Aufgaben

Dieser Abschnitt der Kubernetes-Dokumentation enthält Seiten, die zeigen, wie man einzelne Aufgaben erledigt. Eine Aufgabenseite zeigt, wie man eine einzelne Aufgabe ausführt, typischerweise durch eine kurze Abfolge von Schritten.

Webbenutzeroberfläche (Dashboard)

Stellen Sie die Dashboard-Webbenutzeroberfläche bereit, und greifen Sie auf sie zu, um Sie bei der Verwaltung und Überwachung von Containeranwendungen in einem Kubernetes-Cluster zu unterstützen.

Die kubectl-Befehlszeile verwenden

Installieren und konfigurieren Sie das `kubectl` -Befehlszeilentool, mit dem Kubernetes-Cluster direkt verwaltet werden.

Pods und Container konfigurieren

Ausführen allgemeiner Konfigurationsaufgaben für Pods und Container.

Anwendungen ausführen

Ausführen allgemeiner Aufgaben zur Anwendungsverwaltung, z. B. Aktualisierungen, Einfügen von Informationen in Pods und automatisches horizontales Skalieren der Pods.

Jobs ausführen

Jobs mit Parallelverarbeitung ausführen.

Auf Anwendungen in einem Cluster zugreifen

Konfigurieren Sie den Lastausgleich, die Portweiterleitung oder die Einrichtung von Firewall- oder DNS-Konfigurationen für den Zugriff auf Anwendungen in einem Cluster.

Überwachung, Protokollierung und Fehlerbehebung

Richten Sie die Überwachung und Protokollierung ein, um einen Cluster zu behandeln oder eine Container-Anwendung zu debuggen.

Zugriff auf die Kubernetes-API

Lernen Sie verschiedene Methoden kennen, um direkt auf die Kubernetes-API zuzugreifen.

TLS verwenden

Konfigurieren Sie Ihre Anwendung so, dass sie der Cluster-Stammzertifizierungsstelle (Certificate Authority, CA) vertraut und diese verwendet.

Cluster verwalten

Erfahren Sie allgemeine Aufgaben zum Verwalten eines Clusters.

Föderation verwalten

Konfigurieren Sie Komponenten in einer Clusterföderation.

Managing Stateful Applications

Ausführen allgemeiner Aufgaben zum Verwalten von Stateful-Anwendungen, einschließlich Skalieren, Löschen und Debuggen von StatefulSets.

Cluster-Dämonen

Ausführen allgemeiner Aufgaben zum Verwalten eines DaemonSet, z. B. Durchführen eines fortlaufenden Updates.

GPUs verwalten

Konfigurieren und planen Sie NVIDIA-GPUs für die Verwendung durch Nodes in einem Cluster als Ressource.

Verwalten von HugePages

Konfigurieren und verwalten Sie `HugePages` als planbare Ressource in einem Cluster.

Nächste Schritte

Wenn Sie eine Aufgabenseite schreiben möchten, finden Sie weitere Informationen unter Erstellen einer Pull-Anfrage für Dokumentation.

1 - Werkzeuge installieren

1.1 - Installieren und konfigurieren von kubectl

Verwenden Sie das Kubernetes Befehlszeilenprogramm, kubectl, um Anwendungen auf Kubernetes bereitzustellen und zu verwalten. Mit kubectl können Sie Clusterressourcen überprüfen, Komponenten erstellen, löschen und aktualisieren; Ihren neuen Cluster betrachten; und Beispielanwendungen aufrufen.

Bevor Sie beginnen

Sie müssen eine kubectl-Version verwenden, die innerhalb eines geringfügigen Versionsunterschieds zur Version Ihres Clusters liegt. Ein v1.2-Client sollte beispielsweise mit einem v1.1, v1.2 und v1.3-Master arbeiten. Die Verwendung der neuesten Version von kubectl verhindert unvorhergesehene Probleme.

Kubectl installieren

Nachfolgend finden Sie einige Methoden zur Installation von kubectl.

Installieren der kubectl Anwendung mithilfe der systemeigenen Paketverwaltung

Ubuntu, Debian oder HypriotOS | CentOS, RHEL oder Fedora

```
sudo apt-get update && sudo apt-get install -y apt-transport-https
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /e
sudo apt-get update
sudo apt-get install -y kubectl
```

Installation mit snap auf Ubuntu

Wenn Sie Ubuntu oder eine der anderen Linux-Distributionen verwenden, die den snap Paketmanager unterstützen, können Sie kubectl als snap-Anwendung installieren.

1. Wechseln Sie zum Snap-Benutzer und führen Sie den Installationsbefehl aus:

```
sudo snap install kubectl --classic
```

2. Testen Sie, ob die installierte Version ausreichend aktuell ist:

```
kubectl version
```

Installation mit Homebrew auf macOS

Wenn Sie mit macOS arbeiten und den Homebrew Paketmanager verwenden, können Sie kubectl mit Homebrew installieren.

