Kubernetes
#6. Advanced Scheduling

조대협 (http://bcho.tistory.com)

Agenda

- Taint & toleration
- Node affinity
- Pod Affinity
- TopologyKey

Taint and toleration

Taints

- It allows rejecting deployment of pods to certain node by adding taints to node. (ex : Master node)
- Format : <key>=<value>:<effect>
- Taints effect
  - NoSchedule: Pod won't be scheduled to the node if they don't tolerate the taint
  - PreferNoSchedule: It is soft version of Noschedule, meaning the scheduler will try to avoid scheduling the pod to the node, but will schedule it to the node if it can't schedule it somewhere else
  - NoExecute: Pod is evicted from the node if it is already running on the node, and is not scheduled onto the node if it is not yet running on the node.
    - **tolerationSeconds :** if this pod is running and a matching taint is added to the node, then the pod will stay bound to the node for 3600 seconds, and then be evicted. If the taint is removed before that time, the pod will not be evicted. (Toration의 효과를 적용 받는 기간, 이 기간이 지나면, toration 효과가 없어지고, 해당 Pod는 evit/제거 된다.)

Taint and toleration

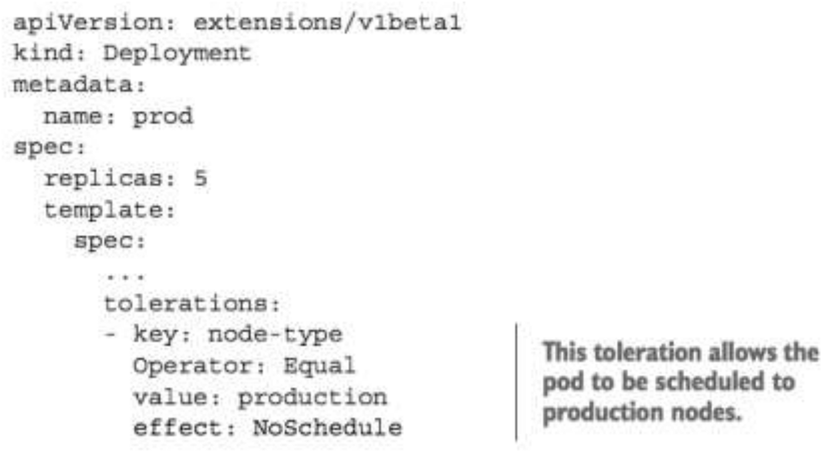
Taints

- Command
    - `kubectl taint nodes [NODE_NAME] [KEY]=[VALUE]:[EFFECT]`
    - `kubectl taint nodes node1 key=value:NoSchedule`
    - `kubectl taint node -l myLabel=X dedicated=foo:PreferNoSchedule`

Taint and toleration

Toleration

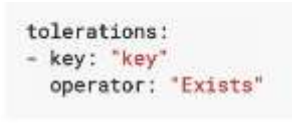
- It is applied to Pod and allows the pods to be deployed to node with matching taints



```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: prod
spec:
  replicas: 5
  template:
    spec:
      ...
      tolerations:
      - key: node-type
        Operator: Equal
        value: production
        effect: NoSchedule
```

