🧠 Kubernetes Resources – Requests & Limits (Deep Dive)

This document explains how **CPU and Memory Requests & Limits** work in Kubernetes, how the **scheduler and kubelet enforce them**, and what happens during **OOMKill and throttling**.

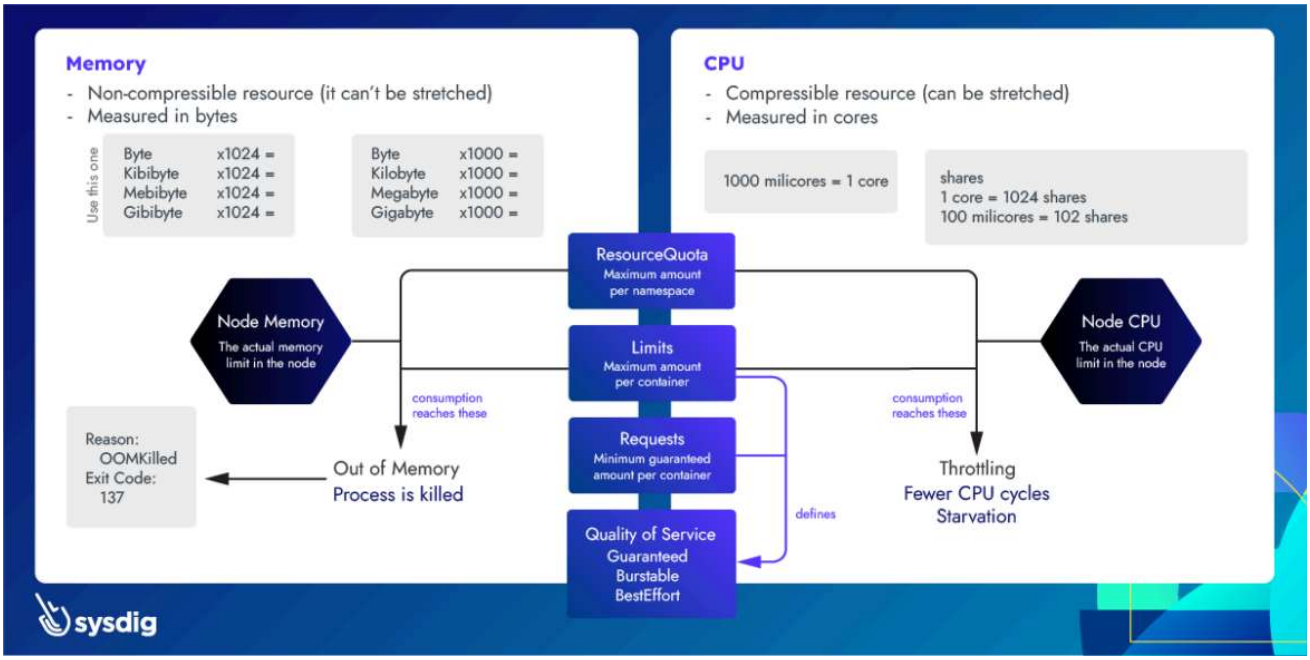
📚 Table of Contents

🧩 What Are Requests & Limits?

Resource	Request	Limit
CPU	Guaranteed minimum	Maximum allowed
Memory	Guaranteed minimum	Hard upper bound

> Request = Scheduling decision Limit = Runtime enforcement

Resource Management Architecture



The pod - Deployment.yaml

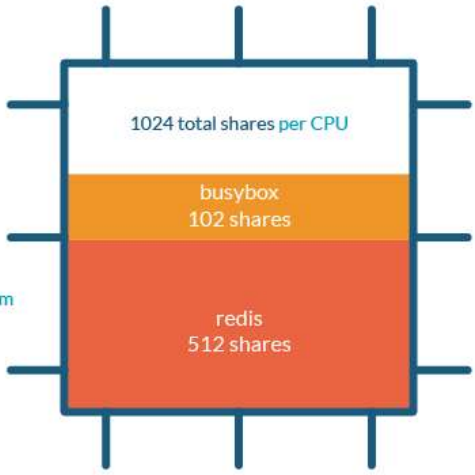
```yaml
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
 name: redis
…
 template:
   spec:
     containers:
       - name: redis
         image: redis:5.0.3-alpine
         resources:
           requests:
             memory: 300Mi
             cpu: 500m
…
       - name: busybox
         image: busybox:1.28
         resources:
           requests:
             memory: 100Mi
             cpu: 100m
```

