



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.

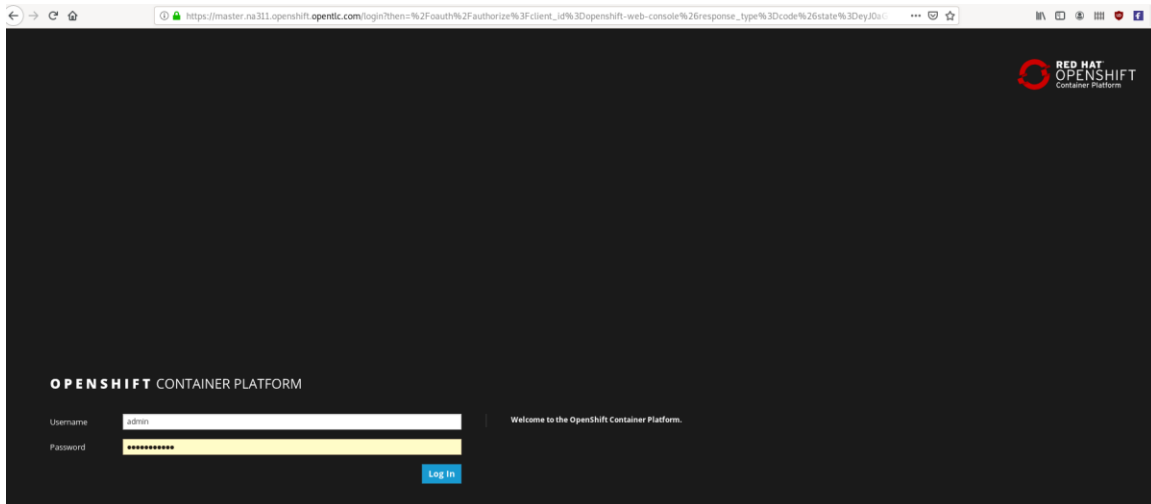
El usuario debe haber realizado los prerequisites.

Solicitar URL del Openshift al presentador.

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

Usuario: (pedir usuario)

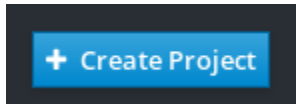
Password: openshift



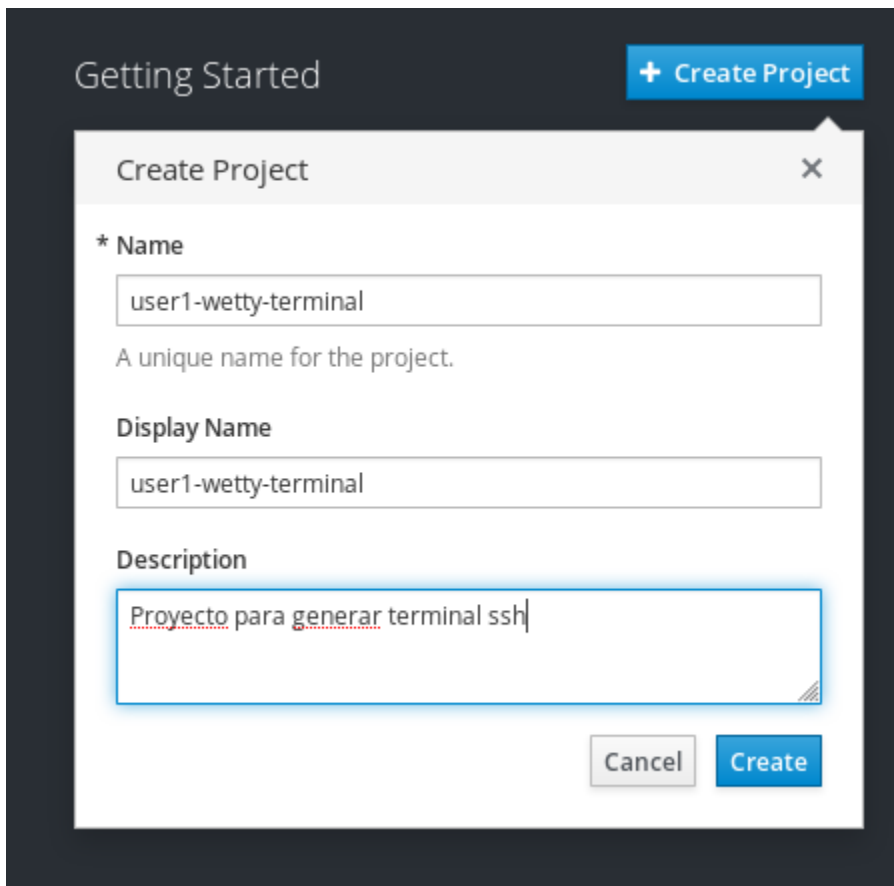
2. Generación de terminal de línea de comandos por medio de un container web tty de NodeJs, necesaria para realizar el laboratorio.

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.



Getting Started + Create Project

Create Project ×

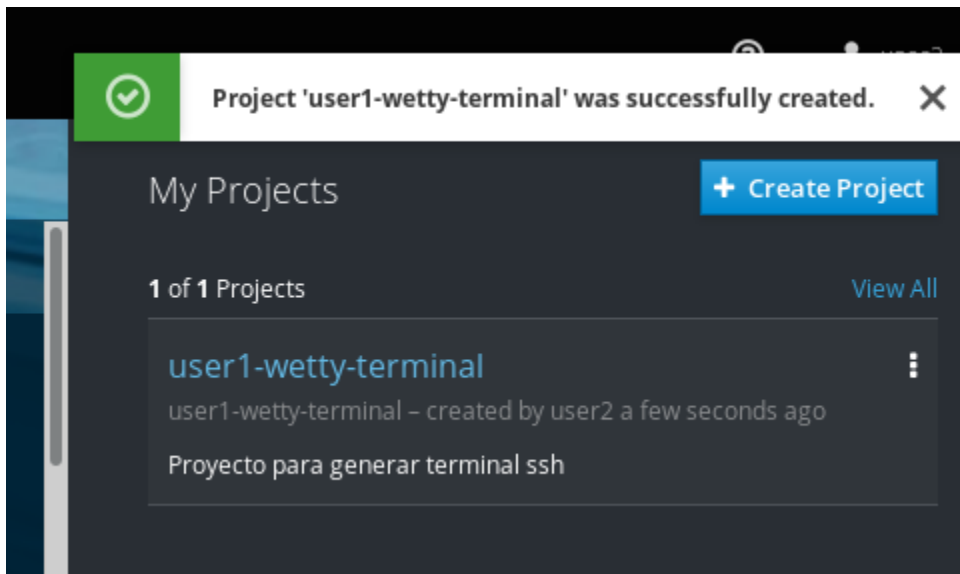
* Name
user1-wetty-terminal
A unique name for the project.

Display Name
user1-wetty-terminal

Description
Proyecto para generar terminal ssh

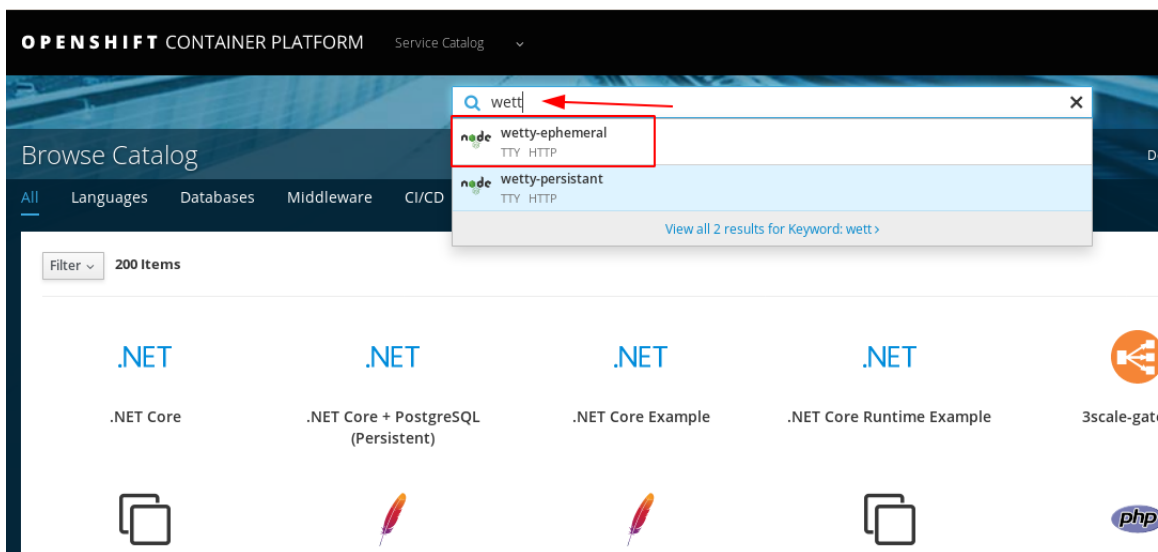
Cancel Create

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:

The screenshot shows a window titled "wetty-ephemeral" with a close button (X) in the top right corner. Below the title bar is a progress bar with four steps: "Information" (1), "Configuration" (2), "Binding" (3), and "Results" (4). The "Information" step is currently selected and highlighted with a blue circle. Below the progress bar, the main content area displays the "node" logo, the text "wetty-ephemeral", and "TTY HTTP". Underneath, it says "Default plan" and "Web Based Terminal Interface (Ephemeral)". At the bottom right of the window, there are three buttons: "Cancel", "< Back", and "Next >". The "Next >" button is highlighted in blue.