1. Führen Sie den Installationsbefehl aus:

```
brew install kubernetes-cli
```

2. Testen Sie, ob die installierte Version ausreichend aktuell ist:

```
kubectl version
```

Installation mit Macports auf macOS

Wenn Sie mit macOS arbeiten und den Macports Paketmanager verwenden, können Sie kubectl mit Macports installieren.

1. Führen Sie den Installationsbefehl aus:

```
sudo port selfupdate
sudo port install kubectl
```

2. Testen Sie, ob die installierte Version ausreichend aktuell ist:

```
kubectl version
```

Installation mit PowerShell von PSGallery

Wenn Sie mit Windows arbeiten und den Powershell Gallery Paketmanager verwenden, können Sie kubectl mit Powershell installieren und aktualisieren.

1. Führen Sie die Installationsbefehle aus (stellen Sie sicher, dass eine `DownloadLocation` angegeben wird):

```
Install-Script -Name install-kubectl -Scope CurrentUser -Force
install-kubectl.ps1 [-DownloadLocation <path>]
```

> **Hinweis:** Wenn Sie keine `DownloadLocation` angeben, wird `kubectl` im temporären Verzeichnis des Benutzers installiert.

   Das Installationsprogramm erstellt `$HOME/.kube` und weist es an, eine Konfigurationsdatei zu erstellen

2. Testen Sie, ob die installierte Version ausreichend aktuell ist:

```
kubectl version
```

> **Hinweis:** Die Aktualisierung der Installation erfolgt durch erneutes Ausführen der beiden in Schritt 1 aufgelisteten Befehle.

Installation auf Windows mit Chocolatey oder scoop

Um kubectl unter Windows zu installieren, können Sie entweder den Paketmanager Chocolatey oder das Befehlszeilen-Installationsprogramm scoop verwenden.

choco | scoop

```
choco install kubernetes-cli
```

2. Testen Sie, ob die installierte Version ausreichend aktuell ist:

```
```
kubectl version
```
```

3. Navigieren Sie zu Ihrem Heimatverzeichnis:

```
cd %USERPROFILE%
```

4. Erstellen Sie das `.kube` -Verzeichnis:

```
mkdir .kube
```

5. Wechseln Sie in das soeben erstellte `.kube` -Verzeichnis:

```
cd .kube
```

6. Konfigurieren Sie kubectl für die Verwendung eines Remote-Kubernetes-Clusters:

```
New-Item config -type file
```

> **Hinweis:** Bearbeiten Sie die Konfigurationsdatei mit einem Texteditor Ihrer Wahl, z.B. Notepad.

Download als Teil des Google Cloud SDK herunter

Sie können kubectl als Teil des Google Cloud SDK installieren.

1. Installieren Sie das Google Cloud SDK.
2. Führen Sie den `kubectl` -Installationsbefehl aus:

```
gcloud components install kubectl
```

3. Testen Sie, ob die installierte Version ausreichend aktuell ist:

```
kubectl version
```

Installation der kubectl Anwendung mit curl

macOS | Linux | Windows

1. Laden Sie die neueste Version herunter:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/$(cur
```

Um eine bestimmte Version herunterzuladen, ersetzen Sie den Befehlsteil `$(curl -s https://storage.googleapis.com/kubernetes-release/release/stable.txt)` mit der jeweiligen Version.

Um beispielsweise die Version v1.24.0 auf macOS herunterzuladen, verwenden Sie den folgenden Befehl:

```
curl -LO https://storage.googleapis.com/kubernetes-release/release/v1.24
```

2. Machen Sie die kubectl-Binärdatei ausführbar.

```
chmod +x ./kubectl
```

3. Verschieben Sie die Binärdatei in Ihren PATH.

```
sudo mv ./kubectl /usr/local/bin/kubectl
```

kubectl konfigurieren

Damit kubectl einen Kubernetes-Cluster finden und darauf zugreifen kann, benötigt es eine kubeconfig Datei. Diese wird automatisch erstellt, wenn Sie einen Cluster mit kube-up.sh erstellen oder einen Minikube-Cluster erfolgreich implementieren. Weitere Informationen zum Erstellen von Clustern finden Sie in den Anleitungen für die ersten Schritte. Wenn Sie Zugriff auf einen Cluster benötigen, den Sie nicht erstellt haben, lesen Sie die Cluster-Zugriff freigeben Dokumentation. Die kubectl-Konfiguration befindet sich standardmäßig unter `~/.kube/config` .

Überprüfen der kubectl-Konfiguration

Überprüfen Sie, ob kubectl ordnungsgemäß konfiguriert ist, indem Sie den Clusterstatus abrufen:

```
kubectl cluster-info
```

Wenn Sie eine URL-Antwort sehen, ist kubectl korrekt für den Zugriff auf Ihren Cluster konfiguriert.

Wenn eine Meldung ähnlich der folgenden angezeigt wird, ist kubectl nicht richtig konfiguriert oder kann keine Verbindung zu einem Kubernetes-Cluster herstellen.

