# Resource requests and limits

Ref - https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/

When you specify a Pod, you can optionally specify how much of each resource a container needs. The most common resources to specify are CPU and memory (RAM).

There are two types of resource types

### Resource requests

The kubelet reserves at least the *request* amount of that system resource specifically for that container to use.

### Resource limits

When you specify a resource *limit* for a container, the kubelet enforces those limits so that the running container is not allowed to use more of that resource than the limit you set.

## Difference b/w limits and requests

If the node where a Pod is running has enough of a resource available, it's possible (and allowed) for a container to use more resource than its request for that resource specifies. However, a container is not allowed to use more than its resource limit .

# Experiments

- 30% CPU usage on a single threaded Node.js app
  - Update the spec from the last slide to decrease the CPU usage. Notice [illegible] though this is a Node.js app where it can go up to 100%

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: cpu-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: cpu-app
  template:
    metadata:
      labels:
        app: cpu-app
    spec:
      containers:
      - name: cpu-app
        image: 100xdevs/week-28:latest
        ports:
        - containerPort: 3000
        resources:
          requests:
            cpu: "100m"
          limits:
            cpu: "300m"
```

Try hitting the server

```
    Type:                    Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   Burstable
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s

Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  17s   default-scheduler  0/2 nodes are available: 2 Insufficient cpu. preemption: 0/2 nodes are available: 2 No pre
emption victims found for incoming pod.
  Normal   NotTriggerScaleUp  3s    cluster-autoscaler  pod didn't trigger scale-up: 1 max node group size reached
→  week-28 git:(main) ✗
```

▼ Request 2 vCPU in 10 replicas

Try requesting more resources than available in the cluster.

```yaml
apiVersion: apps/v1
kind: Deployment
```

```yaml
spec:
```

```yaml
  replicas: 10
  selector:
    matchLabels:
      app: cpu-app
  template:
    metadata:
      labels:
        app: cpu-app
    spec:
      containers:
      - name: cpu-app
        image: 100xdevs/week-28:latest
        ports:
        - containerPort: 3000
        resources:
          requests:
            cpu: "1000m"
          limits:
            cpu: "1000m"
```

```
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   Burstable
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type     Reason            Age   From               Message
  ----     ------            ----  ----               -------
  Warning  FailedScheduling  31s   default-scheduler  0/2 nodes are available: 2 Insufficient cpu. preemption: 0/2 nodes are available: 2 No pre
emption victims found for incoming pod.
  Warning  FailedScheduling  29s   default-scheduler  0/2 nodes are available: 2 Insufficient cpu. preemption: 0/2 nodes are available: 2 No pre
emption victims found for incoming pod.
  Normal   NotTriggerScaleUp  20s   cluster-autoscaler  pod didn't trigger scale-up:
➜  week-28 git:(main) ✗
```