



## **Workshop Introducción a Openshift Container Platform**

## Laboratorio Openshift

Los siguientes ejercicios se van a ejecutar en cada una de las PC de los asistentes, conectándose directamente a un ambiente que se les proveerá para el curso.

Para el laboratorio se va a utilizar los siguientes recursos:

- Acceso a la interfaz gráfica de Openshift
- Acceso al CLI para la ejecución de comandos desde la terminal de Openshift

**Nota:** El presentador les dará acceso a las URL necesarias.

### Prerequisitos

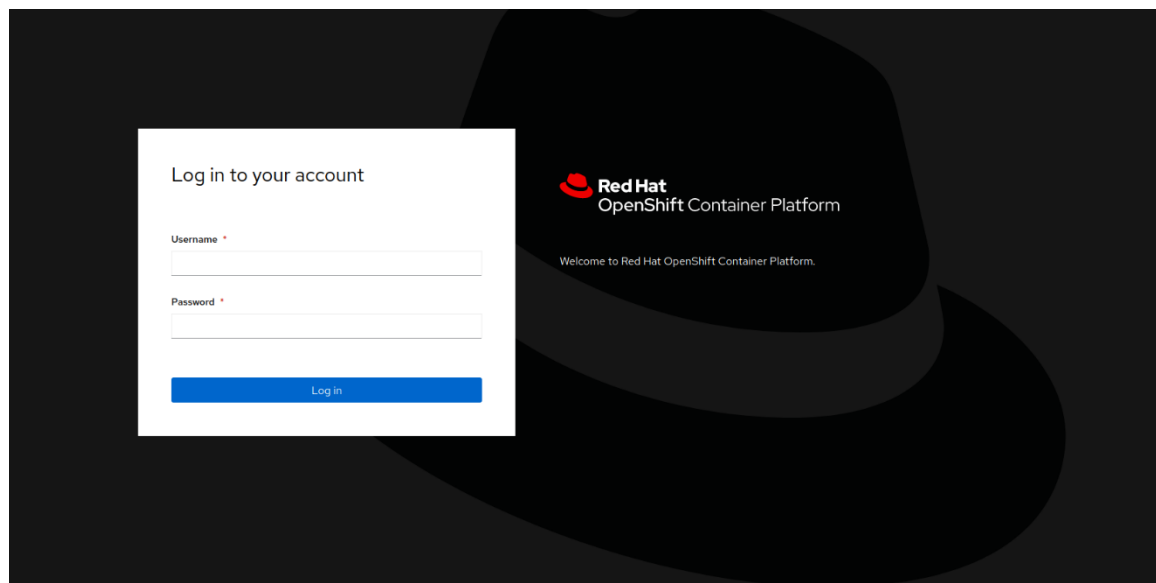
- a) Se le brindará una URL para acceder al ambiente del curso.

**Importante:** Dentro de los ejercicios de los laboratorios deberá sustituir los valores de color rojo por el número de usuario asignado previamente por el presentador.

### Acceso a la interfaz Openshift

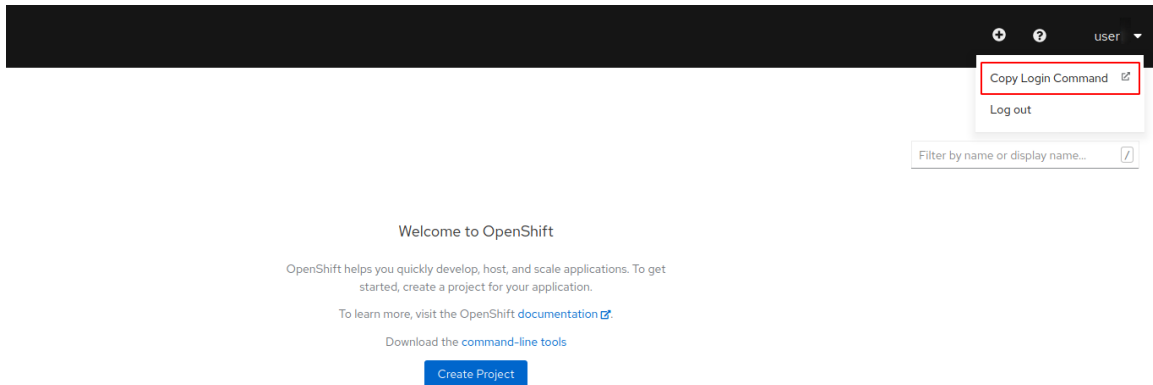
Una vez con acceso a las URL, loguearse en la consola web de Openshift Container Platform, para ello ingresaremos los siguientes datos.

- Usuario: userX
- Password: openshift

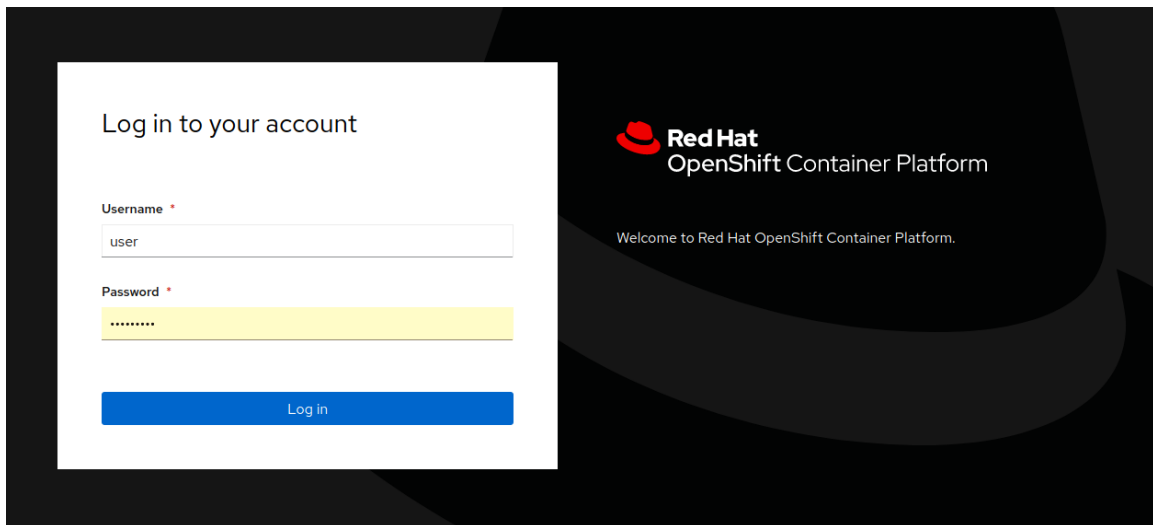


b) Al ingresar a la consola, nos vamos a la esquina derecha.

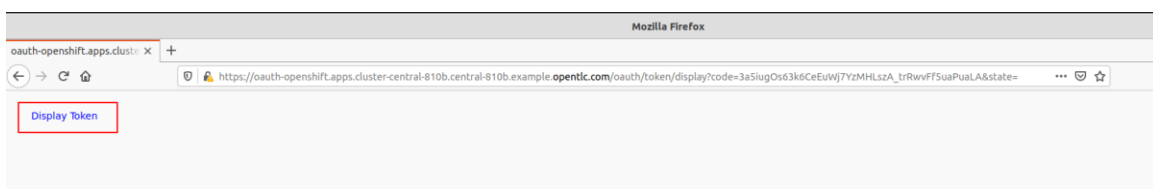
Y copiamos el login command. Para ellos damos clic en **“Copy Login Command”** para poder tener acceso a nuestros laboratorios desde una terminal.



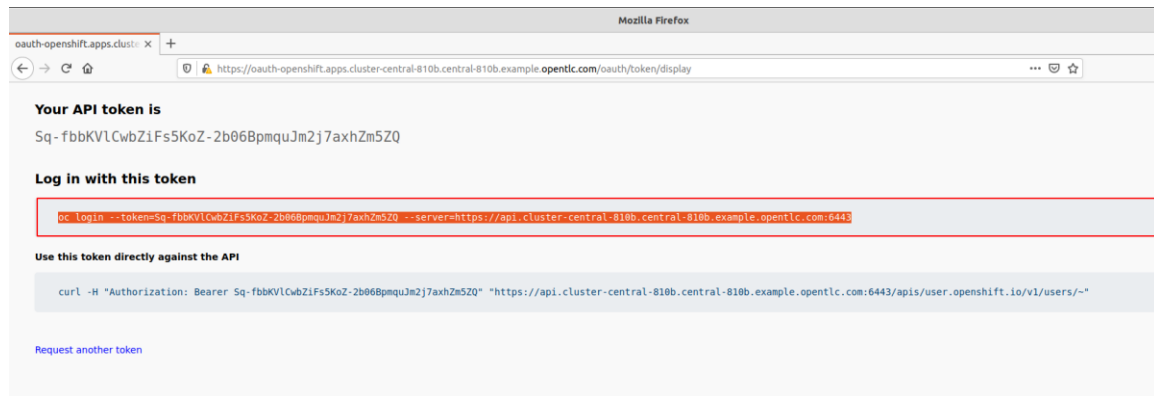
Indicamos nuevamente nuestro usuario y contraseña y luego Login



En la siguiente pantalla damos clic en **“Display Token”**



Y copiamos el valor contenido en la leyenda **“Log in with this Token”** y ese contenido lo pegamos en la consola del CLI que veremos a continuación.

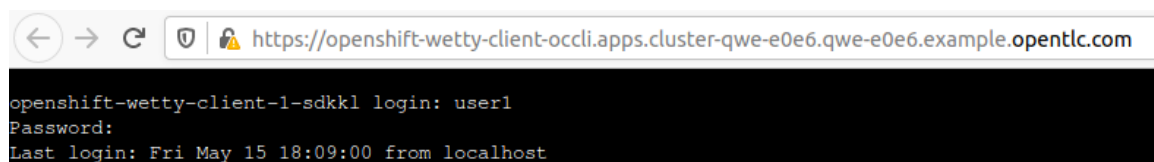


## Acceso a la terminal Openshift CLI

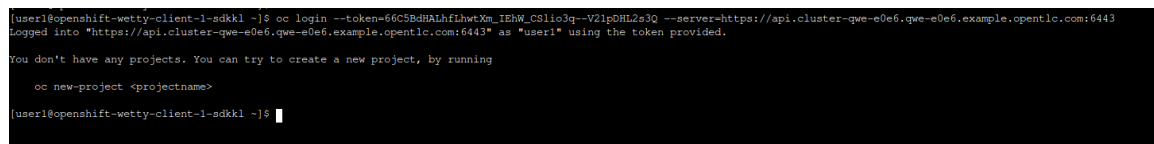
Se ingresa a la URL de la terminal que se le brindó al inicio del laboratorio prevista con el usuario de la consola del CLI que tiene el siguiente formato:

- Usuario: userx
- Password: Passwordx

Donde el valor “x” es el número que se le asignó para el usuario del curso.



Y luego se ingresa el token obtenido desde la interfaz de Openshift en el paso anterior, para loguearnos dentro de nuestro ambiente y poder ejecutar el laboratorio.



Ya luego de esto tenemos nuestro ambiente listo para iniciar con los laboratorios

## 1. Creación de proyecto desde línea de comandos OC.

1.1 Una vez logueados en la consola de OC, creamos el nuevo proyecto con el nombre “userx-lab1”, de la siguiente forma, debería dar el siguiente output.

```
$ oc new-project userx-lab1
```

Now using project "userx-lab1" on server "https://api.cluster-central-810b.central-810b.example.opentlc.com:6443".

You can add applications to this project with the 'new-app' command. For example, try:

```
oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git
```

to build a new example application in Ruby.

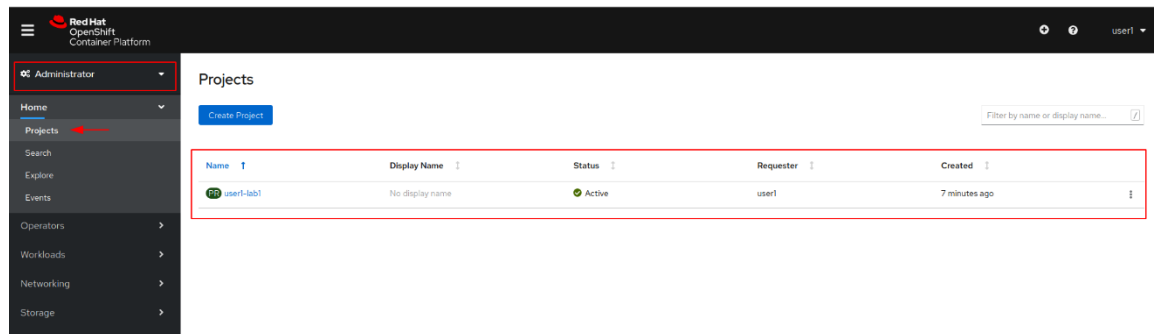
1.2 Verificamos el nuevo proyecto “**oc new-project userx-lab1**” creado desde la línea de comandos oc.

```
$ oc get projects
```

NAME	DISPLAY NAME	STATUS
userx-lab1		Active

1.3 También lo podemos verificar en la consola web.

En la vista Administrator > Home > Projects



## 2. Creación de una aplicación desde línea de comandos OC.

### 2.1 Creamos un nuevo proyecto

```
$ oc new-project userx-lab2
```

Now using project "**oc new-project userx-lab2**" on server

"https://master.na311.openshift.opentlc.com:443".

