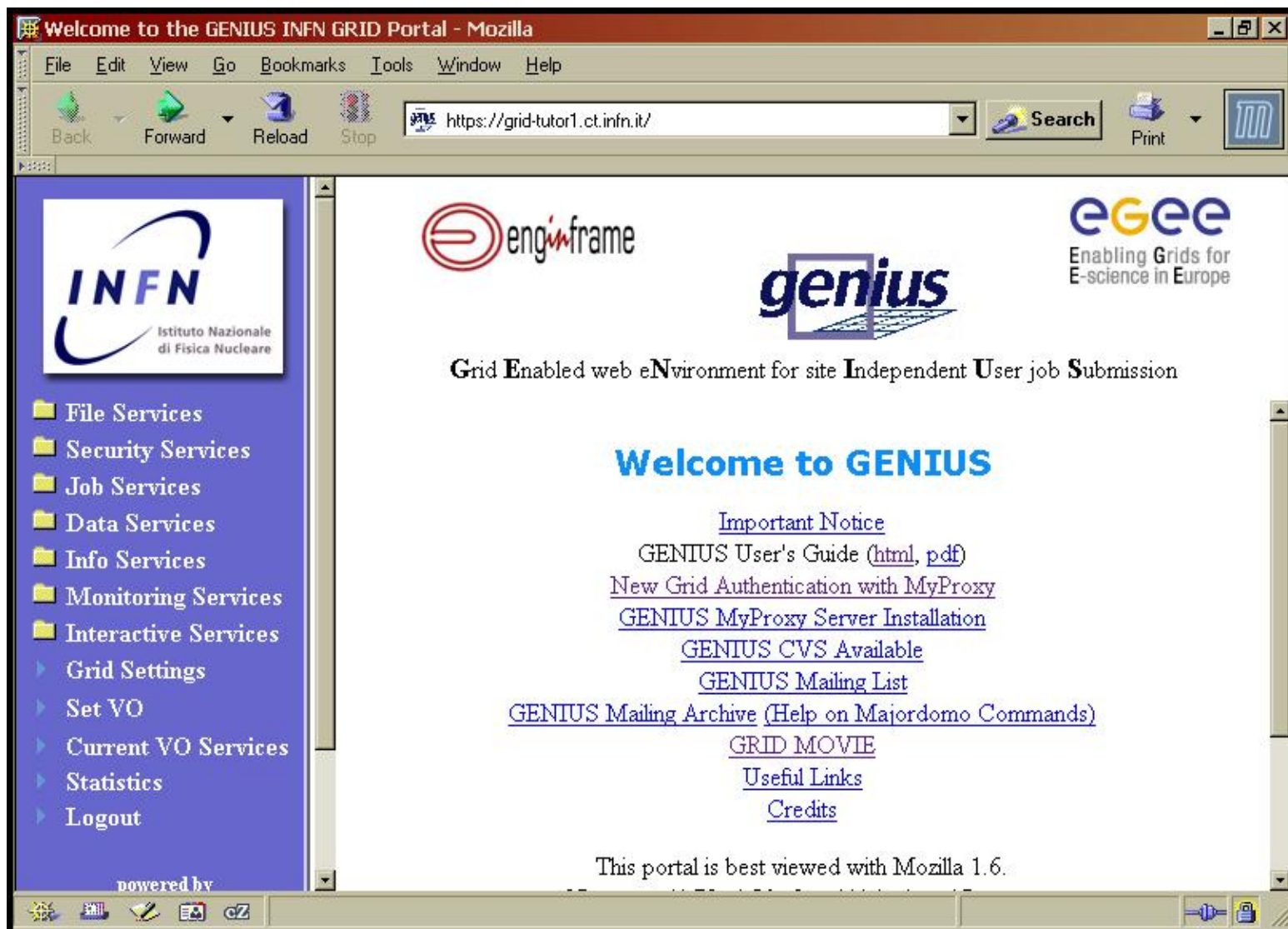


## Login and Configure GENIUS

The aim of this section is to provide you with an understanding of the GENIUS portal and how to run jobs using it. Subsequent sections of the tutorial will assume that the usage of GENIUS is understood.

The first stage in using GENIUS is to configure the various options connected to which Virtual Organization you are using. In EGEE terms a Virtual Organisation is a collection of individuals collaborating within a subject area, a single experiment or any other thematic grouping. For the purposes of this practical the Virtual Organisation (VO) called Gilda will be used. Gilda is a VO setup specially for purposes of training and dissemination with EGEE.

Open the following url in a new browser window or just click the link <https://grid-tutor.ct.infn.it> (if you are redirected to grid-tutor1.ct.infn.it this is not an error, it is just the load-balancing mechanism grid-tutor uses). You should see the following window:

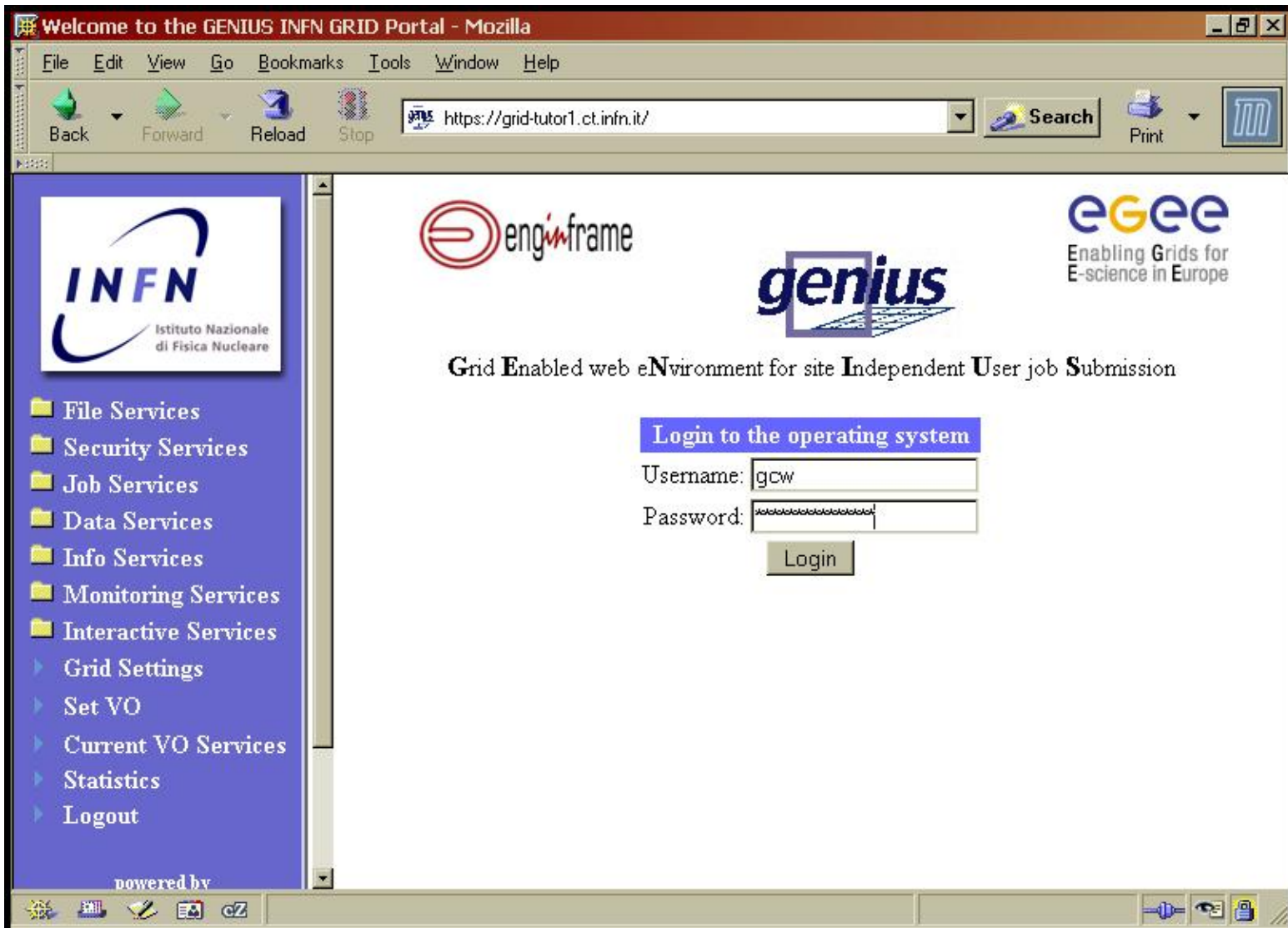


For the sake of clarity the frame on the left with the blue background will from here on be referred to as the sidebar whilst the frame to the right will be referred to as the main window.

grid-tutor is a version of GENIUS that has been customized for use in tutorials and includes features like temporary, 2 week, user accounts for attendees of tutorials and access to the latest stable version of the EGEE grid middleware (at the time of writing this document the latest stable EGEE middleware is LCG 2.2.0). Examples used in the tutorials are installed in each user account and hence these examples may be freely edited without affecting other users. The functionality as well as the style of the portal is deliberately consistent with other implementations of GENIUS and hence you will gain an understanding of how GENIUS is used by various VO's within EGEE.

When using grid-tutor it is necessary to authenticate yourself twice, once to access operating system functionalities and once to access grid functionalities. The first time an element of GENIUS is used that requires one of these authentications an appropriate dialog will appear. This authentication will be valid for 30 minutes after the last task using GENIUS (not necessarily a service request). This means that if your browser window is inactive for 30 minutes or more it will be necessary to repeat the authentication process.

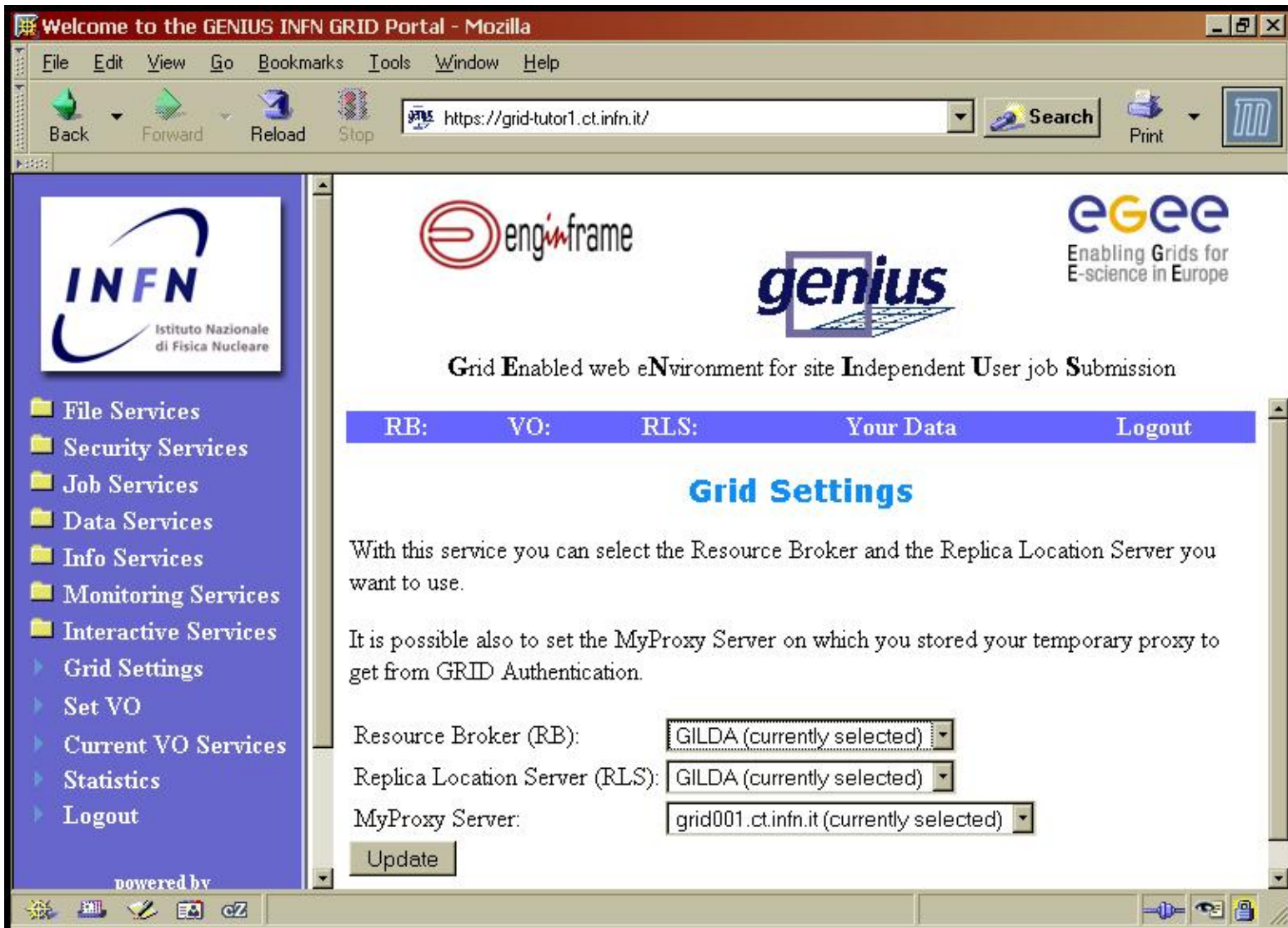
In the sidebar click on the link **Grid Settings**. This section is effectively where you store your preferences. You will be requested to log on to the operating system as shown in below:



If at this point you are asked about remembering your password for future use, then select **No**, or, in Mozilla, **Never for this site**. In Mozilla the dialog looks like this:



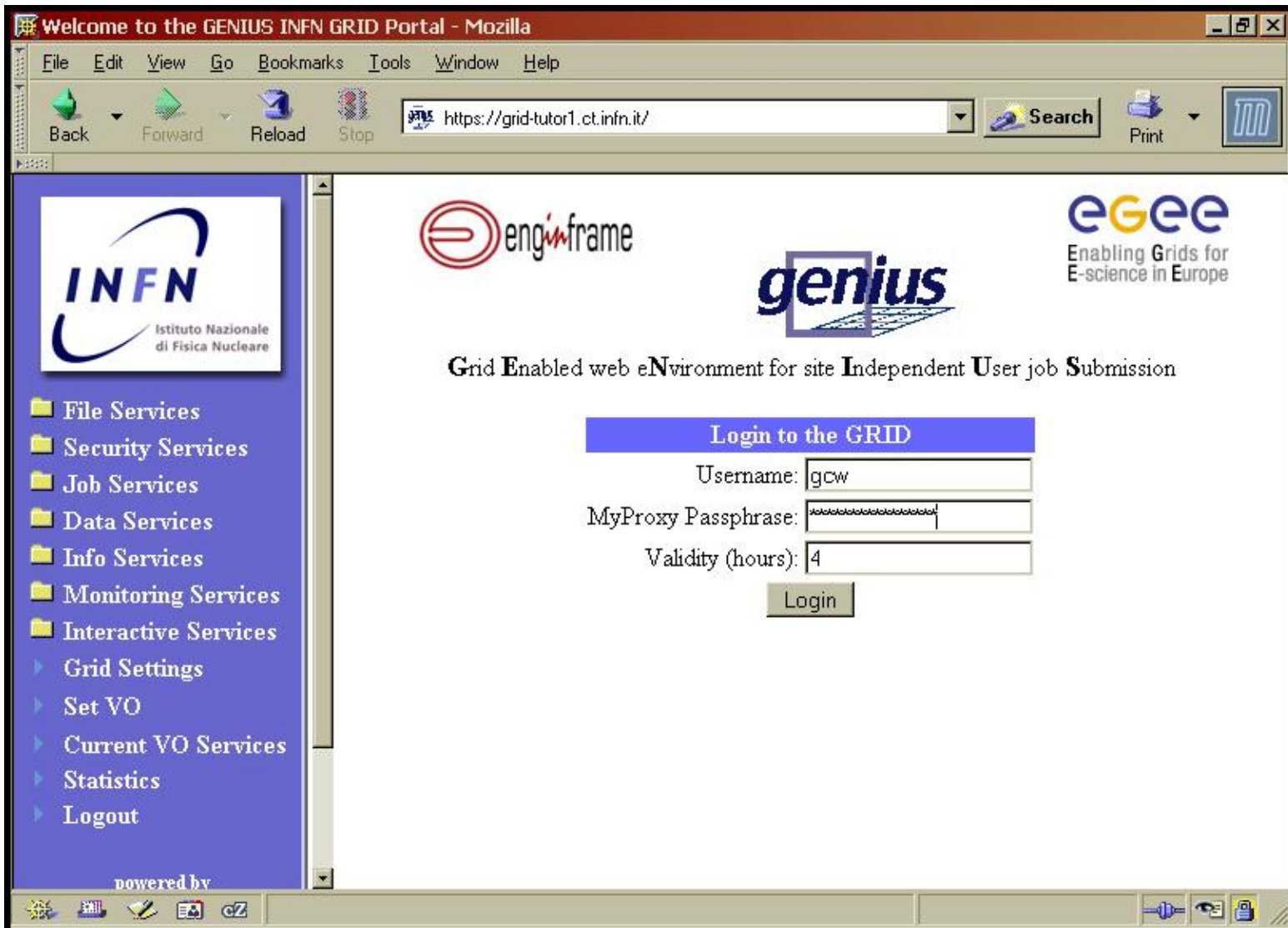
You are now presented with a screen asking for your choice of Resource Broker (RB), Replica Location Service (RLS) and Myproxy Server as shown below:



This dialog serves the dual purpose of both setting your preferences and loading them into the current session. Leave these settings at their default settings. At the time of writing this document these default settings are RB: GILDA, RLS: GILDA and Myproxy Server: grid001.ct.infn.it. These preferences are stored in a file in your account and hence the reason that this required an Operating system login. Click **Update** to continue.

