GUROBI OPTIMIZER REMOTE SERVICES MANUAL

Contents

Introduction

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.

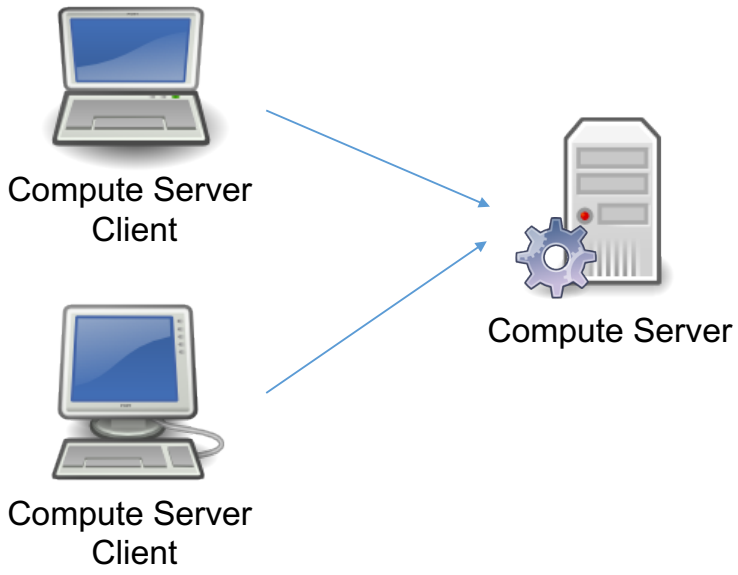
This document is organized into a number of sections. The first section provides an overview of Gurobi Compute Server and Remote Services. The next section, meant for system administrators, provides details on setting up Remote Services. The next sections provide details on using Remote Services and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

Overview

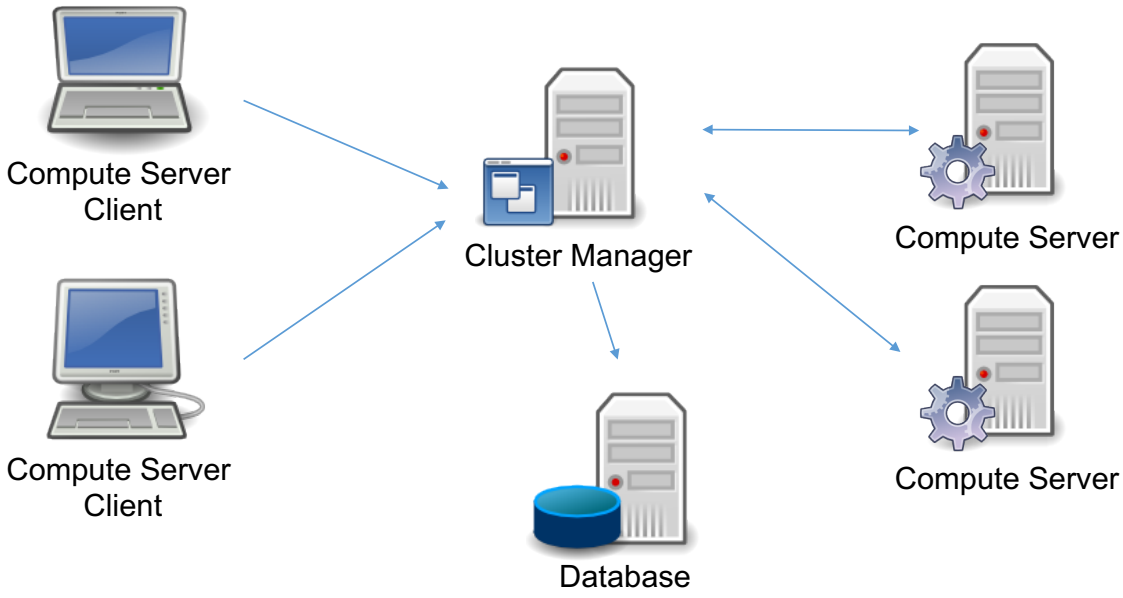
This section gives a quick introduction to the capabilities of Gurobi Remote Services. We first discuss common use cases related to client-server optimization. Then we describe the different components of the architecture and how they can be deployed together. Following this, we review the various security 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.

2.1 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:



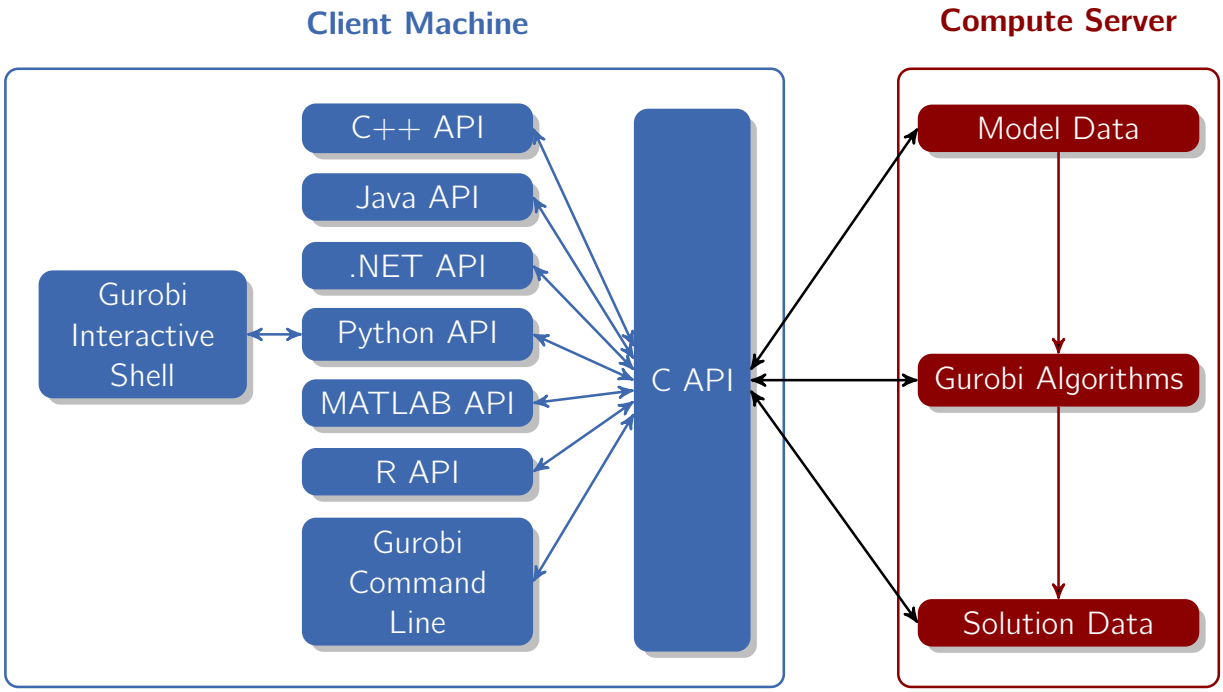
The different configuration options are discussed in a later section.

