

# CI/CD Pipelines en Openshift

## Preparación

Haga un fork del repositorio <https://github.com/everdes-ibm/cotd.git> a su repositorio personal.

Debe tener instalado git en su equipo.

Todo el laboratorio se hará utilizando la consola web de Openshift.

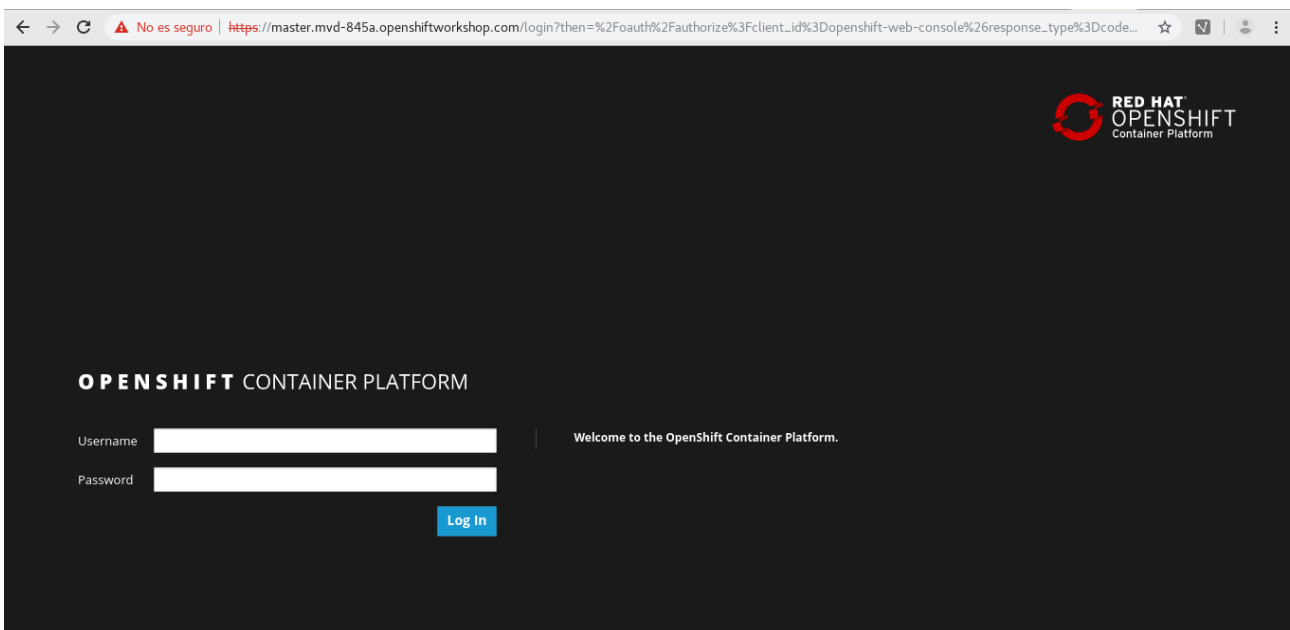
Los ejemplos se han basado en el contenido del libro “Devops With Openshift” que está disponible gratuitamente en <https://www.openshift.com/devops-with-openshift/>

Para disponer un entorno de pruebas puede utilizar la máquina virtual OpenshiftLab que está en el repositorio <https://github.com/everdes-ibm/cicdpipeline>.

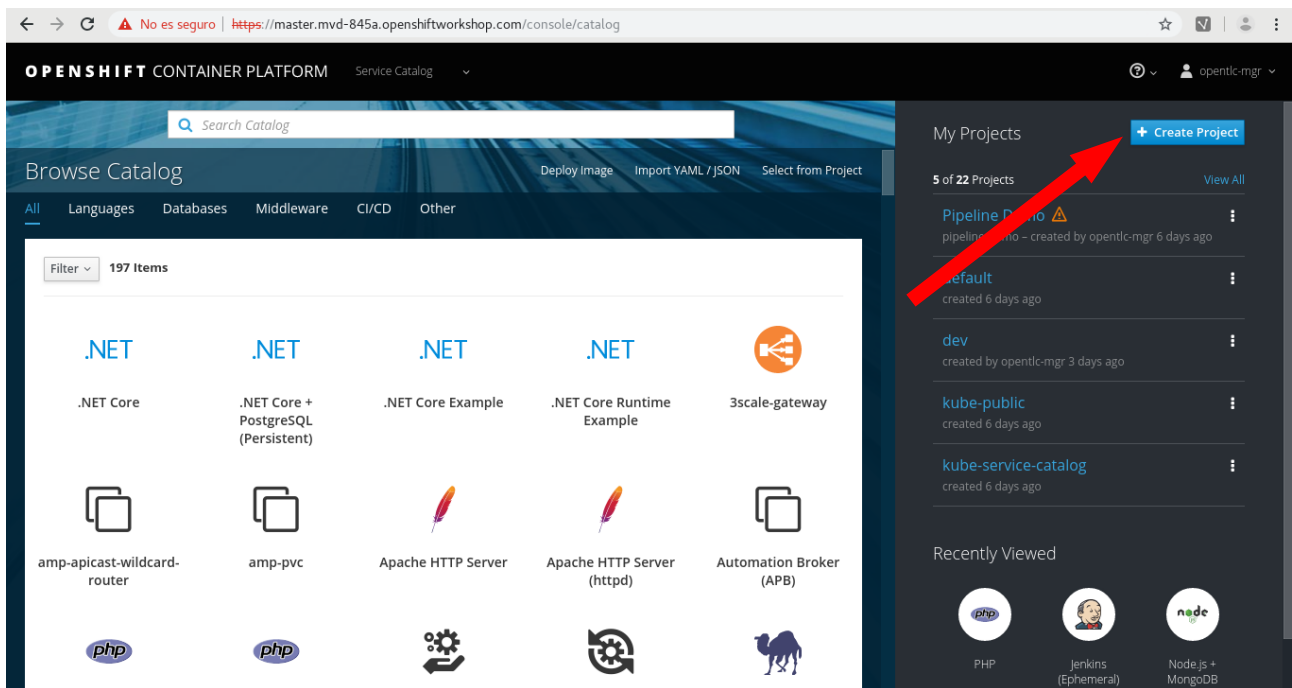
El usuario es admin y la contraseña c3nt0s1!

## Hands-on

Inicie sesión en la consola de Openshift <https://master.mvd-eb44.openshiftworkshop.com/console> con el usuario user1 a user200 y la contraseña openshift



Para crear un proyecto presione el botón [+ Create Project] en la esquina superior derecha.



