Workshop Introducción a Openshift Container Platform

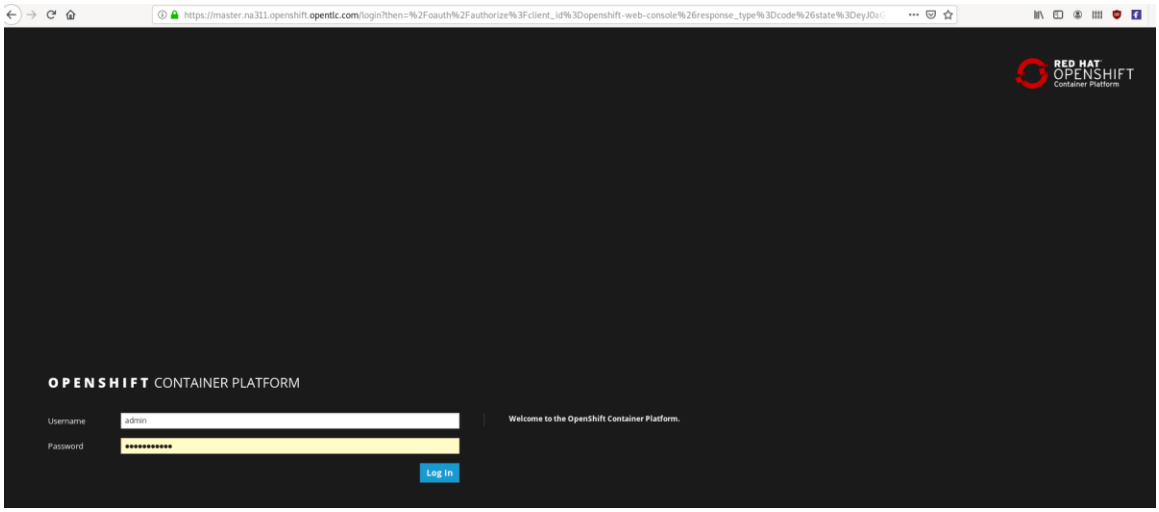
Noviembre 2019

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 prerrequisitos.
Solicitar URL del Openshift al presentador.

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

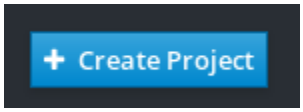
Usuario: (pedir usuario)
Password: openshift



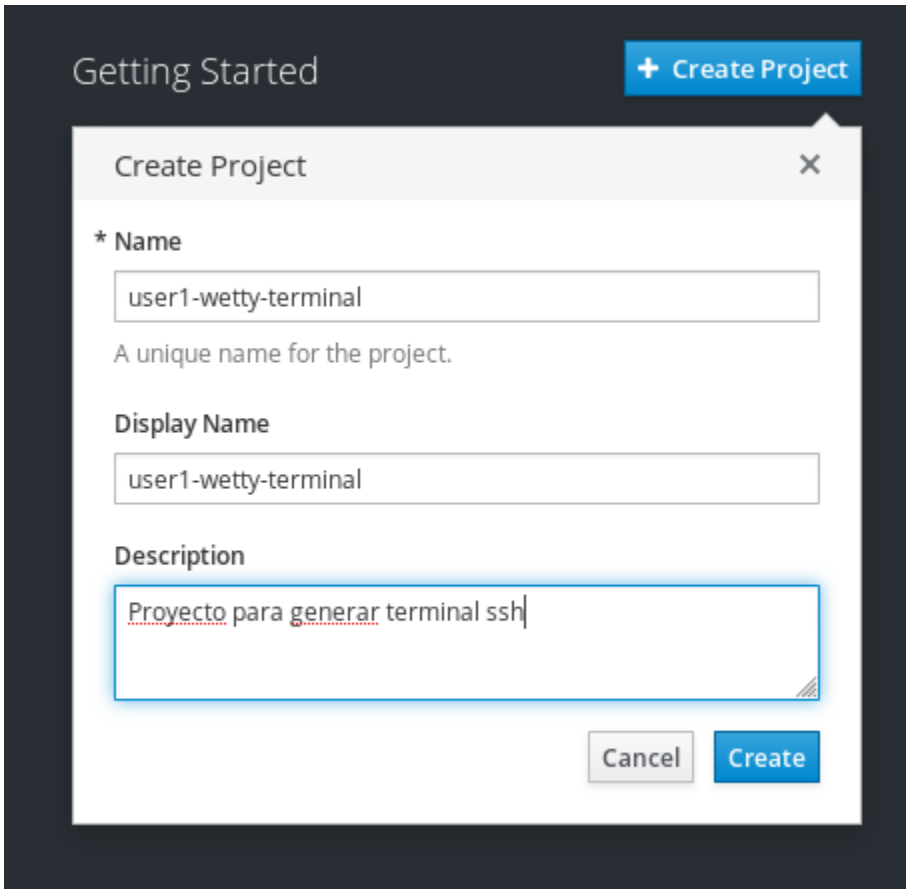
2. Generación de terminal de linea de comandos por medio de un container web tty de NodeJs, necesaria para realizar el laboratorio.

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

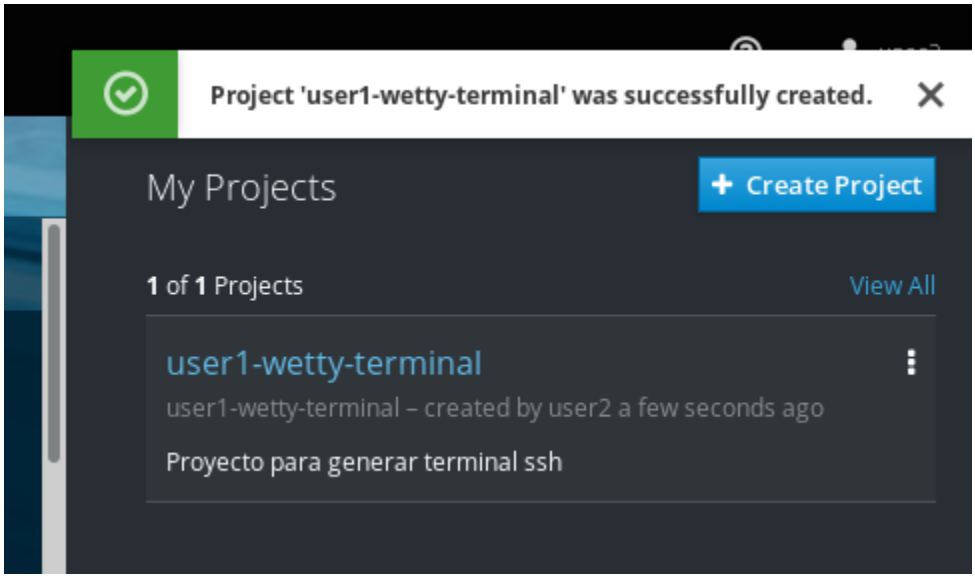
A la izquierda de la interfaz web de Openshift hay un botón que aparece con nombre "Create Project" y le damos click.



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

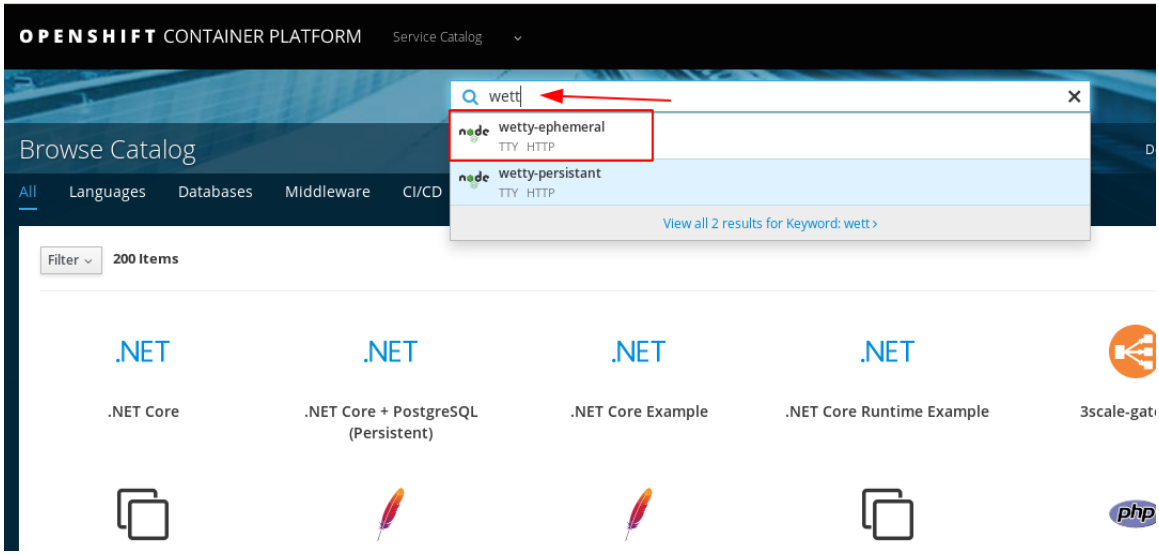


Y le damos click en “Create”, automáticamente se crea el proyecto, deberia dar un output como el siguiente:

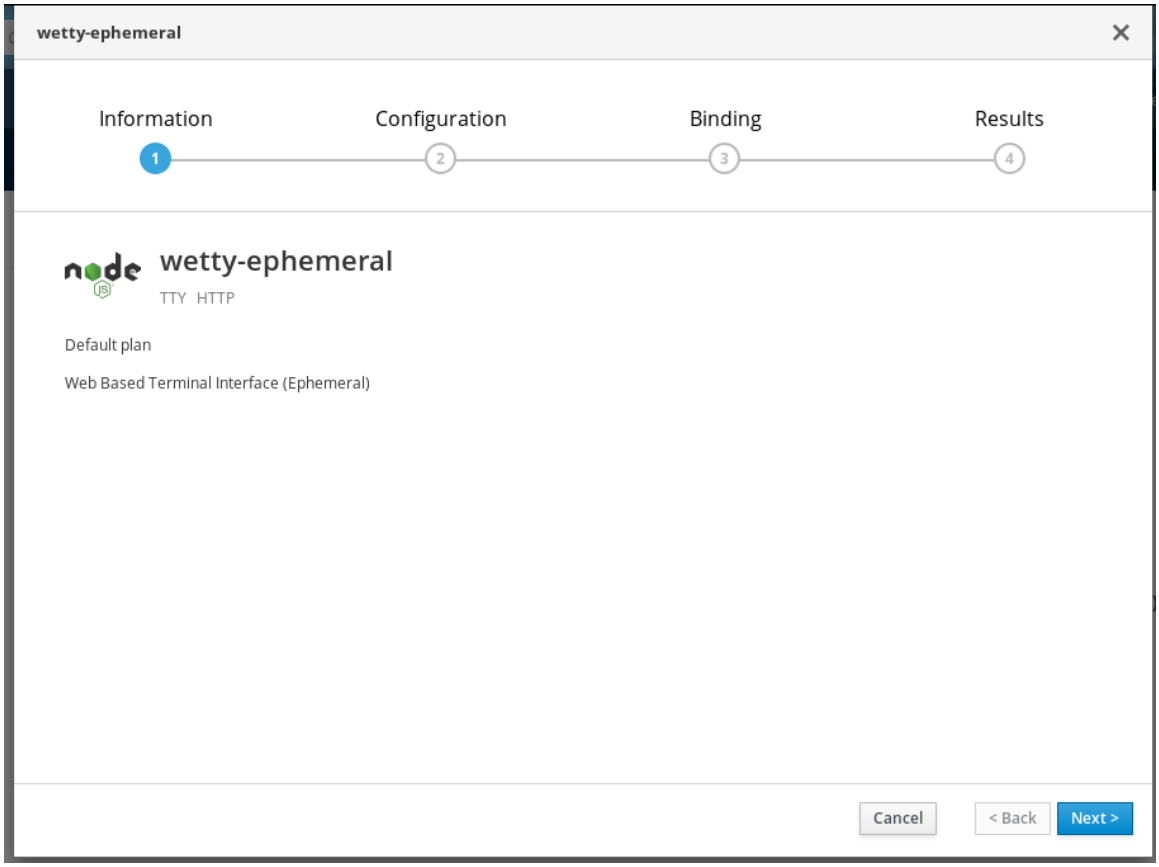


2.2 Ahora vamos a hacer deploy del “wetty-ephemeral”.

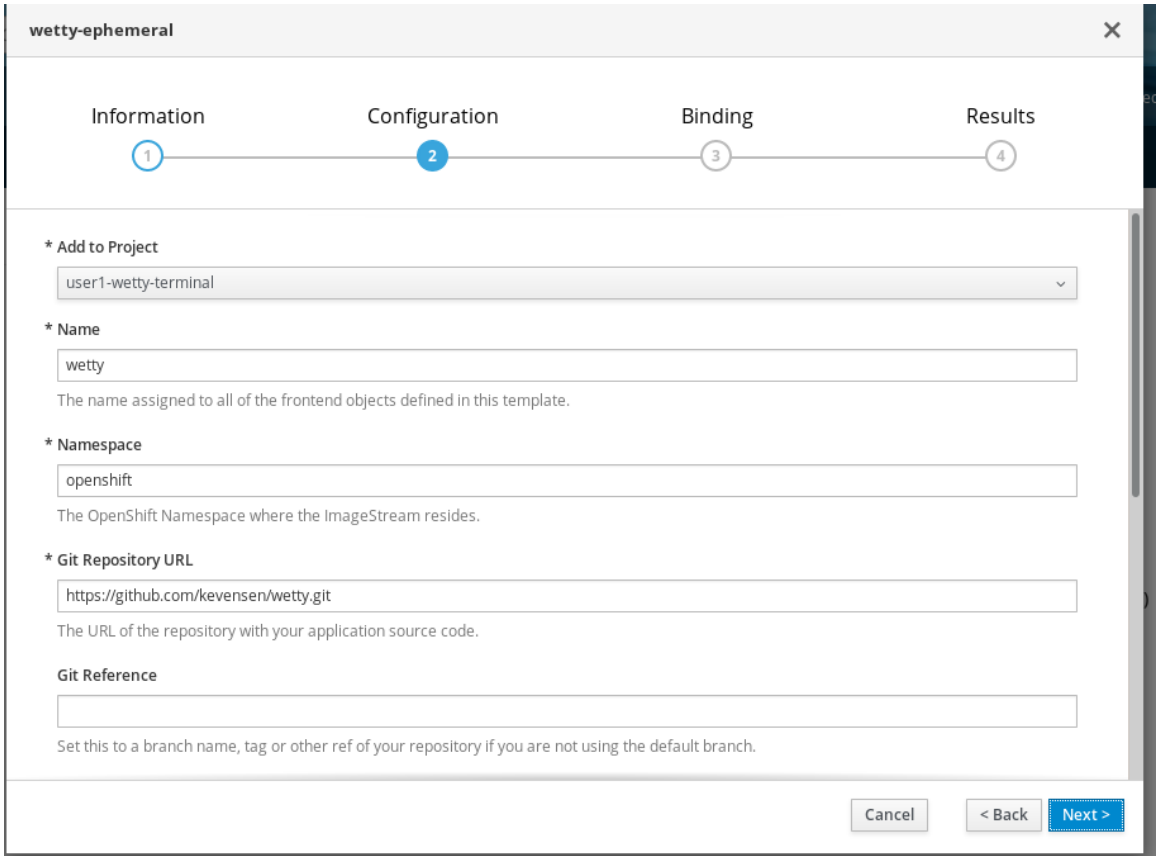
Para esto, buscamos **wetty-ephemeral** template desde el catálogo:

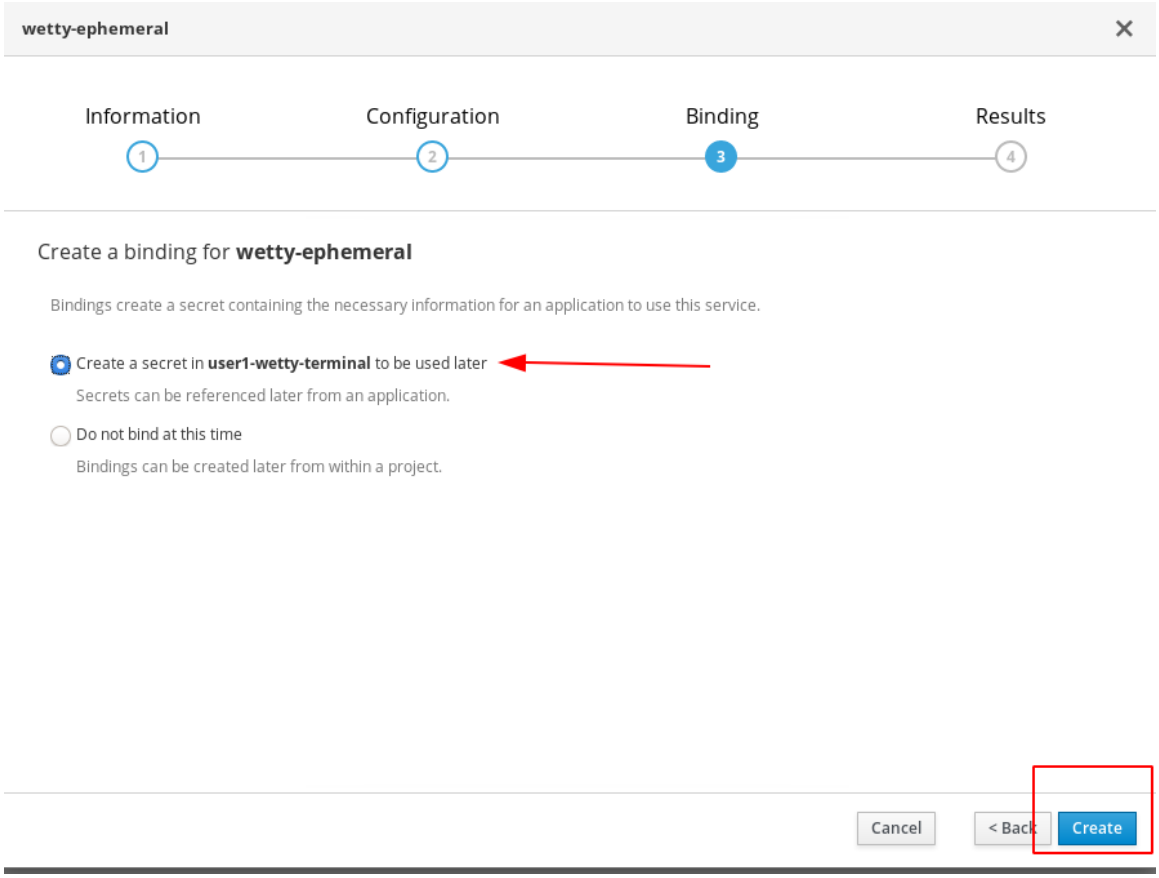


Para esto presione click en “wetty-ephemeral” y luego click en next:

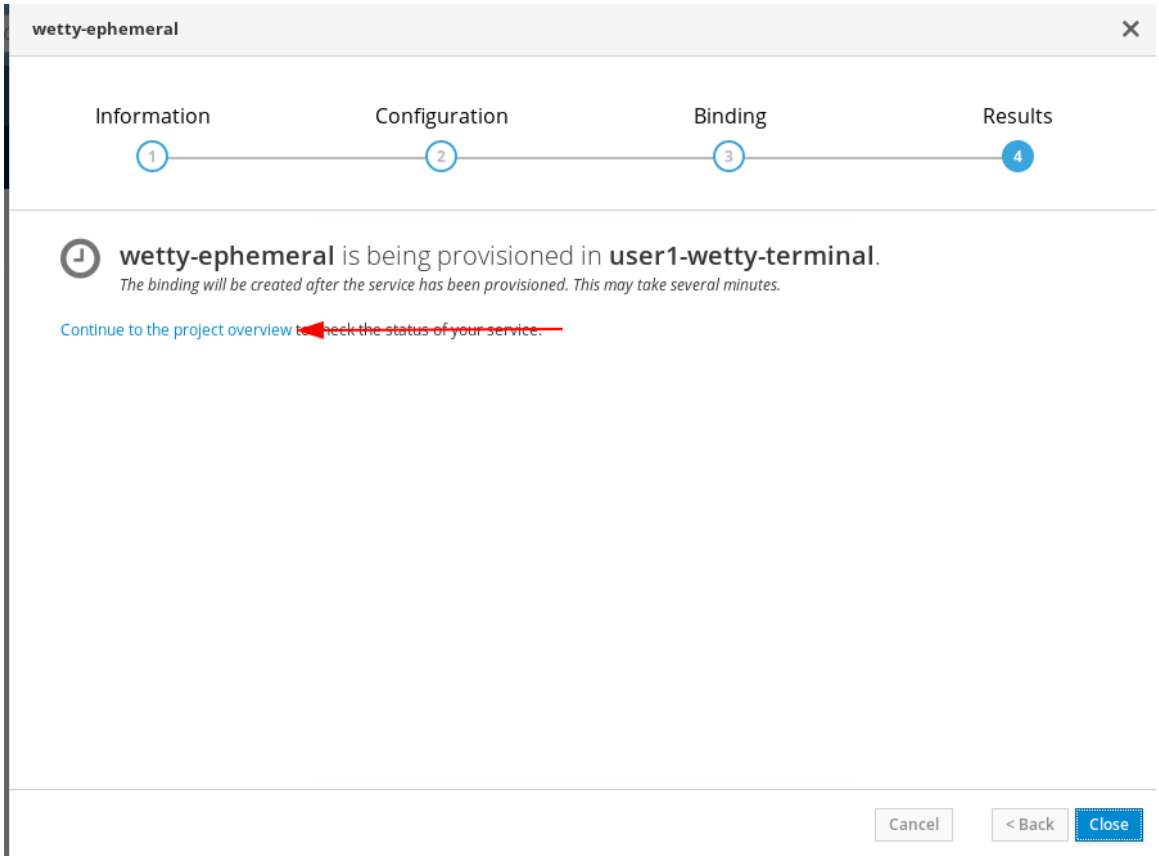


Escogemos el proyecto que creamos anteriormente “**userx**-wetty-terminal”, dejamos todo default y le damos next y luego Create como se muestra en la imagen.

