EXERCISES:

❖ BACKGROUND

Workflow management systems offer a useful abstraction layer to more easily run distributed computations on e-infrastructures. These systems enable the definition of programs as "workflow components" which can then be combined (linked) to each other to perform more complex operations. Each component potentially can run on a different (type of) infrastructure, for example grid, cloud, a local cluster or a server.

WS-PGRADE is a web-portal (or science gateway) to create and run workflows using the gUse workflow management system, which coordinates the execution of jobs in various types of infrastructures. In these practical exercises the goal is to have minimal hands-on experience with this system by creating and executing simple workflows on the local VM cluster and on the Grid.

❖ BASIC USAGE OF WS-PGRADE:

1. PREAMBLE:

- Download and set up a **new** VM with the WS-Pgrade portal installed.
  - Get the new VM from here
  - Import the image in **VirtualBox**
- Download and install your certificate to the new VM.
  - If you need help, check section 2 in the Quick Start Guide (Lecture 16) tutorial.
  - You won't need to request again for a tutor VO membership, unless you haven't done this yet.
- Start a Firefox session in the new VM and open the WS-Pgrade webpage using this link: http://grid-mooc:8080/liferay-portal-6.1.0/
- Sign in at the top left corner using the following account:
  - Email Address: mooc@grid-mooc.test
  - Password: mooc

2. DOWNLOAD YOUR PROXY TO THE PORTAL: (LECTURE 29)

There are two ways to get your proxy on the portal:
A)

- Open a terminal, create a non-vomsified proxy and upload this to the Myproxy server. To do this, type:
  - **$ voms-proxy-init**
  - **$ myproxy-init -l <username> -c 0 -t 100**

Replace <username> with any name of your preference. You will be asked to enter your GRID pass phrase (the password that was sent to you with your certificate) and a MyProxy pass phrase that will be used in the WS-pgrade portal.

- Sign in to the WS-Pgrade portal and go to Security -> Certificate
- Download the proxy you uploaded on the MyProxy server. Fill in the form with the values:
  - Hostname: px.grid.sara.nl
  - Port: 7512
  - Login: the **<username>** you typed in the previous **myproxy-init** command
  - Password: the MyProxy pass phrase you typed in the previous **myproxy-init** command

- Lifetime: 100 hours
- Press **Download** button
- Press **Associate to VO** button
- Choose **tutor** and press **OK**

From now on you can access grid computing and storage resources from the portal as a member of the tutor VO.

B) Alternatively, you can upload your .p12 certificate to the portal.

- Sign in to the WS-Pgrade portal and go to Security -> Certificate -> Upload
- Download the proxy you uploaded as described in A).

3. CREATE A "HELLO WORLD" WORKFLOW (LECTURE 29)

In this exercise you will do the following steps (see more details below):

a) Create script (will be executed by the job)
b) Create abstract workflow (Graph)
c) Create concrete workflow
d) Configure workflow (specify script to run and inputs for the workflow)

a) Create script

- This simple script will be the same used in the practice for grid and cluster job execution: Download the hello-script.sh in your VM

b) Create abstract workflow (graph)

- Go to **Workflow->Graph**
- Choose **Graph Editor**
- On the **Graph Editor** window
  - Create a Job (**Graph-> New Job**)
  - Configure name and description of the job and graph (right mouse, **Properties**)
  - Save graph (**Graph->Save As**)
- On browser window, press **Refresh**
- The new Graph should show on the list

c) Create concrete workflow

- Go to **Workflow->Create Concrete**
- Select a graph
- Fill in name for the workflow
- Press **Ok**(you should see "workflow has been created successfully")

d) Configure the concrete workflow

- Go to **Workflow->Concrete**
- Find the newly created workflow in list, press **Configure**
- Click on the job
- Go to **Job Executable** tab
- Click on **pbs** as the **Type** (this will make the job run on the VM 'cluster')
- For **"Executable code of binary"**: press **Browse,** upload file with script (hello-script.sh)
- Fill a string into the **"Parameter"** box (this will be given as argument to the script for the "hello" message)

- Click on green “Tick” sign
- Should see
  “Save accepted within the browser. Do not forget to upload the modified configuration on server!”
- Close the **Configure** window (top right)
- Save concrete workflow: **press the diskette icon** (should show success). Press **Ok**
- Click green left arrow

From now on this workflow can be executed

4. RUN THE WORKFLOW ON THE VM CLUSTER (LECTURE 29)

In this exercise you will do the following steps (see more details below):
a. Submit workflow
b. Monitor workflow execution
c. Check the output generated by the workflow

a) Submit workflow
- Go to **Workflow->Concrete**
- Find workflow in list, make sure it is configured
- Press **Submit** button, press **Yes** (should see “The workflow has been submitted”), press **Yes**

b) Monitor workflow execution
- Press **Details** button
- Press **Details** again (will see more information about the individual jobs, in this case only one)
- Press **View finished** or **View all contents** (this will show more details of the selected jobs (finished or all))
- Press **std. Output**, **std. Error** to see messages written by the program

c) Check the output
- The results of the “Hello” job are written to the standard output file, which can be inspected by pressing the button **std. Output** in the detailed view of the jobs

5. RUN THE WORKFLOW ON THE GRID (LECTURE 29)

In this exercise you will do the following steps (see more details below):
a. Reconfigure workflow (using gLite instead of pbs)
b. Run workflow again (see exercise 4)

a) *Reconfigure the existing concrete workflow*
- Go to **Workflow->Concrete**
- Find workflow in list, press **Configure**
- Click on the job
- Choose **gLite** as **Type** (this will make the job run on the Grid)
- Choose **tutor** next to Grid
- Browse for the hello-script.sh and fill a string into the **“Parameter”** box
- Click on green “Tick” sign
- Close the Configure window (top right)

- Save concrete workflow (press the diskette icon). Press **Ok**
- Click green left arrow

b) Submit the workflow like before and check the status of this workflow from time to time. The execution might take a while. You can retrieve the results only when the job shows as **finished** (or **error**).

6. RUN A WORKFLOW WITH INPUT/OUTPUT PORTS

In this exercise you will do the following steps (see more details below):

a) Create script
b) Create abstract workflow (graph)
c) Create concrete (as in exercise 3)
d) Configure concrete
e) Run workflow (as in exercise 4 or 5)
f) Retrieve output files and study their content

a) Create script

This script reads one text input file (inputFile) and receives an argument ($1) to generate an output file (outputFile): Download the hello-script-ports.sh in your VM.

b) Create abstract workflow (graph)

- Go to **Workflow->Graph**
- Start **Graph Editor**
- On the **Graph Editor** window
  - Create a Job (**Graph-> New Job** or icon)
  - Create two ports (Select Job, **Graph->New port**)
  - Save graph (**Graph->Save As**)
- On browser window, press **Refresh**
- The new Graph should show on the list

c) Create concrete workflow

d) *Configure the concrete workflow*

- Go to **Workflow->Concrete**
- Find newly created workflow in list, press **Configure**
- Click on the job
- Choose **pbs or glite** as **Type**
- Go to **Job Executable** tab
- For **"Executable code of binary"**: press **Browse** and upload file containing script: hello-script-ports.sh
- Fill a string into the **"Parameter"** box (this will be given as argument to the script)
- Click on green "Tick" sign

The following will configure the inputs of the job

- Go to **Job I/O** tab
- Click on the input port (configuration form will open)
- Fill box **Input Port's Internal File Name** with *inputFile* (this is the name of the input file in the script)
- Click on the big Pi icon as the **Source of input directed to this port**
- Type a string in the box (this will be copied inside a file named *inputFile* and sent to the job)

The following will configure the outputs of the job

- Go to **Job I/O** tab
- Click on the output port (configuration form will open)
- Fill box **Output Port's Internal File Name** with *outputFile* (this is the name of the outFile file generated by the script)
- Click on green "Tick" sign
- Close the Configure window (top right)
- Save concrete workflow (press the diskette icon)
- Click green left arrow

e) Run workflow

f) Retrieve output files

- Go to **Storage->Local**
- Find workflow of interest and press **get Outputs**
- The archive will contain the files and the logs corresponding to each job (look deep into the directories, there are several layers).

7. CREATE AND RUN WORKFLOW WITH TWO COMPONENTS (LECTURE 29)

This exercise is similar to exercise 6, however here two components should be linked such that the data is processed by one and then the other.
The resulting workflow should be executed on pbs and on the grid.

Steps:

a) Create graph:

Similar to exercise 6. The graph should have two components, each one with one input and one output port. The output of component 1 should be linked to the input port of component 2 (click on the output port and drag to the input port to connect them). Use meaningful names for the ports and components, because these will show on the configuration GUI later.

b) Create concrete

c) *Configure* each job separately, as in exercise 6. Use the same script in both jobs, but different strings in the field **Parameter** (Note that you cannot configure the input port of the second component, because the content of the input file will be passed on automatically from the first component.)

d) *Run and monitor the workflow*

e) Retrieve output files

8. SPLIT AND MERGE WORKFLOW (LECTURE 29)

In this exercise you will create a workflow that will split a file into smaller chunks, run jobs for each of the chunks, then merge the results again into a file. This can be achieved using special types of input and output ports (**Generator** and **Collector**).

As an example consider the scripts: splitLines.sh, task.sh, merge.sh and input: uploadInput.txt

Run the example both on pbs and Grid.

9. PARAMETER SWEEP WORKFLOW (LECTURE 29)

In this exercise you will run a workflow for various values in one of the input ports (parameter sweep). This will cause the execution of one job for each of the input values in the list.

The same scripts and the same Graph will be used as for exercise 8. You need to create a new concrete workflow, then configure it to run for a list of files, instead of a single one.
This is done by configuring the input port with a file with special name paramInputs.zip.

After succeeding, change the parameters, make a new archive and run the workflow again.

Run the example both on pbs and Grid.

10. GRID STORAGE (LECTURE 29)

Repeat Exercise 8, this time fetching the input file (uploadInput.txt) from the Grid storage:

- Go to Storage -> LFC -> Get LFC Hosts -> List LFC Host content -> Upload the file uploadInput.txt
- Go to Concrete -> select the workflow 'splitmerge' -> Configure
  - For the 'Split' job -> set Type to 'glite' to run on the Grid
  - Job I/O: splitInput -> Source of input directed to this port: Remote -> lfn:/grid/tutor/<path to your file>
  - Run the workflow

ADDITIONAL EXERCISES:

- Workflows

Choose two grid workflow management systems (WfMS) and find information about them on websites, papers, or personal communication.
As examples of possible systems to choose from, you may use the list contained in Lecture 27. However, there are many others, so feel free to study other that speak more to your own interest and application case.

Based on the study of this documentation, try to briefly answer the following questions for these two WfMS:

1) Can the WfMS submit jobs using gLite middleware? Which other middleware does the system support?

2) How are workflows described to this WfMS? Is there a language or graphical interface for the developer?

3) How are the workflows executed in this system? Does the user start manually? Or are the workflows started automatically somehow?

4) How is the monitoring of workflow execution done? Is there a graphical interface? Can the user see the intermediate status and the results?

5) How does the WfMS handle errors? Does it retry failed jobs? Does it abort the workflow execution when errors occur? What type of information is shown by the WfMS to the user about the errors that occurred?

