CLC **Server**

Administrator

USER MANUAL

Administrator Manual for
CLC Server 24.0.1
Windows, macOS and Linux

March 20, 2024

This software is for research purposes only.

QIAGEN Aarhus
Silkeborgvej 2
Prismet
DK-8000 Aarhus C
Denmark

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

2.5 Allowing 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 Server display name 27

3.3 Adding locations for saving data 28

3.4 Changing the listening port 33

3.5 SSL and encryption 34

3.6 Changing the tmp directory 37

3.7 Setting the amount of memory available for the JVM 37

3.8 Limiting the number of cpus available for use 38

3.9 HTTP settings 38

3.10 External network connections 38

3.11 Proxy settings 38

4 Managing users and groups 40

4.1 Changing the root password 40

4.2 User authentication for the CLC Server 41

4.3 User authentication via the Workbench for built-in authentication 46

5 Access privileges and permissions 48

5.1 Controlling group access to CLC Server data 48

5.2 Controlling access to the server, server tasks and external data 51

5.3 Customized attributes on data locations 54

6 Job processing 59

6.1 Introduction to servers setups 59

6.2 Model I: Master server with dedicated job nodes 60

6.3 Model II: Master server submitting to grid nodes 66

6.4 Model III: Single Server setup 81

6.5 Workflow settings 83

7 Working with external data locations 87

7.1 Import/export directories 88

7.2 Direct data transfer from client systems 89

7.3 AWS Connections in the CLC Server 90

7.4 AWS S3 public buckets 92

7.5 Browse AWS S3 locations 93

7.6 Illumina BaseSpace 95

8 Working with CLC Server File System Locations 96

8.1 Browsing CLC Server File System Locations 96

8.2 Searching CLC Server File System Locations 97

9 Workflows 100

9.1 Installing and configuring workflows 100

9.2 Executing workflows 102

9.3 Updating workflows 104

10 BLAST databases **108**

11 Status and management **111**

11.1 Downloading a license via the web interface . . . 111
11.2 Server maintenance . . . 113
11.3 User statistics . . . 113
11.4 System statistics . . . 115

12 Queue **116**

13 Audit log **117**

14 Server plugins **120**

15 External applications **123**

15.1 External application configurations . . . 125
15.2 Configuring external applications . . . 127
15.3 Using consistent reference data in external applications . . . 144
15.4 Import and export of external application configurations . . . 144
15.5 Updating external application configurations . . . 146
15.6 Example: Velvet (standard external application) . . . 147
15.7 Example: Bowtie (standard external application) . . . 151
15.8 Example: Kraken2 (containerized external application) . . . 156
15.9 Example: MAFFT (containerized external application) . . . 170
15.10 External applications in workflows . . . 176
15.11 Running external applications . . . 178
15.12 Troubleshooting external applications . . . 181

16 CLC Genomics Cloud Access **183**

17 Recycle bins **185**

17.1 Automatic cleanup of recycle bins . . . 185
17.2 Emptying recycle bins . . . 186
17.3 Recycle bin restrictions . . . 188

18 Configuring CLC Workbench connections **189**

19 Troubleshooting **190**

19.1 Check setup 190

19.2 Bug reporting 192

20 Command line tools **194**

21 Appendix **195**

21.1 Use of multi-core computers 195

21.2 Non-exclusive Algorithms 196

21.3 DRMAA libraries 199

21.4 Consumable Resources 200

21.5 Server system communication diagrams 202

21.6 Third party libraries 204

21.7 Read Mapper Reference Caching 204

21.8 Monitoring 205

Bibliography **210**

Index **210**

Chapter 1

Introduction

Welcome to *CLC Server 24.0.1*, 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 21.5.

Information about running analyses on AWS can be found in the *CLC Cloud Module* manual at https://resources.qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Introduction_CLC_Genomics_Cloud.html.

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).

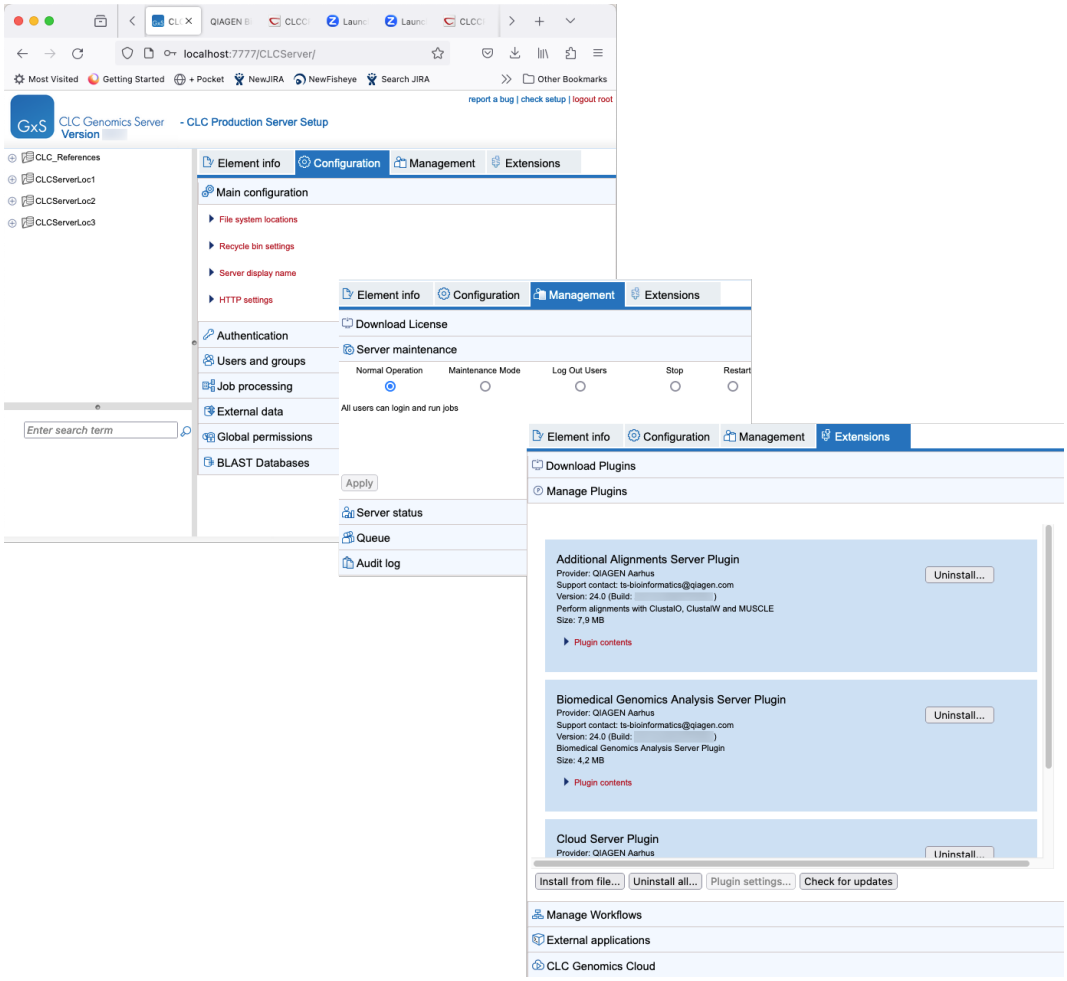


Figure 1.2: *Adminstration and management of a CLC Server is done primarily via the web administrative client. The contents of the three tabs containing primary administrative functionality are shown here.*

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.

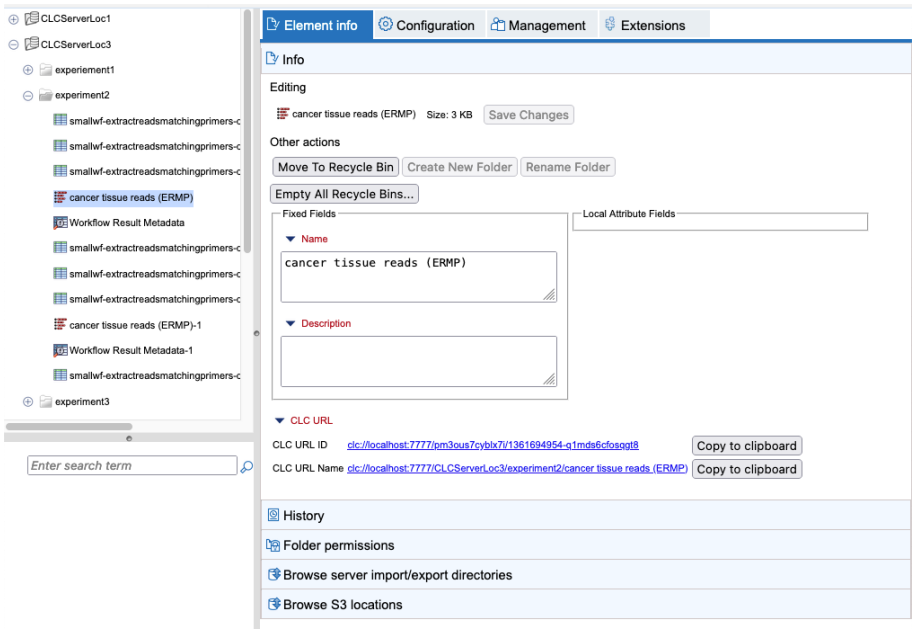


Figure 1.3: *Adminstration actions relating to data stored in CLC Server file locations is done under the Element info tab. Standard users also have access to this tab, but admin-related functionality is not made visible in that case.*

1.1 System requirements

The system requirements of *CLC Server* are:

Server system

- Windows 10, Windows 11, Windows Server 2016, Windows Server 2019 and Windows Server 2022
- Mac: macOS 11, 12, 13 and 14
- 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 CLC Server** This is needed for running analyses on the *CLC Server* or its execution nodes and for submitting jobs via the *CLC Server* to run on the cloud[2]. The *CLC Server* license will allow a certain number of open sessions. This refers to the number of active, individual log-ins from server clients (CLC Workbenches, CLC Server Command Line Tools, or the web client). 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 section 2.6.

[1] Containerized external applications were introduced in *CLC Server* 21.0.

[2] To run jobs on the cloud, the Cloud Server Plugin must be installed, and an AWS Connection must be configured for an account with access to CLC Genomics Cloud infrastructure.

- **A license for** *CLC Workbench* **software** A *CLC Workbench* is used to launch analyses and view the results. It is also used for designing workflows and creating workflow installers. Find the user manuals and deployment manual for the Workbenches at https://digitalinsights.qiagen.com/technical-support/manuals/.
- **A network license, if submitting analyses to grid nodes.** This is described in section 6.3.6.

1.3 CLC Genomics Server

The *CLC Genomics Server* is shipped with the tools listed below,which can be started from *CLC Genomics Workbench* and *CLC Server Command Line Tools*. A subset can be launched using the *CLC Main Workbench*. Please refer to the *CLC Genomics Workbench* manual for details about the tools listed. Import and export tools are not listed explicitly here, but import and export of many sequence related formats as well as other formats is supported.

- 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
  - Annotate with GFF/GTF/GVF file
  - 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
- 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 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
  - 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
  - Extract Reads
  - Filter on Custom Criteria
  - Merge Overlapping Pairs
  - Combine Reports
  - Create Sample Report
  - Modify Report Type
  - Create Track List
  - Merge Annotation Tracks
  - Merge Variant Tracks
  - Convert to Tracks
  - Convert from Tracks
  - Annotate with Exon Numbers
  - Annotate with Nearby Information
  - Annotate with Overlap Information
  - Filter Annotations on Name
  - Filter Based on Overlap
  - Create GC Content Graph
  - Create Mapping Graph
  - Identify Graph Threshold Areas
  - Create Sequence List
  - 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 installing Server plugins (see chapter 14).

Latest improvements

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

https://digitalinsights.qiagen.com/products/qiagen-clc-genomics-server/latest-improvements/current-line/.

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-it authentication system, users and groups can be set up now or later.
8. Add data locations (section 3.3).
9. Check your server setup using the **check setup** link in the upper right corner (section 19.1.
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 19.

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.

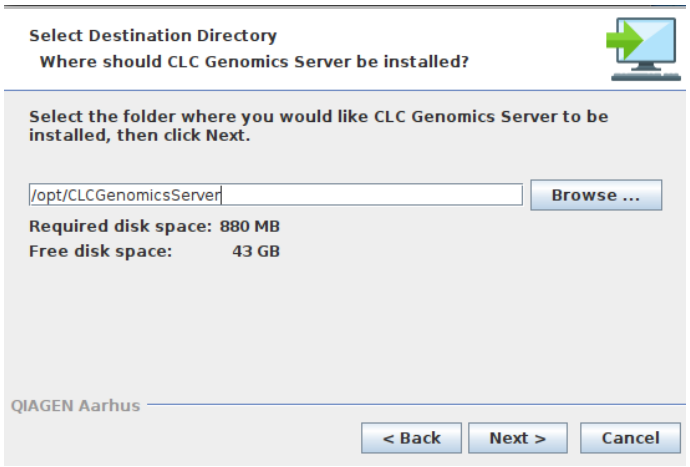


Figure 2.1: *If installing for the first time, the location to install to is specified.*

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 `systemd` service unit file describing the *CLC Server* service, and setting the ownership

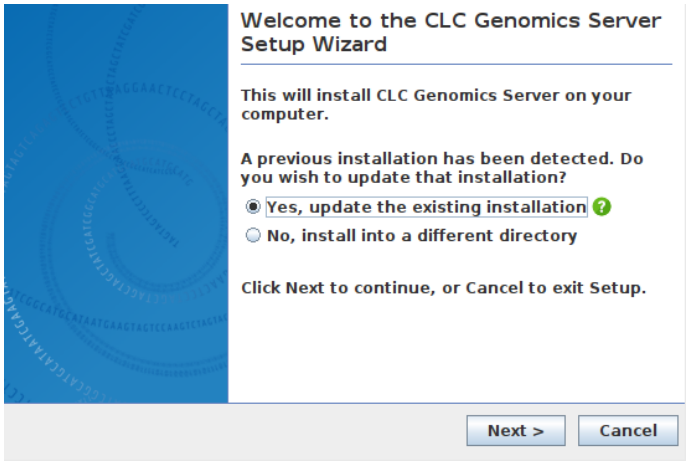


Figure 2.2: *When upgrading, choose to update the existing installation.*

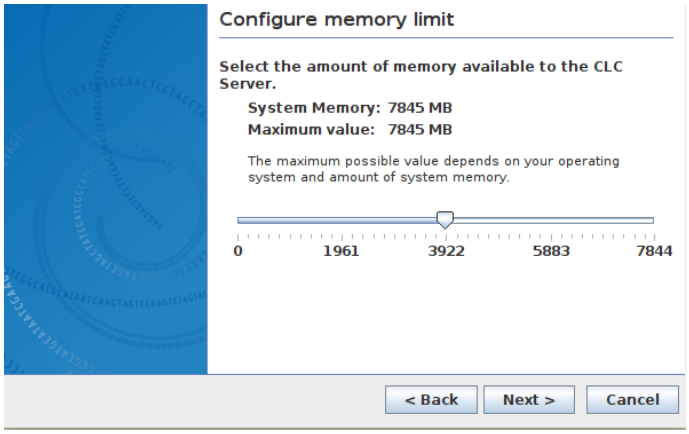


Figure 2.3: *Choose the maximum amount of memory used by the server.*

of the files in the installation area to this user. 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 software.

- **Console mode** This mode is particularly useful when installing 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 the software into the directory "c:\bioinformatics\clc" on a Windows system using silent mode with console output. The same form of command works for any CLC installer.

  ```
  CLCGenomicsServer_23_0_5_64.exe -console -q -dir "c:\bioinformatics\clc"
  ```

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

  ```
  ./CLCGenomicsServer_23_0_5_64.sh -q -c -dir /opt/bioinformatics/clc
  ```

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 11.2. Getting information about who is logged in is described in section 11.3.
- 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 14.
- **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 3.4.
- Enabling SSL is described in section 3.5.

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 3.4.

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 11.2). 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 11.3). 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 11.1).

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.

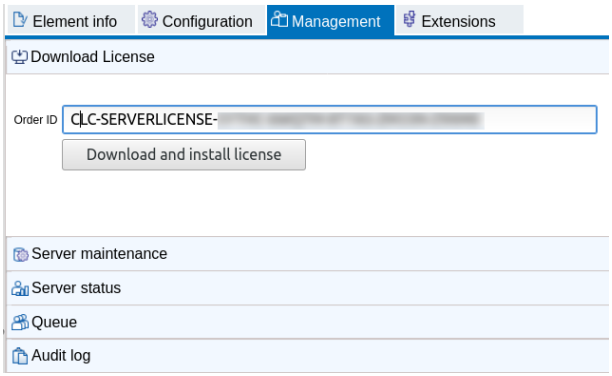


Figure 2.4: *License management is done under the Management tab.*

If you are working on a system that does not have access to the external network, then please refer to section 2.6.1.

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.

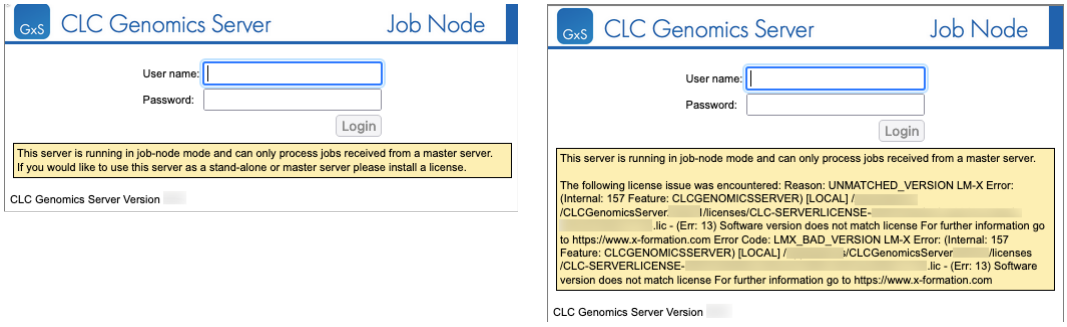


Figure 2.5: *A warning is shown in the web administrative login page for a single server or master node when no license is present for the CLC Server (left) and when a license for the CLC Server is present but is not valid for the version being run (right).*

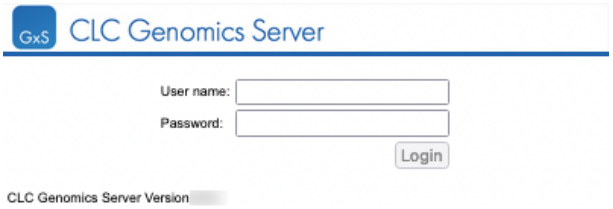


Figure 2.6: *The web administrative interface login page for a single server or master node when a valid license for the CLC Server is present.*

- 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.

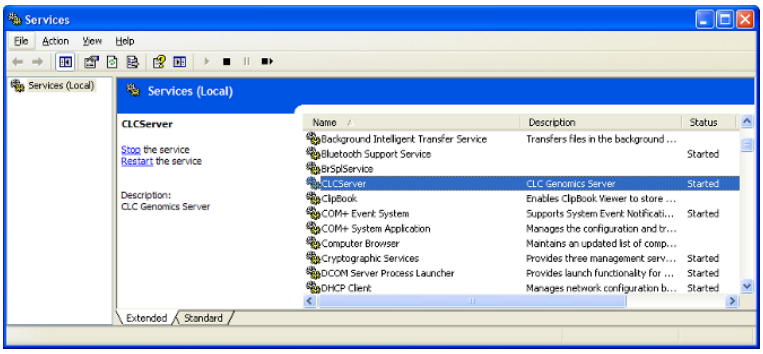


Figure 2.7: *Stopping and restarting the server on Windows by clicking the blue links.*

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 `systemd` service unit file is created in /etc/systemd/system.

This file 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 systemctl start CLCGenomicsServer.service
```

To stop the *CLC Server*:

```
sudo systemctl stop CLCGenomicsServer.service
```

To restart the *CLC Server*:

```
sudo systemctl restart CLCGenomicsServer.service
```

To view the status of the *CLC Server*:

```
sudo systemctl status CLCGenomicsServer.service
```

Start service on boot up:

To enable automatic start of service:

```
sudo systemctl enable CLCGenomicsServer.service
```

To disable automatic start of service:

```
sudo systemctl disable CLCGenomicsServer.service
```

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 configured in the `systemd` service unit file:

   ```
   sudo systemctl start CLCGenomicsServer.service
   ```

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:

   ```
   sudo ./CLCGenomicsServer start-launchd
   ```

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

Configuration of the *CLC Server* is primarily carried out via a web interface when logged in as a user with administrative rights (see section 4.2.1).

The login page can be reached using the host information followed by the port the server is listening on. The default port is `7777`. For example:

`http://clccomputer:7777/` or `http://localhost:7777/`

See section 3.4 for information about changing the listening port.

An admin user is built into the *CLC Server*. The default credentials for this user are:

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

We recommend that you change this password. How to do this is described in section 4.1. Note that there are additional considerations when working with a job node setup. See section 6.2.2 for details.

3.2 Server display name

Set the server display name under:

Configuration ( ) | **Main configuration** ( ) | **Server display name**

This name is visible in the web client (figure 3.1), in CLC Workbenches (figure 3.2), and is reported when a CLC Server Command Line Tools command is run.

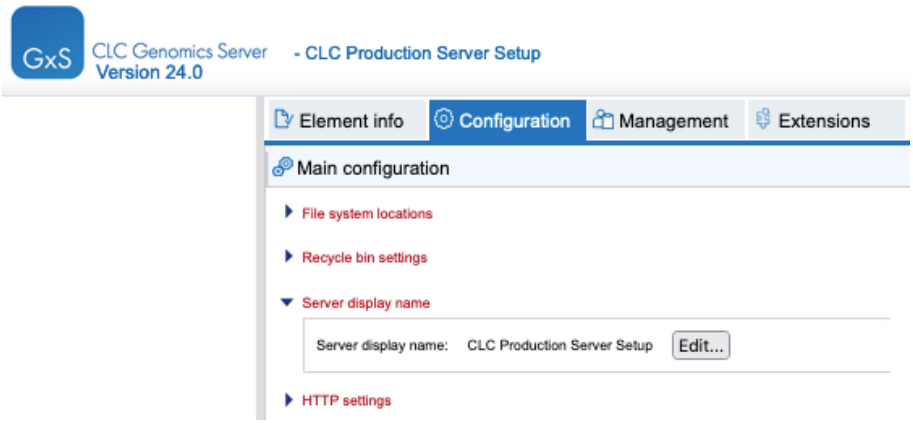


Figure 3.1: *The display name is set under the Configuration tab and is displayed at the top of the web client.*

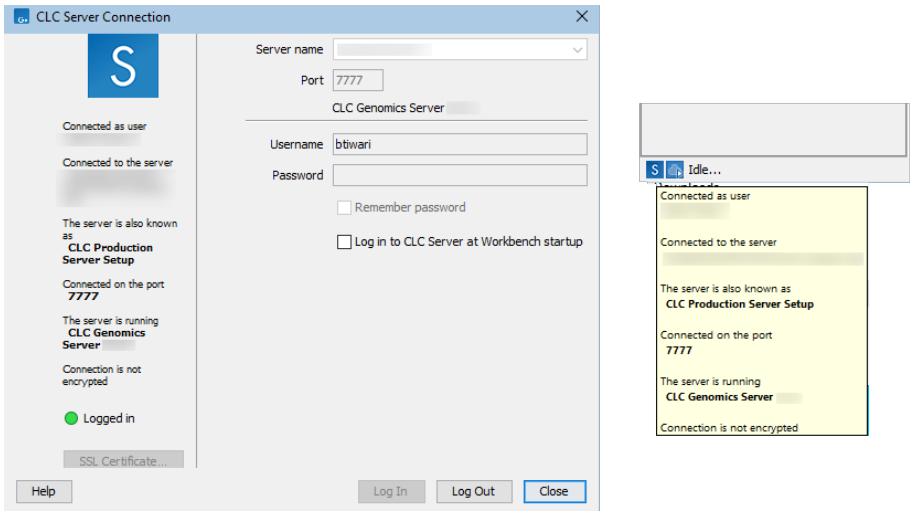


Figure 3.2: *The display name is shown in the CLC Server Connection dialog in CLC Workbenches, as well as in the tooltip revealed when the mouse cursor is hovered over the server connection icon at the bottom left of the Workbench.*

3.3 Adding locations for saving data

Data imported to the *CLC Server* or results generated using the *CLC Server* are usually stored in a CLC Server File System Location. Configuring these is covered in this chapter (section 3.3.1).

Directories on the file system that the *CLC Server* needs read or write access to, for example when reading data for import, for saving exported data to, or for accessing BLAST databases, should be configured as import/export directories, as described in section 7.1.

When an AWS Connection has been configured, data can also be imported from or exported to an AWS S3 bucket, as described in section 7.3.

3.3.1 CLC Server File System Locations

Data storage configuration is done via the 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 3.3.

Configuring CLC Server File System Locations, where data imported to the *CLC Server* or results generated using the *CLC Server* are usually stored, is done by going to:

Configuration ( ) | **Main configuration** ( ) | **File system locations**

See figure 3.3. File System Locations already configured are listed in this area. Those with a P in a blue box have permissions enabled.

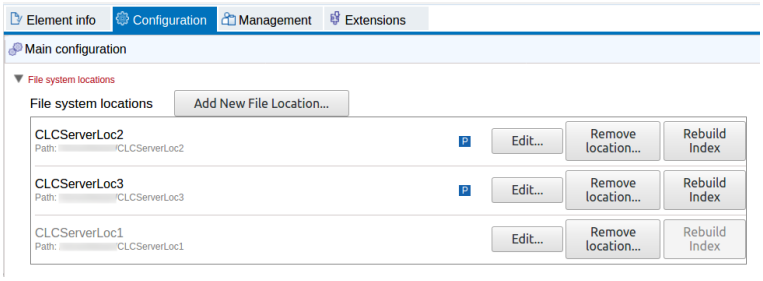


Figure 3.3: *Three CLC Server File system Locations have been configured. Two have permissions settings enabled, as indicated by the Ps in blue boxes. One location, CLCServerLoc1, is disabled, as indicated by its name being in light grey text.*

Each File System Locations that is enabled is included in the navigation area on the left of the web client. Interacting with data stored in these areas is described in section 8.1.

Adding and configuring CLC Server File System Locations

Add a new 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 CLC Server 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 3.3.2.

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 3.4. 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.

Permissions enabled When this box is checked, permissions can be configured for that location (figure 3.5).

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

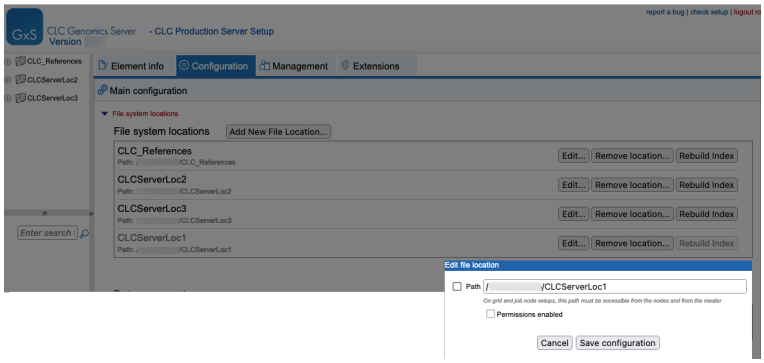


Figure 3.4: *When the checkbox to the left of the Path is unchecked, the file system location is disabled. This makes it unavailable to users. Under the Main Configuration tab, disabled locations are listed using light grey text.*

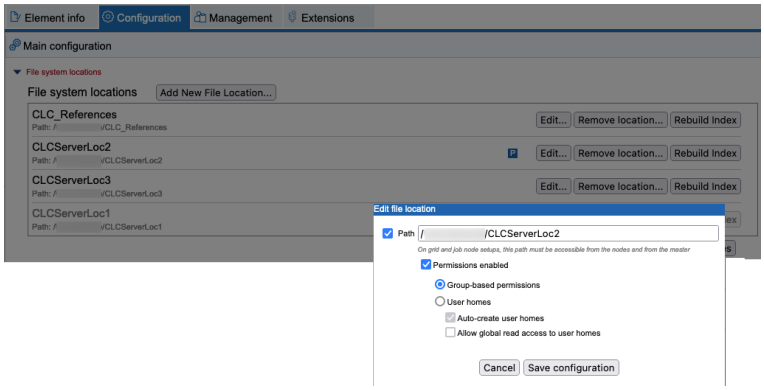


Figure 3.5: *A P in a blue box is shown the location listing for CLC Server File System Locations with permissions enabled, as seen here for CLCServerLoc2. When the Permissions enabled checkbox is checked in the Edit file location dialog, permissions settings are made visible.*

Note: Permissions can be configured for any file system location unless it has the name *CLC_References* (section 3.3.2).

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.
- A user logged into the *CLC Server* from a *CLC Workbench* will see their user home at the top of the list of user home directories. (Folders a user has access to are listed at the top of a given CLC location.)

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 in each CLC Server File System location of the elements in that location. This is used when searching for data. For more details, see section 8.2.1.

Important points about CLC Server File System Locations

- The directory on the file system configured as a CLC Server File System Location should be **dedicated for use** by, and should only be directly accessed only by, the *CLC Server*. Files should **not** be moved manually into this directory or areas under it, for example using standard operating system's command tools, drag and drop, and so on.
- The underlying file system must support file locking.
- Data written to CLC Server File System Locations is owned by the user that runs the *CLC Server* process.
- Areas designated as file system locations should **not** overlap. That is, a directory configured as a CLC Server File System Location should not be a subfolder of another directory configured as a CLC Server File System Locations. Overlapping 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 8.2.1.

File locations for job node set-ups

CLC Server File System Locations should be added **after** 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.

Job nodes write results directly to CLC Server File System Locations. Thus, the following is required for job nodes:

- Job nodes must have access to the directories configured as CLC Server File System Locations.
- The user running the *CLC Server* process on each job node must be the same as the user running the *CLC Server* process on the master node.

One relatively common problem 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.

Further details about job nodes is provided in chapter 6.

3.3.2 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 3.3.1).

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:

https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=References_management.html

3.3.3 Enabling and disabling internal compression of CLC data

By default, CLC format data is stored in an internally compressed format. An option is provided under the **Data compression** heading to turn off internal compression of data elements created by the *CLC Server*.

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

3.4 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 on the master node or single server.
- 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 is used 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).

The port that job nodes and grid nodes should use to communicate with the master is configured in the "Master node port" field under:

> **Configuration** ( ) | **Job processing** ( ) | **Server settings** | **Master and execution node settings**

In situations without other network settings (firewalls, port forwarding, etc.) that would affect this communication, the port number entered in the "Master node port" field is the port that was defined as the listening port. Please note that communication between a master and execution nodes uses HTTP.

Save the configuration to register changes to the server setup.

3.5 SSL and encryption

SSL communication can be configured for communication between the *CLC Server* and *client software* connecting to it (CLC Workbenches, CLC Server Command Line Tools, and the web client). This is particularly relevant for setups where the *CLC Server* is accessible to client software over the internet as well as on a local network.

Logging in to an SSL enabled *CLC Server* from a *CLC Workbench* is described in section 3.5. Logging in using the *CLC Server Command Line Tools* is described in section 3.5. The web

Important notes:

- The default configuration of the *CLC Server* does not use SSL.
- Communication between a master and execution nodes takes place over HTTP. SSL is not supported for this communication.

Enabling SSL on the CLC 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/Certificate_authorities`).

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 3.5).

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:

```
<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*:

```
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 3.6.

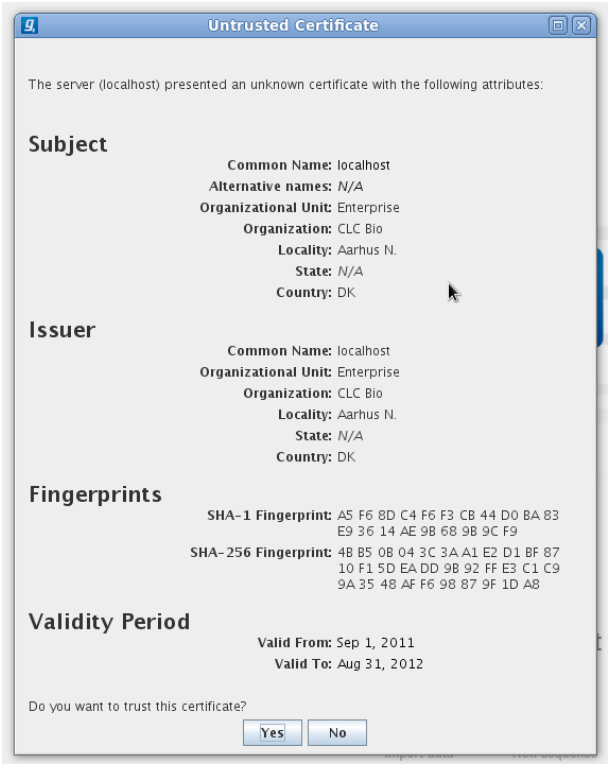


Figure 3.6: *A warning is shown when the certificate is not signed by a recognized CA.*

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 3.7.

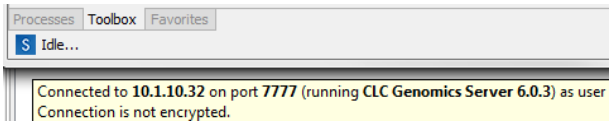


Figure 3.7: *Showing details on the server connection by placing the mouse on the globe.*

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

In order to trust the certificate the `clcsserversslstore` tool must be used:

```
./clcsserversslstore -S localhost -U root -W default -P 8443
The server (localhost) presented an untrusted certificate with the following attributes:
SUBJECT
=======
Common Name         : localhost
Alternative Names   : N/A
Organizational Unit: Enterprise
Organization        : CLC Bio
Locality            : Aarhus N.
State               : N/A
Country             : DK

ISSUER
=======
Common Name         : localhost
Organizational Unit: Enterprise
Organization        : CLC Bio
Locality            : Aarhus N.
State               : N/A
Country             : DK

FINGERPRINTS
============
SHA-1               : A5 F6 8D C4 F6 F3 CB 44 D0 BA 83 E9 36 14 AE 9B 68 9B 9C F9
SHA-256             : 4B B5 0B 04 3C 3A A1 E2 D1 BF 87 10 F1 5D EA DD 9B 92 FF E3 C1 C9 9A 35 48 AF F6 98 87 9F 1D A8

VALIDITY PERIOD
===============
Valid From          : Sep 1, 2011
Valid To            : Aug 31, 2012
Trust this certificate? [yn]
```

Once the certificate has been accepted, the `clcserver` program is allowed to connect to the *CLC Server*.

3.6 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.

3.6.1 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.

3.7 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.

3.8 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 6.3.9.

3.9 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).

3.10 External network connections

Some functionality available in CLC software requires access to specific addresses on the internet. A list of these addresses is at `https://qiagen.secure.force.com/KnowledgeBase/KnowledgeNavigatorPage?id=kA41i000000L5smCAC`.

Job nodes and grid nodes that will run jobs that include functionality listed on that page will need to have access to the relevant sites.

3.11 Proxy settings

Proxy settings for a *CLC Server* setup, are configured in one or more `.vmoptions` files, as described below.

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`.

[1]The method of using of a *cpu.properties* file to limit CPU usage on master nodes and job nodes is deprecated.

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 6.3.11.

Chapter 4

Managing users and groups

Before users start submitting jobs to the *CLC Server*, you should at a minimum set up user authentication (section 4.2) and CLC Server File Locations (section 3.3).

4.1 Changing the root password

The password for the *CLC Server* built-in root user should be changed (figure 4.1). The default credentials for this user are provided in section 3.1.

For single servers and master servers with grid nodes: We recommend that you are logged in as a user in the admin group other than root to change the root user's password. If you change the root user's password while logged in as the root user, log out and log in again right after changing the password. Putting the server into maintenance mode and letting jobs finish running before changing the root password is recommended.

For master servers with job nodes: When establishing a job node setup, it can be convenient to set up the job nodes and attach them *before* changing the root password and before changing the authentication mechanism. The root user's password should be changed while logged in as a user in the admin group other than root. Please refer to section 6.2.2 for further details.

To change the root user's password, go to

Configuration ( ) | Authentication ( ) | Change root password

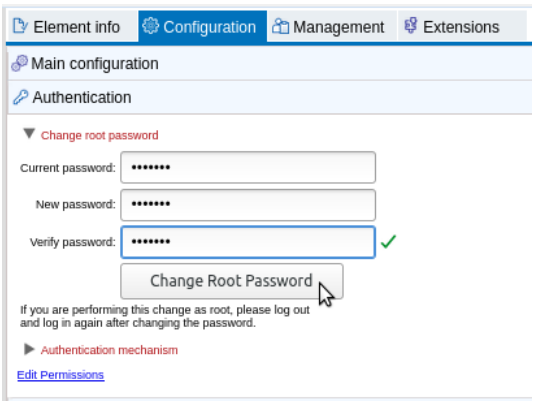


Figure 4.1: *Change the root user's password under the Authentication tab when logged in as an administrative user.*

4.2 User authentication for the CLC Server

On a single server or master node, log in as an administrative user. When first setting up the *CLC Server*, use the built-in administrative user, root. The default credentials for this user are provided in section 3.1.

Specify the user authentication mechanism to use under:

Configuration ( ) | Authentication ( ) | Authentication mechanism

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.

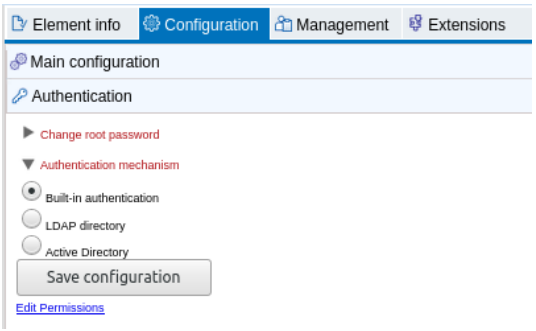


Figure 4.2: *Three modes of user authentication are available. Clicking on the Edit Permissions link at the bottom opens up the Permissions tab, where access to the CLC Server and its functionality can be configured.*

Irrespective of the authentication mechanism selected, we recommend that the password for a given user is not updated while jobs submitted by that user are running on the *CLC Server* (including execution nodes). The change in credentials could cause their jobs to fail, for example due to problems saving results. Similarly, removing a user's group membership while jobs are running could cause a problem saving results, depending on the permissions set on the CLC Server File System Location that the results are meant to be written to.

4.2.1 Giving users administration rights

Users that are members of a group specified as an administrative group with login rights to the *CLC Server* can configure and manage the *CLC Server*.

- **For built-in authentication**: Users should be added to the built-in **admin** group.
- **For Active Directory or LDAP** : A group should be designated in the **Admin group name** field. Any users who should have administrative rights on the *CLC Server* should be added to this group.

Administrative level access for some areas of the *CLC Server* can be granted to users that are not part of a group with administrative rights. See section 5.2 for details. If administrative access to just these areas is needed, this may be preferable to adding the user to an admin group, and thereby granting them full administrative control over the *CLC Server*.

4.2.2 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 4.2.3) 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.

Groups DN Provide the relative path for an OU in the domain to act as the root used by the *CLC Server*. The groups available for selection when setting permissions are limited to those in or below that OU.

Using `OU=bioinformatics, OU=researchers` as an example, the list of groups available when setting permissions would be limited to those contained in the OU `bioinformatics`, which is in the OU `researchers`, which is in the root of the domain. If the Groups DN field is left empty, all groups in the AD will be available for selection when setting group level permissions.

See section 5.1 and section 5.2 for further information about restricting access based on group membership.

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

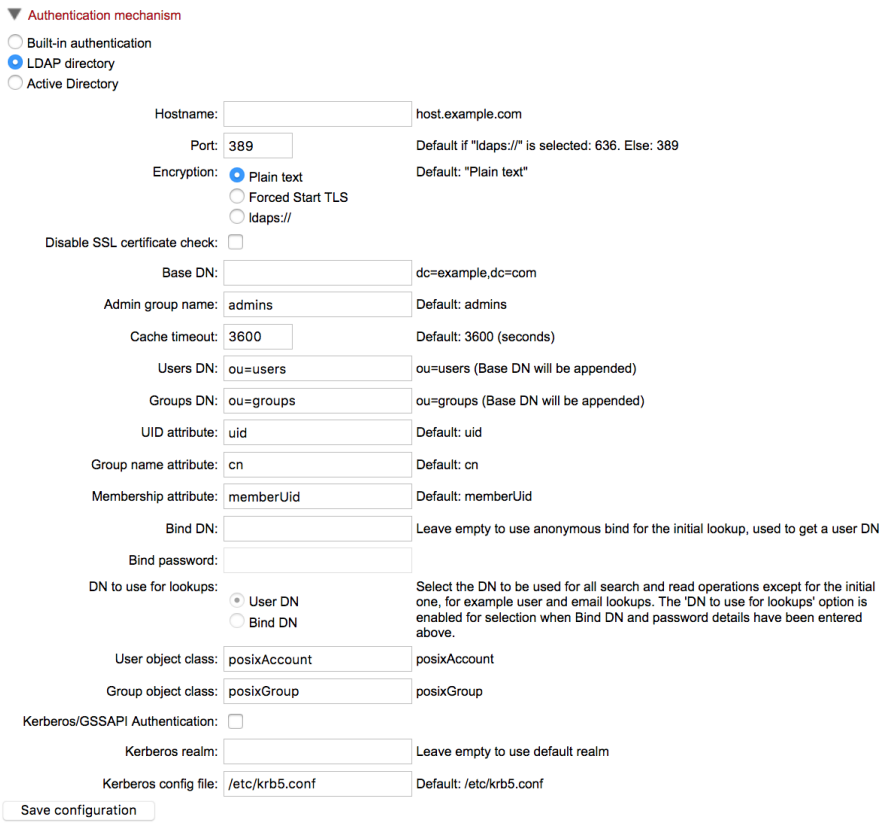


Figure 4.3: *LDAP settings panel*

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:

```
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 servers installation

location.

