Lesson 3 : Reserve Compute Capacity for Applications

Configure an application with resource requests so Kubernetes can make scheduling decisions.

Kubernetes Pod Scheduling – Three step process

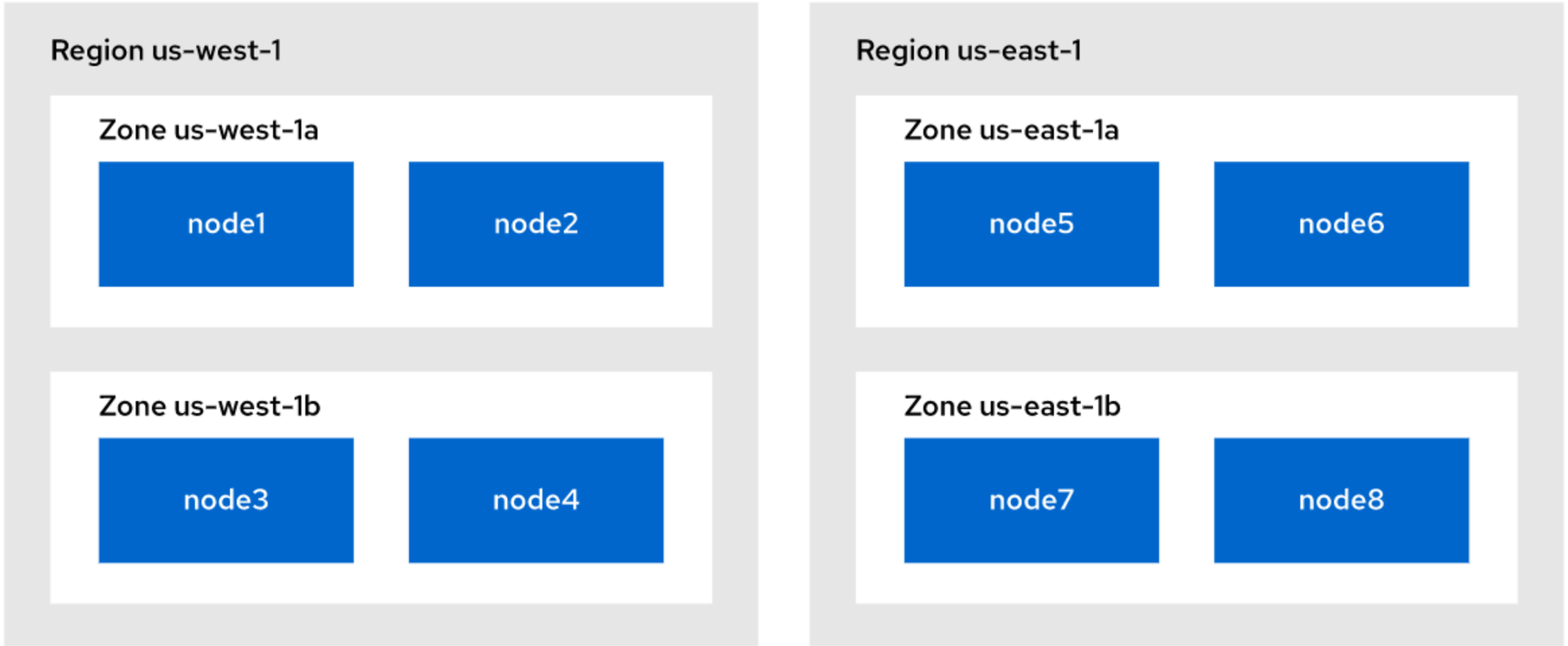
1. Filtering nodes.
   - Predicates : available host ports, disk or memory pressure
   - Node Selector and Node Labels
   - Compute resources requests: CPU, memory, storage
   - Taints and Tolerations:
   - ❖ Default: control plane include taint:

     node-role.kubernetes.io/master:NoSchedule
2. Priority
   - Use weighted score: higher value the better
   - Higher affinity have higher score
   - ❖ Affinity schedule related pods to be close to each other: Reason: Performance
   - ❖ Anti-affinity schedule related pods in different node: Reasons: HA design
3. Best fit
   - If multiple nodes share similar strength and scores: use round-robin fashion