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.

The screenshot shows a configuration window titled "wetty-ephemeral" with a close button (X) in the top right corner. At the top, there is a progress bar with four steps: "Information" (1), "Configuration" (2, currently active), "Binding" (3), and "Results" (4). Below the progress bar, the "Configuration" section contains several fields:

- * Add to Project:** A dropdown menu showing "user1-wetty-terminal".
- * Name:** A text input field containing "wetty". Below it, a note reads: "The name assigned to all of the frontend objects defined in this template."
- * Namespace:** A text input field containing "openshift". Below it, a note reads: "The OpenShift Namespace where the ImageStream resides."
- * Git Repository URL:** A text input field containing "https://github.com/kevensen/wetty.git". Below it, a note reads: "The URL of the repository with your application source code."
- Git Reference:** An empty text input field. Below it, a note reads: "Set this to a branch name, tag or other ref of your repository if you are not using the default branch."

At the bottom right of the window, there are three buttons: "Cancel", "< Back", and "Next >". The "Next >" button is highlighted in blue.

Information

1

Configuration

2

Binding

3

Results

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

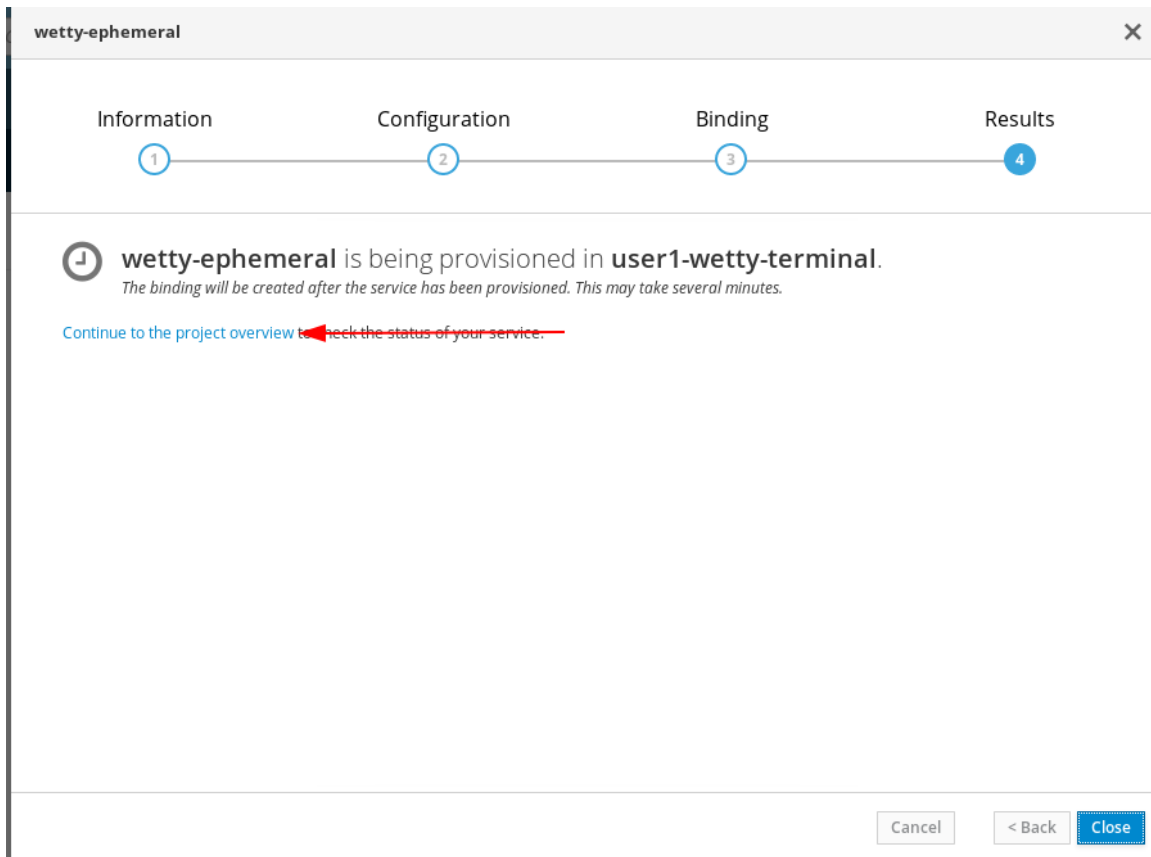
Bindings can be created later from within a project.

Cancel

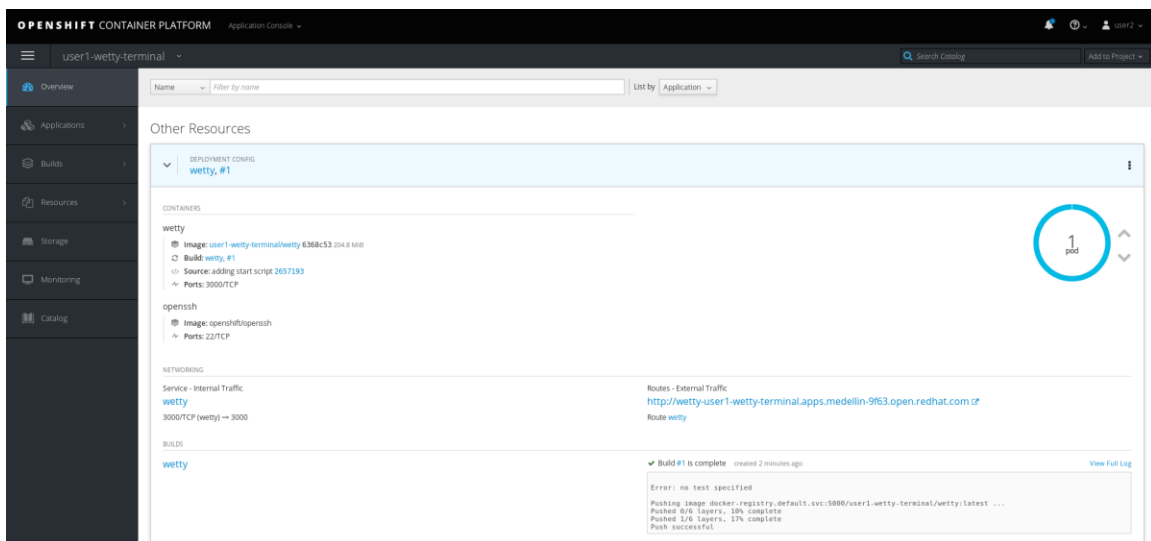
< Back

Create

Click en “Continue to the project overview” link.

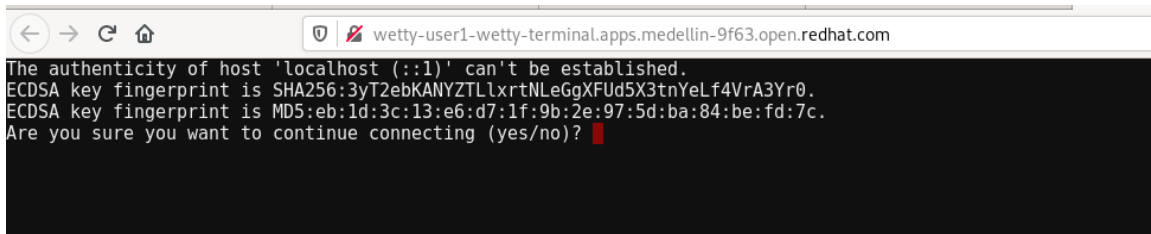


Esperamos que el deployment config esté con 1 pod “Running”, como se muestra a continuación.



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.