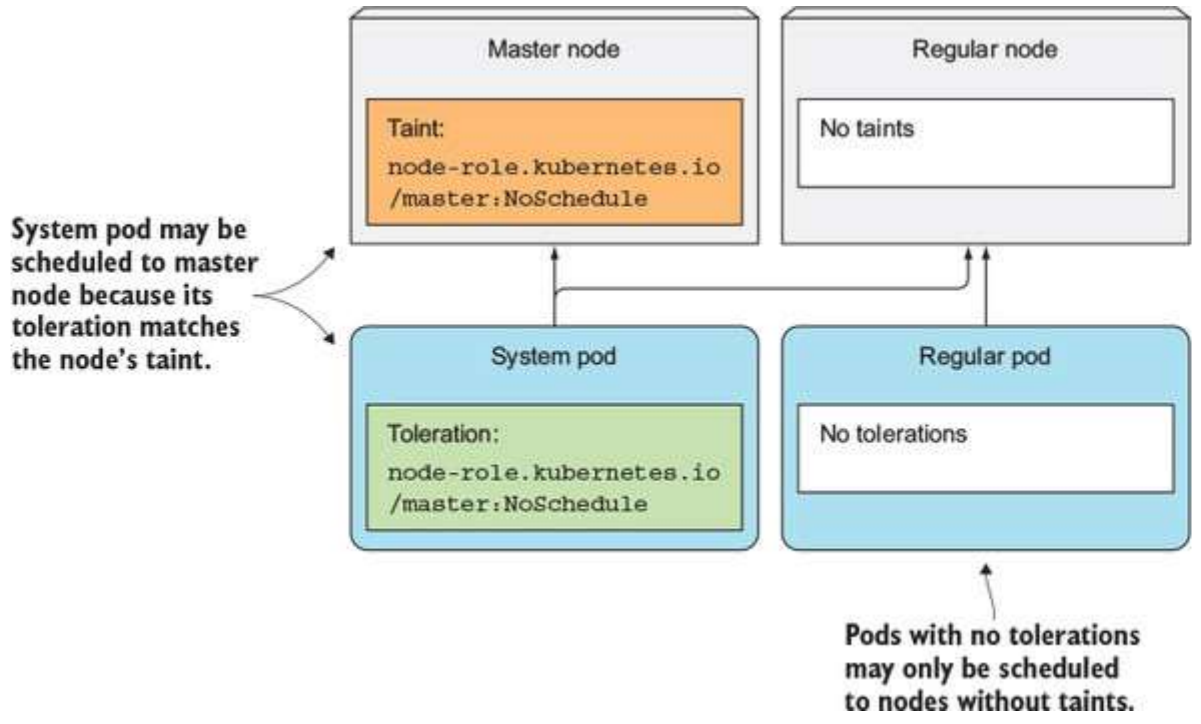
This toleration allows the pod to be scheduled to production nodes.

- Exceptional case



```
tolerations:
- key: "key"
  operator: "Exists"
```

Taint and toleration



Source : https://livebook.manning.com/#!/book/kubernetes-in-action/chapter-16/section-16-3-1

Node affinity

- cf. node selector : it specifies that pod should only be deployed on nodes that matched label (Hard affinity)
- Affinity
  - Provide scheduling affinity to node based on label
  - Compared to node selector
    - It can provide “soft/preference” based affinity. (cf. node selector is hard affinity = must)
    - Label selection is more expressive (not just “AND of exact match”)

Node affinity

From : https://kubernetes.io/docs/concepts/configuration/assign-pod-node/#affinity-and-anti-affinity

```
pod-with-node-affinity.yaml docs/concepts/configuration
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/e2e-az-name
            operator: In
            values:
            - e2e-az1
            - e2e-az2
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: another-node-label-key
            operator: In
            values:
            - another-node-label-value
  containers:
  - name: with-node-affinity
    image: k8s.gcr.io/pause:2.0
```

Node affinity (beta in 1.10)

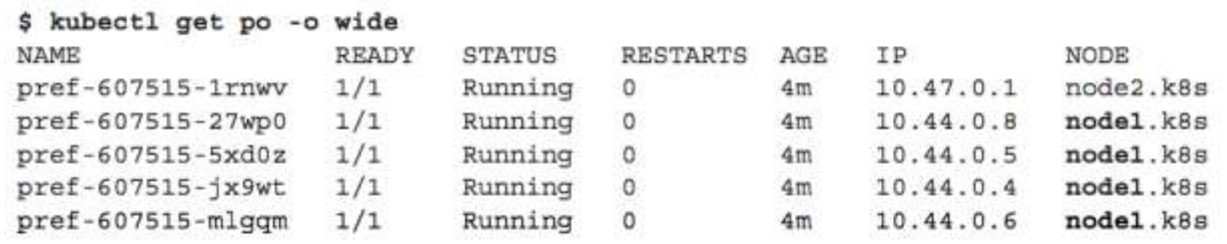
- **Hard affinity : requiredDuringSchedulingIgnoredDuringExecution**
  Same as node selector (but more expressive node selection syntax)
- **Soft affinity : preferredDuringSchedulingIgnoredDuringExecution**
  Try to deploy pod to node that matches selector but if it is not possible the deploy it elsewhere

Label key in node is "kubernetes.io/e2e-az-name" and whose value is either e2e-az1 or e2e-az2
In addition, among nodes that meet that criteria, nodes with a label whose key is another-node-label-key and whose value is another-node-label-value should be preferred.

