PORTAINER

José Manuel Fernández Atienza 2º ASIR

Índice

1.Objetivos que se quieren conseguir y se han conseguido

1.1¿Qué quiero conseguir? · 3

2. Fundamentos teóricos y conceptos

2.1 Kubernetes · 3

2.2 Helm \cdot 3

2.3 Longhorn · 3

2.4 Portainer · 3

3. Descripción detallada de lo que se ha realizado

3.1 Instalación y configuración del clúster \cdot 7

3.2 Instalación de Helm \cdot 7

3.3 Instalación de Longhorn · 7

3.4 Instalación Portainer · 7

4.Uso de Portainer

4.1Creación de aplicaciones con Portainer. \cdot 15

4.2Despliegue Wordpress con Mariadb con volumen persistente \cdot 15

4.3Auto escalado en Portainer \cdot 15

4.4Backups con Longhorn \cdot 15

5.Dificultades que se han encontrado \cdot 27

6.Bibliografía · 27

1.Objetivos que se quieren conseguir y se han conseguido

1.1¿Qué quiero conseguir?

Este proyecto tiene como objetivo implementar una solución simplificada para la gestión de clústeres de Kubernetes mediante el uso de Portainer.

El uso de Portainer, permitirá a los usuarios gestionar su infraestructura de Kubernetes de manera más sencilla e intuitiva. Además, se integrará Longhorn como solución de volúmenes persistentes, ofreciendo una forma eficiente y flexible de manejar el almacenamiento.

Por lo tanto, el objetivo principal del proyecto es proporcionar una solución integral que permita a los usuarios gestionar tantas aplicaciones como almacenamiento con mayor facilidad.

Objetivos

- 1. Configurar un clúster de Kubernetes con tres nodos, un nodo máster y dos nodos de computo
- 2. La Implementación de la herramienta Portainer para gestionar dicho clúster
- 3. Conseguir un despliegue de las aplicaciones en alta disponibilidad
- 4. Lograr el auto escalado de las aplicaciones

2. Fundamentos teóricos y conceptos

2.1 Kubernetes

Kubernetes es una plataforma de orquestación de contenedores de código abierto, diseñada para automatizar la implementación, escalabilidad y gestión de aplicaciones en contenedores.

Originalmente desarrollado por Google, Kubernetes permite a los usuarios gestiona aplicaciones distribuidas en múltiples nodos de manera eficiente.

Componentes de Kubernetes:



Control-plane:

Los componentes que forman el plano de control toman decisiones globales sobre el clúster y detectan y responden a eventos del clúster, como la creación de un nuevo pod cuando la propiedad réplicas de un controlador de replicación no se cumple.

- Kube-apiserver: Componente que expone la API de Kubernetes, recibe las peticiones y actualiza acordemente el estado en etcd.
- etcd: Almacén de datos persistente, consistente y distribuido de clavevalor utilizado para almacenar toda la información del clúster.
- kube-scheduler: Componente que está pendiente de los pods que no tienen ningún nodo asignado y selecciona un nodo donde ejecutarlo.
- Kube-controller-manager: Componente que ejecuta los controladores de Kubernetes. Estos controladores incluyen:
 - Controlador de nodos: responsable de detectar y responder cuando un nodo deja de funcionar.

- Controlador de replicación: responsable de mantener el número correcto de pods.
- Controlador de endpoints: construye el objeto endpoindts, es decir, hace la unión entre los services y los pods.
- Controladores de Tokens y cuentas de servicio: crean cuentas y tokens de acceso a la API por defecto para los nuevos namespaces.
- Cloud-controller-manager: ejecuta controladores que interactúan con proveedores de la nube. Incluye:
 - Controlador de nodos: responsable de detectar y actuar cuando un nodo deja de responder.
 - Controlador de rutas: para configurar rutas en la infraestructura de nube subyacente.
 - Controlador de servicios: para crear, actualizar y eliminar balanceadores de carga en la nube.
 - Controlador de volúmenes: para crear, conectar y montar volúmenes e interactuar con el proveedor de la nube para orquestarlos.

Componentes de Nodo:

Los componentes de nodo corren en cada nodo, manteniendo a los pods en funcionamiento y proporcionando el entorno de ejecución de Kubernetes.

- Kubelet: Agente que se ejecuta en cada nodo de un clúster. Se asegura de que los contenedores estén corriendo en un pod.
- Kube-Proxy: permite abstraer un servicio en Kubernetes manteniendo las reglas de red en el anfitrión y haciendo reenvío de conexiones.
- Runtime de contenedores: software responsable de ejecutar los contenedores

Addons

Los addons son pods y servicios que implementan funcionalidades del clúster. Pueden ser administrados por deployments, ReplicationController y otros. Los addons asignados a un espacio de nombres se crean en el espacio kube-system.

2.2 Helm

Helm es un gestor de paquetes para Kubernetes, utilizado para automatizar la instalación, configuración y gestión de aplicaciones dentro de un clúster de Kubernentes.

Helm facilita la instalación y actualización de aplicaciones complejas a través de paquetes llamados Charts. Un chart es una colección de archivos que describen un conjunto de recursos de Kubernentes necesarios para ejecutar una aplicación, como servicios, configmaps, deployments, secrects.

Ventajas de usar Helm:

- Simplifica los despliegues.
- Configuraciones parametrizables
- Control de versiones
- Ecosistema de charts públicos

2.3 Longhorn

Longhorn es un sistema de almacenamiento en bloques distribuido liviano y fácil de usar para Kubernetes. Es un Software libre y de código abierto. Originalmente desarrollado por Rancher Labs, actualmente de desarrolla por la Cloud Native Computing Foundation.

Características de Longhorn:

- Alto rendimiento: Longhorn es capaz de proporcionar un alto rendimiento al gestionar volúmenes distribuidos entre los nodos del clúster.
- Alta disponibilidad: Longhorn replica automáticamente los volúmenes en varios nodos del clúster, lo que significa que, si un nodo falla, los datos son accesibles desde otro nodo que contenga una réplica.
- Escalabilidad: El almacenamiento es fácilmente escalable, permitiendo añadir nodos de almacenamiento adicionales al clúster.
- Resiliencia a fallos: Longhorn ofrece una recuperación ante fallos rápida y automática.
- Interfaz de usuario: Proporciona una interfaz web intuitiva que permite gestionar y monitorear volúmenes de forma sencilla.
- Copias de seguridad y restauración: Permite crear copias de seguridad de los volúmenes.

