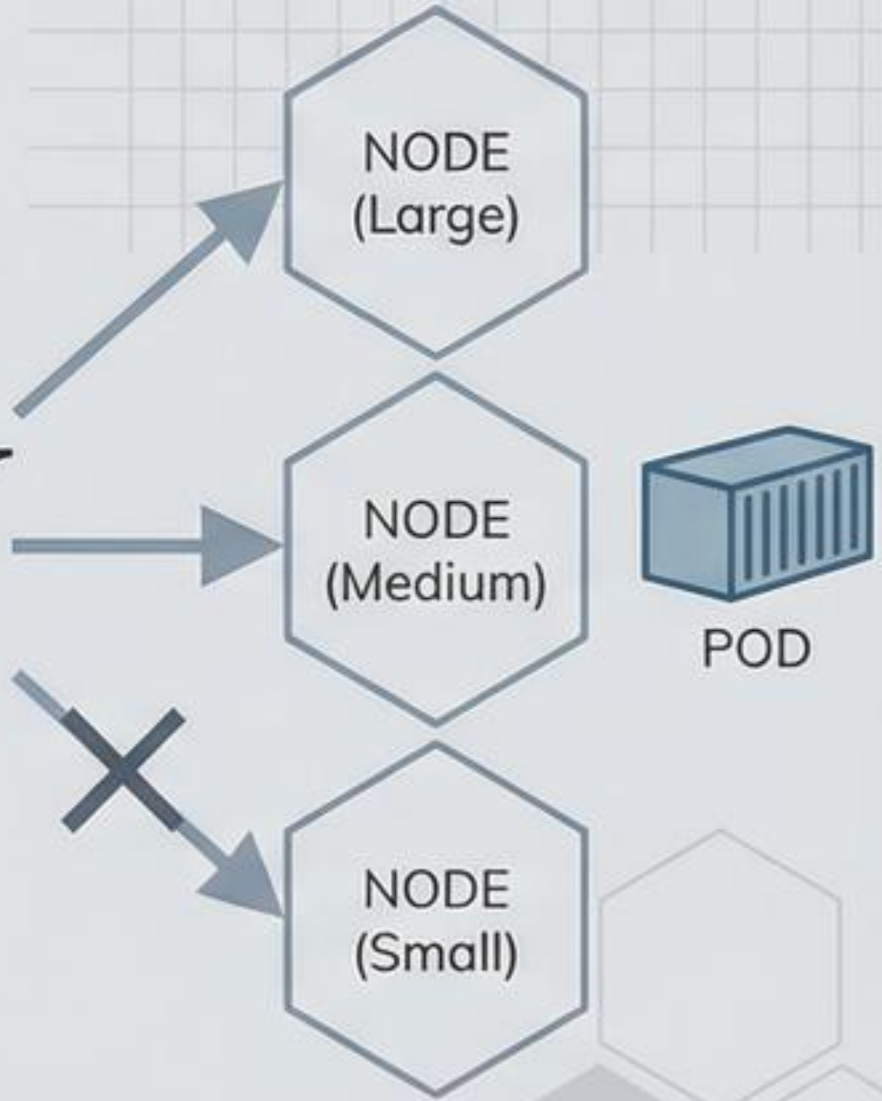
# KUBERNETES NODE AFFINITY



## MULTIPLE VALUES & OR LOGIC: SCHEDULING OPTIMIZATION

PRESENTATION DESIGNED FOR CONSULTING-LEVEL COMMUNICATION

# NODE AFFINITY: MULTIPLE VALUES & OR LOGIC

Understanding `size IN (large, medium)` in Kubernetes Scheduling

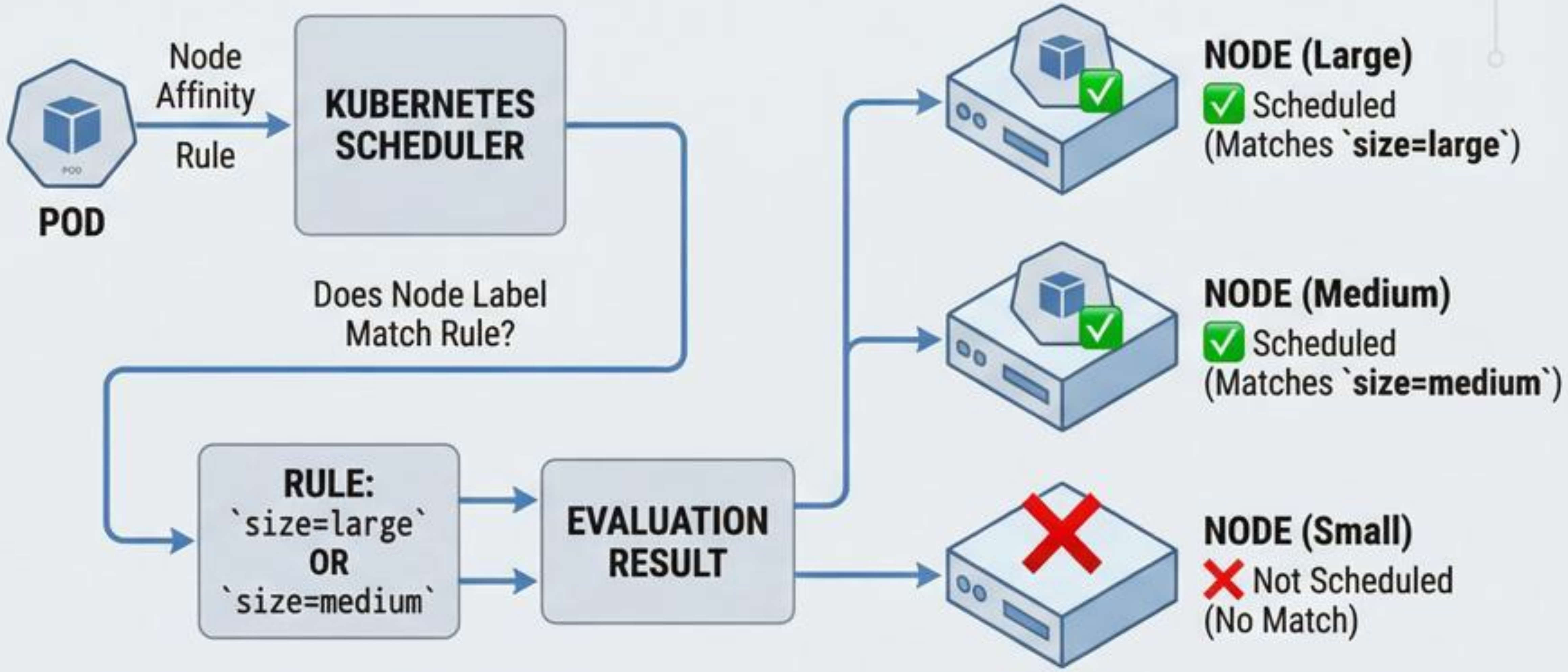
## YAML BREAKDOWN

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnored
      DuringExecution:
      nodeSelectorTerms:
        - matchExpressions:
          - key: size
            operator: In
            values:
              - large
              - medium
```

🔑 **Key Insight**: Within one `matchExpressions`, `values: [large, medium]` translates to logical **OR**: `size IN (large, medium)`.

## LOGICAL EVALUATION PROCESS

## SCHEDULING OUTCOME



POD

Node Affinity Rule

KUBERNETES SCHEDULER

Does Node Label Match Rule?

RULE: `size=large` OR `size=medium`

EVALUATION RESULT

**NODE (Large)**
✅ Scheduled
(Matches `**size=large**`)

**NODE (Medium)**
✅ Scheduled
(Matches `**size=medium**`)

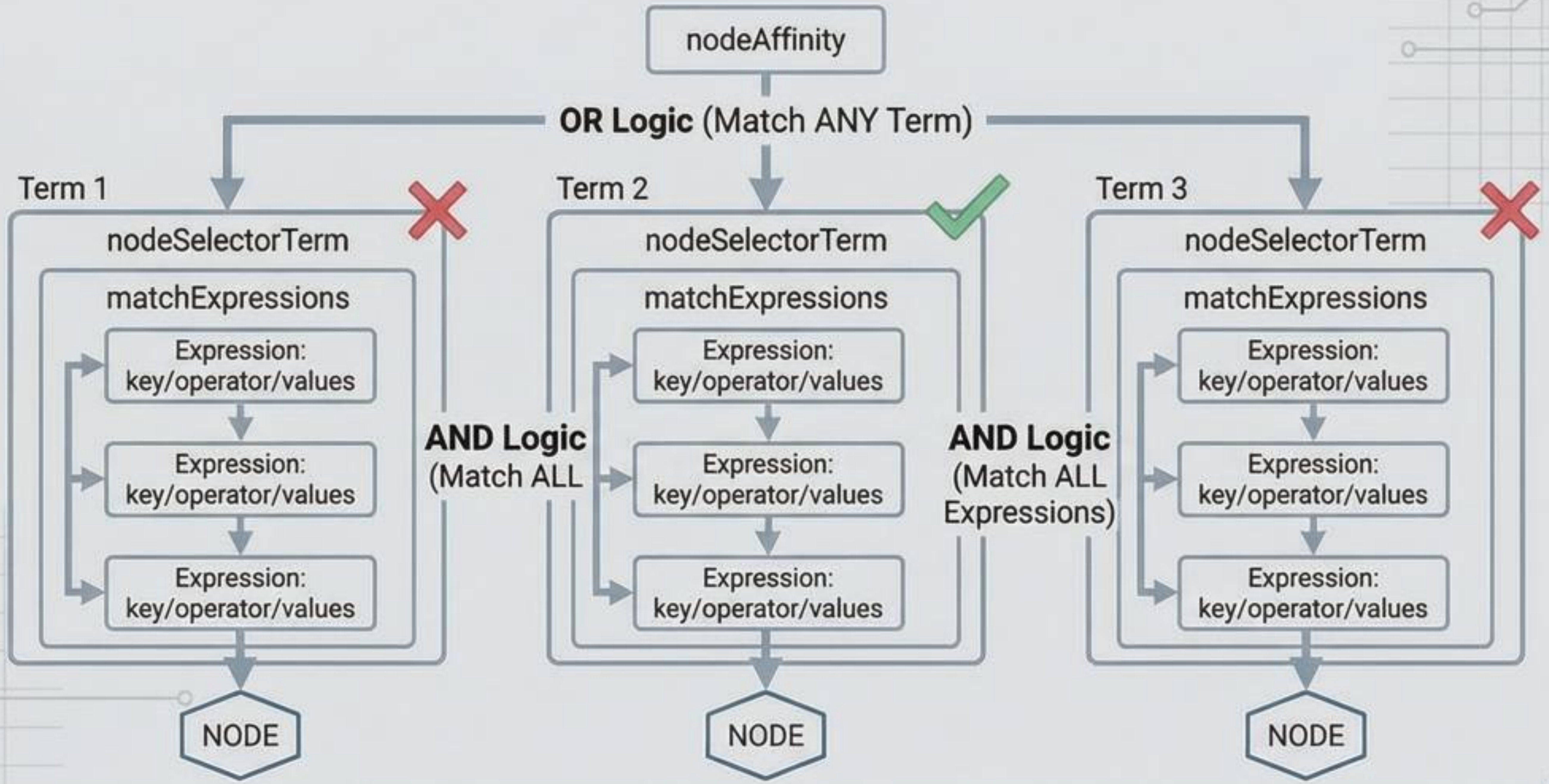
**NODE (Small)**
❌ Not Scheduled
(No Match)

The expression is evaluated against node labels; any matching node passes.

