

GUROBI
OPTIMIZATION

Gurobi Remote Services Guide

Version 14.0

Copyright © 2026, Gurobi Optimization, LLC

Nov 10, 2026

Revision: 50a37dcd5

CONTENTS

1 Overview 3

2 Cluster Setup and Administration 17

3 Using Remote Services 45

4 Cluster Manager 65

5 Programming with Remote Services 127

6 Using Remote Services with Gurobi Instant Cloud 135

7 Appendix A: grb_rs 137

8 Appendix B: grb_rs - Configuration Properties 139

9 Appendix C: grb_rsm 145

10 Appendix D: grb_rsm - Configuration Properties 147

11 Appendix E: grbcluster 151

12 Appendix F: gurobi_cl 153

13 Appendix G: Acknowledgment of 3rd Party Icons 155

14 Release Notes 157

Version 14.0

Gurobi Remote Services is a set of Gurobi features that enables a cluster of one or more machines to perform Gurobi computations on behalf of other machines. The key components of Remote Services are:

- Compute Server, which allows you to offload all Gurobi computations from a client machine onto a remote cluster.
- Distributed Workers, which can be used to perform parallel computation on multiple machines.
- The Cluster Manager, a new (optional) application server that provides secured access to your Remote Services cluster, as well as providing a Web User Interface and a command-line tool that make it easier to manage and monitor your cluster.

This document is organized into a number of sections. The first section provides an *overview of Gurobi Compute Server and Remote Services*. The next section, meant for system administrators, provides details on *setting up Remote Services*. The next sections provide details on *using Remote Services* and *programming with Remote Services*. Finally, we discuss *using Remote Services with Gurobi Instant Cloud*.

CHAPTER

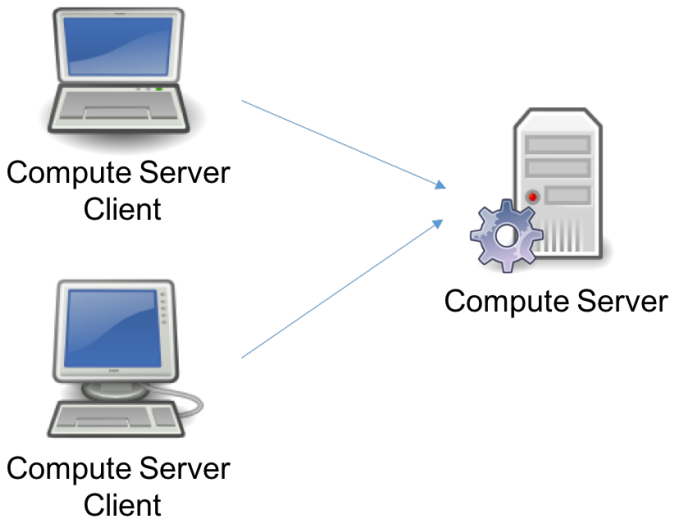
ONE

OVERVIEW

This section gives a quick introduction to the capabilities of Gurobi Remote Services. We first discuss common use cases related to *client-server optimization*. Then we describe the different components of the *architecture* and how they can be deployed together. Following this, we review the various *security* features related to user management, authentication, and encryption. Finally, we give a *simple example* of how to submit an optimization task from a client to a Compute Server cluster using the provided command-line tools.

1.1 Client-Server Optimization

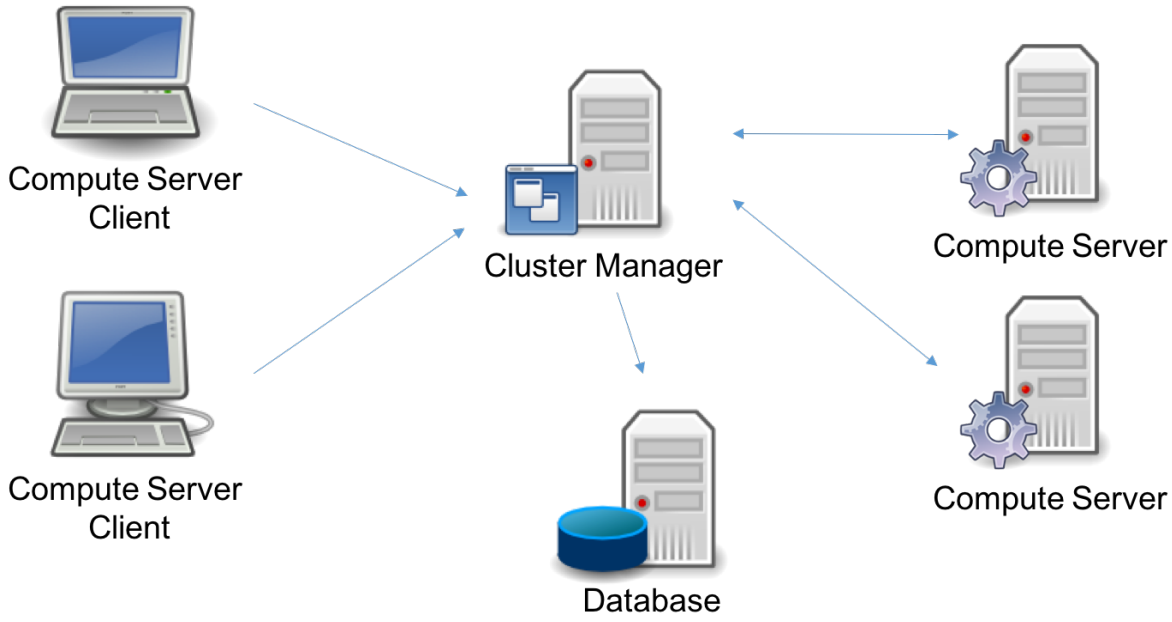
Gurobi Remote Services allow you to offload optimization computations from one or more client programs onto a cluster of servers. We provide a number of different configuration options. In the most basic configuration, a single Compute Server can accept jobs from multiple clients:



More sophisticated configurations are also possible. For example, you can have a Cluster Manager that manages access to multiple Compute Server nodes:

The different configuration options are discussed in a *later section*.

Client programs offload computation using the standard Gurobi *language APIs*. In most cases, users can write their programs without considering where they will run, and can decide at runtime whether to run them locally or on a Compute Server cluster.



Jobs submitted to a Compute Server cluster are *queued and load-balanced*. Jobs can be submitted to run either *interactively or non-interactively*. You can run your optimization jobs on a single Compute Server node, or you can choose a *distributed algorithm* to use multiple nodes in your cluster to work on a single problem.

1.1.1 Client API

When considering a program that uses Gurobi Remote Services, you can think of the optimization as being split into two parts: the client(s) and the Compute Server. A client program builds an optimization model using any of the standard Gurobi interfaces (C, C++, Java, .NET, Python, MATLAB, R). This happens in the left box of this figure:

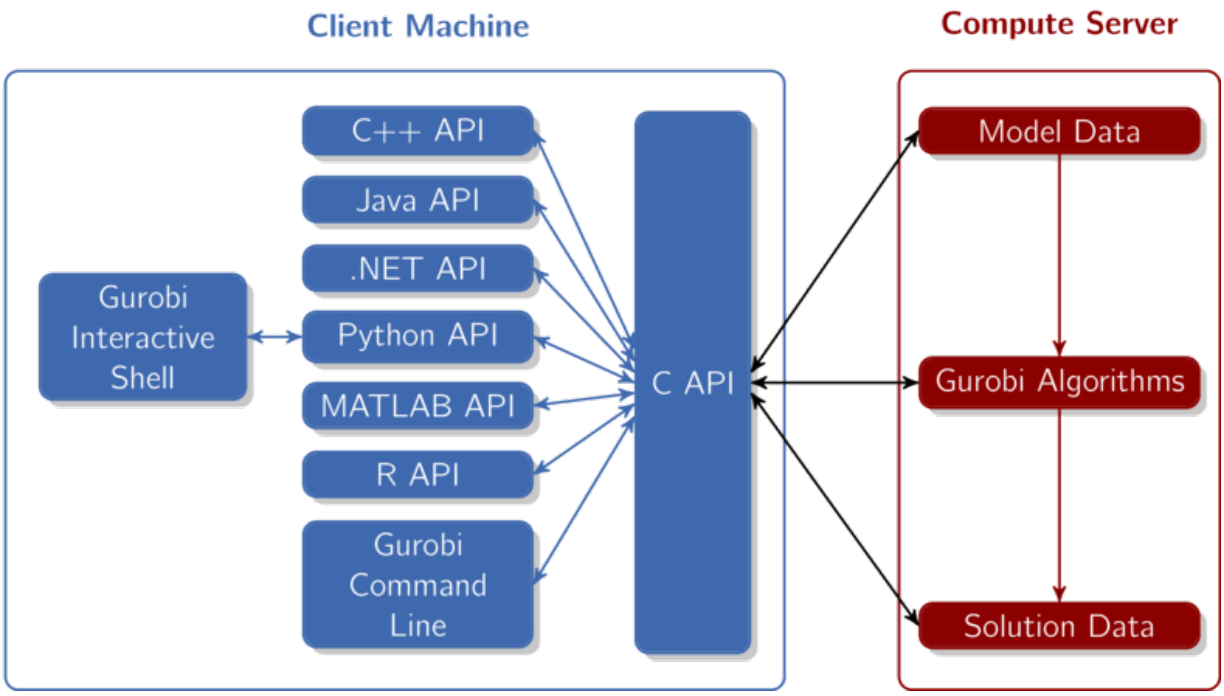
All of our APIs sit on top of our C API. The C API is in charge of building the internal model data structures, invoking the Gurobi algorithms, retrieving solution information, etc. When running Gurobi on a single machine, the C API would build the necessary data structures in local memory. In a Compute Server environment, the C layer transparently ships the data off to the Compute Server. The Gurobi algorithms take the data stored in these data structures as input and produce solution data as output.

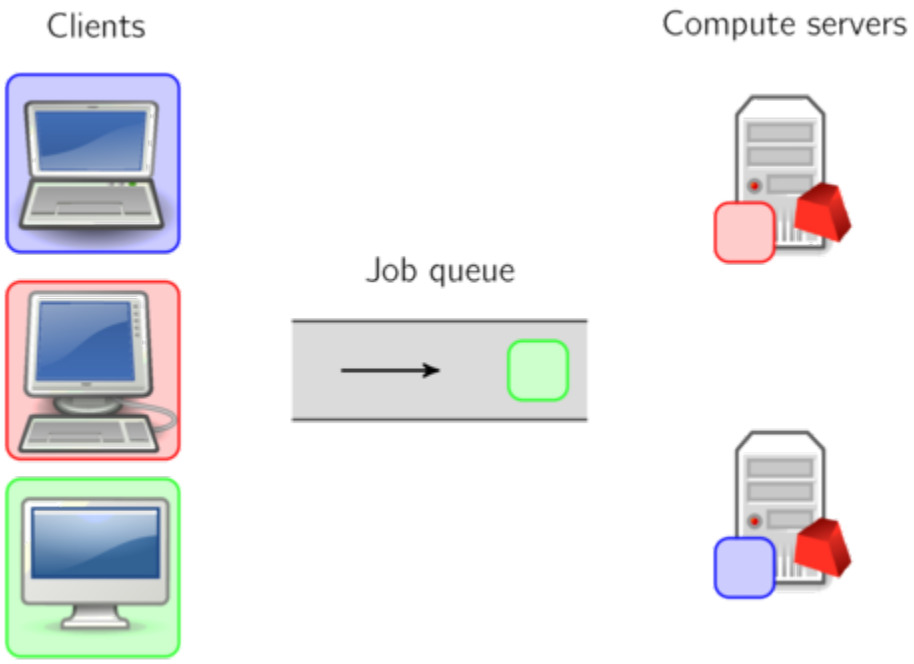
While the Gurobi Compute Server is meant to be transparent to both developers and users, there are a few aspects of Compute Server usage that you do need to be aware of. These include performance considerations, APIs for configuring client programs, and a few features that are not supported for Compute Server applications. These topics will be discussed *later in this document*.

1.1.2 Queuing and Load Balancing

Gurobi Remote Services support queuing and load balancing. You can set a limit on the number of simultaneous jobs each Compute Server will run. When this limit has been reached, subsequent jobs will be queued. If you have multiple Compute Server nodes configured in a cluster, the current job load is automatically balanced among the available servers.

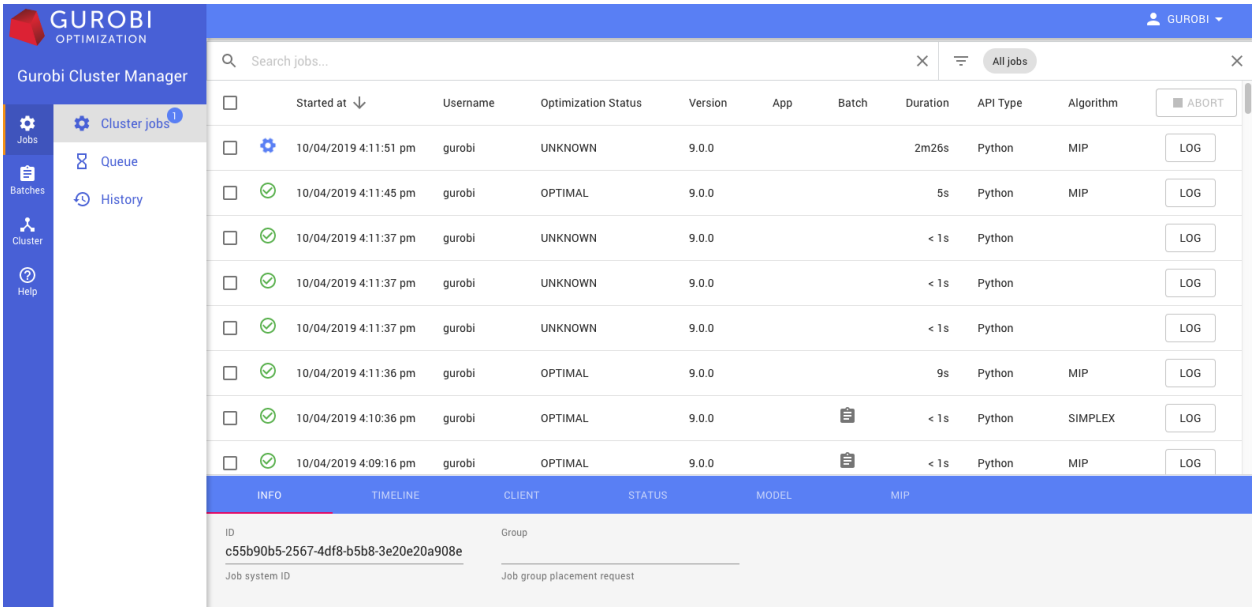
By default, the Gurobi job queue is serviced in a First-In, First-Out (FIFO) fashion. However, jobs can be given different priorities. Jobs with higher priorities are then selected from the queue before jobs with lower priorities.


Client Machine
Compute Server
C++ API
Java API
.NET API
Python API
MATLAB API
R API
Gurobi Command Line
Gurobi Interactive Shell
C API
Model Data
Gurobi Algorithms
Solution Data


Clients
Compute servers
Job queue

1.1.3 Cluster Manager

The optional Gurobi Cluster Manager adds a number of additional features. It improves security by managing user accounts, thus requiring each user or application to be authenticated. It also keeps a record of past optimization jobs, which allows you to retrieve logs and metadata. It also provides a complete Web User Interface:



This interface allows you to monitor cluster nodes and active optimization jobs, and also to retrieve logs and other information for both active and previously-completed jobs.

Finally, the Cluster Manager enables batch optimization. It receives and stages input data, and stores solutions for later retrieval.

Further information about the Cluster Manager will be presented in a *later section*.

1.1.4 Interactive and Non-Interactive Optimization

The standard approach to using a Compute Server is in an interactive fashion, where the client stays connected to the server until the job completes. The alternative is for the client to submit a batch to the server and then immediately disconnect. The client can come back later to query the status of the job and retrieve the solution when the batch is complete.

As we just noted, batch optimization requires a Cluster Manager. The Cluster Manager takes responsibility for storing the optimization model to be solved, submitting a job to the Compute Server cluster, and retrieving and storing the results of that job when it finishes (including the optimization status, the optimization log, the solution, any errors encountered, etc.).

Additional information on batch optimization can be found in a *later section*.

1.1.5 Distributed Algorithms

Gurobi Optimizer implements a number of distributed algorithms that allow you to use multiple machines to solve a problem faster. Available distributed algorithms are:

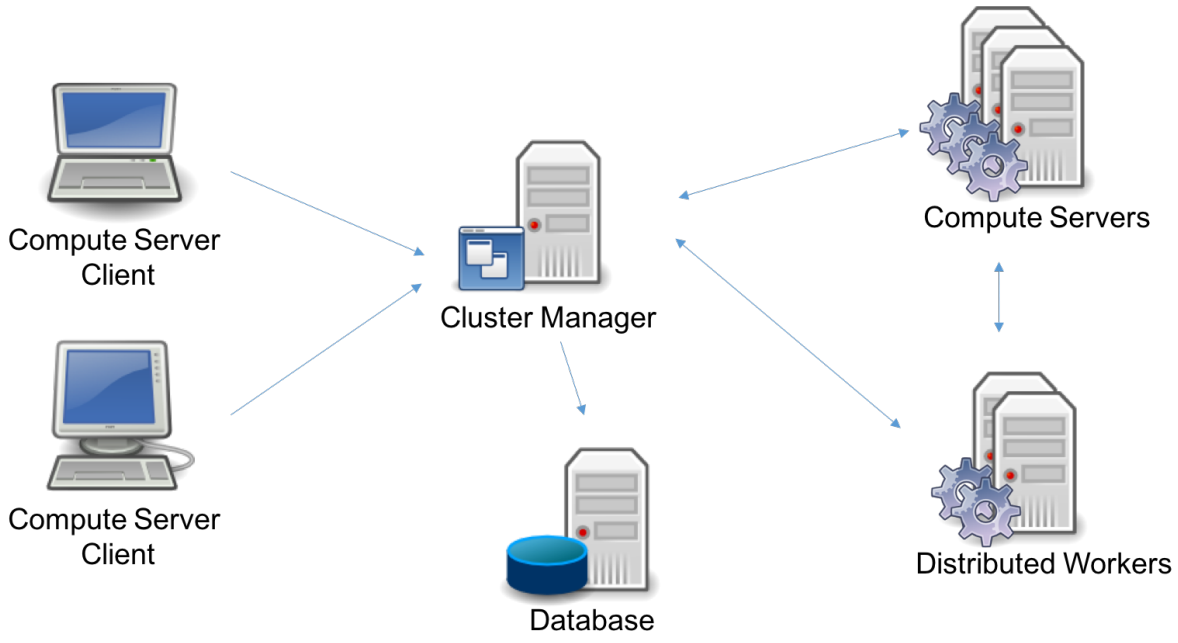
- A distributed MIP solver, which allows you to divide the work of solving a single MIP model among multiple machines. A manager machine passes problem data to a set of worker machines to coordinate the overall solution process.
- A distributed concurrent solver, which allows you to use multiple machines to solve an LP or MIP model. Unlike the distributed MIP solver, the concurrent solver doesn't divide the work among machines. Instead, each machine uses a different strategy to solve the same problem, with the hope that one strategy will be particularly effective and will finish much earlier than the others. For some problems, this concurrent approach can be more effective than attempting to divide up the work.
- Distributed parameter tuning, which automatically searches for parameter settings that improve performance on your optimization model (or set of models). Tuning solves your model(s) with a variety of parameter settings, measuring the performance obtained by each set, and then uses the results to identify the settings that produce the best overall performance. The distributed version of tuning performs these trials on multiple machines, which makes the overall tuning process run much faster.

These distributed algorithms are designed to be nearly transparent to the user. The user simply modifies a few parameters, and the work of distributing the computation among multiple machines is handled behind the scenes by the Gurobi library.

Additional information about distributed algorithms can be found in a *later section*.

1.2 Architecture

Let us now consider the roles of the different Remote Services components. Consider a Remote Services deployment:



The deployment may consist of five distinct components: the *Clients*, the *Cluster Manager*, the *Database*, the *Compute Server nodes*, and the *Distributed Worker nodes*. Several of these are optional, and a few can be replicated for high

availability. This gives a variety of topology options, which we'll discuss *shortly*. First, let us consider the components individually.

1.2.1 Cluster Manager

The Cluster Manager is the central component of the architecture. It provides the following functions:

- **Security.** The Cluster Manager is in charge of authenticating and authorizing all access to the cluster. It does this by managing user accounts and API keys, and by controlling access to all Remote Services nodes (Compute Servers or Distributed Workers).
- **Cluster Monitoring.** The Cluster Manager gives visibility to all operations on the cluster: available nodes, licenses, and jobs. It also records and retains job history, including detailed metadata and engine logs.
- **Batch Management.** The Cluster Manager controls the batch creation process and the storage of input models and output solutions. It also keeps a history of batches. Internally, it communicates with the nodes to submit and monitor batch jobs.
- **REST API.** All of the functions provided by the Cluster Manager are exposed in a REST API. This REST API is used by all built-in clients: `gurobi_cl`, `grbtune`, `grbcluster`, and the Web User Interface. The REST API can also be used by user programs.
- **Web User Interface.** The Cluster Manager includes a Web Application Server that provides a complete and secured Web User Interface to your Compute Server cluster.

The Cluster Manager is optional. You can build a *self-managed Remote Services cluster*, but it will be missing many features.

Cluster Manager installation is covered *in this section*.

1.2.2 Database

The database supports the Cluster Manager. It stores a variety of information, including data with long lifespans, like user accounts, API keys, history information for jobs and batches, and data with shorter lifespans, like input models and their solutions for batch optimization.

How much space does this database require? This will depend primarily on the expected sizes of input and output data for batches. The Cluster Manager will capture and store the complete model at the time a batch is created, and it will store the solution once the model has been solved. These will be retained until they are discarded by the user, or until they expire (the retention policy can be configured by the Cluster Manager system administrator, in the settings section). The data is compressed, but it can still be quite large. To limit the total size of the database, we suggest that you discard batches when you are done with them. Note that discarding a batch doesn't discard the associated (small) metadata; that is kept in the cluster history.

The Cluster Manager can be connected to three types of database servers (see the release notes for the supported versions):

- MongoDB, deployed on-premise, on the Cloud, or hosted by a SaaS provider.
- Amazon Web Services (AWS) DocumentDB, when deployed to AWS.
- Azure CosmosDB, when deployed to Microsoft Azure.

Cluster Manager users must *install and configure their own database* as part of the Compute Server installation process. It can be deployed as a single node or as a cluster for high availability.

1.2.3 Compute Server Node

A Compute Server node is where optimization jobs are executed. Each such node has a job limit that indicates how many jobs can be executed on that node simultaneously. The limit should reflect the capacity of the machine and typical job characteristics. Compute Server nodes support advanced capabilities such as job queueing and load-balancing. Deploying a Compute Server requires a Gurobi license.

Compute Server node installation is covered *in this section*.

1.2.4 Distributed Worker Node

A Distributed Worker node can only be used as a worker in a distributed algorithm. Only one job can run on such a node at a time and it does not support queueing or load balancing. This type of node does not require a Gurobi License.

Distributed Worker installation is covered *in this section*.

1.2.5 Architecture Topologies

Let us now review a few common deployment configurations.

1.2.6 Cluster Manager with a single node

In this deployment, we only need to deploy one instance of a Cluster Manager with the Database and a single Compute Server node. This is appropriate for small environments so that you can offload simple optimization tasks to one Compute Server.

1.2.7 Cluster Manager with multiple nodes

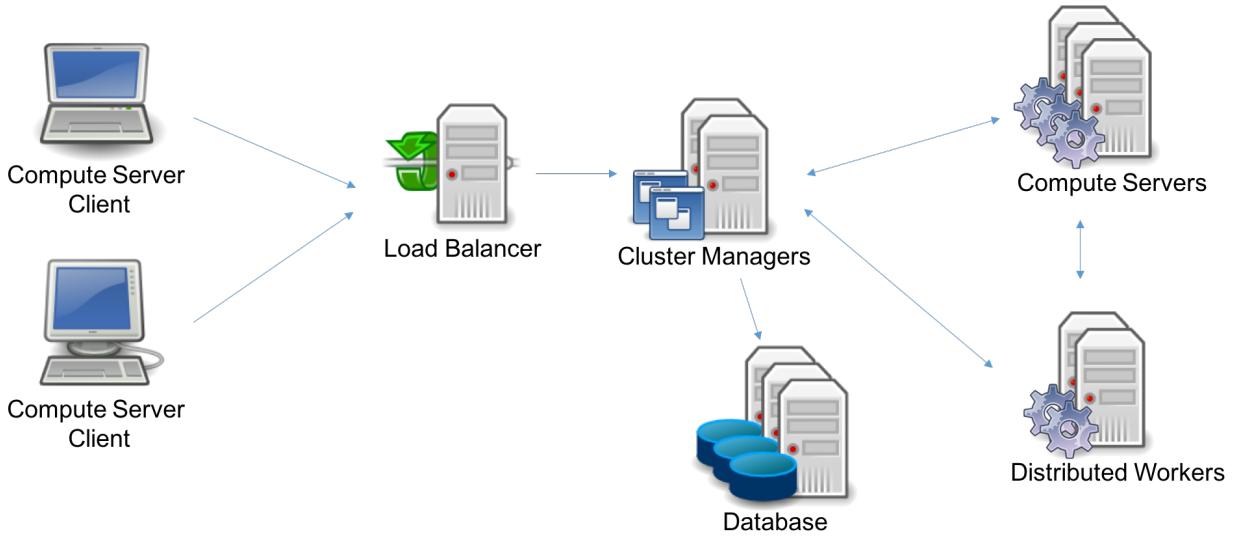
If you need to handle more jobs concurrently, you will need to add more Compute Server nodes. Also, if you want to run distributed algorithms, several Distributed Worker nodes will be needed. To this end, you can deploy one instance of the Cluster Manager (with a Database), and connect those nodes to the Cluster Manager.

1.2.8 Scalable Cluster Manager

If you have even more concurrent users, or if you need a scalable and high available architecture, several instances of the Cluster Manager can be started. In this case, you may need to install and set up a regular HTTP load balancer (such as `Nginx`) in front of the Cluster Manager instances. Cluster Manager server instances are stateless and can be scaled up or down.

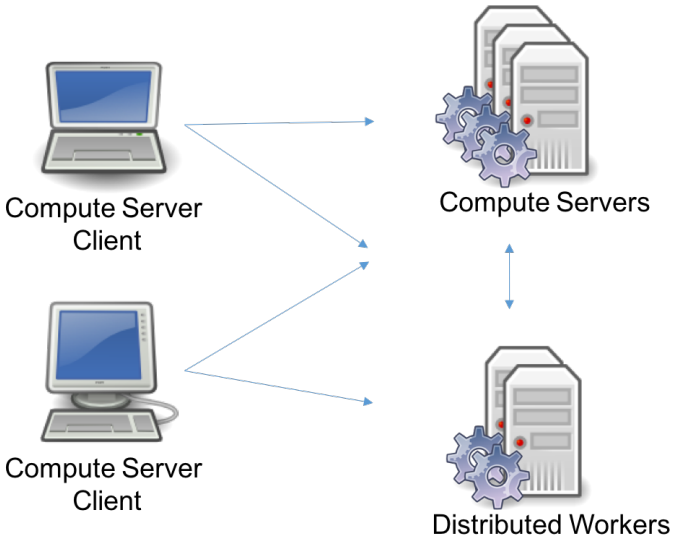
The database itself can be deployed in a cluster. In a MongoDB cluster, one of the nodes is chosen dynamically as the primary, while the others are deemed secondary. Secondary nodes replicate the data from the primary node. In the event of a failure of the primary node, the Cluster Manager will automatically reconnect to a new primary node and continue to operate.

In this deployment, several Compute Server nodes are also recommended. In the event of a node failure, any jobs currently running on the failed node will fail, but new jobs will continue to be processed on the remaining nodes.



1.2.9 Self-Managed Cluster

Finally, Compute Server nodes and Distributed Worker nodes can be deployed by themselves, without a Cluster Manager or a Database. This was actually the only option in Gurobi version 8 and earlier. In this configuration, you will not benefit from the latest features: secured access using user accounts and API keys, persistent job history, batch management, and the Web User Interface.



1.3 Security

Gurobi Remote Services define a set of *user roles* to control privileged access. Access to the Cluster Manager is *authenticated*. For local accounts, a *password policy* can be defined. In addition, the Cluster Manager can be integrated with an *LDAP* repository for centralized account management. Communication can be *encrypted* using TLS v1.2 or later. If a Cluster Manager is not deployed, communication can still be encrypted, but access is controlled through a set of *predefined passwords* instead.

1.3.1 User Roles

Users of Gurobi Remote Services will fall into one of four possible roles: *system administrator*, *administrator*, *standard user*, or *read-only user*. The system administrator is in charge of setting up the cluster, adding and removing nodes, etc. Administrators monitor usage of the cluster. They can monitor the length of the server queue, kill jobs, etc. Standard users are the programs running on client machines that ultimately submit jobs or batches to the cluster. Read-only users can only monitor jobs submitted by other users.

The Gurobi distribution includes a number of tools that are relevant to the people in these roles. These are all covered in much more detail later on, but we will briefly describe how they fit with the various roles here.

1.3.2 System Administrator

The system administrator installs and manages a Remote Services cluster and the different components. Gurobi Remote Services provides the following tools to help with this:

- *grb_rs* is the program that runs on the Compute Server and Distributed Worker nodes. The system administrator will need to configure and start it on all of the nodes of a Remote Services cluster.
- *grb_rsm* is the program that runs the Cluster Manager. The system administrator will need to configure and start it on one or more machines, as needed. The system administrator will also need to set up the Database and configure its connection.
- *grbcluster* is used to issue commands to an already-running cluster. Examples of system administrator commands include adding or removing nodes, and enabling or disabling job processing on a cluster. This tool provides a number of commands; type `grbcluster --help` for a full list.
- Finally, most of the important responsibilities of the system administrator, including user management and cluster health monitoring, can also be performed through the Web User Interface of the Cluster Manager.

For more details, please refer to the section on *setting up and administering a cluster*.

1.3.3 Administrator

An administrator monitors and manages the flow of jobs through a Remote Services cluster. Examples of administrator commands include aborting jobs, changing cluster parameters and checking licenses. The primary tool for doing so is *grbcluster*. You can get a full list of available commands by typing `grbcluster --help`. All of these functions are also exposed in the Web User Interface of the Cluster Manager.

1.3.4 Standard Client

A Remote Services client submits jobs or batches to the cluster. This is done through a user application or through the Gurobi command-line tool *gurobi_cl* (which is documented in the Gurobi Command-Line Tool section of the Gurobi Reference Manual). Submitting a job to a Remote Services cluster is typically just a matter of running the appropriate program. We will provide a simple example in the next section.

Clients can also use the *grbcluster* command to monitor the state of their jobs and of the Remote Services queue. Example commands include listing active jobs, listing recently executed jobs, displaying the log of a recent job, etc. You can get a full list of available commands by typing `grbcluster --help`. `grbcluster` can also be used to submit batches.

Finally, clients can access the Web User Interface of the Cluster Manager. All of the functions provided by `grbcluster` are available in the web application, including submitting batches using a drag-and-drop interface.

1.3.5 Read-only Users

Read-only users can only monitor optimization tasks. They can list jobs and batches, access history, display the log of a job, etc. They are not allowed to submit jobs or batches to the cluster, nor can they abort jobs.

1.3.6 Authentication

The Cluster Manager authenticates all communication using one of two approaches: interactive login using a username and password, or an API key.

When a client provides a username and password, a JWT token is returned that is valid for a relatively short period of time (default is 8 hours and can be changed in the Cluster Manager configuration). This is handy when using the Web User Interface or command-line tools such as `gurobi_cl` or `grbcluster`.

An API key is composed of an access ID and a secret key. API keys are the recommended method for connecting to the Cluster Manager from an application. When creating an API key, you can specify an optional application name and a description to help keep track of how the key is being used. Once a key is created, you can download an associated client license file, which contains the API access ID, the secret key, and the Cluster Manager URL. This file can be used by client applications and command-line tools to connect to the Cluster Manager. The Cluster Manager keeps track of the timestamp and IP address of the last API key usage. The owner of the API key or the system administrator can enable or disable an API key. These features simplify the task of monitoring API keys, detecting unwanted usage, and safely rotating keys by disabling previous keys before permanently deleting them.

For each account, the system administrator can enable or disable interactive login or API key authentication. This can be done at creation time, or it can be done later by editing the account properties. An account that only allows interactive login will not be allowed to create, use, or manage API keys. An account that only allows API key authentication (known as a `system account`) can only be used for programmatic access through the REST API.

The system administrator can disable and later reenable a user account. When an account has been disabled, interactive login and/or API key authentication will fail and access to the Cluster Manager will not be granted. Disabled accounts will appear with a grayed icon in the user account table. The tooltip will indicate the reason.

To simplify installation, the Cluster Manager initially has three default users with predefined passwords:

- standard user: `gurobi` / `pass`
- administrator: `admin` / `admin`
- system administrator: `sysadmin` / `cluster`

You should of course change the passwords or delete these accounts before actually using the cluster.

1.3.7 Password Policy

The Cluster Manager supports password policies to ensure that passwords for local accounts match configurable security standards. Through a password policy, the system administrator can specify the minimum length of a password and can require that a password contain a mix of upper and lower case characters, digits, or symbols. Any changes to a policy will apply only when new passwords are created; previously created password will remain valid. In addition, the system administrator can define a maximum number of failed login attempts before the account is locked. When a user account has been locked, a message will appear on the login page, and the user will be asked to contact a system administrator. The system administrator can then change the password to unlock the user account.

1.3.8 LDAP Integration

The Cluster Manager can be integrated with an LDAP repository. This integration can be configured in the Cluster Manager settings section. The system administrator can specify connection parameters (including the use of LDAPS for encrypted communication), account filtering, and account mapping. Once activated, users will be given access based on the user accounts defined in the LDAP server.

Accounts with a system administrator role can log in using their local passwords, which enables them to administer the cluster even if there is a problem with the LDAP configuration. System administrators will need to log in using the special login page by following the "System Administrator Log in" link at the top of the main login page. System accounts (i.e., accounts with no interactive login) can always gain access using API keys only.

The Cluster Manager continually synchronizes user accounts with the LDAP server in the background. If a given user account is no longer mapped to the LDAP filter in place, it will be disabled. In particular, if an account was created before the integration with the LDAP server and if it is not mapped to the defined LDAP filter in place, it will be disabled. The same policy will apply during migration from local accounts to LDAP. Two important exceptions are system administrator accounts and system accounts, which will not be disabled for this reason.

1.3.9 Encryption

All of the components deployed in a Remote Services cluster can support TLS-encrypted communication. The Cluster Manager and the Compute Server nodes can be configured to use HTTPS with TLS v1.2 or later.

The server components support configurable TLS v1.2 cipher suites and policies with a configuration property (`TLS_CIPHERS`). You can list the supported cipher suites and policies using the (`grb_rs ciphers`) command. TLS v1.3 will be used if the client supports it, and in that case, standard cipher suites for this protocol version will automatically be selected. When using the Gurobi library, TLS v1.3 will be used only when the client is running on a Linux platform.

MongoDB, Amazon DocumentDB, and Azure CosmosDB also support encrypted communications and data encryption at rest, if necessary.

In addition, you have the option to use either the Cluster Manager itself or the load balancer in front of the Cluster Manager to remove the TLS encryption. In this case, HTTPS is used by clients, but internal communication between the Cluster Manager and the nodes would be unencrypted using HTTP. This is convenient when the cluster nodes reside in an isolated, secure network.

1.3.10 Security in a Self-Managed Cluster

When a Remote Services cluster is deployed without a Cluster Manager, authentication is more limited. Each node authenticates access according to predefined passwords, which are stored and optionally hashed in the configuration file. There is a password for each role (standard user, administrator and system administrator). All nodes of the cluster must use the same passwords, and they cannot be changed dynamically. Note that communication can also be encrypted using HTTPS.

1.4 Simple Example

After your cluster has been set up (setup is covered in *this section*), you can submit a job or a batch using either a programming language API, the command-line tools, or the Web User Interface for your Cluster Manager. This section provides a few short examples that use the command-line tools. More complete descriptions of the various interfaces and options will come in a *later section*.

1.4.1 Log In to the Cluster

The first step in submitting a job to the cluster is to log in to the Cluster Manager with the `grbcluster login` command:

```
> grbcluster login --manager=http://localhost:61080 -u=gurobi
info  : Using client license file '/Users/john/gurobi.lic'
Password for gurobi:
info  : User gurobi connected to http://localhost:61080, session will expire on 2019-09..
↪.
```

This command indicates that you want to connect to the Cluster Manager running on port 61080 of machine `localhost` as the `gurobi` user. The output from the command first shows that the client license file `gurobi.lic` located in the home directory of the user will be used to store the connection parameters. It then prompts you for the password for the specified user (in a secure manner). After contacting the Cluster Manager, the client retrieves a session token that will expire at the indicated date and time.

Using this approach to logging in removes the need to display the user password or save it in clear text, which improves security. The session token and all of the connection parameters are saved in the client license file, so they can be used by all of the command-line tools (`gurobi_cl`, `grbtune`, and `grbcluster`). When the token session expires, the commands will fail and you will need to log in again.

1.4.2 Submitting an Interactive Job

Once you are logged in, you can use `gurobi_cl` to submit a job:

```
> gurobi_cl ResultFile=solution.sol stein9.mps
Set parameter CSManager to value "http://server1:61080"
Set parameter LogFile to value "gurobi.log"
Compute Server job ID: 1e9c304c-a5f2-4573-affa-ab924d992f7e
Capacity available on 'server1:61000' - connecting...
Established HTTP unencrypted connection
Using client license file /Users/john/gurobi.lic

Gurobi Optimizer version 13.0.0 build v13.0.0rc0 (linux64)
Copyright (c) 2022, Gurobi Optimization, LLC
```

(continues on next page)

(continued from previous page)

```
...
Gurobi Compute Server Worker version 13.0.0 build v13.0.0rc0 (linux64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
...
Optimal solution found (tolerance 1.00e-04)
Best objective 5.000000000000e+00, best bound 5.000000000000e+00, gap 0.0000%

Compute Server communication statistics:
  Sent: 0.002 MBytes in 9 msgs and 0.01s (0.26 MB/s)
  Received: 0.007 MBytes in 26 msgs and 0.09s (0.08 MB/s)
```

The initial log output indicates that a Compute Server job was created, that the Compute Server cluster had capacity available to run that job, and that an unencrypted HTTP connection was established with a server in that cluster. The log concludes with statistics about the communication performed between the client machine and the Compute Server. Note that the result file `solution.sol` is also retrieved.

This is an interactive optimization task because the connection with the job must be kept alive and the progress messages are displayed in real time. Also, stopping or killing the command terminates the job.

1.4.3 Submitting a Non-Interactive Job

You can use `grbcluster` to create a batch (i.e., a non-interactive job):

```
> grbcluster batch solve ResultFile=solution.sol misc07.mps --download
info  : Batch 5d0ea600-5068-4a0b-bee0-efa26c18f35b created
info  : Uploading misc07.mps...
info  : Batch 5d0ea600-5068-4a0b-bee0-efa26c18f35b submitted with job a9700b72...
info  : Batch 5d0ea600-5068-4a0b-bee0-efa26c18f35b status is COMPLETED
info  : Results will be stored in directory 5d0ea600-5068-4a0b-bee0-efa26c18f35b
info  : Downloading solution.sol...
info  : Downloading gurobi.log...
info  : Discarding batch data
```

This command performs a number of steps. First, a batch specification is created and the batch ID is displayed. Then, the model file is uploaded and a batch job is submitted. Once the job reaches the front of the Compute Server queue, it is processed. At that point, the batch is marked as completed and the result file with the log file is automatically downloaded to the client. By default, the directory name where the result file is stored is the batch ID. Finally, the batch data is discarded, which allows the Cluster Manager to delete the associated data from its database.

This is a non-interactive optimization task because it happens in two distinct phases. The first phase uploads the model to the server and creates a batch. The second waits for the batch to complete and retrieves the result. In general, stopping the client has no effect on a batch once it has been submitted to the Cluster Manager. Our example waits for the completion of the batch, but that's only because we used the `--download` flag. You could check on the status of the batch and download the results whenever (and wherever) they are needed, since they are stored in the Cluster Manager until they are discarded.

CHAPTER

TWO

CLUSTER SETUP AND ADMINISTRATION

This section covers the setup and administration of a Gurobi Remote Services cluster. The intended audience is the system administrator. If you are interested in using a cluster that has already been set up, you should proceed to the *next section*.

This section begins by providing a step-by-step guide for a simple Cluster Manager installation on your local machine. The goal is to help you to quickly gain a basic understanding of the role of each Remote Services component.

Then, we describe in more detail the steps and options for a full deployment. We start by describing how to *download the Remote Services package* and install it on all of the nodes in your cluster. Once this is done, the next step is to install and start the *Cluster Manager*. Note that this step is optional; you can run a self-managed Remote Services cluster (without a Cluster Manager). Next, we explain the steps required to set up a cluster with *one node*, and then the steps to expand the cluster to *multiple nodes*. We also present a discussion of the available *communication options*. Finally, we explain the process of *upgrading* an installation.

2.1 Quick Cluster Manager Installation

The rest of this section lays out the steps required to install and configure Gurobi Remote Services and the Cluster Manager. Before diving into those details, though, we first want to provide a quick, high-level overview. The intent is to give you a basic understanding of the relevant concepts and tools. We suggest that you try these steps on your local machine before performing them on your server.

1. Download and install the Gurobi client and Remote Services packages from our download page. Detailed instructions depend on your platform and are provided *in this section*.

2. Install and start a MongoDB database (as explained in their on-line guide). If you are deploying the Cluster Manager on the AWS platform, you can also use an AWS DocumentDB (as explained in their developer guide). AWS DocumentDB cannot be installed on-premises. If you are deploying the Cluster Manager on the Microsoft Azure platform, you can also use an Azure CosmosDB (as explained in their documentation). Please refer to the releases notes to check on the supported versions of each database offering.

3. Start the Cluster Manager.

   In a new terminal window, start the Cluster Manager executable:

```
> grb_rsm
info  : Gurobi Cluster Manager starting...
info  : Version is 13.0.0
info  : Connecting to database grb_rsm on 127.0.0.1:27017...
info  : Connected to database grb_rsm (version 4.0.4)
info  : Starting cluster manager server (HTTP) on port 61080...
```

The default configuration will start the Cluster Manager on port 61080 and will connect to the database on the local machine. If you have installed the database with other options or want to use an existing database, you can provide a database connection string with the `--database` flag:

```
> grb_rsm --database=....
```

The Cluster Manager has several important options that are detailed *in this section*.

4. Get your Gurobi license.

   Follow the instructions in the Gurobi User Portal to retrieve your license. To avoid conflicts with client license files, you should place your license file in a non-default location:

```
grbgetkey 8f15037e-eae7-4831-9a88-ffe079eabdeb
info  : grbgetkey version 13.0.0
info  : Contacting Gurobi key server...
info  : Key for license ID XXXXX was successfully retrieved
info  : Saving license key...

In which directory would you like to store the Gurobi license key file?
[hit Enter to store it in /Users/john]: /Users/john/tutorial

info  : License XXXXX written to file /Users/john/tutorial/gurobi.lic
info  : You may have saved the license key to a non-default location
info  : You need to set the environment variable GRB_LICENSE_FILE before you can␣
↪use this license key
info  : GRB_LICENSE_FILE=/Users/john/tutorial/gurobi.lic
```

5. Connect a Compute Server node.

   In a new terminal, set the license file variable. For Linux and macOS, use this command:

```
export GRB_LICENSE_FILE=/Users/john/tutorial/gurobi.lic
```

   For Windows, use this command instead:

```
SET GRB_LICENSE_FILE=/Users/john/tutorial/gurobi.lic
```

   Then, start a Remote Services agent, using a few parameters to connect to the manager and to run on port 61000:

```
> grb_rs --manager=http://localhost:61080 --port=61000
info  : Gurobi Remote Services starting...
info  : Version is 13.0.0
info  : Accepting worker registration on port 64121...
info  : Starting API server (HTTP) on port 61000...
info  : Joining cluster from manager
```

   The Remote Services Agent has several important options that are detailed *in this section*.

6. Open the Cluster Manager Web UI in a browser at `http://localhost:61080`.

   You will be asked to log in. You can use one of the three predefined users and passwords (`gurobi/pass`, `admin/admin`, `sysadmin/cluster`). If you navigate to the `cluster` section, you should see the Compute Server node status display.

7. Log in to the Cluster Manager using the command-line tools.

In a new terminal, log in to the Cluster Manager using the appropriate connection parameters. Connection information is stored into your `gurobi.lic` client license file once you connect, so you won't need to include these parameters with each future command.

```
grbcluster login --manager=http://localhost:61080 --username=gurobi
```

Enter the default password 'pass' when prompted.

More options and detailed client configuration is explained *in a following section*.

8. Submit jobs and batches from the command-line tools or the programming language APIs.

   Once you have logged in, you are ready to submit optimizations requests. In the following examples, we will refer to the installation directory of the main Gurobi tools and libraries as <gurobi_installation>.

   You can submit an interactive job:

```
gurobi_cl ResultFile=solution.sol <gurobi_installation>/examples/data/misc07.mps
```

   You can also submit a batch job and wait for the completion to download the results:

```
grbcluster batch solve ResultFile=solution.sol <gurobi_installation>/examples/data/
↪misc07.mps --download
```

   Finally, you can submit a batch with the Python API. The Gurobi distribution includes a complete example:

```
python <gurobi_installation>/examples/python/workforce_batchmode.py
```

   The followup sections give more details on *the command line tools* and *the programming language APIs*.

Let's now dive into more detailed discussions of these steps.

2.2 Installing the Remote Services Package

The Gurobi Remote Services package must be installed on all of the machines that will be part of your cluster. This includes the Compute Server nodes, the Distributed Worker nodes, and the Cluster Manager.

The first step is to download the installer from our download page. You will need to find your platform and choose the corresponding file to download.

Make a note of the name and location of the downloaded file.

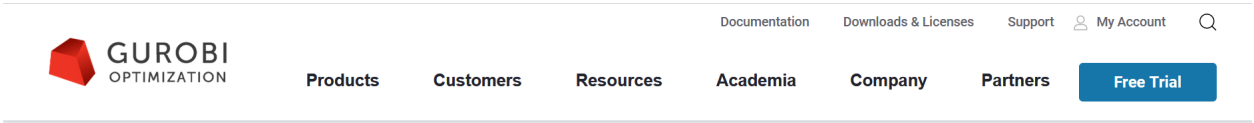
Your next step will depend on your platform:

2.2.1 Linux Installation

On Linux, your next step is to choose a destination directory. We recommend `/opt` for a shared installation (you may need administrator privileges), but other directories will work as well. Copy the Remote Services distribution to the destination directory and extract the contents. Extraction is done with the following command:

```
tar xvfz gurobi_server13.0.0_linux64.tar.gz
```

This command will create a sub-directory `gurobi_server1300/linux64` that contains the complete Linux Remote Services distribution. Assuming that you extracted the Gurobi server archive in the `/opt` directory, your <installdir> (which we'll refer to throughout this document) will be `/opt/gurobi_server1300/linux64`.



