

Machine Learning

## Using quota management (Tech Preview)

Date published: 2020-07-16

Date modified: 2024-05-30

# CLOUDERA

<https://docs.cloudera.com/>

# Legal Notice

© Cloudera Inc. 2024. All rights reserved.

The documentation is and contains Cloudera proprietary information protected by copyright and other intellectual property rights. No license under copyright or any other intellectual property right is granted herein.

Unless otherwise noted, scripts and sample code are licensed under the Apache License, Version 2.0.

Copyright information for Cloudera software may be found within the documentation accompanying each component in a particular release.

Cloudera software includes software from various open source or other third party projects, and may be released under the Apache Software License 2.0 (“ASLv2”), the Affero General Public License version 3 (AGPLv3), or other license terms. Other software included may be released under the terms of alternative open source licenses. Please review the license and notice files accompanying the software for additional licensing information.

Please visit the Cloudera software product page for more information on Cloudera software. For more information on Cloudera support services, please visit either the Support or Sales page. Feel free to contact us directly to discuss your specific needs.

Cloudera reserves the right to change any products at any time, and without notice. Cloudera assumes no responsibility nor liability arising from the use of products, except as expressly agreed to in writing by Cloudera.

Cloudera, Cloudera Altus, HUE, Impala, Cloudera Impala, and other Cloudera marks are registered or unregistered trademarks in the United States and other countries. All other trademarks are the property of their respective owners.

Disclaimer: EXCEPT AS EXPRESSLY PROVIDED IN A WRITTEN AGREEMENT WITH CLOUDERA, CLOUDERA DOES NOT MAKE NOR GIVE ANY REPRESENTATION, WARRANTY, NOR COVENANT OF ANY KIND, WHETHER EXPRESS OR IMPLIED, IN CONNECTION WITH CLOUDERA TECHNOLOGY OR RELATED SUPPORT PROVIDED IN CONNECTION THEREWITH. CLOUDERA DOES NOT WARRANT THAT CLOUDERA PRODUCTS NOR SOFTWARE WILL OPERATE UNINTERRUPTED NOR THAT IT WILL BE FREE FROM DEFECTS NOR ERRORS, THAT IT WILL PROTECT YOUR DATA FROM LOSS, CORRUPTION NOR UNAVAILABILITY, NOR THAT IT WILL MEET ALL OF CUSTOMER’S BUSINESS REQUIREMENTS. WITHOUT LIMITING THE FOREGOING, AND TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, CLOUDERA EXPRESSLY DISCLAIMS ANY AND ALL IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, QUALITY, NON-INFRINGEMENT, TITLE, AND FITNESS FOR A PARTICULAR PURPOSE AND ANY REPRESENTATION, WARRANTY, OR COVENANT BASED ON COURSE OF DEALING OR USAGE IN TRADE.

# Contents

<b>Quota Management overview.....</b>	<b>4</b>
<b>Creating a resource pool for CML.....</b>	<b>4</b>
<b>Enabling Quota Management in CML.....</b>	<b>7</b>
<b>Provisioning a workspace using a configured resource pool.....</b>	<b>8</b>
<b>Quota for CML workloads.....</b>	<b>9</b>
<b>Resource Usage Dashboard.....</b>	<b>10</b>
<b>Limitations for Quota Management.....</b>	<b>11</b>
<b>Yunikorn Gang scheduling.....</b>	<b>11</b>

## Quota Management overview

Quota Management enables you to control how resources are allocated within your CML workspace on user and on group level.

In order to prevent a single session, job, or other workload from consuming all the available cluster resources, you can limit the number of CPUs, the amount of memory allocated by user, business units, or Data Service by defining resource pools that set resource limits.

Pools are organized in a hierarchical manner, defining nodes in a hierarchy based on resource limits.