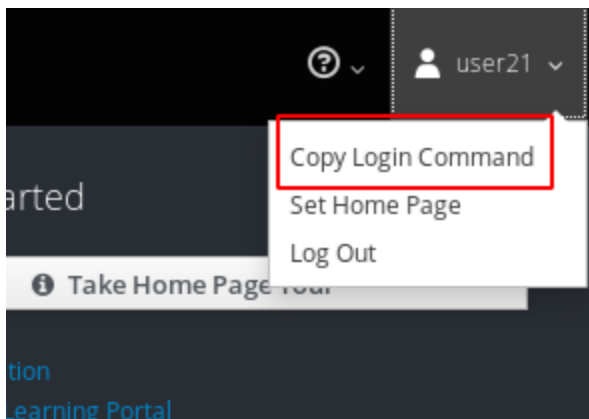
Aceptar la conexión e ingresar el password ssh:

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 debería mostrar un mensaje como el siguiente:

```
[default@wetty-1-9nclg ~]$ oc login https://master.medellin-9f63.open.redhat.com:443 --token=djwLlvU4uWvIMKbej5KslZpWwetCuGbCmiSUQL9EzC4
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 línea de comandos de OC.

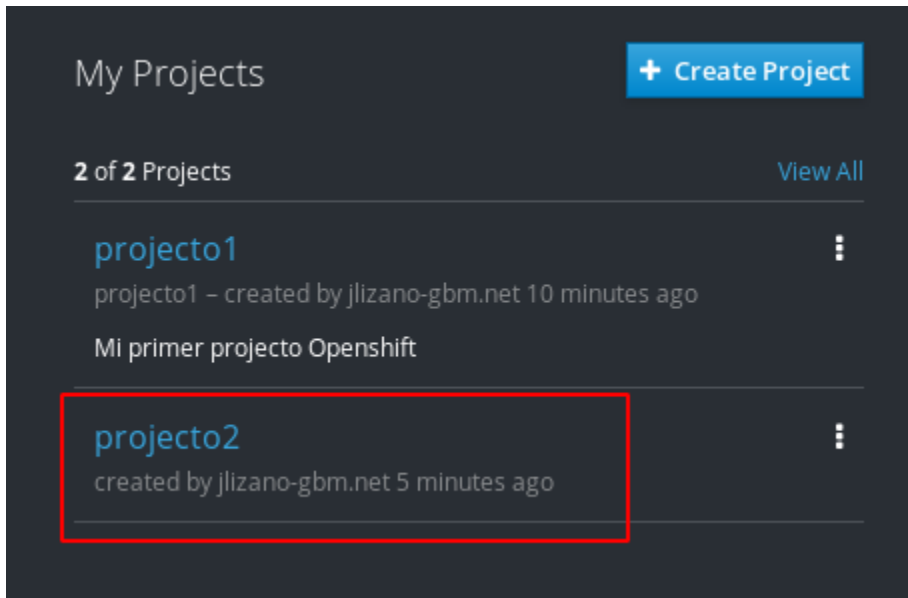
3.1 Ya logueados en la consola de OC, creamos el nuevo proyecto con el nombre “proyecto2”, de la siguiente forma, debería dar el siguiente output.

```
$ oc new-project userx-projecto2
Now using project "projecto2" 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 “**userx-projecto2**” creado desde la línea de comandos oc.

```
$ oc get projects
NAME          DISPLAY NAME   STATUS
projecto1     proyecto1      Active
user-2projecto2                Active
```

3.3 También lo podemos verificar en la consola web.



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

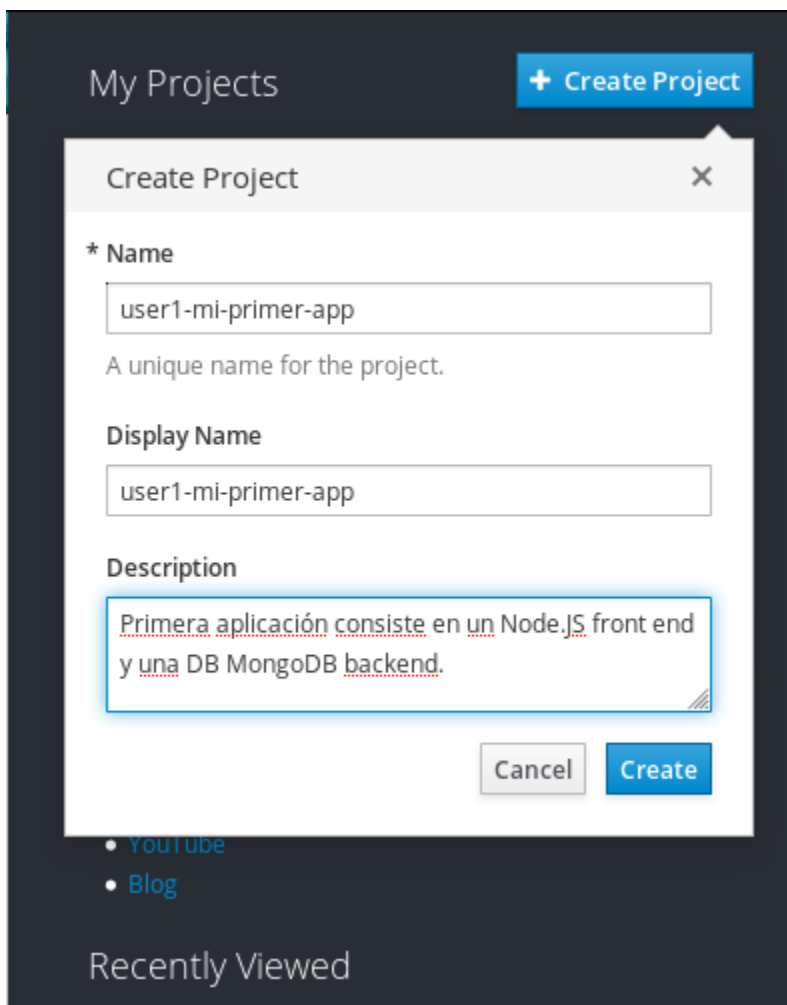
4.1 Creamos un proyecto:

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

Project Name: **userx-mi-primer-app**

Project Display Name: **userx-Mi primer app**

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



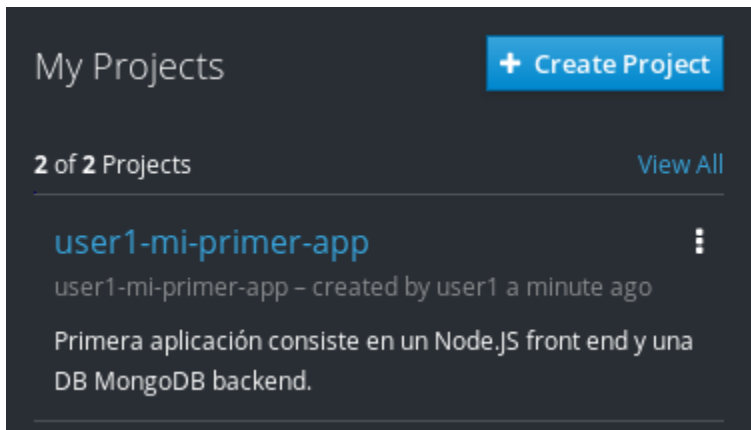
The screenshot shows the 'My Projects' interface with a '+ Create Project' button in the top right. A modal dialog titled 'Create Project' is open, containing the following fields:

- * Name:** A text input field containing 'user1-mi-primer-app'.
- Display Name:** A text input field containing 'user1-mi-primer-app'.
- Description:** A text area containing the text 'Primera aplicación consiste en un Node.JS front end y una DB MongoDB backend.'

At the bottom of the dialog are two buttons: 'Cancel' and 'Create'.

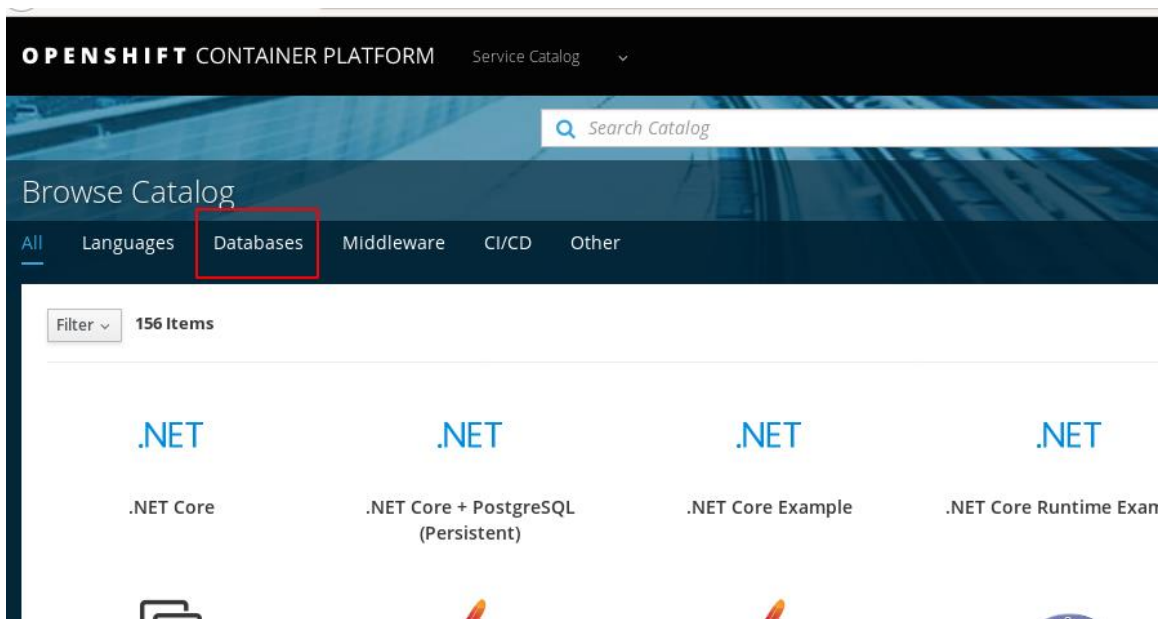
Y le damos “**Create**”