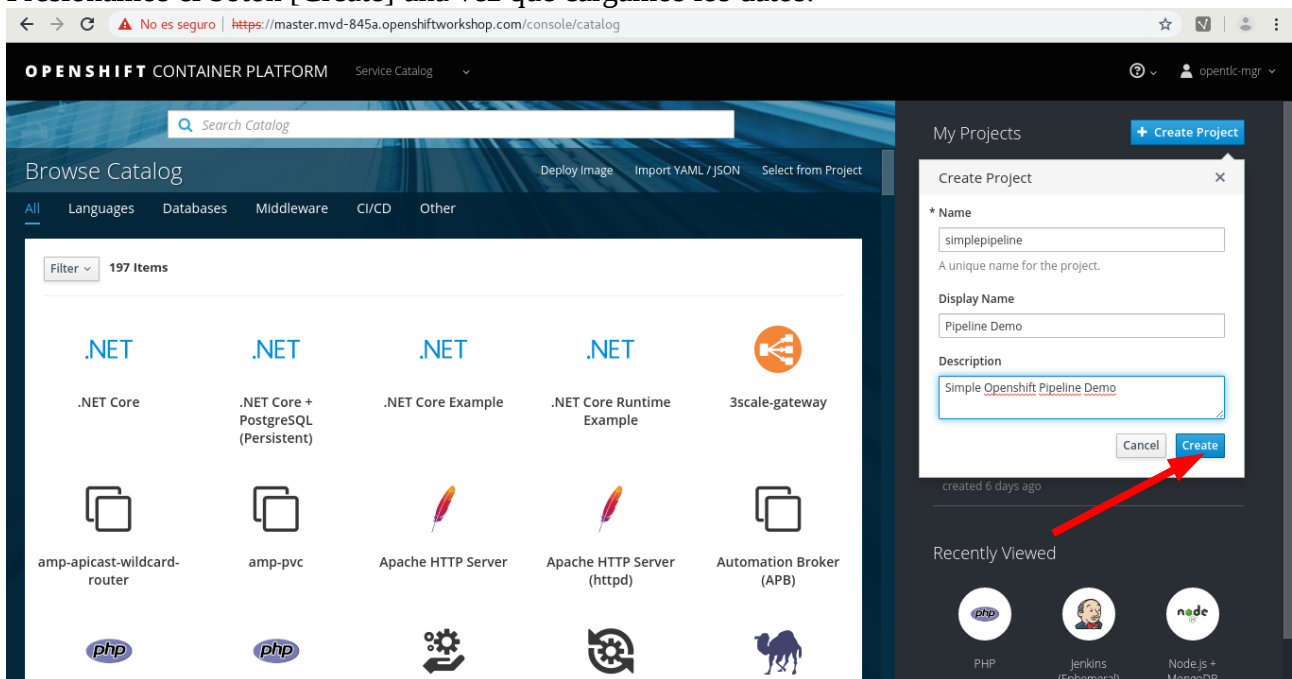
Los datos para el nuevo proyecto son

-Nombre: simplepipeline

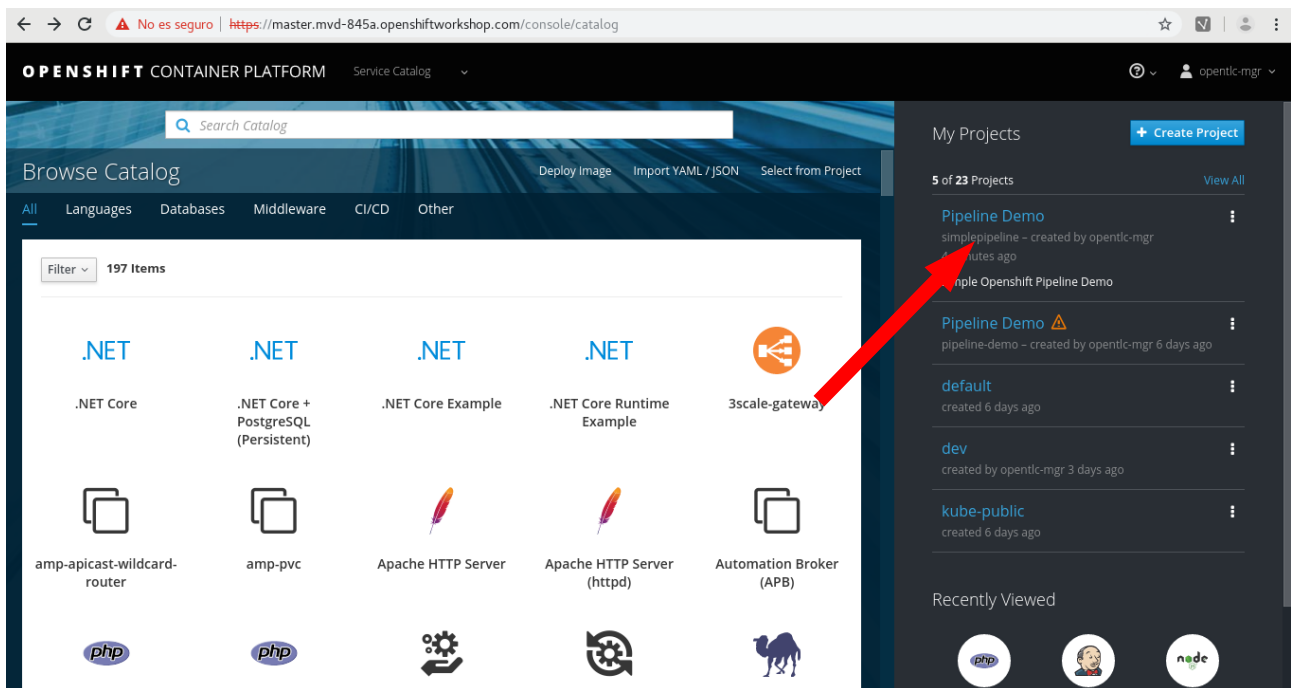
-Display name: Pipeline Demo

-Descripción (opcional): Simple Openshift Pipeline Demo

Presionamos el botón [Create] una vez que cargamos los datos.



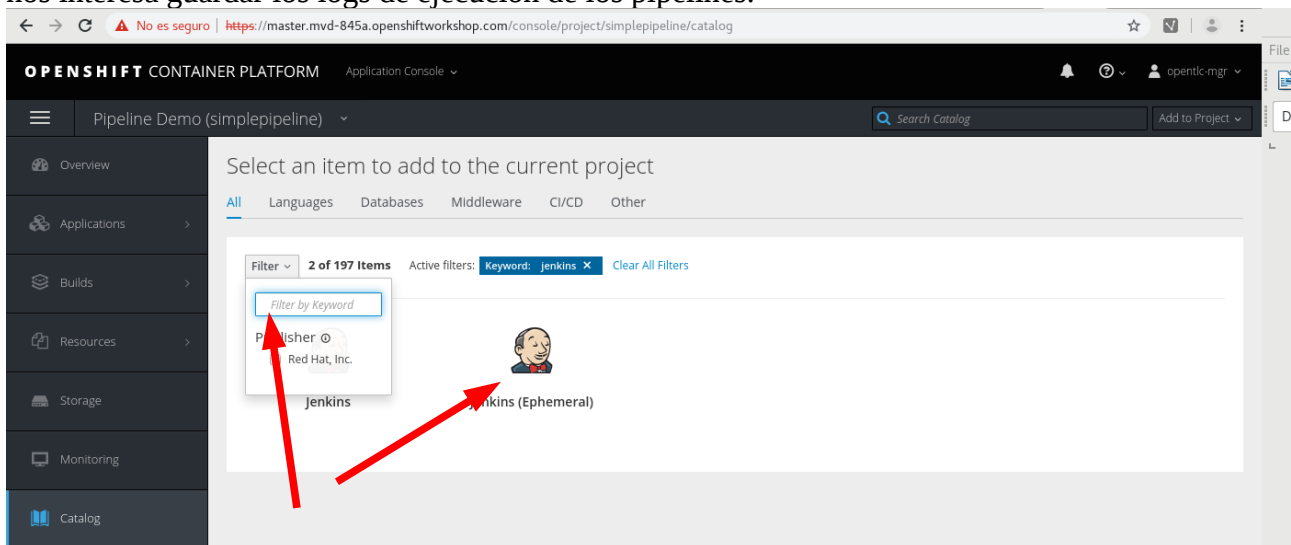
Seleccionamos el proyecto recién creado para trabajar con él.



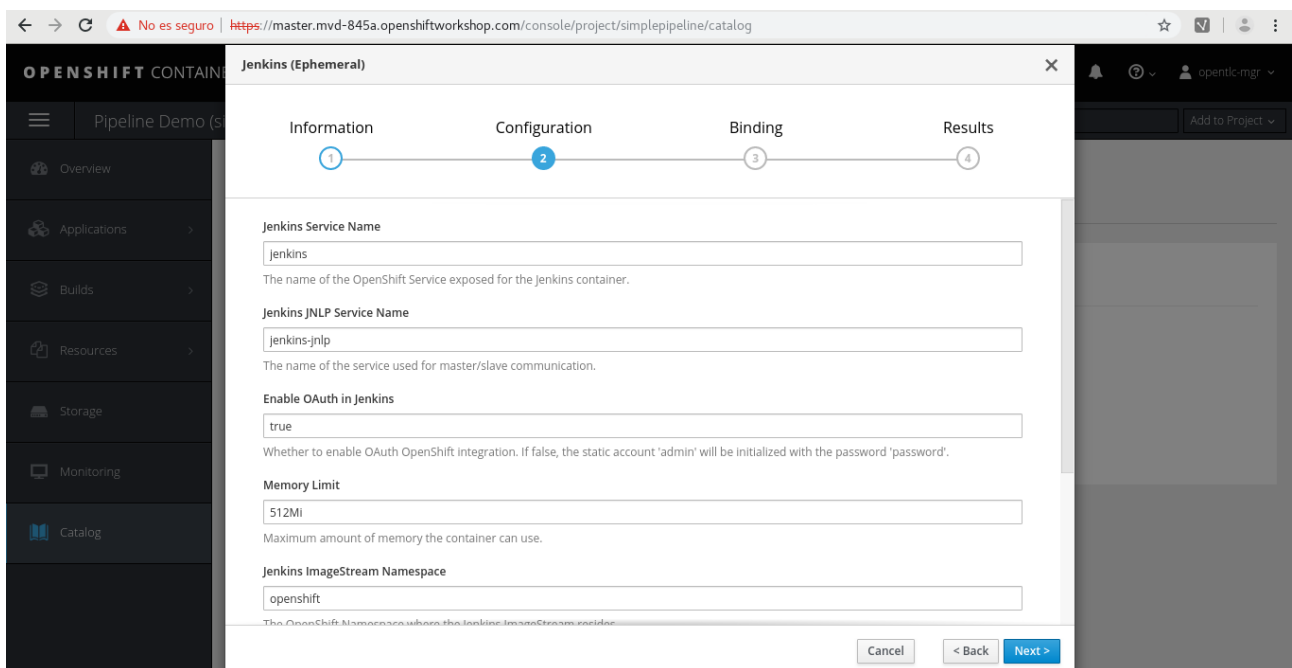
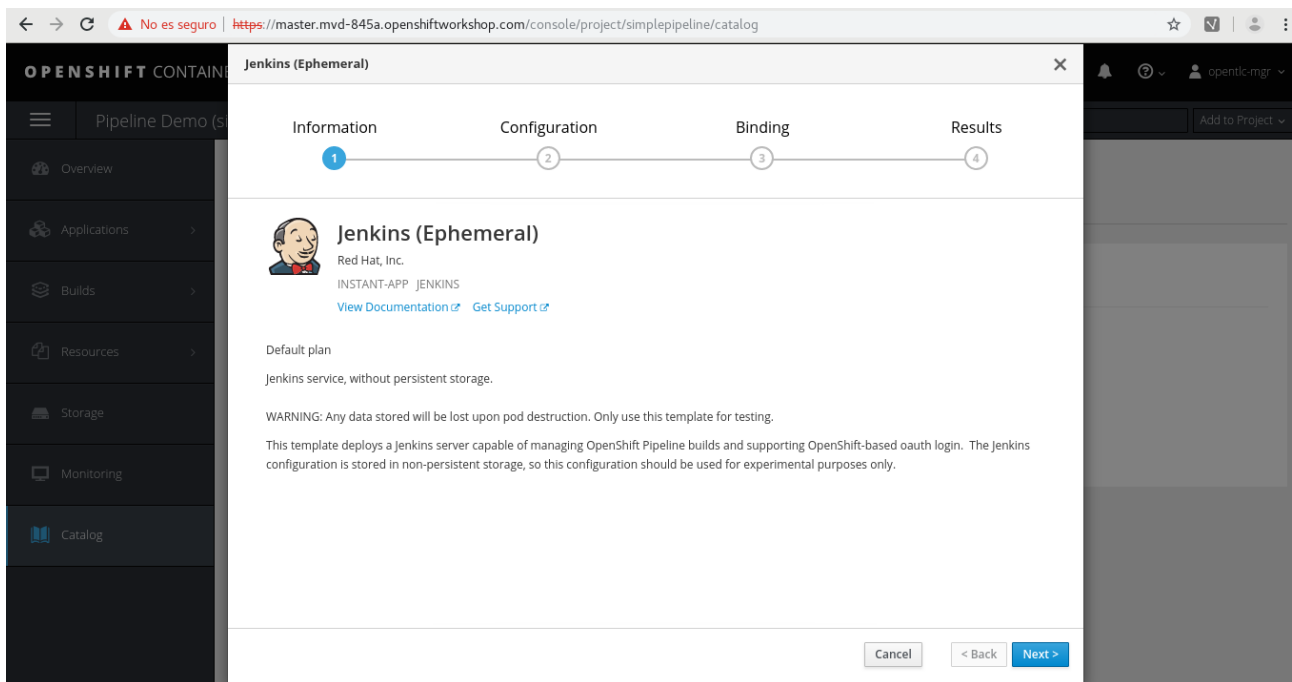
Al entrar en el proyecto tenemos la opción de comenzar a desplegar aplicaciones. Las podemos buscar en el catálogo, desplegar una imagen propia, importar un template o seleccionar una aplicación desde otro proyecto.