A regular workflow of managing quotas constitutes of the following activities:

1. Creating a resource pool for CML
2. Enabling Quota management in CML
3. Provisioning a workplace using a configured resource pool



**Note:**

It is recommended for CML administrators to create at least one resource pool under root.default for CML during the Technical Preview (TP) release. Do not use the root.default resource pool for creating a workspace.

Quota Management cannot be enabled for an existing workspace. It is recommended for CML administrators to provision a new workspace to enable and test this feature.



**Note:** This feature is in Technical Preview and not recommended for production deployments. Cloudera recommends that you try this feature in test or development environments.

### Related Information

[Creating a resource pool for CML](#)

[Enabling Quota Management in CML](#)

[Provisioning a workspace using a configured resource pool](#)

[Limitations for Quota Management](#)

## Creating a resource pool for CML

In the Technical Preview (TP) release you must create at least one resource pool for Quota Management. Do not use root.default resource pool for creating a workspace.

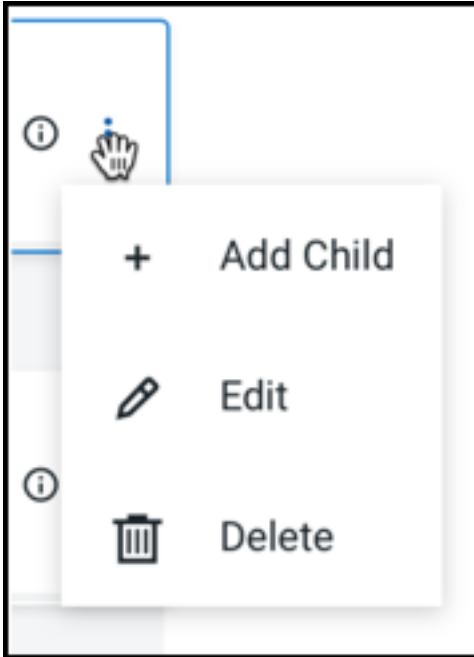
### Procedure

1. Go to Cloudera Manager Resource Utilization tab .
2. Under the root.default resource pool (or any already created custom resource pool), create a child node for CML.

The pools are organized in a hierarchical manner.

CML reserves 30 GB Memory and 20 CPU for the CML workspace installation. The resource pool created for CML must have at least 38 GB of Memory and 22 CPU.

3. Click the actions menu in a pool and select Add Child.




The Add Quota dialog box opens.

### Add Quota ✕


\* Name ?


✓ Total Quota

 Memory

▼

 CPU

▼

 GPU

▼

> Guaranteed Quota

Validity ?

Distribution Policy ?

Inelastic  Elastic

Tags ?

Key	Value	✕
<input type="text"/>	<input type="text"/>	<input type="button" value="✕"/>

#### 4. Enter the following information:

- Name – Enter a name for this resource pool.
- Memory – Use the sliders to select the memory allocation for the pool. Use the drop-down list to select the units.



**Note:** It is recommended to configure memory in GB units.

- CPU – Use the sliders to select the CPU allocation for the pool. You can choose how CPUs are counted by selecting cores or millicores from the drop-down list.



**Note:** It is recommended to configure CPU units in cores.

- GPU – Use the sliders to select the GPU allocation for the pool.



**Note:** It is recommended to configure GPU units in cores.

- Validity – Enter the length of time until the pool can remain active. To keep the pool active indefinitely, enter NEVER\_EXPIRES.



**Note:** The default value for validity is NEVER\_EXPIRES.

- Tags – Tags provide a way to add user-defined name or value pairs as metadata for the pools. Use tag key: experience, value: cml,, which helps CML to identify that the resource pool is configured for CML.



**Note:**

Note that the maximum value of the sliders is limited by the available resources, that is, the total resources of the parent pool and the amount of resources not used by other sibling pools.

## Enabling Quota Management in CML

To enable Quota Management in Cloudera Machine Learning (CML) it needs to be configured. Follow the recommended configuration guidelines.

### Before you begin

Set up the Kubernetes and kubectl as defined in [Prerequisites](#).