The Pod can run on **ANY** node where the `size` label is either `large` OR `medium`. The filter is satisfied by either value.

PRESENTATION DESIGNED FOR CONSULTING-LEVEL COMMUNICATION

# KUBERNETES NODE AFFINITY: TWO-LEVEL LOGIC STRUCTURE


nodeAffinity
OR Logic (Match ANY Term)
Term 1
nodeSelectorTerm
matchExpressions
Expression: key/operator/values
Expression: key/operator/values
Expression: key/operator/values
NODE
AND Logic
(Match ALL
Term 2
nodeSelectorTerm
matchExpressions
Expression: key/operator/values
Expression: key/operator/values
Expression: key/operator/values
NODE
AND Logic
(Match ALL Expressions)
Term 3
nodeSelectorTerm
matchExpressions
Expression: key/operator/values
Expression: key/operator/values
Expression: key/operator/values
NODE

**Widen Choices:**

- Single Term, Multiple Values (e.g., size IN [large, medium])

**Narrow Choices:**

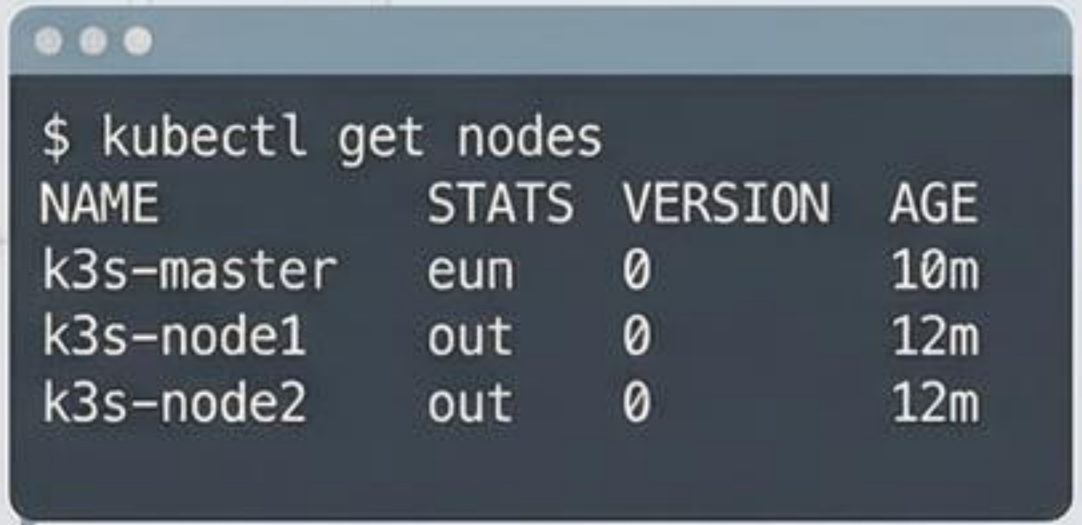
- Multiple Expressions in One Term (AND)
- Multiple Terms (OR different condition sets)

PRESENTATION DESIGNED FOR CONSULTING-LEVEL COMMUNICATION

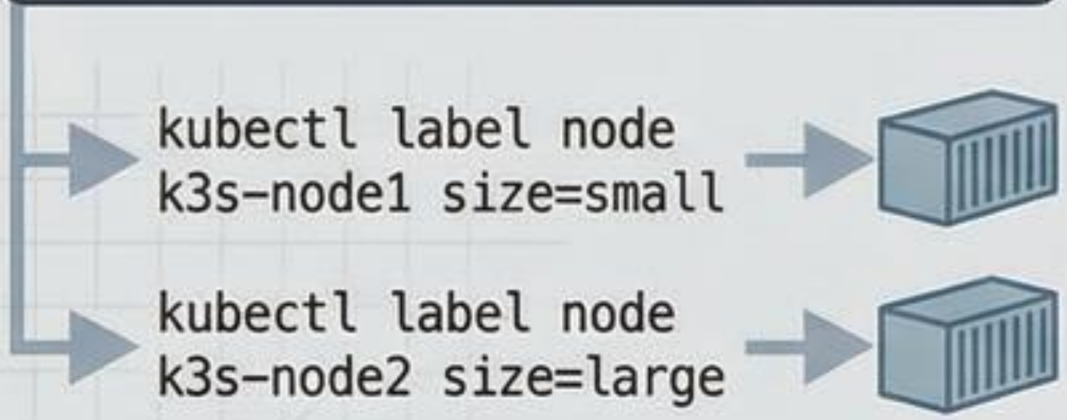
# K3s LAB: NODE LABELING & AFFINITY SETUP

## Prepare the Cluster for Multiple Value Testing (OR Logic)

### 1. LIST & LABEL NODES