For setups with nodes, the following changes are also needed:

- For job node setups: Update the truststore of the CLC Server installation on each job node.
- For grid setups: Update the truststore for each grid worker. This can be done as described above, for each grid worker. The location of grid workers can be found in the configuration of the grid presets in the web administrative interface under **Configuration** ( ) | **Job processing** ( ) | **Server settings**.

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).

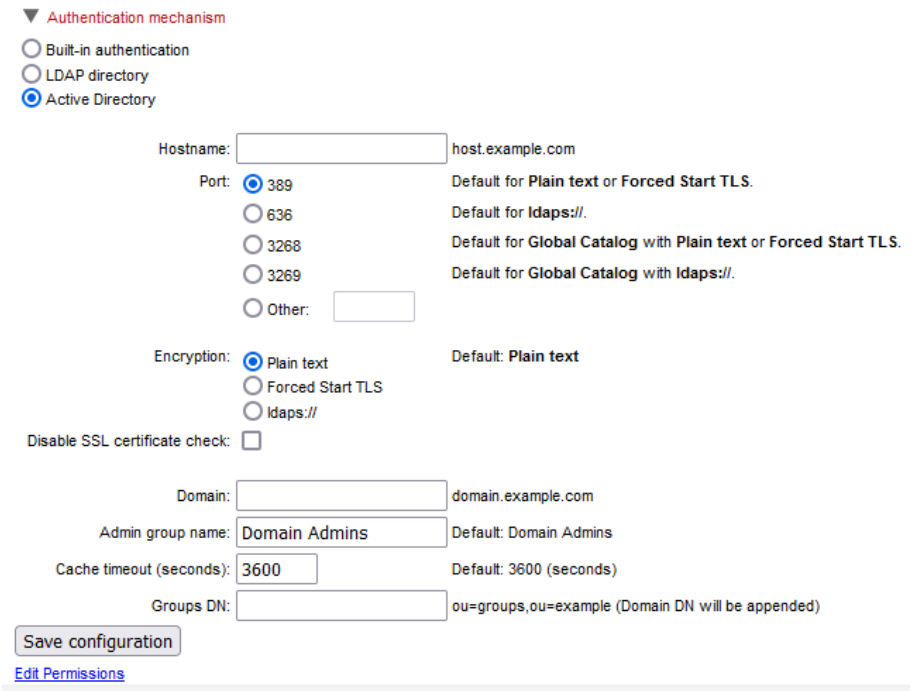


Figure 4.4: *Active Directory settings panel*

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 4.2.2) for other recommendations and configuration details.

4.2.3 Managing users and groups using built-in authentication

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

Managing users via the web 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.

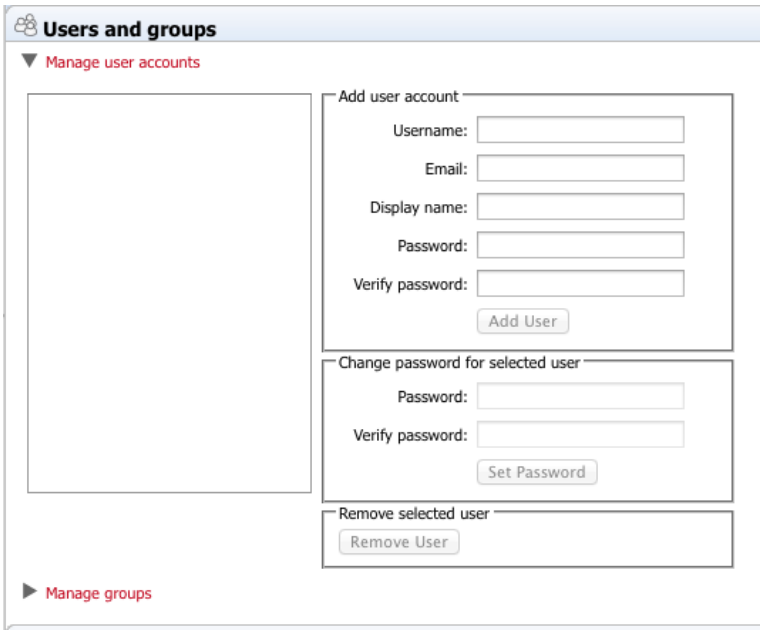


Figure 4.5: *Managing users.*

Managing groups via the web interface

To create or remove groups or change group membership for users when using built-in authentication, expand the **Manage gruops** 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*.

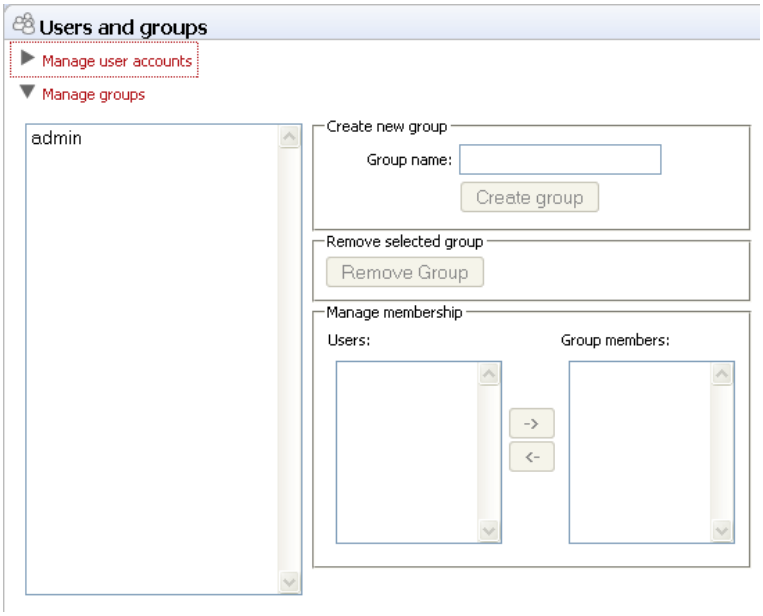


Figure 4.6: *Managing users.*

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.

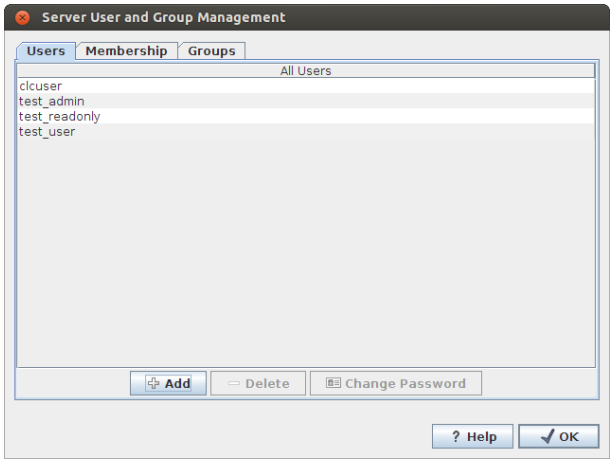


Figure 4.7: *Managing server users via a CLC Workbench*

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).

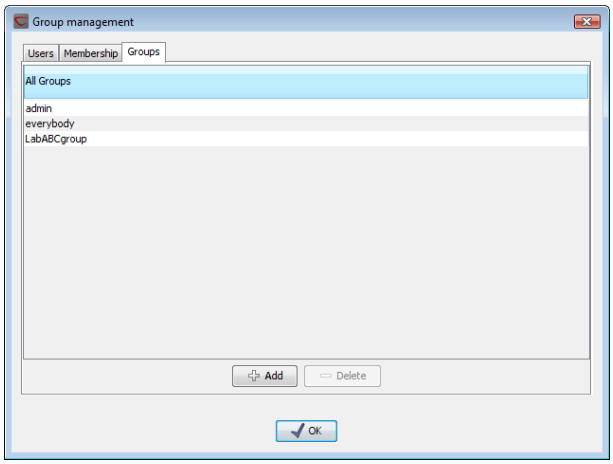


Figure 4.8: *Managing server groups via a CLC Workbench*

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

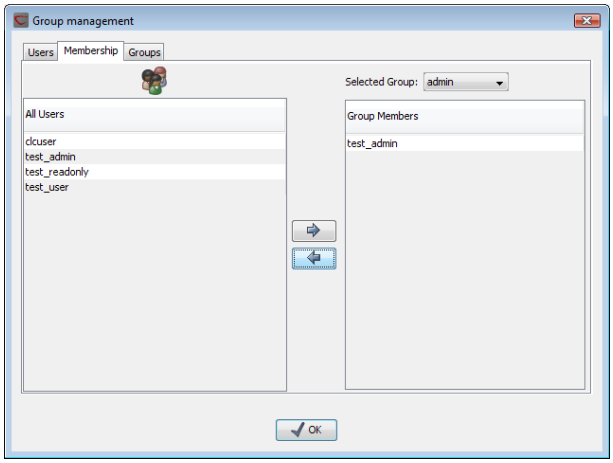


Figure 4.9: *Listing members of a group.*

To add users to or to remove users from a group, click on the **Add** ( ) or **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, access to analyses, external data and related functionality, 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 3.3.1. For information about Server File Locations with the "User homes" permission setting enabled, see section 3.3.1.

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** ( ) 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.

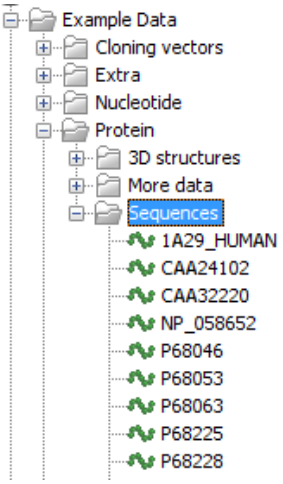


Figure 5.1: *A folder hierarchy as seen through a CLC Workbench Navigation Area.*

It is fine to give write access to just the final folder in a hiearchy. 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.

Folders that a user has access to are listed at the top of a given CLC location in the Navigation Area of a *CLC Workbench* when logged into the *CLC Server* (figure 5.2).

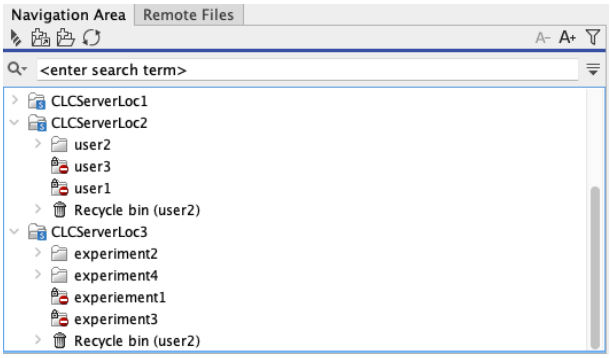


Figure 5.2: *A user called "user2" has logged into a CLC Server with 3 locations. Group-based permissions were enabled for the CLCServerLoc3 location and user2 belongs to a group with access to the experiment2 and experiment4 folders. Thus these are listed at the top of that location in the Workbench Navigation Area. The CLCServerLoc2 location was configured with the "User homes" option, so the user2 folder, accessible by this user, is listed at the top of that location.*

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 set on folders using the web administrative interface or using a *CLC Workbench* acting as a client for the *CLC Server*[1].

Setting group permissions on data folders using the web interface

[1] In *CLC Server* 21.x and earlier, setting group permissions could only be done using a *CLC Workbench*

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 5.3 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 be 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.

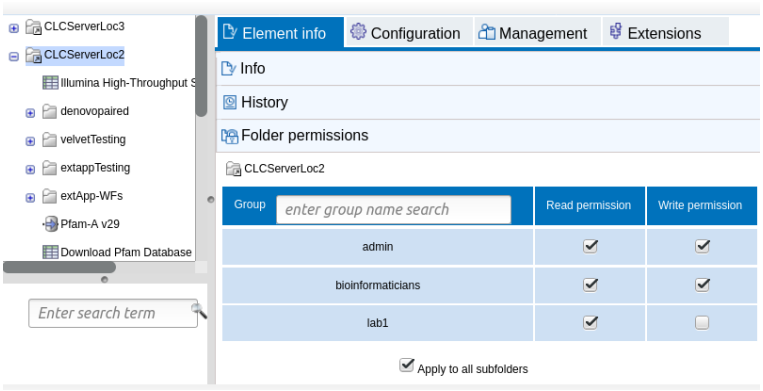


Figure 5.3: *Group permissions being applid to an entire file system location.*

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 (S)**

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

right-click the folder (📁) | **Permissions** (📁)

This will open the dialog shown in figure 5.4.

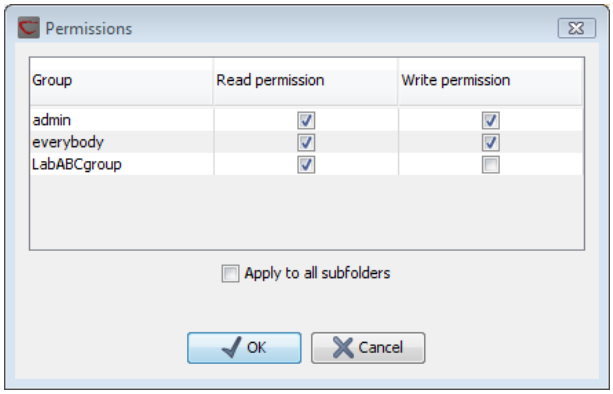


Figure 5.4: *Setting permissions on a folder.*

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 5.4. 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 5.5.

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 5.7). 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.

Figure 5.5: *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.*

- **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 processing tab), 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 5.7). 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.

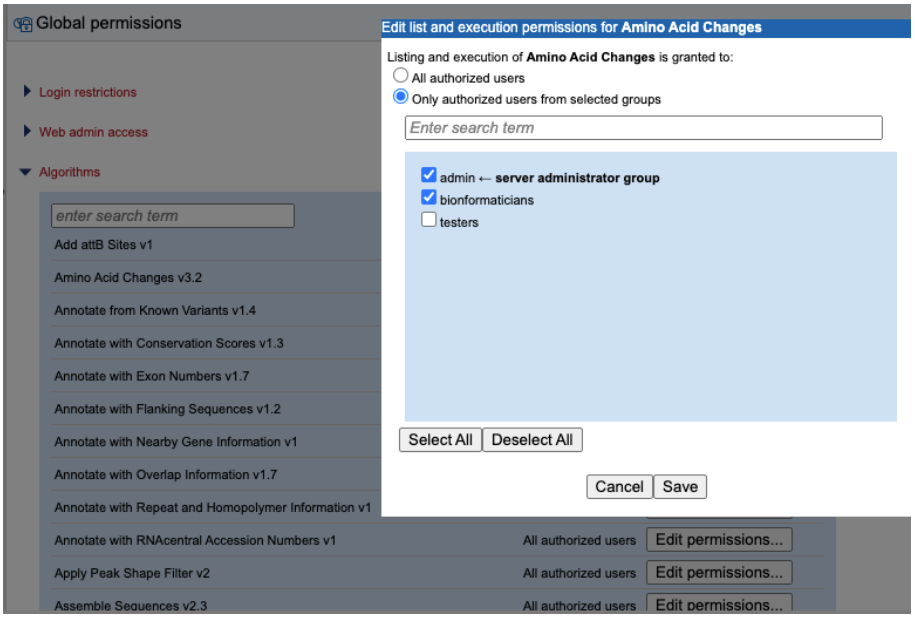


Figure 5.6: *Setting group-level permissions for tools is done under the Algorithms section. By default, all authorized users have access to all tools. Here, access has been limited to members of the admin and bioinformaticians groups.*

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 13.
- **External applications** Configure and administer external applications. See chapter 15.
- **Queue** See a list of the processes running on, or queued to be run on, the *CLC Server*. See chapter 12.
- **Workflows** Install and administer workflows on the *CLC Server*. See chapter 9.

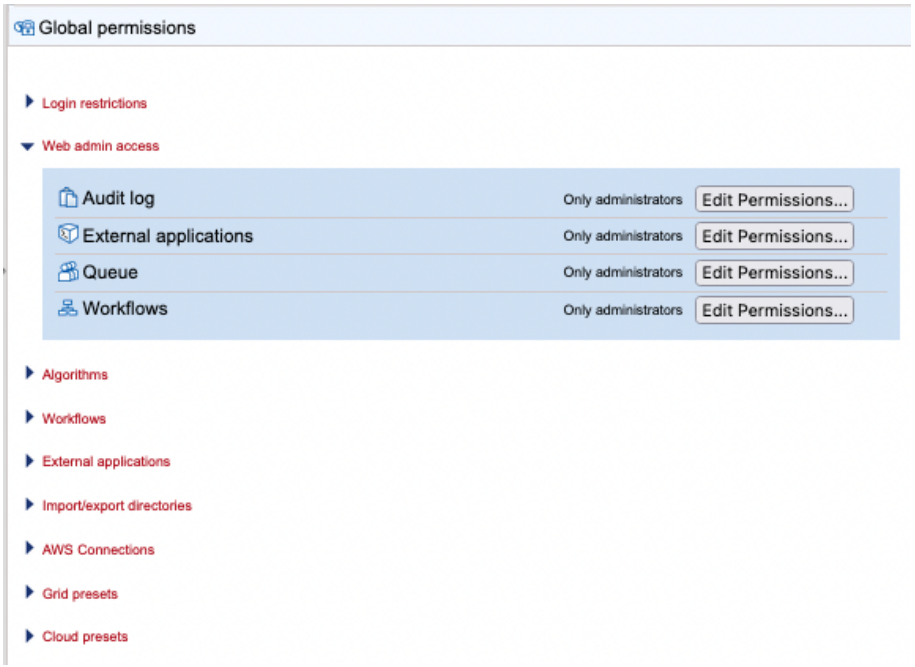


Figure 5.7: *Access can be granted to members of non-admin groups to the Audit log, External applications, Queue, and Workflows tabs in the web administriative interface.*

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 5.8.

Click the **Add Attribute** ( ) button to create a new attribute. This will display the dialog shown in figure 5.9.

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

[2] If the data location is a server location, you need to be a server administrator to do this.

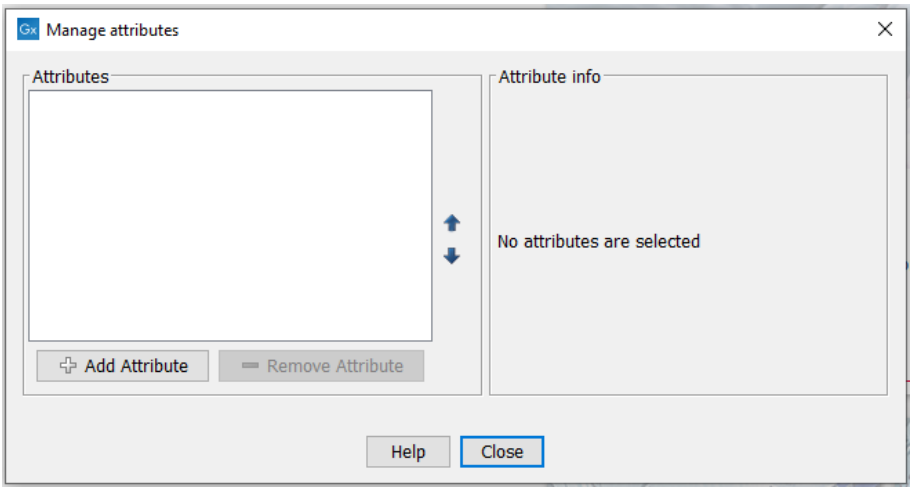


Figure 5.8: *Adding attributes.*

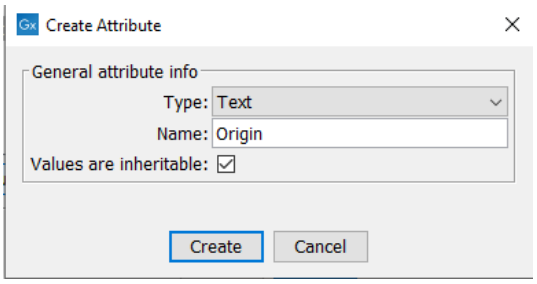


Figure 5.9: *The list of attribute types.*

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.
- **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 5.10).

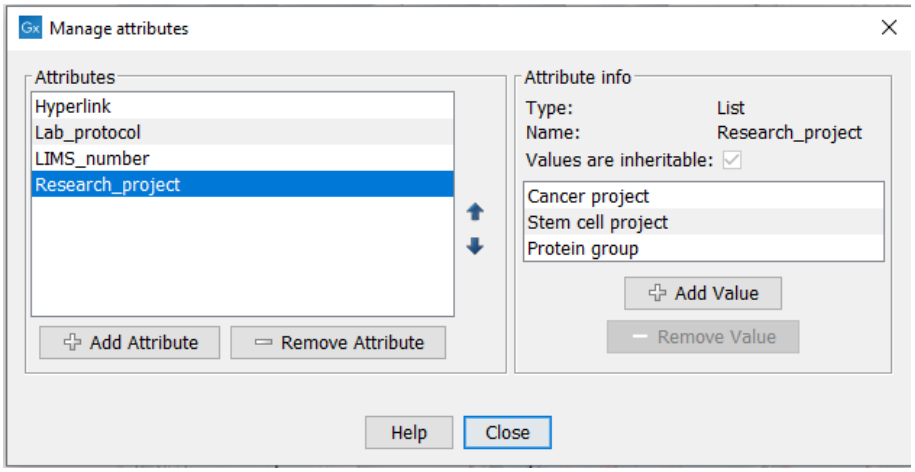


Figure 5.10: *Defining items in a list.*

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 5.11), the end users can start filling in information.

This is done in the element info view:

> **right-click a sequence or another element in the Navigation Area | Show ( ) | Element info ( )**

This will open a view similar to the one shown in figure 5.12.

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).

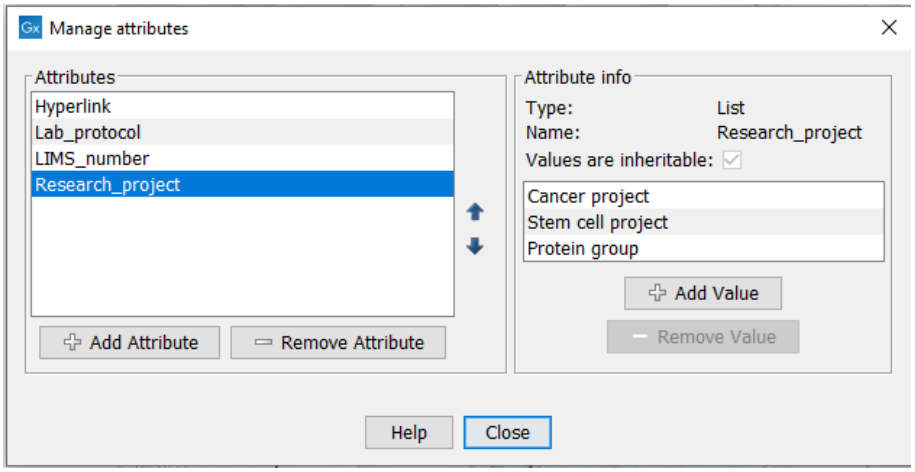


Figure 5.11: *A set of attributes defined in the attribute manager.*

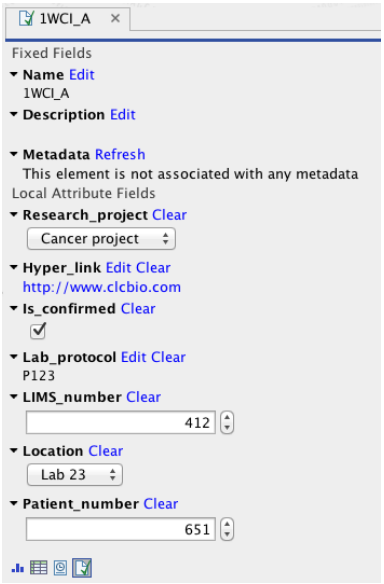


Figure 5.12: *Adding values to the attributes.*

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 5.13).

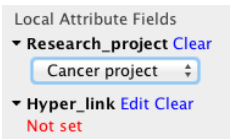


Figure 5.13: *An attribute which has not been set.*

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 5.13, 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 a quick way of changing the attributes of many elements in one go (see section **??**).

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.

Read more about search here: http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Local_search.html.

Chapter 6

Job processing

The *CLC Server* can be run as a single server, where administration and job execution are run on the same system, or it can be run as a master with execution nodes, where administration is done on the master server, and jobs are processed on execution nodes. These can be either nodes dedicated to running CLC software (job nodes) or nodes in an existing compute cluster, making use of a third party scheduler (grid nodes).

This chapter describes the available server setups, options relating to configuring execution nodes and options affecting how parts of workflows are distributed across nodes, where relevant.

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*. See section 6.2 for further details.
- **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. See section 6.3 for further details.
- **Model III: Single Server setup**. In this model, the master and execution node functionality is carried out by a single *CLC Server* instance.

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.

Note: Jobs can also be sent to AWS for execution via the *CLC Server*, regardless of the server setup model chosen. See chapter 16 for details.

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 interface of the *CLC Server* master node as a user with administrative rights. (See section 3.1.)
5. Configure the master node and attach the job nodes. (See section 6.2.3)

   Note that to attach job nodes, you must be logged in as an administrative user that is common to the master node and job node. For newly installed servers, this could be the default, built-in root user with default password. Further details are provided in section 6.2.2.

   The master pushes configuration settings to the attached job nodes, including plugins (see section 6.2.4).

The only configuration work done directly on job nodes is:

- Installing the *CLC Server* software.
- Starting up the *CLC Server* up on each node.
- Changing the built-in administrative user's password under certain circumstances (see section 6.2.2).

6.2.2 User credentials on a master-job node setup

For configurations to be pushed from the master node to job nodes, you must be logged in as a *CLC Server* administrative user common to the master node and the job nodes. The user logged into the master, when node addition and configuration changes are performed, is the same user that logs in to the job nodes to perform changes.

When initially installed, all instances of the *CLC Server* have the same credentials for the default admin user, root (see section 3.1). Logging in as the built-in root user using the default credentials is convenient when configuring a new setup. I.e. On a new setup, attach the job nodes *before* changing the root user's password and before changing the authentication mechanism.

Once the *CLC Server* software on all machines is up and running and the job nodes have been attached to the master, new users can be added, and passwords set, using the built-in authentications system, or a different authentication mechanism can be specified. The root user's password should be changed before the system is made available to users (details provided below).

To add a job node later, the administrative user you are logged in with must already be available, with the same credentials, on the job node to be attached. In many cases, the simplest route will be to log in as the *CLC Server* root user on the job node, which initially involves using the default password, and then change the root user's password to match that on the master node. The job node could now be attached when you are logged in as the root user on the master node.

Changing admin user passwords

Password changes must be done by a user with administrative privileges, and should generally be done on behalf of a different user, i.e. not by the same user that will be affected by the change being made.

For example, when changing the root user's password, log in as an administrative user other than root. This implies that before changing the root password, at least one user with administrative rights on the *CLC Server* should be available. Giving users administration rights is described in section 4.2.1.

While not recommended, if you do not wish to have any administrative user for the *CLC Server* other than the root user, then changing the root user's credentials involves detaching the job nodes, and then manually updating the the root password on each job node as well as the master node. Then log out of the master node, log back in using the new root password, and re-attach the job nodes. The master and job nodes should now all have the same password for the root user.

Background information: When you are logged in as the root user and then change the root user's password, the login session becomes invalid. On a single server, you can just log out and log in again using the new password. However, on a master node with job nodes attached, communication issues between the master and job nodes arise as the login session on the master node becomes invalid, and the job nodes and the master will no longer have the same root password information.

6.2.3 Configuring your setup

The information in this section assumes that on the machine that will act as the master, the *CLC Server* software is installed, the license has been downloaded (section 2.6), and the *CLC Server* has been restarted.

When logged in as a user in the admin group on the master node, go to:

Configuration ( ) | Job processing ( )

Open the **Server settings** section.

Under **Server setup** configure the following:

- **Server mode** - Select `MASTER_NODE` from the list of server modes.
- **CPU limit** - Enter 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.

Save these configuration changes, and then, under **Master and execution node settings** configure the following:

- **Master node host** - Enter the hostname that execution nodes should use to contact the master node. To see a list of host addresses and host names reported by the master node, click on the text "More suggestions".
- **Master node port** - Enter the port that execution nodes should use to contact the master node. This is usually 7777. The listening port of the master may be changed (see section or chapter 3.4). Communication between a master and execution nodes uses HTTP.

Save the configuration by clicking on the **Save Configuration** button under the Fairness factor setting to register the changes 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 below the **Fairness factor** setting.

Before attaching job nodes to the master, the *CLC Server* software will need to be installed and running on the machines that will act as job nodes. **Note:** Do *not* install license files on the job nodes or configure settings on the job nodes. The licensing information and other settings are taken from the master.

Still on the master node, do the following for each job node:

- Click on the **Attach Node** button to configure a new job node (figure 6.1).
- Enter information about the node in the dialog that is presented (figure 6.2):
  - **Host** The host name of the job node.
  - **Port** The port the node can be contacted on. This is normally 7777.
  - **Displayname** The name to show at the top of the web client for that node.
  - **CPU limit** The maximum number of CPUs 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. Further information about this setting is provided below.
  - **Manage Job Types...** (Optional) Specify the types of jobs that can be run on the node. By default, all tools can run on the node. This equates to selecting the "Any installed Server Command" option in the dialog. When the "Only selected Server Commands"

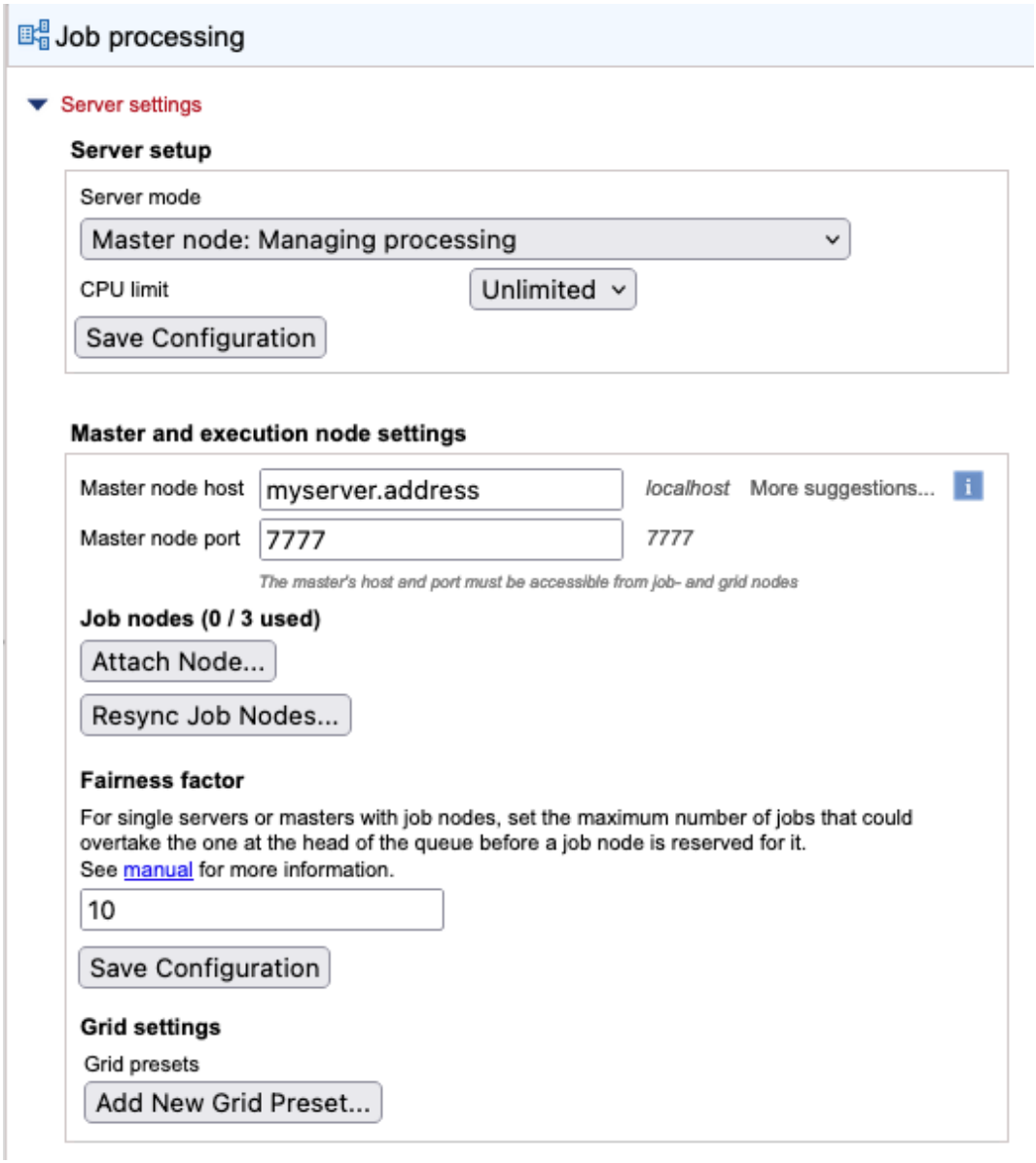


Figure 6.1: *When configuring a job node setup, the master node is configured, and then job nodes are attached by clicking on the "Attach Node..." button to attach job nodes.*

option is selected, a list of tools appears (figure 6.3). Select the tools that can be run on the job node. Enter text in the search field at the top to narrow down the list. Click on the **Modify** button to save this setting.

- Click on **Create** to save the job node settings.

Repeat this process for each job node and click on the **Save Configuration** button under the **Fairness factor** setting when you are done.

A warning dialog is presented if there are types of jobs that are not enabled on any of the nodes.

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

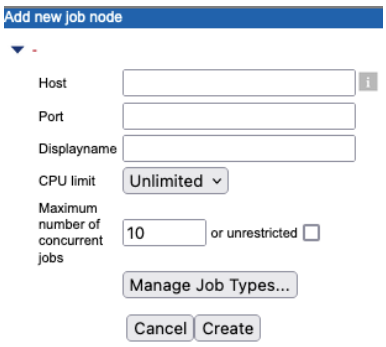


Figure 6.2: *Add new job node.*

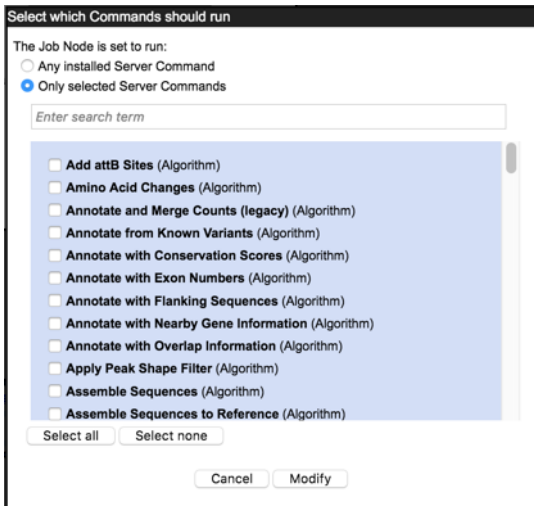


Figure 6.3: *Select Server Commands to run on the job node.*

To test that access works for both job nodes and the master node, click on the "check setup" link in the upper right corner of the web page (see section 19.1).