Home › Gurobi Optimizer – Get the Software

Gurobi Optimizer – Get the Software

Gurobi Optimizer is the Gurobi optimization libraries. In addition to the software, the corresponding README file contains installation instructions. Here is the list of bug fixes for each release.

Gurobi Remote Services is an optional set of Gurobi features that allow a cluster of one or more machines to perform Gurobi computations on behalf of other machines. This software is only for customers who have a license for Gurobi Compute Server or for distributed optimization algorithms.

Current Version

Gurobi Optimizer

	x64 Windows	x64 Linux	MacOS Universal2	x64 AIX	arm64 Linux
9.5.1 Read Me Release-Notes	Gurobi-9.5.1-win64.msi	gurobi9.5.1_linux64.tar.gz	gurobi9.5.1_macos_universal2.pkg	gurobi9.5.1_power64.tar.gz	gurobi9.5.1_armlinux64.tar.gz (experimental)[1]
md5 Checksum	ea726547d2680d56ab3ab965db995819	e3e34d33ca324bb818d02264350671d3	a1786849ff3f14041af102a3fe3c8ad1	3401d854dbec729c953431b58e0cea94	dc8f135c1c4140c4174f7081b3c13753

[1]The new arm64 Linux port is experimental; we encourage users to try it and report any issues. However, it should not be used in production applications yet.

Gurobi Remote Services

	x64 Windows	x64 Linux	MacOS Universal2
9.5.1 Release-Notes	GurobiServer-9.5.1-win64.msi	gurobi_server9.5.1_linux64.tar.gz	gurobi_server9.5.1_macos_universal2.pkg
md5 Checksum	f9d79b578416dc2e89fb454bdea474ed	651ae9ecf53986f91cb29573da32188a	34a1dfa4f1764c3817956a84da0e7e66

The Gurobi Optimizer makes use of several executable files. In order to allow these files to be found when needed, you will have to modify your search path. Specifically, your PATH environment variable should be extended to include <installdir>/bin. Users of the bash shell should add the following line to their .bashrc file:

```
export PATH="${PATH}:/opt/gurobi_server1300/linux64/bin"
```

Users of the csh shell should add the following line to their .cshrc file:

```
setenv PATH "${PATH}:/opt/gurobi_server1300/linux64/bin"
```

You'll need to close your current terminal window and open a new one after you have made these changes in order to pick up the new settings.

In some Linux distributions, applications launched from the Linux desktop won't read .bashrc (or .cshrc). You may need to set the Gurobi environment variables in .bash_profile or .profile instead. Unfortunately, the details of where to set these variables vary widely among different Linux distributions. We suggest that you consult the documentation for your distribution if you run into trouble.

2.2.2 macOS Installation

On macOS, your next step once you've downloaded the Gurobi Remote Services package from our website (e.g., gurobi_server13.0.0_macos_universal2.pkg for Gurobi 13.0.0) is to double-click on the installer and follow the prompts. By default, the installer will place the Gurobi Remote Services 13.0.0 files in /Library/gurobi_server1300/macos_universal2 (note that this is the *system* /Library directory, not your personal `` /Library`` directory). Your <installdir> (which we'll refer to throughout this document) will be /Library/gurobi_server1300/macos_universal2.

2.2.3 Windows Installation

On Windows, your next step is to double-click on the Gurobi Remote Services installer that you downloaded from our website (e.g., GurobiServer-13.0.0-win64.msi for Gurobi 13.0.0).

Note: if you selected *Run* when downloading you've already run the installer and don't need to do it again.

By default, the installer will place the Gurobi 13.0.0 files in directory c:/gurobi_server1300/win64. The installer gives you the option to change the installation target. We'll refer to the installation directory as <installdir>.

2.3 Installing a Cluster Manager

Setting up the optional Cluster Manager involves a few steps. You first need to install the *MongoDB database*, which the Cluster Manager uses to store its data. Then, the *Cluster Manager server* must be *configured* and started (as a *standard process* or as a *service*). Finally, you will need to *verify* your installation.

2.3.1 Installing the Database

The Cluster Manager uses a database to store the data associated with several important features: user profiles, API keys, job history, batch definitions, and batch data.

The Cluster Manager can be connected to three types of database servers (see the release notes for the supported versions):

- MongoDB, deployed on-premise, on the Cloud, or hosted by a SaaS provider.
- Amazon Web Services (AWS) DocumentDB, when deployed to AWS.
- Azure CosmosDB, when deployed to Microsoft Azure.

MongoDB offers various configuration options, including clustering and encryption over the wire and at rest. Please follow the steps as explained in the official MongoDB documentation. The installation steps are well explained for each platform. When the installation is complete, the MongoDB daemon `mongod` should be running and ready for connections on port 27017 (the default). One other option, if you want to avoid installing the database yourself, would be to sign up for a hosted solution in the Cloud.

You can also set up an AWS DocumentDB database if you are deploying the Cluster Manager on the AWS platform (please, refer to their developer guide). DocumentDB offers various configuration options, including clustering and encryption over the wire and at rest. Note that AWS DocumentDB cannot be installed on-premises.

If you are deploying the Cluster Manager on the Microsoft Azure platform, you can also use an Azure CosmosDB database (as explained in their documentation).

Once the database is installed or provisioned in the Cloud, please make a note of the connection string (which is a URL that will be provided to you during installation). We will need this URL to connect the Cluster Manager to the database. The URL has the following form. You should of course substitute the MongoDB username and password that you chose when installing the database:

```
mongodb://[username:password@]host1[:port1][,...hostN[:portN]][/[database][?options]]
```

If you have installed MongoDB on your personal machine for testing purposes, the connection string URL should be the following:

```
mongodb://localhost:27017
```

You can check that your connection string is correct by using the mongo shell and providing the connection string as the first argument:

```
> mongosh mongodb://localhost:27017
Connecting to:       mongodb://localhost:27017
Using MongoDB:          6.0.18
Using Mongosh:          2.3.1
```

2.3.2 Cluster Manager Server (grb_rsm)

You will need to choose one or more machines to act as your Cluster Manager(s). The primary tasks of the Cluster Manager are to provide an API gateway to the cluster and to manage cluster nodes. The Cluster Manager also acts as a web server for the Web User Interface. The cluster manager must be reachable on your network from all client machines and from all cluster nodes.

You will need to run the *Cluster Manager executable* (`grb_rsm`) on your Cluster Manager(s). If you wish to set up a scalable and high-available deployment, you can install and start several instances of the server and place a load balancer such as `Nginx` in front of these servers.

The grb_rsm executable provides several commands and flags to help with configuration and execution. We will review these commands step by step in the following sections. You can see the full list of commands in the *reference section* or by using the command-line help:

```
> grb_rsm --help
```

2.3.3 Configuring the Cluster Manager

The Cluster Manager server has a number of configuration properties that affect its behavior. These can be controlled using a grb_rsm.cnf configuration file. The installation package includes a predefined configuration file that can be used as a starting point (<installdir>/bin/grb_rsm.cnf).

The simplest way to modify the parameters is to edit the default configuration file. Other options are available, though. The grb_rsm process uses the following precedence rules:

- First priority: properties set with a command-line flag (using --config)
- Second priority: a configuration file in the current directory
- Third priority: a configuration file in a fixed, shared directory (C:\gurobi, /opt/gurobi, /Library/gurobi for Windows, Linux, and macOS platforms, respectively)
- Fourth priority: a configuration file in the directory where grb_rsm is located

Most of the properties that are configured through this file are related to communication options or the database connection. The configuration file is read once, when grb_rsm first starts. Subsequent changes to the file won't affect parameter values on a running server.

2.3.4 Configuration file format

The configuration file contains a list of properties of the form PROPERTY=value. Lines that begin with the # symbol are treated as comments and are ignored. Here is an example:

```
# grb_rsm.cnf configuration file
PORT=61080
CLUSTER_TOKEN=GRBTK-BzlUTKg9M/+HUvOpy/EPebc1CsttzOfdrfQshL4QkLm1FA==
DB_URI=mongodb://127.0.0.1:27017
```

While you could create this file from scratch, we recommend you start with the version that is included with the product and modify it instead.

The grb_rsm properties command lists all of the available properties, their default values, and provides documentation for each. Some can be overridden on the grb_rsm command line; the grb_rsm properties command shows the name of the command-line flag you would use. Here are some of the more important ones:

CLUSTER_TOKEN
: The token is a private key that enables different nodes to join the same cluster. All nodes of a cluster and the Cluster Manager must have the same token. We recommended that you generate a brand new token when you set up your cluster. The grb_rs token command will generate a random token, which you can copy into the configuration file.

DB_URI
: This is the connection string to your database.

PORT
: This property indicates what port to use for HTTP or HTTPS communication between the clients and the Cluster Manager. By default, it will use the port 61080.

These properties let you configure some system-level options. You will find additional configuration properties in the settings page of the Cluster Manager. The settings are stored in the database and shared between the different Cluster Manager instances.

2.3.5 Starting the Cluster Manager as a Process

Once you have installed the Remote Services package, you can start `grb_rsm` as a standard process from a terminal window by simply typing `grb_rsm`. This will start the Cluster Manager server on the default port (port 61080):

```
> grb_rsm
info  : Gurobi Cluster Manager starting...
info  : Platform is linux
info  : Version is 13.0.0\ (build v13.0.0rc0)
info  : Connecting to database grb_rsm on 127.0.0.1:27017...
info  : Connected to database grb_rsm (version 4.0.4, host Server1)
info  : Checking 0 cluster nodes
info  : Creating proxy with MaxIdleConns=200 MaxIdleConnsPerHost=32 IdleConnTimeout=130
info  : Starting cluster manager server (HTTP) on port 61080...
```

If you'd like to run `grb_rsm` on a non-default port, use the `--port` flag or set the `PORT` property in the configuration file. For example:

```
> grb_rsm --port=8080
```

2.3.6 Starting the Cluster Manager as a Service

While you always have the option of running `grb_rsm` from a terminal and leaving the process running in the background, we recommended that you start it as a service instead, especially in a production deployment. The advantage of a service is that it will automatically restart itself if the computer is restarted or if the process terminates unexpectedly.

`grb_rsm` provides several commands that help you to set it up as a service. These must be executed with administrator privileges:

grb_rsm install
: Install the service. The details of exactly what this involves depend on the host operating system type and version: this uses `systemd` or `upstart` on Linux, `launchd` on macOS, and *Windows services* on Windows.

grb_rsm start
: Start the service (and install it if it hasn't already been installed).

grb_rsm stop
: Stop the service.

grb_rsm restart
: Stop and then start the service.

grb_rsm uninstall
: Uninstall the service.

Note that the `install` command installs the service using default settings. If you don't need to modify any of these, you can use the `start` command to both install and start the service. Otherwise, run `install` to register the service, then modify the configuration (the details are platform-dependent and are touched on below), and then run `start` the service.

Note that you only need to start the service once; `grb_rsm` will keep running until you execute the `grb_rsm stop` command. In particular, it will start again automatically if you restart the machine.

Note also that the `start` command does not accept any flags or additional parameters. All of the configuration properties must be set in the `grb_rsm.cnf` configuration file. If you need to make a change, edit the configuration file, then use the `stop` command followed by the `start` command to restart `grb_rsm` with the updated configuration.

The exact behavior of these commands varies depending on the host operating system and version:

2.3.7 Linux

On Linux, `grb_rsm` supports two major service managers: `systemd` and `upstart`. The `install` command will detect the service manager available on your system and will generate a service configuration file located in `/etc/systemd/system/grb_rsm.service` or `/etc/init/grb_rsm.conf` for `systemd` and `upstart`, respectively. Once the file is generated, you can edit it to set advanced properties. Please refer to the documentation of `systemd` or `upstart` to learn more about service configuration.

Use the `start` and `stop` commands to start and stop the service. When the service is running, log messages are sent to the Linux `syslog` and to a rotating log file, `grbrsm-service.log`, located in the same directory as `grb_rsm`.

The `uninstall` command will delete the generated file.

2.3.8 macOS

On macOS, the system manager is called `launchd`, and the `install` command will generate a service file in `/Library/LaunchDaemons/com.gurobi.grb_rsm.plist`. Once the file is generated, you can edit it to set advanced properties. Please refer to the `launchd` documentation to learn more about service configuration.

Use the `start` and `stop` commands to start and stop the service. When the service is running, log messages are sent to the macOS `syslog` and to a rotating log file, `grbrsm-service.log`, located in the same directory as `grb_rsm`.

The `uninstall` command will delete the generated file.

2.3.9 Windows

On Windows, the `install` command will declare the service to the operating system. If you wish to set advanced properties for the service configuration, you will need to start the `Services` configuration application. Please refer to the Windows Operating System documentation for more details.

Use the `start` and `stop` commands to start and stop the service. When the service is running, log messages are sent to the Windows event log and to a rotating log file, `grbrsm-service.log`, located in the same directory as `grb_rsm`. Note that the service must run as a user that has write permissions to this directory; otherwise, no log file will be generated.

The `uninstall` command will delete the service from the registry.

2.3.10 Verification

Once you have `grb_rsm` installed and running, your final step is to make sure that it is configured and running correctly. The Cluster Manager initially has three default users with predefined passwords:

- standard user: `gurobi` / `pass`
- administrator: `admin` / `admin`
- system administrator: `sysadmin` / `cluster`

These default accounts are provided to simplify installation; you should change the passwords or delete the accounts before actually using the cluster.

You can check that you can log in using the `sysadmin` account with the `grbcluster` command-line tool:

```
> grbcluster login --manager=http://mymanager:61080 --username=sysadmin
info  : Using client license file '/home/jones/gurobi.lic'
Password for sysadmin:
info  : User gurobi connected to http://mymanager:61080, session will expire on...
```

Note that you can specify the password by using the `--password` flag, but it is more secure to type the password when prompted. `grbcluster` will get an authentication token from the server, and will save it along with other connection parameters into a client license file. Once saved, you can use the other commands of `grbcluster` securely without having to type the password or other information again. The authentication token is valid for a certain period of time, as defined in the Cluster Manager settings (see the interactive session duration). The default is 8 hours. The `--help` flag displays help for the login command.

```
> grbcluster login --help
```

Another important validation step is to make sure you can access the Web User Interface of the Cluster Manager. To do so, just open a Web browser using the server URL:

```
http://mymanager:61080
```

This should display the login page, where you can provide the credentials for the default accounts listed above.

2.4 Installing a Cluster Node

Once the Remote Services package is installed, you will need to set up a *license*, if necessary. Then, the *Remote Services agent* must be *configured* and started as a *standard process* or as a *service*. Finally, you should *verify* your installation.

2.4.1 Licensing

You will need to download and install a license file on all Compute Server nodes (no license file is required for a Distributed Worker node). You will find detailed instructions for downloading a license in the section *How do I retrieve and set up a Gurobi license* of the Getting Started Knowledge Base article.

We will just provide a quick summary of the process here. Your first step is to locate and download your license file from the Gurobi User Portal. When you download the license file, we strongly recommend that you place it in the default location:

- `C:\gurobi\` on Windows
- `/opt/gurobi/` on Linux
- `/Library/gurobi/` on macOS
- The user's home directory

You can also set the environment variable `GRB_LICENSE_FILE` to point to this file.

In order to use the Cluster Manager, you will need to connect at least one Compute Server node to the cluster. When certain operations are requested such as submitting a job or a batch, the Cluster Manager will check the licenses available on the nodes. If none of the nodes have a valid Compute Server license, the operation will not be authorized.

2.4.2 Remote Services Agent (grb_rs)

To form a Remote Services cluster, you need to run the *Remote Services agent* (`grb_rs`) on all of the nodes that make up the cluster. These agents communicate amongst themselves, and also with the Cluster Manager or the clients.

The primary task of the Remote Services agents is to collectively manage the queueing and the execution of jobs. The agents work together to balance the load by assigning a new job to the node with the fewest running jobs whenever possible. If all nodes are at capacity, newly submitted jobs will be queued, and the first node with available capacity will later execute the job. If a new node is added to the cluster, it will immediately start processing queued jobs.

The `grb_rs` executable provides several commands and flags to help in the configuration and execution of the agent. We will review these commands step by step in the following sections. You can see the full list of commands in the *reference section* or by using the command-line help:

```
> grb_rs --help
```

2.4.3 Configuring a Cluster Node

The Remote Services agent has a number of configuration properties that affect its behavior. These can be controlled using a `grb_rs.cnf` configuration file. The installation package includes a predefined configuration file that can be used as a starting point (`<installdir>/bin/grb_rs.cnf`).

The simplest way to modify the parameters is to edit the default configuration file. Other options are available, though. The `grb_rs` process uses the following precedence rules:

- First priority: command-line flag `--config`
- Second priority: a configuration file in the current directory
- Third priority: a configuration file in a shared directory (`C:\gurobi`, `/opt/gurobi`, `/Library/gurobi` for Windows, Linux and macOS platforms, respectively)
- Fourth priority: a configuration file in the directory where `grb_rs` is located

Most of the properties that are configured through this file are related to communication options and job processing options. The configuration file is only read once, when `grb_rs` first starts. Subsequent changes to the file won't affect parameter values on a running server.

2.4.4 Configuration file format

The configuration file contains a list of properties of the form `PROPERTY=value`. Lines that begin with the # symbol are treated as comments and are ignored. Here is an example:

```
# grb_rs.cnf configuration file
PORT=61000
MANAGER=http://mymanager:61080
```

While you could create this file from scratch, we recommend you start with the version of this file that is included with the product and modify it instead.

The command `grb_rs properties` lists all of the available properties, their default values, and provides documentation for each. Some can be overridden on the command-line of `grb_rs`; the name of the command-line flag you would use to do so is provided as well. Some properties are important and must be changed for a production deployment. However, we need to distinguish between deployment with a Cluster Manager and without.

2.4.5 Important Properties with a Cluster Manager

When deploying a node with a Cluster Manager, the configuration is easier and you need to review the following properties:

MANAGER
This is the URL of the manager.

HOSTNAME
This must be the DNS name of the node that can be resolved from the other nodes or from the Cluster Manager. `grb_rs` tries to get a reasonable default value, but this value may still not be resolved by other nodes and could generate connection errors. It this case, you need to override this name in the configuration file with a fully qualified name of your node, for example:

```
HOSTNAME=server1
```

Note that you do not need to give addresses that can be resolved by clients because all communication is routed through the Cluster Manager. The nodes are never accessed directly by the clients.

CLUSTER_TOKEN
The token is a private key that enables different nodes to join the same cluster. All nodes of a cluster and the Cluster Manager must have the same token. We recommended that you generate a brand new token when you set up your cluster. The `grb_rs token` command will generate a random token, which you can copy into the configuration file.

JOBLIMIT
This property sets the maximum number of jobs that can run concurrently when using Compute Server on a specific node. The limit can be changed on a running cluster using the `grbcluster node config --job-limit <new_limit>` command, in which case the new value will persist and the value in the configuration file will be ignored from that point on (even if you stop and restart the cluster).

HARDJOBLIMIT
Certain jobs (those with priority 100) are allowed to ignore the JOBLIMIT, but they aren't allowed to ignore this limit. Client requests beyond this limit are queued, irrespective of their priority. This limit is set to 0 by default which means that it is disabled and jobs with priority 100 will not be queued.

2.4.6 Important Properties without a Cluster Manager

When installing a node that will not be connected to a Cluster Manager, authentication of clients uses predefined passwords that must be stored in the configuration file. The default configuration files must be reviewed and the following properties must be changed for a production deployment:

HOSTNAME
This must be the DNS name of the node that can be resolved from the other nodes or the clients in your network. `grb_rs` tries to get a reasonable default value, but this value may still not be resolved by clients and could generate connection errors. It this case, you need to override this name in the configuration file with a fully qualified name of your node, for example:

```
HOSTNAME=server1
```

If the names cannot be resolved by clients, another option is to use IP addresses directly, in this case set this property to the IP address of the node.

CLUSTER_TOKEN
The token is a private key that enables different nodes to join the same cluster. All nodes of a cluster must have the same token. We recommended that you generate a brand new token when you set up your cluster. The `grb_rs token` command will generate a random token, which you can copy into the configuration file.

PASSWORD
This is the password that clients must supply in order to access the cluster. It can be stored in clear text or hashed. We recommended that you create your own password, and that you store it in hashed form. You can use the `grb_rs hash` command to compute the hashed value for your chosen password. Note that clients must provide the original password (not hashed) and it will be exchanged encrypted if HTTPS is used.

```
grb_rs hash newpass
$$ppEieKZExlBR-pCSUMlmc4oWlG8nZsUOE2IM0hJbzsmV_Yjj
```

Then copy and paste the value in the configuration file:

```
PASSWORD=$$ppEieKZExlBR-pCSUMlmc4oWlG8nZsUOE2IM0hJbzsmV_Yjj
```

The default password is `pass`.

ADMINPASSWORD
This is the password that clients must supply in order to run restricted administrative job commands. It can be stored in clear text or hashed. We recommended that you create your own password, and that you store it in hashed form. You can use the `grb_rs hash` command to compute the hashed value for your chosen password. Note that clients must provide the original password (not hashed) and it will be exchanged encrypted if HTTPS is used. The default password is `admin`.

CLUSTER_ADMINPASSWORD
This is the password that clients must supply in order to run restricted administrative cluster commands. It can be stored in clear text or hashed. We recommended that you create your own password, and that you store it in hashed form. You can use the `grb_rs hash` command to compute the hashed value for your chosen password. Note that clients must provide the original password (not hashed) and it will be exchanged encrypted if HTTPS is used. The default password is `cluster`.

JOBLIMIT
This property sets the maximum number of jobs that can run concurrently when using Compute Server on a specific node. The limit can be changed on a running cluster using the `grbcluster node config --job-limit <new_limit>` command, in which case the new value will persist and the value in the configuration file will be ignored from that point on (even if you stop and restart the cluster).

HARDJOBLIMIT
Certain jobs (those with priority 100) are allowed to ignore the JOBLIMIT, but they aren't allowed to ignore this limit. Client requests beyond this limit are queued, irrespective of their priority. This limit is set to 0 by default which means that it is disabled and jobs with priority 100 will not be queued.

2.4.7 Starting a Cluster Node as a Process

Once you have installed the Remote Services package (including retrieving and installing your license file and, for Linux users, setting your `PATH` variable), starting `grb_rs` as a standard process is quite straightforward. From a terminal window with administrator privileges, simply issue the following command:

```
> grb_rs
```

If you are using a Cluster Manager and you did not set the `MANAGER` configuration property you can specify it on the command-line:

```
> grb_rs --manager=http://mymanager:61080
```

Both commands will start the Remote Services agent on the default port (port 80), and you should see output like the following:

```
info  : Reading configuration file: /home/jones/gurobi_server1300/linux64/bin/grb_rs.cnf
info  : Gurobi Remote Services starting...
info  : Platform is linux
info  : Version is 13.0.0\ (build v13.0.0rc0)
info  : Variable GRB_LICENSE_FILE is not set
info  : License file found at /home/jones/gurobi.lic
info  : Node address is server1
info  : Node FQN is server1
info  : Node has 8 cores
info  : Using data directory /home/jones/gurobi_server1300/linux64/bin/data
info  : Data store created
info  : Available runtimes: [... 13.0.0]
info  : Public root is /home/jones/gurobi_server1300/linux64/resources/grb_rs/public
info  : Starting API server (HTTP) on port 80...
```

If you do not have administrator privileges or if the default port is already in use, you will see an error about opening the port. For example, on Linux you might see an error like this:

```
fatal : Gurobi Remote Services terminated, listen tcp :80: bind: permission denied
```

or

```
fatal : Gurobi Remote Services terminated, listen tcp :80: bind: address already in use
```

Note that `grb_rs` does not have to be run with elevated privileges, but it does need elevated privileges to use the default port 80.

If you would like to run `grb_rs` on a non-default port, use the `--port` flag or set the `PORT` property in the configuration file. For example:

```
> grb_rs --manager=http://mymanager:61080 --port=61000
```

The Remote Services agent (`grb_rs`) needs a directory to store various files, including the runtimes, job metadata, job log files, etc. The default location is a directory named `data`, located in the same directory as the `grb_rs` executable (`<installdir>/bin/data`). If you have a directory named `data` in your current directory, it will use that location instead.

If starting `grb_rs` produces an error message that indicates that there was a problem creating the storage service (as shown below), a likely cause is that another `grb_rs` process is already running.

```
fatal : Error creating storage service: Error opening data store: timeout
```

2.4.8 Starting a Cluster Node as a Service

While you always have the option of running `grb_rs` from a terminal and leaving the process running in the background, we recommended that you start it as a service instead, especially in a production deployment. The advantage of a service is that it will automatically restart itself if the computer is restarted or if the process terminates unexpectedly.

`grb_rs` provides several commands that help you to set it up as a service. These must be executed with administrator privileges:

grb_rs install
: Install the service. The details of exactly what this involves depend on the host operating system type and version: this uses `systemd` or `upstart` on Linux, `launchd` on macOS, and *Windows services* on Windows.

grb_rs start
Start the service (and install it if it hasn't already been installed).

grb_rs stop
Stop the service.

grb_rs restart
Stop and then start the service.

grb_rs uninstall
Uninstall the service.

Note that the `install` command installs the service using default settings. If you don't need to modify any of these, you can use the `start` command to both install and start the service. Otherwise, run `install` to register the service, then modify the configuration (the details are platform-dependent and are touched on below), and then run `start` the service.

Note that you only need to start the service once; `grb_rs` will keep running until you execute the `grb_rs stop` command. In particular, it will start again automatically if you restart the machine.

Note also that the `start` command does not take any flags or additional parameters. All of the configuration properties must be set in the `grb_rs.cnf` configuration file. If you need to make a change, edit the configuration file, then use the `stop` command followed by the `start` command to restart `grb_rs` with the updated configuration.

The one exception is the `JOBLIMIT` property, which can be changed on a live server using `grbcluster`. If you change this property and later restart the server, the new value will persist and the value in the configuration file will be ignored.

The exact behavior of these commands varies depending on the host operating system and version:

2.4.9 Linux

On Linux, `grb_rs` supports two major service managers: `systemd` and `upstart`. The `install` command will detect the service manager available on your system and will generate a service configuration file located in `/etc/systemd/system/grb_rs.service` or `/etc/init/grb_rs.conf` for `systemd` and `upstart`, respectively. Once the file is generated, you can edit it to set advanced properties. Please refer to the documentation of `systemd` or `upstart` to learn more about service configuration.

Use the `start` and `stop` commands to start and stop the service. When the service is running, log messages are sent to the Linux `syslog` and to a rotating log file, `service.log`, located in the same directory as `grb_rs`.

The `uninstall` command will delete the generated file.

2.4.10 macOS

On macOS, the system manager is called `launchd`, and the `install` command will generate a service file in `/Library/LaunchDaemons/grb_rs.plist`. Once the file is generated, you can edit it to set advanced properties. Please refer to the `launchd` documentation to learn more about service configuration.

Use the `start` and `stop` commands to start and stop the service. When the service is running, log messages are sent to the macOS `syslog` and to a rotating log file, `service.log`, located in the same directory as `grb_rs`.

The `uninstall` command will delete the generated file.

2.4.11 Windows

On Windows, the `install` command will declare the service to the operating system. If you wish to set advanced properties for the service configuration, you will need to start the `Services` configuration application. Please refer to the Windows Operating System documentation for more details.

Use the `start` and `stop` commands to start and stop the service. When the service is running, log messages are sent to the Windows event log and to a rotating log file, `service.log`, located in the same directory as `grb_rs`. Note that the service must run as a user that has write permissions to this directory; otherwise, no log file will be generated.

The `uninstall` command will delete the service from the registry.

2.4.12 Verification

Once you have `grb_rs` running, you can check to make sure that you will be able to submit jobs to it.

2.4.13 Log In with a Cluster Manager

As we have *explained earlier*, the Cluster Manager initially creates three default users with predefined passwords:

- standard user: `gurobi` / `pass`
- administrator: `admin` / `admin`
- system administrator: `sysadmin` / `cluster`

These default accounts are provided to simplify installation; you should change the passwords or delete the accounts before actually using the cluster.

You can check that you can log in using the `sysadmin` account with the `grbcluster` command-line tool:

```
> grbcluster login --manager=http://mymanager:61080 --username=sysadmin
info  : Using client license file '/home/jones/gurobi.lic'
Password for sysadmin:
info  : User gurobi connected to http://mymanager:61080, session will expire on...
```

2.4.14 Log In without a Cluster Manager

With a self-managed cluster, there are no user accounts, and the access level is determined by the password used. Here are the default passwords (which can be changed in the *configuration file*):

- standard user: `pass`
- administrator: `admin`
- system administrator: `cluster`

In this case, you need to log in to one of the nodes and provide the system administrator password:

```
> grbcluster login --server=http://server1:61000
info  : Using client license file '/home/jones/gurobi.lic'
Enter password (return to use default):
info  : Connected to https://server1:61000
```

Note that the password you provide is stored in clear in the license file (for future use by other commands). With this in mind, make sure that access to the license file is restricted.

2.4.15 Accessing the Cluster

Once you have verified that you can log in, you should also check the list of nodes with the command:

```
> grbcluster nodes
ID       ADDRESS       STATUS TYPE    LICENSE PROCESSING #Q #R JL IDLE %MEM  %CPU
b7d037db server1:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  10 <1s  10.89 4.99
```

You are ready to submit jobs if both of the following are true:

- the `STATUS` column indicates that one or more servers are `ALIVE`
- the `LICENSE` column indicates that the license is `VALID`.

If `grbcluster` is unable to connect or if it does not show any live nodes, then check your network and the log of the `grb_rs` nodes (the console output or `<installdir>/bin/service.log` if started as a service).

If a node has an `INVALID` license, the `ERROR` field will provide more information about the error. For example:

```
> grbcluster node licenses
ID       ADDRESS       STATUS  TYPE KEY EXP ORG USER APP VER CS    DL  ERROR
b7d037db server1:61000 INVALID NODE                          false 0   No Gurobi license␣
↪found...
```

You may also want to verify that it is possible to submit a job to your cluster. To this end, you may want to identify a machine from which the users will typically submit jobs and install the gurobi client package. Then, you can submit a job with a command like the following:

```
> gurobi_cl misc07.mps
```

For more information on how to install the client and run `gurobi_cl`, please refer to the section about *using Remote Services*.

2.5 Forming a Cluster

As noted earlier, a cluster consists of a set of one or more nodes, all running `grb_rs`. This section explains how to form a cluster. Multi-node clusters provide additional capabilities relative to single-node clusters. For Compute Server, a multi-node cluster will automatically balance computational load among the various member nodes. For distributed algorithms, a multi-node cluster enables various algorithms to distribute work among multiple machines. This section begins by discussing the different *types of nodes* that are needed to support both Compute Server and distributed algorithms. Next, we will explain the *grouping* feature that can be used to create subsets of nodes to process some jobs. Finally, we will discuss the dynamic nature of a cluster. The system administrator can ask individual nodes to *start or stop processing jobs*, which makes it possible to smoothly add or remove nodes from a cluster to simplify maintenance or to scale processing capacity up or down.

2.5.1 Connecting Nodes

Every Remote Services cluster starts with a single node. The steps for starting Remote Services on a single node, either as a *standard process* or as a *service*, were covered in earlier sections.

Before adding nodes into your cluster, you first need to make sure that the cluster token (property CLUSTER_TOKEN in the configuration file) has the same value in each node and in the Cluster Manager. For better security, we recommend that you change the predefined value of the token by generating a new one and pasting the same value into each node configuration file. You can generate a new token with the following command:

```
> grb_rs token
GRBTK-6o4xujs59WJO5508nmaNwc1TtjZJAL1UcwN4vTD4qK4nata8oLr9GnubyXrLTkggc/aw2A==
```

2.5.2 Adding nodes with a Cluster Manager

If you have started a Cluster Manager, you add additional nodes using the exact same command you used to add the first node. You do this by providing the Cluster Manager address. The Cluster Manager acts as a registry of nodes of your cluster, and the nodes will then connect between themselves.

```
> grb_rs --manager=http://mymanager:61080 --port=61000
```

The MANAGER property can also be set through the configuration file:

```
MANAGER=http://mymanager:61080
PORT=61000
```

You won't have the opportunity to provide command-line options when starting grb_rs as a service, so your only option in this case is to provide this information through the configuration file.

If you wish to start multiple grb_rs processes on the same machine for testing purposes (this is not recommended for production use), you will need to make sure each instance of grb_rs is started on a different port and using a different data directory. The command grb_rs init will help you by copying the default configuration and the data directory into a current directory.

For example, to start two nodes on the same machine with a hostname of myserver:

1. In a first terminal window, create a new directory node1,
2. Change your current directory to node1 and run grb_rs init
3. Start the first node:

   ```
   grb_rs --manager=http://mymanager:61080 --port=61000
   ```

4. In a second terminal window, create a new directory node2,
5. Change your current directory to node2 and run grb_rs init
6. Start the second node on a different port and connect to the Cluster Manager:

   ```
   grb_rs --manager=http://mymanager:61080 --port=61001
   ```

2.5.3 Adding nodes to a Self-Managed Cluster

If you have not started a Cluster Manager, nodes must be connected to each other. Once you've started a single-node cluster, you can add nodes using the `--join` flag to `grb_rs` or the `JOIN` configuration property. For example, if you've already started a cluster on the default port of `server1`, you would run the following command on the new node (call it `server2`) to create a two-node cluster:

```
> grb_rs --join=server1
```

In the log output for `server2`, you should see the result of the handshake between the servers:

```
info  : Node server1, transition from JOINING to ALIVE
```

Similarly, the log output of `server1` will include the line:

```
info  : Node server2, added to the cluster
```

If you are using a non-default port, you can specify the target node port as part of the node URL in the `--join` flag. You can specify the port of the current node using the `--port` flag. You can use different ports on different machines, but it is a good practice to use the same one (port 61000 is typically a good choice). The command would look like this:

```
> grb_rs --join=server1:61000 --port=61000
```

The `JOIN` property can also be set through the configuration file:

```
JOIN=server1:61000
PORT=61000
```

Again, you won't have the opportunity to provide command-line options when starting `grb_rs` as a service, so your only option in this case is to provide this information through the configuration file.

Once you've created a multi-node cluster, you can add additional nodes by doing a `JOIN` with any member node. Furthermore, the `--join` flag or the `JOIN` property can take a comma-separated list of node names, so a node can still join a cluster even if one of the member nodes is unavailable. Note that when a list of nodes is specified, the joining node will try to join with all of the specified nodes at the same time. Joining is an asynchronous process, so if some target nodes are not reachable, the joining node will retry before giving up on joining. If all of the nodes are reachable, they will all join and form a single cluster.

If you wish to start multiple `grb_rs` processes on the same machine for testing purposes (this is not recommended for production use), you will need to make sure each instance of `grb_rs` is started on a different port and using a different data directory. The command `grb_rs init` will help you by copying the default configuration and the data directory into a current directory.

For example, to start two nodes on the same machine with a hostname of `myserver`:

1. In a first terminal window, create a new directory `node1`,
2. Change your current directory to `node1` and run `grb_rs init`
3. Start the first node:

   ```
   grb_rs --port=61000
   ```

4. In a second terminal window, create a new directory `node2`,
5. Change your current directory to `node2` and run `grb_rs init`
6. Start the second node on a different port and join the first node:

```
grb_rs --port=61001 --join=myserver:61000
```

2.5.4 Checking the status of your cluster

You can use `grbcluster` to check the status of the cluster:

```
> grbcluster nodes
ID       ADDRESS       STATUS TYPE    LICENSE PROCESSING #Q #R JL IDLE %MEM  %CPU
b7d037db server1:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  10 <1s  61.42 9.72
eb07fe16 server2:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  8  <1s  61.42 8.82
```

The nodes of the cluster constantly share information about their status. Each node can be in one of the following states:

ALIVE
: The node is up and running.

DEGRADED
: The node failed to respond to recent communications. The node could return to the `ALIVE` state if it becomes reachable again. The node will stay in this state until a timeout (controlled by the configuration property `DEGRADED_TIMEOUT`), at which point it is considered as `FAILED`

FAILED
: The node has been in `DEGRADED` state for too long, and has been flagged as `FAILED`. A node will remain in the `FAILED` state for a short time, and it will eventually be removed from the cluster. If the node comes back online, it will not re-join the cluster automatically.

JOINING
: The node is in the process of joining the cluster.

LEAVING
: The node left the cluster. It will stay in that state for a short time period before being removed from the cluster.

2.5.5 Compute Servers and Distributed Workers

A Remote Services cluster is a collection of nodes of two different types:

COMPUTE
: A Compute Server node supports the offloading of optimization jobs. Features include load balancing, queueing and concurrent execution of jobs. A Compute Server license is required on the node. A Compute Server node can also act as a Distributed Worker.

WORKER
: A Distributed Worker node can be used to execute part of a distributed algorithm. A license is not necessary to run a Distributed Worker, because it is always used in conjunction with a manager (another node or a client program) that requires a license. A Distributed Worker node can only be used by one manager at a time (i.e., the job limit is always set to 1).

By default, `grb_rs` will try to start a node in Compute Server mode and the node license status will be `INVALID` if no license is found. In order to start a Distributed Worker, you need to set the `WORKER` property in the `grb_rs.cnf` configuration file (or the `--worker` command-line flag):

```
WORKER=true
```

Once you form your cluster, the node type will be displayed in the `TYPE` column of the output of `grbcluster nodes`:

```
> grbcluster nodes
ID       ADDRESS       STATUS TYPE    LICENSE PROCESSING #Q #R JL IDLE %MEM  %CPU
b7d037db server1:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  10 19m  15.30 5.64
735c595f server2:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  10 19m  10.45 8.01
eb07fe16 server3:61000 ALIVE  WORKER  VALID   ACCEPTING  0  0  1  <1s  11.44 2.33
4f14a532 server4:61000 ALIVE  WORKER  VALID   ACCEPTING  0  0  1  <1s  12.20 5.60
```

The node type cannot be changed once `grb_rs` has started. If you wish to change the node type, you need to stop the node, change the configuration, and restart the node. You may have to update your license as well.

2.5.6 Distributed Optimization

When using distributed optimization, distributed workers are controlled by a manager. There are two ways to set up the manager:

- The manager can be a job running on a Compute Server. In this case, a job is submitted to the cluster and executes on one of the `COMPUTE` nodes as usual. When the job reaches the point where distributed optimization is requested, it will also request some number of workers (see parameters DistributedMIPJobs, ConcurrentJobs, or TuneJobs). The first choice will be `WORKER` nodes. If not enough are available, it will use `COMPUTE` nodes. The workload associated with managing the distributed algorithm is quite light, so the initial job will act as both the manager and the first worker.
- The manager can be the client program itself. The manager does not participate in the distributed optimization. It simply coordinates the efforts of the distributed workers. The manager will request distributed workers (using the WorkerPool parameter), and the cluster will first select the `WORKER` nodes. If not enough are available, it will use `COMPUTE` nodes as well.

In both cases, the machine where the manager runs must be licensed to run distributed algorithms (you should see a `DISTRIBUTED=` line in your license file).

It is typically better to use the Compute Server itself as the distributed manager, rather than the client machine. This is particularly true if the Compute Server and the workers are physically close to each other, but physically distant from the client machine. In a typical environment, the client machine will offload the Gurobi computations onto the Compute Server, and the Compute Server will then act as the manager for the distributed computation.

2.5.7 Grouping

With the Remote Services grouping feature, you can define a subset of the nodes in your cluster as a group, and then submit jobs specifically to that group. This can be quite useful when some nodes in the cluster are different from others. For example, some nodes may have more memory or faster CPUs. Using this feature, you can force jobs to only run on the appropriate type of machines. If all nodes of the requested group are at capacity, jobs will be queued until a member of that group is available.

In order to define a group, you will need to add the `GROUP` property to the `grb_rs.cnf` configuration file and give a name to the group:

```
GROUP=group1
```

The groups are static and can only be changed in the node configuration file. If you wish to change the group of a node, you will need to stop the node, edit the configuration file, and restart the node. A node can only be a member of one group.

The `grbcluster nodes` command displays the assigned group for each node (in the `GRP` column):

```
> grbcluster nodes
ID       ADDRESS       STATUS TYPE    GRP     LICENSE PROCESSING #Q #R JL IDLE %MEM  %CPU
b7d037db server1:61000 ALIVE  COMPUTE group1  VALID   ACCEPTING  0  0  10 19m  15.30 5.64
735c595f server2:61000 ALIVE  COMPUTE group1  VALID   ACCEPTING  0  0  10 19m  10.45 8.01
eb07fe16 server3:61000 ALIVE  WORKER  group2  VALID   ACCEPTING  0  0  1  <1s  11.44 2.33
4f14a532 server4:61000 ALIVE  WORKER  group2  VALID   ACCEPTING  0  0  1  <1s  12.20 5.60
```

You can submit an optimization job to a given group by using the GROUP property of the client license file (see *set up a client license*). You can also set the CSGROUP parameter in the programming interface.

The value of this parameter can be a single group to target a subset of nodes as explained. It can also be a list of groups, and you can also specify a priority for each group. Here is an example to submit a job to the group1 nodes with priority 10, and to group2 with priority 50.

```
group1:10,group2:50
```

Note that if a group is not specified for a submitted job, the job can run on any nodes of any group.

2.5.8 Processing State and Scaling

Each node of the cluster can be in one of three processing states:

ACCEPTING
: The node is accepting new jobs.

DRAINING
: The node is not accepting new jobs, but it is still processing existing jobs.

STOPPED
: The node is not accepting new jobs and no jobs are running.

A node always starts in the ACCEPTING state. If you need to perform maintenance on a node, or if you want the node to leave the cluster in a clean state for other reasons, the system administrator can issue the `node stop` command:

```
> grbcluster node stop --server=server1:61000
```

If jobs are currently running, the state will change to DRAINING until the jobs finish, at which point it will change to STOPPED. If no jobs are running, the state will change to STOPPED immediately. In the DRAINING or STOPPED states, new jobs are rejected on that node. However, the node is still a member of the cluster and all of the other features, commands, and APIs are still active.

Once a node is in the STOPPED state, you can safely remove it from the cluster (to perform maintenance, shut it down, etc.). To return it to the cluster and resume job processing, run the `node start` command:

```
> grbcluster node start --server=server1:61000
```

The flag `--server` is used to target a specific node in the cluster. Adding the `--all` flag requests that the command (e.g., `start` or `stop`) be applied to all nodes in the cluster.

By using the `start` and `stop` with a cluster of Compute Servers, you can effectively scale your cluster up or down, depending on the current cluster workload:

- You can scale down the cluster by stopping the processing on some nodes.
- You can scale up the cluster by starting new nodes or resuming processing on some nodes. As soon as a node starts or resumes processing, it will pick up jobs from the current queue or wait for new jobs to be submitted.

2.6 Communication Options

The Cluster Manager and the nodes running Gurobi Remote Services communicate through a REST API using HTTP by default. If you are using a Cluster Manager, you have a few options for a more secure deployment with HTTPS:

- Use a load balancer listening on HTTPS in front of the Cluster Manager. The load balancer can remove TLS encryption, and forward the communication to the Cluster Manager as HTTP.
- Enable HTTPS on the Cluster Manager and then let it forward the communication to the nodes using HTTP. The nodes themselves will continue to communicate over HTTP only.
- End-to-end HTTPS by enabling HTTPS on the Cluster Manager and the nodes.

If you are not using a Cluster Manager, you still have a few options:

- Enable HTTPS for all of the nodes.
- Set up a *Gurobi Router* and enable HTTPS for the router only.
- End-to-end HTTPS by enabling HTTPS on the Router and the nodes.

Enabling HTTPS on the different components follows the same *principles*. Remote Services also support *self-signed certificates* for testing your deployment. Finally, *firewalls* may have to be configured to let clients connect to the Cluster Manager or the Cluster nodes. In some cases, a Remote Services *router* can be used instead of a Cluster Manager.

2.6.1 Enabling HTTPS

Enabling HTTPS on the Cluster Manager or the nodes follows the same principles. Several properties can be used to configure the communication options. In order to enable HTTPS with TLS data encryption over the wire, you need to set the TLS property.

```
TLS=true
```

You will also need to provide the paths to the private key and the certificate files:

```
TLS_CERT=cert.pem
TLS_KEY=key.pem
```

When HTTPS is enabled on the cluster nodes, the standard HTTPS port 443 is then used as the default instead of port 80. As with the port 80, you will need to start `grb_rs` with elevated privileges. Otherwise, you will get a permission error. On Linux, you'd see an error message like the following:

```
fatal : Gurobi Remote Services terminated, listen tcp :443: bind: permission denied
```

As explained in the installation section, you can change the port using the PORT property. Note that you cannot mix nodes using HTTP and nodes using HTTPS in the same cluster. If you wish to use HTTPS, all of the nodes must be configured in the same way. HTTPS will be used for communication between the nodes.

If you enable HTTPS, you will need to use the prefix `https://` to access the nodes of your cluster:

```
> grbcluster nodes
ADDRESS               STATUS TYPE    LICENSE PROCESSING #Q #R JL IDLE   %MEM  %CPU
https://server1:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  2  46h59m 9.79  0.50
https://server2:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  2  46h46m 8.75  0.00
```

2.6.2 Using HTTPS with Self-Signed Certificates

Using self-signed certificates is not recommended for production deployment as it is less secure, but it can be useful when testing a deployment. When using this mode, the data will be encrypted over the wire, but the certificate will not be validated.

If you do not specify a key and a certificate in the TLS_KEY and TLS_CERT properties, grb_rs and grb_rsm will generate them for you at startup. You can also specify your own self-signed certificate using TLS_KEY and TLS_CERT properties.

To use a self-signed certificate, you will need to activate insecure mode by setting the following property for grb_rs:

```
TLS_INSECURE=true
MANAGER_INSECURE=true
```

At the same time, similar properties must be set for grb_rsm:

```
TLS_INSECURE=true
```

When using grbcluster, you will also need to activate this mode by using the --tls-insecure flag with the login command.

```
> grbcluster login --manager=https://mymanager:61080 --username=sysadmin --tls-insecure
info  : Using client license file '/Users/jones/gurobi.lic'
Password for sysadmin:
info  : User gurobi connected to https://mymanager:61080, session will expire on...
```

When using other clients such as gurobi_cl, grbtune, you can set the GRB_TLS_INSECURE environment variable. In the programming language APIs, there is also a CSTLSInsecure parameter.

