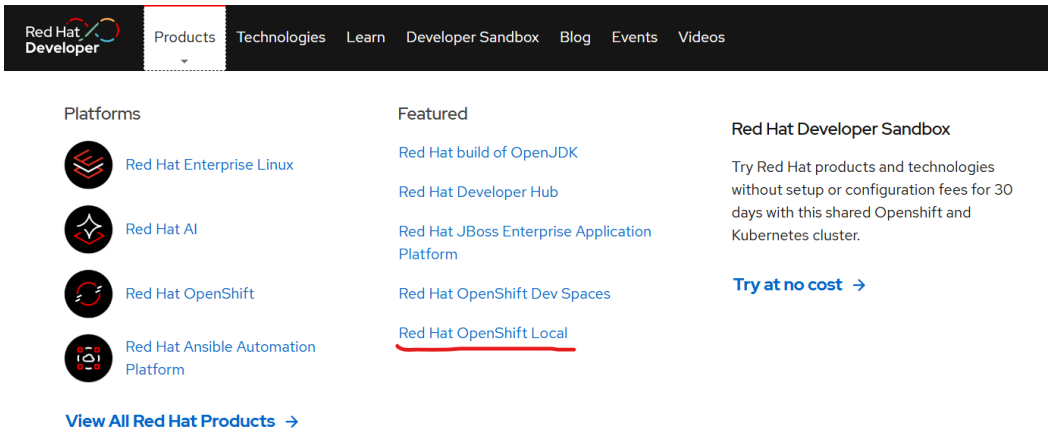
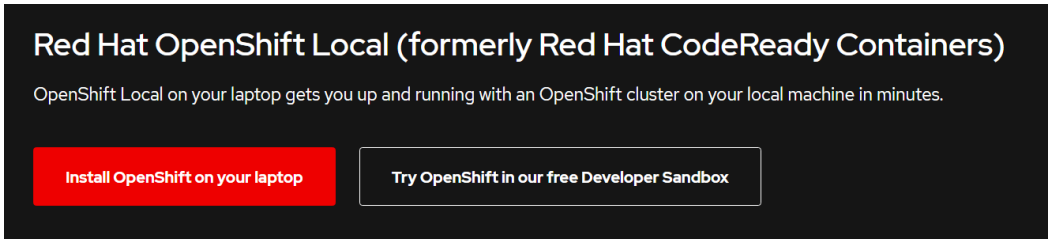
OpenShift

Descargar programa

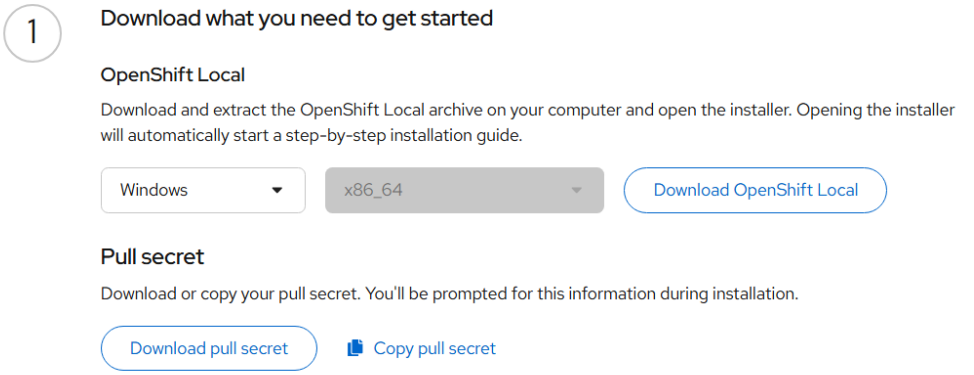
Para descargarlo primero hay que registrarse en la pagina de redhat, seguir los pasos y luego descargar el programa



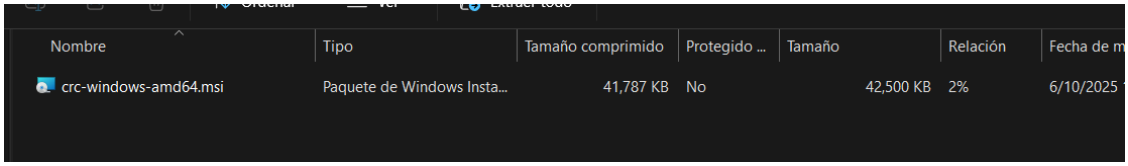
Boton rojo



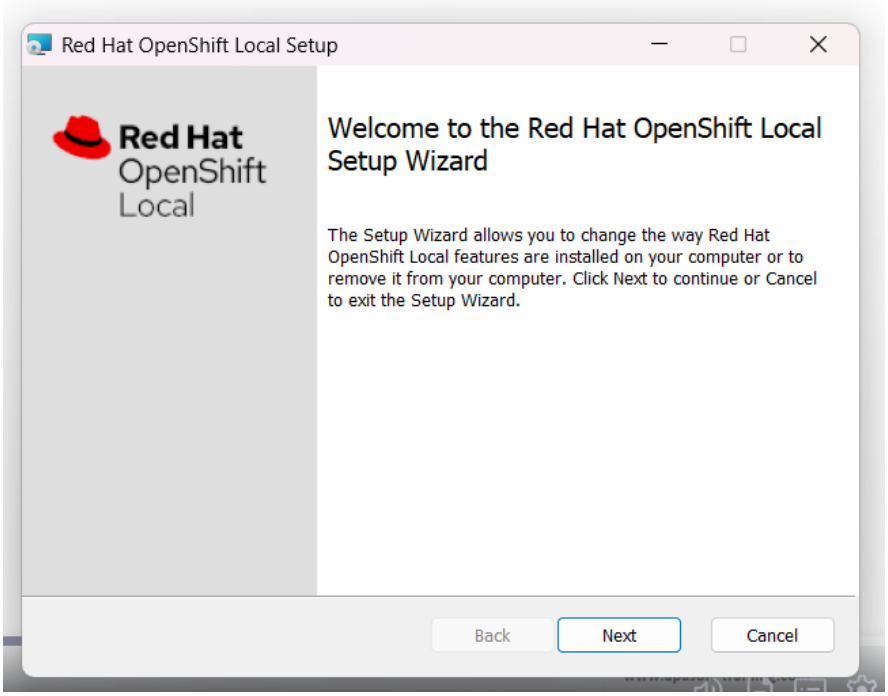
Descargar archivo



Instalar el programa



Seguir la instalacion con next, next, ...



Configurar el setup en modo administrador (esto puede demorar mas de 1h)

```
crc setup
```

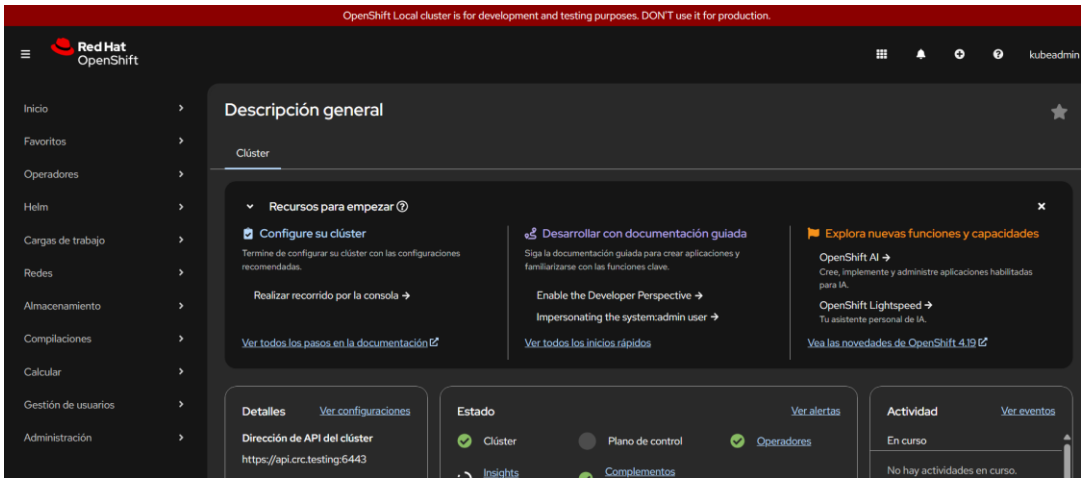
Iniciar openshift sin administrador (demora unos 10 min)

```
crc start
```

Credenciales para el servicio (importante: copiar esta informacion y guardarla)

```
The server is accessible via web console at:
  https://console-openshift-console.apps-crc.testing

Log in as administrator:
  Username: kubeadmin
  Password: au5kW-d7V8B-EHvZr-MUwct

Log in as user:
  Username: developer
  Password: developer

Use the 'oc' command line interface:
  PS> & crc oc-env | Invoke-Expression
  PS> oc login -u developer https://api.crc.testing:6443
```

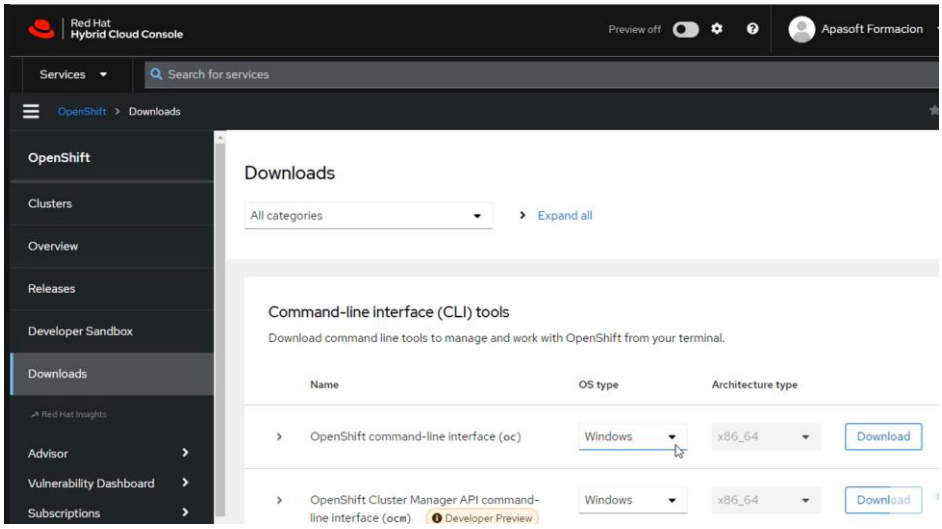
Aceptar el modo inseguro del navegador



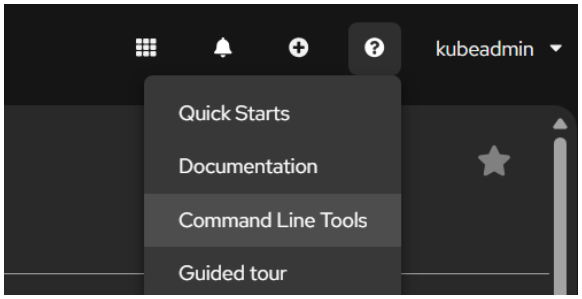
Comandos basicos

```
crc status
crc stop
crc delete
crc start
crc console
crc config
```

Instalar cliente OpenShift



Otra opcion



Crear token de acceso

Es necesario loguearse con un usuario pueder como kubeadmin o developer

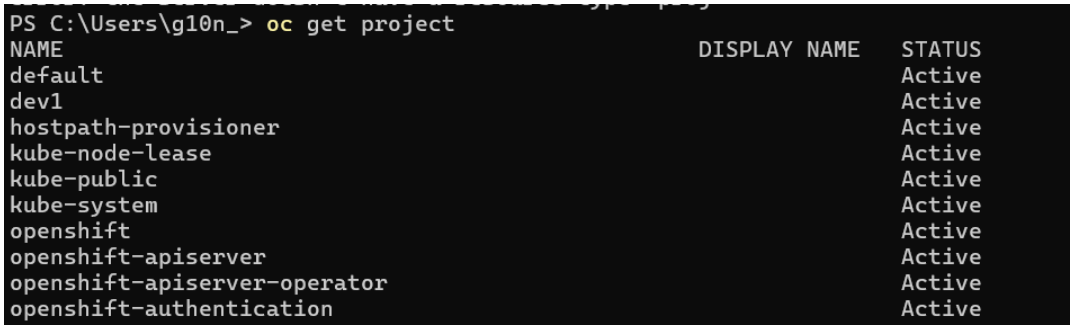

```
oc login --token=sha256~0EIX0xNl9Q_04-iQPRUGf4989OXmmD4TTOL7jDJ6CqE --server=https://api.crc.testing:6443
```

Proyectos

Los proyectos es lo mismos que los namespaces de kubernetes pero con mas funcionalidades como nuevos objetos de openshift

```
oc new-project dev1
```


```
oc get project
```



Setear namespace/project

```
oc config set-context --current --namespace=dev1
```

Configurar cliente OC

```
[Environment]::SetEnvironmentVariable("PATH", $env:PATH + ";C:\Users\g10n_\.oc", [EnvironmentVariableTarget]::User)
$env:PATH = [Environment]::GetEnvironmentVariable("PATH", [EnvironmentVariableTarget]::User)
if (-not ($env:PATH -like '*\.oc*')) { $env:PATH += ';C:\Users\g10n_\.oc' }
oc version
```

