



Enabling Grids for E-science

Practical Work on EGEE using gLite

Marc Ozonne

Peter Praxmarer

Markus Baumgartner

CERN

Geneva/Switzerland

**Grid@work,
Sophia-Antipolis, 10 Octobre 2005**

www.eu-egee.org



- <https://glite-tutor.ct.infn.it>
- Choose 'Grid Settings' and log in to the portal



Grid Enabled web eNvironment for site Independent User job Submission

Login to the operating system

Username:

Password:

- Username: sophiaantipolis01 ... sophiaantipolis30
- Password: GridSOP01 ... GridSOP30

Grid Settings

With this service you can select the Resource Broker and the Replica Location Server you want to use.

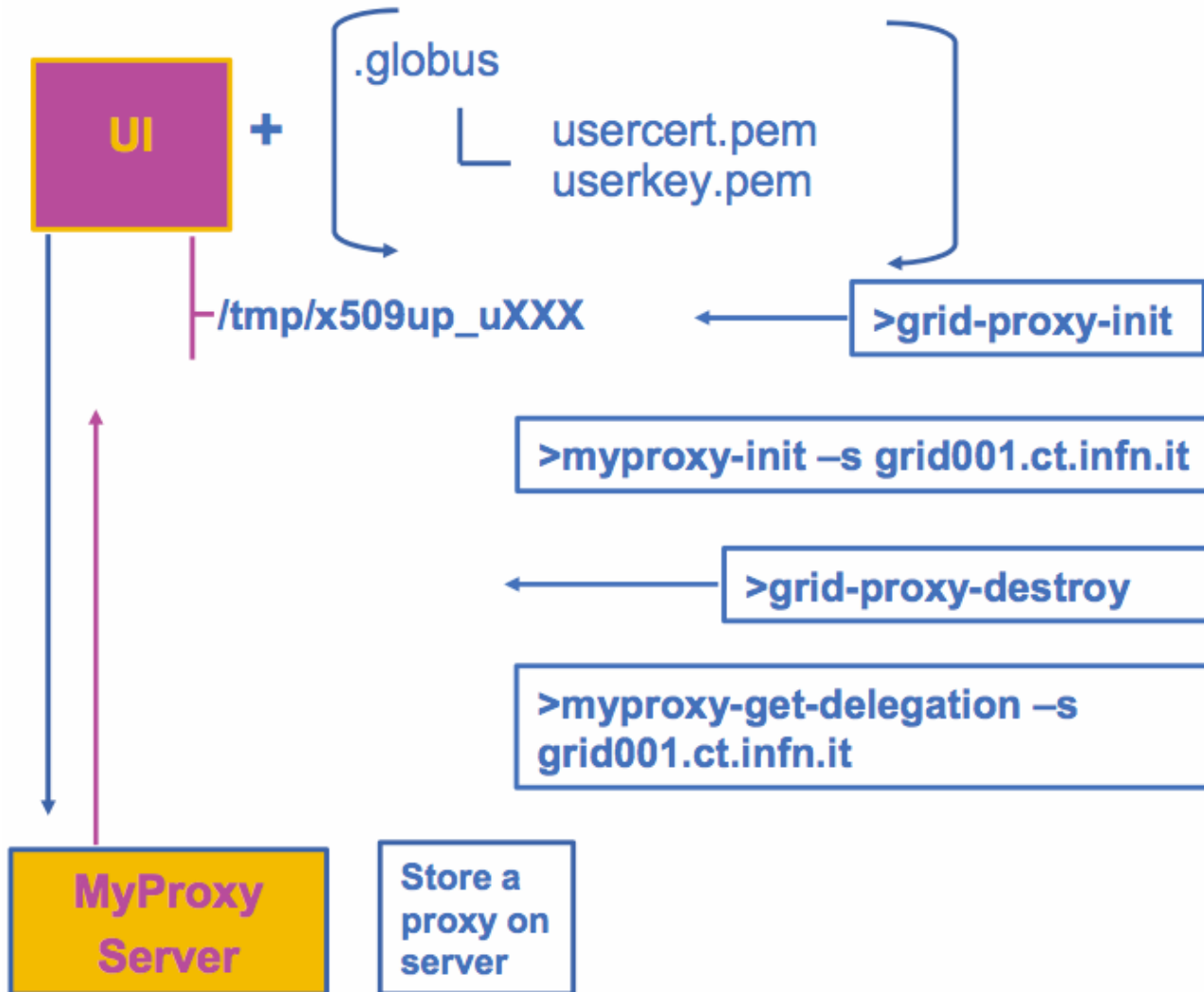
It is possible also to set the MyProxy Server on which you stored your temporary proxy to get from GRID Authentication.