```
The connection to the server <server-name:port> was refused - did you specify the
```

Wenn Sie beispielsweise vorhaben, einen Kubernetes-Cluster auf Ihrem Laptop (lokal) auszuführen, müssen Sie zunächst ein Tool wie minikube installieren und anschließend die oben genannten Befehle erneut ausführen.

Wenn kubectl cluster-info die URL-Antwort zurückgibt, Sie jedoch nicht auf Ihren Cluster zugreifen können, verwenden Sie Folgendes, um zu überprüfen, ob er ordnungsgemäß konfiguriert ist:

```
kubectl cluster-info dump
```

Aktivieren der automatischen Autovervollständigung der Shell

kubectl bietet Autocompletion-Unterstützung für Bash und Zsh, was Ihnen viel Tipparbeit erspart!

Im Folgenden werden die Verfahren zum Einrichten der automatischen Vervollständigung für Bash (einschließlich der Unterschiede zwischen Linux und macOS) und Zsh beschrieben.

Bash on Linux | Bash auf macOS | Zsh

Einführung

Das kubectl-Vervollständigungsskript für Bash kann mit dem Befehl `kubectl completion bash` generiert werden. Durch das Sourcing des Vervollständigungsskripts in Ihrer Shell wird die automatische Vervollständigung von kubectl ermöglicht.

Das Fertigstellungsskript benötigt jedoch **bash-completion**. Dies bedeutet, dass Sie diese Software zuerst installieren müssen (Sie können testen, ob Sie bereits bash-completion installiert haben, indem Sie `type _init_completion` ausführen).

Installation von bash-completion

bash-completion wird von vielen Paketmanagern bereitgestellt (siehe hier). Sie können es mittels `apt-get install bash-completion` oder `yum install bash-completion` , usw.

Die obigen Befehle erstellen `/usr/share/bash-completion/bash_completion` ,Dies ist das Hauptskript für die Bash-Vollendung. Abhängig von Ihrem Paketmanager müssen Sie diese Datei manuell in Ihre `~ / .bashrc` -Datei eingeben.

Um dies herauszufinden, laden Sie Ihre Shell erneut und führen Sie `type _init_completion` aus. Wenn der Befehl erfolgreich ist, ist bereits alles vorbereitet. Andernfalls fügen Sie der `~/.bashrc` -Datei Folgendes hinzu:

```
source /usr/share/bash-completion/bash_completion
```

Laden Sie Ihre Shell erneut und vergewissern Sie sich, dass bash-completion korrekt installiert ist, indem Sie folgendes eingeben: `type _init_completion` .

Aktivieren der automatische Vervollständigung von kubectl

Sie müssen nun sicherstellen, dass das kubectl-Abschlussskript in allen Ihren Shell-Sitzungen verwendet wird. Es gibt zwei Möglichkeiten, dies zu tun:

- Fügen Sie das Vervollständigungsskript Ihrer `~ /.bashrc` -Datei hinzu:

  ```
  echo 'source <(kubectl completion bash)' >>~/.bashrc
  ```

- Fügen Sie das Vervollständigungsskript zum Verzeichnis `/etc/bash_completion.d` hinzu:

  ```
  kubectl completion bash >/etc/bash_completion.d/kubectl
  ```

> **Hinweis:** bash-completion bezieht alle Vervollständigungsskripte aus `/etc/bash_completion.d`.

Beide Ansätze sind gleichwertig. Nach dem erneuten Laden der Shell sollte kubectl autocompletion funktionieren.

Nächste Schritte

Erfahren Sie, wie Sie Ihre Anwendung starten und verfügbar machen.

1.2 - Installation von Minikube

Diese Seite zeigt Ihnen, wie Sie Minikube installieren, ein Programm, das einen Kubernetes-Cluster mit einem einzigen Node in einer virtuellen Maschine auf Ihrem Laptop ausführt.

Bevor Sie beginnen

Die VT-x- oder AMD-v-Virtualisierung muss im BIOS Ihres Computers aktiviert sein. Um dies unter Linux zu überprüfen, führen Sie Folgendes aus und vergewissern Sie sich, dass die Ausgabe nicht leer ist:

```
egrep --color 'vmx|svm' /proc/cpuinfo
```

Einen Hypervisor installieren

Wenn noch kein Hypervisor installiert ist, installieren Sie jetzt einen für Ihr Betriebssystem:

Betriebssystem	Unterstützte Hypervisoren
macOS	VirtualBox, VMware Fusion, HyperKit
Linux	VirtualBox, KVM
Windows	VirtualBox, Hyper-V

> **Hinweis:** Minikube unterstützt auch die Option `--vm-driver=none`, mit der die Kubernetes-Komponenten auf dem Host und nicht in einer VM ausgeführt werden. Die Verwendung dieses Treibers erfordert Docker und eine Linux-Umgebung, jedoch keinen Hypervisor.

Kubectl installieren

- Installieren Sie kubectl gemäß den Anweisungen in kubectl installieren und einrichten.

Minikube installieren

macOS

Die einfachste Möglichkeit, Minikube unter macOS zu installieren, ist die Verwendung von Homebrew:

```
brew install minikube
```

Sie können es auch auf macOS installieren, indem Sie eine statische Binärdatei herunterladen:

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikub
  && chmod +x minikube
```

So fügen Sie die Minikube-Programmdatei auf einfache Weise Ihrem Pfad hinzu:

```
sudo mv minikube /usr/local/bin
```

Linux

> **Hinweis:** Dieses Dokument zeigt Ihnen, wie Sie Minikube mit einer statischen Binärdatei unter Linux installieren. Für alternative Linux-Installationsmethoden siehe Andere Installationsmethoden im offiziellen Minikube-GitHub-Repository.

Sie können Minikube unter Linux installieren, indem Sie eine statische Binärdatei herunterladen:

```
curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikub
  && chmod +x minikube
```

So fügen Sie die Minikube-Programmdatei auf einfache Weise Ihrem Pfad hinzu:

```
sudo cp minikube /usr/local/bin && rm minikube
```

Windows

> **Hinweis:** Um Minikube unter Windows auszuführen, müssen Sie zuerst VirtualBox oder Hyper-V installieren. Hyper-V kann auf drei Versionen von Windows 10 ausgeführt werden: Windows 10 Enterprise, Windows 10 Professional und Windows 10 Education. Weitere Informationen zur Installation finden Sie im offiziellen Minikube GitHub-Repository.

Die einfachste Möglichkeit, Minikube unter Windows zu installieren, ist die Verwendung von Chocolatey (als Administrator ausführen):

```
choco install minikube kubernetes-cli
```

Schließen Sie nach der Installation von Minikube die aktuelle CLI-Sitzung und starten Sie sie neu. Minikube sollte automatisch zu Ihrem Pfad hinzugefügt werden.

Manuelle installation unter Windows

Um Minikube manuell unter Windows zu installieren, laden Sie die Datei `minikube-windows-amd64` herunter, umbenennen Sie sie in `minikube.exe` und fügen Sie sie Ihrem Pfad zu.

Windows Installer

So installieren Sie Minikube manuell unter Windows mit Windows Installer, laden Sie die Datei `minikube-installer.exe` und führen Sie den Installer aus.

Eine bestehende Installation bereinigen

Wenn Sie minikube bereits installiert haben, starten Sie die Anwendung:

```
minikube start
```

Und der Befehl gibt einen Fehler zurück:

```
machine does not exist
```

Müssen Sie die Konfigurationsdateien löschen:

```
rm -rf ~/.minikube
```

Nächste Schritte

- Kubernetes lokal über Minikube ausführen

2 - Einen Cluster verwalten

Lerne allgemeine Aufgaben zur Verwaltung eines Clusters kennen.

