Resource Quotas

- When a Kubernetes cluster is used by multiple **people** or **teams**, **resource management** becomes more important
    - You want to be able to **manage the resources** you give to a person or a team
    - You don't want one person or team **taking up all the resources** (e.g. CPU/Memory) of the cluster
- You can divide your cluster in **namespaces** (explained in next lecture) and enable resource quotas on it
    - You can do this using the **ResourceQuota** and **ObjectQuota** objects

Resource Quotas

- The administrator can set the following resource limits within a namespace:

Resource	Description
requests.cpu	The sum of **CPU requests** of all pods cannot exceed this value
requests.mem	The sum of **MEM requests** of all pods cannot exceed this value
requests.storage	The sum of **storage requests** of all persistent volume claims cannot exceed this value
limits.cpu	The sum of **CPU limits** of all pods cannot exceed this value
limits.memory	The sum of **MEM limits** of all pods cannot exceed this value

Resource Quotas

- The administrator can set the following object limits:

Resource	Description
configmaps	total number of **configmaps** that can exist in a namespace
persistentvolumeclaims	total number of **persistent volume claims** that can exist in a namespace
pods	total number of **pods** that can exist in a namespace
replicationcontrollers	total number of **replicationcontrollers** that can exist in a namespace
resourcequotas	total number of **resource quotas** that can exist in a namespace
services	total number of **services** that can exist in a namespace
services.loadbalancer	total number of **load balancers** that can exist in a namespace
services.nodeports	total number of **nodeports** that can exist in a namespace
secrets	total number of secrets that can exist in a namespace

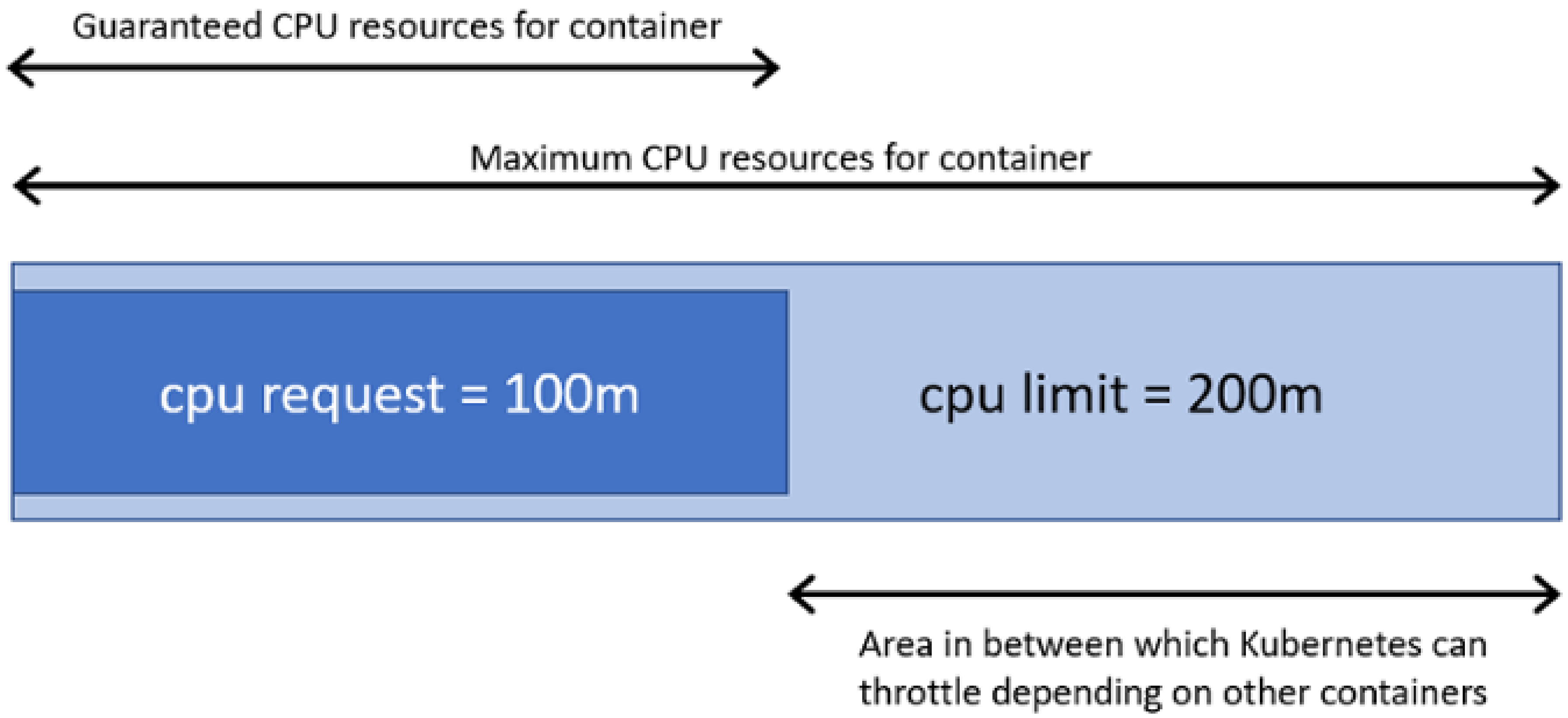
Resource Quotas

- If a capacity quota (e.g. mem / cpu) has been specified by the administrator, then each pod needs to specify capacity quota during creation
  - The administrator can specify default request values for pods that don't specify any values for capacity
  - The same is valid for limit quotas
- If a resource is requested more than the allowed capacity, the server API will give an error 403 FORBIDDEN - and kubectl will show an error

Resource Quotas

- Example of resource quotas:
    - You run a **deployment** with a **pod** with a **CPU resource** request of **200m**
    - 200m = 200 millicpu (or also 200 millicores)
    - 200m = 0.2, which is 20% of a CPU core of the running node
        - If the node has 2 cores, it's still 20% of a single core
    - You can also put a limit, e.g. on 400m
    - Memory quotas are defined by MiB or GiB

Quota and Limits


Guaranteed CPU resources for container
Maximum CPU resources for container
cpu request = 100m
cpu limit = 200m
Area in between which Kubernetes can throttle depending on other containers

Resource Quotas

- Each container can specify **request capacity** and **capacity limits**
    - **Request capacity** is an **explicit request** for resources
        - The scheduler can use the **request capacity** to make decisions on where to put the pod on
        - You can see it as a **minimum amount of resources the pod needs**
    - **Resource limit** is a limit imposed to the container
        - The container will not be able to utilize more resources than specified

Namespaces

- You can then create resource limits within that namespace:

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: compute-resources
  namespace: myspace
spec:
  hard:
    requests.cpu: "1"
    requests.memory: 1Gi
    limits.cpu: "2"
    limits.memory: 2Gi
```

Namespaces

- You can also create object limits:

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: object-counts
  namespace: myspace
spec:
  hard:
    configmaps: "10"
    persistentvolumeclaims: "4"
    replicationcontrollers: "20"
    secrets: "10"
    services: "10"
    services.loadbalancers: "2"
```

THANK YOU!