Resource Broker (RB):

Catalog:

MyProxy Server:


- **Resource Broker** knows all resources available and assigns jobs to them
- **Catalog** defines the Replica Catalog used by default
- **MyProxy Server** defines the server storing the myproxy certificate



- Create your own myproxy certificate:
 - Login to glite-tutor.ct.infn.it using ssh
 - > ssh sophiaantipolisXX@glite-tutor.ct.infn.it
 - Create a proxy certificate
 - > grid-proxy-init
 - Verify the proxy certificate
 - > grid-proxy-info
 - Initialize your MyProxy certificate
 - > myproxy-init -s grid001.ct.infn.it
 - Provide the passphrase "**SOPHIA-ANTIPOLIS**" to decrypt the userkey
 - Choose a passphrase for the MyProxy certificate
 - Verify the MyProxy certificate on the server
 - > myproxy-info -s grid001.ct.infn.it
 - Destroy your proxy certificate
 - > grid-proxy-destroy
 - Get a proxy certificate from the MyProxy server:
 - myproxy-get-delegation -s grid001.ct.infn.it

- **Get information on the currently used proxy certificates**
 - Info on the proxy certificate
 - Info on the certificate stored on the MyProxy server
- **Change the GENIUS login password**
- **Logout from GENIUS and remove the proxy certificate**

Security Services

-  **up**
- ▶ **Info on proxy**
- ▶ **Info on MyProxy**
- ▶ **Change GENIUS Password**
- ▶ **Remove your proxy and Logout**

Exercise 2: Security Services

User Interface on glite-tutor.ct.infn.it	Check with the GENIUS WebUI
<ul style="list-style-type: none"> > grid-proxy-info > myproxy-info -s grid001.ct.infn.it 	<ul style="list-style-type: none"> Click on “Info on proxy“ Click on “Info on MyProxy“
<ul style="list-style-type: none"> > myproxy-destroy -s grid001.ct.infn.it > myproxy-info -s grid001.ct.infn.it 	<ul style="list-style-type: none"> Click on “Info on MyProxy“ (Error)
<ul style="list-style-type: none"> > grid-proxy-destroy > grid-proxy-info > grid-proxy-init 	<ul style="list-style-type: none"> Click on “Info on proxy“(Error)
<ul style="list-style-type: none"> > myproxy-init -s grid001.ct.infn.it > myproxy-info -s grid001.ct.infn.it 	<ul style="list-style-type: none"> Click on “Info on MyProxy“
<ul style="list-style-type: none"> > myproxy-get-delegation -s grid001.ct.infn.it 	
<ul style="list-style-type: none"> > grid-proxy-info 	<ul style="list-style-type: none"> Click on “Info on proxy“

- Create a file named 'myhelloworld.sh'

File Services



up

- ▶ Create a File
- ▶ View a File
- ▶ Edit a File
- ▶ Rename a File/Directory
- ▶ Delete a File/Directory
- ▶ Create a Directory
- ▶ Upload a TAR ball
- ▶ Upload a file
- ▶ Show the Environment

```
#!/bin/sh
```

```
MYNAME="Your Name"
```

```
WORKER_NODE=`hostname`
```

```
echo "Hello ${MYNAME}"
```

```
echo „Greetings from ${WORKER_NODE}!“
```


Exercise 3b: Create a JDL file

File Services



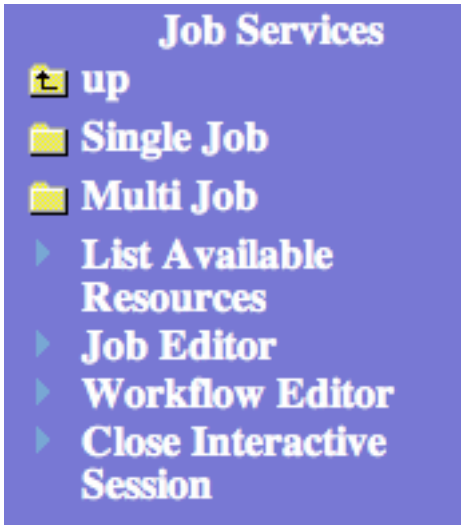
up

- ▶ Create a File
- ▶ View a File
- ▶ Edit a File
- ▶ Rename a File/Directory
- ▶ Delete a File/Directory
- ▶ Create a Directory
- ▶ Upload a TAR ball
- ▶ Upload a file
- ▶ Show the Environment