Configurar cliente OC en WSL (Linux)

```
# Actualizar el sistema WSL
sudo apt update && sudo apt upgrade -y

# Instalar curl y wget si no están instalados
sudo apt install curl wget -y

# Descargar OpenShift CLI (oc) para Linux
wget https://mirror.openshift.com/pub/openshift-v4/clients/ocp/stable/openshift-client-linux.tar.gz

# Extraer el archivo descargado
tar -xzf openshift-client-linux.tar.gz

# Crear directorio para binarios locales si no existe
mkdir -p ~/.local/bin

# Mover el binario oc al directorio local
mv oc ~/.local/bin/

# Hacer el binario ejecutable
chmod +x ~/.local/bin/oc

# Agregar el directorio al PATH en .bashrc
echo 'export PATH="$HOME/.local/bin:$PATH"' >> ~/.bashrc

# Recargar el archivo .bashrc
source ~/.bashrc

# Verificar la instalación de oc
oc version --client

# Obtener la IP de Windows desde WSL
ip route show | grep -i default | awk '{ print $3}'

# Agregar la IP de Windows al archivo hosts (reemplaza 172.23.208.1 con tu IP real)
echo "172.23.208.1 api.crc.testing" | sudo tee -a /etc/hosts

# Verificar que la resolución DNS funciona correctamente
getent hosts api.crc.testing

# Configurar un nuevo cluster apuntando al proxy de Windows
oc config set-cluster crc-wsl --server=http://172.23.208.1:8080 --insecure-skip-tls-verify=true

# Crear un nuevo contexto que use el cluster configurado
oc config set-context crc-wsl --cluster=crc-wsl --user=kubeadmin/api-crc-testing:6443 --namespace=default

# Cambiar al nuevo contexto
oc config use-context crc-wsl

# Verificar la configuración actual
oc config current-context

# Probar la conectividad básica
oc get pods

# Probar conectividad con todos los namespaces
oc get pods --all-namespaces

# Verificar los nodos del cluster
oc get nodes

# Limpiar archivos temporales
rm -f openshift-client-linux.tar.gz kubectl
```

YAML kubernetes/Openshift

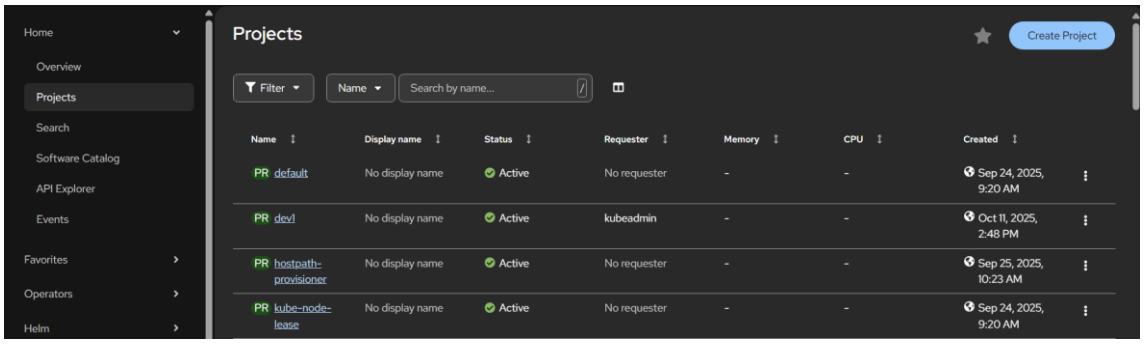
Sigue la misma estructura que kuberntes con los 4 objetos importantes: api, kind, metadata, spec.

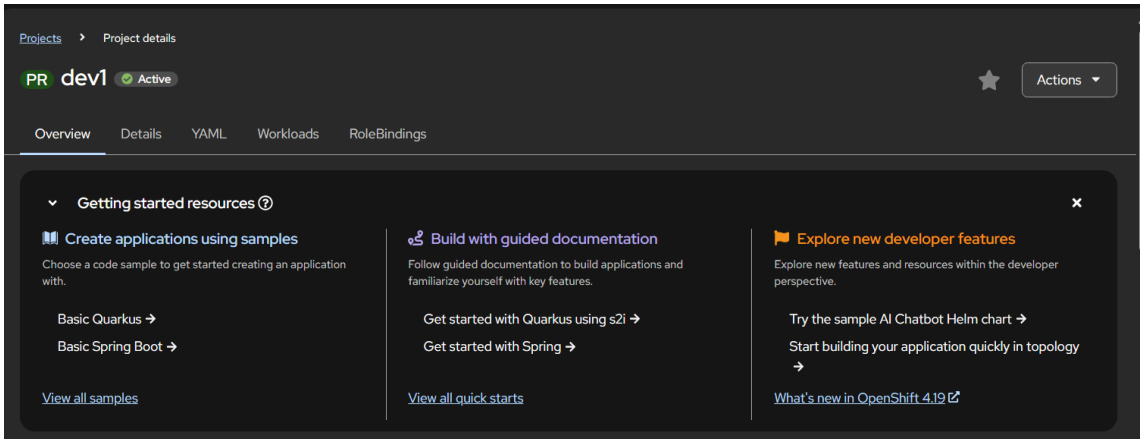
```
apiVersion: project.openshift.io/v1
kind: Project
metadata:
  annotations:
    openshift.io/description: Esto es la descripcion del proyecto
    openshift.io/display-name: Ejemplo de creacion de una proyecto Openshift
    openshift.io/requester: developer
    documentacion: Ejemplo para crear un proyecto en openshift
  name: desa2
  labels:
     tipo: desa
spec:
  finalizers:
  - kubernetes
```

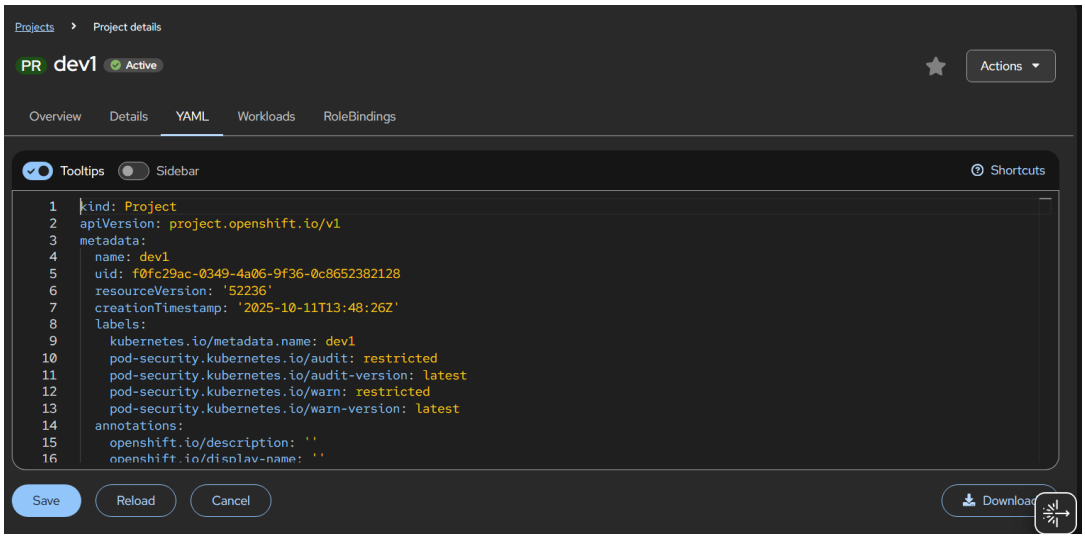
Obtener el yaml de un objeto

```
oc get project dev1 -o yaml
```

Ver proyecto desde la web





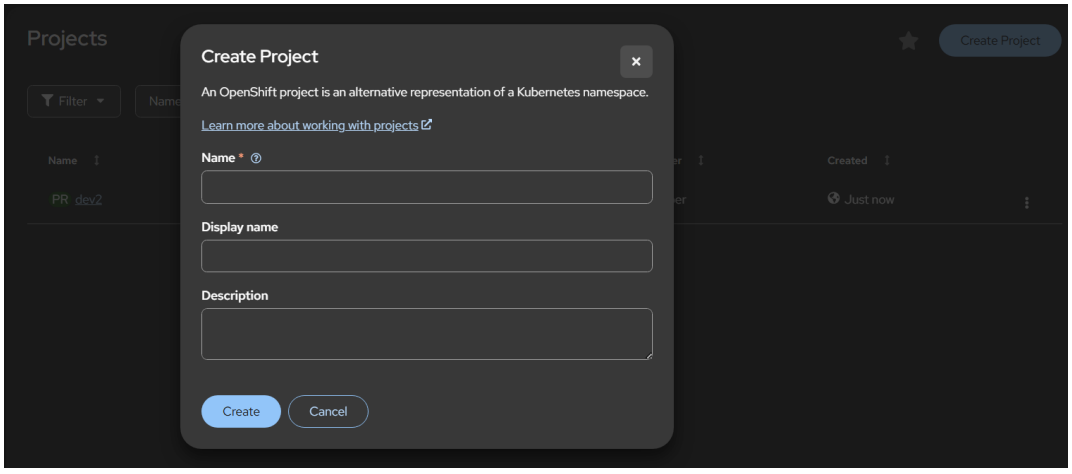


Crear proyecto con yaml

```yaml
apiVersion: project.openshift.io/v1
kind: Project
metadata:
  annotations:
    openshift.io/description: Esto es la descripcion del proyecto
    openshift.io/display-name: Ejemplo de creacion de una proyecto Openshift
    openshift.io/requester: developer
    documentacion: Ejemplo para crear un proyecto en openshift
  name: desa2
  labels:
     tipo: desa
spec:
  finalizers:
  - kubernetes
```

```
oc apply -f proyecto.yaml
```

Desde la web




```
PS C:\Users\g10n_> oc get project dev2
NAME   DISPLAY NAME   STATUS
dev2   Proyecto 2     Active
PS C:\Users\g10n_>
```

IMPORTANTE: si creo un namespace en openshift crea un project y visceversa

Seguridad en Openshift *

Configurar permisos para poder evitar errores de seguridad.

Algunas imágenes que usaremos durante el curso requiere privilegios de acceso como ROOT o bien necesitan ciertos permisos para acceder a volúmenes o puertos.

Por ejemplo postgres, redis, Apache, etc.

Aunque este curso no es de Administración, necesitamos dar ciertos permisos al usuario para que pueda trabajar.

Es necesario ejecutar el siguiente comando en cada uno de los proyectos que creemos durante el curso

```
oc adm policy add-scc-to-user anyuid -z default
```

De esa forma podremos crear objetos y contenedores sin problemas.

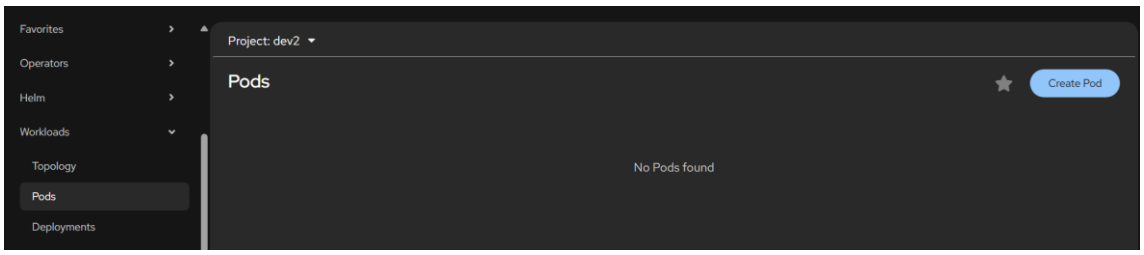
```
PS C:\Users\g10n_> oc adm policy add-scc-to-user anyuid -z default
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:anyuid added: "default"
```

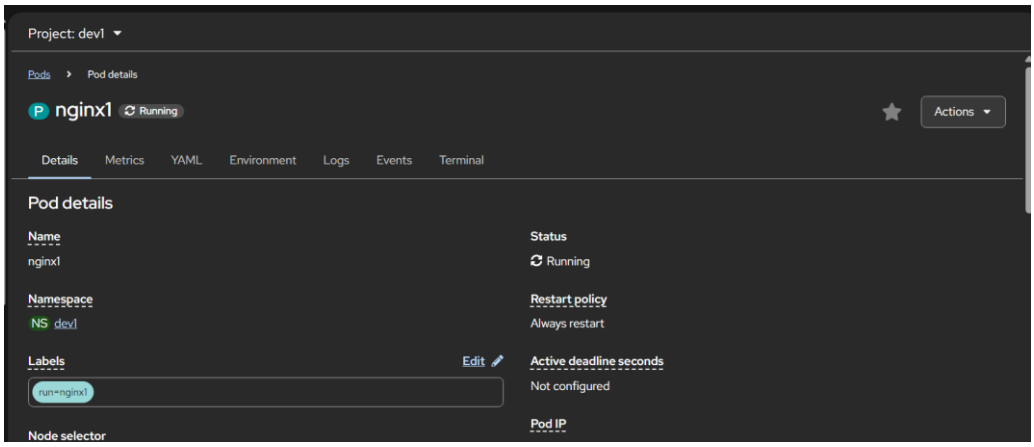
Crear pod

```
oc run nginx1 --image=nginx
oc get pod
oc describe pod nginx1
```

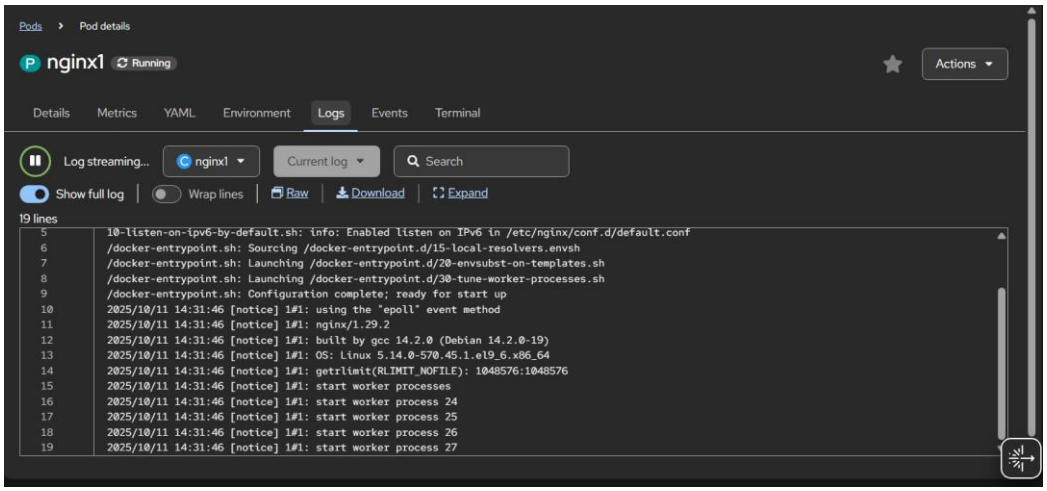
```
PS C:\Users\g10n_> oc run nginx1 --image=nginx
pod/nginx1 created
PS C:\Users\g10n_> oc get pod
NAME     READY   STATUS              RESTARTS   AGE
nginx1   0/1     ContainerCreating   0          4s
PS C:\Users\g10n_> oc describe pod nginx1
Name:             nginx1
Namespace:        dev1
Priority:         0
Service Account:  default
Node:             crc/192.168.126.11
Start Time:       Sat, 11 Oct 2025 15:31:40 +0100
Labels:           run=nginx1
Annotations:      k8s.ovn.org/pod-networks:
                    {"default":{"ip_addresses":["10.217.0.80/23"],"mac_address":"0a:58:0a:d9:00:50","gateway_
17.0.1"],"routes":[{"dest":"10.217.0.0...
                  k8s.v1.cni.cncf.io/network-status:
                    [{
                        "name": "ovn-kubernetes",
                        "interface": "eth0",
                        "ips": [
                            "10.217.0.80"
                        ],
```

