



## 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

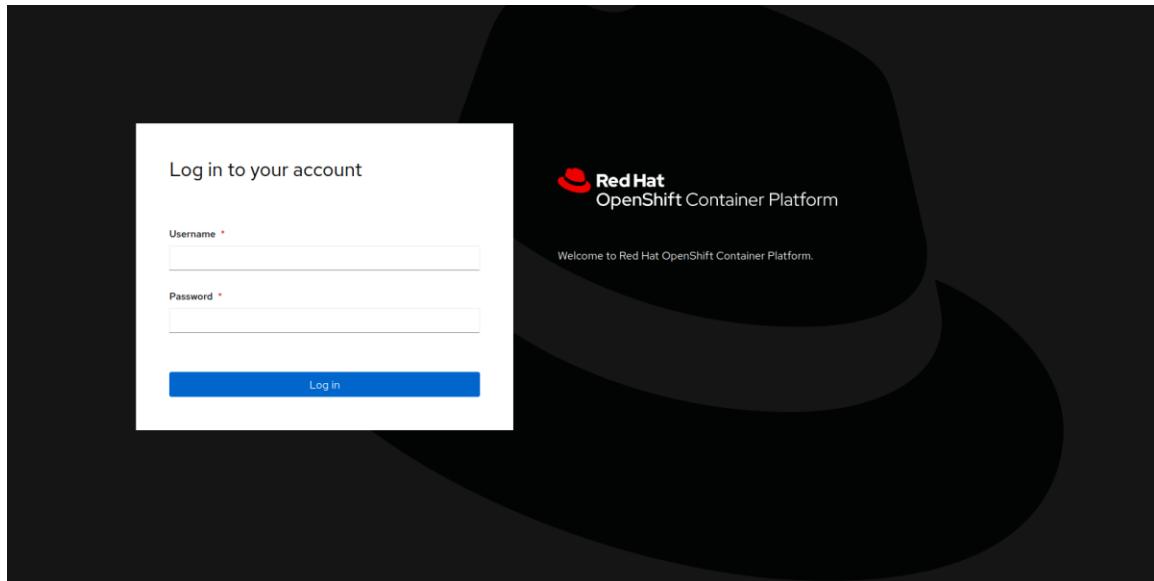
- a) Se le brindará una URL para accesar al ambiente del curso.

**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.

The screenshot shows the OpenShift web interface. At the top right, there is a user dropdown menu with options like '+', '?', 'user', 'Copy Login Command', and 'Log out'. The 'Copy Login Command' option is highlighted with a red box. Below the header, the main content area displays a 'Welcome to OpenShift' message, instructions for creating a project, and a 'Create Project' button. A search bar at the bottom right says 'Filter by name or display name...' with a magnifying glass icon.

Indicamos nuevamente nuestro usuario y contraseña y luego Login

The screenshot shows the Red Hat OpenShift Container Platform login screen. It features a dark background with the Red Hat logo and the text 'Red Hat OpenShift Container Platform'. On the left, there is a white login form titled 'Log in to your account'. It has fields for 'Username \*' (containing 'user') and 'Password \*' (containing '.....'). A blue 'Log in' button is at the bottom. The URL in the browser address bar is 'https://oauth-openshift.apps.cluster-central-810b.central-810b.example.opentc.com/oauth/token/display?code=3a5iugOs63k6CeEuWj7YzMHLszA\_trRwvFf5uaPuaLA&state='.

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

The screenshot shows a Mozilla Firefox browser window with a single tab open to the URL 'https://oauth-openshift.apps.cluster-central-810b.central-810b.example.opentc.com/oauth/token/display?code=3a5iugOs63k6CeEuWj7YzMHLszA\_trRwvFf5uaPuaLA&state='. The page content is a simple confirmation message: 'Display Token' followed by a large red 'OK' button. The 'Display Token' button is highlighted with a red box.

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.



Your API token is  
Sq-fbbKVLcwbZ1Fs5KoZ-2b06BpmquJm2j7axhZm5Z0

**Log in with this token**

```
oc login --token=5q-fbbKVLcwbZ1Fs5KoZ-2b06BpmquJm2j7axhZm5Z0 --server=https://api.cluster-central-810b.central-810b.example.opentlc.com:6443
```

**Use this token directly against the API**

```
curl -H "Authorization: Bearer Sq-fbbKVLcwbZ1Fs5KoZ-2b06BpmquJm2j7axhZm5Z0" "https://api.cluster-central-810b.central-810b.example.opentlc.com:6443/apis/user.openshift.io/v1/users/~"
```

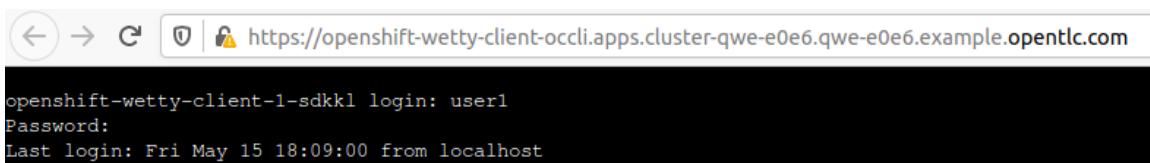
[Request another token](#)

## 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:

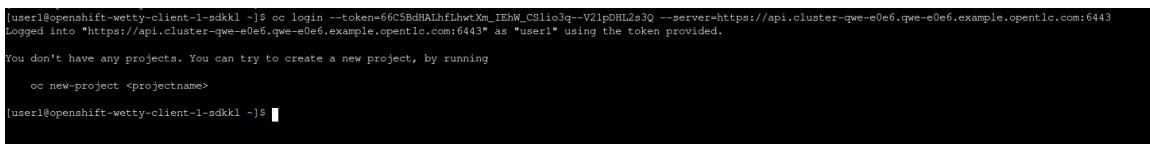
- Usuario: userx
- Password: Passwordx

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



```
openshift-wetty-client-1-sdkkl login: user1
Password:
Last login: Fri May 15 18:09:00 from localhost
```

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.



```
[user1@openshift-wetty-client-1-sdkkl ~]$ oc login --token=66C5BdHALhfIhwXm_IeHw_CSlio3q--V2IpDHLzs3Q --server=https://api.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com:6443
Logged into "https://api.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com:6443" as "user1" using the token provided.

You don't have any projects. You can try to create a new project, by running
  oc new-project <projectname>
[user1@openshift-wetty-client-1-sdkkl ~]$
```

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 linea de commandos 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

Name	Display Name	Status	Requester	Created
userx-lab1	No display name	Active	user1	7 minutes ago