- Create a file named 'myhelloworld.jdl'

```
[
Executable="myhelloworld.sh";
StdOutput="std.out";
StdError="std.err";
VirtualOrganisation="gilda";
InputSandbox={"myhelloworld.sh"};
OutputSandbox={"std.out", "std.err"};
]
```

- **JDL is the acronym for 'Job Description Language'**
 - Specifies what a job consists of: job characteristics, job requirements and preferences, job data requirements
 - More later



- Using Genius :
 - From the Main menu choose “Job Services”
 - Choose “List Available Resources”
 - Specify the previously created JDL file and query for matching resources
- Using the command line interface:
 - In the directory you have created the JDL file
 - Issue the following command:

```
> glite-job-list-match <JDL-file>
```

Single Job

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

- Using **Genious**
- From the “Job Services” menu choose “Single Job”
 - Choose “Job Submission” and specify the JDL file
 - Proceed by pressing the “next” button
 - Submit the job by pressing the “Submit Job” button

- Using the **command line interface**

- Submit the job :

```
> glite-job-submit <JDL-file>
```

- The glite-job-submit command returns the **job identifier** which is going to be used in subsequent slides.

- Genious

Job Queue							
#	Job ID	JDL Name	Last Update	Destination	Status	Exit Code	Action
6	bGVFFbVO93hXuZz0Rt0MkA	/home/pprax/myhelloworld.jdl	Aug 12 13:44:53 2005 CEST	gilda-ce-01.pd.infn.it:2119/jobmanager-lcgpbs-short	Done	0	Get Output

- Query the job status by pressing “Job Queue”
 - The ,Status‘ changes from “Ready‘ to ,Scheduled‘ to ,Running‘, and eventually to ,Done‘

- Command line interface

- This function is equal to the

```
> glite-job-status <JobID>
```

command.

- **Genious**

							Job Queue	
#	Job ID	JDL Name	Last Update	Destination	Status	Exit Code	Action	
6	bGVFFbVO93hXuZz0Rt0MkA	/home/pprax/myhelloworld.jdl	Aug 12 13:44:53 2005 CEST	gilda-ce-01.pd.infn.it:2119/jobmanager-lcgpbs-short	Done	0	Get Output	

- Fetch the output by pressing “Get Output“
- You can inspect the files by clicking on them
- **Command line interface**
- You can do this with
 - > `glite-job-output <JobID>`
- The exact location of the output directory is given to you as the command returns.
- Go there and inspect the files.

- **The JDL is used to define**
 - Job characteristics
 - Job requirements
 - Data requirements
- **Based on Condor's CLASSified ADvertisement language (ClassAd)**
 - Fully extensible
 - A ClassAd
 - *Constructed with the classad construction operator []*
 - *It is a sequence of attributes separated by semi-colons.*
 - *An attribute is a pair (key, value), where value can be a Boolean, an Integer, a list of strings, ...*
 - `<attribute> = <value>;`

- **The supported attributes are grouped into two categories:**
 - **Job Attributes**
 - Define the job itself
 - **Resources**
 - Taken into account by the Workload Manager for carrying out the matchmaking algorithm (to choose the “best” resource where to submit the job)
 - **Computing Resource**
 - *Used to build expressions of Requirements and/or Rank attributes by the user*
 - *Have to be prefixed with “other.”*
 - **Data and Storage resources**
 - *Input data to process, Storage Element (SE) where to store output data, protocols spoken by application when accessing SEs*

- JobType
 - *Normal* (simple, sequential job), *DAG*, *Interactive*, *MPICH*, *Checkpointable*
- Executable (**mandatory**)
 - The command name
- Arguments (**optional**)
 - Job command line arguments
- StdInput, StdOutput, StdError (**optional**)
 - Standard input/output/error of the job
- Environment
 - List of environment settings
- InputSandbox (**optional**)
 - List of files on the UI's local disk needed by the job for running
 - The listed files will be staged automatically to the remote resource
- OutputSandbox (**optional**)
 - List of files, generated by the job, which have to be retrieved

