Diese Anleitung erläutert, wie WordPress und MediaWiki mittels Helm und Jira durch YAML-Dateien in einem Kubernetes-Cluster aufgesetzt werden können.

WordPress

Ziel der Anleitung

Diese Anleitung erklärt Schritt für Schritt, wie Sie MediaWiki auf einem Kubernetes-Cluster in Google Cloud installieren. Dabei wird Helm verwendet, um die Installation und Konfiguration zu vereinfachen.

Schritt 1: Authentifizierung und Einrichtung des Zugriffs

gcloud auth login

gcloud container clusters get-credentials [CLUSTER-NAME] --zone [ZONE] --project [PROJECT-ID]

Schritt 2: Hinzufügen des Helm-Repositories

Zweck: Stellt die Verfügbarkeit des WordPress-Helm-Charts sicher

helm repo add bitnami https://charts.bitnami.com/bitnami

helm repo update

- **Beschreibung:** Fügt das Bitnami Repository zu Helm hinzu und aktualisiert die Liste der verfügbaren Charts, damit Sie das neueste WordPress-Chart installieren können.

Schritt 3: Namespace-Erstellung

Zweck: Trennt die WordPress-Ressourcen in einem eigenen Kubernetes Namespace.

kubectl create namespace wordpress

- **Beschreibung:** Erstellt einen neuen Namespace namens **wordpress**, der hilft, die Ressourcen organisatorisch von anderen Anwendungen im Cluster zu isolieren.

Schritt 4: Installation von WordPress

Zweck: Führt die Installation von WordPress durch und konfiguriert alle benötigten Kubernetes-Ressourcen automatisch.

helm install my-wordpress bitnami/wordpress --namespace wordpress

- **Beschreibung:** Installiert WordPress im **wordpress** Namespace unter Verwendung des Bitnami Helm Charts. Dieser Schritt richtet alle notwendigen Dienste und Pods automatisch ein.

Schritt 5: Nachbereitung der WordPress-Installation

- **Beschreibung:** Nach der Installation von WordPress sind weitere Konfigurationsschritte erforderlich, um Zugangsdaten wie Benutzernamen, Passwort und die IP-Adresse zu ermitteln. Diese Informationen sind essentiell, um auf das WordPress -Interface zugreifen zu können. Folgen Sie den Helm-Chart-Anweisungen oder den Output-Informationen, die nach der Installation angezeigt werden, um diese Konfigurationen vorzunehmen. Dies ist notwendig, um sicherzustellen, dass Sie erfolgreich auf WordPress über die externe IP-Adresse zugreifen können.

Schritt 6: Konfigurieren des Zugriffs

Zweck: Ermöglicht den Zugriff auf WordPress von außerhalb des Kubernetes-Clusters.

kubectl get services --namespace wordpress

- **Beschreibung:** Listet die Dienste auf, die im **wordpress** Namespace laufen, um die externe IP-Adresse oder den Port zu ermitteln, über die/den WordPress zugänglich ist.

- Sobald die Installation abgeschlossen ist und Sie das Passwort entschlüsselt sowie die IP-Adresse ermittelt haben, können Sie auf die WordPress-Webseite zugreifen. Verwenden Sie dazu die externe IP-Adresse im Webbrowser. Loggen Sie sich anschließend mit dem Benutzernamen „user“ und dem entsprechenden Passwort ein. Dies ermöglicht Ihnen den vollständigen Zugriff auf die WordPress-Plattform und die Möglichkeit, diese nach Ihren Bedürfnissen zu konfigurieren und zu nutzen.

MediaWiki

Schritt 1: Authentifizierung und Einrichtung des Zugriffs

gcloud auth login

gcloud container clusters get-credentials [CLUSTER-NAME] --zone [ZONE] --project [PROJECT-ID]

Schritt 2: Hinzufügen des Helm-Repositories

Zweck: Stellt die Verfügbarkeit des MediaWiki-Helm-Charts sicher

helm repo add bitnami https://charts.bitnami.com/bitnami

helm repo update

- **Beschreibung:** Fügt das Bitnami Repository zu Helm hinzu und aktualisiert die Liste der verfügbaren Charts, damit Sie das neueste MediaWiki-Chart installieren können.

