



Workshop Introducción a Openshift Container Platform

Noviembre 2019

Desde cualquier país marque la extensión 3840 (Call Center)

o contáctenos vía email: mercadeo@gbm.net

• Guatemala (502) 2424-2222 • El Salvador (503) 2505-9600 • Honduras (504) 2232-2319/09 •

• Nicaragua (505) 2255-6630 • Costa Rica (506) 2284-3999 • Panamá (507) 300-4800 •

República Dominicana • (809) 566-5161

www.gbm.net



1. 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.

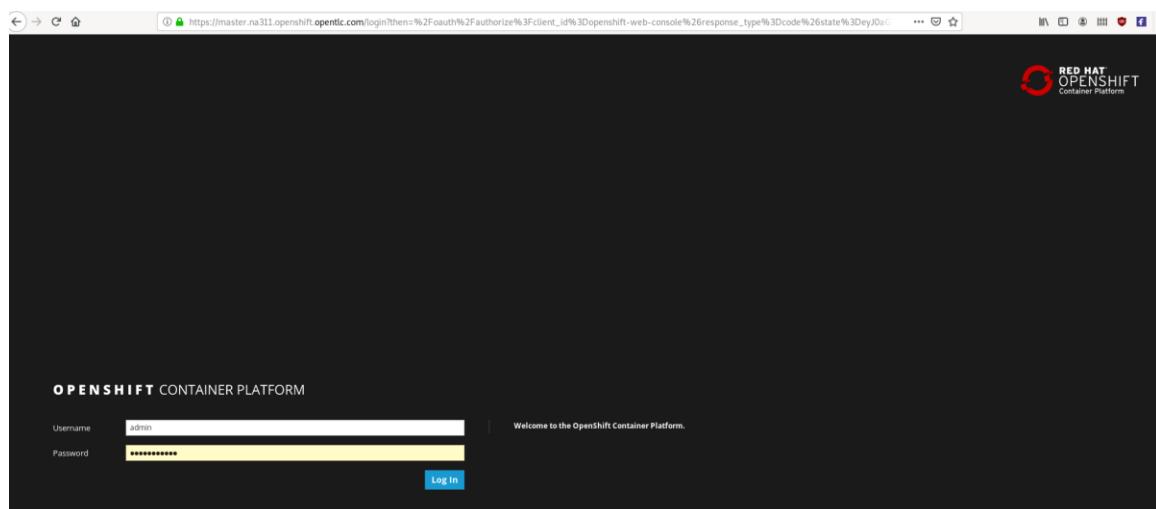
El usuario debe haber realizado los prerequisitos.

Solicitar URL del Openshift al presentador.

1. Loguearse en la consola web de Openshift Container Platform.

Usuario: (pedir usuario)

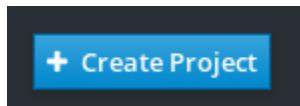
Password: openshift



2. Generación de terminal de linea de comandos por medio de un container web tty de NodeJs, necesaria para realizar el laboratorio.

2.1 Creamos un proyecto con el nombre: **userx-wetty-terminal**, cambiando el userx por el número de usuario correspondiente a cada persona.

A la izquierda de la interfaz web de Openshift hay un botón que aparece con nombre “Create Project” y le damos click.



Nota: El nombre de los proyectos no pueden llevar mayúsculas ni guiones bajos.

The screenshot shows the "Create Project" dialog box over a "Getting Started" interface. The dialog has fields for Name, Display Name, and Description. The "Name" field contains "user1-wetty-terminal". The "Display Name" field also contains "user1-wetty-terminal". The "Description" field contains "Proyecto para generar terminal ssh". There are "Cancel" and "Create" buttons at the bottom.

Getting Started

+ Create Project

Create Project

* Name

user1-wetty-terminal

A unique name for the project.

Display Name

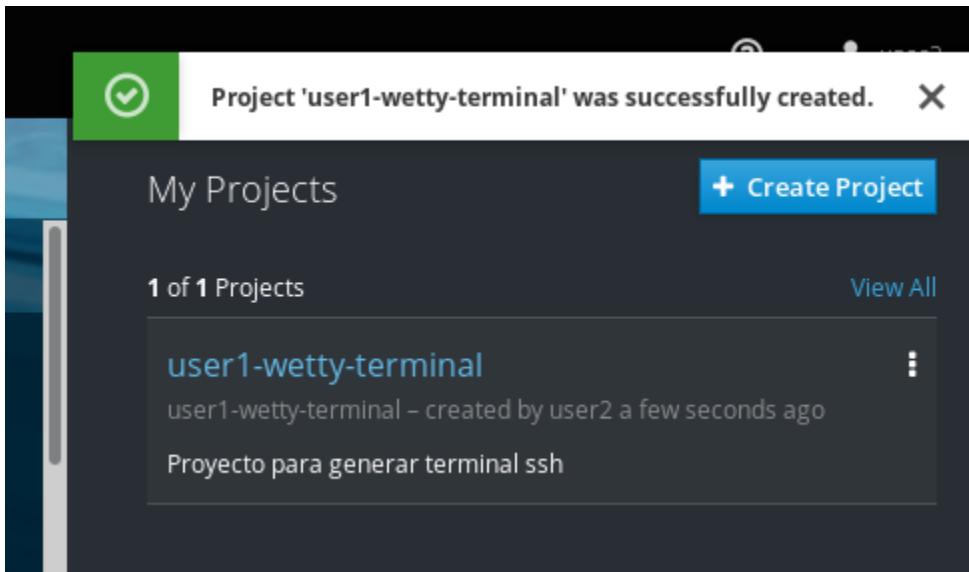
user1-wetty-terminal

Description

Proyecto para generar terminal ssh

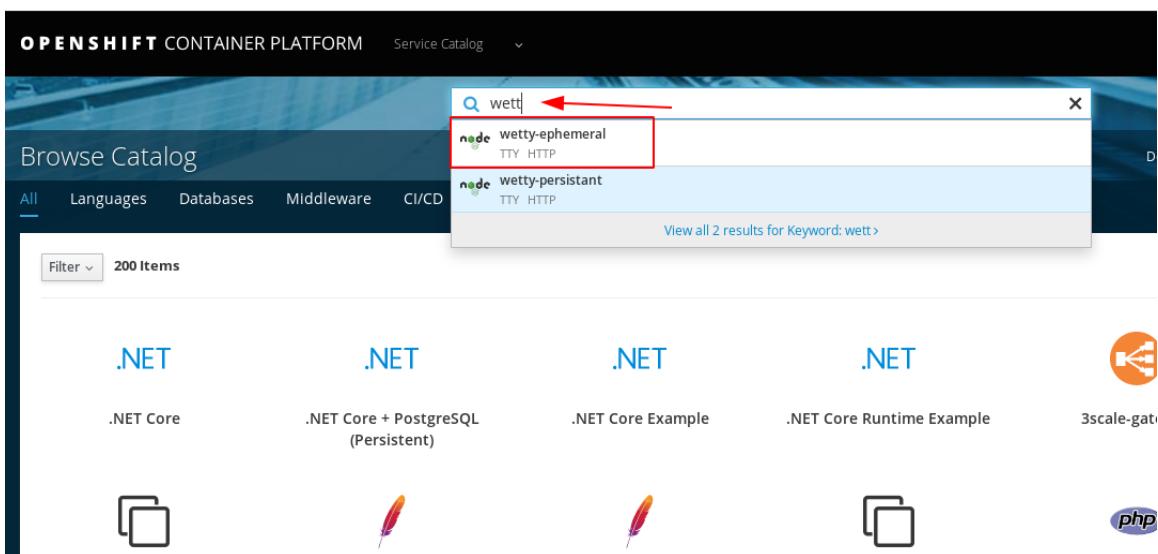
Cancel Create

Y le damos click en “Create”, automáticamente se crea el proyecto, debería dar un output como el siguiente:

