**TP Kubernetes**

Se decide crear una web con Drupal y la base de datos montada en MySQL todo bajo Kubernetes en un laboratorio en VirtualBox local.

## **Entorno**

El siguiente tutorial está diseñado para realizarse en:

- Arquitectura: Virtual Box VM 64 bits

- OS: Ubuntu mini básico de 64 bits

- Red: Adaptador de red en modo bridge (ver configuración de vm en virutal box)

Suponiendo que la vm ya está creada, procedemos a instalar el servicio de microk8s ya que con el mismo podemos ejecutar Kubernetes de forma local.

**Snappy**

Snappy es un sistema de gestión de paquetes e implementación de software diseñado y creado originalmente por Canonical para el sistema operativo de teléfonos Ubuntu. Los paquetes, llamados 'snaps' y la herramienta para usarlos 'snapd', funcionan en un rango de distribuciones de Linux y, por lo tanto, permiten el despliegue de software en sentido ascendente. El sistema está diseñado para funcionar para el internet de las cosas, la nube y la computación de escritorio.

|  |
| --- |
| diego@cloud:~$ sudo apt update && sudo apt install snapd |

### 

### **Extras**

Se recomienda instalar los siguientes paquetes.

|  |
| --- |
| diego@cloud:~$ sudo apt install git vim net-tools openssh-server curl gnupg2 |

## **Instalar microk8s**

MicroK8s instalará un Kubernetes mínimo y liviano que puede ejecutar y usar en prácticamente cualquier máquina.

|  |
| --- |
| diego@cloud:~$ sudo snap install microk8s --classic |

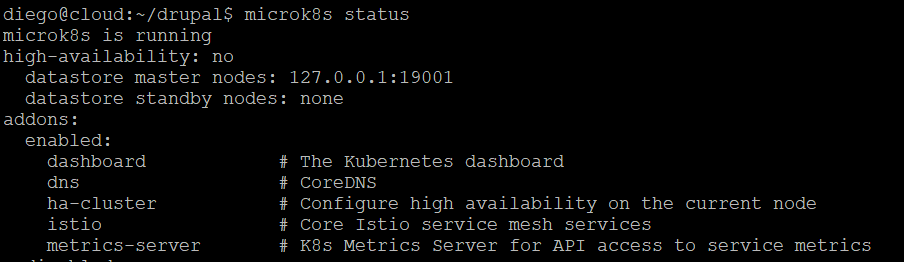
**Grupos**

MicroK8s crea un grupo para permitir el uso continuo de comandos que requieren privilegios de administrador. Para agregar su usuario actual al grupo y obtener acceso al directorio de almacenamiento en caché .kube, ejecute los siguientes dos comandos:

|  |
| --- |
| diego@cloud:~$ sudo usermod -a -G microk8s $USER diego@cloud:~$ sudo chown -f -R $USER ~/.kube |

**Status**

MicroK8s tiene un comando incorporado para mostrar su estado.



**Se procede con los pasos para instalar Drupal y MySQL.**

### **Directorio de trabajo**

Vamos a crear un directorio donde vamos a alojar todos los archivos necesarios para el trabajo.

|  |
| --- |
| diego@cloud:~$ mkdir drupal |

Luego ingresamos al directorio

|  |
| --- |
| diego@cloud:~$ cd drupal |

### 

### **Secrets**

Como primer paso debemos crear un objeto que gestione la contraseña para el usuario de base de datos.

|  |
| --- |
| diego@cloud:~/drupal$ vim kustomization.yaml  #Se crea la password cifrada para los accesos a la base y al usuario root en las apps.  secretGenerator:  - name: mysql-pass  literals:  - password=123456 |

Creamos el objeto:

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl apply -k . |

Lo listamos:

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl get secrets  NAME TYPE DATA AGE  defauokenlt-t-mn282 kubernetes.io/service-account-token 3 3h3m  istio.default istio.io/key-and-cert 3 174m  mysql-pass-2f466cdddc Opaque 1 146m |