Schritt 3: Namespace-Erstellung

Zweck: Trennt die MediaWiki-Ressourcen in einem eigenen Kubernetes Namespace.

kubectl create namespace mediawiki

- **Beschreibung:** Erstellt einen neuen Namespace namens **mediawiki**, der hilft, die Ressourcen organisatorisch von anderen Anwendungen im Cluster zu isolieren.

Schritt 4: Installation von MediaWiki

Zweck: Führt die Installation von MediaWiki durch und konfiguriert alle benötigten Kubernetes-Ressourcen automatisch.

helm install my-mediawiki bitnami/mediawiki --namespace mediawiki

- **Beschreibung:** Installiert MediaWiki im **mediawiki** Namespace unter Verwendung des Bitnami Helm Charts. Dieser Schritt richtet alle notwendigen Dienste und Pods automatisch ein.

Schritt 5: Nachbereitung der MediaWiki-Installation

- **Beschreibung:** Nach der Installation von MediaWiki sind weitere Konfigurationsschritte erforderlich, um Zugangsdaten wie Benutzernamen, Passwort und die IP-Adresse zu ermitteln. Diese Informationen sind essentiell, um auf das MediaWiki-Interface zugreifen zu können. Folgen Sie den Helm-Chart-Anweisungen oder den Output-Informationen, die nach der Installation angezeigt werden, um diese Konfigurationen vorzunehmen. Dies ist notwendig, um sicherzustellen, dass Sie erfolgreich auf MediaWiki über die externe IP-Adresse zugreifen können.

Schritt 6: Konfigurieren des Zugriffs

Zweck: Ermöglicht den Zugriff auf MediaWiki von außerhalb des Kubernetes-Clusters.

kubectl get services --namespace mediawiki

- **Beschreibung:** Listet die Dienste auf, die im **mediawiki** Namespace laufen, um die externe IP-Adresse oder den Port zu ermitteln, über die/den MediaWiki zugänglich ist.

- Sobald die Installation abgeschlossen ist und Sie das Passwort entschlüsselt sowie die IP-Adresse ermittelt haben, können Sie auf die MediaWiki-Webseite zugreifen. Verwenden Sie dazu die externe IP-Adresse im Webbrowser. Loggen Sie sich anschließend mit dem Benutzernamen „user“ und dem entsprechenden Passwort ein. Dies ermöglicht Ihnen den vollständigen Zugriff auf die MediaWiki-Plattform und die Möglichkeit, diese nach Ihren Bedürfnissen zu konfigurieren und zu nutzen.

JIRA

1. AKS-Cluster erstellen

- Zuerst habe ich den Azure CLI-Befehl az aks create verwendet, um einen Azure Kubernetes Service (AKS) Cluster zu erstellen.

az aks create --resource-group MyResourceGroup --name MyAKSCluster --node-count 1 --generate-ssh-keys

2. .YAML-Files anwenden

- Anschließend habe ich kubectl apply verwendet, um die benötigten Ressourcen zu erstellen

kubectl apply -f jira-postgres-volume-claim.yaml

```
kristian [ ~ ]$ kubectl apply -f jira-postgres-volume-claim.yaml
persistentvolumeclaim/jira-postgres-volume-claim created
persistentvolumeclaim/jira-volume-claim created
```

kubectl apply -f jira-deployment.yaml

```
kristian [ ~ ]$ kubectl apply -f jira-deployment.yaml
deployment.apps/jira-deployment created
service/jira-service created
```

