'helm search': Buscando Charts

Helm viene con un poderoso comando de búsqueda. Se puede utilizar para buscar dos tipos diferentes de fuentes:

- helm search hub buscar en Artifact Hub, que enumera charts de Helm de docenas de repositorios diferentes.
- helm search repo busca en los repositorios que ha agregado a su cliente de helm local (con helm repo add). Esta búsqueda se realiza a través de datos locales y no se necesita una conexión de red pública.

Puede encontrar charts disponibles públicamente ejecutando helm search hub:

```
$ helm search hub wordpress
URL                                               CHART VERSION APP
VERSION DESCRIPTION
https://hub.helm.sh/charts/bitnami/wordpress      7.6.7         5.2.4
Web publishing platform for building blogs and ...
https://hub.helm.sh/charts/presslabs/wordpress-...  v0.6.3        v0.6.3
Presslabs WordPress Operator Helm Chart
https://hub.helm.sh/charts/presslabs/wordpress-...  v0.7.1        v0.7.1
A Helm chart for deploying a WordPress site on ...
```

Lo anterior busca todos los charts de wordpress en Artifact Hub.

Sin filtro, helm search hub muestra todos los charts disponibles.

Usando helm search repo, puede encontrar los nombres de los charts en los repositorios que ya has agregado:

```
$ helm repo add brigade https://brigadecore.github.io/charts
"brigade" has been added to your repositories
$ helm search repo brigade
NAME                          CHART VERSION APP VERSION DESCRIPTION
brigade/brigade               1.3.2         v1.2.1      Brigade provides
event-driven scripting of Kube...
brigade/brigade-github-app    0.4.1         v0.2.1      The Brigade GitHub
App, an advanced gateway for...
brigade/brigade-github-oauth  0.2.0         v0.20.0     The legacy OAuth
GitHub Gateway for Brigade
brigade/brigade-k8s-gateway   0.1.0                     A Helm chart for
Kubernetes
brigade/brigade-project       1.0.0         v1.0.0      Create a Brigade
project
brigade/kashti                0.4.0         v0.4.0      A Helm chart for
Kubernetes
```

Helm search utiliza un algoritmo de coincidencia de cadenas difusas, por lo que puede escribir partes de palabras o frases:

```
$ helm search repo kash
NAME            CHART VERSION APP VERSION DESCRIPTION
brigade/kashti  0.4.0         v0.4.0      A Helm chart for Kubernetes
```

El comando search es una buena forma de encontrar paquetes disponibles. Una vez que haya encontrado el paquete que desea instalar, puede usar `helm install` para instalarlo.

'helm install': Instalando un Package

Para instalar un nuevo paquete, use el comando `helm install`. En su forma más simple, se necesitan dos argumentos: un nombre de release que elijas y el nombre del chart que deseas instalar.

```
$ helm install happy-panda bitnami/wordpress
NAME: happy-panda
LAST DEPLOYED: Tue Jan 26 10:27:17 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
** Please be patient while the chart is being deployed **
Your WordPress site can be accessed through the following DNS name from
within your cluster:
   happy-panda-wordpress.default.svc.cluster.local (port 80)
To access your WordPress site from outside the cluster follow the steps
below:
1. Get the WordPress URL by running these commands:
 NOTE: It may take a few minutes for the LoadBalancer IP to be available.
       Watch the status with: 'kubectl get svc --namespace default -w
happy-panda-wordpress'
  export SERVICE_IP=$(kubectl get svc --namespace default
happy-panda-wordpress --template "{{ range (index
.status.loadBalancer.ingress 0) }}{{.}}{{ end }}")
  echo "WordPress URL: http://$SERVICE_IP/"
  echo "WordPress Admin URL: http://$SERVICE_IP/admin"
2. Open a browser and access WordPress using the obtained URL.
3. Login with the following credentials below to see your blog:
 echo Username: user
 echo Password: $(kubectl get secret --namespace default
happy-panda-wordpress -o jsonpath="{.data.wordpress-password}" | base64
--decode)
```

Ahora el chart wordpress está instalado. Tenga en cuenta que la instalación de un chart crea un nuevo objeto *release*. El release anterior se llama happy-panda. (Si desea que Helm genere un nombre por usted, omita el nombre del release y use --generate-name).

