# High-level APIs for managing computations on the HPC systems

**Marek Kasztelnik,** Tomasz Gubała, Piotr Nowakowski, Jan Meizner, Piotr Połeć, Maciej Malawski, Marian Bubak

Academic Computer Centre Cyfronet AGH University of Science and Technology, Kraków, Poland, https://cyfronet.pl
Sano Centre for Computational Medicine, Kraków, Poland, https://sano.science

European Commission | Horizon 2020 European Union funding for Research & Innovation

H2020 EU PROJECT | Topic SC1-DTH-07-2018 | GA: 826494

## Running computations on the Prometheus HPC - typical routine approach

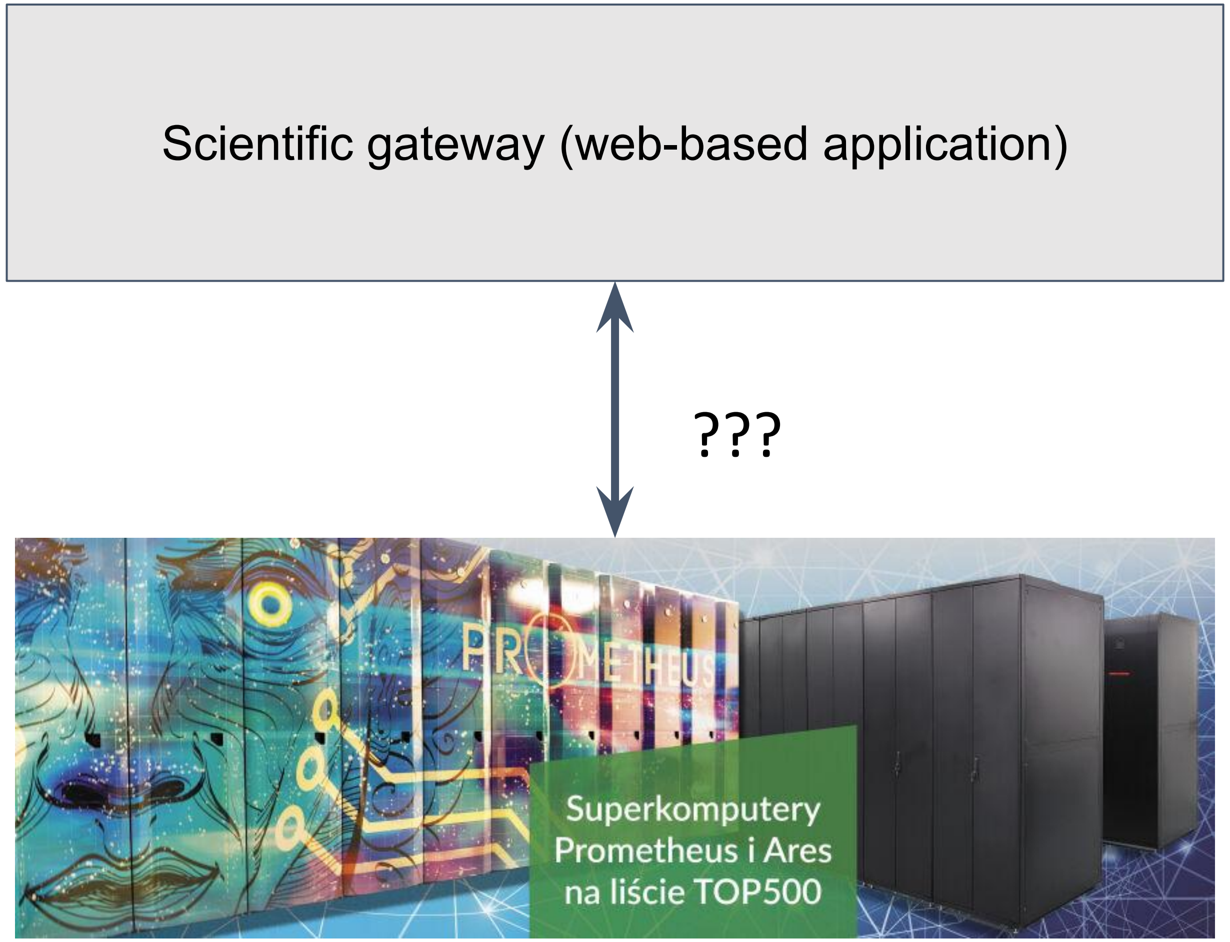
To run a computation on Prometheus usually following steps are performed:

- Copy all inputs (e.g. by using *sftp* - plgrid username and password is required)
- Copy computation code (e.g. by using *sftp* - plgrid username and password are required)
- Create slurm starting script and copy it to Prometheus
- Use ***sbatch*** to start the calculation
- Monitor started job (e.g. by using ***pro-jobs***)
- To retrieve job statistics, you can use ***sacct***
- During execution (and after computation is finished) you can monitor *stdout* and *stderr* by displaying the generated stdout and stderr files (e.g. *tail -f std**)
- After the calculation is finished you can download results (once again e.g. by using *sftp* - PLGrid username and password are required)

The problem:

- Lots of manual work needs to be done (copy files, log in to the cluster, run many commands to start and monitor execution)
- It is hard (or even impossible) to integrate this kind of workflow with third-party application (e.g. workflow/pipeline management tool) because nobody (:-)) will paste PLGrid username/password to a third-party application

## Problem - how to integrate scientific portal with Prometheus?


Scientific gateway (web-based application)
???
PROMETHEUS
Superkomputery Prometheus i Ares na liście TOP500

## Solution (??) - user rights delegation by using grid proxy certificate

Scientific gateway (web-based application)

user proxy certificate

GSI-SSH
GridFTP

User proxy certificate - short-lived certificate signed by long-lived certificate

GSI-SSH - open ssl connection to remote server where user proxy certificate is used to authenticate and delegate user rights

GridFTP - open FTP connection to remote server where user proxy certificate is used to authenticate and delegate user rights

## Solution (??) - user rights delegation by using user proxy certificate

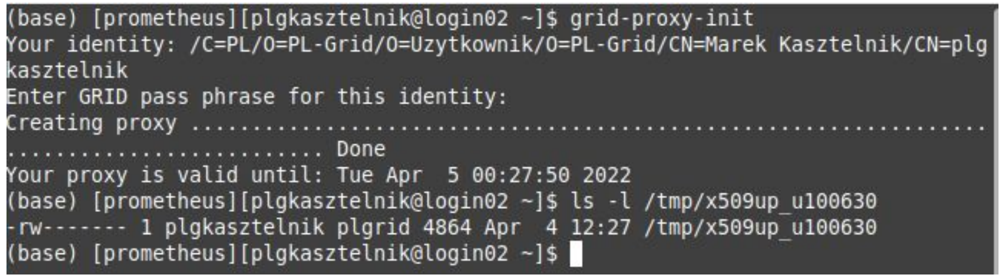
Scientific gateway (web-based application)

user proxy certificate

GSI-SSH
GridFTP

How to get user proxy certificate?

Option1: Log in to Prometheus cluster and run ***grid-proxy-init***



```
(base) [prometheus][plgkasztelnik@login02 ~]$ grid-proxy-init
Your identity: /C=PL/O=PL-Grid/O=Uzytkownik/O=PL-Grid/CN=Marek Kasztelnik/CN=plg
kasztelnik
Enter GRID pass phrase for this identity:
Creating proxy .................................................................
........................... Done
Your proxy is valid until: Tue Apr  5 00:27:50 2022
(base) [prometheus][plgkasztelnik@login02 ~]$ ls -l /tmp/x509up_u100630
-rw------- 1 plgkasztelnik plgrid 4864 Apr  4 12:27 /tmp/x509up_u100630
(base) [prometheus][plgkasztelnik@login02 ~]$ █
```

and upload the generated proxy to the scientific gateway.

This solution has the following drawbacks:

- The proxy is only valid for a short period of time and needs to be refreshed frequently
- We wanted to avoid the command line, didn't we?

## Solution (??) - user rights delegation by using user proxy certificate

Scientific gateway (web-based application)

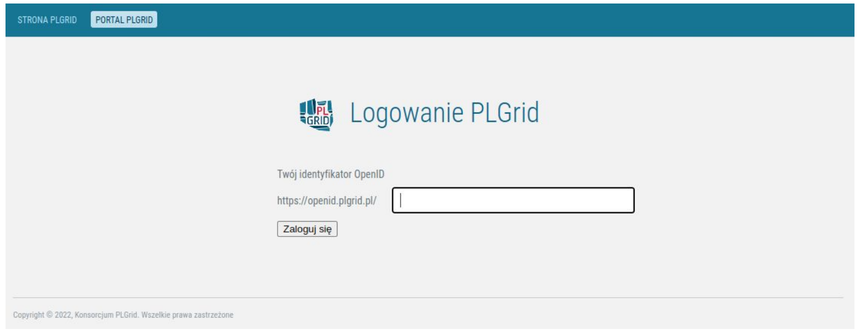
user proxy certificate

Login

User details and
user proxy certificate

GSI-SSH
GridFTP

Option2: Use PLGrid IDP (identity provider)