2.4 Portainer

Portainer es una plataforma de gestión de contenedores de código abierto que ofrece una interfaz de usuario gráfica para facilitar la administración de contenedores, aplicaciones y clústeres de Kubernetes. Originalmente fue diseñado para Docker, pero, en la actualizad ofrece soporte completo para Kubernetes, proporcionando a los usuarios una manera sencilla de gestionar y supervisar recursos de Kubernetes sin necesidad de interactuar con la línea de comandos.



Arquitectura de Portainer

Portainer consta de dos elementos: el servidor de Portainer y el agente de Portainer. Ambos se ejecutan como contenedores en nuestra estructura de Kubernetes. El agente debe implementarse en cada nodo del clúster y configurarse para que informe al contenedor del servidor Portainer.

Un solo servidor aceptará conexiones desde cualquier cantidad de agentes, lo que otorga la capacidad de administrar múltiples clústeres desde una interfaz centralizada. Para ello Portainer requiere persistencia de datos que obtenemos mediante la gestión de volúmenes de Longhorn.

3. Descripción detallada de lo que se ha realizado

En este apartado voy a explicar paso a paso todo el procedimiento que se ha realizado para tener el escenario por completo.

3.1 Instalación y configuración del clúster

Requisitos previos

Necesitaremos instalar estos paquetes en todos los nodos.

Actualizamos todos los paquetes

sudo apt update sudo apt upgrade

Activar los módulos necesarios en el sistema:

```
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf overlay
br_netfilter EOF
sudo modprobe overlay sudo modprobe br_netfilter
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF</pre>
```

Aplicamos los cambios

sudo sysctl -system

Instalamos containerd

sudo apt install containerd

Configuramos ContainerD para que funcione con Kubernetes

containerd config default | sudo tee
/etc/containerd/config.toml >/dev/null 2>&1

Luego editamos el fichero **/etc/containerd/config.toml** y cambiar el parámetro **SystemdCgroup = false** a **true**

<pre>[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.options]</pre>
BinaryName = ""
CriuImagePath = ""
CriuPath = ""
CriuWorkPath = ""
IoGid = 0
IoUid = 0
NoNewKeyring = false
NoPivotRoot = false
Root = ""
ShimCgroup = ""
SystemdCgroup = true

Reiniciamos y ejecutamos al inicio containerd

sudo systemctl restart containerd
sudo systemctl enable container

Instalación de Kubeadm, Kubelet y Kubectl

He seguido los pasos de la documentación oficial:

Instalamos los paquetes necesarios:

```
sudo apt-get install -y apt-transport-https ca-certificates curl
gpg
```

Añadimos los repositorios de Kubernetes

```
curl -fsSL
https://pkgs.k8s.io/core:/stable:/v1.30/deb/Release.key | sudo
gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg
echo 'deb [signed-by=/etc/apt/keyrings/kubernetes-apt-
keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.30/deb/ /' |
sudo tee /etc/apt/sources.list.d/kubernetes.list
```

Actualizamos los repositorios

sudo apt-get update

Instalamos los paquetes

sudo apt-get install -y kubectl kubeadm kubelet

Marcamos los paquetes como hold para que no se actualicen y podamos tener algún problema con las versiones

sudo apt-mark hold kubelet kubeadm kubectl

Iniciamos el servicio de Kubelet antes de usar Kubeadm

sudo systemctl enable --now kubelet

Iniciando el clúster

En el nodo que vamos a usar como controlador del clúster ejecutamos la siguiente instrucción como superusuario

kubeadm init --pod-network-cidr=192.168.0.0/16 --apiserver-certextra-sans=172.22.201.232

Ya que estamos usando como infraestructura tres instancias de OpenStack tendremos que indicar el parámetro --apiserver-cert-extra-sans con la ip flotante del máster, para que el certificado sea válido para esa ip y poder controlar el clúster desde fuera.

Una vez aplicado este comando obtendremos una salida como esta:

```
Your Kubernetes control-plane has initialized successfully!
To start using your cluster, you need to run the following as a
regular user:
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
Alternatively, if you are the root user, you can run: export
KUBECONFIG=/etc/kubernetes/admin.conf
You should now deploy a pod network to the cluster. Run "kubectl
apply -f [podnetwork].yaml" with one of the options listed at:
https://kubernetes.io/docs/concepts/cluster-
administration/addons/
Then you can join any number of worker nodes by running the
following on each as root:
kubeadm join 10.0.0.84:6443 --token b6697i.xsdc63sb9gd0t7hv \
 --discovery-token-ca-cert-hash
sha256:0bcac499fd44015c818165ffac842cb6a8984820ae50e3792ef05fae5
fff18ce
```

En ella podemos ver:

- Las instrucciones necesarias para configurar Kubectl para gestionar el clúster
- La necesidad de instalar un pod para la gestión de la red.
- Las instrucciones para añadir nodos al clúster

Configuración de Kubectl

Para poder controlar el clúster necesitaremos configurar Kubectl, para ello ejecutamos en el nodo máster:

```
mkdir -p $HOME/.kube sudo
cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Con estas instrucciones habremos creado un fichero de configuración para acceder al clúster en el directorio ~/.kube

Ahora para controlar el clúster desde fuera copiaremos el contenido del fichero **~/.kube/config** al fichero de configuración de la máquina que queremos usar para acceder.

Instalación del pod para gestionar la red

Tendremos que instalar un pod que nos permita la comunicación por red de los distintos pods que vamos a ejecutar en el clúster. kubeadm solo soporta plugins de red CNI (Container Network Interface), que es un proyecto que consiste en crear especificaciones y librerías para configure las redes que interconectan los contenedores.

De las distintas alternativas vamos a instalar Calico, para ello:

kubectl apply -f

https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/m anifests/calico.yaml

Comprobamos que todos los pods del namespace kube-system están funcionando con normalidad

kubectl get pods -n kube-system

josema@Debian12:~\$ kubectl get pods -n kub	be-system		
NAME	READY	STATUS	
RESTARTS AGE			
calico-kube-controllers-7dc5458bc6-69z82	1/1	Running	0
2d22h			
calico-node-hcngj	1/1	Running	0
2d22h			
calico-node-vjsmj	1/1	Running	0
2d22h			
calico-node-vqsch	1/1	Running	1
(2d22h ago) 2d22h			
coredns-55cb58b774-m47lk	1/1	Running	0
2d22h			
coredns-55cb58b774-rx5bw	1/1	Running	0
2d22h			
etcd-master	1/1	Running	0
2d22h			
kube-apiserver-master	1/1	Running	0
2d22h			

kube-controller-manager-master	1/1	Running	0
2d22h			
kube-proxy-ktjlk	1/1	Running	0
2d22h			
kube-proxy-qk5r4	1/1	Running	0
2d22h			
kube-proxy-xb2wk	1/1	Running	0
2d22h			
kube-scheduler-master	1/1	Running	0
2d22h		_	
metrics-server-7995dd4965-fcd9p	1/1	Running	0
2d22h			

Uniendo nodos al clúster

En cada nodo que va a formar parte del clúster tenemos que ejecutar como superusuario el comando que nos ofreció la salida del comando Kubeadm al iniciar clúster en el master, en mi caso

kubeadm join 10.0.0.84:6443 --token b6697i.xsdc63sb9gd0t7hv \ --discovery-token-ca-cert-hash sha256:0bcac499fd44015c818165ffac842cb6a8984820ae50e3792ef05fae5 fff18ce

Comprobamos que se han añadido los nodos

josema@Debian12:~\$ kubectl get nodes												
NAME	STATUS	ROLES	AGE	VERSION								
master	Ready	control-plan	e 2d22h	v1.30.8								
nodo1	Ready		2d22h	v1.30.8								
nodo2	Ready		2d22h	v1.30.8								

3.2 Instalación de Helm

Necesitaremos tener instalado Helm para realizar las instalaciones tanto de Portainer como de Longhorn

Para instalar Helm, añadimos los repositorios de Helm