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 the same problem.

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:

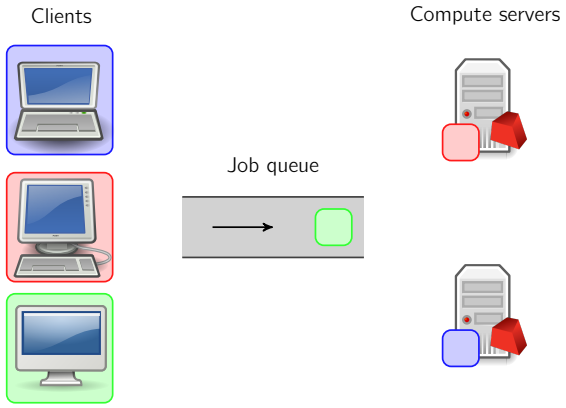


All of our APIs sit on top of our C API. The C API is in charge of building the internal model data structures, invoking the Gurobi algorithms, retrieving solution information, etc. When running Gurobi on a single machine, the C API would build the necessary data structures in local memory. In a Compute 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.

Queuing and Load Balancing

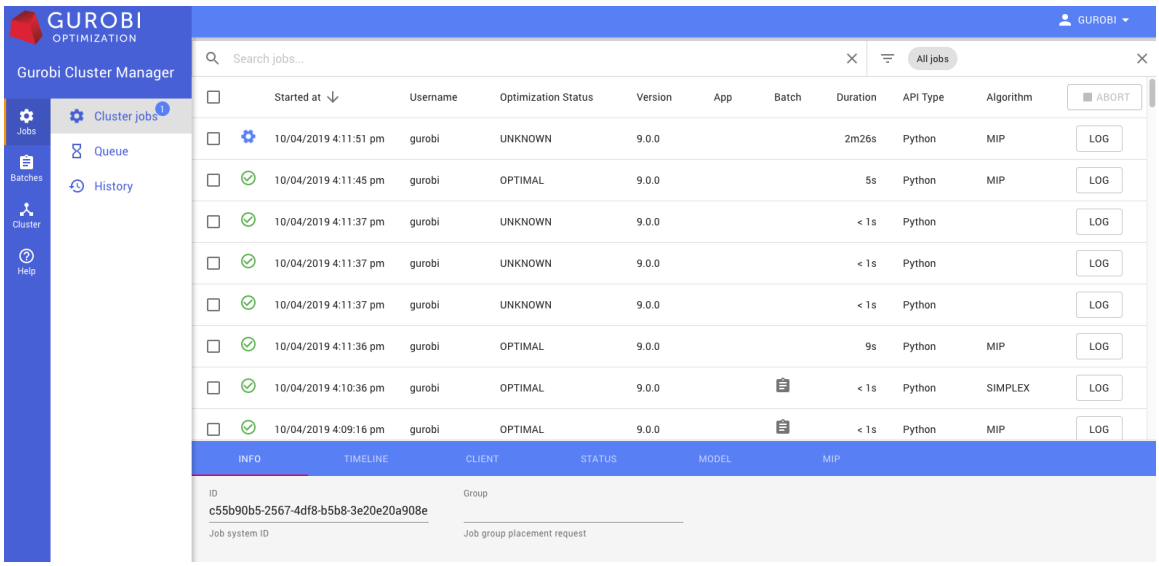
Gurobi Remote Services support queuing and load balancing. You can set a limit on the number of simultaneous jobs each Compute Server will run. When this limit has been reached, subsequent jobs will be queued. If you have multiple Compute Server nodes configured in a cluster, the current job load is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

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.

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.

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:

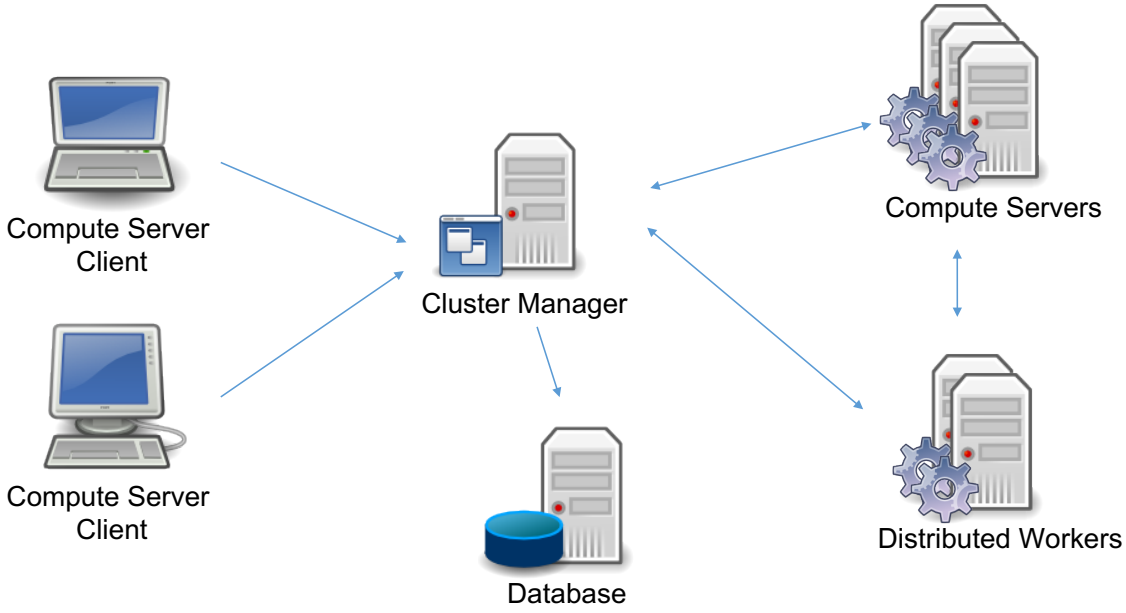
- **A distributed MIP solver**, which allows you to divide the work of solving a single MIP model among multiple machines. A manager machine passes problem data to a set of worker machines to coordinate the overall solution process.
- **A distributed concurrent solver**, which allows you to use multiple 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 whole 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.

