

Lab 2

Nombres y Apellidos: Jhordan Manuel Escobar Soto

La gestión de sus aplicaciones stateless (sin estado) y stateful (con estado) con Kubernetes proporciona eficiencia y simplifica la automatización. Sin embargo, antes de utilizar StatefulSets para sus propias aplicaciones con estado, debería considerar si se da alguno de los siguientes casos:

- Adopta los microservicios
- Crea con frecuencia nuevas huellas de servicio que incluyen aplicaciones stateful (con estado).
- Su solución actual para el almacenamiento de estado no puede escalar para satisfacer la demanda prevista.
- Sus aplicaciones con estado pueden cumplir los requisitos de rendimiento sin utilizar hardware especializado y podrían ejecutarse eficazmente en el mismo hardware utilizado para las aplicaciones sin estado.
- Valora la reasignación flexible de recursos, la consolidación y la automatización por encima de exprimir al máximo y tener un rendimiento altamente predecible.

Si alguna de las viñetas anteriores se aplica a su situación, puede tener sentido utilizar Kubernetes para sus aplicaciones con estado.

Antecedentes:

ConfigMaps: Un tipo de recurso de Kubernetes que se utiliza para desacoplar los artefactos de configuración del contenido de la imagen para mantener la portabilidad de las aplicaciones en contenedores. Los datos de configuración se almacenan como pares clave-valor.

Headless Service: Un Headless Service es un recurso de servicio de Kubernetes que no equilibra la carga detrás de una única IP de servicio. En su lugar, un Headless Service devuelve una lista de registros DNS que apuntan directamente a los pods que respaldan el servicio. Un Headless Service se define declarando la propiedad clusterIP en una especificación de servicio y estableciendo el valor a None. Los StatefulSets requieren actualmente un Headless Service para identificar los pods en la red del cluster.

Stateful Sets: De forma similar a los despliegues en Kubernetes, los StatefulSets gestionan el despliegue y el escalado de los pods dada una especificación de contenedor. Los StatefulSets se diferencian de los Despliegues en que los Pods en un StatefulSet no son intercambiables. Cada pod en un StatefulSet tiene un identificador persistente que mantiene a través de cualquier reprogramación. Los pods de un StatefulSet también están ordenados. Esto proporciona una garantía de que un pod puede crearse antes que los siguientes. En este laboratorio, esto es útil para garantizar que el nodo del plano de control se aprovisiona primero.

PersistentVolumes (PVs) y PersistentVolumeClaims (PVCs): Los PVs son recursos de Kubernetes que representan el almacenamiento en el cluster. A diferencia de los Volúmenes regulares que existen sólo mientras existe el pod que los contiene, los PVs no tienen un tiempo de vida conectado a un pod. Por lo tanto, pueden ser utilizados por múltiples pods a lo largo del tiempo, o incluso al mismo tiempo. Los PV pueden utilizar diferentes tipos de almacenamiento, incluyendo NFS, iSCSI y volúmenes de almacenamiento proporcionados por la nube, como los volúmenes de AWS EBS. Los pods reclaman recursos de PV a través de PVCs.

MySQL replication: Este Laboratorio utiliza un único esquema de replicación primaria y asíncrona para MySQL. Todas las escrituras de la base de datos son gestionadas por un único primario. Las réplicas de la base de datos se sincronizan asincrónicamente con el primario. Esto significa que el primario no esperará a que los datos se copien en las réplicas. Esto

puede mejorar el rendimiento del primario a costa de tener réplicas que no siempre son copias exactas del primario. Muchas aplicaciones pueden tolerar ligeras diferencias en los datos y son capaces de mejorar el rendimiento de las cargas de trabajo de lectura de la base de datos permitiendo a los clientes leer desde las réplicas.

Listar Nodos:

```
yagami@YagamiDesk: ~/labs, X + v
yagami@YagamiDesk:~/labs/lab2$ kubectl get nodes
NAME                                     STATUS    ROLES    AGE   VERSION
ip-172-31-1-181.us-west-2.compute.internal Ready    <none>   28m   v1.21.5-eks-9017834
ip-172-31-24-163.us-west-2.compute.internal Ready    <none>   28m   v1.21.5-eks-9017834
yagami@YagamiDesk:~/labs/lab2$ |
```

Crear YML del ConfigMap

```
yagami@YagamiDesk: ~/labs, X + v
|
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql
  labels:
    app: mysql
data:
  master.cnf: |
    # Apply this config only on the primary
    [mysqld]
    log-bin
  slave.cnf: |
    # Apply this config only on replicas
    [mysqld]
    super-read-only
~
~
```

Crear el ConfigMap

```
yagami@YagamiDesk:~/labs/lab2$ kubectl create -f mysql-config-map.yaml
configmap/mysql created
yagami@YagamiDesk:~/labs/lab2$ |
```

