Referenzen

Dieser Abschnitt der Kubernetes-Dokumentation enthält Referenzinformationen.

API-Referenz

- Kubernetes API Überblick - Übersicht über die API für Kubernetes.
- Kubernetes API Versionen
  - 1.17
  - 1.16
  - 1.15
  - 1.14
  - 1.13

API-Clientbibliotheken

Um die Kubernetes-API aus einer Programmiersprache aufzurufen, können Sie Clientbibliotheken verwenden. Offiziell unterstützte Clientbibliotheken:

- Kubernetes Go Clientbibliothek
- Kubernetes Python Clientbibliothek
- Kubernetes Java Clientbibliothek
- Kubernetes JavaScript Clientbibliothek

CLI-Referenz

- kubectl - Haupt-CLI-Tool zum Ausführen von Befehlen und zum Verwalten von Kubernetes-Clustern.
  - JSONPath - Syntax-Anleitung zur Verwendung von JSONPath expressionen mit kubectl.
- kubeadm - CLI-Tool zur einfachen Bereitstellung eines sicheren Kubernetes-Clusters.
- kubefed - CLI-Tool zur Verwaltung Ihres Clusterverbunds.

Konfigurationsreferenz

- kubelet - Der primäre *Node-Agent*, der auf jedem Node ausgeführt wird. Das Kubelet betrachtet eine Reihe von PodSpecs und stellt sicher, dass die beschriebenen Container ordnungsgemäß ausgeführt werden.
- kube-apiserver - REST-API zur Überprüfung und Konfiguration von Daten für API-Objekte wie Pods, Services und Replikationscontroller.
- kube-controller-manager - Daemon, der die mit Kubernetes gelieferten zentralen Regelkreise einbettet.
- kube-proxy - Kann einfache TCP/UDP-Stream-Weiterleitung oder Round-Robin-TCP/UDP-Weiterleitung über eine Reihe von Back-Ends durchführen.
- kube-scheduler - Scheduler, der Verfügbarkeit, Leistung und Kapazität verwaltet.
- federation-apiserver - API-Server für Cluster Föderationen.

- federation-controller-manager - Daemon, der die zentralen Regelkreise einbindet, die mit der Kubernetes-Föderation ausgeliefert werden.

Design Dokumentation

Ein Archiv der Designdokumente für Kubernetes-Funktionalität. Gute Ansatzpunkte sind Kubernetes Architektur und Kubernetes Design Übersicht.

1 - Standardisiertes Glossar

2 - Kubernetes Probleme und Sicherheit

3 - Verwendung der Kubernetes-API

4 - API Zugriff

5 - API Referenzinformationen

6 - Föderation API

7 - Setup-Tools Referenzinformationen

8 - Befehlszeilen-Werkzeug Referenzinformationen

9 - kubectl CLI

9.1 - kubectl Spickzettel

Siehe auch: Kubectl Überblick und JsonPath Dokumentation.

Diese Seite ist eine Übersicht über den Befehl `kubectl`.

kubectl - Spickzettel

Kubectl Autovervollständigung

BASH

```
source <(kubectl completion bash) # Wenn Sie autocomplete in bash in der aktuelle
echo "source <(kubectl completion bash)" >> ~/.bashrc # Fügen Sie der Bash-Shell
```

Sie können auch ein Abkürzungsalias für `kubectl` verwenden, welches auch mit Vervollständigung funktioniert:

```
alias k=kubectl
complete -F __start_kubectl k
```

ZSH

```
source <(kubectl completion zsh)  # Richten Sie Autocomplete in zsh in der aktuel
echo "if [ $commands[kubectl] ]; then source <(kubectl completion zsh); fi" >> ~/
```

