Pod Placement

Kubernetes Node Placement Strategies

Node Selector

Node Affinity

Taints & Tolerations

Why Node Selection Matters

- Ensure Pods run on the right nodes
- Optimize for performance, cost, or isolation
- Enforce scheduling rules and access control

✅ Node Selector

Simple, label-based filtering

```
spec:
  nodeSelector:
    disktype: ssd
```

- **Matches only exact key-value pairs**
- Fast and easy
- Cannot express advanced logic

Node Affinity

Advanced, expressive rules for node selection

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
        - key: disktype
          operator: In
          values: ["ssd"]
```

- **Supports complex match expressions**
- **Hard (required)** and **soft (preferred)** rules
- Good for flexible scheduling policies

Node Affinity Types

Type	Behavior
`requiredDuringScheduling...`	Pod **must** match rules to be placed
`preferredDuringScheduling...`	Pod **prefers** nodes that match

🚫 Taints and Tolerations

Nodes repel Pods unless tolerated

```
kubectl taint nodes node1 dedicated=group1:NoSchedule
```

```
tolerations:
- key: "dedicated"
  operator: "Equal"
  value: "group1"
  effect: "NoSchedule"
```

- Adds *node-side* restrictions
- Pods need matching tolerations to be scheduled

Taint Effects

Effect	Description
`NoSchedule`	Pod won't be scheduled unless it tolerates
`PreferNoSchedule`	Scheduler avoids but does not block
`NoExecute`	Pod is evicted if it doesn't tolerate

🧠 When to Use What?

- **Node Selector**: Simple, exact needs
- **Node Affinity**: Custom rules and preferences
- **Taints/Tolerations**: Enforce isolation or dedicated node access

🎯 Key Takeaways

- Combine these tools for precise control
- Understand node labeling, tainting, and affinity strategy
- Use **Affinity + Toleration** for real-world production policies

Summary Table

Feature	Node Selector	Node Affinity	Taints & Tolerations
Direction	Pod → Node	Pod → Node	Node → Pod
Matching	Exact match	Expressions + weights	Requires toleration
Complex Rules	❌	✅	✅
Use Case	Simple placement	Flexible scheduling logic	Node-level protection