Debe mostrarse de esta forma:



4.2 Desplegar una base de datos **MongoDB (Ephemeral)**.

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



 Search Catalog

Browse Catalog

All Languages **Databases** Middleware CI/CD Other

All



Mongo



MySQL

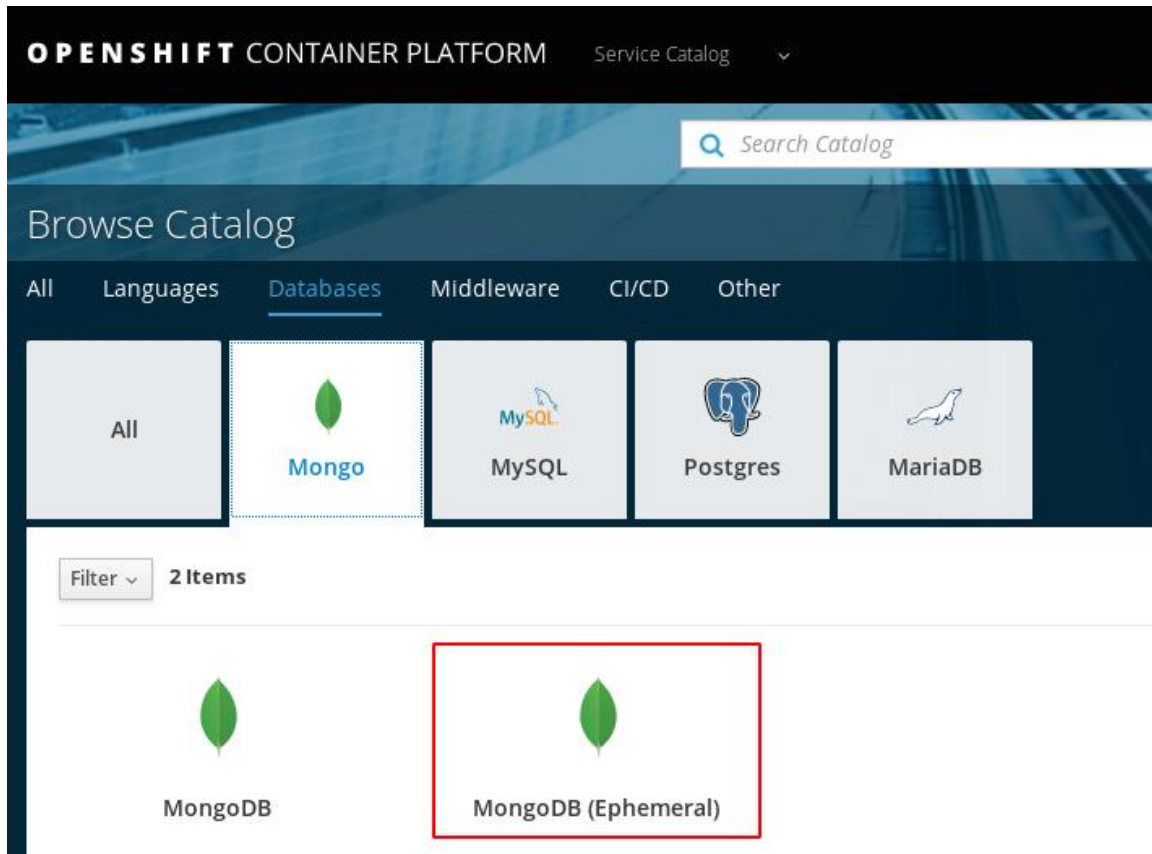


Postgres



MariaDB

4.3 Escogemos Mongo y luego **MongoDB (Ephemeral)**.



4.4 Y le damos next:

MongoDB (Ephemeral)

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 Le agregamos los siguientes datos:

MongoDB (Ephemeral)X

Information

Configuration

Binding

Results

1

2

3

4

* Add to Project

user1-mi-primer-app

* Memory Limit

512Mi

Maximum amount of memory the container can use.

Namespace

openshift

The OpenShift Namespace where the ImageStream resides.

* Database Service Name

mongodb

The name of the OpenShift Service exposed for the database.

MongoDB Connection Username

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

Cancel

< Back

Next >

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

Y le damos “**Create**”:

MongoDB (Ephemeral)

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 **user1-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

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

MongoDB (Ephemeral)

Information Configuration Binding Results

1 2 3 4

✓ MongoDB (Ephemeral) has been added to **user1-mi-primer-app** successfully.

✓ The binding **mongodb-ephemeral-zzvf7-j7q2h** has been created successfully.
The binding operation created the secret [mongodb-ephemeral-zzvf7-credentials-7167t](#) that you may need to reference in your application.

[Continue to the project overview.](#)

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

1 pod

4.7 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.

Other Resources

DEPLOYMENT CONFIG

mongodb, #1

CONTAINERS

mongodb

Image: openshift/mongodb

Ports: 27017/TCP

NETWORKING

Service - Internal Traffic

mongodb

27017/TCP (mongo) → 27017

CONTAINERS

mongodb

Image: openshift/mongodb

Ports: 27017/TCP

NETWORKING

Service - Internal Traffic

mongodb

27017/TCP (mongo) → 27017


Routes - External Traffic

Create Route

1 pod

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

Provisioned Services

 **MongoDB (Ephemeral)**
mongodb-ephemeral-zzvf7

MongoDB database service, without persistent storage. For more information about [container/blob/master/3.2/README.md](#).


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

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

BINDINGS

mongodb-ephemeral-zzvf7-j7q2h
created 4 minutes ago

[Delete](#) | [View Secret](#)

 [Create Binding](#)

4.9 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-zzvf7-credentials-7l67t

mongodb-ephemeral-zzvf7-credentials-7l67t created 4 minutes

Opaque [Hide Secret](#)

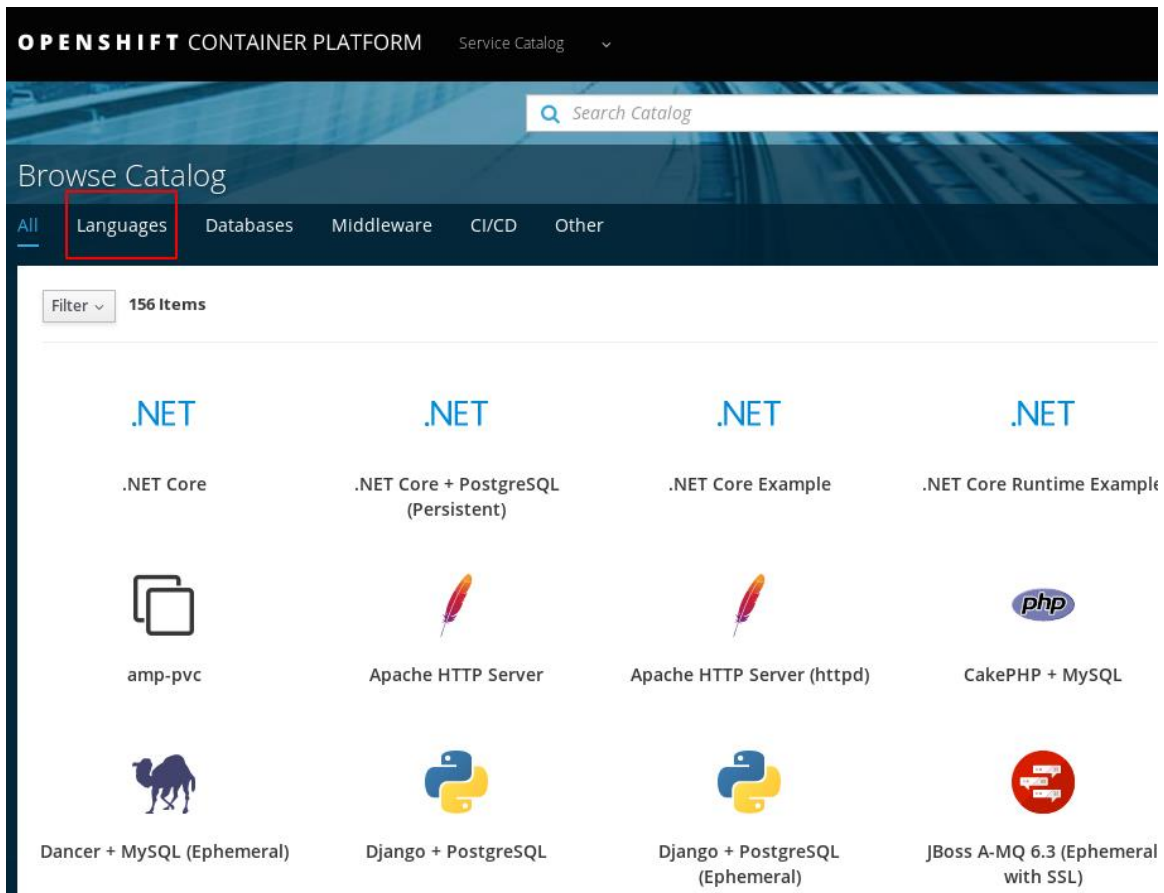
admin_password	LCibcgvSBgP002c5
database_name	sampledb
password	DRLE8tNSIP6VPKTI
uri	mongodb://172.30.65.147:27017
username	userG66

There are no annotations on this resource.