Job processing order on job nodes

When a node has finished a job, it takes the first job in the queue of a type the node is configured to process. Depending on how the system is configured, the job that is first in the queue will not necessarily be the next job processed.

The **Fairness factor** value is the number of times that a job in the queue can be overtaken by other jobs before resources are reserved for it to run. With the default value of 10, 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 the job at the head of the queue will not be overtaken by other jobs. See the information below about concurrent job processing, where the connection between job types and the fairness factor setting is described.

After updating the Fairness factor value, save the configuration by clicking on the **Save Configuration** button under that setting. This saves all changes made to settings in the "Master and execution node" section.

Concurrent job processing

There are three general categories of tools: non-exclusive, streaming and exclusive, described below. The **Maximum number of concurrent jobs** setting applies to tools in the non-exclusive and streaming categories only. Tools in the exclusive category are always run alone.

The maximum allowable value for the number of jobs that can be run concurrently is equal to the number of cores on the system. The default value is 10 or the number of cores on the system, whichever is lowest. If a CPU limit has been set, then the default is 10 or that CPU limit value, whichever is lowest.

The minimum value is 1, which equates to disabling multi-job processing.

Tool categories:

- **Non-exclusive algorithms** Tools with low demands on system resources. These can be run alongside other jobs in this category, as well with alongside a job in the streaming category. An example 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 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".

While non-exclusive algorithms are generally expected to have low demands on system resources, when working with very large genomes, setting a lower limit for the maximum number of concurrent jobs is worth considering.

See Appendix section 21.2 for a list of *CLC Genomics Server* algorithms that can be be run alongside others on a given machine.

Fairness factor and concurrent job processing 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.

Reminder: certain nodes can be reserved for use by only certain tools. See section 5.2.

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 6.1.

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 11.2.

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 11.2.

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 11.

Server plugin installation and removal is described in section 14.

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 processing tab on the master node.
- 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, third party, job scheduling software. When a job is started by the scheduler on a grid node in the local cluster, a stand-alone executable referred to as a **CLC Grid Worker** is started, and the CLC job sent by the user is executed.

Grid nodes must have enough memory and space to carry out the task(s) submitted to them. When a single CLC tool is run, that job runs in its entirety on a single grid node. How tasks within a workflow are executed (e.g. on a single node, or split across nodes) is configurable (see section 6.5.1).

Two locations beyond those needed for a single *CLC Server* setup need to be specified in grid setups. Both are configured within grid presets (see section 6.3.8). They are:

- **A CLC Grid Worker location** A CLC Grid Worker contains a JRE matching the master *CLC Server* JRE, as well as the plugins installed on the master, associated settings files, libraries, etc. These are extracted from the installation area of the *CLC Server* software and deployed to the location specified in a *grid preset* when the grid preset is saved.

- **A shared work directory** This directory is used for communication between the CLC Server and a CLC Grid Worker. Each grid job gets its own, temporary, sub-directory under this area where temporary files are written.

6.3.2 Requirements for CLC Grid Integration

- A grid submission system with a working DRMAA library must already be in place. Further details below.
- 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 must exist on all the grid nodes. This user is seen as the submitter of the grid jobs.
- *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 they must be 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 scheduling systems to be used to execute jobs submitted by a *CLC Server* must have:

- A working DRMAA library.
- A mechanism to limit the number of CLC jobs simultaneously running on the grid nodes to the number of CLC Grid Worker licenses (see below).

Grid integration has been verified 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 21.3, including information relating to compilation, where relevant.

Limiting CLC grid job number to the number of CLC Grid Worker licenses

The grid scheduling system must be configured to limit the number of CLC jobs simultaneously running on the grid nodes to the number of CLC Grid Worker licenses. Where more CLC jobs than this are launched, excess tasks should be 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 6.3.7. 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 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 6.4 shows an overview of the communication involved in running a job on the grid, using OGE as the example submission system.

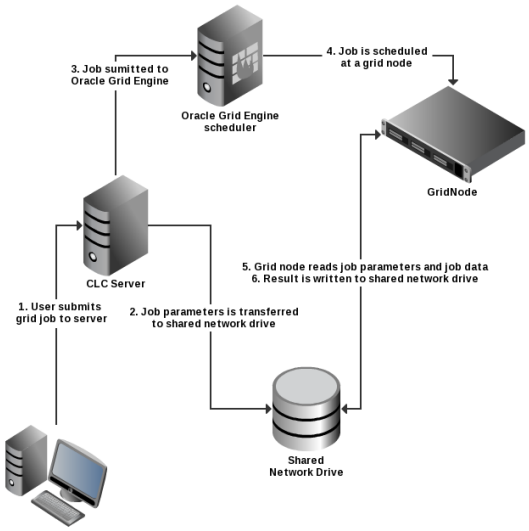


Figure 6.4: *An overview of grid integration, using OGE as the example submission system.*

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

The steps for configuring grid integration for a *CLC Server* are listed below, with a link to the manual section covering that topic in detail. These tasks primarily involve setting up licensing for **CLC Grid Workers** (aka "grid workers"), which are stand-alone executables used to run CLC jobs on grid nodes, and configuring grid presets, each of which includes the location of a CLC Grid Worker.

The steps below assume that the *CLC Server* software is already installed on the machine that is to act as the master, the license for the master has been downloaded section 2.6, and the *CLC Server* has been restarted.

Steps to setup grid integration:

1. Configure the *CLC Server* master node: section 6.3.5.
2. Set up grid worker licensing: section 6.3.6.
3. Configure the CLC grid licenses as a consumable resource in the local grid system: section 6.3.7.
4. Configure and save grid presets: section 6.3.8.
5. (Optional) Create and configure a clcgridworker.vmoptions file in each deployed CLC Grid Worker area: section 6.3.11.
6. 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*.

6.3.5 Configuring the master node for a grid setup

When logged in as a user in the admin group on the master node, go to:

Configuration | **Job processing**

Open the **Server settings** section.

Under **Server setup** configure the following:

- **Server mode** - Select `MASTER_NODE` from the list of server modes.
- **CPU limit** - Enter 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.

Save these configuration changes, and then, under **Master and execution node settings** configure the following:

- **Master node host** - Enter the hostname that execution nodes should use to contact the master node. To see a list of host addresses and host names reported by the master node, click on the text "More suggestions".
- **Master node port** - Enter the port that execution nodes should use to contact the master node. This is usually 7777. The listening port of the master may be changed (see section or chapter 3.4). Communication between a master and execution nodes uses HTTP.

Save the configuration by clicking on the **Save Configuration** button under the Fairness factor setting to register the changes made.

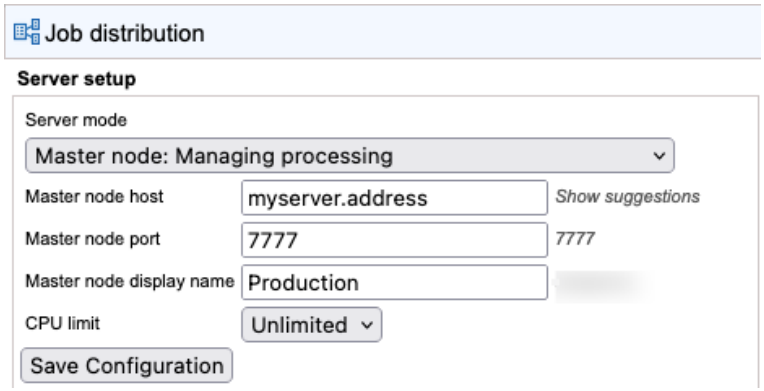


Figure 6.5: *Configuring a master node.*

6.3.6 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

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

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:

```
#License Settings

serverip=host.example.com
serverport=6200
disableborrow=false

autodiscover=false
useserver=true
```

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 6.3.8, 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.

6.3.7 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 21.4

6.3.8 Configuring 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 section 6.3.13). Each grid preset includes the location to deploy a CLC Grid Worker to. The CLC Grid Worker is used to run the CLC job on a grid node.

CLC Grid Worker version and deployment

Each time a grid preset is saved, all grid presets are (re)validated. This includes checking that the version of each CLC Grid Worker matches the *CLC Server* version installed on the master, and checking that all essential folders are present. If either of these checks fail, the CLC Grid Worker is redeployed. These checks are also carried out when the *CLC Server* is restarted.

When plugins are installed or removed from the master, the contents of each CLC Grid Worker's plugin installation directory are updated, ensuring the same plugins are present there as are present on the master.

Configuring grid presets

To configure a grid preset, go to:

Configuration ( ) | Job processing ( )

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

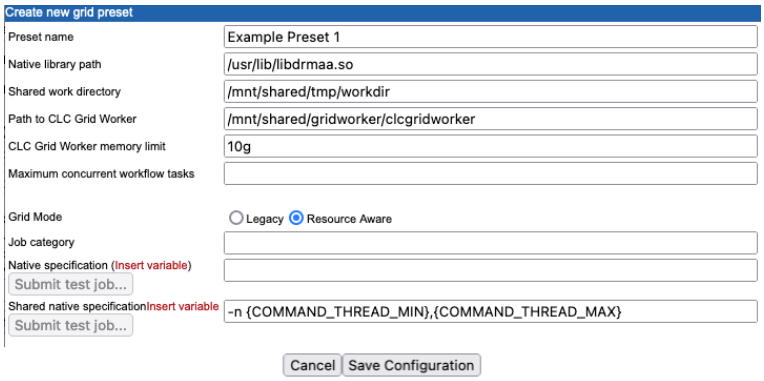


Figure 6.6: *The grid preset configuration form. The Shared native specification field is only visible when the Resource Aware grid mode is selected.*

Preset name The preset name will be specified by users when submitting a job to the grid. See section 6.3.13). 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.

Shared work directory 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.

Path to CLC Grid Worker 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.

A CLC Grid Worker includes a JRE matching the *CLC Server* JRE, associated settings files and essential folders (e.g. `clcgridworker`, `grid_cpa`, `jre`). It is extracted from the installation area of the *CLC Server* software and is deployed to the specified location when the grid preset is saved.

Multiple grid presets can refer to the same grid worker. Defining different grid worker locations in different presets results in multiple grid workers being deployed.

CLC Grid Worker memory limit The maximum amount of memory the CLC Grid Worker java process should use. If left blank, the default memory limit is used.

If providing a value in this field, provide a number followed by a unit, 'g', 'm' or 'k' for gigabytes, megabytes or kilobytes respectively. For example, "10g" limits the CLC Grid Worker java process to a maximum of 10 GB of memory.

Leaving this field blank, i.e. using the default value, should suit most circumstances. See section 6.3.15 for details.

Maximum concurrent workflow tasks The maximum number of concurrent jobs that can be run as part of a workflow execution on a given grid node. See section section 6.3.10 for further details about this setting.

Grid mode There are two grid modes for backwards compatibility reasons. The "Resource Aware" mode is generally recommended. Choosing this mode allows jobs that require few resources to run concurrently on a given node. For this, the field **Shared native specification** must also be configured. This is described further below. If "Legacy mode" is selected, all jobs submitted to the grid from the *CLC Server* will request use of the entire node. "Legacy Mode" is the default, but this may change in the future.

Job category The name of the job category - a parameter passed to the underlying grid system.

Native specification and Shared native specification 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 **Native specification** field contains the information to be passed to the grid scheduler for exclusive jobs, those where the whole execution node will be used.

The **Shared native specification** 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 jobs. This specification is only visible and configurable if the "Resource Aware" grid mode is selected.

See section 6.3.10 for details about configuring concurrent job execution.

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:

Grid mode	**Exclusive Jobs**	**Non-exclusive jobs**
Legacy	Native specification	NA - all jobs treated as exclusive
Resource aware	Native specification	Shared native specification

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.

```
-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:

```
-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 6.3.10.

6.3.9 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 6.3.8.

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 6.3.10.

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.

6.3.10 Multi-job processing on grid

Tools are categorized as *non-exclusive*, *streaming* or *exclusive*. Non-exclusive tools have reasonably low demands for system resources and can be run on the same grid node at the same time. Streaming tools and exclusive tools are run alone on a given grid node.

A list non-exclusive tools is in appendix section 21.2.

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 6.5.1, and the workflow or workflow block includes parallel non-exclusive tasks, then these may run concurrently on the grid node.

The maximum number of concurrent workflow tasks is a setting in each grid preset. The value provided here should align with the machines that jobs will be sent to using the preset. Note that values entered here cannot be validated directly.

Leaving this setting blank in a grid preset means the value for the master will be used. This value can be seen by switching the Server mode setting to "Single server". The default value is 10 or the number of cores on the master's system, whichever is lowest.

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 present in section 6.3.8.

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 6.3.8.

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 6.12 in section 6.5.1 for more detail about workflow blocks.

6.3.11 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 6.3.8.

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:

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

Proxy settings for grid setups are also configured in `.vmoptions` options files, as described in section 3.11.

Setting non-default memory limits for a **grid worker** is *not* done using settings in `.vmoptions` files. Memory limits are described in section 6.3.15.

6.3.12 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 6.7.

6.3.13 Client-side: submitting CLC jobs to the grid

Submitting jobs to grid setup from a CLC Workbench

When you log into a *CLC Server* configured with grid presets from a *CLC Workbench*, an option to submit to the **Grid via CLC Server** will be available in the launch wizards of tools and workflows

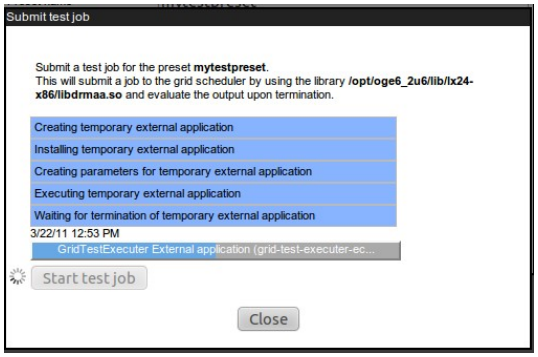


Figure 6.7: *Testing a Grid Preset.*

that can be run on the grid. A list of grid presets to submit the job to is available via a drop-down box (figure 6.8).

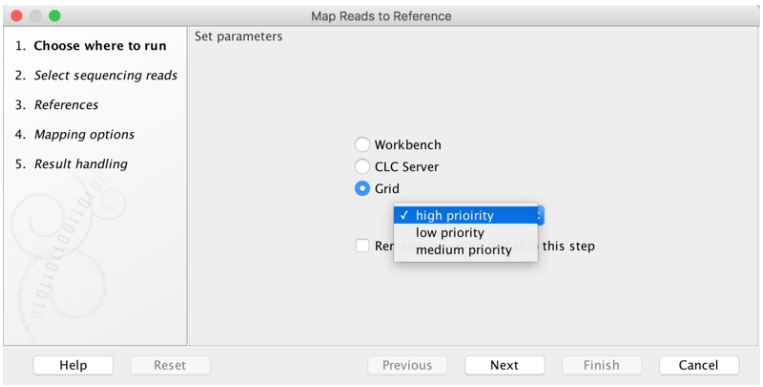


Figure 6.8: *Launching a job to run on a grid setup.*

Submitting jobs to a grid setup using the CLC Server Command Line Tools

To submit a job for execution on a grid setup using the `clcserver` command of the *CLC Server Command Line Tools*, add the -G parameter to the command followed by the name of an available grid preset. A list of presets is returned, along with other information, when the `clcserver` command is run with just the server address, port and user credentials specified. Further details 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`.

6.3.14 Grid Integration Tips

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

- Does your system meets 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 directories configured as CLC Server File System Locations. Please restart the server as the new user. You may need to re-index your CLC File System Locations after you restart the server (see section 8.2.1).
- 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 Serverr* 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.

6.3.15 Understanding memory settings

Java process memory limits

Most *CLC Server* work is done via its java process. The default memory limit for the java process is set according to the following rules:

- If virtual memory is limited (`ulimit -v`): set the limit to a quarter of that value or 50GB, whichever is smaller.

  Otherwise:
- If resident memory is limited (`ulimit -m`), set the limit to half of that value or 50GB, whichever is smaller.

  Otherwise:
- Set the limit to half of the amount of available physical memory or 50GB, whichever is smaller.

The default limit should suit most circumstances. However, there are cases where it may not, for example, where particular queues are designated for low overheads tasks, such as import jobs and trimming jobs, and for high overhead tasks, such as de novo assemblies or read mappings. In such cases, a memory limit can be explicitly configured for each **grid preset**[1]. This is described in section 6.3.8.

External binaries

Some tools use external binaries for one or more computational phases. These include, but are not limited to, those with a de novo assembly or mapping phases and those involving BLAST tools. 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.

[1] Directly specifying memory limits in a clcgridworker.vmoptions file using -Xmx options is not recommended. Such settings override those described on this page.

6.4 Model III: Single Server setup

In the Single Server setup, the *CLC Server* software is installed on a single machine. The master and execution node functionality is carried out by this single CLC Server instance.

The information in this section assumes that the *CLC Server* software is installed, the license has been downloaded (section 2.6), and the *CLC Server*has been restarted.

Log in to the web interface of the *CLC Server* as a user with administrative rights (see section 3.1).

Then go to:

Configuration ( ) | **Job processing ( )** | Server setup

and configure the following fields:

- **Server mode** Select `Single server` from the list of server modes (figure 6.9).
- **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.

  Further information about this settings is provided below.

Save the configuration to register the changes made.

The **Fairness factor** value is the number of times that a job in the queue can be overtaken by other jobs before resources are reserved for it to run. With the default value of 10, 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 the job at the head of the queue will not be overtaken by other jobs. See the information below about concurrent job processing, where the connection between job types and the fairness factor setting is described.

After updating the Fairness factor value, save the configuration by clicking on the **Save Configuration** button under that setting. This saves all changes made to settings in the "Master and execution node" section.

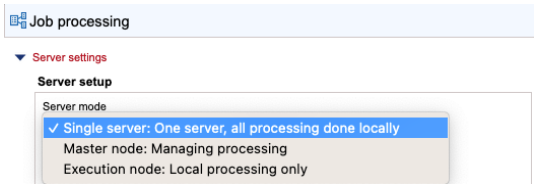


Figure 6.9: *Server mode options. Under normal circumstances, either Single server and Master node is selected. The Execution Node option is generally not manually selected except when troubleshooting.*

Changing the built-in root user's password is described in section 4.1.

Concurrent job processing

Figure 6.10: *Configuring a single server.*

There are three general categories of tools: non-exclusive, streaming and exclusive, described below. The **Maximum number of concurrent jobs** setting applies to tools in the non-exclusive and streaming categories only. Tools in the exclusive category are always run alone.

The maximum allowable value for the number of jobs that can be run concurrently is equal to the number of cores on the system. The default value is 10 or the number of cores on the system, whichever is lowest. If a CPU limit has been set, then the default is 10 or that CPU limit value, whichever is lowest.

The minimum value is 1, which equates to disabling multi-job processing.

Tool categories:

- **Non-exclusive algorithms** Tools with low demands on system resources. These can be run alongside other jobs in this category, as well with alongside a job in the streaming category. An example 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 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".

While non-exclusive algorithms are generally expected to have low demands on system resources, when working with very large genomes, setting a lower limit for the maximum number of concurrent jobs is worth considering.

See Appendix section 21.2 for a list of *CLC Genomics Server* algorithms that can be be run alongside others on a given machine.

Fairness factor and concurrent job processing 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.

6.5 Workflow settings

General workflow settings are found under:

Configuration | **Job processing** | **Workflow settings**

These consist of options for how the tasks of a workflow should be distributed on execution nodes (see section 6.5.1) and options for where intermediate workflow results should be stored (see section 6.5.3).

A general description of workflows and information on installing workflows on the *CLC Server* is provided in chapter 9.

6.5.1 Workflow distribution options

Workflow distribution options are found under:

Configuration | **Job processing** | **Workflow settings**

The three available options, described further below, 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 6.11.

  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 distribution options (figure 6.12) 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

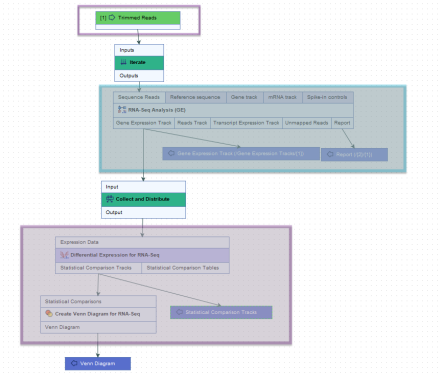


Figure 6.11: *This workflow has one iteration block, shaded in turquoise, and one block after the iteration block, shaded in purple. It also has an optional block above the iteration block, outlined in purple. If a user chooses to import files as the initial action taken when launching this workflow, 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 details about considerations when choosing a workflow distribution option are provided in section 6.5.2.

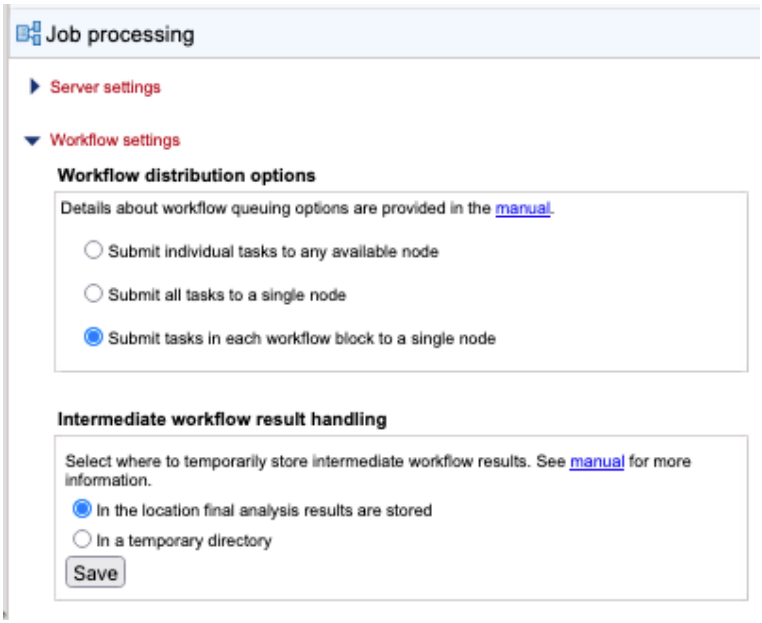


Figure 6.12: *Workflow distribution options determine how workflow tasks are distributed on servers with execution nodes.*

6.5.2 Choosing a workflow distribution 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 distribution 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 distribution 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 wtih 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.

6.5.3 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*: they are needed downstream analysis steps but are then deleted when the analysis successfully completes. Where these intermediate results should be saved is controlled using the **Intermediate workflow result handling** options (figure 6.13):

- **In the location final analysis results are stored** Intermediate results are stored in a dedicated subfolder of the folder that is selected to store outputs 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.

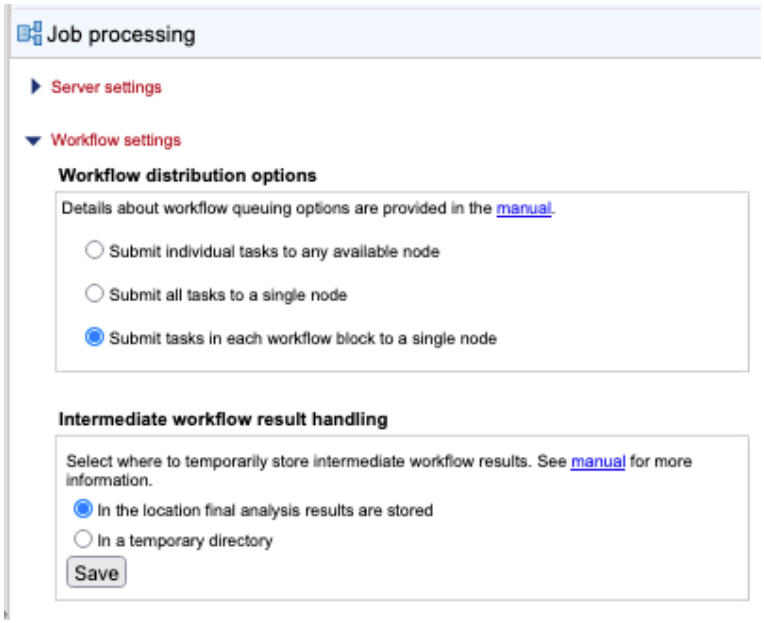


Figure 6.13: *Intermediate workflow result handling options.*

Chapter 7

Working with external data locations

Locations external to the *CLC Server* that it should have access to read and/or write data need to be configured. Such locations include import/export directories, which are typically on locally accessible network file systems, as well as AWS S3 (figure 7.1).

Access to external locations is configured under:

Configuration ( ) | **External data** ( )

Access to BaseSpace does not require explicit configuration in the *CLC Server* (see section 7.6).

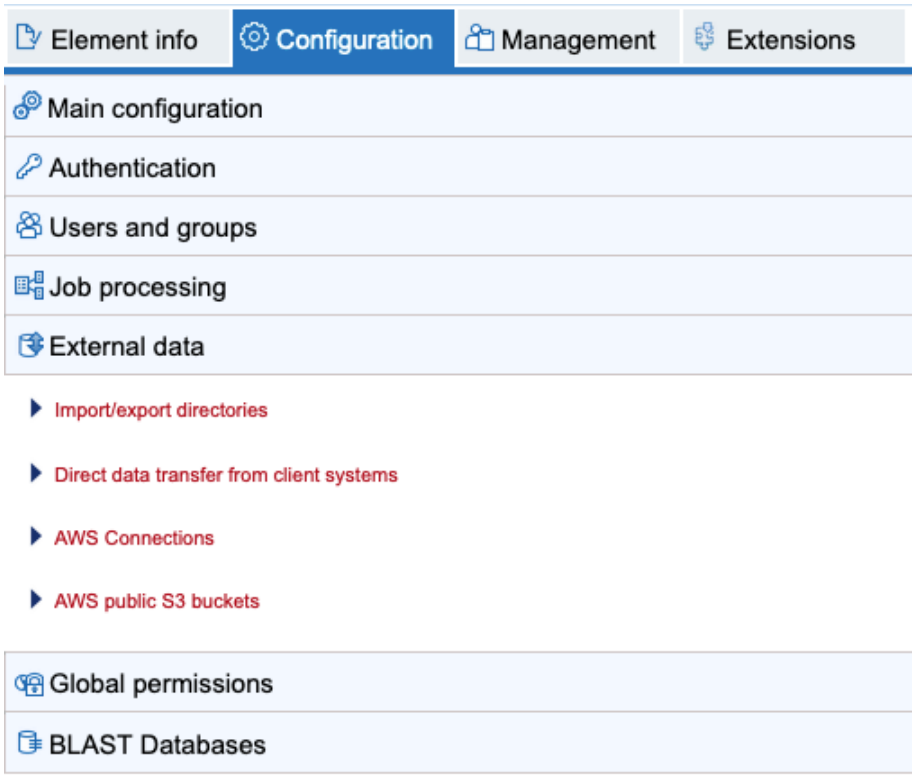


Figure 7.1: *Configuration of data locations external to the CLC Server is done under the External data tab.*

7.1 Import/export directories

Import/export directories are areas designated for use by the *CLC Server* to write to, for example when exporting results, or to read from, for example when importing data or when accessing BLAST databases.

To configure an import/export directory, go to:

Configuration ( ) | **External data** ( ) | **Import/export directories**

Click on the **Add new import/export directory** button and enter the relevant path (figure 7.2). The specified folder and its subfolders will then be available to client software logged into the *CLC Server* when relevant activities are carried out.

At least one import/export directory must be configured to enable direct data transfer from client systems (see section 7.2).

Access to import/export directories can be restricted using settings under:

Configuration ( ) | **Global permissions** ( ) | **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.

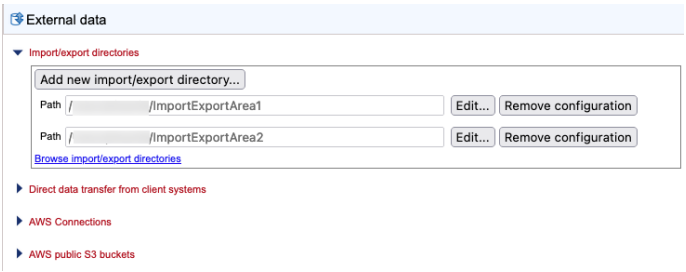


Figure 7.2: *Areas external to the CLC Server that it can read data from and write data to, are configured as import/export directories in the External data area.*

Browsing import/export areas

The contents of import/export directories can be browsed by going to:

Element info ( ) | **Browse server import/export directories** ( )

When a folder or file is selected, information about it is displayed below the file browser (figure 7.3).

URLs for files in import/export locations can be copied using the **Copy to clipboard** button. They can be useful when copying data to AWS S3 locations (see section 7.5), when sharing the location of files with others that have access to the same system, and when specifying files to be imported, or locations to export to, using the *CLC Server Command Line Tools*.

Some data management tasks can be carried out from in this area. Folders or files can be deleted, new folders can be created, and files can be copied from AWS S3 locations or CLC File Server Locations into a selected folder by pasting a file URL into the text field near the bottom, replacing the *Paste URL here...* text.

URLs for data in AWS S3 buckets are shown when browsing under **Element info** ( ) | **Browse S3 locations** ( ).

Data from a CLC File Server Locatios is normally saved to an import/export directory when using export functionality. See `https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Data_export.html` for details. URLs that can be used to copy data directly from a CLC Server File System Location are shown when browsing CLC data in the web client under **Element info** ( ) | **Info** ( ) and can also be obtained by using the Copy function in the Navigation Area of a Workbench.

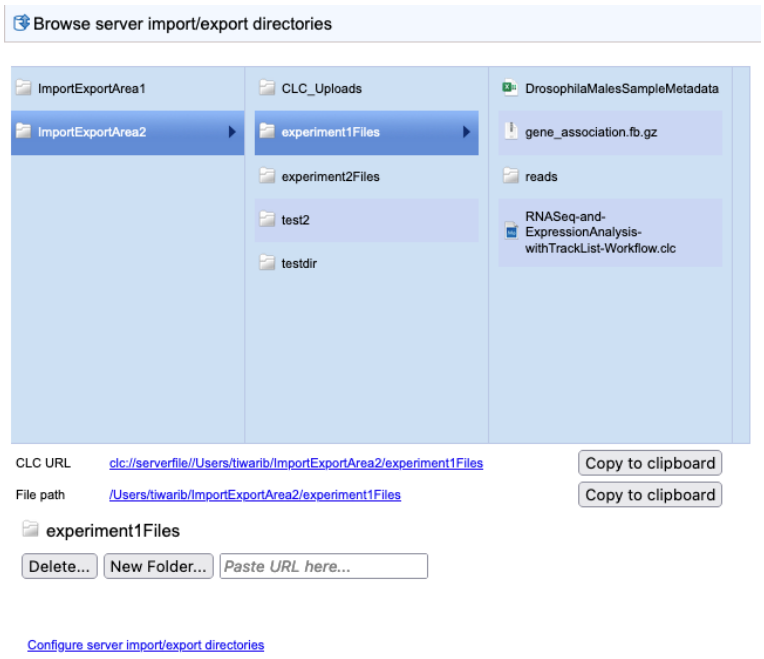


Figure 7.3: *Browse CLC server import/export directories.*

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). To enable direct data transfer from client systems, the option **Files uploaded via**

Import/Export location must be selected under:

Configuration ( ) | **External data** ( ) | **Direct data transfer from client systems**

For this option to be available, at least one import/export directory must be configured (see section 7.1).

When enabled, users of client systems can select data on their local file system for import. The selected files are initially uploaded to the designated import/export directory and then imported.

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.)

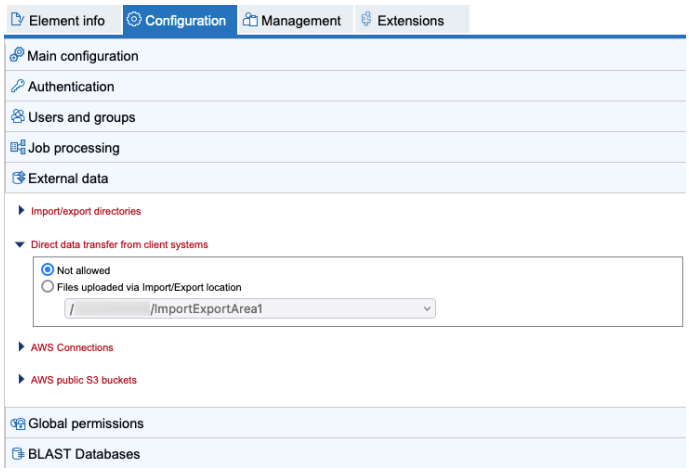


Figure 7.4: *Transfer of data directly from client software is not allowed by default. If at least one import/export directory has been configured, direct data transfer can be enabled by selecting the "Files uploaded via Import/Export location" option and specifying the import/export directory to use.).*

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.

Figure 7.5: *Configure AWS Connections. A green dot in the S3 column indicates a valid connection for accessing S3 buckets. A green dot in the CGC column indicates a valid connection to a CLC Genomics Cloud setup, meaning analyses can be submitted to run on AWS via this CLC Server.*

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. See section 7.5 for further details about this.

7.4 AWS S3 public buckets

Access to AWS S3 public buckets is configured under

Configuration | **External data** | **AWS public S3 buckets**

This requires only the name of the bucket (figure 7.6).

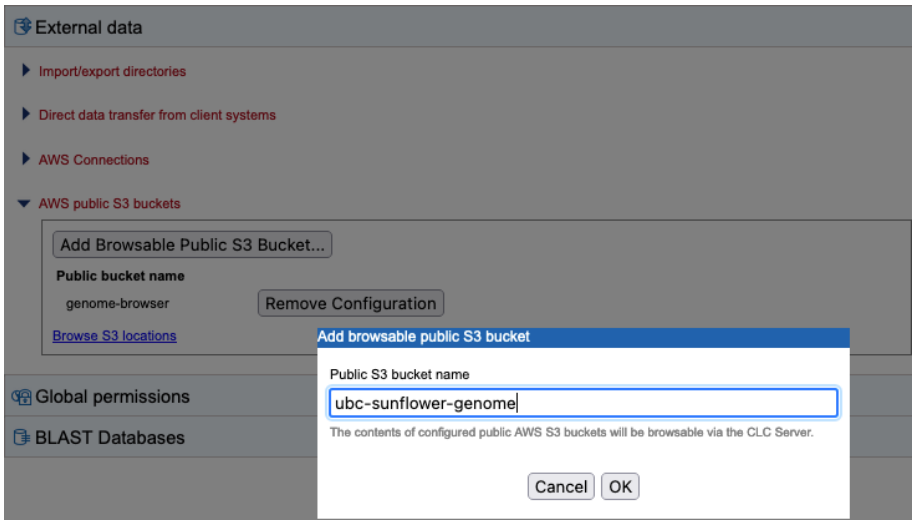


Figure 7.6: *Clicking on the Add Browsable Public S3 Bucket button opens a dialog where the name of a public S3 bucket can be entered.*

Clicking on the **Browse S3 locations** link in the configuration area opens the relevant tab under Element info (figure 7.7). See section 7.5 for further details about this.

Figure 7.7: *The contents of public S3 buckets configured in the CLC Server can be browsed and worked with under the Element info | Browse S3 locations tab.*

7.5 Browse AWS S3 locations

The contents of AWS S3 buckets you have access to can be browsed by going to:

Element info ( ) | **Browse S3 locations** ( ) .

If the message "No active S3 locations found." is visible, it means that either no AWS Connections or AWS S3 public buckets have been configured, or that there is no access for the user logged in. Configuring AWS access is described in section 7.3 and section 7.4.

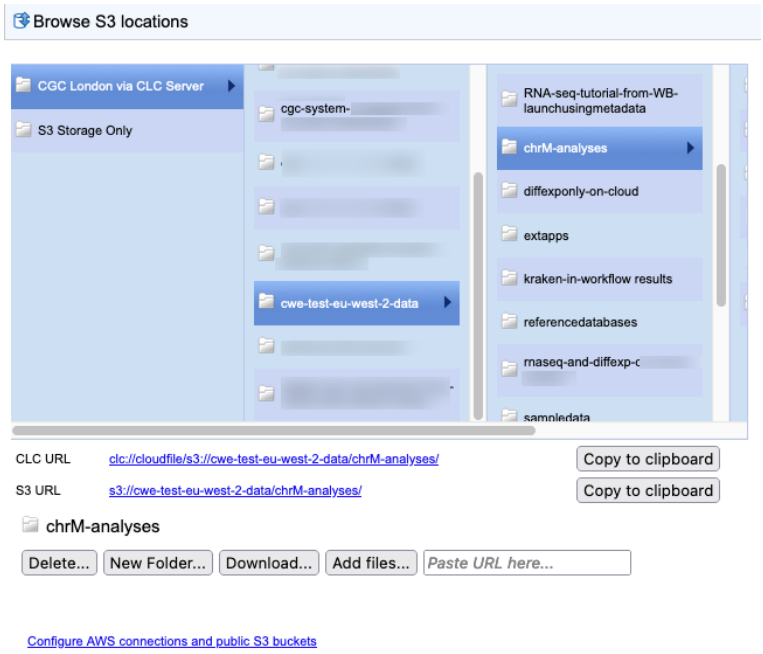


Figure 7.8: *Browse S3 buckets accessible via the CLC Server. Data can be uploaded, downloaded, deleted and new folders created using functionality in this tab.*

When an item in an S3 bucket is selected, URLs to that item are shown below the file listing area and relevant options are enabled (figure 7.8).

The URLs can be copied using the **Copy to clipboard** button. They can be useful when copying data to import/export directories (see section 7.1), when sharing the location of files with others, and when specifying inputs and locations to save to using the *CLC Server Command Line Tools*.