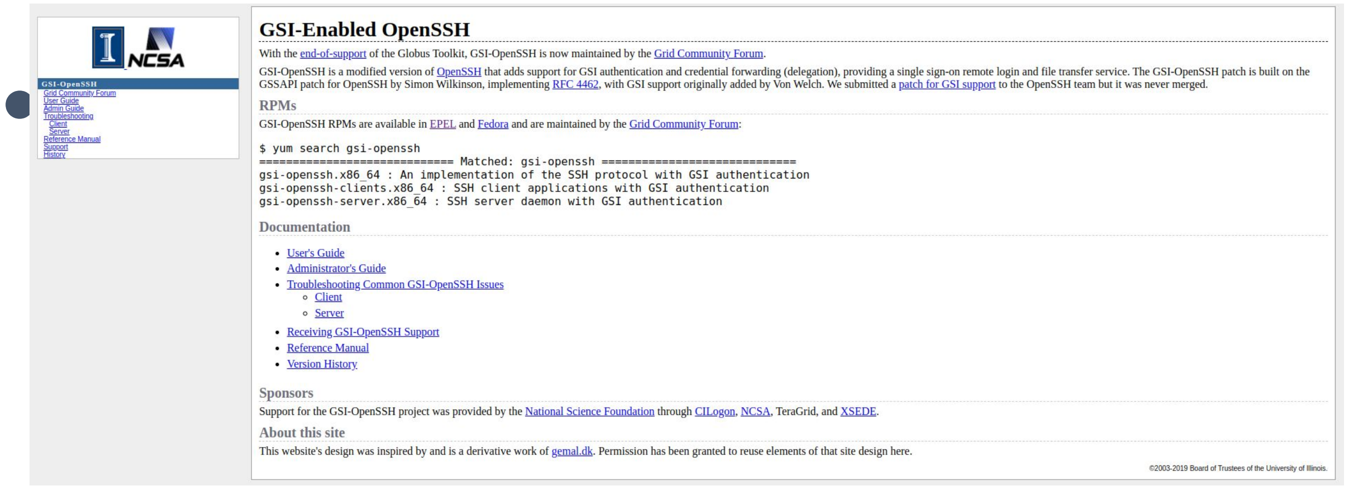
Twój identyfikator OpenID

https://openid.plgrid.pl/

Zaloguj się

Copyright © 2022, Konsorcjum PLGrid. Wszelkie prawa zastrzeżone

# GSI SSH, GridFTP - how to install and use it?



GSI-OpenSSH
Grid Community Forum
User Guide
Admin Guide
Troubleshooting
Client
Server
Reference Manual
Support
History

## GSI-Enabled OpenSSH

With the end-of-support of the Globus Toolkit, GSI-OpenSSH is now maintained by the Grid Community Forum.

GSI-OpenSSH is a modified version of OpenSSH that adds support for GSI authentication and credential forwarding (delegation), providing a single sign-on remote login and file transfer service. The GSI-OpenSSH patch is built on the GSSAPI patch for OpenSSH by Simon Wilkinson, implementing RFC 4462, with GSI support originally added by Von Welch. We submitted a patch for GSI support to the OpenSSH team but it was never merged.

### RPMs

GSI-OpenSSH RPMs are available in EPEL and Fedora and are maintained by the Grid Community Forum:

```
$ yum search gsi-openssh
============================ Matched: gsi-openssh =============================
gsi-openssh.x86_64 : An implementation of the SSH protocol with GSI authentication
gsi-openssh-clients.x86_64 : SSH client applications with GSI authentication
gsi-openssh-server.x86_64 : SSH server daemon with GSI authentication
```

### Documentation

- User's Guide
- Administrator's Guide
- Troubleshooting Common GSI-OpenSSH Issues
  - Client
  - Server
- Receiving GSI-OpenSSH Support
- Reference Manual
- Version History

### Sponsors

Support for the GSI-OpenSSH project was provided by the National Science Foundation through CILogon, NCSA, TeraGrid, and XSEDE.

### About this site

This website's design was inspired by and is a derivative work of gemal.dk. Permission has been granted to reuse elements of that site design here.

©2003-2019 Board of Trustees of the University of Illinois.

Problems:

- Not easy to install
- C based implementation
- Java implementation is outdated and not maintained anymore
- No bindings for modern programming languages

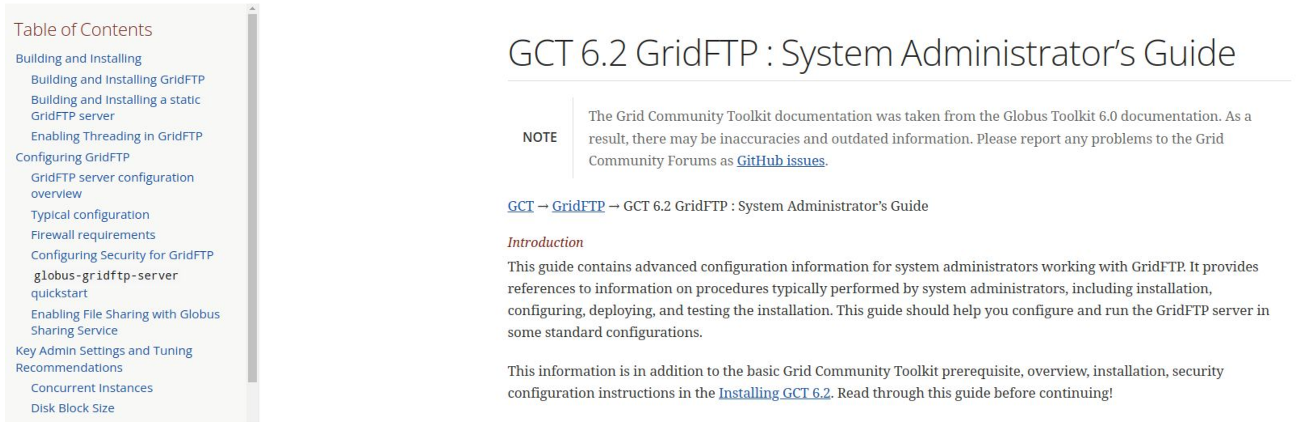


Table of Contents

## GCT 6.2 GridFTP : System Administrator's Guide

NOTE The Grid Community Toolkit documentation was taken from the Globus Toolkit 6.0 documentation. As a result, there may be inaccuracies and outdated information. Please report any problems to the Grid Community Forums as GitHub issues.

GCT → GridFTP → GCT 6.2 GridFTP : System Administrator's Guide

*Introduction*

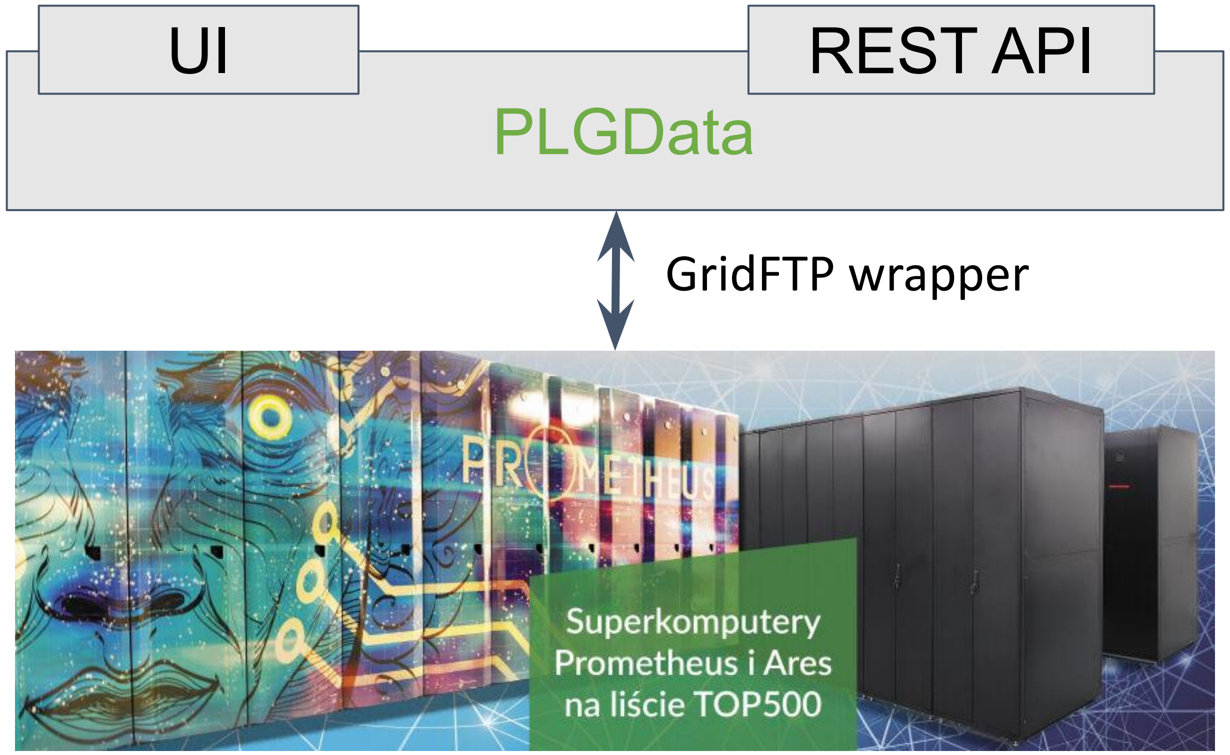
This guide contains advanced configuration information for system administrators working with GridFTP. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation. This guide should help you configure and run the GridFTP server in some standard configurations.

