K8S 44: Kubernetes. Manifests. (Anti)Affinity

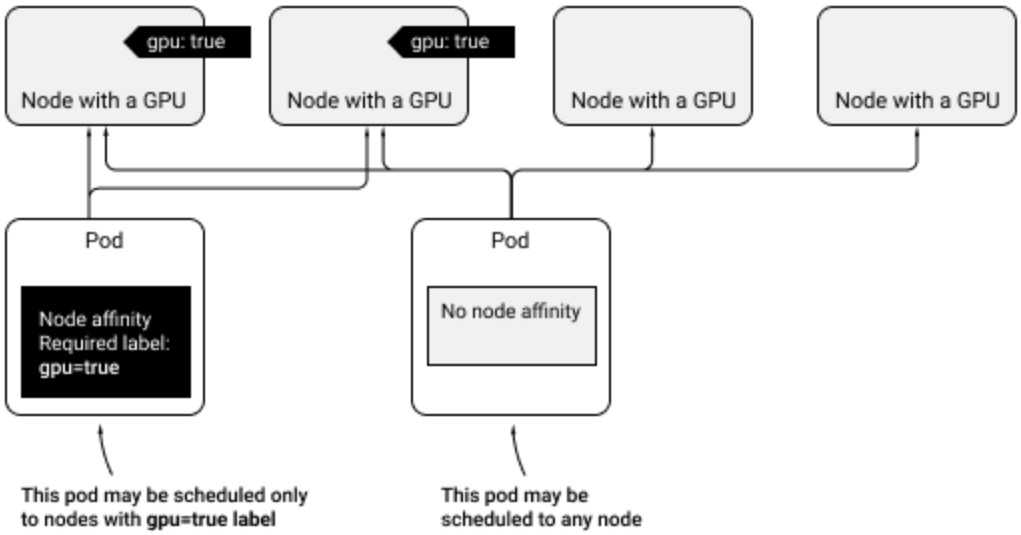
Описание:

nodeAffinity похож на nodeSelector, но работает немного иначе. Этот алгоритм более гибкий и использует метки узла для выбора. Например, у вас есть узлы с меткой gpu=true, а также с автоматическими метками failure-domain.beta.kubernetes.io/region, failure-domain.beta.kubernetes.io/zone, kubernetes.io/hostname (в облаках добавляются автоматически). Простой nodeSelector: gpu-true на Affinity будет выглядеть так:

```
apiVersion: v1
kind: Pod
metadata:
  name: kubia-gpu
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
            - key: gpu
              operator: In
              values:
              - "true"
```

Сейчас используется два типа affinity, которые можно комбинировать:

- requiredDuringSchedulingIgnoredDuringExecution — указывает, какие метки должен содержать узел для попадания в список shcgeduler при распределении pod.
- preferredDuringSchedulingIgnoredDuringExecution — описывает правила, которые не влияют на текущие запущенные pod.