**Volúmenes**

Creamos el archivo .yaml para los volúmenes locales de cada app (Drupal y MySQL)

|  |
| --- |
| diego@cloud:~/drupal$ vim drupal-pv.yaml  #Se crea el volumen local para la app de Drupal con un tamaño de 5G.  #Se podría también generar volúmenes dinámicos, pero me falto probarlo.  apiVersion: v1  kind: PersistentVolume  metadata:  name: drupal-pv  labels:  type: local  spec:  storageClassName: manual  capacity:  storage: 5Gi  accessModes:  - ReadWriteOnce  hostPath:  path: "/opt/storage/drupal" |

|  |
| --- |
| diego@cloud:~/drupal$ vim mysql-pv.yaml  #Se crea el volumen local para la app de MySQL con un tamaño de 5G.  #Se podría también generar volúmenes dinámicos, pero me falto probarlo.  apiVersion: v1  kind: PersistentVolume  metadata:  name: mysql-pv  labels:  type: local  spec:  storageClassName: manual  capacity:  storage: 5Gi  accessModes:  - ReadWriteOnce  hostPath:  path: "/opt/storage/mysql" |

Creamos el archivo .yaml para el Claim de los volúmenes locales de cada app (Drupal y MySQL)

|  |
| --- |
| diego@cloud:~/drupal$ vim drupal-pvc.yaml  #Se crea el volumen claim asociado al volumen local de la app Drupal.  apiVersion: v1  kind: PersistentVolumeClaim  metadata:  name: drupal-pv  spec:  storageClassName: manual  accessModes:  - ReadWriteOnce  resources:  requests:  storage: 5Gi |

|  |
| --- |
| diego@cloud:~/drupal$ vim mysql-pvc.yaml  #Se crea el volumen claim asociado al volumen local de la app MySQL.  apiVersion: v1  kind: PersistentVolumeClaim  metadata:  name: mysql-pv  spec:  storageClassName: manual  accessModes:  - ReadWriteOnce  resources:  requests:  storage: 5Gi |

**Aplicaciones**

Creamos el archivo .yaml para el Deployment de cada app (MySQL y Drupal)

|  |
| --- |
| diego@cloud:~/drupal$ vim mysql.yaml  ---  #Se crea el servicio para MySQL con el puerto correspondiente.  apiVersion: v1  kind: Service  metadata:  name: mysql-service  labels:  app: drupal  spec:  ports:  - port: 3306  selector:  app: drupal  tier: backend  ---  apiVersion: apps/v1  kind: Deployment  metadata:  name: drupal-mysql  labels:  app: drupal  spec:  replicas: 1 #Se crea 1 réplica sola ya que con más de una no logre que funcionara.  selector:  matchLabels:  app: drupal  tier: backend  strategy:  type: Recreate  template:  metadata:  labels:  app: drupal  tier: backend  spec:  containers:  - image: mysql:5.6  imagePullPolicy: IfNotPresent  name: mysql  env:  - name: MYSQL\_DATABASE  value: drupal-db  - name: MYSQL\_ROOT\_PASSWORD  valueFrom:  secretKeyRef: #es la secret reference al archivo llamado Kuztomization.yaml  name: mysql-pass-2f466cdddc  key: password  ports:  - containerPort: 3306  name: mysql  volumeMounts:  - name: mysql-stateful-storage  mountPath: /var/lib/mysql  volumes:  - name: mysql-stateful-storage  persistentVolumeClaim:  claimName: mysql-pv |

|  |
| --- |
| diego@cloud:~/drupal$ vim drupal.yaml  ---  # Se crea el servicio de LoadBalancer para la app Drupal con el puerto 80.  apiVersion: v1  kind: Service  metadata:  name: drupal-service  labels:  app: drupal  spec:  ports:  - port: 80  name: http  protocol: TCP  selector:  app: drupal  tier: frontend  type: LoadBalancer  ---  #Se crea el deployment de Drupal, con 3 replicas y la estrategia de RollingUpdate para poder hacer update o RollBack.  apiVersion: apps/v1  kind: Deployment  metadata:  name: drupal  labels:  app: drupal  tier: frontend  spec:  selector:  matchLabels:  app: drupal  tier: frontend  strategy:  type: RollingUpdate  replicas: 3  template:  metadata:  labels:  app: drupal  tier: frontend  spec:  initContainers:  - name: init-sites-volume  image: drupal:8.6  command: ['/bin/bash', '-c']  args: ['cp -r /var/www/html/sites /data; chown www-data:www-data /data/ -R']  volumeMounts:  - mountPath: /data  name: drupal-pvc  containers:  - image: drupal:8.6 #Se selecciono esta version para poder probar el update a la ultima version.  imagePullPolicy: IfNotPresent  name: drupal  env:  - name: DRUPAL\_DATABASE\_HOST  value: drupal-mysql  - name: DRUPAL\_DATABASE\_PASSWORD  valueFrom:  secretKeyRef: #es la secret reference al archivo llamado Kuztomization.yaml  name: mysql-pass-2f466cdddc  key: password  ports:  - containerPort: 80  name: drupal  volumeMounts:  - name: drupal-pvc  mountPath: /var/www/html/modules  subPath: modules  - name: drupal-pvc  mountPath: /var/www/html/profiles  subPath: profiles  - name: drupal-pvc  mountPath: /var/www/html/sites  subPath: sites  - name: drupal-pvc  mountPath: /var/www/html/themes  subPath: themes  volumes:  - name: drupal-pvc  persistentVolumeClaim:  claimName: drupal-pv |