```
curl https://baltocdn.com/helm/signing.asc | gpg --dearmor | sudo tee
/usr/share/keyrings/helm.gpg > /dev/null
sudo apt-get install apt-transport-https --yes
echo "deb [arch=$(dpkg --print-architecture) signed-
by=/usr/share/keyrings/helm.gpg]
https://baltocdn.com/helm/stable/debian/ all main" | sudo tee
/etc/apt/sources.list.d/helm-stable-debian.list
```

Actualizamos los repositorios e instalamos Helm

sudo apt-get update sudo apt-get install Helm

Con esto ya podremos instalar nuestras aplicaciones usando Helm

3.3 Instalación de Longhorn

Para instalar Longhorn debemos de satisfacer unos requisitos previos instalados en todos los nodos del clúster, para ello consultando la documentación oficial, los requisitos necesarios son:

- Open-iscsi instalado y el demonio iscsid corriendo en todos los nodos
- Tener un cliente NFS instalado en todos los nodos
- Tener instalado los siguientes paquetes: curl, findmnt, grep, awk, blkid, lsblk.

Ahora vamos a pasar a instalar los requisitos previos en todos los nodos:

Instalación de los paquetes necesarios

Instalación de los paquetes necesarios

sudo apt update sudo apt install -y curl findmnt grep awk blkid lsblk

Instalamos el paquete open-iscsi

sudo apt install open-iscsi

Arrancamos los servicios iscsid

sudo systemctl enable iscsid.socket sudo systemctl enable iscsid.service sudo systemctl start iscsid.socket sudo systemctl start iscsid.service

Activamos el módulo iscsi_tcp

sudo modprobe iscsi_tcp

Para permitir que este módulo se active solo al arranque de las máquinas tendremos que añadir una entrada en el fichero /etc/modules con el nombre del módulo, en este caso iscsi_tcp

Instalamos el paquete nfs-common

sudo apt install nfs-common

En la documentación oficial de Longhorn disponemos de un script para comprobar si nuestro clúster cumple con todos los requisitos

https://github.com/longhorn/longhorn/blob/v1.7.2/scripts/environment_check.sh

Compruebo con el script

```
iosema@Debian12:~/Cluster/Longhorn$ ./requirements.sh
       Required dependencies 'kubectl jq mktemp sort printf' are
[INF0]
installed.
[INF0]
       All nodes have unique hostnames.
       Waiting for longhorn-environment-check pods to become ready
[INF0]
(0/2)...
       All longhorn-environment-check pods are ready (2/2).
[INF0]
[INF0]
       MountPropagation is enabled
       Checking kernel release...
[INF0]
[INF0]
       Checking iscsid...
       Checking multipathd...
[INF0]
[INF0]
       Checking packages...
       Checking nfs client...
[INF0]
       Cleaning up longhorn-environment-check pods...
[INF0]
[INF0]
       Cleanup completed.
```

Para instalar Longhorn, añadimos el repositorio Helm de Longhorn

helm repo add longhorn https://charts.longhorn.io

Actualizamos los repositorios de Helm

helm repo update

Instalamos Longhorn

helm install longhorn longhorn/longhorn --namespace longhornsystem --create-namespace --version 1.7.2

Este comando nos creará un nuevo namespace llamado Longhorn-system donde instalará Longhorn

Comprobamos que Longhorn está corriendo correctamente

kubectl -n longhorn-system get pod

josema@Debian12:~/Cluster/Longhorn\$ kubectl -n long	horn-sys [.]	tem get poo	k	
NAME	READY	STATUS	RESTARTS	AGE
csi–attacher–698944d5b–2fz56	1/1	Running	0	2d1h
csi-attacher-698944d5b-2g4qg	1/1	Running	0	2d1h
csi-attacher-698944d5b-6cg9d	1/1	Running	0	2d1h
csi-provisioner-b98c99578-27vfp	1/1	Running	0	2d1h
csi-provisioner-b98c99578-pfqgp	1/1	Running	0	2d1h
csi-provisioner-b98c99578-q225d	1/1	Running	0	2d1h
csi-resizer-7474b7b598-9xg9w	1/1	Running	0	2d1h
csi-resizer-7474b7b598-gwh48	1/1	Running	0	2d1h
csi-resizer-7474b7b598-vwkrd	1/1	Running	0	2d1h
csi-snapshotter-774467fdc7-jn45b	1/1	Running	0	2d1h
csi-snapshotter-774467fdc7-jr4zb	1/1	Running	0	2d1h
csi-snapshotter-774467fdc7-trt9c	1/1	Running	0	2d1h
engine-image-ei-51cc7b9c-tx6nq	1/1	Running	0	2d1h
engine-image-ei-51cc7b9c-vh2r4	1/1	Running	0	2d1h
<pre>instance-manager-35fcd1a915d0e1f4276548b11e3a1df3</pre>	1/1	Running	0	2d1h
<pre>instance-manager-4cfc91fff903489289419a25a1128943</pre>	1/1	Running	0	2d1h
longhorn-csi-plugin-9n6v4	3/3	Running	0	2d1h
longhorn-csi-plugin-jwpgf	3/3	Running	0	2d1h
longhorn-driver-deployer-7d77d779dd-vkcxj	1/1	Running	0	2d1h
longhorn-manager-dcghk	2/2	Running	0	2d1h
longhorn-manager-xzrqz	2/2	Running	0	2d1h
longhorn-ui-7c784d8587-n5tv6	1/1	Running	0	2d1h
longhorn-ui-7c784d8587-wfd5f	1/1	Running	0	2d1h
josema@Debian12:~/Cluster/Longhorn\$				

2º ASIR

3.4 Instalación Portainer

Al igual que con Longhorn instalaré portainer usando Helm. En mi caso voy a usar la versión Busisness de Portainer, ya que he obtenido una prueba gratuita para poder usar con tres nodos. Para poder instalar Portainer necesitaremos persistencia de datos que lograremos gracias a Longhorn. Para comprobar los storageclass que disponemos en nuestro clúster

kubectl get sc

josema@Debian12:~/Cluster/Longhorn\$ kubectl get sc												
NAME	PROVISIONER	RECLAIMPOLICY	VOLUMEBINDINGMODE	ALLOWVOLUMEEXPANSION	AGE							
longhorn (default)	driver.longhorn.io	Delete	Immediate	true	2d1h							
longhorn-static	driver.longhorn.io	Delete	Immediate	true	2d1h							
josema@Debian12:~/Cl	luster/Longhorn\$											
,												

Tendremos que asegurarnos de que Longhorn es el storageclass por defecto, en caso de que no lo sea, podemos hacerlo usando

kubectl patch storageclass longhorn -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-defaultclass":"true"}}'

Para instalar Portainer usando Helm, añadimos los repositorios de Portainer para Helm

