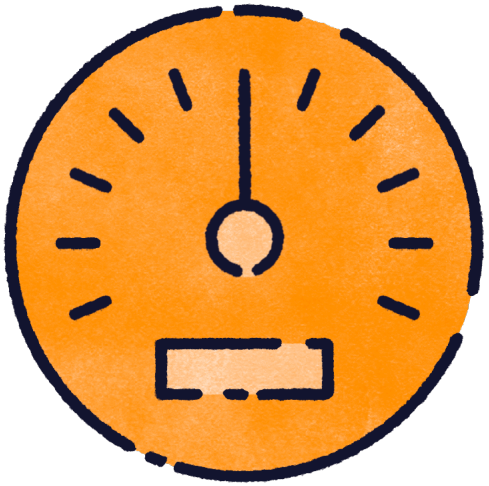
Understand Requests, Limits, and Quotas

Elle Krout
Principal Course Author, Pluralsight

Kubernetes Requests and Limits

Resource Requests and Limits

Requests



Limits

Resource Requests

Desired amount of resources for pod to run

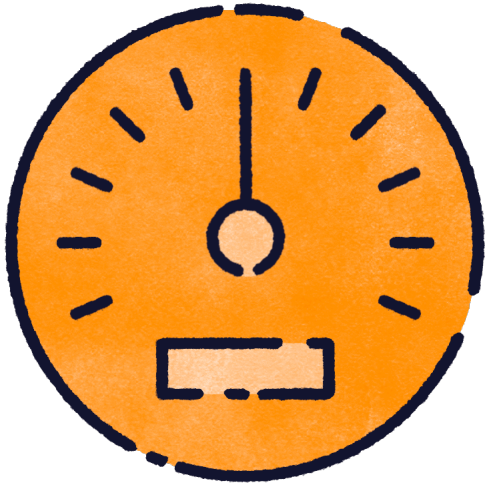
Minimum amount of:

- CPU
- Memory

Used during scheduling cycle to determine worker node

Ensures pod is guaranteed needed resources for its lifecycle

Resource Limits



Maximum about of resources that the pod can use

- CPU
- Memory

Too much memory used: Pod is terminated

Too much CPU used: Pod is throttled

Limits must be higher than requests

Resource Requirements and Limit Benefits

Stable pod performance

Effective resource sharing

Better pod scheduling

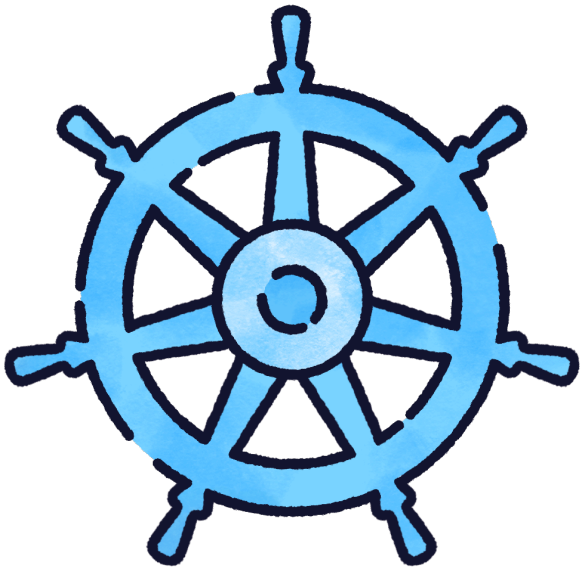
Add Requests and Limits

limited-pod.yaml

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: limited-pod
spec:
  containers:
  - name: app
    image: nginx
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

Exam Tip!

Look for troubleshooting issues such as pods being terminated due to memory or CPU throttling, which can be related pod request and limit definitions.



Kubernetes Requests and Limits

Let you provide Kubernetes with information so it can best place pods on worker nodes, ensuring everything runs smoothly

Kubernetes Limit Ranges

Limit Range Constraints

Minimum and maximum CPU for namespace

Minimum and maximum memory for namespace

Minimum and maximum storage requests per PVC for namespace

Default requests/limits for pods and containers within namespace

LimitRange Controller

1. Pod is created
2. If no requests or limits are defined, applied defaults
3. Checks requests and limit against limit ranges
4. ✗ If requests and limits are not within range: Stops request
5. ✓ If requests and limits are within range: Creates object

Create Limit Range

container-limits.yaml

```yaml
apiVersion: v1
kind: LimitRange
metadata:
  name: container-limits
  namespace: dev
spec:
  limits:
    - type: Container
      max:
        cpu: "1"
        memory: "512Mi"
      min:
        cpu: "100m"
        memory: "128Mi"
      default:
        cpu: "500m"
...
```

Create Limit Range

container-limits.yaml

```yaml
spec:
  limits:
    - type: Container
      max:
        cpu: "1"
        memory: "512Mi"
      min:
        cpu: "100m"
        memory: "128Mi"
      default:
        cpu: "500m"
        memory: "256Mi"
      defaultRequest:
        cpu: "200m"
        memory: "256Mi"
    - type: PersistentVolumeClaim
      max:
        storage: "5Gi"
      min:
        storage: "1Gi"
```