Durante la instalación, el cliente helm imprimirá información útil sobre qué recursos se crearon, cuál es el estado del release y también si hay pasos de configuración adicionales que puede o debe tomar.

Helm no espera hasta que todos los recursos se estén ejecutando antes de salir. Muchos charts requieren imágenes de Docker que tienen un tamaño superior a 600M y pueden tardar mucho en instalarse en el clúster.

Para realizar un seguimiento del estado de un release o para volver a leer la información de configuración, puede utilizar helm status:

```
$ helm status happy-panda
NAME: happy-panda
LAST DEPLOYED: Tue Jan 26 10:27:17 2021
NAMESPACE: default
STATUS: deployed
REVISION: 1
NOTES:
** Please be patient while the chart is being deployed **
Your WordPress site can be accessed through the following DNS name from
within your cluster:
   happy-panda-wordpress.default.svc.cluster.local (port 80)
To access your WordPress site from outside the cluster follow the steps
below:
1. Get the WordPress URL by running these commands:
 NOTE: It may take a few minutes for the LoadBalancer IP to be available.
       Watch the status with: 'kubectl get svc --namespace default -w
happy-panda-wordpress'
  export SERVICE_IP=$(kubectl get svc --namespace default
happy-panda-wordpress --template "{{ range (index
.status.loadBalancer.ingress 0) }}{{.}}{{ end }}")
  echo "WordPress URL: http://$SERVICE_IP/"
  echo "WordPress Admin URL: http://$SERVICE_IP/admin"
2. Open a browser and access WordPress using the obtained URL.
3. Login with the following credentials below to see your blog:
 echo Username: user
 echo Password: $(kubectl get secret --namespace default
happy-panda-wordpress -o jsonpath="{.data.wordpress-password}" | base64
--decode)
```

Lo anterior muestra el estado actual de su release.

Personalización del Charts antes de la instalación

La instalación de la forma que tenemos aquí solo usará las opciones de configuración predeterminadas para este chart. Muchas veces, querrás personalizar el chart para usar tu configuración preferida.

Para ver qué opciones se pueden configurar en un chart, use helm show values:

```
$ helm show values bitnami/wordpress
## Global Docker image parameters
## Please, note that this will override the image parameters, including
dependencies, configured to use the global value
## Current available global Docker image parameters: imageRegistry and
imagePullSecrets
##
# global:
#   imageRegistry: myRegistryName
#   imagePullSecrets:
#     - myRegistryKeySecretName
#   storageClass: myStorageClass
## Bitnami WordPress image version
## ref: https://hub.docker.com/r/bitnami/wordpress/tags/
##
image:
 registry: docker.io
 repository: bitnami/wordpress
 tag: 5.6.0-debian-10-r35
 [..]
```

Luego, puedes sobreescribir cualquiera de estas configuraciones en un archivo con formato YAML y luego pasar ese archivo durante la instalación.

```
$ echo '{mariadb.auth.database: user0db, mariadb.auth.username: user0}' >
values.yaml
$ helm install -f values.yaml bitnami/wordpress --generate-name
```

Lo anterior creará un usuario MariaDB predeterminado con el nombre `user0`, y otorgará a este usuario acceso a una base de datos `user0db` recién creada, pero aceptará el resto de los valores predeterminados para ese chart.

Hay dos formas de pasar los datos de configuración durante la instalación:

- `--values` (o `-f`): Especifique un archivo YAML con sobreescrituras. Esto se puede especificar varias veces y el archivo más a la derecha tendrá prioridad
- `--set`: Especifique sobreescrituras en la línea de comando.

Si se utilizan ambos, los valores pasados en --set se fusionan con los pasados en --values con mayor precedencia. Las sobreescrituras especificadas con --set se guardan en un ConfigMap. Los valores que han sido pasados con --set se pueden ver para un release determinado con helm get values <release-name>. Los valores que han sido pasados con --set se pueden borrar ejecutando helm upgrade especificando --reset-values.