### Procedure

1. Configure the kubectl using the kubeconfig file.
2. Edit the CML control plane deployment:

```
kubectl get deploy dp-mlx-control-plane-app -n cdp-namespace -o yaml > file-name
```

This will save the CML control plane deployment specification.

3. Edit the Cadence Worker deployment with the following command:

```
kubectl get deploy dp-cadence-worker -n <cdp-namespace> -o yaml > <cadence-file-name>
```

The command saves Cadence Worker deployment specification.

4. Take a backup of the above files.

5. Search for the environment variable `ENABLE_UMBRA_INTEGRATION` in both files, and change the value to `true`.
6. Save the deployment files and run the following commands:

```
kubectl apply -f file-name.yaml
```

```
kubectl apply -f <cadence-file-name>.yaml
```

Wait for the new pods to come up.

7. Verify the new pods by running the following command:

```
kubectl get pods -n cdp-namespace
```

## Provisioning a workspace using a configured resource pool

Configuring the quota for a specified workspace requires additional configuration settings besides provisioning the workspace for Quota Management.

Cloudera Machine Learning (CML) reserves 30 GB Memory and 20 CPU for the CML workspace installation.

The minimum requirement for a CML workspace to be provisioned is 38 GB of Memory and 22 CPU.

### Umbra Integration Options

\* Resource Pool

**root.default.cml**

\* CPU (Cores)

A maximum of 100 cores can be set.

\* Memory (GB)

A maximum of 50 GB can be set.

GPU (Cores)

A maximum of 30 cores can be set.

In this example out of 50 Cores and 100 GB allocated for the resource pool `root.default.cml`, 22 Cores and 38 GB are reserved for the CML workspace. Out of these resources, 20 cores and 30GBs will be used for CML, while 2 cores and 8 GBs are available to run workloads.



The rest of the resources is available in root.default.cml resource pool and can be allocated to other ML Workspaces or other Data Services.

## Quota for CML workloads

Quota management is implemented for both user and group level. A Machine Learning (ML) Workspace is allocated a set amount of resources based on configured parameters at provisioning time. Within a workspace, resources available for workloads can be further subdivided into quotas at user and/or team level.

### User and team naming limitations

Follow the kubernetes label naming conventions when naming users and teams:

- The name must be 63 characters or less.
- The name must begin and end with an alphanumeric character ([a-z0-9A-Z]).
- The name can contain dashes (-), underscores (\_), dots (.), and alphanumerics, except for the beginning and ending characters.

### User quota

By default, 8 GB memory and 2 vCPU cores are configured for each user. Such resources are sufficient for running simple sessions but might not be sufficient for the spark workloads if the executors cannot find enough resources. The Cloudera Machine Learning (CML) administrators can configure custom quota for the user on the Site Administration Page.

**Figure 1: User quota settings**

The screenshot shows the 'Site Administration' interface with the 'Quotas' tab selected. It displays 'Default User Quota' and a table of 'Users Quota' for the 'admin' user.

Default User Quota			
Quotas	CPU (vCPU)	Memory (GiB)	GPU
Default (per user)	2	8	0

Users	Quota			Usage		
	CPU (vCPU)	Memory (GiB)	GPU	CPU (vCPU)	Memory (GiB)	GPU
admin	2	8	0	0/2	0/8	0/0

If the quota for a user is used up, the workload remains in the pending state until the required resources are available.

If the quota for the users is modified, it will be reflected when the next workload is submitted.

GPU resources can be edited if the workspace is provisioned with GPU resources.