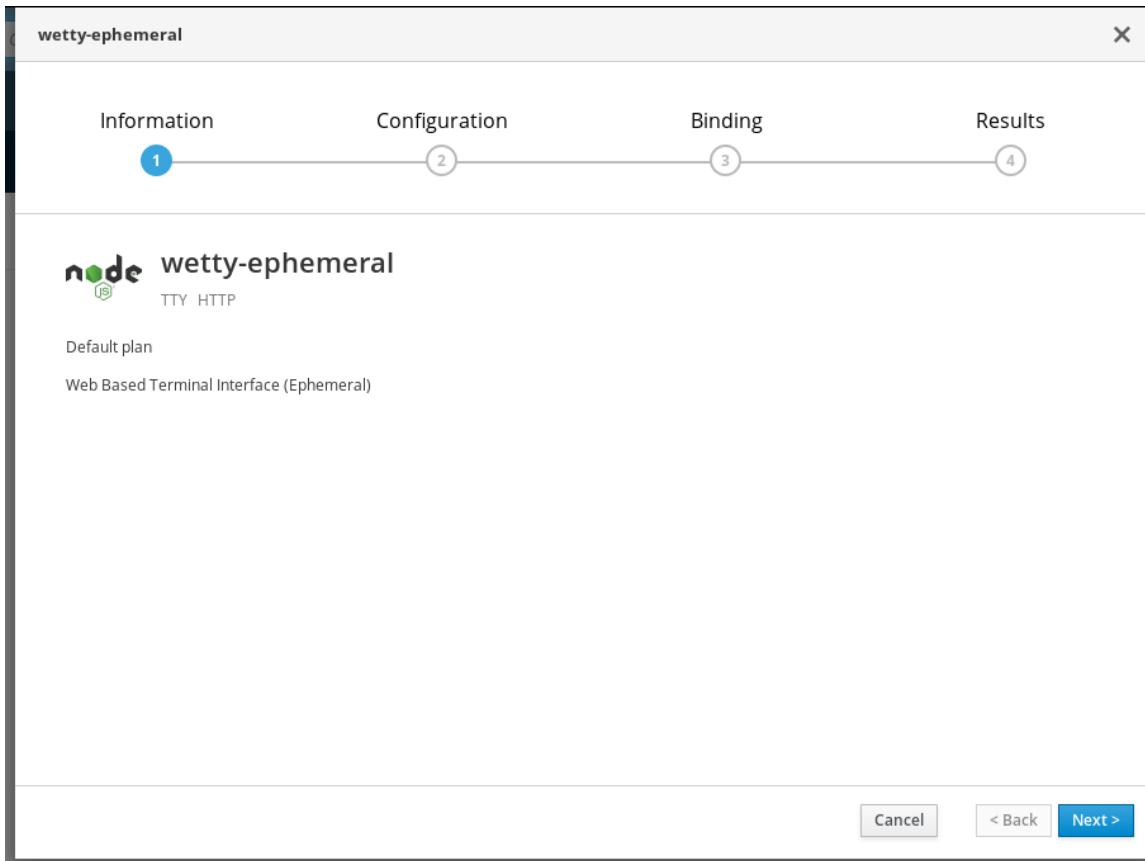


2.2 Ahora vamos a hacer deploy del “wetty-ephemeral”.

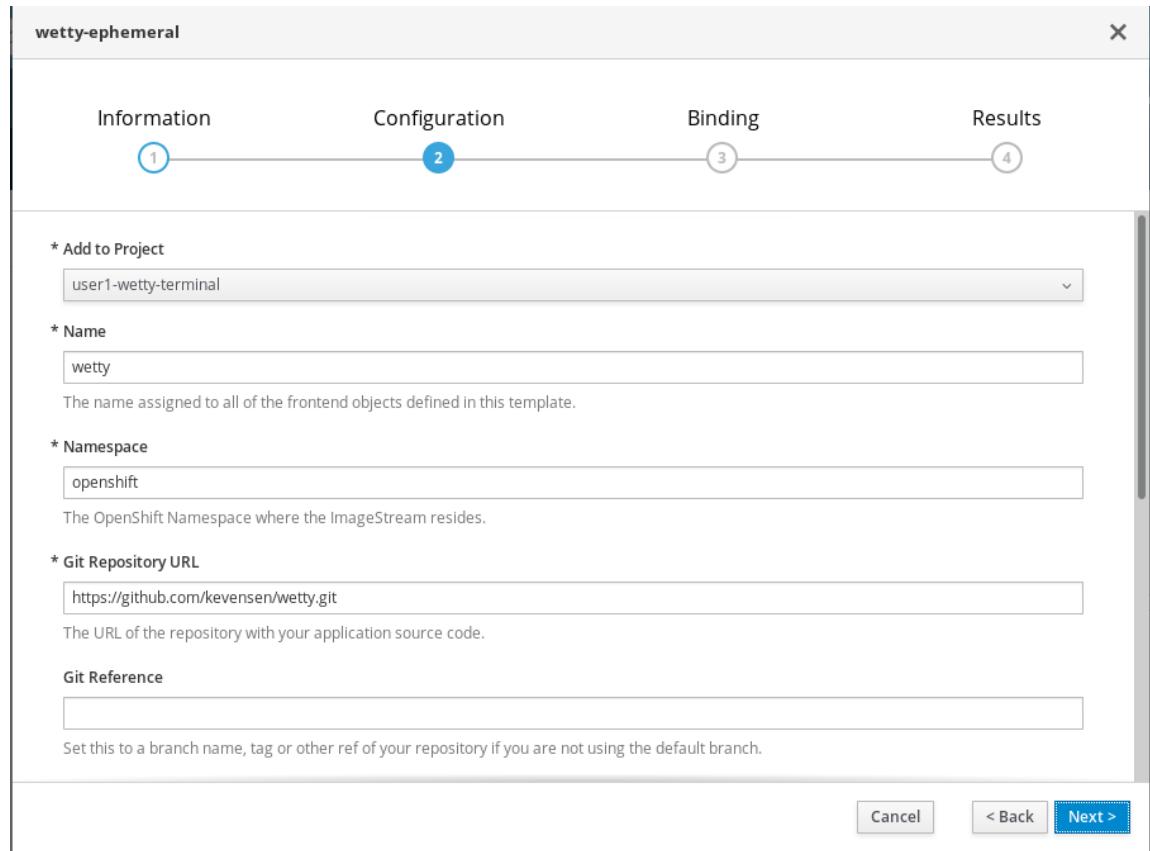
Para esto, buscamos **wetty-ephemeral** template desde el catálogo:



Para esto presione click en “wetty-ephemeral” y luego click en next:



Escogemos el proyecto que creamos anteriormente “**userx-wetty-terminal**”, dejamos todo default y le damos next y luego Create como se muestra en la imagen.



wetty-ephemeral

Information Configuration Binding Results

1 2 3 4

* Add to Project
user1-wetty-terminal

* Name
wetty

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

* Namespace
openshift

The OpenShift Namespace where the ImageStream resides.

* Git Repository URL
<https://github.com/kevensen/wetty.git>

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.

Cancel < Back Next >

wetty-ephemeral X

Information Configuration Binding Results

1 2 3 4

Create a binding for **wetty-ephemeral**

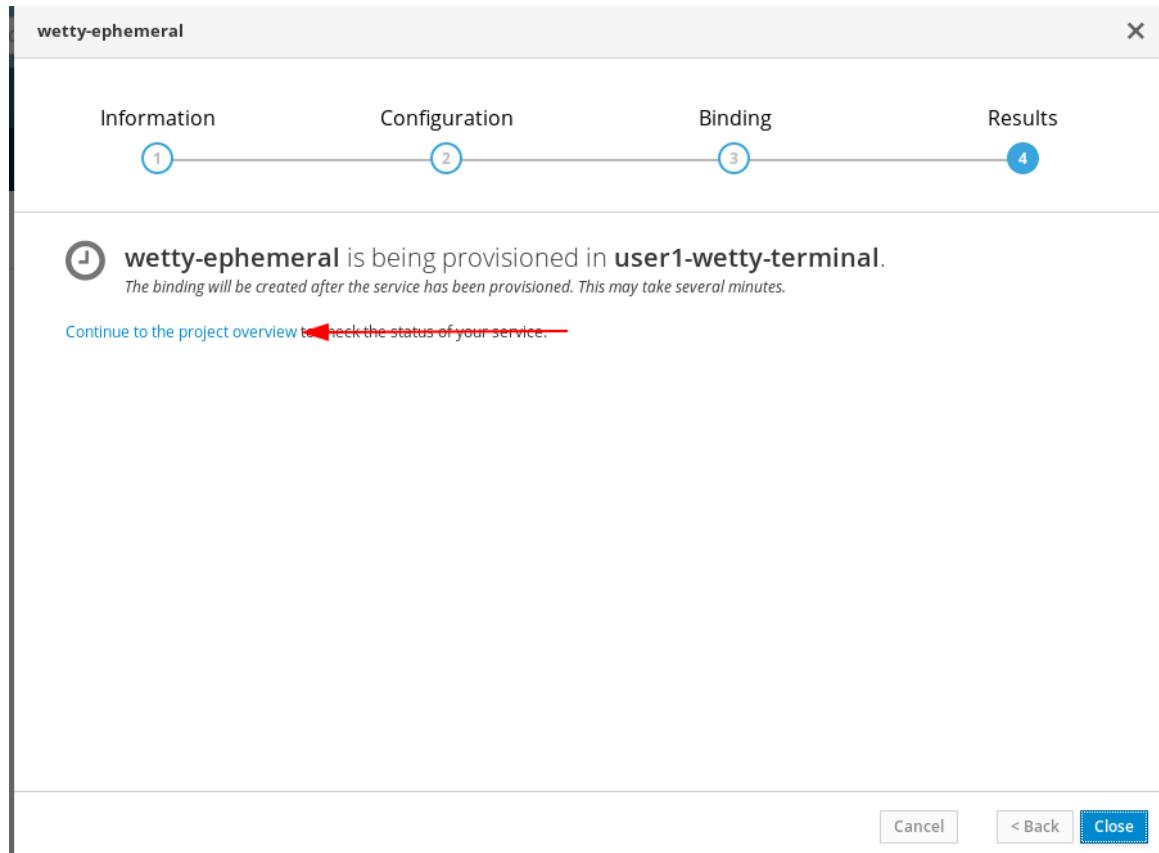
Bindings create a secret containing the necessary information for an application to use this service.

Create a secret in **user1-wetty-terminal** to be used later ←
Secrets can be referenced later from an application.

Do not bind at this time
Bindings can be created later from within a project.

Cancel < Back Create

Click en “Continue to the project overview” link.

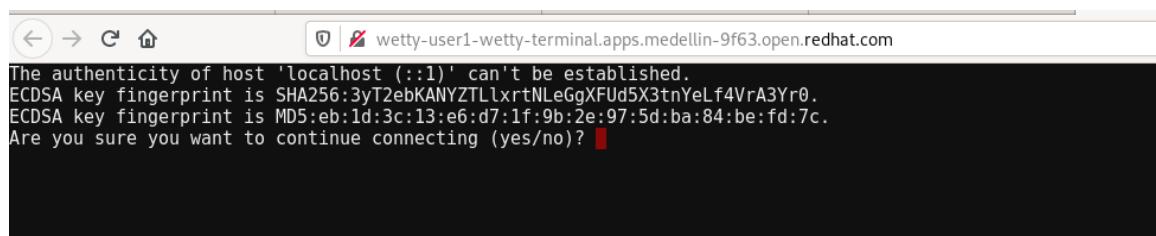


Esperamos que el deployment config esté con 1 pod “Running”, como se muestra a continuación.

The screenshot shows the OpenShift Container Platform interface for the project "user1-wetty-terminal". The left sidebar includes "Overview", "Applications", "Builds", "Resources", "Storage", "Monitoring", and "Catalog". The main area displays "Other Resources" with a "DEPLOYMENT CONFIG" section for "wetty, #1". It lists a "CONTAINERS" section for "wetty" with details: Image: user1-wetty-terminal/wetty 6368c53 204.8 MB, Build: wetty, #1, Source: adding start script 2657193, and Ports: 3000/TCP. To the right, a "1 pod" status is shown with a circled "1". Below this, the "NETWORKING" section shows a "Service - Internal Traffic" entry for "wetty" with port 3000/TCP. The "BUILDS" section shows a completed build for "wetty" with ID #1, indicating success. A "View Full Log" button is also present.