2.2 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.

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.

Database

The database supports the Cluster Manager. It stores a variety of important 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 administrator). 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 uses MongoDB as its database - version 4.0 or later. Cluster Manager users must install and configure their own database as part of the Compute Server installation process. The database can be run on-premise or remotely managed by a SaaS provider. It can be deployed as a single node or as a cluster for high availability.

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.

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.

Architecture Topologies

Let us now review a few common deployment configurations.

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.

Cluster Manager with multiple nodes

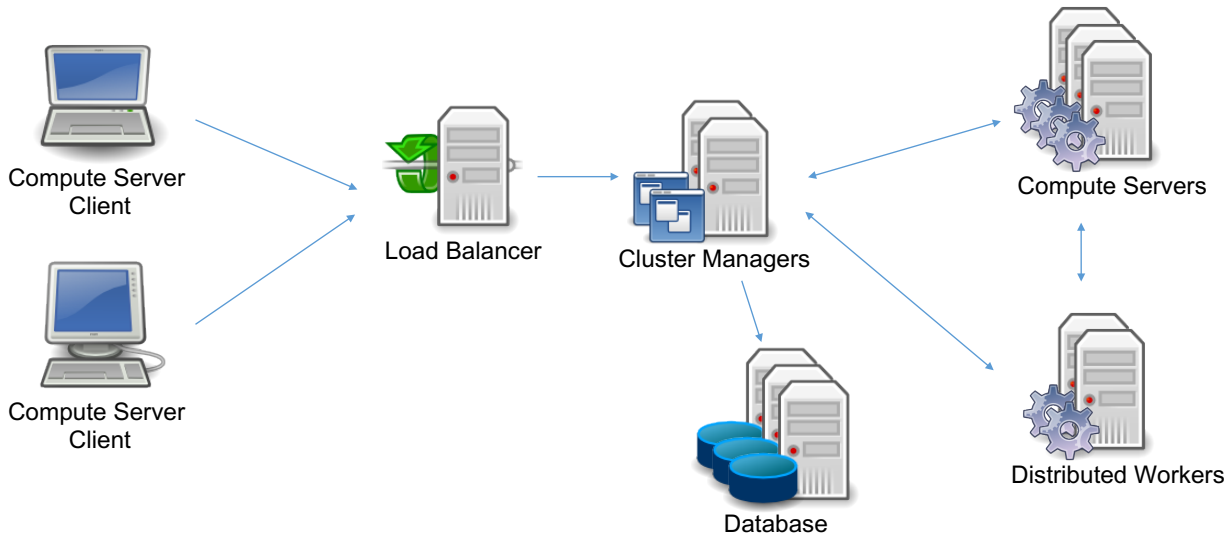
If you need to handle more jobs concurrently, you will need to add more Compute Server nodes. Also, if you want to run distributed algorithms, several Distributed Worker nodes will be needed. To this end, you can deploy one instance of the Cluster Manager (with a Database), and connect those nodes to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

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.

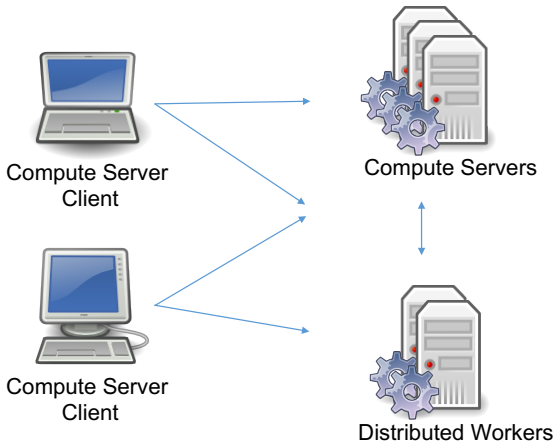
The database itself, as supported by MongoDB, can be deployed in a cluster. In a MongoDB cluster, one of the nodes is chosen dynamically as the primary, while the others are deemed secondary. Secondary nodes replicate the data from the primary node. In the event of a failure of the primary node, the Cluster Manager will choose 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.



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.


Compute Server
Client
Compute Server
Client
Compute Servers
Distributed Workers

2.3 Security

Gurobi Remote Services define a set of user roles to control privileged access. Access to the Cluster Manager is authenticated, and communication can be encrypted. If a Cluster Manager is not deployed, communication can still be encrypted, but access is controlled through a set of predefined passwords instead.

User Roles

Users of Gurobi Remote Services will fall into one of three possible roles: system administrator, administrator, or standard 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.

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.

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.

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.

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.

Authentication

The Cluster Manager authenticates all communication using one of two approaches: 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`.

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.

Applications should use API keys instead. The Cluster Manager lets each user creates an API key, composed of an access ID and a secret key. The user owning the API key or the system administrator can revoke the key at any time. It is also possible to create multiple keys to perform key rotation in your applications.

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. MongoDB also supports encrypted communications and data encryption at rest, if necessary.

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

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.

2.4 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.

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.

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
Using license file /opt/gurobi900/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 9.0.2 build v9.0.2rc0 (linux64)
Copyright (c) 2020, Gurobi Optimization, LLC


...

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.

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.

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. Finally, we present a discussion of the available communication options.

3.1 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 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 you platform and are provided in this section.

