Laboratorio de Kubernetes

1. Trabajando con Kubernetes

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.

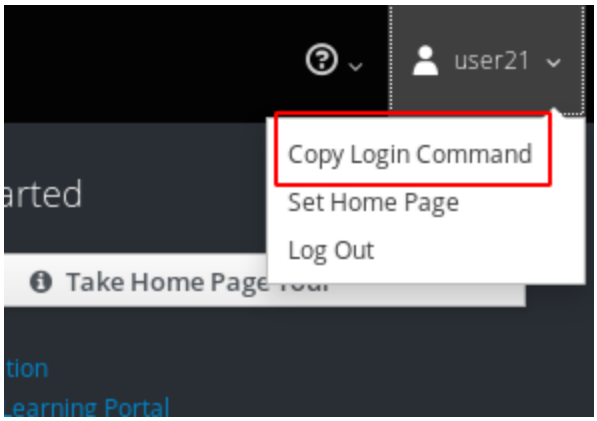
1. Nos logueamos a la URL https://master.sanjose-aa70.open.redhat.com/console/

Usuario: (pedir usuario)

Password: openshift

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

Y copiamos el login command y ese contenido lo pegamos en la consola de cmd de Windows.



Nos deberia mostrar un mensaje como el siguiente:

```
$ oc login https://master.sanjose-aa70.open.redhat.com:443 --token=nc7OkM-5G5GeBIM4nz8H8I_GuxqB3J5_Fhf7CzWecyY

Logged into "https://master.sanjose-aa70.open.redhat.com:443" as "user21" using the token provided.


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


    oc new-project <projectname>
```

Debemos crear un proyecto para trabajar.

```
$ oc new-project projecto-user21

Now using project "projecto-user21" on server "https://master.sanjose-
aa70.open.redhat.com:443".


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


    oc new-app django-psql-example


to build a new example application in Python. Or use kubectl to deploy a simple Kubernetes
application:


    kubectl create deployment hello-node --image=gcr.io/hello-minikube-zero-install/hello-
node
```

2. Crear deployment por medio de archivo yaml con el nombre “user1…n-hello-app-deployment.yaml” y el siguiente contenido.

```
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
```

3. Crear deployment en el clúster de kubernetes.

```
$ kubectl create -f user1-hello-app-deployment.yaml

deployment.apps/hello created
```

4. 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”.

5. 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
```

6. Crear servicio de tipo NodePort para el deployment “hello”

Crear un archivo “yaml” con el nombre “user1-hello-app-service-node-port.yaml “ y el siguiente contenido.

```
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”.

Crear el servicio en el cluster de kubernetes.

```
$ kubectl apply -f user1-hello-app-service-node-port.yaml
service/hello created
```

7. 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.

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”.

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.

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
```