

Deployment of Leiden Grid Infrastructure

M.F. Somers

H. Meiland

December 2008.

Abstract

In this project the Leiden Grid Infrastructure (LGI) was deployed on several computer resources owned by NCF and the Theoretical Chemistry group of the Leiden Institute of Chemistry of the University of Leiden. The LGI middle-ware was setup on a server based on a Linux-Apache-MySQL-PHP (LAMP) stack and the LGI resource daemons were deployed on Huygens, Lisa, both machines owned by NCF, and Harpertown, Woodcrest, Nocona, two BOINC (Berkeley Open Infrastructure for Network Computing) servers and a test laptop. The applications that were made available through this effort are: Gaussian, ADF, VASP, SA-DVR, SPO-DVR and Classical. The project was granted through NCF grant number NRG-2008.02.

Table of content

| | |
|---------------------------------------|----|
| Abstract..... | 1 |
| Table of content..... | 2 |
| | 2 |
| Introduction..... | 3 |
| Deployment of LGL..... | 4 |
| Setup of project web-server..... | 4 |
| Setup of a resource..... | 5 |
| check_system_limits_script..... | 8 |
| job_check_limits_script..... | 8 |
| job_prologue_script..... | 8 |
| job_run_script..... | 8 |
| job_epilogue_script..... | 9 |
| job_check_running_script..... | 9 |
| job_check_finished_script..... | 9 |
| job_abort_script..... | 9 |
| Setup of command line interfaces..... | 10 |
| Conclusions..... | 10 |
| References..... | 11 |
| Appendix A..... | 12 |
| Appendix B..... | 15 |
| Appendix C..... | 18 |
| Appendix D..... | 25 |
| Appendix E..... | 26 |

Introduction

In this part of the report, the LGI middle-ware is briefly described. The LGI middle-ware is especially developed to allow application oriented groups to setup a GRID of computer resources with machines within and out-of their administrative domain. LGI especially allows for enabling applications on a computer GRID for which certain license limits apply or which are not easy to adapt (because the source code is not available). The basic design of the middle-ware is shown in Figure 1:

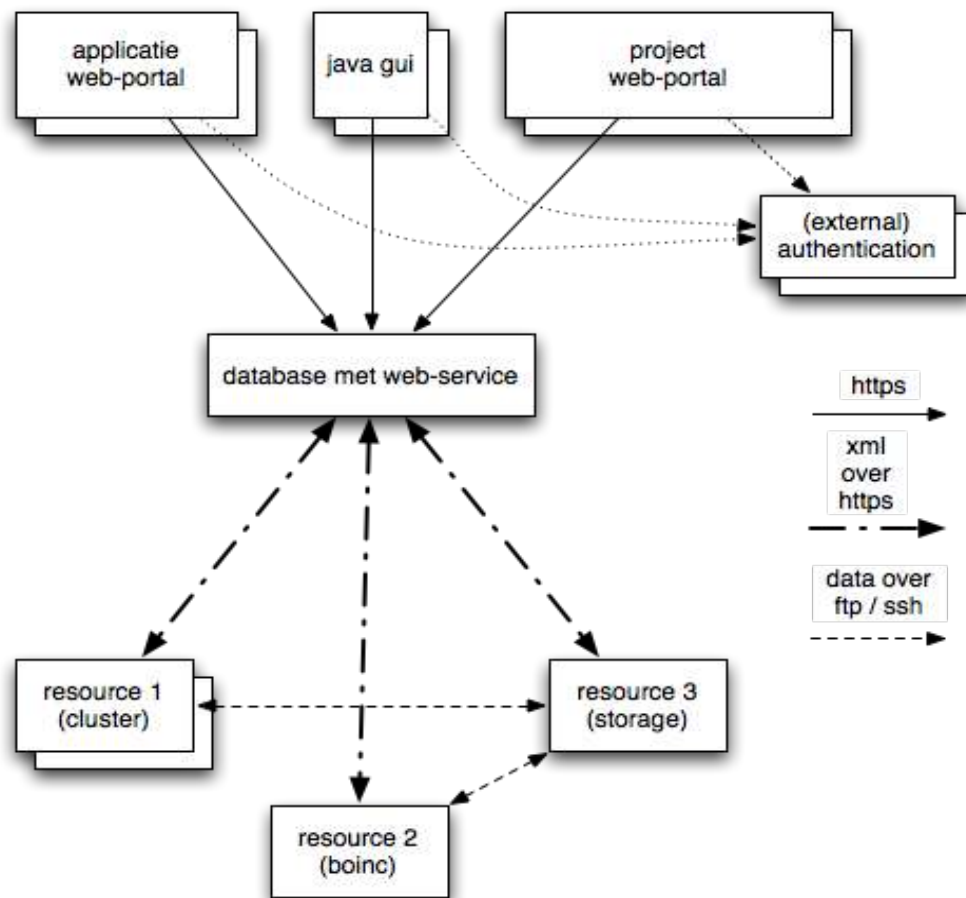


Figure 1: The basic design of LGI.

Users interact with the 'project web-server' and submit a calculation for an application to it using an 'application-interface'. Currently two general application-interfaces have been developed, a command line driven interface which supports commands very much like OpenPBSs 'qsub', 'qstat' and 'qdel', and a web-based interface. In appendices A and B, screen-shots are shown of these two interfaces demonstrating how they can be used.

Resources within LGI regularly poll the project web-server to request work. This is achieved by running a 'resource daemon' on the resource. The daemon is part of the LGI middle-ware and can be compiled for any POSIX system. The daemon was written in standard C++ and only requires the libCurl library together with the STL (Standard Template Library). Resources are easily configured by an XML configuration file. In this configuration file, user access control can be enforced and applications or projects can be enabled or configured by supplying the correct back-end scripts. In appendix C an example resource-daemon configuration file has been supplied. The resource-daemon runs as a non-privileged user on the resource and only requires out-bound traffic to the project web-server.

The communication between the application interfaces, the resource-daemon and the project web-server makes use of XML over https, using X.509 client- and server-certificates all signed by the trusted LGI-CA (Certificate Authority). This allows for both the resources as well as the application-interfaces to securely communicate with the project web-server and implements a single point of entry to the grid for users without the necessity for any 'proxy-certificates'. Furthermore, because the resource-daemon initiates the communication with the project web-server, the daemon can be run just fine behind a firewall, as a non-privileged user and even through a NAT-ed (Network Address Translation) router.

The external authentication shown in Figure 1 is not needed for LGI but can be implemented if desired for instance in a web-portal serving a web-based interface to a certain application.

Deployment of LGI

In this part of the report the actual deployment of LGI is documented. The basic steps of setting-up a project server are given, the steps to deploy the resource-daemon and how to configure it, examples of back-end script are made available for several applications, and finally how to setup the general command line application interface is described.

Setup of project web-server

Setting up a project web-server starts off by installing a LAMP (Linux-Apache-MySQL-PHP)-stack. The setup of such a stack is documented extensively on the web. The next step is to download the LGI middle-ware itself and follow the instructions given in the supplied SETUP.txt file (see appendix C).

Basically, one has to correctly configure Apache, load a new database into MySQL using the LGI.db file, create X.509 keys and certificates with OpenSSL for resources, for the project web-server and for users, and configure the LGI project server by editing the Config.inc file. All these steps are documented with examples in the SETUP.txt file and the reader is referred to that. Once the project web-server is correctly setup, the general web-based application interface is also immediately available. Users can load their personally issued X.509 certificates to identify themselves in a browser of their choice and start using the interface as is demonstrated in the screen-shots in appendix A.

Setup of a resource

Once a project web-server is up and running, the next step is to add resources into the LGI GRID. The basic steps to that are also documented in the SETUP.txt file included in the LGI middle-ware. In this section extra details and examples are made available. First the basic deployment of the daemon is described, then the applications that have been deployed in his project are briefly described and details are given on how they were deployed on several resources.

The first steps of enabling a resource with LGI is to contact the LGI project administrator and request an X.509 certificate and a private key for your resource. Once this is done, one just downloads the LGI middle-ware and compiles the code located in the 'daemon/src' directory. The code has been configured to compile with the GNU C++ compilers using a 'make file' and assumed libCurl is properly installed already on the resource. LibCurl is a well-known library used by many other programs and is generally part of a basic Linux install already. If it is not available, it can be downloaded from <http://curl.haxx.se/libcurl/>. Installing it is trivial and only involves the invocation of 'configure' and 'make'.

Once the code in daemon/src is compiled by invoking 'make' and 'make install', the daemon is ready to be used and an example configuration with appropriate back-end scripts for an example application are included in the downloaded middle-ware. The example application is called 'hello-world' and the configuration file is LGI.cfg (see appendix D). Just adapt the configuration to the correct situation and invoke the daemon for a test run by running it from the daemon subdirectory: './bin/LGI_daemon LGI.cfg'.