The next stage is to select a VO. Any user may be a member of multiple VO's simultaneously. The GENIUS portal requires that users specify which VO they currently wish to use. This decision may be changed at any time. In the sidebar click **Set VO**. You will be presented with the dialog for authenticating yourself on to the grid as shown below:





This authentication connects to the MyProxy server you previously specified. A MyProxy server acts on your behalf to authenticate jobs you run to each system your job uses resources on. For the purpose of this tutorial the myproxy server has already been configured for you. The length of time you specify in this dialog is the amount of time MyProxy should act on your behalf without requiring you to repeat your authentication. Do not set this to a longer time than the default 4 hours. A short time length here increases security but decreases usability whilst a long time length decreases security but increases usability, hence the compromise time length of 4 hours. This is longer than the length of this tutorial.

Set VO is a grid operation because a list of VO's that you have membership of is established from your MyProxy certificate. This MyProxy certificate is the means by which the original authentication to the MyProxy server is made (this step was done for you). For the purposes of this tutorial the VO to be used is GILDA which should appear as the default, and often only, selected VO. You should be now seeing the window below:

Ensure GILDA is the selected VO and then click **Set**. The processes of authenticating to and configuring GENIUS is now complete.

[Next Section](#) ►

## Running a Job and Retrieving its Output

The next stage is to look at how to run a job. An important part of running a job is gathering its output since almost all jobs run on the grid are about analysing some data or running a computation (or a mixture of both) and then getting the results back. The amount of output returned from a job varies greatly since a job can save its output to the shared space of the VO on storage elements or return all of its output to the user (or again a mixture of both). In almost all cases a file recording any error messages whilst running the program on the remote worker node will be set back to the user (note that these errors are not the same as job failure messages, a programming error is very different to an error in the job submission file). We may therefore view a running job as having three stages; firstly submit the job, then monitor the job and finally retrieve the output.

## Submit the Job

In the sidebar click **Job Services**, followed by **Single Job** and then finally **Job Submission**. You should now be presented with the dialog below:

Welcome to the GENIUS INFN GRID Portal - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop <https://grid-tutor1.ct.infn.it/> Search Print

**INFN**  
Istituto Nazionale di Fisica Nucleare

**enginframe**

**genius**

**EGEE**  
Enabling Grids for E-science in Europe

Grid Enabled web eNvironment for site Independent User job Submission

RB: gilda VO: gilda RLS: GILDA Your Data Logout

## Job Submission

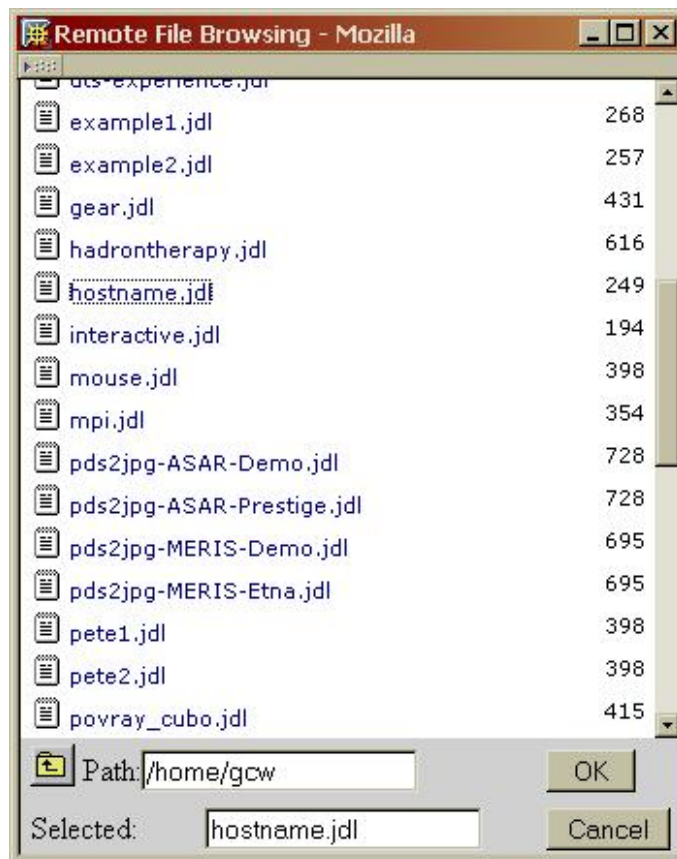
Please, select a JDL file to submit. You may also choose a Resource Broker to connect to.

JDL File

powered by  
EnginFrame 3.2  
compliant with  
LCG-2  
GRID.IT

Single Job  
up  
Job Submission  
Job Queue  
Job Data  
Clean Job Queues  
Close Interactive Job Session

In the EGEE middleware job requirements and details are described in a text file using the Job Description Language. By convention these files are given the extension '.jdl'. The subsequent sections of this practical will be centred around this language. For now however we are only interested in how to submit a job so we will not look into the details of the job we will submit. Click **Select** and you will be presented with a file selection dialog as shown below:



Select the file "hostname.jdl" and then click **OK**. This jdl file is very basic and will just query the hostname of the worker node the job actually runs on. The full path to "hostname.jdl" should now appear in the Job Submission window. Click **Next** to continue.

