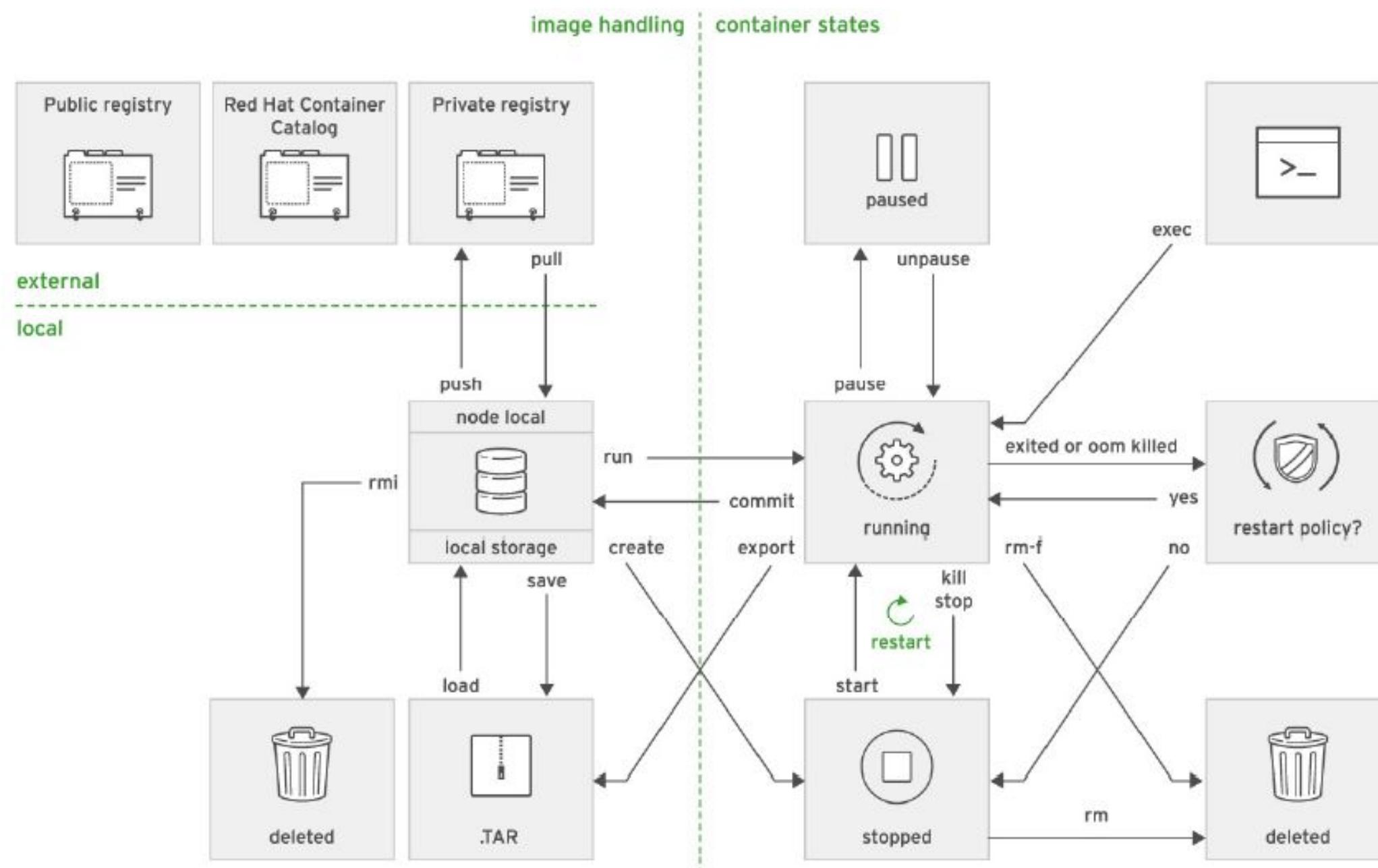
```
# Red Hat  
$ sudo yum -y install podman
```

```
# Fedora  
$ sudo dnf -y install podman
```

Tecnología de Contenedores

Podman CLI

Podman managing subcommands



- **Ejecutar un contenedor**

```
$ sudo podman run rhscl/httpd-24-rhel7
```

- **Verificar estado**

```
$ sudo podman ps -a
```

- **Ejecutar un comando desde el contenedor**

```
$ sudo podman exec http-small cat /etc/hostname
```

```
$ sudo podman exec -l cat /etc/hostname
```

- **Inspeccionar la metadata**

```
$ sudo podman inspect httpd-small
```

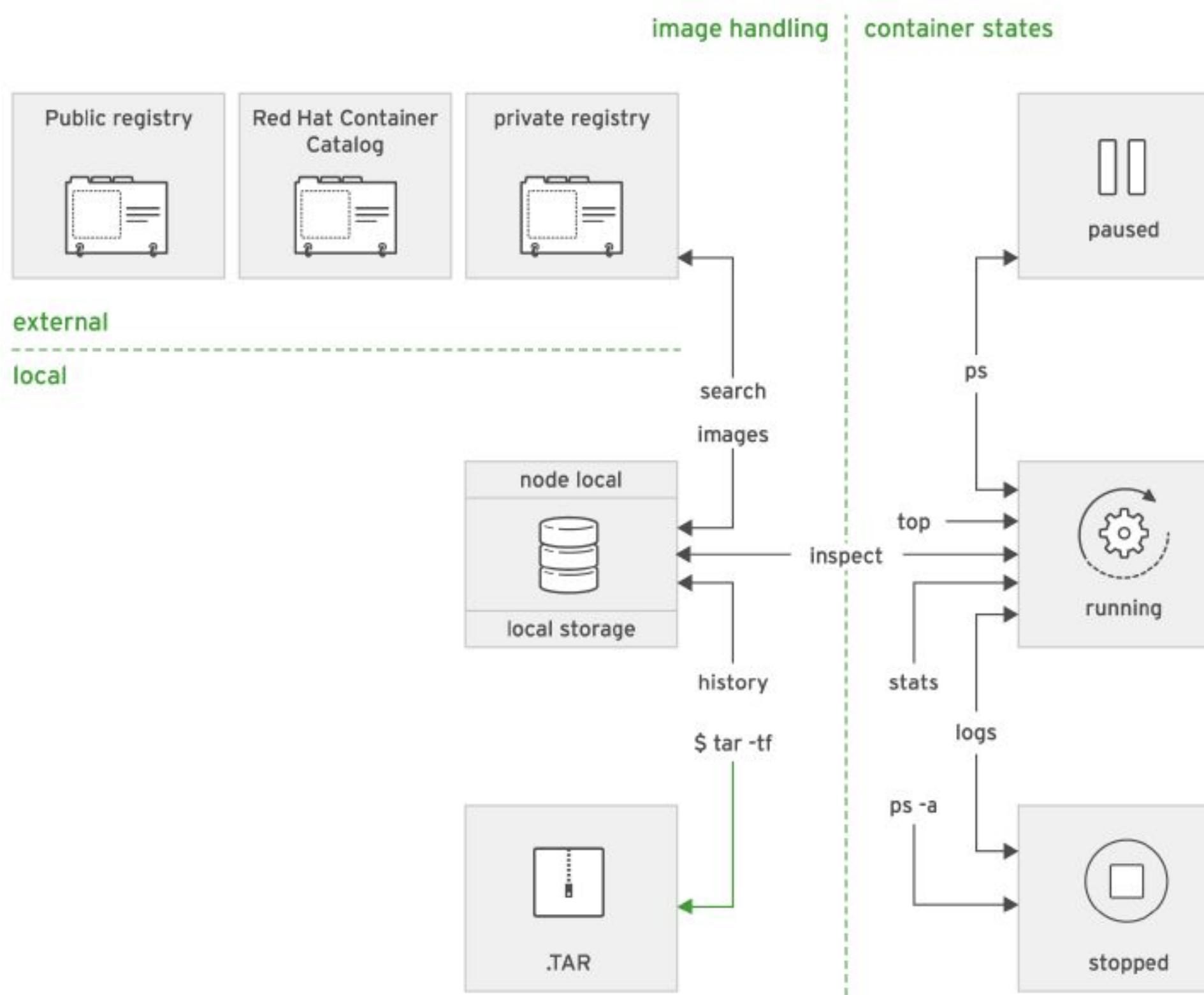
- **Dirección ip del contenedor**

```
$ sudo podman inspect -f '{{.NetworkSettings.IPAdrees}}' httpd-small
```

Tecnología de Contenedores

Podman CLI

Podman query subcommands



- **Detener un contenedor**

```
$ sudo podman stop httpd-small
```

```
$ sudo podman kill -s <SIGNAL ID> httpd-small
```

- **Remover un contenedor**

```
$ sudo podman rm httpd-small
```

- **Logs**

```
$ sudo podman logs httpd-small
```

- **Top de recursos**

```
$ sudo podman top
```

DEMO

GitLab

Docker

Tecnología de Contenedores

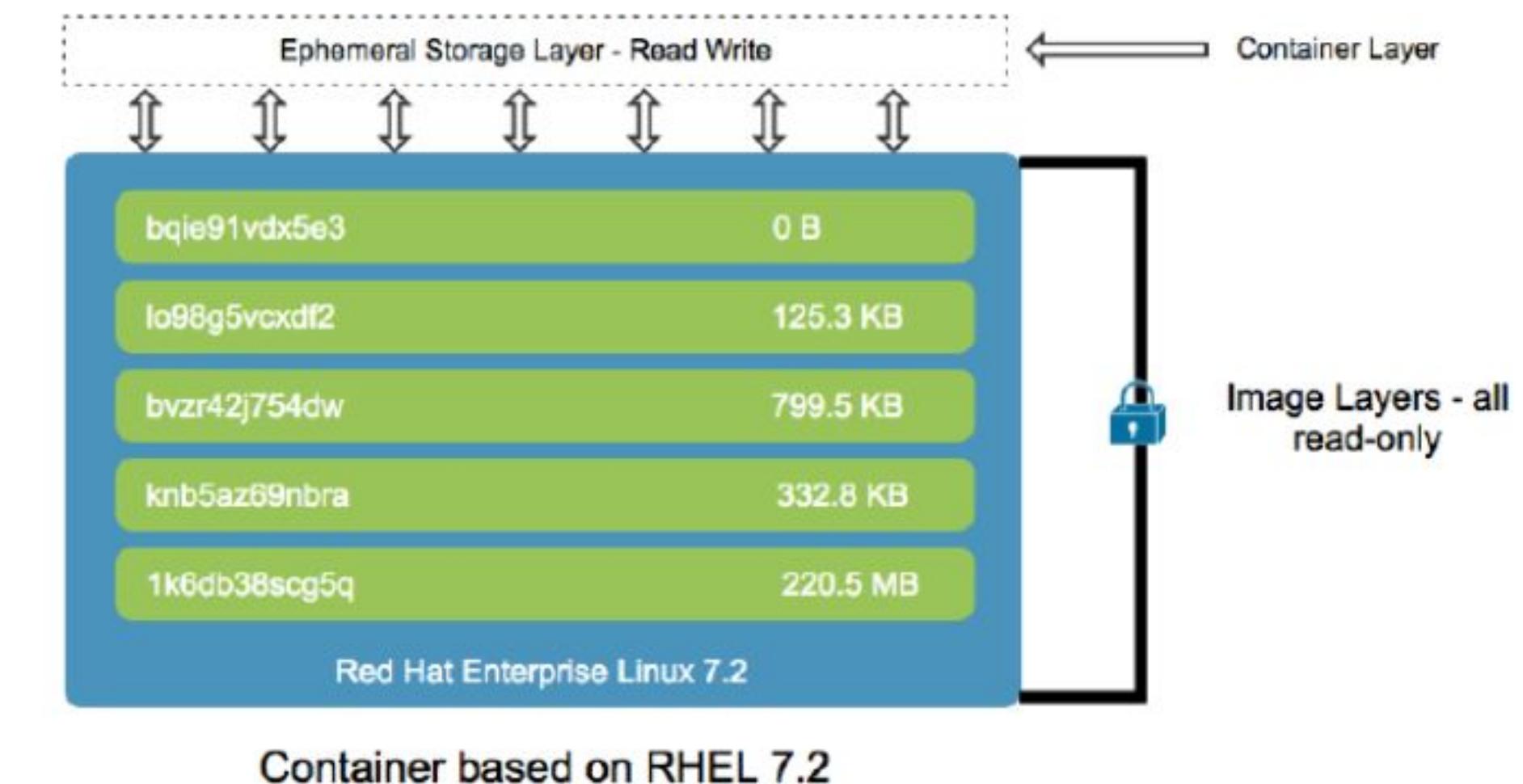
Storage efímero en Contenedores

Storage en contenedores

Se dice que el **storage** de los contenedores es **efímero**, no persiste tras un restart. Las *aplicaciones asumen que inician con un storage vacío*. Permite que los contenedores puedan ser iniciados y detenidos inesperadamente.

Images Containers

- Son inmutables y por capas (layers).
- Un layer superior anula el contenido del layer anterior.



- Path storage efímero

```
$ sudo podman inspect -l -f "{{.GraphDriver.Data.UpperDir}}")
```

Tecnología de Contenedores

Storage efímero en Contenedores

Proceso de creación

- Es ejecutado un contenedor, es creado un layer sobre la base imagen original.
- El storage adicional aloja archivos temporales (RW), logs, etc. Archivos efímeros.
 - **Stop/Start:** Si el contenedor es *detenido (stop)* y *vuelto a iniciar (start)* la **información efímera persiste**.
 - **Remove/Start:** Si el contenedor es *destruido (rm)* y *vuelvo a iniciar* uno nuevo la **información efímera es destruida**.

Misma image dos fs efimeros distintos

```
$ sudo podman run -d --name httpd-1 rhscl/httpd-24-rhel
$ sudo podman run -d --name httpd-2 rhscl/httpd-24-rhel
$ sudo podman inspect httpd-1 -f
"{{.GraphDriver.Data.UpperDir}}"
/var/lib/containers/storage/overlay/e5e0e11df03bc84c5d5f55
48578d524ac63c9ad46e0de60c7a7c2477b6258e8b/diff
$ sudo podman inspect httpd-2 -f
"{{.GraphDriver.Data.UpperDir}}"
/var/lib/containers/storage/overlay/00da1349de8cbd3ff41c6f
4c09ad92d8ae89a25472dd317dd1f61865fd63cf3b/diff
```

*Los contenedores **no deben persistir datos en el espacio efímero**, no solo porque no hay un control de cuanto storage alojar sino porque debido a los layers no es performante para aplicaciones de alto io.*

Tecnología de Contenedores

Storage efímero en Contenedores

Proceso de creación

- Es ejecutado un contenedor, es creado un layer sobre la base imagen original.
- El storage adicional aloja archivos temporales (RW), logs, etc. Archivos efímeros.
 - **Stop/Start:** Si el contenedor es *detenido (stop)* y *vuelto a iniciar (start)* la **información efímera persiste**.
 - **Remove/Start:** Si el contenedor es *destruido (rm)* y *vuelvo a iniciar* uno nuevo la **información efímera es destruida**.

Misma image dos fs efimeros distintos

```
$ sudo podman run -d --name httpd-1 rhscl/httpd-24-rhel
$ sudo podman run -d --name httpd-2 rhscl/httpd-24-rhel
$ sudo podman inspect httpd-1 -f
"{{.GraphDriver.Data.UpperDir}}"
/var/lib/containers/storage/overlay/e5e0e11df03bc84c5d5f55
48578d524ac63c9ad46e0de60c7a7c2477b6258e8b/diff
$ sudo podman inspect httpd-2 -f
"{{.GraphDriver.Data.UpperDir}}"
/var/lib/containers/storage/overlay/00da1349de8cbd3ff41c6f
4c09ad92d8ae89a25472dd317dd1f61865fd63cf3b/diff
```

*Los contenedores **no deben persistir datos en el espacio efímero**, no solo porque no hay un control de cuanto storage alojar sino porque debido a los layers no es performante para aplicaciones de alto io.*

Tecnología de Contenedores

Storage persistente en Contenedores

Host Path

Para la persistencia de datos tras restart de un contenedor, *podman* puede montar directorios de host dentro del contenedor y fuera del espacio de storage efímero.

Para que un contenedor acceda a un directorio en el host:

- UID y GUID
- Permisos necesarios.
- Podman utiliza SELinux, *container_file_t*, para proteger el acceso de otros procesos a los archivos del contenedor

Persistent Container Storage

```
$ sudo mkdir /var/dbfiles  
$ sudo chown -R 27:27 /var/dbfiles  
$ sudo semanage fcontext -a -t container_file_t  
'/var/dbfiles(/.*)?'  
$ sudo restorecon -Rv /var/dbfiles  
$ sudo podman run --name persist-db -d \  
  -v /var/dbfiles:/var/lib/mysql/data \  
  -e MYSQL_USER=user1 \  
  -e MYSQL_PASSWORD=pass1 \  
  -e MYSQL_DATABASE=stock \  
  -e MYSQL_ROOT_PASSWORD=t00r \  
  rhscl/mysql-57-rhel7
```

DEMO

GitLab

Docker

Tecnología de Contenedores

Container Network Interfaces (CNI)

*Cloud Native Computing Foundation (**CNCF**) apoya el proyecto open source Container Network Interfaces (**CNI**). El proyecto tiene como objetivo estandarizar la interfaz de red de los contenedores en entornos nativos de nube, como kubernetes y Openshift.*

Podman usa CNI para implementar una Software-Define Network (SDN) para contenedores en un host. Podman conecta cada contenedor a un virtual bridge y asigna a cada contenedor una ip privada.

La configuración de la red en el host se encuentra en el path

```
$ cat /etc/cni/net.d/87-podman-bridge.conflist
```



**CLOUD NATIVE
COMPUTING FOUNDATION**



<https://www.cncf.io/>

<https://github.com/containernetworking/cni>

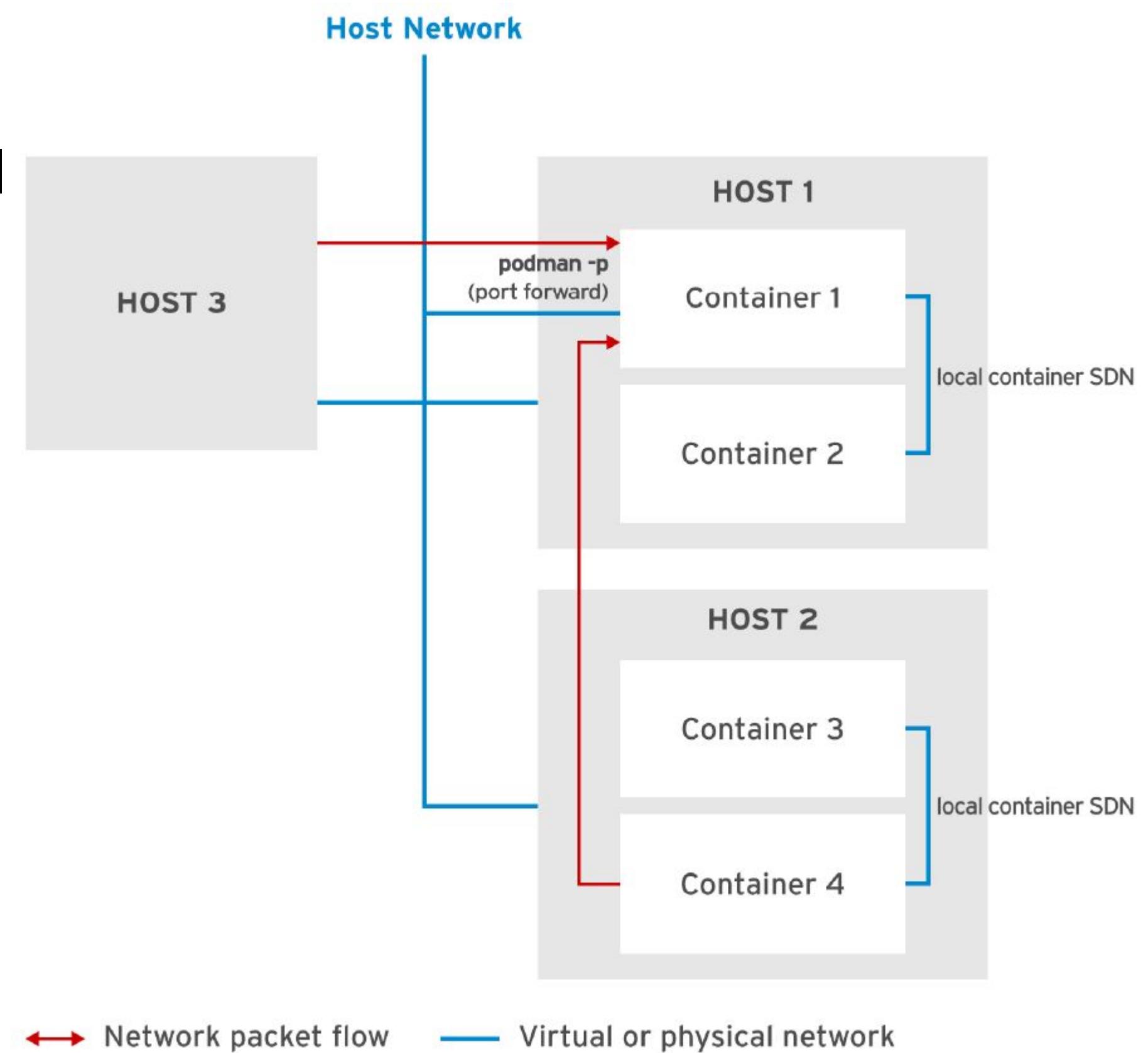
Tecnología de Contenedores

Container Network Interfaces (CNI)

Software-Defined Networking (SDN)

- Un contenedor es creado, es asignada una ip address del pool del host (10.88.0.0/16).
- El contenedor puede comunicarse libremente con los demás contenedores.
- La ip estará reservada para el contenedor mientras esté ejecutándose.
- Stop/Restart Contenedor, renueva ip address.
- Los contenedores creados en diferentes host no pueden comunicarse entre ellos por esta red.
- Para poder comunicarse desde la red de host, es necesario poder hacer un *Port Mapping (port forwarding)*

```
$ sudo podman run -d -p 8080:8080 --name httpd-3 httpd:2.4
```



<https://www.cncf.io/>

<https://github.com/containernetworking/cni>

Tecnología de Contenedores

Image Registries

Image Registries, servicio para la descarga de imágenes de contenedores. Permiten centralizar el creado y mantenimiento de las imágenes de contenedores. El host consulta el repositorio, luego copiar la imagen localmente y posteriormente ejecutar el contenedor de aplicación.

Podman permite operar con una colección de repositorios pudiendo ser publicos o privados.

Repositorio Publico de Red Hat	Repositorio Privado
<ul style="list-style-type: none">• Fuente confiable• Dependencias originales• Libre de vulnerabilidades.• Runtime protection.• RHEL Compatible• RH Support	<ul style="list-style-type: none">• Privacidad.• Restricciones legales.• Evita publicar imágenes de desarrollo.

Tecnología de Contenedores

Image Registries

El archivo de configuración donde se definen los registros es /etc/containers/registries.conf

```
[registries.search]
registries = ["registry.redhat.io", "quay.io"]
```

En el mismo archivo de configuración podemos definir registries inseguras. Es útil para ambientes de desarrollo.

```
[registries.insecure]
registries = ['localhost:5000']
```

- **Buscar en un registro**

```
$ sudo podman search [option] <term>
```

- **Autenticación**

```
$ sudo podman login -u <rhlogin> registry.connect.redhat.com
```

```
$ sudo podman logout registry.connect.redhat.com
```

- **Pull desde el registro al storage local**

```
$ sudo podman pull quay.io/nginx
```

- **Tags** (útiles cuando queremos mover imágenes)

```
$ sudo podman tag mysql devops/mysql
```

- **Push desde el storage local a un registro remoto**

```
$ sudo push devops/mysql
```

Tecnología de Contenedores

Dockerfiles

Un **Dockerfile** es un mecanismo para el creado automatizado de imágenes de contenedores.

Tres pasos para la construcción:

1. Working directory

Directorio de trabajo donde se alojan todos los archivos de configuración.

2. Dockerfile

Manifiesto de ejecución secuencial para la construcción de la imagen del contenedor. FROM primer instrucción.

3. ENTRYPOINT/CMD

- Exec form:

```
ENTRYPOINT ["command", "param1", "param2"]
```

```
CMD ["param1", "param2"]
```

- Shell form:

```
ENTRYPOINT command param1 param2
```

```
CMD param1 param2
```

Nota: Nunca se debe usar una combinación de las dos formas, siempre una u otra.

```
$cat Dockerfile
FROM centos
MAINTAINER Gonzalo Acosta <gonzalo.acosta@semperti.com>
ENV PORT 8080
RUN yum -y install httpd && \
    yum clean all
RUN sed -ri -e "/^Listen 80/c\Listen ${PORT}" \
/etc/httpd/conf/httpd.conf && \
    chown -R apache:apache /etc/httpd/logs/ && \
    chown -R apache:apache /run/httpd/
USER apache
EXPOSE ${PORT}
ADD about.html /var/www/html/
CMD ["httpd", "-D", "FOREGROUND"]
$ sudo podman build -t apache-custom .
```

Tecnología de Contenedores

Dockerfiles layers

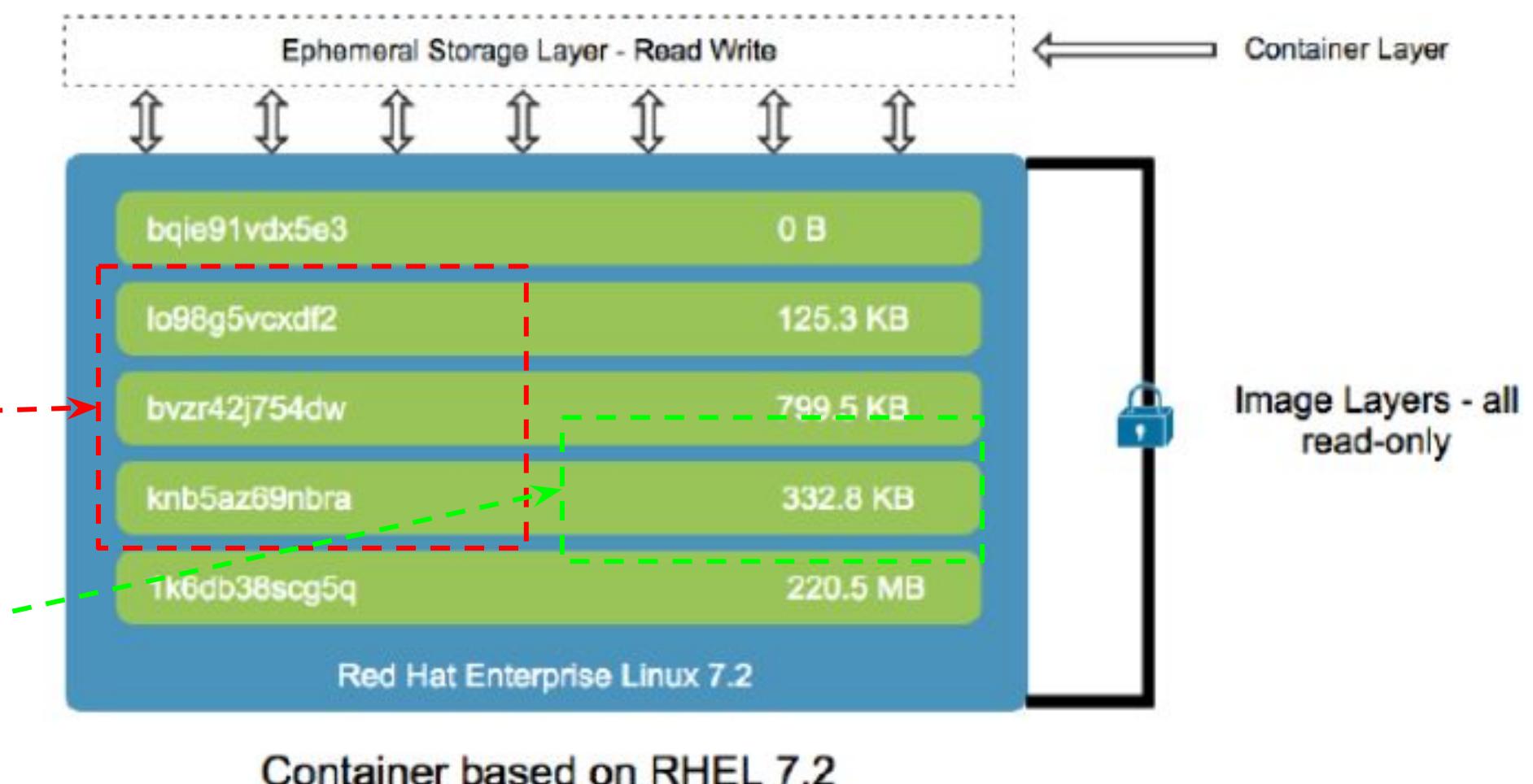
Cada instrucción en el Dockerfile crea un layer sobre la nueva imagen. Si tenemos un Dockerfile con muchas instrucciones este tendrá una gran cantidad de layers en la imagen resultado.

Por esto mismo y para tener la menor cantidad de layer en una imagen es recomendable reemplazar estas acciones consecutivas:

```
RUN yum --disablerepo=* --enablerepo="rhel-7-server-rpms"  
RUN yum update -y  
RUN yum install -y httpd
```

por una sola instrucción que genera un solo layer.

```
RUN yum --disablerepo=* --enablerepo="rhel-7-server-rpms" && \  
yum update -y && \  
yum install -y httpd
```



DEMO

GitLab

Docker

Día 2

Semperti

Orquestador de Contenedores

Kubernetes (CaaS)

Semperti

Container Orchestration

Kubernetes

Que es?



kubernetes

Container
Orchestrator

Workload
Placement

Infrastructure
Abstraction

Desired State

- <https://kubernetes.io/>
- <https://github.com/kubernetes/kubernetes> contribuyen **2491 personas!!!**

Container Orchestration

Kubernetes

Beneficios

Despliegues rápidos

Habilidad para absorber cambios
rápidamente

Habilidad para recuperarse
rápidamente

Oculta complejidad en el clúster



kubernetes

Container Orchestration

Kubernetes

Principios de Kubernetes

Desired State	Controllers	One Master
Configuración Declarativa	Control iterativo del estado	API Master

Container Orchestration

Kubernetes API

One Master

API Master

API Objects

Colección de primitivas que representan el estado del sistema

Habilita la configuración del estado

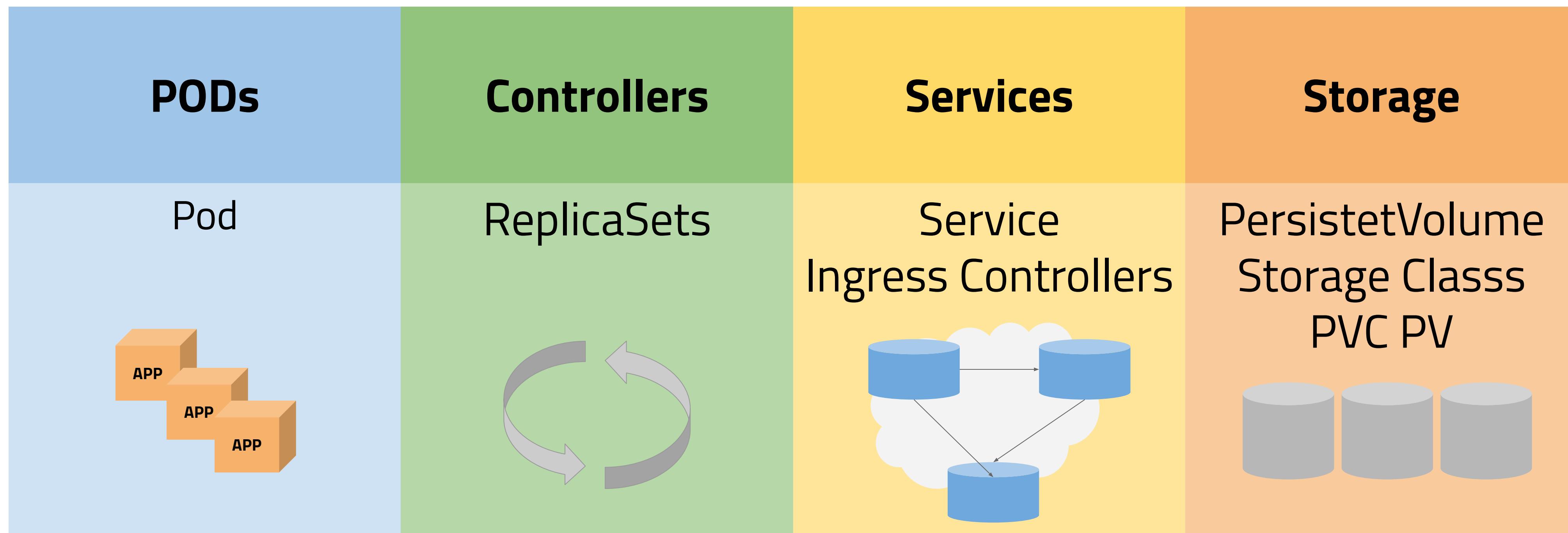
- Declarativo
- Imperativo

Kubernetes API Server

- RESTful API sobre HTTP usa JSON
- La única forma de interactuar con el cluster.
- La única forma en que Kubernetes interactúa con el cluster.
- Serializado y persistente

Container Orchestration

Kubernetes API Objects



Kubernetes API Objects

Container Orchestration

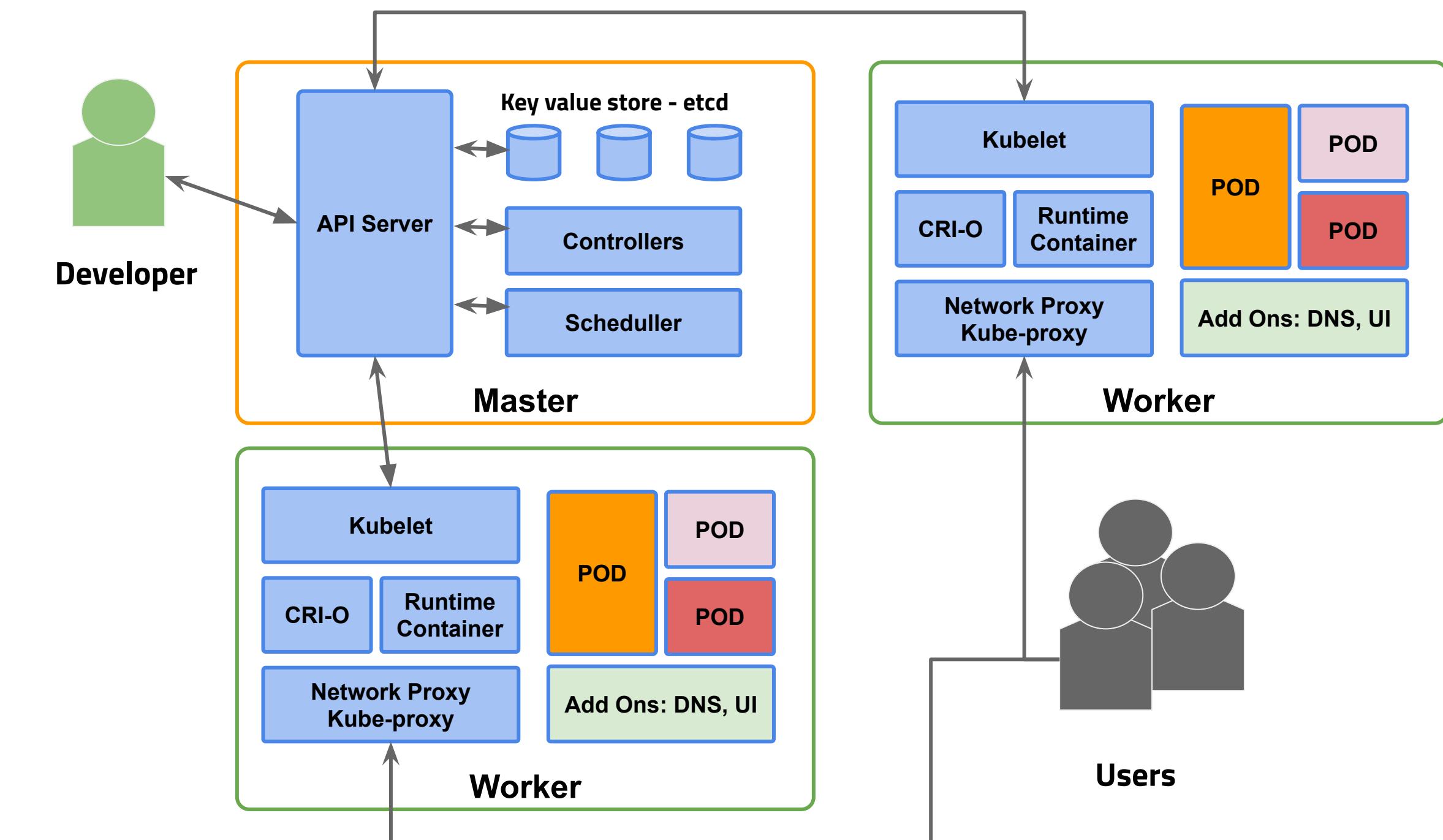
Kubernetes Arquitectura

- **Como funciona?.**

Kubernetes tiene una **arquitectura descentralizada** que no maneja tareas secuenciales. Funciona en base a un **modelo declarativo** e implementa el concepto de **estado deseado (desired state)**.

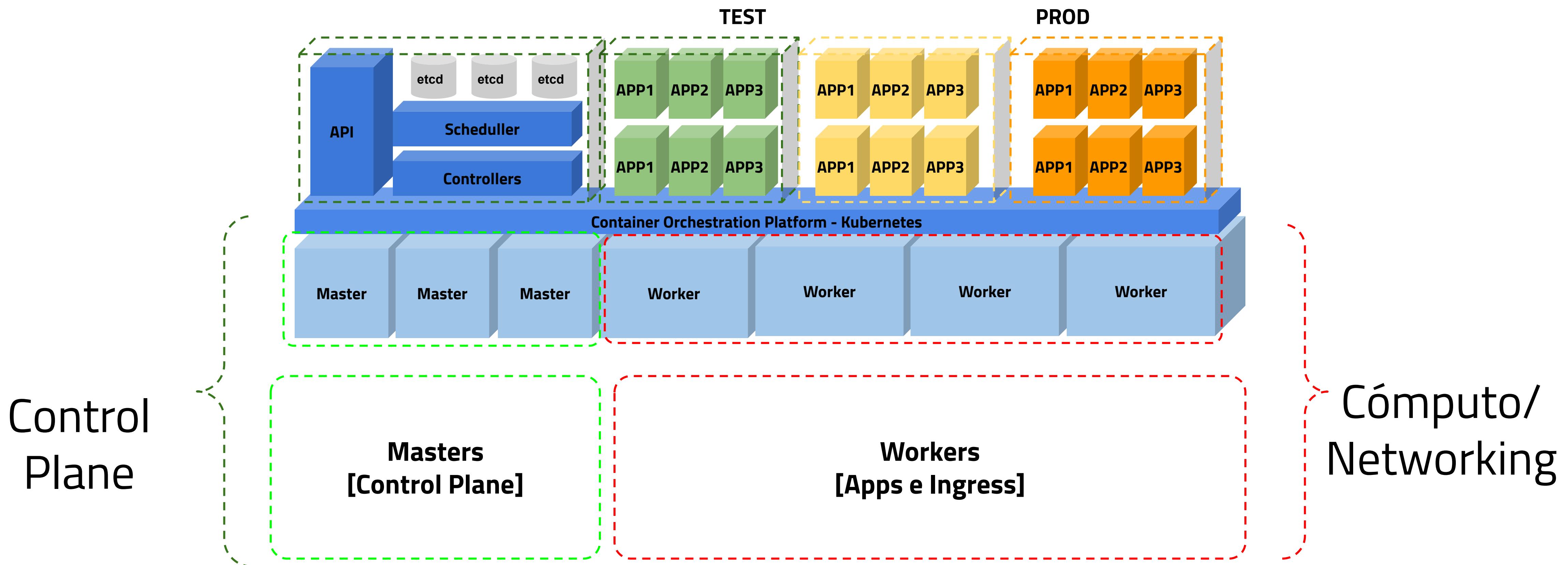
Proceso:

1. Administrador **crea el estado deseado** de una aplicación en un archivo de manifiesto.
2. El archivo es proporcionado a la **API de Kubernetes** mediante la CLI o UI.
3. Kubernetes **almacena** el estado deseado de la aplicación en una base de datos clave valor (etcd).
4. Kubernetes luego **implementa** el estado deseado en las aplicaciones que viven dentro del cluster.
5. Kubernetes **monitorea** continuamente el estado de los elementos del cluster para asegurarse que el estado no varía respecto al estado deseado



Container Orchestration

Kubernetes Arquitectura



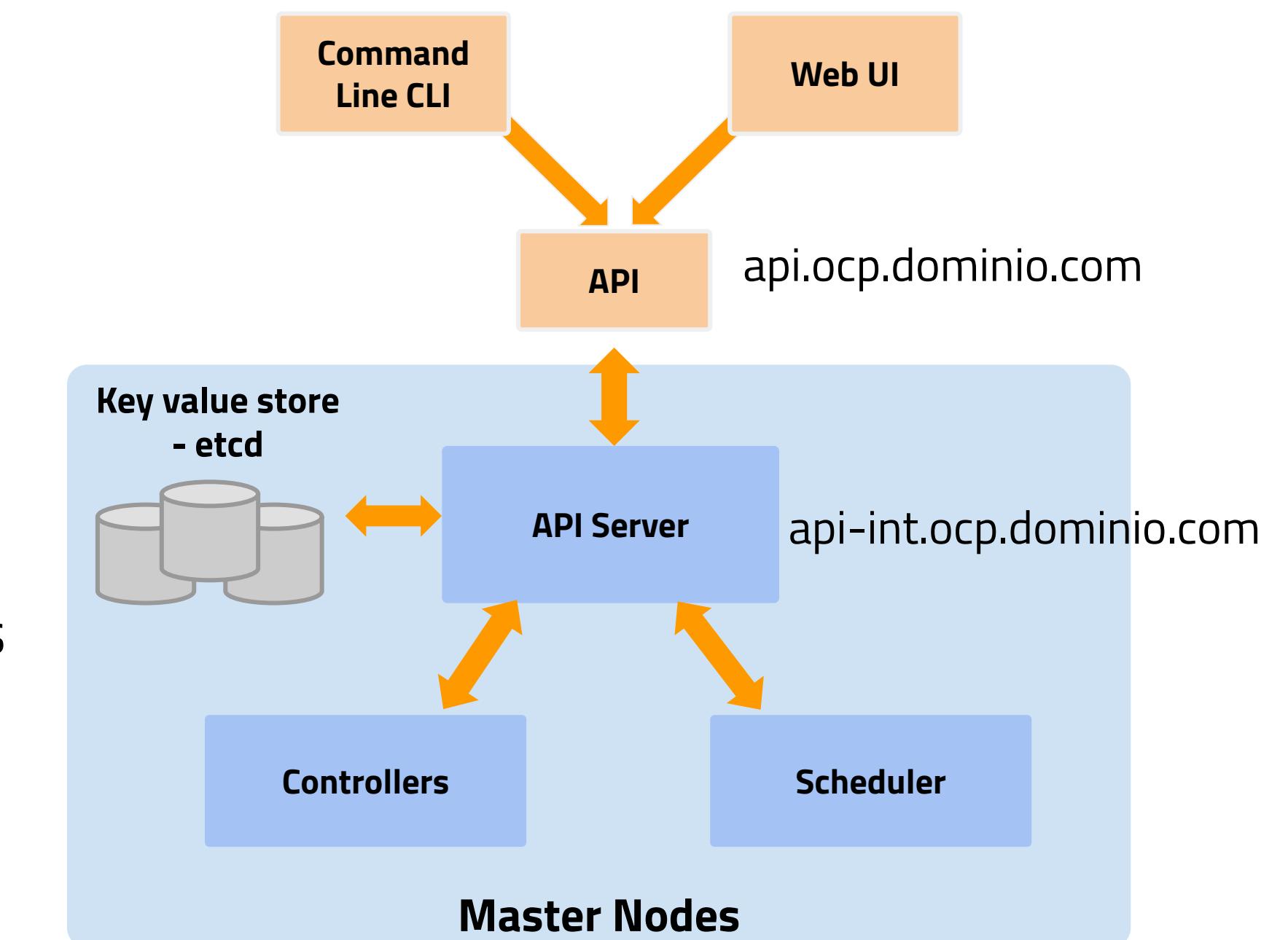
Container Orchestration

Kubernetes Master Nodes (Control Plane)

Que es un master nodes?

El kubernetes **master node** (Master) **recibe** las entradas que son ejecutadas desde la **CLI o UI vía API**.

- **API Server:** Front-End del Control Plane y es el único componente con el que vamos a interactuar. Tanto desde usuarios internos como componentes internos interactúan con la misma API.
- **Key-Value Store (etcd):** key-value store, etcd, **Almacena** configuración y el **estado deseado** del cluster. El master node consulta la base etcd y extraer los parámetros para el estado de los nodos, pods y contenedores.
- **Controller:** el rol es **obtener el estado deseado** desde el API Server. Este chequea el estado actual de los nodos y tiene la tarea de controlar y determinar si existen diferencias y las resuelven.
- **Scheduler:** El scheduler verifica los nuevos request en el API Server y asigna un estado de salud a los nodos. Chequea la disponibilidad de los nodos y despliega pods en el nodo más conveniente.



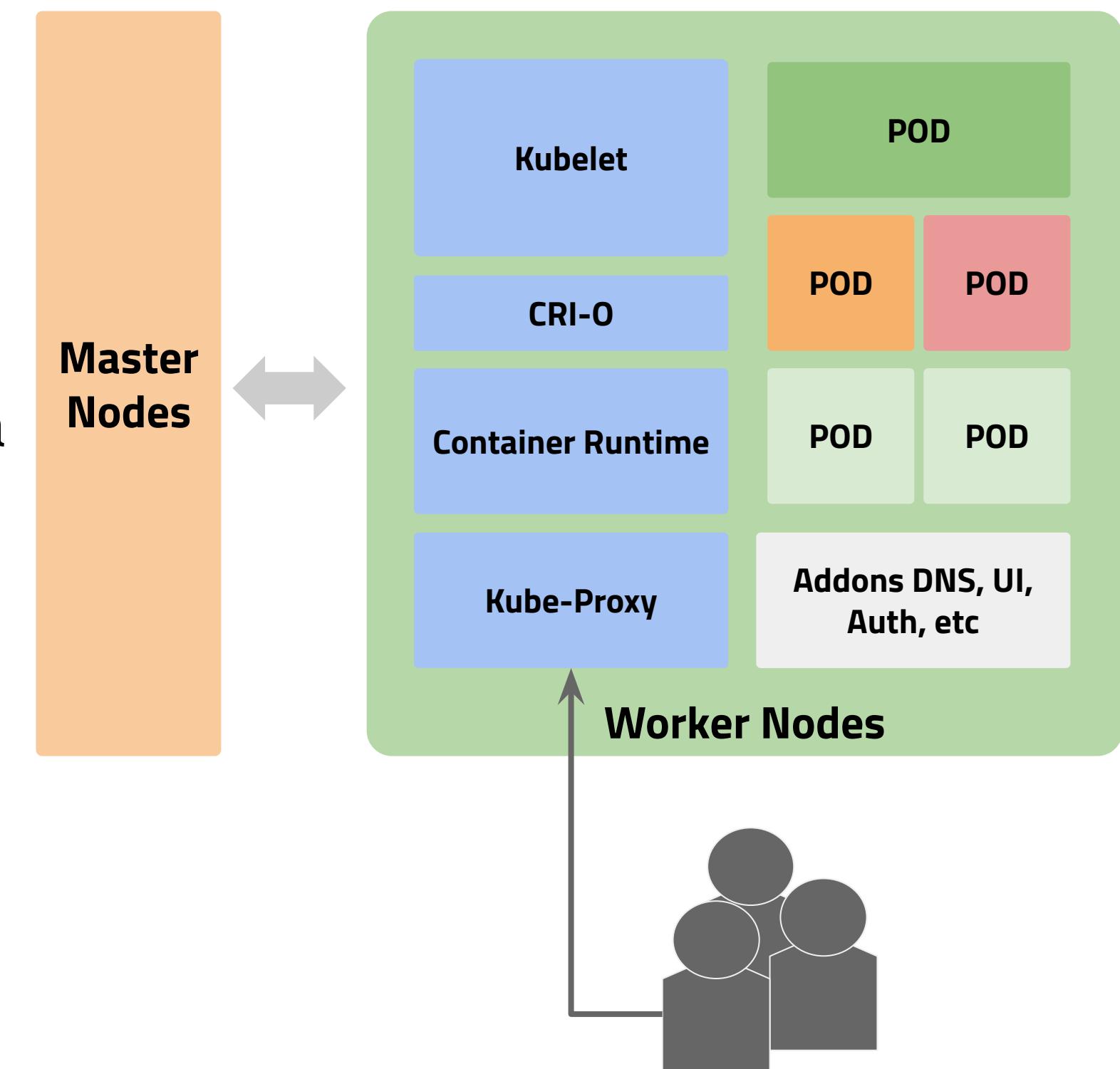
Container Orchestration

Kubernetes Worker Node

Que es un worker node?

Los worker nodes escuchan el API Server a la espera de nuevas tareas. Ejecutan la tarea asignada y luego reportan al master el estado.

- **Kubelet:** Principal agente corre en cada uno de los nodos. Reporta estado de cpu, memoria y storage al control plane. Recibe tareas enviadas desde el API Server, ejecuta y reporta el estado final. Monitorea. El Control Plane en base a la información de kubelet decide cambiar o no el estado deseado.
- **CRI-O y Container Runtime:** CRI-O interfaz para conectar distintos tipos de runtime de containers a K8S. Runtime tecnología para correr contenedores, docker, rkt, runc, etc. Realiza el pull, start, stop de los contenedores.
- **Kube Proxy:** Asegura que cada pod reciba una IP e implementa las reglas ruteo y balanceo.
- **PODs:** Es el minimo elemento a ser scheduleado por kubernetes.



Container Orchestration

Kubernetes POD

Que es un POD?

Un pod es el elemento más pequeño que será programado (scheduled) en Kubernetes.

Un pod sirve como wrapper del contenedor contenedor, el contenedor es quien posee el código de aplicación.

Basado en la disponibilidad de recursos el master node (scheduler) aloja el pod en un nodo específico. Coordina con el container runtime la ejecución del contenedor.