☼ Scheduler can be customized for advanced scheduling situation

Predicates based on region and zone labels

- Affinity: Replica pods, create from same DC or deployment are scheduled to run on nodes having same **region** label
- Anti-Affinity: Replica pods are scheduled to run on nodes that have different values for **zone** label to achieve high availability



Compute resource units

- CPU is measured in units called millicores.
  - Amount of CPU cores on node multiples 1000 to express its total capacity.
  - For example, if a node has 2 cores, CPU capacity would be represented as 2000m.
- Memory is measured in bytes. (Mi, Gi, Ti).
  - 1000 bits vs 1 kilobyte (1024 bits).
- CPU / Memory REQUESTS:
  - Specify minimum amount of resources a container in pod may use on node.
  - Resource requests are used by OpenShift scheduler to find a node with an appropriate fit for your container.
  - It specifies minimum amount of resources that your container may consume.
- CPU / Memory LIMITS:
  - Specify maximum amount of resources a container in pod can use on node.

Defining Resource Requests and Limits for Pods

- A pod / deployment / DC definition can include both resource requests and resource limits:
- Resource requests
  - Used for scheduling
  - Indicate pod cannot run with less than specified amount of compute resources
- Resource limits
  - Used to prevent a pod from using up all cpu and memory from a node
  - Uses Linux Kernel : cgroups feature to enforce the pod's resource limits
- Quotas to track and limit per project level :
  - Object counts: pods, services, quotas, secrets, pvc and replicationcontrollers
  - Compute resources: cpu, memory
- If quota is used, then deployment should define resource request.
- Recommended: define requests and limits even if quotas are not used

