



This project was funded by the European Union's HORIZON-INFRA-2021- EOSC-01 under Grant Agreement number 101057388.



D4.1 Bring Your Own Infrastructure

Work Package 4. Building blocks for a sustainable operating model.

Technical References

| | |
|----------------------------|-------------------------------------|
| <i>Project Acronym</i> | <i>ESG</i> |
| <i>Project Title</i> | <i>EuroScienceGateway</i> |
| <i>Project Coordinator</i> | <i>Albert Ludwig University</i> |
| <i>Project Duration</i> | <i>September 2022 / August 2025</i> |

| | |
|-------------------------------------|--|
| <i>Document</i> | <i>D4.1 Bring Your Own Infrastructure</i> |
| <i>Work Package</i> | <i>Work Package 4. Building blocks for a sustainable operating model.</i> |
| <i>Task</i> | <i>Task 4.1 Bring Your Own Compute (BYOC) Task 4.2 Bring Your Own Storage (BYOS)</i> |
| <i>Dissemination Level*</i> | <i>PU</i> |
| <i>Lead Beneficiary</i> | <i>EGI</i> |
| <i>Contributing Beneficiary/les</i> | <i>AGH-UST, ALU-FR, EGI, INFN, and VIB</i> |



| | |
|--------------------------------|-------------------------|
| <i>Due Date of Deliverable</i> | <i>31st August 2024</i> |
| <i>Actual Submission Date</i> | <i>26th August 2024</i> |

PU = Public

PP = Restricted to other programme participants (including the Commission Services)

RE = Restricted to a group specified by the consortium (including the Commission Services)

CO = Confidential, only for members of the consortium (including the Commission Services)

| <i>Version</i> | <i>Date</i> | <i>Beneficiary</i> | <i>Authors</i> | <i>Approved</i> |
|----------------|-----------------------|--------------------|---|---------------------------------------|
| <i>#1</i> | <i>19th July 2024</i> | <i>EGI.eu</i> | <i>Maiken Pedersen (UiO), Sanjay Srikakulam (ALU), Paul De Geest (VIB), Enol Fernandez (EGI.eu), Andrea Cristofori (EGI.eu), Sebastian Luna-Valero (EGi.eu)</i> | <i>Sebastian Luna-Valero (EGI.eu)</i> |
| <i>#2</i> | <i>22nd Aug 2024</i> | <i>EGI.eu</i> | <i>Marco Tangaro (CNR), Stefano Nicotri (INFN), Maiken Pedersen (UiO), Enol Fernandez (EGI.eu)</i> | <i>Sebastian Luna-Valero (EGI.eu)</i> |



**Funded by
the European Union**

Acknowledgements

This project was funded by the European Union's HORIZON-INFRA-2021- EOSC-01 under Grant Agreement number 101057388.

Disclaimer

This work may rely on data from sources external to the members of the ESG project Consortium. Members of the Consortium do not accept liability for loss or damage suffered by any third party as a result of errors or inaccuracies in such data. The information in this document is provided "as is" and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and neither the European Community nor any member of the ESG Consortium is liable for any use that may be made of the information.

© Members of the ESG Consortium

Executive Summary

This deliverable presents the activities carried out in tasks 4.1 "Bring Your Own Compute (BYOC)" and 4.2 "Bring Your Own Storage (BYOS)", under Work Package 4 "Building blocks for a sustainable operating model".

The overall goal of tasks 4.1 and 4.2 is to make it easier for Galaxy users to connect their accounts in Galaxy to existing, externally managed compute and storage resources. The benefits are twofold: 1) Galaxy administrators do not need to operate and maintain additional IT infrastructure and 2) Galaxy users get extra capacity to execute workflows that are beyond their assigned quotas in Galaxy.



**Funded by
the European Union**

List of Abbreviations

| Abbreviation | Description |
|--------------|--|
| ARC | A dvanced R esource C onnector |
| BYOC | B ring y our O wn C ompute |
| BYOS | B ring y our O wn S torage |
| ESG | E uro S cience G ateway (this project) |
| FAIR | F indable, A ccessible, I nteroperable, R eusable ¹ , a set of principles for scientific data management that emphasizes best practices and machine readability |
| HPC | H igh P erformance C omputing |
| IAM | I ntity and A ccess M anagement |
| IdP | I ntity P rovider |
| IM | I nfrastucture M anager |
| LRMS | L ocal R esource M anagement S ystem |
| NFS | N etwork F ile S ystem |
| REFEDS | the R esearch and E ducation FED erations group |
| Sirtfi | S ecurity I ncident R esponse T rust Framework for F ederated I ntity |
| TOSCA | T opology and O rchestration S pecification for C loud A pplications |
| VO | V irtual O rganization |
| WLCG | W orldwide LHC C omputing G rid |
| WP | W ork P ackage |

¹ <https://doi.org/10.1038/sdata.2016.18>



Table of contents

| | |
|--|-----------|
| Disclaimer | 3 |
| Executive Summary | 3 |
| List of Abbreviations | 4 |
| 1. Introduction | 6 |
| 2. Identity Providers | 6 |
| 2.1. WLCG IAM | 6 |
| 2.2. EGI Check-in | 7 |
| 3. BYOC: Bring your Own Compute | 9 |
| 3.1. Pulsar | 10 |
| 3.1.1. Automated inclusion of Pulsar endpoints in Galaxy | 11 |
| 3.1.2. Pulsar deployment with IM | 12 |
| 3.2. ARC | 15 |
| 3.2.1. ARC job runner | 16 |
| 3.2.2. ARC deployment with IM | 18 |
| 3.3. DIRAC | 22 |
| 4. BYOS: Bring your Own Storage | 24 |
| 4.1. Cloud Storage (S3) [ALU] | 24 |
| 4.1.1. MinIO | 25 |
| 4.2. Onedata | 28 |
| 4.2.1. Onedata — Galaxy File Source Plugin | 28 |
| 4.2.2. Onedata — Galaxy Object Store | 30 |
| 4.2.3. Onedata — BYOS templates | 31 |
| 4.2.4. NextCloud/WebDAV via Onedata | 32 |
| 5. Summary | 34 |



1. Introduction

This deliverable describes the work carried out in the EuroScienceGateway project to make it easier for Galaxy users to connect their accounts in Galaxy to existing, externally managed compute and storage resources. The benefits are twofold: 1) Galaxy administrators do not need to operate and maintain additional IT infrastructure and 2) Galaxy users get extra capacity to execute workflows that are beyond their assigned quotas in Galaxy.