IMPORTANTE: en caso de que el pod inesperadamente no pueda realizar su tarea, Kubernetes no intenta arreglarlo, lo mata y levanta una nueva réplica en su lugar. El nuevo pod es una réplica a excepción de su DNS y la IP.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      volumeMounts:
        - mountPath: /usr/share/nginx/html
          name: www-volume
  volumes:
    - name: www-volume
      hostPath:
        path: /www
        type: Directory
```

Deployment

Scaling, updates and rollback

POD

Small Unit of Deployment in Kubernetes

Container

Application Code

Container Orchestration

Kubernetes Services

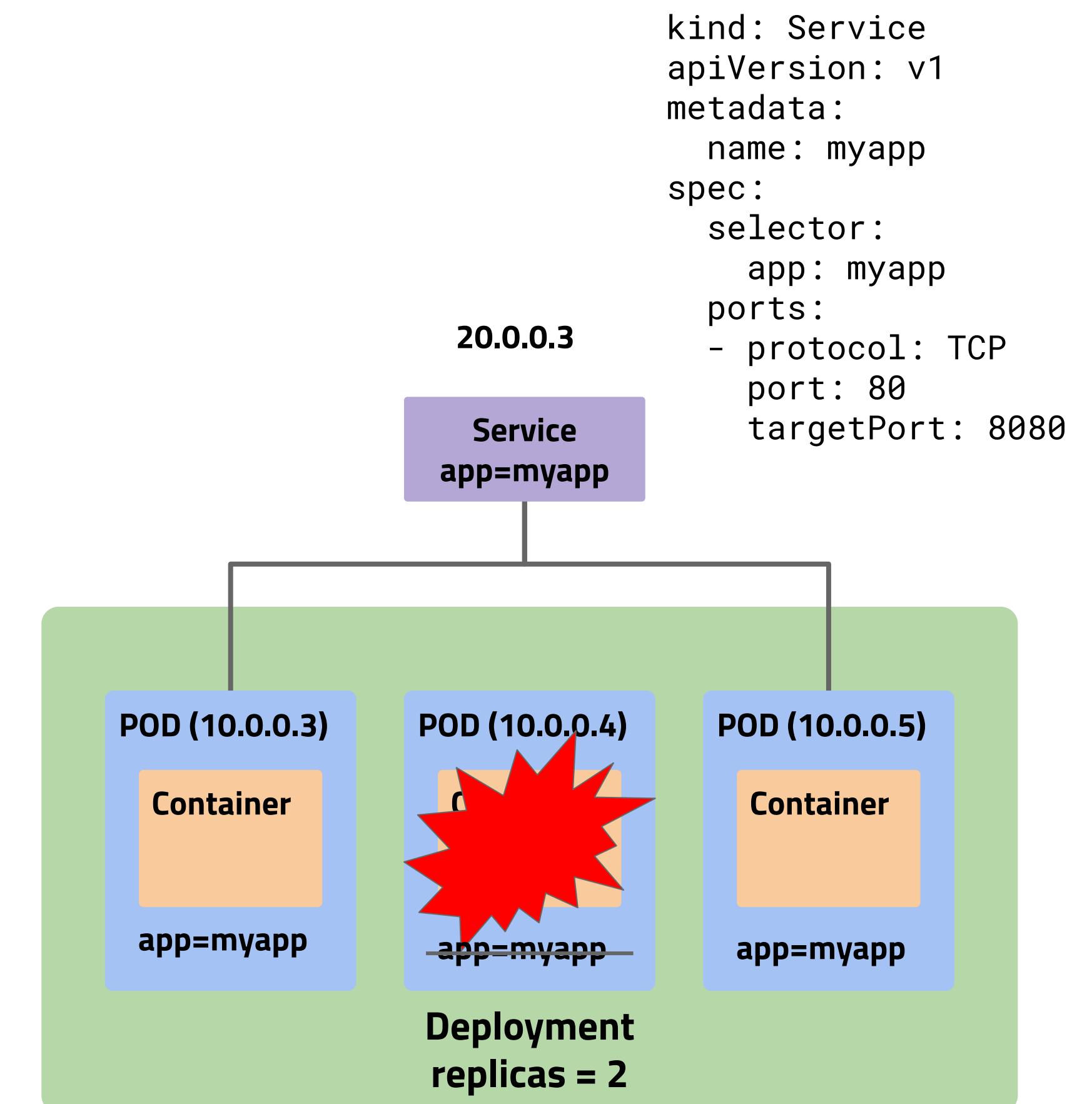
Que es un kubernetes services?

Uno de las principales características que ofrece Kubernetes es que los pods que no funcionan son reemplazados por otros automáticamente.

Al controlar el tráfico de ingreso y egreso, un servicio provee un solo punto de conexión de red estable (ip, dns y port). A través un servicio cualquier pod puede agregarse o eliminarse sin temor que la información básica de red sea modificada.

Como funciona?

Los pods están asociados con servicios a través de pares clave-valor llamado labels o selectores.

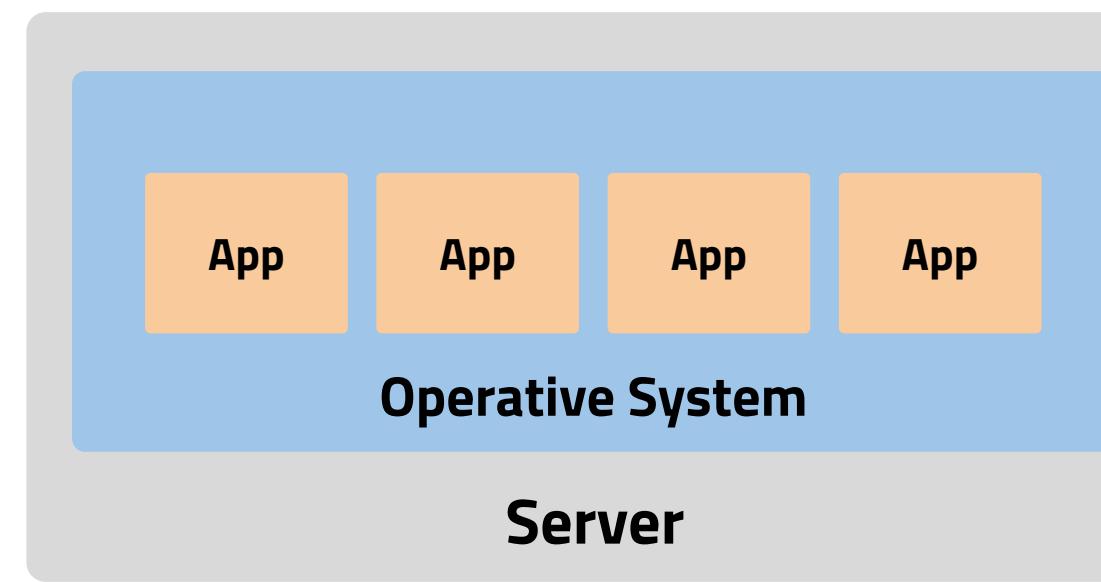


DEMO

GitLab

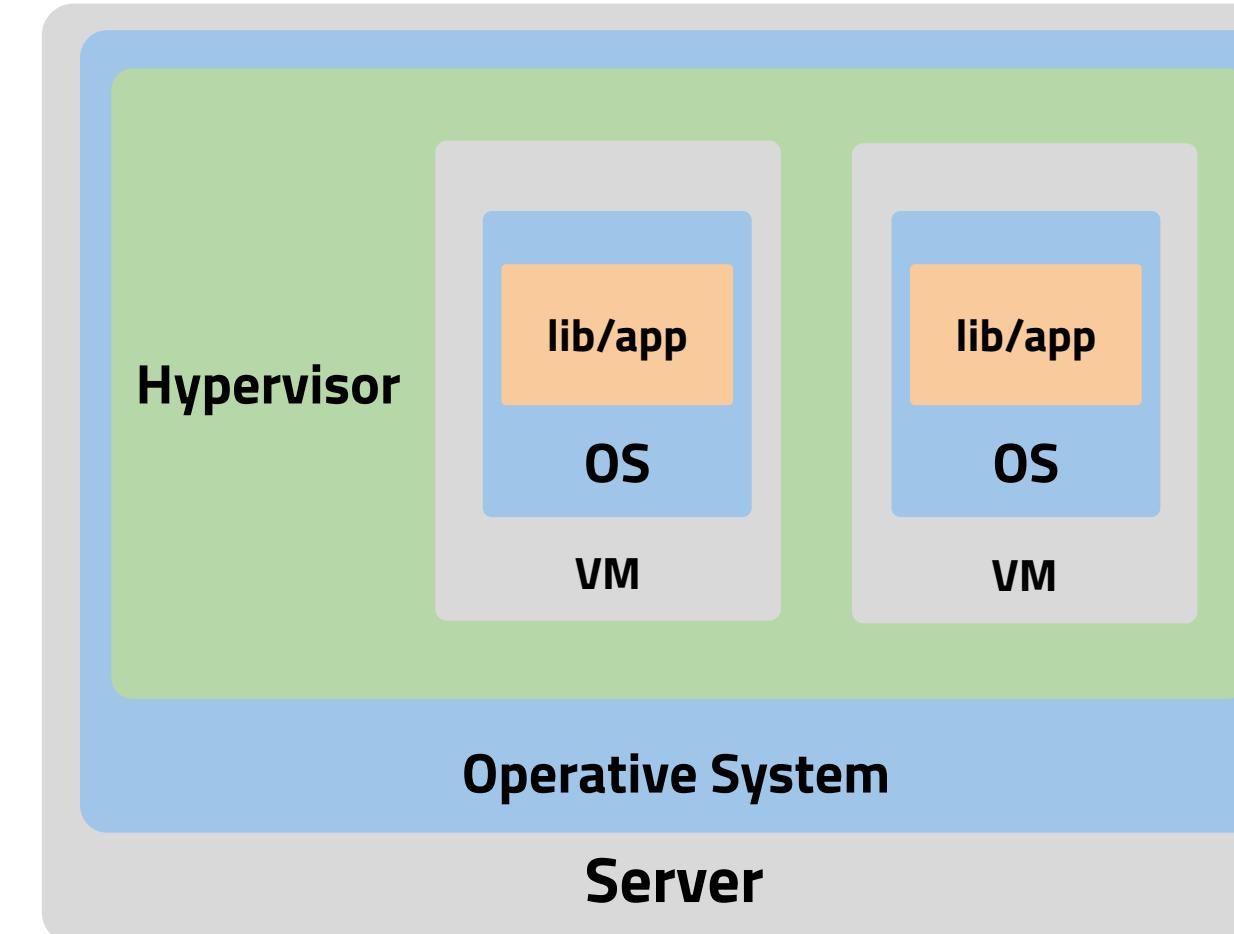
Container Orchestration

Container Deployment



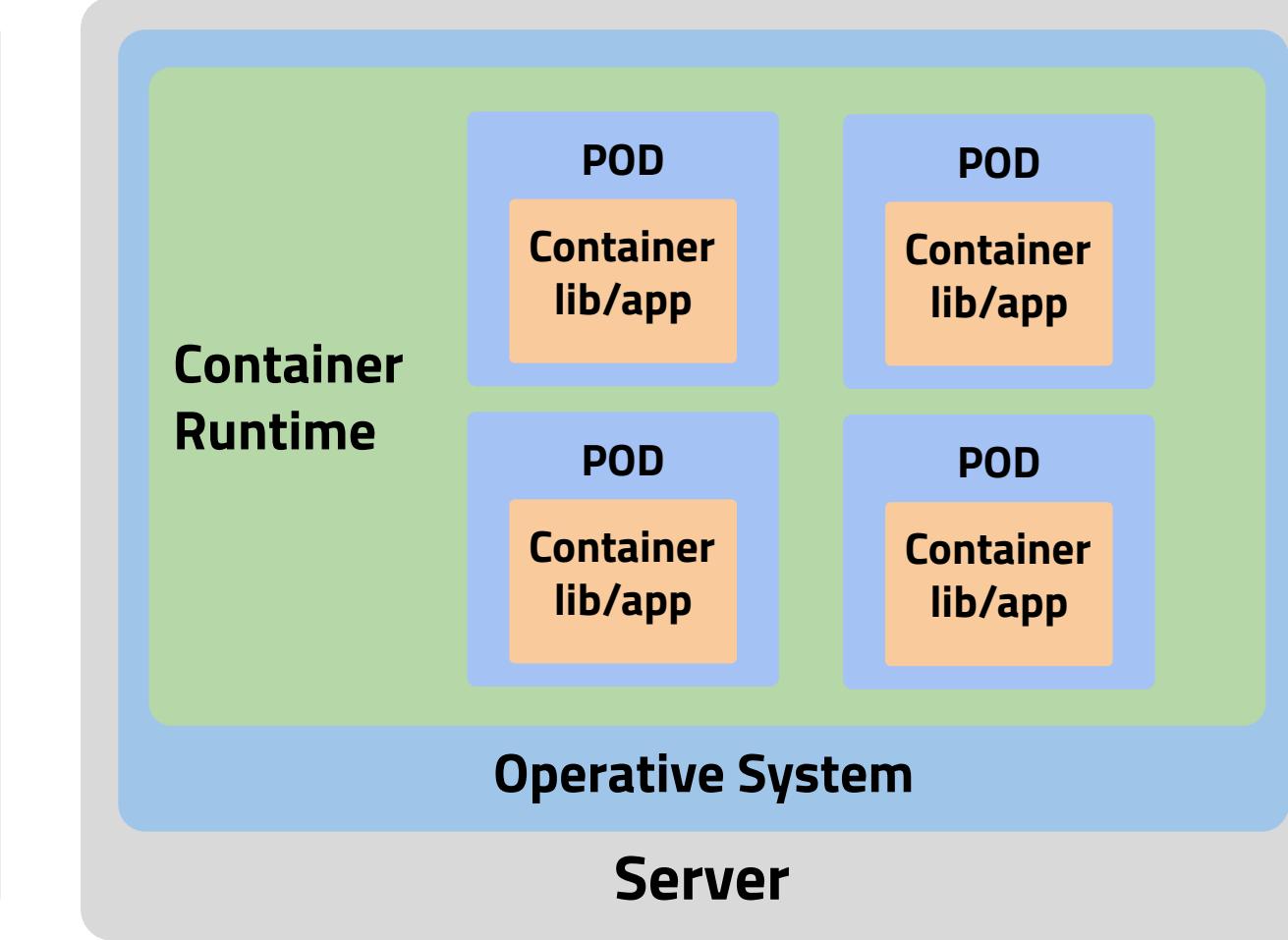
Traditional Deployment

Mismo recurso físico.
Las aplicación puede tomar más recursos de los necesarios, limitando el rendimiento de otras aplicaciones.



Virtualized Deployment

Ambientes aislados (VM) sobre una máquina fisica.
Permiten escalar rápidamente y distribuir los recursos.
VM con su propio SO



Container Deployment

Más flexible y eficiente.
Contenedores con su propia cantidad de memoria, cpu, archivos y espacio de procesos.
Multiples apps mismo SO. Apps independientes.

Container Orchestration

Kubernetes Deployment

Que es un kubernetes deployment?

Un deployment es un template que define el **estado deseado** para el despliegue de una aplicación.

Define:

- Container
- Imagen
- Replicas
- Labels
- Ports
- Resources.

Un deployment realiza el despliegue de los pods via **Replicas Sets**.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp
  labels:
    role: myapp
spec:
  replicas: 2
  selector:
    matchLabels:
      role: myapp
      tier: web
  template:
    metadata:
      labels:
        role: myapp
        tier: web
    spec:
      containers:
        - name: myapp
          image:
            gcr.io/google-samples/hello-app:1.0
          ports:
            - containerPort: 8080
```

Container Orchestration

Kubernetes ReplicationController y ReplicaSets

Que es son los ReplicaSets y ReplicationController?

Esencialmente cumplen la misma función, la de **mantener la cantidad de réplicas** del pod siempre activas.

Los replicaset ofrecen la posibilidad de usar selectores más avanzados.

```
...
spec:
  replicas: 3
  selector:
    matchExpressions:
      - {key: app, operator: In, values: [hellodev, hellotest,
        helloqa]}
      - {key: tier, operator: NotIn, values: [production]}
  template:
    metadata:
      ...
...
```

```
apiVersion: v1
kind: ReplicationController
metadata:
  name: hello
spec:
  replicas: 2
  selector:
    app: hello
  template:
    metadata:
      name: hello
      labels:
        app: hello
        tier: web
    spec:
      containers:
        - name: hello
          image:
            gcr.io/google-samples/hello-app:1.0
          ports:
            - containerPort: 8080
```

```
apiVersion: extension/v1beta1
kind: ReplicaSet
metadata:
  name: hello
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello
  template:
    metadata:
      labels:
        app: hello
        tier: web
    spec:
      containers:
        - name: hello
          image:
            gcr.io/google-samples/hello-app:1.0
          ports:
            - containerPort: 8080
```

Container Orchestration

Kubernetes Labels y Annotation

Que es son los labels y annotation?

Labels: pares de key/value que pueden ser asignados a un objeto dentro de kubernetes. **Organizan y agrupan** objetos. Ej, labels por centro de costo, proyecto, apps, nodo, routes, etc.

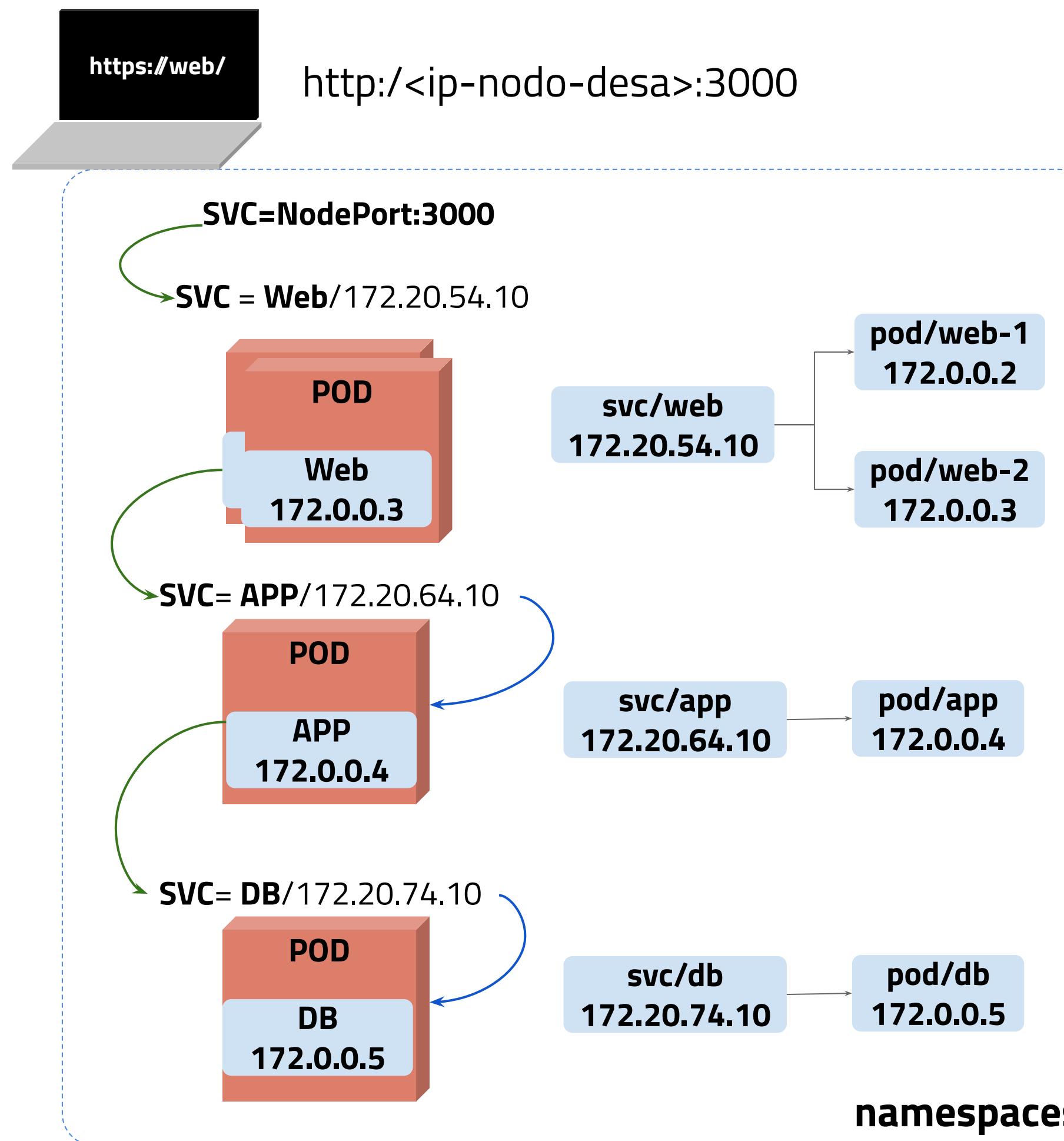
Annotation: claves/valor y pueden ser usados para **agregar metadata** al objeto. Almacen de informacion pero **no son utilizados para seleccionar o identificar** objetos.

IMPORTANTE: Es sumamente necesario tener una buena estrategia de labels para organizar cada uno de los ambientes.

```
apiVersion: v1
items:
- apiVersion: apps/v1
  kind: Deployment
  metadata:
    annotations:
      deployment.kubernetes.io/revision: "1"
    creationTimestamp: "2020-05-08T04:45:58Z"
    generation: 1
    labels:
      role: myapp
    name: myapp
    namespace: workshop
    resourceVersion: "17128907"
    selfLink:
      /apis/apps/v1/namespaces/workshop/deployments/myapp
    uid: a8902581-1cd9-4b3f-98e3-9936ef7c411b
  spec:
    progressDeadlineSeconds: 600
    replicas: 2
    revisionHistoryLimit: 10
    selector:
      matchLabels:
        role: myapp
        tier: web
      strategy:
        [...]
```

Container Orchestration

Kubernetes Namespaces



Cada objeto en kubernetes posee un id y un nombre que denota el tipo de recurso.

Un **namespace** es usado para mantener un **grupo de recursos de manera separada**. Cada nombre de objeto en un namespaces es único.

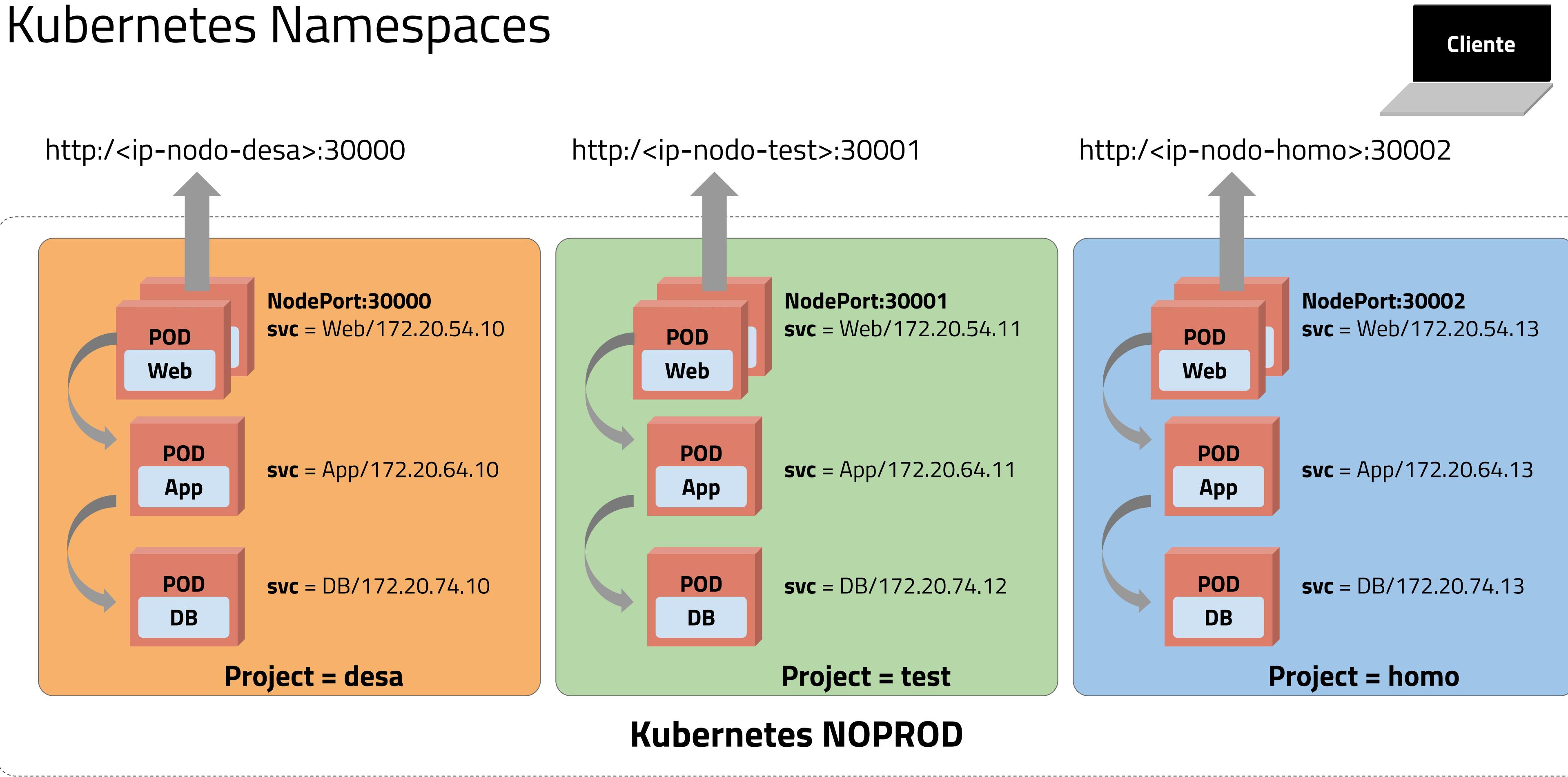
Namespaces = Agrupacion de objetos (pods, svc, deployment, route, cronjobs, jobs, etc).

Recomendaciones a la hora de usar proyectos.

- Ambientes pequeños = 1 namespaces = Ambiente.
- Ambientes grandes = 1 namespace por componente aplicativo.

Container Orchestration

Kubernetes Namespaces



Container Orchestration

Quotas and Resource Limits

Namespaces: DESA

- NetworkPolicies
- Quotas
- LimitRanges
- ServiceAccounts
- RoleBindings
- ClusterRoleBindings
- Configmaps
- Pod

Quotas Small

cpu=2
mem=2G

Network Policies

allow-from-same-namespace
allow-from-ingress-namespace

Namespaces: TEST

- NetworkPolicies
- Quotas
- LimitRanges
- ServiceAccounts
- RoleBindings
- ClusterRoleBindings
- Configmaps
- Pod

Quotas Medium

cpu=4
mem=4G

Network Policies

allow-from-same-namespace
allow-from-ingress-namespace

Namespaces: HOMO

- NetworkPolicies
- Quotas
- LimitRanges
- ServiceAccounts
- RoleBindings
- ClusterRoleBindings
- Configmaps
- Pod

Quotas Large

cpu=8
mem=8G

Network Policies

allow-from-same-namespace
allow-from-ingress-namespace

Kubernetes NO PROD core=36 mem=211

Container Orchestration

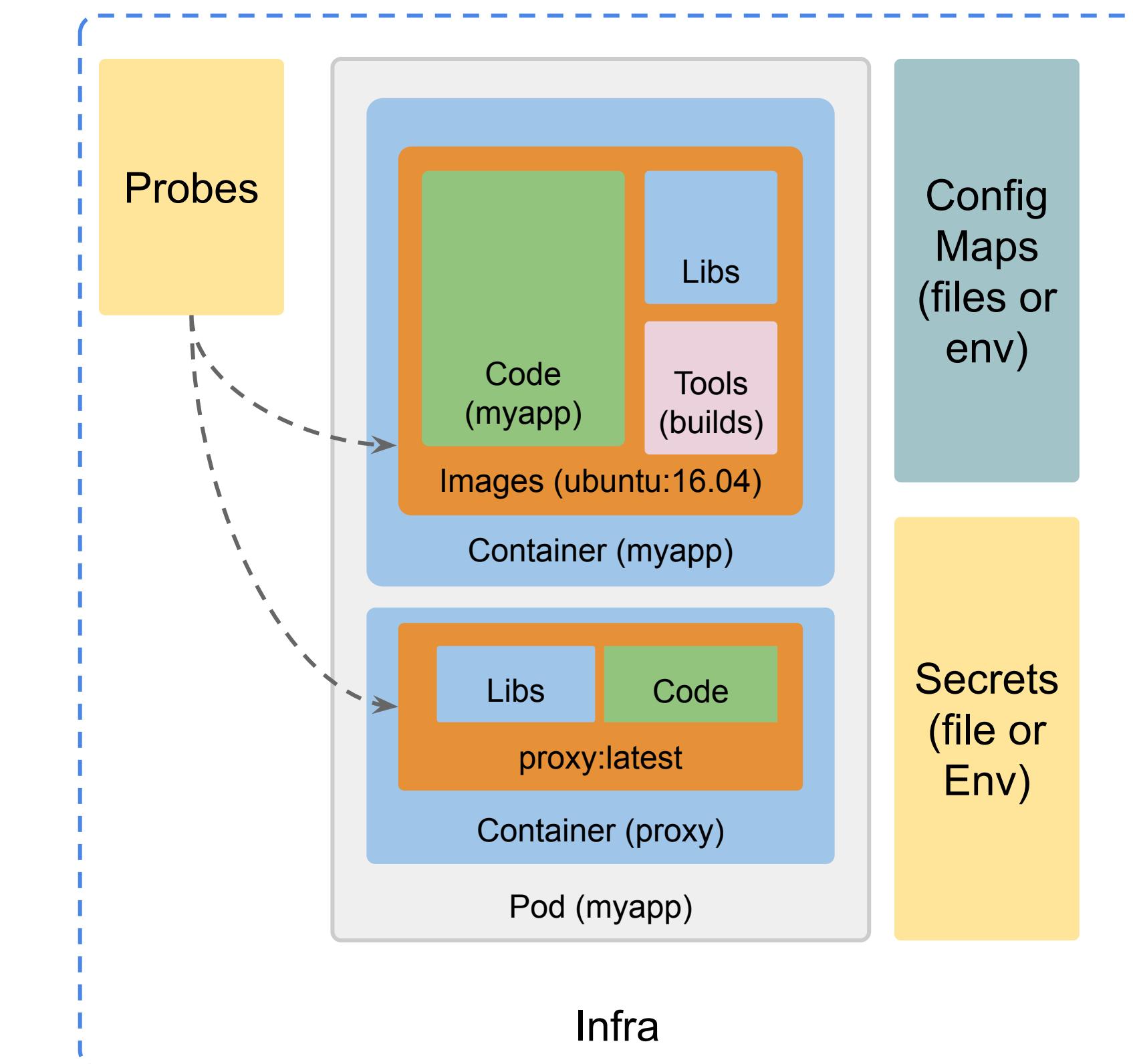
Kubernetes ConfigMaps, Secrets and Probes

- **Configuración y secretos.**

- Los parámetros de configuración deben realizarse en tiempo de ejecución y no alojarse en la imagen.
 - Variables de Entorno
 - Archivos de configuración en un mount point.

- **Readiness and Liveness Probes.**

- Debemos asegurarnos que las aplicaciones siempre funcionan.
- **Readiness Probes:** Verificar que una aplicación está lista para poder aceptar tráfico de servicio
- **Liveness Probes:** y una vez la aplicación esté recibiendo tráfico que su estado de salud sea correcto.



DEMO

GitLab

Día 3

Semperti

Orquestador de Contenedores

Openshift (Paas)

Semperti

OpenShift Container Platform

Requerimientos

Contenedores Requerimientos en Prod	Kubernetes	OpenShift
<ul style="list-style-type: none">• Una fácil comunicación entre el gran número de servicios.• Limitar recursos de aplicación, independientemente del número de contenedores que se ejecuten.• Responder a los picos de consumo incrementando y decrementando los contenedores que se ejecuten.• Reaccionar al deterioro de servicio.• Despliegues de manera gradual sobre los nuevos releases.	<ul style="list-style-type: none">• Service discovery and load balancing.• Horizontal scaling• Self-healing• Automated rollout• Secrets and configuration management• Operators	<ul style="list-style-type: none">• Routes• Multitenancy• Security• Metrics and Logging.• Integrated developer workflow.• Unified UI

Openshift Container Platform

Que es Openshift?

Openshift Container Platform (OCP) es una *plataforma para desarrollar y ejecutar aplicaciones en contenedores* basados en los estándares de la industria (OCI, CNI, etc). Con su base en **Kubernetes**, Openshift ofrece una plataforma **enterprise-ready**.

Ambientes de desarrollo completos **On-Demand**

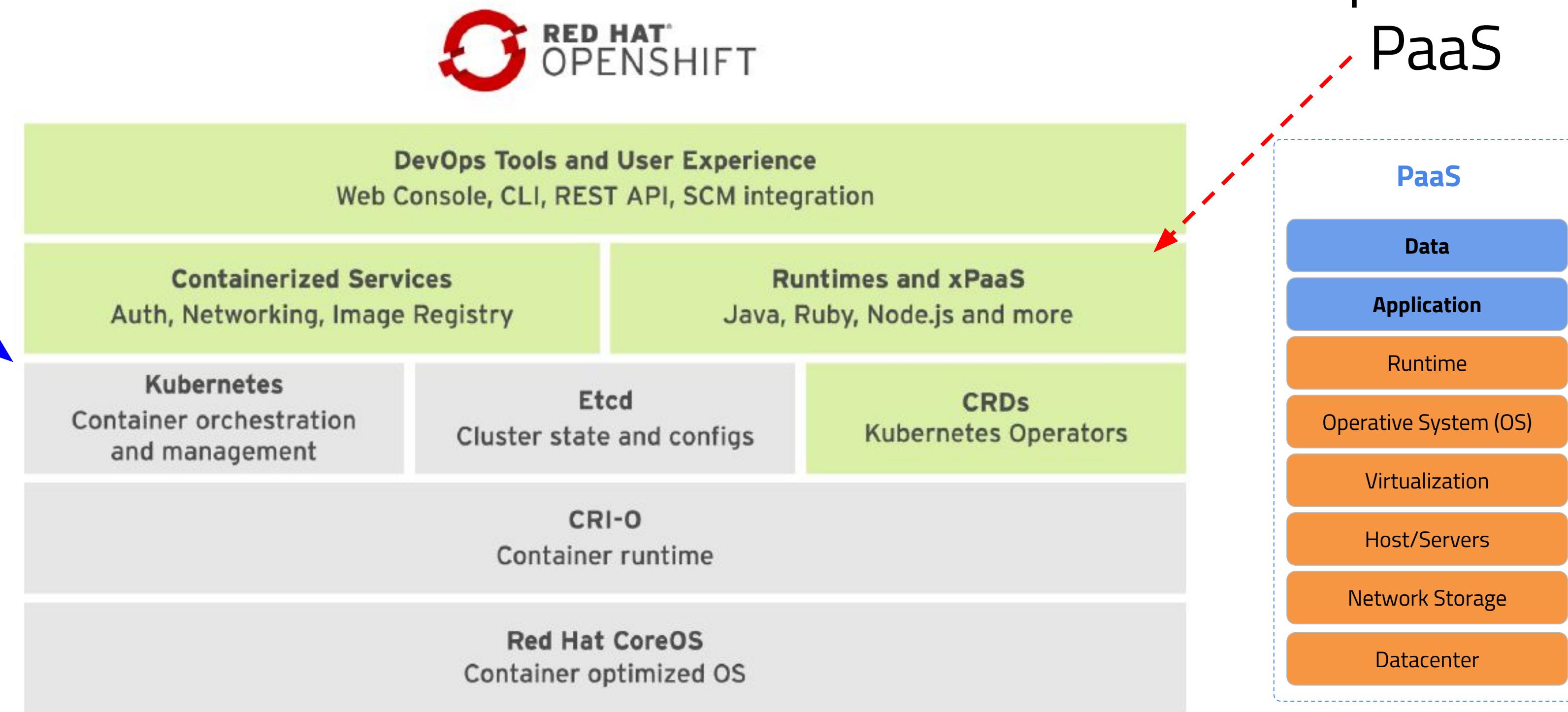
Lenguajes, frameworks, runtimes y base de datos

Herramientas para automatizar el ciclo de vida
(build, deploy y retire)

OpenShift Container Platform

Roles Arquitectura

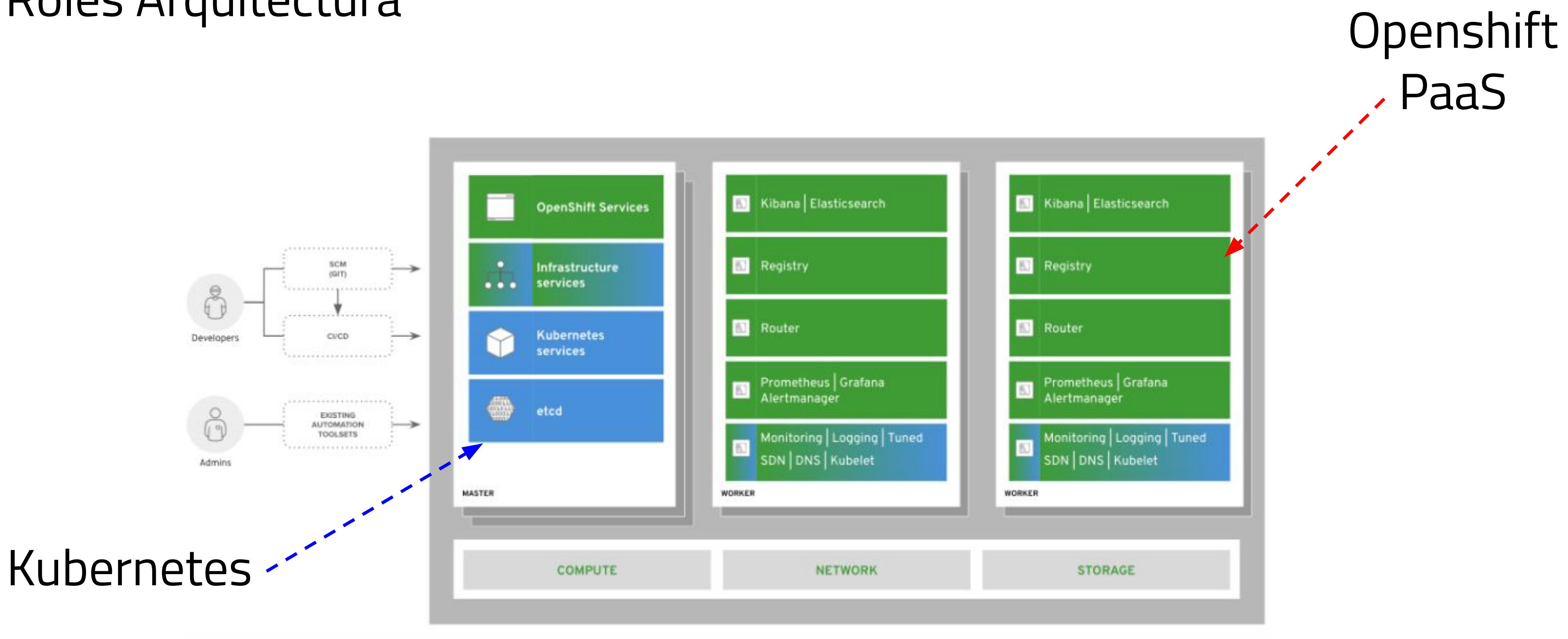
Kubernetes
CaaS



OpenShift **extiende la API de Kubernetes** agregando más funcionalidades. Ofrece *Out Of The Box* toda una **suite de herramientas** que facilitan la adopción de la cultura de desarrollo de aplicaciones basadas en contenedores.

OpenShift Container Platform

Roles Arquitectura



OpenShift Kubernetes Engine, los usuarios pueden experimentar con OpenShift 4 Kubernetes Engine, no toda la plataforma de OpenShift completa. Solo el Core de Kubernetes con las funcionalidades. Esto permite poder hacer uso de las imágenes de RHCOS como sistema inmutable y seguro.

OpenShift Container Platform

Infraestructura Soportada

- OpenShift está soportada sobre RHCOS y RHEL
- OpenShift 4 es una solución de cloud híbrida.
- Soporta proveedores de nube público y privados como tambien corre sobre bare metal.
- AWS, Azure, Google Compute.
- Red Hat Virtualization, Red Hat Openstack Platform.
- Está soportada la instalación sobre infraestructura preexistente (UPI Installation):
 - AWS
 - Azure
 - Google Compute
 - **VMWare vSphere**
 - Red Hat OpenStack Platform
 - IBM Z
 - IBM Power Systems, bare metal.
- Release futuros soportará más proveedores de Cloud.

4.4 Supported Providers



OpenShift Container Platform

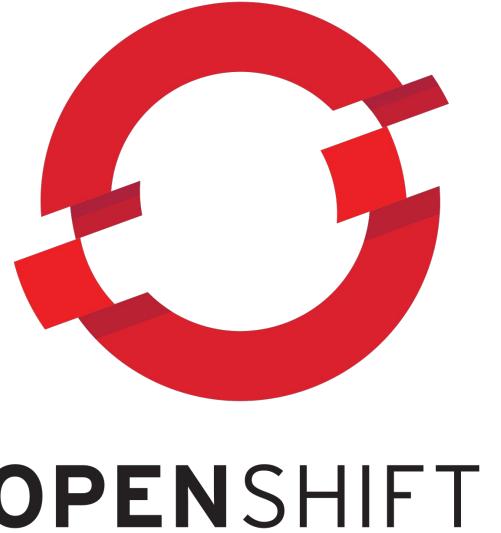
Terminología de Kubernetes



kubernetes

Kubernetes es un servicio de orquestación que simplifica el despliegue, administración y escalado de aplicaciones basadas en contenedores. Ventajas de kubernetes es el uso de nodos para asegurar la resiliencia y escalabilidad.

Termino	Definición
Node	Servidor que hostea aplicaciones en Kubernetes
Master Node	Servidor que administra el Control Plane en el cluster de Kubernetes. Servicios de API o Controllers
Worker Node	Compute Node, worker node que ejecuta la carga de trabajo del cluster. Los pods de aplicación son alojados en este nodo.
Resource	Un recurso contiene la configuración y el estado actual del componente que administrará Kubernetes.
Controller	Un controller es un proceso de Kubernetes que observa los recursos y realiza cambios para mantener el estado deseado de la definición del recurso.
Label	es un par clave valor que son asignados a cualquier recurso dentro de Kubernetes. Los selectores son utilizados para realizar filtros de recursos ya sea para scheduling u operación.
Namespaces	Agrupación lógica de los recursos y procesos de Kubernetes.
Operators	Los Operadores de Kubernetes son plugins, que nos permiten extender la funcionalidad de la api de Kubernetes.



Openshift Container Platform

Terminología de Openshift

Red Hat Openshift Container Platform es un set componentes y servicios construidos sobre *Red Hat CoreOS* y *Kubernetes*. Openshift agrega a la capa de CaaS que ofrece Kubernetes la capa de PaaS, con toda una suite de herramientas que nos permiten una mejor administración remota, incrementar la seguridad, monitoreo, logging, auditoría, self-service y mejores interfaces para los desarrolladores

Termino	Definición
Infra Node	Un nodo similar a un worker node de Kubernetes pero con el fin de alojar servicios de infraestructura como Logging, Metrics, Registry, Routing .
Console	Una web UI mejorada para la administración y el desarrollo. Portal de Self-Service integrando monitoreo, logging, eventos.
Project	Es una extensión para Openshift de los namespaces de Kubernetes. Permite la definición de usuario (User Access Control) para recursos.

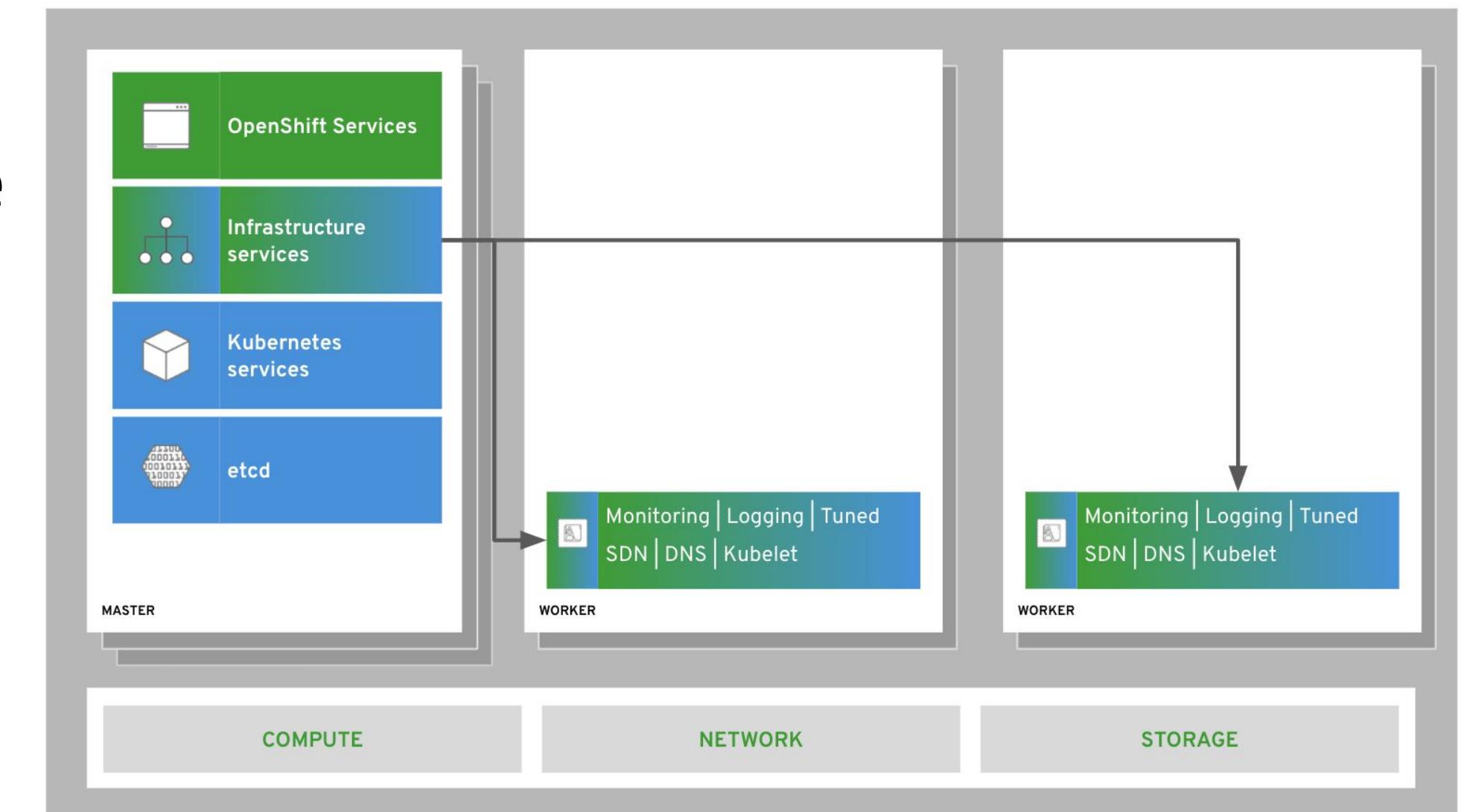
Openshift v4

- CoreOS como sistema mandatorio para los nodos masters y para los nodos workers tanto CoreOS como RHEL.
- Nuevo proceso de instalación.
- Portales de administración y desarrollo mejorados.
- Rediseño del ciclo de vida de aplicaciones, tanto para el cluster como para las aplicaciones de usuario.
- Operator SDK, para la construcción de nuestros propios Operadores de Kubernetes.

OpenShift Container Platform

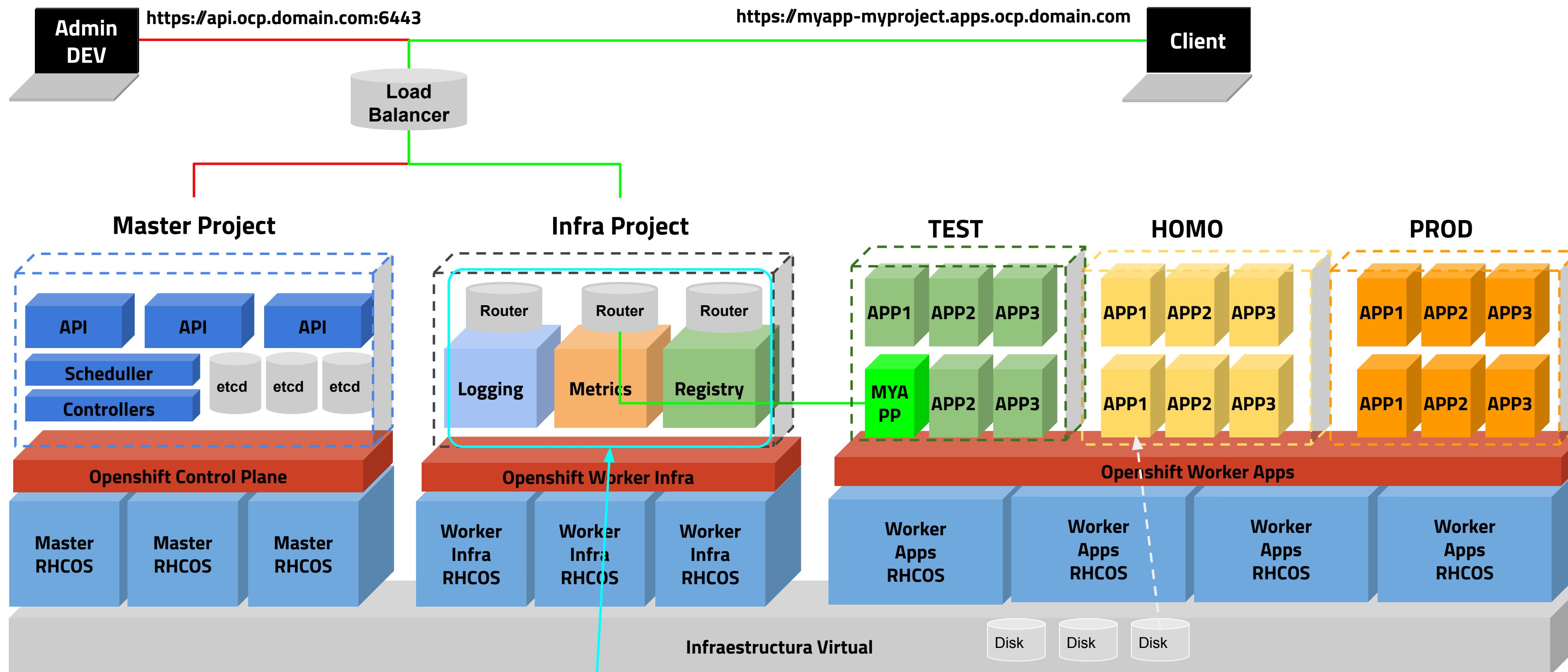
Infra Nodes

- Infra Services in Worker Nodes
- Los nodos master y los nodos worker trabajan en colaboración.
- Los dos tipos de nodos incluyen servicios de kubernetes y de openshift.
- Servicios:
 - Monitoring
 - Logging
 - OS tuning (tuned)
 - Software Define Networking (SDN)
 - Kubelet (OpenShift node process)



OpenShift Container Platform

Infraestructura

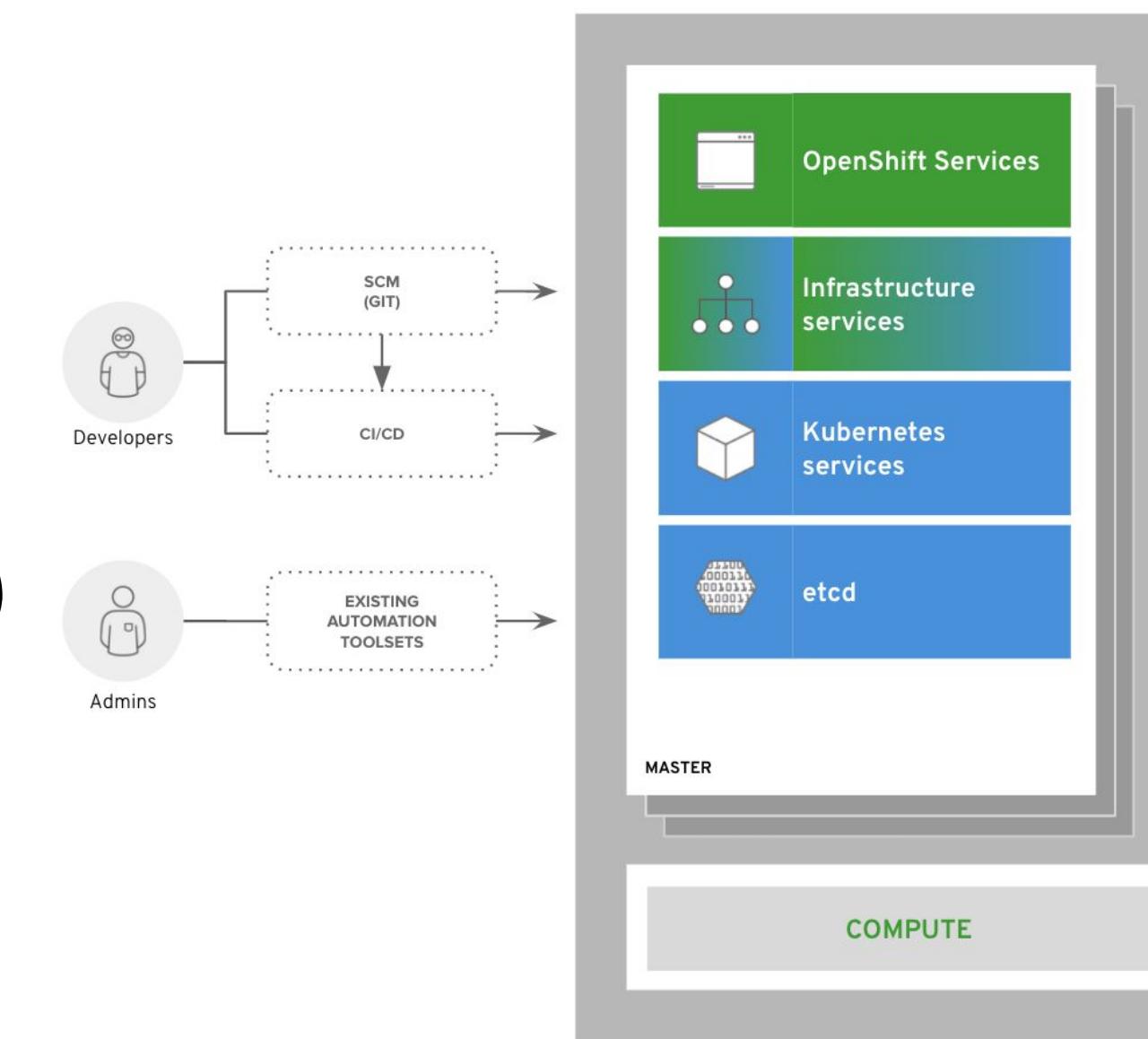


Nodos Worker que alojan
servicios de registry,
logging, metrics, router, etc.