El Formato y Limitaciones de --set

La opción --set toma cero o más pares de nombre/valor. En su forma más simple, se usa así: --set name=value. El equivalente YAML de eso es:

```yaml
name: value
```

Múltiples valores son separados por el caracteres ,. Entonces, --set a=b,c=d se convierte en:

```yaml
a: b
c: d
```

Expresiones más complejas son soportadas. Por ejemplo, --set outer.inner=value es traducida a esto:

```yaml
outer:
  inner: value
```

Las lista pueden ser expresadas Las listas se pueden expresar encerrando valores en { y }. Por ejemplo, --set name={a, b, c} se traduce a:

```yaml
name:
  - a
  - b
  - c
```

A partir de Helm 2.5.0, es posible acceder a los elementos de la lista utilizando una sintaxis de índice de arreglo. Por ejemplo, --set servers[0].port=80 se convierte en:

```yaml
servers:
  - port: 80
```

De esta forma se pueden configurar varios valores. La línea `--set servers[0].port=80,servers[0].host=example` se vuelve:

```yaml
servers:
 - port: 80
   host: example
```

A veces es necesario utilizar caracteres especiales en sus líneas `--set`. Puede usar una barra invertida para escapar de los caracteres; `--set name=value1\,value2` se convertirá en:

```yaml
name: "value1,value2"
```

De manera similar, también puede escapar de las secuencias de puntos, lo que puede resultar útil cuando los chats usan la función `toYaml` para analizar anotaciones, etiquetas y selectores de nodos. La sintaxis para `--set nodeSelector."kubernetes\.io/role"=master` se convertirá en:

```yaml
nodeSelector:
 kubernetes.io/role: master
```

Las estructuras de datos profundamente anidadas pueden ser difíciles de expresar usando `--set`. Se alienta a los diseñadores de charts a considerar el uso de `--set` al diseñar el formato de un archivo `values.yaml` (lea más sobre Archivos Values).

Más Métodos de Instalación

El comando `helm install` puede instalar desde varias fuentes:

- Un repositorio de charts (como hemos visto anteriormente)
- Un archivo de chart local (`helm install foo foo-0.1.1.tgz`)
- Un directorio de chart descomprimido (`helm install foo path/to/foo`)
- Una URL completa (`helm install foo https://example.com/charts/foo-1.2.3.tgz`)

'helm upgrade' y 'helm rollback': Actualizar un Release y Revertir un Fallo

Cuando se lanza una nueva versión de un chart, o cuando desea cambiar la configuración de su release, puede usar el comando helm upgrade.

Una actualización toma un release existente y lo actualiza de acuerdo con la información que proporcione. Dado que los charts de Kubernetes pueden ser grandes y complejos, Helm intenta realizar la actualización menos invasiva. Solo actualizará las cosas que hayan cambiado desde el último release.

```
$ helm upgrade -f panda.yaml happy-panda bitnami/wordpress
```

En el caso anterior, la versión happy-panda se actualiza con el mismo chart, pero con un nuevo archivo YAML:

```
mariadb.auth.username: user1
```

Podemos usar helm get values para ver si esa nueva configuración entró en vigencia.

```
$ helm get values happy-panda
mariadb:
 auth:
   username: user1
```

El comando helm get es una herramienta útil para ver un release en el clúster. Y como podemos ver arriba, muestra que nuestros nuevos valores de panda.yaml se desplegaron en el clúster.

Ahora, si algo no sale según lo planeado durante un release, es fácil retroceder a un release anterior usando helm rollback [RELEASE] [REVISION].

```
$ helm rollback happy-panda 1
```

Lo anterior hace retroceder a nuestro release happy-panda a su primera versión de release. La versión de release es una revisión incremental. Cada vez que ocurre una instalación, actualización o reversión, el número de revisión se incrementa en 1. El primer número de revisión es siempre 1. Y podemos usar `helm history [RELEASE]` para ver los números de revisión de un determinado release.

Opciones Útiles para Instalar/Actualizar/Revertir

Hay varias otras opciones útiles que puede especificar para personalizar el comportamiento de Helm durante una instalación/actualización/reversión. Tenga en cuenta que esta no es una lista completa de banderas del cliente. Para ver una descripción de todos las banderas, simplemente ejecute `helm <comando> --help`.