You can add applications to this project with the 'new-app' command. For example, try:

```
oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git
```

to build a new example application in Ruby.

### 2.2 Crear una nueva aplicación utilizando la imagen latest de wildfly disponible en Docker Hub, para ellos vamos a ejecutar el siguiente comando

```
$ oc new-app docker.io/jboss/wildfly:latest
```

--> Found Docker image 254d174 (2 weeks old) from docker.io for  
"docker.io/jboss/wildfly:latest"

- \* An image stream tag will be created as "wildfly:latest" that will track this image
- \* This image will be deployed in deployment config "wildfly"
- \* Port 8080/tcp will be load balanced by service "wildfly"
- \* Other containers can access this service through the hostname "wildfly"

--> Creating resources ...

imagestream.image.openshift.io "wildfly" created

deploymentconfig.apps.openshift.io "wildfly" created

service "wildfly" 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 svc/wildfly'
```

Run 'oc status' to view your app.

2.3 Verificamos que se hayan creado los objetos de Openshift, entre ellos: **imagestream**, **deploymentconfig**, **pod**, **replicationcontroller** y **service**.

El **replication controller**, debe estar en el mismo valor: **Desired**, **Current** y **Ready**, esto indica que ya la aplicación inició de forma correcta en la cantidad de réplicas deseadas.

```
$ oc get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/wildfly-1-lfmv9	1/1	Running	0	1m

NAME	DESIRED	CURRENT	READY	AGE
replicationcontroller/wildfly-1	1	1	1	1m

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/wildfly	ClusterIP	172.30.74.215	<none>	8080/TCP	1m

NAME	REVISION	DESIRED	CURRENT	TRIGGERED BY
deploymentconfig.apps.openshift.io/wildfly	1	1	1	

config,image(wildfly:latest)

NAME	DOCKER REPO	TAGS	UPDATED
imagestream.image.openshift.io/wildfly	docker-registry.default.svc:5000/jjl-new-apps/wildfly	latest	About a minute ago

2.4 Los servicios se utilizan para la comunicación interna de OpenShift. De esta manera, otra aplicación no necesita saber la dirección IP real del pod o cuántos pods se están ejecutando para una aplicación determinada. Pero dado que los servicios no son accesibles fuera del clúster de OpenShift, debe crear una ruta para exponer el servicio al mundo exterior:

```
$ oc expose svc wildfly
```

```
route.route.openshift.io/wildfly exposed
```

2.5 Verificar la ruta creada:

```
$ oc get route
```

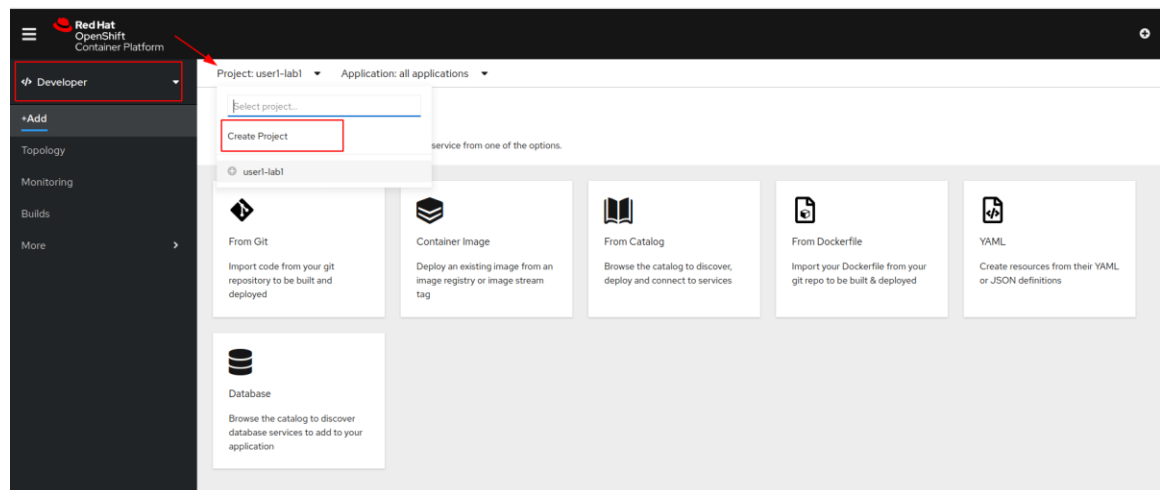
NAME	HOST/PORT	PATH	SERVICES	PORT
wildfly	wildfly-user1-lab1.apps.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com			
wildfly	8080-tcp	None		

2.6 Copiamos la ruta que se creó con el **expose** en un browser, en este caso es: **wildfly-user1-lab1.apps.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com**



### 3. Creación de aplicación PHP desde un repositorio GitHub.

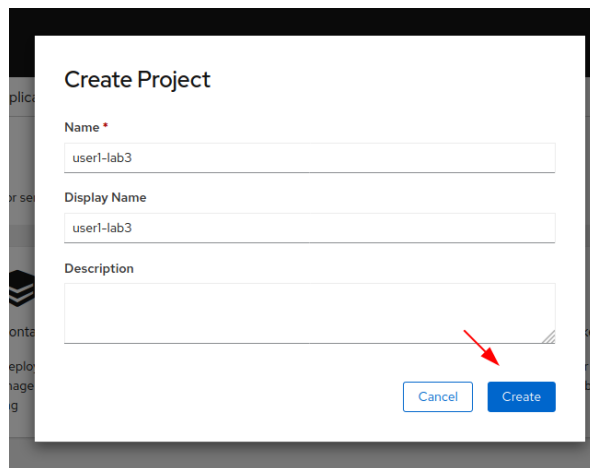
3.1 Para ello vamos a la vista **Developer** y agregamos un proyecto desde la lista desplegable llamada **Project** y damos clic en **Create Project** como se muestra la figura adjunta.



3.2 Le indicamos el nombre **userx-lab3**

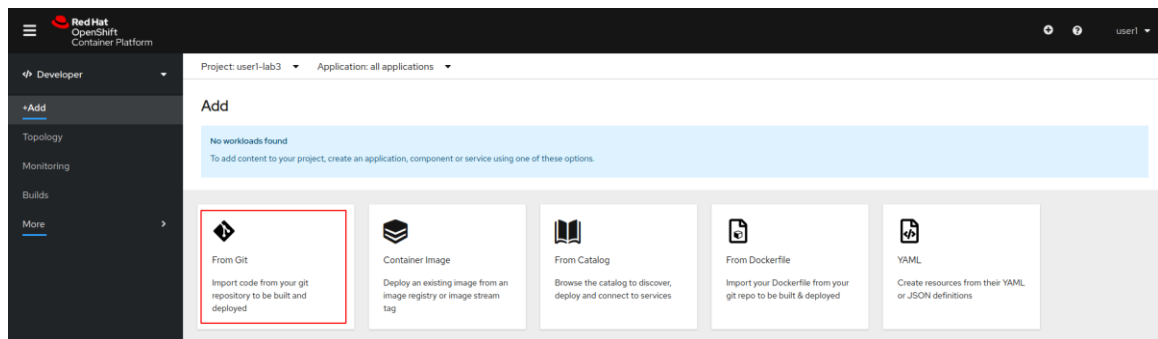
- Name: **userx-lab3**
- Display Name: **userx-lab3**
- Description: **Aplicación PHP**





The image shows a 'Create Project' dialog box in the OpenShift web console. It has three input fields: 'Name' with the value 'user1-lab3', 'Display Name' with the value 'user1-lab3', and an empty 'Description' field. At the bottom right, there are two buttons: 'Cancel' and 'Create'. A red arrow points to the 'Create' button.

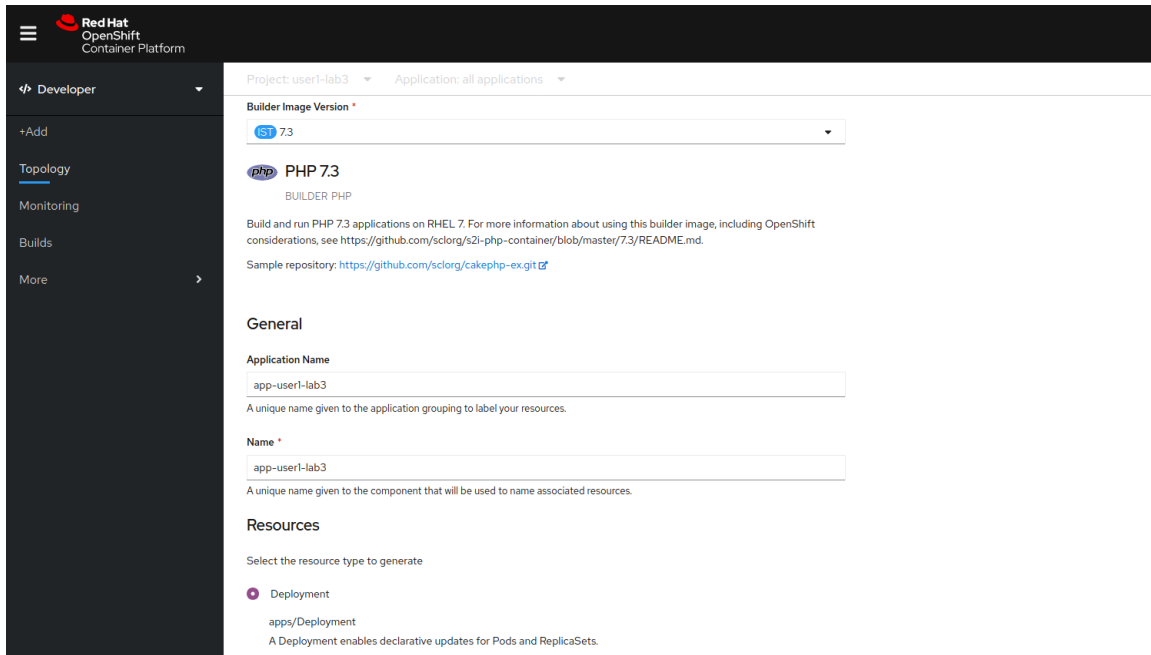
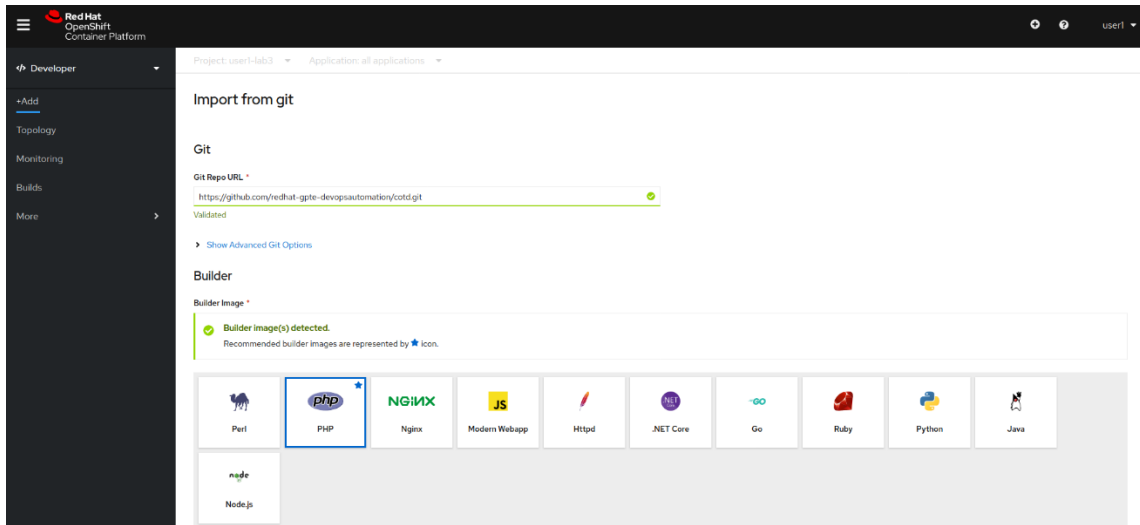
3.3 Procedemos a irnos a la opción **From Git** para crear una aplicación desde un repositorio



3.4 Procedemos a completar la información con los siguientes datos:

- **Git Repo URL:** <https://github.com/redhat-gpte-devopsautomation/cotd.git>
- **Application Name:** app-userx-lab3
- **Name:** app-userx-lab3

Quedando de la siguiente manera:



3.5 Procedemos a crear nuestra aplicación, luego de estos pasos tan sencillos

The screenshot shows the Red Hat OpenShift Developer console interface. On the left is a dark sidebar with a menu containing 'Developer', '+Add', 'Topology', 'Monitoring', 'Builds', and 'More'. The main area is titled 'Project: user1-lab3' and 'Application: all applications'. It contains a form for creating a new application. The 'Application Name' field is filled with 'app-user1-lab3'. Below it, a 'Name' field is also filled with 'app-user1-lab3'. The 'Resources' section has two radio buttons: 'Deployment' (selected) and 'Deployment Config'. The 'Advanced Options' section has a checked checkbox 'Create a route to the application'. At the bottom, there are 'Create' and 'Cancel' buttons. A red arrow points to the 'Create' button.