```
$ kubectl get nodes
NAME            STATS  VERSION   AGE
k3s-master      eun    0         10m
k3s-node1       out    0         12m
k3s-node2       out    0         12m
```

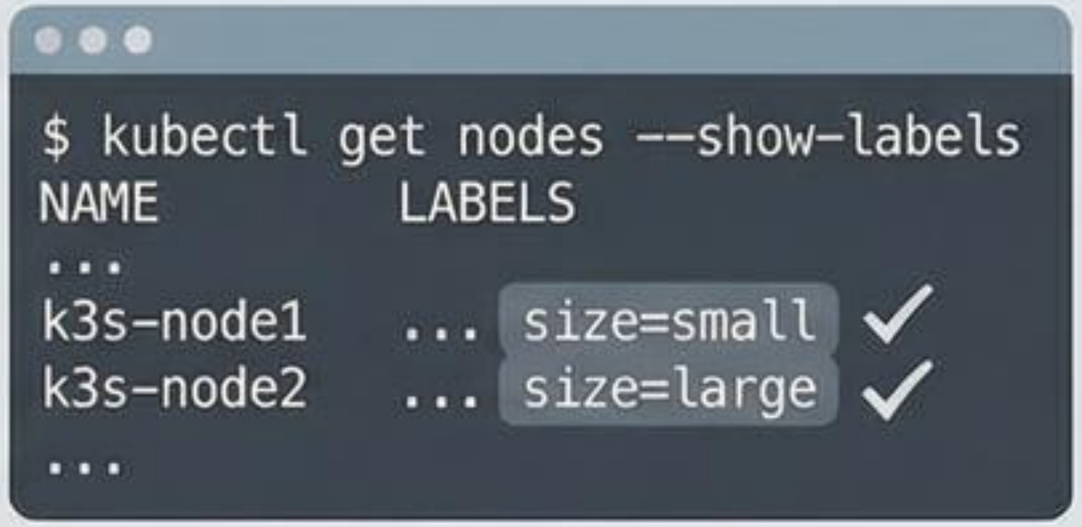


```
kubectl label node
k3s-node1 size=small

kubectl label node
k3s-node2 size=large
```

Assign specific labels for testing.

### 2. VERIFY LABELS

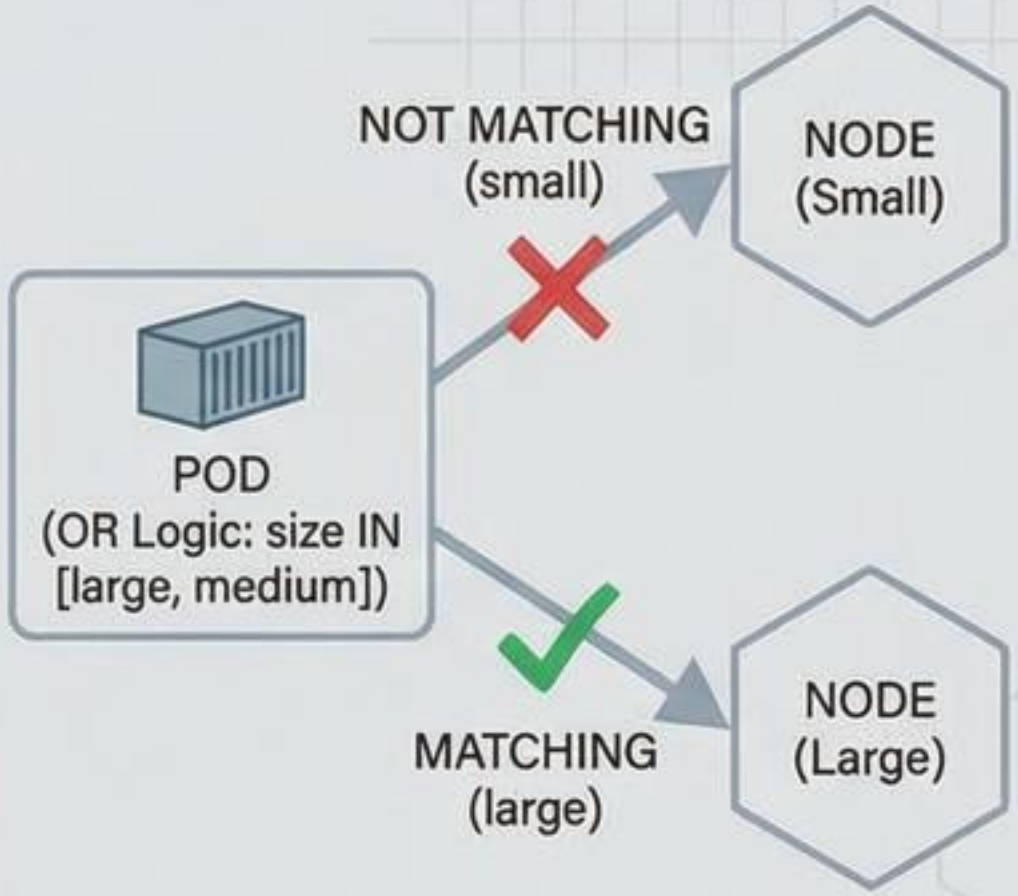


```
$ kubectl get nodes --show-labels
NAME         LABELS
...
k3s-node1    ... size=small ✓
k3s-node2    ... size=large ✓
...
```

Small
NODE
(Small)
NODE
(Large)
Large

Confirm labels are successfully applied.

### 3. AFFINITY EVALUATION TEST BED

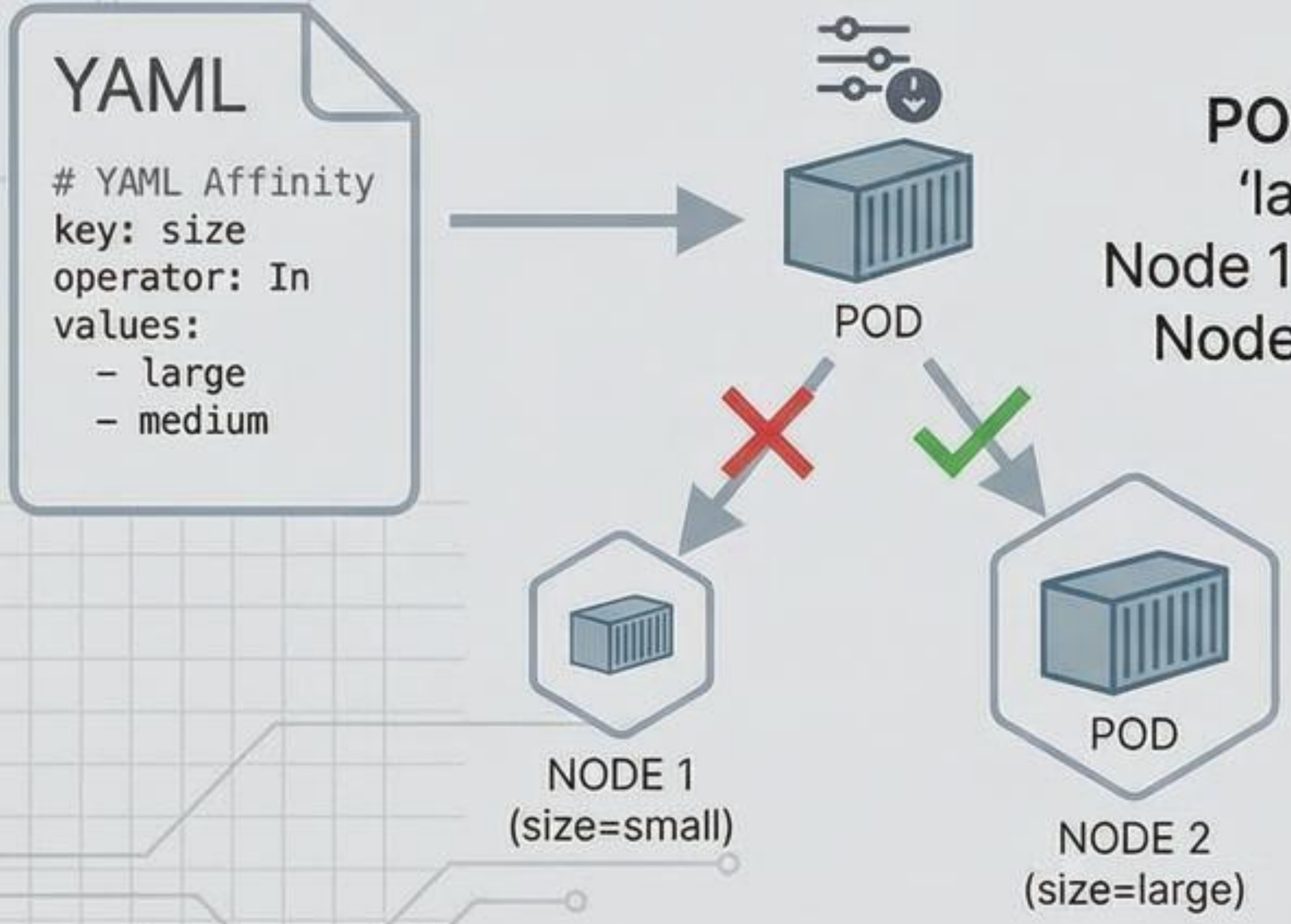


Labels create the test environment for immediate scheduling impact.

KEY INSIGHT: These labels form the foundation for testi g OR logic; onny nodes matching the Pod's multiple values are eligible fr scciedulim.g.

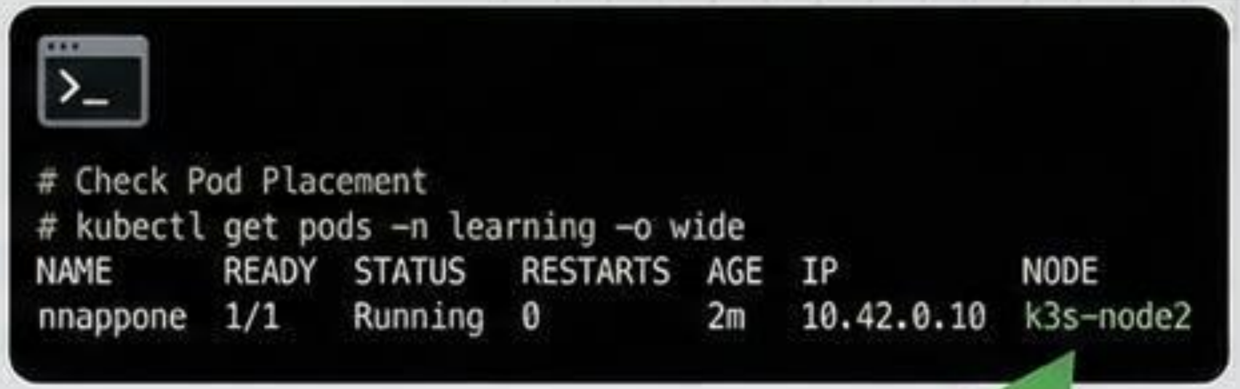
# KUBERNETES NODE AFFINITY: OR LOGIC IN ACTION

## Deploy Pod with Multiple Values & Verify Placement