**Deployments**

Ahora comenzamos a lanzar todos los objetos

Para crear los volúmenes locales corremos el siguiente comando:

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl apply -f drupal-pv.yaml  persistentvolume/drupal-pv created |

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl apply -f mysql-pv.yaml  persistentvolume/mysql-pv created |

Luego consultamos la creación y el estado de ambos

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl get pv  NAME CAPACITY ACCESS MODES RECLAIM POLICY STATUS CLAIM STORAGECLASS REASON AGE  drupal-pv 5Gi RWO Retain Available manual 78s  mysql-pv 5Gi RWO Retain Available manual 62s |

Para crear los Claim de los volúmenes locales corremos el siguiente comando:

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl apply -f drupal-pvc.yaml  persistentvolumeclaim/drupal-pv created |

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl apply -f mysql-pvc.yaml  persistentvolumeclaim/mysql-pv created |

Luego consultamos la creación y el estado de ambos

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl get pvc  NAME STATUS VOLUME CAPACITY ACCESS MODES STORAGECLASS AGE  mysql-pv Bound mysql-pv 5Gi RWO manual 87s  drupal-pv Bound drupal-pv 5Gi RWO manual 79s |

Para crear los Deployments de las APPs corremos el siguiente comando

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl apply -f mysql.yaml  service/mysql-service created  deployment.apps/drupal-mysql created |

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl apply -f drupal.yaml  service/drupal-service created  deployment.apps/drupal created |

Luego consultamos la creación y el estado de ambos

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl get deployment  NAME READY UP-TO-DATE AVAILABLE AGE  drupal-mysql 1/1 1 1 4m39s  drupal 3/3 3 3 2m58s |

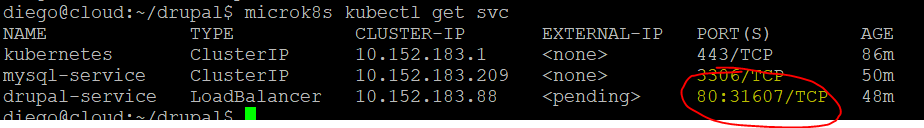
|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl get pod  NAME READY STATUS RESTARTS AGE  drupal-mysql-6995c58f49-mkd2c 1/1 Running 0 4m19s  drupal-549fc98457-29ftb 1/1 Running 0 2m35s  drupal-549fc98457-n7nfn 1/1 Running 0 2m35s  drupal-549fc98457-zngv8 1/1 Running 0 2m35s |

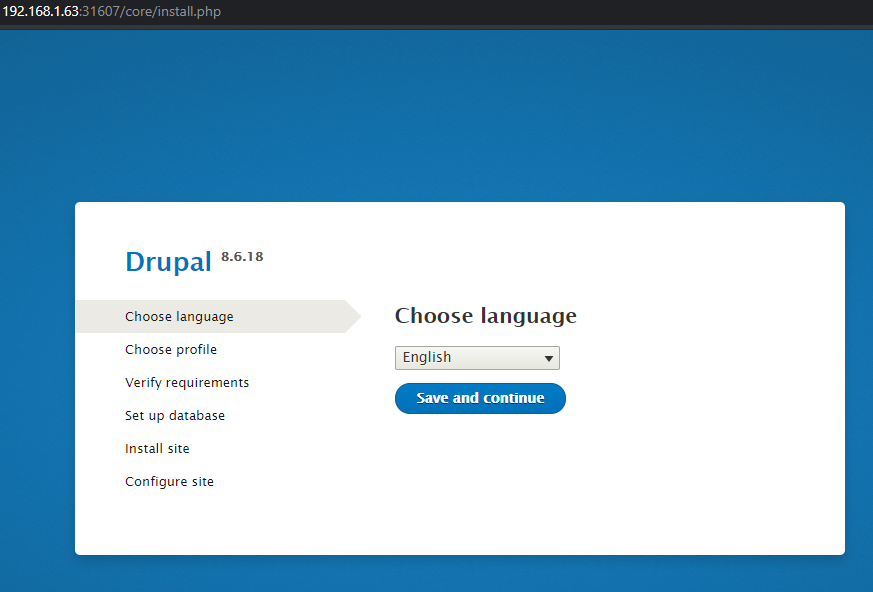
Luego consultamos el estado de ambos servicios

