- **Namespaces:**
  - Partition resources and provide isolation within a cluster.
  - Logically divide a cluster into multiple virtual clusters.
  - Each namespace has its own set of resources.
- **Resource Quotas:**
  - Limit the amount of resources that can be used within a namespace.
  - Help ensure that a namespace does not consume excessive resources.
  - Set limits on CPU, memory, and storage usage within a namespace.
  - Specify limits on the number of pods, services, and other resources that can be created.
- **Together:**
  - Effectively manage and allocate resources within a Kubernetes cluster.
  - Provide better isolation and control over resource usage.

Benefits of using namespaces and resource quotas

- Improve resource utilization
- Prevent resource conflicts
- Improve security
- Make it easier to manage large Kubernetes deployments

Lets do R & D ,

Create a namespace for testing,

Namespace name: vishalk17

- kubectl create ns vishalk17

```
vagrant@k8s-master:~$ kubectl create ns vishalk17
namespace/vishalk17 created
vagrant@k8s-master:~$
vagrant@k8s-master:~$ kubectl get ns
NAME              STATUS   AGE
kube-system       Active   21h
kube-public       Active   21h
kube-node-lease   Active   21h
default           Active   21h
vishalk17         Active   7s
vagrant@k8s-master:~$
```

1. Setting Resource Quotas for a Namespace:

Use the `ResourceQuota` YAML file (`quota.yaml`) when you want to enforce resource limits for a specific namespace. This can be useful in scenarios where you want to allocate a specific amount of resources to a namespace, preventing it from consuming more than the allocated resources. It helps in resource allocation control and ensuring fair sharing of resources among different namespaces.

Kubernets doc: https://kubernetes.io/docs/concepts/policy/resource-quotas/

When several users or teams share a cluster with a fixed number of nodes, there is a concern that one team could use more than its fair share of resources.

Resource quotas are a tool for administrators to address this concern.

A resource quota, defined by a `ResourceQuota` object, provides constraints that limit aggregate resource consumption per namespace. It can limit the quantity of objects that can be created in a namespace by type, as well as the total amount of compute resources that may be consumed by resources in that namespace.

Compute Resource Quota

You can limit the total sum of compute resources that can be requested in a given namespace.

The following resource types are supported:

Resource Name	Description
`limits.cpu`	Across all pods in a non-terminal state, the sum of CPU limits cannot exceed this value.
`limits.memory`	Across all pods in a non-terminal state, the sum of memory limits cannot exceed this value.
`requests.cpu`	Across all pods in a non-terminal state, the sum of CPU requests cannot exceed this value.
`requests.memory`	Across all pods in a non-terminal state, the sum of memory requests cannot exceed this value.
`hugepages-<size>`	Across all pods in a non-terminal state, the number of huge page requests of the specified size cannot exceed this value.
`cpu`	Same as `requests.cpu`
`memory`	Same as `requests.memory`

** Configure Memory and CPU Quotas for a Namespace **

how to set quotas for the total amount memory and CPU that can be used by all Pods running in a namespace.

Create a ResourceQuota

```
apiVersion: v1
kind: ResourceQuota
metadata:
  name: mem-cpu-demo
  namespace: vishalk17
spec:
  hard:
    requests.cpu: "1"
    limits.cpu: "2"

    requests.memory: 500Mi
    limits.memory: 1Gi
```

- save , exit, apply

The `requests.cpu` and `requests.memory` fields specify the minimum amount of CPU and memory that each pod in the namespace must request. If a pod does not specify a request for a resource, the scheduler will automatically assign the minimum request for that resource.

The `limits.cpu` and `limits.memory` fields specify the maximum amount of CPU and memory that each pod in the namespace can consume. If a pod tries to consume more than its limit, the kubelet will kill the pod.

```
vagrant@k8s-master:~$ kubectl get resourcequota -n vishalk17
NAME           AGE   REQUEST                                      LIMIT
mem-cpu-demo   97s   requests.cpu: 0/1, requests.memory: 0/500Mi   limits.cpu: 0/2, limits.memory: 0/1Gi
vagrant@k8s-master:~$
vagrant@k8s-master:~$ kubectl describe resourcequota -n vishalk17
Name:            mem-cpu-demo
Namespace:       vishalk17
Resource         Used  Hard
--------         ----  ----
limits.cpu       0     2
limits.memory    0     1Gi
requests.cpu     0     1
requests.memory  0     500Mi
vagrant@k8s-master:~$
```

