🎯 Kubernetes Scheduling – Labels, Affinity & Taints (Complete Guide)

This repository demonstrates **how Kubernetes schedules Pods onto Nodes** using:

- Labels & Selectors
- Node Affinity (required / preferred)
- Pod Affinity & Anti-Affinity
- Taints & Tolerations (`NoSchedule` , `PreferNoSchedule` , `NoExecute`)

📚 Table of Contents

🧠 Kubernetes Scheduling Basics

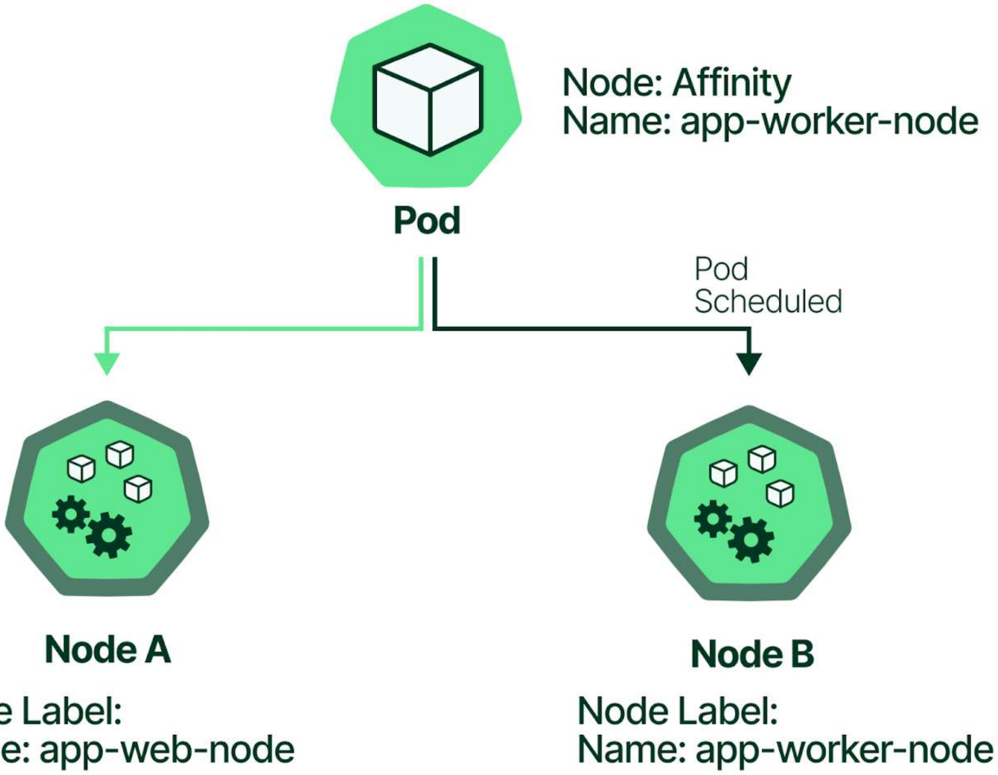
Kubernetes Scheduler decides **WHERE a Pod runs** based on:

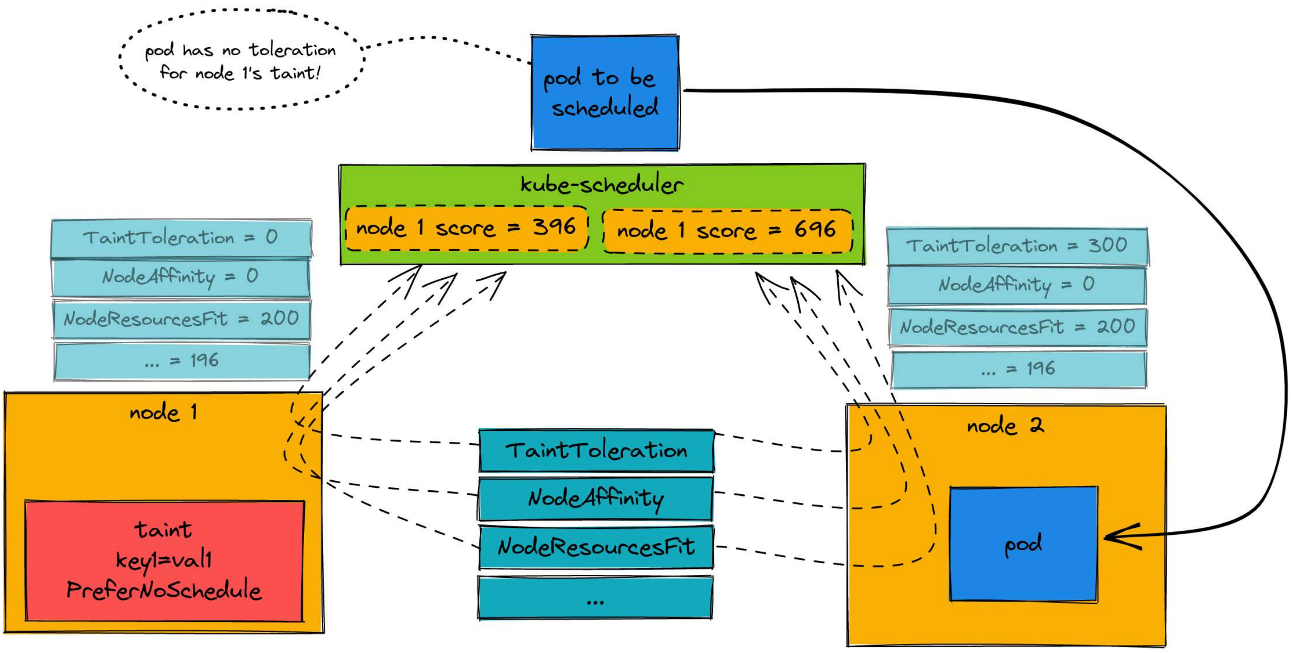
```
Filters (hard rules) → Scoring (soft rules)
```

