Limit Compute Capacity for Applications

Objectives

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

Memory and CPU requests that you define for containers help Red Hat OpenShift Container Platform (RHOCP) to select a compute node to run your pods. However, these resource requests do not restrict the memory and CPU that the containers can use. For example, setting a memory request at 1 GiB does not prevent the container from consuming more memory.

Red Hat recommends that you set the memory and CPU requests to the peak usage of your application. In contrast, by setting lower values, you overcommit the node resources. If all the applications that are running on the node start to use resources above the values that they request, then the compute nodes might run out of memory and CPU.

In addition to requests, you can set memory and CPU *limits* to prevent your applications from consuming too many resources.

Setting Memory Limits

A memory limit specifies the amount of memory that a container can use across all its processes.

As soon as the container reaches the limit, the compute node selects and then kills a process in the container. When that event occurs, RHOCP detects that the application is not working any more, because the main container process is missing, or because the health probes report an error. RHOCP then restarts the container according to the pod `restartPolicy` attribute, which defaults to `Always`.

RHOCP relies on Linux kernel features to implement resource limits, and to kill processes in containers that reach their memory limits:

Control groups (cgroups)
RHOCP uses control groups to implement resource limits. Control groups are a Linux kernel mechanism for controlling and monitoring system resources, such as CPU and memory.

Out-of-Memory killer (OOM killer)
When a container reaches its memory limit, the Linux kernel triggers the OOM killer subsystem to select and then kill a process.

You must set a memory limit when the application has a memory usage pattern that you must mitigate, such as when the application has a memory leak. A memory leak is a bug in the application, which occurs when the application uses some memory but does not free it after use. If the leak appears in an infinite service loop, then the application uses more and more memory over time, and can end up consuming all the available memory on the system. For

these applications, setting a memory limit prevents them from consuming all the node's memory. The memory limit also enables OpenShift to regularly restart applications to free up their memory when they reach the limit.

To set a memory limit for the container in a pod, use the `oc set resources` command:

```
[user@host ~]$ oc set resources deployment/hello --limits memory=1Gi
```

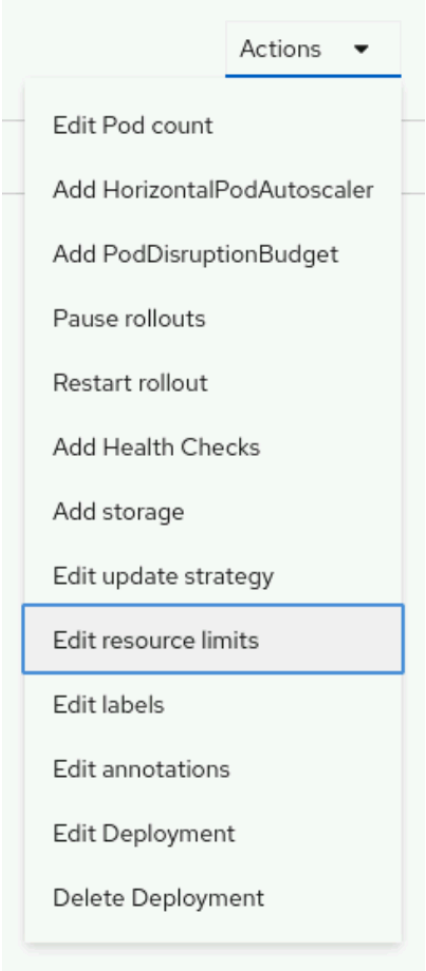
In addition to the `oc set resources` command, you can define resource limits from a file in the YAML format:

```
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
```