2.1 - Verwaltung mit kubeadm

3 - Pods und Container konfigurieren

4 - Daten in Anwendungen injizieren

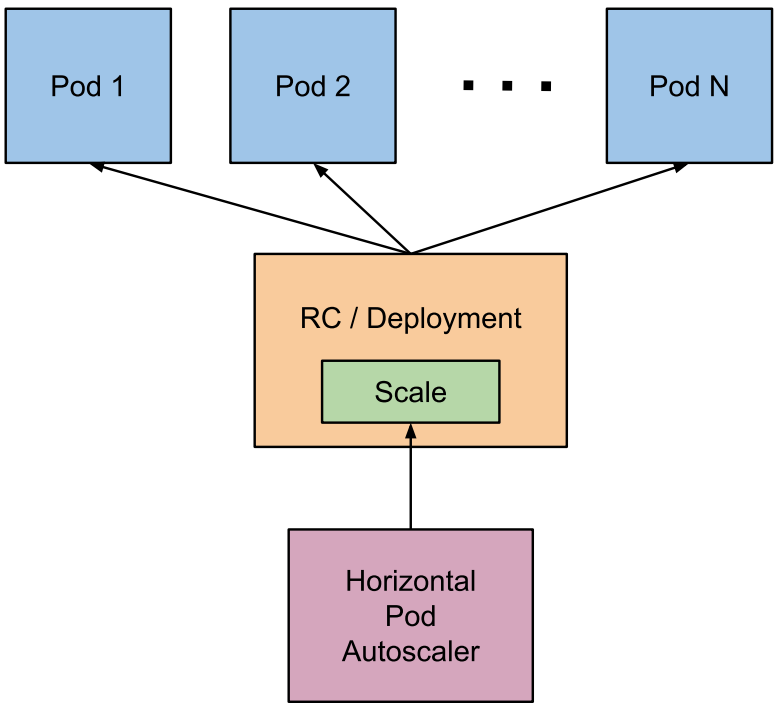
5 - Anwendungen ausführen

5.1 - Horizontal Pod Autoscaler

Der Horizontal Pod Autoscaler skaliert automatisch die Anzahl der Pods eines Replication Controller, Deployment oder Replikat Set basierend auf der beobachteten CPU-Auslastung (oder, mit Unterstützung von benutzerdefinierter Metriken, von der Anwendung bereitgestellten Metriken). Beachte, dass die horizontale Pod Autoskalierung nicht für Objekte gilt, die nicht skaliert werden können, z. B. DaemonSets.

Der Horizontal Pod Autoscaler ist als Kubernetes API-Ressource und einem Controller implementiert. Die Ressource bestimmt das Verhalten des Controllers. Der Controller passt die Anzahl der Replikate eines Replication Controller oder Deployments regelmäßig an, um die beobachtete durchschnittliche CPU-Auslastung an das vom Benutzer angegebene Ziel anzupassen.

Wie funktioniert der Horizontal Pod Autoscaler?



Der Horizontal Pod Autoscaler ist als Kontrollschleife mit einer Laufzeit implementiert, die durch das Flag `--horizontal-pod-autoscaler-sync-period` am Controller Manager gesteuert wird (mit einem Standardwert von 15 Sekunden).

Während jedem Durchlauf fragt der Controller Manager die Ressourcennutzung anhand der in jeder HorizontalPodAutoscaler Definition angegebenen Metriken ab. Der Controller Manager bezieht die Metriken entweder aus der Resource Metrics API (für Ressourcenmetriken pro Pod) oder aus der Custom Metrics API (für alle anderen Metriken).

- Für jede pro Pod Ressourcenmetriken (wie CPU) ruft der Controller die Metriken über die Ressourcenmetriken API für jeden Pod ab, der vom HorizontalPodAutoscaler angesprochen wird. Sofern ein Zielnutzungswert eingestellt ist, berechnet der Controller den Nutzungswert als Prozentsatz der äquivalenten Ressourcenanforderung der Containern in jedem Pod. Wenn ein Ziel-Rohwert eingestellt ist, werden die Rohmetrikenwerte direkt verwendet. Der Controller nimmt dann den Mittelwert der Auslastung oder den Rohwert (je nach Art des angegebenen Ziels) über alle Zielpods und erzeugt ein Quotienten, mit dem die Anzahl der gewünschten Replikate skaliert wird.

Beachte, dass, wenn einige der Container des Pods nicht über den entsprechenden Ressourcenanforderung verfügen, die CPU-Auslastung für den Pod nicht definiert wird und der Autoscaler keine Maßnahmen bezüglich dieser Metrik ergreift. Weitere Informationen zur Funktionsweise des Autoskalierungsalgorithmus finden Sie im folgenden Abschnitt über den Algorithmus.

- Bei benutzerdefinierten Metriken pro Pod funktioniert die Steuerung ähnlich wie bei Ressourcenmetriken pro Pod, nur dass diese mit Rohwerten und nicht mit Nutzungswerten arbeitet.

- Für Objektmetriken und externe Metriken wird eine einzelne Metrik abgerufen, die das jeweilige Objekt beschreibt. Diese Kennzahl wird mit dem Sollwert verglichen, um ein Verhältnis wie oben beschrieben zu erhalten. In der API-Version von `autoscaling/v2beta2` kann dieser Wert optional durch die Anzahl der Pods geteilt werden, bevor der Vergleich durchgeführt wird.

Der HorizontalPodAutoscaler holt Metriken normalerweise aus einer Reihe von aggregierten APIs (`metrics.k8s.io` , `custom.metrics.k8s.io` und `external.metrics.k8s.io`). Die API `metrics.k8s.io` wird normalerweise vom Metrics Server bereitgestellt, der separat gestartet werden muss. Siehe Metrics Server für weitere Anweisungen. Der HorizontalPodAutoscaler kann auch Metriken direkt aus dem Heapster beziehen.

> **Hinweis:**
>
> **FEATURE STATE:** `Kubernetes 1.11 [deprecated]`
>
> Das Verwenden von Metriken aus Heapster ist seit der Kubernetes Version 1.11 veraltet.

Siehe Unterstützung der Metrik APIs für weitere Details.

