Probes

Ein wesentlicher Aspekt von Kubernetes ist seine Fähigkeit, den Zustand der Anwendungen und ihrer Komponenten kontinuierlich zu überwachen und entsprechend darauf zu reagieren. Dazu verwendet Kubernetes sogenannte Probes, die grundlegende Gesundheitschecks für Container durchführen können. In diesem ausführlichen Text werden wir die verschiedenen Arten von Kubernetes Probes sowie ihre Funktionen und Konfigurationen im Detail erklären.

Die Status Phasen eines Pods wärend des Starts

- Pending
- Creating
- Running

Verständnis von Kubernetes Probes

Kubernetes Probes sind Mechanismen, um den Zustand eines Containers zu überwachen. Sie helfen dabei sicherzustellen, dass Anwendungen reibungslos laufen und dass der Cluster bei Bedarf Maßnahmen ergreifen kann, um Probleme zu beheben. Es gibt drei Haupttypen von Probes, die von Kubernetes angeboten werden:

1. **Liveness Probe**
2. **Readiness Probe**
3. **Startup Probe**

Jeder dieser Probes hat einen spezifischen Einsatzzweck, der unterschiedliche Aspekte des Container-Verhaltens überprüft.

1. Liveness Probe

Die Liveness Probe dient dazu, festzustellen, ob ein Container noch am Leben oder möglicherweise in einem fehlerhaften Zustand ist, aus dem er sich nicht mehr erholen kann. Wenn eine Liveness Probe fehlschlägt, geht Kubernetes davon aus, dass der Container abgestürzt ist oder in einem Deadlock-Zustand festhängt. In solchen Fällen wird der Container automatisch neu gestartet.

Anwendungsfälle:

1. Endlosschleifen: Wenn Ihr Anwendungscontainer in eine Endlosschleife gerät und nicht mehr auf externe Anfragen reagiert.
2. Ressourcenerschöpfung: Wenn ein Container in einem Zustand verharrt, in dem er keine neuen Ressourcen bereitstellen kann.

Konfiguration:

Eine Liveness Probe kann auf verschiedene Weisen konfiguriert werden, am häufigsten durch:

1. **HTTP Probe**: Dabei sendet die Probe eine HTTP GET-Anfrage an den Container. Ein HTTP-Statuscode im 2xx Bereich zeigt an, dass der Container gesund ist.

1. **TCP Socket Probe**: Diese Prüfung stellt eine TCP-Verbindung zu einem Container her. Wenn die Verbindung hergestellt werden kann, gilt der Container als gesund.

1. **Exec Probe**: Dabei wird ein bestimmter Befehl innerhalb des Containers ausgeführt. Wenn der Befehl mit einem Exit-Code von 0 endet, gilt der Container als gesund.

Ein Beispiel für eine Liveness Probe-Konfiguration:

```
livenessProbe:
  httpGet:
    path: /health
    port: 8080
  initialDelaySeconds: 5
  periodSeconds: 10
```

In diesem Beispiel führt Kubernetes alle 10 Sekunden einen Gesundheitscheck durch, beginnend 5 Sekunden nach dem Start des Containers.

2. Readiness Probe

Readiness Probes dienen dazu, festzustellen, ob ein Container bereit ist, Anfragen zu empfangen. Anders als bei den Liveness Probes, bei denen Container bei Fehlern neustarten, verhindern Readiness Probes, dass Traffic an Container gesendet wird, die (noch) nicht bereit sind. Wenn eine Readiness Probe fehlschlägt, entfernt Kubernetes den Pod von den Service-Endpunkten, sodass er keinen Verkehr mehr empfängt.

Anwendungsfälle:

1. Langer Startvorgang: Einige Anwendungen benötigen eine Initialisierungsphase, bevor sie bereit sind, Anfragen zu bearbeiten.
2. Externe Abhängigkeiten: Eine Anwendung muss möglicherweise auf einen externen Dienst zugreifen, der nicht verfügbar ist.

Konfiguration:

Readiness Probes werden ähnlich wie Liveness Probes konfiguriert und beinhalten HTTP, TCP oder Exec Proben. Ein Beispiel für eine Readiness Probe-Konfiguration könnte so aussehen:

```
readinessProbe:
  tcpSocket:
    port: 8080
  initialDelaySeconds: 5
  periodSeconds: 10
```

In diesem Beispiel stellt die Readiness Probe alle 10 Sekunden eine TCP-Verbindung zu Port 8080 her, mit einer anfänglichen Verzögerung von 5 Sekunden.

3. Startup Probe

Startup Probes sind speziell darauf ausgelegt, Container-Anwendungen zu unterstützen, die lange Initialisierungszeiten haben. Diese Art von Probe wurde hinzugefügt, um Probleme mit Liveness Probes zu umgehen, die während langer Startvorgänge möglicherweise fehlschlagen, weil die Anwendungen einfach noch nicht bereit sind, aber auch nicht abgestürzt sind.

Anwendungsfälle:

1. Umfangreiche Initialisierungslogik: Einige Anwendungen benötigen eine umfangreiche Setup-Routine, bevor sie tatsächlich gestartet sind.
2. Alte oder komplexe Anwendungen: Solche Anwendungen haben möglicherweise ungewöhnliche Startverhalten, die nicht mit den Standardmechanismen abgedeckt sind.