## Instalación de Jenkins

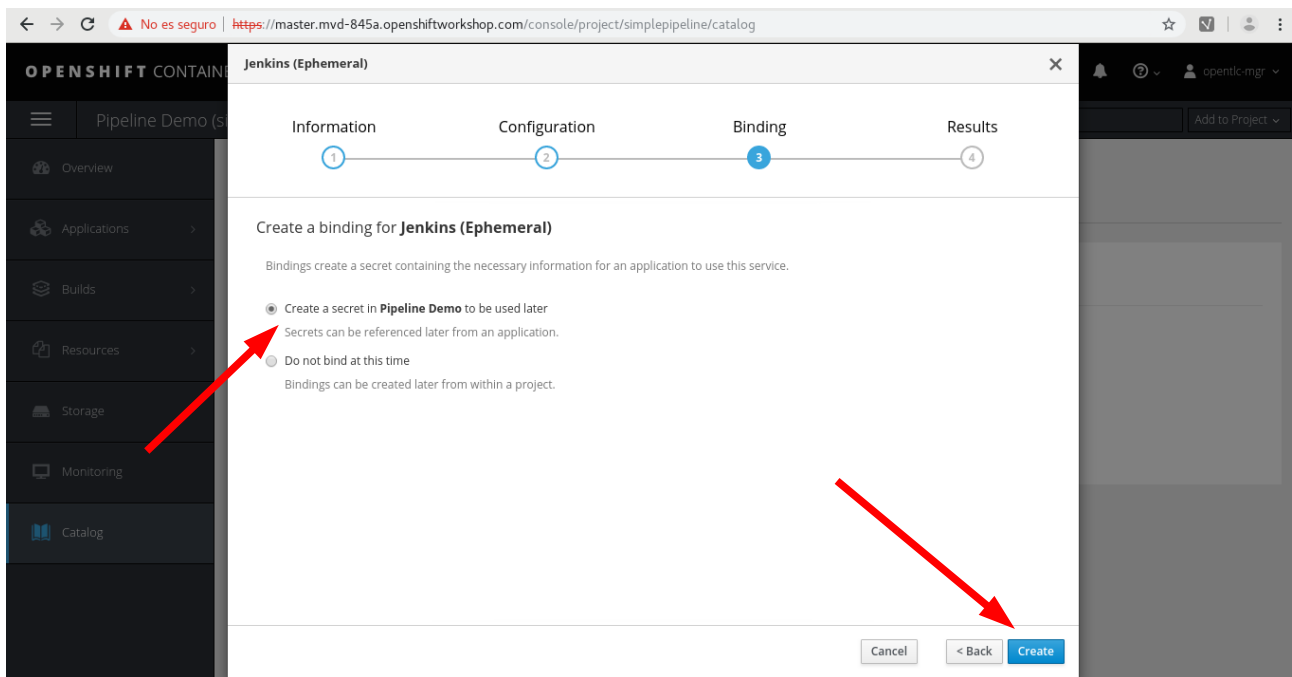
La primer aplicación que vamos a desplegar es Jenkins. Presionamos el botón [Browse Catalog] y vamos a buscar Jenkins. En este caso vamos a elegir la aplicación Jenkins (Ephemeral) ya que no nos interesa guardar los logs de ejecución de los pipelines.



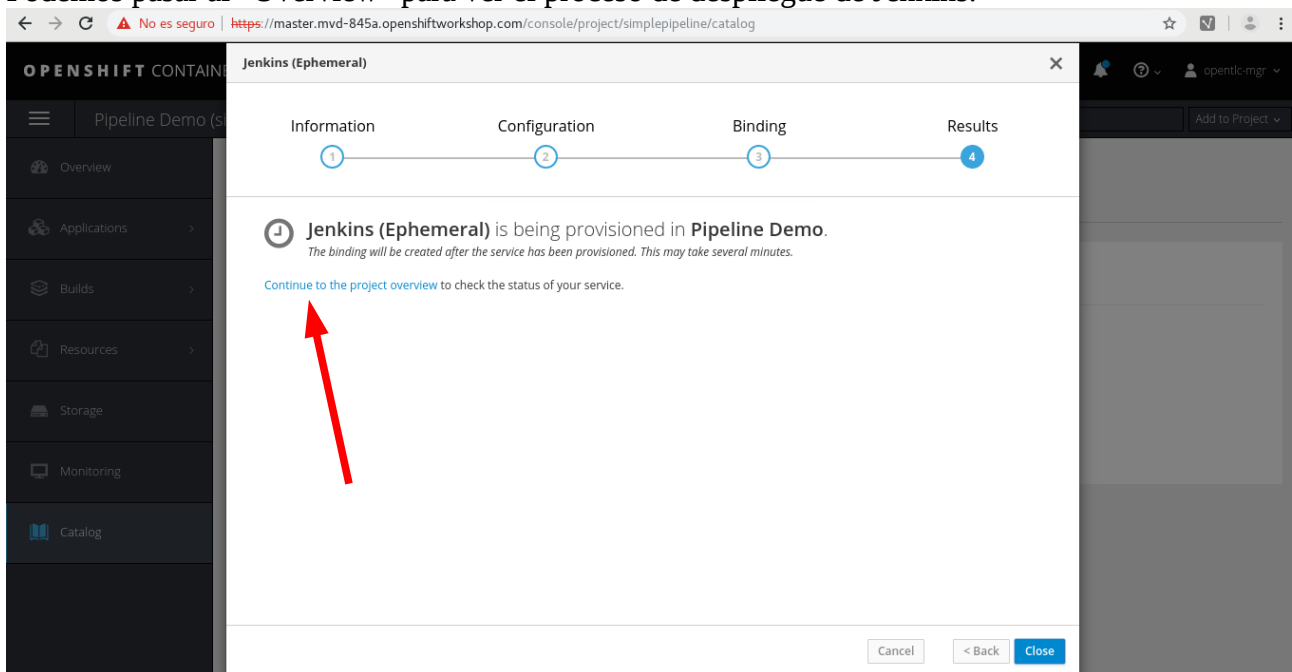
Vamos a llenar los datos de la aplicación manteniendo las opciones por omisión. Presionando el botón siguiente vamos avanzando en la configuración de la aplicación.



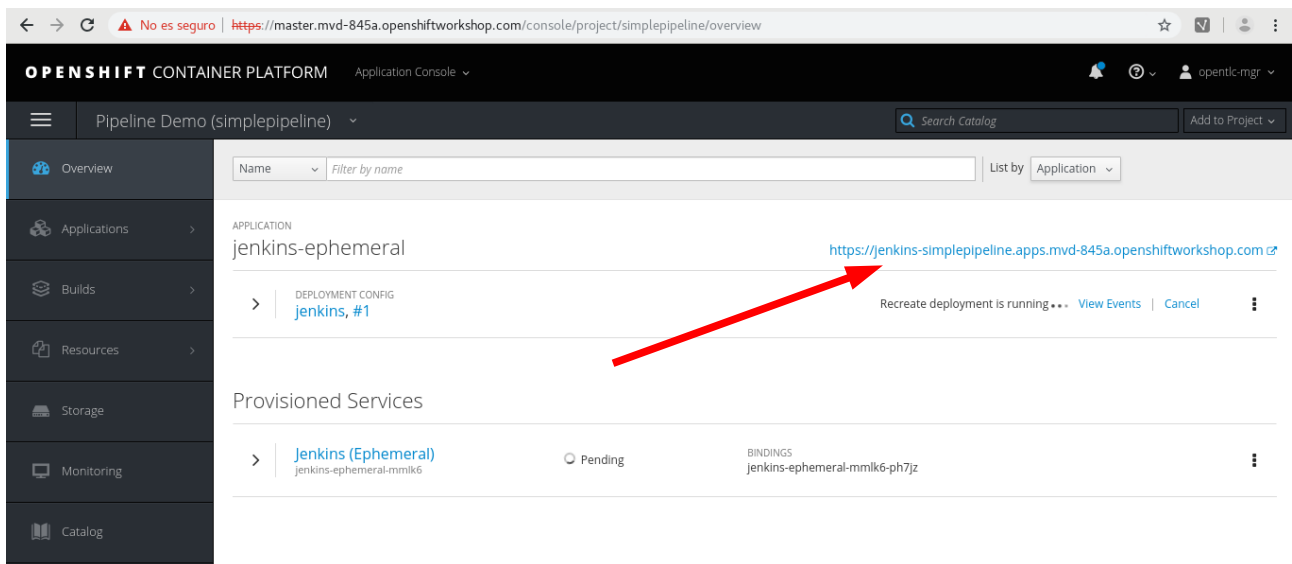
En la siguiente pantalla vamos a seleccionar la opción “Create a secret in Pipeline Demo to be used later”. Esta opción integra Jenkins con Openshift. Finalmente presionamos el botón [Create]