2. Install and start a MongoDB database server 4.0 or later (as explained in their on-line guide).

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 9.0.0
   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 license portal to retreive 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 9.0.0
   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 Mac OS, 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 9.0.0
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 will 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
   ```

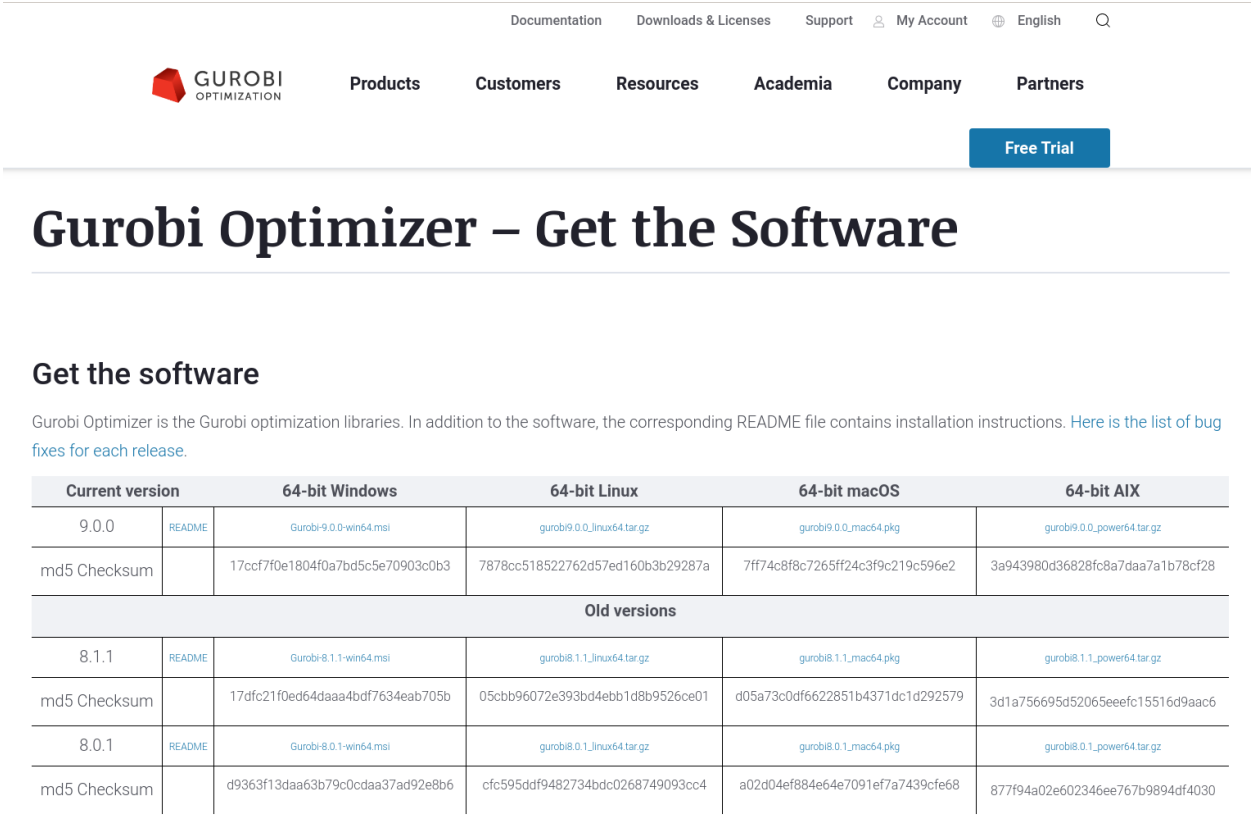
   The followup sections give more details on the command line tools and the programming language APIs.

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

3.2 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:

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_server9.0.2_linux64.tar.gz
```

This command will create a sub-directory `gurobi_server902/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_server902/linux64`.

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_server902/linux64/bin"
```

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

```
setenv PATH "${PATH}:/opt/gurobi_server902/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.

Mac OS Installation

On Mac OS, your next step once you've downloaded the Gurobi Remote Services package from our website (e.g., `gurobi_server9.0.2_mac64.pkg` for Gurobi 9.0.2) is to double-click on the installer and follow the prompts. By default, the installer will place the Gurobi Remote Services 9.0.2 files in `/Library/gurobi_server902/mac64` (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_server902/mac64`.

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-9.0.2-win64.msi` for Gurobi 9.0.2).

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 9.0.2 files in directory `c:/gurobi_server902/win64`. The installer gives you the option to change the installation target. We'll refer to the installation directory as `<installdir>`.

3.3 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.

Installing the Database

The Cluster Manager uses a MongoDB database to store the data associated with several important features: user profiles, API keys, job history, batch definitions, and batch data. You will need to install version 4.0 or later. 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.

Once MongDB 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:

```
> mongo mongodb://localhost:27017
MongoDB shell version v4.0.4
connecting to: mongodb://localhost:27017
MongoDB server version: 4.0.4
```

As noted earlier, you need to install version 4.0 or later. If you try an earlier version, you may see connection error messages like the following when you try to start the Cluster Manager:

```
fatal : Failed to connect to database: server selection error: server selection timeout
current topology: Type: Unknown
```

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
```

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 shared directory (`C:\gurobi`, `/opt/gurobi`, `/Library/gurobi` for Windows, Linux, and Mac OS 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.

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.

`HISTORY_MAX_AGE`: This property indicates how long to keep the jobs and batches in the history. The default is 7 days.

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 9.0.2 (build v9.0.2rc0)
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
```

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 Mac OS, 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:

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.

Mac OS

On Mac OS, the system manager is called `launchd`, and the `install` command will generate a service file in `/Library/LaunchDaemons/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 Mac OS `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.

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.

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 `JWT_EXPIRATION` configuration property (the default is 8 hours). You can display the complete help of the login command using the `--help` flag.

```
> 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.

3.4 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.

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 *Retrieving and Setting Up a Gurobi License* section of the *Gurobi Optimizer Quick Start Guide*.

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 License Center. 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 Mac OS
- 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.

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
```

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 Mac OS 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.

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.

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` 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. This limit is set to 0 by default which means that it is disabled.

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.

```
grb_rs hash newpass
$$ppEieKZExlBR-pCSUMlmc4oWlG8nZsUOE2IMOhJbzsmV_Yjj
```

Then copy and paste the value in the configuration file:

```
PASSWORD=$$ppEieKZExlBR-pCSUMlmc4oWlG8nZsUOE2IMOhJbzsmV_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. 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. 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` 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. This limit is set to 0 by default which means that it is disabled.

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_server902/linux64/bin/grb_rs.cnf
info :  Gurobi Remote Services starting...
info :  Platform is linux
info :  Version is 9.0.2 (build v9.0.2rc0)
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_server902/linux64/bin/data
```

```
info :  Data store created
info :  Available runtimes:  [9.0.0 9.0.0 9.0.2]
info :  Public root is /home/jones/gurobi_server902/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
```

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 Mac OS, 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:

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.

Mac OS

On Mac OS, 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 Mac OS `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.

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.

Verification

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

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...
```

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.

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.

3.5 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.

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==
```

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
```

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
   ```

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.

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.

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.

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.

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 `stop` command:

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

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 `start` command:

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

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.