Red Hat OpenShift Container Platform

Developer

+Add

Topology

Monitoring

Builds

More

Project: user1-lab3 Application: all applications

Application Name

app-user1-lab3

A unique name given to the application grouping to label your resources.

Name \*

app-user1-lab3

A unique name given to the component that will be used to name associated resources.

Resources

Select the resource type to generate

☒ Deployment

apps/Deployment

A Deployment enables declarative updates for Pods and ReplicaSets.

☐ Deployment Config

apps.openshift.io/DeploymentConfig

A Deployment Config defines the template for a pod and manages deploying new images or configuration changes

Advanced Options

☒ Create a route to the application

Exposes your application at a public URL.

Click on the names to access advanced options for [Routing](#), [Build Configuration](#), [Deployment](#), [Scaling](#), [Resource Limits](#) and [Labels](#).

Create Cancel

3.6 Desde la vista **Topology** podemos ver nuestro pod y dándole clic vemos todos atributos configurados para nuestra aplicación.

Desde la pestaña **Resources**, podremos ver el link para acceder a nuestra aplicación.

The screenshot shows the Red Hat OpenShift Developer console interface. The 'Topology' view is selected in the sidebar, showing a diagram of the application architecture. A red box highlights the 'Topology' menu item. The main area displays a diagram of the application architecture, showing a pod labeled 'app-user1-lab3'. A red box highlights the pod. The right sidebar shows the 'Resources' panel for the application 'app-user1-lab3'. The 'Resources' panel has tabs for 'Details', 'Resources', and 'Monitoring'. The 'Resources' tab is selected, showing a list of pods, builds, services, and routes. A red box highlights the 'Routes' section, which contains a link to the application: 'http://app-user1-lab3-user1-lab3.apps.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com/gf'. A red arrow points to this link.

Red Hat OpenShift Container Platform

Developer

+Add

Topology

Monitoring

Builds

More

Project: user1-lab3 Application: all applications

Display

Find by name...

app-user1-lab3

Details Resources Monitoring

Pods

app-user1-lab3-6d944d9447-xx848 Running View logs

Builds

app-user1-lab3 Start Build

Build #1 is complete (3 minutes ago) View logs

Services

app-user1-lab3

Service port: 8080-tcp → Pod Port: 8080

Service port: 8443-tcp → Pod Port: 8443

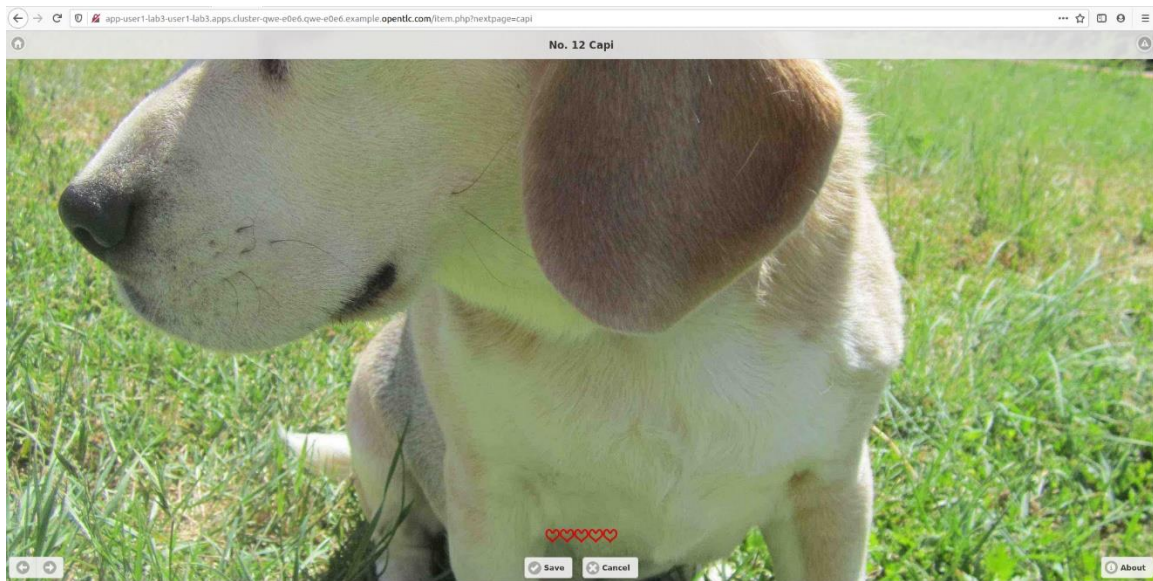
Routes

app-user1-lab3

Location

<http://app-user1-lab3-user1-lab3.apps.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com/gf>

3.7 Luego de dar clic en la ruta nuestra aplicación ya estará expuesta y será gestionada totalmente por Openshift



#### 4. Creación de aplicación PHP (Animales y cuidades) desde la línea de comando CLI.

Explora los detalles de los diversos objetos y observa los eventos asociados. Finalmente, crea una segunda versión de la aplicación y una ruta A / B para la aplicación, y luego observa cómo se enruta el tráfico a las dos versiones de la aplicación.

##### 4.1 Crear un **proyecto** con el nombre “**userx-lab4**”.

```
$ oc new-project userx-lab4
```

Now using project "**oc new-project user1-lab4**" on server

"https://master.na311.openshift.opentlc.com:443".

You can add applications to this project with the 'new-app' command. For example, try:

```
oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git
```

to build a new example application in Ruby.

4.2 Crear una nueva aplicación desde el repositorio de git: <https://github.com/redhat-gpte-devopsautomation/cotd.git>

```
$ oc new-app https://github.com/redhat-gpte-devopsautomation/cotd.git
```

```
--> Found image 6eeec1d (11 months old) in image stream "openshift/php" under tag "7.1" for "php"
```

Apache 2.4 with PHP 7.1

-----

PHP 7.1 available as container is a base platform for building and running various PHP 7.1 applications and frameworks. PHP is an HTML-embedded scripting language. PHP attempts to make it easy for developers to write dynamically generated web pages. PHP also offers built-in database integration for several commercial and non-commercial database management systems, so writing a database-enabled webpage with PHP is fairly simple. The most common use of PHP coding is probably as a replacement for CGI scripts.

Tags: builder, php, php71, rh-php71

- \* The source repository appears to match: php
- \* A source build using source code from <https://github.com/redhat-gpte-devopsautomation/cotd.git> will be created
  - \* The resulting image will be pushed to image stream tag "cotd:latest"
  - \* Use 'start-build' to trigger a new build
  - \* This image will be deployed in deployment config "cotd"
  - \* Ports 8080/tcp, 8443/tcp will be load balanced by service "cotd"
  - \* Other containers can access this service through the hostname "cotd"

```
--> Creating resources ...
```

```
imagestream.image.openshift.io "cotd" created
```

```
buildconfig.build.openshift.io "cotd" created
```

```
deploymentconfig.apps.openshift.io "cotd" created
```

```
service "cotd" created
```

```
--> Success
```

Build scheduled, use 'oc logs -f bc/cotd' to track its progress.

Application is not exposed. You can expose services to the outside world by executing one or more of the commands below:

```
'oc expose svc/cotd'
```

Run 'oc status' to view your app.

4.3 Creamos la ruta para poder acceder la aplicación desde fuera del clúster de Openshift.

```
$ oc expose svc/cotd
route.route.openshift.io/cotd exposed
```

4.4 Revisamos los logs del build.

```
$ oc logs -f bc/cotd
Cloning "https://github.com/redhat-gpte-devopsautomation/cotd.git" ...
Commit:      b53da24b6cfad8e31f0706f9ef8936761cba97e0 (Merge pull
request #1 from StefanoPicozzi/master)
Author:      Wolfgang Kulhanek <wkulhanek@users.noreply.github.com>
Date:   Thu Jan 11 07:38:09 2018 -0500
Using docker-
registry.default.svc:5000/openshift/php@sha256:7a8b79ebc15ebf8e79f6b8624cd028c7
87d7d23d1b36677d7ee1c6e0ef1f3349 as the s2i builder image
---> Installing application source...
=> sourcing 20-copy-config.sh ...
---> 18:30:15   Processing additional arbitrary httpd configuration provided by s2i ...
=> sourcing 00-documentroot.conf ...
=> sourcing 50-mpm-tuning.conf ...
=> sourcing 40-ssl-certs.sh ...

Pushing image docker-registry.default.svc:5000/managing-apps/cotd:latest ...
Pushed 5/6 layers, 87% complete
Pushed 6/6 layers, 100% complete
Push successful
```

4.5 Verificamos que se hayan creado los objetos de Openshift, entre ellos: **imagestream**, **deploymentconfig**, **pod**, **replicationcontroller** y **service**.

El **replication controller**, debe estar en el mismo valor: **Desired**, **Current** y **Ready**, esto indica que ya la aplicación inició de forma correcta en la cantidad de réplicas deseadas.

```
$ oc get all
```

NAME	READY	STATUS	RESTARTS	AGE
pod/cotd-1-build	0/1	Completed	0	1h
pod/cotd-1-zj97x	1/1	Running	0	1h

NAME	DESIRED	CURRENT	READY	AGE
replicationcontroller/cotd-1	1	1	1	1h

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
service/cotd	ClusterIP	172.30.43.124	<none>	8080/TCP,8443/TCP	1h

NAME	REVISION	DESIRED	CURRENT	TRIGGERED BY
deploymentconfig.apps.openshift.io/cotd	1	1	1	config,image(cotd:latest)

NAME	TYPE	FROM	LATEST
buildconfig.build.openshift.io/cotd	Source	Git	1

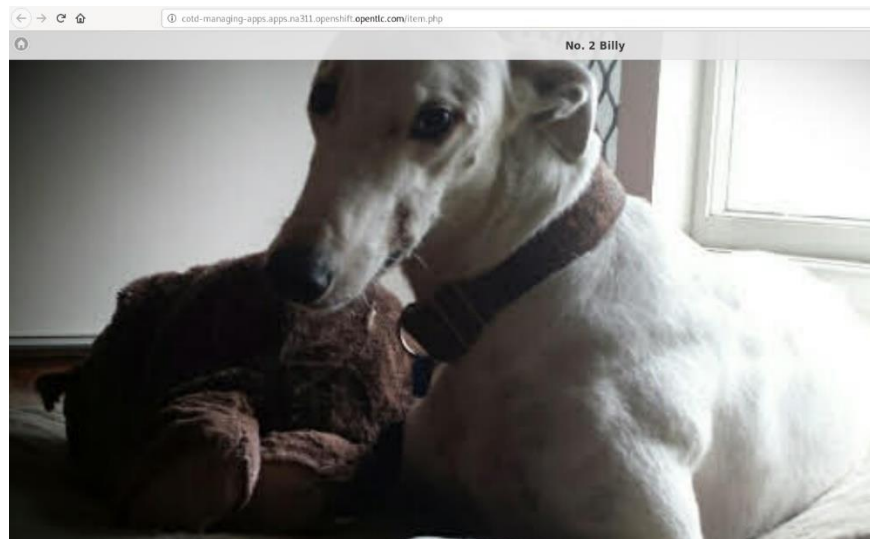
NAME	TYPE	FROM	STATUS	STARTED	DURATION
build.build.openshift.io/cotd-1	Source	Git@b53da24	Complete	About an hour ago	23s

NAME	DOCKER REPO	TAGS	UPDATED
imagestream.image.openshift.io/cotd	docker-registry.default.svc:5000/managing-apps/cotd	latest	About an hour ago

NAME	HOST/PORT	PATH	SERVICES	PORT
route.route.openshift.io/cotd	cotd-user1-lab4.apps.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com	cotd	8080-tcp	None

Lo marcado en rojo es la ruta que debemos copiar en nuestro navegador para acceder a la aplicación.