2.6.3 Firewalls

When a Cluster Manager is used, clients communicate with the Cluster Manager only. The Cluster Manager then communicates with the cluster nodes. If there is a firewall between the clients and the Cluster Manager, the port used by the Cluster Manager will have to be open. The default port is 61080 but you can choose an arbitrary port through the PORT configuration property.

In a self-managed cluster, clients communicate directly with the nodes. They use port 80 for HTTP or 443 for HTTPS by default, but you can choose an arbitrary port through the PORT configuration property. If there is a firewall between the clients and the nodes of the cluster, the chosen port will have to be open. In this case, another option is to set up a *Gurobi Router*.

The command-line tools and the libraries are also compatible with standard proxy settings using environment variables HTTP_PROXY and HTTPS_PROXY. HTTPS_PROXY takes precedence over HTTP_PROXY for https requests. The values may be either a complete URL or a host[:port], in which case the http scheme is assumed.

If you face connectivity issues with firewalls or proxy servers, we suggest you share this section with your network administrator.

2.6.4 Using a Router without a Cluster Manager

If you are installing a self-managed cluster, the clients need to have direct access to each node in the cluster, including the node DNS name and IP address. A Remote Services Router provides a point of contact for all clients and will route the communication to the appropriate node in the cluster, thus allowing you to isolate your cluster from its clients. A Remote Services Router acts as a reverse proxy. Behind a router, the cluster nodes can use private DNS names or IP addresses as long as all of the nodes and the router can communicate together. Only the router must be accessible from the clients.

The router can use either HTTP or HTTPS to communicate with clients, and similarly it can choose either protocol to route traffic to cluster nodes. It is a common to enable HTTPS between the clients and the router, while having the router and the nodes communicate over unencrypted HTTP in a private network. Using this setup only requires you to manage certificates on the router.

You can get more information about the router (`grb_rsr`) by reading the command-line help:

```
grb_rsr --help
```

The router uses a configuration file `grb_rsr.cnf` that must be placed in the same directory as the `grb_rsr` executable. A predefined configuration file with additional comments is provided. The following command lists the available configuration properties:

```
grb_rsr properties
```

Similarly to `grb_rs`, the router can be started as a service and log messages will be stored in the `grbrsr-service.log` rotating file by default. Log messages will also be sent to the `syslog` on macOS and Linux, and to the service event log on Windows.

```
grb_rsr start
```

Here are some examples of how you might refer to a router using a URL (using HTTP or HTTPS, with the standard port or a custom port):

```
http://router.mycompany.com
http://router.mycompany.com:61001
https://router.mycompany.com
https://router.mycompany.com:61001
```

When using the command-line tools `grbcluster` or `gurobi_cl`, you can first log in with a router. The router address will be saved in your license file in the `ROUTER` property so that you can run other commands without needing to specify it again:

```
> grbcluster login --server=http://server1:61000 --router=http://router.mycompany.com
info  : Using client license file '/home/jones/gurobi.lic'
Enter password (return to use default):
info  : Connected to node http://server1:61000 via router http://router.mycompany.com

> grbcluster nodes
ID       ADDRESS               STATUS TYPE    LICENSE PROCESSING #Q #R JL IDLE %MEM  %CPU
b7d037db https://server1:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  10 <1s  66.58 7.97
eb07fe16 https://server2:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  1  <1s  66.58 9.62
```

For the clients using the Gurobi Optimizer API, you will need to either set the `ROUTER` property in the license file or construct an empty environment and set the CSRouter parameter before starting the environment.

For clients using the cluster REST API for monitoring purpose, you will need to use the router URL instead of a node address, and you can pass the selected node address in the header `X-GUROBI-SERVER`. In this way, the client

communicates with the router and the router will use the header value to forward the request to the selected node. In case the node address is incorrect or does not exist, the router will return the HTTP error code 502.

2.7 Upgrading an Existing Installation

If you wish to upgrade to a new Gurobi release, you can of course shut down the cluster, upgrade the Compute Server nodes and the Cluster Manager instances, and restart all of the components. However, if you have set up a scalable architecture with a load balancer in front of the Cluster Manager instances, you can do a rolling upgrade to minimize the impact to the existing users. In this case, we recommend the following process:

1. First, upgrade the Cluster Manager. New Cluster Manager instances can be added to the cluster and old ones can be progressively terminated.
2. Then, upgrade the Compute Server nodes, one at a time. To do so, use the Cluster Manager to stop a node so that new jobs will not be scheduled but existing jobs will continue until completion. Once the node has fully stopped, upgrade the node and add it back to the cluster. This can be repeated for each node.

We also recommend to test this process in a QA environment to understand the different steps and how they apply to your specific environment.

In all cases, when installing the Cluster Manager on an existing deployment, the database schema may need to be upgraded. Please check the release notes to locate the schema for the version you are upgrading to.

For a minor upgrade, the database will be automatically migrated when starting the Cluster Manager. During that process, new fields will be added and indexes will be dropped and recreated. No data will be deleted, and migrating is a safe process. However, a backup of the data is still recommended before the upgrade is started.

For a major upgrade, please carefully follow the directions in the release notes. The upgrade will not be triggered automatically, and you will be advised to take a backup of your database first.

In all cases, you can display the schema information and trigger the upgrade manually with the `grb_rsm database` command. For example, if you were running version 9.1.2, and you are upgrading to 9.5.0, this command will display the following:

```
> grb_rsm database
info  : Current schema version is 1.0.0
info  : Latest schema version is 1.1.0
info  : Please consider upgrading the database schema with 'grb_rsm database --upgrade'␣
↪command
```

If you want to proceed with the upgrade at this time, use the `--upgrade` flag:

```
> grb_rsm database --upgrade
info  : Starting database schema upgrade
info  : Database schema, replacing 6 indexes
info  : Database schema upgraded from version 1.0.0 to version 1.1.0
info  : Database upgraded
```

2.8 Thread-based Load Balancing

In Gurobi version 12, a new load balancing method was introduced for clusters with multiple nodes, providing finer control over job allocation.

Prior to version 12, the load balancing was done solely based on the maximum number of concurrent jobs. The compute server would prefer running a job where the least number of jobs were already running, and would queue jobs above the maximum number of concurrent jobs, the `JOBLIMIT`. With this approach, a node could still become overloaded, if the sum of threads used by all jobs was above the number of physical cores. This new load balancing approach looks at the requirements of each job at a lower granularity using threads.

Administrators can now specify the thread limit for each node, which defines the maximum number of threads that can be reserved by all concurrent running jobs using the `NODE_THREADLIMIT` configuration parameter in `grb_rs.cnf`. This ensures that the maximum number of threads reserved for concurrent jobs remains within the physical capacity of each node. It is also important to specify the node thread limit in all nodes of the cluster to avoid inconsistency in the load balancing process. Finally, while we generally recommend to use the same hardware for all nodes in the cluster to get consistent results, if the hardware is different, the node thread limit can be set in each node as necessary.

Submitted jobs can then specify the maximum number of threads they are allowed to use, referred to as thread reservation. The actual number of threads used by a job may vary depending on the phase or algorithm selected to solve. The thread reservation can be set using the `ThreadLimit` Gurobi environment parameter or the `--thread-limit` command-line flag when submitting a batch with `grbcluster`. If the thread reservation is not specified, or if client uses a version prior to 12.0, the thread reservation will default to the node's thread limit divided by the number of allowed concurrent jobs, with a minimum of 1 thread. For example, if the selected configuration allows a maximum of 16 threads per node and the `JOBLIMIT` is set as the default (2), then for every job that does not have a thread reservation, the default will be 8.

Once a job is submitted, the nodes will collaborate to find an appropriate placement where there is enough threads available. If several nodes are possible, the system will prefer the nodes with the most available threads, and the least number of running jobs (in this order). Note that if a job is submitted with a thread reservation exceeding the thread limit of all nodes, it will be rejected. Otherwise, the job will be queued until a node with enough available threads is found. The load-balancing algorithm is greedy and will select the first job it can run.

CHAPTER

THREE

USING REMOTE SERVICES

You can access and use your cluster from the command-line tools, from the Web User Interface of the Cluster Manager, or from any of our programming language APIs. With only a few exceptions, the feature was designed to be transparent to both the developers and the users of programs that use it.

In this section, we will review the common *client configuration* properties, and then we will explain the most important commands that you can run from the command line: *job commands*, *batch commands*, *repository commands*, and *node commands*. Some commands may be restricted to the administrator role, and others may be supported only if the Cluster Manager was installed. For a complete list of commands, please refer to the command line tool help `grbcluster --help`. Finally, we will describe how the commands can be applied to execute *distributed algorithms*.

We will discuss later how to *program with Remote Services*.

3.1 Client Configuration

In this section, we assume that you already have installed the Gurobi Optimizer package on your client machine. After this, you will need to understand the role of the *client license file* and how `grbcluster` can help in generating it. Finally, we will review the *load balancing and priority management* that can be controlled with some of the configuration properties.

3.1.1 Client License File

A client program needs to be told how to reach a Remote Services cluster. There are generally two ways to do this. The first is through the programming language APIs. We'll discuss this option in a later section on *programming with Remote Services*. The second is through a license file. You can create a client license file yourself or edit an existing one, using your favorite text editor (`Notepad` is a good choice on Windows). The license file should be named `gurobi.lic`.

The license file contains a list of properties of the form `PROPERTY=value`. Lines that begin with the # symbol are treated as comments and are ignored. The license file must be placed in one of the following locations:

- `C:\gurobi\` on Windows
- `/opt/gurobi/` on Linux
- `/Library/gurobi/` on macOS
- The user's home directory

You can also set the environment variable `GRB_LICENSE_FILE` to point to this file.

3.1.2 Connecting to a Cluster Manager

Here are the properties you can set to connect to a Cluster Manager:

CSMANAGER
The URL of the Cluster Manager, including the protocol scheme and port. For example, use `http://mymanager:61080` to access a Cluster Manager using HTTP on port 61080, or `https://mymanager:61443` to access a cluster over HTTPS on port 61443.

CSAPIACCESSID
A unique identifier used to authenticate an application on a cluster.

CSAPISECRET
The secret password associated with an API access ID.

USERNAME
The username to access the cluster.

PASSWORD
The client password to access the cluster.

CSAUTHTOKEN
Used internally to store the JWT authentication token.

These don't all need to be set - you just set the properties that are relevant for the authentication method you are using. If the license file specifies several authentication methods to a Cluster Manager, the following precedence order applies:

- API key defined with `CSAPIACCESSID` and `CSAPISECRET`
- JWT authentication token with `CSAUTHTOKEN`
- Username and password with `USERNAME` and `PASSWORD`

3.1.3 Connecting to a Cluster Node

Here are the properties you can set to connect to a cluster node in a self-managed cluster:

COMPUTESERVER
The fully qualified name of the main node used to access the cluster, plus the protocol scheme and port (if needed). For example, you can just use `server1` to access a cluster using HTTP on the default port, or `https://server1:61000` to access a cluster over HTTPS using port 61000. You can also specify a comma-separated list of names so that other nodes can be used in case the first node can't be reached.

ROUTER
The router URL (if you are using a router).

PASSWORD
The client password to access the cluster. Note that clients must provide the original password (not hashed) and it will be exchanged encrypted if HTTPS is used.

3.1.4 Other Properties

You can also specify additional properties that affect job processing (whether you use a Cluster Manager or not):

CSAPPNAME
Application name. Once defined, the application name will be assigned to all jobs and batches created so that you can better track the activity of the cluster by application.

PRIORITY
Job Priority. Higher priority jobs take precedence over lower priority jobs.

GROUP
Job group. If your cluster has been set up with groups, you can specify the group to submit the job to. The job will only be executed on nodes that are members of this group if specified. The value of this property can also be a list of groups, and you can also specify a priority for each group. For example: `group1:10,group2:50`

QUEUETIMEOUT
Queuing timeout (in seconds). A job that has been sitting in the queue for longer than the specified `QUEUETIMEOUT` value will return with a `JOB_REJECTED` error.

IDLETIMEOUT
Idle job timeout (in seconds). This property allows you to set a limit on how long a Compute Server job can sit idle before the server kills the job.

3.1.5 Examples

Here is an example of a client license file that would allow a client to connect to a Cluster Manager with an API key, and submit all the jobs under a specific application name:

```
CSMANAGER=http://mymanager:61080
CSAPIACCESSID=0e8c35d5-ff20-4e5d-a639-10105e56b264
CSAPISECRET=d588f010-ad47-4310-933e-1902057661c9
CSAPPNAME=app1
```

Here is another example that would allow you to connect a self-managed Compute Server with a specific password, and submit all the jobs with priority 10:

```
COMPUTESERVER=http://server1:61000
PASSWORD=abcd
PRIORITY=10
```

The `gurobi_cl` or `grbcluster` tools provide command-line flags that allow you to set most of these properties. These tools will read the license file, but values specified via these command-line flags will override any values provided in the license file.

3.1.6 Generating a Client License with grbcluster

Your primary tool for issuing cluster commands is a command-line program called `grbcluster`. The format of the command-line tool is as follows (see the *reference section* for more information):

```
grbcluster --help                Display usage
grbcluster command [flags]       Execute a top-level command
grbcluster command --help        Display help about a top-level command
grbcluster group command [flags] Execute a command from a group
```

(continues on next page)

(continued from previous page)

```
grbcluster group command --help  Display help about a command
                                 from a group
```

The first important command is the `login` command, which accepts various flags to allow you to configure your connection. Once your connection is validated, it will save these parameters into the license file. If the license file does not exist, it will create one. If you want to store the license file in a custom location, you can use the environment variable `GRB_LICENSE_FILE`. The command tools `grbcluster`, `gurobi_cl`, and `grbtune` will first read the client license file so that you do not need to specify connection parameters each time.

Here are some examples of the `login` command :

- Log in to a Cluster Manager with a username and password:

  ```
  grbcluster login --manager=http://mymanager:61080 --username=gurobi
  ```

  Note that if a password is necessary, you will be prompted for it. This is more secure than providing one on the command line, but that is an option too (using the `--password` flag).

- Refresh login to a Cluster Manager to extend an expired session:

  ```
  grbcluster login
  ```

- Log in to a Cluster Manager with an API key:

  ```
  grbcluster login --manager=http://mymanager:61080 --access=... --secret=...
  ```

- Log in to a Compute Server in a self-managed cluster:

  ```
  grbcluster login --server=http://server1:61000
  ```

- Log in to a Compute Server that uses a router:

  ```
  grbcluster login --server=http://server1:61001 --router=http://myrouter:61000
  ```

3.1.7 Queueing, Load Balancing, and Job Priorities

As noted earlier, Gurobi Compute Servers support job priorities. You can assign an integer priority between -100 and 100 to each job (the default is 0). When choosing among queued jobs, the Compute Server will run the highest priority job first. Note that servers will never preempt running jobs. You can set the priority in the client license file, or using the `PRIORITY` parameter in the programming language APIs.

We have chosen to give priority 100 a special meaning. A priority 100 job will start immediately, even if this means that a server will exceed its job limit. You should be cautious with priority 100 jobs, since submitting too many at once could lead to very high server loads, which could lead to poor performance and even crashes in extreme cases. Note that this feature must be enabled by the system administrator using the `HARDJOBLIMIT` configuration property.

With the Remote Services grouping feature, the system administrator may have assigned groups to the cluster nodes. This can be quite useful when some nodes in the cluster are different from others. For example, some nodes may have more memory or faster CPUs. Using this feature, you can force jobs to only run on the appropriate type of machines. If all nodes of the requested group are at capacity, jobs will be queued until a member of that group is available.

You can submit an optimization job to a given group by using the `GROUP` property of the client license file. You can also set the `CSGROUP` parameter in the programming interface.

You can use this parameter to target a single group or a list of groups, and you can specify a priority for each group. Here is an example that shows how you would use this parameter to submit a job to group1 with priority 10 and to group2 with priority 50.

```
group1:10,group2:50
```

Note that if no group specified for the submitted job, the job can run on any node.

3.2 Job Commands

In this section, we will review the most important commands to manage your jobs. We assume that the system administrator has installed the cluster and that you have successfully executed the `grbcluster login` command with the appropriate flags to access your cluster.

3.2.1 Submitting Interactive Jobs

The Gurobi command-line tool gurobi_cl can be used to submit optimization jobs. Once you have successfully executed the `grbcluster login` command, you can then submit a job using `gurobi_cl`. This is the example we used earlier in this document:

```
> gurobi_cl ResultFile=solution.sol stein9.mps
Using license file /opt/gurobi950/manager.lic
Set parameter CSManager to value "http://server1:61080"
Set parameter LogFile to value "gurobi.log"
Compute Server job ID: 1e9c304c-a5f2-4573-affa-ab924d992f7e
Capacity available on 'server1:61000' - connecting...
Established HTTP unencrypted connection

Gurobi Optimizer version 13.0.0 build v13.0.0rc0 (linux64)
Copyright (c) 2022, Gurobi Optimization, LLC
...
Gurobi Compute Server Worker version 13.0.0 build v13.0.0rc0 (linux64)
Thread count: 4 physical cores, 8 logical processors, using up to 8 threads
...
Optimal solution found (tolerance 1.00e-04)
Best objective 5.000000000000e+00, best bound 5.000000000000e+00, gap 0.0000%

Compute Server communication statistics:
  Sent: 0.002 MBytes in 9 msgs and 0.01s (0.26 MB/s)
  Received: 0.007 MBytes in 26 msgs and 0.09s (0.08 MB/s)
```

The initial log output indicates that a Compute Server job was created, that the Compute Server cluster had capacity available to run that job, and that an unencrypted HTTP connection was established with a server in that cluster. The log concludes with statistics about the communication performed between the client machine and the Compute Server. Note that the result file `solution.sol` is also retrieved.

This is an interactive optimization task because the connection with the job must be kept alive and the progress messages are displayed in real-time. Also, stopping or killing the command terminates the job.

3.2.2 Listing Jobs

The optimization jobs running on a Compute Server cluster can be listed using the `jobs` command:

```
> grbcluster jobs
JOBID    ADDRESS  STATUS  #Q  STIME               USER  PRIO API
58780a22 server1  RUNNING     2019-04-07 14:36:49 jones 0    gurobi_cl
```

The `jobs` command is actually a shortcut for the `job list` command.

```
> grbcluster job list
JOBID    ADDRESS  STATUS  #Q  STIME               USER  PRIO API
58780a22 server1  RUNNING     2019-04-07 14:36:49 jones 0    gurobi_cl
```

Note that you can get more information by using the `--long` flag. With this flag, you will also display the complete job ID, which is unique, instead of the short ID:

```
> grbcluster jobs --long
JOBID        ADDRESS  STATUS  #Q  STIME               USER  PRIO API       RUNTIME PID  ␣
↪HOST   IP
58780a22-... server1  RUNNING     2019-04-07 14:36:49 jones 0    gurobi_cl 9.5.0   20656␣
↪machine1 [::1]
```

The `jobs` command only shows jobs that are currently running. To obtain information on jobs that were processed recently, run the `job recent` command:

```
> grbcluster job recent
JOBID    ADDRESS  STATUS    STIME               USER  OPT         API
58780a22 server1  COMPLETED 2019-04-07 14:36:54 jones OPTIMAL     gurobi_cl
```

The information displayed by the `jobs` and `job recent` commands can be changed using the `--view` flag. The default view for the two commands is the `status` view. Alternatives are:

```
status   - List all jobs and their statuses
model    - List all jobs, and include information about the models solved
simplex  - List jobs that used the SIMPLEX algorithm
barrier  - List jobs that used the BARRIER algorithm
mip      - list jobs that used the MIP algorithm
```

For example, the `model` view gives details about the model, including the number of rows, columns and non-zeros in the constraint matrix:

```
> grbcluster job recent --view=model
JOBID    STATUS    STIME               ROWS COLS NONZ ALG OBJ  DURATION
58780a22 COMPLETED 2019-04-07 14:36:54 331  45   1034 MIP 30   4.901s
```

To get an explanation of the meanings of the different fields within a view, add the `--describe` flag. For example:

```
> grbcluster job recent --view=model --describe
JOBID     - Unique job ID, use --long to display full ID
STATUS    - Job status
STIME     - Job status updated time
ROWS      - Number of rows
COLS      - Number of columns
NONZ      - Number of non zero
```

(continues on next page)

(continued from previous page)

```
ALG       - Algorithm MIP, SIMPLEX or BARRIER
OBJ       - Best objective
DURATION  - Solve duration
```

For a Mixed-Integer Program (MIP), the `mip` view provides progress information for the branch-and-cut tree. For example:

```
> grbcluster job recent --view=mip
JOBID    STATUS    STIME               OBJBST OBJBND NODCNT SOLCNT CUTCNT NODLFT
58780a22 COMPLETED 2019-04-07 14:36:54 30     30     54868  4      19     0
```

Again, `--describe` explains the meanings of the different fields:

```
> grbcluster job recent --view mip --describe
JOBID     - Unique job ID, use --long to display full ID
STATUS    - Job status
STIME     - Job status updated time
OBJBST    - Current best objective
OBJBND    - Current best objective bound
NODCNT    - Current explored node count
SOLCNT    - Current count of feasible solutions found
CUTCNT    - Current count of cutting planes applied
NODLFT    - Current unexplored node count
```

Note that the `jobs` command provides live status information, so you will for example see current MIP progress information while the solve is in progress.

The other views (`simplex` and `barrier`) are similar, although of course they provide slightly different information.

3.2.3 Accessing Job Logs

Gurobi Optimizer log output from a previous or currently running job can be retrieved by using the `job log` command. For example:

```
> grbcluster job log 58780a22
info  : Found matching job is 58780a22-8acc-499e-b73c-da6f2df59669
  Flow cover: 20
  Zero half: 4

Explored 54868 nodes (381866 simplex iterations) in 4.27 seconds
Thread count was 4 (of 4 available processors)

Solution count 4: 30 31 32 33

Optimal solution found (tolerance 1.00e-04)
Best objective 3.000000000000e+01, best bound 3.000000000000e+01, gap 0.0000%
```

The argument to this command is the `JOBID` for the job of interest (which can be retrieved using the `jobs` command). You can use the full ID or the short ID. If you don't specify a `JOBID`, the command will display the log for the last job submitted.

The `job log` command accepts the following arguments:

```
Usage:
  grbcluster job log <JOBID> [flags]

Flags can be set using --flag=value or the short form -f=value if defined.
A boolean flag can be enabled using --flag or the short form -f if defined.

Flags:
  -b, --begin        Display log from the beginning
  -f, --continuous   Display log continuously until job completion
  -h, --help         help for log
  -n, --lines int    Display only the last n lines (default 10)
```

For example, to get the entire log, from the beginning of the job, use the `-b` (or `--begin`) flag.

```
> grbcluster job log 58780a22 -b
```

You can get a continuous feed of the log for a running optimization job with the `-f` (or `--follow`) flag.

3.2.4 Accessing Job Parameters

The Gurobi Optimizer provides a number of parameters that can be modified by the user. The `job params` command allows you to inspect the values of these parameters in a Compute Server job:

```
> grbcluster job params 58780a22
info  : Found matching job is 58780a22-8acc-499e-b73c-da6f2df59669
ComputeServer=server1
TimeLimit=60
```

The argument to this command is the `JOBID` for the job of interest (which can be retrieved using the `jobs` command). You can use the full ID or the short ID. If you don't specify a `JOBID`, the command will display the changed parameters of the last job submitted.

The following example illustrates how the `grbcluster job params` command can be used in practice. The first step is to start an optimization job on a Compute Server cluster with one modified parameter:

```
> gurobi_cl TimeLimit=120 glass4.mps
```

Once the job starts, you can use the `grbcluster jobs` command to retrieve the associated `JOBID` (or you can read it off from the output of `gurobi_cl`). For jobs that have been already processed, you would run the `job recent` command instead.

```
> grbcluster jobs
JOBID    ADDRESS STATUS  #Q  STIME               USER  PRIO API
7c51bf74 server1 RUNNING     2019-04-07 14:50:56 jones 0    gurobi_cl
```

Once you obtain the `JOBID`, the `job params` command shows the modified parameter settings for the job:

```
> grbcluster job params 7c51bf74
info  : Found matching job is 7c51bf74-ba02-4239-875e-c8ea388f9427
ComputeServer=server1
TimeLimit=120
```

The full list of Gurobi parameters can be found in the Parameter Reference section of the Gurobi Reference Manual.

3.2.5 Aborting Jobs

Jobs that are running on a Compute Server can be aborted by using the `job abort` command. For example:

```
> grbcluster job abort e7022667
```

The following steps illustrate how you would start and subsequently abort a job. First, use the Gurobi command-line tool (`gurobi_cl`) to start a long-running optimization job on your Compute Server:

```
> gurobi_cl glass4.mps
```

Once the job starts, you can use the `grbcluster jobs` command to retrieve the associated `JOBID` (or you can read it off from the output of `gurobi_cl`):

```
> grbcluster jobs
JOBID    ADDRESS STATUS  #Q  STIME               PRIO
8f9b15d9 server1 RUNNING     2017-10-10 17:30:33 0
```

The full or short `JOBID` can be used to abort the job as follows:

```
> grbcluster job abort 8f9b15d9
```

If no `JOBID` is specified, the most recently started job will be aborted.

After the abort command is issued, the status of the job can be retrieved using the `job recent` command:

```
> grbcluster job recent
JOBID    ADDRESS  STATUS  STIME               USER  OPT
8f9b15d9 server1  ABORTED 2017-10-10 17:41:33 user1 OPTIMAL
```

As you can see, the status of the job has changed to `ABORTED`.

3.2.6 Accessing the Job History

The job history is only supported by the Cluster Manager. While the nodes have a limited recent history of jobs, the Cluster Manager is able to record the jobs and the logs over a longer period of time. The exact lifespan is a configuration parameter that can be set by the system administrator. History information is persistent and can be accessed even if the cluster nodes are not available.

By default, the `job history` command will display the last jobs in your cluster.

```
> grbcluster job history
JOBID    BATCHID  ADDRESS       STATUS    STIME               USER   OPT     API
910878b9 4aba4ad3 server1:61000 COMPLETED 2019-09-22 15:55:24 jones  OPTIMAL grbcluster
594d82fc 9bc34333 server1:61000 ABORTED   2019-09-22 15:52:36 admin          grbcluster
ce7ab3a4 2e05810c server1:61000 COMPLETED 2019-09-22 14:20:14 jones  OPTIMAL grbcluster
66d4783b ada0a345 server1:61000 COMPLETED 2019-09-22 14:17:58 admin  OPTIMAL grbcluster
```

In addition, flags are available to query the history by status, username, time period, application name, and more. For example, the following command lists the last two jobs that were `ABORTED` by the user gurobi:

```
> grbcluster job history --status=ABORTED --length=2 --username=gurobi
JOBID    BATCHID  ADDRESS       STATUS  STIME               USER   OPT API
80334e25 5a4764cc server1:61000 ABORTED 2019-09-20 16:53:23 gurobi     Python
0d6d140b f94228b6 server1:61000 ABORTED 2019-09-16 22:04:09 gurobi     Python
```

This `history` command gives you access to the same views as the `job recent` command, but with more filters. For example, the following command shows the model characteristics of the last two jobs that were `COMPLETED` by the application `app1`:

```
> grbcluster job history --status=COMPLETED --length=2  --view=model --app=app1
JOBID    STATUS    STIME               ROWS COLS NONZ ALG OBJ DURATION
b9063f12 COMPLETED 2019-09-19 11:22:18 13   9    45   MIP 5   319ms
964a9405 COMPLETED 2019-09-19 11:22:17 13   9    45   MIP 5   126ms
```

3.3 Batch Commands

In this section, we will review the most important commands available to manage batches. We assume that the system administrator has installed the cluster and that you have successfully executed the `grbcluster login` command with the appropriate flags to access your cluster. Batch management is only available with a Cluster Manager.

3.3.1 Creating Batches

Once you are logged in to a Cluster Manager, you can use `grbcluster` to create a batch. This will submit a non-interactive job. The typical process involves the following three steps:

- Create the batch and submit the non-interactive job. In this step, we indicate what model file we want to solve, and what result file we need. With this information, `grbcluster` will declare the batch, upload the input model file, and submit the solve request as a batch job. At this point, you can disconnect your client machine from the network (i.e., close your laptop). The request will be processed automatically.

```
> grbcluster batch solve glass4.mps ResultFile=solution.sol
info  : Batch ada0a345-aa9e-4d6b-a7f0-05caf345d4e2 created
info  : Uploading glass4.mps...
info  : Batch ada0a345-aa9e-4d6b-a7f0-05caf345d4e2 submitted with job 66d4783b...
```

- Monitoring. While the batch job is being executed, you can monitor the status if you wish. You can reference the batch you submitted using its batch ID.

```
> grbcluster batch status 2e05810c-911f-47ee-b695-27e1244fefd0 --wait
info  : Batch 2e05810c-911f-47ee-b695-27e1244fefd0 status is SUBMITTED
info  : Batch 2e05810c-911f-47ee-b695-27e1244fefd0 status is SUBMITTED
info  : Batch 2e05810c-911f-47ee-b695-27e1244fefd0 status is SUBMITTED
info  : Batch 2e05810c-911f-47ee-b695-27e1244fefd0 status is SUBMITTED
info  : Batch 2e05810c-911f-47ee-b695-27e1244fefd0 status is COMPLETED
```

- Download the results. Once a batch is complete, you can download the log file and any optimization result. By default, results are stores in a directory having the same name as the batch ID. You should also delete the batch so that the Cluster Manager can delete the associated data from the database.

```
grbcluster batch download 2e05810c-911f-47ee-b695-27e1244fefd0
info  : Results will be stored in directory 2e05810c-911f-47ee-b695-27e1244fefd0
info  : Downloading solution.sol...
info  : Downloading gurobi.log...
info  : Discarding batch data
```

You can actually use `grbcluster` to perform all three steps in a single command:

```
> grbcluster batch solve ResultFile=solution.sol misc07.mps --download
info  : Batch 5d0ea600-5068-4a0b-bee0-efa26c18f35b created
info  : Uploading misc07.mps...
info  : Batch 5d0ea600-5068-4a0b-bee0-efa26c18f35b submitted with job a9700b72...
info  : Batch 5d0ea600-5068-4a0b-bee0-efa26c18f35b status is COMPLETED
info  : Results will be stored in directory 5d0ea600-5068-4a0b-bee0-efa26c18f35b
info  : Downloading solution.sol...
info  : Downloading gurobi.log...
info  : Discarding batch data
```

3.3.2 Listing Batches

Optimization jobs running on a Compute Server cluster can be listed by using the `batches` command. The `batches` command is actually a shortcut for the `batch list` command. For example:

```
> grbcluster batches
ID       JOB      CREATED STATUS    STIME   USER  PRIO API        D SIZE   INPUT     ␣
↪OUTPUT
2e05810c ce7ab3a4 2019... COMPLETED 2019... jones 0    grbcluster X 0      glass4.mps␣
↪solution.sol
ada0a345 66d4783b 2019... COMPLETED 2019... jones 0    grbcluster   288960 misc07.mps␣
↪solution.sol
```

Note that you can get more information by using the `--long` flag. With this flag, the command will also display the batch ID and the complete job ID, which is unique, instead of the short ID. To get an explanation of the meanings of the different fields, add the `--describe` flag. For example:

```
> grbcluster batches --describe
ID        - Unique batch ID, use --long to display full ID
JOB       - Unique job ID, use --long to display full ID
CREATED   - Batch created  time
Status    - Batch Status
STIME     - Batch status updated time
USER      - Client username (not displayed if empty or restricted)
APP       - Application name (not displayed if empty or restricted)
PRIO      - Batch priority
API       - API type - Python, C++, Java, .NET, Matlab, R... (not displayed if empty or␣
↪restricted)
D         - Indicate if batch data was discarded
SIZE      - Size of batch
INPUT     - List filenames of input files (not displayed if empty or restricted)
OUTPUT    - List filenames of output files (not displayed if empty or restricted)
RUNTIME   - Batch runtime version, use --long
PID       - Client process ID, use --long (not displayed if empty or restricted)
HOST      - Client hostname, use --long (not displayed if empty or restricted)
IP        - Client IP address, use --long (not displayed if empty or restricted)
APP       - Client application name, use --long (not displayed if empty or restricted)
```

3.3.3 Aborting Batches

Batches submitted to a Cluster Manager can be aborted by using the `batch abort` command. For example:

```
> grbcluster batch abort 9bc34333
```

The following steps illustrate how you would start and subsequently abort a job. First, use the Gurobi command-line tool (`gurobi_cl`) to start a long-running optimization job on your Compute Server:

```
> grbcluster batch solve glass4.mps ResultFile=solution.sol
```

Once the batch is submitted, you can use `grbcluster batches` to monitor your batches:

```
> grbcluster batches
ID       JOB      CREATED Status    STIME   USER  PRIO API        D SIZE   INPUT
4aba4ad3 910878b9 2019... SUBMITTED 2019... jones 0    grbcluster   86579  glass4.mps
```

The full or short `ID` can be used to abort the batch as follows:

```
> grbcluster batch abort 4aba4ad3
```

After the abort command is issued, the status of the batch can be retrieved using the `batches` command:

```
> grbcluster batches
ID       JOB      CREATED Status    STIME   USER  PRIO API        D SIZE   INPUT
4aba4ad3 910878b9 2019... ABORTED   2019... jones 0    grbcluster   86579  glass4.mps
```

As you can see, the status of the batch has changed to `ABORTED`. Note also that the underlying job that was created to execute the batch is also `ABORTED`:

```
JOBID    BATCHID  ADDRESS       STATUS  STIME               USER   OPT     API
910878b9 4aba4ad3 serevr1:61001 ABORTED 2019-09-22 15:55:24 jones          grbcluster
```

3.3.4 Retrying Batches

If a batch fails to execute, you can resubmit that batch. This might for example happen if the node where the batch job was running was shut down or ran out of memory. All of the input files and parameters of the batch specification are still stored by the Cluster Manager, so there is no need to upload them again. You can simply issue the `batch retry` command:

```
grbcluster batch retry edfa28f6-7abc-4af1-80a3-0b7472dcdcf0
info  : Batch edfa28f6-7abc-4af1-80a3-0b7472dcdcf0 submitted for retry with job 9c6f1b59.
↪..
```

Note that a new batch job is created to execute the batch, but the batch specification does not change and you can still use the same batch ID to monitor progress. Note that it is not possible to retry a batch if it is currently running or if the batch data was discarded.

3.3.5 Discarding Batches

A batch has a set of input files and a set of result files that are stored in the Cluster Manager database. This enables the client to submit and disconnect while the batch is processed. Also, the results can be downloaded later when the client is ready. One consequence of this is that batches can consume significant space in the database. We may need to be careful to clean up data. It is important to discard batch data when you are done with it, to free up space in the database. Note that batch metadata is small and will still remain in the batch history for monitoring purposes even after you discard the batch.

By default, when using the `grbcluster` command to download the results, the batch will be discarded automatically. You can change the default behavior by using the `--discard` flag if you may want to download the results again later:

```
> grbcluster batch solve misc07.mps ResultFile=solution.sol --download --discard=false
info  : Batch 076225d7-a1c9-462f-bfef-8e23c81d9f16 created
info  : Uploading misc07.mps...
info  : Batch 076225d7-a1c9-462f-bfef-8e23c81d9f16 submitted with job ef0861e9...
info  : Batch 076225d7-a1c9-462f-bfef-8e23c81d9f16 status is SUBMITTED
info  : Batch 076225d7-a1c9-462f-bfef-8e23c81d9f16 status is COMPLETED
info  : Results will be stored in directory 076225d7-a1c9-462f-bfef-8e23c81d9f16
info  : Downloading solution.sol...
info  : Downloading gurobi.log...
```

You can check the space used by this batch by looking in the `SIZE` column in the output of the `batches` command:

```
> grbcluster batches --batchId=076225d7
ID       JOB      CREATED  Status    STIME   USER  PRIO API        D SIZE   INPUT     ␣
↪OUTPUT
076225d7 ef0861e9 2019...  COMPLETED 2019... jones 0    grbcluster   288960 misc07.mps␣
↪solution.sol
```

In order to discard a batch manually, you can use the `batch discard` command. You can verify that the size of the batch is 0 afterwards. You will also notice that the `D` column is flagged, indicating that the batch was discarded.

```
> grbcluster batch discard 076225d7
info  : Batch 076225d7-a1c9-462f-bfef-8e23c81d9f16 discarded

> ./grbcluster batches --batchId=076225d7
ID       JOB      CREATED  Status    STIME   USER  PRIO API        D SIZE INPUT     ␣
↪OUTPUT
076225d7 ef0861e9 2019...  COMPLETED 2019... jones 0    grbcluster X 0    misc07.mps␣
↪solution.sol
```

Note that the Cluster Manager will automatically discard and delete batches when they are older than the maximum age, as specified in the cluster retention policy. Developers submitting a batch with a programming language API should call the appropriate discard function once results have been retrieved.

3.4 Repository Commands

The file repository is a feature of the Cluster Manager that allows you to store and share Gurobi files (models, solution parameters etc.). The main use is to store models so they can be reused later in different batch configurations.

In this section, we will review the most important file repository commands. We assume that the system administrator has installed the cluster and that you have successfully executed the `grbcluster login` command with the appropriate flags to access your cluster.

3.4.1 Uploading a File to the Repository

Any files supported by the Gurobi Optimizer can be uploaded and shared in the repository. For example, let's upload the `glass4.mps` model file with the following command:

```
grbcluster repo upload /Library/gurobi950/macos_universal2/examples/data/glass4.mps --
↪container=training
info  : Object 3f104672-52c8-4a53-ad79-4a01065ffefd created, upload done in container
↪'training'
```

A container is like a directory and is used to group files together so that you can retrieve them more easily later on. Once uploaded, you can check that the new file is available in the specified container with the following command:

```
grbcluster repo list --container=training
ID       CONTAINER NAME       CREATED             SIZE  USER  USERID
3f104672 training  glass4.mps 2019-09-22 21:51:10 86579 admin admin
```

3.4.2 Using a File from the Repository

You can submit batches that use files from the repository as their inputs. A file can be referenced using the full file object ID or the file object path. A file object path starts with '@' and concatenates the container and the object name with a '/' separator. For example, if the object was uploaded in the container `examples/app1` with name `model.mps`, you can reference it using:

```
@examples/app1/model.mps
```

One possible use of the file repository is to submit several batches that solve the same model using different parameters:

```
grbcluster batch solve @training/glass4.mps ResultFile=solution.sol Threads=2
info  : Batch 7fe26af5-fbec-4e23-ad58-9ab4539f98f3 created
info  : Batch 7fe26af5-fbec-4e23-ad58-9ab4539f98f3 submitted with job 4e359f99...

grbcluster batch solve @training/glass4.mps ResultFile=solution.sol Threads=5
info  : Batch d2434fc9-b3df-4be0-bf2c-4d1eac2acb5e created
info  : Batch d2434fc9-b3df-4be0-bf2c-4d1eac2acb5e submitted with job 8deeb020...
```

The model file was used in two batches, but only needed to be uploaded once.

3.4.3 Deleting a File from the Repository

You can delete a file from the repository once you are done with it, but only if no batch is referencing it. If the file is still in use, you will be notified as in the following example:

```
grbcluster repo delete @training/glass4.mps
fatal : Object is in use by [7fe26af5-fbec-4e23-ad58-9ab4539f98f3]
```

The file may still be in use because a batch is running, or because the client did not yet download the results and discard the batch. Once there are no more references, the file will be deleted and you can check the container again:

```
> grbcluster repo delete @training/glass4.mps
> grbcluster repo list --container=training
ID CONTAINER NAME CREATED             SIZE USER USERID
```

3.5 Node Commands

In this section, we will review the most important commands to monitor the cluster. We assume that the system administrator has installed the cluster and that you successfully executed the `grbcluster login` command with the appropriate flags to access your cluster.

3.5.1 Listing Cluster Nodes

The `nodes` command provides a list of nodes in the cluster, along with status information. This command is a shortcut for the `node list` command. For example:

```
> grbcluster nodes
ID       ADDRESS       STATUS TYPE    LICENSE PROCESSING #Q #R JL IDLE %MEM  %CPU
b7d037db server1:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  10 19m  15.30 5.64
735c595f server2:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  10 19m  10.45 8.01
```

Add the `--describe` flag to see an explanation of each field:

```
> grbcluster nodes --describe

ID        - Unique node ID, use --long to display full ID
ADDRESS   - Node address
STATUS    - Node status (ALIVE, FAILED, JOINING, LEAVING, DEGRADED)
TYPE      - Node type (COMPUTE: Compute Server, WORKER: Distributed Worker)
GRP       - Group name for job affinity (not displayed if empty or restricted)
LICENSE   - License status (N/A, VALID, INVALID, EXPIRED)
PROCESSING- Processing state (ACCEPTING, DRAINING, STOPPED)
#Q        - Number of jobs in queue
#R        - Number of jobs running
JL        - Job Limit (maximum number of running jobs)
IDLE      - Idle time since the last job execution (in minutes)
%MEM      - Percentage of memory currently used on the machine
%CPU      - Percentage of CPU currently used on the machine
STARTED   - Node start time, use --long
RUNTIMES  - Deployed runtime versions, use --long
VERSION   - Remote Services Agent version, use --long
```

3.5.2 Troubleshooting Connectivity Issues

You can test to see if a Remote Services node is reachable with the `node ping` command:

```
> grbcluster node ping --server=server1
Node is not reachable
```

The `node latency` command provides additional details:

```
> grbcluster node latency
ADDRESS              LATENCY    NBERR
server1      1.12813ms  0
server2      1.218103ms 0
```

This will display the latency from the client machine to each node in the cluster.

Add the `--describe` flag to see an explanation of each field:

```
> grbcluster node latency --describe
ADDRESS   - Node address
LATENCY   - latency between the local client and a node
NBERR     - Number of errors
```

3.5.3 Listing Cluster Licenses

The `node licenses` command displays license status information for each node in a cluster:

```
> grbcluster node licenses
ID       ADDRESS   STATUS TYPE     KEY   EXP ORG    USER APP VER CS    DL  ERROR
eb07fe16 server1   VALID  NODE              gurobi          8   true  0
b7d037db server2   VALID  NODE              gurobi          8   true  0
```

Add the `--describe` flag to see an explanation of each field:

```
> grbcluster node licenses --describe
ID        - Unique node ID, use --long to display full ID
ADDRESS   - Node address
STATUS    - License status (N/A, VALID, INVALID, EXPIRED)
TYPE      - License type
KEY       - License Cloud Key (not displayed if empty or restricted)
EXP       - License expiration
VER       - Maximum runtime version supported
CS        - Indicate if Compute Server features are enabled
DL        - Maximum number of workers for a distributed job (Distributed Limit)
ORG       - Assigned organization
USER      - Assigned username
APP       - Assigned application name
ERROR     - License error message
```

If a node has an `INVALID` license, you can run the following command to learn more:

```
> grbcluster node licenses
ID       ADDRESS STATUS  TYPE KEY EXP ORG USER APP VER CS    DL  ERROR
```

(continues on next page)

(continued from previous page)

```
eb07fe16 server1 INVALID NODE                                false 0   No Gurobi license found.
↪..
```

Note that the `node licenses` command can be used at any time to check the validity and attributes of licenses on all the nodes of the cluster (expiration date, distributed limit, etc.).

3.5.4 Changing the Job Limit

Each node of a Remote Services has a job limit, which indicates the maximum number of jobs that can be run simultaneously on that node. This job limit can be changed using the `grbcluster node config` command, together with the `--job-limit=` flag. For example, to change the job limit to 5:

```
> grbcluster node config --job-limit=5
```

Changes to the job limit parameter only apply to the specified node; other nodes in the cluster are unaffected. Once changed, the new value will persist, even if you stop and restart the node.

Recall that you can run the `nodes` command to view the current job limit for each node in a cluster:

```
> grbcluster nodes
ID       ADDRESS       STATUS TYPE    LICENSE PROCESSING #Q #R JL IDLE %MEM  %CPU
b7d037db server1:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  2  19m  15.30 5.64
735c595f server2:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  2  19m  10.45 8.01
```

The `JL` column shows the job limit, which is 2 for both nodes in the cluster in this example.

We can change the limit for one node:

```
> grbcluster node config --server=server1:61000 --job-limit=5
```

By rerunning the `nodes` command, we can see that the limit for `server1` has been changed to 5:

```
> grbcluster nodes
ID       ADDRESS       STATUS TYPE    LICENSE PROCESSING #Q #R JL IDLE %MEM  %CPU
b7d037db server1:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  5  19m  15.30 5.64
735c595f server2:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  2  19m  10.45 8.01
```

3.6 Distributed Algorithms

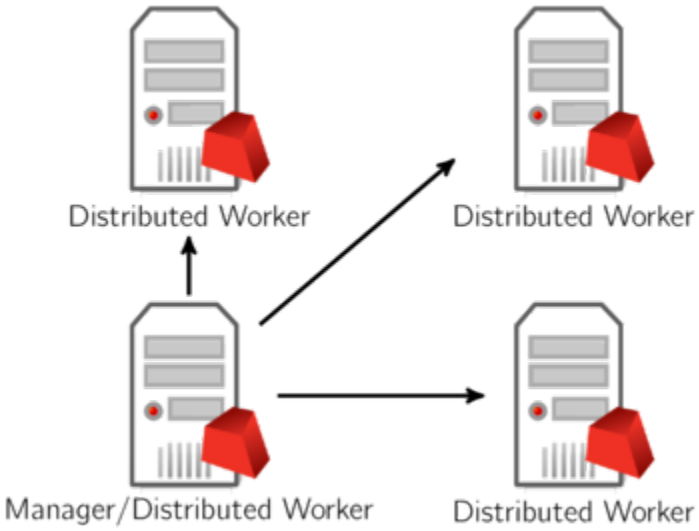
Gurobi Remote Services allow you to perform distributed optimization. All you need is a cluster with more than one node. The nodes can be either Compute Server or Distributed Worker nodes. Ideally these nodes should all give very similar performance. Identical performance is best, especially for distributed tuning, but small variations in performance won't hurt overall results too much.

3.6.1 Distributed Workers and the Distributed Manager

Running distributed algorithms requires several machines. One acts as the manager, coordinating the activities of the set of machines, and the others act as workers, receiving tasks from the manager. The manager typically acts as a worker itself, although not always. More machines generally produce better performance, although the marginal benefit of an additional machine typically falls off as you add more.

