ToiletPaper #102

Helm - Der Package Manager für Kubernetes

Von Dr. Torsten Ackemann, Senior Systems Architect, Competence Center Platforms & Operations

✖ Problem

Wie installiere ich Standard-Applikationen in Kubernetes? Wie konfiguriere ich diese? Wer kümmert sich um die Abhängigkeiten und um Upgrades?
Und: Wie bringe ich meine eigene Applikation in ein Kubernetes? Mit docker-compose ging das so einfach!

✔ Lösung

Mit Helm existiert ein Package-Manager, der es erlaubt, komplexe Applikationen wie etwa Wordpress, Hadoop oder Gitlab mit einem Kommando in ein Kubernetes zu installieren, inklusive Abhängigkeiten wie etwa einer Datenbank.
Im Prinzip verhält sich Helm zu Kubernetes genauso wie yum oder aptitude zu Linux. Ein Paket kann aber mehrfach installiert werden.
Die Pakete werden Helm Charts genannt und bestehen aus den Kubernetes-Ressourcen als Templates im YAML-Format, einer Konfigurationsdatei values.yaml mit den Default-Settings des Charts, einer Liste der Abhängigkeiten zu anderen Helm Charts und noch ein paar Metadaten wie Name und Version.
Weitere Konfiguration kann in Form von weiteren YAMLs beim Aufruf an Helm übergeben werden - so werden die Defaults überschrieben.
Aus den Templates und den Default-Werten sowie aus der übergebenen Konfiguration werden dann die konkreten Ressourcen-Beschreibungen erzeugt und an Kubernetes geschickt. Kubernetes erzeugt diese Ressourcen dann: Docker-Container, Storage-Volumes usw.
Mit einem eigenen Helm-Chart für die selbst geschriebene Applikation kann man ihre genauen Laufzeitbedingungen in Code gießen – wie viele Instanzen sollen laufen, wie sind die Speicheranforderungen, welche anderen Container sollen noch parallel als Sidecars laufen, welche wie konfigurierten Datenbanken werden benötigt, welche Endpoints sollen von außen erreichbar sein, soll die Applikation dauerhaft oder nur täglich um 4 Uhr morgens laufen, wann lebt die Applikation noch und wann ist sie bereit, Traffic entgegenzunehmen, usw. – und das alles natürlich konfigurierbar.
Die Deployments in die Dev-Umgebung auf dem Laptop und in die Prod-Umgebung in der Cloud unterscheiden sich vielleicht nur noch in der Konfigurations-YAML.

➔ Beispiel

Folgende Kommandos starten ein Minikube (Kubernetes in einer VM) auf meinem Laptop, installieren ein komplettes Wordpress mit extra viel Speicher samt Datenbank, Storage Volume und Load Balancer, und zeigen die Wordpress-Seite in meinem Browser an:

```
$ minikube start
Starting local Kubernetes v1.10.0 cluster...
$ helm init
Adding stable repo with URL: https://kubernetes-charts.storage.googleapis.com
$ cat my-values.yaml
resources:
  requests:
    memory: 1024Mi
$ helm install -f my-values.yaml stable/wordpress
==> v1/Service
NAME                          TYPE           CLUSTER-IP     EXTERNAL-IP   PORT(S)                      AGE
inclined-woodpecker-mariadb   ClusterIP      10.110.172.10  <none>        3306/TCP                     0s
inclined-woodpecker-wordpress LoadBalancer   10.108.66.217  <pending>     80:30095/TCP,443:31876/TCP   0s
$ minikube service inclined-woodpecker-wordpress
Opening kubernetes service default/inclined-woodpecker-wordpress in default browser...
```

inclined-woodpecker ist hier ein automatisch generierter Name, sodass man ohne Namens-Clash auch noch ein zweites Wordpress installieren könnte.

+ Weiterführende Aspekte

- www.helm.sh – Die offizielle Dokumentation
- github.com/helm/charts – Die offiziellen Helm Charts z.B. für Wordpress
- coreos.com/operators – Kubernetes Operators: Wem Helm Charts einfach zu statisch sind