ToiletPaper #104

jambit
WHERE INNOVATION WORKS

Minikube vs. Kind vs. K3s – Welches lokale Kubernetes-Cluster eignet sich am besten?

Autor: Maximilian Brenner / DevOps Engineer / Standort Leipzig

➔ Erklärung

Heutzutage gibt es ein paar Tools, die angeben, ein vollwertiges Kubernetes-Cluster (teilweise) zu ersetzen. Mit ihnen kann z. B. jeder Entwickler seine eigene lokale Cluster-Instanz laufen lassen, um damit herumzuexperimentieren, seine Anwendung zu implementieren oder Tests gegen Anwendungen durchzuführen, die während des CI/CDs in K8s laufen. In diesem Beitrag werden wir uns drei dieser Tools näher ansehen, ihre Vor- und Nachteile vergleichen und Use Cases für jedes von ihnen aufzeigen.

✔ Minikube

Minikube ist ein Kubernetes SIGs Projekt und wurde vor mehr als drei Jahren gestartet. Ansatz: Es wird eine VM erzeugt, die im Wesentlichen ein K8s-Cluster mit einer Node ist. Da es eine Reihe von Hypervisoren unterstützt, kann es auf allen wichtigen Betriebssystemen eingesetzt werden. Dies ermöglicht auch die parallele Erstellung mehrerer Instanzen.
Aus User-Sicht ist Minikube ein sehr anfängerfreundliches Tool. Man startet das Cluster mit `minikube start`, wartet ein paar Minuten und schon ist `kubectl` einsatzbereit. Um eine Kubernetes-Version zu bestimmen, kann man den Parameter `--kubernetes-version` verwenden. Eine Auflistung der unterstützten Versionen findet man hier.
Wenn man neu in Kubernetes ist, bietet Minikube einen erstklassigen Support für das Dashboard, das nützlich sein kann. Mit einem einfachen `minikube dashboard` öffnet sich die Anwendung und gibt einen guten Überblick über alles, was im Cluster passiert. Dies wird durch das Minikube Addon-System erreicht, das hilft, Dinge wie Helm, Nvidia GPUs und eine Image Registry im Cluster zu integrieren.

✔ Kind

Kind ist ein weiteres Kubernetes SIGs Projekt, das sich jedoch deutlich von Minikube unterscheidet. Wie der Name schon sagt, wird das Cluster in Docker-Container verschoben. Dies führt zu einer wesentlich schnelleren Startgeschwindigkeit im Vergleich zum Spawnen von VMs.
Das Erstellen eines Clusters ist dem Ansatz von Minikube sehr ähnlich. Wenn man `kind create cluster` ausführt, muss man kurz warten und danach kann es losgehen. Durch die Verwendung verschiedener Namen (`--name`) ermöglicht Kind die parallele Erstellung mehrerer Instanzen.
Eine Funktion, die mir persönlich sehr gefällt, ist die Möglichkeit, meine lokalen Images direkt ins Cluster zu laden. Das erspart mir einige zusätzliche Schritte, wie das Einrichten einer Registry und das ständige Hin- und Herschieben meiner Images jedes Mal, wenn ich meine Änderungen testen möchte. Mit einem einfachen `kind load docker-image my-app:latest` steht das Image für die Verwendung in meinem Cluster zur Verfügung. Sehr nice!
Wenn man nach einer Möglichkeit sucht, ein Kubernetes-Cluster programmatisch zu erstellen, hat Kind Go-Pakete zur Verwendung veröffentlicht. Um mehr darüber zu erfahren, schaut euch doch mal die dazugehörigen GoDocs an und wie KUDO Kind für deren Integrationstests verwendet.

✔ K3s

K3s ist eine von Rancher Labs entwickelte Mini-Version von Kubernetes. Durch das Entfernen überflüssiger Funktionen (Legacy, Alpha, Nicht-Standard, In-Tree-Plugins) und die Verwendung leichtgewichtigerer Komponenten (z. B. sqlite3 statt etcd3) wurde eine deutliche Verkleinerung erreicht. Das Ergebnis ist ein einziges Binary mit einer Größe von etwa 60 MB.
Die Anwendung wird in den K3s-Server und den Agent aufgeteilt. Ersterer fungiert als Manager, während letzterer sich um die eigentliche Workload kümmert. Ich rate davon ab, diese direkt auf eurem Rechner laufen zu lassen, da es zu einem gewissen Durcheinander im lokalen Dateisystem führt. Stattdessen K3s lieber in einen Container packen (z. B. den von rancher/k3s verwenden), was es außerdem ermöglicht, einfach mehrere Instanzen unabhängig voneinander laufen zu lassen.
Eine besonders erwähnenswerte Funktion ist das sogenannte Auto Deployment. Es ermöglicht, Kubernetes-Manifeste und Helm-Charts zu verteilen, indem sie in ein bestimmtes Verzeichnis gelegt werden. K3s sucht nach Änderungen und sorgt dafür, dass sie ohne weitere Interaktion angewendet werden. Dies ist besonders nützlich für CI-Pipelines und IoT-Geräte (beides Target Use Cases von K3s). Man muss lediglich die Konfiguration anlegen/anpassen und K3s sorgt dafür, dass die Deployments auf dem neuesten Stand bleiben.

➔ Fazit

Ich war lange Minikube-User, da es einfach keine Alternativen gab (zumindest habe ich nie von einer gehört). Und um ehrlich zu sein, macht Minikube einen ziemlich guten Job in der lokalen Kubernetes-Entwicklungsumgebung. Man erstellt das Cluster, wartet ein paar Minuten und schon kann es losgehen. Für meine Use Cases (meist das Herumprobieren mit Tools, die auf K8s laufen) ließe sich jedoch aufgrund der schnelleren Einrichtungszeit vollständig auf Kind umsteigen. Wenn man in einer Umgebung mit einem engen Ressourcen-Pool arbeitet oder eine noch schnellere Startzeit benötigt, sollte man K3s definitiv in Betracht ziehen.
Alles in allem machen alle drei Tools, was sie sollen, wobei sie unterschiedliche Ansätze verwenden und sich auf verschiedene Use Cases konzentrieren. Ich hoffe, du verstehst deren Funktionsweise jetzt etwas besser und weißt nun, welches das beste Tool für die Lösung deines anstehenden Problems ist. Teile gern deine Erfahrungen via @ brennerm und erzähl von eigenen Use Cases, die du mit Minikube, Kind oder K3s realisierst.

In folgender Tabelle sind noch mal einige wichtige Fakten zu jedem Tool aufgeführt:

	Minikube	**Kind**	**K3s**
Laufzeit	VM	container	native
Unterstützte Architekturen	AMD64	AMD64	AMD64, ARMv7, ARM64
unterstützte Container-Lau	Docker,CRI-O,containerd,g	Docker	Docker, containerd
Startzeit initial/danach	5:19 / 3:15	2:48 / 1:06	0:15 / 0:15
Speicherbedarf	2 GB	8 GB (Windows, Ma	512 MB
Root erforderlich?	nein	nein	ja (ohne ist gewagt)
Multi-Cluster-Unterstützun	ja	ja	nein (kann durch Container erre
Multi-Node-Unterstützung	nein	ja	ja
Webseite	Minikube	Kind	K3s