- `--timeout`: Un valor de duración de Go para esperar a que se completen los comandos de Kubernetes. El valor por defecto es `5m0s`.
- `--wait`: Espera hasta que todos los pods estén listos, los PVC están vinculados, los Depolyments tengan el mínimo (`Desired` menos `maxUnavailable`) Pods en estado listo y los Services tengan una dirección IP (e Ingress si es un `LoadBalancer`) antes de marcar el release como exitoso. Esperará tanto tiempo como el valor `--timeout`. Si se alcanza el tiempo de espera, el release se marcará como `FAILED` (Fallida). Nota: En escenarios donde el Deployment tiene `replicas` configuradas en 1 y `maxUnavailable` no está configurado en 0 como parte de la estrategia de actualización continua, `--wait` regresará tan pronto como haya satisfecho la condición de mínimo de Pods listos.
- `--no-hooks`: Esto omite la ejecución de ganchos para el comando
- `--recreate-pods` (sólo disponible para `upgrade` y `rollback`): Esta bandera hará que se vuelvan a crear todos los Pods (con la excepción de los Pods que pertenecen a Deployments). (DEPRECADO en Helm 3)

'helm uninstall': Desinstalar un Release

Cuando sea el momento de desinstalar un release del clúster, use el comando helm uninstall:

```
$ helm uninstall happy-panda
```

Esto eliminará el release del clúster. Puedes ver todas tus releases implementados actualmente con el comando helm list:

```
$ helm list
NAME            VERSION UPDATED                         STATUS
CHART
inky-cat        1       Wed Sep 28 12:59:46 2016        DEPLOYED
alpine-0.1.0
```

En el resultado anterior, podemos ver que se desinstaló el release happy-panda.

En versiones anteriores de Helm, cuando se eliminaba un release, quedaba un registro de su eliminación. En Helm 3, la eliminación también elimina el registro del release. Si desea mantener un registro de eliminación del release, use helm uninstall --keep-history. El uso de helm list --uninstalled solo mostrará los releases que se desinstalaron con la bandera --keep-history.

La bandera helm list --all le mostrará todos los registros de releases que Helm ha retenido, incluidos los registros de elementos fallidos o eliminados (si se especificó --keep-history):

```
$  helm list --all
NAME            VERSION UPDATED                         STATUS
CHART
happy-panda     2       Wed Sep 28 12:47:54 2016        UNINSTALLED
wordpress-10.4.5.6.0
inky-cat        1       Wed Sep 28 12:59:46 2016        DEPLOYED
alpine-0.1.0
kindred-angelf  2       Tue Sep 27 16:16:10 2016        UNINSTALLED
alpine-0.1.0
```

Tenga en cuenta que debido a que los releases ahora se eliminan de forma predeterminada, ya no es posible revertir un recurso desinstalado.

'helm repo': Trabajar con Repositorios

Helm 3 ya no viene con un repositorio de charts predeterminado. El grupo de comandos helm repo proporciona comandos para agregar, enumerar y eliminar repositorios.

Puede ver qué repositorios están configurados usando helm repo list:

```
$ helm repo list
NAME            URL
stable          https://charts.helm.sh/stable
mumoshu         https://mumoshu.github.io/charts
```

Y se pueden agregar nuevos repositorios con helm repo add:

```
$ helm repo add dev https://example.com/dev-charts
```

Debido a que los repositorios de charts cambian con frecuencia, en cualquier momento puede asegurarse de que su cliente Helm esté actualizado ejecutando helm repo update.

Los repositorios se pueden eliminar con helm repo remove.

Creación de sus Propios Charts

La Guía de Desarrollo de Chart explica cómo desarrollar sus propios charts. Pero puede comenzar rápidamente usando el comando helm create:

```
$ helm create deis-workflow
Creating deis-workflow
```

Ahora hay un chart en ./deis-workflow. Puede editarlo y crear sus propias plantillas.

A medida que editas tu chart, puede validar que está bien formado ejecutando helm lint.

Cuando llegue el momento de empaquetar el chart para su distribución, puedes ejecutar el comando `helm package`:

```
$ helm package deis-workflow
deis-workflow-0.1.0.tgz
```

Y este chart ahora se puede instalar fácilmente con `helm install`:

```
$ helm install deis-workflow ./deis-workflow-0.1.0.tgz
...
```

Los Charts empaquetados se pueden cargar en repositorios de charts. Consulte la documentación de los Repositorios de Charts de Helm para obtener más detalles.