

Infrastructure **GBM**

Software **GBM**

Services **GBM**

Consulting **GBM**



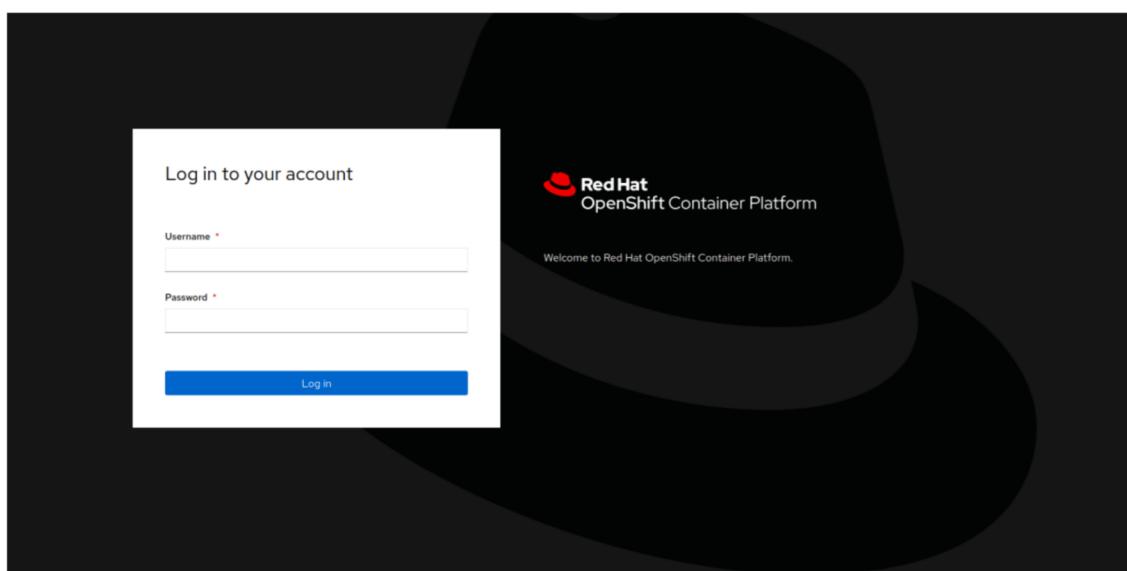
Laboratorio de Kubernetes

Laboratorio Kubernetes

Los siguientes ejercicios se van a ejecutar en la plataforma OpenShift, conectándose directamente a un ambiente que se les proveerá para el curso. Los comandos que se utilizarán en este laboratorio serán ejecutados a través de su terminal (cmd de windows), por lo que suponemos que llegado a este laboratorio, ya ha ejecutado la instalación de oc (openshift client) y kubernetes en su máquina siguiendo los pasos del documento “OC – Kubectl Windows.pdf”, el cual también se proveera en esta actividad.

Prerequisitos

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



Para ingresar se deben indicar los siguientes parámetros

Usuario: *userX* (Se le asignará un usuario durante el curso)

Password: *openshift*

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

Y copiamos el login command. Para ellos damos clic en “Copy Login Command”

The screenshot shows the OpenShift web interface. At the top right, there is a user dropdown menu with options: '+', '?', 'user', and a dropdown arrow. Below this, a red box highlights the 'Copy Login Command' option in a dropdown menu. Other options in the menu include 'Log out' and a search bar labeled 'Filter by name or display name...'. The main content area displays a 'Welcome to OpenShift' message, instructions for getting started, a link to documentation, download links for command-line tools, and a 'Create Project' button.

Indicamos nuestro usuario y contraseña y luego **Login**

The screenshot shows the Red Hat OpenShift Container Platform login page. On the left, a white box contains the 'Log in to your account' form. It has fields for 'Username *' (with 'user' entered) and 'Password *' (with '*****' shown as a yellow redaction). A blue 'Log in' button is at the bottom. On the right, the Red Hat logo is displayed above the text 'OpenShift Container Platform'. Below that, a welcome message reads 'Welcome to Red Hat OpenShift Container Platform.'

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

Display Token

Y copiamos el valor contenido en la leyenda **Log in with this Token** y ese contenido lo pegamos en la consola de cmd de Windows o en la terminal de linux.