Mechanism	Type
Labels	Identification
NodeSelector	Simple filter

Mechanism	Type
Node Affinity	Advanced filter
Pod Affinity	Pod-to-Pod rules
Taints	Node repulsion
Tolerations	Pod permission

🏗️ Scheduling Architecture





```
Pod
 |
 ▼
Scheduler
 |
 ├─ Labels / Affinity
 ├─ Taints / Tolerations
 |
 ▼
Node
```

🏷️ Labels (`label.yml`)

Labels are **key-value metadata** used for selection.

Apply label to node

```
kubectl label node node1 disktype=ssd
```

Example Pod using label selector

```
nodeSelector:
    disktype: ssd
```

✔️ Simple ❌ Limited (exact match only)

🧲 Node Affinity

Node Affinity is the **advanced version of nodeSelector**.

🔒 Required Node Affinity (`affinity-required.yml`)

> **Hard rule** – Pod will NOT schedule if condition fails

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

📌 Use when:

- GPU nodes
- SSD-only workloads
- Compliance requirements

🎯 Preferred Node Affinity (`affinity-preferred.yml`)

> **Soft rule** – Scheduler tries but doesn't force

```
affinity:
  nodeAffinity:
    preferredDuringSchedulingIgnoredDuringExecution:
    - weight: 1
      preference:
        matchExpressions:
        - key: zone
          operator: In
          values:
          - us-east-1a
```

📌 Use when:

- Cost optimization
- Latency preference
- HA setups

⚖️ Required vs Preferred (Critical Difference)

Feature	Required	Preferred
Hard rule	✅	❌
Pod stuck pending	✅	❌
Scheduler scoring	❌	✅
Production critical	✅	⚠️

☠️ Taints & Tolerations

Taints **repel Pods** from Nodes. Tolerations **allow Pods** to be scheduled.

🚫 Taint Node

```
kubectl taint node node1 key=value:NoSchedule
```

This blocks all Pods **unless tolerated**.

✅ Toleration (`taint-toleration.yml`)

```yaml
tolerations:
- key: "key"
  operator: "Equal"
  value: "value"
  effect: "NoSchedule"
```

✔️ Pod can now run on tainted node

🧨 NoExecute Taint (`noexecute.yml`)

> Strongest taint – **evicts running pods**

```yaml
tolerations:
- key: "maintenance"
  operator: "Exists"
  effect: "NoExecute"
  tolerationSeconds: 30
```

Behavior

```
No toleration → immediate eviction
With toleration → stays for X seconds
```

📌 Used for:

- Node maintenance
- Spot instance termination
- Node drain automation

☠️ Taint Effects (Very Important)

Effect	Behavior
NoSchedule	New Pods blocked

Effect	Behavior
PreferNoSchedule	Best-effort
NoExecute	Evict running Pods

🔍 Verification Commands (SRE Style)

```
kubectl get nodes --show-labels
kubectl describe node <node-name>
kubectl get pods -o wide
kubectl describe pod <pod-name>
```

Pending pod debug:

```
kubectl get events --sort-by=.metadata.creationTimestamp
```

🌍 Real-World Use Cases

Scenario	Solution
GPU workloads	Required Node Affinity
Logging agents	Toleration + DaemonSet
DB on SSD	Node labels + affinity
Spot nodes	NoExecute taint
Prod isolation	Taints + tolerations

✅ Production Best Practices

✔ Use **required affinity** for critical workloads ✔ Use **preferred affinity** for optimization ✔ Taint special nodes (GPU, DB, Infra) ✔ Never rely only on nodeSelector ✔ Combine with **PodDisruptionBudgets** ✔ Test failure scenarios

🧠 Interview-Ready One-Liners

- Labels identify, affinity decides
- Required = hard rule, preferred = soft rule
- Taints repel, tolerations allow
- NoExecute evicts running pods
- Scheduler uses filters + scoring