The above basic install serves as a stepping-stone for more complex applications and resources. In this project we deployed several applications (Gaussian, VASP, ADF, SPO-DVR, SA-DVR and Classical) on several resources (Huygens, Lisa, Nocona, Woodcrest, Harpertown, fwnc7003, a laptop and two BOINC servers). In the table below one can see which application was deployed on what resource:

| | Huygens | Lisa | Nocona | Woodcrest | Harpertown | BOINC | fwnc7003 | laptop |
|-------------|---------|------|--------|-----------|------------|-------|----------|--------|
| Gaussian | X | X | X | X | - | - | - | - |
| VASP | X | X | - | X | X | - | - | - |
| ADF | - | - | - | X | X | - | - | - |
| SA-DVR | X | - | - | X | X | - | - | - |
| SPO-DVR | X | - | - | - | - | - | - | - |
| Classical | - | - | - | - | - | X | - | - |
| Hello world | X | X | X | X | X | X | X | X |

Huygens: An IBM pSeries 575, clustered SMP system of 104 nodes of 16 dual core processors (IBM Power6, 4.7 GHz) with 128 Gb or 256 Gb RAM. This machine is owned by NWO-NCF and administrated by SARA and uses a Load Leveler batch manager on a SuSE Linux (SLED 10).

Lisa: A 800 node Beowulf using (E5345) Xeon (quad-core) CPUs with Infiniband. This cluster is owned by NWO-NCF and administrated by SARA and uses the OpenPBS/Torque batch manager on a Debian Linux 4.0.

Nocona: A 38 node Beowulf using two Nocona Xeon CPUs (3 GHz) with Hyper-Threading per node with 8 Gb of RAM and gigabit-ethernet. This cluster is owned and administrated by the Theoretical Chemistry group of Leiden University and uses the OpenPBS batch manager on SuSE 9.3.

Woodcrest: A 30 node Beowulf using E5150 dual core (2.7GHz) and E5345 (2.3GHz) quad core Xeon CPUs per node with 8 Gb or 16 Gb of RAM and gigabit-ethernet. This cluster is owned and administrated by the Theoretical Chemistry group of Leiden University and uses the OpenPBS/Torque batch manager on Scientific Linux 4.3.

Harpertown: A 32 node Beowulf using E5430 (2.7 GHz) quad core Xeon CPUs per node with 16 Gb of RAM and gigabit-ethernet. This cluster is owned and administrated by the Theoretical Chemistry group of Leiden University and uses the OpenPBS/Torque batch manager on CentOS 5.2 Linux.

BOINC: Two servers with two 2.7 Ghz dual core (Dempsey) Xeons with Hyper-Threading and 8 Gb of RAM each. These servers run the 'Leiden Classical' BOINC project and is administrated by the Theoretical Chemistry group of Leiden University using Scientific Linux 4.5.

fwnc7003: A workstation running two Nocona Xeons at 3.0 GHz with Hyper-Threading and 4 Gb RAM. On this workstation the LGI project web-server has been deployed and is administrated by Theoretical Chemistry group of Leiden University using Scientific Linux 4.3.

Laptop: A 10" netbook laptop running Linux on a VIA-M 1.2 GHz CPU with 1Gb RAM. This computer is owned and administrated by M.F. Somers at home using Ubuntu Linux 8.04 LTS.

Gaussian: A well-known quantum chemistry program of Gaussian Inc. It is pre-installed on the machines of NCF and locally we have a binary only copy with a corresponding license for our group members only. Typical Gaussian calculations take about a few days, using 16 Gb of Memory and 64 Gb of scratch disk space using at most four cores.

VASP: A well-known plain-wave density functional program of the Computational Materials Physics group of the Vienna University. For this program a source-code license is available for our group members. Typical VASP calculation takes about a few days using about eight cores or nodes and 2 Gb of memory per core.

ADF: A well-known orbital based density functional program of SCM and the Theoretical Chemistry group of the Free University of Amsterdam. For ADF2007 we have a source-code license available for members of our group. Typical ADF calculation takes about a few days using about eight cores or nodes and 2 Gb of memory per core.

SA-DVR: A molecular quantum dynamics program for simulating diatomic-molecular surface scattering using the symmetry of the surface unit-cell of the Theoretical Chemistry group of the Leiden University. For this code our group has the source-code available as it was developed in the group. Typical SA-DVR calculations take about a week using eight to sixteen cores, uses about 16 Gb of memory and 64 Gb of disk storage.

SPO-DVR: A molecular quantum dynamics program for simulating diatomic-molecular surface scattering of the Theoretical Chemistry group of the Leiden University. For this code our group has the source-code available as it was developed in the group. Typical SPO-DVR calculations take about a week using sixteen or thirty-two cores with 32 Gb of memory and 64 Gb of disk storage.

Classical: A general classical dynamics program especially developed for BOINC and ease of use. Typical Molecular Dynamics runs take about a couple of hours on a single core desktop computer using 64 Mb of memory.

Hello world: A simple test application included in the LGI middle-ware distribution. Typical runs take only minutes.

Given the above applications and computer resources, a very heterogeneous mix of application requirements and hardware were encountered in this LGI-deployment project. Back-end scripts for all these applications for Load Leveler and Torque/OpenPBS were written and are now included in the LGI middle-ware distribution as extra examples.

The most challenging applications were the SA-DVR and the SPO-DVR applications. The reasons for that are, these applications need to make use of resources not owned and administrated by the Theoretical Chemistry group of Leiden, involve several restarts within a single calculation and the files associated with each run are often as large as 64 Gb each. For all the other applications, restart files are usually smaller and less frequently used within a single calculation. The standard repository scheme of LGI (each job submitted to LGI has associated with it a web repository from and to which files can be down- and up-loaded through the two standard interfaces) suffices for those applications.

For the SA-DVR and SPO-DVR applications, a restart repository scheme was implemented in the back-end scripts running either through Load Leveler or Torque/OpenPBS. Now, when the appropriate '<restart_repository_url>' tags are encountered in the job specifics, a restart is assumed and the needed restart files are automatically downloaded from the specified repository. Because it is one of the more challenging applications, the SA-DVR back-end scripts that were developed will be detailed upon here:

check_system_limits_script

This script is used by the daemon to check if a system limit on the resource is met. Only if the script returns zero, work for the application will be requested. For SA-DVR this script just checks if there are any jobs currently queued and waiting to be run on the system for the user under which the daemon is running.

job_check_limits_script

This script is used to check if a job returned by the project web-server is runnable if no system limits were encountered. For SA-DVR this script just returns zero indicating that any SA-DVR job can be run by the resource. A resource administrator can decide to implement extra sanity checks in this script if desired. Details on the information available during the activation if this script are given in the LGI documentation included in the middle-ware distribution.

job_prologue_script

This script is run as a prologue script for the job. For SA-DVR it just returns zero indicating that the job can be run.

job_run_script

This is the main job script to be used for an application. It will be forked to the back-ground by the daemon in a special job run directory. The job run directory contains files giving the script access to the information from the database. Files like LGI_job_id, LGI_job_specifics, LGI_input, LGI_output are used by the script to handle the correct execution of the job. A detailed explanation of what files are present and what they contain is given in the documentation of LGI included into the distribution. For SA-DVR (see appendix E) this script handles the downloading of restart data from the job repositories if needed, it will create a script to be submitted to Torque/OpenPBS or Load Leveler, handles the upload of output files to the job repository and eventually also submits the created script to the local batch manager. Because all data and scripts regarding the actual job are cached on disk in the job run directory, the daemon can safely be restarted and the job will continue to run unaffected.

job_epilogue_script

This script is run after the daemon has determined that the job was finished. For SA-DVR it just returns zero indicating no problems, that the job status can be updated on the project web-server, and that the job directory can be removed by the daemon.

job_check_running_script

This script is used by the daemon to check if the job started through the *job_run_script* is still running. If this script returns zero, it signals that the job is still running. For SA-DVR this script actually invokes the Torque/OpenPBS 'qstat' or the Load Leveler 'llq' command to see if the job is still running or queued on the resource.

job_check_finished_script

This script is invoked by the daemon to check if a non-running job is finished. If this script returns zero, it indicates that the job was finished and the daemon will invoke the *job_epilogue_script*. If the script returns non-zero, the daemon knows that the job wasn't running and also not finished and will therefore start the run sequence with the *job_prologue_script*. For SA-DVR this script simply checks if the file 'finished' was created at the end of the *job_run_script* script. If the 'finished' file wasn't found, it will invoke the Torque/OpenPBS 'qstat' or Load Leveler 'llq' command to see if the job is still queued or running on the resource.

job_abort_script

This script is invoked by the daemon if the job state on the project web-server was changed into the 'aborting' state. If the script returns zero it means that the job was successfully aborted and the *job_epilogue_script* script will be invoked. The job will be marked 'aborted' in the project web-server database afterwards. If the script returns non-zero, it has not successfully aborted the job and the *job_check_running_script* and *job_check_finished_script* scripts are used by the daemon to see if the job was finished eventually. For SA-DVR this script will issue a 'qdel' command for Torque/OpenPBS or an 'llcancel' for Load Leveler if needed.