Crear el YML del Service:

```
yagami@YagamiDesk: ~/labs, X + v
# Headless Service for stable DNS entries of StatefulSet members.
apiVersion: v1
kind: Service
metadata:
  name: mysql
  labels:
    app: mysql
spec:
  ports:
    - name: mysql
      port: 3306
  clusterIP: None
  selector:
    app: mysql

---
# Client service for connecting to any MySQL instance for reads.
# For writes, you must instead to the primary: mysql-0.mysql
apiVersion: v1
kind: Service
metadata:
  name: mysql-read
  labels:
    app: mysql
spec:
  ports:
    - name: mysql
"mysql-services.yaml" 30L, 501C
```

Crear el Service:

```
yagami@YagamiDesk:~/labs/lab2$ kubectl create -f mysql-services.yaml
service/mysql created
service/mysql-read created
```

Crear el YML del default Storage:

```
yagami@YagamiDesk: ~/li  ×  +  ∨  
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
  name: general  
  annotations:  
    storageclass.kubernetes.io/is-default-class: "true"  
provisioner: kubernetes.io/aws-ebs  
parameters:  
  type: gp2
```

Crear el default Storage:

```
yagami@YagamiDesk: ~/labs,  ×  +  ∨  
yagami@YagamiDesk:~/labs/lab2$ kubectl create -f mysql-storageclass3.yaml  
storageclass.storage.k8s.io/general created
```

Crear YML MySQL StatefulSet:

```
yagami@YagamiDesk: ~/labs, X + v
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql
spec:
  selector:
    matchLabels:
      app: mysql
  serviceName: mysql
  replicas: 3
  template:
    metadata:
      labels:
        app: mysql
    spec:
      initContainers:
        - name: init-mysql
          image: mysql:5.7
          command:
            - bash
            - "-c"
            - |
              set -ex
              # Generate mysql server-id from pod ordinal index.
              [[ `hostname` =~ -([0-9]+)$ ]] || exit 1
              ordinal=${BASH_REMATCH[1]}
              echo [mysqld] > /mnt/conf.d/server-id.cnf
"mysql-statefulset1.yaml" 155L, 4517C
```

Crear MySQL StatefulSet:

```
yagami@YagamiDesk:~/labs/lab2$ kubectl create -f mysql-statefulset4.yaml
statefulset.apps/mysql created
```

Solicitar describe de los Volumes group de los persistence group

kubectl describe pv

```
yagami@YagamiDesk:~/labs/lab2$ kubectl describe pv
Name:                pvc-11110584-bebf-4f05-8a2a-b789b969ba32
Labels:              topology.kubernetes.io/region=us-west-2
                    topology.kubernetes.io/zone=us-west-2b
Annotations:         kubernetes.io/createdby: aws-ebs-dynamic-provisioner
                    pv.kubernetes.io/bound-by-controller: yes
                    pv.kubernetes.io/provisioned-by: kubernetes.io/aws-ebs
                    [kubernetes.io/pv-protection]
Finalizers:          [kubernetes.io/pv-protection]
StorageClass:        general
Status:              Bound
Claim:               default/data-mysql-0
Reclaim Policy:      Delete
Access Modes:        RWX
VolumeMode:          Filesystem
Capacity:            2Gi
Node Affinity:
  Required Terms:
    Term 0:          topology.kubernetes.io/zone in [us-west-2b]
                    topology.kubernetes.io/region in [us-west-2]
Message:
Source:
  Type:              AWSElasticBlockStore (a Persistent Disk resource in AWS)
  VolumeID:          aws://us-west-2b/vol-0a44fdb2372bdee0c
  FSType:             ext4
  Partition:         0
  ReadOnly:           false
Events:              <none>
yagami@YagamiDesk:~/labs/lab2$
```

kubectl describe pvc

```
yagami@YagamiDesk:~/labs/lab2$ kubectl describe pvc
Name:                data-mysql-0
Namespace:           default
StorageClass:        general
Status:              Bound
Volume:              pvc-11110584-bebf-4f05-8a2a-b789b969ba32
Labels:              app=mysql
Annotations:         pv.kubernetes.io/bind-completed: yes
                    pv.kubernetes.io/bound-by-controller: yes
                    volume.beta.kubernetes.io/storage-provisioner: kubernetes.io/aws-ebs
Finalizers:          [kubernetes.io/pvc-protection]
Capacity:            2Gi
Access Modes:        RWX
VolumeMode:          Filesystem
Used By:             mysql-0
Events:
  Type      Reason              Age   From                      Message
  ----      -
  Normal    ProvisioningSucceeded 4m53s persistentvolume-controller Successfully provisioned volume pvc-11110584-bebf-4f05-8a2a-b789b969ba32 using kubernetes.io/aws-ebs
yagami@YagamiDesk:~/labs/lab2$ |
```

