Administrator Manual for
CLC Server 23.0.5
Windows, macOS and Linux

September 19, 2023

This software is for research purposes only.
## Contents

1 Introduction .......................................................... 7  
   1.1 System requirements ........................................... 10  
   1.2 Licensing ......................................................... 11  
   1.3 CLC Genomics Server ......................................... 12

2 Installation ......................................................... 17  
   2.1 Quick installation guide ....................................... 17  
   2.2 Installing and running the Server ........................... 17  
   2.3 Installation modes - console and silent .................... 19  
   2.4 Upgrading an existing installation .......................... 20  
   2.5Allowing access through your firewall ...................... 21  
   2.6 Downloading a license ........................................ 22  
   2.7 Starting and stopping the server ............................ 24

3 Basic configuration ............................................... 27  
   3.1 Logging into the administrative interface ................. 27  
   3.2 Adding locations for saving data ............................ 27  
   3.3 Changing the listening port ................................ 32  
   3.4 Changing the tmp directory .................................. 35  
   3.5 Setting the amount of memory available for the JVM ...... 36  
   3.6 Limiting the number of cpus available for use ............ 36  
   3.7 HTTP settings .................................................... 36  
   3.8 Deployment of server information to CLC Workbenches .... 36

4 Managing users and groups ..................................... 37  
   4.1 Logging in the first time and changing the root password 37
## CONTENTS

4.2 User authentication using the web interface ........................................ 37  
4.3 User authentication via the Workbench for built-in authentication ......... 42  

5 Access privileges and permissions ....................................................... 44  
5.1 Controlling group access to CLC Server data .................................. 44  
5.2 Controlling access to the server, server tasks and external data ......... 47  
5.3 Customized attributes on data locations ........................................... 49  

6 Job distribution .................................................................................... 55  
6.1 Introduction to servers setups ......................................................... 55  
6.2 Model I: Master server with dedicated job nodes ......................... 56  
6.3 Model II: Master server submitting to grid nodes ....................... 60  
6.4 Model III: Single Server setup ...................................................... 74  
6.5 Workflow queuing options ........................................................ 75  
6.6 Job running options ........................................................................ 78  

7 Access to files outside CLCServer Locations ...................................... 82  
7.1 Import/export directories ........................................................ 82  
7.2 Direct data transfer from client systems ....................................... 84  
7.3 AWS Connections in the CLC Server ............................................. 85  
7.4 Illumina BaseSpace ......................................................................... 86  

8 Workflows ........................................................................................ 87  
8.1 Installing and configuring workflows ............................................ 87  
8.2 Executing workflows ..................................................................... 89  
8.3 Updating workflows ..................................................................... 90  

9 BLAST databases ................................................................................ 94  
9.1 Adding directories for BLAST databases on the Server ............. 94  
9.2 Adding and removing BLAST databases ..................................... 95  

10 Status and management ................................................................... 97  
10.1 Downloading a license via the web interface ............................. 97  
10.2 Server maintenance ................................................................. 99  
10.3 User statistics ............................................................................... 100
19 Command line tools

20 Appendix

20.1 Use of multi-core computers .................................................. 183
20.2 Non-exclusive Algorithms ..................................................... 184
20.3 DRMAA libraries ................................................................. 187
20.4 Consumable Resources .......................................................... 188
20.5 Server system communication diagrams ................................... 190
20.6 Third party libraries ............................................................... 192
20.7 External network connections .................................................. 192
20.8 Read Mapper Reference Caching .............................................. 193
20.9 Monitoring ................................................................. 194

Bibliography

Index
Chapter 1

Introduction

Welcome to CLC Server 23.0.5, a central element of the CLC product line enterprise solutions.

The latest version of this user manual can be found in pdf and html formats at https://digitalinsights.qiagen.com/technical-support/manuals/.

An overview of the server solution is shown in figure 1.1. The software depicted here is for research purposes only.
Figure 1.1: An overview of the CLC Server solution. Scalability includes running analyses on nodes in an existing grid setup (via DRMAA), running analyses on job nodes, and, when the Cloud Server Plugin is installed and a CLC Genomics Cloud setup is available, running jobs using AWS Batch.

Using server software means that data can be stored centrally and analyses run on compute resources other than a personal computer. After logging in from a CLC Workbench, data in CLC Server data locations are listed in the Workbench’s Navigation Area and analyses can be run on the CLC Workbench or on the CLC Server.

After a job is launched to run on the CLC Server, the CLC Workbench can be used for other work, or it can be closed. The status of jobs submitted via a CLC Workbench is shown in the Workbench’s Processes tab, either in the same working session, or after start up if the Workbench was closed before the analysis completed.

Diagrams summarizing some typical CLC Server setups can be found in section 20.5.


Administering a CLC Server
The CLC Server is administered through a web client. This client is intended primarily for administration tasks, although non-admin users can log in and see listings of the data they have access to via the CLC Server.

There are 4 top level tabs visible when an administrator logs into the client, each containing a set of subtabs for tasks related to that area (figures 1.2 and 1.3).

Figure 1.2: Administration and management of a CLC Server is done primarily via the web administrative client. The contents of the three tabs containing primary administrative functionality are shown here.

The rest of this manual covers how to install and configure a CLC Server, and how to expand its functionality through the installation of plugins, workflows, and provision of external applications.

Some CLC Server administration tasks can also be carried out using the CLC Server Command Line Tools. A full list of tools commands available can be listed using the clcserver command, as described at https://resources.qiagenbioinformatics.com/manuals/clcservercommandlinetools/current/index.php?manual=Basic_usage.html.
Figure 1.3: Administration actions relating to data stored in CLC Server file locations is done under the Element info tab. Standard users also have access to this tab, but admin-related functionality is not made visible in that case.

1.1 System requirements

The system requirements of CLC Server are:

**Server system**

- Mac: macOS 11, 12, and 13
- Linux: RHEL 7 and later, SUSE Linux Enterprise Server 12 and later. The software is expected to run without problem on other recent Linux systems, but we do not guarantee this. To use BLAST related functionality, libnsl.so.1 is required.
- 64 bit
- For CLC Server setups that include job nodes and grid nodes, those nodes must run the same type of operating system as the master CLC Server.
- File system that supports file locking

**Server hardware requirements**

- 16 GB RAM recommended (8 GB RAM required)
- Disk space: 500 GB required. More needed if large amounts of data are analyzed.

**Performance note:** Tools that take advantage of multiple cores do not scale linearly with high numbers of cores. If you plan to use a large system (>64 cores), a setup with job nodes running
on virtual machines provides the potential to use more of the compute capacity. On a job node setup, analyses can be run in parallel, with appropriate cpu limits configurable for each node, as described in section 6.2.3.

**Memory and CPU settings for mapping reads**

For mapping reads to the human genome (3.2 gigabases), or genomes of a similar size, 16 GB RAM is required. Smaller systems can be used when mapping to small genomes.

Larger amounts of memory can help the overall speed of the analysis when working with large datasets, but little gain is expected above about 32 GB of RAM.

Increasing the number of cpus can decrease the time a read mapping takes, however performance gain is expected to be limited above approximately 40 threads.

**Special requirements for de novo assembly**

De novo assembly may need more memory than stated above - this depends both on the number of reads and the complexity and size of the genome. See http://resources.qiagenbioinformatics.com/white-papers/White_paper_on_de_novo_assembly_4.pdf for examples of the memory usage of various data sets.

**Special requirement for the shared filesystem used by the job node setup or grid integration**

The file locking mechanism is required to ensure that all nodes see the latest version of the data stored on the shared filesystem.

**Special requirements for containerized external applications**

Containerized external applications\(^1\) are supported for Unix images run on Unix hosts only. Windows-based images and Windows hosts are not supported. Standard external applications are supported for any CLC Server setup.

### 1.2 Licensing

Three kinds of license can be involved in running analyses on the CLC Server.

- **A license for the server software itself.** This is needed for running analyses via the server. The license will allow a certain number of open sessions. This refers to the number of active, individual log-ins from server clients such as Workbenches, the Command Line Tools, or the web interface to the server. The number of sessions is part of the agreement with QIAGEN when you purchase a license. Information about downloading and installing server licenses is provided in the Installation chapter, in section 2.6.

- **A license for the Workbench software.** CLC Workbenches are client software used to launch analyses on the server and to view the results. Find the user manuals and deployment manual for the Workbenches at https://digitalinsights.qiagen.com/technical-support/manuals/.

\(^1\)Containerized external applications were introduced in CLC Server 21.0.
• A network license if you will be submitting analyses to grid nodes. This is explained in detail in section 6.3.5.

1.3 CLC Genomics Server

The CLC Genomics Server is shipped with the following tools and analyses that can all be started from CLC Genomics Workbench and CLC Server Command Line Tools:

- Import
- Export
- Search for Reads in SRA
- Download Genomes and References management
- Classical Sequence Analysis
  - Create Alignment
  - K-mer Based Tree Construction
  - Create Tree
  - Model Testing
  - Maximum Likelihood Phylogeny
  - Extract Sequences
  - Motif Search
  - Translate to Protein
  - Convert DNA to RNA
  - Convert RNA to DNA
  - Reverse Complement Sequence
  - Find Open Reading Frames
  - Download Pfam Database
  - Pfam Domain Search
  - Find and Model Structure
- Molecular Biology Tools
  - Trim Sequences
  - Assemble Sequences
  - Assemble Sequences to Reference
  - Secondary Peak Calling
  - Find Binding Sites and Create Fragments
  - Add attB Sites
  - Create Entry clone (BP)
  - Create Expression clone (LR)
• BLAST
  – BLAST
  – BLAST at NCBI
  – Download BLAST Databases
  – Create BLAST Database

• Prepare Sequencing Data
  – QC for Sequencing Reads
  – Trim Reads
  – Demultiplex Reads

• Quality Control
  – QC for Targeted Sequencing
  – Target Region Coverage Analysis
  – QC for Read Mapping
  – Whole Genome Coverage Analysis
  – Combine Reports
  – Create Sample Report

• Resequencing Analysis
  – Map Reads to Reference
  – Local Realignment
  – Merge Read Mappings
  – Remove Duplicate Mapped Reads
  – Extract Consensus Sequence
  – Create Consensus Sequences from Variants
  – Basic Variant Detection
  – Fixed Ploidy Variant Detection
  – Low Frequency Variant Detection
  – InDels and Structural Variants
  – Identify Known Mutations from Mappings
  – Copy Number Variant Detection (CNVs)
  – Filter against Known Variants
  – Remove Marginal Variants
  – Remove Homozygous Reference Variants
  – Remove Variants Present in Control Reads
  – Annotate from Known Variants
  – Remove Information from Variants
  – Annotate with Conservation Scores
– Annotate with Exon Numbers
– Annotate with Flanking Sequences
– Annotate with Repeat and Homopolymer Information
– Identify Enriched Variants in Case vs Control Samples
– Identify Shared Variants
– Trio Analysis
– Create Variant Track Statistics Report
– Amino Acid Changes
– Predict Splice Site Effect
– GO Enrichment Analysis
– Download 3D Protein Structure Database
– Link Variants to 3D Protein Structure

• RNA-Seq and Small RNA Analysis
  – Create Expression Browser
  – Quantify miRNA
  – Annotate with RNAcentral Accession Numbers
  – Create Combined miRNA Report
  – Extract IsomiR Counts
  – RNA-Seq Analysis
  – Detect and Refine Fusion Genes
  – PCA for RNA-Seq
  – Create Heat Map for RNA-Seq
  – Create K-medoids Clustering for RNA-Seq
  – Differential Expression in Two Groups
  – Differential Expression for RNA-Seq
  – Create Venn Diagram for RNA-Seq
  – Gene Set Test

• Microarray Analysis
  – Create Box Plot
  – Principal Component Analysis
  – Proportion-based Statistical Analysis
  – Gaussian Statistical Analysis
  – Create MA Plot
  – Create Scatter Plot
  – Create Histogram

• Epigenomics Analysis
  – Histone ChiP-Seq
- Transcription Factor ChIP-Seq
  - Annotate with Nearby Gene Information
  - Map Bisulfite Reads to Reference
- Call Methylation Levels
- Create RRBS-fragment Track
- Learn Peak Shape Filter
- Apply Peak Shape Filter
- Score Regions

• De Novo Sequencing
  - De Novo Assembly
  - Map Reads to Contigs

• Utility Tools
  - Extract Annotated Regions
  - Merge Overlapping Pairs
  - Extract Reads
  - Merge Annotation Tracks
  - Merge Variant Tracks
  - Convert to Tracks
  - Convert from Tracks
  - Filter on Custom Criteria
  - Annotate with Overlap Information
  - Filter Annotations on Name
  - Filter Based on Overlap
  - Create GC Content Graph
  - Create Mapping Graph
  - Identify Graph Threshold Areas
  - Update Sequence Attributes in Lists
  - Split Sequence List
  - Subsample Sequence List
  - Rename Elements
  - Rename Sequences in Lists

• Legacy Tools
  - QIAGEN GeneReader Sequencing Import (legacy)
The functionality of the *CLC Genomics Server* can be extended by installation of Server plugins. This is described further in chapter 13.

**Latest improvements**

*CLC Genomics Server* is under constant development and improvement. A detailed list that includes a description of new features, improvements, bugfixes, and changes for the current version of *CLC Genomics Server* can be found at:

Chapter 2

Installation

2.1 Quick installation guide

The following describes briefly the minimum steps needed to set up a CLC Server, ready for client software to connect to. Links out to more detailed explanations for each step are provided.

1. Download the CLC Server installer and run it. When prompted during the installation process, choose to start the server (section 2.2).

2. Log into the server web administrative interface using a web browser. Use the username root and password default (section 3).

3. Install a license for the software (section 2.6).

4. Restart the server (section 2.7).

5. Ensure the necessary port is open for access by client software (section 2.5).

6. Change the root password (section 4.1).

7. Configure the authentication mechanism (section 4.2). If using the built-in authentication system, users and groups can be set up now or later.

8. Add data locations (section 3.2).

9. Check your server setup using the check setup link in the upper right corner (section 18.1).

10. For job node and grid node setups, configure these as described in chapter 6. After setting up execution nodes, run the check setup tool again and review the report.

If the diagnostic report generated by the check setup tool does not reveal any problems, your CLC Server should be ready for use.

2.2 Installing and running the Server

This chapter covers the initial steps required to install and start the CLC Server. These steps should be straightforward, but if you do run into problems, please refer to the troubleshooting chapter 18.
An overview of the basic steps needed to install and configure a CLC Server is provided in section 2.1.

2.2.1 Installing the Server software

The installation can only be performed by a user with administrative privileges. On some operating systems, you can double click on the installer file icon to begin installation. Depending on your operating system you may be prompted for your password or asked to allow the installation to be performed.

- On Windows systems, you may need to right click on the installer file icon, and choose to Run as administrator.
- On Linux, you would normally install to a central location, which will involve running the installation script as an administrative user - either by logging in as one, or by prefacing the command with sudo. Please check that the installation script has executable permissions before executing it.

Next, you will be asked where to install the software (figure 2.1). If the installer has detected that a version of the software is already installed, an option to update the existing installation or install to a different directory will be offered (figure 2.2). For standard upgrades of the CLC Server, you should choose to update the existing installation.

The directory the software is installed into will be referred to as the server installation directory throughout the rest of this manual.

![Figure 2.1: If installing for the first time, the location to install to is specified.](image)

The installer allows you to specify the maximum amount of memory the CLC Server will be able to utilize (figure 2.3). The range of choice depends on the amount of memory installed on your system and on the type of machine used. If you do not have a reason to change this value, leave it at the default setting.

If you are installing on a Linux system, you are offered the option to specify a user account that will be used to run the CLC Server process. Having a specific, non-root user for this purpose is generally recommended. On a standard setup, this would have the effect of adding this username to the service scripts, which can then be used for starting up and shutting down the CLC Server.
service and setting the ownership of the files in the installation area. Downstream, the user running the CLC Server process will own files created in File Locations, for example, after data import or data analyses.

If you are installing the server on a Windows system you will be able to choose if the service is started manually or automatically by the system.

The installer now extracts the necessary files.

On a Windows system, if you have chosen that the service should be started automatically, the service should also start running at this point. On Linux or Mac, if you have chosen the option to start the system at the end of installation, the service should also have started running. Please note that if you do not already have a license file installed, then the CLC Server process will be running in a limited capacity at this point. Downloading a license is described in section 2.6.

Information on stopping and starting the CLC Server service is provided in section 2.7.

2.3 Installation modes - console and silent

Two installation modes are available to support efficient installation of the Workbench software.

- **Console mode** This mode is particularly useful when installing Workbenches onto remote
systems. On Linux, this mode is enabled by using the option `-c` when launching the installer from the command line. On Windows the option is `-console`.

- **Silent mode** This mode supports hands-off installation. Default answers to all prompts are used, although a non-default installation directory can specified if desired (see below). Silent mode is activated using the `-q` parameter when launching the installer from the command line. On Windows, the `-console` option can be appended after `-q`, that is, as the second parameter, to ensure output to the console.

If desired, you can **specify the directory to install the software to** when running the installer in silent mode. Do this adding the `-dir` option to the command line.

On Windows, the `-console` and the `-dir` options only work when the installer is run in silent mode.

The following is an example of a command that would install a Workbench into the directory "c:\bioinformatics\clc" on a Windows system using silent mode with console output:

```
CLCMainWorkbench_21_0_4_64.exe -console -q -dir "c:\bioinformatics\clc"
```

On a Linux system, a similar command to install into the directory "/opt/clcgenomicsworkbench21" could look like:

```
./CLCGenomicsWorkbench_21_0_4_64.sh -q -c -dir /opt/clcgenomicsworkbench21
```

The `-q` and the `-console` options work for the uninstall program as well.

### 2.4 Upgrading an existing installation

For a single CLC Server, the steps we recommend when upgrading to a new version are:

- Stop the CLC Server service after making sure that nobody is using the server. Mechanisms to help with this, including sending a message to users logged into the CLC Server, can be found in section 10.2. Getting information about who is logged in is described in section 10.3.

- Install the CLC Server software in the same directory the existing version was installed in. All settings will be maintained, for example, the locations data are stored, Import/Export directories, BLAST database locations, Users and Groups, and External Application settings.

If you have a job node setup, you will also need to upgrade the CLC Server software on each job node. Upgrading the software itself on each node is all you need to do. Configurations for job nodes, as well as new or updated plugins, are pushed to them by the master node.

When upgrading between major versions, the extra steps described in section 2.4.1) must also be taken. Major version lines are denoted by the first number in the version. For example, upgrading from software with version 21.0 to version 22.0 involves an upgrade to a new major version line.
2.4.1 Upgrading between major versions

There are a few extra steps needed when upgrading to a new major version line beyond those outlined in section 2.4.

1. Download a license file valid for the new major version on a single server, or on the master server in a job node or grid node setup, if this has not already been done. See section 2.6.

2. Delete old license files. License files can be found in a folder called "licenses" under the installation area of the CLC Server.

3. Restart the CLC Server service. See section 2.7.

4. All users of client software (CLC Workbenches and the CLC Server Command Line Tools) must upgrade their software. Corresponding and compatible software versions are listed at the bottom of the Latest Improvement listings for a given server version. e.g. for the latest release, this can be found at: https://digitalinsights.qiagen.com/products/qiagen-clc-genomics-server/latest-improvements/current-line/.

   • All plugins installed on the CLC Server need to be updated. See section 13.
   • On job nodes, any new tools included in the server upgrade will need to be enabled for the nodes you wish them to be run on. New tools are initially disabled on all job nodes to avoid interfering with a setup where certain nodes are dedicated to running specific types of jobs. Read more about enabling tools on job nodes in section 6.2.3.

Configuration updates required when upgrading to CLC Genomics Server 22.0 or higher

If you have configured custom listening port or SSL settings, these need to be reconfigured after upgrading to CLC Server 22.0 and above from older release lines (21.x and earlier), due to an update to Tomcat.

   • Configuring a custom listening port is described in section 3.3.
   • Enabling SSL is described in section 3.3.1.

When upgrading, a backup of your previous configuration will have been saved to conf/server.xml.backup, under the installation area. This file can be referred to for information relevant for configuring the new conf/server.xml file.

2.5 Allowing access through your firewall

By default, the CLC Server listens for TCP-connections on port 7777. Information on changing the port is provided in section 3.3.

If you are running a firewall on your server system you will have to allow incoming TCP-connections on this port before clients can contact the CLC Server. Consult the documentation of your firewall for information on how to do this.

Besides the public port described above the CLC Server also uses an internal port on 7776. There is no need to allow incoming connections from client machines to this port.
2.6 Downloading a license

Licenses are installed on a single server or on the master node of a job node or grid node setup.

To download and install a license:

- Log into the web administrative interface of the single server or master node as an administrative user.

  When there is no existing license file for the CLC Server, or there is a license file that is not valid for the version being run, a warning message is shown (figure 10.2). In these cases, a license file needs to be downloaded.

  When a valid license is present, the messages in yellow will not be present (figure 10.3). In this case, no further action to update the CLC Server license is needed. You can skip the next steps in this section.

- Under the Management tab, open the Download License tab.

- Enter the Order ID supplied by QIAGEN into the Order ID field and click on the “Download and Install License...” button (figure 10.1).

Please contact ts-bioinformatics@qiagen.com if you have not received an Order ID.

The CLC Server must be restarted for new license files to be loaded. Details about restarting can be found in section 2.7.1.

Each time you download a license file, a new file is created in the licenses folder under the CLC Server installation area. If you are upgrading an existing license file, delete the old file from this area before restarting.

If you are working on a system that does not have access to the external network, then please refer to section 10.1.1.

![Figure 2.4: License management in done under the Management tab.](image)

2.6.1 Download a static license on a non-networked machine

Follow the steps below to download a static license for a server machine that does not have direct access to the external network.
Figure 2.5: A warning is shown in the web administrative login page for a single server or master node when no license is present for the CLC Server (left) and when a license for the CLC Server is present but is not valid for the version being run (right).

Figure 2.6: The web administrative interface login page for a single server or master node when a valid license for the CLC Server is present.

- Determine the host ID of the machine the server software will be running on. This can be done by running the back-end license download tool, which prints the host ID of the system to the terminal. The download tool is located in the installation folder of the CLC Server. The tool name depends on the system you are working on:
  - Linux: downloadlicense
  - Mac: downloadlicense.command
  - Windows: licensedownload.bat

In the case of a job or grid node setup, the host ID should be for the machine that will act as the CLC Server master node.

- Make a copy of this host ID such that you can use it on a machine that has internet access.

- Go to a computer with internet access, open a browser window and go to the server license download web page:
  https://secure.clcbio.com/LmxWSv3/GetServerLicenseFile

  For licenses for server extensions, the license download page is:
  https://secure.clcbio.com/LmxWSv3/GetLicenseFile

- Paste in your license order ID and the host ID that you noted down earlier into the relevant boxes on the webpage.

- Click on ‘download license’ and save the resulting .lic file.

- On the machine with the host ID you specified when downloading the license file, place the license file in the folder called ‘licenses’ in the CLC Server installation directory.

- Restart the CLC software.
2.7 Starting and stopping the server

2.7.1 Microsoft Windows

On Windows based systems the CLC Server can be controlled through the Services control panel. The service is named CLC Genomics Server: CLCGenomicsServer

Choose the service and click the start, stop or restart link as shown in figure 2.7.

![Stopping and restarting the server on Windows by clicking the blue links.](image)

Figure 2.7: Stopping and restarting the server on Windows by clicking the blue links.

Once your CLC Server has started up, log in as an administrative user to configure and to manage it, as described in later chapters.

2.7.2 macOS

On macOS the server can be started and stopped from the command line.

Open a terminal and navigate to the CLC Server installation directory. Once there, the server can be controlled with the following commands.

Remember to replace CLCServer, in the commands listed below, with the name from the CLC Genomics Server: CLCGenomicsServer

To start the server run the command:

```
sudo ./CLCServer start
```

To stop the server run the command:

```
sudo ./CLCServer stop
```

To view the current status of the server run the command:

```
sudo ./CLCServer status
```

You will need to set this up as a service if you wish it to be run that way. Please refer to your operating system documentation if you are not sure how to do this.

Once your CLC Server has started up, log in as an administrative user to configure and to manage it, as described in later chapters.
2.7.3 Linux

You can start and stop the CLC Server service from the command line. You can also configure the service to start up automatically after the server machine is rebooted.

During installation of the CLC Server a service script is placed in /etc/init.d/.

This script will have a name reflecting the server solution, and it includes the name of the custom user account specified during installation for running the CLC Server process.

Starting and stopping the service using the command line:

To start the CLC Server:

```
sudo service CLCGenomicsServer start
```

To stop the CLC Server:

```
sudo service CLCGenomicsServer stop
```

To restart the CLC Server:

```
sudo service CLCGenomicsServer restart
```

To view the status of the CLC Server:

```
sudo service CLCGenomicsServer status
```

Start service on boot up:

On Red Hat Enterprise Linux and SuSE this can be done using the command:

```
sudo chkconfig CLCGenomicsServer on
```

How to configure a service to automatically start on reboot depends on the specific Linux distribution. Please refer to your system documentation for further details.

Troubleshooting

If the CLC Server is run as a service as suggested above, then the files in the installation area of the software and the data files created after installation in CLC Server File Locations will be owned by the user specified to run the CLC Server process. If someone starts up the CLC Server process as root (i.e. an account with super-user privileges) then the following steps are recommended to rectify the situation:

1. Stop the CLC Server process using the script located within the installation area of the CLC Server software. You can do that using the full path to this script, or by navigating to the installation area and running:

```
sudo ./CLCGenomicsServer stop
```

2. Change ownership recursively on all files in the installation area of the software and on all areas specified as Server File Locations.

3. Start the CLC Server service as the specified user by using the service script:

```
sudo service CLCGenomicsServer start
```

4. In case the server still fails to start correctly it can be started in the foreground with output being written to the console to help identify the problem. It is done by running:
Once your CLC Server has started up, log in as an administrative user to configure and to manage it, as described in later chapters.
Chapter 3

Basic configuration

3.1 Logging into the administrative interface

The administrative interface for a running CLC Server is accessed via a web browser. Most configuration occurs via this interface. Simply type the host name of the server machine you have installed the CLC Server software on, followed by the port it is listening on. Unless you change it, the port number is 7777. An example would be http://clccomputer:7777/ or http://localhost:7777/

The default administrative user credentials are:

- **User name**: root
- **Password**: default

Use these details the first time you log in. We recommend that you change this password. Details of how to change the administrative user password is covered in section 4.1.

3.2 Adding locations for saving data

Before using the server for analysis, the areas for storing data imported into or created by the software need to be specified.

On most server setups, this involves configuring a file system location, as described in section 3.2.1.

3.2.1 File system locations

Data storage configuration is done via the administrative web interface. When logged in as a user with administrative privileges, navigate to the Configuration tab, click on the Main configuration tab, and then click on the File system locations heading to expand that section. See figure 3.1.

File locations already configured will be listed. Those with a P in a blue box have permissions enabled.
Figure 3.1: Three file system locations are configured. Two have permissions settings enabled, as indicated by the Ps in blue boxes. One location, CLCServerLoc1, is disabled, as indicated by its name being in light grey text.

File system locations should appear in the list at the left hand side of the web client after they are added, if the location is enabled.

Adding and configuring file system locations

Add a new file system location  Click on the Add New File Location button and then specify the path to the folder where data imported into or created by the CLC Server will be stored. The path provided should point to an existing folder on the server machine that the user running the server process has read and write access to.

If a file system location with the name CLC_References is configured, users logged into a CLC Server from a CLC Genomics Workbench will be able to download data directly to this server area using the Workbench’s Reference Data Manager tool. Special conditions apply to this file system location. These are outlined in section 3.2.2.

Enable or disable access for all users  The checkbox to the left the Path is used to control whether or not this location should be available to users. Access is enabled by default. Unchecking this box and saving the configuration makes the location unavailable for use via any client software. For example, in a CLC Workbench connected to the CLC Server, each enabled location is visible in the Navigation Area, but disabled locations are not.

Disabled locations are listed in light grey text, as shown for CLCServerLoc1 in figure 3.2. As expected for a disabled location, it is not listed in the top left side of the web client, whereas the two enabled locations are listed.

Figure 3.2: When the checkbox to the left of the Path is unchecked, the file system location is disabled. This makes it unavailable to users. Under the Main Configuration tab, disabled locations are listed using light grey text.
Permissions enabled When this box is checked, permissions can be configured for that location (figure 3.3).

Changes to permissions settings first take effect after the server is restarted.

Note: Permissions can be configured for any file system location unless it has the name CLC_References (section 3.2.2).

Group-based permissions After selecting this option, group-level read and write permissions can be configured for the file system location using a CLC Workbench client, as described in section 5.1. Until such access is granted, the file system location and its contents are available only to admin users.

User homes The file system location will be used for user home folders. These are top-level folders with names matching users’ usernames. A user is granted write access only to the folder with a name matching their username. Admin users continue to have access to all folders.

- Auto-create user homes When selected, a user folder is created automatically by the CLC Server when a user first logs in. To create folders only for specific users, deselect this option and use the mkdir command of the CLC Server Command Line Tools, specifying the file system location as the target and providing a user’s username as the name of the folder to create.

- Allow global read access to user homes When selected, all users are granted read access to all user home folders on this file system location. When not selected, users only have access to their own user home area.

When enabling the "User homes" option, please note that:

- Data elements stored directly under the file system location will be readable, but not writable, by all users. While it should only be possible for admin users to place data in this area, if the file system location was in use before the User homes option was selected, then existing data stored at the top level will be accessible to all users.

- Any folder at the top level of a CLC Server file system location with a name matching a user’s username will be treated as a user home area, with read and write access granted to that user on that folder and its contents.

Remove a file system location Clicking on the Remove Location button beside a particular file system location removes it from the CLC Server. The underlying folder and its contents are
not deleted. To re-enable access via the CLC Server, simply configure the same folder as a file system location again.

Rebuild the index The CLC Server maintains an index of all the elements in a data location. This is used when searching for data. There is no need to re-index when adding a new area as a file system location. Rebuilding the index is described in more detail in section 16.2.1.

Important points about CLC Server file system locations

- Folders added as file system locations should be dedicated for use by the CLC Server and should be directly accessed only by the CLC Server.

- The underlying file system must support file locking.

- All the data written to file system locations by the CLC Server will be in clc format and will be owned by the user that runs the CLC Server process.

- Files should not be moved manually into folders designated as CLC Server file system locations, or their subfolders, for example using standard operating system’s command tools, drag and drop, and so on.

- Areas designated as file system locations should not overlap. That is, if a folder has been designated as a file system location, it should not be a subfolder of another area also designated as a file system location, or vice versa. Overlapping file system locations lead to problems with data indexing, which in turn leads to problems finding the stored data. Further information about indexing can be found in section 16.2.1.

File locations for job node set-ups

When you have a job node set-up, all the job node computers need to have access to the same data location folder. This is because the job nodes will write files directly to the folder rather than passing through the master node (which would be a bottleneck for big jobs). Furthermore, the user running the server must be the same for all the job nodes and it needs to act as the same user when accessing the folder no matter whether it is a job node or a master node.

The data location should be added after the job nodes have been configured and attached to the master node. In this way, all the job nodes will inherit the configurations made on the master node.

One relatively common problem faced in this regard is root squashing which often needs to be disabled, because it prevents the servers from writing and accessing the files as the same user - read more about this at http://nfs.sourceforge.net/#faq_b11.

You can read more about job node setups in section 6.

3.2.2 Reference data management

When a file system location called CLC_References is configured, CLC Genomics Workbench users logged into the CLC Server can use the Workbench’s Reference Data Manager to download reference data resources from QIAGEN and some public repositories and store them on the CLC Server. That data can then be used when running server-based analyses.
The **CLC_References** folder has the same requirements as any other area configured as a **CLC Server** file system location (see section 3.2.1).

The following special conditions apply to this area, once established:

- **Reading** All data stored in **CLC_References** can be seen by all users logged into the **CLC Server**.

- **Writing** Data can be downloaded to this area by any user using the **CLC Genomics Workbench Reference Data Manager** configured to refer to the server for reference data. This location is otherwise read-only. Data cannot be placed in this area by any user, including admin users, using standard mechanisms such as copy/paste, etc.