4.6 Verificamos el acceso a la aplicación PHP, es una aplicación que en cada refrescamiento muestra imágenes de mascotas.



4.7 OpenShift hace que sea extremadamente fácil escalar una aplicación simplemente escalando el controlador de replicación o la configuración de implementación. Solo escalaría el controlador de replicación directamente en casos excepcionales cuando no está controlado por una configuración de implementación.

Cuando cambia el número de réplicas de pod solicitadas, OpenShift hace girar nuevos pods o los elimina. Puede usar el comando `oc scale` para cambiar el número de pods solicitados para una aplicación.

Verificamos la cantidad de pods que existen antes de crear más réplicas y para verificar la cantidad de réplicas existentes.

```
$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
cotd-1-build	0/1	Completed	0	1h
cotd-1-zj97x	1/1	Running	0	1h

4.8 Hacemos la escalación directamente en el **deployment config "cotd"** a **3 réplicas**.

```
$ oc scale dc cotd --replicas=3
```

[deploymentconfig.apps.openshift.io/cotd/scaled](https://deploymentconfig.apps.openshift.io/cotd/scaled)

4.9 Verificamos la cantidad de réplicas existentes después de la escalación.

```
$ oc get pods
```

NAME	READY	STATUS	RESTARTS	AGE
cotd-1-build	0/1	Completed	0	1h



cotd-1-jkttb	1/1	Running	0	35s
cotd-1-sqtw7	1/1	Running	0	35s
cotd-1-zj97x	1/1	Running	0	1h

4.10 Espere verse seis **endpoints** para la ruta, 3 para el puerto 8443 y 3 para el puerto 8080. OpenShift enruta el tráfico de manera transparente a cada uno de estos pods.

**\$ oc describe route cotd**

```

Name:                cotd
Namespace:           managing-apps
Created:             About an hour ago
Labels:              app=cotd
Annotations:         openshift.io/host.generated=true
Requested Host:      cotd-user1-lab4.apps.cluster-qwe-e0e6.qwe-
e0e6.example.opentlc.com
                    exposed on router router about an hour ago
Path:                <none>
TLS Termination:     <none>
Insecure Policy:     <none>
Endpoint Port:       8080-tcp

Service:            cotd
Weight:             100 (100%)
Endpoints:          10.1.11.230:8443, 10.1.12.92:8443, 10.1.8.157:8443 + 3 more...

```

4.11 Ahora escalamos de nuevo a un pod.

**\$ oc scale dc cotd --replicas=1**

deploymentconfig.apps.openshift.io/cotd scaled

4.12 Verificamos de nuevo los pods y ahora solo hay 1.

**\$ oc get pods**

NAME	READY	STATUS	RESTARTS	AGE
cotd-1-build	0/1	Completed	0	1h
cotd-1-zj97x	1/1	Running	0	1h

OpenShift incluye la capacidad de establecer **dos o más back-end** para cualquier ruta dada. Esto se puede usar para las **pruebas A / B** donde se implementan dos versiones de la aplicación al mismo tiempo y los clientes se envían aleatoriamente a una de las dos versiones. El enrutamiento A / B se usa con mayor frecuencia para las pruebas de la

interfaz de usuario, por ejemplo, para determinar qué versión de un sitio web genera más conversiones de ventas.

4.13 Realice una segunda versión de la aplicación utilizando **cities** como **selector**. Esto le indicará a la aplicación que muestre imágenes de ciudades en lugar del conjunto predeterminado de imágenes de animales.

```
$ oc new-app --name='cotd2' -l name='cotd2' https://github.com/redhat-gpte-devopsautomation/cotd.git -e SELECTOR=cities
```

```
--> Found image 6eeec1d (11 months old) in image stream "openshift/php" under tag "7.1" for "php"
```

```
Apache 2.4 with PHP 7.1
```

```
-----  
PHP 7.1 available as container is a base platform for building and running various PHP 7.1 applications and frameworks. PHP is an HTML-embedded scripting language. PHP attempts to make it easy for developers to write dynamically generated web pages. PHP also offers built-in database integration for several commercial and non-commercial database management systems, so writing a database-enabled webpage with PHP is fairly simple. The most common use of PHP coding is probably as a replacement for CGI scripts.
```

```
Tags: builder, php, php71, rh-php71
```

```
* The source repository appears to match: php  
* A source build using source code from https://github.com/redhat-gpte-devopsautomation/cotd.git will be created  
  * The resulting image will be pushed to image stream tag "cotd2:latest"  
  * Use 'start-build' to trigger a new build  
* This image will be deployed in deployment config "cotd2"  
* Ports 8080/tcp, 8443/tcp will be load balanced by service "cotd2"  
  * Other containers can access this service through the hostname "cotd2"
```

Ahora ha creado una segunda aplicación, **cotd2**. Debido a que no especificó un selector para la primera aplicación, esa aplicación utilizó las mascotas predeterminadas. Ahora puede determinar mirando la imagen de qué versión de la aplicación es.

4.14 Esta vez no expone el servicio **cotd2** como una **ruta**, pero lo agrega a su ruta anterior como otro **back-end** de ruta.

Con esto **50%** del tráfico se irá a la aplicación **cotd** y **50%** del tráfico se irá al **cotd2**.

```
$ oc set route-backends cotd cotd=50 cotd2=50
route.route.openshift.io/cotd backends updated
```

4.15 Verificamos que el cambio en la ruta se ha realizado de forma correcta.

```
$ oc describe route cotd
Name:                cotd
Namespace:           managing-apps
Created:             2 hours ago
Labels:              app=cotd
Annotations:         openshift.io/host.generated=true
Requested Host:      cotd-user1-lab4.apps.cluster-qwe-e0e6.qwe-
e0e6.example.opentlc.com
                    exposed on router router 2 hours ago
Path:                <none>
TLS Termination:     <none>
Insecure Policy:     <none>
Endpoint Port:       8080-tcp

Service:            cotd
Weight:              50 (50%)
Endpoints:           10.1.12.92:8443, 10.1.12.92:8080

Service:            cotd2
Weight:              50 (50%)
Endpoints:           10.1.8.171:8443, 10.1.8.171:8080
```

4.16 Opcional use **curl** para conectarse a la aplicación cada segundo usando la ruta:

Debe usar **curl** desde la línea de comandos porque todos los navegadores modernos usan cookies para identificar su sesión actual. Y cada vez que actualiza el navegador, se lo envía al mismo servicio al que se lo envió antes, lo cual tiene sentido, porque en el mundo real cuando realiza **pruebas A / B**, desea que una sesión de navegador determinada tenga afinidad con el servicio al que fue enrutado cuando se conectó por primera vez.

```
$ while true; do curl -s http://$(oc get route cotd --template='{{ .spec.host
}}')/item.php | grep "data/images" | awk '{print $5}'; sleep 1; done
data/images/pets/billie.jpg
data/images/cities/adelaide.jpg
data/images/pets/taffy.jpg
```