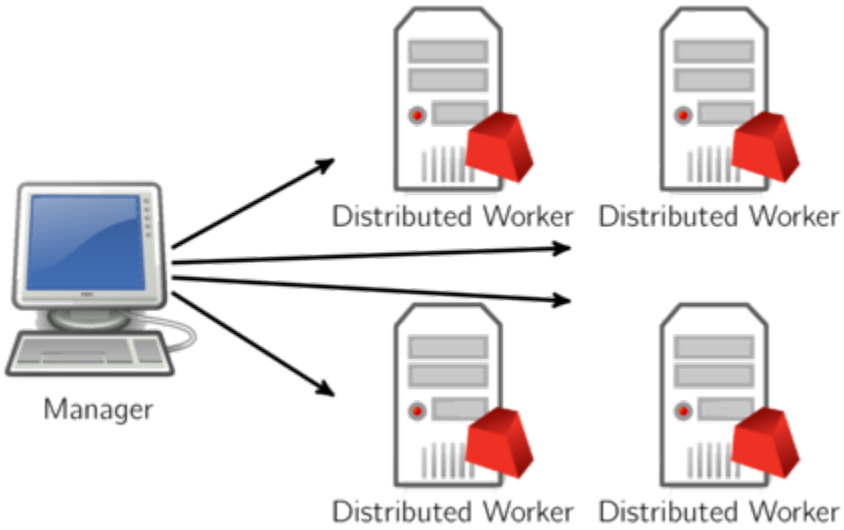
As we mentioned *earlier*, Distributed Workers do not require Gurobi licenses. You can add any machine to a Remote Services cluster to act as a Distributed Worker. The manager does require a distributed algorithm license (you'll see a `DISTRIBUTED=` line in your license file if distributed algorithms are enabled).

A typical distributed optimization will look like the following, with all machines belonging to the same Remote Services cluster:



The workload associated with managing distributed algorithms is quite light, so a machine can handle both the manager and worker roles without degrading performance.

Another option is to use a machine outside of your Remote Services cluster as the manager:



Note that we only allow a machine to act as manager for a single distributed job. If you want to run multiple distributed jobs simultaneously, you'll need multiple manager machines.

3.6.2 Configuration

Before launching a distributed optimization job, you should run the `grbcluster nodes` command to make sure the cluster contains more than one live machine:

```
> grbcluster nodes
```

If you see multiple live nodes, then that cluster is good to go:

```
ID       ADDRESS       STATUS TYPE    LICENSE PROCESSING #Q #R JL IDLE %MEM  %CPU
b7d037db server1:61000 ALIVE  COMPUTE VALID   ACCEPTING  0  0  2  1m   3.00 2.23
eb07fe16 server2:61001 ALIVE  WORKER  N/A     ACCEPTING  0  0  1  1m   2.95 5.33
```

We should reiterate a point that was raised earlier: you do not need a Gurobi license to run Gurobi Remote Services on a machine. While some services are only available with a license, any machine that is running Gurobi Remote Services will provide the Distributed Worker service.

Running a distributed algorithm is simply a matter of setting the appropriate Gurobi parameter. Gurobi supports distributed MIP, concurrent LP and MIP, and distributed tuning. These are controlled with four parameters: DistributedMIPJobs, ConcurrentJobs, and TuneJobs or TuneDynamicJobs, respectively. These parameters indicate how many distinct Distributed Worker jobs you would like to start. Keep in mind that the initial Compute Server job will act as the first worker, except for distributed tuning.

Note that jobs are allocated on a first-come, first-served basis, so if multiple users are sharing a cluster, you should be prepared for the possibility that some or all of your nodes may be busy when you request them. Your program will grab as many as it can, up to the requested count. If none are available, it will return an error.

3.6.3 Running a Distributed Algorithm as an Interactive Job

To give an example, if you have a cluster consisting of two machines (`server1` and `server2`), and if you set `DistributedMIPJobs` to 2 in `gurobi_cl` ...

```
> gurobi_cl DistributedMIPJobs=2 misc07.mps
```

... you should see the following output in the log:

```
Using Compute Server as first worker
Started distributed worker on server2:61000

Distributed MIP job count: 2
```

This output indicates that two worker jobs have been launched, one on machine `server1` and the other on machine `server2`. These two jobs will continue to run until your run completes.

3.6.4 Submitting a Distributed Algorithm as a Batch

With a Cluster Manager, you can also submit your distributed MIP and concurrent MIP as a batch using the `batch solve` command. Distributed tuning is not yet supported. Here is an example:

```
./grbcluster batch solve DistributedMIPJobs=2 misc07.mps
info  : Batch f1026bf5-d5cf-44c9-81f8-0f73764f674a created
info  : Uploading misc07.mps...
info  : Batch f1026bf5-d5cf-44c9-81f8-0f73764f674a submitted with job d71f3ceb...
```

As we can see, the model was uploaded and the batch was submitted. This creates a parent job as a proxy for the client. This job will in turn start two worker jobs because we set DistributedMIPJobs=2. This can be observed in the job history:

```
> grbcluster job history --length=3
JOBID    BATCHID  ADDRESS       STATUS    STIME               USER  OPT     API       ␣
↪PARENT
d71f3ceb f1026bf5 server1:61000 COMPLETED 2019-09-23 14:17:57 jones OPTIMAL grbcluster
6212ed73          server1:61000 COMPLETED 2019-09-23 14:17:57 jones OPTIMAL           ␣
↪d71f3ceb
63cfa00d          server2:61000 COMPLETED 2019-09-23 14:17:57 jones OPTIMAL           ␣
↪d71f3ceb
```

3.6.5 Using a Separate Distributed Manager

While Distributed Workers always need to be part of a Remote Services cluster, note that the distributed manager itself does not. Any machine that is licensed to run distributed algorithms can act as the distributed manager. You simply need to set WorkerPool and WorkerPassword parameters to point to the Remote Services cluster that contains your distributed workers. Note that the Cluster Manager can not act as the distributed manager.

To give an example:

```
> gurobi_cl WorkerPool=server1:61000 WorkerPassword=passwd DistributedMIPJobs=2 misc07.
↪mps
```

You should see the following output in the log:

```
Started distributed worker on server1:61000
Started distributed worker on server2:61000

Distributed MIP job count: 2
```

In this case, the distributed computation is managed by the machine where you launched this command, and the two distributed workers come from your Remote Services cluster.

CHAPTER

FOUR

CLUSTER MANAGER

The Cluster Manager is the central application used to control and monitor your Gurobi compute cluster. With the Cluster Manager, you can process two types of client/server optimization tasks:

- Interactive optimization tasks to create an optimization session controlled remotely.
- Non-interactive optimization tasks to submit and optimize a model automatically as specified in a batch.

4.1 Interactive Optimization

With an interactive optimization task, the client creates a session and all operations are controlled remotely. The client must stay connected until the session is released. The execution of an interactive task follows these steps:

- A client creates a Gurobi environment and triggers the allocation of an interactive job in the cluster. The client will be able to proceed only when the job has been successfully allocated and started. If the cluster is busy, the job may be added to the job queue until a slot becomes available in the cluster.
- Once the Gurobi environment is connected to the remote job, the client stays in control. The client uses the API to build the optimization model, set parameters, start the optimization, and retrieve the results. The client can also change the model and restart the optimization. It can also interrupt the optimization. All communication between the client and server happens behind the scenes.
- Finally, when the client has completed its work, it can release the environment. This will also release the job process running in the cluster. Note that, if the client has an issue or does not close the environment properly, the server will eventually detect that situation and automatically terminate the remote job.

Interactive tasks are best when you need precise control over the optimization process. It is also used by the command line tool `gurobi_cl` to execute and immediately report the engine log and get results.

4.2 Non-Interactive Optimization

With a non-interactive optimization task, the client submits a batch specification and can also immediately disconnect. It can later query the status of the execution and retrieve the solution when the task is complete. The execution of a non-interactive task follows these steps:

- A client creates a batch Gurobi environment and builds an optimization model locally. Variables in the model must be tagged (with user-specified strings), to allow the solution to be interpreted later on.
- Once the model is built and optimization is started, the model and some additional information are uploaded to the Cluster Manager. A batch ID is returned, that allows the client to refer to that batch later (to check on status, retrieve the solution, etc.).

- The Cluster Manager is then in charge of creating a batch job that will be provisioned in the cluster to execute the optimization. The job process will get the model data from the Cluster Manager. It will then perform the optimization and finally store the results back into the Cluster Manager.
- The client (or another application) can use the batch ID to monitor the progress of that batch. The client does not need to maintain a connection with the server.
- Once the batch is complete, the client can retrieve the solution. The solution is exported as a JSON document which contains solution information for all of the tagged variables.

Non-interactive tasks are useful when the client program can simply build a model and later retrieve the solution, without the need for iteration. Non-interactive tasks are also recommended when the connection between the client and the server is not stable, or if the client is likely to disconnect before the optimization run completes. A batch can also be submitted using the command line tool `grbcluster` or by using a simple drag-and-drop interface of the Cluster Manager. In this case, a batch defines all the input files and all the result files that are required.

4.3 Accounts

In order to access the Cluster Manager, clients need to be authenticated and an account must have been created. There are four user roles:

- Standard user

  A Remote Services client submits jobs or batches to the cluster. This is done through a user application or through the Gurobi command-line tool gurobi_cl or `grbcluster`, or using the Cluster Manager Web UI (drag-and-drop interface). Standard users can monitor the optimization tasks by listing jobs and batches, accessing the history, displaying the log of a job, etc.

- Administrator

  An administrator monitors and manages the flow of jobs through a Remote Services cluster. Examples of administrator commands include aborting jobs, changing cluster parameters and checking licenses.

- System administrator

  The system administrator installs and manages the Gurobi Remote Services cluster and the different components. The system administrator will need to configure and start services on one or more machine as necessary. The system administrator is also in charge of managing the user accounts. An administrator is able to abort and discard any jobs or batches.

- Read-only user

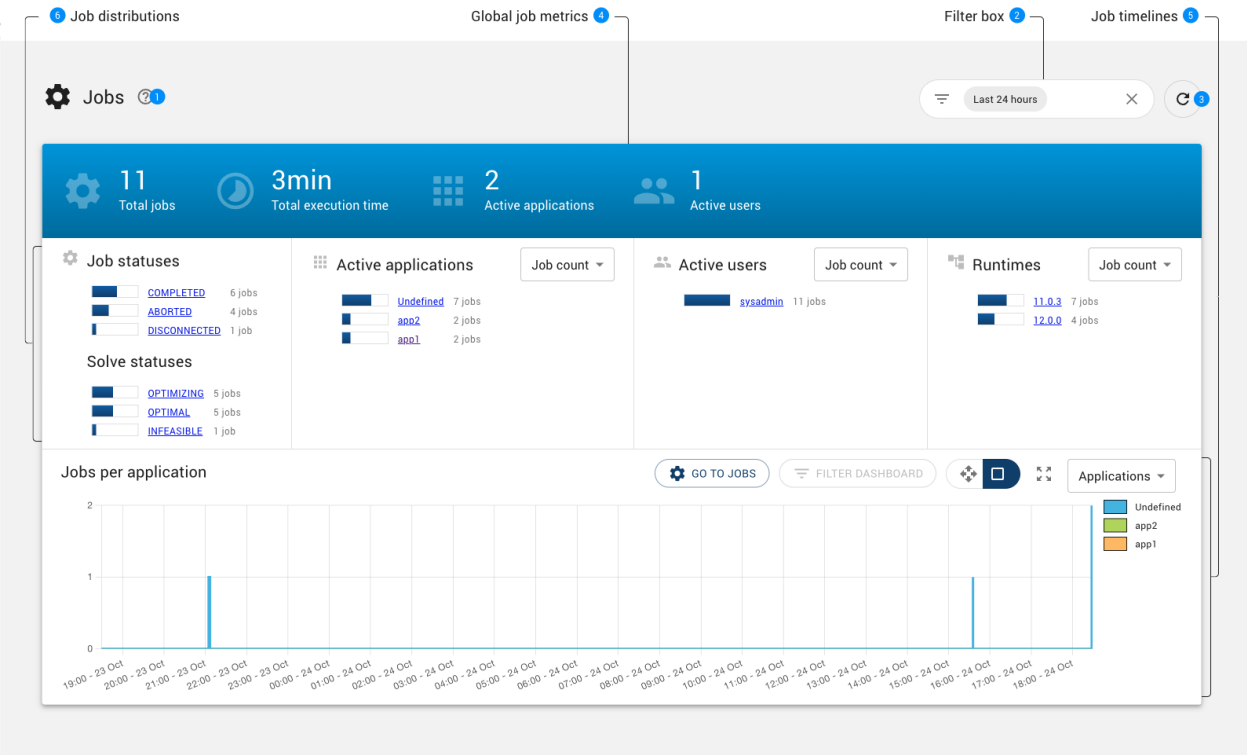
  Read-only users can only monitor the optimization tasks by listing jobs and batches, accessing the history, displaying the log of a job, etc. They are not allowed to submit jobs or batches to the cluster neither can they abort jobs of other users.

4.3.1 Dashboard

Job Dashboard

This job dashboard aggregates information about completed jobs according to several dimensions: job statuses, solves statuses, applications, users and runtime versions. It also provides a timeline for these dimensions so that users can track evolution of these metrics over time. The user can zoom or pan into the timeline, and drill down to the actual job list or filter the dashboard.

1. Help button to navigate to the documentation of the Job dashboard page.



2. Filter box to select the time range and other criteria to filter the Job dashboard page. Details can be found in the *filtering* section.
3. Button to reload the page.

   Note that reloading the page is only effective if the selected time range ends the current day.
4. Global job metrics.

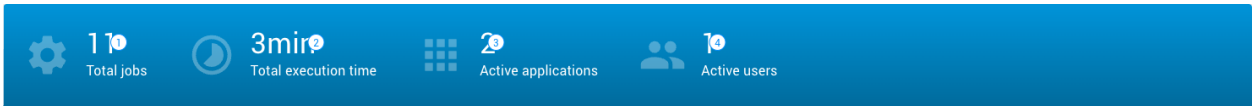
   Details can be found in the *Global job metrics* section.
5. Job timelines.

   Details can be found in the *Job timelines* section.
6. Job distribution.

   Details can be found in the *Job distribution* section.

Global job metrics

Main metrics for the job activity for the selected period of time.



1. Total jobs

   Number of jobs that were running in the selected period of time.

2. Total execution time

   Sum of the execution time of all jobs that were running in the selected period of time.

3. Active applications

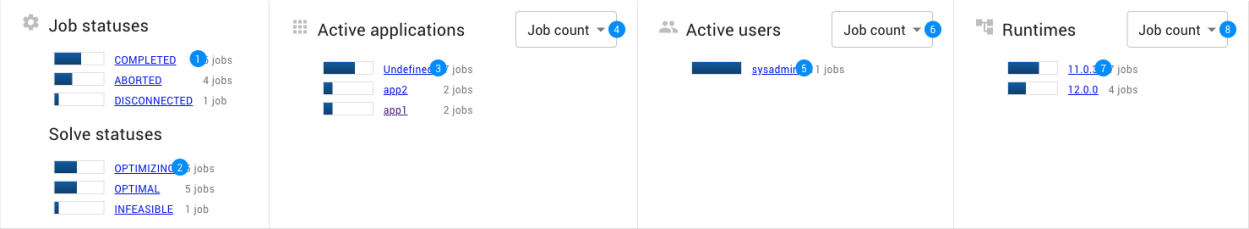
   Number of applications for the jobs that were running in the selected period of time.

4. Active users

   Number of users who have been running jobs on this cluster in the selected period of time.

Job distribution

This row displays the distributions of jobs according to several dimensions: job statuses, solves statuses, applications, users and runtime versions. For some dimensions, you can select the aggregation by the number of jobs or the total job execution time.



1. Job statuses

   Number of jobs that were running on the selected period of time per status, such as `COMPLETED` or `FAILED`.

2. Solve statuses

   Number of jobs that were running on the selected period of time per solve status, such as `OPTIMAL` or `INFEASIBLE`.

3. Active applications

   Number of jobs that were running on the selected period of time per application. The drop down **(4)** at the top right of the chart allows for displaying job execution times instead of number of jobs in the bar chart.

5. Active users

   Number of jobs that were running on the selected period of time per user. The drop down **(6)** at the top right of the chart allows for displaying job execution times instead of number of jobs in the bar chart.

7. Runtimes

   Number of jobs that were running on the selected period of time per runtime. The drop down **(8)** at the top right of the chart allows for displaying job execution times instead of number of jobs in the bar chart.

Rows of horizontal bar chart provide a link to the *Job History page* with a filter set to select the jobs corresponding to the clicked link.

For example, a click on the 

job app `link` of the **Active** applications bar chart will navigate to the *Job History page* with the application filter set to `app1` and the same time range the dashboard page, as illustrated below:



Job timelines

This row displays the selected timeline of the jobs that ran in the selected period of time.



1. Go to jobs

   This button navigates to the *Job History page*, applying the current filter in the dashboard page and providing the visible time range in the timeline chart as the time range filter for the jobs page.

2. Filter dashboard

   Applies the visible time range in the timeline chart to the filter of the dashboard page.

3. 4. 5. Zoom features

   The timeline chart provides two modes for changing the visible time range of the chart.

   - Pan, activated with the Pan **(3)** button In this mode, pressing the left mouse button on the chart and dragging the mouse left of right will scroll horizontally the timeline, unless no zoom in has been performed on the chart.
   - zoom in to a specified time range, activated with the Drag **(4)** button.

     This mode is selected by default.

     In this mode, pressing the left mouse button on the chart and dragging the mouse left of right will select a time range, visible with a blue rectangle. When releasing the left mouse button, the chart will zoom in the zone corresponding to the dragged area.

   At any time, you can restore the timeline chart to its initial zoom state clicking the restore zoom **(5)** button.

   To zoom in/zoom out with the mouse wheel, keep the `Ctrl` key pressed while using the mouse wheel.

6. Timeline selection

   The drop down **(6)** at the top right of the chart allows for selecting the dimension of timelines:

   - Job applications
   - Job statuses
   - Job solve statuses
   - Job users
   - Job runtimes
   - Job solve times

The last choice of the timeline drop down selects a Solve times chart. Each bar of this chart is associated with one job that was running in the selected period of time. The height of the job bar indicates the execution time of the job.

7. Legend

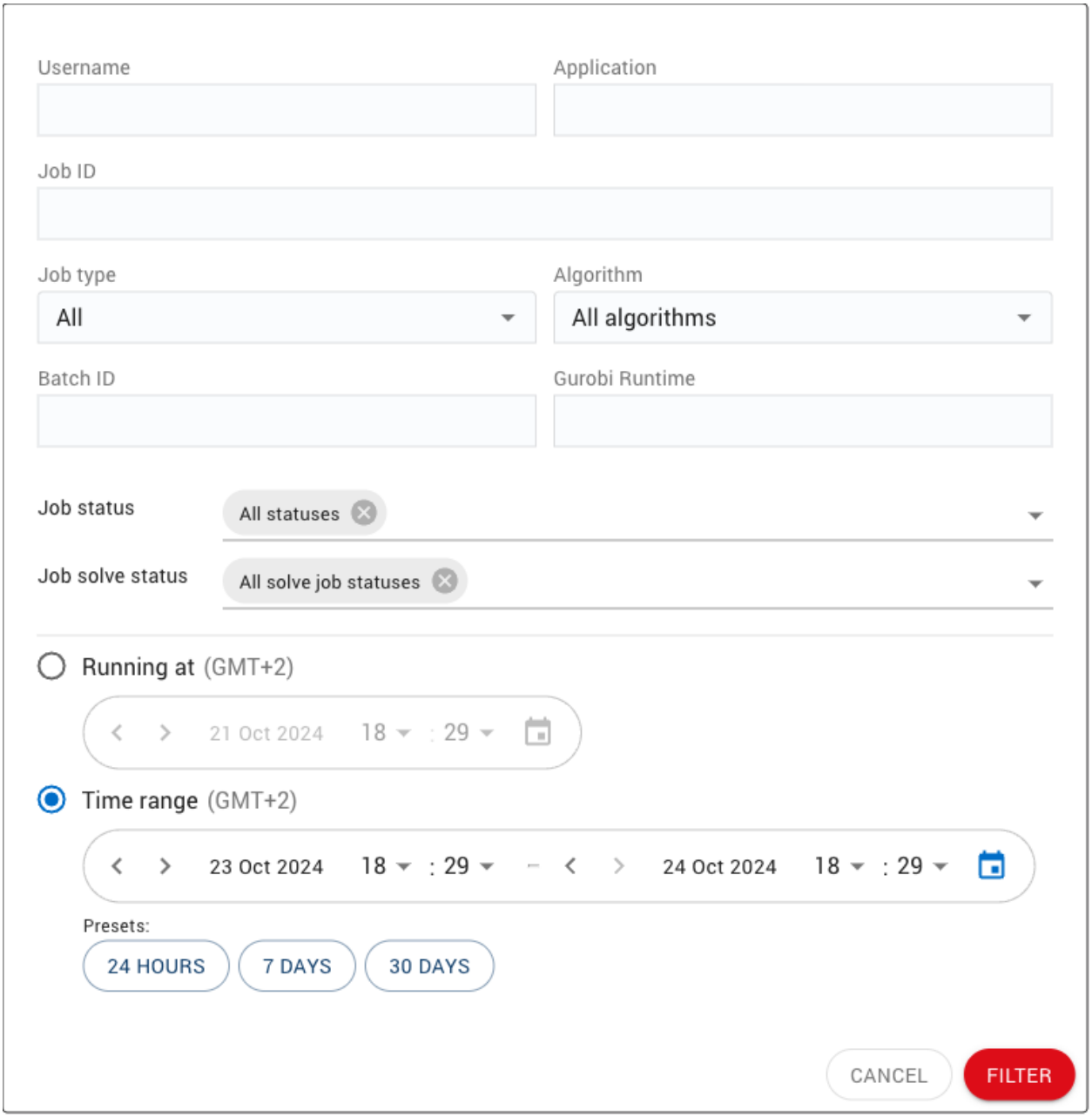
Different values of the selected timeline are listed with their associated color. In the case of job solve times, there is no legend.

Job filtering

The Filter box provides the user with a set of predefined filters as shown below:



When clicking on the left icon of the filter box, a dialog is displayed to edit the filters to apply.

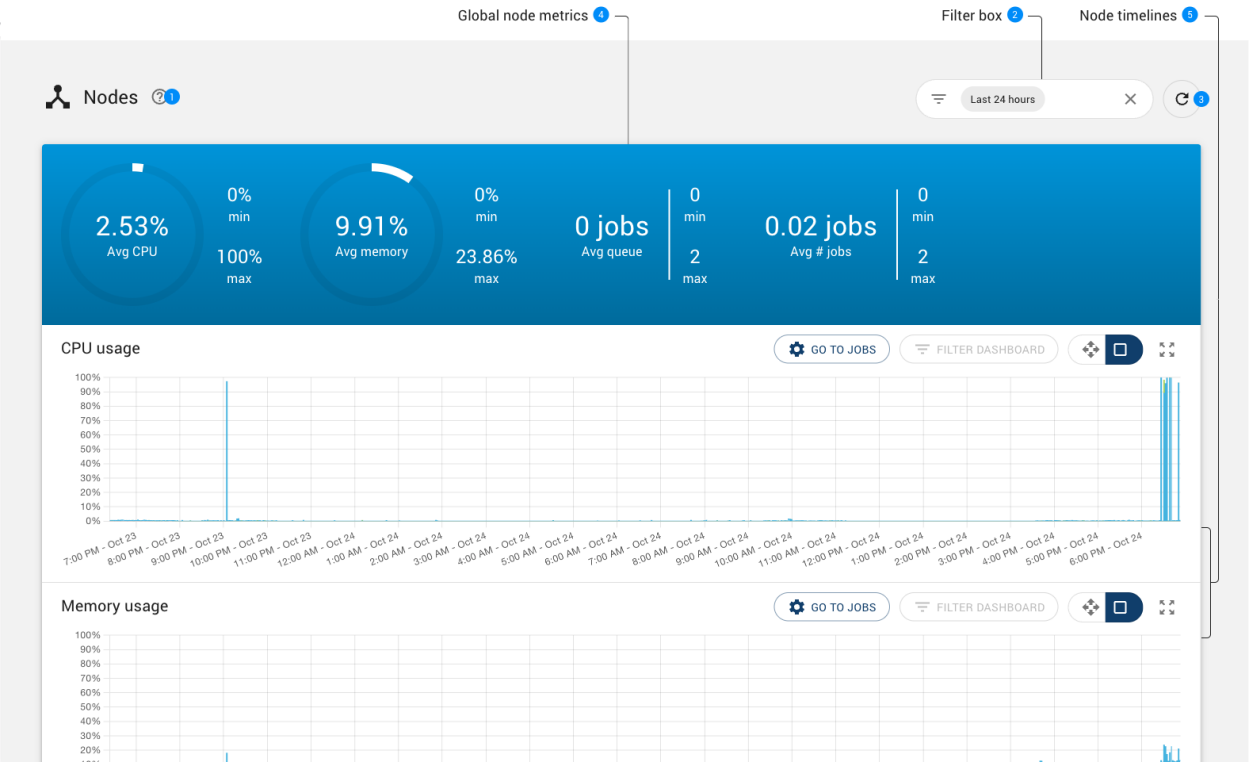


Here are the details of the filters of the filter box:

Filter	Description
Username	Filter jobs which username matches the specified `Username`
Application	Filter jobs which application matches the specified Application. The `Undefined` value can be used to filter jobs not associated with any applications.
Job ID	Filter the job which ID equals to the specified `Job ID`.
Job type	Filter jobs depending on whether they are associated with a batch or not. Possible filter values are: • `All` to not filter jobs on their type. • `Interactive jobs` to filter jobs not associated with a batch. • `Batch jobs` to filter jobs associated with a batch.
Algorithm	Filter jobs depending on the algorithm being applied. Possible values are `All algorithms`, `SIMPLEX`, `MIP`, `BARRIER` or `TUNNING`.
Batch ID	Filter jobs associated with a batch which ID equals to the specified Batch ID. If the Batch ID field is left empty, no filtering is performed based on the batch ID.
Runtime	Filter jobs associated with a batch a specific runtime. Match can be partial, for example "9.5" will match all the technical releases such as "9.5.1" and "9.5.2".
Job status	Filter jobs which status matches one of the selected job status. Possible filter values are: • `All statuses` to not filter jobs on their status. • `Aborted` to filter jobs aborted by the user. • `Failed` to filter jobs that have failed. • `Completed` to filter jobs that have completed. • `Disconnected` to filter jobs that have been disconnected. • `Idle timeout` to filter jobs withe IDLE_TIMEOUT status.
Solve status	Filter jobs which solve status matches one of the selected job solve status. Possible filter values are All job solve statuses, `INIT`, `COMPLETED`, `OPTIMIZING`, `LOADED`, `OPTIMAL`, `INFEASIBLE`, `INF_OR_UNBD`, `UNBOUNDED`, `CUTOFF`, `ITERATION_LIMIT`, `NODE_LIMIT`, `TIME_LIMIT`, `SOLUTION_LIMIT`, `INTERRUPTED`, `NUMERIC`, `SUBOPTIMAL`, `INPROGRESS`, `USER_OBJ_LIMIT`.
Running at	Filter jobs which where running at a specified date that is the jobs started before the selected date and ended after.
Time range	Filter jobs which were running during the selected time range that is the jobs started before the selected end date, and ended after the selected start date. The start and end date of the time range are edited with date field respectively left and right to - character. There are few ways for editing the start and end dates: • Press the < or > to respectively remove or add one day to the edited date. • Press any date field to launch a date dialog letting the user edit the start and end dates. • Press any hour or minute fields to select the start and end time. • Press the calendar button to quickly select the start and end date. • Press one of the preset buttons to set the start date to a specific duration from the current time.

Node dashboard

This node dashboard aggregates information about the usage of nodes of this cluster manager according to several dimensions: CPU usage, memory usage, the job queue and jobs that have been running on the cluster nodes. It provides a timeline for these dimensions so that users can track evolution of these metrics over time. The user can zoom or pan into the timeline, and drill down to the actual job list or filter the dashboard.



1. Help button to navigate to the documentation of the Node dashboard page.
2. Filter box to select the time range to filter the Node dashboard page. Details can be found in the *Filtering nodes* section.
3. Button to reload the page.

   Note that reloading the page is only effective if the selected time range ends the current day.
4. Global node metrics.

   Details can be found in the *Global node metrics* section.
5. Node timelines.

   Details can be found in the *Node timelines* section.

Global node metrics

Main metrics for the activity of nodes for the selected period of time.



CPU Usage

Average **(1)**, minimum **(2)** and maximum **(3)** CPU usage for the selected period of time.

Memory usage

Average **(4)**, minimum **(5)** and maximum **(6)** memory usage for the selected period of time.

Queue

Average **(7)**, minimum **(8)** and maximum **(9)** number of jobs in the queue for the selected period of time.

Number of jobs

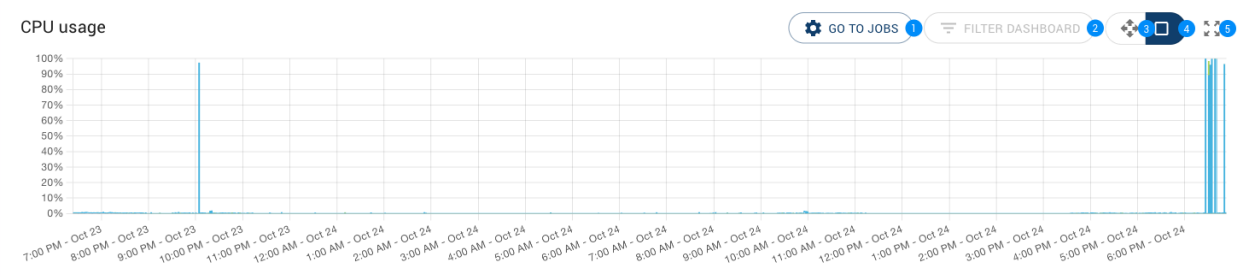
Average **(10)**, minimum **(11)** and maximum **(12)** number of jobs running on the cluster nodes for the selected period of time.

Node timelines

This section displays a list of timelines that each shows the evolution of one node metric overtime, such as the CPU or memory usage.

Node timelines are synchronized with each others. If one timeline is zoomed in into a specific time range, other timelines are automatically zoomed in to same time range. This allows the user to always get a full overview of the node activity when zooming in into a time range of interest.

Node timeline features



1. Go to jobs

   This button navigates to the *Job History page*, applying the current visible time range in the timeline chart as the time range filter for the jobs page.

2. Filter dashboard

   Applies the current visible time range in the timeline chart to the filter of the dashboard page.

3. 4. 5. Zoom features

   The timeline chart provides two modes for changing the visible time range of the chart.

   - Pan, activated with the Pan **(3)** button In this mode, pressing the left mouse button on the chart and dragging the mouse left of right will scroll horizontally the timeline, unless no zoom in has been performed on the chart.
   - zoom in to a specified time range, activated with the Drag **(4)** button

     This mode is selected by default.

     In this mode, pressing the left mouse button on the chart and dragging the mouse left of right will select a time range, visible with a blue rectangle. When releasing the left mouse button, the chart will zoom in the zone corresponding to the dragged area.

At any time, you can restore the timeline chart to its initial zoom state clicking the restore zoom **(5)** button.

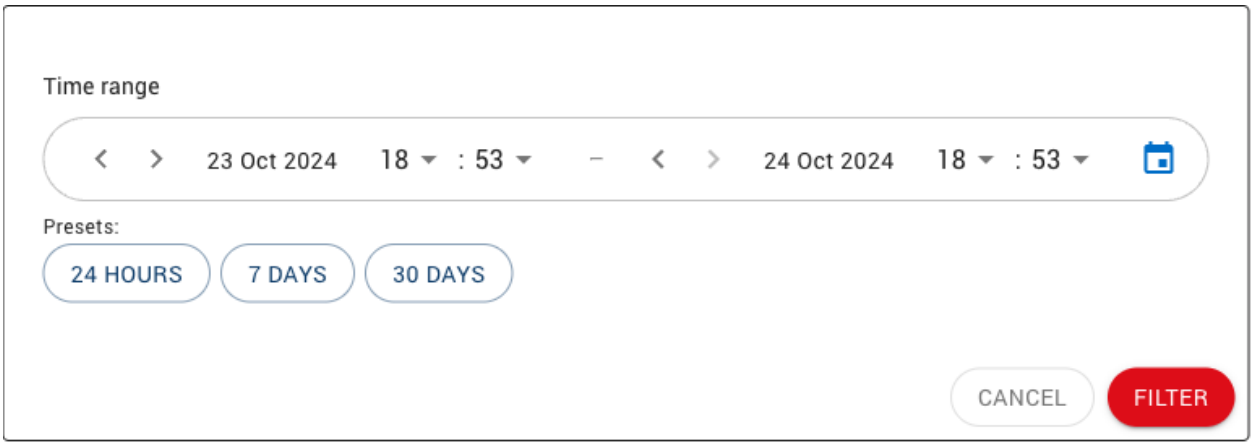
To zoom in/zoom out with the mouse wheel, keep the `Ctrl` key pressed while using the mouse wheel.

Node filtering

The Filter box provides the user with a set of predefined filters as shown below:



When clicking on the left icon of the filter box, a dialog is displayed to edit the filters to apply.



Here are the details of the filters of the filter box:

Filter	Description
Time range	Filter node usage on a specified time range. There are few ways for editing the start and end dates: • Press the < or > to respectively remove or add one day to the edited date. • Press any date field to launch a date dialog letting the user edit the start and end dates. • Press any hour or minute fields to select the start and end time. • Press the calendar button to quickly select the start and end date. • Press one of the preset buttons to set the start date to a specific duration from the current time.

4.3.2 Cluster Jobs

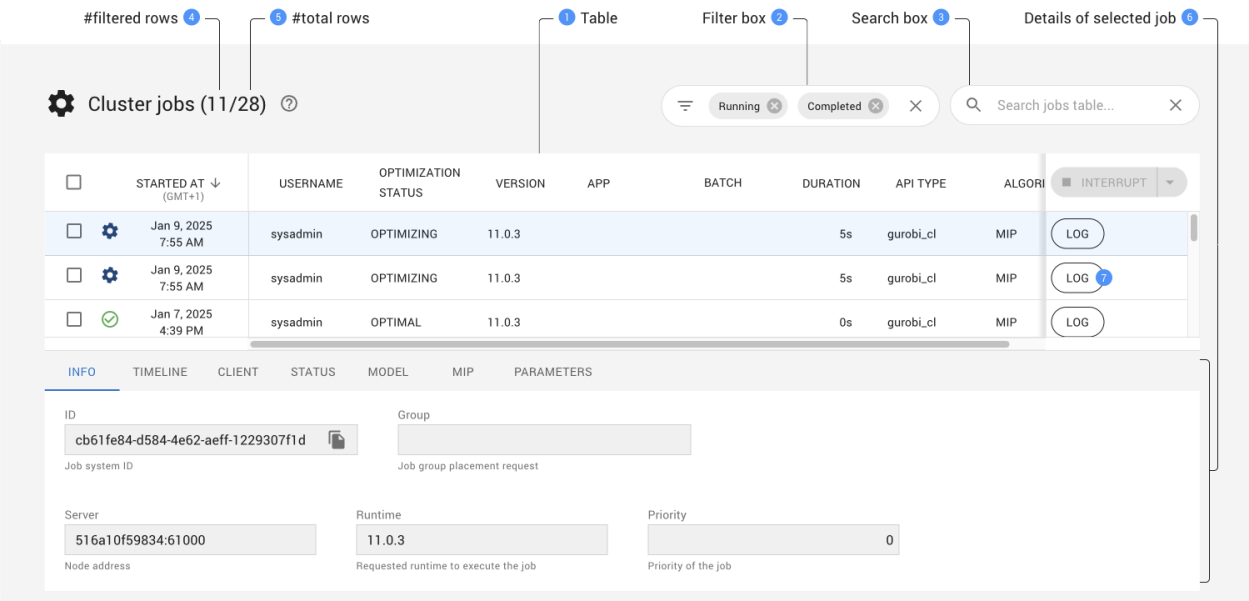
A job is an optimization task performed on behalf of a client. Running a job requires server resources, which must be provisioned in the cluster. If no resources are available, a job can be queued for later processing.

There are two types of jobs:

- **Interactive jobs** handle optimization sessions controlled by a client in real time. The client must stay connected for the duration of the session.
- **Batch jobs** are also generated by a client, but for offline processing. The client uploads a batch specification to the Cluster Manager containing the input data needed for the optimization task. When the job later runs, it retrieves the relevant data from the Cluster Manager, performs the optimization, and stores optimization results back to the Cluster Manager. A client can then retrieve the results.

The Jobs Page

The Jobs page in the Cluster Manager looks like this:



1. This page shows a table of all jobs that have been created on a Cluster Manager (completed, active, and queued). Each line in the table provides information about a job, including the current status, the creation time, the Gurobi version and algorithm used to solve it, and the API used to create the job. The current job status is shown using an icon:

	Job has been queued.
	Job is running.
	Job has been aborted.
	Job has failed.
	Job has successfully completed.
	Job has been rejected.
	Job has been disconnected.
	Job has reached the idle timeout.

Jobs can be sorted by various attributes by clicking on the corresponding column headers. Up and down arrow icons allow you to select the sorting direction.

2. 3. Jobs displayed in the table can be filtered using either the Filter box or the Search box. Details can be found in the *Filtering and Searching* section.

4. 5. The number of rows of the table.

   When the table has been filtered, two numbers are displayed to the right of the page title showing respectively the number of jobs matching the current filtering and the total number of jobs. If no jobs have been filtered, only the total number of jobs wil be displayed.

6. When you select a job from the table (highlighted in blue), the tabs below the table allow you to obtain additional information about that job. Choosing a tab brings up the corresponding information. Note that the title of one of the tabs, as well as the information that is displayed under that tab, depends on the optimization algorithm used for that job. Options are `MIP`, `BARRIER`, or `SIMPLEX`.

7. The **LOG** button opens a new page displaying a dashboard for the job that includes:

   - A panel containing the optimization log for that job.
   - The Objective graph showing the progress of the best objective over the time.
   - Tabs that provide more details about the job (which are equivalent to tabs shown in **(6)**

   See the *Log* section for more details.

Filtering and Searching

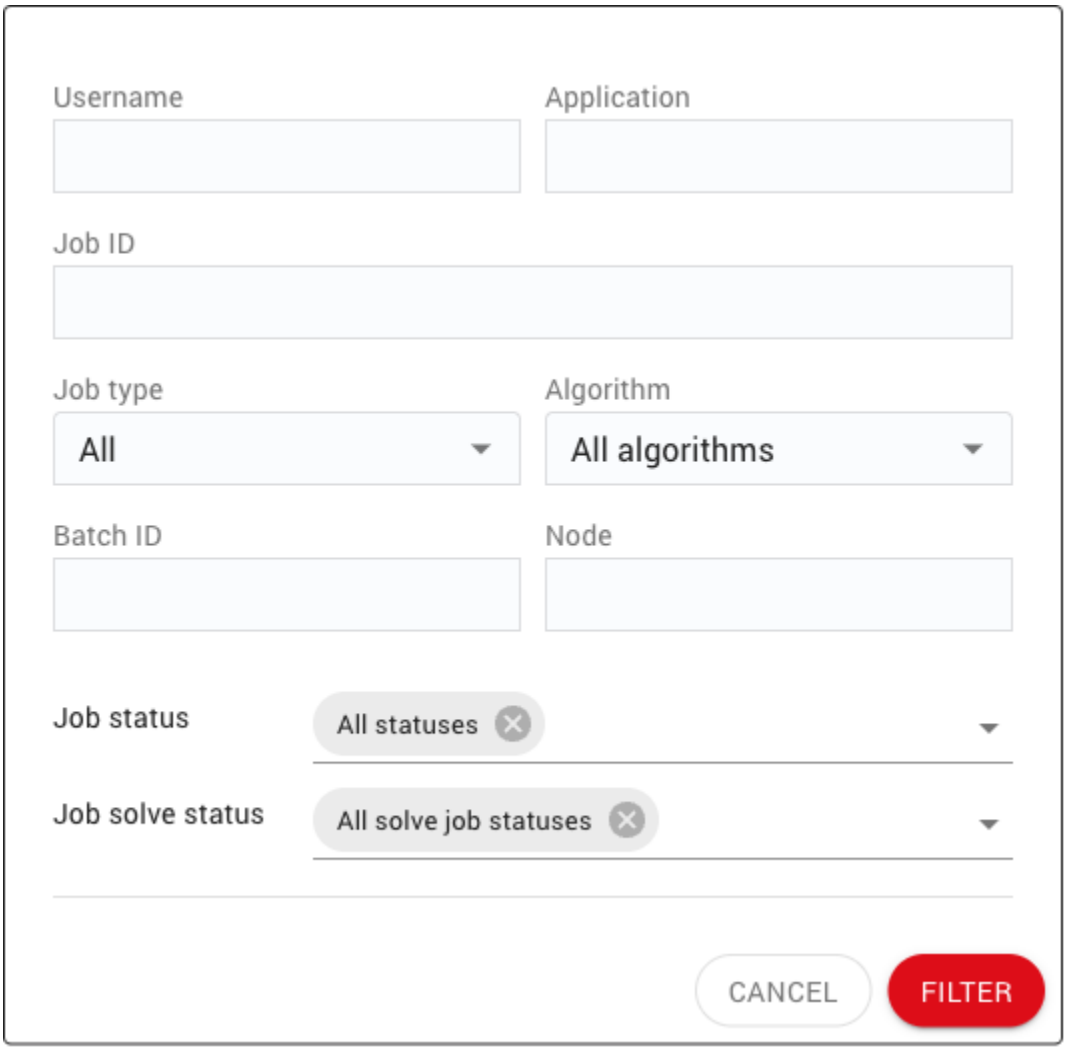
Jobs displayed in the table can be filtered using the Filter box and the Search box.

Filtering

The Filter box provides the user with a set of predefined filters as shown below:



When clicking on the left icon of the filter box, a dialog is displayed to edit the filters to apply.



The content of the filter box varies depending on the job page, whether it is the Cluster jobs, Queue or History job page. The following job filters can be found in all or few of the job pages:

Username	Filter jobs which username matches the specified `Username`
Application	Filter jobs which application matches the specified Application. The `Undefined` value can be used to filter jobs not associated with an application.
Job ID	Filter the job which ID equals to the specified `Job ID`.
Job limit	Specify the maximum number of jobs returned by the server. By default, the limit is set to `200` jobs. Other filter values are `100` or `500` maximum jobs.
Job type	Filter jobs depending on whether they are associated with a batch or not. Possible filter values are: • `All` to not filter jobs on their type. • `Interactive` jobs to filter jobs not associated with a batch. • `Batch` jobs to filter jobs associated with a batch.
Algorithm	Filter jobs depending on the algorithm being applied. Possible values are `All algorithms`, `SIMPLEX`, `MIP`, `BARRIER` or `TUNNING`.
Batch ID	Filter jobs associated with a batch which ID equals to the specified Batch ID. If the Batch ID field is left empty, no filtering is performed based on the batch ID.
Node	Filter jobs which node address matches the specified Node. If the Node field is left empty, no filtering is performed based on the node.
Job status	Filter jobs which status matches one of the selected job status. Possible filter values are • `All statuses` to not filter jobs on their status. • `Aborted` to filter jobs aborted by the user. • `Failed` to filter jobs that have failed. • `Completed` to filter jobs that have completed. • `Disconnected` to filter jobs that have been disconnected. • `Idle timeout` to filter jobs withe `IDLE_TIMEOUT` status.
Solve status	Filter jobs which solve status matches one of the selected job solve status. Possible filter values are `All job solve statuses`, `INIT`, `COMPLETED`, `OPTIMIZING`, `LOADED`, `OPTIMAL`, `INFEASIBLE`, `INF_OR_UNBD`, `UNBOUNDED`, `CUTOFF`, `ITERATION_LIMIT`, `NODE_LIMIT`, `TIME_LIMIT`, `SOLUTION_LIMIT`, `INTERRUPTED`, `NUMERIC`, `SUBOPTIMAL`, `INPROGRESS`, `USER_OBJ_LIMIT`, `WORK_LIMIT`, `MEM_LIMIT`.
Running at	Filter jobs which where running at a specified date. If the Running at checkbox is unchecked, no filtering is applied for this filter and running at fields are disabled. Otherwise, jobs are filtered if their started date is before or equal to the specified date and if their end date os after the selected date.
Time range	Filter jobs which started date are on a specified time range. If the Time range checkbox is unchecked, no filtering is applied on job started date and time range fields are disabled. Otherwise, jobs are filtered if their started date is after the start of the time range and before the end of the time range. The start and end date of the time range are edited with date field respectively left and right to - character. There are few ways for editing the start and end dates: • Press the <> to respectively remove or add one day to the edited date. • Press any date field to launch a date dialog letting the user edit both start and end dates. • Press one of the preset buttons to set the start date to a specific number of days before Today and set the end date to Today.

By default, users having one of the roles `ADMIN`, `SYSADMIN`, or `READONLY` will see all jobs. Other users will only see jobs they have launched. This default filter can be changed from the Filter box.

When the table has been filtered, two numbers are displayed to the right of the page title showing respectively the number of jobs matching the current filtering and the total number of jobs. If no jobs have been filtered, only the total number of jobs wil be displayed.

Searching

The Search box enables users to further refine the results by specifying an arbitrary search string. In order for a job to be displayed, the search string must be present in at least one of the columns in the table for that job, or in the job ID or job batch ID.

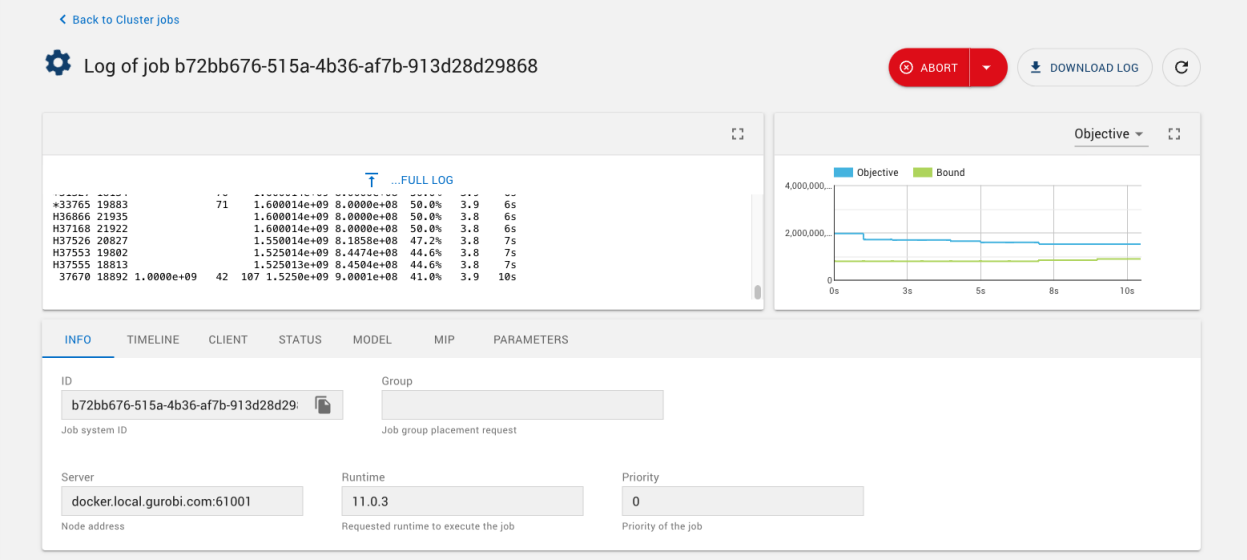
> **Note**
>
> It is important to note that filtering with the Filter box is usually applied server side, while searching with the Search box is always performed client side. That means that the searching is done on results that were already filtered by the server, so it it is preferable to use the Filter box first and then use the Search box to refine the results. Using the Search box as an alternative to the Filter box could result in missing items.