En la web



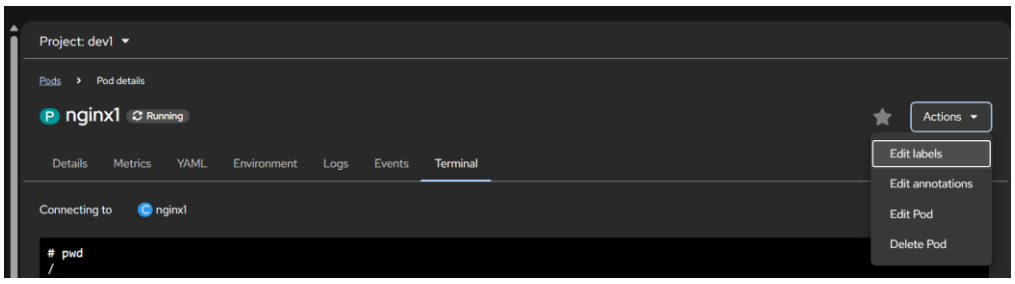


Logs



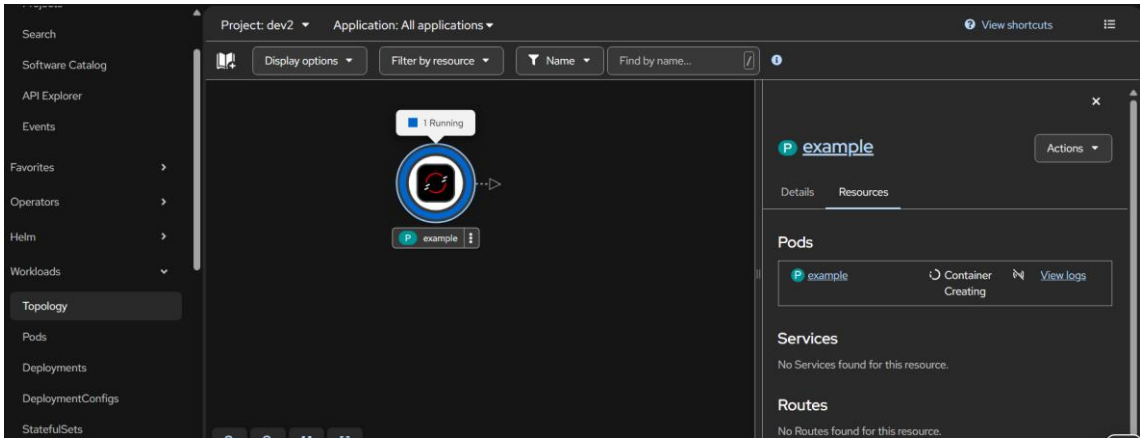
Se puede usar una terminal para ejecutar desde el pod

Editar algunas opciones del pod

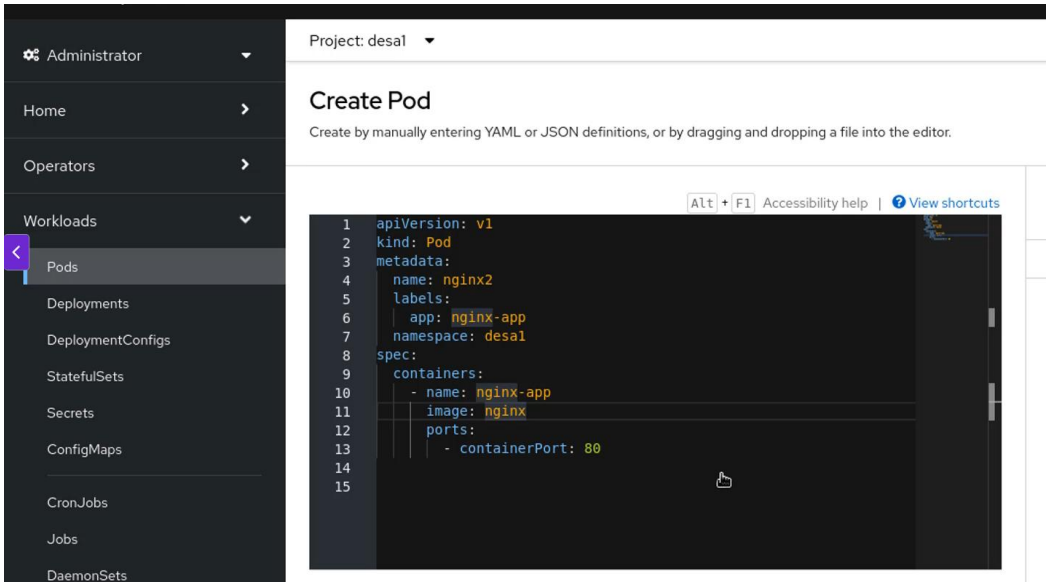


Pods en modo desarrollo

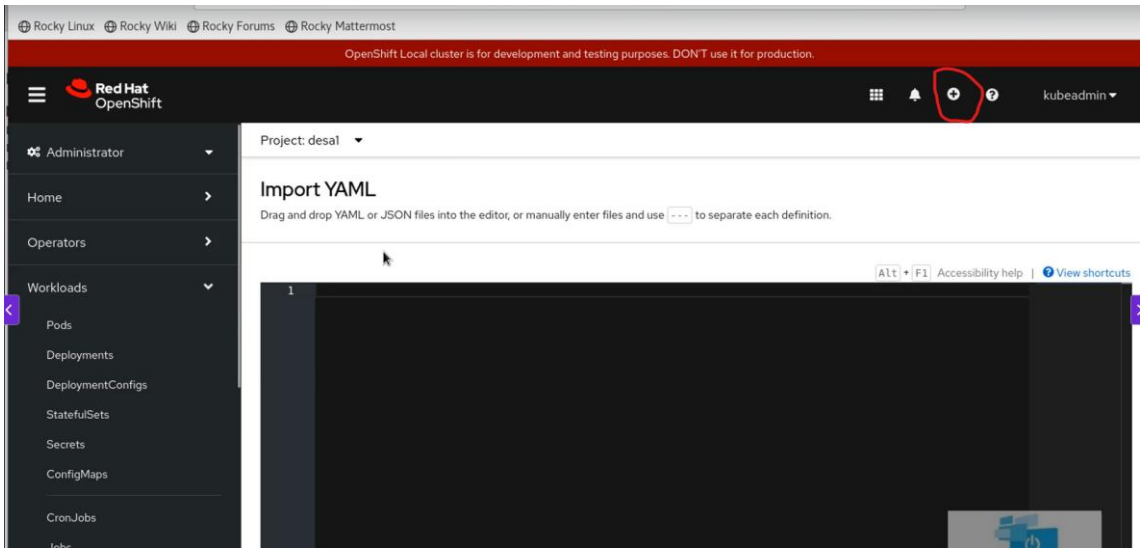
Muestra una seccion de topology, donde podemos ver el estado del pod



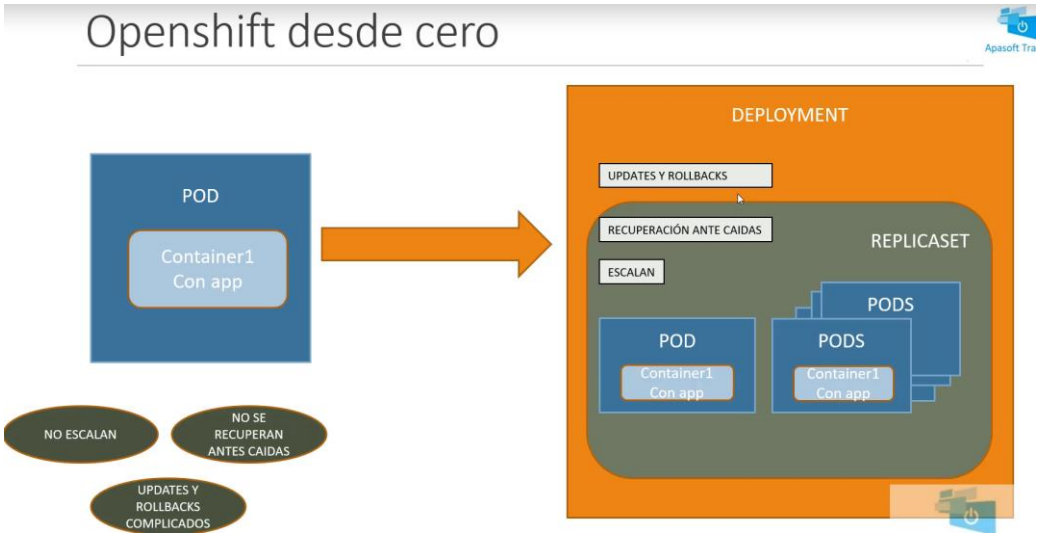
Pod en la web



Crear objetos de yaml desde la web

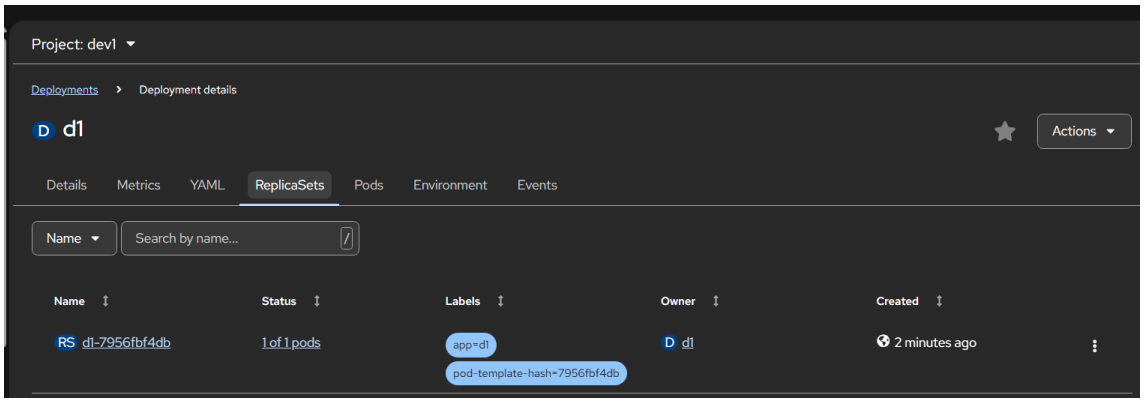


Deployment



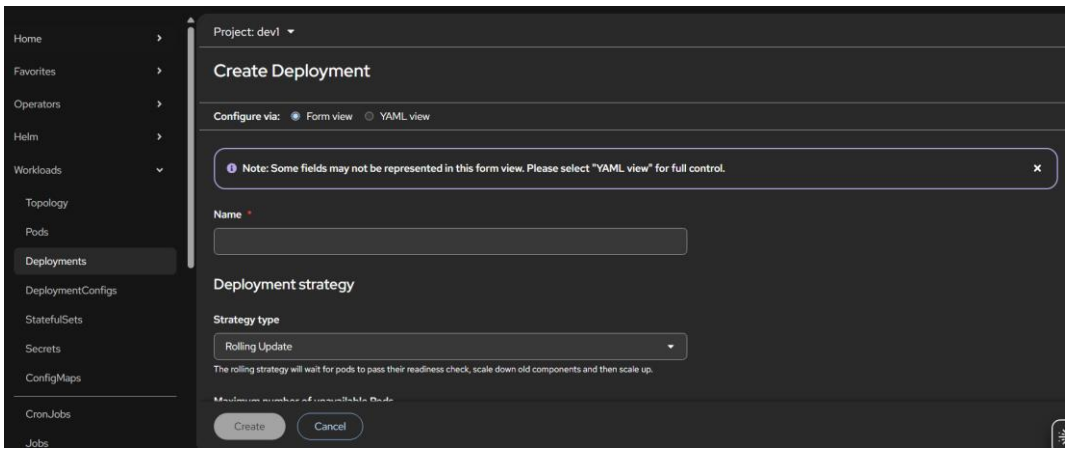

```
oc create deploy d1 --image=httpd
oc get deploy
oc get rs
oc get pod
```

```
PS C:\Users\g10n_> oc create deploy d1 --image=httpd
deployment.apps/d1 created
PS C:\Users\g10n_> oc get deploy
NAME   READY   UP-TO-DATE   AVAILABLE   AGE
d1     1/1     1            1           8s
PS C:\Users\g10n_> oc get svc
No resources found in dev1 namespace.
PS C:\Users\g10n_> oc get rs
NAME            DESIRED   CURRENT   READY   AGE
d1-7956fbf4db   1         1         1       29s
PS C:\Users\g10n_> oc get pod
NAME                  READY   STATUS    RESTARTS   AGE
d1-7956fbf4db-74kp5   1/1     Running   0          34s
nginx1                1/1     Running   0          86m
```