Achieving the “Bring your Own Infrastructure” goal in Galaxy consisted on the following steps:

- Expanding the user login in Galaxy with new Identity Providers (Section 2) so user credentials can be reused to the external computing resources that the user has access to.
- Adding support for new compute and storage endpoints in Galaxy and streamlining the way users connect and configure them. Section 3 focuses on connecting external compute resources (Pulsar, ARC, and DIRAC) whereas Section 4 details the steps to bring external storage capacity to Galaxy (Object Storage in the cloud and Onedata).

The last section of the deliverable will summarize the activities presented.

2. Identity Providers

Galaxy supports *python-social-auth*² for authentication and authorization with federated identity systems. [Release 4.5.0](#) of this library includes two new backends contributed by the EuroScienceGateway Project: EGI Check-in and WLCG IAM.

With both systems now available in *python-social-auth*, the EuroScienceGateway project has started the implementation of the support in Galaxy, which will allow not just to authenticate with the new identities, but also further simplify the Bring Your Own Compute (BYOC) features of Galaxy so users credentials can be reused to access existing infrastructure computing services without the need for reauthentication.

2.1. WLCG IAM

WLCG IAM³ is the solution to power the next generation Authentication and Authorization infrastructure for the Worldwide LHC Computing Grid (WLCG⁴).

The remote ARC sites require user authentication either via X509 certificates or via tokens from supported identity providers (IdPs). It is up to the ARC system administrator to decide what IdPs are supported, which can be easily configured on the ARC endpoint. One such supported IdP is the WLCG IAM - another could be the EGI Check-in as described in the next section.

² *python-social-auth*: <https://github.com/python-social-auth>

³ WLCG IAM: <https://wlcg.cloud.cnaf.infn.it/>

⁴ WLCG: <https://wlcg.web.cern.ch/>



A backend for the WLCG IAM was integrated into the *python-social-auth* library used by Galaxy for the OIDC authentication (<https://github.com/python-social-auth/social-core/pull/820>). A Galaxy instance can then enable this authentication method (and create an OIDC client for WLCG IAM), which allows the user to log into Galaxy using credentials connected to WLCG (see Figure 2.1.1). The ARC job runner then uses the token refresher mechanism (<https://github.com/galaxyproject/galaxy/pull/15300>) to send a fresh token used for authentication towards the ARC remote computing site.

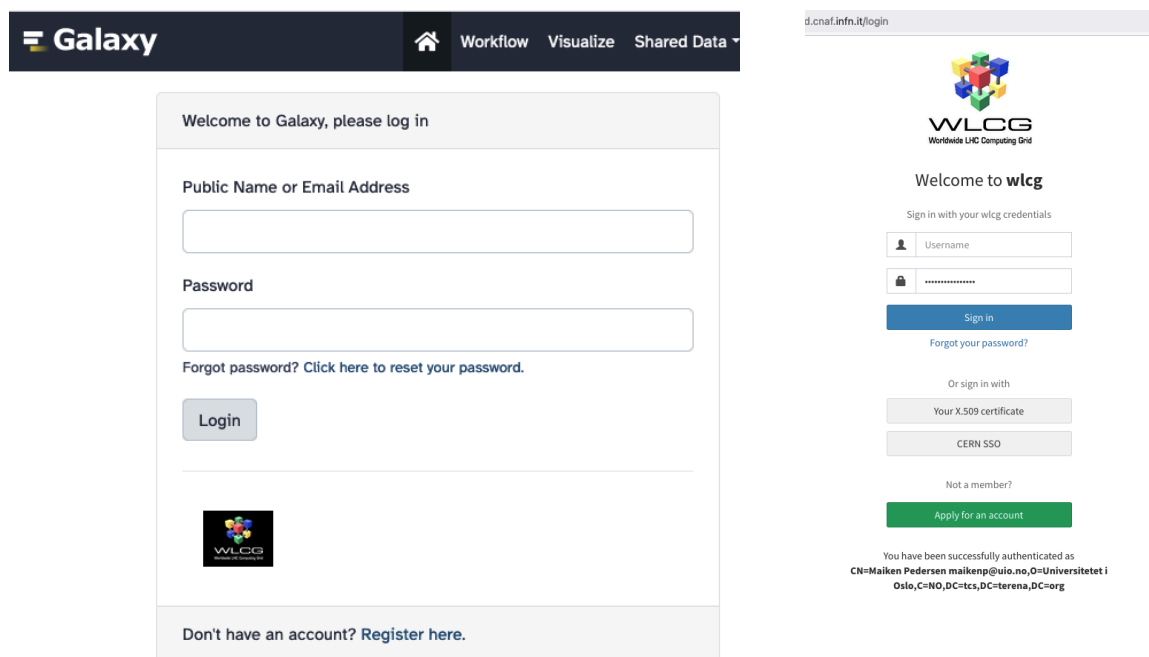


Figure 2.1.1. Screenshots showing how to log into Galaxy with WLCG IAM.

As a result of this work, when a user logs into Galaxy using the WLCG IAM, the same access token can be reused to connect to remote ARC sites without further effort (see Section 3.2 below).

2.2. EGI Check-in

EGI Check-in⁵ brings eduGAIN⁶ and other authentication sources in a REFEDS⁷ and Sirtfi⁸ compliant service that ensures security without compromising productivity. Check-in follows the AARC Blueprint Architecture⁹ and implements existing AAI Interoperability Guidelines, making it compliant with the existing EOSC AAI Federation and ready to interoperate with the rest of the EOSC ecosystem.

⁵ EGI Check-in: <https://www.egi.eu/service/check-in/>

⁶ eduGAIN: <https://edugain.org/>

⁷ REFEDS: <https://refeds.org/>

⁸ Sirtfi: <https://refeds.org/sirtfi>

⁹ AARC Blueprint Architecture (BPA): <https://aarc-community.org/architecture/>



The support for Check-in opens the door to access any existing service in the EGI catalog¹⁰ as for example, the EGI Cloud Compute and Infrastructure Manager for on-demand deployment of Pulsar endpoints (see Section 3.1.2).