- Requirements
 - Job **requirements on computing resources**
 - Specified using attributes of resources published in the Information Service
- Rank
 - **Expresses preference** (how to rank resources that have already met the Requirements expression)
 - Specified using attributes of resources published in the Information Service
 - If not specified, default value defined in the UI configuration file is considered
 - Default: - *other*.**GlueCEStateEstimatedResponseTime** (the lowest estimated traversal time)
 - Default: *other*.**GlueCEStateFreeCPUs** (the highest number of free CPUs) for parallel jobs (see later)

- **InputData**
 - Refers to data used as input by the job: these data are published in the Replica Catalog and stored in the Storage Elements
 - LFNs and/or GUIDs
- **InputSandbox**
 - Executable, files etc. that are sent to the job
- **DataAccessProtocol (mandatory if InputData has been specified)**
 - The protocol or the list of protocols which the application is able to speak with for accessing *InputData* on a given Storage Element
- **OutputSE**
 - The Uniform Resource Identifier of the output Storage Element
 - RB uses it to choose a Computing Element that is compatible with the job and is close to the Storage Element

- **Requirements allow you to influence the matchmaking process**
 - Used to specify software requirements (e.g. a specific software has to be installed on the worker node in order to execute the job)
 - A resource must fulfil the constraints specified in order to be allowed to execute the job (e.g. a specified number of processors must be available; a job requires some minimum amount of RAM being available, etc.)
 - Example:

```
Requirements=other.GlueHostOperatingSystem=="LINUX"  
&& other.GlueCEStateFreeCPUs>=4;
```



Allows you to influence the matchmaking process!

- Objective: Execute a POVRay rendering job
 - Requirements: POVRay needs to be installed on the worker node
- Create the POVRay scene description file “nuggets.pov“:

```
// Thanks to „Dazza“
// http://astronomy.swin.edu.au/~pbourke/povray/scc3/final/
global_settings{radiosity{}}
```

```
#declare f=function{pigment{granite}};
```

```
union {
    isosurface {
        function{x*x+y*y+z*z-f(x,y,z).red}
        pigment{color rgb<1,1,2>/2}
        finish{specular 1}
    }
    sphere{<-.13,-.05,-.7> .05
        pigment{rgb<4,2,0>}
    }
}
translate z*1.5
}
```

```
light_source{0,1}
```

- Create a script named “startNuggets.sh“ which calls “povray“ and renders the scene on the compute element

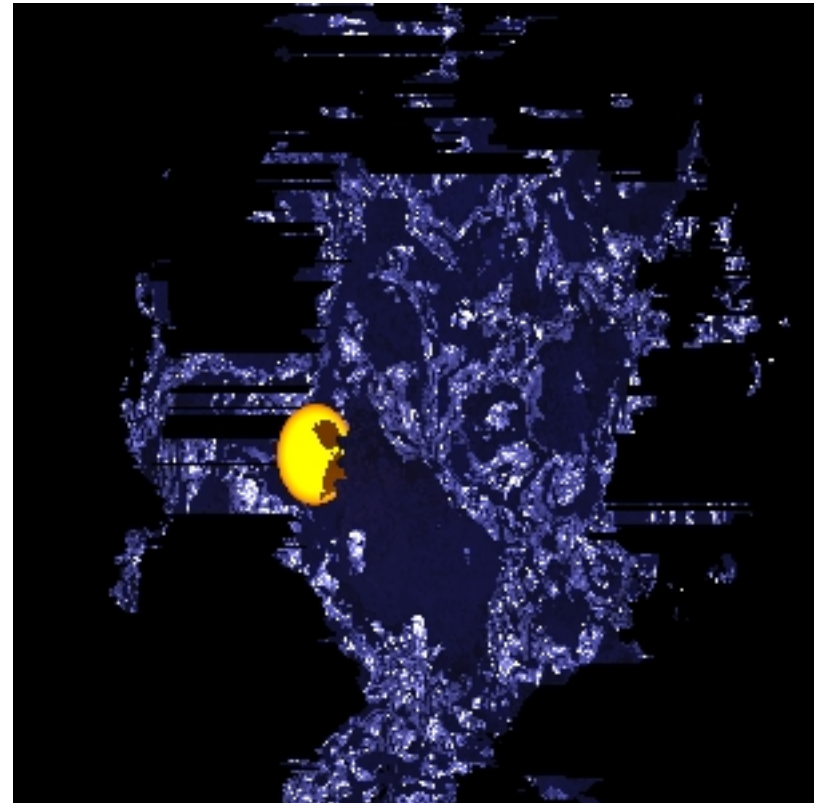
```
#!/bin/sh
```

```
povray +Inuggets.pov +Onuggets.tga +FT +W300 +H300 +V -D +X
```