This information is in addition to the basic Grid Community Toolkit prerequisite, overview, installation, security configuration instructions in the Installing GCT 6.2. Read through this guide before continuing!

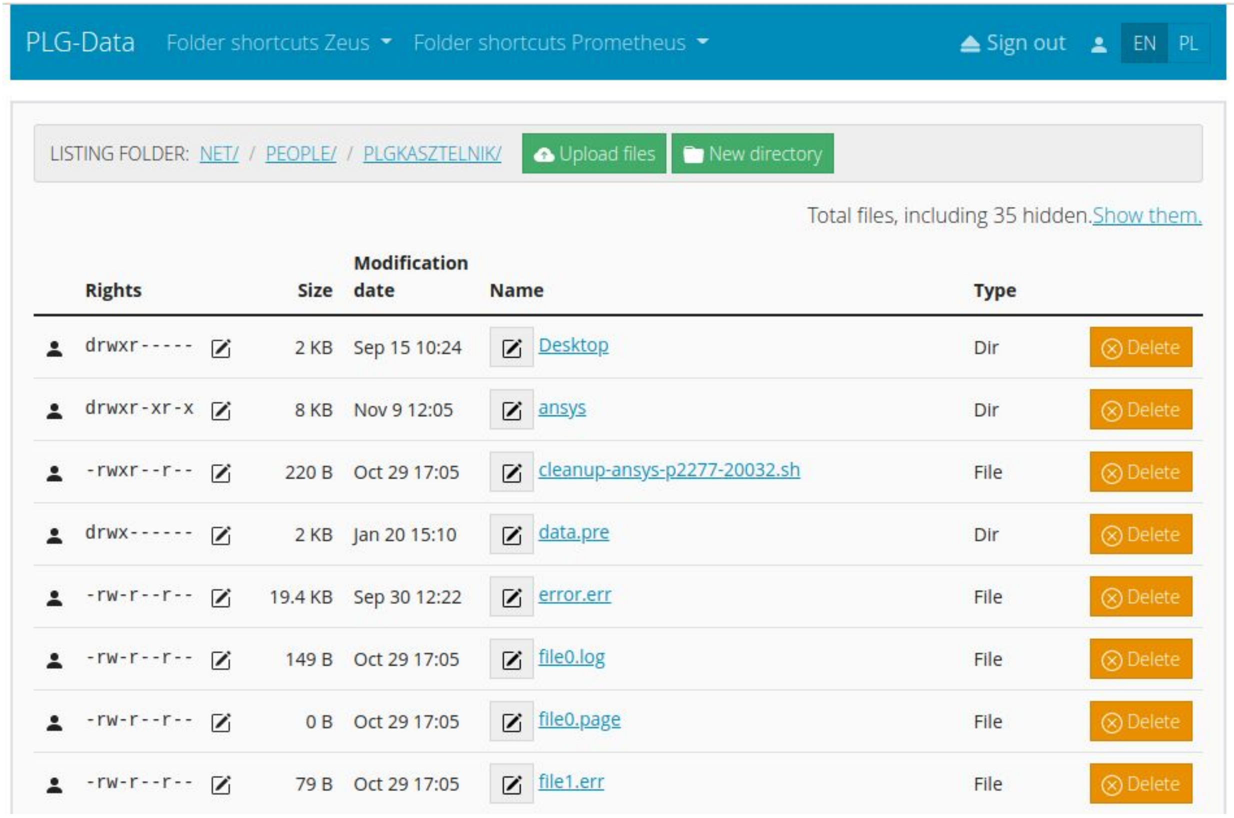
PRIMAGE

## PLGData to the rescue - files management

- PLGrid Data (https://data.plgrid.pl) is a web interface and set of REST APIs for managing files stored on Prometheus.
- It is integrated with the PLGrid security system
- If you have a PLGrid account and access to Prometheus turned on, and have generated a SimpleCA certificate for access delegation, you are able to use this intuitive tool for file management.



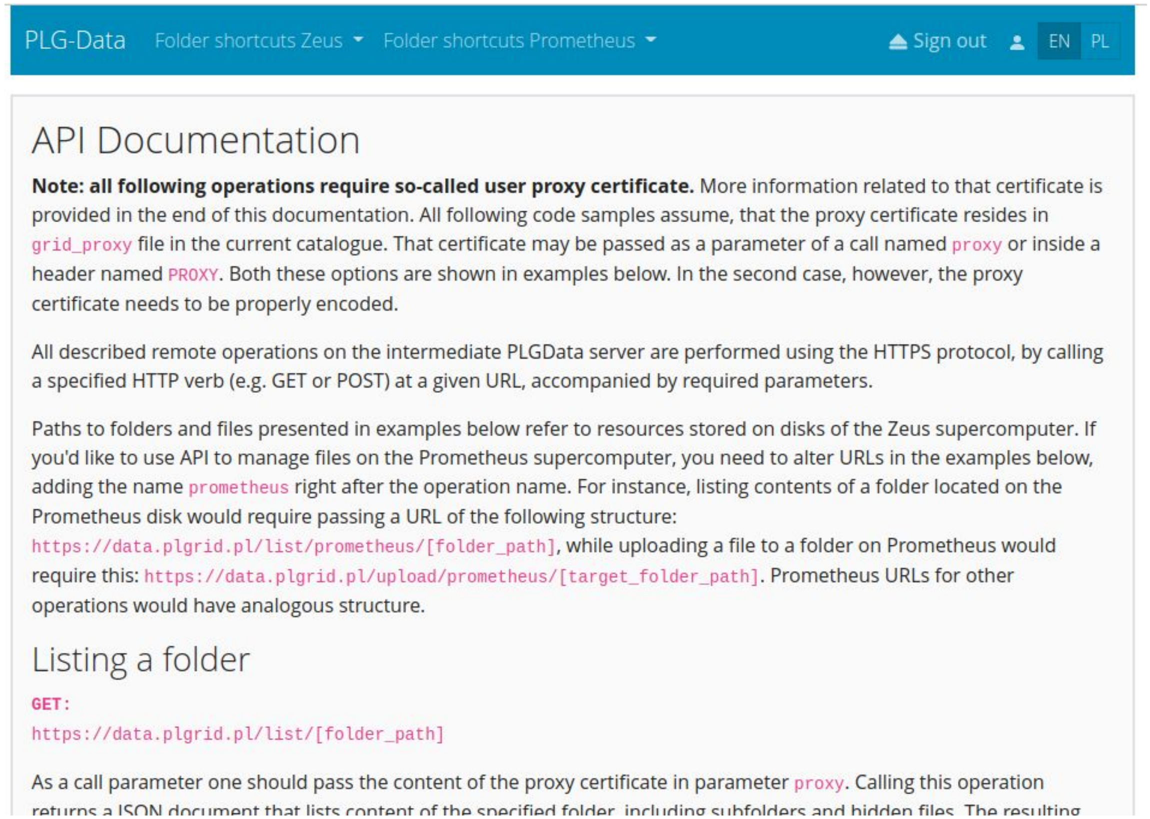
# PLGData to the rescue - how it looks like



LISTING FOLDER: NET/ / PEOPLE/ / PLGKASZTELNIK/

Total files, including 35 hidden.Show them.

| Rights | Size | Modification date | Name | Type | |
|---|---|---|---|---|---|
| drwxr----- | 2 KB | Sep 15 10:24 | Desktop | Dir | Delete |
| drwxr-xr-x | 8 KB | Nov 9 12:05 | ansys | Dir | Delete |
| -rwxr--r-- | 220 B | Oct 29 17:05 | cleanup-ansys-p2277-20032.sh | File | Delete |
| drwx------ | 2 KB | Jan 20 15:10 | data.pre | Dir | Delete |
| -rw-r--r-- | 19.4 KB | Sep 30 12:22 | error.err | File | Delete |
| -rw-r--r-- | 149 B | Oct 29 17:05 | file0.log | File | Delete |
| -rw-r--r-- | 0 B | Oct 29 17:05 | file0.page | File | Delete |
| -rw-r--r-- | 79 B | Oct 29 17:05 | file1.err | File | Delete |



API Documentation

**Note: all following operations require so-called user proxy certificate.** More information related to that certificate is provided in the end of this documentation. All following code samples assume, that the proxy certificate resides in `grid_proxy` file in the current catalogue. That certificate may be passed as a parameter of a call named `proxy` or inside a header named `PROXY`. Both these options are shown in examples below. In the second case, however, the proxy certificate needs to be properly encoded.

All described remote operations on the intermediate PLGData server are performed using the HTTPS protocol, by calling a specified HTTP verb (e.g. GET or POST) at a given URL, accompanied by required parameters.

Paths to folders and files presented in examples below refer to resources stored on disks of the Zeus supercomputer. If you'd like to use API to manage files on the Prometheus supercomputer, you need to alter URLs in the examples below, adding the name `prometheus` right after the operation name. For instance, listing contents of a folder located on the Prometheus disk would require passing a URL of the following structure: `https://data.plgrid.pl/list/prometheus/[folder_path]`, while uploading a file to a folder on Prometheus would require this: `https://data.plgrid.pl/upload/prometheus/[target_folder_path]`. Prometheus URLs for other operations would have analogous structure.