Synchronization with page URL

The Cluster Manager creates permalinks for all tables that can be filtered or searched. Changes applied using either the Filter or Search boxes are automatically reflected in the page URL, thus allowing you to copy and paste the URL to easily retrieve the same display later. For example, the URL `http://localhost:61080/jobs/main?status=RUNNING` would display only jobs that are currently running (assuming the Cluster Manager is running on localhost:61080).

Log and Objective

LOG	The Log button opens the job dashboard. This page displays the optimization log for the job, as well as an Objective progress chart and a set of tabs that allow you to obtain additional information about the job.



Log

Within the job dashboard, the Log panel shows the optimization log. If the job is still running, new log text is automatically appended as soon as it is available.

	If the job ran before you navigated to the Log panel, older log data may not be displayed. The `FULL LOG` button allows you to fetch and display the log from the beginning.
	To download the full job log to a file once the job is completed, click on the `DOWNLOAD LOG` button.

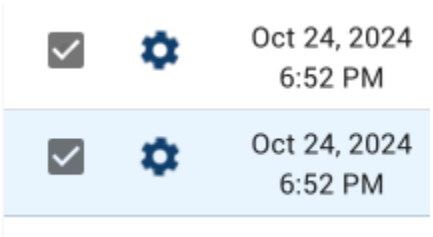
Objective

The Objective chart shows optimization progress over time by displaying objective values for solutions found by the optimization algorithm. For MIP models, the Objective chart can be switched to show progress in the optimality Gap, using the top-right selection box in the Objective panel.

Aborting Jobs

Jobs can be aborted from the Jobs page with the following steps:

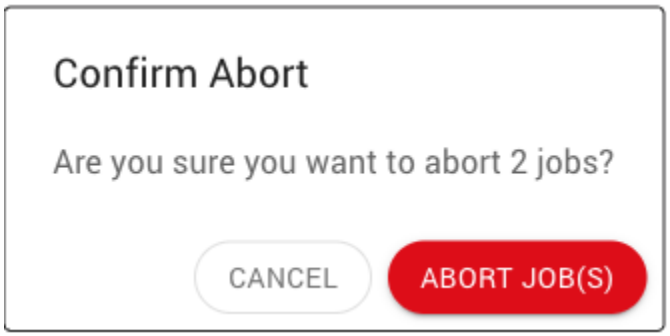
- Select the job(s) to be aborted (by clicking on the checkbox on the far left of the job display).

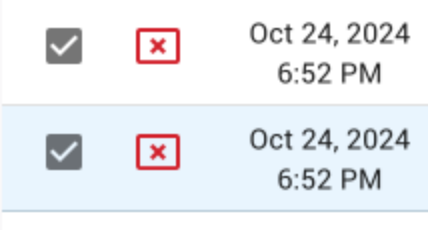


- Click the abort button.



- Confirm that the selected jobs should be aborted.





- Aborted jobs will show a change of status once the abort has been completed and recorded:

Note

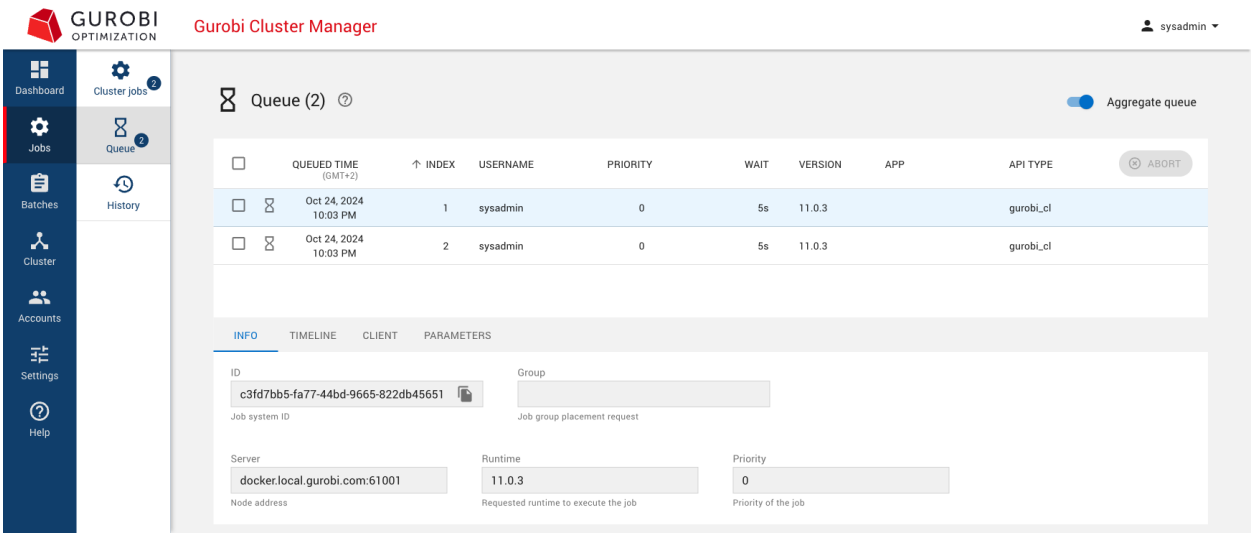
`STANDARD` users can only abort jobs they launched.

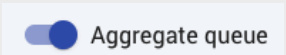
`READONLY` users cannot abort any jobs.

`SYSADMIN` and `ADMIN` users can abort jobs of any users.

Job Queue

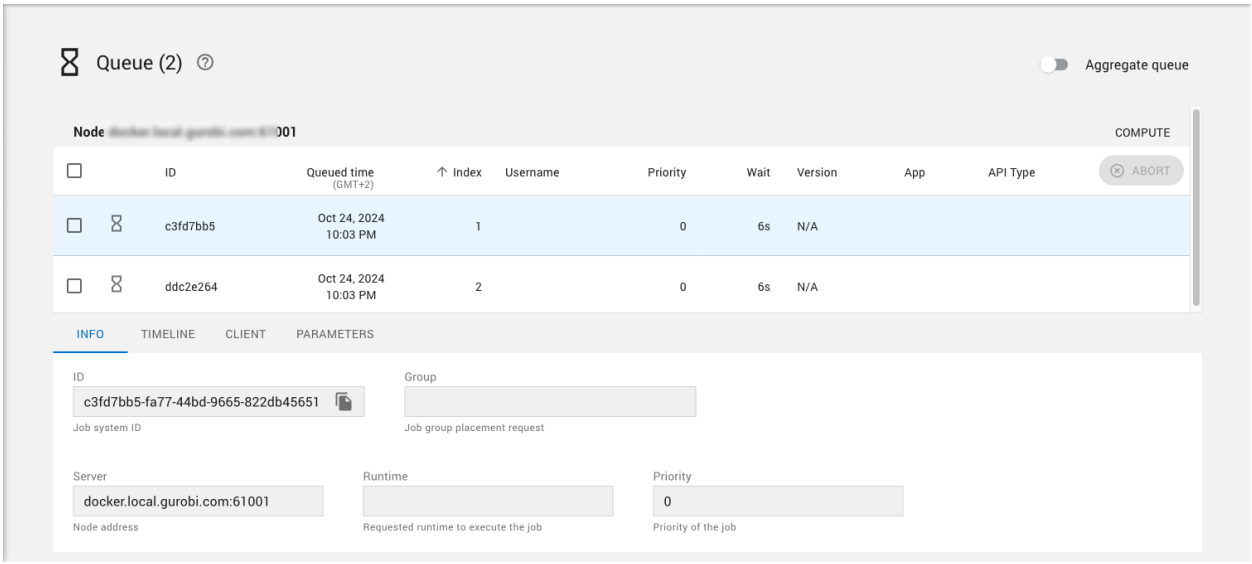
Each Compute Server node has a job limit that constrains the number of jobs that can be run simultaneously. When total available capacity in the cluster has been exhausted, new jobs are queued until capacity becomes available. Queued jobs can be viewed in the Queue page:





A queued job is actually queued in all compatible nodes. The Queue page displays an aggregated view by default, showing only the best position for each job across all nodes. Use the Aggregate queue toggle to see the queue for each individual node.

Queued jobs can be aborted, just like jobs in the Cluster Jobs page. Refer to the *Abort job(s)* section for details.

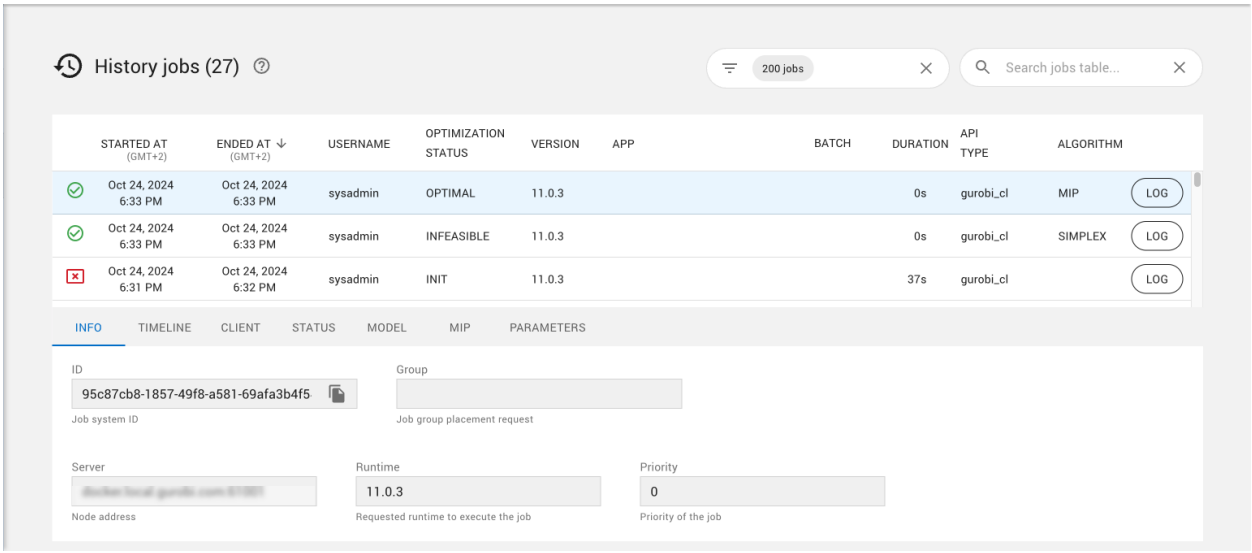


Job History

The Cluster jobs page displays running jobs and recently completed jobs. The Job history page gives access to all completed jobs up to a limit in time that is controlled with the History expiration setting in the *Data Management* settings section.

By default, the Cluster manager stores the completed jobs for the past 30 days.

To access this larger set of jobs, the history page extends the filter box of the Cluster job page with more filters including filtering by starting time or for a specific Gurobi runtime. Refer to the *Filtering and Searching* section for more details.

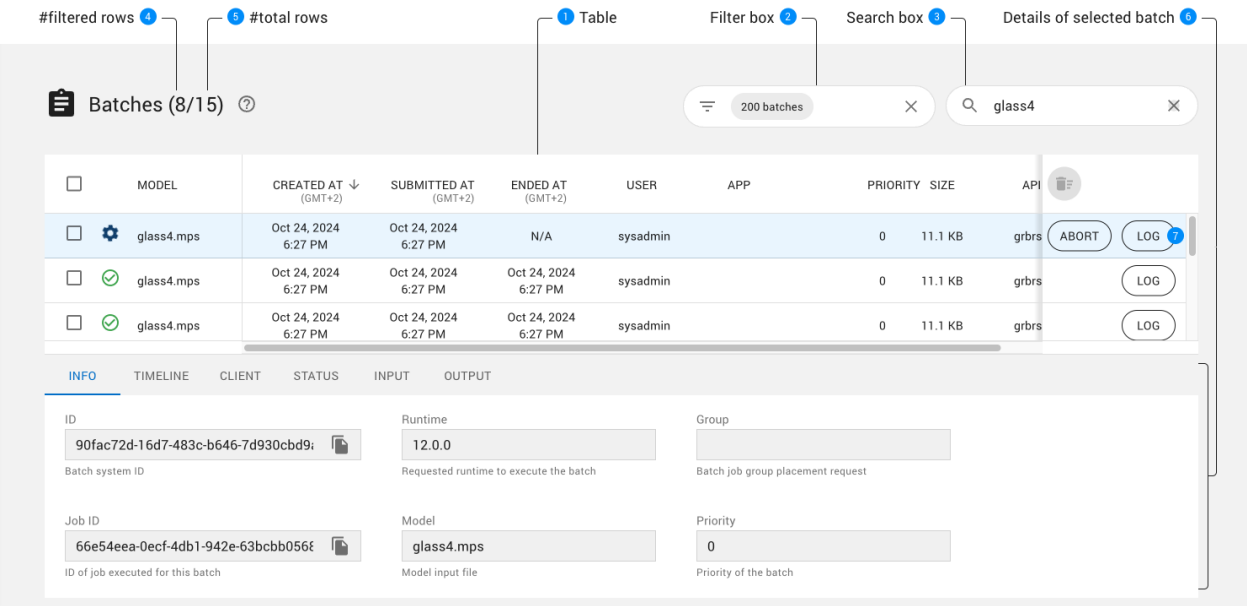


4.3.3 Batches

A batch is a specification for an optimization task. It contains all inputs necessary to perform the task (model data and optimization parameters), as well as a description of the results that are needed upon completion. A batch specification is built off-line and submitted to the Cluster Manager. The Cluster Manager then submits the batch job for execution, at which point the optimization model will be solved and results will be generated. If a batch job fails for reasons outside of its control (e.g., the node it was running on went down), a new job will be created to retry the batch. Once the batch has completed, the results are stored on the Cluster Manager and a client can retrieve them.

The Batches Page

The Batches page in the Cluster Manager looks like this:



1. **This page shows a table of all batches that have been created on the Cluster Manager (created, aborted, and completed). Each line in the table provides information about a batch, including the current status, the creation and completion times, the priority, and the API used to create the job. The current batch status is shown using an icon:**

	Batch has been created.
	Batch has been submitted, and an associated job has been submitted.
	Batch has been aborted; associated job will be aborted (if it hasn't been already).
	Job associated with the batch has failed.
	Job associated with the batch has successfully completed.

   Batches can be sorted by various attributes by clicking on the corresponding column headers. Up and down arrow icons allow you to select the sorting direction.

2. 3. Batches displayed in the table can be filtered using either the Filter box or the Search box. Details can be found in the *Filtering and Searching* section.

4. 5. The number of rows of the table.

   When the table has been filtered, two numbers are displayed to the right of the page title showing respectively the number of batches matching the current filtering and the total number of batches. If no batches have been filtered, only the total number of batches wil be displayed.

6. When you select a batch from the table (highlighted in blue), the tabs below the table allow you to obtain additional information about that batch. Choosing a tab brings up the corresponding information.
7. The **LOG** button opens a new page displaying a dashboard for the job associated with that batch. The page includes:
   - A panel containing the optimization log for that job.
   - The Objective graph showing the progress of the best objective over the time.
   - Tabs that provide more details about the job.

   See the *Log* section for more details.

Filtering and Searching

Batches displayed in the table can be filtered using the Filter box and the Search box.

Filtering

The Filter box provides the user with a set of predefined filters as shown below:



When clicking on the left icon of the filter box, a dialog is displayed to edit the filters to apply.

By default, users having one of the roles `ADMIN`, `SYSADMIN`, or `READONLY` will see all batches. Other users will only see batches they have launched. This default filter can be changed from the Filter box.

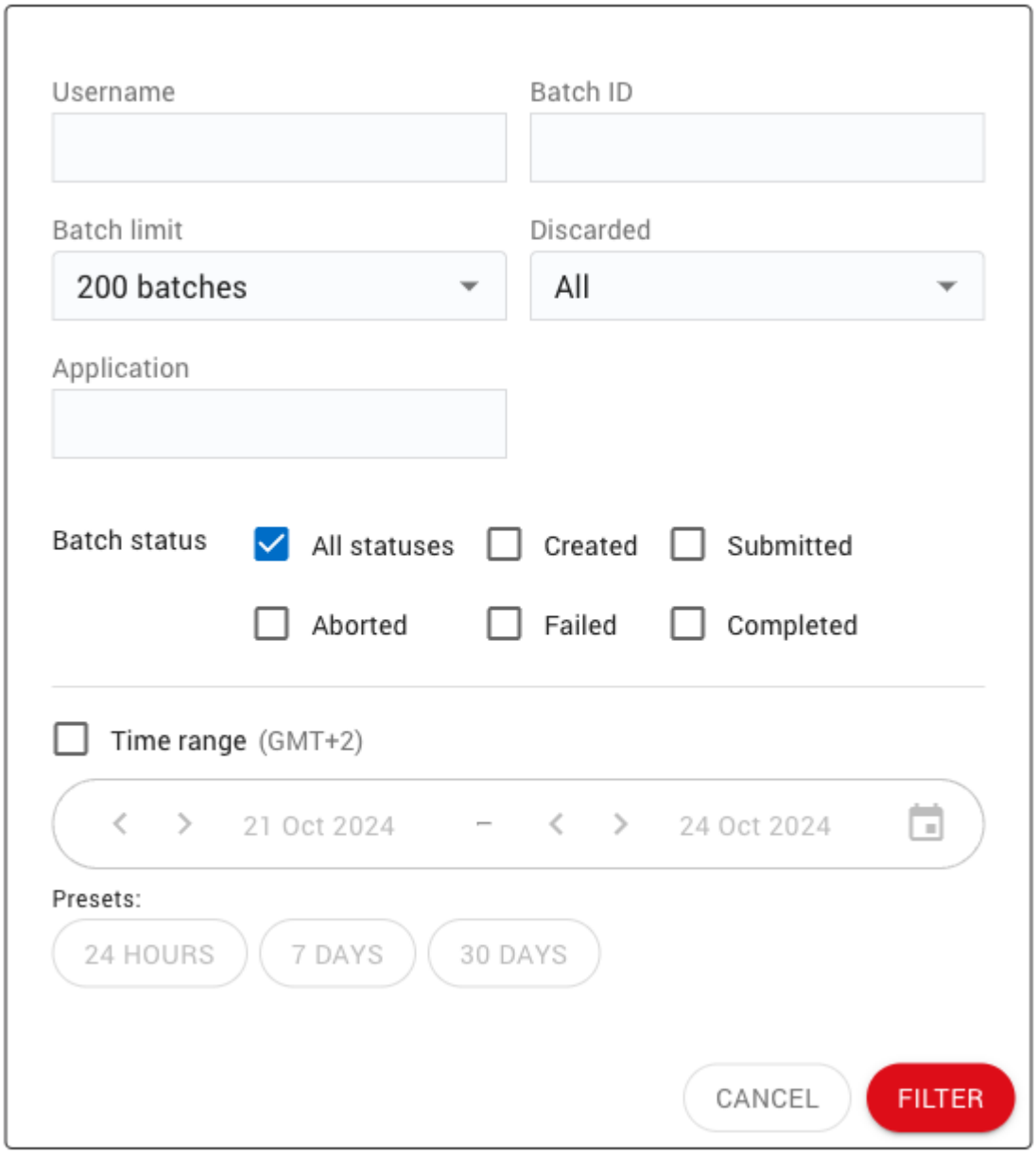
When the table has been filtered, two numbers are displayed to the right of the page title showing respectively the number of batches matching the current filtering and the total number of batches. If no batches have been filtered, only the total number of batches wil be displayed.

Searching

The Search box enables users to further refine the results by specifying an arbitrary search string. In order for a batch to be displayed, the search string must be present in at least one of the columns in table for that batch, or in the job batch ID.

Note

It is important to note that filtering with the Filter box is usually applied server side, while searching with the Search box is always performed client side. That means that the searching is done on results that were already filtered by the server, so it it is preferable to use the Filter box first and then use the Search box to refine the results. Using the Search box as an alternative to the Filter box could result in missing items.


Username
Batch ID
Batch limit
200 batches
Discarded
All
Application
Batch status
All statuses
Created
Submitted
Aborted
Failed
Completed
Time range (GMT+2)
21 Oct 2024
24 Oct 2024
Presets:
24 HOURS
7 DAYS
30 DAYS
CANCEL
FILTER

Synchronization with page URL

The Cluster Manager creates permalinks for all tables that can be filtered or searched. Changes applied using either the Filter or Search boxes are automatically reflected in the page URL, thus allowing you to copy and paste the URL to easily retrieve the same display later. For example, the URL `http://localhost:61080/batches/main?status=SUBMITTED` would display only batches in a submitted state (assuming the Cluster Manager is running on `localhost:61080`).

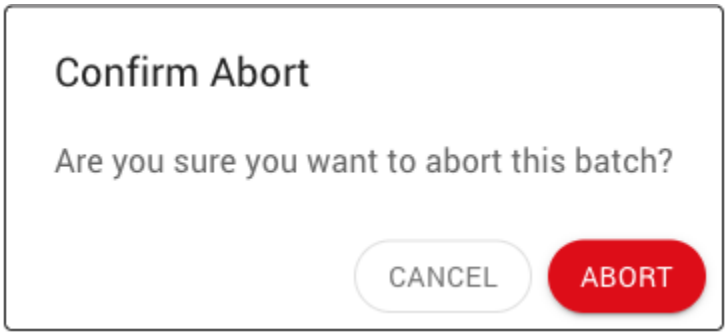
Aborting Batches

Batches can be aborted from the Batches page with the following steps:

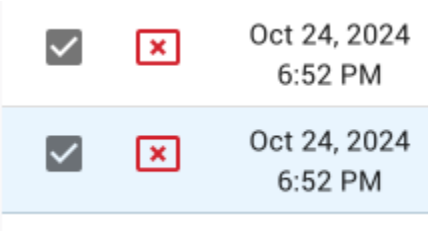
- Click on the `ABORT` button for the batch you want to abort.



- Confirm that the selected batch should be aborted.



- Aborted batches will show a change of status once the abort has been completed and recorded:



Note

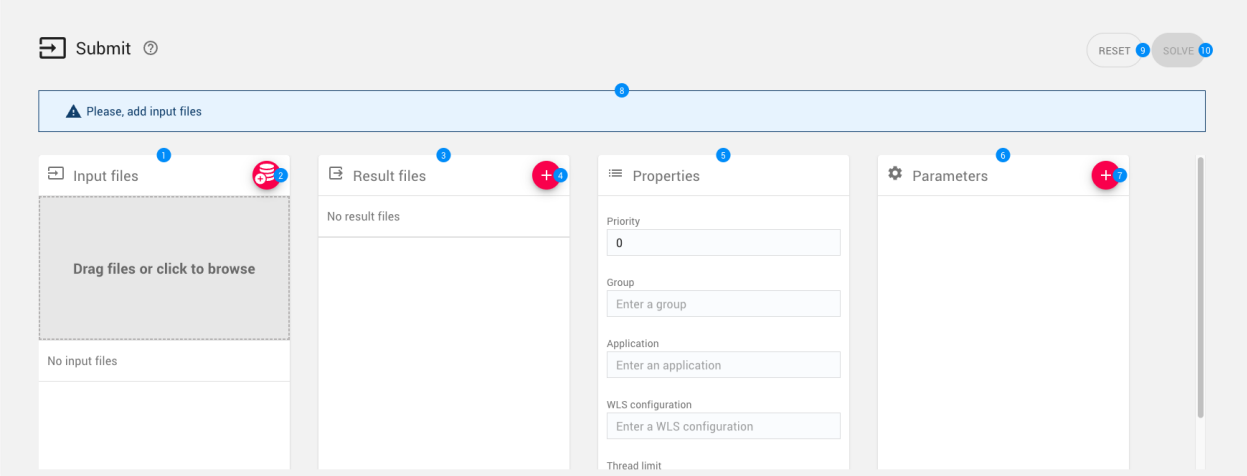
`STANDARD` users can only abort batches they launched.

`READONLY` users cannot abort any batches.

`SYSADMIN` and `ADMIN` users can abort batches of any users.

Submitting Batches

The Submit Batch page allows users to submit batches interactively. It provides an alternative to using the `grbcluster batch` command to submit a batch from the command line (see `grbcluster batch --help` for details on command-line batch submission).



1. 2. The first pane on this page allows you to select the model files for the batch. Files can either be:
   - Dragged-and-dropped into the drop area of the pane.
   - Selected from a file chooser (by clicking on the drop area).
   - Selected from the file repository (by clicking on the repository button) **(2)**.

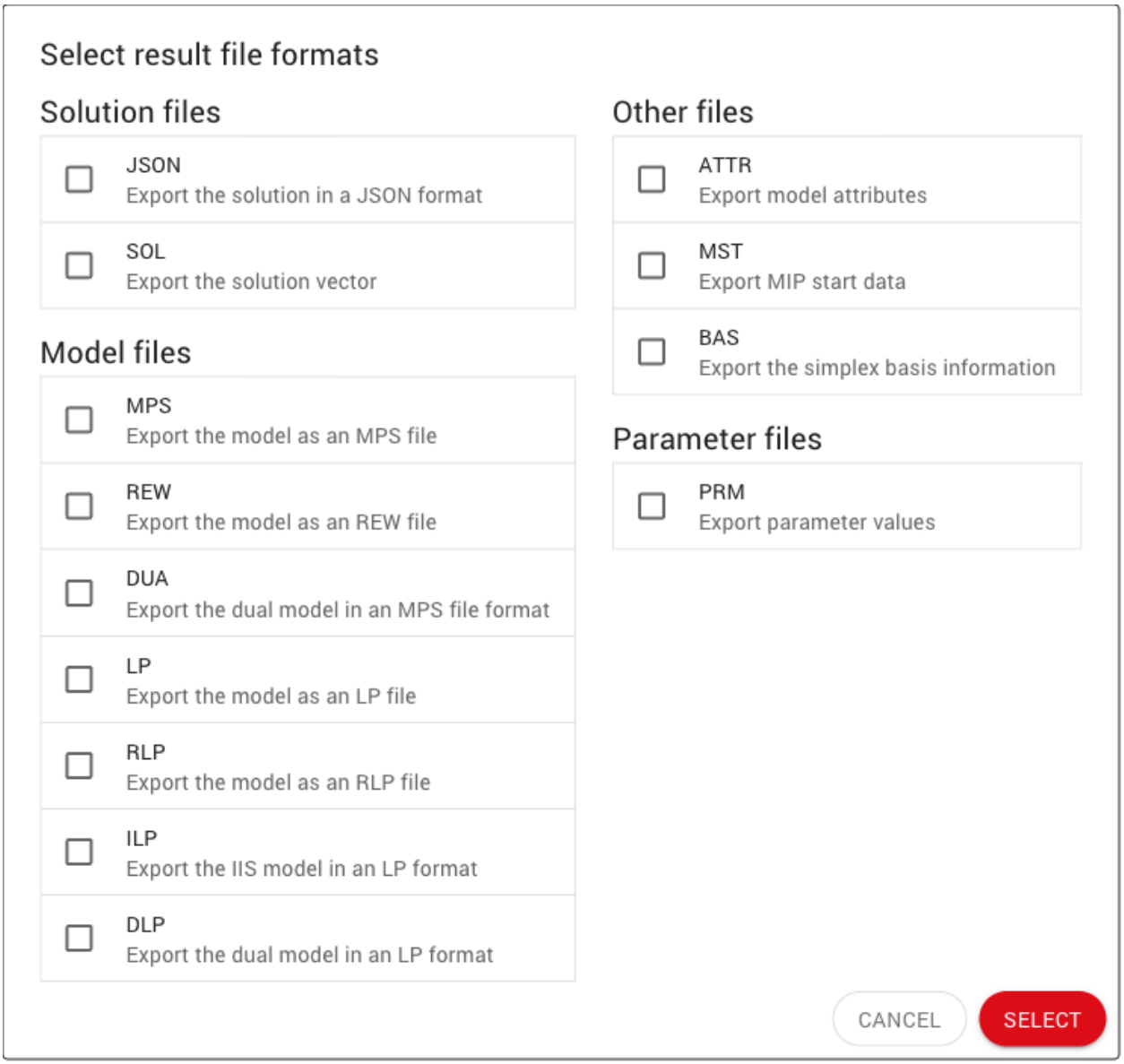
   See the *Repository* section for details on adding files to the repository.
3. 4. The second page allows you to indicate what result files you would like. The top-right button of the pane **(4)** opens a dialog that allows you to select the format of result file(s).

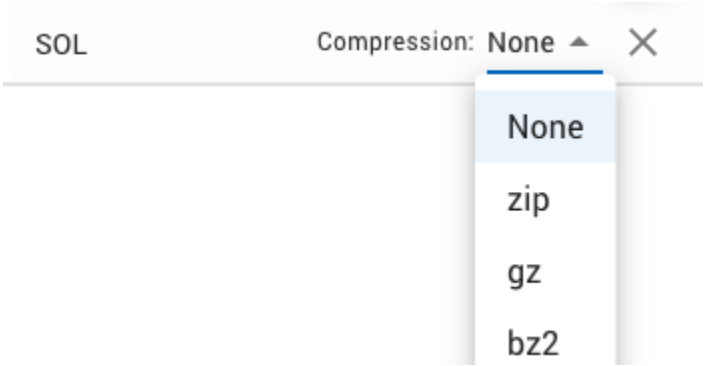
   After result file formats have been selected and the dialog has been closed, the selected formats will appear in the Result files pane.

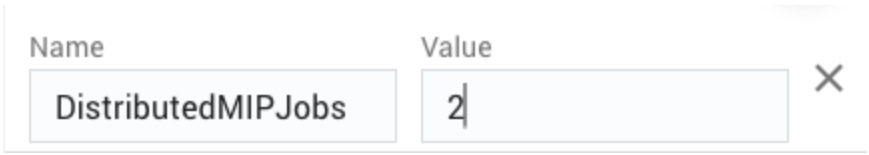
   Note that you can request that the result files be compressed using a dropdown menu:
5. The third pane allows you to configure batch properties. The user can specify the `Priority`, `Group` and `Application` for the batch.
6. 7. The fourth pane allows you to set batch parameters.

   Click on the top-right button **(7)** to add a new Gurobi parameter. These parameters are passed to the optimization engine and affect the solution process. Edit the `Name` and `Value` fields for the new the parameter:
8. The page includes a section that displays messages or warnings that alert the user to problems with the configuration of the batch, and suggests actions to perform to correct those problems..

   For example, if no model has been specified, this pane displays a warning to add input files.
9. Button to reset the batch configuration in this page. Clicking this button resets all edits performed on the Input files, Result files, Properties and Parameters panes.
10. Button to submit the batch as configured on this page. If the button is disabled, there's a problem with the batch, and the message pane **(8)** will provide suggestions for correcting the problem.

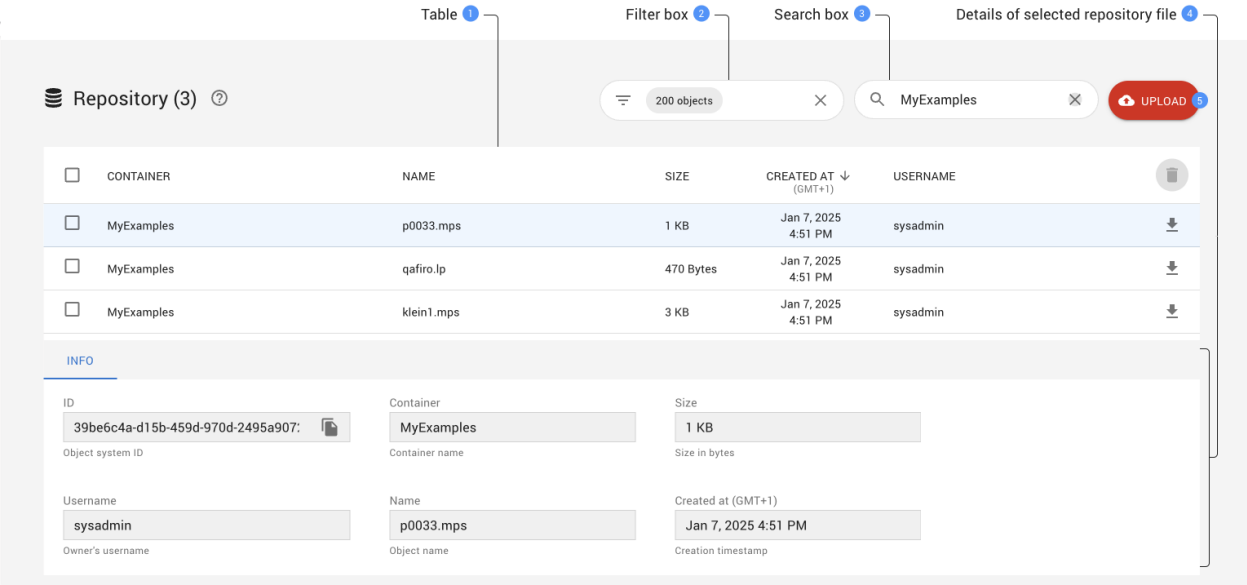

Select result file formats
Solution files
JSON
Export the solution in a JSON format
SOL
Export the solution vector
Model files
MPS
Export the model as an MPS file
REW
Export the model as an REW file
DUA
Export the dual model in an MPS file format
LP
Export the model as an LP file
RLP
Export the model as an RLP file
ILP
Export the IIS model in an LP format
DLP
Export the dual model in an LP format
Other files
ATTR
Export model attributes
MST
Export MIP start data
BAS
Export the simplex basis information
Parameter files
PRM
Export parameter values
CANCEL
SELECT


SOL
Compression: None
None
zip
gz
bz2



Repository

The repository page allows you to upload and manage files that can be reused when submitting a batch. For example, you may want to upload a single model file but submit multiple batches that solve that same model but using different parameters. You may also want to upload a MIP start (`mst`) or attribute (`attr`) file to reuse across multiple models.



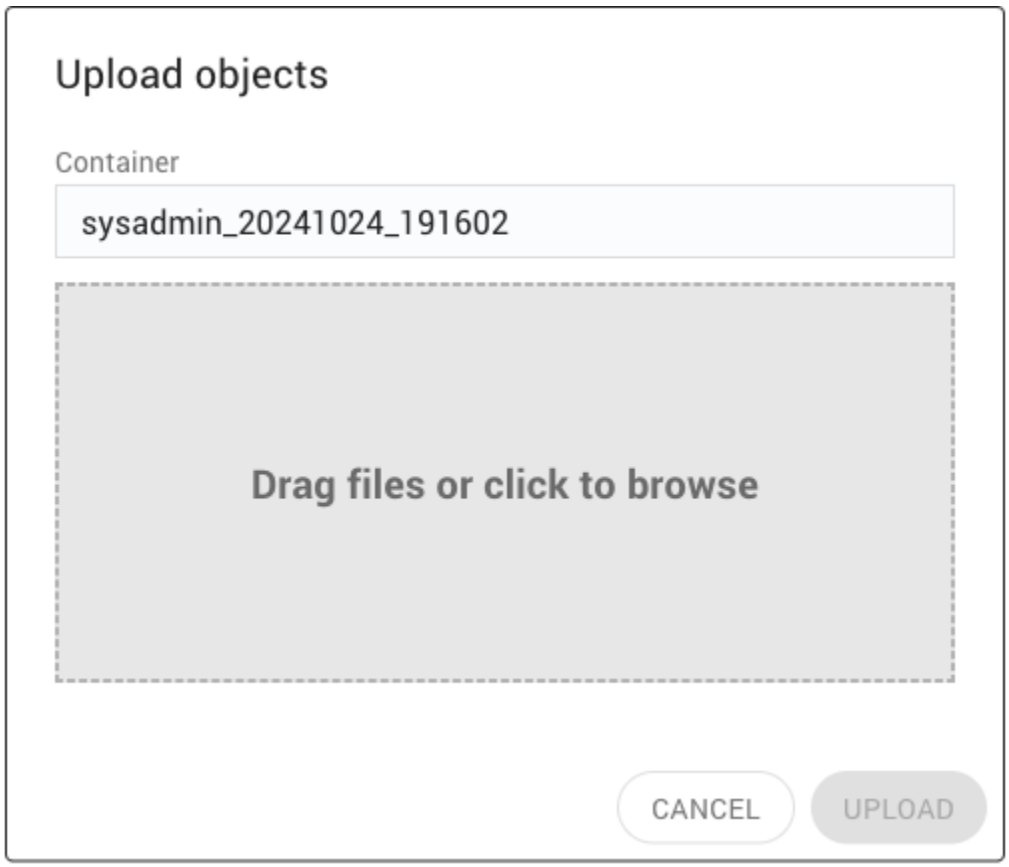
1. This pages shows a table listing the files that have been uploaded to the repository.
2. 3. The list can be filtered using the Filter and Search boxes.
4. When you select a repository file from the table (highlighted in blue), the tabs below the table allow you to obtain additional information about that file. Choosing a tab brings up the corresponding information.
5. You can upload new file(s) to the repository.

   Clicking on the **UPLOAD** button pops up a dialog box where users can drag-and-drop or select files to upload.

4.3.4 Clusters

A cluster is a set of one or more Compute Server nodes where optimization jobs can be performed. Each node has a job limit that indicates the maximum number of jobs that can be executed on that node simultaneously. The limit should reflect the capabilities of the machine (cores, memory, etc.) and typical job characteristics.

Compute Server nodes support a number of advanced capabilities such as job queueing and load-balancing. And if a node fails, new jobs are dispatched to other available nodes.

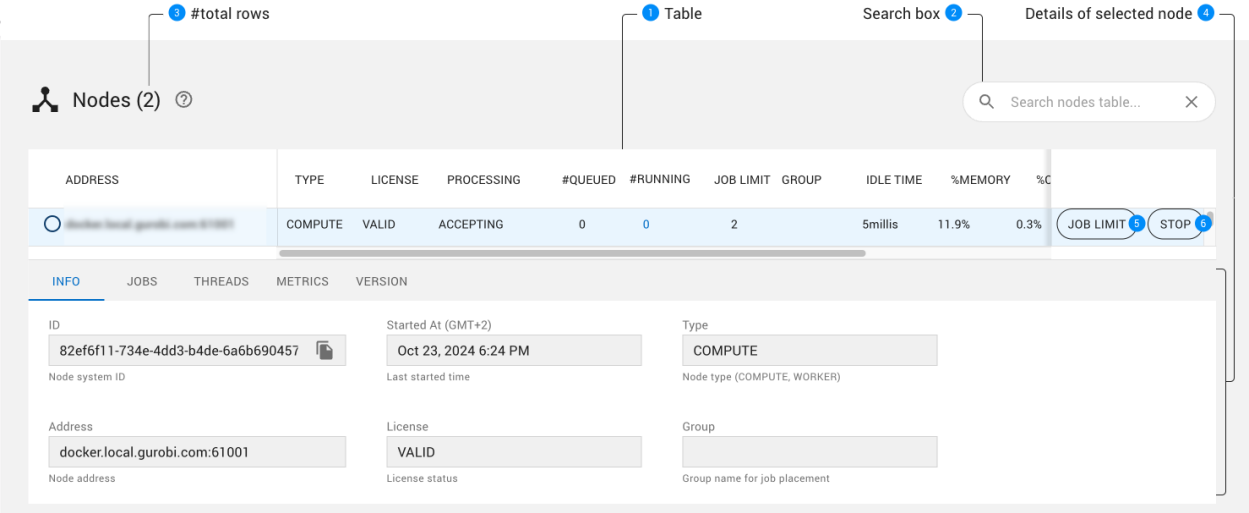


The Cluster Nodes Page

The Cluster Nodes page in the Cluster Manager looks like this:

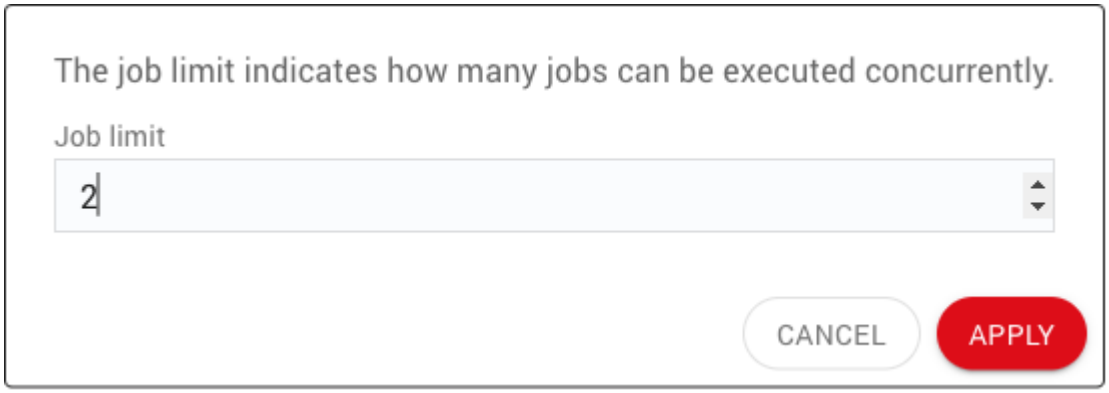
1. This page displays the nodes that have been registered with the Cluster Manager. Nodes can be sorted by various attributes by clicking on the corresponding column headers. Up and down arrow icons allow you to select the sorting direction.
2. Nodes displayed in the table can be filtered using the Search box.
3. The number of rows of the table.

   When the table has been filtered, two numbers are displayed to the right of the page title showing respectively the number of nodes matching the current filtering and the total number of nodes. If no nodes have been filtered, only the total number of nodes is displayed.
4. When you select a node from the table (highlighted in blue), the tabs below the table allow you to obtain additional information about that node. Choosing a tab brings up the corresponding information.
5. The **JOB LIMIT** button allows the user to *change the job limit for that node*.
6. The **STOP** button allows the user to *stop that node*.



Job Limit

The job limit for a node determines how many jobs can be executed simultaneously on that node. This limit can be changed using the JOB LIMIT button **(5)**. This button opens a dialog that allows you to specify a new limit:



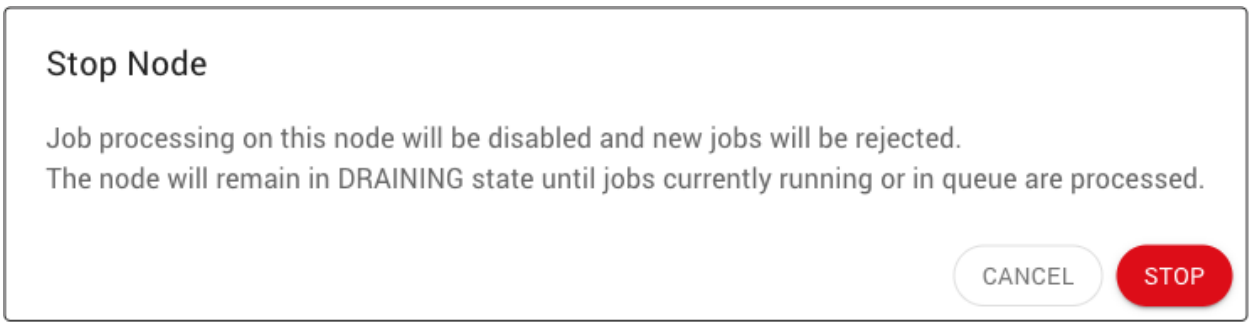
If you submit a job to a cluster where all nodes having reached their limits, the job is added to the *job queue*.

> **Note**
>
> Only users with `SYSADMIN` or `ADMIN` roles can change the job limit.

Stop / Restart

The Cluster Nodes page allows the user to stop or restart a node of the cluster. Stopping the node is done through the STOP button, which is available if the node is alive. When clicked, the button opens a dialog that requests confirmation:

After the node is stopped, the STOP button for the node is replaced by a START button, which allows the user to start the node again.



Note

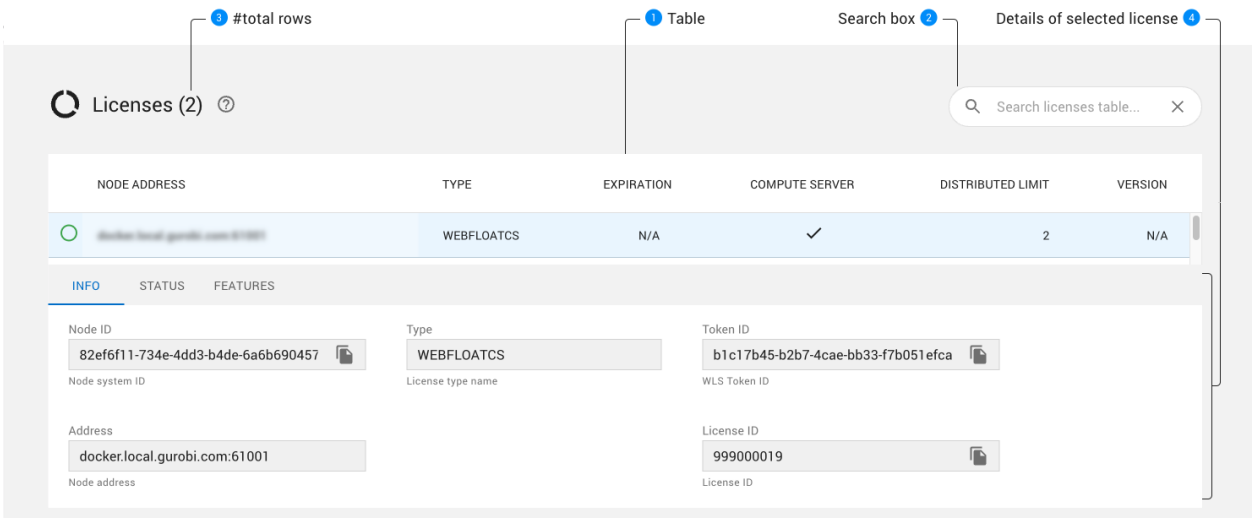
Only users with `SYSADMIN` or `ADMIN` roles can stop or restart a node.

Licenses

Note

Access to the Licenses page is restricted to `SYSADMIN` and `ADMIN` users.

The Licenses page provides details on the licenses being used for each node of the cluster.



1. The license table provides license information for each node.

   The `NODE ADDRESS` column allows you to identify the node that each line refers to.

2. Nodes displayed in the table can be filtered with the Search box.

3. The number of rows of the table.

   When the table has been filtered, two numbers are displayed to the right of the page title showing respectively the number of nodes matching the current filtering and the total number of nodes. If no nodes have been filtered, only the total number of nodes wil be displayed.

