Resource Limits

In **Kubernetes**, **resource limits**

* refer to the **maximum amount of compute resources** (like CPU and memory) that a **container** is allowed to use.
* setting resource limits helps ensure **fair resource sharing**, **application stability**, and **prevents one container from affecting others** by consuming excessive system resources.

**🔹 Why Set Resource Limits?**

Without limits, a container can use **as much CPU or memory as it wants**, which can:

* **Starve other pods** of resources
* **Cause node instability or crashes**
* **Increase cost** in cloud environments

**🔹 Key Resource Types**

Kubernetes supports two main types of resources:

| **Resource** | **Unit** | **Description** |
| --- | --- | --- |
| cpu | Cores or millicores (1, 0.5, 100m) | CPU time |
| memory | Bytes (Mi, Gi, etc.) | RAM usage |

**🔹 requests vs limits**

| **Term** | **Meaning** |
| --- | --- |
| **requests** | The amount of CPU/memory **guaranteed** to the container.  Kubernetes will **reserve** these resources on a node during scheduling. |
| **limits** | The **maximum** CPU/memory the container is allowed to use.  The container **cannot exceed** these values during execution. |

A screenshot of a computer program

AI-generated content may be incorrect.

**🔹 Default Resource Limits (Optional)**

Cluster administrators can define **default requests and limits** using a **LimitRange**:

A screen shot of a computer

AI-generated content may be incorrect.

This applies to **all pods** in the **dev namespace** that don't specify their own limits.

**✅ Best Practices**

* Always set **requests and limits** for critical workloads.
* Monitor usage with tools like **Prometheus**, **Grafana**, or **kubectl top**.