Node affinity

Node affinity & Selector-SpreadPriority

- If you create 5 pods with affinity in 2 node cluster, all 5 pods should be created in an node



```
$ kubectl get po -o wide
NAME                READY   STATUS    RESTARTS   AGE   IP          NODE
pref-607515-1rnwv   1/1     Running   0          4m    10.47.0.1   node2.k8s
pref-607515-27wp0   1/1     Running   0          4m    10.44.0.8   node1.k8s
pref-607515-5xd0z   1/1     Running   0          4m    10.44.0.5   node1.k8s
pref-607515-jx9wt   1/1     Running   0          4m    10.44.0.4   node1.k8s
pref-607515-mlgqm   1/1     Running   0          4m    10.44.0.6   node1.k8s
```

But 1 of 5 pod is created in node 2.
The reason is that besides the node affinity prioritization function, the Scheduler also uses other prioritization functions that to decide where to schedule Pod. Selector-SpreadPriority function, which makes sure pods belonging to the same ReplicaSet are spread around different nodes so a node failure won’t bring the wole service down.
(Reference : Kubernetes in Action/Manning Chapter 16 page 468)

Pod affinity

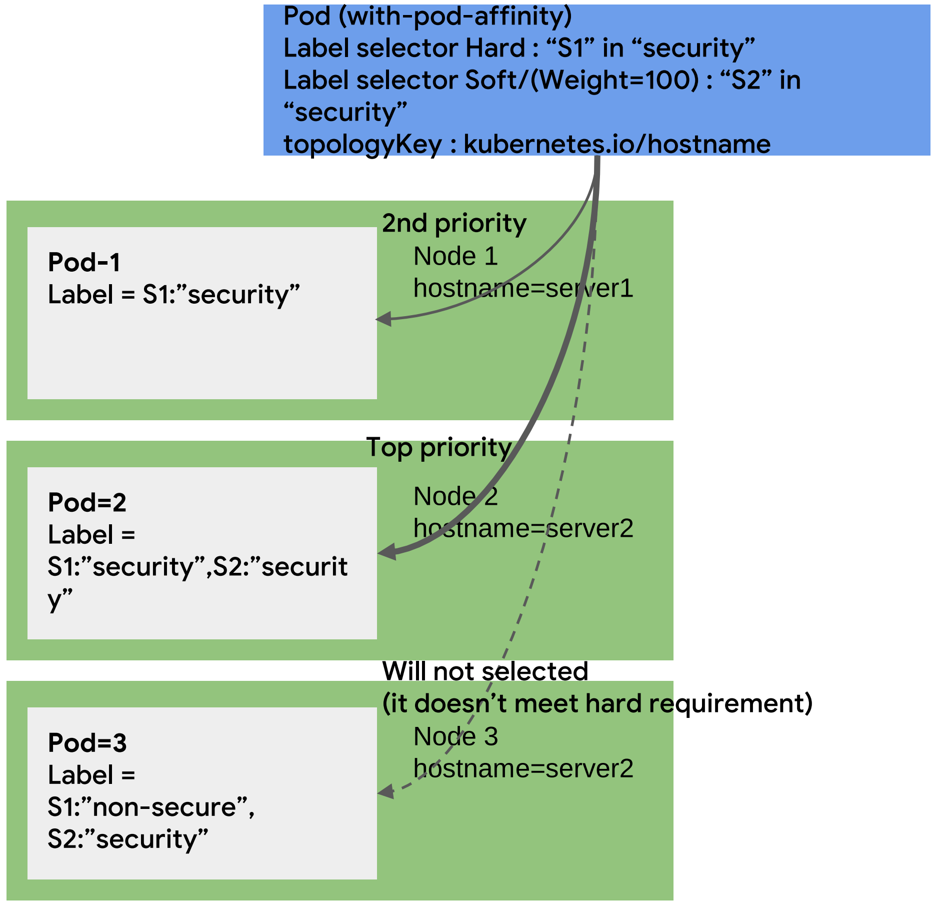
Inter pod affinity (beta in 1.10)