Relativities

Quotas on project

cpu=5000m
memory=10G

deployment/app1

Limits cpu=1G, memory=5G

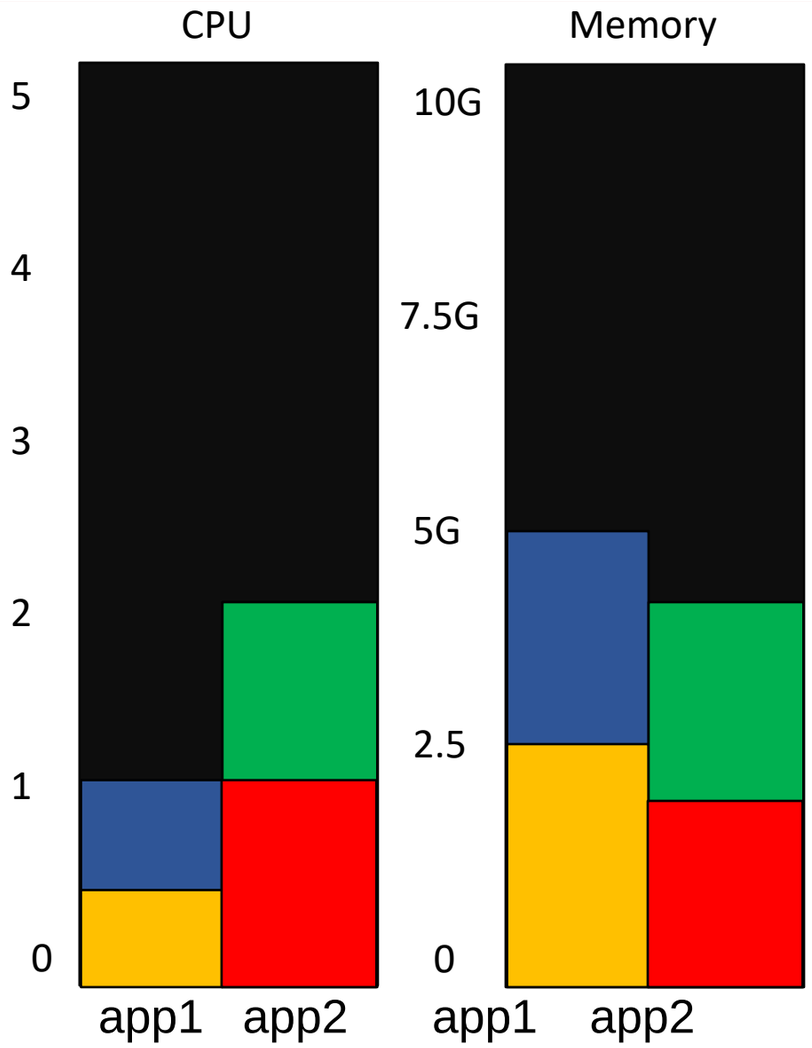
Requests cpu=500M, memory=2.5G

deployment/app2

Limits cpu=2G, memory=4G

Requests cpu=1G, memory=2G

- Total Limits from both applications must not go over quotas
- Requests set the minimum guaranteed value



Configure Resource Resources:

- Set following deployment with cpu and memory limits and requests:

```
$ oc set resources deployment myapp\
   --requests cpu=10m,memory=20Mi \
   --limits cpu=80m,memory=100Mi
```

- Should be defined in deployment or dc resource; not pod!
- Directly edit deployment CRD:

```
$ oc edit deployment/myapp
...output omitted...
resources:
  requests:
     cpu: 10m
     memory: 20Mi
  limits:
     cpu: 80m
     memory: 100Mi
```

- Verify configured value from deployment:

```
$ oc describe deployment/myapp | grep -i requests -A7
```

Inspecting Cluster Compute Resources

- **Only Cluster administrator** can view available and used compute resources of nodes

```
[user@host ~]$ oc describe node master01
Name:               master01
Roles:              control-plane,master,worker
...output omitted...
Capacity:
  cpu:                8
  ephemeral-storage:  125293548Ki
  hugepages-1Gi:      0
  hugepages-2Mi:      0
  memory:             20531668Ki
  pods:               250
Allocatable:
  cpu:                7500m
```

Inspecting Cluster Compute Resources

- **Only Cluster administrator** can view available and used compute resources of nodes

```
  ephemeral-storage:    114396791822
  hugepages-1Gi:        0
  hugepages-2Mi:        0
  memory:               19389692Ki
  pods:                 250
...output omitted...
Non-terminated Pods:                                  (88 in total)
  ... Name           CPU Requests  CPU Limits  Memory Requests  Memory Limits ...
  ... ----           ------------  ----------  ---------------  -------------
  ... controller-... 10m (0%)      0 (0%)      20Mi (0%)        0 (0%)        ...
  ... metallb-...    50m (0%)      0 (0%)      20Mi (0%)        0 (0%)        ...
  ... metallb-...    0 (0%)        0 (0%)      0 (0%)           0 (0%)        ...
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests      Limits
  --------           --------      ------
  cpu                3183m (42%)   1202m (16%)
  memory             12717Mi (67%) 1350Mi (7%)
...output omitted...
```

Inspect pod and node resource usage

- List usage of cpu and memory by **pods**

```
[user@host ~]$ oc adm top pods -n openshift-dns
NAME                  CPU(cores)   MEMORY(bytes)
dns-default-5kpn5     1m           33Mi
node-resolver-6kdxp   0m           2Mi
```

- List usage of cpu and memory by **nodes**

```
[user@host ~]$ oc adm top node master01
NAME       CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master01   1250m        16%    10268Mi         54%
```

- Sort list using **--sort-by** option

  $

Sort list by viewing top cpu/memory usage by **pods**

- Sort list using following options:
  - **-A** | **-n namespace** (default to current namespace)
  - **--sort-by** { **cpu** | **memory** } option

```
[student@workstation ~]$ oc adm top pods -A --sort-by cpu
NAMESPACE                                          NAME
                    CPU(cores)   MEMORY(bytes)
openshift-kube-apiserver                           kube-apiserver-master01
                    285m         2319Mi
openshift-monitoring                               prometheus-k8s-0
                    102m         1036Mi
openshift-etcd                                     etcd-master01
                    94m          1388Mi
openshift-authentication-operator                  authentication-operator-65f7
                    24m          96Mi
openshift-kube-controller-manager                  kube-controller-manager-mast
```

Sort list by viewing top cpu/memory usage by **nodes**

Use **--sort-by** { **cpu** | **memory** } option

```
[student@workstation ~]$ oc adm top nodes --sort-by memory
NAME       CPU(cores)   CPU%   MEMORY(bytes)   MEMORY%
master01   1017m        13%    12839Mi         67%
```

Guided Exercise: Reserve Compute Capacity for Applications

You should be able to:

- Observe that memory resource requests allocate cluster node memory.
- Explore how adjusting resource requests impacts the number of replicas that can be scheduled on a node

Lesson 4 : Limit Compute Capacity for Applications

Configure an application with resource limits so Kubernetes can protect other applications from it.