```
helm repo add portainer https://portainer.github.io/k8s/
helm repo update
```

Instalamos Portainer

```
helm upgrade --install --create-namespace -n portainer portainer
portainer/portainer \\
    --set enterpriseEdition.enabled=true \\
    --set enterpriseEdition.image.tag=2.21.4 \\
    --set tls.force=true
```

Al igual que antes Helm creará un nuevo namespace donde instalará portainer.

Con este comando estamos exponiendo la web de portainer al exterior mediante un servicio NodePort, por lo tanto, podemos acceder mediante la ip pública del nodo master y usando el puerto 30779

```
https://172.22.201.232:30779
```

Una vez dentro nos pedirá crear una nueva contraseña para el usuario Admin

	portainer.io	
✓ New Portainer ins	tallation	
Please create the initial admi	nistrator user.	
Username	admin	
Password	••••••	
Confirm password	•••••	~
▲ The password must b	e at least 12 characters long. 🗸	
Create user		
Allow collection of anonyr	nous statistics. You can find more information about this	in our privacy

4.Uso de Portainer

En este apartado explicaré brevemente algunos casos prácticos del uso de portainer para desplegar aplicaciones.

4.1Creación de aplicaciones con Portainer.

En este apartado mostraremos rápidamente el despliegue de una aplicación sencilla mediante Portainer.

En este caso voy a desplegar un despliegue de Nginx y lo mostraremos mediante un servicio NodePort

Para desplegar las aplicaciones tendremos distintos métodos, mediante una imagen Docker (add with form) o mediante charts de Helm o repositorios (add with manifest

←	\rightarrow C	O 🔒 http	s:// 172.22.201.232 :30779/#!/1/kubernetes/applications	☆ ♡ ④ 約 目
	portainer.io		Applications Application list \mathcal{C}	♬ ⓒ Զadmin ∨
â	Home		O Applications	I≣ Stacks
@	local		Applications Q Search	Remove + Add with form + Create from manifest
	Dashboard		System resources are hidden, this can be changed in the table settings.	▼ Namespace apps ~
C	Custom Templates			
\$	Namespaces		Name ↓↑ Stack ↓↑ Namespace ↓↑ Image ↓↑ A	pplication Type ↓↑ Filters 🗸 Status Published Created ↓↑
♡	Applications		No application qualit	abla.
8	Networking		но аррісація ачала	una.
۵	ConfigMaps & Secrets			Items per page 10 🗸
8	Volumes			
8≣	More Resources			
8	Cluster			

Si seleccionamos add with form, tendremos que especificar el nombre de la imagen Docker que vamos a desplegar, un namespace y el servicio por el cuál ofrecemos la aplicación.

Portainer

\leftarrow	→ C	O 🔒 http	os:// 172.22.201.232 :30779/#	/1/kubernetes/applications/new	\$	∵ ೨	മ ≡
ŧ	portainer.io		Applications > Create an app Create applicat	ication	4 3	음 adm	in ~
â	Home						
@	local		Namespace	apps			~
	>_ kubecti shell		Name*	nginx			
	Dashboard		Registry	Docker Hub (anonymous)			~
C	Custom Templates					_	
۲	Namespaces		Image*	docker.io nginx		👉 Sea	rch
Ø	Applications		Advanced mode				
*	Networking		You are currently using a	n anonymous account to null images from DockerHub and will be limited to 100	pulls every 6 hours. You can confi	aure DockerHu	b
۵	ConfigMaps & Secrets		authentication in the Rec	istries View. Remaining pulls: 100/100		gure bookerria	
8	Volumes		Note	Enter a note about this application			
8≣	More Resources						
8	Cluster						
Adr	ninistration						h,

En mi caso he seleccionado la imagen nginx en el namespace apps.

Si bajamos más abajo encontraremos la sección services

\leftarrow	\rightarrow G	0	6	https://	172.22.2	01.232:30779/#!/	1/kubernetes/app	olications/ne	W			ŝ	${\times}$	۲	பி	≡
*			_		Publis	shing the applic	ation									
Ĩ					> Expl	lanation										
					Clu	usterIP services	NodePort servi	ices 1	LoadBala	ncer services						
â	Home															
					• A	Allow access to traffic	external to the clus	ster via a Node	Port service	a. Not generally re-	commended for Product	ion use.				
<u>@</u>	local				k											
	>_ kubecti shell					NodePort							🗎 Remove ser	vice		
	Dashboard					Annotations ⑦										
C	Custom Templates					+ Add annotation										
۲	Namespaces					Dorte										
Ø	Applications					Ports										
ጽ	Networking					Container port*	80 0	Service port	80	Nodepor	rt e.g. 3008 🗘		🛱 Remove por	t		
۵	ConfigMaps & Secrets					TCP UDP										
8	Volumes															
8≣	More Resources					+ Add port										
8	Cluster															
Adr	ministration				+	Create service										

Aquí tendremos que especificar el puerto usado por el contenedor, y el puerto usado por NodePort para exponer la aplicación, si dejamos el campo NodePort en blanco automáticamente Portainer asignará un puerto aleatorio.

Si continuamos podemos observar más opciones como la replicación de los pods, o la ubicación de estos, además de la reserva de los recursos usados por el pod.

$\leftarrow \rightarrow G$ (2 🔒 https:// 172.22.201.232 :30779/#1/1/kubernetes/applications/new	公	${\times}$	ے ی	j ≡
	Add persisted folder				
	Resource reservations				
🛍 Home	Resource reservations are applied per instance of the application.				
local	(MB) 3895	0			
>_ kubecti shell	CPIL limit @	1.41			
Dashboard					
Custom Templates					
Namespaces	Deployment				
Applications	Select how you want to deploy your application inside the cluster.				
る Networking	Sector Clobal			C	
合 ConfigMaps & Secrets	Run one or multiple instances of this container	be deployed as a DaemonSe	et with an	instance	
Olumes	on each node o	r the cluster			
8 More Resources	v				
8 Cluster	Instance count* 2				
Administration	Enable auto scaling for this application				
兴 User-related	Placement preferences and constraints				

He seleccionado 2 replicas para este pod

Para desplegar la aplicación pulsaremos en el botón Deploy Application

~	\rightarrow C	O 🔒 http	os:// 172.22.201.232 :30779/#!/1/	kubernetes/applicatio	ons		5	9	ා එ ≡
a	portainer.io		Applications Application list \$\appril 1				Ļ	⑦ ₽ a	idmin 🗸