- **Deletion**
  - Data in **CLC_References** can be deleted by admin users via the **CLC Genomics Workbench Reference Data Manager** configured to refer to the server for reference data. No other users can delete data from this area.
  - Data cannot be deleted from this area using standard mechanisms in the **CLC Genomics Workbench**, such as deleting elements in the Navigation Area, etc.
  - Data in this area can be deleted via the Element Info tab of the **CLC Server** web administrative interface.

- **Permissions** Custom permissions cannot be set on a **CLC_References** file system location. The checkbox enabling permissions should not be selected. If it is, only administrative users will be able to read or write to this area using the **CLC Genomics Workbench Reference Data Manager**.

Further information about the Reference Data Manager, including how to configure it to refer to the **CLC_References** server file system location, is in the **CLC Genomics Workbench** manual at:


### 3.2.3 Enabling and disabling internal compression of CLC data

CLC format data is stored in an internally compressed format. An option is provided under the **Data compression** heading to turn off internal data compression. This setting applies to all configured file system locations.

Enabling data compression may impose a performance penalty depending on the characteristics of the hardware used. However, this penalty is typically small, and we generally recommend that this option remains enabled. Turning this option off is likely to be of interest only at sites running a mix of older and newer CLC software, where the same data is accessed by different versions of the software.

Compatibility information:

- A new compression method was introduced with version 22.0 of the **CLC Genomics Workbench**, **CLC Main Workbench** and **CLC Genomics Server**. Compressed data created using those versions can be read by version 21.0.5 and above, but not earlier versions.
• Internal compression of CLC data was introduced in CLC Genomics Workbench 12.0, CLC Main Workbench 8.1 and CLC Genomics Server 11.0. Compressed data created using these versions is not compatible with older versions of the software. Data created using these versions can be opened by later versions of the software, including versions 22.0 and above.

To share specific data sets for use with software versions that do not support the compression applied by default, we recommend exporting the data to CLC or zip format and turning on the export option "Maximize compatibility with older CLC products". That is described further at https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Export_folders_data_elements_in_CLC_format.html

3.3 Changing the listening port

The default listening port for the CLC Server is 7777. This has been chosen to minimize the risk of collisions with existing web-servers using the more familiar ports 80 and 8080. To have the server listen on a different port:

• Navigate to the CLC Server installation directory.
• Locate the file called server.xml in the conf directory.
• Open the file in a text editor and locate the following section

  <Connector port="7777" protocol="HTTP/1.1"
  connectionTimeout="600000" compression="off" />

• Change the port value to desired listening port (8080 in the example below)

  <Connector port="8080" protocol="HTTP/1.1"
  connectionTimeout="600000" compression="off" />

• Restart the service for the changes to take effect. See section 2.7).

If you have a job node setup or a grid node setup, then, after restarting, log into the web administrative interface and update the value in the "Master node port" field under Configuration | Job distribution | Server setup. Then click on Save Configuration button to save the new setting. Information about configuring SSL for the CLC Server is provided in section 3.3.1.

3.3.1 SSL and encryption

The CLC Server supports SSL communication between itself and its clients. This is particularly relevant for setups where the CLC Server is accessible over the internet as well as on a local network.

The default configuration of the server does not use SSL.

Logging in to an SSL enabled CLC Server from a CLC Workbench is described in section 3.3.1. Logging in using the CLC Server Command Line Tools is described in section 3.3.1.
Enabling SSL on the server

A server certificate is required before SSL can be enabled on the CLC Server. This is usually obtained from a Certificate Authority (CA) like Thawte or Verisign (see http://en.wikipedia.org/wiki/CertificateAuthorities).

A signed certificate in a pkcs12 keystore file is also needed. The keystore file is either provided by the CA or it can be generated from the private key used to request the certificate and the signed-certificate file from the CA (see section 3.3.1).

Copy the keystore file to the conf subdirectory of the CLC Server installation folder.

Next, the server.xml file in the conf subdirectory of the CLC Server installation folder has to be edited to enable SSL-connections. Add text like the following text to the server.xml file:

```xml
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true"
    clientAuth="false" sslProtocol="TLS"
    keystoreFile="conf/keystore.pkcs12" keystorePass="tomcat"
    keystoreType="PKCS12"/>
```

Replace keystore.pkcs12 with the name of your keystore file, and replace tomcat with the password for your keystore.

The above settings make SSL available on port 8443. The standard (non-SSL) port would still be 7777, or whatever port number you have configured it to.

Self-signed certificates can be generated if only connection encryption is needed. See http://www.akadia.com/services/ssh_test_certificate.html for further details.

Creating a PKCS12 keystore file

If the certificate is not supplied in a pkcs12 keystore file, it can be put into one by combining the private key and the signed certificate obtained from the CA by using openssl:

```bash
openssl pkcs12 -export -out keystore.pkcs12 -inkey private.key -in certificate.crt -name "tomcat"
```

This will take the private key from the file private.key and the signed certificate from certificate.crt and generate a pkcs12-store in the keystore.pkcs12 file.

Logging in using SSL from a CLC Workbench

When a CLC Workbench connects to the CLC Server, it automatically detects if Secure Socket Layer (SSL) should be used on the port it is connecting to or not.

If SSL is detected, the server’s certificate will be verified and a warning is displayed if the certificate is not signed by a recognized Certificate Authority (CA) as shown in figure 3.4.

When such an "unknown" certificate has been accepted once, the warning will not appear again. It is necessary to log in again once the certificate has been accepted.
When logged into a server, information about the connection can be viewed by hovering the connection icon on the status-panel as shown in figure 3.5.

The icon is gray when the user is not logged in, and a pad lock is overlayed when the connection is encrypted via SSL.

**Logging in using SSL from the CLC Server Command Line Tools**

The **CLC Server Command Line Tools** automatically detects and uses SSL if present on the port it connects to. If the certificate is untrusted the `clcserver` program will refuse to login:

```
./clcserver -S localhost -U root -W default -P 8443
Message: Trying to log into server
Error: SSL Handshake failed. Check certificate.
```

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-A &lt;Command&gt;</td>
<td>Command to run. If not specified the list of commands on the server will be returned.</td>
</tr>
<tr>
<td>-C &lt;Integer&gt;</td>
<td>Specify column width of help output.</td>
</tr>
<tr>
<td>-D &lt;Boolean&gt;</td>
<td>Enable debug mode (default: false)</td>
</tr>
<tr>
<td>-G &lt;Grid Preset value&gt;</td>
<td>Specify to execute on grid.</td>
</tr>
<tr>
<td>-H</td>
<td>Display general help.</td>
</tr>
<tr>
<td>-I &lt;Algorithm Command&gt;</td>
<td>Get information about an algorithm</td>
</tr>
<tr>
<td>-O &lt;File&gt;</td>
<td>Output file.</td>
</tr>
<tr>
<td>-P &lt;Integer&gt;</td>
<td>Server port number. (default: 7777)</td>
</tr>
<tr>
<td>-Q &lt;Boolean&gt;</td>
<td>Quiet mode. No progress output. (default: false)</td>
</tr>
<tr>
<td>-S &lt;String&gt;</td>
<td>Server hostname or IP-address of the CLC Server.</td>
</tr>
<tr>
<td>-U &lt;String&gt;</td>
<td>Valid username for logging on to the CLC Server.</td>
</tr>
<tr>
<td>-V</td>
<td>Display version.</td>
</tr>
<tr>
<td>-W &lt;String&gt;</td>
<td>Clear text password or domain specific password token.</td>
</tr>
</tbody>
</table>
In order to trust the certificate the clcsserversslstore tool must be used:

```bash
classserversslstore -S localhost -U root -W default -P 8443
```

The server (localhost) presented an untrusted certificate with the following attributes:

**SUBJECT**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name</td>
<td>localhost</td>
</tr>
<tr>
<td>Alternative Names</td>
<td>N/A</td>
</tr>
<tr>
<td>Organizational Unit</td>
<td>Enterprise</td>
</tr>
<tr>
<td>Organization</td>
<td>CLC Bio</td>
</tr>
<tr>
<td>Locality</td>
<td>Aarhus N.</td>
</tr>
<tr>
<td>State</td>
<td>N/A</td>
</tr>
<tr>
<td>Country</td>
<td>DK</td>
</tr>
</tbody>
</table>

**ISSUER**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Common Name</td>
<td>localhost</td>
</tr>
<tr>
<td>Organizational Unit</td>
<td>Enterprise</td>
</tr>
<tr>
<td>Organization</td>
<td>CLC Bio</td>
</tr>
<tr>
<td>Locality</td>
<td>Aarhus N.</td>
</tr>
<tr>
<td>State</td>
<td>N/A</td>
</tr>
<tr>
<td>Country</td>
<td>DK</td>
</tr>
</tbody>
</table>

**FINGERPRINTS**

<table>
<thead>
<tr>
<th>Type</th>
<th>Fingerprint</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHA-1</td>
<td>A5 F6 8D C4 F6 F3 CB 44 D0 BA 83 E9 36 14 AE 9B 68 9B 9C F9</td>
</tr>
<tr>
<td>SHA-256</td>
<td>4B B5 0B 04 3C 3A A1 E2 D1 BF 87 10 F1 5D EA ED 9B 92 FF E3 C1 C9 9A 35 4B AF F6 98 87 9F 1D A8</td>
</tr>
</tbody>
</table>

**VALIDITY PERIOD**

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Valid From</td>
<td>Sep 1, 2011</td>
</tr>
<tr>
<td>Valid To</td>
<td>Aug 31, 2012</td>
</tr>
</tbody>
</table>

Trust this certificate? [yn]

Once the certificate has been accepted, the clicserver program is allowed to connect to the CLC Server.

### 3.4 Changing the tmp directory

The **CLC Server** often uses a lot of disk space for temporary files. These are files needed during an analysis, and they are deleted when no longer needed. By default, these temporary files are written to your system default temporary directory. Due to the amount of space that can be required for temporary files, it can be useful to specify an alternative, larger, disk area where temporary files created by the CLC Server can be written.

In the **server installation directory** you will find a file called CLCServer.vmoptions, where CLCServer will be the name of your particular CLC server: CLCGenomicsServer

Open the file in a text editor and add a new line like this: `-Djava.io.tmpdir=/path/to/tmp` with the path to the new tmp directory. Restart the server for the change to take effect (see how to restart the server in section 2.7).

We highly recommend that the tmp area is set to a file system local to the server machine. Having tmp set to a file system on a network mounted drive can substantially affect the speed of performance.

#### 3.4.1 Job node setup

The advice about having a tmp area being set on a local file system is true also for job nodes. Here, the tmp areas for nodes should **not** point to a shared folder. Rather, each node should have a tmp area with an identical name and path, but situated on a drive local to each node.

You will need to edit the CLCServer.vmoptions file on each job node, as well as the master node, as described above. This setting is **not** pushed out from the master to the job nodes.
3.5 Setting the amount of memory available for the JVM

When running the CLC Server, the Java Virtual Machine (JVM) needs to know how much memory it can use. This depends on the amount of physical memory (RAM) and can thus be different from computer to computer. Therefore, the installer investigates the amount of RAM during installation and sets the amount of memory that the JVM can use.

On Windows and Linux, this value is stored in a property file called ServerType.vmoptions (e.g. CLCGenomicsServer.vmoptions) which contains a text like this:

```
-Xmx8192m
```

The number (8192) is the amount of memory in megabytes the CLC Server is allowed to use. This file is located in the installation folder of the CLC Server software.

By default, the value is set to 50% of the available RAM on the system you have installed the software on.

You can manually change the number contained in the relevant line of the vmoptions file for your CLC Server if you wish to raise or lower the amount of RAM allocated to the Java Virtual Machine.

3.6 Limiting the number of cpus available for use

A number of the algorithms in the CLC Server will, in the case of large jobs, use all the cpu available on your system to make the analysis as fast as possible. The maximum number of cpu that can be used can be configured:

- **Master nodes** Configured via the web administrative interface as described in section 6.4\(^1\).
- **Job nodes** Configured via the web administrative interface as described in section 6.2.
- **Grid nodes** Controlled by the grid scheduler. Further information is available in section 6.3.8.

3.7 HTTP settings

Under the Configuration tab, open the Main configuration tab and click on the heading "HTTP settings". Here you can set the time out for the user HTTP session and the maximum upload size (when uploading files through the web interface).

3.8 Deployment of server information to CLC Workbenches

See the Deployment manual at https://digitalinsights.qiagen.com/technical-support/manuals/ for information on pre-configuring the server log-in information when Workbench users log in for the first time.

\(^1\)The method of using a cpu.properties file to limit CPU usage on master nodes and job nodes is deprecated.
Chapter 4

Managing users and groups

4.1 Logging in the first time and changing the root password

When the CLC Server is running, you can log into the web administrative interface using the CLC Server root user default credentials:

- **User name:** root
- **Password:** default

For security reasons, you should change this password, as shown in figure 4.1.

Before users start submitting jobs to the CLC Server, you should at a minimum set up user authentication (section 4.2) and data locations (section 3.2).

**Note:** On a job node setup, it can be convenient to set up the job nodes before changing the authentication mechanism and root password (see section 6).

![Figure 4.1: Change the root password via web administrative interface.](image)

4.2 User authentication using the web interface

After the CLC Server has been installed and is running, log into the CLC Server root account. If you have not already changed the password for the root user, see section 4.1.
Then specify the user authentication mechanism to use by clicking on the Authentication mechanism heading under:

Configuration ( ) | Authentication ( )

The modes of authentication available are shown in figure 4.2.

If LDAP or Active Directory is selected, a settings panel is revealed, where the details of the integration are entered.

Members of a group specified as an administrative group with login rights to the CLC Server are able to configure and manage the CLC Server. For the built-in authentication method, this means adding particular users to the built-in admin group. For Active Directory or LDAP, this means designating a group in the box labeled Admin group name and adding any users who should be administrators of the CLC Server to this group.

As well as specifying locations used to store data, members of the administration group will be able to specify which areas should have permissions enabled, and to set specific permissions on particular folders, as described in section 5.

Figure 4.2: Three modes of user authentication are available. Clicking on the Edit Permissions link at the bottom opens up the Permissions tab, where access to the CLC Server and its functionality can be configured, as described in section 5.2.

4.2.1 Authentication options

Built-in authentication

Using built-in authentication, you create users, set passwords, assign users to groups and manage groups using the CLC Server web administrative interface (see section 4.2.2) or using a CLC Workbench (see section 4.3). All the user information is stored on the CLC Server and is not accessible from other systems.

LDAP directory

Using the LDAP directory option, information needed during authentication and group memberships is retrieved from the specified LDAP directory (figure 4.3).

Configuration option information

Encryption The default is Plain text, with options available for encryption using Start TLS ("Forced Start TLS") or LDAP over SSL ("ldaps://").
**Admin group name** Provide the name of the admin group. This setting is case sensitive.

**DN to use for lookups** This allows you to choose which bind should be used for read and search operations. If no bind DN have been entered an unauthenticated bind will be used to do the initial lookup (lookup of users DN based on the username), and all other read and search operations will be performed with users binds. If the **Bind DN** and **Bind password** have been filled in, you have the choice between using the 'Bind DN' or the 'User DN' for read and search operations, the 'Bind DN' will in this case always be used for the initial lookup.

**User object class and Group object class** Intended for use where the standard posixAccount and posixGroup classes are not appropriate.

**Kerberos/GSSAPI Authentication** Enable the LDAP integration to use Kerberos/GSSAPI.

**Certificates** If your LDAP server uses a certificate that is not generally trusted by the server system that the CLC Server software is running on, then it must be added to the truststore of the CLC Server installation (CLC_SERVER_BASE/jre/lib/security/cacerts, where CLC_SERVER_BASE is the server installations root location). This can be done with the keytool shipped with Java installations (also available in the CLC_SERVER_BASE/jre/bin/keytool), with a command like:

```shell
CLC_SERVER_BASE/jre/bin/keytool -import -alias \ ldap_certificate -file LDAP_CERTIFICATE.cer -keystore \ CLC_SERVER_BASE/jre/lib/security/cacerts -storepass changeit
```
Replace **LDAP_CERTIFICATE** with the path to the certificate your LDAP server uses for Start TLS/LDAPS connections. Replace **CLC_SERVER_BASE** with the path to the server's installation location. For a node setup, this must be done for all job nodes as well.

**Caution:** If you update the server installation or reinstall the server, all imported certificates will be removed, and have to be imported again. You should also be aware that certificates have an expiration date, and will not be trusted after this date. Make sure to add a new certificate in advance of the expiration date.

**Active Directory**

Using the Active Directory option, information needed during authentication and group memberships is retrieved from the specified Active Directory. Encryption options (Start TLS and LDAP over SSL) are available (figure 4.4).

![Active Directory settings panel](image)

**Hostname** We recommend entering a Global Catalog in the hostname field. This avoids the CLC Server being redirected to several different Domain Controllers to obtain information about users and groups, and can thereby speed up the response time considerably in complex network environments. When a Global Catalog is specified, the port number must be configured to either

- **3268** LDAP, plain/startTLS, comparable to port 389, or
- **3269** LDAPS, SSL, comparable to 636

Please see the notes in the LDAP section (see section 4.2.1) for other recommendations and configuration details.
4.2.2 Managing users and groups using built-in authentication

This information only applies if built-in authentication is being used.

Managing users via the web administrative interface

To create or remove users, or change a password when using built-in authentication, expand the Manage user accounts heading under:

Configuration ( ) Users and groups ( )

This will display the panel shown in figure 4.5.

Managing groups via the web administrative interface

To create or remove groups or change group membership for users when using built-in authentication, expand the Manage groups heading under:

Configuration ( ) Users and groups ( )

This will display the panel shown in figure 4.6.

The same user can be a member of several groups.

Users who should have access to the administrative part of the server should be part of the "admin" group which is the only group with special permissions by default. The admin group already exists in a default setup of the CLC Server.

You will always be able to log in as the CLC Server root user, and this user has administrative level access rights.
4.3 User authentication via the Workbench for built-in authentication

This information only applies if built-in authentication is being used. If LDAP or AD is being used, the menus described here will be disabled.

Users and groups can be managed through the Workbench by logging into the CLC Server as an administrative user and then going to the Workbench menu:

File | Manage Server Users and Groups

This will display the dialog shown in figure 4.7.

Click on the Add (➕) button to create a new user. Enter the name of the user and enter a password. You will be asked to re-type the password. If you wish to change the password at a later time, select the user in the list and click Change password (🛠).

To delete a user, select the user in the list and click Delete (➖).

Access rights are granted to groups, not users, so a user has to be a member of one or more
groups to get access to the data location.

Adding and removing groups is done in the Groups tab (see figure 4.8).

![Figure 4.8: Managing server groups via a CLC Workbench](image)

To create a new group, click the Add (>Create) button and enter the name of the group. To delete a group, select the group in the list and click the Delete (Remove) button.

When a new group is created, it is empty. To assign users to a group, click on the Membership tab. In the Selected group box, you can choose among all the groups that have been created. When you select a group, you will see its members in the list below (see figure 4.9). To the left you see a list of all users.

![Figure 4.9: Listing members of a group.](image)

To add users to or to remove users from a group, click on the Add (Create) or Remove (Remove) buttons. To create new users, see section 4.3.
Chapter 5

Access privileges and permissions

Server administrators can restrict access and actions available to members of specified groups at various levels, ranging from access to the CLC Server itself, to fine grained control over the types of jobs they can launch, and where they are allowed to run those jobs.

Limiting access to CLC Server data in a given Server File Location is described in section 5.1. Access to the CLC Server itself, and defining access to analyses and external data, is described in section 5.2.

5.1 Controlling group access to CLC Server data

Folders are the basic unit for controlling access to data. Permissions applied to a folder also apply to the data elements within it. This section describes how to control access to folders on server file locations where permissions have been enabled and the “Group-based permissions” option has been selected, as described in section 3.2.1. For information about Server File Locations with the "User homes" permission setting enabled, see section 3.2.1.

When group-based permissions are enabled on a file system location, initially only the root user or users in a configured admin group have access to data stored there. Permissions can then be granted on folders to specified groups using the web administrative interface or using a CLC Workbench acting as a client for the CLC Server, as described in section 5.1.1.

Members of groups can be granted two types of access:

Read access Elements in these folders can be listed, opened or copied using the CLC Server Command Line Tools, the web client, or a CLC Workbench, for example by browsing in the Navigation Area, searching, or clicking the "Originates from" link in the History (دراس) of a data element.

Write access New elements and subfolders can be created in this area, and changes made to existing data or folders can be saved.

Note: To access a folder and its contents, a user must be a member of a group with read access to all the folders above it in the hierarchy. In the example shown in figure 5.1, to access the Sequences folder, the group must have access to both the Example Data and Protein folders.
It is fine to give write access to just the final folder in a hierarchy. For example, in the hierarchy shown in figure 5.1, read access could be granted to the *Example Data* and *Protein* folders, with read and write access granted to just the *Sequences* folder.

Please see section 5.1.2 for further details about the system behavior relating to permissions.

### 5.1.1 Setting permissions on folders

Group-based permissions can be set on folders using the web administrative interface or using a *CLC Workbench* acting as a client for the *CLC Server*.

**Setting group permissions on data folders using the web interface**

When logged into the web administrative interface, go to the *Element info* tab, and click on the *Folder permissions* tab. Select a folder in the Navigation Area on the left-hand side. A view like that in figure 5.2 will appear if the folder is in a location with group-level permissions enabled. Top-level locations can also be selected.

Read and/or write permission for individual groups can be set by checking the relevant boxes. Checking the “Apply to all subfolders” option will apply the same settings to all subfolders and the elements in those subfolders. This operation should be applied with caution as it will overwrite any existing permission settings in subfolders.

**Setting group permissions on data folders using a CLC Workbench**

If you are not already logged into the *CLC Server* from the *CLC Workbench* as an administrative user, log in by selecting the menu option:

**Connections | CLC Server Connection **

You can then set permissions on folders within File Locations that have had group-based permissions enabled.

right-click the folder | Permissions

---

1 In *CLC Server* 21.x and earlier, setting group permissions could only be done using a *CLC Workbench*.
CHAPTER 5. ACCESS PRIVILEGS AND PERMISSIONS

Figure 5.2: Group permissions being applied to an entire file system location.

This will open the dialog shown in figure 5.3.

Figure 5.3: Setting permissions on a folder.

Set the relevant permissions for each of the groups and click OK.

If you wish to apply the permissions to all subfolders recursively, check Apply to all subfolders in the dialog shown in figure 5.3. This operation should be applied with caution as it will overwrite any existing permission settings in subfolders.

5.1.2 Technical notes about permissions and security

All data stored in CLC Server file system locations are owned by the user that runs the CLC Server process. Group-based permissions and "user homes" permissions on file system locations are additional layers within the CLC Server, and are not part of your operating system’s permission system.

This means:

- Enabling and configuring permissions on server locations only affects users accessing data through CLC software. If users have direct access to the data, using for example general system tools, permissions set on the data via the CLC Server has no effect. Changing the ownership of the files using standard system tools is not recommended and will usually lead to serious problems with data indexing and hamper your work on the CLC Server.

- Without permissions enabled on a file system location, all users logging into the CLC Server are able to access all data within that file system location, and write data to that file system
locations. All files created within such a file system location are then also accessible to all users of the CLC Server.

If the "User homes" option is selected any data already present in the top level folder of such a file system location is readable by any user. In addition, if this area previously had group permissions applied to it, those permissions will no longer be active.

### 5.2 Controlling access to the server, server tasks and external data

Access to the various aspects of the CLC Server can be controlled using settings under:

- **Configuration**
- **Global Permissions**

See figure 5.4.

![Configuration and Global Permissions](image)

Figure 5.4: Access can be controlled for many aspects of the CLC Server, including restricting access to certain tools and granting administrative access for certain types of actions.

Group-level access can be configured for the following:

- **Login restrictions** Restrict who can log into the CLC Server.
• **Web admin access** Give admin-level access to defined areas in the web administrative interface. This is described further below.

• **Algorithms** Restrict access to specific tools to specified groups (figure 5.6). Enter a term in the search field to list tools with names containing that term.

• **Workflows** Restrict access to workflows installed on the CLC Server. To grant administrative access to such workflows, set the permissions under the "Web admin access" section.

• **External applications** Restrict access to External Applications that have been configured and made available on the CLC Server. To grant administrative access to configure and install external applications, set the permissions under the "Web admin access" section.

• **Import/export directories** Restrict access to file system areas not part of the CLC data setup that the CLC Server is able to access. These are described in section 7.1.

• **AWS Connections** Restrict access to AWS Connections configured under the External Data tab.

• **Grid presets** For grid node setups only: Restrict access to grid presets, which are used for sending jobs to a particular queue with particular parameters. Note that grid presets are identified by name. If you change the name of a preset under the Job Distribution settings section, then this, in effect, creates a new preset. In this situation, if you had access permissions previously set, you would need to reconfigure those settings for this, now new, preset.

• **Cloud presets** Only relevant when the Cloud Server Plugin is installed and an AWS Connection has been configured for accessing an AWS account with CLC Genomics Cloud resources in place. Access to each cloud preset can be customized.

To edit permissions, click on the relevant heading and then click on the relevant **Edit Permissions** button in the expanded list (figure 5.6). In the edit dialog, if the option **Only authorized users from selected groups**, is selected, a list of groups is shown, allowing access to be granted or removed, as relevant.

**Web admin access**

By default, only members of the admin group can access administrative areas of the web interface. Settings under the "Web admin access" heading allow access to be extended to members of other groups to the following areas:

• **Audit log** See and work with the audit log, available from under the Management tab. See chapter 12.

• **External applications** Configure and administer external applications. See chapter 14.

• **Queue** See a list of the processes running on, or queued to be run on, the CLC Server. See chapter 11.

• **Workflows** Install and administer workflows on the CLC Server. See chapter 8.
Figure 5.5: Setting group-level permissions for tools is done under the Algorithms section. By default, all authorized users have access to all tools. Here, access has been limited to members of the admin and bioinformaticians groups.

Figure 5.6: Access can be granted to members of non-admin groups to the Audit log, External applications, Queue, and Workflows tabs in the web administrative interface.

### 5.3 Customized attributes on data locations

Location-specific attributes can be set on all elements stored in a given data location. Attributes could be things like company-specific information such as LIMS id, freezer position etc. Attributes are set using a CLC Workbench acting as a client to the CLC Server.

Note that the attributes scheme belongs to a particular data location, so if there are multiple data locations, each will have its own set of attributes.
To configure which fields that should be available\(^2\) go to the Workbench:

right-click the data location | Location | Attribute Manager

This will display the dialog shown in figure 5.7.

![Adding attributes dialog](image)

Figure 5.7: Adding attributes.

Click the Add Attribute button to create a new attribute. This will display the dialog shown in figure 5.8.

![Create Attribute dialog](image)

Figure 5.8: The list of attribute types.

First, select what kind of attribute you wish to create. This affects the type of information that can be entered by the end users, and it also affects the way the data can be searched. The following types are available:

- **Checkbox.** This is used for attributes that are binary (e.g. true/false, checked/unchecked and yes/no).

- **Text.** For simple text with no constraints on what can be entered.

- **Hyper Link.** This can be used if the attribute is a reference to a web page. A value of this type will appear to the end user as a hyper link that can be clicked. Note that this attribute can only contain one hyper link. If you need more, you will have to create additional attributes.

- **List.** Lets you define a list of items that can be selected (explained in further detail below).

- **Number.** Any positive or negative integer.

\(^2\)If the data location is a server location, you need to be a server administrator to do this.
• **Bounded number.** Same as number, but you can define the minimum and maximum values that should be accepted. If you designate some kind of ID to your sequences, you can use the bounded number to define that it should be at least 1 and max 99999 if that is the range of your IDs.

• **Decimal number.** Same as number, but it will also accept decimal numbers.

• **Bounded decimal number.** Same as bounded number, but it will also accept decimal numbers.

When a data element is copied, attribute values are transferred to the copy of the element by default. To prevent the values for an attribute from being copied, uncheck the **Values are inheritable** checkbox.

When you click **OK**, the attribute will appear in the list to the left. Clicking the attribute will allow you to see information on its type in the panel to the right.

Lists are a little special, since you have to define the items in the list. When you choose to add the list attribute in the left side of the dialog, you can define the items of the list in the panel to the right by clicking **Add Item** (see figure 5.9).

![Figure 5.9: Defining items in a list.](image)

Remove items in the list by pressing **Remove Item**.

**Removing attributes** To remove an attribute, select the attribute in the list and click **Remove Attribute**. This can be done without any further implications if the attribute has just been created, but if you remove an attribute where values have already been given for elements in the data location, it will have implications for these elements: The values will not be removed, but they will become static, which means that they cannot be edited anymore.

If you accidentally removed an attribute and wish to restore it, this can be done by creating a new attribute of exactly the same name and type as the one you removed. All the "static" values will now become editable again.

When you remove an attribute, it will no longer be possible to search for it, even if there is "static" information on elements in the data location.
Renaming and changing the type of an attribute is not possible - you will have to create a new one.

**Changing the order of the attributes** You can change the order of the attributes by selecting an attribute and click the **Up** and **Down** arrows in the dialog. This will affect the way the attributes are presented for the user.

### 5.3.1 Filling in values

When a set of attributes has been created (as shown in figure 5.10), the end users can start filling in information.

![Figure 5.10: A set of attributes defined in the attribute manager.](image)

This is done in the element info view:

- **right-click a sequence or another element in the Navigation Area | Show ![Show](image) | Element info ![Element info]**

This will open a view similar to the one shown in figure 5.11.

You can now enter the appropriate information and **Save**. When you have saved the information, you will be able to search for it (see below).

Note that the element (e.g. sequence) needs to be saved in the data location before you can edit the attribute values.

When nobody has entered information, the attribute will have a "Not set" written in red next to the attribute (see figure 5.12).

This is particularly useful for attribute types like checkboxes and lists where you cannot tell, from the displayed value, if it has been set or not. Note that when an attribute has not been set, you cannot search for it, even if it looks like it has a value. In figure 5.12, you will **not** be able to find this sequence if you search for research projects with the value "Cancer project", because it has not been set. To set it, simply click in the list and you will see the red "Not set" disappear.

If you wish to reset the information that has been entered for an attribute, press "Clear" (written in blue next to the attribute). This will return it to the "Not set" state.

The **Folder editor**, invoked by pressing **Show** on a given folder from the context menu, provides
5.3.2 What happens when a clc object is copied to another data location?

The user supplied information, which has been entered in the Element info, is attached to the attributes that have been defined in this particular data location. If you copy the sequence to another data location or to a data location containing another attribute set, the information will become fixed, meaning that it is no longer editable and cannot be searched for. Note that attributes that were "Not set" will disappear when you copy data to another location.

If the element (e.g. sequence) is moved back to the original data location, the information will again be editable and searchable.

If the e.g. Molecule Project or Molecule Table is moved back to the original data location, the information will again be editable and searchable.

5.3.3 Searching custom attributes

When a text-based attribute is created, it becomes available for searching using Quick Search or Local Search.

**Quick Search:** Precede the name of the attribute with "attrib_" and then add a colon followed by the query term, e.g. "attrib_color:blue".

You must use whole words when searching for elements containing particular values for local attributes.
For searching purposes, **words** are the terms on either side of a space, hyphen or underscore in a name. The names of elements and folders are split into words when indexing.

In the **Local Search** tool, the local attributes are available to select from a drop down list, in addition to Name and Path.

Chapter 6

Job distribution

The CLC Server has the concept of distributing jobs to nodes. This means having a master server with the primary purpose of handling user access, serving data to users and starting jobs, and one or more nodes, which execute the jobs submitted to them. This chapter describes the server setups that are available for the CLC Server as well as some job running options available for single servers and those running CLC job nodes.

6.1 Introduction to servers setups

The three models for running the CLC Server are:

- **Model I: Master server with dedicated job nodes.** In this model, a master server submits CLC jobs directly to machines running the CLC Server for execution. In this setup, a group of machines (from two upwards) have the CLC Server software installed on them. The system administrator assigns one of them as the master node. The master controls the queue and distribution of jobs and compute resources. The other nodes are job nodes, which execute the computational tasks they are assigned. This model is simple to set up and maintain, with no other software required. However, it is not well suited to situations where the compute resources are shared with other systems because there is no mechanism for managing the load on the computer. This setup works best when the execute nodes are machines dedicated to running a CLC Server. Further details about this setup can be found in section 6.2

- **Model II: Master server submitting to grid nodes.** In this model, a master server submits tasks to a local third party scheduler. That scheduler controls the resources on a local computer cluster (grid) where the job will be executed. This means that it is the responsibility of the native grid job scheduling system to start the job. When the job is started on one of the grid nodes, a CLC Grid Worker, which is a stand-alone executable including all the algorithms on the server, is started with a set of parameters specified by the user. Further details about this setup can be found in section 6.3.

