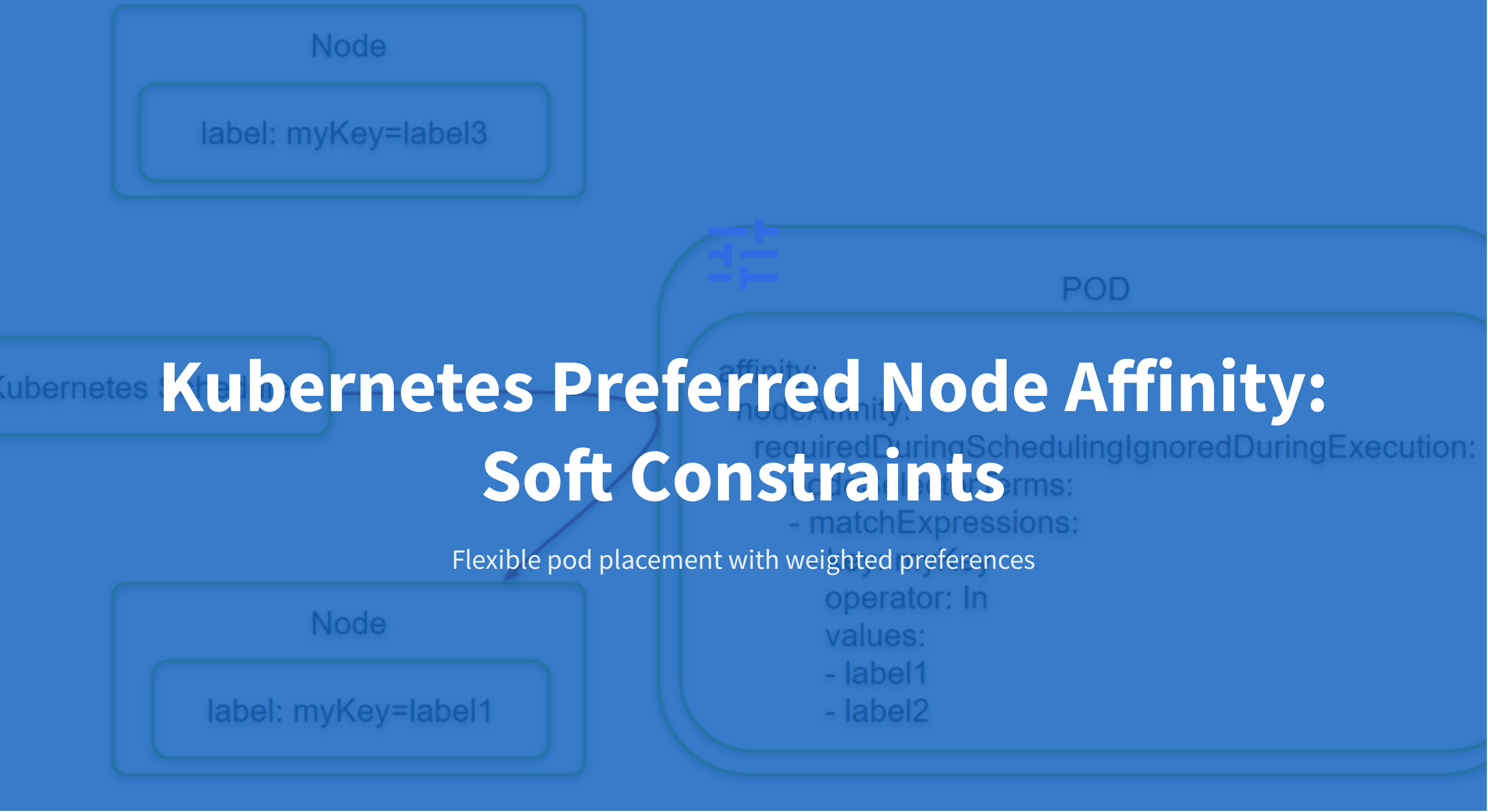

Kubernetes Preferred Node Affinity: Soft Constraints

Flexible pod placement with weighted preferences

Understanding Preferred Node Affinity

Key Concepts

- **preferredDuringSchedulingIgnoredDuringExecution** - Soft rule (scheduler tries, but won't block)
- **weight** - How much to favor this rule (1 = low, 100 = high)
- **preference** - What node attributes are preferred