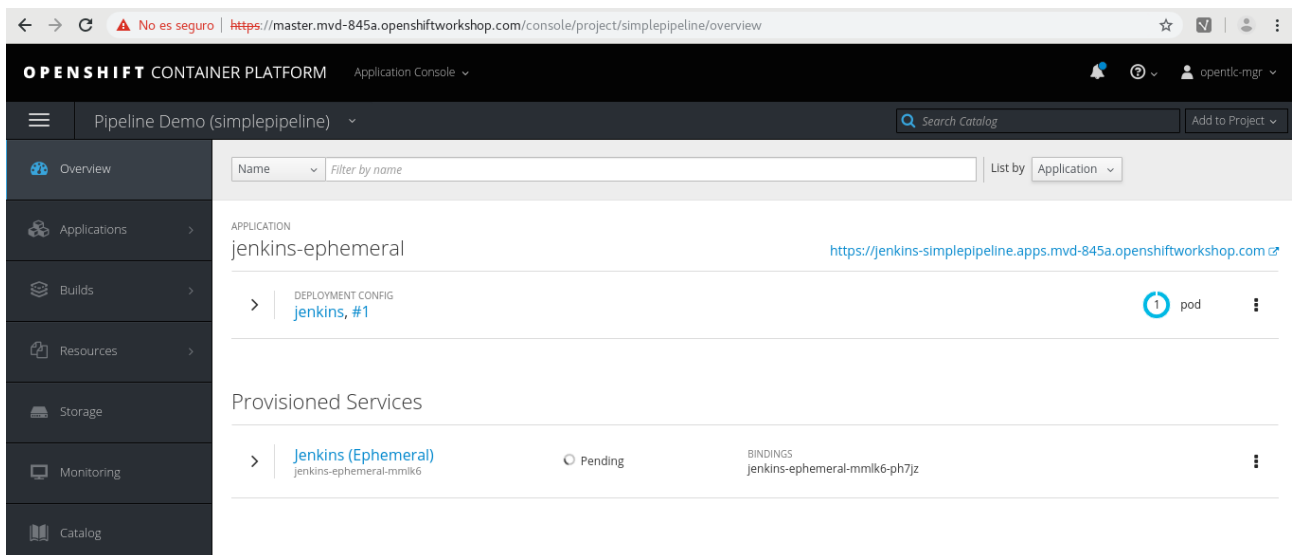
Podemos pasar al “Overview” para ver el proceso de despliegue de Jenkins.



En esta ventana podemos ver como va el despliegue. En la esquina superior derecha vemos el link para acceder a Jenkins, que vamos a utilizar más adelante.



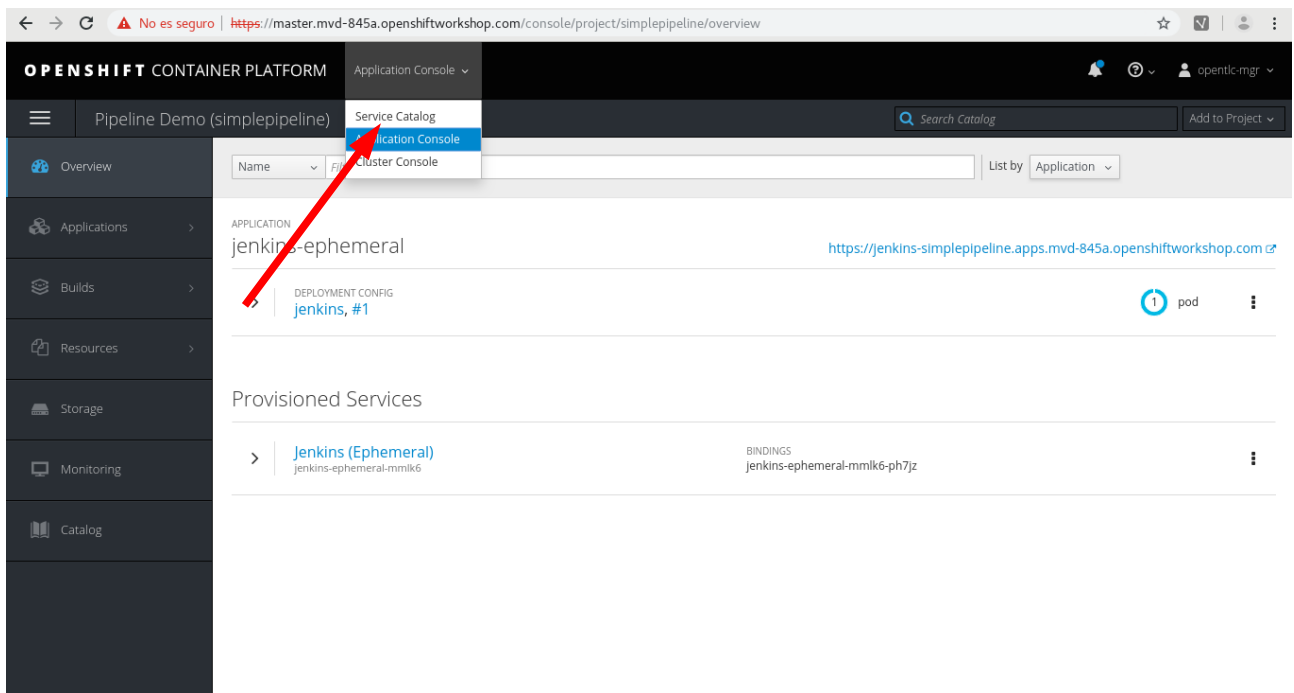
Una vez finalizado el deployment vamos a ver los pods que están desplegados.



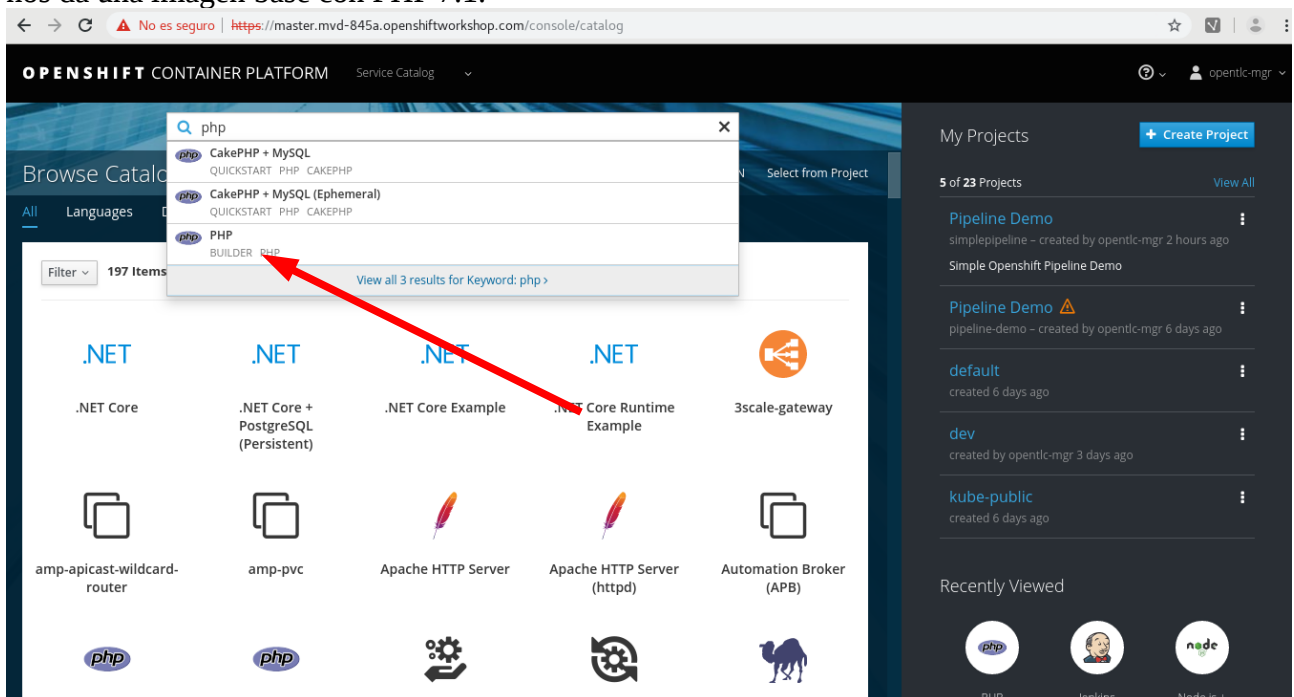
## Instalar aplicación de prueba

Ahora podemos desplegar nuestra aplicación. Vamos a desplegar una aplicación sencilla, desarrollada en PHP. Vamos a utilizar el mecanismo de S2I (Source to Image). Consiste en generar una imagen de contenedor a partir de una imagen base y del código de la aplicación, obtenido de GIT. Este mecanismo genera una imagen que se almacena en el registro interno de OpenShift. OpenShift automáticamente genera la configuración para el build de la imagen (buildconfig) y también para el deploy, incluyendo la cantidad de pods que deben estar activos, y la estrategia de actualización (por omisión se utiliza rolling release).

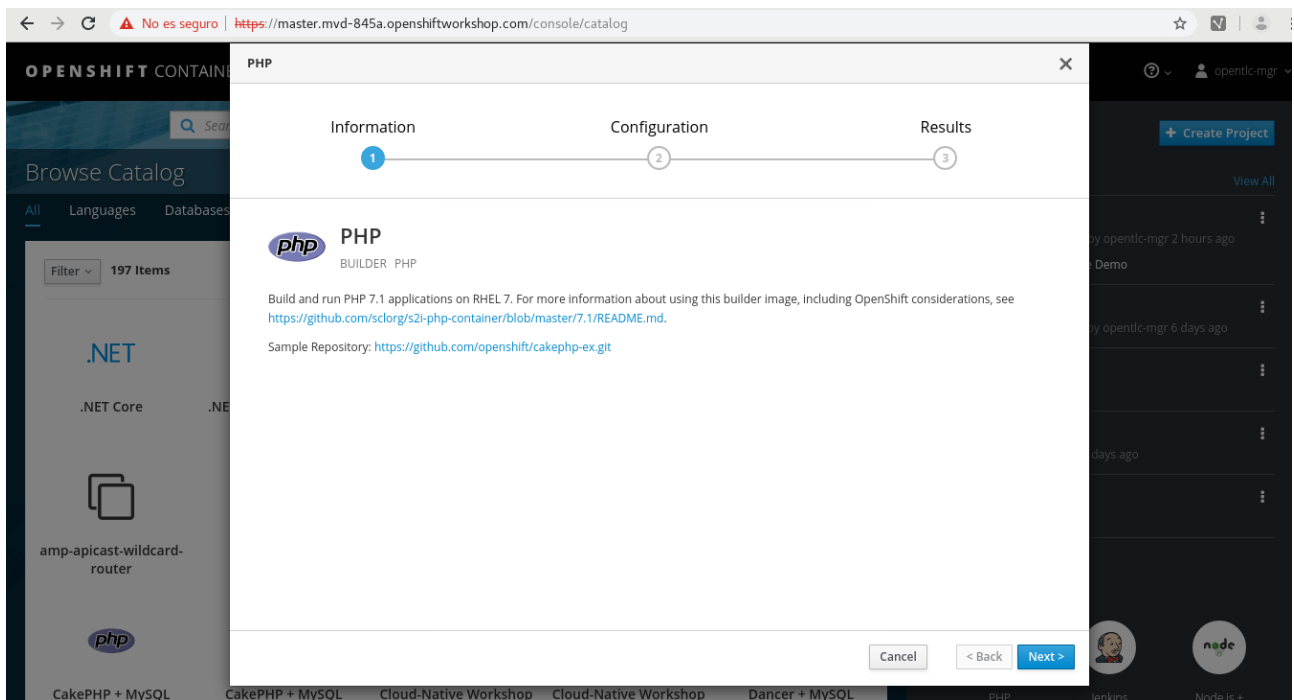
Desde el menú desplegable en la parte superior a la izquierda seleccionamos la opción “Service Catalog”



