

OpenShift Desarrollo

Gonzalo Acosta <gonzalo.acosta@semperti.com>

Agenda

Día 1	Día 2	Día 3	Día 4	Día 5
Container Technology	Container Orchestration K8S and OCP	Builds and Deployments Application Configuration	Authentication and Authorization Resource Management	Application Placement Monitoring Logging Networking

Día 1

Semperti

Container Technologies

Semperti

Container Technology

Introducción

Nuestra industria está cambiando. No importa en qué tipo de compañía estés trabajando. Tampoco importa el tipo de industria en la que estés adentro. *Las herramientas de software son la base en las que se construyen las compañías del futuro sean pequeñas o grandes.*

En el centro de este cambio está la evolución del desarrollo del software. *Se necesitan prácticas ágiles para responder a la competencia o los cambios de mercado.*

Los **contenedores** de linux se han lanzado al mercado como **la nueva base sobre la que construir conexiones entre el desarrollo y la operación**. Y aprovechar la infraestructura para las aplicaciones, desde la nube pública hasta la infraestructura privada, es una necesidad.



Container Technology

Despliegue tradicional

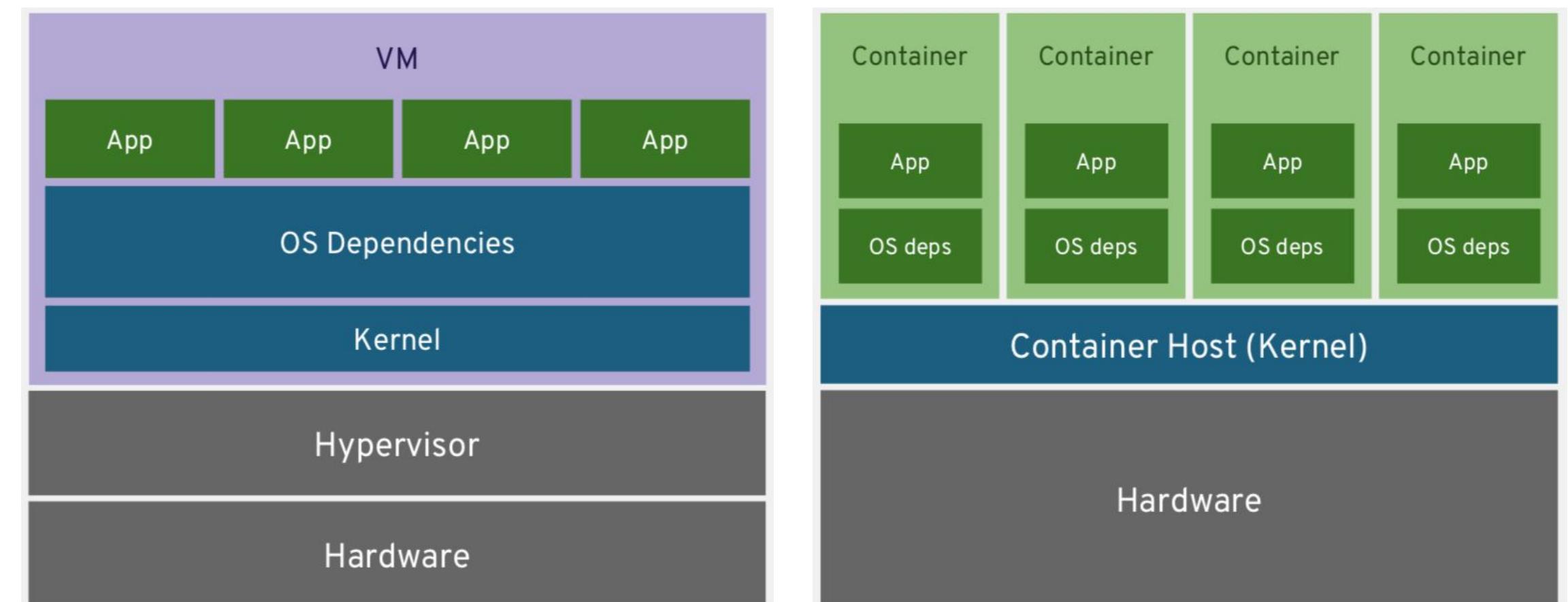
Provisión tradicional

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



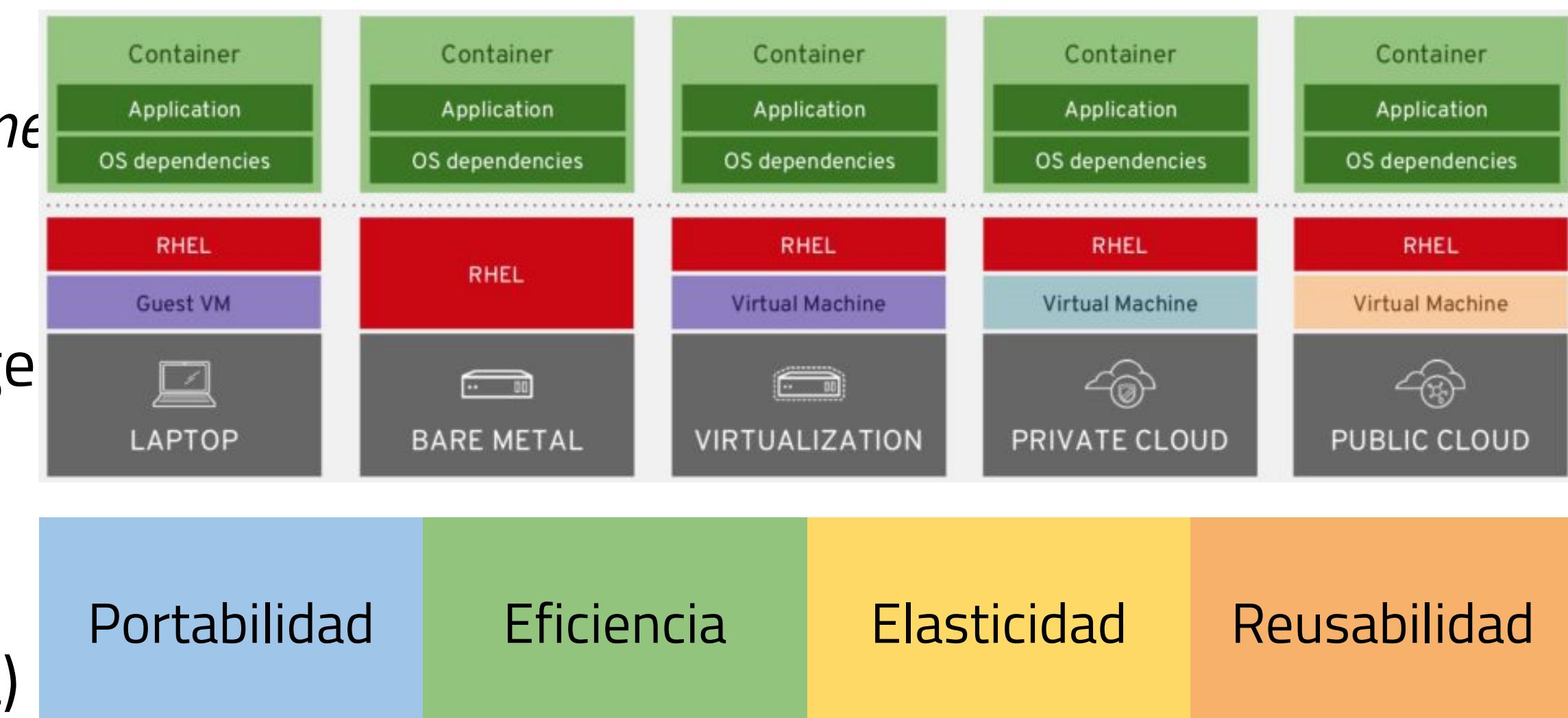
Container Technology

Contenedores

Provisión en contenedores

Un **contenedor** es un set de uno o más procesos de manejo 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)



Container Technology

Estandares

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.

Container Network Interfaces (CNI).

Proyecto sponsoreado por CNCF, tiene como objetivo estandarizar la interfaz de red de los contenedores en entornos nativos de nube, como kubernetes y Openshift.

<https://opencontainers.org>

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



Container Technology

Beneficios de las aplicaciones en contenedores

- **Low hardware footprint**

Los contenedores usan características internas del sistema operativo para poder crear un ambiente aislado. Control Groups (cgroup).

- **Environment Isolation**

Los contenedores trabajan en ambientes cerrados donde los cambios en el host o en otras aplicaciones no afectan al funcionamiento del contenedor.

- **Quick Deployment**

Los contenedores pueden instanciarse rápidamente porque no necesitan de toda la capa de sistema operativo que se encuentra por debajo.

- **Multiple environment deployment**

Toda las dependencias de la aplicación y su configuración de ambiente se encuentran encapsuladas en la imagen del contenedor.

- **Reusability**

Una imagen de un contenedor de base de datos puede ser utilizada por diferentes desarrolladores para levantar instancias individuales.



Container Technology

Features del kernel para contenedores

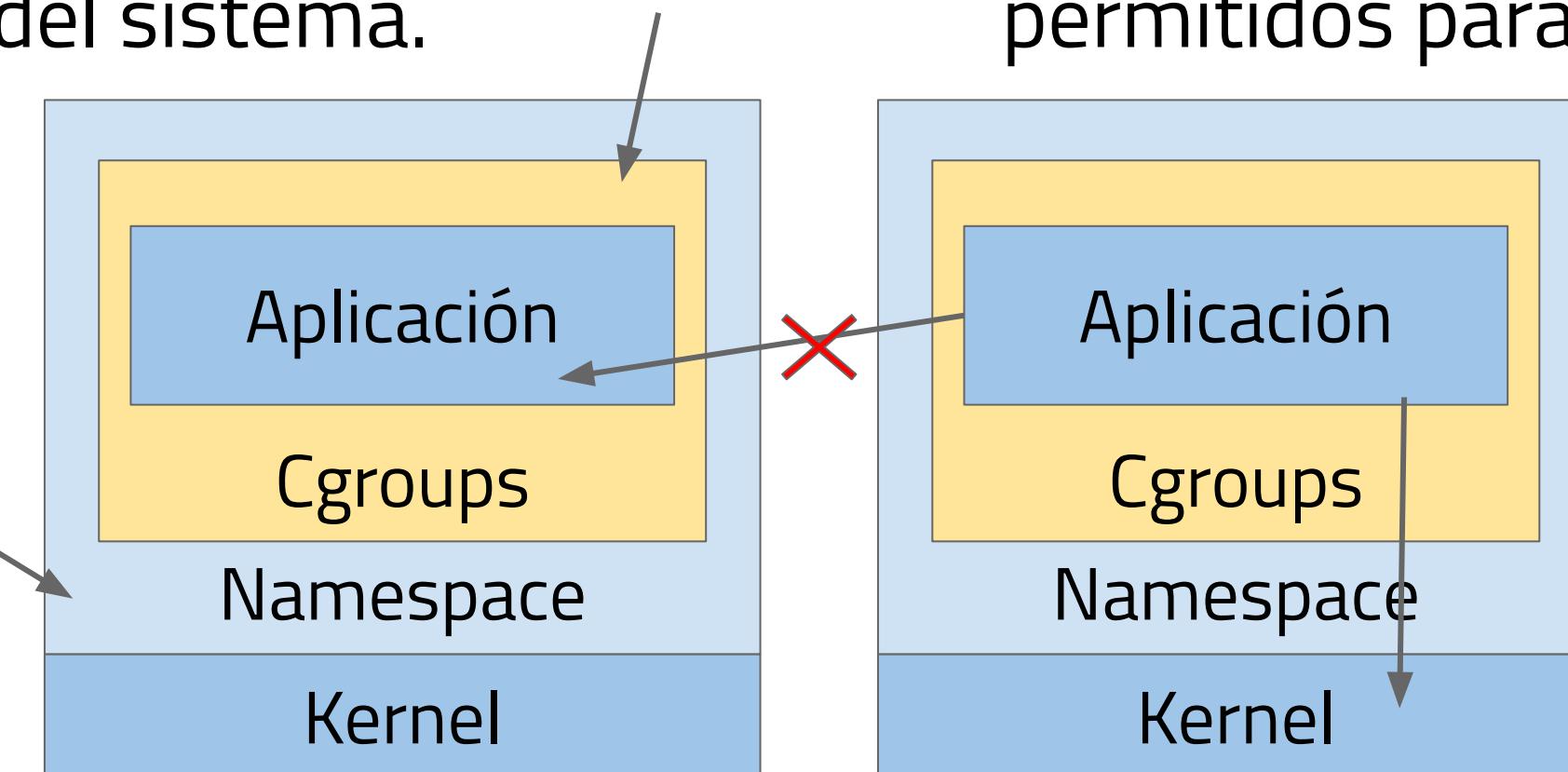
El kernel de linux posee una serie de features que permiten aislar procesos:

- **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.



- **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:**

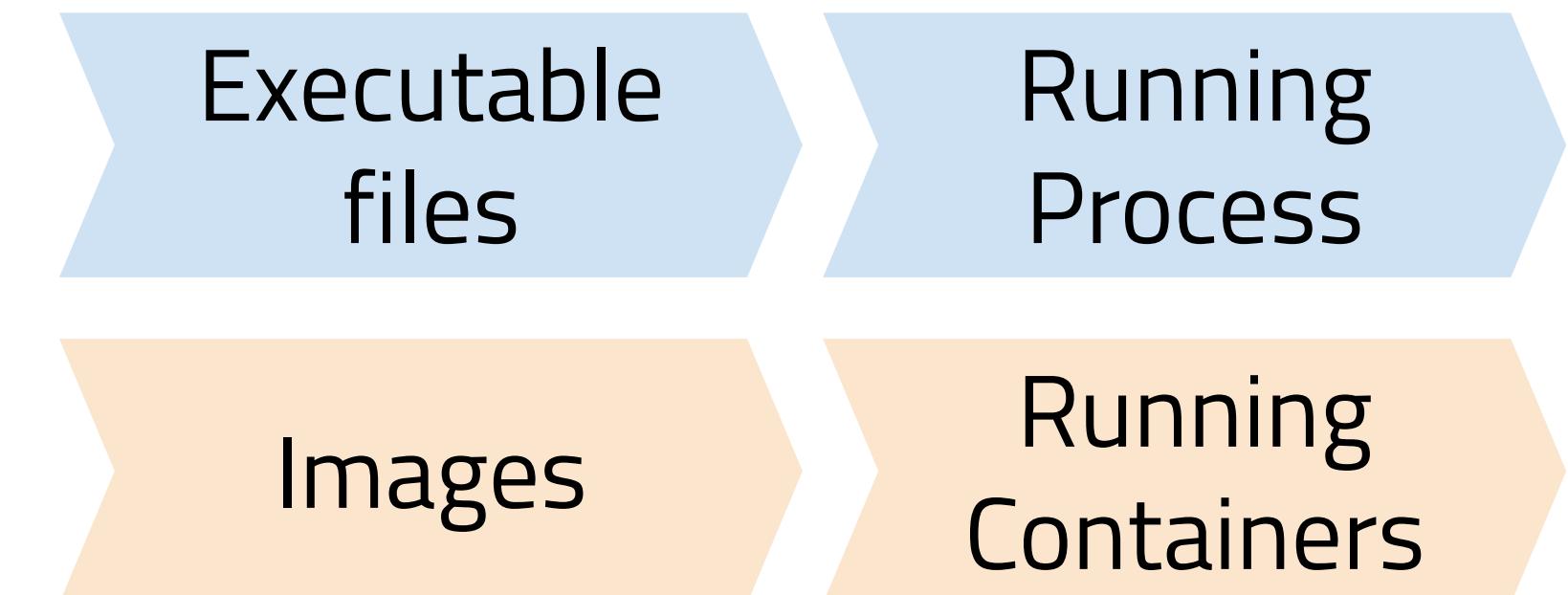
SELinux (Security-Enhanced Linux) definen el sistema de control de acceso para procesos. El kernel de linux utiliza SELinux para proteger procesos de otros procesos y también limitar el acceso a archivos de sistema del host.

Container Technology

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

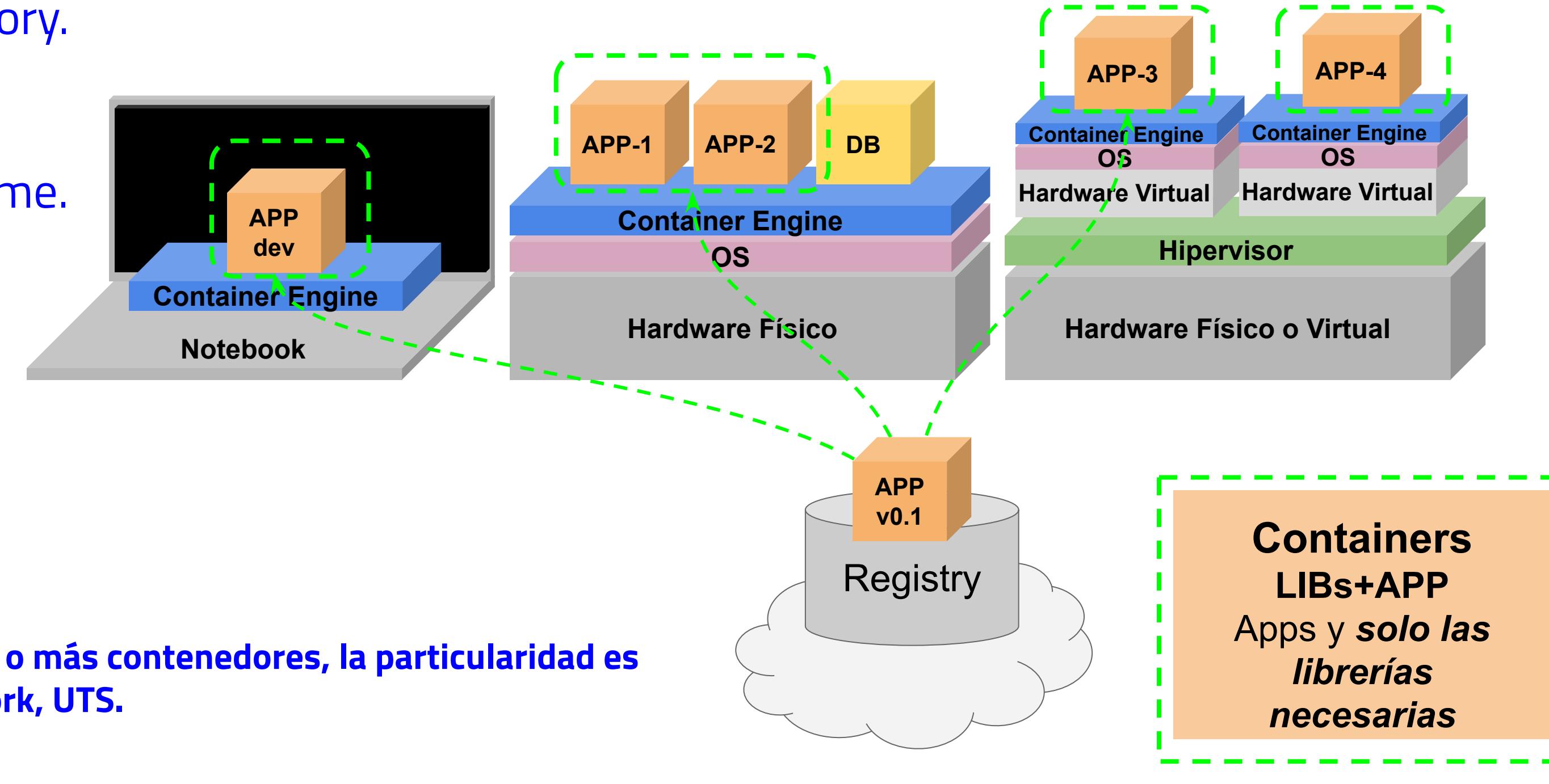


Container Technology

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.



Container Technology

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.

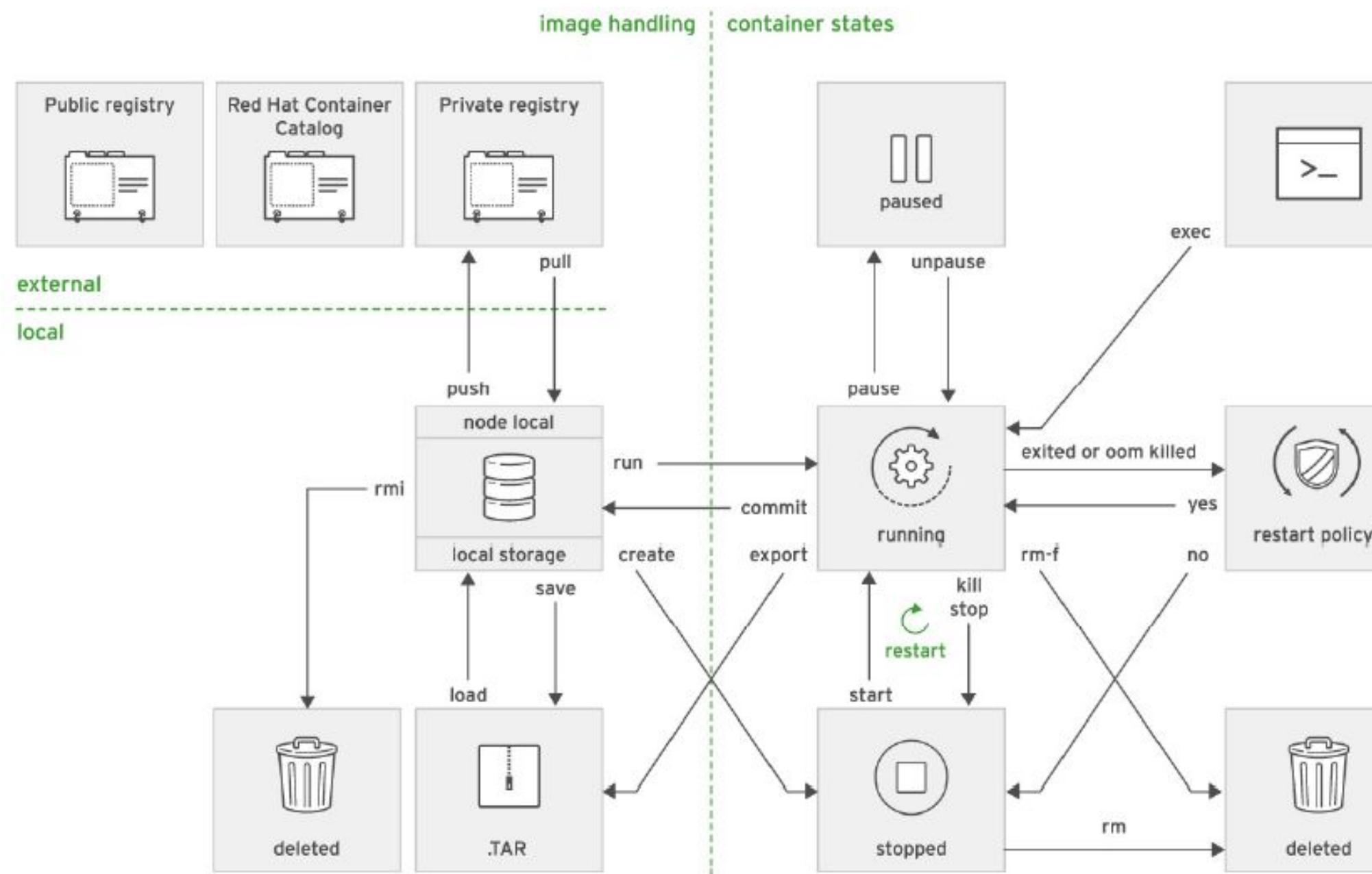
```
# Red Hat  
$ sudo yum -y install podman
```

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

Container Technology

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
```

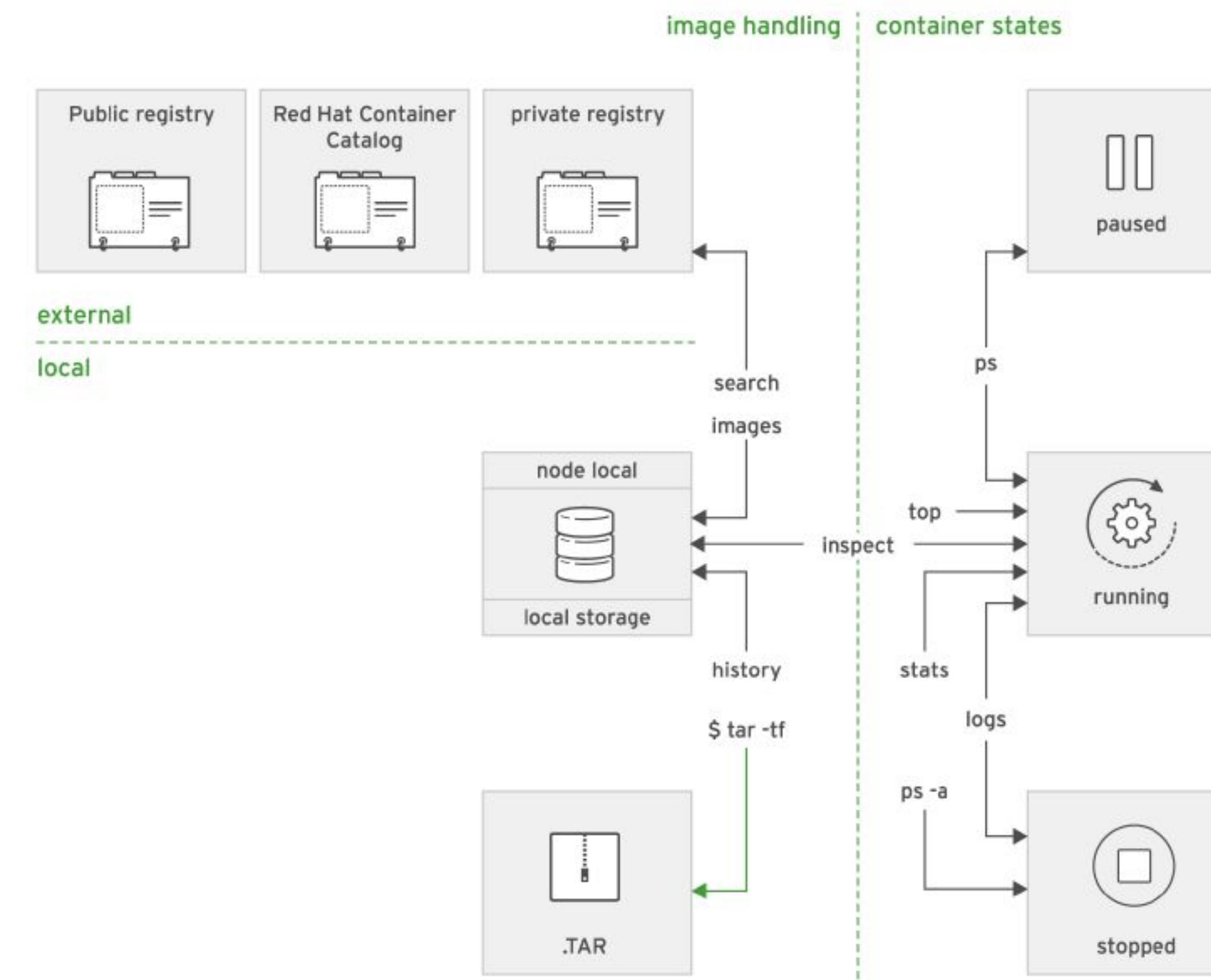
- **Dirección ip del contenedor**

```
$ 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

Lab 1 - Ejercicio 1 a 4

Tecnología de Contenedores

Almacenamiento efímero en Contenedores

Almacenamiento en contenedores

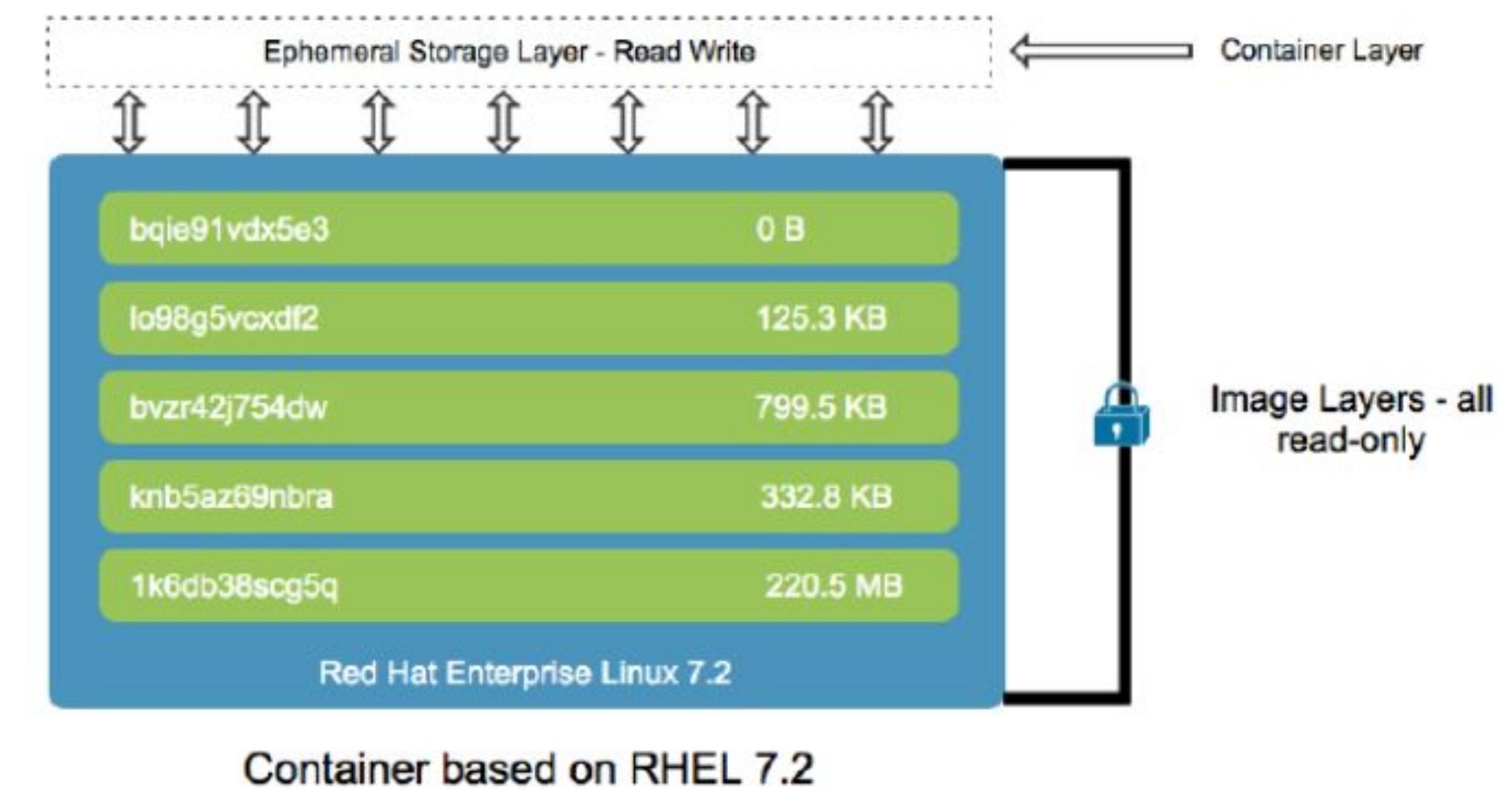
Se dice que el almacenamiento (**storage**) de los contenedores es **efímero**, no persiste tras un restart. Las *aplicaciones asumen que inician con un storage vacío*. Permite que los contenedores puedan ser iniciados y detenidos inesperadamente.

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**.

*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.*

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/e5e0e11df03bc84c5  
d5f5548578d524ac63c9ad46e0de60c7a7c2477b6258e8b/diff  
  
$ sudo podman inspect httpd-2 -f  
"{{.GraphDriver.Data.UpperDir}}"  
  
/var/lib/containers/storage/overlay/00da1349de8cbd3ff  
41c6f4c09ad92d8ae89a25472dd317dd1f61865fd63cf3b/diff
```

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

Lab 1 - Ejercicio 6

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-Defined 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/>

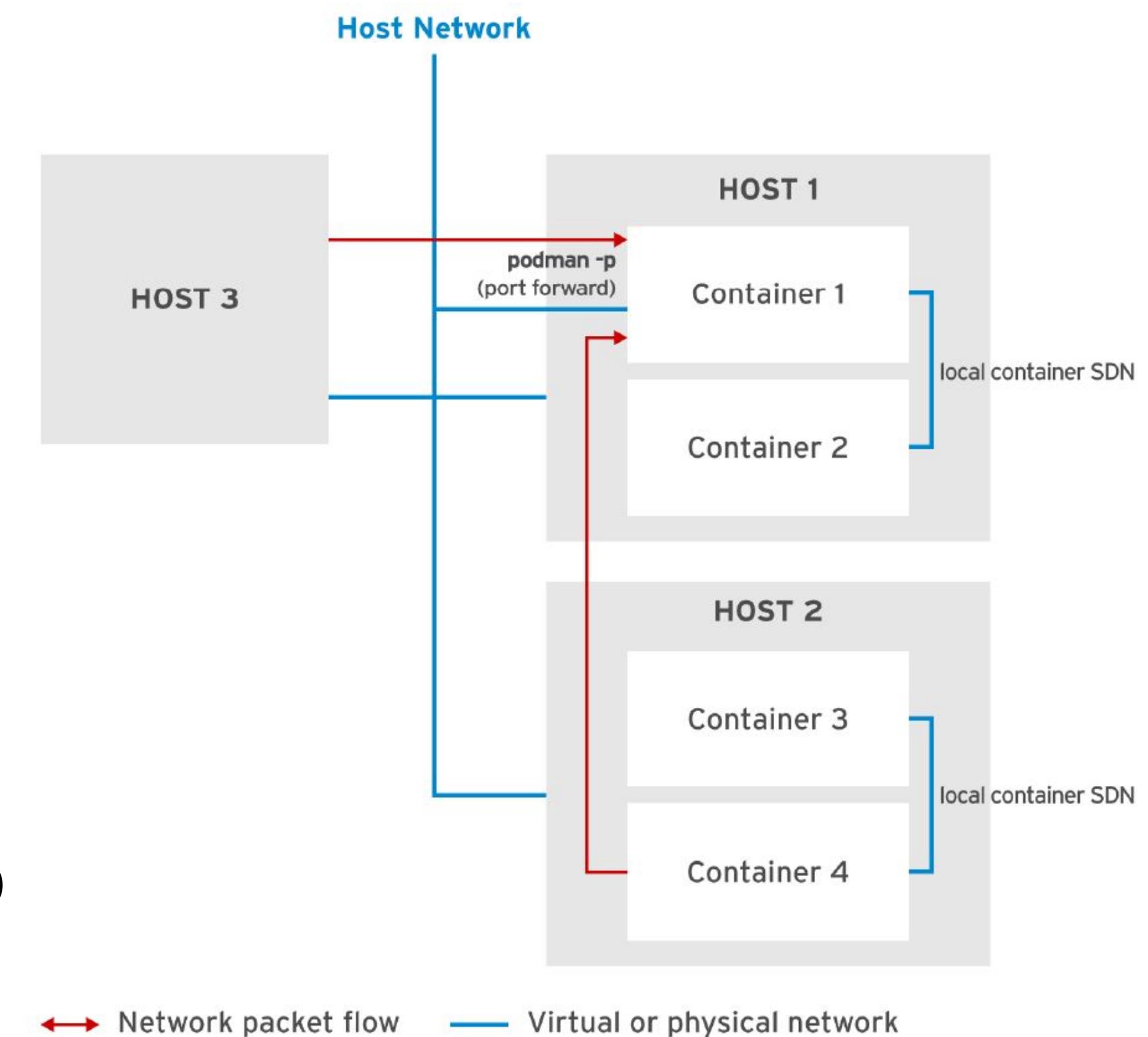
<https://github.com/containernetworking/cni>

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
```

Tecnología de Contenedores

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:

- **Working directory**

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

- **Dockerfile**

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

- **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.

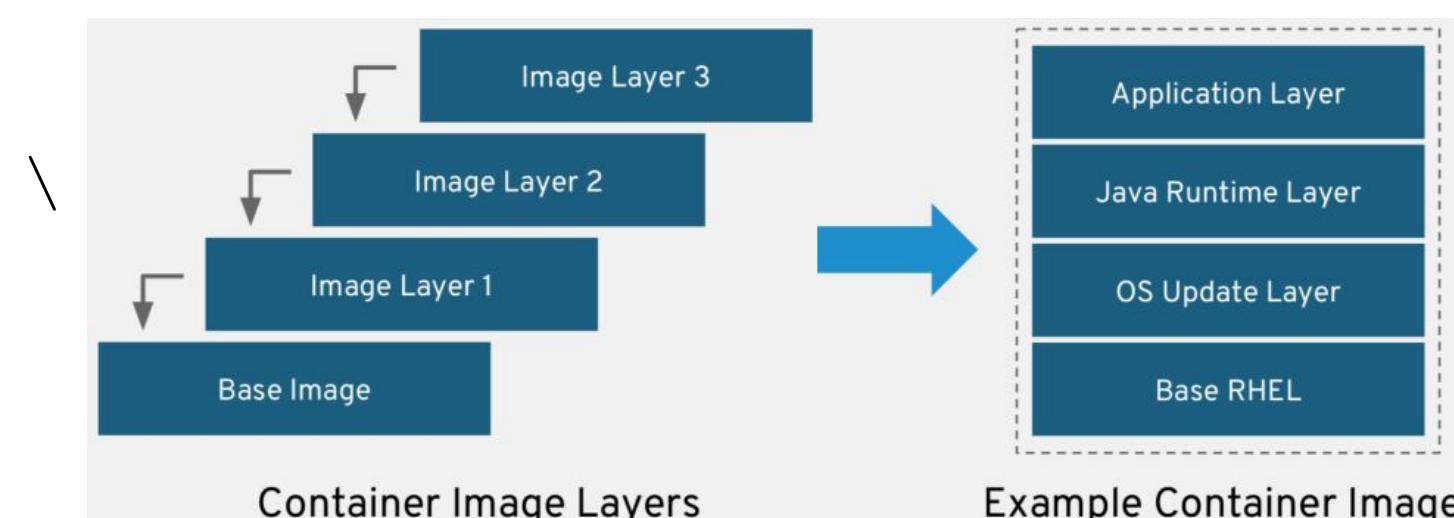
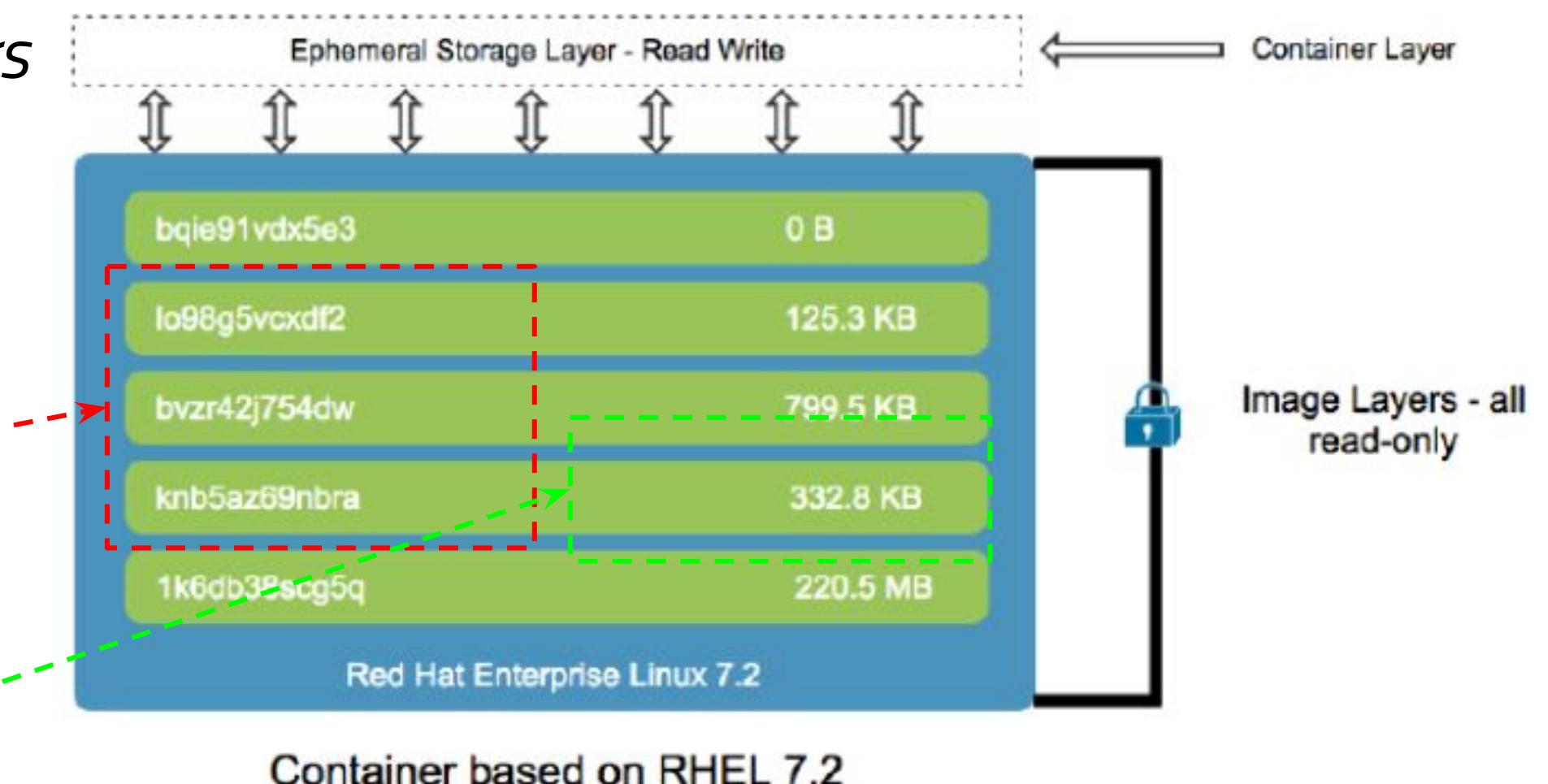
Dockerfile con muchas instrucciones este *tendrá una gran cantidad de layers en la imagen resultado.*

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  
RUN yum clean all -y
```

por una sola instrucción que genera un solo layer.

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



Tecnología de Contenedores

Dockerfiles LABEL, ENV, USER

LABEL

- La instrucción LABEL define la metadata de la imagen.
- Los labels son pares key-value agregado a la imagen.

```
LABEL version="2.0" \
      description="Este es un ejemplo de imagen" \
      creationDate="01-08-2020"
```

ENV

- Utilizada para declarar variables de entorno

```
ENV MYSQL_ROOT_PASSWORD="my_password" \
      MYSQL_DATABASE="my_database"
```

USER

- Por cuestiones de seguridad Openshift no permite correr imágenes de contenedores como root (UID=0).

```
RUN chgrp -R 0 directory && \
      chmod -R g=u directory
```

Variable	Descripción
io.openshift.tags	Describe funcionalidades categoría de imágenes. Labels separados por coma.
io.k8s.description	Usada para llevar más información sobre la imagen y servicios a quienes la consuman
io.openshift.expose-services	Información de cómo son expuestos los puertos. PORT/PROTO:NAME

DEMO

Lab 1 - Ejercicio
7 y 8

Principles and patterns

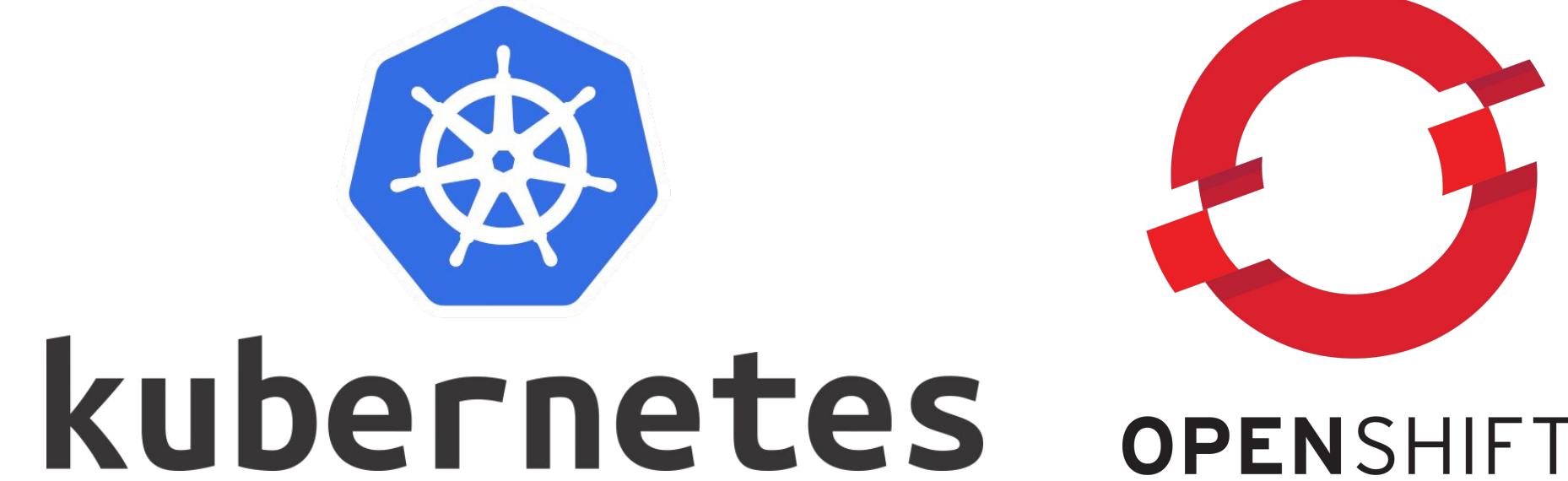
Containerized applications

Semperti

Principles and patterns

Technologies

Si bien los patrones de diseño y las metodologías de desarrollo producen aplicaciones que escalan de manera adecuada, la infraestructura y el entorno influyen en la operación sistema implementado.

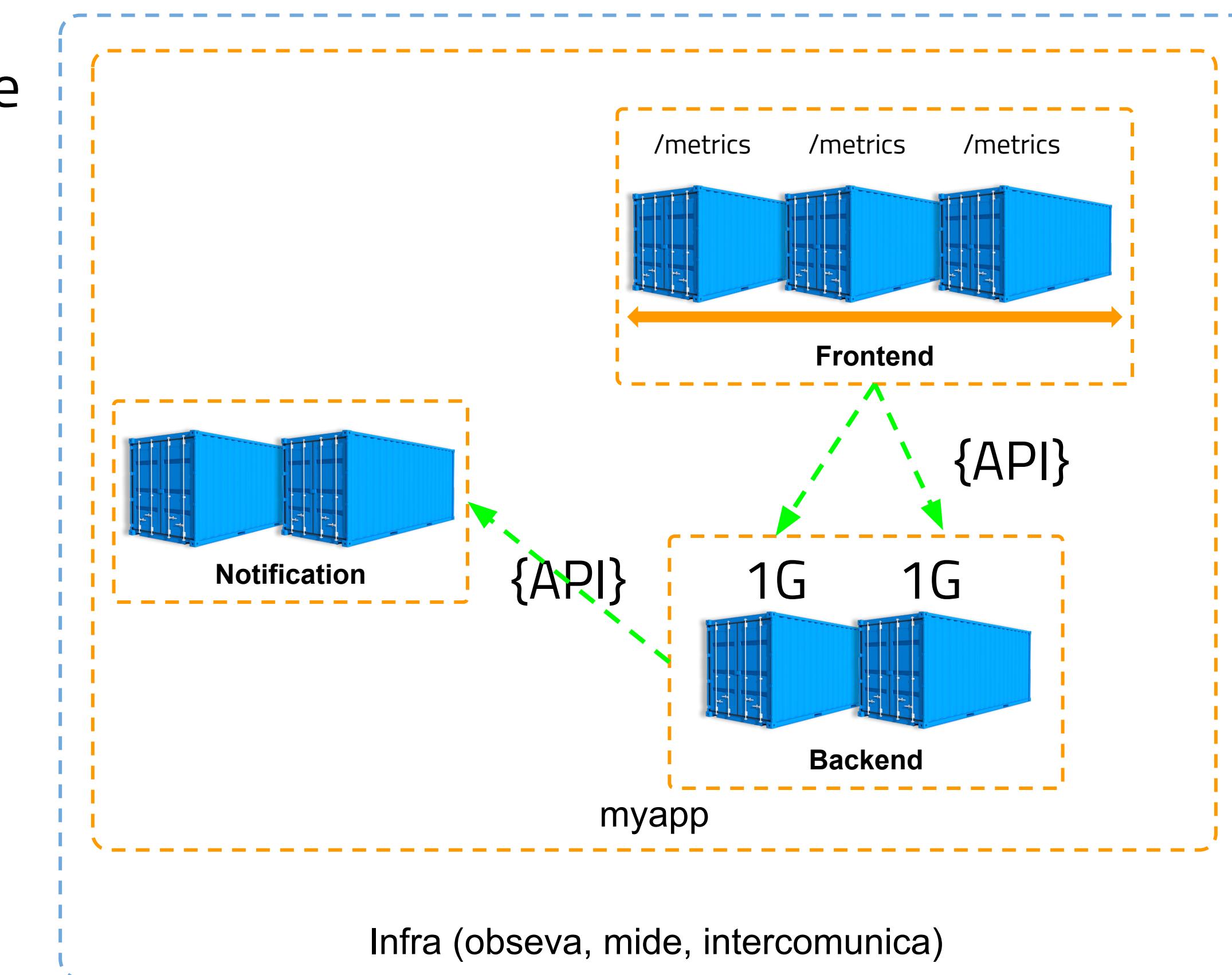


El correcto uso nos dará mayor flexibilidad, control y capacidad de respuesta

Principles and patterns

Scalability Application

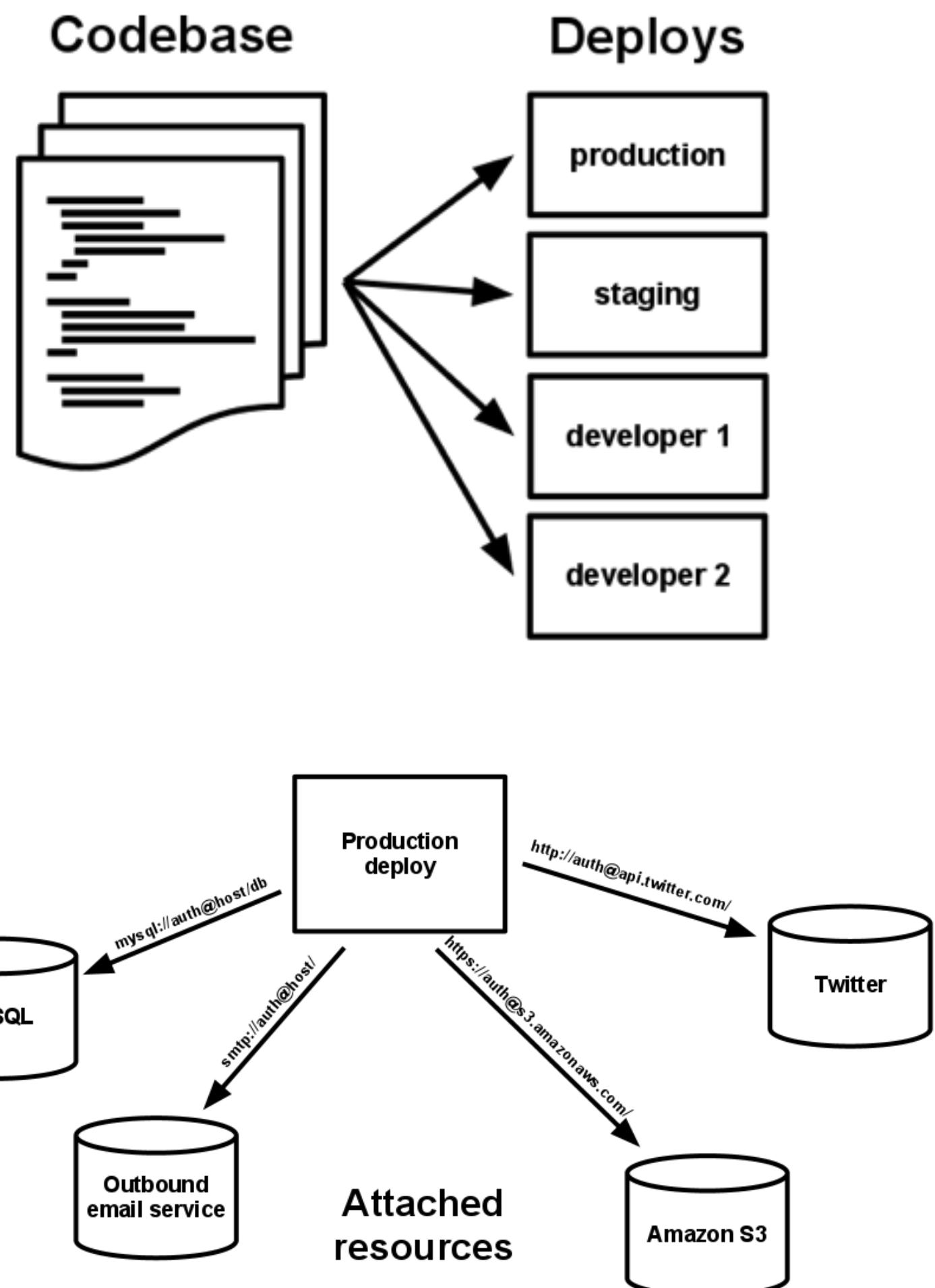
- **Escalado Horizontal:** Ajustar el número de copias, distribuir carga y aumentar disponibilidad.
- La **descomposición** del monolito en componentes discretos de un solo propósito.
- Los lineamientos de una arquitectura de **microservicios** debe adoptar características de **resiliencia, métrica, observabilidad** para poder adaptarse a entornos elásticos.



Principles and patterns

Twelve Factors

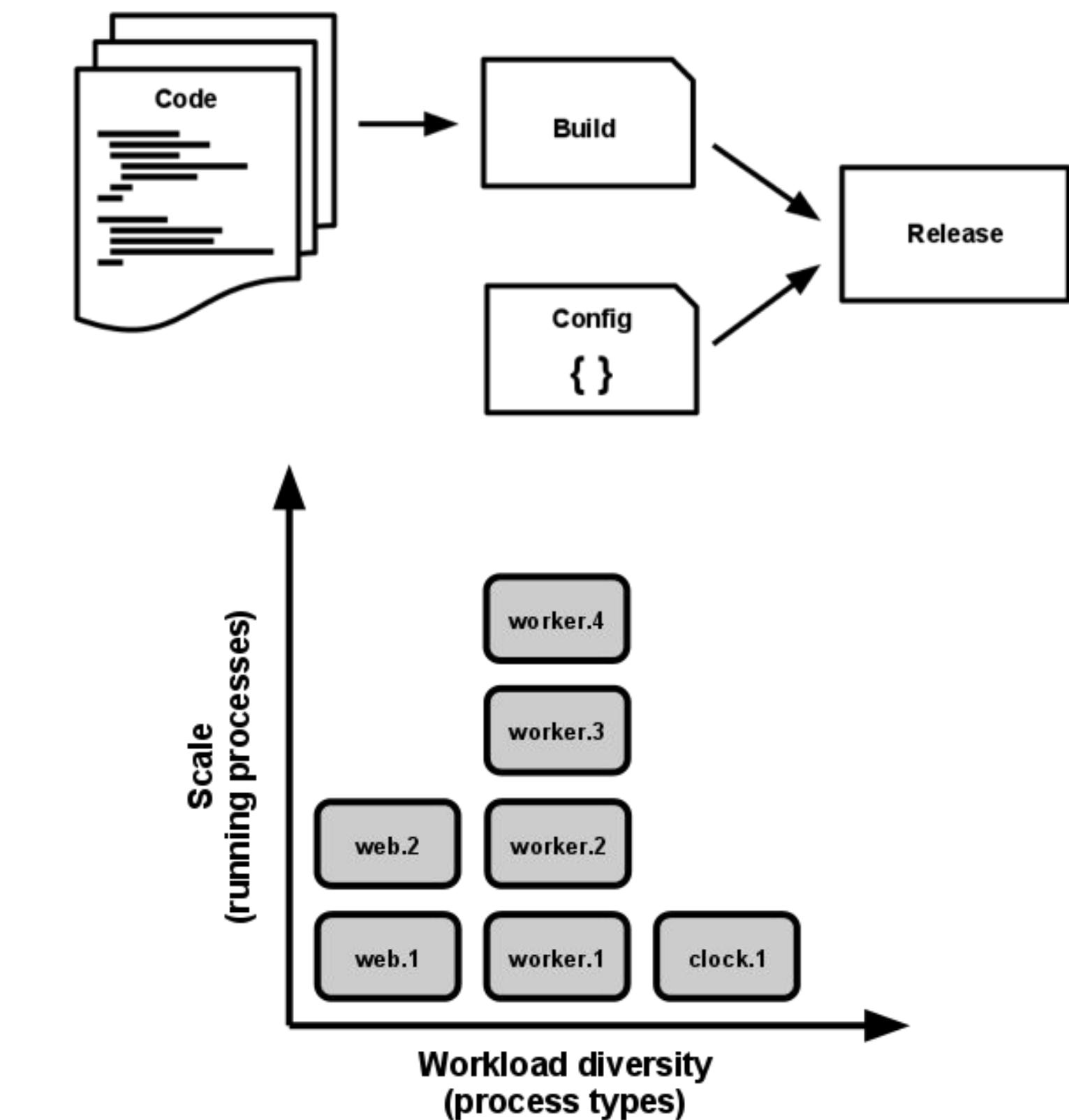
- **Codebase:** Un código base sobre el que hacer el control de versiones y múltiples despliegues. (Git, Mercurial, Subversion).
- **Dependencies:** Declarar y aislar explícitamente las dependencias (manifiesto). Python (pip, virtualenv), Ruby (Gemfile, bundle exec).
- **Config:** Separar los parámetros de configuración de la aplicación y definirla en el despliegue, no en la aplicación. No hardcoding.
Env Vars.
- **Backing services:** El consumo de recursos remotos como locales debe ser abstraído vía conexión de red. Database, email, messages queue, mem cache.



Principles and patterns

Twelve Factors

- **Build, release, run:** Build Stage debe estar separado de Release Stage y Run Stage.
 - Build stage, crea artefacto desde el código fuente;
 - Release stage combina artefacto con configuración.
 - Run Stage ejecuta el release. Rollback, id release, timestamp, version [1].
- **Processes:** Las aplicaciones son implementadas en procesos que no deben persistir estado (stateless process). Realizar offloading de estado al backend service.
- **Port binding:** Publicar servicios mediante la asignación de puertos. App autocontenido. Tornado, python. Jetty, java.
- **Concurrency:** Las aplicaciones deben escalar bajo un modelo procesos (ej, php bajo apache). Correr múltiples copias de la aplicación concurrentemente potencialmente entre distintos nodos, beneficia el escalado.



[1] <https://docs.docker.com/develop/develop-images/multistage-build/>
[2] <https://12factor.net>

Principles and patterns

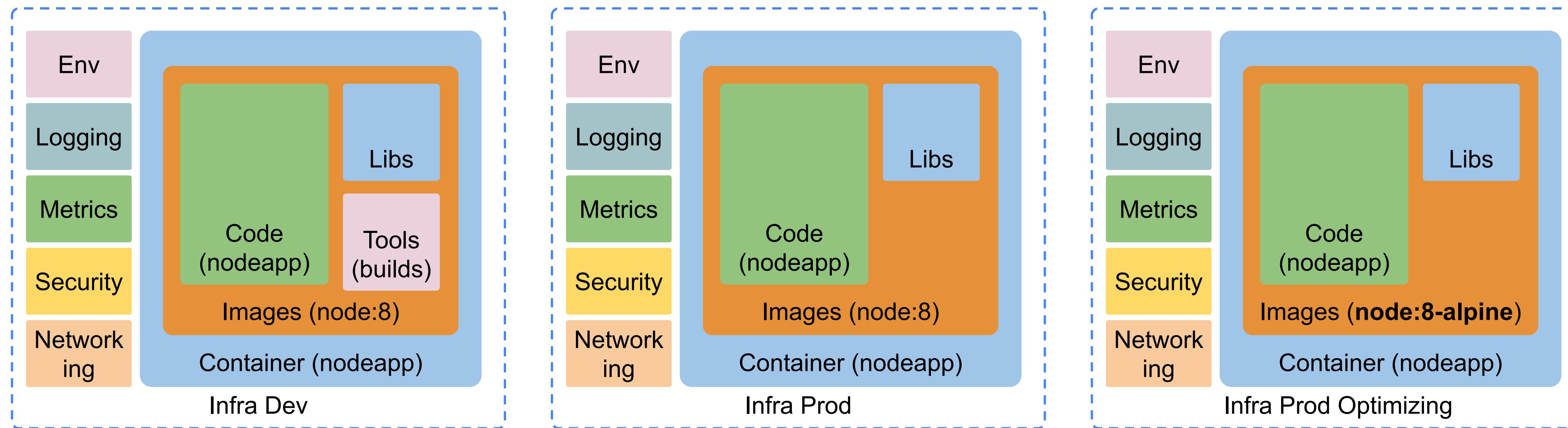
Twelve Factors

- **Disposability:** Los procesos deberían poder comenzar rápidamente y detenerse de manera segura. Procesos desechables. Implementación de colas de mensajes para retomar trabajos. Operaciones idempotentes. Preparados para la finalización de procesos inesperada o vía SIGTERM.
- **Dev/prod parity:** Mantener desarrollo, staging y prod lo más similar posible. App diseñada para CD. Tres puntos importantes.
- **Logs:** Transmitir los registros de logs a la salida estándar, recolectarlos y procesarlos por un agente externo que los administre.
- **Admin processes:** Los procesos de administración debe ejecutarse a un release y deben ser enviado con el código de aplicación.

Dev vs Prod	Traditional app	12-factor app
Time between deploys	Weeks	Hours
Code authors vs code deployers	Different people	Same people
Dev vs Prod Env	Divergent	As similar as possible

Principles and patterns

Containerized applications



Desa Env con tools de
build
[681 mb]

Prod Env sin tools de
build
[76.7 mb]

Prod Env sin tools y con
imagen reducida, minimo
footprint
[69.7 mb]

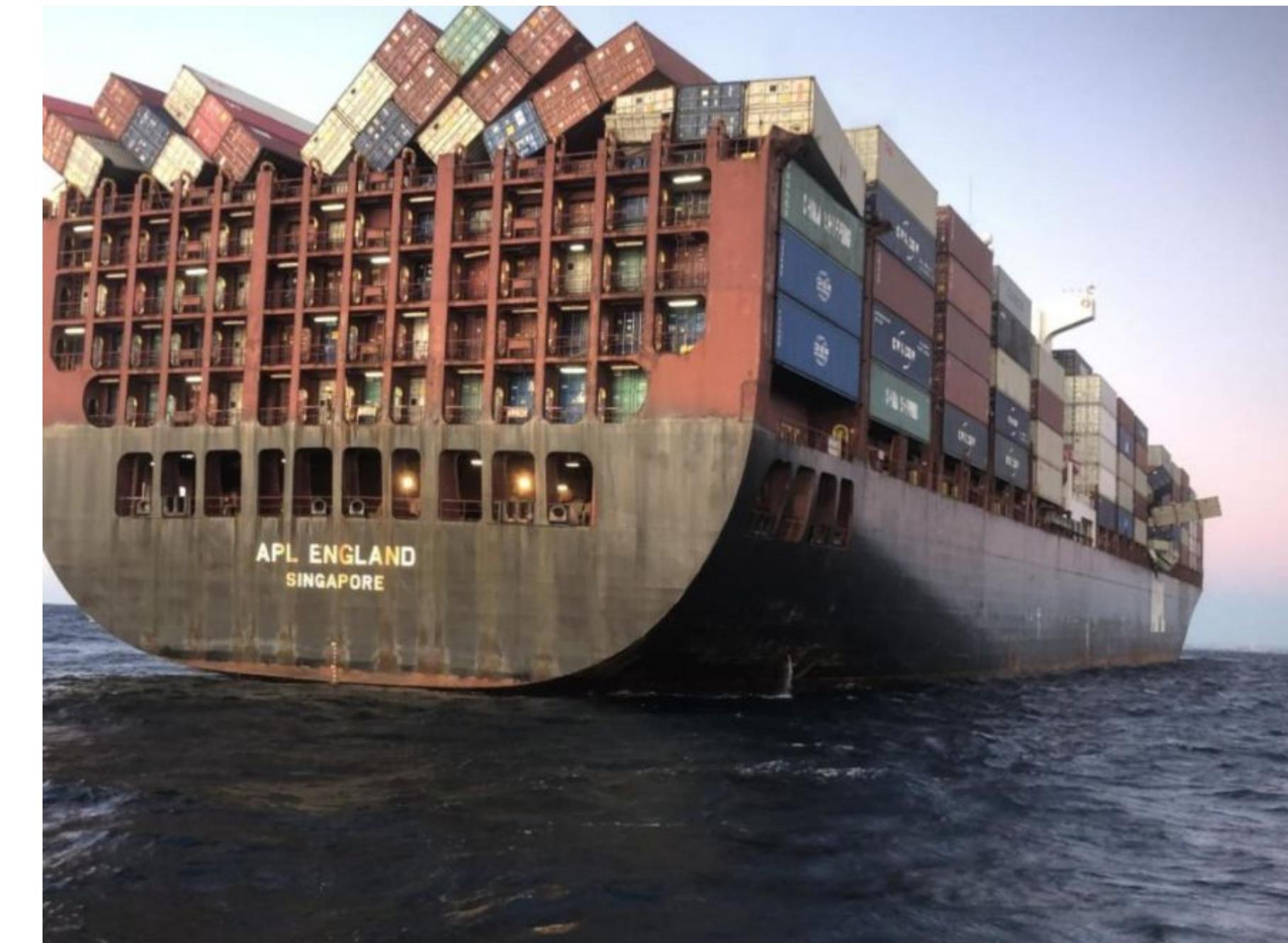
Tecnología de Contenedores

Limitaciones de los contenedores

Como el número de contenedores administrados en una organización crece, el trabajo manual comienza a complicarse de manera exponencial

En ambiente productivos empresariales requieren:

- Fácil comunicación entre un largo número de servicios.
- Limitaciones de recursos en aplicaciones.
- Responder a los picos de consumo.
- Reaccionar a la deterioridad de servicios.
- Permitir los rollout graduales para nuevos releases.



Tecnología de Contenedores

Resumen

- Arquitectura
- Ciclo de vida y administración con Podman
- Container Network Interfaces
- Almacenamiento persistente
- Imágenes y registro de imágenes
- Dockerfiles
- Limitaciones



Día 2

Semperti

Container Orchestration

Resumen

- Kubernetes
- Openshift
- Arquitecturas y conceptos core
- Network workflow
- Deployment workflow



Container Orchestration

Kubernetes (CaaS) y Openshift (PaaS)

Semperti

Container Orchestration (CaaS)

Overview Kubernetes

Kubernetes es un servicio de orquestación que simplifica el despliegue, administración y escalado de aplicaciones en contenedores.

- La unidad más pequeña de administración en Kubernetes es el pod.
- Un pod consiste en uno o más contenedores con recursos de almacenamiento e ip que comparte una misma aplicación.
- Kubernetes también usa pods para orquestar contenedores de aplicación



kubernetes

Container Orchestrator

Workload Placement

Infrastructure Abstraction

Desired State

<https://kubernetes.io/>

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

Kubernetes = Proyecto Open Sources

Container Orchestration (CaaS)

Características Kubernetes

Características de Kubernetes

- Service Discovery and Load Balancing
- Horizontal Scaling
- Self-healing
- Automate rollout
- Secrets and configuration manager
- Operators

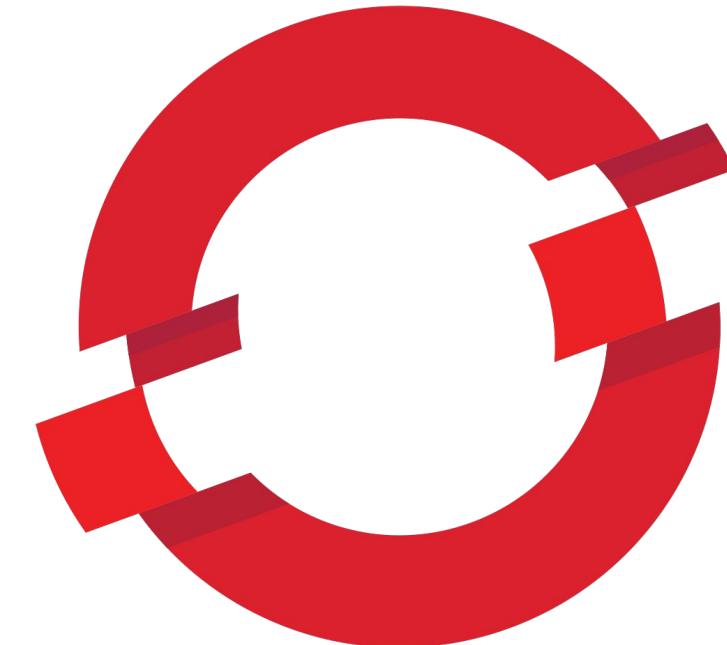


Container Orchestration (PaaS)

Overview Openshift

Red Hat Openshift Container Platform (RHOCP) es un set de componentes modulares y servicios de construcción de aplicaciones que trabaja sobre la infraestructura de Kubernetes.

Openshift agrega características de manera que una solución de Contenedores como Servicio (CaaS) evoluciona a un producto de Plataforma como Servicio (PaaS)



Openshift = Producto de Red Hat con base en
tecnología Open Sources

Container Orchestration (PaaS)

Características Openshift

Características de Openshift

- Integración con workflow de desarrollo
- Rutas para exposición de servicios.
- Metricas y Logging
- Interfaz UI (Self Service)
- Build Tools S2I, Operators
- Soporta Java, Python, Ruby, Node.js, Perl, PHP, .NET y mas.
- RHCOS is FIPS-Compliance
- Multitenancy
- Red Hat CoreOS y Red Hat Enterprise Linux
- Seguridad, privacidad, compliance y requerimientos gubernamentales.
- Runtime and xPaaS

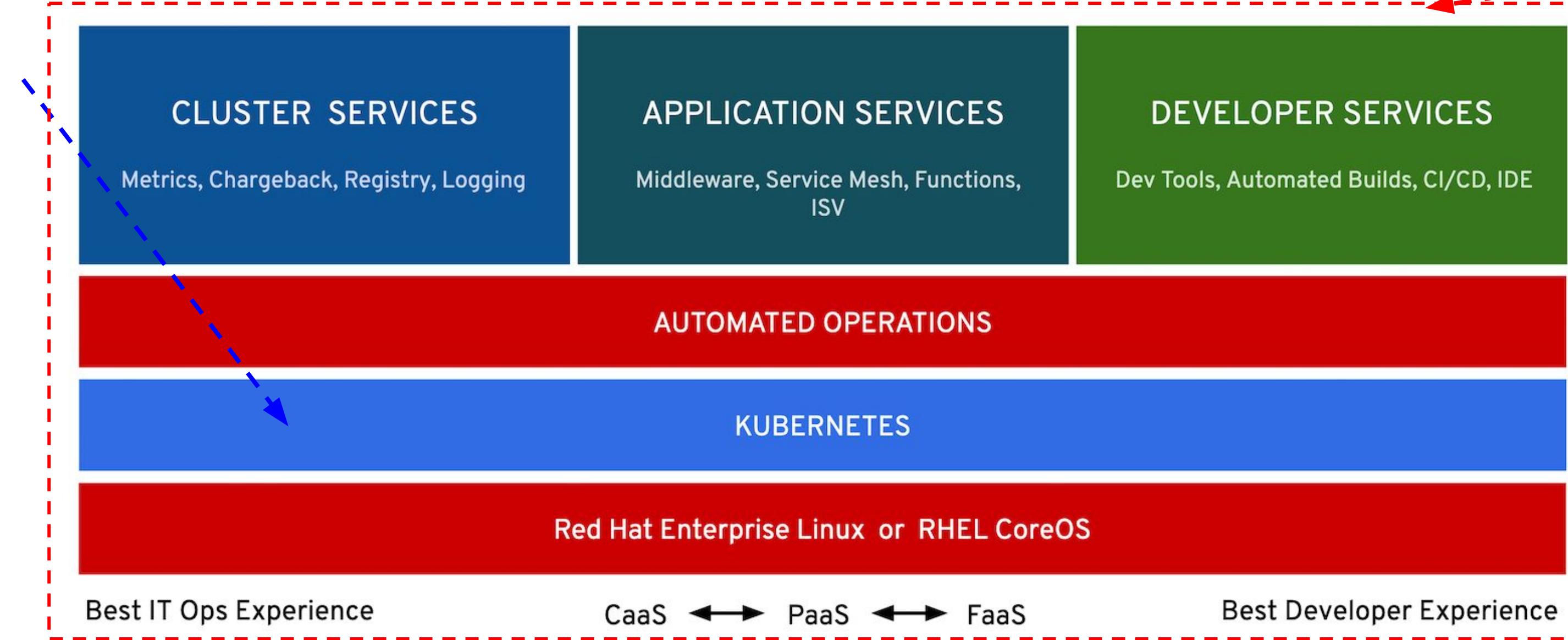


OpenShift Container Platform (PaaS)

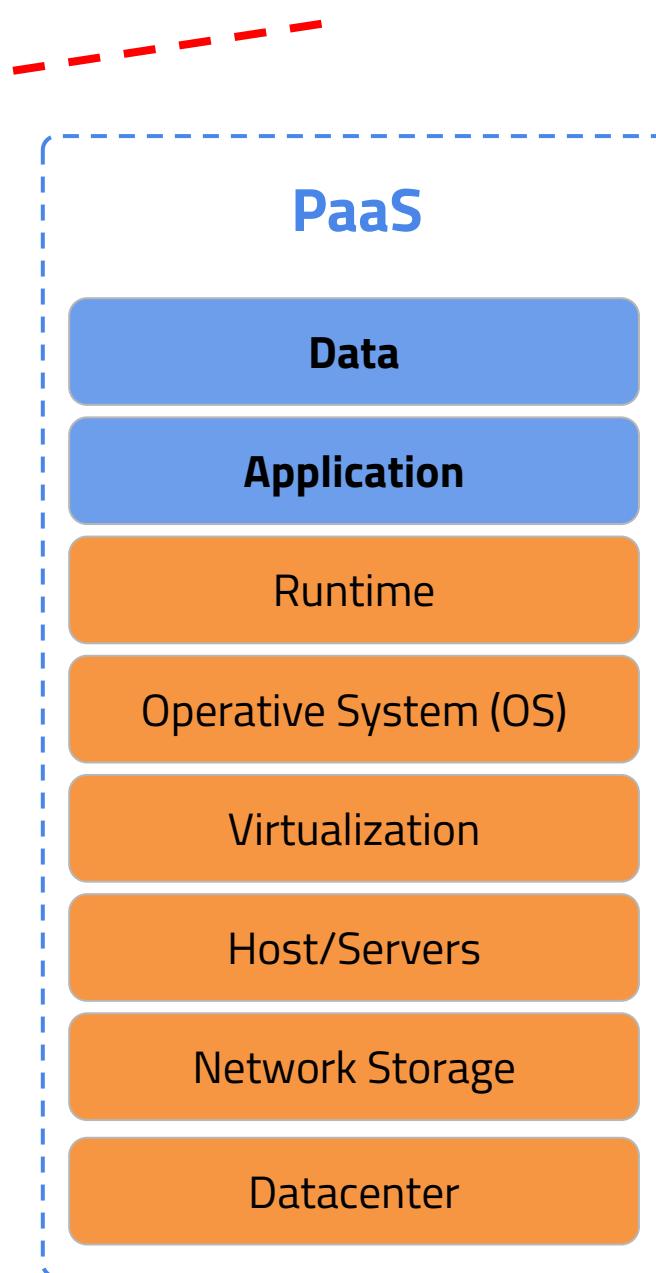
Roles Arquitectura

Kubernetes

CaaS



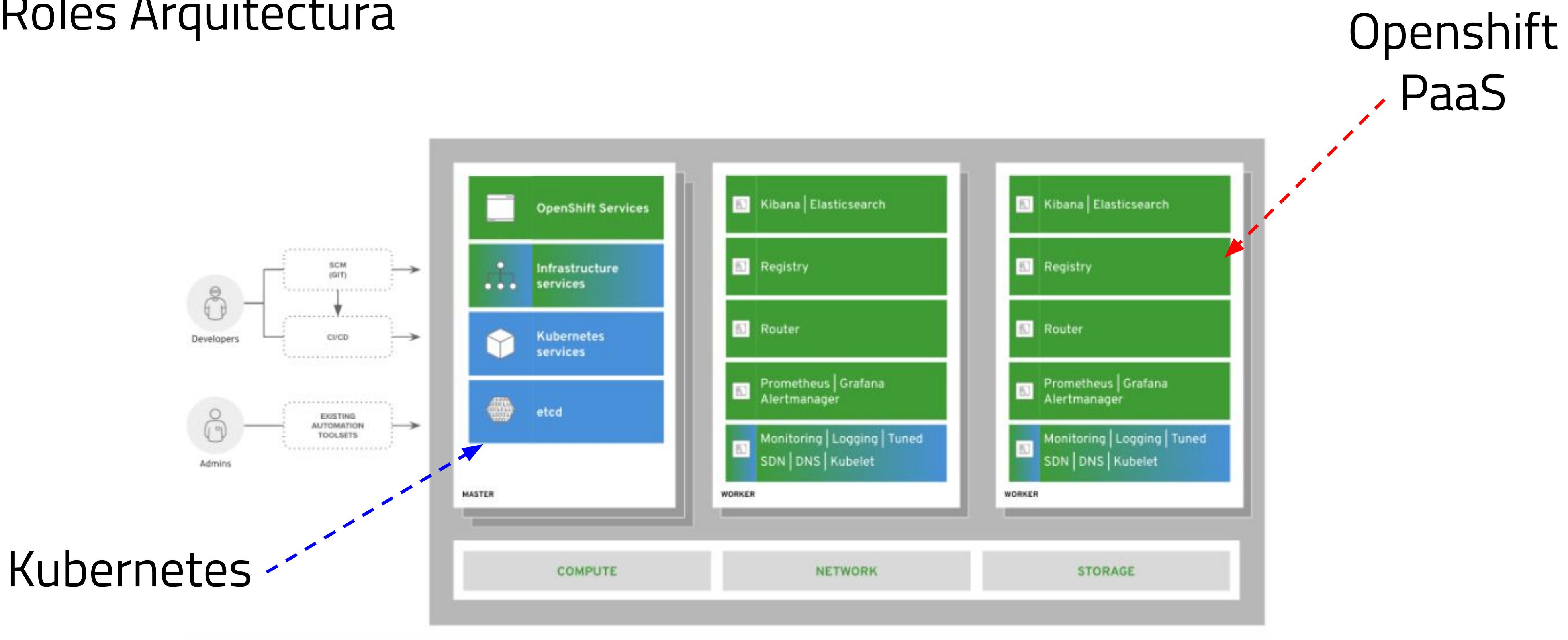
OpenShift
CaaS-PaaS-FaaS



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 Core Concepts

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 Core Concepts

Kubernetes Principios

Principios de Kubernetes

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

Estos conceptos de diseño son heredados por OpenShift

OpenShift Core Concepts

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

Full Stack Automation (IPI)



Pre-existing Infrastructure (UPI)



* Note: Planned for an upcoming 4.3.z release on April 30th

□ Denotes new addition in OCP 4.4

OpenShift Core Concepts

Nodes

- OpenShift corre sobre RHCOS y RHEL
- OpenShift posee dos tipos de nodos:
 - Worker
 - Master.
- Nodos son instancias de RHCOS o RHEL son creados al momento de la instalación.
 - Workers: Donde son alojadas las aplicaciones de negocio.
 - Masters: Administra los nodos del clusters y otros componentes de software que corren sobre todos los nodos.



Command Line