Der Autoscaler greift über die Scale Sub-Ressource auf die entsprechenden skalierbaren Controller (z.B. Replication Controller, Deployments und Replika Sets) zu. Scale ist eine Schnittstelle, mit der Sie die Anzahl der Replikate dynamisch einstellen und jeden ihrer aktuellen Zustände untersuchen können. Weitere Details zu der Scale Sub-Ressource findest du hier.

Details zum Algorithmus

Vereinfacht gesagt arbeitet der Horizontal Pod Autoscaler Controller mit dem Verhältnis zwischen dem gewünschten metrischen Wert und dem aktuellen metrischen Wert:

```
desiredReplicas = ceil[currentReplicas * ( currentMetricValue / desiredMetricValu
```

Wenn beispielsweise der aktuelle metrische Wert `200m` und der gewünschte Wert `100m` ist, wird die Anzahl der Replikate verdoppelt, da `200.0 / 100.0 == 2.0` ist. Wenn der aktuelle Wert jedoch `50m` ist, halbieren sich die Anzahl der Replikate `50.0 / 100.0 == 0.5` . Es wird auf die Skalierung verzichtet, wenn das Verhältnis ausreichend nahe bei 1,0 liegt (innerhalb einer global konfigurierbaren Toleranz, vom Flag `--horizontal-pod-autoscaler-tolerance` , das standardmäßig auf 0,1 gesetzt ist).

Wenn ein `targetAverageValue` oder `targetAverageUtilization` angegeben wird, wird der `currentMetricValue` berechnet, indem der Mittelwert der gegebenen Metrik über alle Pods im Skalierungsziel des HorizontalPodAutoscaler berechnet wird. Vor der Überprüfung der Toleranz und der Entscheidung über die finalen Werte berücksichtigen wir jedoch die Pod Readiness und fehlende Metriken.

Alle Pods mit einem gesetzten Zeitstempel zur Löschung (d.h. Pods, die gerade heruntergefahren werden) und alle ausgefallenen Pods werden verworfen.

Wenn einem bestimmten Pod Metriken fehlen, wird es für später zurückgestellt; Pods mit fehlenden Metriken werden verwendet, um den endgültigen Skalierungsmenge anzupassen.

Wenn bei der Skalierung anhand der CPU ein Pod noch nicht bereit ist (d.h. er wird noch initialisiert) *oder* der letzte metrische Punkt für den Pod vor dessen Einsatzbereitschaft liegt, wird auch dieser Pod zurückgestellt.

Aufgrund technischer Einschränkungen kann der HorizontalPodAutoscaler Controller nicht genau bestimmen, wann ein Pod zum ersten Mal bereit ist, wenn es darum geht, bestimmte CPU Metriken festzulegen. Stattdessen betrachtet er eine Pod als "not yet ready", wenn dieser noch nicht bereit ist und geht in "unready" über, innerhalb eines kurzen, konfigurierbaren Zeitfensters seit dem Start. Dieser Wert wird mit dem Flag `--horizontal-pod-autoscaler-initial-readiness-delay` konfiguriert und ist standardmäßig auf 30 Sekunden eingestellt. Sobald ein Pod bereit ist, betrachtet er jeden Übergang zu Bereit als den ersten, wenn dies innerhalb einer längeren, konfigurierbaren Zeit seit seinem Start erfolgt ist. Dieser Wert wird mit dem Flag `--horizontal-pod-autoscaler-cpu-initialization-period` gesetzt und dessen Standardwert beträgt 5 Minuten.

Das Basisskalenverhältnis `currentMetricValue / desiredMetricValue` wird dann mit den restlichen Pods berechnet, die nicht zurückgestellt oder von den oben genannten Kriterien entsorgt wurden.

Wenn es irgendwelche fehlenden Metriken gab, berechnen wir den Durchschnitt konservativer, vorausgesetzt, dass die Pods 100% des gewünschten Wertes bei der Verringerung und 0% bei einer Vergrößerung verbrauchten. Dadurch wird die Dimension einer beliebigen potenziellen Skalierung verringert.

Wenn außerdem noch nicht bereite Pods vorhanden sind und es ohne Berücksichtigung fehlender Metriken oder noch nicht bereiter Pods skaliert wurde, wird konservativ davon ausgegangen, dass die noch nicht bereiten Pods 0% der gewünschten Metrik verbrauchen, was die Dimension einer Skalierung weiter dämpft.

Nach Berücksichtigung der noch nicht bereiten Pods und fehlender Metriken wird der Nutzungsgrad neu berechnet. Wenn das neue Verhältnis die Skalierungsrichtung umkehrt oder innerhalb der Toleranz liegt, wird das weitere Skalieren übersprungen. Andernfalls wird das neue Verhältnis zur Skalierung verwendet.

Beachte, dass der *ursprüngliche* Wert für die durchschnittliche Auslastung über den HorizontalPodAutoscaler Status zurückgemeldet wird, ohne die noch nicht bereiten Pods oder fehlende Metriken zu berücksichtigen, selbst wenn das neue Nutzungsverhältnis verwendet wird.