4.10 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**”.



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

The screenshot shows the 'Browse Catalog' interface. At the top, there are tabs for 'All', 'Languages', 'Databases', 'Middleware', 'CI/CD', and 'Other'. The 'Languages' tab is selected and highlighted with a red box. Below the tabs, there are icons for various languages: 'All', 'Java', 'JavaScript' (highlighted with a red box), '.NET', 'Perl', and 'Ruby'. Below the language icons, there is a 'Filter' dropdown and a '3 Items' indicator. Three items are listed: 'Node.js' (highlighted with a red box), 'Node.js + MongoDB', and 'Node.js + MongoDB (Ephemeral)'. Below the list, there is a progress bar with three steps: 'Information' (1), 'Configuration' (2), and 'Results' (3). The 'Information' step is currently active. Below the progress bar, the 'Node.js' section is displayed, including the Node.js logo, the text 'Node.js BUILDER NODEJS', and a description: '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>'. At the bottom right, there are three buttons: 'Cancel', '< Back', and 'Next >' (highlighted with a red box).

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 >

4.12 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**”

Node.js

Information

Configuration

Results

1

2

3

* Add to Project

user1-mi-primer-app

Version

10 — latest

* Application Name

miprimerapp

* Git Repository

https://github.com/openshift/nodejs-ex.git

Try Sample Repository ↗

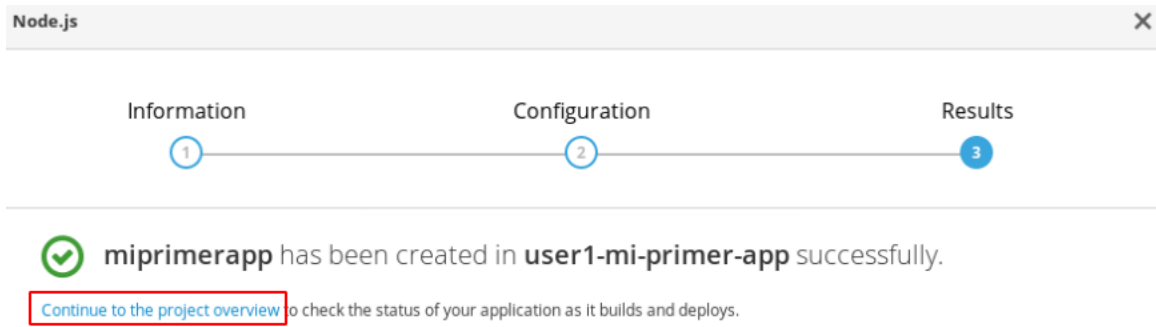
If you have a private Git repository or need to change application defaults, view [advanced options](#).

Cancel

< Back

Create

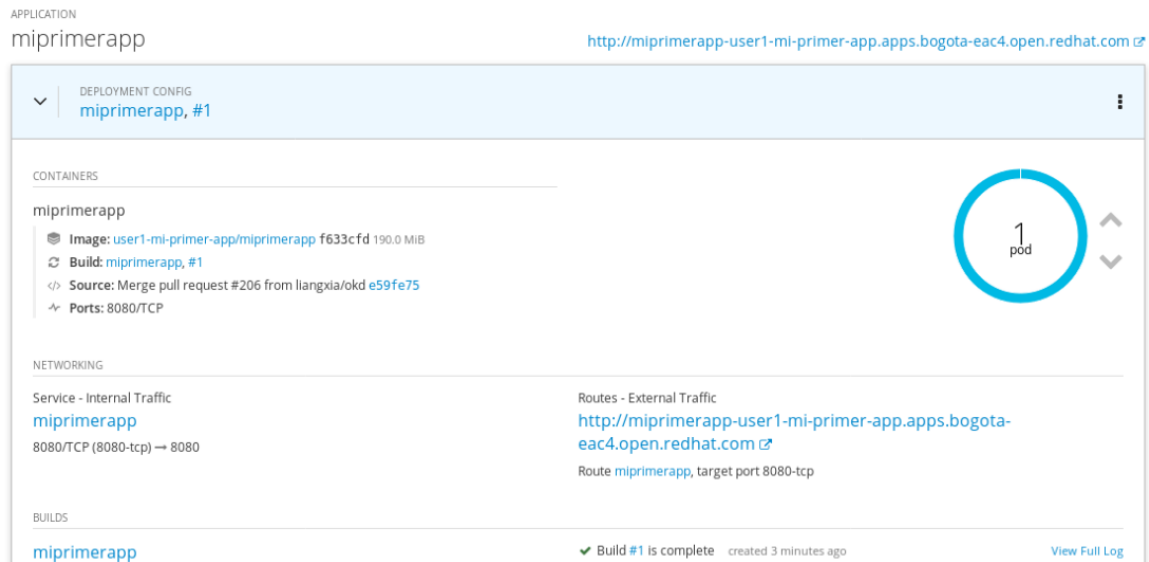
4.13 Desde la página de resultados, click en “Continue to the project overview”.



4.14 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-user1-mi-primer-app.apps.bogota-eac4.open.redhat.com> .



APPLICATION

miprimerapp







DEPLOYMENT CONFIG

miprimerapp, #1

CONTAINERS

miprimerapp

-  **Image:** [user1-mi-primer-app/miprimerapp f633cfd](#) 190.0 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

4.15 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 Heroku dashboard for a user named 'user1-mi-primer-app'. The left sidebar contains navigation links: Overview, Applications (selected), Builds, Resources, Storage, Monitoring, and Catalog. The main content area shows the 'Deployments' page for the application 'miprimerapp', which was created 6 minutes ago. Below this, there are tabs for History, Configuration, Environment, and Events. The 'History' tab is active, showing a deployment labeled '#1' which is active and was created 5 minutes ago. A 'View Log' link is provided for this deployment. Below the deployment information, there is a table with a filter by label input and a table of deployment history.

Deployment	Status
#1 (latest)	Active, 1 replica

4.16 Ingresamos a “Environment” dentro de “miprimerapp”.

[Deployments](#) > **miprimerapp**

miprimerapp created 7 minutes ago

app miprimerapp

History Configuration **Environment** Events

Container miprimerapp

Name	Value
<input type="text" value="Name"/>	<input type="text" value="Value"/>

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

Environment From ⓘ

Config Map/Secret	Prefix ⓘ
<input type="text" value="Select a resource"/>	<input type="text" value="Add prefix"/>

[Add ALL Values from Config Map or Secret](#)

4.17 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**.

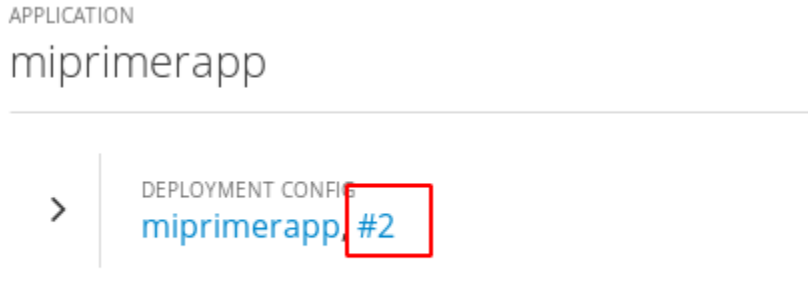
Y le damos click en “**Save**”.

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](#)

4.18 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.



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.