## 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 ello 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:

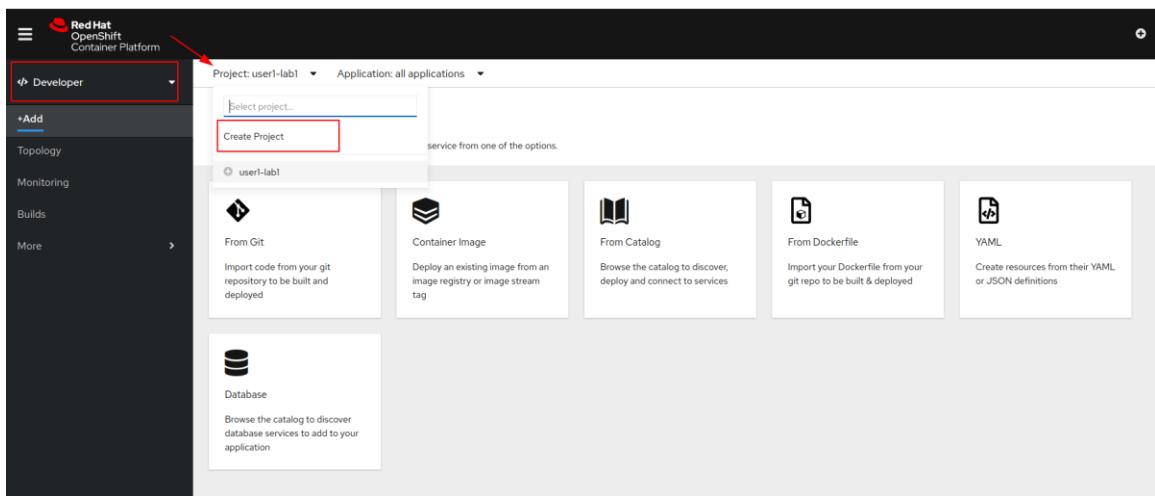
```
$ oc get route
NAME HOST/PORT PATH SERVICES PORT
TERMINATION WILDCARD
wildfly wildfly-user1-lab1.apps.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com
wildfly 8080-tcp None
```

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.

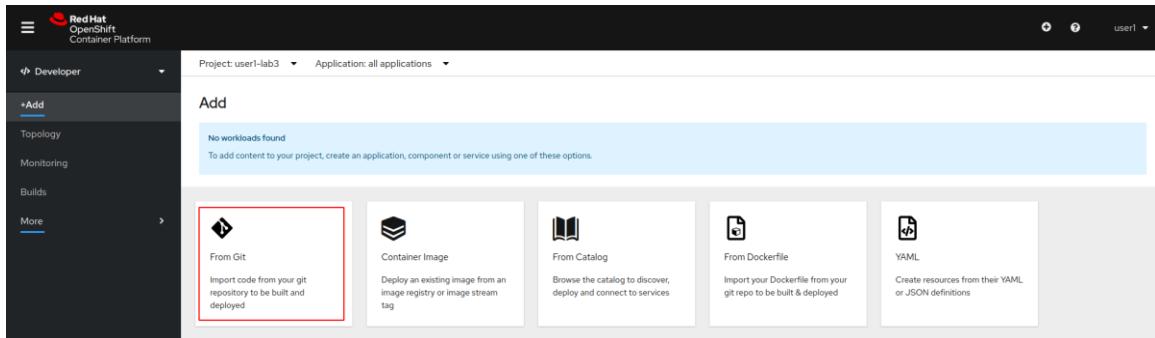


3.2 Le indicamos el nombre userx-lab3

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

This is a 'Create Project' dialog box. It contains three input fields: 'Name \*' with the value 'user1-lab3', 'Display Name' with the value 'user1-lab3', and 'Description' which is empty. At the bottom right, there are two buttons: 'Cancel' and 'Create', with a red arrow pointing to the 'Create' button.

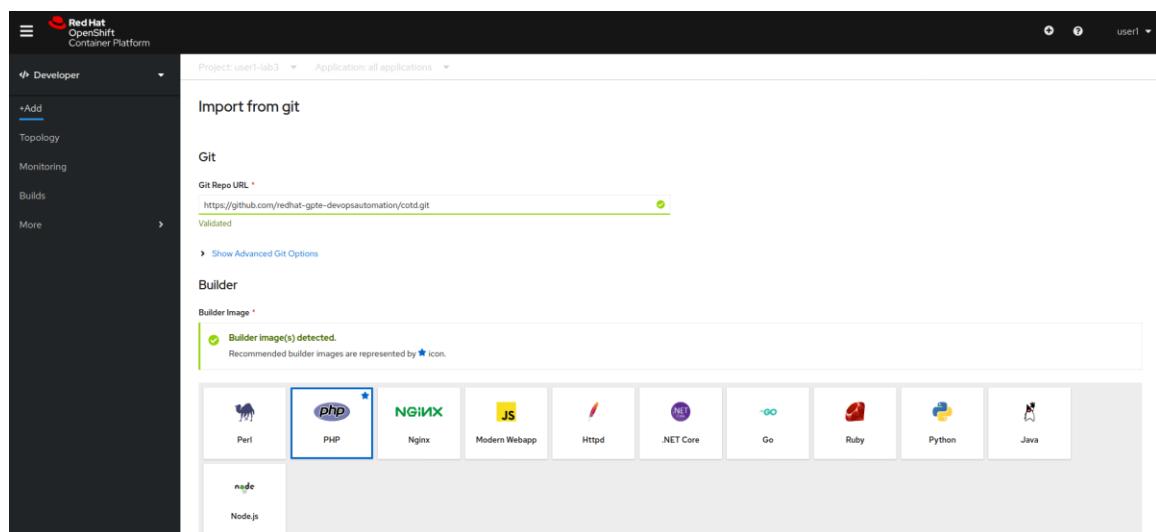
### 3.3 Procedemos a irnos a la opción **From Git** para crear una aplicación desde un repositorio



### 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:



The screenshot shows the Red Hat OpenShift Container Platform interface. In the top left, there's a navigation bar with a menu icon, the Red Hat logo, and the text "Red Hat OpenShift Container Platform". Below this, a header bar displays "Project: user1-lab3" and "Application: all applications". A dropdown menu for "Builder Image Version" is open, showing "S2I 7.3" as the selected option. The main content area is titled "Topology" and shows a "PHP 7.3" builder image card. The card includes a brief description: "Build and run PHP 7.3 applications on RHEL 7. For more information about using this builder image, including OpenShift considerations, see <https://github.com/sclorg/s2i-php-container/blob/master/7.3/README.md>". It also mentions a sample repository: <https://github.com/sclorg/cakephp-ex.git>. The configuration section is titled "General" and contains fields for "Application Name" (set to "app-user1-lab3") and "Name" (also set to "app-user1-lab3"). Below this is a "Resources" section where "Deployment" is selected as the resource type to generate, with a description of what it does. The entire form is contained within a dark-themed card.