â	Home						≔ Stacks		
<u>@</u>	local		Applications	Q [\$earch		🔋 Remove 🕇 A	add with form + C	reate from manifes	st i
	Dashboard		 System resources are hidde 	n, this can be changed in	the table settings.		♥ Namespace	apps	~
Ľ	Custom Templates								
۲	Namespaces		○ > Name ↓↑ Stack	↓↑ Namespace ↓↑	Image J1 Appli	ication Type $\downarrow \uparrow$ Filters $oldsymbol{\nabla}$	Status	Published	Created U
Ø	Applications		nainx -	apps	nginx:latest Deplo	ovment	Replicated 0 /	2 Yes	2024-12-17
*	Networking					-,	,		
۵	ConfigMaps & Secrets							Items per page	10 ~
8	Volumes								

Esperamos que se realice el deploy

←	\rightarrow G	🔿 🔓 http	s:// 172.22.201.232 :30779/#!/1/k	ubernetes/applicatio	ons		☆	♥ @	ම එ ≡
a	portainer.io	×	Applications Application list \mathcal{C}				Ļ	? 2	idmin ~
â	Home						≔ Stacks		
۲	local	×	Applications	Q Search		Remove	+ Add with form + (Create from manifes	st :
▣	Dashboard		O System resources are hidder	, this can be changed in	the table settings	l.	▼ Namespace	apps	~
Ľ	Custom Templates								
۲	Namespaces		→ Name ↓↑ Stack	Namespace 🗤	Image ↓↑	Application Type 🕕 🕫	Iters 🖓 Status	Published	Created U
Ŷ	Applications		nginy -	anns	ndiny:latest	Deployment	Replicated 2	12 Yes	2024-12-17
*	Networking	~		uppo	nginxiacor		• Replicated 2	2 103	2024 12 17
۵	ConfigMaps & Secrets							items per page	10 ~
8	Volumes								

Cuando el deploy haya finalizado podremos ver nuestra aplicación desplegada en nuestro namespace apps y como vemos disponemos de dos replicas.

Para acceder a la aplicación necesitaremos conocer el puerto del servicio NodePort donde está expuesto, para ello entramos a los detalles de la aplicación

$\leftarrow \rightarrow C$	O 🔓 http	os:// 172.22.201.232 :30779/#!/	1/kubernetes/applica	ations/apps/nginx?re	source-type=Depl	oyment 🏠	♡ (2)	பி	=
portainer.io		Edit this application		ර Redeploy හි Rollb	ack to previous confi	guration ()			
l Home		Accessing the applic	ation						
🙆 local		This application is expose	ed through service(s) as	below:					
>_ kubecti shell		Service name	Туре	Cluster IP	External IP	Container port	Service port(s)		
Dashboard		nainx	NodePort	10.110.118.118		80	80 : 31855/TCP		
Custom Templates									

Como vemos nuestra aplicación esta expuesta en el puerto 31855 por lo tanto, accederemos mediante la ip pública del nodo master y el puerto

		S S 1 -
Welcome to nginx!		
If you see this page, the nginx web server is successfully working. Further configuration is required.	installed and	
For online documentation and support please refer to <u>ngin</u> Commercial support is available at <u>nginx.com</u> .	<u>nx.org</u> .	
Thank you for using nginx.		

Como vemos hemos accedido a nuestra aplicación de Nginx

Si seleccionamos5 la opción add with form veremos las distintas formas que tenemos de añadir nuestra aplicación

$\leftarrow \ \ \rightarrow \ \ \mathbf{G}$	O 🔒 https:// 172.22.201.232 :30779/#!/1/kubernetes/deploy?referrer=kubernetes.ap	pplications ☆ 🛛 🕲 🖞 🗄	=
portainer.io	Create from manifest $arsigma$	4 🔿 2 admin v	
 Home local kubecti shell 	Deploy Deploy from	۵ Logs	
 ๗ Dashboard ๗ Custom Templates ⊗ Namespaces ⊗ Applications 	Web editor URL C	pository Custom template	
Networking ConfigMaps & Secrets Volumes More Resources Cluster	 Deploy to Use namespace(s) specified from manifest Namespace apps 	~	

Observamos que podemos añadir aplicaciones mediante:

- Editor web: escribiendo directamente en la web nuestra definición de app en Yaml.
- URL: especificando la URL de una imagen Docker
- Repositorio: especificando la url de nuestro repositorio GitHub donde tenemos nuestros ficheros Yaml

- Ficheros personalizados: Subiendo nuestros ficheros Yaml
- Helm: usando charts provisionados por Helm

4.2Despliegue Wordpress con Mariadb con volumen persistente

En este apartado voy a realizar un despliegue completo de una aplicación wordpres conectado a una base de datos MariaDB usando volúmenes persistentes

Para comenzar empezaremos desplegando la base datos mariaDB, para ello añadiremos nuestra aplicación mediante una imagen Docker especificando las variables necesarias para la creación de la base datos.

Empezaremos definiendo el namespace, el nombre y la imagen Docker

~	\rightarrow C	O 🔓 http	https:// 172.22.201.232 :30779/#!/1/kubernetes/applications/new									
1	portainer.io		Applications > Create an app Create applicat	ilication	1	¢ 0	OC	adm	in	~		
â	Home			k								
6	local		Namespace	apps					\sim			
	>_ kubecti sheli		Name*	mariadb								
	Dashboard		Registry	Docker Hub (aponymous)					~			
Ľ	Custom Templates											
٢	Namespaces		Image*	docker.io mariadb				🌶 Sea	rch			
\$	Applications		Advanced mode									

Nos desplazamos hasta la parte de Variables de entorno para añadir las variables de entorno

- MARIADB_USER
- MARIADB_PASSWORD
- MARIADB_ROOT_PASSWORD
- MARIADB_DATABASE

	>_ kubecti shell		Environm	nent variables									
	Dashboard		O Switch 1	Switch to advanced mode to copy & paste multiple variables									
Ľ	Custom Templates		name*	MARIADB_USER	value	josema	1						
۲	Namespaces												
0	Applications		name*	MARIADB_PASSWORD	value	josema							
å	Networking	~	name*	MARIADE ROOT DASSWORD	value	locema	A						
۵	ConfigMaps & Secrets		name		Value	Joseffia							
8	Volumes		name*	MARIADB_DATABASE	value	I wordpress							
8≣	More Resources	~	+ Add :	an environment variable									
00	Cluster	~	, Add a										

Para crear un nuevo volumen persistente nos desplazamos hasta carpetas persistentes y tendremos que especificar el punto de montaje del volumen, si deseamos crear un nuevo volumen y el tamaño. Ya que uso Longhorn creare el

volumen en Longhorn

	local	Path in container*	container* /var/lib/mysql Ne		Existing volume	Requested size*	5	$\hat{\cdot}$	GB 🗸	Ē	
	>_ kubecti sheli	Storage longho	rn 🗸								