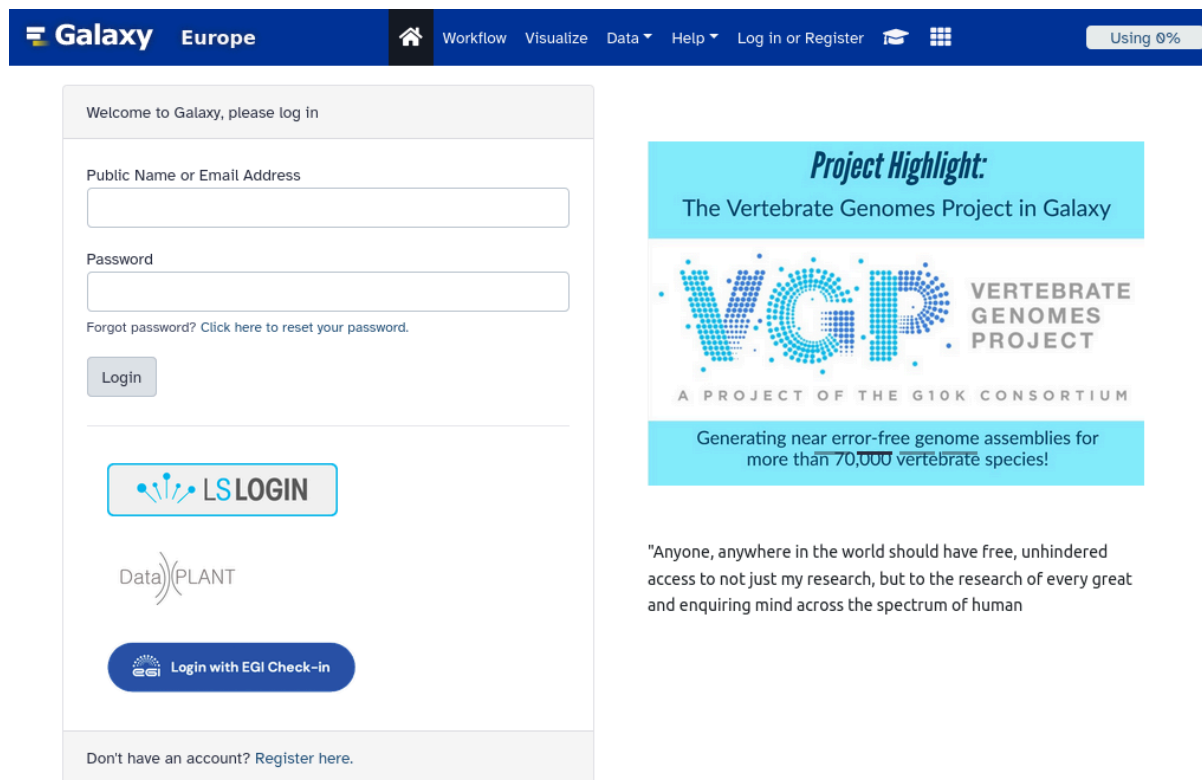


Figure 2.2.1. Screenshot showing the Galaxy login page with the EGI Check-in button.

For instance, figure 2.2.1 shows the EGI Check-in button in the UseGalaxy.eu login page. Users clicking on EGI Check-in will be redirected to the Check-in page (Figure 2.2.2) and presented with multiple options for login. In the ideal scenario, users search and find their home institution and reuse their existing credentials. Alternatively, users can also reuse existing credentials in popular social media services such as Facebook, LinkedIn or Google. Since the introduction of Check-in in May 2024, 87 users have logged in into the UseGalaxy.eu instance.

¹⁰ EGI Services for Research: <https://www.egi.eu/services/research/>



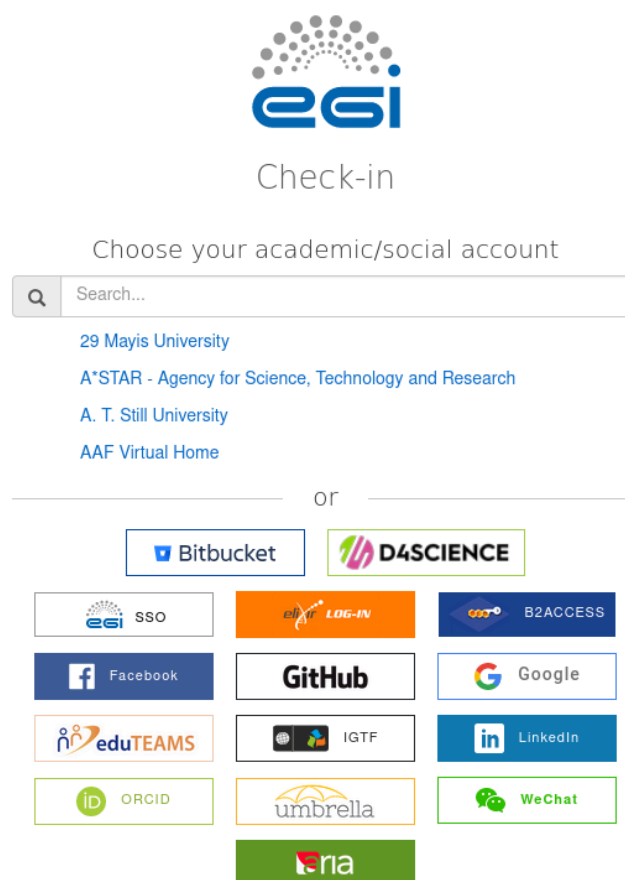


Figure 2.2.2. Screenshot showing the EGI Check-in page.

A popular choice among researchers is the use of ORCID¹¹ credentials, since it allows accessing the same credentials when moving between different job positions across their career. With EGI Check-in, ORCID is one of the multiple options for the users to simplify access across services.

