62. Node affinity

> The primary purpose of "node affinity" feature is to ensure that pods are hosted on particular nodes.

> we've done this previously using "Node selectors"
The challenge in "Node Selector" is that we cannot provide advanced expressions like [OR] or [NOT] as node selectors.

- Large OR Medium
- NOT Small

Node-selector pod defination file

```
apiVersion:
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
  - name: data-processor
    image: data-processor
  nodeSelector:
    size: Large
```

```
- Large OR Medium
- NOT small
```

⇒ The "Node affinity" feature provide us with advanced capabilities to limit the pod placement on specific nodes.

So the simple "node selector" specification will now look like this with "node affinity" although both does exactly the same thing

⇒ Place the pod on Large node.

```
apiVersion:
Kind:
metadata:
  name: myapp-pod
spec:
  Containers:
    - name: data-processor
      image: data-processor
  affinity:
    nodeAffinity:
      {requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - Key: size
            Operator: In
            Values:
            - Large
```

Node affinity type

"Required" means the node affinity values in "pod definition file" should match with node label if not then pod will not be scheduled on the node

The `operator: In` ensures the pod will be placed on a node, whose label size has any value in the list of values specified here. In this case, it is just "large"

If you think, your pod should be placed on a "large or medium node", you could simply add the value to the list of values like this

Similar like left side copy the same # in the "affinity" under "values", add list of values - large, - Medium.

```
affinity:
  nodeAffinity:
  requiredDuringSchedulingIgnoredDuringExecution:
    nodeSelectorTerms:
      - matchExpressions:
        - key: size
          operator: In
          values:
          - Large
          - Medium
```

(nodeAffinity type)

you could use `not in` operator to say something like `size not in Small`, where 'node affinity' will match the nodes with a size not set to small.

```
nodeSelectorTerms:
- matchExpressions:
  -key: size
   operator: NotIn
   values:
   - small
```

we know that we have only set the label size to large and medium nodes. The smaller nodes don't even have the label set, so we don't really have to even check the value of the label. As long as we're sure we don't set a label size to the smaller node, we the exists operator will give us the same result. The Exist operator will check if the label size exists on the nodes, & you don't need the Value section

for that as it doesnot compare the values.

[There are other operators as well for which we've to check the documentation]

```
nodeSelectorTerms:
- matchExpressions:
    - key: size
      operator: Exists
```

⇒ when the pods are created, these rules are considered and the pods are placed onto the right nodes

1. But what if "node affinity" could not match a node with a given expression, In this case what if there is no node with the label called size?

2. Say we have the labels and the pods are scheduled, what if someone changes the label on the node at a future point in time?

will the pod will still lay on the node??

This is answered by a property under node affinity which is called "node affinity type"

Node Affinity Types:-

Available:-
→ if the affinity rule matches then the pod will be scheduled, else the scheduler won't

1) requiredDuringSchedulingIgnoredDuringExecution
2) preffered → scheduler schedules the pods irrespective of whether it matches the affinity rules or not DuringSchedulingIgnoredDuringExecution

planned:-

requiredDuringSchedulingRequiredDuringExecution

The type of node affinity defines the behaviour of the Scheduler with respect to node affinity & the stages in the life cycle of the pod

There are 2 states in lifecycle of a pod when considering node affinity.

1) during Scheduling
2) During Execution.

During Scheduling

⇒ It is the state where a pod does not exists & is created for the first time

⇒ We have no doubt that when a pod is first created the affinity rules specified are considered to place the pods on the right nodes

what if with matching labels are not available

For eg: we forgot to label the node as large, that is, where the type of node affinity used comes into play

If you select,

requiredDuringScheduling IgnoredDuring Execution,

⇒ the scheduler will mandate that the pod be placed on a node with a given affinity rules, if it cannot find one, the pod will not be scheduled

⇒ This is used in the case, where the placement of the pod is crucial, if the matching node doesnot exists the pod will not be scheduled

⇒ If the pod placement is less important than running the workload itself, in that case, you could set it to "preferred"

⇒ And in cases where a matching node is not found, the Scheduler will simply ignore "node affinity" rules & place the pod on any available node.

The 2nd part of property is during "Execution"

⇒ During execution is the state where a pod has been already running & a change is made in the environment that affects node affinity

Such as

* a change in the label of a node

For eg:-

An administrator removed the label we set earlier called "size = large" from the node.

Now what would happen to the pods that're running on the node? as you can see the 2 types of node affinity has this value set to ignored, which means, pods will continue to run and any changes in node affinity will not impact them once they are scheduled

	During Scheduling	During Execution
Type 1	Required	Ignored
Type 2	Preferred	Ignored

requiredDuringSchedulingRequiredDuringExecution

⇒ The new type expected in the future will only have a difference in the during execution phase.

⇒ A new option called required during execution is introduced which will evict any pods that are running on nodes that do not meet affinity rules.

⇒ In the earlier example, a pod running on the large node will be evicted or terminated if the label large is removed from the node