- Create the JDL file nuggets.jdl for rendering the scene:

```
[  
Executable="startNuggets.sh";  
StdOutput="nuggets.out";  
StdError="nuggets.err";  
  
InputSandbox={"startNuggets.sh", "nuggets.pov" };  
OutputSandbox={"nuggets.out", "nuggets.err", "nuggets.tga"};  
  
Requirements= Member("POVRAY-3.5",  
    other.GlueHostApplicationSoftwareRunTimeEnvironment);  
]
```

- Submit the job
- Fetch the output as soon as it is available
- View the resulting image with your preferred image viewer



- **Objective:** Write your own C/C++ program and execute it on the Grid

- Create a file “myprogram.cpp” with the following content:

```
#include <iostream>

int main() {
    std::cout << “Hello World!” << std::endl;
}
```

- Compile and run

```
g++ -o myprogram myprogram.cpp && ./myprogram
```


- Create the JDL file ,myprogram.jdl' with the following content:

```
[  
    Executable="myprogram";  
  
    StdOutput="std.out";  
    StdError="std.err";  
  
    InputSandbox="myprogram";  
    OutputSandbox={"std.out","std.err"};  
]
```

- Submit it to the Grid
- Check the status
- Fetch and inspect the output

- Metainformation about the state of the Grid is provided through R-GMA (Relational - Grid Monitoring Architecture)
- In order to use the R-GMA commands you will need a working proxy.
- To start working with R-GMA just run the command `rgma` from the prompt `[glite-tutor] /home/sophiaantipolisXX > rgma`

```
Welcome to the R-GMA virtual database for Virtual Organisations.  
=====
```

```
Your local R-GMA server is:
```

```
https://rgmasrv.ct.infn.it:8443/R-GMA
```

```
You are connected to the following R-GMA Registry services:
```

```
https://rgmasrv.ct.infn.it:8443/R-GMA/RegistryServlet
```

```
You are connected to the following R-GMA Schema service:
```

```
https://rgmasrv.ct.infn.it:8443/R-GMA/SchemaServlet
```

```
Type "help" for a list of commands.
```

```
rgma>
```

- **Displays help for a specific command**

```
rgma> help <command>
```

- **Exit the R-GMA command line**

```
rgma> exit or quit
```

- **To show a list of all table names:**

```
rgma> show tables
```

- **To show information about a table MyTable :**

```
rgma> describe MyTable
```

- **To show a table of properties for the current session:**

```
rgma> show properties
```

- **To show a list of all R-GMA producers that produce the table MyTable :**

```
rgma> show producers of MyTable
```

- Show all tables that are present in the Schema:

```
show tables
```

- Choose one table where you want insert your information and see its attribute:

```
describe <NameTable>
```

- Some example queries:

```
select * from Site
```

```
select SysAdminContact,UserSupportContact from Site
```

```
select Name from Site where Latitude > 0
```

```
select Endpoint, Type from Site,Service
```

```
where Site.Name = Service.Site_Name and Latitude > 0
```

- **Browse the file catalog using**
> glite-catalog-ls -l /
- **Create a new directory in the catalog's root**
> glite-catalog-mkdir /<account>
- **Explore the properties of the newly created directory**
> glite-catalog-stat /<account>
- **Store a local file on a storage element and assign it a LFN (Logical File Name)**
> glite-put ~/myhelloworld.sh lfn:///<account>/myhelloworld.sh

- **Browse the file catalog again**
> glite-catalog-ls -l /<account>
- **Retrieve the file from the storage element**
> glite-get lfn:///<account>/myhelloworld.sh test.sh
- **Remove it from the storage element**
> glite-rm lfn:///<account>/myhelloworld.sh
- **Browse the file catalog**
> glite-catalog-ls -l /<account>
- **Remove the directory from the catalog's root**
> glite-catalog-rmdir /<account>

- <http://www.glite.org/>
- <http://gilda.ct.infn.it/>
- <http://www.eu-egee.org/>
- <http://www.egee.nesc.ac.uk/>
- <http://www.cern.ch/>