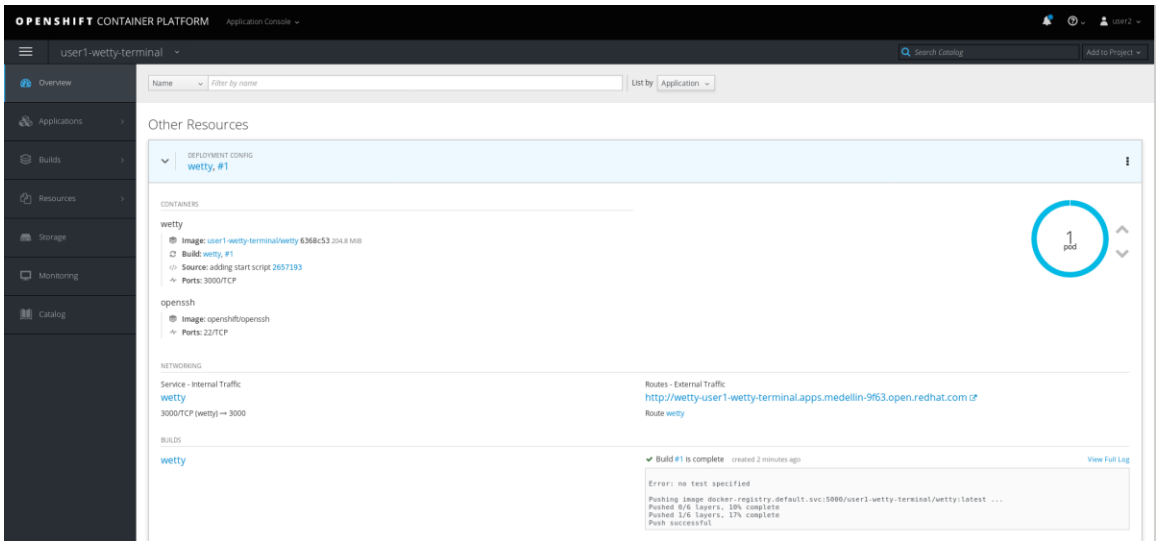



wetty-ephemeral
Information
Configuration
Binding
Results
1
2
3
4
Create a binding for wetty-ephemeral
Bindings create a secret containing the necessary information for an application to use this service.
Create a secret in user1-wetty-terminal to be used later
Secrets can be referenced later from an 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.

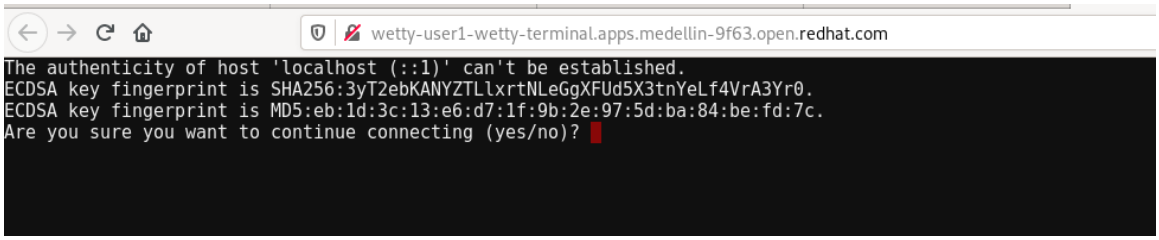


Esperamos que el deployment config esté con 1 pod “Running”, como se muestra a continuación.



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.

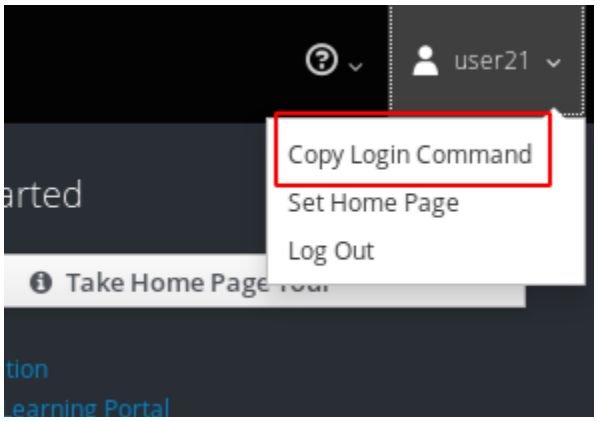
Al darle click se nos deberá abrir una página como la siguiente.



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 deberia mostrar un mensaje como el siguiente:

```
[default@wetty-1-9nclg ~]$ oc login https://master.medellin-9f63.open.redhat.com:443 --
token=djwLIvU4uWvIMKbej5KslZpWwetCuGbCmiSUQL9EzC4
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 projecto por linea de comandos de OC.

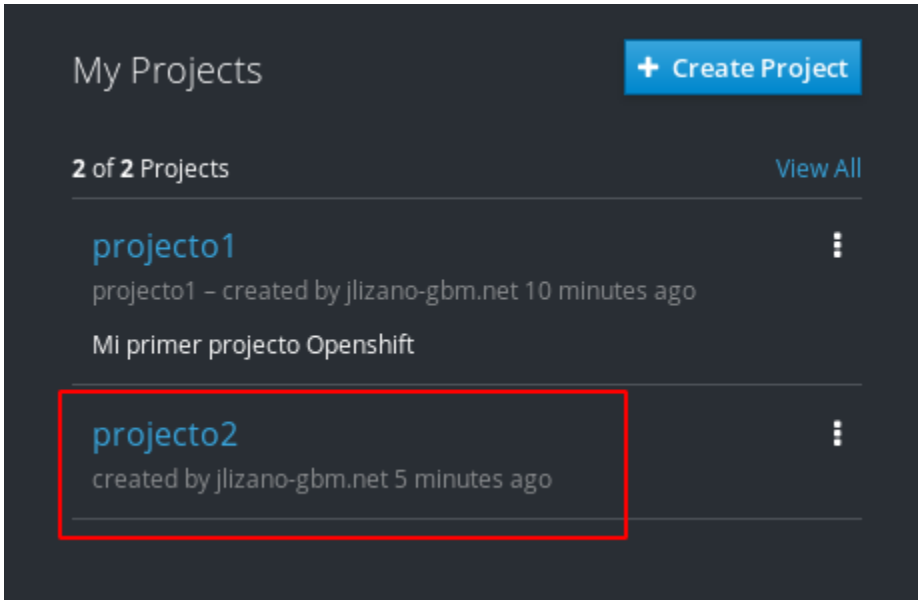
3.1 Ya logueados en la consola de OC, creamos el nuevo projecto con el nombre "projecto2", de la siguiente forma, deberia dar el siguiente output.

```
$ oc new-project miusuario-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 projecto "miusuario-projecto2" creado desde la linea de commandos oc.

```
$ oc get projects
NAME        DISPLAY NAME    STATUS
projecto1   projecto1        Active
projecto2                    Active
```

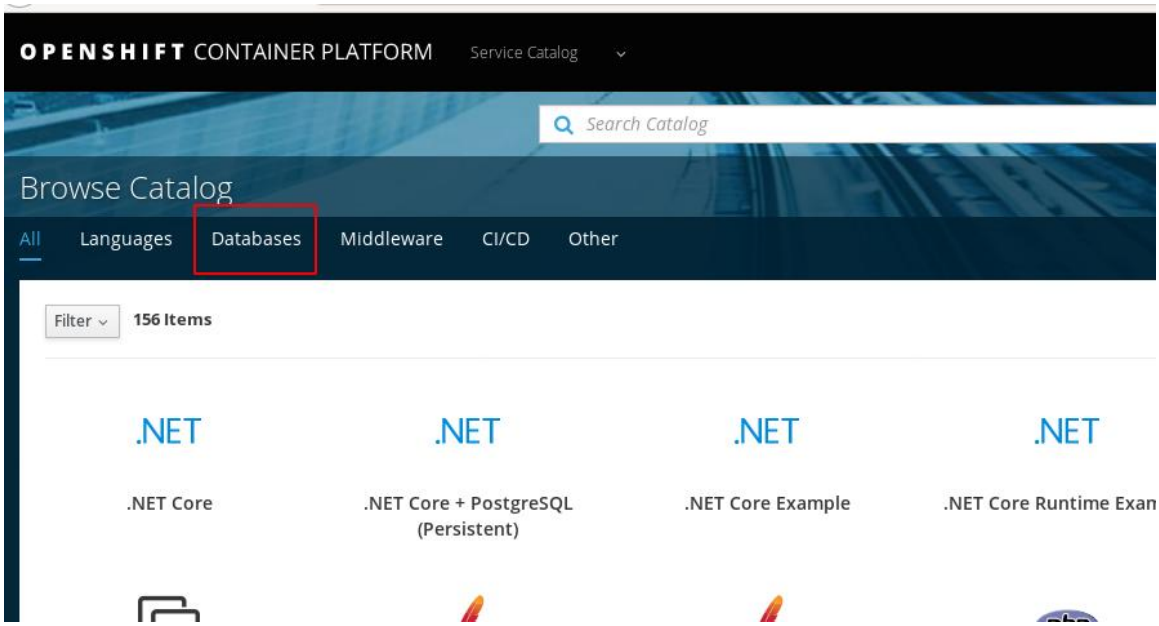
3.3 También lo podemos verificar en la consola web.

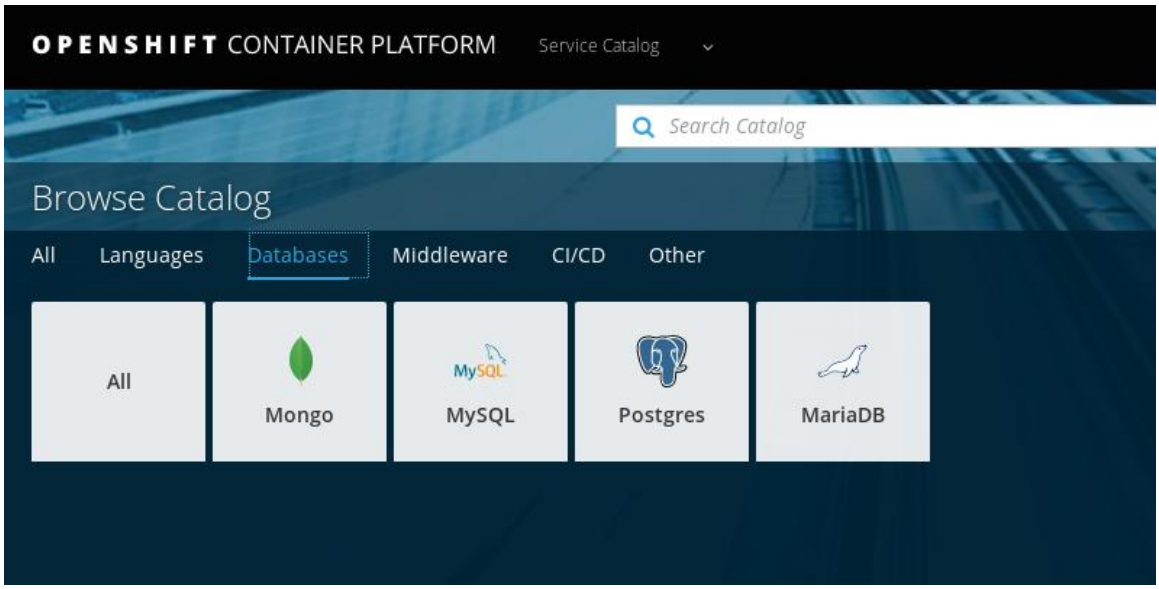


4. Ejercicio desplegar aplicación Node.JS front end con conexión a una base de datos MongoDB, desde la interfaz gráfica de Openshift.

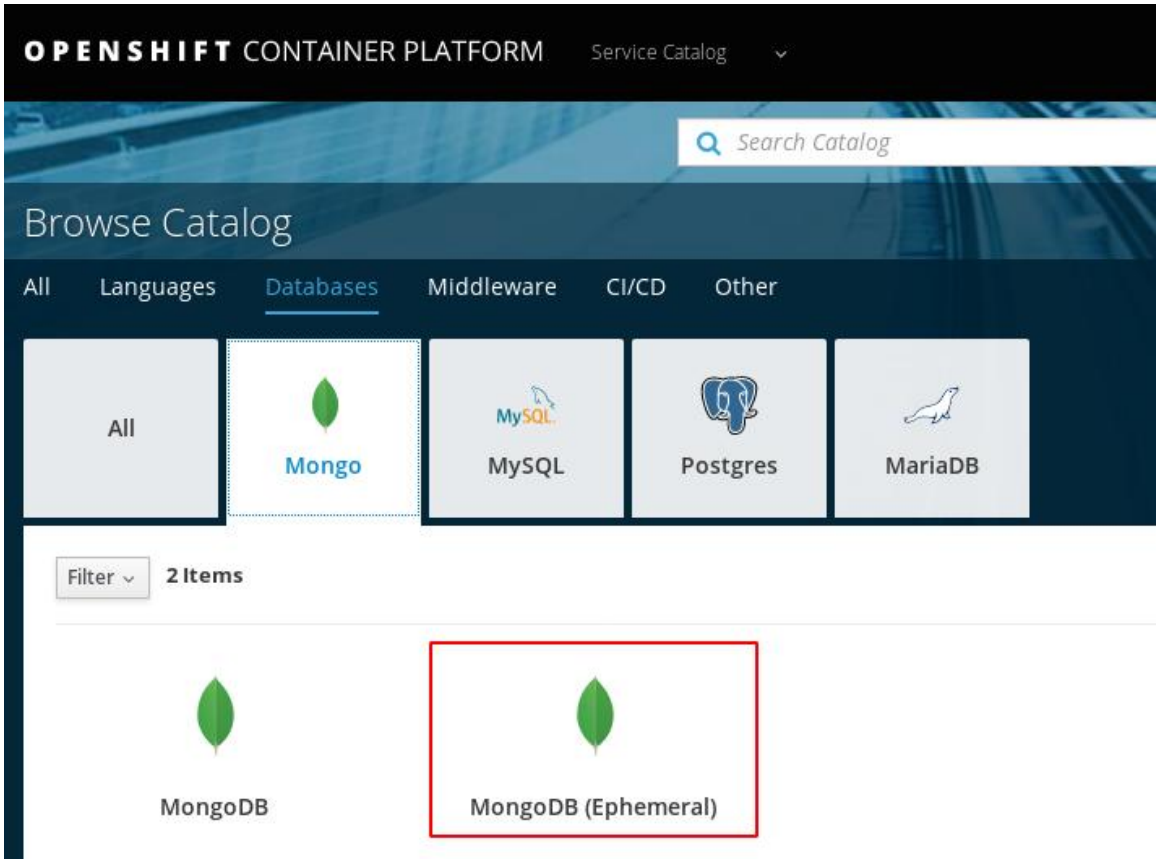
4.1 Desplegar una base de datos MongoDB (Ephemeral).

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

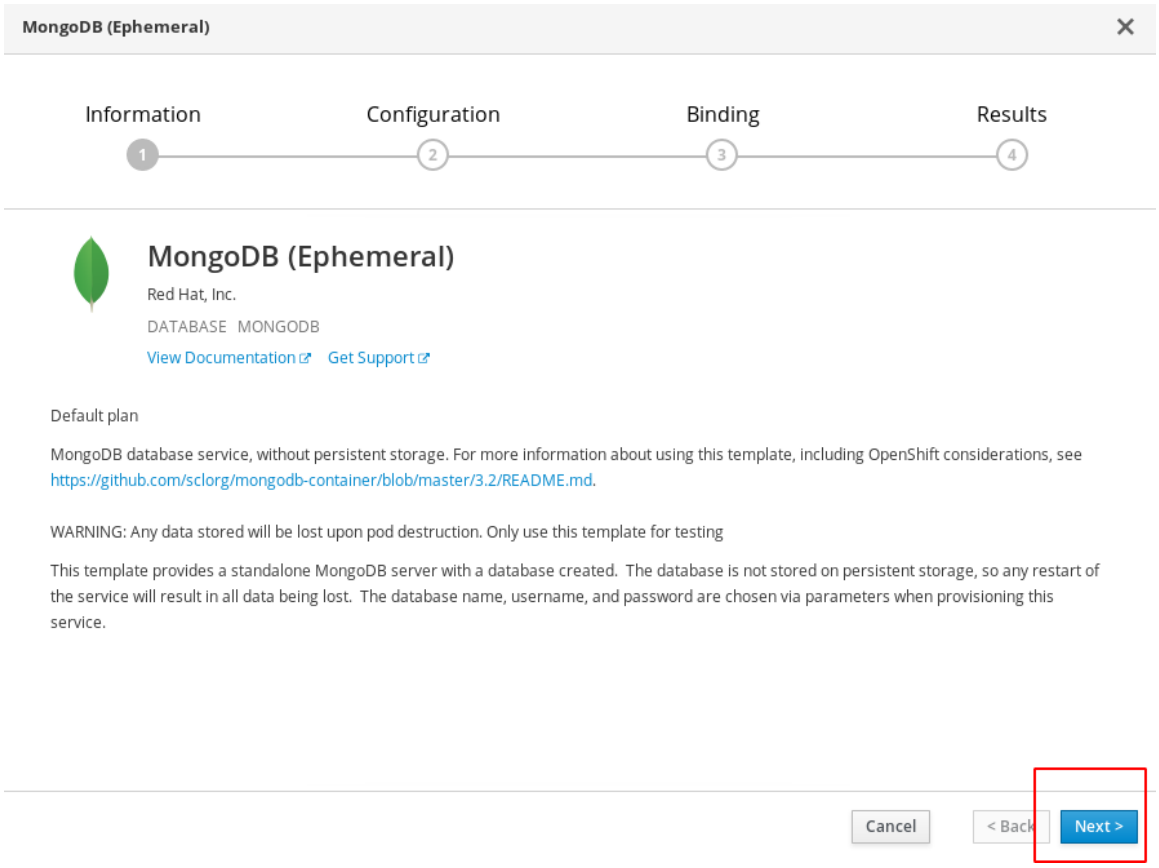




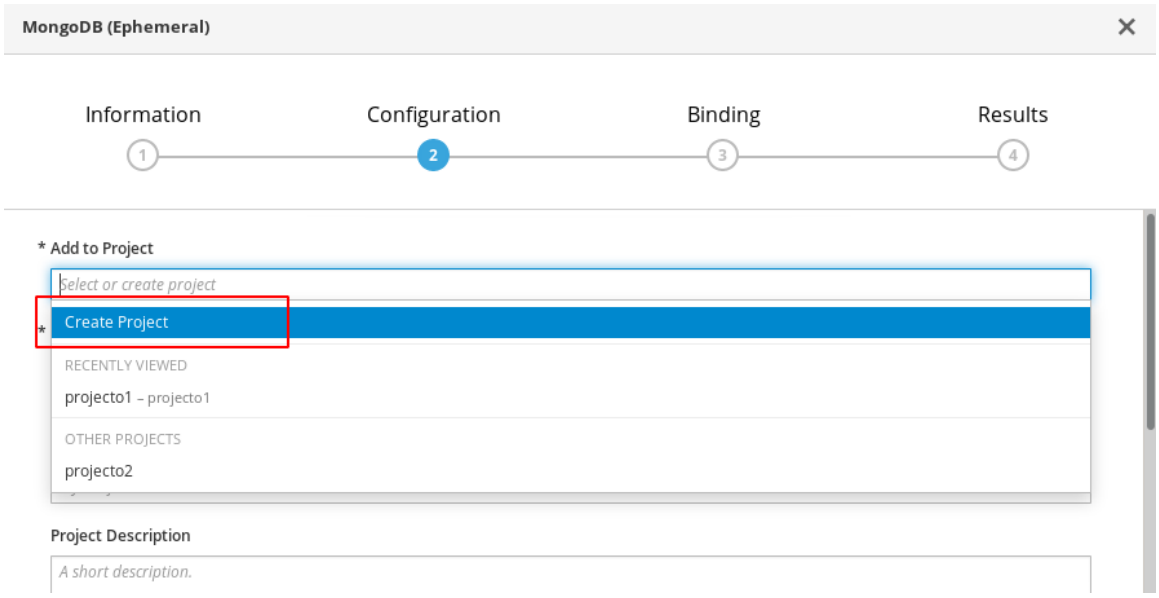
4.2 Escogemos Mongo y luego MongoDB (Ephemeral).