OpenShift Container Platform

Acceso

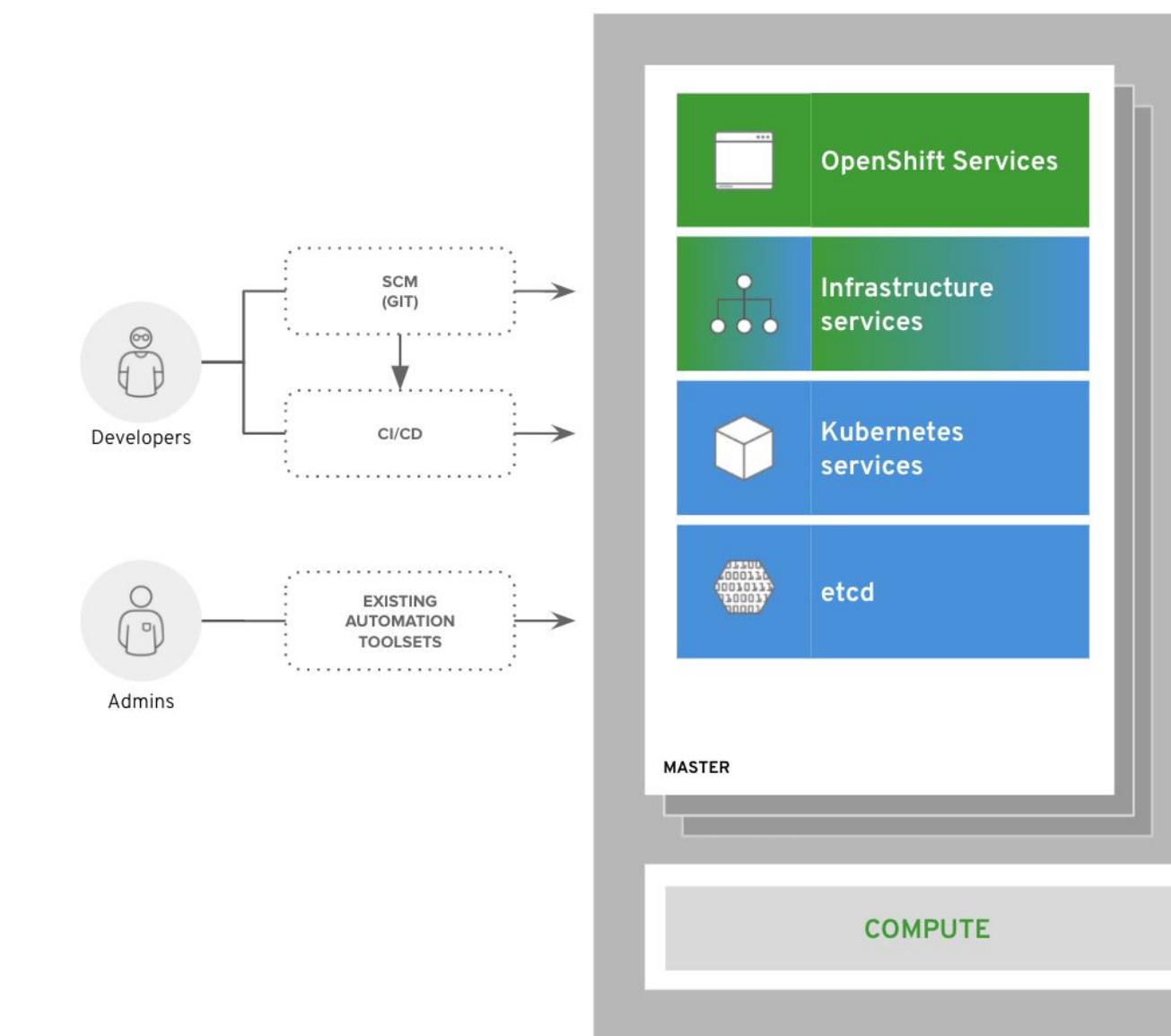
- Todos los usuarios acceden a Openshift a través de la misma interfaz estándar.
 - Web UI, CLI, integración con IDEs, todos son autenticados y autorizados vía RBAC a través del servicio de la API.
 - Usuarios no necesitan acceder a los nodos de Openshift (no acceso por ssh, solo admin task)
 - Integración con sistemas de CI/CD a través de interfaces (Jenkins, Tekton)
 - Openshift desplegado sobre RHCOS que habilita el uso de la nueva generación de sistema de administración y herramientas de monitoreo.
 - OpenShift implementado en RHEL permite el uso de herramientas de administración y monitoreo de sistemas existentes



OpenShift Container Platform

API and Authentication

- Los nodos master proveen un único endpoint de acceso API Service y todas las herramientas del sistema interactúan con él.
- API Service incluida en OpenShift y Kubernetes
- Todo request administrativo requiere que pase a través del servicio de la API.
- Todo request a la API es encriptado (SSL) y autenticado.
- Autorización granular a los recursos está dado por un sistema de control de acceso por roles (RBAC).
- Masters, son capaces de vincularse a sistemas externos de gestión de identidad.
 - LDAP, Active Directory, OAuth provisto por Google, Github, etc



DEMO

GitLab

OpenShift Container Platform

Health, Scaling, Remediation Pods

- **Health and Scaling**

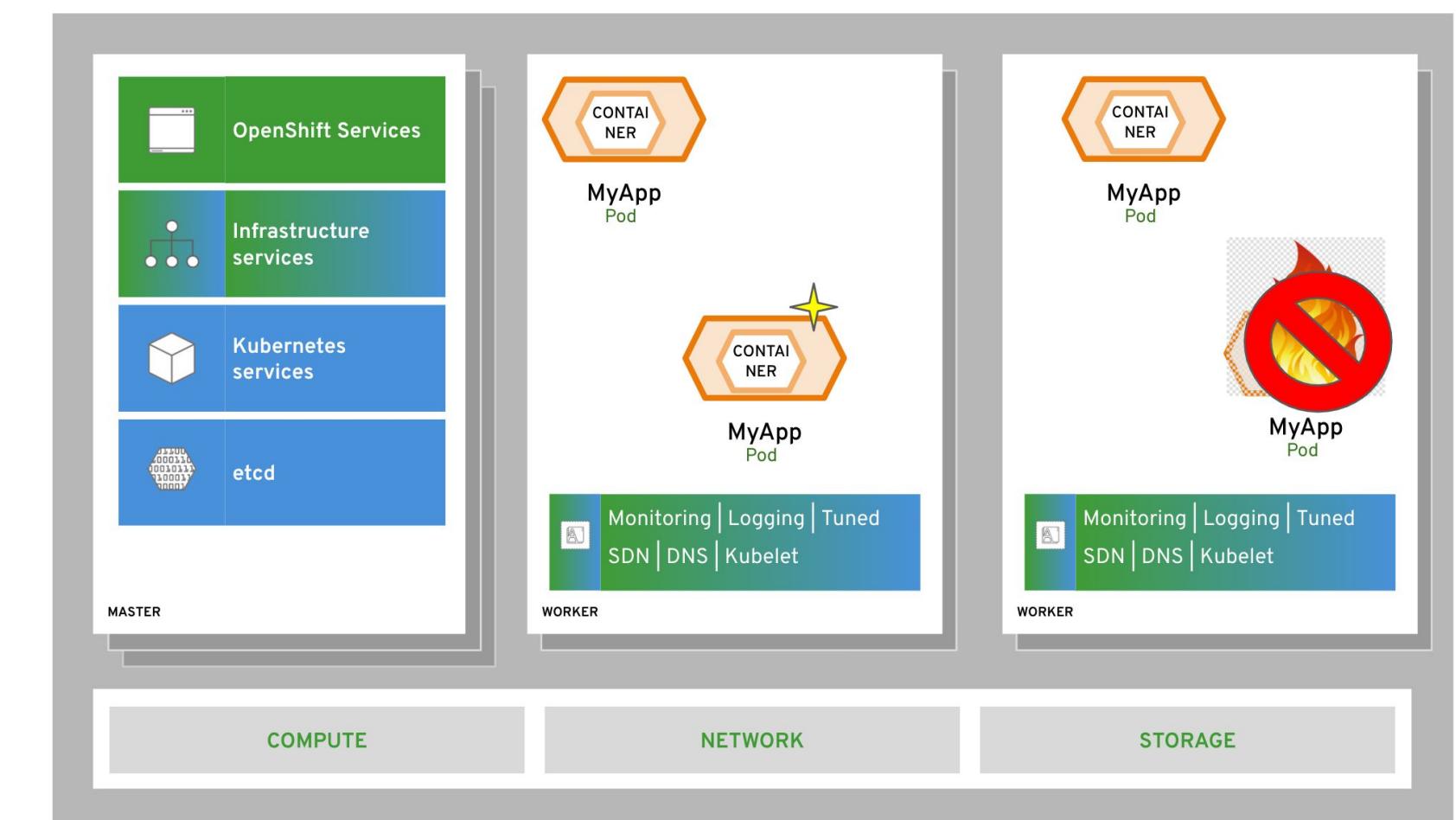
- Master nodes chequean constantemente la salud de los pods
 - Readiness probe
 - Liveness probe
- Los pods pueden *escalar automáticamente* en base a métricas de CPU y Memoria.

- **Unhealthy Pods**

- Qué sucede cuando el control plane detecta que hubo una falla?. Qué sucede si un contenedor falla internamente?

- **Remediation**

- Si falla un liveness probe restart del pod
- Si falla un readiness probe, se lo marca como fallido y no se elimina. No es agregado como endpoint.
- Service solo envía tráfico a los pods Ready
- Master nodes orquesta automáticamente y mantiene la disponibilidad.

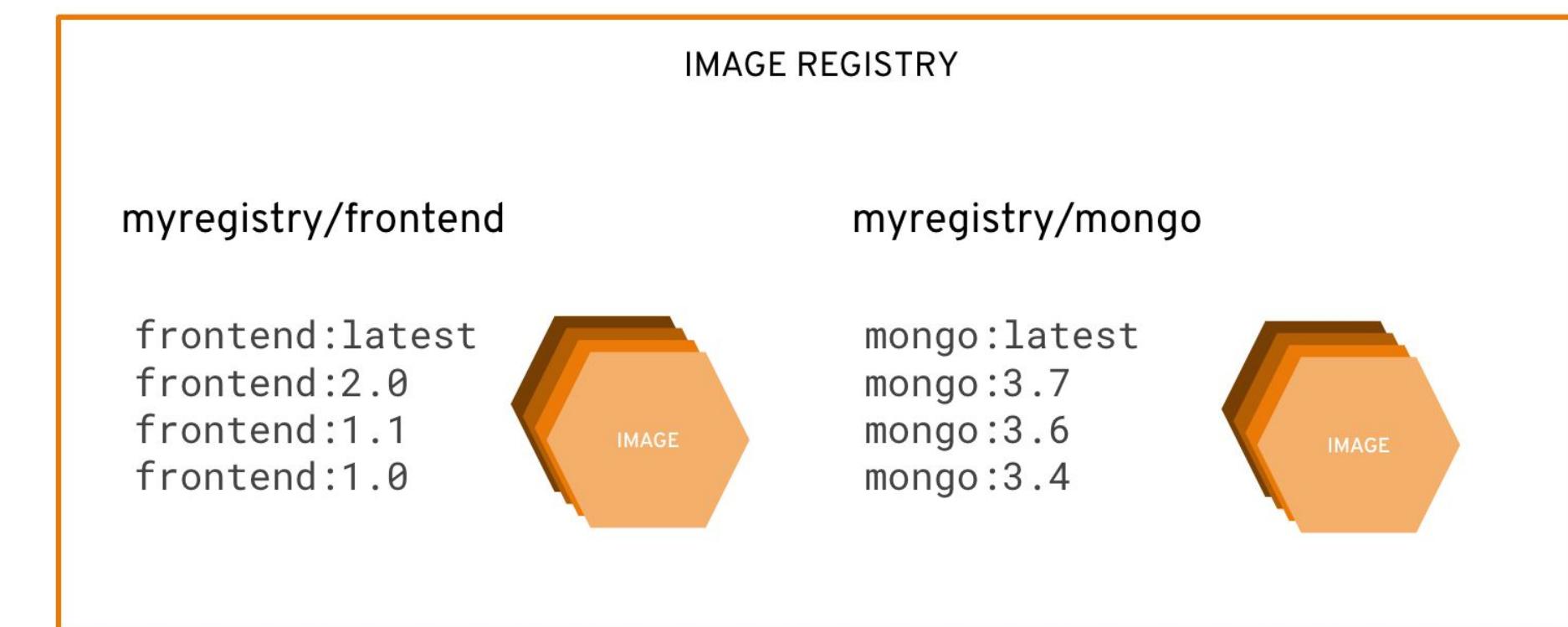


OpenShift Container Platform

Integrated Registry

Integrated Registry

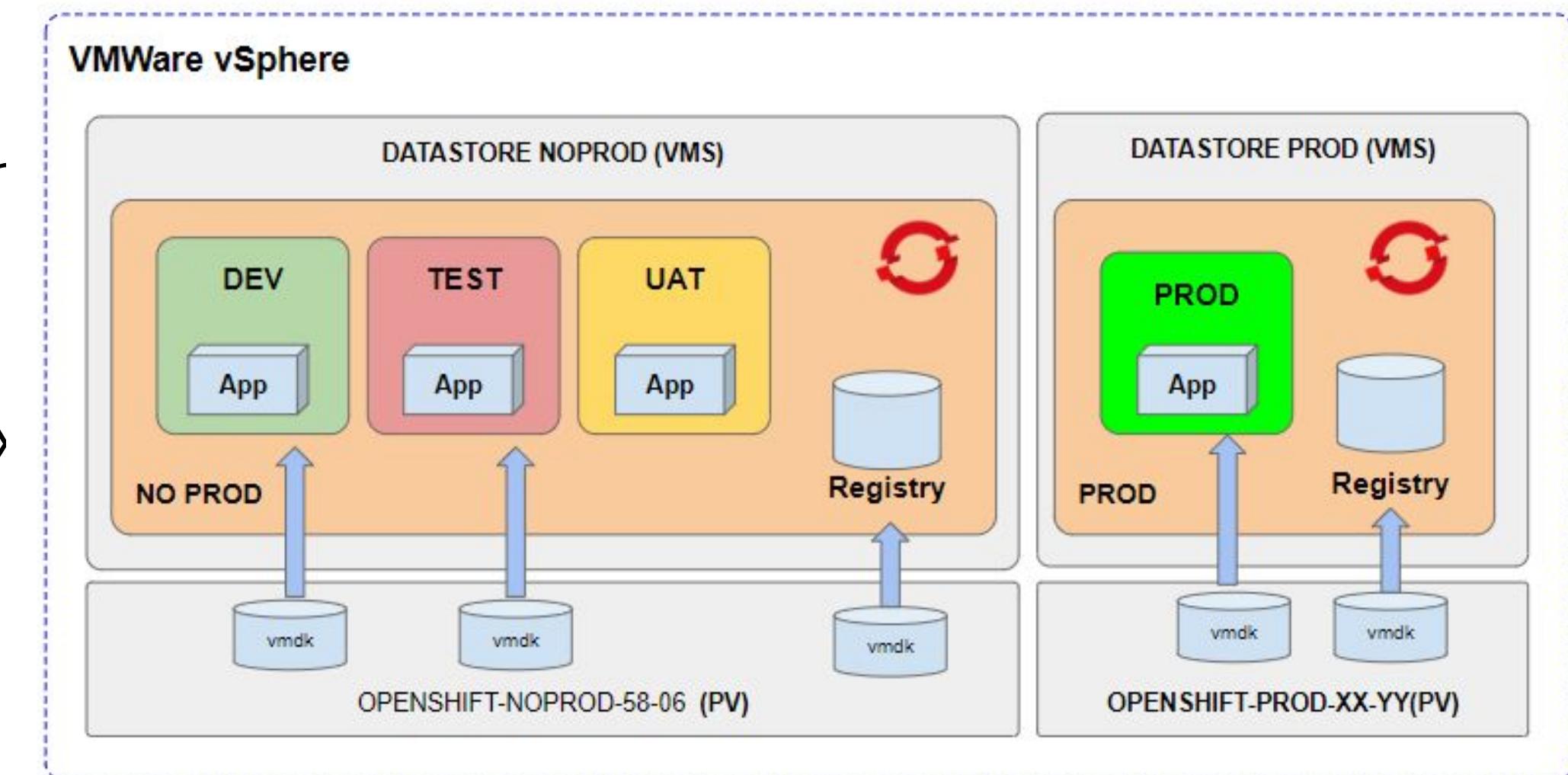
- OpenShift incluye una registry integrada para alojar imágenes de contenedores.
- Cuando una nueva imagen es creada es pusheada a la registry, OpenShift agrega información a la imagen como:
 - namespaces
 - name
 - image metadata
- varios componentes reaccionan con la creación de una nueva imagen es creada. Procesos de build y deployment.



OpenShift Container Platform

Persistent Storage

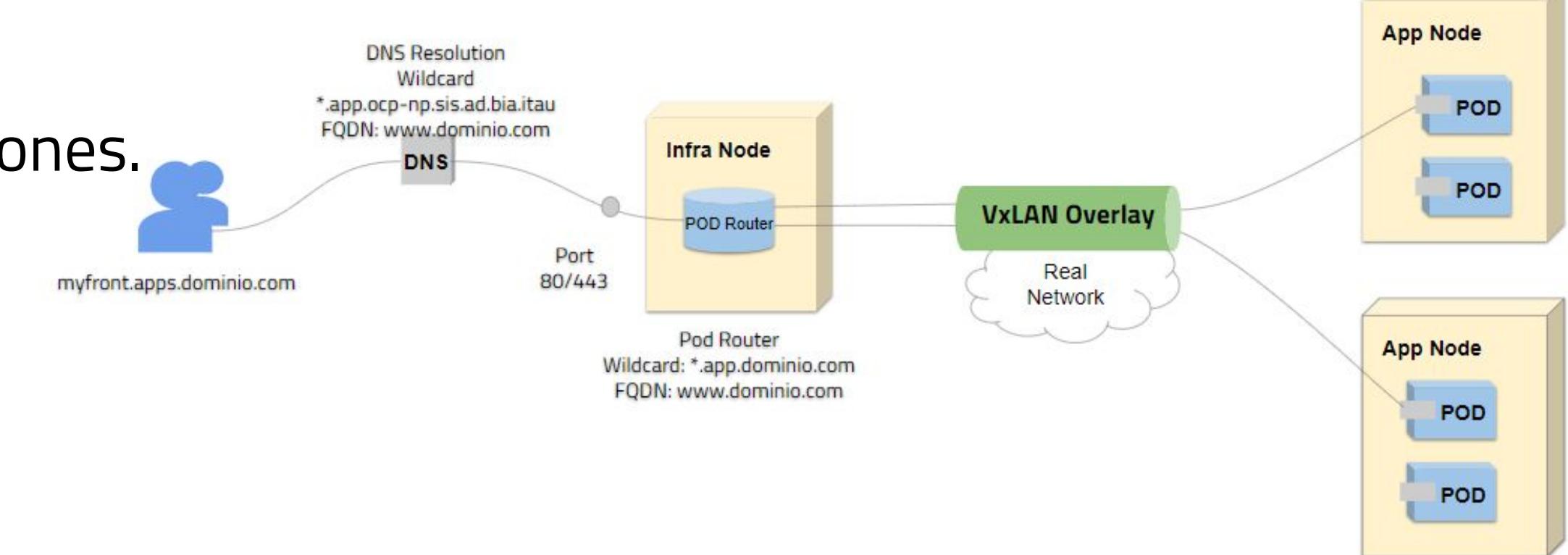
- Los contenedores son nativamente efímeros.
 - Los datos no son guardados cuando el contenedor es reiniciado.
- OpenShift provee un subsistema de persistencia de datos que conecta el storage corporativo a los pods que lo requieran.
 - La persistencia de storage permite crear aplicaciones statefull.
- OpenShift permite la integración de múltiples proveedores de storage (Kubernetes storage plugins).
 - La variedad se compone de:
 - Raw disk: iSCSI, Fiber Channel (directo al nodo como hostpath)
 - Enterprise NFS (Isilon, NFS Server, Gluster NFS, etc)
 - Provision dinamica de storage para pods:
 - On-Premise: VMware vSphere (datastore), block volume RW
 - Cloud Solution: Ceph, AWS EBS, pDisk



Openshift Container Platform

Routing Layer

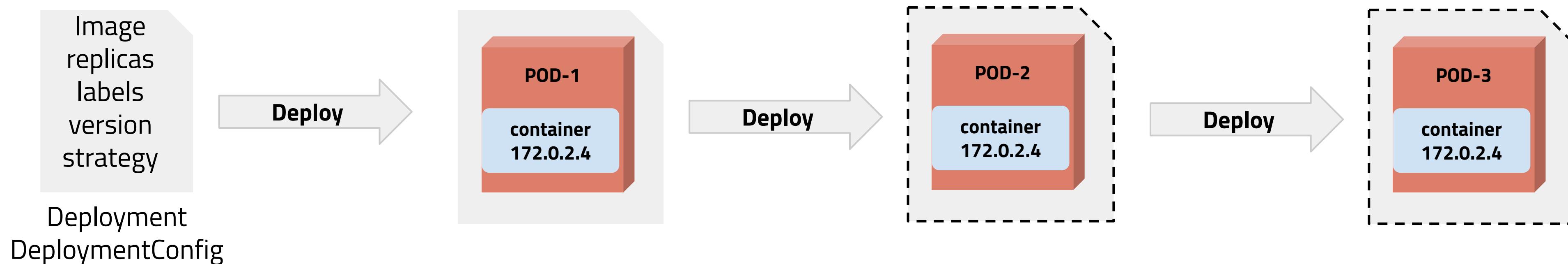
- La capa de ruteo provee acceso externo a las aplicaciones que se encuentran dentro del cluster de Openshift
- Trabaja en conjunto con la capa de servicio.
- Corre en pods dentro de Openshift, pod routers (ingress-controller)
- Provee:
 - Para los usuarios externos, balanceo automatico a pods.
 - Load balancing y auto-routing detectando pods en estado unhealthy
- Routing layer desacoplado y extensible.
 - Es posible distribuir la capa de acceso entre múltiples ingress-controller desacoplando el acceso a las aplicaciones.



OpenShift Container Platform

ReplicaSet and ReplicationController

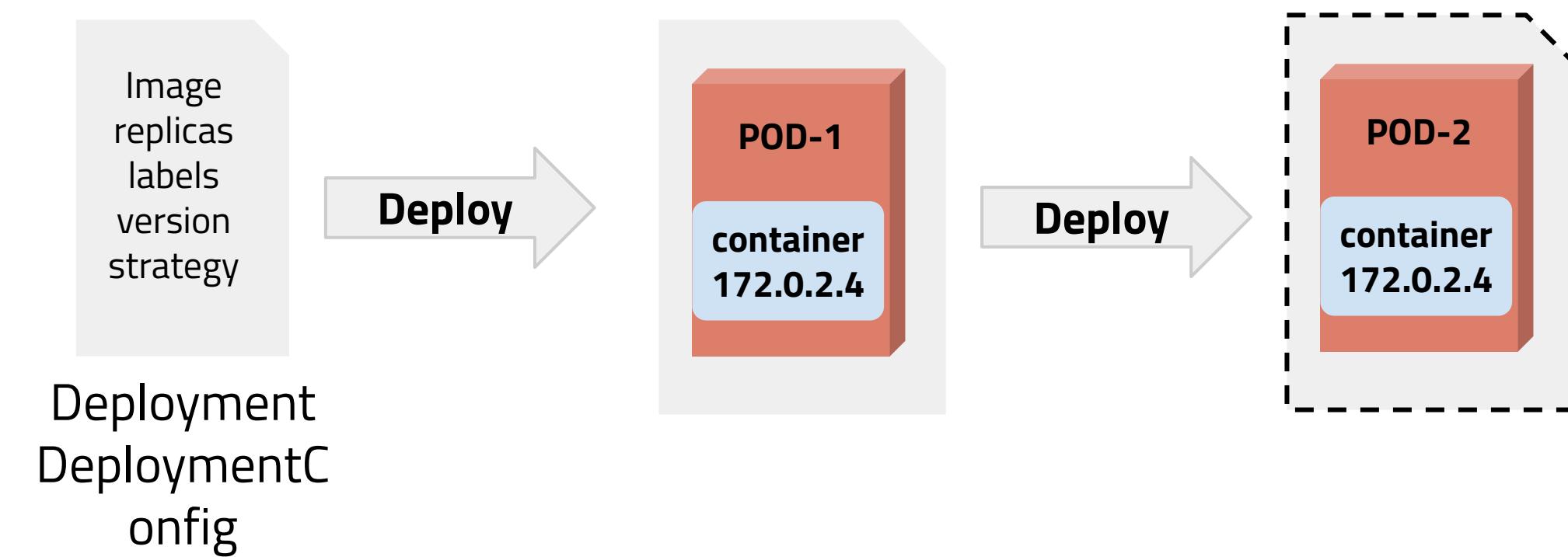
- Dos implementaciones para desplegar contenedores:
 - ReplicaSet = Kubernetes
 - ReplicacionController = OpenShift
- Aseguran el número de réplicas de pods que serán ejecutadas.
- Si un pod existente es eliminado, el replication controller lo instanciará nuevamente.
- Si hay mas pods corriendo de los que se necesitan, ReplicaSet eliminará tantos como sea necesario



OpenShift Container Platform

Deployments vs DeploymentConfig

- Define cómo será el rollout de la nueva versión del pod.
- Identifica:
 - Images name
 - Numero de pods
 - Labels target para el despliegue de los pods.
- Update en base a triggers:
 - Version
 - Strategy
 - Change Images
- Crean replicaSet (deployment) o replicationController (DeploymentConfig)



OpenShift Container Platform

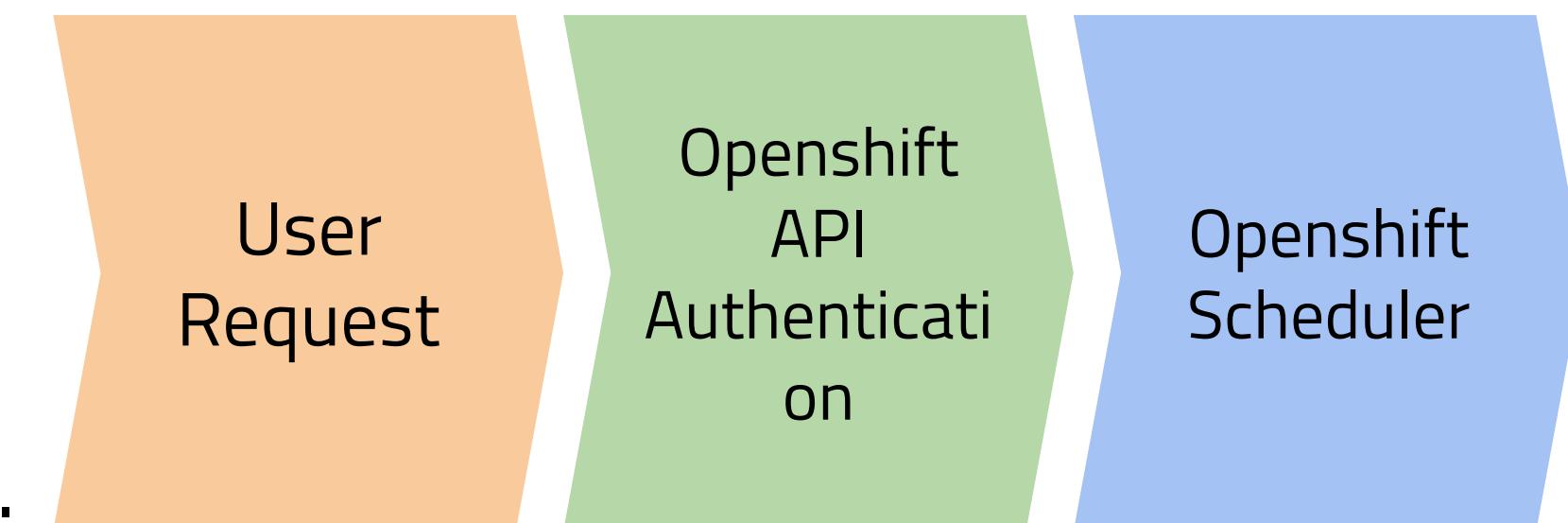
Container Deployment Workflow

El workflow de trabajo para el despliegue de aplicaciones es:

1. Un nuevo requerimiento para el creado de una aplicación es enviado vía Web Console, UI o Rest API.

2. OpenShift API Authentication

- Es aprobado el request, se evalúan permisos de usuarios, recurso de quotas y otra información.
- Son creados los recursos que son necesarios: DeploymentConfig, replicationController, service, routes, persistent storage claim, etc.



3. OpenShift Scheduler

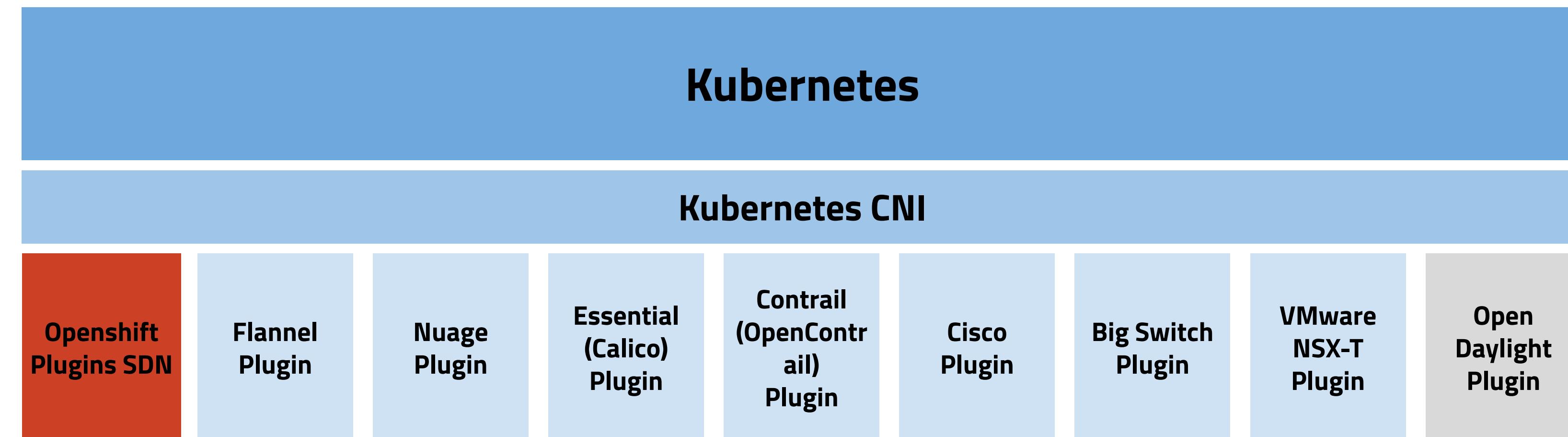
- Es designado un nodo worker para cada pod, son evaluados los recursos, carga, labels y se distribuye la aplicación en múltiples nodos para mayor disponibilidad.
- Es ejecutado el contenedor (pod) en el worker node.

OpenShift Container Platform

Networking Workflow

OpenShift Networking

- Contenedores de networking basados en Open vSwitches
- Platform-wide routing tier
- Permite la integración con SDN de terceros (Cisco ACI, NSX, etc)
- Integrado con DNS, ruteo y balanceo de carga.



OpenShift Container Platform

Networking Workflow

Route (resources)

- Recurso definidos en el alcance de un proyecto.
- Los servicios internos pueden ser alcanzados externamente por una entrada de dns (dinámica o no)
- Una ruta está compuesta por:
 - Nombre
 - Servicio, selector.
 - Seguridad (TLS, opcional)
- Los router definen rutas, los servicios proveen los endpoints de conexión para el balanceo.

OpenShift Container Platform

Networking Workflow

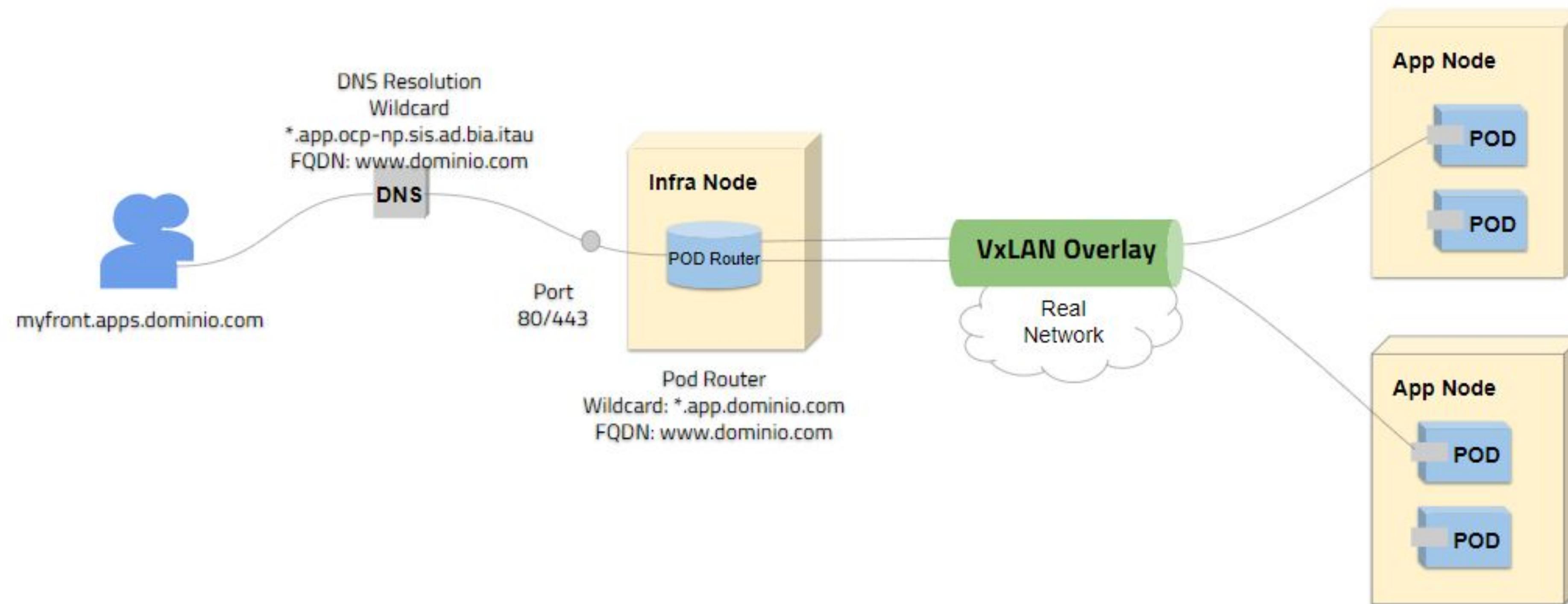
Router

- Multi-tier application de facil despliegue.
 - Requiere una capa de enrutamiento para llegar a las aplicaciones por fuera de OpenShift.
- Los routers pueden correr en cualquier nodos worker (multiple ingress controller)
 - Administrador crear wildcard de DNS (registros CNAME o A) sobre el DNS Server.
 - DNS Server deben resolver las ips de los nodos worker donde estén corriendo los pods router (binding port 80, 443)
- Router son los puntos de ingreso para el tráfico destinado a los pods hosteados en OpenShift.
 - Router resuelven las rutas externas (<https://myapp-myproject.apps.ocp.semperti.com>)
 - Los pod router actuan como load balancer proxy para conectar directamente a los pods (dns > nPODs)

WARNING!: No confundir resource route con pod routers

Openshift Container Platform

Networking Workflow



Openshift Container Platform

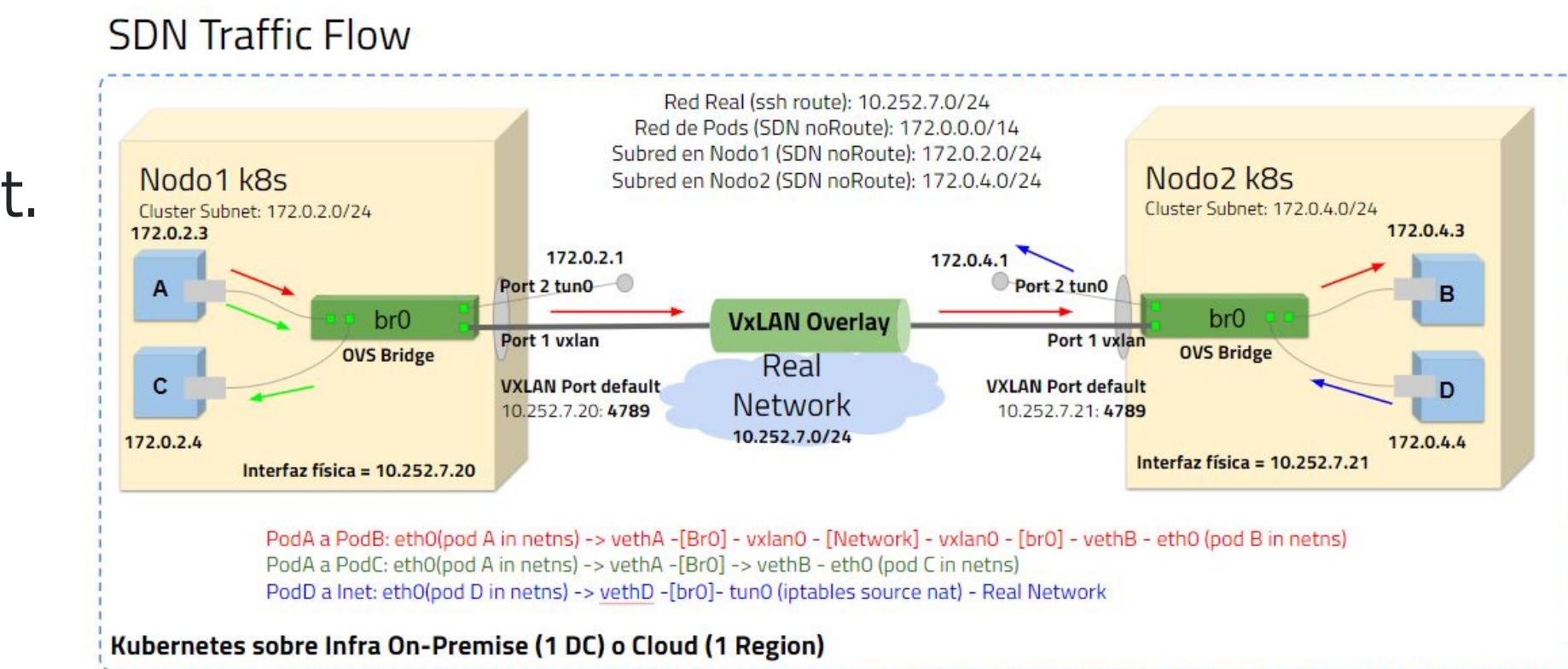
Pod Connectivity

Pod Connectivity

- Los pod usan redes de Openshift en el host para conectarse con otros pods y redes externas.
- Todos pods conectan sus interfaces de red a un **Open vSwitch (ovs)**, uno por nodo.
- Tres escenarios de tráficos distintos, graficados en la imagen Openshift SDN.
 - Un pod transmite paquetes a otro pods en otro nodo.
 - Un pod transmite paquetes a otro pod en el mismo nodo.
 - Un pod transmite paquetes a otro system externo al cluster de Openshift.

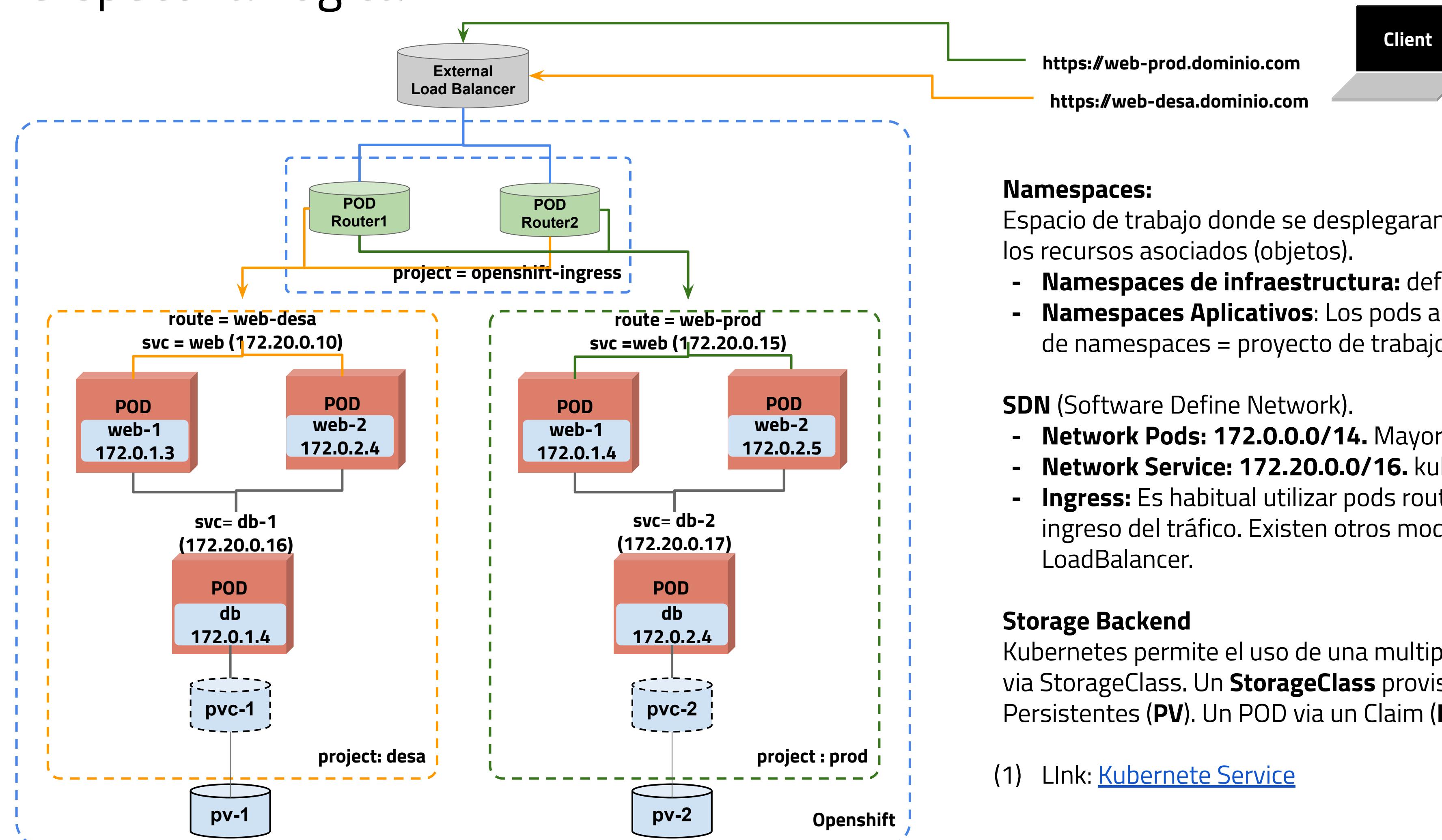
Service y Pods

- Los servicios ofrecen un acceso permanente a través de una sola ip address a un grupo similar de pods.
- Internamente, lo servicios balancean el tráfico hacia los pods que actúan como backend.
- Los pods backend pueden ser agregados o removidos desde el servicios de manera arbitraria mientras los servicios estén disponibles.
- Los servicios tienen un nombre de dns interno asociado a la ip.
 - Ejemplo: myservice.myproject.svc.cluster.local
 - Cuando un servicio es requerido, el nodo de Openshift hace un proxy-pass entre el servicio y el pod.



OpenShift Container Platform

Perspectiva Lógica



DEMO

GitLab

Despliegue de Aplicaciones

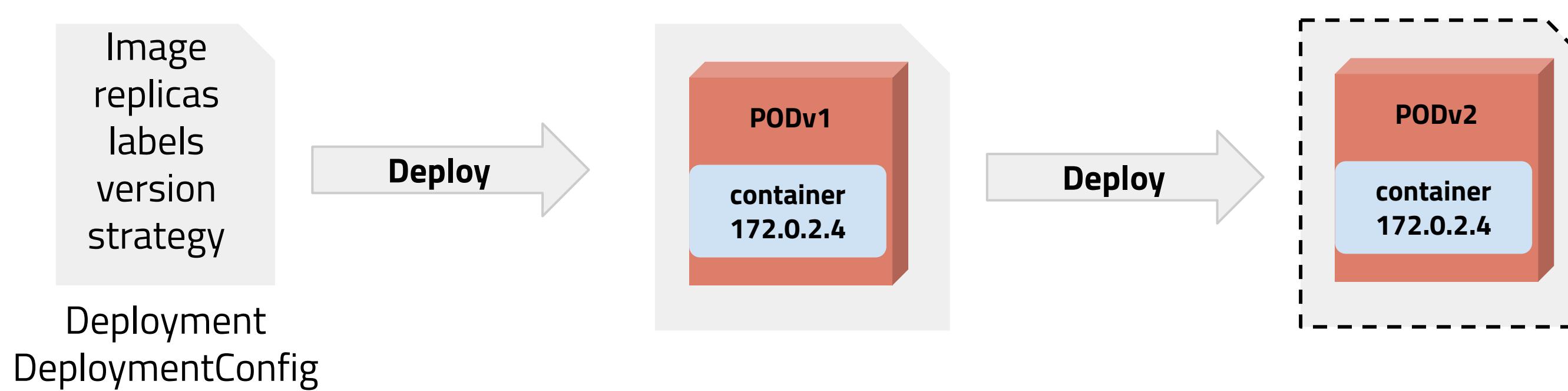
Semperti

Openshift Container Platform

Deployments

Deployments

- Openshift deployments proveen administración fina de aplicaciones.
 - Basada en templates definidos por los usuarios llamados "deploymentConfig"
- Existen diferentes tipos de templates para poder desplegar aplicaciones.
 - Templates, Helm, Kustomize.
- Openshift es Kubernetes
 - Kubernetes "Deployment" >> Openshift "DeploymentConfig"
 - Kubernetes "ReplicaSet" >> Openshift "ReplicationController"



OpenShift Container Platform

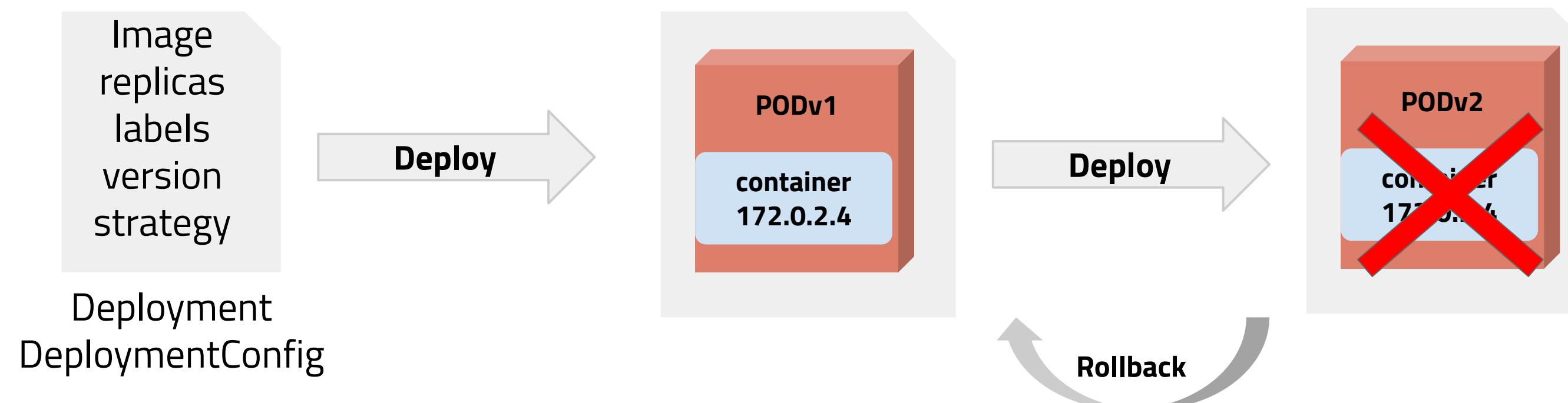
Deployments Features and Rollback

Deployment Features

- Deployment Configuration: Template for running application.
 - Contiene version number.
 - Cada vez que se ejecuta un nuevo deployment, se genera una nueva versión del replicationController.
- Triggers para automatizar deployments en base a eventos.
- Estrategias para la transición hacia una nueva versión.
- Rollback a versiones previas en caso de un deploy falle.
 - Manual o Automatico.
- La replicación puede ser manual o automatizada

Rollbacks

- Deployment permite rollbacks
 - Rollback a una versión previa de la aplicación
 - Puede ser pedido via REST API, CLI o Web Console
- La configuración del deployment soporta rollback automáticos a la última versión que ha funcionado.



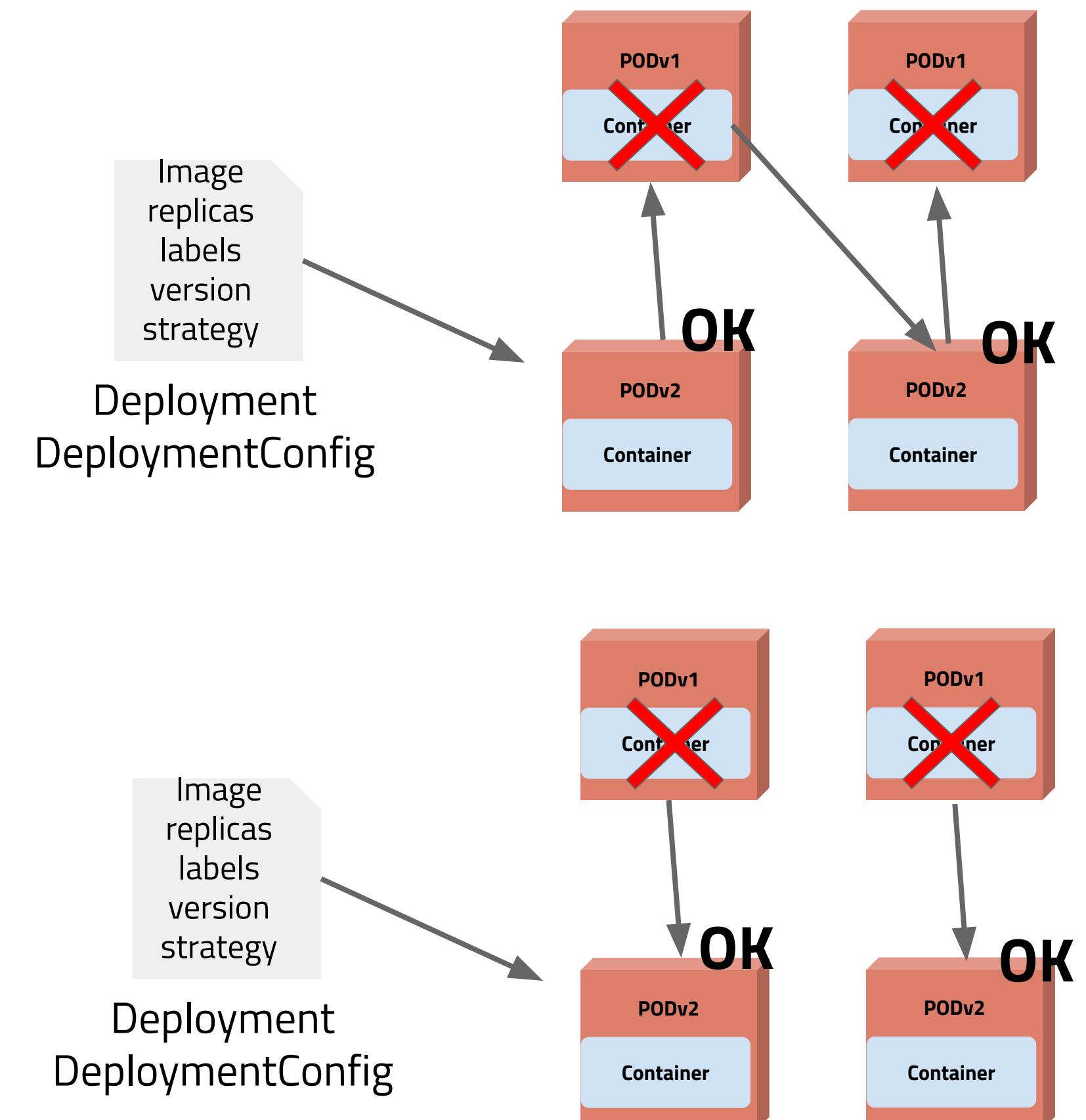
OpenShift Container Platform

Rolling and Recreates Strategy

Rolling Strategy

Permite una actualización continua.