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

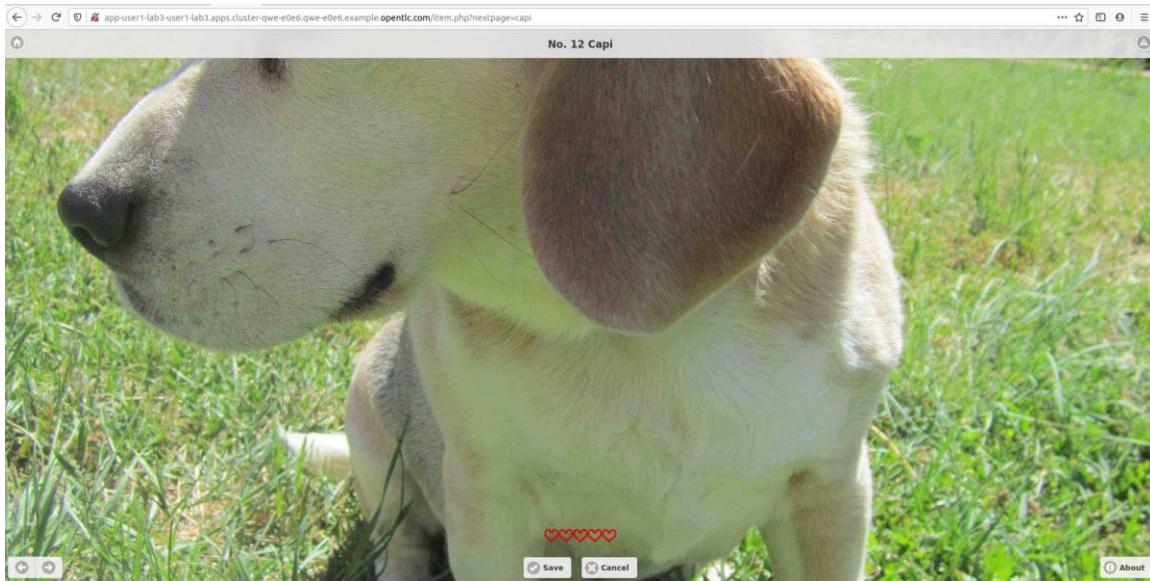
This screenshot continues the application creation process from the previous one. The interface is identical, showing the "Topology" view with the "PHP 7.3" builder image. The "General" and "Resources" sections are the same. A new section titled "Advanced Options" has been added at the bottom. It contains a checked checkbox for "Create a route to the application" with the note "Exposes your application at a public URL". Below this, a note says "Click on the names to access advanced options for [Routing](#), [Build Configuration](#), [Deployment](#), [Scaling](#), [Resource Limits](#) and [Labels](#)". A red arrow points to the "Create" button at the bottom of the form. The "Cancel" button is also visible.

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 accesar a nuestra aplicación.

The screenshot shows the Red Hat OpenShift web interface. On the left, there's a sidebar with 'Developer' selected, showing options like '+Add', 'Topology' (which is highlighted with a red box), 'Monitoring', 'Builds', and 'More'. The main area shows a project named 'user1-lab3' and an application named 'all applications'. A 'Display' dropdown is set to '5'. On the right, there's a detailed view of a pod named 'app-user1-lab3'. The 'Resources' tab is highlighted with a red box. The 'Routes' section shows a single route entry: 'http://app-user1-lab3-user1-lab3.apps.cluster-qwe-e0e6.qwe-e0e6.example.opentic.com'. An arrow points from the text above to this route entry.

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 cuidados) 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 accesar 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       config,image(cotd:latest)

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

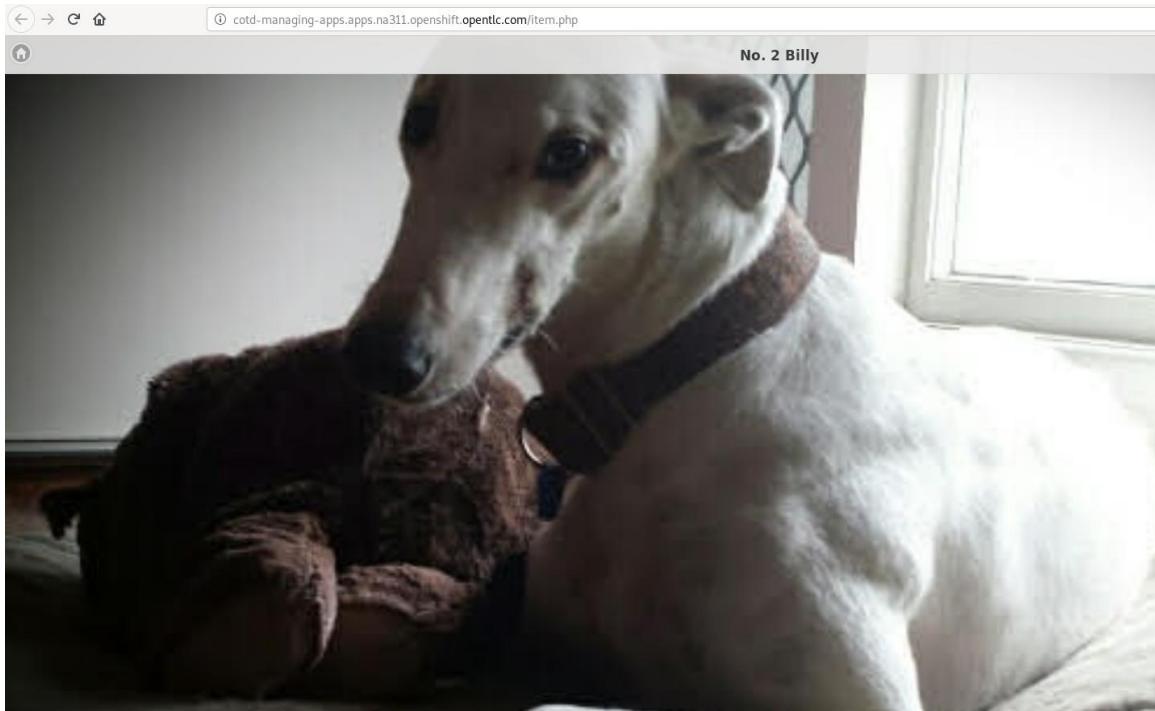
NAME      TYPE      FROM      STATUS    STARTED      DURATION
build.build.openshift.io/cotd-1  Source    Git@b53da24  Complete  About an hour ago
23s