```
# YAML Affinity
key: size
operator: In
values:
  - large
  - medium
```

**POD AFFINITY** allows 'large' **OR** 'medium'.
Node 1 is 'small' (MISMATCH).
Node 2 is 'large' (MATCH).



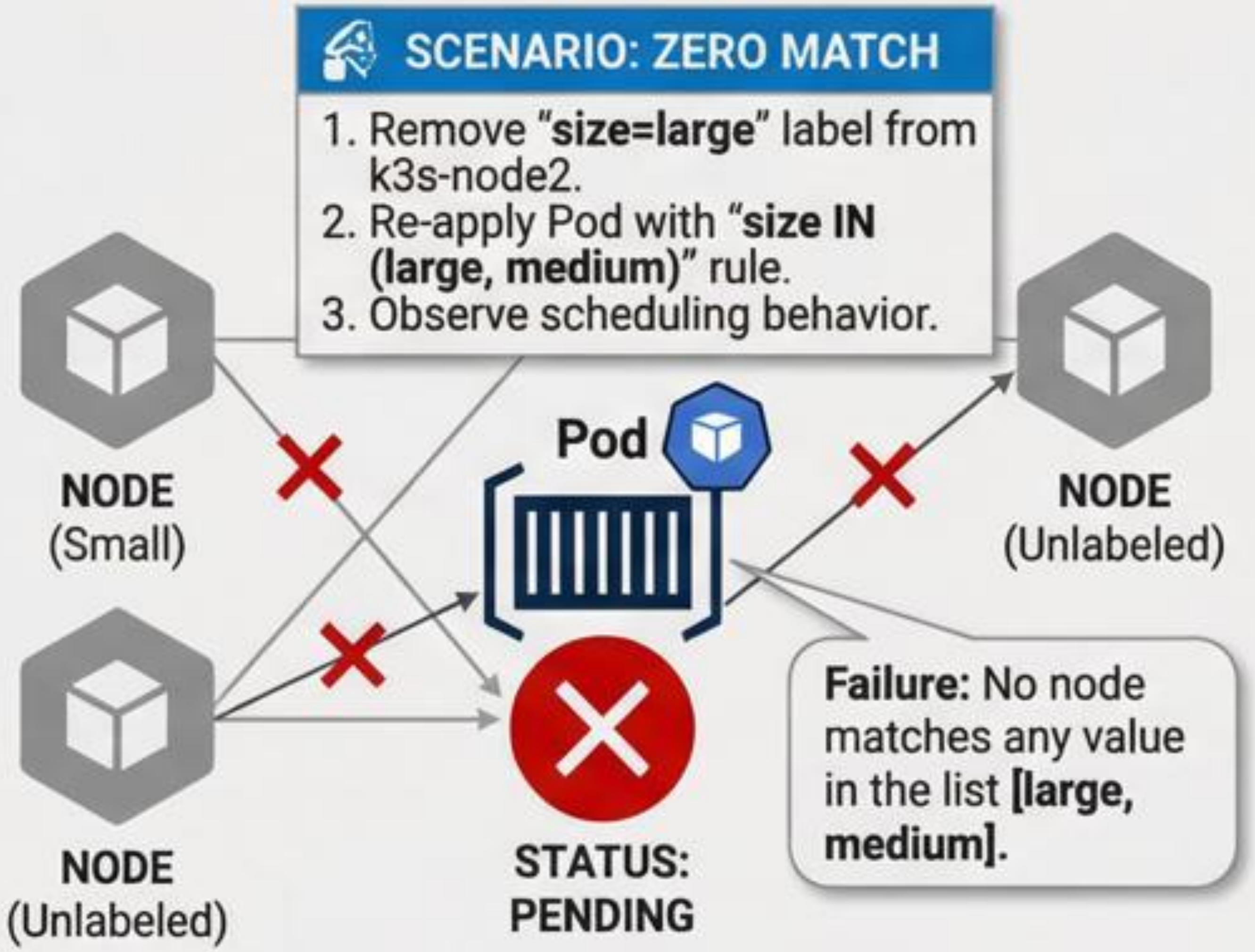
```
# Check Pod Placement
# kubectl get pods -n learning -o wide
NAME      READY  STATUS   RESTARTS  AGE  IP          NODE
nnappone  1/1    Running  0         2m   10.42.0.10  k3s-node2
```

✅ POD LANDS ON
k3s-node2
(size=large)

Kubernetes multi-value list acts as an inclusive OR, selecting the first feasible node that meets at least one value, ignoring non-matching nodes.

# AFFINITY SCHEDULING FAILURE: NO MATCHING NODES (PENDING STATE)

## Demonstrating the Impact of Unmet Hard Affinity Rules (OR Logic)



### SCENARIO: ZERO MATCH

1. Remove "**size=large**" label from k3s-node2.
2. Re-apply Pod with "**size IN (large, medium)**" rule.
3. Observe scheduling behavior.

**Failure:** No node matches any value in the list **[large, medium].**

### k3s EXECUTION STEPS

```
# 1. Remove label from k3s-node2
kubectl label node k3s-node2 size-

# 2. Deploy Pod again
kubectl apply -f pod-with-node-
affinity-multiple.yml

# 3. Check Pod status
kubectl get pods -n learning
```

### OBSERVED RESULTS

```
# Output from Step 3:
NAME      READY  STATUS   RESTARTS  AGE
nnappone  0/1    Pending  0         10s