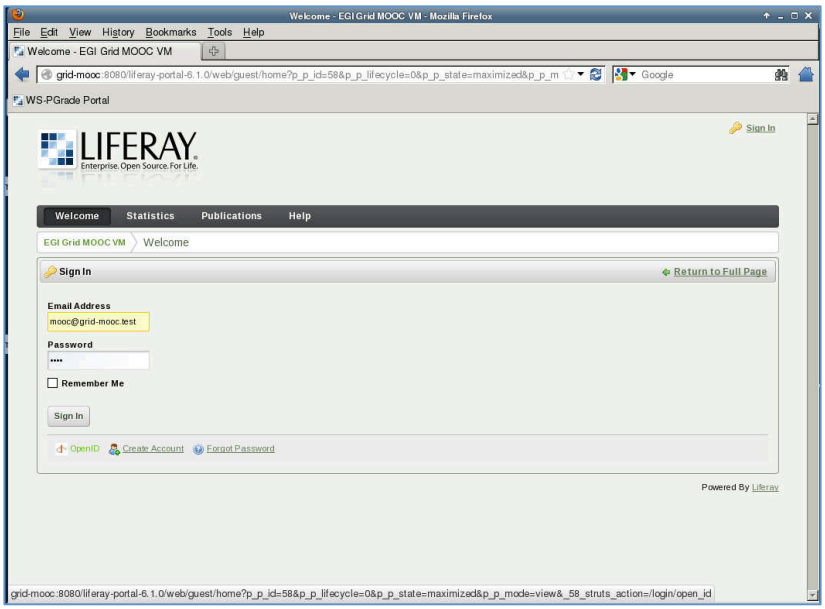
- Science Gateways

Search the web for science gateways in your own area of interest. For example, in astronomy or for grid computing. Suggestion: use the pointers in lecture 30. Then answer the following questions for one of them:

1) What can one do on this gateway? For example, access data, run predefined applications, etc.
2) Who can access the gateway? Is it open or for a closed community?
3) What do you need to bring into the gateway to be able to use it? For example, own data, grid certificate, payment, etc.
4) Which team or organization maintains the gateway?
5) Would this science gateway actually help your research in some way? Why (not)?

SOLUTIONS:

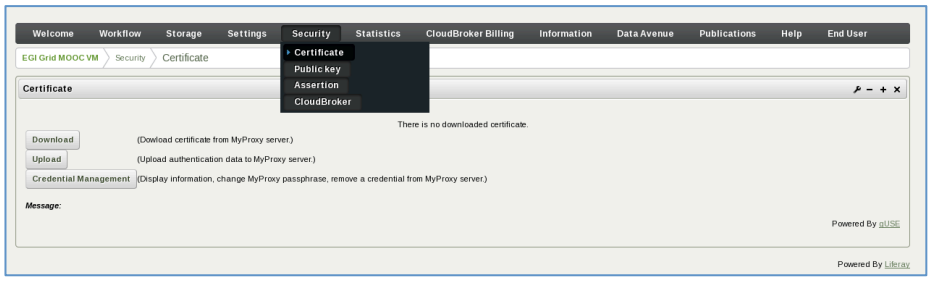
1. PREAMBLE:



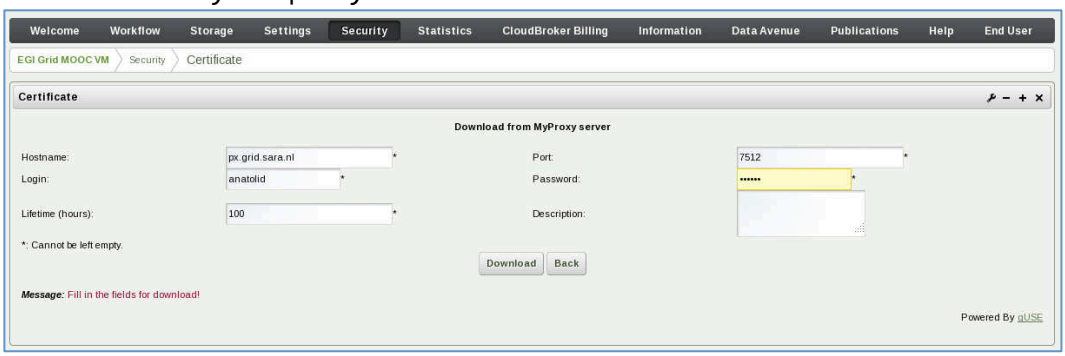
2. DOWNLOAD YOUR PROXY TO THE PORTAL:

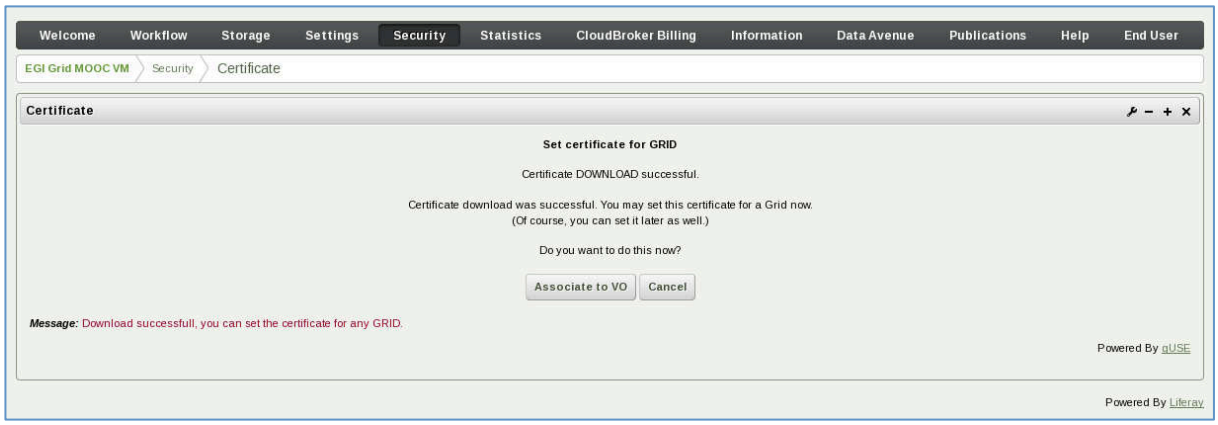
A)
Open a Terminal in your VM and type:

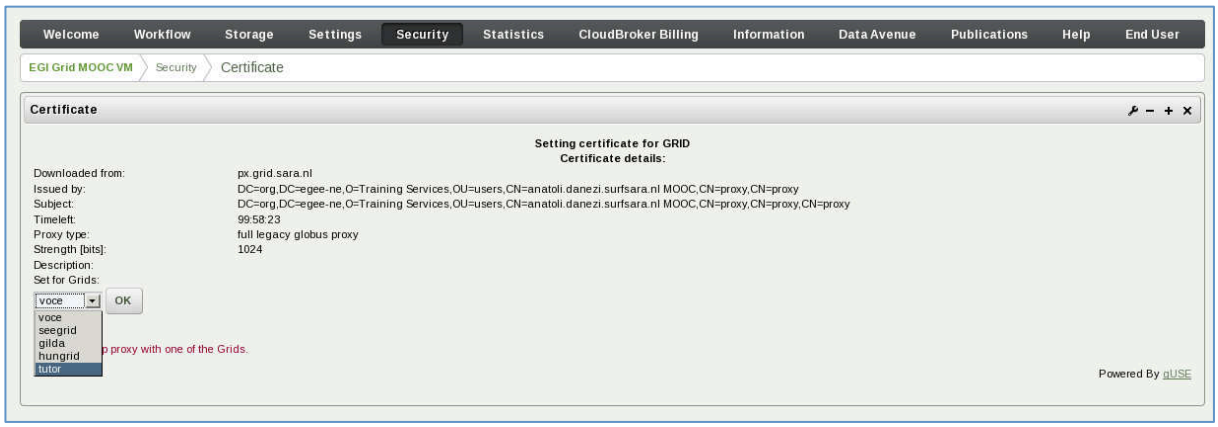
- **$ voms-proxy-init**
- **$ myproxy-init -l <username> -c 0 -t 100**
- Sign into the portal here: http://grid-mooc:8080/liferay-portal-6.1.0/
- Go to Security -> Certificate

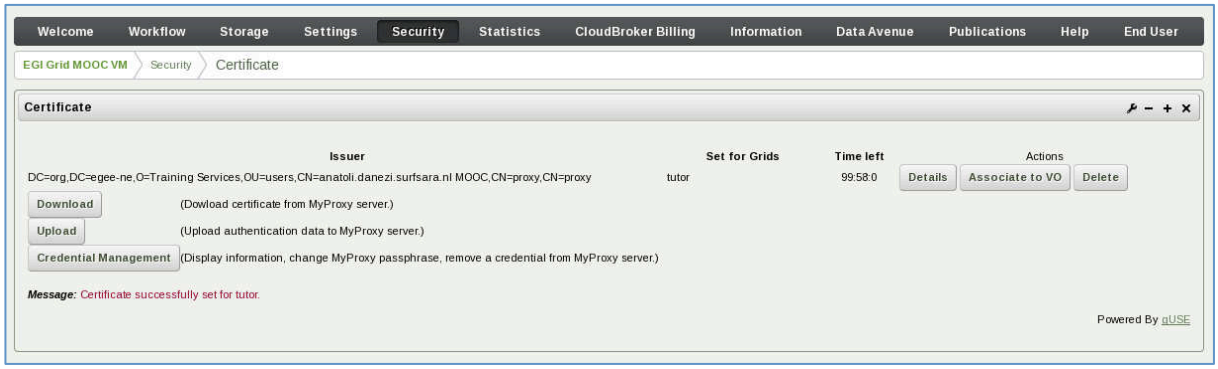


- Download your proxy:



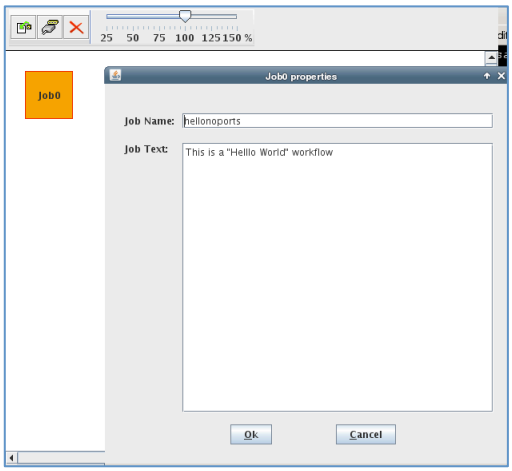


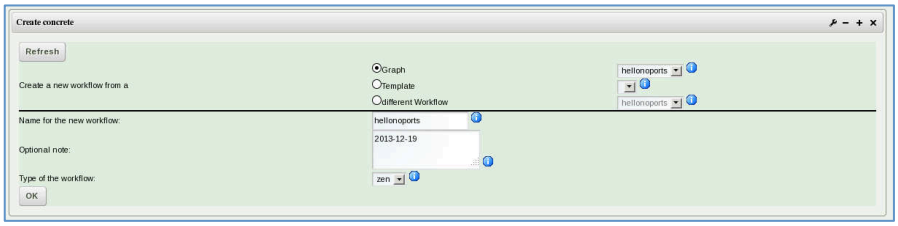


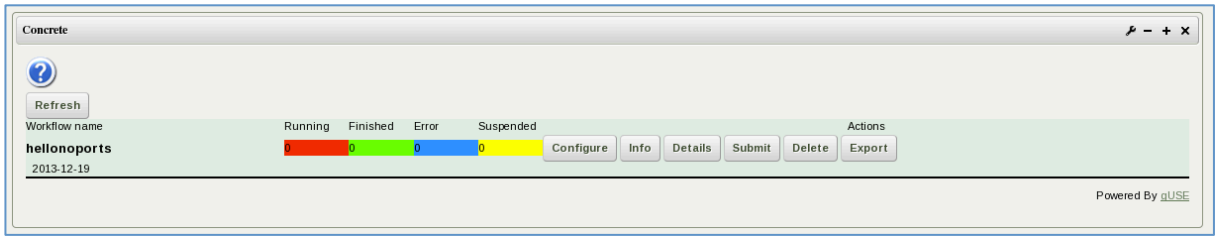


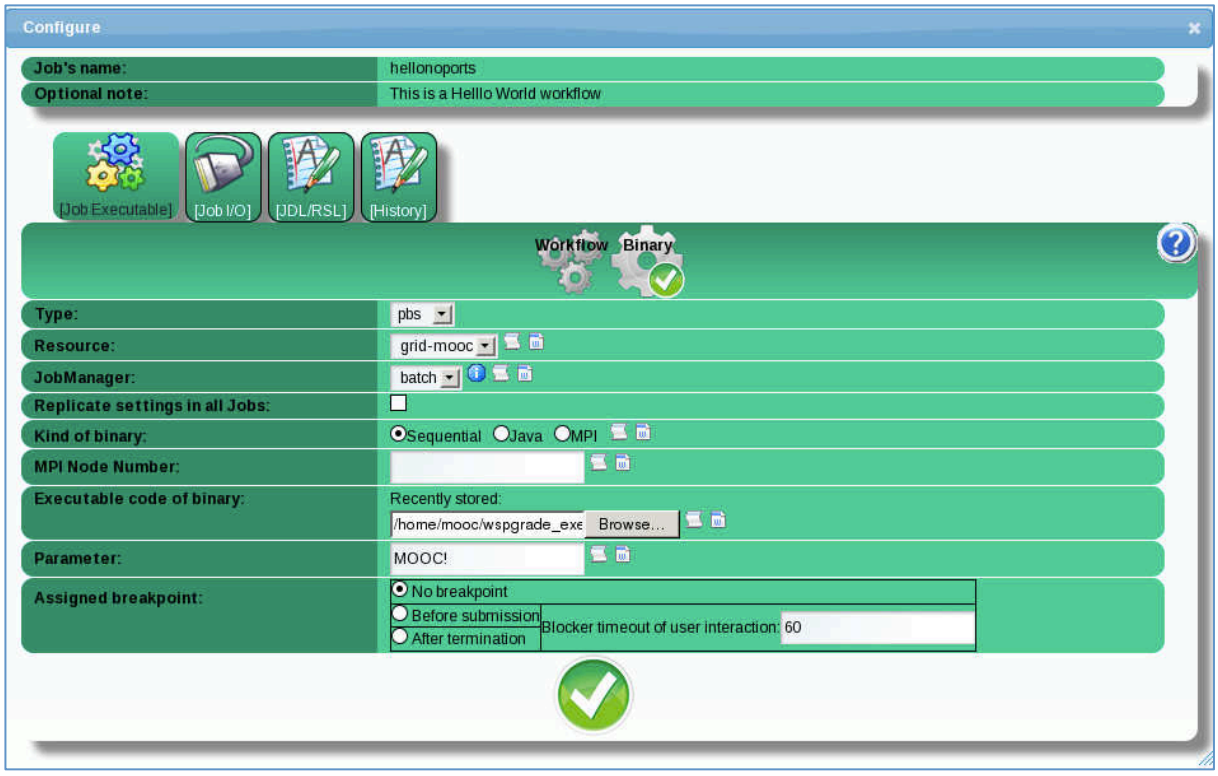
3. CREATE A "HELLO WORLD" WORKFLOW

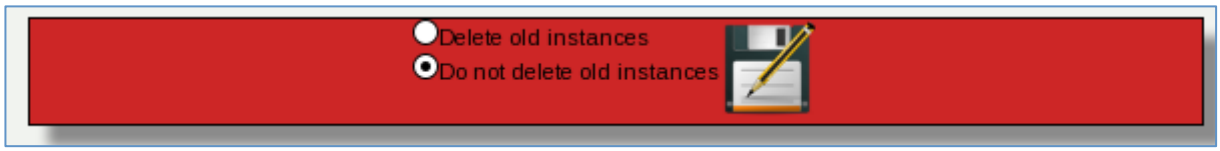
- Download the hello-script.sh in your VM


25 50 75 100 125 150 %
Job0
Job0 properties
Job Name:
hellonoports
Job Text:
This is a "Helllo World" workflow
Ok
Cancel

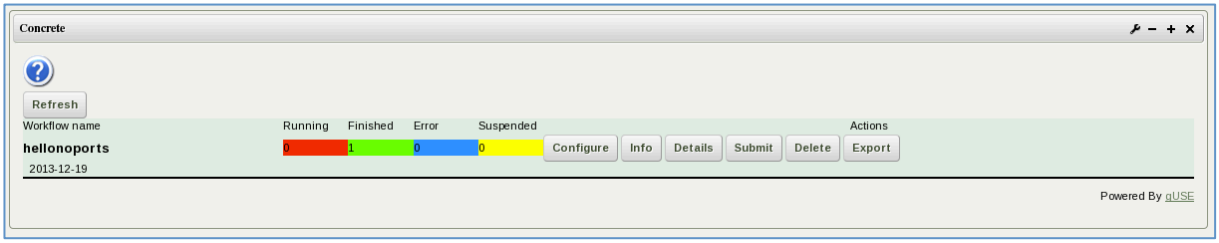

Create concrete
Refresh
Create a new workflow from a
Graph
Template
different Workflow
hellonoports
Name for the new workflow:
hellonoports
Optional note:
2013-12-19
Type of the workflow:
zen
OK

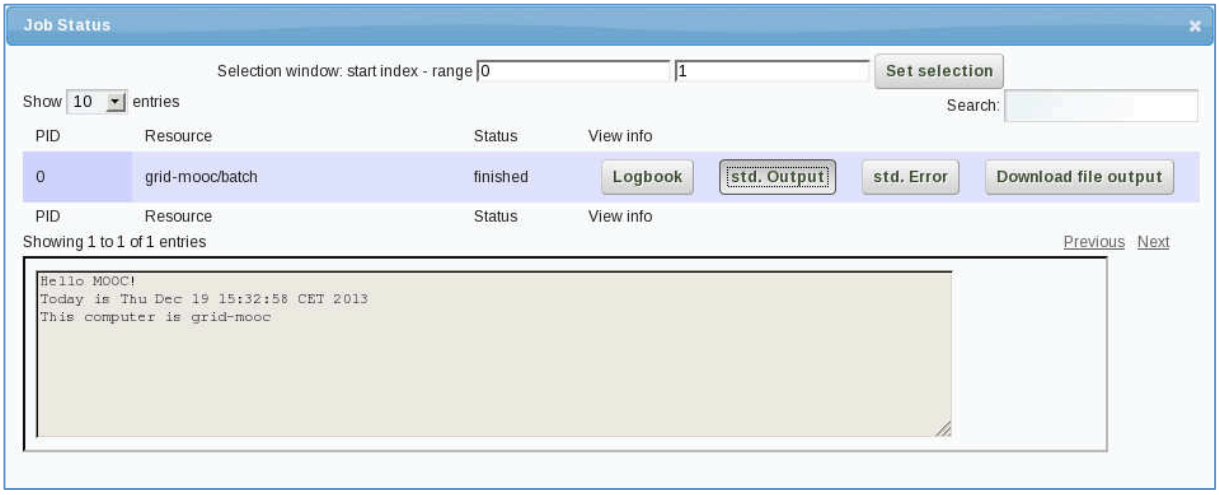

Concrete
Refresh
Workflow name
Running
Finished
Error
Suspended
Actions
hellonoports
2013-12-19
0
0
0
0
Configure
Info
Details
Submit
Delete
Export
Powered By gUSE


Configure
Job's name:
hellonoports
Optional note:
This is a Helllo World workflow
[Job Executable]
[Job I/O]
[JDL/RSL]
[History]
Workflow
Binary
Type:
pbs
Resource:
grid-mooc
JobManager:
batch
Replicate settings in all Jobs:
Kind of binary:
Sequential
Java
MPI
MPI Node Number:
Executable code of binary:
Recently stored:
/home/mooc/wspgrade_exe
Browse...
Parameter:
MOOC!
Assigned breakpoint:
No breakpoint
Before submission
After termination
Blocker timeout of user interaction:
60