3.6 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 terminates 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.

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
```

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...

> 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
```

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

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.

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 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 Mac 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.

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 develoeprs 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. Finally, we will describe how the commands can be applied to execute distributed algorithms.

We will discuss later how to program with Remote Services.

4.1 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.

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 your home directory or in one of the following locations:

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

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

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`

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.

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.

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.

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
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
  ```

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.

4.2 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.

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/gurobi900/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 9.0.2 build v9.0.2rc0 (linux64)
Copyright (c) 2020, Gurobi Optimization, LLC


...

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 terminate the job.

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 8.1.1   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 nonzeros 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
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.

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.

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 *Parameters* section of the Gurobi Reference Manual.

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`.

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
```

4.3 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.

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
```

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)
```

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 you batches:

```
> grbcluster jobs
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
```

Retrying Batches

If a batch fails to execute, you can resubmit that batch. This might for example happen if the node 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.

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.

4.4 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.

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/gurobi900/mac64/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
```

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.

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
```

4.5 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.

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
```

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
```

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
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.).

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
```

4.6 Distributed Algorithms

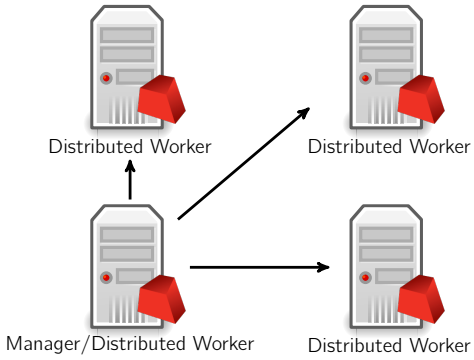
Gurobi Remote Services allow you to perform distributed optimization. All you need is a cluster with more than one node. The nodes can be either Compute Server or Distributed Worker nodes. Ideally these nodes should all give very similar performance. Identical performance is best, especially for distributed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

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.

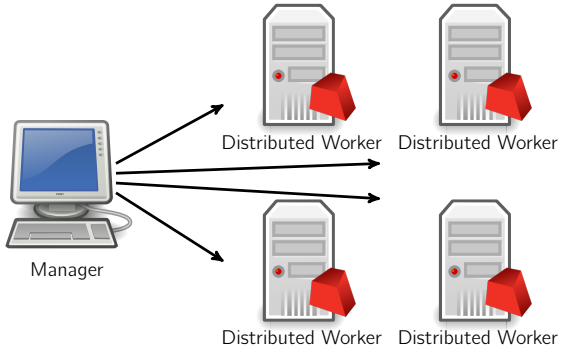
As we mentioned earlier, Distributed Workers do not require Gurobi licenses. You can add any machine to a Remote Services cluster to act as a Distributed Worker. The manager does require a distributed algorithm license (you'll see a `DISTRIBUTED=` line in your license file if distributed algorithms are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

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 three parameters: `DistributedMIPJobs`, `ConcurrentJobs`, and `TuneJobs`, 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.

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.

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 `TuneJobs` to 2 in `grbtune` ...

```
> grbtune TuneJobs=2 misc07.mps
```

... you should see output that looks like the following:

```
Capacity available on 'server1:61000' - connecting...
...
Using Compute Server as first worker
Started distributed worker on server2:61000

Distributed tuning: launched 2 distributed worker jobs
```

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 tuning run completes.

Similarly, if you launch distributed MIP ...

```
> 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
```

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
```

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.

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).

To give a simple example, if you'd like your Python program to offload the optimization computation to a Compute Server named `server1`, you could say:

```
env = Env(empty=True)
env.setParam(GRB.Param.ComputeServer, "server1:61000")
env.setParam(GRB.Param.ServerPassword, "passwd")
env.start()
model = read("misc07.mps", env)
model.optimize()
```

An equivalent Java program would look like this:

```
GRBEnv env = new GRBEnv(true);
env.set(GRB.StringParam.ComputeServer, "server1:61000");
env.set(GRB.StringParam.ServerPassword, "passwd");
env.start();
GRBModel model = new GRBModel(env, "misc07.mps");
model.optimize();
```

We refer you to 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 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
    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++, `GBModel.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. You will only receive `MESSAGE`, `BARRIER`, `SIMPLEX`, `MIP`, `MIPSOL` and `MULTIOBJ` callbacks; you will not receive `PRESOLVE` or `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. You can still request that the optimization be terminated from any of the callbacks you receive, though. Please refer to the `Callback Code` section of the Gurobi Reference Manual 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.

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.

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`.

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.

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` section of the Gurobi Reference Manual) for details on the different callback types.

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).

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 and administer 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 Mac OS
- 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 all 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.

6.3 Administrative Commands

Gurobi Instant Cloud allows you to perform administrative commands (to abort a job, change the job limit, etc.), but you'll need to retrieve the administrator password to do so. The administrator password can be retrieved from the Instant Cloud Manager. For a running machine, you will find the administrator password in the `Machines` area, within machine details under the `PASSWORDS` tab. If you have no running machine, you can find it in the `Settings` area, under passwords settings. The password will be prompted if not specified in the license file.

For example, you can abort a specific job:

```
> grbcluster job abort 3545d9de-8de4-4666-9491-5d60e2e56186
```

Or, you can set the job limit of a specific Compute Server node:

```
> grbcluster node config --server=ip-172-31-62-109  --job-limit=10
```

6.4 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.

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
  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

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
      --manager-insecure      Disable certificate verification if TLS is used
                              to communicate with the Cluster Manager