Weight Range

Use higher weights for stronger preferences

● 1-30: Low ● 31-70: Medium ● 71-100: High

YAML Example

```yaml
affinity: nodeAffinity:
preferredDuringSchedulingIgnoredDuringExecution:
  - weight: 1 # ← Importance (1-100)
    preference:
     matchExpressions:
     - key: size
       operator: In
                 values: - small
```

Behavior Scenarios

Scenario	Outcome
size=small node exists + has resources	**Scheduled there** (preferred)
No size=small node	**Scheduled elsewhere** (no failure!)
Multiple preferred rules	**Scores nodes** → picks highest total weight

Key Benefits

- **Flexible scheduling** - Pods always schedule somewhere
- **Optimization focus** - Best-effort placement for better performance
- **Cost-effective** - Can prefer cheaper instances when available
- **Graceful fallback** - Works even if preferred nodes unavailable

Lab: Preferred Affinity in Action

1 Label Node as "small"

```
# List nodes
kubectl get nodes

# Label ONLY ONE node
kubectl label node k3s-node1 size=small

# Verify
kubectl get nodes --show-labels | grep -E
"k3s-node1|k3s-node2"
```

2 Deploy Pod

```
# Create namespace
kubectl create namespace learning

# Apply Pod with preferred affinity
kubectl apply -f pod-with-preferred-node-
affinity.yml

# Check where it runs
kubectl get pods -n learning -o wide
```

3 Expected Result

```
# Pod runs on k3s-node1 (the "small" node)
# But if k3s-node1 is full, it might run on
k3s-node2
# → That's OK with preferred affinity!
```

4 Test Fallback Behavior

```
# Taint "small" node to make it unschedulable
kubectl taint node k3s-node1
test=unschedulable:NoSchedule

# Deploy a SECOND Pod (same spec)
kubectl run nnappone2 -n learning --image=nginx --
restart=Never \\ --overrides='{"spec":{"affinity":
{"nodeAffinity":
{"preferredDuringSchedulingIgnoredDuringExecution":
[{"weight":1,"preference":{"matchExpressions":
[{"key":"size","operator":"In","values":
["small"]}]}}]}}}'

# Check placement
kubectl get pods -n learning -o wide
```

Expected: nnappone2 runs on k3s-node2 (fallback!)

Clean Up

```
kubectl delete pod nnappone nnappone2 -n learning
kubectl taint node k3s-node1 test:NoSchedule- #
remove taint
kubectl label node k3s-node1 size- # remove label
kubectl delete namespace learning
```

Required vs Preferred Affinity

Feature	required...	preferred...
Constraint Type	! Hard (must match)	Soft (try to match)
Scheduling Failure?	Yes (if no match)	No (always schedules)
Use Case	"Must run on GPU"	"Prefer SSD, but OK on HDD"
Weight	N/A	1–100 (higher = stronger preference)

Best Practice: Use **preferred** for optimization, **required** for hard requirements

Real-World Examples & Summary

Cost Optimization Example

$ Weighted Preferences for Cost Savings

YAML Configuration

```
affinity: nodeAffinity:
preferredDuringSchedulingIgnoredDuringExecution:
- weight: 100 # Strong preference preference:
matchExpressions: - key:
node.kubernetes.io/instance-type operator: In
values: [t3.small] # cheap spot instances -
weight: 50 # Medium preference preference:
matchExpressions: - key:
topology.kubernetes.io/zone operator: In values:
[us-east-1a]
```

"Strongly prefer cheap instances, and somewhat prefer zone A"

Key Points

- **Soft constraint** — Never blocks scheduling
- **Weight (1-100)** — Express preference strength
- **Graceful fallback** — Works even if preferred nodes unavailable
- **Multi-node clusters** — Perfect for production environments
- **Combine with taints** — Simulate node unavailability

Best Practice

Use **preferred** for optimization and **required** for hard requirements