Defining Resource Requests and Limits for Pods

- A pod / deployment / DC definition can include both resource requests and resource limits:
- Resource requests
  - Used for scheduling
  - Indicate pod cannot run with less than specified amount of compute resources
- **Resource limits**
  - Used to prevent a pod from using up all cpu and memory from a node
  - Uses Linux Kernel : cgroups feature to enforce the pod's resource limits
- Quotas to track and limit per project level :
  - Object counts: pods, services, quotas, secrets, pvc and replicationcontrollers
  - Compute resources: cpu, memory
- If quota is used, then deployment should define resource request.
- Recommended: define requests and limits even if quotas are not used

Setting Memory Limits

- Specifies the amount of memory that a container can use across all its processes.
- Once limit is reached, compute node selects and kills process in the container.
- If health probes is configured, corresponding action is taken
  - example: Probes failed at threshold, cluster restart container according to pod restartPolicy (default:always)
- Uses Linux kernel features to implement kill switch:
  - Control groups (cgroups) **:** control and monitor system resource
  - Out-of-Memory killer (OOM Killer): select and kill processes
- Common case use of setting memory limit:
  - Application tends to have memory leakage pattern - bugs
  - Application kept uses more and more memory over time - Infinite service loop
- Limits enables OpenShift to regularly restart applications to free up their memory

RHOCP restarts pod bcos OOM event

```
[user@host ~]$ oc get pod hello-67645f4865-vvr42 -o yaml
...output omitted...
status:
...output omitted...
  containerStatuses:
  - containerID: cri-o://806b...9fe7
    image: registry.access.redhat.com/ubi9/nginx-120:1-86
    imageID: registry.access.redhat.com/ubi9/nginx-120:1-86@sha256:1403...fd3
    lastState:
      terminated:
        containerID: cri-o://bbc4...9eb2
        exitCode: 137
        finishedAt: "2023-03-08T07:56:06Z"
        reason: OOMKilled
        startedAt: "2023-03-08T07:51:43Z"
    name: hello
    ready: true
    restartCount: 1
...output omitted...
```

RHOCP updates pod's **lastState** attribute and set reason to **OOMKilled**

Setting CPU Limits

- Specifies maximum amount of cpu in milicores or cores that a container can consume
- IF node's CPU is under pressure; node shares CPU resource between containers according to CPU requests value
- CPU requests and limits do not account for hardware differences in nodes
  - Even equal amount of cores in separate nodes doesn't means same performance
  - Example: different make (Intel/AMD), different clock speeds (3 GHz / 6 GHz)
- Example of setting cpu limits
  - Current setup: Intel Xeon 8 cores @ 3 GHz
  - 1 core = 1000 milicores
  - Say appA requires 300 Mhz = 100 milicores

Setting CPU/Memory Limits using **oc set resources** cmd

- Set the limit

```
[student@workstation ~]$ oc set resources dc/postgresql --limits cpu=100m,memory=256Mi
deploymentconfig.apps.openshift.io/postgresql resource requirements updated
```