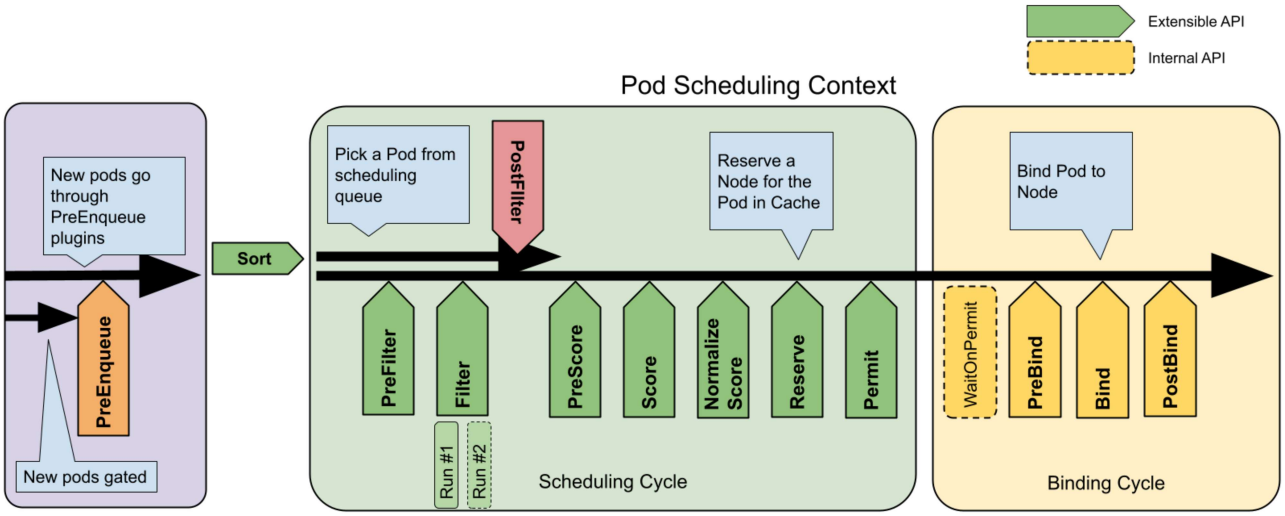
CPU Requests

Kubernetes assigns
1024 shares per core.
1 core = 1000 millicores = 1000m

1024 * 0.5 = 512 shares

1024 * 0.1 = 102 shares






```
Pod Spec
  |
  ▼
Scheduler ──(requests)──▶ Node selection
  |
  ▼
Kubelet ──(limits)──▶ cgroups
  |
  ▼
Linux Kernel (OOM / CPU throttling)
```

⚖️ Requests vs Limits (Critical Truth Table)

Scenario	Result
Request > Node capacity	Pod Pending
Memory > Limit	OOMKilled
CPU > Limit	Throttled
No requests	BestEffort Pod
No limits	Unlimited usage

📦 YAML Example – Requests & Limits (`request-limits.yml`)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: stress-test
  labels:
    tier: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: stress-test
  template:
    metadata:
      labels:
        app: stress-test
        tier: frontend
    spec:
      containers:
        - name: stress-test
          image: polinux/stress
          resources:
            requests:
              memory: "64Mi"
              cpu: "250m"
            limits:
              memory: "128Mi"
              cpu: "500m"
          command:
            ["stress", "--vm", "1", "--vm-bytes", "256M", "--timeout", "60s"]
```

Meaning:

- Pod **guaranteed** 200m CPU, 256Mi memory
- Pod **cannot exceed** 500m CPU, 512Mi memory

🧠 Scheduler Behavior (IMPORTANT)

Scheduler only looks at **requests**, never limits.

```
Node Capacity = 4 CPU / 8Gi
Pod Requests = 500m CPU / 1Gi
```

➡️ Scheduler allows **8 such pods** ➡️ Limits are ignored during scheduling

💥 Memory Over Limit – OOM Scenario

(request-limits-over-mem-limits.yml)

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: stress-test
  labels:
    tier: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: stress-test
  template:
    metadata:
      labels:
        app: stress-test
        tier: frontend
    spec:
      containers:
        - name: stress-test
          image: polinux/stress
          resources:
            requests:
              memory: "3000Mi"
              cpu: "250m"
            limits:
              memory: "3000Mi"
              cpu: "500m"
          command:
            ["stress", "--vm", "1", "--vm-bytes", "4096M", "--timeout", "60s"]
```

What happens?

- App crosses **256Mi**

- Kernel triggers **OOMKill**
- Pod restarts
- Status:

```
OOMKilled
```

Check:

```
kubectl describe pod <pod-name>
```

🧠 Memory Is NOT Throttled ❌

```
CPU → throttled
Memory → killed
```

There is **NO memory throttling in Linux**

🐢 CPU Limit – Throttling Scenario

```yaml
resources:
  limits:
    cpu: "500m"
```

When app uses more:

- Pod is **slowed down**
- No restart
- No error logs
- Hard to detect

Check throttling:

```
kubectl describe pod
```

Or Prometheus:

```
container_cpu_cfs_throttled_seconds_total
```

🎯 QoS Classes (VERY IMPORTANT)

QoS Class	Condition
Guaranteed	Requests = Limits
Burstable	Requests < Limits
BestEffort	No requests/limits

```
kubectl get pod <pod> -o jsonpath='{.status.qosClass}'
```

🚨 Real On-Call Scenarios

Issue	Root Cause
Pod Pending	Requests too high
Random restarts	Memory limit too low
Slow app	CPU throttling
Node OOM	No memory limits
HPA not working	Missing requests

🔍 Debugging Commands (SRE)

```
kubectl top pods
kubectl top nodes
kubectl describe pod <pod>
kubectl get events
```

Check OOM:

```
kubectl logs <pod> --previous
```

✅ Production Best Practices

✔ Always set **requests** ✔ Always set **memory limits** ✔ Avoid CPU limits for latency-sensitive apps ✔ Use **Burstable QoS** for web apps ✔ Use **Guaranteed QoS** for critical services ✔ Use **VPA recommendations** ✔ Monitor OOMKills aggressively

🧭 Recommended Values (Real-World)

App Type	CPU Request	CPU Limit	Memory
Web API	200m	❌	512Mi
Batch Job	500m	1	1Gi
Database	1	1	4Gi
JVM App	500m	❌	2Gi

> ❌ CPU limits often removed in production

🧠 Interview-Ready One-Liners

- Scheduler only cares about **requests**
- Memory limit breach = **OOMKill**
- CPU limit breach = **throttling**
- BestEffort pods die first
- Guaranteed pods die last