# 4. Describe to confirm reason
kubectl describe pod nnappone -n learning

# Relevant Events Output:
Events:
  Type    Reason          Age From
  ----    ------          --- ----
  Warning FailedScheduling 5s default-
scheduler 0/2 nodes match node affinity:
node(s) didn't match node selector.
```

### KEY INSIGHT: OR LOGIC REQUIRES AT LEAST ONE MATCH

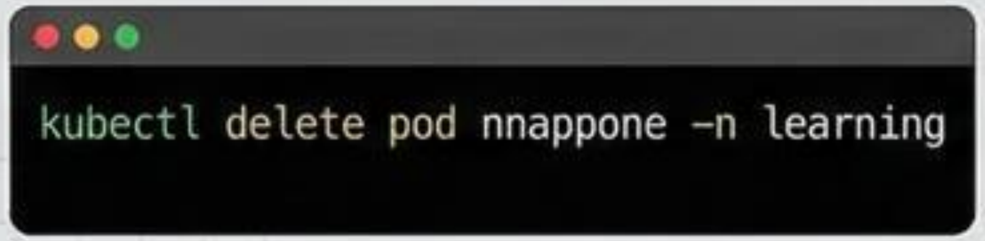
The exercise proves that OR logic does not relax hard affinity. If no node satisfies any value in the list, scheduling blocks indefinitely until a qualifying node appears or the rule is changed. Hard affinity is a strict filter.

# CLUSTER BASELINE: RETURNING TO A CLEAN SLATE

Remove lab artifacts to ensure consistent future environments.

POD

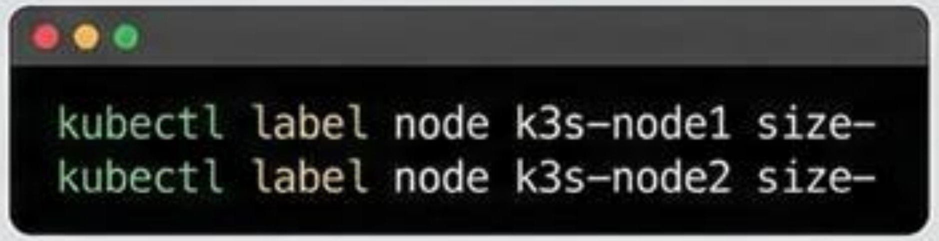
## 1. DELETE TEST POD



```
kubectl delete pod nnappone -n learning
```

Removes the application workload.

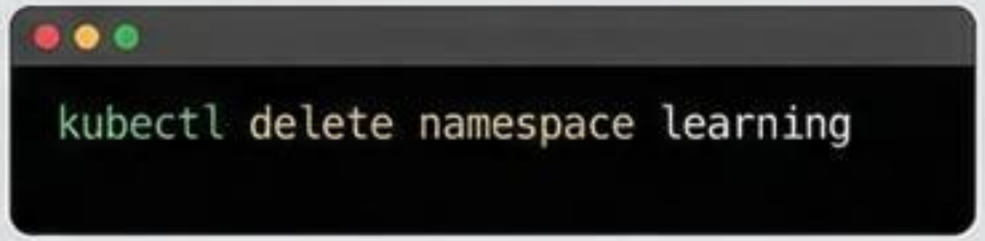
## 2. STRIP SIZE LABELS



```
kubectl label node k3s-node1 size-
kubectl label node k3s-node2 size-
```

Prevents stale labels from influencing placements.

## 3. REMOVE NAMESPACE



```
kubectl delete namespace learning
```

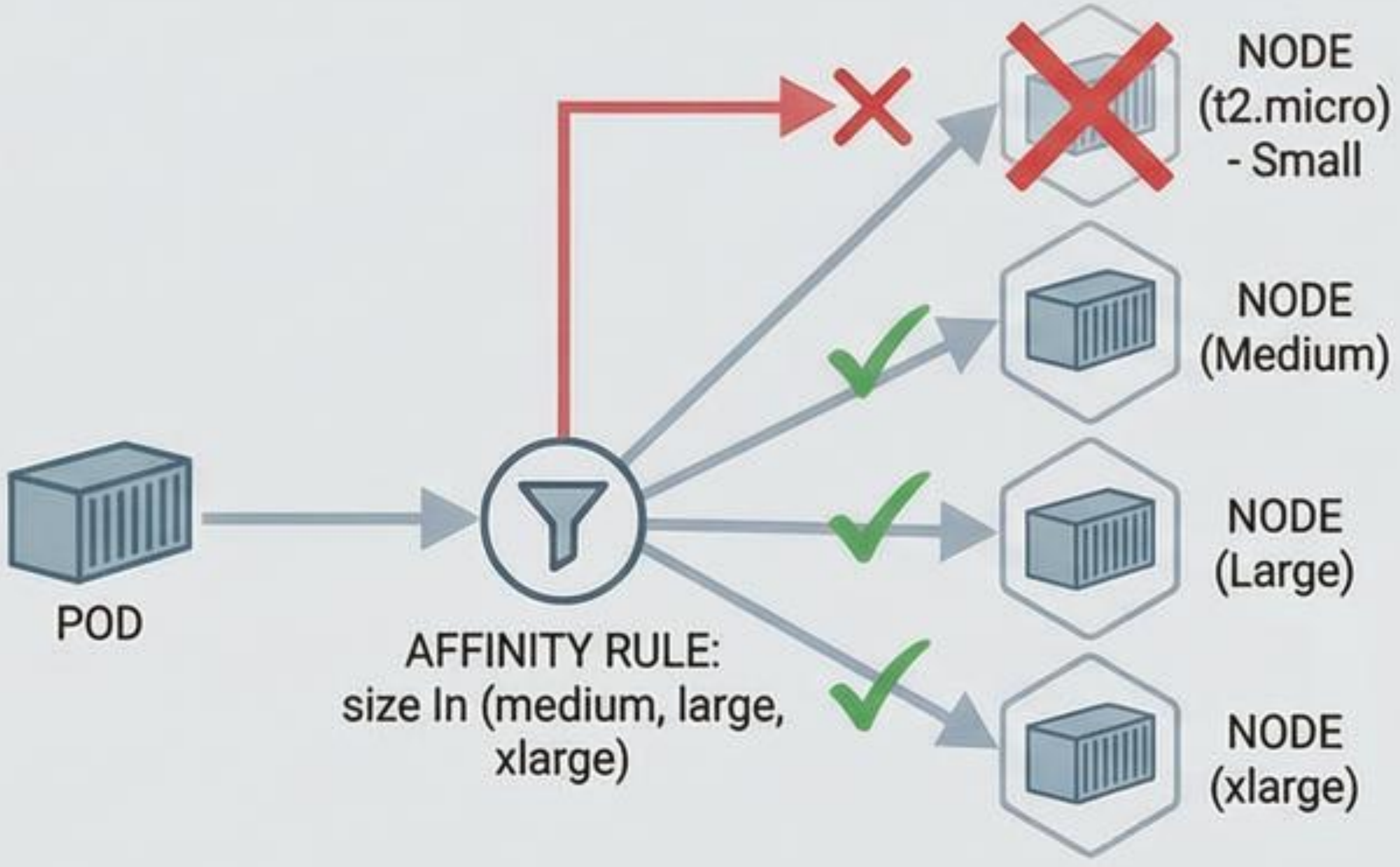
Eliminates all associated resources.

**PRO TIP:** Consistent cleanup habits avoid surprise affinity matches during later demos and keep GitOps diffs minimal.

# KUBERNETES NODE AFFINITY: MULTIPLE VALUES & OR LOGIC (REAL-WORLD USE CASES)

## SCHEDULING OPTIMIZATION: AVOIDING RESOURCE STARVATION

### PRODUCTION CLUSTERS: CAPACITY & LIFECYCLE



Avoid Small Instances. Widens placement pool to stable sizes while excluding t2.micro equivalents.

### AFFINITY LOGIC: SINGLE EXPRESSION, MULTIPLE TIERS