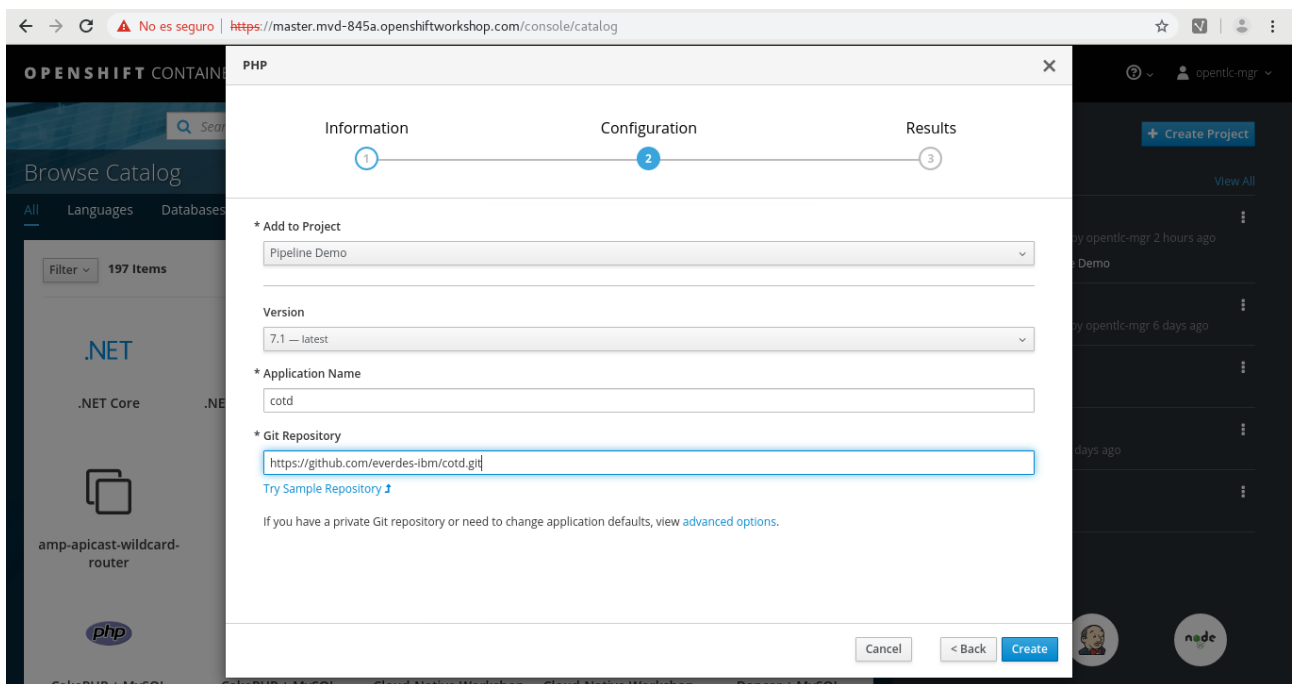
En el cuadro de búsqueda escribimos PHP, y seleccionamos la tercera opción, BUILDER PHP. Esto nos da una imagen base con PHP 7.1.



Comenzamos la configuración de la nueva aplicación presionando el botón [Next]



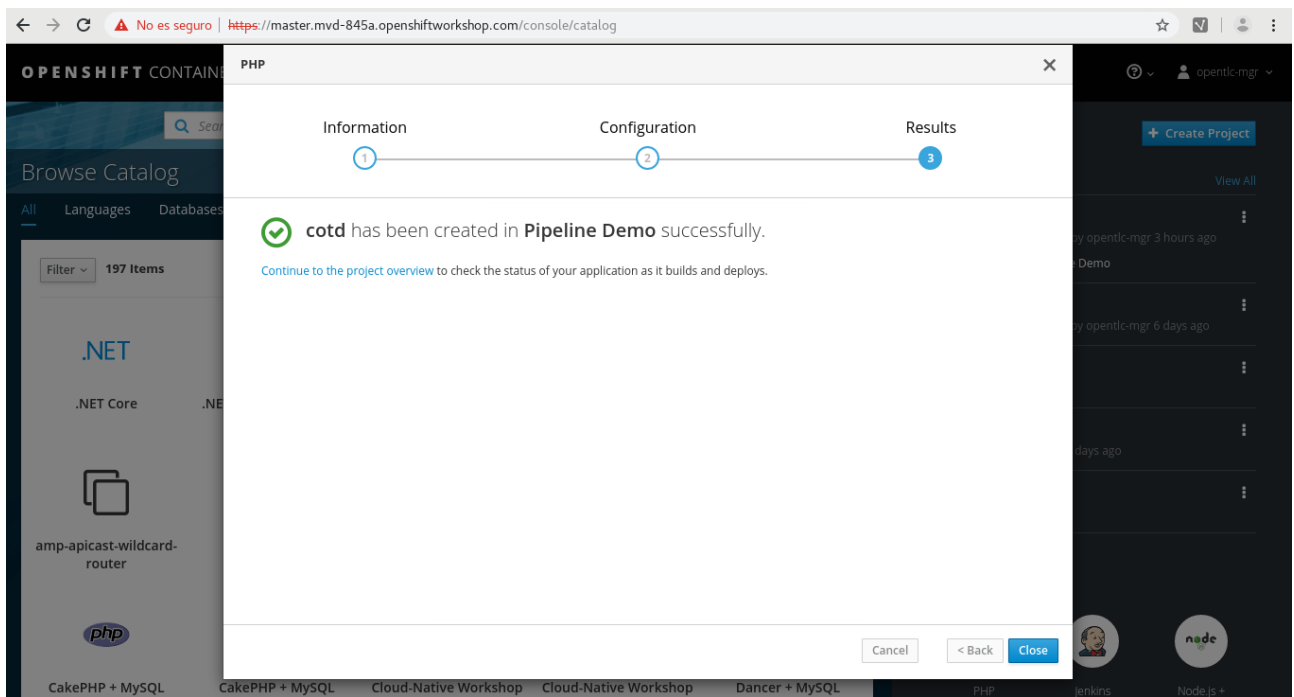
Completar los datos para generar la imagen:  
Add to Project: Pipeline Demo



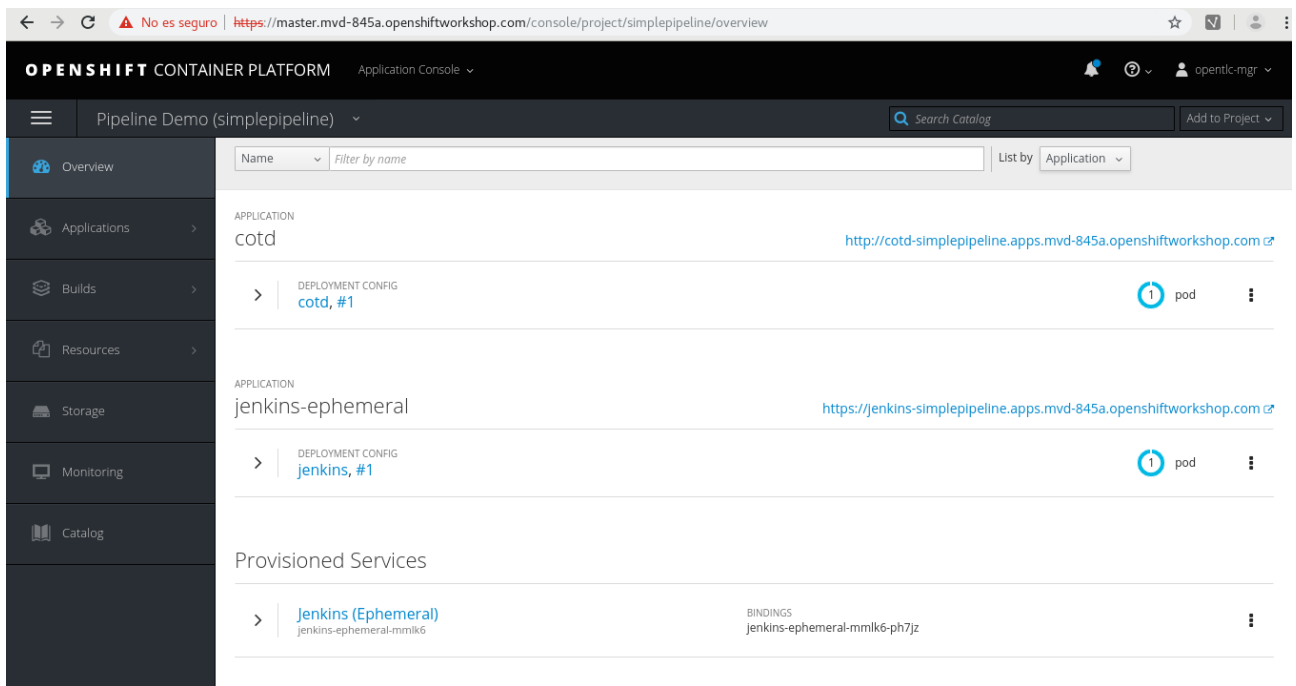
Versión: 7.1 – latest  
Application name: cotd  
Git Repository: <https://github.com/everdes-ibm/cotd.git> #Use su propio repositorio.