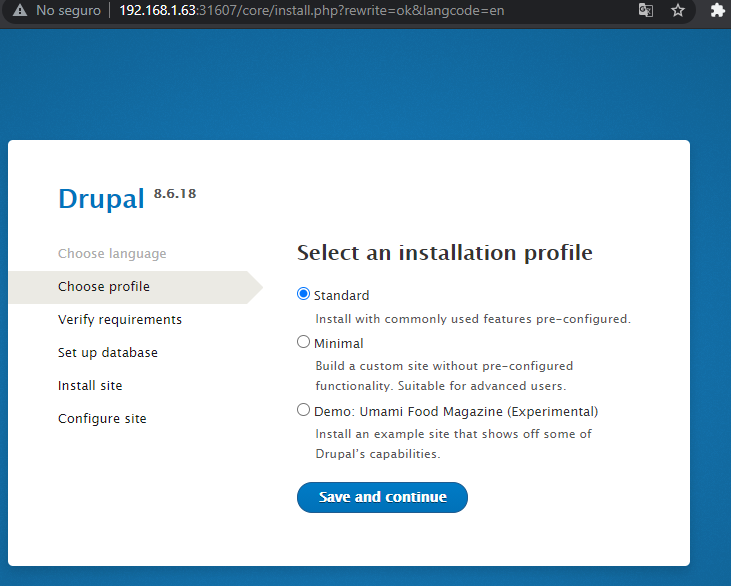
|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl get svc  NAME TYPE CLUSTER-IP EXTERNAL- IP PORT(S) AGE  kubernetes ClusterIP 10.152.183.1 <none> 443/TCP 43m  mysql-service ClusterIP 10.152.183.209 <none> 3306/TCP 6m58s  drupal-service LoadBalancer 10.152.183.88 <pending> 80:31607/TCP 5m18s |

Procedemos con la instalación de Drupal

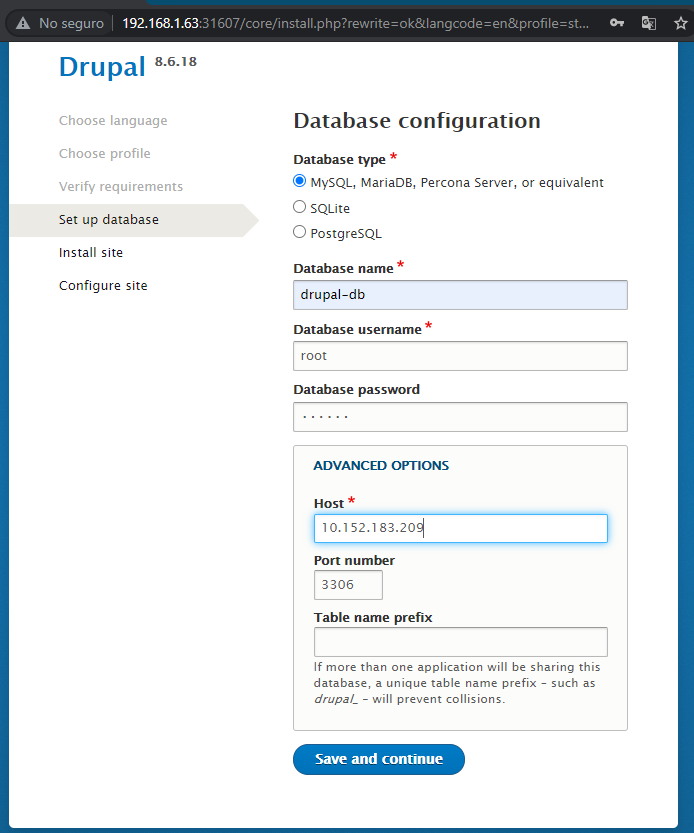
Accedemos a la IP de la VM Local y al puerto asignado al servicio de Drupal:



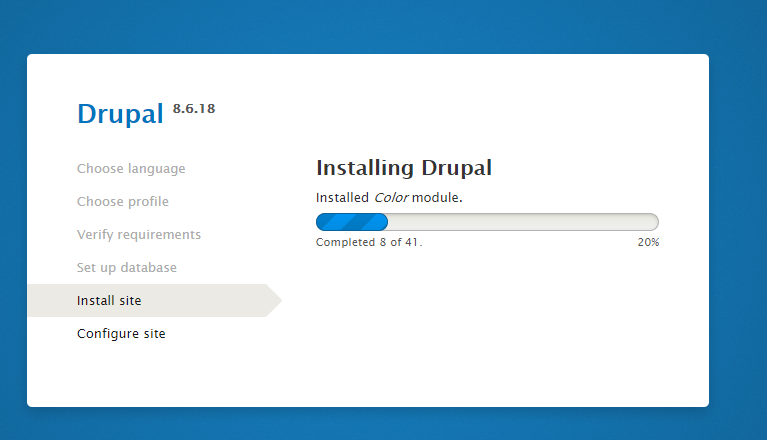
Seleccionamos el Idioma Inglés

Seleccionamos la opción Standard

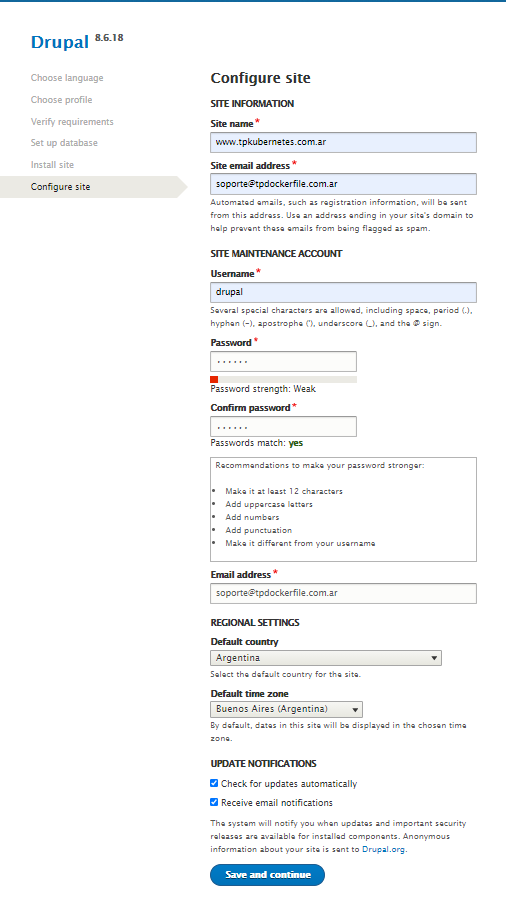
Cargamos los datos correspondientes



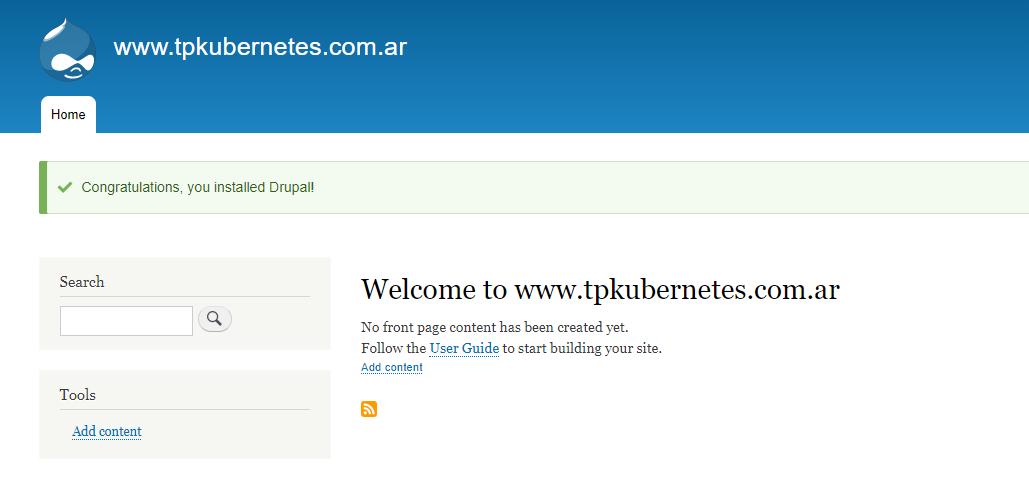
Si todo es correcto debería aparecernos “Installing Drupal”



Una vez que termine la instalación, nos pedirá los datos básicos para iniciar por primera vez.