4.3 Y le damos next:



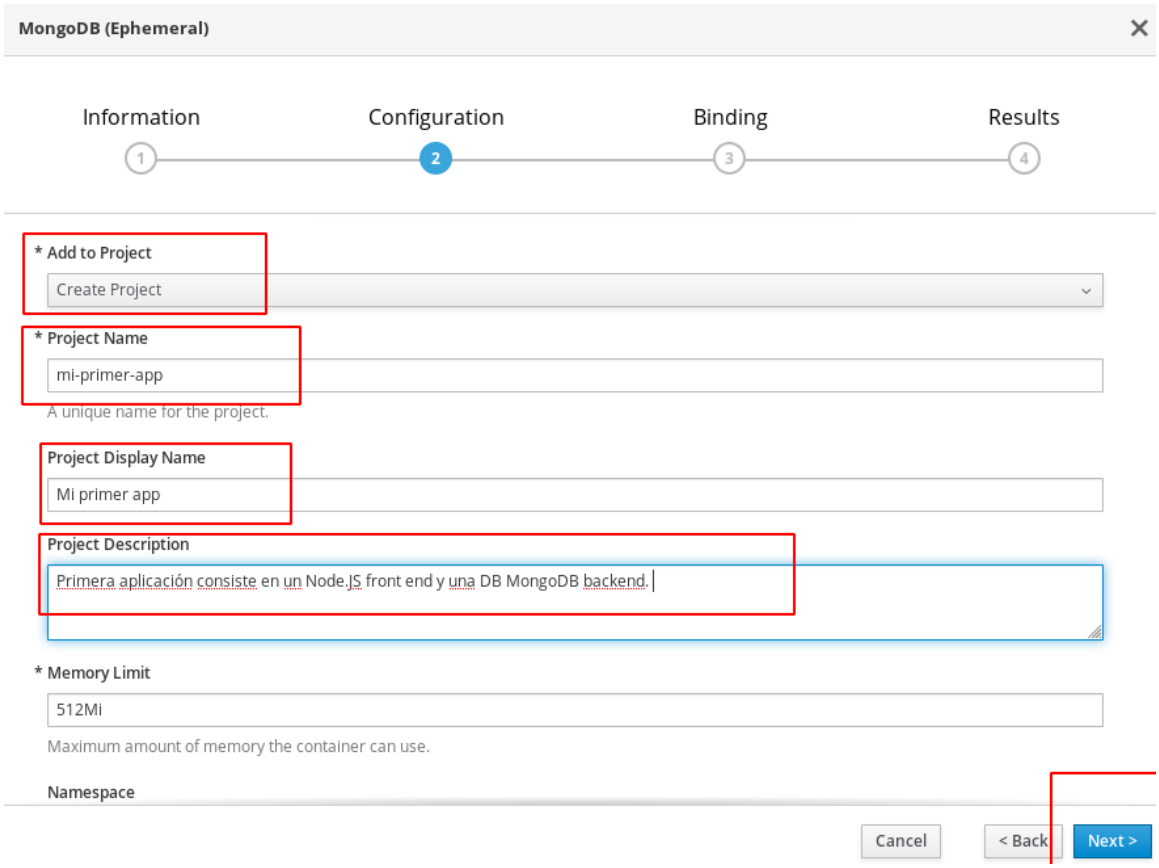
4.4 Procedemos con la creación de un proyecto nuevo de la siguiente forma:



4.5 Le agregamos los siguientes datos al proyecto:

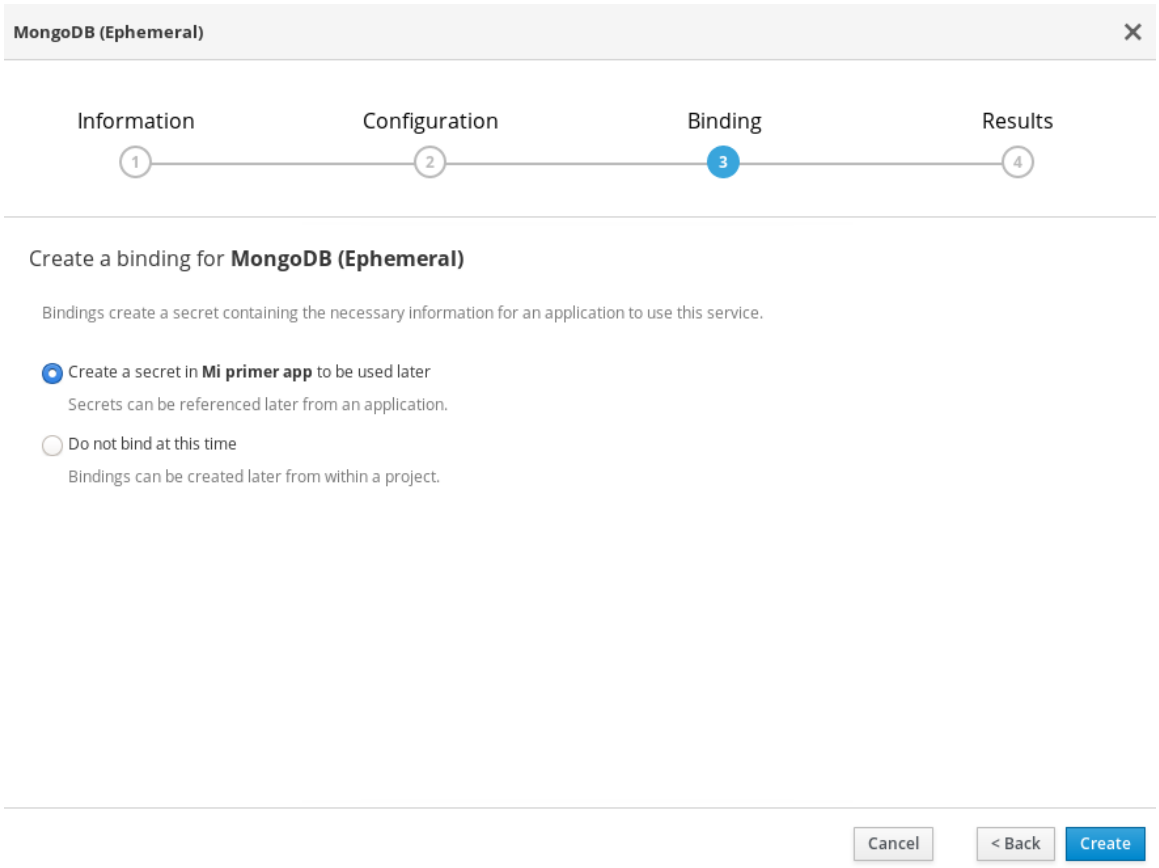
Project Name: usuario-mi-primer-app
Project Display Name: usuario-Mi primer app
Project Description: Primera aplicación consiste en un Node.JS front end y una DB MongoDB backend.

Y le damos “Next”

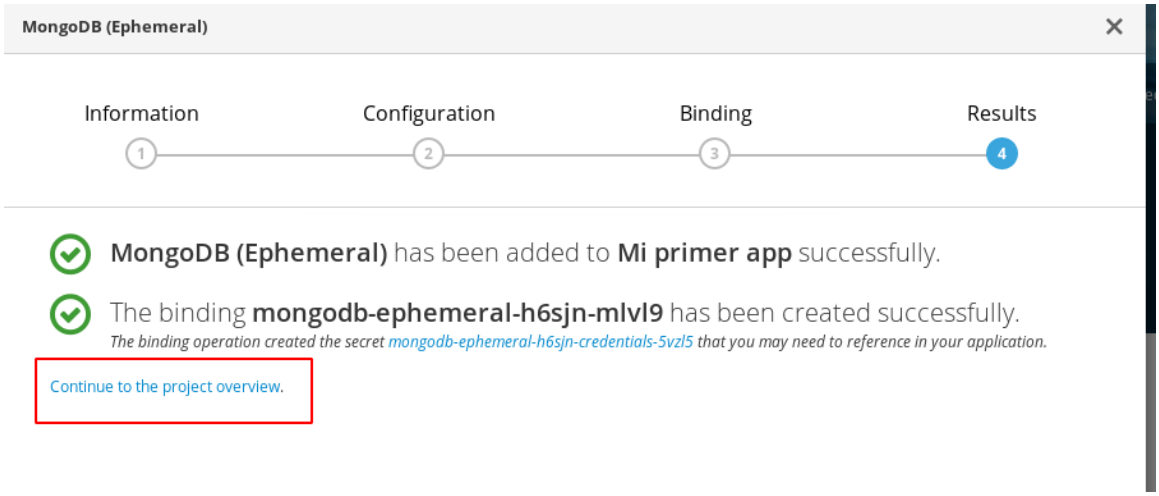


4.6 En la página de “Bindings”, seleccionamos “Create a secret in **Mi primer app** to be used later”

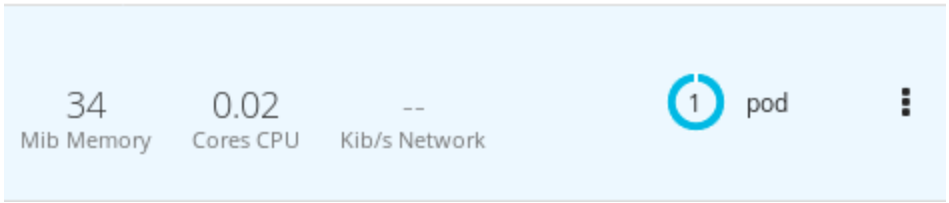
Y le damos ”Create”:



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

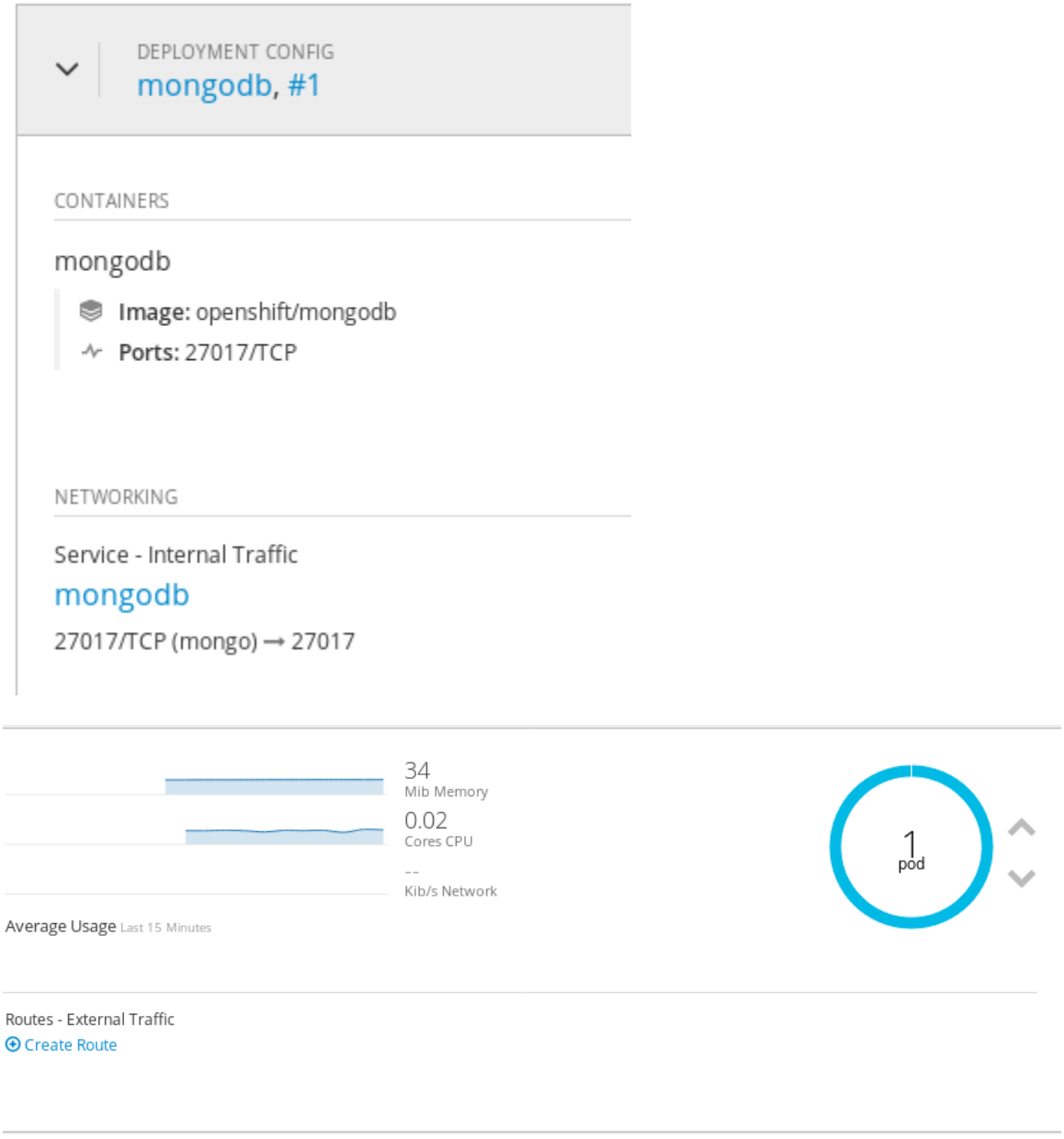


Observe que el deployment está “Running” y se puede ver un pod disponible (indicado en azúl).

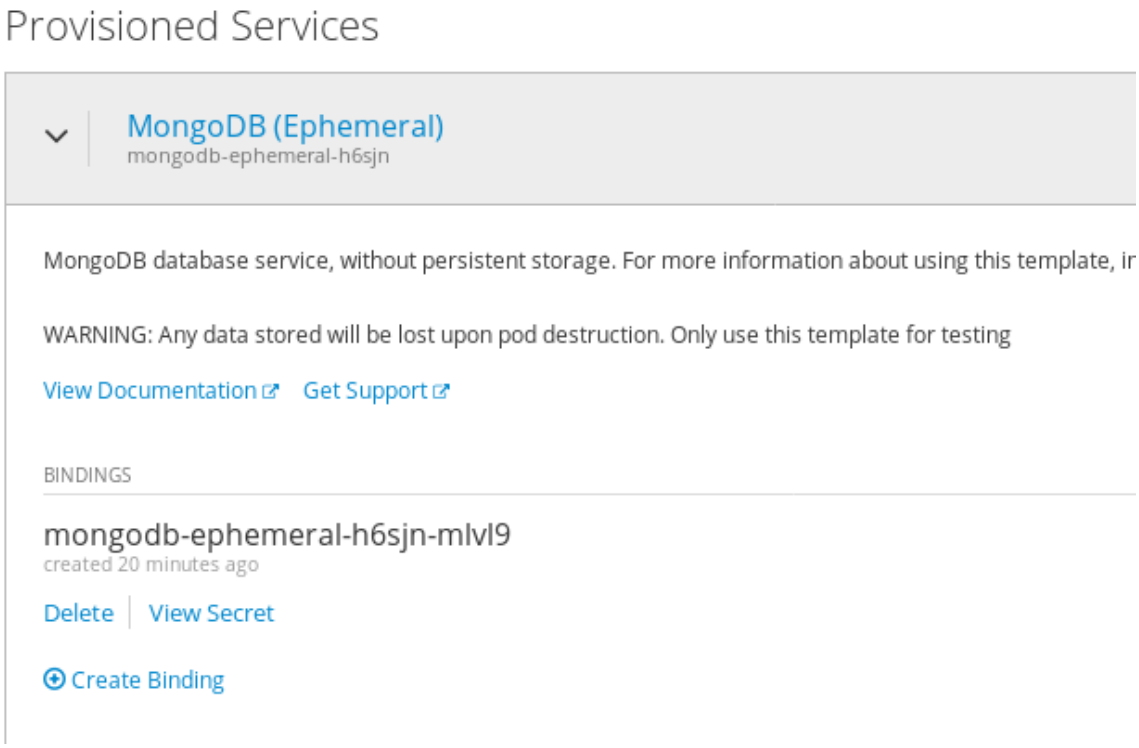


4.8 Click en MongoDB deployment para revisar los detalles entre ellos:

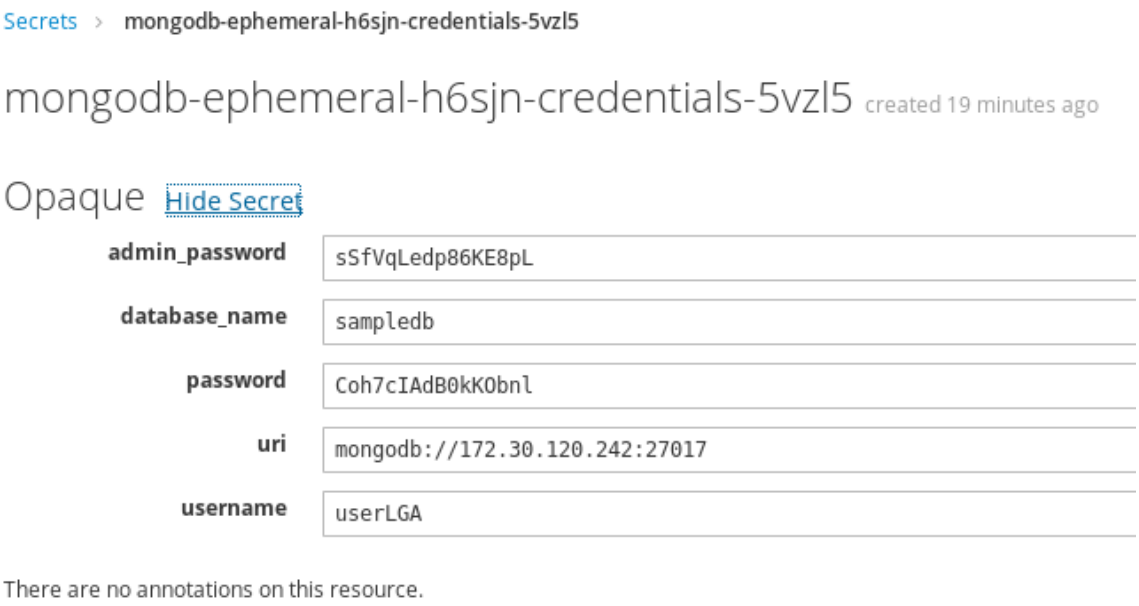
- La imagen usada para la base de datos MongoDB.
- El puerto en el cuál el servicio de MongoDB está escuchando dentro del contenedor.
- El nombre del servicio: “mongodb”, por el cual es accesible la base de datos.
- No hay rutas creadas para el servicio “mongodb”, lo cual está bien porque la base de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



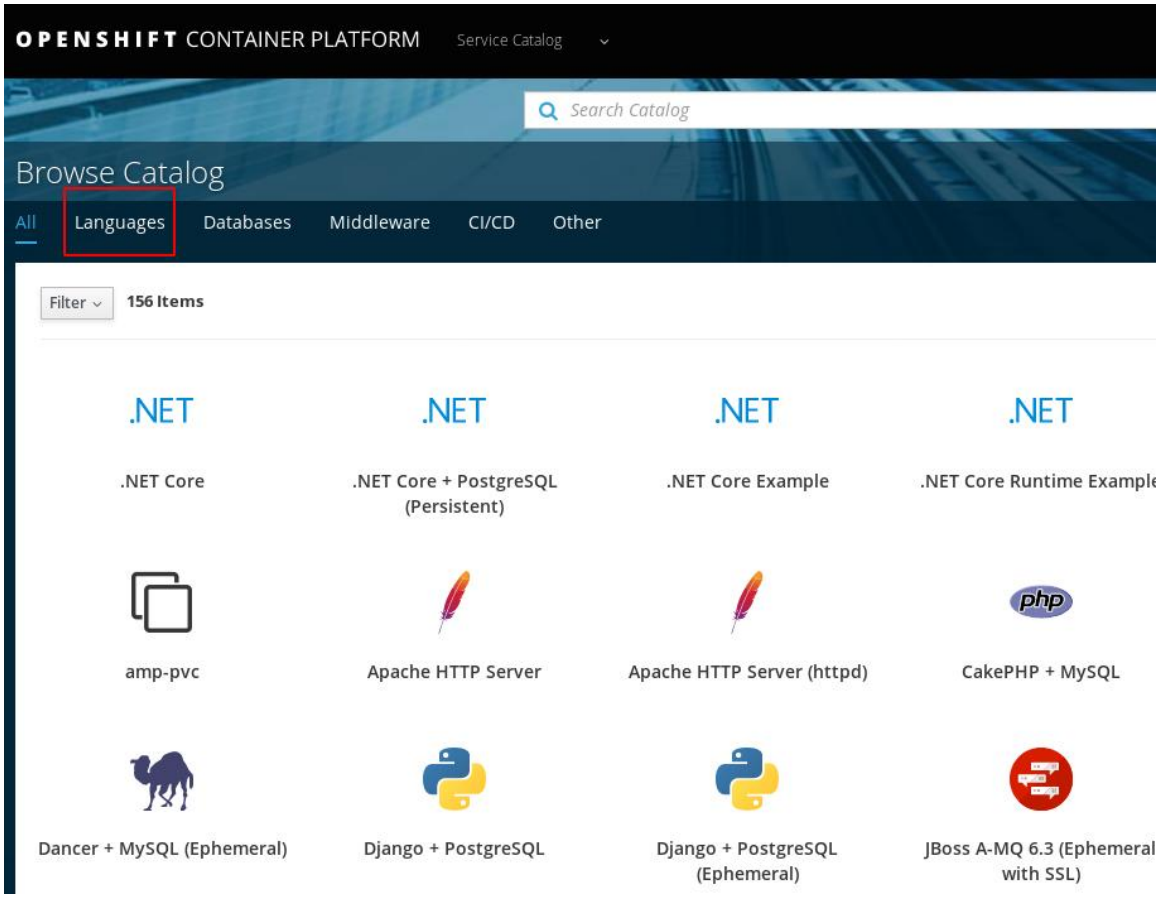
4.9 Nos movemos a “Provisioned Services” y expandimos MongoDB (Ephemeral).
Observe que el binding creado es similiar a: “mongodb-ephemeral-h6sjn-mlvl9”



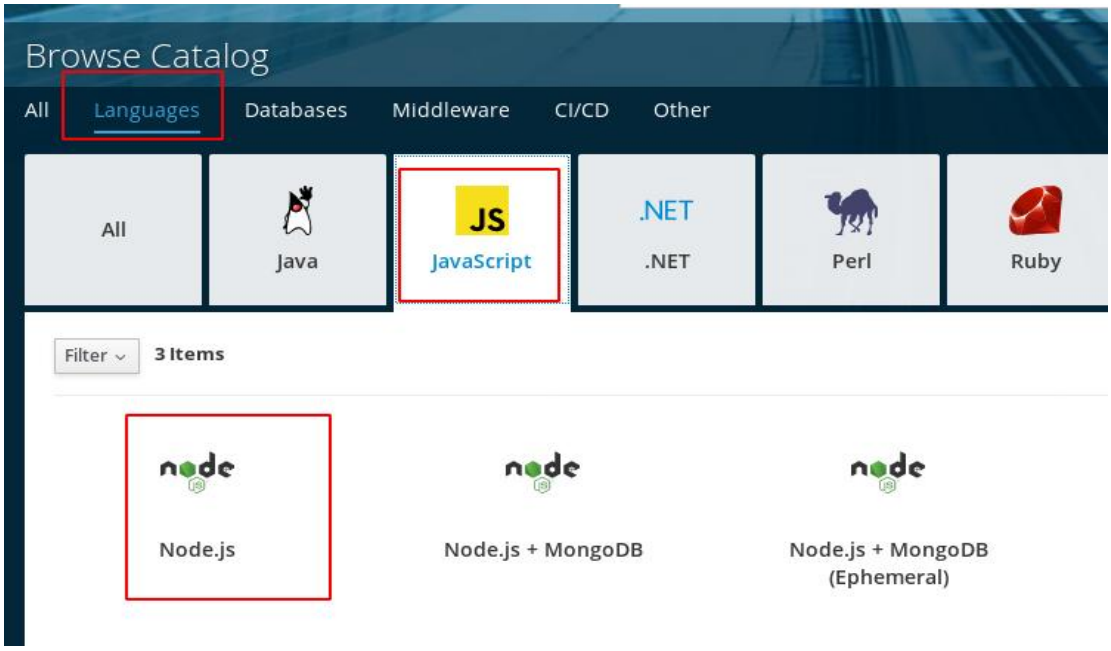
4.10 Le damos click en “View Secret” y nos aparecerá una página como la siguiente:
Le damos click en Reveal Secret para ver el contenido.
Observe que el “uri” apunta a la IP del servicio de la dirección de la base de datos.