The actions available are:

- **Delete files or folders** If a folder has been selected, clicking on the **Delete...** button opens a dialog offering the option to delete the folder and its contents, or to specify individual files from that folder to be deleted. You are asked for confirmation of your intention to proceed with deletion.
- **Add a new folder** Click on the **New Folder...** button to create a new folder within the folder selected in the browsing area.
- **Download files or folders** Data can be downloaded to a CLC File System Location or to an import/export directory.

If a folder has been selected, clicking on the **Download...** button opens a dialog with the options **Download folder...**, to download the folder and its contents, and **Select files...**, to select individual files in that folder for download (figure 7.9).

When selecting individual files, the files in the selected folder are listed, along with their sizes, and the total download size of the selected files (figure 7.10). After download, a list of the downloaded files is presented under the Results area at the bottom of that dialog.

Download starts when the location to download to has been selected.

CLC data downloaded to a CLC location, can be opened using a *CLC Workbench* connected to the *CLC Server* and used in downstream analyses.

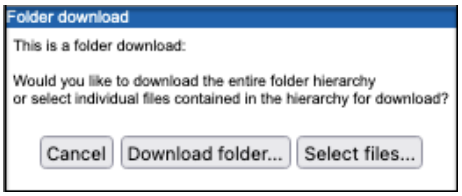


Figure 7.9: *If a folder is selected when the Download... button is clicked, you get the option to download the folder (and its contents) or to select files from within the folder to download.*

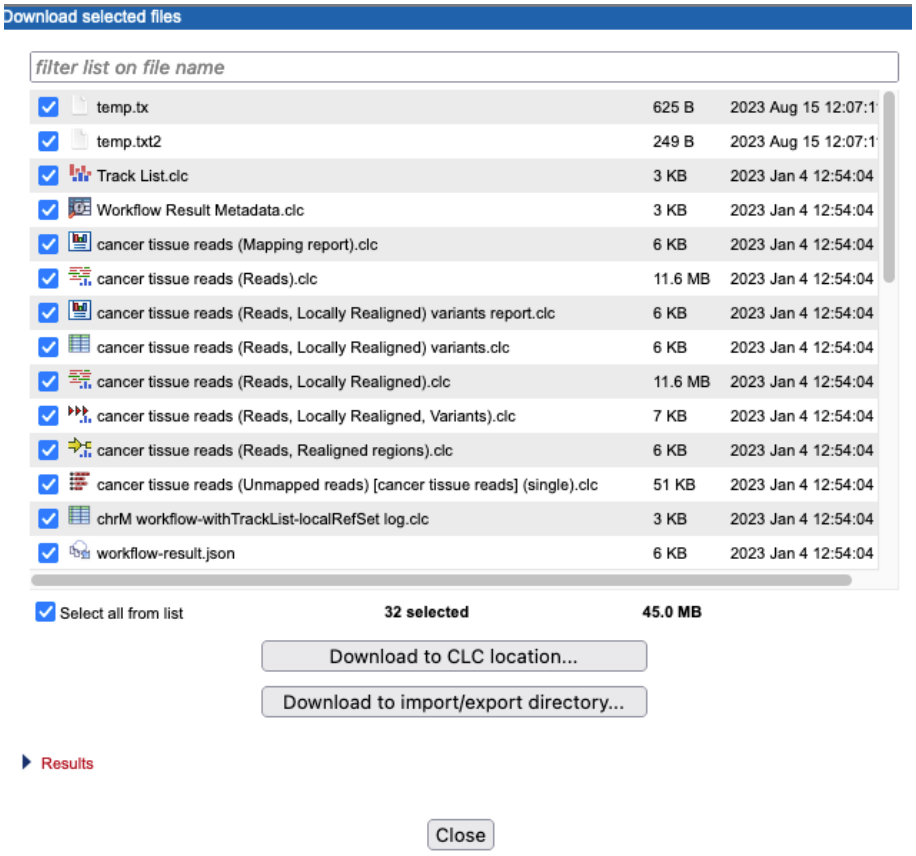


Figure 7.10: *When downloading files from S3, a list of the selected files and the total size of the download is displayed. You will be prompted for a specific location to download to.*

Note: AWS charges for downloading data from AWS S3.

- **Upload data**

  Data from a CLC File System Location, from an import/export directory or from AWS S3 can be uploaded to the selected folder:

  There are two ways to indicate the data to upload:

  - **Using a graphical file chooser** Click on the **Add files...** button to select files and folders using a graphical file browser. The first step will be to specify the source of the data to upload (figure7.11).

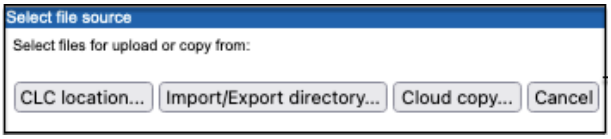


Figure 7.11: *Choose the source of data to upload to AWS S3.*

    A graphical file browser will open allowing selection of the files and/or folders to upload.

  - **Providing a URL** URLs for data in a CLC Server File System Location are shown when browsing CLC data in the web client under **Element info** ( ) | **Info** ( ) and can also be obtained by using the Copy function in the Navigation Area of a Workbench.

    URLs for data in an import/export area are shown when browsing import/export directories under **Element info** ( ) | **Browse server import/export directories** ( ).

    URLs for data in AWS S3 buckets are available when browsing using the web client, as described on this page.

The actions that can be taken in practice depend on the permissions for accessing the selected AWS S3 bucket and settings in the *CLC Server* for the user logged in.

Note: If you are using a CLC Genomics Cloud setup 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.

7.6 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.

Optionally, a regional instance ID and client credentials can be configured in the *CLC Workbench* Preferences. See `https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Advanced_preferences.html`

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

Chapter 8

Working with CLC Server File System Locations

Sections in this chapter describe functionality in the *CLC Server* web interface for browsing CLC data, viewing data logs and data history, and searching for data in CLC Server File System Locations.

For working with the same data via the Navigation Area of a *CLC Workbench* connected to the *CLC Server*, see https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Navigation_Area.html.

8.1 Browsing CLC Server File System Locations

A navigation area showing the configured CLC File System Locations is available in the left side of the web interface (figure 8.1). When an element or folder is selected there, information about the selected item is displayed under

Element info ( ) | **Info** ( )

The CLC URLs at the bottom are direct links to the element. These can be shared with others with access to this system, who can then refer to the data via URL. The 2 URL forms refer to the same data element. The human readable form will change if you move or rename the element, while the other form will remain the same as long as the element remains in the same CLC File System Location.

When the element selected is a log file, contents of the log are also displayed in this area.

Various actions are available, including moving data to the recycling bin, as described in chapter 17.

Viewing CLC data history

When an element is selected, its history can be seen under

Element info ( ) | **History** ( )

See figure 8.2.

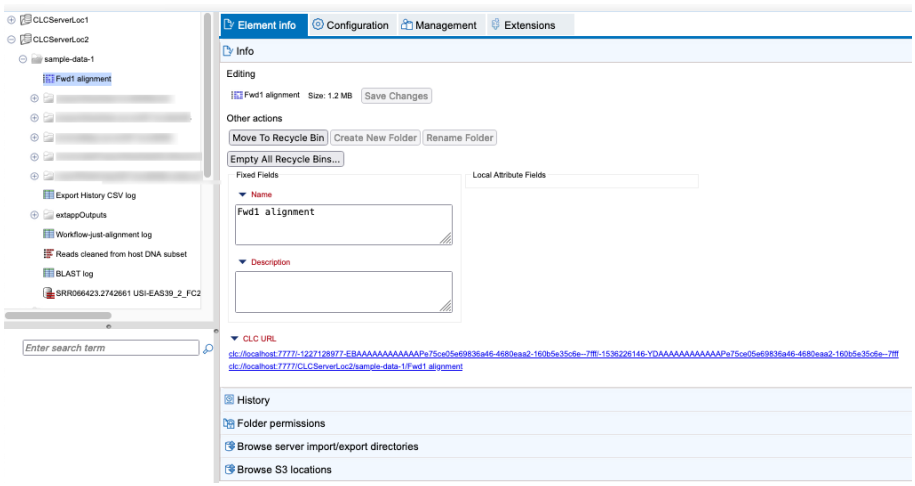


Figure 8.1: *CLC File System Locations are listed in the left side of the web interface. Information about a selected element or folder can be seen in the Info tab to the right.*

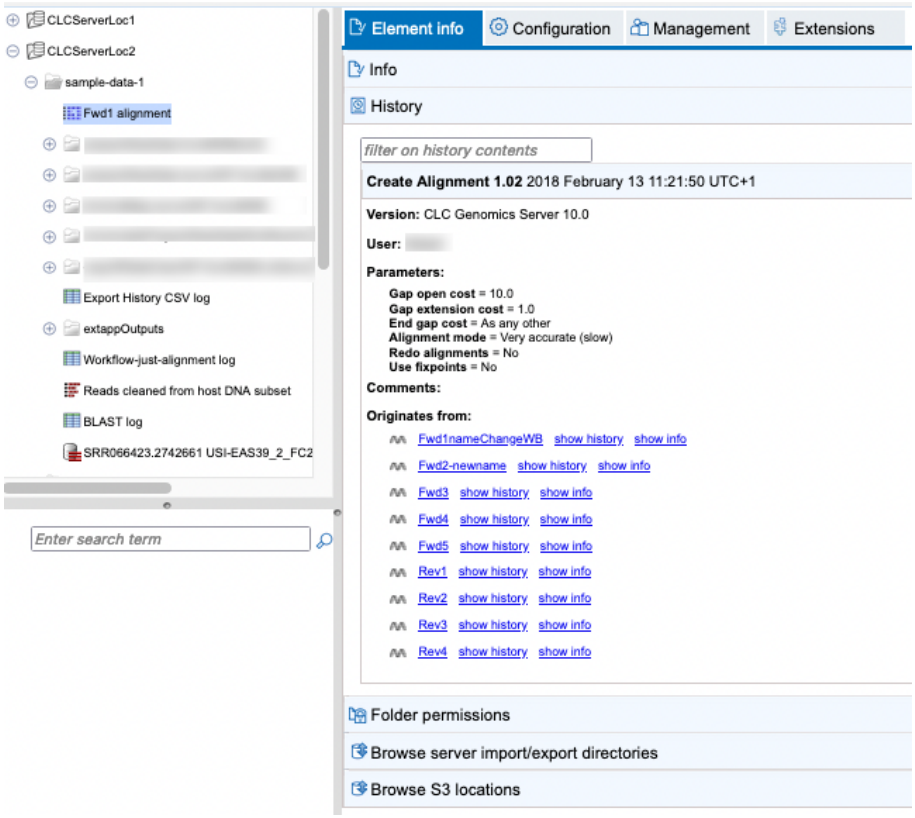


Figure 8.2: *The history of a selected element can be seen in the History tab.*

8.2 Searching CLC Server File System Locations

The search field under the *CLC Server* locations on the left of the web interface can be used to search in available CLC Server File System 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 8.3).

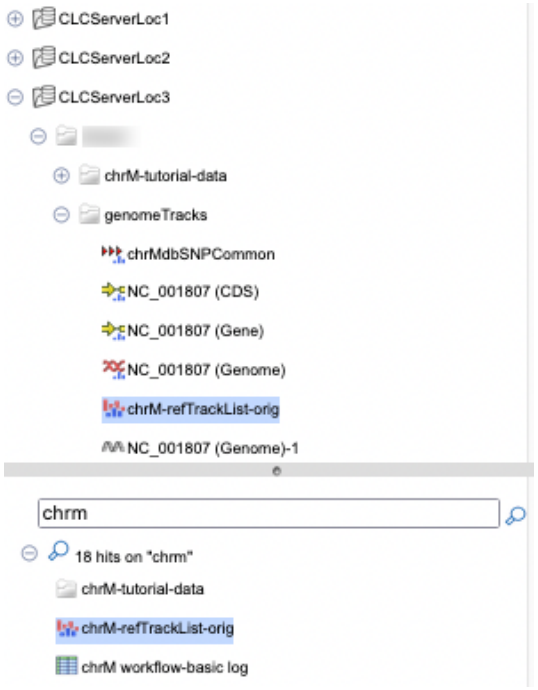


Figure 8.3: *A search was run for the term chrm, and 18 results were returned (only some are shown in this image). Clicking on one of the results in the list at the bottom revealed its location in the browser area at the top.*

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 8.2.1.

8.2.1 Rebuilding the index

The *CLC Server* maintains an index of all the elements for each CLC Server File System 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 CLC Server File System 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 8.4).

Note: Rebuilding an index can take a long time for locations with a lot of data elements.

When adding a new File System 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 File System Locations will not work, and data associations with CLC Metadata tables stored in these 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 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 System Locations, you can either:

1. Rebuild the index for each 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 locations have many data elements as re-indexing large locations can take some time.

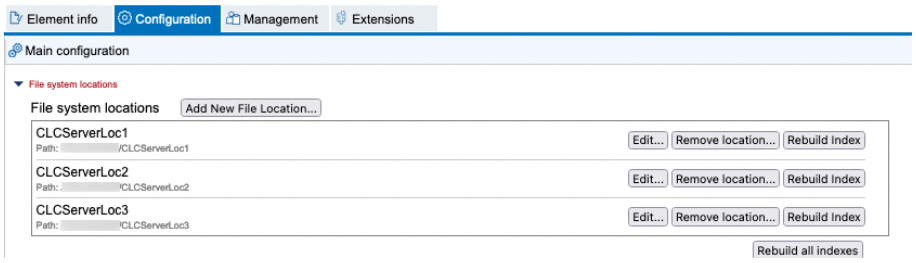


Figure 8.4: *The index for each CLC Server File System Location can be rebuilt. This is useful when searching does not return expected results, but can take a long time for locations with many data elements.*

Details about indexing operations can be seen in the **Audit log**, under the **Management** tab.

Chapter 9

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.

This chapter focuses on server-specific aspects of workflows. General information about workflows are available at: http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Workflows.html.

9.1 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:

http://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Creating_workflow_installation_file.html.

When logged into the web administrative interface as a user with rights to administer workflows, workflows can be installed by going to:

Extensions ( ) | Manage Workflows ( )

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

After installation a green check mark ( ) is shown to the left of the workflow name when the workflow is ready to use. If there is an issue that needs to be attended to, a red exclamation mark ( ) is shown (figure 9.1).

If the workflow installer includes bundled reference data, that data will be stored in 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.

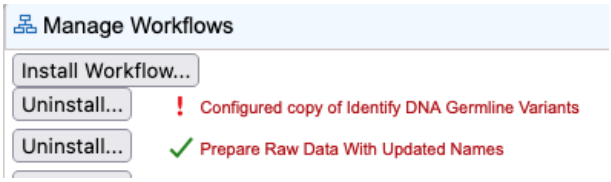


Figure 9.1: *The workflow called Prepare Raw Data with Updated Names is ready to use. There is a problem with the other workflow, as indicated by the red exclamation mark.*

Click on the name of the workflow to reveal a pictorial representation of it. Further details about any problems are presented in this view (figure 9.2) .

Workflow elements with red text can be configured. Clicking on a configurable element opens a dialog where values of open parameters can be changed and a list of the locked parameters is provided (figure 9.3).

Open parameters can be locked, if desired. When a parameter is locked, its value cannot be changed when launching the workflow. Locking and unlocking parameters is described further at 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 most recent change and the name of the administrator who made those changes are reported at the top of the workflow configuration view.

Further information about updating workflows is provided in section 9.3.

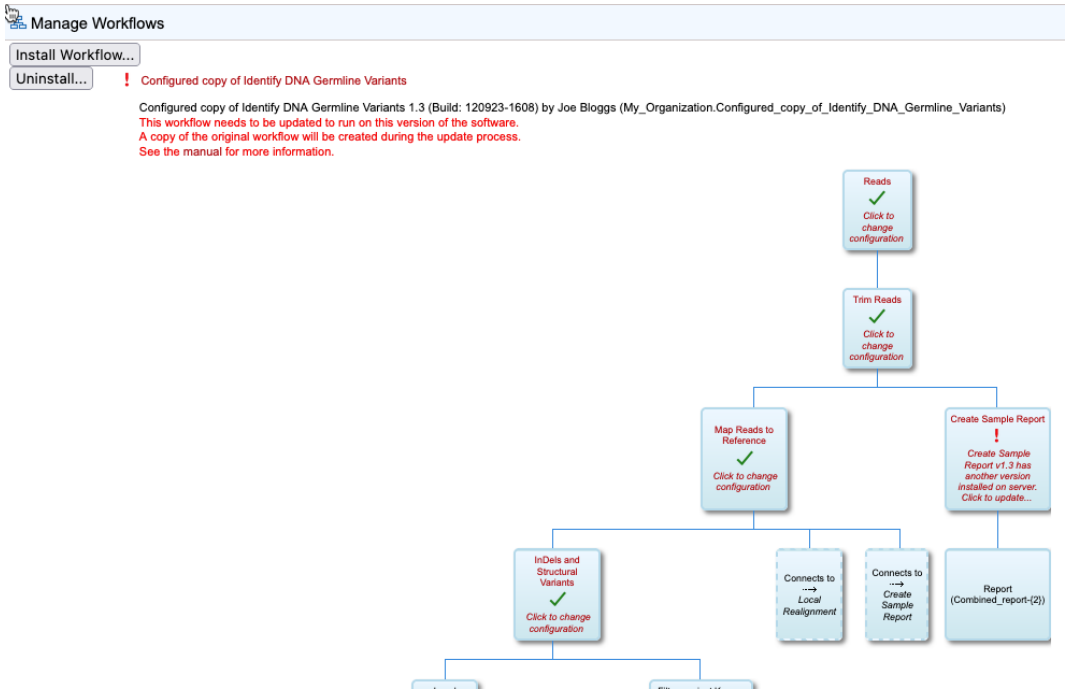


Figure 9.2: *A workflow with an issue is visible. The Create Sample Report element needs to be updated as the version of the tool on the CLC Server is different to the one that the workflow was configured to use.*

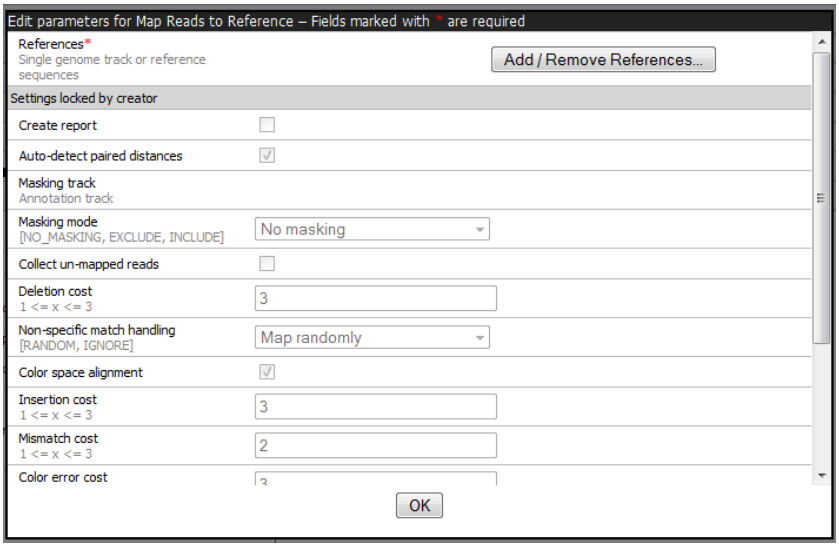


Figure 9.3: *Configuring a workflow element. One parameter in this element can be configured. The rest were locked in the workflow before the workflow installer was created.*

9.2 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 processing** tab. See section 6.5.1 for details, including key considerations when choosing the most efficient option for a particular setup.

When connected to a *CLC Server* from a *CLC Workbench*, you can run workflows on the server. These can be:

- Workflows installed on the *CLC Server*.
- Workflows installed on the *CLC Workbench*.
- Template workflows available from the Toolbox.
- Workflows saved in the Navigation Area (i.e not installed).

You will be blocked from running a *CLC Workbench* workflow on the *CLC Server* if:

- The workflow contains one or more tools on the *CLC Workbench* differs with that on the *CLC Server*.
- The workflow contains tools not available on the *CLC Workbench*.

The rest of this section focuses on running workflows installed on a *CLC Server*.

Running workflows installed on a CLC Server

Workflows installed on the *CLC Server* can be launched using a *CLC Workbench* or the *CLC Server Command Line Tools*.

In a *CLC Workbench*, workflows installed on a server are listed in the **Installed Workflows** folder of the **Toolbox** with a blue S in their icon (figure 9.4).

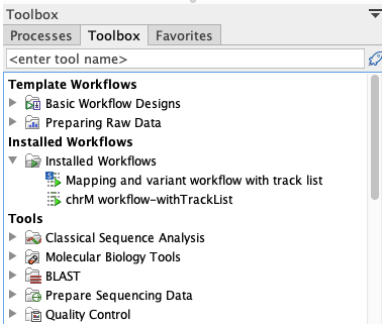


Figure 9.4: *One workflow is installed on the* **CLC Server***, signified by the blue S in its icon. The other workflow is installed in the Workbench.*

Workflows available from a *CLC Server* can be run on any available execution environment (figure 9.5).

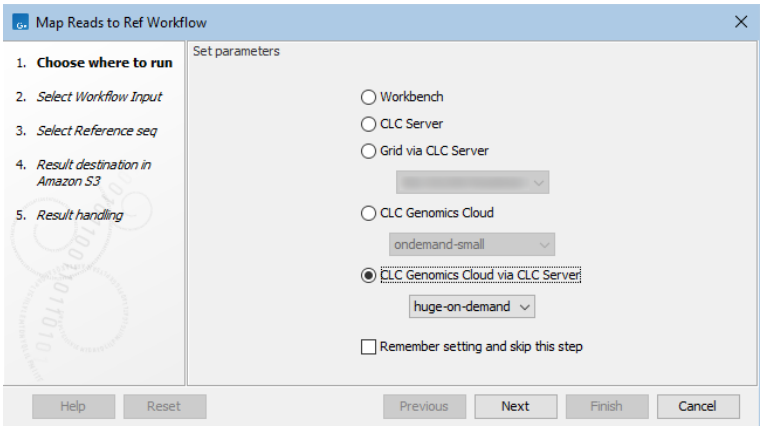


Figure 9.5: *When launching a workflow from a Workbench, the first wizard step displays a list of environments the workflow can be run on.*

9.3 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 installer was created. The exception to this is external applications available from the *CLC Server*, where the version present on the server at the time the workflow was created is included.

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. Thus, 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.

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 indicated with an exclamation mark (figure 9.6). When a workflow can be updated directly in the web administrative interface, the "Update Workflow" button, at the bottom, is enabled. For workflows that cannot be updated this way, a message is presented stating this, with some tips for how to proceed. Details of both these situations are outlined below.

Updating workflows via the server web administrative interface

The "Update Workflow" button, just under the workflow, is enabled when a workflow can be updated in the web administrative interface. Clicking on this button, or any of the elements with exclamation marks, starts the update. See figure 9.6.

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 (figure 9.7). 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 (figure 9.8).

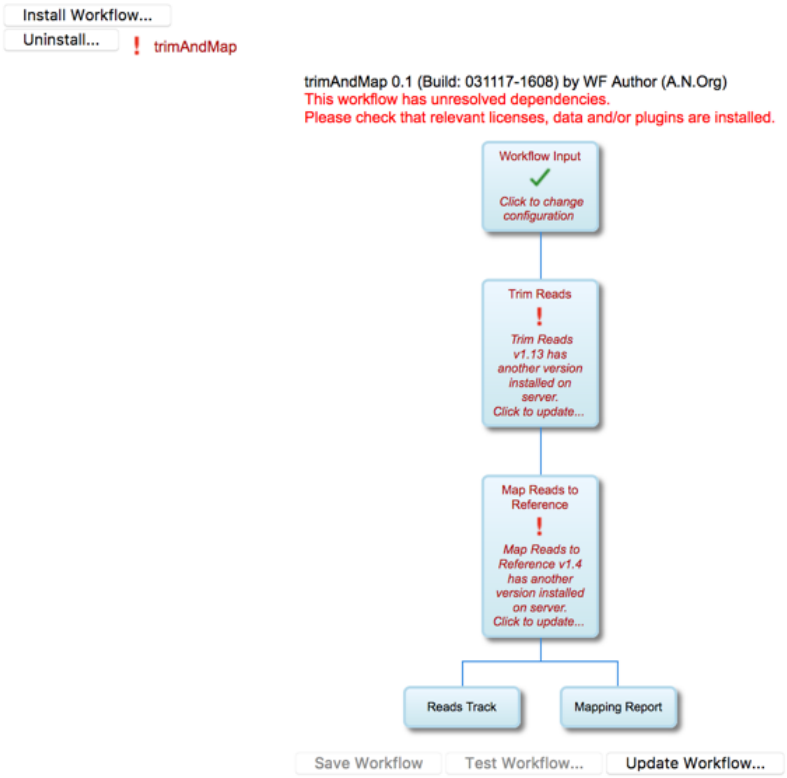


Figure 9.6: *Click on the "Update Workflow" button, or on an element marked with an exclamation mark to start the update.*

If you click on the copy of the original workflow, a button labeled "Re-enable Workflow" appears (figure 9.9). 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.

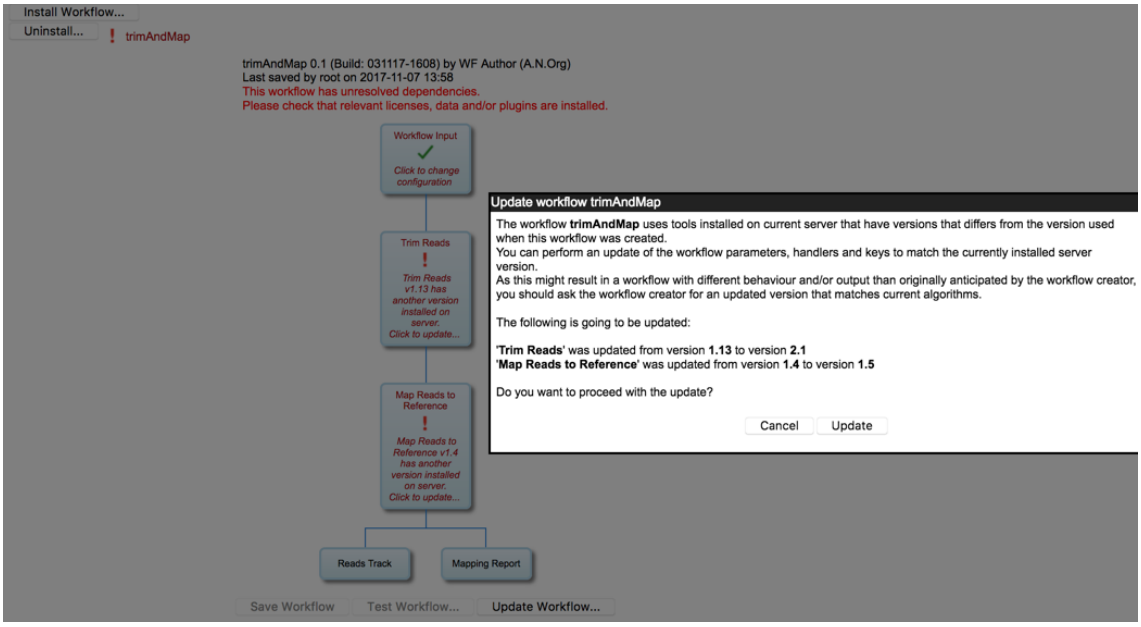


Figure 9.7: *Details about the upgrade are presented, and you can choose whether to proceed with the update, or cancel it.*

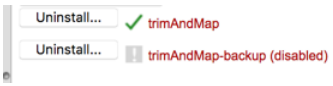


Figure 9.8: *In addition to the updated version of the workflow, marked with a green check mark, a copy of the original workflow is kept. It is disabled and has the original name with "-backup (disabled)" appended.*

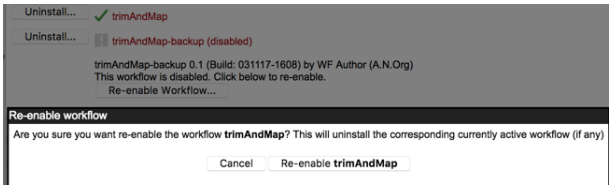


Figure 9.9: *After a workflow has been updated, it is possible to re-enable the original workflow.*

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 9.10).

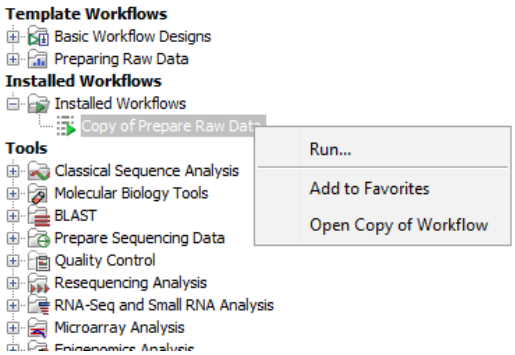


Figure 9.10: *Open a copy of an installed workflow by right-clicking on its name in the Workbench Toolbox.*

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 worfklow, 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 9.1.

Chapter 10

BLAST databases

The *CLC Server* supports running BLAST jobs submitted from client software. BLAST databases stored in an import/export directory can be searched using query sequences selected from a CLC File System Location.

Making BLAST databases accessible via a CLC Server

The location where BLAST databases are stored is specified in the *CLC Server* web administrative interface by going to:

Configuration | **BLAST Databases**

Click on the **Edit BLAST Database Locations** button at the bottom of this tab and select an import/export directory where BLAST databases are stored (figure 10.1).

If the folder containing the BLAST databases is not already configured as an import/export directory, this must be done first (see section 7.1).

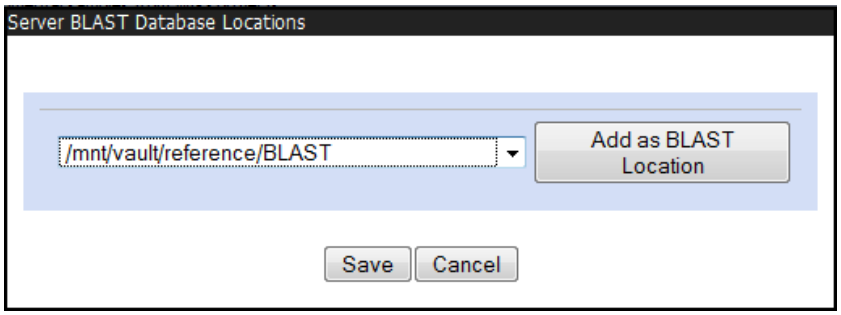
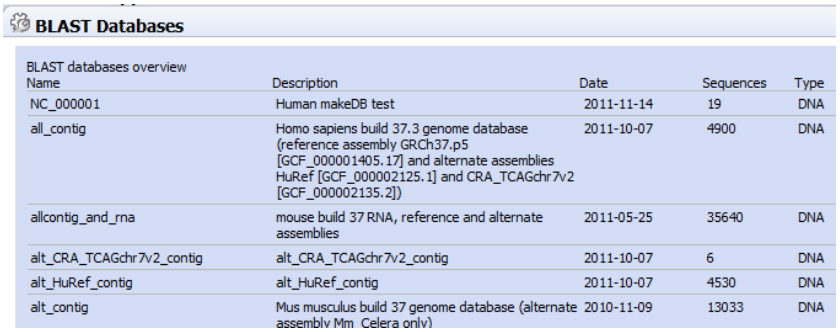


Figure 10.1: *Adding import/export directories as BLAST database locations.*

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 10.2).

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.
- **Description.** Detailed description of the contents of the database.



Name	Description	Date	Sequences	Type
NC_000001	Human makeDB test	2011-11-14	19	DNA
all_contig	Homo sapiens build 37.3 genome database (reference assembly GRCh37.p5 [GCF_000001405.17] and alternate assemblies HuRef [GCF_000002125.1] and CRA_TCAGchr7v2 [GCF_000002135.2])	2011-10-07	4900	DNA
allcontig_and_rna	mouse build 37 RNA, reference and alternate assemblies	2011-05-25	35640	DNA
alt_CRA_TCAGchr7v2_contig	alt_CRA_TCAGchr7v2_contig	2011-10-07	6	DNA
alt_HuRef_contig	alt_HuRef_contig	2011-10-07	4530	DNA
alt_contig	Mus musculus build 37 genome database (alternate assembly Mm_Celera only)	2010-11-09	13033	DNA

Figure 10.2: *A view of part of the listing of BLAST databases configured for access by the CLC Server.*

- **Date.** The date the database was created.
- **Sequences.** The number of sequences in the database.
- **Type.** The type can be either nucleotide (DNA) or protein.
- **Total size (1000 residues).** The number of residues in the database, either bases or amino acid.
- **Location.** The location of the database.

To the right of the Location information is a link labeled Delete that can be used to delete a BLAST database.

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 *CLC Server* using a *CLC Workbench*, 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, 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 a reas 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 10.2, it is possible to delete the databases by clicking the **Delete** link at the far right-hand side of the database information.

Chapter 11

Status and management

CLC Server status and management functionality is provided under the **Management ( )** tab (figure 11.1). 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 12 and the audit log is described in chapter 13.

11.1 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 11.2). 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 11.3). 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 11.1).

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.

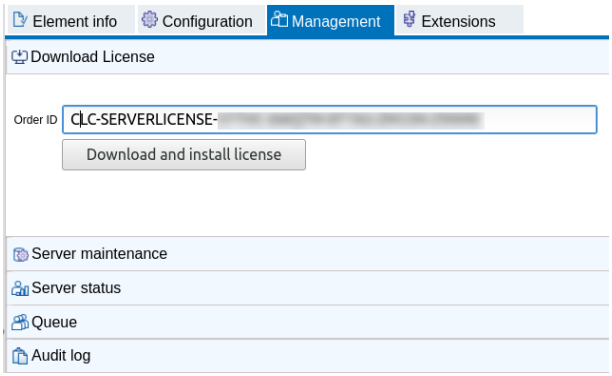


Figure 11.1: *License management is done under the Management tab.*

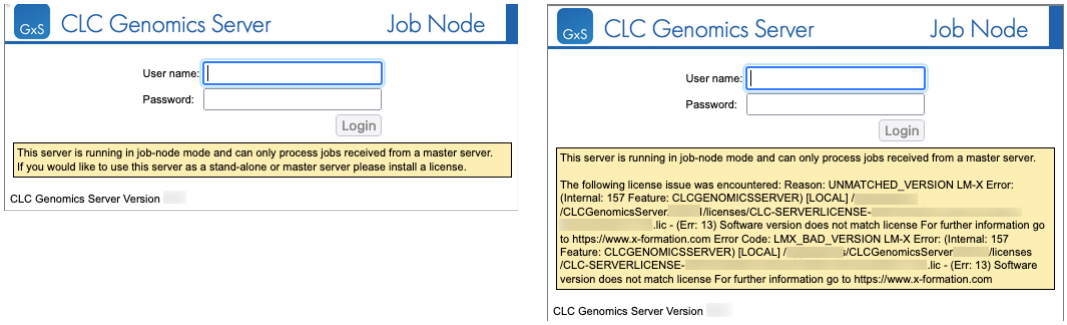


Figure 11.2: *A warning is shown in the web administrative login page for a single server or master node when no license is present for the CLC Server (left) and when a license for the CLC Server is present but is not valid for the version being run (right).*

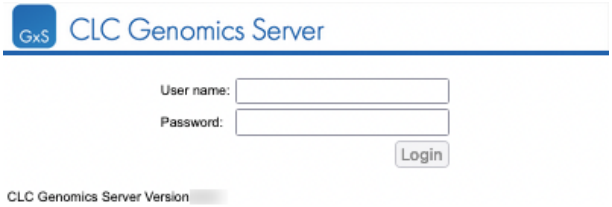


Figure 11.3: *The web administrative interface login page for a single server or master node when a valid license for the CLC Server is present.*

If you are working on a system that does not have access to the external network, then please refer to section 2.6.1.

11.2 Server maintenance

The Server maintenance section is found at:

Management ( ) | Server maintenance ( )

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 11.4).

- **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.

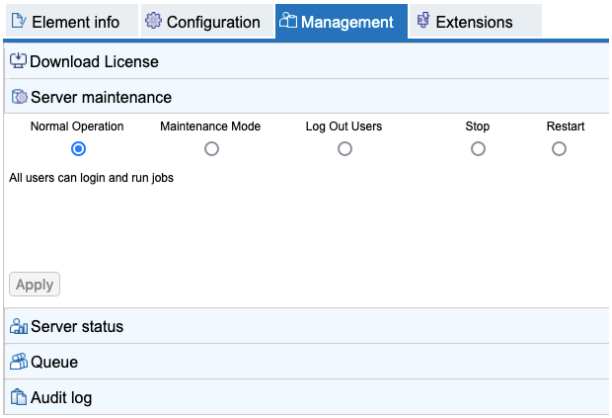


Figure 11.4: *The server administrator can control the operating mode of the CLC Server from under the Server maintenance tab.*

11.3 User statistics

The User statistics section is found under

Management ( ) | Server status ( )

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 11.5).

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

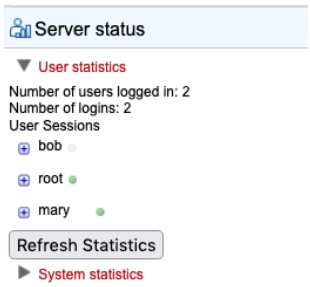


Figure 11.5: *Information about the number of users and active sessions (logins) is provided in the User statistics area. Here, two users are logged in: mary and root. A user called bob has previously logged in but does not have an active session at this point in time.*

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 sybmol to the left of a username to expand the information about that user's sessions (figure 11.6). 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 11.7). 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.