	Dashboard										
C	Custom Templates	+ Add persisted	folder								

Para que la base de datos sea accesible por WordPress necesitamos exponer la aplicación mediante un servicio ClusterIp, especificando el puerto 3306 que es el usado por defecto por MariaDB

ClusterIP services 1 NodePort services LoadBalancer services ClusterIP services 1 NodePort service, optionally exposing externally to the outside world via an ingress. ClusterIP Annotations Applications Applications Applications Contrainer port* 3308 Contrainer port* 308 Contrainer po	+ norteiner is	u	> Explanation
 Home I coal 2 kubect shell Dashboard Custom Templates Namespaces Applications A todamoral ConfigMaps & Secrets Volumes More Resources Cluster 	BUSINESS EDITION		ClusterIP services 1 NodePort services LoadBalancer services
 local kubectishell Dashboard ClusterIP Annotations ? Applications Applications Networking Contignaps & Secrets Volumes Contignaps & Secrets Volumes Cluster Cluster + Add port + Create service 	n Home		O Publish internally in the cluster via a ClusterIP service, optionally exposing externally to the outside world via an ingress.
 □ Dashboard □ Dashboard ○ Custom Templates ○ Applications ○ Applications ○ ConfigMaps & Secrets □ Volumes □ Volumes □ More Resources □ Cluster → Create service 	Iocal Local Local Local		ClusterIP Remove service
 Custom Templates Namespaces Applications Networking ConfigMaps & Secrets Volumes Kore Resources Cluster Cluster 	🖬 Dashboard		+ Add annotation
 Namespaces Applications Networking ConfigMaps & Secrets Volumes More Resources Cluster 	Custom Templates		k k
Applications A Networking ConfigMaps & Secrets Volumes More Resources Cluster	S Namespaces		Ports
♣ Networking ✓ ▲ ConfigMaps & Secrets Expose via Ingress ⑦ ♥ Volumes + Add port ﷺ More Resources ✓ ♥ Cluster + Create service	Applications		Container port* 3306 Service port* 3306 TCP UDP
A ConfigMaps & Secrets Volumes More Resources Cluster	ቆ Networking	~	
B Volumes + Add port B Cluster + Create service	合 ConfigMaps & Secrets		
8 Cluster Clus	Olumes		+ Add port
8 Cluster + Create service	8 More Resources	~	
	8 Cluster	~	+ Create service

Con esto ya tendríamos la base de datos accesible desde otro pod del Clúster, Ahora desplegaremos Wordpress.

Tendremos que desplegar nuestra app WordPress en el mismo namespace en el que hemos desplegado la base de datos

a	Portainer.io	Applications > Create an applicat	plications > Create an application create application ${\cal O}$					
â	Home							
6	local	Namespace	apps		~			
	>_ kubectl shell	Name*	wordpress					
	Dashboard	Registry	Docker Hub (anonymous)		~			
C	Custom Templates							
\$	Namespaces	Image*	docker.io wordpress		👉 Search			
⇔	Applications	Advanced mode						

Ahora tendremos que definir las variables de entorno para que Wordpress pueda acceder a la base de datos

- WORDPRESS_DB_HOST
- WORDPRESS_DB_USER
- WORDPRESS_DB_NAME
- WORDPRESS_TABLE_PREFIX

Portainer

	>_ kubecti shell		Environm	nent variables			
	Dashboard		Advance	ced mode			
C	Custom Templates		Switch	to advanced mode to copy a paste multiple variables			
۲	Namespaces		name*	WORDPRESS_DB_HOST	value	mariadb	Û
\$	Applications		name*	WORDPRESS DB USER	value	losema	fil
ጽ	Networking	~				,	
۵	ConfigMaps & Secrets		name*	WORDPRESS_DB_PASSWORD	value	josema	Û
8	Volumes		namo*	WODDDDESS DD NAME	value	wardprace	A
8≣	More Resources	~	name	WORDFRESS_DB_NAME	value	Withhese	
8	Cluster	~	name*	WORDPRESS_TABLE_PREFIX	value	wp	
Adr			+ Add a	an environment variable 🗘 Load variables from .env file			

2º ASIR

Ahora tendemos que exponer nuestra aplicación mediante un servicio NodePort para poder acceder a ella desde fuera

<i>.</i>	BUSINESS EDITION	Publishing the application	
â	Home	> Explanation	
		ClusterIP services NodePort services 1 LoadBalancer s	ervices
*	local	Allow access to traffic external to the cluster via a NodePort service. Not	enerally recommended for Production use.
	Dashboard	MadeDart	
Ľ	Custom Templates		
۵	Namespaces	Amotations	
♡	Applications	+ Add annotation	
*	Networking	Ports	
۵	ConfigMaps & Secrets		
8	Volumes	Container port* 80 C Service port* 80 C	Nodeport e.g. 300€ ♀
8≣	More Resources	TCP UDP	
8	Cluster		
Adı		+ Add port	

Desplegamos nuestra aplicación y esperamos a que se creen los pods

←	\rightarrow G	0 🔒 h	nttps:// 172 .	22.201.232:307	79/#!/1/kub	ernetes/applicatio		\$		ා එ ≡	
a	portainer.io		Applica	ations plication l	ist 🛛				Ċ	⑦ A a	dmin 🗸
â	Home				Ø	Applications					
@	local		6	Applicatio	ns	Q Şearch		Remove	+ Add with form + Cre	ate from manifes	t :
	Dashboard		0	System resources	are hidden, th	is can be changed in	the table settings.		▼ Namespace	apps	~
Ľ	Custom Templates				k						
۲	Namespaces			> Name ↓↑	Stack 11	Namespace $\downarrow\uparrow$	lmage ↓↑	Application Type $\downarrow\uparrow$	Filters 🖓 Status	Publis	hed Creat
Ø	Applications			mariadb	-	apps	mariadb:latest	Deployment	Replicated	1 / 1 Yes	2024-
8	Networking										
۵	ConfigMaps & Secrets			nginx	-	apps	nginx:latest	Deployment	 Replicated 	2 / 2 Yes	2024-
8	Volumes							Dealerment	Deallasted	4 / 4 - V	0004
8≣	More Resources			wordpress	-	apps	wordpressitatest	Deployment	 Replicated 	1/1 Yes	2024-
0	Cluster	~								Items per page	10 ~

Accedemos a nuestra app Wordpress y realizar la instalación de Wordpress



4.3. Auto escalado en Portainer

Para poder realizar el auto escalado en un clúster de Kubernetes deberemos tener instalado la función de métricas. En mi caso he instalado metric-server.

kubectl apply -f https://github.com/kubernetes-sigs/metricsserver/releases/latest/download/components.yaml

Con esto desplegaremos el servicio de métricas en nuestro clúster, pero obtendremos un error de certificados, para solucionar este error, tendremos que editar el despliegue y añadir los siguientes argumentos:

```
args:
    - -kubelet-insecure-tls
    - -kubelet-preferred-address-types=InternalIP
```

kubectl edit pod -n kube-system metrics-server-7995dd4965-fcd9p



Comprobamos que se está ejecutando correctamente el servicio de métricas ejecutando el siguiente comando

josema@De	ebian12:~/Clus	ster/Lor	nghorn\$ kubectl [.]	top nodes
NAME	CPU(cores)	CPU%	MEMORY(bytes)	MEMORY%
master	251m	12%	1949Mi	51%
nodo1	188m	9%	2038Mi	53%
nodo2	205m	10%	2552Mi	66%

Con este servicio ya podemos configurar el auto escalado en Portainer, como sabemos Portainer es una herramienta muy sencilla de usar para todo tipo de usuarios.

Para usar el auto escalado simplemente nos bastaría con seleccionar la opción de auto escalado y seleccionar el número mínimo y máximo de replicas, y el tanto por ciento de uso CPU al que debe activarse el auto escalado.

En mi caso he configurado para que se desplieguen como máximo 5 replicas y que se repliquen cuando se use un 20% de CPU.

Para ello voy a realizar un nuevo despliegue de una nueva aplicación nginx llamada nginx-escalable para probar

Portainer

local		Deployment					
>_ kubecti shell		Select how you want to	o deploy your application	inside the cluster.			
🖬 Dashboard				0	Global		С
🖉 Custom Templates		Replica	nutiple instances of this	container	Application	will be deployed as a DaemonSe	et with an instance
Namespaces					on each no	de of the cluster	
Applications							
ቆ Networking	~	Instance count*	1	0			
合 ConfigMaps & Secrets		Enable auto scaling for					
🖯 Volumes		this application					
8 More Resources	~	Minimum instances	^	Maximum instances	^	anget CPO usage (%)	^
	0		~	5	~	24	2

2º ASIR

Desplegamos la aplicación .

Para realizar las pruebas de carga a un servidor web he usado la utilidad apache2utils para realizar peticiones simultaneas al servidor.

Para realizar las peticiones:

ab -n 1000000 -c 500 http://172.22.201.232:30743/

Ahora veremos en directo la salida del comando kubectl get pods para ver cómo se crean las réplicas.

NAME nginx-escalable-b7fcdb5d5-5lwsf nginx-escalable-b7fcdb5d5-8wftr nginx-escalable-b7fcdb5d5-9qcmz nginx-escalable-b7fcdb5d5-mjclx nginx-escalable-b7fcdb5d5-pnr4x traefik-deployment-667459fcf8-cdkb7 whoami-57b48994d9-x5sf7	READY 1/1 1/1 1/1 1/1 1/1 1/1 1/1	STATUS Running Running Running Running Running Running	RESTARTS 0 0 0 0 0 0 0	AGE 84s 37s 52s 52s 52s 27h 27h	
66% 180 75% 193 80% 204 90% 494 95% 1175 98% 1206 99% 1240 100% 15395 (longest request) josema@Debian12:~/Cluster/traefik\$ ab This is ApacheBench, Version 2.3 <\$Re Copyright 1996 Adam Twiss, Zeus Techn Licensed to The Apache Software Found Benchmarking 172.22.201.232 (be patie) -n 1000 evision: nology Lt lation, h	0000 -c 500 1913912 \$> cd, http:// ttp://www.) http://172 /www.zeustec apache.org/	2.22.201.232:30743 h.net/	37
Completed 100000 requests					

Como vemos se han replicado los pods hasta llegar al máximo de replicas. Como vemos tenemos 4 pods creados más tarde que el primer pod.

4.4.Backups con Longhorn

En este apartado comprobaremos la facilidad que nos ofrece Longhorn para restaurar backups de los volúmenes.

Para ello usaremos el despliegue de Wordpress con la base de datos, en el cual vamos a crear dos entradas. Cuando creemos la primera entrada realizaremos el snapshot y luego crearemos la segunda entrada para comprobar que al restaurar la snapshot no existe la segunda entrada.

$\leftarrow \ \ \rightarrow \ \ \mathbf{G}$	○ 172.22.201.232:32580/wp-	3	% ☆	ଓ ୬ ମି ≡		
🚯 🏦 Wordpress-Jose	ma 😋 1 🛡 0 🕂 Añadir					Hola, josema 🥅
🍘 Escritorio	Entradas Añadir una nueva entrada				Opciones de p	oantalla ▼ Ayuda ▼
📌 Entradas 🛛 🔸	Todo (2) Publicados (2)					Buscar entradas
Todas las entradas Añadir una nueva	Acciones en lote V Aplicar	as las fechas 🗸 🗌 Todas las ca	ategorías 🗸 🛛 Filtrar			2 elementos
entrada	🗌 Título 崇	Autor	Categorías	Etiquetas	P ÷	Fecha 븆
Categorias Etiquetas	Antes del snapshot	josema	Sin categoría	_	_	Publicada 17/12/2024 a las 23:54
9) Medios 📕 Páginas	📄 ¡Hola, mundo!	josema	Sin categoría	-	P	Publicada 17/12/2024 a las 23:53
Comentarios	🗋 Título 🛱	Autor	Categorías	Etiquetas	• ÷	Fecha 🖨

Tenemos la primera entrada creada, me dispongo a crear el snapshot del volumen de mariadb

Desde la interfaz web de Longhorn navegamos hasta el volumen asociado a mariadb y seleccionamos la opción crear snapshot