```
$ oc get nodes -o wide
```

Openshift Core Concepts

Nodes Host

Nodos Host

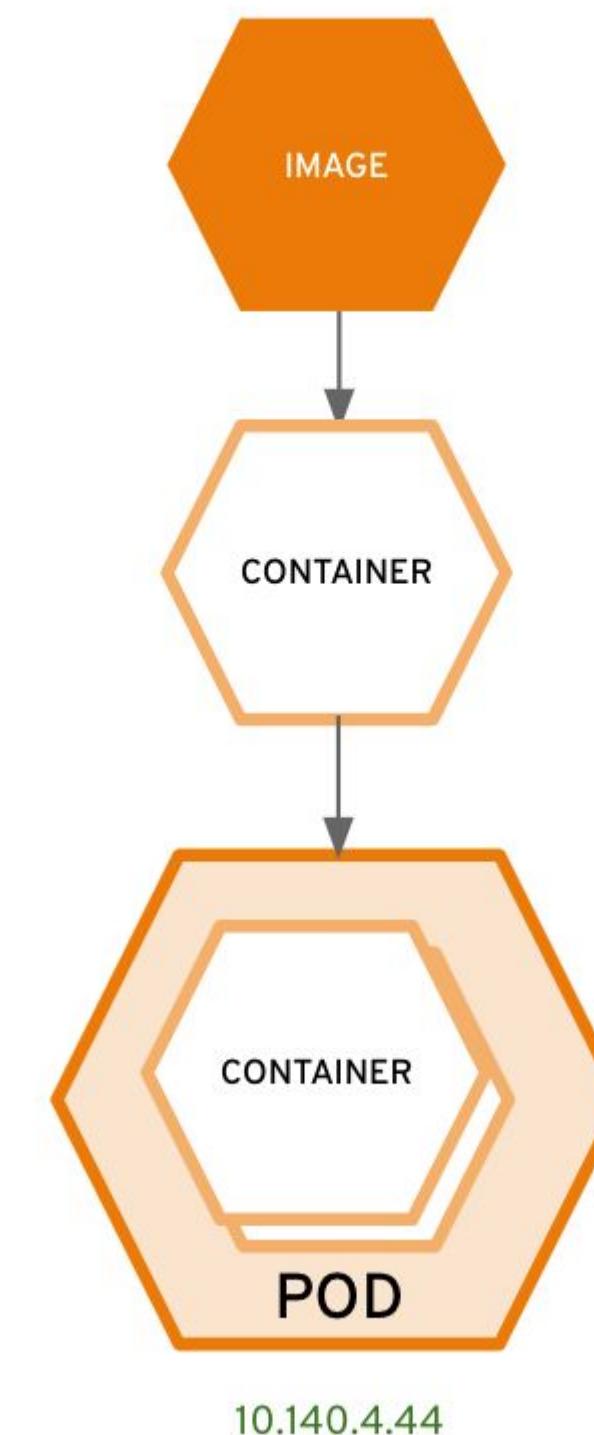
- Las instancias de aplicaciones y componentes corren en contenedores OCI-Compliant
- Imagenes: son las aplicaciones, como el binario.
- Contenedores: runtime sobre el que corre la imagen.
- Openshift Worker node puede correr una gran cantidad de contenedores.
- La capacidad del nodo está relacionada con la cantidad de Memoria y CPU que tenga el nodo.
- Cloud, hardware físico o virtualizado.



Openshift Core Concepts

Contenedores

- Uno o más contenedores puede ser desplegados juntos en un host.
- Consiste en un grupo de contenedores con recursos compartidos como volúmenes y direcciones IP.
- Unidad de computo mas pequeña para despegar y administrar.
- Pueden contener una o más aplicaciones acopladas ejecutadas en un contexto compartido.
 - Sidecar container junto con el pod de negocio.
- En Openshift, los pods reemplazan a los contenedores de aplicaciones individuales como unidad mínima a desplegar.



Command Line

```
$ oc explain pods  
$ oc get pods -o yaml  
$ oc describe pod PODNAME
```

Openshift Core Concepts

PODs

- Unidad mínima de orquestación en Openshift
- Openshift programa y ejecuta todos los contenedores en pods sobre nodos.
- Complejas aplicaciones pueden ser desplegadas en múltiples pods, cada uno de estos con sus contenedores.
- Interactúa con agentes externos como también internos
- Openshift ejecuta imágenes de contenedores en un "container wrapper" que agrega metadata llamado **POD**
- Es posible correr múltiples contenedores en un mismo pod.
- Aplicaciones diferentes, como base de datos y aplicaciones no comparten el mismo pod.
- Permiten individualizar componentes de aplicaciones para que sea fácil escalar horizontalmente.
- Las aplicaciones son conectadas entre ellas por medio de *services*



Command Line

```
$ oc get pods -o wide  
$ oc describe pod PODNAME
```

OpenShift Core Concepts

POD

- Son aplicaciones o instancias de algo.
- Usamos oc get pods para ver los pods corriendo en un ambiente, usualmente un proyecto.
- Los **Proyectos (projects)** tambien son llamados **namespaces**.
- Algunos pods tiene le objetivo de realizar tareas de deploy y de build.
- Los pods de aplicacion son aquellos que quedan en estado **running**

```
$ oc get pods -n myproject
```

NAME	READY	STATUS	RESTARTS	AGE
cakephp-mysql-example-1-build	0/1	Completed	0	2d19h
cakephp-mysql-example-1-deploy	0/1	Completed	0	2d19h
cakephp-mysql-example-1-hook-pre	0/1	Completed	0	2d19h
cakephp-mysql-example-1-tgs4q	1/1	Running	0	2d19h
mysql-1-2qvhn	1/1	Running	0	2d19h
mysql-1-deploy	0/1	Completed	0	2d19h

```
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
```

Deployment

Scaling, updates and rollback

POD

Small Unit of Deployment in Kubernetes

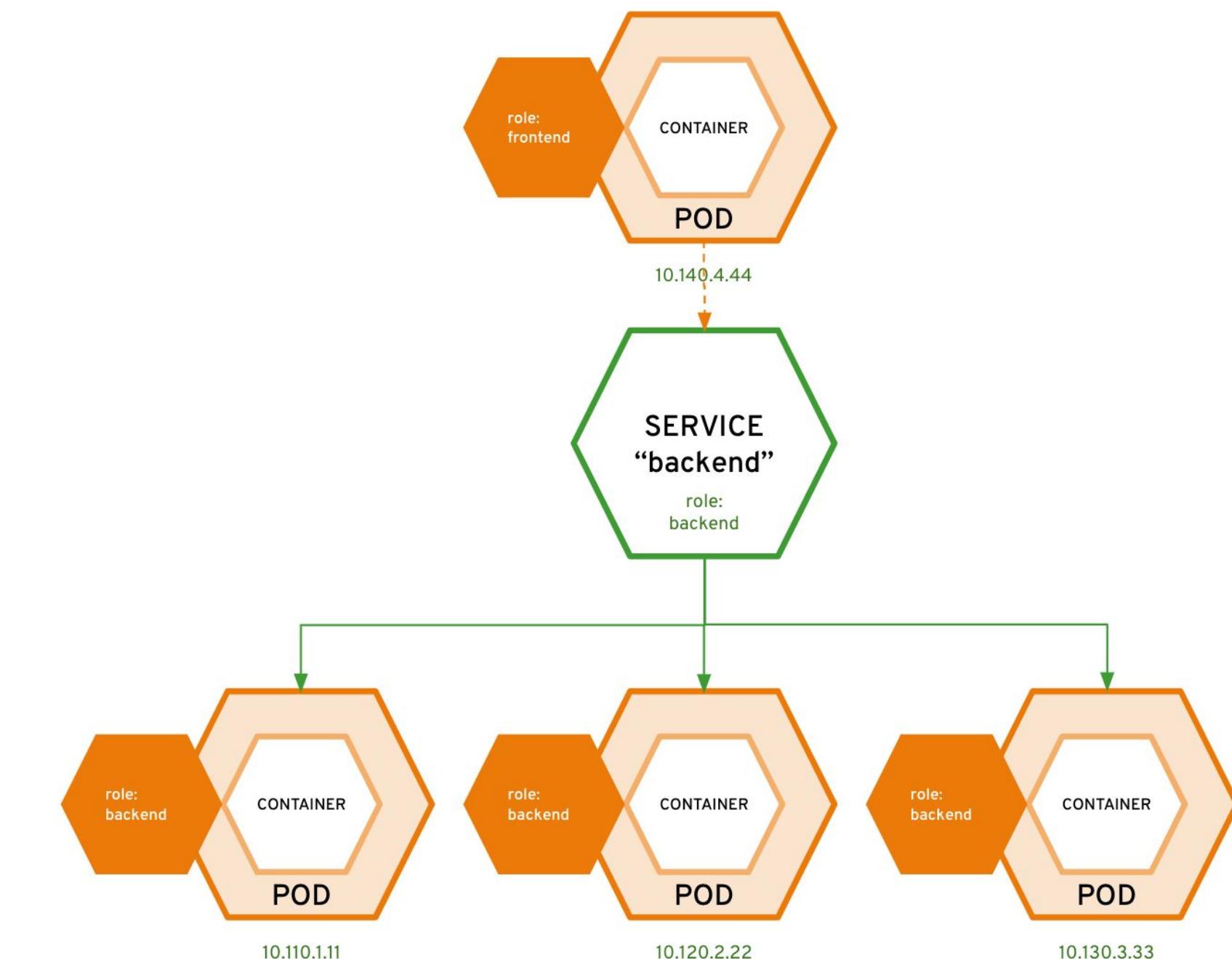
Container

Application Code

OpenShift Core Concepts

Services

- Definen de manera lógica un set de pods y la política de acceso.
- Provee una única dirección ip y nombre de dns para acceder para que otras aplicaciones accedan a un conjunto de pods que ofrecen un mismo servicio.
- La capa de servicio brinda la conexión entre aplicaciones.
 - Ejemplo, el frontend consume el backend por medio del servicio (imagen).
- Los servicios permiten un balanceo interno a través de componentes aplicativos (reglas de iptables replicadas entre nodos)
- OpenShift inyecta información de servicio dentro de los contenedores que son ejecutados de manera que sea fácil poder descubrirlos (variables de entornos con datos de los servicios)



Command Line

```
$ oc get services  
$ oc explain service
```

OpenShift Core Concepts

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.

Command line

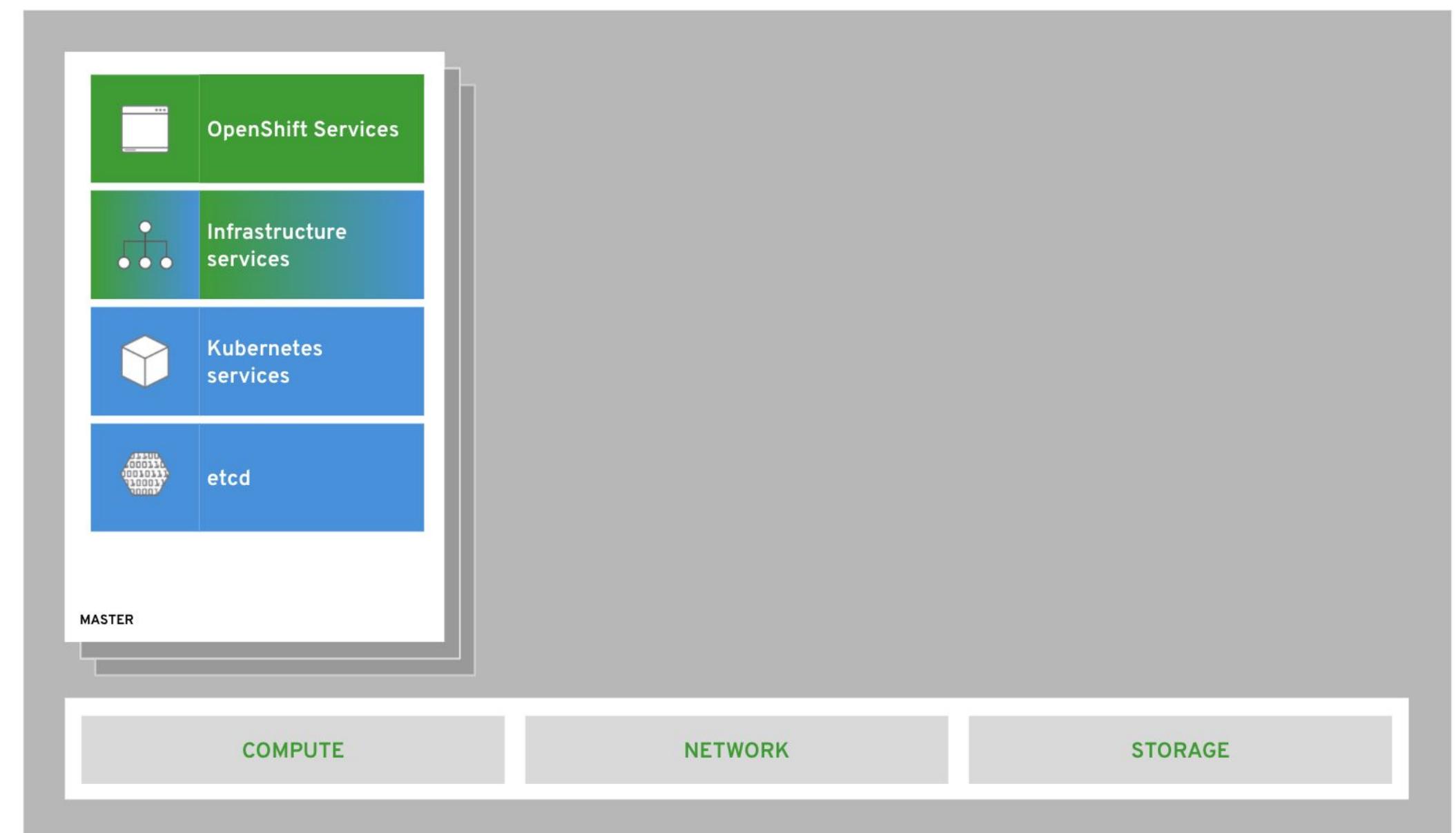
```
$ oc explain labels  
$ oc get RESOURCENAME --show-labels
```

```
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:  
        [...]
```

OpenShift Core Concepts

Master Nodes

- Instancias de RHCOS
- Función primaria
- Orquestar la actividad de los nodos
- Conoce y mantiene el **estado deseado** en el ambiente de OpenShift.
- Tres master por alta disponibilidad.
 - Azul = Kubernetes
 - Verde = OpenShift
- Algunos componentes de infraestructura son únicos para OpenShift



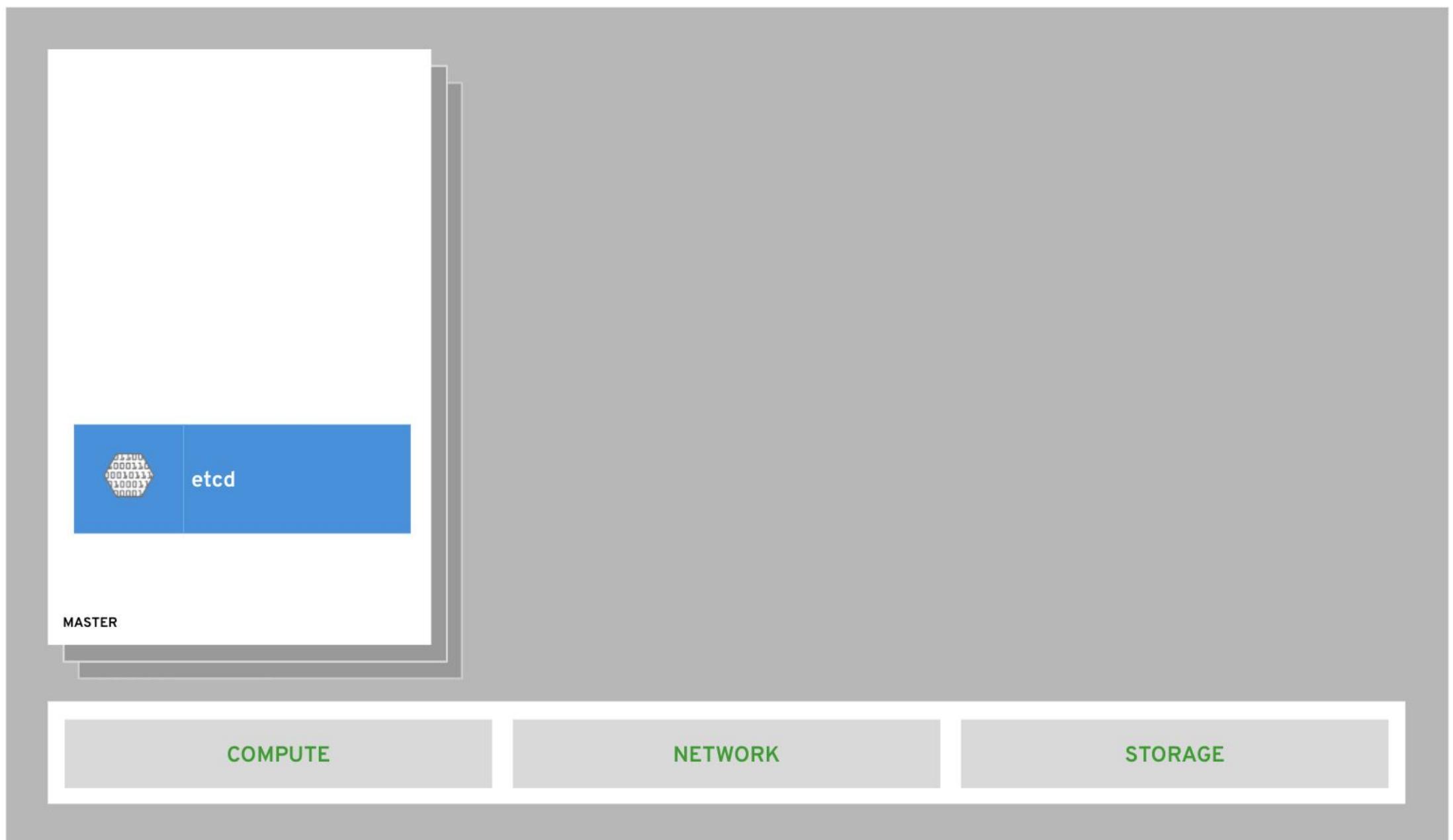
OpenShift Core Concepts

Etcd

- El estado deseado como el actual es almacenado en una base de datos distribuida clave-valor llamada etcd.
- Etcd también almacena información:
 - Reglas RBAC
 - Información de ambiente de la aplicación
 - Información de usuario que no es de aplicación

Command line

```
$ oc get pods -n openshift-etcd  
$ oc get crd etcds.operator.openshift.io
```



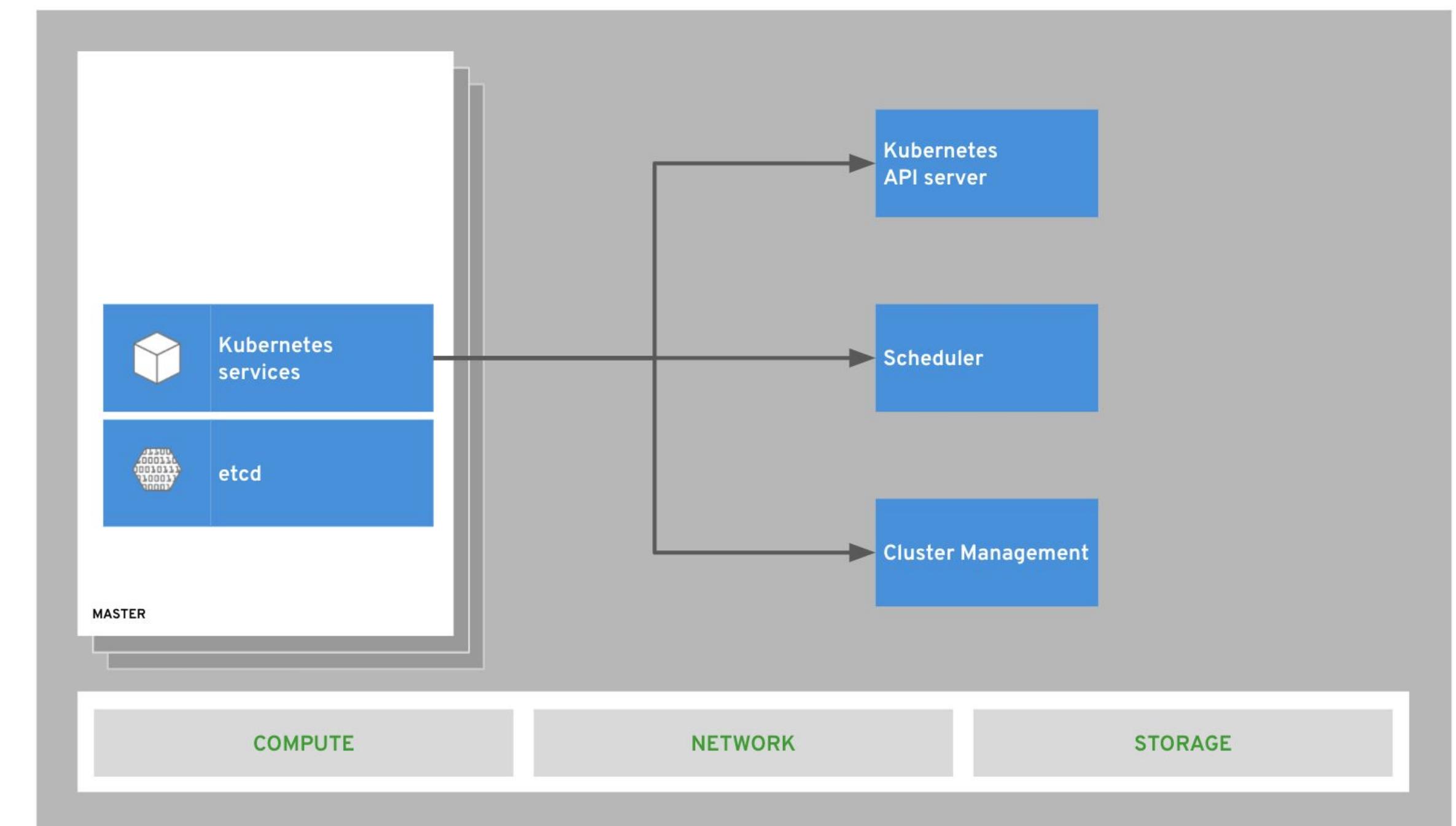
OpenShift Core Concepts

Master Service - Kubernetes core componentes

- OpenShift incluye servicios de Kubernetes.
- Kubernetes no es alterado, byte-bytes es idéntico a la provista por el proyecto de la Cloud Native Computing Foundation.
- Provee:
 - Kubernetes API Server
 - Scheduler
 - Cluster Management Services

Command line

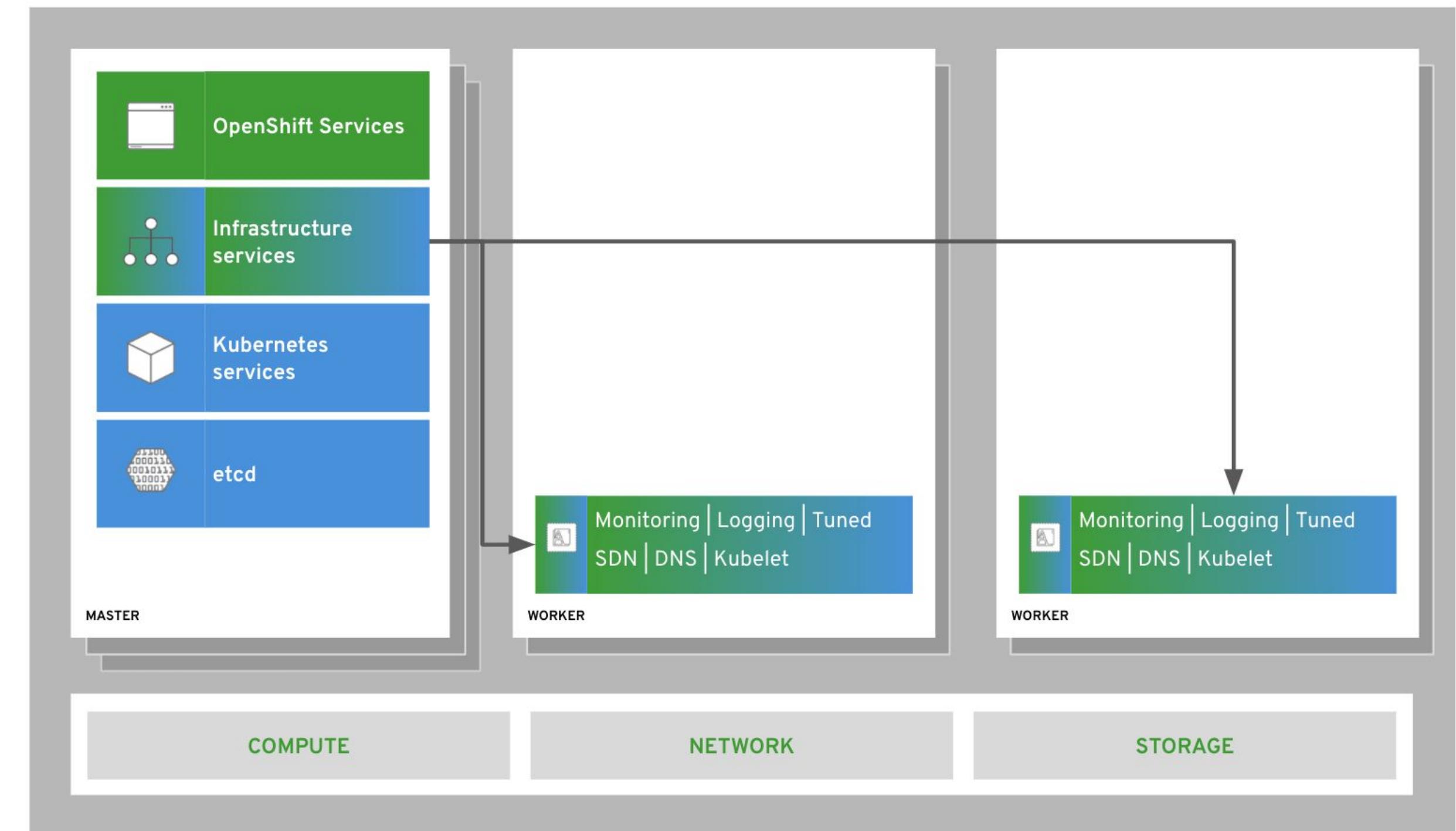
```
$ oc get nodes -l  
node-role.kubernetes.io/master  
$ oc get crd | egrep  
"apiserver|scheduler|controller"
```



OpenShift Core Concepts

Master - Infrastructure services

- Nodos master y workers trabajar en colaboración.
- Servicios de OpenShift y Kubernetes están incluidos.
- Services:
 - Monitoring
 - Logging
 - OS tuning
 - Software Defined Networking (SDN)
 - DNS
 - Kubelet (OpenShift node process)



Command Line

```
$ oc get nodes -l node-role.kubernetes.io/infra  
$ oc get nodes -l node-role.kubernetes.io/worker
```

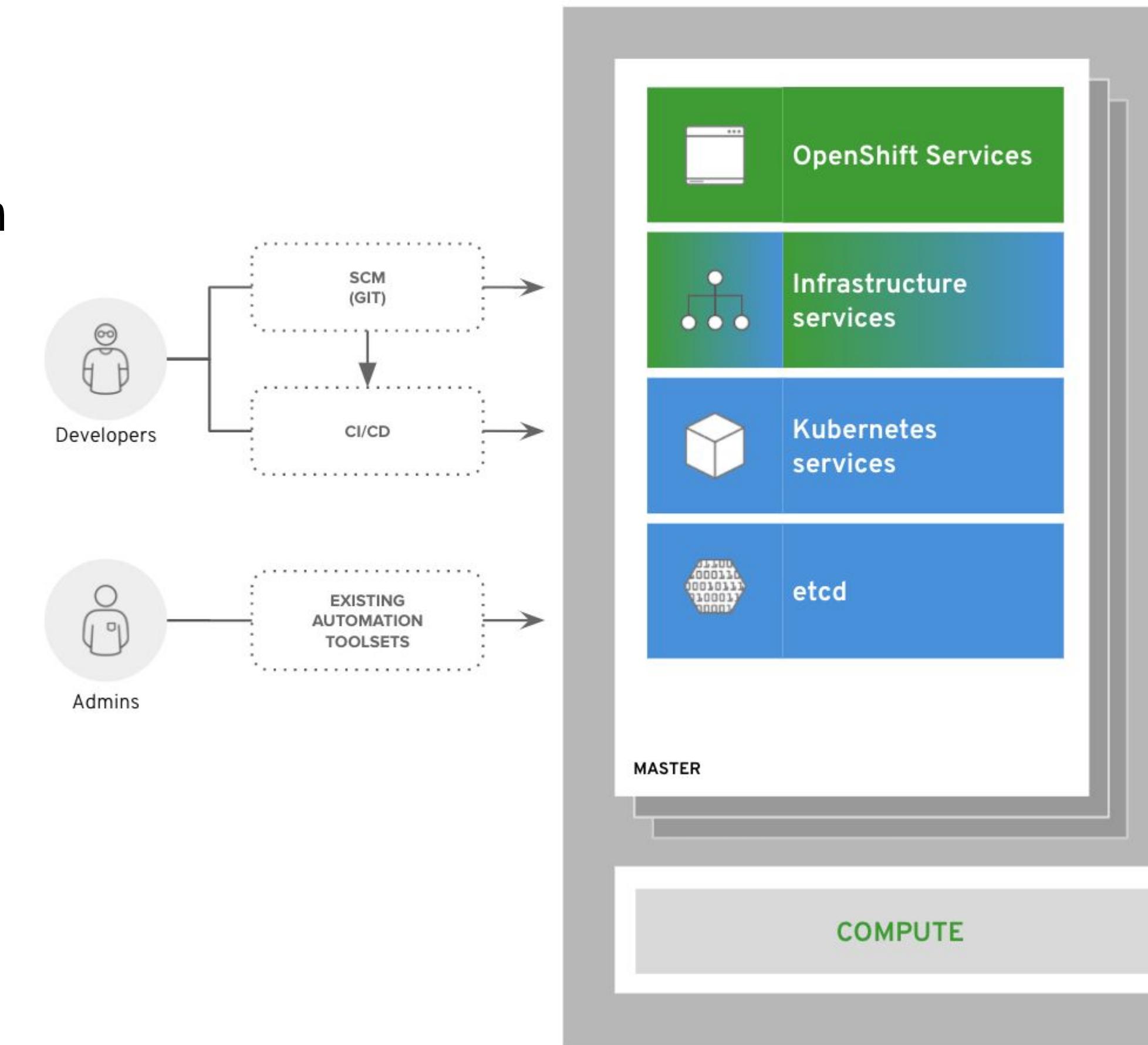
OpenShift Core Concepts

Master - API and Authentication

- Los nodos master poseen un solo API Endpoint donde todos los componentes del sistema interactúan con él.
- Incluido OpenShift y Kubernetes services.
- Todos requerimiento administrativo (OC CLI) es realizado via esta API.
- Todos los API request son encriptados SSL y *autenticados*.
- La *autorización* granular a los recurso está dada por un sistema de control de acceso basado en roles (RBAC).
- Masters, son capaces de vincularse a sistemas externos de gestión de identidad.
 - LDAP, Active Directory, OAuth proviste por Google, Github, etc.

Command Line

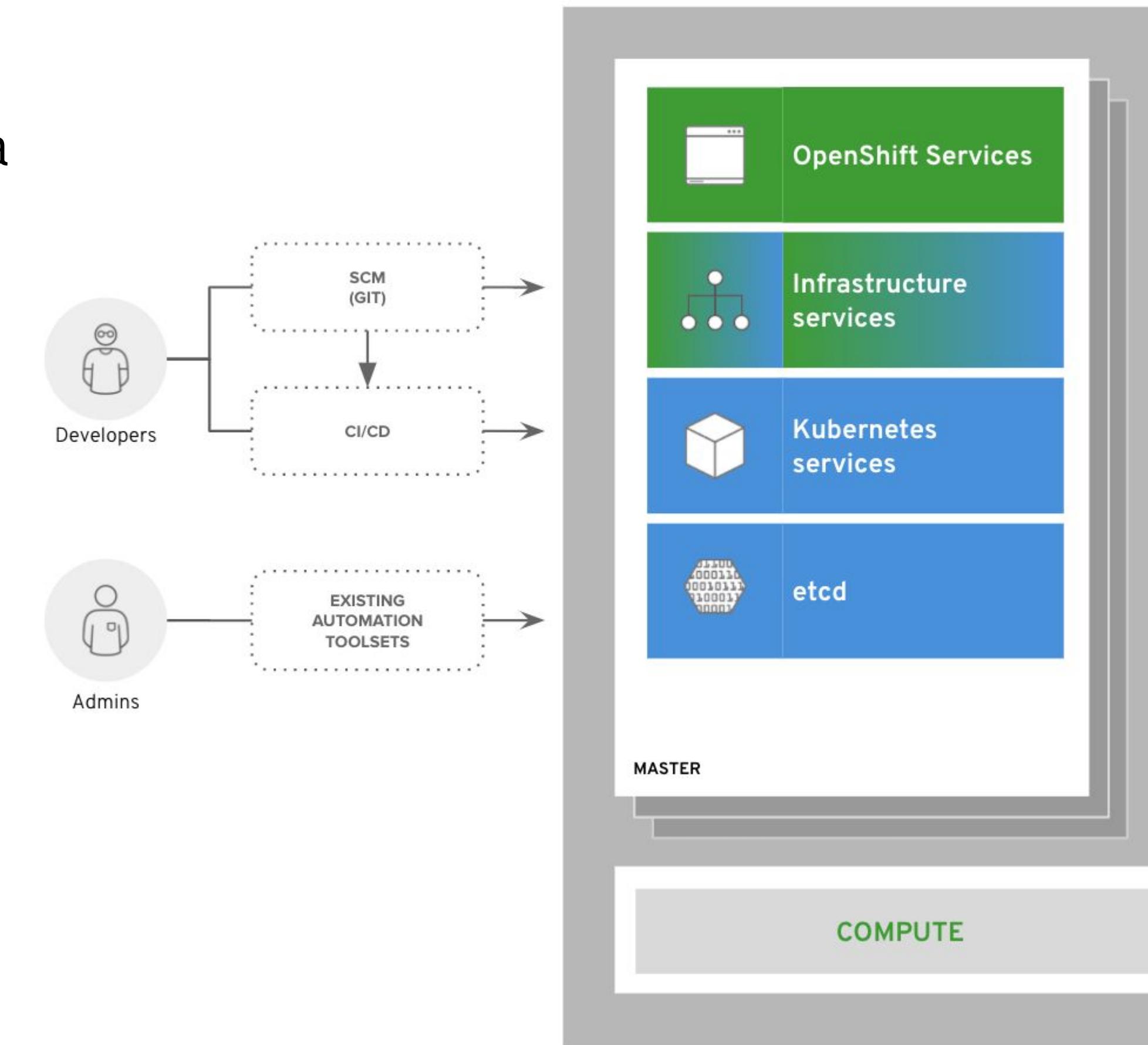
```
$ oc get pods -n openshift-apiserver
```



OpenShift Core Concepts

Access

- 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



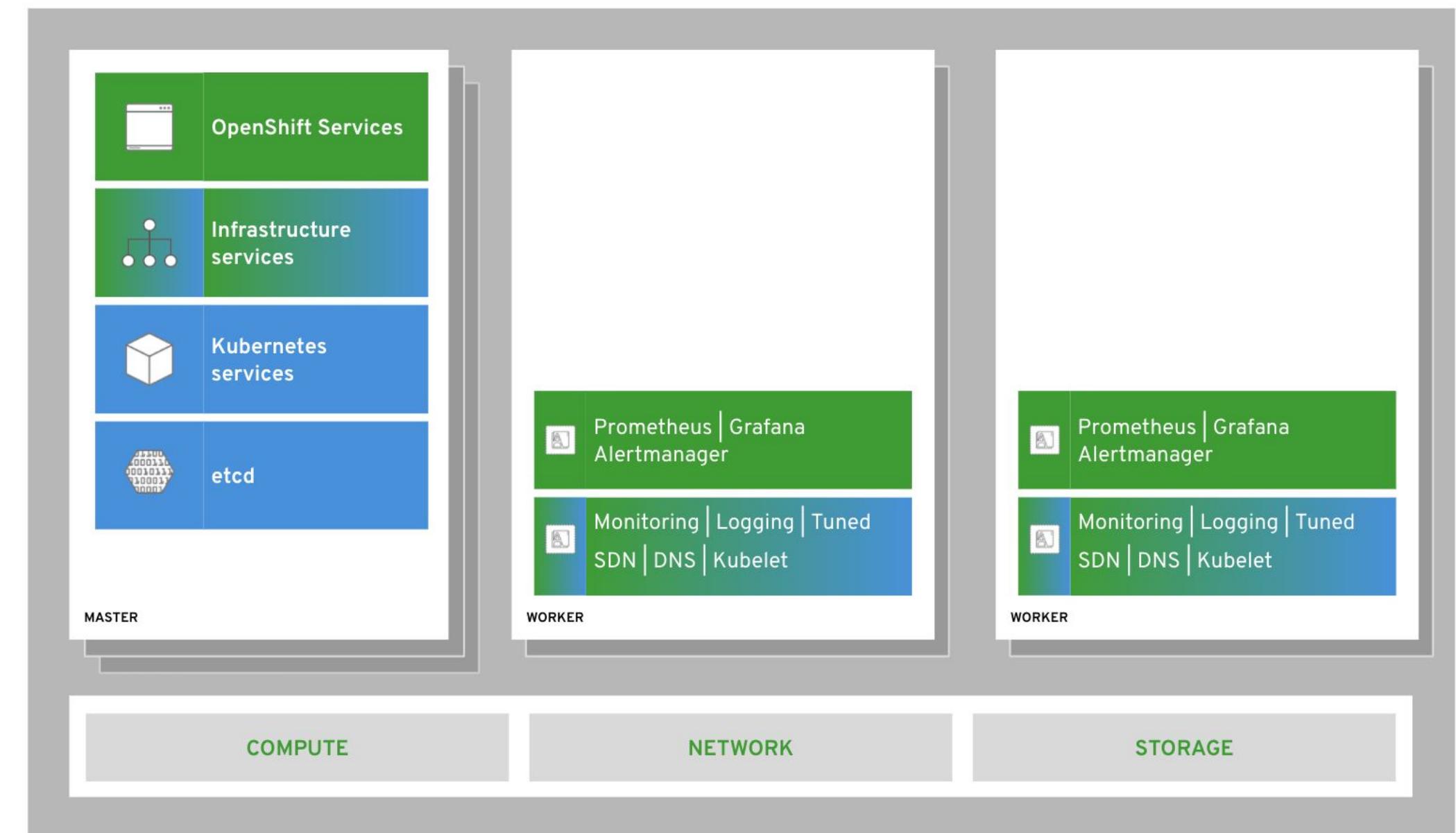
OpenShift Core Concepts

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.



Command line

```
$ oc set probe dc/<deploy_name> --readiness --get-url <http://> --initial-delay-seconds=<time_in_sec>
$ oc set probe dc/<deploy_name> --liveness --get-url <http://> --initial-delay-seconds=<time_in_sec>
```

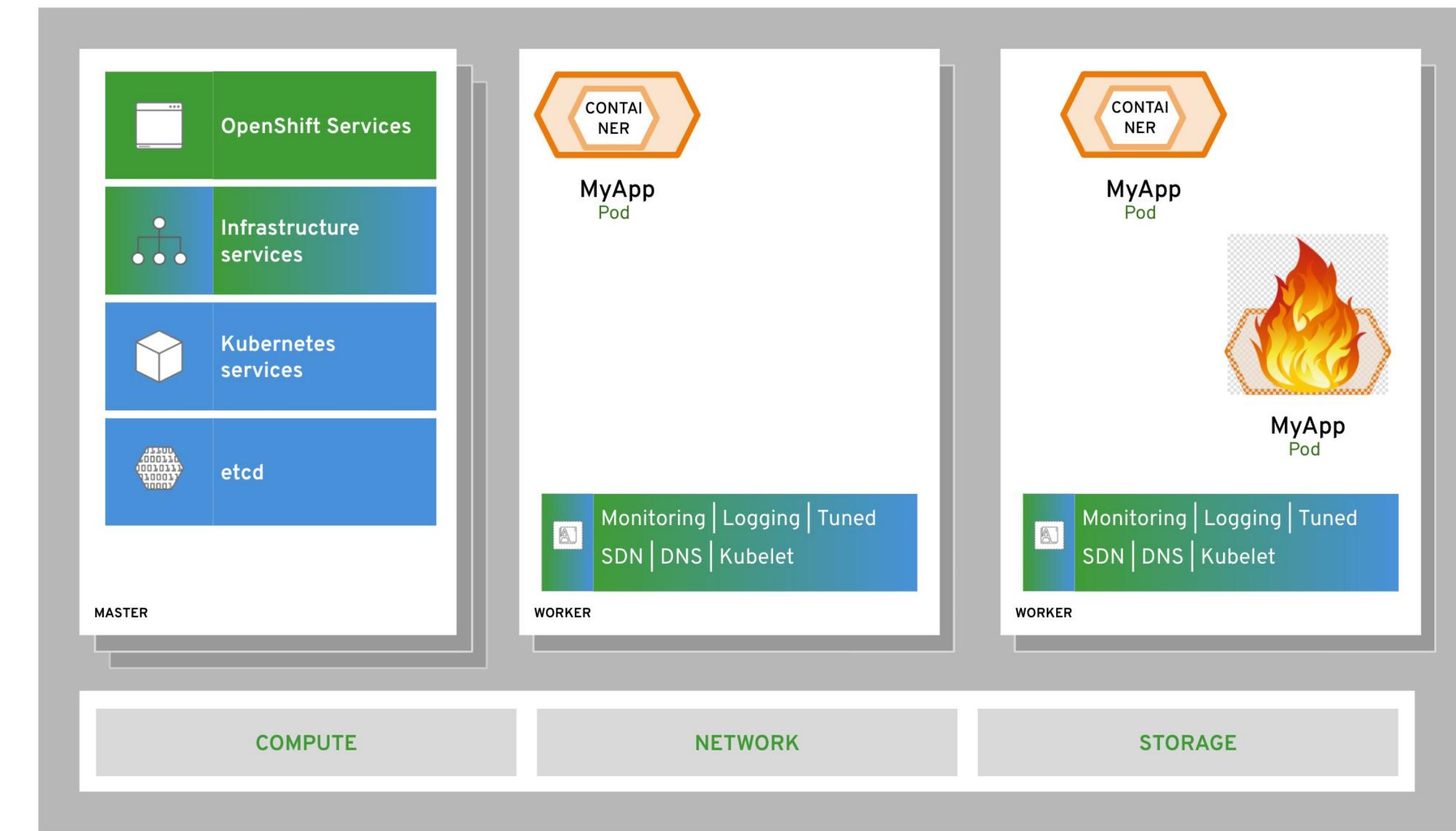
OpenShift Core Concepts

Unhealthy Pods

- ¿Qué sucede cuando un master node detecta en su probe que un pod falla?
- ¿Qué sucede si un contenedor dentro de un pod termina (exit) porque hubo un crash u otro problema?

Command line

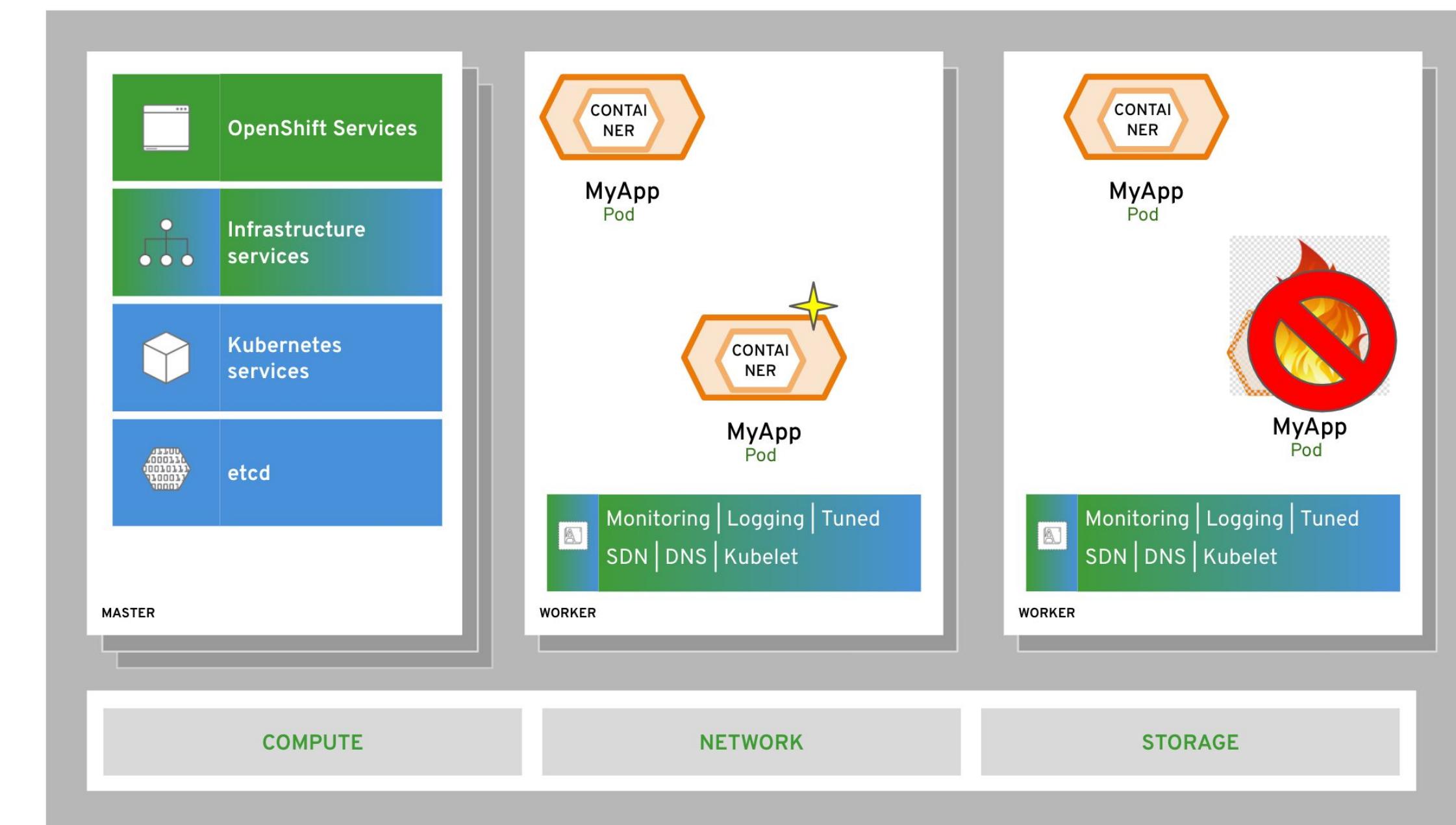
```
$ oc get pods -o wide  
$ oc get pods -w
```



OpenShift Core Concepts

Remediation Pod Failures

- Los nodos master automáticamente reinician el pod que falla a iterativos probes debido a fallas.
- Los pods que fallan con mucha frecuencia se los marca como incorrectos y no se reinician temporalmente.
- La capa de servicios (service) solo envía tráfico a los pods en buen estado.
- Los nodos master organizan automáticamente este comportamiento para mantener la disponibilidad del servicio.



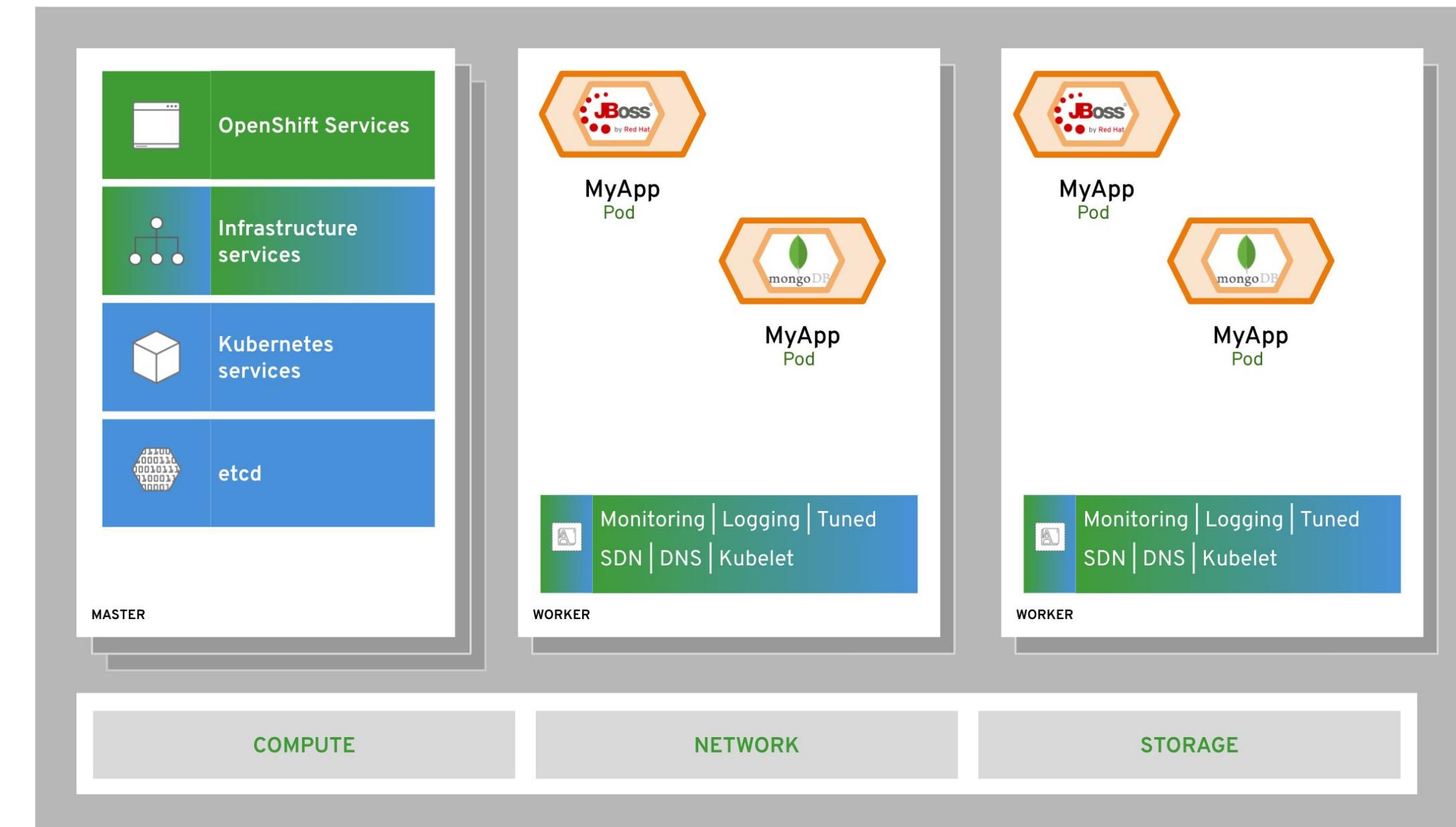
OpenShift Core Concepts

Scheduler

- Es el componente responsable de determinar la ubicación de los pods en los nodos.
- Para realizar la ubicación de un pod se tiene en cuenta la memoria actual, la CPU, labels de selección, etc.
- Para una alta disponibilidad del servicio los pods de aplicación se distribuyen entre distintos nodos workers

Command line

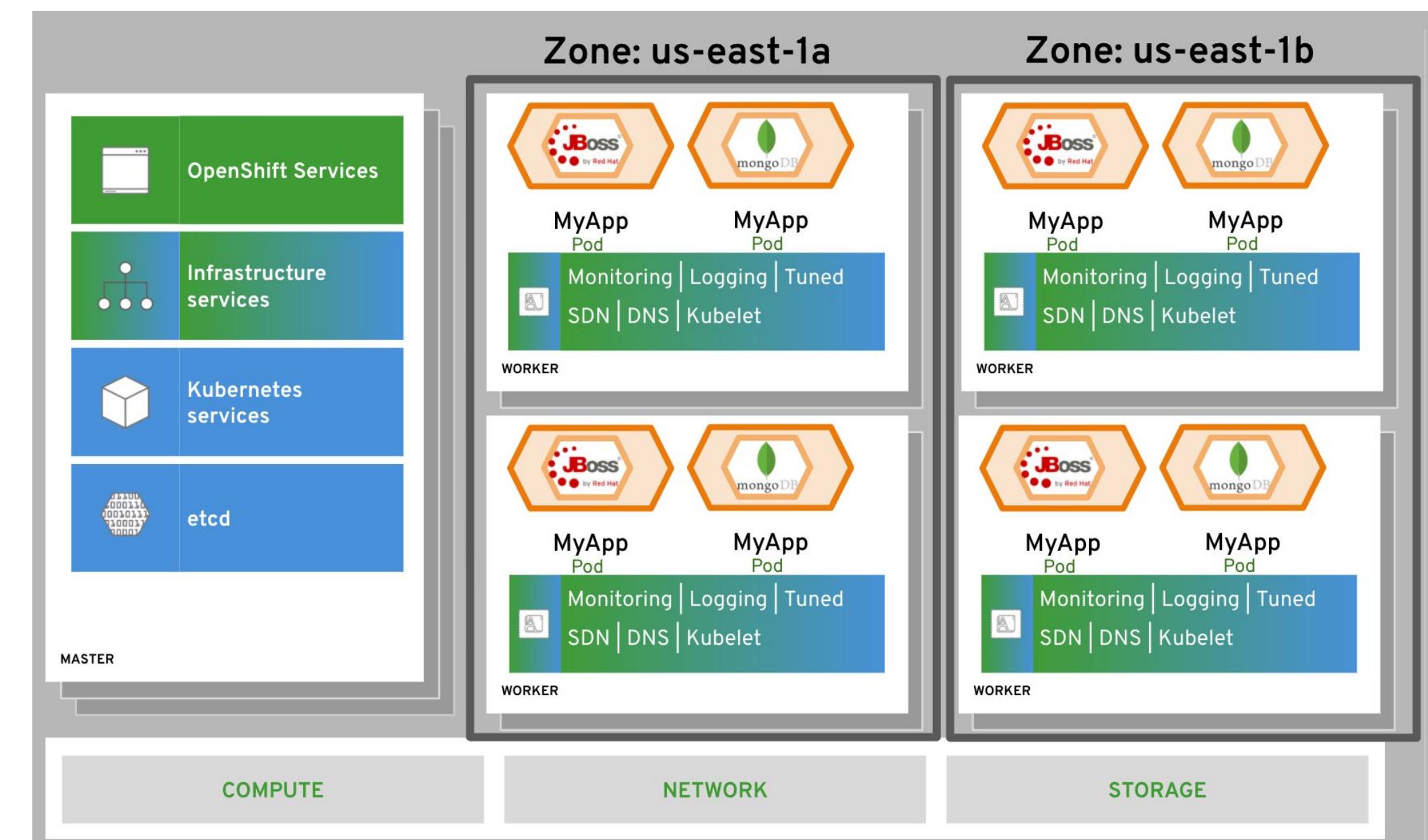
```
$ oc get pods -o wide  
$ oc get pods -w
```



OpenShift Core Concepts

Scheduler

- Es posible organizar los pods en mapas topológicos para los deployments de aplicaciones, sean por regiones, zonas, zones de red, etc.
- Es posible administrar complejos escenarios para distintas cargas de trabajo.
- Para la configuración es utilizada una combinación de grupos de nodos y labels selectores.
 - Por ejemplo, despliegues de aplicaciones por zona o regiones.
- Para ambientes de nube es posible escalar automáticamente nodos.



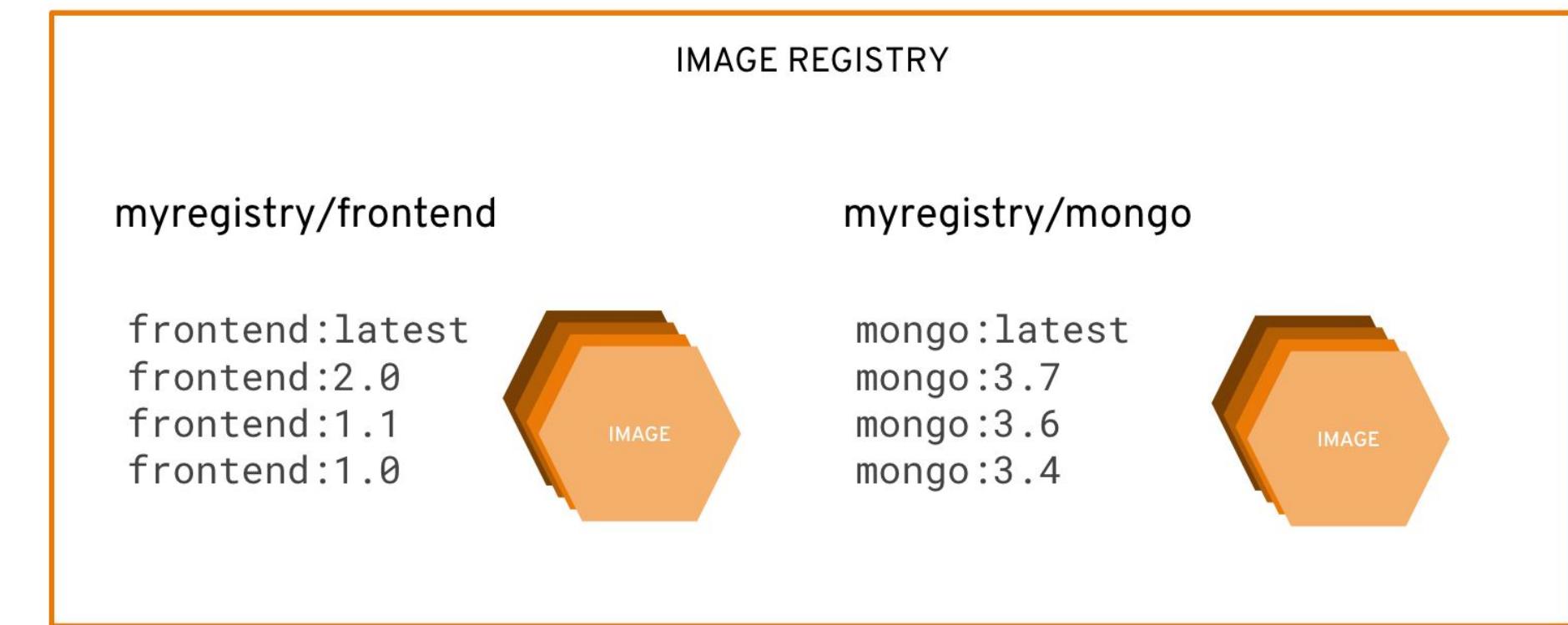
OpenShift Core Concepts

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.

Command line