Kubectl Kontext und Konfiguration

Legen Sie fest, welcher Kubernetes-Cluster mit `kubectl` kommuniziert und dessen Konfiguration ändert. Lesen Sie die Dokumentation Authentifizierung mit kubeconfig über mehrere Cluster hinweg für ausführliche Informationen zur Konfigurationsdatei.

```
kubectl config view # Zusammengeführte kubeconfig-Einstellungen anzeigen.

# Verwenden Sie mehrere kubeconfig-Dateien gleichzeitig und zeigen Sie die zusamm
KUBECONFIG=~/.kube/config:~/.kube/kubconfig2 kubectl config view

# Zeigen Sie das Passwort für den e2e-Benutzer an
kubectl config view -o jsonpath='{.users[?(@.name == "e2e")].user.password}'

kubectl config view -o jsonpath='{.users[].name}'    # den ersten Benutzer anzeig
kubectl config view -o jsonpath='{.users[*].name}'    # eine Liste der Benutzer e
kubectl config current-context                       # den aktuellen Kontext anze
kubectl config use-context my-cluster-name           # Setzen Sie den Standardkon

# Fügen Sie Ihrer kubeconf einen neuen Cluster hinzu, der basic auth unterstützt
kubectl config set-credentials kubeuser/foo.kubernetes.com --username=kubeuser --

# Legen Sie einen Kontext fest, indem Sie einen bestimmten Benutzernamen und eine
kubectl config set-context gce --user=cluster-admin --namespace=foo \
  && kubectl config use-context gce
```

```
kubectl config unset users.foo                    # delete user foo
```

Apply

`apply` verwaltet Anwendungen durch Dateien, die Kubernetes-Ressourcen definieren. Es erstellt und aktualisiert Ressourcen in einem Cluster durch Ausführen von `kubectl apply`. Dies ist die empfohlene Methode zur Verwaltung von Kubernetes-Anwendungen in der Produktion. Lesen Sie die ausführliche Kubectl Dokumentation für weitere Informationen.

Objekte erstellen

Kubernetes Manifeste können in Json oder Yaml definiert werden. Die Dateierweiterungen `.yaml`, `.yml`, und `.json` können verwendet werden.

```
kubectl apply -f ./my-manifest.yaml            # Ressource(n) erstellen
kubectl apply -f ./my1.yaml -f ./my2.yaml      # aus mehreren Dateien erstellen
kubectl apply -f ./dir                         # Erstellen Sie Ressourcen in alle
kubectl apply -f https://git.io/vPieo          # Ressource(n) aus URL erstellen
kubectl create deployment nginx --image=nginx  # Starten Sie eine einzelne Instan
kubectl explain pods,svc                       # Zeigen Sie die Dokumentation für

# Erstellen Sie mehrere YAML-Objekte aus stdin
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Pod
metadata:
  name: busybox-sleep
spec:
  containers:
  - name: busybox
    image: busybox
    args:
    - sleep
    - "1000000"
---
apiVersion: v1
kind: Pod
metadata:
  name: busybox-sleep-less
spec:
  containers:
  - name: busybox
    image: busybox
    args:
    - sleep
    - "1000"
EOF

# Erstellen Sie ein "Secret" mit mehreren Schlüsseln
cat <<EOF | kubectl apply -f -
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: $(echo -n "s33msi4" | base64 -w0)
  username: $(echo -n "jane" | base64 -w0)
EOF
```