Listing a folder

```
GET:
https://data.plgrid.pl/list/[folder_path]
```

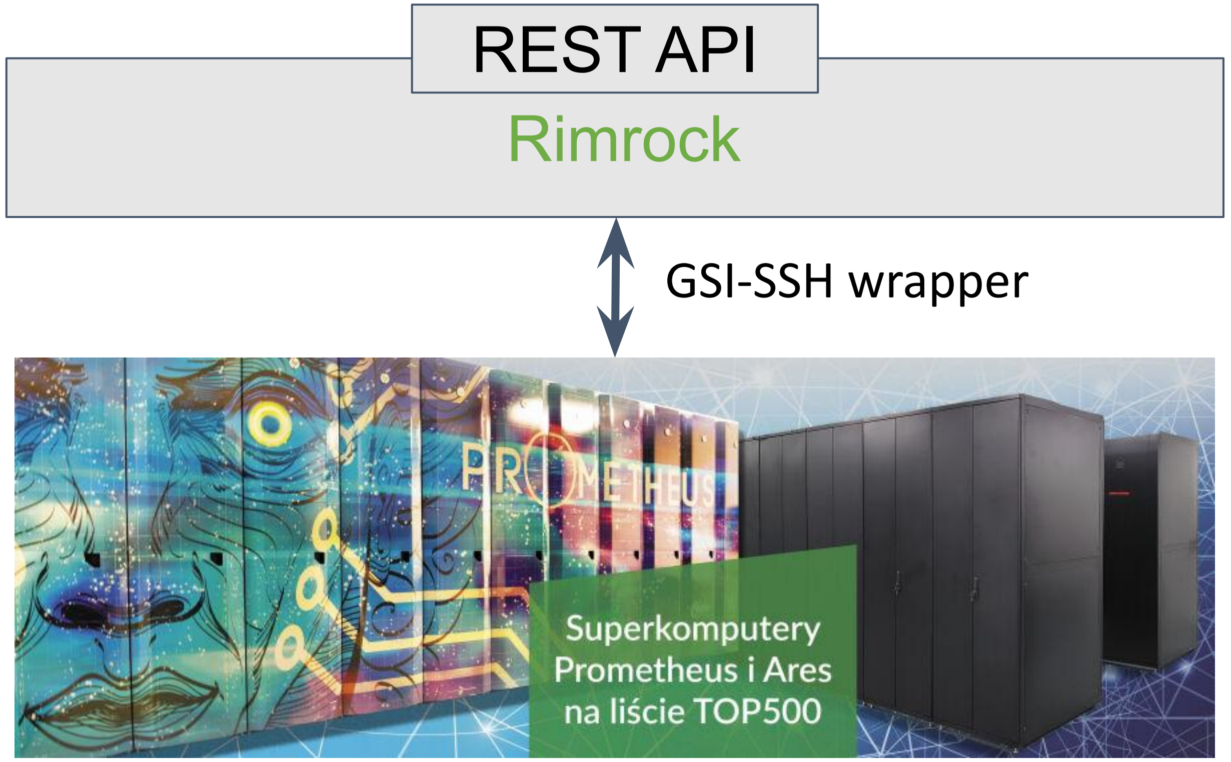
As a call parameter one should pass the content of the proxy certificate in parameter `proxy`. Calling this operation returns a JSON document that lists content of the specified folder, including subfolders and hidden files. The resulting

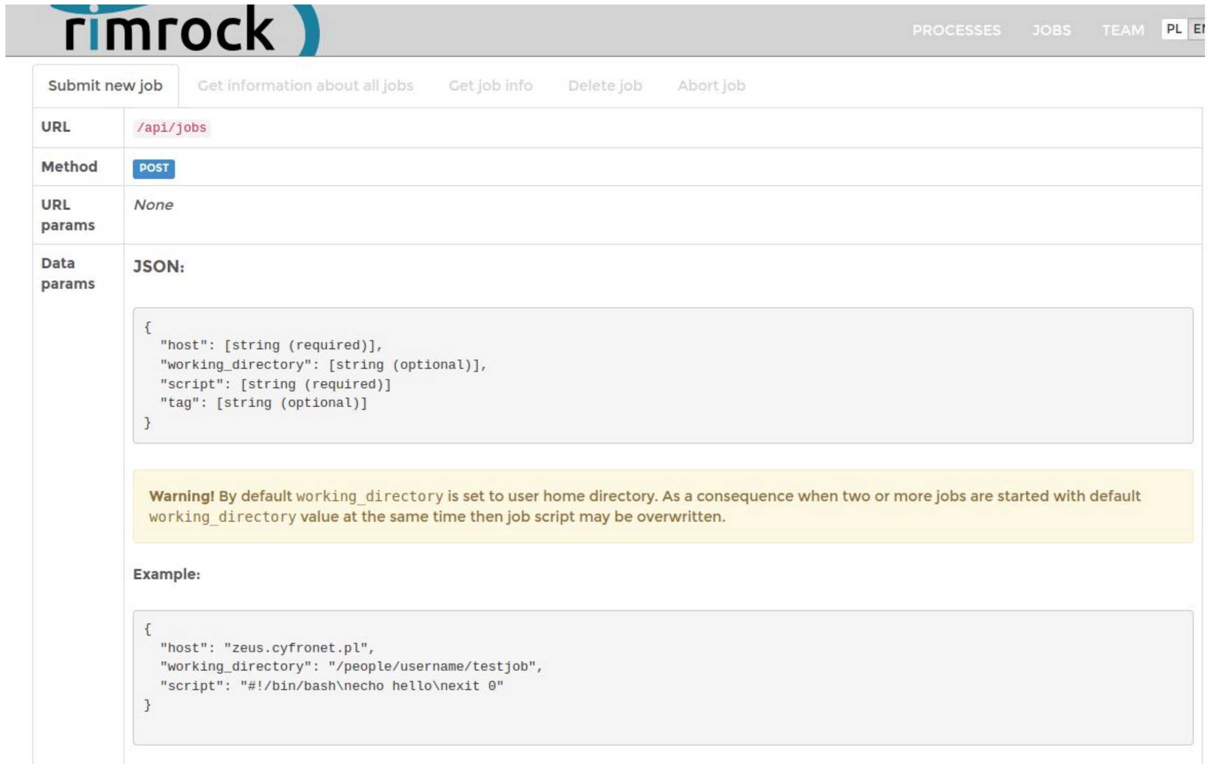
```
~ % curl -X POST https://data.plgrid.pl/upload/people/plguserlogin/zzzz
-F proxy="`cat grid_proxy`" -F "file=@graph.png"
```

PRIMAGE

## Rimrock to the rescue - jobs managements

- Rimrock ([https://rimrock.plgrid.pl](https://rimrock.plgrid.pl)) delivers a REST APIs to talk to integrated HPC infrastructures, including Prometheus
- All REST requests are secured by proxy certificates, which enable delegation of user identities





Submit new job | Get information about all jobs | Get job info | Delete job | Abort job

| | |
|---|---|
| URL | /api/jobs |
| Method | POST |
| URL params | None |
| Data params | JSON: |

```
{
  "host": [string (required)],
  "working_directory": [string (optional)],
  "script": [string (required)]
  "tag": [string (optional)]
}
```

**Warning!** By default `working_directory` is set to user home directory. As a consequence when two or more jobs are started with default `working_directory` value at the same time then job script may be overwritten.