Una vez terminado nos muestra lo siguiente:



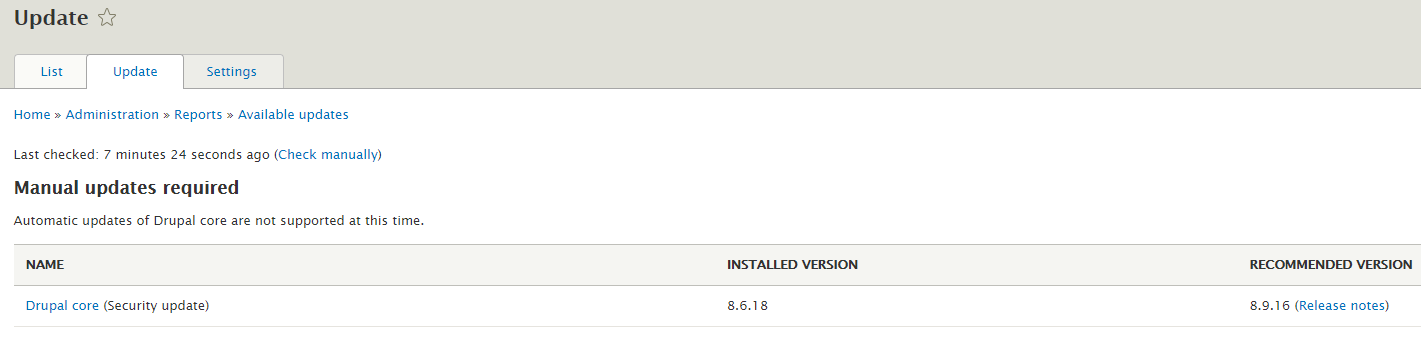
Ahora ya podemos hacer pruebas de Kubernetes como por ejemplo:

Borrar PODs y ver que se vuelven a crear automáticamente.

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl get pod  NAME READY STATUS RESTARTS AGE  drupal-mysql-6995c58f49-mkd2c 1/1 Running 0 110m  drupal-549fc98457-29ftb 1/1 Running 0 108m  drupal-549fc98457-n7nfn 1/1 Running 0 108m  drupal-549fc98457-zngv8 1/1 Running 0 108m  diego@cloud:~/drupal$ microk8s kubectl delete pod drupal-549fc98457-zngv8  pod "drupal-549fc98457-zngv8" deleted  diego@cloud:~/drupal$ microk8s kubectl get pod  NAME READY STATUS RESTARTS AGE  drupal-mysql-6995c58f49-mkd2c 1/1 Running 0 111m  drupal-549fc98457-29ftb 1/1 Running 0 109m  drupal-549fc98457-n7nfn 1/1 Running 0 109m  drupal-549fc98457-rzfvx 1/1 Running 0 27s  diego@cloud:~/drupal$ microk8s kubectl get deployment  NAME READY UP-TO-DATE AVAILABLE AGE  drupal-mysql 1/1 1 1 111m  drupal 3/3 3 3 109m |

Hacer Updates de nuestra aplicación o RollBacks

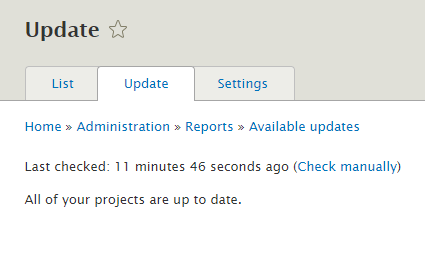
Version antes del Update 8.6.18



En el comando indicamos que vaya a la versión 8.9.x

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl set image deployment drupal drupal=drupal:8.9 --all  deployment.apps/drupal image updated |

Como podemos ver ya quedo actualizado a la versión indicada



Comprobamos que se ha creado un nuevo RecordSet, y unos nuevos pods con la nueva versión de la imagen.

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl get rs  NAME DESIRED CURRENT READY AGE  drupal-mysql-6995c58f49 1 1 1 122m  drupal-69d8979656 3 3 3 5m59s  drupal-549fc98457 0 0 0 120m |

Para hacer un RollBack se usa el siguiente comando:

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl rollout undo deployment/drupal  deployment.apps/drupal rolled back |

Redirección de puertos para acceder al servicio

|  |
| --- |
| diego@cloud:~/drupal$ microk8s kubectl port-forward deployment/drupal 8080:80 --address 0.0.0.0 Forwarding from 0.0.0.0:8080 -> 80 |

Para poder acceder por el nombre del sitio web debemos registrarlo localmente en nuestro archivo Host.

(IP de la Virtual “Nombre de sitio”)