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

miprimerapp-mi-primer-app.apps.na311.openshift.opentlc.com

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>
```

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: 3

DB Connection Info:

Type: MongoDB
URL: mongodb://172.30.120.242:27017/sampledb

5. Crear una aplicación desde línea 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
```

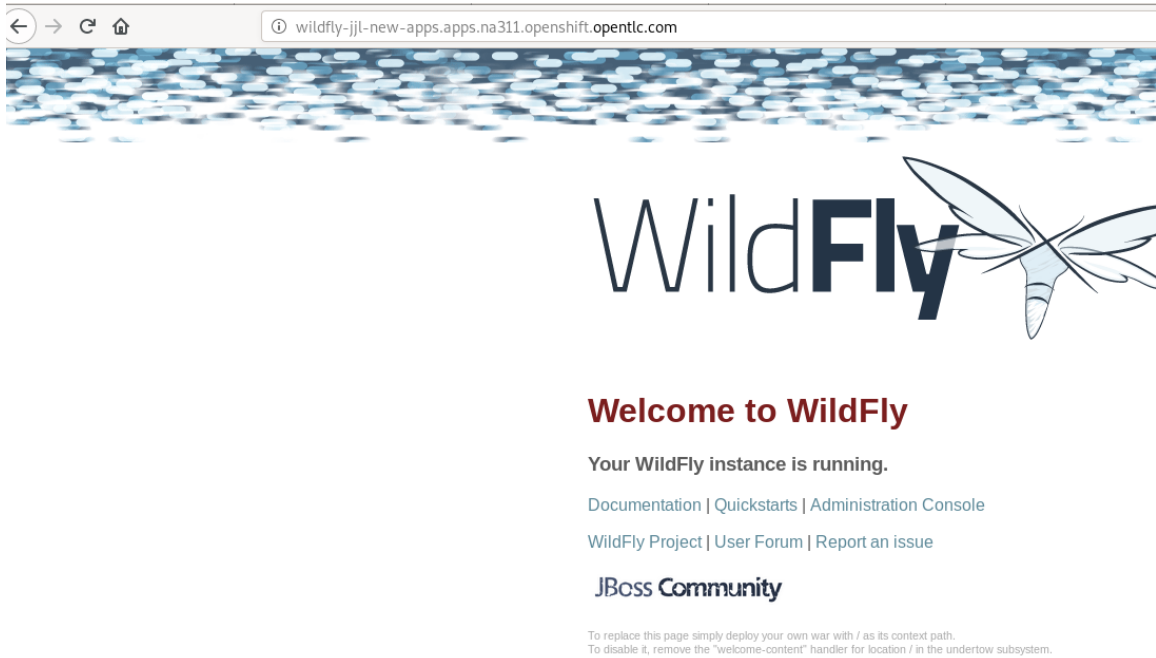
```
route.route.openshift.io/wildfly exposed
```