```
data/images/cities/canberra.jpg
data/images/pets/milo.jpg
data/images/cities/perth.jpg
data/images/pets/billy_2.jpg
data/images/pets/deedee.jpg
data/images/cities/wellington.jpg
data/images/pets/neo.jpg
data/images/cities/adelaide.jpg
```

Podemos ver como el **50%** del **tráfico** va **hacia cities** y el **50%** **hacia pets**.

## 5. Parksmap App Ejercicio

En esta práctica de laboratorio, implementaremos el componente web de la aplicación ParksMap, que también se llama parksmapy utiliza el mecanismo de descubrimiento de servicios de OpenShift para descubrir los servicios de back-end implementados.

5.1. Se debe crear un nuevo proyecto. Para ello vamos a la vista **Developer** y agregamos un proyecto desde la lista desplegable llamada **Project** y damos clic en **Create Project** como se muestra la figura adjunta.

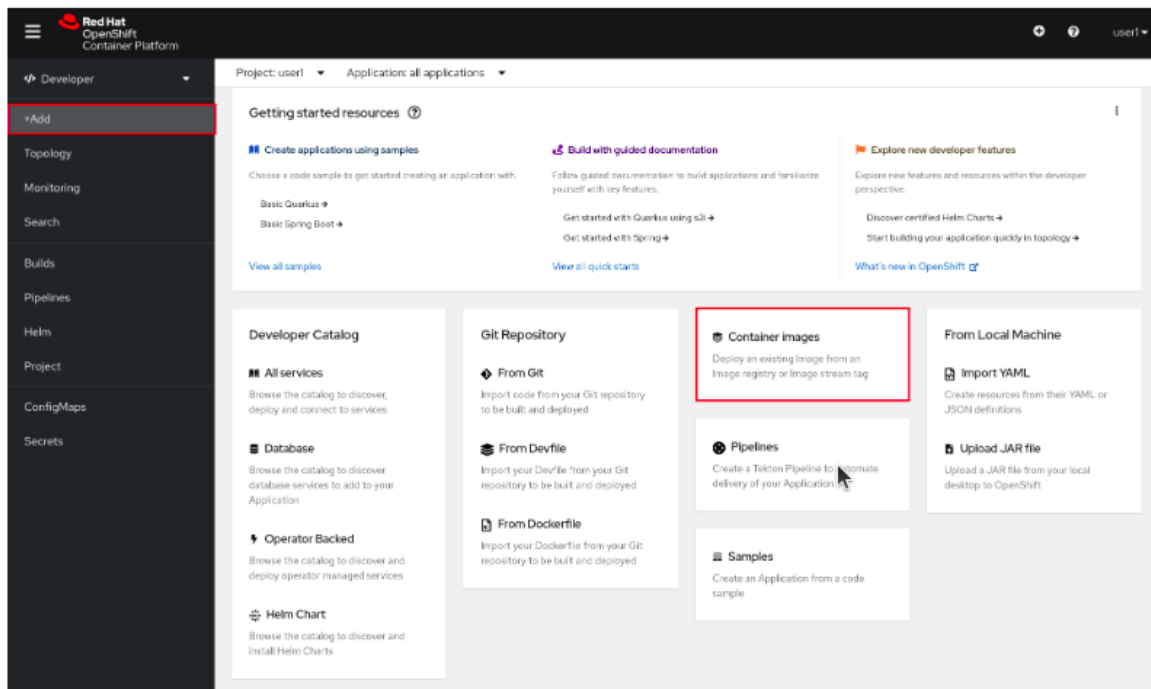
5.2. Le indicamos el nombre al proyecto:

Name: **userx-lab5**

Display Name: **userx-lab5**

Description: Parksmap App

5.3. En el menú de la izquierda, haz clic en +Agregar . Verá una pantalla donde tiene múltiples opciones para implementar la aplicación en OpenShift. Haga clic en Imagen de contenedor para abrir un cuadro de diálogo que le permitirá especificar la información de la imagen que desea implementar.



5.4. En el campo Nombre de la imagen, copie y pegue lo siguiente en el cuadro:  
**quay.io/openshiftroadshow/parksmat:latest**

OpenShift luego irá al registro de contenedor especificado e interrogará la imagen.

Su pantalla terminará luciendo algo como esto:

## Deploy Image

### Image

Deploy an existing image from an image stream or image registry.

☒ Image name from external registry

quay.io/openshiftroadshow/parksmat:latest



Validated

To deploy an image from a private repository, you must [create an image pull secret](#) with your image registry credentials.

☐ Allow images from insecure registries

☐ Image stream tag from internal registry

## General

**Application name**

parksmap-app

A unique name given to the Application grouping to label your resources.

**Name \***

parksmap

A unique name given to the component that will be used to name associated resources.

## Resources

Select the resource type to generate

☒ Deployment

apps/Deployment

A Deployment enables declarative updates for Pods and ReplicaSets.

☐ DeploymentConfig

apps.openshift.io/DeploymentConfig

A DeploymentConfig defines the template for a Pod and manages deploying new Images or configuration changes.

## Advanced options

☒ Create a route to the Application

Exposes your Application at a public URL

›

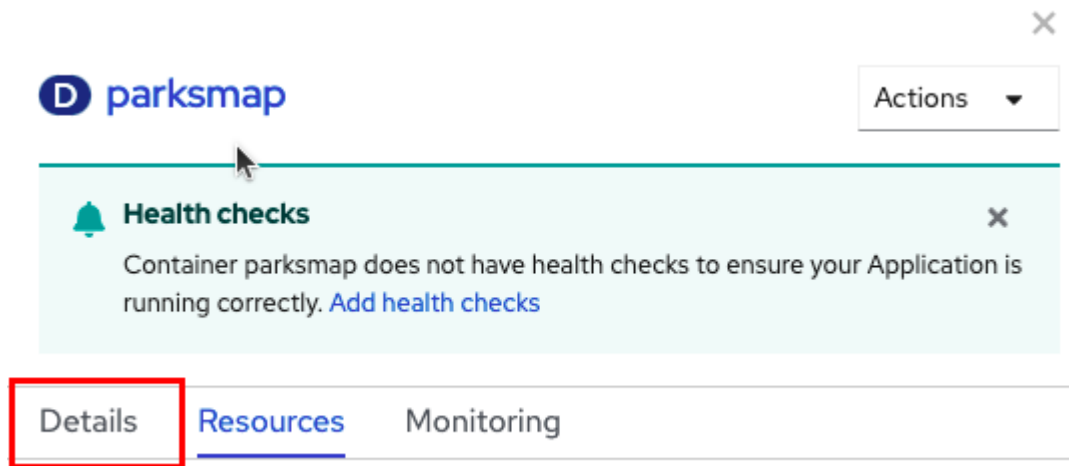
Show advanced Routing options

Click on the names to access advanced options for [Health checks](#), [Deployment](#), [Scaling](#), [Resource limits](#) and [Labels](#).

5.5. En **Runtime Icon**, puede seleccionar el icono que se usará en OpenShift Topology View para la aplicación. Puede dejar el ícono predeterminado de OpenShift o, dado que esta interfaz está hecha con Spring Boot, puede elegir spring-boot.

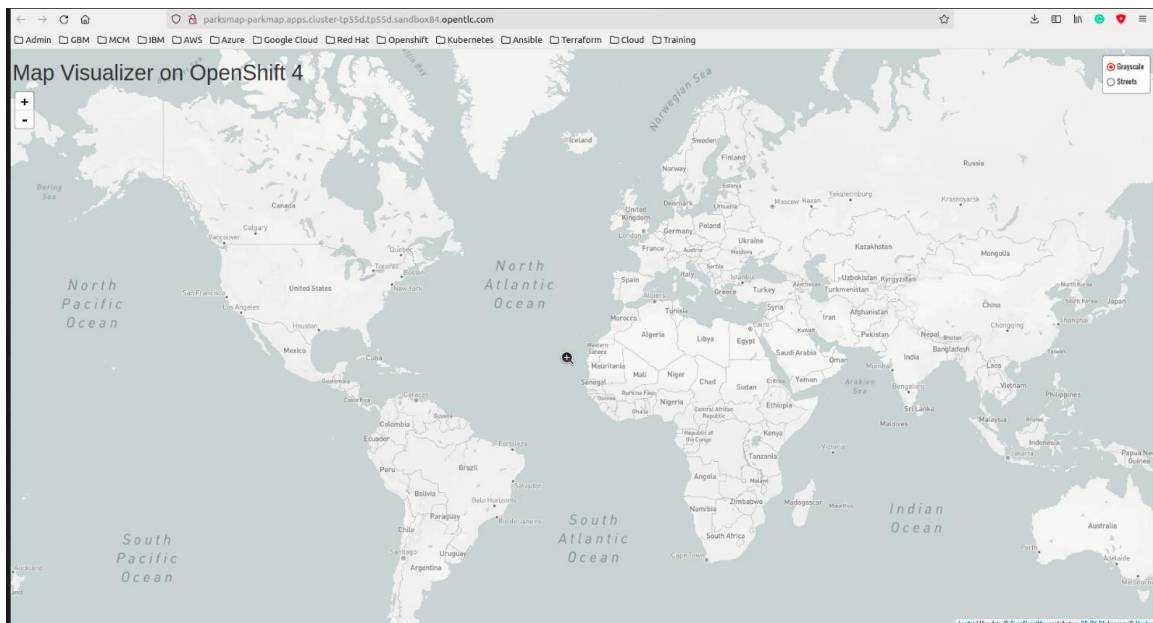
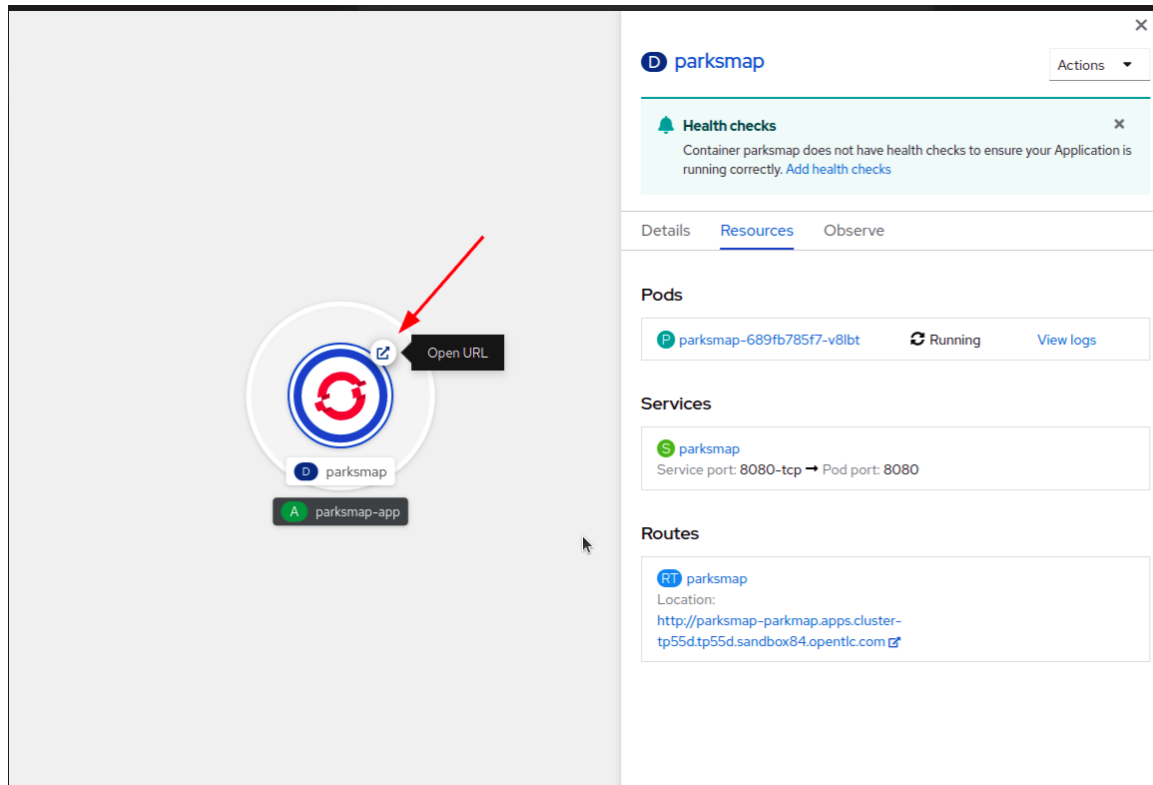
### 5.6. Ejercicio: Examinando el Pod.

Si hace clic en la parksmapi entrada en la vista Topología, verá información sobre esa configuración de implementación. La pestaña **Recursos** puede mostrarse de forma predeterminada. Si es así, haga clic en la pestaña **Detalles** .



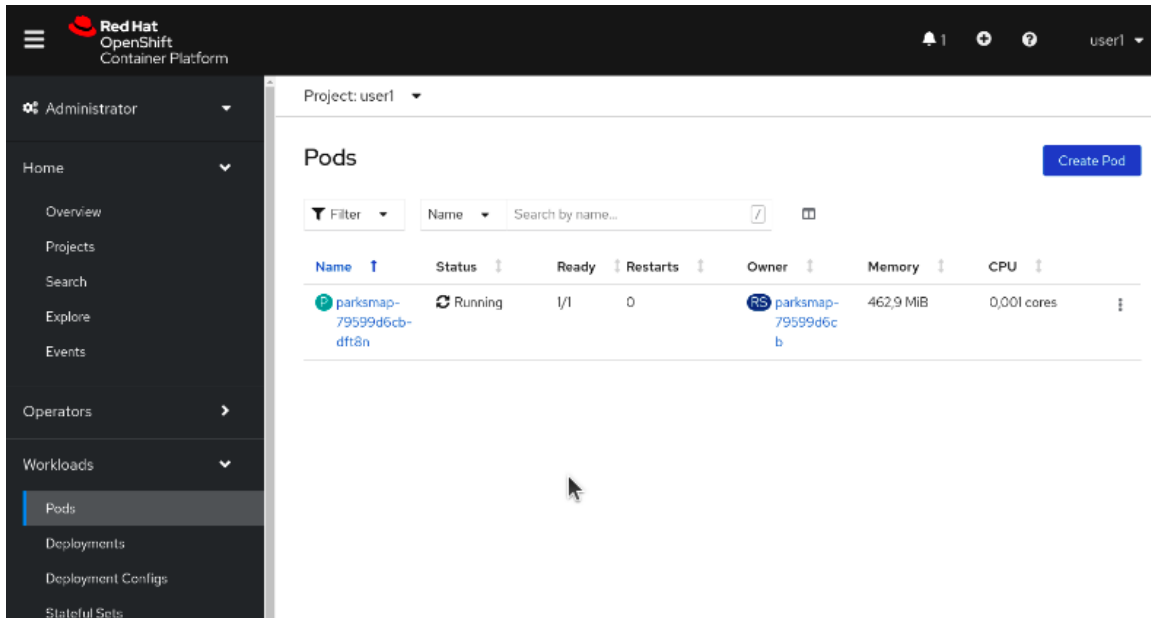
En ese panel, verá que hay un solo **Pod** creado por sus acciones.

### 5.7. Al darle click al link de la ruta, nos muestra la app que contiene un mapa mundial:



5.8. También puede obtener una lista de todos los Pods creados dentro de su Proyecto, navegando a Cargas de trabajo → Pods en la perspectiva del Administrador de la consola web.





Este **Pod** contiene un solo contenedor, que resulta ser la aplicación `parksmap`: una aplicación Spring Boot/Java simple.

5.9. También puede examinar **Pods** desde la línea de comandos:

**oc project userx-lab5**

**oc get pods**