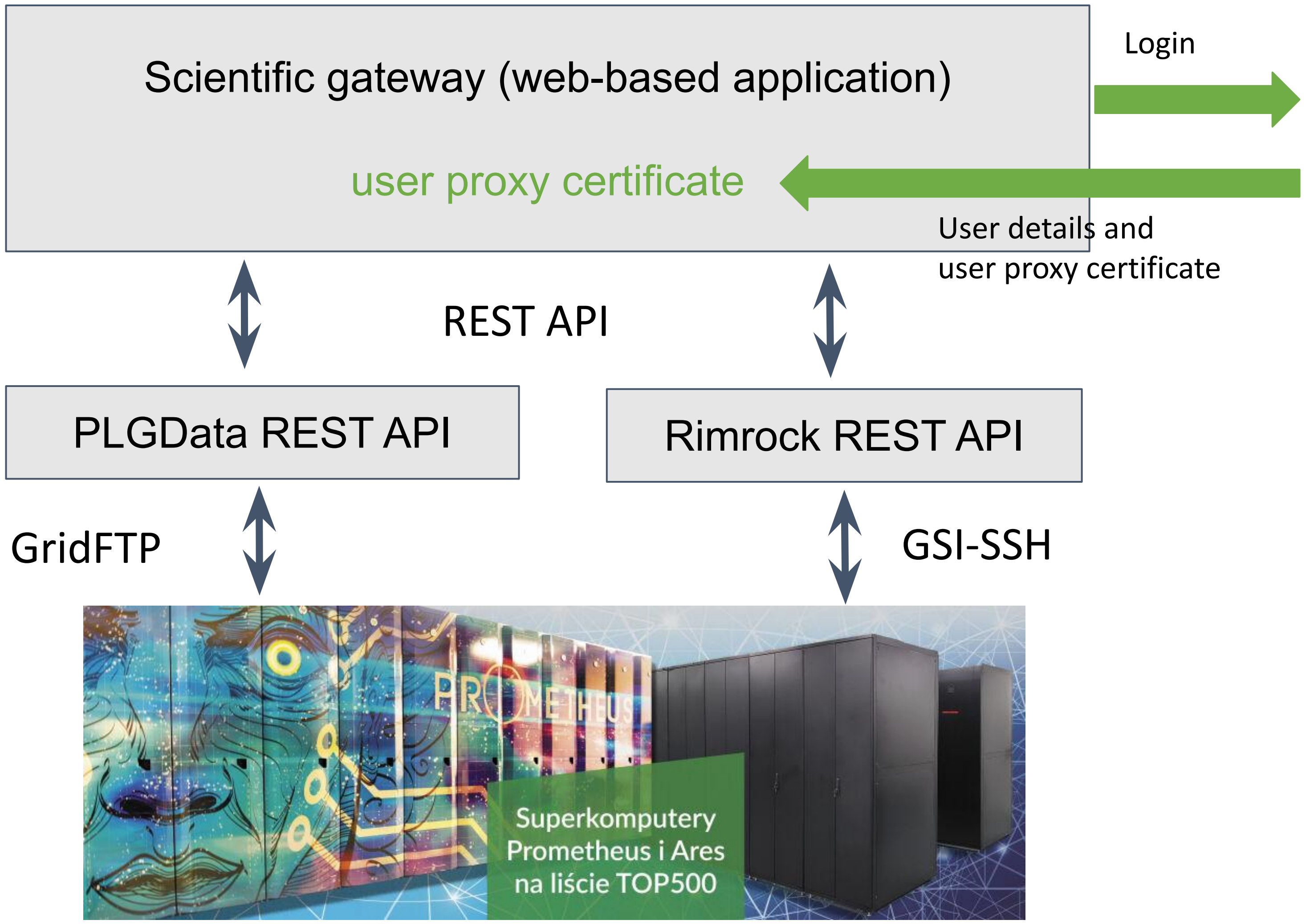
**Example:**

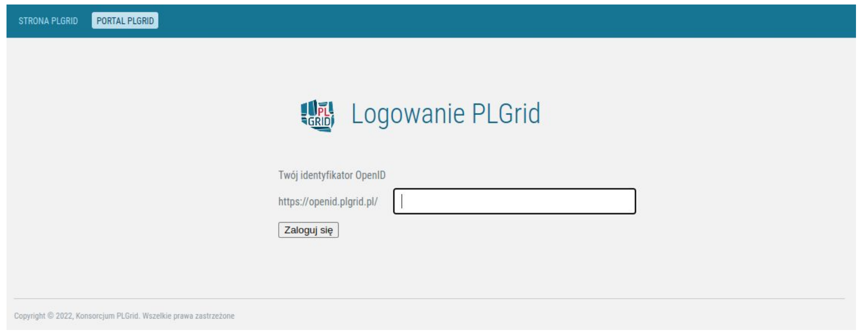
```
{
  "host": "zeus.cyfronet.pl",
  "working_directory": "/people/username/testjob",
  "script": "#!/bin/bash\necho hello\nexit 0"
}
```

```
curl -k -X POST --data '{"host":"pro.cyfronet.pl", "script":"#!/bin/bash\n#SBATCH -A {grantid}\necho hello\nexit 0"}' \
 --header "Content-Type:application/json" --header "PROXY:$proxy" https://rimrock.plgrid.pl/api/jobs
```

# Solution !!!

Option2: Use PLGrid IDP





Twój identyfikator OpenID

https://openid.plgrid.pl/

Zaloguj się

Copyright © 2022, Konsorcjum PLGrid. Wszelkie prawa zastrzeżone

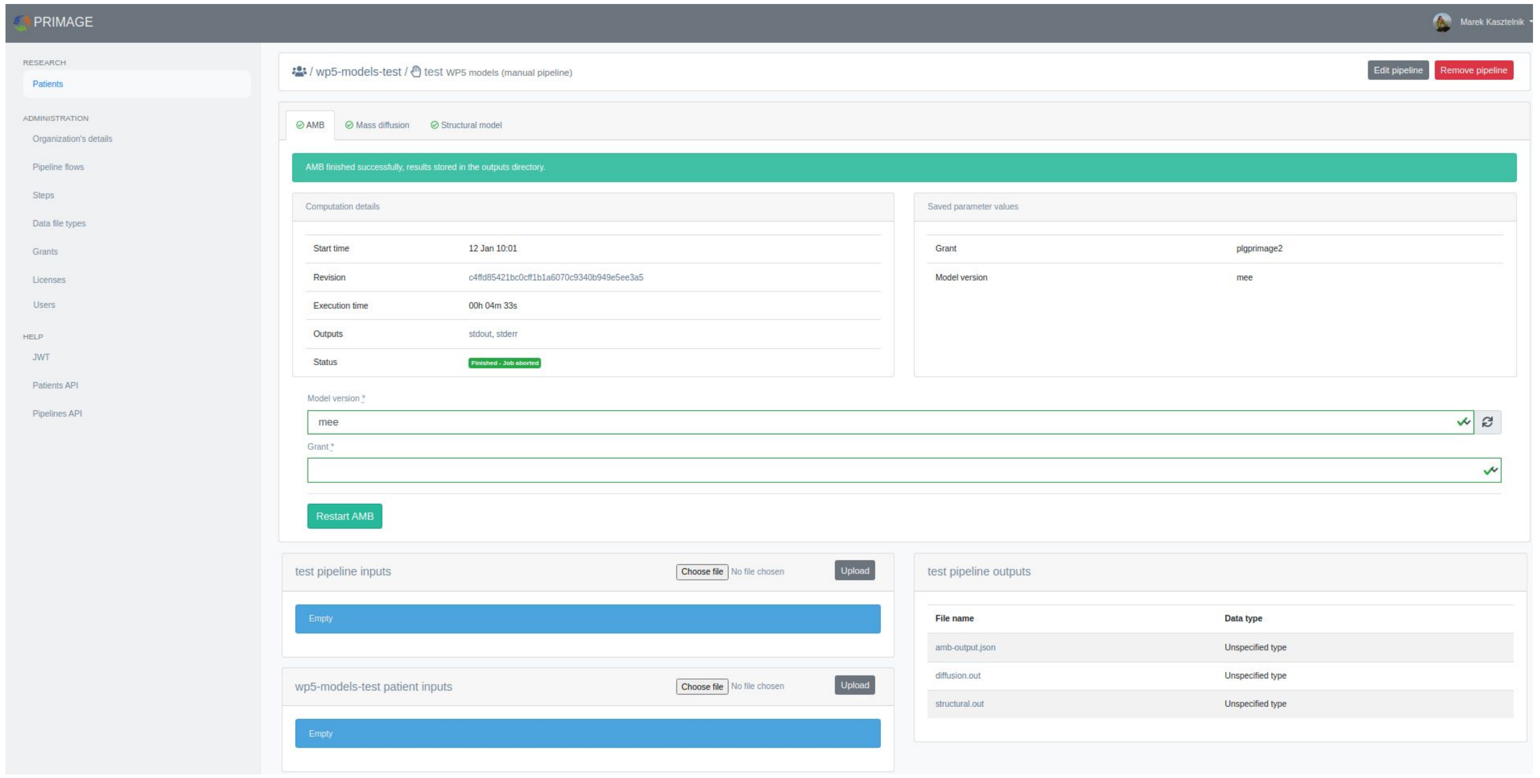
## Read world example -> Model Execution Environment (MEE)

**High-level service to manage data and computations in the context of a patient cohort.**

- Select patient (or group of patients)
- Choose computation pipeline (series of calculation with input/output dependencies)
- Start the pipeline
- Browse the results

**Features:**

- Integrated with Prometheus (automatic user credential delegation)
- Selecting data, starting the calculations, browsing the results in one place
- Automatic and manual pipeline execution



/ wp5-models-test / test WP5 models (manual pipeline)

AMB finished successfully, results stored in the outputs directory.

Computation details

| | |
|---|---|
| Start time | 12 Jan 10:01 |
| Revision | c4ffd85421bc0cff1b1a6070c9340b949e5ee3a5 |
| Execution time | 00h 04m 33s |
| Outputs | stdout, stderr |
| Status | Finished - Job aborted |

Saved parameter values

| | |
|---|---|
| Grant | plgprimage2 |
| Model version | mee |

Model version *: mee

Grant *:

test pipeline inputs

Empty

wp5-models-test patient inputs

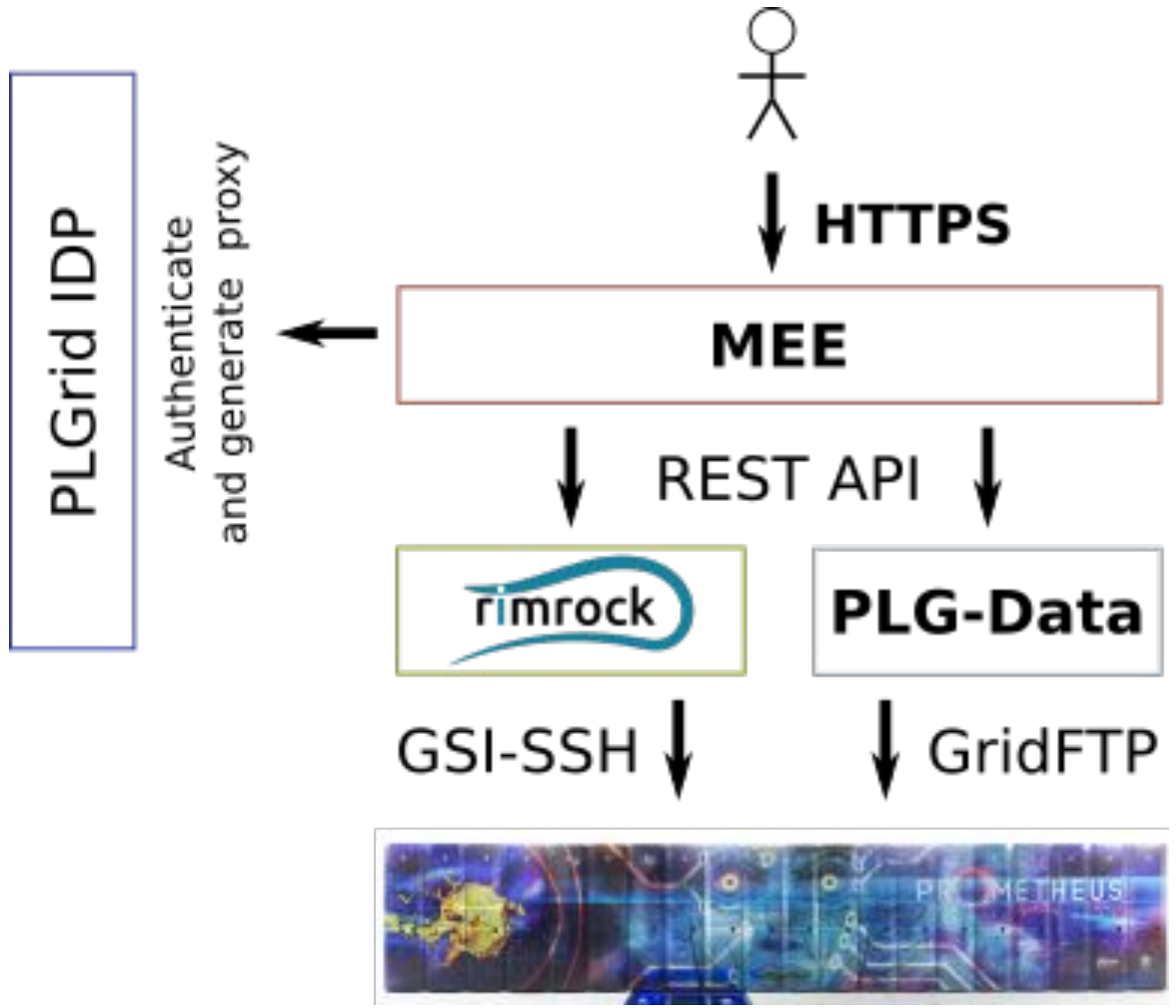
Empty

test pipeline outputs

| File name | Data type |
|---|---|
| amb-output.json | Unspecified type |
| diffusion.out | Unspecified type |
| structural.out | Unspecified type |

PRIMAGE

## Real world example -> MEE architecture



**Basic flow:**

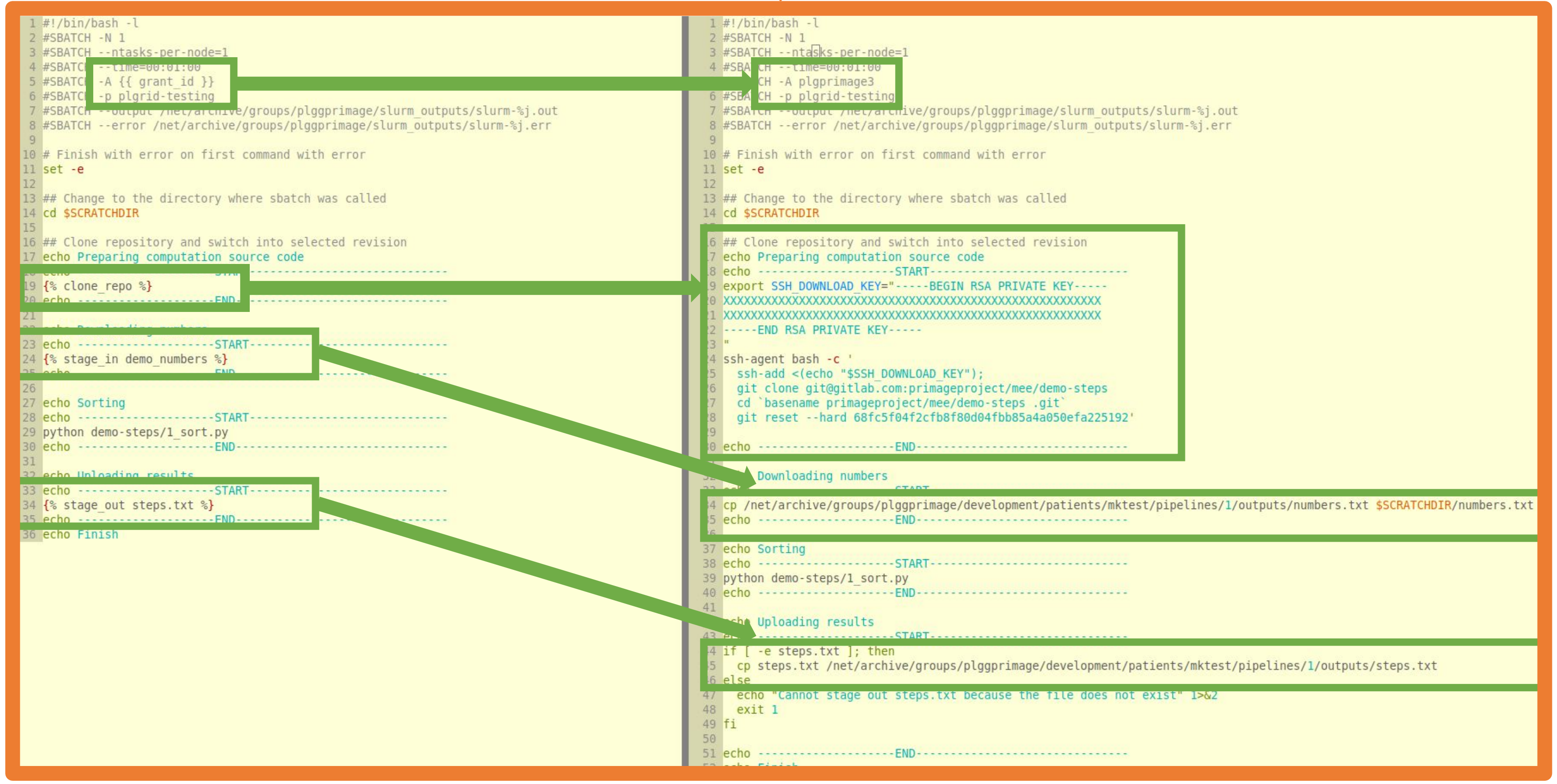
- When user logs in, a short-lived user certificate is stored in the DB
- When browsing pipeline inputs and outputs the proxy certificate is used to delegate user rights to the infrastructure by using PLGData REST API
- When calculation is started or managed, the proxy certificate is used to delegate user rights to the infrastructure by using Rimrock REST API

## MEE - running computations

1. **MEE** checks if all required inputs are present and shows an error when inputs are missing
2. Fetch model starting script template from GIT repository in the version specified by the user
3. Generate model starting script basing on the inputs provided by the user
4. Submit job to HPC queuing system (Slurm)
5. Upload required inputs to HPC
6. Monitor job execution
7. Fetch results from HPC

# MEE - running computations

1. **MEE** checks if all required inputs are present and shows an error when inputs are missing
2. Fetch model starting script template from GIT repository in the version specified by the user
3. Generate model starting script basing on the inputs provided by the user
4. Submit job to HPC queuing system (Slurm)
5. Upload required inputs to HPC
6. Monitor job execution
7. Fetch results from HPC