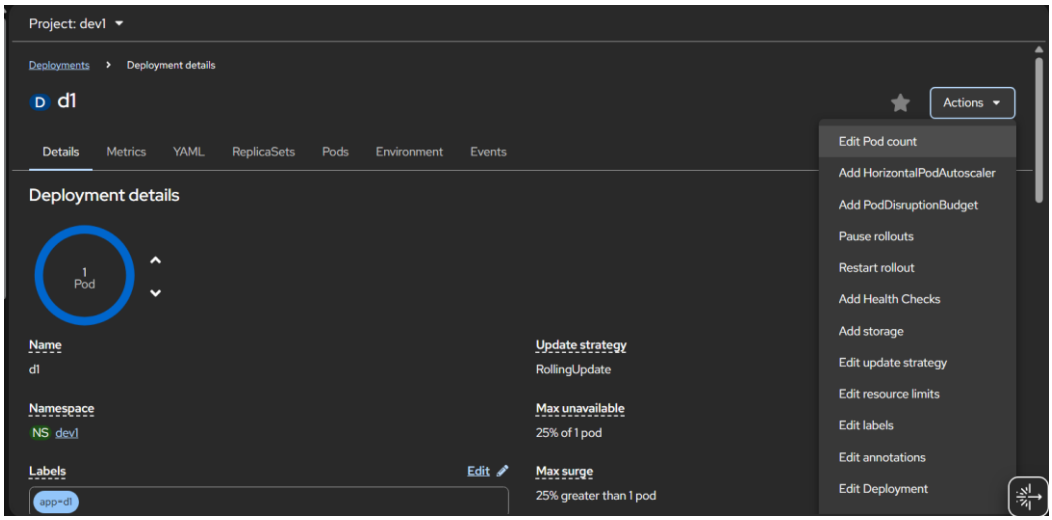


Crear deployment en la web

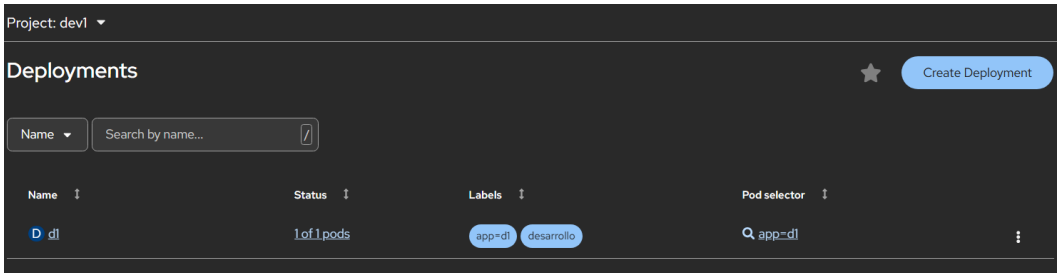
Se puede crear de 2 formas: del tipo formulario y del tipo yaml



Acciones de los deployments



Aplicar cambios en deployments




```
oc get deploy d1 -o wide --show-labels
oc label deploy d1 "responsable=pepe"
```

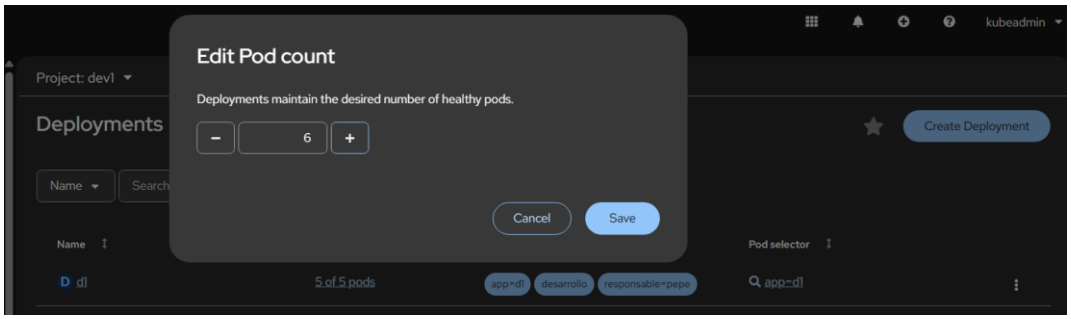
```
PS C:\Users\g10n_> oc get deploy d1 -o wide --show-labelds
error: unknown flag: --show-labelds
See 'oc get --help' for usage.
PS C:\Users\g10n_> oc get deploy d1 -o wide --show-labels
NAME   READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES   SELECTOR   LABELS
d1     1/1     1            1           3h41m   httpd        httpd    app=d1     app=d1,desarrollo=
PS C:\Users\g10n_> oc label deploy d1 "responsable=pepe"
deployment.apps/d1 labeled
PS C:\Users\g10n_> oc get deploy d1 -o wide --show-labels
NAME   READY   UP-TO-DATE   AVAILABLE   AGE     CONTAINERS   IMAGES   SELECTOR   LABELS
d1     1/1     1            1           3h42m   httpd        httpd    app=d1     app=d1,desarrollo=,responsable=pepe
```

Escalar deployment

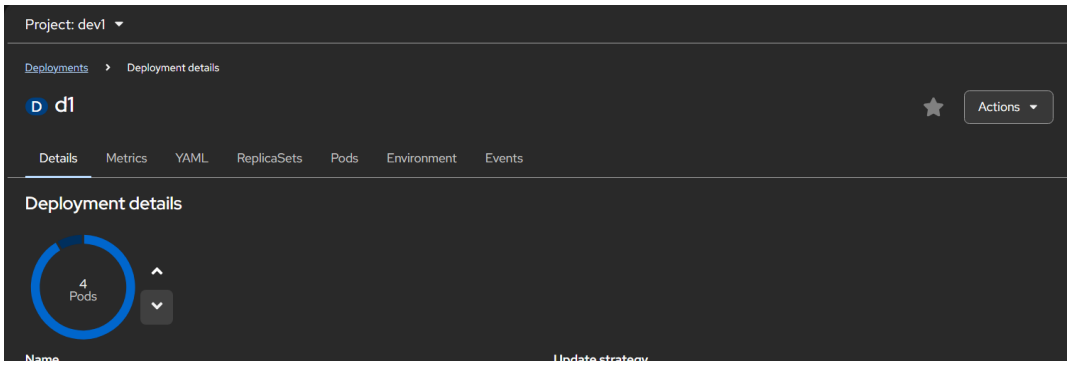
```
oc scale deploy d1 --replicas=5
```

```
PS C:\Users\g10n_> oc scale deploy d1 --replicas=5
deployment.apps/d1 scaled
PS C:\Users\g10n_> oc get pods
NAME                    READY   STATUS    RESTARTS   AGE
d1-7956fbf4db-5fln5     1/1     Running   0          12s
d1-7956fbf4db-74kp5     1/1     Running   0          3h44m
d1-7956fbf4db-hzmv5     1/1     Running   0          12s
d1-7956fbf4db-l5thg     1/1     Running   0          12s
d1-7956fbf4db-l6wtb     1/1     Running   0          12s
nginx1                  1/1     Running   0          5h10m
```

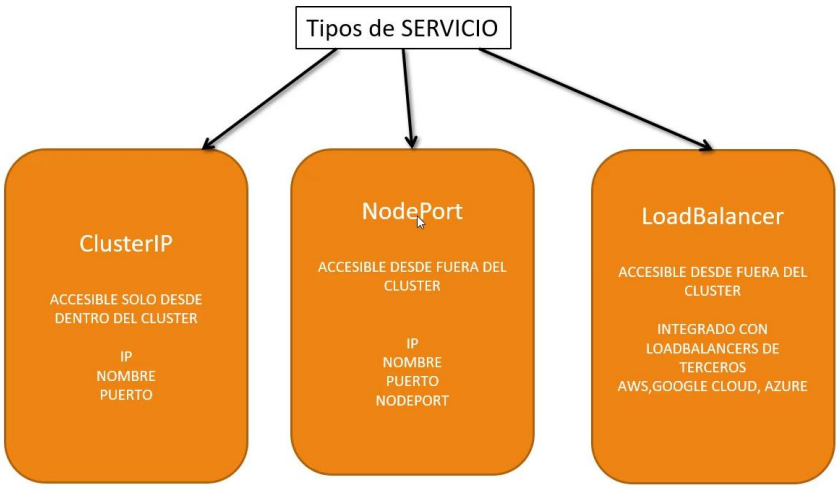
Aumentar desde la web



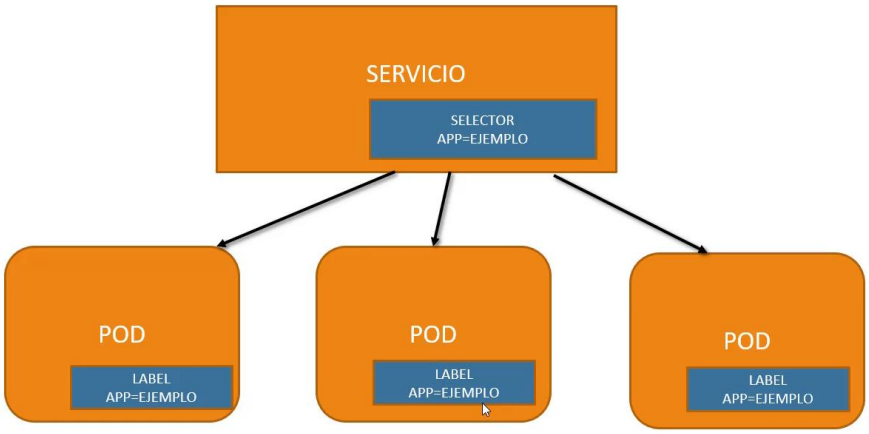
Aumentar o disminuir desde dentro del deployment, en el circulo



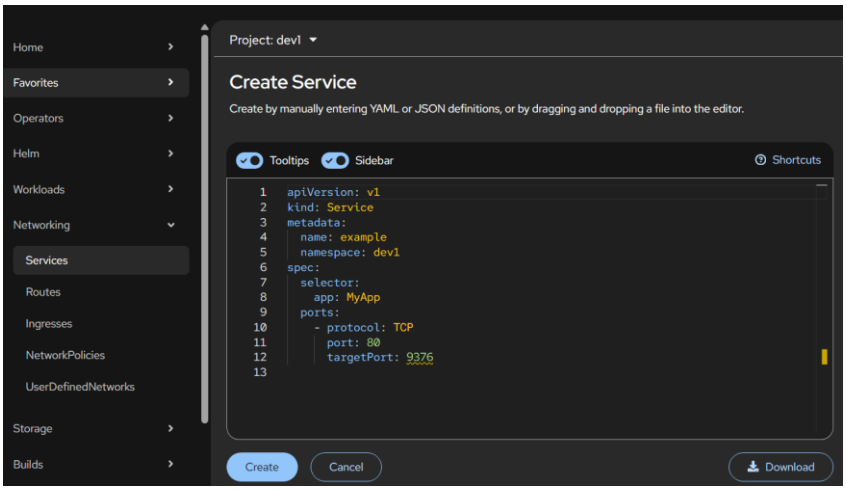
Servicios



El servicio apunta a los labels de los pods asociados




```
oc expose --name=apache1-svc deploy apache1 --type=NodePort
```




```
oc create deploy nginx-dep --image=nginx -n=dev1
oc get pods -o wide --show-labels
#label=nginx-dep
```

En la web crear el siguiente servicio en yaml:

```
apiVersion: v1
kind: Service
metadata:
 name: nginx-svc
 namespace: dev1
spec:
 selector:
  app: nginx-dep
 ports:
  - protocol: TCP
    port: 80
    targetPort: 80
    nodePort: 31800
 type: NodePort
```

Estaba apareciendo este error en el pod

0/1 nodes are available: 1 node(s) had untolerated taint {node.kubernetes.io/disk-pressure: }. preemption: 0/1 nodes are available: 1 Preemption is not helpful for scheduling.

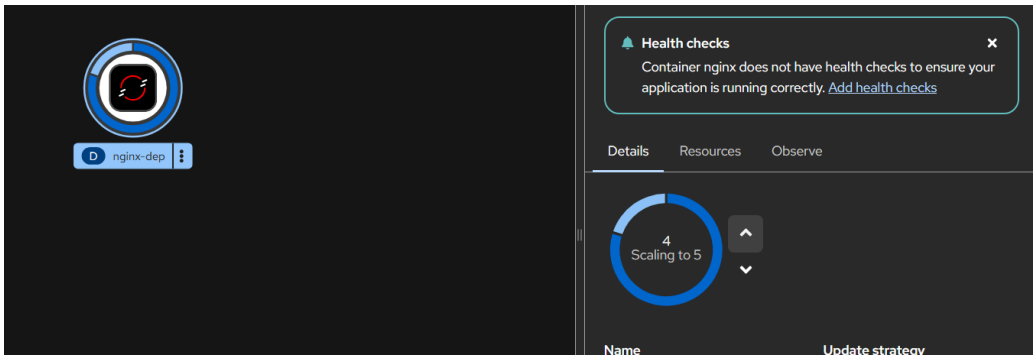
Ademas

```
PS C:\Users\g10n_\OneDrive\Escritorio\Curso Platzi\DevOps\11. Kubernetes\ku
ernetes> k get pods -n dev1
NAME                     READY   STATUS                   RESTARTS   AGE
nginx-dep-5879cc9-4nmdq  0/1     Completed                0          2m57s
nginx-dep-5879cc9-4tr6f  0/1     ContainerStatusUnknown   0          2m47s
nginx-dep-5879cc9-5jmwz  0/1     ContainerStatusUnknown   0          2m49s
nginx-dep-5879cc9-5zbsl  0/1     ContainerStatusUnknown   0          2m49s
nginx-dep-5879cc9-6dbhh  0/1     ContainerStatusUnknown   0          2m48s
nginx-dep-5879cc9-74xkw  0/1     ContainerStatusUnknown   0          2m47s
nginx-dep-5879cc9-7lsgn  0/1     ContainerStatusUnknown   0          2m49s
nginx-dep-5879cc9-c26t4  0/1     ContainerStatusUnknown   0          2m49s
```

Se soluciono este error eliminando los pods que ocupababan espacion innecesario en el cluster

oc delete pods -n dev1 --field-selector=status.phase!=Running

Creando mas pods desde la web




```
PS C:\Users\g10n_> oc get pods -n dev1
NAME                     READY   STATUS    RESTARTS   AGE
nginx-dep-5879cc9-4qkdd  1/1     Running   0          30s
nginx-dep-5879cc9-f72rv  1/1     Running   0          31s
nginx-dep-5879cc9-f9jtv  1/1     Running   0          31s
nginx-dep-5879cc9-wjwpx  1/1     Running   0          29s
nginx-dep-5879cc9-z6p2t  1/1     Running   0          14m
```