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
TERMINATION WILDCARD
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
```

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 ver 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 enruteamiento 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:

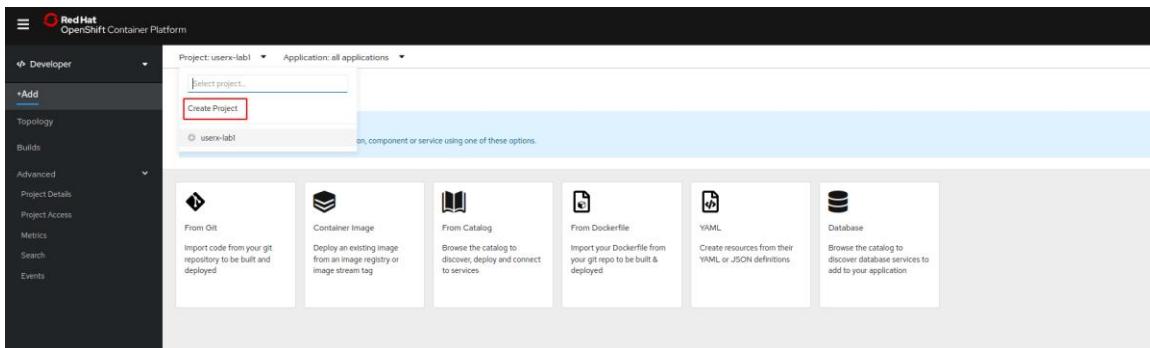
Debe usar **curl** desde la línea de comandos porque todos los navegadores modernos usan cookies para identificar su sesión actual. Y cada vez que actualiza el navegador, se lo envía al mismo servicio al que se lo envió antes, lo cual tiene sentido, porque en el mundo real cuando realiza **pruebas A / 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. Desplegar aplicación Node.JS con conexión a una base de datos MongoDB, desde Openshift.

5.1 Creamos un 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.



- Name: **userx-lab5**
- Display Name: **userx-lab5**
- Description: **Aplicación que consiste en un Node.JS front end y una DB MongoDB backend.**

A screenshot of the "Create Project" dialog. It has a title "Create Project". There are three input fields: "Name \*" with value "user1-lab5", "Display Name" with value "user1-lab5", and "Description" with value "Aplicación que consiste en un Node.JS front end y una DB MongoDB backend.". At the bottom are two buttons: "Cancel" and "Create" (which is blue).

Y damos clic en el botón “Create”

## 5.2 Seleccionamos la opción From Catalog

The screenshot shows the Red Hat OpenShift Container Platform interface. In the top navigation bar, 'Project: userx-mi-primer-app' and 'Application: all applications' are selected. On the left, a sidebar menu is open under 'Developer' with 'Advanced' expanded, showing options like 'Project Details', 'Project Access', 'Metrics', 'Search', and 'Events'. A red arrow points to the '+Add' button. The main content area is titled 'Add' and displays five options: 'Import Git', 'Container Image', 'From Catalog' (which is highlighted with a red box), 'From Dockerfile', and 'YAML'. Below each option is a brief description.

## 5.3 Seleccionamos la opción nodejs+mongoDB para ello filtramos por la palabra "nodejs"

The screenshot shows the 'Developer Catalog' interface. In the top navigation bar, 'Project: user1-lab5' is selected. The left sidebar lists categories such as 'All Items', 'Languages', 'Databases', 'Middleware', 'CI/CD', 'Other', and 'Type' (with checkboxes for 'Operator Backed (0)', 'Helm Charts (1)', 'Builder Image (1)', 'Template (3)', and 'Service Class (0)'). A search bar at the top has 'nodejs' typed into it, which is highlighted with a red box. The main content area shows several catalog items: 'node' (Builder Image), 'node' (Template), 'node' (Template) (highlighted with a red box), and 'HELM' (Helm Charts). Each item has a brief description and a link to more information.

Y le damos al botón “Instantiate Template”

The screenshot shows the details page for the 'Node.js + MongoDB (Ephemeral)' template. At the top, the title 'Node.js + MongoDB (Ephemeral)' is displayed, along with the provider 'Provided by Red Hat, Inc.' and a large blue 'Instantiate Template' button, which is highlighted with a red box. Below this, there are sections for 'Provider' (Red Hat, Inc.), 'Support' (with a 'Get support' link), 'Created At' (May 14, 8:24 am), and 'Documentation' (link to <https://github.com/sclorg/nodejs-ex>). A warning message states: 'WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing.'

5.4 Para efectos del laboratorio usarmos los repositorios de aplicaciones y base de datos provistos por Red Hat para facilitar su configuración.

Cambiamos los siguientes valores:

- Name: appdb-userX
- Git Repository URL: <https://github.com/sclorg/nodejs-ex.git>

Quedando la configuración de la siguiente manera:

### Instantiate Template

Namespace \*

Name \*

The name assigned to all of the frontend objects defined in this template.

Namespace \*

The OpenShift Namespace where the ImageStream resides.

Version of NodeJS Image \*

Version of NodeJS image to be used (6, 8, or latest).

Version of MongoDB Image \*

Version of MongoDB image to be used (3.6 or latest).

**Memory Limit \***

Maximum amount of memory the Node.js container can use.

**Memory Limit (MongoDB) \***

Maximum amount of memory the MongoDB container can use.

**Volume Capacity \***

Volume space available for data, e.g. 512Mi, 2Gi

**Git Repository URL \***

The URL of the repository with your application source code.

**Git Reference**

Set this to a branch name, tag or other ref of your repository if you are not using the default branch.

**Context Directory**

Set this to the relative path to your project if it is not in the root of your repository.

**Application Hostname**

The exposed hostname that will route to the Node.js service, if left blank a value will be defaulted.

**GitHub Webhook Secret**

Github trigger secret. A difficult to guess string encoded as part of the webhook URL. Not encrypted.

**Generic Webhook Secret**

A secret string used to configure the Generic webhook.

**Database Service Name \***

**MongoDB Username**

Username for MongoDB user that will be used for accessing the database.

**MongoDB Password**

Password for the MongoDB user.

**Database Name \***

**Database Administrator Password**

Password for the database admin user.

**Custom NPM Mirror URL**

The custom NPM mirror URL

**Create** **Cancel**

Una vez creado nuestra configuración para nuestro nodejs y mongodb podremos ver desde la vista Topology, ambos componentes y sus recursos.

En el caso de MongoDB, podremos ver su pod, su deploy config y los valores de configuración. Para ello seleccionamos el Deploy config llamado “mongodb”

The screenshot shows the Rancher interface for managing Kubernetes resources. In the top navigation bar, there is a search bar labeled "Find by name..." and a "Actions" dropdown. Below the navigation, there are two main sections: "Topology" and "Deployments".

In the "Topology" section, there are three icons representing different components: a blue circle with a white "DC" and "mongodb" label, a green circle with a white "DC" and "appdb-user1" label, and a blue circle with a white "DC" and "nodejs" label. The first icon is circled in red.

In the "Deployments" section, there is a card for the "mongodb" deployment. The card has tabs for "Details", "Resources", and "Monitoring", with "Details" being the active tab and highlighted with a red border. The card displays the following information:

- Pods:** 1 pod
- Name:** mongodb
- Namespace:** user1-lab5
- Labels:** app=nodejs-mongodb-example, template=nodejs-mongodb-example, template.openshift.io/generation=680a6611-cd4...
- Latest Version:** 1
- Message:** config change
- Update Strategy:** Recreate
- Min Ready Seconds:** Not Configured
- Pod Selector:** name=mongodb
- Node Selector:** No selector
- Triggers:** ImageChange, ConfigChange

Para validar los detalles y valores como usuario y password de la base de datos podemos dirigirnos a la vista **Administrator > Workloads > Deployments Configs > Seleccionar el deploy config llamado mongodb y nos dirigimos a la pestaña "Environment"**

The screenshot shows the Red Hat OpenShift Container Platform interface. The left sidebar is titled 'Administrator' and includes sections for Home, Projects, Search, Explore, Operators, Workloads, Deployments, Deployment Configs, Secrets, Config Maps, Cron Jobs, Jobs, Daemon Sets, and Replica Sets. The 'Deployment Configs' section is currently selected. The main content area shows 'Deployment Configs > Deployment Config Details' for 'mongodb'. The 'Environment' tab is highlighted with a red box. Below it, under 'Container: mongodb', there is a table for 'Single values (env)'. It lists four variables: MONGODB\_USER, MONGODB\_PASSWORD, MONGODB\_DATABASE, and MONGODB\_ADMIN\_PASSWORD. Each variable has a dropdown menu showing its value. A red box also highlights this table.

Ahora en el caso de nuestra aplicación nodejs la cual está configurada por medio del repositorio <https://github.com/sclorg/nodejs-ex> para ver su configuración, pod, rutas nos dirigimos a la vista **Developer > Topology** y seleccionamos del deployment config **appdb-userx**

The screenshot shows the Red Hat OpenShift Container Platform interface. The top navigation bar includes 'Project: user1-lab5', 'Application: all applications', 'View shortcuts', and a search bar. The main content area shows the 'Topology' view for the 'appdb-userx' deployment config. On the left, there are icons for 'mongodb' and 'appdb-user1', with 'appdb-user1' circled in red. To the right, there is a detailed view for 'appdb-user1'. It shows 'Details', 'Resources', and 'Monitoring' tabs, with 'Resources' selected. Under 'Pods', it shows one pod named 'appdb-user1-1-8xdfc' in 'Running' status. Under 'Builds', it shows a build named 'appdb-user1' that is complete. Under 'Services', it shows a service named 'appdb-user1' with a service port of 'web' mapped to a pod port of '8080'. Under 'Routes', it shows a single route named 'appdb-user1' with a location of 'http://appdb-user1-lab5.apps.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com'. A red box highlights the 'Routes' section.

Ver el apartado rutas y damos clic en la ruta expuesta por la aplicación

Welcome to your Node.js application on OpenShift

**How to use this example application**  
For instructions on how to use this application with OpenShift, start by reading the [Developer Guide](#).

**Deploying code changes**  
The source code for this application is available to be forked from the [OpenShift GitHub repository](#). You can configure a webhook in your repository to make OpenShift automatically start a build whenever you push your code:

1. From the Web Console homepage, navigate to your project
2. Click on Browse > Builds
3. Click the link with your BuildConfig name
4. Click the Configuration tab
5. Click the "Copy to clipboard" icon to the right of the "GitHub webhook URL" field
6. Navigate to your repository on GitHub and click on repository settings > webhooks > Add webhook
7. Paste your webhook URL provided by OpenShift in the "Payload URL" field
8. Change the "Content type" to "application/json"
9. Leave the defaults for the remaining fields — that's it!

After you save your webhook, if you refresh your settings page you can see the status of the ping that Github sent to OpenShift to verify it can reach the server.

Note: adding a webhook requires your OpenShift server to be reachable from GitHub.

**Working in your local Git repository**  
If you forked the application from the OpenShift GitHub example, you'll need to manually clone the repository to your local system. Copy the application's source code Git URL and then run:

```
$ git clone <git_url> <directory_to_create>
# Within your project directory
```

**Managing your application**  
Documentation on how to manage your application from the Web Console or Command Line is available at the [Developer Guide](#).

**Web Console**  
You can use the Web Console to view the state of your application components and launch new builds.

**Command Line**  
With the [OpenShift command line interface](#) (CLI), you can create applications and manage projects from a terminal.

**Development Resources**

- [OpenShift Documentation](#)
- [Openshift Origin GitHub](#)
- [Source To Image GitHub](#)
- [Getting Started with Node.js on OpenShift](#)
- [Stack Overflow questions for OpenShift](#)
- [Git documentation](#)

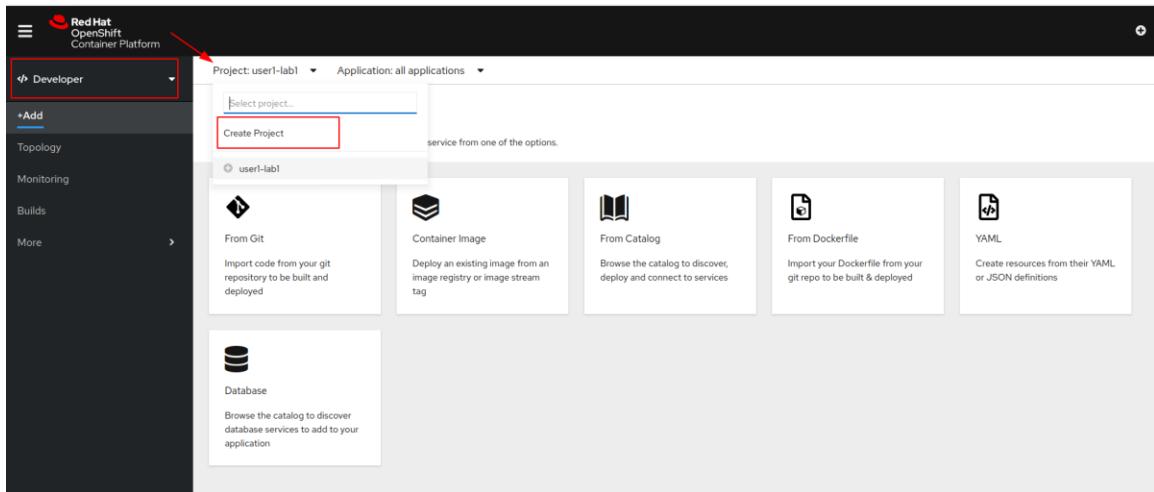
**Request information**  
Page view count:  

**DB Connection Info:**  
Type: MongoDB  
URL: mongodb://172.30.233.44:27017/sampledbs

Para verificar que la aplicación se unió a la base de datos, ver el label “**DB Connection Info**”, esto con detalles sobre la conexión a la base de datos. También se puede ver que cada vez que se refresca la página, el “**Page View Count**” se incrementa.

## 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**

The dialog box has a title 'Create Project'. It contains three input fields: 'Name \*' with value 'user1-lab6', 'Display Name' with value 'user1-lab6', and 'Description' with value 'Servidor Jenkins'. At the bottom are 'Cancel' and 'Create' buttons.

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

The screenshot shows the Developer Catalog interface for a project named "user1-lab6". The left sidebar has sections for All Items, Languages, Databases, Middleware, CI/CD (which is selected and highlighted with a red box), and Other. Under CI/CD, there are filters for Type (Operator Backed, Helm Charts, Builder Image, Template, Service Class) and a search bar. A "Group By: None" dropdown is also present. The main area displays four items, each with a Jenkins icon and a "Template" button. The third item, "Jenkins (Ephemeral)", is highlighted with a red dashed box. The description for this template states: "Jenkins service, without persistent storage. WARNING: Any data stored will be lost upon...".

### 6.4 Lo seleccionamos y damos click en “Instantiate Template”.

The screenshot shows the details page for the "Jenkins (Ephemeral)" template. At the top, there is a header with a user icon and the name "user1". Below the header, the template's name "Jenkins (Ephemeral)" and provider "Red Hat, Inc." are displayed. A large blue "Instantiate Template" button is highlighted with a red box. To the right of the button, there are sections for "Provider", "Description", "Support", "Documentation", and "Created At". The "Provider" section shows "Red Hat, Inc.". The "Description" section includes a warning: "WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing.". The "Documentation" section provides a link: "[https://docs.okd.io/latest/using\\_images/other\\_images/jenkins.html](https://docs.okd.io/latest/using_images/other_images/jenkins.html)". The "Created At" section shows the date and time: "May 14, 8:24 am".

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

### Instantiate Template

Namespace \*

Jenkins Service Name