```
$ oc get images  
$ oc get imagesstreams  
$ oc adm top images
```



[**Red Hat Container Catalog \(RHCC\)**](#) es una fuente segura, certificada y actualizada de imágenes de contenedores

[**Red Hat Software Collections Library \(RHSC\)**](#), es una colección de software para desarrolladores.

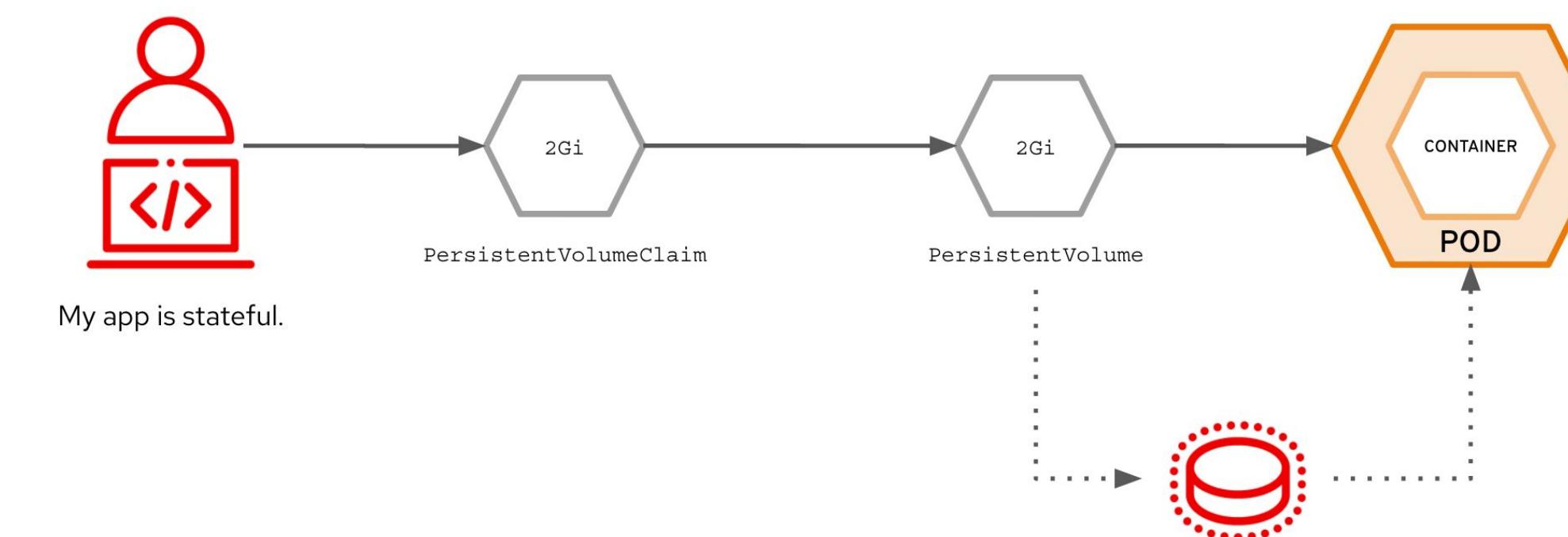
Arquitectura y Conceptos

Application Data

- Los contenedores son nativamente efímeros
 - Los datos no son guardados cuando los contenedores son reiniciados o recreados.
- Openshift provee un subsistema de persistencia de datos para que los pods puedan conectarse a un sistema de almacenamiento.
 - Permite poder construir aplicaciones que persistan estado.
- Openshift provee una amplia matriz de compatibilidad de tecnologías de storage incluyendo:
 - Raw devices: iSCSI, Fiber channel.
 - Enterprise NFS
 - Cloud-type Options: Ceph, AWS EBS, pDisk, VMware.

Command line

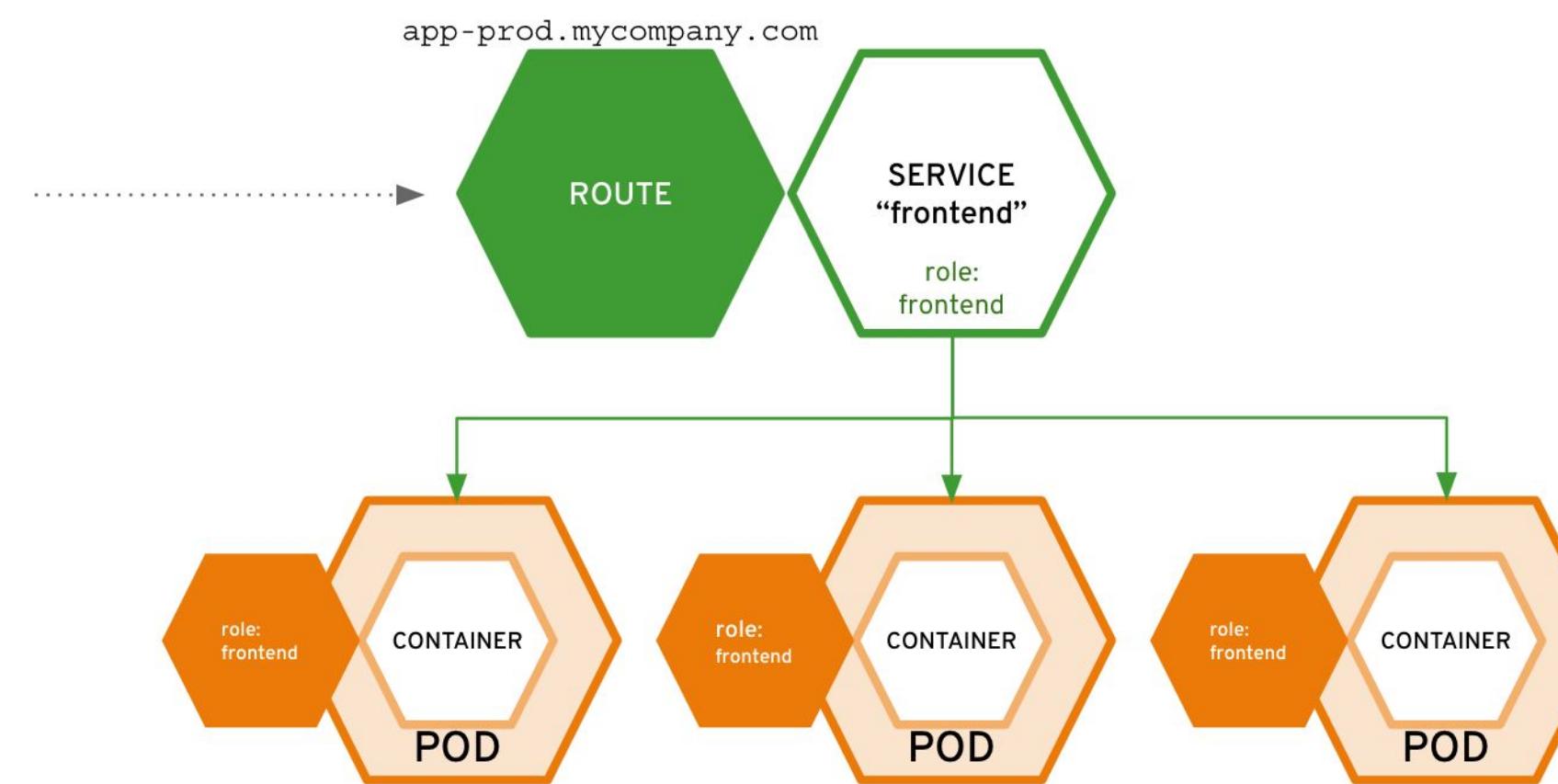
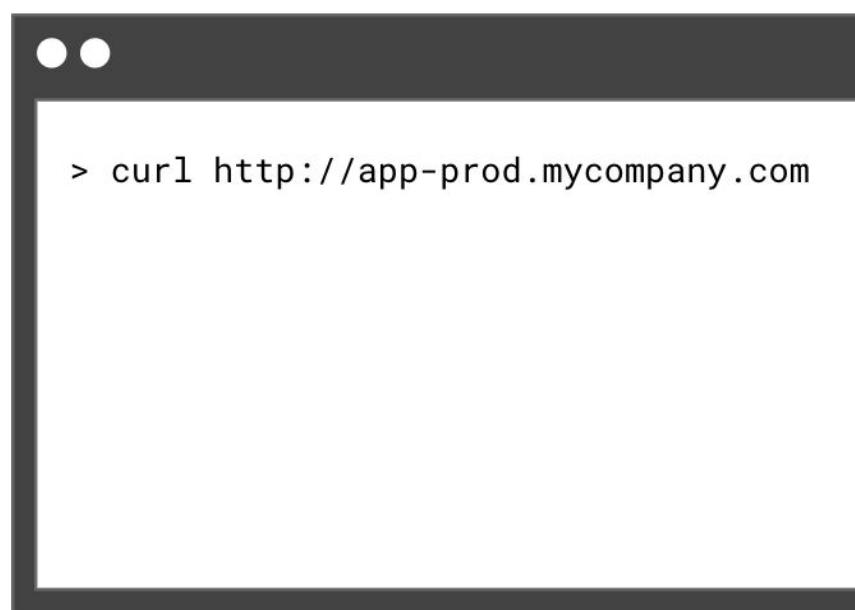
```
$ oc get pv # cluster admin  
$ oc get pvc  
$ oc get pods -o yaml  
$ oc describe pod PODNAME
```



Openshift Core Concepts

Routing Layer

- Provee conexión de acceso externo a aplicaciones que corran dentro de Openshift.
- Trabaja de manera cercana con la capa de servicios.
- La capa de ruteo corre como pods dentro de Openshift.
- Provee:
 - Balanceo de carga automático para los pods que requieren acceso externo a clientes.
 - Balanceo de carga y auto-routing cuando los pods están en estado unhealthy.
- La cara de ruteo es *pluggable and extensible*.
- Opciones incluido non-Openshift software routing.



Command line

```
$ oc get services  
$ oc get routes
```

OpenShift Core Concepts

ReplicaSet y Replication Controller

El objetivo es asegurar el número de réplicas de pods deseados se mantenga todo el tiempo.

Dos implementaciones para este fin:

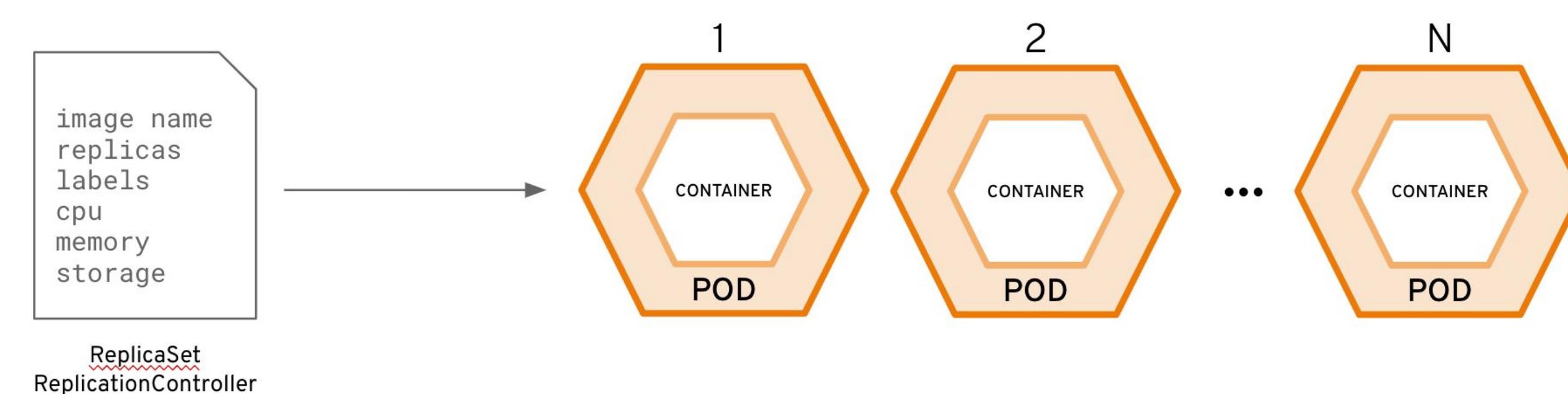
- ReplicaSet = Kubernetes
- ReplicationController = OpenShift

Si un pod termina (exit) o es borrado (deleted), ReplicationController instancia nuevos.

Si mas pods son requeridos, ReplicaSet los borra y mantiene tantos como sean necesarios.

Command line

```
$ oc get replicaset  
$ oc get rs  
$ oc get replicationcontroller  
$ oc get rc  
$ oc explain rs  
$ oc explain rc
```



OpenShift Core Concepts

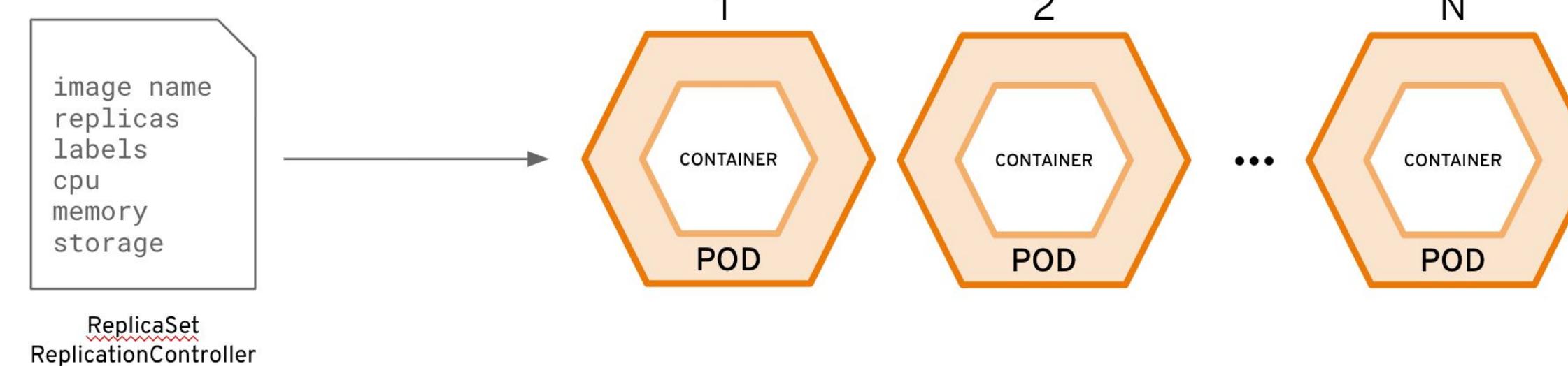
Deployment and DeploymentConfig

- Definen cómo será el roll out de las nuevas versiones de pods.
- Identifica:
 - Nombre de imagen
 - Número de réplicas
 - Label asociado a los nodos donde serán desplegados los pods.

- Update pods en base a:
 - Version
 - Estrategia
 - Triggers de cambio.
- Crean ReplicaSet o ReplicationControllers.

Command line

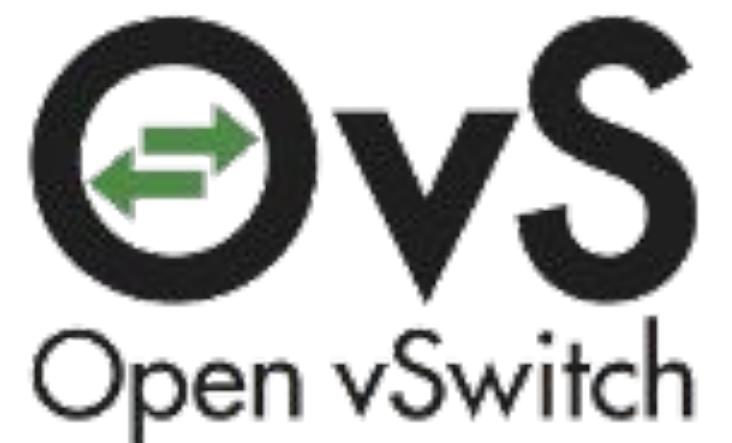
```
$ oc get deployment  
$ oc get deploy  
$ oc get deploymentconfig  
$ oc get dc  
$ oc explain dc  
$ oc explain deployment
```



Openshift Core Concepts

OpenShift Networking

- Container Networking está basada en tecnología de red llamada Open vSwitch
- 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 automático 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.

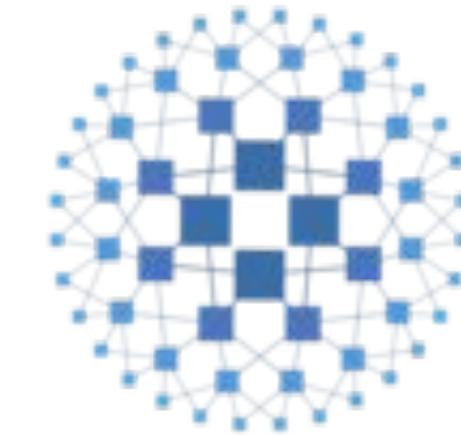


Openshift Core Concepts

OpenShift Networking Resources

Route (Openshift Resource)

- Expone los servicios permitiendo el acceso externo a las aplicaciones internas.
- Consiste en un nombre de ruta, selector de servicio y opcional configuración de seguridad.
- Los routers pueden consumir definidas rutas y endpoints específicos por servicio.
- Proveen conectividad por nombre (DNS)



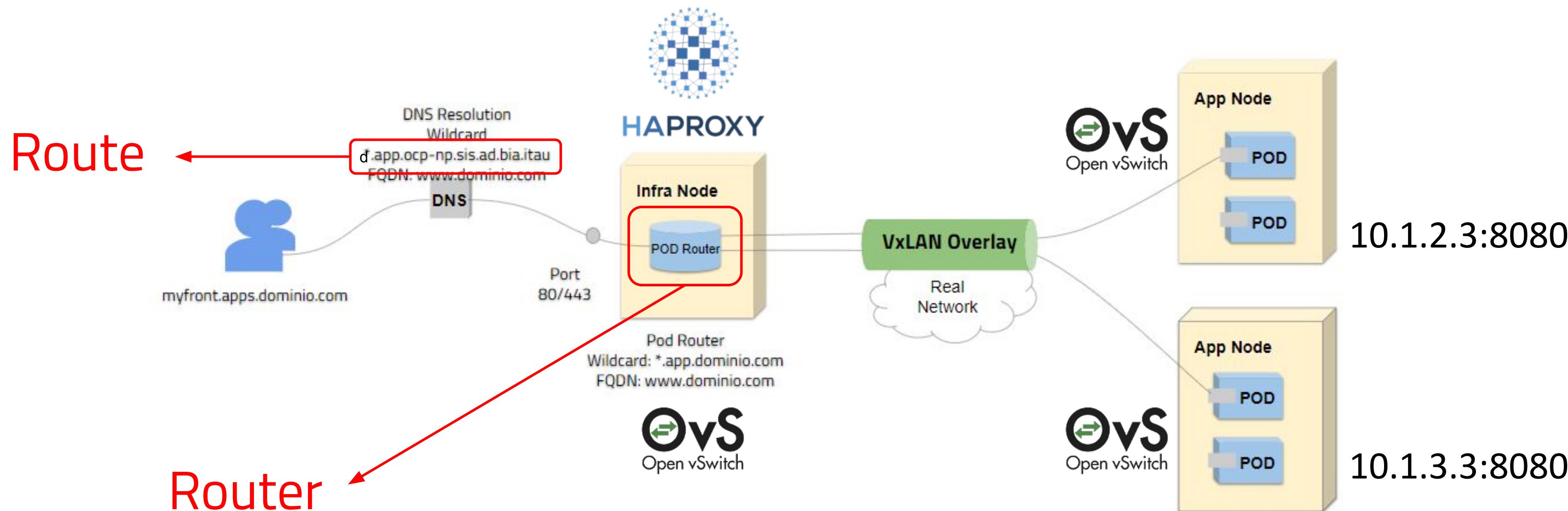
Routers (Openshift Pod)

- Multi-tier application desplegada como pods.
 - La capa de ruteo es requerida para poder alcanzar externamente las aplicaciones.
- Los pod router pueden correr en cualquier nodo worker.
 - El administrador crea un wildcard de dns (CNAME o Registro de tipo A) en un servidor de DNS.
 - La entrada de DNS resuelve los nodos donde están alojados los pod routers.
- Routers son el punto de ingreso para el tráfico destinados a los pods sobre la plataforma.
 - Pod routers resuelven los request externos (<https://myapp.apps.ocp.domain.com>)
 - Funcionan como proxy request para redirigir el tráfico a los pods que atenderán la petición.

Arquitectura y Conceptos

OpenShift Networking Scenario

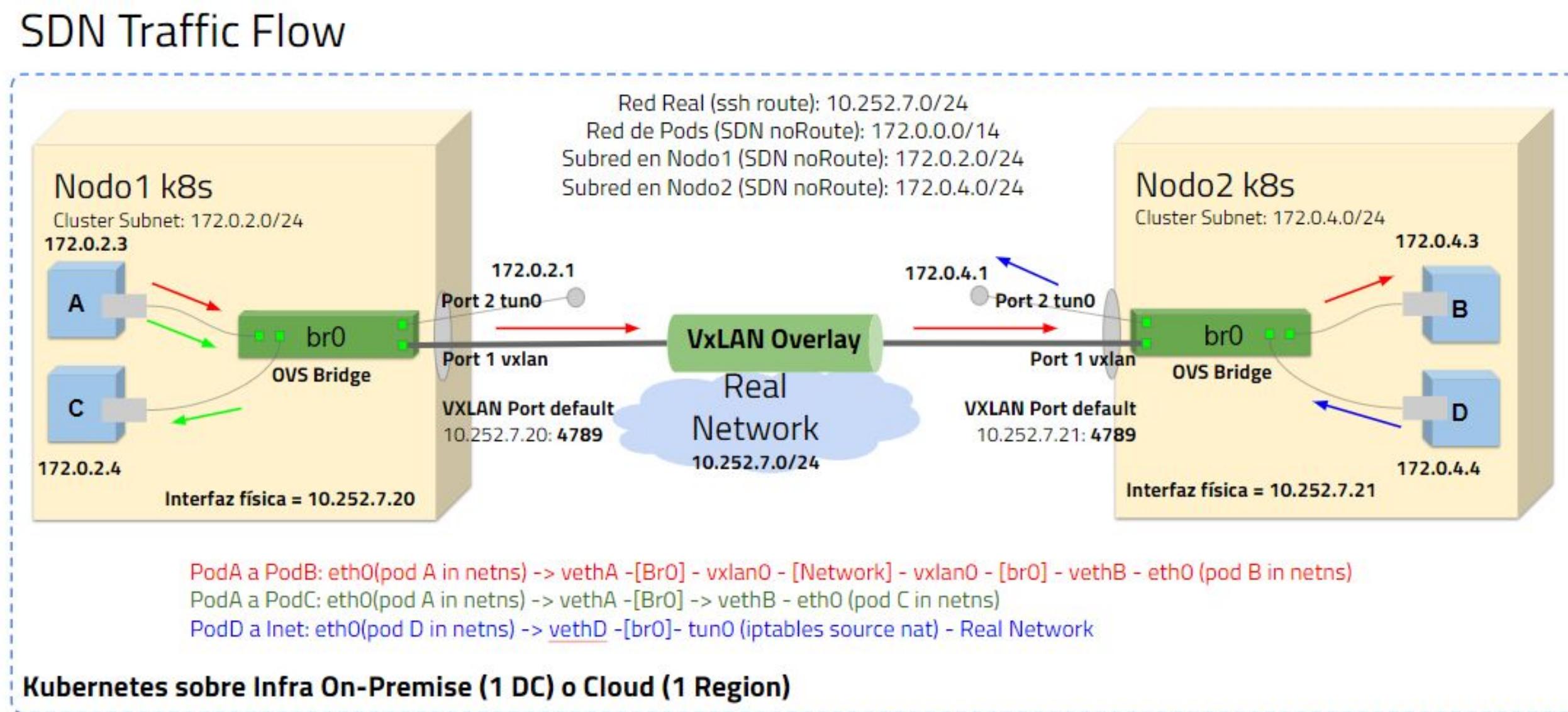
- Los clientes externos ingresan al browser la dirección myapp-project.apps.ocp.dominio.com:80
 - DNS resuelve un host donde corre el pod router.
 - Usando openshift-sdn overlay network:
 - El router chequea si la ruta existe para el requerimiento
 - El router hace un proxies request al pod interno con IP:port (10.1.2.3:8080)



Openshift Core Concepts

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.



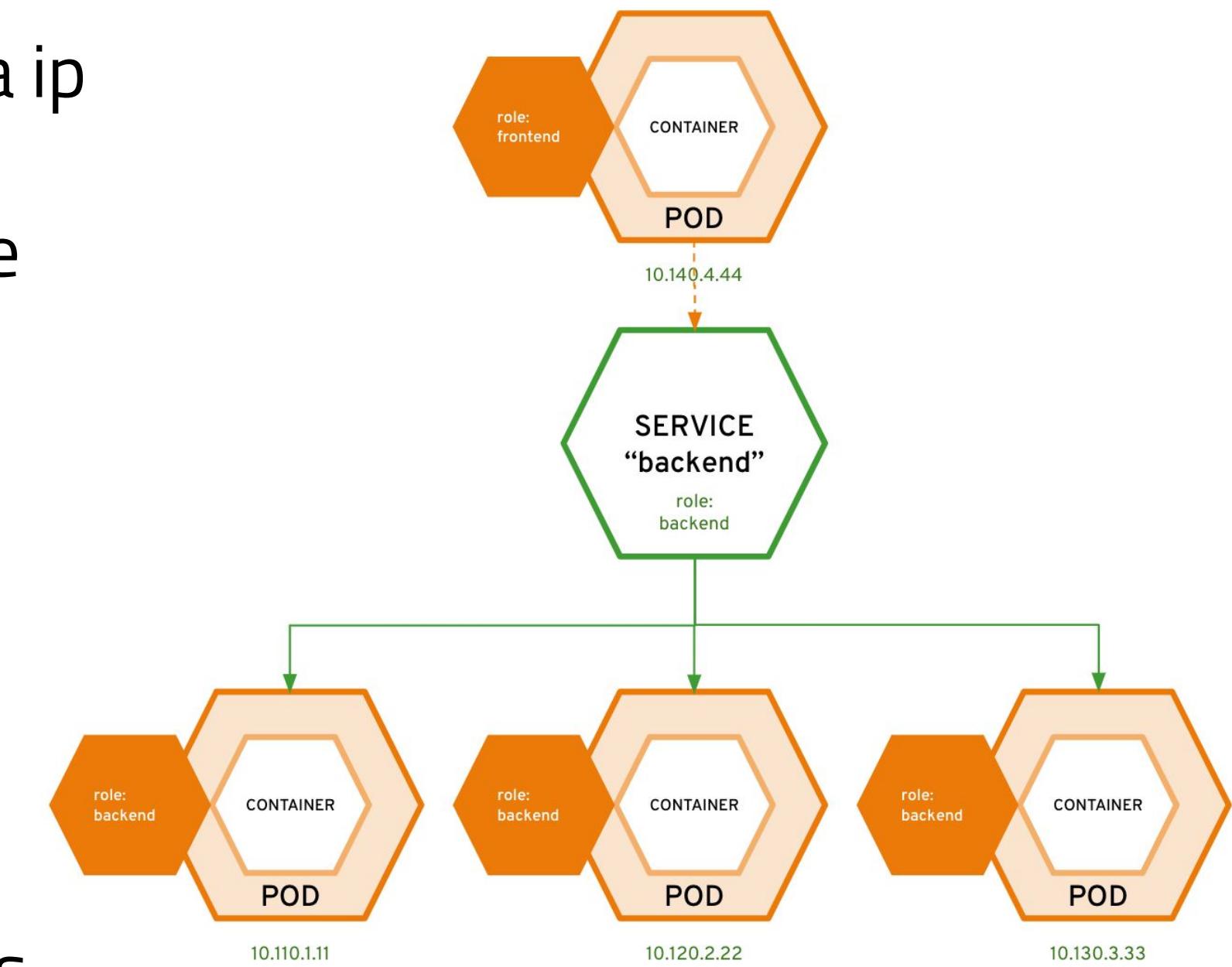
OpenShift Core Concepts

Service and Pods

- Los servicios ofrecen un acceso permanente a través de una sola ip a un grupo similar de pods.
- Internamente, los servicios balancean el tráfico hacia los pods que actúan como backend.
- Los pods backend pueden ser agregados o removidos desde el servicio 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
 - Existe un servicio de DNS interno en OpenShift.

Cuando un pod envía tráfico a un servicio, este presenta uno o más pods.

- Los contenedores envían tráfico a un ip de servicio 10.140.4.44:8080
- El servicio realizar un proxy request a la ip del pod 10.120.2.22:8080



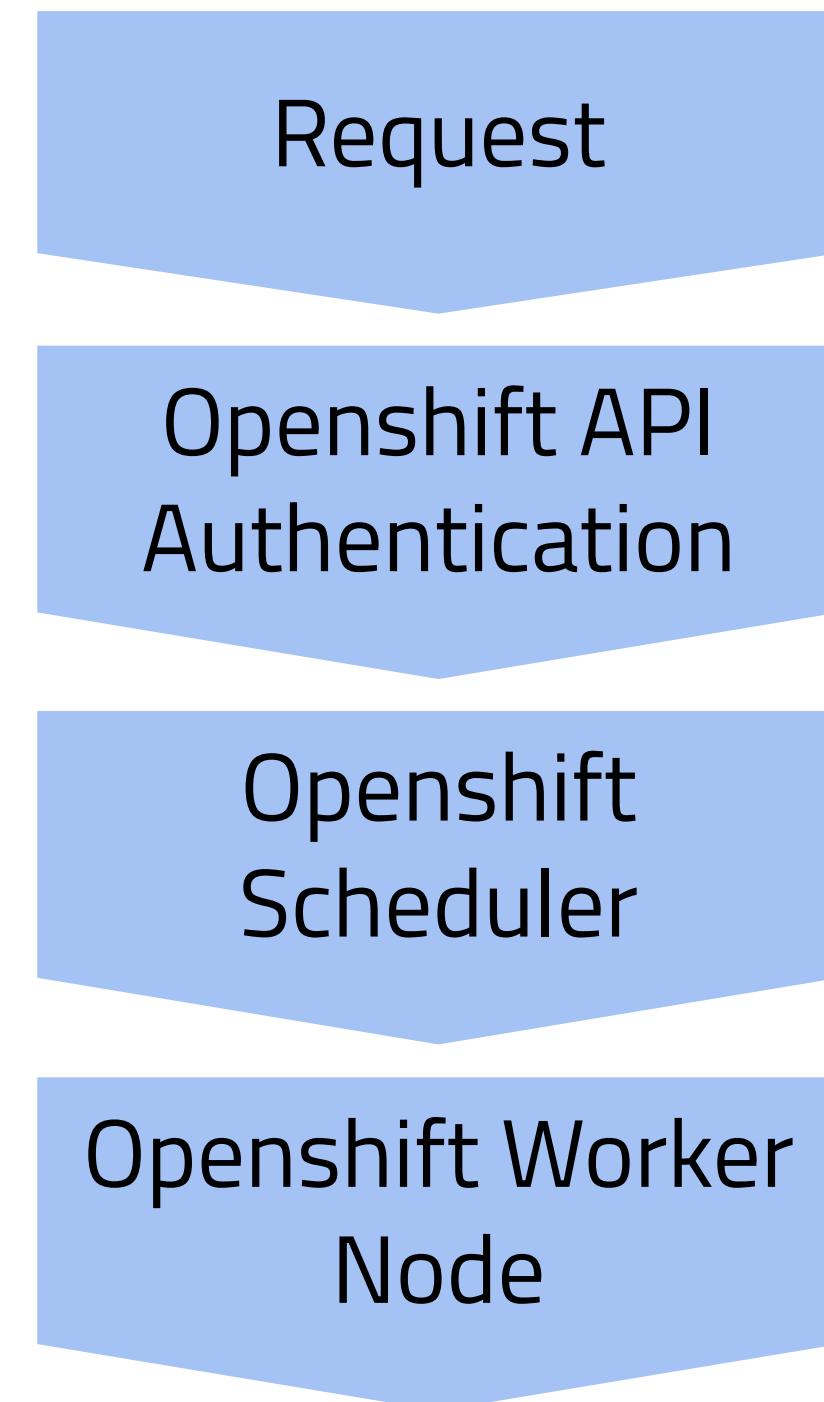
Command line

```
$ oc get routes  
$ oc get services
```

OpenShift Core Concepts

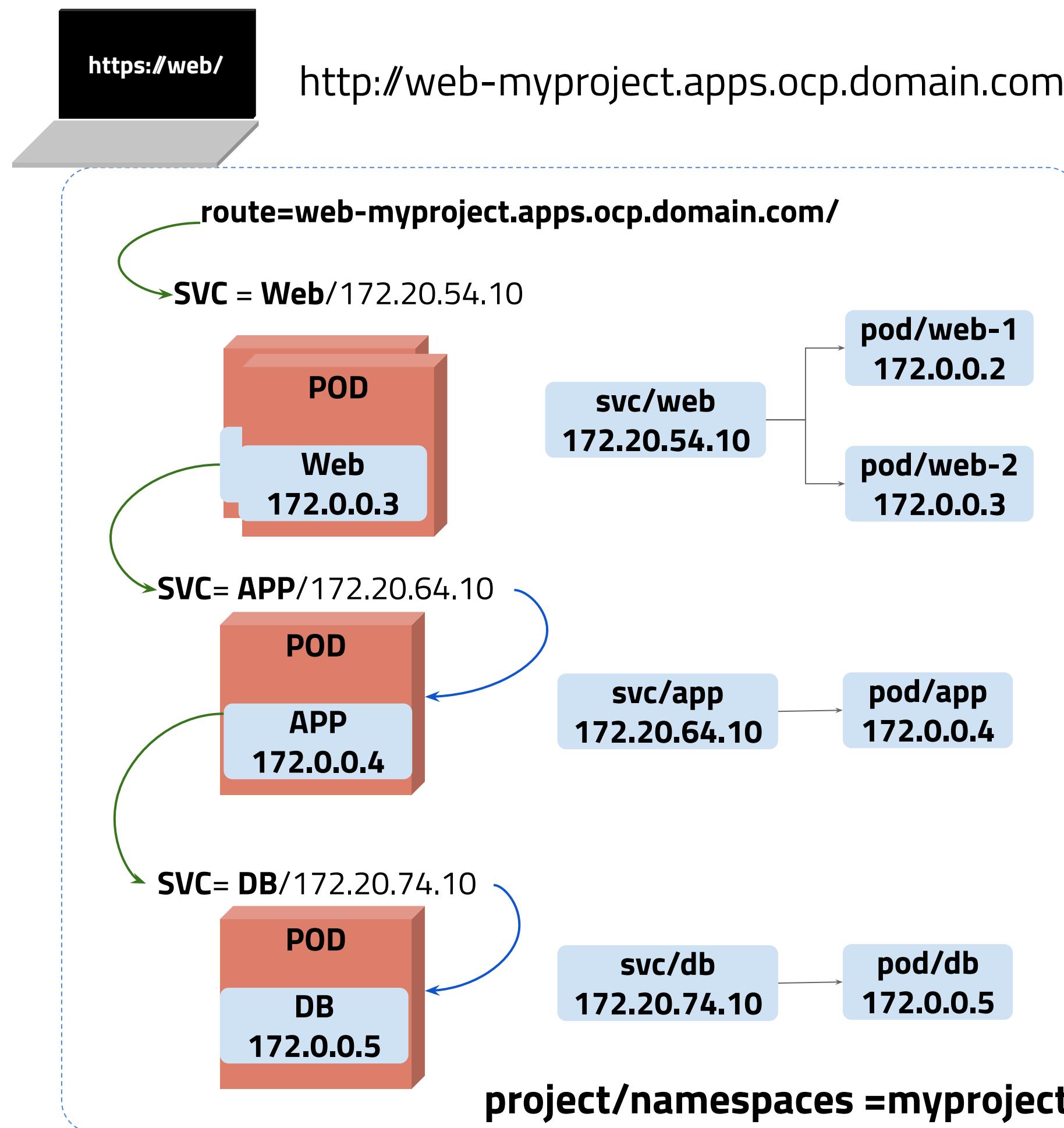
Container Deployment Workflow

- Nuevas aplicaciones son creadas vía CLI, Web Console o API.
- **OpenShift API/authentication:**
 - Aprueba el request, son validados los permisos de usuario, los resource quotas y otra información.
 - Crear los recursos que soportan el despliegue aplicativo: deployment configuration, replication controllers, services, routes, persistent storage claims
- **OpenShift scheduler:**
 - Designa un nodo worker por cada pod, considerando recursos disponibles, carga y distribuye las aplicaciones entre múltiples nodos para mayor disponibilidad.
- **OpenShift worker node:**
 - Descarga la imagen desde el registro de imágenes que esté disponible.
 - Inicia el pod aplicativo.



Openshift Core Concepts

Openshift project



- 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).

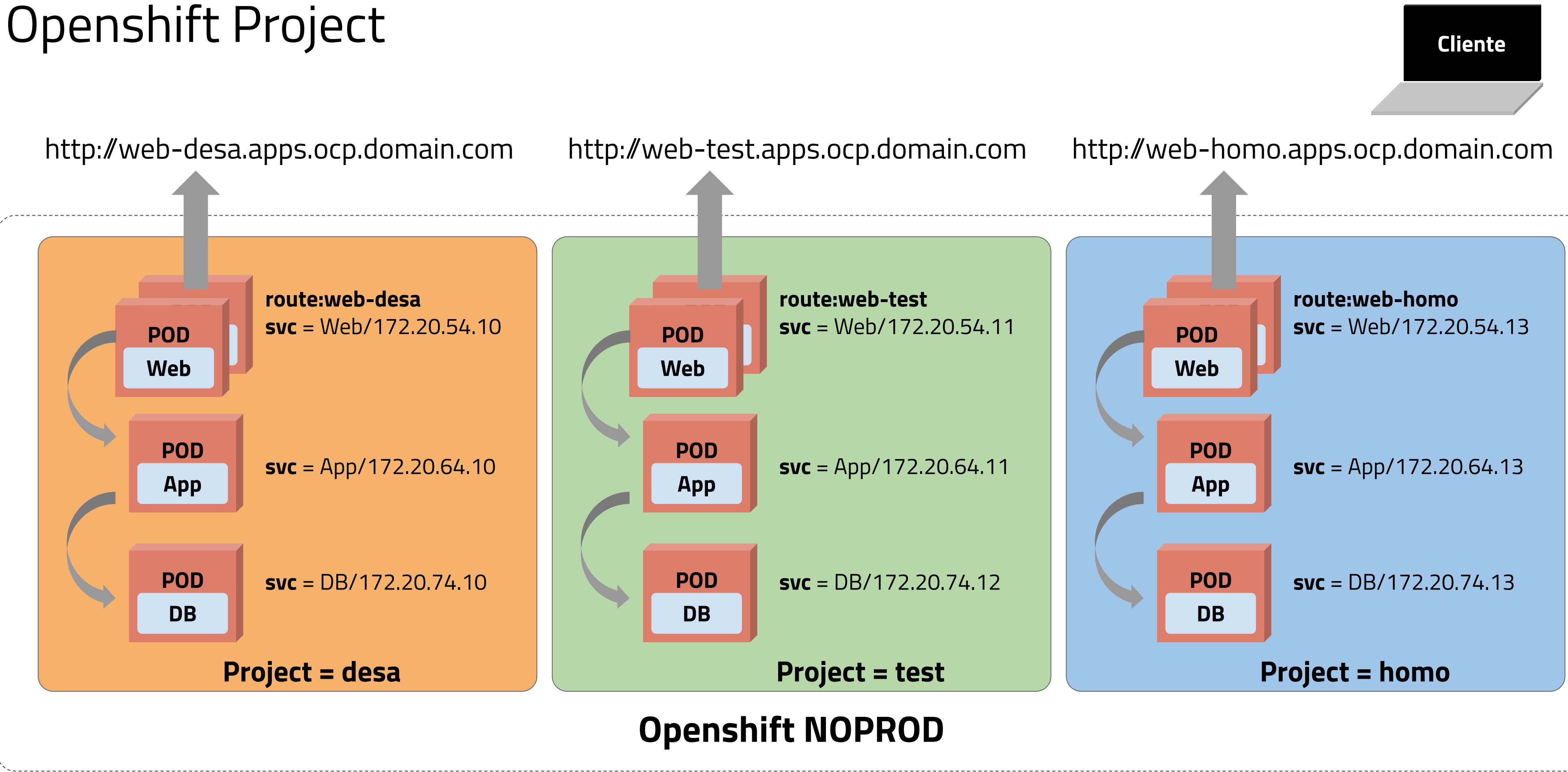
- Un **proyecto** en Openshift es igual que un namespace solo que extiende funcionalidad para alojar datos de usuario.

Recomendaciones a la hora de usar **proyectos**.

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

OpenShift Core Concepts

OpenShift Project



Arquitectura y Conceptos

Quotas and Resource Limits

Project: 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

Project: 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

Project: homo

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

Quotas Large

cpu=8
mem=8G

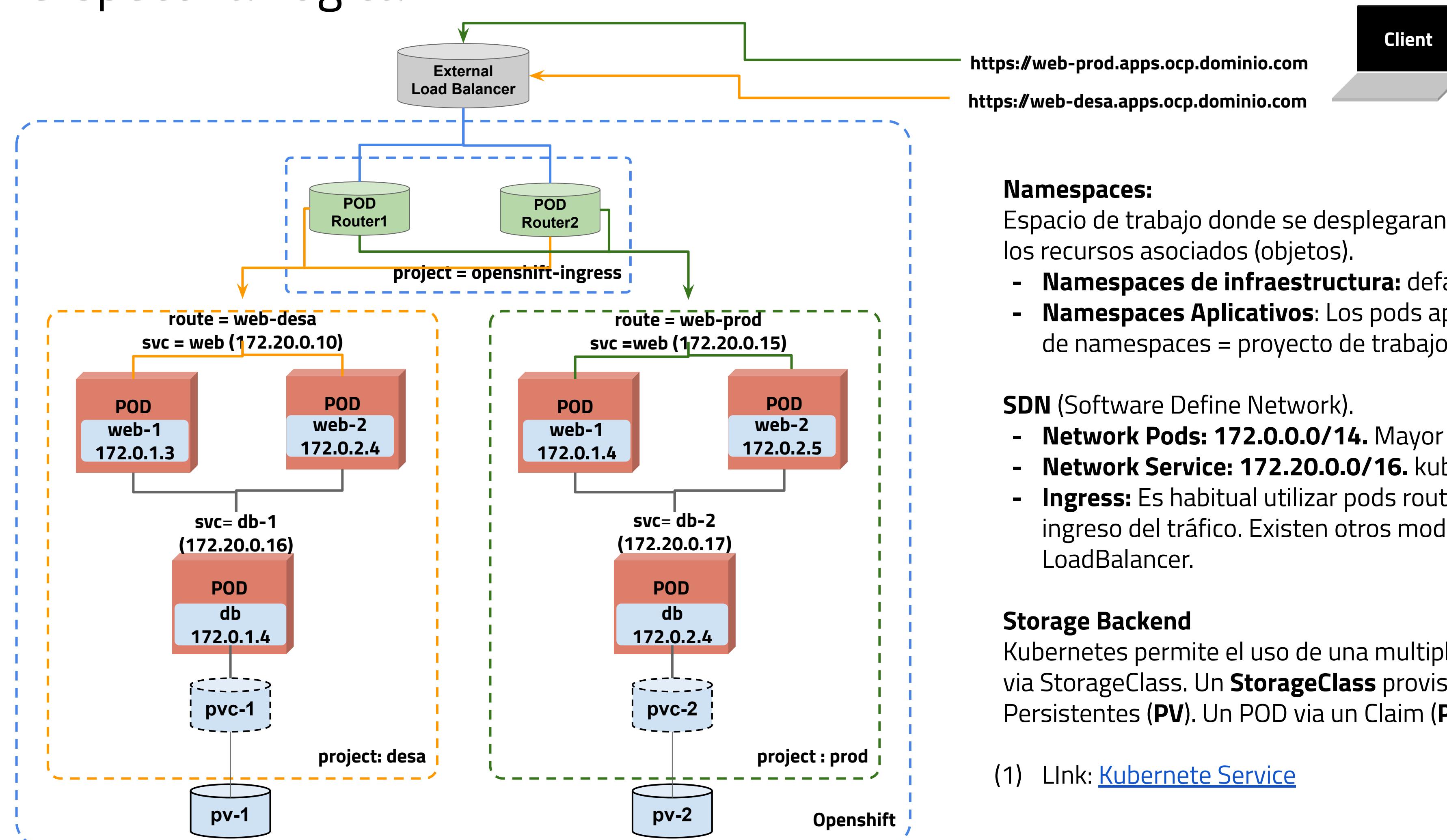
Network Policies

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

OpenShift NO PROD core=36 mem=211

OpenShift Core Concepts

Perspectiva Lógica



Namespaces:

Espacio de trabajo donde se desplegarán pods y se definirán todos los recursos asociados (objetos).

- **Namespaces de Infraestructura:** `default`, `openshift-*`.
- **Namespaces Aplicativos:** Los pods aplicativos contenidos dentro de namespaces = proyecto de trabajo.

SDN (Software Define Network).

- **Network Pods:** `172.0.0.0/14`. Mayor tamaño, pods > svc.
- **Network Service:** `172.20.0.0/16`. `kube-proxy` via iptables. (1)
- **Ingress:** Es habitual utilizar pods router (`haproxy` o `nginx`) para el ingreso del tráfico. Existen otros modos como `nodePort` o `LoadBalancer`.

Storage Backend

Kubernetes permite el uso de una multiplicidad de storage backend via `StorageClass`. Un **StorageClass** provisóna Volumenes Persistentes (**PV**). Un POD via un Claim (**PVC**) reclama un PV.

(1) Link: [Kubernetes Service](#)

OpenShift Core Concepts

Application Deployment

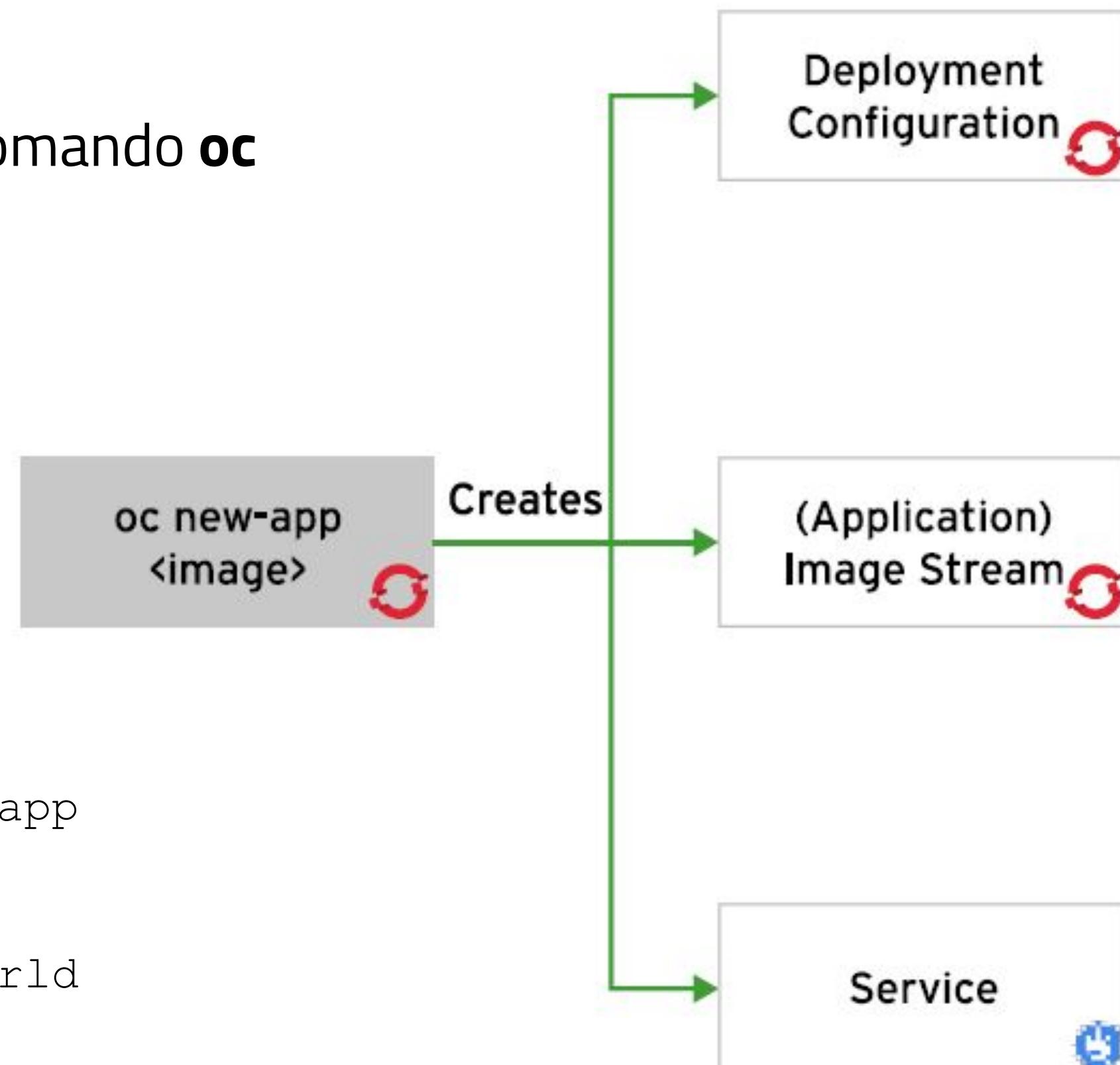
Una manera simple de desplegar aplicaciones es utilizar el comando **oc new-app**

```
$ oc new-app mysql -e MYSQL_USER=user \  
-e MYSQL_PASSWORD=pass \  
-e MYSQL_DATABASE=testdb
```

El comando permite multiples estrategias de despliegue que seran vistas posteriormente.

