What’s the diffrence between (Request and Limit)

**Kubernetes defines Limits as the** **maximum amount of a resource** to be used by a container. This means that the container can never consume more than the memory amount or CPU amount indicated.

**Requests, on the other hand, are the minimum guaranteed amount of a resource** that is reserved for a container.

**Requests**

* **Purpose:** Requests are used to specify the minimum amount of resources a container requires. Kubernetes uses these values to schedule pods onto nodes. By declaring a resource request, you ensure that the pod is only scheduled on a node that has enough resources to meet this requirement.
* **Impact on Scheduling**:

The scheduler considers the total resource requests of all pods when determining if a node can accommodate a new pod. If the sum of resource requests for all running pods plus the new pod exceeds the resources available on a node, the new pod won’t be scheduled on that node.

* **Example**: If a container has a CPU request of 500m (which stands for 0.5 CPU cores) and 256Mi of memory, Kubernetes ensures that at least these amounts of CPU and memory are allocated to the container when it runs.

**Limits**

* **Purpose**: Limits are used to specify the maximum amount of resources a container can consume. Setting limits helps prevent a single container from monopolizing resources and impacting other containers running on the same node.
* **Impact on Resource Usage**: If a container exceeds its memory limit, it might be terminated (OOMKilled—Out of Memory Killed) and potentially restarted. For CPU, exceeding the limit doesn’t terminate the container but results in throttling, where the container’s CPU usage is restricted to the limit specified.
* **Example:** If a container has a CPU limit of 1 CPU and 512Mi of memory, it can use up to 1 CPU core and 512Mi of memory. If it tries to use more memory than 512Mi, it will be killed and restarted by Kubernetes. If it uses more than 1 CPU core, the CPU usage will be throttled, potentially affecting performance.

**How Requests and Limits Work Together**

1. **Resource Guarantees**: The combination of requests and limits helps provide a balance between resource guarantees and protection against resource overconsumption. Requests ensure that a container gets the resources it needs to run effectively, while limits protect the cluster by preventing any single container from using too many resources.
2. **Quality of Service (QoS)**: Kubernetes assigns a QoS class to pods based on the resource requests and limits specified. The QoS classes are:
   * **Guaranteed**: If both requests and limits for CPU and memory are set and they are equal, the pod is assigned to this class. Guaranteed pods are less likely to be evicted in case of resource pressure.
   * **Burstable**: If requests and limits are set but are not equal, or if only requests are set, the pod is in this class. It has more flexibility but might be evicted under heavy resource pressure.
   * **BestEffort**: If neither requests nor limits are set, the pod falls into this class. It has the lowest priority for resource allocation and is the first to be evicted in times of resource shortage.

**Practical Considerations**

* **Setting Requests and Limits**: It’s a good practice to set both requests and limits for your containers. This way, you can achieve both stable performance (through requests) and fair resource sharing (through limits).
* **Monitoring and Adjusting**: Continuously monitor the resource usage of your containers. If you notice that containers are consistently using resources beyond their requests but below their limits, you might need to adjust the values to better match actual usage patterns and avoid performance bottlenecks.