General Flags:
      --version               Display version information
      --help                  Display usage
```

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 to administrate the jobs. The password can be in clear or can be hashed using 'grb_rs hash' command for better security.

`AWS`: Type bool, use `--aws` to override on the command line. Enable 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. Enable Azure configuration using user-data.

`CLIENT_DETAILS_ADMIN`: Type bool. Indicates client details such as host, IP 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 is displayed to any user running the command line tool grbcluster or the REST API. If this property is set to true, only the administrator will be able to 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 administrate the cluster. The password can be in clear or can be hashed using 'grb_rs hash' command for better security.

`CLUSTER_TOKEN`: Type string. Unique cluster identifier. The token is en encrypted key to let nodes communicates between each other. 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 to evict a node that is DEGRADED from the cluster. 0 for no timeout.

`FILE_DESCRIPTOR_LIMIT`: Type int (default 2048). Maximum number of file descriptors.

FIXED_JOBLIMIT: Type bool. Indicates if the job limit can be changed once the node 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.

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, for example to shutdown 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. Disable 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 for more than the timeout, 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 are first generated in an input directory. The output data is similarly stored in an output data. By default, these directories are deleted once a batch is complete to save space. However, using this property, the files can be kept until the jobs is evicted of the recent history (see MAX_RECENT.)

LICENSEID: Type string. Cloud license ID.

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.

`MANAGER`: Type string, use `--manager` to override on the command line. Cluster Manager URL

`MANAGER_INSECURE`: Type bool. Indicate if connection to manager is using TLS insecure

`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.

`NOQUEUE`: Type bool. Disable job queueing.

`NO_CONSOLE`: Type bool, use `--no-console` to override on the command line. Disable the console log.

`NO_LOCAL_DISK`: Type bool (default true). Indicates if local disk can be used to store node files, solution files etc

`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.

`STRICT_RUNTIME_MATCHING`: Type bool (default true). Indicates if 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.

`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_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.

`USERNAME_ADMIN`: Type bool. Indicates that job username is only accessible as an admin user When a job is submitted, the client process username is recorded. By default, this information is displayed to any user running the command line tool grbcluster or the REST API. If this property is set to true, only the administrator will be able to access this information.

`VERBOSE`: Type bool, use `--verbose` to override on the command line. Enable verbose logging.

`WORKER`: Type bool, use `--worker` to override on the command line. Declare the node as a distributed worker.

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 Helper Commands:
  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

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
      --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

General Flags:
      --version                Display version information
      --help                   Display usage
```

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). Max age of authentication information (seconds)

`CLUSTER_TOKEN`: Type string. Unique cluster identifier. The token is en encrypted key to let the manager communicates with cluster 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

`HISTORY_MAX_AGE`: Type int (default 7). Limit the job history to a number of days.

`HTTP_HEALTH_SERVER`: Type bool. Enable an additional HTTP server of /ping 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 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)

`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)

`PORT`: Type int, use `--port` to override on the command line. Port number for the REST API.

`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_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.

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')


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.
```

Appendix F: gurobi_cl

```
Usage:
  gurobi_cl --help                              Display usage
  gurobi_cl [flags] [param=value]* filename     Optimize a model file
  gurobi_cl [flags]                             Execute a command

Gurobi parameters are documented in the Gurobi Reference Manual.

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:
  -h, --help             Display usage
      --license          Display license information
  -t, --tokens           List license tokens currently in use
  -v, --version          Display version information

Compute Server and Cluster Manager Flags:
      --group=string     Cluster group placement, overrides license file GROUP
      --manager=string   Cluster Manager URL, overrides license file CSMANAGER
  -p, --password=string  Password, overrides license file PASSWORD
                           (default "pass")
      --priority=int     Job priority, overrides license file PRIORITY
                           (default 0, min -100, max 100)
  -r, --router=string    Router URL, overrides license file property ROUTER
  -s, --server=string    Cluster representative node address,
                           overrides license file COMPUTESERVER
      --tls-insecure     Skip TLS certificate verification,
                           overrides license file CSTLSINSECURE
      --username=string  Username for Cluster Manager authentication
                           overrides license file USERNAME

Instant Cloud Flags:
      --accessid=string  Access ID, overrides license file CLOUDACCESSID
      --secretkey=string Secret Key, overrides license file CLOUDKEY
      --pool=string      Pool name, overrides license file POOL

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 some connection parameters (server, password, router). If the
license file references a Gurobi Instant Cloud pool, it will resolve the
connection parameters of the pool.

The server URL can also specify the protocol and the port:
   server.company.com                  Use HTTP on standard port 80
   server.company.com:61000            Use HTTP on port 61000
   https://server.company.com          Use HTTPS on standard port 443
   https://server.company.com:61000    Use HTTPS on port 61000
```

```
If you wish to get the status of your compute server cluster, list the nodes
and the jobs, or check the status of your licenses, please use the grbccluster
command line tool. To learn more about grbcluster, type the following command:
  grbcluster --help

gurobi_cl 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.

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 --server=server.company.com --password=pass misc07.mps

Visit www.gurobi.com/documentation/9.0 for further details on how to use
this program.
```

Appendix G: Acknowledgement of 3rd Party Icons

The icons used in this document come from the Open Security Architecture.

Appendix H: Open Source Component Licenses

In this section, we list the different open source components used in the remote services implementation. For each component, the reference indicates an URL to access the source code or a public NPM module name.

```
Component reference: https://github.com/pkg/errors

Copyright (c) 2015, Dave Cheney <dave@cheney.net>
All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this
  list of conditions and the following disclaimer.

* Redistributions in binary form must reproduce the above copyright notice,
  this list of conditions and the following disclaimer in the documentation
  and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS"
AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR
SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER
CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY,
OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

---------

Component reference: https://gopkg.in/natefinch/lumberjack.v2

The MIT License (MIT)

Copyright (c) 2014 Nate Finch

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the ‘‘Software’’), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.
```

THE SOFTWARE IS PROVIDED ‘‘AS IS’’, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Component reference: https://github.com/aws/aws-sdk-go
Component reference: https://github.com/spf13/cobra

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes

of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

   (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices stating that You changed the files; and

(c) You must retain, in the Source form of any Derivative Works
that You distribute, all copyright, patent, trademark, and
attribution notices from the Source form of the Work,
excluding those notices that do not pertain to any part of
the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its
distribution, then any Derivative Works that You distribute must
include a readable copy of the attribution notices contained
within such NOTICE file, excluding those notices that do not
pertain to any part of the Derivative Works, in at least one
of the following places: within a NOTICE text file distributed
as part of the Derivative Works; within the Source form or
documentation, if provided along with the Derivative Works; or,
within a display generated by the Derivative Works, if and
wherever such third-party notices normally appear. The contents
of the NOTICE file are for informational purposes only and
do not modify the License. You may add Your own attribution
notices within Derivative Works that You distribute, alongside
or as an addendum to the NOTICE text from the Work, provided
that such additional attribution notices cannot be construed
as modifying the License.

You may add Your own copyright statement to Your modifications and
may provide additional or different license terms and conditions
for use, reproduction, or distribution of Your modifications, or
for any such Derivative Works as a whole, provided Your use,
reproduction, and distribution of the Work otherwise complies with
the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise,
any Contribution intentionally submitted for inclusion in the Work
by You to the Licensor shall be under the terms and conditions of
this License, without any additional terms or conditions.
Notwithstanding the above, nothing herein shall supersede or modify
the terms of any separate license agreement you may have executed
with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade
names, trademarks, service marks, or product names of the Licensor,
except as required for reasonable and customary use in describing the
origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or
agreed to in writing, Licensor provides the Work (and each
Contributor provides its Contributions) on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or
implied, including, without limitation, any warranties or conditions
of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A
PARTICULAR PURPOSE. You are solely responsible for determining the
appropriateness of using or redistributing the Work and assume any
risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory,
whether in tort (including negligence), contract, or otherwise,
unless required by applicable law (such as deliberate and grossly