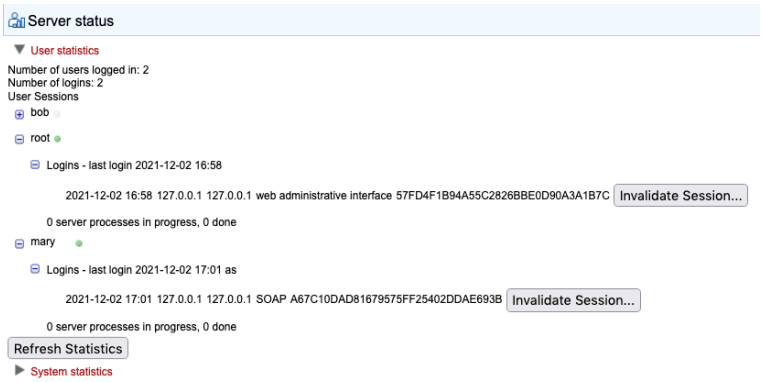


Figure 11.6: *Details about jbloggs' session can be seen by clicking on the small button to the left of that username.*

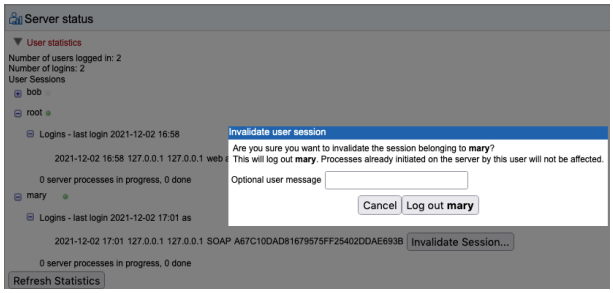


Figure 11.7: *Clicking on the Invalidate Session button forcibly logs a user out of the CLC Server. A message can be provided that will be displayed to that user.*

11.4 System statistics

The System statistics section is found under

Management ( ) | **Server status** ( )

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 11.8. 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.

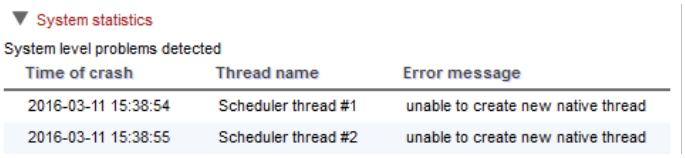


Figure 11.8: *System level problems detected and reported in the system statistics area.*

Chapter 12

Queue

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:

Management ( ) | Queue ( )

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 12.1).

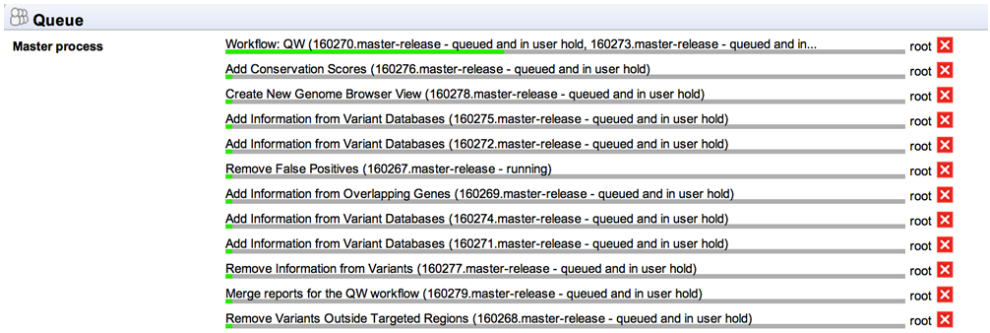


Figure 12.1: *An example of the process queue on a grid setup.*

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.

Chapter 13

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 ( ) | **Audit log** ( )

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. Jobs submitted to run on a *CLC Genomics Cloud* setup have a **CGC** link 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 13.1). For jobs running or previously run on a *CLC Genomics Cloud* setup, information is also available by clicking on the **CGC** link in the status column. Further information about this is provided in the https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Using_CLC_Server_monitor_review_cloud_jobs.html manual.

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 13.2 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.

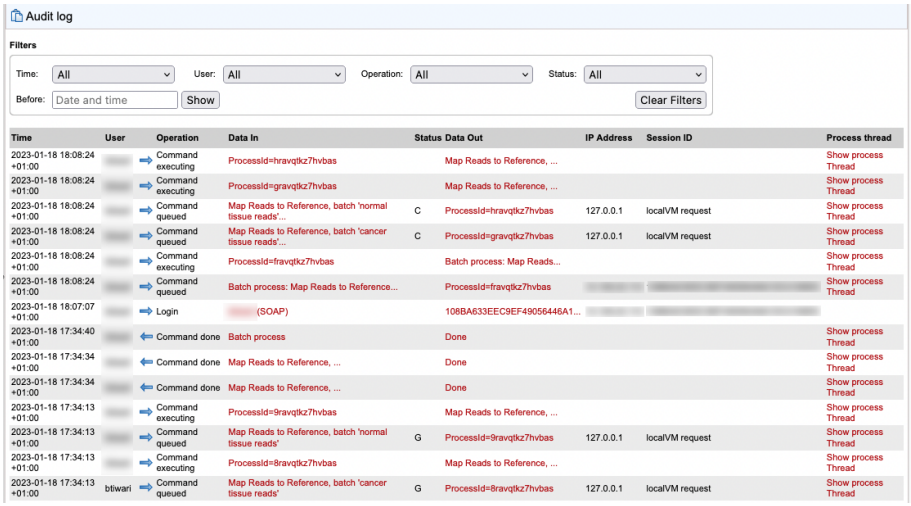


Figure 13.1: *Each operation is logged in the audit log. Items in red are links. Clicking on these allows you to drill down to detailed information. To see just the operations associated with a given job, click on "Show process Thread" on the right hand side. "Command done operations" are preceded by an arrow pointing left. All other commands have arrows pointing right.*

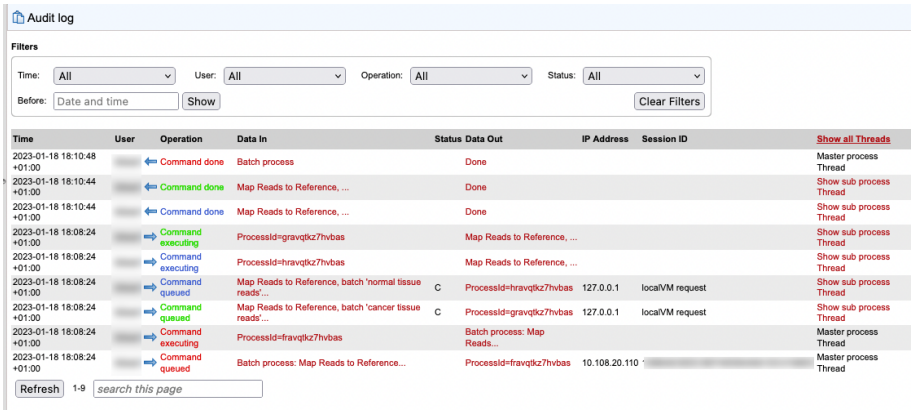


Figure 13.2: *The process threads for a batch run of Map Reads to Reference containing 2 batch units are shown. Operations related to the batch as a whole are in red. The 2 mapping jobs each have three process threads shown. Those for the first batch unit are in green, and those for the second are in blue.*

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 13.3).

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.

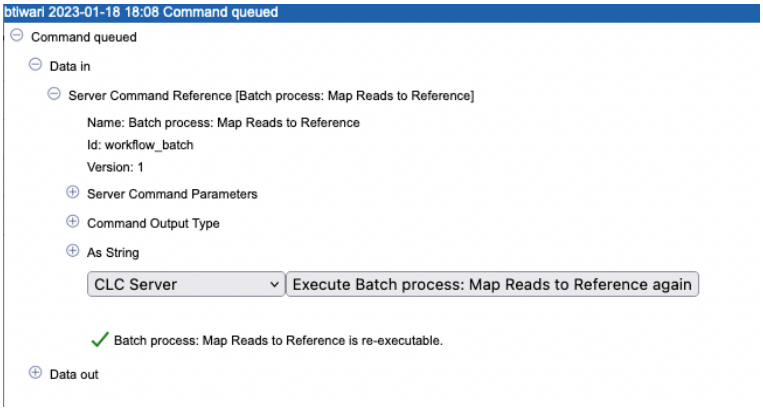


Figure 13.3: *The link in the audit log to "Batch process: Map Reads to Reference" in the "Data in" column was clicked upon. A button to re-execute that job is available in the window that pops up.*

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 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 14

Server plugins

Download and install server plugins and server extensions

Plugins, including server extensions (commercial plugins), are installed by going to the **Extensions ( )** 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 14.1).

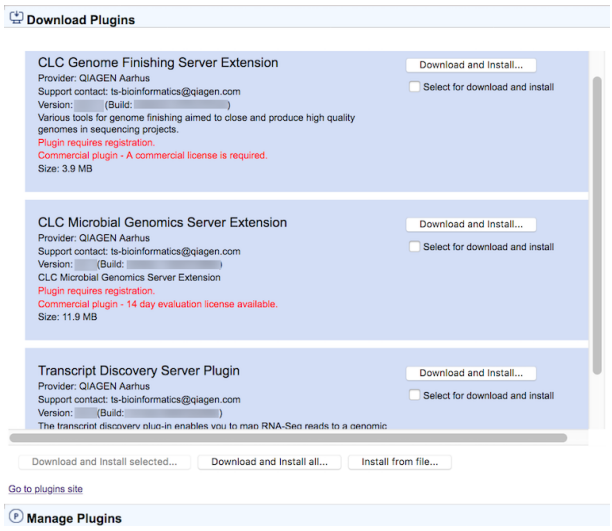


Figure 14.1: *Installing plugins and server extensions is done in the Download Plugins area under the Extensions tab.*

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.

Managing installed server plugins

Installed plugins can be updated or uninstalled, from under the **Manage Plugins** ( ) area (figure 14.2), under the **Extensions ( )** tab.

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.

Links to related documentation

- Maintenance Mode: section 11
- Restarting the server: section 2.7.1
- Plugins on job node setups: section 6.2.4
- Grid worker re-deployment: section 6.3

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.

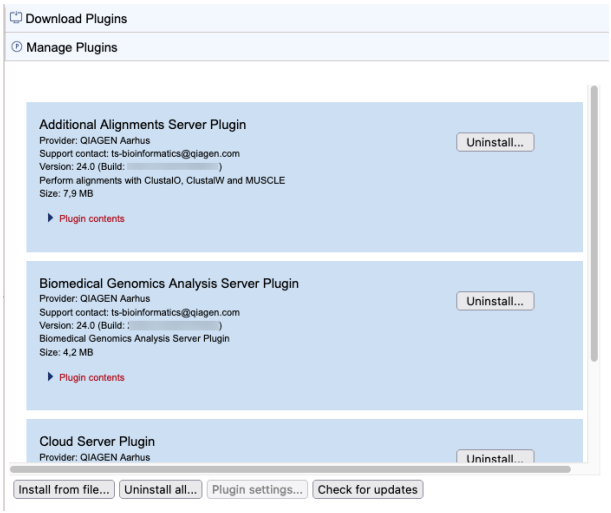


Figure 14.2: *Managing installed plugins and server extensions is done in the Manage Plugins area under the Extensions tab. Clicking on Plugin contents opens a list of the tools delivered by the plugin.*

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

Server extensions are commercial plugins, and thus 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. 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.

Chapter 15

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.
- **Containerized external applications**, where an application is executed from within a Docker container. Containerized external applications are only supported on Linux-based *CLC Server* setups. Making use of containerized external applications in a cloud environment is described in https://resources.qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Using_external_applications_on_cloud_via_CLC_Workbench.html).

Figure 15.1 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

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.

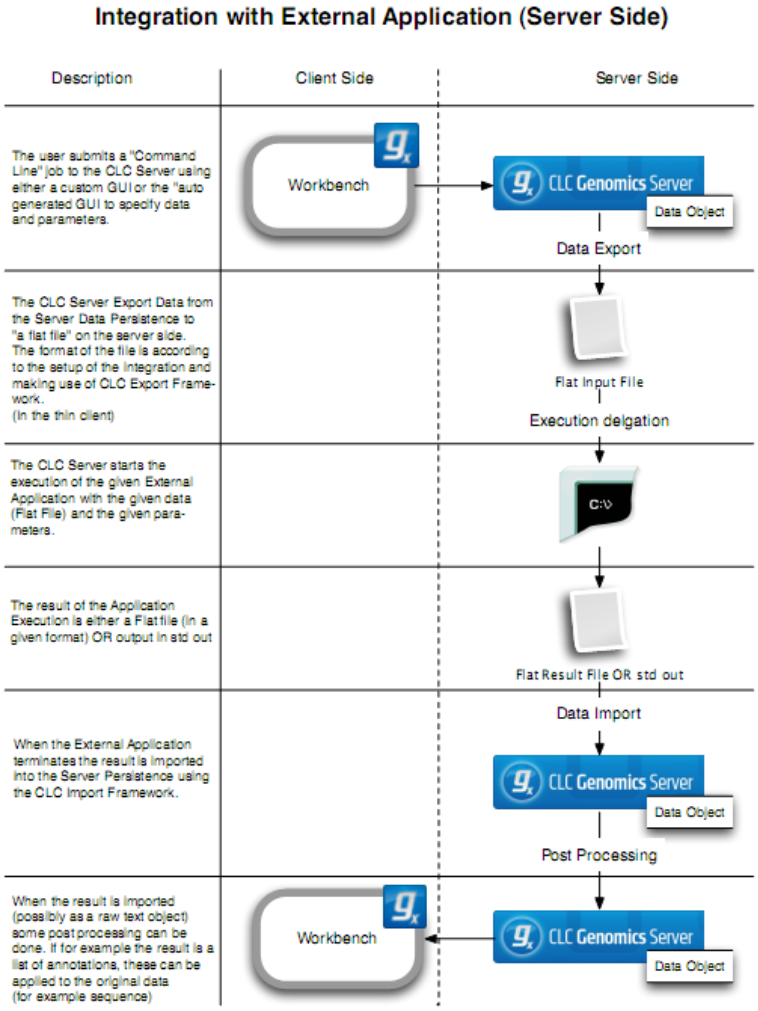


Figure 15.1: *An overview of the flow of a standard external application when launched from a CLC Genomics Workbench.*

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.
- 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, or see updates to existing ones, from a client application, you must log out of and back into the *CLC Server*.
- As soon as a new, enabled application is saved, it becomes available for use by client software the next time it connects to the *CLC Server*. External applications can be disabled, so that they are not available to use.
- By default, all users can use enabled external applications. Limiting access to certain groups is described in section 5.2.
- The *CLC Server* administrator, or members of groups given the relevant permissions, can configure external applications, including controlling whether they are enabled or disabled. Granting permission to configure external applications to members of a non-administrative group is described in section 5.2.

15.1 External application configurations

External applications and the containerized execution environment are configured under:

Extensions ( ) | External applications ( )

See figure 15.2.

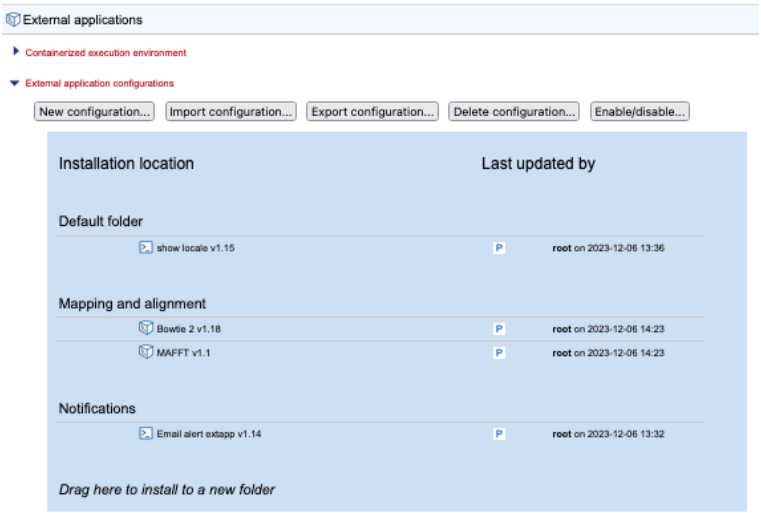


Figure 15.2: *External applications and the containerized execution environment are configured under the External applications tab.*

For quick reference:

- Configuration of the containerized execution environment is described in section 15.1.1.

- Creating and editing external application configurations is described in section 15.2.
- Importing and exporting external application configurations is described in section 15.4.
- Including external applications in workflows is described in section 15.10.
- Launching external applications is described in section 15.11.
- Suggestions for troubleshooting external applications are provided in section 15.12.

15.1.1 Configuring the containerized execution environment

To run containers as external applications on the *CLC Server*, the containerized execution environment must be enabled and configured. This is done under the **Containerized execution environment** area under the External applications tab (figure 15.3). 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 15.2.

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

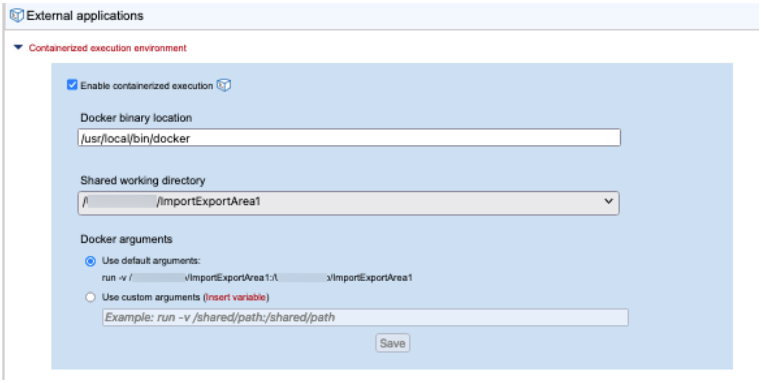


Figure 15.3: *The server's container execution environment is configured under the External applications tab.*

To configure the *CLC Server* for executing containers as external applications, expand the **Containerized execution environment** area (figure 15.3), and:

- Check the **Enable containerized execution** checkbox.
- 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 15.2.6. These are system variables set on the external application process started by the *CLC Server*.

15.2 Configuring external applications

External applications are configured and managed under:

Extensions ( ) | External applications ( ) | External application configurtations

Creating and editing external application configurations is described in section 15.2.

Importing and exporting external application configurations is described in section 15.4.

Note: External applications are enabled by default. Thus, as soon as a new external application is saved, it will be available for use via client software. To keep an external application hidden from client software, mark it as disabled. Access can also be limited by applying permissions.

15.2.1 Editing external applications

To create a new external application, click on the **New configuration...** button (figure 15.4). To open the external application editor for an existing external application, click on its name (figure 15.5).

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 application name The name of the external application, as it will be seen by client software. Click on the small icon to the left to add a description (figure 15.6). This description will then be available in client software. For example, in a *CLC Workbench*, it is available in a tooltip when the mouse cursor is hovered over the external application name

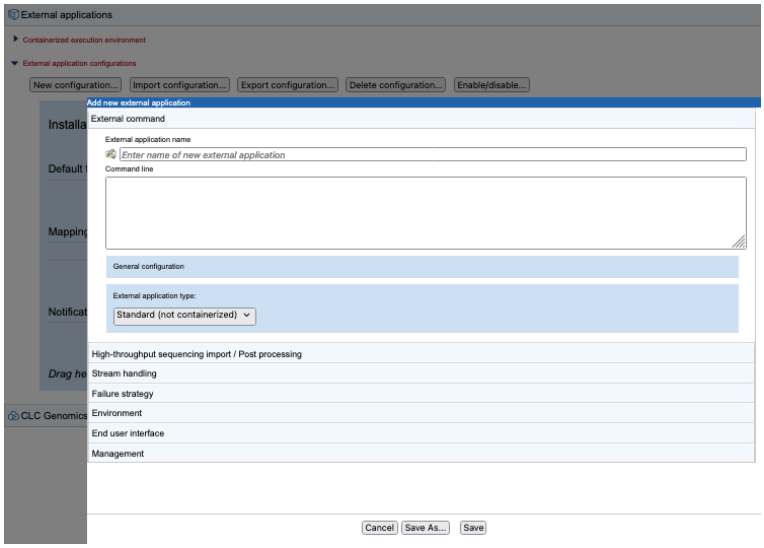


Figure 15.4: *The external applications editor before any configuration has been added. Different aspects are configured under the individual tabs.*

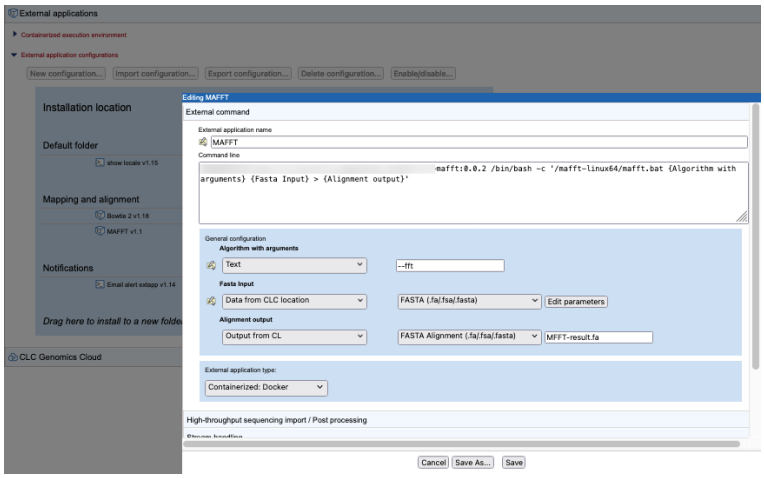


Figure 15.5: *The external applications editor opened by clicking on for the name of an existing external application configuration.*

in the Toolbox panel at the bottom left of the Workbench, and is included in information displayed in the Quick Launch tool (figure 15.7).

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 15.2.2).

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 15.2.3).

Stream handling The handling of information sent to the standard out and standard error streams by the underlying application (section 15.2.4).

Failure strategy Define the failure strategy for the external application (section 15.2.5).

Environment Environment variables that should be present for the external application when it executes (section 15.2.6).

End user interface Settings affecting how an external application is displayed in a *CLC Workbench* client (section 15.2.7).

Management Information and settings relating to the status of the external application, such as the version, whether it is enabled or disabled, and functionality to set access permissions (section 15.2.8).

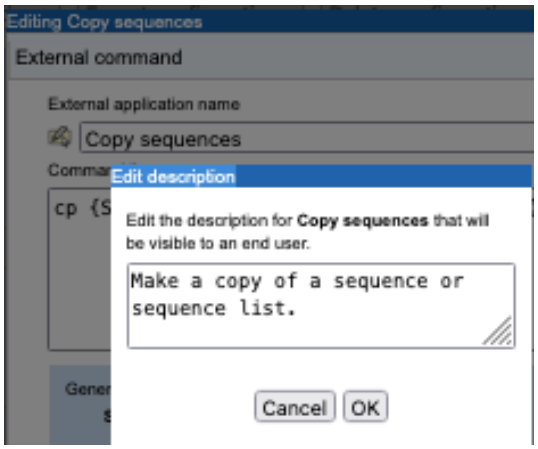


Figure 15.6: *Click on the small icon to the left of the "External application name" field to add a description.*

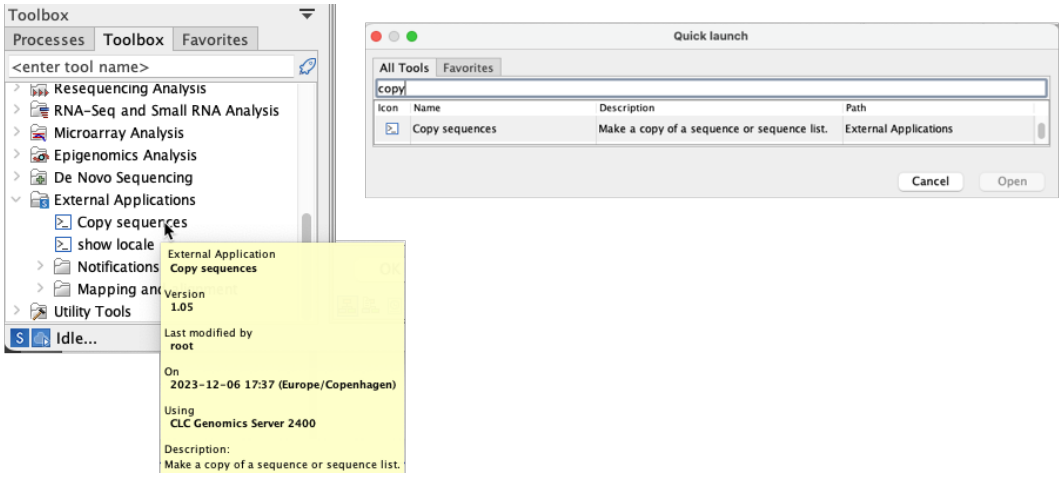


Figure 15.7: *The external application description is visible from Workbenches connected to the CLC Server via a tooltip in the Toolbox (left) and in the Quick Launch tool (right)*

15.2.2 External command

The sections under the **External command** editor tab (figure 15.8) 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 ( ) 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 15.1.1.

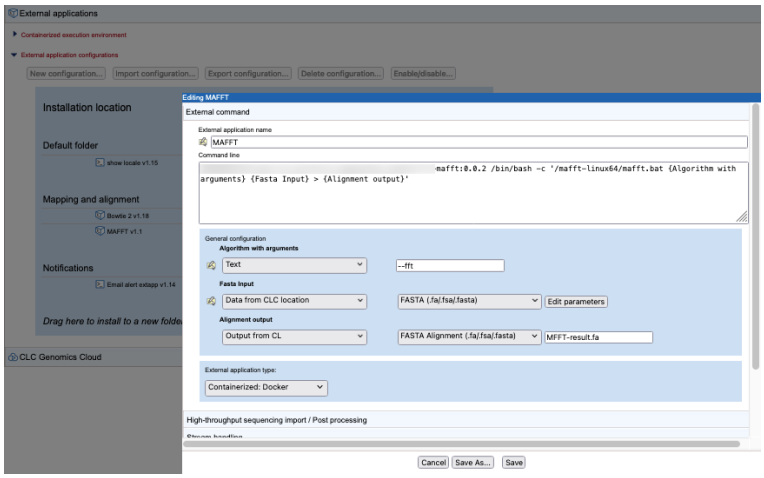


Figure 15.8: *The External command tab in the external application editor*

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, an external application using the cp (copy) command with 2 positional parameters is shown in figure 15.9.

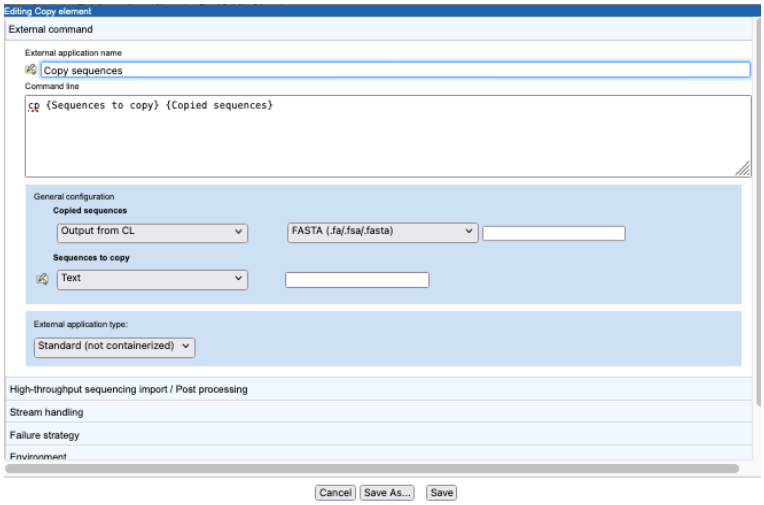


Figure 15.9: *This simple external application includes the 2 arguments the cp command expects: the source and the destination.*

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 15.1.1. 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 15.10, the command for a containerized external application running the alignment program MAFFT is shown as an example.

Parameter value types

Details of parameter value types are provided below. Particularly important types for external application configurations are **Data from CLC location**, which is the usual choice for parameters specifying CLC data as input to be analyzed by the command line application, and **Output 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.

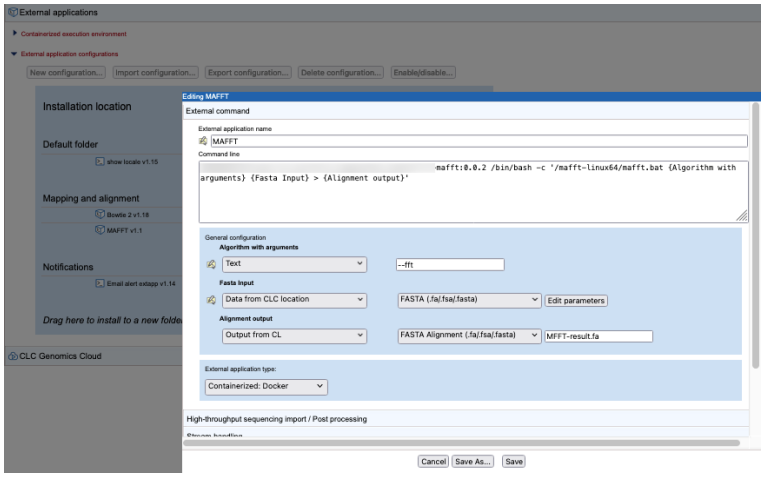


Figure 15.10: *The command line for a containerized external application contains an reference to the image followed by the command to run from the container and the parameters to provide to that command. Parameters in curly brackets are substituted at run time.*

- **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 **??**). A window then appears with a list of configurable parameters, (figure 15.11).
  - **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 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 15.6 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 15.9 for an example.

A very simple configuration illustrating parameter configuration is shown in figure **??** 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 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 from CL

The **Output from CL** option is used for parameters that define an output of the command line application. This output can be a file or a folder containing files. When **Output from CL** is 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 throughtput 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 15.2.3.

- **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: https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=List_bioinformatic_data_formats.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*.

  When **Output 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 15.11). 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.

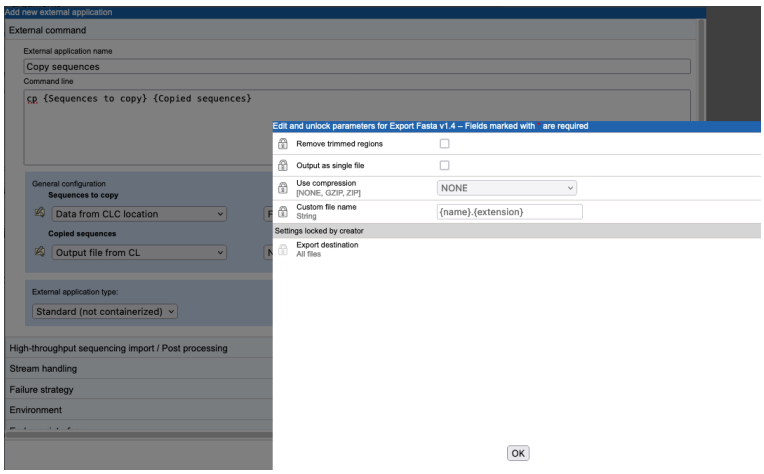


Figure 15.11: *Clicking on the "Edit parameters" button for the "Sequences to copy" parameter brings up a window with the editable parameters for the selected exporter. Parameters with a locked symbol beside them are not shown to, and are thus not configurable by, the end user.*

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 15.2.4. 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.

15.2.3 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 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-throughtput 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 (figure 15.12).
- 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.
- 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

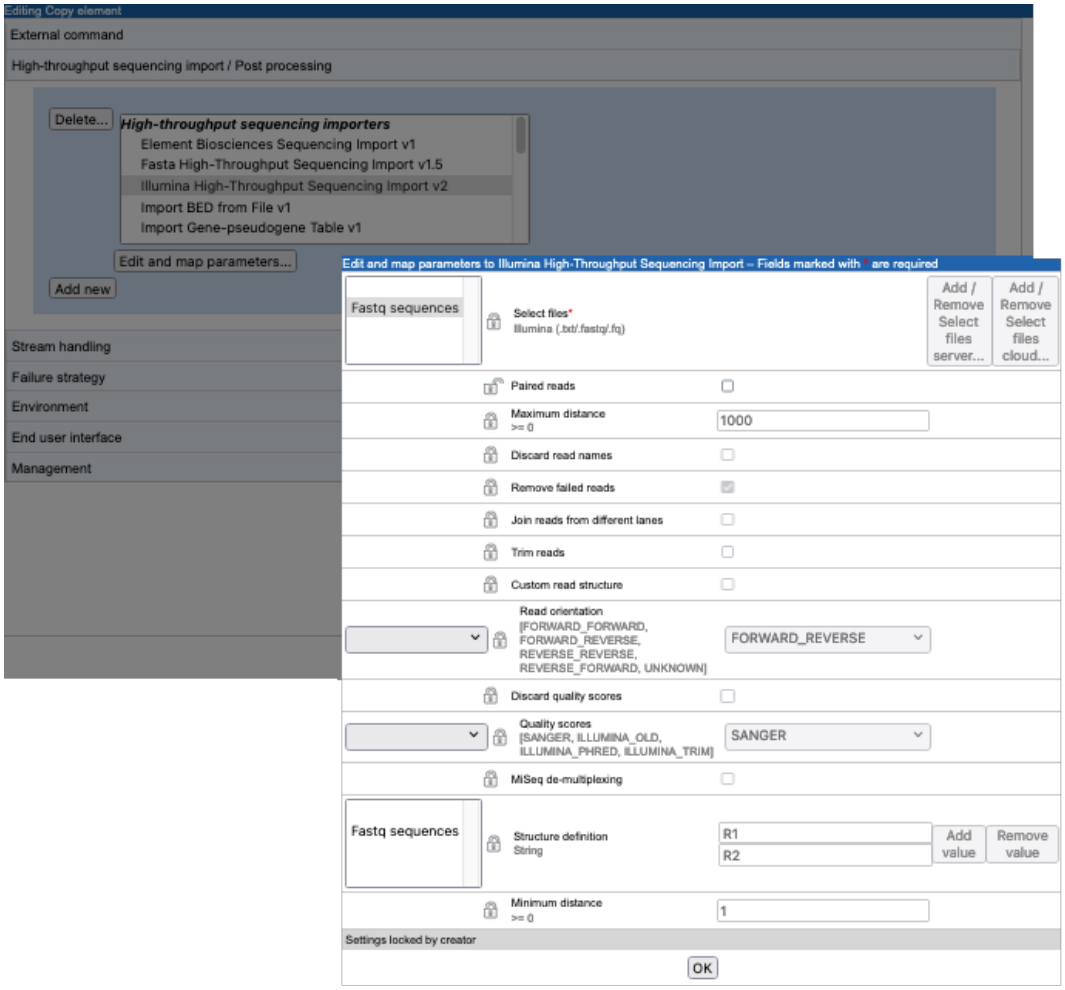


Figure 15.12: *After clicking on the "Add new" button, the Illumina High-Throughput Sequencing Import tool was selected. Clicking on the "Edit and map parameters..." button opens a window to configure that importer. Here, a parameter called "Fastq sequnces" has been selected sa the input, and the Paired reads setting was unlocked, and that option then deselected.*

example, after saving the changes shown in figure 15.12, the text in the General configuration area for that output would say: "Linked with Illumina High-Throughput Sequencing Import".

To remove the connection between a parameter and a high throughput sequencing importer or post processing tool, the simplest option is usually to delete the relevant configuration in the High-throughput sequencing import / Post processing area.

15.2.4 Stream handling

Handling standard out and standard error streams for the external application is configured under the **Stream handling** tab, shown in figure 15.13.

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

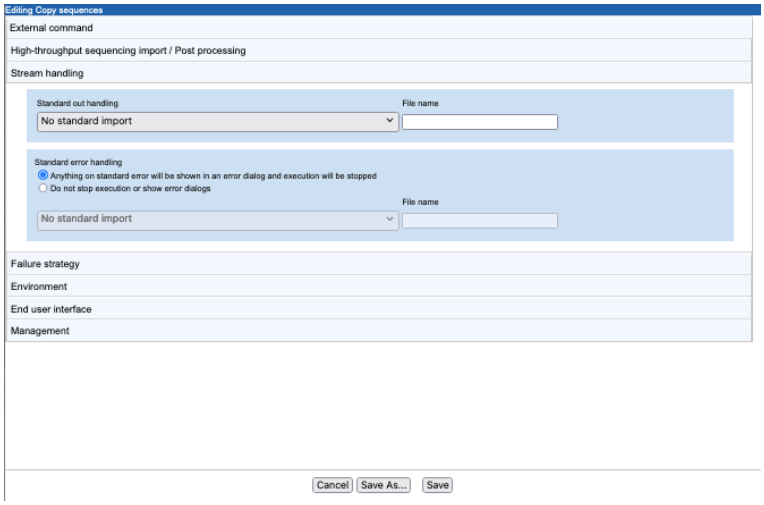


Figure 15.13: *The handling of standard out and standard error is configured under the Stream handling tab.*

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.

15.2.5 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 15.14).

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.

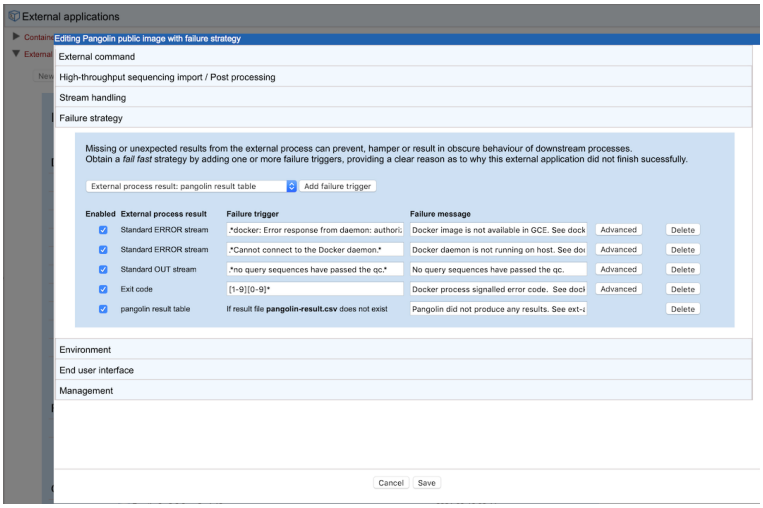


Figure 15.14: *Failure triggers for an external application are configured under the Failure strategy tab.*

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.

15.2.6 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 15.15, an environment variable named "*HELLO*" with value "*hello world*" would be present in the execution environment of the external application.