In appendix D the *job_run_script* script is given for the SA-DVR application, the other back-end scripts for the other applications are now also available as examples in the LGI middle-ware distribution.

Setup of command line interfaces

In general, to setup the command line interface to LGI, the LGI middle-ware needs to be downloaded. Then in the 'daemon/src' subdirectory one invokes 'make' and 'make install'. The same procedure was used to setup a resource or a project web-server. The 'daemon/bin' subdirectory will now contain several executables: LGI_daemon, LGI_filetransfer, LGI_qstat, LGI_qsub, LGI_qdel, xml, csv, hash, binhex and hexbin.

The command line interface can now already be used, however one still needs to supply several flags to the commands. It is also possible to create a '~/.LGI' subdirectory in which the files 'privatekey', 'certificate', 'ca_chain', 'defaultproject', 'defaultserver', 'groups' and 'user' contain the needed parameters that otherwise should have been supplied to the individual commands each and every time. If the ~/.LGI subdirectory is present, the LGI_* commands will use the values stored there as defaults for the needed parameters.

Now that the command line interface is set-up, it can be used as is demonstrated in appendix B. It is now also rather trivial to use the LGI_* commands in other job control managers like ADFJobs for instance. They can now also be used in scripts as all the commands support the '-x' switch that will make the command output in XML format that's easily parsed by the xml, csv, hash, hexbin and binhex utilities also supplied. The same utilities have also been used in all of the back-end scripts when configuring an application on a resource (see appendix E).

Conclusions

In this project the LGI middle-ware was used to set-up a application oriented computer GRID on resources owned by NCF and the theoretical chemistry group of Leiden University. The resources owned by NCF are not within the administrative domain of the theoretical chemistry group. The applications Gaussian, VASP, ADF, SA-DVR, SPO-DVR and Classical have been deployed on several resources and back-end scripts were created for them for several batch-managers (LoadLeveler and OpenPBS/Torque). These back-end scripts are now made part of the LGI middle-ware distribution and serve, together with the already includes 'hello_world' application and the 'SETUP.txt' documentation, as working examples of how to use the middle-ware.

Having successfully deployed such a wide range of application, relevant for a theoretical chemistry group, on such a diverse range of resources clearly shows the easy of use and applicability of the LGI middle-ware for groups similar to the one in Leiden. The LGI middle-ware will also be used in several other project in Leiden; the Cyttron project, and, the TI-Pharma project. Other future projects in which

LGI may take a center-role are: the development of user friendly (graphical) (web-based) interfaces to the SA-DVR and SPO-DVR applications, the adaptation of the Leiden Classical Builder java 3D user interface to directly use LGI and the inclusion of Condor based pools of workstations as resources. Perhaps other groups would like to also set-up their own LGI based grid, or perhaps even share resources for certain applications. They could very well decide only to make use of the Leiden set-up project web-server for themselves. Future work on LGI itself could be the inclusion of a SOAP-based application programming interface to allow for work-flow editors like Taverna, to be used directly with LGI.

References

The LGI middle-ware and the documentation can be found at <http://fwnc7003.leidenuniv.nl/LGI>.

More information about Taverna can be found at <http://taverna.sourceforge.net/>.

More information about Cyttron can be found at <http://www.onderzoekinformatie.nl/en/oi/nod/onderzoek/OND1309180/>.

More information of TI-Pharma can be found at <http://tipharma.com/pro1/general/start.asp?i=2&j=5&k=0&p=0&itemid=344>.

Appendix A

This appendix shows screen-shots of the general LGI web-based interface. More detailed screen-shots can be found in the documentation part of the LGI middle-ware distribution and at http://fwnc7003.leidenuniv.nl/LGI/docs/screen_shots.

How to submit a job:

SeaMonkey

File Edit View Go Bookmarks Tools Window Help

https://fwnc7003.wks.gorlaeus.net/LGI/basic_interface/basic_interface_submit_job_form.php?project=LGI&groups=mark@laptop&sid= Search

Home Bookmarks Scientific Linux Distros

Leiden Grid Infrastructure basic interface at 14 Oct 2008 13:10 UTC

| | |
|-------------------------------|---|
| Project: | LGI |
| This project server: | https://fwnc7003.wks.gorlaeus.net/LGI |
| Project master server: | https://fwnc7003.wks.gorlaeus.net/LGI |
| User: | mark@laptop |
| Groups: | mark@laptop |

Specify job details

| | |
|---------------------------|--|
| Application: | hello_world |
| Extra owners: | |
| Extra read access: | |
| Target resources: | any |
| Job specifics: | |
| File(s) to upload: | /home/mark/Documents/temp/hello_world <input type="button" value="Browse..."/> |
| | <input type="button" value="Browse..."/> |

Input:

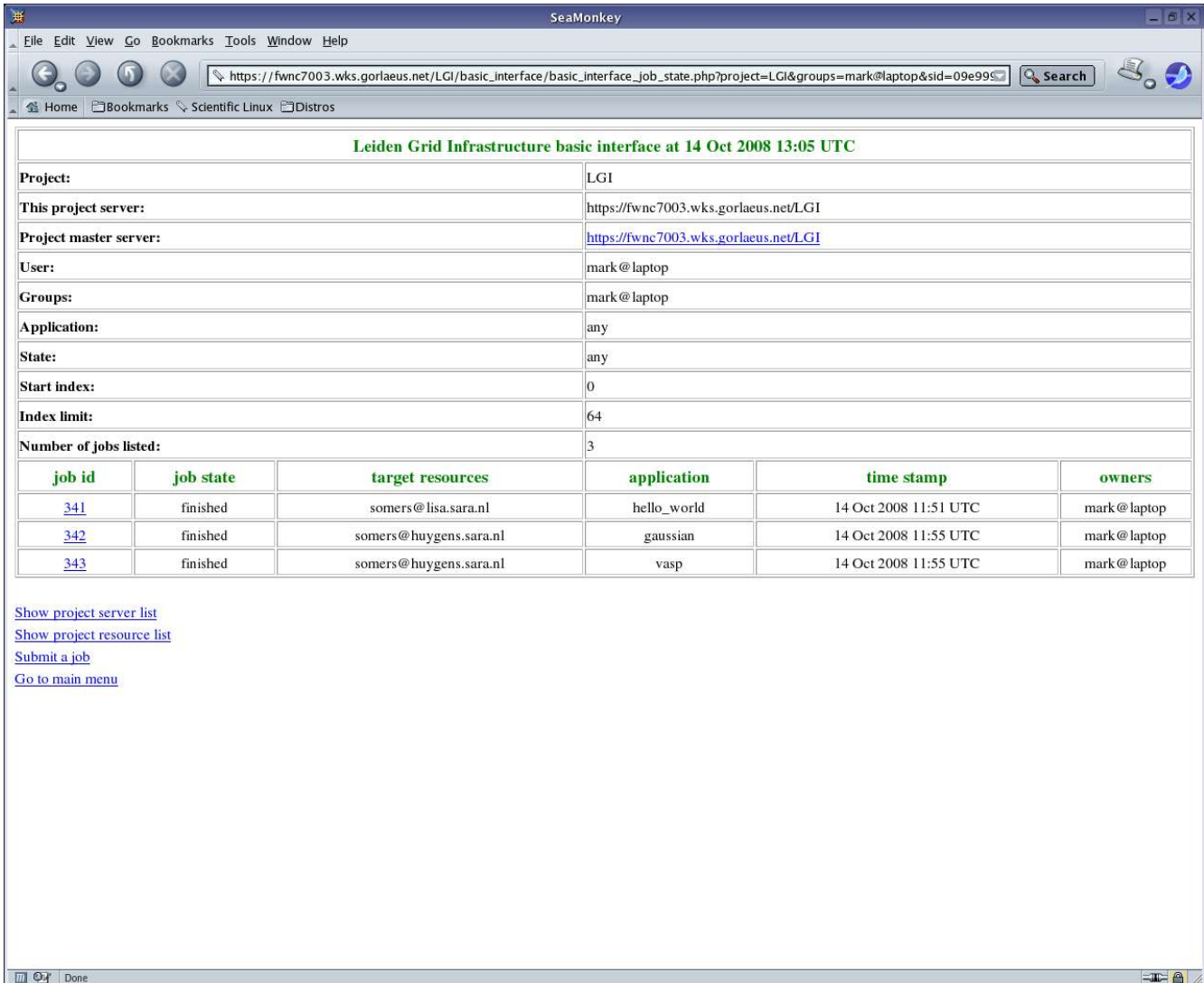
test it..