```
      negligent acts) or agreed to in writing, shall any Contributor be
      liable to You for damages, including any direct, indirect, special,
      incidental, or consequential damages of any character arising as a
      result of this License or out of the use or inability to use the
      Work (including but not limited to damages for loss of goodwill,
      work stoppage, computer failure or malfunction, or any and all
      other commercial damages or losses), even if such Contributor
      has been advised of the possibility of such damages.

   9. Accepting Warranty or Additional Liability. While redistributing
      the Work or Derivative Works thereof, You may choose to offer,
      and charge a fee for, acceptance of support, warranty, indemnity,
      or other liability obligations and/or rights consistent with this
      License. However, in accepting such obligations, You may act only
      on Your own behalf and on Your sole responsibility, not on behalf
      of any other Contributor, and only if You agree to indemnify,
      defend, and hold each Contributor harmless for any liability
      incurred by, or claims asserted against, such Contributor by reason
      of your accepting any such warranty or additional liability.

   END OF TERMS AND CONDITIONS

   APPENDIX: How to apply the Apache License to your work.

      To apply the Apache License to your work, attach the following
      boilerplate notice, with the fields enclosed by brackets "[]"
      replaced with your own identifying information. (Don't include
      the brackets!)  The text should be enclosed in the appropriate
      comment syntax for the file format. We also recommend that a
      file or class name and description of purpose be included on the
      same "printed page" as the copyright notice for easier
      identification within third-party archives.

   Copyright [yyyy] [name of copyright owner]

   Licensed under the Apache License, Version 2.0 (the "License");
   you may not use this file except in compliance with the License.
   You may obtain a copy of the License at

       http://www.apache.org/licenses/LICENSE-2.0

   Unless required by applicable law or agreed to in writing, software
   distributed under the License is distributed on an "AS IS" BASIS,
   WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
   See the License for the specific language governing permissions and
   limitations under the License.

---------

Component reference: https://github.com/boltdb/bolt

The MIT License (MIT)

Copyright (c) 2013 Ben Johnson

Permission is hereby granted, free of charge, to any person obtaining a copy of
```

this software and associated documentation files (the ‘‘Software’’), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED ‘‘AS IS’’, WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Component reference: https://github.com/julienschmidt/httprouter

Copyright (c) 2013 Julien Schmidt. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are met:
    * Redistributions of source code must retain the above copyright
      notice, this list of conditions and the following disclaimer.
    * Redistributions in binary form must reproduce the above copyright
      notice, this list of conditions and the following disclaimer in the
      documentation and/or other materials provided with the distribution.
    * The names of the contributors may not be used to endorse or promote
      products derived from this software without specific prior written
      permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS ‘‘AS IS’’ AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL JULIEN SCHMIDT BE LIABLE FOR ANY
DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Component reference: https://github.com/kardianos/osext

Copyright (c) 2012 The Go Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

   * Redistributions of source code must retain the above copyright

notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
   * Neither the name of Google Inc. nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Component reference: https://github.com/satori/go.uuid

Copyright (C) 2013-2016 by Maxim Bublis <b@codemonkey.ru>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Component reference: https://github.com/shirou/gopsutil

gopsutil is distributed under BSD license reproduced below.

Copyright (c) 2014, WAKAYAMA Shirou
All rights reserved.

Redistribution and use in source and binary forms, with or without modification,

```
are permitted provided that the following conditions are met:

 * Redistributions of source code must retain the above copyright notice, this
   list of conditions and the following disclaimer.
 * Redistributions in binary form must reproduce the above copyright notice,
   this list of conditions and the following disclaimer in the documentation
   and/or other materials provided with the distribution.
 * Neither the name of the gopsutil authors nor the names of its contributors
   may be used to endorse or promote products derived from this software without
   specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED
WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE
DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR
ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES
(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON
ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS
SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.


-------
internal/common/binary.go in the gopsutil is copied and modifid from golang/encoding/binary.go.



Copyright (c) 2009 The Go Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

   * Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
   * Neither the name of Google Inc. nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
```

Component reference: https://github.com/spf13/pflag

Component reference: https://github.com/urfave/negroni

OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: https://go.googlesource.com/sys

Copyright (c) 2009 The Go Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions are
met:

   * Redistributions of source code must retain the above copyright
notice, this list of conditions and the following disclaimer.
   * Redistributions in binary form must reproduce the above
copyright notice, this list of conditions and the following disclaimer
in the documentation and/or other materials provided with the
distribution.
   * Neither the name of Google Inc. nor the names of its
contributors may be used to endorse or promote products derived from
this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
"AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR
A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT
OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT
LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE,
DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY
THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT
(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE
OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Component reference: https://github.com/Showmax/go-fqdn

Copyright since 2015 Showmax s.r.o.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Component reference: https://github.com/phyber/negroni-gzip

The MIT License (MIT)

Copyright (c) 2013 Jeremy Saenz
              2014 David O'Rourke

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Component reference: https://github.com/dgrijalva/jwt-go

Copyright (c) 2012 Dave Grijalva

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Component reference: https://github.com/inconshreveable/mousetrap

Copyright 2014 Alan Shreve

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

```
   http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.



---------

Component reference: https://github.com/StackExchange/wmi

The MIT License (MIT)

Copyright (c) 2013 Stack Exchange

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of
the Software, and to permit persons to whom the Software is furnished to do so,
subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS
FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR
COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER
IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.