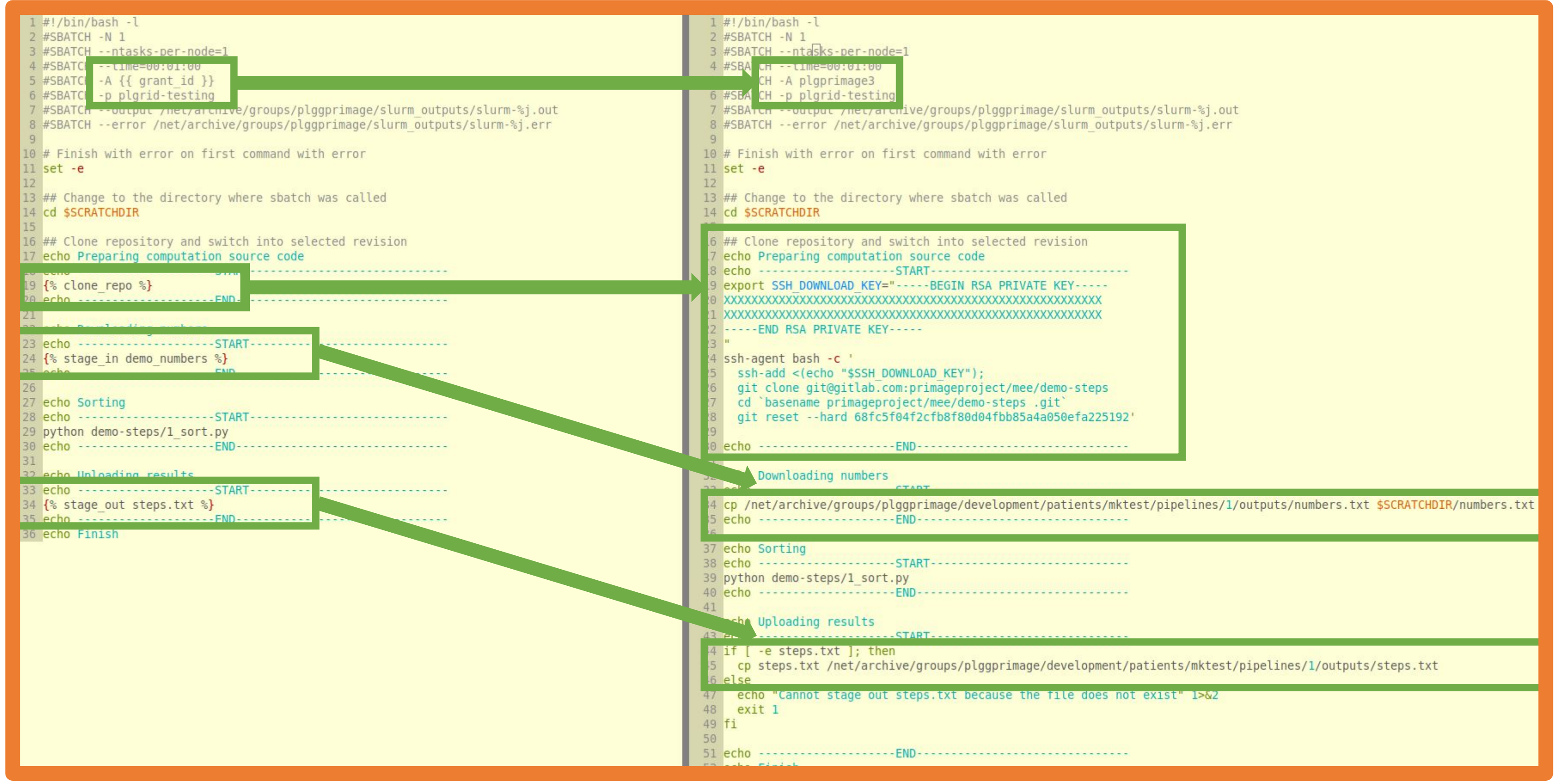
```
#!/bin/bash -l
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:01:00
#SBATCH -A {{ grant_id }}
#SBATCH -p plgrid-testing
#SBATCH --output /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.out
#SBATCH --error /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.err

# Finish with error on first command with error
set -e

## Change to the directory where sbatch was called
cd $SCRATCHDIR

## Clone repository and switch into selected revision
echo Preparing computation source code
echo --------------------START------------------------------
{% clone_repo %}
echo --------------------END--------------------------------

echo Downloading numbers
echo --------------------START------------------------------
{% stage_in demo_numbers %}
echo --------------------END--------------------------------

echo Sorting
echo --------------------START------------------------------
python demo-steps/1_sort.py
echo --------------------END--------------------------------

echo Uploading results
echo --------------------START------------------------------
{% stage_out steps.txt %}
echo --------------------END--------------------------------
echo Finish
```

```
#!/bin/bash -l
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:01:00
#SBATCH -A plgprimage3
#SBATCH -p plgrid-testing
#SBATCH --output /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.out
#SBATCH --error /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.err

# Finish with error on first command with error
set -e

## Change to the directory where sbatch was called
cd $SCRATCHDIR

## Clone repository and switch into selected revision
echo Preparing computation source code
echo --------------------START------------------------------
export SSH_DOWNLOAD_KEY="-----BEGIN RSA PRIVATE KEY-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-----END RSA PRIVATE KEY-----
"
ssh-agent bash -c '
  ssh-add <(echo "$SSH_DOWNLOAD_KEY");
  git clone git@gitlab.com:primageproject/mee/demo-steps
  cd `basename primageproject/mee/demo-steps .git`
  git reset --hard 68fc5f04f2cfb8f80d04fbb85a4a050efa225192'

echo --------------------END--------------------------------

echo Downloading numbers
echo --------------------START------------------------------
cp /net/archive/groups/plggprimage/development/patients/mktest/pipelines/1/outputs/numbers.txt $SCRATCHDIR/numbers.txt
echo --------------------END--------------------------------

echo Sorting
echo --------------------START------------------------------
python demo-steps/1_sort.py
echo --------------------END--------------------------------

echo Uploading results
echo --------------------START------------------------------
if [ -e steps.txt ]; then
  cp steps.txt /net/archive/groups/plggprimage/development/patients/mktest/pipelines/1/outputs/steps.txt
else
  echo "Cannot stage out steps.txt because the file does not exist" 1>&2
  exit 1
fi

echo --------------------END--------------------------------
echo Finish
```

PRIMAGE

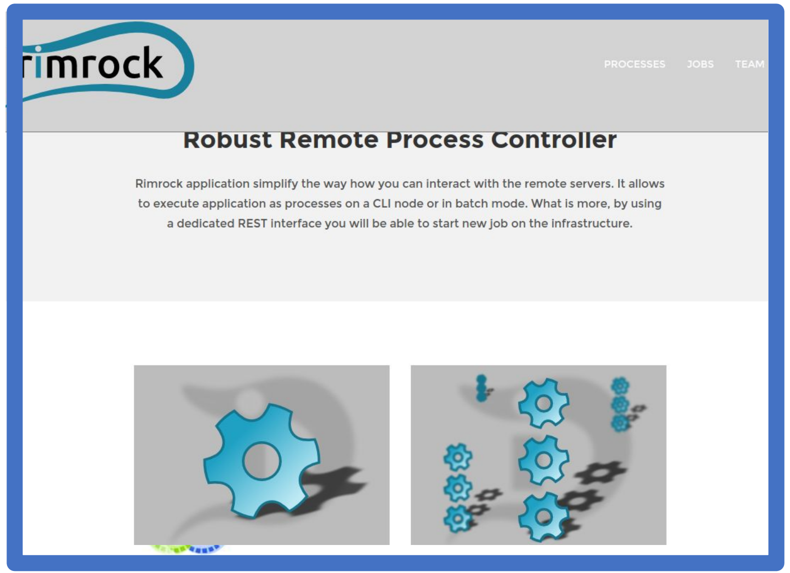
# MEE - running computations

1. **MEE** checks if all required inputs are present and shows an error when inputs are missing
2. Fetch model starting script template from GIT repository in the version specified by the user
3. Generate model starting script basing on the inputs provided by the user
4. Submit job to HPC queuing system (Slurm)
5. Upload required inputs to HPC
6. Monitor job execution
7. Fetch results from HPC



```
#!/bin/bash -l
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:01:00
#SBATCH -A {{ grant_id }}
#SBATCH -p plgrid-testing
#SBATCH --output /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.out
#SBATCH --error /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.err

# Finish with error on first command with error
set -e

## Change to the directory where sbatch was called
cd $SCRATCHDIR

## Clone repository and switch into selected revision
echo Preparing computation source code
echo --------------------START------------------------------
{% clone_repo %}
echo --------------------END--------------------------------

echo Downloading numbers
echo --------------------START------------------------------
{% stage_in demo_numbers %}
echo --------------------END--------------------------------

echo Sorting
echo --------------------START------------------------------
python demo-steps/1_sort.py
echo --------------------END--------------------------------

echo Uploading results
echo --------------------START------------------------------
{% stage_out steps.txt %}
echo --------------------END--------------------------------
echo Finish
```

```
#!/bin/bash -l
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:01:00
#SBATCH -A plgprimage3
#SBATCH -p plgrid-testing
#SBATCH --output /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.out
#SBATCH --error /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.err

# Finish with error on first command with error
set -e

## Change to the directory where sbatch was called
cd $SCRATCHDIR

## Clone repository and switch into selected revision
echo Preparing computation source code
echo --------------------START------------------------------
export SSH_DOWNLOAD_KEY="-----BEGIN RSA PRIVATE KEY-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-----END RSA PRIVATE KEY-----
"
ssh-agent bash -c '
  ssh-add <(echo "$SSH_DOWNLOAD_KEY");
  git clone git@gitlab.com:primageproject/mee/demo-steps
  cd `basename primageproject/mee/demo-steps .git`
  git reset --hard 68fc5f04f2cfb8f80d04fbb85a4a050efa225192'

echo --------------------END--------------------------------

echo Downloading numbers
echo --------------------START------------------------------
cp /net/archive/groups/plggprimage/development/patients/mktest/pipelines/1/outputs/numbers.txt $SCRATCHDIR/numbers.txt
echo --------------------END--------------------------------

echo Sorting
echo --------------------START------------------------------
python demo-steps/1_sort.py
echo --------------------END--------------------------------

echo Uploading results
echo --------------------START------------------------------
if [ -e steps.txt ]; then
  cp steps.txt /net/archive/groups/plggprimage/development/patients/mktest/pipelines/1/outputs/steps.txt
else
  echo "Cannot stage out steps.txt because the file does not exist" 1>&2
  exit 1
fi

echo --------------------END--------------------------------
```