4. The tabs below the table provide more details about the selected license. Choosing a tab brings up the corresponding information.

4.3.5 Accounts

> **Note**
>
> Access to the Cluster Manager Users page is restricted to `SYSADMIN` users.

Users

To access the Cluster Manager, a user needs an account and the appropriate credentials to authenticate against that account.

Overview

User Roles

Cluster Manager users will have one of four user roles:

- **Standard user**

  A standard user submits jobs or batches to the cluster. This is done through a user application, the Gurobi command-line tools, or interactively using the Cluster Manager Web Interface.

- **Administrator**

  An administrator monitors and manages the flow of jobs and batches submitted by standard users. The administrator can abort or discard jobs or batches, change some cluster parameters, and check node licenses.

- **System administrator**

  The system administrator is in charge of managing user accounts and cluster nodes.

- **Read-only user** Read-only users can only monitor optimization tasks. They can list jobs and batches, access job history, display the log of a job, etc. They are not allowed to submit jobs or batches to the cluster, nor can they abort jobs.

Authentication Type

The Cluster Manager supports two types of authentication:

- Interactive login

  An interactive login requires the user to provide their username and password. Upon successful login, the server generates a session token which is valid for a relatively short period of time (default is 8 hours and can be changed in the Cluster Manager settings). The user will not be asked to log in again until this token has expired, which is handy when using the Web User Interface or the command-line tools. Interactive login can also be mapped to an LDAP server for centralized management. If SAML is configured this interactive login will be managed by the Identity provider configured.

- API keys

  When using an API key, the user or application must provide an access ID and a secret key. When creating an API key, you can specify an optional application name and description to help keep track of how the key is

being used. Once a key is created, you can download an associated client license file, which contains the API access ID, the secret key, and the Cluster Manager URL. This license file can be used by client applications and command-line tools to connect to the Cluster Manager. The Cluster Manager keeps track of the timestamp and IP address of the last API key usage.

The system administrator can enable or disable interactive login or API Key authentication, either at *creation time*, or it can be done later by *editing* the account properties. An account that only allows interactive login will not be allowed to create, use and manage API keys. An account that only allows API key authentication (known as a system account) can only be used for programmatic access through the REST API.

LDAP Integration

The Cluster Manager can be integrated with an LDAP repository. This integration can be configured in the Cluster Manager settings section. The system administrator can specify connection parameters (including the use of LDAPS for encrypted communication), account filtering, and account mapping. Once activated, users will be given access based on the user accounts defined in the LDAP server. Only accounts with a system administrator role will be able to log in using their local passwords.

System administrators need to log in using a special login page by following the link "System Administrator Log in" at the top of the main login page. They have local accounts that enable them to administer the cluster even if there is a problem with the LDAP configuration. System accounts (i.e., accounts with no interactive login) can always gain access using API keys.

The Cluster Manager continually synchronizes user accounts with the LDAP server in the background. If a given user account is no longer mapped to the LDAP filter in place, it will be disabled. In particular, if an account was created before the integration with the LDAP sever and if it is not mapped to the defined LDAP filter in place, it will be disabled. The same policy will apply during migration from local accounts to LDAP. Two important exceptions are system administrator accounts and system accounts, which will not be disabled for this reason.

Note that it may be necessary to rename and merge users when migrating an existing Cluster Manager setup from local authentication to LDAP authentication, to avoid losing API keys and job history when the username is adjusted to match the LDAP username.

SAML Integration

The Cluster Manager can be integrated with a SAML 2.0 directory. This integration can be configured in the Cluster Manager settings section. The system administrator can specify connection parameters and account mapping. Once activated, users will be given access based on the user accounts defined in the SAML directory. Accounts with a system administrator role can log in using their local passwords, which enables them to administer the cluster even if there is a problem with the SAML configuration. System administrators will need to log in using the special login page by accessing the `/login/admin` link at the top of the main login page. System accounts (i.e., accounts with no interactive login) can always gain access using API keys.

Once SAML 2.0 has been configured, the `SYSADMIN` can login using the page mentioned above with the local system password set in the Cluster Manager, or using the regular SAML Login flow if the `username` matches the user provided by the IdP.

If a user is enabled in SAML but disabled in cluster manager, a redirection to the error page will be performed and, a log in the system will indicate the cause.

After configuring SAML 2.0 as the Authentication Provider, the username attribute mapping will be employed to associate existing users. Any user not found in the system will be treated as a new user. If it is necessary for existing users to match the format of new usernames, system administrators can rename the users to the desired format in the user table.

Command line login for SAML with `grbcluster`

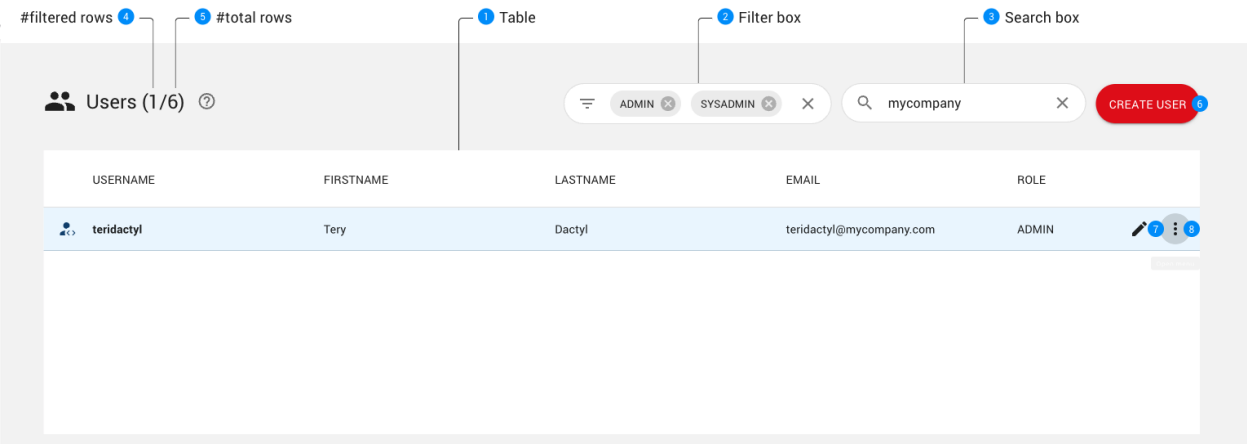
If the cluster manager is configured for SAML Authentication, it's essential to follow the authorization flow for secure access.

When you initiate the login process, the cluster manager will generate a unique URL along with a Device ID Code. By default, this action will automatically open a web browser for your convenience. However, if you wish to disable this automatic browser launch, you can use the `--no-browser` flag.

The login operation itself must be conducted within the web browser. It's important to clarify that this web browser doesn't have to be on the same machine where the login was initiated. You can access it from any device that has connectivity to the cluster manager. This flexibility ensures that you can perform the authorization operation from a location that is most convenient for you.

The Users Page

The Users page lists the users of this Cluster Manager.



1. This page shows a table of all batches that have been created on the Cluster Manager (created, aborted, and completed). Each line in the table provides information about a batch, including the current status, the creation and completion times, the priority, and the API used to create the job. The current batch status is shown using an icon:

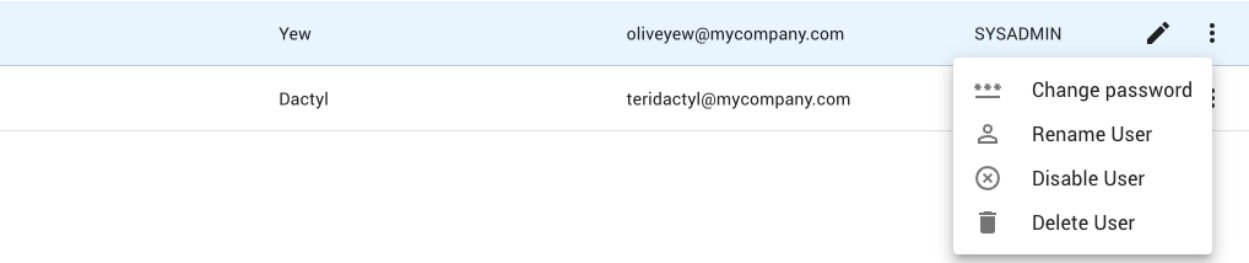
	User is enabled for interactive login and API keys.
	User is currently disabled, but could use interactive login and API keys if enabled.
	User is enabled for interactive login only.
	User is disabled, but could use interactive login if enabled.
	User is enabled for API key use only (system account).
	User is disabled, but could use API keys if enabled.

2. 3. Users displayed in the table can be filtered using either the Filter box or the Search box. Details can be found in the Filtering and Searching section.

4. 5. The number of rows of the table.

   When the table has been filtered, two numbers are displayed to the right of the page title showing respectively the number of users matching the current filtering and the total number of users. If no users have been filtered, only the total number of users wil be displayed.

6. The CREATE USER button opens a dialog to *create* a user.
7. The EDIT button opens a dialog that allows you to *edit information about the user*.
8. The MENU button opens a menu that gives access to various actions that can be performed on a user account (e.g., deleting or disabling the user).



Filtering and Searching

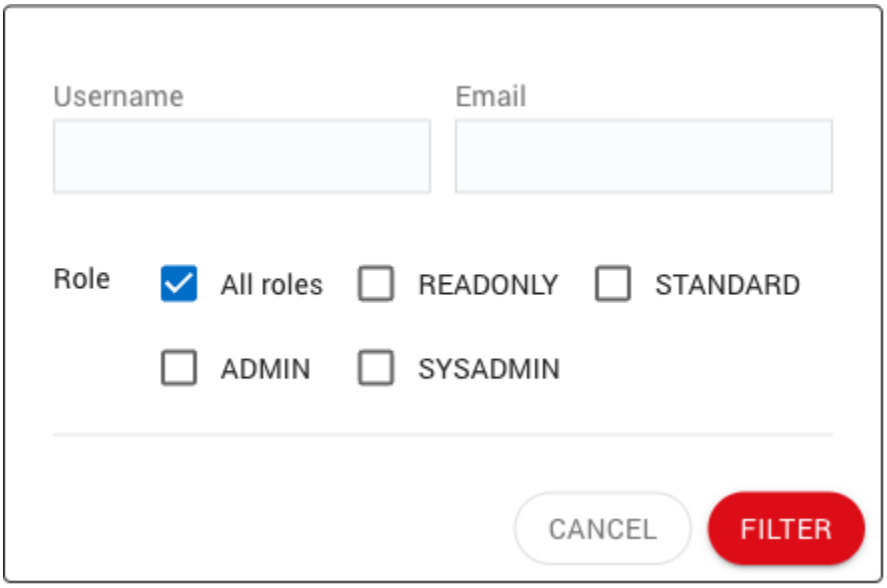
Users displayed in the table can be filtered using the Filter box and the Search box.

Filtering

The Filter box provides the user with a set of predefined filters as shown below:



When clicking on the left icon of the filter box, a dialog is displayed to edit the filters to apply.



Searching

The Search box enables users to further refine the results by specifying an arbitrary search string. In order for a user to be displayed, the search string must be present in at least one of the columns in the table for that user.

> **Note**
>
> It is important to note that filtering with the Filter box is usually applied server side, while searching with the Search box is always performed client side. That means that the searching is done on results that were already filtered by the server, so it it is preferable to use the Filter box first and then use the Search box to refine the results. Using the Search box as an alternative to the Filter box could result in missing items.

Synchronization with page URL

The Cluster Manager creates permalinks for all tables that can be filtered or searched. Changes applied using either the Filter or Search boxes are automatically reflected in the page URL, thus allowing you to copy and paste the URL to easily retrieve the same display later. For example, the URL `http://localhost:61080/accounts/users?role=ADMIN` would display only users with ADMIN roles (assuming the Cluster Manager is running on `localhost:61080`).

Creating a User

The CREATE USER button **(6)** opens a dialog that allows you to provide the necessary properties for a new user and create an account.

Any property except the Username can be *edited* after the account has been created. Usernames must be unique among all users registered to a Cluster Manager.

Editing a User

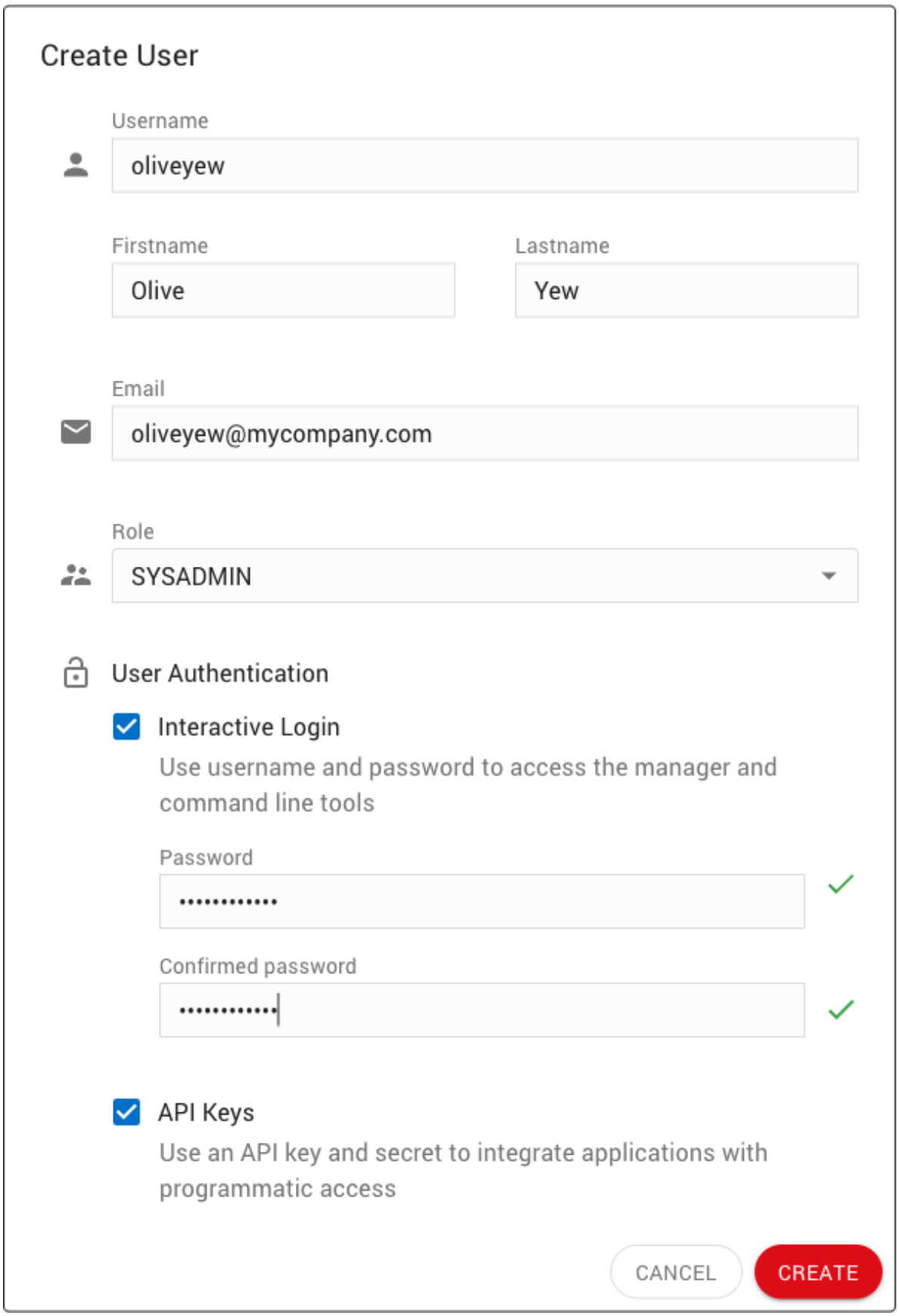
User properties can be edited using the EDIT button **(7)**.

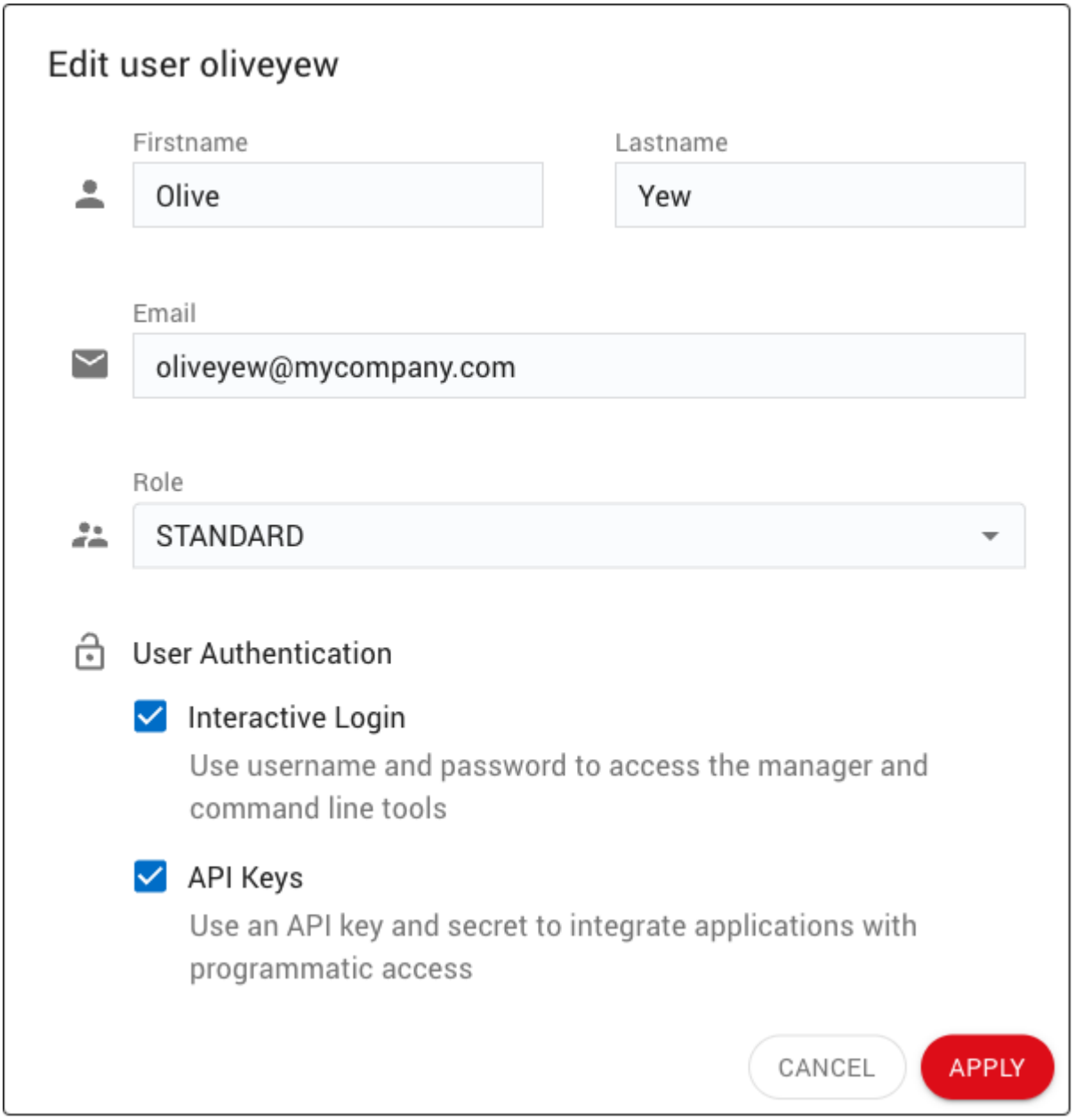
Deleting a User

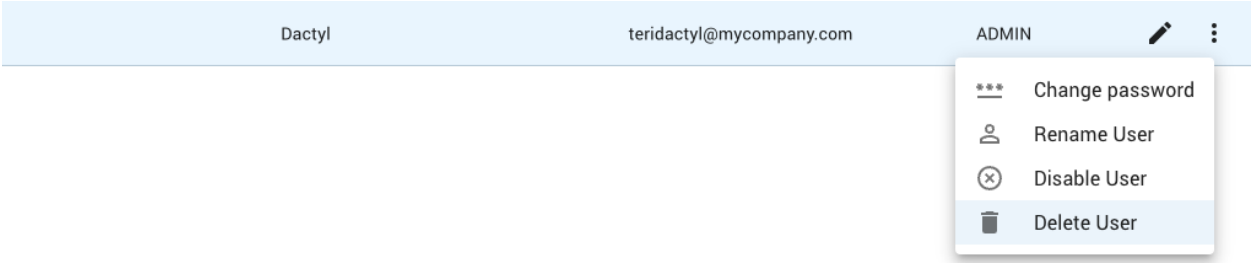
System administrators can delete a user:

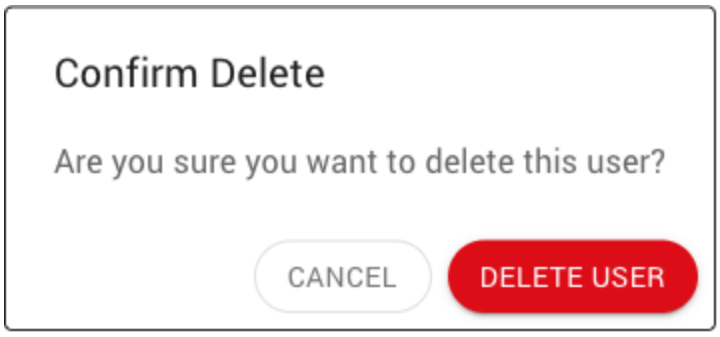
- Click on the menu button **(8)** for the user and select the Delete User menu item.
- Confirm the deletion.

> **Warning**
>
> Deleting a user is permanent and cannot be undone.


Create User
Username
oliveyew
Firstname
Olive
Lastname
Yew
Email
oliveyew@mycompany.com
Role
SYSADMIN
User Authentication
Interactive Login
Use username and password to access the manager and command line tools
Password
Confirmed password
API Keys
Use an API key and secret to integrate applications with programmatic access
CANCEL
CREATE


Edit user oliveyew
Firstname
Olive
Lastname
Yew
Email
oliveyew@mycompany.com
Role
STANDARD
User Authentication
Interactive Login
Use username and password to access the manager and command line tools
API Keys
Use an API key and secret to integrate applications with programmatic access
CANCEL
APPLY

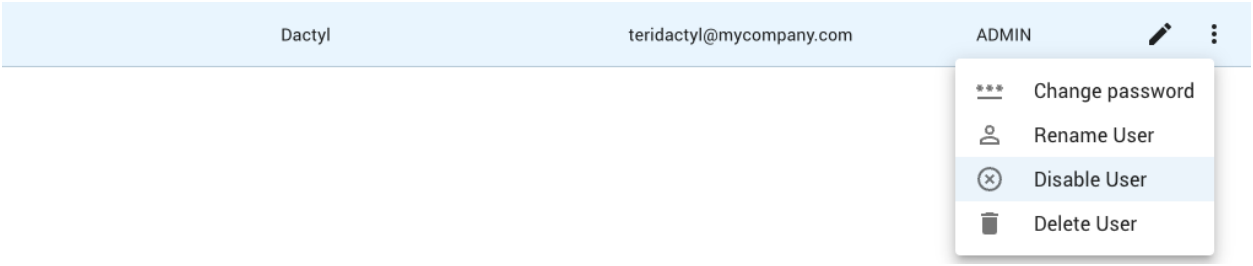

Dactyl
teridactyl@mycompany.com
ADMIN
Change password
Rename User
Disable User
Delete User



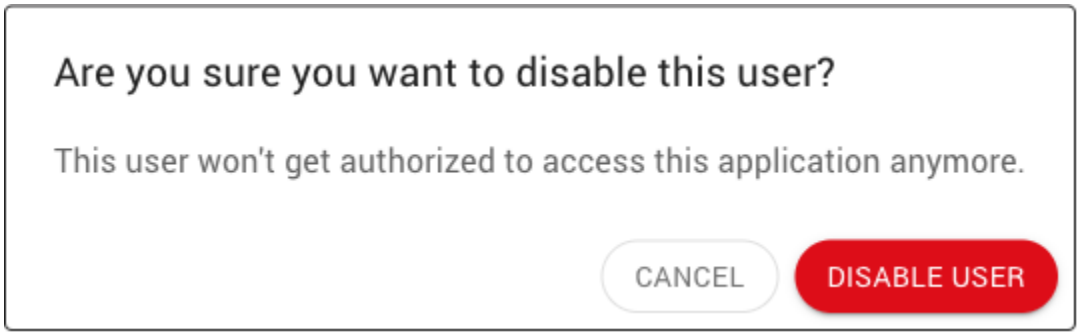
Disabling a User

Disabling a user temporarily revokes all access from that user (interactive login and API keys). To disable a user:

- Click on the menu button **(8)** for the user and select the Disable User menu item.



- Confirm the user can be disabled.

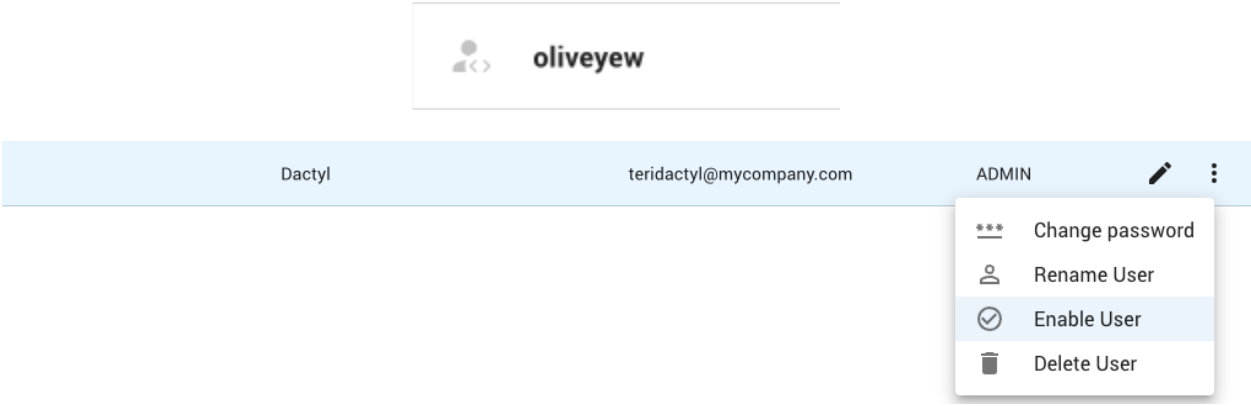


- Once the user has been disabled, the authentication type icon will be grayed out:

Enabling a User

A system administrator can follow similar steps to enable a user:

- Click on the menu button **(8)** for the user and select the Enable User menu item.
- Once the user has been enabled, the authentication type icon will no longer be grayed out:



Changing User Password

When *creating* a new user, the system administrator must define a password to enable the user to interactively log in to the Cluster Manager. Users can later change their passwords from their profile page. A system administrator can also change a user password (e.g., if the user forgot their password):

- Click on the menu button **(8)** for the user and select the Change Password menu item.
- Enter a new password for the user.

  If the password does not comply with *password policy*, the Apply button will be disabled, and hints will be displayed in red for fixing the problem.

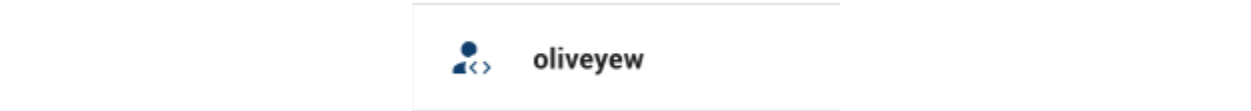
> **Note**
>
> Passwords must comply with the chosen Password Policy for the Cluster Manager, as defined in the *Password Policy* settings.

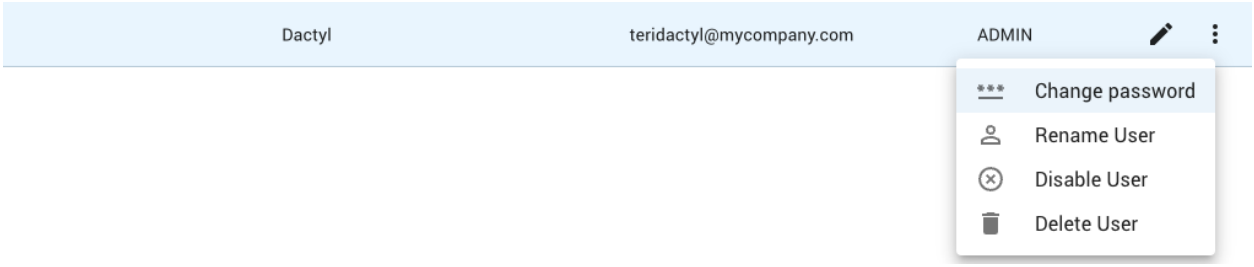
Renaming a User

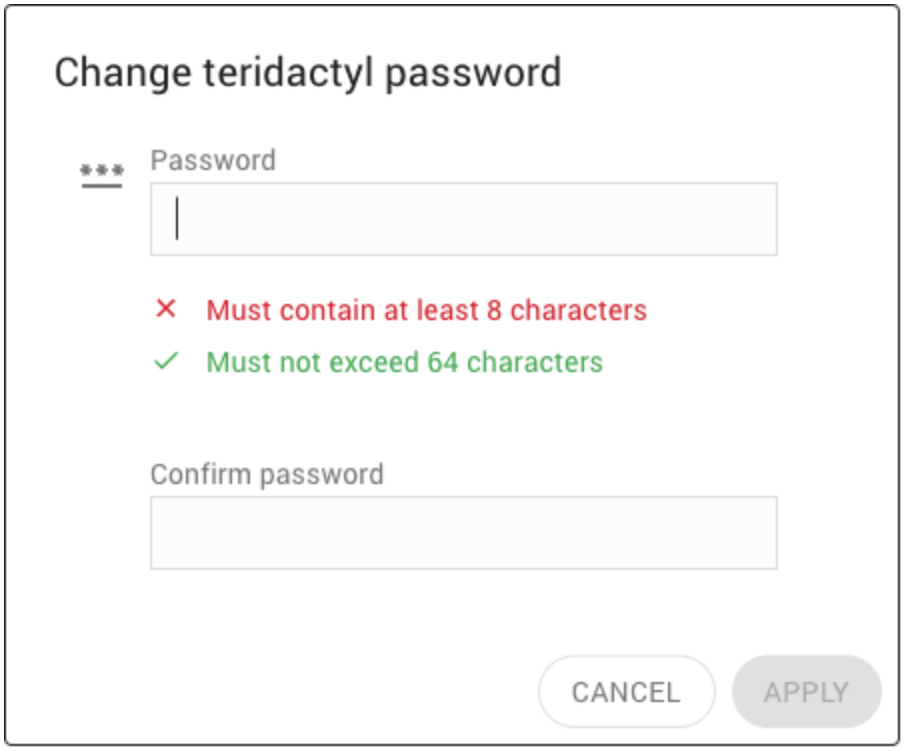
The system administrator can give a user a new username. That changes the username of that user permanently. When renaming a user, all the API keys, job history and batch history will be owned by the new username. If a user is renamed and there is already a user with that name, the Cluster Manager will give the option to merge the two accounts. That merged user would then own all of the API keys, job history and batch history of the original user.

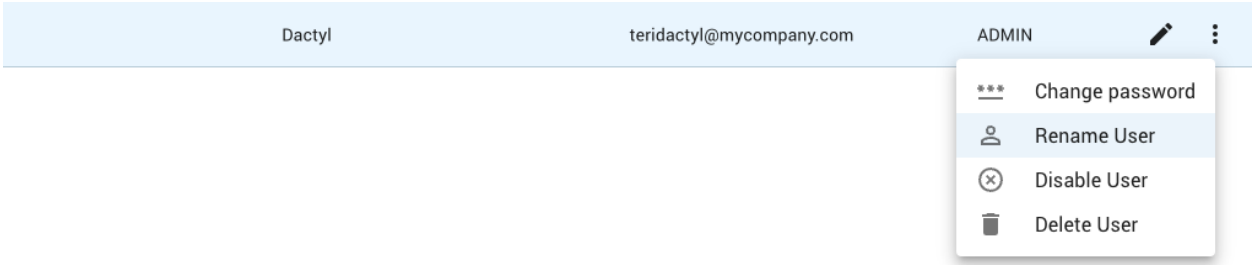
To rename a user:

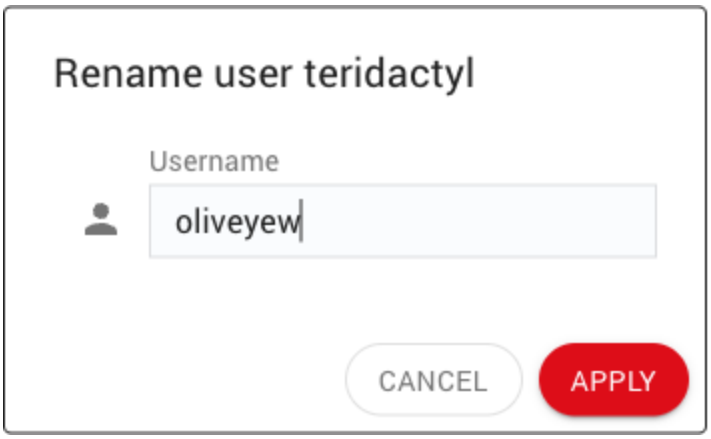
- Click on the menu button **(8)** for the user and select the Rename User menu item.
- In the dialog that pops up, enter a username for the new user.
- If the specified username matches the username of an existing user, a dialogs pops up to confirm that the two accounts should be merged:

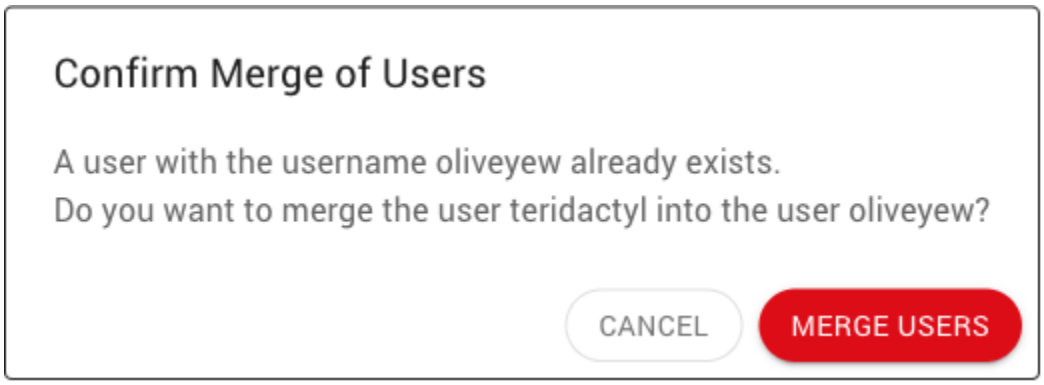



Dactyl
teridactyl@mycompany.com
ADMIN
Change password
Rename User
Disable User
Delete User


Change teridactyl password
Password
Must contain at least 8 characters
Must not exceed 64 characters
Confirm password
CANCEL
APPLY


Dactyl
teridactyl@mycompany.com
ADMIN
Change password
Rename User
Disable User
Delete User





API Keys

An API key (Application Programming Interface key) is composed of an access ID and a secret key, API keys are the recommended approach for connect applications to the Cluster Manager. When the Cluster Manager authenticates access using an API key, all actions are performed on behalf of the user who owns that key. There are two ways to create an API key:

- The user can create their own API key from the *API keys* page.
- The system administrator can create an API key on the behalf of a user from the system administrator *Accounts API keys* page.

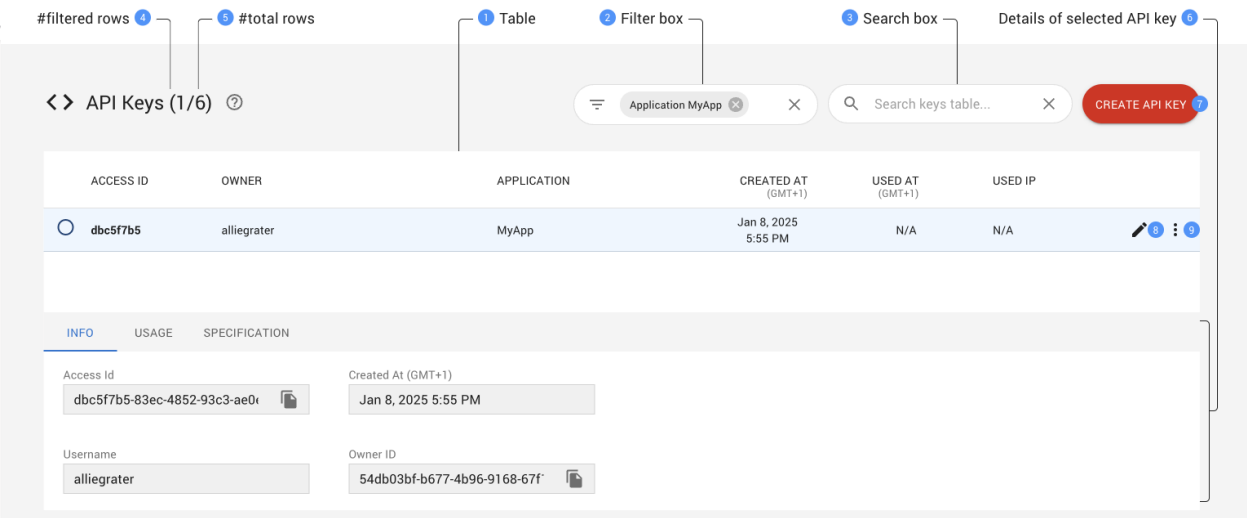
When creating an API key, you can specify an optional application name and description to help keep track of how the key is being used. Once a key is created, you can download an associated client license file, which contains the API access ID, the secret key, and the Cluster Manager URL. This license file can be used by client applications and command-line tools to connect to the Cluster Manager. The Cluster Manager keeps track of the timestamp and IP address of the last API key usage.

An API key can be enabled or disabled by either the owner or the system administrator.

This set of API key features was designed to simplify the task of monitoring API keys, detecting unwanted usage, and safely rotating keys by disabling previous keys before permanently deleting them.

The API Keys Page

The API Keys page allows the system administrator to manage keys owned by users of the Cluster Manager.



1. This page provides a table of API keys. Keys can be sorted by various attributes by clicking on the corresponding column headers. Up and down arrow icons allow you to select the sorting direction.

   The icon in the first column shows the status of each API key:

	API key is enabled and can be used to access the Cluster Manager API.
	API key is disabled and cannot be used to access the Cluster Manager API.

2. 3. API keys displayed in the table can be filtered either using the Filter box or the Search box. Details can be found in the *Filtering and Searching* section.

4. 5. The number of rows of the table.

   When the table has been filtered, two numbers are displayed to the right of the page title showing respectively the number of API keys matching the current filtering and the total number of API keys. If no users have been filtered, only the total number of API keys wil be displayed.

7. The CREATE API KEY button opens a dialog to *create an API key*.
8. The EDIT button opens a dialog to *edit the API key*.
9. The MENU button opens a menu that gives access to various actions that can be performed on the API key (e.g., deleting or disabling the key).

Filtering and Searching

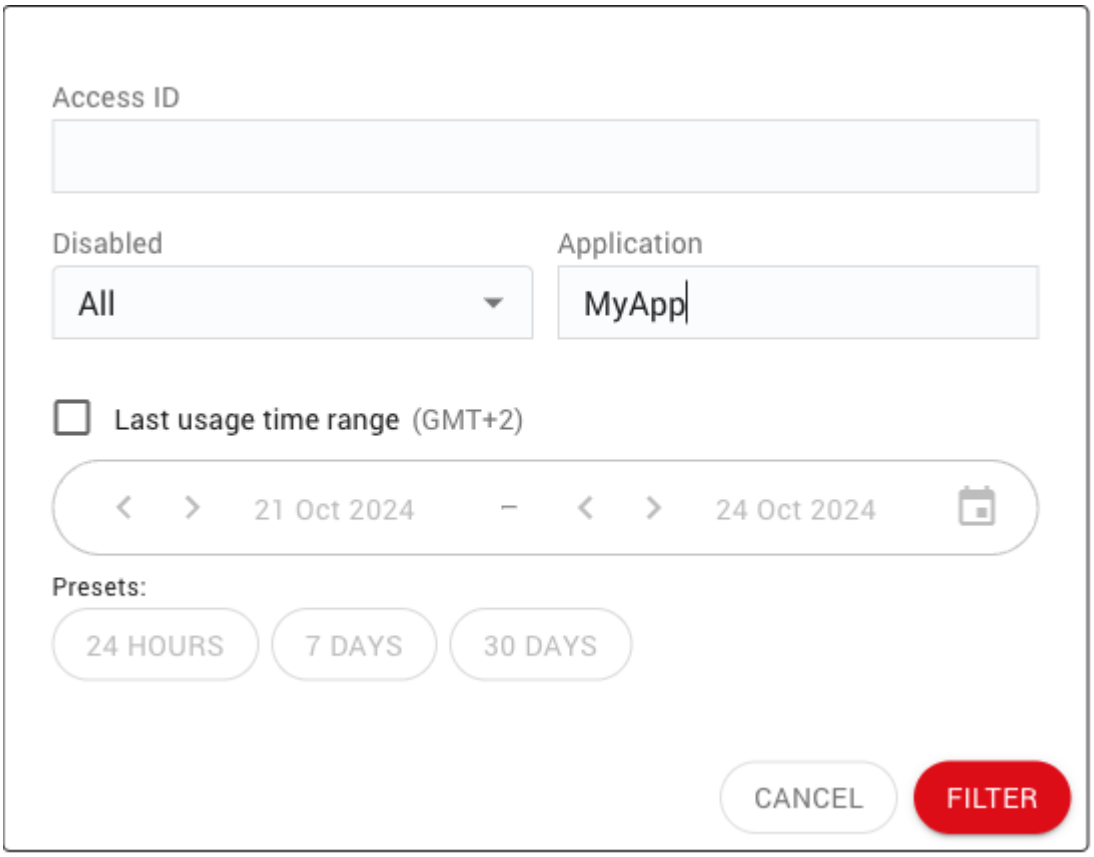
API keys displayed in the table can be filtered using the Filter box and the Search box.

Filtering

The Filter box provides the user with a set of predefined filters as shown below:



When clicking on the left icon of the filter box, a dialog is displayed to edit the filters to apply.



Searching

The Search box enables users to further refine the results by specifying an arbitrary search string. In order for an API key to be displayed, the search string must be present in at least one of the columns in the table for that key.

Note

It is important to note that filtering with the Filter box is usually applied server side, while searching with the Search box is always performed client side. That means that the searching is done on results that were already filtered by

the server, so it it is preferable to use the Filter box first and then use the Search box to refine the results. Using the Search box as an alternative to the Filter box could result in missing items.

Synchronization with page URL

The Cluster Manager creates permalinks for all tables that can be filtered or searched. Changes applied using either the Filter or Search boxes are automatically reflected in the page URL, thus allowing you to copy and paste the URL to easily retrieve the same display later. For example, the URL `http://localhost:61080/accounts/keys?appname=MyApp` would display only API keys associated with the MyApp application (assuming the Cluster Manager is running on `localhost:61080`).

Creating an API Key

The CREATE API KEY button **(7)** opens a dialog that allows you to provide the necessary properties and create a new API key.

Editing an API Key

The EDIT button **(8)** opens a dialog that allows you edit the properties of an API key:

Deleting an API Key

System administrators can delete an API key:

- Click on the menu button **(9)** for the API key and select the Delete API Key menu item.
- Confirm the API key can be deleted.