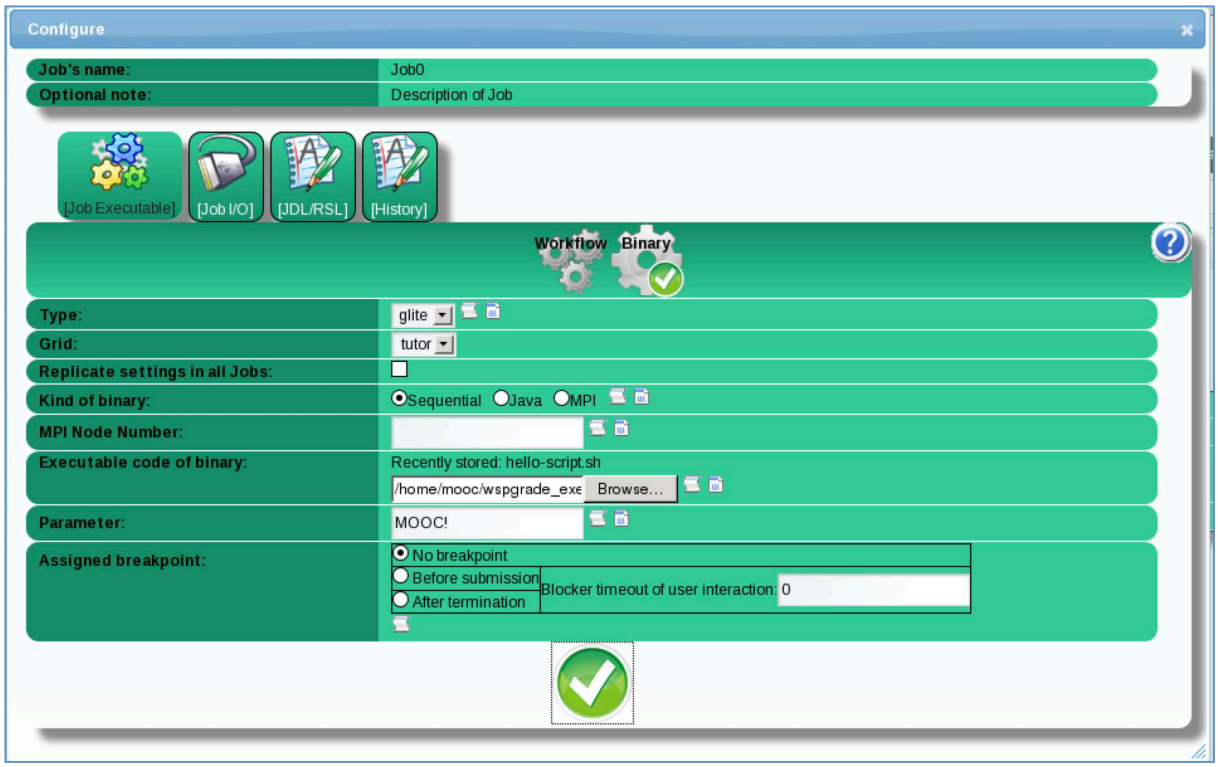


4. RUN THE WORKFLOW ON THE VM CLUSTER

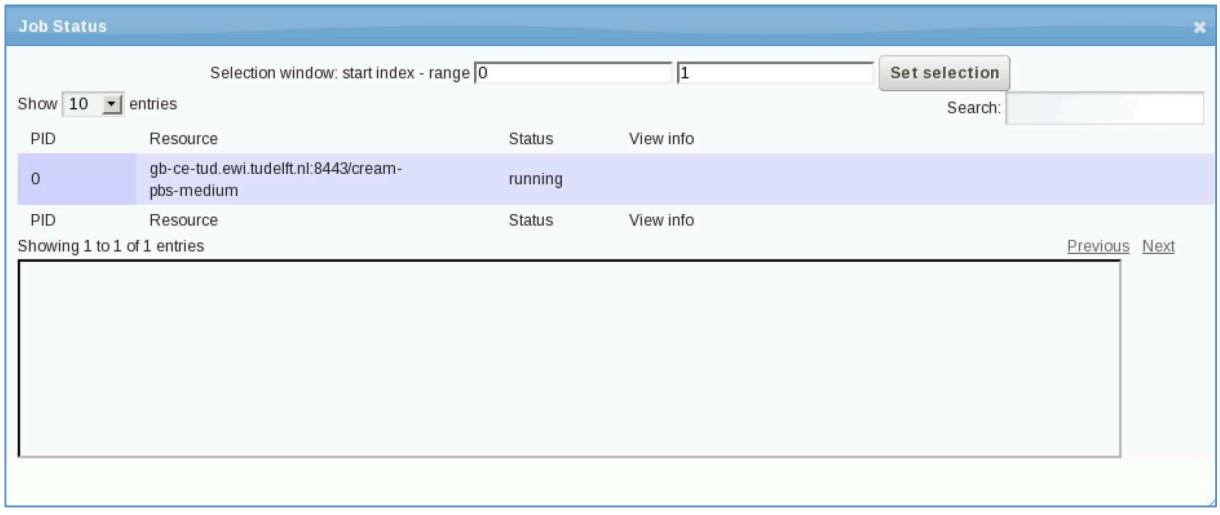


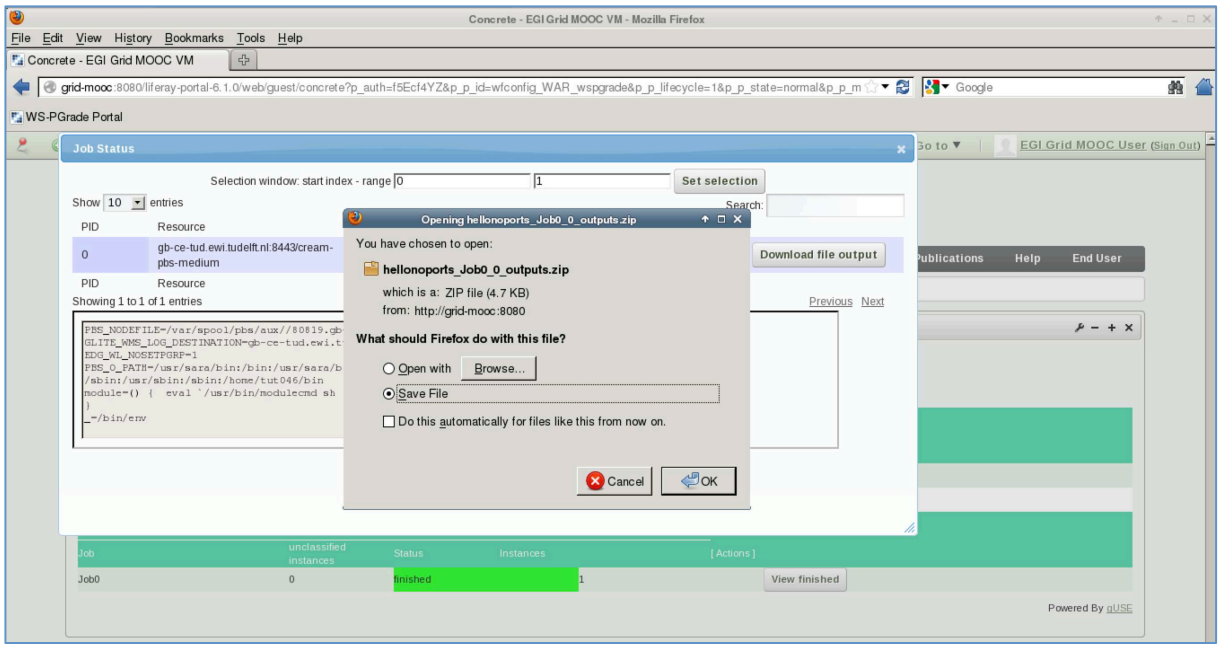


5. RUN THE WORKFLOW ON THE GRID


Configure
Job's name:
Job0
Optional note:
Description of Job
[Job Executable]
[Job I/O]
[JDL/RSL]
[History]
Workflow
Binary
Type:
glite
Grid:
tutor
Replicate settings in all Jobs:
Kind of binary:
Sequential
Java
MPI
MPI Node Number:
Executable code of binary:
Recently stored: hello-script.sh
/home/mooc/wspgrade_exe
Browse...
Parameter:
MOOC!
Assigned breakpoint:
No breakpoint
Before submission
After termination
Blocker timeout of user interaction:
0

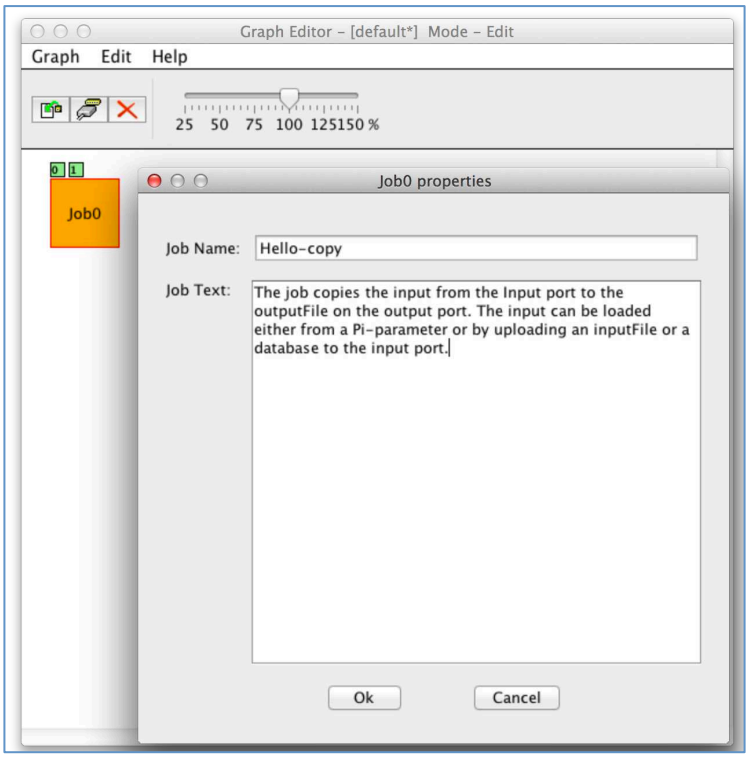

Job Status
Selection window: start index - range
0
1
Set selection
Show
10
entries
Search:
PID
Resource
0
tutor WMS
PID
Resource
Showing 1 to 1 of 1 entries
Previous
Next
finished
Details
Delete
Cost of Instance
running
Details
Suspend
Cost of Instance
Status
Instances
[Actions]
running
1
View running

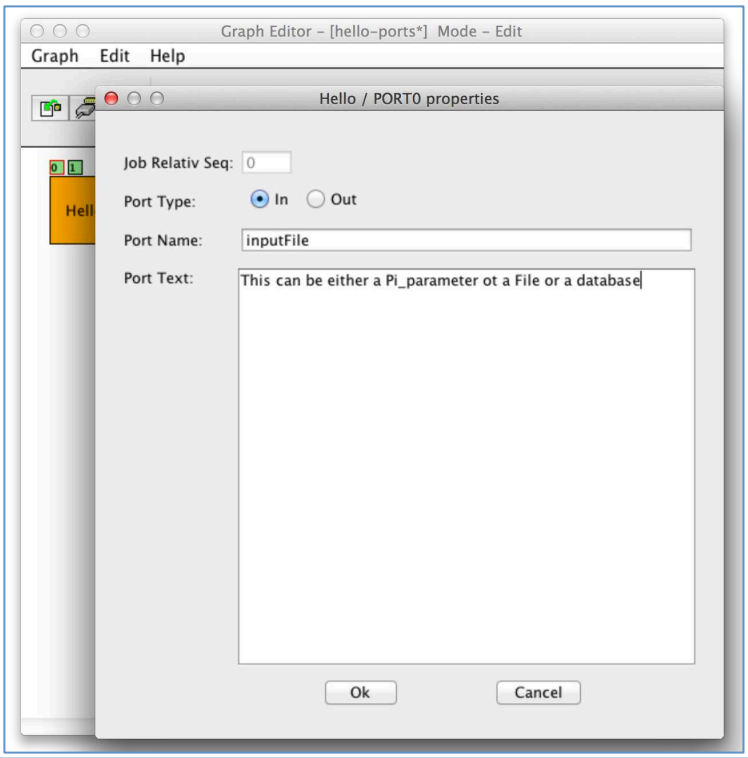

Job Status
Selection window: start index - range
0
1
Set selection
Show
10
entries
Search:
PID
Resource
Status
View info
0
gb-ce-tud.ewi.tudelft.nl:8443/cream-pbs-medium
running
PID
Resource
Status
View info
Showing 1 to 1 of 1 entries
Previous
Next

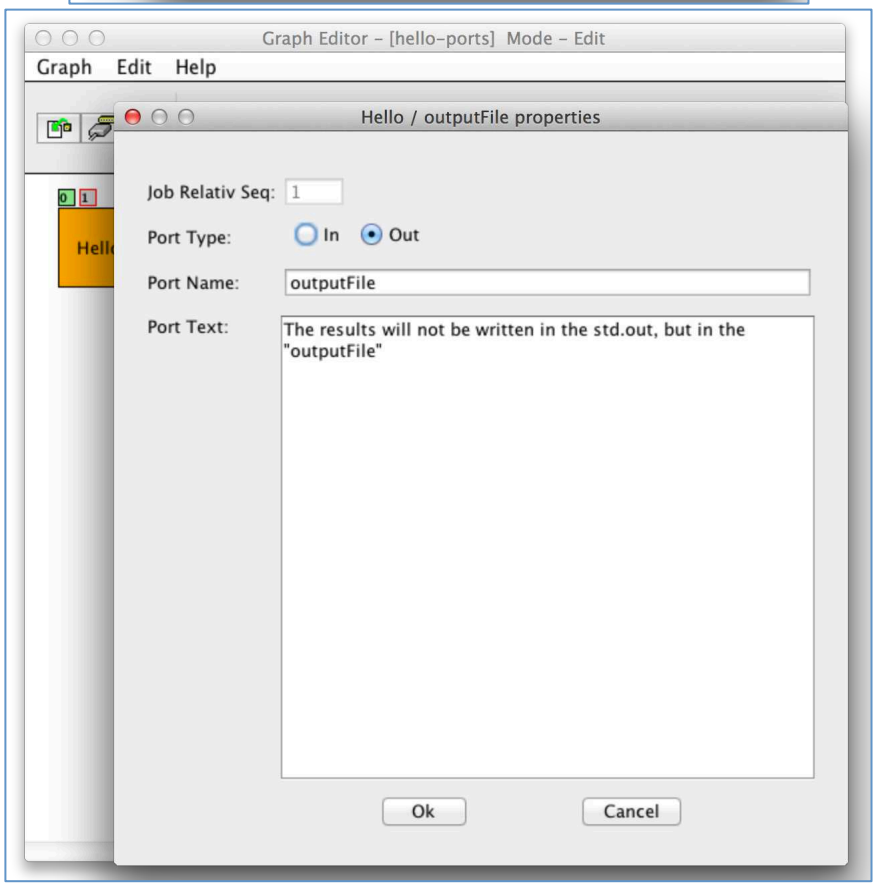


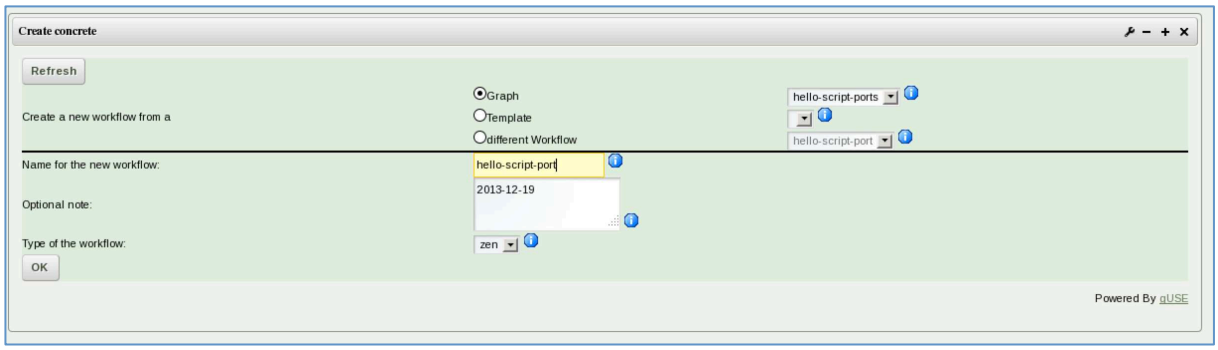
6. RUN A WORKFLOW WITH INPUT/OUTPUT PORTS

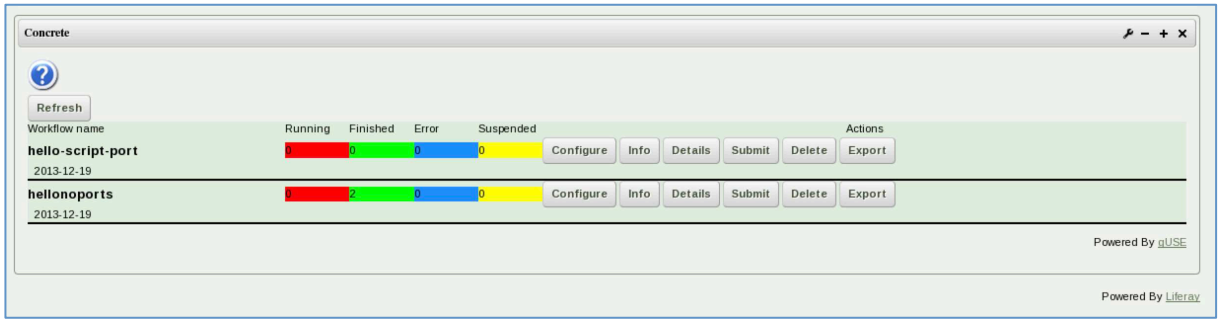
Download the hello-script-ports.sh in your VM