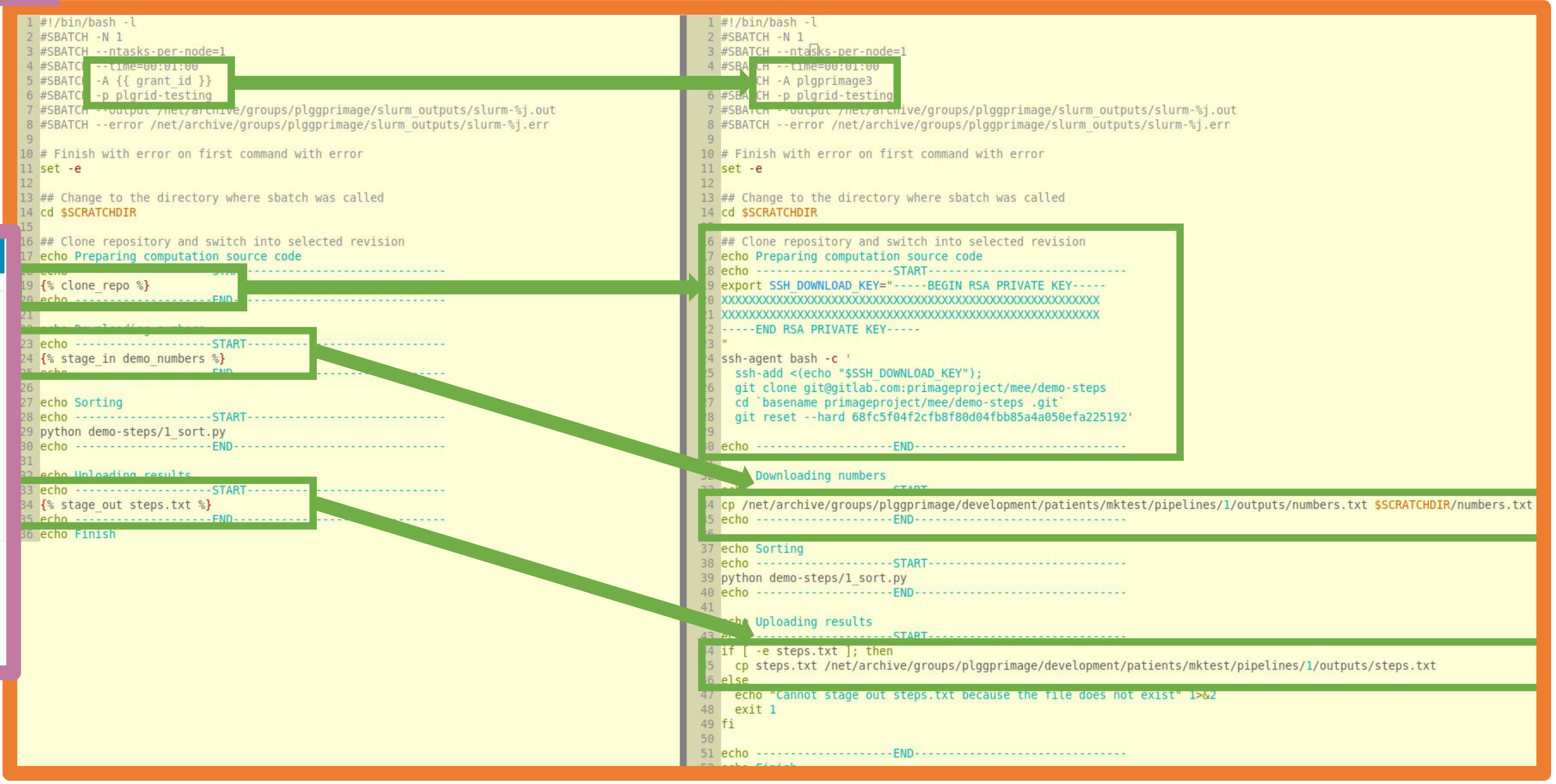


Robust Remote Process Controller

Rimrock application simplify the way how you can interact with the remote servers. It allows to execute application as processes on a CLI node or in batch mode. What is more, by using a dedicated REST interface you will be able to start new job on the infrastructure.

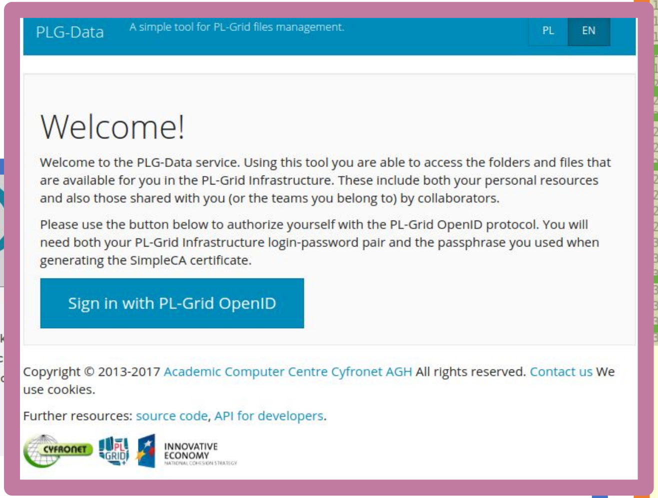
# MEE - running computations

1. **MEE** checks if all required inputs are present and shows an error when inputs are missing
2. Fetch model starting script template from GIT repository in the version specified by the user
3. Generate model starting script basing on the inputs provided by the user
4. Submit job to HPC queuing system (Slurm)
5. Upload required inputs to HPC
6. Monitor job execution
7. Fetch results from HPC



```
#!/bin/bash -l
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:01:00
#SBATCH -A {{ grant_id }}
#SBATCH -p plgrid-testing
#SBATCH --output /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.out
#SBATCH --error /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.err

# Finish with error on first command with error
set -e

## Change to the directory where sbatch was called
cd $SCRATCHDIR

## Clone repository and switch into selected revision
echo Preparing computation source code
echo --------------------START------------------------------
{% clone_repo %}
echo --------------------END--------------------------------

echo Downloading numbers
echo --------------------START------------------------------
{% stage_in demo_numbers %}
echo --------------------END--------------------------------

echo Sorting
echo --------------------START------------------------------
python demo-steps/1_sort.py
echo --------------------END--------------------------------

echo Uploading results
echo --------------------START------------------------------
{% stage_out steps.txt %}
echo --------------------END--------------------------------
echo Finish
```

```
#!/bin/bash -l
#SBATCH -N 1
#SBATCH --ntasks-per-node=1
#SBATCH --time=00:01:00
#SBATCH -A plgprimage3
#SBATCH -p plgrid-testing
#SBATCH --output /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.out
#SBATCH --error /net/archive/groups/plggprimage/slurm_outputs/slurm-%j.err

# Finish with error on first command with error
set -e

## Change to the directory where sbatch was called
cd $SCRATCHDIR

## Clone repository and switch into selected revision
echo Preparing computation source code
echo --------------------START------------------------------
export SSH_DOWNLOAD_KEY="-----BEGIN RSA PRIVATE KEY-----
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
-----END RSA PRIVATE KEY-----
"
ssh-agent bash -c '
  ssh-add <(echo "$SSH_DOWNLOAD_KEY");
  git clone git@gitlab.com:primageproject/mee/demo-steps
  cd `basename primageproject/mee/demo-steps .git`
  git reset --hard 68fc5f04f2cfb8f80d04fbb85a4a050efa225192'

echo --------------------END--------------------------------

echo Downloading numbers
echo --------------------START------------------------------
cp /net/archive/groups/plggprimage/development/patients/mktest/pipelines/1/outputs/numbers.txt $SCRATCHDIR/numbers.txt
echo --------------------END--------------------------------

echo Sorting
echo --------------------START------------------------------
python demo-steps/1_sort.py
echo --------------------END--------------------------------

echo Uploading results
echo --------------------START------------------------------
if [ -e steps.txt ]; then
  cp steps.txt /net/archive/groups/plggprimage/development/patients/mktest/pipelines/1/outputs/steps.txt
else
  echo "Cannot stage out steps.txt because the file does not exist" 1>&2
  exit 1
fi

echo --------------------END--------------------------------
echo Finish
```



Welcome!

Welcome to the PLG-Data service. Using this tool you are able to access the folders and files that are available for you in the PL-Grid Infrastructure. These include both your personal resources and also those shared with you (or the teams you belong to) by collaborators.

Please use the button below to authorize yourself with the PL-Grid OpenID protocol. You will need both your PL-Grid Infrastructure login-password pair and the passphrase you used when generating the SimpleCA certificate.

Sign in with PL-Grid OpenID

Copyright © 2013-2017 Academic Computer Centre Cyfronet AGH All rights reserved. Contact us We use cookies.

Further resources: source code, API for developers.

## Conclusions

**External applications can easily be integrated with the Prometheus cluster with**:

- **PLGrid identity provider** to log in and generate a proxy certificate for the user
- **PLGData** to manage files stored on the cluster using a modern web-based UI or REST API
- **Rimrock** to manage computations started on the cluster by using a REST API

**The usefulness of these tools has been proven by many applications integrated with the infrastructure, e.g.:**

- **Model Execution Environment** - an environment that enables computational models to be developed in a simple and organized manner and easily deployed to the available HPC infrastructure. The platform promotes managing simulations in a way that guarantees repeatability, replicability, and reproducibility
- **EPISODES Platform** - European plate induced seismicity observations and dataset platform

Rimrock and PLGData are released under an open-source license.
Grab your version at https://gitlab.com/cyfronet

http://dice.cyfronet.pl/

Marek Kasztelnik | m.kasztelnik@cyfronet.pl