---------

Component reference: https://github.com/go-ole/go-ole

The MIT License (MIT)

Copyright © 2013-2017 Yasuhiro Matsumoto, <mattn.jp@gmail.com>

Permission is hereby granted, free of charge, to any person obtaining a copy of
this software and associated documentation files (the "Software"), to deal in
the Software without restriction, including without limitation the rights to
use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies
of the Software, and to permit persons to whom the Software is furnished to do
so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
```

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: https://github.com/mongodb/mongo-go-driver.git

Apache License
Version 2.0, January 2004
http://www.apache.org/licenses/

TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

1. Definitions.

"License" shall mean the terms and conditions for use, reproduction,
and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by
the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all
other entities that control, are controlled by, or are under common
control with that entity. For the purposes of this definition,
"control" means (i) the power, direct or indirect, to cause the
direction or management of such entity, whether by contract or
otherwise, or (ii) ownership of fifty percent (50%) or more of the
outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity
exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications,
including but not limited to software source code, documentation
source, and configuration files.

"Object" form shall mean any form resulting from mechanical
transformation or translation of a Source form, including but
not limited to compiled object code, generated documentation,
and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or
Object form, made available under the License, as indicated by a
copyright notice that is included in or attached to the work
(an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object
form, that is based on (or derived from) the Work and for which the
editorial revisions, annotations, elaborations, or other modifications
represent, as a whole, an original work of authorship. For the purposes
of this License, Derivative Works shall not include works that remain

separable from, or merely link (or bind by name) to the interfaces of,
the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including
the original version of the Work and any modifications or additions
to that Work or Derivative Works thereof, that is intentionally
submitted to Licensor for inclusion in the Work by the copyright owner
or by an individual or Legal Entity authorized to submit on behalf of
the copyright owner. For the purposes of this definition, "submitted"
means any form of electronic, verbal, or written communication sent
to the Licensor or its representatives, including but not limited to
communication on electronic mailing lists, source code control systems,
and issue tracking systems that are managed by, or on behalf of, the
Licensor for the purpose of discussing and improving the Work, but
excluding communication that is conspicuously marked or otherwise
designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity
on behalf of whom a Contribution has been received by Licensor and
subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
copyright license to reproduce, prepare Derivative Works of,
publicly display, publicly perform, sublicense, and distribute the
Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of
this License, each Contributor hereby grants to You a perpetual,
worldwide, non-exclusive, no-charge, royalty-free, irrevocable
(except as stated in this section) patent license to make, have made,
use, offer to sell, sell, import, and otherwise transfer the Work,
where such license applies only to those patent claims licensable
by such Contributor that are necessarily infringed by their
Contribution(s) alone or by combination of their Contribution(s)
with the Work to which such Contribution(s) was submitted. If You
institute patent litigation against any entity (including a
cross-claim or counterclaim in a lawsuit) alleging that the Work
or a Contribution incorporated within the Work constitutes direct
or contributory patent infringement, then any patent licenses
granted to You under this License for that Work shall terminate
as of the date such litigation is filed.

4. Redistribution. You may reproduce and distribute copies of the
Work or Derivative Works thereof in any medium, with or without
modifications, and in Source or Object form, provided that You
meet the following conditions:

   (a) You must give any other recipients of the Work or
       Derivative Works a copy of this License; and

   (b) You must cause any modified files to carry prominent notices
       stating that You changed the files; and

   (c) You must retain, in the Source form of any Derivative Works

that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

(d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be

liable to You for damages, including any direct, indirect, special,
incidental, or consequential damages of any character arising as a
result of this License or out of the use or inability to use the
Work (including but not limited to damages for loss of goodwill,
work stoppage, computer failure or malfunction, or any and all
other commercial damages or losses), even if such Contributor
has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing
the Work or Derivative Works thereof, You may choose to offer,
and charge a fee for, acceptance of support, warranty, indemnity,
or other liability obligations and/or rights consistent with this
License. However, in accepting such obligations, You may act only
on Your own behalf and on Your sole responsibility, not on behalf
of any other Contributor, and only if You agree to indemnify,
defend, and hold each Contributor harmless for any liability
incurred by, or claims asserted against, such Contributor by reason
of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

APPENDIX: How to apply the Apache License to your work.

To apply the Apache License to your work, attach the following
boilerplate notice, with the fields enclosed by brackets "[]"
replaced with your own identifying information. (Don't include
the brackets!) The text should be enclosed in the appropriate
comment syntax for the file format. We also recommend that a
file or class name and description of purpose be included on the
same "printed page" as the copyright notice for easier
identification within third-party archives.

Copyright [yyyy] [name of copyright owner]

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.

Component reference: @date-io/*

MIT License

Copyright (c) 2017 Dmitriy Kovalenko

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal

in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: @material-ui/*

The MIT License (MIT)

Copyright (c) 2014 Call-Em-All

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: @axios/*

Copyright (c) 2014-present Matt Zabriskie

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in

all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN
THE SOFTWARE.

Component reference: babel-polyfill

MIT License

Copyright (c) 2014-present Sebastian McKenzie and other contributors

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Component reference: classnames

The MIT License (MIT)

Copyright (c) 2018 Jed Watson

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR

IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: js-cookie

MIT License

Copyright (c) 2018 Copyright 2018 Klaus Hartl, Fagner Brack, GitHub Contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: jss

The MIT License (MIT)
Copyright (c) 2014-present Oleg Isonen (Slobodskoi) & contributors

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: moment

Copyright (c) JS Foundation and other contributors

Permission is hereby granted, free of charge, to any person
obtaining a copy of this software and associated documentation
files (the "Software"), to deal in the Software without
restriction, including without limitation the rights to use,
copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the
Software is furnished to do so, subject to the following
conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES
OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING
FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR
OTHER DEALINGS IN THE SOFTWARE.

Component reference: prop-types

MIT License

Copyright (c) 2013-present, Facebook, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: query-string

MIT License

Copyright (c) Sindre Sorhus <sindresorhus@gmail.com> (sindresorhus.com)

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: react
Component reference: react-dom
Component reference: @hot-loader/react-dom

MIT License

Copyright (c) Facebook, Inc. and its affiliates.

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: react-hot-loader

MIT License

Copyright (c) 2016 Dan Abramov

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: react-router-dom

MIT License

Copyright (c) React Training 2016-2018

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: recharts

The MIT License (MIT)

Copyright (c) 2015 recharts

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.

Component reference: swagger-ui
Component reference: swagger-ui-dist

Copyright 2019 SmartBear Software

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
[apache.org/licenses/LICENSE-2.0](http://www.apache.org/licenses/LICENSE-2.0)

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.