- Soportan life-cycle hooks.
- Esperan que el pod pase los probes de readiness antes de realizar el scale down del viejo componente.
 - Si un pod no pasa el probe de readiness no es agregado al servicio.
- Usado por default si no es especificada otro modo de deploy.
 - *Rolling*
 - *Recreate*



OpenShift Container Platform

Deployment Strategy

Rolling Deployment Strategy Process

- Pasos para el proceso de rolling update.
 - Ejecutan pre life-cycle hooks
 - Scale up del nuevo deployment por uno o mas pods (basado en maxSurge value)
 - Se espera que el readiness check sea completado.
 - Scale down de los viejos pods (basado en maxUnavailable value)
 - Repetir scaling hasta que los deployments alcancen el número deseado y los viejos sean reemplazados.
 - Ejecutan los post life-cycle hooks.
- Cuando es realizado el scale down, el pod espera que este ready.
 - Deciden si el scale up decide si afecta la disponibilidad.
- Si es realizado un scale-up de un pod nunca está ready, el deployment termina por timeout.
 - El resultado del deployment es fallido.

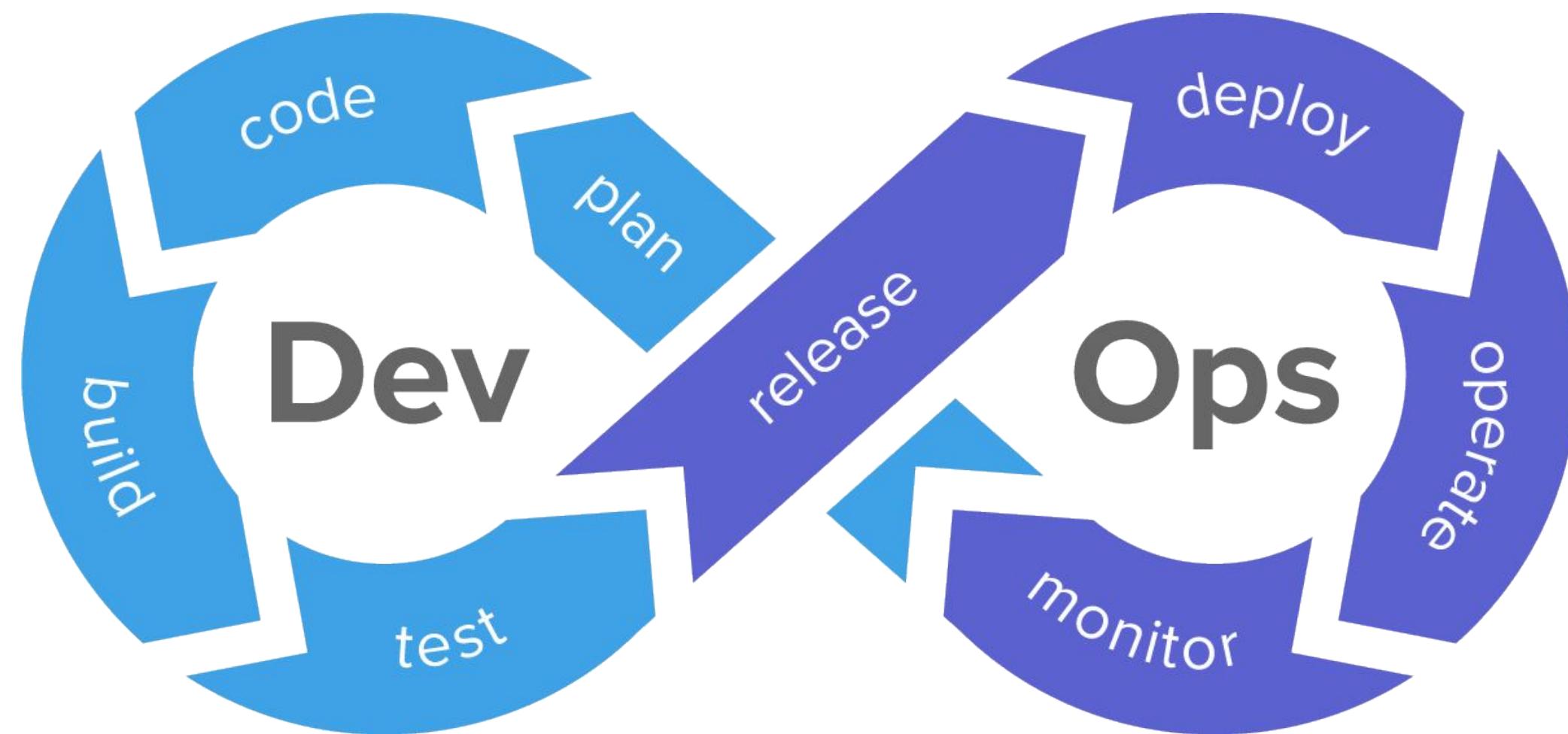
Recreate Strategy

- Soporta life-cycle hooks para injectar código en el proceso de deployments.
- Los pasos en la estrategia de recreate:
 - Ejecuta pre life-cycle hook
 - Scale down a la versión previa
 - Scale up a la nueva versión
 - Ejecuta post life-cycle hook

OpenShift Container Platform

Workflow CI/CD

DevOps = Continuous Integration (CI) + Continuous Deployment (CD)

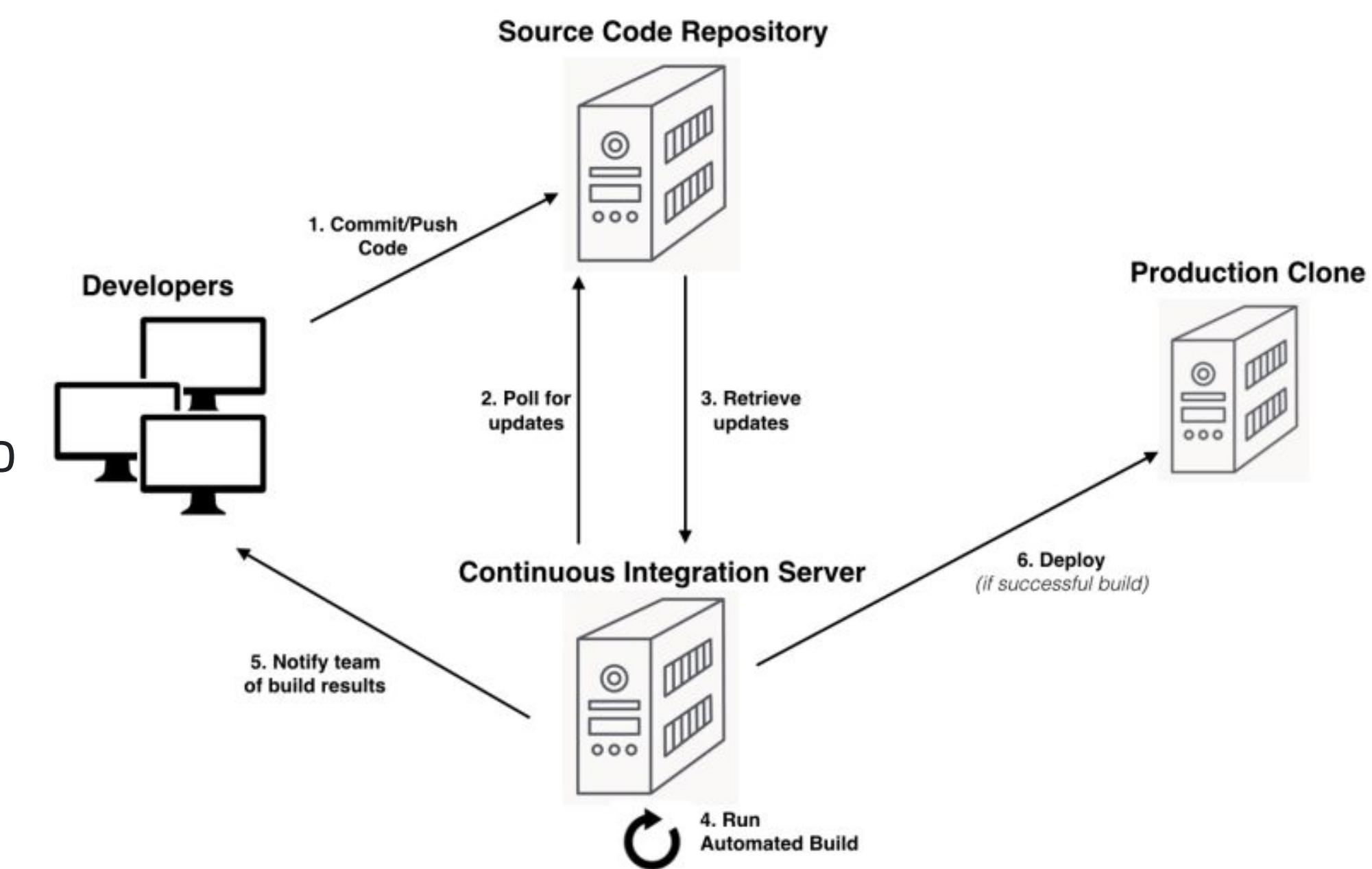


OpenShift Container Platform

Workflow CI/CD

Continuous Integration (CI)

- **Verificar la integridad del build:** Chequear si el código puede ser pulleado desde el repositorio de código y construir el deployment. El proceso de build puede incluir la compilación, packaging, configuración del software.
- **Validar el resultado de los test:** Correr los test creados por los desarrolladores en ambientes similares a los de producción. Verificar que el código fuente no este roto como efecto de un nuevo commit.
- **Chequeos de integración entre múltiples sistemas:** Los test de integración son usados para validar el funcionamiento con otros sistemas.
- **Reportar problemas:** Alertas a los equipos afectados para reparar los problemas.



OpenShift Container Platform

Builds and Source to Images (S2I)

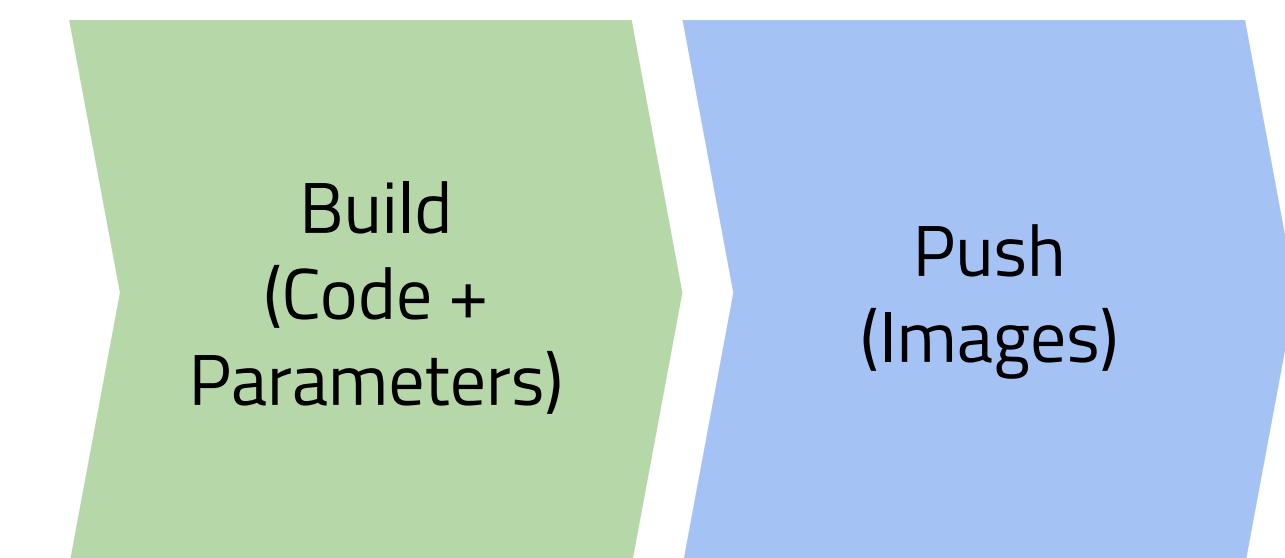
Builds and Souce to Images (s2i)

Builds se define como proceso que transforma los parámetros de entrada en un objeto resultado. Definimos el input como el código fuente y el output una imagen de aplicación. En openshift el recurso para representar el proceso de build es el [buildConfig](#). El proceso queda definido como:

1. Crear un contenedor desde un build images.
2. Push del contenedor a la registry interna.

Respecto a las estrategias de build, Openshift ofrece una gran cuatro alternativas.

- Container build. > Runnable Container Images
- Source-to-Images build. > Runnable Container Images
- Custom Builds > Lo que el desarrollador diga.
- Pipelines Build (deprecado) > Jenkins Pipelines ejecutable por el plugins de jenkins (deprecado)



OpenShift Container Platform

Images Streams

Images Streams

Cada proceso de build genera una nueva imagen, nuevo release de la misma aplicación. Este conjunto de imágenes de la misma aplicación se lo llama **imageStreams**

Representa un conjunto de imágenes de contenedores OCI compliance identificadas cada una con un **tag**.

```
oc get is -o yaml
```

Conceptos claves

- Nuevas imágenes de contenedor son agregadas como ImagesStreams
- Builds y Deployments observan los images streams
 - Reciben notificaciones cuando una nueva imagen es agregada.
 - Reaccionan realizando un nuevo build o deployment.

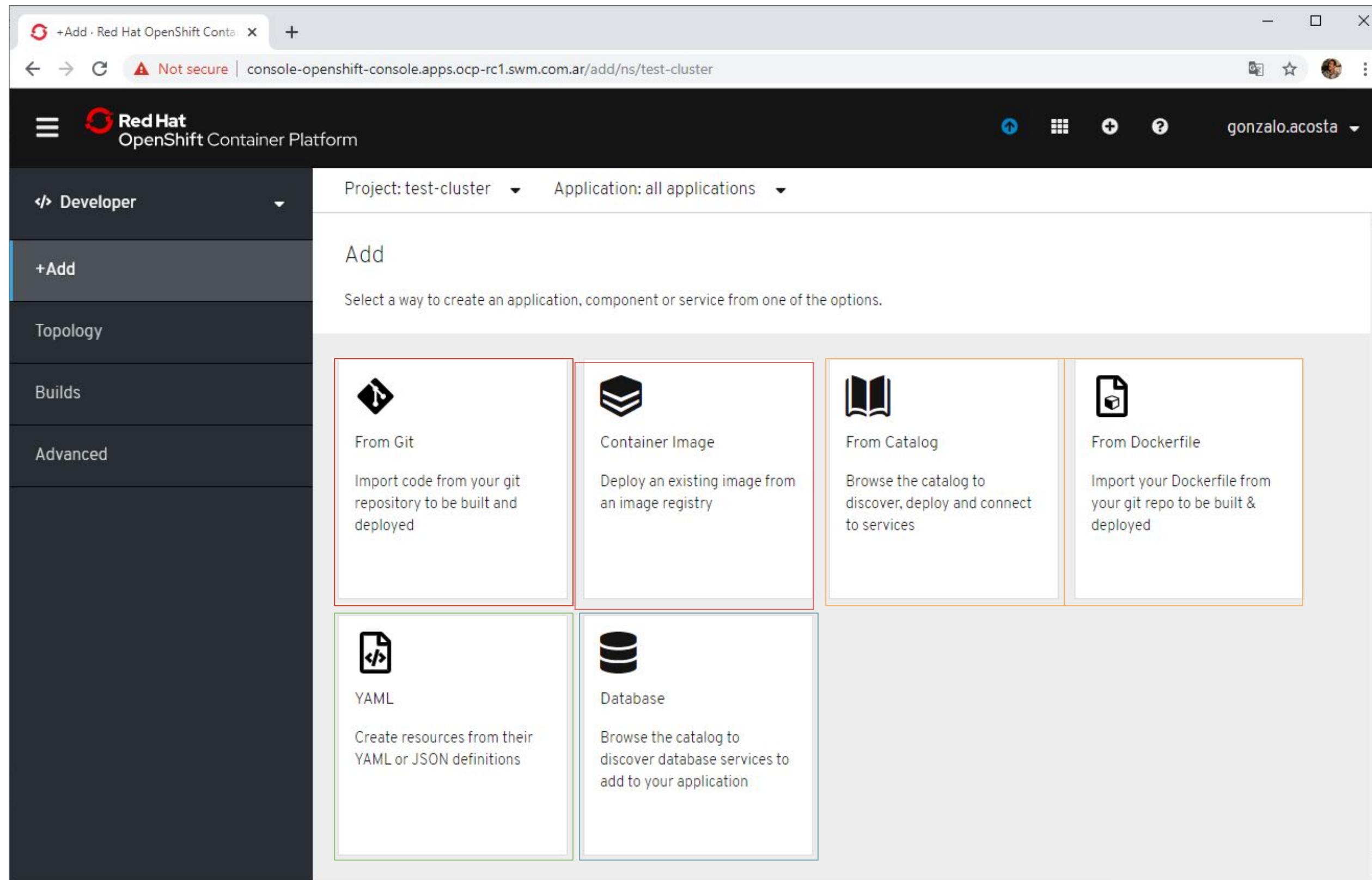


DEMO

GitLab

Catalogo Self Service

Developer Portal < >



From Git: Para despliegues Source to Images (s2i), tomando como partida el código fuente de un repositorio git.

Catalog Container: Desplegar una aplicación desde la imagen de un container previamente realizada.

From Catalog: Desplegar una aplicación desde un catálogo nutrido provisto por Red Hat. En este encontraremos todos los frameworks de desarrollo.

From Dockerfile: Construcción de una aplicación en base a un Dockerfile en un repo git.

YAML: En caso de que contemos con un Template YAML para el despliegue de la app.

Databases: En caso de que se requiera desplegar una base de datos Mongo, MariaDB, MySQL o PostgreSQL para instancias efímeras o persistentes single instance (no cluster ni distribuidas).

Despliegue de aplicaciones

Kubernetes Operators [Cluster-Admin]

Un Operator representa el conocimiento de la **acción humana** definido en software para administrar aplicaciones.

Machine Operators

Encargados del despliegue de VMs y su escalado (Cloud Providers)

Cluster Operators

Encargados del despliegue de todas las herramientas de infraestructura. Ingress, registry, logging y metrics

Update Operators

Encargado del ciclo de update de toda la plataforma

Catalogo Self Service

OperatorHub

Administrador # > Operators > OperatorHub

The screenshot shows the Red Hat OpenShift OperatorHub interface. The left sidebar has a navigation menu with sections like Home, Dashboards, Projects, Search, Explore, Events, Operators (selected), Workloads, Networking, Storage, and Builds. Under Operators, 'OperatorHub' is selected, and 'Installed Operators' is shown. The main content area is titled 'OperatorHub' and describes it as a place to discover operators from the Kubernetes community and Red Hat partners. It shows a grid of operators categorized by type (Community, Red Hat, Akka) and provider. The grid includes items like '3scale' (Community, provided by Red Hat), 'Akka Cluster Operator' (Red Hat, provided by Lightbend, Inc.), 'AMQ Broker' (Community, provided by Red Hat), and 'AMQ Certificate Manager' (Community, provided by Red Hat). A total of 155 items are listed.

Un Operador representa el conocimiento de la **acción humana** definido en software para administrar aplicaciones.

Red Hat: Operadores de productos provistos y mantenidos por Red Hat.

Vendors: Operadores de productos de terceros provistos y mantenidos por vendors particulares. Ejemplo New Relic.

Comunidad: Operadores de proyectos comunitarios provistos y mantenidos por la comunidad OpenSource.

Despliegue de aplicaciones

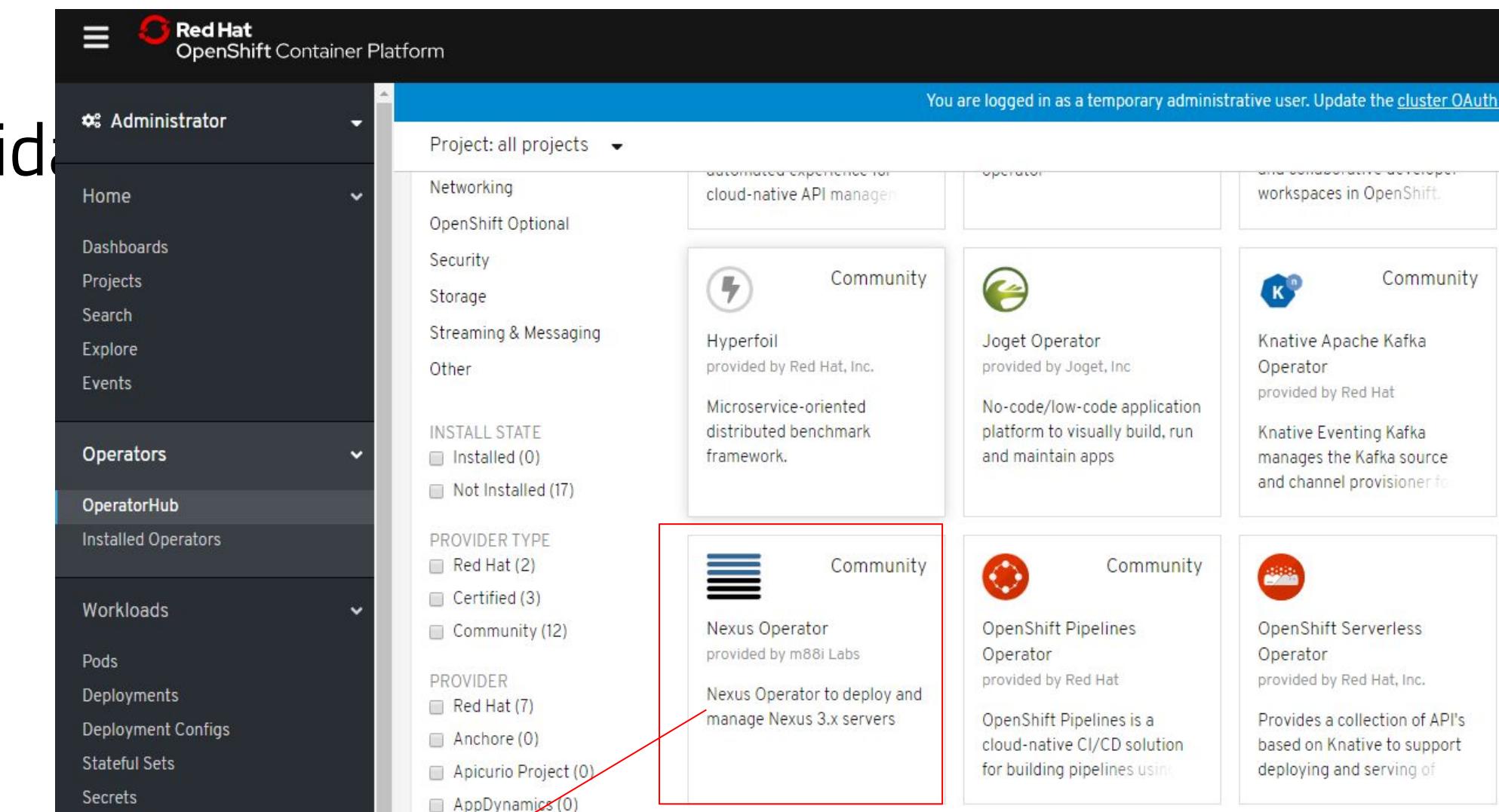
Kubernetes Operators

Operator

- Pod en non-terminating con misión definida.
- Cambia despliegues y mantiene carga de trabajo activa.
- Pod que corre playbooks de Ansible o código GO.

Objetivo

- Instalar y desinstalar aplicaciones
- Actualizar
- Escalar de manera horizontal
- Auto-Curado
- Backup & Restore.



Custom Resource
Definition
[CRD]

-Nexus

Custom Resource
[CR]
(Instancias de un CRD)

-Nexus Dev
-Nexus Prod

Perspectiva Lógica

Configuración de proyectos con Operadores

Namespaces: DESA

- NetworkPolicies
- Quotas
- LimitRanges
- ServiceAccounts
- RoleBindings
- ClusterRoleBindings
- Configmaps
- Pod

Quotas Small

cpu=2
mem=2G

Network Policies

allow-from-same-namespace
allow-from-ingress-namespace

Namespaces: TEST

- NetworkPolicies
- Quotas
- LimitRanges
- ServiceAccounts
- RoleBindings
- ClusterRoleBindings
- Configmaps
- Pod

Quotas Medium

cpu=4
mem=4G

Network Policies

allow-from-same-namespace
allow-from-ingress-namespace

Namespaces: HOMO

- NetworkPolicies
- Quotas
- LimitRanges
- ServiceAccounts
- RoleBindings
- ClusterRoleBindings
- Configmaps
- Pod

Quotas Large

cpu=8
mem=8G

Network Policies

allow-from-same-namespace
allow-from-ingress-namespace

La configuración del namespaces puede ser administradas de manera manual o por medio de un operador asignando labels

OpenShift NO PROD core=36 mem=211

DEMO

GitLab

Día 4

Semperti

Día 4

- Instalación de Openshift
- Autenticación
- Administración de Grupos
- RBAC
- Service Account
- Resource Management

Modos de Instalación

Semperti

OpenShift v4 Life Cycle

Easy install, easy configure and easy extended.



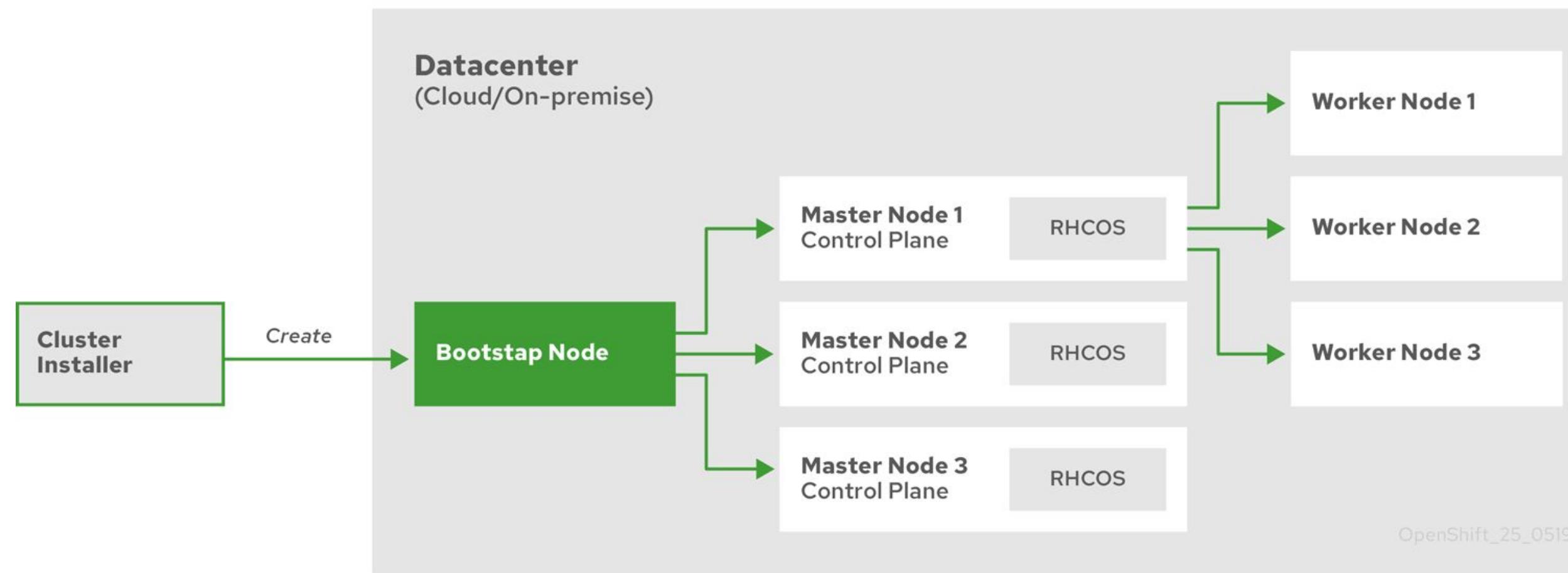
La instalación se realiza desde el nodo **bastión** haciendo uso del a tools `openshift-installer`. El bootstrap es quien inicia el proceso de instalación.

RH CoreOS desalienta el uso de SSH en post de la estandarización obteniendo nodos inmutables. Configuración de nodos (machine config) basados en ignitions files.

SDK para el desarrollo de Operators. **OperatorHub** como self service portal para la instalación de componentes sobre el cluster.

OpenShift Installation

Proceso de Instalación

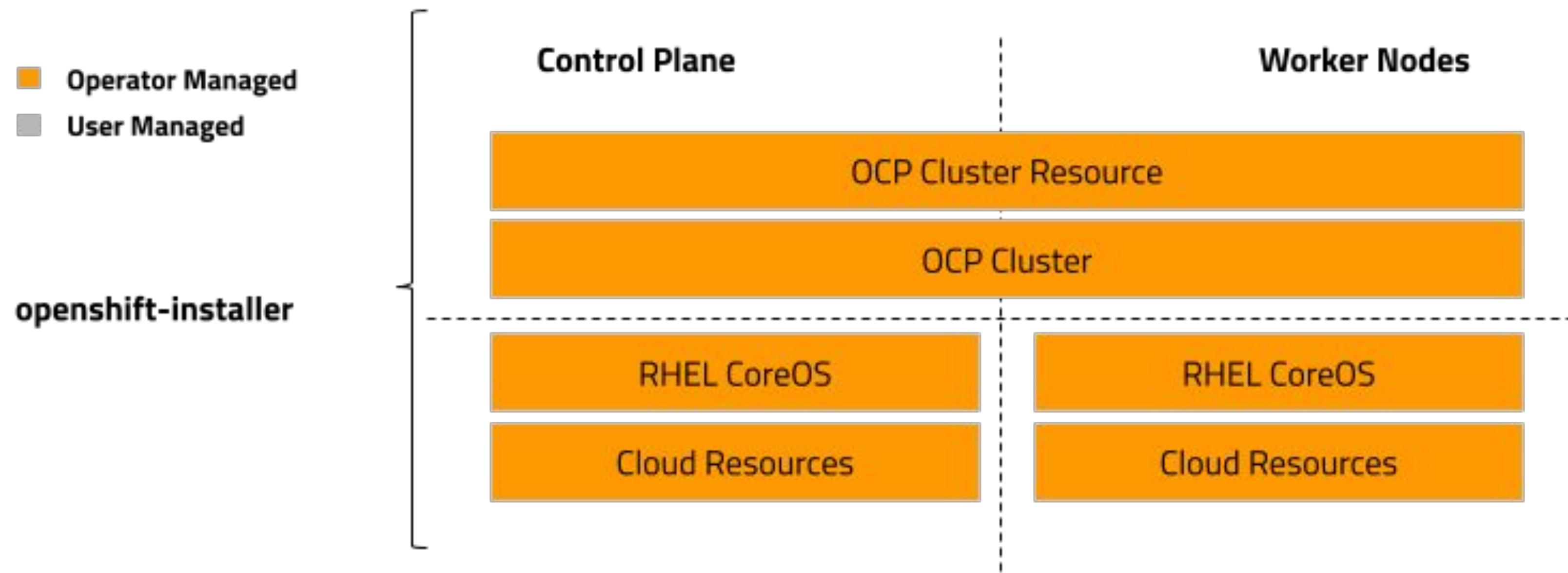


1. Creado de manifiesto de instalación con `openshift-installer`
2. Bootstrap inicia y levanta los recursos necesario para que los masters inicien.
 1. Master al iniciar toma información del bootstrap para finalizar su booteo.
 2. Master utiliza el bootstrap para formar su replica de etcd.
 3. Bootstrap levanta un control plane temporal de Kubernetes usando el nuevo etcd cluster.
 4. Control plane temporal programa el nuevo control plane productivo en los masters.
 5. Control plane temporal finaliza, mientras que el control plane productivo brinda servicio.

Bootstrap inyecta componentes específicos de OpenShift para la formación de un nuevo control plane y es apagado.

Modos de Instalación

Installer Provisioner Infrastructure (IPI)



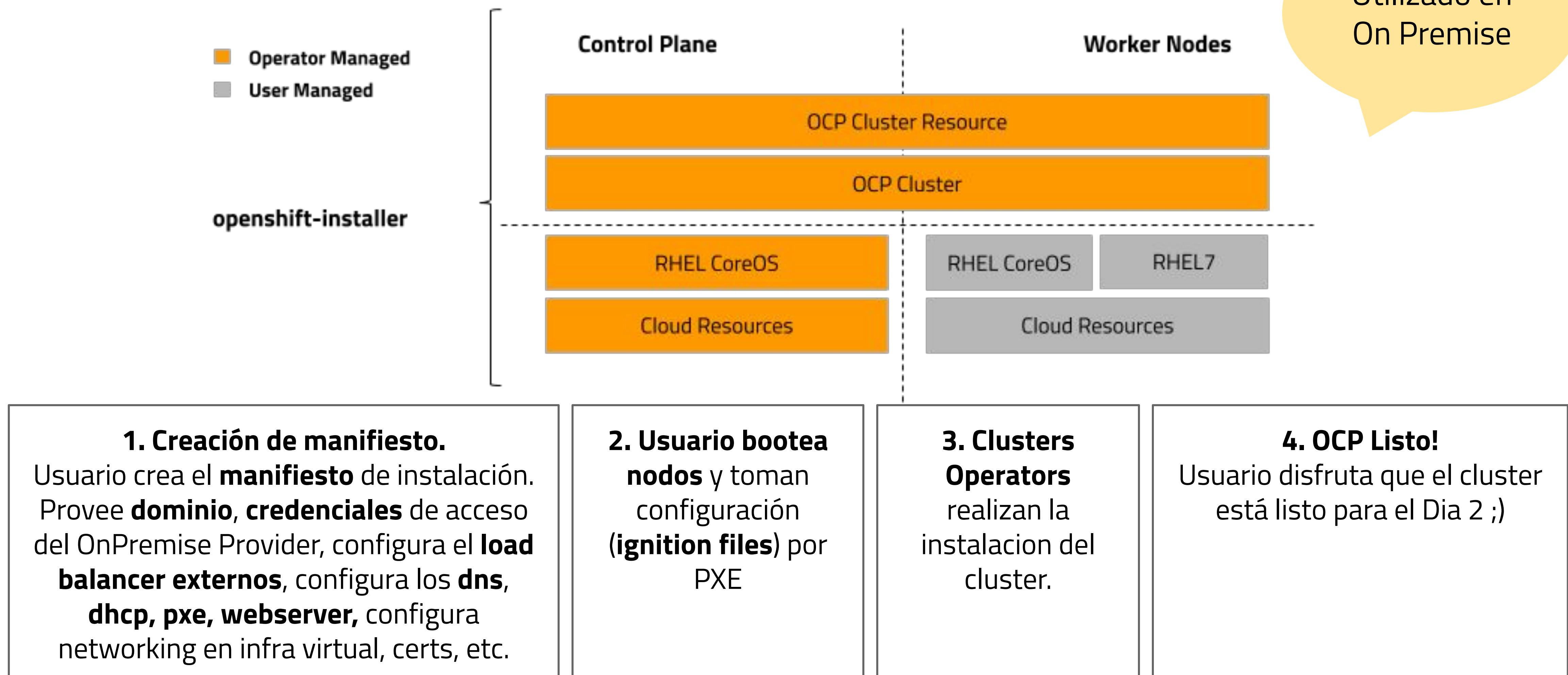
1. Usuario crea el **manifiesto** de instalación. Provee **dominio**, **credenciales** de acceso del Cloud Provider y lanza la instalación.

2. **Clusters Operators** realizan la instalacion del cluster.

3. **OCP Listo!**, Usuario disfruta que el cluster está listo para el Dia 2 ;)

Modos de Instalación

User Provisioner Infrastructure (UPI)



Modos de Instalación

IPI vs UPI

	IPI	UPI
Build Network	Installer	User
Setup Load Balancer	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	Installer: RHEL CoreOS + RHEL7
Node Provisioning / Autoscaling	Yes	Solamente para providers con OpenShift Machine API Support
Customization & Provider Support	Best Practices: AWS	Yes: AWS, Bare Metal & VMware

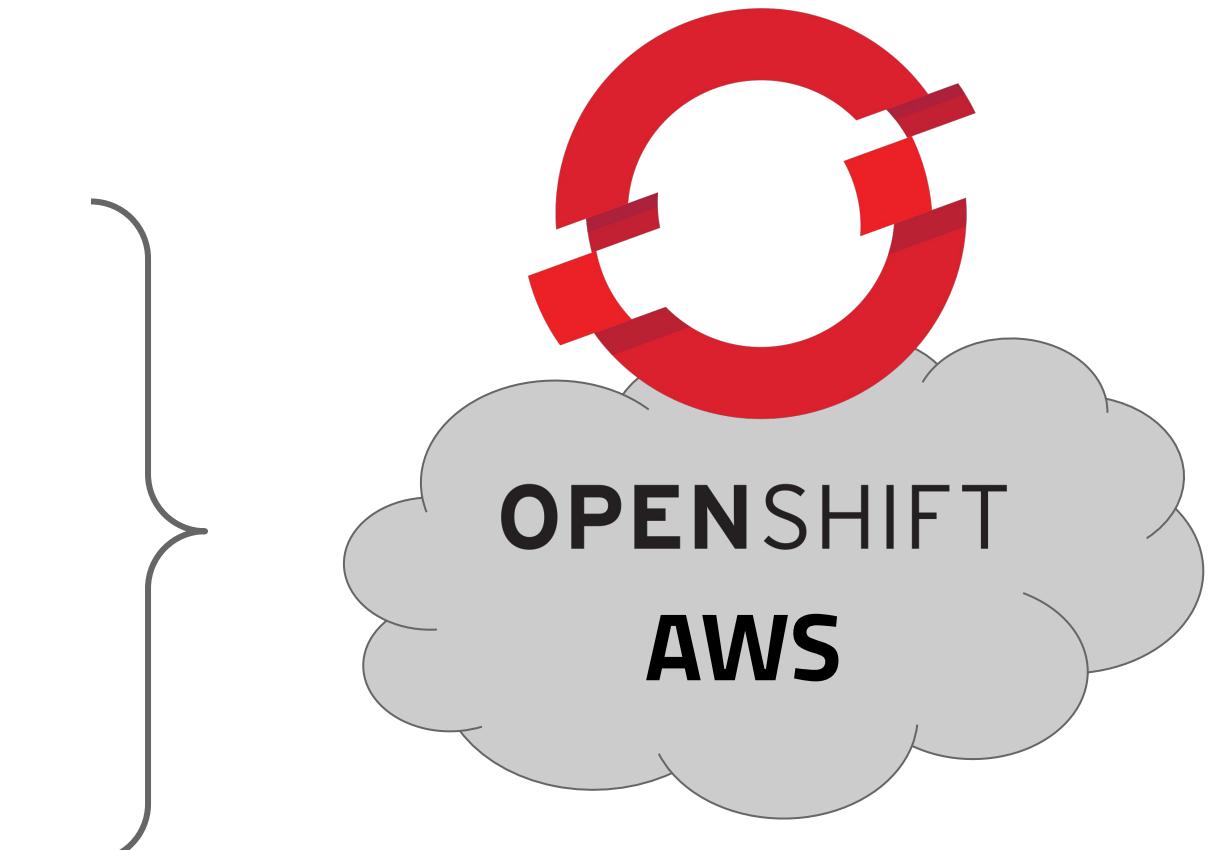
Modos de Instalación

IPI AWS Pre Requisitos

Pre requisitos AWS IPI

- **Openshift-installer** tools en una estación de trabajo.
- **Credenciales AWS** (Key y Secrets)
- **Openshift Pull Secrets** (cloud.redhat.com)
- **Dominio en Route53.**
 - Instalador crea objetos debajo del dominio y debe existir.
- **Private/Public key** (ssh-keygen -f ~/.ssh/cluster-key)

```
$ openshift-install create cluster --dir=$HOME/mycluster  
> SSH Public Key  
> Platform: aws  
> Region: us-east-1  
> Base Domain: ocp.semperti.com  
> Pull Secret: *****
```



Modos de Instalación

UPI Install - Día 1 - On Premise

Día 1 - Instalación

Devaluar cada aspecto de la infraestructura donde va a ser instalada.

- On Prem **solo UPI Install**.
- **Dominio y subdominio dedicado** que será el Cluster ID del cluster.
 - Domain: labs.semperti.com
 - Cluster ID: ocp4
- **DNS api y el wildcard** de dns estaran dentro del subdominio *.ocp4.labs.semperti.com
- **External Load Balancer** Mínimamente se necesitan dos balanceos externos (si o si)
 - LB API Server: VIP con backend master nodes.
 - LB Wildcard Apps: VIP con backend infra nodes.

- **Networking Nodes**
 - DHCP
 - Static IP
 - (requiere custom ignition files)
- **Storage Backend**
 - NFS
 - VMWare
 - OCS

```
$ openshift-install create cluster --dir=$HOME/mycluster  
Boot Bootstrap > Boot Masters > Boot Workers  
Aceptar CSR
```

Enjoy OpenShift Cluster!!!

<https://twitter.com/queltosh/status/1259067319239815170?s=20>

Modos de Instalación

UPI AWS Install - Día 2 - On Premise

Día 2 - Configuraciones Adicionales

- Configuraciones adicionales
 - **Ingress Controllers**
 - **LDAP**
 - **Certificados Customs** (ingress controller y API Server)
 - **Storage Backend** (NFS, VMWare Dynamic Provision)
 - **ClusterLogging** (ElasticSearch)
 - **IMPORTANTE!!! Curator, replicas, seguir el estado de salud.**
- Agregado de nuevos nodos.



<https://gitlab.semperi.com/gonzalo.acosta/workshop-openshift-administration/blob/master/04/labs/openshift-aws-install.sh>

DEMO

Revisión de Implementación en
sobre nodo Bastión

Cluster Machine Management

Semperti

Cluster Management

Machine Operator

- Openshift 4 es administrado vía Operadores
 - Creados durante la instalación
 - Administradores a través de custom resources (extensión de la API de Kubernetes)
- Efectivamente el cluster es operador por sí mismo.
 - Incluso el sistema operativo RHCOS.
- Posterior al proceso de deploy, el mismo cluster administra los upgrades.
- Cambio de la configuración del cluster son realizados via custom resources (CRs)
 - Ejemplo: Cambio en la configuración de los Ingress Controller, OAuth, etc.

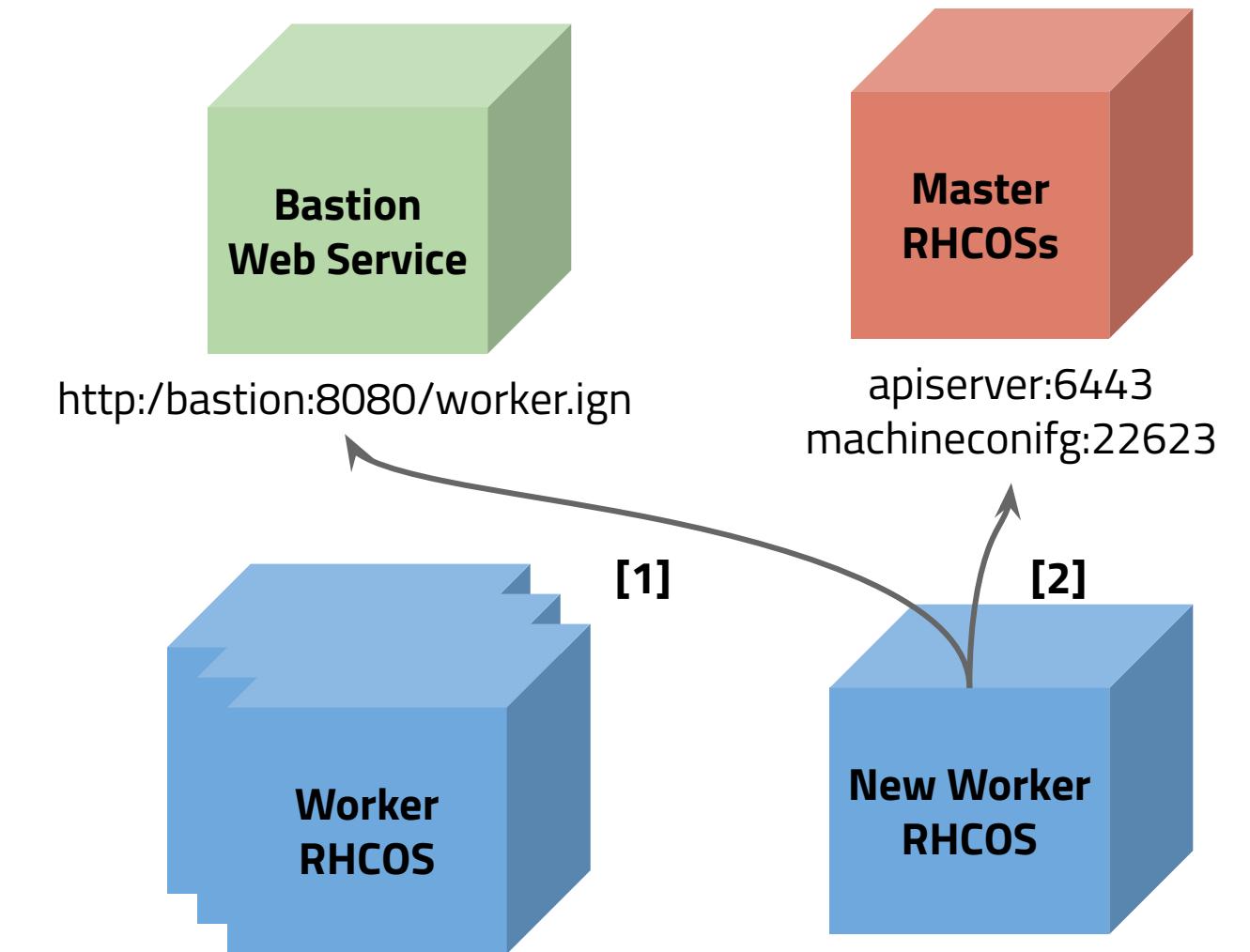


Cluster Management

Kubernetes Machine API

- Importante, custom resource administrados por el cluster
- **Manages Machines** (Nodos en la terminología de Openshift)
 - Create, update, delete machines
 - También son creados nodos de infraestructura (VM) (*aplica a Cloud Providers*)
 - En instalaciones On Premise el agregado de nuevos nodos es vía proceso de boot+Ignition file (mismo proceso de instalación)
- Usar **MachineSets** para controlar sets de maquinas
 - Ejemplo: modificar archivos de configuración dentro de los nodos RHCOS

Proceso de agregado de nuevo nodo

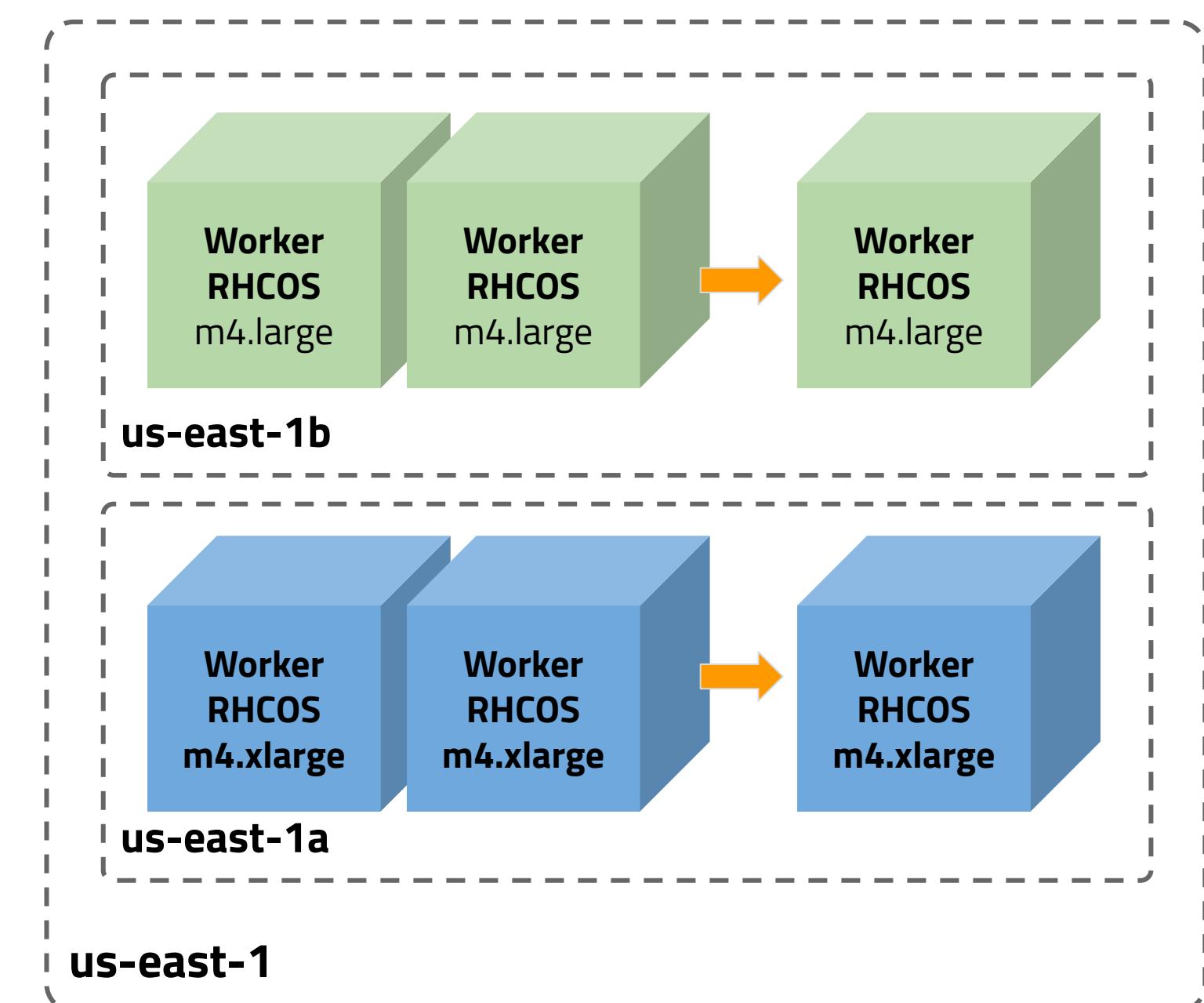


1. En el proceso de boot busca el archivo de ignition file `worker.ign`
2. Busca configuración de booteo al servicio de `MachineConfig` port 22623

Cluster Management

MachineSets

- Machineset representan:
 - Set de machines
 - Abstracción de infraestructura.
- En RHOP 4, Abstracción de AWS (EC2)
- El instalador crea MachineSets por cada each Availability Zone en una Region
- Ejemplo:
 - us-east-1 tiene seis Availability Zones: us-east-1a hasta us-east-1f
 - us-east-2 tiene tres Availability Zones: us-east-2a, us-east-2b and us-east-2c

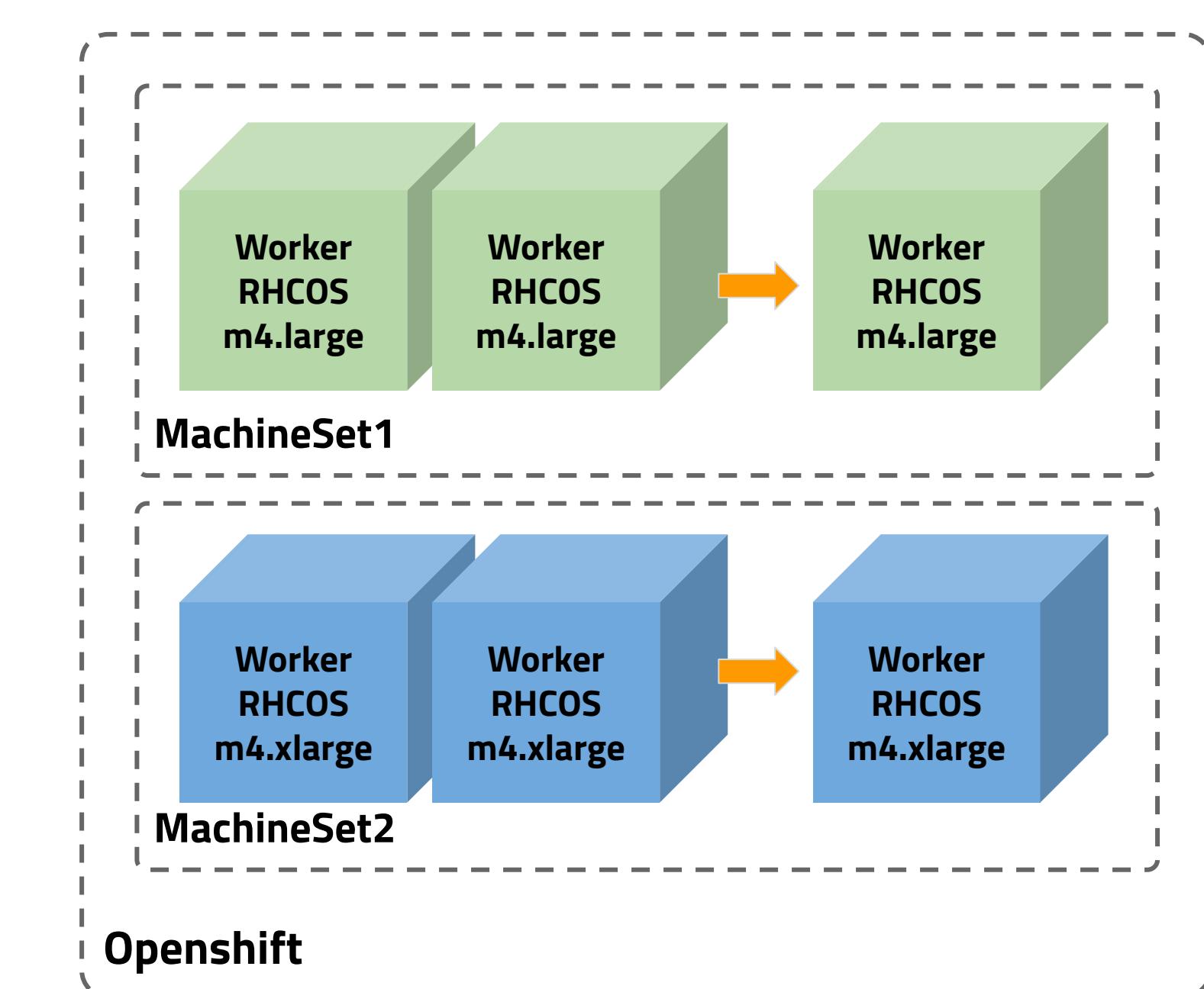


Cluster Management

Templates MachineSets

- Machinesets incluyen templates para nuevas máquinas
 - Instance types
 - Disks
 - Labels and taints
- Editar machinesets para cambiar la configuración por default de nuevas máquinas
- Crear nuevas máquinas por grupos tecnológico.
- Scale machinesets para agregar/remover maquinas del cluster

```
$ oc get machinesets -n openshift-machine-api
```



DEMO

Revisión de Machine API

Autenticación y Autorización

Semperti

Autenticación

Visión General

La mayoría de los cluster de Openshift son ambientes multi-usuario

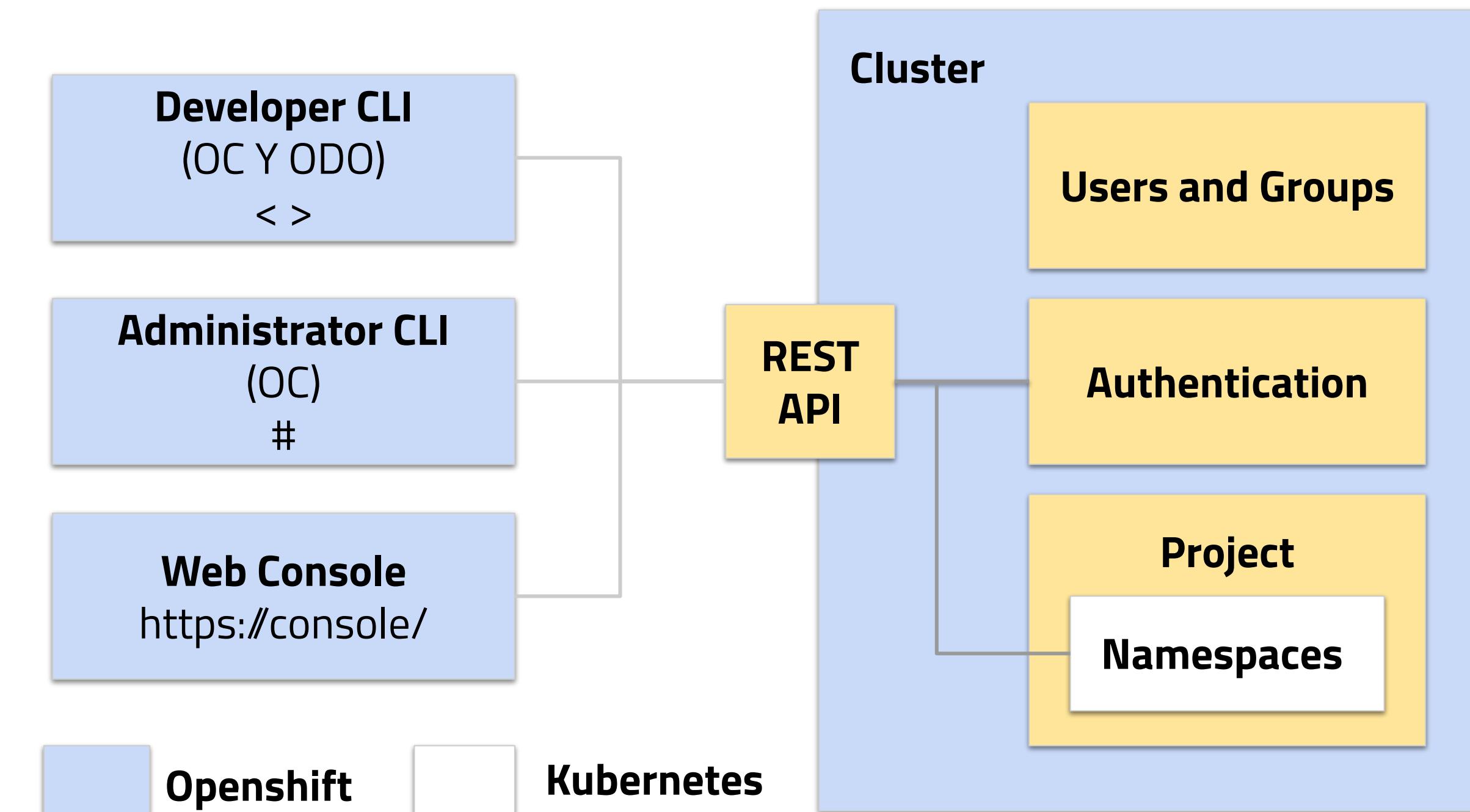
Desarrolladores	Operaciones	Seguridad	Administradores
Desarrollan y despliegan aplicaciones	Monitoreo del sistema y su estado de salud, responden incidentes.	Acceso de auditoría y seguridad de aplicación	Encargados del mantenimiento de la infraestructura

Proceso de autenticación, todos los perfiles de personas deben pasar un proceso sobre el cual deben validar su identidad en el acceso al cluster.