```
affinity:
  nodeAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
      nodeSelectorTerms:
      - matchExpressions:
          - key: size
            operator: In
            values:
              - medium
              - large
              - xlarge
```

- **PREVENTS RESOURCE STARVATION**: Ensures Pods have adequate resources, reducing eviction risk.
- **COMBINES ACCEPTABLE TIERS**: Illustrates how OR logic simplifies rules by grouping multiple acceptable values.
- **REDUCES EVICTION RISK**: Prioritizes stable, sufficient nodes over potentially unstable small instances.

PRESENTATION DESIGNED FOR CONSULTING-LEVEL COMMUNICATION

# Node Affinity Logic: Combining OR and AND within a Single Term

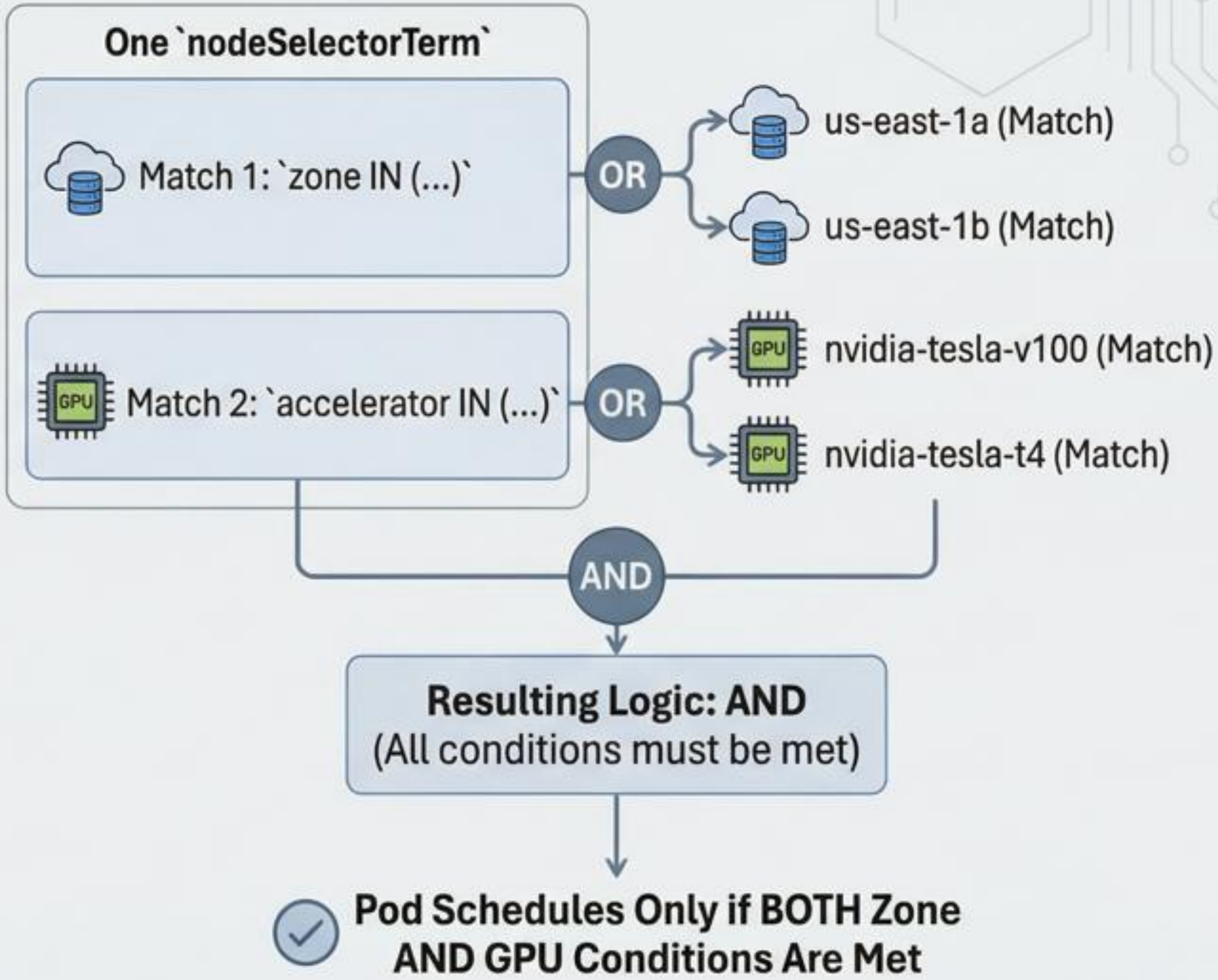
## Real-World Use Case: Multi-AZ, Multi-GPU Pod Scheduling

Goal: Spread across zones (OR) AND restrict to specific GPUs (AND)