Luego click en la url que aparece en el deployment config en la parte de “Routes”, la url debe ser algo así: <http://wetty-user1-wetty-terminal.apps.medellin-9f63.open.redhat.com>.

Al darle click se nos deberá abrir una página como la siguiente.



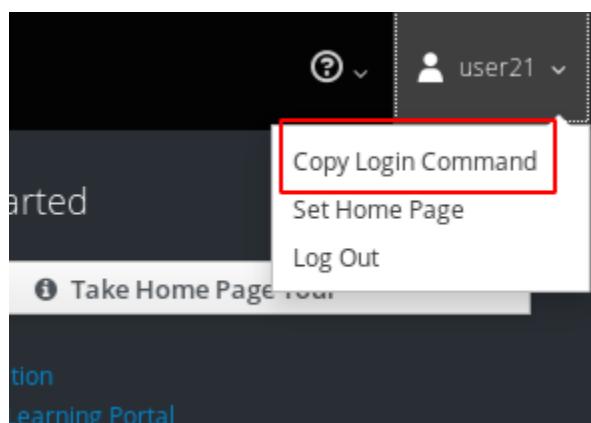
Aceptar la conexión e ingresar el password ssh:

Password: wetty

2.3 Loguearse a linea de comandos “oc”.

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

Y copiamos el login command y ese contenido lo pegamos en la web terminal que abrimos en el paso anterior.



Nos deberia mostrar un mensaje como el siguiente:

```
[default@wetty-1-9nclg ~]$ oc login https://master.medellin-9f63.open.redhat.com:443 --token=djwL1vU4uWvIMKbej5KslZpWwetCuGbCmiSUQL9EzC4
The server uses a certificate signed by an unknown authority.
You can bypass the certificate check, but any data you send to the server could be
intercepted by others.
Use insecure connections? (y/n):y

Logged into "https://master.medellin-9f63.open.redhat.com:443" as "user2" using the
token provided.

You have one project on this server: "user1-wetty-terminal"

Using project "user1-wetty-terminal".Welcome! See 'oc help' to get started.
[default@wetty-1-9nclg ~]$
```

3. Creación de proyecto por linea de comandos de OC.

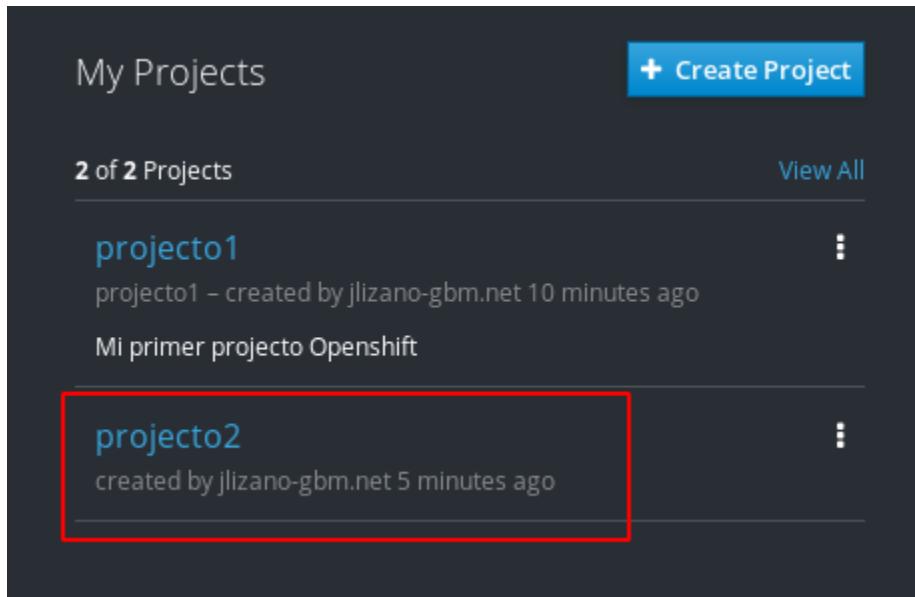
3.1 Ya logueados en la consola de OC, creamos el nuevo proyecto con el nombre “projeto2”, de la siguiente forma, deberia dar el siguiente output.

```
$ oc new-project miusuario-projeto2
Now using project "projeto2" 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.
```

3.2 Verificamos el nuevo proyecto “miusuario-projeto2” creado desde la linea de commandos oc.

```
$ oc get projects
NAME      DISPLAY NAME      STATUS
projeto1  projeto1          Active
projeto2            Active
```

3.3 También lo podemos verificar en la consola web.



My Projects

+ Create Project

2 of 2 Projects

View All

projecto1

projecto1 – created by jlizano-gbm.net 10 minutes ago

Mi primer proyecto Openshift

projecto2

created by jlizano-gbm.net 5 minutes ago

- Ejercicio desplegar aplicación Node.JS front end con conexión a una base de datos MongoDB, desde la interfaz gráfica de Openshift.

4.1 Desplegar una base de datos MongoDB (Ephemeral).

Nos movemos al service catalog y seleccionamos “Databases” tab.

The screenshot shows two instances of the OpenShift Container Platform Service Catalog interface. Both instances have the 'Databases' tab selected, indicated by a red box around the tab name. In the top instance, several .NET services are listed under the 'Databases' tab, including '.NET Core', '.NET Core + PostgreSQL (Persistent)', '.NET Core Example', and '.NET Core Runtime Exam'. In the bottom instance, specific database services like 'Mongo', 'MySQL', 'Postgres', and 'MariaDB' are listed. The interface includes a search bar, a 'Service Catalog' dropdown, and a 'Browse Catalog' header.

4.2 Escogemos Mongo y luego MongoDB (Ephemeral).

The screenshot shows the OpenShift Container Platform Service Catalog interface. At the top, the title "OPENSHIFT CONTAINER PLATFORM" and "Service Catalog" are visible. Below the title, there is a search bar with the placeholder "Search Catalog". The main area is titled "Browse Catalog" and features a navigation bar with categories: All, Languages, Databases (which is underlined), Middleware, CI/CD, and Other. Under the Databases category, there are five service cards: "All" (gray background), "Mongo" (highlighted with a dashed border), "MySQL" (white background), "Postgres" (white background), and "MariaDB" (white background). Below the service cards, there is a "Filter" button and a "2 Items" indicator. In the main list area, two items are shown: "MongoDB" (gray background) and "MongoDB (Ephemeral)" (white background, enclosed in a red rectangular box).

4.3 Y le damos next:

MongoDB (Ephemeral)

X

Information Configuration Binding Results

1 2 3 4

 MongoDB (Ephemeral)

Red Hat, Inc.

DATABASE MONGODB

[View Documentation](#) [Get Support](#)

Default plan

MongoDB database service, without persistent storage. For more information about using this template, including OpenShift considerations, see <https://github.com/sclorg/mongodb-container/blob/master/3.2/README.md>.

WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing

This template provides a standalone MongoDB server with a database created. The database is not stored on persistent storage, so any restart of the service will result in all data being lost. The database name, username, and password are chosen via parameters when provisioning this service.

Cancel < Back Next >

4.4 Procedemos con la creación de un proyecto nuevo de la siguiente forma:

MongoDB (Ephemeral) X

Information Configuration Binding Results

1 2 3 4

* Add to Project

Select or create project

* Create Project Create Project

RECENTLY VIEWED

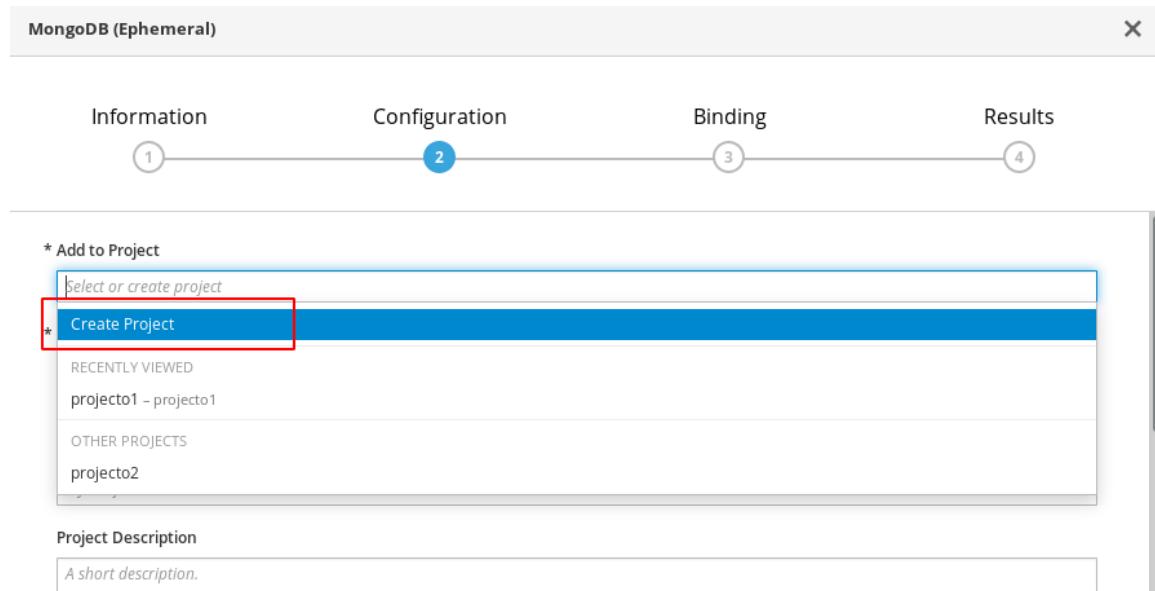
projecto1 - proyecto1

OTHER PROJECTS

projecto2

Project Description

A short description.