- **Model III: Single Server setup.** In this model, the master and execution node functionality is carried out by a single CLC Server instance.

Figure 6.1 shows a schematic overview.
For models I and II, the master server and job nodes, or master server and grid nodes must run on the same type of operating system. It is not possible to have a master server running Linux and a job node running Windows, for example.

![Figure 6.1: An overview of the job distribution possibilities.](image)

### 6.2 Model I: Master server with dedicated job nodes

#### 6.2.1 Overview: Model I

A CLC Server job node setup involves a master node and one or more job nodes. The master node manages incoming jobs, including placing tasks in the queue, while the job nodes execute the jobs. Each job node is expected to be dedicated to running the CLC Server software. A more detailed overview is given in section 6.1.

The general steps for setting up this model are:

1. Install the CLC Server software on all the machines involved. (See section 2.2.)
2. Install the license on the machine that will act as the master node. (See section 2.6.)
3. Start up the CLC Server software on the master server. Then start up the software on the job nodes. (See section 2.7.)
4. Log in to the web administrative interface for the CLC Server of the master node. (See section 3.1.)
5. Configure the master node, attach the job nodes and configure the job nodes via the administrative interface on the master node.

Almost all configuration for the CLC Server cluster is done via the CLC Server web administrative interface on the master node. This includes the installation of plugins. See section 6.2.4. Job nodes inherit their configurations from the master node.

The only work done directly on the machines that will run as job nodes is

- Installation of the CLC Server software.
- Starting up the software up on each node.
- The changing of the built-in administrative login credentials under certain circumstances. See section 6.2.2.
6.2.2 User credentials on a master-job node setup

When initially installed, all instances of the CLC Server will have the default admin user credentials. If you have a brand new installation, and you are happy to use the default administrative login credentials (see section 3.1) during initial setup, you do not need to change anything.

Once the CLC Server software on all machines is up and running and the job nodes have been attached to the master, changes to passwords for the built-in authentications system, which includes the default admin user, root, will be pushed from the master to the job nodes. You do not need to manually change the password on each job node.

If you wish to change the default administrative password before attaching the job nodes to the master, then please log into the web administrative interface of each machine running the CLC Server software and setup identical details on each one.

The master node needs access to the job nodes to be able to push configurations to them. Thus, if you change the admin password on the master server and later wish to attach a new job node, you will need to log into the web administrative interface of the job node and set the root password for the CLC Server software to match that of the master server. Until that is done, the master will not be able to communicate with the job node because the root admin passwords are different. Once the master can communicate with the job node, it can push other configurations to the job node.

6.2.3 Configuring your setup

If you have not already, please download and install your license to the master node. (See section 2.6.) Do not install license files on the job nodes. The licensing information, including how many job nodes you can run, are all included in the license on the master node.

To configure your master/execution node setup, go to the Job distribution tab in the web administrative interface on the master node:

Configuration | Job distribution

Then enter the following information:

- **Server mode** - select MASTER_NODE.
- **Master node host** - Enter the master node host name. Click on the "Show suggestions" text next to this field to see information about the server that can be useful when configuring this option.
- **Master node port** - usually 7777
- **Master node display name** - the name shown in the top bar of the web interface for the CLC Server
- **CPU limit** - The maximum number of CPU the CLC Server should use. This is set to unlimited by default, meaning that up to all cores of the system can be used.

Click on the button **Save Configuration** to register the changes just made.
If the **Attach Node** button in the Job nodes section is greyed out, please ensure that the server mode selected is **MASTER_NODE** and that you have clicked on the **Save Configuration** button to save your configuration changes.

Then, for each job node:

- Click on the **Attach Node** button to specify a job node to attach. See figure 6.2.

- Enter information about the node in the fields (see figure 6.3).

- Optionally, click on the button labeled "Manage Job Types" to specify the types of jobs that can be run on each node. See figure 6.4. The default is "Any installed Server Command". If you choose instead "Only selected Server Commands", a search field and a list of all server command names and types will appear. The search field can be used to narrow down the server commands by name or type. Only the tools selected can then be run on that particular job node. Click **Modify** when you are done.

- Click **Create**.

Repeat this process for each job node you wish to attach and click **Save Configuration** when you are done.
You will get a warning dialog if there are types of jobs that are not enabled on any of the nodes. Note that when a node has finished a job, it will take the first job in the queue that is of a type the node is configured to process. This then means that, depending on how you have configured your system, the job that is number one in the queue will not necessarily be processed first.

Once set up, job nodes automatically inherit all configurations on the master node.

To test that access works for both job nodes and the master node, you can click "check setup" in the upper right corner as described in section 18.1.

Troubleshooting job node setups

- **Disable root squashing** Root squashing often needs to be disabled because it prevents the servers from writing and accessing the files as the same user. Read more about this at http://nfs.sourceforge.net/#faq_b11.

- **Bringing job nodes back in sync with the master** It is expected the job nodes will stay in sync with the master. However, if one of the job nodes gets out of sync, click on the Resync job nodes button, which can be seen in figure 6.2.

Before resyncing, ensure no jobs are running on any nodes. We recommend using Maintenance Mode for this, as this allows current jobs to complete but stops further jobs from being submitted. Once all the running jobs have completed, maintenance tasks can be safely carried out. Maintenance Mode is described further in section 10.2.

Resyncing nodes will detach and then reattach all job nodes, and includes a full reinstallation of all installed plugins from the master as well as reapplication of all settings on the master to each job node. When resyncing is complete, restart your setup. This can be done using the Restart option under Server maintenance area, as described in section 10.2.

### 6.2.4 Installing Server plugins on job nodes

You only need to install or uninstall plugins on the master server. The CLC Server master and all job nodes need to be restarted to complete the installation or removal or plugins. This is easily
accomplished by using the Restart option in the Server Maintenance section under the Status and management area on the master server, which restarts the master and all the job nodes. See section 10.

Server plugin installation and removal is described in section 13.

Once a plugin is installed, you should check that the plugin is enabled for each job node you wish to make available for users to run that particular task on. To do this:

- Go to the Job Distribution tab in the master nodes web administrative interface
- Click on the link to each job node
- Click in the box by the relevant task, marking it with a check mark if you wish to enable it.

6.3 Model II: Master server submitting to grid nodes

6.3.1 Overview: Model II

A CLC Server with grid integration allows jobs to be offloaded from a master server to grid nodes using local grid submission/queuing software to handle job scheduling and submission. A more detailed overview is given in section 6.1.

A given CLC algorithm runs on a single grid node. Thus, each grid node employed for a given task must have enough memory and space to carry out that entire task.

The grid system uses two locations for its deployment:

- **Path to CLC Grid Worker** in the grid preset editor. This location is used for plugins, a grid version of the server, i.e., the code that is executed when a grid job is started, licences, libraries and more. Grid workers are redeployed in two situations: 1) When the server starts up and 2) If the configuration of one of the grid presets is changed, in which case the grid workers of all presets are redeployed. In addition, the grid workers are updated when a plugin is installed or removed.

- **Shared work directory**. This location is where each grid job gets its own sub directory in which it places its temporary data, e.g., the description of the job to be executed, the configurations files that the grid version of the server needs in order to setup persistence models and log files. This location is only deployed once, either when the grid job starts executing (in case of workflow jobs) or when the grid job is queued (in all other cases).

6.3.2 Requirements for CLC Grid Integration

- A functional grid submission system must already be in place. Please also see the section on supported job submission systems below.

- The DRMAA library for the grid submission system to be used. See Appendix section 20.3 for more information about DRMAA libraries for the supported grid submission systems.

- The CLC Server must be installed on a Linux based system configured as a submit host in the grid environment.
• The user running the CLC Server process is seen as the submitter of the grid job, and thus this user must exist on all the grid nodes.

• CLC Server file locations holding data that will be used must be mounted with the same path on the grid nodes as on the master CLC Server and accessible to the user that runs the CLC Server process.

• A CLC Network License Manager with one or more available CLC Grid Worker licenses must be reachable from the execution hosts in the grid setup.

**Supported grid scheduling systems**

Grid integration in CLC Genomics Server is done using DRMAA. We have verified grid integration using the following third party scheduling systems:

• SLURM 16.05.2 and 21.08.1
• UNIVA 8.4.1 and 8.6.1.7
• LSF 9.1.1 and 10.1
• PBS Pro 2020.1.4 and 2021.1.1

Notes about DRMAA for each of the grid scheduling systems are provided in the appendix 20.3, including information relating to compilation, where relevant.

Integration using other grid scheduling systems is anticipated to work as long there is a working DRMAA library and a mechanism to limit the number of CLC jobs launched for execution such that when this number exceeds the number of CLC Grid Worker licenses, excess tasks are held in the queue until a license becomes available.

For SLURM, the number of CLC Grid Worker licenses can be configured as described on https://slurm.schedmd.com/licenses.html. For LSF and UNIVA, a "Consumable Resource" would be configured, as described in section 6.3.6. Relevant information about configuring consumable resources for PBS Pro can be found in the adminstrator’s guide for that scheduling software.

TORQUE from Adaptive Computing is an example of a system that works for submitting CLC jobs, but that cannot be supported because it does not provide a means of limiting the number of CLC jobs. As far as we know, there is no way to limit the number of CLC jobs sent simultaneously to the cluster to match the number of CLC Grid Worker licenses. So, with TORQUE, if you had three Grid Worker licenses, up to three jobs could be run simultaneously. However, if three jobs are already running and you launch a fourth job, then this fourth job will fail because there would be no license available for it. This limitation can be overcome, allowing you to work with systems such as TORQUE, if you control the job submission in some other way so the license number is not exceeded. One possible setup for this is if you have a one-node-runs-one-job setup. You could then set up a queue where jobs are only sent to a certain number of nodes, where that number matches the number of CLC Grid Worker licenses you have.
6.3.3  Technical overview

Figure 6.5 shows an overview of the communication involved in running a job on the grid, using OGE as the example submission system.

![Figure 6.5: An overview of grid integration, using OGE as the example submission system.](image)

The steps of this figure are in detail:

1. From the Workbench the user invokes an algorithm to be run on the grid. This information is sent to the master server running the CLC Server.

2. The master server writes a file with job parameters to a shared work directory of the grid execution nodes. The job parameters contain identifiers mapping to the job data placed in the CLC server data location. The job parameters file is automatically deleted when it is no longer used by the grid node.

3. Now the server invokes `qsub` through the specified DRMAA native library. Then `qsub` transfers the job request to the grid scheduler. Since the user that runs the CLC Server process has invoked `qsub`, the grid node will run the job as this CLC-Server user.

4. The job scheduler will choose a grid node based on the parameters given to `qsub` and the user that invoked `qsub`.

5. The chosen grid node will retrieve CLC Grid Worker executable and the job parameters from the shared file system and start performing the given task.

6. After completion of the job, the grid node will write the results to the server’s data location. After this step the result can be accessed by the Workbench user through the master server.

6.3.4  Setting up the grid integration

CLC jobs are submitted to a local grid via a special, stand-alone executable called a `clcgridworker`. In the documentation, this executable is also referred to as the CLC Grid Worker.
The following steps are taken to setup grid integration for CLC jobs. These steps are described in more detail in the sections that follow. It is assumed that your CLC Server software is already installed on the machine that is to act as the master.

1. Set up the licensing of the grid workers, as described in section 6.3.5
2. Configure the CLC grid licenses as a consumable resource in the local grid system, as described in section 6.3.6
3. Configure and save grid presets, as described in section 6.3.7
4. Optionally, create and edit a clcgridworker.vmoptions file in each deployed CLC Grid Worker area, as described in section 6.3.10.
5. Test your setup by submitting some small tasks to the grid via a CLC Server client, such as a CLC Genomics Workbench or the CLC Server Command Line Tools. Ideally, these would be tasks already known to run smoothly directly on your CLC Server.

### 6.3.5 Licensing of grid workers

There are two main steps involved in setting up the licenses for your CLC Grid Workers.

**Step 1: Obtain your CLC Grid Worker network licenses and make them available for use**

CLC Grid Worker licenses are served by the CLC Network License Manager software. For information on installing and configuring that software, and on downloading and making your CLC Grid Worker licenses available for use, please refer to the CLC Network License Manager manual at:

https://resources.qiagenbioinformatics.com/manuals/clcnetworklicensemanager/current/

A pdf version is available at


**Step 2: Configure the location of the CLC Network License Manager for your CLC Grid Workers**

One license is used for each CLC Grid Worker script launched. When the CLC Grid Worker starts running on a node, it will attempt to get a license from the CLC Network License Manager, and when the job is complete, the license will be returned.

To provide the details needed for CLC Grid Workers to communicate with the CLC Network License Manager, use a text editor to open the file: gridres/settings/license.properties under the installation area of your CLC Server.

The file will look something like this:

```plaintext
#License Settings

serverip=host.example.com
serverport=6200
disableborrow=false
```
For grid use, we recommend that autodiscovery is turned off, i.e. `autodiscover=false` and that the host name or IP address of your CLC Network License Manager is specified using the `serverip` property.

After you have configured your grid presets, see section 6.3.7, and have saved them, those presets are deployed to the location you specify in the Path to CLC Grid Worker field of the preset. Along with the `clcgridworker` script, the license settings file is also deployed.

If you need to change your license settings, we recommend that you edit the license.properties file under `gridres/settings/license.properties` of your CLC Server installation, and then re-save each of your grid presets. This will re-deploy the CLC Grid Workers, including the changed license.properties file.

### 6.3.6 Configuring licenses as a consumable resource

Since there is a limitation on the number of licenses available, it is important that the local grid system is configured so that the number of CLC Grid Worker scripts launched is never higher than the maximum number of licenses installed. If the number of CLC Grid Worker scripts launched exceeds the number of licenses available, jobs unable to find a license will fail when they are executed.

Some grid systems support the concept of a consumable resource. Using this, you can set the number of CLC grid licenses available. This will restrict the number of CLC jobs launched to run on the grid at any one time to this number. Any job that is submitted while all licenses are already in use should sit in the queue until a license becomes available. **We highly recommend that CLC grid licenses are configured as a consumable resource on the local grid submission system.**

Information about how a consumable resource can be configured for LSF has been provided by IBM and can be found in Appendix section 20.4.

### 6.3.7 Configure grid presets

A grid preset contains information needed for jobs to be handled by the CLC Server and submitted to the grid scheduling system. Multiple grid presets can be configured. Users specify the relevant grid preset when submitting a job. See also section 6.3.12.

To configure a grid preset, log into the web interface of the CLC Server master and navigate to:

- Configuration | Job distribution

In the Grid setup section, under Grid Presets, click on the Add New Grid Preset... button.

**Preset name** The preset name will be specified by users when submitting a job to the grid. See section 6.3.12). Preset names can contain alphanumeric characters and hyphens. Hyphens cannot be used at the start of preset names.

**Native library path** The full path to the grid-specific DRMAA library.
CHAPTER 6. JOB DISTRIBUTION

Figure 6.6: The grid preset configuration form. The Shared native specification field is only visible when the Resource Aware grid mode is selected.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Shared work directory</strong></td>
<td>The path to a directory that can be accessed by both the CLC Server and the Grid Workers. Temporary directories are created within this area during each job run to hold files used for communication between the CLC Server and Grid Worker.</td>
</tr>
<tr>
<td><strong>Path to CLC Grid Worker</strong></td>
<td>The path to a directory on a shared file system that is readable from all execution hosts. If this directory does not exist, it will be created. The CLC Grid Worker, including a JRE matching the CLC Server JRE and associated settings files, are extracted from the installation area of the CLC Server software and are deployed to this location when the grid preset is saved, and also whenever plugins are updated.</td>
</tr>
<tr>
<td><strong>Job category</strong></td>
<td>The name of the job category - a parameter passed to the underlying grid system.</td>
</tr>
<tr>
<td><strong>CLC Grid Worker memory limit</strong></td>
<td>The maximum amount of memory the CLC Grid Worker java process should use. This field is useful if the memory limit set by auto-detection is not appropriate. The value is given as a number followed by a unit, 'g', 'm' or 'k' for gigabytes, megabytes or kilobytes respectively. For example, &quot;10g&quot; would limit the CLC Grid Worker java process to using a maximum of 10 GB of memory. If left blank, auto-detection is used to determine the limit. See section 6.3.14 for further details.</td>
</tr>
<tr>
<td><strong>Grid mode</strong></td>
<td>There are two grid modes for backwards compatibility reasons. The &quot;Resource Aware&quot; mode is generally recommended. Choosing this mode allows jobs that require few resources to run concurrently on a given node. For this, the field <strong>Shared native specification</strong> must also be configured. This is described further below. If &quot;Legacy mode&quot; is selected, all jobs submitted to the grid from the CLC Server will request use of the entire node. &quot;Legacy Mode&quot; is the default, but this may change in the future.</td>
</tr>
<tr>
<td><strong>Native specification and Shared native specification</strong></td>
<td>Parameters to be passed to the grid scheduler are specified here. For example, a specific grid queue or limits on numbers of cores. Clicking on the f(x) next to the field name pops up a box containing the variables that will be evaluated at run time. These are described further below. The <strong>Native specification</strong> field contains the information to be passed to the grid scheduler for exclusive jobs, those where the whole execution node will be used. The <strong>Shared native specification</strong> contains the information to be passed to the grid scheduler for jobs classified as non-exclusive. Such jobs can share the execution node with other ...</td>
</tr>
</tbody>
</table>
jobs. This specification is only visible and configurable if the "Resource Aware" grid mode is selected.

<table>
<thead>
<tr>
<th>Grid mode</th>
<th>Exclusive Jobs</th>
<th>Non-exclusive jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Legacy</td>
<td>Native specification</td>
<td>NA - all jobs treated as exclusive</td>
</tr>
<tr>
<td>Resource aware</td>
<td>Native specification</td>
<td>Shared native specification</td>
</tr>
</tbody>
</table>

Table 6.1: Summary of grid modes and the specifications used for exclusive jobs, requiring a whole node, and non-exclusive jobs, which can share a node with other jobs.

Exclusive, streaming and non-exclusive tasks are described in section 6.6.1 and further configuration for running concurrent jobs is described in the section on Multi job processing on grid 6.3.9

Below are examples of OGE-specific arguments one could provide in the native specification field of a grid preset. Please refer to your grid scheduler documentation for information on the options available for your grid system.

**Example 1:** To redirect standard output and error output, this in a native specification field:

```
-o <path/to/standard_out> -e <path/to/error_out>
```

would result in the following `qsub` command being generated:

```
qsub my_script -o <path/to/standard_out> -e <path/to/error_out>
```

**Example 2:** To use a specific OGE queue for all jobs, this in a native specification field:

```
-hard -l qname=<name_of_queue>
```

would lead to the following `qsub` command:

```
qsub my_script -q queue_name
```

**f(x) - adding variables to be evaluated at run-time**

Grid Presets are essentially static in nature, with most options being defined directly in the preset itself. There are, however, 5 variables that can be specified that will be evaluated at runtime.

To aid with the required syntax, click on the f(x) link and choose the variable to insert. The variables can also be entered directly into the specification by typing the variable name between a pair of curly brackets.

The available variables are:

- **USER_NAME** The name of the user who submitted the job. This variable might be added to log usage statistics or to send an email to an address that includes the contents of this variable. For example, something like the following could be put into a native specification field:
CHAPTER 6. JOB DISTRIBUTION

-M {USER_NAME}@yourmailserver.com

**COMMAND_NAME**  The name of the CLC Server command to be executed on the grid by the clcgridworker executable. One example of the use of this variable is to specify 
-`q {COMMAND_NAME}` if there were certain commands to be submitted to queues of the same name as the command.

**COMMAND_ID**  The ID of the CLC Server command to be executed on the grid.

**COMMAND_THREAD_MIN**  A value passed by non-exclusive jobs indicating the minimum number of threads required to run the command being submitted. This variable is only valid for Shared native specifications.

**COMMAND_THREAD_MAX**  A value passed by non-exclusive jobs indicating the maximum number of threads supported by the command being submitted. This variable is only valid for Shared native specifications.

**Using functions in native specifications**
Two functions can be used in native specifications `take_lower_of` and `take_higher_of`. These are invoked with the syntax: `{#function arg1, arg2, [... argn]}`: These functions are anticipated to be primarily of use in Shared native specifications when limiting the number of threads or cores that could be used by a non-exclusive job, and where the grid system requires a fixed number to be specified, rather than a range.

- **take_lower_of**  Evaluates to the lowest integer value of its argument.
- **take_higher_of**  Evaluates to the highest integer-value of its argument.

In both cases, the allowable arguments are integers or variable names. If an argument provided is a string that is not a variable name, or if the variable expands to a non-integer, the argument is ignored. For instance `{#take_lower_of 8,4,FOO}` evaluates to 4 and ignores the non-integer, non-variable "FOO" string. Similarly, `{#take_higher_of 8,4,FOO}` evaluates to 8 and ignores the non-integer, non-variable "FOO" string.

An example of use of the `take_lower_of` function in the context of running concurrent jobs on a given grid node is provided in section 6.3.9.

### 6.3.8 Controlling the number of cores utilized

This section covers basics related to setting limits on core or thread usage via a CLC Server grid preset. Parameter details are specific to the grid scheduler being used. We provide some information below for some schedulers, but please refer to the grid scheduler documentation for full details.

When using "Legacy mode" grid mode, or when running exclusive jobs with the "Resource Aware" grid mode, the default for all jobs is to assume they have access to all cores on the node they are run on. Details about grid modes can be found in section 6.3.7.

When configuring a core or thread limit for exclusive jobs, i.e. those that will require use of a whole node, the relevant parameter(s) and integer value(s) to be used by the grid scheduler are entered directly in the Native specification field. To specify different core limits for different
types of tasks, one could set up multiple presets with different values supplied in the Native specification field of each.

Configuration of core requirements is central to supporting concurrent execution of non-exclusive jobs on a grid node. This is done by specifying core or thread requirements in the Shared native specification, making use of the variables `COMMAND_THREAD_MIN`, `COMMAND_THREAD_MAX` and optionally the functions `take\_lower\_of` and `take\_higher\_of`. Further information about this is provided in section 6.3.9.

**Configuration of OGE**

1) **CPU Core usage when not using parallel environment**

In the CLC Server, there is an environmental variable, which when set to 1, specifies that the number of allocated slots should be interpreted as the maximum number of cores a job should be run on. To set this environmental variable, add the following to the Native specification of the grid preset:

```
-v CLC_USE_OGE_SLOTS_AS_CORES=1
```

In this case, the maximum number of cores the job should use will be set to the number of slots allocated by OGE for the job.

2) **Limiting CPU core usage when using the parallel environment feature**

The parallel environment feature can be used to limit the number of cores used by the CLC Server jobs when running on the grid. When the parallel environments feature is used, the number of allocated slots is interpreted as the maximum number of cores to be used by the job. The parallel environment must be setup by the grid administrator in such a way that the number of slots corresponds to the number of cores.

The syntax in the Native specification for using parallel environments is:

```
-pe <pe-name> <min-cores>-<max-cores>
```

where `pe-name` is the name of the parallel environment, and a range of cores is specified with integers, e.g. 1-4.

An example as might be entered into a Native specification when using parallel environments is:

```
-l cfl=1 -l qname=64bit -pe clc 1-3
```

Here, the clc parallel environment is selected and 1 to 3 cores are requested.

**Configuration of PBS Pro**

With PBS Pro the number of cores to use is specified with a single number. This request can be granted (the process is scheduled) or denied (the process is not scheduled). The number of cores are requested as a resource: `-l nodes=1:ppn=X`, where `X` is the number of cores. Please ensure that the number of nodes requested is 1.

An example as might be entered into a Native specification is: 

```
-q bit64 -l nodes=1:ppn=2
```

This would request 2 cores and the job would be put in the `bit64` queue.

**Configuration of LSF**

With LSF the number of cores to use is specified with the `-n` option. This parameter can accept a single argument or two arguments. A single argument, `-n X`, is a request for exactly `X` cores.
Two arguments, \(-n \ X,Y\), (separated by a comma), is a request for between X and Y cores.

### 6.3.9 Multi-job processing on grid

Certain types of CLC Server jobs, known as "non-exclusive" jobs, can be scheduled to run concurrently on the same grid node when appropriate. Non-exclusive jobs are those that have reasonably low demands for system resources. A list of such jobs is provided in the Appendix, section 20.2.

There are two ways non-exclusive jobs can be configured to run concurrently on a grid node:

1. **In the context of a workflow or workflow block executed on a single grid node**
   
   If the server has been configured to send all tasks in a workflow to a single node, or to send tasks in a workflow block to a single node, as described in section 6.5, and the workflow or workflow block includes parallel non-exclusive tasks, then if these may run concurrently on the grid node.

   By default, up to 10 such non-exclusive jobs can be run concurrently. This value can be changed in the "Maximum number of concurrent jobs" field, available in the "Server setup" area of the Job Distribution tab. That field is visible when the server mode "Single server" has been selected, as shown in figure 6.7. The value configured is passed through to the CLC Server queue with the grid worker.

   ![Figure 6.7: The Maximum number of concurrent jobs setting is visible when the Single server mode is selected.](image)

   A grid setup can be run with the Single server or Master node setting selected. To adjust the "Maximum number of concurrent jobs" if the Master node option is selected, temporarily change the mode to Single server, set the desired value, and then change the mode back to Master node.

   **Limitation:** The maximum value that can be entered in the "Maximum number of concurrent jobs" field is the number of cores on the master server. Setting that maximum value is the equivalent of checking the unlimited box. If this value is smaller than the desired value, as might happen when grid nodes have many more cpu than the master server, then we recommend leaving this field blank so that the default value of 10 is used.

2. **In any other context**
   
   Information about the CPU or thread requirements of the jobs must be passed to the grid scheduler. Non-exclusive algorithms expose their CPU or thread usage, and this
information can be passed on to the grid scheduler via the COMMAND_THREAD_MIN and
COMMAND_THREAD_MAX variables in the Shared native specification of a grid preset.
The variable COMMAND_THREAD_MAX would be used alone as an argument when a
single value should be specified, or both the variables COMMAND_THREAD_MIN and
COMMAND_THREAD_MAX can be provided when a range is required. An example of
specifying a range is shown in the image of a grid preset in section 6.3.7.
One can also use the functions take_lower_of and take_higher_of for settings
relevant to configuring multiple job processing. For example, to specify 4 as the maximum
number of cores to be used by a non-exclusive job, the following could be used as the
argument to the relevant parameter in the Shared native specification of a grid preset:
{#take_lower_of COMMAND_THREAD_MAX, 4}. As the non-exclusive job passes on
its thread usage requirements via the COMMAND_THREAD_MAX variable, this evaluates to
4 if that requirement is higher than 4 or the value specified by the job if is lower than 4.
Further details about grid preset configuration, including Shared native specifications,
functions and variables, can be found in section 6.3.7.

Licensing notes

Each CLC Grid Worker launched, whether it is to run alone on a node or run alongside a
job already running on a particular node, will attempt to get a license from the CLC Network
License Manager. Once the job is complete, the license will be returned.
If the server has been configured to send all tasks in a workflow to a single node, only a
single CLC Grid Worker will be launched for a given workflow run. Thus, irrespective of the
number of concurrently running jobs in such a workflow run, only a single gridworker license
is used.
If the server has been configured so that each task in a workflow is submitted separately,
then a CLC Grid Worker will be launched for each task, resulting in the use of a gridworker license
for each task, including non-exclusive jobs executed concurrently.
If the server has been configured so that each block of a workflow is submitted to run on
a single node, a CLC Grid Worker will be launched for each block. If the workflow contains
no iteration blocks, then it, by definition, consists of a single block and will use a single
gridworker license. If a workflow contains one or more iteration blocks, it will consume a
number of licenses equal to the number of iterations of these blocks plus one for each
additional, non-iterated block. See figure 6.13 in section 6.5 for more detail about workflow
blocks.

6.3.10 Other grid worker options

Some java options can be set specifically for each grid worker. To do this, create a file called
clcgridworker.vmoptions and place it in the folder specified in the Path to CLC Grid
Worker field of the relevant grid preset. For information on the relationship between grid workers
and grid presets, see section 6.3.7.
Options that may be of particular use for grid workers include:

- -Djava.io.tmpdir=<path> Set the location where temporary files should be put.
- -Dskip_lazytmp_cleanup=<boolean> When set to true,
  -Dskip_lazytmp_cleanup=true, the grid worker will not attempt to clean up temporary
data from previous analyses that, for whatever reason, were not cleared up previously. This option is intended for systems where temp areas for grid nodes are not local and the specified shared temp location is on a file system that does not support global file locking. As we highly recommend that temporary data areas are local, this option should rarely be of use.

For example, if a `clcgridworker.vmoptions` was created, containing the following line, it would, for the grid worker specified in a given preset, set a temporary directory for the grid nodes, overriding the default that would otherwise apply:

```bash
-Djava.io.tmpdir=/path/to/tmp
```

Proxy settings for grid setups are also configured in `.vmoptions` options files, as described in section 20.7.1.

**Memory settings for grid workers** If memory limits set using auto-detection, described in section 6.3.14, need to be overridden, then we recommend this be done using the **CLC Grid Worker memory limit** field of the grid preset configuration, as described in section 6.3.7. In earlier versions of CLC Server, a line of the form `-Xmx<value>` may have been added to a `clcgridworker.vmoptions` file to specify this limit. If setting limits in a grid preset, any `-Xmx` settings from `clcgridworker.vmoptions` files, should be removed, as values in these files will override those in a grid preset.

### 6.3.11 Testing a Grid Preset

There are two types of tests that can be run to check a Grid Preset. The first runs automatically whenever the **Save Configuration** button in the Grid Preset configuration window is pressed. This is a basic test that looks for the native library you have specified. The second type of test is optional, and is launched if the **Submit test job...** button is pressed. This submits a small test job to your grid and the information returned is checked for things that might indicate problems with the configuration. While the job is running, a window is visible highlighting the jobs progression as shown in figure 6.8.

![Figure 6.8: Testing a Grid Preset.](image-url)
CHAPTER 6. JOB DISTRIBUTION

6.3.12 Client-side: starting CLC jobs on the grid

Starting grid jobs

Submitting jobs to grid nodes from a CLC Workbench

Once the CLC Server is configured and you are logged into it via a CLC Workbench, an extra option will appear in the first dialog box presented when setting up a task that could be executed on the CLC Server. At this stage, you can choose to execute the task on the machine the Workbench is running on, the CLC Server machine, or to submit the job to one of the available grid presets. To submit to the grid is as simple as choosing from among the grid presets in the drop down box, as shown in figure 6.9.

![Figure 6.9: Starting the job on the grid.](image)

Submitting jobs to grid nodes using the CLC Server Command Line Tools

Submitting jobs for execution on grid nodes using the `lcserver` command of the CLC Server Command Line Tools involves adding the `-G` parameter to the command, followed by the name of the grid preset to send the job to as the parameter value. A list of available grid presets is returned, along with other information about the lcserver command, if the command is run with just the server address, port and user credentials specified. Further details about the `lcserver` command can be found in the CLC Server Command Line Tools manual at [https://resources.qiagenbioinformatics.com/manuals/clcservercommandlinetools/current/index.php?manual=Basic_usage.html](https://resources.qiagenbioinformatics.com/manuals/clcservercommandlinetools/current/index.php?manual=Basic_usage.html).

6.3.13 Grid Integration Tips

If you are having problems with your CLC Grid Integration, please check the following points:

- Does your system meet the requirements of the CLC Grid Integration tool 6.3.2? For example, please check that the machine the CLC Server is running on is configured as a submit host for your grid system.

- The user running the CLC Server process is the same user seen as the submitter of all jobs to the grid. Does this user exist on your grid nodes? Does it have permission to submit to the necessary queues, and to write to the shared directories identified in the Grid Preset(s) and any `clcgridworker.vmoptions` files?