- Verify the definition

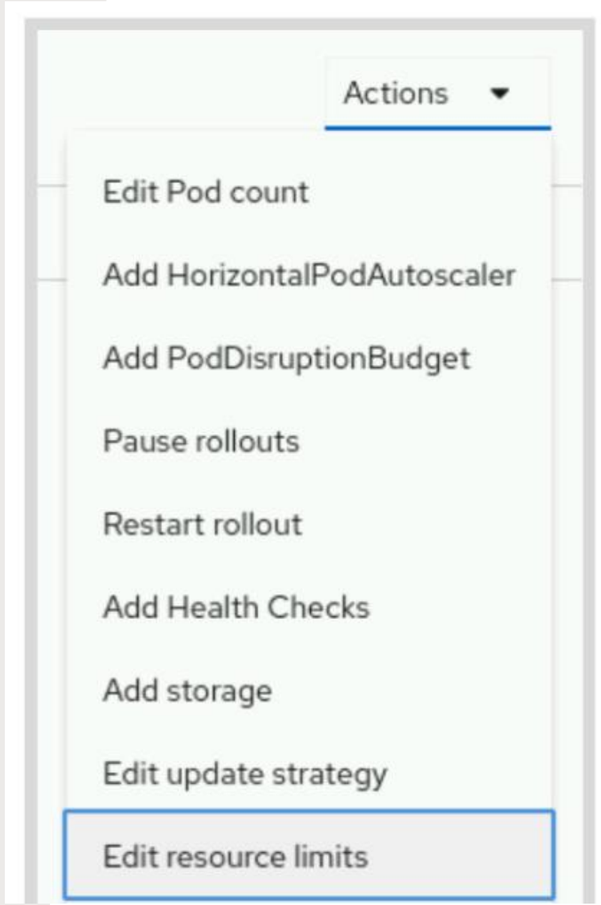
```
[student@workstation ~]$ oc describe dc/postgresql | grep -i limit -A2
    Limits:
      cpu:      100m
      memory:   256Mi
```

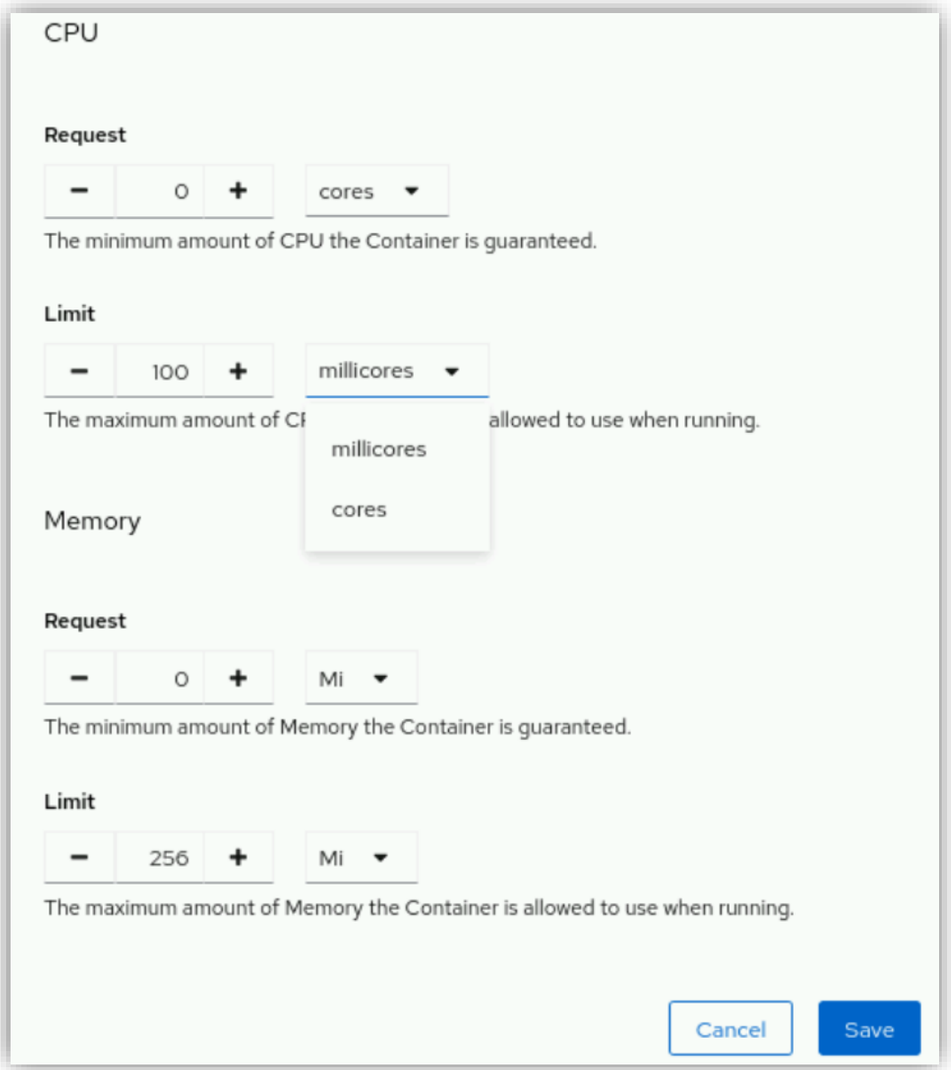
Setting CPU/Memory Limits using YAML manifest file

```
apiVersion: apps/v1
kind: Deployment
...output omitted...
  spec:
    containers:
    - image: registry.access.redhat.com/ubi9/nginx-120:1-86
      name: hello
      resources:
        requests:
          cpu: 100m
          memory: 500Mi
        limits:
          cpu: 200m
          memory: 1Gi
```