4.5 Le agregamos los siguientes datos al proyecto:

Project Name: usuario-mi-primer-app

Project Display Name: usuario-Mi primer app

Project Description: Primera aplicación consiste en un Node.JS front end y una DB MongoDB backend.

Y le damos "Next"

MongoDB (Ephemeral) X

Information Configuration Binding Results

1 2 3 4

* Add to Project Create Project

* Project Name mi-primer-app
A unique name for the project.

Project Display Name Mi primer app

Project Description Primera aplicación consiste en un Node.JS front end y una DB MongoDB backend.

* Memory Limit 512Mi
Maximum amount of memory the container can use.

Namespace Container

Cancel < Back Next >

4.6 En la página de “Bindings”, seleccionamos “Create a secret in **Mi primer app** to be used later”

Y le damos “Create”:

MongoDB (Ephemeral) X

Information Configuration Binding Results

1 2 3 4

Create a binding for MongoDB (Ephemeral)

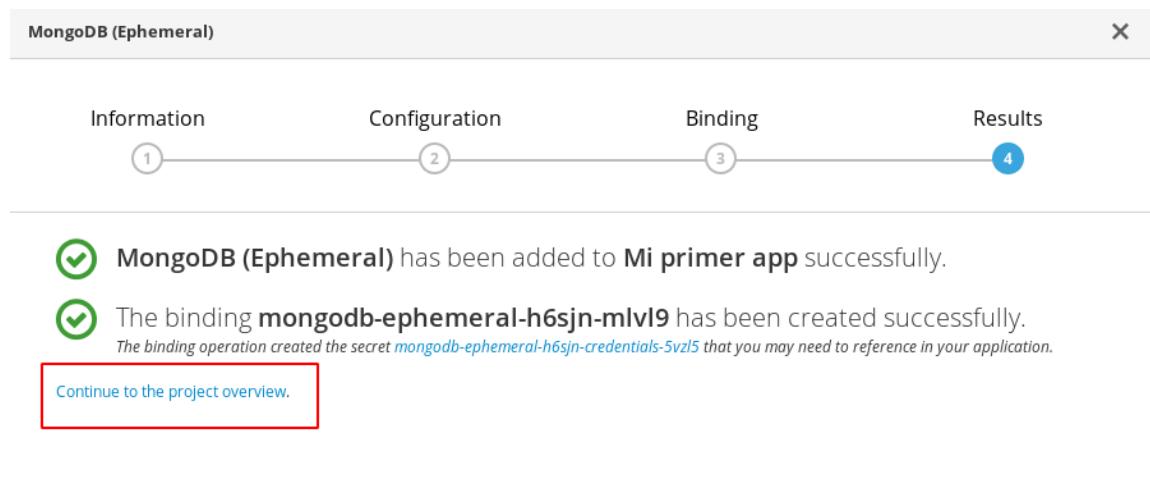
Bindings create a secret containing the necessary information for an application to use this service.

Create a secret in **Mi primer app** to be used later
Secrets can be referenced later from an application.

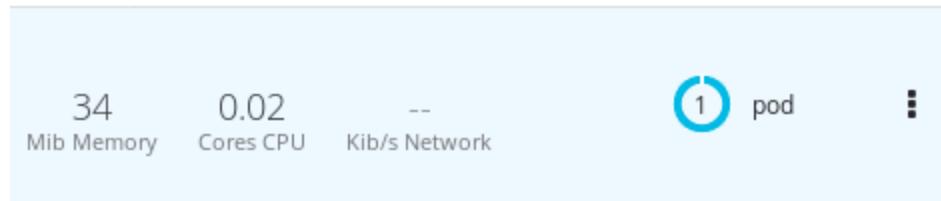
Do not bind at this time
Bindings can be created later from within a project.

Cancel < Back Create

4.7 En la página de resultados le damos click en “Continue to the project overview”.



Observe que el deployment está “Running” y se puede ver un pod disponible (indicado en azúl).



4.8 Click en MongoDB deployment para revisar los detalles entre ellos:

- La imagen usada para la base de datos MongoDB.
- El puerto en el cuál el servicio de MongoDB está escuchando dentro del contenedor.
- El nombre del servicio: “mongodb”, por el cual es accesible la base de datos.
- No hay rutas creadas para el servicio “mongodb”, lo cual está bien porque la base de datos no está expuesta afuera del cluster de Openshift.

DEPLOYMENT CONFIG
mongodb, #1

CONTAINERS

mongodb

- Image: openshift/mongodb
- Ports: 27017/TCP

NETWORKING

Service - Internal Traffic

mongodb

27017/TCP (mongo) → 27017

Average Usage Last 15 Minutes

34	Mib Memory
0.02	Cores CPU
--	Kib/s Network

1 pod

Routes - External Traffic

[Create Route](#)

4.9 Nos movemos a “Provisioned Services” y expandimos MongoDB (Ephemeral). Observe que el binding creado es similar a: “mongodb-ephemeral-h6sjn-mlvl9”

Provisioned Services

The screenshot shows the “MongoDB (Ephemeral)” service page. At the top, there is a dropdown menu and the service name. Below the service name, a note states: “MongoDB database service, without persistent storage. For more information about using this template, see the documentation.” A warning message follows: “WARNING: Any data stored will be lost upon pod destruction. Only use this template for testing.” Below the note, there are two links: “View Documentation” and “Get Support”. Under the heading “BINDINGS”, it lists a single binding named “mongodb-ephemeral-h6sjn-mlvl9”, which was created 20 minutes ago. There are “Delete” and “View Secret” buttons next to the binding name. Below these buttons is a “Create Binding” button.

4.10 Le damos click en “View Secret” y nos aparecerá una página como la siguiente:
Le damos click en Reveal Secret para ver el contenido.
Observe que el “uri” apunta a la IP del servicio de la dirección de la base de datos.

[Secrets](#) > [mongodb-ephemeral-h6sjn-credentials-5vzl5](#)

[mongodb-ephemeral-h6sjn-credentials-5vzl5](#) created 19 minutes ago

Opaque [Hide Secret](#)

admin_password	sSfVqLedp86KE8pL
database_name	sampledb
password	Coh7cIAdB0kK0bnl
uri	mongodb://172.30.120.242:27017
username	userLGA

There are no annotations on this resource.

4.11 Creamos una aplicación Node.JS.

Click en “Openshift Container Platform” en la columna izquierda de la pantalla para volver al home screen con el catálogo.

En el catálogo, seleccionar “Languages”.

The screenshot shows the OpenShift Container Platform interface. At the top, there's a navigation bar with the text "OPENShift CONTAINER PLATFORM" and "Service Catalog". Below this is a search bar with the placeholder "Search Catalog". The main area is titled "Browse Catalog" and has tabs for "All", "Languages", "Databases", "Middleware", "CI/CD", and "Other". The "Languages" tab is highlighted with a red box. Below the tabs, there's a filter section with a dropdown labeled "Filter" and the text "156 Items". The catalog lists several items under the ".NET" category:

Category	Item Name	Description
.NET	.NET Core	
	.NET Core + PostgreSQL (Persistent)	
.NET	.NET Core Example	
	.NET Core Runtime Example	
.NET	amp-pvc	
	Apache HTTP Server	
.NET	Apache HTTP Server (httpd)	
	CakePHP + MySQL	

Below the ".NET" section, there are other categories:

Category	Item Name	Description
Dancer + MySQL (Ephemeral)	Dancer + MySQL (Ephemeral)	
	Django + PostgreSQL	
Django + PostgreSQL (Ephemeral)	Django + PostgreSQL (Ephemeral)	
	JBoss A-MQ 6.3 (Ephemeral with SSL)	

4.12 Click en “JavaScript” y seleccionar “Node.js” y le damos Next en la página de “Service provisioning wizard”

Browse Catalog

All Languages Databases Middleware CI/CD Other

All Java JavaScript .NET Perl Ruby

Filter ▾ 3 Items

Node.js Node.js + MongoDB Node.js + MongoDB (Ephemeral)

Node.js

Information Configuration Results

1 2 3

node Node.js
BUILDER NODEJS

Build and run Node.js 10 applications on RHEL 7. For more information about using this builder image, including OpenShift considerations, see <https://github.com/bucharest-gold/centos7-s2i-nodejs>.

Sample Repository: <https://github.com/openshift/nodejs-ex.git>

Cancel < Back Next >

4.13 Use lo siguiente para configurar la aplicación.

Add to Project: Verifique que se utilice el mismo proyecto donde se desplegó la DB MongoDB.

Version: 10 – latest

Application Name: miprimerapp

Git Repository: <https://github.com/openshift/nodejs-ex.git>

Después click en “Create”

The screenshot shows the 'Node.js' configuration dialog. It has three tabs: 'Information' (step 1), 'Configuration' (step 2, currently selected), and 'Results' (step 3). The 'Configuration' tab contains four input fields, each with a red box highlighting it:

- * Add to Project: Mi primer app
- Version: 10 — latest
- * Application Name: miprimerapp
- * Git Repository: <https://github.com/openshift/nodejs-ex.git>

Below these fields is a link: Try Sample Repository ↗. At the bottom right are three buttons: 'Cancel', '< Back', and a blue 'Create' button, which is also highlighted with a red box.

4.14 Desde la página de resultados, click en “Continue to the project overview”.

The screenshot shows the results of creating a Node.js application named "miprimerapp". At the top, there are two tabs: "Information" (marked with a circled "1") and "Configuration" (marked with a circled "2"). Below the tabs, a green checkmark icon indicates success, followed by the message: "miprimerapp has been created in **Mi primer app** successfully." A red box highlights the blue link "Continue to the project overview" which is described as "to check the status of your application as it builds and deploys".

4.15 Verifique que el deployment está iniciado.

También verifique que hay un pod “Running” con el nombre “miprimerapp”, click para ver más detalles.

Una vez que el pod esté “Running”, click en la ruta que debería mostrarse similar a:

<http://miprimerapp-mi-primer-app.apps.na311.openshift.opentlc.com> .