Presione el botón [Create] y luego dirijase a la opción de “Overview”.

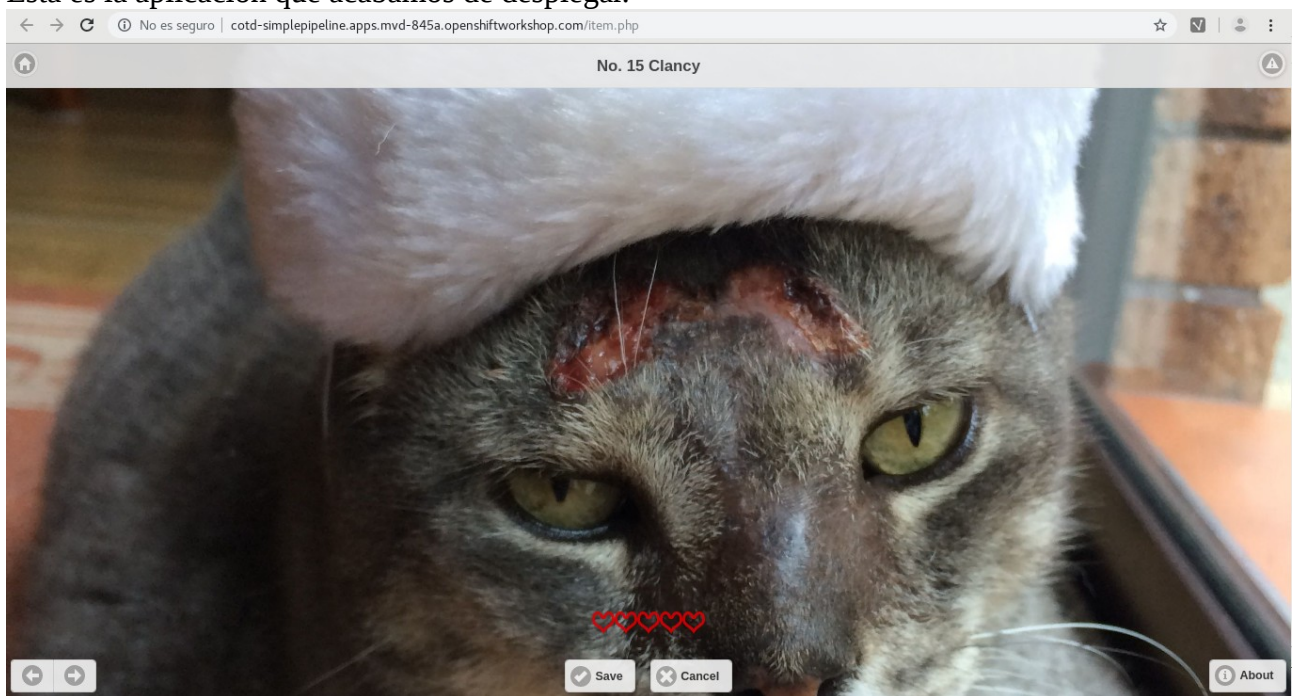




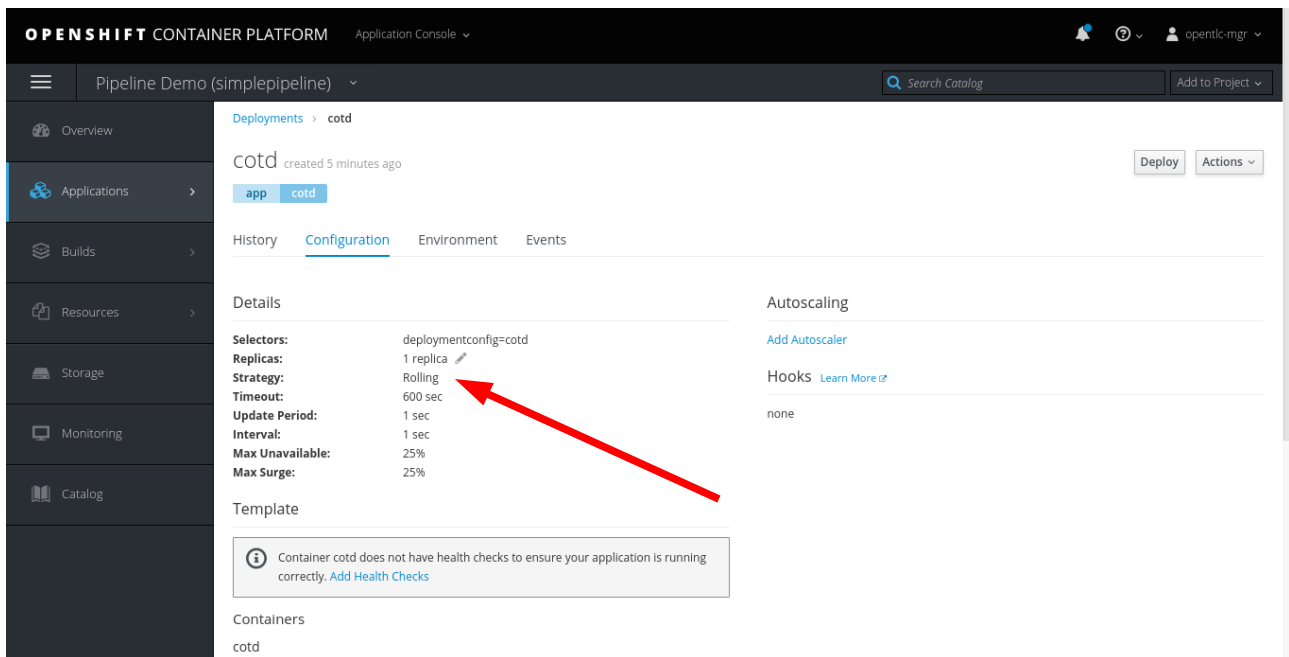
Puede ver desplegada la aplicación junto con Jenkins desplegado en el paso anterior. Si va al enlace en la esquina superior derecha, va a acceder a la aplicación.



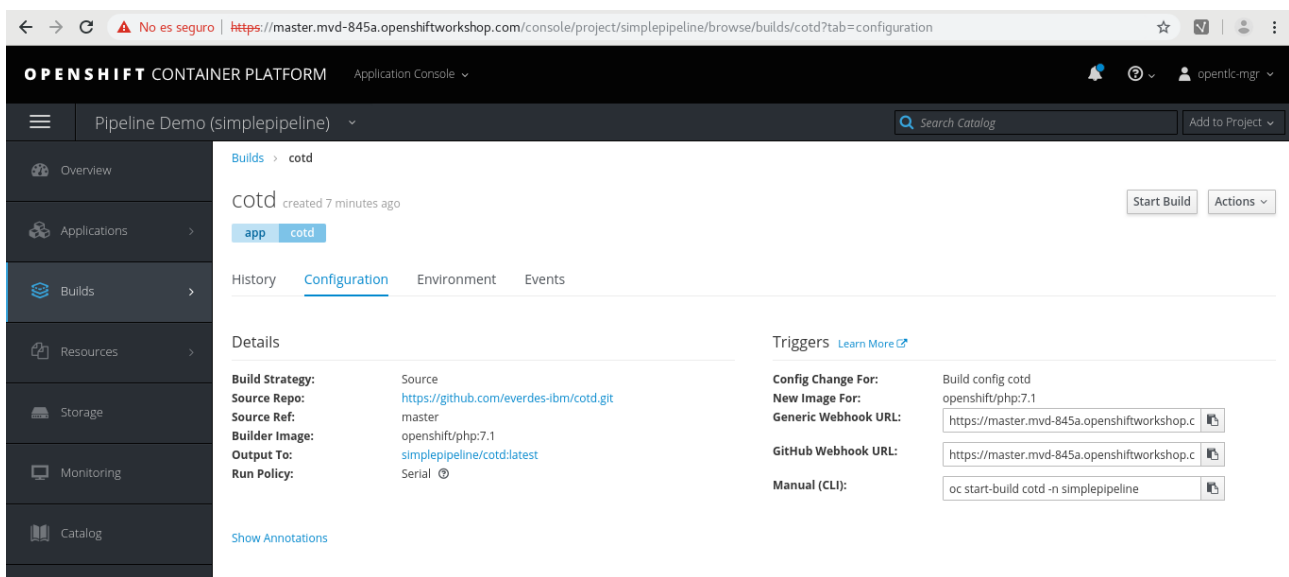
Esta es la aplicación que acabamos de desplegar.



Esta es la configuración de deployment que acabamos de realizar. Note la cantidad de pods que tienen que estar activos y la estrategia de deploy (rolling)



En la opción “Builds” del menú lateral, podemos ver el build y su configuración. Notar sobre la derecha los webhooks. Los vamos a usar como parte del pipeline.