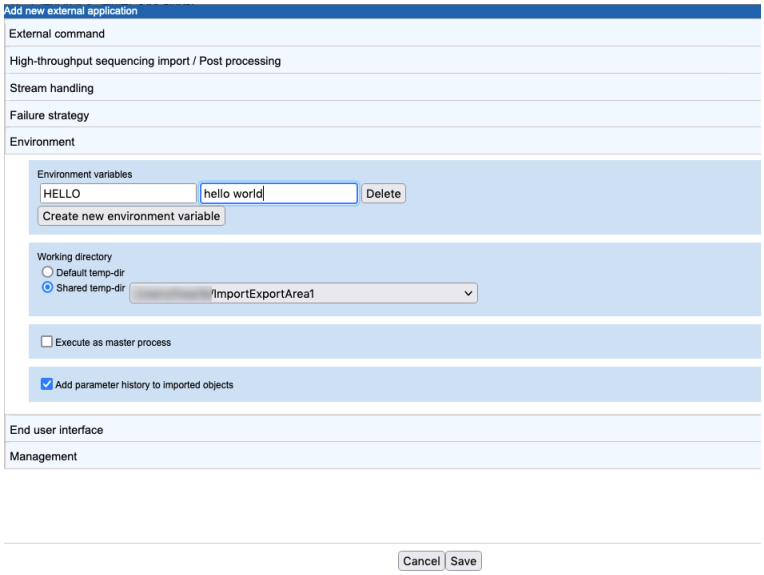


Figure 15.15: *Under the Environment tab, settings related to the environment an external application will be run on are configured.*

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 15.1.1.

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*(see 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. The history entries for the simple "Copy sequences" example described earlier in this chapter are shown in figure 15.17. The 3 genera steps of an external application, export from the CLC Server, running the command line, and importing result into the CLC Server, each have their own history entry.

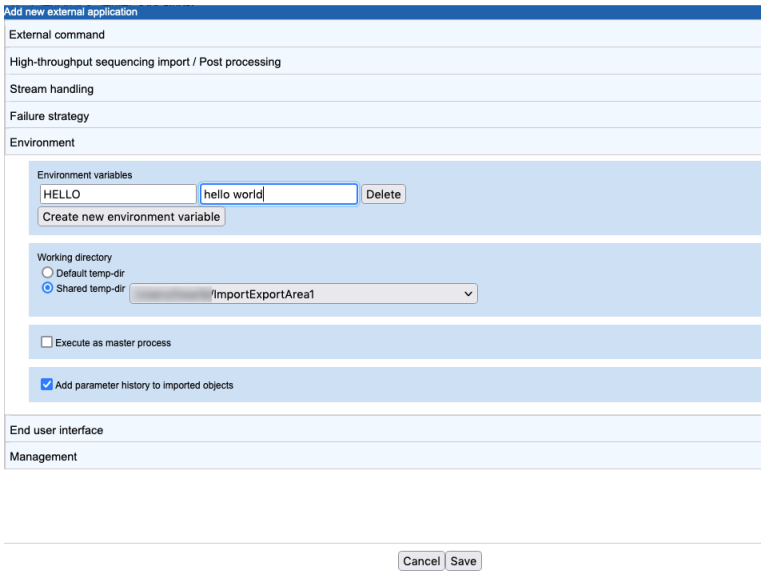


Figure 15.16: *Settings related to the environment an external application will be run on are configured under the Environment tab.*

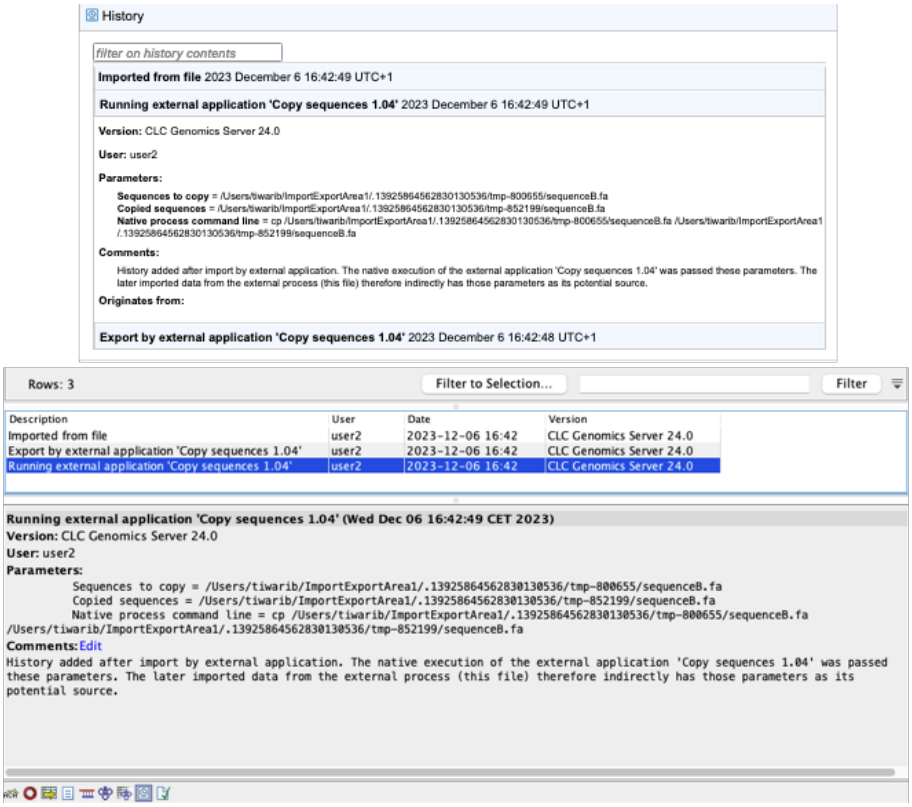


Figure 15.17: *An example of the history information provided for a CLC data element generated by an external application in the web client (top) and in a CLC Workbench (bottom). In both cases, the history entry relating to running the native command is open.*

15.2.7 End user interface

Settings under the *End user interface* tab (figure 15.18) 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. External applications in the "Default folder" are listed directly under Toolbox | External Applications (figure 15.19).

  External applications can also be moved to different subfolders by dragging and dropping them directly in the External applications tab.

  Additional notes:

  - Folder names are case sensitive; "foldername" and "Foldername" refer to 2 different subfolders.
  - Subfolders of subfolders are 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.

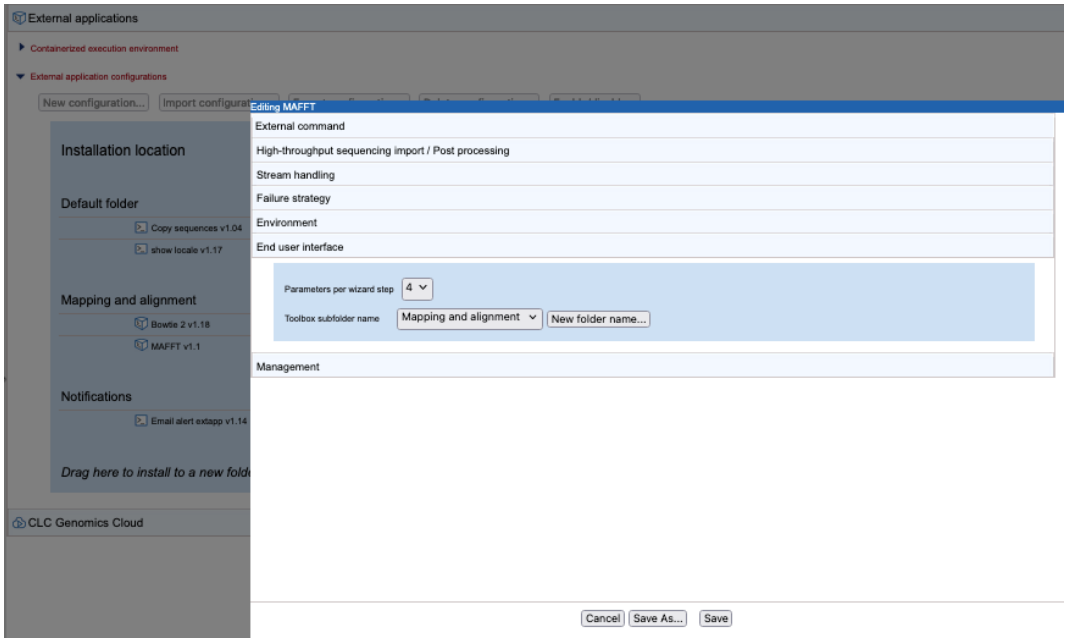


Figure 15.18: *End user interface settings affect how the external application is presented in a CLC Workbench client.*

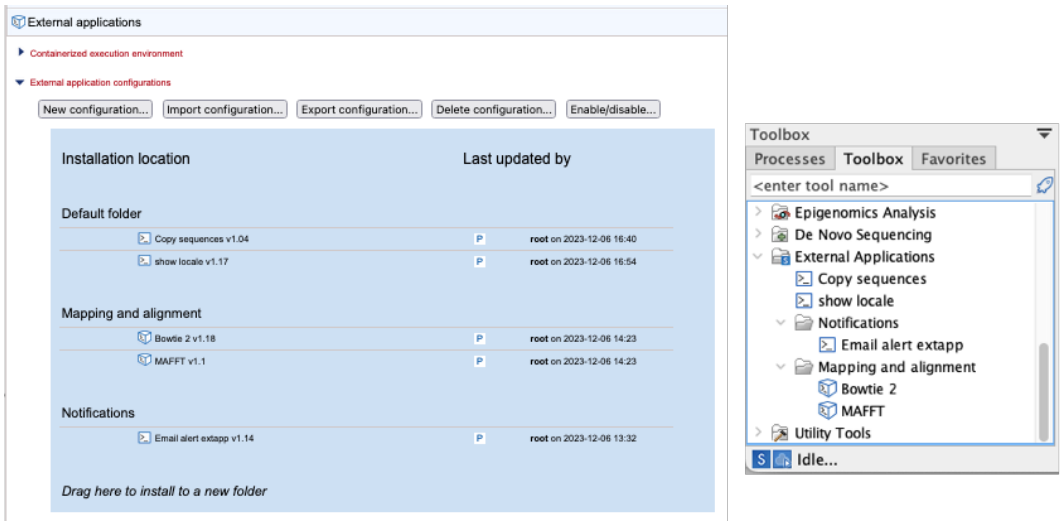


Figure 15.19: *The folder structure configured in the web client (left) is what Workbench users will see in their Toolbox when logged into the CLC Server (right).*

- Changes to the subfolder location, or any other configuration changes to an external application, are seen by Workbench users when they next log into the *CLC Server*.

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.

15.2.8 Management

Under the Management tab (figure 15.20), is the following:

- The version of the external application.
- The user who created the external application and when it was created,
- The user who last modified the configuration and when that was done.
- Functionality for enabling or disabling the external application. When disabled, it will not be accessible to run from client software.
- Functionality to set access permissions on the external application. This can also be done under Configuration | Global Permissions.

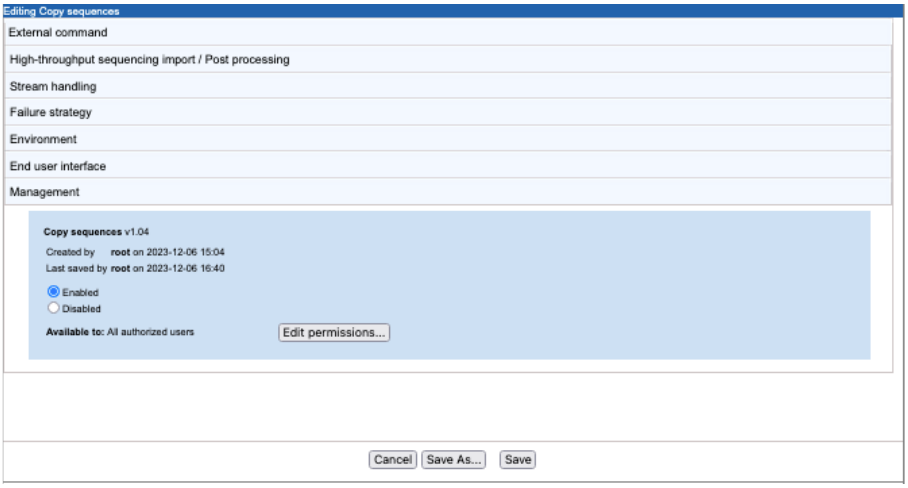


Figure 15.20: *The Management tab of the external application editor*

15.3 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 3.3.2.

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 (see section 7.1).

Data import and export are described in the client software manuals: https://digitalinsights.qiagen.com/technical-support/manuals/.

15.4 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 15.21). 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: `https://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.

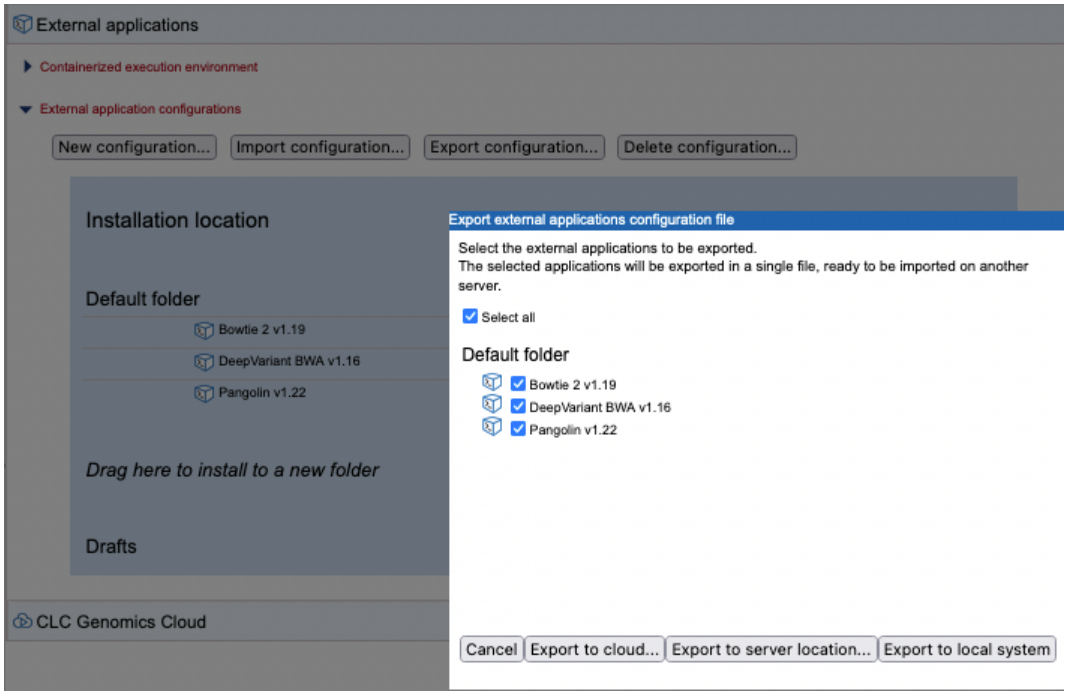


Figure 15.21: *One ore more external applications configurations can be exported to an XML file in the location of your choice.*

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 15.22).:

- **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.

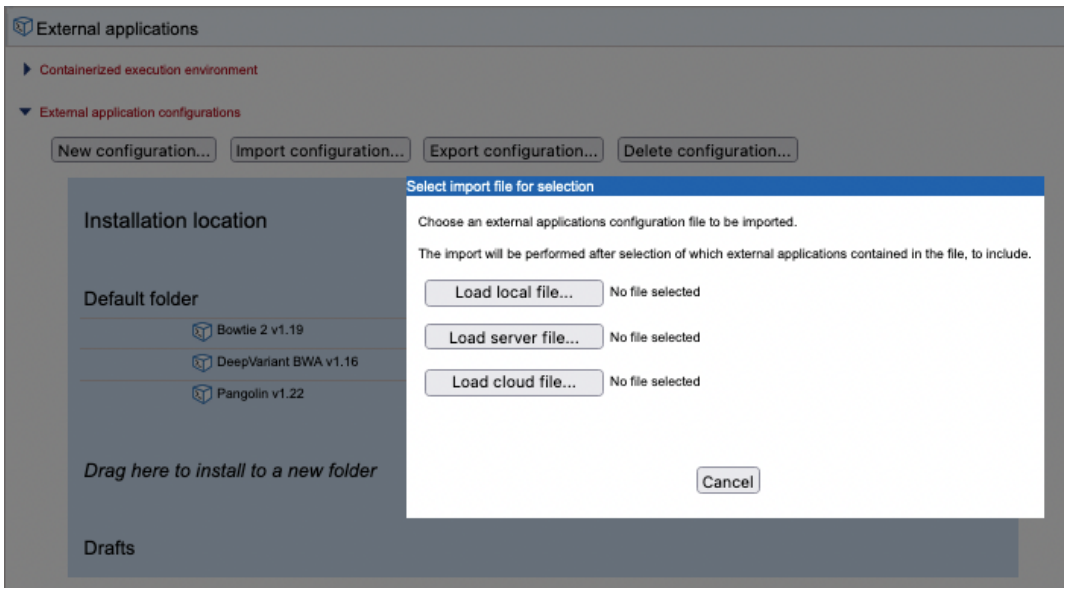


Figure 15.22: *External applications configurations can be imported from local folders, CLC Server import/export directories and cloud locations*

For information on importing external application intended for use on the cloud into a *CLC Workbench*, please refer to the *CLC Cloud Module* manual: https://resources.qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Using_external_applications_on_cloud_via_CLC_Workbench.html.

15.5 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 enabled external applications, a button labeled **Updates are needed** appears in the "External application configurations" area (figure 15.23). 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

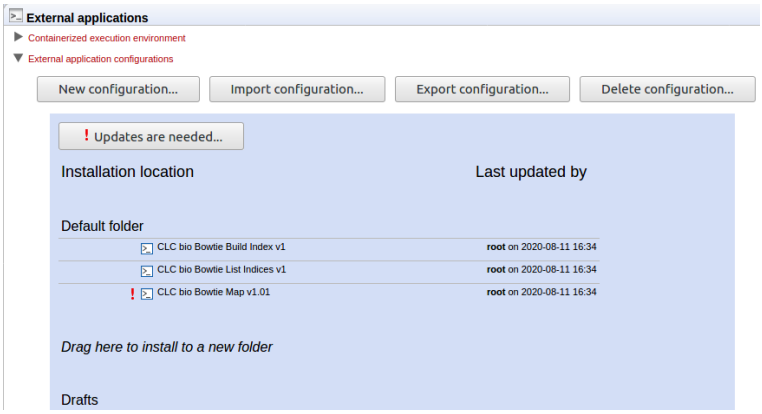


Figure 15.23: *The CLC bio Bowtie Map external application needs updating, as indicated by the exclamation mark by its name. The "Updates are needed" button is visible, reflecting this. Clicking on that button allows us to auto-update all (possible) external application configurations in a single action.*

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 disabled external applications and such 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 15.24).

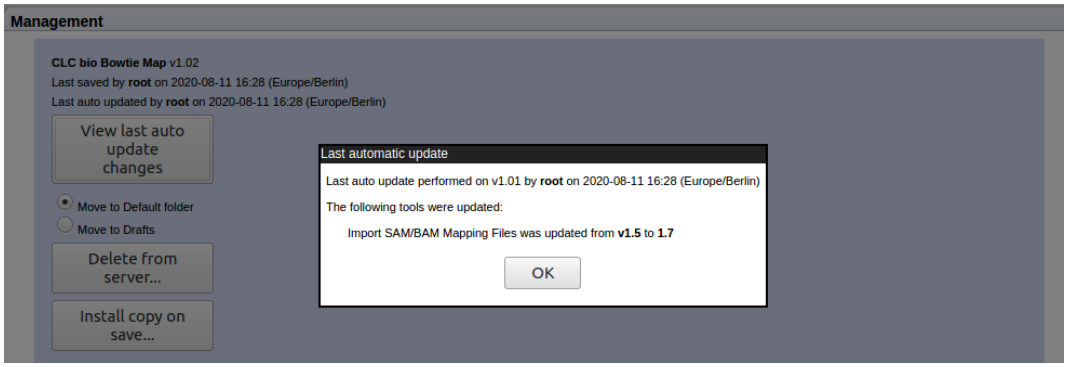


Figure 15.24: *If auto-updates have been carried out, details of the latest one can be viewed under the Management tab of the external application configuaton editor. Here, the View last auto-update changes button has been clicked, revealing information about what was done.*

15.6 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*.

15.6.1 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 15.25.
- 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 15.2.6 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 15.5.

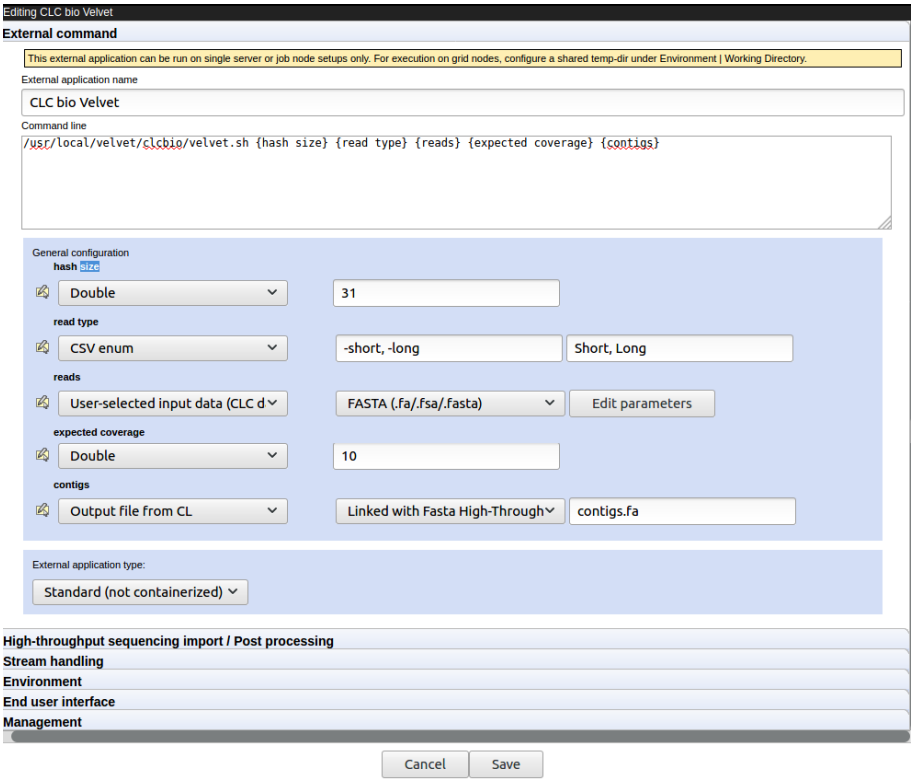


Figure 15.25: *The Velvet configuration after the external application configuration file has been imported.*

15.6.2 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 15.26.
- 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.

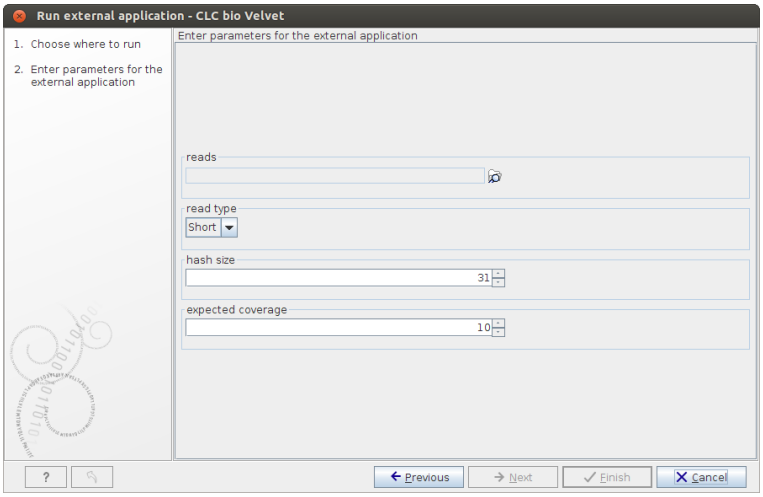


Figure 15.26: *Configuring Velvet parameters from a Workbench.*

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

15.6.3 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 15.25, 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.

15.7 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.

15.7.1 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 15.7.3 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.

15.7.2 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 15.27).

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.

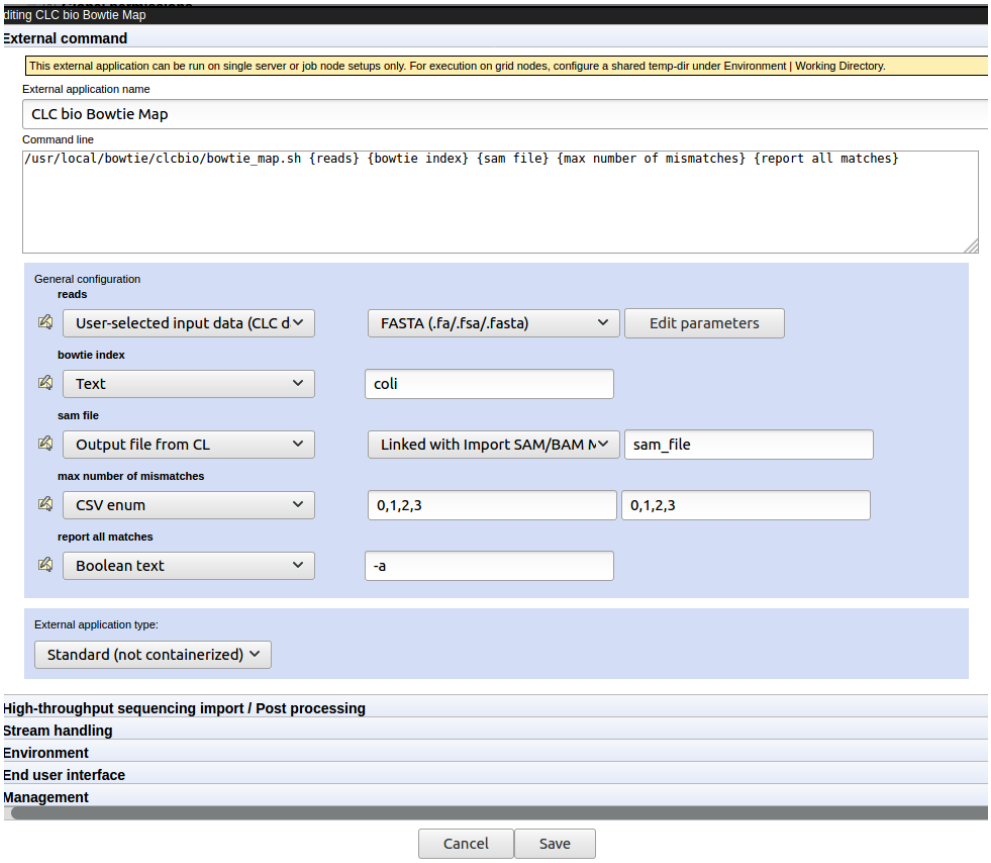


Figure 15.27: *The External command tab of the CLC bio Bowtie Map configuration is visible in the external application editor.*

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 paramaters** 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 15.28 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 15.28.

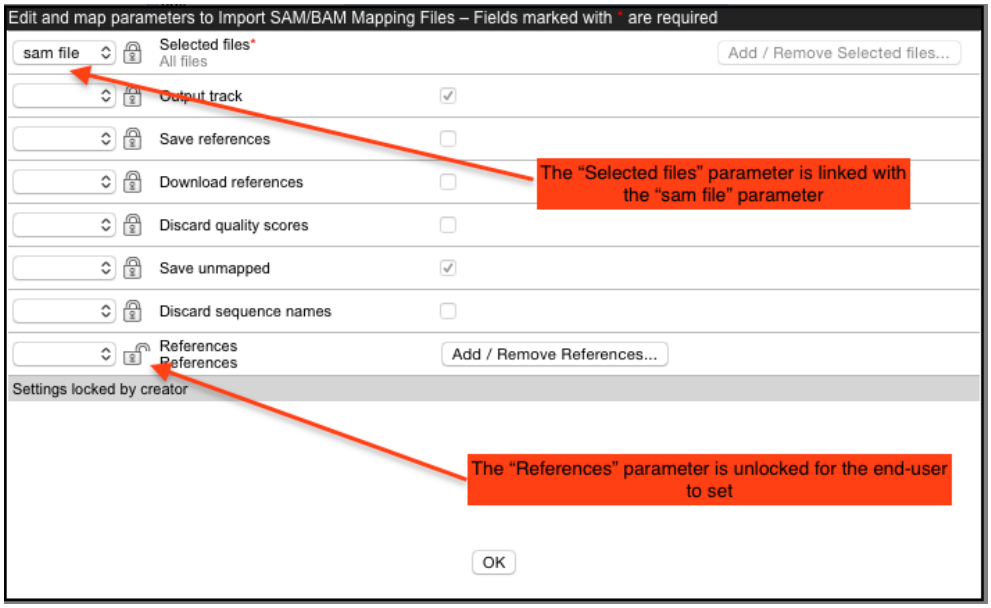


Figure 15.28: *Configuration of Import SAM/BAM Mapping Files for import of a sam file after mapping using Bowtie.*

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.

15.7.3 Setting the path for temporary data

The **Environment** handling shown in figure 15.29 allows you to specify a folder for temporary data and add additional environment variables to be set when running the external application.

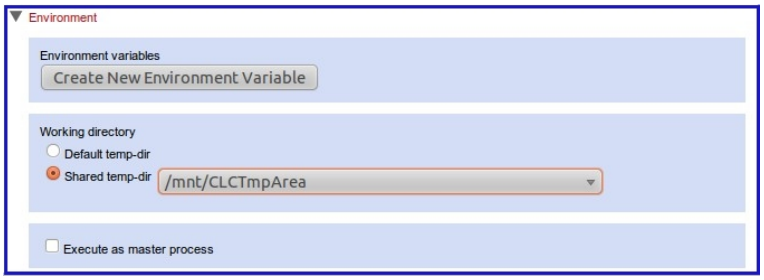


Figure 15.29: *Where to save temporary data needs to be defined for Bowtie.*

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.

15.7.4 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.

15.8 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 15.8.1 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 15.30.

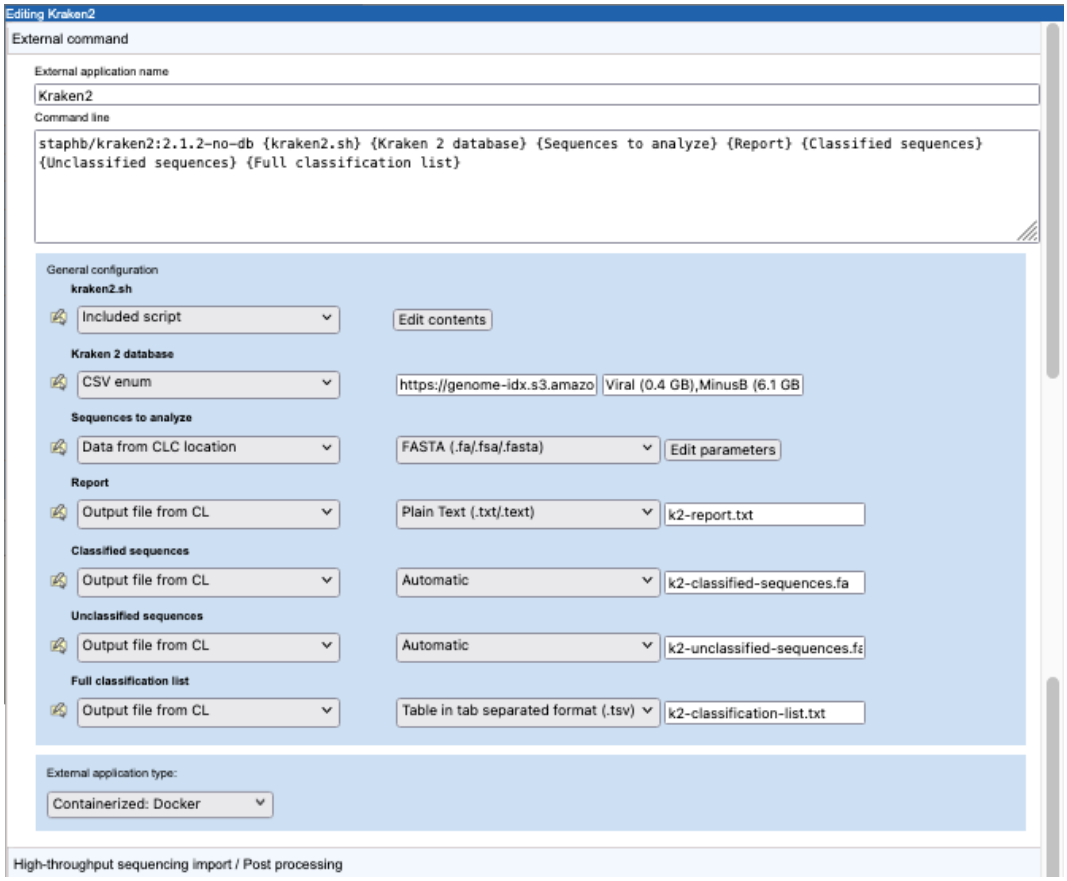


Figure 15.30: *Configuration a containerized external application for Kraken2.*

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 15.32). 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 resultfiles 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 15.33.

- **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 15.31).

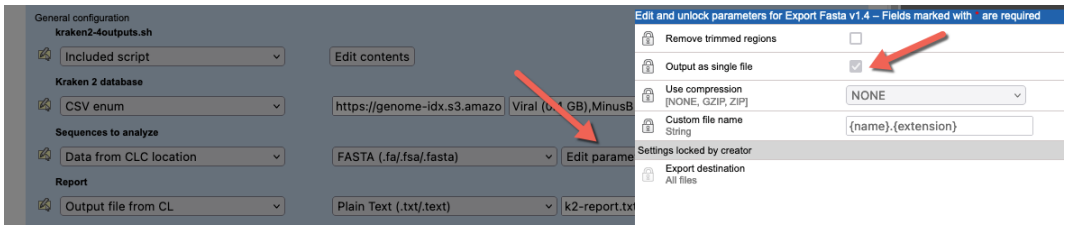


Figure 15.31: *The FASTA export option "Output as single file" has been enabled. If 2 or more sequence lists are selected as input, they will be exported to a single file, which is then used by the Kraken2 command.*

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 https://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 https://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 https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Working_with_tables.html.)

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.

  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.

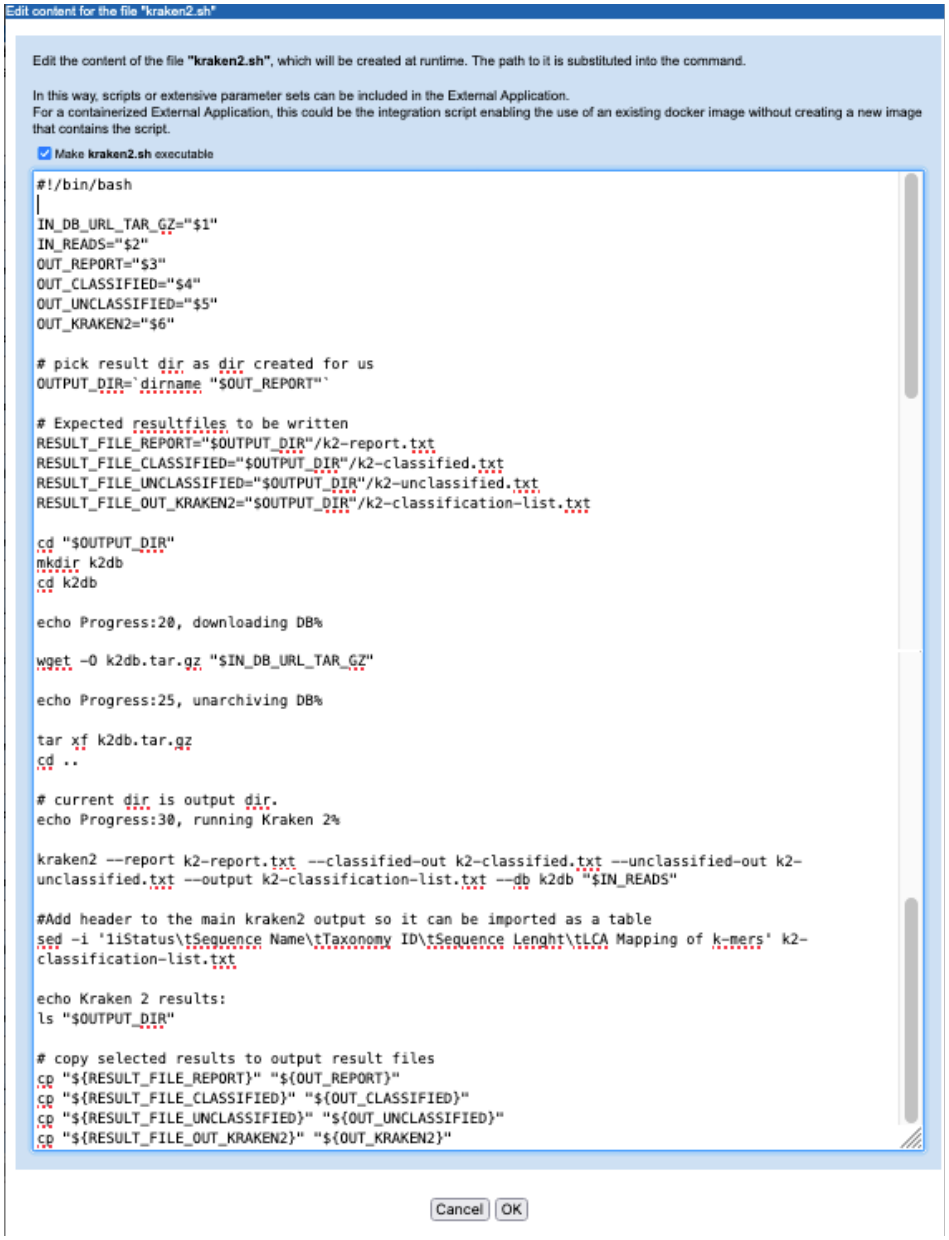


Figure 15.32: *A script is defined that will be run in the Docker container. This version includes the commands to download and unpack the Kraken2 database specified by the user when launching the application.*

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.

