How scheduling works?

When a Pod is created , it is not assigned to any specific Node initially. instead, the Pod is marked as "unscheduled" and is added to a scheduling queue. The scheduler continuously watches this queue and selects an appropriate Node for each unscheduled Pod.The scheduler uses a set of rules to determine which nodes are eligible for scheduling. These rules include:

Resource requirements:
The scheduler looks at the CPU and memory requirements specified in the pod's configuration and ensures that the selected node has enough available resources to run the pod.

Node capacity:
The scheduler considers the capacity of each node in the cluster, including the amount of available CPU, memory, and storage, and selects a node that has sufficient capacity to meet the pod's requirements

Once the scheduler has identified a set of eligible nodes, it evaluates each node's fitness and assigns a score based on these factors. The node with the highest score is selected, and the pod is scheduled to run on that node.

Taints and tolerations:
Nodes in a Kubernetes cluster can be tainted to indicate that they have specific restrictions on the pods that can be scheduled on them. Pods can specify tolerations for these taints, which allow them to be scheduled on the tainted nodes.

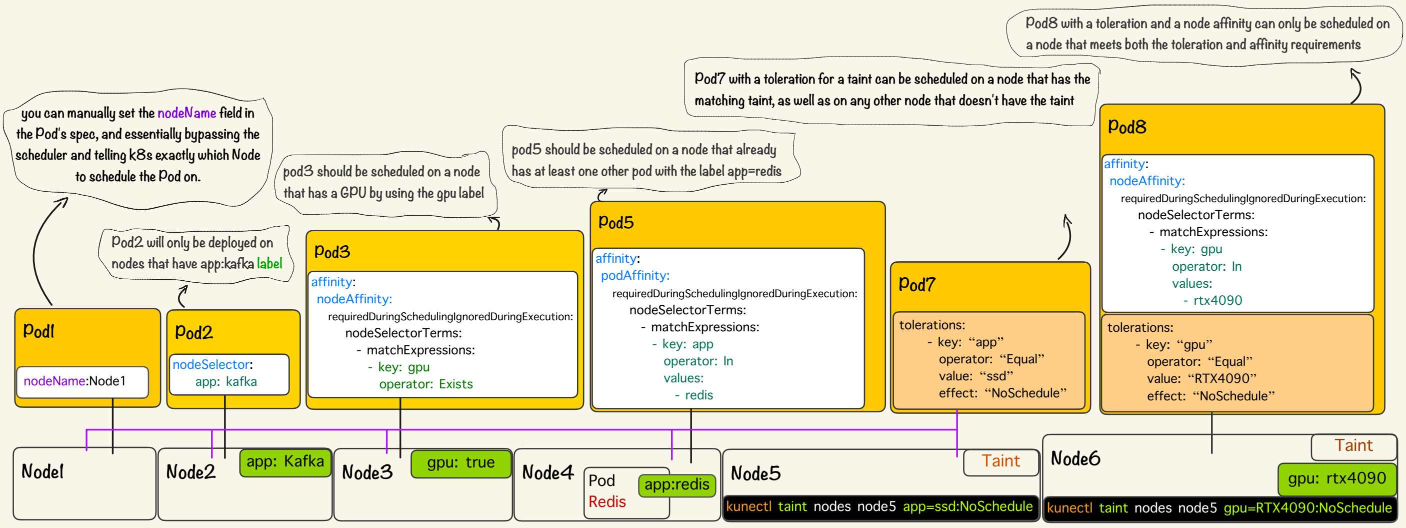
Node selectors:
Users can also specify node selectors, which are labels that are applied to nodes in the cluster. The scheduler can use these selectors to filter out nodes that don't match the pod's requirements.

Kubernetes also provides the ability to filter nodes based on various attributes before selecting them for scheduling. This allows users to specify additional constraints, such as selecting only nodes with specific labels or taints.

(Node,Pod) Affinity/anti-Affinity:
Kubernetes allows users to specify affinity and anti-affinity rules that control which nodes pods can be scheduled on. For example, a pod may be required to run on a node that has a specific label, or it may be prohibited from running on a node that already has a pod with a certain label.

If the scheduler is unable to find a suitable node for the pod, the pod remains unscheduled and enters a pending state until a suitable node becomes available.

You can constrain a Pod to run on specific nodes or prefer to run on particular nodes. There are several recommended approaches to achieve this, including Node Selector, Affinity/Anti-affinity, and Taint.



