Configure Taints and Tolerants.

Understanding Taints:

In Kubernetes, a "taint" is a key-value pair associated with a node that marks the node to repel certain pods. Taints are used to influence the scheduling decisions made by the Kubernetes scheduler. When a node is tainted, only pods with matching tolerations can be scheduled onto that node.

Key & Lock formula

Let's use an analogy of a key and lock for a house to explain taints and tolerations in Kubernetes.

Imagine you have a house (which represents a Kubernetes node), and you want to control who can enter and use it. In this analogy:

1. **Taints**: Taints act like locks on your house's doors. You can install different types of locks on each door to restrict access. For example, you might have a lock that requires a specific key or code to open. Similarly, in Kubernetes, taints are applied to nodes to restrict which pods can be scheduled onto them. Each taint has a key (the taint itself) that prevents pods without the corresponding toleration from being scheduled on the node.

2. **Tolerations**: Tolerations act like keys that allow certain pods to bypass the locks (**taints**) and enter the house (**node**). When you have a key that matches the lock on a door, you can enter the house without any issues. Similarly, in Kubernetes, pods can specify tolerations that match the taints on nodes. These tolerations act as keys that allow the pods to be scheduled on nodes despite the taints.

So, in summary:

- Taints are like locks on the doors of your house, restricting access to certain pods (visitors).
- Tolerations are like keys that pods (visitors) can carry to bypass the locks (taints) and be scheduled on nodes.

Taint Key Concepts

Taint Effect:

Determines how the taint affects pod scheduling on the node.

- **NoSchedule**: Pods will not be scheduled onto the tainted node unless they have a matching toleration.
- **PreferNoSchedule**: Scheduler tries to avoid scheduling pods onto the tainted node but can do so if necessary.
- **NoExecute**: Existing pods on the node without matching tolerations are evicted.

Taint Syntax:

```
kubectl taint nodes <node-name> key1=value1:<taint-effect>
```

A taint is defined with three components:

- **Key**: A string that identifies the taint.

- **Value**: An optional string value for the taint.
- **Effect**: Specifies the effect of the taint from the above mentioned.

Syntax to Remove a Taint:

To remove the taint added by the command above, you can run:

```
kubectl taint nodes <node-name> key1=value1:<taint-effect>-
```

It is same as adding a taint but at the end we are adding a hyphen“**-**”.

So using the above concepts we can add a taint to a node, now let’s see how we can bypass these taints using tolerations.

Tolerations Key Concepts::

“**Tolerations**” are mechanisms used by pods to tolerate (or accept) certain taints on nodes. When a pod has a toleration matching the taint of a node, it can be scheduled on that node.

Toleration Syntax:

A toleration is defined with three components:

- **Key**: The key of the taint to tolerate.

- **Value**: The value of the taint to tolerate (optional).
- **Effect**: Specifies the effect of the taint to tolerate.

```
tolerations:
- key: "key1"
  operator: "Equal"
  value: "value1"
  effect: "NoSchedule"
```

Toleration Operators:

- **Equal**: Requires an exact match for the taint key and value.
- **Exists**: Requires only the presence of the taint key, irrespective of its value.
- **Exists with an Effect**: Requires only the presence of a taint with a specific effect, irrespective of its key and value.

Example

In this example, the nginx pod has a toleration that matches the taint with `nginx-node=true` and `NoSchedule` effect, allowing it to be scheduled on nodes with that taint.

Let’s first add a taint to a node using the below command

```
kubectl taint nodes <node-name> nginx-node=true:NoSchedule
kubectl taint nodes pool-448noeajl-o5sgi nginx-node=true:NoSchedule
node/pool-448noeajl-o5sgi tainted
```

With above command the node is tainted with nginx-node=true:NoSchedule. Now let’s create a nginx pod with out any tolerations and see if it schedule on any node.

nginx-pod.yaml

When you try to create this pod in your Kubernetes cluster, it won’t be placed on the target node and it will be in pending state since we have added taint to the worker node.

```
$ k apply -f nginx-pod.yaml
pod/nginx created

$ k get po
NAME    READY   STATUS    RESTARTS   AGE
nginx   0/1     Pending   0          3s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  102s  default-scheduler  0/1 nodes are available: 1 node(s) had untolerated taint {nginx-node: true}. preemption: 0/1 nodes are available: 1 Preemption is not helpful for scheduling..
  Normal   NotTriggerScaleUp  82s  cluster-autoscaler  pod didn't trigger scale-up:
```

Now modify the nginx pod manifest by adding tolerations to it same as a below:

Let’s apply the above manifest file and see if it schedule on target node.

```
$ kubectl apply -f nginx-pod.yaml
pod/nginx created
```

Now the pod is schedule on target worker node

```
kubectl get pod -o wide
NAME    READY   STATUS    RESTARTS   AGE   IP            NODE                NOMINATED
NODE   READINESS GATES
nginx   1/1     Running   0          22s   10.244.0.27   pool-448noeajl-o5sgi   <none>           <none>
```

Finally we have successfully implemented Taints and Tolerations.

Use Cases and Best Practices:

1. **Isolating Critical Workloads**: Taint critical nodes and add tolerations to critical pods to ensure they run only on these nodes.
2. **Dedicated Nodes for Specialized Tasks**: Taint nodes with hardware accelerators or specialized hardware and add tolerations to pods requiring those resources.
3. **Node Maintenance**: Use taints to gracefully evacuate nodes for maintenance by tainting them with “NoExecute” effect and add tolerations to workload pods to allow them to drain safely.

If you found this blog insightful and are eager to delve deeper into topics like AWS, cloud strategies, Kubernetes, or anything related, I’m excited to connect with you on LinkedIn. Let’s spark meaningful conversations, share insights, and explore the vast realm of cloud computing together.

Feel free to reach out, share your thoughts, or ask any questions. I look forward to connecting and growing together in this dynamic field!

Happy deploying! 🚀

Happy Kubernetings! 🚀