The screenshot shows the "miprimerapp" application overview. At the top, the URL "http://miprimerapp-mi-primer-app.apps.na311.openshift.opentlc.com" is displayed. Below the URL, there is a summary of resource usage: 29 Mib Memory, 0 Cores CPU, and -- Kib/s Network. To the right, a large teal circle contains the number "1" with the word "pod" underneath, indicating one pod is running. Below this, under the heading "Routes - External Traffic", is the URL "http://miprimerapp-mi-primer-app.apps.na311.openshift.opentlc.com". A note below the URL states "Route miprimerapp, target port 8080-tcp". At the bottom of the screenshot, a message indicates "Build #1 is complete" and was "created 9 minutes ago".

APPLICATION

miprimerapp



DEPLOYMENT CONFIG

miprimerapp, #1

CONTAINERS

miprimerapp

- Image:** mi-primer-app/miprimerapp 250838b 187.3 MiB
- Build:** [miprimerapp, #1](#)
- Source:** Merge pull request #206 from liangxia/okd [e59fe75](#)
- Ports:** 8080/TCP

NETWORKING

Service - Internal Traffic

miprimerapp

8080/TCP (8080-tcp) → 8080

BUILDS

miprimerapp

4.16 Agregar Unión entre la base de datos MongoDB backend y la aplicación Node.JS desplegadas en el proyecto “mi primer app”

En la columna izquierda seleccionamos “Applications”, dentro de “Applications”, seleccionamos “Deployments” y le seleccionamos dentro de “Deployments”, “miprimerapp”.

The screenshot shows the MongoDB Atlas interface for the project "Mi primer app". On the left sidebar, under the "Applications" section, the "Deployments" tab is selected. A single deployment named "miprimerapp" is listed, created 19 minutes ago. The "History" tab is active, showing a deployment log entry: "Deployment #1 is active. View Log" (created 18 minutes ago). Below the log, there is a table titled "Deployment" with one row: "#1 (latest)" which is "Active, 1 replica".

Deployment	Status
#1 (latest)	Active, 1 replica

4.17 Ingresamos a “Environment” dentro de “miprimerapp”

The screenshot shows the Kubernetes UI interface. At the top, there is a navigation bar with 'Deployments' and 'miprimerapp'. Below this, the deployment name 'miprimerapp' is displayed with a creation timestamp 'created 20 minutes ago'. A horizontal navigation bar below the deployment name includes tabs for 'History', 'Configuration', 'Environment' (which is underlined, indicating it is the active tab), and 'Events'. The main content area is titled 'Container miprimerapp'. Under this title, there is a 'Name' field containing the placeholder 'Name'. Below the field are two links: 'Add Value' and 'Add Value from Config Map or Secret'. A section titled 'Environment From ②' follows, with a 'Config Map/Secret' heading and a dropdown menu labeled 'Select a resource'. At the bottom of the form is a 'Save' button.

4.18 Esta aplicación Node.JS puede recibir la configuración de la base de datos a través de variables de entorno. Desafortunadamente, los nombres de variables que la aplicación espera son diferentes de los nombres de campo en su binding secret. Si fueran idénticos, podrías vincular todo el secreto. Pero no lo son, por lo que debe vincular campos individuales.

Le damos click a “Add Value from Config Map or Secret”.

Y agregamos 4 variables como las siguientes:

Name	Select a Resource	Secret Key
MONGODB_USER	Select your secret from the pull-down list.	username
MONGODB_DATABASE	Select your secret from the pull-down list.	database_name
MONGODB_PASSWORD	Select your secret from the pull-down list.	password
MONGODB_ADMIN_PASSWORD	Select your secret from the pull-down list.	admin_password

Se debe agregar una variable de entorno regular con el Name:

DATABASE_SERVICE_NAME y mongodb como Value.

Y le damos click en “Save”.

Container miprimerapp

Name	Value
MONGODB_USER	mongodb-ephemeral-h6sjn... ▾ username
MONGODB_DATABASE	mongodb-ephemeral-h6sjn... ▾ database_name
MONGODB_PASSWORD	mongodb-ephemeral-h6sjn... ▾ password
MONGODB_ADMIN_PASSWORD	mongodb-ephemeral-h6sjn... ▾ admin_password
DATABASE_SERVICE_NAME	mongodb

[Add Value](#) | [Add Value from Config Map or Secret](#)

4.19 Le damos Click en “Overview” en la columna izquierda del panel.
Verifique que el número a la par del deployment configuration “miprimerapp” incrementó ya que se cambió la configuración.

APPLICATION

miprimerapp

> DEPLOYMENT CONFIG
miprimerapp, #2

Nota: Cada vez que se cambia algún parámetro de la configuración, Openshift crea una nueva versión del deployment configuration y hace un redeploy de la aplicación. Esto asegura que podamos hacer rollback en caso de algún problema.

Para verificar que la aplicación se unió o ligó correctamente a la base de datos, darle click en la ruta de la aplicación de nuevo, se esperaría ver un botón en la esquina inferior derecha con el label “DB Connection Info”, esto con detalles sobre la conexión a la base de datos.

4.20 También se puede ver que cada vez que se refresca la página, el “Page View Count” incrementa en 1.

The screenshot shows a web browser window with the URL `miprimerapp-mi-primer-app.apps.na311.openshift.opentlc.com`. The page title is "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:

- From the Web Console homepage, navigate to your project
- Click on Browse > Builds
- Click the link with your BuildConfig name
- Click the Configuration tab
- Click the "Copy to clipboard" icon to the right of the "GitHub webhook URL" field
- Navigate to your repository on GitHub and click on repository settings > webhooks > Add webhook
- Paste your webhook URL provided by OpenShift in the "Payload URL" field
- Change the "Content type" to 'application/json'
- 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>
```

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://172.30.120.242:27017/sampled`

5. Crear una aplicación desde linea de comandos OC.

5.1 Crear un nuevo proyecto:

Cambien "jjl" por sus iniciales

```
$ oc new-project jjl-new-apps
Now using project "jjl-new-apps" 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.

5.2 Crear una nueva aplicación utilizando la imagen latest de wildfly disponible en Docker Hub,

```
$ 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.
```

5.3 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
```

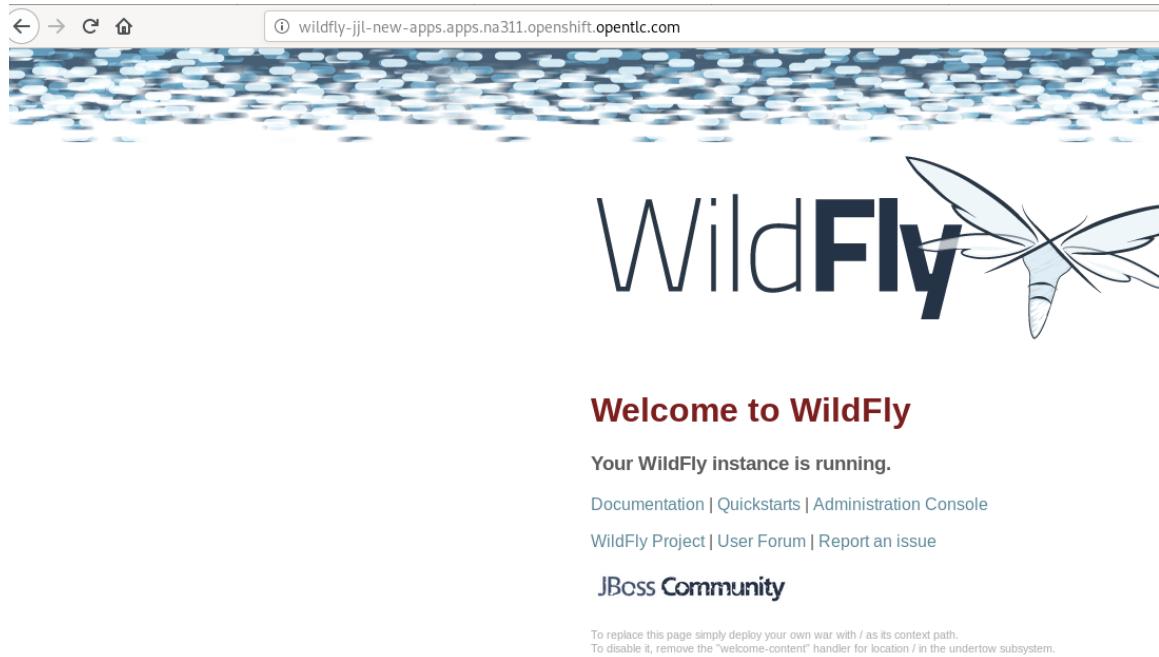
5.4 Los servicios se utilizan para la comunicación interna de OpenShift. De esta manera, otra aplicación no necesita saber la dirección IP real del pod o cuántos pods se están ejecutando para una aplicación determinada. Pero dado que los servicios no son accesibles fuera del clúster de OpenShift, debe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

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

5.5 Verificar la ruta creada:

```
$ oc get route
NAME HOST/PORT PATH SERVICES PORT
TERMINATION WILDCARD
wildfly wildfly-jjl-new-apps.apps.na311.openshift.opentlc.com wildfly 8080-tcp
None
```

5.6 Copiamos la ruta que se creó con el expose en un browser, en este caso es:
<http://wildfly-jjl-new-apps.apps.na311.openshift.opentlc.com/>



6. Ejercicio: Crear una aplicación PHP simple desde un repositorio de GitHub y explora sus diversas partes utilizando la interfaz de línea de comando (CLI) de OpenShift. 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.

6.1 Crear un proyecto con el nombre “miusuario-managing-apps”.

```
$ oc new-project managing-apps
Now using project "managing-apps" 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.