## Configuración del Pipeline

El pipeline es un tipo de build, que lo que hace es ejecutar determinados pasos. El pipeline que vamos a usar está en el repositorio <https://github.com/everdes-ibm/cicdpipeline/pipelinebuild.yml> y contiene lo siguiente

```
kind: "BuildConfig"
apiVersion: "v1"
metadata:
  name: "sample-pipeline"
spec:
  strategy:
    jenkinsPipelineStrategy:
      jenkinsfile: |-
```

```

node('master') {
  stage 'build'
  openshiftBuild(buildConfig: 'cotd', showBuildLogs: 'true')
  stage 'deploy'
  openshiftDeploy(deploymentConfig: 'cotd')
}
triggers:
- github:
  secret: r3dh4t1!
  type: github

```

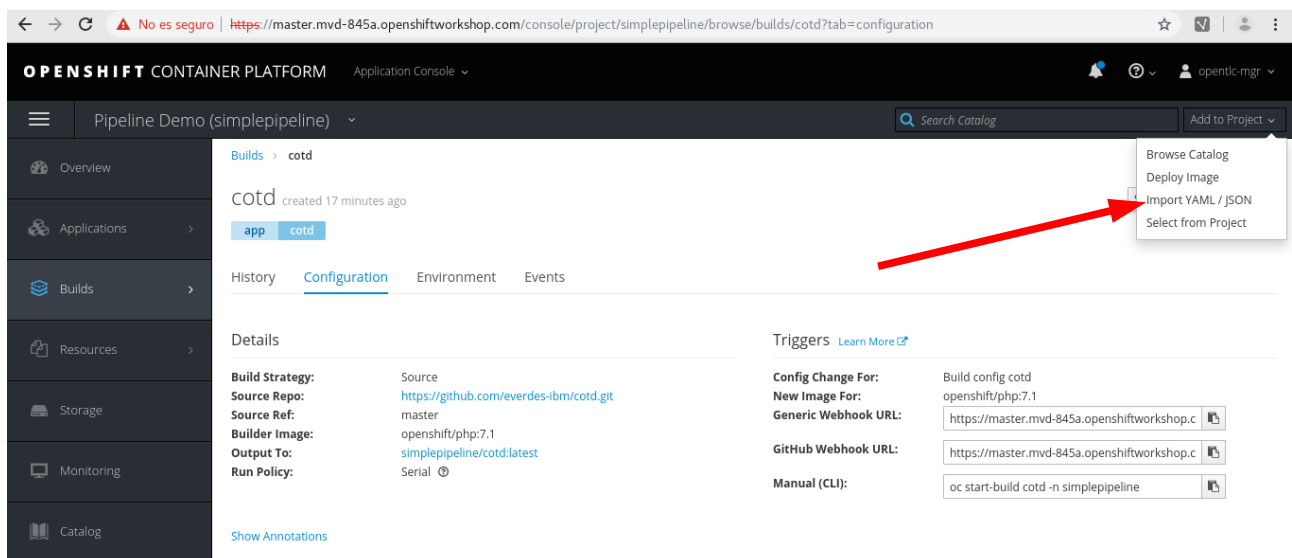
Para agregarlo a openshift podemos seleccionar en el menú desplegable sobre la esquina superior derecha “Add to project” la opción “Import YAML/JSON”

También se puede especificar un Jenkinsfile en un repositorio git.

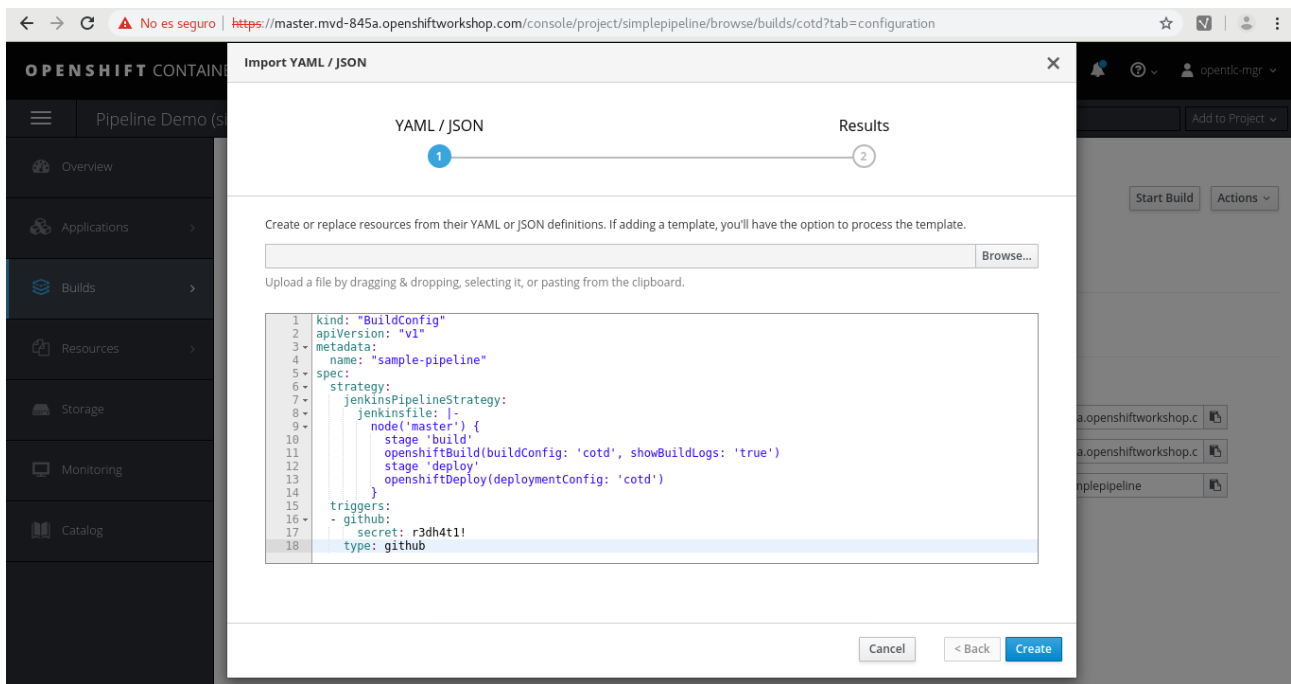
```

kind: "BuildConfig"
apiVersion: "v1"
metadata:
  name: "sample-pipeline"
spec:
  source:
    git:
      ref: master
      uri: https://github.com/everdes-ibm/cicdopenshift
  strategy:
    jenkinsPipelineStrategy:
      jenkinsfilePath: Jenkinsfile
      type: JenkinsPipeline
  triggers:
  - github:
    secret: r3dh4t1!
    type: github

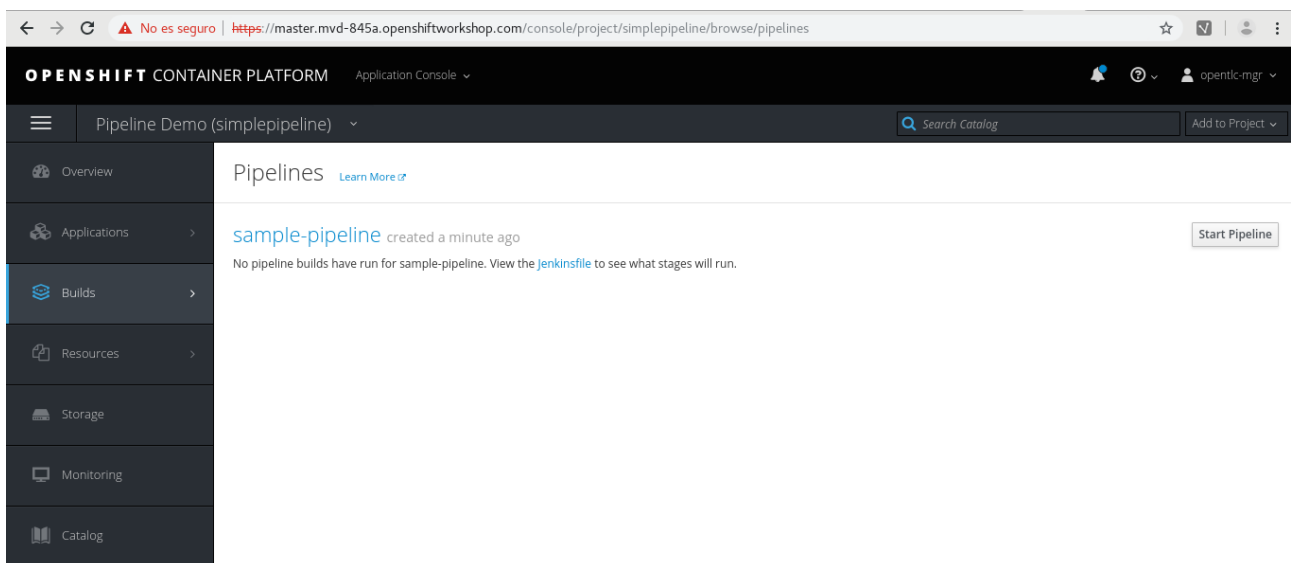
```



Podemos buscar el archivo en nuestra máquina local, ej. Nuestra copia local del repositorio, o pegarlo/ingresarlo en el cuadro de texto.



Al presionar el botón create vamos a agregar el pipeline. Ahora podemos ir a la opción “Builds” del menú lateral y seleccionar “Pipelines” para ver el pipeline que acabamos de crear.



Podemos disparar el pipeline manualmente, para probar que todos los pasos se ejecutan sin problemas. Vamos a ver cada etapa del pipeline como se va ejecutando y cuanto tiempo lleva.

The screenshot shows the OpenShift Container Platform console. The left sidebar contains navigation links: Overview, Applications, Builds, Resources, Storage, Monitoring, and Catalog. The main content area is titled 'sample-pipeline-1' and shows a progress bar for a build that is 'Running' and has a duration of 14s. Below this, the 'Status' section indicates the pipeline is 'Running' and started 'a few seconds ago'. The 'Configuration' section shows the 'Build Strategy' as 'Jenkins Pipeline' and the 'Source Type' as 'None'. A 'Jenkinsfile' is displayed with the following content:

```
node('master') {
  stage 'build'
  openshiftBuild(buildConfig: 'cotd', showBuildLogs: 'true')
}
```

En la configuración del pipeline podemos ver que también podemos configurar webhooks, para disparar el pipeline cuando hacemos un cambio de código en nuestro repositorio.

