Despliegue de aplicaciones con Helm

Despliegue de aplicaciones con Helm

Como hemos estudiado en las unidades anteriores, una aplicación real completa se compone de un conjunto amplio de objetos que definen Deployments, ConfigMaps, Services, etc. La API de Kubernetes no nos ofrece un "superobjeto" que defina una aplicación completa.

Necesitamos herramientas para gestionar la aplicación completa: empaquetado, instaladores, control de la aplicación en producción, etc. En esta unidad vamos a estudiar Helm, que es un software que nos permite empaquetar aplicaciones completas y gestionar el ciclo completo de despliegue de dicha aplicación.

Helm usa un formato de empaquetado llamado **charts**. Un chart es una colección de archivos que describen un conjunto de recursos que nos permite desplegar una aplicación en Kubernetes.

Los **charts** son distribuidos en distintos repositorios, que podremos dar de alta en nuestra instalación de Helm. Para buscar los distintos charts y los repositorios desde los que se distribuyen podemos usar la página Artifact Hub.

Instalación de Helm

Helm se distribuye como un único binario que podemos instalar de distintas formas. La última versión del programa la podemos encontrar en esta página y podemos ver los distintos métodos de instalación en la documentación oficial.

Una vez instalado podemos ver la versión de Helm que tenemos instalada:

```
helm version
```

Los siguiente es indicar un repositorio para que podamos empezar a trabajar con helm, para ello:

```
helm repo add "stable" "https://charts.helm.sh/stable" --force-update
```

Y ahora el repositorio `stable` corresponde a `charts.helm.sh/stable`:

```
helm repo list
NAME    URL
stable  https://charts.helm.sh/stable
```

Vídeo: Introducción a Helm

https://www.youtube.com/embed/tRBCdOYOfnU

Vídeo: Introducción a Helm

Gestión de charts y despliegue de aplicaciones

Como hemos visto anteriormente, por defecto, tenemos instalado un repositorio:

```
helm repo list
NAME    URL
stable  https://charts.helm.sh/stable
```

Podemos buscar más repositorios de charts buscando en la página Artifact Hub, por ejemplo podemos añadir el repositorio de charts de Bitnami de la siguiente manera:

```
helm repo add bitnami https://charts.bitnami.com/bitnami
"bitnami" has been added to your repositories
```

Y podemos comprobar que hemos añadido un nuevo repositorio:

```
helm repo list
NAME    URL
stable  https://charts.helm.sh/stable
bitnami https://charts.bitnami.com/bitnami
```

Si queremos actualizar la lista de charts ofrecidos por los repositorios:

```
helm repo update
Hang tight while we grab the latest from your chart repositories...
...Successfully got an update from the "stable" chart repository
...Successfully got an update from the "bitnami" chart repository
Update Complete. ⎈Happy Helming!⎈
```

Buscar charts

Como hemos comentado anteriormente, los charts los podemos buscar en la página Artifact Hub o los podemos buscar desde la línea de comandos, por ejemplo si queremos buscar un chart relacionado con `nginx`:

```
helm search repo nginx
NAME                                    CHART VERSION   APP VERSION     DESCRIPTION
bitnami/nginx                           9.3.0           1.21.0          Chart for t
bitnami/nginx-ingress-controller        7.6.12          0.47.0          Chart for t
```

```
stable/nginx-ingress            1.41.3          v0.34.1         DEPRECATED!
stable/nginx-ldapauth-proxy     0.1.6           1.13.5          DEPRECATED
...
```

Para obtener información sobre el chart `bitnami/nginx` podemos buscar en Artifact Hub.

Todos los ficheros yaml que forman parte de un chart están parametrizados, es decir cada propiedad tiene un valor por defecto, pero a la hora de instalarlo se puede cambiar. Por ejemplo, ¿qué tipo de Service se creará al instalar el chart `bitnami/nginx`? Por defecto, el parámetro `service.type` tiene como valor `LoadBalancer`, pero si queremos un Service de tipo `NodePort`, podremos redefinir este parámetro a la hora de instalar el chart.