Solicitar get

```
kubectl get statefulset
```

```
yagami@YagamiDesk:~/labs/lab2$ kubectl get statefulset
NAME      READY   AGE
mysql     0/3     7m41s
yagami@YagamiDesk:~/labs/lab2$ |
```

Crear sesion temporal en contenedor

```
kubectl run mysql-client --image=mysql:5.7 -i -t --rm --restart=Never --\
/usr/bin/mysql -h mysql-0.mysql -e "CREATE DATABASE mydb; CREATE TABLE mydb.notes (note
VARCHAR(250)); INSERT INTO mydb.notes VALUES ('VALOR TEMPORAL);"
```

```
yasani@YasaniDesk:~/labs/lab2$ kubectl run mysql-client --image=mysql:5.7 -i -t --restart=Never -- /usr/bin/mysql -h mysql-0.mysql -e "CREATE DATABASE mydb; CREATE TABLE mydb.notes (note VARCHAR(250)); INSERT INTO mydb.notes VALUES ('VALOR TEMPORAL');"
```

Ejecutar el siguiente query

```
kubectl run mysql-client --image=mysql:5.7 -i -t --rm --restart=Never -- \
  /usr/bin/mysql -h mysql-read -e "SELECT * FROM mydb.notes"
```

```
yagami@YagamiDesk:~/labs/lab2$ kubectl run mysql-client --image=mysql:5.7 -i -t --rm --restart=Never -- /usr/bin/mysql -h mysql-read -e "SELECT * FROM mydb.notes"
If you don't see a command prompt, try pressing enter.
Error attaching, falling back to logs: unable to upgrade connection: container mysql-client not found in pod mysql-client_default
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
pod "mysql-client" deleted
pod default/mysql-client terminated (Error)
yagami@YagamiDesk:~/labs/lab2$
```

Ejecutar el siguiente comando para listar los pods:

```
kubectl get pod -o wide
```

```
yagami@YagamiDesk: ~/labs, X + v
yagami@YagamiDesk:~/labs/lab2$ kubectl get pods -o wide
NAME      READY   STATUS    RESTARTS   AGE   IP            NODE                                                    NOMINATED NODE   READINESS GATES
mysql-0    1/2     CrashLoopBackOff   6        6m54s   172.31.1.43   ip-172-31-1-181.us-west-2.compute.internal             <none>            <none>
yagami@YagamiDesk:~/labs/lab2$
```

Ejecute un comando SQL que genere la ID del servidor MySQL para confirmar que las solicitudes se distribuyen a diferentes pods

```
kubectl run mysql-client-loop --image=mysql:5.7 -i -t --rm --restart=Never -- bash -ic "while sleep 1; do /usr/bin/mysql -h mysql-read -e 'SELECT @@server_id'; done"
```

```

[sguest@sguest000:~/labs/lab2]$ sudo systemctl restart mysql-client-loop --image=mysql:5.7 -i -t --rm --restart=Never -- bash -ic 'while sleep 1; do /usr/bin/mysql -h mysql-read -e 'SELECT @@server_id'; do
if you don't see a command prompt, try pressing enter.
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
ERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)
AERROR 2003 (HY000): Can't connect to MySQL server on 'mysql-read' (111)

```

Ingrese el siguiente comando para simular que el nodo que ejecuta el pod mysql-2 fuera de servicio por mantenimiento

```
node=$(kubectl get pods --field-selector metadata.name=mysql-2 -
o=jsonpath='{.items[0].spec.nodeName}')
kubectl drain $node --force --delete-local-data --ignore-daemonsets
```

```
yagami@YagamiDesk:~/labs/lab2$ node=$(kubectl get pods --field-selector metadata.name=mysql-2 -o=jsonpath='{.items[0].spec.nodeName}')
error: error executing jsonpath "['.items[0].spec.nodeName]": Error executing template: array index out of bounds: index 0, length 0. Printing more information for debugging the template:
template was:
{.items[0].spec.nodeName}
object given to jsonpath engine was:
map[string]interface {}{"apiVersion":"v1", "items":[]interface {}{}, "kind":"List", "metadata":map[string]interface {}{"resourceVersion":"","selfLink":""}}
```

Observe cómo se reprograma el pod mysql-2 en un nodo diferente:

```
kubectl get pod -o wide -watch
```

```
yagami@YagamiDesk:~/labs/lab2$ kubectl get pod -o wide --watch
NAME          READY   STATUS             RESTARTS   AGE   IP            NODE                                          NOMINATED NODE   READINESS GATES
mysql-0       1/2     CrashLoopBackOff    7          12m   172.31.1.43   ip-172-31-1-181.us-west-2.compute.internal <none>            <none>
mysql-client-loop 1/1     Running             0          4m8s   172.31.16.223 ip-172-31-24-163.us-west-2.compute.internal <none>            <none>
```