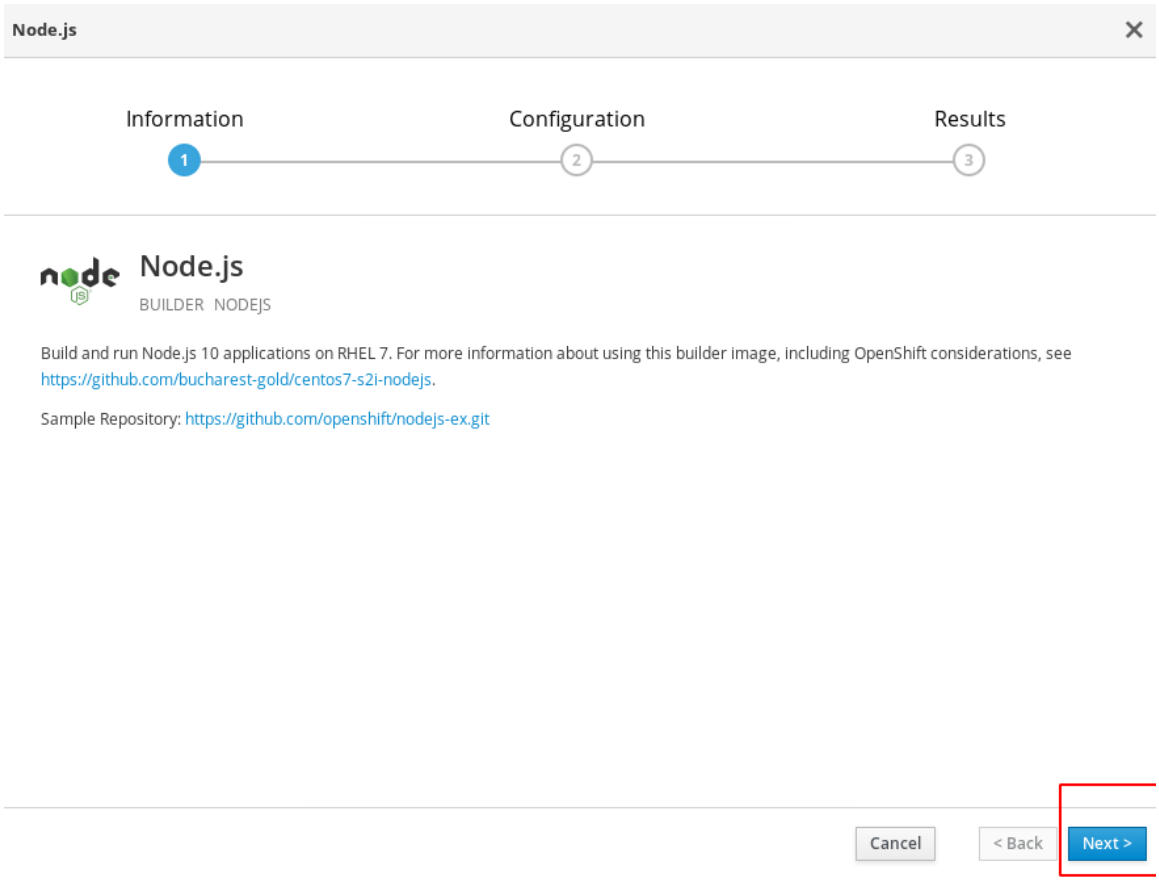


4.11 Creamos una aplicación Node.JS.
Click en “Openshift Container Platform” en la columna izquierda de la pantalla para volver al home screen con el catálogo.
En el catálogo, seleccionar “Languages”.



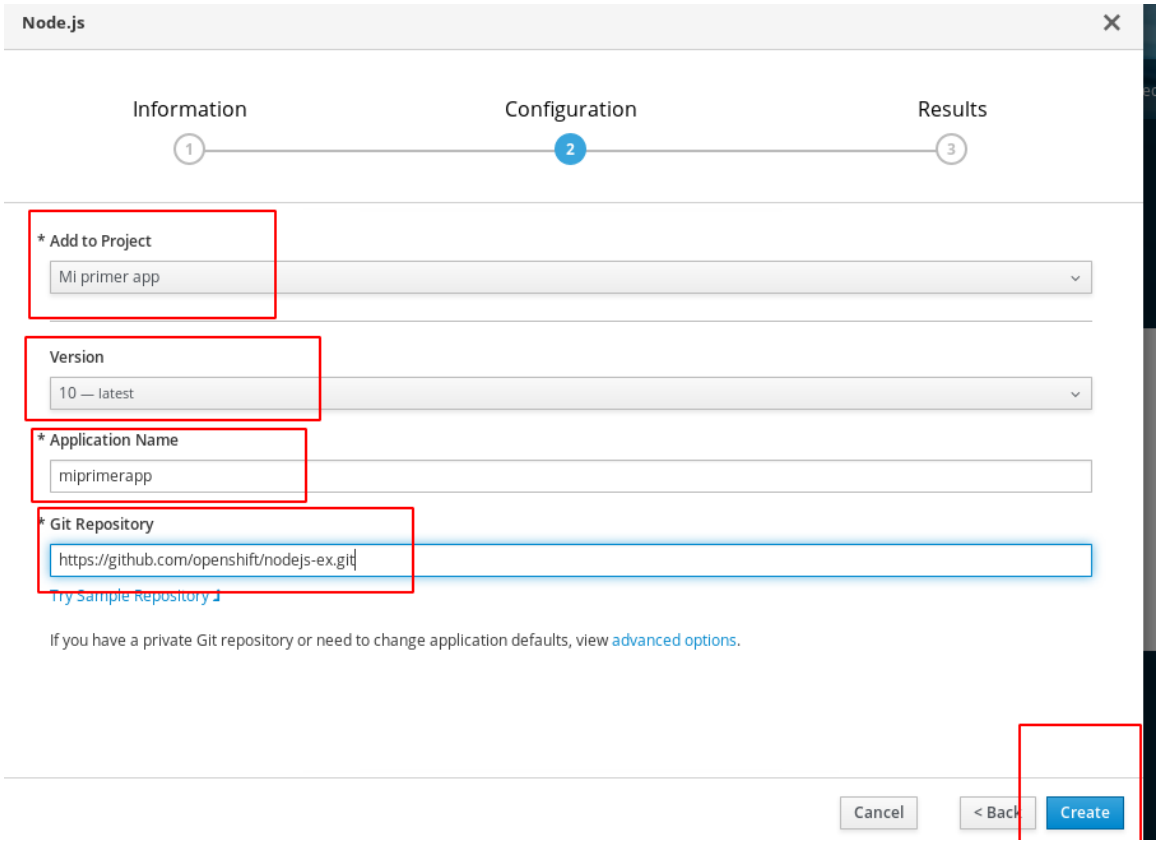
4.12 Click en “JavaScript” y seleccionar “Node.js” y le damos Next en la página de “Service provisioning wizard”



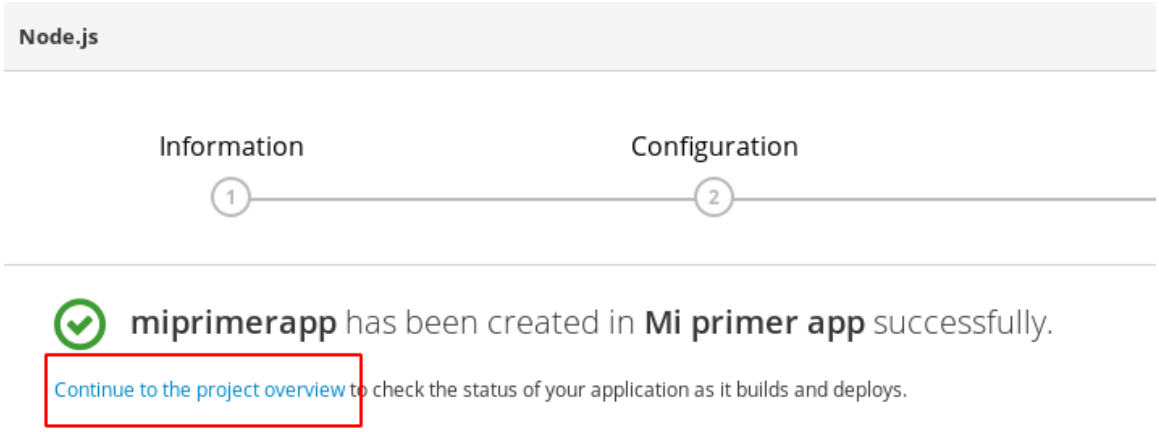


4.13 Use lo siguiente para configurar la aplicación.
Add to Project: Verifique que se utilice el mismo projecto donde se desplegó la DB MongoDB.
Version: 10 – latest
Application Name: miprimerapp
Git Repository: https://github.com/openshift/nodejs-ex.git

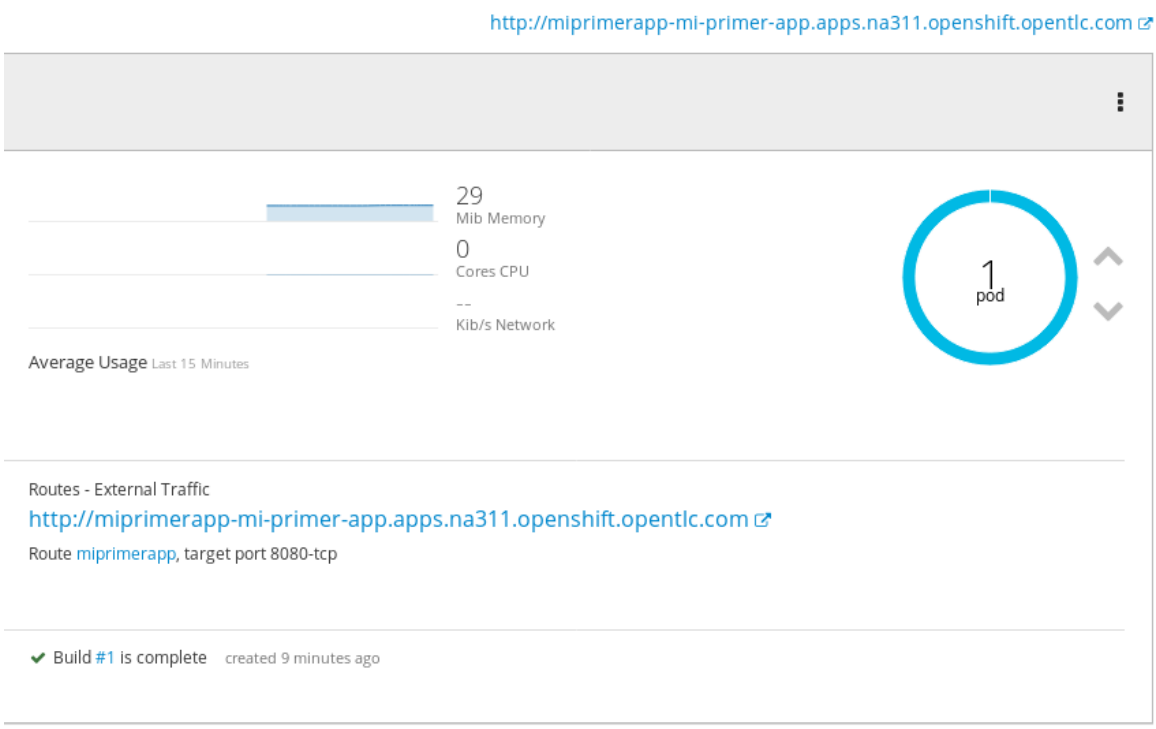
Después click en “Create”



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



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 deberia mostrarse similiar a: http://miprimerapp-mi-primer-app.apps.na311.openshift.opentlc.com .



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

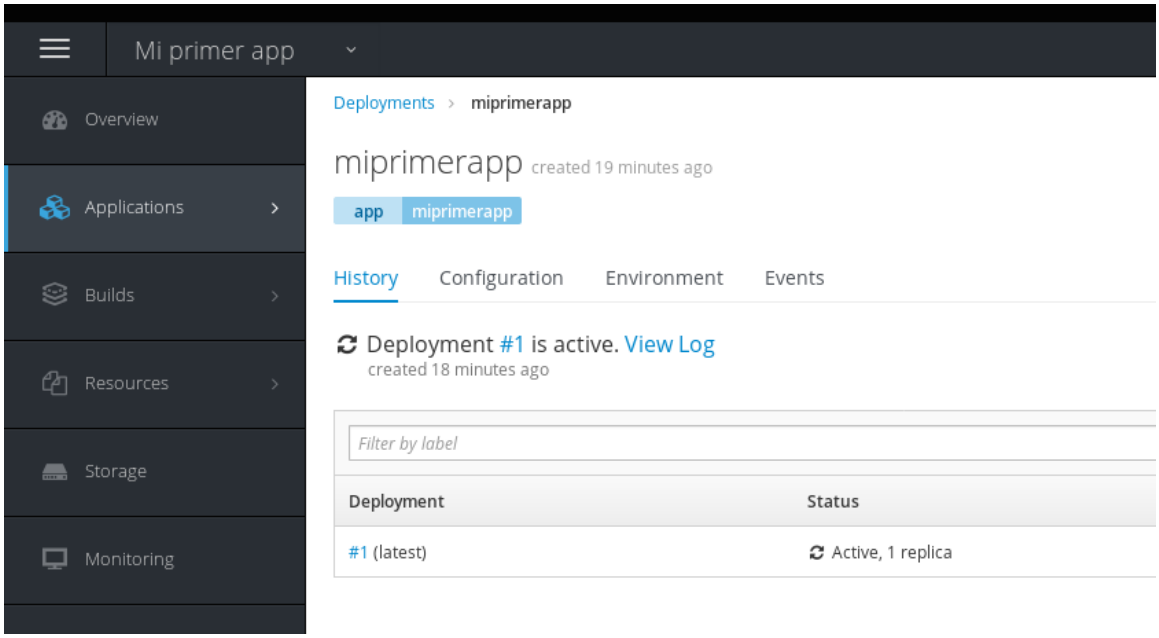
8080/TCP (8080-tcp) → 8080

BUILDS

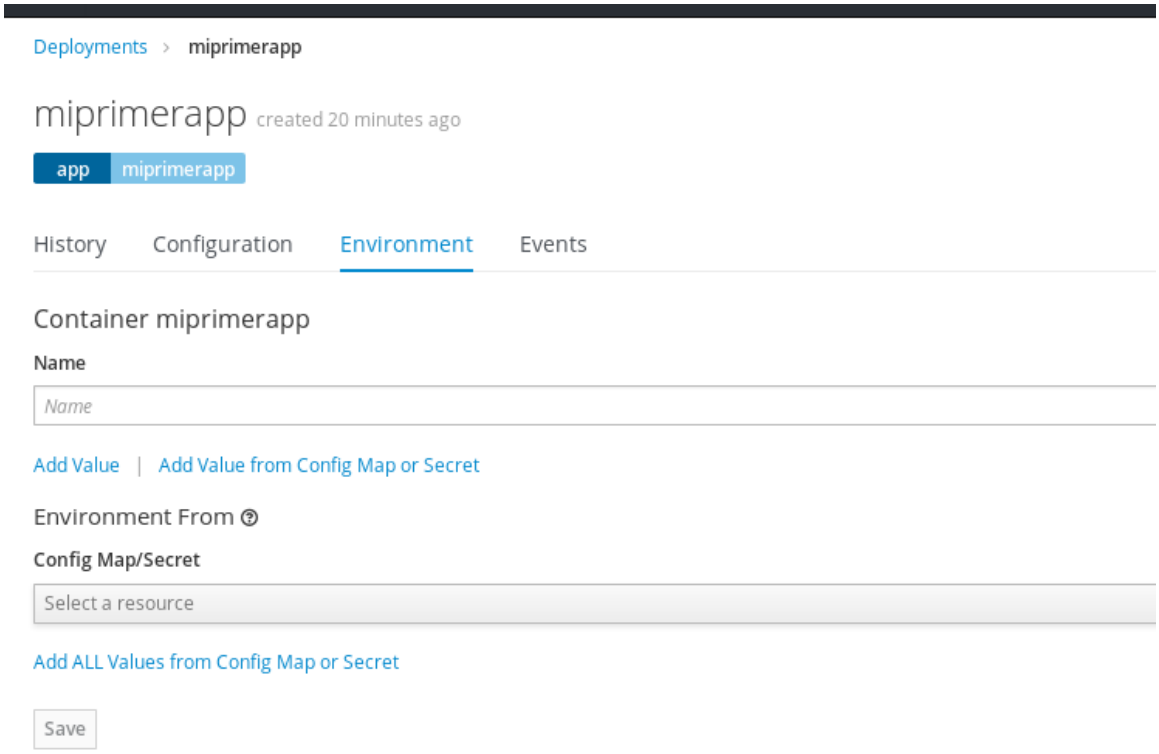
miprimerapp

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”.



4.17 Ingresamos a “Environment” dentro de “miprimerapp”

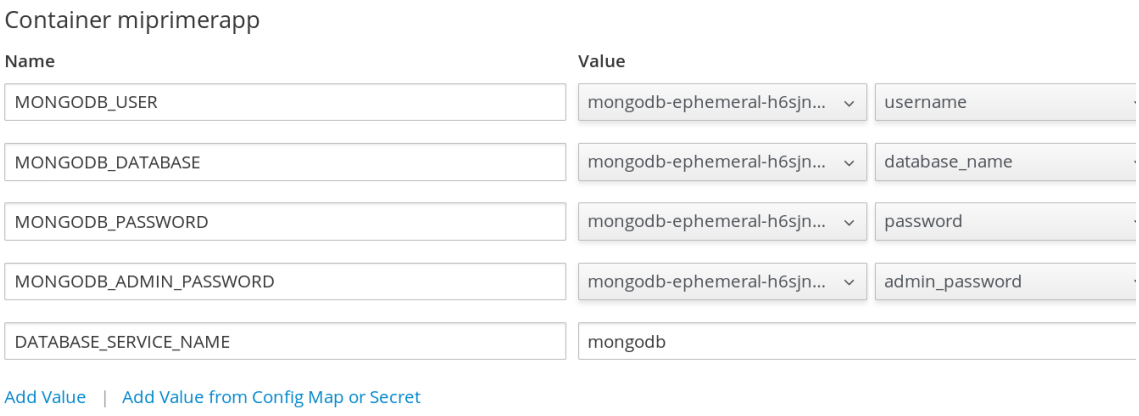


4.18 Esta aplicación Node.JS puede recibir la configuración de la base de datos a través de variables de entorno. Desafortunadamente, los nombres de variables que la aplicación espera son diferentes de los nombres de campo en su binding secret. Si fueran idénticos, podrías vincular todo el secreto. Pero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

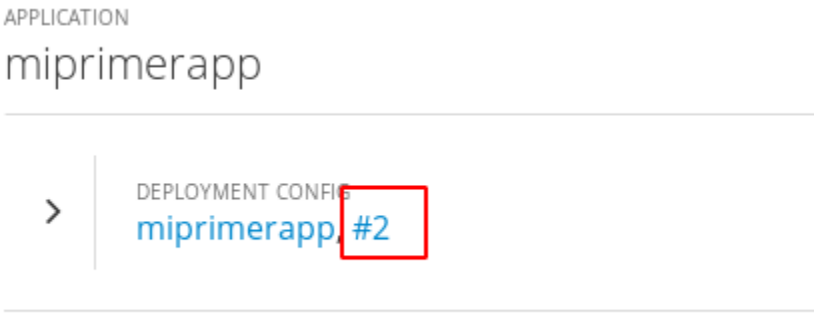
Le damos click a “Add Value from Config Map or Secret”.
Y agregamos 4 variables como las siguientes:

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”.



