Kubernetes Cluster installieren (Baremetal)

Einleitung

In diesem Artikel geht es darum, wie wir einen Kubernetes Cluster aufsetzen können, um von den Funktionalitäten von Kubernetes zu profitieren. Kubernetes ist mittlerweile eine weitverbreitete Orchestrationsplattform für Container.

Voraussetzungen

Um ein Kubernetes Cluster zu installieren und dieser Anleitung zu folgen, müssen die folgenden Voraussetzungen erfüllt sein:

- Mindestens Server mit einem installierten **Debian 11** oder **Debian** 12
- Pro Server mindestens **2 virtuelle CPU-Kerne**
- Pro Server mindestens **2 GB Arbeitsspeicher**
- Pro Server mindestens **20 GB freier Festplattenplatz**
- Administrationsbenutzer
- Stabile Internetverbindung

Installation

Netzwerkdesign

In dieser Anleitung werden wir 2 Server betreiben. Einer wird als Master Node fungieren, und der zweite Server als Worker Node. Der Master Node ist für die Verwaltung des Kubernetes Clusters zuständig. Der Worker Node führt nur die sogenannten Pods auf.

Hostbeschreibung	Hostname	IP-Adresse
Master Node	srv-kub-master	192.168.10.200
Worker Node	srv-kub-worker1	192.168.10.201

Installation

Im ersten Schritt installieren wir ein paar benötigte Pakete und deaktivieren auf jedem Node den Swap-Speicher. Dies ist zwar nicht zwingend erforderlich, aber funktioniert in der Regel besser.

```
apt install -y sudo curl socat -y
sudo swapoff -a
sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab
```

Im nächsten Schritt installieren wir die containerd Laufzeitumgebung und stellen ein paar Dinge ein. Dazu führen wir die folgenden Befehle aus. Zuerst stellen wir ein paar Kernel Parameter ein:

```
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
br_netfilter
EOF
sudo modprobe overlay
sudo modprobe br_netfilter
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-k8s.conf
net.bridge.bridge-nf-call-iptables = 1
net.ipv4.ip_forward = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF
```

Um die Änderungen jetzt zu übernehmen für wir den folgenden Befehl aus:

```
sudo sysctl --system
```

Jetzt installieren wir das Paket containerd und stellen wieder ein paar Dinge ein.

```
sudo apt update
sudo apt -y install containerd
containerd config default | sudo tee /etc/containerd/config.toml >/dev/null 2>&1
```

Im nächsten Schritt setzen wir den "*cgroupdriver*" auf allen Nodes.

```
sudo nano /etc/containerd/config.toml
```

Dort müssen wir in dem Pfad `[plugins."io.containerd.grpc.v1.cri".containerd.runetimes.runc.options]` die Option SystemdCgroup auf true verändern.

Danach starten wir den Dienst von containerd einmal neu.

```
sudo systemctl restart containerd
sudo systemctl enable containerd
```

Im nächsten Schritt fügen wir das Kubernetes Apt Repository hinzu.

```
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg] https://pkgs.k8s.io/core:/stable:/v1.28/deb/
/" | sudo tee /etc/apt/sources.list.d/kubernetes.list
curl -fsSL https://pkgs.k8s.io/core:/stable:/v1.28/deb/Release.key | sudo gpg --dearmor -o
/etc/apt/keyrings/kubernetes-apt-keyring.gpg
```

Jetzt installieren wir die Kubernetes Tools auf unseren Servern.

```
sudo apt update
sudo apt install kubelet kubeadm kubectl -y
sudo apt-mark hold kubelet kubeadm kubectl
```

Zum Testen, ob alles geklappt hat, können wir einmal `kubectl version` ausführen.

Kubernetes Konfiguration

Jetzt konfigurieren wir unseren Kubernetes Cluster und verbinden die einzelnen Hosts miteinander, um die grundlegenden Funktionen von Kubernetes zu erhalten.

Wir erstellen im ersten Schritt eine yaml Datei, welches die Konfiguration von unserem Cluster enthält und passen diese Datei unserem Belieben an.

Dieser Schritt muss nur auf dem Master Node durchgeführt werden!

```
nano kubelet.yaml
```

```
apiVersion: kubeadm.k8s.io/v1beta3
kind: InitConfiguration
---
apiVersion: kubeadm.k8s.io/v1beta3
kind: ClusterConfiguration
kubernetesVersion: "1.28.0" # Ersetzen mit deiner eingesetzen Version
controlPlaneEndpoint: "k8s-master"
---
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
```

Nachdem wir die Datei erstellt haben, initialisieren wir jetzt unser Kubernetes Cluster.

```
sudo kubeadm init --config kubelet.yaml
```

> Wenn alles geklappt hat, sollte eine Meldung auftauchen, das dass **Control-Plane** erfolgreich initialisiert wurde. Wenn dies der Fall ist, sehen wir auch die entsprechenden Befehle damit die anderen **Worker Nodes** als **Worker** oder als **Master** beitreten können. Diese Befehle können wir bei Bedarf auch neu erstellen. Für das erste die Befehle zur Seite kopieren.

Wir müssen im Anschluss noch einmal die Befehle ausführen, die benötigt werden, damit mit dem Control-Plane kommuniziert werden kann.

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Um zu testen, ob der Cluster richtig hochgefahren wurde, können wir die folgenden Befehle ausführen.

```
kubectl get nodes
kubectl cluster-info
```

Jetzt können wir auf den Worker Nodes die Befehle ausführen, um dem Kubernetes Cluster beizutreten. Wenn die Nodes beigetreten sind, sollten diese mit dem Befehl `kubectl get nodes` ersichtlich sein. Damit die Nodes im Status hochgefahren werden, brauchen wir sogenannte Netzwerk Add-ons. Wir verwenden hier Calico.

Um Calico zu installieren, führen wir den folgenden Befehl aus:

```
kubectl apply -f https://raw.githubusercontent.com/projectcalico/calico/v3.26.1/manifests/calico.yaml
```

Um zu überprüfen, ob die Calico Pods laufen, führen wir den folgenden Befehl aus:

```
kubectl get pods -n kube-system
```

Kubernetes Cluster testen

Um das Kubernetes Cluster zu testen, führen wir den folgenden Befehl auf dem Master Node aus:

```
kubectl create deployment nginx-app --image=nginx --replicas 2
kubectl expose deployment nginx-app --name=nginx-web-svc --type NodePort --port 80
kubectl describe svc nginx-web-svc
```

Jetzt sollten wir, wenn alles geklappt hat, eine Ausgabe mit dem entsprechenden Port erhalten. Das heißt, wenn wir eine Webanfrage auf die externe IP mit dem angegebenen Port starten, sollte uns die "NGINX Welcome Page" begrüßen.

Mit dem folgenden Befehl können wir das gestartete Deployment wieder stoppen und löschen:

```
kubectl delete deployment nginx-app
```

Revision #3
Created 22 October 2024 21:03:43 by Phillip U.
Updated 29 October 2024 10:16:36 by Phillip U.