```
$ oc new-app --docker-image=myregistry.com/mycompany/myapp  
--name=myapp
```

```
$ oc new-app https://github.com/openshift/ruby-hello-world  
--name=ruby-hello
```



→ created by `oc new-app`

OpenShift Core Concepts

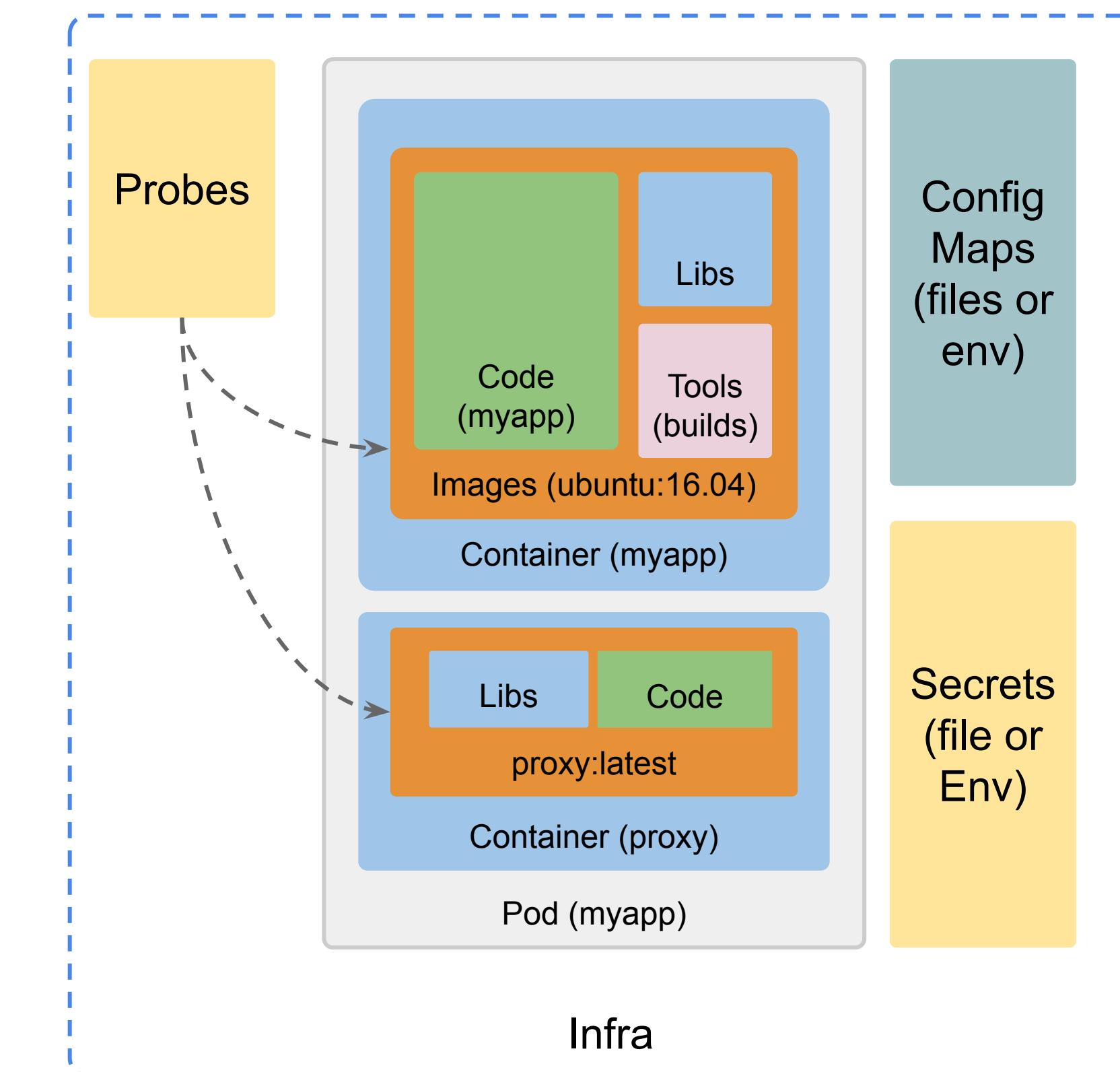
Application Configuration ConfigMaps, Secrets and Probes

- **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.



Web Console

Semperti

Web Console

Overview

- Dos perspectivas
 - Administrator
 - Developer
- Corre como un pod en OCP
- Customizable
- Integra metricas

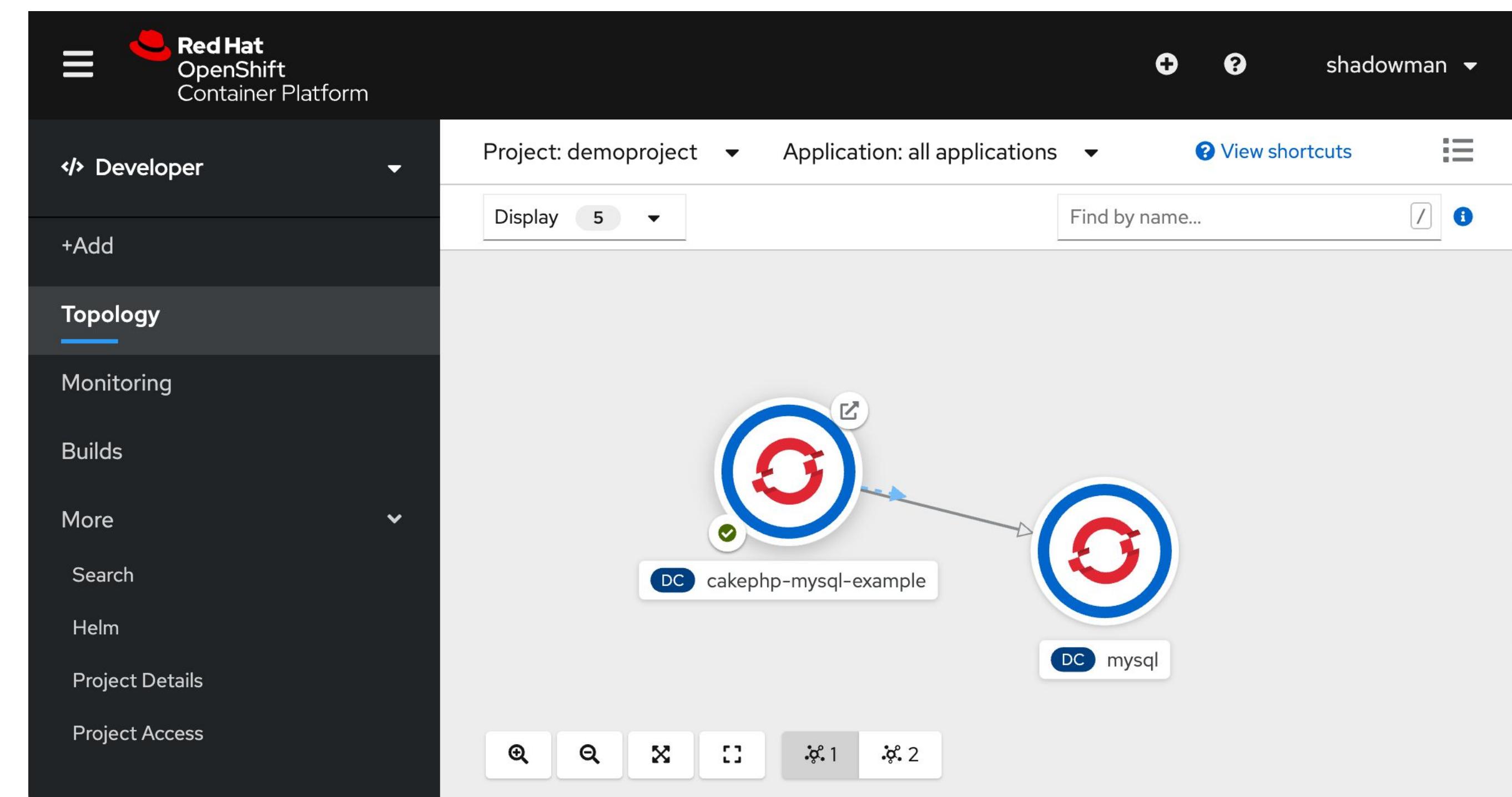
The screenshot shows the Red Hat OpenShift Container Platform Web Console. The top navigation bar includes the Red Hat logo, 'Red Hat OpenShift Container Platform', a user dropdown for 'shadowman', and icons for '+' (Create), '?', and a search bar. The left sidebar has a 'Administrator' dropdown menu open, showing 'Administrator' (selected), 'Developer', 'Search', 'Explore', and 'Events'. Below the sidebar are sections for 'Operators', 'Workloads', 'Serverless', and 'Networking'. The main content area is titled 'Projects' and features a 'Create Project' button. A table lists one project: 'PR demoproject' (Name), 'Demo Project' (Display Name), 'Active' (Status), and 'shadowman' (Requester). A filter bar at the bottom allows searching by name or display name.

Name ↑	Display Name ↓	Status ↓	Requester ↓
PR demoproject	Demo Project	✓ Active	shadowman

Web Console

Developer Perspective

- Vista topológica
 - Contrado en la aplicación
 - Muestra estado de componentes, rutas, pods etc.
 - Se pueden crear relaciones entre aplicaciones arrastrando las flechas.
 - Facil agregado de componentes aplicativos.



Web Console

Developer Perspective

- Mas avanzado
- Detalle de proyectos
 - Status, Utilización, Eventos, Quotas
- Acceso a proyectos
 - Control Usuarios y Grupos
- Metricas

The screenshot shows the Red Hat OpenShift Container Platform Web Console interface. The top navigation bar includes the Red Hat logo, 'Red Hat OpenShift Container Platform', a search bar, and user information for 'shadowman'. The left sidebar, titled 'Developer', contains links for '+Add', 'Topology', 'Monitoring', 'Builds', 'More' (with options for 'Search', 'Helm', 'Project Details', and 'Project Access'), and 'Metrics'. The main content area displays the 'demoproject' details under the 'Project: demoproject' header. The 'Overview' tab is selected, showing the project's status as 'Active' with a green checkmark icon. Other tabs include 'Details', 'YAML', 'Workloads', and 'Role Bindings'. The 'Details' section shows the project's name ('demoproject'), requester ('shadowman'), and labels ('No labels'). The 'Status' section shows the project is active. The 'Inventory' section shows 0 Deployments, 6 Pods, 0 PVCs, and 2 Services. The 'Utilization' section shows CPU usage at 4.56m over the last hour, with a chart showing usage fluctuating between 2m and 6m. On the right side, there are sections for 'Activity' (showing no ongoing activities) and 'Recent Events' (with a 'Pause' button). A bottom footer bar includes links for 'Documentation', 'Community', 'Support', and 'Contact Us'.

Web Console

Administrator Perspective

- Overview
 - Status, Actividad/Eventos, Inventario, Capacidad, Utilización
- Administración de todos los objetos del mismo tipo.
- Logging y metricas
Vista avanzadas para: Updates, Operators, CRDs, role bindings, resource quotas

The screenshot shows the Red Hat OpenShift Container Platform Web Console. The top navigation bar includes the Red Hat logo, 'Red Hat OpenShift Container Platform', and user account information ('admin'). The left sidebar menu is titled 'Administrator' and contains the following items: Home (selected), Overview, Projects, Search, Explore, Events, Operators, Workloads, Networking, Storage, Builds, Monitoring, Compute, User Management, and Administration. The main content area is titled 'Overview' under 'Cluster'. It displays cluster details such as Cluster API Address (https://api.cluster-6681.green.osp.opentlc.com:6443), Cluster ID (71e6b74f-b43c-428a-bb85-d68d2acaec32), Provider (OpenStack), and OpenShift Version (4.4.3). It also shows cluster utilization metrics for CPU and Memory over a 1-hour period. The right side of the screen features a 'Recent Events' section with a 'Pause' button and a detailed log of events from May 6, 2024, involving 'Nameserver li...' and other system components.

DEMO

Lab 2
Overview de
OpenShift

Container Orchestration

Resumen

- Kubernetes
- Openshift
- Arquitecturas y conceptos core
- Network workflow
- Deployment



Día 3

Semperti

Despliegue de Aplicaciones

Overview

- Builds and Deployments
 - Deployment
 - Builds and S2I
 - Images Streams
 - Templates
- Application Configuration
 - Vars Env
 - ConfigMaps
 - Secrets
 - Downwars API



OpenShift Build and Deploy

Semperti

OpenShift Container Platform

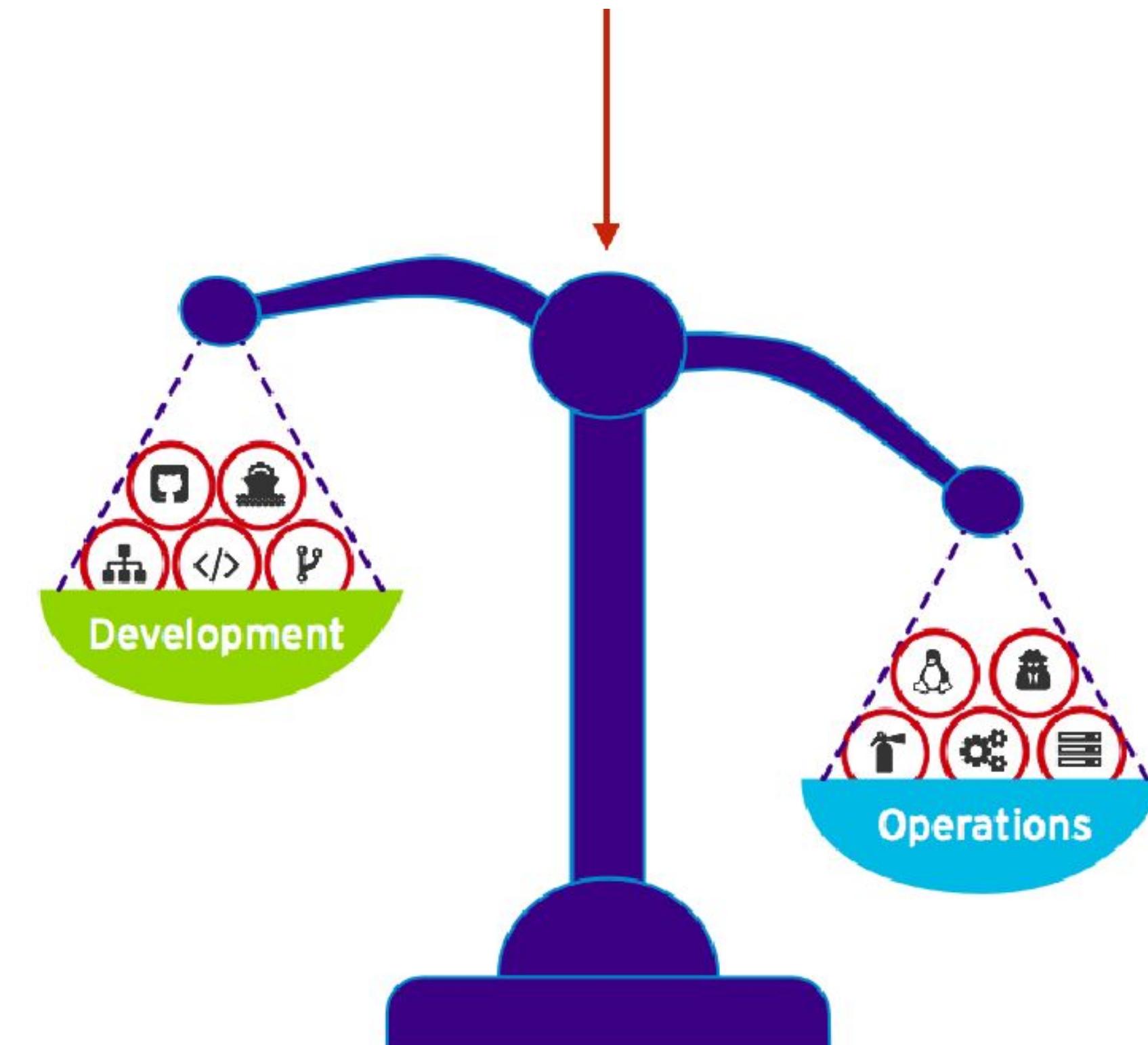
DevOps and CI/CD

- **Desarrolladores se ocupan de::**

- Building applications
- Automating tests
- Continuous integration
- Performance tuning
- Debugging

- **Operaciones se ocupa de:**

- Deploying applications
- Managing applications, infrastructure
- Reliability
- Security
- Compliance



OpenShift Container Platform

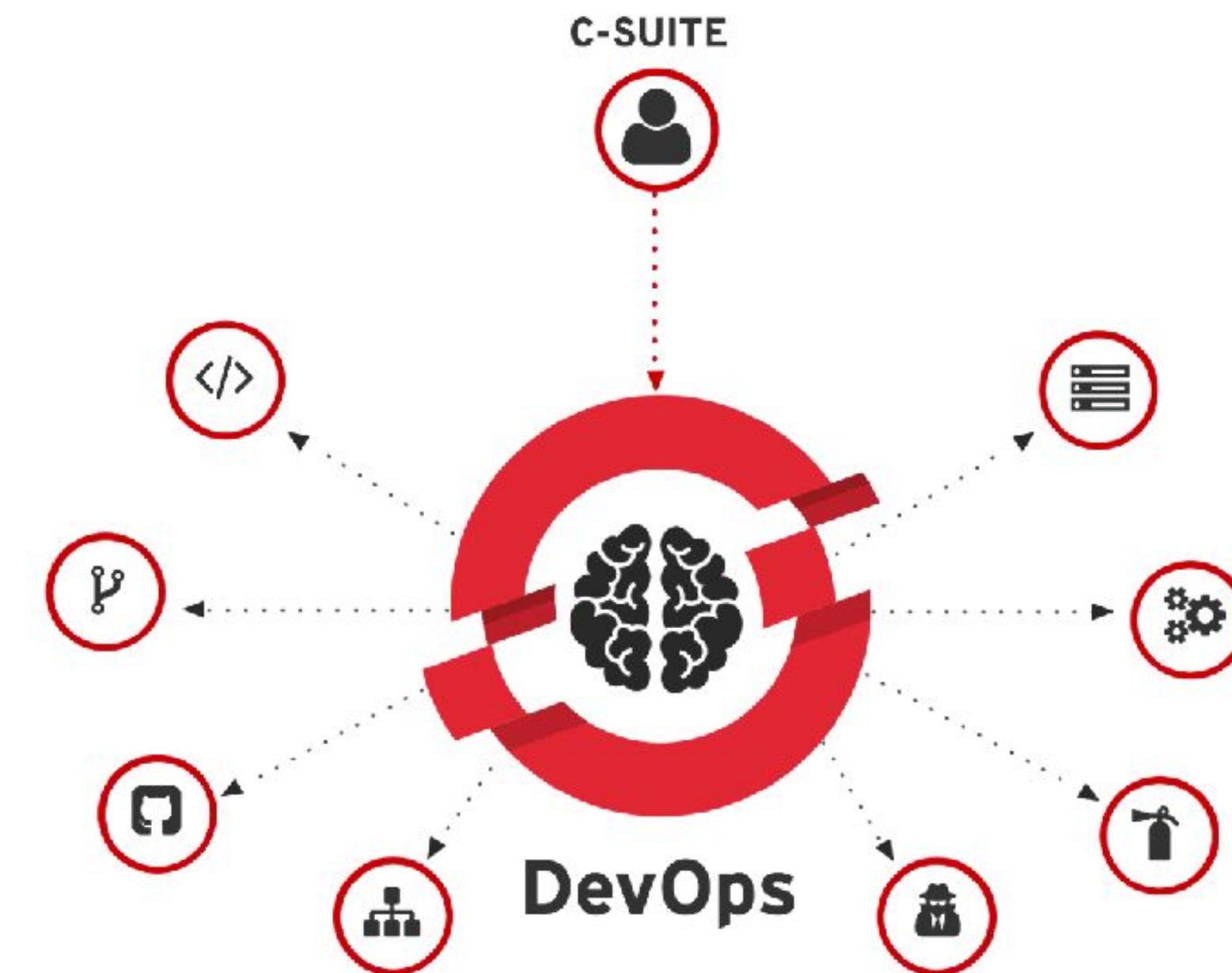
Application Development with Container Platforms

Physical	Virtual	With Container Platform
<ol style="list-style-type: none">1. Have idea2. Get budget3. Submit hardware acquisition request4. Wait5. Get hardware6. Rack and stack hardware7. Install operating system8. Install OS patches/fix-packs9. Create user accounts10. Deploy framework/application server11. Deploy testing tools12. Test testing tools13. Code14. Configure production servers (buy if needed)15. Push to production16. Launch17. Order more servers to meet demand18. Wait19. Deploy new servers20. Etc.	<ol style="list-style-type: none">1. Have idea2. Get budget3. Submit VM request4. Wait5. Deploy framework/application server6. Deploy testing tools7. Test testing tools8. Code9. Configure production VMs10. Push to production11. Launch12. Order more production VMs to meet demand13. Wait14. Deploy application to new VMs15. Etc.	<ol style="list-style-type: none">1. Have idea2. Get budget3. Select best tools4. Code5. Test6. Launch7. Scale automatically

OpenShift Container Platform

What DevOps Is About - Collaboration and Cooperation

- No silos
- Mejora la comunicación y consenso
- Incrementa la agilidad y eficiencia
- Acelera el time-to-market



Openshift Container Platform

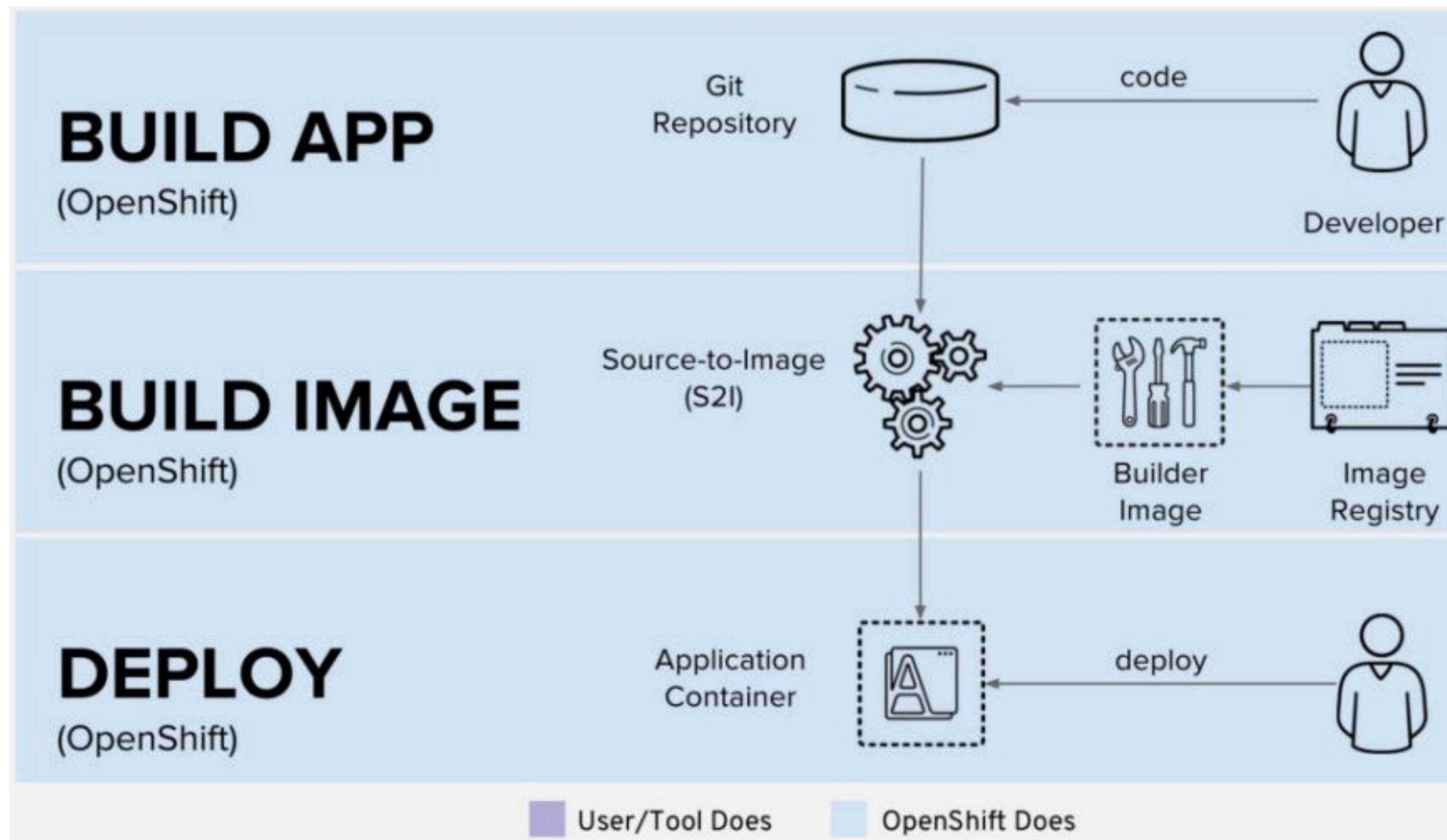
Container Image Builds and Deployments

Existen varias maneras de poder desplegar contendores en Openshift

Deploy source code	Deploy App Binary	Deploy Container Images
{ }	/bin/app	/bin/app

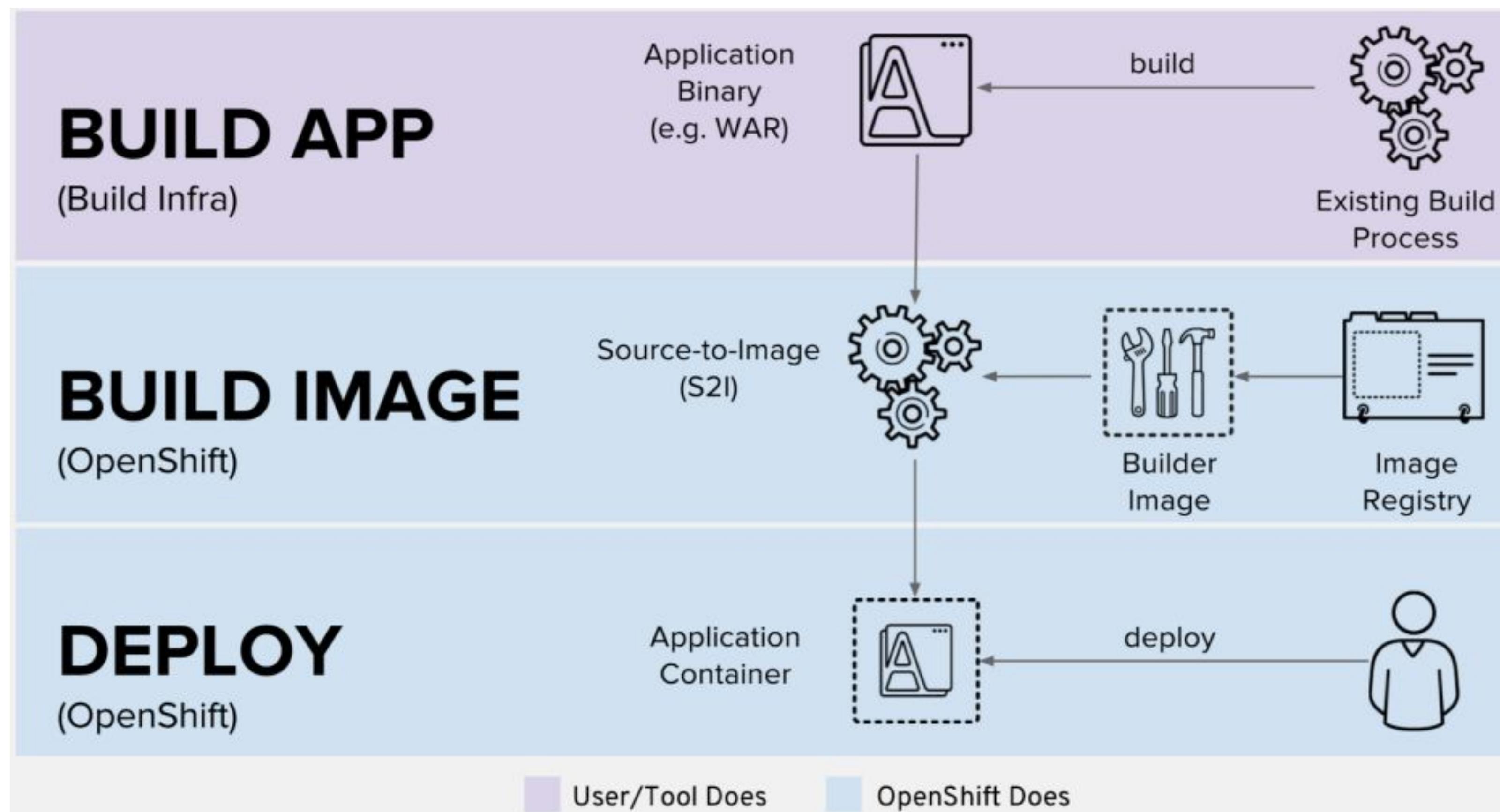
Openshift Container Platform

Deploying Source Code with S2I



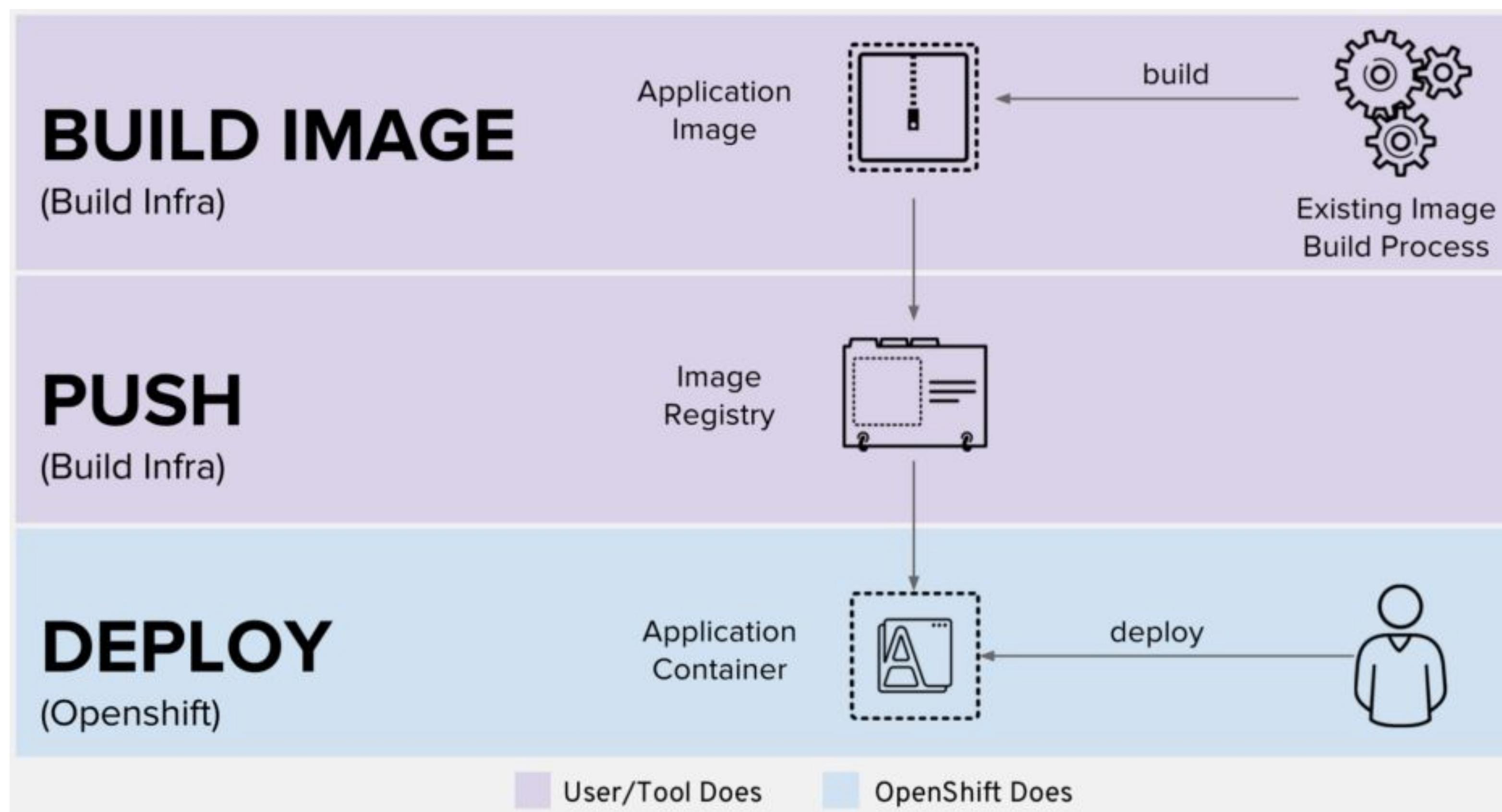
Openshift Container Platform

Deploying an Application Binary with S2I



Openshift Container Platform

Deploying a Container Image



Openshift Container Platform

Deploying Application

Dockerfile

```
$ oc new-app --strategy docker \
http://gitserver.example.com/mydockerfileproject
```

Source Code

```
$ oc new-app http://gitserver.example.com/mygitrepo
```

Con new flow o -i

```
$ oc new-app php~http://gitserver.example.com/mygitrepo
$ oc new-app -i php http://gitserver.example.com/mygitrepo
$ oc new-app -i myis --strategy source \
--code http://gitserver.example.com/mygitrepo
```

Imagen

```
$ oc new-app registry.example.com/mycontainerimage
$ oc new-app --docker-image \
registry.example.com/mycontainerimage
```

LANGUAGE FILES

Java EE	pom.xml
Node.js	App.json package.json
PHP	Index.php composer.json
Python	Requirement.txt config.py
Perl	Index.pl cpanfile

El comando oc new-app creará

- DeploymentConfig > ReplicationController
- ImageStream > Image
- Service
- BuildConfig > build

Deployments

Semperti

OpenShift Deployments

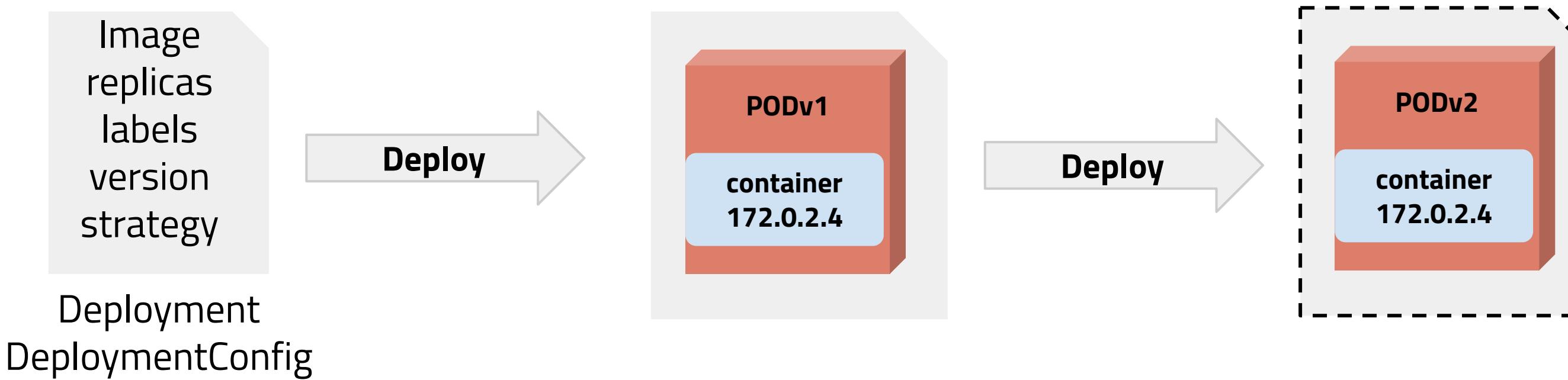
Deployments (deploy) y DeploymentConfig (dc)

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"

Image
replicas
labels
version
strategy

Deployment
DeploymentConfig



Command line

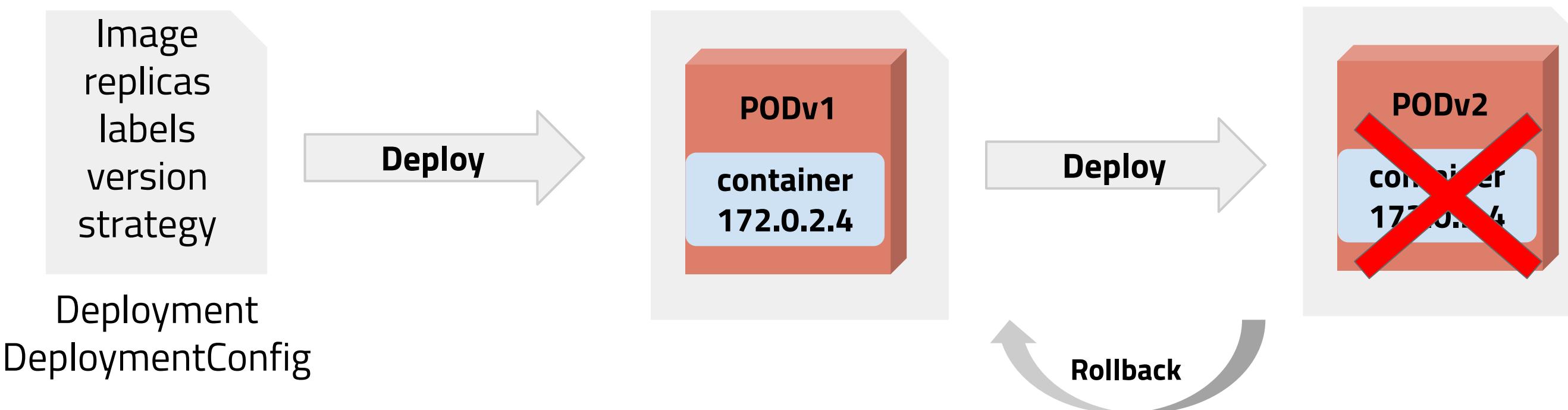
```
$ oc get deploy  
$ oc get deploymentconfig
```

OpenShift Deployments

Deployments Features and Rollback

Deployment Features

- Deployment Config: template para desplegar aplicaciones.
 - Contiene número de versión.
 - Cada vez que se ejecuta un nuevo deployment, se genera una nueva versión del Replication Controller.
- 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.

Command line

```
$ oc get dc -o yaml  
$ oc explain  
dc.spec.strategy
```

OpenShift Deployments

Rolling and Recreates Strategy

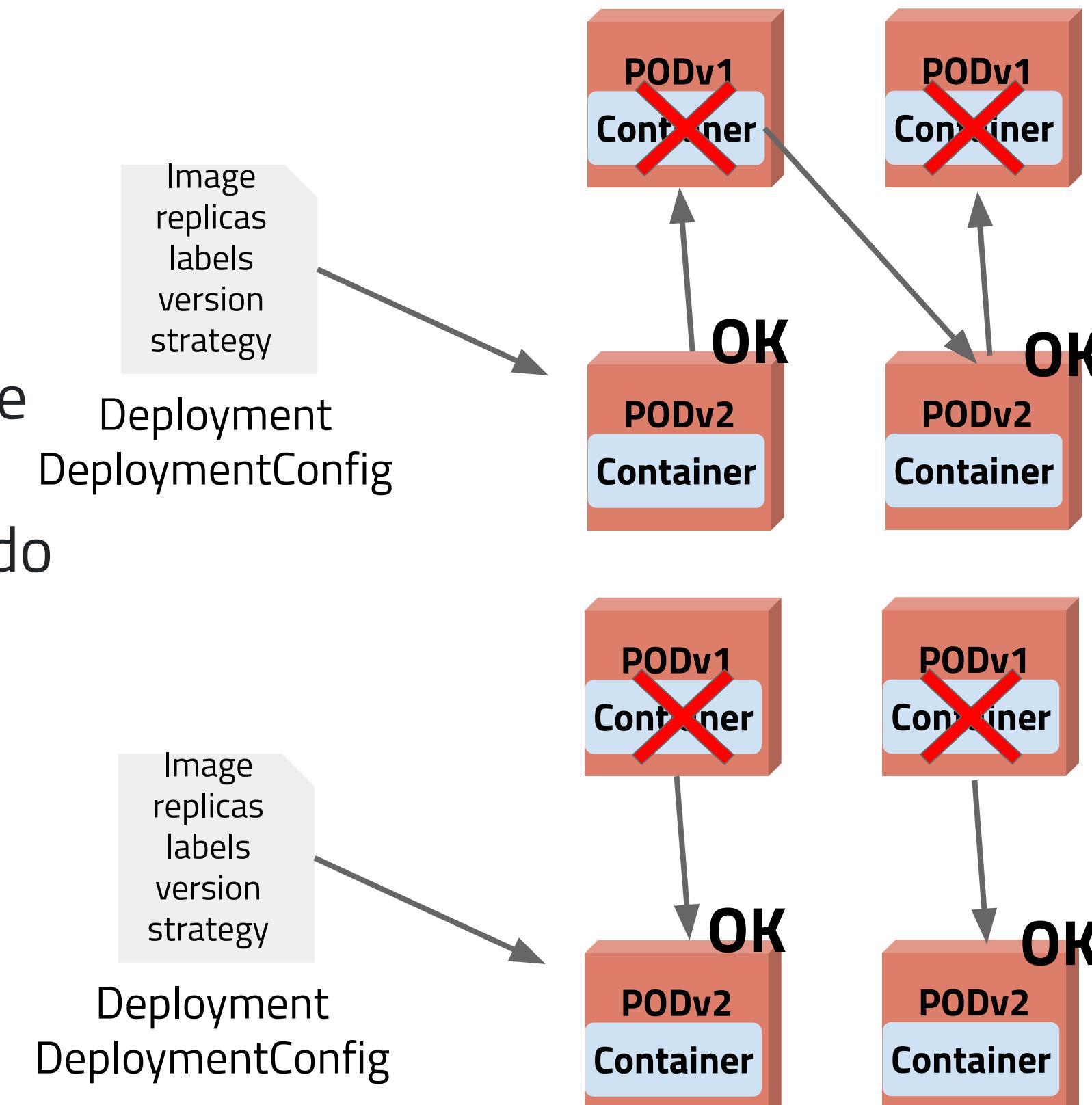
Rolling Strategy

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*

Command line

```
$ oc get dc -o yaml  
$ oc explain dc.spec.strategy
```



OpenShift Deployments

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 valor de maxUnavailable)
 - 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 version previa
 - Scale up a la nueva version
 - Ejecuta post life-cycle hook

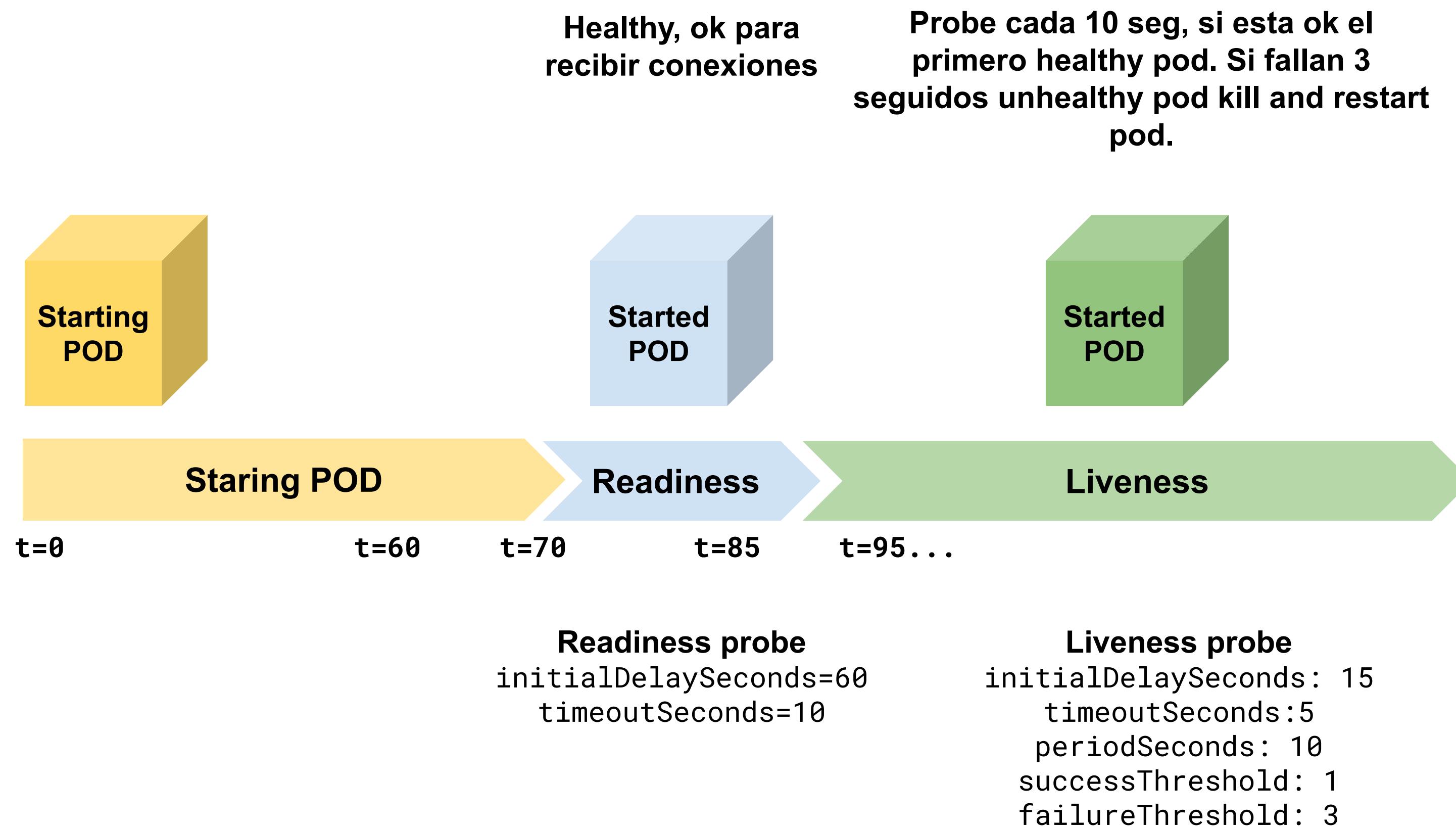
Command line

```
$ oc get dc -o yaml  
$ oc explain dc.spec.strategy  
$ oc get all -l app=app_name  
$ oc get pods -w
```

OpenShift Deployments

Monitoring Application Health

OpenShift ofrece dos mecanismos para monitorear el estado de las aplicaciones



OpenShift Deployments

Monitoring Application Health

HTTP Check

```
...  
readinessProbe:  
  httpGet:  
    path: /health  
    port: 8080  
  initialDelaySeconds: 15  
  timeoutSeconds: 1  
...
```

Container Execution Checks

```
...  
livenessProbe:  
  exec:  
    command:  
      - cat  
      - /tmp/health  
  initialDelaySeconds: 15  
  timeoutSeconds: 1  
...
```

TCP Socket Checks

```
...  
livenessProbe:  
  tcpSocket:  
    port: 8080  
  initialDelaySeconds: 15  
  timeoutSeconds: 1  
...
```

- **initialDelaySeconds:** Determina cuánto tiempo espera después de que el contenedor comience el proceso de inicio. Mandatorio.
- **timeoutSeconds:** Determina cuánto tiempo espera para que el probe finalice. Si el tiempo excede, el probe es fallido. Mandatorio.
- **periodSeconds:** Frecuencia de chequeo. Default 1. Opcional.
- **successThreshold:** Mínimo número consecutivo de probes ok para ser considerado exitoso, si no se cumplen fallido. Opcional.
- **Failurethreshold:** Mínimo número consecutivos de fallas para ser considerado al probe fallido. Default 3.

OpenShift Deployments

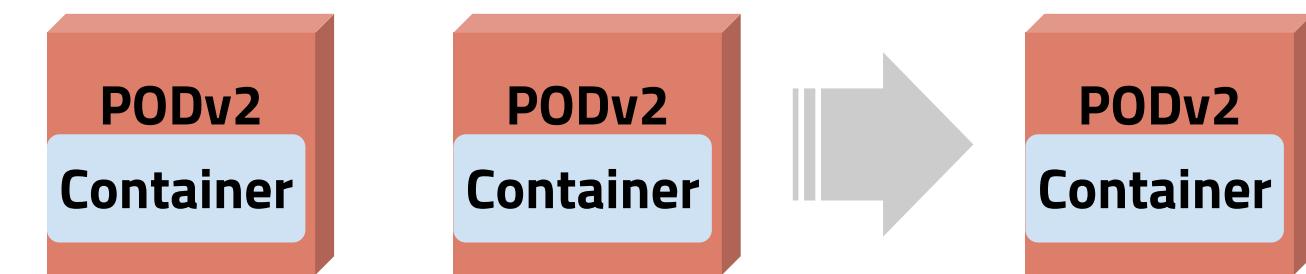
Application Scaling

Application Scaling

El escalado horizontal es la capacidad de, agregar o eliminar, pods al ReplicaSet o Replication Controller

- Usamos el comando `oc scale --replicas=`
- Usamos `--replicas=0` para pausar una aplicación
- El escalado puede ser automático basados en dos tipos de reglas
 - CPU
 - Memoria (tech preview)
 - Otras reglas (threald, connections, bandwith, etc) deben ser disparadas externamente.

```
$ oc autoscaling <dc/rc>/<NAME_RESOURCE> --min <int>  
--max <int> --cpu-percent <int>
```



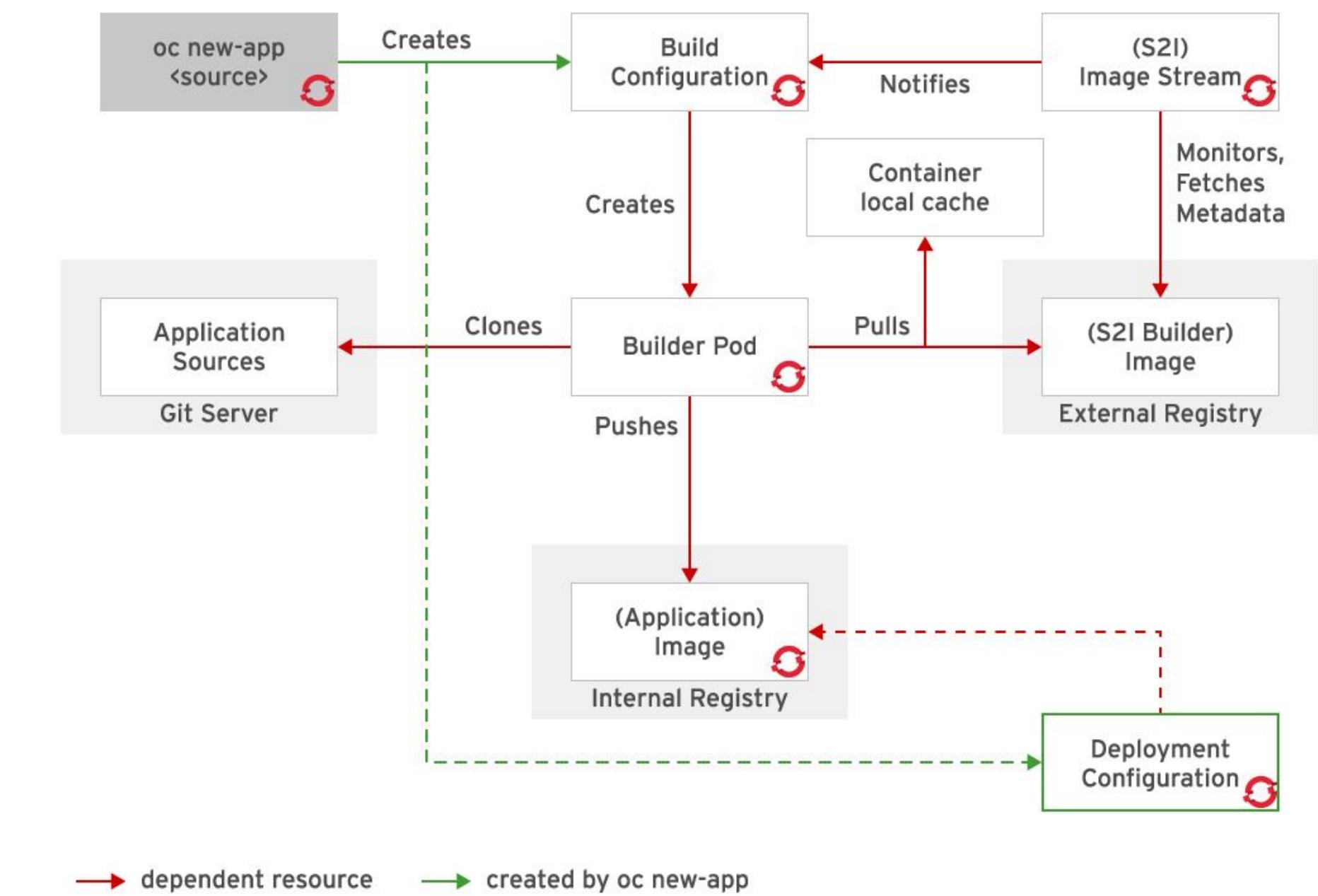
Build and S2I

Semperti

Builds and S2I

Builds (build)

- Es el proceso de transformación un input + parámetros y resultado un objeto
 - Más comúnmente usado como parámetro de entrada el código fuente y el resultado como una imagen.
- Objeto BuildConfig: Definición de un proceso completo de build.
- OpenShift trabaja sobre Kubernetes
 - Crear containers de build imagesUna vez terminado el proceso de build es pusheada la imagen a la registry integrada.



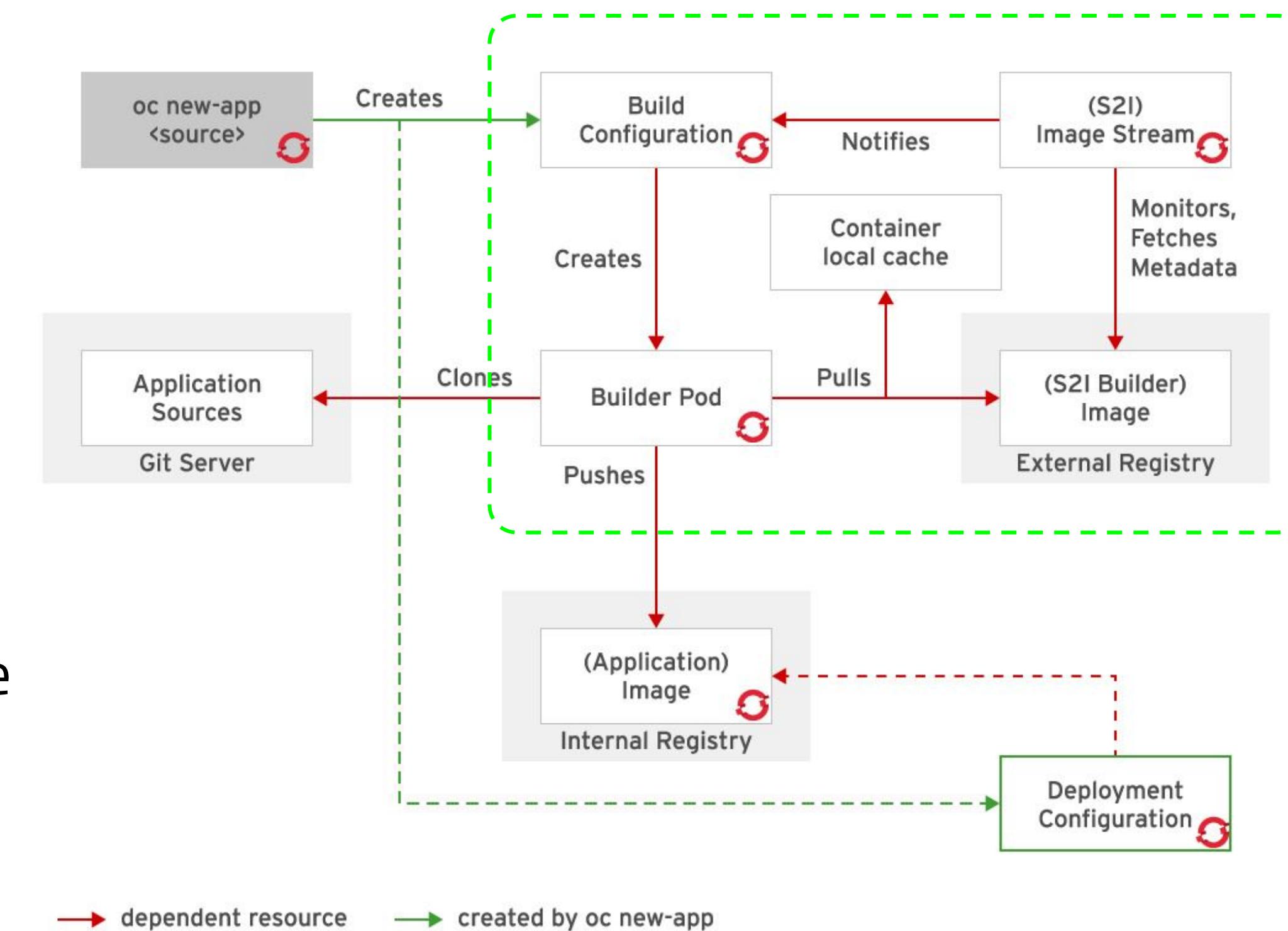
Command line

```
$ oc get buid -o yaml  
$ oc explain build
```

Builds and S2I

Build Strategies

- El sistema de compilación en OpenShift proporciona soporte extensible
 - La API soporta distintas estrategias de build.
- Cuatro estrategias de build:
 - Container build
 - Source-to-Image (S2I) build
 - Custom build
 - Pipeline build (deprecated)
- El objeto resultado del proceso de build depende que build es usado.
 - Container and S2I: Runnable images
 - Custom: Cualquier build image que el autor haya especificado (ej, compilación de un rpm).
 - Pipeline: Ejecución de Jenkins pipelines por Jenkins Pipeline plug-in (deprecated)



Command Line

\$ `oc explain build.spec.strategy`

Builds and S2I

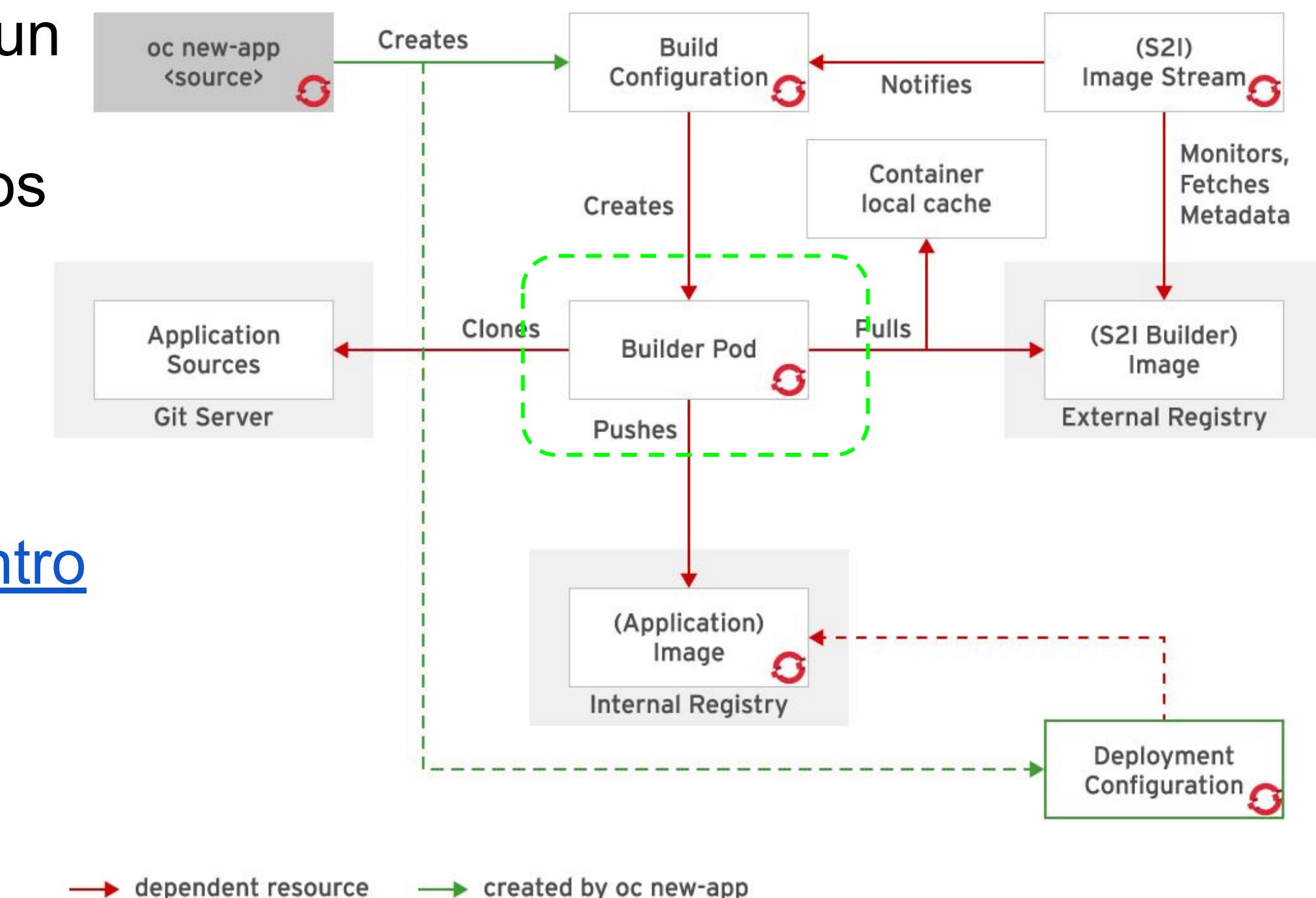
Builds and S2I

Container Build

- El contenedores en la estrategia de build invoca un comando **podman build**
- Espera en un repositorio un Dockerfile y artefactos requeridos para construir la imagen ejecutable.

Referencias

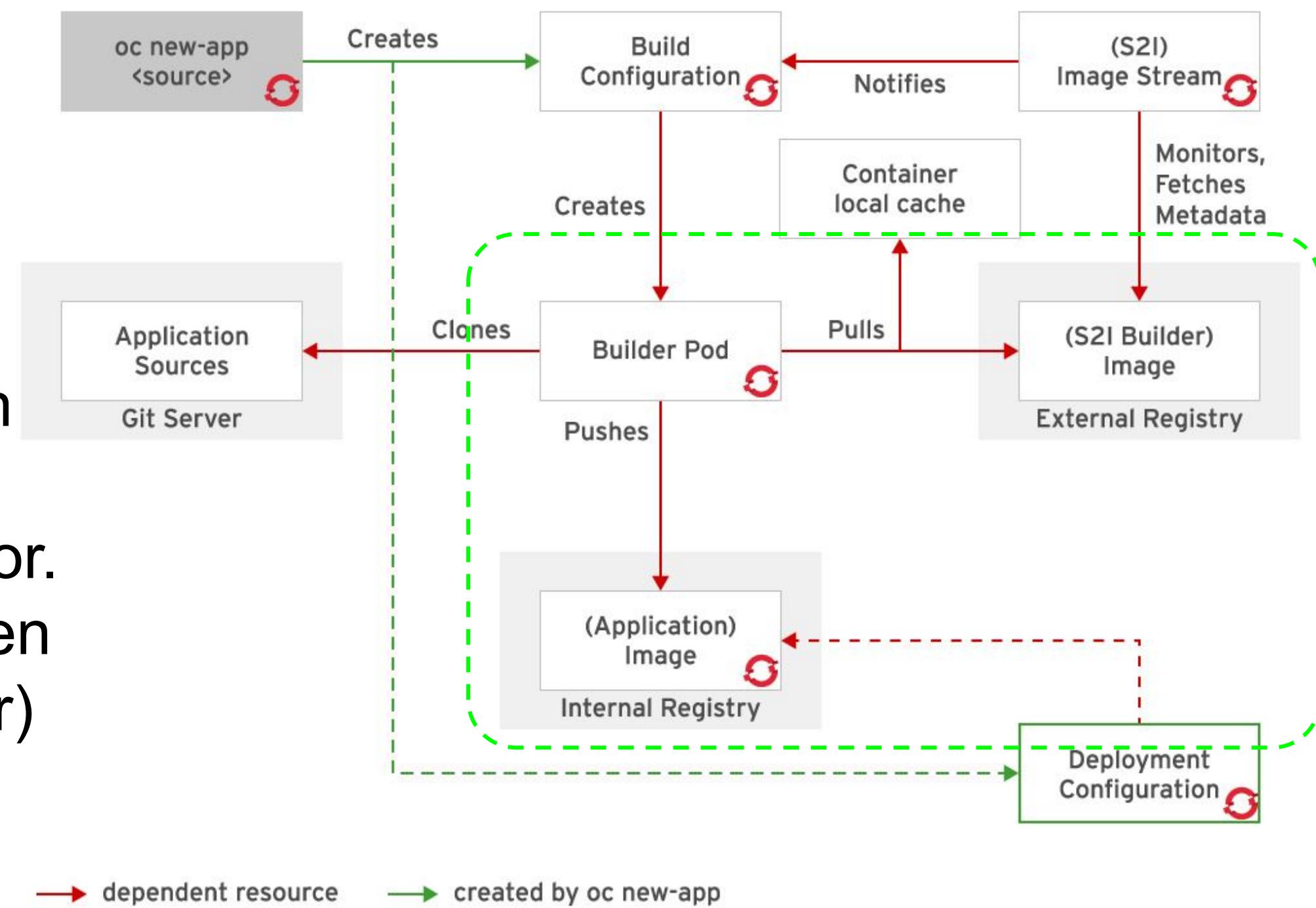
- podman build command: [podman-build docs](#)
podman es funcionalmente equivalente a docker: [Intro to Podman](#)



Builds and S2I

Source-to-Image (S2I) Build

- S2I: Herramienta para crear imágenes de contenedores reproducibles.
 - Produce imágenes listas para poder correr (ready-to-run)
 - Inyecta la aplicación desde el código fuente en la imagen del contenedor de build (s2i builder image) y crea una nueva imagen de contenedor.
 - La nueva imagen (app image) incorpora la imagen base (build image) y el código compilado (builder)
 - Lista para usar con el comando **podman run**
- S2I soporte builds incrementales
 - Reutilizar downloads previos de dependencias, artefactos construidos previamente, etc.



→ dependent resource → created by `oc new-app`

Command Line

```
$ oc explain build  
$ oc explain buildconfig
```

Builds and S2I

Source-to-Image (S2I) Process

El proceso involucra tres componentes fundamentales

- **Application source code**

Código fuente de la aplicación.

- **S2I scripts**

Sets de scripts que son ejecutados para el proceso de build que producirá la imagen que tendremos como resultado.

- **S2I builder image**

Un contenedor de imagen que requiere el ambiente de ejecución runtime para la aplicación. Compiladores, interpretes, scripts, dependencias, etc.

Proceso de S2I

1. Crea un contenedor basado en una imagen de S2I y un tar con el código de aplicación

2. Antes de ejecutar el script de **assemble**, se extrae el código y el contenido se guarda en una ubicación específica --destination o io.openshift.s2i.destination del imagen builder. Default /tmp

3. Si es hay un build incrementar, el script de **assemble** restaura los artefactos de compilación guardados previamente en un archivo tar por la secuencia de comandos **save-artifacts**.

4. El script de assemble crear la aplicación desde el código fuente y los binarios los aloja en el path apropiados para su posterior ejecución.

5. Si es un build incremental, el script save-artifact es ejecutado y gaurda los artefactos de la compilación en un archivo tar.

6. Luego del script **assemble** haya finalizado, es comiteado el contenedor y corrido el script de run para setear el comando **CMD** para la instrucción final.

```
$ skopeo inspect \
docker://myregistry.example.com/rhscl/php-70-rhel7 \
| grep io.openshift.s2i.scripts-url
```

Builds and S2I

Source-to-Image (S2I) Build Advantages

Image flexibility	<ul style="list-style-type: none">Los scripts S2I pueden injectar código de aplicación en la mayoría de las imágenes de contenedores, aprovechando el ecosistema existenteUtiliza tar para injectar la fuente de la aplicación
Speed	<ul style="list-style-type: none">El proceso de ensamblaje realiza una gran cantidad de operaciones complejas sin crear una nueva capa en cada pasoLos scripts S2I reutilizan los artefactos almacenados en la versión anterior de la imagen de la aplicación
Patchability	<ul style="list-style-type: none">Reconstruya la aplicación de manera consistente si la imagen subyacente necesita un parche debido a un problema de seguridad
Operational efficiency	<ul style="list-style-type: none">Restrinja las operaciones de compilación y evite acciones arbitrarias permitidas por DockerfilesEvite abusos accidentales o intencionales del sistema de
Operational security	<ul style="list-style-type: none">S2I restringe las operaciones realizadas como usuario root y puede ejecutar scripts como usuario no rootEvita el riesgo de exponer el sistema host a la escalada de privilegios de root mediante compilaciones de Dockerfile arbitrariasEl proceso de creación de contenedores ya no lo ejecuta el usuario con privilegios de contenedor
Ecosystem	<ul style="list-style-type: none">S2I fomenta el ecosistema compartido de imágenesLe permite aprovechar las mejores prácticas para las aplicaciones
Reproducibility	<ul style="list-style-type: none">Las imágenes producidas pueden incluir todas las entradasVersiones específicas de herramientas de compilación y dependenciasGarantiza que la imagen se pueda reproducir con precisión

Builds and S2I

Custom Build

- **Custom build strategy:** Puede ser reemplazada el image builder por uno personalizado aprovechando el proceso de build.
 - Usando sus propias imágenes builds permite customizar todo el proceso de build
- Custom builder image debe ser una imagen OCI-compliant que tenga embebido el proceso de build en su lógica.
 - Ejemplos: Building RPMs o contenedores de imágenes base
 - Ejemplos de implementaciones de custom builder images:
 - openshift/origin-custom-docker-builder image on [Docker Hub](#)

Command Line

```
$ oc explain build.spec.strategy.customStrategy
```

Builds and S2I

Triggering Builds

Dos tipos de triggers para re-buildear una imagen

- **Image change triggers**

Una imagen dispara la reconstrucción de una nueva imagen de contenedor cuando la imagen padre ha cambiado.

Ejemplo: cambia la imagen base de python o java y la aplicación es re compilada.

- **Configuration change triggers:**

Un cambio de configuración en el template de build dispara un rebuild de la imagen de contenedor.

Command Line

```
$ oc set triggers bc/name --from-image=project/image:tag
```

```
$ oc set triggers bc/name --from-image=project/image:tag --remove
```

Builds and S2I

Webhooks Triggers

Openshift webhook triggers son HTTP API endpoints que permiten comenzar un nuevo build. Usamos webhooks para integrar openshift con un Sistema de Control de Version (VCS)

- GitLab
- GitHub
- Bitbucket

Command Line

```
$ oc set triggers bc/name --from-gitlab  
$ oc set triggers bc/name --from-gitlab --remove
```

Builds and S2I

Post -commit Build Hooks

Openshift provee la funcionalidad de realizar validaciones durante el proceso de build. Post-commit build hooks ejecuta comandos lanzados desde un contenedor temporal antes de ser pusheada la imagen a la registry.

Casos de uso:

- Units Test
- Integrar un build con aplicaciones externas a traves HTTP API.
- Falla el build dependiendo de una validación no funcional
- Envía un email al desarrollador con warning de un nuevo build.

Command Line

```
$ oc set build-hook bc/name --post-commit --command -- bundle exec rake test --verbose
```

```
$ oc set build-hook bc/name --post-commit \  
--script="curl http://api.com/user/${USER}"
```

Builds and S2I

Creating an S2I builder image

Para poder crear un propia imagen de build debemos instalar localmente s2i tools del repo rhel-server-rhscl-7-rpms

```
$ s2i create image_name directory
└── Dockerfile
└── Makefile
└── .s2i
    └── bin
    └── assemble
    └── run
    └── save-artifacts
    └── usage
    └── test
└── run
└── test-app
```

```
$ podman build -t builder_image .
$ s2i build src builder_image tag_name
$ podman run -u 1234 -d -p 8080:8080 nginx-test
```

Import to Openshift

```
$ podman tag s2i-nginx \
myregistry.example.com:5000/s2i-nginx

$ podman push myregistry.example.com:5000/s2i-nginx

$ oc import-image s2i-nginx \
--from myregistry.example.com:5000/s2i-nginx \
--confirm --insecure=true

$ oc new-app --name nginx-test \
s2i-nginx~git_repository
```

Build and S2I

Command Line

Comenzar un proceso de build

```
$ oc start-build BUILD
```

Cancelar un proceso de build

```
$ oc cancel-build BUILD
```

Editar BuildConfig

```
$ oc edit bc BUIL
```

Borrar un BuildConfig o Build

```
$ oc delete bc BUILDCONFIG  
$ oc delete build BUILD
```

Detallar

```
$ oc describe <BUILDCONFIG|BUILD>
```

Logs

```
$ oc logs -f BUILD
```

Prunning

```
$ oc adm prune
```

Image Streams

Image Stream (is)

- Incluye imágenes de contenedores compatibles con el estándar OCI identificadas con etiquetas.
- Presenta una vista virtual única de imágenes relacionadas
 - Similar al repositorio de imágenes de contenedores
- Puede contener imágenes de cualquiera de los siguientes:
 - Repositorio de imágenes en el registro integrado de OpenShift Container Platform
 - Otras secuencias de imágenes
 - Ejemplo: Repositorios de imágenes de contenedores de registros externos
- Nuevas imágenes son asociadas a un imagestream

Command Line

```
$ oc explain images
$ oc explain imagesstream
$ oc start-build --follow bc/BUILDCONFIG_NAME
```

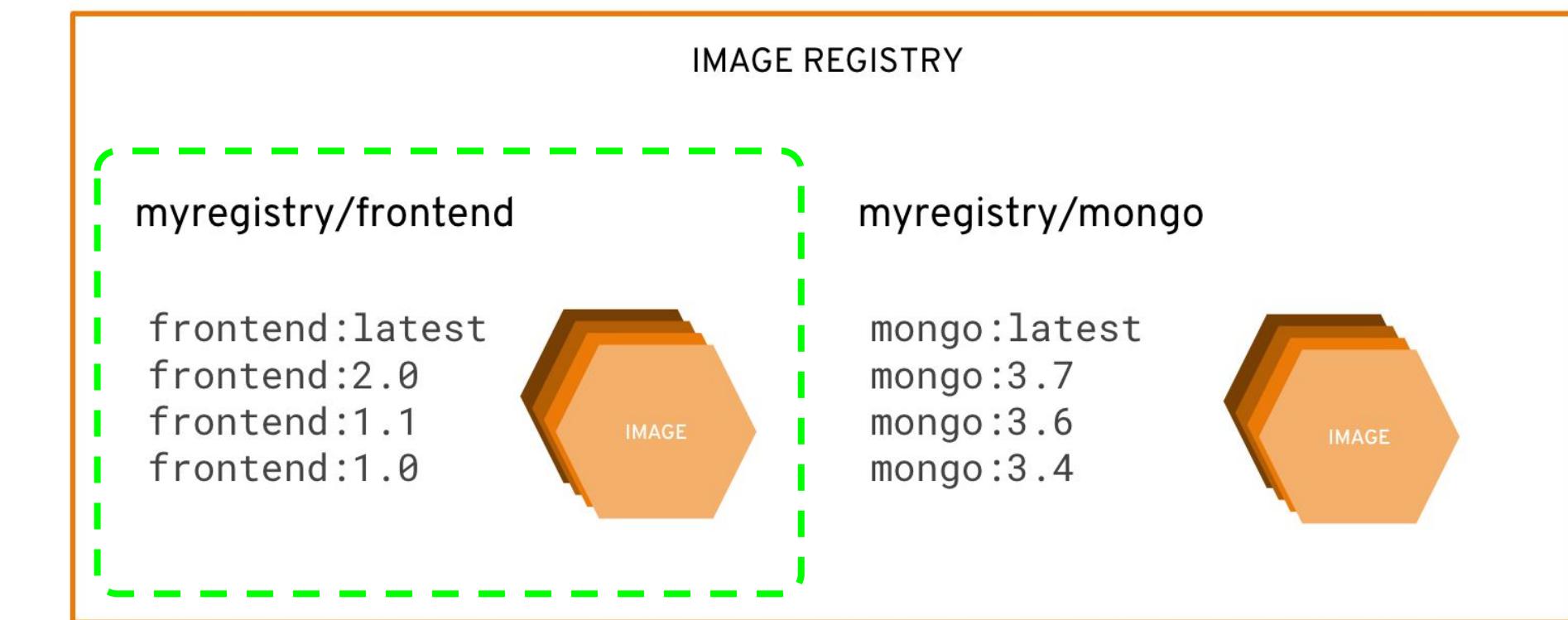


Image Streams

Image Stream and tags

- Builds y Deployments reaccionan a images streams
 - Recibe notificaciones cuando una nueva imagen es agregada
 - Reaccionan generando un nuevo build o deploy.

Una forma de crear un imagestream:

```
$ oc import-image --confirm --from registry.acme.example.com:5000/acme/awesome  
--insecure
```

El número de imagestream predefinidos está en el proyecto openshift como en el proyecto donde se haya Desplegado una aplicación. oc new-app.

Un imagestream puede definir multiples tags.

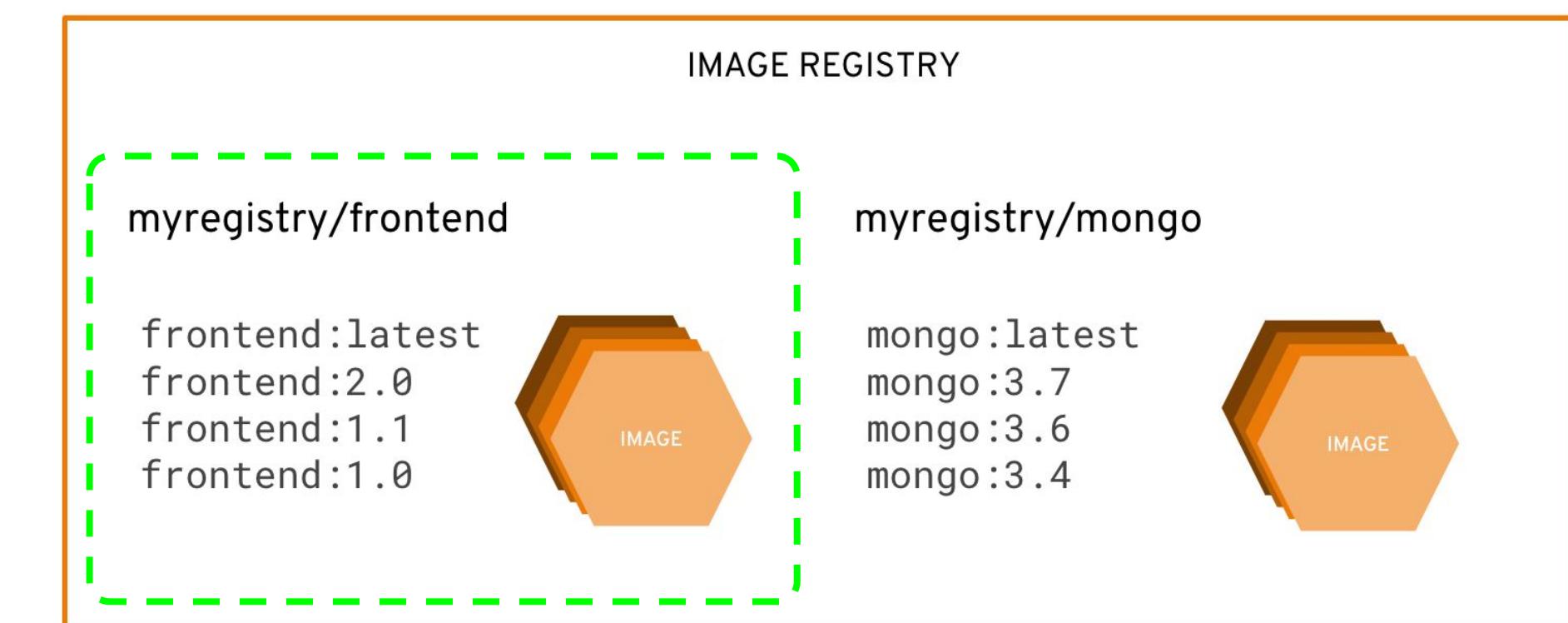


Image Registry

Image Registry y Skopeo

Integrate Image Registry

- Utilizada para alojar todas las imágenes construidas dentro del cluster.
- El proyecto openshift-image-registry aloja el servicio y es desplegado por un Operador.

External Image Registry

- Puede provisionarse una image registry externa
- ```
$ sudo yum install docker-distribution skopeo
```

### Skopeo

Es una herramienta que trabaja bajo el estándar oci para manipular imágenes de contenedores.