Wenn mehrere Metriken in einem HorizontalPodAutoscaler angegeben sind, wird die Berechnung für jede Metrik durchgeführt, und dann wird die größte der gewünschten Replikanzahl ausgewählt. Wenn eine dieser Metriken nicht in eine gewünschte Replikanzahl umgewandelt werden kann (z.B. aufgrund eines Fehlers beim Abrufen der Metriken aus den Metrik APIs), wird diese Skalierung übersprungen.

Schließlich, kurz bevor HPA das Ziel skaliert, wird die Skalierungsempfehlung aufgezeichnet. Der Controller berücksichtigt alle Empfehlungen innerhalb eines konfigurierbaren Fensters und wählt aus diesem Fenster die höchste Empfehlung aus. Dieser Wert kann mit dem Flag `--horizontal-pod-autoscaler-downscale-stabilization` konfiguriert werden, das standardmäßig auf 5 Minuten eingestellt ist. Dies bedeutet, dass die Skalierung schrittweise erfolgt, wodurch die Auswirkungen schnell schwankender metrischer Werte ausgeglichen werden.

API Objekt

Der Horizontal Pod Autoscaler ist eine API Ressource in der Kubernetes `autoscaling` API Gruppe. Die aktuelle stabile Version, die nur die Unterstützung für die automatische Skalierung der CPU beinhaltet, befindet sich in der `autoscaling/v1` API Version.

Die Beta-Version, welche die Skalierung des Speichers und benutzerdefinierte Metriken unterstützt, befindet sich unter `autoscaling/v2beta2` . Die in `autoscaling/v2beta2` neu eingeführten Felder bleiben bei der Arbeit mit `autoscaling/v1` als Anmerkungen erhalten.

Weitere Details über das API Objekt kann unter dem HorizontalPodAutoscaler Objekt gefunden werden.

Unterstützung des Horizontal Pod Autoscaler in kubectl

Der Horizontal Pod Autoscaler wird, wie jede API-Ressource, standardmäßig von `kubectl` unterstützt. Ein neuer Autoskalierer kann mit dem Befehl `kubectl create` erstellt werden. Das auflisten der Autoskalierer geschieht über `kubectl get hpa` und eine detaillierte Beschreibung erhält man mit `kubectl describe hpa` . Letzendlich können wir einen Autoskalierer mit `kubectl delete hpa` löschen.

Zusätzlich gibt es einen speziellen Befehl `kubectl autoscale` zur einfachen Erstellung eines Horizontal Pod Autoscalers. Wenn du beispielsweise `kubectl autoscale rs foo --min=2 --max=5 --cpu-percent=80` ausführst, wird ein Autoskalierer für den ReplicaSet *foo* erstellt, wobei die Ziel-CPU-Auslastung auf `80%` und die Anzahl der Replikate zwischen 2 und 5 gesetzt wird. Die Detaildokumentation von `kubectl autoscale` kann hier gefunden werden.

Autoskalieren während rollierender Updates

Derzeit ist es in Kubernetes möglich, ein rollierendes Update durchzuführen, indem du den Replikationscontroller direkt verwaltest oder das Deployment Objekt verwendest, das die zugrunde liegenden Replica Sets für dich verwaltet. Der Horizontal Pod Autoscaler unterstützt nur den letztgenannten Ansatz: Der Horizontal Pod Autoscaler ist an das Deployment Objekt gebunden, er legt die Größe für das Deployment Objekt fest, und das Deployment ist für die Festlegung der Größen der zugrunde liegenden Replica Sets verantwortlich.

Der Horizontal Pod Autoscaler funktioniert nicht mit rollierendem Update durch direkte Manipulation vom Replikationscontrollern, d.h. du kannst einen Horizontal Pod Autoscaler nicht an einen Replikationscontroller binden und rollierend aktualisieren (z.B. mit `kubectl rolling-update`). Der Grund dafür ist, dass beim Erstellen eines neuen Replikationscontrollers durch ein rollierendes Update der Horizontal Pod Autoscaler nicht an den neue Replikationscontroller gebunden wird.

Unterstützung von Abklingzeiten/Verzögerungen

Bei der Verwaltung der Größe einer Gruppe von Replikaten mit dem Horizontal Pod Autoscaler ist es möglich, dass die Anzahl der Replikate aufgrund der Dynamik der ausgewerteten Metriken häufig schwankt. Dies wird manchmal als *thrashing*, zu deutsch *Flattern*, bezeichnet.

Ab v1.6 kann ein Cluster Operator dieses Problem mitigieren, indem er die globalen HPA Einstellungen anpasst, die als Flags für die Komponente `kube-controller-manager` dargelegt werden:

Ab v1.12 erübrigt ein neues Update des Algorithmus die Notwendigkeit der Verzögerung beim hochskalieren.