So, all has been set it seems

Situation 1 : What if i dont set limit and request in pod manifest file

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
```

- Save , apply

```
vagrant@k8s-master:~$ kubectl apply -f nginx-pod.yml -n vishalk17
Error from server (Forbidden): error when creating "nginx-pod.yml": pods "nginx" is forbidden: failed quota: mem-cpu-demo: must specify
limits.cpu for: nginx; limits.memory for: nginx; requests.cpu for: nginx; requests.memory for: nginx
vagrant@k8s-master:~$
```

Noticeable Changes :

- Pod will not create until and unless you specify limit and request in manifest file of pod

Situation 2 : add limit and request to the pod

2.1 keeping limit and request value lower compared to resource quota

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    resources:
      limits:
        memory: "300Gi"
        cpu: "800m"
      requests:
        memory: "200Mi"
        cpu: "200m"
```

- Save and apply

```
vagrant@k8s-master:~$ kubectl apply -f nginx-pod.yml -n vishalk17
Error from server (Forbidden): error when creating "nginx-pod.yml": pods "nginx" is forbidden: exceeded quota: mem-cpu-demo, requested:
limits.memory=300Gi, used: limits.memory=0, limited: limits.memory=1Gi
vagrant@k8s-master:~$
```

Noticeable Changes :

- Pod will not create I assume its because of the limit we have lower below requested the one defined in RQ

2.2 keeping request equal and limits equal compared to RQ

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    resources:
      limits:
        memory: "1Gi"
        cpu: "2"
      requests:
        memory: "500Mi"
        cpu: "1"
```

```
vagrant@k8s-master:~$ kubectl apply -f nginx-pod.yml -n vishalk17
pod/nginx created
vagrant@k8s-master:~$ kubectl get pods -n vishalk17
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          8s
vagrant@k8s-master:~$ kubectl get resourcequota -n vishalk17
NAME           AGE   REQUEST                                      LIMIT
mem-cpu-demo   32m   requests.cpu: 1/1, requests.memory: 500Mi/500Mi   limits.cpu: 2/2, limits.memory: 1Gi/1Gi
vagrant@k8s-master:~$
```

Noticeable Changes :

- Pod created

2.3 keeping request equal and limits lower compared to RQ

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    resources:
      limits:
        memory: "800Mi"
        cpu: "1.5"
      requests:
        memory: "500Mi"
        cpu: "1"
```

- Save, apply

```
vagrant@k8s-master:~$
vagrant@k8s-master:~$ kubectl apply -f nginx-pod.yml -n vishalk17
pod/nginx created
vagrant@k8s-master:~$ kubectl get pods -n vishalk17
NAME    READY   STATUS    RESTARTS   AGE
nginx   1/1     Running   0          5s
vagrant@k8s-master:~$
```

Noticeable Changes :

- Pod created

Summary, in case you have defined RQ for namespace,

- In pod specification request must be equal to request defined in RQ else pod will not create
- Acceptable : In pod specification limit can be lower or equal to limit defined in RQ
- Pod will not create : if no limits defined in pod specification.

Situation 3 : What if I dont set Resource Quota for namespace and add limit and request to the pod

Check if I have any resource quota applied to the namespace or not , if applied then I will remove to achieve this situation.

```
vagrant@k8s-master:~$
vagrant@k8s-master:~$ kubectl get resourcequota -n vishalk17
No resources found in vishalk17 namespace.
vagrant@k8s-master:~$
```

Good to go.

First of all lets check how much cpu and memory using by particular container in a pod

- vi httpd.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-deployment
  labels:
    app: apache
spec:
  replicas: 1
  selector:
    matchLabels:
      app: apache
  template:
    metadata:
      labels:
        app: apache
    spec:
      containers:
      - name: apache
        image: httpd:latest
        ports:
        - containerPort: 80
```

- Save and apply