[Show project server list](#)

[Show project resource list](#)

Done

How to list your jobs:



Leiden Grid Infrastructure basic interface at 14 Oct 2008 13:05 UTC

| | |
|------------------------|---|
| Project: | LGI |
| This project server: | https://fwnc7003.wks.gorlaeus.net/LGI |
| Project master server: | https://fwnc7003.wks.gorlaeus.net/LGI |
| User: | mark @ laptop |
| Groups: | mark @ laptop |
| Application: | any |
| State: | any |
| Start index: | 0 |
| Index limit: | 64 |
| Number of jobs listed: | 3 |

| job id | job state | target resources | application | time stamp | owners |
|---------------------|-----------|------------------------|-------------|-----------------------|---------------|
| 341 | finished | somers@lisa.sara.nl | hello_world | 14 Oct 2008 11:51 UTC | mark @ laptop |
| 342 | finished | somers@huygens.sara.nl | gaussian | 14 Oct 2008 11:55 UTC | mark @ laptop |
| 343 | finished | somers@huygens.sara.nl | vasp | 14 Oct 2008 11:55 UTC | mark @ laptop |

[Show project server list](#)
[Show project resource list](#)
[Submit a job](#)
[Go to main menu](#)

How to get details on a job:

SeaMonkey

File Edit View Go Bookmarks Tools Window Help

https://fwnc7003.wks.gorlaeus.net/LGI/basic_interface/basic_interface_job_state.php?job_id=341&groups=mark@laptop&project=LGI Search

Home Bookmarks Scientific Linux Distros

Leiden Grid Infrastructure basic interface at 14 Oct 2008 13:07 UTC

| | |
|-------------------------------|---|
| Project: | LGI |
| This project server: | https://fwnc7003.wks.gorlaeus.net/LGI |
| Project master server: | https://fwnc7003.wks.gorlaeus.net/LGI |
| User: | mark@laptop |
| Groups: | mark@laptop |

Job details

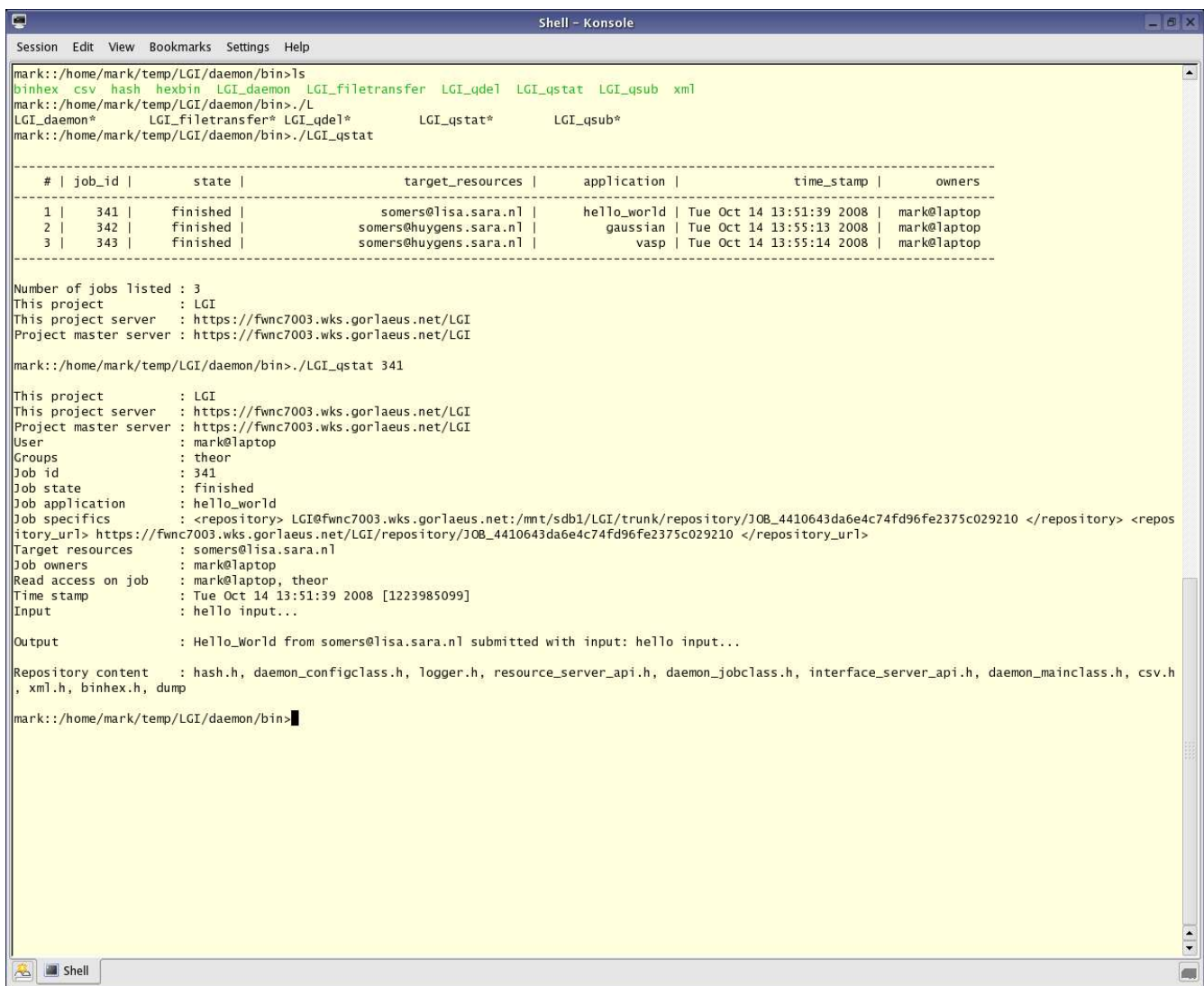
| | |
|--------------------------|--|
| Job ID: | 341 |
| Application: | hello_world |
| State: | finished |
| State time stamp: | 14 Oct 2008 11:51 UTC |
| Owners: | mark@laptop |
| Read access: | mark@laptop, theor |
| Target resources: | somers@lisa.sara.nl |
| Job specifics: | <repository> LGI@fwnc7003.wks.gorlaeus.net/mnt/sdb1/LGI/trunk/repository/JOB_4410643da6e4c74fd96fe2375c029210 </repository> <repository_url> https://fwnc7003.wks.gorlaeus.net/LGI/repository/JOB_4410643da6e4c74fd96fe2375c029210 </repository_url> |
| Repository: | https://fwnc7003.wks.gorlaeus.net/LGI/repository/JOB_4410643da6e4c74fd96fe2375c029210 |
| Input: | hello input... |
| Output: | Hello_World from somers@lisa.sara.nl submitted with input: hello input... |

[Abort or Delete this job](#)
[Show job list](#)
[Show project server list](#)
[Show project resource list](#)
[Submit a job](#)
[Go to main menu](#)

Appendix B

This appendix shows screen-shots of the general LGI command line interface. More detailed screen-shots can be found in the documentation part of the LGI middle-ware distribution and at http://fwnc7003.leidenuniv.nl/LGI/docs/screen_shots.