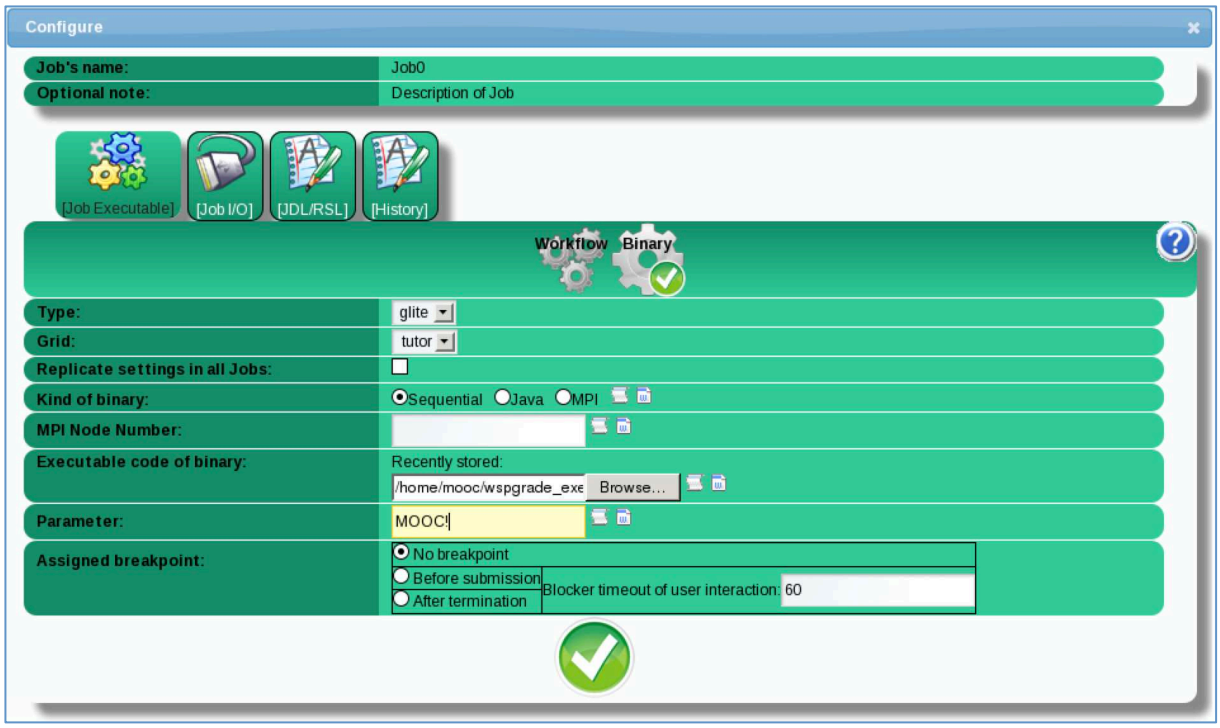


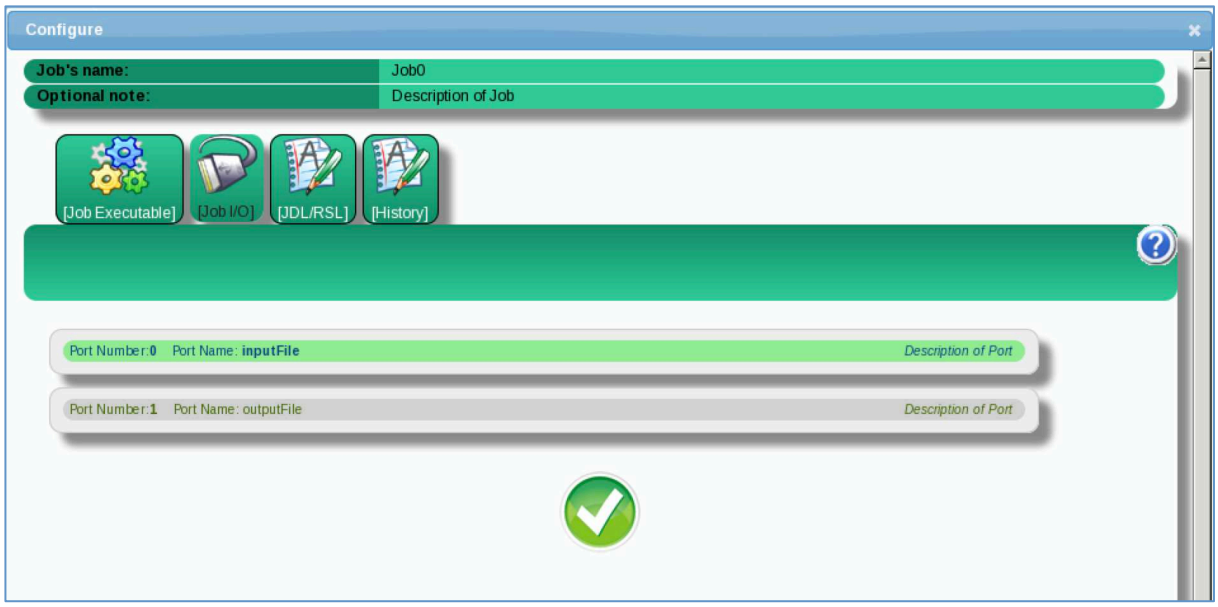

Graph Editor - [hello-ports*] Mode - Edit
Graph Edit Help
Hello / PORT0 properties
Job Relativ Seq: 0
Port Type: In Out
Port Name: inputFile
Port Text: This can be either a Pi_parameter ot a File or a database
Ok
Cancel

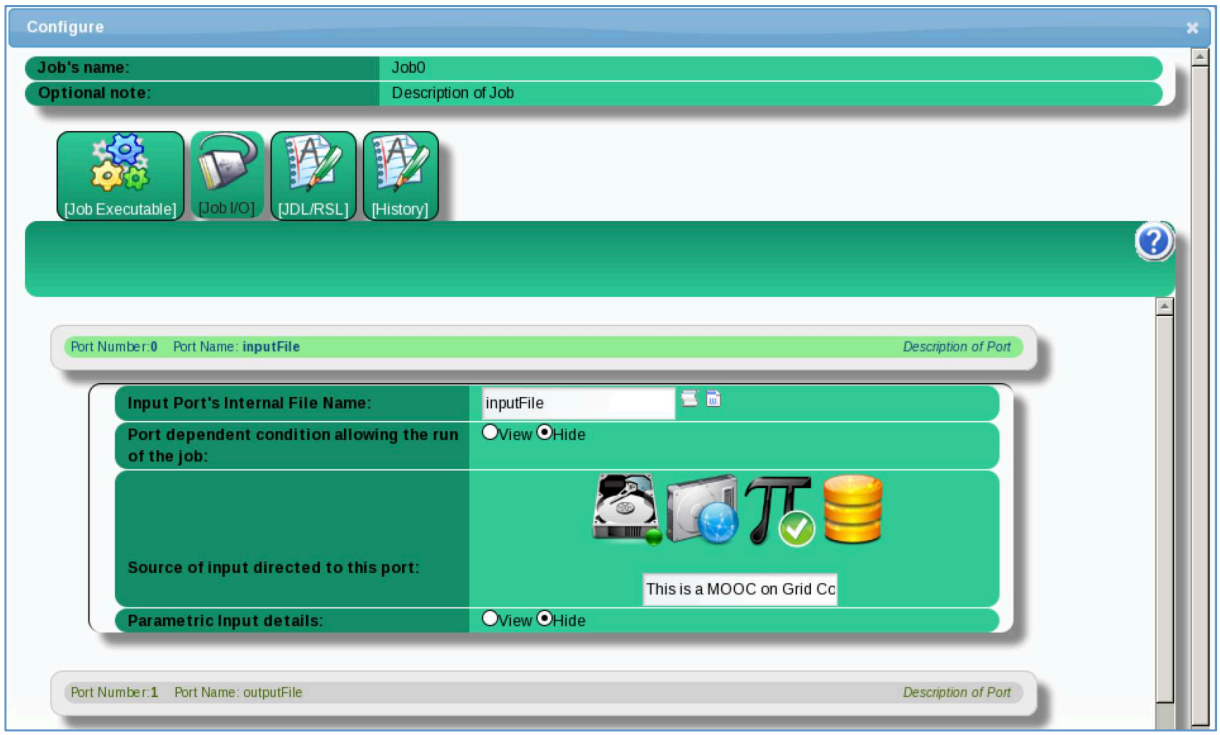

Graph Editor - [hello-ports] Mode - Edit
Graph Edit Help
Hello / outputFile properties
Job Relativ Seq: 1
Port Type: In Out
Port Name: outputFile
Port Text: The results will not be written in the std.out, but in the "outputFile"
Ok
Cancel

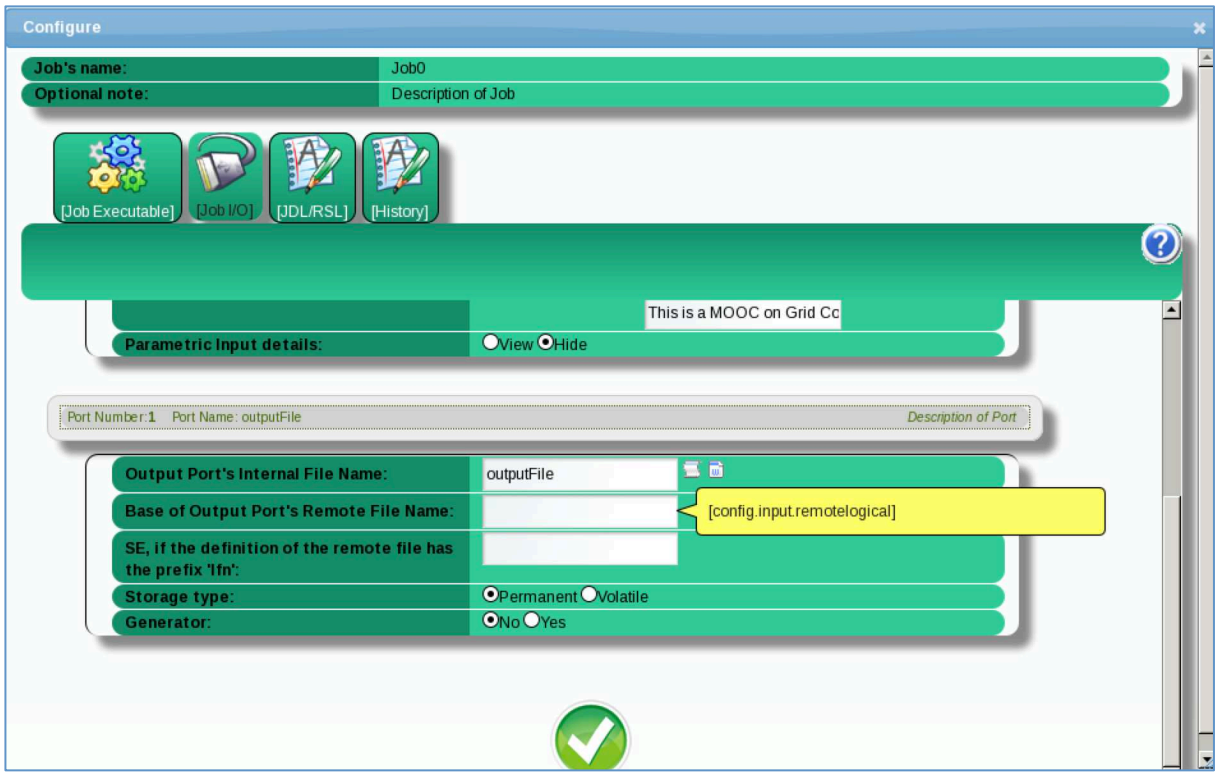

Create concrete
Refresh
Create a new workflow from a
Graph
Template
different Workflow
hello-script-ports
hello-script-port
Name for the new workflow:
hello-script-port
Optional note:
2013-12-19
Type of the workflow:
zen
OK
Powered By gUSE