```
[user5@openshift-wetty-client-79d584d887-9sflc ~]$ oc get pods
NAME                                READY    STATUS    RESTARTS   AGE
parksmap-689fb785f7-v8lbt          1/1      Running   0           5m31s
[user5@openshift-wetty-client-79d584d887-9sflc ~]$
```

## 5.10. Ejercicio de Escalado

Para obtener más detalles, podemos mirar en el **ReplicaSet (RS)**.

Eche un vistazo al **ReplicaSet (RS)** que se creó para usted cuando le dijo a OpenShift que levantara la `parksmap` imagen:

**oc get rs**

**oc get deployment**

Esto nos permite saber que, en este momento, esperamos que se implemente un **Pod** (Desired), y tenemos un **Pod** realmente implementado ( Current). Cambiando el número deseado, podemos decirle a OpenShift que queremos más o menos **Pods** .

**HorizontalPodAutoscaler** de OpenShift monitorea efectivamente el uso de la CPU de un conjunto de instancias y luego manipula los RC en consecuencia.

#### 5.11. Ejercicio: escalar la aplicación

Escalemos nuestra "aplicación" parksmapi hasta 2 instancias. Podemos hacer esto con el scalecomando. También puede hacer esto incrementando el Recuento deseado en la consola web de OpenShift. Elija uno de estos métodos; Es tu elección.

**oc scale --replicas=2 deployment/parksmapi**

5.12. También puede escalar hasta dos pods en la perspectiva del desarrollador. En la vista Topología, primero haga clic en la parksmapi configuración de implementación y seleccione la pestaña Detalles:

The screenshot shows the OpenShift console interface for the 'parksmap' deployment. On the left, a large circular icon with a green power button symbol is labeled 'parksmap'. Below it, a smaller icon labeled 'workshop' is visible. The main panel on the right displays the deployment details for 'parksmap' in the 'user1' namespace. A large blue circle with '1 pod' inside indicates the current state. To the right of this circle are up and down arrow icons. The deployment details include: Name: parksmap, Update Strategy: RollingUpdate, Namespace: user1, Max Unavailable: 25% of 1 pod, Labels: app=workshop, app.kubernetes.io/com...=parks..., app.kubernetes.io/inst...=parks..., app.kubernetes.io/par...=worksh..., app.openshift.io/run...=spring-b..., app.openshift.io/runtime-n...=us..., component=parksmap, role=frontend, Max Surge: 25% greater than 1 pod, Progress Deadline Seconds: 600 seconds, and Min Ready Seconds: Not Configured. At the bottom, there is a Pod Selector section with 'app=parksmap' and a Node Selector section which is empty. A mouse cursor is pointing at the up arrow icon next to the pod count.

parksmap

workshop

1 pod

Name: parksmap

Update Strategy: RollingUpdate

Namespace: user1

Max Unavailable: 25% of 1 pod

Labels: app=workshop, app.kubernetes.io/com...=parks..., app.kubernetes.io/inst...=parks..., app.kubernetes.io/par...=worksh..., app.openshift.io/run...=spring-b..., app.openshift.io/runtime-n...=us..., component=parksmap, role=frontend

Max Surge: 25% greater than 1 pod

Progress Deadline Seconds: 600 seconds

Min Ready Seconds: Not Configured

Pod Selector: app=parksmap

Node Selector:

5.13. A continuación, haga clic en el icono ^ junto a la visualización de pod para escalar hasta 2 pods.

The screenshot shows the Kubernetes Dashboard interface for a deployment named 'parksmap'. On the left, a large circular icon represents the deployment. The main panel displays the 'Details' tab, showing the deployment's configuration. A red box highlights the pod count, which is currently 1. The deployment is in the 'user1' namespace and has a pod selector of 'app=parksmap'. The update strategy is 'RollingUpdate'.

Para verificar que cambiamos el número de réplicas, emita el siguiente comando:  
**oc get rs**

NAME	DESIRED	CURRENT	READY	AGE
parksmap-65c4f8b676	2	2	2	23m

Puedes ver que ahora tenemos 2 réplicas. Verifiquemos la cantidad de pods con el **oc get pods** comando:

NAME	READY	STATUS	RESTARTS	AGE
parksmap-65c4f8b676-fxcrq	1/1	Running	0	92s
parksmap-65c4f8b676-k5gkk	1/1	Running	0	24m

```
[user5@openshift-wetty-client-79d584d887-9sflc ~]$ oc scale --replicas=2 deployment/parksmmap
deployment.apps/parksmmap scaled
[user5@openshift-wetty-client-79d584d887-9sflc ~]$ oc get rs
NAME                                DESIRED    CURRENT    READY    AGE
parksmmap-689fb785f7                2          2          2        11m
[user5@openshift-wetty-client-79d584d887-9sflc ~]$ oc get deployment
NAME    READY    UP-TO-DATE    AVAILABLE    AGE
parksmmap  2/2      2             2            11m
[user5@openshift-wetty-client-79d584d887-9sflc ~]$ oc get pods
NAME                                READY    STATUS    RESTARTS    AGE
parksmmap-689fb785f7-7gxzg          1/1      Running    0            51s
parksmmap-689fb785f7-v8lbt          1/1      Running    0            11m
[user5@openshift-wetty-client-79d584d887-9sflc ~]$
```

#### 5.14. Application "Self Healing"

Debido a que los RS de OpenShift **están** monitoreando constantemente para ver que la cantidad deseada de **Pods** realmente se está ejecutando, también puede esperar que OpenShift "arregle" la situación si alguna vez no es correcta. ¡Estarías en lo cierto!

Ya que tenemos dos **Pods** funcionando en este momento, veamos qué sucede si "accidentalmente" matamos uno. Ejecute el `oc get pods` comando nuevamente y elija un nombre de **pod** . Luego, haz lo siguiente:

**`oc delete pod parksmmap-65c4f8b676-k5gkk && oc get pods`**

```
pod "parksmmap-65c4f8b676-k5gkk" deleted
NAME                                READY    STATUS    RESTARTS    AGE
parksmmap-65c4f8b676-bjz5g          1/1      Running    0            13s
parksmmap-65c4f8b676-fxcrcq          1/1      Running    0            4m48s
```

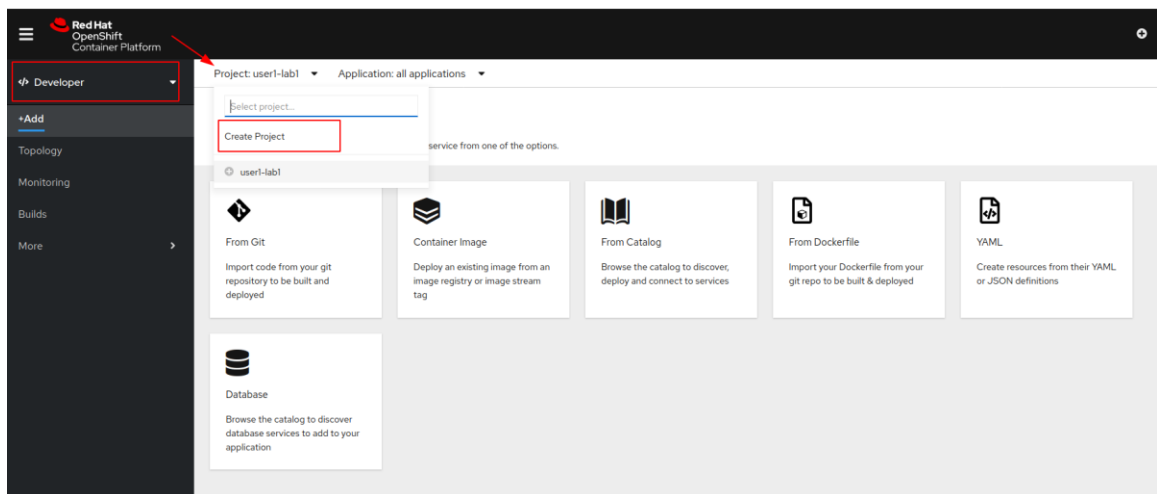
¿Notaste algo? Se eliminó un contenedor y ya se está creando un nuevo contenedor.

Además, los nombres de los **Pods** han cambiado ligeramente. Esto se debe a que OpenShift detectó casi de inmediato que el estado actual (1 **Pod** ) no coincidía con el estado deseado (2 **Pods** ) y lo solucionó programando otro **Pod** .