- Node affinity : affinity between pod and node
- Pod affinity : give affinity between Pods themselves based on label of pod which is already running (Same node or Different node)
- Like node affinity , it has two affinity (Hard/Soft)
    - (Hard) requiredDuringSchedulingIgnoredDuringExecution
    - (Soft) preferredDuringSchedulingIgnoredDuringExecution
- It needs to specify topologyKey and labelSelector
- Use case
    - Run db pod in same rack of backend server
    - Run front server in same zone of backend server
    - Run clustered instances in different nodes

Pod affinity

Inter pod affinity example

```
apiVersion: v1
kind: Pod
metadata:
 name: with-pod-affinity
spec:
 affinity:
  podAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
   - labelSelector:
     matchExpressions:
     - key: security
      operator: In
      values:
      - S1
    topologyKey: failure-domain.beta.kubernetes.io/zone
  podAffinity:
   preferredDuringSchedulingIgnoredDuringExecution:
   - weight: 100
    podAffinityTerm:
     labelSelector:
      matchExpressions:
      - key: security
       operator: In
       values:
       - S2
     topologyKey: kubernetes.io/hostname
 containers:
 - name: with-pod-affinity
  image: k8s.gcr.io/pause:2.0
```

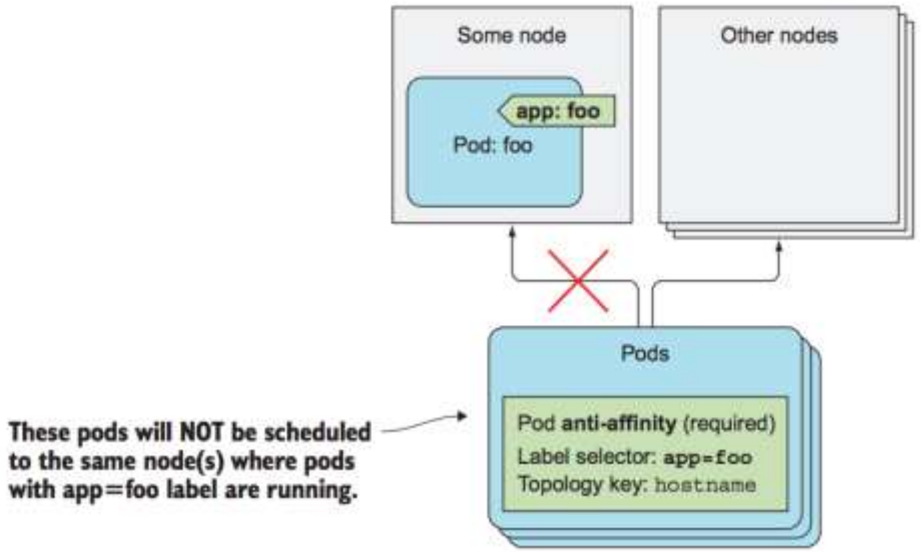

Pod (with-pod-affinity)
Label selector Hard : “S1” in “security”
Label selector Soft/(Weight=100) : “S2” in “security”
topologyKey : kubernetes.io/hostname
Pod-1
Label = S1:”security”
2nd priority
Node 1
hostname=server1
Top priority
Pod=2
Label = S1:”security”,S2:”security”
Node 2
hostname=server2
Will not selected
(it doesn’t meet hard requirement)
Pod=3
Label = S1:”non-secure”, S2:”security”
Node 3
hostname=server2