Autenticación

REST API

```
$ odo login -u developer -p developer <url api  
openshift>  
$ oc login -u developer -p developer <url api  
openshift>  
$ oc login -u kubeadmin -p password1! <url  
api openshift>
```

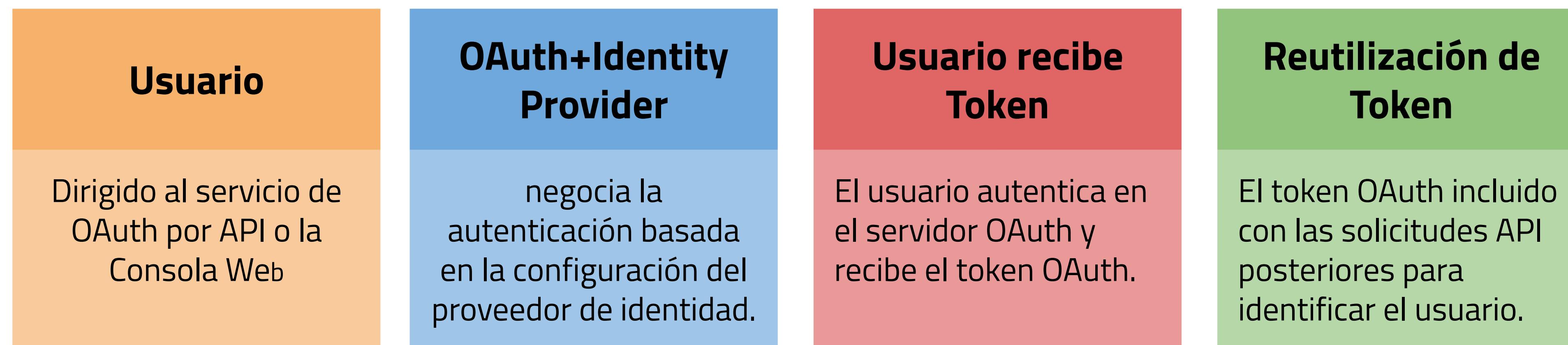


Ambiente	Web Console	REST API
NO PROD	https://console-openshift-console.apps.ocp-np.sis.ad.bia.itau	https://api.ocp-np.sis.ad.bia.itau:6443
PROD	https://console-openshift-console.apps.ocp-pa.sis.ad.bia.itau	https://api.ocp-pa.sis.ad.bia.itau:6443
PROD	https://console-openshift-console.apps.ocp-pb.sis.ad.bia.itau	https://api.ocp-pb.sis.ad.bia.itau:6443

Autenticación

Proceso de token

El proceso de autenticación en Openshift tiene los siguientes pasos:



La vida útil predeterminada del token OAuth es de 86000 segundos (24hs) después el usuario debe volver a loguearse.

Autenticación

OAuth Tokens y Authentication

OAuth Tokens

Verificación del token OAuth:

- El chequeo de validación del token es realizado por la API de OpenShift y otros componentes.
 - El token OAuth activo puede ser obtenido por **oc get oauthaccesstokens**
 - Es eliminado el token de acceso una vez finalizada la sesión.

- Usuario Conectado
 - \$ oc whoami shows current user

- Web UI
 - \$ oc whoami --show-console

- Server API
 - \$ oc whoami --show-server

- Token
 - \$ oc whoami --show-token

- Oauth Introspection
 - oc login --loglevel=9 -u USER -p PASSWORD
 - API_URL

Authentication

\$ oc login -u USER -p PASSWORD API_URL

1. Es distinta la URL de la API y la URL de la consola. API URL

<https://api.CLUSTERDOMAIN:6443>

2. Los datos de la sesión son almacenados en ~/.kube/config

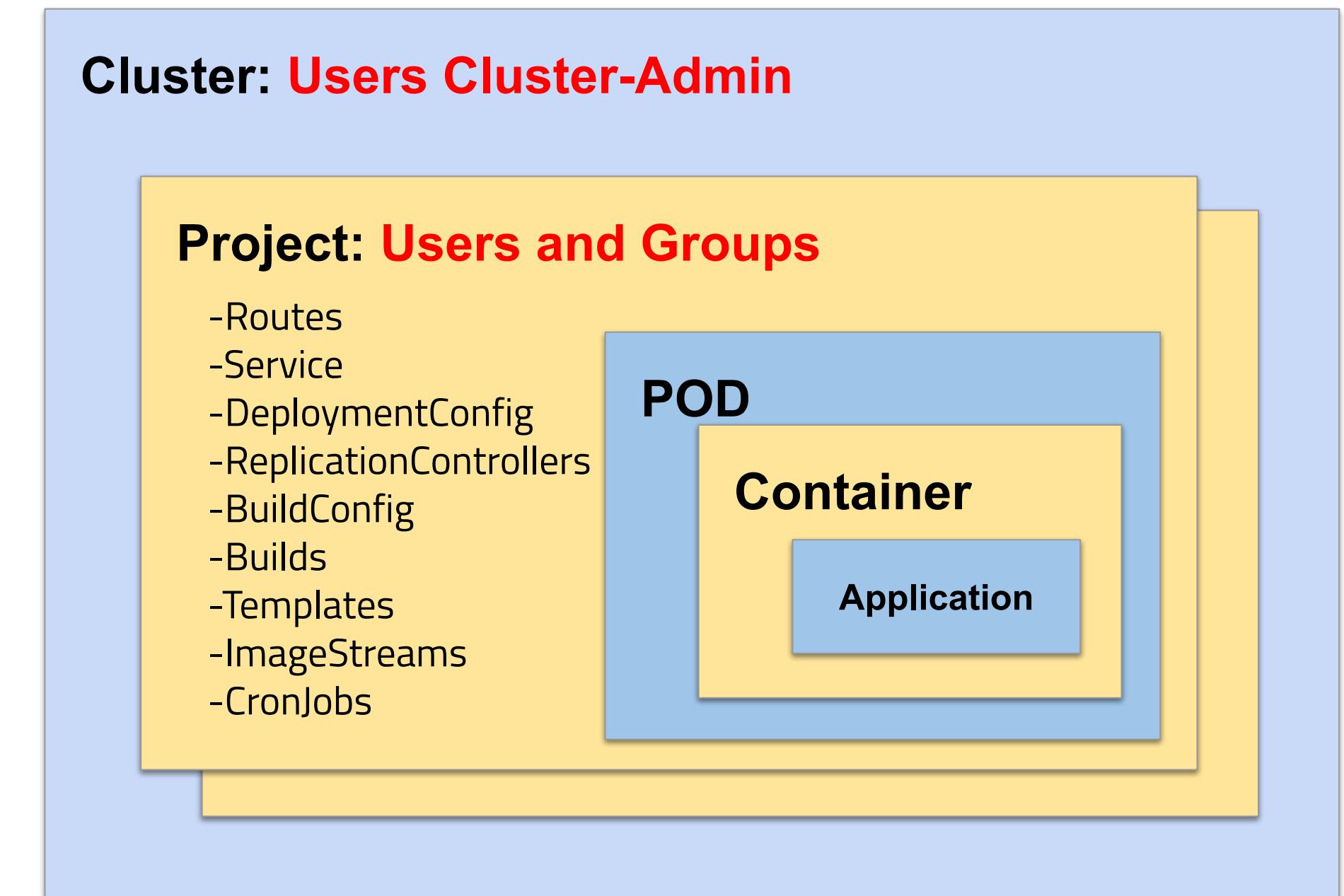
Es posible alternar entre distintos archivos kubeconfig, para esto podemos setear la ubicación en la variable de entorno KUBECONFIG o puede ser pasada en el parametro del comando con --config

Autenticación

Identity Providers

Openshift necesita reconocer la identidad de las personas para poder autenticar la API.

- **Usuarios** identificados por un **Identity Providers**:
 - HTPasswd
 - Keystone
 - LDAP
 - Basic authentication
 - Request header
 - Otros en link.
- **Grupos**, dos opciones de creación:
 - Manual.
 - **Sincronizados de manera programada** vía cron Jobs contra los grupos de AD. Se pueden utilizar whitelist.



Autenticación

Configuración de Identity Providers y Kubeadmin

- **Configuración**

Es administrado por el *cluster-authentication-operator*:

- Configurado por la instancia "cluster" del Custom Resource **oauth.config.openshift.io**
- spec.identityProviders lists additional methods for user authenticationspec.tokenConfig.accessTokenMaxAgeSeconds configura la vida útil de las sesiones.

- **Kubeadmin**

kubeadmin (kube:admin) user provisto inicialmente por el cluster para fines administrativos.

- Deshabilitar de kubeadmin user una vez configurado el cluster.
- La password del usuario kubeadmin es almacenada como un secrets en el namespace kube-system.
- El usuario kubeadmin debe ser eliminado luego de haber otorgado permisos de cluster-admin a un equipo de administradores.

```
$ oc delete secret kubeadmin -n kube-system
```

Group Management

ABM y Recomendaciones de Administración

Los grupos hacen que RBAC haga sentido.

- Un usuario no debería tener una visión completa del cluster, debe pertenecer a un grupo y el grupo debe tener permisos administrativo.
- OK: Grupo "security-audit" tiene acceso de vista full sobre todo el cluster.
 - Práctica recomendada para grupos es representar roles organizacionales dentro de Openshift
 - Ejemplos: Examples of groups:
 - Application development teams, team leads, quality assurance
 - Platform administrators, security, operations

Los grupos pueden ser administrados de manera **manual o automatizada**.

- Un proceso automatizado puede mantener los grupos sincronizados
- Un proceso manual solo es requerido cuando un proceso automatizado no puede aplicarse.
 - Para poder administrar grupos es necesario tener permisos administrativos de cluster-admin.
 - La administración de grupos no puede ser delegadas a usuarios que no son cluster-admin

Las acciones que pueden ser aplicadas a la hora de crear grupos son las que siguen.

- Listar grupos y miembros
\$ oc get groups

- Crear nuevos grupos
\$ oc adm groups new GROUP

- Agregar usuario
\$ oc adm groups add-users GROUP USER

- Remover un usuario
\$ oc adm groups remove-users GROUP USER

- Borrar Grupo
oc delete group GROUP

Group Management

System groups

Grupos de sistema predeterminados. Grupos virtuales incorporados sin definición de recurso de grupo asociado.

Group	Description
system:authenticated	Authenticated users
system:authenticated:oauth	Users authenticated with OAuth
system:cluster-admins	Built-in cluster administrators such as kubeadmin system:admin
system:serviceaccounts	All OpenShift service accounts
system:serviceaccounts:NAMESPACE	All OpenShift service accounts in specific project namespace
system:unauthenticated	Anonymous unauthenticated requests

Cluster Management

Troubleshooting

- **system:admin** user bypasses OAuth authentication

- system:admin no usa login de consola
- Autenticación es realizada por TLS
- Las credenciales TLS son almacenadas en el archivo kubeconfig creado durante el proceso de instalación.
- Almacenar un copia de archivo en un lugar seguro, es de uso administrativo y tiene acceso full al cluster

```
$ export KUBECONFIG=OCP4_INSTALL_DIR/auth/kubeconfig
```

```
$ oc config use-context admin
```

```
$ oc whoami
```

```
system:admin
```

- **Logs** proyecto openshift-authentication

```
$ oc get pods -n openshift-authentication
```

```
$ oc logs -n openshift-authentication-operator deployment/authentication-operator
```

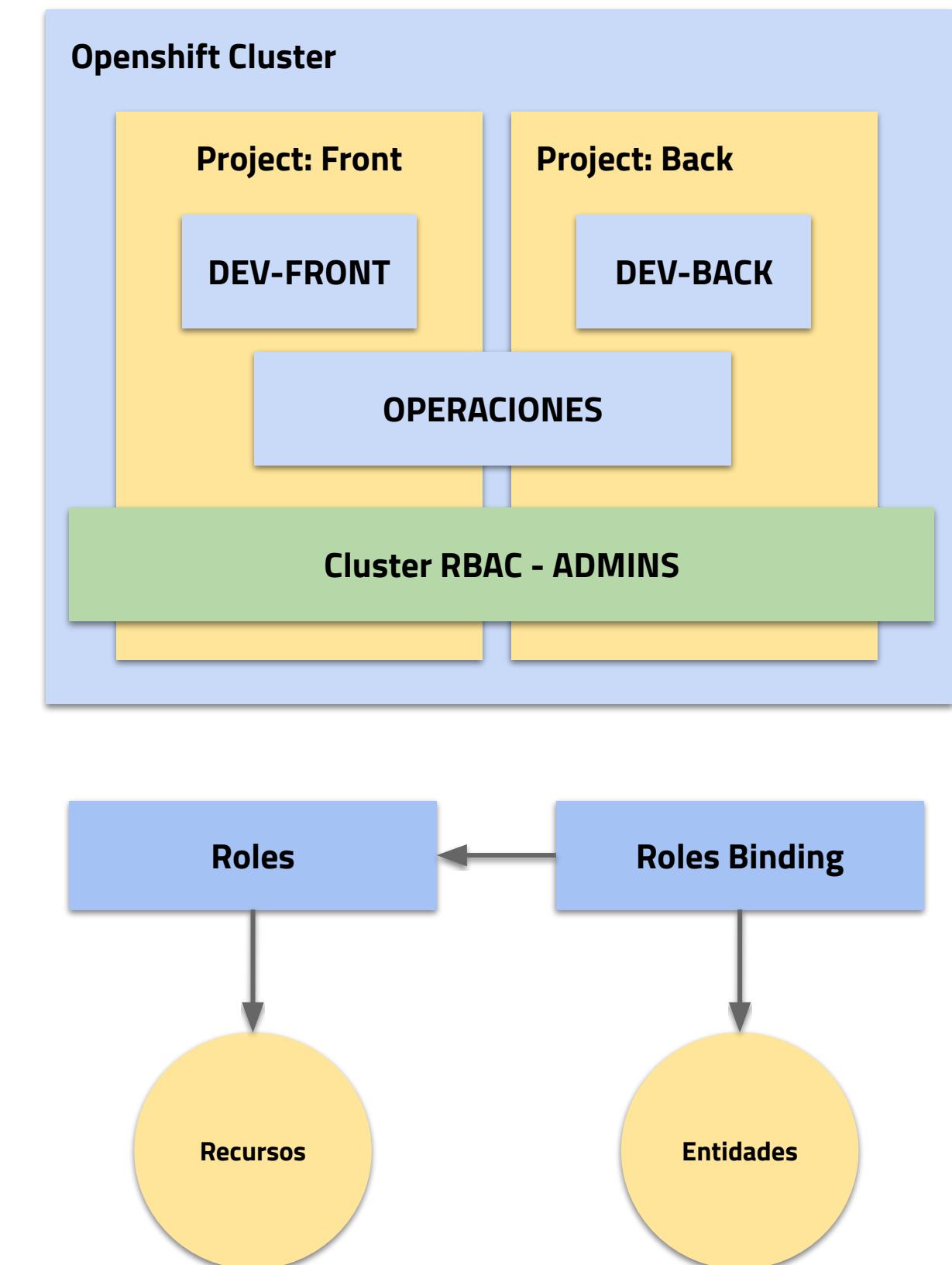
```
$ oc logs -n openshift-authentication oauth-openshift-564976c856-ffhwr
```

Autorización

Vision General

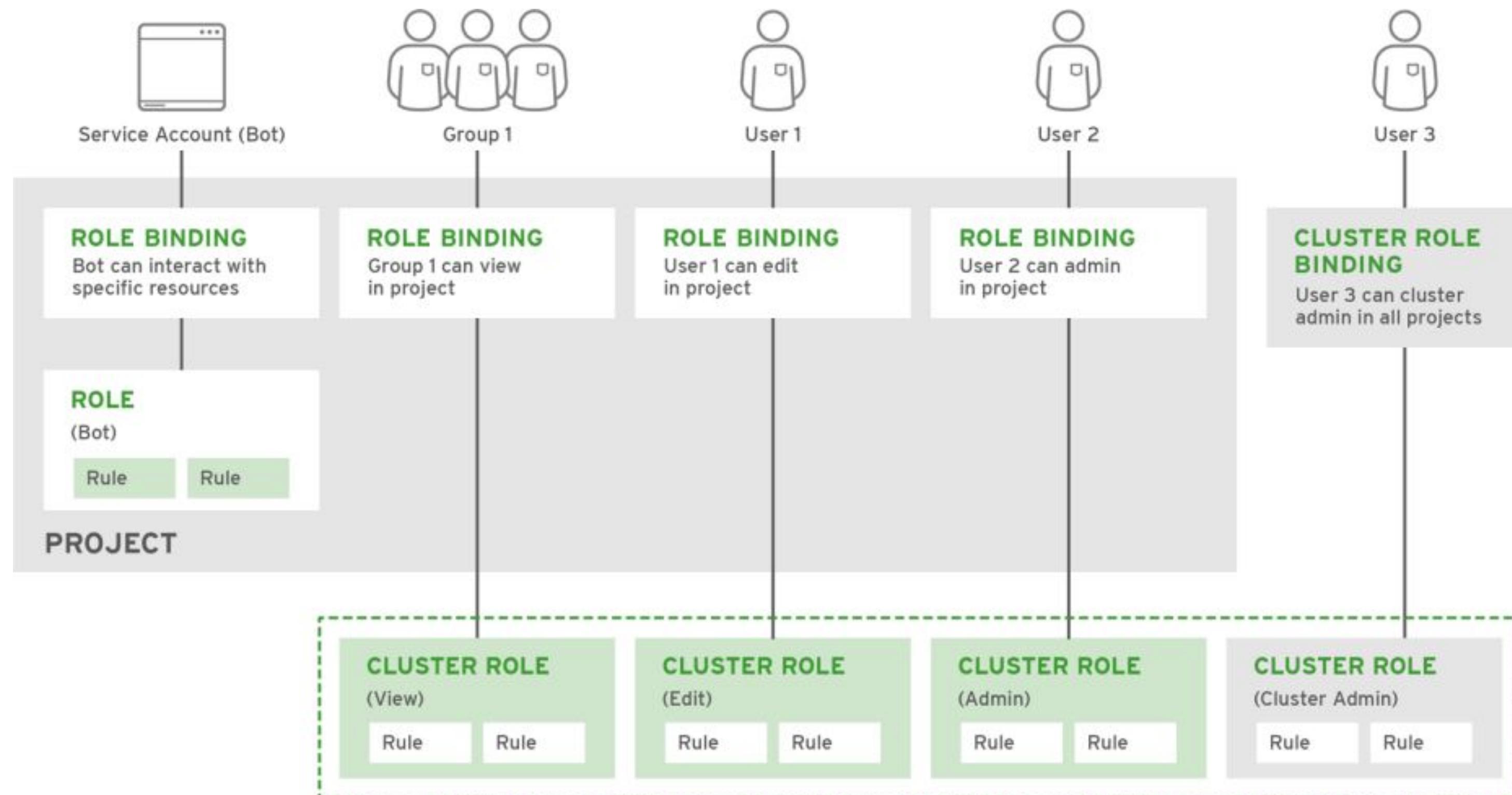
Logs objetos RBAC determinan si el usuario puede o no realizar una acción específica con respecto al tipo de recurso. Deny by default, es la política de acceso predeterminadas por Openshift RBAC.

- **Roles:** mapa de acciones permitidas (verbos) a recursos de un proyecto. (LIST pods)
- **ClusterRoles:** mapa de acciones permitidas (verbos) a recursos de cluster como recursos en un proyecto. Alcance todos los proyectos. (REMOVE nodes).
- **RoleBindings:** concede acceso asociando Roles o ClusterRoles a usuarios o grupos para acceder dentro un namespace.
- **ClusterRoleBindings:** concede acceso asociando ClusterRoles a usuarios o grupos para acceder a recursos o recursos de tipo cluster en cualquier namespace.



Autorización

Vision General



TIP: El usuario con acceso para crear RoleBindings o ClusterRoleBindings puede otorgar acceso, un usuario no puede otorgar acceso que no tiene

Autorización

Roles and Rules

- **Roles, cluster roles** = colección de reglas (rules)
- **Rule** = lista de verbos permitidos. (create, delete, get, list, patch, update)
 - El sujeto de la regla (RULE subject) son generalmente grupos de recursos de API (api-groups)
 - Se pueden listar los recursos y grupos de apis (api-groups) con oc api-resources El sujeto puede estar limitado a nombres de recursos específicos
- **Describe roles**
\$ oc get clusterrole
\$ oc describe clusterrole -o yaml

Verb	Description
create	Create resource
delete	Delete resource
deletecollection	Delete collection of resources
get	Get resource
list	Get multiple resources
patch	Apply patch to change resource
update	Update resource
watch	Watch for changes on websocket

Autorización

Roles and Rules

```
$ oc describe clusterrole basic-user
Name:      basic-user
Labels:    <none>
Annotations: openshift.io/description: A user that can get basic information about projects.
            rbac.authorization.kubernetes.io/autoupdate: true
PolicyRule:
  Resources           Non-Resource URLs  Resource Names  Verbs
  selfsubjectrulesreviews []                []             [create]
  selfsubjectaccessreviews.authorization.k8s.io []                []             [create]
  selfsubjectrulesreviews.authorization.openshift.io []                []             [create]
  clusterroles.rbac.authorization.k8s.io     []                []             [get list watch]
  clusterroles          []                []             [get list]
  clusterroles.authorization.openshift.io     []                []             [get list]
  storageclasses.storage.k8s.io              []                []             [get list]
  users                 []                [~]            [get]
  users.user.openshift.io                   []                [~]            [get]
  projects               []                []             [list watch]
  projects.project.openshift.io             []                []             [list watch]
  projectrequests         []                []             [list]
  projectrequests.project.openshift.io      []                []             [list]
```

Managing Cluster Roles and Roles

Role-Based Access Control

Agregar Role Bindings en un namespace

Add cluster role to user to manage resources in namespace:	<code>oc policy add-role-to-user CLUSTER_ROLE USER -n NAMESPACE</code>
Add namespace role to user to manage resources in namespace:	<code>oc policy add-role-to-user ROLE USER -n NAMESPACE --role-namespace=NAMESPACE</code>
Add cluster role to group to manage resources in namespace:	<code>oc policy add-role-to-group CLUSTER_ROLE GROUP -n NAMESPACE</code>
Add namespace role to group to manage resources in namespace:	<code>oc policy add-role-to-group ROLE GROUP -n NAMESPACE --role-namespace=NAMESPACE</code>

Remover Role Bindings en un namespace

Remove cluster role from group in namespace	<code>oc policy remove-role-from-group CLUSTER_ROLE GROUP -n NAMESPACE</code>
Remove namespace role from group in namespace	<code>oc policy remove-role-from-group ROLE GROUP -n NAMESPACE --role-namespace=NAMESPACE</code>
Remove all role bindings for group in namespace	<code>oc policy remove-user GROUP -n NAMESPACE</code>

Managing Cluster Roles and Roles

Role-Based Access Control

Remove Role Bindings in a namespace

Remove cluster role from user in namespace	oc policy remove-role-from-user CLUSTER_ROLE USER -n NAMESPACE
Remove namespace role from user in namespace	oc policy remove-role-from-user ROLE USER -n NAMESPACE --role-namespace=NAMESPACE
Remove all role bindings for user in namespace	oc policy remove-user USER -n NAMESPACE

Cluster Role Binding Management

Add cluster role to user	oc adm policy add-cluster-role-to-user CLUSTER_ROLE USER
Add cluster role to group	oc adm policy add-cluster-role-to-group CLUSTER_ROLE GROUP
Remove cluster role from user	oc adm policy remove-cluster-role-from-user CLUSTER_ROLE USER
Remove cluster role from group	oc adm policy remove-cluster-role-from-group CLUSTER_ROLE GROUP

Service Account (SA) Security Context Constraints (SCC)

Semperti

Service Account

Vision General

Los **service account** son identidades non-person para integración de aplicaciones.

Tareas:

- Cada pod es ejecutado con un service account
- Usado por agentes externos para acceder al cluster API
- Acceso administrativo con role bindings y cluster role bindings.
 - DeploymentConfigs usan el pod deployer que corre con el service account *deployer*
 - Aplicaciones en contenedores de pods hacen llamadas a la API.
 - Jenkins servers usan API para crear pods con el agente de jenkins y son ejecutados con un service account jenkins
 - Operadores usan cluster API para observar los custom resource y la API para manejar ConfigMaps, deployments, etc

Name	Description
builder	Service account usado para build pods, push images
deployer	Service account usado para desplegar pods, implementar DeploymentConfig rollout, rollback.
default	Service account usado para los pods.

Create	oc create serviceaccount NAME -n NAMESPACE
List	oc get serviceaccount NAME -n NAMESPACE
Describe	oc describe serviceaccount NAME -n NAMESPACE
Delete	oc delete serviceaccount NAME -n NAMESPACE

Service Account

Service Account Token

Premisas claves

- Service account usan tokens para autenticar contra la API del cluster.
- En un contenedor que se encuentra ejecutando encontramos el token en el archivo /run/secrets/kubernetes.io/serviceaccount/token.
- Los token son usados por agentes externos para actuar dentro del cluster con el rol del service account.
- Permite el acceso externo a la registry por parte de agentes.
- Permite a un servidor de jenkins poder correr agentes que serán temporalmente jenkins-slave.

Listar los service account token

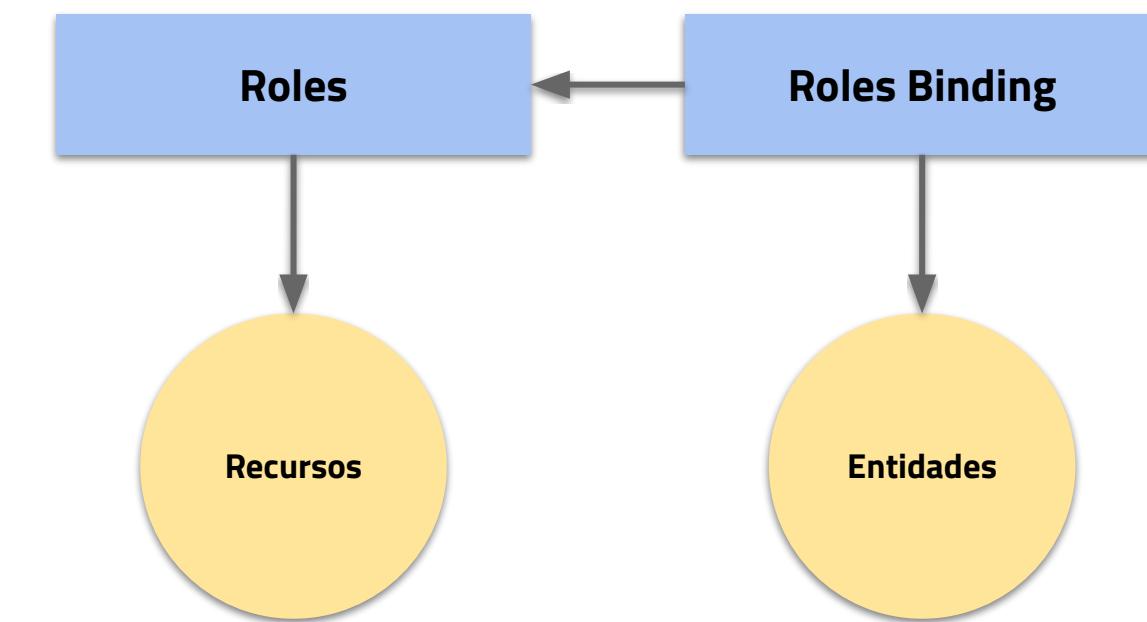
```
oc get secret --field-selector=type=kubernetes.io/service-account-token -n NAMESPACE
```

Obtener el token activo del service account

```
oc serviceaccount get-token SERVICE_ACCOUNT -n NAMESPACE
```

TIP: Por defecto no existe rotación de los token de los service account, para poder rotarlo hay que eliminarlo y volver a crearlo.

RBAC igual que para un usuario o grupo



Security Context Constraints

SCC

Role Base Access Control (RBAC) permite que es lo que tienen que hacer los usuarios, en contraposición, los **Security Context Constraints (SCCs)**, nos permiten controlar las acciones que toman los pods.

- Acciones que pueden realizar los pods.
- Donde pueden acceder los pods.

SCC definen las condiciones con las que deben ser ejecutados los pods para que sean aceptados en el sistema. Los SCCs le permiten al administrador tener control sobre:

- Agrega ciertas características al contenedor antes de ser agregado al sistema
- uso de directorios de host como volúmenes
- Contexto de SELinux
- User ID
- Uso de host namespaces y networking
- Asignación de FSGroup para volúmenes de pods
- Configuración de grupos suplementarios permitidos
- Uso de root filesystem como RO.
- Uso de tipos de volúmenes.
- Configuración de perfiles Secure Computing Mode (SECCOMP) permitidos.

Security Context Constraints

SCC CLI

- **Add SCC User**

```
oc adm policy add-scc-to-user SCC_NAME USER_NAME
```

- **Add SCC Group**

```
oc adm policy add-scc-to-group SCC_NAME  
GROUP_NAME
```

- **Remove SCC User**

```
oc adm policy remove-scc-from-user SCC_NAME  
USER_NAME
```

- **Remove SCC Group**

```
oc adm policy remove-scc-from-group SCC_NAME  
USER_NAME
```

SCC	Description
anyuid	Allow containers to run as any user ID, <i>including</i> root user (uid=0)
hostaccess	Allow containers to access host file systems, network, and process table with restricted user ID
hostmount-anyuid	Allow containers to access host file system using host mounts, run as any user ID
hostnetwork	Allow containers access to host networking, host ports
node-exporter	Reserved for use by Prometheus node exporter
nonroot	Allow containers to run as any user ID <i>except</i> root user (uid=0)
privileged	Allow access to all privileged and host features—most relaxed access, use only for cluster administration
restricted	Deny access to all host features, require pod containers to run with restricted UID (default SCC)

Día 5

Semperti

Overview

- Controller and Scheduler
- Resource Management
- Cluster Monitoring
- Cluster Logging
- Networking
- Openshift HA

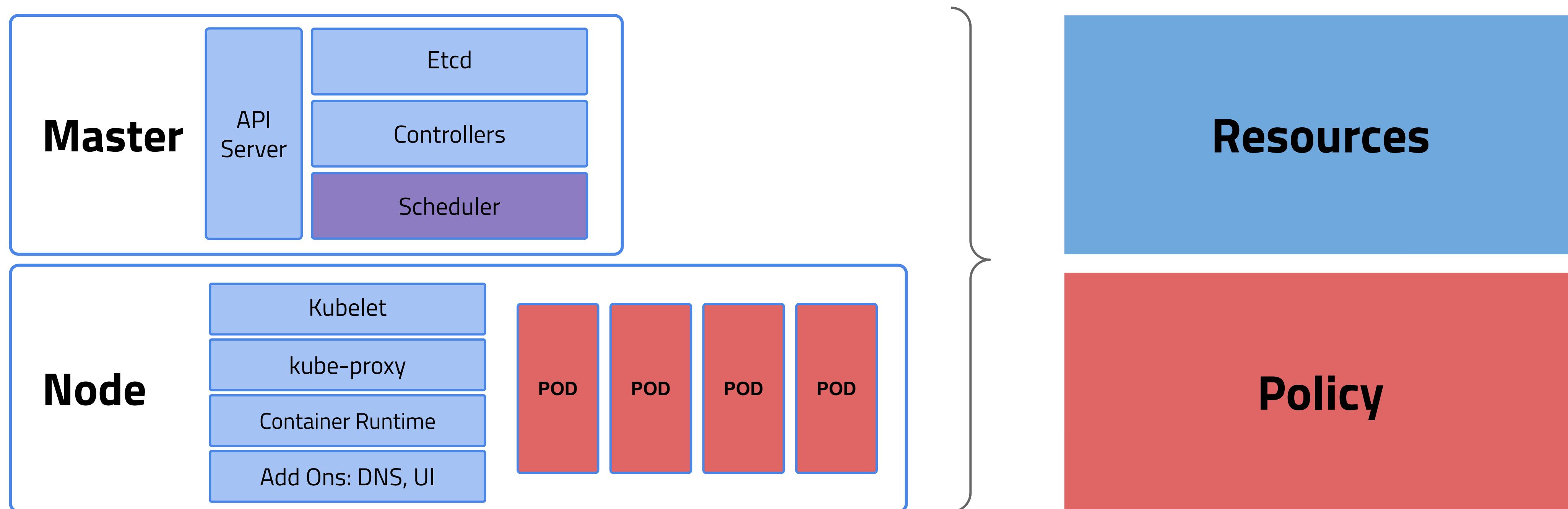
Controller and Scheduler

Semperti

Controller and Scheduler

Scheduling in Openshift

El único trabajo que tiene kubernetes es realizar el startup de pod



Controller and Scheduler

Scheduling Process

El subprocesso de scheduler observa el API Server y verifica Unscheduled Pods

Selección del nodo

Update *nodeName* en el objeto Pod

Kubelet observa el API Server por trabajos

Container Runtime toma la señal desde kubelet para poder instanciar el/los pods

Controller and Scheduler

Node Selection

Filtering

1. Identificar de todos los nodos
2. Aplicar filtros
3. Obtener nodos filtrados
4. Hard constraints

Scoring

1. Funciones de Scoring
2. Nodos viables.
3. Policy constraints

Binding

1. Selected node list.
2. Ties are broken.
3. Update API Object.

Controller and Scheduler

Resource Request

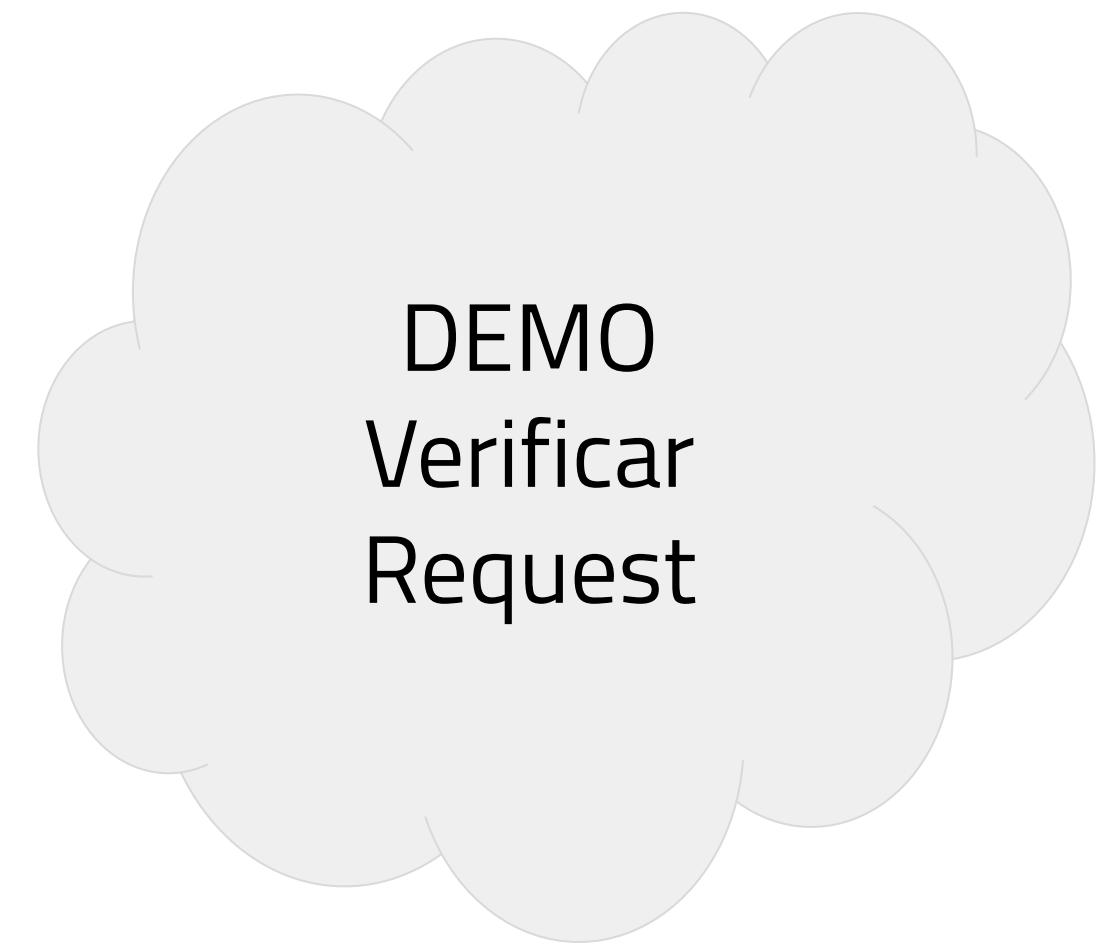
Configurando el parámetro `request`, permitirá al scheduler buscar un nodo que pueda alojar la carga de trabajo requerida por el pod.

Definiendo el valor de `request` garantizamos los recursos de:

- CPU
- Memoria

Que serán alocados en el nodo.

Los pods que necesiten ser alojados en un nodo, pero no tengan la suficiente cantidad de recursos disponibles quedarán en estado *pending*.



Controller and Scheduler

Controlling Scheduling

Node Selector

Affinity

Taint and
Toleration

Node Cordoring

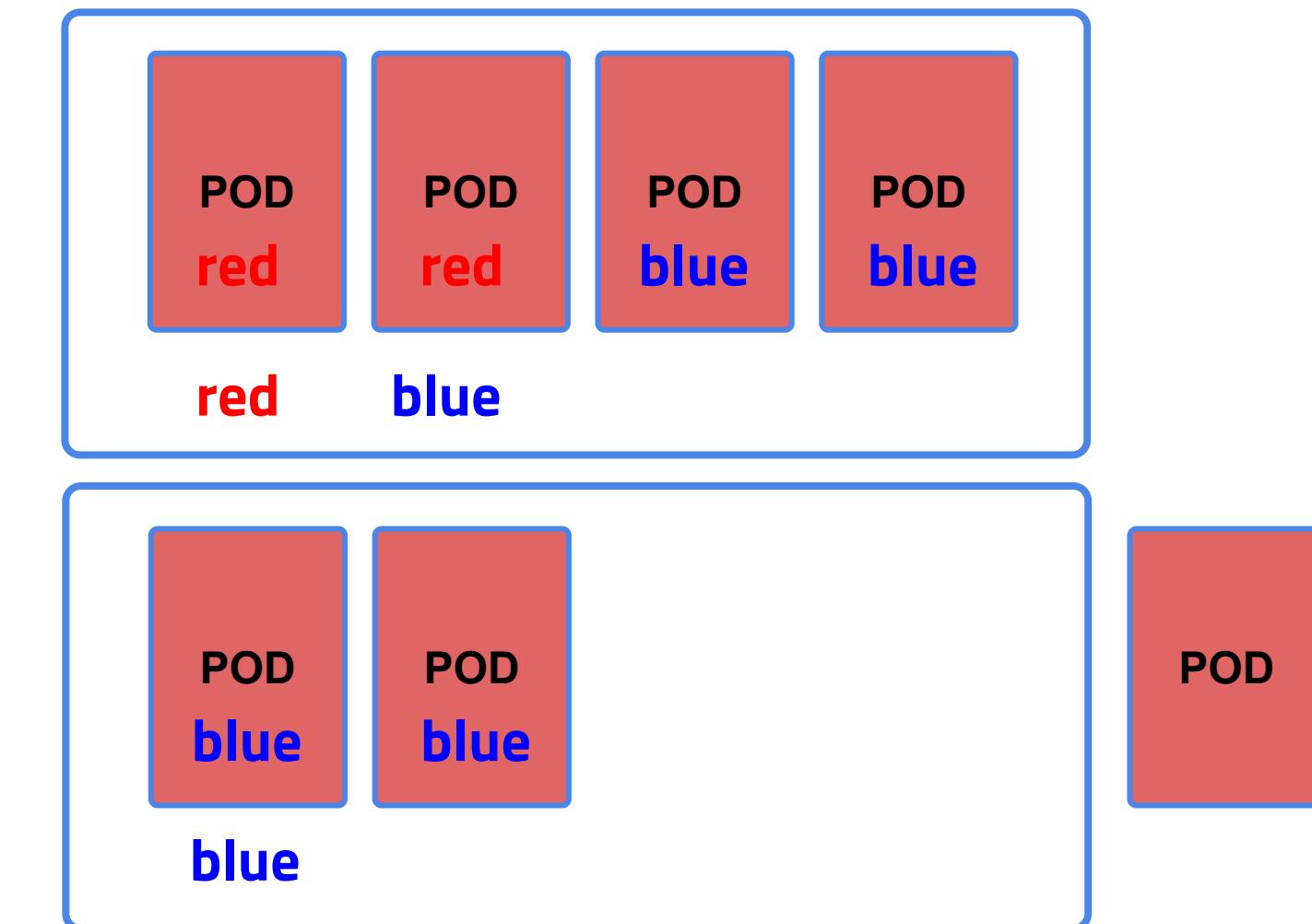
Manual
Scheduling

Controller and Scheduler

Node Selector

Postulados de la selección de nodos

- `nodeSelector` asigna pods a nodos usando Labels y Selectors
- Aplicar Labels a los nodos
- Scheduler será el que asignará Pods a nodos que realicen el matching de labels.
- Chequeo simple de key/value sobre `matchLabels`.
- El mapeo de pods y nodos es realizados en base a:
 - Especiales requerimientos de hardware.
 - Aislar de cargas de trabajo.
- El pod que tenga un label y no coincida con ninguno en los nodos quedará en estado de pending.
- El pod que no tenga definido `nodeSelector`, será alojado en cualquier nodo.