3. *BYOC: Bring your Own Compute*

The goal of Bring Your Own Compute is to allow Galaxy users to make use of computing resources external to Galaxy that they have access to. For example, these resources can be credits in the cloud or sponsored HPC resources at their home institution. The desired steps to bring their own computing resources are these: 1) log into Galaxy, 2) go to user preferences, 3) add external computing endpoint and credentials, 4) select new external endpoint as preferred destination in Galaxy, and 5) execute Galaxy workflows using the new endpoint.

The EuroScienceGateway project has enabled Galaxy users to bring their own computing resources in three different ways: 1) via streamlined deployment of Pulsar endpoints (Section 3.1); 2) via the Advanced Resource Connector (Section 3.2), and 3) via DIRAC (Section 3.3).

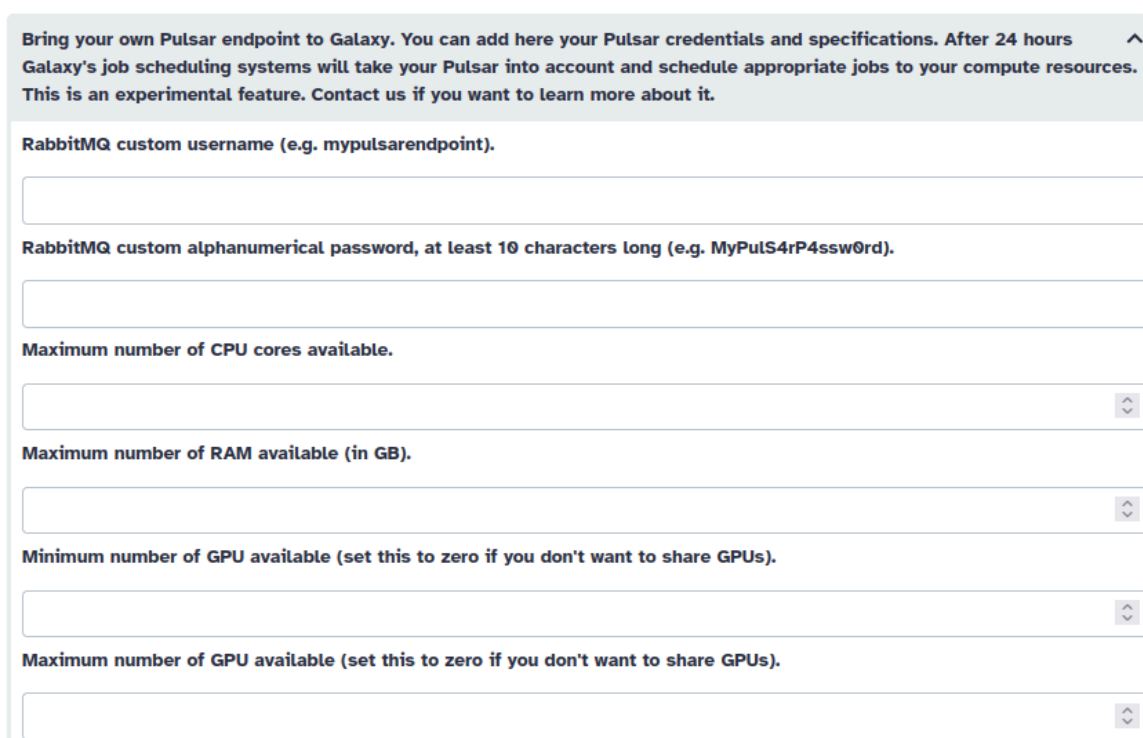
¹¹ ORCID: <https://orcid.org/>



3.1. Pulsar

Pulsar¹² is a Python server application that allows a Galaxy server to run jobs on remote systems (including Windows) without requiring a shared mounted file system. Unlike traditional Galaxy job runners - input files, scripts, and config files may be transferred to the remote system, the job is executed, and the results are transferred back to the Galaxy server - eliminating the need for a shared file system.

As part of the work carried out in the EuroScienceGateway project, a new form has been added to Galaxy under the user preferences menu (see Figure 3.1.1 below) that allows the users to connect external computing resources with Galaxy via a Pulsar endpoint. This form gathers details such as the credentials for the Message Queue (RabbitMQ, used to communicate Pulsar with Galaxy), plus the specifics of their available compute resources such as RAM, CPU cores and GPUs.



Bring your own Pulsar endpoint to Galaxy. You can add here your Pulsar credentials and specifications. After 24 hours Galaxy's job scheduling systems will take your Pulsar into account and schedule appropriate jobs to your compute resources. This is an experimental feature. Contact us if you want to learn more about it.

RabbitMQ custom username (e.g. mypulsarendpoint).

RabbitMQ custom alphanumeric password, at least 10 characters long (e.g. MyPulS4rP4ssw0rd).

Maximum number of CPU cores available.

Maximum number of RAM available (in GB).

Minimum number of GPU available (set this to zero if you don't want to share GPUs).

Maximum number of GPU available (set this to zero if you don't want to share GPUs).

Figure 3.1.1. User preferences menu in Galaxy to Bring Your Own Compute with Pulsar.

Task 3.5 in Work Package 3 has also added another form that allows the user to select the preferred Pulsar endpoint where the user wishes to run their jobs (Figure 3.1.2).

¹² Pulsar: <https://pulsar.readthedocs.io/>



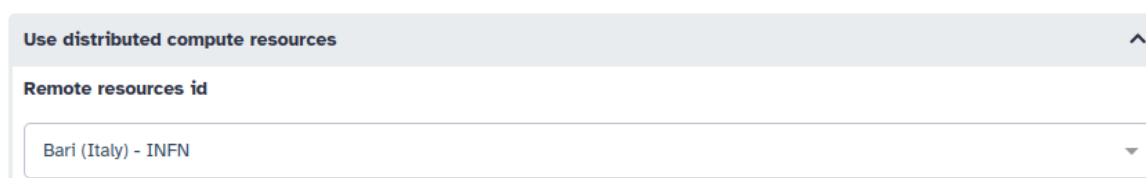


Figure 3.1.2. User preferences menu in Galaxy to select the preferred Pulsar endpoint.

With these two forms the user first connects external compute resources to Galaxy (Figure 3.1.1) and second, requests Galaxy to send his/her jobs to the desired destination (Figure 3.1.2).