Your API token is
B-JYkUTnSZLo_buWH1fZw6mSSPoMnkyd0IdTODOVlmo

Log in with this token

```
oc login --token=B-JYkUTnSZLo_buWH1fZw6mSSPoMnkyd0IdTODOVlmo --server=https://api.cluster-car-b782.car-b782.example.opentlc.com:6443
```

Use this token directly against the API

```
curl -H "Authorization: Bearer B-JYkUTnSZLo_buWH1fZw6mSSPoMnkyd0IdTODOVlmo" "https://api.cluster-car-b782.car-b782.example.opentlc.com:6443/apis/user.openshift.io/v1/users/~"
```

Request another token

Nos debería mostrar un mensaje como el siguiente:

```
$ oc login --token=B-JYkUTnSZLo_buWH1fZw6mSSPoMnkyd0IdTODOVlmo --server=https://api.cluster-car-b782.car-b782.example.opentlc.com:6443
```

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://api.cluster-car-b782.car-b782.example.opentlc.com:6443" as "user1"  
using the token provided.
```

You don't have any projects. You can try to create a new project, by running

```
oc new-project <projectname>
```

Listo, ya tenemos configurado lo necesario para poder ejecutar con un ambiente de Kubernetes, en este caso OpenShift

Laboratorio 1. Trabajando con Kubernetes

1. Debemos crear un proyecto para trabajar.

Nota: Sustituta todas las “**x**” por el número de usuario asignado durante el curso

```
$ oc new-project proyecto-userx
```

Now using project "proyecto-userx" on server "https://api.cluster-car-b782.car-b782.example.opentlc.com:6443".

You can add applications to this project with the 'new-app' command. For example, try:

```
oc new-app centos/ruby-25-centos7~https://github.com/sclorg/ruby-ex.git
```

to build a new example application in Ruby.

2. Se le van a proporcionar los siguientes archivos para trabajar con ellos
 - user-hello-app-service-node-port.yaml
 - user-hello-app-deployment.yaml
3. Ver el deployment por medio de archivo yaml con el nombre “user-hello-app-deployment.yaml”.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello
  labels:
    role: hello
spec:
  replicas: 3
  selector:
    matchLabels:
      role: hello
      tier: web
  template:
    metadata:
      labels:
        role: hello
        tier: web
```

```
spec:  
  containers:  
    - name: hello-app  
      image: gcr.io/google-samples/hello-app:1.0  
    ports:  
      - containerPort: 8080
```

4. Para ello vamos a la ubicación del archivo *user-hello-app-deployment.yaml* y vamos a crear un deployment en el clúster de kubernetes.

```
$ kubectl create -f user-hello-app-deployment.yaml
```

```
deployment.apps/hello created
```

5. Revisar el status de los pods

```
$ kubectl get pods  
  
NAME        READY  STATUS    RESTARTS   AGE  
hello-d86597c56-8pvqb  1/1   Running   0          9m  
hello-d86597c56-dcmvx  1/1   Running   0          9m  
hello-d86597c56-ttkzm  1/1   Running   0          9m
```

Deben estar 3 pods en status “Running”.

6. Describir pods.

Para describir cualquiera de los pods se ejecuta el siguiente comando: “kubectl describe pod nombredelpod”, debe dar un output como el siguiente:

```
$ kubectl describe pod hello-d86597c56-ttkzm
```

```
Name:      hello-d86597c56-ttkzm  
  
Namespace: mi-primer-app-user1  
  
Priority:  0
```

Node: node1.sanjose-4d68.internal/192.168.0.52

Start Time: Sun, 24 Nov 2019 21:08:53 -0600

Labels: pod-template-hash=842153712

role=hello

tier=web

Annotations: kubernetes.io/limit-ranger: LimitRanger plugin set: cpu, memory request for container hello-app; cpu, memory limit for container hello-app

openshift.io/scc: restricted

Status: Running

IP: 10.1.4.10

IPs: <none>

Controlled By: ReplicaSet/hello-d86597c56

Containers:

hello-app:

Container ID:

docker://4c2a2e7c613a137d90afd1cb0febb665ddf09c001ee56779d5ff79b8d88db458

Image: gcr.io/google-samples/hello-app:1.0

Image ID: docker-pullable://gcr.io/google-samples/hello-app@sha256:c62ead5b8c15c231f9e786250b07909daf6c266d0fcddd93fea882eb722c3be4

Port: 8080/TCP

Host Port: 0/TCP

State: Running

Started: Sun, 24 Nov 2019 21:08:55 -0600

Ready: True

Restart Count: 0

Limits:

cpu: 500m

memory: 1536Mi

Requests:

cpu: 50m

memory: 256Mi

Environment: <none>

Mounts:

/var/run/secrets/kubernetes.io/serviceaccount from default-token-b7v9f (ro)

Conditions:

Type	Status
------	--------

Initialized	True
-------------	------

Ready	True
-------	------

ContainersReady	True
-----------------	------

PodScheduled	True
--------------	------

Volumes:

default-token-b7v9f:

Type: Secret (a volume populated by a Secret)

SecretName: default-token-b7v9f

Optional: false

QoS Class: Burstable

Node-Selectors: node-role.kubernetes.io/compute=true

Tolerations: node.kubernetes.io/memory-pressure:NoSchedule

Events:

Type	Reason	Age	From	Message
------	--------	-----	------	---------

-----	-----	-----	-----	-----
-------	-------	-------	-------	-------

Normal Scheduled 12m default-scheduler	Successfully assigned mi-primer-app-user1/hello-d86597c56-ttkzm to node1.sanjose-4d68.internal
--	--

```
Normal Pulled 12m kubelet, node1.sanjose-4d68.internal Container image  
"gcr.io/google-samples/hello-app:1.0" already present on machine
```

```
Normal Created 12m kubelet, node1.sanjose-4d68.internal Created container
```

```
Normal Started 12m kubelet, node1.sanjose-4d68.internal Started container
```

7. Ver el service por medio de archivo yaml con el nombre “user-hello-app-service-node-port.yaml”.

```
apiVersion: v1
```

```
kind: Service
```

```
metadata:
```

```
  name: hello
```

```
spec:
```

```
  type: NodePort
```

```
  ports:
```

```
    - port: 8080
```

```
      targetPort: 8080
```

```
      nodePort: 30000
```

```
  selector:
```

```
    role: hello
```

La forma de ligar o unir el servicio con el deployment es por medio del selector “role: hello”.

8. Crear el servicio en el cluster de kubernetes.

```
$ kubectl apply -f user-hello-app-service-node-port.yaml
```

```
service/hello created
```

9. Escalar deployment agregando más réplicas, actualmente tenemos 3 réplicas.

Revisamos la cantidad de réplicas.

```
$ kubectl get pods
NAME      READY  STATUS  RESTARTS  AGE
hello-d86597c56-8pvqb  1/1   Running  0          46m
hello-d86597c56-dcmvx  1/1   Running  0          46m
hello-d86597c56-ttkzm  1/1   Running  0          46m
```

Vamos a proceder a escalar el deployment de 3 réplicas a 6 réplicas, este procedimiento se ejecuta de forma fácil con 1 solo comando.

10. Primero revisamos el nombre del deployment.

```
$ kubectl get deployment
NAME  DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
hello  3        3        3          3          50m
```

Ya tenemos el nombre del deployment el cuál es “hello”.

11. Ejecutamos comando para escalar el deployment de 3 a 6 réplicas.

```
$ kubectl scale deployment/hello --replicas=6
deployment.extensions/hello scaled
```

Automáticamente se realiza la escalación a 6 réplicas en este caso.

12. Verificamos la cantidad de pods disponibles y verificamos los cambios en el deployment.

```
$ kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
hello-d86597c56-8pvqb	1/1	Running	0	55m
hello-d86597c56-8pwcq	1/1	Running	0	5m
hello-d86597c56-dcmvx	1/1	Running	0	55m
hello-d86597c56-gmsvc	1/1	Running	0	5m
hello-d86597c56-rhzl7	1/1	Running	0	5m
hello-d86597c56-ttkzm	1/1	Running	0	55m

```
$ kubectl get deployment
```

NAME	DESIRED	CURRENT	UP-TO-DATE	AVAILABLE	AGE
hello	6	6	6	6	56m