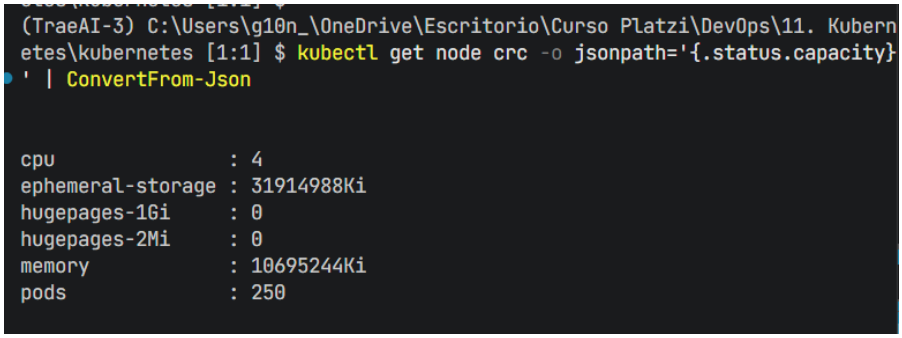
Podemos verificar que el servicio habilita correctamente los endpoints para cada pod

```
PS C:\Users\g10n_> oc describe svc nginx-svc -n dev1
Name:                     nginx-svc
Namespace:                dev1
Labels:                   <none>
Annotations:              <none>
Selector:                 app=nginx-dep
Type:                     NodePort
IP Family Policy:         SingleStack
IP Families:              IPv4
IP:                       10.217.5.38
IPs:                      10.217.5.38
Port:                     <unset>  80/TCP
TargetPort:               80/TCP
NodePort:                 <unset>  31800/TCP
Endpoints:                10.217.0.144:80,10.217.0.149:80,10.217.0.150:80 + 2 more...
Session Affinity:         None
External Traffic Policy:  Cluster
Internal Traffic Policy:  Cluster
Events:                   <none>
PS C:\Users\g10n_> oc get pods -o wide -n dev1
NAME                     READY   STATUS    RESTARTS   AGE     IP             NODE   NOMINATED NODE   READINESS GATES
nginx-dep-5879cc9-4qkdd  1/1     Running   0          2m14s   10.217.0.151   crc    <none>           <none>
nginx-dep-5879cc9-f72rv  1/1     Running   0          2m15s   10.217.0.150   crc    <none>           <none>
nginx-dep-5879cc9-f9jtv  1/1     Running   0          2m15s   10.217.0.149   crc    <none>           <none>
nginx-dep-5879cc9-wjwpx  1/1     Running   0          2m13s   10.217.0.152   crc    <none>           <none>
nginx-dep-5879cc9-z6p2t  1/1     Running   0          15m     10.217.0.144   crc    <none>           <none>
PS C:\Users\g10n_>
```

Aumentar los recursos en CRC local *

Verificar los recursos actuales

```
crc config view
```



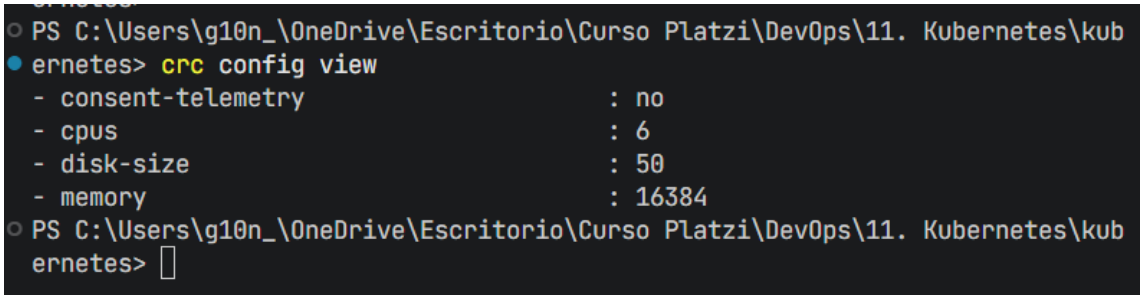
Capacidad de CRC actual:

- CPU : 4 cores
- Memoria : ~10.4GB (10,695,244 Ki)
- Almacenamiento : ~31GB (31,914,988 Ki)
- Pods máximos : 250

Script para aumentar capacidades

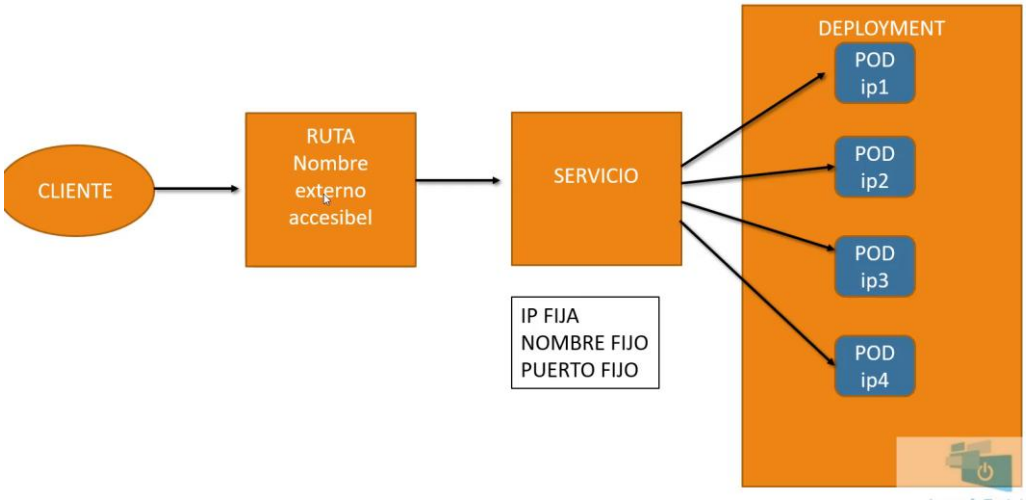
```
# 1. Detener CRC
crc stop

# 2. Configurar más recursos
crc config set memory 16384      # 16GB RAM
crc config set cpus 6            # 6 CPU cores
crc config set disk-size 50      # 50GB disco

# 3. Reiniciar CRC
crc start
```

Estado actual



Routes (propio de Openshift)

- **Rutas**
  - Una ruta es un objeto que permite exponer una aplicación web al tráfico externo en un clúster de OpenShift.
  - Crea una URL que redirige las solicitudes de los clientes al servicio subyacente y por tanto al deploy o al deploymentConfig de la aplicación.
  - Es por tanto, un punto de entrada para nuestras aplicaciones
  - Son muy parecidas a los Ingress controller de Kubernetes.
  - De hecho, podemos seguir usando Ingress en un cluster de OpenShift



Crear ruta usando la linea de comandos

```
oc expose svc web1-svc -n dev1
```

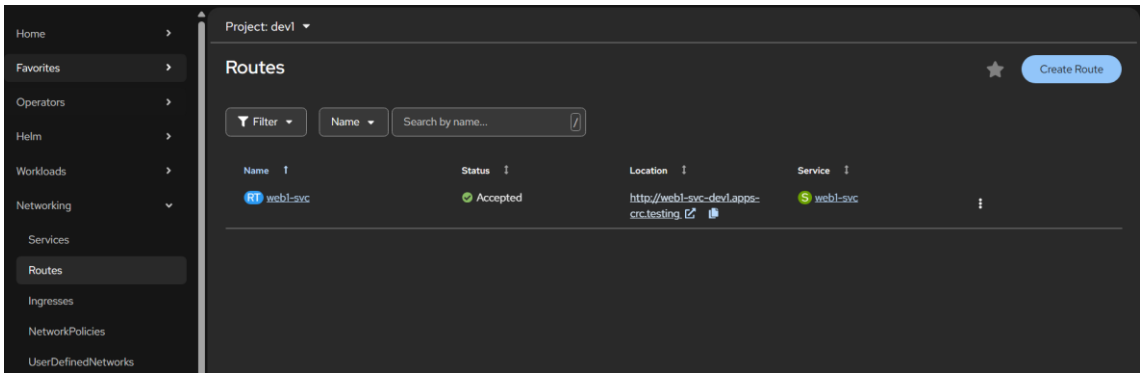
```
PS C:\Users\g10n_> oc get svc -n dev1
NAME        TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
nginx-svc   NodePort   10.217.5.38    <none>        80:31800/TCP   47m
web1-svc    NodePort   10.217.5.129   <none>        80:30002/TCP   27m
PS C:\Users\g10n_> oc expose svc web1-svc -n dev1
route.route.openshift.io/web1-svc exposed
PS C:\Users\g10n_> oc get route -n dev1
NAME       HOST/PORT                          PATH   SERVICES   PORT   TERMINATION   WILDCARD
web1-svc   web1-svc-dev1.apps-crc.testing            web1-svc   80                   None
PS C:\Users\g10n_>
```

```
web1-svc-dev1.apps-crc.testing
```

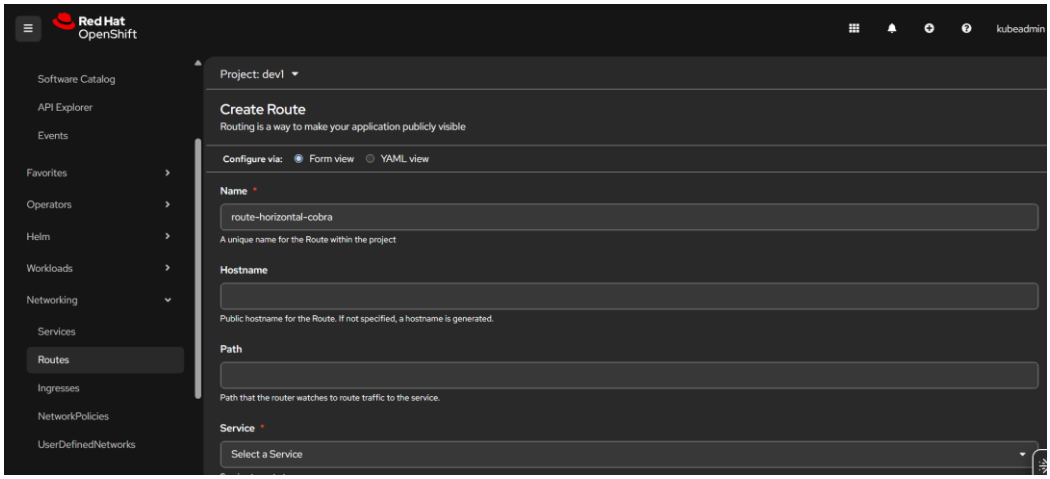
Probarlo en la web como (sin considerar la S en http)

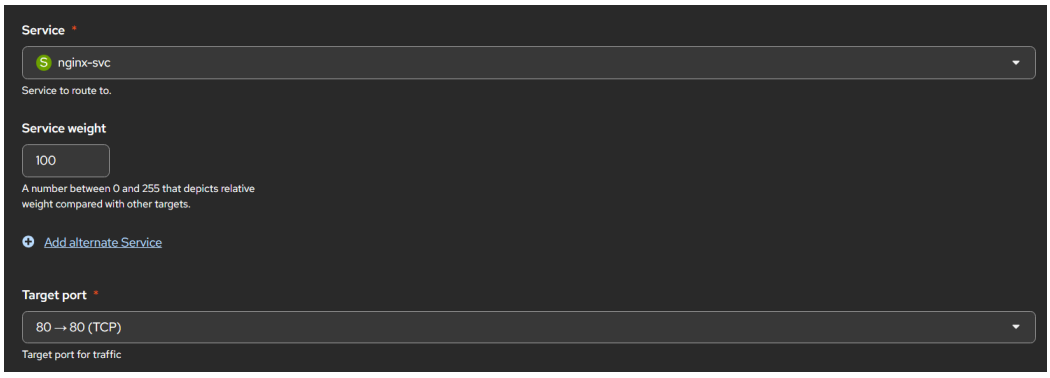
```
http://web1-svc-dev1.apps-crc.testing/
```

Tambien podemos verlo desde el admin de Openshift

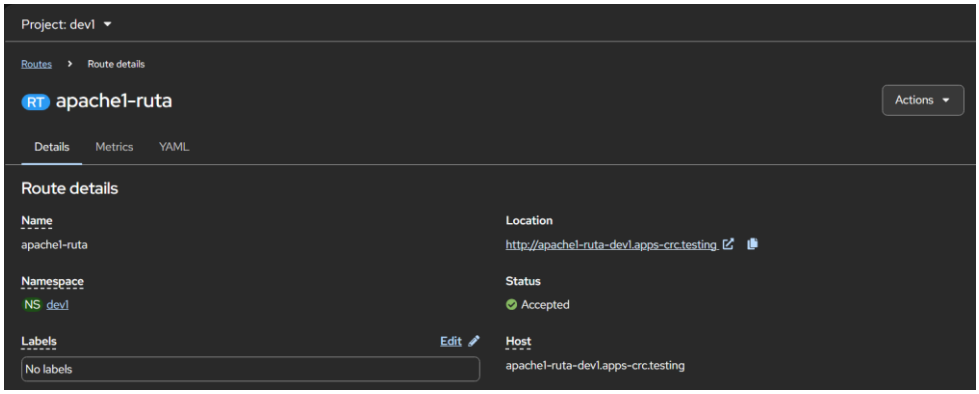


Crear ruta desde el formulario





Ruta creada



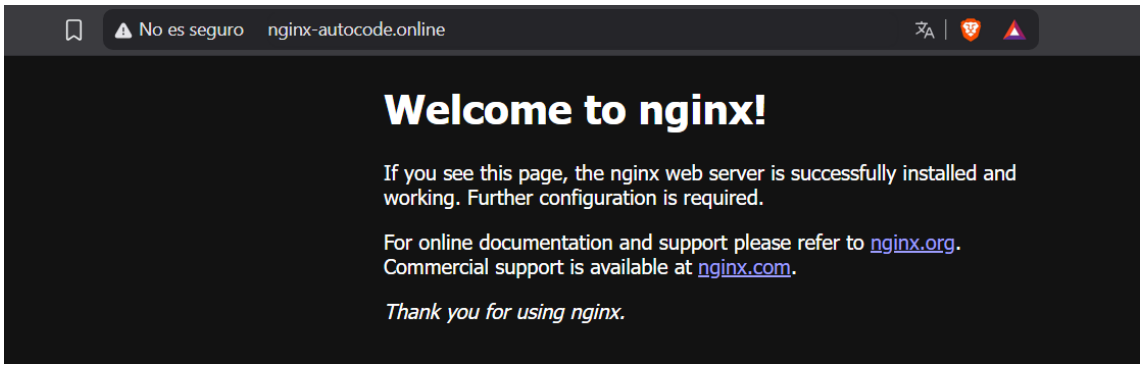
Crear ruta con dominio personalizado (solo de prueba)