```
$ skopeo inspect docker://registry.example.com/myimage
$ skopeo copy --dest-tls-verify=false oci:myimage
docker://registry.example.com/myimage
$ skopeo delete --tls-verify=false docker://registry.example.com/myimage
```

Referencia: <https://github.com/containers/skopeo>

# Image Registry

## Granting External Access to the Registry

Dos roles son los que están definidos para interactuar con el registro

- **system:registry**
  - Este rol permite a los usuarios realizar un pull de imágenes desde la registry interna
- **system:image-builder**
  - Este rol permite a los usuario realizar el push de imágenes a la registry interna
- **system:image-puller**
  - Este rol permite a los service account que corren los pods poder descargar imagenes

Son asignados de la siguiente manera

```
$ oc adm policy add-role-to-user system:registry user_name
$ oc adm policy add-role-to-user system:image-builder user_name
$ oc policy add-role-to-group -n img_project system:image-puller \
system:serviceaccounts:app_project
```

Para poder acceder es necesario autenticarse

```
$ export TOKEN=$(oc whoami -t)
$ podman login -u myuser -p $TOKEN myregistry.example.com
$ skopeo copy --creds=myuser:$TOKEN docker://myregistry.example.com/...
```

# Troubleshooting

## Build and Deploy

- **Copy files**

```
$ oc cp POD_NAME:/path/file /path/local/file
$ oc cp /path/local/file POD_NAME:/path/file
```

- **Remote shell**

```
$ oc rsh POD_NAME
```

- **Remote command**

```
$ oc exec -it POD_NAME /bin/bash
```

- **Port Forwarding**

```
$ oc port-forward POD_NAME 8888:8080
8888 > local port
8080 > container port
```



# Troubleshooting

## Build and Deploy

Son varios los puntos que hay que revisar a la hora de analizar un problema:

- **Estado**

```
$ oc status
$ oc get RESOURCE
```

- **Descripción del recurso**

```
$ oc describe RESOURCE RESOURCE_NAME
$ oc get RESOURCE RESOURCE_NAME -o yaml
```

- **Logs de recurso**

```
$ oc logs RESOURCE
$ oc logs build/BUILD_NAME
```

- **Eventos**

```
$ oc events
```



# Troubleshooting

## Build and Deploy

### Dockerfile SCC como root

```
$ oc create serviceaccount myserviceaccount

$ oc patch dc/demo-app --patch \
'{"spec": {"template": {"spec": {"serviceAccountName":
"myserviceaccount" } } } }'

$ oc adm policy add-scc-to-user anyuid -z
myserviceaccount
```



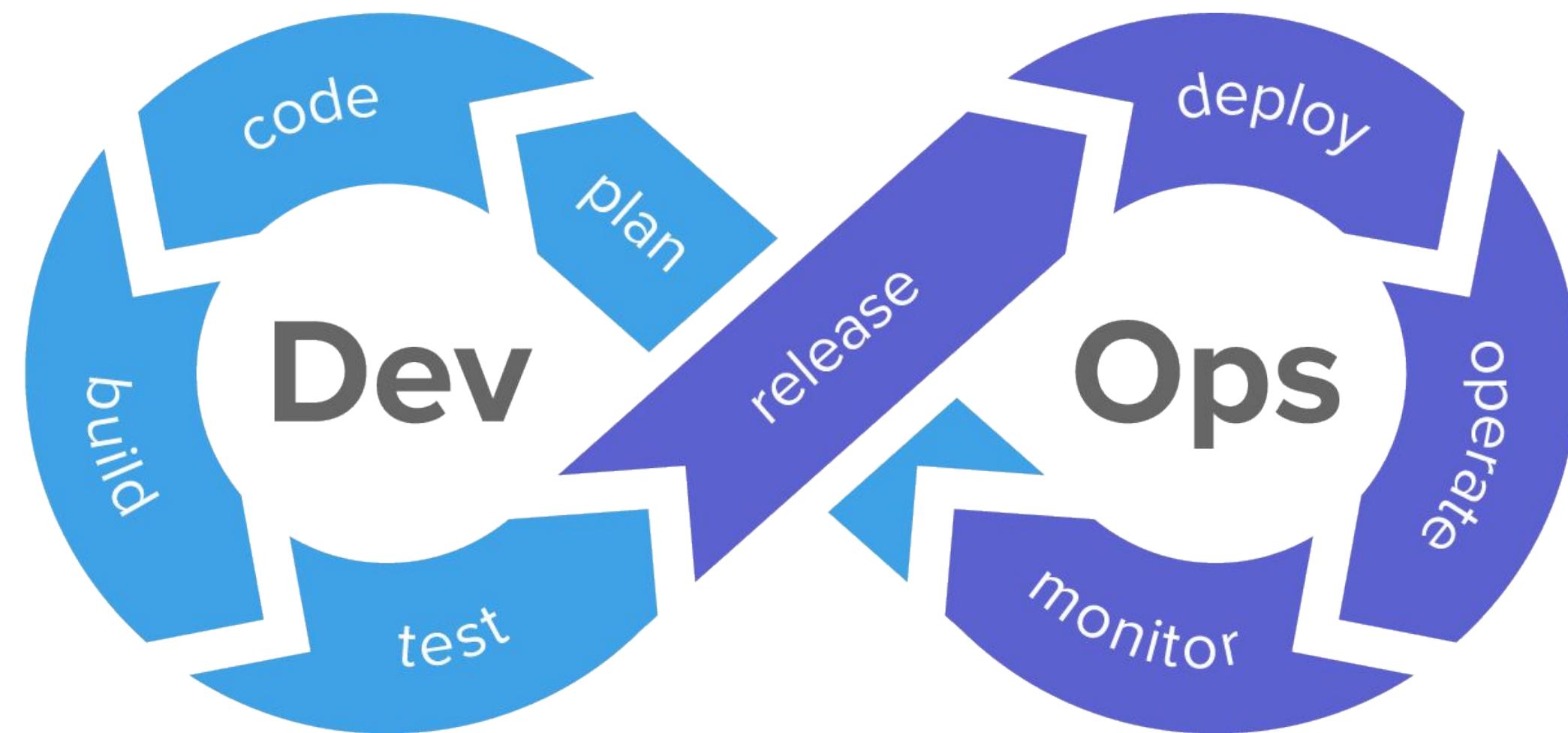
# DevOps en Openshift

*Semperti*

# Build and Deploy (DevOps)

Workflow CI/CD

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

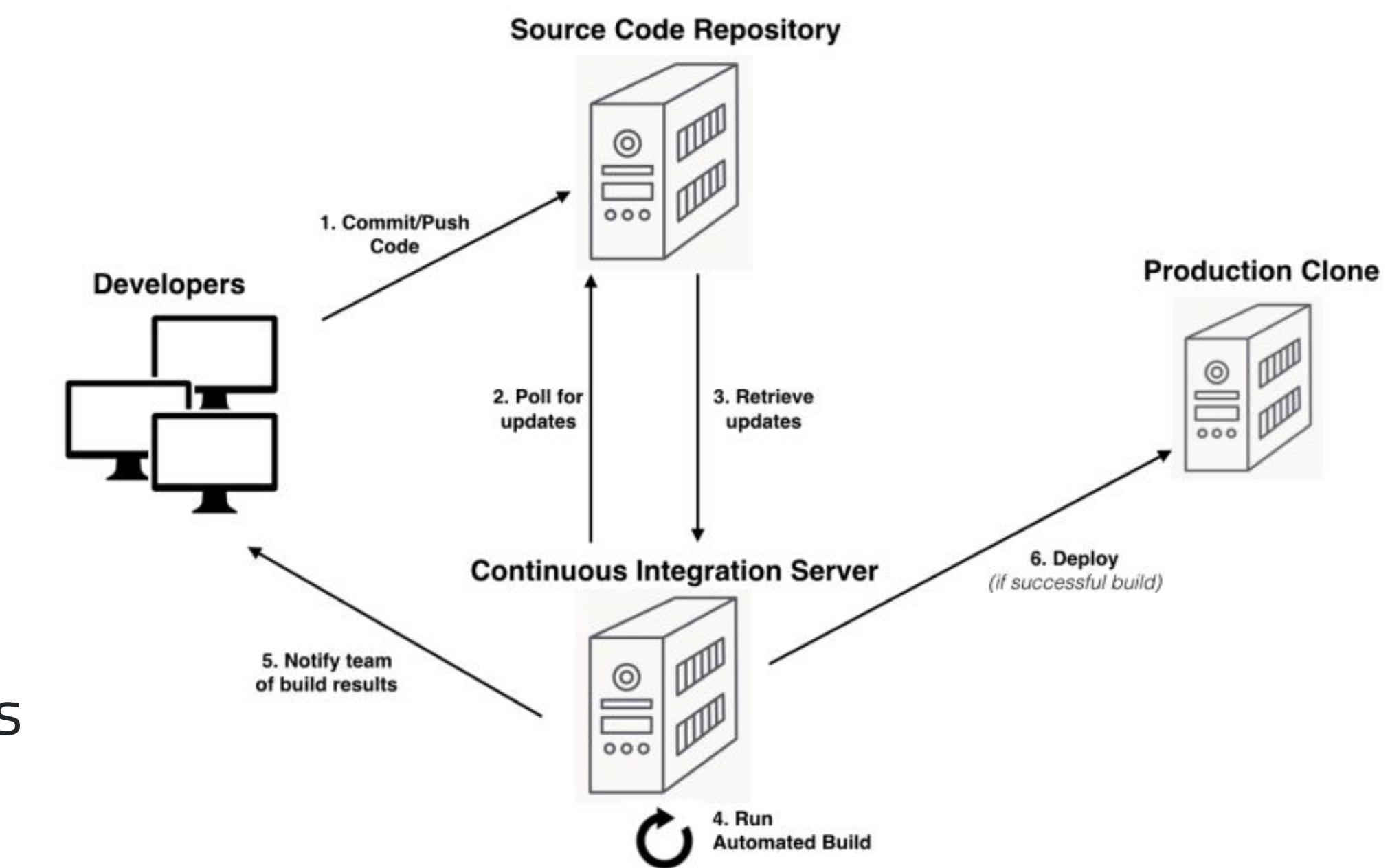


# Build and Deploy (DevOps)

## Workflow CI/CD

### Continuous Integration (CI)

- **Verificar la integridad del build:** Chequear si el código puede ser obtenido 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.

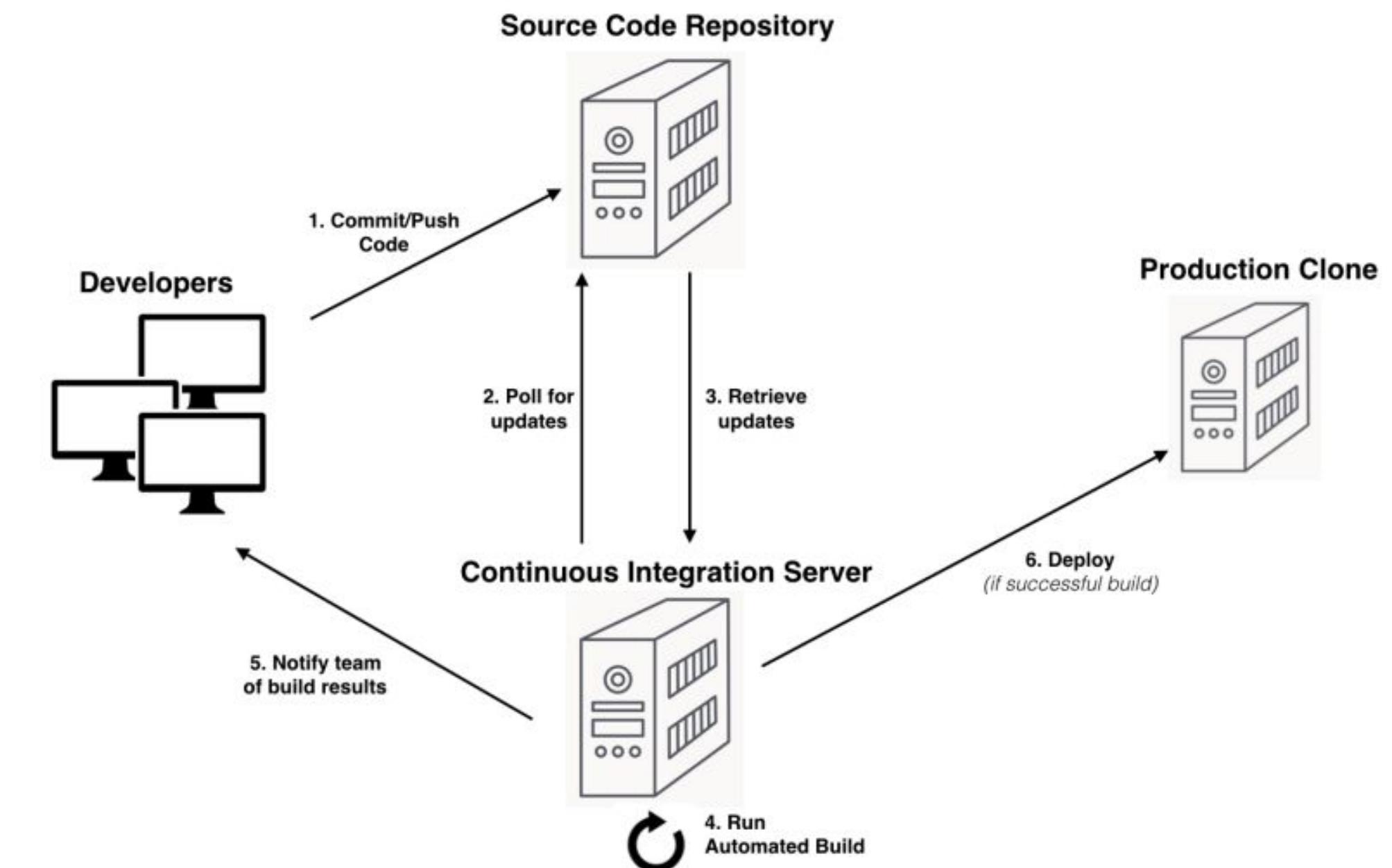


# Build and Deploy (DevOps)

## Workflow CI/CD

¿ Cómo es ejecutado el CI?

- Puede ser ejecutado de manera manual, pero usualmente es ejecutado por una herramienta.
- En DevOps, CI es mandatorio que sea ejecutado por una herramienta (jenkins, tekton, etc).
  - La herramienta corre los scripts de automatización creados por los desarrolladores.
  - Evita la interacción humana durante el proceso de CI.



# Build and Deploy (DevOps)

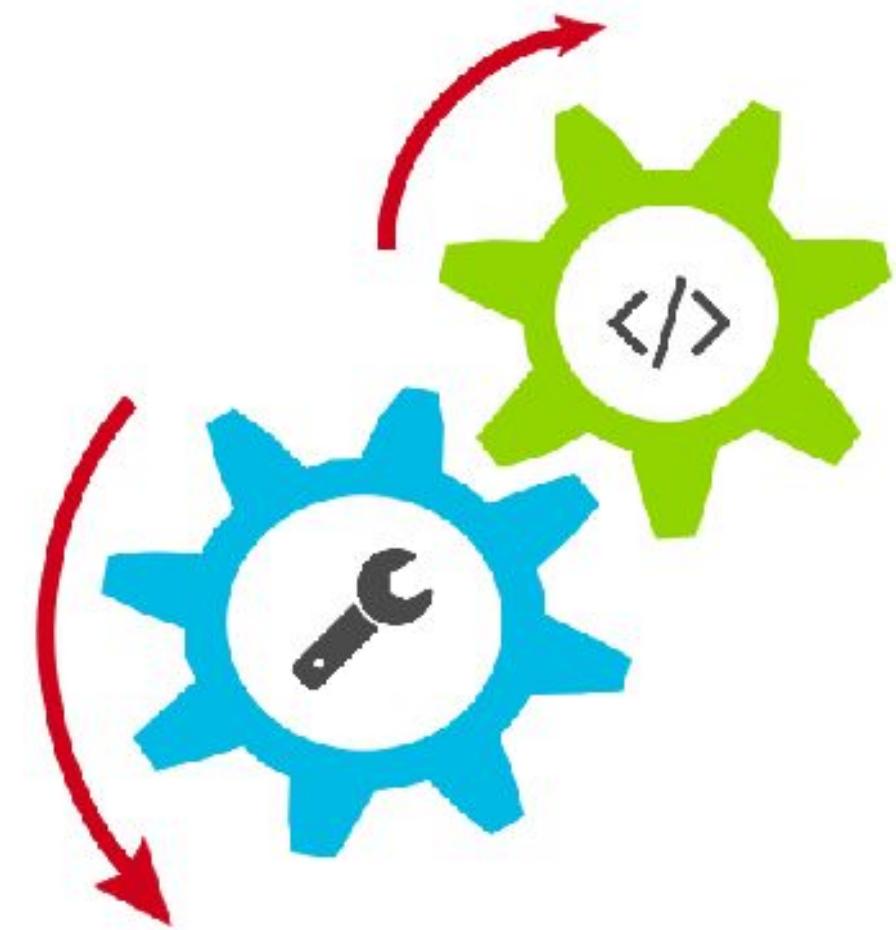
## CI Benefits

|                                                 |                                                                                                                                                                                                                                                                                                                                                                                                         |
|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Rápido feedback                                 | <ul style="list-style-type: none"><li>○ Notificaciones de estado inmediato luego de la construcción del código.</li><li>○ Reduce el tiempo de discovery y fix de posibles fallas.</li></ul>                                                                                                                                                                                                             |
| Reduce riesgos                                  | <ul style="list-style-type: none"><li>○ La integración muchas veces reduce los tiempos del proyecto.</li><li>○ Salud del software medido usando unit test y reportes de código.</li></ul>                                                                                                                                                                                                               |
| Team ownership                                  | <ul style="list-style-type: none"><li>○ Ya no "nosotros contra ellos"</li><li>○ Todos reciben informes periódicos sobre el estado de la compilación</li><li>○ Permite una mayor visibilidad del proyecto, todos pueden detectar tendencias y tomar decisiones efectivas</li><li>○ Crea confianza para agregar funciones al proyecto</li><li>○ Todos a bordo con el estado actual del proyecto</li></ul> |
| Construir software que permita ser desplegable. | <ul style="list-style-type: none"><li>○ El proceso de compilación genera software implementable</li><li>○ Objetivo, crear software que se pueda implementar en cualquier momento</li><li>○ Muchos equipos de desarrollo luchan con esto</li><li>○ No significa que deba implementarse, pero es un buen release candidate.</li></ul>                                                                     |
| Proceso Automatizado                            | <ul style="list-style-type: none"><li>○ Automatizar la construcción ahorra tiempo, costos y esfuerzo</li><li>○ El proceso se ejecuta igual cada vez</li><li>○ Libera a los desarrolladores para que realicen trabajos de mayor valor</li></ul>                                                                                                                                                          |

# Build and Deploy (DevOps)

## CI Best Practices

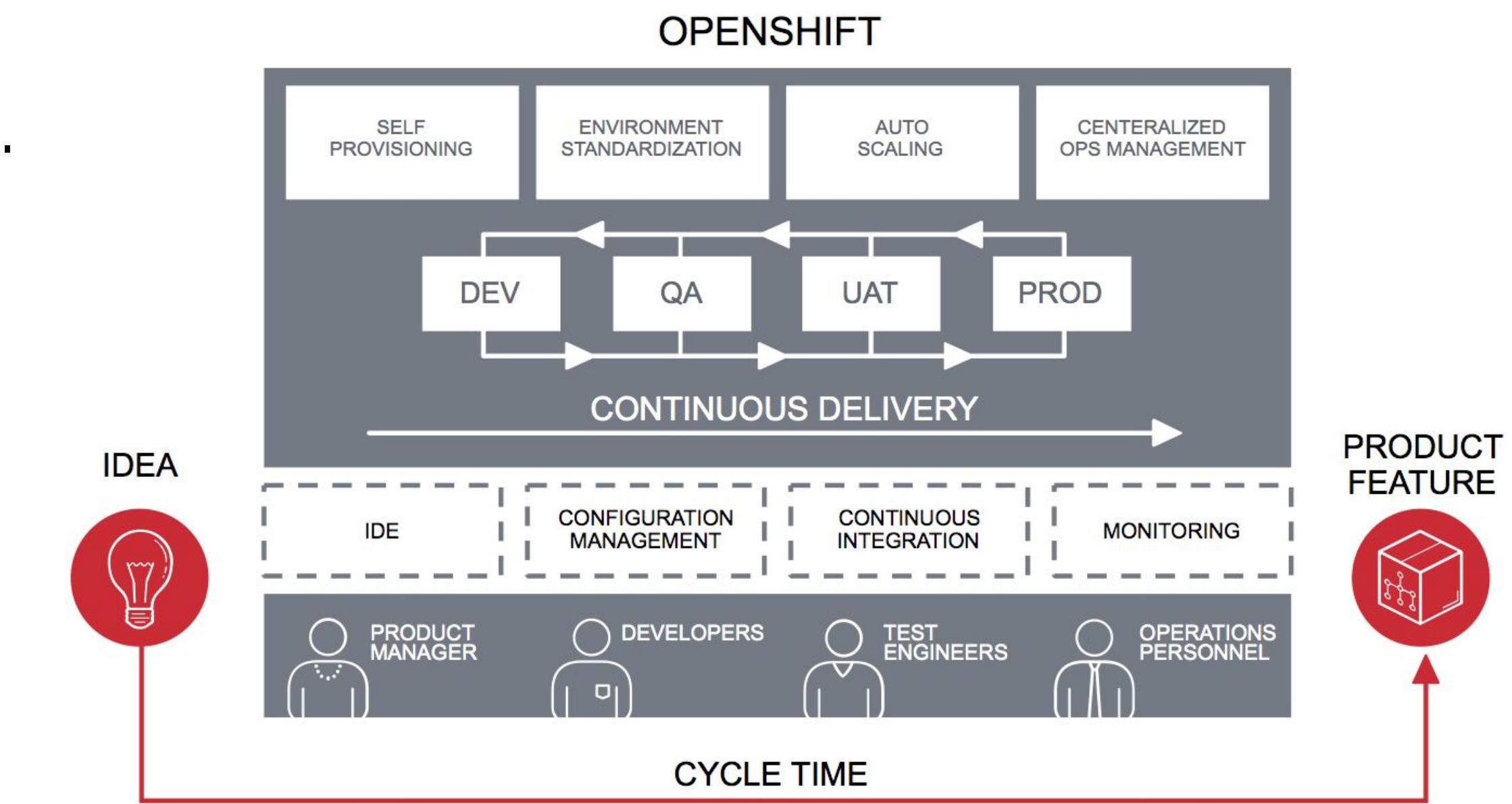
- Mantener el repositorio de código
- Automatizar la compilación
- Make build self-testing
- Asegurar que todos realizar commits todo los días
- Mantener el compilado rápido
- Test en ambientes idénticos a prod
- Permite que sea fácil la obtención de entregables.
- Permite que todos puedan ver como han sido los builds aplicativos



# Build and Deploy (DevOps)

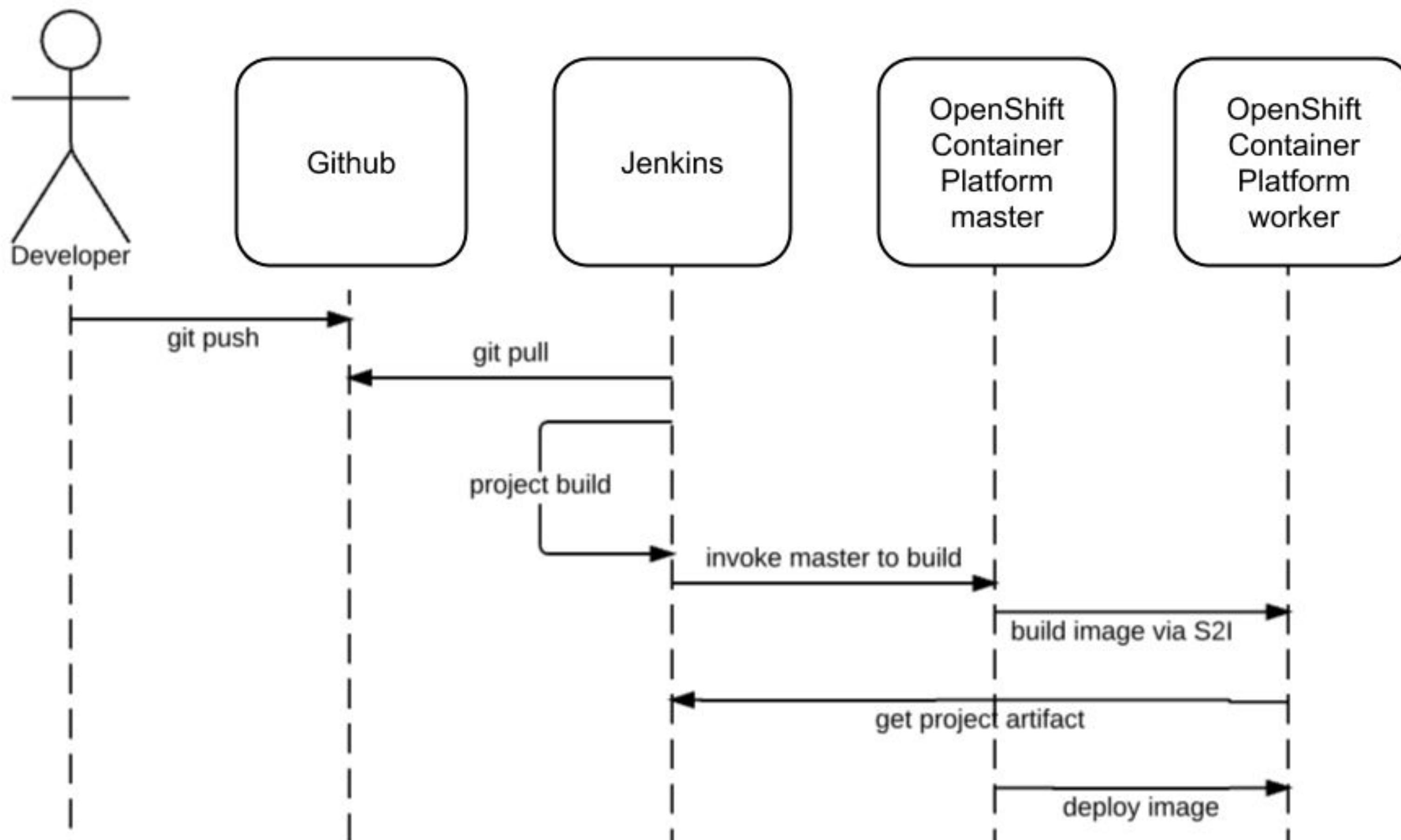
Openshift facilita la cultura DevOps

- Openshift provee “configuration in code”
- Realizado a traves de entornos estandarizados, como Linux containers, automatizando la provisión.
- Integración con Jenkins Server para el proceso de compilación.
- Jenkins provee realizar los testing automatizados, deployments de aplicaciones sobre ambientes de Openshift.
- Estas características acortan el tiempo de ciclo desde la idea hasta la característica del producto implementada



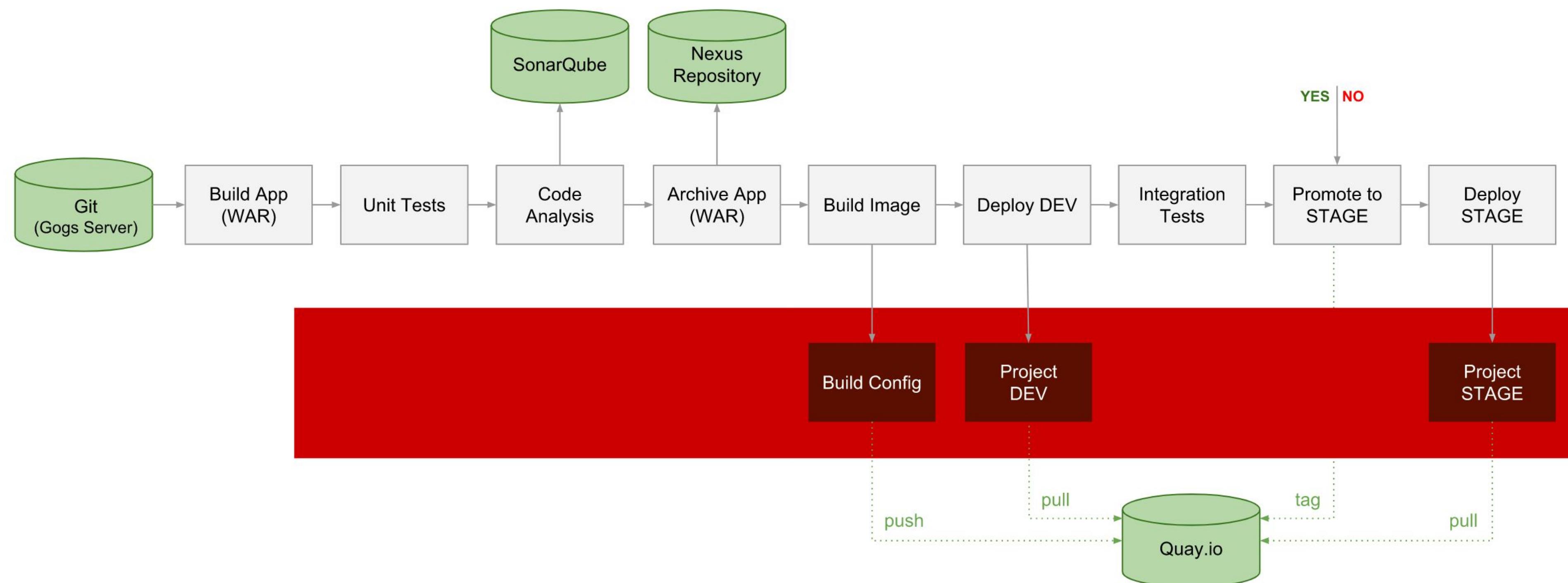
# Build and Deploy (DevOps)

## CI Using OpenShift



# Build and Deploy (DevOps)

CD Using Openshift

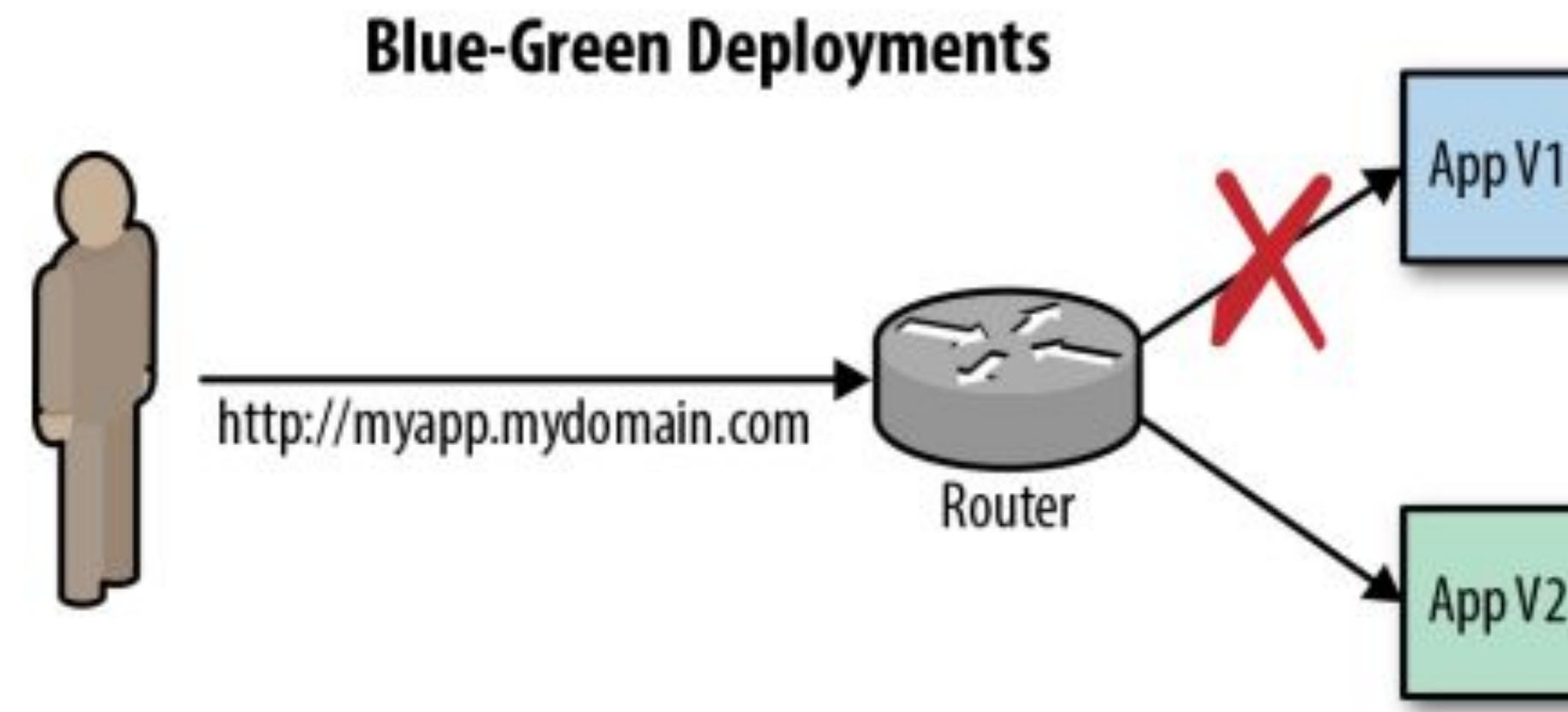


# Estrategias de Despliegue

*Semperti*

# Estrategias de despliegues

## Blue/Green Deployments



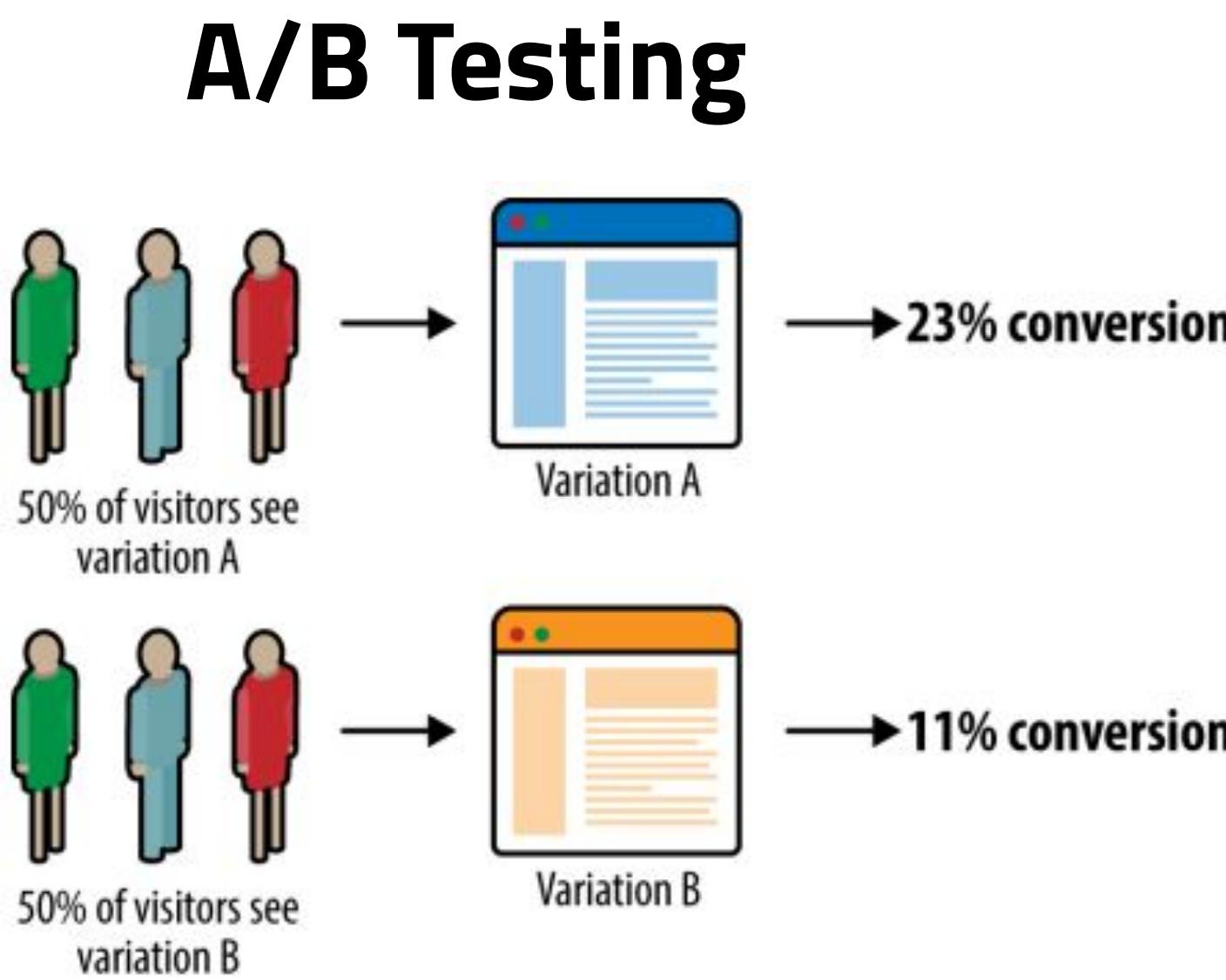
```
$ oc expose service blue --name=bluegreen

switch service to green
$ oc patch route/bluegreen -p '{"spec": {"to": {"name": "green"}}}'

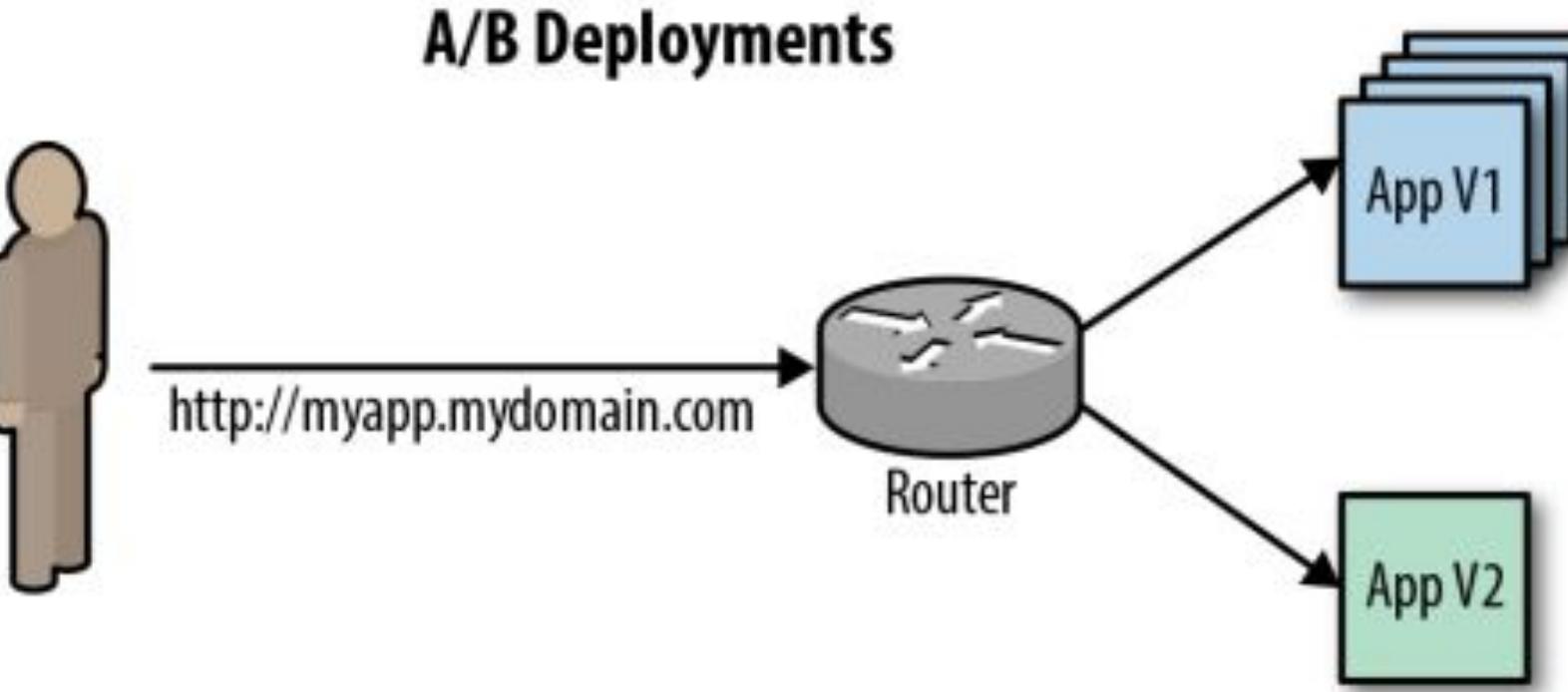
switch back to blue again
$ oc patch route/bluegreen -p '{"spec": {"to": {"name": "blue"}}}'
```

# Estrategias de despliegues

## A/B Deployments



### A/B Deployments



```
$ oc expose service appA --name=appA -l name='appA'
$ oc expose service appB --name=appB -l name='appB'
$ oc expose service ab --name='ab' -l name='ab'
$ oc annotate route/ab haproxy.router.openshift.io/balance=roundrobin
$ oc set route-backends ab appA=100 appB=0
$ oc set route-backends ab appA=80 appB=20
```

# Estrategias de despliegues

## Canary Deployments

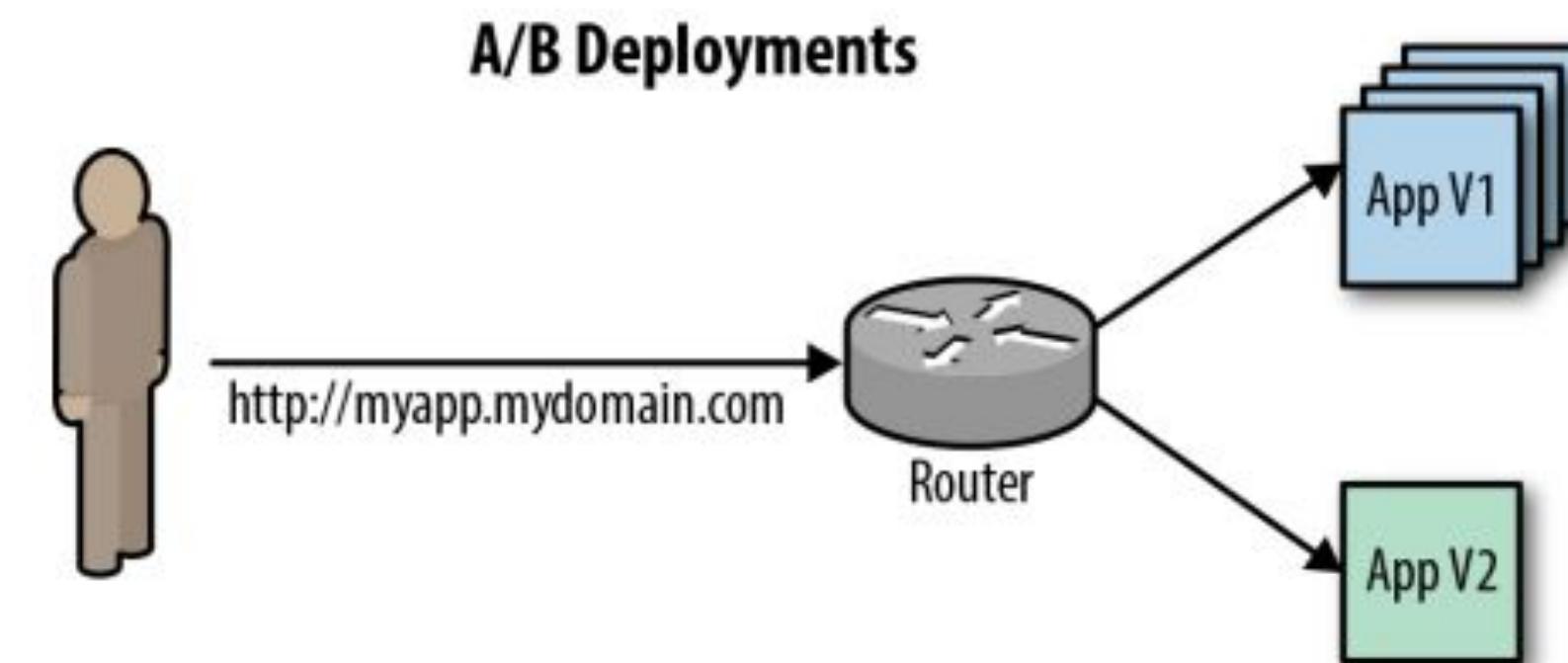
Es una variante de A/B más sofisticada. Mismo pre-seteo iniciar que A/B.

```
$ oc expose service appA --name=appA -l name='appA'
$ oc expose service appB --name=appB -l name='appB'
$ oc expose service ab --name='ab' -l name='ab'
$ oc annotate route/ab haproxy.router.openshift.io/balance=roundrobin
$ oc set route-backends ab appA=100 appB=0
$ oc set route-backends ab appA=80 appB=20
```

**!!!En pod router y como cluster-admin seleccionar qué equipo podrá ver que servicio!!!**

### frontend public

```
Custom acl
block users not in 192.168.137.0/24 network from accessing city
acl network_allowed src 192.168.137.0/24
acl host_city hdr(host) -i city-cotd.192.168.137.3.xip.io
acl restricted_page path_beg /
http-request deny if restricted_page host_city !network_allowed
```



# Templates

*Semperti*

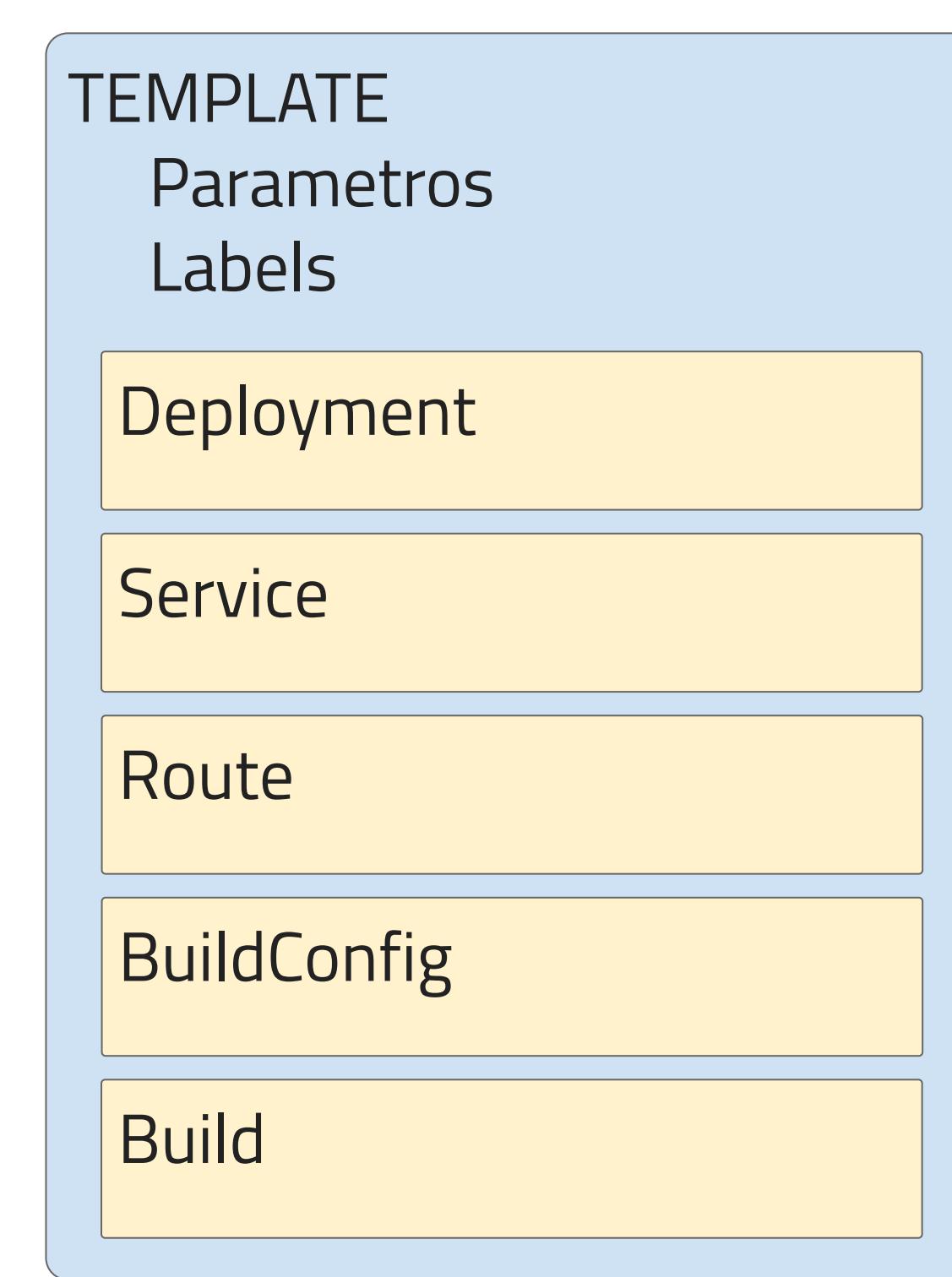
# Template

## Overview

- Describe un set de recursos que pueden personalizar, procesar y procesar para producir configuración.
- Lista parametrizada para crear recursos:
  - Services
  - Pods
  - Routes
  - Build configurations
- Define un set de labels para aplicar a todos los recursos creados por un template

### Para que lo usamos?

- Crear instant apps y desplegar aplicaciones para desarrolladores o clientes.
- Se pueden usar variables pre seteadas o valores random, por ejemplo para password.



# Template

## Template Service Broker (TSB)

- TSB permite visualizar el service catalog en Templates Openshift
  - Muestras por defecto Instant App y Quickstart templates
  - Puede disponibilizar cualquier template que exista, por ejemplo:
    - Red Hat
    - Cluster Administrator, developer, user
    - Third-Party vendor
- Por default TSB muestra objetos globales disponibles en el proyecto openshift
  - Puede ser configurado por el cluster admin para que sea visualizado por cualquier proyecto.

# Template

## Template element

- **Name:** Nombre del template
- **Description:** Opcional, descripción del template
- **Parameters:** lista de parametros utilizado, password, username, etc
- **Labels:** Lista de labels para aplicar a los recursos creados por el template
- **Object:** Lista de recursos creados: pods, services, routes, etc.

```
apiVersion: template.openshift.io/v1
kind: Template
labels:
 app: blog
metadata:
 annotations:
 description: Template para despliegue de aplicaciones
 Openshift Blog.
 name: blog
 namespace: openshift
objects:
- apiVersion: v1
 kind: Secret
...
- apiVersion: v1
 kind: ImageStream
...
parameters:
- description: La URL del repositorio de codigo.
 displayName: Git Repository URL
 name: SOURCE_REPOSITORY_URL
 required: true
 value: https://github.com/semperti/blog-py
```

# Template

## Labels y Objetos

### Labels

- Usado para administrar los recursos generados
- Aplica a todos los recurso que son generados desde el template
- Usado para organizar, agrupar, seleccionar objetos y recursos.
- Recursos y pods “taggeados” con labels
- Permite a los servicios y replication controllers:
  - Indicar que pod está relacionado a:
  - Referencias de grupos de pods
  - Trata los pods con diferentes contenedores de Docker como entidades similares

### Objetos

- Puede ser cualquier recurso que necesite ser creado por el template
- Ejemplo: Pods, services, replication controllers, routes
- Otros objetos puede ser build config, images streams.

# Template

## Parametros

- Comparten valores de configuración entre diferentes objetos.
- Ejemplo: database username, password, o port qué necesitan el pod de fron para poder conectarse a la base de dato.
- Los valores pueden ser estáticos o generados por el template.
- El template puede definir valores de parámetros que son tomados por defecto.
- Puede referenciarse en cualquier campo de texto en el campo de lista de objetos
- Ejemplo:
- Para especificar valores generados, es posible setear una expresión de valores específicos.  
parameters:
  - name: PASSWORD
  - description: "The random user password"
  - generate: expression
  - from: "[a-zA-Z0-9]{12}"

# Template

## Uploading Template

- Puede crearse configuración desde un template usando la CLI o la consola administrativa.
  - Para usar la web console, template tiene que existir en un proyecto o global template library.
- Puede crearse desde un archivo JSON o YAML, el upload se realizar via CLI pasando como parámetro el archivo.

```
$ oc create -f <filename>
```
- Para subir un template en un proyecto diferente usar -n y el nombre del proyecto.