Disabling an API Key

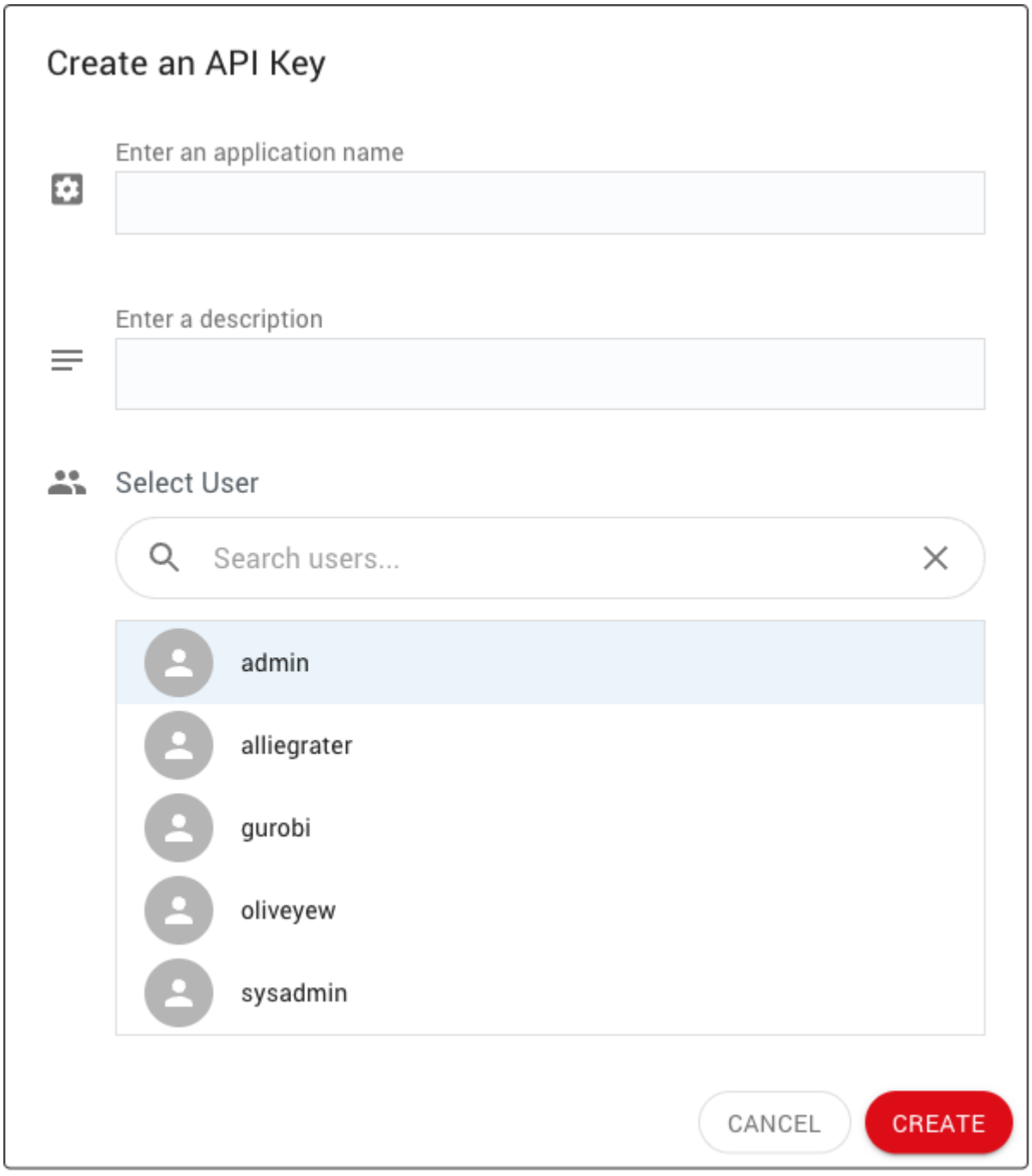
System administrators can also disable an API key:

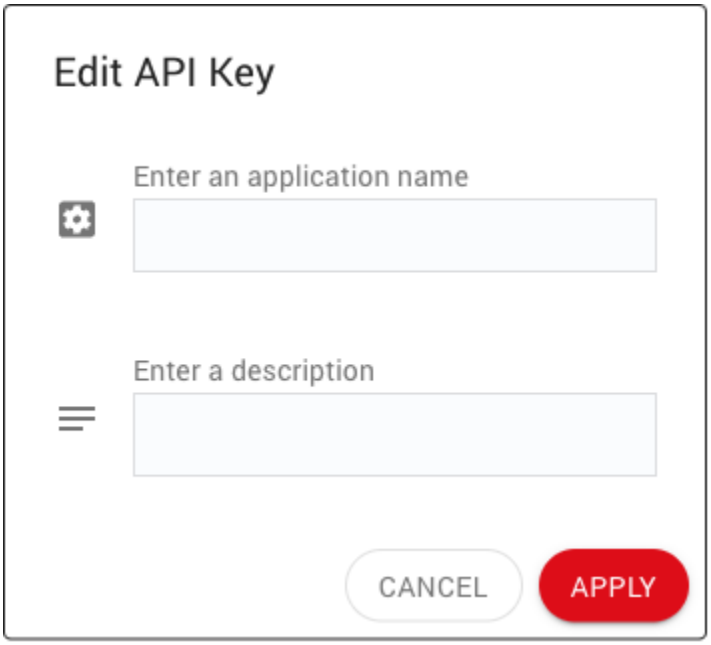
- Click on the menu button **(9)** for the API key and select the Disable API Key menu item.
- Confirm the API key can be disabled.
- Once the API key has been disabled, the icon in the first column will be grayed out:

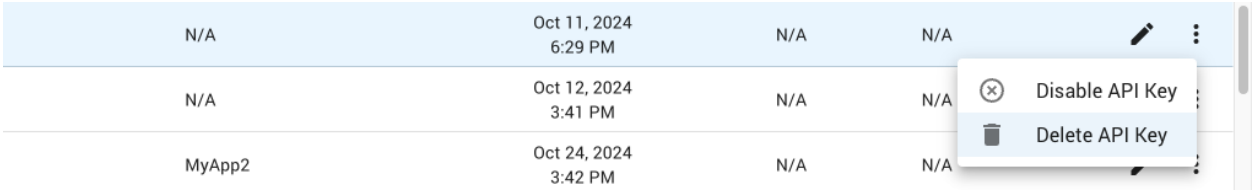
Enabling an API Key

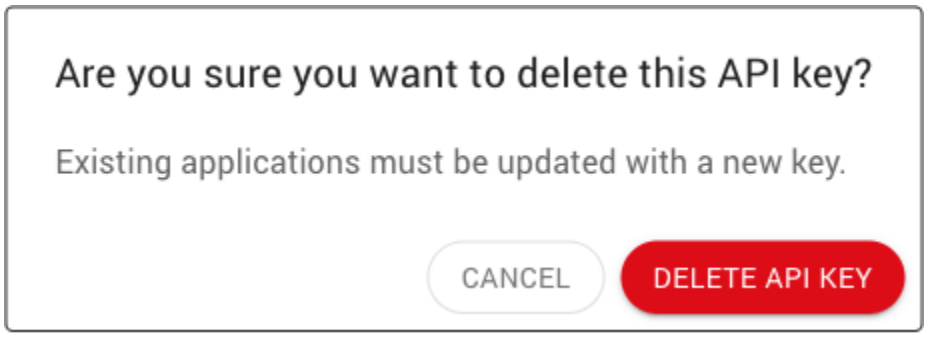
The system administrator can follow similar steps to enable an API key:

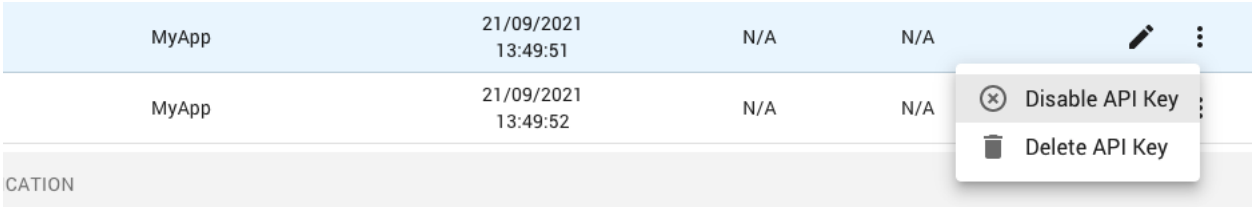
- Click on the menu button **(9)** for the API key and select the Enable API Key menu item.
- Once the API key has been enabled, the icon in the first column will no longer be grayed out:

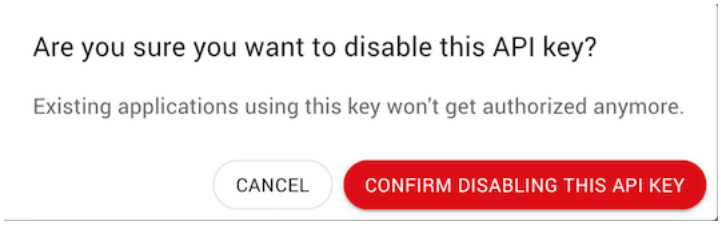

Create an API Key
Enter an application name
Enter a description
Select User
Search users...
admin
alliegrater
gurobi
oliveyew
sysadmin
CANCEL
CREATE

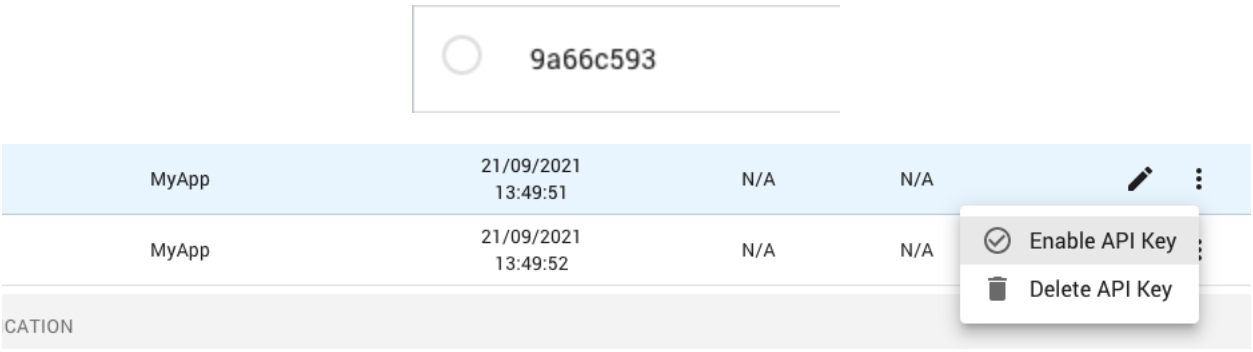

Edit API Key
Enter an application name
Enter a description
CANCEL
APPLY


N/A
Oct 11, 2024
6:29 PM
N/A
N/A
N/A
Oct 12, 2024
3:41 PM
N/A
N/A
MyApp2
Oct 24, 2024
3:42 PM
N/A
N/A
Disable API Key
Delete API Key


Are you sure you want to delete this API key?
Existing applications must be updated with a new key.
CANCEL
DELETE API KEY


MyApp
21/09/2021
13:49:51
N/A
N/A
MyApp
21/09/2021
13:49:52
N/A
N/A
CATION
Disable API Key
Delete API Key


Are you sure you want to disable this API key?
Existing applications using this key won't get authorized anymore.
CANCEL
CONFIRM DISABLING THIS API KEY



4.3.6 Settings

> **Note**
>
> Access to Cluster Manager settings is restricted to `SYSADMIN` users.

Security

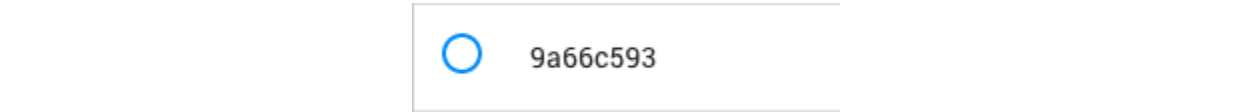
Password Policy

The Cluster Manager supports password policies to ensure that passwords for local accounts match configurable security standards. With a password policy, the system administrator can specify the minimum length of a password and can require that a password contain a mix of upper and lower case characters, digits, or symbols. Any changes to a policy will apply only when new passwords are created; previously created password will remain valid.

Password Requirements

Password requirements allow the system administrator to define a set of rules that enforce password strength. Weak passwords may result in unauthorized access and/or exploitation of the application's resources.

	Password Length Password length must be between the given minimum and maximum values. Strong password should contain at least 8 characters.
aA	Character Case Indicates whether passwords must contain a mix of upper and lower case characters (e.g., a-z, A-Z). Strong password should contain both.
01	Digits Optionally requires passwords to contain at least one number (e.g., 0-9). Strong password should contain at least one digit.
!@	Symbols Optionally requires passwords to contain at least one symbol (e.g., `!@#$%^&*()_+|~-=`{}[]:";'<>?,./`). Strong password should contain at least one symbol.



Account Lockout

When activated, this option enables the system administrator to define a maximum number of failed login attempts before the account is locked. When a user account has been locked, a message will appear on the login page, and the user will be asked to contact a system administrator. The system administrator can then change the user's password to unlock their account.

Authentication

SAML 2.0 Configuration

The Cluster Manager can be integrated with a SAML 2.0 directory. The system administrator can specify connection parameters and account mapping. Once activated, users will be given access based on the user accounts defined in the SAML directory. Accounts with a system administrator role can log in using their local passwords, which enables them to administer the cluster even if there is a problem with the SAML configuration. System administrators will need to log in using the special login page by accessing the `/login/admin` link at the top of the main login page. System accounts (i.e., accounts with no interactive login) can always gain access using API keys.

Once SAML 2.0 has been configured, the `SYSADMIN` can login using the page mentioned above with the local system password set in the Cluster Manager, or using the regular SAML Login flow if the `username` matches the user provided by the IdP.

If a user is enabled in SAML but disabled in cluster manager, a redirection to the error page will be performed and, a log in the system will indicate the cause.

After configuring SAML 2.0 as the Authentication Provider, the username attribute mapping will be employed to associate existing users. Any user not found in the system will be treated as a new user. If it is necessary for existing users to match the format of new usernames, system administrators can rename the users to the desired format in the user table.

When selecting the ID Format from the Identity Provider, make sure to select one of the following:

- Persistent: `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
- Email: `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`

Connection

Identity Provider SSO URL	This is the URL where the Identity Provider provides the endpoint to send authentication requests. • Azure: When using Microsoft Azure AD, please use the SAML-P sign-on endpoint `https://login.microsoftonline.com/[TenantIDGUID]/saml2` • JumpCloud: Use the IDP URL defined within the SSO settings `https://sso.jumpcloud.com/saml2/[custom identifier]` • Google: Use the SSO URL provided in the Google Identity Provider details `https://accounts.google.com/o/saml2/idp?idpid=[idpId]` • Okta: Use the provided SSO URL when the Integration has been created
Identity Provider ID	The Identity Provider ID identifies the authentication server in the SAML 2.0 protocol. The Cluster Manager will make sure that authentication response are issued by a provider matching this ID, otherwise the authentication request will be rejected. • Azure: when using Microsoft Azure AD, please use the tenant identity URI `https://sts.windows.net/[TenantIDGUID]/` • JumpCloud: use the IdP Entity ID defined within the SSO settings `https://gurobi.contoso.com` • Google: use the Entity ID provided in the Google Identity Provider details `https://accounts.google.com/o/saml2/?idpid=[idpId]` • Okta: use the provided IdP ID generated when the Integration has been created
Request Binding	Preferred method to perform the authentication request from the Cluster Manager to the Identity Provider. Azure, JumpCloud, Google and Okta does support any of the following values. • HTTP-Redirect: the SAML request is encoded and sent as URL query parameters in an HTTP GET request. This is the default binding method. • HTTP-POST: the SAML request is sent as a form within the body of an HTTP POST request.
Service Provider ID	The Service Provider ID identifies the Cluster Manager in the SAML protocol. It is passed as the `Issuer` when sending the authentication request to the Identity Provider. The Identity Provider also includes this ID as the Audience in the authentication response. The cluster Manager will make sure that the `audience` matches the specified value, otherwise the authentication request will be rejected. The Cluster Manager generates a default value based on the URL where it is deployed, for example: `https://gurobi.contoso.com`. • Azure: when using Microsoft Azure AD, please use the `Application ID URI`. • JumpCloud: the property can be assigned in the `SP Entity ID`. • Google: the property can be assigned in the `Entity ID`. • Okta: the property can be assigned in the `Audience URI (SP Entity ID)`.
Cluster Manager base URL	This field indicates the base URL to receive the authentication responses. It must specify the base URL that can be accessed from the browser when the identity provider has authenticated a user using a POST command. The Cluster Manager generates a default value based on the URL where it is deployed, for example `https://gurobi.contoso.com:61080`. The complete endpoint URL consists of this base URL and the `/api/v1/saml2/callback` path. When configuring the Identity Provider, please use the complete URL. When configuring the Identity provider: • Azure: add a Web platform with the redirect URI as the complete endpoint URI. • JumpCloud: the property can be assigned in the `ACS URLs`. • Google: the property can be assigned in the `ACS URL`. • Okta: the property can be assigned in `Single sign-on URL`.

Mapping

A claim is a piece of information about a user or entity. Claims are used to represent various attributes, such as name, email address, first name, last name and username. The mapping configuration is used to match the claims coming from the Identity Provider into Cluster Manager. `Email` and `Username` are mandatory mappings, and usually they can be the same claim.

Email	SAML claim for the Email • Azure: When using Microsoft Azure AD, please use the identity claim for the name attribute `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name` • JumpCloud, Google and Okta: the email claim is called `email`.
Username	SAML claim for the username, it should be unique. In many instances, the email assertion serves as an attribute to uniquely identify users.
First name	SAML claim for the first name • Azure: When using Microsoft Azure AD, please use the identity claim for the given name `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname` • JumpCloud, Google and Okta: the claim is called `firstName`.
Last name	SAML assertion for the last name • Azure: When using Microsoft Azure AD, please use the identity assertions for the surname `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname` • JumpCloud, Google and Okta: the assertion is called `lastName`.

Certificates

When the Identity Provider replies to the authenticated request, claims are returned as digitally signed XML assertions. In order to ensure response integrity, you can include the digital certificates provided by the IdP used to verify the assertions. The certificates must adhere to the X.509 standard.



In order to add a new certificate click on this button and copy and paste the certificate provided by your identity provider. When more than one certificate is required, new field for certificates can be added clicking on the same button.



If a certificate field was added by mistake, it can be removed by clicking on the trash bin icon.

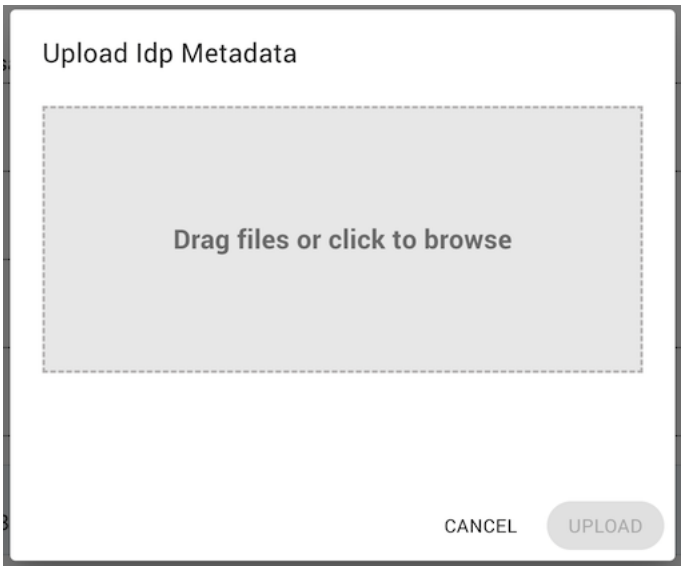
The Identity Provider has established expiration dates for the Certificates, making it crucial to regularly verify these expirations and keep the Certificates list in the settings up-to-date.

Upload IdP metadata



The Identity Provider (IdP) typically provides a metadata `XML` file that contains essential information about the IdP's configuration and capabilities. This XML can be used to automatically configure the following properties, if they are provided: `Identity Provider SSO URL`, `Identity Provider ID`, `Request Binding` and `Certificates`.

To upload a metadata file, click on this button. A new dialog will be displayed, allowing you to upload a valid `XML` file.



After selecting a metadata file, click on `UPLOAD`. If the metadata file is valid, the information will be loaded into the corresponding fields. Otherwise, a specific error message will be displayed.

Download Metadata



Some Identity providers, such as JumpCloud, support the upload of a Service Provider XML Metadata file. The Cluster Manager provides the capability to generate this file by clicking on this button. The following information is generated based on the SAML Protocol:

- `EntityDescriptor - entityID (Service Provider ID)`
- `Consumer service - location (Cluster Manager base URL + /api/vi/saml2/callback)`
- `Consumer service - binding (urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST)`
- `AttributeConsumingService - RequestedAttributes (firstName, lastName and email)`

View Metadata configuration



If the Identity Provider does not support uploading a metadata file, the properties mentioned above can be reviewed and copied manually by clicking on this button.

Case sensitivity

Configure case sensitivity for SAML configuration.

Disable username case sensitivity	Defines if usernames are case-sensitive. By default, case sensitivity is enabled so that users with different uppercase and lowercase letters are handled as different users. For example, `JohnDoe` and `johndoe` are different users by default. If case sensitivity is disabled, usernames will match even if some letters are uppercase or lowercase. For example, `JohnDoe` and `johndoe` will be the same user.
Validate case sensitivity	When disabling the case-sensitivity configuration, you need to make sure that existing usernames do not generate conflicts. For example, if you had `JohnDoe` and `johndoe` as different users, then disabling case-sensitivity will lead to conflicts with these users. This action allows you to run a validation check that will report if duplicates are detected. In this case, rename or merge the users accordingly before changing the configuration. If there is no conflicts, the cluster manager will proceed with the migration of users to be case-insensitive when saving the settings, and this operation cannot be undone. 

Logout

Configure logout for SAML configuration.

Logout page URL	Defines the URL of the page to redirect to when the user logs out. If not defined, the user is redirected to `/logout`.

Command line login for SAML with `grbcluster`

If the cluster manager is configured for SAML Authentication, it's essential to follow the authorization flow for secure access.

When you initiate the login process, the cluster manager will generate a unique URL along with a Device ID Code. By default, this action will automatically open a web browser for your convenience. However, if you wish to disable this automatic browser launch, you can use the `--no-browser` flag.

The login operation itself must be conducted within the web browser. It's important to clarify that this web browser doesn't have to be on the same machine where the login was initiated. You can access it from any device that has connectivity to the cluster manager. This flexibility ensures that you can perform the authorization operation from a location that is most convenient for you.

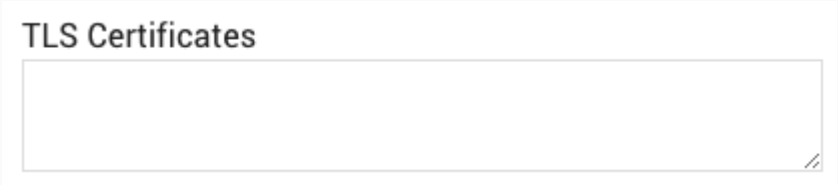
LDAP Configuration

The Cluster Manager can be integrated with a Lightweight Directory Access Protocol (LDAP) repository. The system administrator can specify connection parameters (including the use of LDAPS for encrypted communication), account filtering, and account mapping. Once activated, users will be given access based on the user accounts defined in the LDAP server.

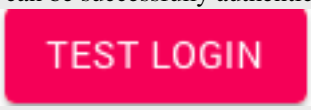
Accounts with a system administrator role can log in using their local passwords, which enables them to administer the cluster even if there is a problem with the LDAP configuration. System administrators will need to log in using the special login page by following the "System Administrator Log in" link at the top of the main login page. System accounts (i.e., accounts with no interactive login) can always gain access using API keys. The Cluster Manager continually synchronizes user accounts with the LDAP server in the background.

If a given user account is no longer mapped to the LDAP filter in place, it will be disabled. In particular, if an account was created before the integration with the LDAP server and if it is not mapped to the defined LDAP filter in place, it will be disabled. The same policy will apply during migration from local accounts to LDAP. Two important exceptions are system administrator accounts and system accounts, which will not be disabled for this reason.

Connection

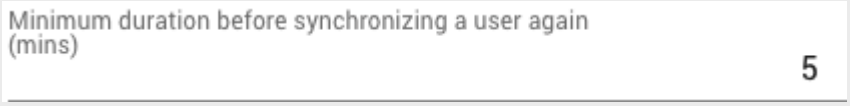
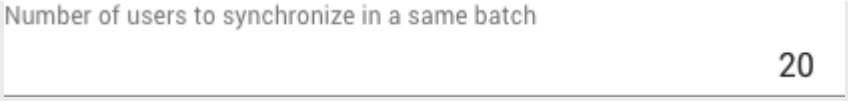
LDAP Server host	Specifies the Host name or IP address of your organization's LDAP server.
LDAP Server port	The LDAP server port. The default port is `636` if LDAPS is selected, or `389` otherwise.
Bind DN and password	The admin Bind DN and password, which allow the LDAP connection to gain access to the directory. Ex: `CN=Administrator,CN=Users,DC=mycompany,DC=com`
Encryption - LDAPS	LDAPS is the "LDAP over SSL" protocol, which communicates over a secure port such as 636. It establishes a secure connection before any communication is performed with the LDAP server. Check the `Use LDAPS` checkbox to enable encryption. When using encryption, a public or private SSL certificate must be provided in the TLS Certificate field:  This is used to set up the LDAPS directory. The Disable Certificate Validation box determines whether LDAPS will validate certificate signatures. When activated, no validation will be performed, which is insecure.
Test connection	 The TEST CONNECTION button is used for testing your parameters to the LDAP server. If you cannot connect to the LDAP server, user mapping and filtering will fail as well.

Filters and Mapping

Base DN	The base DN subtree that is used when searching for user entries on the LDAP server.
Filter template	Filter for finding entries in the LDAP base DN (groups) subtree that match the group name. The `'%s'` string must be included to indicate where the provided user entry should be found. Ex: `(&(uid=%s)(memberOf=cn=Technical Users \(Dev/Support/IT),ou=Users,dc=company,dc=com))`
Unique Username Attribute	Used for mapping the unique username identifier. If no unique username attribute is specified, it will default to using `uid`. Alternatively, you can specify other values such as `sAMAccountName` for use with Microsoft Active Directory. It's important that the attribute name matches the one used in the filter template. Ex: `sAMAccountName`
Firstname and Lastname	LDAP attributes for user first name and last name. Firstname ex: `'givenName'` Lastname ex: `'sn'`
Email Attribute	LDAP attribute for user email. Ex: `['mail','email','userPrincipalName']`
Test Mapping	To make sure that user IDs, real names, email addresses, and groups have been mapped correctly, click on the `TEST MAPPING` button. 
Test Login	As a final test, to ensure users can log in, the TEST LOGIN button prompts for a username and password and ensures that this user can be successfully authenticated with the LDAP server. 

Synchronization

Configure synchronization between the LDAP server database and the Cluster Manager database. The synchronization with the LDAP server is a background process that runs at the given frequency. At each iteration, it will synchronize a given number of users, i.e. batch size, that have not been synchronized for at least the given minimum duration (Minimum age). These parameters help tune the synchronization to avoid overloading the LDAP server with too many requests.

Minimum Age	 Accounts already synchronized within this time limit will not be synchronized.
Frequency	 Frequency of the background synchronization.
Batch	 The field above defines the synchronization batch size.

Case sensitivity

Configure case sensitivity for LDAP configuration.

Disable username case sensitivity	Defines if usernames are case-sensitive. By default, case sensitivity is enabled so that users with different uppercase and lowercase letters are handled as different users. For example, `JohnDoe` and `johndoe` are different users by default. If case sensitivity is disabled, usernames will match even if some letters are uppercase or lowercase. For example, `JohnDoe` and `johndoe` will be the same user.
Validate case sensitivity	When disabling the case-sensitivity configuration, you need to make sure that existing usernames do not generate conflicts. For example, if you had `JohnDoe` and `johndoe` as different users, then disabling case-sensitivity will lead to conflicts with these users. This action allows you to run a validation check that will report if duplicates are detected. In this case, rename or merge the users accordingly before changing the configuration. If there is no conflicts, the cluster manager will proceed with the migration of users to be case-insensitive when saving the settings, and this operation cannot be undone. 

Logout

Configure logout for SAML configuration.

Logout page URL	Defines the URL of the page to redirect to when the user logs out. If not defined, the user is redirected to `/logout`.

Local Configuration

The authentication configuration for local authentication managed by the Cluster Manager.

Case sensitivity

Configure case sensitivity for LDAP configuration.

Disable username case sensitivity	Defines if usernames are case-sensitive. By default, case sensitivity is enabled so that users with different uppercase and lowercase letters are handled as different users. For example, `JohnDoe` and `johndoe` are different users by default. If case sensitivity is disabled, usernames will match even if some letters are uppercase or lowercase. For example, `JohnDoe` and `johndoe` will be the same user.
Validate case sensitivity	When disabling the case-sensitivity configuration, you need to make sure that existing usernames do not generate conflicts. For example, if you had `JohnDoe` and `johndoe` as different users, then disabling case-sensitivity will lead to conflicts with these users. This action allows you to run a validation check that will report if duplicates are detected. In this case, rename or merge the users accordingly before changing the configuration. If there is no conflicts, the cluster manager will proceed with the migration of users to be case-insensitive when saving the settings, and this operation cannot be undone. 

Logout

Configure logout for SAML configuration.

Logout page URL	Defines the URL of the page to redirect to when the user logs out. If not defined, the user is redirected to `/logout`.

System

System - Authentication

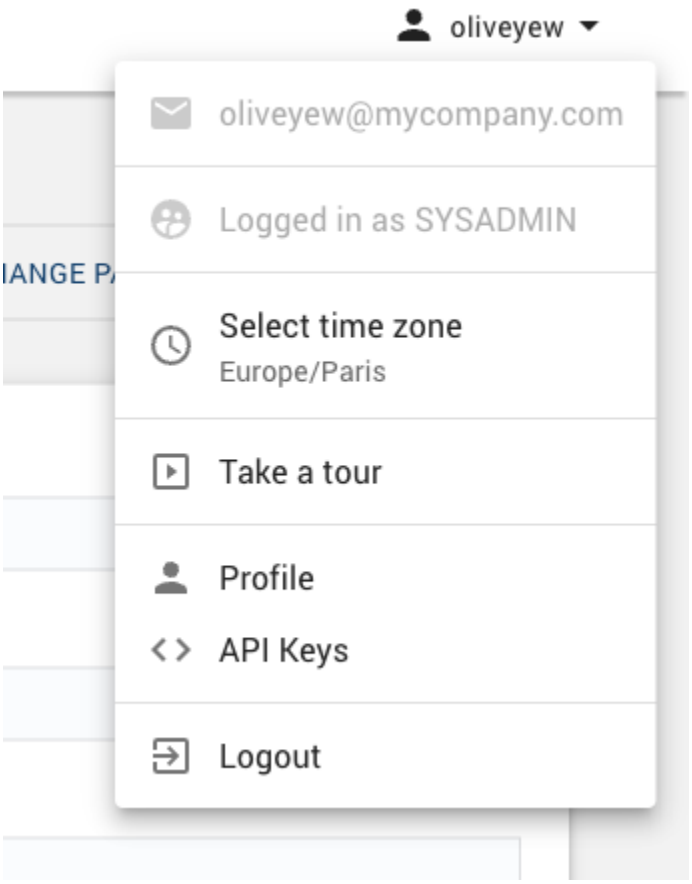
Interactive session duration (minutes)
When users are authenticated with an interactive login, a session token is created (JWT) to grant access for a specific duration. When the token expires, the user must reauthenticate.
Authentication cache max age (secs)
Authenticating users or applications through an interactive login or API key requires access to the database. A cache is used to make this process more efficient. Cached entries expire once they reach the specified maximum age. Some actions such as disabling a user or an API key will only take affect once this maximum age has been reached and all cached copies have been discarded.

Data Management

Incomplete object expiration (hours)
Due to networking or client issues, some objects (e.g., model files) may not be completely uploaded. A background process deletes these incomplete entries after the specified expiration time has elapsed.
History expiration (days)
Jobs and batches are deleted in the background when they expire. Note that if the number of jobs and batches to delete is very large, these deletions will be performed in blocks to avoid overloading the database.
History cleanup batch size
When jobs have reached their expiration period, they will be deleted by blocks. This parameter configure the block size for the deletion process. This needs to be tuned according to the usage pattern. If the block size is too small, the process may not be able to keep up with the deletion. If the block size is too large, it may overload the database.
Metrics expiration (days)
Metrics are deleted in the background when they expire after the selected number of days, it is recommended this setting to be equals to History expiration (days)
Metrics refresh interval (seconds)
Metrics are stored with the Max value pulled within the refresh interval, change this setting to poll faster the metrics and have more precision on the stored metrics.

4.3.7 User Menu

The user menu allows the current user to obtain information about their account and take certain actions. To open the user menu, click on your username at the top right of the page:



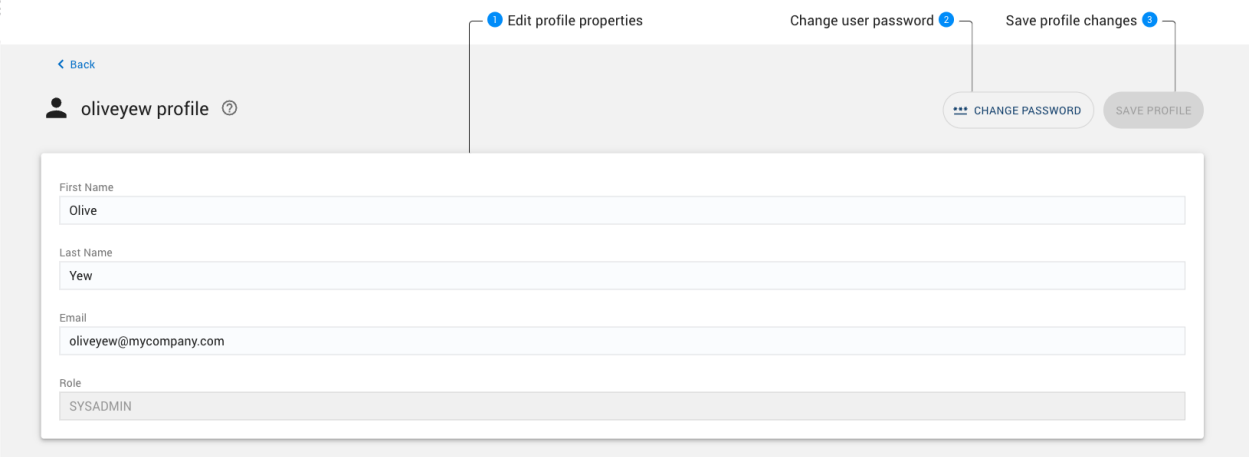
The meanings of the drop-down entries are as follows:

	Displays the email address of the logged-in user.
	Displays the role of the user (if different from STANDARD).
	Launches a dialog that gives a tour and overview of the Cluster Manager.
	Navigates to the *Profile page*, where the user can edit their profile information.
	Navigates to the *User API keys page*, where the user can edit their API keys.
	Logs the user out and opens the login page.

Profile

The profile page

The Profile page allows users to edit information associated with their accounts.



1. Properties can be edited in this panel.
2. The CHANGE PASSWORD button opens a dialog that allows the user to change their password for interactive login to the Cluster Manager.
3. The SAVE PROFILE button saves any changes made to this panel (**1**).

Editing the User Profile

Users with local accounts can update their profile information on the Profile page as follows:

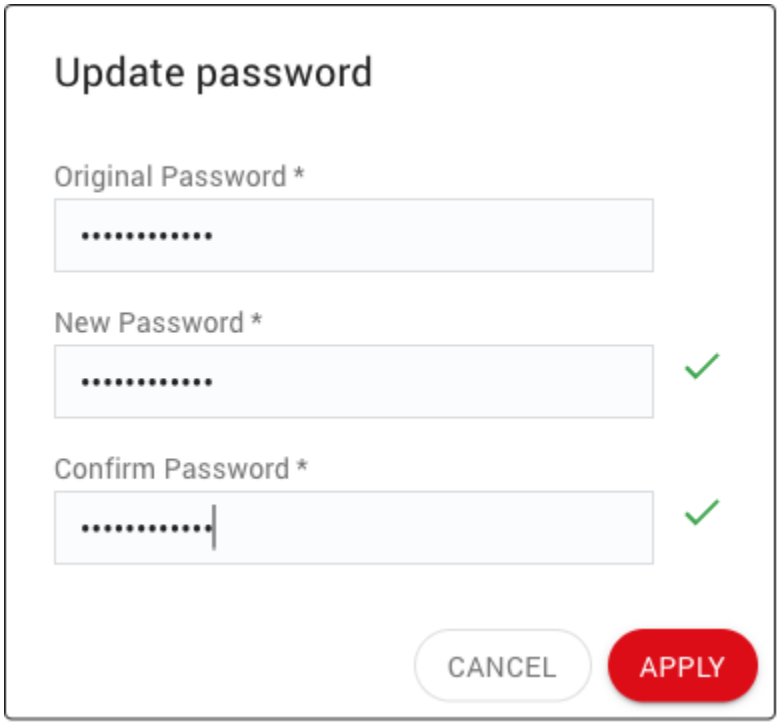
- Edit the new profile information in the form (**1**).
- Save those changes by clicking on the SAVE button (**3**).

Changing the User Password

Users with local accounts can change their passwords with the following steps:

- Click on the CHANGE PASSWORD button (**2**).
- Enter the current and new password in the Update Password dialog:

> **Note**
>
> Passwords must comply with the Password Policy as defined be a SYSADMIN user in the *Password Policy* settings.



API Keys

An API key (Application Programming Interface key) is composed of an access ID and a secret key. API keys are the recommended approach for connecting applications to a Cluster Manager. When the Cluster Manager authenticates access using an API key, all related actions are performed on behalf of the user who owns that key.

When creating an API key, you can specify an optional application name and description to help keep track of how the key is being used. Once a key is created, you can download an associated client license file, which contains the API access ID, the secret key, and the Cluster Manager URL. This license file can be used by client applications and command-line tools to connect to the Cluster Manager. The Cluster Manager keeps track of the timestamp and IP address of the last API key usage.

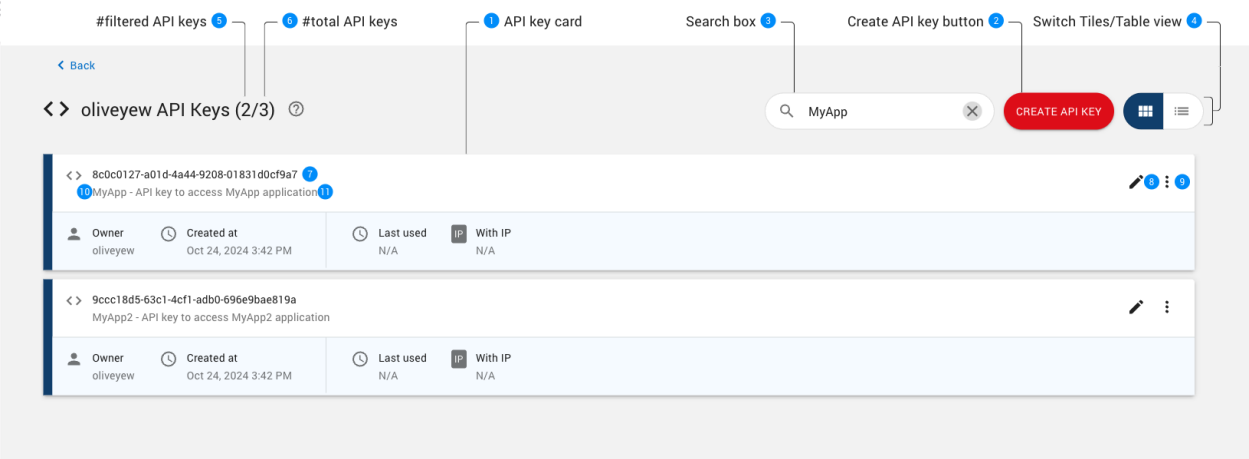
An API key can be enabled or disabled by either the owner or the system administrator.

This set of API key features was designed to simplify the task of monitoring API keys, detecting unwanted usage, and safely rotating keys by disabling previous keys before permanently deleting them.

The API Keys page

The API Keys page displays all of the keys associated with your account:

1. When the page uses the Tiles display mode **(4)**, API keys are displayed as tiles (cards). Each card displays properties associated with the API key and lists several possible actions:
    7. Access code for the API key.
    8. Button to *edit* the application name and description for the API key.
    9. Button to access a menu that exposes additional possible actions (to *delete* or *disable* the API key).
    10. Application name for the API key, if any.
    11. Description for the API key, if any.



2. Button opens a dialog to *create* a new API key.
3. API keys can be filtered using the *Search* box.
4. Button switches the display mode: - Tiles mode displays API keys as tiles (cards) **(1)** - Table mode displays API keys as rows of a table. Details can be found *here*.
5. 6. The number of tiles / rows of the table.

   When the page has been filtered, two numbers are displayed to the right to the page title showing respectively the number of API keys matching the current filtering and the total number of API keys. If no API keys have been filtered, only the total number of API keys is displayed.

Searching

API keys can be filtered using the Search box **(3)**. The user can provide an arbitrary search string, and an API key matches if that string is found in any of the properties associated with that key.

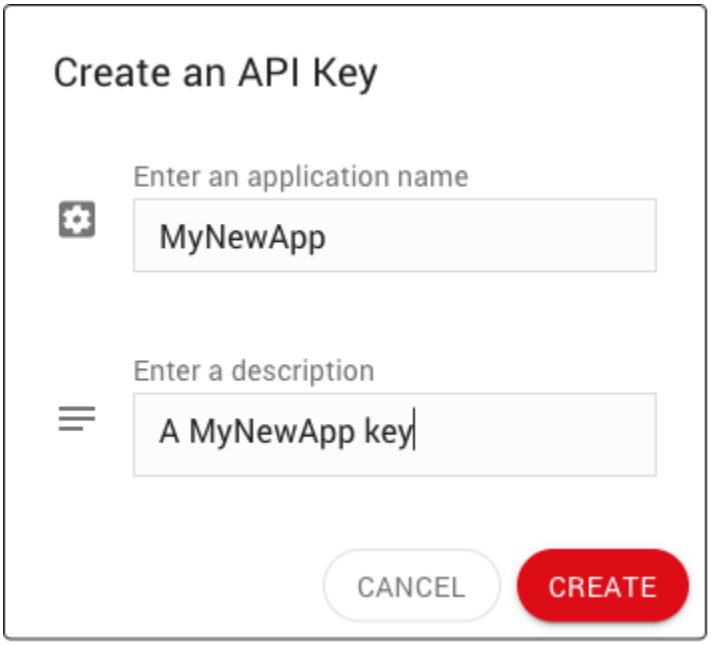
Synchronization with page URL

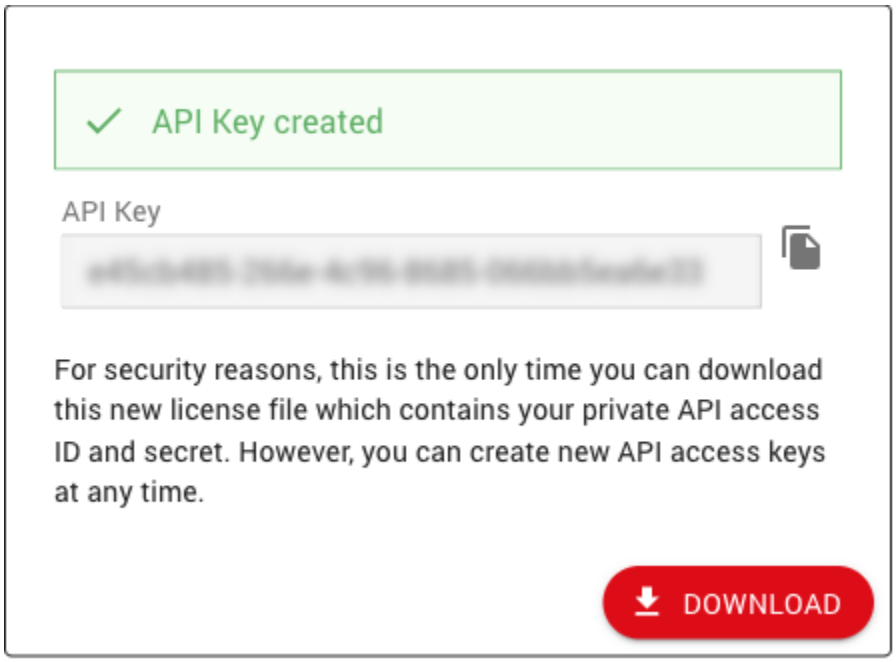
The Cluster Manager creates permalinks for all tables that can be filtered or searched. Changes applied to Search boxes are automatically reflected in the page URL, thus allowing you to copy and paste the URL to easily retrieve the same display later. For example, the URL `http://localhost:61080/editprofile?search=MyApp` would display only API keys associated with the MyApp application (assuming the Cluster Manager is running on `localhost:61080`).

Creating an API key

Users can create new API keys with the following steps:

- Click on the CREATE API KEY button **(2)**
- Edit the API keys properties in the dialog and click on the CREATE button:
- Another dialog will pop up, displaying the access code for the newly created API key:
- Click on the download button to download a license file associated with this new API key.

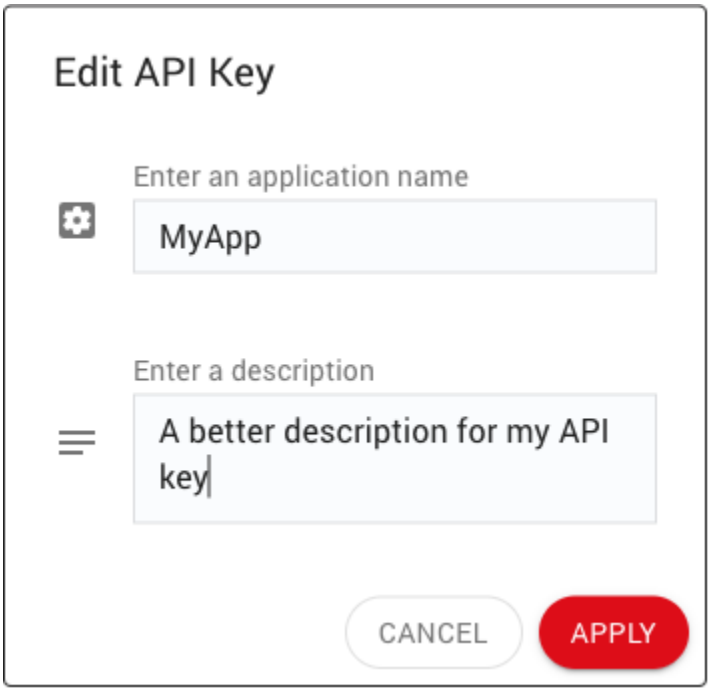

Create an API Key
Enter an application name
MyNewApp
Enter a description
A MyNewApp key
CANCEL
CREATE


API Key created
API Key
For security reasons, this is the only time you can download this new license file which contains your private API access ID and secret. However, you can create new API access keys at any time.
DOWNLOAD

Editing an API Key

Users can change the properties of API keys they own with the following steps:

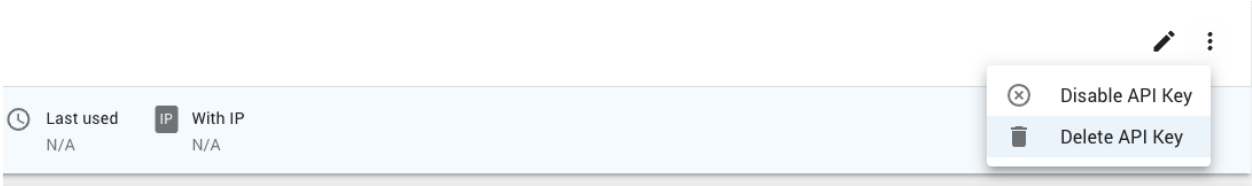
- On the card for the API key, click on the EDIT button **(8)**.
- Edit the API key properties in the dialog and click on the APPLY button.



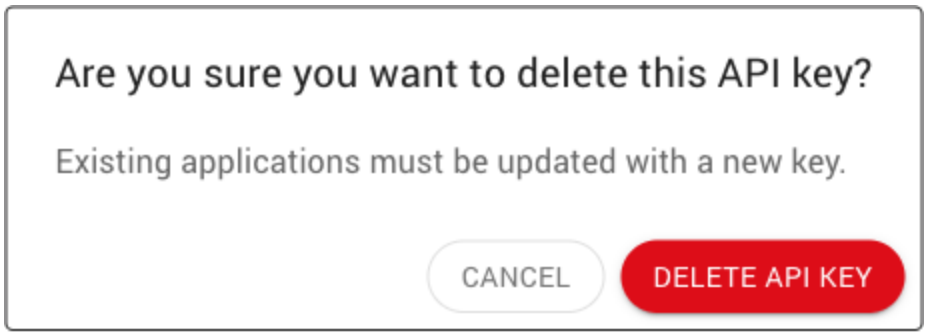
Deleting an API key

Users can delete API keys they own with the following steps:

- On the card for the API key, click on the menu button **(9)** and select the Delete API Key menu item.



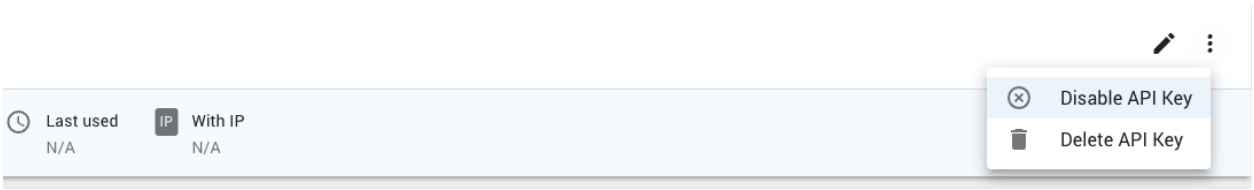
- Confirm that the API key can be deleted.