```
[user5@openshift-wetty-client-79d584d887-9sflc ~]$ oc delete pod parksmapi-689fb785f7-7gxzg && oc get pods
pod "parksmapi-689fb785f7-7gxzg" deleted
NAME          READY   STATUS             RESTARTS   AGE
parksmapi-689fb785f7-2r569  0/1     ContainerCreating   0           1s
parksmapi-689fb785f7-v8lbt  1/1     Running             0           14m
[user5@openshift-wetty-client-79d584d887-9sflc ~]$ oc get pods
NAME          READY   STATUS    RESTARTS   AGE
parksmapi-689fb785f7-2r569  1/1     Running   0           20s
parksmapi-689fb785f7-v8lbt  1/1     Running   0           14m
[user5@openshift-wetty-client-79d584d887-9sflc ~]$
```

## 6. Laboratorio Jenkins: Creando un entorno de CI / CD utilizando Jenkins con una aplicación Java EE

6.1 Se debe crear un nuevo proyecto. Para ello vamos a la vista **Developer** y agregamos un proyecto desde la lista desplegable llamada **Project** y damos clic en **Create Project** como se muestra la figura adjunta.



6.2 Le indicamos el nombre userx-lab6

- Name: **userx-lab6**
- Display Name: **userx-lab6**
- Description: **Servidor Jenkins**

### Create Project

Name \*

user1-lab6

Display Name

user1-lab6

Description

Servidor Jenkins

Cancel Create

6.3 Ir al catálogo, click en CI/CD, y click en Jenkins (Ephemeral).

Project: user1-lab6

#### Developer Catalog

Add shared apps, services, or source-to-image builders to your project from the Developer Catalog. Cluster admins can install additional apps which will show up here automatically.

All Items

CI/CD

Filter by keyword...

Group By: None

4 items

Jenkins

Jenkins

Jenkins (Ephemeral)

Jenkins (Ephemeral)

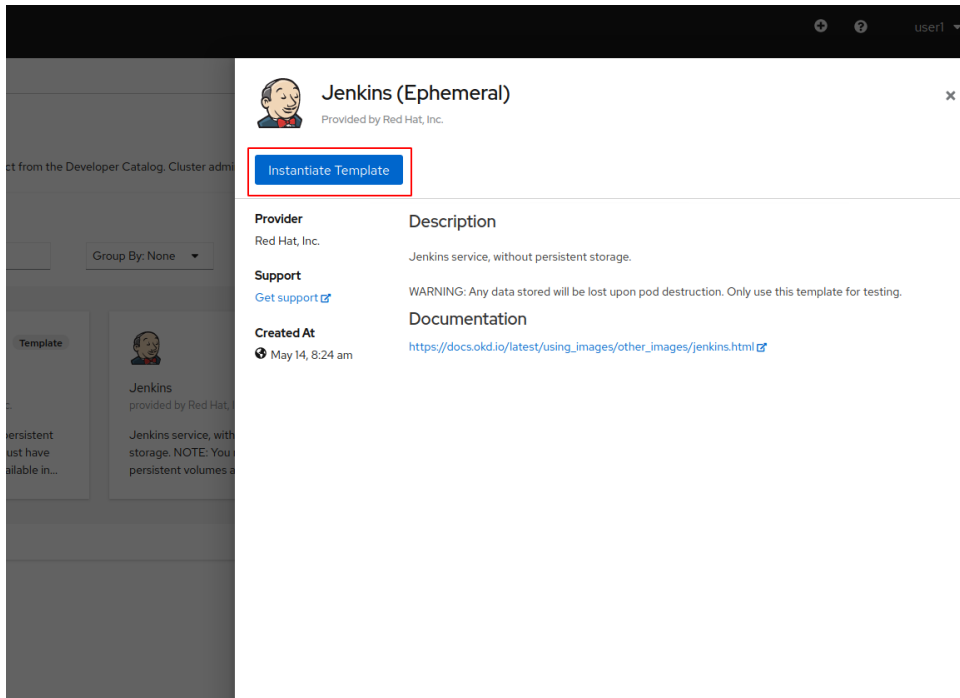
Jenkins service, with persistent storage. NOTE: You must have persistent volumes available in...

Jenkins service, with persistent storage. NOTE: You must have persistent volumes available in...

Jenkins service, without persistent storage. WARNING: Any data stored will be lost upon...

Jenkins service, without persistent storage. WARNING: Any data stored will be lost upon...

6.4 Lo seleccionamos y damos click en "Instantiate Template".



6.5 Se configura la aplicación de Jenkins con los siguientes valores.

#### Instantiate Template

<div><b>Namespace *</b> <div>user1-lab6</div></div> <div><b>Jenkins Service Name</b> <div>jenkins</div><p>The name of the OpenShift Service exposed for the Jenkins container.</p></div> <div><b>Jenkins JNLP Service Name</b> <div>jenkins-jnlp</div><p>The name of the service used for master/slave communication.</p></div> <div><b>Enable OAuth in Jenkins</b> <div>true</div><p>Whether to enable OAuth OpenShift integration. If false, the static account 'admin' will be initialized with the password 'password'.</p></div> <div><b>Memory Limit</b> <div>2Gi</div><p>Maximum amount of memory the container can use.</p></div> <div><b>Jenkins ImageStream Namespace</b> <div>openshift</div><p>The OpenShift Namespace where the Jenkins ImageStream resides.</p></div> <div><b>Disable memory intensive administrative monitors</b> <div>true</div><p>Whether to perform memory intensive, possibly slow, synchronization with the Jenkins Update Center on start. If true, the Jenkins core update monitor and site warnings monitor are disabled.</p></div>	<div> <b>Jenkins (Ephemeral)</b> INSTANT-APP JENKINS <a href="#">View documentation</a> <a href="#">Get support</a></div> <p>Jenkins service, without persistent storage.</p> <p>WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing.</p> <hr/> <p>The following resources will be created:</p> <ul style="list-style-type: none"><li>• DeploymentConfig</li><li>• RoleBinding</li><li>• Route</li><li>• Service</li><li>• ServiceAccount</li></ul>
--	--



#### Memory Limit

2Gi

Maximum amount of memory the container can use.

#### Jenkins ImageStream Namespace

openshift

The OpenShift Namespace where the Jenkins ImageStream resides.

#### Disable memory intensive administrative monitors

true

Whether to perform memory intensive, possibly slow, synchronization with the Jenkins Update Center on start. If true, the Jenkins core update monitor and site warnings monitor are disabled.

#### Jenkins ImageStreamTag

jenkins:2

Name of the ImageStreamTag to be used for the Jenkins image.

#### Allows use of Jenkins Update Center repository with invalid SSL certificate

false

Whether to allow use of a Jenkins Update Center that uses invalid certificate (self-signed, unknown CA). If any value other than 'false', certificate check is bypassed. By default, certificate check is enforced.

Create

Cancel

Y damos clic al botón “Create”

6.6 Una vez creado nuestro pod de Jenkins, desde la vista Topology podemos ver los parámetros de su configuración, la ruta de acceso y demás. Verificamos el estado del pod, damos unos minutos para que se complete la configuración. Para ello damos clic en la ruta mostrada en el apartado “Routes” para acceder a nuestro servidor Jenkins.

Project: user1-lab6 Application: all applications View shortcuts

Display 5 Find by name...

**jenkins** Actions

Details Resources Monitoring

**Pods**

jenkins-l-8qgrb Running View logs

**Builds**

No Build Configs found for this resource.

**Services**

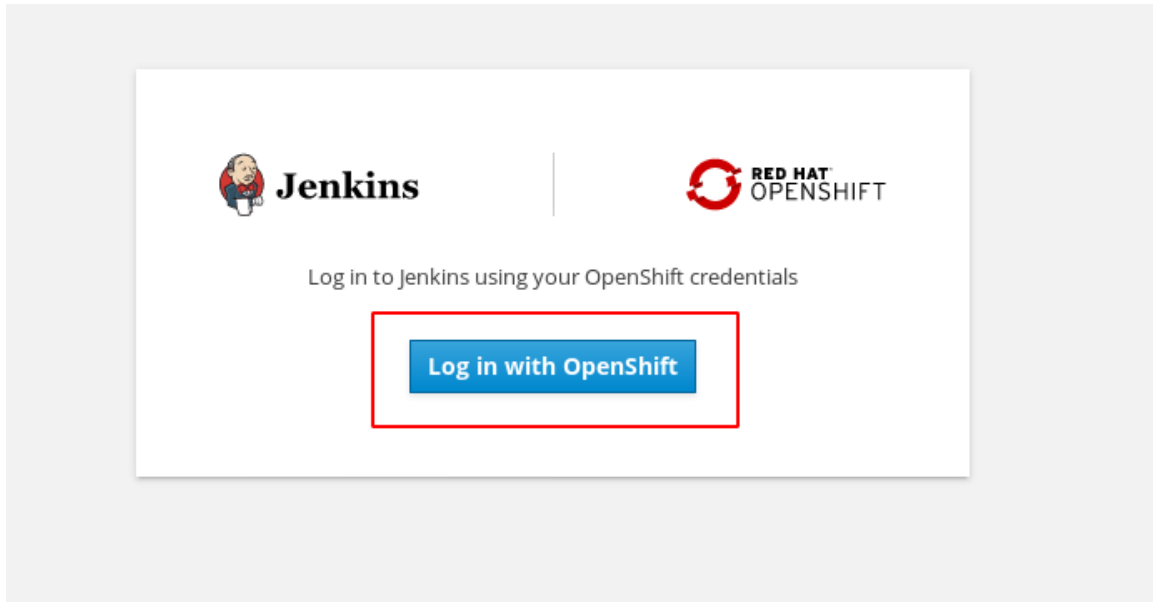
jenkins-jplp Service port: agent → Pod Port: 50000

jenkins Service port: web → Pod Port: 8080

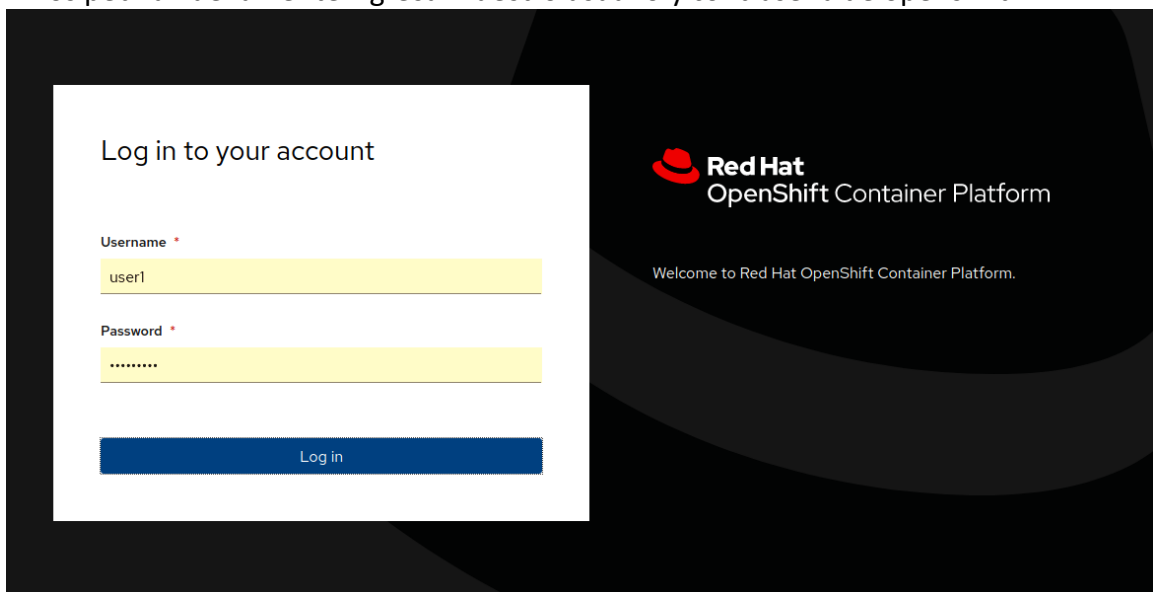
**Routes**

jenkins Location: <https://jenkins-user1-lab6.apps.cluster-qwe-e0e6.qwe-e0e6.example.openshift.com/gf>

6.7 Al abrir la ruta de Jenkins, debemos aceptar el certificado y darle click en “**Login with Openshift**”, para abrir el Jenkins.



Y nos pedirá nuevamente ingresar nuestro usuario y contraseña de openshift



Nos pedirá aceptar autorizaciones de acceso y daremos clic al botón “Allow selected permissions”

# Authorize Access

Service account `jenkins` in project `ci-cd-user1` is requesting permissions

## Requested permissions



### **user:info**

Read-only access to your user information (including username, identities, and group membership)



### **user:check-access**

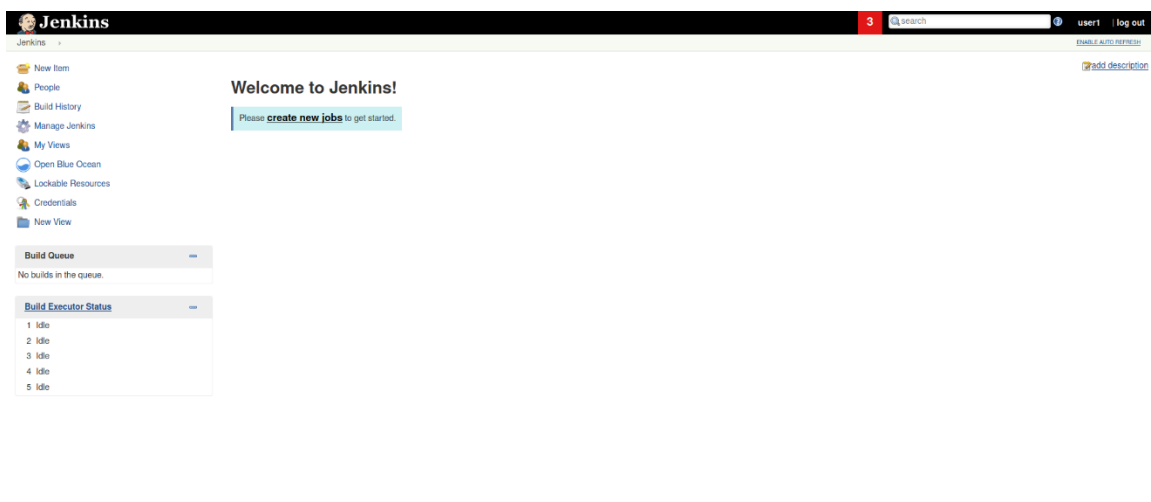
Read-only access to view your privileges (for example, "can I create builds?")

You will be redirected to <https://jenkins-ci-cd-user1.apps.medellin-9f63.open.redhat.com>

Allow selected permissions

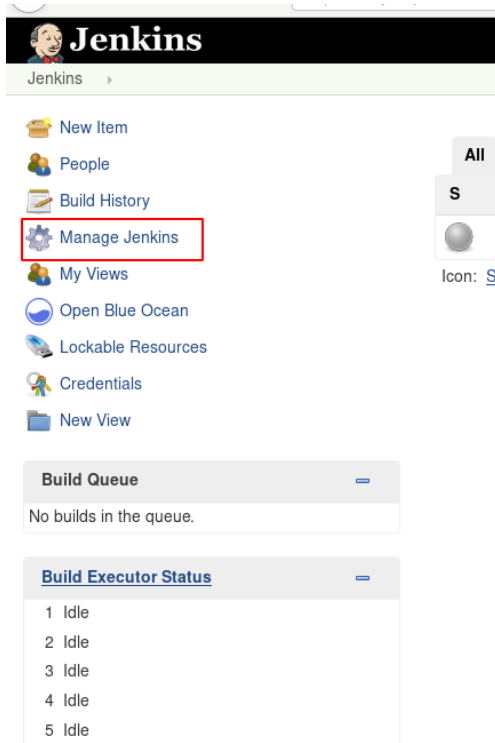
Deny

Luego de esto ya tendremos acceso a nuestro servidor Jenkins




6.8 Debemos activar el **maven** en el Jenkins esto para poder hacer despliegue de aplicaciones Java.

Para esto le damos **click** desde el Jenkins donde dice **"Manage Jenkins"**.



Le damos click en “**Global Tool Configuration**”

 **Jenkins**

Jenkins ▾

New Item

People

Build History

**Manage Jenkins**

My Views

Open Blue Ocean

Lockable Resources

Credentials

New View

**Build Queue**

No builds in the queue.

**Build Executor Status**

1 Idle

2 Idle

3 Idle

4 Idle

5 Idle

## Manage Jenkins

Jenkins root URL is empty but is required for the proper operation of many Jenkins features like the REST API. Please provide an accurate value in [Jenkins configuration](#).


Builds in Jenkins run as the virtual SYSTEM user with full permissions by default. This can be a security issue, it is recommended to install a plugin implementing build authentication, and to override the default permissions.


✗ No implementation of access control for builds is present. It is recommended that you install a plugin implementing build authentication.


You have not configured the CSRF issuer. This could be a security issue. For more information see the [Security section](#).


You can change the current configuration using the Security section [CSRF Protection](#).


Agent to master security subsystem is currently off. [Please read the documentation](#) and consider enabling it.

 **Configure System**  
Configure global settings and paths.


 **Configure Global Security**  
Secure Jenkins; define who is allowed to access/use the system.

 **Configure Credentials**  
Configure the credential providers and types

 **Global Tool Configuration**  
Configure tools, their locations and automatic installers.

 **Reload Configuration from Disk**  
Discard all the loaded data in memory and reload everything from file system. Useful when you have changed the configuration files on the disk.


Vamos a ver una imagen como la siguiente.

 Jenkins

Jenkins > Global Tool Configuration

Back to Dashboard

Manage Jenkins

 Global Tool Configuration

Maven Configuration

Default settings providerUse default maven settings

Default global settings providerUse default maven global settings

OpenShift Client Tools

OpenShift Client Tools installationsAdd OpenShift Client Tools

List of OpenShift Client Tools installations on this system

JDK

JDK installationsAdd JDK

List of JDK installations on this system

Git

Git installations

Git

NameDefault

Path to Git executablegit

☐ Install automatically

Add Git

Mercurial

Mercurial installationsAdd Mercurial

List of Mercurial installations on this system

Buscamos **Maven** y lo agregamos:

List of Ant installations on this system

Maven

Maven installations

Add Maven

List of Maven installations on this system

Agregamos el nombre “**maven-userx**”, como se muestra en la imagen y le damos **click** en “**Save**”.

**Maven**

Maven installations

Add Maven

Maven

Name

☒ Install automatically

Choose this option to let Jenkins install this tool for you on demand.

If you check this option, you'll then be asked to configure a series of "installer"s for this tool, where each installer is a script that will install the tool on the agent.

For a platform-independent tool (such as Ant), configuring multiple installers for a single tool does not allow you to run a different set up script depending on the agent environment.

Install from Apache

Version

Add Installer

Add Maven

List of Maven installations on this system

**Docker**

Docker installations

Add Docker

List of Docker installations on this system

Save Apply

Ahora que Jenkins está listo, configura un proyecto en OpenShift para mantener la aplicación que se creará utilizando la línea de comando.

6.9. Desde la línea de comandos de **oc**, y **logueados** a la misma, creamos el proyecto: **"user1-cicd-tareas"**

```
$ oc new-project user1-cicd-tareas
```

Now using project "user1-cicd-tareas" on server

```
"https://api.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com:6443".
```

You can add applications to this project with the 'new-app' command. For example, try:

```
oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git
```

```
to build a new example application in Ruby.[default@wetty-1-9nclg ~]$
```

6.10 Creamos la aplicación que vamos a utilizar con Jenkins con el siguiente comando:  
**"oc new-app jboss-eap71-openshift:1.3 https://github.com/redhat-gpte-devopsautomation/openshift-tasks"**

```
$ oc new-app jboss-eap71-openshift:1.3 https://github.com/redhat-gpte-devopsautomation/openshift-tasks
```

6.11 Procedemos a **exponer** el **servicio** para poder **accesar** a la aplicación fuera del cluster de Openshift.