```
$ oc create -f <filename> -n <project>
```
- Una vez subido está disponible para poder ser usado desde la consola web o la línea de comandos.

# Template

## Generating a Configuration

- oc process examina el template, genera los parametros, la salida es la configuración en formato JSON
  - La configuración es creada con oc create -f

```
$ oc process -f <filename>
```
- Para sobreescibir parámetros definidos en el JSON/YAML file, agregar -v opción y parámetro
  - Ejemplo: Sobreescibir ADMIN\_USERNAME and MYSQL\_DATABASE para crear configuración con variables de entorno personalizadas:

```
$ oc process -f my_template_file.json -v ADMIN_USERNAME=root,MYSQL_DATABASE=admin
```

# Template

## Creating an Application From a Template

- `oc new-app` puede instanciar un template almacenado o un archivo template
  - Para instanciar un template almacenado hay que especificar el nombre y los argumentos
  - Para instanciar un template almacenado hay que especificar el nombre del archivo y los argumentos

Crear aplicaciones desde un template almacenado en Openshift

```
$ oc new-app --template=ruby-helloworld-sample --param=MYSQL_USER=admin
```

Crear aplicaciones desde un template almacenado en un archivo

```
$ oc new-app --file=./example/myapp/template.json --param=MYSQL_USER=admin
```

# Template

## Processing Template Parameters

Puede ser necesario crear templates de manera separada

- El equipo de base de datos crear el deploy del template de la base
- El equipo de desarrollo crear los deploy template del frontend

Tratar de que las dos aplicaciones se conecten de manera conjunta.

- Procesar y crear el template de front-end
- Extraer los valores de mysql credenciales desde el archivo de configuración.
- Procesar y crear el template de base de datos.

Sobreescibir los valores con los valores extraídos desde el archivo de frontend.

# Application Configuration

*Semperti*

# Application Configuration

## Overview

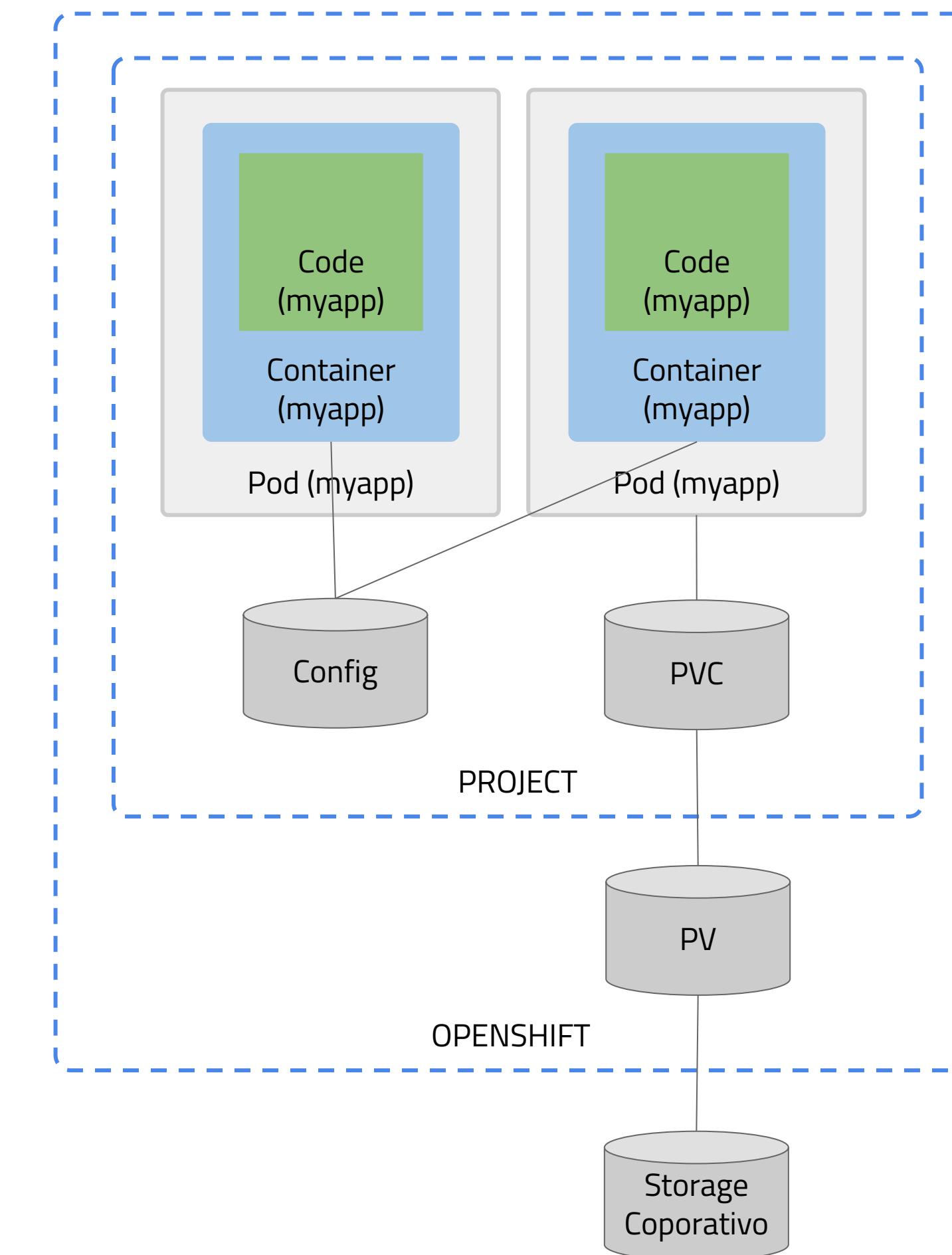
- Application Configuration Data Storage
  - EmptyDir Volumes
  - Persistent Volumes
  - Environment Variables
  - Secrets
  - ConfigMaps
  - Downward API

# Application Configuration

## Application Configuration Data Storage

- Para aplicaciones cloud-based:
  - Nunca almacenar datos de configuración en el código fuente de la aplicación.
  - Usar herramientas de configuración externa en su lugar
- Externalizar el almacenamiento de configuración permite:
  - Desplegar aplicaciones en diferentes ambientes usando el mismo código fuente/binarios.
  - Externalizar los datos de conexión
    - Database location
    - Passwords, secrets
    - Host names, ports

OpenShift® ofrece un set de herramientas para poder externalizar los datos de configuración.



# Application Configuration

## Persistent Application Storage with Volumes

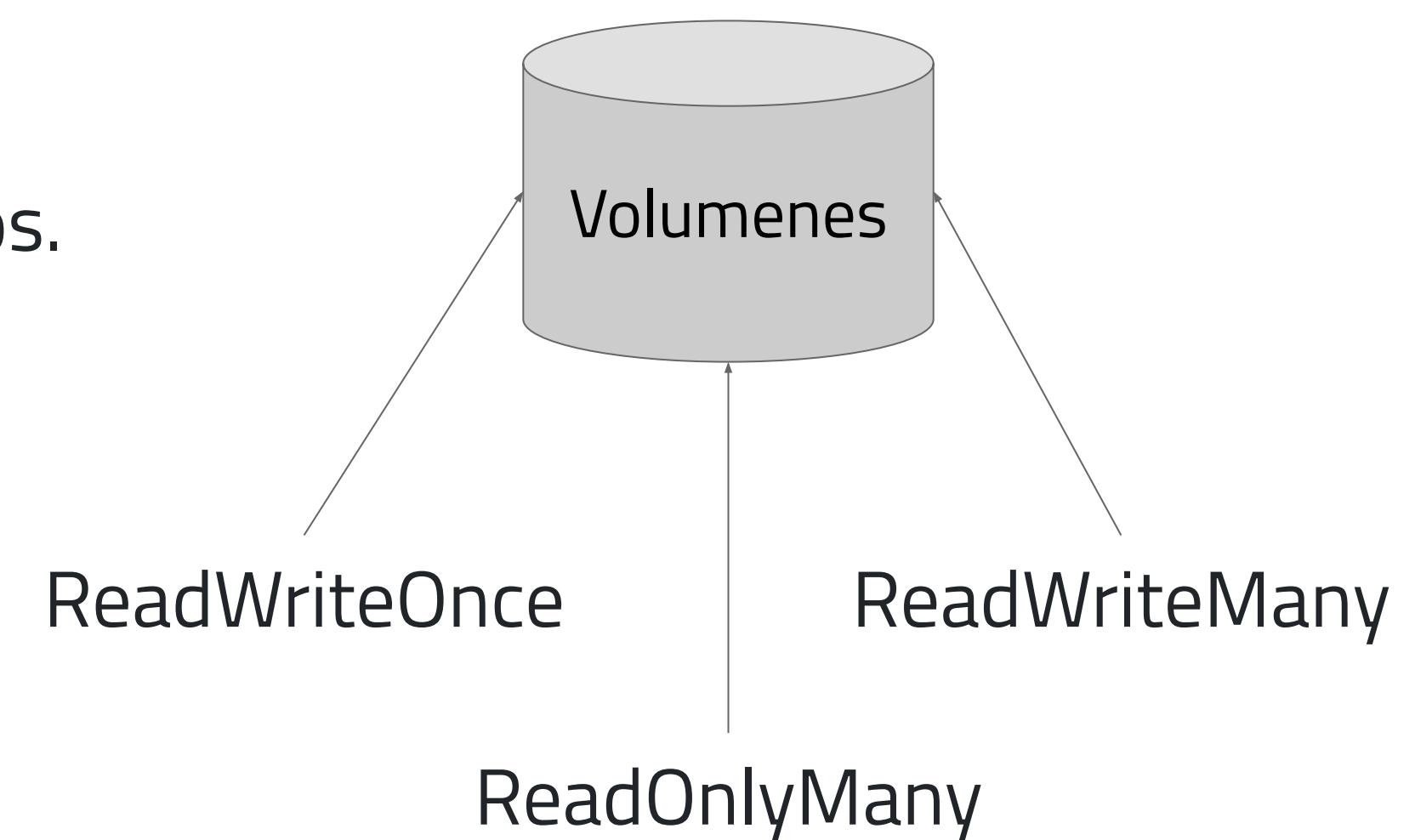
- Contenedores no persisten datos por defecto.
  - El contenido es limpiado cuando es reiniciado, se pierde cualquier almacenamiento en el storage local.
- Muchas aplicaciones necesitan poder almacenar estado.
  - Datos de aplicación: Databases, flat files, etc.
  - Datos de configuración de aplicación.
  - Archivos de logs
- **Volumes:** Punto de montaje disponibles para pods y contenedores
  - Openshift despliega en los pods volúmenes a través de **PersistentVolumes** (PVs) y **PersistentVolumeClaims** (PVCs)
- Openshift puede usar volúmenes para injectar datos de configuración y secretos:
  - Environment variable, secrets, ConfigMaps, Downwards APIs



# Application Configuration

## Volumes

- Volúmenes aparecen en los pods como un file system montado.
  - Accesibles por todos con contenedores en el pod.
  - Los datos en un volumen son preservados a través de los reinicios.
- Volúmenes son representados en Kubernetes mediante el recurso PersistenVolume y creados por el cluster-admin.
- Los PVs son consumidos por los pods creando PVCs.



### Modos de acceso:

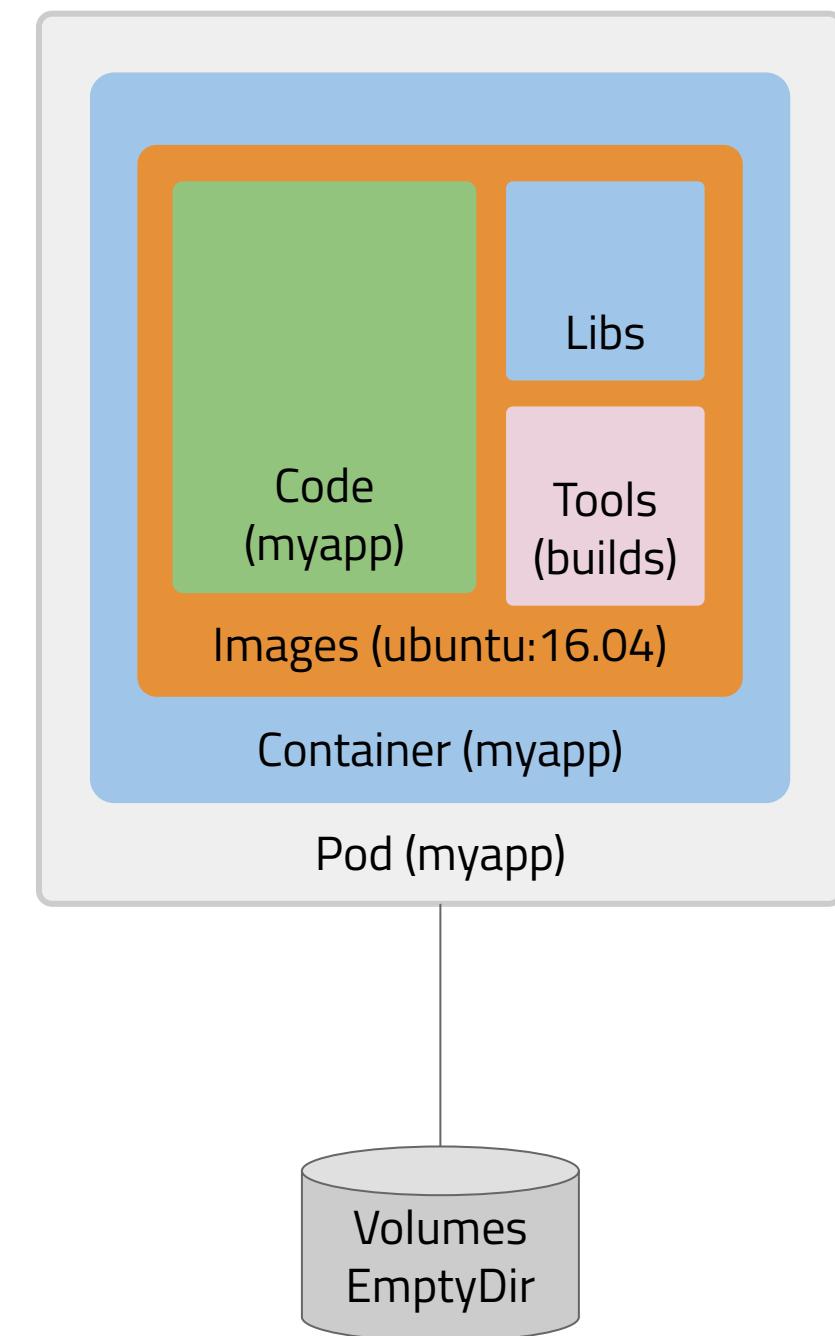
- ReadWriteOnce: El volumen puede ser montado por un pod en RW.
- ReadOnlyMany: El volumen puede ser montado por varios pods en RO
- ReadWriteMany: El volumen puede ser montado por varios pod en RW

# Application Configuration

## emptyDir

- El tipo de volumen mas simple es **emptyDir**
  - Directorio temporal que tiene el mismo ciclo de vida del pod.
  - Creado cuando se crea el pod.
  - Inicialmente vacío
- Contenedores en un pod pueden leer y escribir (RW) el mismo volumen emptyDir
- El volumen emptyDir puede ser montado en diferentes paths en cada container en un pod.
  - Cuando un pod es removido desde el nodo por alguna razón, los datos en el emptyDir es eliminado.

No persisten datos, solamente para datos que tengan utilidad en el momento en el que el pod este siendo ejecutado.



# Application Configuration

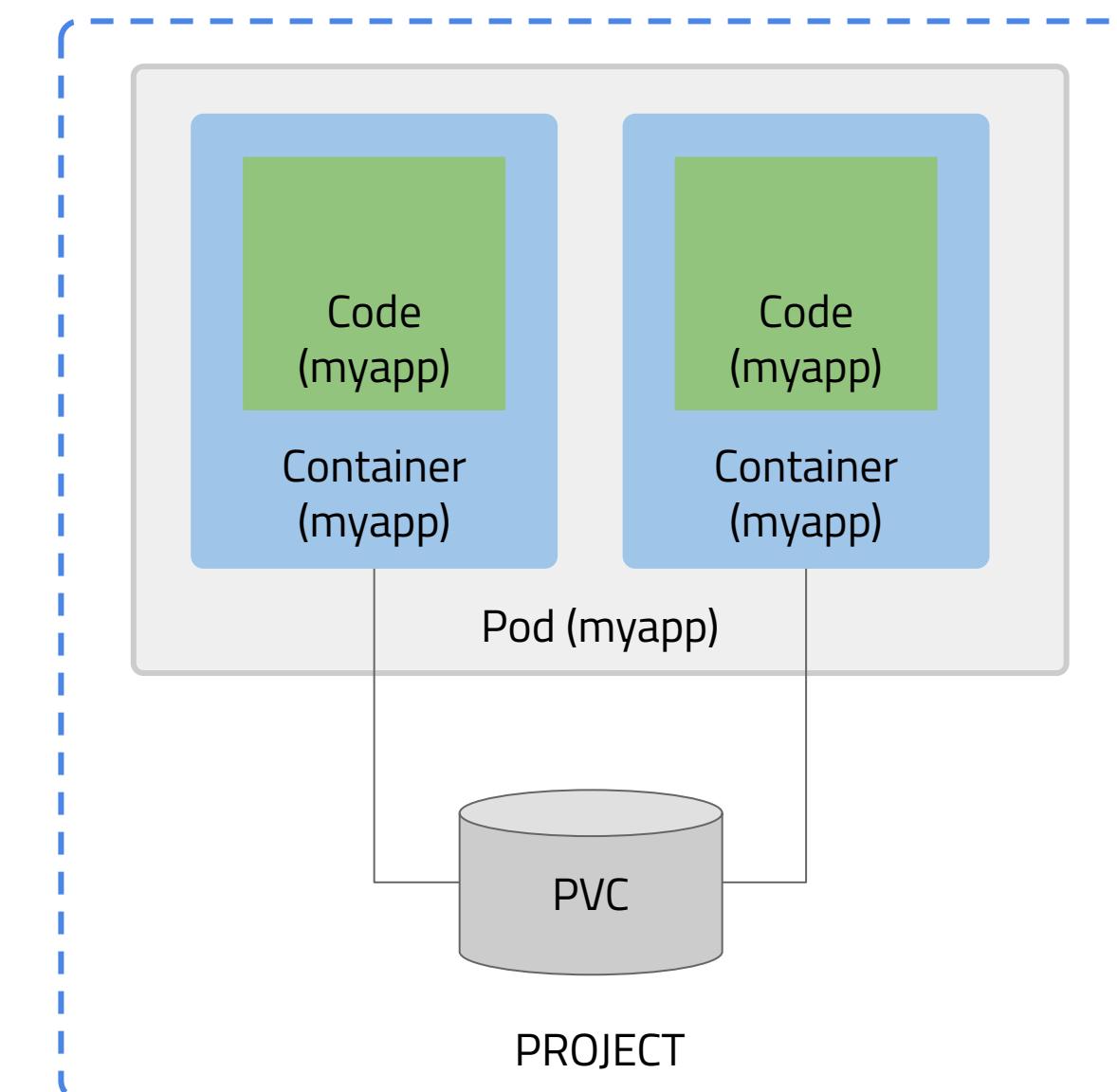
## emptyDir Use Case

- emptyDir es adecuado para:
  - Scratch space.
  - Checkpointing
  - ContA obtiene archivos de un svcX y los aloja en el volumen emptyDir, ContB monta el mismo volumen y procesa los datos obtenidos por ContA.
- Default: emptyDir almacena los datos en el nodo.
  - monta un tmpfs (RAM-backed file system)

Los archivos escritos en este volumen en memoria cuenta como la memoria asignada al pod.

### Command Line

```
$ oc describe pod
$ oc get pods -o yaml
$ oc get deployment
```

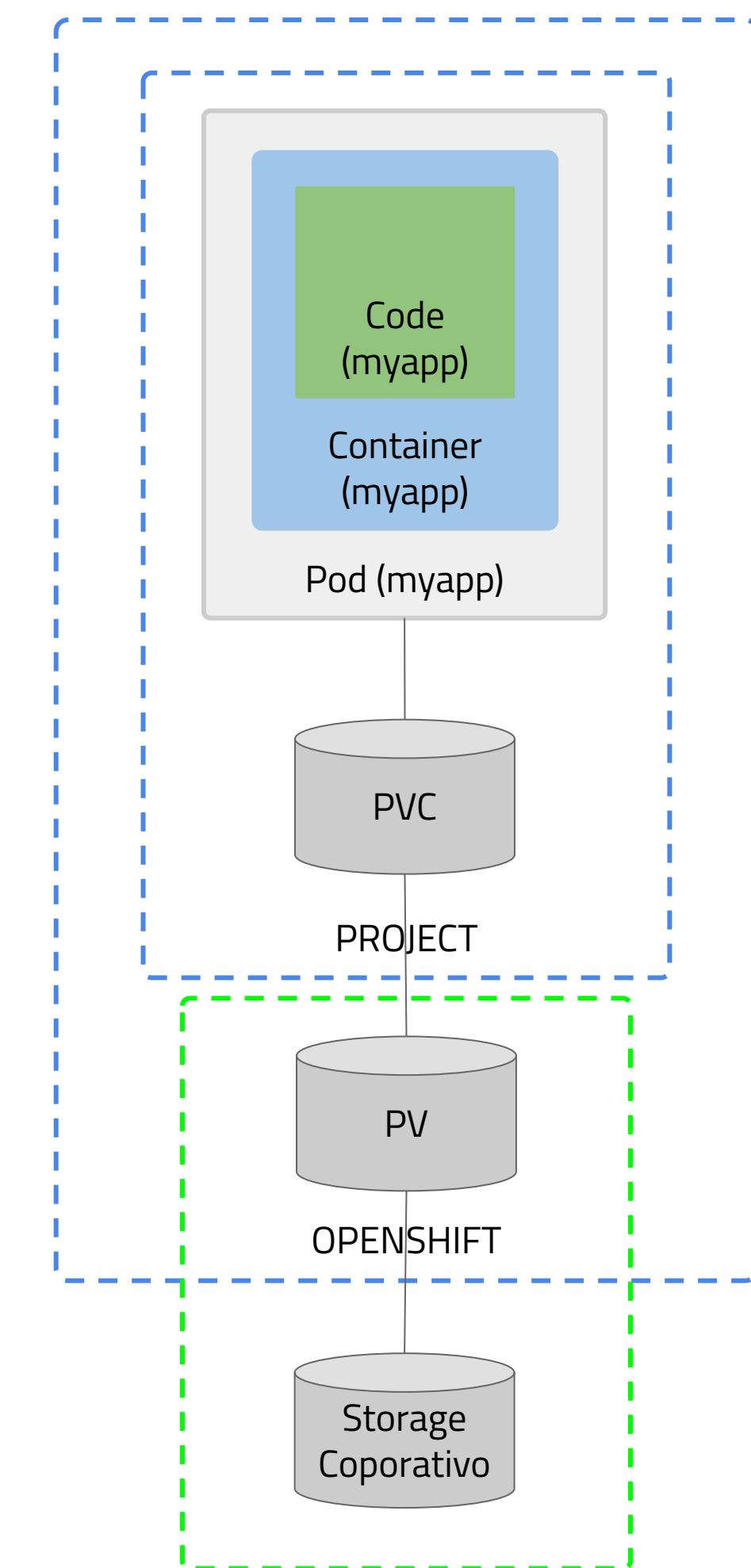


# Application Configuration

## PersistentVolumes

### Persistent Volumes

- Objeto **PersistentVolume**: Recurso de Storage en Openshift/Kubernetes cluster
  - Provisionado por el cluster admin, es quien crea los PersistentVolume
- Creados desde una amplía lista de tecnologías, incluidas:
  - NFS mounts
  - GlusterFS
  - Ceph® RBD
  - AWS Elastic Block Store (EBS)
  - Google GCE
  - OpenStack® Cinder
  - iSCSI
  - Fibre Channel

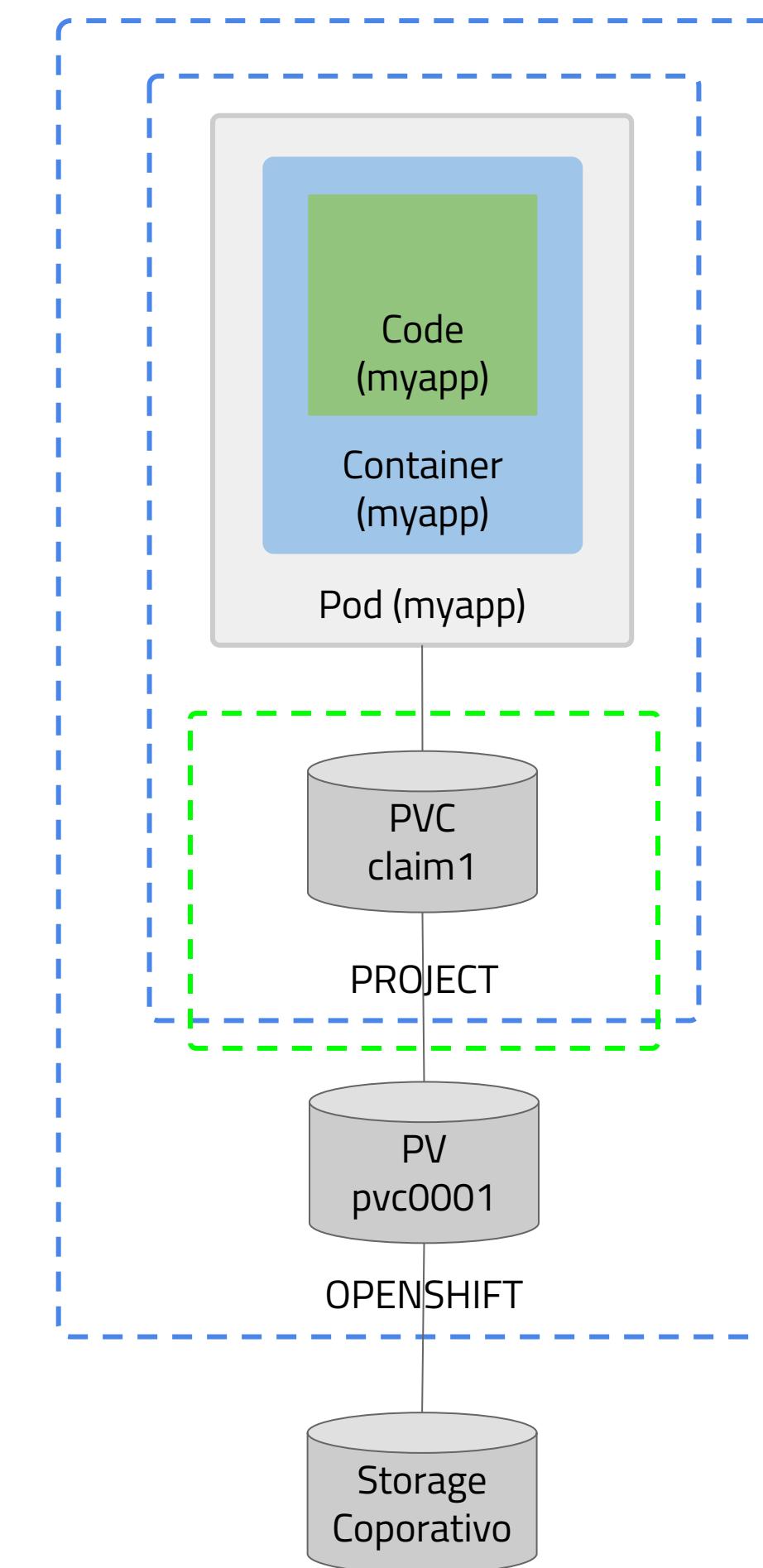


# Application Configuration

## Persistent Volumes Claim

### Persistent Volume Claims

- El desarrollador realizar claims de PersistentVolumes con PersistentVolumeClaim
  - En el claim se especifica el tamaño.
- El claim enlaza con el volumen que matchea con el requerimiento.
  - El desarrollador puede hacer multiples claims diferentes según la quota de recursos se lo permita
- La definición del objeto es:
  - Crear un archivo con la definición: pvc0001.yml
  - Crear el recurso deseado: oc create -f pvc0001.yml
  - También puede usarse la consola web para este fin.



# Application Configuration

## Using PersistentVolumeClaims

### Uso **PersistentVolumeClaims**

- El desarrollador asocia un **PersistentVolumeClaim** en el pod
- Resultado de un claim es un bind de un PVC a un PV disponible o creado automaticamente.

```
$ oc get pvc claim1 -o yaml
apiVersion: "v1"
kind: "PersistentVolumeClaim"
metadata:
 name: "claim1"
spec:
 accessModes:
 - "ReadWriteOnce"
 resources:
 requests:
 storage: "5Gi"
 volumeName: "pv0001"
```

- Definición de un PVC en un Pod:

```
apiVersion: "v1"
kind: "Pod"
metadata:
 name: "mypod"
 labels:
 name: "frontendhttp"
spec:
 containers:
 -
 name: "myfrontend"
 image: "nginx"
 ports:
 -
 containerPort: 80
 name: "http-server"
 volumeMounts:
 -
 mountPath: "/var/www/html"
 name: "pvol"
 volumes:
 -
 name: "pvol"
 persistentVolumeClaim:
 claimName: "claim1"
```

# Environment Variables

*Semperti*

# Environment Variable

## Environment Variables

### Environment Variables

- Seteadas para todo el pods
- Varias maneras de poder setear las.
- Permite el acceso a Secrets y ConfigMaps
- Son leidas a traves del SO, lenguaje de programación vía API.

- Node.JS:

```
var app_user = process.env.APP_USER
```

- Python:

```
os.environ.get('APP_USER')
```

### Scope

- Deployment/DeploymentConfig
- Replication controllers
- Build configurations

### Setting

- Para setear una variable de entorno desde la CLI

```
$ oc set env <object-selection> KEY_1=VAL_1 ...
KEY_N=VAL_N [<set-env-options>]
[<common-options>]
```

- Ejemplo:

```
$ oc set env dc/registry STORAGE=/data
$ oc set env dc/registry --overwrite
STORAGE=/opt
$ env | grep RAILS_ | oc set env rc/r1 -e -
```

### Unsetting

- Para borrar una variable de entorno se usa - antes de key:

```
$ oc set env <object-selection> KEY_1- ...
KEY_N- [<common-options>]
```

- Ejemplo:

```
$ oc set env dc/d1 ENV1- ENV2-
$ oc set env rc --all ENV-
```

# Environment Variable

## Environment Variables

### Listar

- Para listar las variables de entorno en un pod o un pod template:

```
$ oc set env <object-selection> --list
[<common-options>]
```

- Ejemplo: Listas las variables de entorno del pod p1:

```
$ oc set env pod/p1 --list
```

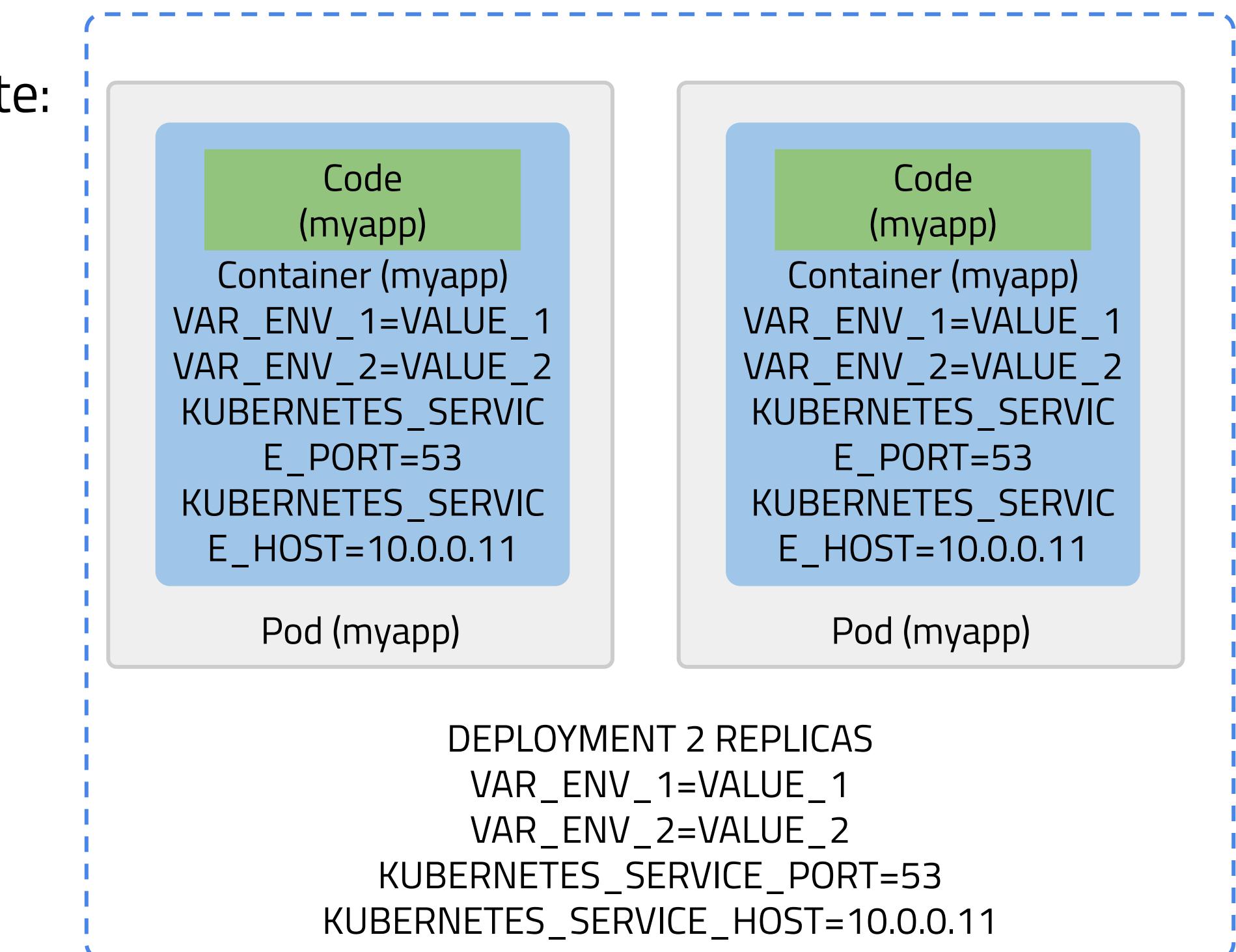
El proceso de service discovery agrega automáticamente las variables de entorno

<SVCNAME>\_SERVICE\_HOST  
<SVCNAME>\_SERVICE\_PORT

- Ejemplo: KUBERNETES\_SERVICE expone el port TCP 53 y la ip 10.0.0.11

- Produce estas variables de entorno

```
KUBERNETES_SERVICE_PORT=53
KUBERNETES_PORT_53_TCP=tcp://10.0.0.11:53
KUBERNETES_SERVICE_HOST=10.0.0.11
```



# Secrets

*Semperti*

# Secrets

## Objects

### Tipo de Objeto

- Provee un mecanismo para mantener la información sensible encodeada como base 64.
  - Passwords
  - Credenciales de registro proviados.
- Desacopla el contenido sensible de la configuración de los pods al ser utilizado.
- Los secretos pueden ser montados en contenedores usando volúmenes.
- Usado por el sistema para mejorar las acciones a favor de los pods (ver secrets creados por default)



### Propiedades

- Referenciado independientemente de la definición.
- Los volúmenes que se montan en los pods son volúmenes temporales de tipo tmpfs.
  - No persisten en filesystem del nodo.
- Compartidos en un mismo namespace o proyecto.
- El secrets se debe crear antes que el pod que lo va a utilizar.
  - Cuando se cambia un secrets, no cambia en el pod hay que recrearlo.

# Secrets

## Creado y uso con Var Env

### Creado de secretos

- Crear un objeto secret con los datos del secreto.
- Actualizar el pod service account para permitir referenciar el secreto.
- Crear un pod que consuma el secreto como:
  - Environment variable
  - Archivo, usando volúmenes
- Ejemplo:

```
echo 'admin' > ./user.txt
echo 'r3dh4t1!' > ./password.txt
oc create secret generic printenv-secret
--from-file=app_user=user.txt
--from-file=app_password=password.txt
```

### Usando variables de entorno

- Agregar el secret al deployment usando el comando oc env:
  - APP\_USER
  - APP\_PASSWORD
- Ejemplo:

```
oc env dc/printenv
--from=secret/printenv-secret
```
- Se puede agregar un prefijo a las variables de un secret con oc env --prefix= command to set:
  - DB\_APP\_USER
  - DB\_APP\_PASSWORD
- Ejemplo:

```
oc env dc/printenv
--from=secret/printenv-secret --prefix=DB_
```

# Secrets

## Uso con Volumenes

### Usando volumenes

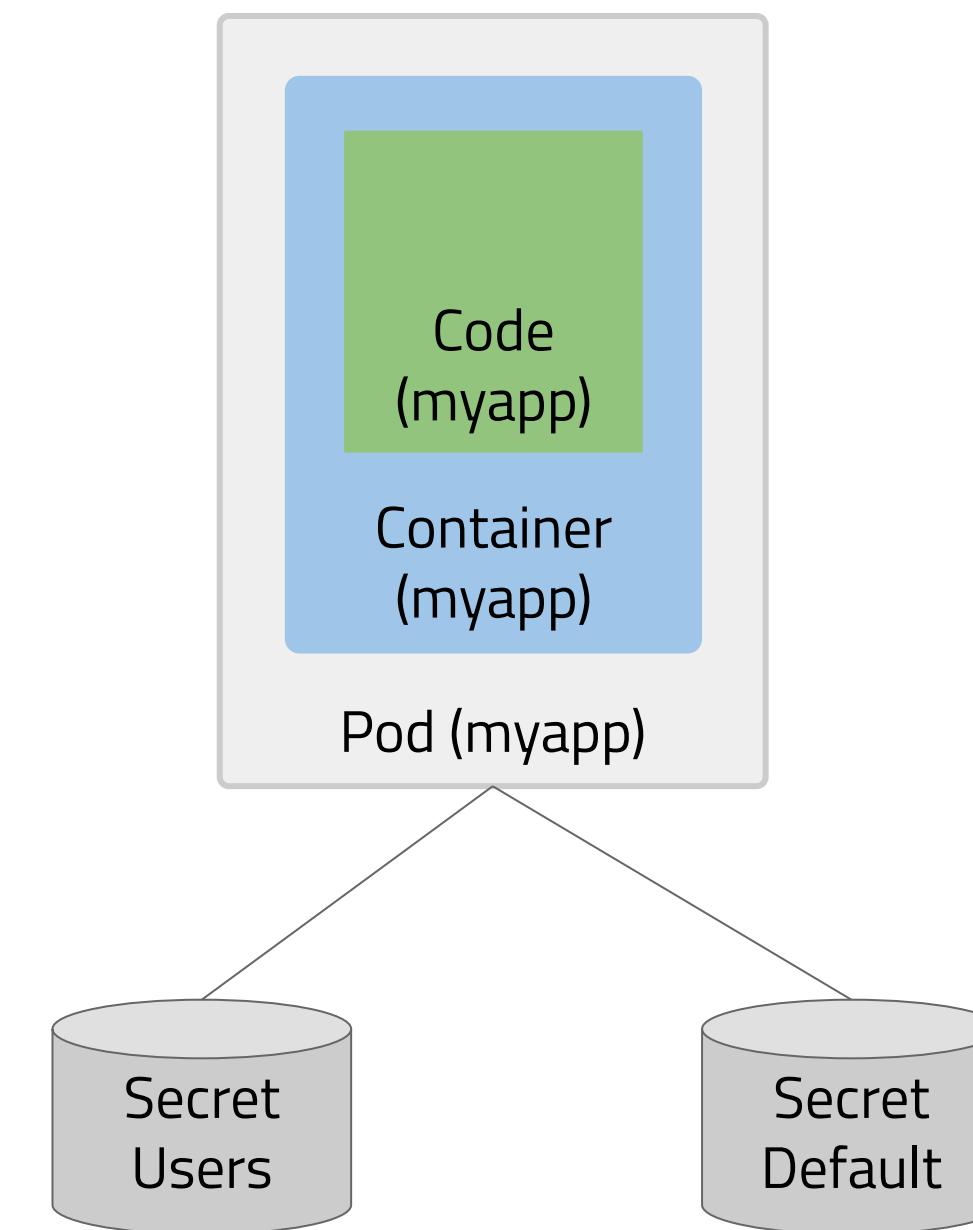
- Montar un secreto es similar que montar un volumen en un pod.

Ejemplo:

```
oc set volume dc/printenv --add --overwrite
--name=db-config-volume --mount-path /dbconfig/
--secret-name=printenv-secret
```

- El ejemplo crear los puntos de montaje con los archivos y dentro el valor:

- /dbconfig/app\_user
- /dbconfig/app\_password



# Secrets

## Command Line

### Listar Secrets

```
$ oc get secrets mysecret -o yaml
```

### Borrar Secrets

```
$ oc delete secrets mysecret
```

### Editar Secrets

```
$ oc edit secrets mysecret
```

### Convertir a base64

```
$ echo 'newpassword' | base64
bmV3cGFzc3dvcmQK
```

### Patch Secrets

```
$ oc patch secret/mysecret --patch '{"data":{"password":"bmV3cGFzc3dvcmQK"}}'
```

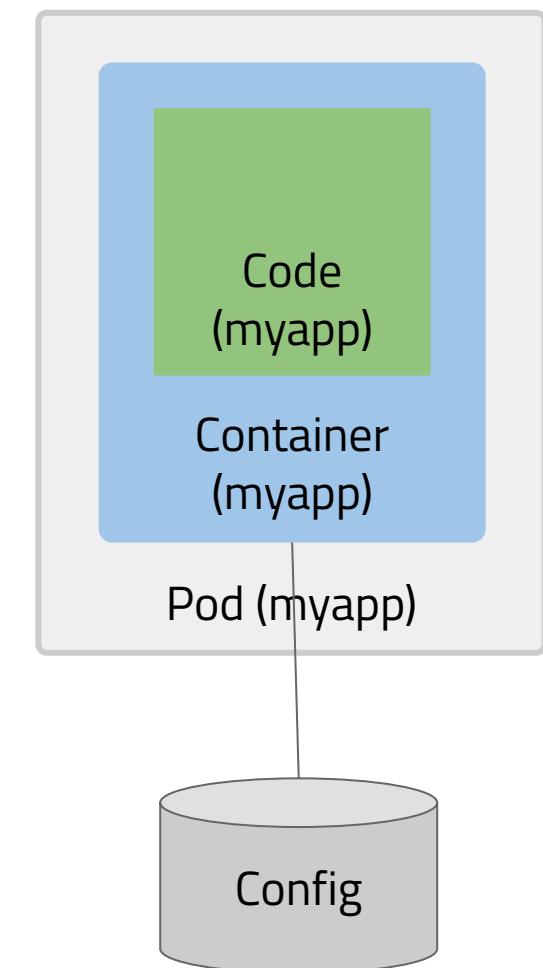
# ConfigMaps

*Semperti*

# ConfigMaps

## Overview

- Proveen un mecanismo para injectar datos de configuración a los contenedores.
- Mantiene a los contenedores agnósticos a OpenShift
- Almacenan:
  - Información detallada (propiedades individuales)
  - Archivos de configuración completos, JSON, etc.
- Mantienen datos de configuración en pares key/value que puede ser:
  - Consumidos por los pods.
  - Usados para almacenar datos de sistema para componentes controladores.
- Similar a los secrets.
  - Convenientemente soportar trabajar con strings que no sea información sensible.



# Config Maps

## Consuming and Creating

### Consumiendo ConfigMaps

- Datos de configuración consumidos en un pod para:
  - Popular valores como variables de entorno.
  - Setear argumentos para comandos en contenedor.
  - Popular archivos de configuración como volumen
- Usuarios y componentes de sistema requieren datos de configuración via ConfigMaps

### Crear

- Crear un ConfigMap desde:
  - Directorios
  - Archivos específicos
  - Literales
- Para crear un ConfigMaps:

```
oc create configmap <configmap_name>
[options]
```

### Ejemplo con literales

```
oc create configmap special-config \
--from-literal=APP_USER=admin \
--from-literal=APP_PASSWORD=passw0rd1
```

### Ejemplo desde directorio

```
$ ls example-files
game.properties
ui.properties
$ cat example-files/game.properties
enemies=aliens
lives=3
enemies.cheat=true
$ cat example-files/ui.properties
color.good=purple
color.bad=yellow
```

# Config Maps

## Creating from Directory / From Files

### Creando ConfigMap desde directorio: -o yaml

- Para ver los detalles ConfigMap, usar -o yaml
- Ejemplo:

```
$ oc get configmaps game-config -o yaml
apiVersion: v1
data:
 game.properties: |-
 enemies=aliens
 lives=3
 enemies.cheat=true
 ui.properties: |-
 color.good=purple
 color.bad=yellow
kind: ConfigMap
[...]
```

### Crear un ConfigMap desde un archivo

- Pasar la opción --from-file, especificando un solo archivo.
- Puede especificar --from-file muchas veces
  - Ejemplo:

```
oc create configmap game-config-2 \
--from-file=example-files/game.properties \
--from-file=example-files/ui.properties
```

### Crear un ConfigMap desde archivo: --from file

- Cuando es pasado el valor --from-file especificar <key>=<value>:
  - <key> como key en ConfigMap
  - <value> usado como archivo que contienen el archivo de ConfigMaps
- Ejemplo:

```
oc create configmap game-config-3 \
--from-file=game-special-key=example-files/game.properties
```

# Config Maps

## Consuming ConfigMap in Pod: Example

### Consumiendo un ConfigMaps en un Pod: Ejemplo

```
$ oc get configmap special-config -o yaml
apiVersion: v1
kind: ConfigMap
metadata:
 name: special-config
 namespace: default
data:
 APP_USER: admin
 APP_PASSWORD: passw0rd1
```

### Consumiendo un ConfigMaps en un Pod: configMapKeyRef

- Para consumir un key de un ConfigMap en un pod, se usa configMapKeyRef
- Ejemplo:

```
[...]
spec:
 containers:
 - name: test-container
 image: gcr.io/google_containers/busybox
 command: ["/bin/sh", "-c", "env"]
 env:
 - name: APP_USER_KEY
 valueFrom:
 configMapKeyRef:
 name: special-config
 key: APP_USER
[...]
```

# Config Maps

## Consuming ConfigMap: Volumes

### Consumiendo un ConfigMaps en Pod: Volume Ejemplo

```
[...]
spec:
 containers:
 - name: test-container
 image:
 gcr.io/google_containers/busybox
 command: ["/bin/sh", "cat",
 "/etc/config/APP_USER"]
 volumeMounts:
 - name: config-volume
 mountPath: /etc/config
 volumes:
 - name: config-volume
 configMap:
 name: special-config
 restartPolicy: Never
```

### Consumiendo un ConfigMaps en Pod: Custom Volume Paths

- Creado de custom path donde el archivo es localizado

```
spec:
 containers:
 - name: test-container
 image: gcr.io/google_containers/busybox
 command: ["/bin/sh", "cat",
 "/etc/config/path/to/special-key"]
 volumeMounts:
 - name: config-volume
 mountPath: /etc/config
 volumes:
 - name: config-volume
 configMap:
 name: special-config
 items:
 - key: APP_USER
 path: path/to/special-key
```

# Config Maps

## Command Line

### Listar ConfigMaps

```
$ oc get cm/myconf -o yaml
```

### Borrar ConfigMaps

```
$ oc delete cm myconf
```

### Editar ConfigMaps

```
$ oc edit cm myconf
```

### Patch ConfigMaps

```
$ oc patch configmap/myconf --patch '{"data":{"key1":"newvalue1"}}'
```

# Config Maps

## Restricciones de usar ConfigMaps

### Restricciones en el uso de ConfigMaps

- Deben ser creado antes de ser consumidos por un pod.
- Deben residir en un proyecto.
  - No puede compartirse ConfigMaps entre proyectos.
- Cuando se actualiza un ConfigMaps no es actualizado en el pod, debe ser reiniciado.



# Downward API

*Semperti*

# Downward API

## Overview

- Permite a los contenedores consumir información de objetos de la API.
  - Pod Name
  - Namespaces
  - Resource value
- Usado Como:
  - Variables de entorno.
  - Volumenes
- Campos dentro del pod seleccionados mediante el tipo de API FieldRef
  - FieldRef tiene dos tipos:
    - filePath: Path al campo a seleccionar, relativo al pod.
    - apiVersion: API version para interpretar el selector de filePath

| Selector                           | Description     | Consumed Through                   |
|------------------------------------|-----------------|------------------------------------|
| metadata.name                      | Pod name        | Environment variables, volumes     |
| metadata.namespace                 | Pod namespace   | Environment variables, volumes     |
| metadata.labels                    | status.podIP    | Pod IP                             |
| Environment variables, not volumes | Pod labels      | Volumes, not environment variables |
| metadata.annotations               | Pod annotations | Volumes, not environment variables |

# Downward API

## Consumo

### Consumo a través de variables de entorno

```
apiVersion: v1
kind: Pod
metadata:
 name: dapi-env-test-pod
spec:
 containers:
 - name: env-test-container
 image: gcr.io/google_containers/busybox
 command: ["/bin/sh", "-c", "env"]
 env:
 - name: MY_POD_NAME
 valueFrom:
 fieldRef:
 fieldPath: metadata.name
 - name: MY_POD_NAMESPACE
 valueFrom:
 fieldRef:
 fieldPath: metadata.namespace
 restartPolicy: Never
```

### Consumo a través de volúmenes

```
terminationMessagePolicy: File
 volumeMounts:
 - name: podinfo
 mountPath: /downward
 readOnly: false
 .
 .
 .
terminationGracePeriodSeconds: 30
 volumes:
 - name: podinfo
 downwardAPI:
 items:
 - path: "pod_labels"
 fieldRef:
 fieldPath: metadata.labels
```

# Downward API

## Consumo

### Resource and Limits usando var envs

```
spec:
 containers:
 - name: test-container
 image:
 gcr.io/google_containers/busybox:1.24
 command: ["/bin/sh", "-c",
 "env"]
 resources:
 requests:
 memory: "32Mi"
 cpu: "125m"
 limits:
 memory: "64Mi"
 cpu: "250m"
...
 env:
 - name: MY_CPU_REQUEST
 valueFrom:
 resourceFieldRef:
 resource: requests.cpu
 - name: MY_CPU_LIMIT
 valueFrom:
 resourceFieldRef:
 resource: limits.cpu
 - name: MY_MEM_REQUEST
 valueFrom:
 resourceFieldRef:
 resource: requests.memory
 - name: MY_MEM_LIMIT
 valueFrom:
 resourceFieldRef:
 resource: limits.memory
```

### Resource and Limits usando volumenes

```
spec:
 containers:
 - name: client-container
...
 volumes:
 - name: podinfo
 downwardAPI:
 items:
 - path: "cpu_limit"
 resourceFieldRef:
 containerName: client-container
 resource: limits.cpu
 - path: "cpu_request"
 resourceFieldRef:
 containerName: client-container
 resource: requests.cpu
 - path: "mem_limit"
 resourceFieldRef:
 containerName: client-container
 resource: limits.memory
 - path: "mem_request"
 resourceFieldRef:
 containerName: client-container
 resource: requests.memory
```

# Application Configuration

## Resumen

- Application Configuration Data Storage
  - EmptyDir Volumes
  - Persistent Volumes
  - Environment Variables
  - Secrets
  - ConfigMaps
  - Downward API

Día 4

*Semperti*

# 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.*

# Users and Projects

## Login

### Login and Authentication

- Todos los usuarios deben autenticarse con Openshift
- Las solicitudes de API que carecen de autenticación válida se autentican como usuario anónimo.
- Las políticas determinar que usuario es autorizado a hacer algo.

### Web Console Authentication

- Es accedida vía una url provista por el administrador.
- Proporcionar credenciales de inicio de sesión para obtener el token para realizar llamadas a la API
- Utilizada para navegar entre proyectos.

### CLI Login

- Usado con cliente oc (CLI) para loguear de la misma manera que la consola web.
- Proporcionar credenciales de inicio de sesión para obtener el token para realizar llamadas a la API

```
oc login -u <my-user-name> --server="<master-api-public-addr>:<master-public-port>"
```

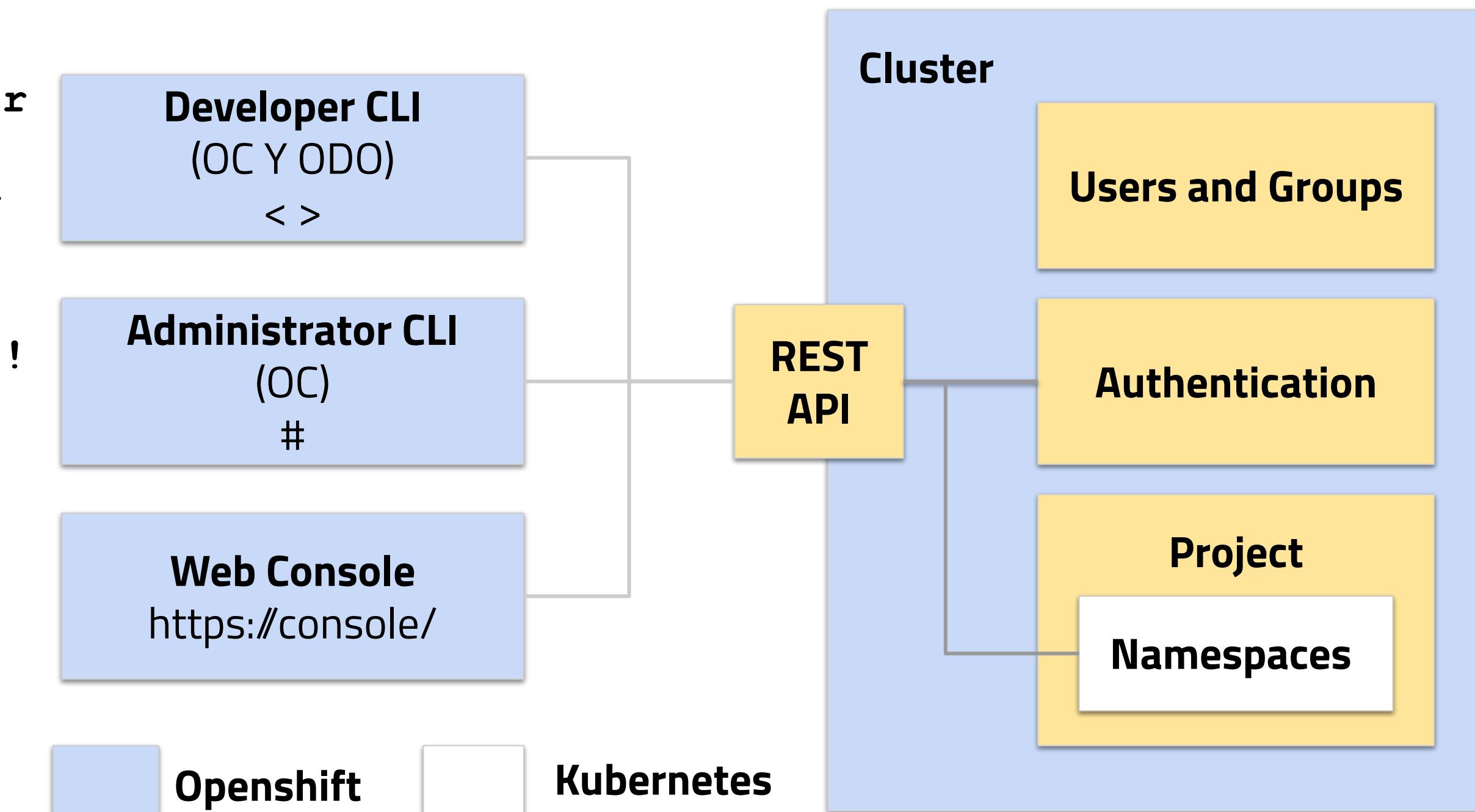
Los administradores pueden hacer que el clúster genere una clave para la autenticación sin contraseña

# 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>
```

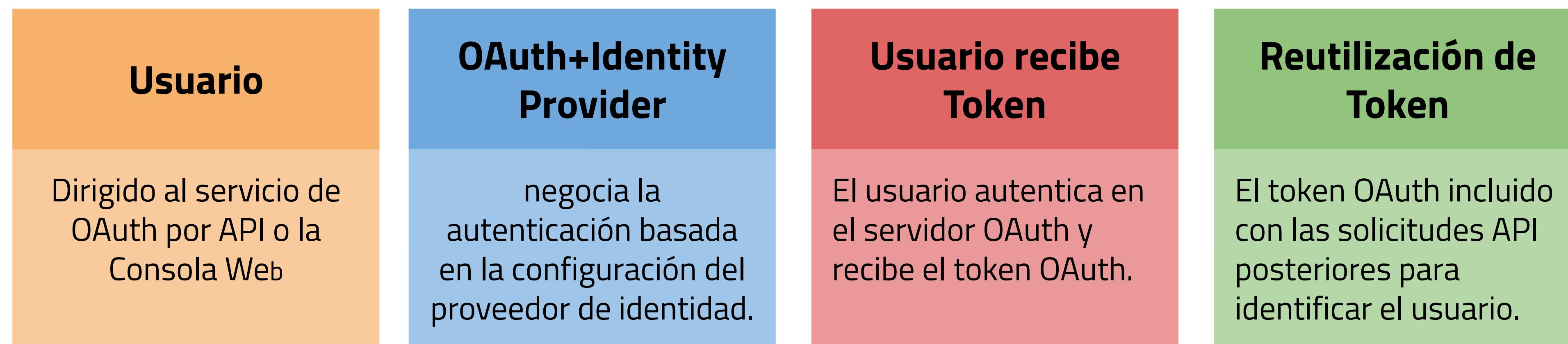


| Ambiente | Web Console                                                                                                                               | REST API                                                                                      |
|----------|-------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------|
| NO PROD  | <a href="https://console-openshift-console.apps.ocp-np.sis.ad.bia.itau">https://console-openshift-console.apps.ocp-np.sis.ad.bia.itau</a> | <a href="https://api.ocp-np.sis.ad.bia.itau:6443">https://api.ocp-np.sis.ad.bia.itau:6443</a> |
| PROD     | <a href="https://console-openshift-console.apps.ocp-pa.sis.ad.bia.itau">https://console-openshift-console.apps.ocp-pa.sis.ad.bia.itau</a> | <a href="https://api.ocp-pa.sis.ad.bia.itau:6443">https://api.ocp-pa.sis.ad.bia.itau:6443</a> |
| PROD     | <a href="https://console-openshift-console.apps.ocp-pb.sis.ad.bia.itau">https://console-openshift-console.apps.ocp-pb.sis.ad.bia.itau</a> | <a href="https://api.ocp-pb.sis.ad.bia.itau:6443">https://api.ocp-pb.sis.ad.bia.itau:6443</a> |

# 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.

### 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`

### Command Line

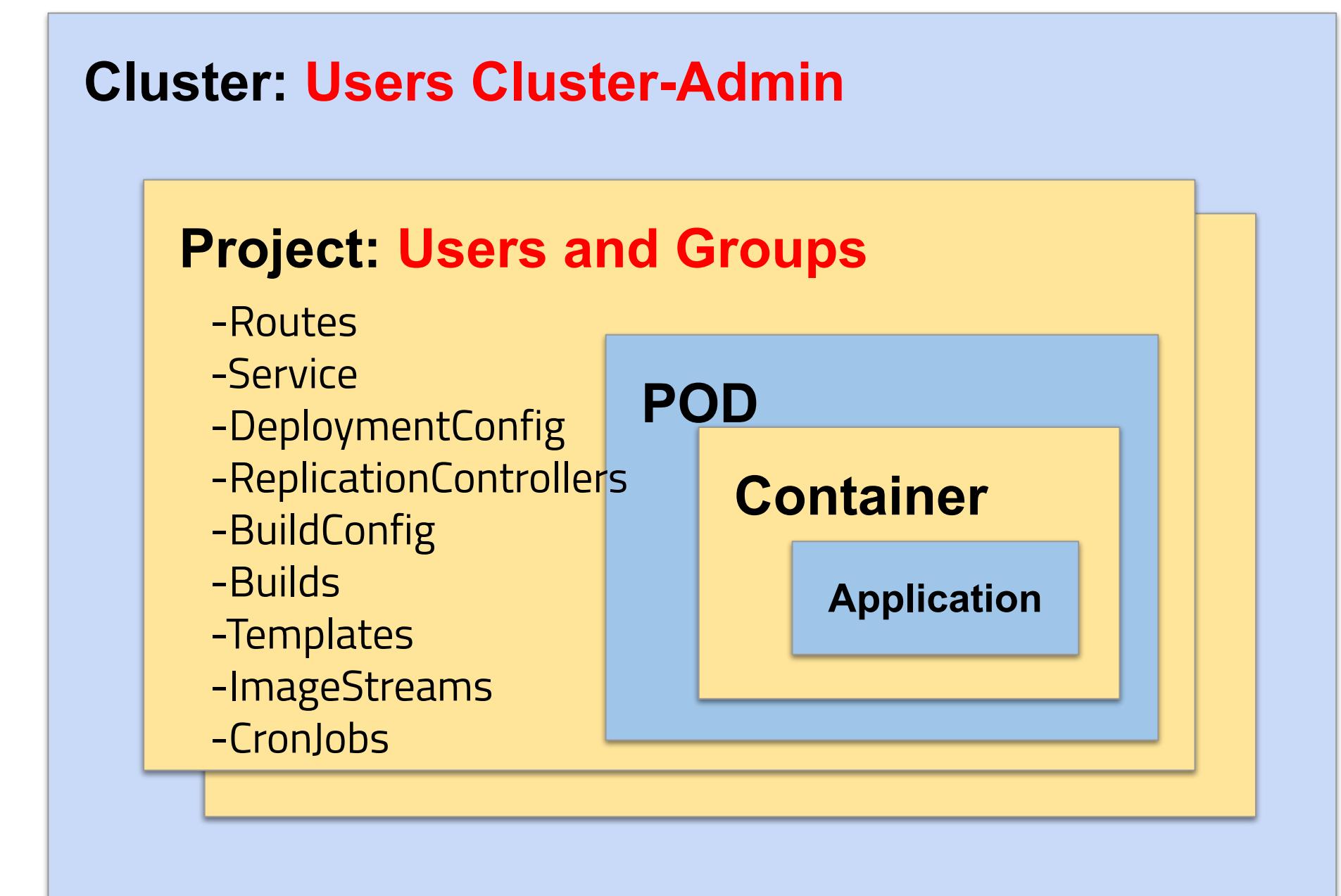
- Usuario Conectado
  - **oc whoami shows current user**
  - **oc whoami --show-console**
  - **oc whoami --show-server**
  - Token
    - **oc whoami --show-token**
    - Oauth Introspection
- **oc login --loglevel=9 -u USER -p PASSWORD API\_URL**

# 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.



Identity Providers:

<https://docs.openshift.com/container-platform/4.2/authentication/understanding-identity-provider.html>

# 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
```

# Users and Projects

## Projects

- Permite a los grupos de usuarios y desarrolladores trabajar de manera conjunta.
- Unidad de aislamiento y colaboración.
- Define el alcance de los recursos.
- Permite a los administradores y colaboradores del proyecto administrar los recursos
- Restringe y rastrea el uso de recursos con cuotas y límites
- Kubernetes namespaces con annotation adicionales.
- Vehículo central para administrar el acceso a los recursos para usuarios regulares.
- Permite que la comunidad de usuarios organice y gestione el contenido de forma aislada de otras comunidades.



## Command Line

```
$ oc explain project
$ oc explain namespaces
$ oc get projects
```

# Users and Projects

## Projects

### Usuarios:

- Reciben el acceso a los proyectos desde el administrador.
- Tienen acceso a los proyectos propios si se les permite crearlos.

### Cada proyecto tiene sus propios:

- **Objetos:** Pods, Services, ReplicationControllers, etc.
- **Políticas:** reglas que especifican qué usuarios pueden o no realizar acciones en los objetos.
- **Restricciones:** quotas para objetos que pueden limitarse por cantidad.
- **Service Account:** usuarios que actúan automáticamente con acceso a los objetos del proyecto.

### Command Line

```
$ oc get users
$ oc get groups
```

### Project: 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

# Users and Projects

## Users and User Types

- Toda interacción con Openshift está asociada a un usuario.
- Sistema de permisos agregando roles a los usuarios y grupos.

Tipos de usuarios:

| Usuarios Regulares                                                                                                                                                                                                                                           | Usuarios de Sistema                                                                                                                                                                                                                                                           |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <ul style="list-style-type: none"><li>• Cómo están representados la mayoría de los usuarios interactivos de Openshift</li><li>• Creado automáticamente en el sistema en el primer inicio de sesión</li><li>• Representado con objetos tipo usuario</li></ul> | <ul style="list-style-type: none"><li>• Muchos de ellos creados automáticamente cuando es desplegada la infraestructura.</li><li>• Creados para interactuar con la API de manera segura.</li><li>• Incluye: cluster administrator, per-node user, services account.</li></ul> |

## Command Line

```
$ oc get users
$ oc get groups
```

# 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                                         |
|-----------------------------------------|-----------------------------------------------------|
| <b>system:authenticated</b>             | Usuarios autenticados                               |
| <b>system:authenticated:oauth</b>       | Usuarios autenticados con OAuth                     |
| <b>system:cluster-admins</b>            | Cluster administrators como kubeadmin system:admin  |
| <b>system:serviceaccounts</b>           | Todos los service account                           |
| <b>system:serviceaccounts:NAMESPACE</b> | Todos los service account en en proyecto específico |
| <b>system:unauthenticated</b>           | Usuarios anónimos que pueden autenticarse.          |

# 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

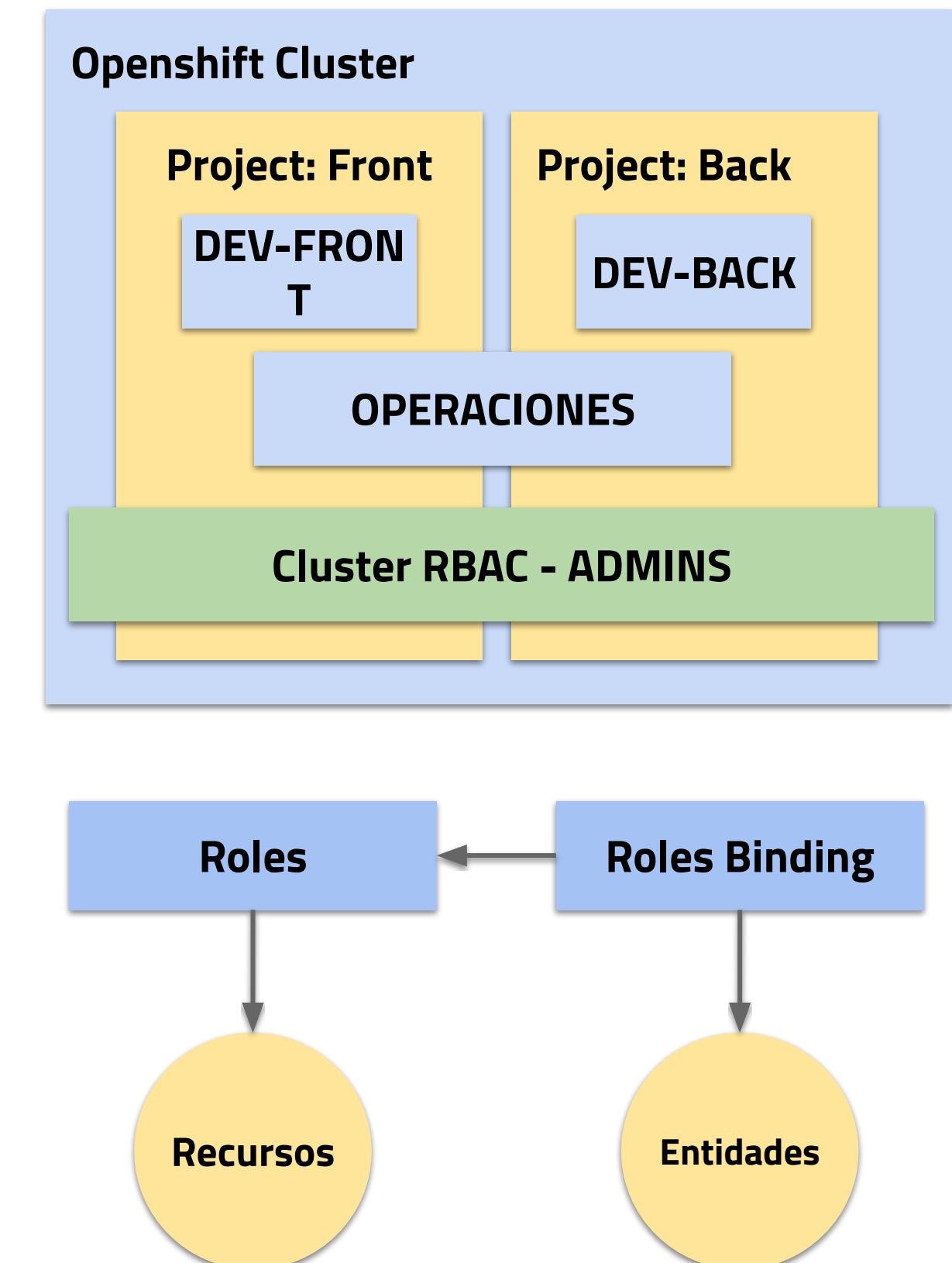
```
$ oc get pods -n openshift-authentication
$ oc logs -n openshift-authentication-operator deployment/authentication-operator
$ oc logs -n openshift-authentication oauth-openshift-564976c856-ffhwr
```

# Autorización

## Vision General

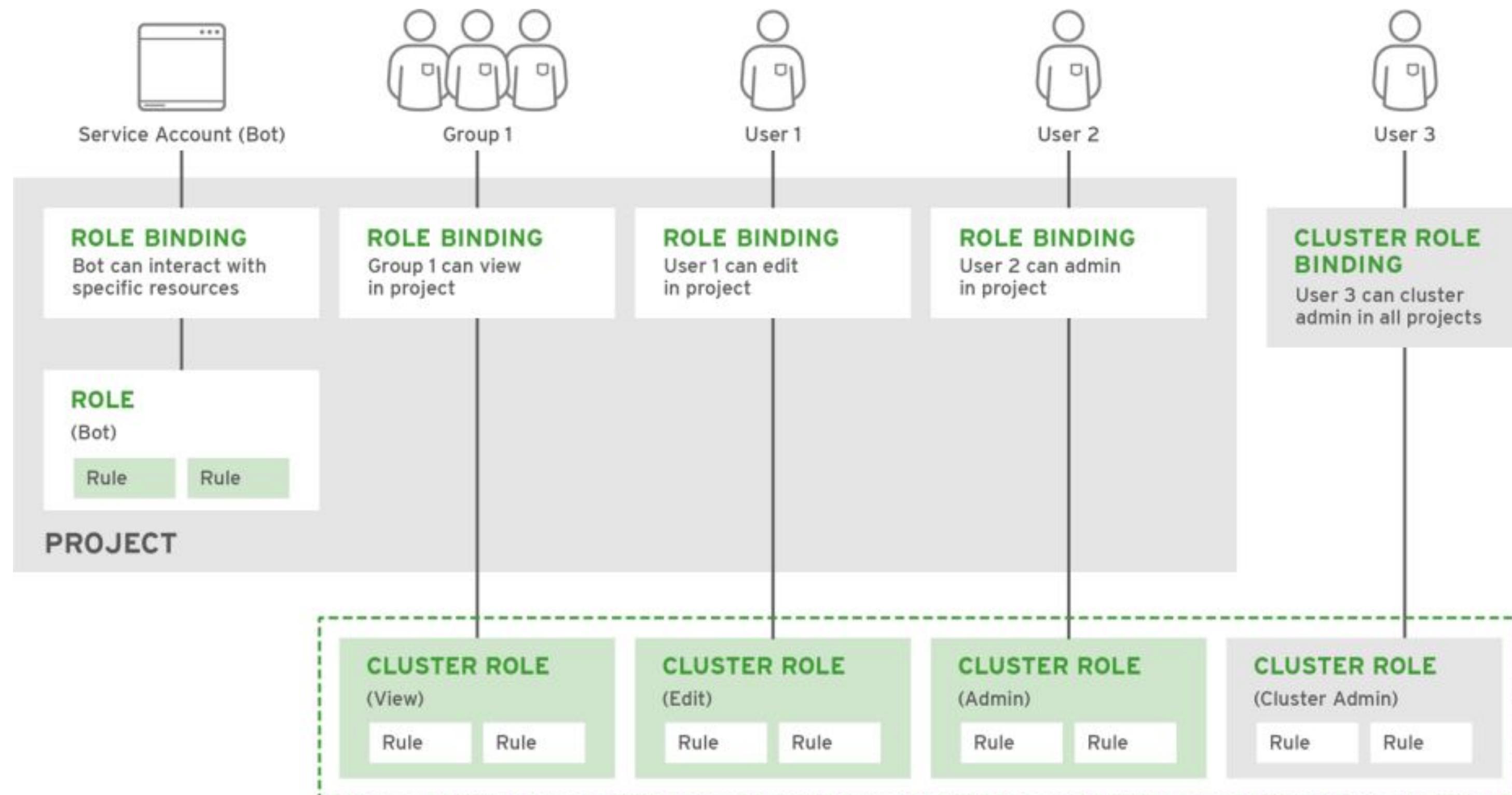
Logs objetos RBAC determinan si el usuario puede o no realizar una acción específica con respecto al tipo de recurso. Deny by default, es la política de acceso predeterminadas por Openshift RBAC.

- **Roles:** mapa de acciones permitidas (verbos) a recursos de un proyecto. (LIST pods)
- **ClusterRoles:** mapa de acciones permitidas (verbos) a recursos de clúster 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 en un namespace

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

### Cluster Role Binding Management

|                                |                                                                              |
|--------------------------------|------------------------------------------------------------------------------|
| Add cluster role to user       | <code>oc adm policy add-cluster-role-to-user CLUSTER_ROLE USER</code>        |
| Add cluster role to group      | <code>oc adm policy add-cluster-role-to-group CLUSTER_ROLE GROUP</code>      |
| Remove cluster role from user  | <code>oc adm policy remove-cluster-role-from-user CLUSTER_ROLE USER</code>   |
| Remove cluster role from group | <code>oc adm policy remove-cluster-role-from-group CLUSTER_ROLE GROUP</code> |

# **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                                                                                |
|-----------------|--------------------------------------------------------------------------------------------|
| <b>builder</b>  | Service account usado para build pods, push images                                         |
| <b>deployer</b> | Service account usado para desplegar pods, implementar DeploymentConfig rollout, rollback. |
| <b>default</b>  | Service account usado para los pods.                                                       |

### Command Line