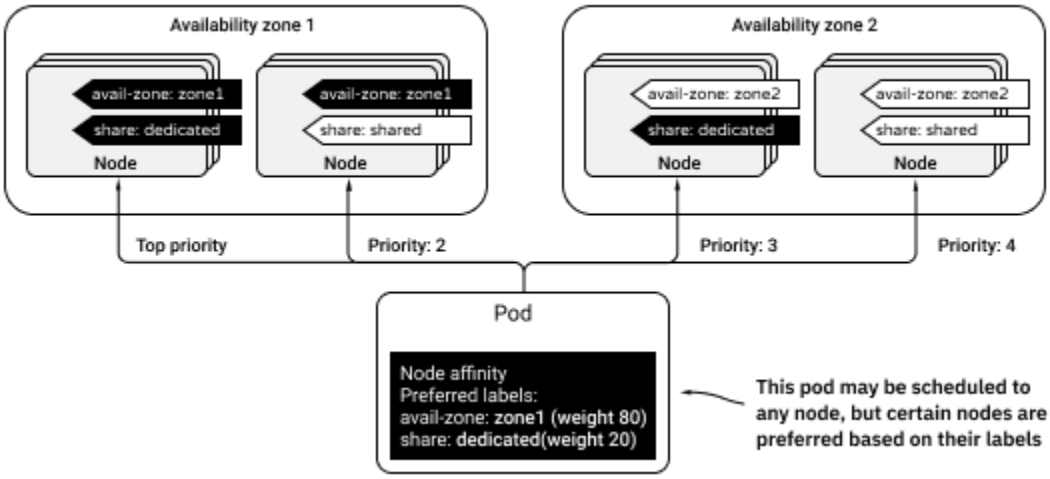


И даже более того, можно дополнительно управлять «весом» узла в алгоритме балансировки. Например:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: pref
spec:
  template:
    ...
    spec:
      affinity:
        nodeAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 80
            preference:
              matchExpressions:
              - key: availability-zone
                operator: In
                values:
                - zone1
          - weight: 20
            preference:
              matchExpressions:
              - key: share-type
                operator: In
                values:
```

```
            - dedicated
```

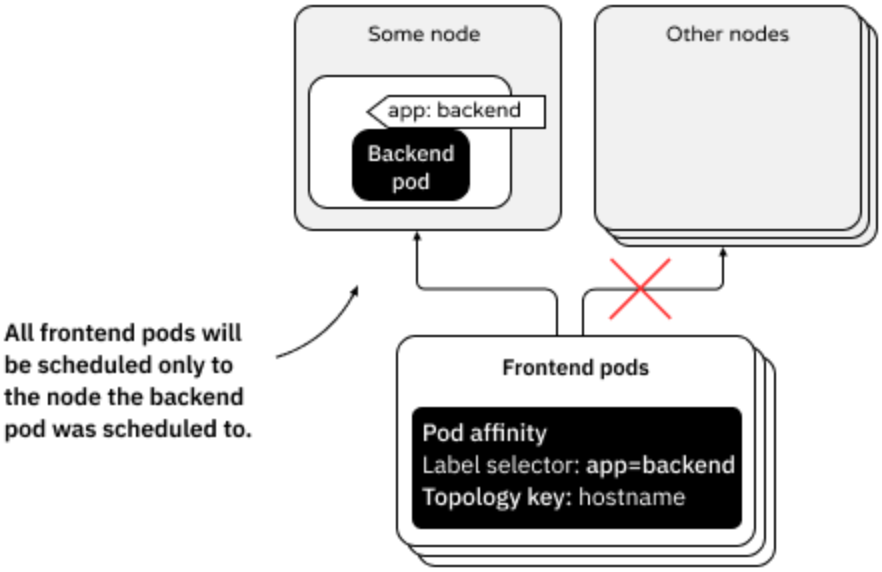
Данное описание означает, что приоритет получают узлы с меткой zone и меткой dedicated.



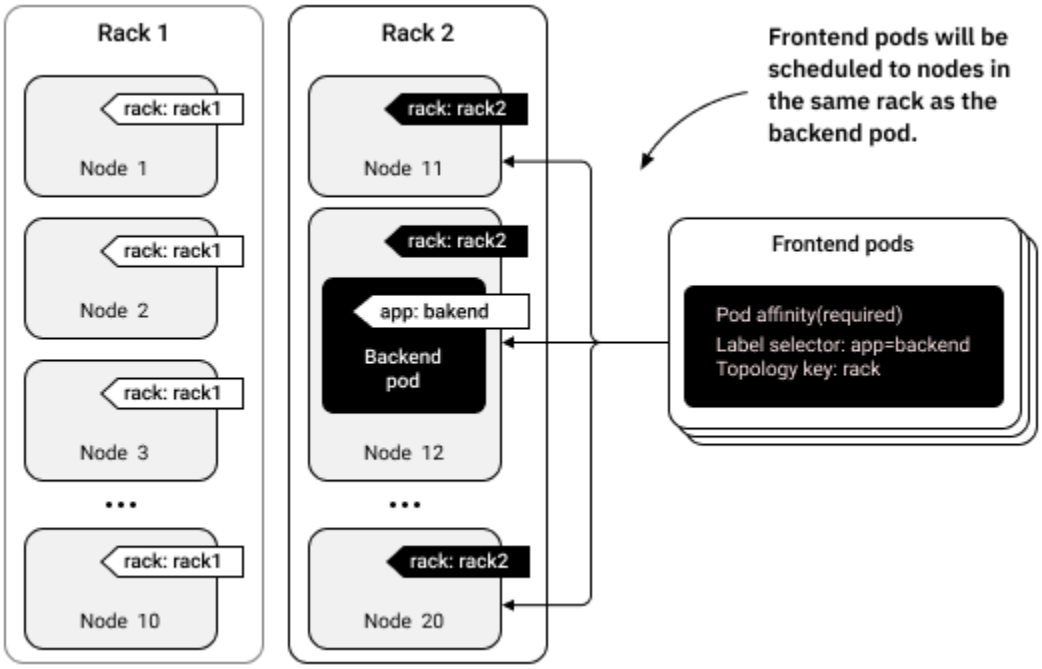
Так можно влиять не только на описание nodeSelector, но и на распределение Pod:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: frontend
spec:
  replicas: 5
  template:
    ...
    spec:
      affinity:
        podAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
          - topologyKey: kubernetes.io/hostname
            labelSelector:
              matchLabels:
                app: backend
```