Suchen und Anzeigen von Ressourcen

```
# Get Befehle mit grundlegenden Ausgaben
kubectl get services                          # Listen Sie alle Dienste im Namesp
kubectl get pods --all-namespaces             # Listen Sie alle Pods in allen Nam
kubectl get pods -o wide                      # Listen Sie alle Pods im Namespace
kubectl get deployment my-dep                 # Listen Sie eine bestimmte Bereits
kubectl get pods                              # Listen Sie alle Pods im Namespace

# Describe Befehle mit ausführlicher Ausgabe
kubectl describe nodes my-node
kubectl describe pods my-pod

kubectl get services --sort-by=.metadata.name # Listen Sie Dienste nach Namen sor

# Listen Sie Pods Sortiert nach Restart Count auf
kubectl get pods --sort-by='.status.containerStatuses[0].restartCount'

# Erhalten Sie die Versionsbezeichnung aller Pods mit der Bezeichnung app=cassand
kubectl get pods --selector=app=cassandra -o \
  jsonpath='{.items[*].metadata.labels.version}'

# Alle Worker-Knoten abrufen (verwenden Sie einen Selektor, um Ergebnisse auszusc
# die ein Label mit dem Namen 'node-role.kubernetes.io/master' tragen).
kubectl get node --selector='!node-role.kubernetes.io/master'

# Zeigen Sie alle laufenden Pods im Namespace an
kubectl get pods --field-selector=status.phase=Running

# Rufen Sie die externe IP aller Nodes ab
kubectl get nodes -o jsonpath='{.items[*].status.addresses[?(@.type=="ExternalIP"

# Listet die Namen der Pods auf, die zu einem bestimmten RC gehören
# Der Befehl "jq" ist nützlich für Transformationen, die für jsonpath zu komplex
sel=${$(kubectl get rc my-rc --output=json | jq -j '.spec.selector | to_entries |
echo $(kubectl get pods --selector=$sel --output=jsonpath={.items..metadata.name}

# Labels für alle Pods anzeigen (oder jedes andere Kubernetes-Objekt, das labelli
# Verwendet auch "jq"
for item in $( kubectl get pod --output=name); do printf "Labels for %s\n" "$item

# Prüfen Sie, welche Nodes bereit sind
JSONPATH='{range .items[*]}{@.metadata.name}:{range @.status.conditions[*]}{@.typ
 && kubectl get nodes -o jsonpath="$JSONPATH" | grep "Ready=True"

# Listen Sie alle Secrets auf, die derzeit von einem Pod verwendet werden
kubectl get pods -o json | jq '.items[].spec.containers[].env[]?.valueFrom.secret

# Ereignisse nach Zeitstempel sortiert auflisten
kubectl get events --sort-by=.metadata.creationTimestamp
```

Ressourcen aktualisieren

Ab Version 1.11 ist das `rolling-update` veraltet (Lesen Sie CHANGELOG-1.11.md für weitere Informationen), verwenden Sie stattdessen `rollout` .

```
kubectl set image deployment/frontend www=image:v2               # Fortlaufende A
kubectl rollout undo deployment/frontend                         # Rollback zur v
kubectl rollout status -w deployment/frontend                    # Beobachten Sie

# veraltet ab Version 1.11
kubectl rolling-update frontend-v1 -f frontend-v2.json           # (veraltet) For
kubectl rolling-update frontend-v1 frontend-v2 --image=image:v2  # (veraltet) Änd
kubectl rolling-update frontend --image=image:v2                 # (veraltet) Akt
```

```
kubectl rolling-update frontend-v1 frontend-v2 --rollback        # (veraltet) Bri

cat pod.json | kubectl replace -f -                              # Ersetzen Sie e

# Ersetzen, löschen und Ressource neu erstellen. Dies führt zu einer temprären Un
kubectl replace --force -f ./pod.json

# Erstellen Sie einen Dienst für eien replizierten nginx Webserver, der an Port 8
kubectl expose rc nginx --port=80 --target-port=8000

# Aktualisieren Sie die Image-Version (Tag) eines Einzelcontainer-Pods auf Versio
kubectl get pod mypod -o yaml | sed 's/\(image: myimage\):.*$/\1:v4/' | kubectl r

kubectl label pods my-pod new-label=awesome                      # Label hinzufüg
kubectl annotate pods my-pod icon-url=http://goo.gl/XXBTWq       # Eine Anmerkung
kubectl autoscale deployment foo --min=2 --max=10                # Automatische S
```

Ressourcen patchen

```
kubectl patch node k8s-node-1 -p '{"spec":{"unschedulable":true}}' # Aktualisiere

# Aktualisieren Sie das Image eines Containers; spec.containers[*].name ist erfor
kubectl patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-serve-

# Aktualisieren Sie das Image eines Containers mithilfe eines Json-Patches mit Po
kubectl patch pod valid-pod --type='json' -p='[{"op": "replace", "path": "/spec/c

# Deaktivieren Sie eine Bereitstellung von livenessProbe durch verwenden eines Js
kubectl patch deployment valid-deployment  --type json   -p='[{"op": "remove", "p

# Fügen Sie einem Positionsarray ein neues Element hinzu
kubectl patch sa default --type='json' -p='[{"op": "add", "path": "/secrets/1", "
```

Ressourcen bearbeiten

Bearbeiten Sie eine beliebige API-Ressource in einem Editor.

```
kubectl edit svc/docker-registry                      # Bearbeiten Sie den Dienst
KUBE_EDITOR="nano" kubectl edit svc/docker-registry   # Verwenden Sie einen alter
```