3.1.1. Automated inclusion of Pulsar endpoints in Galaxy

Currently, the inclusion of a Pulsar endpoint in Galaxy is made by hand by a Galaxy administrator, submitting a pull request to the Github repository of the infrastructure playbook¹³, containing the information gathered by the form described in the previous section. The pull request must then be approved to finalize the addition.

A script is currently under development¹⁴ to automate the gathering of the information submitted by the user via the form, give it the correct structure to create the pull request and submit it.

The script, written in python, authenticates to the Galaxy instance (which the Pulsar endpoint should be attached to) via an API key (read from a locally stored secrets json file) belonging to a service account created by the administrators of the server specifically for this purpose, and fetches the information submitted by the user, in json form, parsing it and storing it into variables, which are then reformatted in json/yaml suitable to integrate the infrastructure playbook repository with a pull request, which is then automatically generated and submitted.

The script is then fed into a CI/CD server (based on Jenkins¹⁵) which executes it on a periodic basis.

In this way, the procedure of adding a Pulsar endpoint to a Galaxy instance is completely automated and doesn't need the intervention of an administrator, nor any contact and exchange of credentials with the user, in any step. All the sensitive information is encrypted using ansible vault, using a key stored only in the local secrets file described above (which can be generated automatically). The only step which is left not automated (by design) and needs an action by an administrator is the approval of the pull request.

Using the architecture described above, the system is usable *as is* in any Galaxy instance, and needs only the administrative application credentials in order to be executed.

¹³Infrastructure playbook Github repository: <https://github.com/usegalaxy-eu/infrastructure-playbook/>

¹⁴https://github.com/stefanonicotri/automatic_pulsar_endpoint_addition

¹⁵ Jenkins: <https://www.jenkins.io/>



3.1.2. Pulsar deployment with IM

In an attempt to streamline the Bring Your Own Compute approach, work has been carried out to deploy HTCondor clusters¹⁶ preconfigured with Pulsar on the EGI Federated Cloud¹⁷ using the Infrastructure Manager¹⁸ (or IM). With this, end users with limited technical expertise just need a web browser and a few clicks to connect Galaxy with external computing resources on the cloud. There is an accompanying news item¹⁹ with a video recording for the end user to get started.

Below are the steps that a user needs to follow to make use of computing resources in the EGI Federated Cloud. A similar flow (steps 3 to 7) should be followed with other cloud providers:

1. Create an EGI Check-in account. Check out the documentation²⁰.
2. Enroll²¹ in a Virtual Organization (VO) with enough quota for HTCondor deployments.
3. Upload the VGCN²² Virtual Machine Image (built by WP3) to the cloud provider.
4. Visit <https://im.egi.eu/> and configure credentials in IM (Figure 3.1.2.1).
5. Select HTCondor and Pulsar in Infrastructure Manager (figures 3.1.2.2 and 3.1.2.3)
6. Configure HTCondor and Pulsar with Infrastructure Manager (Figure 3.1.2.4).
7. Connect Pulsar with Galaxy (see Section 3.1.1 above).

Once the EGI Check-in account has been created and the membership to a Virtual Organization has been approved, the user needs to configure the credentials²³ in Infrastructure Manager (see Figure 3.1.2.1).

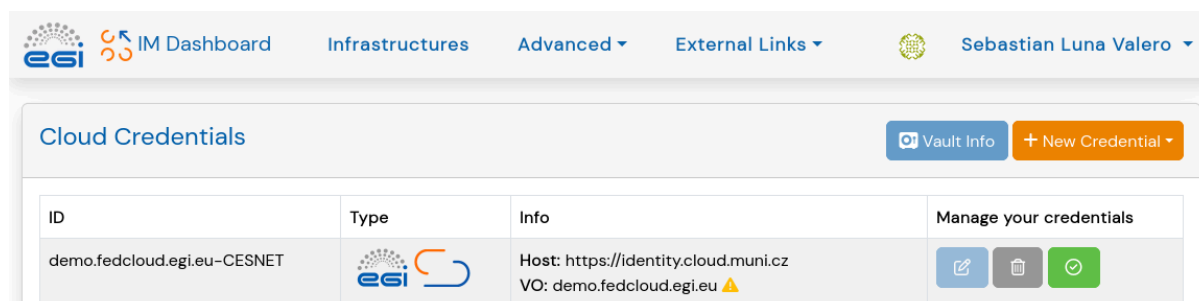


Figure 3.1.2.1. Configure credentials²⁴ in Infrastructure Manager

Out of all the options available in Infrastructure Manager, the user needs to click on HTCondor first (Figure 3.1.2.2), and “add” Pulsar later (Figure 3.1.2.3). Although Pulsar supports other LRMS systems (e.g. Slurm²⁵), HTCondor is the default option, since it has been used and

¹⁶ HTCondor: <https://htcondor.org/>

¹⁷ EGI Federated Cloud: <https://www.egi.eu/service/cloud-compute/>

¹⁸ Infrastructure Manager: <https://www.egi.eu/service/infrastructure-manager/>

¹⁹ News item: <https://galaxyproject.org/news/2023-10-31-esg-byoc-im/>

²⁰ Create an EGI Check-in account: <https://docs.egi.eu/users/aai/check-in/signup/>

²¹ Enroll in a Virtual Organization: <https://docs.egi.eu/users/aai/check-in/joining-virtual-organisation/>

²² VGCN: <https://github.com/usegalaxy-eu/vgcn>

²³ [Create credential in IM](#)

²⁴ See: <https://docs.egi.eu/users/compute/orchestration/im/dashboard/#cloud-credentials>

²⁵ Slurm: <https://slurm.schedmd.com/documentation.html>



tested by several European pulsar endpoints. Therefore, at the moment, the only way to deploy a Pulsar endpoint with Infrastructure Manager is with HTCondor.