5.5 Verificar la ruta creada:

```
$ oc get route
```

NAME	HOST/PORT	PATH	SERVICES	PORT
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 “**userx-managing-apps**”.

```
$ oc new-project user1-managing-apps
```

```
Now using project "user1-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 acceder 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	1	config,image(cotd:latest)

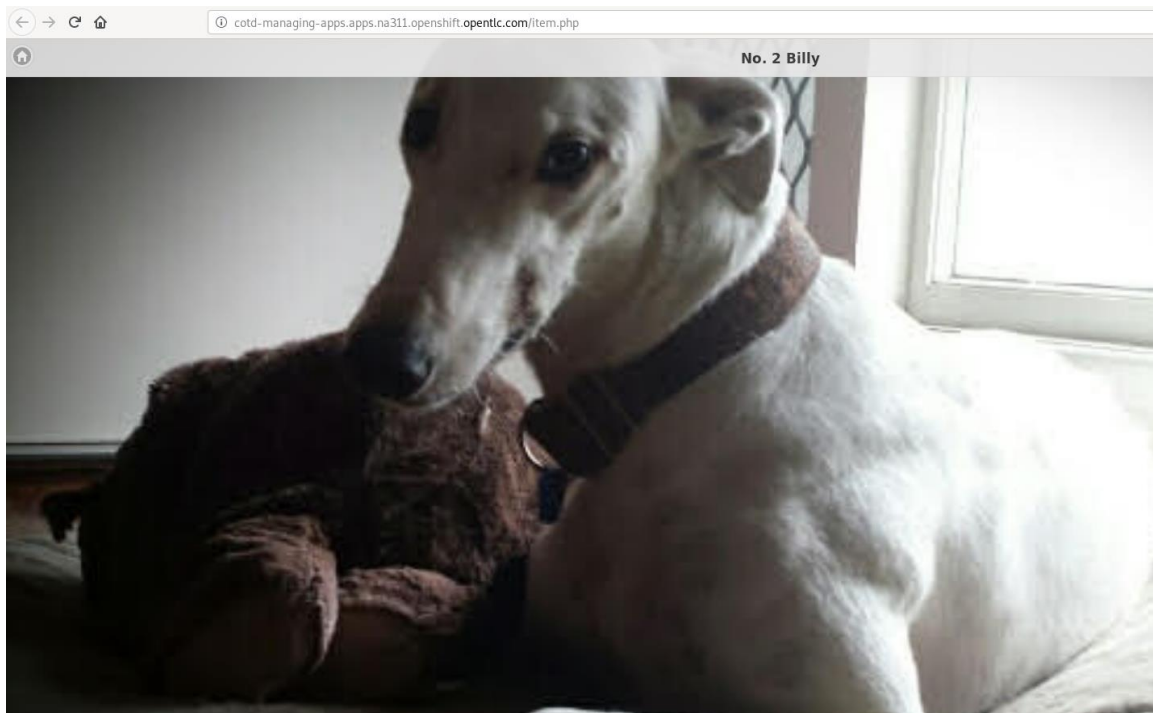
NAME	TYPE	FROM	LATEST
buildconfig.build.openshift.io/cotd	Source	Git	1

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

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

NAME	HOST/PORT	PATH	SERVICES	PORT
route.route.openshift.io/cotd	cotd-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-sqt7	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 enrutamiento A / B se usa con mayor frecuencia para las pruebas de la interfaz de usuario, por ejemplo, para determinar qué versión de un sitio web genera más conversiones de ventas.

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"

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 ctd --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.

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

Create Project:

Project Name: userx-ci-cd

Project Display Name: user1-ci-cd

Description: Proyecto para instalacion de Jenkins

My Projects

+ Create Project

Create Project X

* Name

user1-ci-cd

A unique name for the project.

Display Name

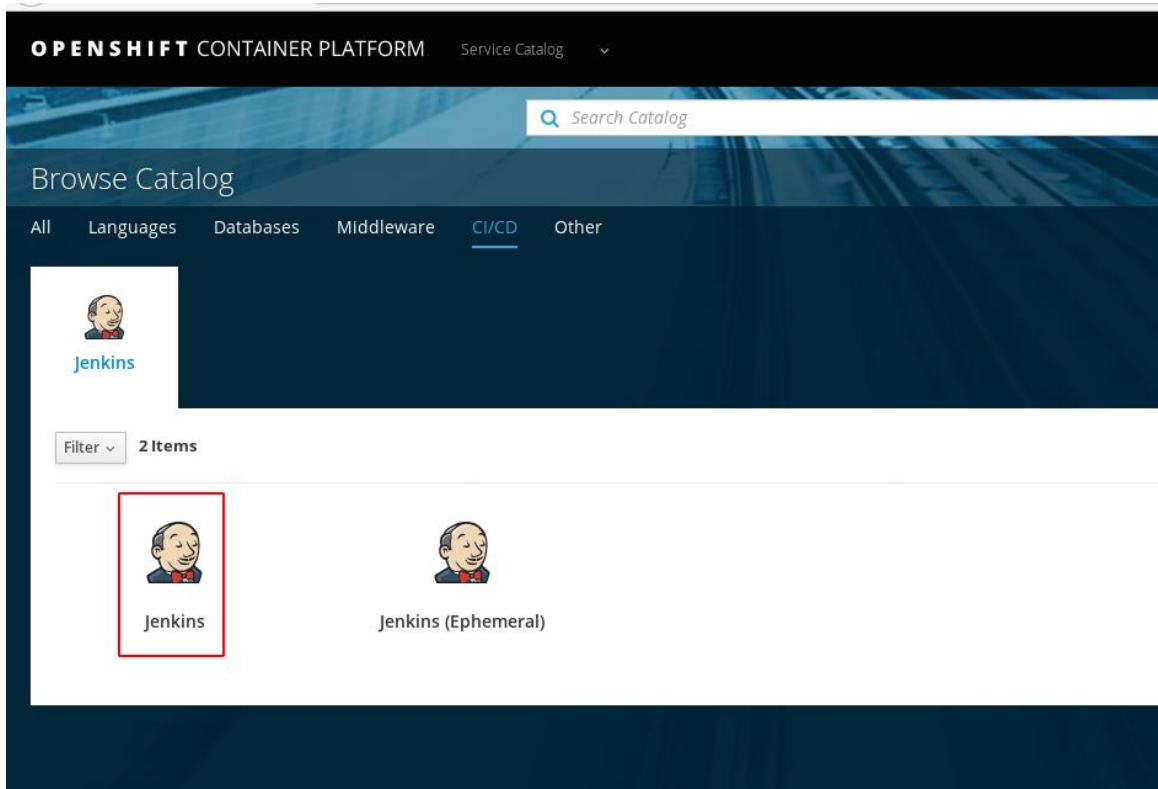
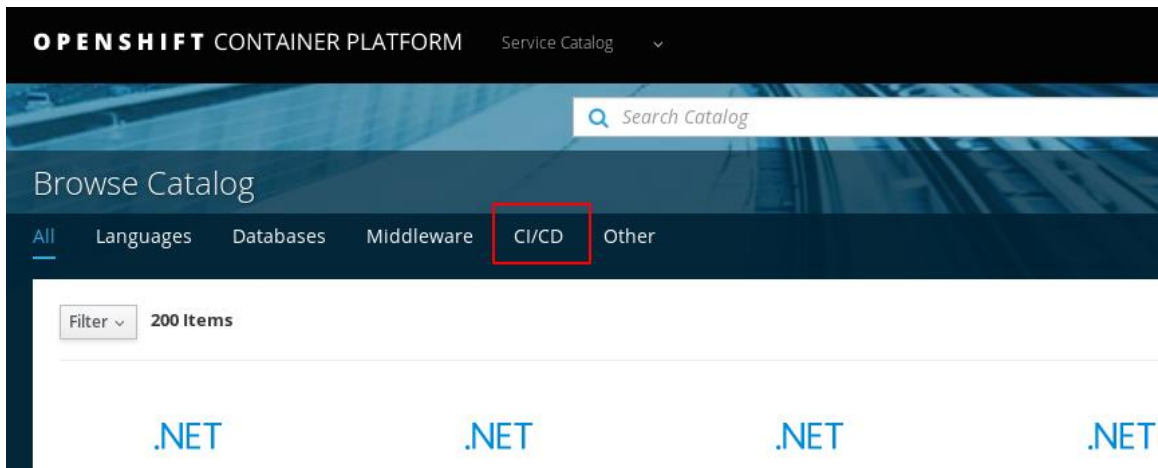
user1-ci-cd

Description

Proyecto para instalacion de Jenkins

Cancel Create

7.3 Ir al catálogo, click en CI/CD, y click en Jenkins.



Le damos click en “Next”.

Information

1

Configuration

2

Binding

3

Results

4



Jenkins

Red Hat, Inc.

INSTANT-APP JENKINS

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

Default plan

Jenkins service, with persistent storage.

NOTE: You must have persistent volumes available in your cluster to use this template.

This template deploys a Jenkins server capable of managing OpenShift Pipeline builds and supporting OpenShift-based oauth login.

[Cancel](#)[< Back](#)[Next >](#)

7.4 Se debe completar el despliegue del Jenkins y cambiar solamente los siguientes datos:

Add to Project : userx-ci-cd

Memory Limit: 2Gi

Disable memory intensive administrative monitors: true

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

Jenkins

Information

Configuration

Binding

Results

1

2

3

4

* Add to Project

user1-ci-cd

Jenkins Service Name

jenkins

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

Jenkins JNLP Service Name

jenkins-jnlp

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

Enable OAuth in Jenkins

true

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.

Cancel

< Back

Next >

Information

Configuration

Binding

Results

1

2

3

4

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

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.

Jenkins ImageStreamTag

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

Fatal Error Log File

When a fatal error occurs, an error log is created with information and the state obtained at the time of the fatal error.

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

Jenkins

Information

Configuration

Binding

Results

1

2

3

4

Create a binding for **Jenkins**

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

☐ Create a secret in **user1-ci-cd** 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

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

Jenkins

Information

Configuration

Binding


Results

1

2

3

4



Jenkins is being provisioned in **user1-ci-cd**.

This may take several minutes.

Continue to the project overview to check the status of your service.

Cancel

< Back

Close

7.6 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.

Name ▾Filter by name

APPLICATION
jenkins-persistent

▾DEPLOYMENT CONFIG
jenkins, #1

CONTAINERS
jenkins
Image: openshift/jenkins

NETWORKING
Service - Internal Traffic
jenkins
80/TCP (web) → 8080
Service - Internal Traffic
jenkins-jnlp
50000/TCP (agent) → 50000

Application

<https://jenkins-ci-cd-user1.apps.medellin-9f63.open.redhat.com>

1 pod

Routes - External Traffic
<https://jenkins-ci-cd-user1.apps.medellin-9f63.open.redhat.com>
Route jenkins

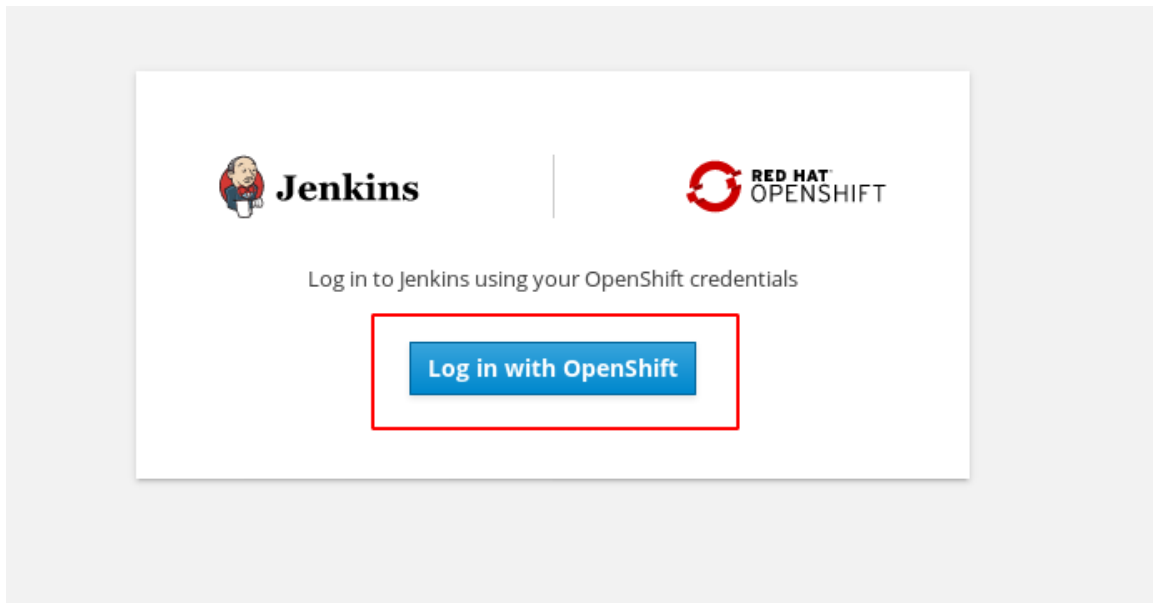
Routes - External Traffic
[Create Route](#)

Le damos click en la ruta de acceso de Jenkins.

<https://jenkins-ci-cd-user1.apps.medellin-9f63.open.redhat.com>

1 pod

7.7 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 **loguarnos** a la **consola web de openshift**.

Authorize Access

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










Requested permissions

- ☒ **user:info**
Read-only access to your user information (including username, identities, and group membership)
- ☒ **user:check-access**
Read-only access to view your privileges (for example, "can I create builds?")

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

Allow selected permissions

Deny

-  [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

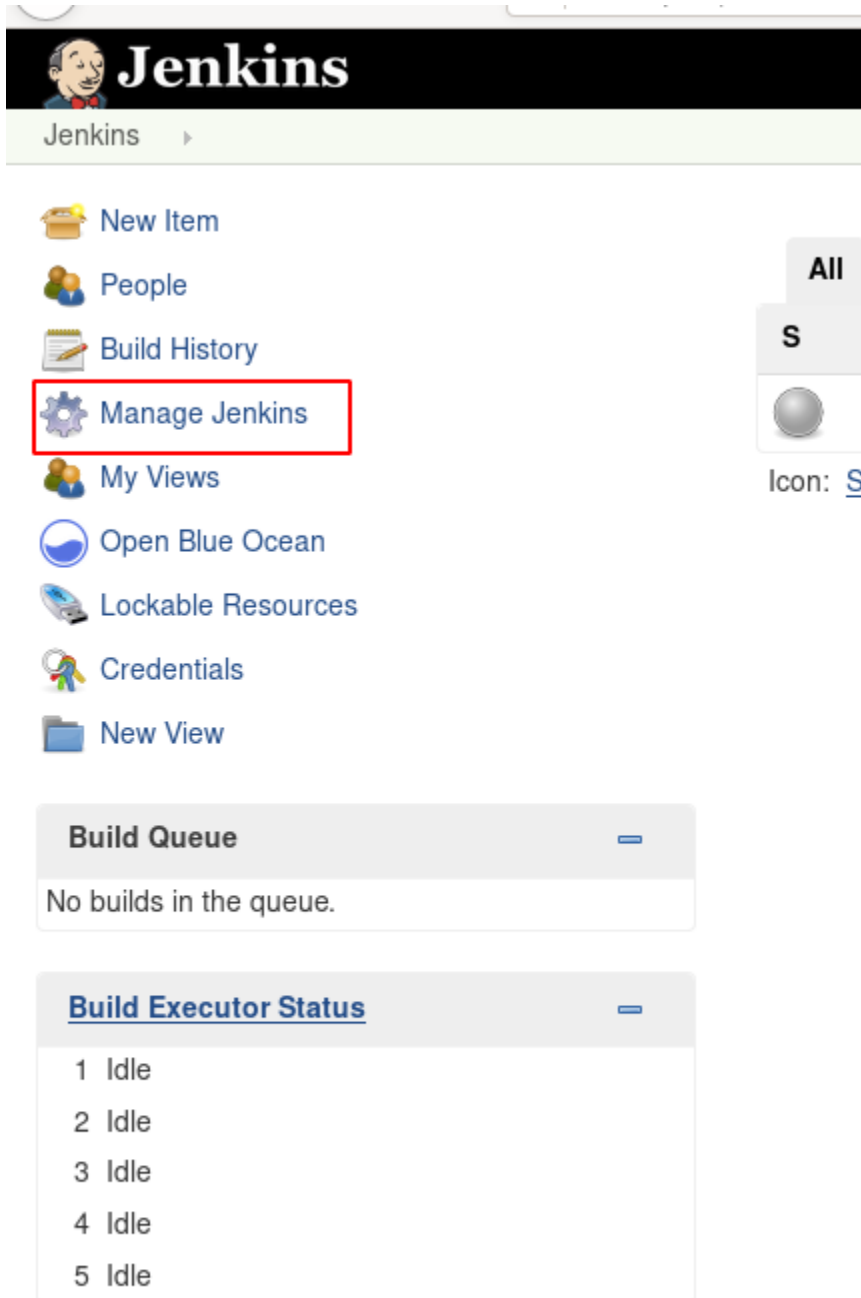
All +

S	W	Name ↓	Last Success
		OpenShift Sample	N/A


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

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


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





Le damos click en “Global Tool Configuration”


**Jenkins**


Jenkins ▾ ▶


 [New Item](#)


 [People](#)


 [Build History](#)


 **[Manage Jenkins](#)**


 [My Views](#)

 [Open Blue Ocean](#)


 [Lockable Resources](#)

 [Credentials](#)

 [New View](#)

Build Queue 

No builds in the queue.

Build Executor Status 

1	Idle
2	Idle
3	Idle
4	Idle
5	Idle

Manage Jenkins


Jenkins root URL is empty but is required for the proper operation of many Jenkins features like [Jenkins configuration](#). 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 behavior.


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


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


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

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

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

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

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

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

Vamos a ver una imagen como la siguiente.

The screenshot shows the Jenkins 'Global Tool Configuration' page. The header includes the Jenkins logo and navigation links like 'Back to Dashboard' and 'Manage Jenkins'. The main section is titled 'Global Tool Configuration' with a wrench icon. It contains several tool configuration sections: 'Maven Configuration' with buttons for 'Use default maven settings' and 'Use default maven global settings'; 'OpenShift Client Tools' with an 'Add OpenShift Client Tools' button; 'JDK' with an 'Add JDK' button; 'Git' with a form to add a new installation (Name: Default, Path: git, Install automatically: unchecked) and an 'Add Git' button; and 'Mercurial' with an 'Add Mercurial' button. Each section has a link to view the list of installations on the system.

Buscamos **Maven** y lo agregamos:

This close-up shows the 'Maven' section of the configuration page. It includes the text 'List of Ant installations on this system' above the section header. Below the 'Maven installations' header, the 'Add Maven' button is highlighted with a red rectangle. At the bottom, it says '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 configuration interface. At the top, there's a section for 'Maven' with a sub-header 'Maven installations'. Below this, there's an 'Add Maven' button. A table with one row is shown, with the 'Name' column containing 'maven-user1'. Below the table, there's a checkbox labeled 'Install automatically' which is checked. A text box explains that choosing this option lets Jenkins install the tool on demand and that checking it will lead to configuring a series of 'installers'. Below this, there's a section for 'Install from Apache' with a 'Version' dropdown set to '3.6.3'. Further down, there's an 'Add Installer' button. At the bottom of the Maven section, there's an 'Add Maven' button and a link to 'List of Maven installations on this system'. Below the Maven section is a section for 'Docker' with a sub-header 'Docker installations'. It has an 'Add Docker' button and a link to 'List of Docker installations on this system'. At the very bottom, there are two buttons: 'Save' and 'Apply'. The 'Save' button is highlighted with a red rectangle.

Maven

Maven installations

Add Maven

Maven
Name maven-user1

☒ Install automatically

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

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

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

Install from Apache

Version 3.6.3

Add Installer

Add Maven

List of Maven installations on this system

Docker

Docker installations

Add Docker

List of Docker installations on this system

Save Apply

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

7.9 Desde la línea 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 ~]$
```