6.2 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.
```

6.3 Creamos la ruta para poder accesar la aplicación desde fuera del clúster de Openshift.

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

6.4 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
```

6.5 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)

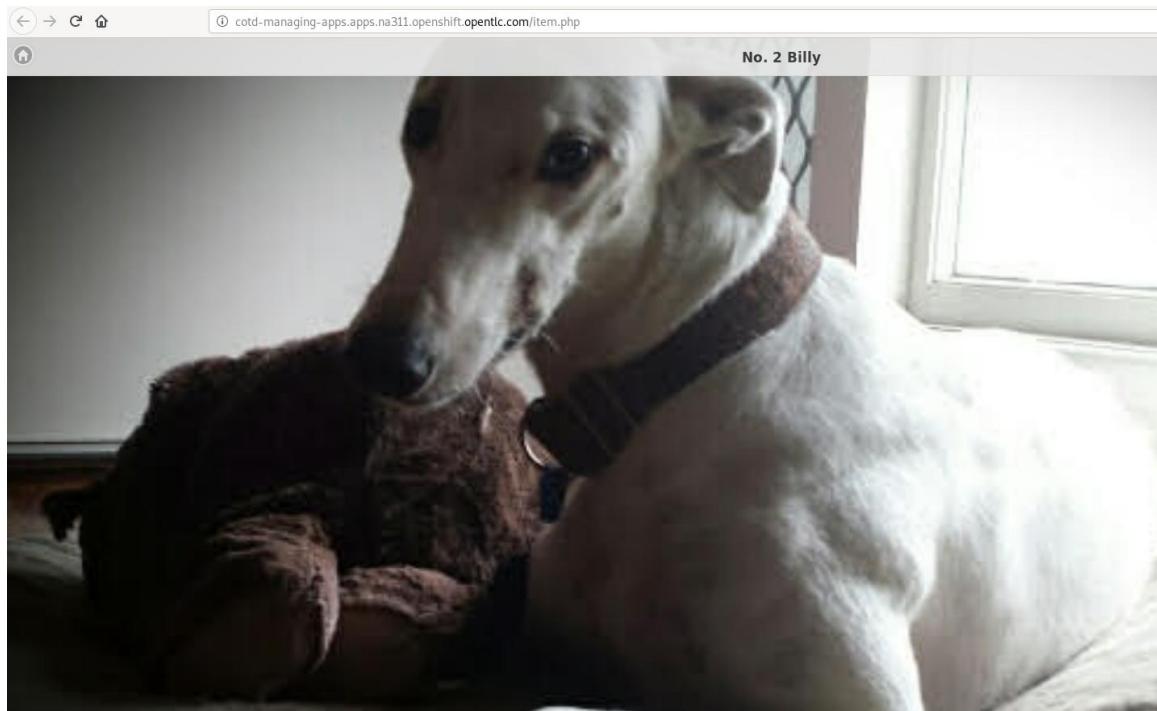
NAME      TYPE      FROM      LATEST
buildconfig.build.openshift.io/cotd  Source  Git      1

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-managing-apps.apps.na311.openshift.opentlc.com
cotd  8080-tcp      None
```

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



6.7 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
```

Hacemos la escalación directamente en el deployment config “cotd” a 3 réplicas.

```
$ oc scale dc cotd --replicas=3
deploymentconfig.apps.openshift.io/cotd scaled
```

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
```

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-managing-apps.apps.na311.openshift.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...
```

6.8 Ahora escalamos de nuevo a un pod.

```
$ oc scale dc cotd --replicas=1
deploymentconfig.apps.openshift.io/cotd scaled
```

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.

6.9 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"

--> Creating resources with label name=cotd2 ...
imagestream.image.openshift.io "cotd2" created
buildconfig.build.openshift.io "cotd2" created
deploymentconfig.apps.openshift.io "cotd2" created
service "cotd2" created
--> Success
Build scheduled, use 'oc logs -f bc/cotd2' 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/cotd2'  
Run 'oc status' to view your app.
```

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.

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
```