$\leftarrow \ \ \rightarrow \ \ C$	0 172.22.201.2	2.22.201.232 :30672/#/volume/pvc-84c3f1f0-2568-4168-b5cd-63b8a8fb1e27						\bigtriangledown	۲	ப்	≡
V LONGHORN	迪 Dashboard	🗔 Node	E Volume	C Recurring Job	🕽 Backup	ll Setting ∨					
⊟Volume / pvc-84c3	f1f0-2568-4168-	b5cd-63b8a	a8fb1e27							:≡ v	
				Create a new snapsho	ot. You can create	a backup by clicking	any snapshot be	low and se	lectin	g 'Bao	kup'.
Snapshots and Back	ups					🗄 Tak	e Snapshot	Create	Back		
•	d12527dc a few seconds ago	Volume	Head				Show Syst	tem Hidder	: O		

Creamos la segunda entrada en el blog

$\leftarrow \ \rightarrow \ G$	○ 🏠 172.22.201.232:32580/wp-adm	in/edit.php?post_type	=post		× ☆	ତ ୬ ଛେ ≦
🛞 🏠 Wordpress-Jose	ma 📀 1 📮 0 🕂 Añadir					Hola, josema 🔲
🚳 Escritorio	Entradas Añadir una nueva entrada				Opciones de p	oantalla ▼ Ayuda ▼
🖈 Entradas 🛛 🔸	Todo (3) Publicados (3)					Buscar entradas
Todas las entradas Añadir una nueva	Acciones en lote 🗸 Aplicar Todas las	s fechas 🗸 🗸 Todas las	categorías 🗸 🛛 Filtrar			3 elementos
entrada	🗌 Título 🛱 👘 📥	Autor	Categorías	Etiquetas	P ÷	Fecha ≑
Categorías Etiquetas	Despues del snapshot	josema	Sin categoría	_	_	Publicada 17/12/2024 a las 23:57
😋 Medios 📕 Páginas	Antes del snapshot	josema	Sin categoría	_	-	Publicada 17/12/2024 a las 23:54
Comentarios	🗋 ¡Hola, mundo!	josema	Sin categoría	-	P	Publicada 17/12/2024 a las 23:53
Apariencia	☐ Título ₹	Autor	Categorías	Etiquetas	P ÷	Fecha 🔷

Con la segunda entrada creada nos disponemos a eliminar el deployment de mariadb

josema@Debian12:~/Cluster/wordpress-kubernetes\$ kubectl delete
deployments.apps mariadb-deployment

deployment.apps "mariadb-deployment" deleted

Si intentamos entrar a WordPress, nos mostrara que no tiene conexión con la base de datos.



Para restaurar la snapshot, nos dirigimos al volumen asociado a mariadb, asociamos dicho volumen en modo mantenimiento a cualquier nodo del clúster

$\leftarrow \rightarrow C$	⑦ № 172.22.201.232:30672/#/vo	☆	${igsidential}$	۲	പ	≡			
	迪 Dashboard 🖵 Node		🗍 Backup						
⊟ Volume					Custom Column	Crea	ate Vol	ume	
Delete Attach t	o host					×	G	0	
	* Host: nodo1			\vee	0		Dperati	on	
	Maintenance : 🗹						≡ ~		
					Cancel	ок			

Nos dirigimos al snapshot y seleccionamos la opción recuperar.

$\leftarrow \ \ \rightarrow \ \ \mathbf{G}$	○ 🇞 172.22.201.232:30672/#/volume/pvc-84c3f1f0-2568-4168-b5cd-63b8a8fb1e27						\$3	9	۹	பி	≡
V LONGHORN	迪 Dashboard	🖬 Node	E Volume	C Recurring Job	🕽 Backup	IS Setting ∨					
≣Volume / pvc-84c3	3f1f0-2568-4168-	b5cd-63b8	a8fb1e27							:= v	
Snapshots and Back	ups					🖯 🖯 Tak	e Snapshot	Create	Backu		
							Show Sys	tem Hidden	0		
•	df2527dc 6 minutes ago	▶ Volume	Head								
	Revert 🖢										
	Back up										
	Clone Volume										
	Delete										

2º ASIR

Con el volumen recuperado, desasociamos el volumen y volveremos a realizar el deployment de mariadb.

Volvemos a acceder a WordPress y comprobamos que solo debería de existir una entrada.

$\leftarrow \ \ \rightarrow \ \ \mathbf{G}$	⑦ № 172.22.201.232:32580/wp	ڌ	₩	ල ම එ ≡		
🚯 🏠 Wordpress-Josen	ma 🔂 1 투 0 🕂 Añadir					Hola, josema 🕅
🍘 Escritorio	Entradas Añadir una nueva entrada				Opciones de pa	antalla ▼ Ayuda ▼
🖈 Entradas 🛛 🔸	Todo (2) Publicados (2)	Ŕ				Buscar entradas
Todas las entradas Añadir una nueva	Acciones en lote 🗸 Aplicar Too	das las fechas 🗸 🔽 Todas las cat	egorías 🗸 🛛 Filtrar			2 elementos
entrada	🗌 Título 🌻	Autor	Categorías	Etiquetas	— ‡	Fecha 🗘
Categorias Etiquetas	Antes del snapshot	josema	Sin categoría	_	_	Publicada 17/12/2024 a las 23:54
9) Medios I Páginas	📄 ¡Hola, mundo!	josema	Sin categoría	-	P	Publicada 17/12/2024 a las 23:53
Comentarios	☐ Título \$	Autor	Categorías	Etiquetas	P ÷	Fecha 븆

Como vemos se ha restaurado el snapshot correctamente

5. Dificultades que se han encontrado

En este proyecto he encontrado las siguientes dificultades:

- Dificultad a la hora de instalar el servicio de métricas en el clúster, ya que tenía problemas con los certificados.
- Dificultad a la hora de realizar el auto escalado debido a que no se activa el servicio de métricas.
- Dificultad de implementar un controlador ingress para exponer los servicios.

6.Bibliografía

https://kubernetes.io/es/docs/reference/setup-tools/kubeadm/

https://longhorn.io/docs/1.7.2/

https://docs.portainer.io