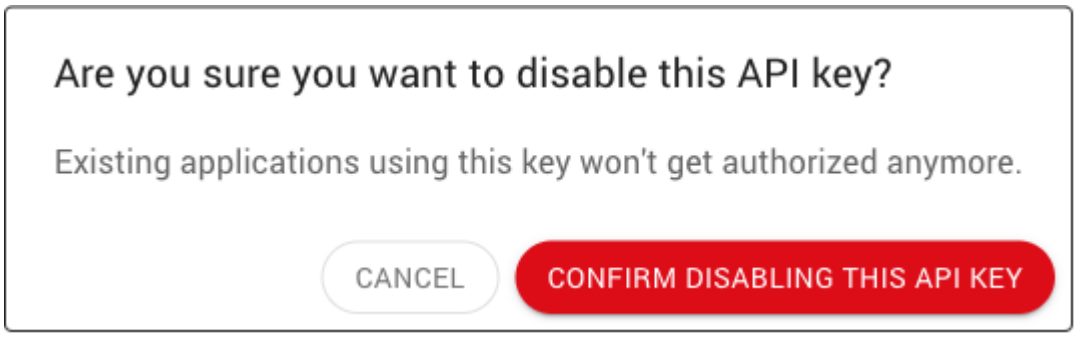
Disabling an API Key

User can disable API keys they own with the following steps:

- Click on the menu button **(9)** and select the Disable API Key menu item.



- Confirm that the API key can be disabled.

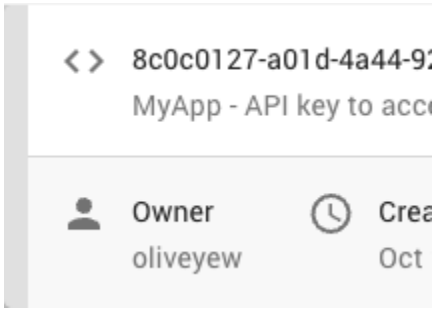


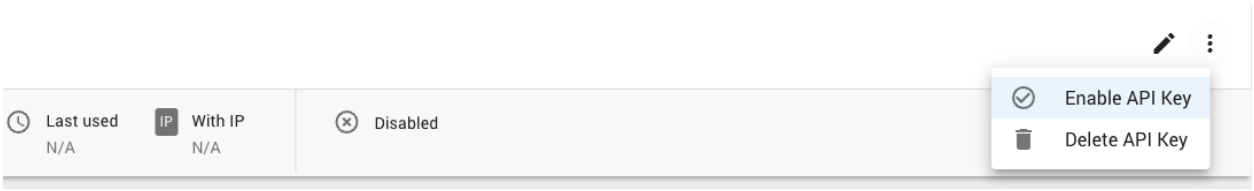
- Once completed, the left border for the API key will be grayed out:

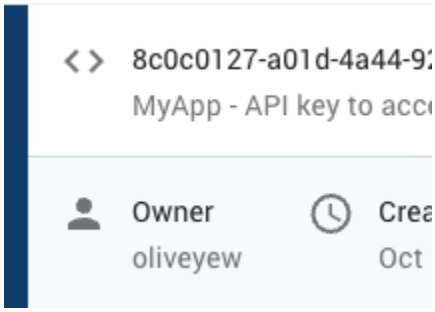
Enabling an API Key

Users can enable API keys they own with the following steps:

- Click on the menu button **(9)** and select the Enable API Key menu item.
- Once completed, the left border for the API key will turn blue:


8c0c0127-a01d-4a44-92
MyApp - API key to acc
Owner
oliveyew
Crea
Oct


Last used
N/A
With IP
N/A
Disabled
Enable API Key
Delete API Key


8c0c0127-a01d-4a44-92
MyApp - API key to acc
Owner
oliveyew
Crea
Oct

CHAPTER

FIVE

PROGRAMMING WITH REMOTE SERVICES

While applications that use Remote Services can generally be built without having to consider where they will be run, there are a few aspects of Remote Services that programmers should be aware of. These are covered in this section.

5.1 Using an API to Create a Compute Server Job

As was noted earlier, a Remote Services client program will always need to be told how to reach the Remote Services cluster. This can be done in two ways. The first is through a license file. This approach is described in an *earlier discussion*. It requires no changes to the application program itself. The same program can perform optimization locally or remotely, depending on the settings in the license file.

Your second option for specifying the desired Compute Servers is through API calls. You would first construct an `empty environment` (using `GRBemptyenv` in C or the appropriate `GRBEnv` constructor in the object-oriented interfaces), then set the appropriate parameters on this environment (typically ComputeServer and ServerPassword), and then start the empty environment (using `GRBstartenv` in C or `env.start()` in the object-oriented interfaces). In Python, you can directly give the appropriate parameters to the environment when creating it.

Once a model is solved and you have retrieved the results from it, we recommend you dispose of it immediately. If you don't need the environment for another model, dispose of the environment as well, to release the remote job. The following example shows how to offload the optimization computation to a Compute Server named `server1` in various APIs, with proper resource management.

C

C++

.NET

Java

Matlab

Python

R

```
    GRBenv   *env   = NULL;
    GRBmodel *model = NULL;
    int error = 0;

    error = GRBemptyenv(&env);
    if (error) goto QUIT;
    error = GRBsetstrparam(env, GRB_STR_PAR_COMPUTESERVER, "server1:61000");
    if (error) goto QUIT;
```

(continues on next page)

(continued from previous page)

```
     error = GRBsetstrparam(env, GRB_STR_PAR_SERVERPASSWORD, "passwd");
     if (error) goto QUIT;
     error = GRBstartenv(env);
     if (error) goto QUIT;

     error = GRBreadmodel(env, "misc07.mps", &model);
     if (error) goto QUIT;
     error = GRBoptimize(model);
     if (error) goto QUIT;

     /* Extract the solution */
     /* ... */

QUIT:
     GRBfreemodel(model);
     GRBfreeenv(env);

     /* return error; */
```

```
GRBEnv env = GRBEnv(true);
env.set(GRB_StringParam_ComputeServer, "server1:61000");
env.set(GRB_StringParam_ServerPassword, "passwd");
env.start();

GRBModel model = GRBModel(env, "misc07.mps");
model.optimize();

// Extract the solution
// ...
```

```
using GRBEnv env = new GRBEnv(true);
env.Set("ComputeServer", "server1:61000");
env.Set("ServerPassword", "passwd");
env.Start();

using GRBModel model = new GRBModel(env, "misc07.mps");
model.Optimize();

// Extract the solution
// ...
```

```
GRBEnv env = null;
GRBModel model = null;
try {
     env = new GRBEnv(true);
     env.set(GRB.StringParam.ComputeServer, "server1:61000");
     env.set(GRB.StringParam.ServerPassword, "passwd");
     env.start();
     model = new GRBModel(env, "misc07.mps");
     model.optimize();
     // Extract the solution
```

(continues on next page)

(continued from previous page)

```
    // ...
} finally {
    if (model != null) {
        model.dispose();
    }
    if (env != null) {
        env.dispose();
    }
}
```

```
params.ComputeServer = 'server1:61000';
params.ServerPassword = 'passwd';

model = gurobi_read('misc07.mps');
result = gurobi(model, params);

% Extract the solution
% ...
```

```
with gp.Env(
    params={
        "ComputeServer": "server1:61000",
        "ServerPassword": "passwd",
    }
) as env, gp.read("misc07.mps", env=env) as model:
    model.optimize()
    # Extract the solution
    ...
```

```
params <- list(
  ComputeServer = 'server1:61000',
  ServerPassword = 'passwd'
)

model <- gurobi_read('misc07.mps')
result <- gurobi(model, params)

# Extract the solution
# ...
```

We refer you to the Environments section of the Gurobi Reference Manual for details on these routines.

5.2 Using an API to Create a Batch

Batch Optimization is a feature only available with the Gurobi Cluster Manager. It allows a client to create a model, tag a set of relevant elements in that model, and then submit that model as a batch. A unique batch ID is returned in response, allowing the client to query and monitor the status of the batch (submitted, completed, etc.). Once the batch request has been processed, the client can retrieve its solution and the relevant attributes of the tagged elements in the model as a JSON document.

In this section, we just want to introduce the principles of the API by briefly illustrating these steps. The code snippets that follow show the main concepts, but aren't necessarily complete programs. You can refer to the Batch Optimization section of the Gurobi Reference Manual for details on the full API and for a complete and functional example.

The first step in this process is to create a batch environment by connecting to a Cluster Manager and enabling batch mode with the CSBatchMode parameter. In order to authenticate the application with the Cluster Manager, you have two options. The first is to use your user name and password, by setting the UserName and ServerPassword parameters. The second, which we recommend, is to use API keys and set the CSAPIAccessID and CSAPISecret parameters instead. We set these parameters directly in the code snippet for simplicity, but we recommended that you set them in the license file or read their values from environment variables to avoid the need to hard-code them.

Then you can build a model, tag the variables and other elements that you will want to export in the solution, and finally you can submit the batch, which gives a batch ID.

The following Python code illustrates these steps:

```
import gurobipy as gp

# create a batch environment
with gp.Env(empty=True) as env:
    env.setParam('CSManager',     'http://localhost:61080')
    env.setParam('CSAPIAccessID', '0e8c35d5-ff20-4e5d-a639-10105e56b264')
    env.setParam('CSAPISecret',   'd588f010-ad47-4310-933e-1902057661c9')
    env.setParam('CSBatchMode',   1)
    env.start()

    # build the model
    with gp.read('misc07.mps', env) as model:

        # set tags to control the solution export
        [...]

        # submit and get the batch ID
        batchid = model.optimizeBatch()
```

Then your client or application can monitor the batch status. Here is an example that accesses and prints the current status:

```
# create a batch environment
with gp.Env(empty=True) as env:
    env.setParam('CSManager',     'http://localhost:61080')
    env.setParam('CSAPIAccessID', '0e8c35d5-ff20-4e5d-a639-10105e56b264')
    env.setParam('CSAPISecret',   'd588f010-ad47-4310-933e-1902057661c9')
    env.setParam('CSBatchMode',    1)
    env.start()

    # get the batch information
```

(continues on next page)

(continued from previous page)

```
    with gp.Batch(batchid, env) as batch:

        print("Batch ID {}: Error code {} ({})".format(
            batch.BatchID, batch.BatchErrorCode, batch.BatchErrorMessage))
```

Once the batch is complete, you can retrieve the solution as a JSON object:

```
# create a batch environment
with gp.Env(empty=True) as env:
    env.setParam('CSManager',     'http://localhost:61080')
    env.setParam('CSAPIAccessID', '0e8c35d5-ff20-4e5d-a639-10105e56b264')
    env.setParam('CSAPISecret',   'd588f010-ad47-4310-933e-1902057661c9')
    env.setParam('CSBatchMode',   1)
    env.start()

    # get the batch information
    with gp.Batch(batchid, env) as batch:

        # Get JSON solution as string, create dict from it
        sol = json.loads(batch.getJSONSolution())

        # Pretty printing the general solution information
        print(json.dumps(sol["SolutionInfo"], indent=4))
```

5.3 Performance Considerations on a Wide-Area Network (WAN)

While using Gurobi Compute Server doesn't typically require you to make any modifications to your code, performance considerations can sometimes force you to do some tuning when your client and server are connected by a slow network (e.g., the internet). We'll briefly talk about the source of the issue, and the changes required to work around it.

In a Gurobi Compute Server, a call to a Gurobi routine can result in a network message between the client and the server. An individual message is not that expensive, but sending hundreds or thousands of messages could be quite time-consuming. Compute Server does a few things to reduce the number of such messages. First, it makes heavy use of caching. If you request an attribute on a single variable, for example, the client library will retrieve and store the value of that attribute for all variables, so subsequent requests won't require additional communication. In addition, our *lazy update* approach to model building allows us to buffer additions and modifications to the model, You can feel free to build your model one constraint at a time, for example. Your changes are communicated to the server in one large message when you request a model update.

Having said that, we should add that not all methods are cached or buffered. As a result, we suggest that you avoid doing the following things:

- Retrieving the non-zero values for individual rows and columns of the constraint matrix (using, for example, `GRBgetconstrs` in C, `GRBModel::getRow` in C++, `GRBModel.getRow` in Java, `GRBModel.GetRow` in .NET, and `Model.getRow` in Python).
- Retrieving individual string-valued attributes.

Of course, network overhead depends on both the number of messages that are sent and the sizes of these messages. We automatically perform data compression to reduce the time spent transferring very large messages. However, as you may expect, you will notice some lag when solving very large models over slow networks.

5.4 Callbacks

As you might imagine, since the actual optimization task runs on a remote system in a Compute Server environment, Gurobi callbacks give different behavior than they do when the task runs locally. In particular, callbacks are both less frequent and more restrictive to avoid communication overhead and a loss in performance.

You will receive `MESSAGE`, `BARRIER`, `SIMPLEX`, `PRESOLVE`, `MIP`, `MIPSOL`, and `MULTIOBJ` callbacks; you will not receive `MIPNODE` callbacks. As a result, you will only have access to a subset of the callback information that you would be able to obtain when running locally. Also, callbacks can't be used in *batch mode*.

You can still request that the optimization shall be terminated from any of the callbacks you receive, though. Please refer to the Callback Codes section for more information on the various callback codes.

5.5 Developing for Compute Server

Using Gurobi Compute Server typically requires no changes to your program. This section covers the few exceptions.

5.5.1 Coding for Robustness

Client-server computing introduces a few robustness situations that you wouldn't face when all of your computation happens on a single machine. Specifically, by passing data between a client and a server, your program is dependent on both machines being available, and on an uninterrupted network connection between the two systems. The queuing and load balancing capabilities of Gurobi Compute Server can handle the vast majority of issues that may come up, but you can take a few additional steps in your program if you want to achieve the maximum possible robustness.

The one scenario you may need to guard against is the situation where you lose the connection to the server while the portion of your program that builds and solves an optimization model is running. Gurobi Compute Server will automatically route queued jobs to another server, but jobs that are running when the server goes down are interrupted (the client will receive a `NETWORK` error). If you want your program to be able to survive such failures, you will need to architect it in such a way that it will rebuild and resolve the optimization model in response to a `NETWORK` error. The exact steps for doing so are application dependent, but they generally involve encapsulating the code between the initial Gurobi environment creation and the last Gurobi call into a function that can be reinvoked in case of an error.

For examples of properly disposing of the Gurobi model and environment, read *Using an API to Create a Compute Server Job*.

5.5.2 Features Not Supported in Compute Server

As noted earlier, there are a few Gurobi features that are not supported in Compute Server. We've mentioned some of them already, but we'll give the full list here for completeness. You will need to avoid using these features if you want your application to work in a Compute Server environment.

The unsupported features are:

- **User cuts:** The `MIPNODE` callback isn't supported, so you won't have the opportunity to add your own cuts. User cuts aren't necessary for correctness, but applications that heavily rely on them may experience performance issues.
- **Multithreading within a single Gurobi environment:** This isn't actually supported in Gurobi programs in general, but the results in a Compute Server environment are sufficiently difficult to track down that we wanted to mention it again here. All models built from an environment share a single connection to the Compute Server. This one connection can't handle multiple simultaneous messages. If you wish to call Gurobi from multiple threads in the same program, you should make sure that each thread works within its own Gurobi environment.

- **Advanced simplex basis routines:** The C routines that work with the simplex basis (`GRBFSolve`, `GRBBSolve`, `GRBBinvColj`, `GRBBinvRowi`, and `GRBgetBasisHead`) are not supported.

5.6 Distributed Algorithm Considerations

The distributed algorithms have been designed to be nearly indistinguishable from the single machine versions. Our hope is that, if you know how to use the single machine version, you'll find it straightforward to use the distributed version. The distributed algorithms respect all of the usual parameters. For distributed MIP, you can adjust strategies, adjust tolerances, set limits, etc. For concurrent MIP, you can allow Gurobi to choose the settings for each machine automatically or you can use `concurrent environments` to make your own choices. For distributed tuning, you can use the usual tuning parameters, including TuneTimeLimit, TuneTrials, and TuneOutput.

5.6.1 Performance Across Distributed Workers

There are a few things to be aware of when using distributed algorithms, though. One relates to relative machine performance. As we noted earlier, distributed algorithms work best if all of the workers give very similar performance. For example, if one machine in your worker pool were much slower than the others in a distributed tuning run, any parameter sets tested on the slower machine would appear to be less effective than if they were run on a faster machine. Similar considerations apply for distributed MIP and distributed concurrent. We strongly recommend that you use machines with very similar performance. Note that if your machines have similarly performing cores but different numbers of cores, we suggest that you use the Threads parameter to make sure that all machines use the same number of cores.

5.6.2 Callbacks

Another difference between the distributed algorithms and our single-machine algorithms is in the callbacks. The distributed MIP and distributed concurrent solvers do not provide the full range of callbacks that are available with our standard solvers. They will only provide the `MIP`, `MIPNODE`, and `POLLING` callbacks. See the Callback Codes section of the Gurobi Reference Manual for details on the different callback types.

5.6.3 Logging

The distributed algorithms provide slightly different logging information from the standard algorithms. Consult the Distributed MIP Logging section of the Gurobi Reference Manual for details.

5.7 Cluster REST API

Gurobi Remote Services also expose a REST API to support advanced integration and monitoring. The API follows standard REST principles and can be used from a variety languages and tools (Java, Python, Node, curl, ...).

If you are using a self-managed cluster (without a Cluster Manager), the REST API is provided by the nodes of the cluster and is fairly basic. It covers monitoring functions only, providing information on the nodes and the jobs processed by the cluster.

To access an API endpoint, you will also need to provide the access password in the header `X-GUROBI-CSPASSWORD`. Some endpoints are restricted and the administrator password will be required. Detailed, interactive documentation is available in Swagger format, and can be accessed directly from a cluster node. For example:

```
http://server1/swagger.html
```

If you are using a Cluster Manager, a more extensive REST API is provided that covers not only the monitoring of nodes and jobs, but also the management of users, batches and repository files. In fact, all of the functions exposed by `grbcluster` are supported. Some endpoints are restricted, and administrator or system administrator authentication will be required. You can generate API keys and pass the access ID and the secret key in the `X-GUROBI-ACCESS-ID` `X-GUROBI-SECRET-KEY` headers respectively. Complete Swagger documentation is available in the Cluster Manager Web User Interface (in the `Help` section).

CHAPTER

SIX

USING REMOTE SERVICES WITH GUROBI INSTANT CLOUD

Our Gurobi Instant Cloud product is built on top of either Amazon's Elastic Compute Cloud (EC2) platform or Microsoft's Azure platform. When you launch an Instant Cloud machine, we launch a machine instance on EC2 or Azure and then start Gurobi Remote Services on that machine. You also have the option of launching multiple machines, in which case we'll create a Remote Services cluster on those machines. Once you have *set up your client* with a client license file, you will be able to use `grbcluster` to *monitor* the cluster. Note, however, that cluster administrative commands are not accessible, since the Gurobi Instant Cloud Manager already plays the cluster administrator role. Note also that communication with your Instant Cloud machine will always use HTTPS, and it will go through a *region router*.

6.1 Client Setup

To access the cluster started by Instant Cloud, you first need to download the machine or pool license file from the Instant Cloud manager. You can download the default license file from the license panel, the pool license from the pool panel, or the machine license from the machine panel. Then, you need to save this file in your home directory or in one of the following locations:

- `C:\gurobi\` on Windows
- `/opt/gurobi/` on Linux
- `/Library/gurobi/` on macOS
- The user's home directory

You can also set the environment variable `GRB_LICENSE_FILE` to point to this file.

6.2 Client Commands

Once you have set up your client license file and started an Instant Cloud instance, you can use `grbcluster` to list the nodes in your cluster or issue other client commands. Instances can be started by submitting a job through the `gurobi_cl` command-line tool, through the Gurobi programming language APIs, or manually through the Instant Cloud Manager website.

If you try to run `grbcluster` without first starting an instance, you will get the following error:

```
fatal : Instant Cloud pool default has no machines
```

If your instance is in the process of starting, you will get the following error:

```
fatal : Instant Cloud pool default is not ready
```

If your instance is up and running, `grbcluster` will list the nodes in your cluster:

```
> grbcluster nodes
ADDRESS          STATUS TYPE    GRP              LICENSE #Q #R JL IDLE  %MEM  %CPU
ip-172-31-31-180 ALIVE  COMPUTE m-HkQmbubhWH1g7m VALID   0  0  2  12m0s 27.08 1.98
ip-172-31-62-109 ALIVE  COMPUTE m-HJSXmb_-2WBkLX VALID   0  0  2  12m0s 27.49 0.00
```

To obtain additional details (about the license file, the cloud pool, or the name of the server), you can use the verbose mode with the `-v` flag:

```
> grbcluster -v nodes
verb  : Reading license file /licenses/gurobi.lic
verb  : Accessing Instant Cloud pool 999999-pool5
verb  : Using remote services on node ip-172-31-31-180
ADDRESS          STATUS TYPE    GRP              LICENSE #Q #R JL IDLE  %MEM  %CPU
ip-172-31-31-180 ALIVE  COMPUTE m-HkQmbubhWH1g7m VALID   0  0  2  12m0s 27.08 1.98
ip-172-31-62-109 ALIVE  COMPUTE m-HJSXmb_-2WBkLX VALID   0  0  2  12m0s 27.49 0.00
```

You can use `grbcluster` to perform most of the same *client commands* on an Instant Cloud cluster that you'd perform on a cluster running locally. You can monitor running and recently processed jobs, access log files, view parameters, etc.

However, you will not be able to execute administrative commands such as aborting a job, or changing the configuration of a node. To do so, please use the Instant Cloud Manager or the Instant Cloud REST API instead.

6.3 Region Router

Starting with version 8.0, all communications between clients and the Gurobi Instant Cloud use the HTTPS protocol. This means that your communications are secured and encrypted using standard internet protocols. In addition, Gurobi servers enforce the latest encryption policies (TLS v1.2 and above only). For better security, the dedicated machines started by Instant Cloud on your behalf cannot be accessed directly. All communications must transit through a secured and highly-available region router acting as a reverse proxy. This also facilitates the integration with clients, as only the standard HTTPS protocol and standard port 443 need to be open if a firewall is in place.

The region router is automatically detected and used based on the pool definition or the machine license file.

CHAPTER

SEVEN

APPENDIX A: GRB_RS

```
Usage:
  grb_rs [flags]           Start the remote services as a standard process
  grb_rs --help            Display usage
  grb_rs command [flags]   Execute a specific command
  grb_rs command --help    Display more information about a command

Flags can be set using --flag=value or the short form -f=value if defined.
A boolean flag can be enabled using --flag or the short form -f if defined.

Configuration Helper Commands:
  aws         Display machine information when running on Amazon Web Services
  azure       Display machine information when running on Microsoft Azure
  ciphers     List the supported TLS cipher suites and policies
  hash        Hash a password
  init        Clone the default data directory and configuration to current
                directory
  token       Generate a cluster token
  properties  Display help about configuration properties

Service Commands:
  install     Install the service
  restart     Start or restart the service (install the service if necessary)
  start       Start the service (install the service if necessary)
  stop        Stop the service
  uninstall   Uninstall the service

With no command, grb_rs will start the remote services as a standard process
and the following flags are available for quick configuration. The full list
of properties can be displayed with the 'grb_rs properties' command and the
properties can be set in the grb_rs.cnf configuration file.

Logging Flags:
      --console-ts             Add timestamps to console log messages
      --logfile string         Log to a rotating log file
      --logfile-max-age int    Limit the rotating log file to a number of days
      --logfile-max-size int   Limit the size of each file to a size in Mb
      --no-console             Disable log to console
      --syslog                 Log to syslog or Windows event log
  -v, --verbose                Enable verbose logging
```

(continues on next page)

(continued from previous page)

```
Configuration Flags:
  -c, --config string          Location of the configuration file
                                 (default: 'grb_rs.cnf')
      --data string            Location of the data root directory
                                 (default: 'data')
      --group                  Tags the node with a group name
      --hostname               Overrides the public name on this name
      --idle-shutdown int      Shutdown if the server is idle for more than the
                                 specified time limit (minutes)
      --join URL               Join a cluster using the specified cluster
                                 representative node address
      --manager URL            Register the node with a Cluster Manager
      --port int               Start the node on the given port
      --service                Indicates if it is started by a service manager
      --worker                 Declare this node as a distributed worker

Security Flags:
      --tls                    Use TLS encryption between nodes
      --tlscert string         Path to TLS certificate file
      --tlskey string          Path to TLS key file
      --tls-insecure           Use TLS encryption between nodes but disable
                                 verification of certificates
      --tls-ciphers            A comma-separated list of supported cipher policies or
                                 suites used for TLS secure communication
      --manager-insecure       Disable certificate verification if TLS is used
                                 to communicate with the Cluster Manager

General Flags:
      --version                Display version information
      --help                   Display usage
```

CHAPTER

EIGHT

APPENDIX B: GRB_RS - CONFIGURATION PROPERTIES

The following list of properties can also be displayed using the `grb_rs properties` command.

ADMINPASSWORD: Type string
Client password for administrator access. The password can be sent in the clear or can be hashed using the 'grb_rs hash' command for better security.

AWS: Type bool
Use `--aws` to override on the command line.

Enables AWS configuration using ec2 user-data.

AWS_HOSTNAME_MODE: Type string
Indicates how to get the node name; deprecated. See CLOUD_HOSTNAME_MODE.

AZURE: Type bool
Use `--azure` to override on the command line.

Enables Azure configuration using user-data.

CLIENT_DETAILS_ADMIN: Type bool
Indicates that client details such as hostname and IP address are only accessible as an admin user. When a job is submitted, the client hostname, IP, and process ID are recorded. By default, this information can be displayed by any user through the grbcluster command-line tool or the REST API. If this property is set to true, only the administrator can access this information.

CLOUDKEY: Type string
Cloud license key.

CLOUD_HOSTNAME_MODE: Type string
Indicates how to get the node name on the AWS or Azure: 'public' or 'private'. The public mode will assign the public DNS name or IP, whereas the private mode will assign the base name of the private DNS name. The private mode is used with a Gurobi router.

CLUSTER_ADMINPASSWORD: Type string
Client password to administer the cluster. The password can be stored in the clear or can be hashed using 'grb_rs hash' command for better security.

CLUSTER_TOKEN: Type string
Unique cluster identifier. The token is an encrypted key used to secure communication between nodes. All nodes of a cluster must have the same token. Use 'grb_rs token' command to generate a new token.

CONSOLE_TS: Type bool
Use `--console-ts` to override on the command line.

Add timestamps to console log messages.

DATA_DIR: Type string (default data)
Use `--data` to override on the command line.

Root directory to store remote services data.

DEGRADED_TIMEOUT: Type int (default 60)
Timeout for evicting a DEGRADED node from the cluster; 0 for no timeout.

FILE_DESCRIPTOR_LIMIT: Type int (default 2048)
Maximum number of file descriptors.

FIXED_JOBLIMIT: Type bool
Indicates whether the job limit can be changed once a node has started.

FIXED_NODE_THREADLIMIT: Type bool
Indicates whether the node thread limit can be changed once a node has started.

GROUP: Type string
Node grouping for job affinity assignment.

HARDJOBLIMIT: Type int (default 0)
A hard limit on the number of simultaneous client jobs. Certain jobs (those with priority 100) are allowed to ignore the JOBLIMIT, but they aren't allowed to ignore this limit. Client requests beyond this limit are queued. Use 0 to disable.

HELMET: Type bool (default true)
Enable Helmet secure HTTP headers request protection. If there is connectivity issue please disable Helmet request protection

HOSTNAME: Type string
Use `--hostname` to override on the command line.

Advertised hostname of the cluster node.

IDLESHUTDOWN: Type int (default -1)
Use `--idle-shutdown` to override on the command line.

Idle time limit (minutes) to trigger a shutdown of the server, -1 to disable.

IDLESHUTDOWN_COMMAND: Type string
Command to execute when the idle shutdown is reached (typically a command to shut down the machine).

IDLESHUTDOWN_STOPPED: Type int (default -1)
Idle time limit (minutes) to trigger a shutdown of the machine once the processing state is STOPPED; -1 to disable.

IDLETIMEOUT: Type int (default 0)
Default idle timeout, in seconds. If a job does not send a command for more than the timeout, it will be terminated. Use 0 to disable.

IGNOREPRIORITIES: Type bool
Disables job priority handling.

JOBLIMIT: Type int (default 2)
A limit on the number of client jobs that are allowed to run on the server at a time. Client requests beyond this limit are queued.

JOIN: Type string
Use `--join` to override on the command line.

List of other nodes to join.

JOIN_TIMEOUT: Type int (default 20)
Timeout for a successful join; use 0 to disable.

KEEPALIVE_TIMEOUT: Type int (default 60)
Default keep-alive timeout, in seconds. If a job does not send a keep-alive message within the timeout interval, it will be terminated.

KEEP_BATCH_DATA: Type bool
Indicates if temporary batch files must be kept. When a batch job is executed, input data is first generated in an input directory. Output data is similarly stored in an output directory. By default, these directories are deleted once the batch is completed to save space. However, using this property, the files can be kept until the job is evicted from the recent history (see MAX_RECENT.)

LICENSEID: Type string
Cloud license ID.

LOGFILE: Type string
Use `--logfile` to override on the command line.

Enables logging to a rotating log file.

LOGFILE_MAX_AGE: Type int (default 5)
Use `--logfile-max-age` to override on the command line.

Limits the rotating log file to a number of days.

LOGFILE_MAX_SIZE: Type int (default 500)
Use `--logfile-max-size` to override on the command line.

Limits the size of each file to a size in MB.

MANAGER: Type string
Use `--manager` to override on the command line.

Cluster Manager URL.

MANAGER_INSECURE: Type bool
Indicates whether the connection to the manager should use TLS insecure.

MANAGER_TIMEOUT: Type int (default 300)
Timeout (in seconds) for all REST API calls to the Cluster Manager.

MAX_QUEUE: Type int (default 1000)
Maximum number of jobs in the queue.

MAX_RECENT: Type int (default 50)
Maximum number of executed jobs in the recent history.

MEMLIMIT: Type float64 (default -1)
Limits the total amount of memory of a job (in GB, i.e., 10^9 bytes). If more is needed, Gurobi will fail with an OUT_OF_MEMORY error. Note that it is not possible to retrieve solution information after a termination error.

NODE_THREADLIMIT: Type int (default 0)
A limit on the number of threads that are allowed to run on the server at a time. Client requests beyond this limit are queued.

NOQUEUE: Type bool
Disables job queueing.

NO_CONSOLE: Type bool
Use `--no-console` to override on the command line.

Disables the console log.

NO_LOCAL_DISK: Type bool (default true)
Indicates that local disk may not be used by workers (to store node files, solution files, etc.). You can set this to false to permit local files if you are confident that local disk space won't be exhausted.

PASSWORD: Type string
Client password to access the cluster. The password can be in clear or can be hashed using 'grb_rs hash' command for better security.

PORT: Type int
Use `--port` to override on the command line.

Port number for the REST API.

REGISTRATION_PORT: Type int
Port used to register worker, 0 means a dynamic port.

SOFTMEMLIMIT: Type float64 (default -1)
Limits the total amount of memory of a job (in GB, i.e., 10^9 bytes) available to Gurobi. If more is needed, Gurobi will terminate with a MEM_LIMIT status code, leading to a graceful exit of the optimization, such that it is possible to retrieve solution information afterwards or (in the case of a MIP solve) resume the optimization.

STRICT_RUNTIME_MATCHING: Type bool (default true)
Indicates whether matching of client and runtime version is strict. When the matching is strict, the runtime having the same technical release will be selected. When it is not strict, the runtime having the latest technical release will be selected.

SYSLOG: Type bool
Use `--syslog` to override on the command line.

Log to syslog or Windows event log.

THREADLIMIT: Type int (default -1)
Maximum number of threads used by a worker.

TIMELIMIT: Type float64 (default -1)
Limits the total time expended during optimization (in seconds). If the limit is reached, the Optimization will return the TIME_LIMIT status.

TLS: Type bool
Use `--tls` to override on the command line.

Enables TLS encryption protocol.

TLS_CERT: Type string
Use `--tlscert` to override on the command line.

Path to TLS certificate file. If not specified, a self-signed certificate will be generated.

TLS_CIPHERS: Type string
Use `--tls-ciphers` to override on the command line.

A comma-separated list of supported cipher policies or suites used for TLS secure communication. Use grb_rs ciphers to list the supported policies and ciphers for this release If not specified, the Default Policy of secure cipher suites is used.

TLS_INSECURE: Type bool
Use `--tls-insecure` to override on the command line.

Enables TLS encryption protocol but skips certificate verification. This mode can be used with self-signed certificate so that data is encrypted.

TLS_KEY: Type string
Use `--tlskey` to override on the command line.

Path to TLS key file. If not specified, a key will be generated to self-sign a certificate.

USERNAME_ADMIN: Type bool
Indicates that job username is only accessible as an admin user. When a job is submitted, the username for

the client process is recorded. By default, this information can be displayed by any user through the grbcluster command-line tool or the REST API. If this property is set to true, only the administrator can access this information.

VERBOSE: Type bool

Use `--verbose` to override on the command line.

Enables verbose logging.

WLS_JOB_CONFIG: Type string

Defines the default WLS On-Demand config for jobs that are submitted without a configuration. Setting this configuration is particularly useful for jobs submitted using versions where WLS On-Demand configuration was not yet supported.

WORKER: Type bool

Use `--worker` to override on the command line.

Declare the node as a distributed worker.

CHAPTER

NINE

APPENDIX C: GRB_RSM

```
Usage:
  grb_rsm [flags]           Start the Cluster Manager as a standard process
  grb_rsm --help            Display usage
  grb_rsm command [flags]   Execute a specific command
  grb_rsm command --help    Display more information about a command

Flags can be set using --flag=value or the short form -f=value if defined.
A boolean flag can be enabled using --flag or the short form -f if defined.

Configuration Commands:
  ciphers     List the supported TLS cipher suites and policies
  database    Display database information and perform upgrades
  properties  Display help about configuration properties
  user        Special user actions

Service Commands:
  install     Install the service
  restart     Start or restart the service (install the service if necessary)
  start       Start the service (install the service if necessary)
  stop        Stop the service
  uninstall   Uninstall the service

Logging Flags:
      --console-ts             Add timestamps to console log messages
      --logfile string         Log to a rotating log file
      --logfile-max-age int    Limit the rotating log file to a number of days
      --logfile-max-size int   Limit the size of each file to a size in Mb
      --no-console             Disable log to console
      --syslog                 Log to syslog or Windows event log
  -v, --verbose                Enable verbose logging

Configuration Flags:
  -c, --config string          Location of the configuration file (default: 'grb_rsm.cnf
↪')
      --database string        MongoDB database URL
      --port int               Start the server on the given port
      --service                Indicates that it is started by a service manager

Security Flags:
      --tls                    Use TLS for communication encryption
```

(continues on next page)

(continued from previous page)

```
      --tlscert string         Path to TLS certificate file
      --tlskey string          Path to TLS key file
      --tls-insecure           Use TLS but disable verification of certificates,
                                 works with self-signed certificates
      --tls-ciphers            A comma-separated list of supported cipher policies or
                                 suites used for TLS secure communication

General Flags:
      --version                Display version information
      --help                   Display usage
```

CHAPTER

TEN

APPENDIX D: GRB_RSM - CONFIGURATION PROPERTIES

The following list of properties can also be displayed using the `grb_rsm properties` command.

AUTH_CACHE_AGE: Type int (default 30)
Maximum age of authentication information (seconds); DEPRECATED. This property has been migrated to the Cluster Manager Settings stored into the Database.

CLUSTER_TOKEN: Type string
Unique cluster identifier. The token is an encrypted key used to secure communication between the manager and the cluster nodes. All nodes of a cluster and the manager must have the same token. Use 'grb_rs token' command to generate a new token.

CONSOLE_TS: Type bool
Use `--console-ts` to override on the command line.

Add timestamps to console log messages.

DB_URI: Type string (default mongodb://127.0.0.1:27017)
MongoDB connection string.

HELMET: Type bool (default true)
Enable Helmet secure HTTP headers request protection. If there is connectivity issue please disable Helmet request protection

HISTORY_MAX_AGE: Type int (default 30)
Limit the job history to a number of days; DEPRECATED. This property has been migrated to the Cluster Manager Settings stored into the Database.

HTTP_HEALTH_SERVER: Type bool
Enable an additional HTTP server dedicated to processing health-check (/ping) requests. When using TLS, it may be useful to keep a simple HTTP for health-check.

HTTP_HEALTH_SERVER_PORT: Type int (default 9091)
Indicates the port for the additional HTTP health-check server.

IDLE_CONN_TIMEOUT: Type int (default 130)
maximum amount of time an idle (keep-alive) connection will remain idle before closing itself. Zero means no limit.

JWT_EXPIRATION: Type int (default 480)
Expiration of session tokens (in minutes); DEPRECATED. This property has been migrated to the Cluster Manager Settings stored into the Database.

LOGFILE: Type string
Use `--logfile` to override on the command line.

Enable logging to a rotating log file.

LOGFILE_MAX_AGE: Type int (default 5)
Use `--logfile-max-age` to override on the command line.

Limit the rotating log file to a number of days.

LOGFILE_MAX_SIZE: Type int (default 500)
Use `--logfile-max-size` to override on the command line.

Limit the size of each file to a size in MB.

MAX_IDLE_CONNS: Type int (default 200)
Maximum number of connections in the idle connection pool.

MAX_IDLE_CONNS_PER_HOST: Type int (default 32)
Maximum idle (keep-alive) connections to keep per-host.

NO_CONSOLE: Type bool
Use `--no-console` to override on the command line.

Disable the console log.

OBJECT_NOT_CLOSED_MAX_AGE: Type int (default 1)
Limit the time an object must be closed before being deleted (hours); DEPRECATED. This property has been migrated to the Cluster Manager Settings stored into the Database.

PORT: Type int
Use `--port` to override on the command line.

Port number for the REST API.

SETTINGS_CACHE_AGE: Type int (default 5)
Max age of settings information (seconds).

SYSLOG: Type bool
Use `--syslog` to override on the command line.

Log to syslog or Windows event log.

TLS: Type bool
Use `--tls` to override on the command line.

Enable TLS encryption protocol.

TLS_CERT: Type string
Use `--tlscert` to override on the command line.

Path to TLS certificate file. If not specified, a self-signed certificate will be generated.

TLS_CIPHERS: Type string
Use `--tls-ciphers` to override on the command line.

A comma-separated list of supported cipher policies or suites used for TLS secure communication. Use grb_rsm ciphers to list the supported policies and ciphers for this release If not specified, the Default Policy of secure cipher suites is used.

TLS_INSECURE: Type bool
Use `--tls-insecure` to override on the command line.

Enable TLS encryption protocol but skip certificate verification. This mode can be used with self-signed certificate so that data is encrypted.

TLS_KEY: Type string
Use `--tlskey` to override on the command line.

Path to TLS key file. If not specified, a key will be generated to self-sign a certificate.

VERBOSE: Type bool

Use `--verbose` to override on the command line.

Enable verbose logging.

CHAPTER

ELEVEN

APPENDIX E: GRBCLUSTER

```
Usage:
  grbcluster --help                Display usage
  grbcluster command [flags]       Execute a top-level command
  grbcluster command --help        Display help about a top-level command
  grbcluster group command [flags] Execute a command from a group
  grbcluster group command --help  Display help about a command
                                   from a group

Flags can be set using --flag=value or the short form -f=value if defined.
A boolean flag can be enabled using --flag or the short form -f if defined.

As a first step, the login command must be executed to set the connection
parameters and save them into your client license file. You can list all the
options by getting the help for this command:
  grbcluster login --help

Some commands or group of commands may only be used with a Cluster Manager,
and will be denoted with an asterisk (*).

Command Groups:
  apikey*      Manage API keys
  batch*       Submit, list and manage batches
  job          Monitor and manage the optimization jobs
  node         Monitor and manage cluster nodes
  profile*     Display or manage your profile
  repo*        Manage the artifact repository
  user*        Manage users
For each group, type 'grbcluster group --help' for more options.

Account Commands:
  login        Setup connection parameters
  logout       Clear out connection session
  passwd*      Change password of the current user

Shortcut Commands:
  batches*     List the active batches (same as 'batch list')
  jobs         List the active jobs (same as 'job list')
  nodes        List the cluster nodes (same as 'node list')
```

(continues on next page)

(continued from previous page)

```
Global Flags:
      --console-ts   Add timestamps to console log messages
      --help         Display usage
  -v, --verbose      Enable verbose logging
      --version      Display version information

If a valid Gurobi license file is accessible at the predefined locations or
using the variable GRB_LICENSE_FILE, the license file will provide default
values for connection parameters (server, password, router etc). If the
license file references a Gurobi Instant Cloud pool, it will resolve the
connection parameters of the pool. When using the login command, the
connection parameters will be saved to this client license file.

grbcluster is compatible with standard proxy settings using environment
variables HTTP_PROXY and HTTPS_PROXY. HTTPS_PROXY takes precedence over
HTTP_PROXY for https requests. The values may be either a complete URL or
a "host[:port]", in which case the "http" scheme is assumed.
```

CHAPTER

TWELVE

APPENDIX F: GUROBI_CL

```
Usage:
  gurobi_cl --help                              Display usage
  gurobi_cl [flags]                             Execute a command
  gurobi_cl [param=value]* [option]* filename  Optimize a model file

Flags can be used to execute specific commands:
  -h, --help             Display usage
      --license          Display license information
  -t, --tokens           List license tokens currently in use
  -v, --version          Display version information

When optimizing a model, options can be used:
  -q, --printquality     Display solution quality measures after solving

Command-line only parameters:
  ConcurrentSettings Create concurrent environments from a list of .prm files
  MultiObjSettings   Create multi-objective settings from a list of .prm files
  InputFile          Import data into a model before beginning optimization

Gurobi parameters are documented in the Gurobi Reference Manual:
https://docs.gurobi.com/projects/optimizer/en/12.0/parameters

If a valid Gurobi license file is accessible at the predefined locations or
using the variable GRB_LICENSE_FILE, default values for the license parameters
will be inferred from that file. Most of these values can be overwritten using
the corresponding license parameters.

Examples:
  gurobi_cl misc07.mps
  gurobi_cl Record=1 Method=2 ResultFile=p0033.sol InputFile=p0033.mst \
            InputFile=p0033.hnt.gz LogFile=p0033.log p0033.mps
  gurobi_cl ComputeServer=server.company.com ServerPassword=pass misc07.mps

For further details on how to use this program, visit
https://docs.gurobi.com/projects/optimizer/en/12.0/gurobi_cl
```

CHAPTER

THIRTEEN

APPENDIX G: ACKNOWLEDGMENT OF 3RD PARTY ICONS

The icons used in this document come from the Open Security Architecture.

CHAPTER

FOURTEEN

RELEASE NOTES

You will find below the release notes for Gurobi Remote Services 14.0 and each technical releases. Release notes for Gurobi Optimizer can be found here.

14.1 New or Improved Features

14.1.1 Version 14.0.x

Removed Features

Support for **Amazon DocumentDB** versions below 5.0 has been discontinued. Amazon DocumentDB 4.0 reports MongoDB wire protocol version 7, while current MongoDB drivers require wire protocol version 8, which corresponds to MongoDB 4.2 or later. If your Cluster Manager uses a version of Amazon DocumentDB below 5.0, you must upgrade your cluster to Amazon DocumentDB 5.0 or later before upgrading the Cluster Manager. Amazon Web Services supports in-place major version upgrades from the AWS Management Console or the AWS CLI.

Support for **Azure Cosmos DB for MongoDB** versions below 7.0 has been discontinued. If your Cluster Manager uses an older Azure Cosmos DB version, you must upgrade your cluster to version 7.0 or later before upgrading the Cluster Manager. See Microsoft's Azure Cosmos DB for MongoDB upgrade guide for instructions.

Global Updates

Gurobi Compute Server now supports Gurobi Optimizer 14.0.x, in addition to past releases 12.0.x and 13.0.x. Support for Gurobi Optimizer 11.0.x has been discontinued; it is no longer supported nor packaged in the compute server.

Cluster Manager

The Cluster Manager Web UI has been rewritten in TypeScript. This brings stronger type safety across the frontend codebase, catching a broad class of errors at build time rather than at runtime, and sets up the UI for faster, safer feature development going forward. The rewrite preserves existing functionality and does not require any action from users.

14.2 Supported Components

In the tables below, additions and removals since the previous version are highlighted.

14.0.0

Remote Services 14.0.0 supports the following platforms:

Platform	Operating System	Notes
Windows® 64-bit (win64)	Windows 10, 11 Windows Server 2016, 2019, 2022, 2025	
Linux® x86-64 64-bit (linux64)	Red Hat® Enterprise Linux 8, 9, 10 SUSE® Enterprise Linux 15, 16 Ubuntu® 22.04, 24.04, 26.04 Amazon® Linux 2, 2023	
macOS® 64-bit (macos_universal2)	14 (Sonoma)[1], 15 (Sequoia), 26 (Tahoe)	
Linux arm64 64-bit (linux64)	Red Hat Enterprise Linux 8, 9, 10 SUSE Enterprise Linux 15, 16 Ubuntu 22.04, 24.04, 26.04 Amazon Linux 2, 2023	

Browsers:

Browser	Minimum Version	Notes
Chrome	87	
Firefox	87	
Edge	44	
Safari	14	

Databases:

Database	Version	Notes
MongoDB®	7.0 and up	
Amazon DocumentDB	4.0 and 5.0 and up	Support for versions below 5.0 has been discontinued, see *Release Notes*.
Azure Cosmos DB®	4.2 7.0 and up	Support for versions below 7.0 has been discontinued, see *Release Notes*.

[1] Support dropped since macOS 14 has reached end of support.

14.3 Deprecation Notices

There are currently no active deprecation notices. Support for Amazon DocumentDB versions below 5.0 and Azure Cosmos DB for MongoDB versions below 7.0, previously deprecated, has been discontinued as of version 14.0.0. See *Release Notes*.

14.4 Fixed Issues

14.4.1 Version 14.0.0

Fixed a network deadlock that could occur on the Compute Server's internal command channel between the server and its workers while a solve is running. If a response to a worker command was lost in transit, the server had no safe way to retry the request without risking the worker re-executing the command or the connection stalling. The channel now uses a call-index based idempotent retry protocol, active only for the duration of a solve: each request carries a monotonically increasing call index, and repeating a request with the same index replays the previously stored response instead of re-issuing the command to the worker. This resolves the deadlock and is most noticeable in clusters running large numbers of small, short-running jobs.