SCHEDULING

Scheduling

Pod Affinity Rules

Pods which may communicate a lot or share data may operate best if co-located, which would be a form of affinity. For greater fault tolerance, you may want Pods to be as separate as possible, which would be anti-affinity. These settings are used by the scheduler based on the labels of Pods that are already running. As a result, the scheduler must interrogate each node and track the labels of running Pods. Clusters larger than several hundred nodes may see significant performance loss. Pod affinity rules use **`In`**, **`NotIn`**, **`Exists`**, and **`DoesNotExist`** operators.

Click on the boxes to learn more about Pod affinity rules

Pod Affinity Rules

Close

requiredDuringSchedulingIgnoredDuringExecution

The use of **`requiredDuringSchedulingIgnoredDuringExecution`** means that the Pod will not be scheduled on a node unless the following operator is true. If the operator changes to become false in the future, the Pod will continue to run. This could be seen as a hard rule.

Close

preferredDuringSchedulingIgnoredDuringExecution

Similarly, **`preferredDuringSchedulingIgnoredDuringExecution`** will choose a node with the desired setting before those without. If no properly-labeled nodes are available, the Pod will execute anyway. This is more of a soft setting, which declares a preference instead of a requirement.

podAffinity

Close

With the use of **`podAffinity`**, the scheduler will try to schedule Pods together.

podAntiAffinity

Close

The use of **`podAntiAffinity`** would cause the scheduler to keep Pods on different nodes.