- Are your CLC Server file locations mounted with the same path on the grid nodes as on the master CLC Server and accessible to the user that runs the CLC Server process?
If you installed the CLC Server as root, and then later decided to run it as a non-privileged user, please ensure that you stop the server, recursively change ownership on the CLC Server installation directory and any data locations assigned to the CLC Server. Please restart the server as the new user. You may need to re-index your CLC data locations (section 16.2.1) after you restart the server.

Is the SGE_ROOT variable set early enough in your system that it is included in the environment of services? Alternatively, did you edit the CLC Server startup script to set this variable? If so, the script is overwritten on upgrade - you will need to re-add this variable setting, either to the startup script, or system wide in such a way that it is available in the environment of your services.

Is your DRMAA library 64-bit? DRMAA must be for 64-bit systems for it to work.

6.3.14 Understanding memory settings

Most work done by the CLC Server is done via its java process. However, some tools use external native binaries for the computational phases. These include those with a de novo assembly or mapping phases and those involving BLAST tools.

Java process

Each grid preset can have its own memory limit for the java process. This can be useful where separate queues are used for low overheads tasks, such as import jobs and trimming jobs, and high overhead tasks, such as de novo assemblies or read mappings.

Unless a memory limit is explicitly configured for a grid worker, auto-detection is used to determine the value to apply. This should be appropriate for most circumstances. The limit is determined as follows:

- If virtual memory is limited (ulimit -v), a quarter of that value or 50GB, whichever is smaller. Otherwise:
- If resident memory is limited (ulimit -m), half of that value or 50GB, whichever is smaller. Otherwise:
- Half of the amount of available physical memory or 50GB, whichever is smaller.

Order of priority for java process memory settings The first value found according to the list below will be applied:

1. A memory limit specified in a clcgridworker.vmoptions file. This method is not recommended.
2. A memory limit specified in a CLC Grid Preset. If values determined by auto-detection are not appropriate, this method can be used to specify a custom value. See section 6.3.7.
3. Auto-detection, as described above.

External binaries
Some tools make use of external binaries during one or more phases. Memory usage by external binaries is not restricted by limits set for the java process. For this reason, we recommend caution if you plan to submit jobs of these types to nodes that are being used simultaneously for other work.

### 6.4 Model III: Single Server setup

In this model, the master and execution node functionality is carried out by a single CLC Server instance. Here, the CLC Server software is installed on a single machine. Jobs submitted to the server are executed on this same machine.

To designate the system as a single server, after installation and starting the server, go to the Job distribution tab in the web administrative interface:

![Configuration and Job distribution options](image)

Select the option SINGLE_SERVER from the drop down list of Server modes.

Figure 6.10: The configuration options for the types of machines running the CLC Server. The choices of relevance under normal circumstances are SINGLE_SERVER and MASTER_NODE. An administrator will not usually need to manually choose the Execution Node option. This option is there primarily to allow for troubleshooting.

You can then configure aspects of the server:

- **Master node host** - usually set to localhost for Single server mode. Click on the "Show suggestions" text next to this field to see information about the server that can be useful if localhost is not a suitable option for your setup.

- **Master node port** - usually 7777.

- **Master node display name** - the name shown in the top bar of the web interface for the server.

- **CPU limit** - The maximum number of CPU the CLC Server should use. This is set to unlimited by default, meaning that up to all cores of the system can be used.

- **Maximum number of concurrent jobs** - Limit the maximum number of jobs that are allowed to run concurrently on the single server. For further information about this setting see section 6.6.3.
6.5 Workflow queuing options

Workflow queuing options are found under:

Configuration (.Configuration) | Job distribution (Job distribution)

There are three options for how workflows should be submitted to nodes (figure 6.13). These options reflect the three organizational levels of workflows:

- **Tasks** The individual tasks that comprise the workflow
- **The whole workflow** All tasks in the workflow
- **Blocks** Sections of a workflow. For workflows without control flow elements, the entire workflow is a single block. For workflows with control flow elements:
  - Tasks downstream of an *Iteration* element and above a *Collect and Distribute* element form a block.
  - Tasks downstream of an *Iteration* element with no subsequent *Collect and Distribute* element form a block.
  - Linked workflow steps outside an iteration block form a block. See figure 6.12.

Further details about control flow elements are provided in the Workflow chapter of the CLC Genomics Workbench manual available in html or pdf format from https://digitalinsights.qiagen.com/technical-support/manuals/

The workflow queuing options are:

- **Submit individual tasks to any available node** Each task of a workflow is scheduled separately for execution. For example, a workflow with 10 tasks would result in 10 jobs being submitted. Each of those jobs can be sent to any available node with adequate resources when that step is ready to be run.

- **Submit all tasks to a single node** A single job submission is made for a workflow, regardless of how many tasks that workflow consists of. All tasks of that workflow are run on the same node. This option was previously called "Single entity".

- **Submit tasks in each workflow block to a single node** Each iteration of a block of a workflow, and each additional block outside iteration blocks, is scheduled separately for
Figure 6.12: This workflow has one iteration block, shaded in turquoise, and one block after the iteration block, shaded in purple. It also has an optional block above the iteration block, outlined in purple. If a user chooses to import files as the initial action taken when launching this workflow, the import would be executed as an initial block. If they choose files already in a CLC File Location, the first block is the iteration block.

execution. Each job can be sent to any available node with adequate resources when that block is ready to be run. For workflows consisting of just one block, (no control flow elements), this option behaves just like the Submit all tasks to a single node option.

Further information about workflow queuing options is provided in section 6.5.1.

Figure 6.13: Workflow queuing options determine how workflow jobs are queued on servers with execution nodes.

6.5.1 Choosing a workflow queuing option

In general terms:

- With the Submit individual tasks to any available node option, each task is scheduled individually. Where large numbers of workflows are submitted and each workflow consists
of several tasks, it can lead to a large scheduling overhead, with thousands of jobs being placed in the queue. However, on a multi-node system with spare capacity, this option provides the highest potential level of parallelization.

- With the **Submit all tasks to a single node** option, a single job submission is made for a workflow, regardless of how many tasks that workflow consists of, so scheduling overhead is minimized. However, only non-exclusive jobs within a workflow have the potential be executed in parallel.

- With the **Submit tasks in each workflow block to a single node** option, the scheduling overhead is low, with a single job submission per workflow block to be executed. Here, iterated workflow blocks have the potential to be executed in parallel on multi-node setups.

Further considerations relating to the choice of workflow queuing option include:

- Workflow blocks and full workflows run on a single node can leverage caching mechanisms, which aids performance.

  On systems that frequently run at or close to capacity, the opportunity for gains through concurrent processing of parallel workflow tasks on multiple nodes is much lower than on a system with spare capacity. So here, the performance gains using the **Submit all tasks to a single node** or **Submit tasks in each workflow block to a single node** option can outweigh those of submitting tasks separately, even when they are computationally intensive tasks.

- Where resource allocation is a focus, such as where many users are sharing the resources, the **Submit all tasks to a single node** option may help with resource access by different users.

  For example, consider a grid node setup with 20 nodes, where one user submits 15 workflows with 10 tasks in each workflow. If each task is submitted as an individual job, those 150 jobs can be submitted across all 20 nodes. When the next user submits a job, it would be queued behind these 150 jobs. With the **Submit all tasks to a single node** option, the 15 workflows would have been sent to a maximum of 15 nodes, leaving 5 nodes available for the other user’s job. Where workflows do not contain iteration blocks, the **Submit tasks in each workflow block to a single node** option would have the same effect.

- Blocks or entire workflows are only submitted to nodes capable of running all the included tasks. Where node hardware is not homogeneous or where certain job nodes have been dedicated to running only certain types of analyses, this should be considered when selecting the **Submit all tasks to a single node** or **Submit tasks in each workflow block to a single node** options.

- For grid node setups only: Where demand for gridworker licenses exceeds the number available, jobs wait in the queue until a license becomes available. The number of gridworker licenses needed to execute all tasks in a given workflow depends on the workflow queuing option selected.

  With the **Submit individual tasks to any available node** option, the number of gridworker licenses needed equals the number of tasks. When the **Submit all tasks to a single node** option is selected, only one gridworker license will be needed for the whole workflow. This is also the case with the **Submit tasks in each workflow block to a single node** option when running workflows without an iteration block. For a workflow with one or more iteration
blocks, the number of licenses used will be equal to the number of iterations of the blocks within the workflow plus one for each additional, non-iterated block.

6.6 Job running options

Job running options are found under:

Configuration  |  Job distribution

These settings affect how jobs are queued and run on single server and job node setups, as well as where intermediate results of workflows are stored.

![Job running options settings](image)

Figure 6.14: The default settings in the "Job running options" section of the administrative interface.

6.6.1 Multi-job processing

There are three general categories of tools on the CLC Server: non-exclusive, streaming and exclusive, described in more detail below. Non-exclusive or streaming jobs can run concurrently alongside others of the non-exclusive type on a given machine. Those defined as exclusive cannot be run on the same server or node at the same time as other jobs of any type.

When the "Enable" option in the Multi-job processing area is selected, more than one analysis can run simultaneously on a CLC Server in single server mode or on a job node. This setting does not affect jobs sent to grid nodes. Concurrent job processing on grid nodes is described in section 6.3.9.

The maximum number of jobs that can be run concurrently on a single server or job node can be configured, as described in section 6.6.3.

Running non-exclusive and streaming jobs concurrently on a single server or job node can be disabled by setting the "Multi-job Processing" option to "Disable" (figure 6.15). Click on Save to save changes to this setting.
Non-exclusive, streaming and exclusive job types

- **Non-exclusive algorithms** Tools with low demands on system resources. They can be run alongside other jobs in this category, as well with alongside a job of the streaming category, described below. An example of a non-exclusive algorithm is "Convert from Tracks".

- **Streaming algorithms** Tools with high I/O demands, that is, much reading from and writing to disk is needed. These cannot be run with other jobs in the streaming category but can be run alongside jobs in the non-exclusive category. Examples of streaming algorithms are the NGS data import tools.

- **Exclusive algorithms** Tools optimized to utilize the machine they are running on. They have high I/O bandwidth, memory, or CPU requirements and therefore should not be run at the same time as other jobs on the same machine. An example is "Map Reads to Reference".

See Appendix section 20.2 for a list of CLC Genomics Server algorithms that can be run alongside others on a given machine.

### 6.6.2 Fairness factor

The fairness factor defines the number of times that a job in the queue can be overtaken by other jobs before resources are reserved for it to run. This setting affects the queue for single servers and job node setups.

In a situation where there are many non-exclusive jobs and some exclusive jobs being submitted, it is desirable to be able to clear the queue at some point to allow the exclusive job to have a system to itself so it can run. The fairness factor setting is used to determine how many jobs can move ahead of an exclusive job in the queue before the exclusive job will get priority and a system will be reserved for it. The same fairness factor applies to streaming jobs being overtaken in the queue by non-exclusive jobs.

The default value for this setting is 10. With this value set, a job could be overtaken by 10 others before resources are reserved for it that will allow it to run. A fairness factor of 0 means that a node will be reserved for the job at the head of the queue.

This value can be configured under the **Fairness factor** section of:

- Configuration ( CONFIG )  |  Job distribution ( JOB )  |  Job running options ( OPT )
6.6.3 Concurrent jobs per node

The maximum number of jobs that can be run on a single server or on each job node can be configured. The maximum allowable value is equal to the number of cores on the relevant system. The default value is 10 or the number of cores on the system, whichever is lowest. If a CPU limit has been set on the single server or node, then the default is 10 or that CPU limit value, whichever is lowest.

See section 6.6.1 for further information about the types of jobs that can be run at the same time. While non-exclusive algorithms are generally expected to have low demands on system resources, when working with very large genomes, a smaller maximum value can make sense. Alternatively, certain nodes could be reserved for use by only certain tools, as described in section 5.2.

**Single server setup** To configure the maximum number of jobs that can be run concurrently on a single server, go to the section **Server setup** under the Job Distribution tab of the web administration page. Enter the desired value in the box labeled "Maximum number of concurrent jobs" (figure 6.16).

![Server setup](image)

Figure 6.16: Set the maximum number of concurrent jobs or check the "unrestricted" box. If unrestricted is chosen, the maximum number of jobs that can be run concurrently is equal to the number of cores on the system.

**Job node setup** To configure the maximum number of jobs that can be run concurrently on a given job node, go to the section **Job queuing options** under the Job Distribution tab of the web administration page. Enter a value for each job node listed under the "Concurrent jobs per node" section (figure 6.17).

6.6.4 Intermediate workflow result handling

Workflows contain a series of tasks, most of which produce data. Some data elements produced during a workflow execution are **intermediate results**. These are used temporarily, during the workflow run, to provide input to the next step of the analysis. Intermediate results are deleted once the workflow execution successfully completes.

Where intermediate results should be saved is controlled using the **Intermediate workflow result handling** options.
Figure 6.17: Set the maximum number of concurrent jobs or check the "unrestricted" box for any job node. If unrestricted is chosen, the maximum number of jobs that can be run concurrently is equal to the number of cores on that job node.

- **In the location final analysis results are stored** Intermediate results are stored in a subfolder of the output folder selected when the workflow is launched.

- **In a temporary directory** Intermediate results are stored in a temporary directory on the system the job is executed on, i.e. the single server, job node or grid node. This is usually faster than storing intermediate results under the output folder, particularly if the output folder is on a network drive. Note that if this option is selected, there must be enough temporary space to hold all the intermediate results on the systems that workflows are executed on.
Chapter 7

Access to files outside CLC Server Locations

7.1 Import/export directories

In various situations, the CLC Server needs to read from or write to files on the filesystem outside areas intended for CLC data. Examples include when the data will be imported into the CLC Server and when BLAST databases are made available to be searched using functionality of the CLC Server. Areas that the CLC Server should have read and write access to are configured as Import/export directories.

In the External data area under the Configuration tab, click on the "Import/export directories" heading to add import/export directories. Click on the Add new import/export directory button to enter the relevant path (figure 7.1). The specified folder and its subfolders will then be available to CLC Workbenches logged into the CLC Server when relevant activities, such as import and export, are carried out.

Permissions can be set on import/export directories, restricting access to certain groups. This is done under the Global permissions area of the web administrative interface, under the Configuration tab.

When direct data transfer is enabled, CLC Workbench users can import data from their local file system to the CLC Server. These files are transferred over the network, placed as temporary files in the area configured for this purpose, which will be an import/export directory, and then imported. Direct data transfer configuration is described in section 7.2.

Whether or not direct data transfer is enabled, users connecting to the CLC Server via client software can select files from areas configured as import/export directories.

Requirements for import/export areas

- Folders configured as import/export directories must be readable and writable by the user that runs the CLC Server process. Users logged into the CLC Server from their Import/export directories can access files in that area, and potentially write files to that area. However, it is the user running the server process that actually interacts with the file system.

- Import/export directories should NOT be set to subfolders of any defined CLC file system
location. CLC Server file system locations are intended for data imported into or generated by CLC software. Import/export directories are intended for holding other data, for example, sequencing data that will be imported, data that is exported from the CLC Server, or BLAST databases.

- On grid setups and job node setups, an area configured as import/export directory must be a shared directory, accessible from the nodes and from the master server.

**Figure 7.1:** Import/export directories are configured in the External data area. One of these areas can be selected when enabling direct data transfer.

**Browsing import/export areas** You can browse the contents of import/export directories by clicking on the **Browse directories...** button (figure 7.1). If you then select a file (figure 7.2), and click on the **OK** button, a window will pop up containing the CLC URL and file path for that file (figure 7.3). This can be useful if you are using the CLC Server Command Line Tools.

**Figure 7.2:** The contents of import/export directories can be browsed via the web interface. Select a file if you wish to determine the CLC URL or path of that file.
7.2 Direct data transfer from client systems

By default, transfer of data or files local to client software directly to the CLC Server is not allowed (figure 7.4).¹

Settings related to direct data transfer are found under the External data area under the Configuration tab. Click on the "Direct data transfer from client systems" heading.

To enable direct data transfer, the option Files uploaded via Import/Export location must be selected. For this option to be available, at least one import/export directory must be configured (see section 7.1).

If permissions are enabled on the selected import/export directory, the relevant groups must be granted write permission to be allowed to transfer data from their client system to the CLC Server directly.

The default setting, "Not allowed", is conservative, and limits certain functionality, including:

- Importing data from a client system (e.g., a system the CLC Workbench or CLC Server Command Line Tools is on) to a CLC Server file system location is not allowed.
- Workflows cannot be installed on the CLC Server using the functionality provided by the CLC Workbench Workflow Manager.
- Running a standard external application that has parameters referring to files local to a client system is not possible.
- Data cannot be downloaded to a CLC_References location on the CLC Server via a CLC Workbench Reference Data Manager.
- Installing or updating plugins on connected job nodes will fail. (When pushing plugin updates to the nodes, the master is acting as a client to a job node.)

¹The previous default option, "Files uploaded via temporary file location on server system(s) (legacy)" was removed in CLC Genomics Server 20.0. When upgrading from an earlier version where that option was selected, the new default, "Not allowed" is applied in the updated version.
CHAPTER 7. ACCESS TO FILES OUTSIDE CLC SERVER LOCATIONS

Figure 7.4: By default, transfer of data directly by client software to the CLC Server is not allowed. The option allowing such transfers, "Files uploaded via Import/Export location" is disabled if no import/export directory are configured.

7.3 AWS Connections in the CLC Server

Access to AWS accounts is configured under:

Configuration | External data | AWS Connections

Click on the Add AWS Connection button. The following information should be entered in the configuration dialog:

- **Connection name**: A short name of your choice, identifying the AWS account. This name will be shown as the name of the data location when importing data to or exporting data from Amazon S3.

- **AWS access key ID**: The access key ID for programmatic access for your AWS IAM user.

- **AWS access secret key**: The secret access key for programmatic access for your AWS IAM user.

- **AWS Region**: An AWS region. Select from the drop-down list.

- **AWS Partition**: The AWS partition for your account.

AWS connections are used when:

- Accessing AWS S3 locations, to import data from or export data to.

- Submitting analyses to a CLC Genomics Cloud setup, if available on that AWS account.

All traffic to and from AWS is encrypted using a minimum of TLS version 1.2. AWS credentials entered are stored, obfuscated, in the server configuration files.

AWS connections are listed, along with information about their status. These can be edited or deleted. The status is indicated using colors. Green indicates the connection is valid and is ready for use.
Connections to a CLC Genomics Cloud, indicated in the CGC column, require the Cloud Server Plugin to be installed, and access to a CLC Genomics Cloud.

Access to AWS connections can be limited to specified groups using options available under the Global permissions tab in the CLC Server web administrative interface.

Clicking on the Browse S3 locations link opens the relevant tab under Element info.

### 7.4 Illumina BaseSpace

When working in a CLC Genomics Workbench, data can be imported from Illumina BaseSpace via the CLC Server (or grid nodes) without explicit configuration in either the CLC Workbench or the CLC Server web administrative interface.


Importing directly from BaseSpace using the CLC Server Command Line Tools is not supported.
Chapter 8

Workflows

A workflow consists of a series of tools where the output of one tool is connected as the input to another tool. A workflow created using a CLC Workbench can be run on a CLC Server when all the tools in the workflow are available on the CLC Server, and the workflow is:

- **Stored in a CLC File Location**, and then opened from the Navigation Area of a CLC Workbench logged into the server and run from the Workflow Editor.
- **Installed on a CLC Workbench** that is logged into the server. When launching, the user can choose to run the workflow on the CLC Workbench or CLC Server.
- **Installed on a CLC Server**. Such workflows can be launched from a CLC Workbench logged into the server. When launching, the user can choose to run the workflow on the CLC Workbench or CLC Server. These workflows can also be launched using the CLC Server Command Line Tools.

A benefit of installing workflows on the CLC Server is that it allows control over the version of a workflow being used, as well as easy deployment of new versions when desired. Changes made to such centrally installed workflows become immediately available for all CLC Workbench users logged into the server.


8.1 Installing and configuring workflows

Workflows installed on a CLC Server are available to launch from the Toolbox of CLC Workbench clients or using the CLC Server Command Line Tools.

Workflows can be installed on the CLC Server using a workflow installer file, or installed as the final step when creating a workflow installer in a CLC Workbench. To install a workflow, you must have the relevant permissions (section 5.2).

Creating workflow installer files is described at:
When logged into the web administrative interface as a user with rights to administer workflows, workflows can be installed by going to:

**Configuration (🍎) | Workflows (🍊)**

Click on the **Install Workflow** button and select a workflow installer file.

Once installed, a validated (✔️) or attention (❗️) status icon will be shown to the left of the workflow name, as shown in figure 8.1.

![Figure 8.1: The workflow named Small Variant Calling Workflow is installed on the server and has been validated. The workflow below, named qctargetsequencing-bxwb5 is installed, but there are issues that need to be addressed, as indicated by the red exclamation mark.](http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Creating_workflow_installation_file.html)

If the workflow installer included bundled reference data, those data elements will be installed into a subfolder of a folder called **Plugin_data** in the first available file system location. The subfolder name will include the name and version of the workflow.

Workflow elements with red text are ones that can be configured. Click on such an element to bring up a dialog with a listing the parameters that can be configured, as well as an overview of the locked parameters. An example is shown in figure 8.2.

Open parameters can be configured, and can also be locked if desired, so that the parameter cannot be changed when executing the workflow. Locking and unlocking parameters is described further here: [http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Basic_configuration_workflow_elements.html](http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Basic_configuration_workflow_elements.html).

If changes to the parameter settings of an installed workflow are made, the timestamp of the
Figure 8.2: In this example only one parameter can be configured, the rest of the parameters are locked for the user.

most recent change and the name of the administrator who made those changes are reported at the top of the workflow configuration view.

8.2 Executing workflows

Execution of workflows on a server setup

How workflows are run on a multi-node server setup depends on options configured under the Job Distribution tab of the web administrative interface. This is described in section 6.5, including outlining key considerations when choosing the most efficient option for a particular setup.

A user perspective on executing workflows

When you log in to a CLC Server from a CLC Workbench, workflows installed on the CLC Server are listed in the Installed Workflows folder of the Toolbox. These workflows have a blue S in their icon, signifying that they are installed on the server (figure 8.3).

Figure 8.3: One workflow is installed on the CLC Server, as signified by the blue S in its icon. The other workflow is installed locally, on the Workbench.

Workflows available from a CLC Server can be run on any available execution environment (figure 8.4).
CHAPTER 8. WORKFLOWS

8.3 Updating workflows

CLC software tools are versioned. The version of a tool in a workflow is the version that was available on the CLC Workbench when the workflow was created. The exception to this is external applications available from the CLC Server, where the version present on the server at the time workflow was created is included.

If a new tool version becomes available through upgrading the CLC Server, upgrading plugins, or updating an external application, then any workflow containing such tools must be updated. That is, the version of a tool in a workflow must be the same as the version present in CLC software for that workflow to be executable.

An exclamation mark (¡) is presented beside any workflow that needs to be updated. Clicking on the workflow name opens up a view where each element that needs to be updated is also indicated with an exclamation mark. See figure 8.5. Workflows that can be updated directly in the web administrative interface will have a button at the bottom labeled "Update Workflow" enabled. For workflows that cannot be updated this way, a message will appear stating this, and providing some tips of how to proceed. Details of both these situations are outlined below.

Updating workflows via the server web administrative interface

A button labeled "Update Workflow" just under the workflow is enabled for workflows that can be updated directly in the web administrative interface. Clicking on this button, or any of the elements with exclamation marks, starts the update. See figure 8.5.

Note! If a tool has been updated with a new parameter, then an updated workflow that includes that tool will have that new parameter configured with the default value.

When updating, a window appears containing information about the changes to be enacted if you proceed. If errors have occurred these will also be displayed. See figure 8.6. Accept the changes by pressing the "Update" button. The update can also be canceled at this point if desired.

After pressing the "Update" button, the updated workflow will be marked with a green check mark (✔). A copy of the original workflow is also kept. It is disabled and has the original name with "-backup (disabled)" appended. An example is shown in figure 8.7.

Figure 8.4: When launching a workflow from a Workbench, the first wizard step displays a list of environments the workflow can be run on.
Figure 8.5: Click on the "Update Workflow" button, or on an element marked with an exclamation mark to start the update.

If you click on the copy of the original workflow, a button labeled "Re-enable Workflow" appears (figure 8.8). Clicking on this button re-enables the original workflow and uninstalls the updated version of the workflow.

**Updating workflows that cannot be updated via the server web administrative interface**

Some installed workflows cannot be updated directly in the web administrative client. Common situations where this can occur include:

1. Workflows containing tools provided by plugins not installed on the CLC Server.
2. Workflows containing tools from server extensions (commercial plugins) that require a license, but either the license is not present or it does not support the version of the server extension that is installed.
3. Workflows containing tools not on the version of the CLC Server running.
4. Workflows containing tools that cannot be upgraded directly due to the nature of the changes made to them in the updated CLC Server.

To resolve the first 2 circumstances, check install any needed plugins and licenses, restart the CLC Server, and check the status of workflows under the **Workflows** tab of the web administrative interface.
Figure 8.6: Details about the upgrade are presented, and you can choose whether to proceed with the update, or cancel it.

Figure 8.7: In addition to the updated version of the workflow, marked with a green check mark, a copy of the original workflow is kept. It is disabled and has the original name with "-backup (disabled)" appended.

Figure 8.8: After a workflow has been updated, it is possible to re-enable the original workflow.

To address the third and fourth issues, new versions of the workflows must be made on a CLC Workbench and then installed on the CLC Server. For this, a Workbench version that the installed workflow can be run from is needed, as well as the latest version of the Workbench.

**Updating installed workflows when using software in a higher major version line**

To update an installed workflow after upgrading to software in a higher major version line, you need a copy of the older Workbench version, which the installed workflow can be run on, as well as the latest version of the Workbench.

To start, open a copy of the installed workflow in a version of the Workbench it can be run on. This is done by selecting the workflow in the **Installed Workflows** folder of the **Toolbox** in the bottom left side of the Workbench, then right-clicking on the workflow name and choosing the
option "Open Copy of Workflow" (figure 8.9).

Figure 8.9: Open a copy of an installed workflow by right-clicking on its name in the Workbench Toolbox.

Save the copy of the workflow. One way to do this is to drag and drop the tab to the location of your choice in the Navigation Area.

Close the older Workbench and open the new Workbench version. In the new version, open the workflow you just saved. Click on the OK button if you are prompted to update the workflow.

After checking that the workflow has been updated correctly, including that any reference data is configured as expected, save the updated workflow. Finally, click the Installation button to install the workflow, if desired.

If the above process does not work when upgrading directly from a much older Workbench version, it may be necessary to upgrade step-wise by upgrading the workflow in sequentially higher major versions of the Workbench. OR

The updated workflow can now be installed on the CLC Server as described in section 8.1.
Chapter 9

BLAST databases

The CLC Server supports running BLAST jobs submitted from client software. Users can select elements containing query sequences from Server data locations (see section 3.2.1) to search against other sequences held in Server data locations, or against BLAST databases stored in an area configured as an import/export directory (see section 7.1).

9.1 Adding directories for BLAST databases on the Server

In the web interface of the server, you can configure your Server for BLAST databases:

Configuration ( ) | BLAST Databases ( )

Add a folder where the Server should look for BLAST databases by clicking on the Edit BLAST Database Locations button at the bottom of this tab.

The folder selected must already be configured as an import/export directory (see section 7.1) to be listed in the drop-down menu figure 9.1. This is necessary because BLAST databases are stored outside CLC data locations, but must be accessible to the CLC Server process.

![Figure 9.1: Adding import/export directories as BLAST database locations.](image)

Once added as a BLAST Database Location, the CLC Server will search this directory for any BLAST databases and list them under the BLAST tab in the web interface (figure 9.2).

This overview is similar to the one you find in the Workbench BLAST manager for local databases including the following in formation:

- Name. The name of the BLAST database.
9.2 Adding and removing BLAST databases

BLAST databases can be added in two ways:

- Place pre-formatted databases in the directory selected as BLAST database location on the server file system. The CLC Server will automatically detect the database files and list the database as target when running BLAST. You can download pre-formatted database from e.g. ftp://ftp.ncbi.nih.gov/blast/db/.

- Run the Create BLAST Database tool via your Workbench, and choose to run the function on the Server when offered the option in the Workbench Wizard. You will get a list of the BLAST database locations that are configured on your Server. The final window of the wizard offers you a location to save the output to. The output referred to is the log file for the BLAST database creation. The BLAST databases themselves are stored in the designated BLAST database folder you chose earlier in the setup process.

A note on permissions: To create BLAST databases on the Server, using the Workbench interface, the user running the CLC Server process must have file system level write permission on the import/export directory that you have configured to hold BLAST database.
By default, if you do not change any permissions within CLC Server, all users logging into the CLC Server (e.g., via their Workbench, or via the Command Line Tools), will be able to create BLAST databases in the areas you have configured to hold BLAST databases.

If you wish to restrict the ability to create BLAST databases to these areas completely, but still wish your users to be able to access the BLAST databases to search against, then set the file system level permissions on the import/export directory so they are read-only.

When listing the databases as shown in figure 9.2, it is possible to delete the databases by clicking the Delete link at the far right-hand side of the database information.
Chapter 10

Status and management

CLC Server status and management functionality is provided under the Management ( ) tab (figure 10.1). This chapter describes functionality for downloading licenses, putting the server into maintenance mode, stopping and restarting the server, and related activities.

Access to the server queue is described in chapter 11 and the audit log is described in chapter 12.

10.1 Downloading a license via the web interface

Licenses are installed on a single server or on the master node of a job node or grid node setup.

To download and install a license:

- Log into the web administrative interface of the single server or master node as an administrative user.
  
  When there is no existing license file for the CLC Server, or there is a license file that is not valid for the version being run, a warning message is shown (figure 10.2). In these cases, a license file needs to be downloaded.
  
  When a valid license is present, the messages in yellow will not be present (figure 10.3). In this case, no further action to update the CLC Server license is needed. You can skip the next steps in this section.

- Under the Management ( ) tab, open the Download License ( ) tab.

- Enter the Order ID supplied by QIAGEN into the Order ID field and click on the “Download and Install License...” button (figure 10.1).

Please contact ts-bioinformatics@qiagen.com if you have not received an Order ID.

The CLC Server must be restarted for new license files to be loaded. Details about restarting can be found in section 2.7.1.

Each time you download a license file, a new file is created in the licenses folder under the CLC Server installation area. If you are upgrading an existing license file, delete the old file from this area before restarting.
If you are working on a system that does not have access to the external network, then please refer to section 10.1.1.

**Figure 10.1:** License management is done under the Management tab.

**Figure 10.2:** A warning is shown in the web administrative login page for a single server or master node when no license is present for the CLC Server (left) and when a license for the CLC Server is present but is not valid for the version being run (right).

**Figure 10.3:** The web administrative interface login page for a single server or master node when a valid license for the CLC Server is present.

### 10.1.1 Download a static license on a non-networked machine

Follow the steps below to download a static license for a server machine that does not have direct access to the external network.

- Determine the host ID of the machine the server software will be running on. This can be done by running the back-end license download tool, which prints the host ID of the system to the terminal. The download tool is located in the installation folder of the CLC Server. The tool name depends on the system you are working on:
  - Linux: downloadlicense
In the case of a job or grid node setup, the host ID should be for the machine that will act as the CLC Server master node.

- Make a copy of this host ID such that you can use it on a machine that has internet access.
- Go to a computer with internet access, open a browser window and go to the server license download web page:
  https://secure.clcbio.com/LmxWSv3/GetServerLicenseFile
  For licenses for server extensions, the license download page is:
  https://secure.clcbio.com/LmxWSv3/GetLicenseFile
- Paste in your license order ID and the host ID that you noted down earlier into the relevant boxes on the webpage.
- Click on 'download license' and save the resulting .lic file.
- On the machine with the host ID you specified when downloading the license file, place the license file in the folder called 'licenses' in the CLC Server installation directory.
- Restart the CLC software.

### 10.2 Server maintenance

The Server maintenance section is found at:

Management (dı) | Server maintenance (tı)

Settings under the Server maintenance tab allow an administrator to change the operating mode of the server and send out messages to users of the CLC Server (see figure 10.4).