Controller and Scheduler

Node Selector

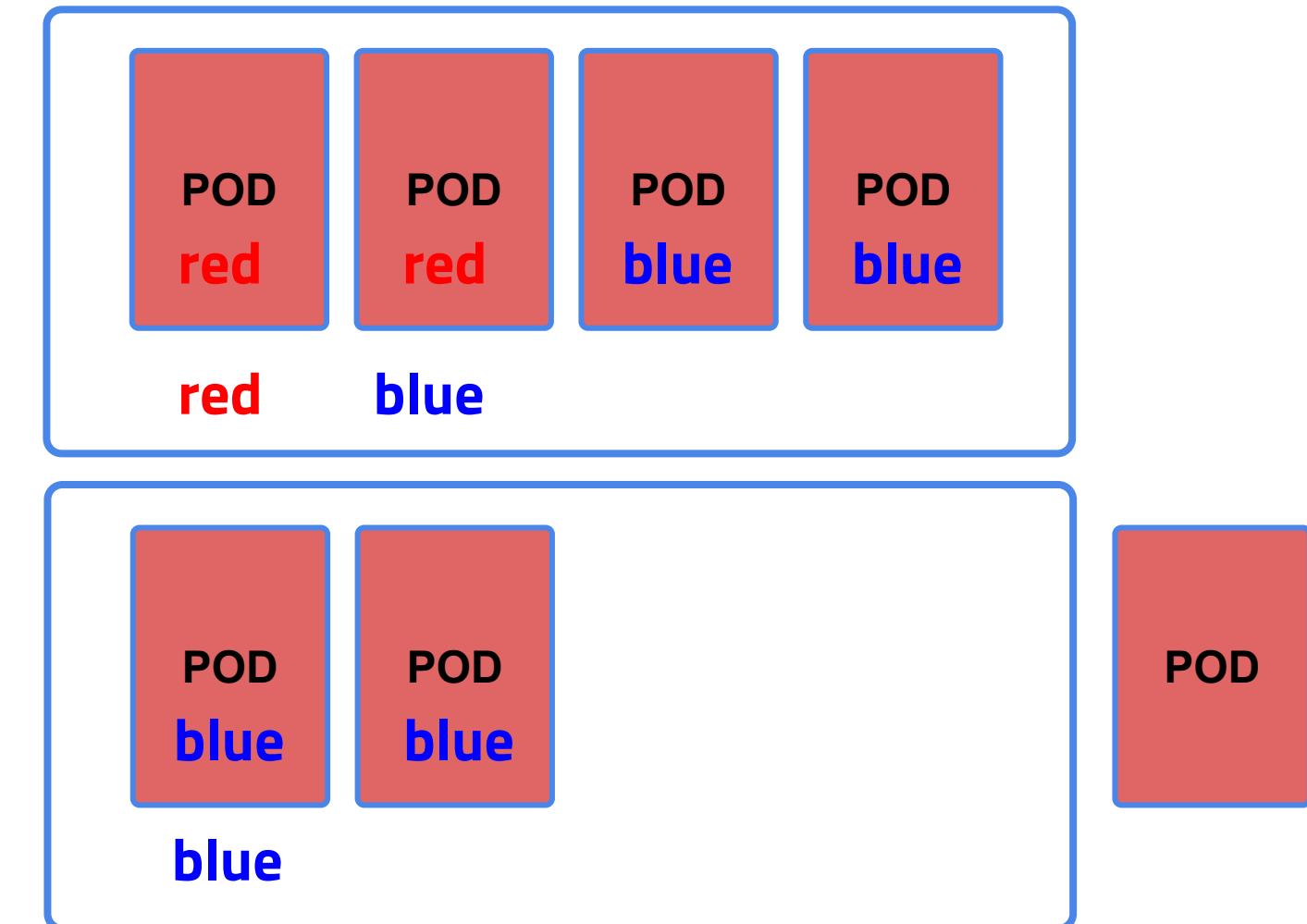
Asignar pods a nodos usando nodeSelector

1. Etiquetar los nodos worker

```
$ oc label node worker1 zone=rojo
```

2. Definir en la especificación del deployment, dc, pod.

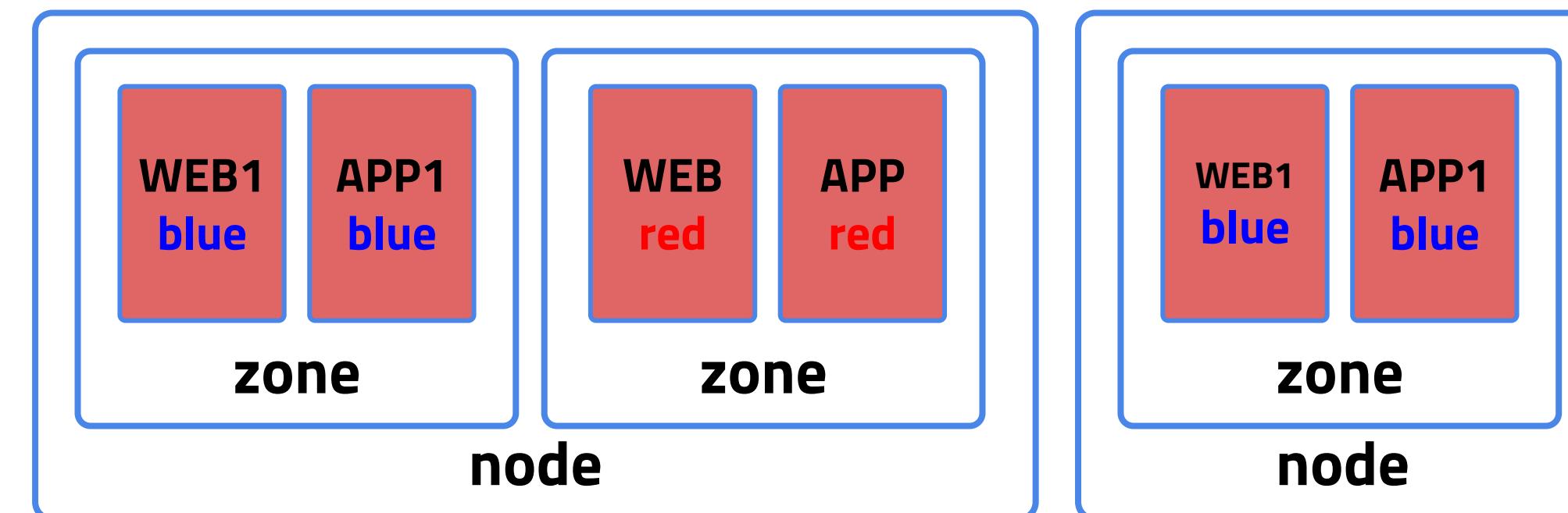
```
spec:  
  containers:  
    - name: hello-world  
      image: gcr.io/google-samples/hello-app:1.0  
      port:  
        - containerPort: 8080  
nodeSelector:  
  zone: red
```



Controller and Scheduler

Affinity and Anti-Affinity

- **nodeAffinity** usa labels en los nodos para tomar una decisión con matchExpressions.
 - requiredDuringSchedulingIgnoredDuringExecution
 - preferredDuringSchedulingIgnoredDuringExecution
- **podAffinity**: programa los pods sobre un mismo nodo, crea zonas con otro pod.
- **podAntiAffinity**: programa los pods sobre un nodo diferente al definido en la regla, crea zonas con otros pod.



```
spec:  
containers:  
- name: hello-world-cache  
...  
affinity:  
podAffinity:  
  
requiredDuringSchedulingIgnoredDuringExecution:  
- labelSelector:  
  matchExpressions:  
  - key: app  
    operator: In  
    values:  
    - hello-world-web  
topologyKey: "kubernetes.io/hostname"
```

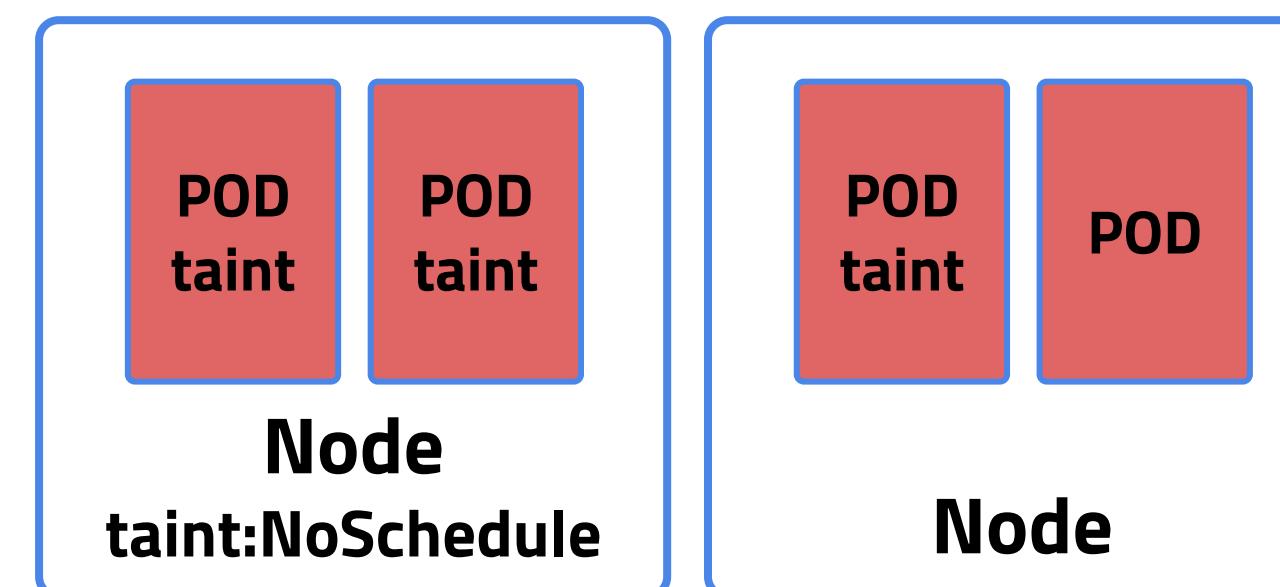
Controller and Scheduler

Taints and Tolerations

- **Taints:** habilidad para controlar cuales pods son programados a los nodos.
- **Tolerations:** permite que un pod ignore un Taints y puede ser programado con normalidad sobre nodos que están marcados como Taints.

Usados en escenarios donde los cluster admin necesitan poder desplegar pods sin ninguna dependencia de las reglas de usuario.

key=value:effect



```
kubectl taint nodes node1  
key=MyTaint:NoSchedule  
  
spec:  
  containers:  
    - name: hello-world  
      image:  
        gcr.io/google-samples/hello-app:1.0  
      ports:  
        - containerPort: 8080  
  tolerations:  
    - key: "key"  
      operator: "Equal"  
      value: "MyTaint"  
      effect: "NoSchedule"
```

Controller and Scheduler

Node Cordon, Node Drain and Manually Scheduler

Node Cordon

- Marca los nodos como unschedulable
- Previene que nuevos nodos sean programados en un nodo
- No afecta a los nodos existentes.
- Usado para preparar un nodo antes de reiniciarlo.

```
oc cordon node1
```

Node Drain

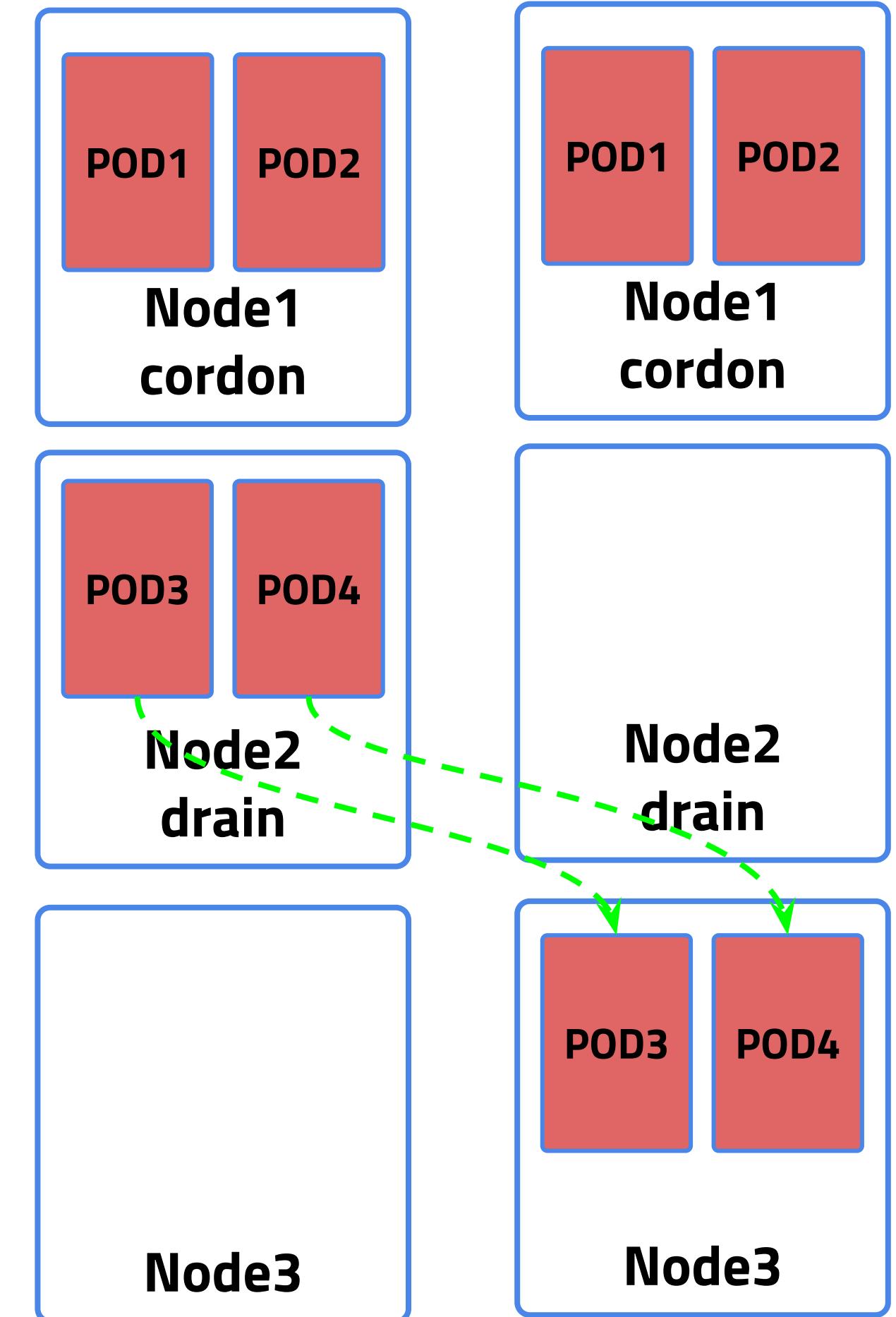
Si es necesario evacuar los pods de un nodo

```
oc drain node1 --ignore-daemonsets
```

Manually Scheduler

Puede ser forzado el alojado de un pod en un nodo por medio de la directiva nodeName en la definición del pod.

El nombre de nodo debe existir.



Resource Management

Semperti

Resource Management

Vision General

¿Qué son?

Son mecanismos para ejercer controles y límites sobre los recursos disponibles en el clúster. Que tiene como propósito asegurar que las cargas de trabajo puedan ejecutarse de manera confiable y predecible.

¿Por qué es necesario?

Para evitar que los usuarios con buenas intenciones sean malos vecinos, que usuarios malintencionados afecten otras cargas de trabajo. Permitir un rendimiento predecible de la aplicación en todos los entornos

Controles

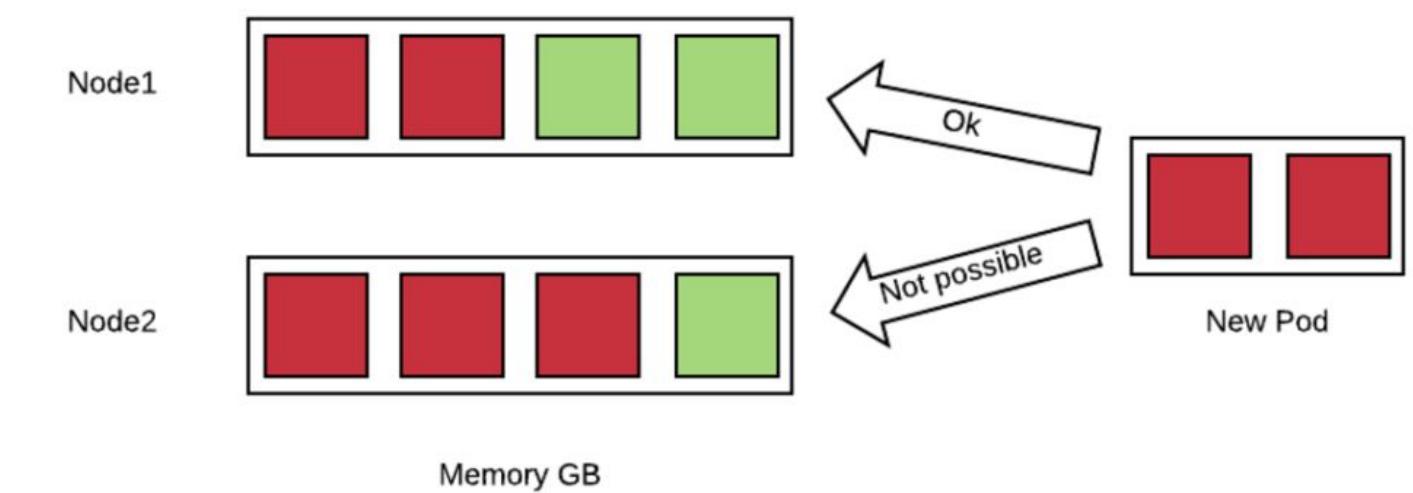
Objetos

Recursos

Resource Management

Tipos de recursos

- **Objetos**
 - Pods, replication Controller / replicaSets, service, etc
- **Recursos**
 - CPU
 - Memoria
 - Storage
- **Controles:**
 - **Request**
 - Cantidad mínima de recursos que un contenedor necesita para ejecutarse
 - **Limits**
 - Cantidad máxima de recursos que un contenedor puede consumir
 - **Limit Range**
 - **Resource Quota**



Resource Management

CPU Request

- **CPU Request**

Cantidad mínima de CPU requerida para que se ejecute el contenedor

- Usar menos no afecta la disponibilidad o el rendimiento

- **CPU Limits**

La cantidad máxima de CPU puede consumir un pod.

- Usar más provoca que el sistema finalice el contenedor

- **Request and Limits** asignados, basados en milicores

- Milicore = unidad de medida de la CPU, representada como m
- Basado en el valor de 1000m por CPU
- Ejemplo: la mitad de la CPU == 500

```
$ oc describe pod my-example-1-ntdfw
Name:      my-example-1-ntdfw
Namespace:  my-project
Start Time: Tue, 14 May 2019 14:02:54 -0700
Labels:    app=my-example
Status:    Running
IP:        10.129.2.15
Controlled By: ReplicationController/my-example-1
Containers:
my-example:
  Container ID:
  Image:      nginx
  Ports:      8080/TCP, 8443/TCP
  State:     Running
  Ready:     True
  Restart Count: 0
Limits:
  cpu: 300m
  memory: 200Mi
Requests:
  cpu: 200m
  memory: 100Mi
```

Resource Management

Memory Request

- **Memory Request:** cantidad mínima de memoria requerida para que se ejecute el contenedor
 - Usar menos no afecta la disponibilidad o el rendimiento
- **Memory Limits:** la cantidad máxima de memoria puede consumir un pod.
 - El uso de más causa que el contenedor termine y (opcionalmente) se reinicie
 - Memoria medida con Mi, Gi, etc.

TIP: ¡Cómo se comporta el sistema cuando excede los límites es una distinción importante entre la memoria y los recursos de la CPU!

```
$ oc describe pod my-example-1-ntdfw
Name:      my-example-1-ntdfw
Namespace:  my-project
Start Time: Tue, 14 May 2019 14:02:54 -0700
Labels:    app=my-example
Status:    Running
IP:        10.129.2.15
Controlled By: ReplicationController/my-example-1
Containers:
my-example:
  Container ID:
  Image:      nginx
  Ports:      8080/TCP, 8443/TCP
  State:      Running
  Ready:     True
  Restart Count: 0
Limits:
  cpu: 300m
  memory: 200Mi
Requests:
  cpu: 200m
  memory: 100Mi
```

Resource Management

Quality of Service (QoS)

QoS Classes

- Clase de QoS asignada a pod basada en request y limits
- Ayuda al scheduler a determinar donde debe ejecutarse un pod

Clases de QoS

- **BestEffort:** Aplicado cuando el contenedor request y limits no están definidos
- **Burstable:** Aplicado cuando request está definido y limits no está definido, el pod puede consumir cuanto quiera.
- **Guaranteed:** Aplicado cuando request y limits son iguales.

Memory Guaranteed y CPU Burstable

Limits:

cpu: 300m

memory: 200Mi

Requests:

cpu: 200m

memory: 200Mi

Resource Management

Comportamiento de CPU y Memoria

Comportamiento de QoS de la CPU

BestEffort:

- Container utiliza la mayor cantidad de CPU posible, pero con la prioridad más baja

Burstable:

- Garantizada cantidad mínima de CPU, puede usar más con menor prioridad

Guaranteed:

- Garantizada y limitada a la cantidad especificada de CPU que no puede exceder

QoS de memoria

Best Effort

- El contenedor utiliza tanta memoria como sea posible.
- Mínimo no garantizado
- Máxima prioridad para terminar para liberar memoria del anfitrión

Burstable

- Garantizada la cantidad mínima de memoria
- Puede usar tanto como esté disponible
- Finaliza después de los contenedores BestEffort para liberar memoria del host

Guaranteed

- Garantizada y limitada a la cantidad especificada de memoria
- Terminado (opcionalmente reiniciado) si excede la cantidad

Resource Management

LimitRange

Visión general

- Se basa en la funcionalidad de request y limits.
- Especifica la cantidad mínima y máxima de recursos que puede solicitar:
 - Pods y contenedores.
 - Imágenes y flujos de imágenes
 - Persistent Storage Claim (PVC)
- Se pueden definir valores predeterminados, por ejemplo:
 - Controle cuán grande (limits) o pequeño (request) puede ser un contenedor
 - Agregar valores predeterminados para ser injectados en la especificación del pod cuando no se especifica

Resource Management

LimitRange

Uso de LimitRange común.

En pod y contenedores

- *Cantidad máxima* que se puede definir para la CPU del contenedor o los límites de memoria
- *Cantidad mínima* que se puede definir para la CPU del contenedor o las solicitudes de memoria
- *MaxLimitRequestRatio* para evitar que el límite sea significativamente mayor que la solicitud. Si un nuevo recurso viola alguna restricción enumerada, se rechaza

```
apiVersion: "v1"
kind: "LimitRange"
metadata:
  name: "core-resource-limits"
spec:
  limits:
    - type: "Pod"
      max:
        cpu: "2"
        memory: "2Gi"
      min:
        cpu: "200m"
        memory: "6Mi"
    - type: "Container"
      max:
        cpu: "2"
        memory: "1Gi"
      min:
        cpu: "100m"
        memory: "4Mi"
      default:
        cpu: "300m"
        memory: "200Mi"
      defaultRequest:
        cpu: "200m"
        memory: "100Mi"
    maxLimitRequestRatio:
      cpu: "10"
```

Resource Management

Gestión de LimitRange

- Solo los administradores del clúster pueden crear, modificar o eliminar
 - Los usuarios pueden ver rangos límite en sus proyectos
- Establecer objetos LimitRange en cada proyecto de usuario (muy recomendable)
 - Considere la plantilla de solicitud de proyecto u operador Asegurar que los rangos límite creados, mantenidos en todos los proyectos de usuarios

```
$ oc describe limitrange core-resource-limits
Name:      core-resource-limits
Namespace: my-project
Type       Resource  Min   Max   Default Request  Default Limit  Max Limit/Request Ratio
Pod        cpu       200m  2     -              -              -
Pod        memory    6Mi   1Gi   -              -              -
Container  cpu       100m  2     200m          300m          10
Container  memory    4Mi   1Gi   100Mi         200Mi         -
```

Resource Management

Resource Quota

Resource Quota

- Se basa en la funcionalidad de request / limits
- Restringe el consumo total de recursos por proyecto o entre proyectos.
 - Cantidad total de recursos y storage consumidos en el proyecto
 - Cantidad de objetos específicos creados en proyecto
- Diferente de LimitRange
 - No restringe las especificaciones para recursos individuales
 - Ejemplo: sin LimitRange definido, el usuario puede solicitar un pod con 50Gi de memoria si está dentro de la cuota de memoria.

Tipos de cuota: recursos

- CPU y memoria
 - Suma de solicitudes de CPU y / o memoria
 - Suma de límites de CPU y / o memoria
- Storage
 - Suma de todos los request de storage y/o PVC

```
---  
apiVersion: v1  
kind: ResourceQuota  
metadata:  
  name: compute-resources  
spec:  
  hard:  
    pods: "4"  
    requests.cpu: "1"  
    requests.memory: 1Gi  
    limits.cpu: "2"  
    limits.memory: 2Gi
```

```
---  
apiVersion: v1  
kind: ResourceQuota  
metadata:  
  name: core-object-counts  
spec:  
  hard:  
    configmaps: "10"  
    persistentvolumeclaims: "4"  
    replicationcontrollers: "20"  
    secrets: "10"  
    services: "10"  
    services.loadbalancers: "2"
```

Resource Management

Cuotas multiproyecto

- Útil si es necesario definir la cuota para las cargas de trabajo distribuidas en múltiples proyectos
 - Ejemplo: la unidad de negocios asignó cierta cantidad de recursos en un clúster compartido con cargas de trabajo de aplicaciones distribuidas en muchos proyectos
- Cuotas definidas usando objetos ClusterResourceQuota

```
oc create clusterresourcequota my-cluster-resource-quota \
--project-label-selector=org=myorg \
--hard=pods=10 --hard=secrets=20
```

Resource Management

Resources type

En cada uno tenemos un total de recursos a utilizar (recursos asignado al a VM). Dentro del nodo la distribución es la siguiente:



- **System-reserved**

Reserva de recursos para todos los procesos no-pods excluyendo los kube-reserved.

- **Kube-reserved**

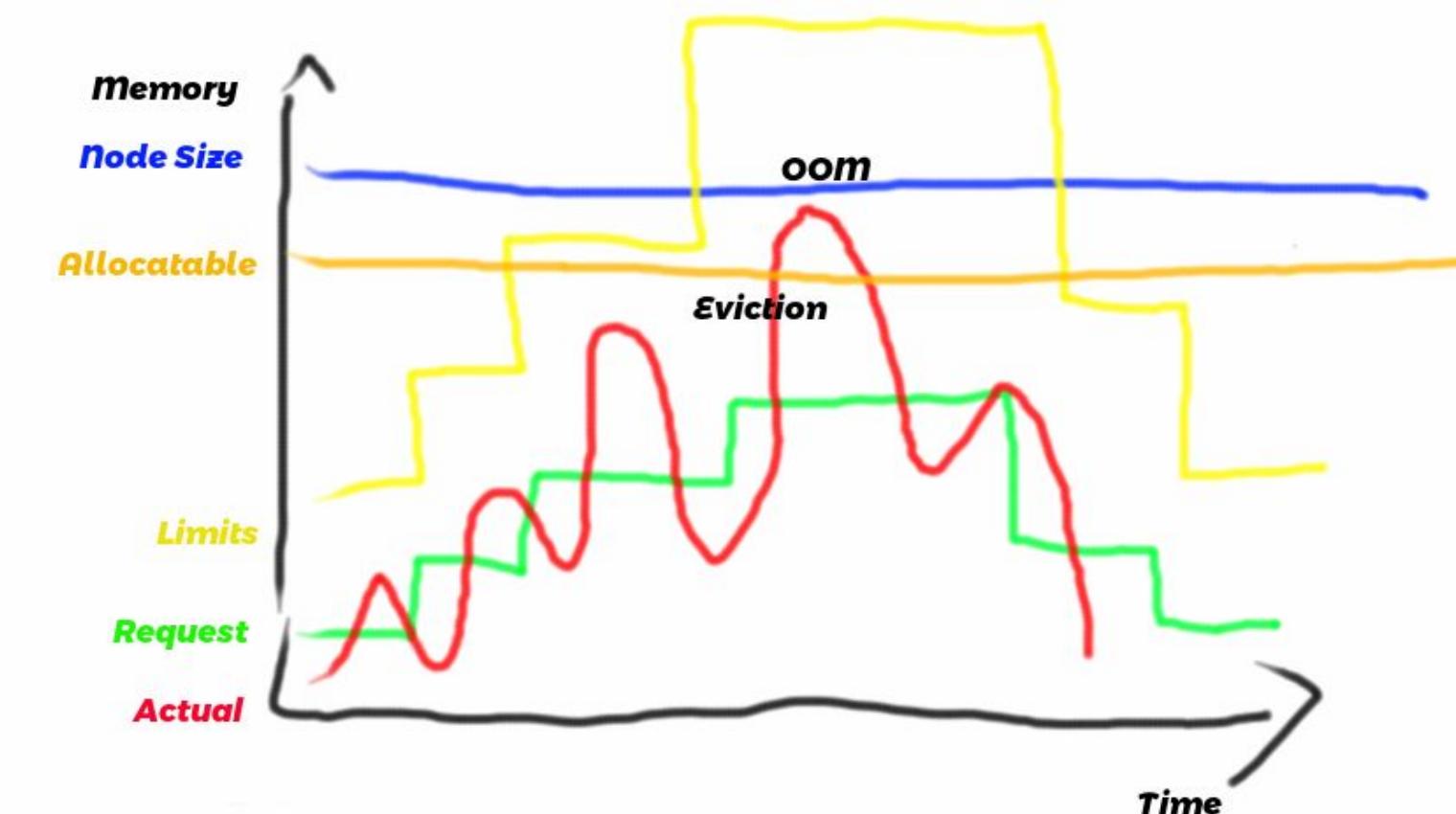
Reserva de recursos para el servicio de kubelet.

- **Hard eviction Threshold**

Límite de recurso donde actual el proceso de pod eviction.

- **Allocatable**

Conjunto de recursos restantes para asignar a los pods de aplicación



Resource Management

Protected Resource Node

Resources Type:

- **Compressible resources:**
 - CPU, block i/o and network i/o
- **Incompressible resources:**
 - Memory, disk spaces, local container disk spaces (docker storage)

Cómo protegemos el nodo de Resource Pressure?

Openshift tiene el mecanismo de Pod Evicting para proteger los recursos. Una vez alcanzado el umbral de Evicting, son elegidos por políticas de QoS.

Evicting

- **Hard Evicting**
 - No Shutdown Graceful.
- **Soft Evicting**
 - Shutdown graceful.

POD QoS	When
Guaranteed	Request = Limit
Burstable	Request < Limit
Best Effort	Request and limit are not defined.

```
> cat kubelet.conf
```

```
...
```

```
kubeletArguments:
```

eviction-hard:

- memory.available<500Mi

kube-reserved:

- "cpu=<cpu>,memory=<mem>"

system-reserved:

- "cpu=<cpu>,memory=<mem>"

Resource Management

Requests

● Requests

```
$ oc describe node worker0.ocp4.labs.semperti.local
...
Addresses:
  InternalIP: 10.252.7.235
  Hostname: worker0.ocp4.labs.semperti.local
Capacity:
  cpu:           4
  ephemeral-storage: 51876844Ki
  hugepages-2Mi: 0
  memory:        8161604Ki ←
  pods:          500
Allocatable:
  cpu:           3500m
  ephemeral-storage: 46735957528
  hugepages-2Mi: 0
  memory:        7010628Ki ←
  pods:          500
...
Non-terminated Pods: (24 in total)
  ...

```

Non-terminated Pods:	(25 in total)	CPU Requests	CPU Limits	Memory Requests	Memory Limits	AGE
Namespace	Name					
-----	-----	-----	-----	-----	-----	-----
bookinfo	productpage-v1-5f598fbff4-p15hd	10m (0%)	0 (0%)	128Mi (1%)	0 (0%)	5d
bookinfo	ratings-v1-85957d89d8-62m7c	10m (0%)	0 (0%)	128Mi (1%)	0 (0%)	5d
bookinfo	reviews-v2-67b465c497-v25zs	10m (0%)	0 (0%)	128Mi (1%)	0 (0%)	5d
bookinfo	reviews-v3-7bd659b757-spmhv	10m (0%)	0 (0%)	128Mi (1%)	0 (0%)	5d
monitoring	grafana-deployment-64f8b9cdd9-nr2j1	0 (0%)	0 (0%)	0 (0%)	0 (0%)	18d
monitoring	grafana-operator-6565c5964-vjpsm	0 (0%)	0 (0%)	0 (0%)	0 (0%)	18d
openshift-cluster-node-tuning-operator	tuned-h6z86	10m (0%)	0 (0%)	50Mi (0%)	0 (0%)	23d
openshift-cluster-storage-operator	csi-snapshot-controller-operator-8446c6c86b-c25gx	10m (0%)	0 (0%)	50Mi (0%)	0 (0%)	18d
openshift-dns	dns-default-tj7zm	110m (3%)	0 (0%)	70Mi (1%)	512Mi (7%)	23d
openshift-image-registry	node-ca-9grgx	10m (0%)	0 (0%)	10Mi (0%)	0 (0%)	23d
openshift-ingress	router-default-6464454d6d-9kgww	100m (2%)	0 (0%)	256Mi (3%)	0 (0%)	18d
openshift-machine-config-operator	machine-config-daemon-z2kfm	40m (1%)	0 (0%)	100Mi (1%)	0 (0%)	23d
openshift-monitoring	alertmanager-main-1	105m (3%)	100m (2%)	245Mi (3%)	25Mi (0%)	18d
openshift-monitoring	alertmanager-main-2	105m (3%)	100m (2%)	245Mi (3%)	25Mi (0%)	18d
openshift-monitoring	grafana-c8fd867b6-qhpj9	5m (0%)	0 (0%)	120Mi (1%)	0 (0%)	18d
openshift-monitoring	kube-state-metrics-57d9f45856-qh5nd	4m (0%)	0 (0%)	120Mi (1%)	0 (0%)	18d
openshift-monitoring	node-exporter-gdm2d	9m (0%)	0 (0%)	210Mi (3%)	0 (0%)	23d
openshift-monitoring	openshift-state-metrics-897c8d755-rwj2h	120m (3%)	0 (0%)	190Mi (2%)	0 (0%)	18d
openshift-multus	prometheus-k8s-1	76m (2%)	200m (5%)	1234Mi (18%)	50Mi (0%)	18d
openshift-operators	multus-5sh72	10m (0%)	0 (0%)	150Mi (2%)	0 (0%)	23d
openshift-sdn	istio-node-n7qpf	20m (0%)	0 (0%)	200Mi (2%)	0 (0%)	6d2h
openshift-sdn	ovs-nphv	100m (2%)	0 (0%)	400Mi (5%)	0 (0%)	23d
openshift-user-workload-monitoring	sdn-rtng6	100m (2%)	0 (0%)	200Mi (2%)	0 (0%)	23d
workshop	prometheus-user-workload-0	360m (10%)	200m (5%)	1194Mi (17%)	50Mi (0%)	18d
	blog-1-kkv8t	100m (2%)	0 (0%)	4Mi (0%)	0 (0%)	8m44s
Allocated resources:						
(Total limits may be over 100 percent, i.e., overcommitted.)						
Resource Requests Limits						
cpu	1434m (40%)	600m (17%)				
memory	5560Mi (81%)	602Mi (9%)				
ephemeral-storage	0 (0%)	0 (0%)				

Este valor representa la memoria que ha sido reservada, no la memoria que está siendo utilizada.

Si no se especifica qué cantidad de recursos necesita el contenedor, OpenShift asume cero (0). Esto implica que si no especifica request de recursos, puede quedarse sin recursos mientras OpenShift seguirá pensando que todos sus nodos tienen capacidad disponible.

Best Practice: especificar request para cada uno de los recursos!

Resource Management

Node Overcommitment

Node Overcommit es cuando la **suma de los Resources Limits** es mayor que los recurso que pueden alocarse. Cuando todos los pods de un nodo solicitan su límite de recursos llegamos a un “Resource Pressure”.

- **Cómo lo detectamos?**

- `oc describe node <node name>`
- Dashboard Grafana Kubernetes/Compute Resources/Cluster

- **Cuanto overcommit debemos permitir?**

- Identificada la carga de trabajo, aumentar periodicamente.
- Límite tolerable de pods en estado Evicted.

- **Como podemos forzar el overcommit?.**

- Por proyecto utilizar LimitRanges.

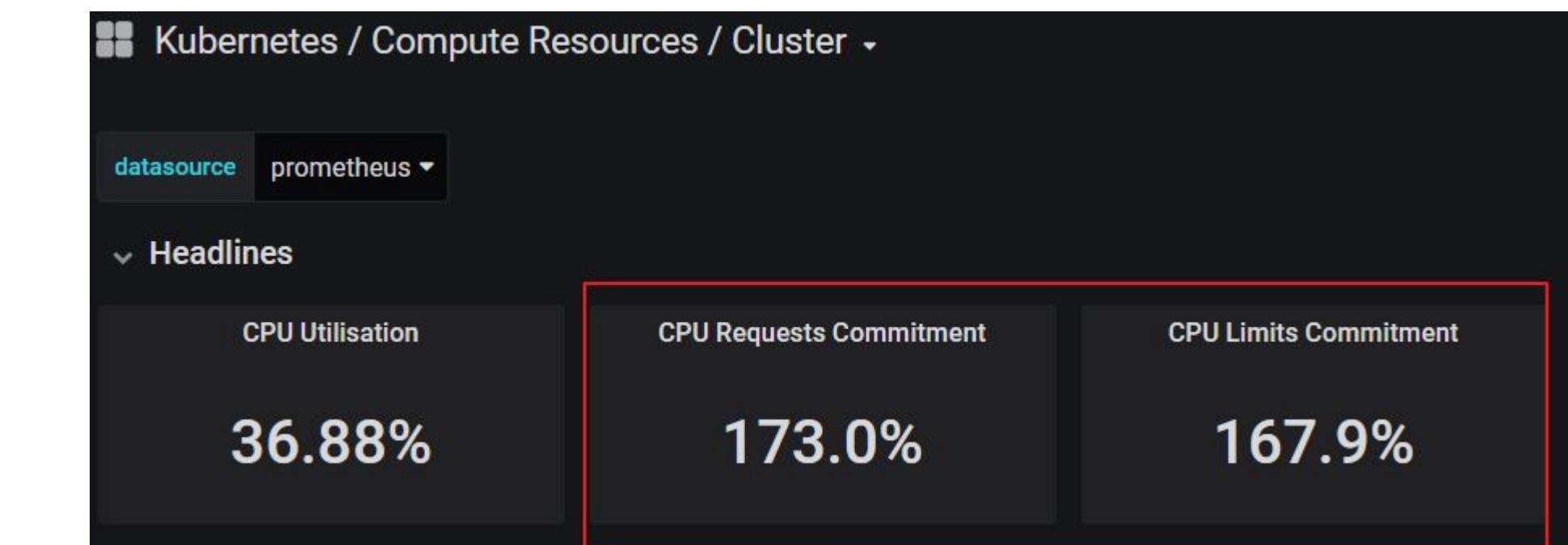
Recomendaciones

1. Definir **overcommitment policy**
2. Admin/Desarrollador tiene que tener la capacidad de poder chequear la disponibilidad del cluster (**Grafana Dashboard**).
3. **Crear alertas** que indiquen el nivel de overcommit tolerado.

```
$ oc describe node worker2.ocp4.labs.semperti.local | tail -6
```

Resource	Requests	Limits
cpu	3670m (104%)	4300m (122%)
memory	6808Mi (45%)	4376Mi (29%)
ephemeral-storage	0 (0%)	0 (0%)

Events: <none>



Resource Management

Kubelet Configuration

Como editar los parametros de kubelet?

- **Via machineconfig:**

- oc edit machineconfig <01-worker-kubelet> # (01-master-kubelet). Modificar la entrada del archivo /etc/kubernetes/kubelet.conf.

- **Via KubeletConfig Operator**

- Crear un custom resource con la definición que se desea impactar.

```
$ oc get kubeletconfig set-max-pods -o yaml | grep -A6 "spec"
```

```
spec:
```

```
  kubeletConfig:
```

```
    maxPods: 500
```

```
  machineConfigPoolSelector:
```

```
    matchLabels:
```

```
      custom-kubelet: large-pods
```

```
  Status:
```

```
$ oc label node worker2.ocp4.labs.semperti.local custom-kubelet=large-pods
```

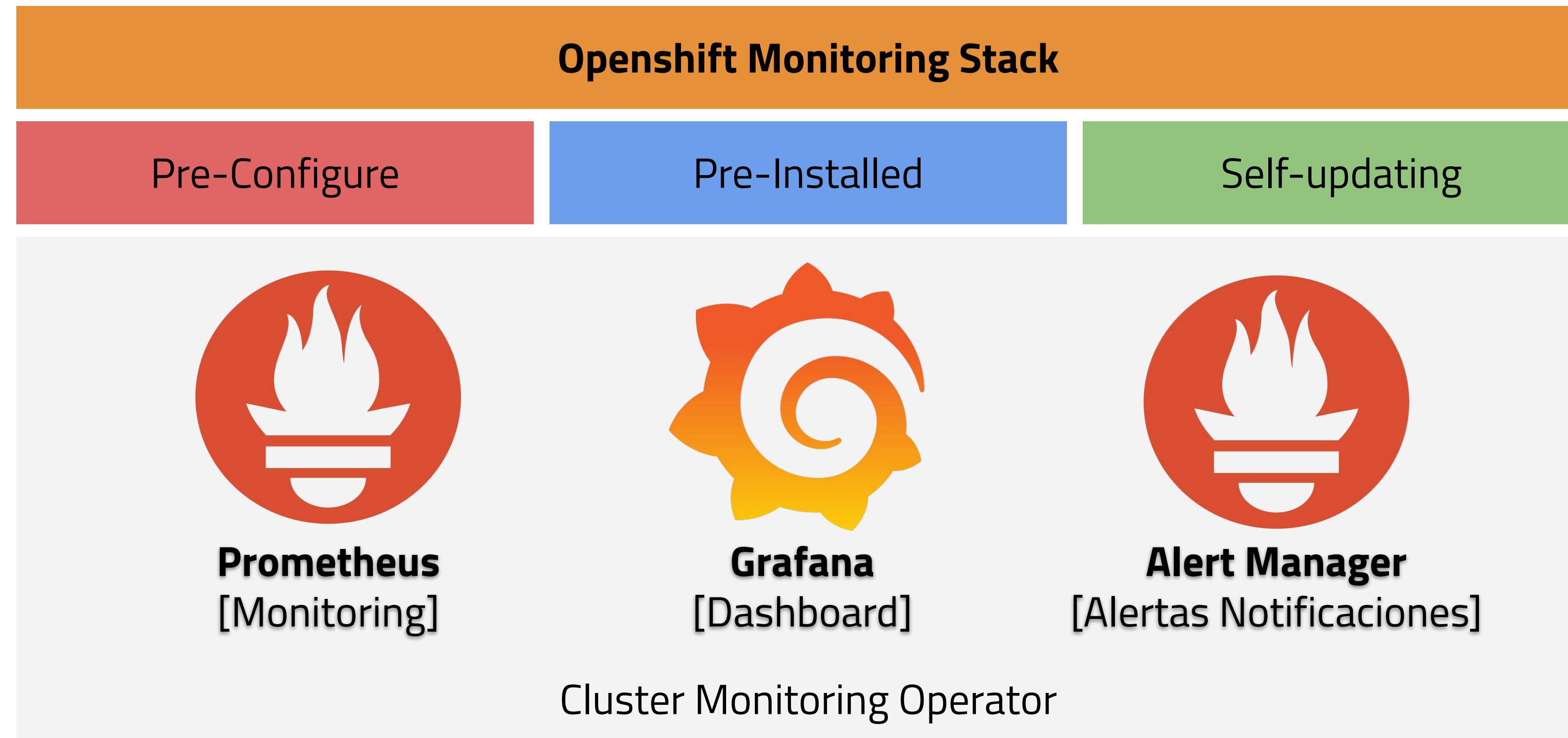
Todas las configuración que se hagan por cualquiera de los dos metodos impactan en un archivo de configuracion en cada uno de los nodos de Openshift en el path /etc/kubernetes/kubelet.conf

Monitoreo

Semperti

Monitoring

Cluster Monitoring Operator

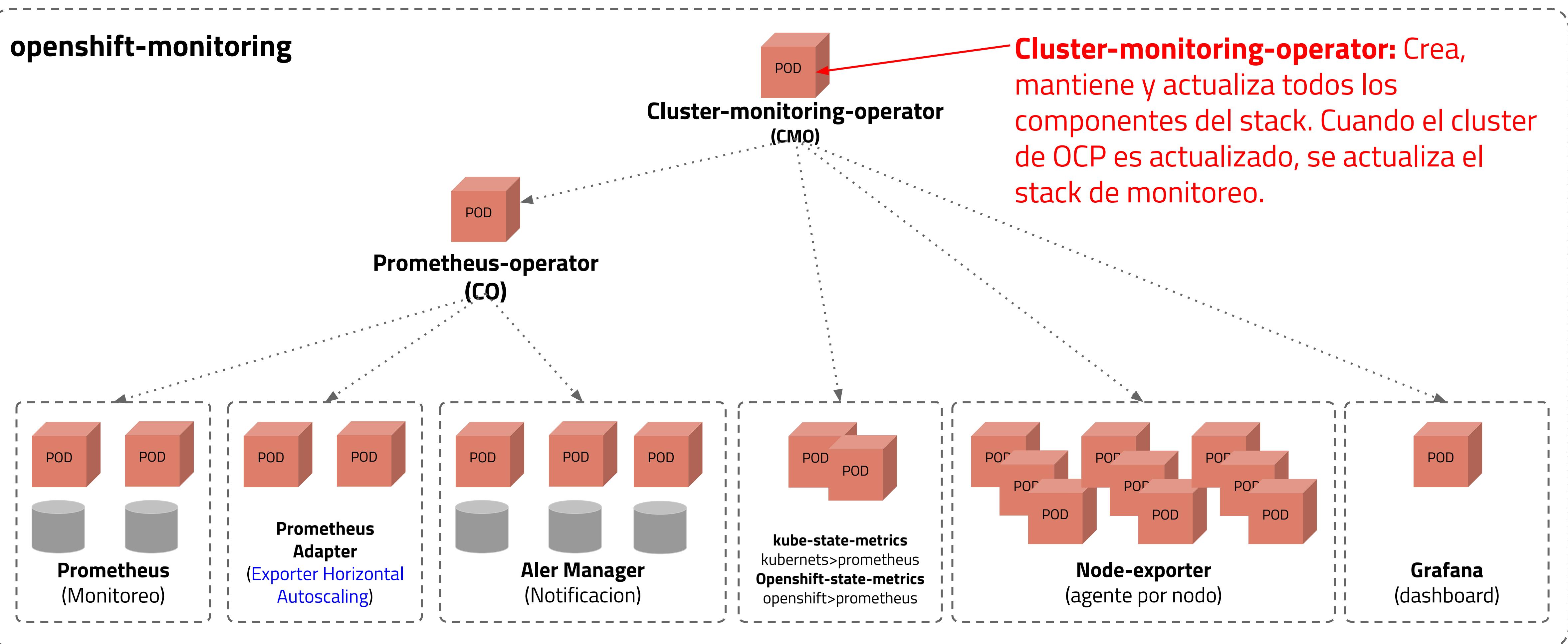


ALERT! Para asegurar la compatibilidad con futuros Openshift updates, solamente está soportada la configuración del stack que se detalla en la documentación

https://access.redhat.com/documentation/en-us/openshift_container_platform/4.3/html-single/monitoring/index

Monitoring

Cluster Monitoring Operator



Monitoring

Cluster Monitoring Operator

Endpoints de monitoreo:

- CoreDNS
- Elasticsearch (if Logging is installed)
- etcd
- Fluentd (if Logging is installed)
- HAProxy
- Image registry
- Operator Lifecycle Manager (OLM)
- Telemeter client
- Kubelets
- Kubernetes apiserver
- Kubernetes controller manager
- Kubernetes scheduler
- Metering (if Metering is installed)
- OpenShift apiserver
- OpenShift controller manager

IMPORTANTE: las configuración de monitoreo NO se debe cambiar, en caso de falla se debe reportar a Red Hat y **en caso de querer monitorear componentes aplicativos se debe buscar otra solución de monitoreo adicional, pudiendo ser el mismo stack u otro producto.**

Monitoring

Cluster Monitoring Operator

Casos explicitamente **NO** soportados

- Crear objetos **ServiceMonitor** adicionales en los **openshift-* namespaces**.
- Crear objetos **ConfigMaps** o **PrometheusRules** inesperados.
- Modificar recursos del stack.
- Usar recursos del stack para otro propósito.
- Agregar nuevas reglas de alertas.
- Modificar Grafana.

Monitoring

Configuración de componentes

Configuración, todo el stack desde **ConfigMaps cluster-monitoring-config**
\$ oc edit configmap **cluster-monitoring-config** -n openshift-monitoring

Sintaxis **cluster-monitoring-config**

```
apiVersion: v1
kind: ConfigMap
data:
  config.yaml: |
    prometheusK8s:
      Retention: 24hs
      nodeSelector:
        node-role.kubernetes.io/infra: ""
      volumeClaimTemplate:
        metadata:
          name: my-prometheus-claim
        spec:
          storageClassName: thin
      resources:
        requests:
          storage: 100Gi
```

Componente	Key
Prometheus Operator	prometheusOperator
Prometheus	prometheusK8s
Alertmanager	alertmanagerMain
kube-state-metrics	kubeStateMetrics
Grafana	grafana
Telemeter Client	telemeterClient
Prometheus Adapter	k8sPrometheusAdapter

Monitoring

Alert Manager, notificaciones.

Configuración, debemos modificar el **secret** llamado **alertmanager-main** que tiene enbebido en base64 el archivo **alertmanager.yaml**

```
$ oc -n openshift-monitoring get secret alertmanager-main --template='{{ index .data "alertmanager.yaml" }}' |base64 -d > alertmanager.yaml
```

```
$ cat alertmanager.yaml
global:
  resolve_timeout: 5m
route:
  group_wait: 30s
  group_interval: 5m
  repeat_interval: 12h
  receiver: default
  routes:
    - match:
        alertname: Watchdog
        repeat_interval: 5m
        receiver: watchdog
```

```
...
  - match:
      service: example-app
      routes:
        - match:
            severity: critical
            receiver: team-frontend-page
  receivers:
    - name: default
    - name: watchdog
    - name: team-frontend-page
    pagerduty_configs:
      - service_key: "your-key"
```

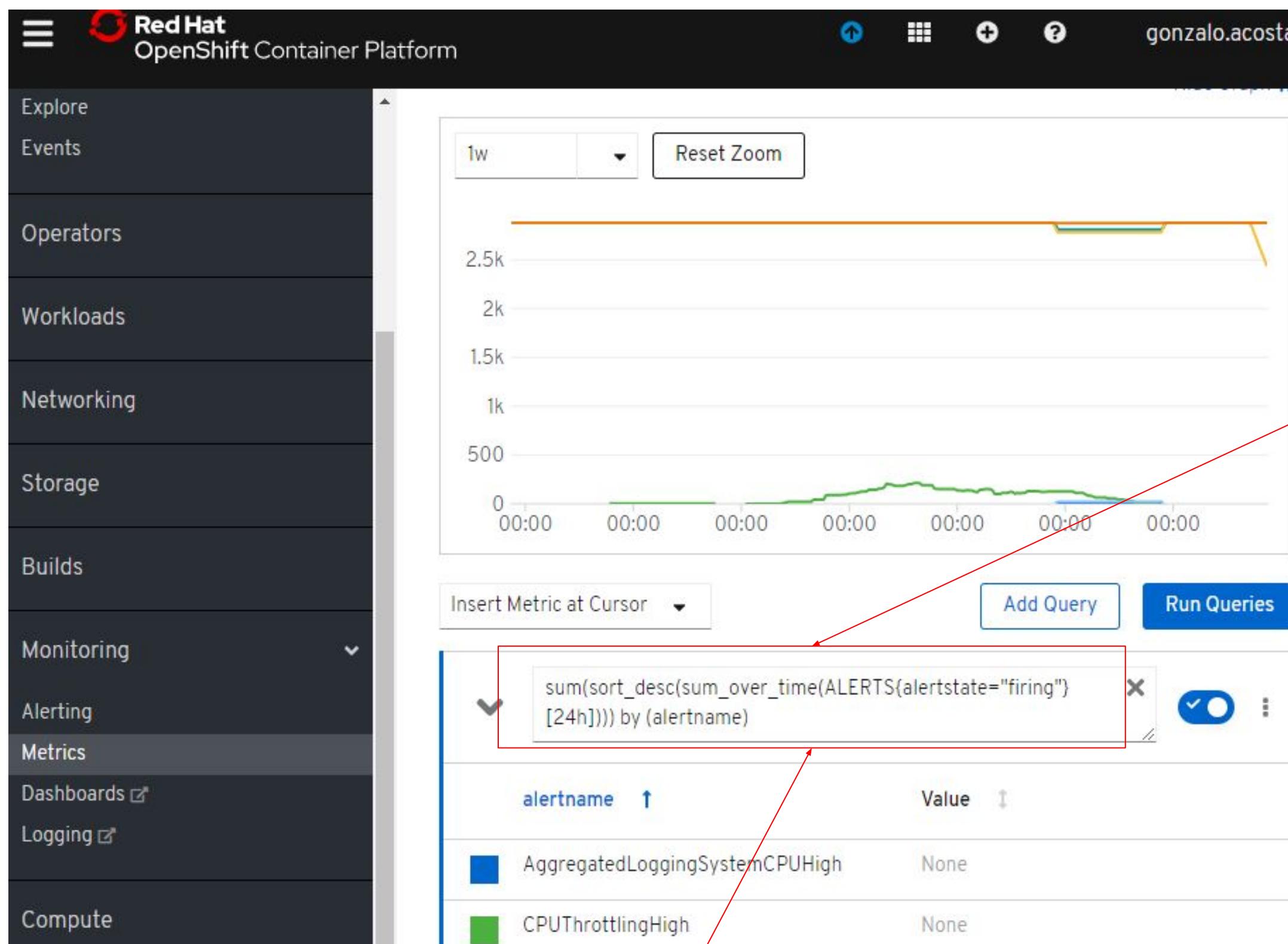
Tipos de receivers:

- <inhibit_rule>
- <http_config>
- <tls_config>
- <receiver>
- <email_config>
- <hipchat_config>
- <pagerduty_config>
- <slack_config>
- <webhook_config>
- <wechat_config>

Monitoring

Métricas

Vista desde la UI



Por medio de lenguaje
PromQL podemos consultar
la base de datos de métricas,
sacando reportes y
graficando en la UI

`sum(sort_desc(sum_over_time(ALERTS{alertstate="firing"}[24h]))) by (alertname)`

<https://prometheus.io/docs/prometheus/latest/querying/basics/>

Monitoring

Métricas - PromQL

CPU Usage

```
topk(25, sort_desc(sum(avg_over_time(pod:container_cpu_usage:sum{container="",pod!="",namespace='ns1'}[5m])) BY (pod, namespace)))
```

Memory Usage

```
topk(25, sort_desc(sum(avg_over_time(container_memory_working_set_bytes{container="",pod!="",namespace='ns1'}[5m])) BY (pod, namespace)))
```

Filesystem Usage

```
topk(25, sort_desc(sum(pod:container_fs_usage_bytes:sum{container="",pod!="",namespace='ns1'}) BY (pod, namespace)))
```

Receive Bandwidth

```
topk(25, sort_desc(sum(rate(container_network_receive_bytes_total{ container="POD", pod!="", namespace='ns1'})[5m])) BY (namespace, pod))
```

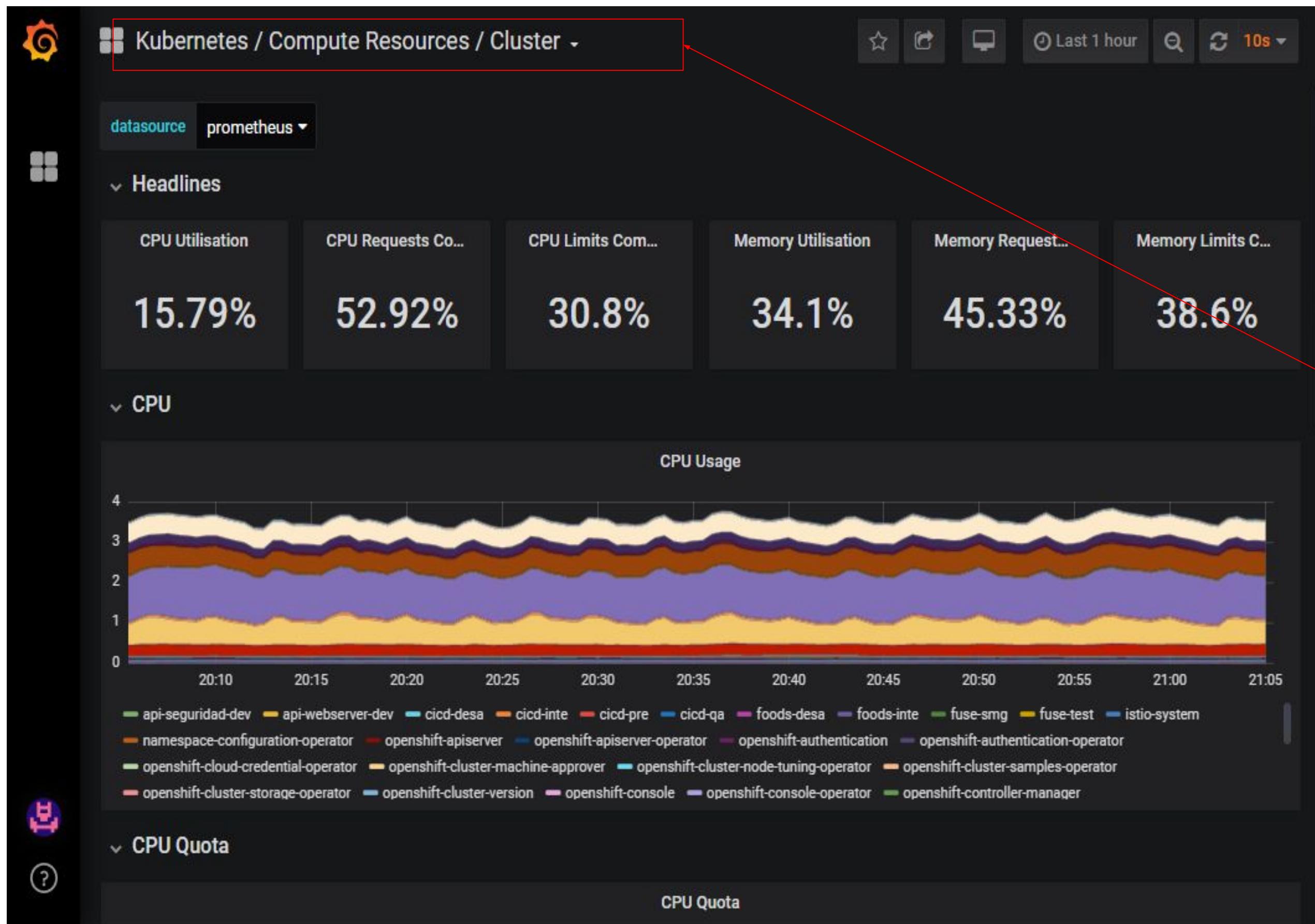
Transmit Bandwidth

```
topk(25, sort_desc(sum(rate(container_network_transmit_bytes_total{ container="POD", pod!="", namespace='ns1'})[5m])) BY (namespace, pod))
```

Monitoring

Grafana

Vista desde la UI



Grafana ofrece una multiplicidad de dashboard pre armados con métricas listas para ser utilizadas, toma la información de la base de datos de métricas de prometheus.