```
oc create serviceaccount NAME -n NAMESPACE
oc get serviceaccount NAME -n NAMESPACE
oc describe serviceaccount NAME -n NAMESPACE
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

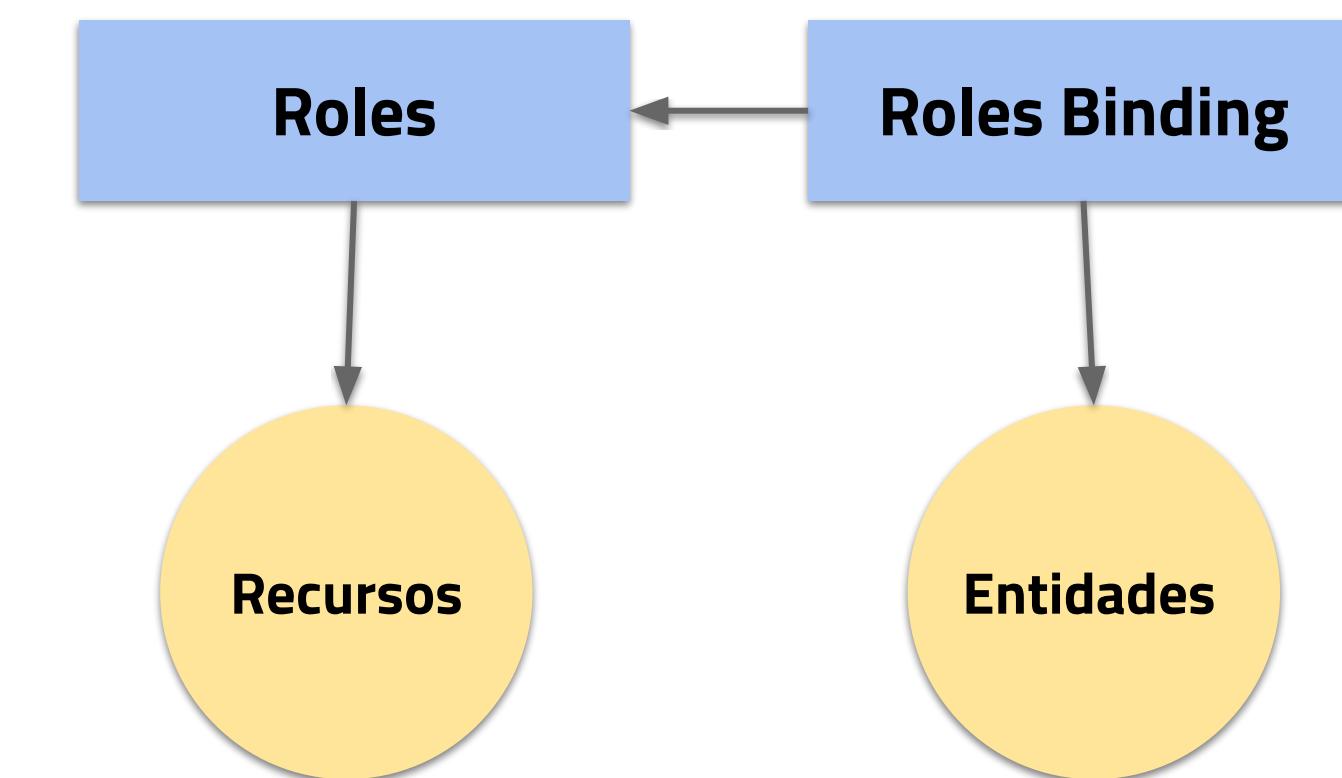
```
oc get secret --field-selector=type=kubernetes.io/service-account-token -n
NAMESPACE
```

### 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

## SCCs

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                                                                                                                              |
|-------------------------|------------------------------------------------------------------------------------------------------------------------------------------|
| <b>anyuid</b>           | Permite a los contenedores correr con cualquier user id, <i>incluido</i> root user (uid=0)                                               |
| <b>hostaccess</b>       | Permite a los contenedores acceder a archivos de sistema del host, network y tabla de procesos restringido al id de usuario              |
| <b>hostmount-anyuid</b> | Permite a los contenedores acceder a los archivos de sistema usando host mount, ejecutarlo como cualquier user id.                       |
| <b>hostnetwork</b>      | Permite a los contenedores acceder al networking del host, host ports.                                                                   |
| <b>node-exporter</b>    | Reservado para el usuario de Prometheus node exporter.                                                                                   |
| <b>nonroot</b>          | Permite a los contenedores correr con cualquier id de usuario excepto root (uid=0)                                                       |
| <b>privileged</b>       | Permite el acceso privilegiado y host features - acceso más relajado, usado solo por cluster-admins.                                     |
| <b>restricted</b>       | Negado el acceso a todos los host features, requerido a los pods (containers) correr con id de usuario restringido. <b>(default SCC)</b> |

# 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**

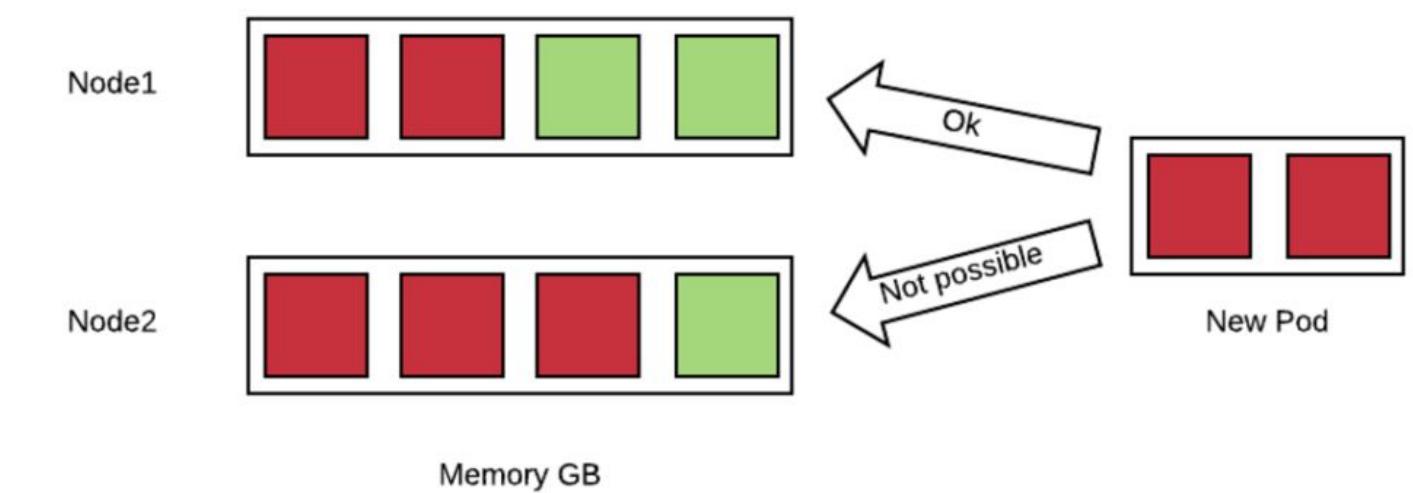
**Objetos**

**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**

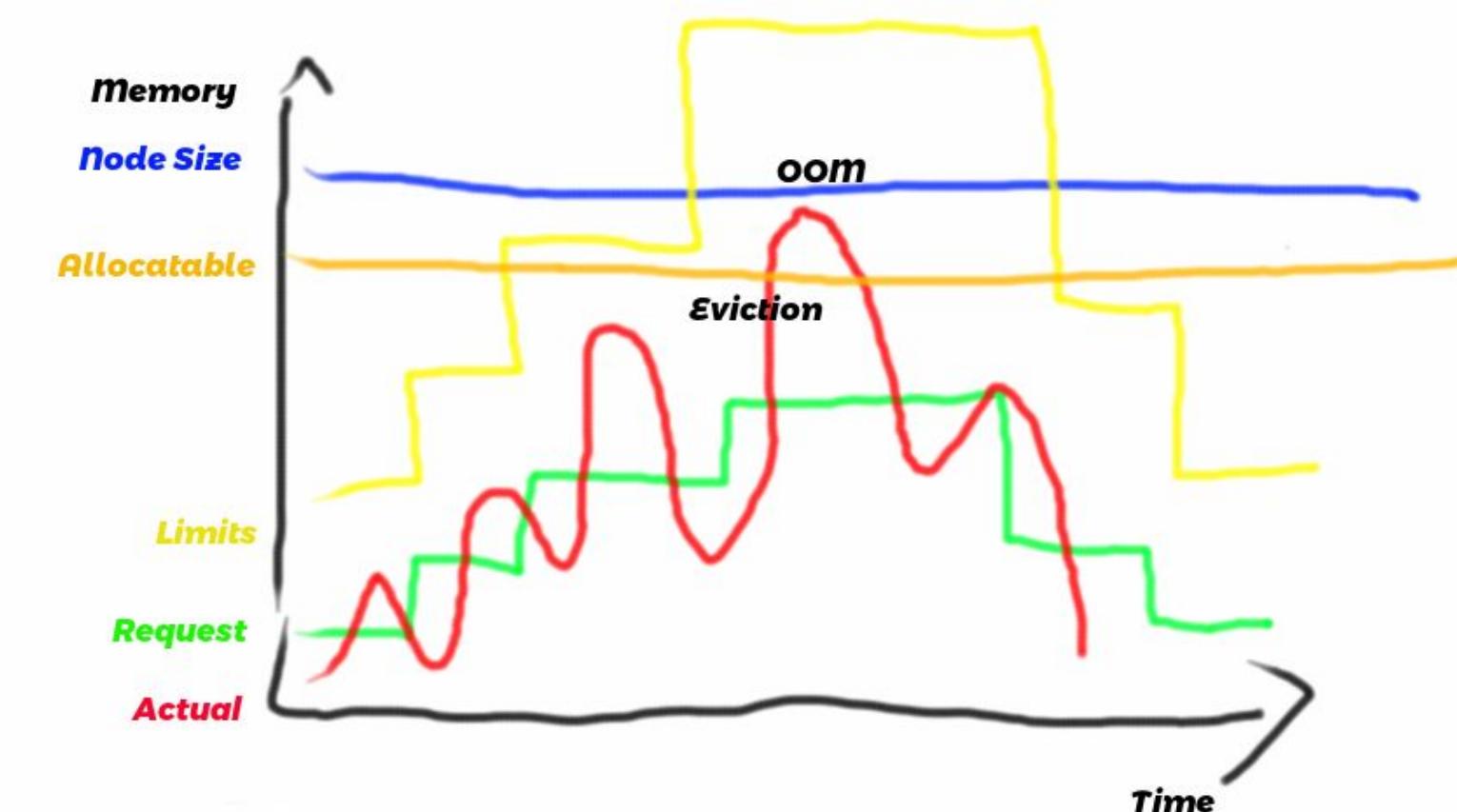
Reserva de recursos para el servicio de kubelet.

- **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)
 ...

```

| Namespace                                                    | Name                                              | CPU Requests | CPU Limits | Memory Requests | Memory Limits | AGE   |
|--------------------------------------------------------------|---------------------------------------------------|--------------|------------|-----------------|---------------|-------|
| (25 in total)                                                |                                                   |              |            |                 |               |       |
| 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-57df945856-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-nphfv                                         | 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.
- Limite tolerable de pods en estado Evicted.

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

- Por proyecto utilizar LimitRanges.

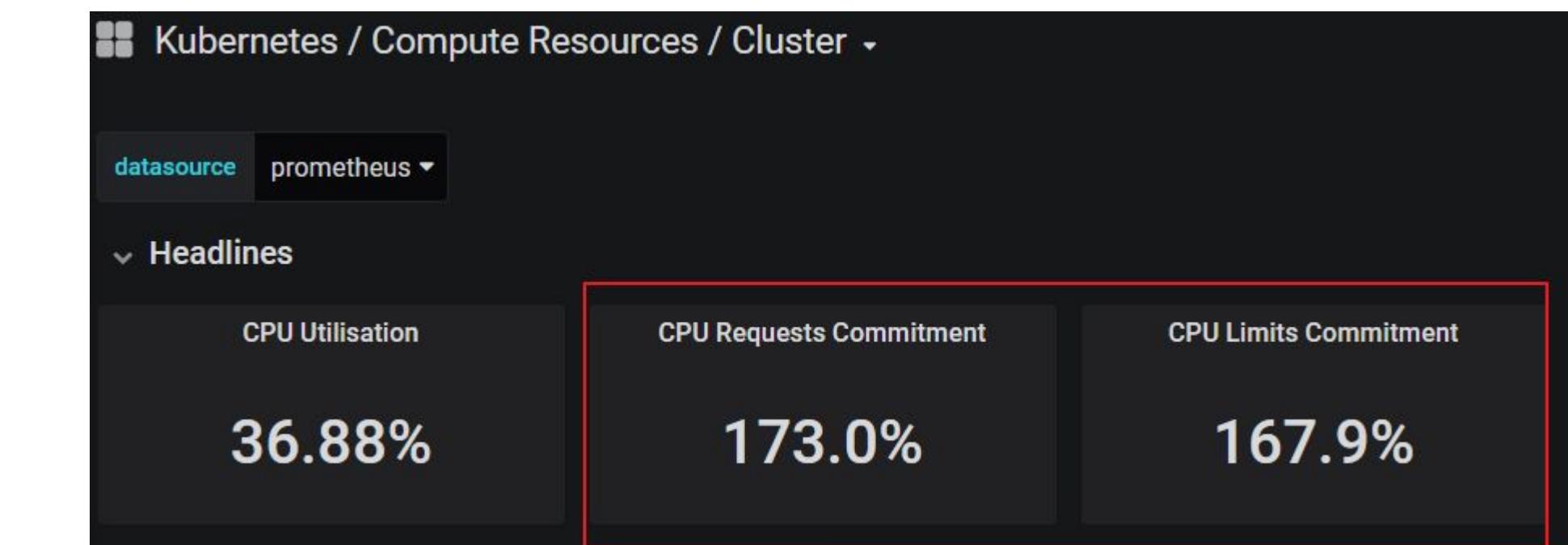
### Recomendaciones

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
```

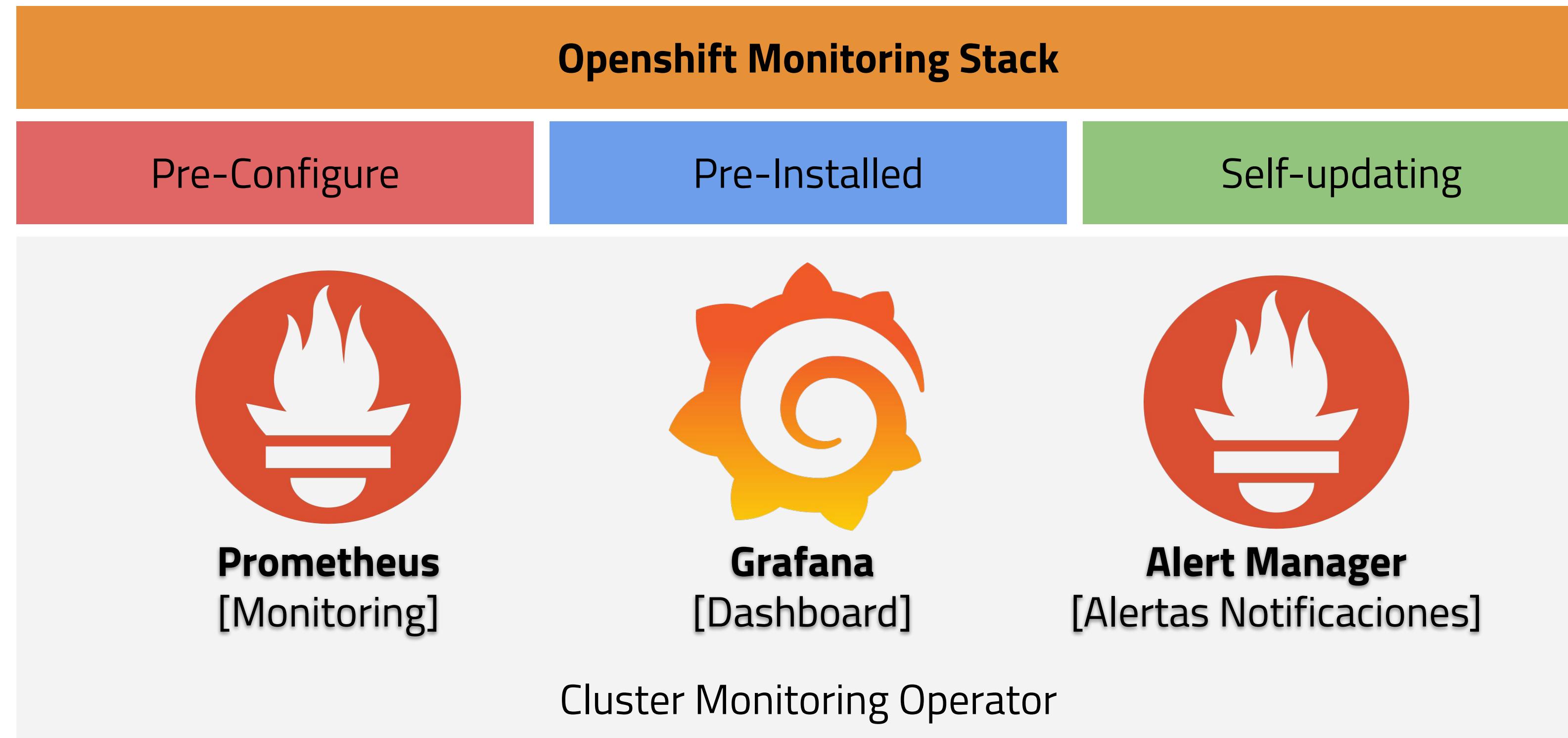
Todas las configuración que se hagan por cualquiera de los dos metodos impactan en un archivo de configuracion en cada uno de los nodos de Openshift en el path /etc/kubernetes/kubelet.conf

# Monitoreo

*Semperti*

# Monitoring

## Cluster Monitoring Operator

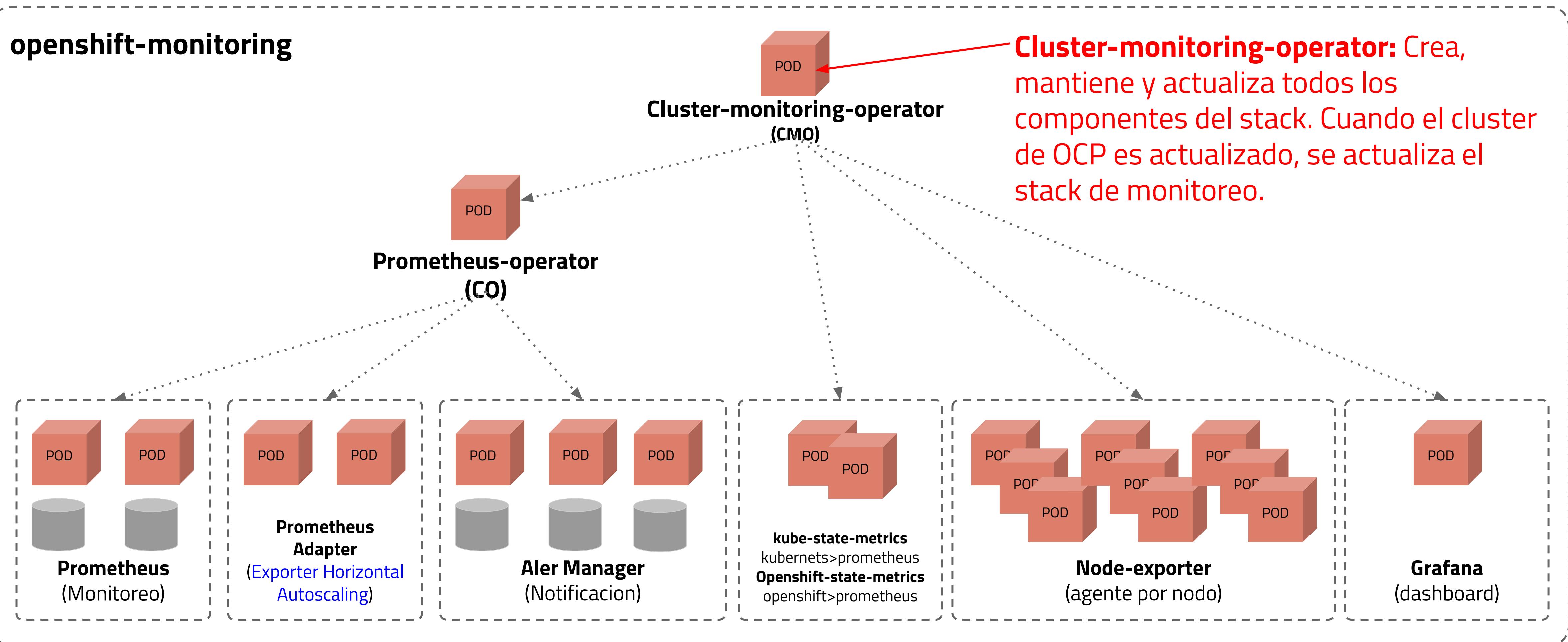


**ALERT!** Para asegurar la compatibilidad con futuros OpenShift updates, solamente está soportada la configuración del stack que se detalla en la documentación

[https://access.redhat.com/documentation/en-us/openshift\\_container\\_platform/4.3/html-single/monitoring/index](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 | <b>prometheusOperator</b>   |
| Prometheus          | <b>prometheusK8s</b>        |
| Alertmanager        | <b>alertmanagerMain</b>     |
| kube-state-metrics  | <b>kubeStateMetrics</b>     |
| Grafana             | <b>grafana</b>              |
| Telemeter Client    | <b>telemeterClient</b>      |
| Prometheus Adapter  | <b>k8sPrometheusAdapter</b> |

# 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"
```

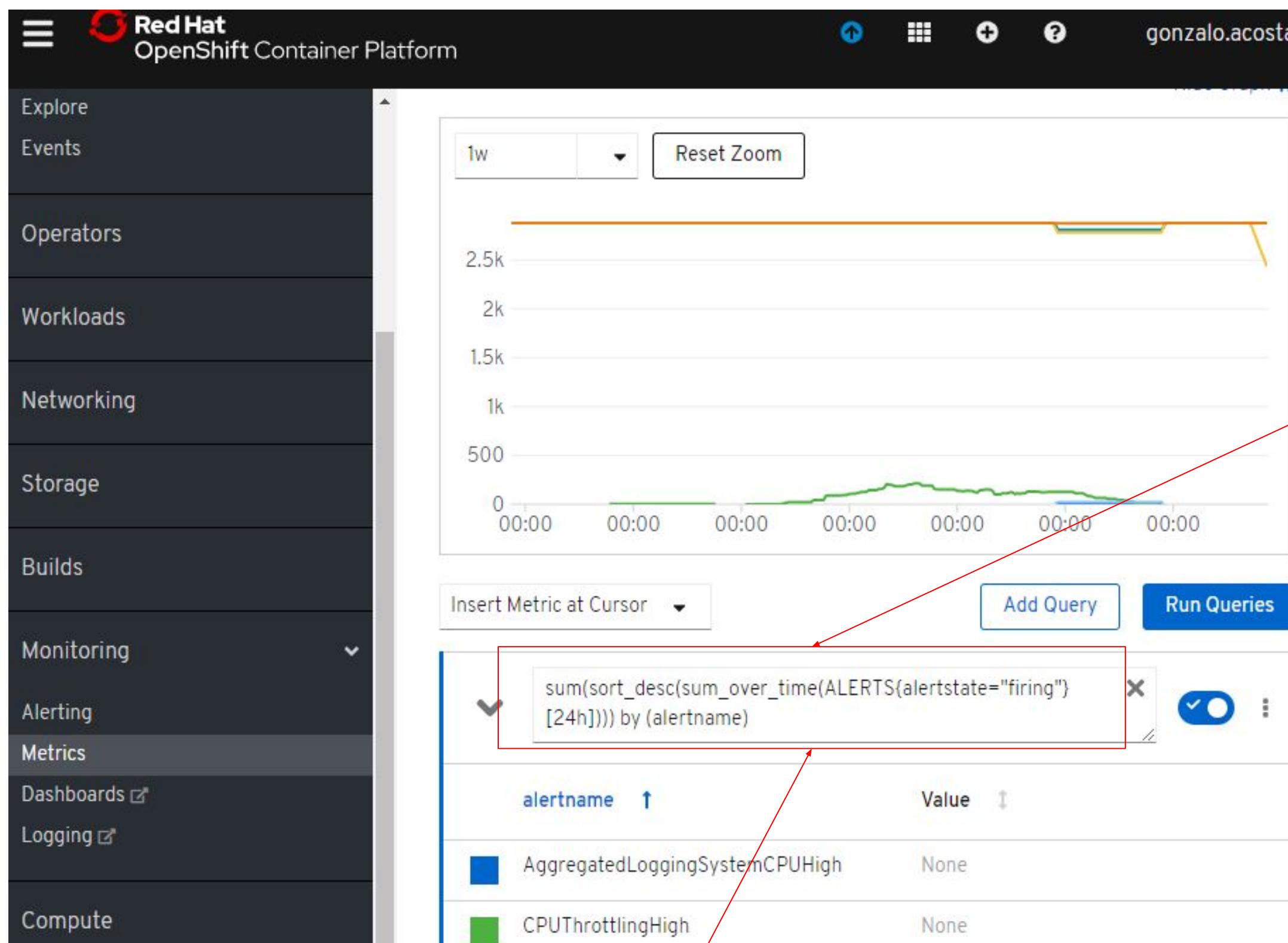
Tipos de receivers:

- <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))
```

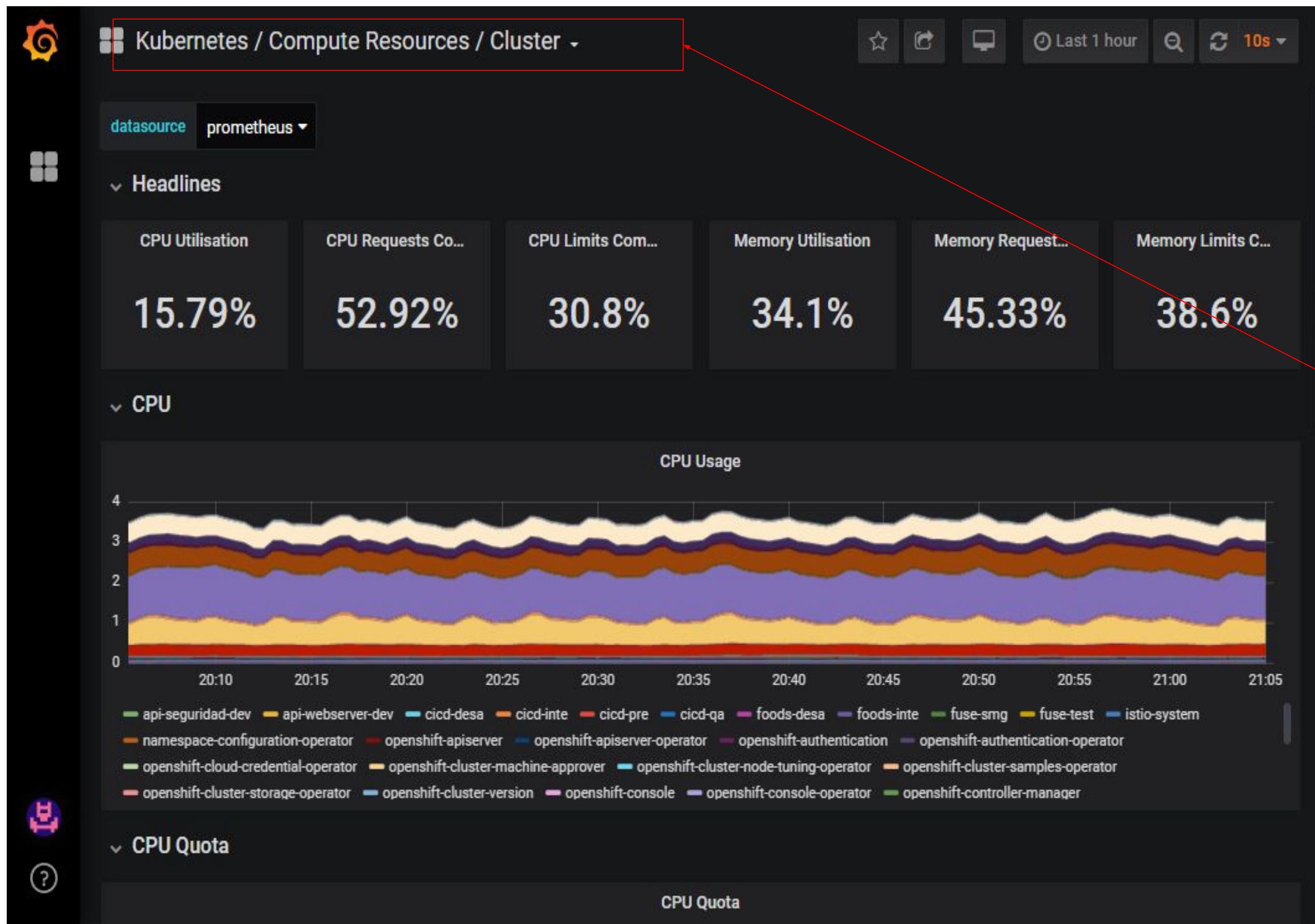
### Transmit Bandwidth

```
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 metricas 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

Día 5

*Semperti*

# Controller and Scheduler

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

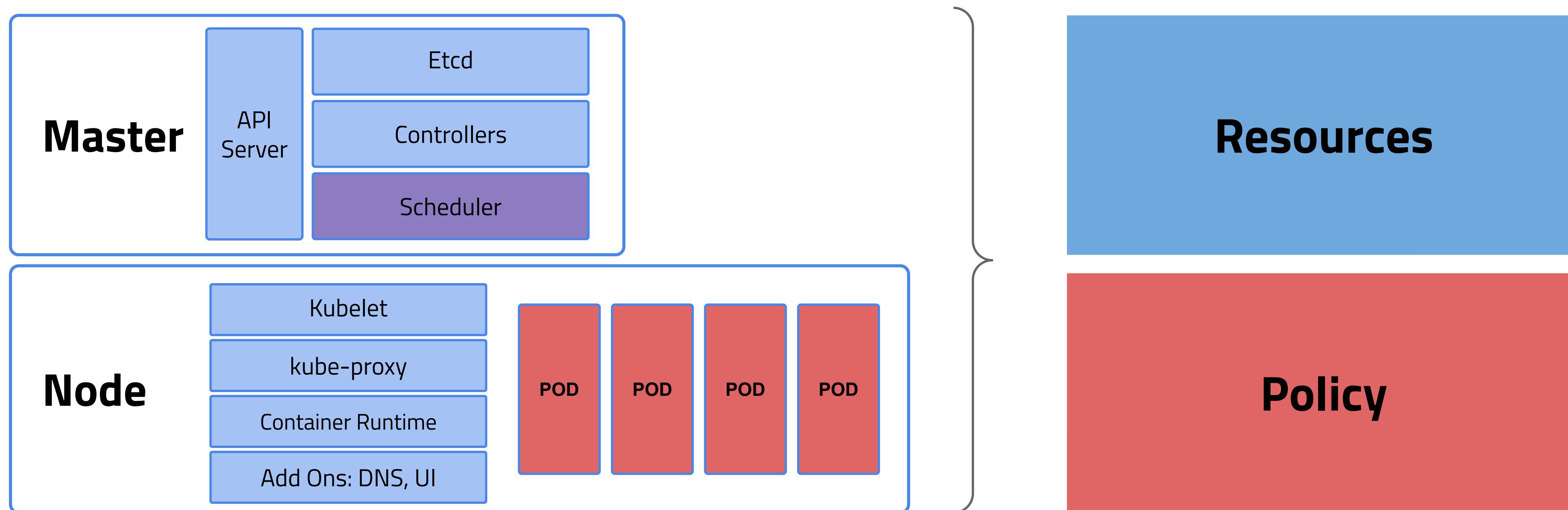
# Controller and Scheduler

*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

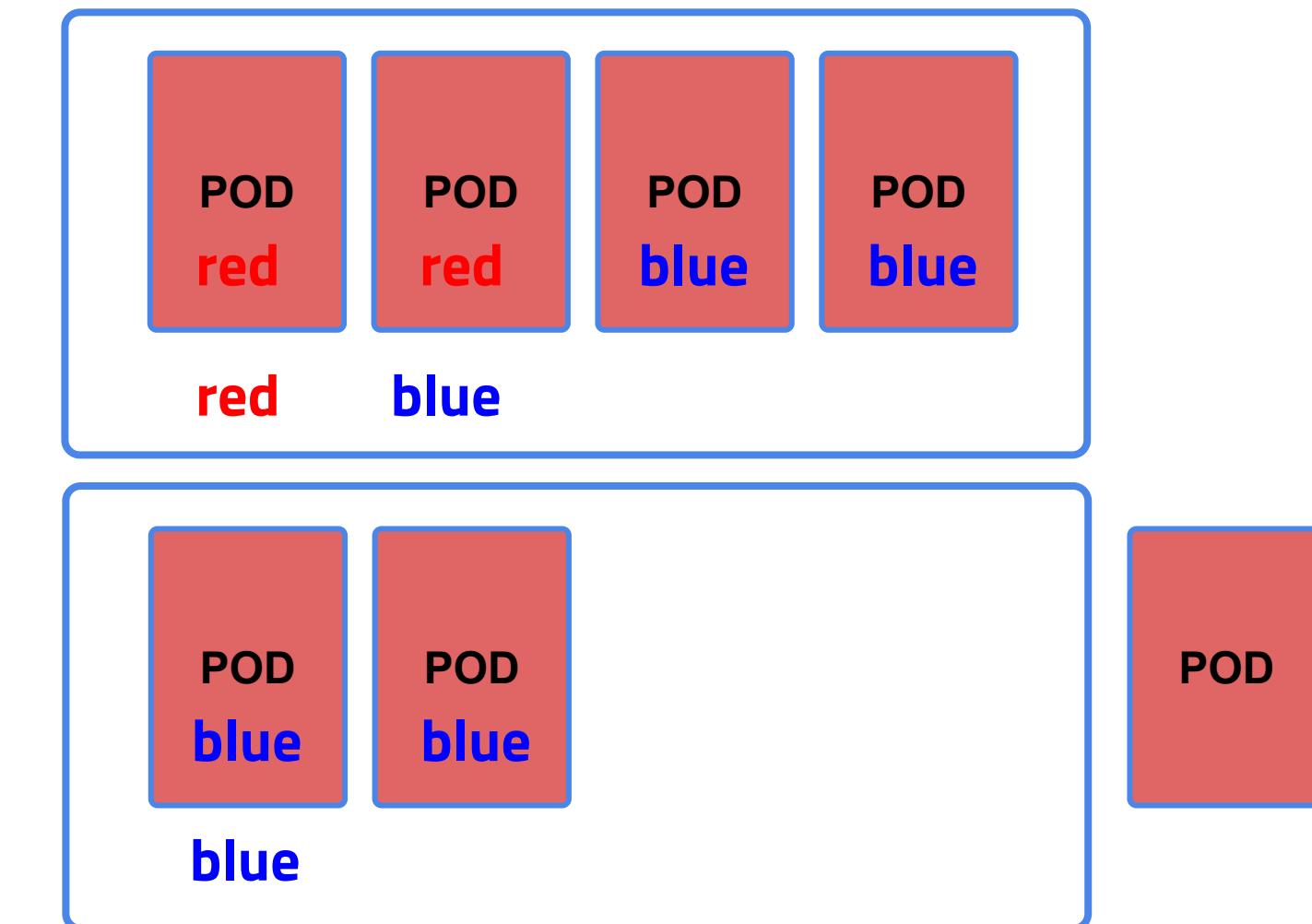
Manual  
Scheduling

# 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

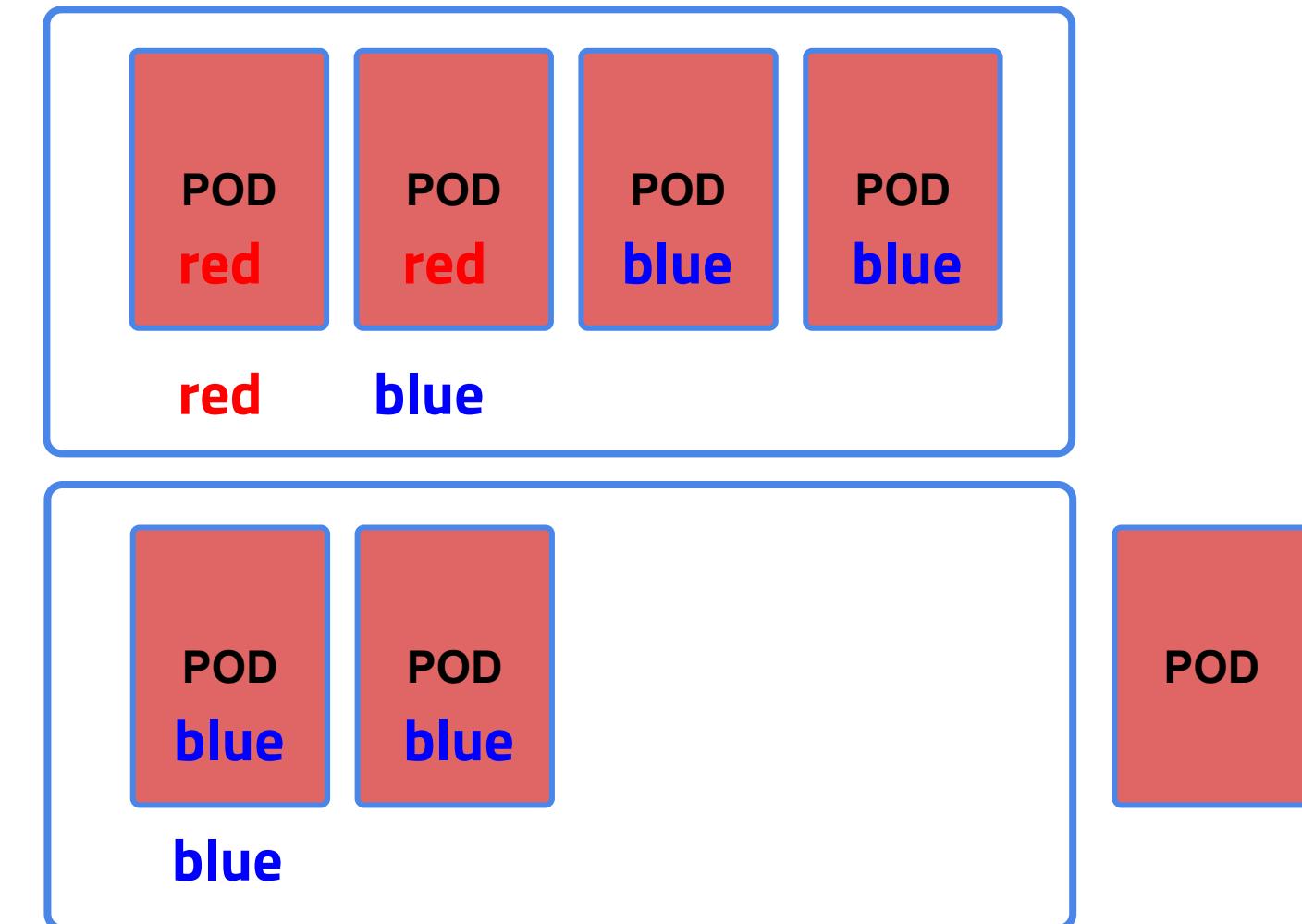
Asignar pods a nodos usando nodeSelector

1. Etiquetar los nodos worker

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

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

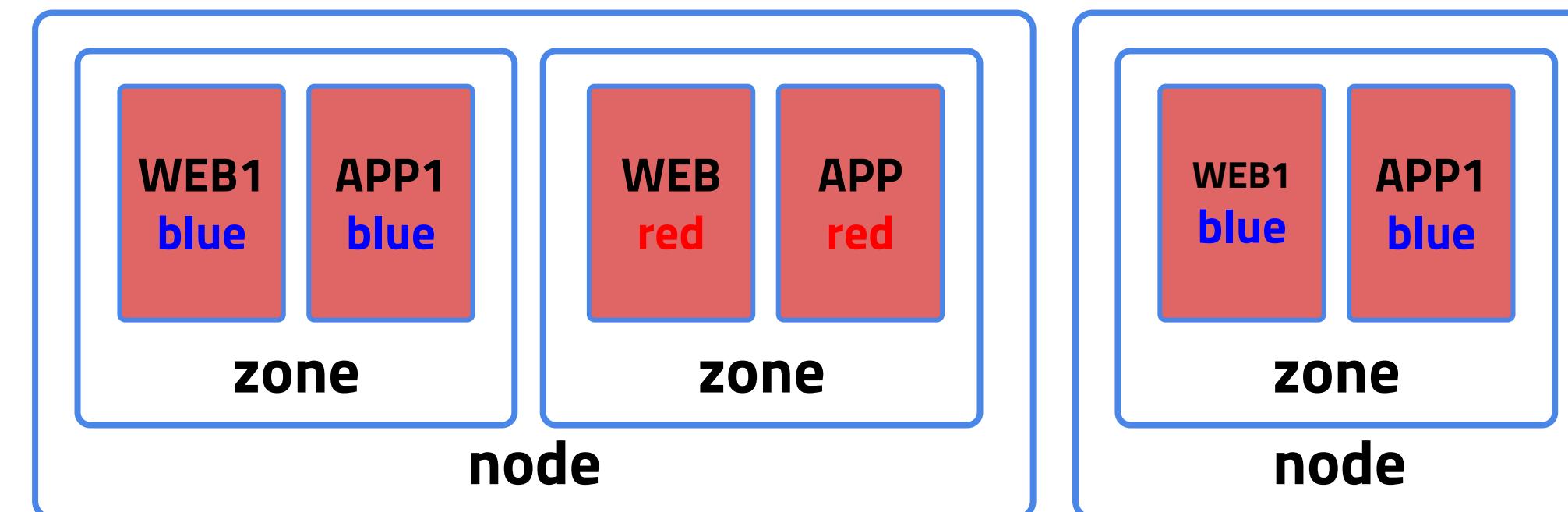
```
spec:
 containers:
 - name: hello-world
 image: gcr.io/google-samples/hello-app:1.0
 port:
 - containerPort: 8080
nodeSelector:
 zone: red
```



# 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"
```

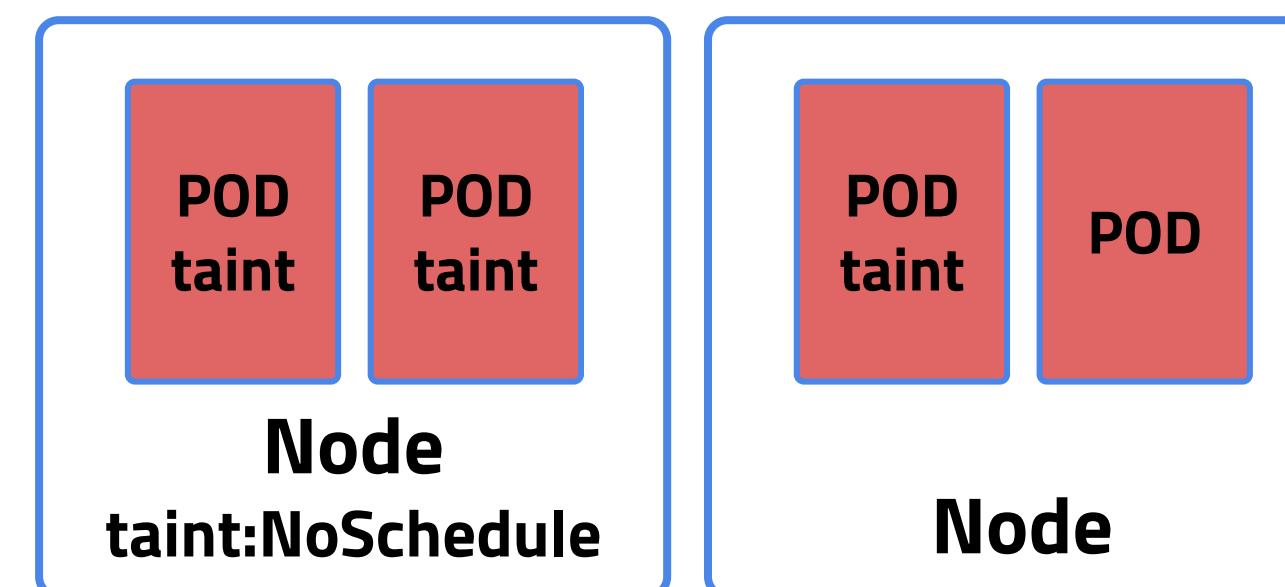
# Controller and Scheduler

## 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

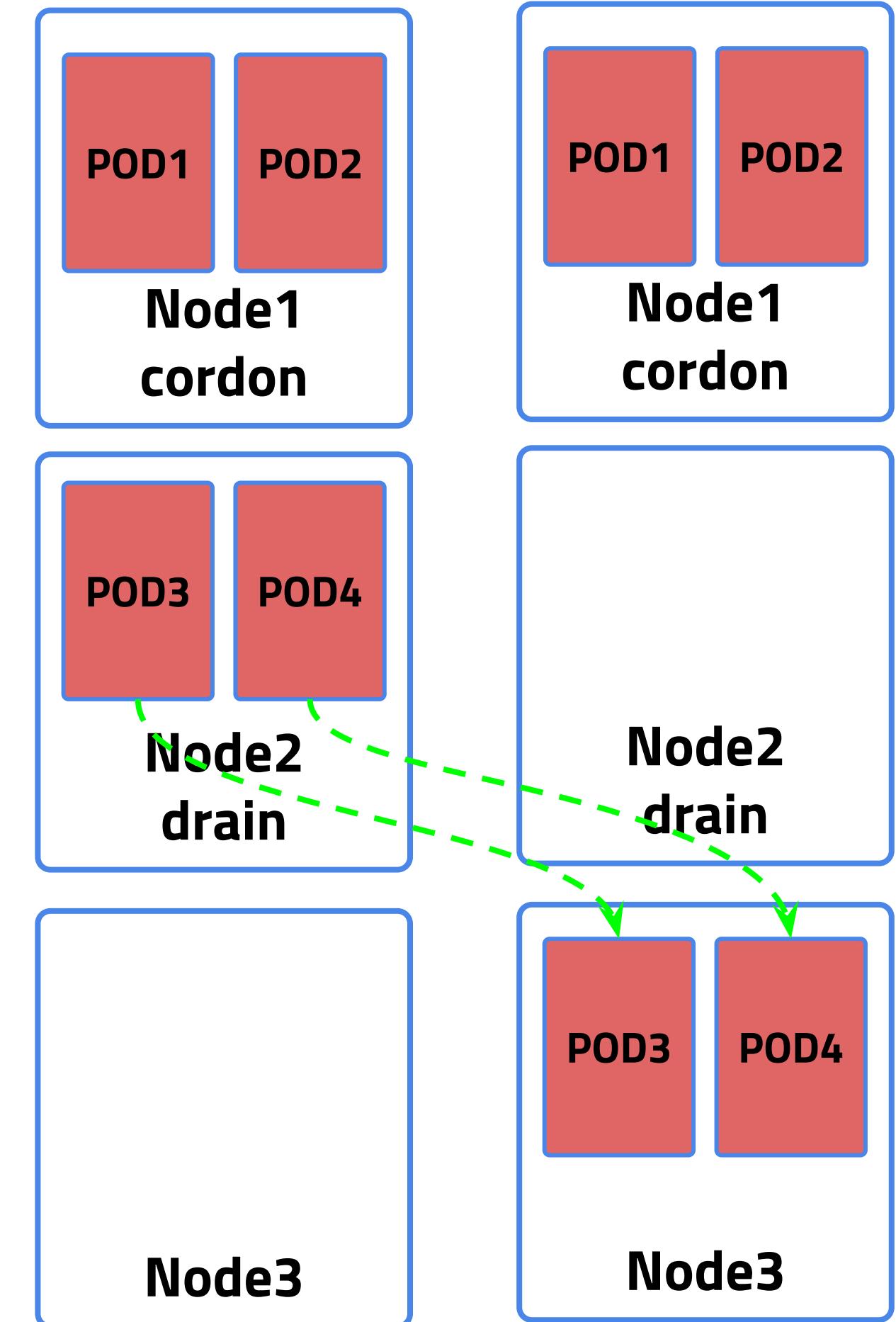
Si es necesario evacuar los pods de un nodo

```
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.

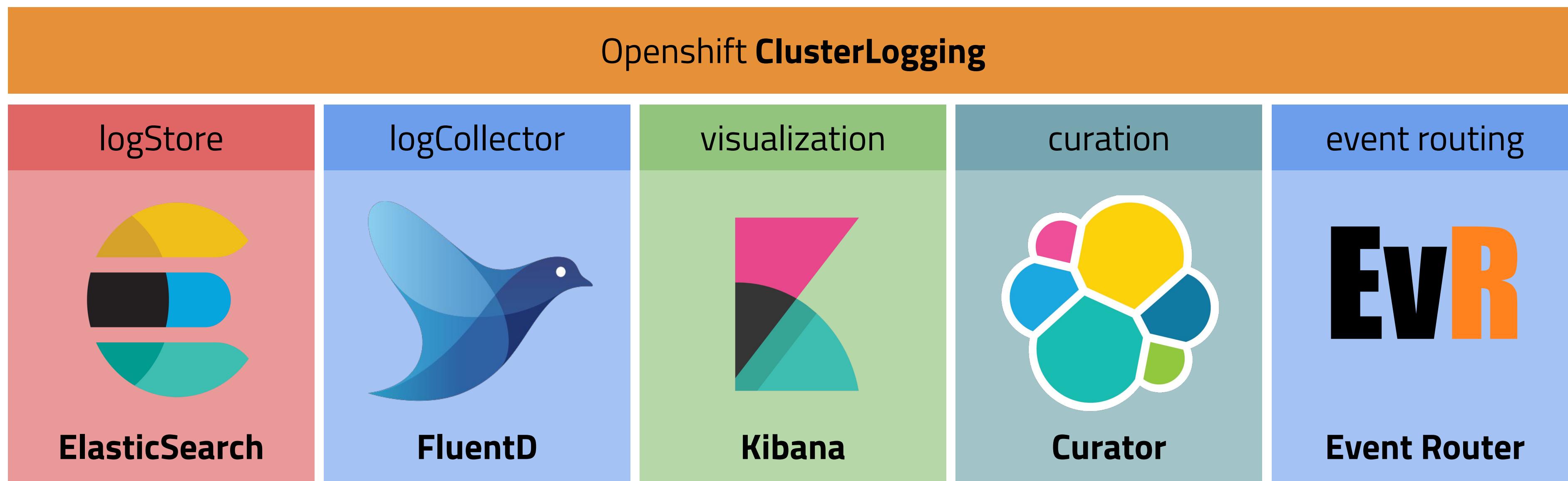


# Logging

*Semperti*

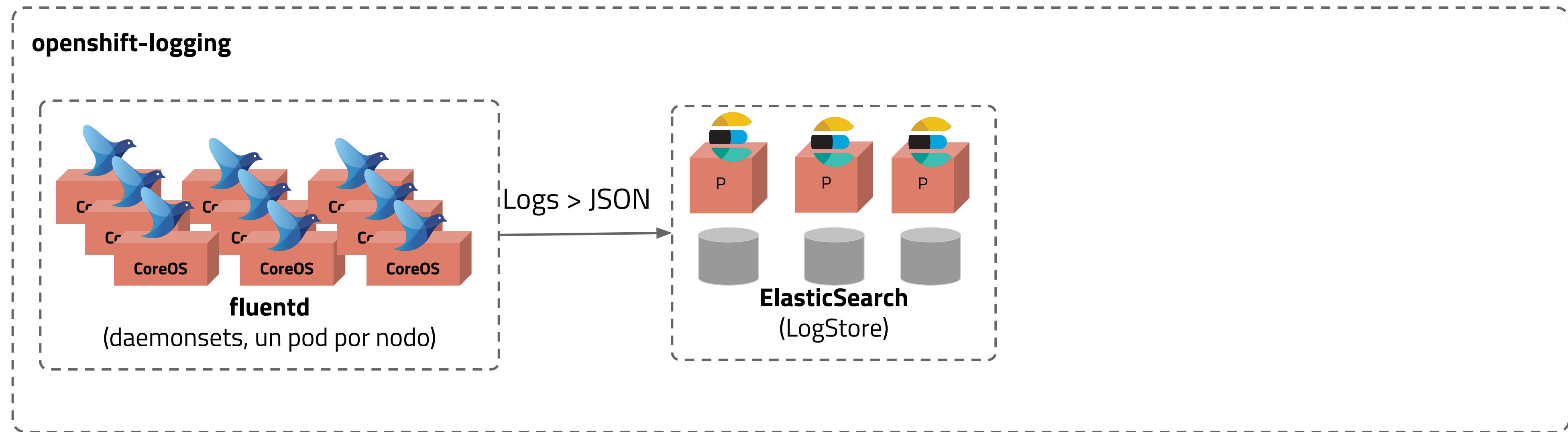
# Logging

## ClusterLogging



# Logging

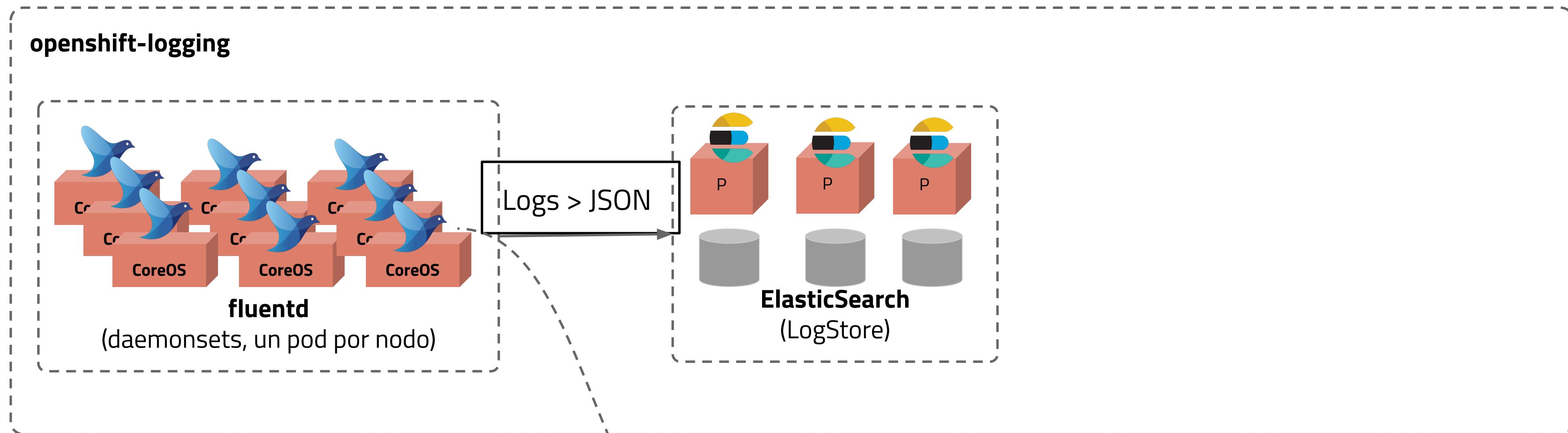
## ClusterLogging flujo de trabajo



**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



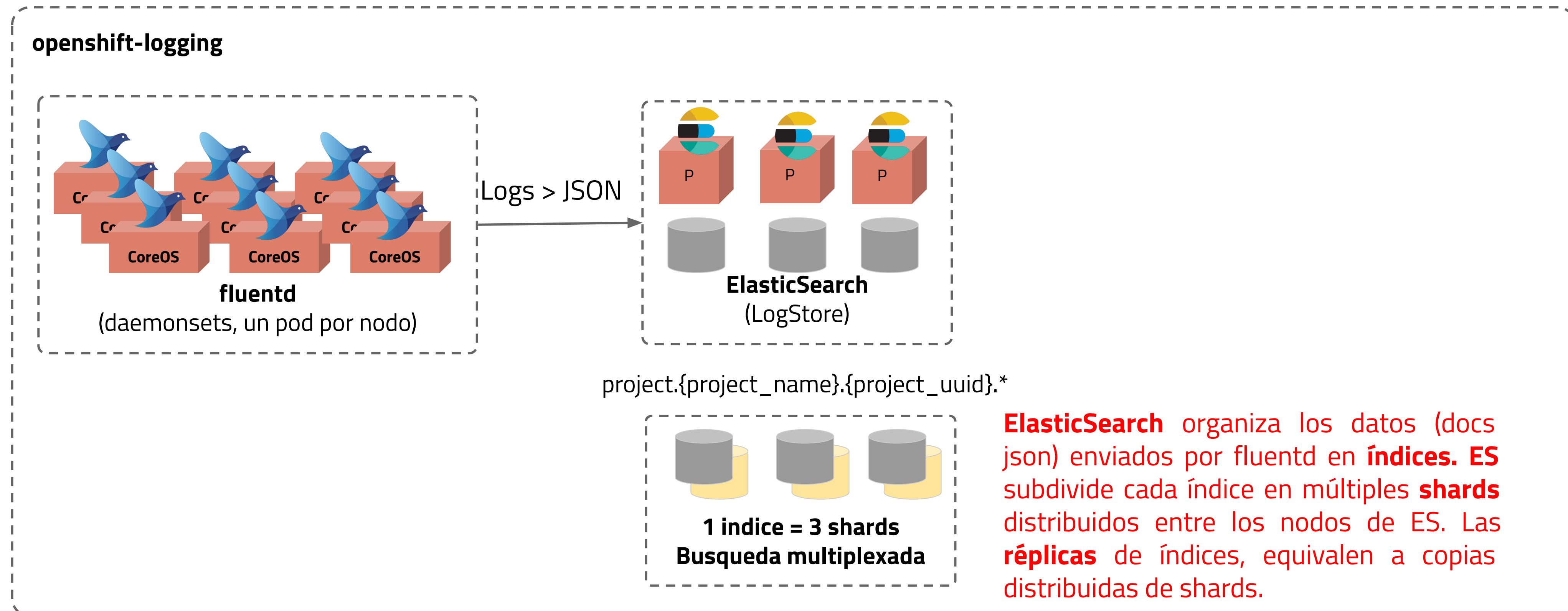
**FluentD** colecta los logs de cada uno de los nodos. **Journald** es la fuente de logs del sistema y provee información de **log message**, del **runtime de container** y **OCP**. FluentD transforma los logs en formato texto a documentos JSON para ser insertado en el logStore ES.



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

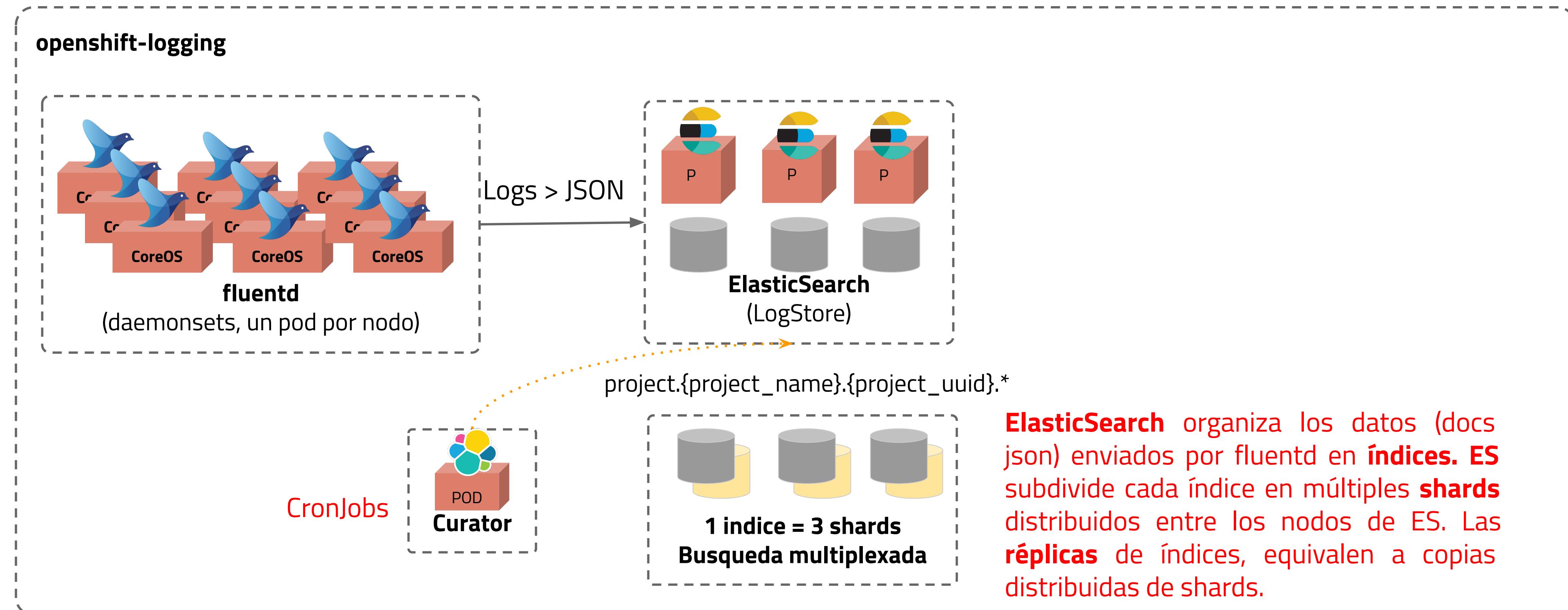
# Logging

## ClusterLogging flujo de trabajo



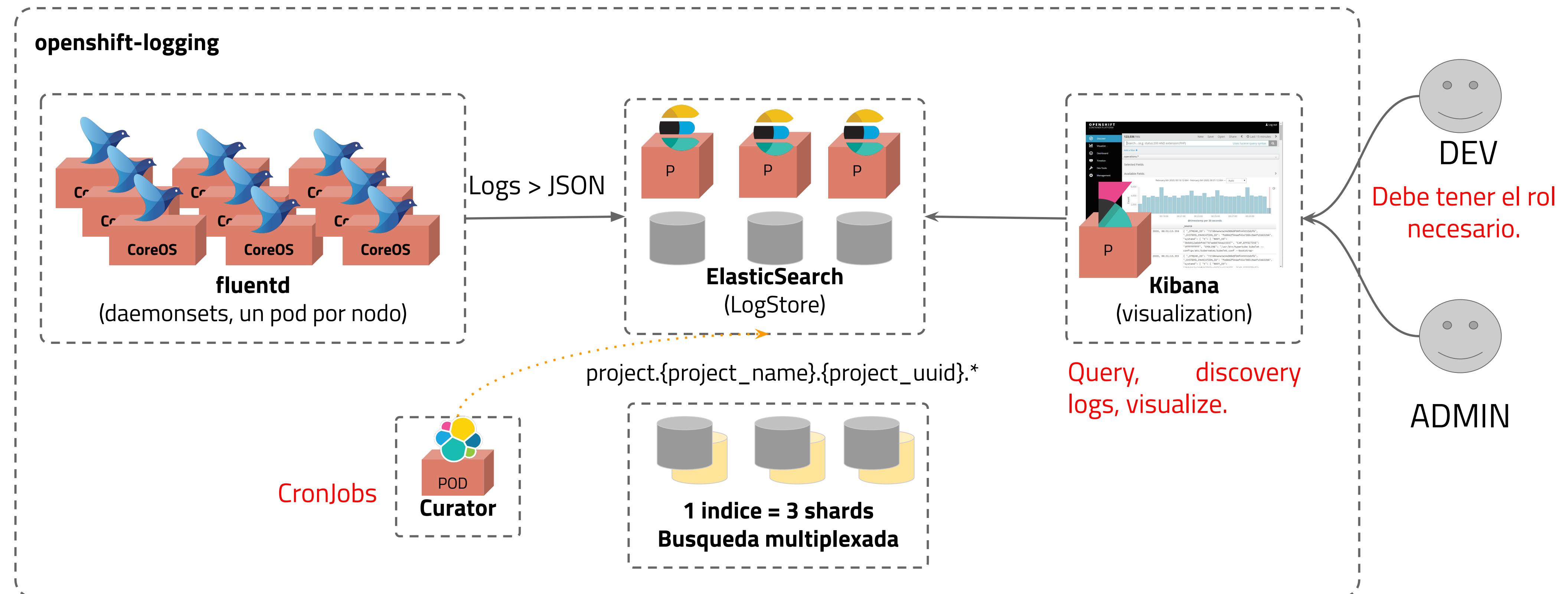
# Logging

## ClusterLogging flujo de trabajo



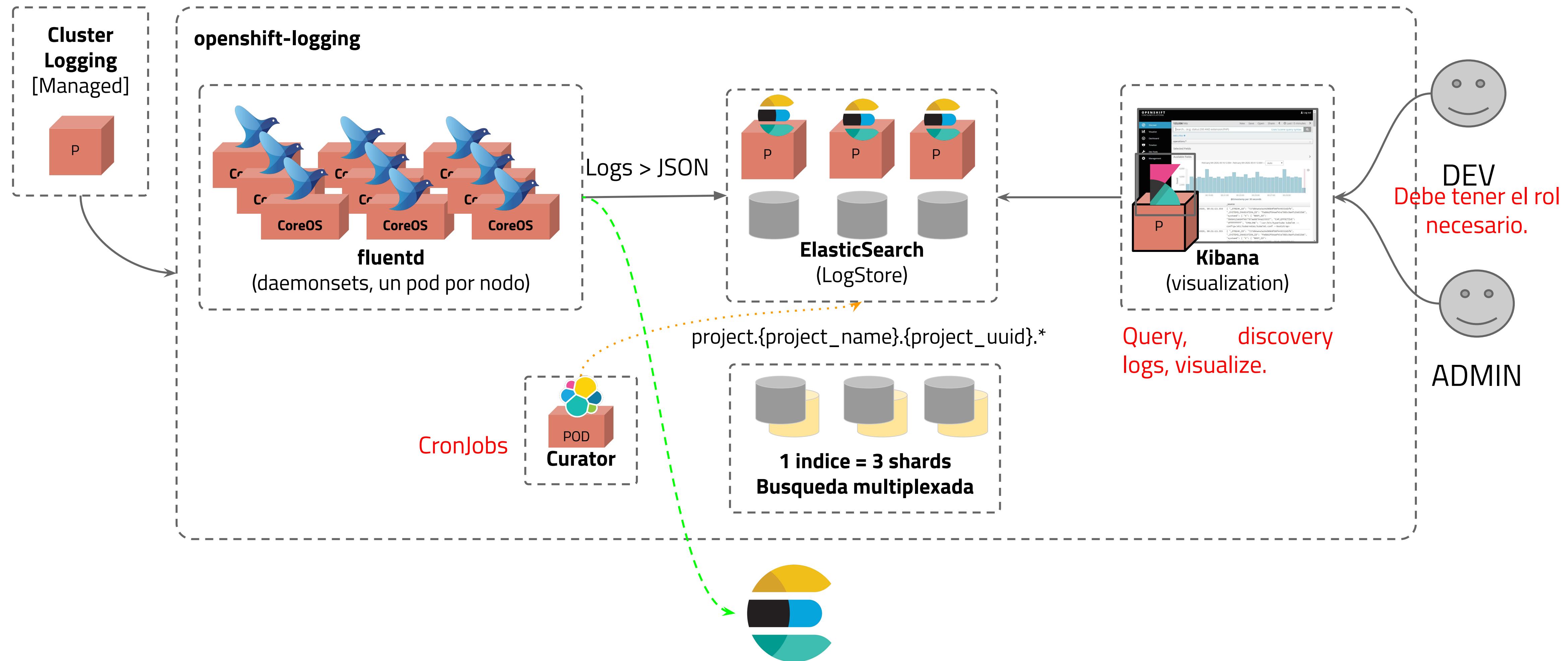
# Logging

## 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.

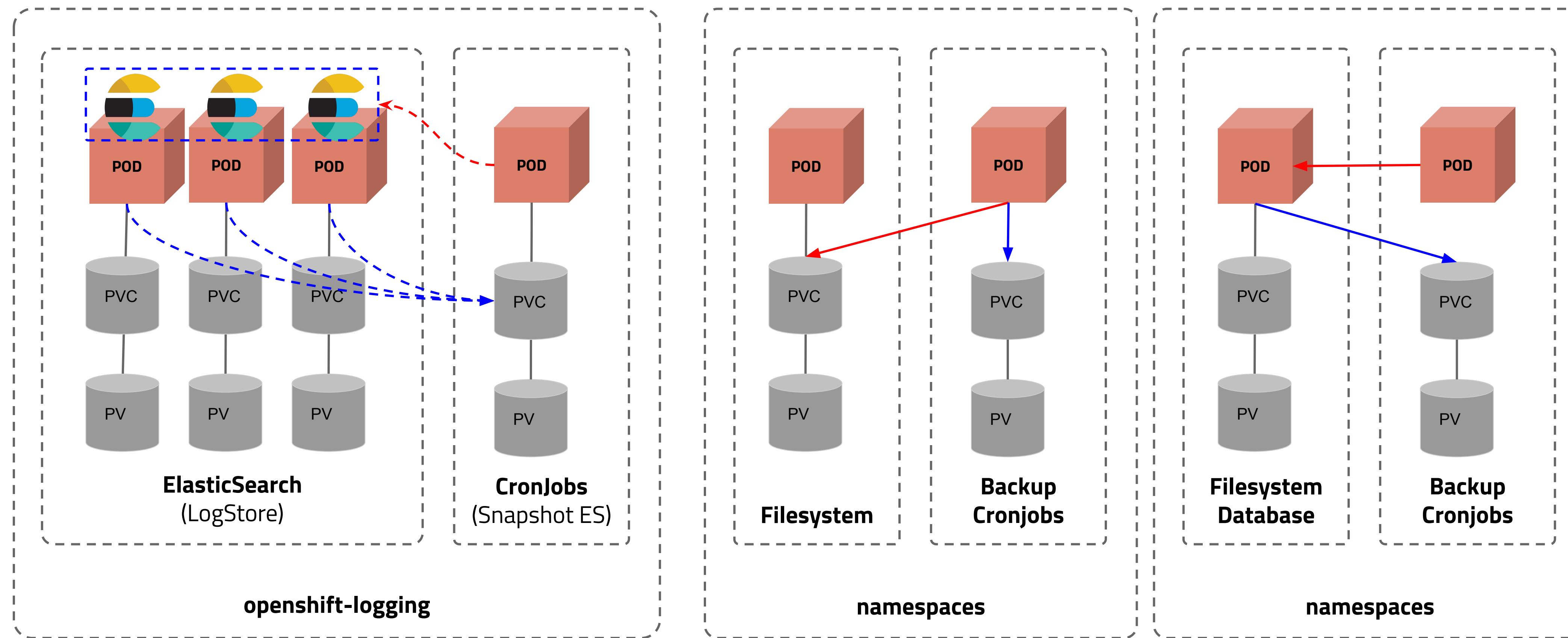
```
$ sudo /usr/local/bin/etcd-snapshot-restore.sh
/home/core/snapshot_db_kuberources_<datetimestamp>.tar.gz $INITIAL_CLUSTER
```

[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/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](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*

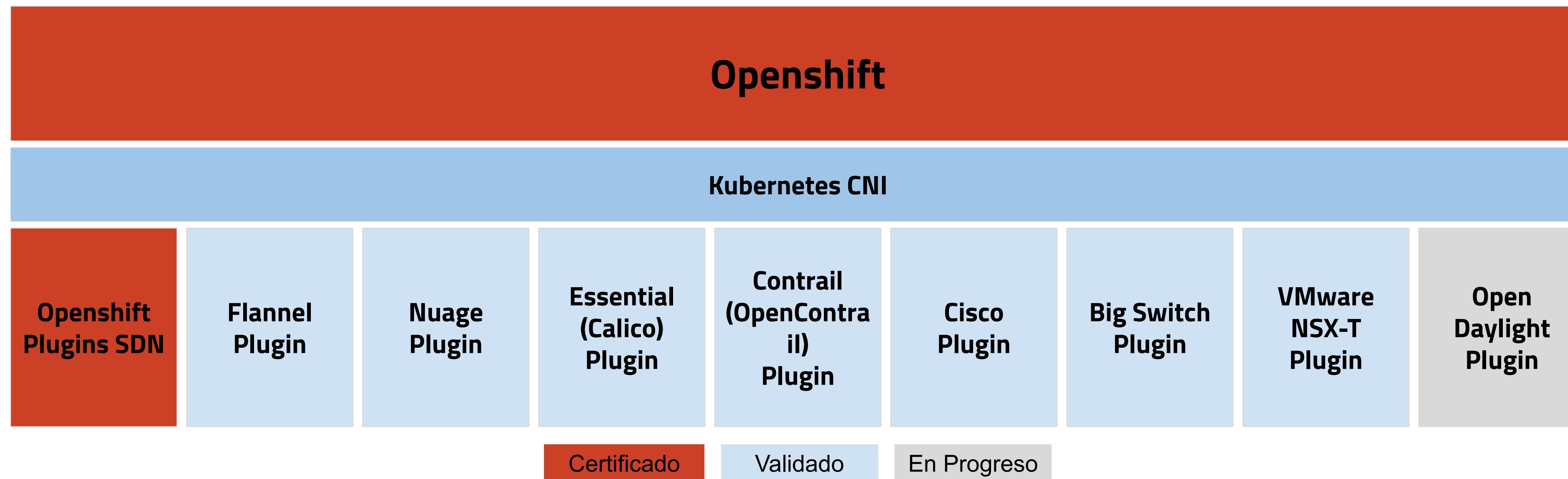
# Management Networking

## 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
```