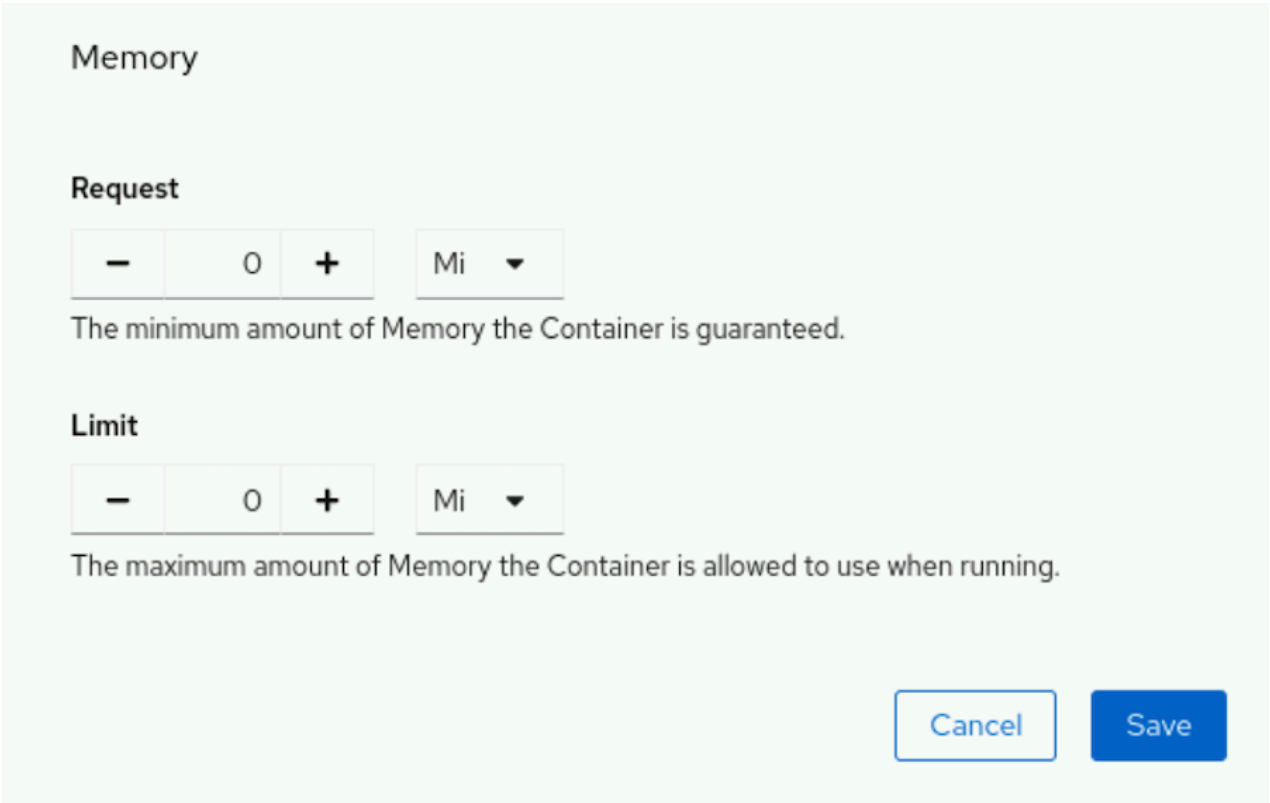
When RHOCP restarts a pod because of an OOM event, it updates the pod's `lastState` attribute, and sets the reason to `OOMKilled`:

```
[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...fd34
    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...
```

To set a memory limit for the container in a pod from the web console, select a deployment, and click **Actions → Edit resource limits**.



Set memory limits by increasing or decreasing the memory on the **Limit** section.



Setting CPU Limits

CPU limits work differently from memory limits. When the container reaches the CPU limit, RHOCP inhibits the container's processes, even if the node has available CPU cycles. The application continues to work, but at a slower pace.

In contrast, if you do not set a CPU limit, then the container can consume as much CPU as is available on the node. If the node's CPU is under pressure, for example because several containers are running CPU-intensive tasks, then the Linux kernel shares the CPU resource between all these containers, according to the CPU requests value for the containers.

You must set a CPU limit when you require a consistent application behavior across clusters and nodes. For example, if the application runs on a node where the CPU is available, then the application can execute at full speed. On the other hand, if the application runs on a node with CPU pressure, then the application executes at a slower pace.

The same behavior can occur between your development and production clusters. Because the two environments might have different node configurations, the application might run differently when you move it from development to production.

> **NOTE**
>
> Clusters can have differences in hardware configuration beyond what limits observe. For example, two clusters' nodes might have CPUs with equal core count and unequal clock speeds.
>
> Requests and limits do not account for these hardware differences. If your clusters differ in such a way, take care that requests and limits are appropriate for both configurations.

By setting a CPU limit, you mitigate the differences between the configuration of the nodes, and you experience a more consistent behavior.