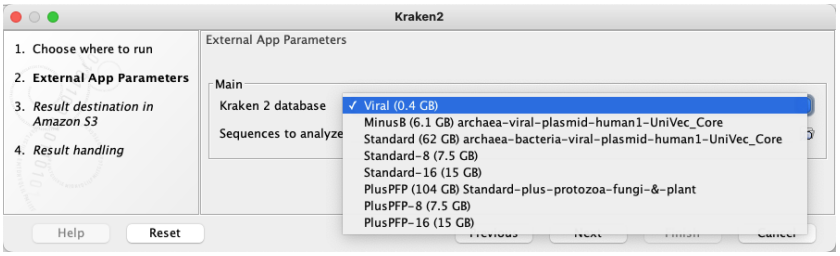


Figure 15.33: *The CSV enum parameter type used for the Kraken 2 database parameter results in a drop-down list of options in the Workbench wizard for the application.*

- **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 wrote the information to, `Kraken2-docker-log.txt`. We also specified that the plain text importer be used to import the file (figure 15.34).

  The base name of this file appears in an output channel of the corresponding workflow element (figure 15.35).

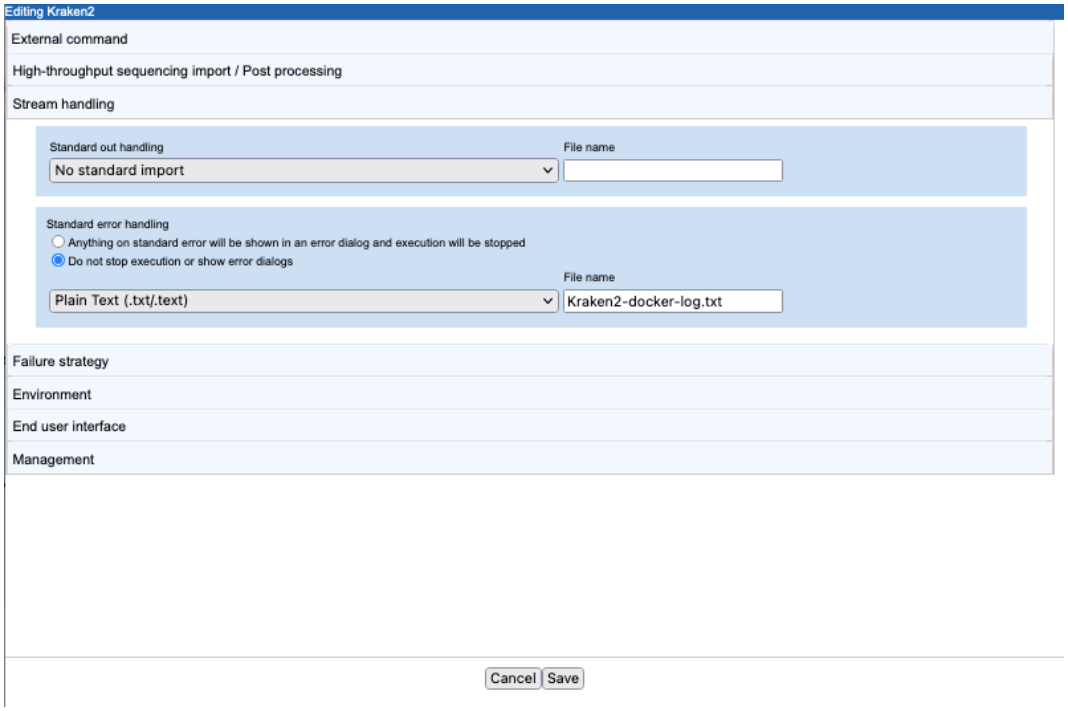


Figure 15.34: *The CSV enum parameter type used for the Kraken 2 database parameter results in a drop-down list of options in the Workbench wizard for the application.*

Making the external application available for use

When this external application is saved, it becomes available to run on the *CLC Server* unless its status is set to Disabled.

To run external applications on the cloud, the *CLC Cloud Module* is needed. See `https://resources.`

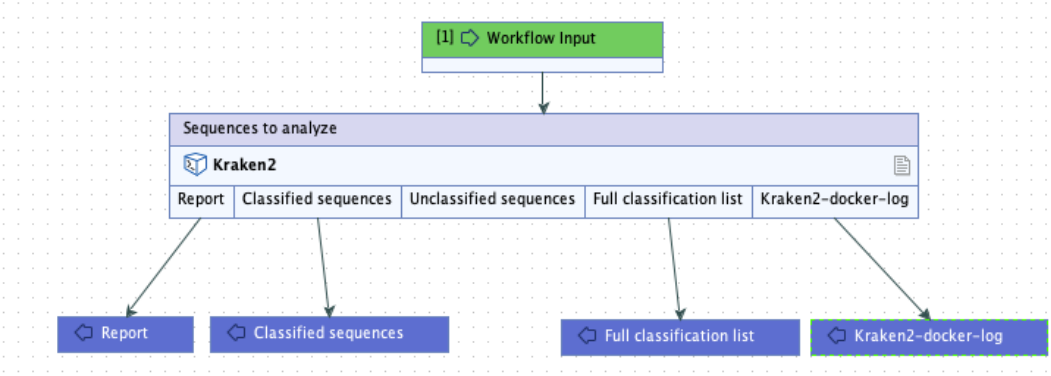


Figure 15.35: *A workflow containing the external application. In workflows, the outputs to collect can be specified. Here, all outputs except the unclassified sequences will be returned by the analysis.*

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 https://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 15.36.

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 15.37).

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.

Interacting with files on AWS S3 via a *CLC Workbench* is described at https://resources.

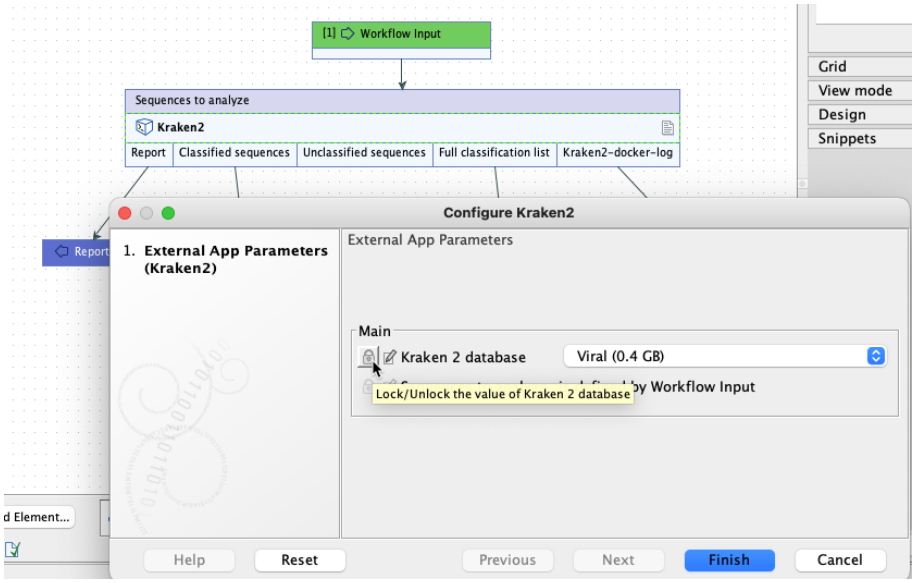


Figure 15.36: *By default, the database parameter will be locked in the workflow element. Configure the workflow element to unlock this parameter to allow users to select a database when launching the workflow.*

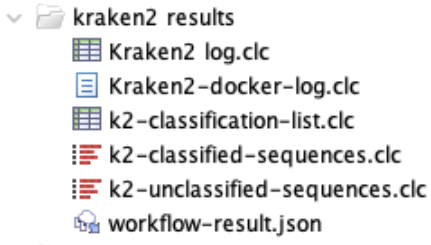


Figure 15.37: *Outputs from a Kraken2 containerized external application after running it on AWS using functionality provided by the CLC Cloud module.*

qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Working_with_AWS_S3_using_Remote_Files_tab.html. Interacting with files on AWS S3 via the *CLC Server* web interface is described in section 7.5.

15.8.1 Extending the Kraken2 external application for paired data

Here, we expand upon the configuration described in section 15.8 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 15.39 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 15.38).
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 15.39).
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 15.39).

   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-through 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`.

   For details of how filenames are interpreted, please refer to https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=llumina.html.

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.

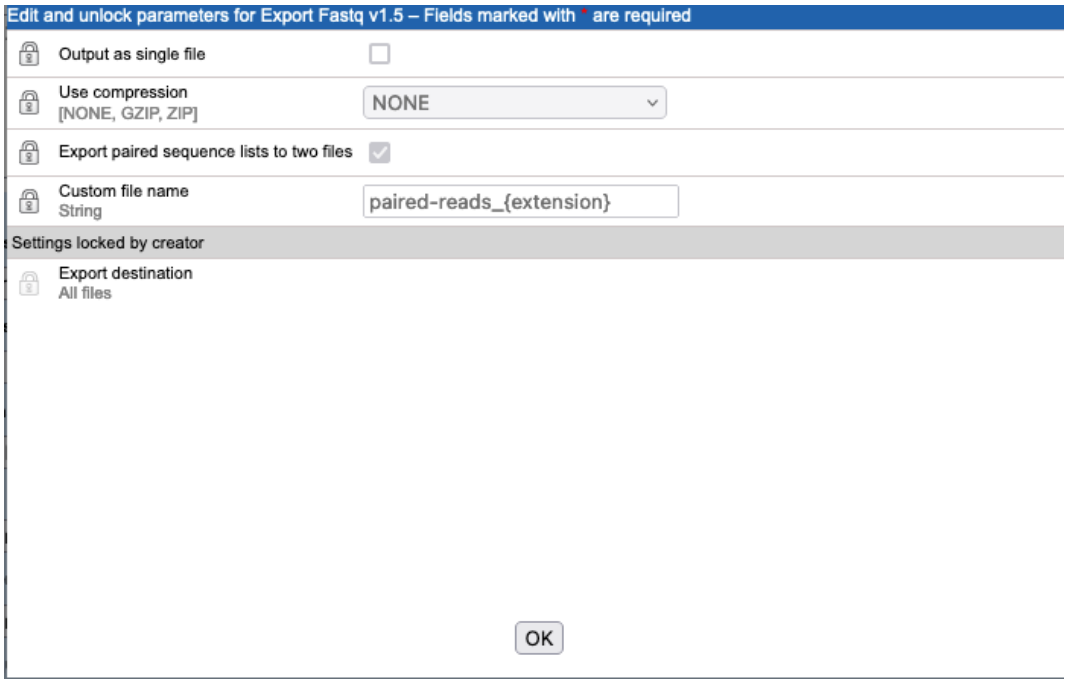


Figure 15.38: *Clicking on the "Edit parameters" button for the "Sequences to analyze" parameter opens up a configuration window for the FASTQ exporter. The default settings are shown here, with the "Export paired sequence lists in two files" option enabled.*

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.

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.

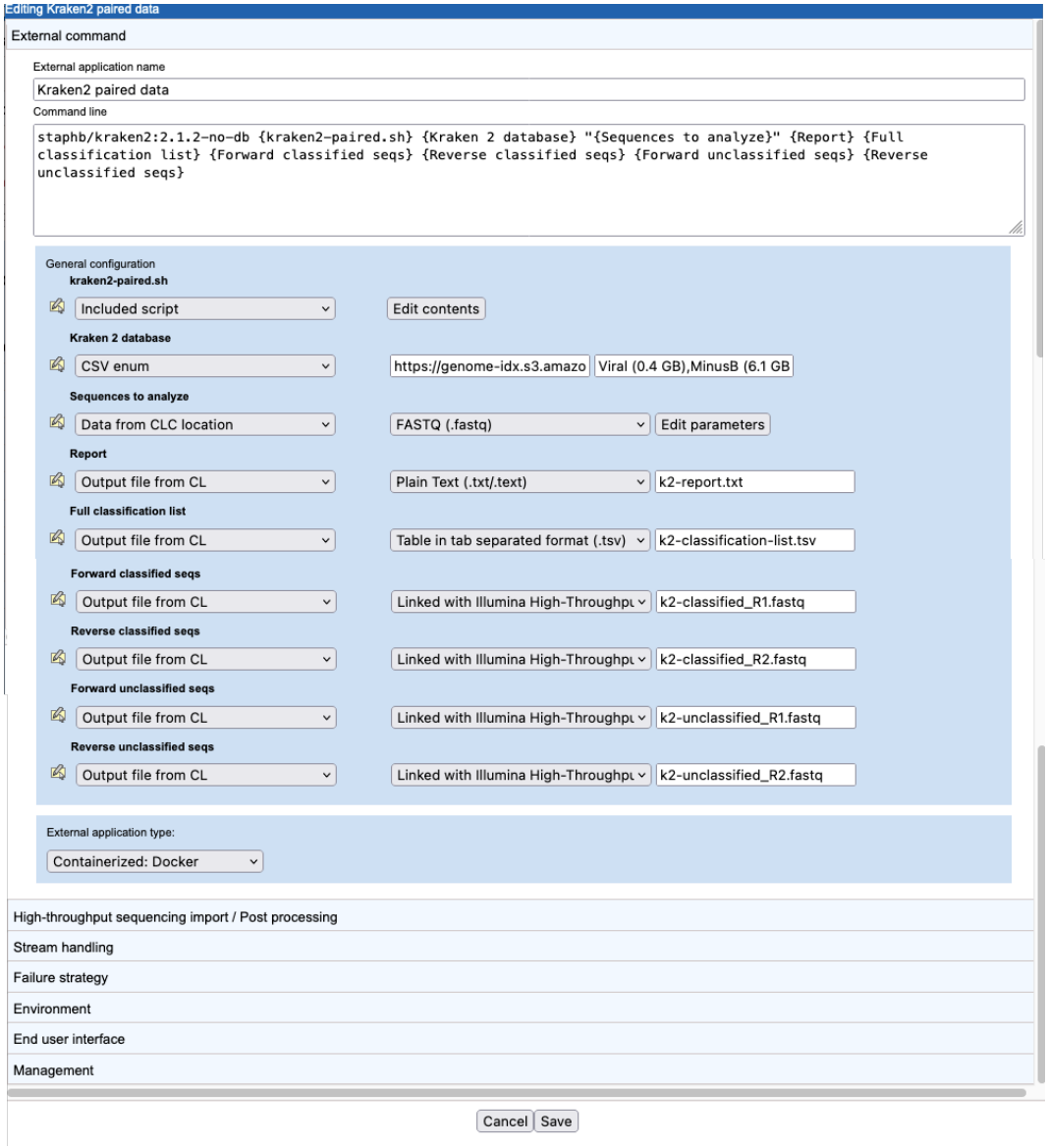


Figure 15.39: *The external application command and top level configuration to support running Kraken2 with paired sequences as input and for handling the paired sequences Kraken2 generates as output.*

3. Click on **Add new** and select an importer. Here, we select the Illumina importer (figure 15.40).

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 15.41).

   Other importer settings Keep the default settings for all the other parameters.

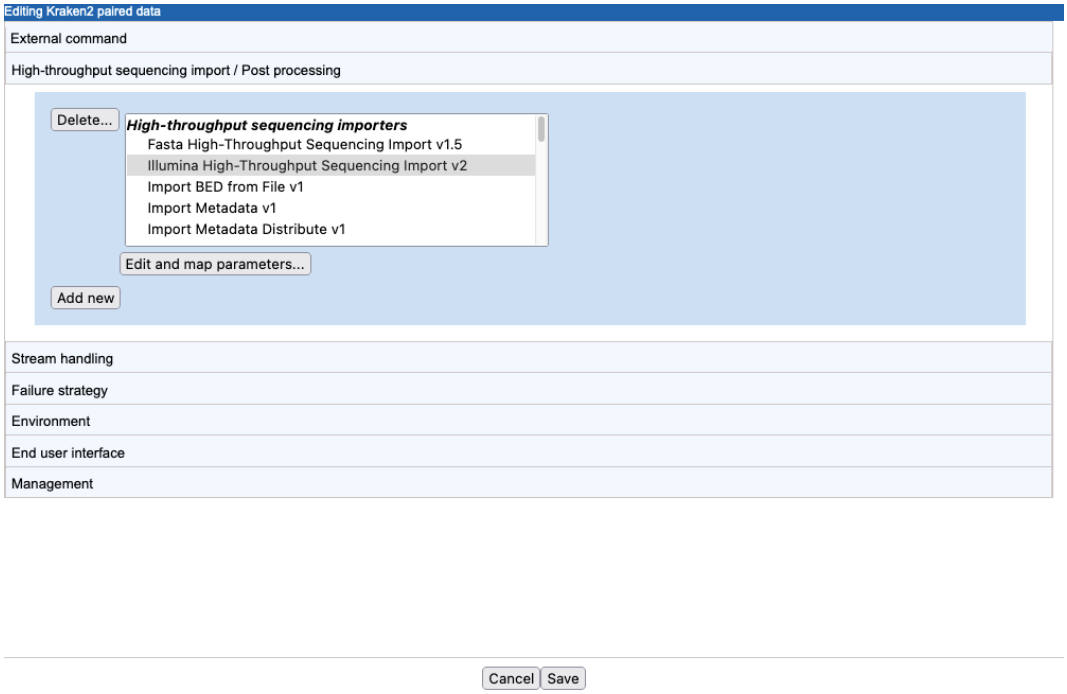


Figure 15.40: *A new import/post processing element has been created, and the Illumina high-throughput sequencing importer has been selected.*

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 15.42). If you don't see this immediately, save the external application configuration and re-open it.

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.

The `Kraken2` command line is described at
`https://github.com/DerrickWood/kraken2/wiki/Manual`.

The other changes in the script are to handle the increased number of files being output by Kraken2 (figure 15.43).

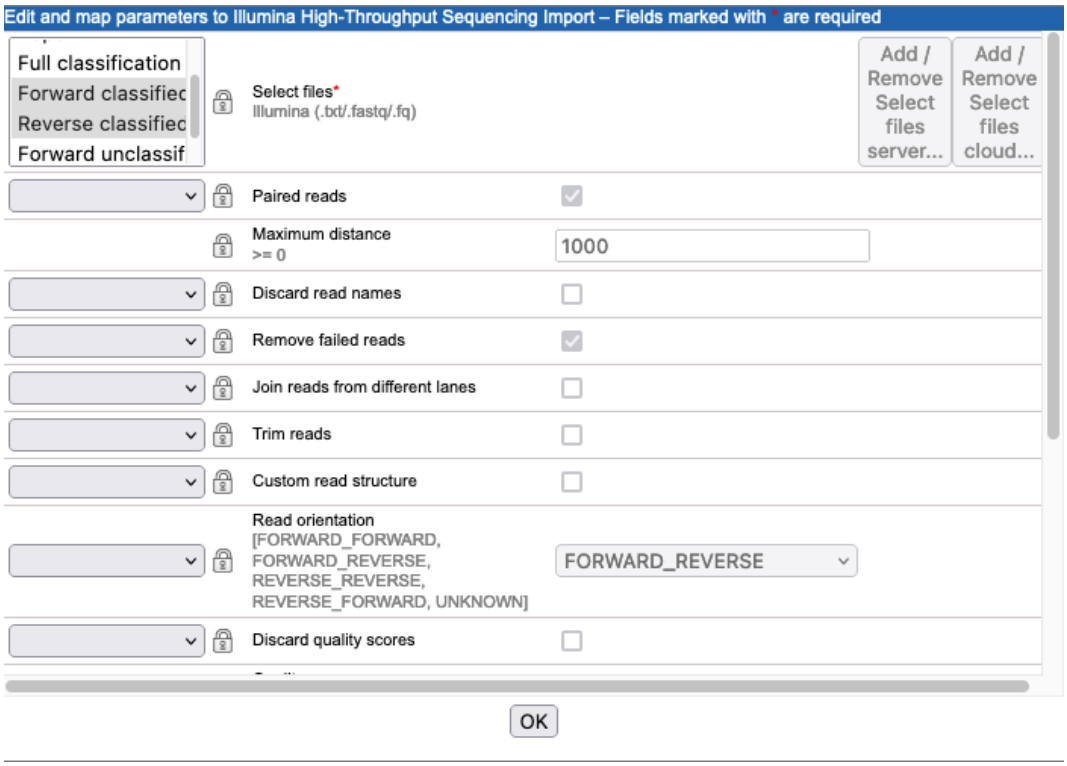


Figure 15.41: *The Illumina High-Throughput Sequencing Import configuration. Here, the parameters corresponding to the 2 FASTQ files containing classified sequences have been selected. values.*

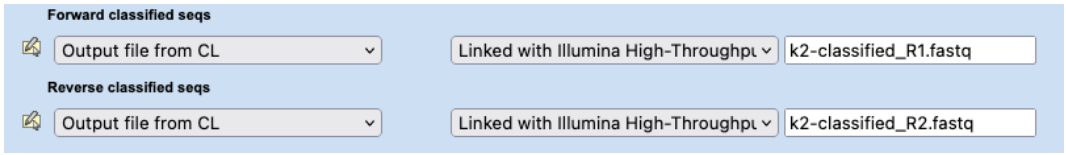


Figure 15.42: *The information in the "General configuration" area is updated to reflect the link between output parameters and the Illumina importer.*

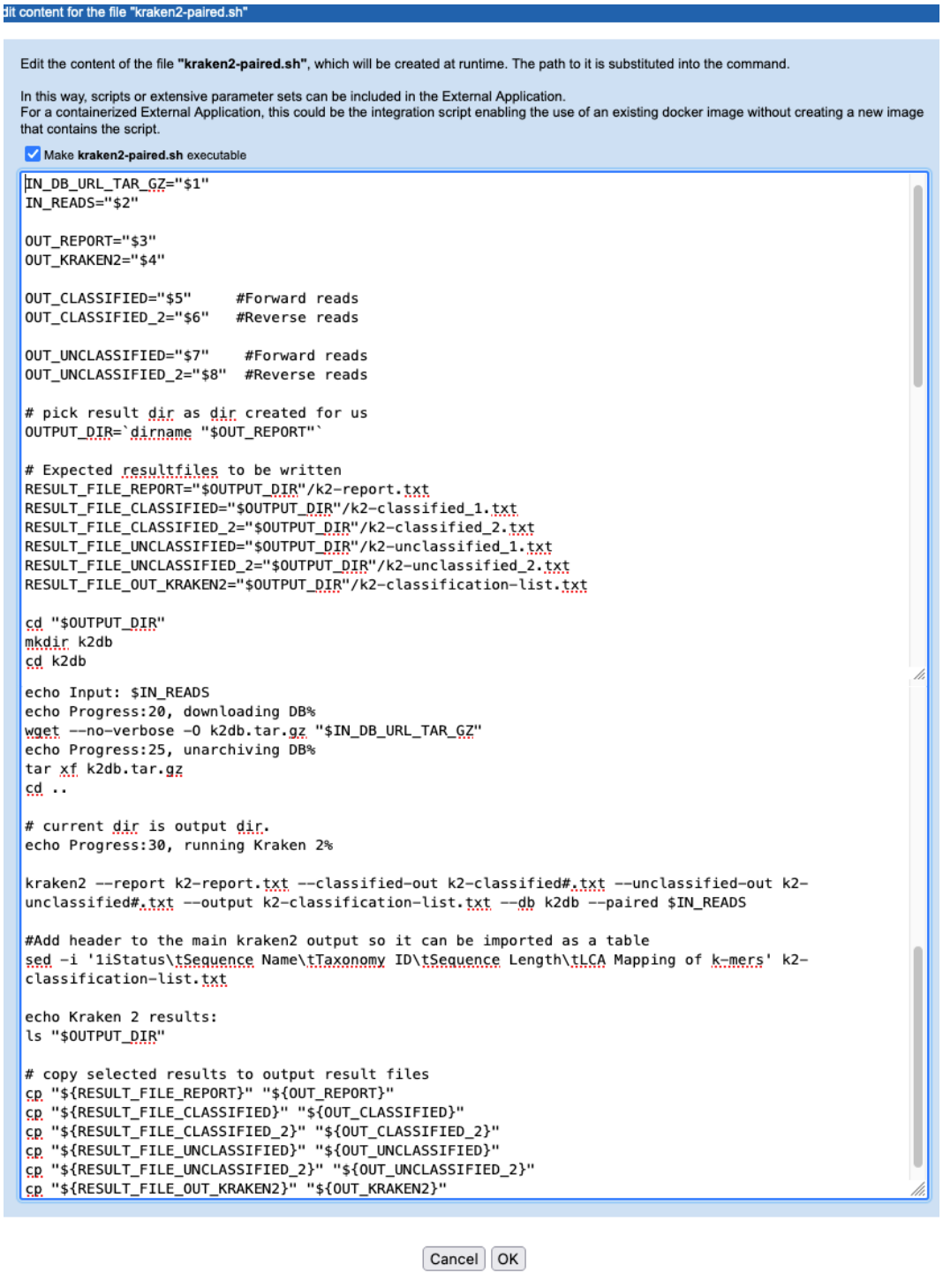


Figure 15.43: *The kraken2-paired.sh script, with the Kraken2 command updated to indicated that the data to analyze is paired, and with variables added to handle paired sequence data being passed to the script and out of the script.*

15.9 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/.

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 15.44.

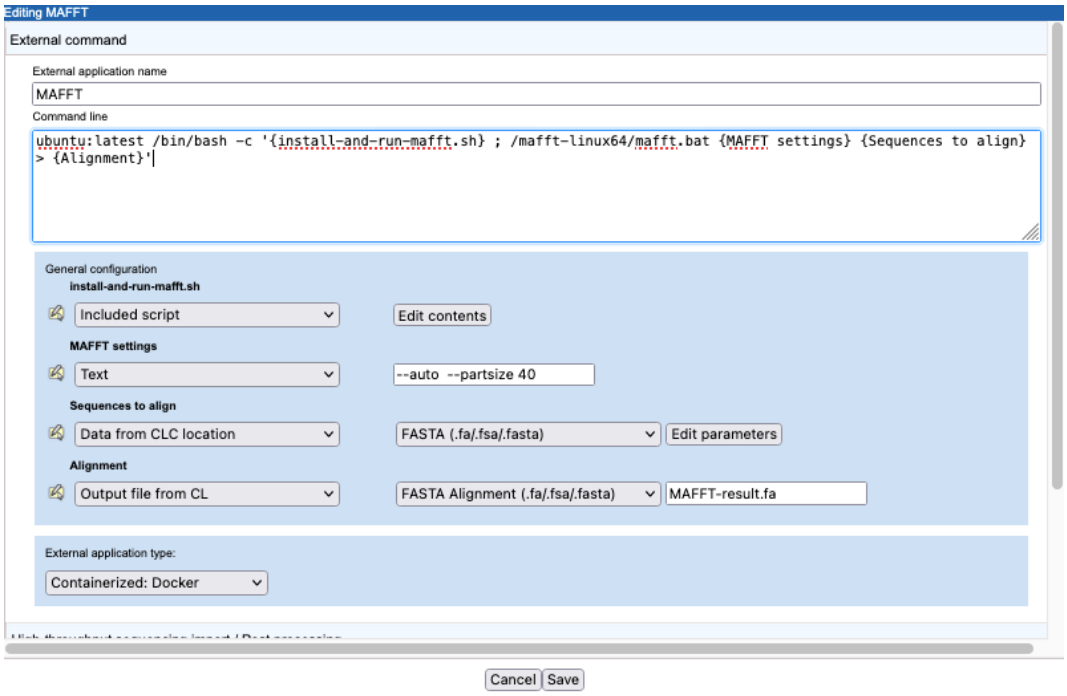


Figure 15.44: *Configuration a containerized external application for MAFFT. A publicly available Ubuntu image is pulled and all subsequent information in the command is run in the container created.*

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 15.49).

  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 15.8 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 15.45).

  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 15.46. 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 15.2.2 for more information about external application parameter types.

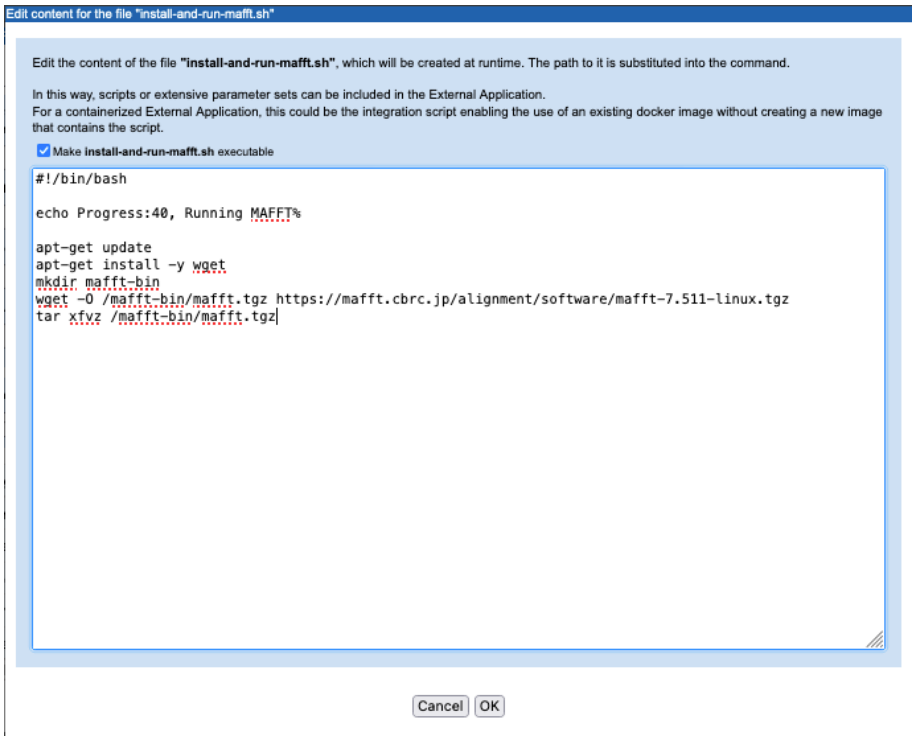


Figure 15.45: *A script is defined that will be run in the Docker container. It includes the steps needed to make MAFFT available to run in the container.*

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 15.47.

- **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".

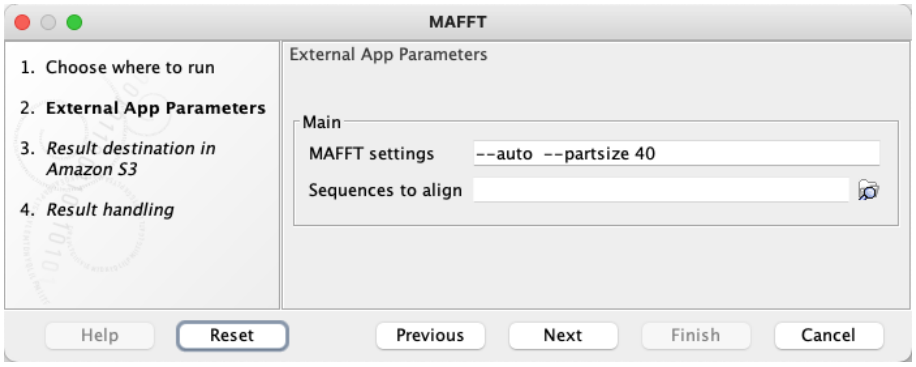


Figure 15.46: *The wizard presented to Workbench users when they launch the MAFFT external application. They select the sequences they wish to align, and can, if they wish, edit the options being passed to MAFFT.*

- **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 15.48).

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 15.48). "Sequences to align", will be used in the *CLC Workbench* wizard (figure 15.46) 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 its status is set to Disabled.

To run external applications on the cloud, the *CLC Cloud Module* is needed. See https://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.

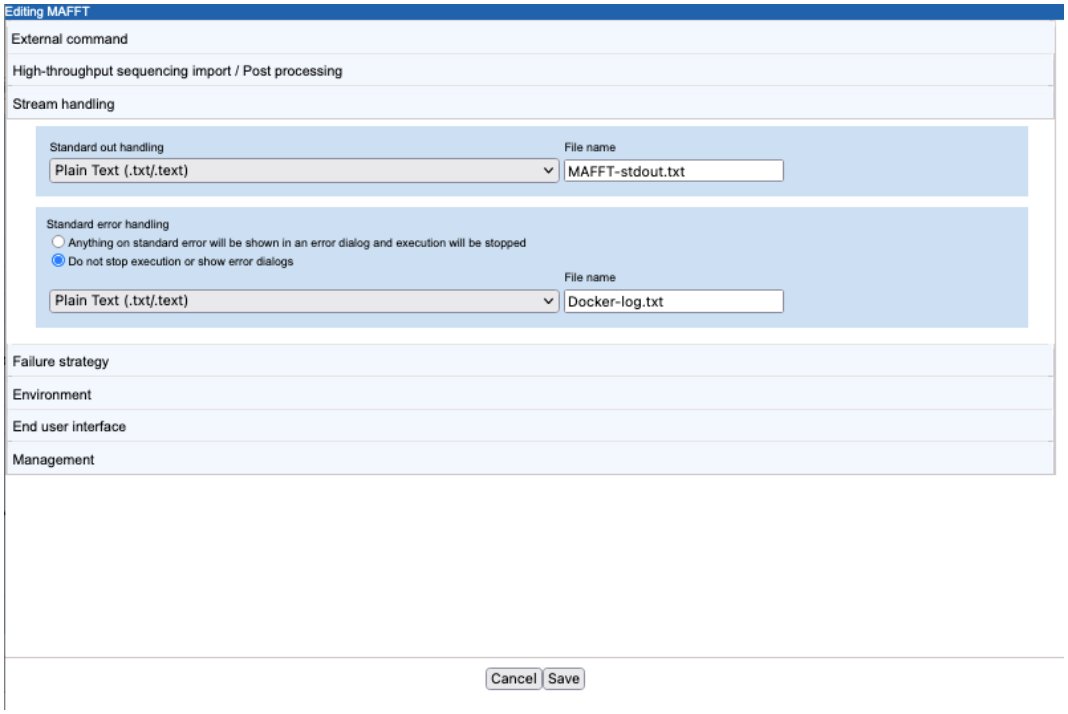


Figure 15.47: *Defining stream handling for the the MAFFT containerized external application.*

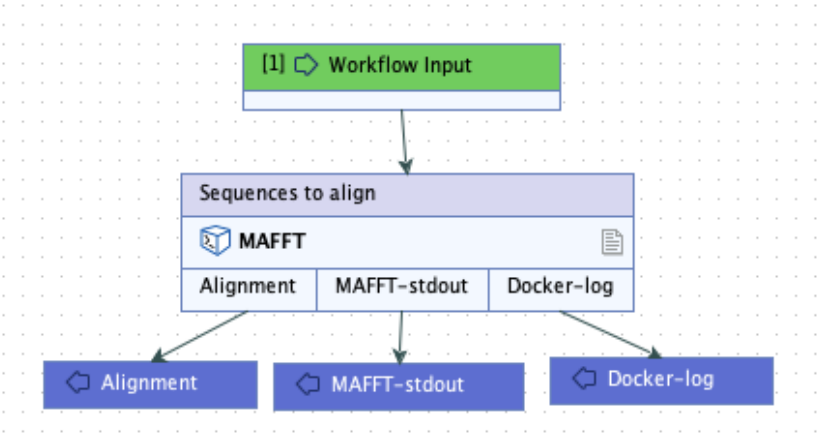


Figure 15.48: *A workflow containing the external application. In workflows, the outputs to collect can be specified. Here, all the outputs are configured to be saved.*

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 https://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 15.49.

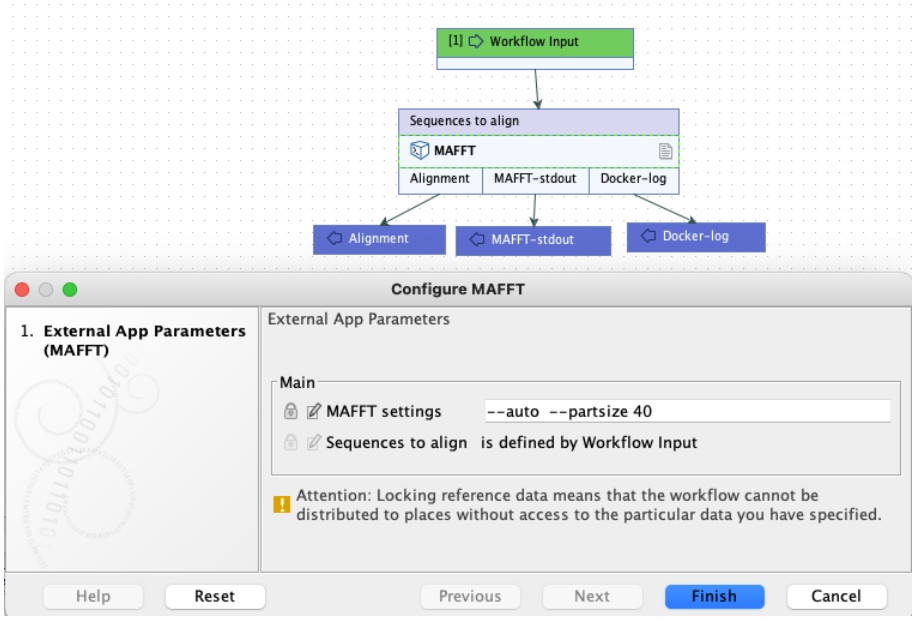


Figure 15.49: *By default, parameters are locked in workflow elements, as shown here. To allow users to configure the "MAFFT settings" option when launching this workflow, the parameter must be unlocked.*

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 15.50).

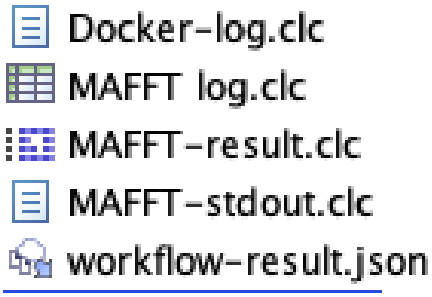


Figure 15.50: *Outputs from a MAFFT containerized external application after running it on AWS using functionality provided by the CLC Cloud module.*

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.

Interacting with files on AWS S3 via a *CLC Workbench* is described at https://resources.qiagenbioinformatics.com/manuals/clccloudmodule/current/index.php?manual=Working_with_AWS_S3_using_

Remote_Files_tab.html. Interacting with files on AWS S3 via the *CLC Server* web interface is described in section 7.5.