Разберемся в topologyKey. Pod попадает на узел, на котором уже есть pod с меткой app=backend. Scheduler сначала выбирает ноды с заданным topologyKey, затем фильтрует их по признаку наличия Pod с app=backend.



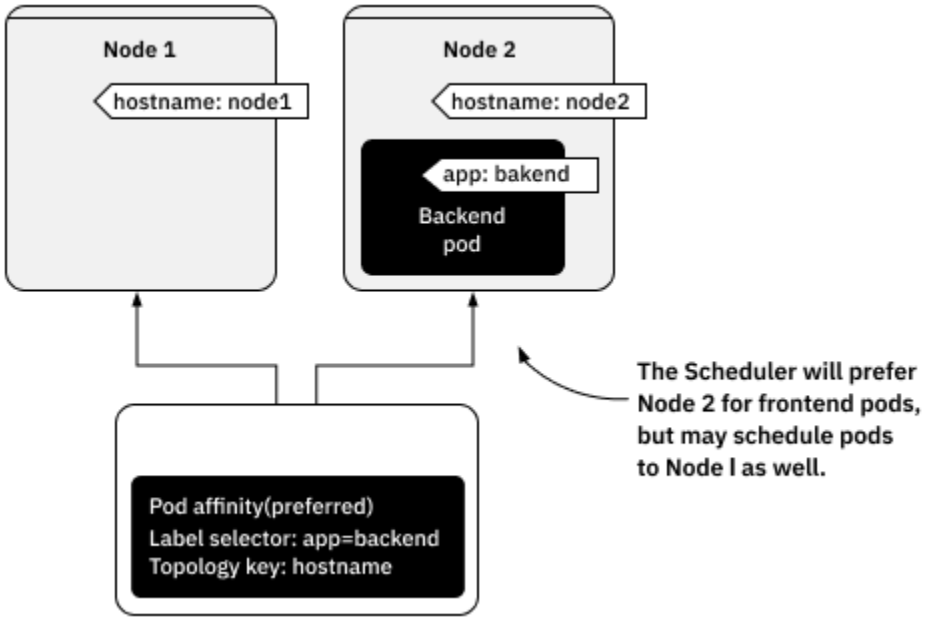
matchLabels можно заменить на matchExpression и написать более мощное выражение для выбора узла. Например, по стойкам в датацентре.