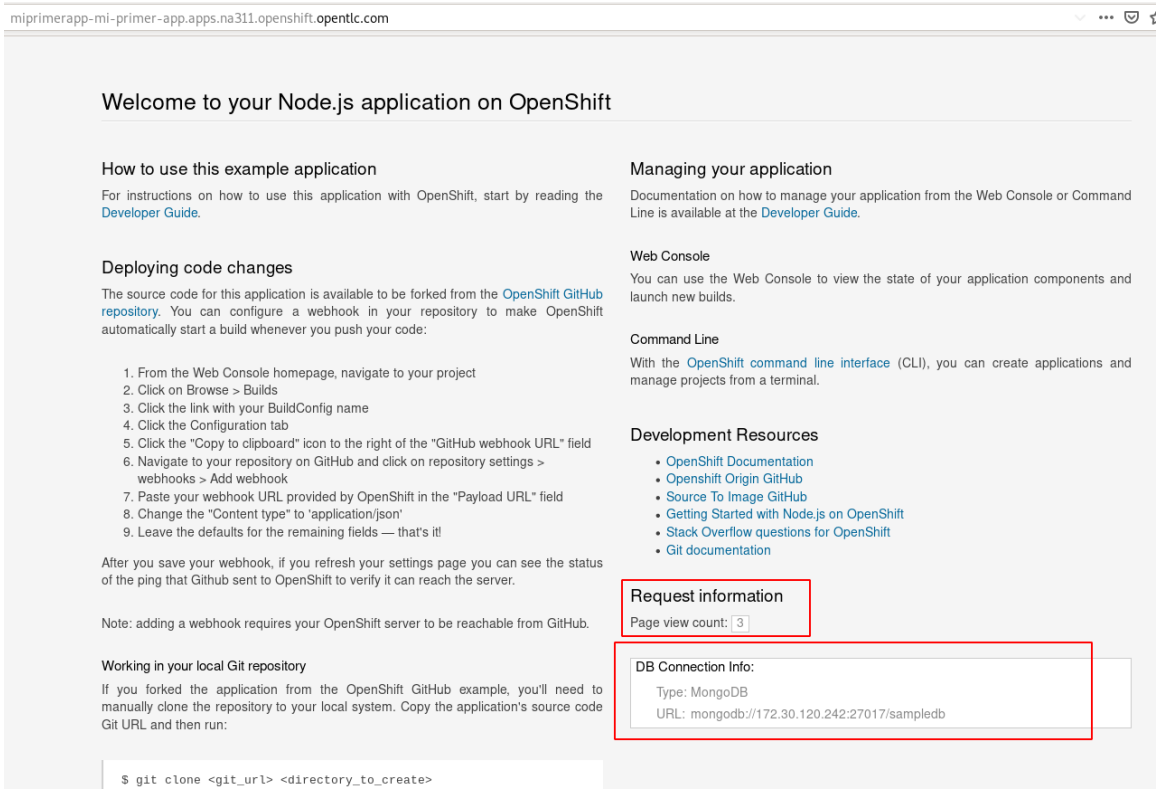
4.19 Le damos Click en “Overview” en la columna izquierda del panel.
Verifique que el número a la par del deployment configuration “miprimerapp” incrementó ya que se cambió la configuración.



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 esperaria 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.



5. Crear una aplicación desde linea de comandos OC.

5.1 Crear un nuevo projecto:
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
```

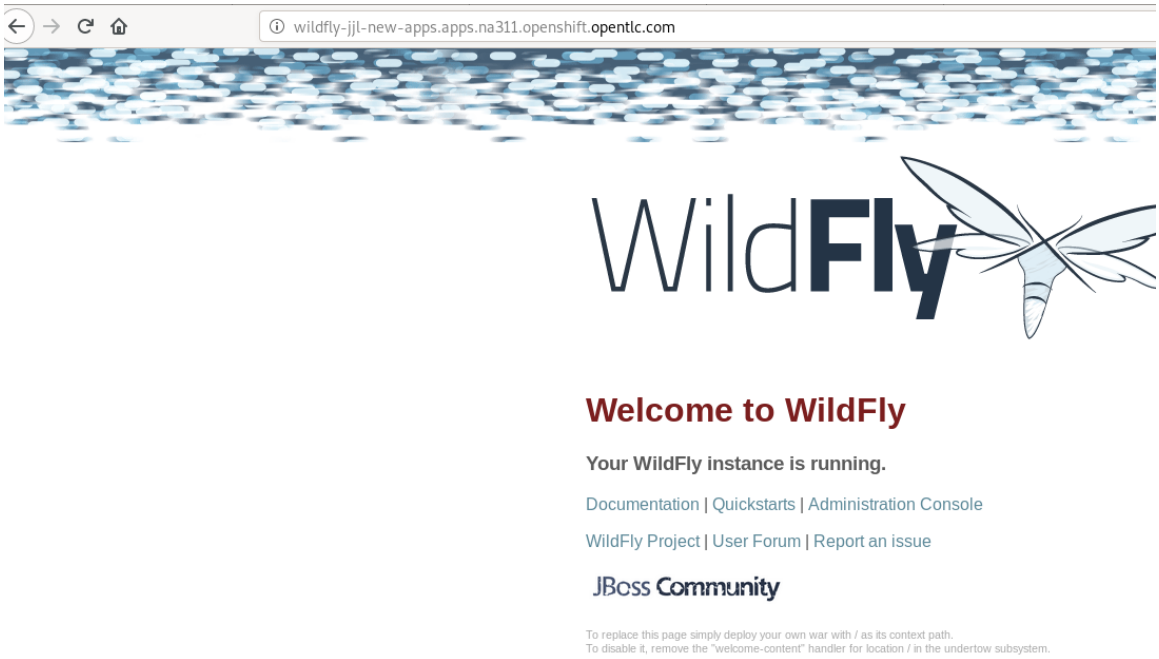
5.4 Los servicios se utilizan para la comunicación interna de OpenShift. De esta manera, otra aplicación no necesita saber la dirección IP real del pod o cuántos pods se están ejecutando para una aplicación determinada. Pero dado que los servicios no son accesibles fuera del clúster de OpenShift, debe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

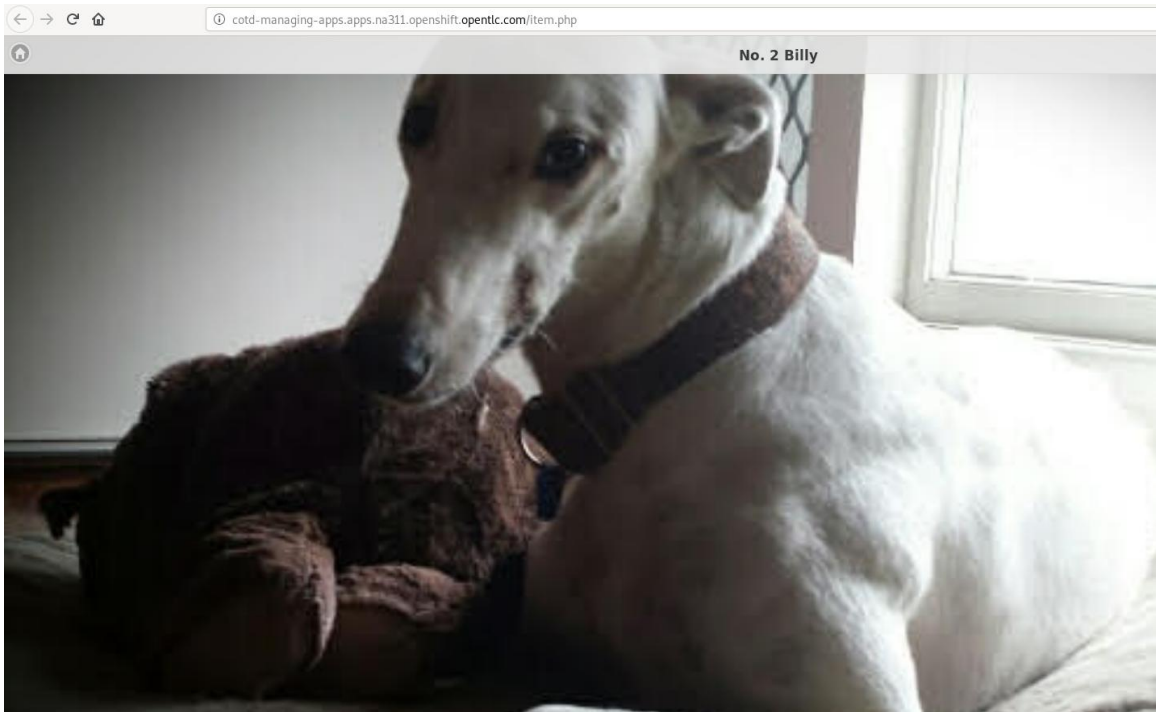
6.4 Revisamos los logs del build.

```
$ 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         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 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"

--> 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:

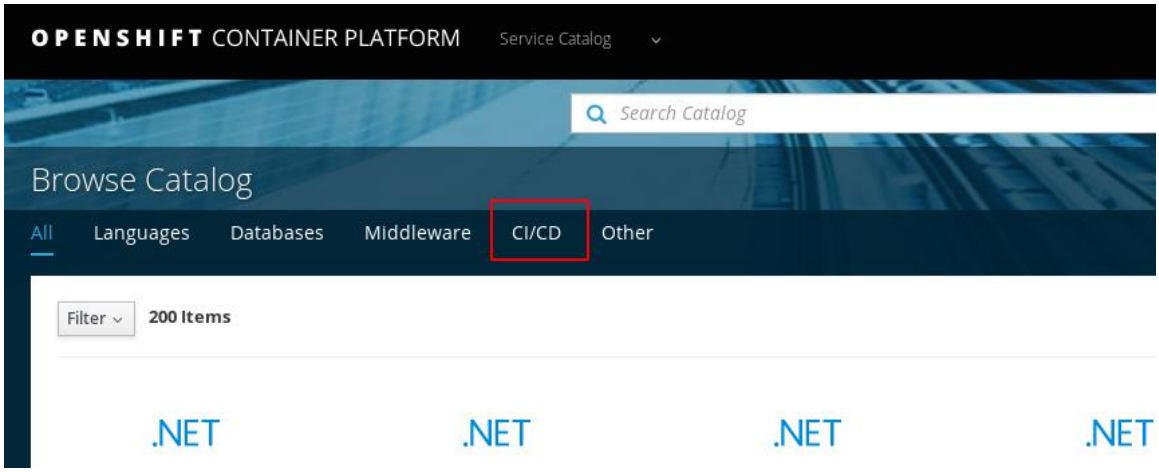
Debe usar curl desde la línea de comandos porque todos los navegadores modernos usan cookies para identificar su sesión actual. Y cada vez que actualiza el navegador, se lo envía al mismo servicio al que se lo envió antes, lo cual tiene sentido, porque en el mundo real cuando realiza pruebas A / B, desea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