Monitoring

Monitoreo de servicios propios

Dos opciones:

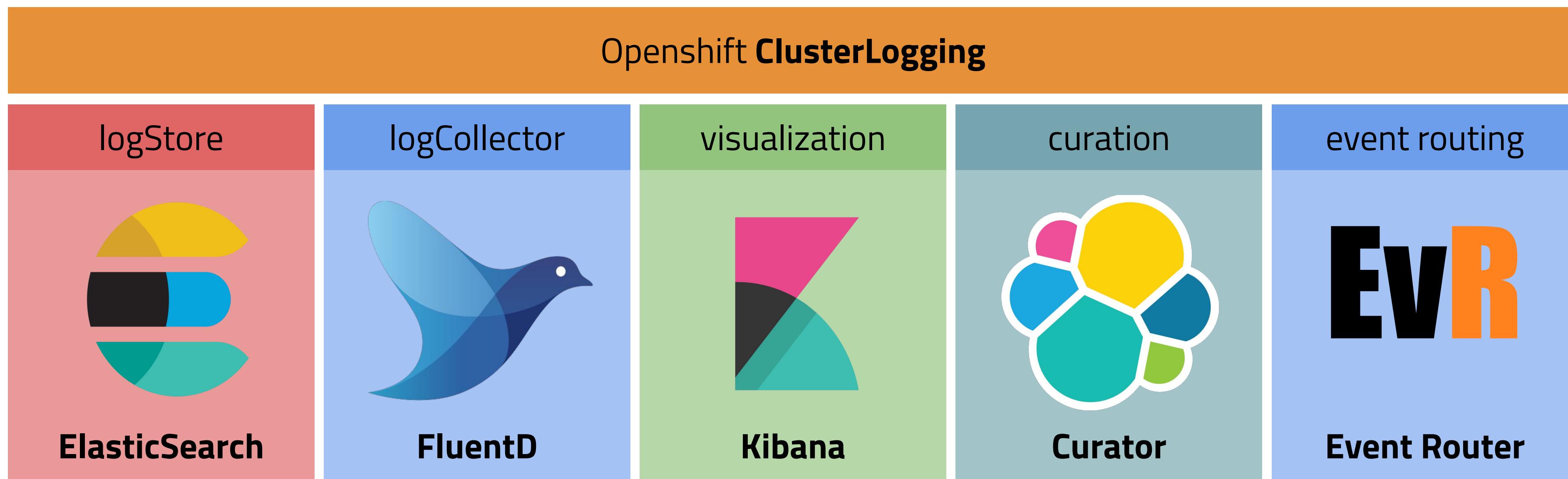
1. Utilizando **Openshift Cluster Monitoring Operator** (**ALERT!** En Tech Preview)
 - a. Procedimiento
 - i. **Editar ConfigMap** cluster-monitoring-config y agregar techPreviewUserWorkload: true al archivo config.yaml.
 - ii. **Crear Role** que permita recolectar métricas.
 - iii. **Asignar Role** (rolebinding) a un usuario.
 - iv. **Crear ServiceMonitor** en el proyecto aplicativo y definir el endpoint.
 - v. **Crear** la regla de alerta (**PrometheusRule**) en el proyecto donde vive la aplicación.
 - vi. **Asignar permisos** de vista a un usuario **developer** para que pueda ver las métricas de su servicio.
 - vii. **Consultar metricas** desde WebUI.
2. Utilizar **OperatorHub** para instalar una **herramienta de terceros** o la integración con una existente.
 - a. OperatorHub>Monitoring

Logging

Semperti

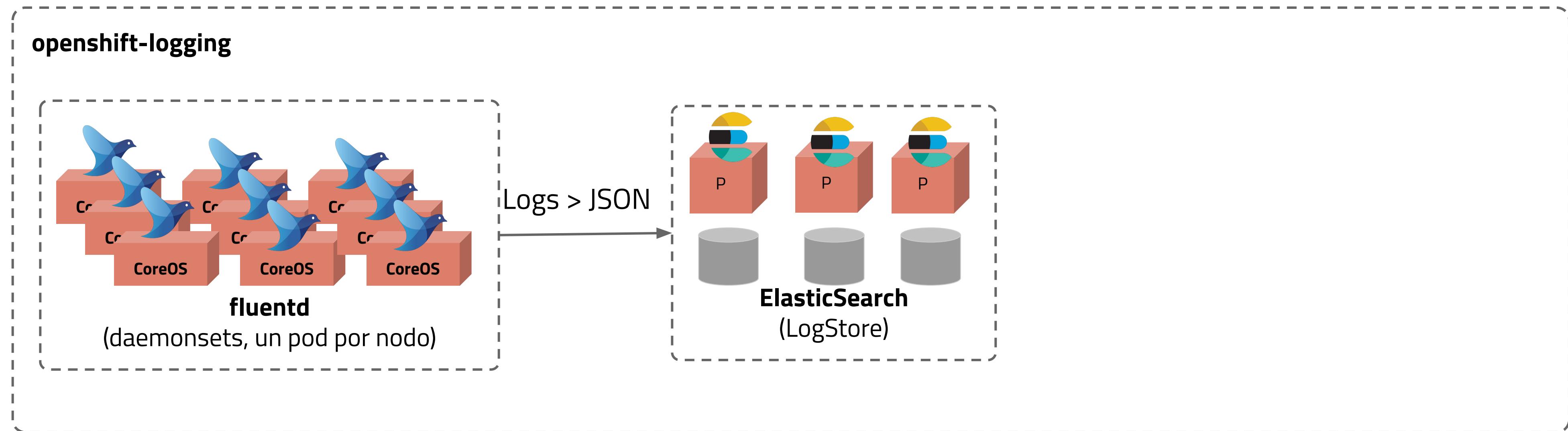
Logging

ClusterLogging



Logging

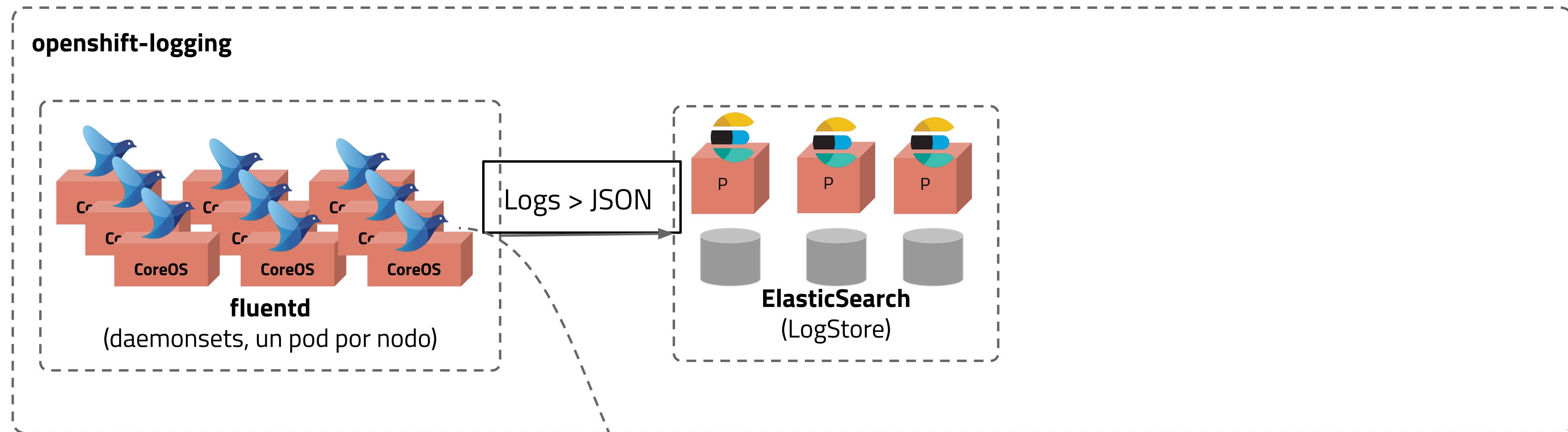
ClusterLogging flujo de trabajo



FluentD colecta los logs de cada uno de los nodos. **Journald** es la fuente de logs del sistema y provee información de **log message**, del **runtime de container** y **OCP**. FluentD transforma los logs en formato texto a documentos JSON para ser insertado en el logStore ES.

Logging

ClusterLogging flujo de trabajo



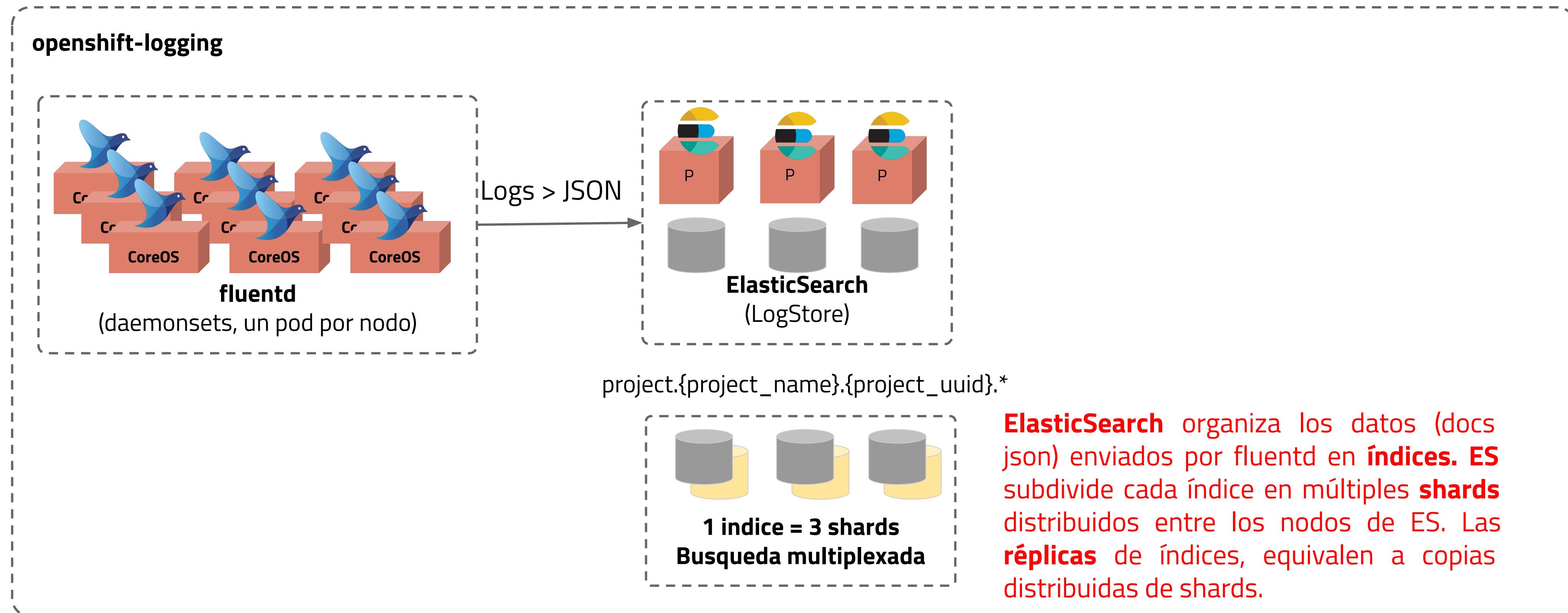
FluentD colecta los logs de cada uno de los nodos. **Journald** es la fuente de logs del sistema y provee información de **log message**, del **runtime de container** y **OCP**. FluentD transforma los logs en formato texto a documentos JSON para ser insertado en el logStore ES.



FluentD es quien en su configuración puede decidir dónde redirigir los logs, si al ES interno o si es a otro ES, Splunk.

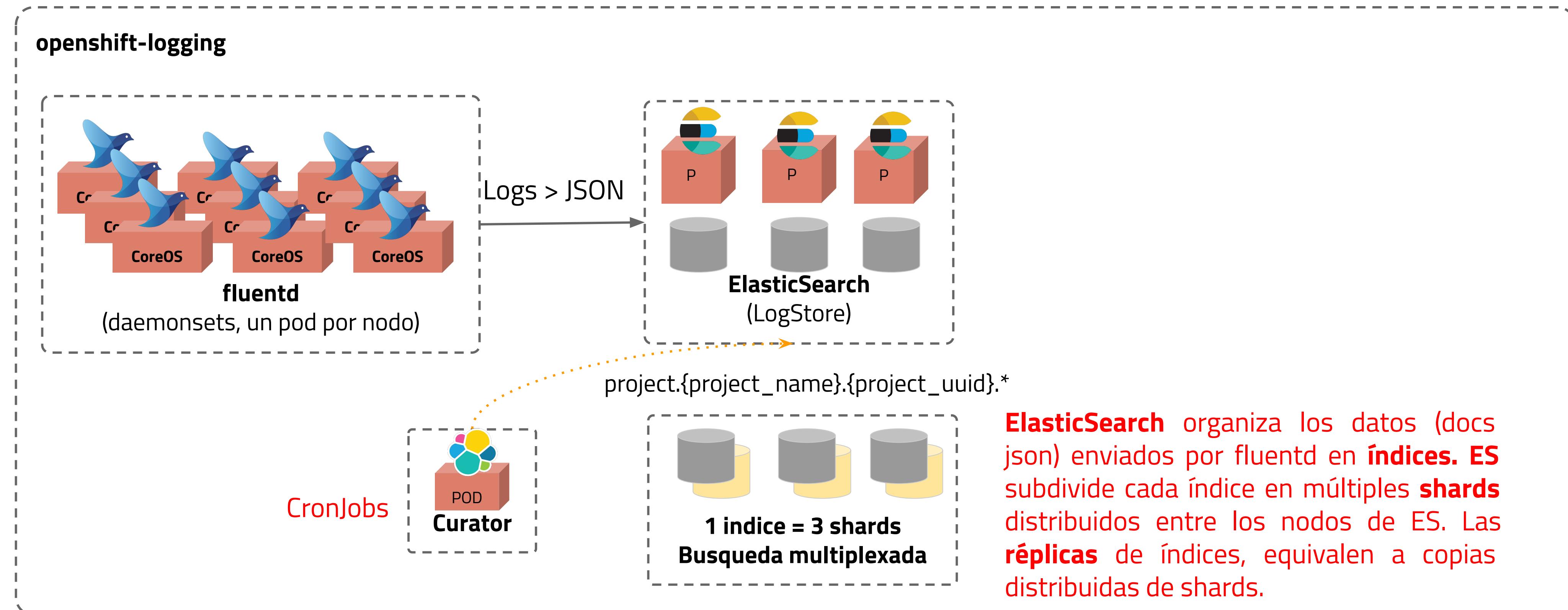
Logging

ClusterLogging flujo de trabajo



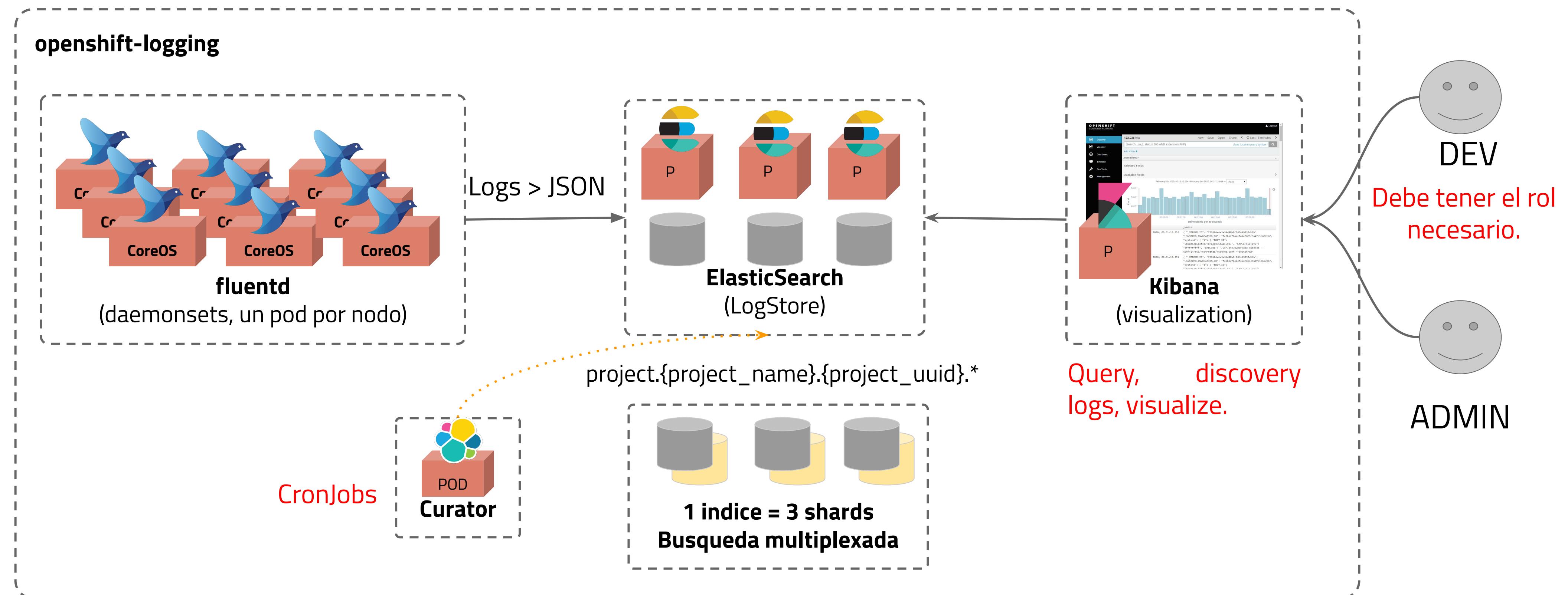
Logging

ClusterLogging flujo de trabajo



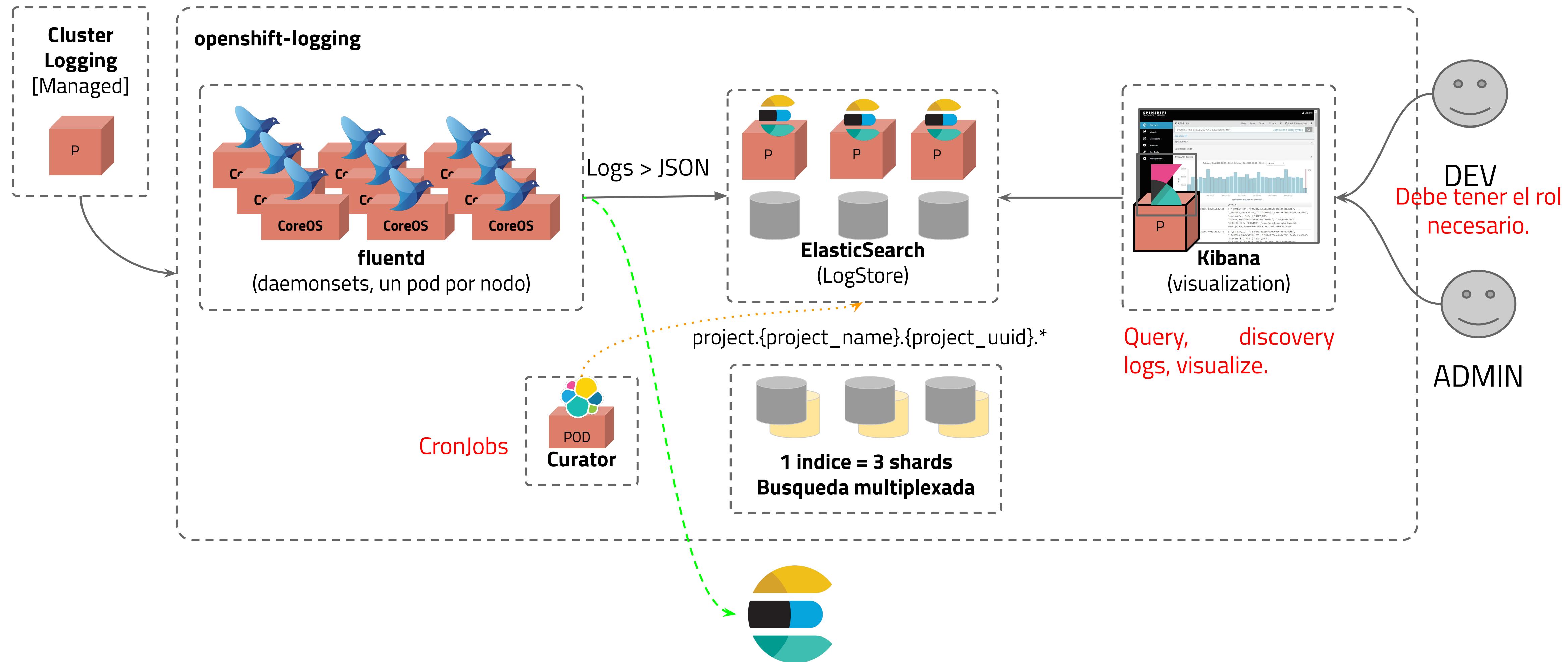
Logging

ClusterLogging flujo de trabajo



Logging

ClusterLogging flujo de trabajo



Logging

Links

Cluster Logging

<https://docs.openshift.com/container-platform/4.3/logging/cluster-logging.html>

Cambiar el modo de Operación

<https://docs.openshift.com/container-platform/4.3/logging/config/cluster-logging-management.html>

ElasticSearch

<https://docs.openshift.com/container-platform/4.3/logging/config/cluster-logging-elasticsearch.html>

Curator

<https://docs.openshift.com/container-platform/4.3/logging/config/cluster-logging-curator.html>

Enviar logs a un dispositivo externo

<https://docs.openshift.com/container-platform/4.3/logging/config/cluster-logging-external.html>

Backup

Semperti

Backup & Restore

Control Plane

Pre requisito

SSH sobre un nodo master como root, utilizo key de ssh.

```
$ ssh -i id_rsa core@<masternode>
```

Backup Etcd - Control Plane.

1. Ejecutar script de backup.

```
$ sudo /usr/local/bin/etcd-snapshot-backup.sh ./assets/backup
```

2. El formato del archivo de backup.

```
./assets/backup/snapshot_db_kuberources_<datetimestamp>.tar.gz
```

Restore Etcd - Control Plane.

1. Copiar el archivo de backup en todos los nodos en /home/core
2. Setear la variable INITIAL_CLUSTER con la lista de miembros <name>=<url> en todos los nodos.
3. Correr el script de restore.

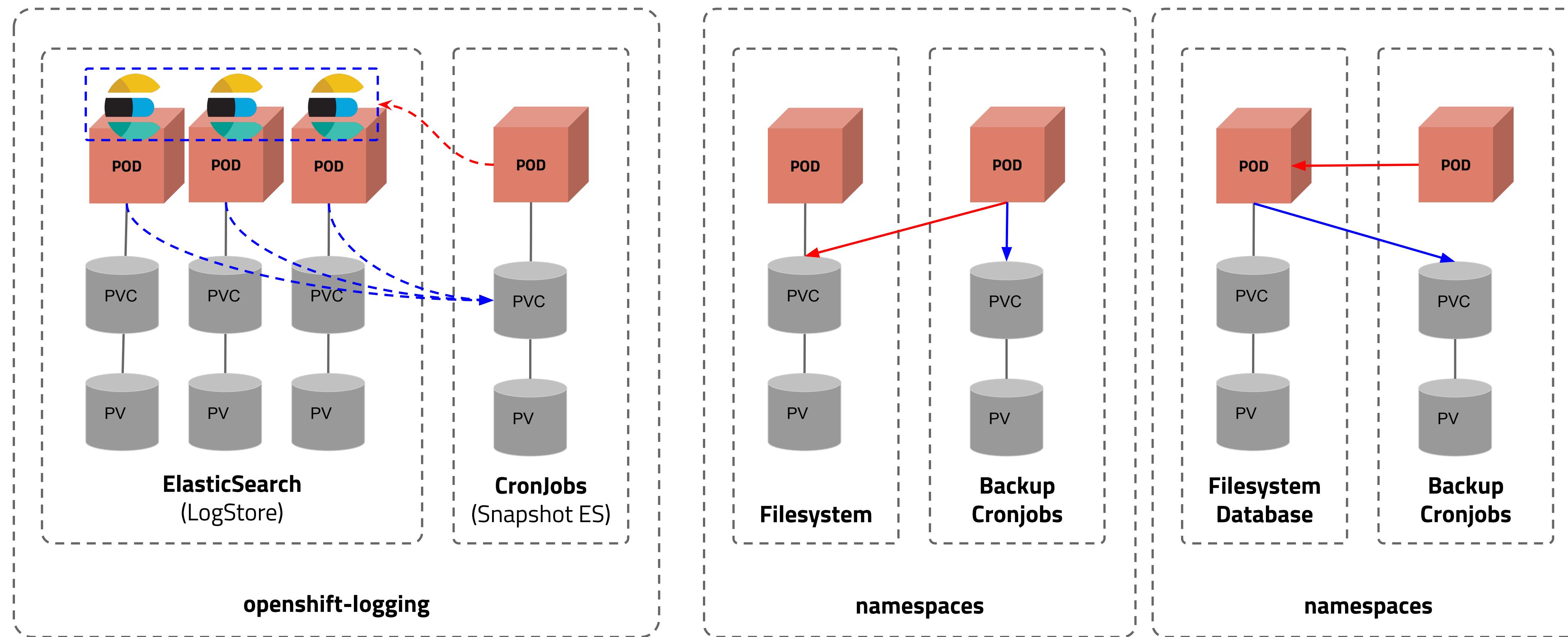
```
$ sudo /usr/local/bin/etcd-snapshot-restore.sh  
/home/core/snapshot_db_kuberources_<datetimestamp>.tar.gz $INITIAL_CLUSTER
```

https://docs.openshift.com/container-platform/4.3/backup_and_restore/backing-up-etcd.html

https://docs.openshift.com/container-platform/4.3/backup_and_restore/disaster_recovery/scenario-2-restoring-cluster-state.html#dr-restoring-cluster-state

Backup & Restore

Persistencia de datos aplicativos.



Backup & Restore

Persistencia de datos aplicativos.

Problemas y consideraciones

- **Poca frecuencia** en el ejercicio de backup y restore ante un problema de DR.
- **Kubernetes no** posee una capa de abstracción nativa para **realizar backups y restores**. Extensiones propietarias deben ser utilizadas como [Velero](#) y [Fossul](#) o considerar abordar la solución directamente desde el storage.
- El restore de datastores de gran volumen suele ser un tiempo mucho más amplio de lo que el negocio puede soportar.



Management Networking

Semperti

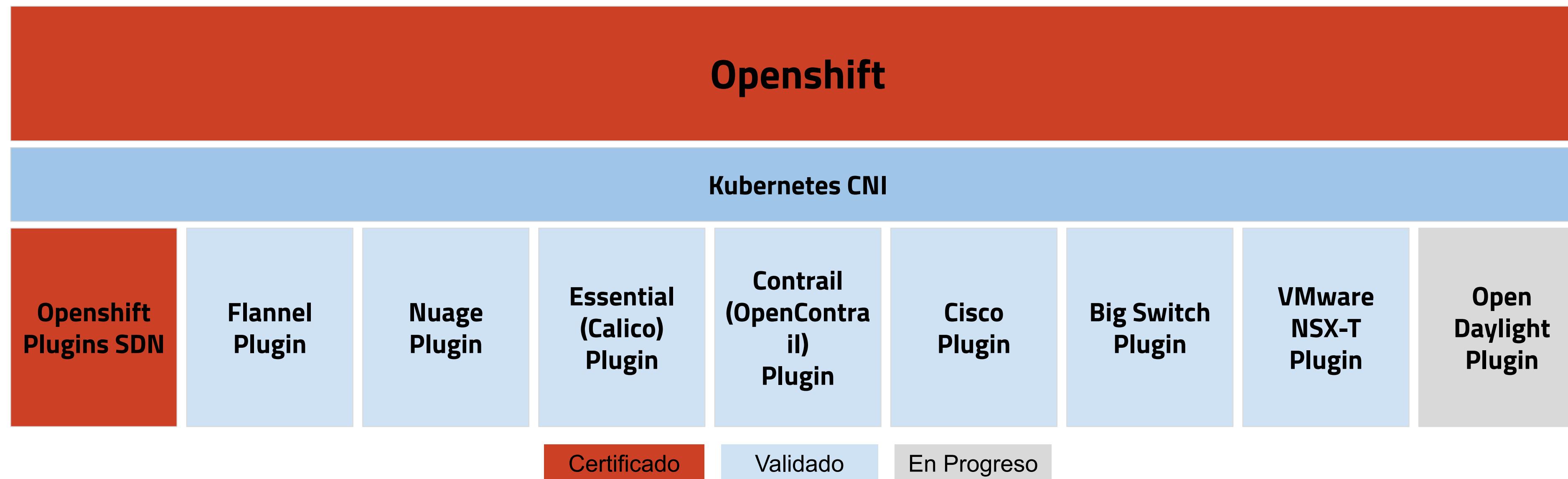
Management Networking

Visión General

- OpenShift Networking Overview
- Networking Operators
- OpenShift SDN
- Egress
- Kube-proxy
- Routes

Openshift SDN

Container Network Interfaces



Link RH: <https://docs.openshift.com/container-platform/4.2/networking/understanding-networking.html>

Management Networking

Networking Operators

Openshift Networking Goals

- Asegurar que los pods puedan comunicarse unos con otros.
- Asignar a cada pod una IP address del pool interno.
- Pods puede tratar con host externos por medio de:
 - Port allocation and networking
 - Naming and service discovery.
- Load Balancing

Management Networking

OpenShift Networking Overview

Tres network operators principales en OpenShift 4.

Cluster Network Operator (CNO)

DNS Operator

Ingress Operator

Cada uno de estos operadores cumple una función específica en la OpenShift SDN

Management Networking

Cluster Network Operator

Cluster Network Operator

- Despliega y administra los componentes del cluster network.
- Implementa una network API desde operator.openshift.io API group.
- Responsable del despliegue del plugin Openshift SDN usando DaemonSets
- Puede desplegar diferentes plugins de SDN

Deployment

```
$ oc get -n openshift-network-operator deployment/network-operator
```

Cluster Network Operator Status

```
$ oc get clusteroperator/network
```

```
$ oc get network.operator/cluster -o yaml
apiVersion: operator.openshift.io/v1
kind: Network
metadata:
  creationTimestamp: "2019-06-02T17:56:15Z"
  generation: 1
  name: cluster
  resourceVersion: "1384"
  selfLink:
    /apis/operator.openshift.io/v1/networks/cluster
  uid: b705f4f0-855f-11e9-a105-02cf5ed2d57c
spec:
  clusterNetwork:
    - cidr: 10.128.0.0/14
      hostPrefix: 23
  defaultNetwork:
    type: OpenShiftSDN
    openshiftSDNConfig:
      mode: NetworkPolicy
      vxlanPort: 4789
      mtu: 1450
      useExternalOpenvswitch: false
  serviceNetwork:
    - 172.30.0.0/16
status: {}
```

Management Networking

DNS Operator

DNS Operator

- Despliega y administra el servicio de DNS del Cluter CoreDNS
- Implementa dns API desde operator.openshift.io API group\
- Despliega CoreDNS usando DaemonSets
- Crea Service para DaemonSet
- Configura Kubelet para que les sea indicado a los pods usar CoreDNS service IP.

DNS Operator Deployment:

```
$ oc get -n openshift-dns-operator deployment/dns-operator
```

DNS Operator status:

```
$ oc get clusteroperator/dns
```

```
$ oc get dns.operator/default -o yaml
apiVersion: operator.openshift.io/v1
kind: DNS
metadata:
  creationTimestamp: "2019-06-02T17:58:01Z"
  finalizers:
    - dns.operator.openshift.io/dns-controller
  generation: 1
  name: default
  resourceVersion: "941800"
  selfLink:
    /apis/operator.openshift.io/v1/dnses/default
  uid: f6953617-855f-11e9-a105-02cf5ed2d57c
spec: {}
status:
  clusterDomain: cluster.local
  clusterIP: 172.30.0.10
  conditions:
    - lastTransitionTime: "2019-06-02T17:58:25Z"
      message: Minimum number of Nodes running
      DaemonSet pod
      reason: AsExpected
      status: "True"
      type: Available
```

Management Networking

Ingress Operator

Ingress Operator

- Despliega y administra uno o mas Ingress Controller basados en HAProxy.
- Implementa el ingresscontroller API desde config.openshift.io API group
- Ingress Controller adicionales puede ser creados. Los nuevos seran nuevos pods de HAProxy + load balancer (IPI) o HAProxy (UPI) + balanceo por parte del cliente.

Ingress operator Deployment:

```
$ oc get -n openshift-ingress-operator deployment/ingress-operator
```

Ingress operator status:

```
$ oc get clusteroperator/ingress
```

Podemos aumentar el número de replicas con

```
oc patch -n openshift-ingress-operator ingresscontroller/default --patch  
'{"spec":{"replicas": 3}}' --type=merge
```

```
Name: default
Namespace: openshift-ingress-operator
Labels: <none>
Annotations: <none>
API Version: operator.openshift.io/v1
Kind: IngressController
...
Spec:
  Replicas: 2
...
Domain:
apps.cluster-a43f.sandbox1895.opentlc.com
Endpoint Publishing Strategy:
  Load Balancer:
    Scope: External
    Type: LoadBalancerService
  Observed Generation: 1
  Selector: Selector:
    ingresscontroller.operator.openshift.io/deployment-ingre
sscontroller=default
  Tls Profile:
    Ciphers:
      TLS_AES_128_GCM_SHA256
...
  Min TLS Version: VersionTLS12
Events: <none>
```

Management Networking

Nodes

Configuración en los nodos

- Openshift almacena la configuración de los nodos y subnets en la base de datos etcd.
- Cuando un nodo es agregado al cluster:
 - Subnet es alocada desde la cluster network.
 - Reglas de Openflow son agregadas para direccionar paquetes de red a los nodos remotos
 - Cuando un nodo es eliminado del cluster las subredes sub neteadas de la subred del cluster quedan disponibles.

Device Node

- br0 - bridge donde serán conectados los nodos.
- tun0 - por usado para el tráfico de egress fuera del cluster
- vxlan0 - port usado para el egress de tráfico hacia otro nodo del cluster
- vethX - port en OVS para conectar las veth de pods

```
$ oc rsh ovs-chmq6 -n openshift-sdn
sh-4.2# ovs-vsctl show
896d4dbb-82ca-4892-8f1c-81a7ff326238
    Bridge "br0"
        fail_mode: secure
        Port "br0"
            Interface "br0"
                type: internal
        Port "veth5511d1e6"
            Interface "veth5511d1e6"
        Port "tun0"
            Interface "tun0"
                type: internal
        Port "veth759d51f4"
            Interface "veth759d51f4"
        Port "vxlan0"
            Interface "vxlan0"
                type: vxlan
                options: {dst_port="4789", key=flow,
remote_ip=flow}
                ovs_version: "2.11.0"
```

Management Networking

Pods

- Nuevos pods obtienen su ip address desde la subred del host.
- Pods son conectados usando una veth interface
- Un lado del veth es conectado al pod
- Un lado del veth es conectado al br0 en el OVS
- Reglas OpenFlow son dinamicamente agregadas al OVS DB

```
$ oc rsh ovs-chmq6 -n openshift-sdn
sh-4.2# ovs-ofctl dump-flows br0 -O OpenFlow13
cookie=0x0, duration=244856.899s, table=0, n_packets=295460, n_bytes=24738451, priority=400,ip,in_port=tun0,nw_src=10.131.0.1
actions=goto_table:30
cookie=0x0, duration=244856.899s, table=0, n_packets=13927660, n_bytes=9560034620, priority=300,ct_state=-trk,ip actions=ct(table=0)
cookie=0x0, duration=244856.899s, table=0, n_packets=313961, n_bytes=46400504,
priority=300,ip,in_port=tun0,nw_src=10.131.0.0/23,nw_dst=10.128.0.0/14 actions=goto_table:25
```

Management Networking

Kubernetes Service

Kubernetes Service

- Kubernetes Network Proxy (kube-proxy)
- Kube-proxy corre en cada nodo.
- Implementando sobre reglas de iptables en OpenShift
- Network rules reenvie las conexiones a los endpoints asociados a los servicios.

```
$ oc debug node/worker1 --image=rhel7/rhel-tools
sh-4.4# iptables-save | grep a-network-test
-A KUBE-SERVICES -d 172.30.78.212/32 -p tcp -m comment --comment "a-network-test/example: cluster IP" -m tcp --dport 8080 -j
KUBE-SVC-NRQ7NG2MM3ZIPEW
sh-4.4# iptables-save | grep KUBE-SVC-NRQ7NG2MM3ZIPEW
:KUBE-SVC-NRQ7NG2MM3ZIPEW - [0:0]
-A KUBE-SERVICES -d 172.30.78.212/32 -p tcp -m comment --comment "a-network-test/example: cluster IP" -m tcp --dport 8080 -j
KUBE-SVC-NRQ7NG2MM3ZIPEW
-A KUBE-SVC-NRQ7NG2MM3ZIPEW -m statistic --mode random --probability 0.33332999982 -j KUBE-SEP-YXTOFVFTDE2Z2ZYN
-A KUBE-SVC-NRQ7NG2MM3ZIPEW -m statistic --mode random --probability 0.50000000000 -j KUBE-SEP-4NHG7AZOC2Z4ZFPP
-A KUBE-SVC-NRQ7NG2MM3ZIPEW -j KUBE-SEP-5SCL2ZSG5JXLXYEL
```

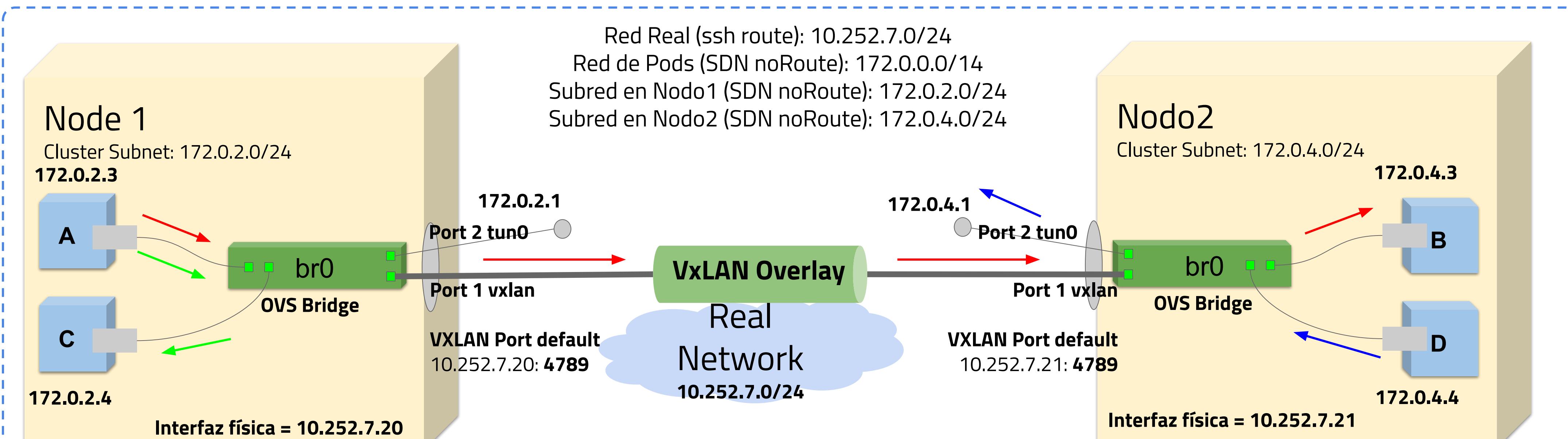
Management Networking

OpenShift SDN Modes

- Utiliza **Open vSwitch (OVS)** para configurar el overlay de red.
- Tres tipos de plugins de SDN.
 - **ovs-subnet**: red de tipo plana, todos los pods se comunican con todos.
 - **ovs-multitenant**: aislación a nivel de proyecto (namespace) para la comunicación pod a pod.
 - Único VNID por proyecto.
 - Default posee el VNID 0 y es global.
 - **ovs-networkpolicy**: permite un grado de aislación más granular utilizando objetos de kubernetes llamados **Network Policy**

Management Networking

OpenShift SDN Modes



PodA a PodB: eth0(pod A in netns) -> vethA -[Br0] - vxlan0 - [Network] - vxlan0 - [br0] - vethB - eth0 (pod B in netns)

PodA a PodC: eth0(pod A in netns) -> vethA -[Br0] -> vethB - eth0 (pod C in netns)

PodD a Inet: eth0(pod D in netns) -> vethD -[br0]- tun0 (iptables source nat) - Real Network

Kubernetes sobre Infra On-Premise (1 DC) o Cloud (1 Region)

SDN:

<https://docs.openshift.com/container-platform/4.2/networking/openshift-sdn/about-openshift-sdn.html>

Interface TAP/TUN: <https://en.wikipedia.org/wiki/TUN/TAP> (SDN: tun > layer 3 > ip // tap > layer 2 > mac)

Kubernetes Service: <https://kubernetes.io/docs/concepts/services-networking/service/>

Management Networking

Network Policies

Overview

- Requiere configuración: `redhat/openshift-ovs-networkpolicy`
- Habilita el control de acceso a puertos de un modo más granular.
- Tráfico puede ser controlado a nivel namespace, pod, port.
- Pods con uno o más reglas de NetworkPolicy que apunten a ellas serán aisladas.
- Las definiciones de recursos de EgressNetworkPolicy restringen de pods a recursos externos al SDN

Políticas básicas

- Son aplicadas a nivel namespaces (projects)
- Unidireccional: OpenShift SDN soporta políticas de ingress solamente.
- Adicional: Pueden agregarse varias políticas a un namespace
- Usa label y selectos para seleccionar pods destino, namespaces y pods origines.
- No se necesita ser cluster-admin para agregar, cambiar y eliminar políticas.
- Los cluster role admin y edit pueden agregar, modificar y borrar reglas de networkpolicy.