Pod8 with a toleration and a node affinity can only be scheduled on a node that meets both the toleration and affinity requirements

Pod7 with a toleration for a taint can be scheduled on a node that has the matching taint, as well as on any other node that doesn't have the taint

you can manually set the nodeName field in the Pod's spec, and essentially bypassing the scheduler and telling k8s exactly which Node to schedule the Pod on.

pod5 should be scheduled on a node that already has at least one other pod with the label app=redis

pod3 should be scheduled on a node that has a GPU by using the gpu label

Pod2 will only be deployed on nodes that have app:kafka label

Pod1

```
nodeName:Node1
```

Pod2

```
nodeSelector:
  app: kafka
```

Pod3

```
affinity:
 nodeAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
     nodeSelectorTerms:
       - matchExpressions:
         - key: gpu
           operator: Exists
```

Pod5

```
affinity:
 podAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
     nodeSelectorTerms:
       - matchExpressions:
         - key: app
           operator: In
           values:
             - redis
```

Pod7

```
tolerations:
     - key: "app"
       operator: "Equal"
       value: "ssd"
       effect: "NoSchedule"
```

Pod8

```
affinity:
 nodeAffinity:
   requiredDuringSchedulingIgnoredDuringExecution:
     nodeSelectorTerms:
       - matchExpressions:
         - key: gpu
           operator: In
           values:
             - rtx4090
tolerations:
     - key: "gpu"
       operator: "Equal"
       value: "RTX4090"
       effect: "NoSchedule"
```

```
kunectl taint nodes node5 app=ssd:NoSchedule
```

```
kunectl taint nodes node5 gpu=RTX4090:NoSchedule
```

Pod affinity is used to ensure that a pod is scheduled on a node that has other pods running with certain characteristics, while node affinity refers to the preference of a pod to be scheduled on a specific node based on its labels

taints allow to mark a node as unsuitable for certain pods, preventing them from being scheduled on that node unless they have a matching toleration

tolerations, affinity, and node selectors are defined on pods, while Labels and taints are defined on nodes

Labels & selector

labels are a powerful mechanism for grouping and organizing related objects, such as Pods, Services, Deployments, and more. Labels are key-value pairs that can be attached to Kubernetes objects, and they can be used for a variety of purposes, such as grouping related objects for easy management, selecting objects for operations such as scaling or updating, and enabling fine-grained access control

there are several ways to use labels to group objects in Kubernetes

- Grouping by object type: You can use labels to group objects based on their type, such as Pods, Services, Deployments, ConfigMaps
- Grouping by application: You can use labels to group objects based on the application they belong to, such as a web application, a database, or a caching layer
- Grouping by functionality: You can use labels to group objects based on their functionality, such as front-end components, back-end components, databases, caches, authentication services, video processing services

Annotations

Annotations are similar to labels, but they are designed to store additional information that is not used for grouping or selection, They can be used to store information such as version numbers, timestamps, configuration details, and other metadata that is useful for debugging, monitoring, or other purposes

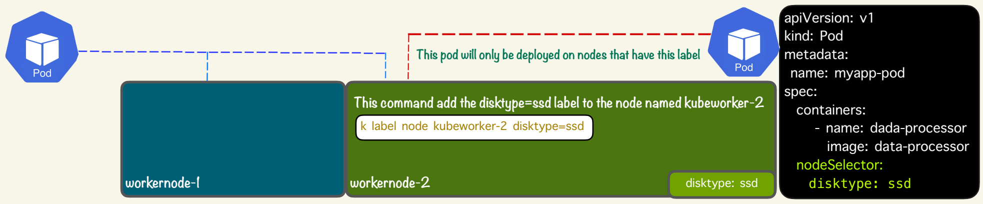
you can use annotations to configure the Nginx ingress controller. However, for more complex configurations, it can be easier to maintain and manage your Nginx configuration by using a ConfigMap

```
annotations:
 nginx.ingress.kubernetes.io/proxy-cache: "on"
 nginx.ingress.kubernetes.io/proxy-cache-path: "/data/nginx/cache"
 nginx.ingress.kubernetes.io/proxy-cache-max-size: "100m"
```

Annotations can be up to 256 kilobytes in size, allowing you to store more complex metadata with Kubernetes objects (labels are limited to 63 characters)

Node selector

NodeSelector is a feature in Kubernetes that allows you to specify a set of labels that a node must have in order for a pod to be scheduled on that node. When you create a pod, you can specify a NodeSelector in the pod spec that will be used to match against the labels of all the nodes in the cluster. If any node has labels that match the NodeSelector, then the pod can be scheduled on that node. also for more complex and multiple constraints such as deploying a Pod on two nodes with different labels, it's better to use Affinity or Anti-Affinity



This pod will only be deployed on nodes that have this label

This command add the disktype=ssd label to the node named kubeworker-2

```
k label node kubeworker-2 disktype=ssd
```

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
spec:
  containers:
     - name: dada-processor
       image: data-processor
  nodeSelector:
    disktype: ssd
```

If a pod's NodeSelector specifies labels that don't exist on any node, the pod won't be scheduled until a node is labeled appropriately

Pending...

This command removes the disktype label from the node named kubeworker-2

```
k label node kubeworker-2 disktype-
```