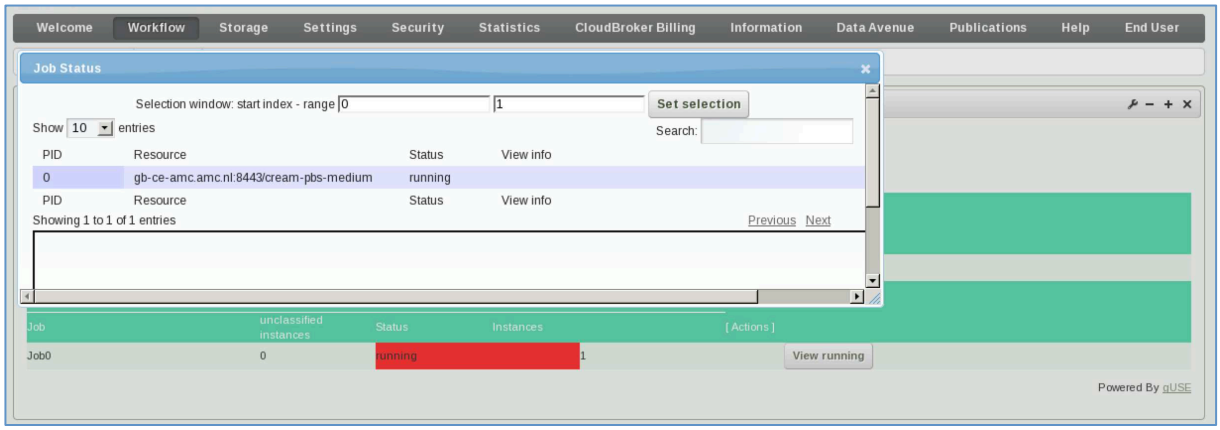

Concrete
Refresh
Workflow name
Running
Finished
Error
Suspended
Actions
hello-script-port
2013-12-19
Configure
Info
Details
Submit
Delete
Export
hellonoports
2013-12-19
Powered By gUSE
Powered By Liferay


Configure
Job's name:
Job0
Optional note:
Description of Job
[Job Executable]
[Job I/O]
[JDL/RSL]
[History]
Workflow
Binary
Type:
glite
Grid:
tutor
Replicate settings in all Jobs:
Kind of binary:
Sequential
Java
MPI
MPI Node Number:
Executable code of binary:
Recently stored:
/home/mooc/wspgrade_exe
Browse...
Parameter:
MOOC!
Assigned breakpoint:
No breakpoint
Before submission
After termination
Blocker timeout of user interaction:
60


Configure
Job's name:
Job0
Optional note:
Description of Job
[Job Executable]
[Job I/O]
[JDL/RSL]
[History]
Port Number:0 Port Name: inputFile
Description of Port
Port Number:1 Port Name: outputFile
Description of Port

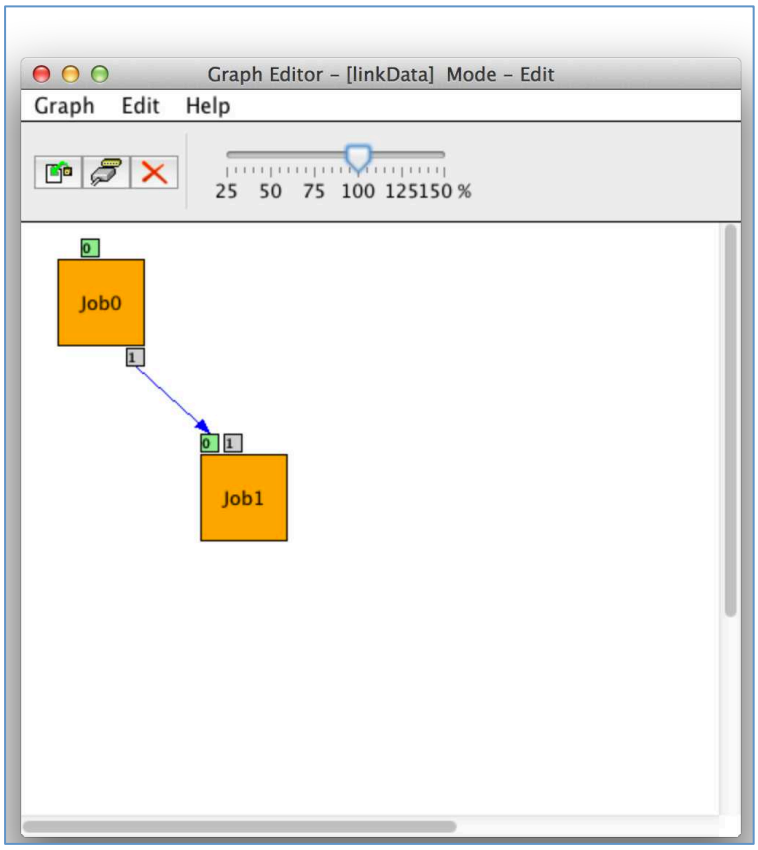

Configure
Job's name:
Job0
Optional note:
Description of Job
[Job Executable]
[Job I/O]
[JDL/RSL]
[History]
Port Number:0 Port Name: inputFile
Description of Port
Input Port's Internal File Name:
inputFile
Port dependent condition allowing the run of the job:
View Hide
Source of input directed to this port:
This is a MOOC on Grid Cc
Parametric Input details:
View Hide
Port Number:1 Port Name: outputFile
Description of Port

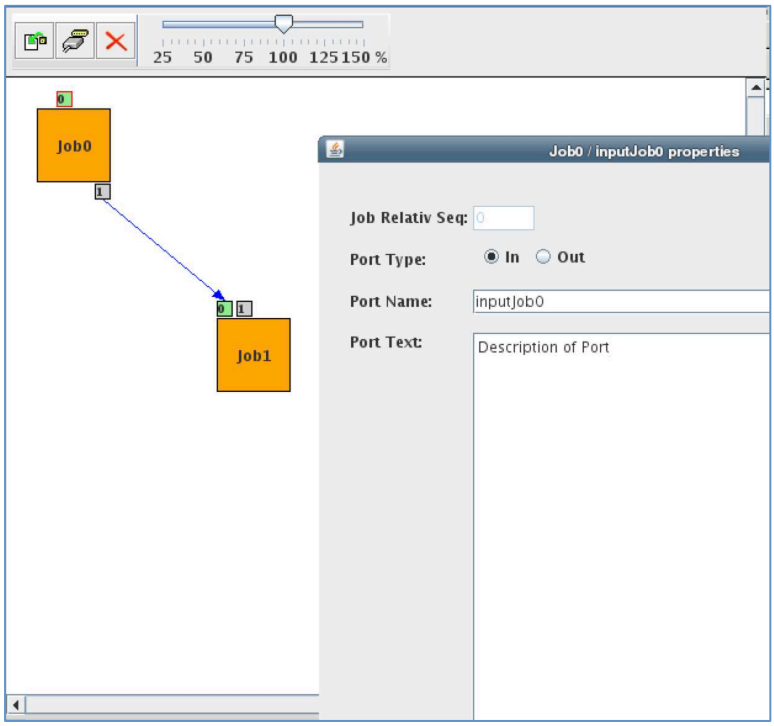


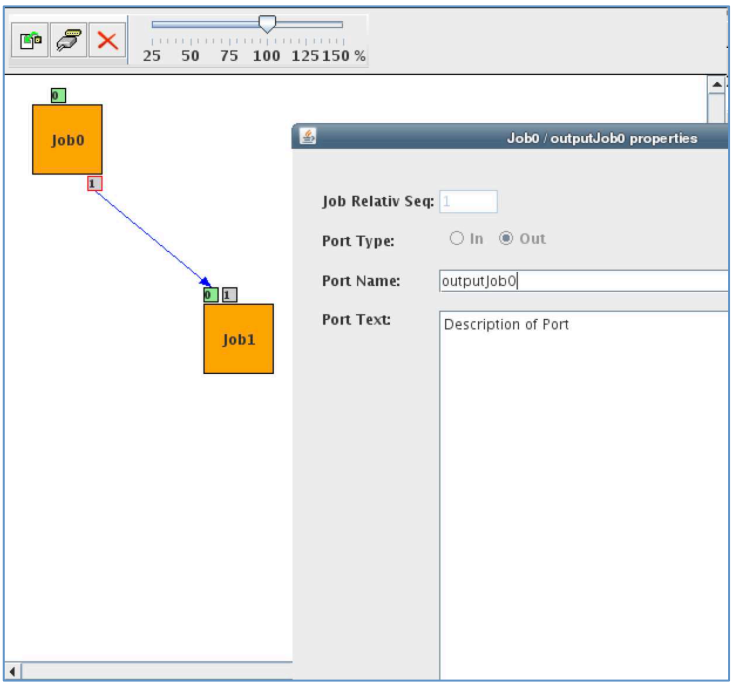


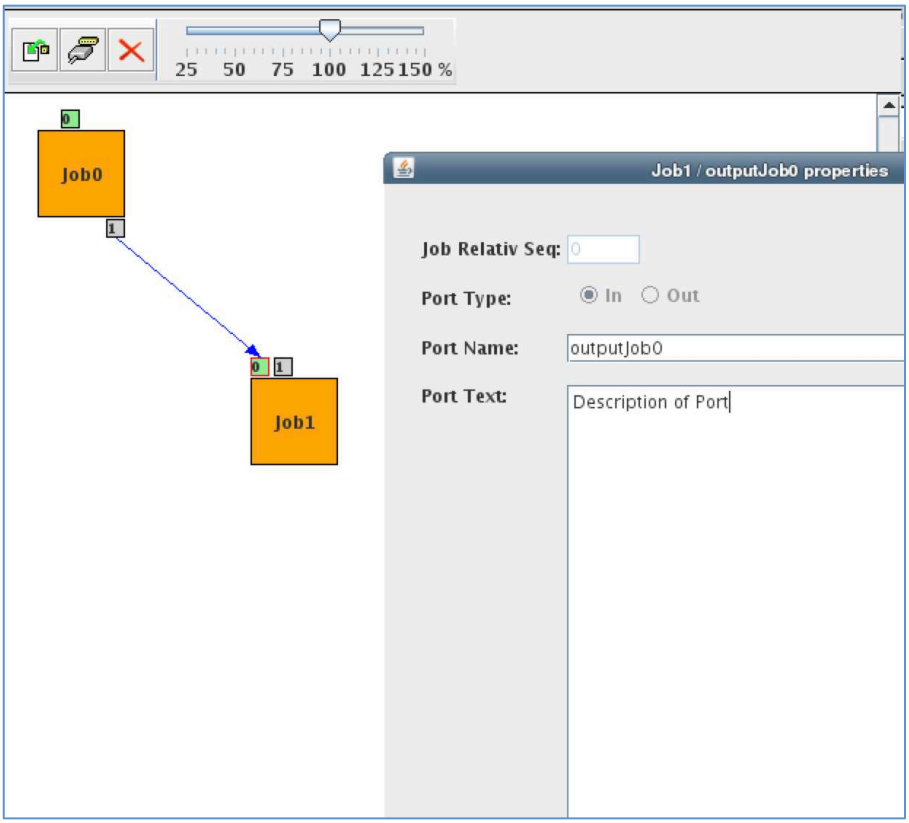
7. CREATE AND RUN WORKFLOW WITH TWO COMPONENTS