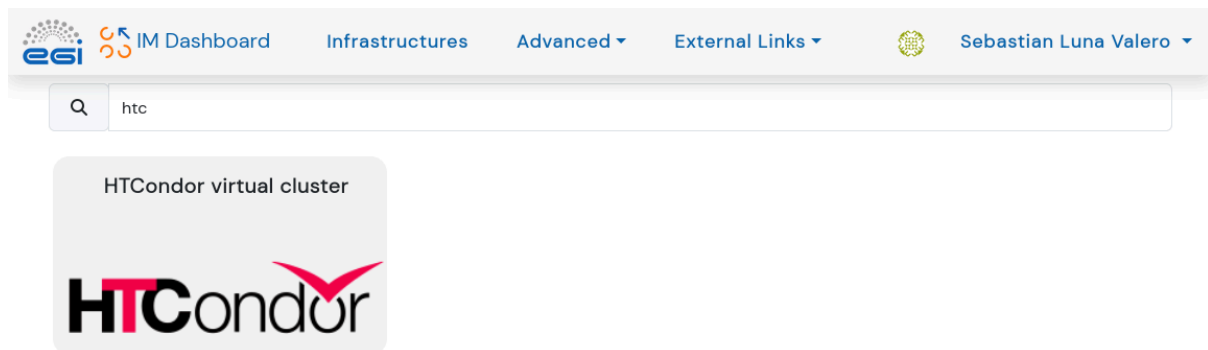


Figure 3.1.2.2. Select the HTCondor option in Infrastructure Manager

After “adding” Pulsar in Infrastructure Manager, the user should click on “Configure” (Figure 3.1.2.3) to move on with the deployment.

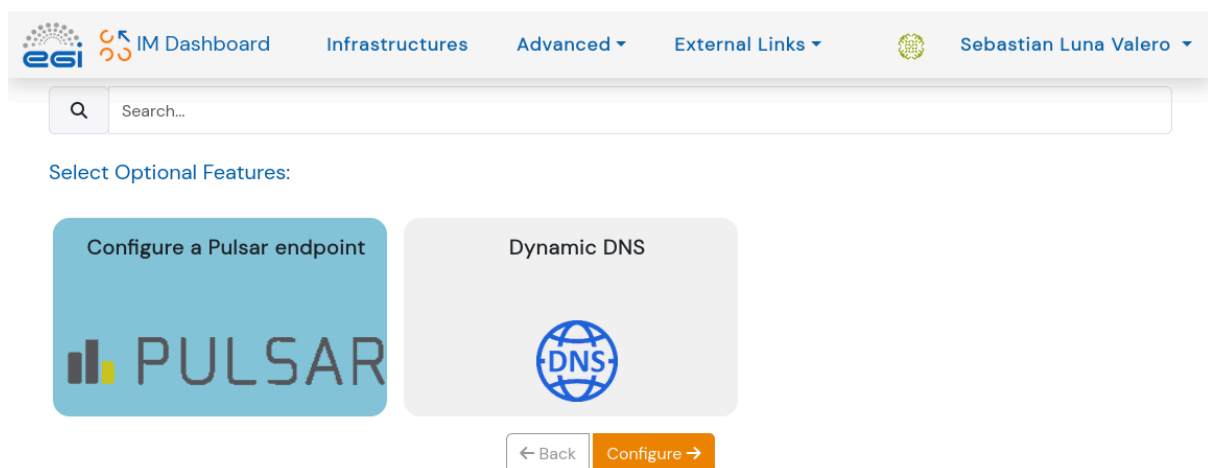


Figure 3.1.2.3. Select the Pulsar option in Infrastructure Manager

Before clicking “Submit” to start the deployment (Figure 3.1.2.4), the user needs to fill out information about:

- Number of CPUs, RAM for the front-end node of the HTCondor cluster
- Number of CPUs, RAM for each worker node of the HTCondor cluster
- HTCondor credentials for secure communication between HTCondor nodes
- Pulsar credentials to communicate with Galaxy
- Cloud Provider and Virtual Machine Image. Here the user must choose the VGCN image provided by WP3, that should have been previously uploaded to the cloud.

The screenshot shows the 'HTCondor virtual cluster + Pulsar' configuration form. At the top, there's a description: 'TOSCA for launching an HTCondor Virtual Cluster. TOSCA for configuring a Pulsar endpoint'. Below this is a text field for 'Infrastructure Name' with the value 'description'. There are two tabs: 'Pulsar configuration' (active) and 'Cloud Provider Selection'. Under 'Pulsar configuration', there are fields for 'User for the message queue' (value: 'changeme') and 'Password for the message queue' (masked with dots). At the bottom are 'Submit' and 'Cancel' buttons.

Figure 3.1.2.4. Configure the deployment of Pulsar with HTCondor

All going well the deployment completes as shown in Figure 3.1.2.5. Infrastructure Manager shows: 1) the name of the deployment, 2) a unique ID for it, 3) the cloud where the deployment happened (with additional information about the specific site and VO, in the case of the EGI Federated Cloud), 4) status of the deployment, 5) information about the number of Virtual Machines deployed, and 6) the *Actions* menu.

The screenshot shows the 'My Infrastructures' table. It has a search bar with 'pulsar' and a table with the following data:

| Name | Infrastructure uuid | Cloud Type | Cloud Info | Status | VMs | Actions |
|--------|--------------------------------------|------------|---|------------|-------|---------|
| pulsar | bf2f2e38-73dd-11ee-a461-6e5e8122d1e3 | | Site: IISAS-FedCloud VO: vo.usegalaxy.eu | configured | 0 1 2 | Outputs |

Below the table, it says 'Showing 1 to 1 of 1 entry (filtered from 9 total entries)'.

Figure 3.1.2.5. Pulsar deployment completed.

Clicking on the *Outputs* button (below *Actions*, Figure 3.1.2.5) in Infrastructure Manager provides the following information: username for SSH, public SSH key and public IP address to



connect via SSH. For more information about possible actions, please refer to the documentation²⁶.

In order to make HTCondor and Pulsar deployable with the Infrastructure Manager three steps had to be carried out: 1) make sure there is a suitable Ansible²⁷ role, 2) TOSCA²⁸ types are properly defined, and 3) TOSCA templates²⁹ are configured to tight everything together for Infrastructure Manager.