```
vagrant@k8s-master:~$
vagrant@k8s-master:~$ kubectl apply -f httpd.yml -n vishalk17
deployment.apps/apache-deployment created
vagrant@k8s-master:~$
vagrant@k8s-master:~$ kubectl top pods -n vishalk17
NAME                                CPU(cores)   MEMORY(bytes)
apache-deployment-6696dd9569-t5wkc   1m           6Mi
vagrant@k8s-master:~$
```

Situation 3.1 : What if I set limit memory to 4Mi and keep cpu same as 1m

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-deployment
  labels:
    app: apache
spec:
  replicas: 1
  selector:
    matchLabels:
      app: apache
  template:
    metadata:
      labels:
        app: apache
    spec:
      containers:
      - name: apache
        image: httpd:latest
        ports:
        - containerPort: 80
        resources:
          limits:
            memory: "4Mi"
            cpu: "10m"
          requests:
            memory: "3Mi"
            cpu: "1m"
```

```
vagrant@k8s-master:~$ kubectl get all -n vishalk17
NAME                                     READY   STATUS             RESTARTS      AGE
pod/apache-deployment-6d9b8fb897-f9k4m   0/1     CrashLoopBackOff   3 (26s ago)   102s

NAME                                 READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/apache-deployment    0/1     1            0           102s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/apache-deployment-6d9b8fb897   1         1         0       102s
vagrant@k8s-master:~$
```

Noticeable Changes :

- Pod created but container crashing , Reason dont have enough source to run apache

Situation 3.2 : What if I set limit memory to 10Mi than required 6Mi and keep cpu same

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-deployment
  labels:
    app: apache
spec:
  replicas: 1
  selector:
    matchLabels:
      app: apache
  template:
    metadata:
      labels:
        app: apache
    spec:
      containers:
      - name: apache
        image: httpd:latest
        ports:
        - containerPort: 80
        resources:
          limits:
            memory: "10Mi"
            cpu: "10m"
          requests:
            memory: "3Mi"
            cpu: "1m"
```

```
vagrant@k8s-master:~$ kubectl get all -n vishalk17
NAME                                    READY   STATUS    RESTARTS   AGE
pod/apache-deployment-6f6c4cccb8-pl7xn   1/1     Running   0          20s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/apache-deployment   1/1     1            1           20s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/apache-deployment-6f6c4cccb8   1         1         1       20s
vagrant@k8s-master:~$
vagrant@k8s-master:~$ kubectl top pods -n vishalk17
NAME                                 CPU(cores)   MEMORY(bytes)
apache-deployment-6f6c4cccb8-pl7xn   5m           6Mi
vagrant@k8s-master:~$
```

Noticeable Changes :

- Container created,

Situation 3.2 : What if I set limit not a request

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: apache-deployment
  labels:
    app: apache
spec:
  replicas: 1
  selector:
    matchLabels:
      app: apache
  template:
    metadata:
      labels:
        app: apache
    spec:
      containers:
      - name: apache
        image: httpd:latest
        ports:
        - containerPort: 80
        resources:
          limits:
            memory: "10Mi"
            cpu: "10m"
```

```
vagrant@k8s-master:~$ kubectl apply -f httpd.yml -n vishalk17
deployment.apps/apache-deployment created
vagrant@k8s-master:~$
vagrant@k8s-master:~$ kubectl get all -n vishalk17
NAME                                    READY   STATUS              RESTARTS   AGE
pod/apache-deployment-8dd5754dd-94qsz   0/1     ContainerCreating   0          9s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/apache-deployment   0/1     1            0           9s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/apache-deployment-8dd5754dd    1         1         0       9s
vagrant@k8s-master:~$ kubectl get all -n vishalk17
NAME                                    READY   STATUS    RESTARTS   AGE
pod/apache-deployment-8dd5754dd-94qsz   1/1     Running   0          15s

NAME                                READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/apache-deployment   1/1     1            1           16s

NAME                                           DESIRED   CURRENT   READY   AGE
replicaset.apps/apache-deployment-8dd5754dd    1         1         1       16s
vagrant@k8s-master:~$ kubectl top pods -n vishalk17
```

```
vagrant@k8s-master:~$ kubectl top pods -n vishalk17
NAME                                CPU(cores)   MEMORY(bytes)
apache-deployment-8dd5754dd-94qsz   1m           6Mi
vagrant@k8s-master:~$
```

Noticeable Changes :

- Container created, Limits = request

Summary :

request = mentioned
limit = mentioned
no issues (source can be used upto limits , above that container fail)

---

request = not mentioned
limit = mentioned
limit = request (source can be used upto limits , above that container fail)

---

request = mentioned
limit = not mentioned
unlimited resources (default to zero) (no limits defined, container will not fail and will not has limitation on using compute resources)

My devops repo :

- https://github.com/vishalk17/devops

My telegram channel:

- https://t.me/vishalk17_devops

Contact:

Telegram : t.me/vishalk17

vishalk17 My youtube Channel :

- YouTube https://www.youtube.com/@vishalk17