Pod affinity

Pod Anti-Affinity

Deploy pod with different place. Use *podAntiAffinity* instead of *podAffinity*

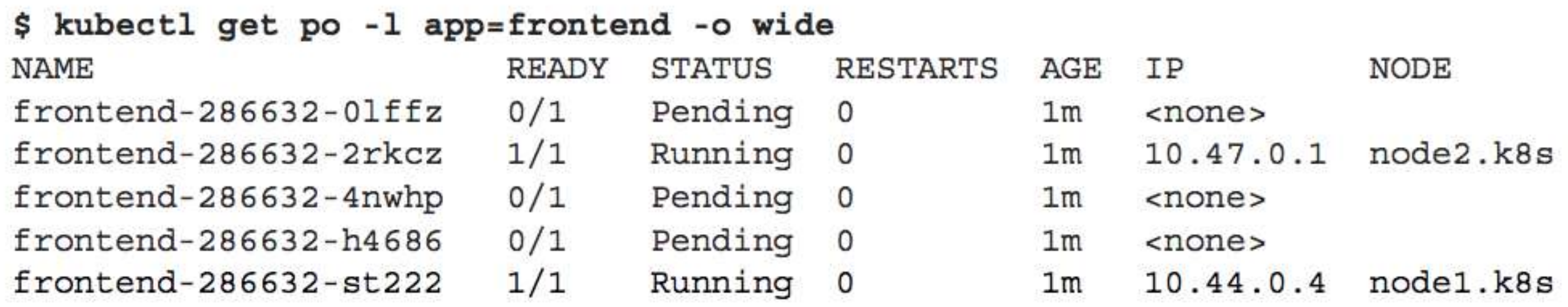
```
apiVersion: extensions/v1beta1
 kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 5
  template:
    ...
     spec:
       Affinity:
         podAntiAffinity:

requiredDuringSchedulingIgnoredDuringExecution:
              - topologyKey: kubernetes.io/hostname
                labelSelector:
                   matchLabels:
                           app: backend
```


Some node
app: foo
Pod: foo
Other nodes
Pods
Pod anti-affinity (required)
Label selector: app=foo
Topology key: hostname
These pods will NOT be scheduled to the same node(s) where pods with app=foo label are running.

Ref : Kubernetes in action book 475

Pod affinity

Pod Anti-Affinity
As you can see, only two pods were scheduled—one to node1, the other to node2. The three remaining pods are all Pending, because the Scheduler isn't allowed to schedule them to the same nodes.



```
$ kubectl get po -l app=frontend -o wide
NAME                     READY   STATUS    RESTARTS   AGE   IP          NODE
frontend-286632-0lffz    0/1     Pending   0          1m    <none>
frontend-286632-2rkcz    1/1     Running   0          1m    10.47.0.1   node2.k8s
frontend-286632-4nwhp    0/1     Pending   0          1m    <none>
frontend-286632-h4686    0/1     Pending   0          1m    <none>
frontend-286632-st222    1/1     Running   0          1m    10.44.0.4   node1.k8s
```

