# How To Determine What Limits Exists for Containers Running in Clusters

**Downloaded from Epic Games Confluence**
Date: 2025-07-12 04:07:13
Original URL: https://confluence-epicgames.atlassian.net/wiki/spaces/CDE/pages/81068311

Document Level Classification

200

# Introduction

When trying to locate what resources.limits exists for containers running in your cluster and how close is runtime to those limits. There are a number of tools we recommend using to get that data.

- Newrelic NRQL
- kubectl-view-allocations
- kube-capacity

Each of the suggested  command line tools requires that you run **aop auth account-env** and **aop kube cluster-name**. To use NewRelic's NRQL you need to login to NewRelic via [Okta apps page](#).

# Using kubectl

```
alias nodeusage='kubectl get nodes --no-headers | awk '\''{print $1}'\'

# Results

ip-10-105-103-111.ec2.internal
  Resource                           Requests          Limits
  cpu                                1935m (12%)       2 (12%)
  memory                             1869255936 (6%)   4226176768 (14%)

ip-10-105-132-203.ec2.internal
  Resource                           Requests          Limits
  cpu                                1835m (11%)       2 (12%)
  memory                             1839255936 (6%)   4146176768 (14%)

ip-10-105-76-26.ec2.internal
  Resource                           Requests          Limits
  cpu                                1735m (10%)       2 (12%)
  memory                             1765855616 (5%)   3967918848 (13%)
```

# kubectl-view-allocations

This kubectl plugin lists allocations for resources (cpu, memory, gpu,...) defined in for the workloads. It allows you to see what workloads requested **requests** and **limits**.

Example usage

```
# all resource groups
kubectl-view-allocations -g resources -n epic-system

 Resource                     Requested        Limit  Allocatable    Fr
  attachable-volumes-aws-ebs         __           __         75.0
  cpu                         (12%) 5.7    (20%) 9.3         47.7    38
  ephemeral-storage                  __           __       721.5G
  memory                      (7%) 6.3G  (16%) 13.9G       82.9Gi  70.0
  pods                        (3%) 20.0    (3%) 20.0        702.0   682

# pod resources
kubectl-view-allocations -g pod -n epic-system

# This example shows that this workload didn't apply resources.limits t
kubectl-view-allocations -g pod -n team-online-infra-platform
 Resource                          Requested     Limit  Allocatable
  attachable-volumes-aws-ebs              __        __         75.0
  cpu                              (13%) 6.0        __         47.7
  ├─ teleport-6f9d8c54d6-8xfwt           2.0        __           __
  ├─ teleport-6f9d8c54d6-tfsz5           2.0        __           __
  └─ teleport-6f9d8c54d6-tq7d5           2.0        __           __
  ephemeral-storage                       __        __       721.5G
  memory                        (29%) 24.0Gi        __       82.9Gi  58
  ├─ teleport-6f9d8c54d6-8xfwt         8.0Gi        __           __
  ├─ teleport-6f9d8c54d6-tfsz5         8.0Gi        __           __
  └─ teleport-6f9d8c54d6-tq7d5         8.0Gi        __           __
  pods                              (0%) 3.0  (0%) 3.0        702.0   6
```

# kube-capacity

provides an overview of the resource requests, limits, and utilization in a Kubernetes cluster.

Example usage

```
kube-capacity --pods -n team-online-infra-platform
NODE                             POD                          CPU REQUES
*                                *                            6000m (12%

ip-10-106-112-66.ec2.internal    *                            2000m (12%
ip-10-106-112-66.ec2.internal    teleport-6f9d8c54d6-tfsz5    2000m (12%

ip-10-106-130-175.ec2.internal   *                            2000m (12%
ip-10-106-130-175.ec2.internal   teleport-6f9d8c54d6-8xfwt    2000m (12%

ip-10-106-66-205.ec2.internal    *                            2000m (12%
ip-10-106-66-205.ec2.internal    teleport-6f9d8c54d6-tq7d5    2000m (12%
```

# NRQL

Exposes your Kubernetes data providing this information also. You can add the results of the query to your NewRelic Dashboard.

Example output: https://onenr.io/0a7j9Eme3jO

```
SELECT average(k8s.container.memoryLimitBytes) as MemoryLimit ,
average(k8s.container.memoryRequestedBytes) as MemoryRequests,
average(k8s.container.cpuRequestedCores) as CPURequested ,
average(k8s.container.cpuLimitCores) as CPULimits
FROM Metric
FACET k8s.podName
WHERE k8s.clusterName = 'dead-dev-teleport'
SINCE 1 hour AGO
```

Example output: https://onenr.io/09MR27KDbjY

```
FROM K8sNodeSample
SELECT (average(cpuRequestedCores) * uniqueCount(nodeName)) AS 'CPU req
```

```
average(cpuUsedCores) as 'CPU used',
(average(allocatableCpuCores) * uniqueCount(nodeName)) AS 'CPU total',
(average(memoryRequestedBytes) * uniqueCount(nodeName)) AS 'Memory requ
average(memoryUsedBytes) as 'Memory used',
(average(allocatableMemoryBytes) * uniqueCount(nodeName)) AS 'Memory to
FACET clusterName
LIMIT MAX
```

**Page Information:**
Page ID: 81068311
Space: Cloud Developer Platform
Downloaded: 2025-07-12 04:07:13