Agregar la ruta en el archivo hosts de Windows

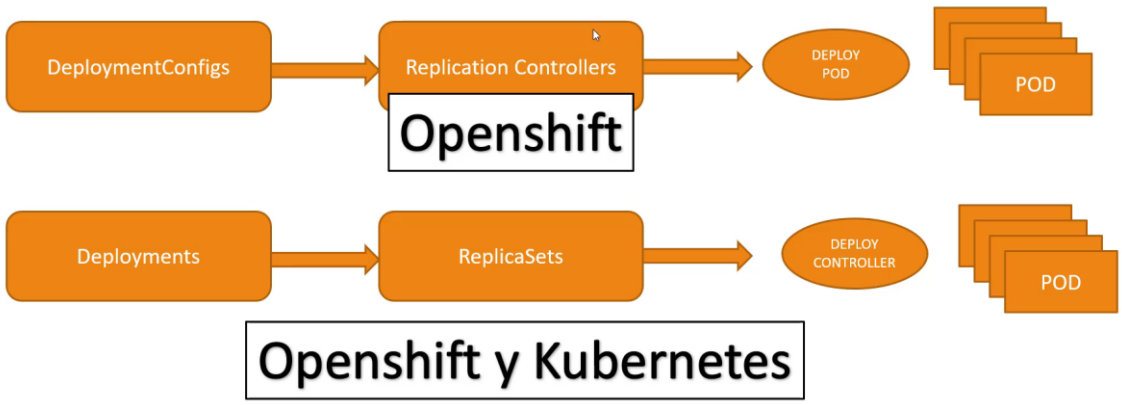
C:\Windows\System32\drivers\etc\hosts

127.0.0.1	nginx-autocode.online

Probar nuevamente la URL personalizada



DeploymentConfigs (Deprecados)



- Características de un DeploymentConfig
  - Un DeploymentConfig, es en realidad una plantilla para lanzar aplicaciones.
  - Dispone de Triggers que disparan depliegues automáticos ante determinados eventos.
  - Existen estrategias configurables para determiner la transición desde la version anterior a la actual.
  - Hay una serie de hooks (lifecycle hooks) que permiten configurar un comportamiento en diferentes momentos del ciclo de vida del despliegue.
  - Versionado de la aplicación, para poder realizar rollbacks tanto de forma manual como automática.
  - Se soporta escalado manual y automático de las réplicas.

- Diferencias con los Deployment de Kubernetes
  - Los DeploymentConfigs se basan en la consistencia mientras que los Deployments prefienren la disponbilidad
  - Automatic rollbacks: los Deployments no tienen rolling back automático al último ReplicaSet desplegado en caso de fallo.
  - Lifecycle hooks: Los Deployments NO soportan lifecycle hooks.
  - Custom strategies: Los Deployments no disponen de estrategias personalizadas de despliegue.

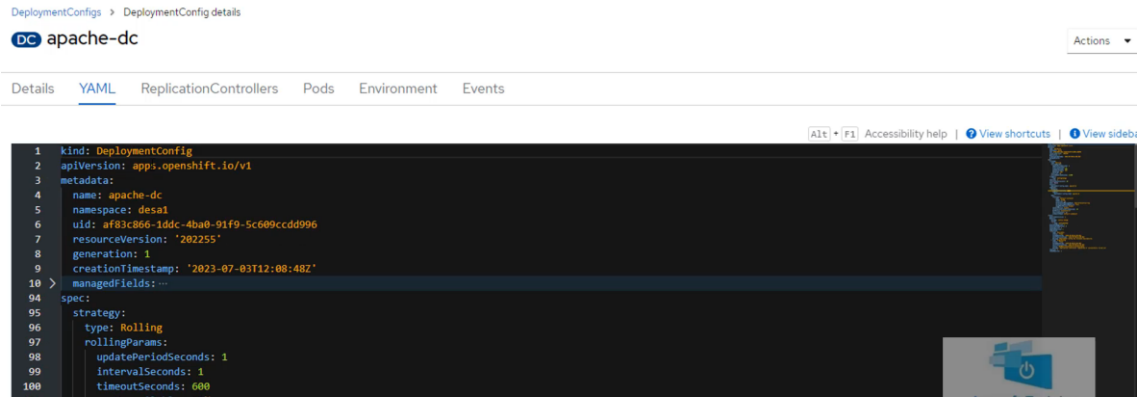
- Diferencias con los Deployment de Kubernetes
  - ROLLOVER
    - Los Deployment pueden tener tantos ReplicaSets activos como sea necesarios, de forma que el deployment controller gestionara la bajada de los antiguos y la subida de los nuevos.
    - Los DeploymentConfigs solo pueden tener un deployer pod funcionando. Por tanto, solo 2 ReplicationControllers pueden estar activos al mismo tiempo.

- Diferencias con los Deployment de Kubernetes
  - ESCALADO PROPORCIONAL
    - Dado que el Deployment controller es el que gestiona los tamaños de los nuevos y de los viejos ReplicaSets sepueden escalar los rollouts que en ese momento estén en marcha..
    - DeploymentConfigs no pueden ser escalados mientras estamos en un rollout porque el DeploymentConfig controller no es capaz de gestionarlo.

Comandos

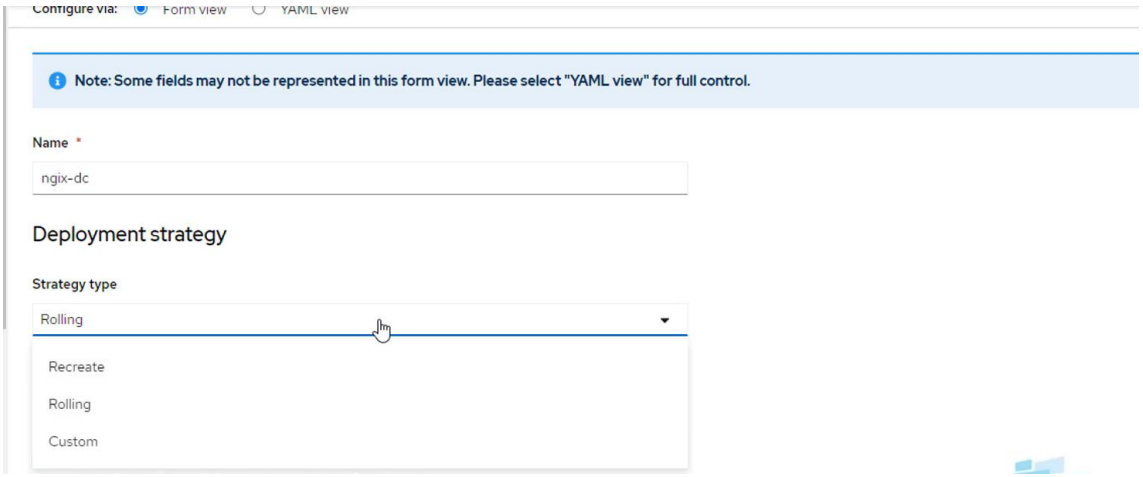
create deploymentconfig apache-dc --image=httpd oc get rc # Replication Controller oc get pod # Crea un pod adicional

```
openshift-->oc create deploymentconfig apache-dc --image=httpd
deploymentconfig.apps.openshift.io/apache-dc created
openshift-->oc get dc
NAME        REVISION   DESIRED   CURRENT   TRIGGERED BY
apache-dc   1          1         1         config
openshift-->oc get rc
NAME          DESIRED   CURRENT   READY   AGE
apache-dc-1   1         1         1       41s
openshift-->oc get pod
NAME                          READY   STATUS      RESTARTS   AGE
apache-dc-1-deploy            0/1     Completed   0          73s
apache-dc-1-zkfvw             1/1     Running     0          66s
apache1-89d978569-fkss2       1/1     Running     2          21h
apache1-89d978569-vftl7       1/1     Running     2          21h
d1-67d4b54b64-gdnmb           1/1     Running     2          23h
d2-9f46d8bf9-vc767            1/1     Running     2          22h
nginx-dep-7ccf5fddd7-lbn7p    1/1     Running     2          17h
nginx-dep-7ccf5fddd7-xkrqx    1/1     Running     2          17h
web1-cf7cc764f-lglpw          1/1     Running     2          16h
web1-cf7cc764f-zwgdt          1/1     Running     2          16h
openshift-->
```



Desde la web

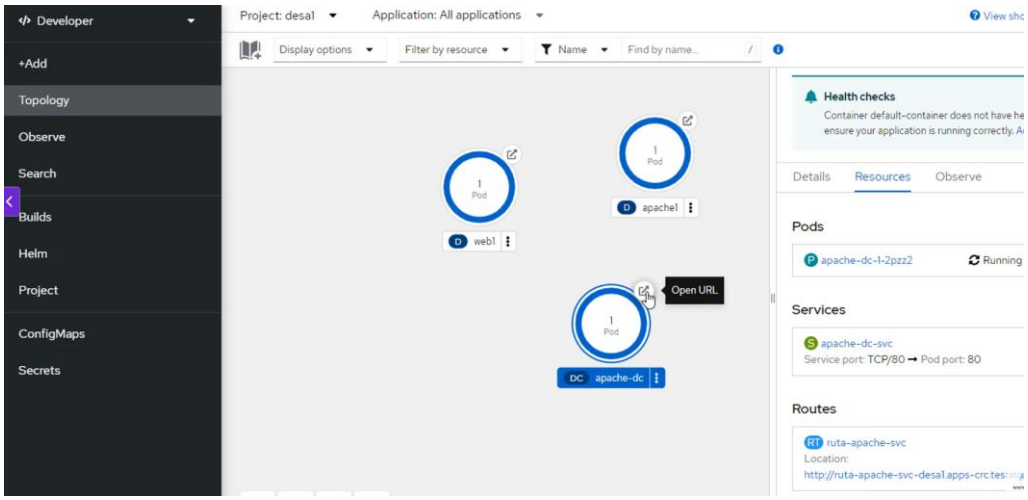
Tiene una opcion mas de estrategia de deployment, que es la de custom



Crear un servicio y ruta al DC

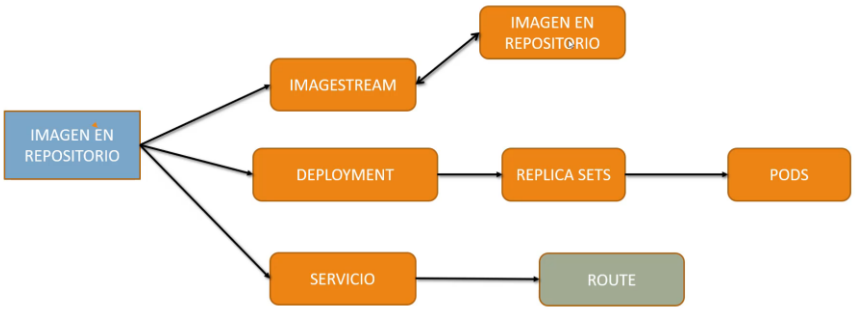
```
openshift-->oc get dc
NAME        REVISION   DESIRED   CURRENT   TRIGGERED BY
apache-dc   1          1         1         config
ngix-dc     1          2         2         config
openshift-->oc expose --name=apache-dc-svc deploymentconfig apache-dc --port=80
service/apache-dc-svc exposed
openshift-->oc get svc
NAME           TYPE        CLUSTER-IP      EXTERNAL-IP   PORT(S)        AGE
apache-dc-svc  ClusterIP   10.217.5.32     <none>        80/TCP         7s
apache1-svc    NodePort    10.217.5.53     <none>        80:31767/TCP   26h
ngix-svc       NodePort    10.217.4.253    <none>        80:31800/TCP   25h
web1-svc       NodePort    10.217.5.105    <none>        80:30002/TCP   24h
openshift-->oc expose svc apache-dc-svc --name ruta-apache-svc
route.route.openshift.io/ruta-apache-svc exposed
openshift-->oc get route
NAME            HOST/PORT                                  PATH   SERVICES       PORT   TERMINATION   WILDCARD
apache-ruta     www.apasoft-prueba.com                            apache1-svc    80                   None
ruta-apache-svc ruta-apache-svc-desa1.apps-crc.testing            apache-dc-svc  80                   None
web1-svc        web1-svc-desa1.apps-crc.testing                   web1-svc       80                   None
openshift-->
```

Los pods que tienen rutas configuradas tienen un icono de “Open Url”



Despliegue de aplicaciones

- Las aplicaciones se pueden desplegar en OpenShift de varias maneras posibles, tanto desde la consola web como del cliente oc.
- Los principales métodos para implementar una aplicación son:
  - Desplegar una aplicación desde una imagen existente.
  - Construir y desplegar una aplicación desde código fuente contenido en un repositorio GIT y usando Source-to-Image builder (S2I)
  - Construir y desplegar una aplicación desde código fuente contenido en un repositorio GIT y usando un Dockerfile.
  - Desde plantillas
- OBJETOS GENERADOS EN UN DEPLOY DE IMAGEN