¿Y cómo sabemos los parámetros que tiene definido cada chart y sus valores por defecto?. Estudiando la documentación del chart en Artifact Hub. En concreto para el chart con el que estamos trabajando, accediendo a la url https://artifacthub.io/packages/helm/bitnami/nginx. También podemos obtener esta información ejecutando el siguiente comando:

```
helm show all bitnami/nginx
```

Instalación del chart

Para instalar el chart ejecutamos la siguiente instrucción:

```
helm install serverweb bitnami/nginx --set service.type=NodePort
```

Como vemos hemos nombrado el chart desplegado (`serverweb`), indicado el chart (`bitnami/nginx`) y, en este caso, hemos redefinido el parámetro `service.type`.

Cuando se despliega el chart se nos ofrece información que nos muestra cómo acceder a la aplicación:

```
NOTES:
** Please be patient while the chart is being deployed **

NGINX can be accessed through the following DNS name from within your cluster:

    serverweb-nginx.default.svc.cluster.local (port 80)

To access NGINX from outside the cluster, follow the steps below:

1. Get the NGINX URL by running these commands:

    export NODE_PORT=$(kubectl get --namespace default -o jsonpath="{.spec.ports[0]
```

```
export NODE_IP=$(kubectl get nodes --namespace default -o jsonpath="{.items[0].
echo "http://${NODE_IP}:${NODE_PORT}"
```

Si queremos acceder a la aplicación desde el exterior debemos ejecutar las tres últimas instrucciones, que nos muestran la ip de nuestro cluster y el puerto asignado al Service NodePort.

Siempre podemos volver a ver esta información ejecutando la siguiente instrucción:

```
helm status serverweb
```

Podemos comprobar los Deployments que hemos realizado con Helm, ejecutando:

```
helm ls
NAME            NAMESPACE       REVISION        UPDATED
serverweb       default         1               2021-06-29 19:11:15.975016119 +020(
```

Y podemos comprobar también los recursos que se han creado en el cluster:

```
kubectl get all
NAME                                    READY   STATUS    RESTARTS   AGE
pod/serverweb-nginx-7b7f75d476-kxq8j    1/1     Running   0          56s

NAME                        TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)        A
service/kubernetes          ClusterIP   10.96.0.1      <none>        443/TCP        5
service/serverweb-nginx     NodePort    10.99.80.141   <none>        80:30137/TCP   5

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/serverweb-nginx     1/1     1            1           56s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/serverweb-nginx-7b7f75d476     1         1         1       56s
```

Por último, para desinstalar una aplicación completa, ejecutamos:

```
helm delete serverweb
release "serverweb" uninstalled
```

Aunque no entra en el ámbito de este curso, hay que indicar que si nosotros desarrollamos una aplicación podemos empaquetarla para instalarla con Helm, creando nuestros propios charts. Para más información puedes entrar en la documentación oficial.

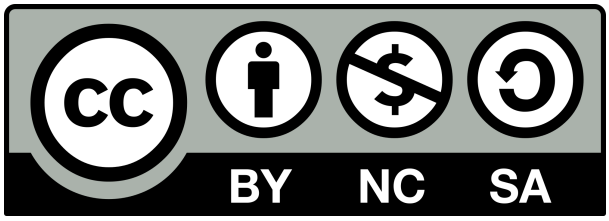
Vídeo: Gestión de charts y despliegue de aplicaciones

https://www.youtube.com/embed/UlkXAFHvrkw

Vídeo: Gestión de charts y despliegue de aplicaciones

Créditos

Materiales desarrollados por:
Alberto Molina Coballes
José Domingo Muñoz Rodríguez
Propiedad de la Consejería de Educación y Deporte de la Junta de Andalucía



2021