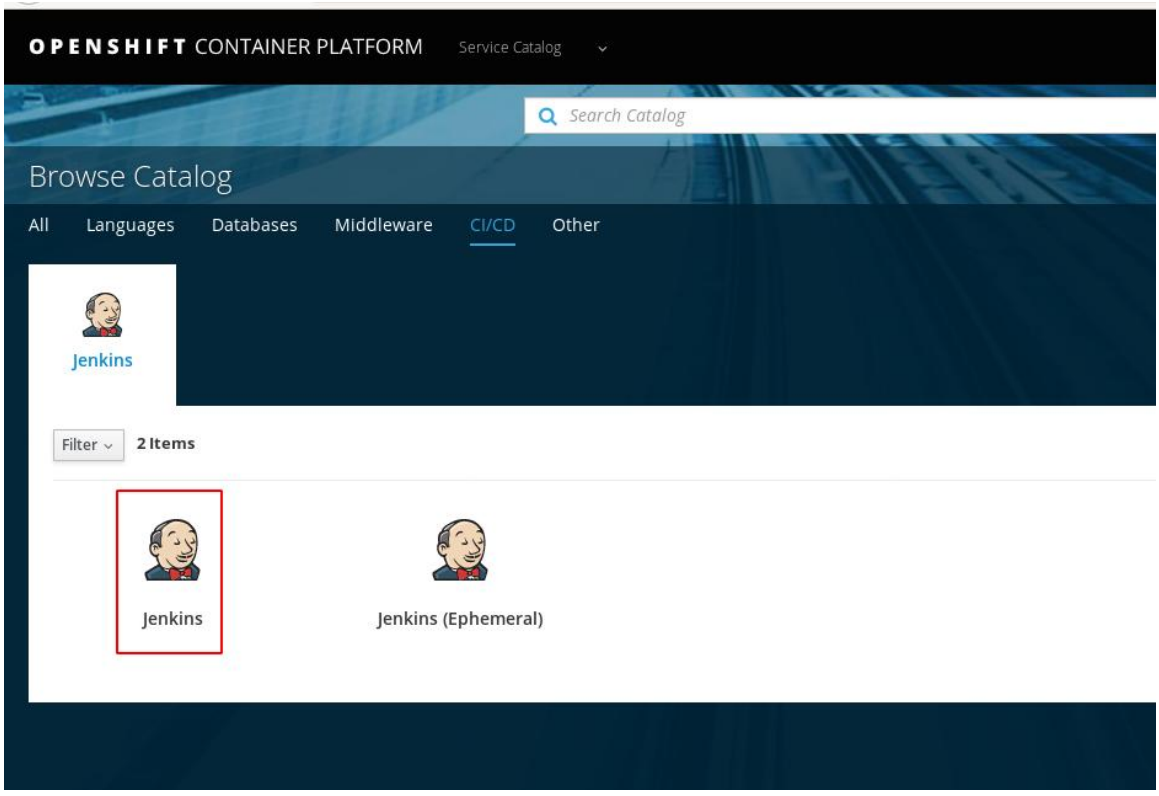
```
$ while true; do curl -s http://$(oc get route cotd --template='{{ .spec.host }}')/item.php
| grep "data/images" | awk '{print $5}'; sleep 1; done
data/images/pets/billie.jpg
data/images/cities/adelaide.jpg
data/images/pets/taffy.jpg
data/images/cities/canberra.jpg
data/images/pets/milo.jpg
data/images/cities/perth.jpg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

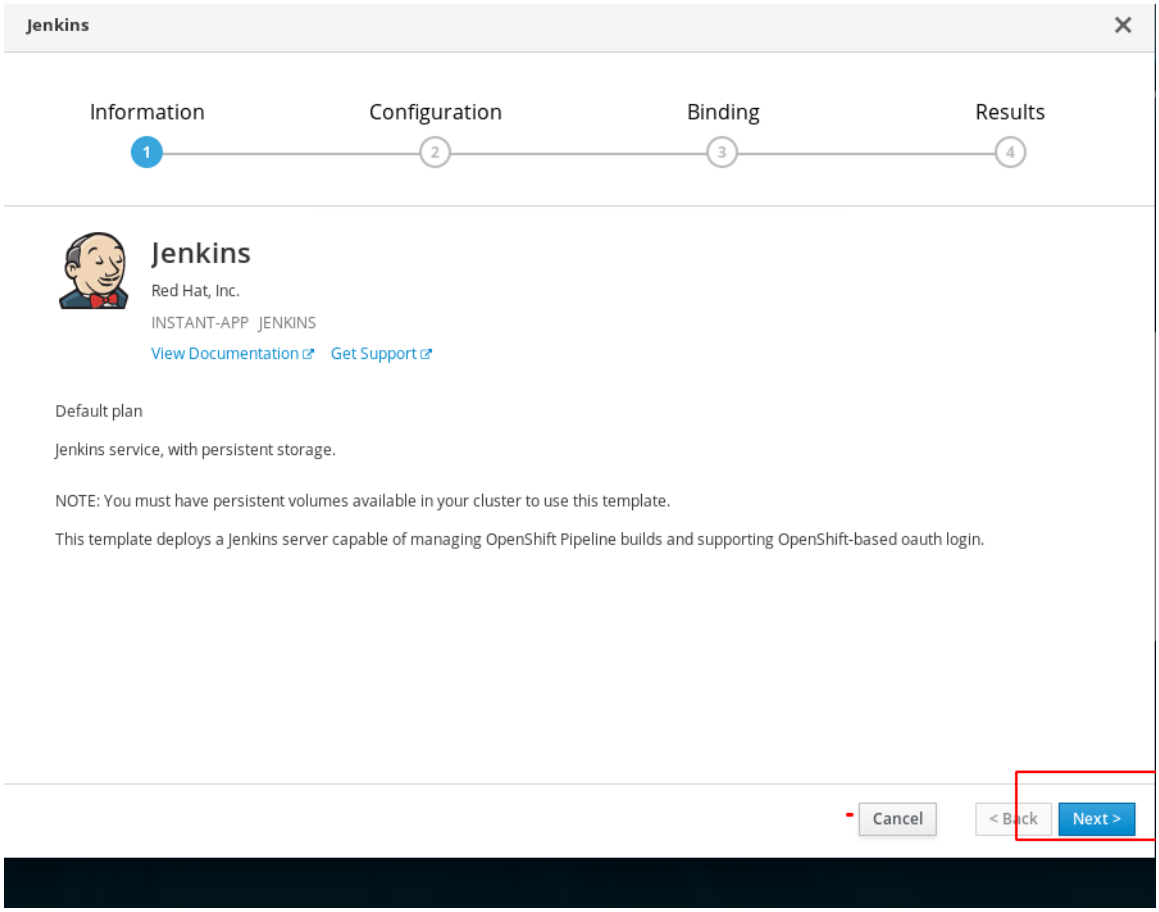
7. En esta práctica de laboratorio, explora un entorno simple de CI / CD utilizando Jenkins como el servidor de CI / CD. Crea un contenedor Jenkins y luego usa el servidor Jenkins para construir e implementar una aplicación Java EE simple. Configura la aplicación en un proyecto separado y prepara el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

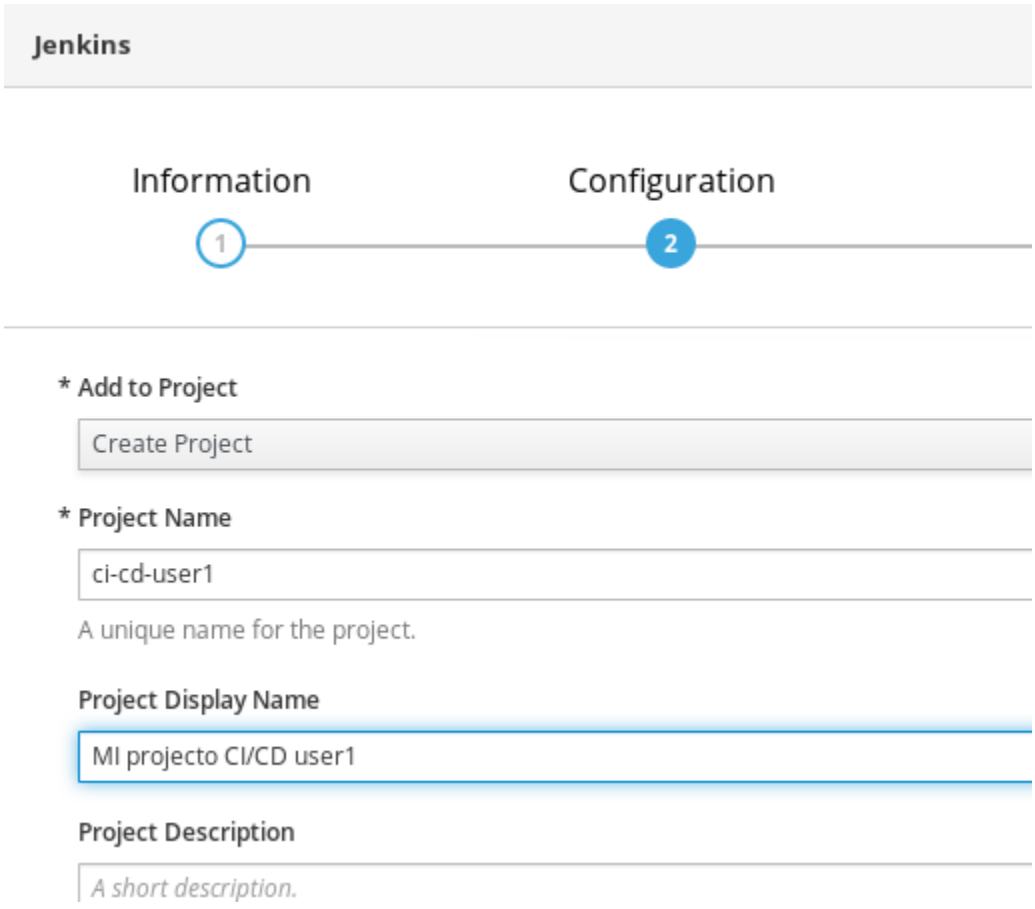


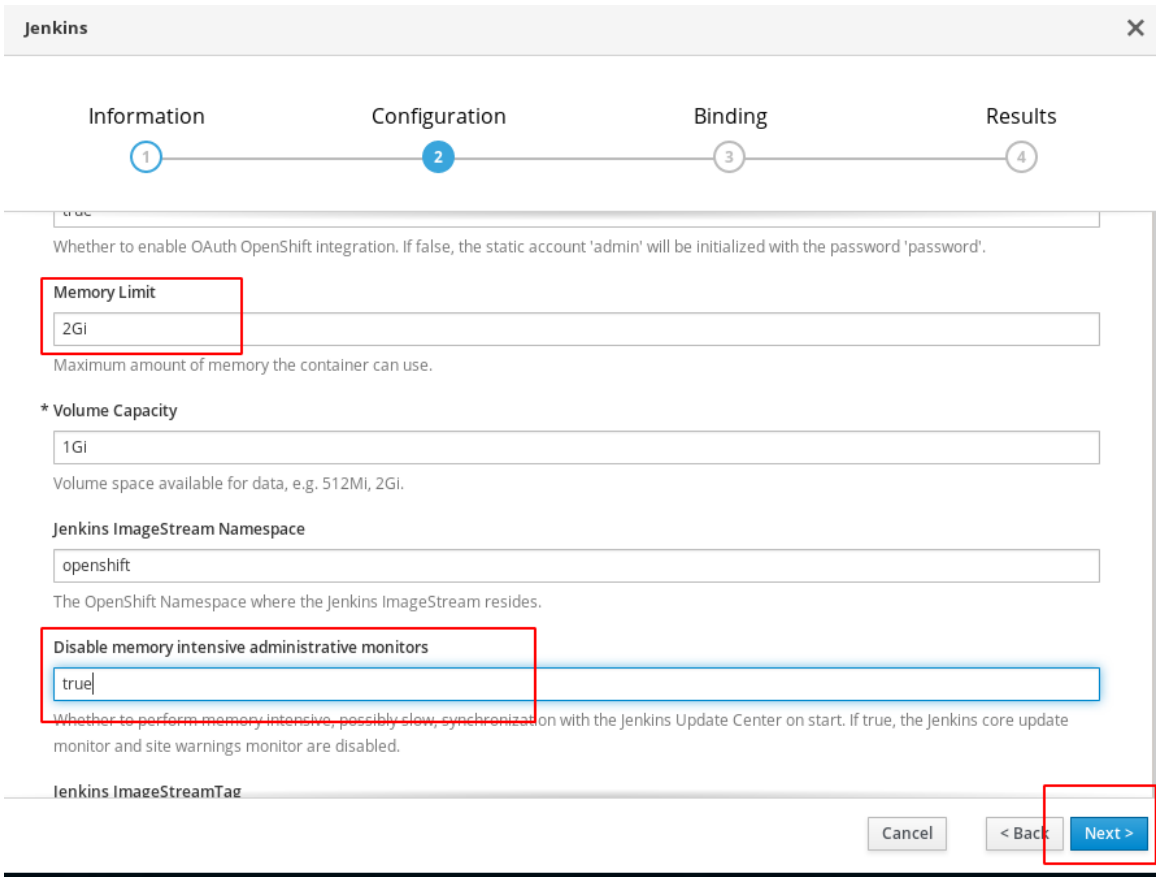


Le damos click en “Next”.

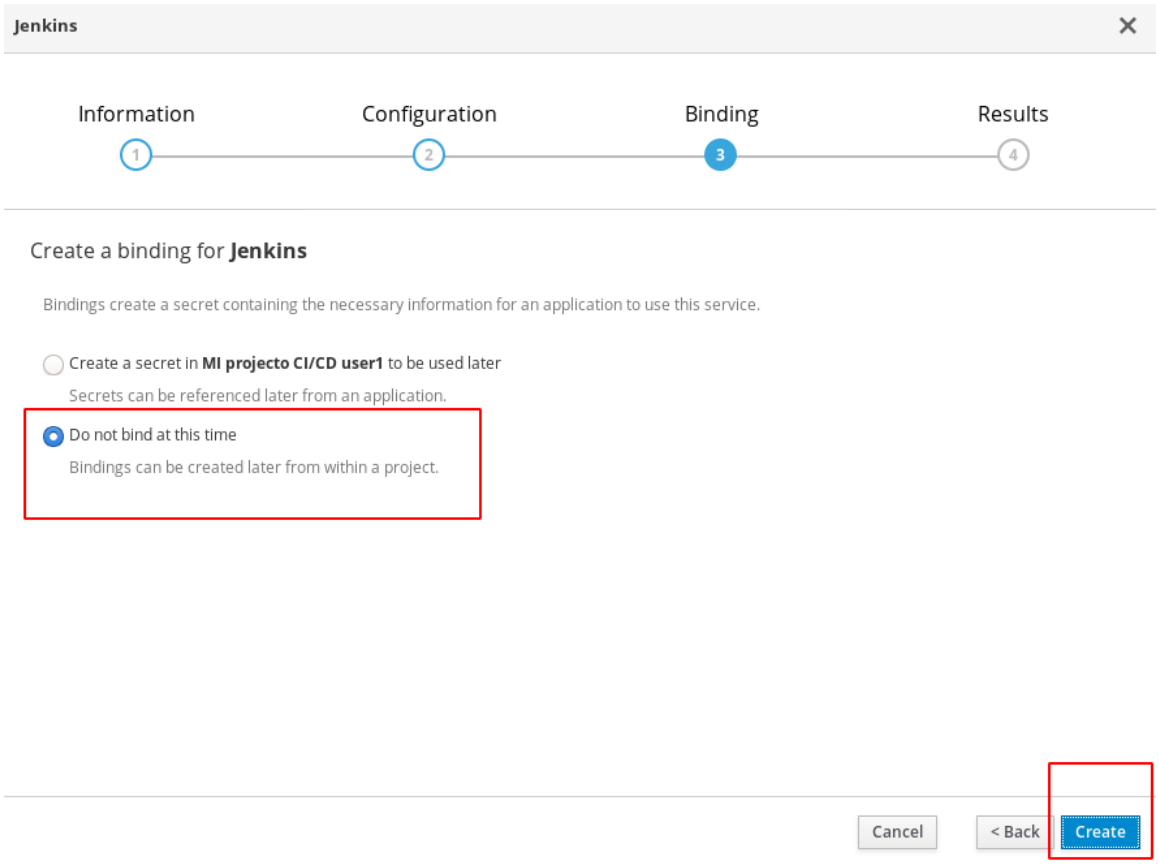


7.2 Se debe crear un nuevo proyecto y cambiar lo siguiente:
Create Project
Project Name: userx-ci-cd
Project Display Name: Mi Projecto CI/CD - userx
Memory Limit: 2Gi
Disable memory intensitive administrative monitors: true
Click en “Next” para continuar con la siguiente página.

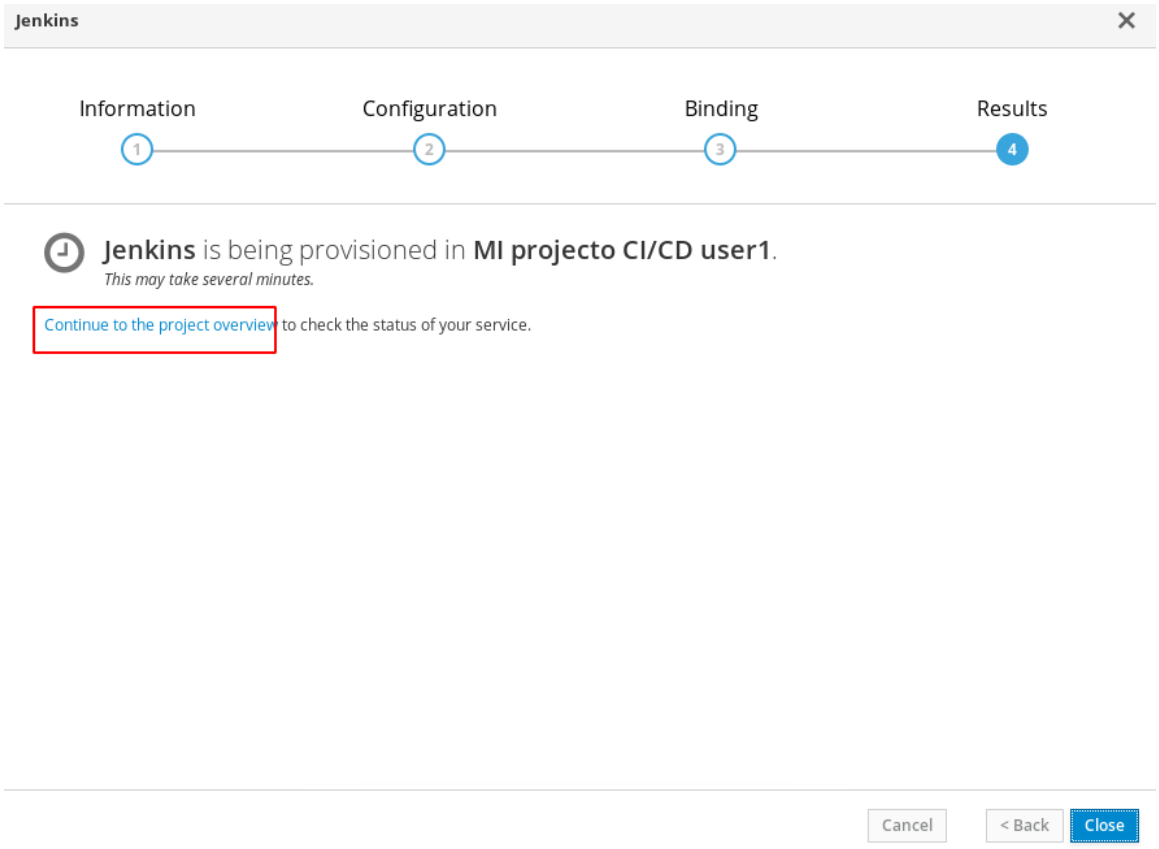



Jenkins
Information
Configuration
Binding
Results
1
2
3
4
Whether to enable OAuth OpenShift integration. If false, the static account 'admin' will be initialized with the password 'password'.
Memory Limit
2Gi
Maximum amount of memory the container can use.
* Volume Capacity
1Gi
Volume space available 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 >

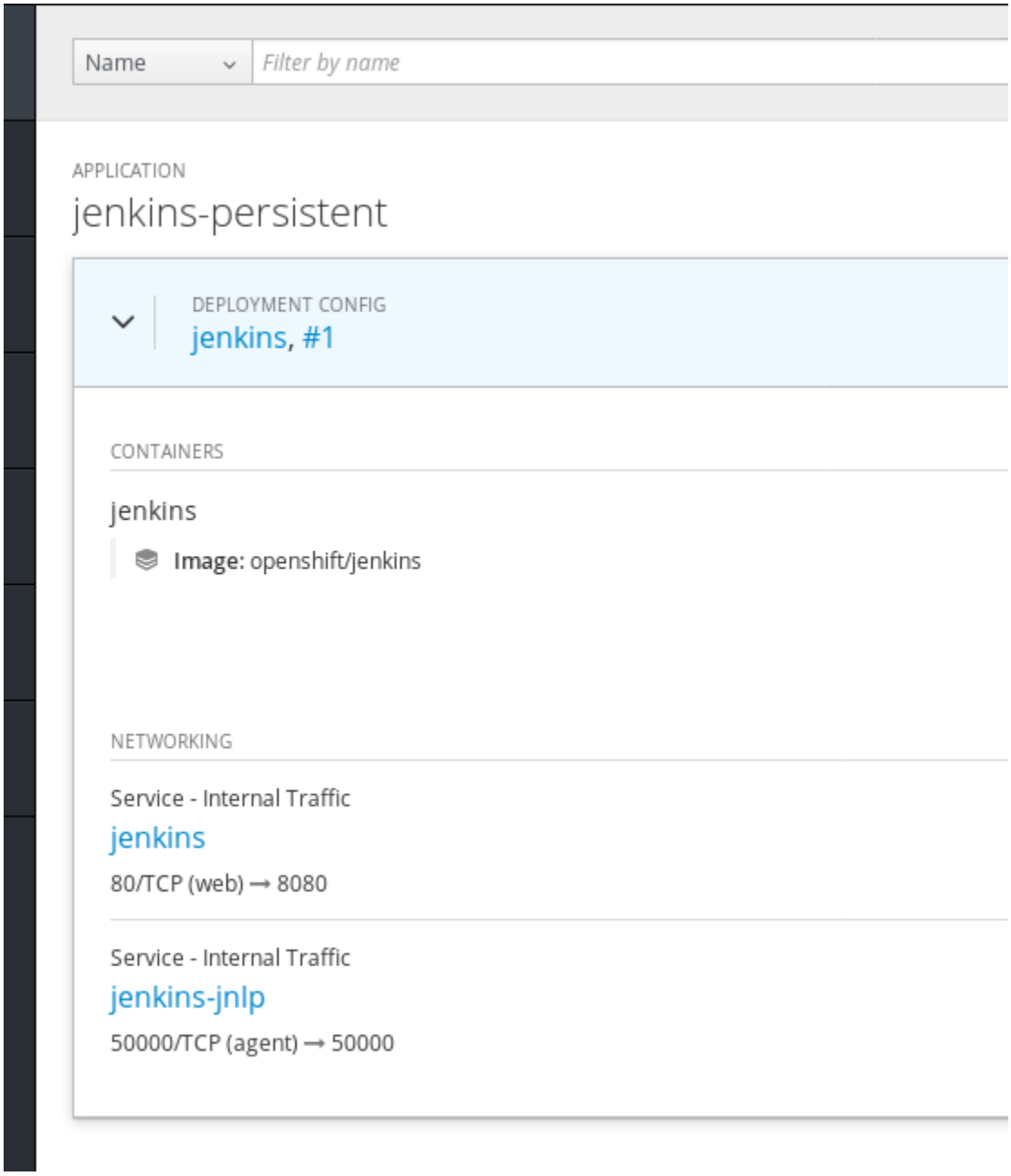
7.3 En la página de Bindings, dejar “Do not bind at this time” y seleccionar y click en Crear.

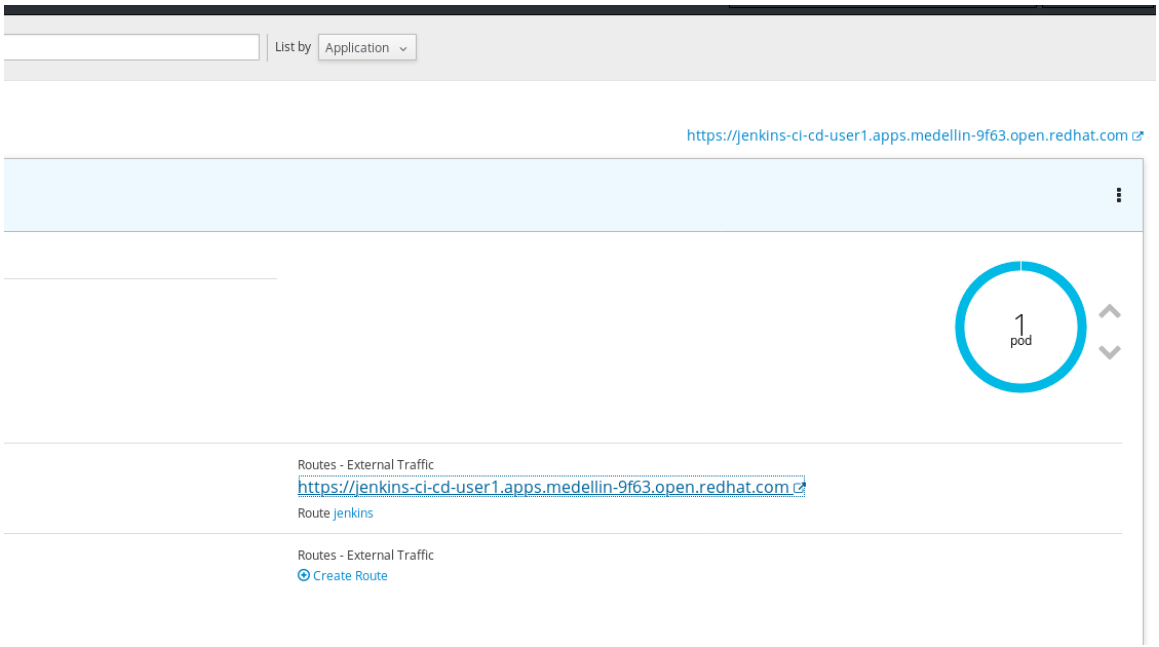


En la página de resultados, click en “Continue to Project overview”.

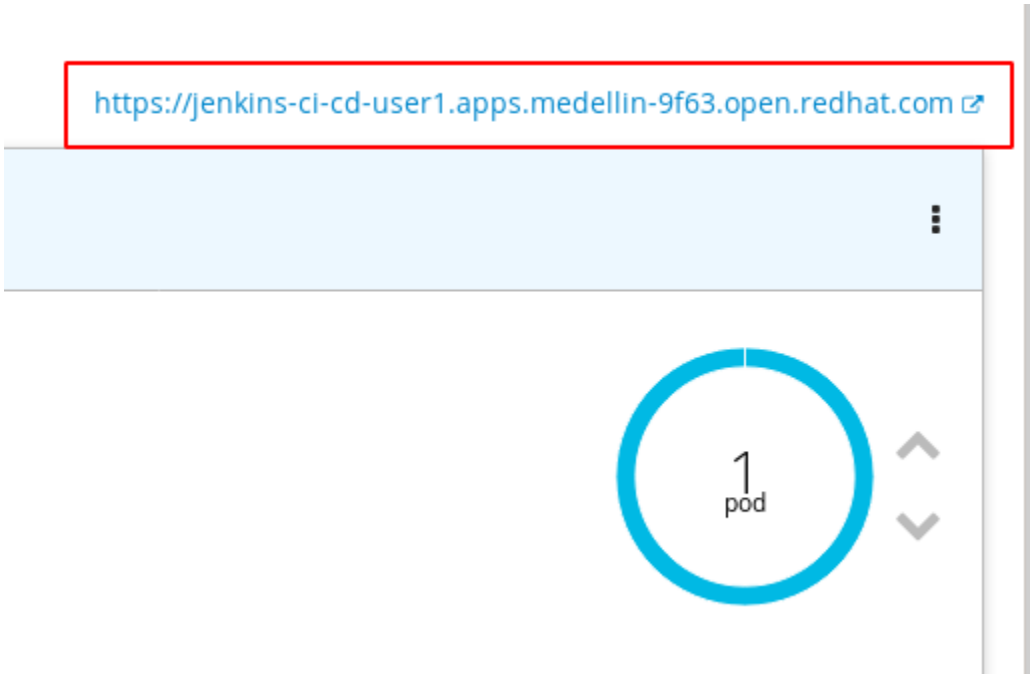


7.4 Abrir el jenkins deployment y esperar a que el Jenkins esté totalmente arriba y funcional, debe mostrarse de la siguiente forma:
Para que esto sea asi, debemos esperar varios minutos.

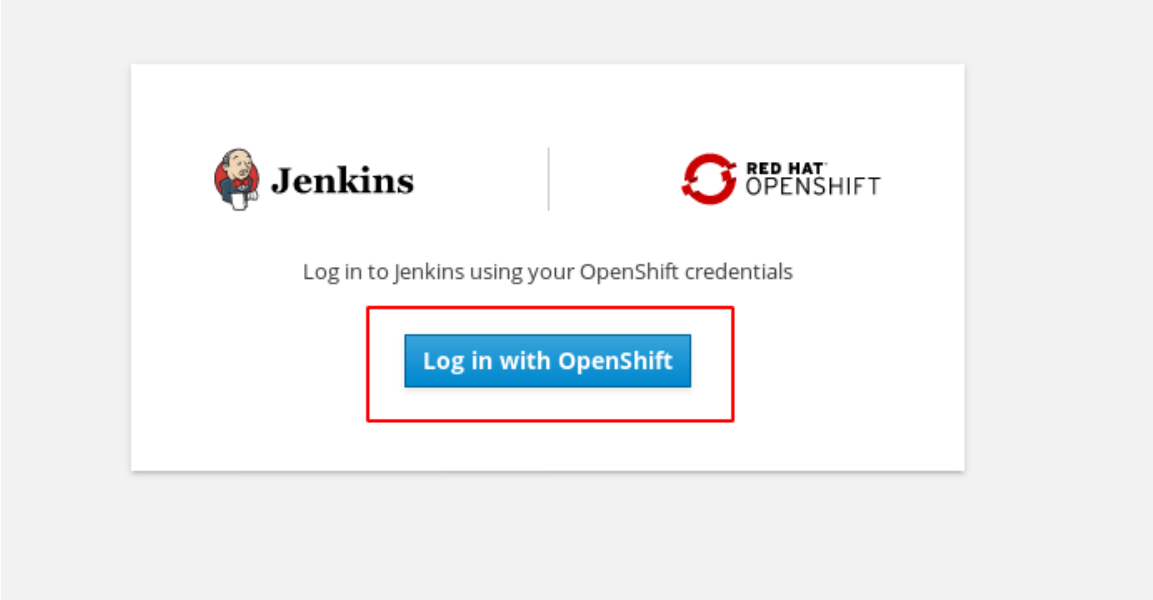




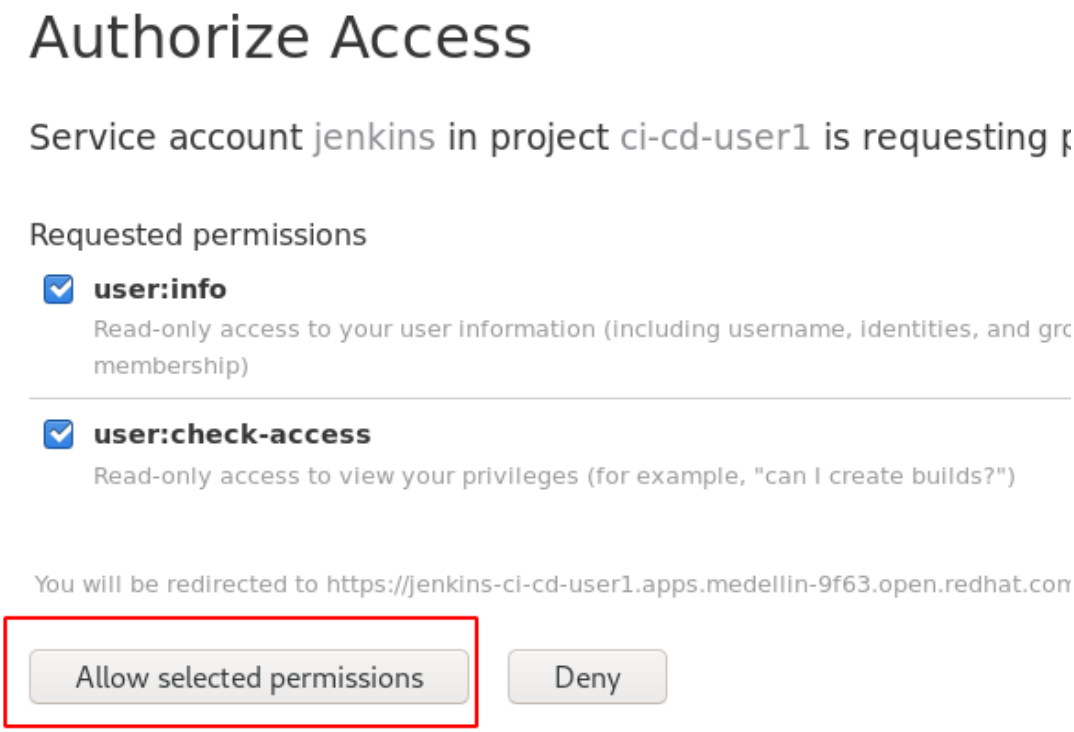
Le damos click en la ruta de acceso de Jenkins.

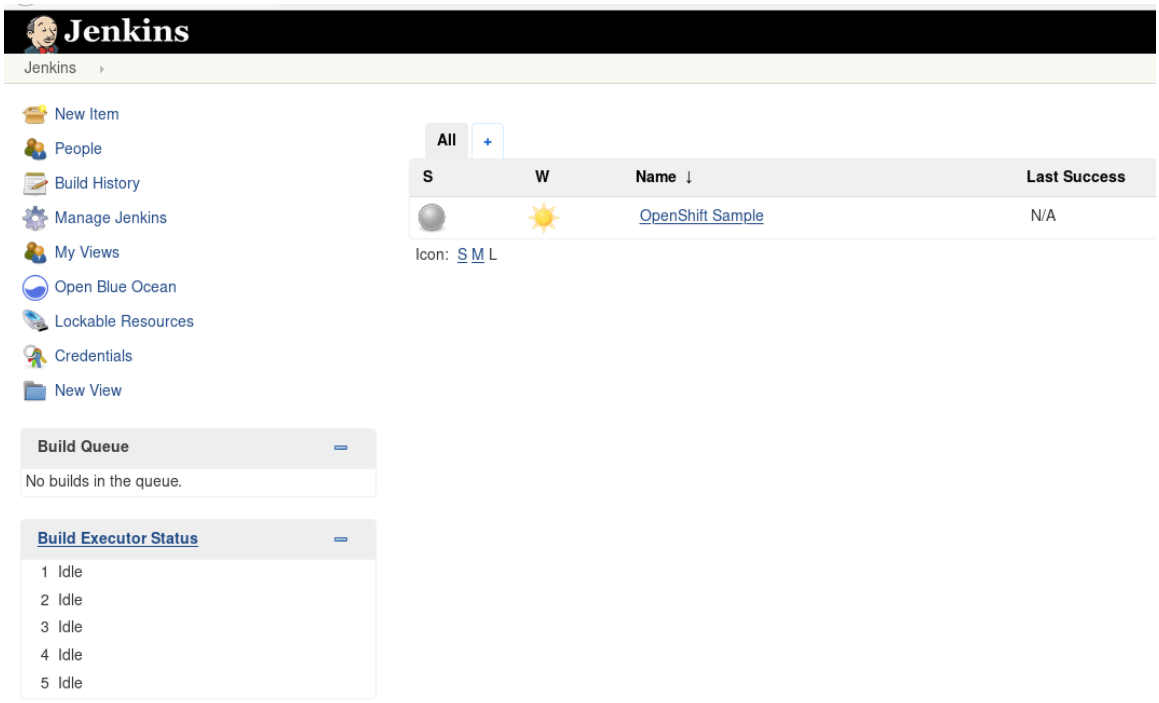


7.5 Al abrir la ruta de Jenkins, debemos aceptar el certificado y darle click en “Login with Openshift”, para abrir el Jenkins.

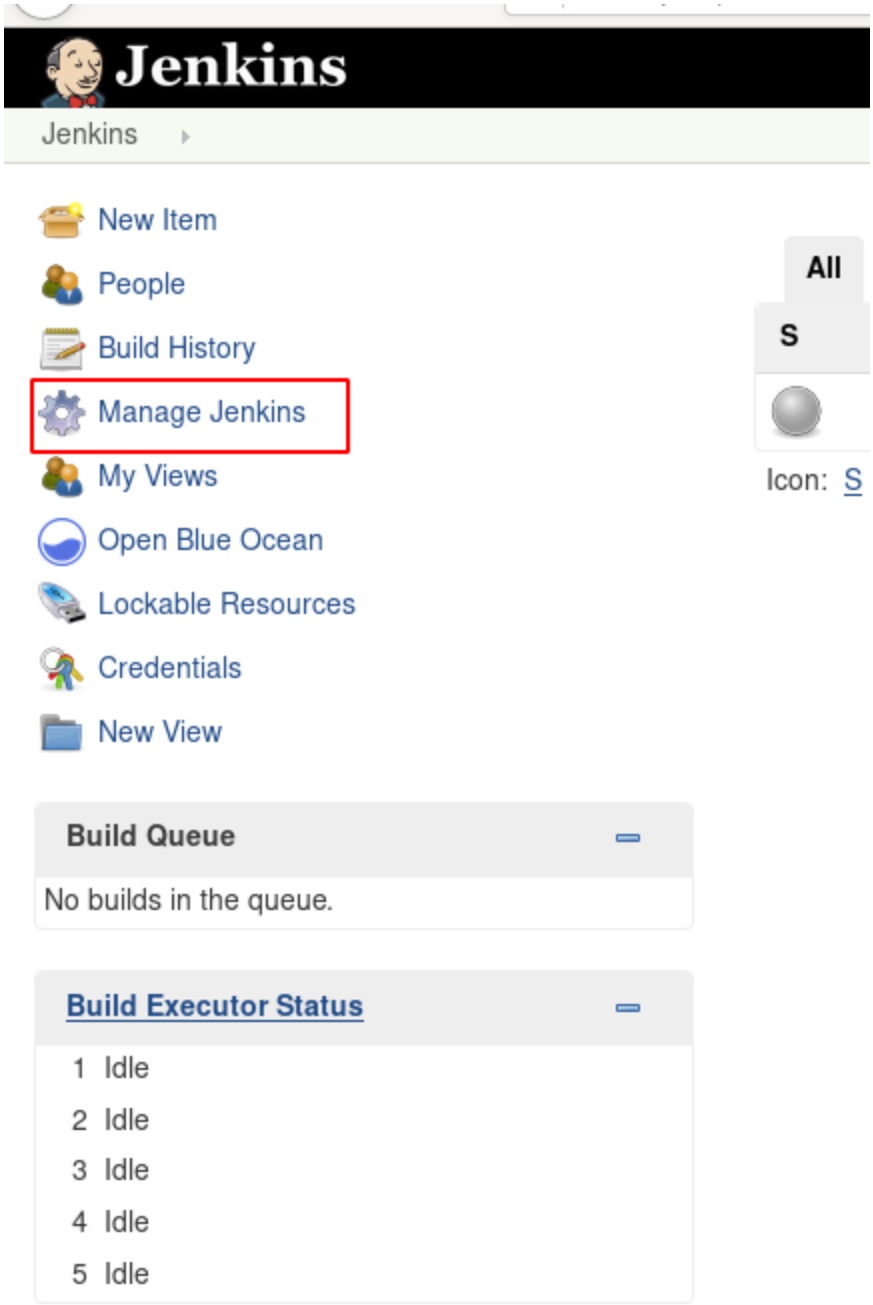


Si nos pide usuario y password, debemos usar los mismos que utilizamos para loguernos a la consola web de openshift.

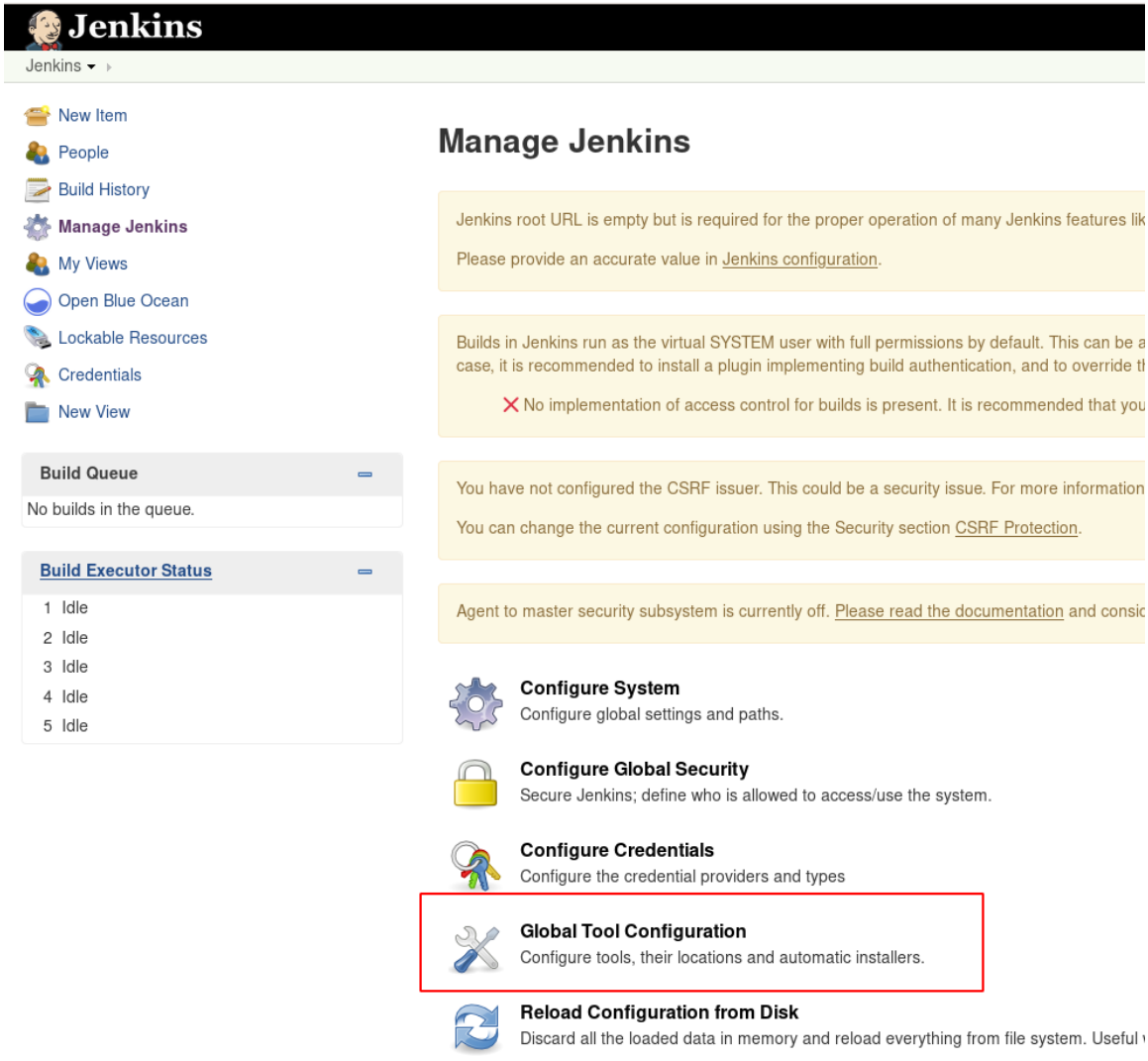



Jenkins
Jenkins
New Item
People
Build History
Manage Jenkins
My Views
Open Blue Ocean
Lockable Resources
Credentials
New View
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
3 Idle
4 Idle
5 Idle
All
+
S
W
Name ↓
Last Success
OpenShift Sample
N/A
Icon: S M L

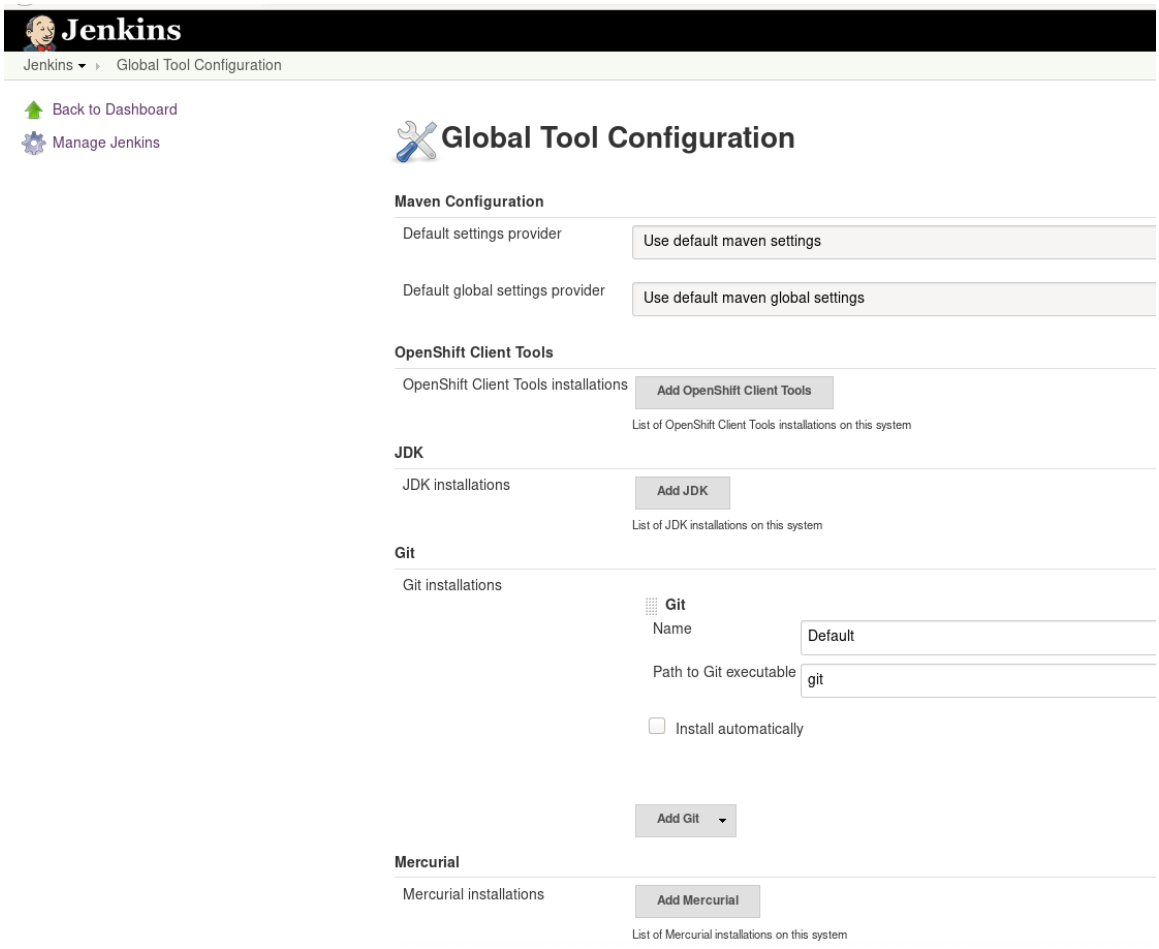
7.6 Debemos activar el maven en el Jenkins esto para poder hacer despliegue de aplicaciones Java.
Para esto le damos click desde el Jenkins donde dice “Manage Jenkins”.



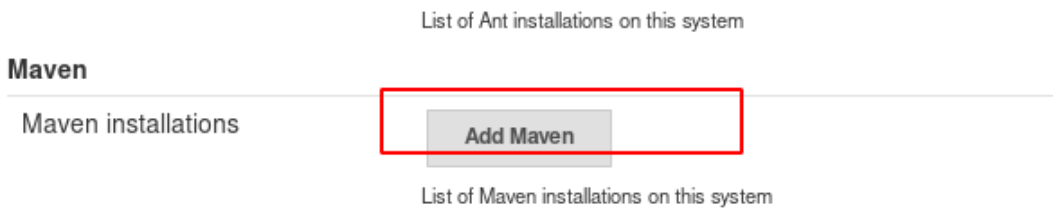
Le damos click en “Global Tool Configuration”



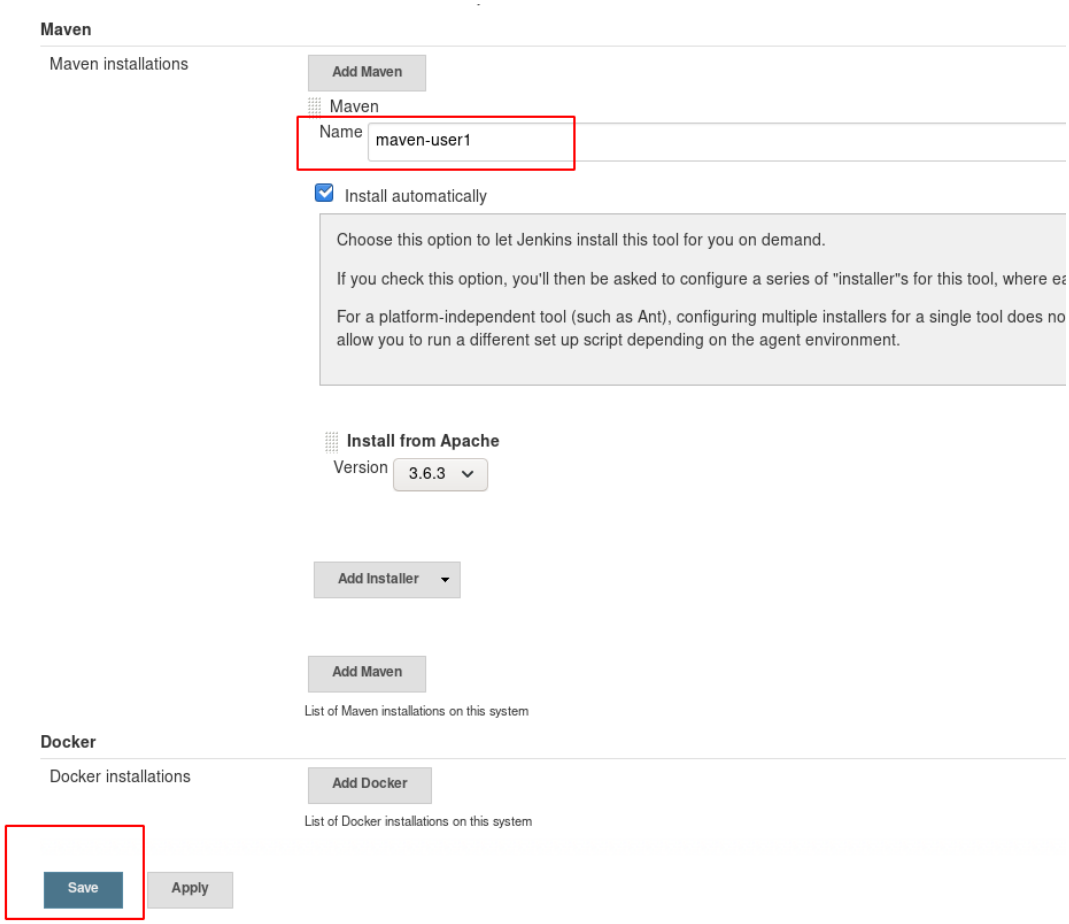
Vamos a ver una imagen como la siguiente.



Buscamos **Maven** y lo agregamos:



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



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 ~]$
```