This next window allows you to select the Computing Element (CE) your job will run on. A CE is the front end to a group of worker nodes that actually run the various jobs. Typically a CE will have three job queues you can submit your job to. A short queue for jobs that are expected to run in a short amount of time, a long queue for jobs that will take a longer time and then an infinite queue for jobs that are expected to be extremely long running. One of the reasons for having these multiple queues is so that short jobs do not have to wait on long jobs before they can run. For now however we will just let the Resource Broker (RB) choose for us, as shown below:

The screenshot shows the "Welcome to the GENIUS INFN GRID Portal" in a Mozilla browser window. The address bar shows "https://grid-tutor1.ct.infn.it/". The page features logos for INFN (Istituto Nazionale di Fisica Nucleare), EnginFrame, genius, and eGEE (Enabling Grids for E-science in Europe). The main heading is "Grid Enabled web eNvironment for site Independent User job Submission". A navigation bar includes "RB: gilda", "VO: gilda", "RLS: GILDA", "Your Data", and "Logout". Below this, a message states: "Now you may also choose a specific Computing Element for your job." A form contains a "JDL File Selected" dropdown menu with "/home/gcw/hostname.jdl" selected, and a "Specify the CE Resource" dropdown menu with "Let the GILDA Resource Broker choose" selected. A "Submit Job" button is located below the form. On the left side, there is a sidebar with a "Single Job" section containing links for "Job Submission", "Job Queue", "Job Data", "Clean Job Queues", and "Close Interactive Job Session". At the bottom left, it says "powered by EnginFrame 3.2 compliant with LCG-2 GRID.IT".



compliant with  
[LCG-2](#)  
[GRID.IT](#)

We can now finally submit the job by clicking **Submit Job**. If the job successfully submitted then your browser window should look like this:

Welcome to the GENIUS INFN GRID Portal - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop <https://grid-tutor1.ct.infn.it/> Search Print

**INFN**  
Istituto Nazionale di Fisica Nucleare

**enginframe**

**genius**

**EGEE**  
Enabling Grids for E-science in Europe

Grid Enabled web eNvironment for site Independent User job Submission

RB: gilda VO: gilda RLS: GILDA Your Data Logout

```
Selected Virtual Organisation name (from --config-vo option): gilda
Connecting to host grid004.ct.infn.it, port 7772
Logging to host grid004.ct.infn.it, port 9002

===== edg-job-submit Success =====
The job has been successfully submitted to the Network Server.
Use edg-job-status command to check job current status. Your job identifi

- https://grid004.ct.infn.it:9000/1CWQ9B44b2sf2y101VVYag

The edg_jobId has been saved in the following file:
/home/gcw/.genius/.tmp_submittedjob_gcw
=====
```

powered by  
[EnginFrame 3.2](#)  
compliant with  
[LCG-2](#)  
[GRID.IT](#)

This is actually the output from the command line command that submitted the job (GENIUS just acts as a wrapper of these functionalities). The important point to note here is that <https://grid004.ct.infn.it:9000/1CWQ9B44b2sf2y101VVYag> is NOT a url. In fact this is the unique job identifier that has been created out of the url for the RB the job was submitted to plus a series of random characters.

We have now successfully submitted our job.

### Monitor the Job


To monitor our job in the side bar select **Job Queue**. The following dialog is then shown:



Welcome to the GENIUS INFN GRID Portal - Mozilla

File Edit View Go Bookmarks Tools Window Help




Back Forward Reload Stop <https://grid-tutor1.ct.infn.it/> Search Print



**Single Job**

- up
- Job Submission
- Job Queue
- Job Data
- Clean Job Queues
- Close Interactive Job Session

powered by  
[EnginFrame 3.2](#)  
compliant with  
[LCG-2](#)  
[GRID.IT](#)

Grid Enabled web eNvironment for site Independent User job Submission

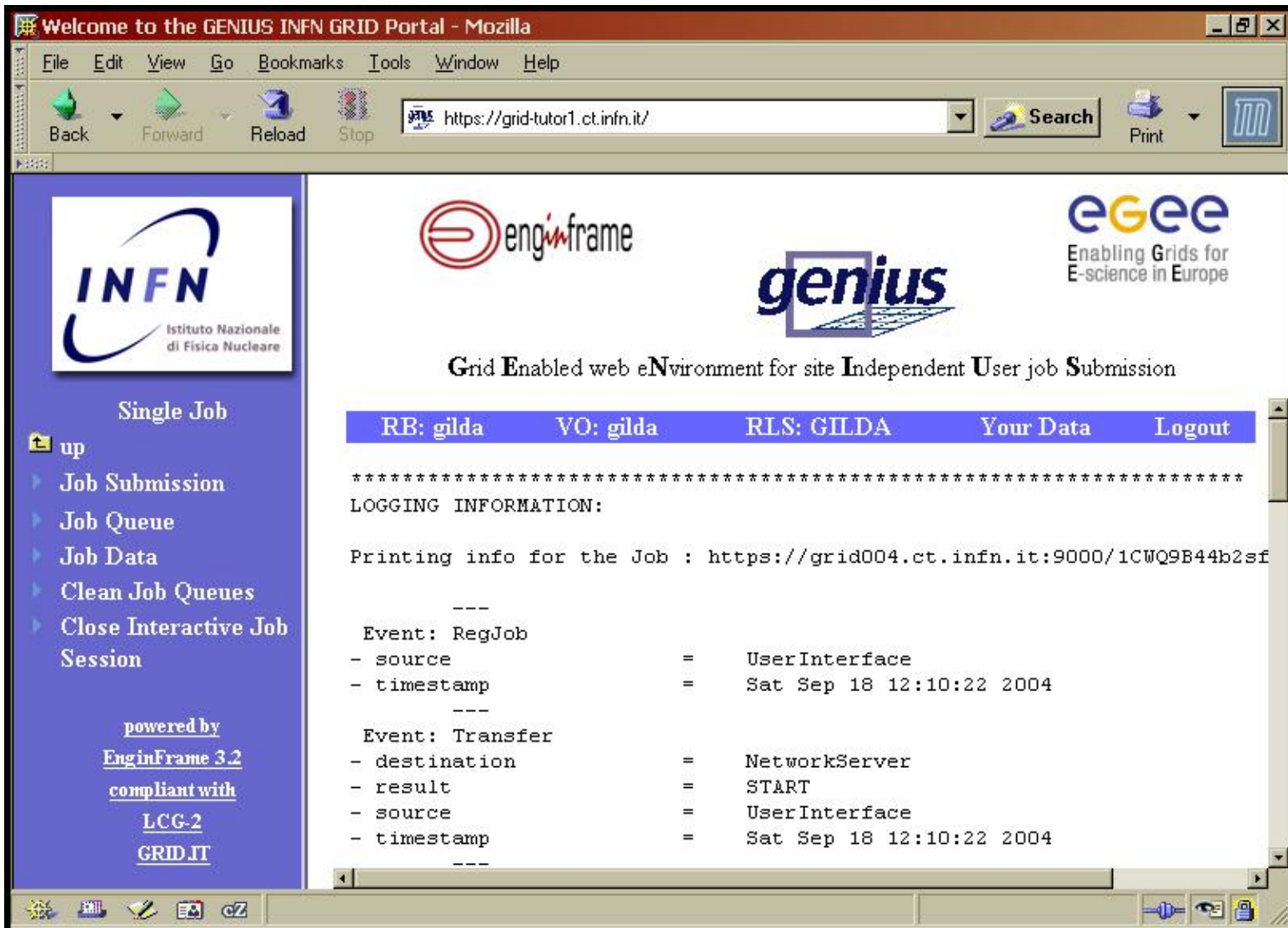
RB: gilda
VO: gilda
RLS: GILDA
Your Data
Logout