The Ansible role for Pulsar was created in [PR#1](#) and later [improved](#) but assuming that the VGNC Virtual Machine Image is used for the deployment. This way, the Ansible role for Pulsar only needs to set the right credentials to access the Message Queue in Galaxy and restart the Pulsar service accordingly for the changes to take effect. On the other hand, in the case of the Ansible role to deploy HTCondor the effort was focused on updating an existing role to support HTCondor versions 10.0, 10.x, 23.0, and 23.x for the following list of Operating Systems: Ubuntu 20.04 and 22.04, Red Hat Enterprise Linux 7, 8 and 9 (see [PR#10](#), [PR#12](#), and [PR#16](#)). Further testing raised the need to explicitly open the Network File System (NFS³⁰) port in firewalld³¹ on [PR#13](#).

TOSCA types and artifacts for HTCondor were added on [PR#110](#) and for Pulsar on [PR#112](#). TOSCA templates for Infrastructure Manager were added for HTCondor on [PR#110](#) and Pulsar on [PR#444](#). Altogether this makes it possible for the user to deploy HTCondor and Pulsar with a few clicks just using a web browser as shown in Figures 3.1.2.2 to 3.1.2.5 above.

A requirement was raised by the EuroScienceProject where the Pulsar credentials in the Message Queue of Galaxy may need to be reestablished for different reasons. Initially, Infrastructure Manager did not allow to reconfigure Pulsar credentials after deployment, but the suggestion was raised in issue [#484](#) and solved in [PR#135](#). This is bringing a new feature to Infrastructure Manager which may be also interesting to other deployments in other projects, so it is worth mentioning it here as an example of co-design and improvement for the service.

3.2. ARC

In the scientific domain of high energy particle physics, the Worldwide LHC Computing Grid (WLCG) was created in order to handle the huge compute and storage needs produced by the experiments at LHC. WLCG combines about 1.4 million computer cores and 1.5 exabytes of storage from over 170 sites in 42 countries. What ties the sites together is the middleware

²⁶ Actions in Infrastructure Manager:

<https://docs.egi.eu/users/compute/orchestration/im/dashboard/#list-of-actions>

²⁷ Ansible role: <https://github.com/grycap/ansible-role-htcondor>

²⁸ TOSCA types for [HTCondor](#) and [Pulsar](#)

²⁹ TOSCA templates for [HTCondor](#) and [Pulsar](#)

³⁰ Network File System: https://en.wikipedia.org/wiki/Network_File_System

³¹ firewalld: <https://firewalld.org/>



installed in front of each site, one of these being Nordugrid ARC³² (Advanced Resource Connector).

The EuroScienceGateway project is integrating ARC with Galaxy in two ways: 1) developing a Galaxy Job Runner for ARC (Section 3.2.1) and streamlining the deployment of ARC in the EGI Federated Cloud with Infrastructure Manager (Section 3.2.2)

3.2.1. ARC job runner

The Galaxy Job Runner for ARC uses the newly developed (in context of Nordugrid ARC) ARC Python REST client to do the actual communication with the remote ARC computing site. The ARC sites will be a new flavor of the already existing Pulsar sites.

As part of the work carried out in the EuroScienceGateway project, a new form has been added to Galaxy under the user preferences menu (see Figure 3.2.1.1 below) that allows the users to connect external computing resources with Galaxy via ARC. This form only gathers the URL to connect Galaxy with the remote ARC endpoint.



Figure 3.2.1.1 User preferences menu in Galaxy to select the preferred ARC endpoint.

A first prototype of the ARC job runner has been drafted ([PR#16653](#)) to demonstrate that from within Galaxy, a user can send a job from the Galaxy web interface to a remote ARC computing site, and receive the outputs back into Galaxy once the job is done (Figure 3.2.1.2 and Figure 3.2.1.3 below).

One of the key components of ARC is its data staging and caching capabilities. If the job requires input data that is not local to the Galaxy server (and ARC client), ARC - on the remote server side - fetches this input data on behalf of the job (and caches it for possible reuse later on). A demo tool was written in order to demonstrate the basic functionalities needed to submit a job to an ARC compute site, including the remote data staging, as can be seen in the figures 3.2.1.2 and 3.2.1.3 below. In the demo tool the URLs of the remote input data are listed in the file "remote_list.txt", which in this case is handled as an input file to the Galaxy job. The ARC job runner provides the necessary job description that the remote ARC site needs. It is constructed from the input provided in the Galaxy tool interface, in addition to the hard-coded parameters set for the different runner destinations (like the number of CPU hours and amount of memory the jobs are allowed to allocate). This is in turn sent to the remote ARC server by the ARC REST client so that the job can be run on the remote sites underlying batch system. The ARC client on the Galaxy server uploads any local input files to the remote ARC

³² ARC: <https://www.nordugrid.org/arc/arc7>



server as part of the job submission routine. ARC on the server side is responsible for downloading any remote input files (like the ones listed in the demo tools “remote_list.txt” file) as already explained.

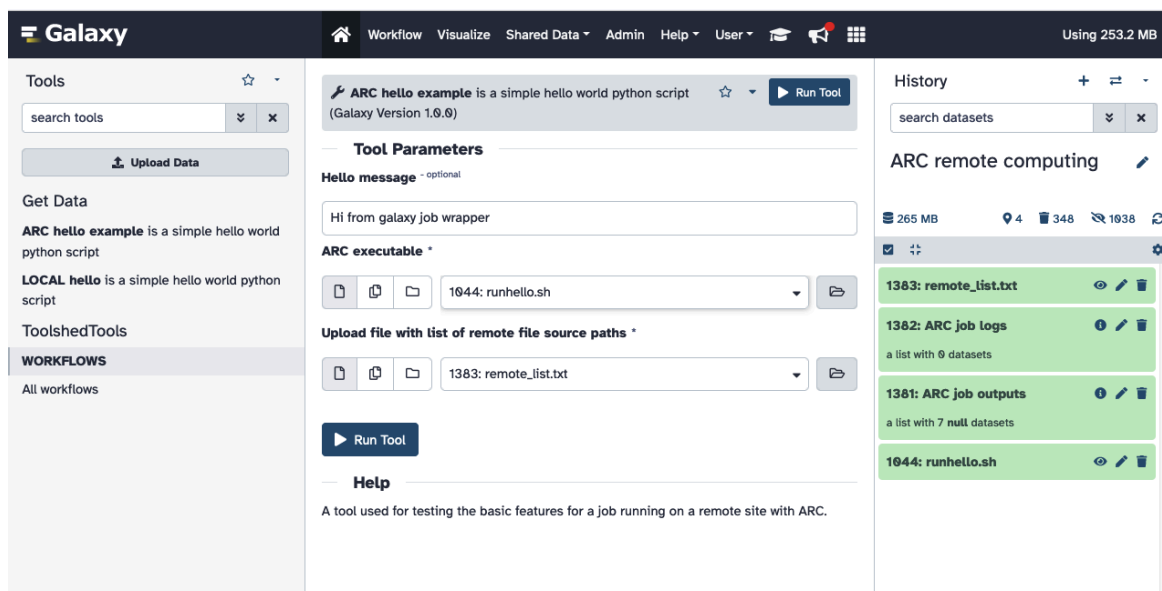


Figure 3.2.1.2. Demo tool to show how to send a “hello world” job to ARC via Galaxy.