- `--horizontal-pod-autoscaler-downscale-stabilization` : Der Wert für diese Option ist eine Dauer, die angibt, wie lange der Autoscaler warten muss, bis nach Abschluss des aktuellen Skalierungsvorgangs ein weiterer Downscale durchgeführt werden kann. Der Standardwert ist 5 Minuten (`5m0s`).

> **Hinweis:** Beim Abstimmen dieser Parameterwerte sollte sich ein Clusterbetreiber der möglichen Konsequenzen bewusst sein. Wenn der Wert für die Verzögerung (Abklingzeit) zu groß eingestellt ist, kann es zu Beschwerden kommen, dass der Horizontal Pod Autoscaler nicht auf Änderungen der Arbeitslast reagiert. Wenn der Verzögerungswert

> jedoch zu kurz eingestellt ist, kann es vorkommen, dass die Skalierung der eingestellten Replikate wie gewohnt weiter flattert.

Unterstützung von mehrere Metriken

Kubernetes 1.6 bietet Unterstützung für die Skalierung basierend auf mehreren Metriken. Du kannst die API Version `autoscaling/v2beta2` verwenden, um mehrere Metriken für den Horizontal Pod Autoscaler zum Skalieren festzulegen. Anschließend wertet der Horizontal Pod Autoscaler Controller jede Metrik aus und schlägt eine neue Skalierung basierend auf diesen Metrik vor. Die größte der vorgeschlagenen Skalierung wird als neue Skalierung verwendet.

Unterstützung von benutzerdefinierte Metriken

> **Hinweis:** Kubernetes 1.2 bietet Alpha Unterstützung für die Skalierung basierend auf anwendungsspezifischen Metriken über speziellen Annotations. Die Unterstützung für diese Annotations wurde in Kubernetes 1.6 zugunsten der neuen autoskalierenden API entfernt. Während die alte Methode zum Sammeln von benutzerdefinierten Metriken weiterhin verfügbar ist, stehen diese Metriken dem Horizontal Pod Autoscaler nicht mehr zur Verfügung, ebenso wenig wie die früheren Annotations zur Angabe, welche benutzerdefinierten Metriken zur Skalierung vom Horizontal Pod Autoscaler Controller berücksichtigt werden sollen.

Kubernetes 1.6 bietet Unterstützung für die Verwendung benutzerdefinierter Metriken im Horizontal Pod Autoscaler. Du kannst benutzerdefinierte Metriken für den Horizontal Pod Autoscaler hinzufügen, die in der `autoscaling/v2beta2` API verwendet werden. Kubernetes fragt dann die neue API für die benutzerdefinierte Metriken ab, um die Werte der entsprechenden benutzerdefinierten Metriken zu erhalten.

Die Voraussetzungen hierfür werden im nachfolgenden Kapitel Unterstützung für die Metrik APIs geklärt.

Unterstützung der Metrik APIs

Standardmäßig ruft der HorizontalPodAutoscaler Controller Metriken aus einer Reihe von APIs ab. Damit dieser auf die APIs zugreifen kann, muss der Cluster Administratoren sicherstellen, dass:

- Der API Aggregations Layer aktiviert ist.
- Die entsprechenden APIs registriert sind:
  - Für Ressourcenmetriken ist dies die API `metrics.k8s.io` , die im Allgemeinen von metrics-server bereitgestellt wird. Es kann als Cluster-Addon gestartet werden.
  - Für benutzerdefinierte Metriken ist dies die API `custom.metrics.k8s.io` . Diese wird vom "Adapter" API Servern bereitgestellt, welches von Anbietern von Metrik Lösungen beliefert wird. Überprüfe dies mit deiner Metrik Pipeline oder der Liste bekannter Lösungen. Falls du deinen eigenen schreiben möchtest hilft dir folgender boilerplate um zu starten.
  - Für externe Metriken ist dies die `external.metrics.k8s.io` API. Es kann sein, dass dies durch den benutzerdefinierten Metrik Adapter bereitgestellt wird.
- Das Flag `--horizontal-pod-autoscaler-use-rest-clients` ist auf `true` oder ungesetzt. Wird dies auf `false` gesetzt wird die Heapster basierte Autoskalierung aktiviert, welche veraltet ist.

Nächste Schritte

- Design Dokument Horizontal Pod Autoscaling.
- kubectl autoscale Befehl: kubectl autoscale.
- Verwenden des Horizontal Pod Autoscaler.

6 - Jobs ausführen

Führen Sie Jobs mit paralleler Verarbeitung aus.

7 - Auf Anwendungen in einem Cluster zugreifen

8 - Überwachung, Protokollierung und Fehlerbehebung

9 - Kubernetes erweitern

10 - TLS

11 - Föderation

12 - Cluster-Daemons verwalten

13 - Service Catalog installieren