### Group quota

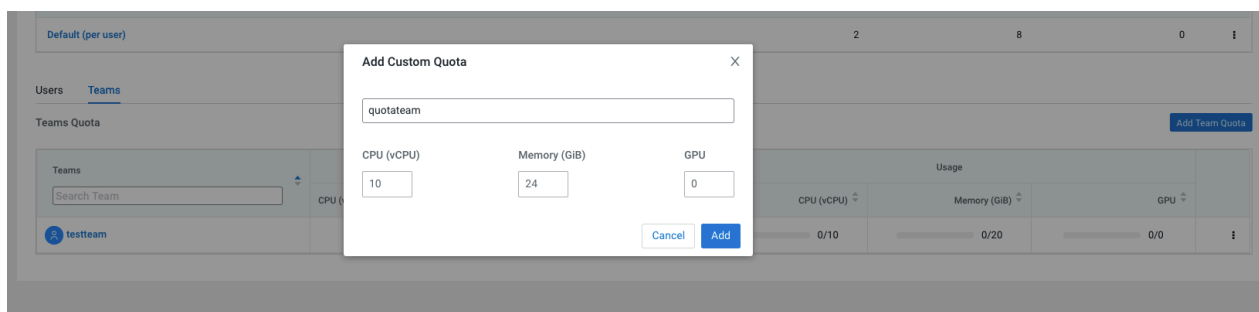
There is no quota configured by default for a team or group.

1. As a CML administrator configure a custom quota for a group under Site Administration Quotas tab.
2. Configure a quota for a group, at the Quotas Teams tab.

3. Click the Add Team Quota and a popup enables you to add a custom team quota.

**Figure 2:**

**Figure 3: Custom quota**



If the quota for the group is used up, the workload remains in PENDING state until the required resources are available.

If the quota for the group is modified, it will be reflected when the next workload is submitted.

### Quota enforcement

The resources are allocated based on the project context from which the workloads are created.

- If a workload is created in a project, which is developed in users' context, it will always take the user quota.
- If a workload is created in a project which is developed in a team's context, it will take resources from the team's quota, provided the team has a quota configured.
- If a workload is created in a project which is developed in a team's context and the team does not have a quota configured, it will take the resources from the users' quota.

### Related Information

[Creating a team](#)

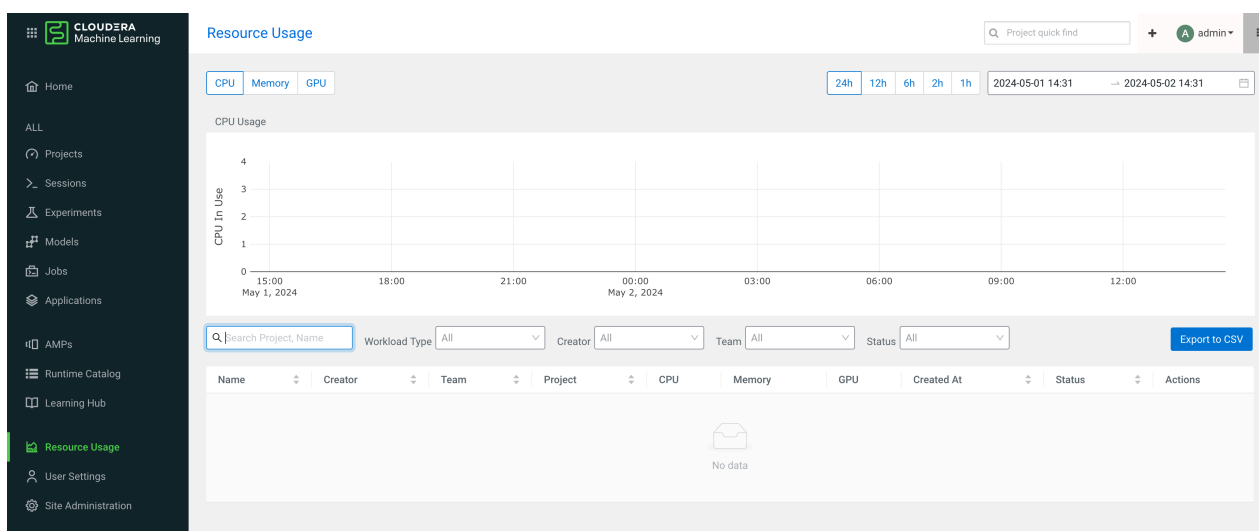
## Resource Usage Dashboard

The Resource Usage Dashboard is developed on top of Quota Management to depict the resource usage metrics. The dashboard can only be accessed when Quota Management is enabled.