```
$ oc expose svc openshift-tasks
```

6.12 **Apagar** todos los **triggers automáticos**.

Debido a que está creando esta aplicación utilizando un **pipeline de Jenkins**, es **Jenkins quien debe tener control total sobre lo que sucede en este proyecto**. Por defecto, la aplicación se vuelve a implementar cada vez que hay una nueva imagen disponible. Sin embargo, si reconstruye la imagen a través de Jenkins, es posible que desee ejecutar algunas pruebas antes de volver a implementar la aplicación.

```
$ oc set triggers dc openshift-tasks --manual
```

6.13 Otorgue a la cuenta de servicio los permisos correctos para editar objetos en este proyecto para permitir que Jenkins construya e implemente la aplicación.

Para esto ejecutamos el siguiente comando: verificar que estemos utilizando el proyecto correspondiente, se subrayan en negro para su atención.


**oc policy add-role-to-user edit system:serviceaccount:userx-lab6:jenkins -n userx-cicd-tareas**

```
$ oc policy add-role-to-user edit system:serviceaccount:userx-lab6:jenkins -n userx-cicd-tareas
```


6.14 Crear pipeline en Jenkins.


**Loguearse a Jenkins**, en el **navigator** en la izquierda, **click** en **"New Item"**.





 **Jenkins**


Jenkins ▾ ▸


 **New Item**


 People


 Build History


 Manage Jenkins

 My Views



 Open Blue Ocean

 Lockable Resources

 Credentials

 New View

All +

S	W	Name ↓
		<a href="#">OpenShift S...</a>

Icon: [S](#) [M](#) [L](#)

Build Queue ▾

No builds in the queue.

[Build Executor Status](#) ▾

1 Idle

2 Idle

3 Idle

4 Idle

6.15. En la página de “**New Item**”, lo llenamos de la siguiente forma.

**Enter an Item name: userx-tasks**

Seleccionar “**Pipeline**” para el tipo de job.

Y le damos **click** en “**Ok**”

**Enter an item name**

userx-tasks

» Required field

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build something other than software build.

**Pipeline**  
Orchestrates long-running activities that can span multiple organizing complex activities that do not easily fit in freestyle projects.

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations.

**Bitbucket Team/Project**  
Scans a Bitbucket Cloud Team (or Bitbucket Server Repository) for new builds.

**Folder**  
Creates a container that stores nested items in it. Useful for organizing namespaces, so you can have multiple things of the same type.

**GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories.

OK

6.16. En la carpeta donde están los documentos del workshop, verificamos el archivo “**pipeline-openshift-tasks.yaml**”, lo abrimos y modificamos el valor que vemos en color rojo en el texto a continuación.

```
node {  
  stage('Build Tasks') {  
    openshift.withCluster() {  
      openshift.withProject("userx-cicd-tareas") {
```

```
    openshift.selector("bc", "openshift-tasks").startBuild("--wait=true")
  }
}
stage('Tag Image') {
  openshift.withCluster() {
    openshift.withProject("userx-cicd-tareas") {
      openshift.tag("openshift-tasks:latest", "openshift-tasks:${BUILD_NUMBER}")
    }
  }
}
stage('Deploy new image') {
  openshift.withCluster() {
    openshift.withProject("userx-cicd-tareas") {
      openshift.selector("dc", "openshift-tasks").rollout().latest();
    }
  }
}
}
```

Asegúrese de que el project/namespace “**userx-cicd-tareas**” apunta al nombre real del proyecto en la opción “**openshift.withProject**” con el que nosotros creamos en pasos anteriores, debe coincidir con nuestro usuario.

Este texto lo debemos pegar en el campo de **Pipeline**, después de haberlo modificado, este campo está en la última parte de la configuración del pipeline, para ser específicos en la **Definition Pipeline script**.

Y hay que darle **click** en “**Save**” a este **pipeline job**.

## Pipeline

Definition

Pipeline script

Script

```
1 node {
2   stage('Build Tasks') {
3     openshift.withCluster() {
4       openshift.withProject("user1-cicd-tareas") {
5         openshift.selector("bc", "openshift-tasks").startBuild("--wait=true")
6       }
7     }
8   }
9   stage('Tag Image') {
10    openshift.withCluster() {
11      openshift.withProject("user1-cicd-tareas") {
12        openshift.tag("openshift-tasks:latest", "openshift-tasks:${BUILD_NUMBER}")
13      }
14    }
15  }
16  stage('Deploy new image') {
17  }
```

☒ Use Groovy Sandbox

[Pipeline Syntax](#)

Save

Apply

6.17. En la página de Jenkins, **click** en “**Build Now**” .

# Jenkins

Jenkins > Tasks >

[Back to Dashboard](#)
[Status](#)
[Changes](#)

[Build Now](#)

[Delete Pipeline](#)
[Configure](#)
[Full Stage View](#)
[Open Blue Ocean](#)
[Rename](#)
[Pipeline Syntax](#)

[Recent Changes](#)

## Stage View

No data available. This Pipeline

## Build History

[trend](#)

[RSS for all](#)
[RSS for failures](#)

## Pipeline Tasks

Nos esperamos a que se realice el **build** de forma correcta, el mismo **dura alrededor de 3 minutos**.

# Jenkins

Jenkins > user1-tasks >

[Back to Dashboard](#)
[Status](#)
[Changes](#)

[Build Now](#)

[Delete Pipeline](#)
[Configure](#)
[Full Stage View](#)
[Open Blue Ocean](#)
[Rename](#)
[Pipeline Syntax](#)

[Recent Changes](#)

## Stage View

Average stage times:  
(Average full run time: ~2min -28s)

Build Tasks	Tag Image	Deploy new image
2min 20s	1s	988ms

## Build History

[trend](#)

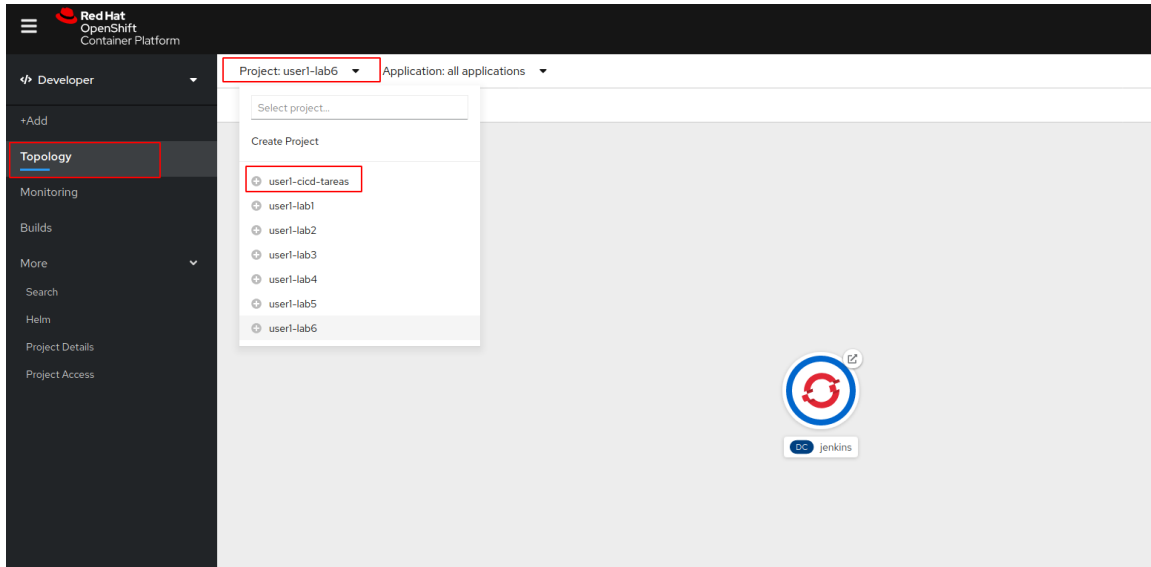
[Atom feed for all](#)
[Atom feed for failures](#)

## Pipeline user1-tasks

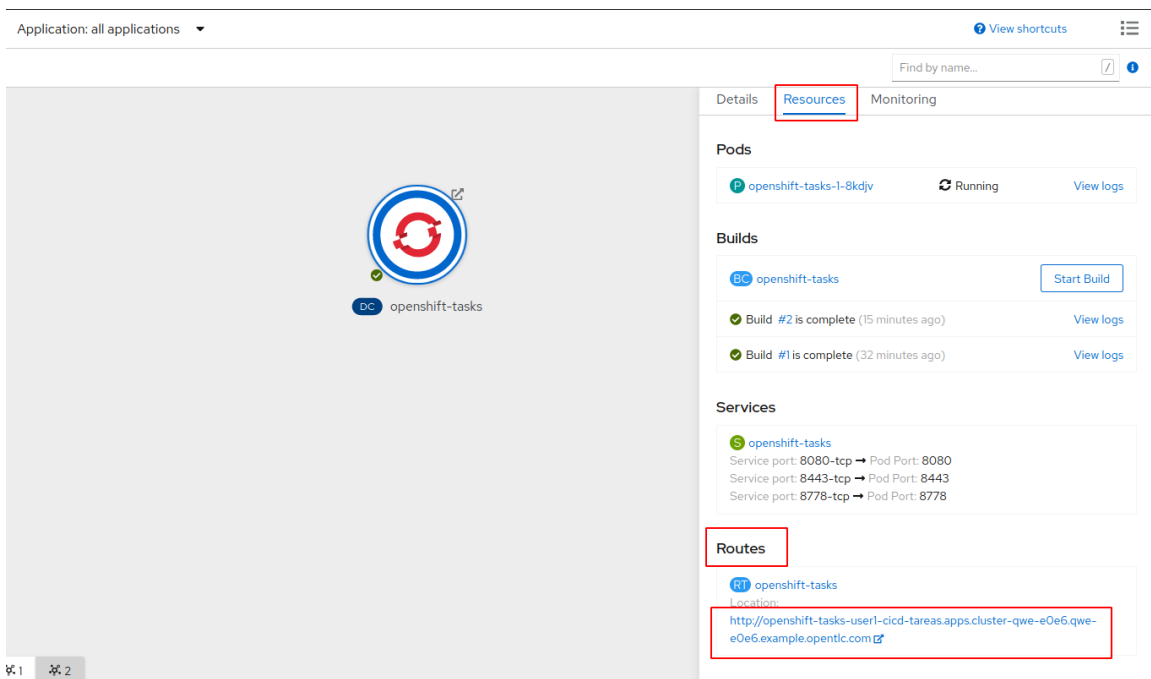
## Permalinks

- Last build (#1), 1 min 12 sec ago

6.18. Ya con el **build** realizado de forma correcta, ingresamos a la consola web de Openshift, abrimos el **proyecto donde hicimos el deploy**, en este caso: “**userx-cicd -tareas**”. Para ellos desde la vista Topology desplegamos la lista de proyectos y buscamos el proyecto llamado “**userx-cicd -tareas**”



Para ingresar al URL, dentro de la parte de **Resources**, hay una parte de rutas, en esa debe estar el URL de acceso a la aplicación. Para ello damos clic en nuestro Deploy Config y nos vamos a la sección de rutas.



6.19. Abrimos la página, desde la ruta que se muestra en el proyecto “**user1-cicd-tareas**”, en este caso sería: <http://openshift-tasks-user1-cicd-tareas.apps.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com/>