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-managing-apps.apps.na311.openshift.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
```

6.10 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.

7. En esta práctica de laboratorio, explora un entorno simple de CI / CD utilizando Jenkins como el servidor de CI / CD. Crea un contenedor Jenkins y luego usa el servidor Jenkins para construir e implementar una aplicación Java EE simple. Configura la aplicación en un proyecto separado y prepara el proyecto para que lo controle Jenkins en lugar de OpenShift.

7.1 Realice login en la consola web de Openshift.
Ir al catálogo, click en CI/CD, y click en Jenkins.

The screenshot shows the OpenShift Container Platform Service Catalog interface. At the top, there is a navigation bar with the title "OPENSHIFT CONTAINER PLATFORM" and a "Service Catalog" dropdown. Below the navigation bar is a search bar labeled "Search Catalog". The main area is titled "Browse Catalog" and features a navigation menu with tabs: "All" (selected), "Languages", "Databases", "Middleware", "CI/CD" (which is highlighted with a red box), and "Other". Below the menu, there is a filter button labeled "Filter" and a count of "200 Items". The main content area displays several service cards, each with a ".NET" label. One card, located at the bottom left, has a Jenkins icon and the name "Jenkins".

This screenshot shows the same OpenShift Service Catalog interface as the previous one, but it focuses on the Jenkins service card. The Jenkins card is highlighted with a red box. The card features a Jenkins icon and the text "Jenkins". Below the card, there is a filter button labeled "Filter" and a count of "2 Items". To the right of the Jenkins card, another Jenkins card is visible with the text "Jenkins (Ephemeral)".

Le damos click en “Next”.

The screenshot shows a web-based configuration interface for a Jenkins application. At the top, there is a navigation bar with four tabs: "Information" (highlighted with a blue circle containing the number 1), "Configuration" (with a blue circle containing the number 2), "Binding" (with a blue circle containing the number 3), and "Results" (with a blue circle containing the number 4). Below the tabs, the "Information" section displays the Jenkins logo, the text "Jenkins", "Red Hat, Inc.", "INSTANT-APP JENKINS", and links to "View Documentation" and "Get Support". A note below states: "Default plan" and "Jenkins service, with persistent storage." A note at the bottom says: "NOTE: You must have persistent volumes available in your cluster to use this template." Another note below it says: "This template deploys a Jenkins server capable of managing OpenShift Pipeline builds and supporting OpenShift-based oauth login." At the bottom right, there are three buttons: "Cancel", "< Back", and "Next >". The "Next >" button is highlighted with a red rectangle.

7.2 Se debe crear un nuevo proyecto y cambiar lo siguiente:

Create Project

Project Name: userx-ci-cd

Project Display Name: Mi Projecto CI/CD - userx

Memory Limit: 2Gi

Disable memory intensive administrative monitors: true

Click en “Next” para continuar con la siguiente página.

Jenkins

Information Configuration

1 2

* Add to Project

Create Project

* Project Name

ci-cd-user1

A unique name for the project.

Project Display Name

MI proyecto CI/CD user1

Project Description

A short description.

Jenkins

Information Configuration Binding Results

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

Memory Limit

2Gi

Maximum amount of memory the container can use.

*** Volume Capacity**

1Gi

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

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

Cancel < Back **Next >**

The screenshot shows the Jenkins configuration interface. The top navigation bar has tabs for Information, Configuration, Binding, and Results, with Configuration being the active tab. Below the tabs is a progress bar with four numbered circles (1, 2, 3, 4). The main content area contains several configuration fields. Some fields have red boxes around them: the 'Memory Limit' input field (containing '2Gi'), the 'Disable memory intensive administrative monitors' input field (containing 'true'), and the 'Next >' button at the bottom right. The 'Disable memory intensive administrative monitors' field is described as being used for synchronization with the Jenkins Update Center on start, with the note that it disables the Jenkins core update monitor and site warnings monitor if set to true.

7.3 En la página de Bindings, dejar “Do not bind at this time” y seleccionar y click en Crear.

The screenshot shows the Jenkins Binding creation interface. At the top, there are four tabs: Information (1), Configuration (2), Binding (3), and Results (4). The Binding tab is active. Below the tabs, the title "Create a binding for Jenkins" is displayed. A descriptive text states: "Bindings create a secret containing the necessary information for an application to use this service." Two options are presented:

- Create a secret in **MI proyecto CI/CD user1** to be used later
Secrets can be referenced later from an application.
- Do not bind at this time
Bindings can be created later from within a project.

At the bottom right, there are three buttons: "Cancel", "< Back", and a blue "Create" button, which is highlighted with a red box.

En la página de resultados, click en “Continue to Project overview”.

The screenshot shows a Jenkins provisioning status page titled "Jenkins". At the top, there is a navigation bar with four tabs: "Information" (step 1), "Configuration" (step 2), "Binding" (step 3), and "Results" (step 4). Step 4 is highlighted with a blue circle and a number "4". Below the tabs, a message states: "Jenkins is being provisioned in MI proyecto CI/CD user1. This may take several minutes." A red box highlights the link "Continue to the project overview" in blue text. At the bottom right, there are three buttons: "Cancel", "< Back", and a blue "Close" button.

7.4 Abrir el jenkins deployment y esperar a que el Jenkins esté totalmente arriba y funcional, debe mostrarse de la siguiente forma:
Para que esto sea así, debemos esperar varios minutos.

The screenshot shows the OpenShift application details page for 'jenkins-persistent'. At the top, there is a search bar with 'Name' and 'Filter by name' options. Below the search bar, the application name 'jenkins-persistent' is displayed. Under 'APPLICATION', there is a 'DEPLOYMENT CONFIG' section titled 'jenkins, #1'. In the 'CONTAINERS' section, there is one container named 'jenkins' with the image 'openshift/jenkins'. Under 'NETWORKING', there is a 'Service - Internal Traffic' section for 'jenkins' with port 80/TCP (web) mapped to 8080. There is also a 'Service - Internal Traffic' section for 'jenkins-jnlp' with port 50000/TCP (agent) mapped to 50000.

Deployment Config	Container	Image	Port	Target Port
jenkins, #1	jenkins	openshift/jenkins	80/TCP (web)	8080

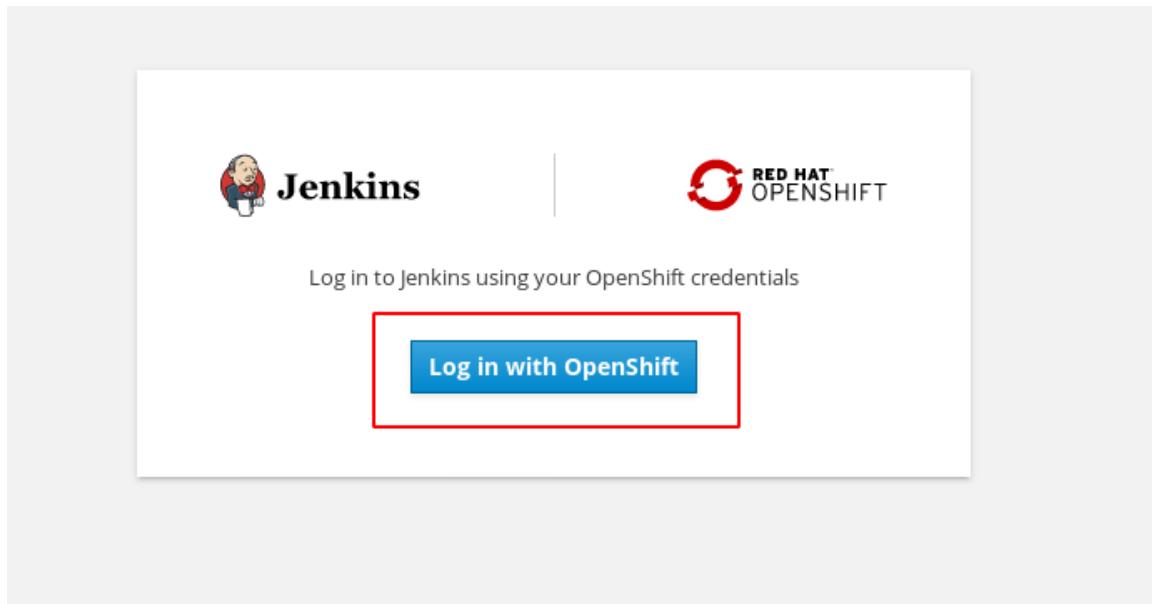
Service	Type	Port	Target Port
jenkins	Service - Internal Traffic	80/TCP (web)	8080
jenkins-jnlp	Service - Internal Traffic	50000/TCP (agent)	50000

The screenshot shows the Jenkins application details page in the OpenShift web console. At the top, there is a search bar and a dropdown menu set to "List by Application". Below the header, the URL <https://jenkins-ci-cd-user1.apps.medellin-9f63.open.redhat.com> is displayed. A large blue circular icon indicates "1 pod". On the right side, there are three vertical dots and two arrows pointing up and down. Below the URL, the text "Routes - External Traffic" is followed by the URL again and a "Route Jenkins" link. Further down, another "Routes - External Traffic" section is shown with a "Create Route" link.

Le damos click en la ruta de acceso de Jenkins.

This screenshot shows the same Jenkins application details page as the previous one, but with a red rectangular box highlighting the URL <https://jenkins-ci-cd-user1.apps.medellin-9f63.open.redhat.com>. The rest of the interface remains identical to the first screenshot.

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



Si nos pide usuario y password, debemos usar los mismos que utilizamos para loguernos a la consola web de openshift.

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>

 Jenkins

Jenkins >

New Item All +

S W Name ↓ Last Success

OpenShift Sample N/A

Icon: S M L

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

7.6 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”.

The screenshot shows the Jenkins dashboard. At the top, there is a navigation bar with the Jenkins logo and the word "Jenkins". Below the navigation bar, there is a sidebar with several links: "New Item", "People", "Build History", "Manage Jenkins" (which is highlighted with a red box), "My Views", "Open Blue Ocean", "Lockable Resources", "Credentials", and "New View". To the right of the sidebar, there is a "Views" section with three items: "All" (selected), "S", and a circular icon. Below the sidebar, there are two main sections: "Build Queue" (which says "No builds in the queue.") and "Build Executor Status" (which lists "1 Idle", "2 Idle", "3 Idle", "4 Idle", and "5 Idle").

Le damos click en “Global Tool Configuration”

The screenshot shows the Jenkins Manage Jenkins interface. On the left, there's a sidebar with links like New Item, People, Build History, Manage Jenkins (which is selected), My Views, Open Blue Ocean, Lockable Resources, Credentials, and New View. Below this are two expandable sections: Build Queue (empty) and Build Executor Status (listing 1-5 Idle). The main content area is titled "Manage Jenkins". It contains several status boxes: one about Jenkins root URL being empty (with a note to provide an accurate value in Jenkins configuration); another about builds running as SYSTEM user (not recommended); a CSRF protection warning; a master security subsystem off warning; and a "Configure System" link. The "Global Tool Configuration" link is highlighted with a red box.

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 th...

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 consic...](#)

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 i...

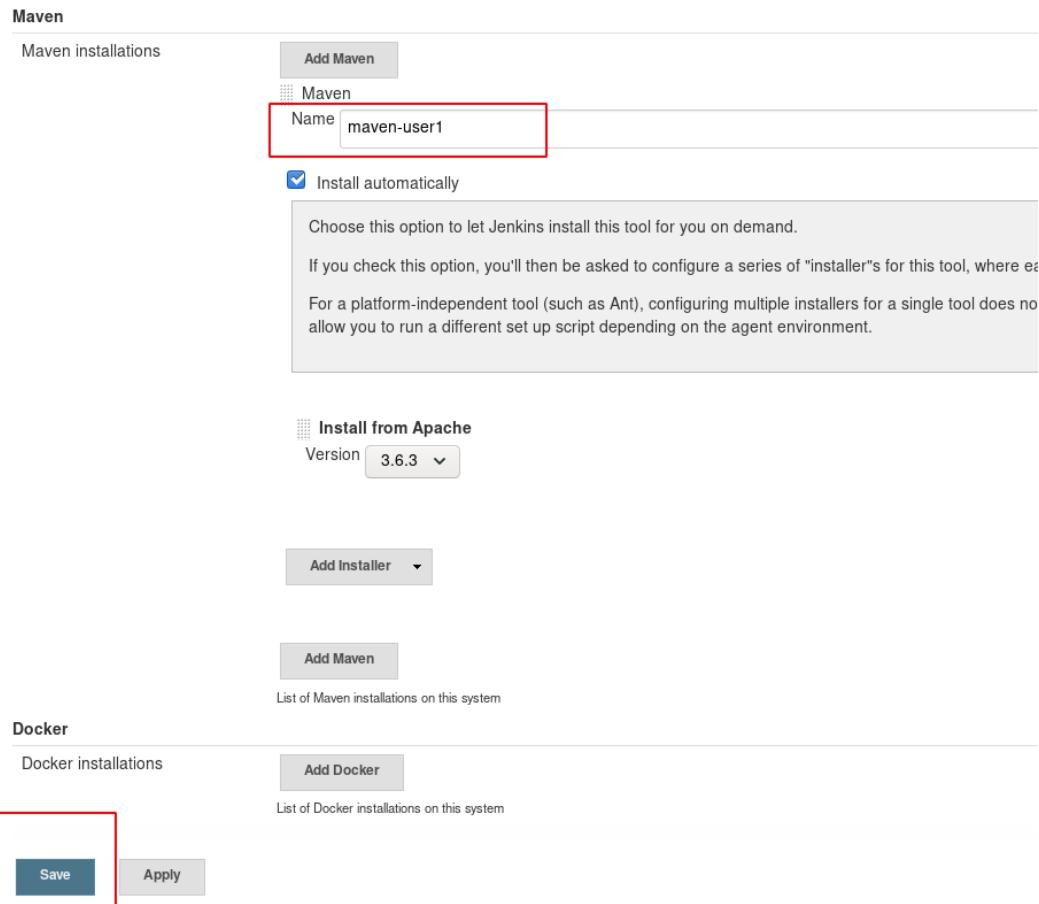
Vamos a ver una imagen como la siguiente.

The screenshot shows the Jenkins Global Tool Configuration page. At the top, there are links to 'Back to Dashboard' and 'Manage Jenkins'. The main section is titled 'Global Tool Configuration' with a wrench icon. It includes sections for 'Maven Configuration', 'OpenShift Client Tools', 'JDK', 'Git', and 'Mercurial'. The 'Git' section is expanded, showing an 'Add Git' button and a configuration for a 'git' installation with 'Name' set to 'Default' and 'Path to Git executable' set to 'git'. A checkbox for 'Install automatically' is also present. Other sections like Maven and Mercurial show their respective installation lists and 'Add' buttons.

Buscamos **Maven** y lo agregamos:

This screenshot shows the 'Maven' section of the Jenkins Global Tool Configuration. It displays a list of 'Maven installations' and an 'Add Maven' button. The 'Add Maven' button is highlighted with a red rectangle.

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



The screenshot shows the Jenkins configuration interface for adding a Maven installation. In the 'Maven' section, under 'Maven installations', there is a form for adding a new Maven instance. The 'Name' field contains 'maven-user1' and is highlighted with a red box. Below it, a checkbox labeled 'Install automatically' is checked, with a descriptive note explaining its function. Further down, there is an 'Install from Apache' section with a dropdown menu set to '3.6.3'. In the 'Docker' section, there are 'Docker installations' and 'Add Docker' buttons. At the bottom, there are 'Save' and 'Apply' buttons, with the 'Save' button also highlighted with 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.

7.7 Desde la linea de comandos de oc, y logueados a la misma, creamos el proyecto: "user1-cicd-tareas"

```
[default@wetty-1-9nclg ~]$ oc new-project user1-cicd-tareas
Now using project "user1-cicd-tareas" on server
"https://master.medellin-9f63.open.redhat.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.[default@wetty-1-9nclg ~]\$