- **Normal Operation** The CLC Server is running.
- **Maintenance Mode** Current jobs are allowed to run and complete, but submission of new jobs is restricted. While the server is in maintenance mode, users already logged in can check the progress of their jobs or view their data, but they cannot submit new jobs. Users not already logged in cannot log in. An administrator can write a warning message, for example, to inform users about the expected period of time the server will be in maintenance mode.
- **Log Out Users** All users currently logged in will be logged out. All running jobs will be allowed to complete. No users can log in while in this mode. An administrator can also write a warning message for the users.
- **Shut down** The CLC Server and any attached job nodes will shut down.
- **Restart** The CLC Server and any attached job nodes will be shut down and restarted.
CHAPTER 10. STATUS AND MANAGEMENT

10.3 User statistics

The User statistics section is found under

Management ('à) | Server status (à)

In this area, information is provided about the number of users logged in, the number of active sessions, and information about each active session (figure 10.5).

A green dot indicates that that user is logged into the CLC Server. Two green dots indicate that the user is logged in twice. For example, they could be logged in from 2 Workbenches running on two different systems. A grey dot means they have previously logged in but are not logged in at this time.

Click on the small plus symbol to the left of a username to expand the information about that user’s sessions (figure 10.6). To log a particular user out, click on the Invalidate Session... button. This opens a dialog where you can write a message to display to the user (figure 10.7). This message is displayed via the user’s active session. For example, if they are logged into a Workbench, a dialog will pop up saying they have been logged out of the CLC Server, followed by the message provided. This action forcibly logs the user out of the CLC Server but it does not stop jobs already submitted or running on the server.
CHAPTER 10. STATUS AND MANAGEMENT

10.4 System statistics

The System statistics section is found under

Management (_visit) | Server status (_visit)

Crashed threads, suggesting system level problems, are reported in this area. In some instances, a system restart may be needed to resolve the issue.

The message "No system level problems detected" is shown in this area if no problems have been detected. An example of the information provided when a problem is detected is shown in figure 10.8. In the case shown, the job submission threads were dead, with the problem reported here and in more detail in the CLC Server log files.

Figure 10.6: Details about jbloggs’ session can be seen by clicking on the small button to the left of that username.

Figure 10.7: Clicking on the Invalidate Session button forcibly logs a user out of the CLC Server. A message can be provided that will be displayed to that user.

Figure 10.8: System level problems detected and reported in the system statistics area.
A list of all the processes that are currently in the CLC Server queue, including jobs in progress can be viewed under the Queue tab:

For each process, you are able to **Cancel** the processes. At the top, you can see the progress of the process that is currently running (figure 11.1).

On single servers or job node setups, after a request to cancel a job is received, the job will remain in the process queue while the cancellation is handled, including cleaning up resources (figure 11.2).
Figure 11.2: The queue just after a job was cancelled on a job node setup, while resources were in the process of being cleaned up.
Chapter 12

Audit log

The audit log records the actions performed on the CLC Server.

Working with the audit log via the web interface

The audit log can be viewed under:

Management ( yapılabilir ) | Audit log ( yapılabilir )

By default, only administrative users have access to this, but this can be extended to members of other groups (see section 5.2).

When first opened, the 50 most recent operations are listed. Older operations can be loaded by clicking on a link at the bottom of the table. Fields at the top can be used to limit the list of operations based on when an action was taken, by whom, what sort of operation it was, and on the job status (success of failure).

Jobs submitted to run on a grid node have a G in the status column and jobs submitted to run on a CLC Genomics Cloud setup have a C in the status column. Every command submitted to the queue goes through the phases "Queued", "Executing", and "Done". The status of failed jobs is visualized by a red exclamation mark in the Status column of a "Command done" entry.

Much information about jobs is available by clicking on the link for each operation. To see all operations for a given job, click on the Show process thread link at the right hand side (figure 12.1).

In cases such as batch jobs, where there is a master process that submits other jobs to the queue, the process threads are color coded. An example is shown in figure 12.2 for a batch job with 2 batch units. Here, the process threads for each batch unit could be viewed by clicking on Show sub process thread at the right hand side.

Click on Show all Threads to return to the full audit log listing.

Note: Data management operations such as copying, deleting and adding files are not Server actions and are thus not recorded in the audit log.

When troubleshooting, it can sometimes be useful to re-launch a particular job. This can be done by finding the "Command queued" operation for the job of interest, and clicking on the link in the "Data in" column (figure 12.3).
CHAPTER 12. AUDIT LOG

Figure 12.1: Each operation is logged in the audit log. Items in red are links. Clicking on these allows you to drill down to detailed information. To see just the operations associated with a given job, click on "Show process Thread" on the right hand side. "Command done operations" are preceded by an arrow pointing left. All other commands have arrows pointing right.

Figure 12.2: The process threads for a batch run of Map Reads to Reference containing 2 batch units are shown. Operations related to the batch as a whole are in red. The 2 mapping jobs each have three process threads shown. Those for the first batch unit are in green, and those for the second are in blue.

Audit log database and text files

Audit log information is stored in a database. Once a month, and when the CLC Server is started up, entries in the audit log older than 3 months are deleted.

The limit the audit log database can grow to is 64 GB. If a new entry will push the size past this limit, the system will remove some of the oldest entries so that is is possible for newer entries to be added.

Audit information is also written to text-based log files. Upon the first activity on a given date, a new log file called audit.log is created. This file is then used for logging that activity and subsequent Server activities on that day. When this new audit.log file is created, the file that previously had that name is renamed to audit.<actual events date>.log. These log files are
CHAPTER 12. AUDIT LOG

Figure 12.3: The link in the audit log to "Batch process: Map Reads to Reference" in the "Data in" column was clicked upon. A button to re-execute that job is available in the window that pops up.

retained for 31 days. When the creation of a new audit.log file is triggered, audit log files older than 31 days are checked for and deleted.

The audit log files can be found under the Server installation area under webapps/CLCServer/WEB-INF.

The audit log text files are tab delimited and have the following fields:

- Date and time
- Log level
- Operation: Login, Logout, Command queued, Command done, Command executing, Change server configuration, Server lifecycle; more may be added and existing may be changed or removed.
- Users
- IP Address
- Process name (when operation is one of the Command values) or description of server lifecycle (when operation is Server lifecycle)
- Process identifier - can be used to differentiate several processes of the same type.
- Status - can be used to identify whether the entry was successful or not, e.g. if a job execution failed it will be marked here. Any number other than 0 means failed.
Chapter 13

Server plugins

Download and install server plugins and server extensions

Plugins, including server extensions (commercial plugins), are installed by going to the Extensions (unità) tab in the web administrative interface of the single server, or the master node of a job node or grid nod setup, and opening the Download Plugins (_area (figure 13.1).

![Download Plugins](https://digitalinsights.qiagen.com/products-overview/plugins/)

Figure 13.1: Installing plugins and server extensions is done in the Download Plugins area under the Extensions tab.

If the machine has access to the external network, plugins can be both downloaded and installed via the CLC Server administrative interface. To do this, locate the plugin in the list under the Download Plugins (area and click on the Download and Install... button.

To download and install multiple plugins at once on a networked machine, check the "Select for download and install" box beside each relevant plugin, and then click on the Download and Install All... button.

If you are working on a machine without access to the external network, server plugin (.cpa) files can be downloaded from: https://digitalinsights.qiagen.com/products-overview/plugins/ and installed by browsing for the downloaded file and clicking on the Install from File... button.
The CLC Server must be restarted to complete the installation or removal of plugins and server extensions. All jobs still in the queue at the time the server is shut down will be dropped and would need to be resubmitted. To minimize the impact on users, the server can be put into Maintenance Mode. In brief: running in Maintenance Mode allows current jobs to run, but no new jobs to be submitted, and users cannot log in. The CLC Server can then be restarted when desired. Each time you install or remove a plugin, you will be offered the opportunity to enter Maintenance Mode. You will also be offered the option to restart the CLC Server. If you choose not to restart when prompted, you can restart later using the option under the Server maintenance (￼) tab.

For job node setups only:

- Once the master CLC Server is up and running normally, then restart each job node CLC Server so that the plugin is ready to run on each node. This is handled for you if you restart the server using the functionality under

  Management (￼) | Server maintenance (￼)

- In the web administrative interface on the master CLC Server, check that the plugin is enabled for each job node.

Installation and updating of plugins on connected job nodes requires that direct data transfer from client systems has been enabled, which is done by the CLC Server administrator, under the "External data" tab.

Grid workers will be re-deployed when a plugin is installed on the master server. Thus, no further action is needed to enable the newly installed plugin to be used on grid nodes.

Links to related documentation

- Maintenance Mode: section 10
- Restarting the server: section 2.7.1
- Plugins on job node setups: section 6.2.4
- Grid worker re-deployment: section 6.3

Managing installed server plugins and server extensions

Installed plugins are managed and can be uninstalled from under the Manage Plugins (￼) area (figure 13.2), under the Extensions (￼) tab.

As when installing plugins, the CLC Server must be restarted to complete the action. See above for links to further information.

The list of tools delivered with a server plugin can be seen by clicking on the Plugin contents link to expand that section. Workflows delivered with a server plugin are not shown in this listing.

Plugin compatibility with the server software

The version of plugins and server extensions installed must be compatible with the version of the CLC Server being run. A message is written under an installed plugin’s name if it is not compatible with the version of the CLC Server software running.
CHAPTER 13. SERVER PLUGINS

Figure 13.2: Managing installed plugins and server extensions is done in the Manage Plugins area under the Extensions tab. Clicking on Plugin contents opens a list of the tools delivered by the plugin.

When upgrading to a new major version of the CLC Server, all plugins will need to be updated. This means removing the old version and installing a new version.

Incompatibilities can also arise when updating to a new bug fix or minor feature release of the CLC Server. We recommend opening the Manage Plugins area after any server software upgrade to check for messages about the installed plugins.

Server extension licensing

Commercial plugins, known as server extensions, require a license to be installed in the CLC Server before analysis tools delivered by the extension can be used. The license only needs to be present on the master server on grid and job node setups\(^1\). If a license file is present, but it is valid only for an older version of the plugin, or it has expired, a warning will be shown.

Please refer to the appendix of the individual commercial plugin manuals for detailed information about licensing of that product.

---
\(^1\)Prior to CLC Genomics Server 11.0, module licenses were also required on nodes.
Chapter 14

External applications

Non-interactive command line applications and scripts can be integrated into the CLC environment by configuring them as *external applications*. Once configured and installed, external applications are available to launch via the graphical menu system of a CLC Workbench or via the CLC Server Command Line Tools. External applications can also be included in workflows, allowing complex, reproducible analyses involving CLC tools and other, non-CLC tools to be easily configured and launched by end users.

There are two types of external application:

- **Standard external applications**, where a command line application is run directly on the server system.


Figure 14.1 shows an overview of the actions and data flow that occur when a standard external application is launched on the CLC Server via a CLC Workbench. The data flow can be summarized as follows:

1. An end user specifies input data and sets values for parameters when launching the external application from a *CLC Workbench* or *CLC Server Command Line Tools*.

2. The *CLC Server* exports the input data from the *CLC Server* to a temporary file.

3. The *CLC Server* launches the underlying application (standard external applications) or the container containing the underlying application (containerized external applications), and provides the parameter values specified by the user and the input data from the temporary file. The underlying tool or container runs as a separate process to the *CLC Server*. That process is owned by the same user that owns the *CLC Server* process.

4. When the command line application has finished, results are handled as specified in the external application configuration. Usually this involves results being imported into the CLC

110
environment and saved to the location specified by the user when the external application was launched.

5. Results imported into the CLC environment are available for viewing and further analysis, for example using a CLC Workbench.

Temporary files are created outside the CLC environment during the execution of third party (non-CLC) tools and are deleted after the process completes.

![Figure 14.1: An overview of the flow of a standard external application when launched from a CLC Genomics Workbench.](image)

**Key information about external applications**

- The CLC Server software should be run by an unprivileged user. Like other CLC Server tasks, external applications processes are owned by the same logical user that owns the CLC Server process itself. If the system’s root user is running the CLC Server process, then tasks run via the External Applications functionality will also be executed by the root system user. This is usually undesirable.

- For a standard external application, the underlying tool must be available on all the systems where an external application can be run by the CLC Server.
• For containerized external applications, it is sufficient that the containerized execution environment is configured on all the systems where the external application can be run.

• An external application configuration must be saved before it is available for use via a CLC Workbench or CLC Server Command Line Tools. Configurations stored in the Drafts area are not made available for use.

• A folder called "External Applications" appears in the CLC Workbench Toolbox menu when a CLC Workbench is connected to a CLC Server with available external applications.

• Updates to existing external application configurations are registered in the CLC Workbench during a single login session. To discover new external applications from a client application, you must log out of and back into the CLC Server.

• The CLC Server administrator, or members of groups given the relevant permissions, configure external applications, including controlling which are made available for use. Granting permission to configure external applications to members of a non-administrative group is described in section 5.2.

• By default, all users have access to available external applications. The CLC Server administrator can also control which groups have access to each external application. See section 5.2.

14.1 External application configurations

Individual external applications and the containerized execution environment are configured under the **External applications** tab, under the **Extensions** area of the CLC Server web administrative interface (figure 14.2).

Figure 14.2: Individual external applications and the containerized execution environment are configured under the External applications tab.

For quick reference:

• Configuration of the containerized execution environment is described in section 14.1.1.

• Creating and editing external application configurations is described in section 14.2.

• Importing and exporting external application configurations is described in section 14.4.

• Including external applications in workflows is described in section 14.10.
• Launching external applications is described in section 14.11.
• Suggestions for troubleshooting external applications are provided in section 14.12.

14.1.1 Configuring the containerized execution environment

To run containers as external applications, the containerized execution environment must be enabled and configured. This is done under the External applications tab (figure 14.3). The full docker command executed when a containerized external application is launched combines the "docker" command with parameters configured here, followed by the parameters specified in individual external command configurations, as described in section 14.2.

Containerized external applications are only supported on CLC Server systems running on Linux. Currently, we support Docker containers.

Figure 14.3: The server’s container execution environment is configured under the External applications tab.

To enable the execution of containerized external applications, check the Enable containerized execution checkbox.

To configure the CLC Server for executing containers as external applications, expand the Containerized execution environment area (figure 14.3), and:

• Enter the full path to the Docker client in the top field.
  Note that the user running the CLC Server application must have permission to run the docker executable.

• Select the Shared working directory from the drop-down list of import/export directories.
  This directory will be bind mounted into the container launched and used to exchange data between the container and the CLC Server. This directory must therefore be shareable between the containers and the system that the CLC Server is running on.
  Configuring import/export directories for the CLC Server is described in section 7.1.

• Specify the arguments to be passed to the docker command when a containerized external application is launched on this server setup. Arguments specific to particular external applications are specified in the individual external application configurations.
– **Use default arguments** With this option selected, the ".v" parameter is passed to docker followed by the shared working directory on the host system, selected above, and a mount point within the container, separated by a colon. The initial part of any command executed for containerized external applications will take the form:

```
docker run -v <import-export-dir>:<mount-point-in-image>
```

For convenience, the mount point in the container is set to be the same as the shared working directory location on the host system. If you wish to specify a different mount point, select the "Use custom arguments" option, described below, instead.

– **Use custom arguments** All arguments specified here are passed to docker when a containerized external application is launched. This list of arguments must include the ".v" option followed by the path to the shared working directory selected above and a mount point within the container, separated by a colon. Any environment variables specified here are considered **system variables** set for the running CLC Server.

**Note:** Environment variables can also be specified for individual external applications, as described in section 14.2.6. These are system variables set on the external application process started by the CLC Server.

### 14.2 Configuring external applications

External applications are created and configured under the **External applications** tab, under the **Extensions** area of the CLC Server web administrative interface.

Creating and editing external applications is described in section 14.2.1. Existing configurations can also be exported and imported. This is described in section 14.4.

**Note:** Immediately after creating and saving an external application configuration, that external application will be made available for use by client software by default. To keep the external application hidden, drag it to the Drafts area.

#### 14.2.1 Editing external applications

Creating and editing external applications is done in a dedicated editor (figure 14.4). To open this editor, log into the CLC Server web administrative interface, open the **External Applications** tab, expand the "External applications configuration" area, and then either double click on the name of an existing external application or click on the **New configuration...** button.

An overview of the type of information configured under each of the editor tabs is provided below, along with links to more detailed information:

**External command:** The name of an external application, as seen by end users, the type of the external application (standard or containerized), the command line including its parameters, further configuration of parameters with names within curly brackets (section 14.2.2).

**High-throughput sequencing import / Post-processing** Where relevant to the external application, configuration of NGS importer or other CLC post-processing tools that should act on the results of the command line application (section 14.2.3).
Stream handling  The handling of information sent to the standard out and standard error streams by the underlying application (section 14.2.4).

Failure strategy  Define the failure strategy for the external application (section 14.2.5).

Environment  Environment variables that should be present for the external application when it executes (section 14.2.6).

End user interface  Settings affecting how an external application is displayed in a CLC Workbench client (section 14.2.7).

Management  Settings relating to the status of the external application (draft or installed) and other management tasks (section 14.2.8).

14.2.2 External command

The sections under the **External command** editor tab (figure 14.4) are:

**External application name**  The name seen in the CLC Workbench Toolbox menu and given to the corresponding workflow element for this external application. This name is also used as the basis of the name to use to launch the external application using the CLC Server Command Line Tools.

**Command line**  The command run when the external application is launched. The information to provide differs for standard external applications and containerized external applications, and is described in more detail below for each case.

Parameters values that should be substituted at run time are written within \{curly brackets\}. This includes parameters that should be configurable by the end user. Other parameters and values are written as normal in this field.
General configuration  Parameter values specified in *curly brackets* in the Command line field will have a corresponding entry in the General configuration area. There, these values are configured, including specifying their type, and for some types, configuring the values to use or to be offered to end users to select from. The description of a parameter can be configured by clicking the tooltip icon (_tooltip_ icon next to the parameter. The description will be displayed in clients, for example as a tooltip when running the external application from the CLC Workbench, or in the help listed for this application via the CLC Server Command Line Tools. The parameter value types are described in detail further below.

External application type  External applications can be "Standard (non-containerized)" or "Containerized: Docker". Standard external applications are executed directly on a server system. Containerized external applications are run from within containers.

To run containerized external applications, the containerized execution environment must be enabled and configured, as described in section 14.1.1.

Command lines for standard external applications  The Command line field should contain the path to the application and all parameters to be passed to that application. For illustration, a simple example of the cp (copy) command with 2 positional parameters is shown in figure 14.4.

Command lines for containerized external applications  The Command line field should contain only the parameters to send to docker that are not already configured for the containerized execution environment, described in section 14.1.1. These will usually be aspects of the command specific to running the individual external application.

For example, for a container with a command that takes one argument, the information written in this field could take the form:

```
<image-identifier> <command-to-run-from-image> <parameter>
```

The full docker command executed when an external application is launched combines the information configured for the containerized execution environment with the information provided in the Command line field. So, for example, if the default configuration settings for the containerized execution environment were used, the full docker command run when this external application is launched would take the form:

```
docker run -v <import-export-dir>:<mount-point-in-image> \
    <image-identifier> <command-to-run-from-image> <parameter>
```

In figure 14.5, the command for a containerized external application running the alignment program MAFFT is shown as an example. That example is described in more detail in section 14.9.

Parameter value types  Details of parameter value types are provided below\(^1\). Particularly important types for external application configurations are Data from CLC Location, which is the usual choice for parameters

\(^1\)The parameter type Boolean compound was made legacy in version 21.0 and was retired for version 23.0. Please update configurations that still rely on this parameter type. For the moment, existing external applications containing this parameter type can still be executed and edited.
specifying CLC data to be analyzed by the command line application, and **Output file from CL**, which is the usual choice for specifying results generated by the underlying application.

To see a brief description while working in the web administrative interface, select a value type, and then hover the mouse cursor over it.

- **Inputs**
  - **Data from CLC Location** - The end user will be prompted for one or more CLC data elements on the **CLC Server**. In the General configuration area, the appropriate exporter should be selected, so that the format of the data being provided to the command line application will be as needed. Each exporter can be configured further by clicking on the **Edit parameters** button (figure 14.4). A window then appears with a list of configurable parameters, (figure 14.6).
  
  - **External file** - The end user will be prompted for one or more input files stored in an Import/Export area or an AWS S3 location that is accessible via configurations in the **CLC Server**. Files can be configured so they are pre-selected for the end user, but the end user can deselect pre-configured files when launching the external application.
  
  - **Local file** - The end user can select a file from their local system. The full path to the file transferred into the temporary job location will be provided as the parameter value at run time. The **CLC Server** must be configured to allow direct data transfer from client systems for this option to be usable. If it is not, the parameter will not be configurable by the end user and they will see a message saying server upload is disabled when they try to launch the external application.

- **Outputs**
  - **Output file from CL** The type to select when specifying results generated by the underlying application. Further details about this option are provided below.
• General parameters

  - **Text** - The end user can provide text that will be substituted into the command at runtime. A default value can be configured.
  - **Integer** - The end user can provide a whole number that will be substituted into the command at runtime. A default value can be configured. If no value is set, then 0 is the default used.
  - **Double** - The end user can provide a number that will be substituted into the command at runtime. A default value can be configured. If no value is set, then 0 is the default used.
  - **CSV enum** - A drop down list is presented to a Workbench end user, from which they can choose a desired option. The corresponding value will be substituted into the command at runtime. To configure this parameter type, enter a comma delimited list of the values to be substituted at runtime into the first box, and a comma delimited list of corresponding labels to display to end users in the second box. Each entry in a given list should be unique and the two lists should be of equal length. For an example, please see section 14.6 on setting up Velvet as an external application.
  - **Boolean text** - A checkbox is shown in the Workbench wizard interface. If the user checks the box, the given text will be substituted into the command at runtime. If the box is unchecked, this means that no value will be substituted.

• Settings not visible to users

  - **Context substitute** - This will be substituted at runtime by the value selected: The options available are:
    * **CPU limit max cores** - The core limit defined for the server that executes the command will be substituted.
    * **Name of user** - The name of the user who launched the external application will be substituted.
  - **Included script** - A script provided as a value for this parameter type becomes accessible to the external process at runtime. This enables integration scripts or extensive parameter files to be included in the External Application and injected into the execution context, rather than being an external dependency. For containerized External Applications this may be the injected integration that enables the direct use of a public available container, avoiding the need to create an proprietary derived version of the container containing your own integration script. Refer to section 14.9 for an example.

A very simple configuration illustrating parameter configuration is shown in figure 14.4 for the cp command. In the **General configuration** area, the **Sequences to copy** parameter is set to **Data from CLC Location** meaning that the end user will specify the data to be copied from a CLC File Location. That data will be exported to a fasta format file. The **Copied sequences** parameter is set to type **Output file from CL**, indicating that this is the output from the command, and the standard fasta importer was selected for importing the results into the **CLC Server**.
Output file from CL

The Output file from CL option is used for parameters that define an output of the command line application. Once selected, a drop down list appears with options for how the output should be handled:

- **Where results should not be imported into the CLC Server**, choose the option *No standard import or map to high throughput sequencing importer*.

- **To import results into the CLC Server using a high throughput sequencing (NGS) importer**, choose the option *No standard import or map to high throughput sequencing importer* and then configure the importer to use under the High-throughput sequencing import / Post processing tab, described in section 14.2.3.

- **To import the results using a standard importer**, choose the importer to use from the drop down list presented. If the import type **Automatic** is selected, the importer used is determined by the filename suffix in combination with a check of the format of the elements in the file. If the file type is not recognized, it will be imported as an external file. A list of file formats, including the expected filename suffix for each format, can be found in the appendix of the CLC Genomics Workbench manual: Read more about search here: [http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Local_search.html](http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Local_search.html).

The third, empty field can be used to enter the name of the file the external process is expected to produce. If left blank, the base name of the file produced by the command line tool will be used as the base name for the data element imported into the CLC Server. Specifying a default filename in the third field, including the relevant suffix (e.g. .fasta, .xlsx), is recommended.

When Output file from CL is selected for at least one value, the end user will need to provide a location on the CLC Server, or a cloud destination if in a cloud context, to store results. This will be the case even if the output of the external file will not be imported, as log files will still be written to the location selected.

Setting default values and locking parameters

When configuring exports and imports, default settings can be configured, and you can control which values are editable when launching the external applications. Click on the Edit parameters button when available, and then click on the symbol of a lock image to open or lock that parameter (figure 14.6). Once unlocked, changes can be made. A parameter with an unlocked symbol beside it will be displayed to the end user and its value will be editable. Locked parameters are not shown when launching the external application and cannot be changed by end users.

A tip for exploring how many files an exporter will generate

A simple way to explore how many files an exporter will generate with a given configuration is to set up an external application using the echo command and a single parameter linked to the exporter of interest. Configure the "Standard out handling" option, selecting the "Plain text" option, described in section 14.2.4. The output from such an external application is a file, which
is re-imported into the CLC Server as a text file. This file contains the full paths to the files the exporter created.

If an exporter is configured in a way that will lead to multiple output files, then the full path to each output file will be substituted in the command at runtime. The external application itself must be able to handle the outputs generated.

14.2.3 High-throughput sequencing import / Post-processing

Data generated by the external tool can be imported using a high throughput sequencing importer or processed after import by a post processing tool on the CLC Server. To configure a high throughput sequencing importer or post processing tool:

- Choose the option Output file from CL for a parameter value in the General configuration area under the External command tab.

- Choose the option "No standard import or map to high throughput sequencing importer". Despite the name, this will also allow connections to be made to post processing tools.

- Open the High-throughput sequencing import / Post processing tab.

- Click on the Add new button.

- Select the high throughput sequencing import or post processing tool of interest from the list.

- Click on the button below the list called Edit and map parameters.... This button appears when a tool is selected.

A new window showing all the parameters for that tool will appear. See figure 14.7 for an example.

- Select the relevant external application parameter to be used as input to the tool by clicking on it in the list.
For high throughput sequencing tools, the input parameter is usually called "Selected files". For post processing tools, the name of the input field varies, but the word "(input)" is usually present by that parameter name.

Figure 14.7: Clicking on the "Edit and map parameters..." button opens a window for configuring the selected tool. Here, the parameter "Copied sequences" has been selected by clicking on it, thereby specifying that the file associated with that parameter should be used as input to the tool. All the parameters here have a locked symbol beside them, so none of them will be editable or shown to end users.

- Configure any of the other parameters that you wish to, for example:
  - Change the parameter values. To edit fields that are locked by default, click on the symbol of the lock image to open the lock, then make changes.
  - Unlock parameters that should be visible and editable by end users by clicking on the lock symbol.

After configuration changes are saved, text in the General Configuration panel will be updated, making it clear which tool has was selected for use with a given command parameter. For example, if the post processing tool: "Import SAM/BAM Mapping Files" was configured and the relevant output mapped to it, the text in the General configuration area for that output would show: "Linked with Import SAM/BAM Mapping Files" in the second field.

If you change your mind about linking to a high throughput sequencing importer or post processing tool from an external application parameter, you must remove the connection between the two. This would usually be done by deleting the relevant configuration in the High-throughput sequencing import / Post processing area.

14.2.4 Stream handling

Handling standard out and standard error streams for the external application is configured under the **Stream handling** tab, shown in figure 14.8.
How standard out and standard error should be handled is configured under the Stream handling tab.

The contents of standard out can be ignored or imported into the CLC Server using a standard importer. For applications where standard out contains the main result, choosing an appropriate importer would generally make sense. It can also be useful to import information sent to standard out for debugging purposes. In the File name field, the name of the raw file used to temporarily store the output from standard out must be specified.

By default, if information is sent to standard error by an application, the tool is stopped and the information is reported to the end user. The contents of standard error can instead be imported into the CLC Server by selecting the option Do not stop execution or show error dialogs. If the contents of standard error are imported, the File name must be specified in the same way as for standard out.

The names entered into the File name fields for standard out and standard error handling are visible to end-users, without a file extension, in the following places:

- This name is given to the data element imported into the CLC Server.
- This name is used as the label for the output channel of the workflow element corresponding to the external application.
- This name is the default name for output elements connected to the relevant workflow output channels.

### 14.2.5 Failure strategy

A failure strategy can be defined for each external application. This supports a fail fast strategy, where the external application will fail on a defined trigger, and write defined messages.

Under the Failure strategy tab of an external application, a process result, what should trigger its failure, and the message to post about that failure are configured (figure 14.9).

The types of results that can be included in a failure trigger configuration are:

- A result file produced by the native process, as configured in the External command tab.
- The std out process stream
- The std error process stream
- An exit code from the terminated process

Where the external process result selected is a result file, the failure trigger is always the non-existence of the result file. For the other three result types, the trigger is defined using Java regular expressions, which are matched against the process result. For std out and std error streams, the expression is matched against each line. In the case of exit codes, the expression is matched against a string representation of the exit code integer value.

Figure 14.9: Failure triggers for an external application are configured under the Failure strategy tab.

Other notes about failure strategies and troubleshooting

- When external applications fail, any results files produced despite the job failure will be posted. In such cases, the std out and std error files can contain valuable troubleshooting information.
- When an external application is included in a workflow, the final, substituted command line executed by the external application and the native process exit code are posted to the workflow log, as well as any errors contained in the server result (thus also all failures triggered).
- Audit log entries for external applications include the native process command line and the exit value.

14.2.6 Environment

Aspects of the environment within which the external application will be run are configured under the Environment tab.
Environmental variables

Environment variables that should be present for the external application when it executes can be created here, and default values set. In the example shown in figure 14.10, an environment variable named "HELLO" with value "hello world" would be present in the execution environment of the external application.

![Image](image_url)

Figure 14.10: Under the Environment tab, settings related to the environment an external application will be run on are configured.

Working directory

The Working directory setting specifies where temporary files should be created. This setting is only relevant to standard (non-containerized) external applications. For containerized external applications, the working directory is defined in the containerized execution environment, as described in section 14.1.1.

Default temp-dir Temporary files will be placed under the directory specified by the `java.io.tmpdir` setting for the system.

Shared temp-dir Temporary files will be placed under the directory you specify. The files will be accessible to all execution nodes and the master server without needing to be moved between machines.

To be available for use as a shared temp-dir, a directory must be:

- Already configured as an Import/export directory, as described in section 7.1.
- A shared directory, accessible by all machines where the external application will be executed.

Grid environments: For an external application to be executable on grid nodes, the Shared temp-dir option must be chosen for the working directory.

Job node setups: For an external applications to run on job nodes, either Working directory option is fine. However, if the directory specified by `java.io.tmpdir` is not an area shared between
machines, and it usually is not, choosing Default temp-dir means that files will need to be moved between the master and job nodes.

**Execute as master process**

When the **Execute as master process** option is enabled, the command line application will be executed on the master machine of a job node or grid setup. Export, import and post-processing steps are still run on the execution environment selected by the user when launching the external application. This setting has no effect for single server setups, where all execution happens on the same system.

With this option unselected, all stages of the external application are run on the execution environment selected when launching the external application. This is the default situation.

**Execute as master process** is not recommended for use with memory or cpu intensive tasks as the command line application will be launched on the master system without consideration of how busy that system is, or what processing capabilities it has.

**For grid setups:** Whether or not the **Execute as master process** option is enabled, export, import and post-processing steps will be run on a grid node if a grid execution environment is selected when launching an external application. This is why the Working directory option must be set to **Shared temp-dir** when running an external application on a grid setup.

**Add parameter history to imported objects**

When checked, information is added to the history of CLC data elements created as a result of an external application.

The history information added includes parameter values supplied for the native command by the external application author, as well as values supplied by a user when launching the external application. For relevant elements, it includes the final, substituted command line (figure 14.12).

![Figure 14.11: Under the Environment tab, settings related to the environment an external application will be run on are configured.](image)
14.2.7 End user interface

Settings under the End user interface tab (figure 14.13) affect the external application presentation in a CLC Workbench client:

- **Parameters per wizard step**: The number of parameters to present in a given wizard step when the application is launched. The default value is 4. With this value, up to 4 parameters will appear on each wizard page that the user steps through when launching the job.

