Sidecar Containers

Sidecar-Container sind sekundäre Container, die zusammen mit dem Hauptanwendungscontainer im selben Pod ausgeführt werden. Diese Container erweitern die Funktionalität des primären Anwendungscontainers durch zusätzliche Dienste oder Funktionen wie Protokollierung, Überwachung, Sicherheit oder Datensynchronisierung, ohne den primären Anwendungscode direkt zu verändern.

In der Regel befindet sich in einem Pod nur ein Anwendungscontainer. Wenn Sie beispielsweise eine Webanwendung haben, die einen lokalen Webserver benötigt, fungiert dieser als Sidecar und die Webanwendung selbst als Anwendungscontainer.

Sidecar-Container in Kubernetes

Kubernetes implementiert Sidecar-Container als Sonderfall von Init-Containern. Sidecar-Container bleiben auch nach dem Pod-Start aktiv. In diesem Dokument wird der Begriff „reguläre Init-Container“ verwendet, um Container zu bezeichnen, die nur während des Pod-Starts ausgeführt werden.

Sofern in Ihrem Cluster das Feature-Gate „SidecarContainers“ aktiviert ist (seit Kubernetes v1.29 standardmäßig), können Sie eine Neustartrichtlinie für Container im Feld „initContainers“ eines Pods festlegen. Diese neustartbaren Sidecar-Container sind unabhängig von anderen Init-Containern und den Hauptanwendungscontainern innerhalb desselben Pods. Sie können gestartet, gestoppt oder neu gestartet werden, ohne den Hauptanwendungscontainer und andere Init-Container zu beeinträchtigen.

Sie können einen Pod auch mit mehreren Containern ausführen, die nicht als Init- oder Sidecar-Container gekennzeichnet sind. Dies ist sinnvoll, wenn die Container innerhalb des Pods für dessen Gesamtfunktionalität erforderlich sind, Sie aber nicht steuern müssen, welche Container zuerst gestartet oder gestoppt werden. Sie können dies auch tun, wenn Sie ältere Versionen von Kubernetes unterstützen müssen, die kein restartPolicy-Feld auf Containerebene unterstützen.

Sidecar-Container und Pod-Lebenszyklus

Wenn ein Init-Container mit der RestartPolicy „Always“ erstellt wird, wird er gestartet und bleibt während der gesamten Lebensdauer des Pods aktiv. Dies kann hilfreich sein, um unterstützende Dienste getrennt von den Hauptanwendungscontainern auszuführen.

Wenn für diesen Init-Container eine ReadinessProbe angegeben ist, wird deren Ergebnis verwendet, um den Bereitschaftszustand des Pods zu bestimmen.

Da diese Container als Init-Container definiert sind, profitieren sie von denselben Reihenfolge- und Sequenzgarantien wie reguläre Init-Container. So können Sie Sidecar-Container mit regulären Init-Containern für komplexe Pod-Initialisierungsabläufe kombinieren.

Im Vergleich zu regulären Init-Containern werden in initContainern definierte Sidecars nach dem Start weiter ausgeführt. Dies ist wichtig, wenn in .spec.initContainers für einen Pod mehr als ein Eintrag vorhanden ist. Sobald ein Sidecar-Init-Container läuft (das Kubelet hat den Startstatus für diesen Init-

Container auf „true“ gesetzt), startet das Kubelet den nächsten Init-Container aus der geordneten .spec.initContainers-Liste. Dieser Status wird entweder dadurch erreicht, dass im Container ein Prozess ausgeführt wird und kein Starttest definiert ist, oder dadurch, dass der Starttest erfolgreich war.

Nach der Pod-Beendigung verschiebt Kubelet die Beendigung der Sidecar-Container, bis der Hauptanwendungscontainer vollständig gestoppt ist. Die Sidecar-Container werden dann in umgekehrter Reihenfolge wie in der Pod-Spezifikation heruntergefahren. Dadurch bleibt sichergestellt, dass die Sidecars betriebsbereit bleiben und andere Container im Pod unterstützen, bis ihr Dienst nicht mehr benötigt wird.

Jobs mit Sidecar-Containern

Wenn Sie einen Job mit Sidecar-Containern im Kubernetes-Stil definieren, verhindert der Sidecar-Container in jedem Pod nicht, dass der Job nach Abschluss des Hauptcontainers abgeschlossen wird.