**Job Queue**

#	Job ID	JDL Name	Last Update	Destinati
1	<a href="#">1CWQ9B44b2sf2yl01VVYag</a>	<a href="#">/home/gcw/hostname.jdl</a>	Sat Sep 18 12:10:48 2004	gilda-ce-01.pd.infn.it:2119/jo

https://grid-tutor1.ct.infn.it/inf.grid.xml?\_uri=/inf.grid/job-queue

Note: this automatically updates on a regular interval, should you however wish to force the screen to reload do NOT reload the whole window by clicking on the refresh button, this will take you back to the grid-tutor home page. To force the page to refresh click again on the **Job Queue** link in the sidebar. From here we can obtain information about the job we just submitted. If you firstly click on the url for Job ID you can see the log of how your job is progressing as shown below:



This is useful for tracking what route your job is taking to reach the worker node and when it reached each point. Click the "Back" browser button to return to the previous dialog. Clicking on the link of the jdl file just shows the contents of the jdl file, which is not of interest here. What is of interest is the final column which gives the job status. There are several values that can appear in this column as explained in the following table:

Flag	Meaning
Submitted	The job has been submitted and a log of this has been made by the Logging and Bookkeeping service.
Wait	The RB is attempting to find available CE's that support the jobs requirements.
Ready	The RB is sending the job to the selected CE.
Scheduled	The job has been scheduled by the queue manager on the CE.
Running	The job is currently running on a WN behind your select CE queue.
Done	The job terminated without grid errors.
Cleared	The job output has been retrieved.
Abort	The job was aborted by the middleware.
Cancelled	The job was cancelled by the user.

When watching the job run it sometimes appears that the job is taking a very long time for each stage. This is not always the case. The information being displayed about the status of the job is being passed from the Logging and Bookkeeping service. This service polls the actually grid elements involved with your job on a regular interval so will not notice a change of state until the next time that element is polled. Very occasionally your job might be aborted, this is normally caused by a site on the grid that is not configured properly, in this case your job will normally be automatically resubmitted to a different site.

Once your job shows the status as "Done" you can move on to retrieving the jobs output.





#### Retrieve the output.

There should now be a button on the end of the line describing your job that says Get Output. Click this button. You are now presented with the below dialog:

Welcome to the GENIUS INFN GRID Portal - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop <https://grid-tutor1.ct.infn.it/> Search Print

Grid Enabled web eNvironment for site Independent User job Submission

RB: gilda	VO: gilda	RLS: GILDA	Your Data	Logout
<a href="#">Destroy</a>	Directory contents - 20040918_141820_1CWQ9B44b2sf2yl01VVYag			
	<a href="#">hostname.err</a>	0	<a href="#">hostname.out.txt</a>	23

Single Job

- up
- Job Submission
- Job Queue
- Job Data
- Clean Job Queues
- Close Interactive Job Session



powered by  
EnginFrame 3.2  
compliant with  
LCG-2  
GRID.IT


This dialog can also be reached after retrieving the jobs data by going to **Job Data** in the sidebar and then selected the output of your job by using its Job ID as shown below:

Welcome to the GENIUS INFN GRID Portal - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop <https://grid-tutor1.ct.infn.it/> Search Print



Grid Enabled web eNvironment for site Independent User job Submission

RB: gilda	VO: gilda	RLS: GILDA	Your Data	Logout	
Name			Created on	Items	Actions
			<a href="#">tmp1095341822026.ef</a>	Sep 16, 2004 3:37:02 PM	1 <a href="#">Destroy</a>
			<a href="#">20040918_141820_1CWQ9B44b2sf2yl01VVYag</a>	Sep 18, 2004 2:18:11 PM	2 <a href="#">Destroy</a>

Single Job

- up
- Job Submission
- Job Queue
- Job Data
- Clean Job Queues
- Close Interactive Job Session

powered by  
EnginFrame 3.2  
compliant with



powered by  
[EnginFrame 3.2](#)  
compliant with  
[LCG-2](#)  
[GRID.IT](#)

Done



Click on either of the files that have been retrieved from your job. If the program run by the job was successful the file "hostname.err" should be empty. The file "hostname.out" should contain the name of the WN the job ran on.

### Clean the GENIUS job queues

The final stage to running the job is to clean up the data. Since the job is only a test job there is no need to keep the output from the job. This stage should be run after each example in the subsequent sections. In the sidebar click **Clean Job Queues**. You are now presented with this dialog:

Leave the value of Select Queue on "Current RB" and then click **Clean**. If you now go to **Job Data** you should find that there is no data available.