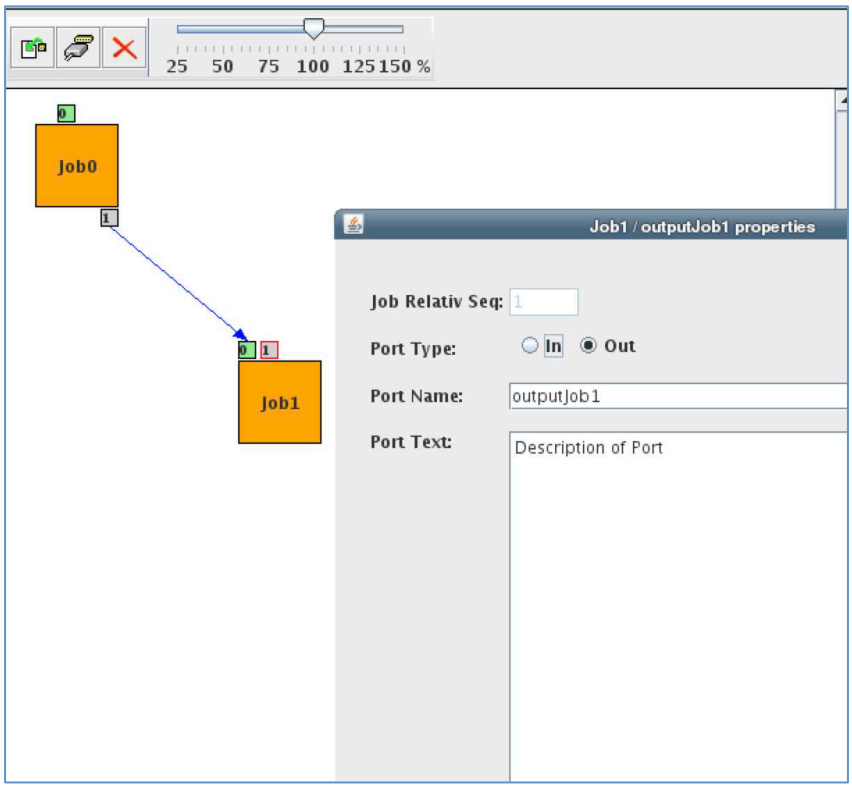
Download linkedData1.sh, linkedData2.sh

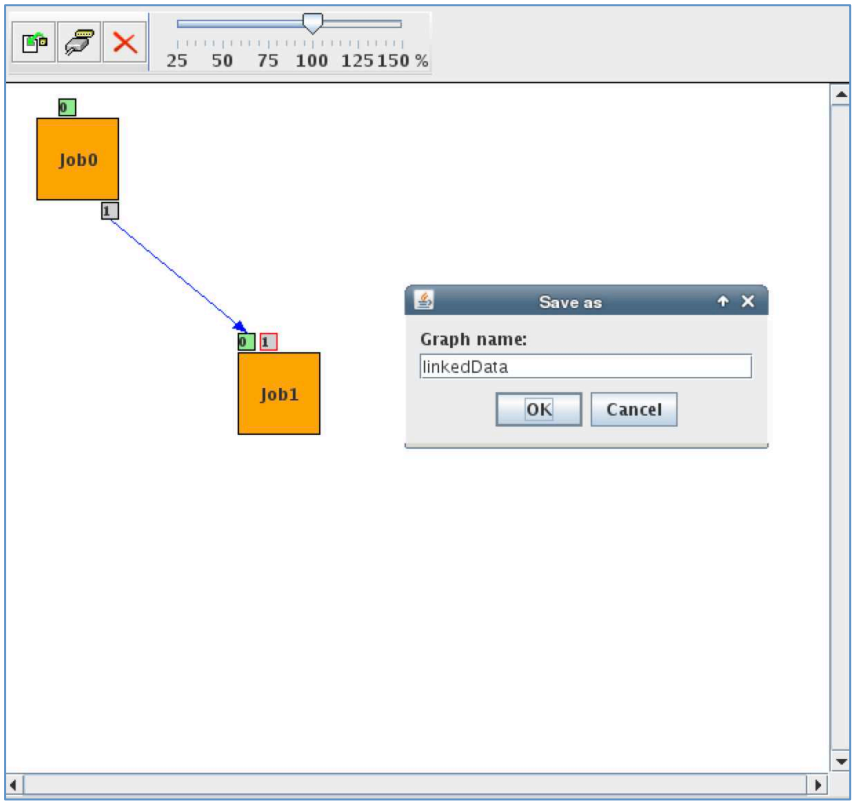

Graph Editor – [linkData] Mode – Edit
Graph
Edit
Help
25 50 75 100 125150 %
0
Job0
1
0 1
Job1

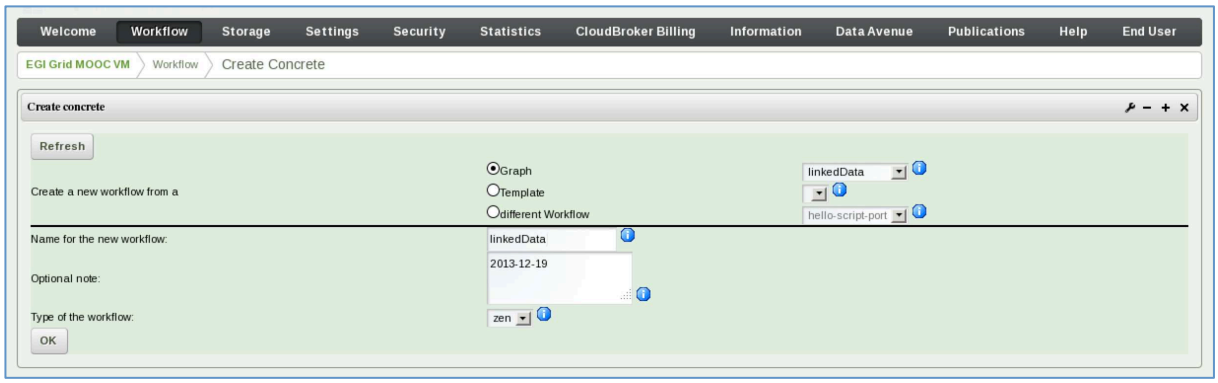

25 50 75 100 125150 %
0
Job0
1
0 1
Job1
Job0 / inputJob0 properties
Job Relativ Seq:
0
Port Type:
In
Out
Port Name:
inputJob0
Port Text:
Description of Port

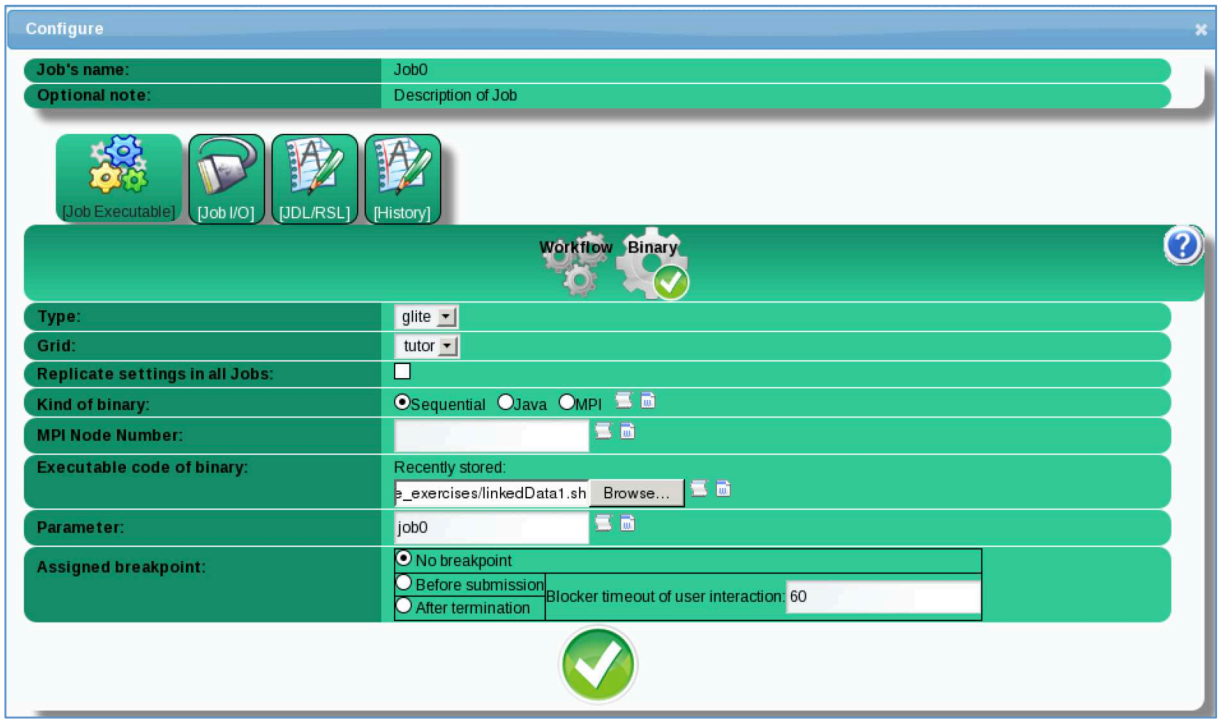

25 50 75 100 125150 %
Job0
Job1
Job0 / outputJob0 properties
Job Relativ Seq:
Port Type:
In
Out
Port Name:
outputJob0
Port Text:
Description of Port

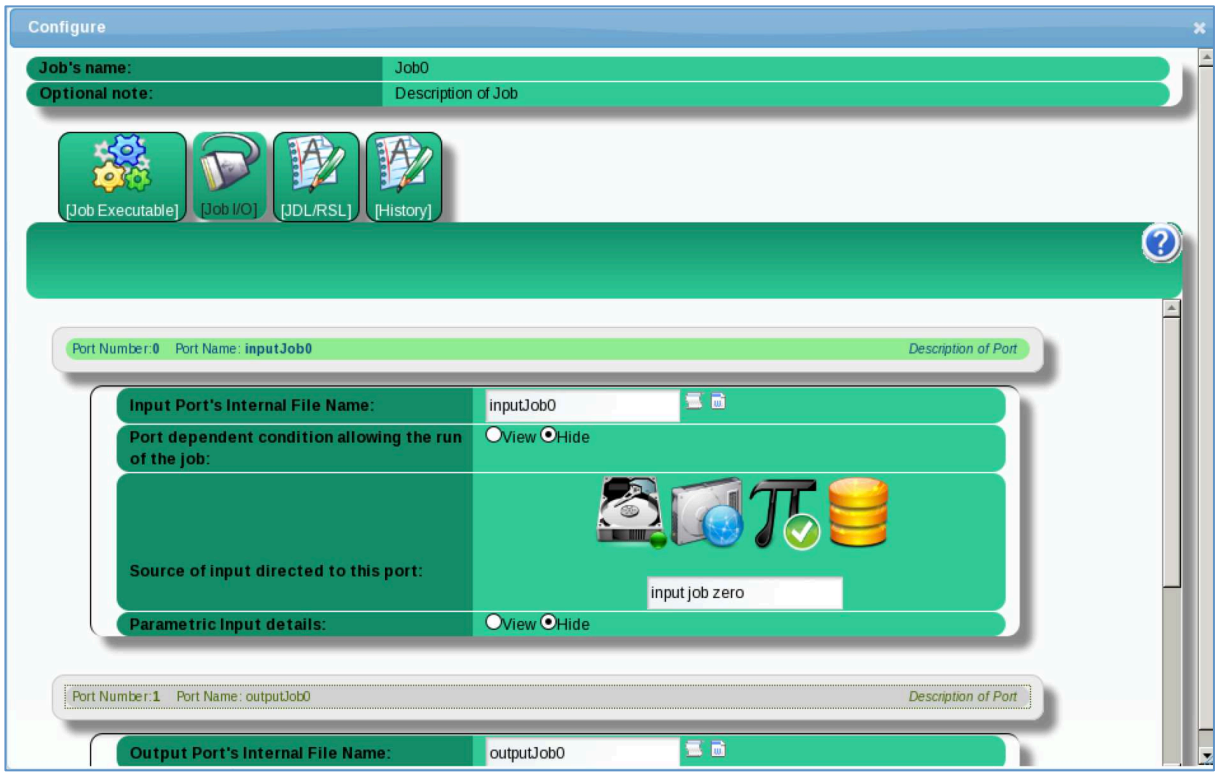

25 50 75 100 125150 %
Job0
Job1
Job1 / outputJob0 properties
Job Relativ Seq:
Port Type:
In
Out
Port Name:
outputJob0
Port Text:
Description of Port