How to list your jobs and get details on a job:



```

mark:/home/mark/temp/LGI/daemon/bin>ls
binhex csv hash hexbin LGI_daemon LGI_filetransfer LGI_qdel LGI_qstat LGI_qsub xml
mark:/home/mark/temp/LGI/daemon/bin>./L
LGI_daemon*      LGI_filetransfer* LGI_qdel*      LGI_qstat*      LGI_qsub*
mark:/home/mark/temp/LGI/daemon/bin>./LGI_qstat

-----
# | job_id |      state |      target_resources |      application |      time_stamp |      owners
-----
1 |   341 | finished | somers@lisa.sara.nl | hello_world | Tue Oct 14 13:51:39 2008 | mark@laptop
2 |   342 | finished | somers@huygens.sara.nl | gaussian | Tue Oct 14 13:55:13 2008 | mark@laptop
3 |   343 | finished | somers@huygens.sara.nl | vasp | Tue Oct 14 13:55:14 2008 | mark@laptop
-----

Number of jobs listed : 3
This project          : LGI
This project server   : https://fwnc7003.wks.gorlaeus.net/LGI
Project master server : https://fwnc7003.wks.gorlaeus.net/LGI

mark:/home/mark/temp/LGI/daemon/bin>./LGI_qstat 341

This project          : LGI
This project server   : https://fwnc7003.wks.gorlaeus.net/LGI
Project master server : https://fwnc7003.wks.gorlaeus.net/LGI
User                  : mark@laptop
Groups                : theor
Job id                : 341
Job state              : finished
Job application        : hello_world
Job specifics          : <repository> LGI@fwnc7003.wks.gorlaeus.net:/mnt/sdb1/LGI/trunk/repository/JOB_4410643da6e4c74fd96fe2375c029210 </repository> <repos
itory_url> https://fwnc7003.wks.gorlaeus.net/LGI/repository/JOB_4410643da6e4c74fd96fe2375c029210 </repository_url>
Target resources      : somers@lisa.sara.nl
Job owners            : mark@laptop
Read access on job    : mark@laptop, theor
Time stamp            : Tue Oct 14 13:51:39 2008 [1223985099]
Input                 : hello input...

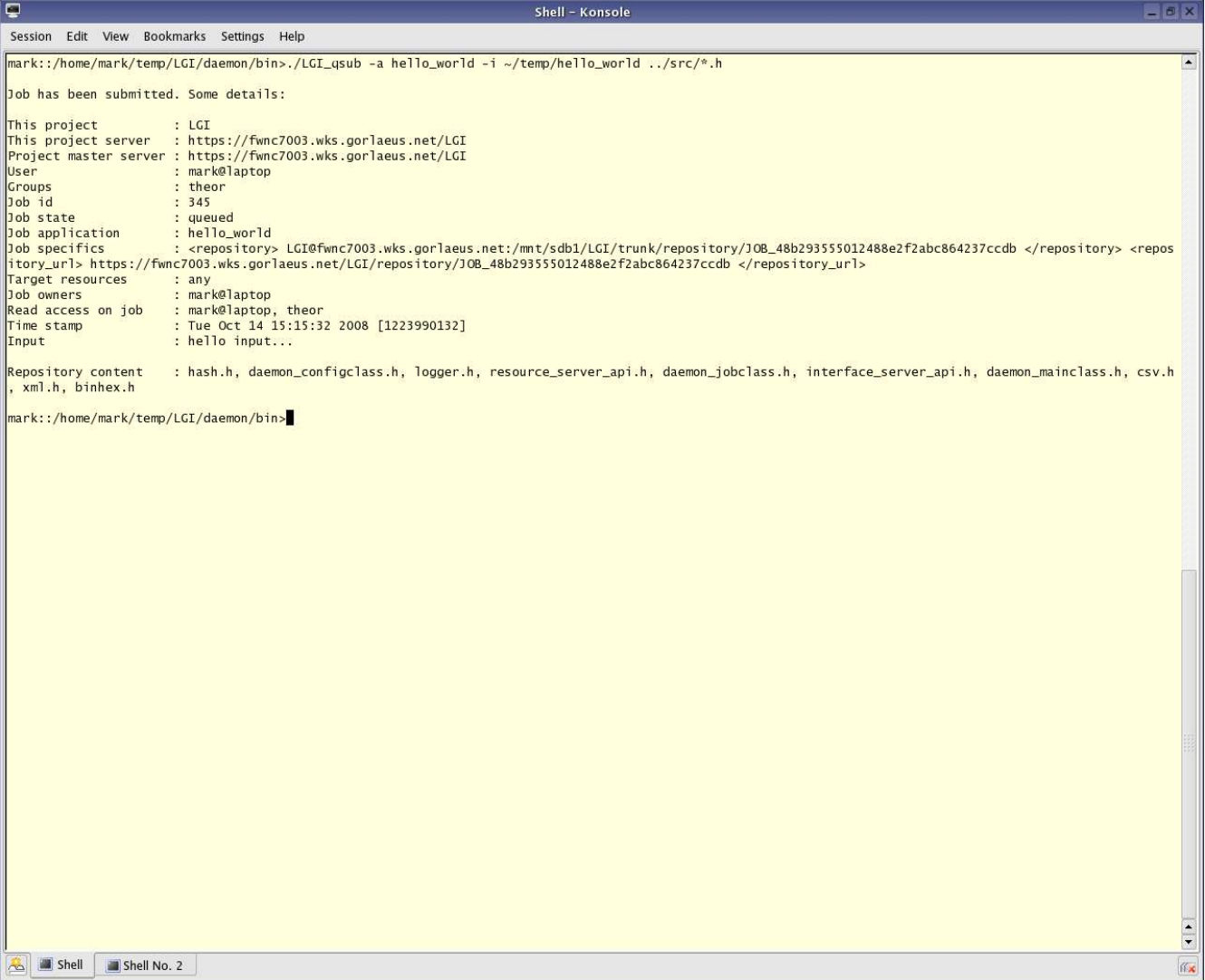
Output                : Hello_World from somers@lisa.sara.nl submitted with input: hello input...

Repository content    : hash.h, daemon_configclass.h, logger.h, resource_server_api.h, daemon_jobclass.h, interface_server_api.h, daemon_mainclass.h, csv.h
, xml.h, binhex.h, dump

mark:/home/mark/temp/LGI/daemon/bin>

```

How to submit a job:



```
mark@:/home/mark/temp/LGI/daemon/bin> ./LGI_qsub -a hello_world -i ~/temp/hello_world ./src/*.h

Job has been submitted. Some details:

This project      : LGI
This project server : https://fwnc7003.wks.gorlaeus.net/LGI
Project master server : https://fwnc7003.wks.gorlaeus.net/LGI
User              : mark@laptop
Groups            : theor
Job id            : 345
Job state         : queued
Job application   : hello_world
Job specifics     : <repository> LGI@fwnc7003.wks.gorlaeus.net:/mnt/sdb1/LGI/trunk/repository/JOE_48b293555012488e2f2abc864237ccdb </repository> <repository_url> https://fwnc7003.wks.gorlaeus.net/LGI/repository/JOE_48b293555012488e2f2abc864237ccdb </repository_url>
Target resources  : any
Job owners        : mark@laptop
Read access on job : mark@laptop, theor
Time stamp        : Tue Oct 14 15:15:32 2008 [1223990132]
Input             : hello input...

Repository content : hash.h, daemon_configclass.h, logger.h, resource_server_api.h, daemon_jobclass.h, interface_server_api.h, daemon_mainclass.h, csv.h, xml.h, binhex.h

mark@:/home/mark/temp/LGI/daemon/bin>
```


How to delete a job:

```

Shell - Konsole
Session Edit View Bookmarks Settings Help

This project      : LGI
This project server : https://fwnc7003.wks.gorlaeus.net/LGI
Project master server : https://fwnc7003.wks.gorlaeus.net/LGI
User              : mark@laptop
Groups            : theor
Job id            : 345
Job state         : queued
Job application    : hello_world
Job specifics      : <repository> LGI@fwnc7003.wks.gorlaeus.net:/mnt/sdb1/LGI/trunk/repository/JO8_48b293555012488e2f2abc864237ccdb </repository> <repository_url> https://fwnc7003.wks.gorlaeus.net/LGI/repository/JO8_48b293555012488e2f2abc864237ccdb </repository_url>
Target resources   : any
Job owners        : mark@laptop
Read access on job : mark@laptop, theor
Time stamp        : Tue Oct 14 15:15:32 2008 [1223990132]
Input             : hello input...

Repository content : hash.h, daemon_configclass.h, logger.h, resource_server_api.h, daemon_jobclass.h, interface_server_api.h, daemon_mainclass.h, csv.h, xml.h, binhex.h

mark::/home/mark/temp/LGI/daemon/bin>./LGI_qstat

-----
# | job_id | state | target_resources | application | time_stamp | owners
-----
1 | 341 | finished | somers@lisa.sara.nl | hello_world | Tue Oct 14 13:51:39 2008 | mark@laptop
2 | 342 | finished | somers@huygens.sara.nl | gaussian | Tue Oct 14 13:55:13 2008 | mark@laptop
3 | 343 | finished | somers@huygens.sara.nl | vasp | Tue Oct 14 13:55:14 2008 | mark@laptop
4 | 344 | running | mark@fwnc7003.wks.gorlaeus.net | hello_world | Tue Oct 14 15:15:25 2008 | mark@laptop
5 | 345 | queued | any | hello_world | Tue Oct 14 15:15:32 2008 | mark@laptop
-----

Number of jobs listed : 5
This project      : LGI
This project server : https://fwnc7003.wks.gorlaeus.net/LGI
Project master server : https://fwnc7003.wks.gorlaeus.net/LGI

mark::/home/mark/temp/LGI/daemon/bin>./LGI_qdel 341

Job 341 deleted from project LGI on server https://fwnc7003.wks.gorlaeus.net/LGI

mark::/home/mark/temp/LGI/daemon/bin>./LGI_qstat

-----
# | job_id | state | target_resources | application | time_stamp | owners
-----
1 | 342 | finished | somers@huygens.sara.nl | gaussian | Tue Oct 14 13:55:13 2008 | mark@laptop
2 | 343 | finished | somers@huygens.sara.nl | vasp | Tue Oct 14 13:55:14 2008 | mark@laptop
3 | 344 | running | mark@fwnc7003.wks.gorlaeus.net | hello_world | Tue Oct 14 15:15:25 2008 | mark@laptop
4 | 345 | queued | any | hello_world | Tue Oct 14 15:15:32 2008 | mark@laptop
-----

Number of jobs listed : 4
This project      : LGI
This project server : https://fwnc7003.wks.gorlaeus.net/LGI
Project master server : https://fwnc7003.wks.gorlaeus.net/LGI

mark::/home/mark/temp/LGI/daemon/bin>

```

Appendix C

This appendix shows the SETUP.txt file, which is part of the LGI distribution:

HOW TO SETUP LGI -----

In this document a howto is presented on the setup of LGI. The document is divided into four parts. Part I will show you how to setup a resource within an existing LGI project, part II will show you how to setup a new LGI project with a project server. Part III will show you how to add an extra project server into an existing LGI project. The last part part IV will detail on the maintenance of your LGI project servers.