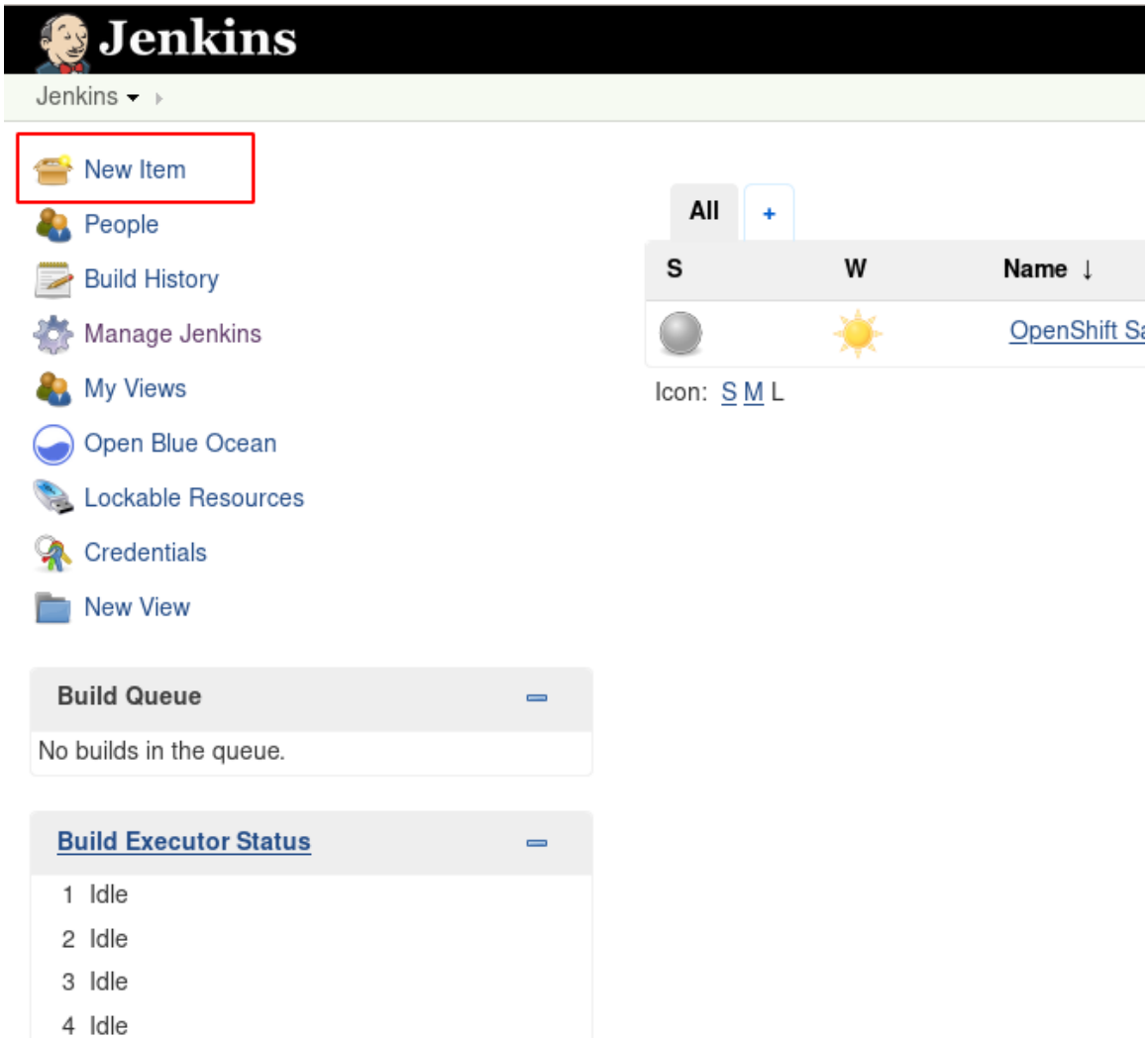
7.11 Otorgue a la cuenta de servicio los permisos correctos para editar objetos en este proyecto para permitir que Jenkins construya e implemente la aplicación.