Create Limit Range

container-limits.yaml

```yaml
spec:
  limits:
    - type: Pod
      max:
        cpu: "1"
        memory: "512Mi"
      min:
        cpu: "100m"
        memory: "128Mi"
      default:
        cpu: "500m"
        memory: "256Mi"
      defaultRequest:
        cpu: "200m"
        memory: "256Mi"
    - type: PersistentVolumeClaim
      max:
        storage: "5Gi"
      min:
        storage: "1Gi"
```

Create Limit Range

container-limits.yaml

```yaml
spec:
  limits:
    - type: Pod
      max:
        cpu: "1"
        memory: "512Mi"
      min:
        cpu: "100m"
        memory: "128Mi"
      default:
        cpu: "500m"
        memory: "256Mi"
      defaultRequest:
        cpu: "200m"
        memory: "256Mi"
    - type: PersistentVolumeClaim
      max:
        storage: "5Gi"
      min:
        storage: "1Gi"
```

Create Limit Range

container-limits.yaml

```yaml
spec:
  limits:
    - type: Pod
      max:
        cpu: "1"
        memory: "512Mi"
      min:
        cpu: "100m"
        memory: "128Mi"
      default:
        cpu: "500m"
        memory: "256Mi"
      defaultRequest:
        cpu: "200m"
        memory: "256Mi"
    - type: PersistentVolumeClaim
      max:
        storage: "5Gi"
      min:
        storage: "1Gi"
```

View Limit Range Usage

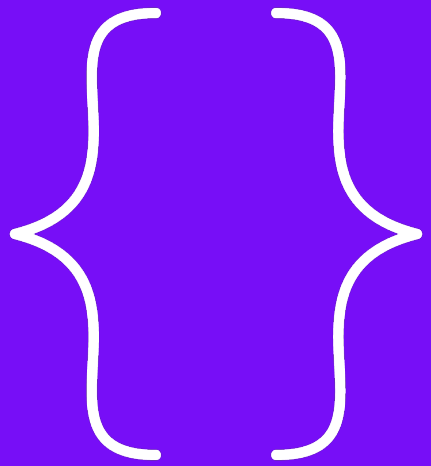
```
> kubectl describe limits <limit_name> --namespace <namespace>

Name:                   container-limits
Namespace:              dev2
Type                    Resource  Min    Max    Default Request  Default Limit  Max
Limit/Request Ratio
----                    --------  ---    ---    ---------------  -------------
-----------------------
Container               cpu       100m   1      200m             500m           -
Container               memory    128Mi  512Mi  256Mi            256Mi          -
PersistentVolumeClaim   storage   1Gi    5Gi    -                -              -
```

View Limit Ranges

```
> kubectl get limits --namespace <namespace>

NAME               CREATED AT
container-limits   2025-08-11T18:47:30Z
```



Limit Ranges

Work as a safeguard, ensuring workloads operate within defined and predictable boundaries

Kubernetes Resource Quotas

Resource Quotas

Limit total number of defined objects per namespace

Limit total amount of CPU/memory used per namespace

Limit total storage volume size per namespace

Create Resource Quota

dev-quota.yaml

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: dev-quota
  namespace: dev
spec:
  hard:
    pods: "10"
    requests.cpu: "4"
    requests.memory: "8Gi"
    limits.cpu: "8"
    limits.memory: "16Gi"
```

Create Resource Quota

Add Scope

dev-quota.yaml

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: dev-quota
  namespace: dev
spec:
  hard:
    pods: "10"
    requests.cpu: "4"
    requests.memory: "8Gi"
    limits.cpu: "8"
    limits.memory: “16Gi"
  scopes:
    - BestEffort
```

Resource Quota Scopes

Scope	Matches
`Terminating`	Pods where `.spec.activeDeadlineSeconds >= 0`
`NotTerminating`	Pods where `.spec.activeDeadlineSeconds` is `nil`
`BestEffort`	Pods that have best effort quality of service
`NotBestEffort`	Pods that do not have best effort quality of service
`PriorityClass`	Pods that references the specified priority class
`CrossNamespacePodAffinity`	Pods that have cross-namespace pod affinity terms
`VolumeAttributesClass`	PersistentVolumeClaims that reference the specified volume attributes class

Create Resource Quota

Add Scope

dev-quota.yaml

```yaml
apiVersion: v1
kind: ResourceQuota
metadata:
  name: dev-quota
  namespace: dev
spec:
  hard:
    pods: "10"
    requests.cpu: "4"
    requests.memory: "8Gi"
    limits.cpu: "8"
    limits.memory: “16Gi"
  scopes:
    scopeSelector:
      matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values: ["medium"]
```

View Limit Range Usage

```
> kubectl describe quota --namespace <namespace>

Name:       dev-quota
Namespace:  default
Resource    Used  Hard
--------    ----  ---
cpu         0     4
memory      0     8Gi
pods        0     10
```

Resource Quotas

Essential tool for controlling resource consumption

Ensures workloads stay within defined limits

Demo: Restricting Pod Creation with Requests, Limits, and Resource Quotas

Exam Scenario

Create a namespace, `staging`

This namespace should have no more than 15 total pods, whose combined resources should not exeed 1500m CPU and 2Gi memory

Individual resources within this namespace should not exceed 1 CPU or 1Gi memory, with defaults requests set to 100m CPU and 128Mi memory; default limits should be 500m CPU and 512Mi memory

Once the namespace and related resource definitions have been created, attempt to deploy the pod found at `test-staging-pod.yaml`. Resolve any issues with the pod definition