```
nodeSelectorTerms:
- matchExpressions:
  - key: topology.kubernetes.io/zone
    operator: In
    values:
      - us-east-1a
      - us-east-1b
  - key: accelerator
    operator: In
    values:
      - nvidia-tesla-v100
      - nvidia-tesla-t4
```

OR Logic
(Fault Tolerance)

OR Logic
(Hardware Choice)



One `nodeSelectorTerm`

Match 1: `zone IN (...)`

Match 2: `accelerator IN (...)`

OR

us-east-1a (Match)

us-east-1b (Match)

OR

nvidia-tesla-v100 (Match)

nvidia-tesla-t4 (Match)

AND

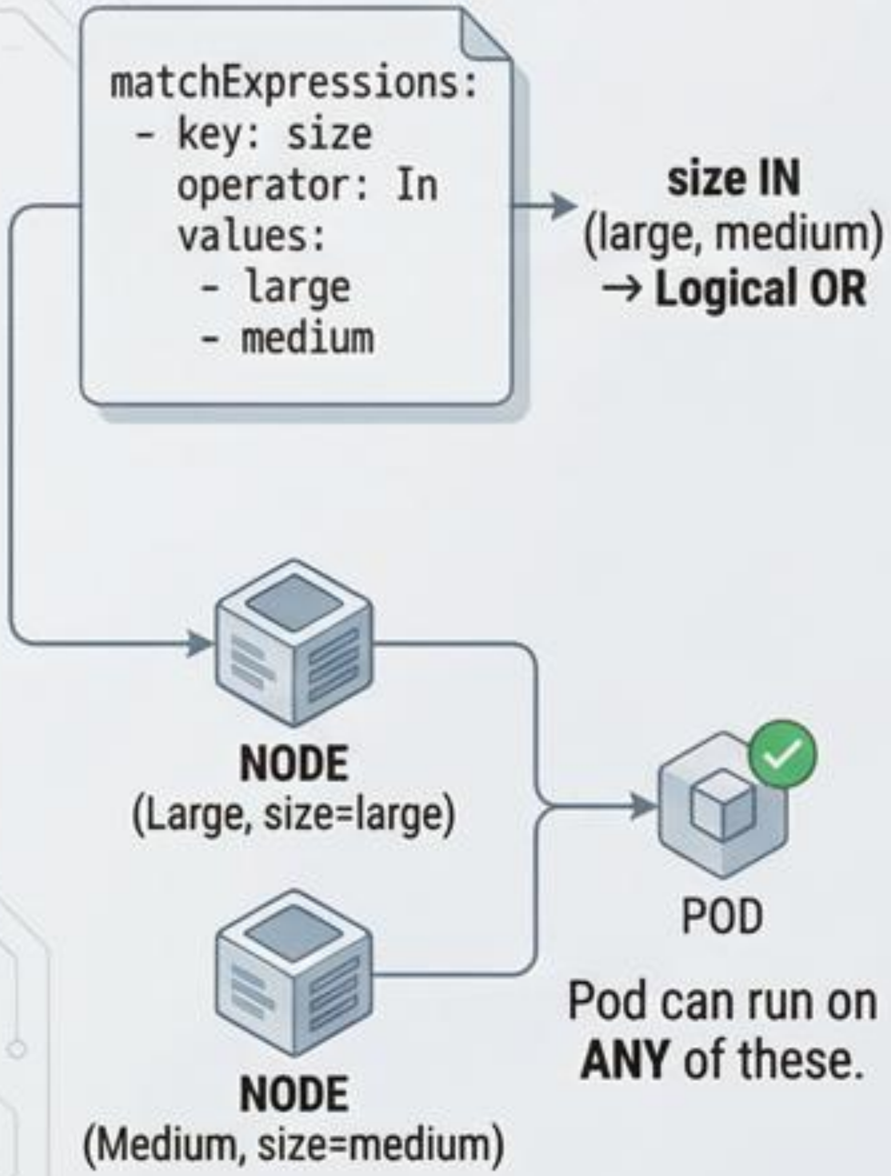
**Resulting Logic: AND**
(All conditions must be met)

**Pod Schedules Only if BOTH Zone AND GPU Conditions Are Met**

# KUBERNETES NODE AFFINITY: MULTIPLE VALUES & LOGIC OPTIMIZATION

Addressing Common Questions: OR, AND, Negation, and Combination.

## OR Logic: `values: [large, (Same `matchExpression`)

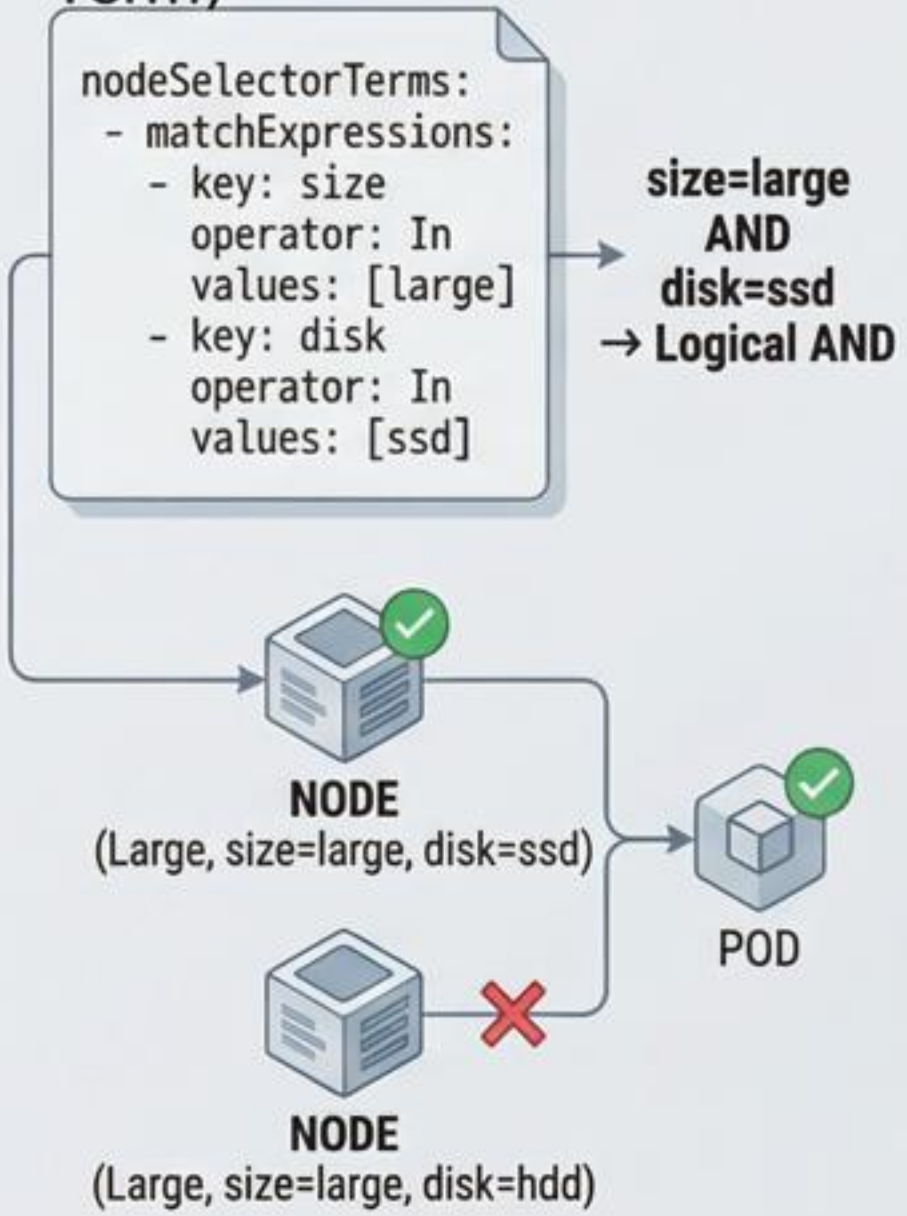


```
matchExpressions:
  - key: size
    operator: In
    values:
      - large
      - medium
```

Pod can run on **ANY** of these.

Key Insight: Single `matchExpression` with multiple values acts as OR.

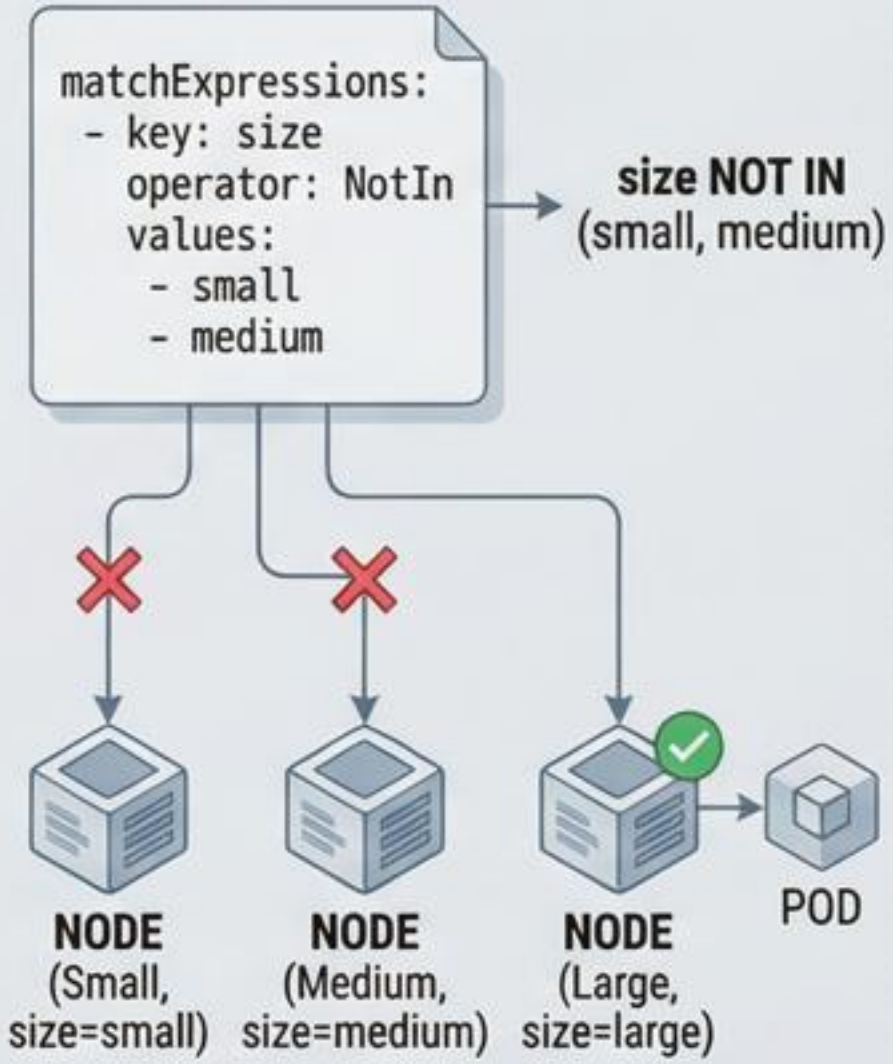
## AND Logic: Multiple `matchExpressions` (Same Term)



```
nodeSelectorTerms:
  - matchExpressions:
    - key: size
      operator: In
      values: [large]
    - key: disk
      operator: In
      values: [ssd]
```

Pod requires ALL conditions to be true.

## Negation: `operator: NotIn`