Descordone el nodo que drenó para que los pods se puedan programar en él nuevamente:

```
kubectl uncordon $node
```

```
yagami@YagamiDesk:~/labs/lab2$ kubectl uncordon ip-172-31-24-163.us-west-2.compute.internal
node/ip-172-31-24-163.us-west-2.compute.internal already uncordoned
```

Elimine el pod mysql-2 para simular una falla de nodo y observe cómo se reprograma automáticamente:

```
kubectl delete pod mysql-2
kubectl get pod mysql-2 -o wide -watch
```

```
yagami@YagamiDesk:~/labs/lab2$ kubectl delete pod mysql-2
Error from server (NotFound): pods "mysql-2" not found
yagami@YagamiDesk:~/labs/lab2$
```

```
yagami@YagamiDesk:~/labs/lab2$ kubectl get pod -o wide --watch
NAME          READY   STATUS             RESTARTS   AGE   IP            NODE                                          NOMINATED NODE   READINESS GATES
mysql-0       1/2     CrashLoopBackOff    7          12m   172.31.1.43   ip-172-31-1-181.us-west-2.compute.internal <none>            <none>
mysql-client-loop 1/1     Running             0          4m8s   172.31.16.223 ip-172-31-24-163.us-west-2.compute.internal <none>            <none>
mysql-client-loop 0/1     Error               0          5m13s  172.31.16.223 ip-172-31-24-163.us-west-2.compute.internal <none>            <none>
mysql-0       1/2     Error               8          16m   172.31.1.43   ip-172-31-1-181.us-west-2.compute.internal <none>            <none>
mysql-0       1/2     CrashLoopBackOff    8          16m   172.31.1.43   ip-172-31-1-181.us-west-2.compute.internal <none>            <none>
```

Escale el número de réplicas hasta 5:

```
kubectl scale --replicas=5 statefulset mysql
```

```
yagami@YagamiDesk: ~/labs, × + v
yagami@YagamiDesk:~/labs/lab2$ kubectl scale --replicas=5 statefulset mysql
statefulset.apps/mysql scaled
yagami@YagamiDesk:~/labs/lab2$
```


Observe cómo se programan nuevos pods en el clúster:

```
kubectl get pods -l app=mysql -watch
```

```
statefulset.apps/mysql scaled
yagami@YagamiDesk:~/labs/lab2$ kubectl get pods -l app=mysql -watch
error: name cannot be provided when a selector is specified
yagami@YagamiDesk:~/labs/lab2$ |
```

Verifique que vea las nuevas ID de servidor MySQL:

```
kubectl run mysql-client-loop --image=mysql:5.7 -i -t --rm --restart=Never -- \
bash -ic "while sleep 1; do /usr/bin/mysql -h mysql-read -e 'SELECT @@server_id'; done"
```

```
yagami@YagamiDesk: ~/labs, X + v
yagami@YagamiDesk:~/labs/lab2$ kubectl run mysql-client-loop --image=mysql:5.7 -i -t --rm --restart=Never -- bash -ic "while sleep 1; do /usr/bin/mysql -h mysql-read -e 'SELECT @@server_id'; done"
Error from server (AlreadyExists): pods "mysql-client-loop" already exists
yagami@YagamiDesk:~/labs/lab2$ |
```

Confirme que los datos estén replicados en el nuevo pod mysql-4:

```
kubectl run mysql-client --image=mysql:5.7 -i -t --rm --restart=Never -- \
/usr/bin/mysql -h mysql-4.mysql -e "SELECT * FROM mydb.notes"
```

```
yagami@YagamiDesk: ~/labs, X + v
yagami@YagamiDesk:~/labs/lab2$ kubectl run mysql-client --image=mysql:5.7 -i -t --rm --restart=Never -- /usr/bin/mysql -h mysql-4.mysql -e "SELECT * FROM m
ydb.notes"
ERROR 2005 (HY000): Unknown MySQL server host 'mysql-4.mysql' (22)
pod "mysql-client" deleted
pod default/mysql-client terminated (Error)
yagami@YagamiDesk:~/labs/lab2$ |
```

Muestra la IP virtual interna del punto final de lectura de mysql:

```
kubectl get services mysql-read
```

```
yagami@YagamiDesk:~/labs/lab2$ kubectl get services mysql-read
NAME          TYPE        CLUSTER-IP    EXTERNAL-IP  PORT(S)    AGE
mysql-read    ClusterIP   10.100.232.38 <none>       3306/TCP   71m
yagami@YagamiDesk:~/labs/lab2$ |
```