7.8 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"

```
[default@wetty-1-9nclg ~]$ oc new-app jboss-eap71-openshift:1.3
https://github.com/redhat-gpte-devopsautomation/openshift-tasks
--> Found image 420be76 (10 months old) in image stream "openshift/jboss-eap71-
openshift" under tag "1.3" for "jboss-eap71-openshift:1.3"
JBoss EAP 7.1
-----
Platform for building and running JavaEE applications on JBoss EAP 7.1
Tags: builder, javaee, eap, eap7
* The source repository appears to match: jee
* A source build using source code from https://github.com/redhat-gpte-
devopsautomation/openshift-tasks will be created
  * The resulting image will be pushed to image stream tag "openshift-tasks:latest"
  * Use 'start-build' to trigger a new build
  * This image will be deployed in deployment config "openshift-tasks"
  * Ports 8080/tcp, 8443/tcp, 8778/tcp will be load balanced by service "openshift-
tasks"
  * Other containers can access this service through the hostname "openshift-tasks"
--> Creating resources ...
imagestream.image.openshift.io "openshift-tasks" created
buildconfig.build.openshift.io "openshift-tasks" created
deploymentconfig.apps.openshift.io "openshift-tasks" created
service "openshift-tasks" created
--> Success
Build scheduled, use 'oc logs -f bc/openshift-tasks' 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/openshift-tasks'
```

```
Run 'oc status' to view your app.  
[default@wetty-1-9nclg ~]$
```

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

```
[default@wetty-1-9nclg ~]$ oc expose svc openshift-  
tasksroute.route.openshift.io/openshift-tasks exposed
```

7.10 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.

```
[default@wetty-1-9nclg ~]$ oc set triggers dc openshift-tasks --  
manualdeploymentconfig.apps.openshift.io/openshift-tasks triggers  
updated[default@wetty-1-9nclg ~]$
```

7.11 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 utiizando el proyecto correspondiente, se subrayan en negro para su atención.

```
oc policy add-role-to-user edit system:serviceaccount:ci-cd-user1:jenkins -n user1-cicd-tareas
```

```
[default@wetty-1-9nclg ~]$ oc policy add-role-to-user edit system:serviceaccount:ci-cd-user1:jenkins -n user1-cicd-tareas  
role "edit" added: "system:serviceaccount:ci-cd-user1:jenkins"  
[default@wetty-1-9nclg ~]$
```

7.12 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 dropdown menus for 'All' and '+' buttons. Below the search bar is a table header with columns 'S', 'W', and 'Name' (sorted by Name). Underneath the table, it says 'Icon: S M L'. A table row is partially visible with icons for a grey circle, a yellow sun, and the text 'OpenShift Sa...'. At the bottom of the dashboard, there are two sections: 'Build Queue' (which says 'No builds in the queue.') and 'Build Executor Status' (which lists four idle executors: 1 Idle, 2 Idle, 3 Idle, 4 Idle).

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

Enter an Item name: Tasks

Seleccionar “Pipeline” para el tipo de job.

Y le damos click en “Ok

The screenshot shows the Jenkins 'New Item' configuration page. The 'Item name' field is filled with 'Tasks' and has a red border around it. Below the field, a note says '» Required field'. A red box highlights the 'Pipeline' project type, which is described as 'Orchestrates long-running activities that can span multiple build agents. Suitable for CI/CD pipelines'. Other project types listed include 'Freestyle project', 'Multi-configuration project', 'Bitbucket Team/Project', 'Folder', 'GitHub Organization', 'Multibranch Pipeline', and 'External Job'. At the bottom, there's a note: 'If you want to create a new item from other existing, you can use this option:' followed by a 'Copy from' button and an 'OK' button.

Enter an item name

Tasks

» Required field

Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining

Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable

Multi-configuration project
Suitable for projects that need a large number of different configurations, such

Bitbucket Team/Project
Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories

Folder
Creates a container that stores nested items in it. Useful for grouping things to

as long as they are in different folders.

GitHub Organization
Scans a GitHub organization (or user account) for all repositories matching so

Multibranch Pipeline
Creates a set of Pipeline projects according to detected branches in one SCM

External Job
This type of job allows you to record the execution of a process run outside Jenkins. See [the documentation for more details](#).

If you want to create a new item from other existing, you can use this option:

Copy from

OK

Type to autocomplete

7.14 Pegar el siguiente contenido en la parte de Pipeline script.

```
node {  
    stage('Build Tasks') {  
        openshift.withCluster() {  
            openshift.withProject("user1-cicd-tareas") {  
                openshift.selector("bc", "openshift-tasks").startBuild("--wait=true")  
            }  
        }  
    }  
    stage('Tag Image') {  
        openshift.withCluster() {  
            openshift.withProject("user1-cicd-tareas") {  
                openshift.tag("openshift-tasks:latest", "openshift-tasks:${BUILD_NUMBER}")  
            }  
        }  
    }  
    stage('Deploy new image') {  
        openshift.withCluster() {  
            openshift.withProject("user1-cicd-tareas") {  
                openshift.selector("dc", "openshift-tasks").rollout().latest();  
            }  
        }  
    }  
}
```

Asegúrese de que el project/namespace “user1-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.

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

The screenshot shows a pipeline configuration interface. At the top, there's a header with tabs for "Pipeline" and "Definition". Below that, the "Pipeline script" tab is selected, showing a Groovy 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    openshift.withCluster() {  
18      openshift.withProject("user1-cicd-tareas") {  
19        openshift.selector("bc", "openshift-tasks").startBuild("--wait=true")  
20      }  
21    }  
22  }
```

Below the script, there's a checkbox labeled "Use Groovy Sandbox" which is checked. Underneath the script area, there's a link labeled "Pipeline Syntax".

At the bottom of the interface, there are two buttons: a dark green "Save" button and a light blue "Apply" button.

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

The screenshot shows the Jenkins Pipeline Tasks page. At the top left is the Jenkins logo and the word "Jenkins". Below it, the navigation path is "Jenkins > Tasks >". On the left side, there is a sidebar with several options: "Back to Dashboard", "Status", "Changes", "Build Now" (which is highlighted with a red box and a red arrow pointing to it), "Delete Pipeline", "Configure", "Full Stage View", "Open Blue Ocean", "Rename", and "Pipeline Syntax". To the right of the sidebar, the main content area has a title "Pipeline Tasks" in a large, bold font. Below it is a "Stage View" section with the message "No data available. This Pipeline has no stages defined." At the bottom left, there is a "Build History" section with a search bar containing "find" and two RSS feed links: "RSS for all" and "RSS for failures". To the right of the "Build History" section is a "Permalinks" section.

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

 Jenkins

Jenkins ▾ > Tasks ▾

[Back to Dashboard](#)

[Status](#)

[Changes](#)

[Build Now](#)

[Delete Pipeline](#)

[Configure](#)

[Full Stage View](#)

[Open Blue Ocean](#)

[Rename](#)

[Pipeline Syntax](#)

Pipeline Tasks

 Recent Changes

Stage View

Average stage times:
(Average full run time: ~2min 48s)

Build Tasks	Tag Image	Deploy new image
2min 27s	3s	1s
2min 27s	3s	1s

#1 Nov 30, 2019 10:05 PM

No Changes

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

Permalinks

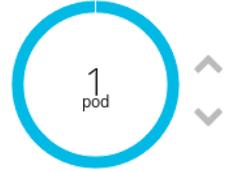
7.16 Ya con el build realizado de forma correcta, ingresamos a la consola web de Openshift, abrimos el proyecto donde hicimos el deploy, en este caso: "user1-cicd-tareas".

Para ingresar al URL, dentro de la parte de deployment config, hay una parte de rutas, en esa debe estar el URL de acceso a la aplicación

The screenshot shows the OpenShift web interface for the project "user1-cicd-tareas". The left sidebar includes links for Overview, Applications, Builds, Resources, Storage, Monitoring, and Catalog. The main content area displays the "openshift-tasks" application under the APPLICATION section. Key details shown include:

- DEPLOYMENT CONFIG:** openshift-tasks, #2
- CONTAINERS:** openshift-tasks
 - Image: user1-cicd-tareas/openshift-tasks 1455b13 633.2 MiB
 - Build: openshift-tasks, #2
 - Source: Merge pull request #1 from jeichler/https 308f21f
 - Ports: 8080/TCP and 2 others
- NETWORKING:** Service - Internal Traffic
 - openshift-tasks
 - 8080/TCP (8080-tcp) → 8080 and 2 others
- BUILDS:** openshift-tasks

<http://openshift-tasks-user1-cicd-tareas.apps.medellin-9f63.open.redhat.com>



Routes - External Traffic

<http://openshift-tasks-user1-cicd-tareas.apps.medellin-9f63.open.redhat.com>

Route openshift-tasks, target port 8080-tcp

✓ Build #2 is complete created 5 minutes ago

[View Full Log](#)

```
Cloning "https://github.com/redhat-gpte-devopsautomation/openshift-tasks" ...
Commit: 308f21ff69fa357410b4b64db73b99877b115009 (Merge pull request #1 from jeichler/https)
Author: Wolfgang Kulhanek <wkulhanek@users.noreply.github.com>
Date: Tue Aug 20 03:01:58 2019 +0900
'/tmp/src/configuration/application-roles.properties' -> '/opt/eap/standalone/configuration/application-roles.pr...
'/tmp/src/configuration/application-users.properties' -> '/opt/eap/standalone/configuration/application-users.pr...
Pushing image docker-registry.default.svc:5000/user1-cicd-tareas/openshift-tasks:latest ...
Pushed 6/7 layers, 87% complete
Pushed 7/7 layers, 100% complete
Push successful
```

7.17 Abrimos la página, desde la ruta que se muestra en el proyecto “cicd-tareas, en este caso sería: <http://openshift-tasks-user1-cicd-tareas.apps.medellin-9f63.open.redhat.com/>

The screenshot shows the OpenShift Task console interface. It consists of five main 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" (red).
- Info:** A table with the following data:

Pod Hostname	openshift-tasks-2-lrps6
Pod IP	null
Used Memory	184 MB
Session ID	0PXxhr0CMwlP35T_zSC1ltNBrGfM2kqTLc7XLu7s
- Messages:** Displays the message "Nothing to report."