The Resource Usage Dashboard consists of two parts.

- The Time series chart shows each resource's usage, based on the filtered time range.
- The Workloads' usage table shows how much resources are utilized by each workload, based on the filtered time range.

**Figure 4: Resource Usage Dashboard**



The resource options on the top left side (CPU, Memory and GPU) are used to show or filter the selected resource usage, based on a filtered time range.

The date and time range options on the top right side are global filters which are used to filter both the time-series chart and the workloads' usage table.

There is also an export button to download the Workloads' usage table into CSV format.

## Limitations for Quota Management

Follow the recommendations on the limitations of Cloudera Machine Learning (CML) Quota Management.



**Note:** Do not use the root.default resource pool in the Technical Preview (TP) release to provision the CML workspace. Create at least one resource pool for CML.

If the resource pool gets deleted, the underlying CML workspace and workloads will also be deleted. However, the stale entries will still be available in the CML UI, reflecting that the workspace still exists but the underlying applications on the cluster are deleted.

If the quota for users or groups is modified, it will be reflected when the next workload (session or job) is submitted.

## Yunikorn Gang scheduling

Yunikorn Gang Scheduling is the default scheduling mechanism in Cloudera Machine Learning. Yunikorn schedules the workload pods when Quota Management is enabled.

Gang scheduling is a scheduling mechanism in Yunikorn where Yunikorn schedules an application (a set of pods) only when all the resources necessary for bringing up all the pods in the application are available in the cluster. If the resources for bringing up all the pods are not available, the application will not be scheduled, thus helping to use cluster resources efficiently.

Most workloads need only a single pod and work without any additional configuration in the CML. Those workloads however, that take additional worker pods, need additional configuration of Gang parameters to work optimally.

For example, in case there is a Spark job which creates additional executor pods, the driver pod is created in the default setting with the Gang Scheduler and the executor pods use the regular scheduling (that is, the pods will not be scheduled with a Gang Scheduler). However, this is not optimal to run the workloads so. It is important to specify the expected number of worker pods that the Spark driver pod will schedule. Complete the configuration with the following environment variable:

ENV\_GANG\_MIN\_MEMBER - It sets the number of worker pods apart from the driver pod.

**Figure 5: Environment variable**

The screenshot shows a configuration panel for a runtime image. It includes sections for 'Runtime Image', 'Schedule', 'Resource Profile', 'Timeout In Minutes', and 'Environment Variables'. The 'Environment Variables' section contains a table with one row for 'ENV\_GANG\_MIN\_MEMBER' with a value of '2'. Below the table are input fields for adding new variables and a 'Delete' button for the existing one.

**Runtime Image**  
- docker-private.infra.cloudera.com/cloudera/cds/m-runtime-jupyterlab-python3.8-standard:2023.08.2-b8

**Schedule**  
Manual

**Resource Profile**  
2 vCPU / 4 GiB Memory

**Timeout In Minutes (optional)** 30  Kill on Timeout  
Jobs exceeding timeout send warning email if notifications enabled.

**Environment Variables**

Name	Value	Actions	Hide/Show Value
ENV_GANG_MIN_MEMBER	2	Delete	
<input type="text"/>	<input type="text"/>	Add	

Environment variables will override the [project environment](#).



**Note:**

Although the environment variable can also be configured at project level, it is recommended to configure it at workload level (jobs, models and applications).

**Model scheduling**

For model workloads, CML configures the same number of driver pods as the number of replicas configured for the model. Consequently, the models will not be displayed if the system cannot allocate the resources for all the replicas configured. It is therefore recommended to increase the resource quota or set the replicas properly.