7.10 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"**

```
[default@wetty-1-9nclg ~]$ oc new-app jboss-eap71-openshift:1.3
```

```
https://github.com/redhat-gpte-devopsautomation/openshift-tasks
```

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

```
[default@wetty-1-9nclg ~]$ oc expose svc openshift-tasks
```

7.12 **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.

```
[default@wetty-1-9nclg ~]$ oc set triggers dc openshift-tasks --manual
```

7.13 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: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
```


7.14 Crear pipeline en Jenkins.


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





Jenkins


Jenkins ▾ ▶


 **New Item**


 People


 Build History


 Manage Jenkins

 My Views



 Open Blue Ocean

 Lockable Resources

 Credentials

 New View

All +

S	W	Name ↓
		OpenShift S...

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

Build Queue

No builds in the queue.

Build Executor Status

1 Idle

2 Idle

3 Idle

4 Idle

7.15 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**”

The screenshot shows the Jenkins 'New Item' configuration page. At the top, there is a section titled 'Enter an item name' with a text input field containing 'userx-tasks'. Below this, there is a list of job types: 'Freestyle project', 'Pipeline', 'Multi-configuration project', 'Bitbucket Team/Project', 'Folder', and 'GitHub Organization'. The 'Pipeline' option is selected and highlighted with a red box. At the bottom left, there is a blue 'OK' button, also highlighted with a red box. The 'Freestyle project' description reads: 'This is the central feature of Jenkins. Jenkins will build something other than software build.' The 'Pipeline' description reads: 'Orchestrates long-running activities that can span mul organizing complex activities that do not easily fit in fr'. The 'Multi-configuration project' description reads: 'Suitable for projects that need a large number of differ'. The 'Bitbucket Team/Project' description reads: 'Scans a Bitbucket Cloud Team (or Bitbucket Server F'. The 'Folder' description reads: 'Creates a container that stores nested items in it. Usef namespace, so you can have multiple things of the sa'. The 'GitHub Organization' description reads: 'Scans a GitHub organization (or user account) for all i'.

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

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

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

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

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

Pipeline

Definition

Pipeline script

Script

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


☒ Use Groovy Sandbox

[Pipeline Syntax](#)

Save


Apply


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



Jenkins

Jenkins » Tasks »



[Back to Dashboard](#)
[Status](#)
[Changes](#)
[Build Now](#) 
[Delete Pipeline](#)
[Configure](#)
[Full Stage View](#)
[Open Blue Ocean](#)
[Rename](#)
[Pipeline Syntax](#)


 [Recent Changes](#)



Stage View

No data available. This Pipeline

Build History


 **Build History** [trend](#) 




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


Permalinks


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)


#1

Nov 30 16:05

No Changes

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


Permalinks


 **Build History**

[trend](#)

#1

Nov 30, 2019 10:05 PM

 RSS for all

 RSS for failures

7.18 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: “**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 displays the OpenShift web console interface. On the left is a dark sidebar with navigation links: Overview, Applications, Builds, Resources, Storage, Monitoring, and Catalog. The top header shows the current project as 'user1-cicd-tareas'. The main content area is titled 'APPLICATION openshift-tasks'. Below this, a 'DEPLOYMENT CONFIG' section highlights 'openshift-tasks, #2'. The 'CONTAINERS' section lists details for the 'openshift-tasks' container: Image (user1-cicd-tareas/openshift-tasks 1455b13, 633.2 MiB), Build (openshift-tasks, #2), Source (Merge pull request #1 from jeichler/https 308f21f), and Ports (8080/TCP and 2 others). The 'NETWORKING' section shows a 'Service - Internal Traffic' for 'openshift-tasks' with ports '8080/TCP (8080-tcp) → 8080 and 2 others'. The 'BUILDS' section at the bottom lists '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
```

7.19 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.medellin-9f63.open.redhat.com/>

Logger

Log Info

Log Warning

Log Error

Load Generator

Seconds

Load!

Danger Zone

HEALTHY

Toggle Health

Kill Instance

Info

Pod Hostname	openshift-tasks-2-lrps6
Pod IP	null
Used Memory	184 MB
Session ID	0PXxhR0CMwIP35T_zSC1itNBrgfM2kqTLC7XLu7s

Messages

Nothing to report.