Crear varios recursos en el cluster con Openshift

oc new-app apasoft/blog

```
PS C:\Users\g10n_> oc new-app apasoft/blog
--> Found container image 5ada8e2 (5 years old) from Docker Hub for "apasoft/blog"

    Python 3.5
    ----------
    Python 3.5 available as container is a base platform for building and running various Python 3.5 applications and fr
ameworks. Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a s
imple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with i
ts interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most p
latforms.

    Tags: builder, python, python35, python-35, rh-python35

    * An image stream tag will be created as "blog:latest" that will track this image

--> Creating resources ...
    imagestream.image.openshift.io "blog" created
    deployment.apps "blog" created
    service "blog" created
--> Success
    Application is not exposed. You can expose services to the outside world by executing one or more of the commands be
low:
     'oc expose service/blog'
    Run 'oc status' to view your app.
PS C:\Users\g10n_>
```

oc expose service/blog

Consultar ImageStream

oc get is

```
PS C:\Users\g10n_> oc get is
NAME   IMAGE REPOSITORY                                                TAGS     UPDATED
blog   default-route-openshift-image-registry.apps-crc.testing/dev1/blog   latest   About a minute ago
PS C:\Users\g10n_>
```

Describir objeto

oc describe is blog

```
PS C:\Users\g10n_> oc describe is blog
Name:                   blog
Namespace:              dev1
Created:                2 minutes ago
Labels:                 app=blog
                        app.kubernetes.io/component=blog
                        app.kubernetes.io/instance=blog
Annotations:            openshift.io/generated-by=OpenShiftNewApp
                        openshift.io/image.dockerRepositoryCheck=2025-10-14T18:09:03Z
Image Repository:       default-route-openshift-image-registry.apps-crc.testing/dev1/blog
Image Lookup:           local=false
Unique Images:          1
Tags:                   1
```

Ver todas las imágenes del cluster

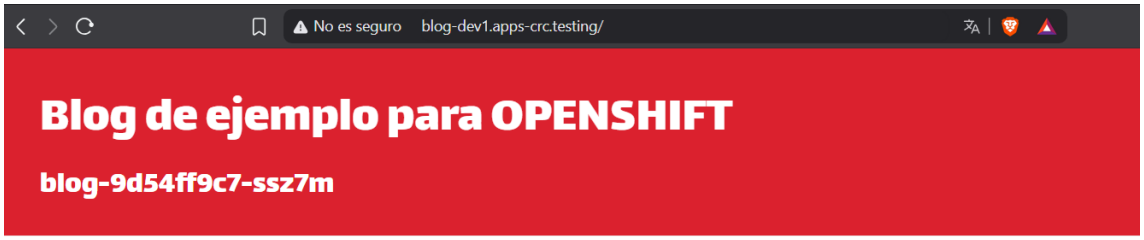
oc get images

Filtrar imagen

oc get images | Select-String "blog" #windows
#oc get images | grep blog # Linux

```
PS C:\Users\g10n_> oc get images | Select-String "blog" #windows

sha256:9c8d3cedcd722ae6a9508ca532102501dfb796600c53a61d5178cdfaf22e02ed
apasoft/blog@sha256:9c8d3cedcd722ae6a9508ca532102501dfb796600c53a61d5178cdfaf22e02ed
```

```
PS C:\Users\g10n_> oc get deploy
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
blog        1/1     1            1           7m46s
nginx-dep   1/1     1            1           21h
web1        2/2     2            2           21h
PS C:\Users\g10n_> oc get svc
NAME        TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
blog        ClusterIP   10.217.4.251   <none>        8080/TCP       7m56s
nginx-svc   NodePort    10.217.5.38    <none>        80:31800/TCP   21h
web1-svc    NodePort    10.217.5.129   <none>        80:30002/TCP   21h
PS C:\Users\g10n_> oc get route
NAME         HOST/PORT                          PATH   SERVICES    PORT       TERMINATION   WILDCARD
blog         blog-dev1.apps-crc.testing                blog        8080-tcp                 None
nginx-ruta   nginx-autocode.online                     nginx-svc   80                       None
web1-svc     web1-svc-dev1.apps-crc.testing            web1-svc    80                       None
PS C:\Users\g10n_> oc get pod
NAME                     READY   STATUS    RESTARTS   AGE
blog-9d54ff9c7-ssz7m     1/1     Running   0          8m22s
nginx-dep-5879cc9-z6p2t  1/1     Running   2          21h
web1-fd4f6ff7-fddzc      1/1     Running   2          21h
web1-fd4f6ff7-hw4vb      1/1     Running   2          21h
PS C:\Users\g10n_>
```

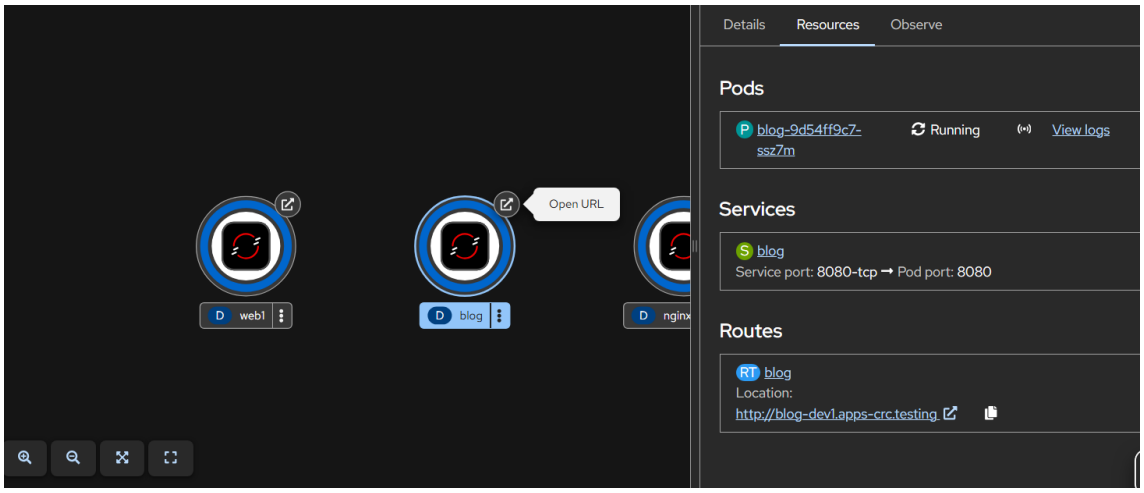


March 14, 2017, 3:26 a.m.

What is OpenShift Origin?

Origin is the upstream community project that powers OpenShift. Built around a core of Docker container packaging and Kubernetes container cluster management, Origin is also augmented by application lifecycle management functionality and DevOps tooling. Origin provides a complete open source container application platform.

En la web Topology



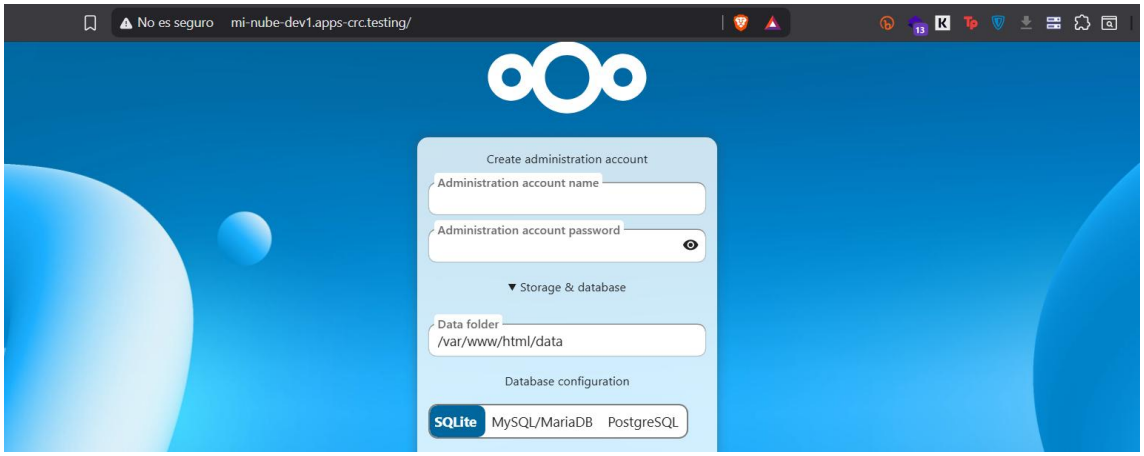
Nueva aplicación con DeploymentConfig

```
oc new-app --name=nginx10 --as-deployment-config=true --image=nginx
oc get dc
oc get rc
oc get pod
```

Desplegar la imagen nextcloud

```
oc new-app --name=mi-nube nextcloud
```

```
PS C:\Users\g10n_> oc new-app --name=mi-nube nextcloud
--> Found container image b97ea40 (2 weeks old) from Docker Hub for "nextcloud"

    * An image stream tag will be created as "mi-nube:latest" that will track this image

--> Creating resources ...
    imagestream.image.openshift.io "mi-nube" created
    deployment.apps "mi-nube" created
    service "mi-nube" created
--> Success
    Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:
     'oc expose service/mi-nube'
    Run 'oc status' to view your app.
PS C:\Users\g10n_> oc expose service/mi-nube
route.route.openshift.io/mi-nube exposed
PS C:\Users\g10n_> oc get is mi-nube
NAME      IMAGE REPOSITORY                                                      TAGS     UPDATED
mi-nube   default-route-openshift-image-registry.apps-crc.testing/dev1/mi-nube   latest   30 seconds ago
PS C:\Users\g10n_> oc get svc mi-nube
NAME      TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
mi-nube   ClusterIP   10.217.4.175   <none>        80/TCP    62s
PS C:\Users\g10n_> oc get route mi-nube
NAME      HOST/PORT                          PATH   SERVICES   PORT     TERMINATION   WILDCARD
mi-nube   mi-nube-dev1.apps-crc.testing             mi-nube    80-tcp                 None
PS C:\Users\g10n_> oc get all -l app=mi-nube
Warning: apps.openshift.io/v1 DeploymentConfig is deprecated in v4.14+, unavailable in v4.10000+
NAME             TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)   AGE
service/mi-nube  ClusterIP   10.217.4.175   <none>        80/TCP    93s

NAME                     READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/mi-nube  1/1     1            1           93s

NAME                                    IMAGE REPOSITORY                                                      TAGS     UPDATED
imagestream.image.openshift.io/mi-nube  default-route-openshift-image-registry.apps-crc.testing/dev1/mi-nube   latest   About a minute ago

NAME                                HOST/PORT                          PATH   SERVICES   PORT     TERMINATION   WILDCARD
route.route.openshift.io/mi-nube    mi-nube-dev1.apps-crc.testing             mi-nube    80-tcp                 None
PS C:\Users\g10n_>
```



Comprobar objetos generados sin lanzar aplicación

```
oc new-app --name=ejemplo --image=tomcat -o yaml > tomcat.yaml
```

Se puede entrar al yaml para modificarlo

```
g10n92@Giancarlo:~$ oc new-app --name=ejemplo --image=tomcat -o yaml > tomcat.yaml
g10n92@Giancarlo:~$ oc get deploy
NAME      READY   UP-TO-DATE   AVAILABLE   AGE
mi-nube   1/1     1            1           3d22h
g10n92@Giancarlo:~$ cat tomcat.yaml
apiVersion: v1
items:
- apiVersion: image.openshift.io/v1
  kind: ImageStream
  metadata:
    annotations:
      openshift.io/generated-by: OpenShiftNewApp
    creationTimestamp: null
    labels:
      app: ejemplo
```

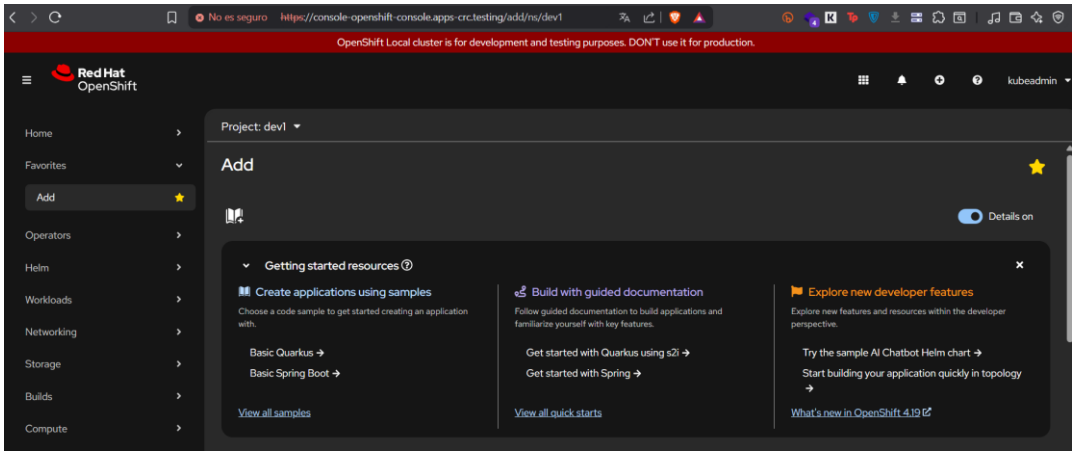
Aplicar cambios

```
oc apply -f tomcat.yaml
```

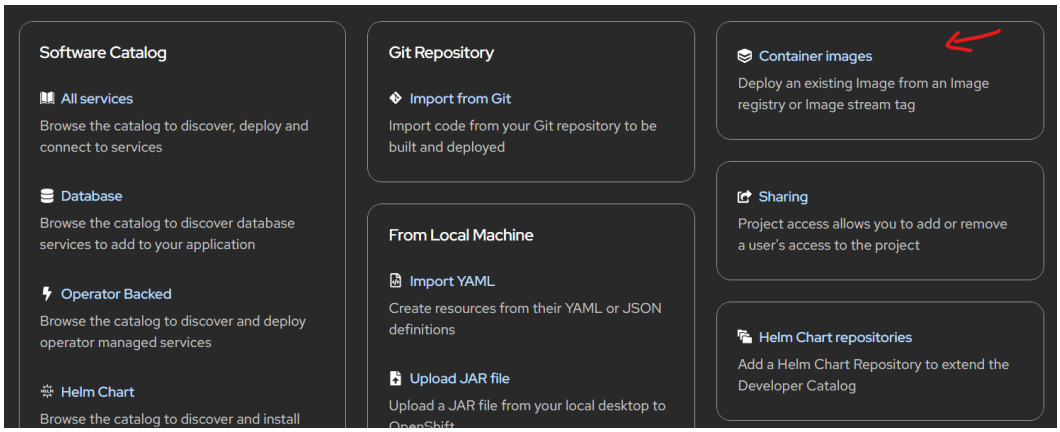
```
metadata: {}
g10n92@Giancarlo:~$ oc apply -f tomcat.yaml
imagestream.image.openshift.io/ejemplo created
deployment.apps/ejemplo created
service/ejemplo created
```

