What are Resource Requests and Limits in Kubernetes?

In Kubernetes, **each container in a Pod can declare:**

- 🔶 **Resource Requests** → *Minimum* amount of CPU/memory guaranteed to the container.
- 🔶 **Resource Limits** → *Maximum* amount of CPU/memory a container is allowed to use.

🧠 Think of it like booking food at a buffet:

- **Request** = You book at least 1 plate of food for yourself — the system will reserve it.
- **Limit** = You can eat at most 2 plates — if you want more, you're not allowed.

🔍 CPU & Memory Units

Resource	Unit	Example
CPU	`millicores`	`500m` = 0.5 core
Memory	`Mi` or `Gi`	`128Mi`, `1Gi`

Mi = Mebibyte

- **Full form: Mebibyte**
- **1 MiB = 1,048,576 bytes = 2^{20} bytes**
- It's close to 1 MB (Megabyte), but not exactly the same.

Gi = Gibibyte

- **Full form: Gibibyte**
- **1 GiB = 1,073,741,824 bytes = 2^{30} bytes**
- It's close to 1 GB (Gigabyte), but slightly more.

Why use Mi and Gi instead of MB and GB?

Because:

- **MB (Megabyte) = 1,000,000 bytes** (decimal - base 10)
- **GB (Gigabyte) = 1,000,000,000 bytes**

So, to avoid confusion between binary and decimal sizes, standards like **MiB and GiB** were introduced.

Example on Kind Cluster

Let's define a Pod with CPU and memory requests/limits.

1️⃣ Create a Kind cluster

```
bash
----------
kind create cluster --name res-demo
```

2️⃣ Create a Pod with resource limits

`pod-resources.yaml`

```
yaml
----------
apiVersion: v1
kind: Pod
metadata:
  name: resource-demo
spec:
  containers:
    - name: busybox
      image: busybox
      command: ["sh", "-c", "while true; do echo running; sleep 5; done"]
      resources:
        requests:
          memory: "64Mi"
          cpu: "250m"
        limits:
          memory: "128Mi"
          cpu: "500m"
```

3️⃣ Apply the Pod

```
bash
----------
kubectl apply -f pod-resources.yaml
```

Check the pod:

```
bash
----------
kubectl get pod resource-demo
kubectl describe pod resource-demo
```

You'll see the memory and CPU requests/limits under **Containers > Resources**.

4️⃣ Verify via Metrics (Optional - Requires Metrics Server)

If you want to monitor actual usage, install the Kubernetes metrics server:

```
bash

kubectl apply -f https://github.com/kubernetes-sigs/metrics-
server/releases/latest/download/components.yaml


----------
kubectl top pod resource-demo
```

(Note: metrics-server doesn't come pre-installed on Kind, extra setup needed.)

⚠️ What happens if usage exceeds limits?

- **If memory exceeds limit** → The container is **killed** (OOMKilled).
- **If CPU exceeds limit** → The container is **throttled**, not killed.

💡 Bonus: Why Requests & Limits Matter?

- Kubernetes uses **requests** to decide **where to schedule** the pod.
- **Limits** help prevent one container from hogging all the resources in the node.

🧹 Cleanup

```
bash
----------
kind delete cluster --name res-demo
```

simulate an OOM (Out of Memory) error

⚠️ Goal

- We will create a Pod with:
  - **Memory limit = 100Mi**

- Then we'll run a script inside the container that tries to **allocate 200Mi of memory**.
- This will **crash** the container and you'll see `OOMKilled` in pod status.

🛠️ Step-by-Step: Simulate OOMKilled in Kind

1️⃣ Start your Kind cluster (if not already running)

```
bash
----------
kind create cluster --name oom-demo
```

2️⃣ Create a pod with low memory limit

```
yaml
----------
# pod-oom.yaml
apiVersion: v1
kind: Pod
metadata:
  name: oom-test
spec:
  containers:
    - name: memory-eater
      image: busybox
      command: ["sh", "-c", "x=$(head -c 200M </dev/zero | tr '\0' 'x'); sleep
300"]
      resources:
        limits:
          memory: "100Mi"
        requests:
          memory: "50Mi"
```

Save as `pod-oom.yaml`, then apply:

```
bash
----------
kubectl apply -f pod-oom.yaml
```

3️⃣ Watch the pod status

```
bash
----------
kubectl get pod oom-test -w
```

After a few seconds, the pod will restart with `STATUS: CrashLoopBackOff`.

4️⃣ Confirm it was OOMKilled

```
bash
----------
```

```
kubectl describe pod oom-test
```

Look for something like:

```
makefile
----------
State: Terminated
Reason: OOMKilled
```

🧹 Cleanup

```
bash
----------
kubectl delete pod oom-test
kind delete cluster --name oom-demo
```

2What happens if a container exceeds its CPU limit in Kubernetes?

A. It gets more CPU resources automatically
B. It is throttled and restricted from using more CPU
C. It crashes immediately
D. The Pod is terminated

Answer: B. It is throttled and restricted from using more CPU

Which of the following statements is true about memory limits in Kubernetes?

A. Exceeding memory limit causes throttling
B. Exceeding memory limit has no effect
C. Exceeding memory limit causes the container to be killed
D. Memory limits are optional and cannot be enforced

Answer: C. Exceeding memory limit causes the container to be killed

What is the default behavior if you don't specify resource requests and limits for a container?

A. The container is not allowed to run
B. Kubernetes assigns zero resources
C. The container can use any amount of resources on the node
D. Kubernetes assigns random values

Answer: C. The container can use any amount of resources on the node