[◀ Previous Section](#)

[Next Section ▶](#)

## Edit A File

The final task you need to learn so as to be able to run jobs in GENIUS is how to edit a file. The menu for file services is at the top level so in the sidebar click **up** twice. From here you can select **File Services** and then **Edit a File**. You should be presented with the below dialog:

Welcome to the GENIUS INFN GRID Portal - Mozilla

File Edit View Go Bookmarks Tools Window Help

Back Forward Reload Stop  Search Print

**INFN**  
Istituto Nazionale di Fisica Nucleare

**enginframe**

**genius**

**eGEE**  
Enabling Grids for E-science in Europe

Grid Enabled web eNvironment for site Independent User job Submission

RB: gilda VO: gilda RLS: GILDA Your Data Logout

## Edit a File

With this service you may edit a file in your home directory.

Select File to Edit

powered by  
[EnginFrame 3.2](#)  
compliant with  
[LCG-2](#)

As previously select the file "hostname.jdl". Note that you can now select any file from your account as opposed to just a jdl file as is the case when submitting a job. To continue click **Open**. You are then presented with the following:

Here you can edit the contents of the file and save the file to a new name. For the moment just save this file to the new name of "hostnamenew.jdl" by editing the value in the FileName box. To save these changes click **Save**.

[Previous Section](#)

[Next Section](#)



## Introduction

In the previous section you submitted the job "hostname.jdl" to gilda. If you now look at the source you will see the following:

Line Number	Code
1	Type = "Job";
2	JobType = "Normal";
3	Executable = "/bin/hostname";
4	StdOutput = "hostname.out";
5	StdError = "hostname.err";
6	OutputSandbox = {"hostname.err","hostname.out"};
7	Arguments = "-f";
8	RetryCount = 7;

As you can see the jdl file is basically a list of parameters and the associated values where the value can actually be a single value, a list of values or a complex expression. For the purposes of this tutorial all of these parameters (except Arguments) must be present in each jdl file and have appropriate values set. There are cases where not all of these parameters are needed, but this tutorial will not be looking at these cases.

To explain what is going on here we will look at each line in turn:

1. The value of Type must always be Job. This line exists in the jdl for historical reasons only.
2. The JobType specifies the style of the job. For all the exercises in this tutorial a job type of Normal will be used. Possible other values include Interactive and Checkpointable for jobs which require a more sophisticated level of control during their run.
3. The Executable parameter specifies the executable/command that will be run on the CE. There are two possibilities here. Firstly you can specify an executable that lies already on the remote CE. In this case the absolute path, possibly including environmental variables should be specified. The other possibility is to provide a local filename as the executable, which will be looked at in the second exercise.
4. The standard output (STDOUT) of the executable specified will be saved to the filename given as the value to the StdOutput parameter. Note this file will be local to the remote CE.
5. As for STDOUT but this time for standard error (STDERR).
6. The OutputSandbox parameter specifies a list of files that were generated by the executable and that you want to retrieve (the easiest way of viewing simple files).
7. Any arguments that are needed by the executable are specified in the Arguments parameter. Note that this must a single string value. Multiple arguments are passed as part of the string in the same way as they would on the command line (normally just space separated).
8. This final parameter (RetryCount) specifies how many times the job should be resubmitted in the case of a grid component failing (not the same as an error in the executable).