Part I : Howto setup a resource -----

Setting up a resource within an LGI project is rather easy. Just follow the following steps:

- Untar or checkout the source tree in '~/LGI'.
- Get in contact with the project administrator to get the X.509 certificate and a private key for your resource.
- Copy the key and certificate files into the '~/LGI/certificates' directory.
- Make sure that libcurl is installed. Visit <http://curl.haxx.se/libcurl/> if you do not have this yet.
- Go to the '~/LGI/daemon/src' directory and invoke make. If you have a different compiler installed than GNU's g++, or you have libcurl installed in a different location, adjust the first few lines of the file Makefile to your needs.
- If you want, you can also invoke make with the 'make install' command. If you do that the directory '~/LGI/daemon/bin' will be created and you can include that in your PATH for convenience.
- Edit the '~/LGI/daemon/LGI.cfg' example configuration file. It has been setup to run the hello_world application by just simple forking scripts. Other example scripts on how to use the 'hello_world' example using Torque / PBS or LoadLeveler are included to in the corresponding '~/LGI/daemon/hello_world_XX_scripts' subdirectories. Please make sure you refer to the right LGI project server, use the correct project name and use the correct private key and certificate files from the '~/LGI/certificates' directory. Also be sure to use absolute paths in the configuration file so it is possible to invoke the daemon from any other directory. Finally set the run directory correctly if you do not want it to be the default.
- Start the daemon by invoking the daemon like:

```
~/LGI/daemon/src/LGI_daemon -l ~/LGI/daemon/LGI.log ~/LGI/daemon/LGI.cfg
```

You can now inspect the log file '~/LGI/daemon/LGI.log' and see if everything was setup correctly.

- You can gracefully stop the dameon by sending it a 'kill' signal:

```
killall LGI_daemon
```

While the daemon is running, you can edit the configuration file without any problems. You can also edit the scripts in the mean time, only newly spawned jobs will be affected by these changes.

Keep in mind for a project administrator, you should also include the new resource into the projects database. See part IV for an example.

Part II : Howto setup a project server

Setting up your own project is not too hard either. You can decide to be a fully independent project by setting up your own X.509 Certificate Authority, or you can decide to be a sub project and thus be an X.509 sub-CA of the LGI-CA. In this part, it is shown how to become an independant project and be your own X.509 CA.

- Make sure libcurl, Perl, PHP, MySQL, Apache and OpenSSL are installed on your system.
- Make sure MySQL is configured correctly and mysqld is running.
- Make sure PHP is configured correctly.
- Switch to the user apache and make sure you install the source tree into the home of user apache where it will be served:

```
sudo su -l -s /bin/bash apache
```

```
tar -zxf LGI.tar.gz -C /var/www/html
```

- Use OpenSSL to create a CA private key and certificate with which you can create and sign your project server, resource and client certificates:

Start by creating a private key for your CA:

```
openssl genrsa -out /var/www/html/LGI/certificates/exampleCA.key 4096
```

Then create an X.509 selfsigned certificate for it:

```
openssl req -new -x509 -days 365 -set_serial 0 -extensions v3_ca
-key /var/www/html/LGI/certificates/exampleCA.key
-out /var/www/html/LGI/certificates/exampleCA.crt
```

and create a CA serial number file:

```
echo "0100" > /var/www/html/LGI/certificates/exampleCA.srl
```

Now create a private key for your server:

```
openssl genrsa -out /var/www/html/LGI/certificates/exampleserver.key 4096
```

Then create a sign request for it:

```
openssl req -new -key /var/www/html/LGI/certificates/exampleserver.key
-out /var/www/html/LGI/certificates/exampleserver.crs
```

and in filling in the commonname of the CA certificate, be sure to use the correct LGI format 'apache@exampleserver.somewhere.org; exampleprojectname'.

Now create the CA signed server certificate from this request:

```
openssl x509 -req -in /var/www/html/LGI/certificates/exampleserver.crs
-days 365 -CA /var/www/html/LGI/certificates/exampleCA.crt
-CAkey /var/www/html/LGI/certificates/exampleCA.key
-CAserial /var/www/html/LGI/certificates/exampleCA.srl
-out /var/www/html/LGI/certificates/exampleserver.crt
```

You should now safely store the CA private key file somewhere not on this server computer that is connected to the internet. Or, at least, if you do not do that, encrypt the file using a password protection:

```
openssl aes-256-cbc -e -in /var/www/html/LGI/certificates/exampleCA.key
-out /var/www/html/LGI/certificates/exampleCA.key.aes; rm
/var/www/html/LGI/certificates/exampleCA.key
```

Now copy the public certificates to a place that Apache will serve:

```
cp /var/www/html/LGI/certificates/exampleCA.crt /var/www/html/LGI
cp /var/www/html/LGI/certificates/exampleserver.crt /var/www/html/LGI
```

- Make sure Apache is setup correctly to use client and server certificates for https and check that the following options have been set correctly into the https virtual host configuration:

```
SSLCertificateFile /var/www/html/LGI/certificates/exampleserver.crt
SSLCertificateKeyFile /var/www/html/LGI/certificates/exampleserver.key
SSLCertificateChainFile /var/www/html/LGI/certificates/exampleCA.crt
SSLCACertificateFile /var/www/html/LGI/certificates/exampleCA.crt
SSLVerifyClient require
SSLVerifyDepth 5
SSLOptions +ExportCertData

<Files ~ "\.(cgi|shtml|phtml|php3?)$" >
    SSLOptions +StdEnvVars
</Files>
```

Also make sure the AccessFileName has been set correctly in the Apache main configuration so that the LGI document tree can be protected by .htaccess files and make sure the important directories are carefully protected:

```
DocumentRoot "/var/www/html"
AccessFileName .htaccess

<Directory "/var/www/html/LGI">
    AllowOverride All
</Directory>

<Directory "/var/www/html/LGI/repository/JOB_*">
    AllowOverride None
    Options -ExecCGI
    php_flag engine off
    SSLRequireSSL
    SSLRequire ( %{SSL_CLIENT_VERIFY} == "SUCCESS" )
    Script PUT /LGI/repository/put.cgi
    Script DELETE /LGI/repository/delete.cgi
    <Limit GET PUT DELETE>
        Allow from all
    </Limit>
</Directory>

<Directory "/var/www/html/LGI/repository">
    Deny from all
    <Files "repository_content.php">
        SSLRequireSSL
        SSLRequire ( %{SSL_CLIENT_VERIFY} == "SUCCESS" )
        Allow from all
    </Files>
    <Files "put.cgi">
        SSLRequireSSL
        SSLRequire ( %{SSL_CLIENT_VERIFY} == "SUCCESS" )
        Allow from all
        Options +ExecCGI
        AddHandler cgi-script .cgi
    </Files>
    <Files "delete.cgi">
        SSLRequireSSL
```

```

        SSLRequire ( %{SSL_CLIENT_VERIFY} == "SUCCESS" )
        Allow from all
        Options +ExecCGI
        AddHandler cgi-script .cgi
    </Files>
</Directory>

<Directory "/var/www/html/LGI/inc">
    Deny from all
</Directory>

<Directory "/var/www/html/LGI/tools">
    Deny from all
</Directory>

<Directory "/var/www/html/LGI/daemon">
    Deny from all
</Directory>

<Directory "/var/www/html/LGI/certificates">
    Deny from all
</Directory>

<Directory "/var/www/html/LGI/basic_interface">
    php_value upload_max_filesize 16M
    php_value post_max_size 16M
</Directory>

<Directory "/var/www/html/LGI/interfaces">
    php_value upload_max_filesize 16M
    php_value post_max_size 16M
</Directory>

<Directory "/var/www/html/LGI/resources">
    php_value upload_max_filesize 16M
    php_value post_max_size 16M
</Directory>

Make sure you have an .htaccess file in the '/var/www/html/LGI/certificates',
'/var/www/html/LGI/inc', '/var/www/html/LGI/tools' and '/var/www/html/LGI/daemon'
directories with the content:

Deny from all

Make sure you have an .htaccess file in the '/var/www/html/LGI/repository'
directory with the content:

Deny from all
<Files "repository_content.php">
    SSLRequireSSL
    SSLRequire ( %{SSL_CLIENT_VERIFY} == "SUCCESS" )
    Allow from all
</Files>
<Files "put.cgi">
    SSLRequireSSL
    SSLRequire ( %{SSL_CLIENT_VERIFY} == "SUCCESS" )
    Allow from all
    Options +ExecCGI
    AddHandler cgi-script .cgi
</Files>
<Files "delete.cgi">
    SSLRequireSSL
    SSLRequire ( %{SSL_CLIENT_VERIFY} == "SUCCESS" )
    Allow from all
    Options +ExecCGI
    AddHandler cgi-script .cgi
</Files>