Setting CPU/Memory Limits using Web Console

- **Workloads → Deployment → Actions > Edit resource limits**





Viewing Requests, Limits, and Actual Usage

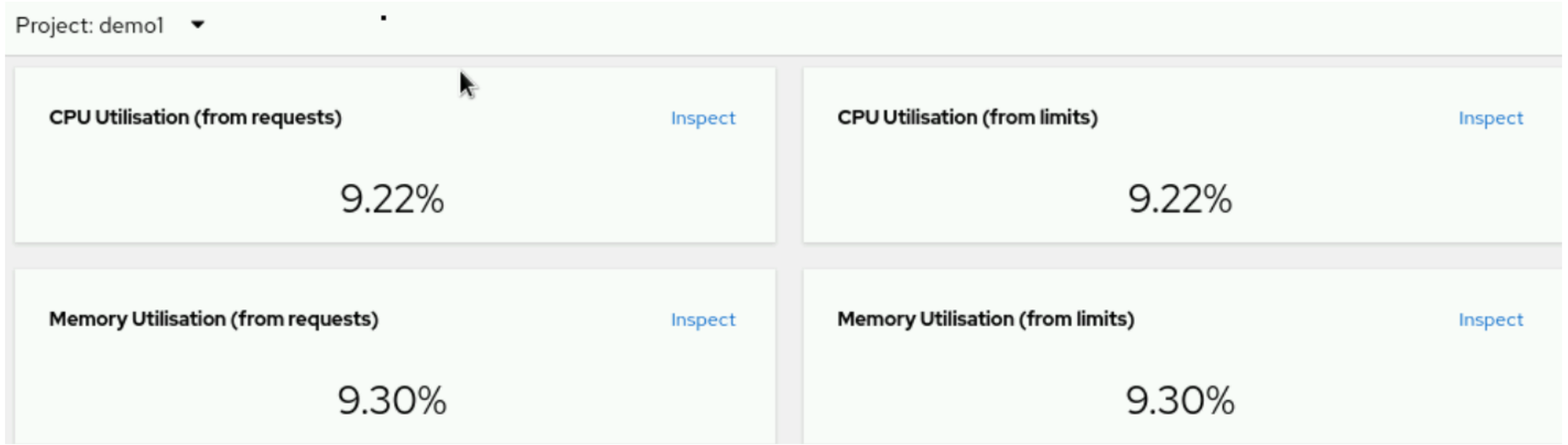
```
[user@host ~]$ oc describe node master01
Name:               master01
Roles:              control-plane,master,worker
...output omitted...
Non-terminated Pods:                                  (88 in total)
  ... Name           CPU Requests  CPU Limits  Memory Requests  Memory Limits ...
  ... ----           ------------  ----------  ---------------  -------------
  ... controller-... 10m (0%)      0 (0%)      20Mi (0%)        0 (0%)        ...
  ... metallb-...    50m (0%)      0 (0%)      20Mi (0%)        0 (0%)        ...
  ... metallb-...    0 (0%)        0 (0%)      0 (0%)           0 (0%)        ...
...output omitted...
Allocated resources:
  (Total limits may be over 100 percent, i.e., overcommitted.)
  Resource           Requests       Limits
  --------           --------       ------
  cpu                3183m (42%)    1202m (16%)
  memory             12717Mi (67%)  1350Mi (7%)
...output omitted...
```

