**What will happen if we don’t define any resources limit for CPU and memory ?**

If no resource requests (e.g., `cpu` and `memory`) are defined in a deployment manifest, the **kube-scheduler** will schedule the pods based on the **default behavior** and available node resources. Here's what happens in detail:

## 1. No Resource Requests

When no `resources.requests` or `resources.limits` are specified in the manifest:

- Kubernetes **does not reserve any specific amount of CPU or memory** for the pod.
- The pod is treated as if it requires minimal resources (essentially `0` CPU and memory requests).
- This means the pod can be scheduled onto any node with available capacity, even if the node is almost fully utilized.

---

## 2. Scheduling Behavior

The **kube-scheduler** attempts to find a suitable node for the pod using these steps:

### a. Filtering

The scheduler filters out nodes that cannot run the pod. Since no resource requests are defined, the scheduler doesn't exclude nodes based on resource capacity. Instead, other criteria like:

- Node taints and tolerations,
- Node selectors,
- Affinity/anti-affinity rules,
- Node readiness.

### b. Scoring

Among the filtered nodes, the scheduler scores them based on various factors, such as:

- Node utilization,
- Preferred affinity rules,
- Spreading pods across zones or nodes.

Since no resource requests are defined, node resource utilization will not significantly influence scoring.

### c. Assigning the Pod

The pod is assigned to the node with the highest score.

---

## 3. Risks of Not Defining Resources

While the scheduler can still schedule the pod, not defining resources introduces several risks:

### a. Overcommitment

- If a pod doesn’t specify resource requests, it might share resources with other pods on the same node, leading to resource contention.
- Nodes might run out of resources, causing performance degradation or instability.

### b. QoS Class

- Pods with no resource requests are classified as **BestEffort** quality of service (QoS) class.
- BestEffort pods are the first to be evicted if the node runs out of memory or CPU.

### c. Lack of Isolation

- Other pods or system processes on the same node can starve your pod of resources, as Kubernetes won't reserve any minimum amount for it.

---

## 4. Default Resource Requests (Optional)

If a **ResourceQuota** or **LimitRange** is defined in the namespace, Kubernetes may automatically assign default resource requests and limits to the pod. These defaults ensure the pod has at least minimal resources reserved and can be scheduled accordingly.

---

## 5. Best Practices

To avoid issues:

- Always define `resources.requests` for predictable scheduling and resource reservation.
- Define `resources.limits` to prevent pods from consuming excessive resources.
- Use namespace-level policies like **ResourceQuotas** or **LimitRanges** to enforce defaults and limits.

Example manifest with resource requests:

```yaml
resources:
  requests:
    memory: "256Mi"
    cpu: "500m"
  limits:
    memory: "512Mi"
    cpu: "1"
```

By defining these, you ensure that your pod is scheduled more reliably and performs consistently.