```

Make sure you have an .htaccess file in the directories
 '/var/www/html/LGI/basic_interface', '/var/www/html/LGI/interfaces' and

`/var/www/html/LGI/resources` with the following content:

```
php_value upload_max_filesize 16M
php_value post_max_size 16M
```

Make sure you have an `index.html` file in the subdirectories `/var/www/html/LGI/repository`, `/var/www/html/LGI/inc`, `/var/www/html/LGI/interfaces`, `/var/www/html/LGI/servers`, `/var/www/html/LGI/resources`, `/var/www/html/LGI/daemon`, `/var/www/html/LGI/tools` and `/var/www/html/LGI/certificates` with the content:

```
<html>
  ! No browsing allowed !
</html>
```

Finally check your Apache configuration with the `'apachectl configtest'` command.

!!!! WORD OF CAUTION !!!!

The above configuration of Apache seems superfluous but ensures a secure environment. Please be very carefull when diverting from the above suggested configuration. Web application security is a complicated matter and a lot of care has been taken to ensure a secure default configuration that will work out-of-the box.

!!!! WORD OF CAUTION !!!!

- Now create a MySQL user and database for your project with the database name equal to your project name `"exampleprojectname"`:

```
mysqladmin -u root password "yourmysqlrootpassword"
```

```
mysqladmin -u root -p create "exampleprojectname"
```

```
echo 'GRANT ALL PRIVILEGES ON exampleprojectname.* to
      "examplemysqluser"@"localhost" IDENTIFIED BY "examplemysqluserpasswd"
      | mysql -u root -p mysql
```

```
mysql -u examplemysqluser -p exampleprojectname < /var/www/html/LGI/LGI.db
```

- Insert your project server entry into the database:

```
export CERTIFICATE=`cat /var/www/html/LGI/certificates/exampleCA.crt`
```

```
echo -e "INSERT INTO active_resources SET resource_name=
'apache@exampleserver.somewhere.org',project_server=1,
url='https://exampleserver.somewhere.org/LGI',
client_certificate='$CERTIFICATE\n'" | mysql -u
examplemysqluser -p exampleprojectname
```

- Make sure you edit the project web server configuration file `'/var/www/html/LGI/inc/Config.inc'`:

```
$Config[ "SERVER_URL" ] = "https://exampleserver.somewhere.org/LGI";
$Config[ "SERVER_NAME" ] = "apache@exampleserver.somewhere.org";
$Config[ "SERVER_SSL_CERTIFICATE_FILE" ] = "../certificates/exampleserver.crt";
$Config[ "SERVER_SSL_KEY" ] = "../certificates/exampleserver.key";
$Config[ "SERVER_SSL_CA_CERTIFICATE_URL" ] =
  "https://exampleserver.somewhere.org/LGI/exampleCA.crt";
$Config[ "SERVER_SSL_CA_CERTIFICATE_FILE" ] = "../certificates/exampleCA.crt";
$Config[ "MYSQL_URL" ] = "localhost";
$Config[ "MYSQL_USER" ] = "examplemysqluser";
$Config[ "MYSQL_PASSWD" ] = "examplemysqluserpasswd";
$Config[ "MYSQL_DEFAULT_DATABASE" ] = "exampleprojectname";
$Config[ "REPOSITORY_DIRECTORY" ] = "/var/www/html/LGI/repository";
$Config[ "REPOSITORY_SERVER_NAME" ] = $Config[ "SERVER_NAME" ];
$Config[ "REPOSITORY_URL" ] = $Config[ "SERVER_URL" ]."/repository";
```

- The last thing to do is to edit a crontab entry for the apache user

```
crontab -u apache -e
```

Add the following lines:

```
MAILTO=""
HOME=/var/www
*/10 * * * * /usr/bin/php -f /var/www/html/LGI/tools/cron_job.php
```

You could also choose to use a different user account to setup your LGI project web-server. Just be sure that the user apache is part of the group the new user is part of too so that apache can serve the content.

If you happen to run several projects on the same server, please adapt the above cron_job.php file to correctly maintain all of them. This is done by simply adding the project name to the array of projects as is demonstrated in cron_job.php.

Part III : Howto setup a second slave project server

After having successfully setup your first LGI master project server yourself, adding slave servers is fairly straightforward. Follow the same procedure on the slave server as was described in part II for the master server, but now there is no need to setup a CA. You only have to generate a server key and a certificate request for the slave server, and obviously sign that request with your CA certificate.

However, at the point where you insert the server entry into the fresh database in part II, use that command now to insert the MASTER server data into the slave database:

(on the slaveserver)

```
wget https://exampleserver.somewhere.org/exampleCA.crt
-O /var/www/html/LGI/certificates/exampleCA.crt

wget https://exampleserver.somewhere.org/exampleserver.crt
-O /var/www/html/LGI/certificates/exampleserver.crt

export CERTIFICATE=`cat /var/www/html/LGI/certificates/exampleserver.crt`

echo -e "INSERT INTO active_resources SET resource_name=
'apache@exampleserver.somewhere.org',project_server=1,
url='https://exampleserver.somewhere.org/LGI',
client_certificate='$CERTIFICATE\n'" | mysql -u
exampleslavemysqluser -p exampleprojectname
```

and insert the slave server details into the master server database:

(on the masterserver)

```
wget https://exampleslaveserver.somewhere.org/exampleslaveserver.crt
-O /var/www/html/LGI/certificates/exampleslaveserver.crt

export SLAVECERTIFICATE=`cat /var/www/html/LGI/certificates/exampleslaveserver.crt`

export SLAVEINSERTQUERY="INSERT INTO active_resources SET resource_name=
'apache@exampleslaveserver.somewhere.org',project_server=2,
url='https://exampleslaveserver.somewhere.org/LGI',
client_certificate='$SLAVECERTIFICATE\n'"

export ESCAPEDSLAVEINSERTQUERY=`echo -e "$SLAVEINSERTQUERY" | sed "s/'/\\\\\\\\\\'/g"`

export UPDATEQUERY="INSERT INTO updates SET servers='any',
update_query='$ESCAPEDSLAVEINSERTQUERY'"
```

```
echo "$SLAVEINSERTQUERY" | mysql -u examplemysqluser -p exampleprojectname

echo "$UPDATEQUERY" | mysql -u examplemysqluser -p exampleprojectname
```

At this point the two servers are linked together and the crontab entries of both will ensure that any updates pending in the 'updates' table of the master server database are transferred to the slave server.

Users can now use both the slave and the master server independantly to submit jobs. The resource daemon will automatically request work from all servers of the project.

Part IV: Updating and maintaining project servers:

By default on each project server, the basic interface is active and any user is allowed to have only two jobs in the database. If you want certain users or groups of users to be able to submit more, you can add that user or group with a new limit for a certain application. One should do that like:

```
php -f tools/cron_job.php

export QUERY="INSERT INTO users_allowed SET user_name='user',
    application='hello_world', job_limit=10"

echo "$QUERY" | mysql -u examplemysqluser -p exampleprojectname

export ESCAPEDQUERY=`echo "$QUERY" | sed "s/'/\\\\\\\\'/g"`

export UPDATEQUERY="INSERT INTO updates SET
    servers='apache@exampleserver.somewhere.org',
    update_query='$ESCAPEDQUERY'"

echo "$UPDATEQUERY" | mysql -u examplemysqluser -p exampleprojectname
```

By also inserting an update query into the server database for this, slave servers will automatically synchronize and also apply them if needed. It also allows the easy addition of more slave servers as was shown in part III. To make this easier, one can use the UpdatedDB script located in the tools subdirectory:

```
tools/UpdatedDB "INSERT INTO users_allowed SET user_name='user',
    application='hello_world', job_limit=10"
    "apache@exampleserver.somewhere.org" -u examplemysqluser
    -p exampleprojectname
```

To insert a new resource into the database use the following commands:

```
export CERTIFICATE=`cat newresource.crt`

tools/UpdatedDB "INSERT INTO active_resources SET resource_name=
    'user@newresource.somewhere.org',project_server=0, url=
    'user@newresource.somewhere.org', client_certificate=
    '$CERTIFICATE\\n'" "any" -u examplemysqluser -p exampleprojectname
```

Beware, if you have password protected the account for user examplemysqluser in MySQL, you need to type in your password twice with this UpdatedDB script.