Unterschiede zu Anwendungscontainern

Sidecar-Container laufen parallel zu Anwendungscontainern im selben Pod. Sie führen jedoch nicht die primäre Anwendungslogik aus, sondern stellen unterstützende Funktionen für die Hauptanwendung bereit.

Sidecar-Container haben eigene, unabhängige Lebenszyklen. Sie können unabhängig von Anwendungscontainern gestartet, gestoppt und neu gestartet werden. Das bedeutet, dass Sie Sidecar-Container aktualisieren, skalieren und warten können, ohne die Hauptanwendung zu beeinträchtigen.

Sidecar-Container nutzen dieselben Netzwerk- und Speicher-Namespaces wie der primäre Container. Diese gemeinsame Nutzung ermöglicht eine enge Interaktion und die gemeinsame Nutzung von Ressourcen.

Aus Kubernetes-Sicht ist die ordnungsgemäße Beendigung des Sidecar-Containers weniger wichtig. Wenn andere Container die zugewiesene Zeit für die ordnungsgemäße Beendigung ausschöpfen, erhalten die Sidecar-Container das SIGTERM-Signal, gefolgt vom SIGKILL-Signal, bevor sie ordnungsgemäß beendet werden können. Daher sind Exitcodes ungleich 0 (0 bedeutet erfolgreiches Beenden) für Sidecar-Container bei der Pod-Beendigung normal und sollten von externen Tools grundsätzlich ignoriert werden.

Unterschiede zu Init-Containern

Sidecar-Container arbeiten parallel zum Hauptcontainer, erweitern dessen Funktionalität und bieten zusätzliche Dienste.

Sidecar-Container laufen parallel zum Hauptanwendungscontainer. Sie sind während des gesamten Lebenszyklus des Pods aktiv und können unabhängig vom Hauptcontainer gestartet und gestoppt werden. Im Gegensatz zu Init-Containern unterstützen Sidecar-Container Probes zur Steuerung ihres

Lebenszyklus.

Sidecar-Container können direkt mit den Hauptanwendungscontainern interagieren, da sie wie Init-Container stets dasselbe Netzwerk nutzen und optional auch Volumes (Dateisysteme) gemeinsam nutzen können.

Init-Container werden vor dem Start der Hauptcontainer gestoppt, sodass sie keine Nachrichten mit dem Anwendungscontainer in einem Pod austauschen können. Der Datenaustausch erfolgt unidirektional (z. B. kann ein Init-Container Informationen in einem emptyDir-Volume ablegen).

Das Ändern des Images eines Sidecar-Containers führt nicht zum Neustart des Pods, sondern löst einen Container-Neustart aus.

Ressourcenfreigabe innerhalb von Containern

Angesichts der Ausführungsreihenfolge für Init-, Sidecar- und App-Container gelten folgende Regeln für die Ressourcennutzung:

- Der höchste Wert einer bestimmten Ressourcenanforderung oder eines Limits, das für alle Init-Container definiert ist, ist die effektive Init-Anforderung/das Limit. Wenn für eine Ressource kein Limit angegeben ist, wird dies als höchstes Limit betrachtet.
- Die effektive Anforderung/das Limit des Pods für eine Ressource ist die Summe aus dem Overhead des Pods und dem höheren Wert von:
  - der Summe aller Anfragen/Limits für eine Ressource von Nicht-Init-Containern (App- und Sidecar-Container)
  - der effektiven Init-Anforderung/dem Limit für eine Ressource
- Die Planung basiert auf den effektiven Anforderungen/Limits, was bedeutet, dass Init-Container Ressourcen für die Initialisierung reservieren können, die während der Lebensdauer eines Pods nicht genutzt werden.
- Die QoS-Stufe (Qualität des Dienstes) des Pods entspricht der QoS-Stufe für Init-, Sidecar- und App-Container gleichermaßen.

Quoten und Limits werden basierend auf der effektiven Pod-Anforderung und dem Limit angewendet.

Sidecar-Container und Linux-Cgroups

Unter Linux basieren Ressourcenallokationen für Pod-Level-Kontrollgruppen (Cgroups) auf der effektiven Pod-Anforderung und dem Limit, genauso wie beim Scheduler.