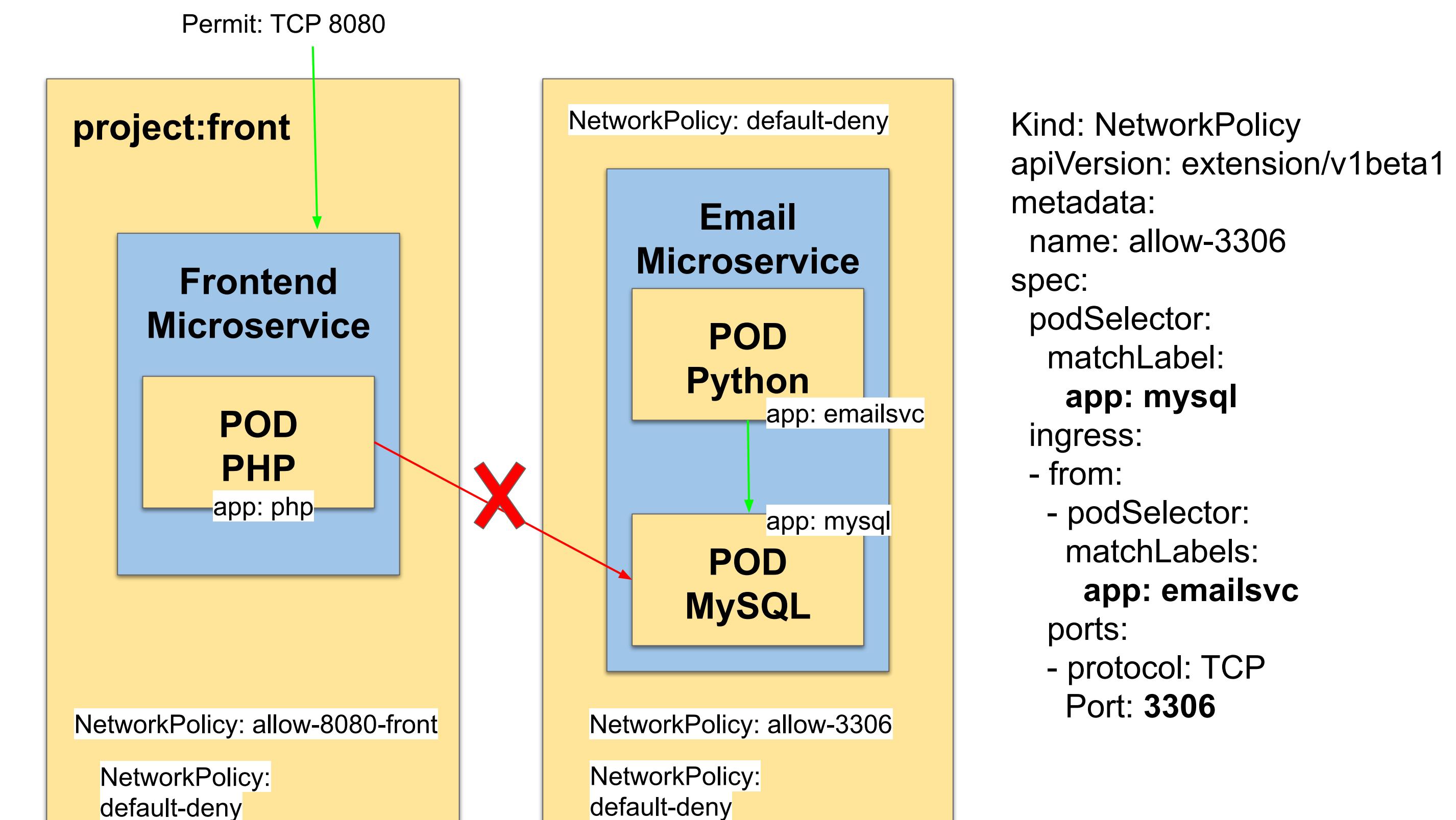
Management Networking

Network Policies

El objeto **Network Policy** dentro de Openshift permite definir las reglas de firewall entre namespaces

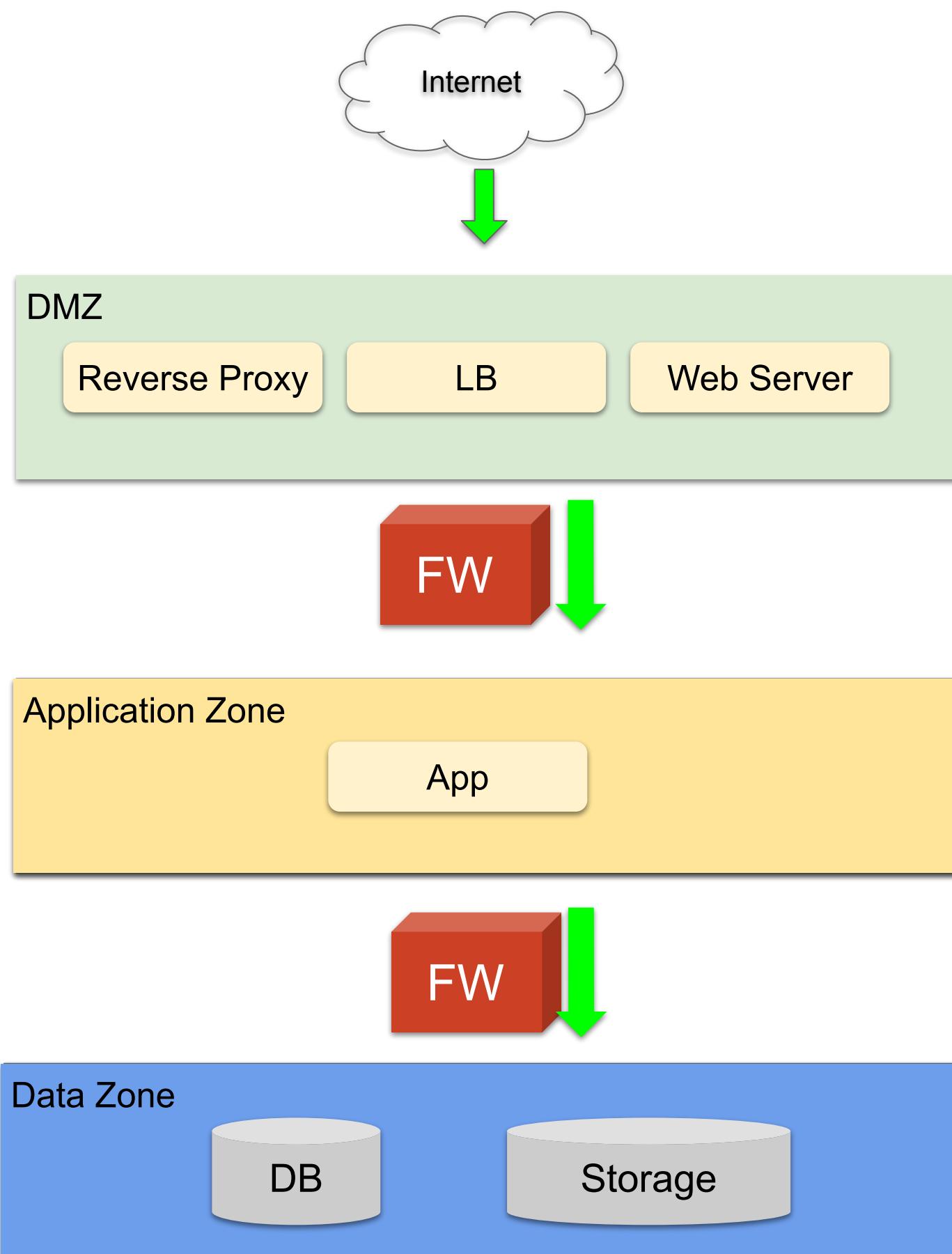
- **Microsegmentacion**

- Permite la configuración de *políticas a nivel de pod*.
- Aplica al tráfico entrante (*ingress*) para pods y servicios.
- Permite restringir el tráfico entre pods en un project/namespaces
- Permite el tráfico específico desde otros project/namespaces



Network Zones

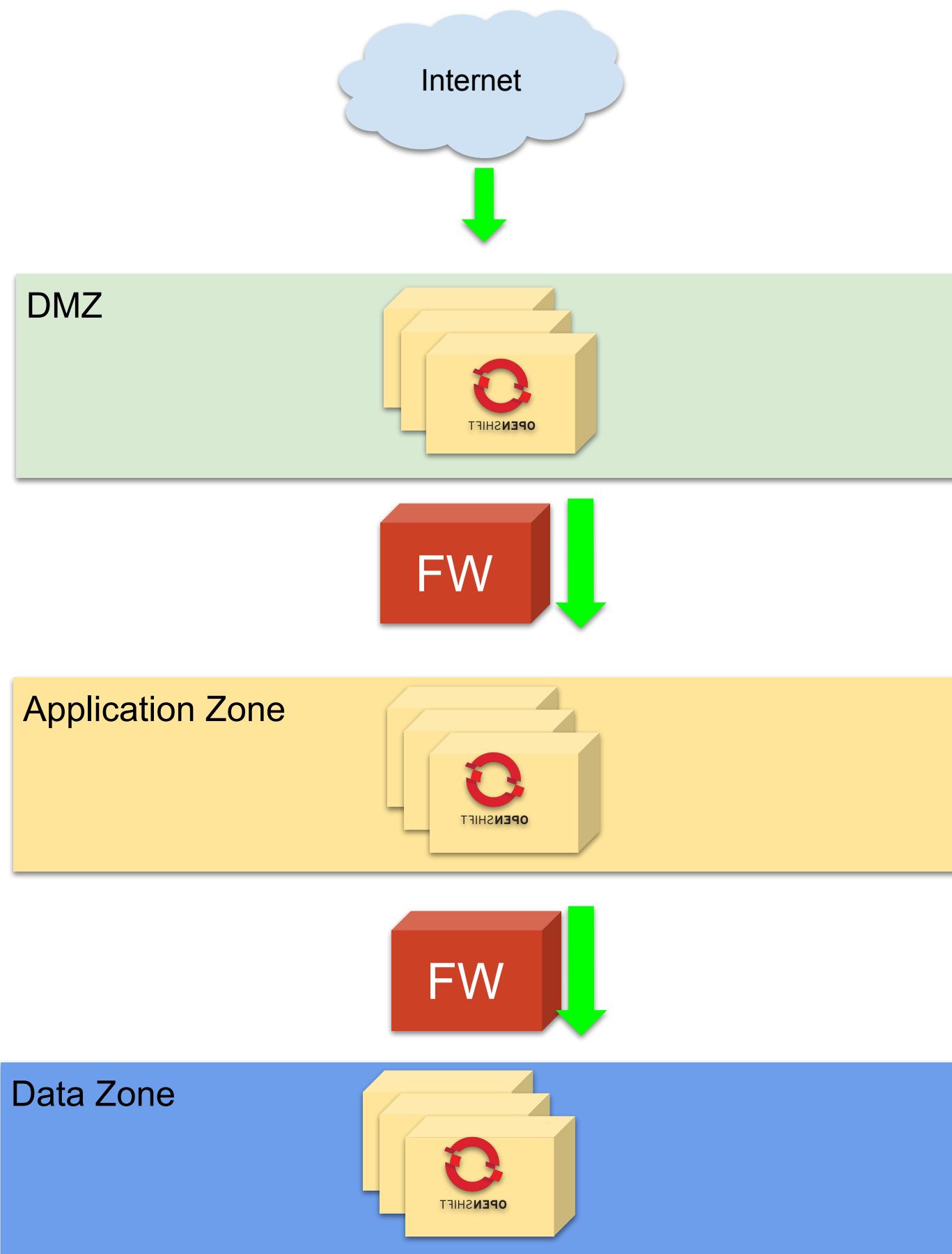
Opción 1: Openshift Cluster por Zonas



- El tráfico externo de internet ingresa por la zona desmilitarizada **DMZ**.
- Permitir tráfico específico desde ciertos orígenes origen.
- Desde DMZ a Application Zones.
- Permitir tráfico específico desde App Zone hacia Data Zone

Network Zones

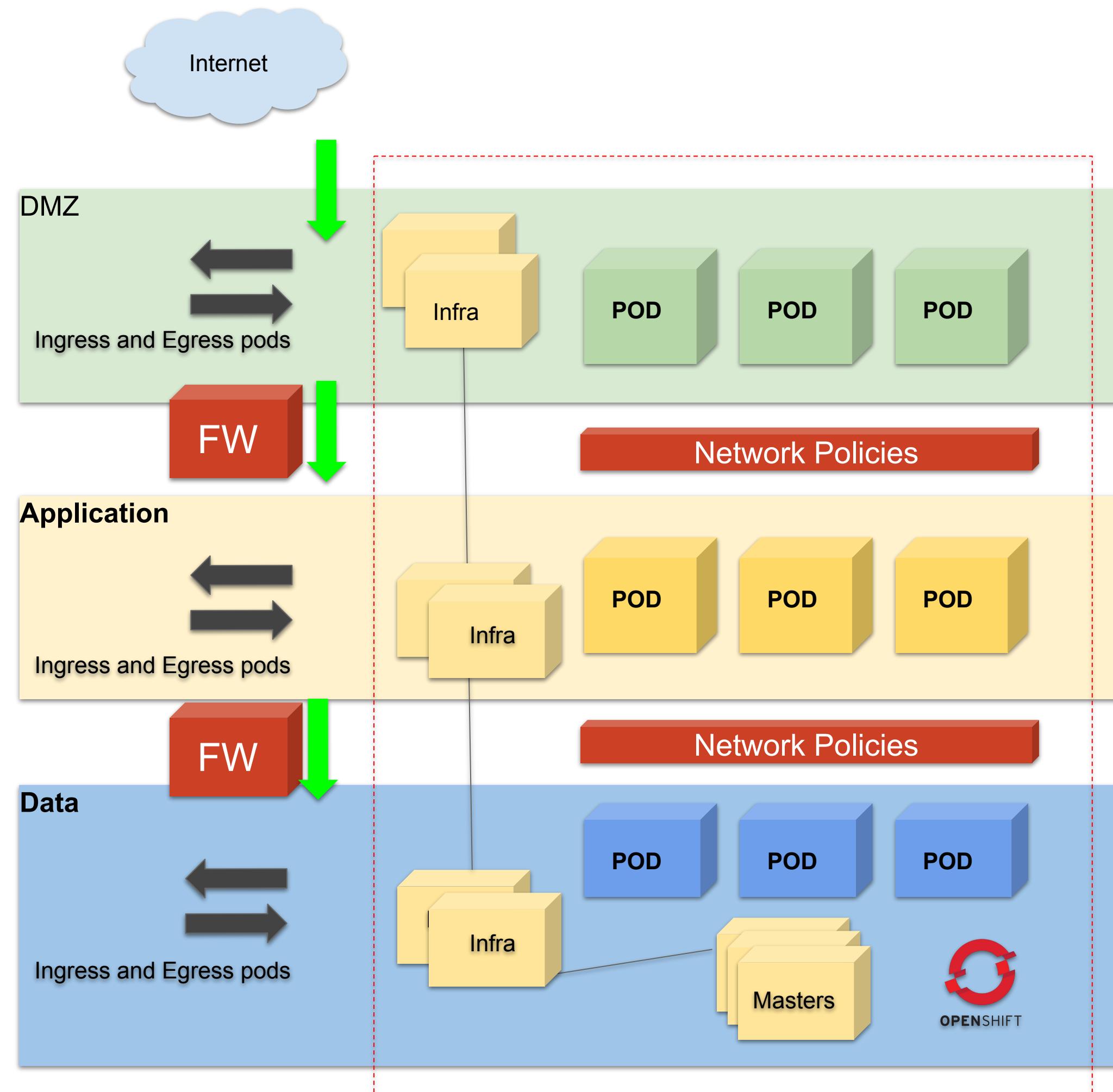
Opción 1: Openshift Cluster por Zonas



- Usualmente el modelo que demuestra ser compliance con los estándares de los equipos de seguridad y regulaciones.
- Acciones adicionales necesarias para proteger las Master's API y otras URL en DMZ que no son necesarias exponer a internet.
- Costo de mantenimiento más elevado.

Network Zones

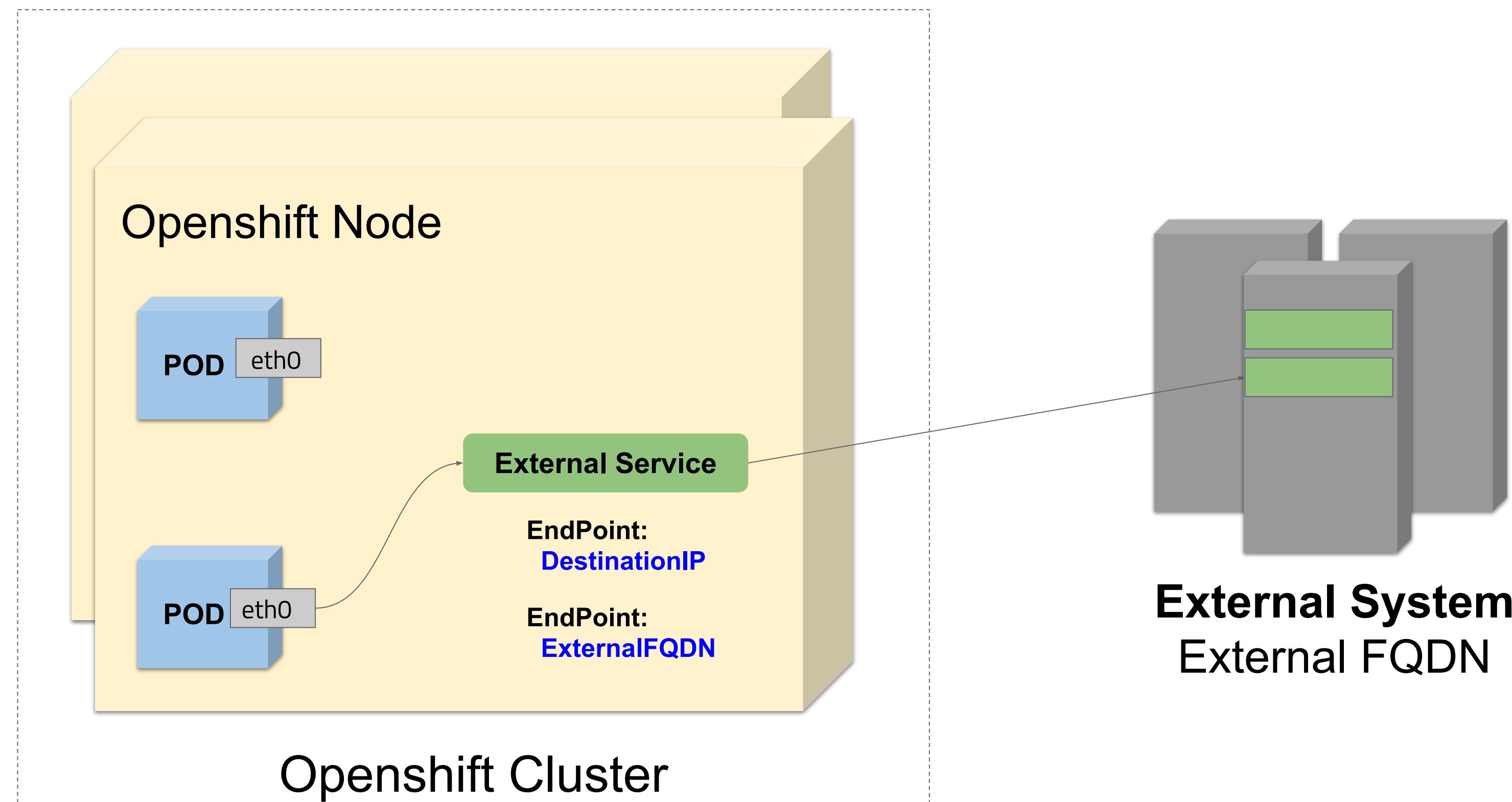
Opción 2: Openshift Cluster cubriendo Multiples Zonas



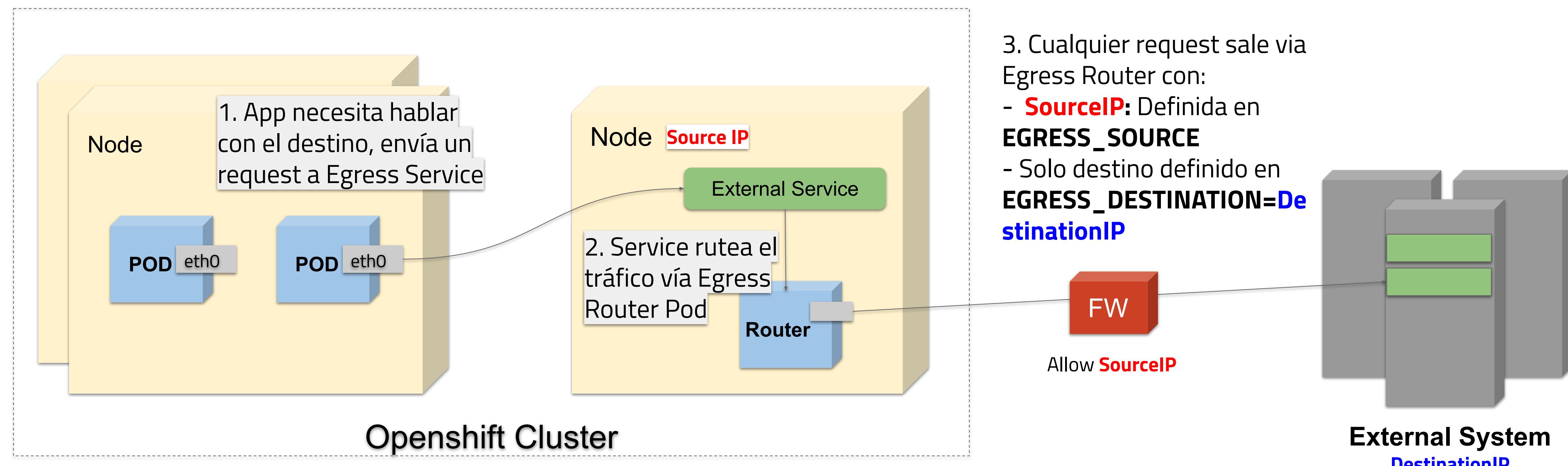
- Los pods de aplicación corren en un solo Openshift Cluster microsegmentado con Network Policies.
- Nodos de infra en cada zona corren pods de Ingress y Egress para cada zona.
- Si es requerida aislación física de pods, se puede utilizar el selector de nodos (nodeSelector).
- Los nodos masters se encuentran en la zona menos expuesta.
- Recomendación, microsegmentación via SDN.

Egress Service

La aplicación se conecta con el sistema externo hablando con **External Service** quien define en su Endpoint la **IP/Port** destino o el **FQDN** destino.

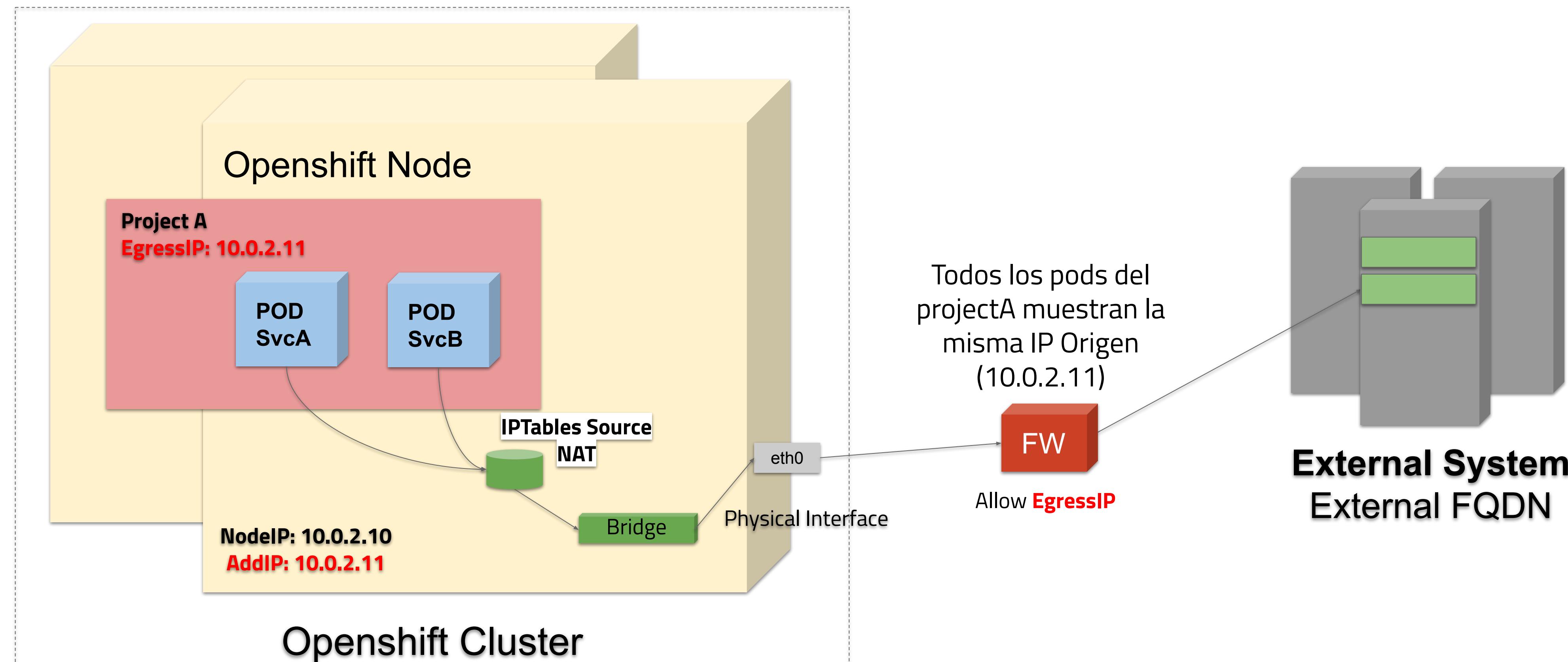


Egress Router



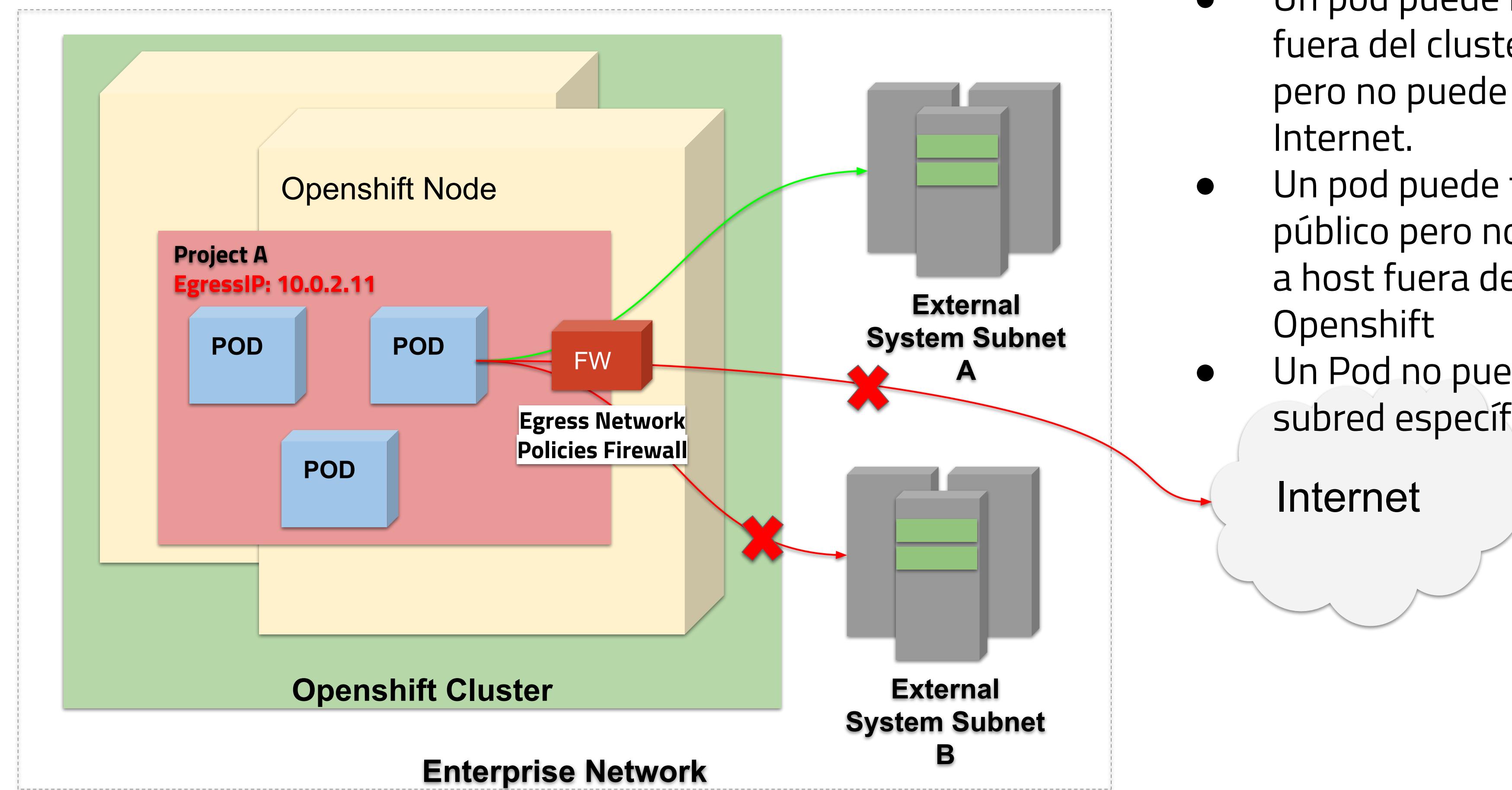
Egress vía Static IP

Un proyecto (Project A) define puede definir una ip de Egress por namespaces de modo que todos los pods que vivan en ese proyecto tenga la misma sourceIP.



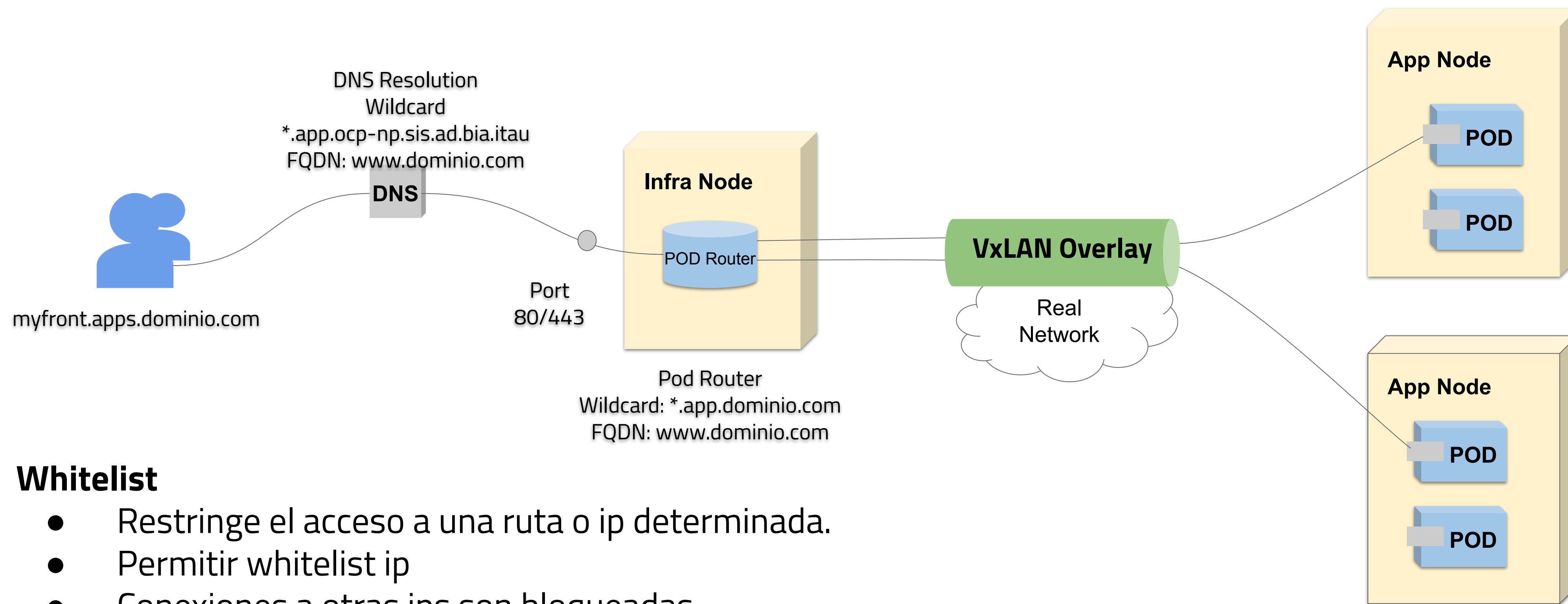
Egress Firewall

Un cluster admin puede limitar la salida de los pods a determinados sistemas aplicando **Network Policies**



Openshift Ingress Router

Traffic Flow

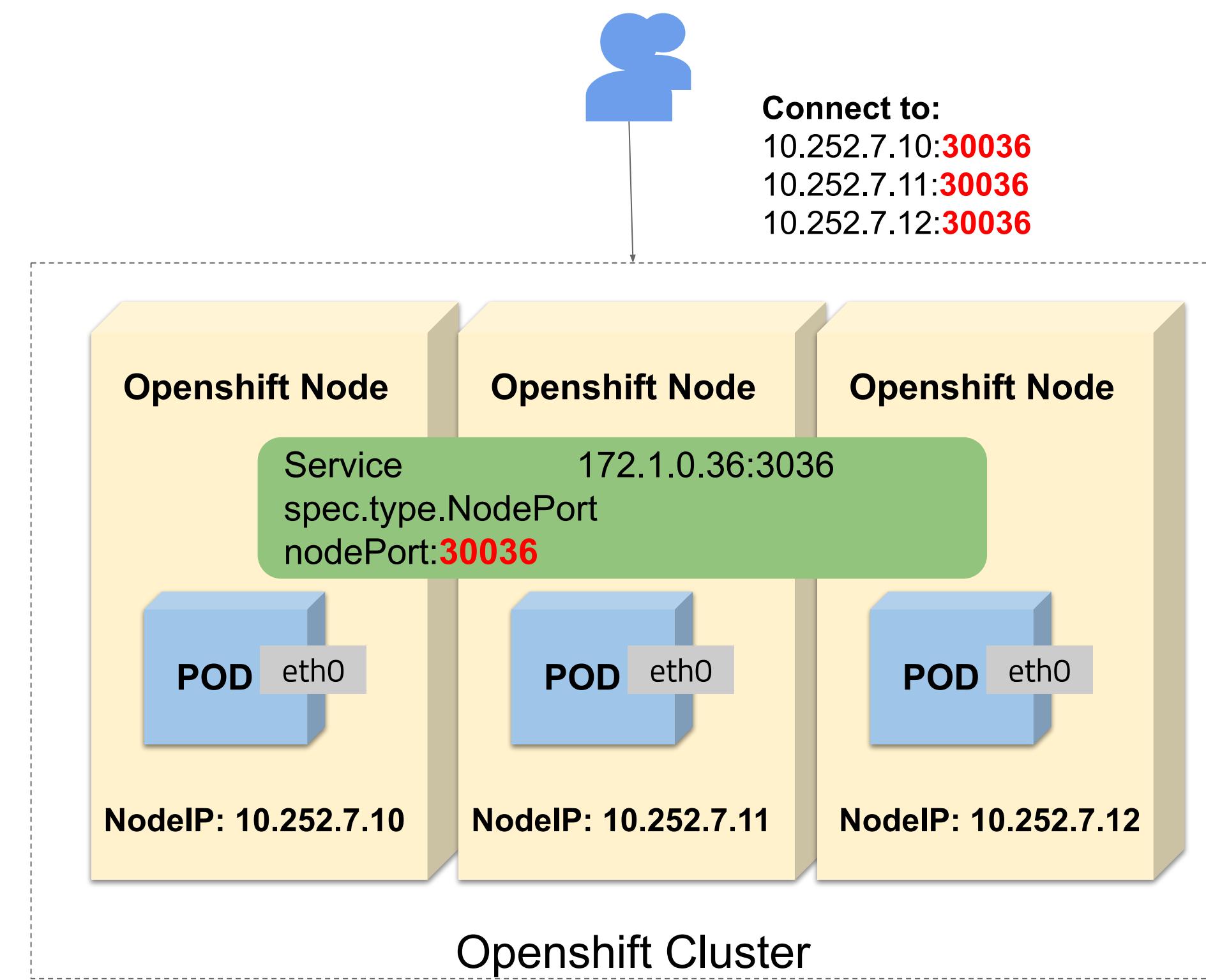


Whitelist

- Restringe el acceso a una ruta o ip determinada.
 - Permitir whitelist ip
 - Conexiones a otras ips son bloqueadas
- metada:
- annotations:
- haproxy.router.openshift.io/ip_whitelist: 192.168.2.10 192.168.1.12

<https://docs.openshift.com/container-platform/4.2/networking/routes/route-configuration.html>

Openshift Ingress - NodePort



El servicio escucha en un único puerto en todos los nodos.

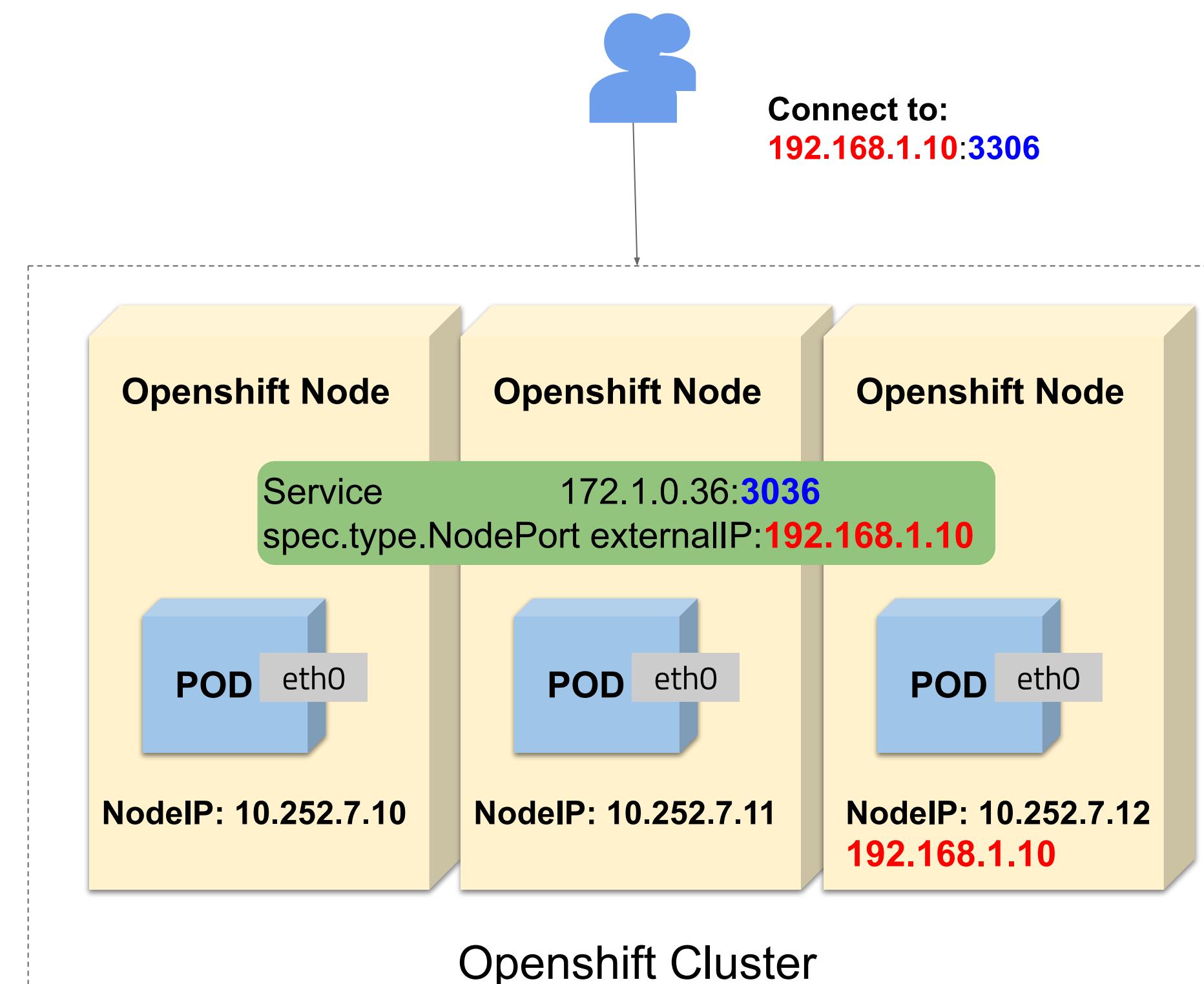
Puerto es random y asignado dinámicamente entre 30000-32767.

Todos los nodos actúan como ingress point en el nodo asignado.

Todos los nodos del cluster redirigen el tráfico al servicio y este al endpoint aun cuando el pod no esté en el nodo.

Reglas de firewall de los nodos deben ser agregadas para el puerto asignado dinámicamente.

Openshift Ingress - External IP



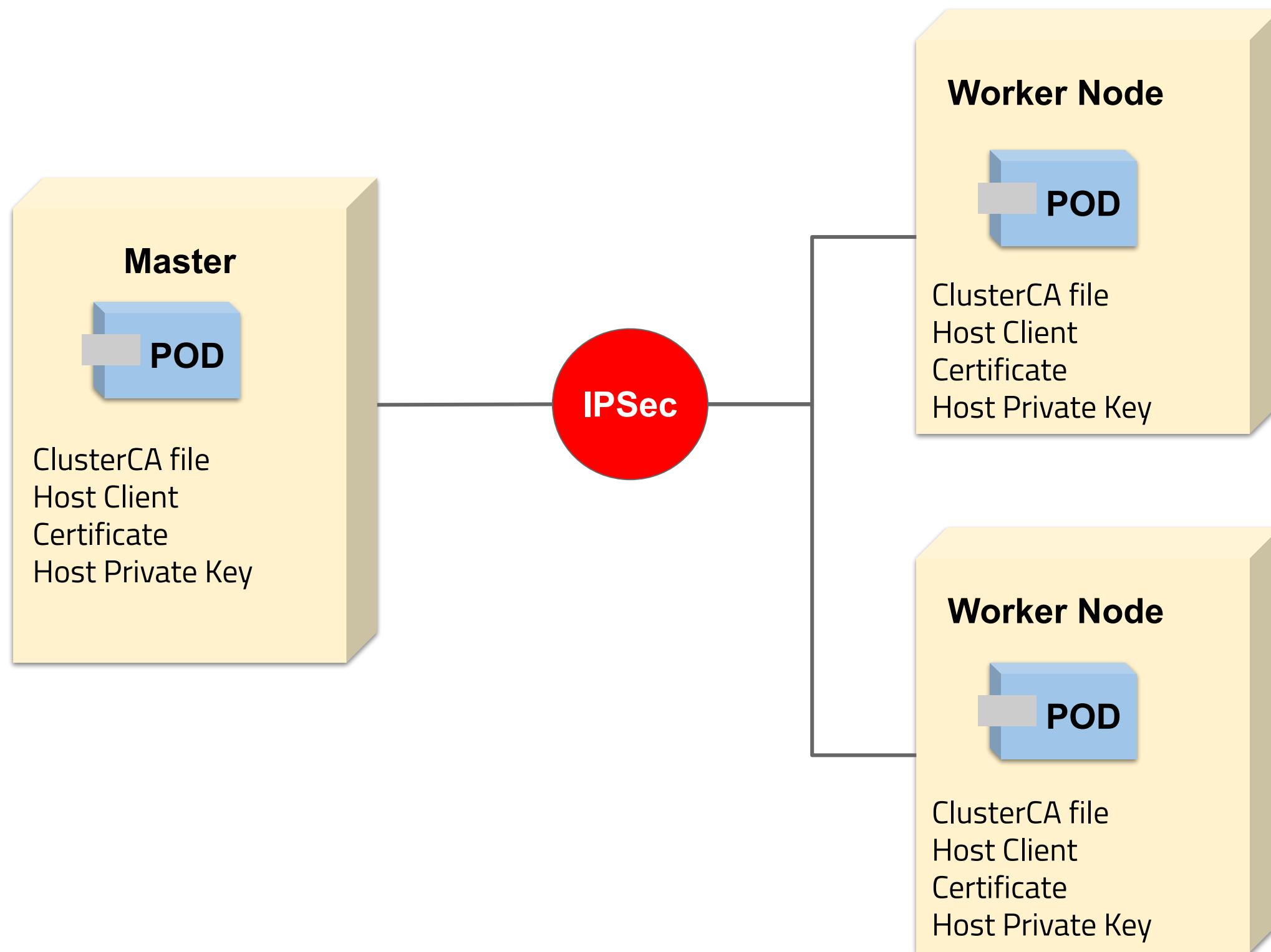
Admin define una External IP, puede ser un rango y la asignación se hace dinámica.

Openshift asigna ambas una Internal IP y una External IP a un Servicio. O solo se puede asignar la External IP.

El nodo quién tiene la External IP es el nodo Ingress Point del tráfico de servicio.

External IP puede ser una VIP. Para reasignación o alta disponibilidad de la VIP, puede utilizarse un pod de **ipfailover** (pod privilegiado).

Secured Communication between Host

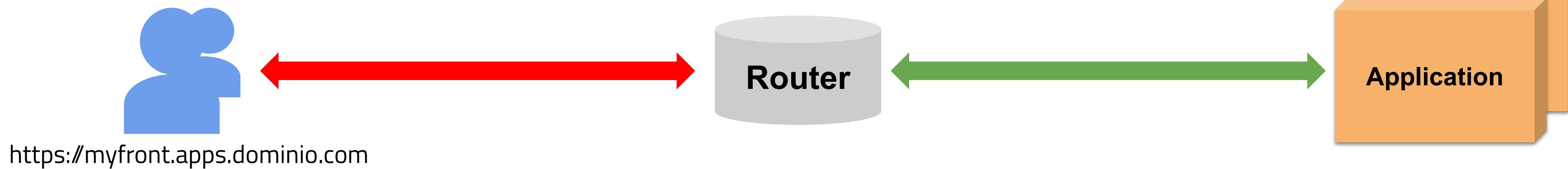


Toda la comunicación entre nodos del cluster es segura con IPsec.

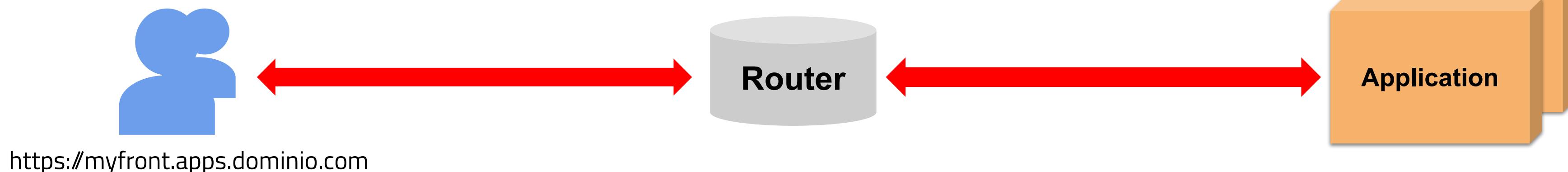
- Encripción entre Master y Worker Nodes (L3)
- OpenshiftCA y Certs

Application Layer - Openshift Router SSL

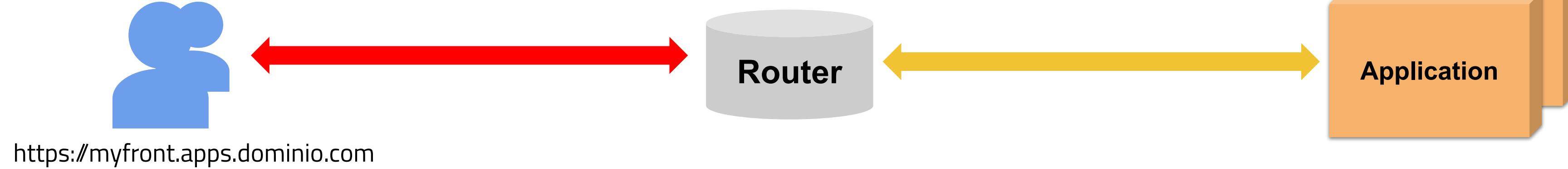
Edge Termination



Passthrough Termination



Reencrypt



Alta Disponibilidad (HA)

Semperti

Kubernetes un Sitio

High Availability (HA)

Application

Virtual Machines

Physical
Machines

Storage

Electric & Power

Hypervisor

Network
Partition

Cooling System

Kubernetes un Sitio

Application HA

Builds Apps

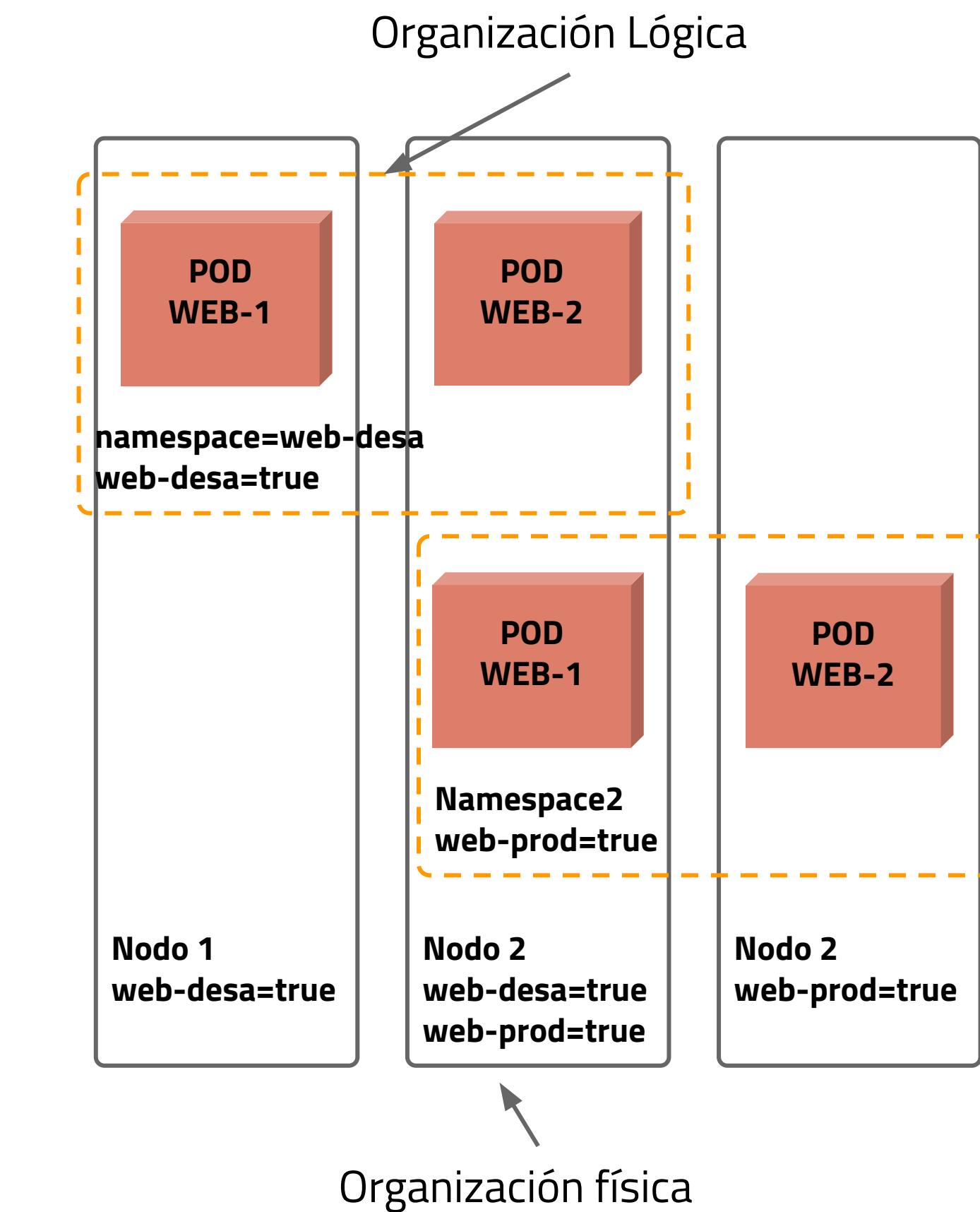
- Construir aplicaciones con componentes **escalables y con capacidad de sostener la pérdida de PODs.**
 - Al menos 2 pods por componente
 - Apps Web, message tier, datastore, etc.

Deploy Apps

- Default, los pods **en nodos diferentes.**
- Estrategias de Rolling Updates, uno a la vez.
- Usar **Horizontal Pod Autoscaling (HPA) (1).**
 - Múltiples métricas pueden ser utilizadas
 - CPU, Memoria, custom por medio de stack de metricas.

NodeSelectors

- Reglas de afinidad, placement groups, no desplegar apps en un mismo nodo.
- Labels!!.. Via Network-Zone, AZ, Rack, Tecnología.
- User **NodeSelector (2)** en base a labels. Ej. network-zone, AZ, racks, tecnología de computo, etc.



- (1) <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/>
(2) <https://kubernetes.io/docs/concepts/configuration/assign-pod-node/>

Kubernetes un Sitio

Application HA

Stateful Apps

- Las aplicaciones que persistan datos tiene que tener la capacidad de poder realizar un **re-attach del volumen** persistente (PV).
- **NO HostPath**, utilizar storage backend dependiendo y definir de manera certera la necesidad. Storage ReadWriteOne o ReadWriteMany.

Application Health Check

Detectar cuando un pod está falla y tomar acciones.

- **Readiness probes:** Si no pasa probes, no se agrega como endpoint.
- **Liveness probes:** Pod running, checks periodicos, si no pasa se reinicia.
 - Tipos:
 - HTTP checks:
 - Container execution
 - TCP Socket

Consumo de Computo

Monitoreo constante de aplicaciones para ajustar Request y Limits de CPU y Memoria.

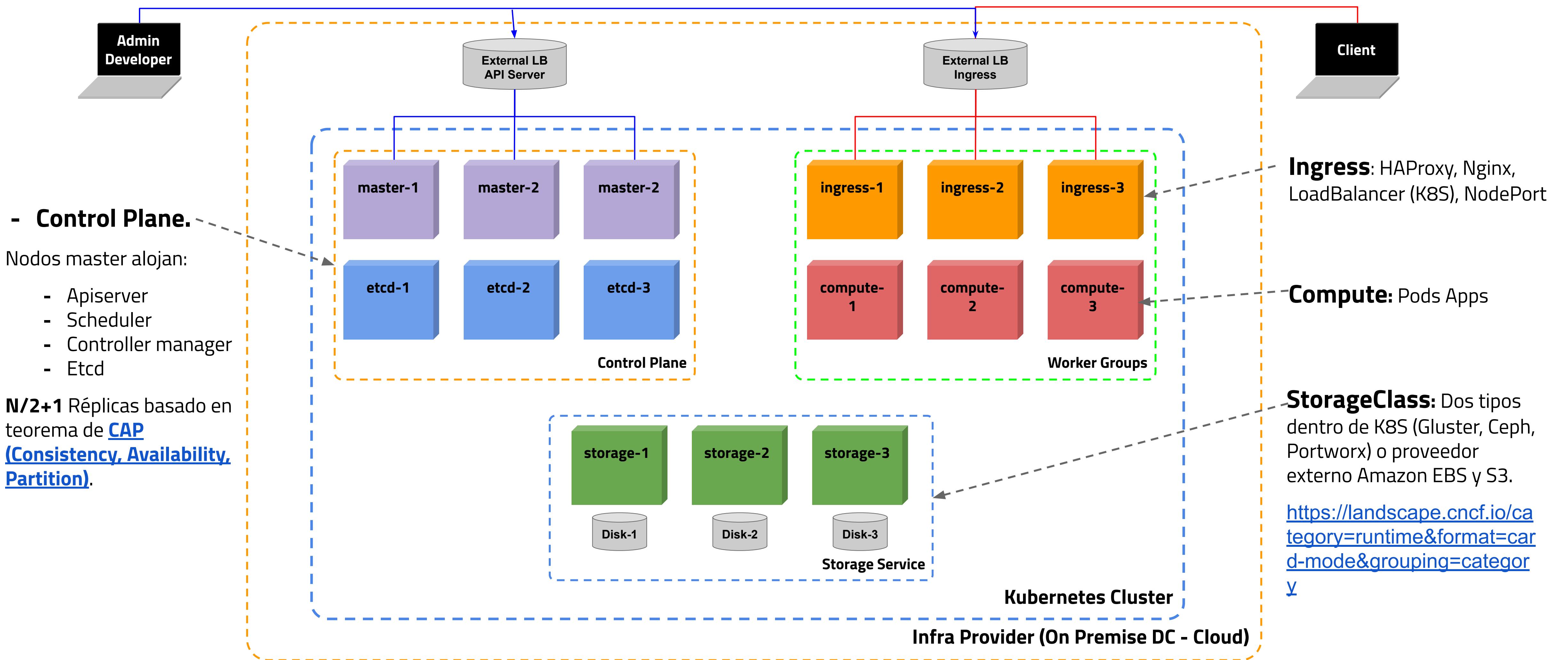
<https://docs.openshift.com/container-platform/4.3/applications/application-health.html>

<https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/>

```
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-http
spec:
  containers:
  - name: liveness-http
    image: k8s.gcr.io/liveness
    args:
    - /server
    livenessProbe:
      httpGet:
        # host: my-host
        # scheme: HTTPS
        path: /healthz
        port: 8080
        httpHeaders:
        - name: X-Custom-Header
          value: Awesome
      initialDelaySeconds: 15
      timeoutSeconds: 1
    name: liveness
```

Kubernetes un Sitio

Architecture HA (Control Plane)



Mas información

Links de interés

Disaster Recovery with Containers? You Bet!

<https://keithtenzer.com/2018/03/21/disaster-recovery-with-containers-you-bet/>

The difference between disaster avoidance and recovery

<https://searchservervirtualization.techtarget.com/feature/The-difference-between-disaster-avoidance-and-recovery>

Disaster Recovery Strategies for Applications Running on OpenShift

<https://blog.openshift.com/disaster-recovery-strategies-for-applications-running-on-openshift/>

Stateful Workloads and the Two Data Center Conundrum

<https://blog.openshift.com/stateful-workloads-and-the-two-data-center-conundrum/>

Deploying OpenShift Applications to Multiple Datacenters - Part 0

<https://blog.openshift.com/deploying-openshift-applications-multiple-datacenters/#>

Connecting Multiple OpenShift SDNs with a Network Tunnel - Part 1

<https://blog.openshift.com/connecting-multiple-openshift-sdns-with-a-network-tunnel/>

Connecting Multiple OpenShift SDNs with a Network Tunnel – Part 2: Service Proxying and Discovery

<https://blog.openshift.com/connecting-multiple-openshift-sdns-with-a-network-tunnel-part-2-service-proxying-and-discovery/>

Openshift SDN Encrypted Tunnel

<https://github.com/raffaelespazzoli/openshift-sdn-encrypted-tunnel>

Istio Multicluster on OpenShift - Part 3

<https://blog.openshift.com/istio-multicluster-on-openshift/>

Combining Federation V2 and Istio Multicluster - Part 4

<https://blog.openshift.com/combining-federation-v2-and-istio-mycluster/>

A Self-Hosted Global Load Balancer for OpenShift - Part 5

<https://blog.openshift.com/a-self-hosted-global-load-balancer-for-openshift/>

Self-Serviced, End-to-End Encryption for Kubernetes Applications, Part 2: a Practical Example

<https://blog.openshift.com/self-serviced-end-to-end-encryption-for-kubernetes-applications-part-2-a-practical-example/>

Kubernetes Federation

<https://platform9.com/blog/kubernetes-federation-what-it-is-and-how-to-set-it-up/>

Multicluster Controller

<https://github.com/admiraltyio/multicluster-controller>

Muchas gracias!

Gonzalo Acosta <gonzalo.acosta@semperti.com>

Semperti