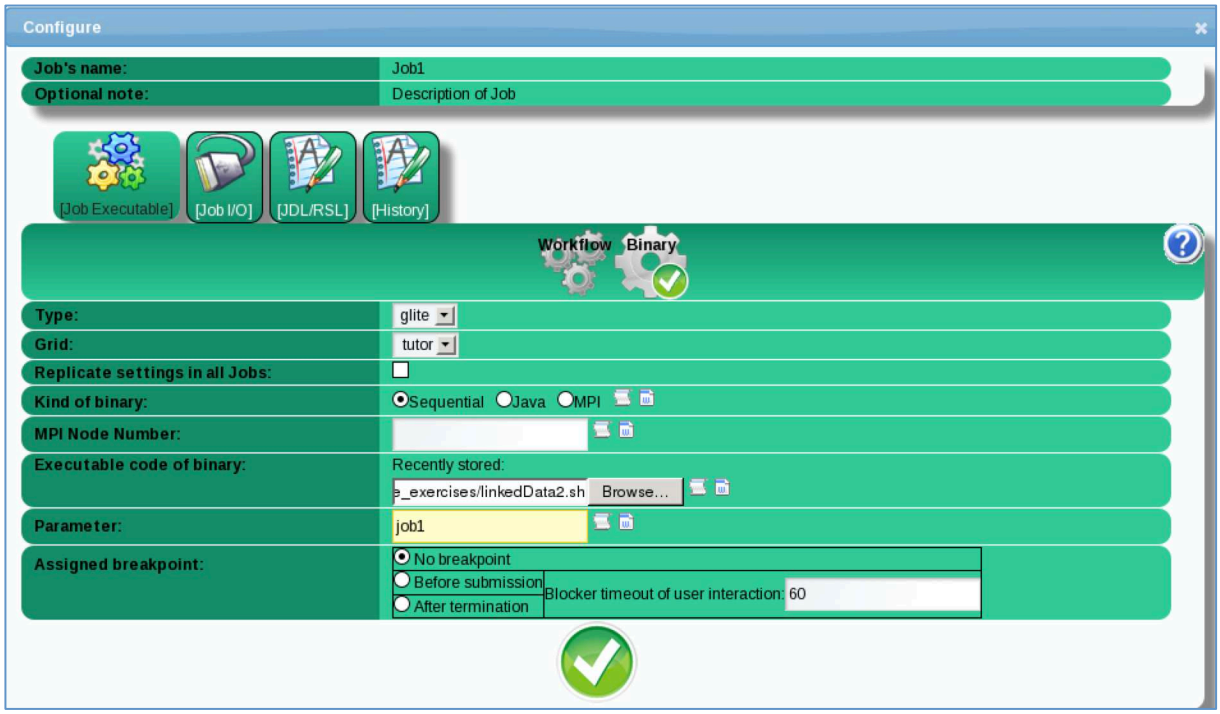

25 50 75 100 125 150 %
Job0
Job1
Job1 / outputJob1 properties
Job Relativ Seq:
1
Port Type:
In
Out
Port Name:
outputJob1
Port Text:
Description of Port

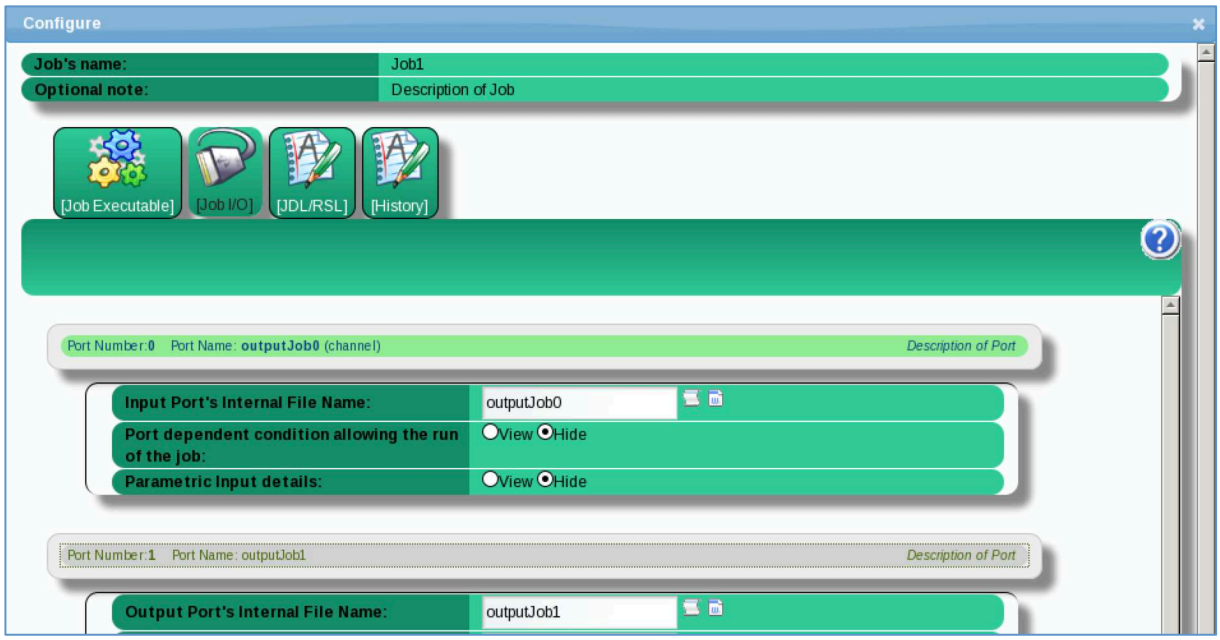

25 50 75 100 125 150 %
Job0
Job1
Save as
Graph name:
linkedData
OK
Cancel

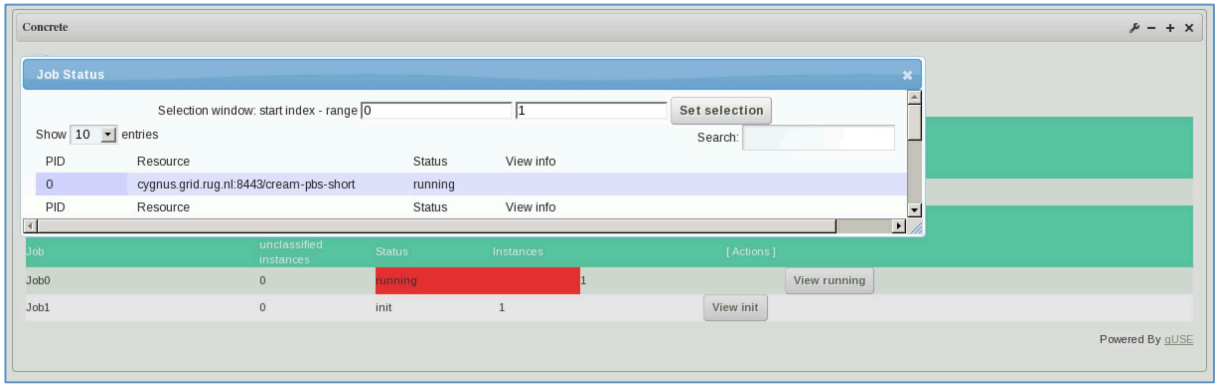

Welcome
Workflow
Storage
Settings
Security
Statistics
CloudBroker Billing
Information
Data Avenue
Publications
Help
End User
EGI Grid MOOC VM
Workflow
Create Concrete
Create concrete
Refresh
Create a new workflow from a
Graph
Template
different Workflow
linkedData
hello-script-port
Name for the new workflow:
linkedData
Optional note:
2013-12-19
Type of the workflow:
zen
OK

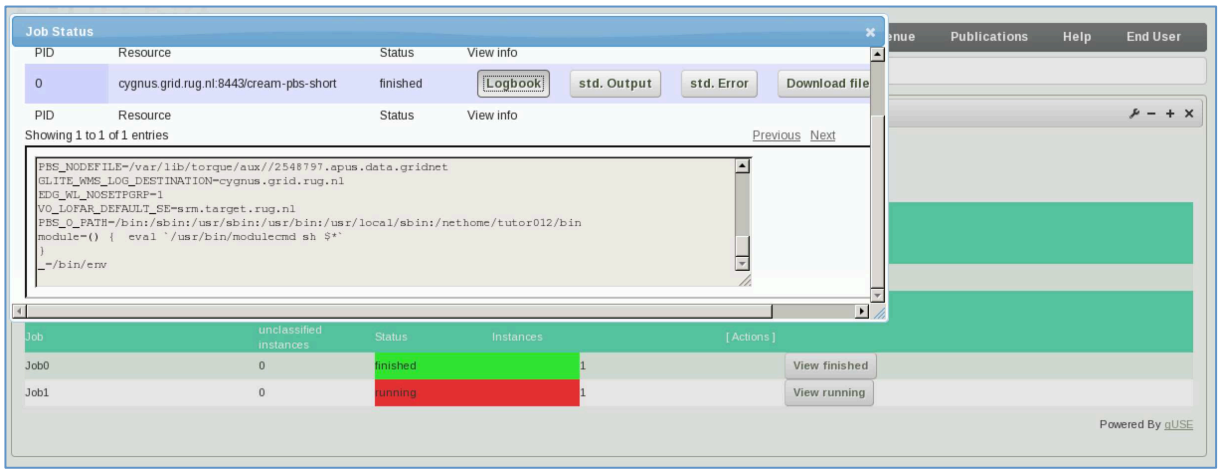

Configure
Job's name:
Job0
Optional note:
Description of Job
[Job Executable]
[Job I/O]
[JDL/RSL]
[History]
Workflow
Binary
Type:
glite
Grid:
tutor
Replicate settings in all Jobs:
Kind of binary:
Sequential
Java
MPI
MPI Node Number:
Executable code of binary:
Recently stored:
Browse...
Parameter:
job0
Assigned breakpoint:
No breakpoint
Before submission
After termination
Blocker timeout of user interaction:
60


Configure
Job's name:
Job0
Optional note:
Description of Job
[Job Executable]
[Job I/O]
[JDL/RSL]
[History]
Port Number:0 Port Name: inputJob0
Description of Port
Input Port's Internal File Name:
inputJob0
Port dependent condition allowing the run of the job:
View Hide
Source of input directed to this port:
input job zero
Parametric Input details:
View Hide
Port Number:1 Port Name: outputJob0
Description of Port
Output Port's Internal File Name:
outputJob0


Configure
Job's name:
Job1
Optional note:
Description of Job
[Job Executable]
[Job I/O]
[JDL/RSL]
[History]
Workflow
Binary
Type:
glite
Grid:
tutor
Replicate settings in all Jobs:
Kind of binary:
Sequential Java MPI
MPI Node Number:
Executable code of binary:
Recently stored:
s_exercises/linkedData2.sh
Browse...
Parameter:
job1
Assigned breakpoint:
No breakpoint
Before submission
After termination
Blocker timeout of user interaction:
60


Configure
Job's name:
Job1
Optional note:
Description of Job
[Job Executable]
[Job I/O]
[JDL/RSL]
[History]
Port Number:0 Port Name: outputJob0 (channel)
Description of Port
Input Port's Internal File Name:
outputJob0
Port dependent condition allowing the run of the job:
View Hide
Parametric Input details:
View Hide
Port Number:1 Port Name: outputJob1
Description of Port
Output Port's Internal File Name:
outputJob1


Concrete
Job Status
Selection window: start index - range 0 1
Set selection
Show 10 entries
Search:
PID Resource Status View info
0 cygnus.grid.rug.nl:8443/cream-pbs-short running
PID Resource Status View info
Job unclassified instances Status Instances [Actions]
Job0 0 running 1 View running
Job1 0 init 1 View init
Powered By gUSE


Job Status
PID Resource Status View info
0 cygnus.grid.rug.nl:8443/cream-pbs-short finished Logbook std. Output std. Error Download file
PID Resource Status View info
Showing 1 to 1 of 1 entries
Previous Next
PBS_NODEFILE=/var/lib/torque/aux//2548797.apus.data.gridnet
GLITE_WMS_LOG_DESTINATION=cygnus.grid.rug.nl
EDG_WL_NOSETPGRP=1
VO_LOFAR_DEFAULT_SE=srm.target.rug.nl
PBS_O_PATH=/bin:/sbin:/usr/sbin:/usr/bin:/usr/local/sbin:/nethome/tutor012/bin
module=() { eval `/usr/bin/modulecmd sh $*`
}
_=/bin/env
Publications Help End User
Job unclassified instances Status Instances [Actions]
Job0 0 finished 1 View finished
Job1 0 running 1 View running
Powered By gUSE