Viewing Requests, Limits, and Actual Usage

The following command displays the resource usage for the pods in the current project:

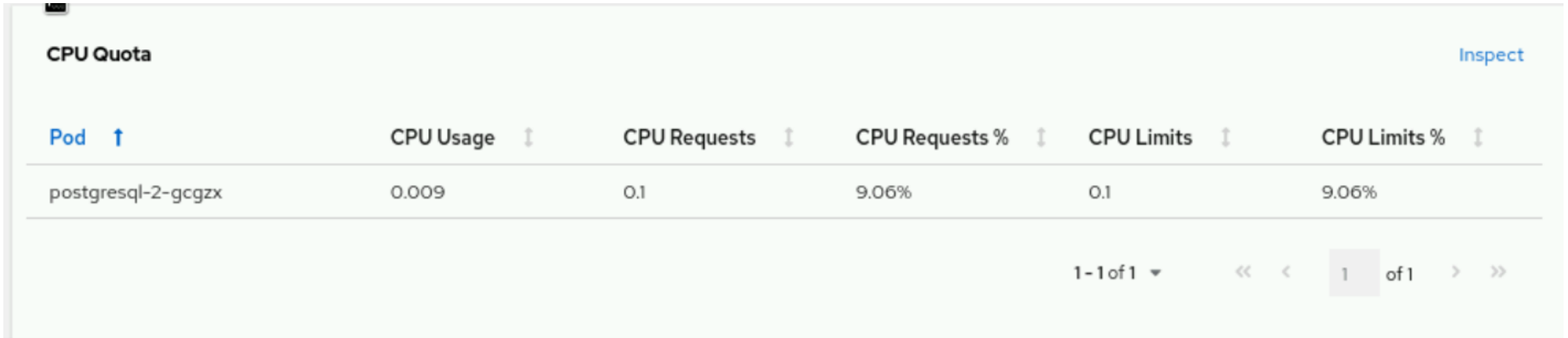
```
[user@host ~]$ oc adm top pods -n openshift-console
NAME                          CPU(cores)   MEMORY(bytes)
hello-67645f4865-vvr42        121m         309Mi
intradb-6f8d7cfffb-fz55b      0m           440Mi
```

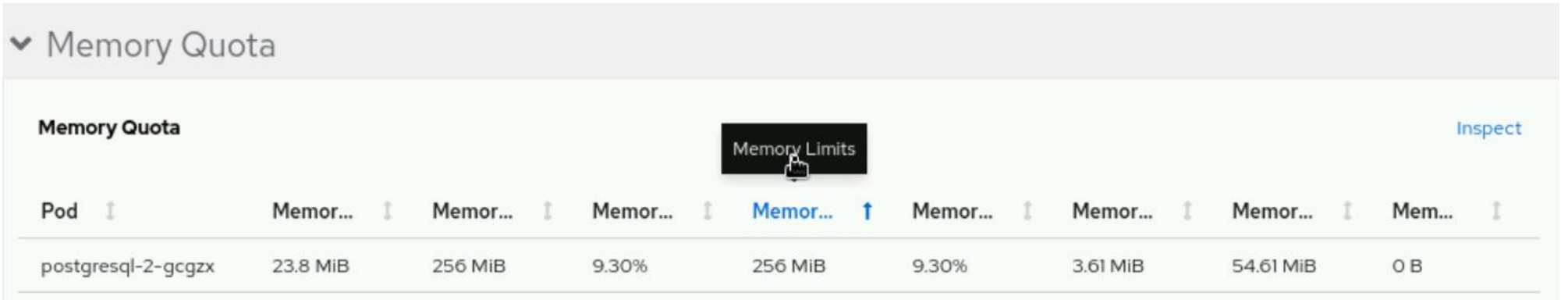
View Requests, Limits by Project using web console



View Requests, Limits by Pods using web console

- Viewing Requests, Limits, and Actual Usage





Guided Exercise: Limit Compute Capacity for Applications

You should be able to:

- You should be able to monitor the memory usage of an application, and set a memory limit for a pod.