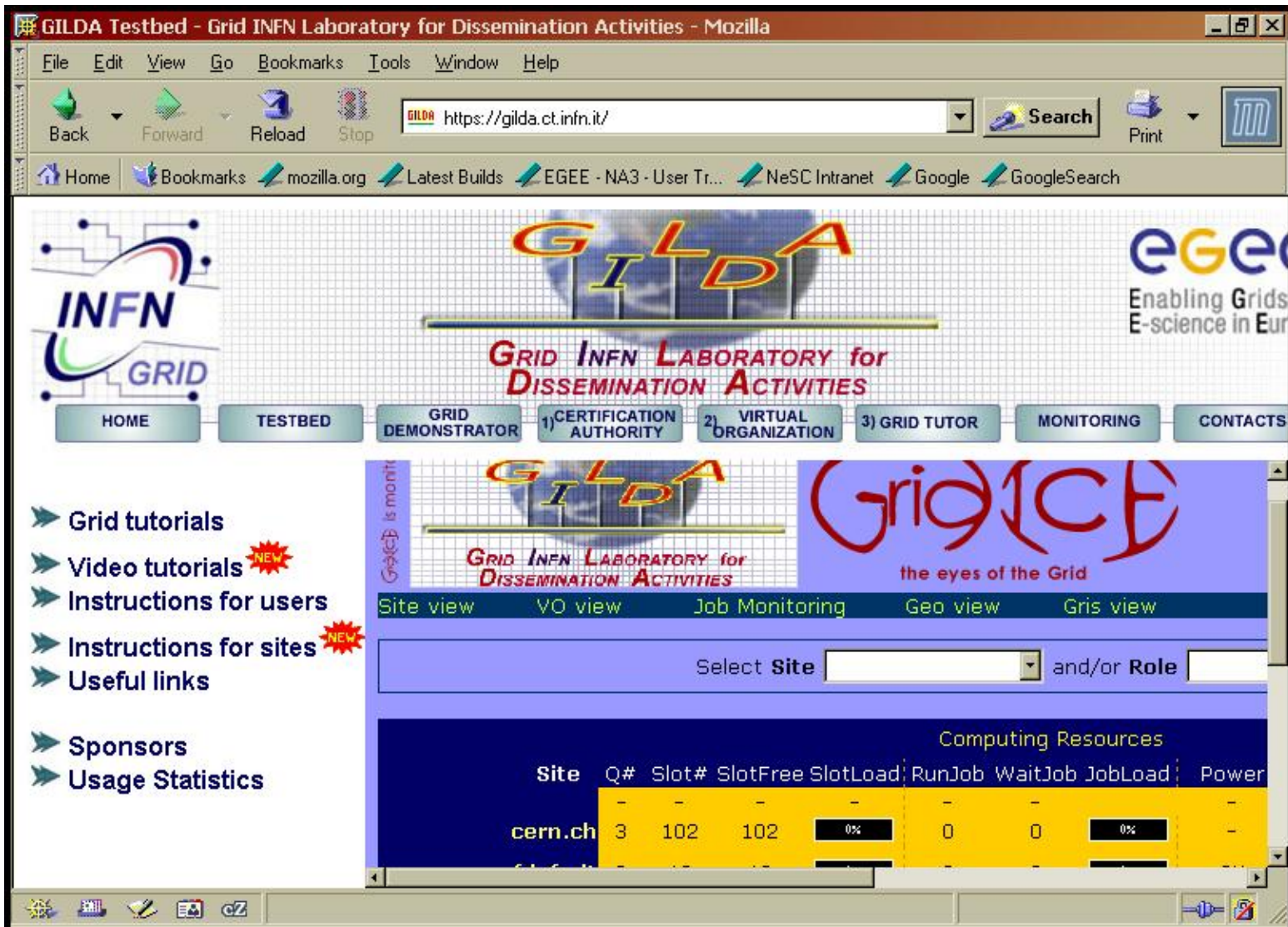
## Exercise 1

Modify "hostname.jdl" so that instead of calling /bin/hostname it calls the script "start.sh" instead. This file (as with all files used in this tutorial) already exists in your account and looks like the following:

```
#!/bin/sh
sleep 5
hostname -f
```

You will need to change the executable to /bin/sh and the argument to start.sh. You will also need to use the new parameter InputSandbox. The InputSandbox is a list of all the files that need to be sent (in EGEE terminology 'staged') to the CE. For this example you need to send start.sh . A typical usage of bash scripts in the grid is to set environmental variables the actual executable needs and to run a combination of different programs consecutively on the same CE. Once you have made these edits run this job and check your output to see if your job was successful. As stated previously a successful job will have an empty "hostname.err" whilst "hostname.out" will contain the name of the CE the job ran on.

Whilst you are waiting for your job to return its output, open a new browser window (or just click the following link) at <https://gilda.ct.infn.it/>. Now click on the Monitoring button. You should now be seeing a window similar to the following image:



You are now looking at GridIce, a system for monitoring the state of the grid, in particular the usage of different parts of the grid. This version is just monitoring gilda. Explore gilda by clicking on the names of the different sites and then the names of the grid elements at each site.

## Exercise 2

As well as bash scripts you can write your own programs in C. Modify your jdl file to run the executable "myhostname". The source for myhostname can be found in "myhostname.c" and is:

```
#include <stdio.h>
#include <malloc.h>

#define BUF_SIZE 1000
int main( int argc, char *argv[] ) {
    char *hostname;
    hostname = (char *) malloc(BUF_SIZE);
    gethostname(hostname,255);
    printf("host is %s\n",hostname);
    free(hostname);
    return 1;
}
```

Note that this time you do not need the Arguments parameter. After making the changes run this job and check for success (this should be done after every exercise).

## Exercise 3

In exercise 2 we saw how a C program could be run as a grid job by sending the program as part of the input sandbox. In many cases however this is not a practical solution, for example the executable itself is very large and requires many files in many folders, or alternately you do not have access to the same compiler and environment as is used on the CE (as is the case in this tutorial). The solution then is to compile your program on the CE itself. The easiest way to do this is by using a bash script to run the compiler and then the executable. Modify your jdl file so that the script "buildandrun.sh" is run with the parameter "myhostname". You will need to ensure that both "buildandrun.sh" and "myhostname.c" are staged to the CE. The script "buildandrun.sh" is simply:

```
#!/bin/sh
gcc $1.c -o $1
./$1
```

## Exercise 4

In all of the previous examples you have been working with programs that had no dependencies on software other than the software installed on all WN's. It is normally not a realistic option to send all of this dependant software with the jdl, and in many cases the software installed already on some of the WN's. Similarly there has been no question raised as to what hardware the WN has (e.g. number of CPU's or amount of memory). The jdl file is able to handle these job requirements by the use of the GLUE Schema. All CE's advertise what software their WN's have installed and what their hardware/operating system is to a database on the Information Service. When deciding where to send a job the Resource Broker matches the requirements you have specified against all of the CE's in your VO to see which (if any) can run your job. Consider the following example (the file "demtools.jdl" in your account):

```
Type = "Job";
JobType = "Normal";
Executable = "/bin/sh";
StdOutput = "demtools.out";
StdError = "demtools.err";
InputSandbox = {"start_demtools.sh", "mount_sainte_helens_WA.dem", "grand_canyon_AZ.dem"};
OutputSandbox = {"demtools.out", "demtools.err", "mount_sainte_helens_WA.ppm", "mount_sainte_helens_WA.wrl", "grand_canyon_AZ.ppm", "grand_canyon_AZ.wrl"};
RetryCount = 7;
Arguments = "start_demtools.sh";
Requirements = Member("DEMTOOLS-1.0", other.GlueHostApplicationSoftwareRunTimeEnvironment) || Member("DEMTOOLS", other.GlueHostApplicationSoftwareRunTimeEnvironment);
```

The final line in this file states that the job requires the CE to either support DEMTOOLS or DEMTOOLS-1.0. If you wish to run this example then when you have the output you should open in your browser either "mount\_sainte\_helens\_WA.wrl" or "grand\_canyon\_AZ.wrl"

The Glue Schema supports many such job requirements. For this exercise add to your jdl the requirement

```
other.GlueCEUniqueID == "grid010.ct.infn.it:2119/jobmanager-lcgpbs-short";
```

When you submit the job and reach the point where you usually let the RB choose where to run the job, have a look at the other options in the drop down menu, you should not be surprised at what the other options are.

[◀ Previous Section](#)