15.10 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 15.51, 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 15.51, 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

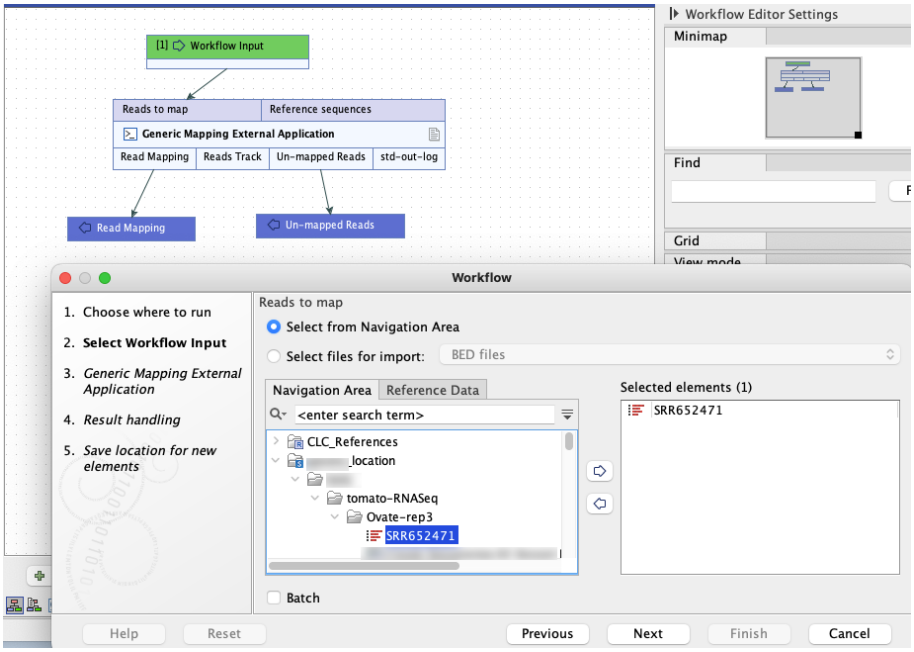


Figure 15.51: *An external application workflow element with default naming. Names are reflected in the launch wizard, both in the steps listed on the left hand side and in the main part of the wizard.*

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 15.52). 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

Please refer to the Workflow chapter of the *CLC Genomics Workbench* manual for information about workflows: https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Workflows.html.

The following are pages within that chapter that contain information specifically referred to above:

- How to add elements to a workflow: https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Adding_elements_workflow.html
- Anatomy of a workflow element: https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Anatomy_workflow_elements.html
- Renaming elements and locking and unlocking workflow parameters: https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=Basic_configuration_workflow_elements.html
- Configuring input and output elements, including renaming: https://resources.qiagenbioinformatics.

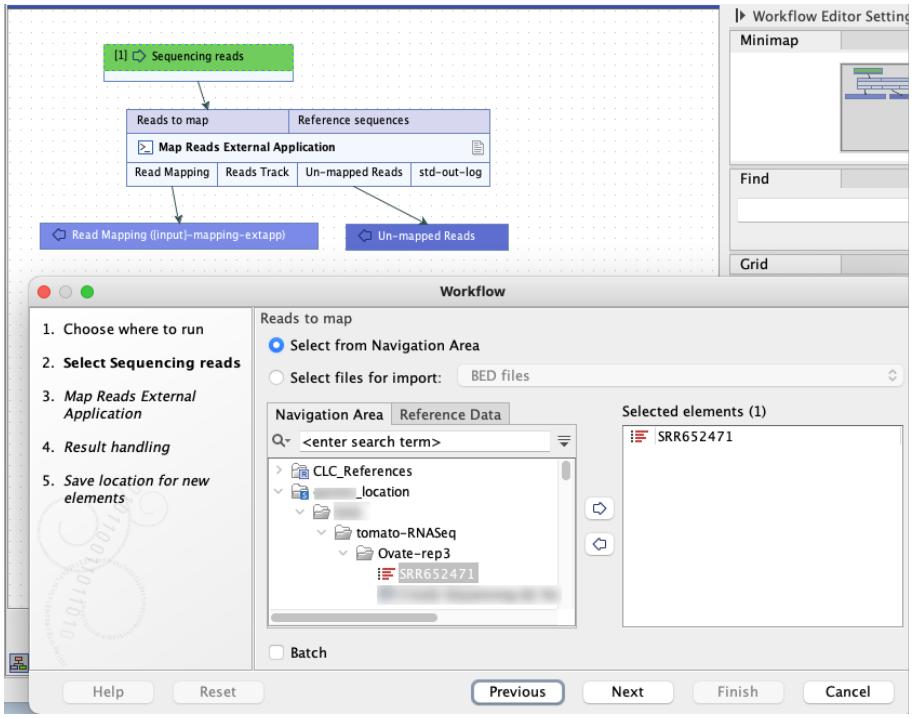


Figure 15.52: *The name of this external application workflow element was updated, as was the name of the workflow Input element (green box) and the name of the mapping output. The element name and input name changes are reflected in the launch wizard.*

com/manuals/clcgenomicsworkbench/current/index.php?manual=Basic_configuration_workflow_elements.html

15.11 Running external applications

External applications can be executed using a *CLC Workbench* or using the *CLC Server Command Line Tools*.

15.11.1 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 15.53. 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 15.54 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 15.55).

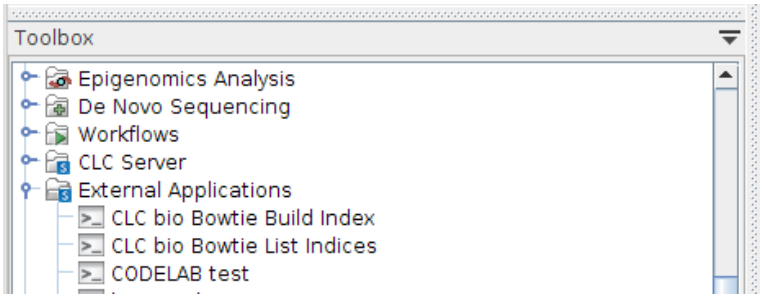


Figure 15.53: *Selecting the external application to run.*

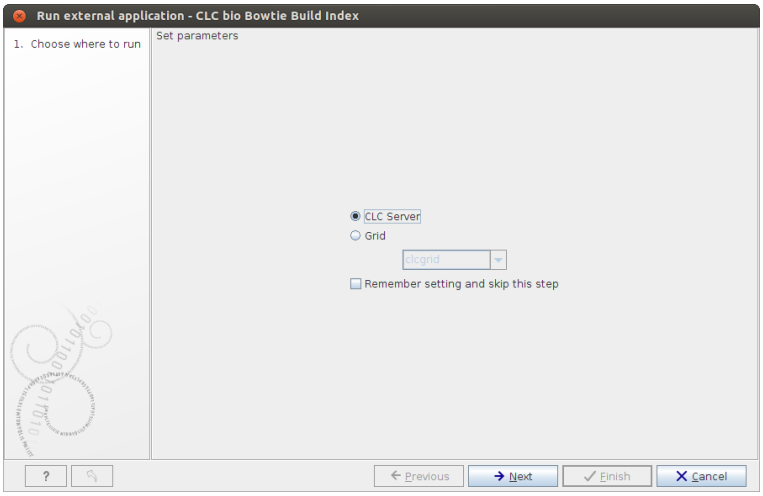


Figure 15.54: *Selecting execution environment.*

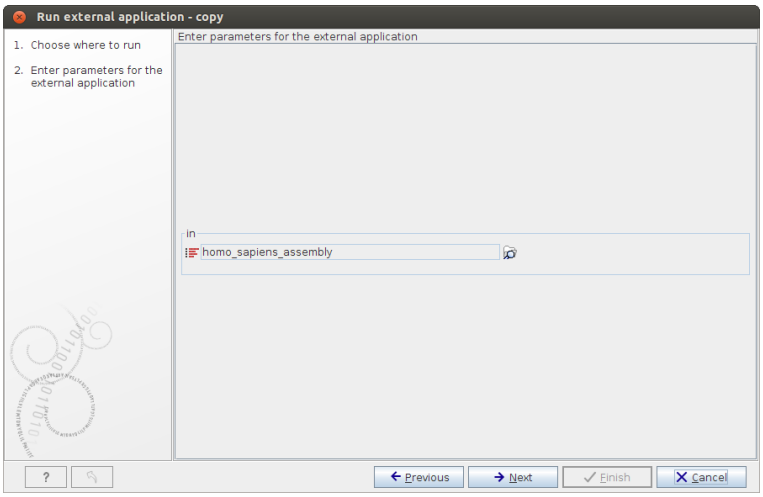


Figure 15.55: *Configurable parameters of an external application are presented in the wizard. Here, a sequence list has been selected for the "in" parameter.*

Launching external applications as part of a workflow

External applications can be included in workflows, as described in section 15.10. Workflows stored in the Navigation Area can be launched from within the Workflow Editor and installed workflows can be launched from the Workbench Toolbox.

15.11.2 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:

Type                                    Valid input
----                                    -----------
<Integer>                               A decimal number in the range
[-2147483648;2147483647]
Example: 42
<Boolean>                               The string true or false
Example: true
<String>                                Any valid string. It is recommended
to enclose all strings in '' to
avoid issues with the shell
misinterpreting spaces or double
quotes
Example: 'text="My text"'
<ClcFileUrl>                            A valid path to a file on the server
or in the local file system
Example: clc://serverfile/tmp/export
<ClcObjectUrl>                          A valid path to a Clc object on the
server or locally
Example: clc://server/pstore1/Variant1

Option                                  Description
------                                  -----------
-A <Command>                            Command currently set to 'copy'
-C <Integer>                            Specify column width of help output.
-D <Boolean>                            Enable debug mode (default: false)
-G <Grid preset names>                  Specify to execute on grid.
-H                                      Display general help.
-O <File>                               Output file.
-P <Integer>                            Server port number. (default: 7777)
-Q <Boolean>                            Quiet mode. No progress output.
(default: false)
-S <String>                             Server hostname or IP-address of the
CLC Server.
-U <String>                             Valid username for logging on to the
CLC Server
-V                                      Display version.
-W <String>                             Clear text password or domain
specific password token.
-d, --destination <ClcServerObjectUrl>  Destination for import from External
Application
--in <ClcServerObjectUrl>               Model object(s) to be exported to
FASTA (.fa/.fsa/.fasta)
Error: Missing required options: d, in
```

15.12 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 15.56. For containerized external applications, messages from docker are posted to standard error, so collecting standard error for such applications is particularly recommended.

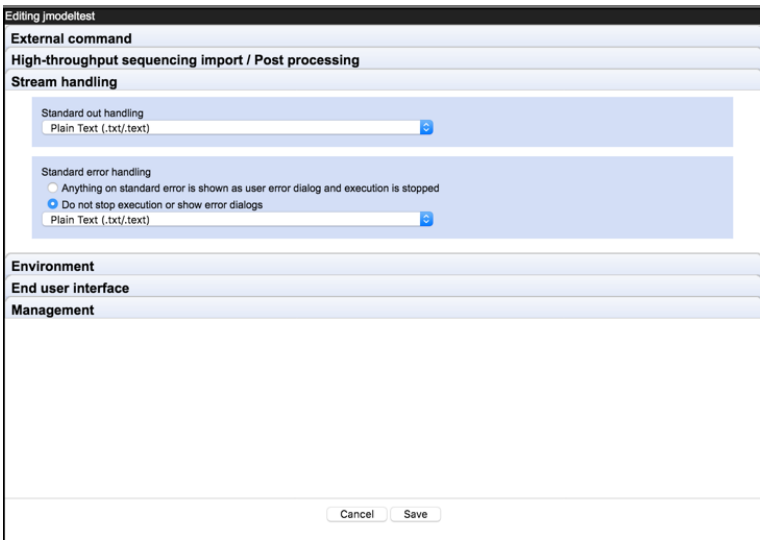


Figure 15.56: *Standard error and standard out from the command line application can be collected and imported so it can be reviewed.*

- 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 15.2.6. 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 executation environment settings, as described in section 15.1.1. 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 16

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.

Prerequisites for CLC Genomics Cloud access

To submit jobs to a *CLC Genomics Cloud* setup via a *CLC Server*, the following is required:

- The *Cloud Server Plugin* has been installed.
- An AWS connection has been configured using credentials giving access to an AWS account with AWS Batch queues for a *CLC Genomics Cloud* setup (see sectio 7.3) .
- At least one cloud preset has been configured.

The rest of this section focuses on configuring cloud presets. For other information about *CLC Genomics Cloud*, see https://digitalinsights.qiagen.com/plugins/clc-cloud-module/.

CLC Genomics Cloud presets

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 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.

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 System 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. Of note:

- A particular user's recycle bin can only be accessed and emptied by themselves and by administrative users. Permission to empty recycle bins can be restricted to just administrators (see section 17.3).
- Permissions applied to the data prior to deletion are no longer in effect on elements in a recycle bin.
- It is not possible to grant other users permission to access data in your recycle bin.

Note: A recycle bin without a user name beside it is present for backwards compatibility with software versions predating the concept of a per-user recycle bins.

The CLC_References location: Deleting data from this *CLC Server* location is different than other locations and does not involve recycle bins. Please refer to section 3.3.2 for details.

Further documentation about working with *CLC Server* recycle bins:

- Automatic recycle bin cleanup: section 17.1
- Emptying recycle bins: section 17.2
- Restricting the emptying of recycle bins to administrators: section 17.3

17.1 Automatic cleanup of recycle bins

Automatic recycle bin emptying can be configured for each File System Location by going to

Configuration ( ) | **Main ( )** | **Recycle bin settings** | **Automatic recycle bin cleanup**

Click on the **Configure** button beside a File System Location (figure 17.1) to enable this functionality, and to configure the minimum age of data to delete and the frequency that recycle bins should be checked.

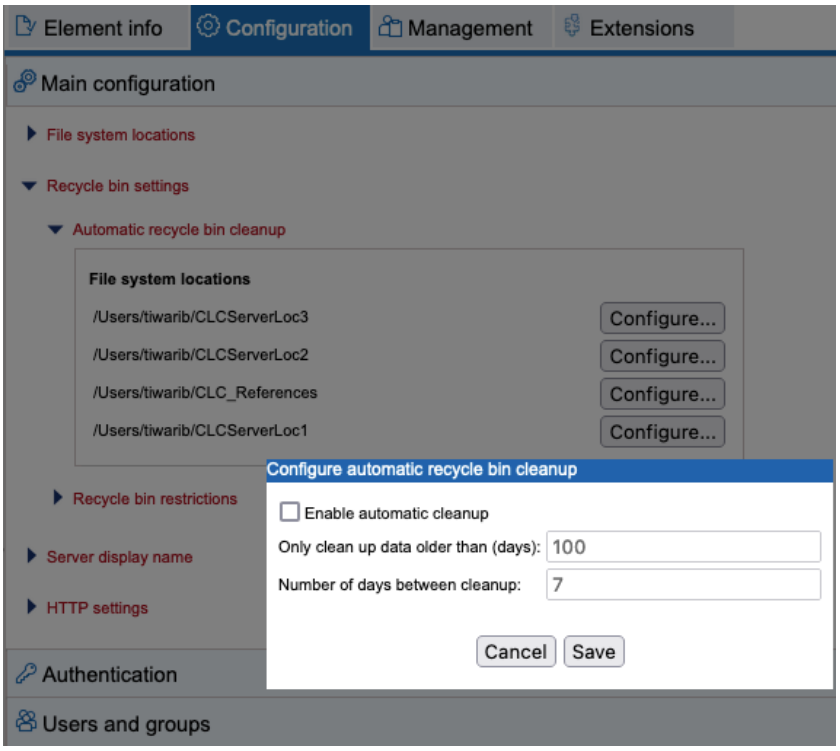


Figure 17.1: *Click on the Configure button beside a location to enable and configure automatic cleanup of the recycle bins in that area.*

17.2 Emptying recycle bins

Emptying recycle bins using the web interface

When logged into the web interface, go to the **Element info ( )** tab. Recycle bins you have access to are listed at the bottom of each File System Location in the Navigation Area, on the left. Administrators will see all recycle bins. A given user will see only their own recycle bin.

To empty a given recycle bin, click on that recycle bin in the Navigation Area. Then click on the **Empty Recycle Bin...** button in the **Info** tab, on the right.

Administrators can empty all recycle bins for a given File System Location by clicking on that location in the Navigation Area, and then clicking on the **Empty All Recycle Bins...** button that will be visible under the *Info* tab (figure 17.2).

You will be asked for confirmation before contents of recycle bins are deleted. The action of emptying a recycle bin cannot be undone.

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.

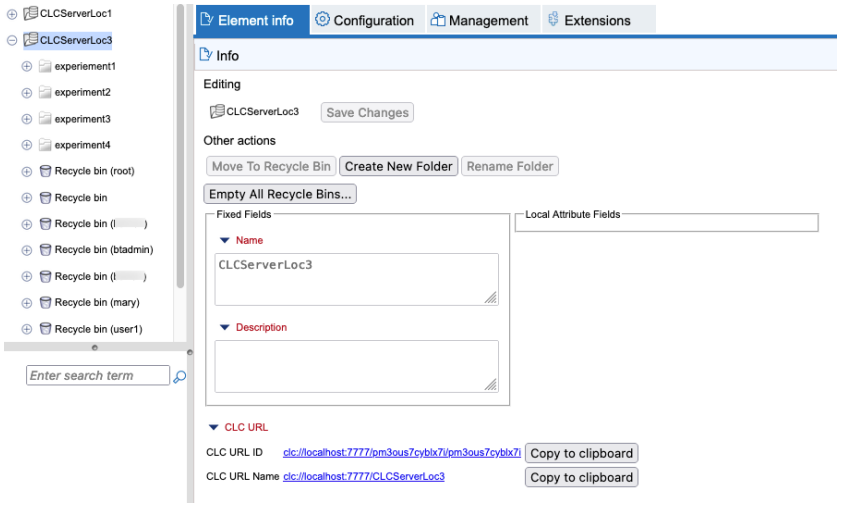


Figure 17.2: *An administrator has clicked on the CLCServerLoc3 location and could now empty all recycle bins in that location by clicking on the Empty All Recycle Bins button under the Info tab, on the right.*

Emptying recycle bins using a CLC Workbench

When logged in to the *CLC Server* from the *CLC Workbench*, the recycle bins you have permission to see are listed at the bottom of each CLC Server File System Location in the Navigation Area (figure 17.3).

To empty specific recycle bin, right-click on it and select the **Empty Recycle Bin** option in the menu that appears.

Administrators can see or can empty all recycle bins in a given File System Location by right-clicking on that location in the Navigation Area, then selecting from the menu that appears:

Location | **Show All Recycle Bins**

or

Location | **Empty All Recycle Bins**

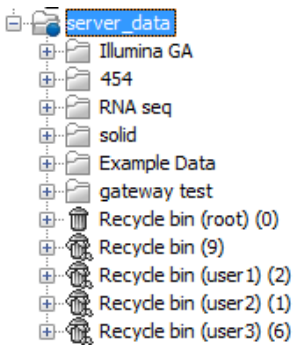


Figure 17.3: *Showing all recycle bins for a particular server File System Location.*

The action of emptying a recycle bin cannot be undone.

17.3 Recycle bin restrictions

By default, a user can empty their own recycle bins. To change this so that only administrators can empty recycle bins, go to:

Configuration | **Main** | **Recycle bin settings** | **Recycle bin restrictions**

and enable the "Only admins can empty recycle bins" option (figure 17.4).

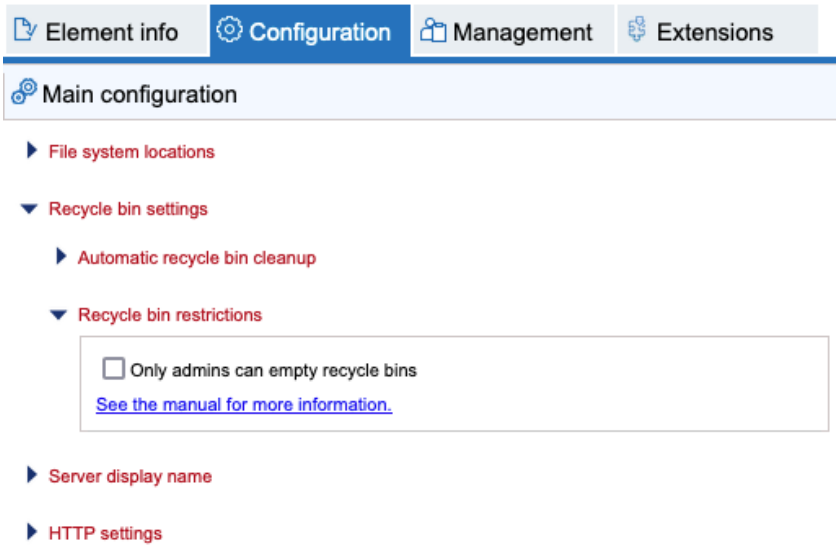


Figure 17.4: *Enable the option under Recycle bin restrictions to restrict the emptying of recycle bins to administrators. By default the option is not enabled, as shown here, so users can delete the contents of their own recycle bins.*

Intermediate workflow result handling and data deletion

To avoid any risk of users deleting data held on a *CLC Server*, we recommend that, in addition to enabling the option described above, the intermediate workflow result handling option be set to **In a temporary directory** (see section 6.5.3).

The other option is to store intermediate workflow results in subfolders of the location the final results will be stored in. With that option, a user could, by accident or design, move data into a intermediate workflow results subfolder, which would result in that data being deleted when the workflow intermediate results are deleted.

Chapter 18

Configuring CLC Workbench connections

To log into a *CLC Server* from a *CLC Workbench*, the server name and port, as well as a username and password need to be configured in the Workbench. This can be done directly using a configuration dialog in the Workbench, or by providing a settings file with the connection details. The latter can be useful for deploying server connection details to multiple Workbench installations.

Configuring the server connection using the CLC Server Connection dialog is described at https://resources.qiagenbioinformatics.com/manuals/clcgenomicsworkbench/current/index.php?manual=CLC_Server_connection.html.

Configuring server connection information using a property file is described at https://resources.qiagenbioinformatics.com/manuals/workbenchdeployment/current/index.php?manual=Connecting_CLC_Server.html.

Chapter 19

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 6.3.14.
- Troubleshooting **External Applications** is covered in section 15.12.

19.1 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 19.1). Clicking on the warning message at the top right opens a diagnostic report, where details of the problems can be found (figure 19.2). 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.

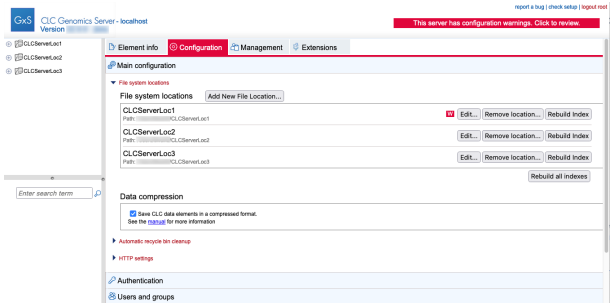


Figure 19.1: *If issues with the server configuration have been found, a red band and warning message is presented. Click on the warning message to open the diagnostic report. The red icon beside the first file system location also provides a signal that there is a problem.*

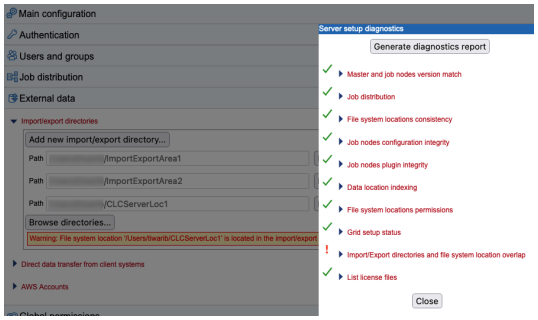


Figure 19.2: *Tests that passed are marked with a green check mark. Tests that failed are marked with a red exclamation mark. Here, a problem with the setup of import/export directories has been found. Such an issue is also indicated in the relevant configuration area, as seen in the background here.*

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 19.3.
- 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 8.2.1.
- 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.

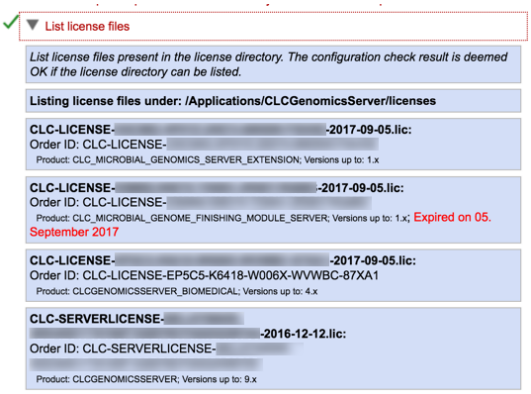


Figure 19.3: *Click on the List license files item in the report to see the list of the license files found in the licenses subfolder of the installation area. Products and versions supported are reported, and any expired license is noted with red text.*

19.2 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 19.4. 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.

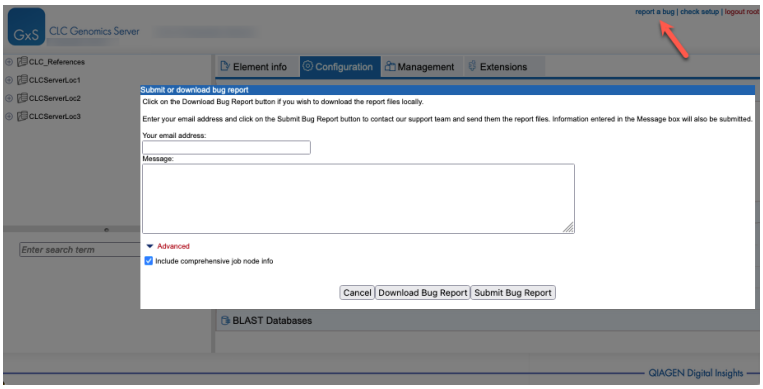


Figure 19.4: *Submitting a bug report.*

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 20

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:

- **html:** http://resources.qiagenbioinformatics.com/manuals/clcservercommandlinetools/current/
- **pdf:** http://resources.qiagenbioinformatics.com/manuals/clcservercommandlinetools/current/User_Manual.pdf

Chapter 21

Appendix

21.1 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

21.2 Non-exclusive Algorithms

Tools categorized as *non-exclusive* or *streaming* are listed in table below. Non-exclusive jobs have low resource demands and can be run concurrently on an execution node. Streaming jobs are I/O intensive. On single servers or job nodes, these can be run alongside non-exclusive tools, but not alongside other streaming tools on a given execution node. Exclusive tools always run alone. On grid nodes, streaming tools are treated as exclusive, and thus are always run alone.

Documentation about configuring simultaneous job execution:

- For job nodes, see section 6.2.3.
- For grid setups, see section 6.3.10.
- For single servers, see section 6.4.

Tool	Streaming
Add attB Sites	
Amino Acid Changes	X
Annotate from Known Variants	X
Annotate with Conservation Scores	X

Tool	Streaming
Annotate with Exon Numbers	X
Annotate with Flanking Sequences	X
Annotate with Nearby Gene Information	
Annotate with Overlap Information	X
Annotate with Repeat and Homopolymer Information	
Apply Peak Shape Filter	
Assemble Sequences	
Assemble Sequences to Reference	
BLAST at NCBI	
Call Methylation Levels	
ChIP-Seq Analysis	
Convert DNA To RNA	X
Convert from Tracks	X
Convert RNA to DNA	X
Convert to Tracks	X
Copy Number Variant Detection (CNVs)	
Count-based statistical analysis	
Create Alignment	
Create BLAST Database	
Create Box Plot	
Create Entry Clone (BP)	
Create Expression Browser	
Create Expression Clone (LR)	
Create GC Content Graph Track	
Create Histogram	
Create K-medoids Clustering for RNA-Seq	
Create MA Plot	
Create Mapping Graph Track	
Create RRBS-fragment Track	
Create Sample Report	
Create Sequence Statistics	
Create Track from Experiment	
Create Track List	
Create Tree	
Create Venn Diagram for RNA-Seq	
Demultiplex Reads	
Download 3D Protein Structure Database	X
Download BLAST Databases	X
Download Pfam Database	X
Extract Annotations	
Extract Consensus Sequence	
Extract IsomiR Counts	
Extract Reads	
Extract Sequences	X
Filter against Known Variants	X
Filter Annotations on Name	X
Filter Based on Overlap	X

Tool	Streaming
Find and Model Structure	X
Find Binding Sites and Create Fragments	
Find Open Reading Frames	
Gaussian Statistical Analysis	
Gene Set Test	
GO Enrichment Analysis	
Hierarchical Clustering of Samples	
High-throughput sequencing importers (e.g. Illumina)	X
Identify Enriched Variants in Case vs Control Samples	X
Identify Graph Threshold Areas	
Identify Known Mutations from Mappings	
Import Primer Pairs	
Import SAM/BAM/CRAM Mapping Files	X
Import Tracks from File	
InDels and Structural Variants	
Learn Peak Shape Filter	
Link Variants to 3D Protein Structure	X
Merge Annotation Tracks	X
Merge Overlapping Pairs	
Merge Read Mappings	X
Merge Variant Tracks	
Motif Search	
Pfam Domain Search	X
Predict Splice Site Effect	
Principal Component Analysis	
Probabilistic Variant Detection	
Proportion-based Statistical Analysis	
Quality-based Variant Detection	
Quantify miRNA	
QC for Read Mapping	
QC for Sequencing Reads	X
QC for Targeted Sequencing	
Remove Duplicate Mapped Reads	
Remove Homozygous Reference Variants	
Remove Information from Variants	X
Remove Marginal Variants	X
Remove Variants Present in Control Reads	X
Rename Sequences in Lists	
Reverse Complement Sequence	
Subsample Sequence List	
Secondary Peak Calling	
Score Regions	
Split Sequence List	
Transcription Factor ChIP-Seq	
Translate to Protein	
Trim Sequences	
TRIO analysis	

Tool	Streaming
Update Sequence Attributes in Lists	
Whole Genome Coverage Analysis	

21.3 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.

21.3.1 DRMAA for SLURM

Information about DRMAA for SLURM can be found at https://slurm.schedmd.com/download.html.

21.3.2 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.

21.3.3 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 DRMMA 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`

21.3.4 DRMAA for OGE or SGE

OGE/SGE comes with a DRMAA library.

21.4 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.<clustername>:

```
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.

21.5 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

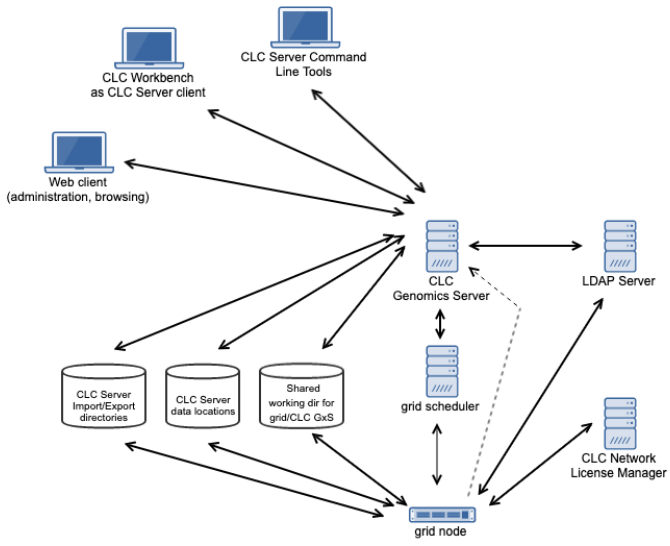


Figure 21.1: *A representative CLC Server setup, where job execution is carried out on grid nodes, and authentication has been configured to use LDAP. The dotted line between grid node and the master CLC Server represents the communication that triggers an index update when a job has created files in a CLC File System Location.*

Representative CLC Server grid setup using built-in authentication

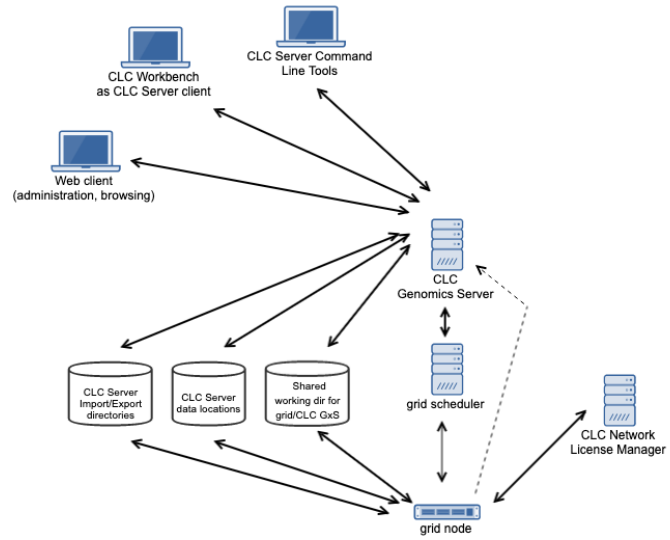


Figure 21.2: *A representative CLC Server setup, where job execution is carried out on grid nodes, and the built-in authentication mechanism is being used. The dotted line between grid node and the master CLC Server represents the communication that triggers an index update when a job has created files in a CLC File System Location.*

Representative CLC Server job node setup with LDAP authentication

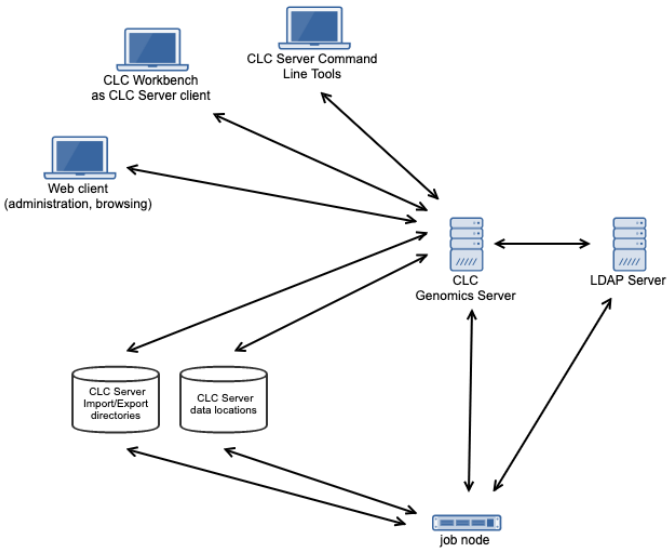


Figure 21.3: *A representative CLC Server setup, where job execution is carried out on job nodes, and and authentication has been configured to use LDAP.*

Representative CLC Server job node setup using built-in authentication

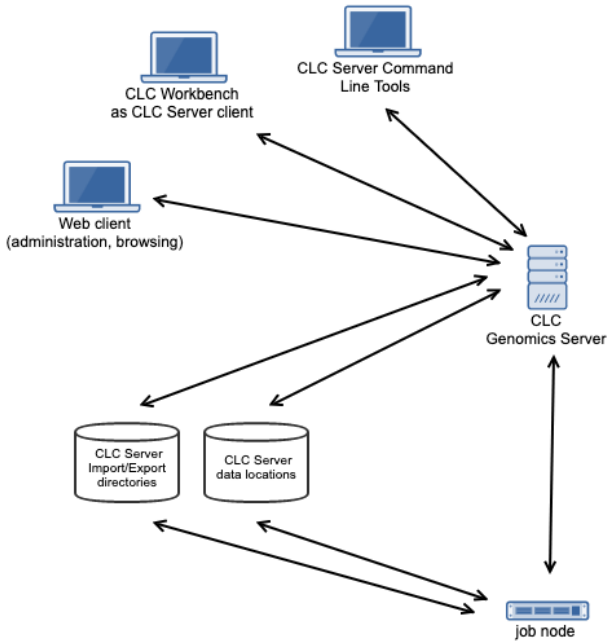


Figure 21.4: *A representative CLC Server setup, where job execution is carried out on job nodes, and the built-in authentication mechanism is being used.*

21.6 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.

21.7 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.

21.8 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.

21.8.1 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:

```
monitorRole readonly
controlRole readwrite
```

3. Create the password file `CLC_SERVER_BASE/conf/jmxremote.password` and write the following in it:

```
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.

21.8.2 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.

21.8.3 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 sence 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 umber 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.

21.8.4 Job node metrics

Object name: `com.clcbio.server:type=JobNodes`
`com.clcbio.server:type=JobNodes,name=<job node name>`

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 sence 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 Hight 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

[Langmead et al., 2009] Langmead, B., Trapnell, C., Pop, M., and Salzberg, S. L. (2009). Ultrafast and memory-efficient alignment of short dna sequences to the human genome. *Genome Biol*, 10(3):R25.

[Zerbino and Birney, 2008] Zerbino, D. R. and Birney, E. (2008). Velvet: algorithms for de novo short read assembly using de Bruijn graphs. *Genome Res*, 18(5):821–829.

Index

Active directory, 45
AD, 45
AWS S3, 90
AWS S3, Public buckets, 92

BaseSpace, 95
BLAST, 108

Command-line installation, 19
Communication, diagrammatic, 202
Concurrent jobs per node, 65, 81
Configuring setup, 61
    Master node for grid setups, 69
consumable resource, 200
Cores, restrict usage, 38
CPU, restrict usage of, 38

Data
    deletion of, 185
Data compression, 33
Data locations
    Data compression, 33
    File system locations, 28, 31
    Job nodes, 31
    Reference data locations, 32
Deleting data, 185
Direct data transfer settings, 89
DRMAA, 199

Encrypted connection, 34
External applications, 123
    Running from a Workbench, 178
    Running using the command line, 180

GSSAPI, 44

HTTPS, 34

Index, rebuilding, 98
Installing Server plugins on job nodes, 66

JMS, 205

Kerberos, 44

LDAP, 44
License
    non-networked machine, 22

Memory allocation, 37
Metrics, 205
Monitoring, 205
Multi-job processing on grid, 76

permissions, 48
proxy, 38

Quiet installation, 19

RAM, 37
Read mapping
    Reference cache, 204

Secure socket layer, 34
Silent installation, 19
SSL, 34
Status and management, 111
Status and running modes, 113
Surveillance, 205
System requirements, 10
System statistics, 115

Third party libraries, 204
tmp directory, how to specify, 37

Upgrading an existing installation, 20
User credentials, 60
User statistics, 113

.vmoptions, memory allocation, 37

Workflows
    Configuring, 100
    Installing, 100
    Running, 102
    Updating, 104

Xmx argument, 37