Или прописать приоритет в выборе узла с app=backend, но позволить Scheduler запускать Pod на другом узле, при нехватке ресурсов:

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: frontend
spec:
```

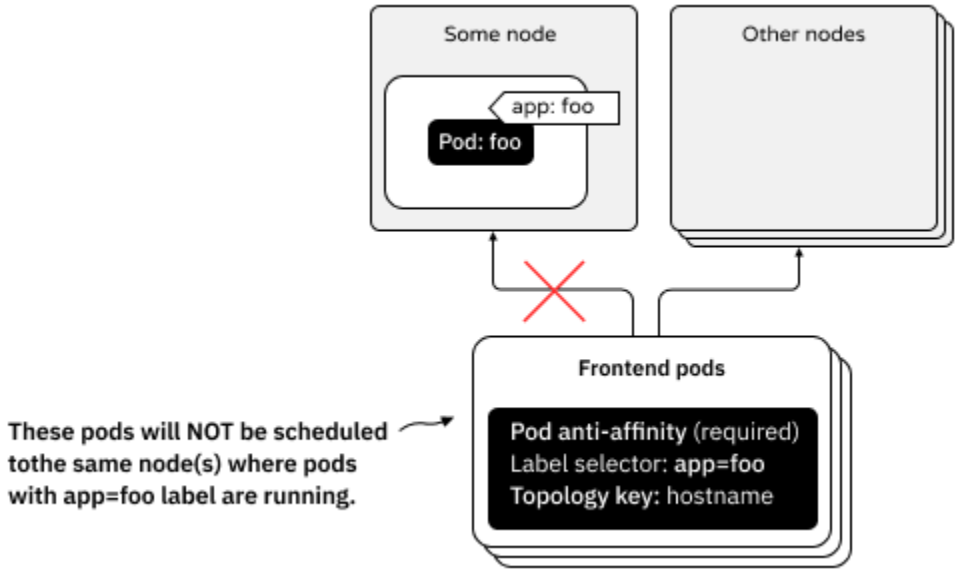
```
  replicas: 5
  template:
    ...
    spec:
      affinity:
        podAffinity:
          preferredDuringSchedulingIgnoredDuringExecution:
          - weight: 80
            podAffinityTerm:
              topologyKey: kubernetes.io/hostname
              labelSelector:
                matchLabels:
                  app: backend
      containers: ...
```



antiAffinity работает точно наоборот:

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

```
    labels:
    app: frontend
  spec:
    affinity:
      podAntiAffinity:
        requiredDuringSchedulingIgnoredDuringExecution:
        - topologyKey: kubernetes.io/hostname
          labelSelector:
            matchLabels:
              app: frontend
    containers: ...
```



Данный механизм позволяет распределять поды как угодно, в том числе сделать запуск подов в единичном экземпляре на каждой ноде (хотя для этого есть другой вид ресурсов, с которым мы познакомимся немного позднее).

Полезные ссылки:

- Assigning Pods to Nodes (official docs)
- Affinity
- CKA Labs (14): Kubernetes Affinity and Anti-Affinity

Задание:

1. Скопируйте манифест Deployment из предыдущего задания и переименуйте в rbm-nodeAffinity с реплика-фактором 2.
2. Замените правила nodeSelector в манифесте на аналогичное правило в формате nodeAffinity.
3. Примените манифест и выведите список подов, связанных с этим Deployment (команды и вывод сохранить).
4. Назначьте одной из нод с меткой task: nodeSelector новую метку task: nodeAntiAffinity (команду и вывод сохраните).
5. Добавьте 2 правила affinity или antiAffinity к Deployment, которые реализуют следующее:

- запуск только в единичном экземпляре на ноду;
- запретить запуск на нодах с меткой task: nodeAntiAffinity.

6. Примените измененный манифест и выведите список подов, связанных с этим Deployment с параметром -o wide (команды и вывод сохраните).
7. На проверку отправьте манифест, все выводы и причину, почему запустился только один под из двух, связанных с этим Deployment.