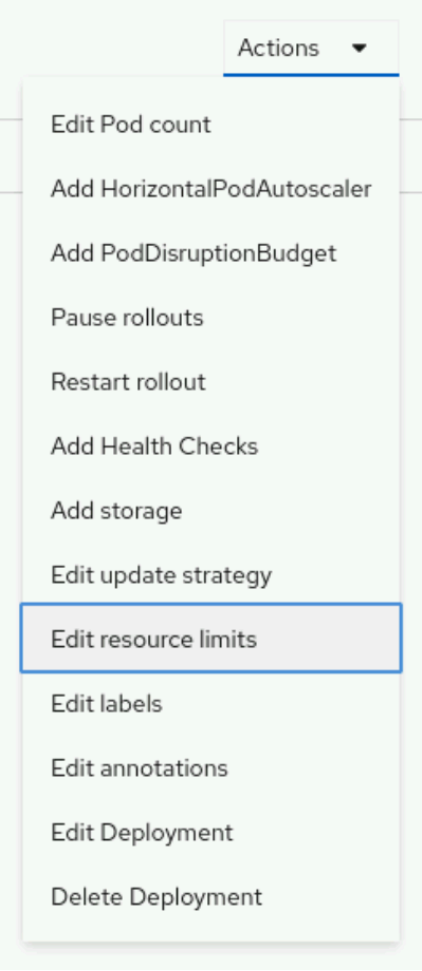
To set a CPU limit for the container in a pod, use the `oc set resources` command:

```
[user@host ~]$ oc set resources deployment/hello --limits cpu=200m
```

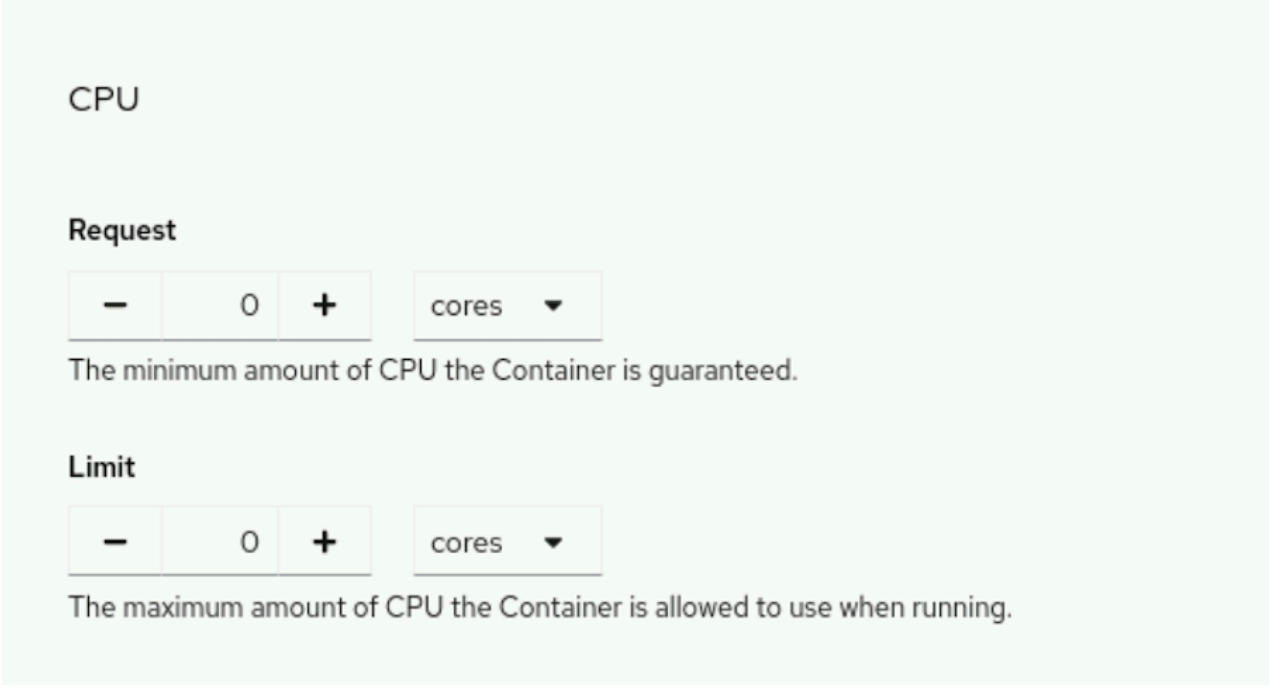
You can also define CPU limits from a file in the YAML format:

```
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
```

To set a CPU limit for the container in a pod from the web console, select a deployment, and click **Actions → Edit resource limits**.



Set CPU limits by increasing or decreasing the CPU on the **Limit** section.