The name of the OpenShift Service exposed for the Jenkins container.

Jenkins JNLP Service Name

The name of the service used for master/slave communication.

Enable OAuth in Jenkins

Whether to enable OAuth OpenShift integration. If false, the static account 'admin' will be initialized with the password 'password'.

Memory Limit

Maximum amount of memory the container can use.

Jenkins ImageStream Namespace

The OpenShift Namespace where the Jenkins ImageStream resides.

Disable memory intensive administrative monitors

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.

**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.

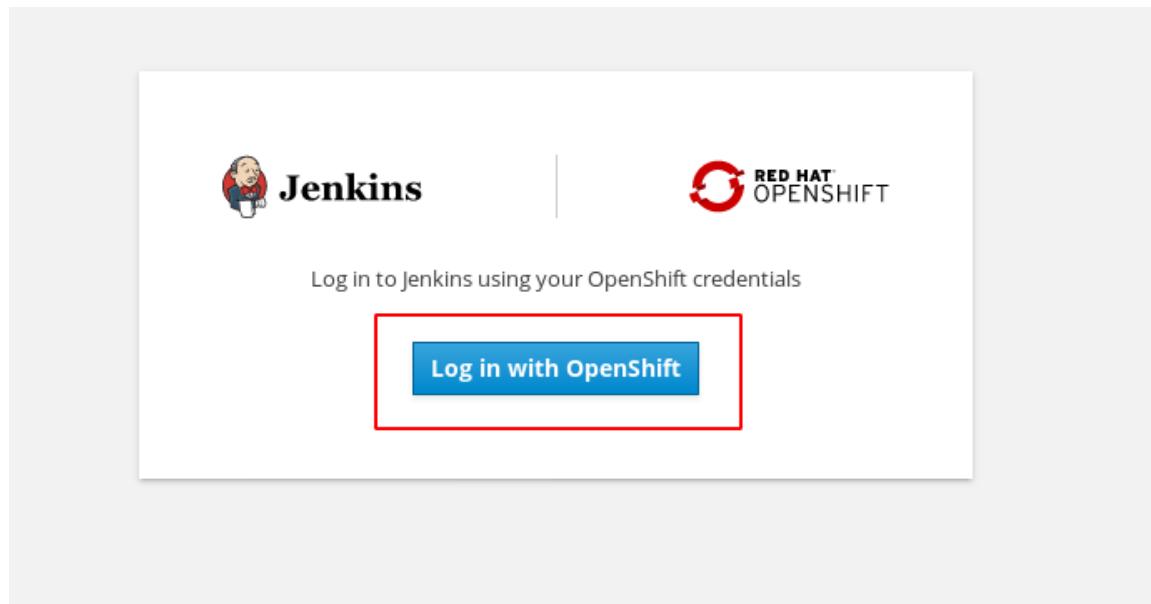
**Create** **Cancel**

Y damos clic al botón “Create”

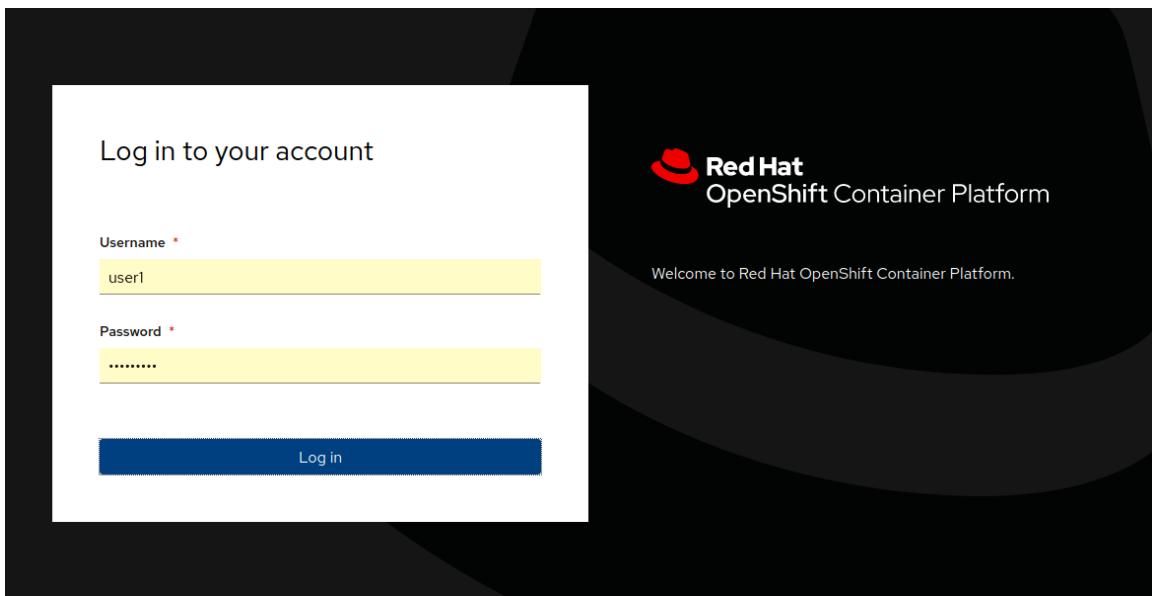
6.6 Una vez creado nuestro pod de Jenkins, desde la vista Topology podemos ver los parámetros de su configuración, la ruta de acceso y demás. Verificamos el estado del pod, damos unos minutos para que se complete la configuración. Para ello damos clic en la ruta mostrada en el apartado “Routes” para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

The screenshot shows the OpenShift Application Catalog interface. At the top, it says "Project: user1-lab6" and "Application: all applications". Below that is a search bar and a "Display" dropdown set to "5". On the right, there's a "View shortcuts" button and a "Find by name..." input field. The main area shows a pod named "jenkins" with a blue Jenkins icon. A red circle highlights this icon. To the right of the pod are tabs for "Details", "Resources" (which is selected), and "Monitoring". Under "Resources", there's a "Pods" section showing "jenkins-1-8gprb" as "Running" with a "View logs" button. Below that is a "Builds" section stating "No Build Configs found for this resource.". Under "Services", two services are listed: "jenkins-jnlp" (Service port: agent → Pod Port: 50000) and "jenkins" (Service port: web → Pod Port: 8080). A red arrow points from the text "https://jenkins-user1-lab6.apps.cluster-qwe-e0e6.qwe-e0e6.example.openshift.com" in the "Routes" section to the "jenkins" service entry. The "Routes" section also contains the URL "https://jenkins-user1-lab6.apps.cluster-qwe-e0e6.qwe-e0e6.example.openshift.com" with a "View" link. At the bottom left are standard browser navigation buttons.

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**

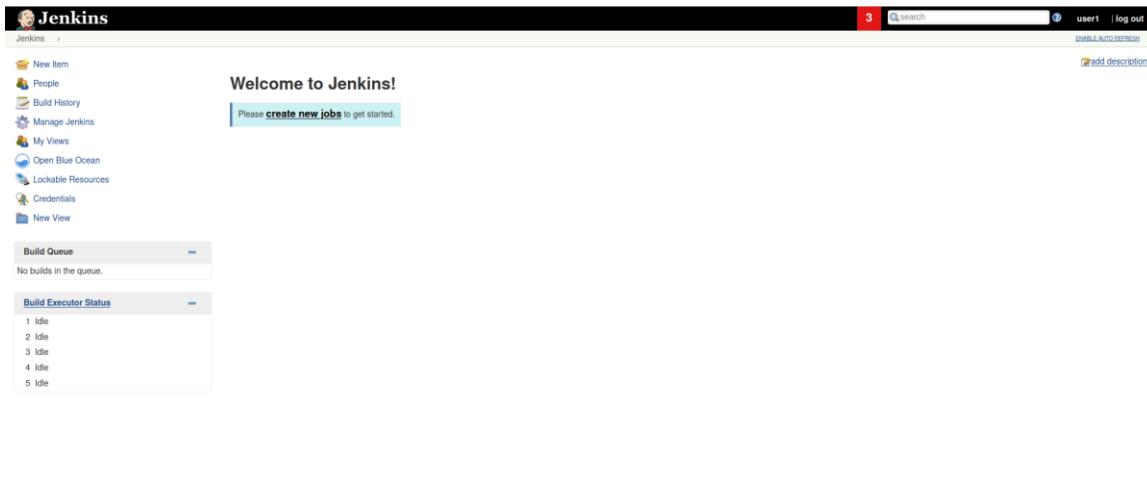
Read-only access to your user information (including username, identities, and group membership)

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

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

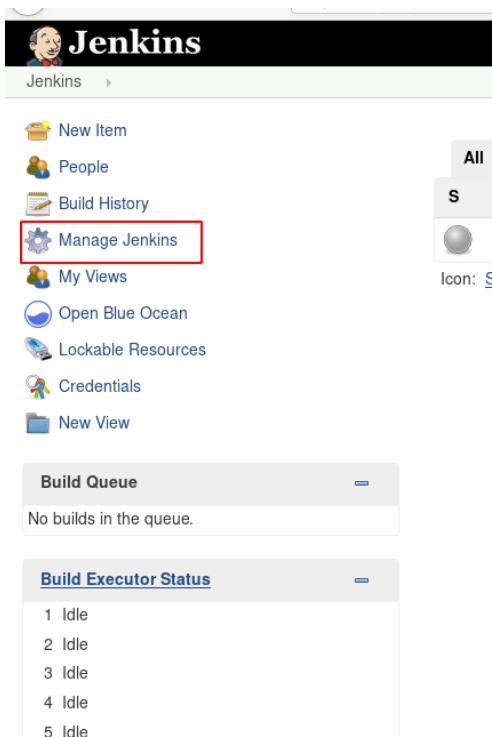
You will be redirected to <https://jenkins-ci-cd-user1.apps.medellin-9f63.open.redhat.com>

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

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...  
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 case, it is recommended to install a plugin implementing build authentication, and to override the default build user.

✗ No implementation of access control for builds is present. It is recommended that you...

You have not configured the CSRF issuer. This could be a security issue. For more information...  
