Tolerierungen (Tolerations) unter Kubernetes

In Kubernetes ermöglichen `Tolerierungen`, dass Pods auf Knoten platziert werden können, die bestimmte `Taints` (Verunreinigungen) aufweisen. Sie sind das Gegenstück zu „Taints“, die auf Knoten angewendet werden, um zu verhindern, dass Pods standardmäßig auf ihnen geplant werden.

Stell dir vor, du hast einen Kubernetes-Cluster und möchtest bestimmte Knoten für spezielle Aufgaben reservieren, z.B. für GPU-intensive Workloads oder für kritische Systemkomponenten. Du möchtest verhindern, dass gewöhnliche Pods auf diesen speziellen Knoten landen und deren Ressourcen unnötig belegen. Hier kommen Taints und Tolerierungen ins Spiel.

—

Wie funktionieren Taints und Tolerierungen zusammen?

1. **Taints auf Knoten:**
Ein Administrator kann einem Knoten einen oder mehrere Taints hinzufügen. Ein Taint ist ein Schlüssel-Wert-Paar, das mit einem „Effekt“ verknüpft ist. Der Effekt bestimmt, was passiert, wenn ein Pod keine entsprechende Tolerierung für diesen Taint hat.

Beispiele für Effekte:

- NoSchedule: Pods ohne entsprechende Tolerierung werden `nicht` auf diesem Knoten geplant. Bestehende Pods bleiben jedoch bestehen.
- PreferNoSchedule: Kubernetes versucht, Pods ohne entsprechende Tolerierung `nicht` auf diesem Knoten zu planen. Es ist jedoch keine harte Einschränkung und kann unter Umständen umgangen werden.
- NoExecute: Pods ohne entsprechende Tolerierung werden `nicht` auf diesem Knoten geplant, und `bestehende Pods werden von diesem Knoten entfernt`, wenn sie keine Tolerierung haben.

Beispiel für das Hinzufügen eines Taints zu einem Knoten:

```
kubectl taint nodes node1 specialpurpose=true:NoSchedule
```

Dieser Befehl fügt dem Knoten `node1` den Taint `specialpurpose=true` mit dem Effekt `NoSchedule` hinzu.

2. **Tolerierungen in Pods:**
Um zu erlauben, dass ein Pod auf einem Knoten mit einem bestimmten Taint geplant wird, musst du dem Pod entsprechende Tolerierungen hinzufügen. Eine Tolerierung spezifiziert, welche Taints der Pod tolerieren kann.
Beispiel für Tolerierungen in einer Pod-Definition (YAML):

```
apiVersion: v1
```

```
kind: Pod
metadata:
  name: my-special-pod
spec:
  containers:
  - name: my-container
    image: my-image
  tolerations:
  - key: "specialpurpose"
    operator: "Equal"
    value: "true"
    effect: "NoSchedule"
```

In diesem Beispiel hat der Pod `my-special-pod` eine Tolerierung für den Taint `specialpurpose=true` mit dem Effekt `NoSchedule`. Das bedeutet, dieser Pod kann auf Knoten geplant werden, die diesen spezifischen Taint haben.

Wichtige Felder einer Tolerierung:

- `key`: Der Schlüssel des Taints, den die Tolerierung behandelt.
- `operator`: Bestimmt, wie der Schlüssel und Wert des Taints abgeglichen werden.
  - `Exists`: Der Pod toleriert jeden Taint mit dem angegebenen `key`, unabhängig vom `value`. (In diesem Fall darf `value` nicht angegeben werden).
  - `Equal`: Der Pod toleriert den Taint, wenn sowohl `key` als auch `value` exakt übereinstimmen.
- `value`: Der Wert des Taints, wenn `operator` auf `Equal` gesetzt ist.
- `effect`: Optional. Der Effekt des Taints, den die Tolerierung behandeln soll. Wenn nicht angegeben, toleriert der Pod Taints mit dem angegebenen Schlüssel und Wert für `alle` Effekte.

—

Anwendungsfälle für Taints und Tolerierungen

- **Dedizierte Knoten:** Reserviere bestimmte Knoten für spezifische Anwendungen (z.B. Datenbanken, Machine Learning).
- **Knoten mit spezieller Hardware:** Markiere Knoten mit GPUs oder anderen speziellen Hardwarekomponenten, damit nur Pods, die diese Ressourcen benötigen, darauf geplant werden.
- **Problemknoten:** Isoliere Knoten, die vorübergehend Probleme haben oder gewartet werden müssen, und verschiebe Pods von ihnen weg (mit `NoExecute`).
- **Mandantenfähigkeit:** Trenne Workloads verschiedener Teams oder Projekte auf dedizierten Knotengruppen.

—

Unterschied zu Node Affinity/Anti-Affinity

- **Taints und Tolerierungen:** Sind eine `negativ orientierte` Regel. Sie verhindern

standardmäßig die Platzierung von Pods, es sei denn, der Pod hat eine explizite Tolerierung.
- **Node Affinity/Anti-Affinity:** Sind eine `positiv orientierte` Regel. Sie drücken eine Präferenz oder eine harte Anforderung aus, Pods auf bestimmten Knoten zu platzieren oder sie davon fernzuhalten.

Man kann beide Mechanismen auch kombinieren, um eine sehr detaillierte Steuerung der Pod-Platzierung zu erreichen.