- **Toolbox subfolder name**: The name of a subfolder under the CLC Workbench Toolbox | External Applications folder, where the external application should be listed. If a subfolder of the specified name does not already exist, it will be created. When this field is empty, the application will be listed directly under Toolbox | External Applications. Subfolder organization is also reflected under the main External applications tab, where the configurations are grouped according to subfolder. External applications without a subfolder specified are placed in the Default folder area. Configurations in draft status are always placed in the Drafts area, no matter what the subfolder designation is.
The text entered is case sensitive; "foldername" and "Foldername" refer to 2 different subfolders.

The creation of subfolders of subfolders is not supported. If you enter text like "myfolder/subfolder", a single folder of exactly that name would be listed under the External Applications folder in the Workbench Toolbox.

Dragging external application configurations between *Installation locations* under the External applications tab has the effect of updating the subfolder configuration. The new location in the Toolbox will be seen by end users when they next log into the CLC Server from the CLC Workbench.

**Other configuration aspects that affect CLC Workbench end user experience**

The following aspects of an external application configuration are seen when using a CLC Workbench client:

- Under the External command tab, the external application name is the name displayed under the Toolbox menu and is used as the name of the corresponding workflow element.

- Under the External command tab, the name entered for parameter values between *curly brackets* is used as:
  - A parameter label in the launch wizard
  - A parameter name for the CLC Server Command Line Tools
  - The name of the output channel of the external application workflow element.

- Under the Stream handling tab, the file names entered for standard out and standard error handling are used as the names of output channels for the external application workflow element.

- Where a default value is specified, that value is generally displayed in the launch wizard.

- The parameters listed in exporter configuration windows are *all* those available for the exporter. By default, these are all locked, and thus are not shown to end users when the external application is launched. Unlocking some of these can sometimes make sense, but unlocking all of them can result in a confusing, and sometimes conflicting, set of options for end users.

### 14.2.8 Management

General information about the external application is presented under the **Management** tab, including the version, the user that last updated the configuration and the time of the last update. Where exporters and importers have been updated, as described in section 14.5, information about such "auto-updates" is also reported here.

The following activities can be done under this tab:

- **Move to Drafts** Choose this option if you do not want the external application to be available to CLC Server users to use.
The option **Move to Default folder** only affects new (as yet unsaved) configurations and those in the Drafts folder. External applications placed in the Default folder, or any folder except Drafts, will be available for **CLC Server** users to use.

Moving applications between folders can also be done using drag-and-drop in the main External application configurations area.

- **Delete from server...** Delete the external application configuration. This does not delete any underlying third party tools.

Deleting external application configurations can also be done by clicking on the **Delete configuration...** button at the top of the main External application configurations area.

- **Install copy on Save...** When the **Save...** button is pressed, any updates made to the original external application are saved, and a copy is saved to the same folder as the original. The copy is given a version number of 1.

![Figure 14.14](image.png)

**Figure 14.14:** Under the **Management** tab, information about an external application is provided and general management options are available.

### 14.3 Using consistent reference data in external applications

For external applications that include third party tools that use reference data, such as reference genomes, annotations, primer sets, etc., the same reference data should be used for both the third party and CLC processing steps. To achieve this, reference data must be in locations and formats that can be used by each relevant application. This usually means either importing reference data into a **CLC Server** or exporting it from a **CLC Server** and placing it where the third party application can access it when it is run.

**Reference data is imported** into a **CLC Server** using client software, either a CLC Workbench or the CLC Server Command Line Tools. Data to be imported should be copied into an Import/export directory first, from where it can be imported.

If you do not already have the reference data you need, the Reference Data Manager of the CLC Genomics Workbench can be used to download data from QIAGEN servers and some public repositories such as Ensembl to the **CLC Server**. Further details about this can be found in section 3.2.2.

**Reference data is exported** from a **CLC Server** using standard export functionality of the client software, where the relevant data elements are selected in the Navigation Area of a CLC Workbench, or specified using CLC URLs when using the **CLC Server Command Line Tools**. Exported data is put into an import/export directory, from where it can be moved to the location of your choice.

Import/export directories are configured under the Main tab of the server web administrative interface, as described in section 7.1.
Data import and export are described in the client software manuals: https://digitalinsights.qiagen.com/technical-support/manuals/.

14.4 Import and export of external application configurations

External application configurations can be exported from or imported into a CLC Server, facilitating backup and exchange, as well as use by a CLC Workbench running jobs on the cloud.

**Exporting external application configurations**

To export external applications configurations, click on the Export configuration... button. Then check the "Select all" box, or select one or more of the external applications listed, and click on one of the Export... buttons at the bottom (figure 14.15). This will produce a single XML file containing the configurations of the selected external applications.

You will need to have a valid AWS Connection to be able to select an AWS S3 location to save the configuration file to. See section 7.3. Further information on exporting external applications intended for use on the cloud, is provided in the CLC Cloud Module manual: http://resources.qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Using_external_applications_on_cloud_via_CLC_Workbench.html.

For standard external applications, this configuration file is all that is needed. For containerized external applications, the containerized execution environment settings are not included in this XML file, so if sharing with another CLC Server setup, those settings will still need to be configured.

Figure 14.15: One ore more external applications configurations can be exported to an XML file in the location of your choice.
Importing external application configurations

External application configuration files can be imported to a CLC Server by clicking on the **Import configuration...** button. The following options are available in the dialog that opens (figure 14.16):.

- **Load local file...** Import from your local file system.
- **Load server file...** Import from a CLC Server import/export directory.
- **Load cloud file...** Import from AWS S3.

If the selected file contains configurations for external applications already installed on your CLC Server, another dialog will open where you can specify the configurations to overwrite.

![Figure 14.16: External applications configurations can be imported from local folders, CLC Server import/export directories and cloud locations](image)


### 14.5 Updating external application configurations

When an external application is run, data is exported from the CLC Server and results are imported into the CLC Server. If the versions of the exporters and importers differ between the external application configuration and the CLC Server, the external application configuration will need to be updated. This can happen when upgrading to a new server version or when importing external applications configuration files generated on an older server.

When updates to exporters or importers are needed for published external applications, a button labeled **Updates are needed** appears in the "External application configurations" area.
Red exclamation marks are also displayed next to the individual external applications that need to be updated, and within the external application editor, beside the name of the tab where the update is needed, usually the "High throughput importers/Post processing" tab, and also under the Management tab.

Configurations can be individually updated. Alternatively, click on the **Updates are needed** button. This brings up the "Update External Applications" window, which contains a list of the configurations that need updating. Clicking on the **Update** button in that window will update all the configurations that can be auto-updated. Mouse over the name of individual external applications to see information about what needs to be updated.

Any configuration that cannot be updated from the "Update External Applications" window has to be updated manually. An example where a configuration cannot be auto-updated is where a tool has been replaced. These are relatively rare events.

**Note:** The need for updates is not reported for external application in the Drafts area and these external applications are not listed in the "Update External Applications" window.

Under the Management tab, the most recent auto-update carried out on an external application configuration is recorded separately to the latest update made manually. If there has been at least one auto-update carried out, clicking on the button labeled **View last auto-update changes** will show information about what was done (figure 14.18).
14.6 Example: Velvet (standard external application)

In this example, we configure Velvet [Zerbino and Birney, 2008], a popular de novo assembler for next-generation sequencing data, as a standard external application.

The Velvet package includes two programs that need to be run consecutively. The external applications system on the CLC Server is designed to call one program, so a wrapper script is needed to make the needed consecutive calls to the Velvet applications.

An example script as well as a configuration file are available in a zip file at https://resources.qiagenbioinformatics.com/external-applications/velvet-example.zip. These will be used to illustrate how this sort of application can be configured as an external application on a CLC Server.

14.6.1 Installing Velvet

To get started, you need to do the following:

- Install Velvet on the server computer. The program and installation information is available from https://github.com/dzerbino/velvet/tree/master. If you have job nodes, Velvet will need to be installed on all the nodes that will be configured to run it.

- Download the scripts and configuration files from https://resources.qiagenbioinformatics.com/external-applications/velvet-example.zip. These files have been created assuming that Velvet is installed in /usr/local/velvet. If it is installed elsewhere, please update the files with the correct path to the program on your server.

- Check to ensure execute permissions are set on the velvetg and velveth executable files in the Velvet installation directory. These must be executable by the user that owns the CLC Server process.

- Unzip the velvet-example.zip file and place the clcbio folder and its contents in the Velvet installation directory. This contains a script (velvet.sh) that links the two Velvet programs, velvetg and velveth, together. If the Velvet binaries are not in the folder /usr/local/velvet, you will need to edit the line that starts with exe= to include the correct path.
• Set the permissions on the velvet.sh script in the clcbio subfolder so that it can be executed by the user that owns the CLC Server process.

• Import the velvet.xml configuration file: Log into the server via the web administrative interface and go to the External applications tab. Expand the "External applications configuration" section and click on the Import Configuration... button.

After the configuration file has been imported, an external application named CLC bio Velvet will appear in the "Default folder" area for the external applications configurations. Click on the CLC bio Velvet name. An external application editor should open, with the External command tab open, as shown in figure 14.19.

Figure 14.19: The Velvet configuration after the external application configuration file has been imported.

• Update the path to the Velvet installation at the very top if necessary.

If you wish to execute this job on grid nodes, then a shared temp-dir must be specified in the Working directory section of the Execution area of the configuration. See section 14.2.6 for details.

If you see a small red exclamation point beside the external application name, then something is wrong and needs to be attended to. The specific area where there is a problem should also be identified by a red exclamation mark. This is seen, for example, when the versions of a High-throughput sequencing import or Post-processing step specified in the configuration file is different to the version on the CLC Server. This can occur when the configuration was set up on an older version of the CLC Server than the one running. Rectifying this is straight forward, and is described in section 14.5.
14.6.2 Running Velvet from the Workbench

To run Velvet, open a Workbench and connect to a server where the Velvet external application is installed. Then:

- Go to:
  Toolbox | External Applications (⿰) | CLC bio Velvet (⿰)

- Confirm where you wish to run the job.

- Select (⿰) the reads to assemble and configure the Velvet parameters, as shown in figure 14.20.

- Click Next.

- Specify where to save the results.

- Click Finish.

The process that follows has four steps:

1. The sequencing reads are exported by the server to a fasta format file. This is a temporary file that will be deleted when the process is done.

2. The velvet script is executed using the fasta file and the user-specified parameters as input.

3. The resulting output file is imported into the save location specified in the save step of the Workbench dialog, and the user is notified that the process is done.

4. All temporary files are deleted.
14.6.3 Understanding the Velvet configuration

The Velvet configuration file is explained here as a specific example of all external application configuration files.

Going back to figure 14.19, there is a text field at the top. This is where the command expression is created, in this case:

```
/opt/local/velvet/clcbio/velvet.sh {hash size} {read type} {reads} {expected coverage} {contigs}
```

The first part is the path to the script. The following parts are parameters that are interpreted by the server when calling the script. Parameters to be interpreted are surrounded by curly brackets `{ }`. Note that each parameter entered in curly brackets gets an entry in the panel below the command line expression.

The first parameter, `hash size`, can be entered as a **Double** (which is a number that can take decimal values in computer parlance). The user provides a value when they launch Velvet. A default value is provided in the configuration (31).

The second parameter is `read type`, which has been configured as a **CSV enum** which means a list of possible values that the user can choose from. The first part of the configuration information consists of the parameters to be used when calling the velvet script ( `-short`, `-shortPaired`, `-long`, `-longPaired`), and the second part is the more human-readable representation that is to be shown in the Workbench (Short, Short Paired, Long, Long Paired).

The third parameter is `reads` which is used for the input data. When the **User-selected input data** option is chosen, a list of all the available export formats is presented. In this case, Velvet expects a fasta file. When a user starts Velvet from the Workbench, the server starts exporting the selected input data from clc format to a temporary fasta file before running the script.

The **expected coverage** parameter is similar to hash size.

The last parameter, `contigs`, represents the output file, as indicated by the choice of "Output file from CL" as the type. Here, you specify how the output from velvet should be handled. A list of standard import formats is provided, as well as the option not to import using those tools. Choosing not to import using those tools means that you can choose a high throughput importer instead from the section High-throughput sequencing import/ Post-processing section.

For this example, Do not import is the action set for the contigs parameter. Then, below, in the High-throughput sequencing import/ Post-processing section, the Fasta High-throughput Sequencing Import tool has been selected. Thus, when the results from velvet are ready, they are imported into the CLC Server using that tool and saved where the user indicates when they run the job.

14.7 Example: Bowtie (standard external application)

In this example, we show how to integrate Bowtie [Langmead et al., 2009], a popular tool for mapping short sequencing reads to a reference sequence, as a standard external application. Here, two post processing tools will be configured, allowing us to import more than one output generated by bowtie into the CLC Server.
Importing the configuration file provided as an example, and following the instructions in this section, leads to three tools being made available to users logged into the CLC Server via their Workbench or the Command Line Tools: CLC bio Bowtie Build Index, CLC bio Bowtie List Indices and CLC bio Bowtie Map.

14.7.1 Installing Bowtie

To get started:

- Install Bowtie from http://bowtie-bio.sourceforge.net/index.shtml. We assume that Bowtie is installed in /usr/local/bowtie but you can just update the paths if it is placed elsewhere.

- Download the scripts and configuration files from https://resources.qiagenbioinformatics.com/external-applications/bowtie-pp2.zip

- Place the clcbio folder and contents in the Bowtie installation directory. This folder contains the scripts used to wrap the Bowtie functionality. Those wrapper scripts are what is then configured via the External Applications folder.

- Make sure execute permissions are set on these wrapper scripts and on the executable files located in the Bowtie installation directory. The user that will execute these files will be the user that started the CLC Server process.

- Import the bowtie-pp2.xml configuration file: Log into the server via the web administrative interface and go to the External applications (tab). Expand the "External applications configuration" section and click on the Import Configuration... button.

The bowtie-pp2.xml file contains configurations for three tools associated with Bowtie: CLC Bowtie build index, CLC Bowtie list indices and Bowtie Map. If you already have a set of indices you wish to use and the location of these is known to the system via the BOWTIE_INDEXES, then you can just use the Bowtie Map tool via the Workbench and specify the index to use by name.

Otherwise, you can build the index to use using the CLC Bowtie build index tool. Here, unless you edit the wrapper scripts in the files you download from CLC bio, the indices will be written to the directory indicated by the BOWTIE_INDEXES environmental variable. If you have not specified anything for this, indices will likely be written into the folder called indexes in the installation area of Bowtie. Please ensure that your users have appropriate write access to the area indices should be written to.

From ftp://ftp.cbcb.umd.edu/pub/data/bowtie_indexes/ you can download pre-built index files of many model organisms. Download the index files relevant for you and extract them into the indexes folder in the Bowtie installation directory.

When configuring Bowtie to run as an external application on master-node setups, the Environment configuration will need to be edited. See 14.7.3 for details. In addition, Bowtie indices will have to be placed somewhere accessible to all nodes. One option could be areas configured as Import/Export areas on the CLC Server.

The rest of this section focuses on understanding the integration of the Bowtie Map tool in particular.
14.7.2 Understanding the Bowtie Map configuration

After the `bowtie-pp2.xml` configuration file has been imported, click the CLC bio Bowtie Map name to see the configuration (figure 14.21).

![Figure 14.21: The External command tab of the CLC bio Bowtie Map configuration is visible in the external application editor.](image)

From an end-user perspective, when the configuration on the CLC Server is complete, they will be able to launch the CLC bio Bowtie Map tool via their Workbench Toolbox. A wizard will appear, within which they will select the sequencing reads to be mapped, identify the pre-built index file of the reference sequence to use and set a few parameters. The bowtie executable will then be executed on the server system and the results generated will be imported into the CLC Server using post processing tools. The sam mapping file is imported using the Import SAM/BAM Mapping Files tool. A fasta file of sequences mapping to multiple locations is imported using the Fasta High-Throughput Sequencing Import tool.

Below, we step through the General configuration panel and then explain the configuration of the post processing tools that handle the outputs from the bowtie analysis.

**General configuration panel**

Each of the parameters (items within curly brackets) written into the "Command line" box is presented as an item in the General configuration panel. There, we define the type of information each parameter expects or represents and default values, where relevant.

To understand how these parameters relate to the information that will be passed to the native
bowtie executable, please refer to the bowtie_map.sh script in the clcbio folder that should now be in place in the bowtie distribution folder.

Stepping through the parameters in the order they appear in the Command line area of the configuration, and thus the order they appear in the General config panel:

- The **reads** parameter refers to the data that will be provided to bowtie to map. The **User-selected input data** option means the user will be able to select data in a CLC File System Location. This data will be exported from there, such that the bowtie tool can use it. The second element in this line specifies the format the data should be exported in. This is set to **FASTA (.fa/.fsa/.fasta)** as this is the format the bowtie tool expects sequencing read data.

- The **index** parameter is expecting the name of a bowtie index. Specifying the type **Text** for this parameter means a user will see a box in the Workbench Wizard that they can type text into. Here, a default name, “coli” has been specified, which can be changed by a user launching CLC bio Bowtie Map. When setting up a tool like this, it would be simpler for users, and much less subject to error, if the type **CSV enum** were selected, and a specified set of indices were listed. Then, a drop down list of options would be provided to the user in a Workbench Wizard, when launching the external application, rather than relying on users typing in the correct names of available bowtie indices.

- The **sam file** parameter refers to the sam mapping file that bowtie will generate as one of its results file. Thus, the type is set to **Output file from CL**. Import of sam files into the CLC Server involves a tool that requires user input. Thus, a post processing tool is configured. This can be seen immediately by the text in the second drop-down box: “Linked with Import SAM/BAM Mapping Files”. If a parameter with type **Output file from CL** is not mapped to a parameter of a post processing step, the text displayed is “Do not standard import / map to high-throughput sequencing importer”. Mapping of outputs to post processing tools is described in more detail below.

The last entry in the configuration of the **sam file** parameter is the name of the sam output file that bowtie should generate. Here it is set to **sam_output**. This file name is used by the bowtie command. The Workbench or Command Line Tools user never sees it.

- The **max number of multimatches** parameter allows a user to select the maximum number of locations a read can map equally well to for it to be included in the mapping. The type is set to **CSV enum**, which means a user will be able to select a value from a drop down list of the 3 values listed in the last field (2,3,4). The first value will be the default. The values in the middle field are those passed to the bowtie wrapper script and then onto bowtie. So, for example, if “2,3,4” were entered in the middle field, and “two, three, four” in the last field, a user could select the option "two", and bowtie would be sent the value 2.

- The **multimatch filename** parameter refers to another output from bowtie, this one containing fasta formatted reads that match to multiple locations of the reference equally well. Since it is a result file, the type is set to **Output file from CL**. We have decided to use a post processing tool to bring the results back into the CLC Server, the Fasta High-Throughput Sequencing Import tool.
• The max number of mismatches parameter allows a user to select the maximum number of mismatches to be allowed between a read and the reference in order for a read to be considered as matching the reference at that location. The type is set to CSV enum, and is presented to a user in the same way as the max number of multimatches parameter described above.

• The report all matches option is one that can be turned on or off. Thus it is set to type Boolean text. A user will be presented with a checkbox they can select or deselect in the Workbench Wizard. The value in the text field, here "-a", is the one bowtie will be passed if the user selects the checkbox. If the user does not select the checkbox, this parameter will not be sent to bowtie.

Post processing - importing the results from Bowtie

If you expand click on the High-throughput sequencing import /Post-processing link below the General configuration area, you will see that there are two post-processing tools selected: the Import SAM/BAM Mapping Files tool and the Fasta High-Throughput Sequencing Import tool.

In each case, clicking on the Edit and map parameters button below it will bring up the configuration window for that tool. Here, several types of configuration can be carried out.

1. Mapping of outputs of the external application to inputs of the post processing tool.

2. Locking or unlocking of parameters, determining which parameters users can alter when launching the tool via the Workbench or Command Line Tools.

3. Setting default values for parameters of the external application.

Here, we step through the configuration of the Import SAM/BAM Mapping Files tool. The configuration of the Fasta High-Throughput Sequencing Import is similar.

The parameters in this configuration window are the Import SAM/BAM Mapping Files tool parameters, just as would be offered when that tool is launched directly in a CLC Workbench.

A locked lock symbol by a parameter means that the user will not be given access to this option when launching the tool. Default settings for lock parameters are used. The locked parameters shown in figure 14.22 indicate that a track will be output rather than a stand-alone read mapping, unmapped reads will be saved, references will not be downloaded from an external source and, had they been, downloaded references will not be saved. Quality scores and sequence names will be kept (not discarded).

By contrast, the References parameter is unlocked. When using the Import SAM/BAM Mapping Files tool, users need to specify where the relevant reference sequences are. Thus, this option should be made available for users to configure when the tool is being launched.

The input to the Import SAM/BAM Mapping Files also needs to be defined. This is done by mapping the relevant output from the bowtie command to the input parameter for the Import SAM/BAM Mapping Files tool. The output from bowtie is defined by the "sam file" parameter, and the relevant input parameter in the import tool is "Selected files". A drop down list of potentially relevant parameters appears to the left of the "Selected files" parameter. In our example, this has already been mapped to the "sam file" parameter of the command, as shown in figure 14.22.
CHAPTER 14. EXTERNAL APPLICATIONS

Figure 14.22: Configuration of Import SAM/BAM Mapping Files for import of a sam file after mapping using Bowtie.

Note: The drop-down lists of possibly relevant parameters provided in the post processing tool configuration window are populated based only on the types of parameters (in the General config pane). Any parameters of a type that could be relevant are presented. This means that some parameters appearing in these lists may not make sense contextually.

14.7.3 Setting the path for temporary data

The Environment handling shown in figure 14.23 allows you to specify a folder for temporary data and add additional environment variables to be set when running the external application.

Figure 14.23: Where to save temporary data needs to be defined for Bowtie.

Post-processing steps need to access the results files of the external application. Thus, if you are running on a master-node setup, the directory you choose for these results files must be shared, that is, accessible to all nodes you plan to have as execution nodes for this external application. This is because different stages of your task could be run on different nodes. For example, the export process could run on a different node than the actual execution of the Bowtie script and the post processing. Thus, in a master-node setup, be it using grid or CLC execution nodes, having this shared temporary area eliminates the overhead of transferring the temporary files between nodes.
14.7.4 Tools for building index files

We have also included scripts and configurations for building index files using the external applications on CLC Server. This also includes the possibility of listing the index files available. To get these to work, please make sure the path to the Bowtie installation directory is correct. The Bowtie distribution itself also includes scripts to download index files of various organisms.

14.8 Example: Kraken2 (containerized external application)

In this section, a containerized external application for Kraken2 is described. This example illustrates the use of a publicly available Docker image in combination with an "Included script", as well as how reference databases, in this case, the pre-built Kraken 2/Bracken databases from https://benlangmead.github.io/aws-indexes/k2, can be accessed from an external application.

More information about the Kraken2 taxonomic sequence classifier can be found at https://github.com/DerrickWood/kraken2/wiki.

Using this external application, users select a single-end, nucleotide sequence list and a public Kraken2 database to use in the analysis. The outputs include a Kraken2 report, a file containing other classification information, sequences lists for classified and unclassified sequences, and a Docker log. See section 14.8.1 for information about extending this external application configuration to create one able to handle paired sequences.

All activity takes place in a Docker container. No local installation of scripts or reference databases is done for this example.

Defining the Kraken2 command line and configuring the parameters

A command line and general configuration of a Kraken2 external application are shown in figure 14.24.

The external application type is set to "Containerized: Docker". Thus, the information in the "Command line" field will be appended to the command specified in the Containerized execution environment settings for the CLC Server. The Docker image from GitHub staphb/kraken2:2.1.2-no-db is the Kraken2 2.1.1 (no db) image. Further details about this can be found at https://hub.docker.com/r/staphb/kraken2.

The parameters in curly brackets in the command line are substituted at run time, either with values specified by the user or values specified in the configuration. Here, this includes an "Included script", which contains commands run within the container, and parameters relating to the input data, results and Docker logging information.

Further details about parameters with values substituted at run time:

- **kraken2.sh** Configured as an "Included script". The script is executed in the Docker container.
  
  Script contents are entered by clicking on the Edit contents button in the General configuration area.
Key activities defined in this script include downloading and unpacking the relevant Kraken2 database, as specified by the user when launching the application, and running the Kraken2 program (figure 14.26). Other steps include reading in information provided by other variables in the command line, and ensuring that results are written to the location expected by the CLC Server so that the results can be successfully imported.

Note the lines starting with the copy command `cp` at the end of the script: it is here that the outputs from Kraken2 are copied to the location expected by the external application. We determined the relevant directory earlier (`OUTPUT_DIR='dirname "$OUT_REPORT"'`) and set up a variable per output file in the commands under `# Expected result files to be written`. Note that this means the filenames specified in the Kraken2 command (e.g. `k2-classified.txt`) are not relevant in the long term.

In this script, the database file is downloaded using `wget` and an https URL configured in the **Kraken 2 database** parameter, described below. Later in this section we describe other options that can be considered for using external files.

- **Kraken 2 database** Configured as a "CSV enum" parameter type. A comma separated list of public URLs for Kraken2 databases is entered in the first field, and a corresponding comma separate list of names for those databases, which is what is presented to users, is entered in the second field. The user specifies one of these databases when launching the application. How this looks in a CLC Workbench is shown in figure 14.27.

- **Sequences to analyze** The Kraken2 command in the `kraken2.sh` script expects a single
FASTA file. Thus, in the "Sequences to analyze" configuration, we specify that data will be selected from a CLC location, and exported to FASTA format.

By default, the FASTA exporter creates one file for each sequence list. This is alright if users will only ever select one sequence list as input. However, to account for users selecting multiple sequence lists for a single Kraken2 analysis, the export settings should be adjusted (figure 14.25).

Figure 14.25: The FASTA export option "Output as single file" has been enabled. If 2 or more sequence lists are selected as input, they will be exported to a single file, which is then used by the Kraken2 command.

Note: To run a Kraken2 analysis for each sequence list individually, create a workflow containing a Kraken2 external application element and then run that workflow in batch mode. Running workflows in batch mode is described at [http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Running_workflows_in_batch_mode.html](http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Running_workflows_in_batch_mode.html).

The remaining 4 parameters, described below, are configured with the type Output from CL. This is used for outputs generated by the application. How each output should be handled is part of the configuration. This can include configuring import, some types of post-processing as well as choosing not to do anything with that output. In this example, we choose to import each of the 4 outputs created by the kraken2 command line in the kraken2.sh script.

- **Kraken2 Report** By specifying "Plain Text (.txt/.text)", we indicate that the file should be imported as text. This will be viewable in a CLC Workbench, but there is no scope for interaction with the contents.

- **Classified sequences** and **Unclassified sequences** The handling is set to "Automatic", meaning the standard importer should be used, and it should automatically detect the relevant importer to use. Kraken2 outputs these sequences in FASTA format. The output file names specified in the configuration use the suffix .fa, which is one of the accepted suffixes for FASTA files. (See [http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Standard_import.html](http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Standard_import.html)).

- **Full classification list** The format of the file is explicitly configured to ensure that the tab delimited text output by Kraken2 is imported as a table. To import the file as a table, a header line is needed. The output from Kraken2 doesn’t include a header line, so this is added using a command in the kraken2.sh script. Once imported, standard functionality for working with tables can be used. (See [http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Working_with_tables.html](http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Working_with_tables.html))
Figure 14.26: A script is defined that will be run in the Docker container. This version includes the commands to download and unpack the Kraken2 database specified by the user when launching the application.

Some other options for working with reference database from an external application

In this example, a user specifies the database to use from a drop-down list, and that database is then obtained from a corresponding URL, as defined in a "CSV enum" type parameter. Some other methods that support the use of external files include:

- Place a copy of the database files of interest in your AWS S3 account or a CLC Server import/export directory. The **Kraken 2 database** parameter could then be configured as an "External file". If desired, the parameter can be configured so a particular file is pre-selected.
The CSV enum parameter type used for the Kraken 2 database parameter results in a drop-down list of options in the Workbench wizard for the application.

This option may be of particular interest if you are using custom databases.

If the external application will be run on the cloud, placing the file on AWS S3 is the more efficient option. However, you will pay for storage there. If you chose to store the database in a CLC Server import/export directory, then the database file would be uploaded to AWS every time the external application was run.

- Configure the parameter type as "Text". This creates a text field, where a user could enter the URL of their choice. This is less user-friendly then the configuration in the example and also the "External file" option described in the point above, but it does provide maximum flexibility.

- Use the same setup as the example, but obtain the files using a public S3 URL. For this, you would include installation of the AWS CLI in the included script, so you could copy the files from the public S3 bucket to your own bucket.

- Create your own Docker image containing the database(s) of interest. Here, for example, you could take the publicly available Docker image used in the example, and extend it with the contents of your choice and then refer to this new Docker image in the external application command line.

**Settings under the Stream handling tab**

Under the Stream handling tab, we define how information sent to standard out and standard error should be handled. Depending on the application, information in these streams can be useful for troubleshooting.

- **Standard out handling** Kraken2 reports only a list of the names of the output files to standard out, which is not of particular interest for users. Thus in this configuration, we chose not to save the information written to standard out.

- **Standard error handling** Docker reports its progress to standard error. As this information can be useful for troubleshooting, We specified that execution should not be stopped when information is written to this stream and provided the name of a file to write the information to, `Kraken2-docker-log.txt`. We also specified that the plain text importer be used to import the file (figure 14.28).

  The base name of this file appears in an output channel of the corresponding workflow element (figure 14.29).
Figure 14.28: The CSV enum parameter type used for the Kraken 2 database parameter results in a drop-down list of options in the Workbench wizard for the application.

Figure 14.29: A workflow containing the external application. In workflows, the outputs to collect can be specified. Here, all outputs except the unclassified sequences will be returned by the analysis.

Making the external application available for use

When this external application is saved, it becomes available to run on the CLC Server unless it is put in the Drafts folder.

To run external applications on the cloud, the CLC Cloud Module is needed. See http://resources.qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Using_external_applications_on_cloud_via_CLC_Workbench.html. The external applications need to be installed on the CLC Workbench that will be used for submitting jobs or for creating workflows that contain the external application. To do this, export the configuration(s) to an AWS S3 bucket accessible from the CLC Workbench. From the CLC Workbench, right-click on the configuration file in AWS S3 under the
Remote Files tab, and choose the option **Install External Applications**. The external application(s) in the configuration file will be installed and made available from the **External Applications Cloud** folder in the Toolbox.

To export to a cloud location, a valid AWS Connection must be configured on the **CLC Server**, as described in section 7.3. To install external applications from an AWS S3 location, a valid AWS Connection must be configured in the **CLC Workbench**, as described at [http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=AWS_Connections.html](http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=AWS_Connections.html).

Reminder: If you plan to run external applications on the cloud via the **CLC Server Command Line Tools**, they must be included in a workflow, and that workflow must be installed on the **CLC Server**.

Of note when creating workflows: options are usually locked by default. To unlock parameters in a workflow element, double click on the central part of the element, or right-click on it, and choose the option **Configure...**. Then check for the lock/unlock icon beside each setting. See figure 14.30.

![Figure 14.30](image)

**Figure 14.30:** *By default, the database parameter will be locked in the workflow element. Configure the workflow element to unlock this parameter to allow users to select a database when launching the workflow.*

**Results from the Kraken2 external application**

The outputs from Kraken2 and the standard error information written by Docker are available in CLC format when the application has finished. A log of the job is also available. If the external application was run on the cloud, a file called workflow-result.json will also be present among the results (figure 14.31).

These results will be in the location specified by the user when launching the application. If the job was run on a **CLC Server**, that will be in a **CLC Server** location. If the job was run on the cloud, the results will be in an AWS S3 location.

14.8.1 Extending the Kraken2 external application for paired data

Here, we expand upon the configuration described in section 14.8 to create a Kraken2 external application for analyzing paired sequence data.

The steps involved in updating the original configuration, which handled single end data, involve:

- Updating the external application command, changing settings where needed and adding any new parameters necessary.
- Linking outputs from Kraken2 to an NGS importer that can correctly import pairs of files containing paired end sequences into sequence lists in the CLC software.
- Editing the included script to updated the Kraken2 command so it handles paired data, and ensuring it handles all the inputs and outputs as needed.

Below, we describe these steps in the order listed, but the order they are carried out does not matter.

Updates to the external application command line and top level configuration

The command and configuration shown in figure 14.33 takes into account how Kraken2 handles paired sequence data and how CLC software handles paired sequence data.