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...

**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...

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 provider [Use default maven settings](#)

Default global settings provider [Use default maven global settings](#)

### OpenShift Client Tools

OpenShift Client Tools installations [Add OpenShift Client Tools](#)  
List of OpenShift Client Tools installations on this system

### JDK

JDK installations [Add JDK](#)  
List of JDK installations on this system

### Git

Git installations

Git
Name: Default
Path to Git executable: git

Install automatically

[Add Git](#)

### Mercurial

Mercurial installations [Add Mercurial](#)  
List of Mercurial installations on this system

Buscamos **Maven** y lo agregamos:

### 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**”.

The screenshot shows the Jenkins Global Tool Configuration interface. At the top, there's a header with the Jenkins logo and the text "Jenkins". Below it, the left sidebar has sections for "Maven" and "Docker".

**Maven Section:**

- "Maven installations" list: An empty list.
- "Add Maven" button: A grey button labeled "Add Maven".
- "Maven" configuration card:
  - "Name": "maven-user1" (highlighted with a red box).
  - "Install automatically": A checked checkbox with a tooltip explaining it allows Jenkins to install the tool on demand.
  - "Install from Apache": A section with a "Version" dropdown set to "3.6.3".
- "Add Installer" button: A grey button labeled "Add Installer".

**Docker Section:**

- "Docker installations" list: An empty list.
- "Add Docker" button: A grey button labeled "Add Docker".
- "Save" and "Apply" buttons: Two buttons at the bottom of the Docker section, with the "Save" button highlighted by a red box.

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 **acceder** 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.

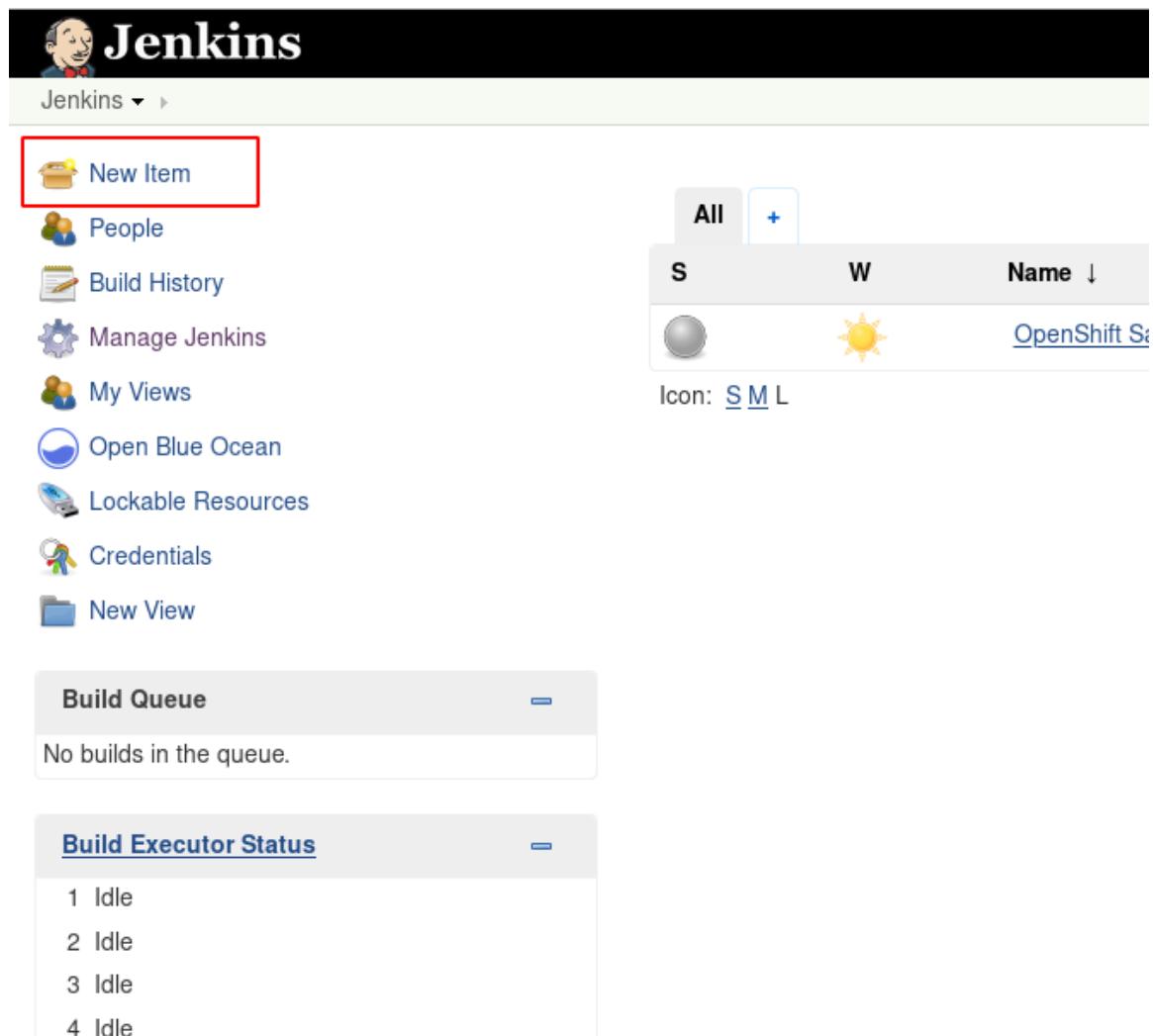
Para esto ejecutamos el siguiente comando: verificar que estemos utilizando el proyecto correspondiente, se subrayan en negro para su atención.

**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**”.



The screenshot shows the Jenkins dashboard. On the left, there is a sidebar with various links: New Item (highlighted with a red box), People, Build History, Manage Jenkins, My Views, Open Blue Ocean, Lockable Resources, Credentials, and New View. To the right of the sidebar, there is a search bar with fields for 'All', 'S', 'W', and 'Name' (sorted by name). Below the search bar, there is a section titled 'Icon:' with options 'S M L'. Underneath the sidebar, there is a 'Build Queue' section stating 'No builds in the queue.' At the bottom, there is a 'Build Executor Status' section showing four idle executors labeled 1, 2, 3, and 4.

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

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

Build Queue			
No builds in the queue.			

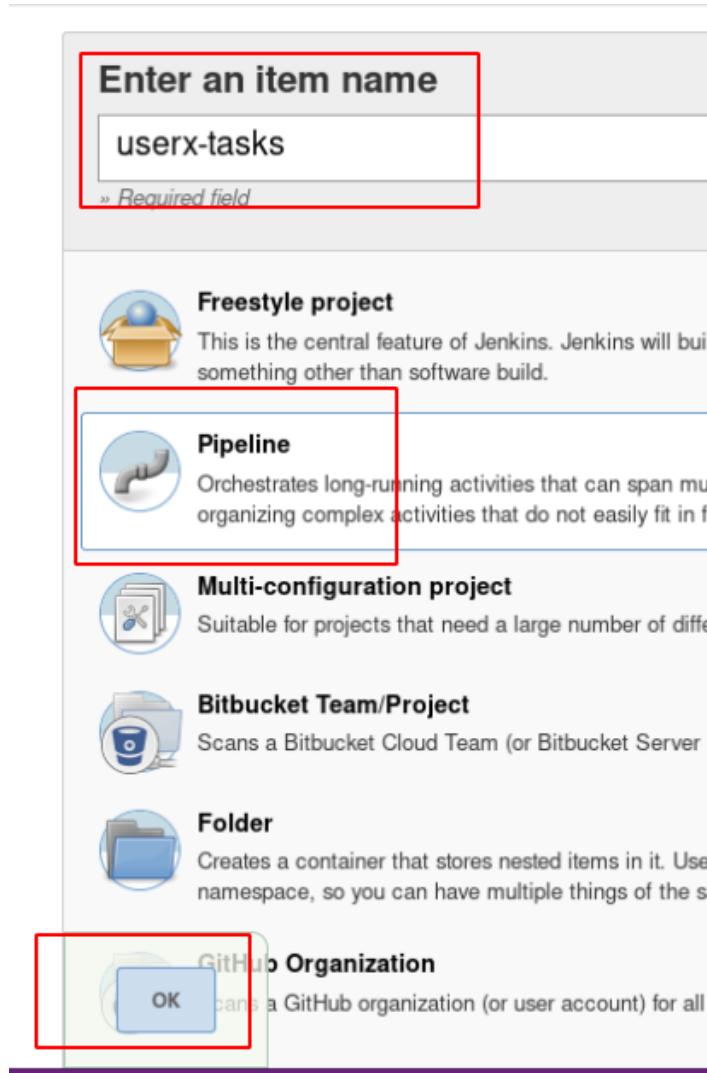
Build Executor Status			
1	Idle		
2	Idle		
3	Idle		
4	Idle		

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**”



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**.

The screenshot shows a 'Pipeline' configuration interface. The 'Pipeline script' tab is active. The script content is:

```
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    openshift.withCluster()  
18    openshift.selector("bc", "openshift-tasks").startBuild()  
19  }
```

Below the script area, there is a checked checkbox labeled 'Use Groovy Sandbox'. At the bottom left, there are two buttons: 'Save' (dark blue) and 'Apply' (light blue).

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

The screenshot shows the Jenkins Pipeline Tasks page. On the left, there is a sidebar with various options: Back to Dashboard, Status, Changes, Build Now (which is highlighted with a red box and has a red arrow pointing to it), Delete Pipeline, Configure, Full Stage View, Open Blue Ocean, Rename, and Pipeline Syntax. To the right, there is a "Pipeline Tasks" section with a "Stage View" sub-section containing the message "No data available. This Pipeline". Below the stage view is a "Permalinks" section.

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

The screenshot shows the Jenkins Pipeline user1-tasks page. On the left, there is a sidebar with various options: Back to Dashboard, Status, Changes, Build Now, Delete Pipeline, Configure, Full Stage View, Open Blue Ocean, Rename, and Pipeline Syntax. The "Build History" section shows a single build (#1) from May 16, 2020 at 7:29 AM. The "Stage View" section displays the execution times for three stages: Build Tasks (2min 20s), Tag Image (1s), and Deploy new image (988ms). The "Permalinks" section contains a link to the last build.

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**”

The screenshot shows the Red Hat OpenShift Container Platform interface. On the left, there's a sidebar with various options like 'Developer', '+Add', 'Topology' (which is highlighted with a red box), 'Monitoring', 'Builds', 'More', 'Search', 'Helm', 'Project Details', and 'Project Access'. The main area has a dropdown menu for 'Project: user1-lab6' and 'Application: all applications'. A sub-menu for 'Create Project' is open, showing several projects: 'user1-cicd-tareas' (highlighted with a red box), 'user1-lab1', 'user1-lab2', 'user1-lab3', 'user1-lab4', 'user1-lab5', and 'user1-lab6'. Below this, there's a Jenkins icon with a circular arrow symbol.

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.

The screenshot shows the Red Hat OpenShift Container Platform interface with the 'Resources' tab selected (highlighted with a red box). The main area displays information for the 'openshift-tasks' deployment config, including a Jenkins icon, a status summary, and sections for 'Pods', 'Builds', and 'Services'. Under 'Services', there's a 'Routes' section (highlighted with a red box) which lists a single route: 'http://openshift-tasks-user1-cicd-tareas.apps.cluster-qwe-e0e6.qwe-e0e6.example.opentlc.com'.

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/>

The screenshot shows the "OpenShift Tasks Demo (Wolfgang's Baseline Version 1.0)" application. At the top, there is a navigation bar with a "Home" button. Below the navigation bar, the main interface is divided into several sections:

- Logger:** Contains three buttons: "Log Info", "Log Warning", and "Log Error".
- Load Generator:** Contains a "Seconds" input field and a "Load" button.
- Danger Zone:** Contains three buttons: "HEALTHY" (green), "Toggle Health" (red), and "Kill Instance" (dark red).
- Info:** A table with the following data:

Pod Hostname	openshift-tasks-1-8kdjv
Pod IP	null
Used Memory	191 MB
Session ID	A0jQ1pmIWTO68Bv1PjKAu4-wgvNemy5HblbxNs
- Messages:** Displays the message "Nothing to report."