```
matchExpressions:
  - key: size
    operator: NotIn
    values:
      - small
      - medium
```

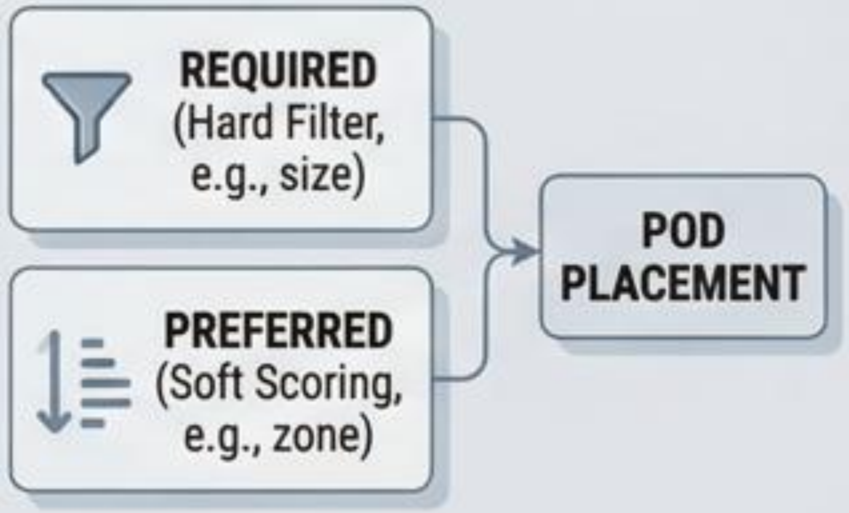
Explicitly excludes unwanted nodes (e.g., small, medium). Equivalent candidate set to In [large, xlarge] if only those exist.

## Practical Tips

**Value Order Irrelevant:** `values: [A, B]` is identical to `values: [B, A]`.

`[large, medium]` = `[medium, large]`

**Combining Required & Preferred:** Use both for fine-grained control.

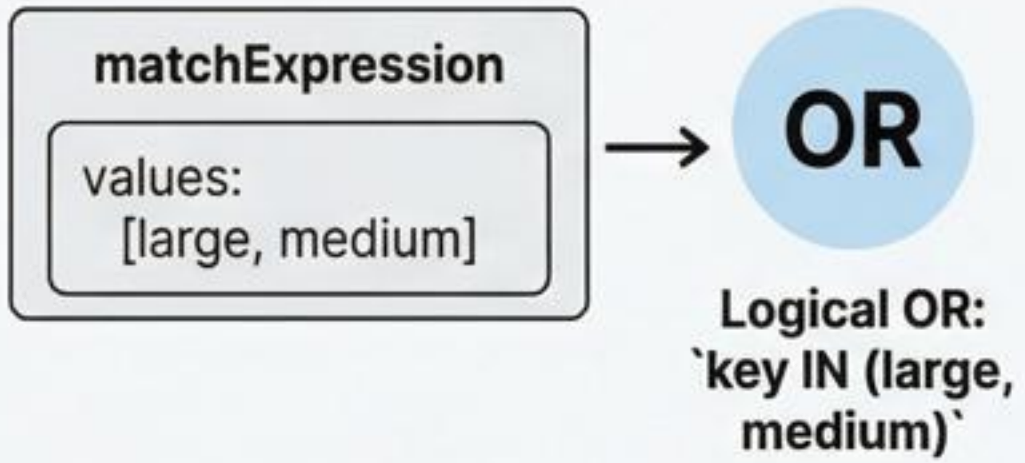


Implement hard constraints plus soft scoring for optimization.

PRESENTATION DESIGNED FOR CONSULTING-LEVEL COMMUNICATION

# Node Affinity Logic: The Three-Rule Mantra

## 1. Within a Single matchExpression



matchExpression

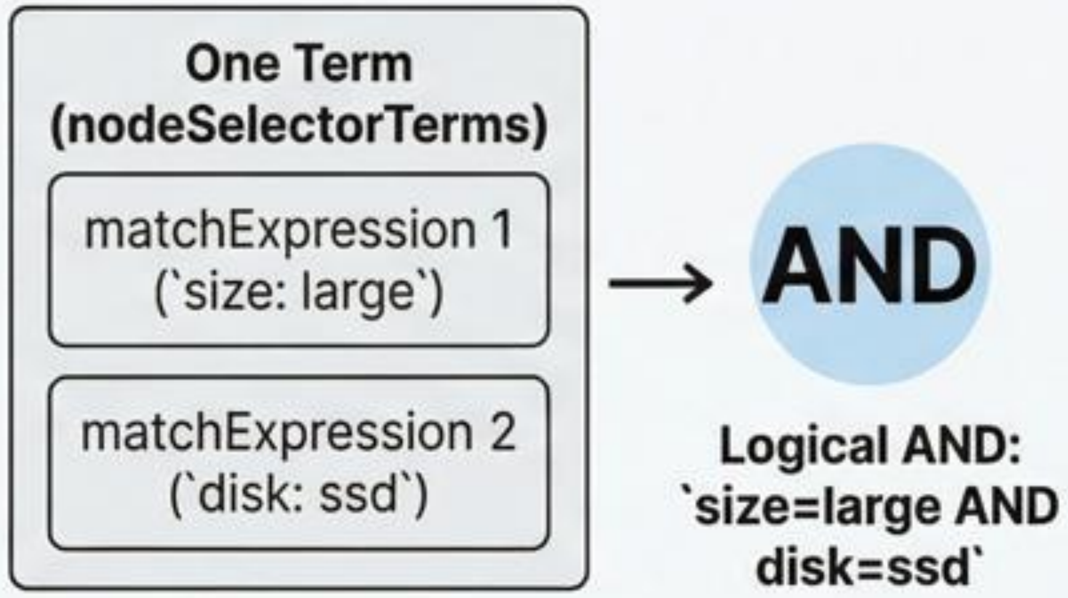
values:
[large, medium]

→ OR

Logical OR:
`key IN (large, medium)`

Use for OR on one label.

## 2. Across Multiple matchExpressions



One Term (nodeSelectorTerms)

matchExpression 1
(`size: large`)

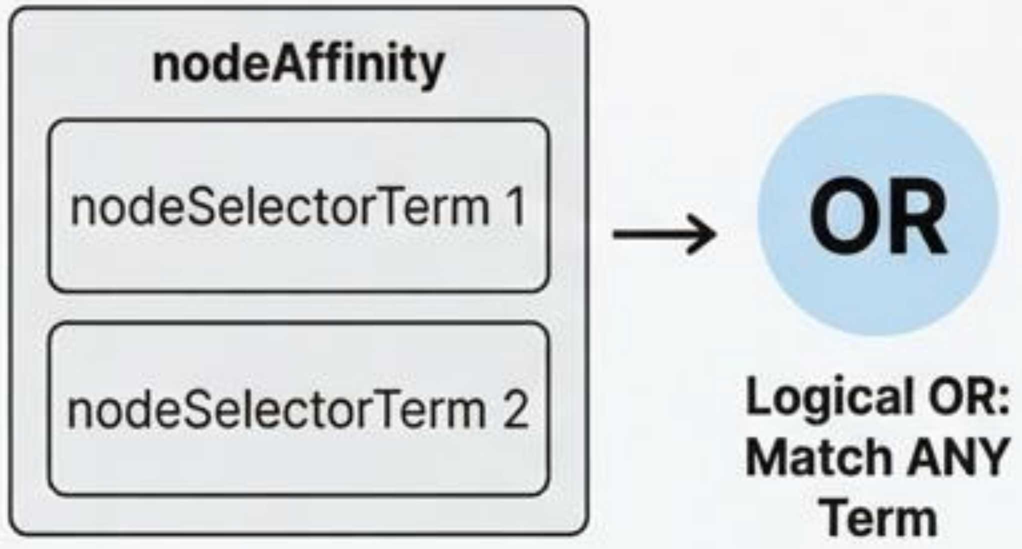
matchExpression 2
(`disk: ssd`)

→ AND

Logical AND:
`size=large AND disk=ssd`

Use for AND across labels.

## 3. Across Multiple nodeSelectorTerms



nodeAffinity

nodeSelectorTerm 1

nodeSelectorTerm 2

→ OR

Logical OR:
Match ANY Term

Use for OR between groups of conditions.

## Mastery & Benefits

- ✓ Read manifests quickly.
- ✓ Predict scheduling outcomes.
- ✓ Keep clusters reliable & intentions clear.