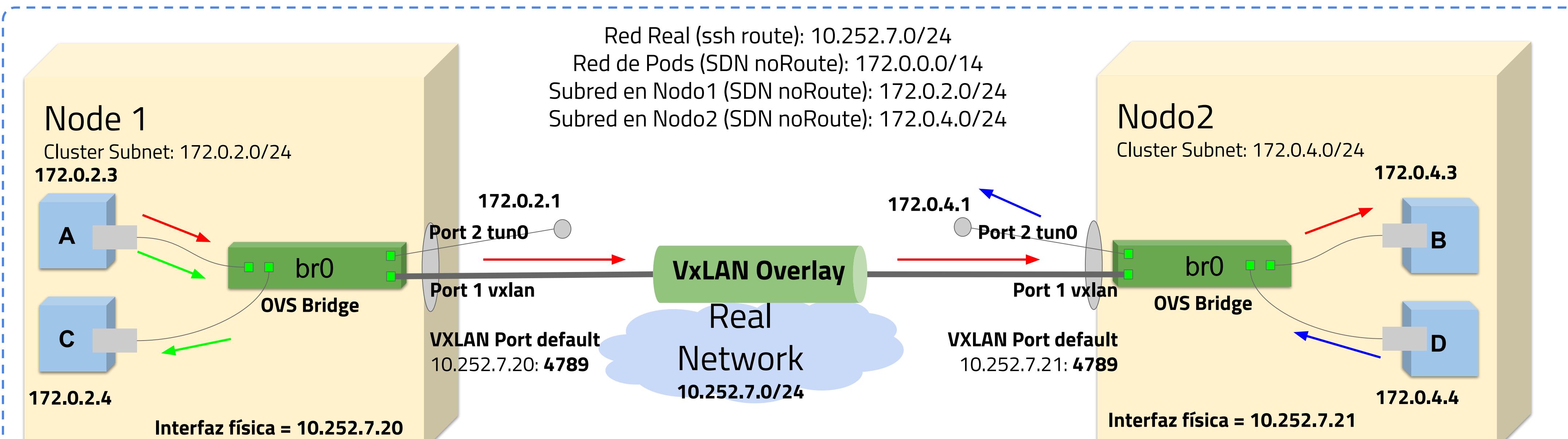
# Management Networking

## 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.

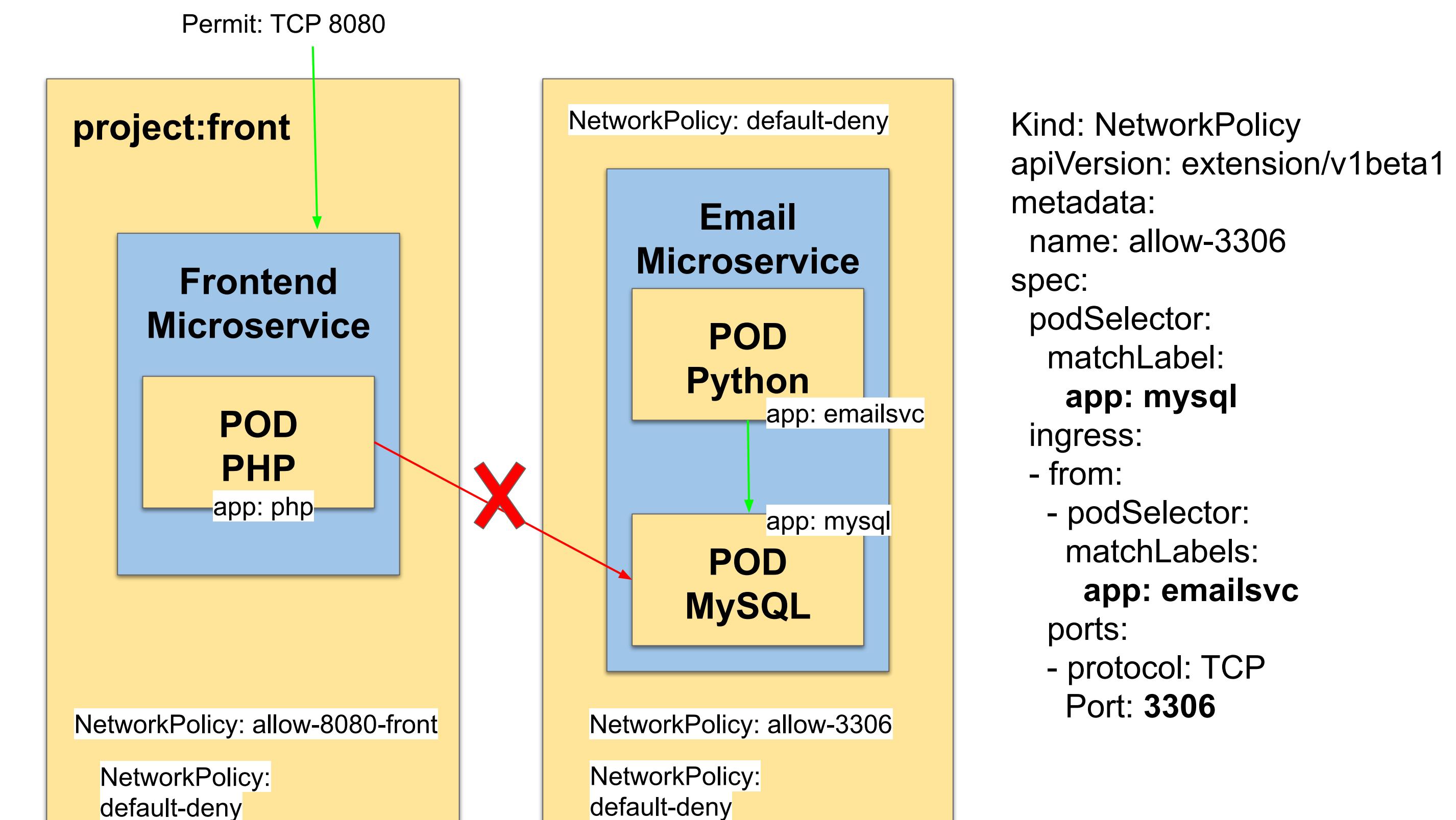
# Management Networking

## Network Policies

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

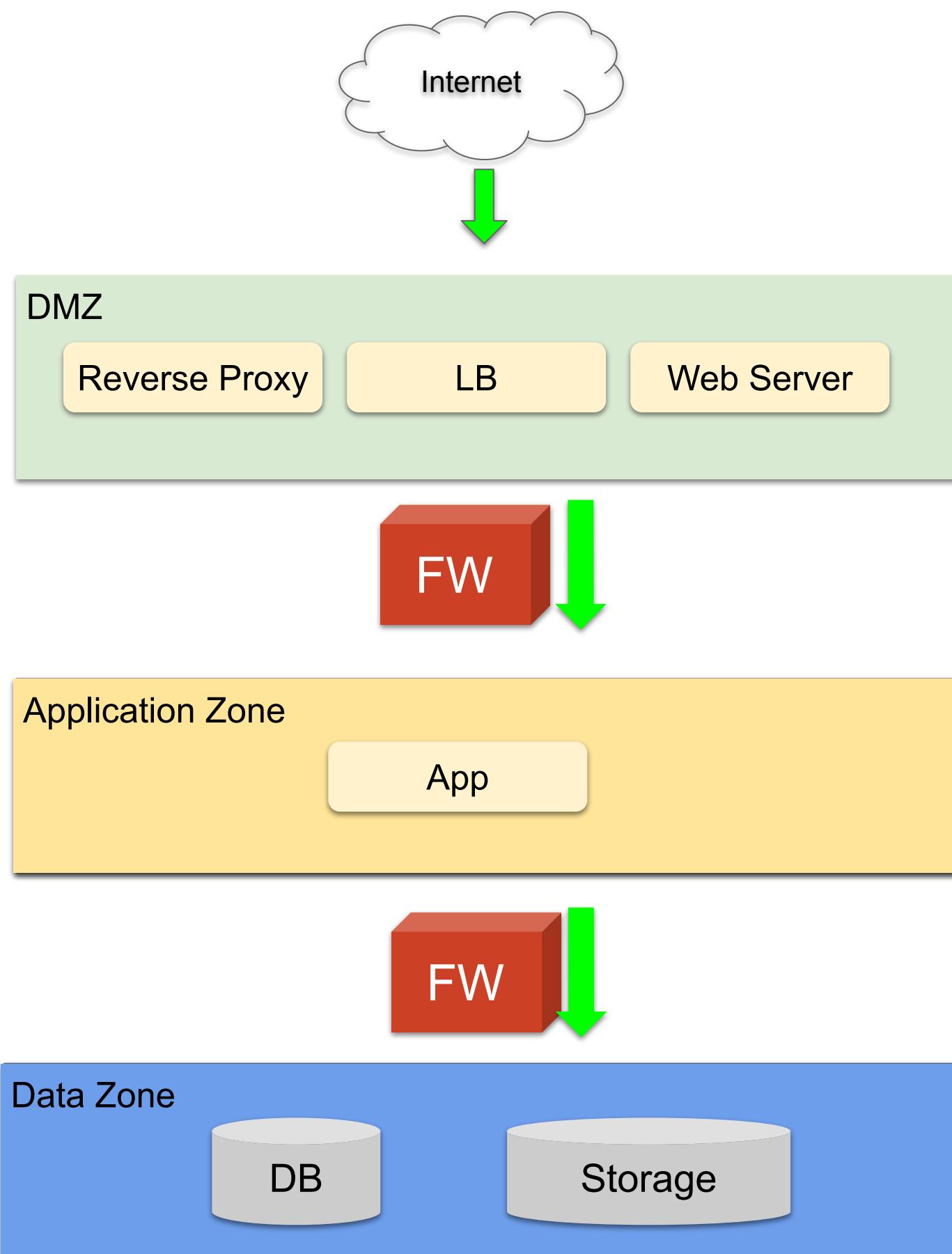
- **Microsegmentacion**

- 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

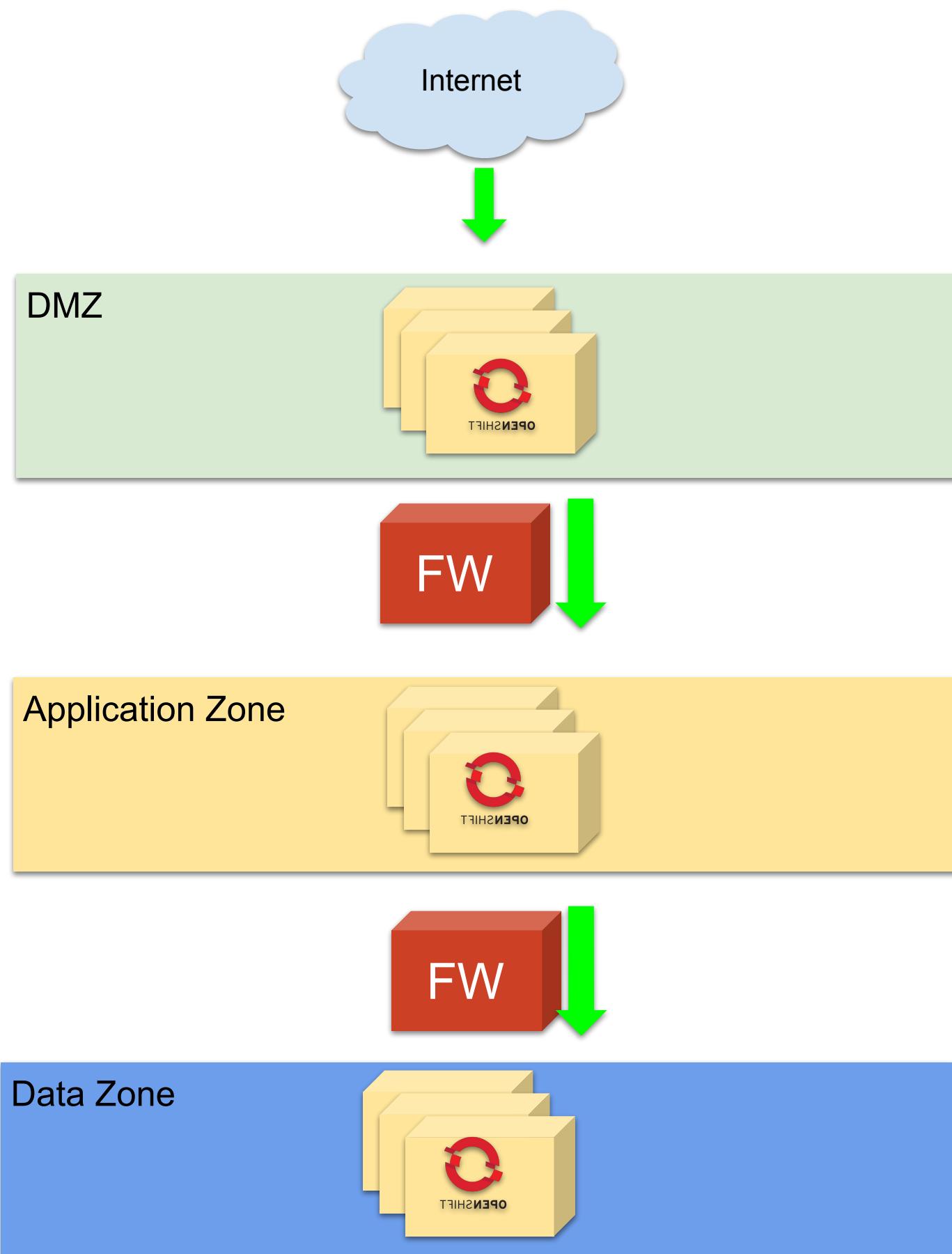
## Opción 1: Openshift Cluster por Zonas



- 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

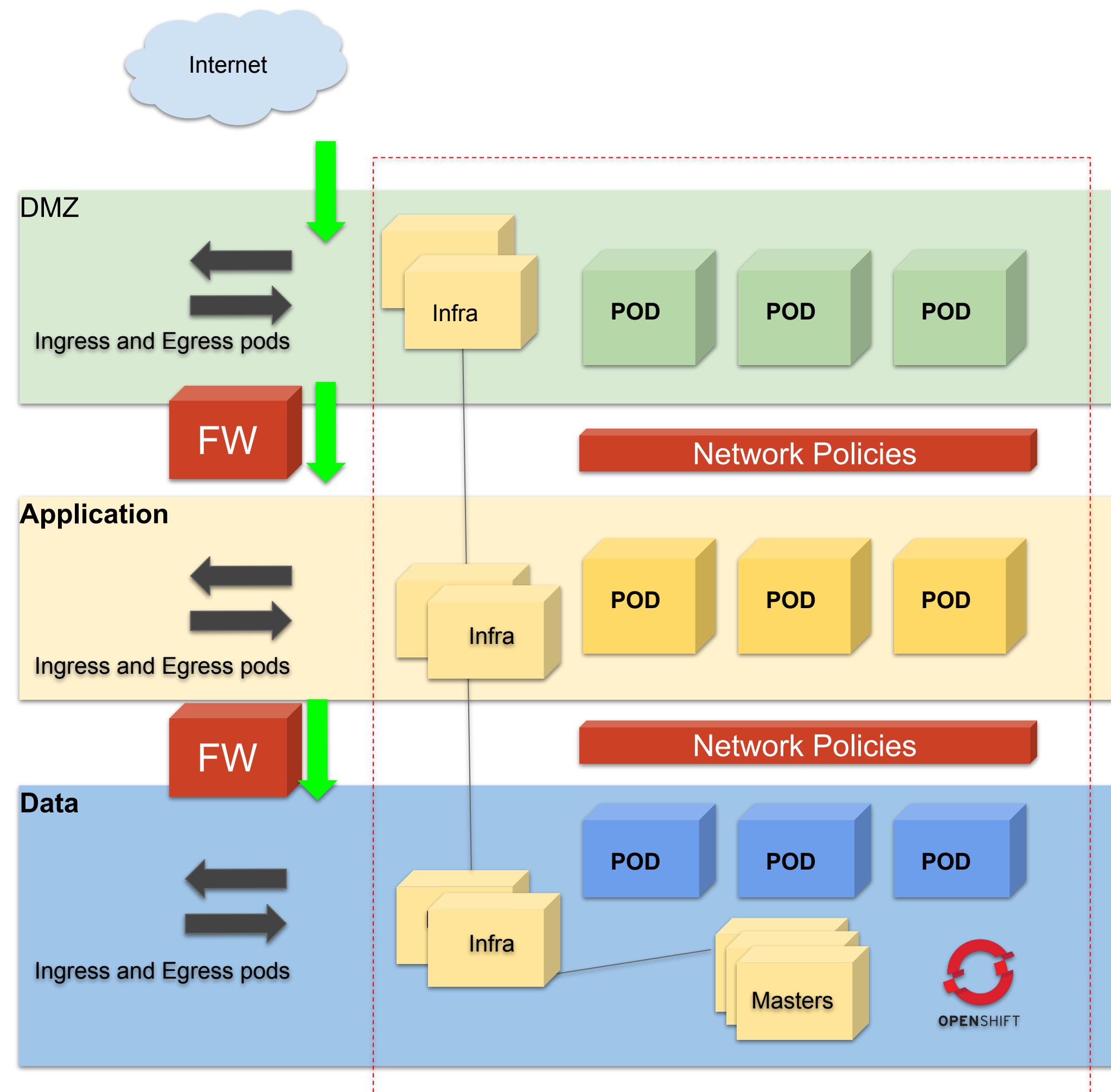
## Opción 1: Openshift Cluster por Zonas



- 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

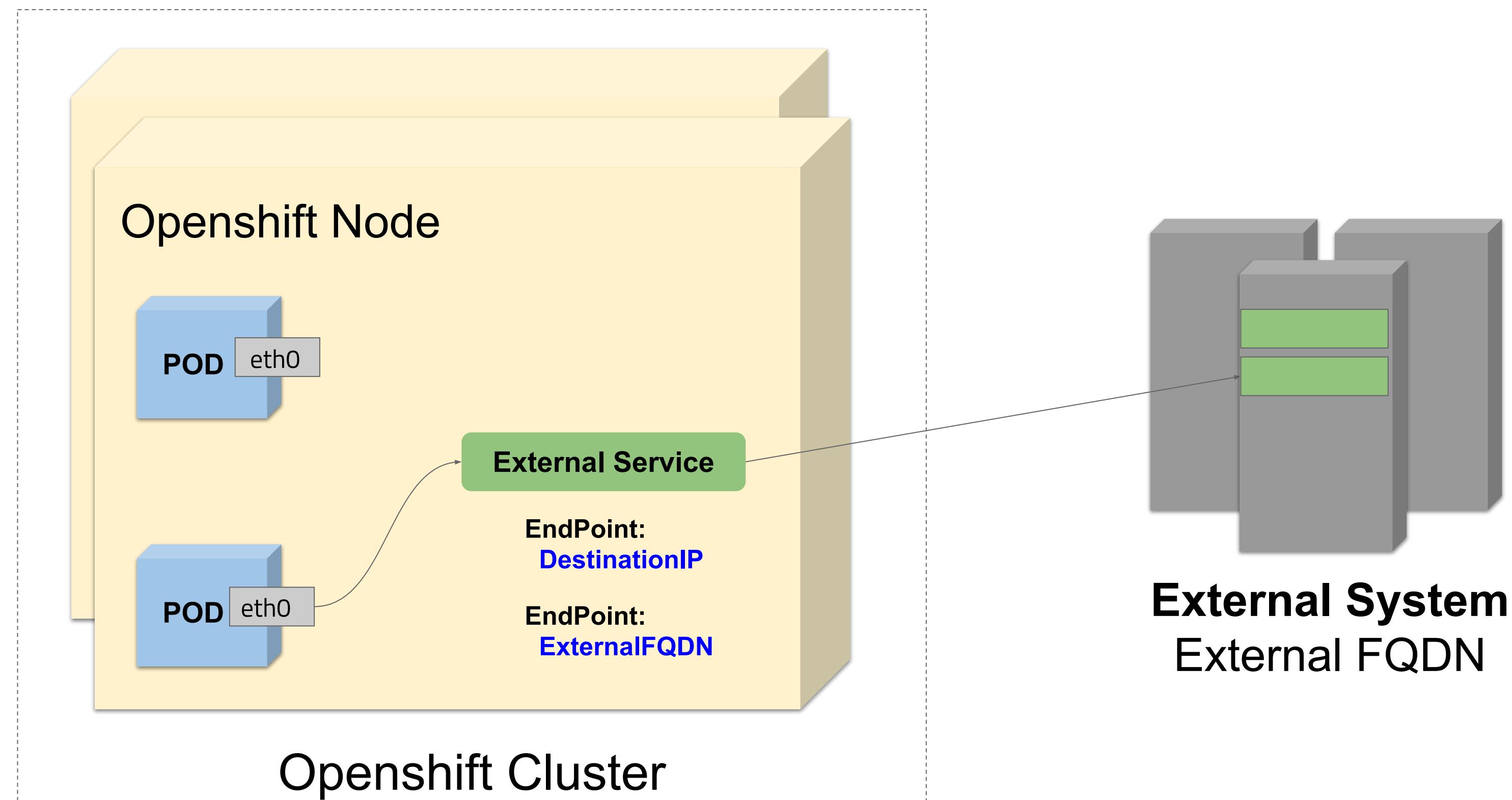
## Opción 2: Openshift Cluster cubriendo Multiples Zonas



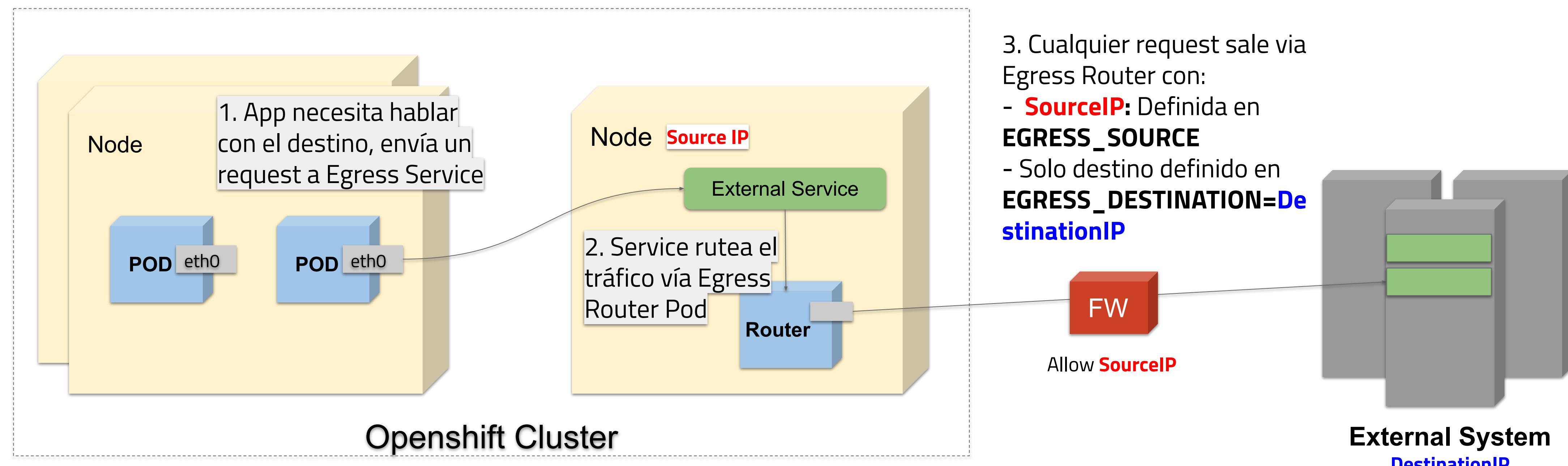
- Los pods de aplicación corren en un solo Openshift Cluster microsegmentado con Network Policies.
- Nodos de infra en cada zona corren pods de Ingress y Egress para cada zona.
- Si es requerida aislación física de pods, se puede utilizar el selector de nodos (nodeSelector).
- Los nodos masters se encuentran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

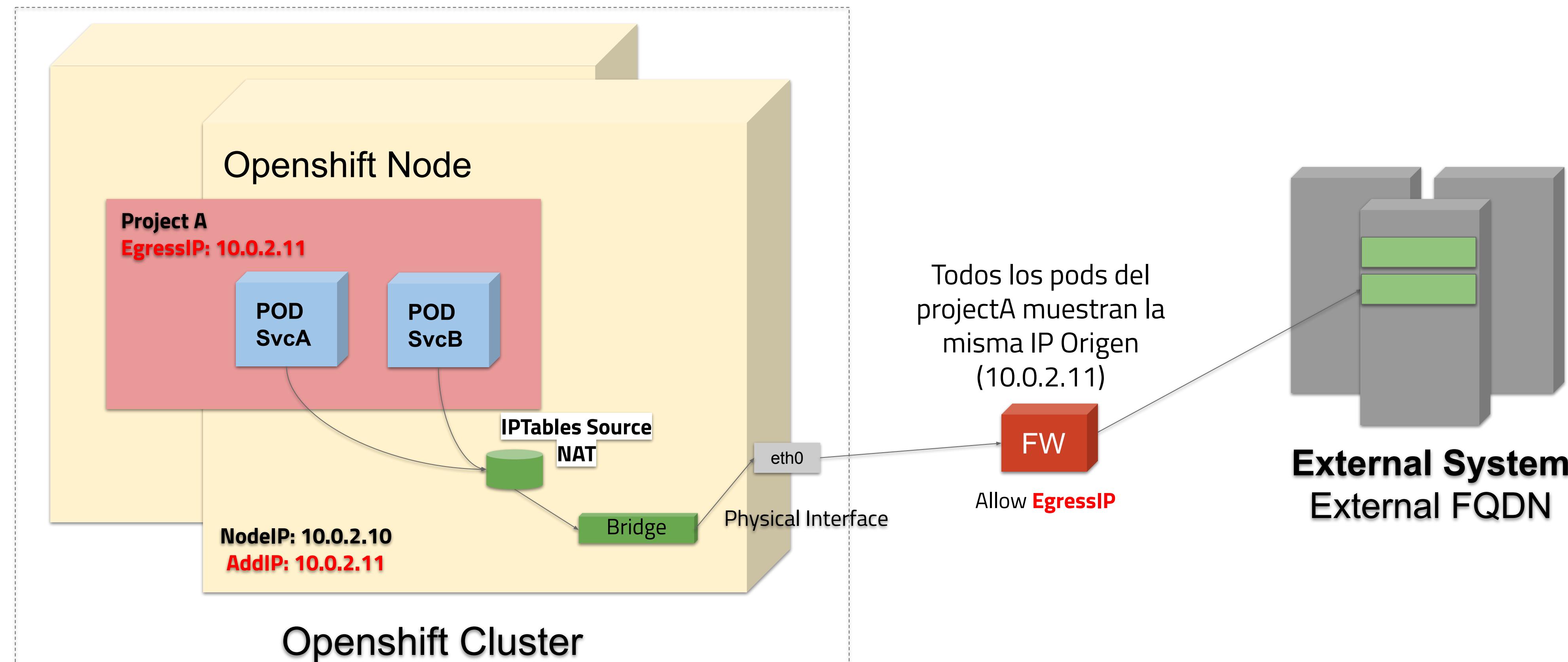


# Egress Router



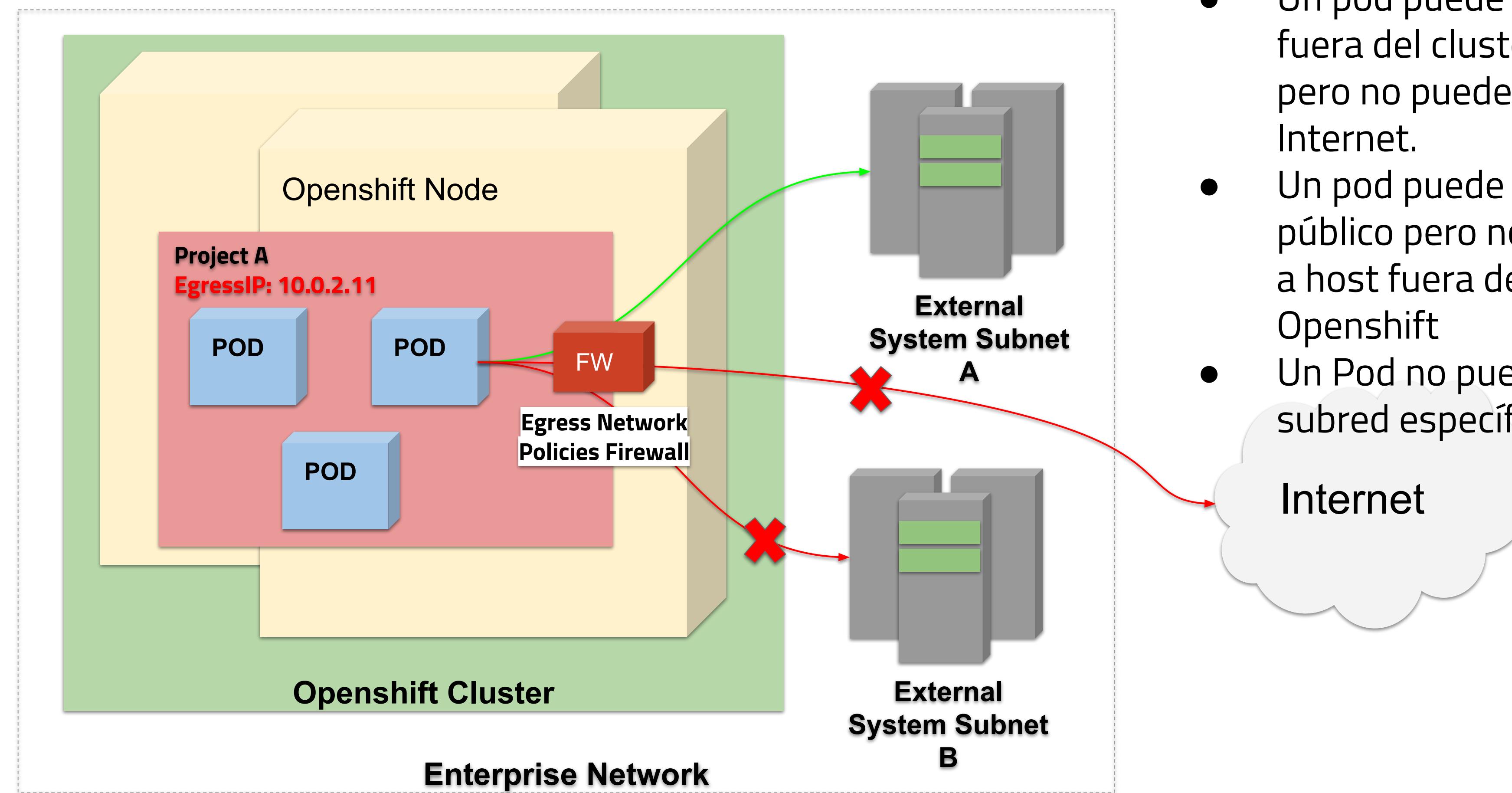
# Egress vía Static IP

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

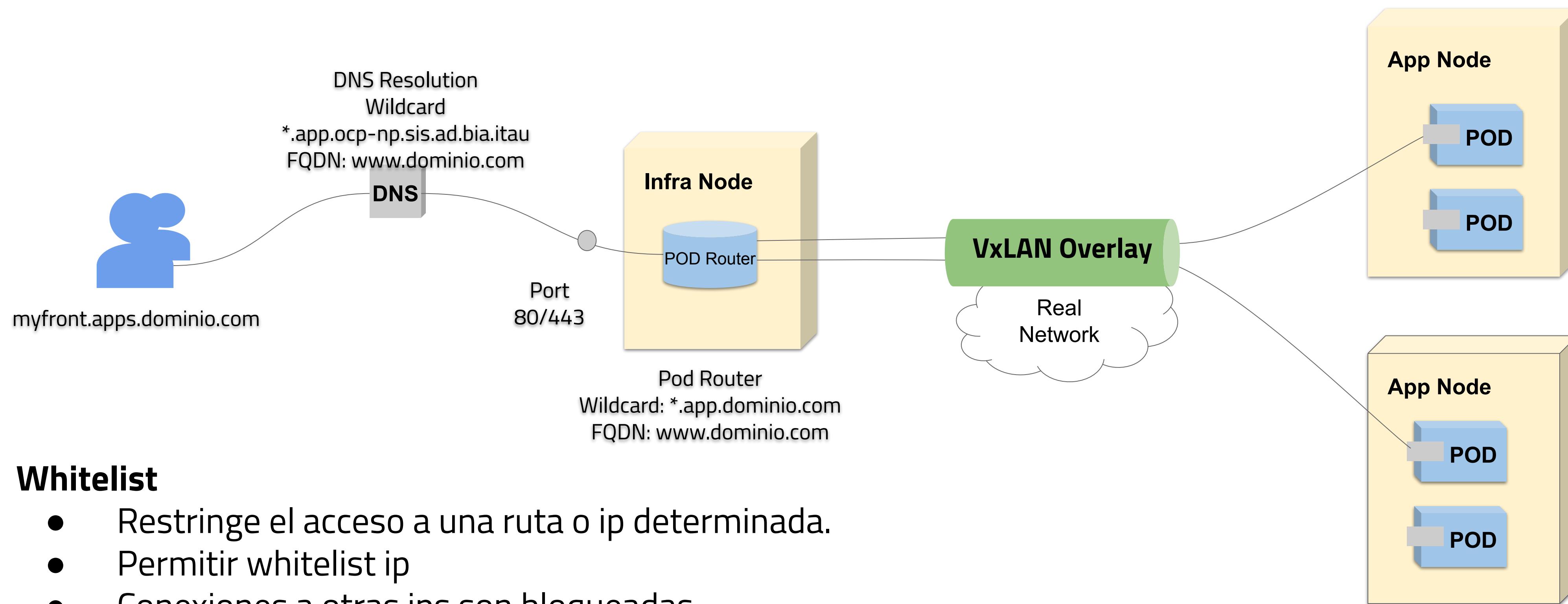
Un cluster admin puede limitar la salida de los pods a determinados sistemas aplicando **Network Policies**



- Un pod puede hablar a host fuera del cluster de Openshift pero no puede conectar a Internet.
- Un pod puede tener acceso público pero no puede conectar a host fuera del Cluster de Openshift
- Un Pod no puede llegar a una subred específica.

# Openshift Ingress Router

## Traffic Flow

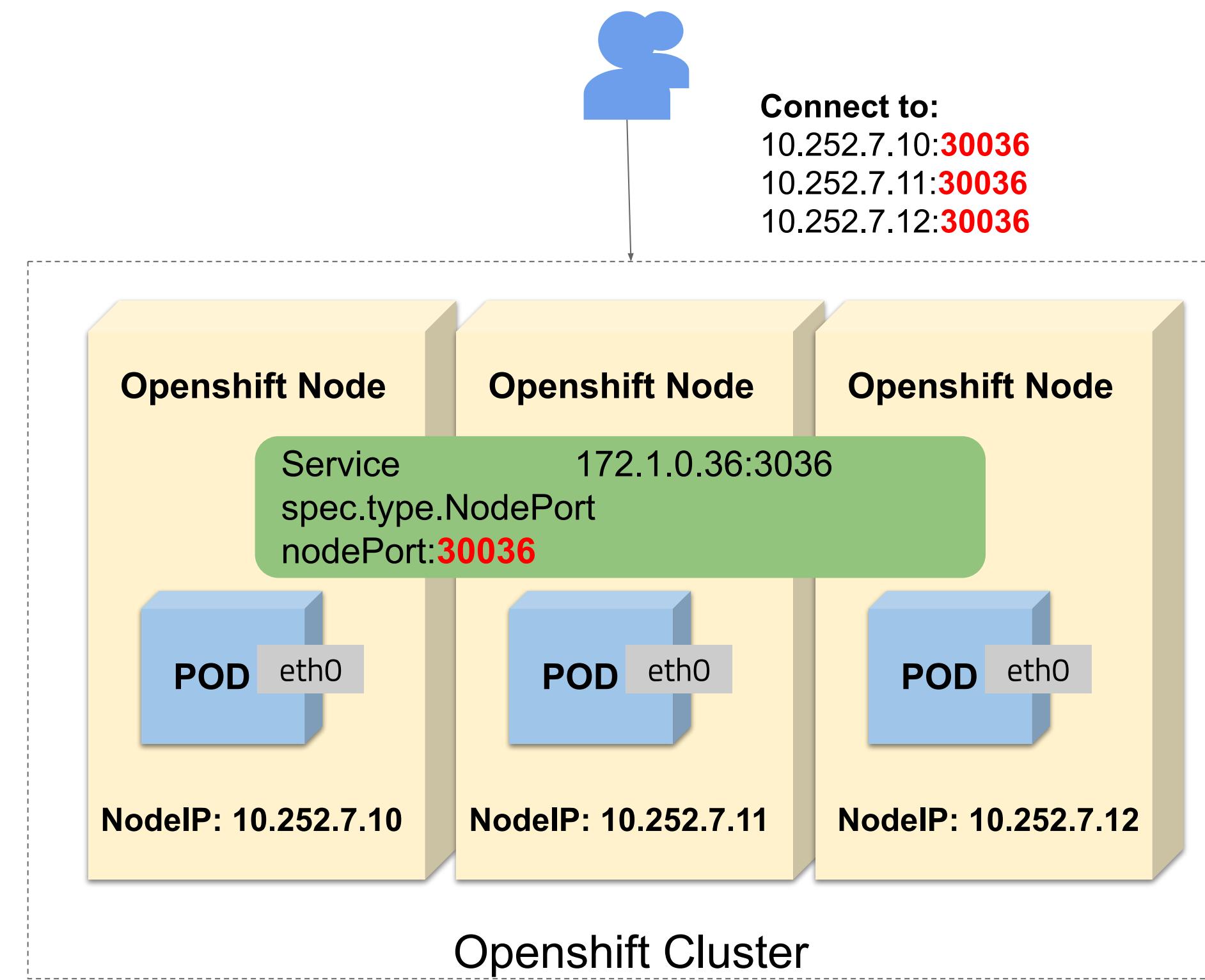


### Whitelist

- 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.

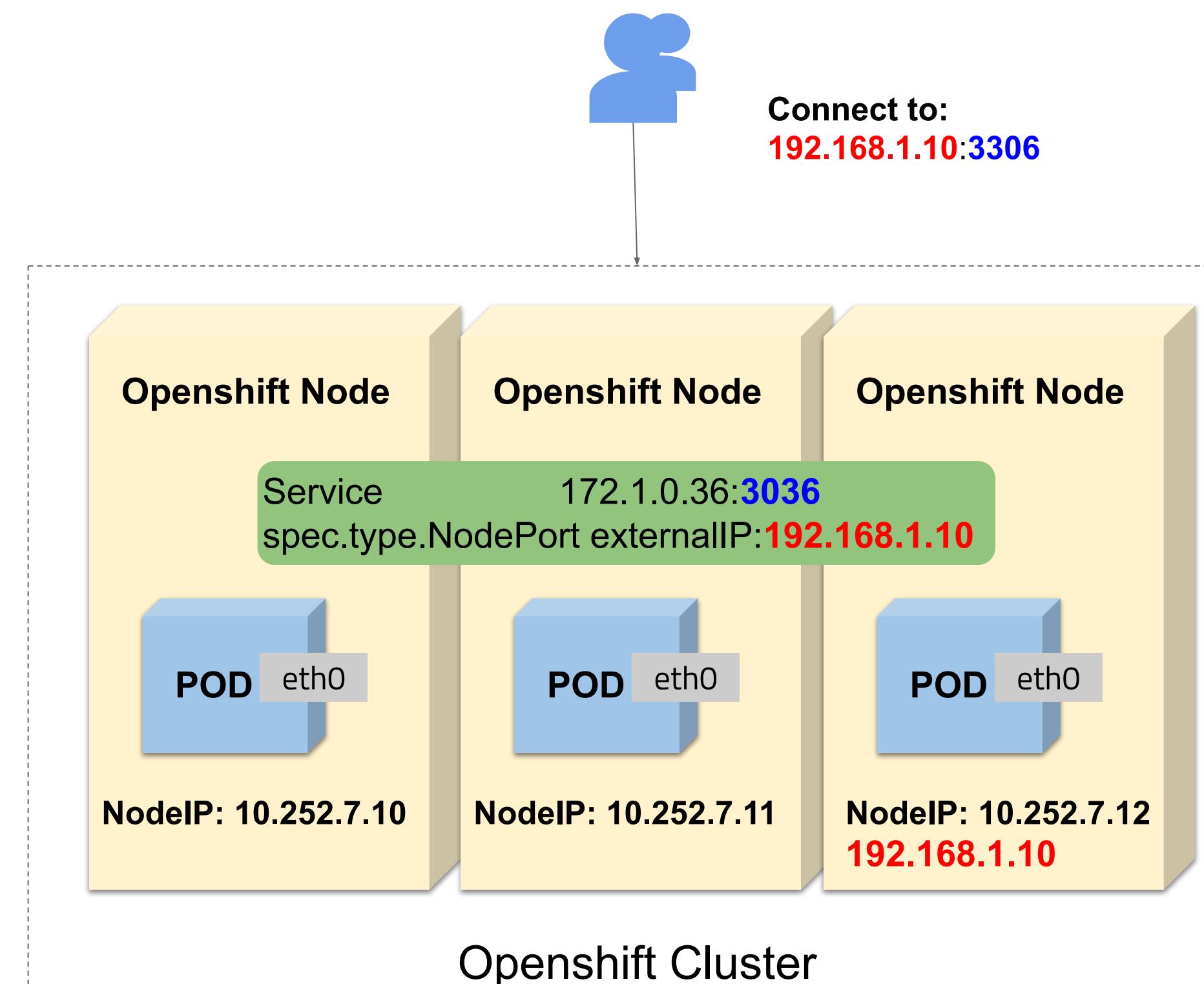
Puerto es random y asignado dinámicamente entre 30000-32767.

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



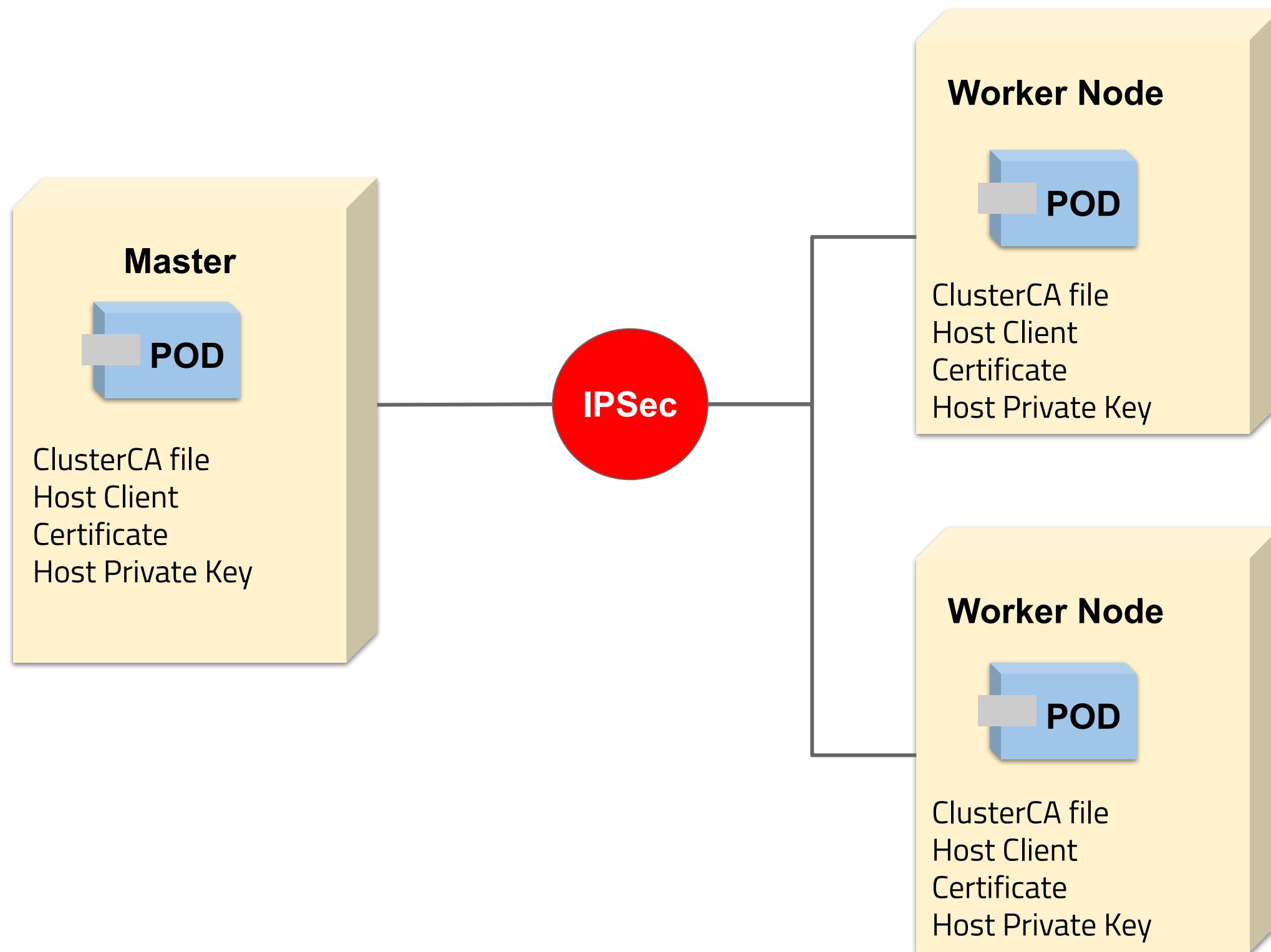
Admin define una External IP, puede ser un rango y la asignación se hace dinámica.

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

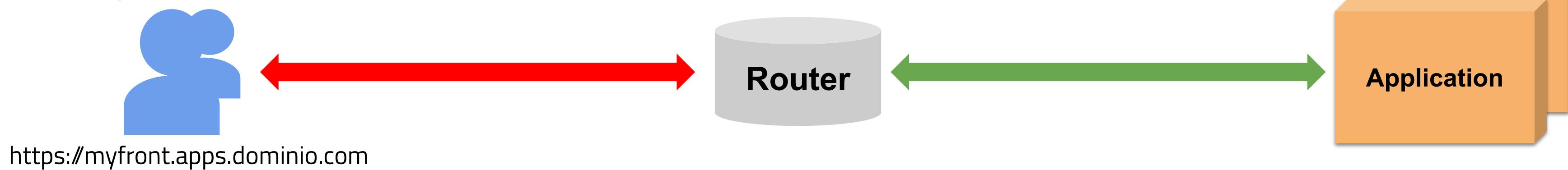


Toda la comunicación entre nodos del cluster es segura con IPsec.

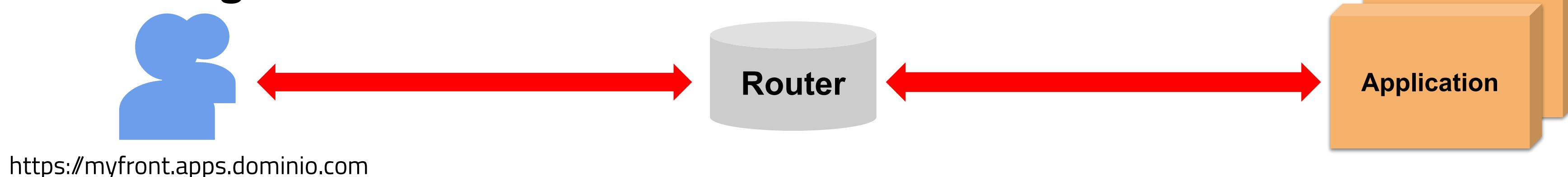
- Encripción entre Master y Worker Nodes (L3)
- OpenshiftCA y Certs

# Application Layer - Openshift Router SSL

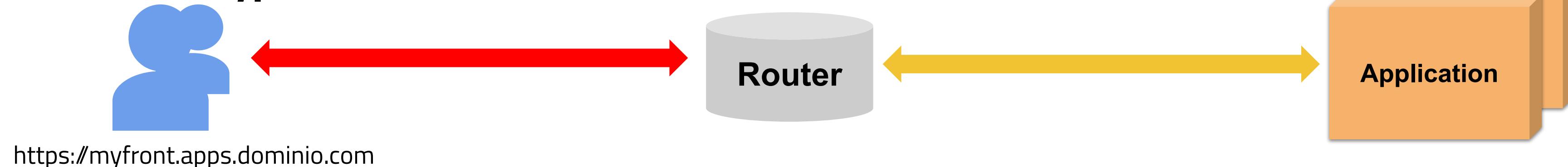
## Edge Termination



## 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

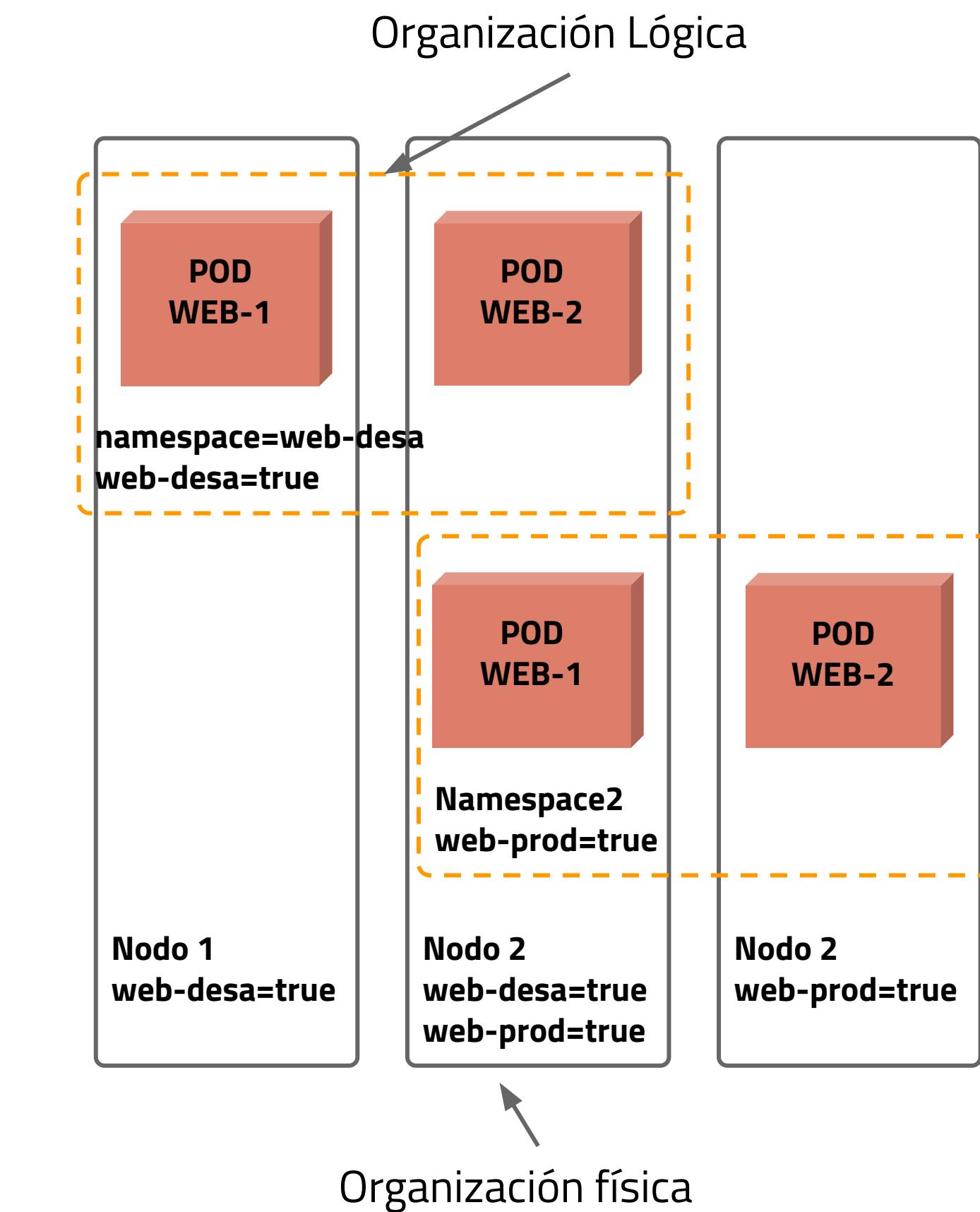
- Construir aplicaciones con componentes **escalables y con capacidad de sostener la pérdida de PODs.**
  - Al menos 2 pods por componente
    - Apps Web, message tier, datastore, etc.

### 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.

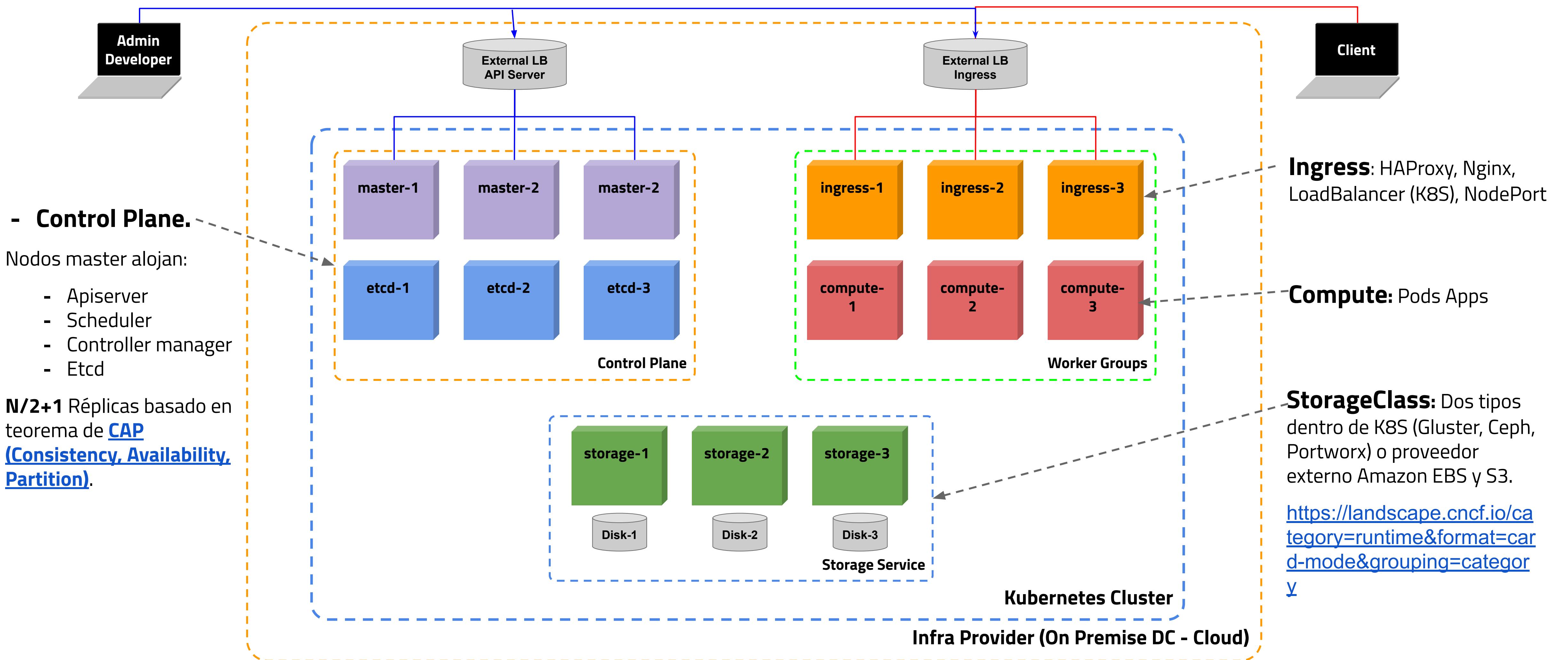
<https://docs.openshift.com/container-platform/4.3/applications/application-health.html>

<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-multicluster/>

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](mailto:gonzalo.acosta@semperti.com)>

*Semperti*