Desplegar proyectos desde la web

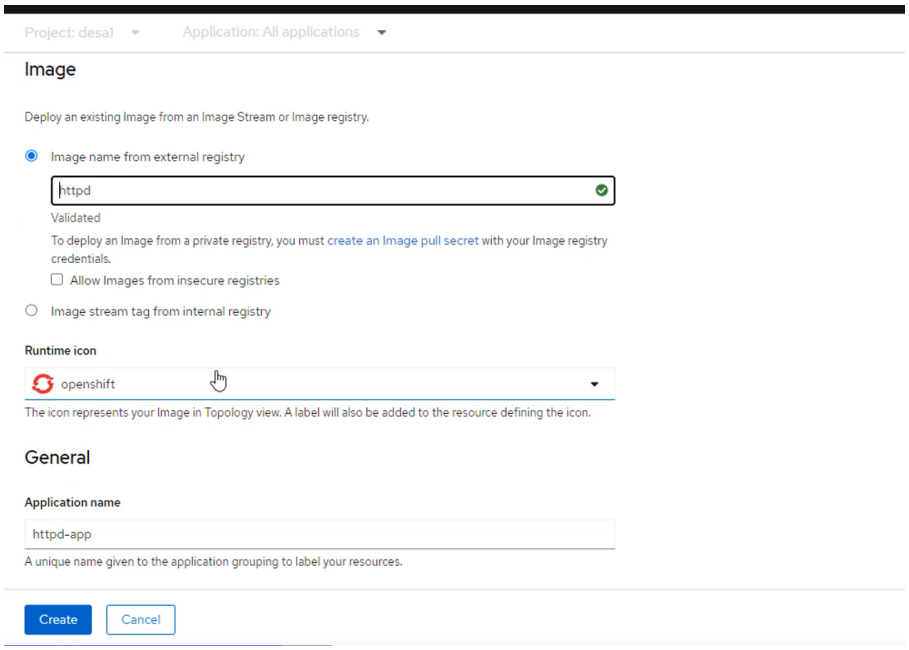
Entrar a la web desde la URL: add/ns/dev1[project], y agregarlo como favorito, porque no se encuentra facilmente



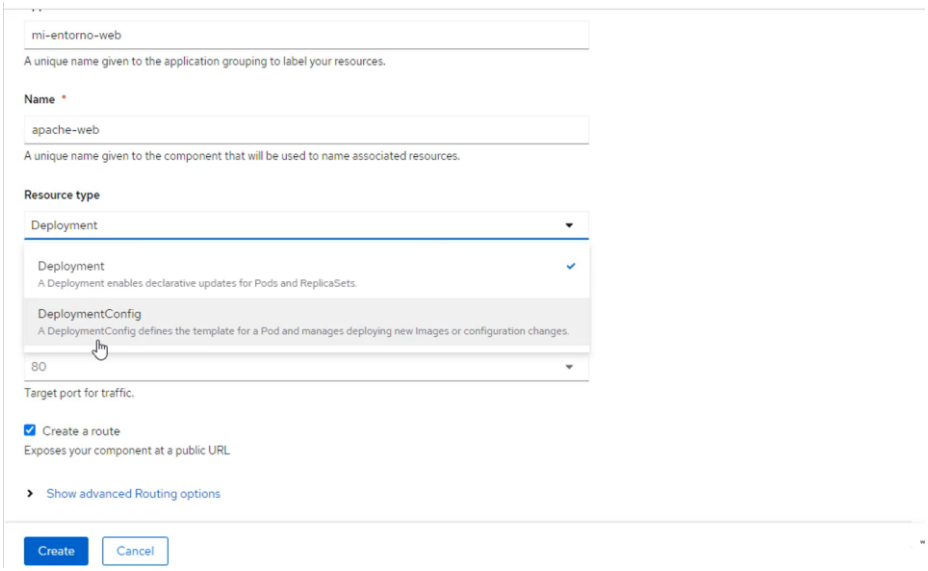
Luego vamos a elegir “Container images”



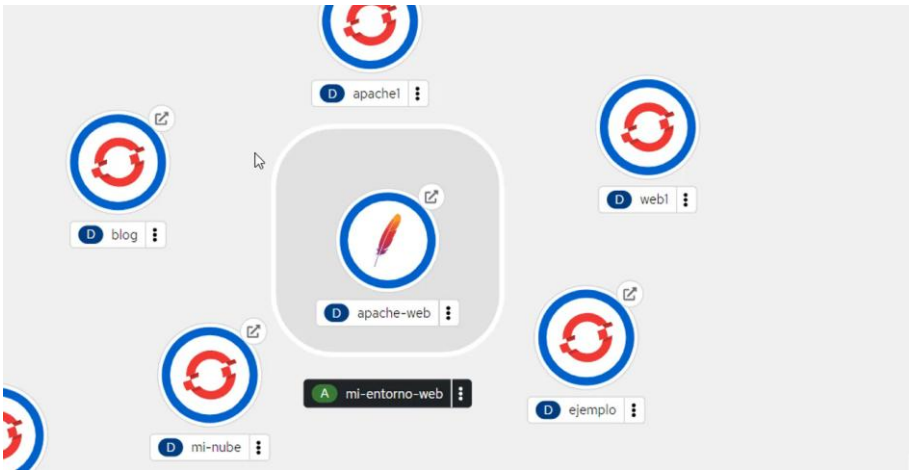
Poner el nombre de la imagen, por defecto busca en docker hub, se puede elegir icono



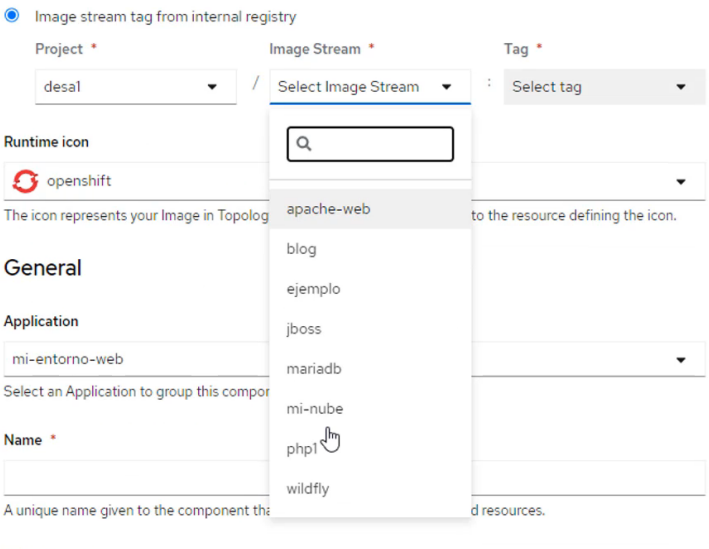
Se puede elegir otras configuraciones



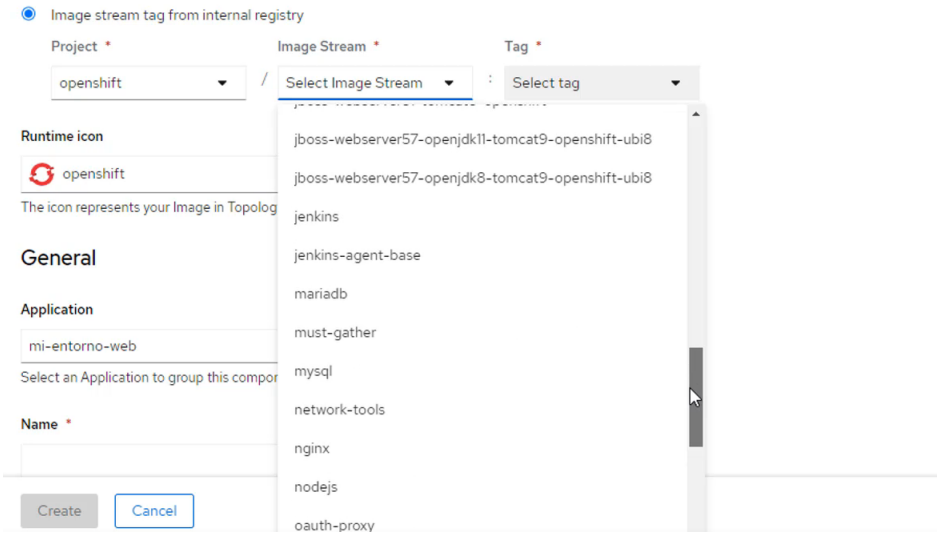
Para la version web podemos ver el nombre de la aplicación como esta arriba, dentro de este puede existir varios deployments



Crear desde una imagen interna



El proyecto openshift ya viene con varias imagen para reutilizar

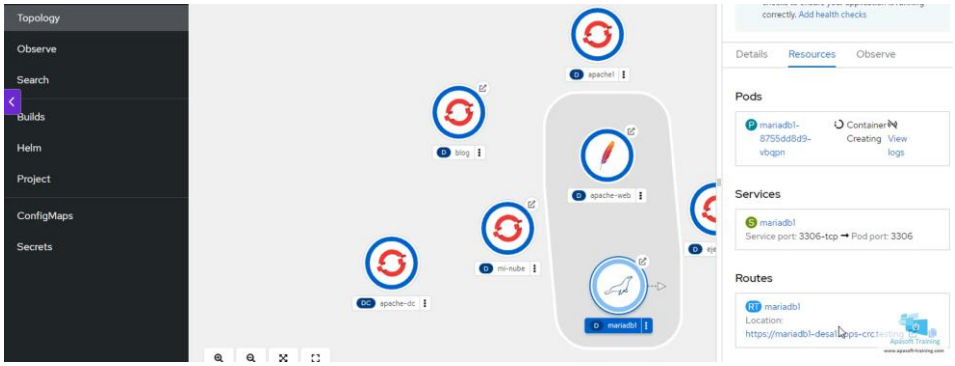


Se puede consultar desde la consola tambien

oc get is -n openshift

Buscar uno en concreto

Agrupacion de la aplicaciones



Comandos interesantes

Entrar a la consola de pod

```
oc rsh pod_name
```

Modificar en caliente el archivo de openshift (se despliega de nuevo el pod)

```
oc edit dc ejemplo
```

Esto se hace porque tiene un trigger configurado

```
      terminationGracePeriodSeconds: 30
  test: false
  triggers:
  - type: ConfigChange
status:
```

Ver variables de entorno de un recurso de openshift como DC, PODs, etc

```
oc set env pod/ejemplo-variables --list
oc set env dc/ejemplo-variables --list
```

```
[openshift@curso Variables]$ oc set env poc/ejemplo-variables-2-ljml9 --list
error: the server doesn't have a resource type "poc"
[openshift@curso Variables]$ oc set env pod/ejemplo-variables-2-ljml9 --list
# pods/ejemplo-variables-2-ljml9, container variables
NOMBRE=CURSO DE OPENSHIFT
PROPIETARIO=Apasoft Training
VERSION=v1
[openshift@curso Variables]$ oc set env dc/ejemplo-variables --list
# deploymentconfigs/ejemplo-variables, container variables
NOMBRE=CURSO DE OPENSHIFT
PROPIETARIO=Apasoft Training
VERSION=v1
```

Agregar nuevas varibles

```
oc set env pod/ejemplo-variables RESPONSABLE=pedro
oc set env pod/ejemplo-variables --list
```

Se vuelven a desplegar nuevos pods

```
[openshift@curso Variables]$ oc set env dc/ejemplo-variables RESPONSABLE=pedro
deploymentconfig.apps.openshift.io/ejemplo-variables updated
[openshift@curso Variables]$ oc set env dc/ejemplo-variables --list
# deploymentconfigs/ejemplo-variables, container variables
NOMBRE=CURSO DE OPENSHIFT
PROPIETARIO=Apasoft Training
VERSION=v1
RESPONSABLE=pedro
[openshift@curso Variables]$ oc get pod
NAME                          READY   STATUS        RESTARTS   AGE
ejemplo-variables-1-deploy    0/1     Completed     0          13m
ejemplo-variables-2-9xm6d     1/1     Terminating   0          11m
ejemplo-variables-2-deploy    0/1     Completed     0          11m
ejemplo-variables-2-ljml9     1/1     Terminating   0          10m
ejemplo-variables-3-deploy    0/1     Completed     0          25s
ejemplo-variables-3-gf8kd     1/1     Running       0          20s
ejemplo-variables-3-hpwtw     1/1     Running       0          16s
[openshift@curso Variables]$
```

Sobre escribir variables

```
oc set env pod/ejemplo-variables --overwrite RESPONSABLE=rosa
oc set env pod/ejemplo-variables --list
```

Recordar que esto lanza una nueva version

```
[openshift@curso Variables]$ oc set env dc/ejemplo-variables --list
# deploymentconfigs/ejemplo-variables, container variables
NOMBRE=CURSO DE OPENSHIFT
PROPIETARIO=Apasoft Training
VERSION=v1
RESPONSABLE=Rosa
[openshift@curso Variables]$ oc get dc
NAME               REVISION   DESIRED   CURRENT   TRIGGERED BY
ejemplo-variables  4          2         2         config
```

Quitar variables

```
oc set env pod/ejemplo-variables RESPONSABLE-
oc set env pod/ejemplo-variables --list
```

```
[openshift@curso Variables]$ oc get dc
NAME               REVISION   DESIRED   CURRENT   TRIGGERED BY
ejemplo-variables  5          2         2         config
[openshift@curso Variables]$
```

Tambien lo puedes ver con el comando edit

```
oc edit dc/ejemplo-variables
```

```
  spec:
    containers:
    - env:
      - name: NOMBRE
        value: CURSO DE OPENSHIFT
      - name: PROPIETARIO
        value: Apasoft Training
      - name: VERSION
        value: v1
      - name: RESPONSABLE
        value: Rosa
      image: gcr.io/google-samples/node-hello:1.0
      imagePullPolicy: IfNotPresent
      name: variables
```