Kraken2:

- Kraken2 expects paired data in 2 files, one containing the first member of each pair, and another containing the second member of each pair.
- Kraken2 returns paired sequence data in 2 files, one containing the first member of each pair, and another containing the second member of each pair.

CLC software:
• The FASTQ exporter can export paired sequence data to 2 files. (The FASTA importer, used in the single end example, cannot.)

• Each input parameter in the external application general command represents a single input that a CLC software user will be prompted for. Here, a sequence list selected by the user will be exported to 2 FASTQ format files, which are then passed to the included script. We must handle this situation, where a single parameter in the command is associated with more than one file.

• The Illumina high-throughput sequence importer can import pairs of files as paired sequence data. Information in the filenames is used to determine which file contains the first member of a pair and which contains the second member of a pair.

Using this knowledge, we can adjust the external application accordingly:

1. The Sequences to analyze parameter is configured to use the FASTQ exporter, to export paired sequence data to 2 files (figure 14.32).

2. The Sequences to analyze parameter is used to pass 2 files to the included script, so that parameter is put in quotation marks in the Command line field (figure 14.33).

3. For each set of paired sequences generated by Kraken2, there are 2 parameters in the external application command line, one for each file (figure 14.33).

   The parameter names for outputs from the third party tool are not important, as they are not seen by end users, however, for transparency, we reflected our assumption that the data provided will be in forward-reverse orientation in the parameter names, i.e. |Forward classified seqs|, Reverse classified seqs, Forward unclassified seqs, and Reverse unclassified seqs.

4. For the sequence output parameters, we supply filenames that match our desired handling of the FASTQ files by the Illumina high-throughput sequencing importer: files with the first member of each pair have names ending with _R1.fastq, files with the second member of each pair have names ending with _R2.fastq.


5. We link the parameters for Kraken2 sequence outputs with the Illumina high-throughput sequencing importer. Details of how this is done are provided below.

**Linking to a High-throughput importer**

To handle the import of paired reads, a high-throughput sequence (NGS) importer is needed. Such importers require more configuration than the standard importers, i.e. those that can be directly selected in the "General configuration" area.

The steps to configure the import the 2 FASTQ files containing the classified sequences output by Kraken2 are described below. The same steps then need to be done to configure the import of the unclassified sequences.
Figure 14.32: Clicking on the "Edit parameters" button for the "Sequences to analyze" parameter opens up a configuration window for the FASTQ exporter. The default settings are shown here, with the "Export paired sequence lists in two files" option enabled.

1. Use the default setting: "No standard import or map to high-throughput sequencing importer" for the Forward classified seqs and Reverse classified seqs parameters.

2. Click on the High-throughput sequencing import / Post-processing tab (below the general configuration area) to open it.

3. Click on Add new and select an importer. Here, we select the Illumina importer (figure 14.34).

4. Click on Edit and map parameters....

5. Configure the importer.

   **Inputs** All parameters in the Command line that are configured as type "Output file from CL" are available for selection as inputs to the importer. Select the 2 for the classified sequences output: Forward classified seqs and Reverse classified seqs (figure 14.35).

   **Other importer settings** Keep the default settings for all the other parameters.

6. Click on the OK button to finish and save the importer configuration.

The Forward classified seqs and Reverse classified seqs parameters in the "General configuration" area will be updated with the link to the Illumina importer (figure 14.36). If you don’t see this immediately, save the external application configuration and re-open it.
CHAPTER 14. EXTERNAL APPLICATIONS

Figure 14.33: The external application command and top level configuration to support running Kraken2 with paired sequences as input and for handling the paired sequences Kraken2 generates as output.

Updates to the included script

The included script must be updated: Kraken2 command needs to be adjusted and changes made to the external application configuration need to be handled.

Changes to the Kraken2 command:

- `--paired` has been added to indicate that the sequences being provided are paired.
- `k2-classified#.txt` and `k2-unclassified#.txt` are given as values for the parameters `--classified-out` and `--unclassified-out` respectively. The `#` symbol is a convention used by Kraken2 when working with paired data.
Figure 14.34: A new import/post processing element has been created, and the Illumina high-throughput sequencing importer has been selected.


The other changes in the script are to handle the increased number of files being output by Kraken2 (figure 14.37).
Figure 14.35: The Illumina High-Throughput Sequencing Import configuration. Here, the parameters corresponding to the 2 FASTQ files containing classified sequences have been selected. values.

Figure 14.36: The information in the "General configuration" area is updated to reflect the link between output parameters and the Illumina importer.
Figure 14.37: The kraken2-paired.sh script, with the Kraken2 command updated to indicated that the data to analyze is paired, and with variables added to handle paired sequence data being passed to the script and out of the script.
14.9 Example: MAFFT (containerized external application)

In this section, a containerized external application for MAFFT, an alignment program for amino acid or nucleotide sequences, is described. This example illustrates the use of a publicly available Docker image in combination with an "Included script" and extension of the external application command line beyond a single line.

More information about MAFFT is available from [https://mafft.cbrc.jp/alignment/software/](https://mafft.cbrc.jp/alignment/software/).

Using the resulting external application, a CLC software user specifies sequences to align and a location to save the alignment to. They can also optionally configure MAFFT settings. The alignment is run in a Docker container, and the results are then imported back into the CLC software.

Defining the MAFFT command line and configuring the parameters

The command line and general configuration of a MAFFT external application are shown in figure 14.38.

![Configuration a containerized external application for MAFFT](image)

Figure 14.38: Configuration a containerized external application for MAFFT. A publicly available Ubuntu image is pulled and all subsequent information in the command is run in the container created.

The external application type is set to "Containerized: Docker". Thus, the information in the "Command line" field will be appended to the command specified in the Containerized execution environment settings for the CLC Server.

The parameters in curly brackets are substituted at run time with the values specified.
Getting the MAFF software into the external application

In this example, the steps to obtain and unpack MAFFT are run in the Docker container. There are other ways this can be done. Choices for how to get the MAFFT software running in the container include:

- Have in an included script the steps for downloading MAFFT from the software maintainer’s site and unpacking it (as done in this example).
- Download MAFFT to a location accessible to the external application (e.g. your S3 bucket if you are running the external application on the cloud), and access it from there, either:
  - By copying it into the container, using a command in the included script, or
  - By providing the path to the MAFFT binary as the value for an "External file" parameter type.

A key difference between these 2 options is that users will not see, nor be able to alter, information in an included script, while values in an External file parameter are visible and configurable when launching the external application. Note though that when the external application is included in a workflow, you can define whether or not this option should be configurable by users (figure 14.43).

If you are concerned about the availability of the software, or network stability of the site hosting it, then having a copy of the software in an area under your control may be preferable.

Extending the command line with /bin/bash -c

In this example, the MAFFT software (mafft.bat) is called using a line in the external application command line field (after the semi-colon), rather than being included within the script. This approach can make writing the script simpler, and may make it easier for external application authors, to keep track of the roles of the various parameters being passed to the bioinformatics application.

For comparison, see the Kraken 2 example in section 14.8 for a case where all commands are contained in an included script.

Both approaches are equally valid.

Further details about parameters with values substituted at run time:

- **install-and-run-mafft.sh** A script to be executed in the Docker container. The parameter type is set to "Included script" in the General configuration area.

  The "Included script" contents are entered by clicking on the Edit contents button in the General configuration area. The activities defined in the script include downloading and unpacking MAFFT (figure 14.39).

  Here, `{install-and-run-mafft.sh}` is specified as the argument to `/bin/bash -c` in the Command line field, allowing commands external to the included script to be run after those in the script have completed. Here, the `mafft.bat` command is appended this way.
• **MAFFT settings** A Text field where a user can configure parameters to pass to the MAFFT software. How this looks in a CLC Workbench is shown in figure 14.40. There is no validation of the information entered into a Text field when launching, so this approach assumes users know about the MAFFT program.

• **Sequences to align** MAFFT expects sequences in FASTA format. We thus specify that data will be selected from a CLC location, and exported to FASTA format.

• **Alignment** This defines the output from the external command. The output from MAFFT is in FASTA alignment format and we specify the name of the file to create. The external application then knows what importer to use to import the results into the CLC software.

See section 14.2.2 for more information about external application parameter types.

![Image of script](image.png)

Figure 14.39: A script is defined that will be run in the Docker container. It includes the steps needed to make MAFFT available to run in the container.

**Settings under the Stream handling tab**

Under the Stream handling tab, we define how information sent to standard out and standard error should be handled. The information in these streams can be useful for troubleshooting. The settings in this example are shown in figure 14.41.

• **Standard out handling** MAFFT reports its progress to standard out. This could be useful for troubleshooting, so we choose to collect this information and then import it using the plain text importer. The name of the file to write the information to is specified as "MAFFT-stdout.txt".
Figure 14.40: The wizard presented to Workbench users when they launch the MAFFT external application. They select the sequences they wish to align, and can, if they wish, edit the options being passed to MAFFT.

- **Standard error handling** Docker reports its progress to standard error. This information can be useful for troubleshooting. We thus indicate that execution should not be stopped when information is written to this stream. The name of the file to write the information to is specified as "Docker-log.tx", and we configured it to be imported using the plain text importer.

The base names of these files are used to name the output channels of the corresponding workflow element (figure 14.42).

**A general note about names**

External application names and the names of parameters that refer to inputs a user will select are presented to users and administrators in various places. The names of files containing standard out or standard error information are also visible. For example, here, the name, "MAFFT", will be the name used in the External Applications section of the server web administrative interface, as well as in the CLC Workbench Toolbox and the corresponding workflow element (figure 14.42). "Sequences to align", will be used in the CLC Workbench wizard (figure 14.40) and in the corresponding workflow element.

**Making the external application available for use**

When this external application is saved, it becomes available to run on the CLC Server unless it is put in the Drafts folder.

To run external applications on the cloud, the CLC Cloud Module is needed. See [http://resources.qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Using_external_applications_on_cloud_via_CLC_Workbench.html](http://resources.qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Using_external_applications_on_cloud_via_CLC_Workbench.html). The external applications need to be installed on the CLC Workbench that will be used for submitting jobs or for creating workflows that contain the external application. To do this, export the configuration(s) to an AWS S3 bucket accessible from the CLC Workbench. From the CLC Workbench, right-click on the configuration file in AWS S3 under the Remote Files tab, and choose the option **Install External Applications**. The external application(s) in the configuration file will be installed and made available from the **External Applications Cloud** folder in the Toolbox.
CHAPTER 14. EXTERNAL APPLICATIONS

Figure 14.41: Defining stream handling for the MAFFT containerized external application.

Figure 14.42: A workflow containing the external application. In workflows, the outputs to collect can be specified. Here, all the outputs are configured to be saved.

To export to a cloud location, a valid AWS Connection must be configured on the CLC Server, as described in section 7.3. To install external applications from an AWS S3 location, a valid AWS Connection must be configured in the CLC Workbench, as described at http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=AWS_Connections.html.

Reminder: If you plan to run external applications on the cloud via the CLC Server Command Line Tools, they must be included in a workflow, and that workflow must be installed on the CLC Server.

Of note when creating workflows: options are usually locked by default. To unlock parameters in a workflow element, double click on the central part of the element, or right-click on it, and...
choose the option **Configure**... Then check for the lock/unlock icon beside each setting. See figure 14.43.

Figure 14.43: *By default, parameters are locked in workflow elements, as shown here. To allow users to configure the "MAFFT settings" option when launching this workflow, the parameter must be unlocked.*

**Results from the MAFFT external application**

The alignment and the text files containing the standard out and standard error information are available in CLC format when the application has finished. A log of the job is also available. If the external application was run on the cloud, a file called workflow-result.json will also be present among the results (figure 14.44).

Figure 14.44: *Outputs from a MAFFT containerized external application after running it on AWS using functionality provided by the CLC Cloud module.*

These results will be in the location specified by the user when launching the application. If the job was run on a CLC Server, that will be in a CLC Server location. If the job was run on the cloud, the results will be in an AWS S3 location.

Remote_Files_tab.html. Interacting with files on AWS S3 via the CLC Server web interface is described in section 16.4.

14.10 External applications in workflows

Workflows are made up of workflow elements connected together. External application workflow elements are listed in the External Applications folder of the Add Elements dialog, available when editing workflows. Links to general documentation about workflows are provided at the bottom of this page.

Input channels, output channels and requirements for workflow elements

Workflow elements for analyses consist of input channels, the main body, and output channels. For an external application to be available for use in a workflow, it must be configured so that there is at least one output channel.

Output channels correspond to parameters configured as the type "Output file from CL", and to standard out and standard error streams configured for import. When an "Output file from CL" parameter in the external application is linked to a high-throughput sequencing importer or post-processing tool, the number of output channels depends on the number of outputs that tool produces. For the external application used in the workflow in figure 14.45, the configuration had a single "Output file from CL" parameter, which was linked to a high-throughput sequencing importer: Import SAM/BAM Mapping Files. Information from the standard out stream is also collected and imported. This configuration results in 4 output channels, 3 associated with the outputs from the SAM/BAM importer, and the fourth for the information from standard out. In the figure, Output elements have been connected to 2 of the output channels. Only these results will be saved when the workflow is run.

Input channels correspond to external application parameters configured as the type "Data from CLC Location". For example, in figure 14.45, there is a workflow element for an external application called "Generic Mapping External Application". The external application configuration included two "Data from CLC Location" parameters, one for reads and one for reference sequences. This is reflected by the 2 input channels. Here, a workflow Input element has been linked to the input channel for the reads. An Input element could also be linked to the channel for reference sequences, but this is optional and was not done here.

Controlling the options that can be configured when launching the workflow

Unlocked parameters are available for configuration when launching the workflow. Locked parameters are not presented for configuration when launching workflows.

This means that for linked high-throughput sequencing importers and post-processing tools, there are 2 layers of control: one layer in the external application configuration itself and one in the workflow element. Thus, if control over which parameters are available when launching the workflow should lie with the workflow author, these parameters need to be unlocked in the external application configuration. In both layers, most parameters are locked by default.

Controlling the names of options and outputs from workflows
The default names of workflow elements and parameters can be left as they are, but they can also be edited, which can make it easier for those launching the workflow to understand what information is being requested (figure 14.46). This is especially important when there are multiple instances of the same high-throughput sequencing importer or post-processing tool in the external application configuration, and parameters in these have been unlocked. If these parameter names are all left as the defaults, their names will be identical in the launch wizard, which could be confusing.

Links to workflow documentation


The following are pages within that chapter that contain information specifically referred to above:

- Configuring input and output elements, including renaming: http://resources.qiagenbioinformatics.com
CHAPTER 14. EXTERNAL APPLICATIONS

14.11 Running external applications

External applications can be executed using a CLC Workbench or using the CLC Server Command Line Tools.

14.11.1 Using a CLC Workbench to launch external applications

Launching external applications from the Toolbox

After connecting to a CLC Server with external applications configured and available to client systems, external applications can be executed by going to:

Toolbox | External Applications

External applications are listed as individual tools in the Toolbox, as shown in figure 14.47. Depending on how they were configured, they may be located within subfolders of the External Applications folder.

When an external application is launched, the dialog shown in figure 14.48 is displayed. Depending on your setup, there may be other execution environment to choose from. For example, grid presets will be shown if they have been configured as described in section 6.3.

Progress through wizard steps, configuring any settings necessary (figure 14.49).
Launching external applications as part of a workflow

External applications can be included in workflows, as described in section 14.10. Workflows stored in the Navigation Area can be launched from within the Workflow Editor and installed workflows can be launched from the Workbench Toolbox.
14.11.2 Using CLC Server Command Line Tools to launch external applications

The CLC Server Command Line Tools can be used to launch CLC tools, workflows and external applications, directly or within a workflow context, on a CLC Server. General information about using the CLC Server Command Line Tools can be found in the manual for that software: https://digitalinsights.qiagen.com/technical-support/manuals/.

When launching an external application, the CLC Server execution context will be used unless another execution environment is specified. For example, by adding the -G option, a grid preset can be specified.

Running a CLC Server Command Line Tools command with missing or invalid parameters results in messages about the problem being returned. For example, trying to invoke the copy external application described earlier in this chapter with no arguments yields the output below. The final line makes clear the problem: the -d and -in parameters need to be specified in the command for the job to be executed.

clcserver -S <HOSTNAME> -U <USER> -W <PASSWORD> -A copy
Message: Trying to log on to server
Message: Login successful
The following options are available through the command line and the types are as follows:

<table>
<thead>
<tr>
<th>Type</th>
<th>Valid input</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;Integer&gt;</td>
<td>A decimal number in the range [-2147483648;2147483647]</td>
</tr>
<tr>
<td>Example:</td>
<td>42</td>
</tr>
<tr>
<td>&lt;Boolean&gt;</td>
<td>The string true or false</td>
</tr>
<tr>
<td>Example:</td>
<td>true</td>
</tr>
<tr>
<td>&lt;String&gt;</td>
<td>To enclose all strings in ‘’ to avoid issues with the shell</td>
</tr>
<tr>
<td>Example:</td>
<td>'text=&quot;My text&quot;'</td>
</tr>
<tr>
<td>&lt;ClcFileUrl&gt;</td>
<td>A valid path to a file on the server or in the local file system</td>
</tr>
<tr>
<td>Example:</td>
<td>clc://serverfile/tmp/export</td>
</tr>
<tr>
<td>&lt;ClcObjectUrl&gt;</td>
<td>A valid path to a Clc object on the server or locally</td>
</tr>
<tr>
<td>Example:</td>
<td>clc://server/pstore1/Variant1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-A &lt;Command&gt;</td>
<td>Command currently set to ‘copy’</td>
</tr>
<tr>
<td>-C &lt;Integer&gt;</td>
<td>Specify column width of help output. (default: false)</td>
</tr>
<tr>
<td>-D &lt;Boolean&gt;</td>
<td>Enable debug mode. (default: false)</td>
</tr>
<tr>
<td>-G &lt;Grid preset names&gt;</td>
<td>Specify to execute on grid.</td>
</tr>
<tr>
<td>-H</td>
<td>Display general help.</td>
</tr>
<tr>
<td>-O &lt;File&gt;</td>
<td>Output file.</td>
</tr>
<tr>
<td>-P &lt;Integer&gt;</td>
<td>Server port number. (default: 7777)</td>
</tr>
<tr>
<td>-Q &lt;Boolean&gt;</td>
<td>Quiet mode. No progress output.</td>
</tr>
<tr>
<td>-s &lt;String&gt;</td>
<td>Server hostname or IP-address of the CLC Server.</td>
</tr>
<tr>
<td>-U &lt;String&gt;</td>
<td>Valid username for logging on to the CLC Server.</td>
</tr>
<tr>
<td>-v</td>
<td>Display version.</td>
</tr>
<tr>
<td>-W &lt;String&gt;</td>
<td>Clear text password or domain.</td>
</tr>
<tr>
<td>-d, --destination &lt;ClcServerObjectUrl&gt;</td>
<td>Destination for import from External Application</td>
</tr>
<tr>
<td>-in &lt;ClcServerObjectUrl&gt;</td>
<td>Model object(s) to be exported to FASTA (.fa/.fsa/.fasta)</td>
</tr>
</tbody>
</table>

Error: Missing required options: d, in
14.12 Troubleshooting external applications

There is no check for the consistency of the external application configuration when it is set up. If there are problems, these will first be seen by the end user, when the application is executed. For example, in a CLC Workbench, summary information will shown in a dialog. This may help to identify the issue. If this summary information does not help, try opening the Advanced tab, where an extended error message should be visible.

Below are some tips to aid with troubleshooting issues with external applications.

- Configure the external application to import standard out and/or standard error as text. This makes it possible to check error messages posted by the external application. See figure 14.50. For containerized external applications, messages from docker are posted to standard error, so collecting standard error for such applications is particularly recommended.

- If your external application was previously working and then stops working, check if the configuration was recently changed. Each time a change is made the version number of the external application is updated. The name of the user who made the most recent change in the listing under the External application configurations area, and also under the Management tab of the editor for each individual external application. Only users with admin privileges are able to edit external applications configurations.

- Check the third party application/container can be run:
  - For standard external applications:
    - Is the application being found? Check that the path to the application is complete and correct.
    - Is the application executable, and can it be executed by the user running the CLC Server process?
    - If there is a wrapper script calling the third party application, does it contain the correct path to the application? Is the wrapper script executable?
For containerized external applications:

* Does the user running the CLC Server have permission to run Docker? (Is that user a member of the docker group?)
* Is the reference to the container in the Command line field of the external application configuration valid?
* Is the CLC Server running on a Windows system? Containerized external applications are only supported for Linux-based setups.

- Check the import/export directory configurations, where relevant.

  - For standard external applications, the default temp dir is used by default for temporary files, but an import/export directory may be specified for this purpose instead, as described in section 14.2.6. If this has been done, check the import/export directory is available and is writable by the user running the CLC Server process.

  - For containerized external applications, an import/export directory is specified as the shared working directory in the containerized execution environment settings, as described in section 14.1.1. Besides being available and writable by the user running the CLC Server process, the selected directory must be shareable between the host system and containers.

- Check for conflicts in the naming of the external application.

  If your users will only access external applications via a CLC Workbench, then you do not have to worry about what name you choose when setting up the configuration. However, if they plan to use the CLC Server Command Line Tools, to interact with your CLC Server, then please ensure that you do not use the same name as any of the CLC Server internal commands. You can get a list of these by running the clcserver command, with your CLC Server details and no tool specified. I.e. a command of the form:

  `clcserver -S <host> -P <port> -U <username> -W <password or token>`
Chapter 15

CLC Genomics Cloud Access

The CLC Genomics Cloud is a combination of AWS resources and CLC software that supports running analyses submitted by a CLC Workbench or CLC Server on AWS EC2 instances and storing results on AWS S3.

This sections describes the configuration of cloud presets. For further details, please see https://digitalinsights.qiagen.com/plugins/clc-cloud-module/.

When submitting a job to a CLC Genomics Cloud setup via a CLC Server, the submitter specifies a cloud preset to use. Each preset is configured with the name of a single AWS Batch queue. When a given preset is selected, jobs are sent to the corresponding AWS Batch queue.

By default, all cloud presets are available for all users of the CLC Server. Access to each preset can be restricted to specified groups using options available under the Global permissions tab in the CLC Server web administrative interface.

Creating and editing CLC Genomics Cloud presets

To create or edit cloud presets, log into the CLC Server web administrative interface and go to:

Extensions ( ) | CLC Genomics Cloud ( )

Click on the Add Preset... button.

The preset name is what the user of client software specifies when launching their analysis. The AWS Batch queue is where jobs will be submitted to when this preset is selected. AWS Batch queue settings are configurable within AWS.

Prerequisites for CLC Genomics Cloud access

To submit jobs to a CLC Genomics Cloud setup via a CLC Server, the following must be in place:

- The Cloud Server Plugin must be installed on the CLC Server.
- Access to an AWS account with AWS Batch queues for a CLC Genomics Cloud setup in place.
- An AWS connection configured in the CLC Server using credentials giving access to that AWS account (section 7.3).
Chapter 16

Browsing and searching for data

16.1 Browsing CLC Server data locations

This section focuses on working with data through the CLC Server web interface. For working with the same data via the Navigation Area of a CLC Workbench connected to the CLC Server, please refer to the Workbench documentation: https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Navigation_Area.html

Browsing CLC data

A graphical view of CLC file system locations and their contents is available in the left side of the web interface. When an element is selected in the navigation area, information about the element is shown in the Info tab (figure 16.1).

The CLC URLs at the bottom are direct links to those elements, which can be shared with others with access to your system. The 2 URLs are just 2 different ways to point at the same element.

When the element selected is a log file, contents of the log are also displayed here.

Various actions are available, including moving data to the recycling bin, as described in chapter 17.

CLC data history

When an element is selected, its history can be seen in the History tab (figure 16.2).
Figure 16.1: CLC file locations are listed in the left side of the web interface. Information about a selected element or folder can be seen in the Info tab to the right.

Figure 16.2: The history of a selected element can be seen in the History tab to the right.

16.2 Searching CLC Server data locations

The search field under the CLC Server locations on the left of the web interface can be used to search in available CLC data locations. Enter terms within element or folder names, or a CLC URL. Searches are not case sensitive. Elements found are listed under the search field.

Click on an element in the results list and the location of that file in the main file listing will
be shown. If the Info tab is open, the first mouse-click on the search result causes information about that element to be displayed, and a second click reveals its location (figure 16.3).

Figure 16.3: A search was run for the term chrm, and 18 results were returned (only some are shown in this image). Clicking on one of the results in the list at the bottom revealed its location in the browser area at the top.

Searching with terms in names relies on search indexes. The relevant index is updated each time you create a new element or update an existing element. If an index gets out of date, searches may not return all the relevant results.

Note: The search system was substantially updated in version 23.0. The changes require new search indexes to be built after upgrading the CLC Server to 23.0 and above from an earlier version. If searches are not working or data associations with CLC Metadata tables are not recognized please contact your server administrator. See also the notes about this in section 16.2.1.

16.2.1 Rebuilding the index

The CLC Server maintains an index of all the elements for each CLC data location. These indexes are used when searching for data elements, and they are also needed for associations between data elements and CLC Metadata tables to be recognized.

Click on the Rebuild Index button next to a data location in the File system locations area under Configuration | Main configuration, to rebuild that location’s index. Click on the Rebuild all indexes button to rebuild indexes for all the locations (figure 16.4).

Note: Rebuilding an index can take a long time for locations with a lot of data elements.

When adding a new data location, re-building the index is the default option and indexes are updated with new elements are created or updated. Thus, they are generally expected to be up
to date. However, if searches are not returning the results expected, or data associated with CLC Metadata Tables is not being found, then re-indexing can solve those problems.

Important notes when upgrading to version 23.0 from an earlier version

Updates to the search system introduced in version 23.0 require search indexes to be manually built when upgrading from an earlier CLC Server to 23.0 and higher. Until that is done, searches for data in CLC Server data locations will not work, and data associations with CLC Metadata tables stored in CLC Server locations will not be recognized.

New indexes are stored in a folder called searchindex2 folder, in the installation area of the CLC Server.

Old indexes are not automatically deleted. After building new search indexes, the searchindex folder in the installation area and its contents can be deleted. That folder contains the old indexes. This can be worth doing if you have data locations with many elements, as indexes for those can take a lot of space.

Notes when moving the CLC Server

If you install the CLC Server on a new system, but are configuring existing directories as CLC Server file locations, you can either:

1. Rebuild the index for each data location on the new system, or

2. Replace the searchindex2 folder in the installation area of the new setup with the searchindex2 folder from the installation area of the old setup.

Copying the indexes may be preferable if your data locations have many data elements as re-indexing large locations can take some time.

Figure 16.4: The index for each file system location can be rebuilt. This is useful when searching does not return expected results, but can take a long time for locations with many data elements.

Details about indexing operations can be seen in the Audit log, under the Management tab.

16.3 Browse a CLC Server import/export directory

Under the Browse server import/export directories, you can browse the data held in areas configured as import/export directories in the External data area, under the Configuration tab (figure 16.5).

If folder or file is selected, information about it is displayed below the file browser.
16.4 Browse AWS S3 locations

In the **Browse S3 locations** area, you can browse data held in the S3 buckets you have access to via the AWS Connections configured in the **External data** area, under the Configuration tab. When an item is selected, options are enabled to download or delete it. When a folder is selected, options are enabled to upload to that folder, download from that folder, create a new folder or to delete the folder and its contents (figure 16.6). The URLs provided at the bottom are particularly useful if you are using the *CLC Server Command Line Tools*.

If you choose to download, a window pops up listing the selected files and providing the total download size (figure 16.7). You can download to a CLC location or to a CLC Server import/export area. When CLC data has been downloaded to a CLC location, it can be opened and inspected using a CLC Workbench connected to the CLC Server, or used for further analysis. For other file types, a CLC Server import/export directory would usually be selected.

*Note: AWS charges for downloading data from AWS S3.*

If the message "No active S3 locations found." is visible, it means that either no AWS Connections have been configured, or that there are no AWS Connections valid for the user logged into the CLC Server.

*Note: If you are using a CLC Genomics Cloud on AWS, non-admin users may see one or more entries called CGC system bucket. These cannot be browsed. They are used for system files, and are not intended for storing sample data or results. When logged in as an admin user, these buckets are listed with their full names, which start with cgc-system. They can be browsed but there is generally no need to do so.*
Figure 16.6: Browse S3 locations accessible via AWS Connections configured in the CLC Server. Data can be uploaded, downloaded, deleted and new folders created using functionality in this tab.

Figure 16.7: When downloading data from S3, a list of the selected files and the total size of the download is displayed. After clicking on the relevant download button, you will be prompted for a specific location to download to.
Chapter 17

Recycle bins

When users delete data from the Navigation Area of their CLC Workbench, it is placed in a recycle bin. Each user has an individual recycle bin in each File Location they have access to, including CLC Server File System Locations.

The recycle bin is a special concept that is not included in the permission control system. Any permissions applied to the data prior to deletion are no longer in effect. A particular user’s recycle bin can only be accessed by themselves and by administrative users. It is not possible to grant other users permission to access data in your recycle bin.

The recycle bin without a name beside it contains data deleted in versions of the CLC Server predating the concept of a per-user recycle bin. It can only be accessed by server administrators by selecting Show All Recycle Bins.

An exception: Deletion of data held in a CLC Server file system location named CLC_References is different than for other file system locations. Please refer to section 3.2.2 for details.

Recycle bins can be configured to automatically be emptied after a specified period. Server administrators can also access and empty the recycle bins of other users through the web administrative interface or using a CLC Workbench acting as a client for the CLC Server. We describe these functionalities below.

Enabling and configuring automatic cleanup of recycle bins

Recycle bins in any File System Location can be automatically emptied. By default this functionality is not enabled. When it is enabled, the default is to remove data that has been in the recycling bin for more than 100 days. To configure automatic emptying of recycle bins, go to the Configuration tab, and click on the Automatic recycle bin cleanup header. Click on the Configure button beside a File System Location (figure 17.1).

The recycle bin without a name beside it, which contains data deleted in versions of the CLC Server predating the concept of a per-user recycle bin, is not emptied by the automatic cleanup facility.

Emptying recycle bins using the web administrative interface

When logged into the web administrative interface, go to the Element info tab. All recycle bins in
a given location will be listed at the bottom of the location listing in the Navigation Area.

All recycle bins in a given location can be emptied in a single action, by clicking on a File Location or any folder or element in that location in the Navigation Area, and clicking on the button to **Empty All Recycle Bins**... under the **Info** tab.

An individual recycle bin can be emptied by clicking on it in the Navigation Area, and then clicking on the **Empty Recycle Bin**... bin button under the **Info** tab.

You will be asked for confirmation before contents of recycle bins are deleted.

The listing of recycle bins in a given location can be refreshed at any time by going to the Navigation Area, closing the folder containing those recycle bins, and then opening that folder again.

**Emptying recycle bins using a CLC Workbench**

If you are not already logged into the CLC Server from the CLC Workbench as an administrative user, log in by selecting the menu option:

**Connections | CLC Server Connection**

Then list all the recycle bins at the bottom of the server File System Location in the Navigation Area, as shown in figure 17.2:

**right-click the recycle bin ( ) | Location | Show All Recycle Bins**

Administrators an also empty the recycle bin of a particular user:

**right-click the recycle bin ( ) | Empty**

All recycle bins can be emptied in one go:

**right-click the data location ( ) | Location | Empty All Recycle Bins**

Please note that these operations cannot be undone.
Figure 17.2: Showing all recycle bins for a particular server File System Location.
Chapter 18

Troubleshooting

Various provisions are in place to help support the administration of your CLC Server. These include

- Automatic checks for consistency when certain types of configuration changes are made.
- The "check setup" tool, which can be run at any time from the web administrative interface, and which runs automatically when an administrator logs in.
- Functionality to report bugs and other problems.

In this chapter, we cover the last two features.

Troubleshooting tips can also be found in manual sections dedicated to specific areas, including:

- Troubleshooting server startup and permissions on Linux is covered in section 2.7.3.
- Grid Integration Tips are given in section 6.3.13.
- Troubleshooting External Applications is covered in section 14.12.

18.1 Check setup

The check setup tool checks your CLC Server is set up correctly. It runs automatically each time an administrative user logs into the web administrative interface. It can also be run on demand by clicking on the check setup link in the top right hand corner of the web interface and then clicking on the Generate Diagnostics Report button in the window that appears.