Ressourcen skalieren

```
kubectl scale --replicas=3 rs/foo                                 # Skaliert ein
kubectl scale --replicas=3 -f foo.yaml                            # Skaliert eine
kubectl scale --current-replicas=2 --replicas=3 deployment/mysql  # Wenn die aktu
kubectl scale --replicas=5 rc/foo rc/bar rc/baz                   # Skaliert mehr
```

Ressourcen löschen

```
kubectl delete -f ./pod.json                                      # Lösch
kubectl delete pod,service baz foo                                # Lösch
```

```
kubectl delete pods,services -l name=myLabel                    # Lösch
kubectl -n my-ns delete po,svc --all                            # Lösch
```

Interaktion mit laufenden Pods

```
kubectl logs my-pod                                 # Pod-Logdatei ausgeben (stdo
kubectl logs my-pod --previous                      # Pod-Logdatei für eine vorhe
kubectl logs my-pod -c my-container                 # Pod Container-Logdatei ausg
kubectl logs my-pod -c my-container --previous      # Pod Container-Logdatei für
kubectl logs -f my-pod                              # Pod-Logdatei streamen (stdo
kubectl logs -f my-pod -c my-container              # Pod Container-Logdatei stre
kubectl run -i --tty busybox --image=busybox -- sh  # Pod als interaktive Shell a
kubectl attach my-pod -i                            # An laufenden Container anhä
kubectl port-forward my-pod 5000:6000               # Lauscht auf Port 5000 auf d
kubectl exec my-pod -- ls /                         # Befehl in vorhandenem Pod a
kubectl exec my-pod -c my-container -- ls /         # Befehl in vorhandenem Pod a
kubectl top pod POD_NAME --containers               # Zeigt Metriken für einen be
```

Mit Nodes und Clustern interagieren

```
kubectl cordon my-node                                                # Markiert
kubectl drain my-node                                                 # Entleert
kubectl uncordon my-node                                              # Markiert
kubectl top node my-node                                              # Metriken
kubectl cluster-info                                                  # Adressen
kubectl cluster-info dump                                             # Ausgabe d
kubectl cluster-info dump --output-directory=/pfad/zum/cluster-status # Aktuellen

# Wenn bereits ein Taint mit diesem Key und Effekt vorhanden ist, wird sein Wert
kubectl taint nodes foo dedicated=special-user:NoSchedule
```

Ressourcentypen

Liste aller unterstützten Ressourcentypen mit ihren Kurzbezeichnungen, der API-Gruppe, unabhängig davon ob sie im Namespace liegen, und der Art:

```
kubectl api-resources
```

Andere Operationen zum Erkunden von API-Ressourcen:

```
kubectl api-resources --namespaced=true      # Alle Ressourcen im Namespace
kubectl api-resources --namespaced=false     # Alle nicht im Namespace befindlich
kubectl api-resources -o name                # Alle Ressourcen mit einfacher Ausg
kubectl api-resources -o wide                # Alle Ressourcen mit erweiterter Au
kubectl api-resources --verbs=list,get       # Alle Ressourcen, die "list" und "g
kubectl api-resources --api-group=extensions # Alle Ressourcen in der API-Gruppe
```

Ausgabe formatieren

Um Details in einem bestimmten Format an Ihr Terminalfenster auszugeben, können Sie entweder das `-o` oder `--output` Flag zu einem unterstützten `kubectl` Befehl anhängens.

Ausgabeformat	Beschreibung
`-o=custom-columns=<spec>`	Ausgabe einer Tabelle mit einer durch Kommas getrennten Liste benutzerdefinierter Spalten
`-o=custom-columns-file=<dateiname>`	Drucken Sie eine Tabelle mit der benutzerdefinierten Spaltenvorlage in der `<dateiname>` Datei
`-o=json`	Ausgabe eines JSON-formatierten API-Objekts
`-o=jsonpath=<template>`	Ausgabe der in einem jsonpath-Ausdruck definierten Felder
`-o=jsonpath-file=<dateiname>`	Ausgabe der in einem jsonpath-Ausdruck definierten Felder in der `<dateiname>` Datei
`-o=name`	Ausgabe von nur dem Ressourcennamen und nichts anderes
`-o=wide`	Ausgabe im Klartextformat mit zusätzlichen Informationen. Bei Pods ist der Node-Name enthalten
`-o=yaml`	Gibt ein YAML-formatiertes API-Objekt aus

Kubectl Ausgabe Ausführlichkeit und Debugging

Die Ausführlichkeit von Kubectl wird mit den Flags `-v` oder `--v` gesteuert, gefolgt von einer Ganzzahl, die die Protokollebene darstellt. Allgemeine Protokollierungskonventionen für Kubernetes und die zugehörigen Protokollebenen werden hier beschrieben.