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**

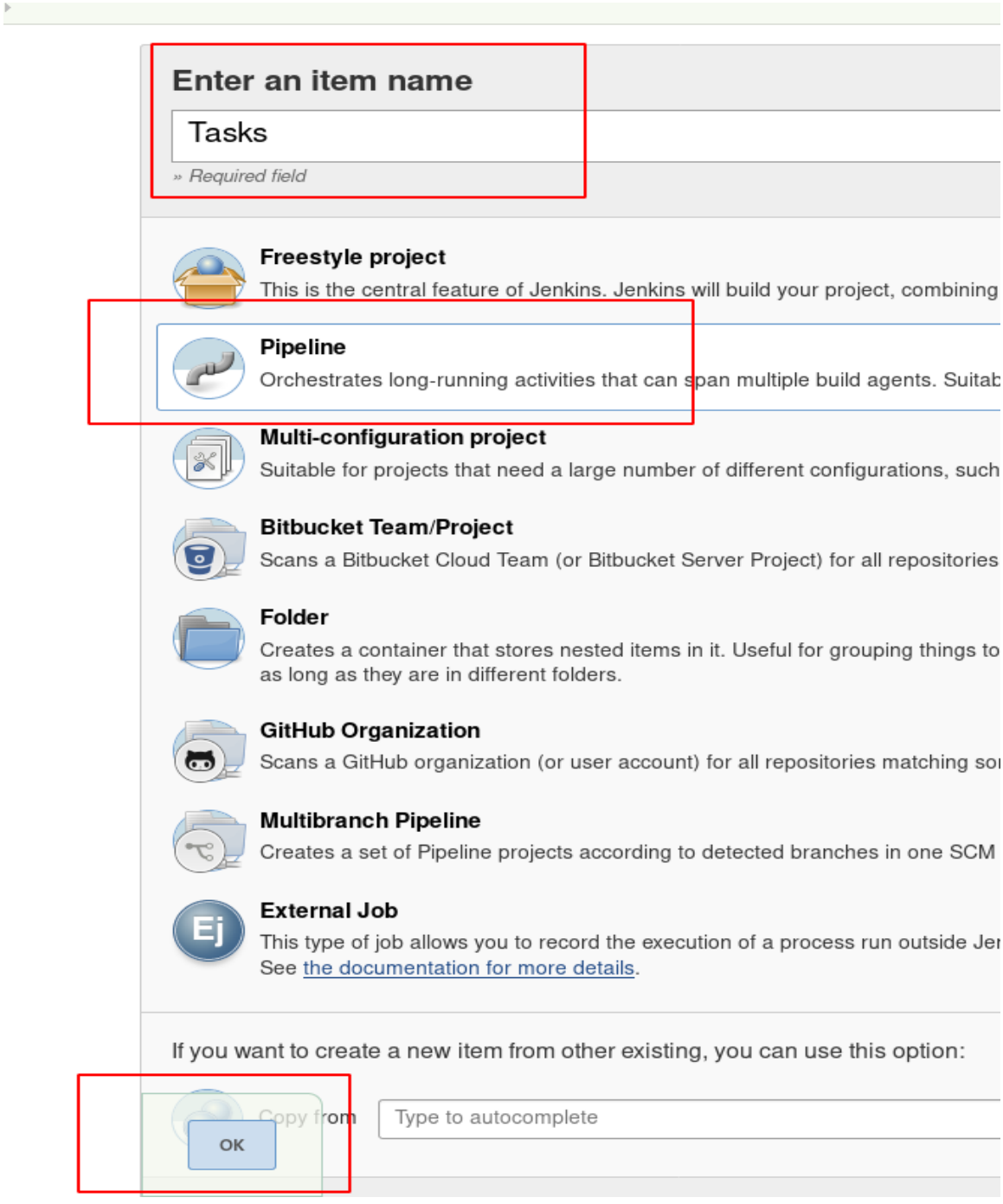
```
[default@wetty-1-9nclg ~]$ oc policy add-role-to-user edit system:serviceaccount:ci-cd-
user1:jenkins -n user1-cicd-tareas
role "edit" added: "system:serviceaccount:ci-cd-user1:jenkins"
[default@wetty-1-9nclg ~]$
```

7.12 Crear pipeline en Jenkins.

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



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

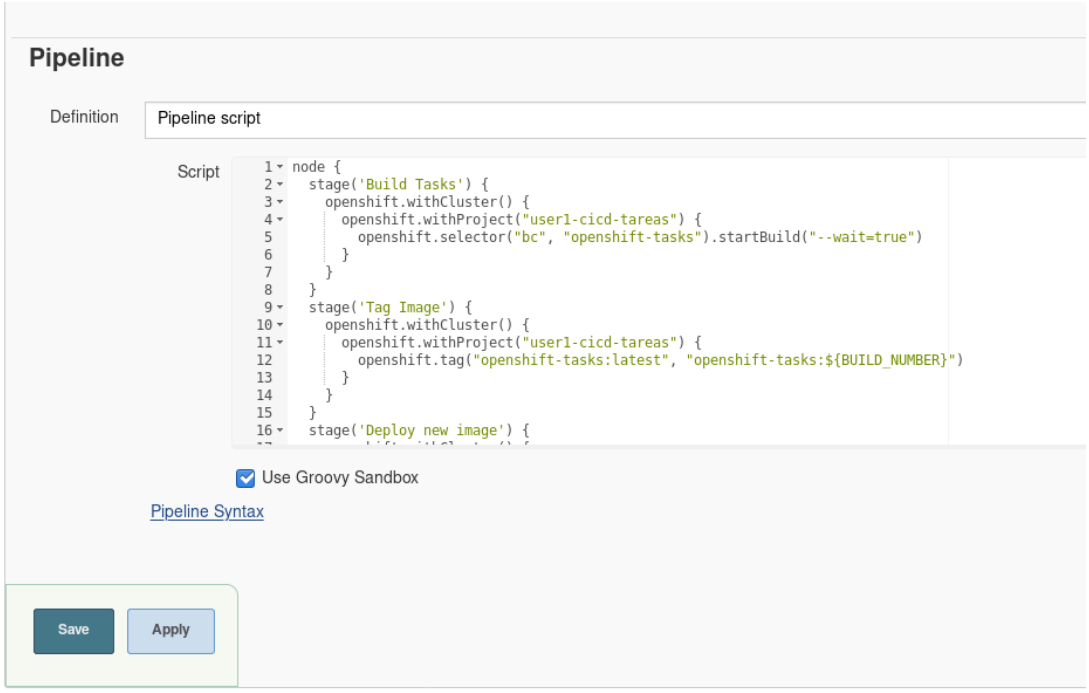


7.14 Pegar el siguiente contenido en la parte de Pipeline script.

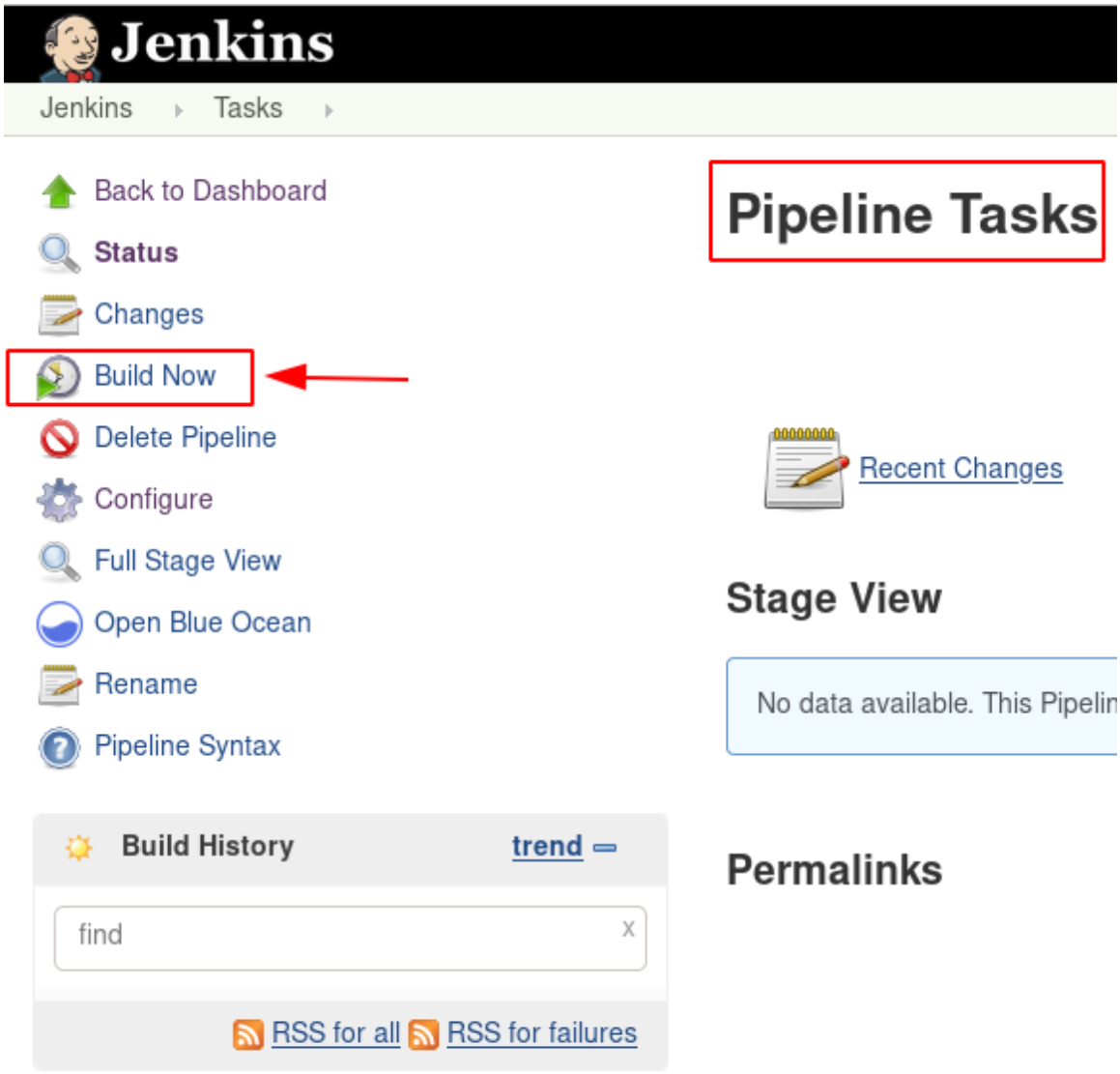
```
node {
  stage('Build Tasks') {
    openshift.withCluster() {
      openshift.withProject("user1-cicd-tareas") {
        openshift.selector("bc", "openshift-tasks").startBuild("--wait=true")
      }
    }
  }
  stage('Tag Image') {
    openshift.withCluster() {
      openshift.withProject("user1-cicd-tareas") {
        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.

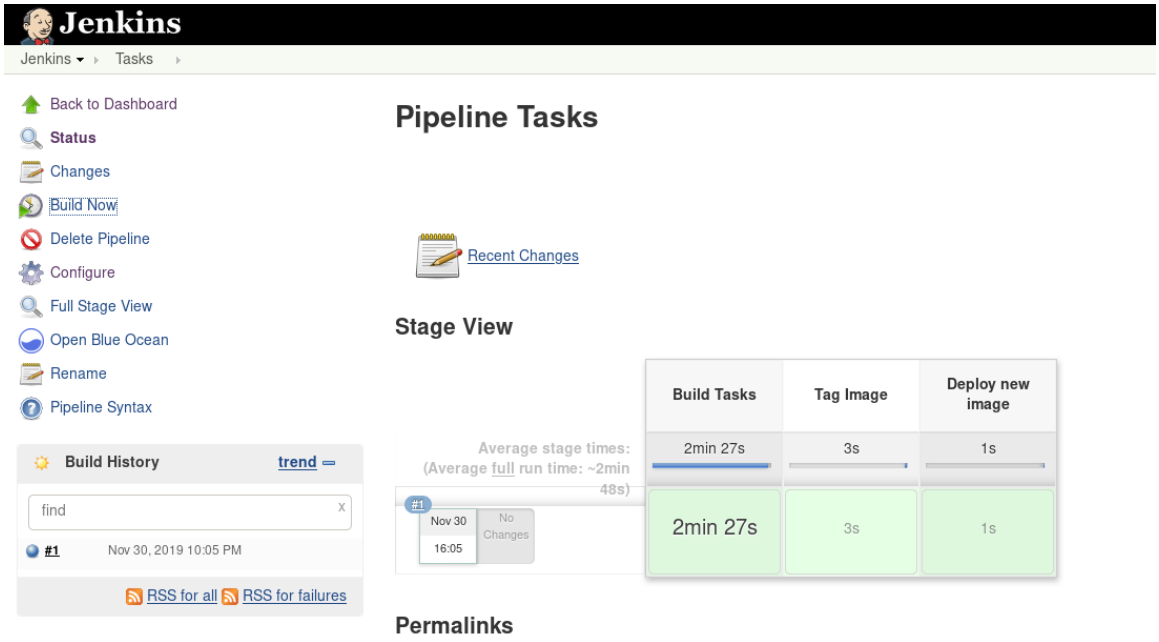
Y hay que darle click en “Save” a este pipeline job.



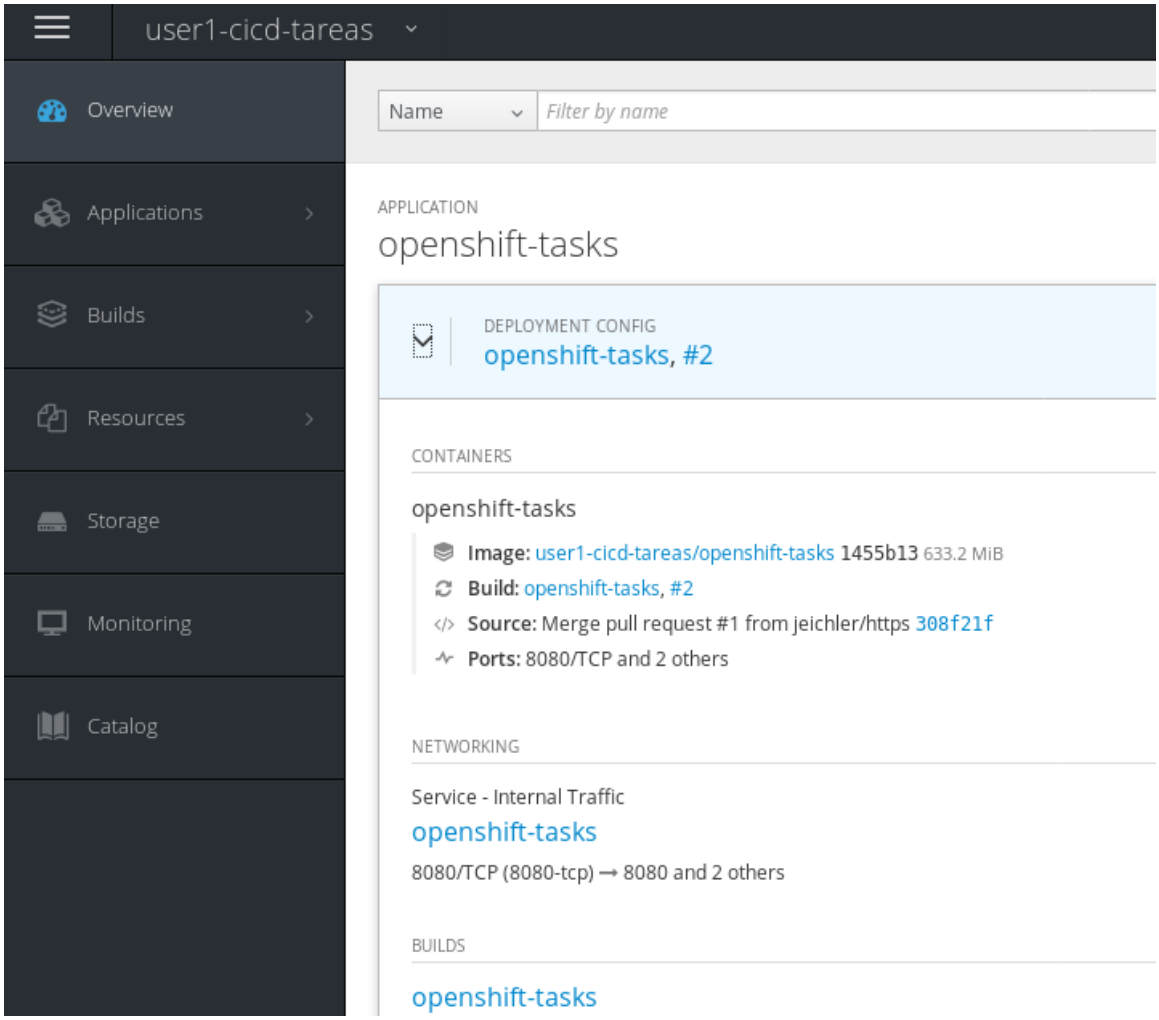
7.15 En la página de Jenkins, click en “Build Now” .



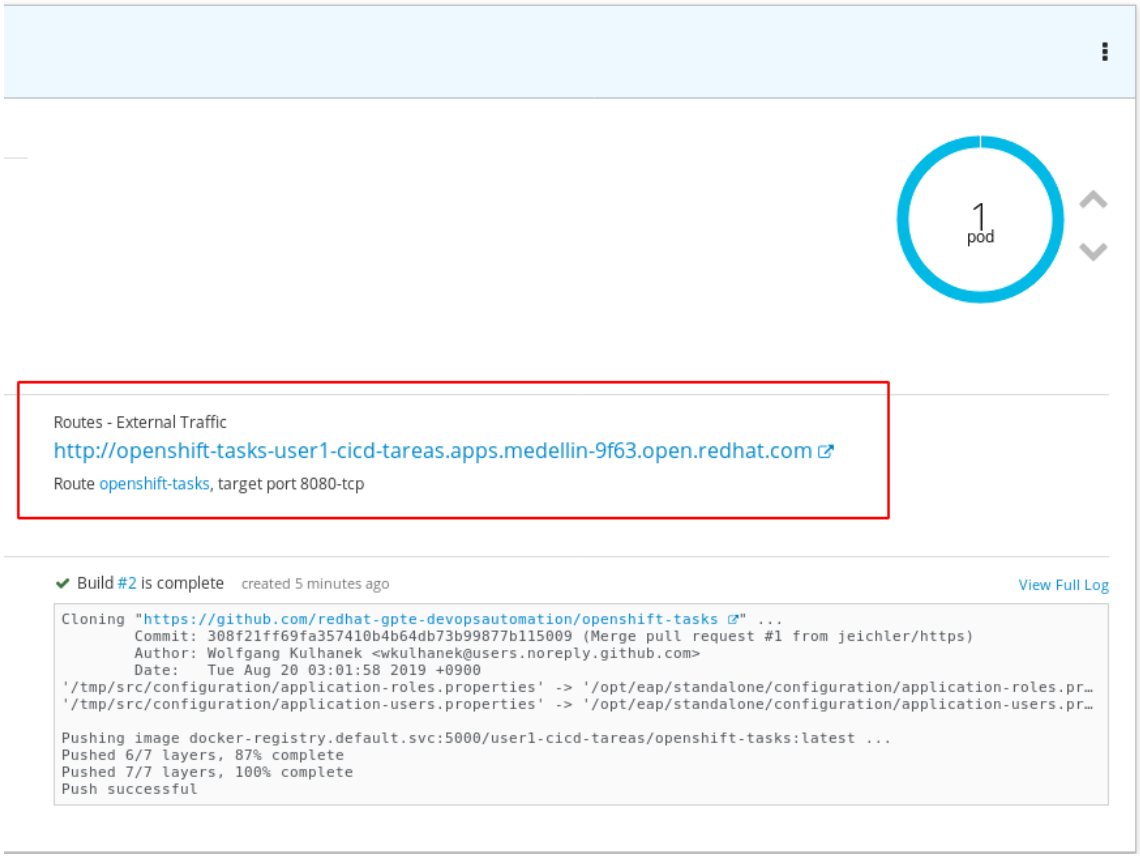
Nos esperamos a que se realice el build de forma correcta, el mismo dura alrededor de 3 minutos.



7.16 Ya con el build realizado de forma correcta, ingresamos a la consola web de Openshift, abrimos el projecto 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



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



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/