If issues are found, a red band and a warning message with a red background will be visible (figure 18.1). Clicking on the warning message at the top right opens a diagnostic report, where details of the problems can be found (figure 18.2). When the tool is run on demand, the diagnostic report will be displayed whether or not problems are found.

In the report, any tests where problems are identified are marked with a red exclamation mark. A green check mark is placed beside tests that passed. Click on any of the test names to see further information.
Indications of some problems are also given in the relevant area of the web interface. Examples are shown in the figures in this section.

Figure 18.1: If issues with the server configuration have been found, a red band and warning message is presented. Click on the warning message to open the diagnostic report. The red icon beside the first file system location also provides a signal that there is a problem.

Figure 18.2: Tests that passed are marked with a green check mark. Tests that failed are marked with a red exclamation mark. Here, a problem with the setup of import/export directories has been found. Such an issue is also indicated in the relevant configuration area, as seen in the background here.

Additional notes:

• A green check mark is presented beside "List license files" when the contents of the "licenses" folder in the installation area of the CLC Server can be listed. Click on this item to see a list of the licenses found and the products and versions supported by those licenses found are reported. Information about expired licenses is also presented. See figure 18.3.

• The "Data location indexing" section contains a list of the enabled file system locations and the number of files indexed.
  
  – On a job node setup, if there are communication issues between the job nodes and the master node, the status will indicate there is a problem.
  
  – The message "No data locations indexed" indicates that either no locations have been configured, or that all configured locations have been disabled.
  
  – If the number of files looks lower than expected, this can suggest that a file system location should be re-indexed.
  
  Rebuilding indexes is described in section 16.2.1.

• A green check mark is presented beside "Grid setup status" in two cases:
You have configured a grid setup and it is configured correctly.
- You have not configured a grid setup.

![List license files](image1)

Figure 18.3: Click on the List license files item in the report to see the list of the license files found in the licenses subfolder of the installation area. Products and versions supported are reported, and any expired license is noted with red text.

### 18.2 Bug reporting

Please contact our Support team if you have problems with the installation and configuration of your CLC Server. It can often be helpful if you provide information about the server configuration and the logs directly. To do this, you can submit a bug report from the web administrative interface by clicking on the **report a bug** link in the top right corner. This opens a bug report dialog, as shown in **18.4**. Enter the relevant information with as much detail as possible. If the server has access to the internet, click on the **Submit Bug Report** button to send the report directly to QIAGEN Bioinformatics Support. If the server does not have access to the internet, click on **Download bug report** to create a zip file that you can attach to an email you send to ts-bioinformatics@qiagen.com from a machine connected to the network.

Where logs or configuration information are unlikely to be relevant to your questions, please feel free to email the Support team directly at ts-bioinformatics@qiagen.com.

![Submitting a bug report](image2)

Figure 18.4: Submitting a bug report.
The bug report generated when you click on the report a bug link includes the following information:

- CLC Server log files
- A subset of the audit log showing the last events that happened on the CLC Server
- The CLC Server configuration files

No password information is included in the bug report.

In a job node setup you can include the information from the job nodes by checking the Include comprehensive job node info checkbox in the Advanced part of the dialog.

The process of gathering the information for the bug report can take a while, especially for job node setups.

If a CLC Workbench user experiences a server-related error, it is also possible to submit a bug report from a Workbench error dialog if they are presented with one. The same archive is included as when submitting a bug report from the server web interface.

All data sent to ts-bioinformatics@qiagen.com is treated confidentially.
Chapter 19

Command line tools

CLC Server Command Line Tools is a command-line client for the CLC Server. Installation and basic usage information can be found in the CLC Server Command Line Tools manual:

Chapter 20

Appendix

20.1 Use of multi-core computers

The tools listed below can make use of multi-core CPUs. This does not necessarily mean that all available CPU cores are used throughout the analysis, but that these tools benefit from running on computers with multiple CPU cores.

- Amino Acid Changes
- Annotate from Known Variants
- Annotate with Conservation Scores
- Annotate with Exon Numbers
- Annotate with Flanking Sequences

- Basic Variant Detection
- BLAST (will not scale well on many cores)

- Call Methylation Levels
- Copy Number Variant Detection
- Create Alignment
- Create RRBS-fragment Track

- Demultiplex Reads
- De Novo Assembly
- Differential Expression
- Differential Expression in Two Groups

- Filter against Known Variants
- Filter based on Overlap
- Fixed Ploidy Variant Detection

- GO Enrichment Analysis

- Identify Enriched Variants in Case vs Control Samples
- InDels and Structural Variants
- K-mer Based Tree Construction
- Link Variants to 3D Protein Structure
- Local Realignment
- Low Frequency Variant Detection
- Map Bisulfite Reads to Reference
- Map Reads to Contigs
- Map Reads to Reference
- Maximum Likelihood Phylogeny
- Merge Annotation Tracks
- Merge Variant Tracks
- Model Testing
- Predict Splice Site Effect
- QC for Read Mapping
- QC for Sequencing Reads
- QC for Targeted Sequencing
- Remove Variants Present in Control Reads
- Remove Marginal Variants
- RNA-Seq Analysis
- Trim Reads
- Trio Analysis

## 20.2 Non-exclusive Algorithms

Below is a list of algorithms which are non-exclusive, meaning that multiple jobs of these types can be run concurrently on a given node, be that a single server, job or grid node.

Algorithms marked as **Streaming** are I/O intensive and two streaming algorithms will not be run at the same time. When running on grid, **Streaming** algorithms are treated as exclusive, meaning that they will never run in conjunction with other algorithms (or themselves).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Streaming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Add attB Sites</td>
<td>X</td>
</tr>
<tr>
<td>Amino Acid Changes</td>
<td>X</td>
</tr>
<tr>
<td>Annotate from Known Variants</td>
<td>X</td>
</tr>
<tr>
<td>Annotate with Conservation Scores</td>
<td>X</td>
</tr>
<tr>
<td>Annotate with Exon Numbers</td>
<td>X</td>
</tr>
<tr>
<td>Annotate with Flanking Sequences</td>
<td>X</td>
</tr>
<tr>
<td>Annotate with Nearby Gene Information</td>
<td></td>
</tr>
<tr>
<td>Annotate with Overlap Information</td>
<td>X</td>
</tr>
<tr>
<td>Annotate with Repeat and Homopolymer Information</td>
<td></td>
</tr>
<tr>
<td>Apply Peak Shape Filter</td>
<td></td>
</tr>
<tr>
<td>Assemble Sequences</td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>Streaming</td>
</tr>
<tr>
<td>--------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Assemble Sequences to Reference</td>
<td></td>
</tr>
<tr>
<td>BLAST at NCBI</td>
<td></td>
</tr>
<tr>
<td>Call Methylation Levels</td>
<td></td>
</tr>
<tr>
<td>ChiP-Seq Analysis</td>
<td></td>
</tr>
<tr>
<td>Convert DNA To RNA</td>
<td>X</td>
</tr>
<tr>
<td>Convert from Tracks</td>
<td>X</td>
</tr>
<tr>
<td>Convert RNA to DNA</td>
<td>X</td>
</tr>
<tr>
<td>Convert to Tracks</td>
<td>X</td>
</tr>
<tr>
<td>Copy Number Variant Detection (CNVs)</td>
<td></td>
</tr>
<tr>
<td>Count-based statistical analysis</td>
<td></td>
</tr>
<tr>
<td>Create Alignment</td>
<td></td>
</tr>
<tr>
<td>Create BLAST Database</td>
<td></td>
</tr>
<tr>
<td>Create Box Plot</td>
<td></td>
</tr>
<tr>
<td>Create Entry Clone (BP)</td>
<td></td>
</tr>
<tr>
<td>Create Expression Browser</td>
<td></td>
</tr>
<tr>
<td>Create Expression Clone (LR)</td>
<td></td>
</tr>
<tr>
<td>Create GC Content Graph Track</td>
<td></td>
</tr>
<tr>
<td>Create Histogram</td>
<td></td>
</tr>
<tr>
<td>Create K-medoids Clustering for RNA-Seq</td>
<td></td>
</tr>
<tr>
<td>Create MA Plot</td>
<td></td>
</tr>
<tr>
<td>Create Mapping Graph Track</td>
<td></td>
</tr>
<tr>
<td>Create RRBS-fragment Track</td>
<td></td>
</tr>
<tr>
<td>Create Sample Report</td>
<td></td>
</tr>
<tr>
<td>Create Sequence Statistics</td>
<td></td>
</tr>
<tr>
<td>Create Track from Experiment</td>
<td></td>
</tr>
<tr>
<td>Create Track List</td>
<td></td>
</tr>
<tr>
<td>Create Tree</td>
<td></td>
</tr>
<tr>
<td>Create Venn Diagram for RNA-Seq</td>
<td></td>
</tr>
<tr>
<td>Demultiplex Reads</td>
<td></td>
</tr>
<tr>
<td>Download 3D Protein Structure Database</td>
<td>X</td>
</tr>
<tr>
<td>Download BLAST Databases</td>
<td>X</td>
</tr>
<tr>
<td>Download Pfam Database</td>
<td>X</td>
</tr>
<tr>
<td>Extract Annotations</td>
<td></td>
</tr>
<tr>
<td>Extract Consensus Sequence</td>
<td></td>
</tr>
<tr>
<td>Extract IsomiR Counts</td>
<td></td>
</tr>
<tr>
<td>Extract Reads</td>
<td>X</td>
</tr>
<tr>
<td>Fasta High-Throughput Sequencing Import</td>
<td>X</td>
</tr>
<tr>
<td>Filter against Known Variants</td>
<td>X</td>
</tr>
<tr>
<td>Filter Annotations on Name</td>
<td>X</td>
</tr>
<tr>
<td>Filter Based on Overlap</td>
<td>X</td>
</tr>
<tr>
<td>Find and Model Structure</td>
<td>X</td>
</tr>
<tr>
<td>Find Binding Sites and Create Fragments</td>
<td></td>
</tr>
<tr>
<td>Find Open Reading Frames</td>
<td></td>
</tr>
<tr>
<td>Gaussian Statistical Analysis</td>
<td></td>
</tr>
<tr>
<td>Gene Set Test</td>
<td></td>
</tr>
<tr>
<td>GO Enrichment Analysis</td>
<td></td>
</tr>
<tr>
<td>Algorithm</td>
<td>Streaming</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------</td>
<td>-----------</td>
</tr>
<tr>
<td>Hierarchical Clustering of Samples</td>
<td></td>
</tr>
<tr>
<td>Identify Enriched Variants in Case vs Control Samples</td>
<td>X</td>
</tr>
<tr>
<td>Identify Graph Threshold Areas</td>
<td></td>
</tr>
<tr>
<td>Identify Known Mutations from Mappings</td>
<td></td>
</tr>
<tr>
<td>Illumina High-Throughput Sequencing Import</td>
<td>X</td>
</tr>
<tr>
<td>Import Primer Pairs</td>
<td></td>
</tr>
<tr>
<td>Import SAM/BAM Mapping Files</td>
<td>X</td>
</tr>
<tr>
<td>Import Tracks from File</td>
<td></td>
</tr>
<tr>
<td>InDels and Structural Variants</td>
<td></td>
</tr>
<tr>
<td>Ion Torrent High-Throughput Sequencing Import</td>
<td>X</td>
</tr>
<tr>
<td>Learn Peak Shape Filter</td>
<td></td>
</tr>
<tr>
<td>Link Variants to 3D Protein Structure</td>
<td>X</td>
</tr>
<tr>
<td>Merge Annotation Tracks</td>
<td></td>
</tr>
<tr>
<td>Merge Overlapping Pairs</td>
<td></td>
</tr>
<tr>
<td>Merge Read Mappings</td>
<td>X</td>
</tr>
<tr>
<td>Merge Variant Tracks</td>
<td></td>
</tr>
<tr>
<td>Motif Search</td>
<td></td>
</tr>
<tr>
<td>PacBio High-Throughput Sequencing Import</td>
<td>X</td>
</tr>
<tr>
<td>Pfam Domain Search</td>
<td>X</td>
</tr>
<tr>
<td>Predict Splice Site Effect</td>
<td></td>
</tr>
<tr>
<td>Principal Component Analysis</td>
<td></td>
</tr>
<tr>
<td>Probabilistic Variant Detection</td>
<td></td>
</tr>
<tr>
<td>Proportion-based Statistical Analysis</td>
<td></td>
</tr>
<tr>
<td>Quality-based Variant Detection</td>
<td></td>
</tr>
<tr>
<td>Quantify miRNA</td>
<td></td>
</tr>
<tr>
<td>QC for Read Mapping</td>
<td></td>
</tr>
<tr>
<td>QC for Sequencing Reads</td>
<td>X</td>
</tr>
<tr>
<td>QC for Targeted Sequencing</td>
<td></td>
</tr>
<tr>
<td>Remove Duplicate Mapped Reads</td>
<td></td>
</tr>
<tr>
<td>Remove Homozygous Reference Variants</td>
<td></td>
</tr>
<tr>
<td>Remove Information from Variants</td>
<td>X</td>
</tr>
<tr>
<td>Remove Marginal Variants</td>
<td>X</td>
</tr>
<tr>
<td>Remove Variants Present in Control Reads</td>
<td>X</td>
</tr>
<tr>
<td>Rename Sequences in Lists</td>
<td></td>
</tr>
<tr>
<td>Reverse Complement Sequence</td>
<td></td>
</tr>
<tr>
<td>Subsample Sequence List</td>
<td></td>
</tr>
<tr>
<td>Sanger High-Throughput Sequencing Import</td>
<td>X</td>
</tr>
<tr>
<td>Secondary Peak Calling</td>
<td></td>
</tr>
<tr>
<td>Score Regions</td>
<td></td>
</tr>
<tr>
<td>Split Sequence List</td>
<td></td>
</tr>
<tr>
<td>Transcription Factor ChiP-Seq</td>
<td></td>
</tr>
<tr>
<td>Translate to Protein</td>
<td></td>
</tr>
<tr>
<td>Trim Sequences</td>
<td></td>
</tr>
<tr>
<td>TRIO analysis</td>
<td></td>
</tr>
<tr>
<td>Update Sequence Attributes in Lists</td>
<td></td>
</tr>
<tr>
<td>Whole Genome Coverage Analysis</td>
<td></td>
</tr>
</tbody>
</table>
20.3 DRMAA libraries

Distributed Resource Management Application API (DRMAA) libraries are provided by third parties. Please refer to the distributions for instructions for compilation and installation, and contact the DRMAA providers if problems arise. QIAGEN Bioinformatics Support is not able to troubleshoot DRMAA library issues.

Information in this section of the manual is provided as a courtesy, but should not be considered a replacement for reading the documentation that accompanies the DRMAA distribution itself.

20.3.1 DRMAA for SLURM

Information about DRMAA for SLURM can be found at https://slurm.schedmd.com/download.html.

20.3.2 DRMAA for LSF

The source code for this library can be downloaded from https://github.com/PlatformLSF/lsf-drmaa. Please refer to the documentation that comes with the distribution for full instructions. Of particular note are the configure parameters "--with-lsf-inc" and "--with-lsf-lib" parameters, used to specify the path to LSF header files and libraries respectively.

20.3.3 DRMAA for PBS Pro

Source code for this library can be downloaded from http://sourceforge.net/projects/pbspro-drmaa/.

Please refer to the documentation that comes with the distribution for full instructions.

Some items of particular note:

- The The --with-pbs= parameter is used to specify the path to the PBS installation root. The configure script expects to be able to find lib/libpbs.a and include/pbs_ifl.h in the given root area, along with other files.

- SSL is needed. The configure script expects that linking with "ssl" will work, thus libssl.so must be present in one of the system's library paths. On Red Hat and SUSE you will have to install openssl-devel packages to get that symlink (or create it yourself). The install procedure will install libdrmaa.so to the provided prefix (configure argument), which is the file the CLC Server needs to know about.

- Special steps need to be taken to compile DRMAA against PBS Pro 2021.1.1. These are outlined below.

The PBS DRMAA library can be configured to work in various modes as described in the README file of the pbs-drmaa source code. We have experienced the best performance, when the CLC Server has access to the PBS log files and pbs-drmaa is configured with wait_thread 1.

Compiling against PBS Pro 2021.1.1
To compile against PBS Pro 2021.1.1, a library (lsec) must be added in the configure script included in the DRMAA distribution. We successfully compiled DRMAA from http://sourceforge.net/projects/pbspro-drmaa/files/pbs-drmaa/1.0/pbs-drmaa-1.0.19.tar.gz/download against PBS Pro 2021.1.1 by doing the following:

- Ran the command `sed -i 's/-lpbs/-lpbs -lsec/g' configure`
- Ran the command `./configure --with-pbs=/PATH/TO/PBS`
- Ran the make command.
  - make fails at this point.
  - Step through the same commands a second time:
    - `sed -i 's/-lpbs/-lpbs -lsec/g' configure`
    - `./configure --with-pbs=/PATH/TO/PBS`
    - `make`
      - make should succeed this time.
    - Then, with appropriate privileges, in our case, root privileges, we ran `make install`

20.3.4 DRMAA for OGE or SGE

OGE/SGE comes with a DRMAA library.

20.4 Consumable Resources

Setting up Consumable Resources with LSF

The following information was provided by IBM.

If you have questions or issues with setting up a consumable resource for LSF, please refer to your LSF documentation. For questions not covered there, please contact ruzhuchen@us.ibm.com and achristi@ca.ibm.com.

LSF has the ability to do "license scheduling" and ensure that CLC Server jobs running under LSF are only dispatched when there are available CLC Grid Worker licenses. When such scheduling is configured, CLC jobs for which no free licenses are available would stay in "pend" status, waiting for a CLC Grid Worker license to become available.

There are two parts to making use of this type of scheduling:

1. Configure the consumable resource in LSF.
2. Specify a clcbio license reservation when jobs are submitted to LSF.

Configuring the consumable resource in LSF

Add a consumable resource called clcbio in $LSF_ENVDIR/lsf.shared:
Begin Resource
RESOURCENAME TYPE INTERVAL INCREASING DESCRIPTION # Keywords
mips Boolean () () (MIPS architecture)
...
...
clcbio Numeric () N (clcbio license)
End Resource

Add the number of clcbio licenses in $LSF_ENVDIR/lsf.cluster.<clusternam e>: 

Begin ResourceMap
RESOURCENAME LOCATION
# CLCBIO license resource
clcbio (14@[all]) # 14 clcbio licenses can be used 
# ....
End ResourceMap

This example shows a configuration for 14 CLC Grid Worker licenses, which means that up to 14 
CLC jobs can be running on the LSF cluster at the same time. This integer needs to be changed 
to the number of licenses you own.

The configuration shown here assumes the CLC Grid Worker licenses can only be use in the LSF 
cluster as LSF will manage the free token count from the scheduling side.

In this context, LSF does not replace or directly talk to the LMX license server for CLC licenses. 
Rather, LSF manages the CLC Grid Worker license reservations internally.

Specify a clcbio license reservation when jobs are submitted to LSF

CLC jobs submitted to LSF need to have a clcbio license reservation specified.

This can be done in several different ways:

- via the CLC Grid Preset "Native Specification" field. (This is the most convenient method.) 
  Simply add:

  -R "rusage[clcbio=1]"

  to this field.
- via the batch job submission command line
- using the RES_REQ line inside the lsb.queues file
- via an application profile (lsb.applications)

Important: After any LSF configuration file changes, one needs to reconfigure LSF for the changes 
to take effect. That is, run:

lsadmin reconfig
badmin reconfig
These are "safe" commands to run. That is, pending LSF jobs will continue to "pend" in status and running LSF jobs will continue to run.

20.5 Server system communication diagrams

The communication between CLC software and other local resources for some common CLC Server setups are summarized in the diagrams below.

These diagrams do not include representation of communication with external resources, such as accessing reference data from remote locations, downloading licenses for CLC software, or interacting with a CLC Genomics Cloud setup.

Representative CLC Server grid setup with LDAP authentication

Figure 20.1: A representative CLC Server setup, where job execution is carried out on grid nodes, and authentication has been configured to use LDAP. The dotted line between grid node and the master CLC Server represents the communication that triggers an index update when a job has created files in a CLC data location.

Representative CLC Server grid setup using built-in authentication
Figure 20.2: A representative CLC Server setup, where job execution is carried out on grid nodes, and the built-in authentication mechanism is being used. The dotted line between grid node and the master CLC Server represents the communication that triggers an index update when a job has created files in a CLC data location.

Representative CLC Server job node setup with LDAP authentication

Figure 20.3: A representative CLC Server setup, where job execution is carried out on job nodes, and authentication has been configured to use LDAP.

Representative CLC Server job node setup using built-in authentication
CHAPTER 20. APPENDIX

20.6 Third party libraries

The CLC Server includes a number of third party libraries.

Please consult the files named NOTICE and LICENSE in the server installation directory for the legal notices and acknowledgements of use.

For the code found in this product that is subject to the Lesser General Public License (LGPL) you can receive a copy of the corresponding source code by sending a request to our support team at ts-bioinformatics@qiagen.com.

20.7 External network connections

Some functionality available in CLC software requires access to specific addresses on the Internet. A list of the internet sites accessed and a listing of the tools involved can be found at https://qiagen.secure.force.com/KnowledgeBase/KnowledgeNavigatorPage?id=kA41i000000L5smCAC.

The list of sites there can be referred to when configuring firewall settings for networks that utilize a whitelist approach. For CLC Servers with nodes, any nodes that need to execute functionality listed on that webpage will need access to the relevant sites.

20.7.1 Proxy settings

Proxy settings for a CLC Server setup, are configured in one or more .vmoptions files, as described below.
CHAPTER 20. APPENDIX

Proxy settings on a single CLC Server or a master node

Add the lines below to the CLCGenomicsServer.vmoptions file on the single server or master node. This file is located in the installation area of the CLC Server software.

-Dhttp.proxyHost=
-Dhttp.proxyPort=
-Dhttp.nonProxyHosts="localhost|127.0.0.1|192.168.*"

Add the relevant information for your site after the = signs. For -Dhttp.nonProxyHosts, provide a pipe-separate (|) list of hosts that should be reached directly, bypassing the proxy. Typically this is localhost and a list of relevant internal hosts. Full addresses or patterns that start or end with a '*' wildcard can be provided.

Further information about these settings can be found at https://docs.oracle.com/javase/8/docs/technotes/guides/net/proxies.htmlE.

Proxy settings on a job node setup

In addition to the settings described above for the master node, the CLCGenomicsServer.vmoptions file on each job node must also have those proxy settings configured.

As on the master node, the CLCGenomicsServer.vmoptions file can be found in the installation area of the CLC Server software on each job node.

Proxy settings on a grid node setup

In addition to the settings described above for the master node, a clcgridworker.vmoptions file must be present in each deployed grid worker area with the proxy settings described above added.

Further details about clcgridworker.vmoptions files is provided in section 6.3.10.

20.8 Read Mapper Reference Caching

In some cases, repeated mappings against the same reference will result in a dramatically reduced runtime because the internal data structure used for mapping the reads, which is reference specific, can be reused. This has been enabled by storing files in the system tmp folder as a caching mechanism. Only a certain amount of disk space will be used and once reaching the limit, the oldest files are cleaned up. Consequently, the reference data structure files will automatically have to be recreated if the cache was filled or the tmp folder was cleaned up.

The default space limit is 16 GB.

The reference cache size can be changed by creating a file called readmapper.properties, with an entry referencecachesize = <size in bytes>, for example

referencecachesize = 8589934592

A copy of this file should then be placed into the settings folder in the following locations:

- Under the installation area of the CLC Server on the single server or master node.
• Under the installation area of the CLC Server of each job node in a job node setup.
• Under each grid worker folder defined in your grid presets on a grid node setup.

20.9 Monitoring

We recommend that you monitor the health and performance of the servers that the CLC Server software is running on and also monitor some key metrics of the CLC Server itself. This will enable you to react quickly to any problems that occur and can aid in optimization of server performance.

With regards to the servers’ physical resources, monitoring the amount of free memory and disk space is recommended, as consumption of these can be substantial depending on types and numbers of analyses being run.

Monitoring metrics of the CLC Server itself enables you to keep tabs on how well the jobs processing is going. The metrics the CLC Server provides are available as JMX\(^1\) attributes. Software from a third party will be necessary to set up the monitoring of these attributes. Numerous software products for this are available and most support JMX. If your monitoring software supports the raising of alarms, you can set up triggers based on these metrics to receive alerts when a situation arises that needs attention.

20.9.1 Setting up JMX monitoring

No special configuration is necessary if monitoring will take place locally. However, JMX must be enabled for remote monitoring. This is done by adding a few settings to the server vmoptions file. The relevant .vmoptions file is located in the root of the server installation folder of a single server, or the root of the installation folder of the master server in the case of a node setup (Hereafter this folder will be referred to as CLC_SERVER_BASE).

Enable JMX with no security Add the following to the .vmoptions file:

- Dcom.sun.management.jmxremote.port=9999
- Dcom.sun.management.jmxremote.ssl=false
- Dcom.sun.management.jmxremote.authenticate=false

Enable JMX with authentication

1. Add the following to the .vmoptions file, instead of the lines provided in the section above.

- Dcom.sun.management.jmxremote.port=9999
- Dcom.sun.management.jmxremote.ssl=true
- Dcom.sun.management.jmxremote.authenticate=true
- Dcom.sun.management.jmxremote.password.file=../conf/jmxremote.password
- Dcom.sun.management.jmxremote.access.file=../conf/jmxremote.access

\(^1\)https://en.wikipedia.org/wiki/Java_Management_Extensions
2. Create the access authorization file `CLC_SERVER_BASE/conf/jmxremote.access` and write the following in it:

```plaintext
monitorRole readonly
controlRole readwrite
```

3. Create the password file `CLC_SERVER_BASE/conf/jmxremote.password` and write the following in it:

```plaintext
monitorRole clcserver
controlRole clcserver
```

Further information about setting up JMX monitoring can be found in the Oracle guide on Monitoring and Management Using JMX Technology [http://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html](http://docs.oracle.com/javase/8/docs/technotes/guides/management/agent.html).

### 20.9.2 Completed process metrics

Object name: `com.clcbio.server:type=CompletedProcesses`

The "completed process" metrics can be used to evaluate the processes that are complete either because they were successful or because they failed. Canceled processes are ignored. The CLC server provides a temporal view of the latest completed processes, with two different temporal views available: time frame view or a history view that includes a fixed number of the most recently completed processes.

The size of the time frame to view and the number of entries for the history view can be configured directly through JMX or by editing the `Monitoring.properties` file. This properties file is located in this folder: `CLC_SERVER_BASE/settings/`. Please change the default settings to values that fit your specific setup. Lower values will generally result in a more reactive monitoring solution, but values that are too low may result lead to false alarms.

The following is a list of all the process related metrics that are available:

**Number of processes in history**

**Attribute name:** `NumberOfProcessesInHistory`

The number of processes currently in the history. Successful and failed processes are included. Canceled processes are not included. The maximum size of the history can be set using the `SizeOfHistory` attribute, available through JMX and the configuration file.

**Number of failed processes in history**

**Attribute name:** `NumberOfFailedProcessesInHistory`

The number of failed processes currently in the history.

**Fraction of failed processes in history**

**Attribute name:** `FractionOfFailedProcessesInHistory`
The fraction of the processes in the history that have failed. This fraction is not of much value if the number of processes in the history is very low.

**Number of processes within time frame**

**Attribute name:** NumberOfProcessesWithinTimeFrame

The number of processes that have been completed between now and a variable number of milliseconds earlier. Both successful and failed processes are included. Canceled processes are not included. The time frame can be set using the TimeFrameInMilliseconds attribute, that is available through JMX and the configuration file.

**Number of failed processes within time frame**

**Attribute name:** NumberOfFailedProcessesWithinTimeFrame

The number of failed processes that have been completed between now and a variable number of milliseconds earlier.

**Fraction of failed processes within time frame**

**Attribute name:** FractionOfFailedProcessesWithinTimeFrame

The fraction of failed processes compared to the total number of processes that have been completed between now and a variable number of milliseconds from now. This fraction is not of much value if the number of processes in the time frame is very low.

### 20.9.3 Process execution metrics

**Object name:** com.clcbio.server:type=ProcessExecution

Process execution metrics allow measurement of how many jobs are being processed and how many are in a queue because they are waiting for available processing resources.

On job node setups, the object names are available on all nodes, but it only makes sense to monitor them on the master node since this is where the jobs are managed. If a grid is used to process the jobs, the actual queue will be a part of the grid system, which results in the processes being moved almost instantly to the "currently processing" state and the CLC Server queue itself is then empty.

**Currently processing**

**Attribute name:** CurrentlyProcessing

Number of processes being executed at the moment.

**Waiting for resources**

**Attribute name:** WaitingForResources
On a job node setup, the number of processes that are queued and are waiting for a node to be available for processing. This does not include processes that are waiting for output from another process. It includes only processes that have the input they need and are ready to be processed, but where no resources are available at that moment.

### 20.9.4 Job node metrics

Object name: `com.clcbio.server:type=JobNodes`

With the job node metrics you can monitor a master node’s connection with its job nodes. The object name is available on all nodes, but it only makes sense to monitor this on the master node. There are two sets of attributes to monitor. One set provides an aggregated view of all the job nodes while the other provides individual attributes for each job node.

Communication errors are only reported if the server uses the General queue. If the High throughput queue is used, the master node never contacts the job nodes on its own initiative and therefore there is no support for monitoring the job nodes through JMX in the current version of the server.

Apart from communication error attributes, the set of individual attributes also includes information about the host and port of a given job node. This information is not meant for monitoring as such, but is included for convenience, as when an error does arise identification of the job node involved is usually necessary.

**Communication failed**

Attribute name: `CommunicationFailed`

This attribute is set to true if the master node is currently having problems communicating with this particular job node.

**Seconds since communication failed**

Attribute name: `SecondsSinceCommunicationFailed`

The number of seconds since the communication with the job node first failed. As soon as the master node succeeds in connecting with the job node again, this value returns to zero. This attribute can be useful if you want to avoid reacting to very short fallouts in communication.

**Max seconds since communication failed**

Attribute name: `MaxSecondsSinceCommunicationFailed`

The maximum number of seconds since communication with any of the job nodes first failed. The advantage of this metric is that monitoring of it can be set up once and does not need to be changed if a job node is attached or detached.

**Number of job nodes**

Attribute name: `NumberOfJobNodes`
The number of job nodes currently attached to the master server.

**Number of failed job nodes**

Attribute name: NumberOfFailedJobNodes

The number of job nodes the master server currently has problems communicating with.
Bibliography


Index

Active directory, 40
AD, 40
AWS S3, 85

BaseSpace, 86
BLAST, 94

Command-line installation, 19
Communication, diagrammatic, 190
Concurrent jobs per node, 80
Configuring setup, 57
consumable resource, 188
Cores, restrict usage, 36
CPU, restrict usage of, 36

Data
deletion of, 175
Data compression, 31
Data locations
Data compression, 31
File system locations, 27, 30
Job nodes, 30
Reference data locations, 30
Deleting data, 175
Direct data transfer settings, 84
DRMAA, 187

Encrypted connection, 32
External applications, 110
Running from a Workbench, 163
Running using the command line, 165

Fairness factor, 79
GSSAPI, 40
HTTPS, 32

Index, rebuilding, 171
Installing Server plugins on job nodes, 59
JMS, 194
Kerberos, 40

LDAP, 40
License
non-networked machine, 22, 98

Memory allocation, 36
Metrics, 194
Monitoring, 194
Multi-job Processing, 78
Multi-job processing on grid, 69
permissions, 44
proxy, 192
Quiet installation, 19
RAM, 36
Read mapping
Reference cache, 193

Secure socket layer, 32
Server commands, specify on job nodes, 58
Silent installation, 19
SSL, 32
Status and management, 97
Status and running modes, 99
Surveillance, 194
System requirements, 10
System statistics, 101

Third party libraries, 192
tmp directory, how to specify, 35

Upgrading an existing installation, 20
User credentials, 57
User statistics, 100

.xmoptions, memory allocation, 36

Workflows
Configuring, 87
Installing, 87
Running, 89
Updating, 90

Xmx argument, 36