Ausführlichkeit	Beschreibung
`--v=0`	Allgemein nützlich, damit dies für den Bediener IMMER sichtbar ist.
`--v=1`	Eine vernünftige Standardprotokollebene, wenn Sie keine Ausführlichkeit wünschen.
`--v=2`	Nützliche Informationen zum stabilen Status des Dienstes und wichtige Protokollnachrichten, die möglicherweise zu erheblichen Änderungen im System führen. Dies ist die empfohlene Standardprotokollebene für die meisten Systeme.
`--v=3`	Erweiterte Informationen zu Änderungen.
`--v=4`	Debug-Level-Ausführlichkeit.
`--v=6`	Angeforderte Ressourcen anzeigen
`--v=7`	HTTP-Anforderungsheader anzeigen
`--v=8`	HTTP-Anforderungsinhalt anzeigen
`--v=9`	HTTP-Anforderungsinhalt anzeigen, ohne den Inhalt zu kürzen.

Nächste Schritte

- Lernen Sie mehr im Überblick auf kubectl.

- Erkunden Sie kubectl Optionen.
- Und ebenfalls die kubectl Nutzungskonventionen um zu verstehen, wie man es in wiederverwendbaren Skripten verwendet.
- Entdecken Sie mehr Community kubectl Spickzettel.

10 - Tools

Kubernetes enthält mehrere integrierte Tools, die Ihnen bei der Arbeit mit dem Kubernetes System helfen.

Kubectl

`kubectl` ist ein Kommandozeilenprogramm für Kubernetes. Es steuert den Kubernetes Clustermanager.

Kubeadm

`kubeadm` ist ein Kommandozeilenprogramm zur einfachen Bereitstellung eines sicheren Kubernetes-Clusters auf physischen oder Cloud-Servern oder virtuellen Maschinen (derzeit in alpha).

Kubefed

`kubefed` ist ein Kommandozeilenprogramm um Ihnen bei der Verwaltung Ihrer Verbundcluster zu helfen.

Minikube

`minikube` ist ein Tool, das es Ihnen einfach macht, einen Kubernetes-Cluster mit einem einzigen Knoten lokal auf Ihrer Workstation für Entwicklungs- und Testzwecke auszuführen.

Dashboard

`Dashboard` , die webbasierte Benutzeroberfläche von Kubernetes ermöglicht es Ihnen containerisierte Anwendungen in einem Kubernetes-Cluster bereitzustellen Fehler zu beheben und den Cluster und seine Ressourcen selbst zu verwalten.

Helm

`Kubernetes Helm` ist ein Tool zur Verwaltung von Paketen mit vorkonfigurierten Kubernetes-Ressourcen, auch bekannt als Kubernetes charts.

Verwenden Sie Helm um:

- Beliebte Software verpackt als Kubernetes charts zu finden und zu verwenden
- Ihre eigenen Applikationen als Kubernetes charts zu teilen
- Reproduzierbare Builds Ihrer Kubernetes Anwendungen zu erstellen
- Intelligenten Verwaltung von Ihren Kubernetes manifest files
- Verwalten von Versionen von Helm Paketen

Kompose

`Kompose` ist ein Tool, das Docker Compose Benutzern hilft, nach Kubernetes zu wechseln.

Verwenden Sie Kompose um:

- Ein Docker Compose Datei in Kubernetes Objekte zu übersetzen

- Von Ihrer lokalen Docker Entwicklung auf eine Kubernetes verwaltete Entwicklung zu wechseln
- v1 oder v2 Docker Compose `yaml` Dateien oder Distributed Application Bundles zu konvertieren