HELM (Kubernetes)

Helm es un manejador de paquetes de Kubernetes. Te permite buscar, modificar e instalar servicios en tu Clúster de Kubernetes. Contiene Chars, que son un grupo de manifiesto que se instalan en tu Clúster.

Puedes tener varias copias con diferente release, que no son versiones, sino configuración que vayas a implementarle.

Para empezar, usamos:

helm init

Para buscar el servicio en el repositorio de helm usamos:

helm search (ejemplo) # Por ejemplo, si buscamos wordpress: helm search wordpress, nos saldrá la versión stable/wordpress

Por defecto no genera la release, así que habría que generarle un nombre con:

helm install stable/wordpress --generate-name

Este en concreto, crea un Load Balancer, **CUIDADO** con usarlo con Minikube, que es un Clúster Local. Para que podamos emular una IP pública desde Minikube, desde otra terminal que DEBE estar corriendo, usamos:

minikube tunnel

Si hacemos un **kubectl get all**, veremos todo lo que nos ha creado.

Al desplegarlo, el char se muestra en la salida, donde se menciona la IP pública, el usuario de Wordpress y su contraseña (en este caso concreto). Podemos exportarlas a variables de entornos y hacerle un simple **echo** para verla.

Podemos inspeccionar un paquete de Helm con:

helm inspect (nombre_paquete) # Por ejemplo, helm inspect stable/wordpress

Podemos customizar algunos de ellos y cambiar cosas como los puertos, habilitar métricas con Prometheus, la imagen que utiliza ese paqueteetc…

Para configurarlo en un yaml:

helm inspect values (nombre_paquete) > ejemplo.yaml

Ahora, para instalar el paquete personalizado:

helm install -f ejemplo.yaml (nombre_paquete) --generate-name

Crear nuestros propios Charts

Creamos un chart con:

helm create (nuestrochart)

Por defecto, esto crea un paquete de ejemplo que tiene algunas plantillas creadas de cosas útiles para cualquier chart.

Si queremos hacer una prueba (como ocurre con *terraform plan*) de lo que se desplegaría con este chart, usamos, dentro del directorio creado:

helm install --dry-run debug .

Todo esto se basa en las plantillas que ha creado por defecto. Por ejemplo podríamos editar el values.yaml, donde podríamos añadir cosas como healthcheck, readinessProbe, LivenesssProbe.

Una buena práctica, para comprobar si hay fallos de sintaxis en nuestro chart:

helm lint

Una vez tengamos nuestro paquete formado, queremos empaquetarlo de nuevo para que otras personas o equipos usen nuestro chart modificado. Para ello, empaquetamos con:

helm package (nombrechar)/ # Ponemos aquí el directorio de nuestro chart

Se genera un .tgz

Ahora debemos generar un index para saber que chart están dentro de nuestro repositorio. Para ello usamos:

helm repo index .

Esto buscará todos los .tgz de nuesta carpeta que contiene toda la información de nuestro chart.

Si lo subimos a GitHub, posteriormente podemos extraerlo desde helm con:

helm repo add (nombre) (url de nuestro repositorio de GitHub) # Puede ser cualquier nombre

Ahora podemos hacer una búsqueda con:

helm search repo (nombre_chart)

Se instala exactamente igual:

helm install helm (nombre_repo/nombre_chart)