Please read in the docs subdirectory the full details on the structure of the database, what the fields mean and what values you can assign to them with MySQL queries like demonstrated.

Appendix D

This appendix shows the example configuration of an LGI resource daemon:

```
<LGI>
  <ca_certificate_file> /home/mark/temp/LGI/certificates/LGI+CA.crt </ca_certificate_file>
  <resource>
    <resource_certificate_file> /home/mark/temp/LGI/certificates/mark@laptop.crt
  </resource_certificate_file>
    <resource_key_file> /home/mark/temp/LGI/certificates/mark@laptop.key
  </resource_key_file>
    <run_directory> /home/mark/temp/LGI/daemon/runhere </run_directory>
    <owner_allow> <any> 10 </any> </owner_allow>
    <owner_deny> nobody </owner_deny>
    <job_limit> 20 </job_limit>

    <number_of_projects> 1 </number_of_projects>

    <project number='1'>
      <project_name> LGI </project_name>
      <project_master_server> https://fwnc7003.leidenuniv.nl/LGI
    </project_master_server>

      <owner_allow> <any> 5 </any> </owner_allow>
      <owner_deny> nobody </owner_deny>
      <job_limit> 10 </job_limit>

      <number_of_applications> 1 </number_of_applications>

      <application number='1'>
        <application_name> hello_world </application_name>

        <owner_allow> <any> 2 </any> </owner_allow>
        <owner_deny> nobody </owner_deny>
        <job_limit> 4 </job_limit>
        <max_output_size> 4096 </max_output_size>

        <check_system_limits_script> /
home/mark/temp/LGI/daemon/hello_world_scripts/check_system_limits_script </check_system_limits_script>
        <job_check_limits_script> /
home/mark/temp/LGI/daemon/hello_world_scripts/job_check_limits_script </job_check_limits_script>
        <job_check_running_script> /
home/mark/temp/LGI/daemon/hello_world_scripts/job_check_running_script </job_check_running_script>
        <job_check_finished_script> /
home/mark/temp/LGI/daemon/hello_world_scripts/job_check_finished_script </job_check_finished_script>
        <job_prologue_script> /
home/mark/temp/LGI/daemon/hello_world_scripts/job_prologue_script </job_prologue_script>
        <job_run_script> /
home/mark/temp/LGI/daemon/hello_world_scripts/job_run_script </job_run_script>
        <job_epilogue_script> /
home/mark/temp/LGI/daemon/hello_world_scripts/job_epilogue_script </job_epilogue_script>
        <job_abort_script> /
home/mark/temp/LGI/daemon/hello_world_scripts/job_abort_script </job_abort_script>
      </application>

    </project>

  </resource>
</LGI>
```

Appendix E

This appendix shows the `job_run_script` for the SA-DVR application for Torque/OpenPBS:

```
#!/bin/csh

setenv LGI_DIR /home/${USER}/LGI
setenv CPUS `${LGI_DIR}/daemon/bin/xml -i LGI_job_specifics cpus | grep -E '^([0-9]{1,2})$`
if ( ${CPUS} == "" ) then
    setenv CPUS 1
endif
setenv WALLTIME `${LGI_DIR}/daemon/bin/xml -i LGI_job_specifics walltime | grep -E '^([0-9]{1,2}):([0-9]{1,2}):([0-9]{1,2})$`
if ( ${WALLTIME} == "" ) then
    setenv WALLTIME "12:00:00"
endif

cat > ./pbs_script << END_OF_PBS_SCRIPT
#PBS -S /bin/csh
#PBS -l nodes=01:ppn=${CPUS},walltime=${WALLTIME}
#PBS -N SA-DVR_pbs
#PBS -o LGI_output

setenv LGI_DIR /home/${USER}/LGI

setenv REPOSITORY_URL `${LGI_DIR}/daemon/bin/xml repository_url < ${PBS_O_WORKDIR}/
LGI_job_specifics\`
setenv RESTART_REPOSITORY_URL `${LGI_DIR}/daemon/bin/xml restart_repository_url < ${PBS_O_WORKDIR}/
LGI_job_specifics\`

setenv NODE `hostname -f`
cd /scratch/${USER}/${PBS_JOBID}

setenv CWD `pwd`

if ( ${REPOSITORY_URL} != "" ) then

    setenv REPOSITORY_CONTENT `${LGI_DIR}/daemon/bin/LGI_filetransfer -j ${PBS_O_WORKDIR} -x list ${
REPOSITORY_URL}\`

    setenv NROFFILES `echo ${REPOSITORY_CONTENT} | ${LGI_DIR}/daemon/bin/xml number_of_files\`
    setenv FILE_LIST ""

    foreach nr ( `seq 1 ${NROFFILES}` )
        setenv FILE_ATTR number=\`${nr}\`
        setenv FILE_DATA `echo ${REPOSITORY_CONTENT} | ${LGI_DIR}/daemon/bin/xml file ${FILE_ATTR}\`
        setenv FILE_NAME `echo ${FILE_DATA} | ${LGI_DIR}/daemon/bin/xml file_name\`
        setenv FILE_LIST "${FILE_LIST} ${FILE_NAME}"
    end

    if ( "${FILE_LIST}" != "" ) then
        setenv DATE `date`
        echo "SA-DVR_PBS started retrieving repository ${REPOSITORY_URL} to ${NODE}:${CWD} at ${DATE}."
        ${LGI_DIR}/daemon/bin/LGI_filetransfer -j ${PBS_O_WORKDIR} download ${REPOSITORY_URL} \${
FILE_LIST} > /dev/null
    else
        echo "No files found in repository."
    endif

    if ( ${RESTART_REPOSITORY_URL} != "" ) then
        setenv DATE `date`
        echo "SA-DVR_PBS started retrieving restart data from repository ${RESTART_REPOSITORY_URL} to \${
NODE}:${CWD} at ${DATE}."
```

```

    \${LGI_DIR}/daemon/bin/LGI_filetransfer -j \${PBS_O_WORKDIR} download \${RESTART_REPOSITORY_URL}
POT.dat > /dev/null
    \${LGI_DIR}/daemon/bin/LGI_filetransfer -j \${PBS_O_WORKDIR} download \${RESTART_REPOSITORY_URL}
QOVERLAP.out > /dev/null; mv QOVERLAP.out QOVERLAP.in
    \${LGI_DIR}/daemon/bin/LGI_filetransfer -j \${PBS_O_WORKDIR} download \${RESTART_REPOSITORY_URL}
WF.out > /dev/null; mv WF.out WF.in
    \${LGI_DIR}/daemon/bin/LGI_filetransfer -j \${PBS_O_WORKDIR} download \${RESTART_REPOSITORY_URL}
BK.out > /dev/null; mv BK.out BK.in
    \${LGI_DIR}/daemon/bin/LGI_filetransfer -j \${PBS_O_WORKDIR} download \${RESTART_REPOSITORY_URL}
SMATRIX.out > /dev/null; mv SMATRIX.out SMATRIX.in
endif

if ( -f DVR.inp ) then
    echo "DVR.inp was uploaded into repository... Ignoring other input..."
else
    echo "Creating DVR.inp from input..."
    dos2unix -n \${PBS_O_WORKDIR}/LGI_input DVR.inp > /dev/null
endif

echo "Copying potential data files and executable..."
cp \${LGI_DIR}/daemon/SA-DVR/* .

setenv DATE `date`
echo "SA-DVR_PBS started SA-DVR on node \${NODE} at \${DATE}."
setenv OMP_NUM_THREADS ${CPUS}
setenv OMP_NUM_DYNAMIC FALSE
./SA-DVR.x > DVR.out

setenv FILE_LIST `ls *.out *.dbg DVR.inp POT.dat`
if ( "\${FILE_LIST}" != "" ) then
    setenv DATE `date`
    echo "SA-DVR_PBS started sending back the output from \${NODE}:\${CWD} to repository \${
{REPOSITORY_URL} at \${DATE}."
    \${LGI_DIR}/daemon/bin/LGI_filetransfer -j \${PBS_O_WORKDIR} upload \${REPOSITORY_URL} \${
{FILE_LIST} > /dev/null
else
    echo "No files locally created."
endif

else
    echo "ERROR: No repository was specified for job... Stopping..."
endif

setenv DATE `date`
echo "SA-DVR_PBS finished at \${DATE}."
touch \${PBS_O_WORKDIR}/finished
END_OF_PBS_SCRIPT

qsub pbs_script > ./running

```