3. Pods überwacht

- Danach habe ich kubectl get pods -n jira verwendet, um die Pods im Jira-Namespace zu überprüfen und den Status der laufenden Dienste zu überwachen.
- Ich habe auch kubectl describe pod genutzt, um detaillierte Informationen über einen bestimmten Pod zu erhalten.

kubectl get pods -n jira

kubectl describe pod jira-<pod-id> -n jira

4. Logs überprüft

- Ich habe kubectl logs verwendet, um die Logs des Jira-Pods zu lesen. Dies half mir sicherzustellen, dass der Dienst korrekt gestartet ist und um etwaige Fehler zu erkennen.

kubectl logs jira-7574f9d647-c47s2 -n jira

5. Jira-Dienst bearbeitet

- Ich habe kubectl edit svc verwendet, um den Jira-Dienst zu bearbeiten und die Konfiguration anzupassen. Dieser Befehl öffnete die YAML-Datei des Dienstes zur Bearbeitung. Ich habe den Typ von NodePort zu LoadBalancer umgeschrieben.

kubectl edit svc jira -n jira

6. Dienste überprüft:

- Ich habe kubectl get svc verwendet, um die in meinem Cluster laufenden Dienste anzuzeigen. So konnte ich einen Überblick über die externen und internen IP-Adressen der Dienste gewinnen.

kubectl get svc -n jira

7. Jira im Webbrowser öffnen

- Nachdem ich die IP-Adresse entziffert habe, konnte ich diese endlich im Web-Browser eingeben und Jira dort anwenden.

https://20.242.170.232:8080

Installationsanleitung für Portainer auf einem Kubernetes-Cluster in Google Cloud

Schritt 1: Authentifizierung und Konfiguration des Cluster-Zugriffs

Zweck: Stellt sicher, dass Sie Befehle an Ihren Kubernetes-Cluster senden können.

gcloud auth login

gcloud container clusters get-credentials [CLUSTER-NAME] --zone [ZONE] --project [PROJECT-ID]

- **Beschreibung:** Erlaubt die Kommunikation mit Ihrem Kubernetes-Cluster über **kubectl**, nachdem Sie sich erfolgreich authentifiziert haben.

Schritt 2: Hinzufügen des Helm-Repositories für Portainer

Zweck: Ermöglicht den Zugriff auf die neuesten Portainer-Helm-Charts.

helm repo add portainer https://portainer.github.io/k8s/

helm repo update

- **Beschreibung:** Fügt das offizielle Portainer Helm Repository hinzu und aktualisiert Ihre lokale Helm-Chart-Liste, sodass Sie das neueste Chart installieren können.

Schritt 3: Erstellung eines Namespace für Portainer

Zweck: Isoliert die Portainer-Installation in einem eigenen Bereich des Clusters.

kubectl create namespace portainer

- **Beschreibung:** Durch die Erstellung eines eigenen Namespaces für Portainer wird sichergestellt, dass die Portainer-Ressourcen getrennt von anderen Anwendungen verwaltet werden können.

Schritt 4: Installation von Portainer

Zweck: Installiert Portainer in Ihrem Cluster zur effektiven Verwaltung Ihrer Kubernetes-Ressourcen

helm install portainer portainer/portainer --namespace portainer

- **Beschreibung:** Dieser Schritt nutzt das Portainer Helm Chart, um Portainer schnell und sicher im spezifizierten Namespace zu installieren. Portainer wird dann alle nötigen Komponenten wie Deployment, Service und ggf. Persistent Volumes einrichten.

Schritt 5: Zugriff auf Portainer konfigurieren

Zweck: Ermöglicht den Zugriff auf die Portainer-Weboberfläche, um Ihre Kubernetes-Ressourcen zu überwachen und zu verwalte

kubectl get services --namespace portainer

- **Beschreibung:** Ermittelt die IP-Adresse oder den Port, der/die für den Zugriff auf die Portainer-Weboberfläche verwendet wird. Abhängig von Ihrer Service-Konfiguration kann dies ein LoadBalancer mit einer externen IP sein oder ein NodePort.