You can override this behavior by using the following environment variable: ENV\_DISABLE\_GANG\_SCHEDULER.

If the value for this variable is set to true for the workload, the Gang Scheduling will be disabled only for that workload. (This is not recommended.)

**Placeholder pods**

Yunikorn completes the Gang Scheduling work by creating placeholder pods for all the resources needed for the workload. If you configure the ENV\_GANG\_MIN\_MEMBER variable with the value 5 and the system cannot allocate resources for all the 6 placeholder pods (5 workers + 1 driver), the driver pod will remain in PENDING state.

**Figure 6: Gang Scheduling pods**

```

Context: default
Cluster: default
User: default
K8s Rev: v0.25.10 +v0.31.3
K8s Rev: v1.26.10+rke2r1
CPU: 13%
MEM: 54%

Pod(s) [agtest-user-1] [7]
NAME                 PP  READY  RESTARTS  STATUS      CPU  MEM  NCPU/B  NCPU/L  NMEM/B  NMEM/L  IP             NODE
fq-linogee5my14o    ●  0/0     0          Pending     0    0     0       0       0       0       n/a          n/a          n/a
tg-fq-linogee5my14s-agtest-user-driver-fq-linogee5my-5tvarph1ft  ●  1/1     0          Running     0    0     0       0       0       0       10.42.1.187    ecs-13r9kz-4.vpc.c1
tg-fq-linogee5my14s-agtest-user-executor-fq-linogee5-8e43d77fas  ●  1/1     0          Running     0    0     0       0       0       0       10.42.3.191    ecs-13r9kz-2.vpc.c1
tg-fq-linogee5my14s-agtest-user-executor-fq-linogee5-9z166srchc  ●  1/1     0          Running     0    0     0       0       0       0       10.42.2.258    ecs-13r9kz-3.vpc.c1
tg-fq-linogee5my14s-agtest-user-executor-fq-linogee5-czzapccf8  ●  0/0     0          Pending     0    0     0       0       0       0       n/a          n/a          n/a
tg-fq-linogee5my14s-agtest-user-executor-fq-linogee5-1ljv4q15ay  ●  0/0     0          Pending     0    0     0       0       0       0       n/a          n/a          n/a
tg-fq-linogee5my14s-agtest-user-executor-fq-linogee5-ks8009auks  ●  0/0     0          Pending     0    0     0       0       0       0       n/a          n/a          n/a

```

By default, the scheduler keeps placeholder pods in CML for 60 seconds. If the scheduler cannot allocate resources within 60 seconds, the workload will be terminated with a FAILED status, and all the pods will be terminated.

Increase the default placeholder timeout value for a workload if needed by configuring the following environment variable:

ENV\_GANG\_PLACEHOLDER\_TIMEOUT - It sets the number of seconds until scheduling.

**Figure 7: Environment Gang placeholder**

### Schedule

Manual

### Resource Profile

2 vCPU / 4 GiB Memory

Timeout In Minutes (optional) 30

Kill on Timeout

Jobs exceeding timeout send warning email if notifications enabled.

### Environment Variables

Name	Value	Actions	Hide/Show Value
ENV_GANG_PLACEHOLDER_TIMEOUT	300	Delete	

In the example, the placeholder pods will be terminated after 300 seconds.

### Configuring resources

By default, the resources for the executors are set similarly as those of the drivers. If the executor resources are different from those of the driver, the values can be configured using the following environment variables:

- Set the executor resources for CPUs with: ENV\_GANG\_CPU\_REQUEST.
- Set the executor resources for memory with: ENV\_GANG\_MEMORY\_REQUEST.
- Set the executor resources for GPUs with: ENV\_GANG\_GPU\_REQUEST.

Only configure the resources that are different from that of the driver. If the CPU and memory capacity needs for the executor are the same as for the driver, only configure the ENV\_GANG\_GPU\_REQUEST variable.

### Related Information

[Troubleshooting guide for Gang Scheduler](#)