Viewing Requests, Limits, and Actual Usage

By using the RHOCP command-line interface, cluster administrators can view compute usage information on individual nodes. The `oc describe node` command displays detailed information about a node, including information about the pods that are running on the node. For each pod, it shows CPU requests and limits, as well as memory requests and limits. If you do not specify a request or limit, then the pod shows a zero for that column. The command also displays a summary of all the resource requests and limits.

```
[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...
```

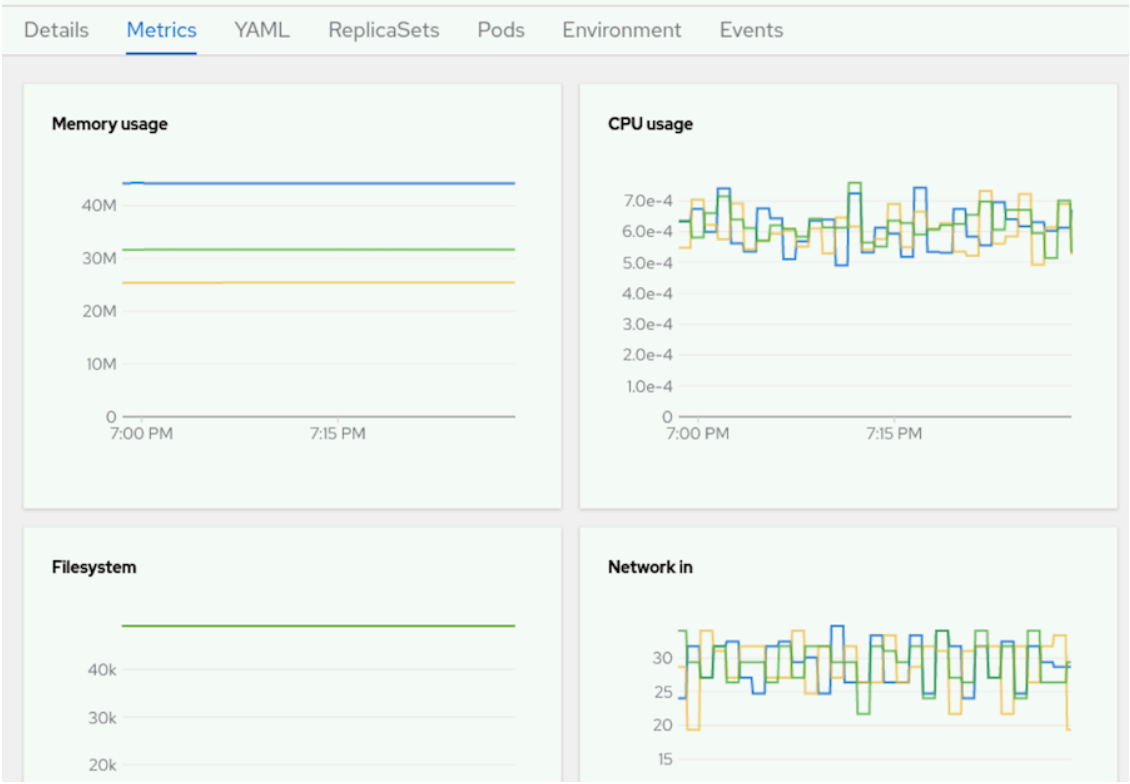
The `oc describe node` command displays requests and limits. The `oc adm top` command shows resource usage. The `oc adm top nodes` command shows the resource usage for nodes in the cluster. You must run these commands as a cluster administrator.

The `oc adm top pods` command shows the resource usage for each pod in a project.

The following command displays the resource usage for the pods in the `openshift-console` project:

```
[user@host ~]$ oc adm top pods -n openshift-console
NAME                          CPU(cores)   MEMORY(bytes)
console-6689c8589c-bcpb7      1m           48Mi
downloads-79658645bd-hx649    1m           31Mi
```

To visualize the consumption of resources from the web console, select a deployment, and click the **Metrics** tab. From this tab, you can view the usage for memory, CPU, the file system, and incoming and outgoing traffic.



REFERENCES

`cgroups`(7) man page

For more information about resource limits, refer to the *Configuring Cluster Memory to Meet Container Memory and Risk Requirements* section in the *Working with Clusters* chapter in the Red Hat OpenShift Container Platform 4.18 *Nodes* documentation at https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html-single/nodes/index#nodes-cluster-resource-configure