Ref : Kubernetes in action book
475

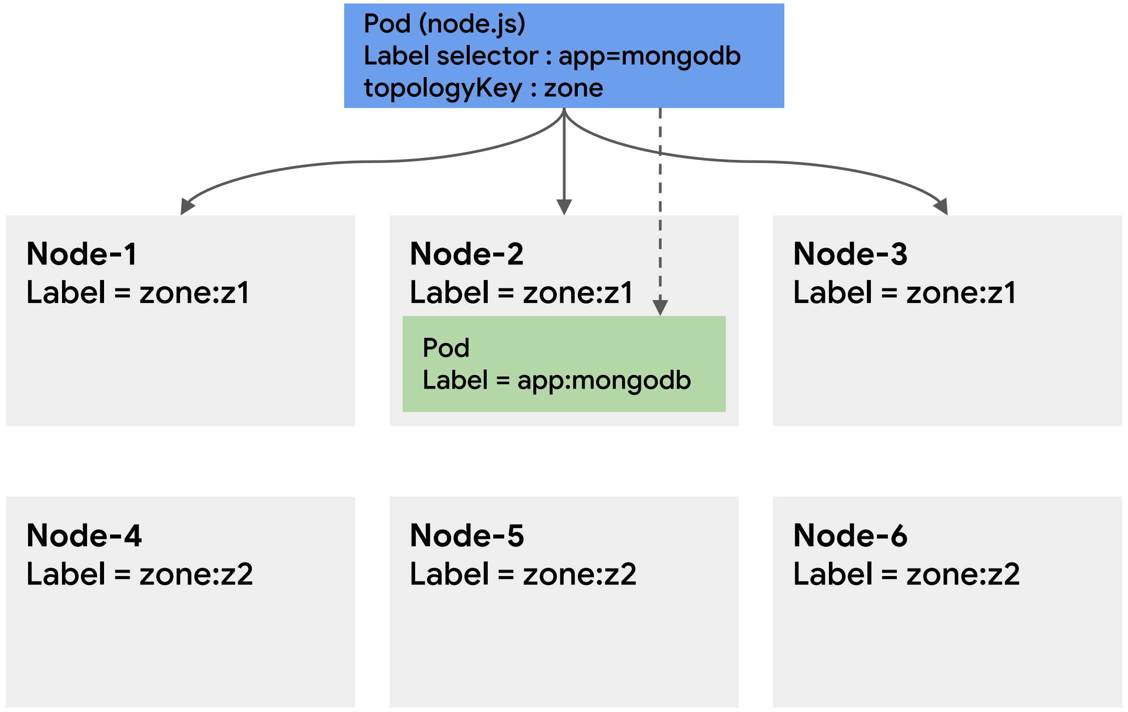
Pod affinity & topologyKey

topologyKey

- Key for node label
- Node affinity is just used to select same node. Pod affinity can be used to select same node with the pod. TopologyKey extends place concept like from same node to same rack, same zone, same region.
- How it works
  When the scheduler is deciding where to deploy a pod based on affinity setting, it find out *node* based on affinity and get topologyKey value from the node. And the pod will be deployed to one of nodes that has the matched “toplogyKey” value
  (Pod 배포시 Affinity에 의해서 배포될 노드를 먼저 계산하고, 그 노드의 topologyKey를 얻은 후에, 그 toplogyKey에 해당하는 라벨을 가진 노드에 배포한다.)

Pod affinity & topologyKey

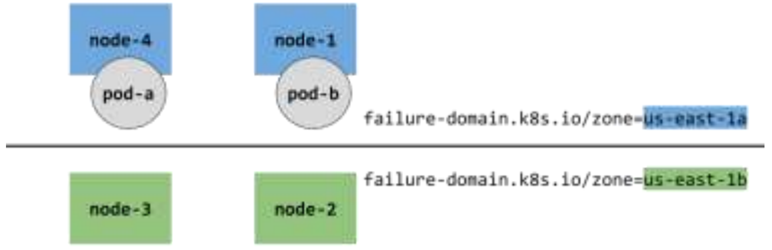
topologyKey example

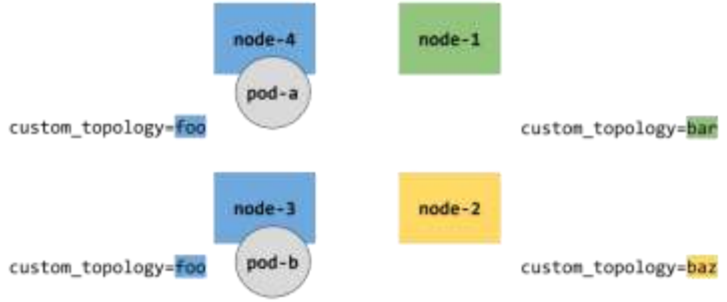


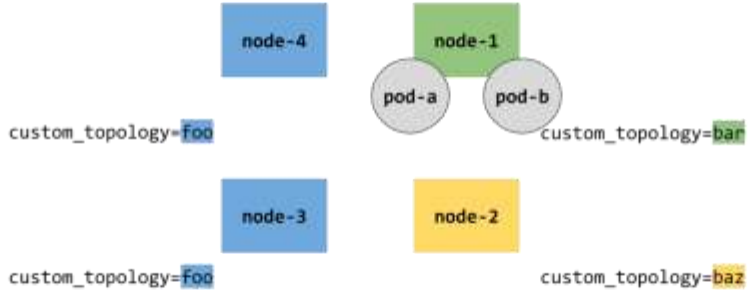
Assume that pod(node.js) has affinity with label selector app=mongodb. The matched pod is running on Node-2. pod(node.js) has "zone" as a topologykey. So pod(node.js) get the zone value "z1" from Node-2.
Based on the value, pod2 will select one of node which has zone=z1 (Node-1,Node-2,Node-3)

Pod affinity & topologyKey

topologyKey Examples


node-4
pod-a
node-1
pod-b
failure-domain.k8s.io/zone=us-east-1a
failure-domain.k8s.io/zone=us-east-1b
node-3
node-2





Reference : https://medium.com/kokster/scheduling-in-kubernetes-part-2-pod-affinity-c2b217312ae1

End of document