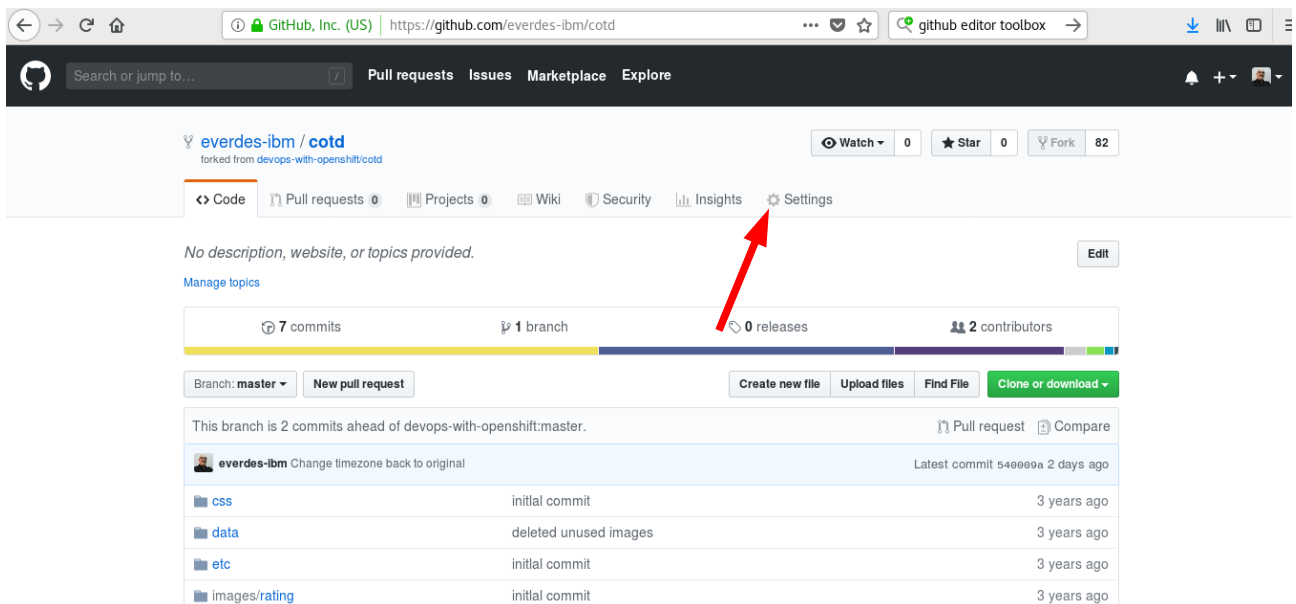
The screenshot shows the OpenShift Container Platform console with the 'sample-pipeline' configuration page. The left sidebar is the same as in the previous screenshot. The main content area shows the 'Configuration' tab for 'sample-pipeline'. The 'Details' section shows the 'Build Strategy' as 'Jenkins Pipeline' and the 'Run Policy' as 'Serial'. A 'Jenkinsfile' is displayed with the following content:

```
node('master') {
  stage 'build'
  openshiftBuild(buildConfig: 'cotd', showBuildLogs: 'true')
  stage 'deploy'
  openshiftDeploy(deploymentConfig: 'cotd')
}
```

On the right side, the 'Triggers' section is visible. It includes a 'GitHub Webhook URL' field with the value 'https://master.mvd-845a.openshiftworkshop.c' and a 'Manual (CLI)' field with the value 'oc start-build sample-pipeline -n simplepipelin'. A red arrow points to the 'Manual (CLI)' field.

Para configurar el webhook vamos a copiar la URL para Github.

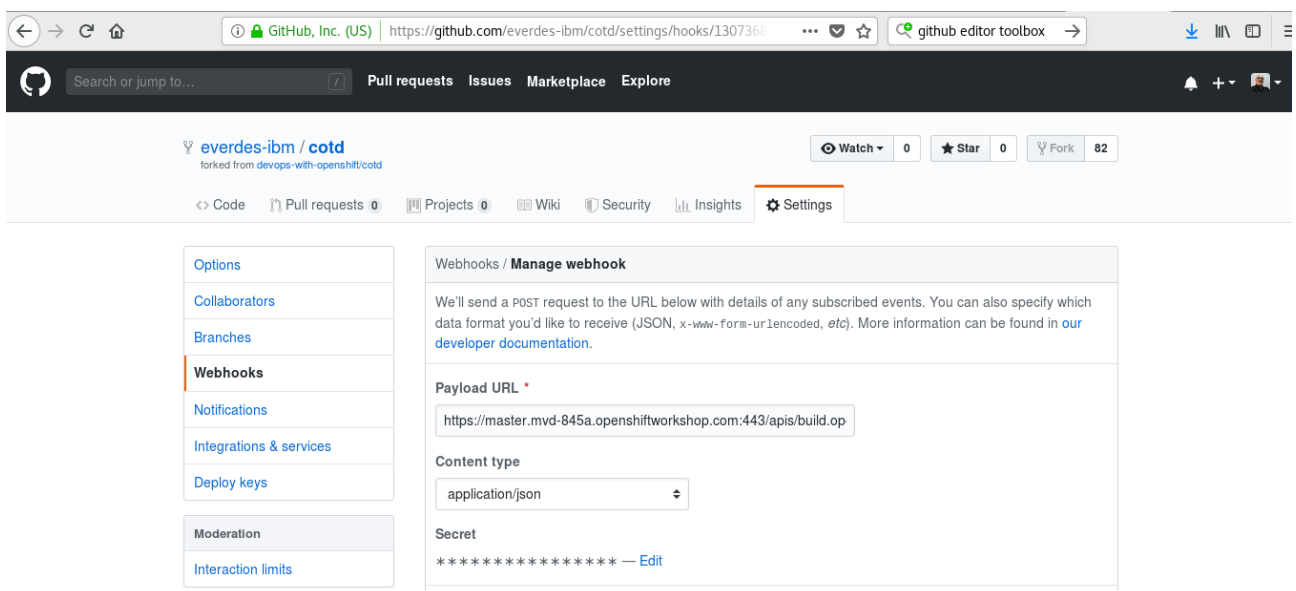
Ahora vamos a nuestro repositorio en github para configurar el webhook. Tenemos que seleccionar la opción “Settings” en el menú superior del repositorio.



Pegamos al url que copiamos desde Openshift. Tenemos que tener cuidado, porque la URL contiene el secret que configuramos en el pipeline, así que tenemos que sustituirlo por <secret>

El Content type es application/json .

El secret es el que configuramos en el pipeline.



Si continuamos más abajo en la página tenemos un par de acciones que configurar.

SSL verification vamos a ponerlo en Disable porque nuestro Openshift no tiene un certificado válido.

Luego podemos configurar que tipo de acción va a disparar el pipeline, por omisión es cuando hacemos push.

Interaction limits

\*\*\*\*\* — Edit

**SSL verification**  
 By default, we verify SSL certificates when delivering payloads.  
☐ Enable SSL verification ☒ Disable (not recommended)

**Which events would you like to trigger this webhook?**  
☒ Just the push event.  
☐ Send me everything.  
☐ Let me select individual events.

☒ **Active**  
 We will deliver event details when this hook is triggered.

[Update webhook](#) [Delete webhook](#)

**Recent Deliveries**

Más abajo podemos ver las pruebas de conectividad realizadas para entregar a Openshift el aviso.

[Update webhook](#) [Delete webhook](#)

**Recent Deliveries**

✓ a548c4ec-bbbe-11e9-90b0-f55b50095806 2019-08-10 19:31:50

✓ f302c990-bbbd-11e9-8600-07adfd8932ef 2019-08-10 19:26:51

Request Response **200** [Redeliver](#) ⌚ Completed in 0.14 seconds.

**Headers**

Request URL: https://master.mvd-845a.openshiftworkshop.com:443/apis/build.openshift.io/v1  
 Request method: POST  
 content-type: application/json  
 Expect:  
 User-Agent: GitHub-Hookshot/e8c77c3  
 X-GitHub-Delivery: f302c990-bbbd-11e9-8600-07adfd8932ef  
 X-GitHub-Event: ping  
 X-Hub-Signature: sha1=d87245d5d552880d037a1c781b668f1a27b75694

**Payload**

```
{
  "zen": "Approachable is better than simple.",
  "hook_id": 130736846,
  ...
}
```

Realizamos un push al repositorio

```
[everdes@oc6101831263 ~]$ git config --global user.name Enrique Verdes
[everdes@oc6101831263 ~]$ git config --global user.email enrique.verdes@ibm.com
[everdes@oc6101831263 ~]$ git clone https://github.com/everdes-ibm/cotd.git
Cloning into 'cotd'...
...
Resolving deltas: 100% (25/25), done.
[everdes@oc6101831263 ~]$ cd cotd/
[everdes@oc6101831263 cotd]$ vim item.php
[everdes@oc6101831263 cotd]$ git add item.php
[everdes@oc6101831263 cotd]$ git commit -m "Change default SELECTOR to cats"
[master 98d0fad] Change default SELECTOR to cats
1 file changed, 1 insertion(+), 1 deletion(-)
[everdes@oc6101831263 cotd]$ git push
warning: push.default is unset; its implicit value is changing in
...
To https://github.com/everdes-ibm/cotd.git
```



```
540009a..98d0fad master -> master
[everdes@oc6101831263 cotd]$
```

Podemos ver como el push disparó la ejecución del pipeline

The screenshot shows the OpenShift Container Platform console interface. The top navigation bar includes the OpenShift logo, the text 'CONTAINER PLATFORM', and the 'Application Console' dropdown. The left sidebar contains a menu with 'Overview', 'Applications', 'Builds', 'Resources', 'Storage', 'Monitoring', and 'Catalog'. The main content area is titled 'Pipelines' and shows the 'sample-pipeline' created 18 minutes ago. A 'Start Pipeline' button is visible. Below this, the 'Recent Runs' section displays two runs: 'Build #2' (a few seconds ago) and 'Build #1' (16 minutes ago). Each run shows a progress bar for 'build' and 'deploy' stages. 'Build #2' shows a successful build (16s) and a failed deploy (6s). 'Build #1' shows a successful build (17s) and a successful deploy (26s). The 'Average Duration' is 57s. At the bottom of the 'Recent Runs' section, there are links for 'View Pipeline Runs' and 'Edit Pipeline'.

Run	Build Status	Build Duration	Deploy Status	Deploy Duration
Build #2 a few seconds ago <a href="#">View Log</a>	Success	16s	Failed	6s
Build #1 16 minutes ago <a href="#">View Log</a>	Success	17s	Success	26s