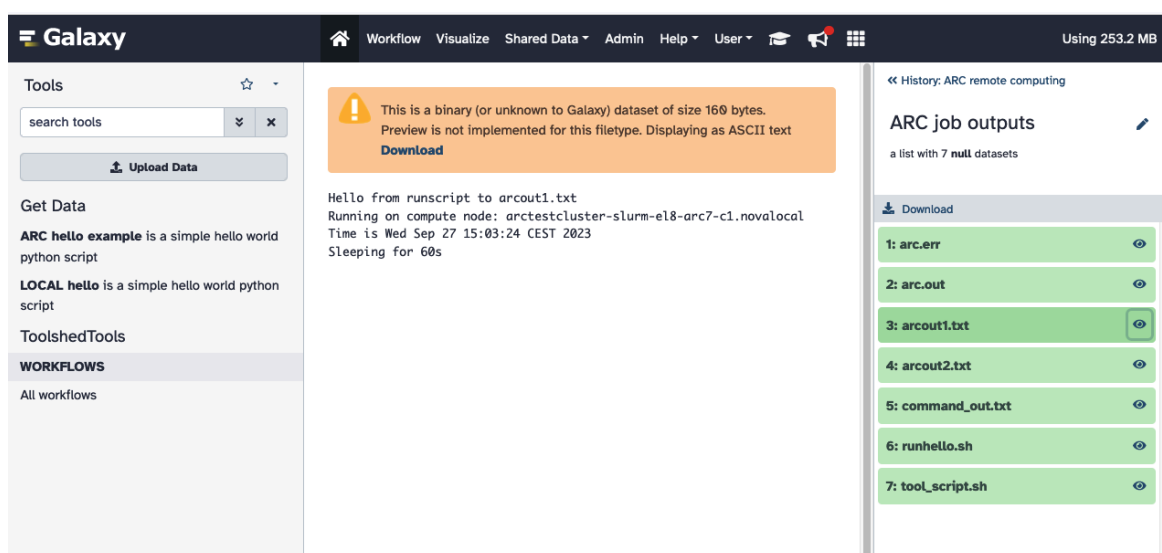


Figure 3.2.1.3. Output returned by the demo tool once a job has been submitted to ARC.

The current prototype implementation uploads the whole working directory of the ARC job to Galaxy once the job is done. This means that all input and output files will be uploaded to Galaxy history, in addition to the ARC logs produced for this job. For the final version of the

ARC job runner, this will instead incorporate Galaxy's filtering of outputs defined in the relevant Galaxy tool.

The remote ARC sites require user authentication either via X509 certificates or via tokens from supported identity providers (IdPs). One such supported IdP is the WLCG IAM. See Section 2.1 above explaining how WLCG IAM was integrated into Galaxy.

Ongoing and future work

For the ARC job runner to be fully functional there are several components that still need to be finalized.

The ARC job runner can as of now not handle generic Galaxy tools. This is because the internal handling of file paths needs to be implemented for remote job submission. Only parts of this are in place for simple jobs as the prototype job shown above. The full path handling will be implemented by applying the existing PULSAR path handling to the ARC job runner. The same goes for the handling of output files. In Galaxy, a tool can implement complex filtering and logic with respect to what output files should be made available to the user once the job is done. It is the Galaxy components themselves that do this job, but when the job runs remotely, this is not convenient. The solution to handle input and output files correctly in Galaxy for ARC remote job submission is well understood, and work is ongoing ([offline-collector branch](#)).

Furthermore, as submission to an ARC remote site requires an access token, and in the case that a user has access to several ARC remote sites (for instance two HPC centers), it would be convenient that the user easily can obtain access tokens dynamically in Galaxy depending on what ARC site the job is sent to. This is because the two sites may be configured differently only allowing tokens from certain IdPs. How to allow this in Galaxy is currently being discussed with the Galaxy developers.

Finally, the way ARC sites are managed in Galaxy needs to improve. For instance: a user wishing to submit to an ARC site, should be able to select the site from a dynamic drop-down list of ARC sites. In the current implementation the user can only configure a single site at a time through the user preferences as shown above. The user should be able to configure several sites, and be able to choose one of the preconfigured sites during the submission step or before. There should also be a possibility for the Galaxy admin to add ARC sites available for a certain group of users, or add publically available ARC sites. The sites added by the Galaxy admin should be presented to the user for selection too. In addition, the TPV should be able to take into account ARC sites when scheduling jobs.

3.2.2. ARC deployment with IM

Users with access to cloud resources in the EGI Federated Cloud can also benefit from the available TOSCA template in Infrastructure Manager (or IM) for the automated deployment of ARC. Please refer to Section 3.1.2 to learn more details about Infrastructure Manager and how



to configure the credentials. The steps below assume the following: 1) the user has created an EGI Check-in account, and 2) is a member of a Virtual Organization with cloud resources to deploy ARC.

These are the steps for the user:

1. Visit <https://im.egi.eu/> and login using EGI Check-in.
2. Select Slurm and ARC in Infrastructure Manager.
3. Configure Slurm and ARC with Infrastructure Manager.
4. Connect ARC with Galaxy (see Figure 3.2.1.1)

IM will deploy a cluster of virtual machines with Slurm and ARC in the login node. Therefore, the first step for the user is to select "Slurm" in IM (see Figure 3.2.2.1)

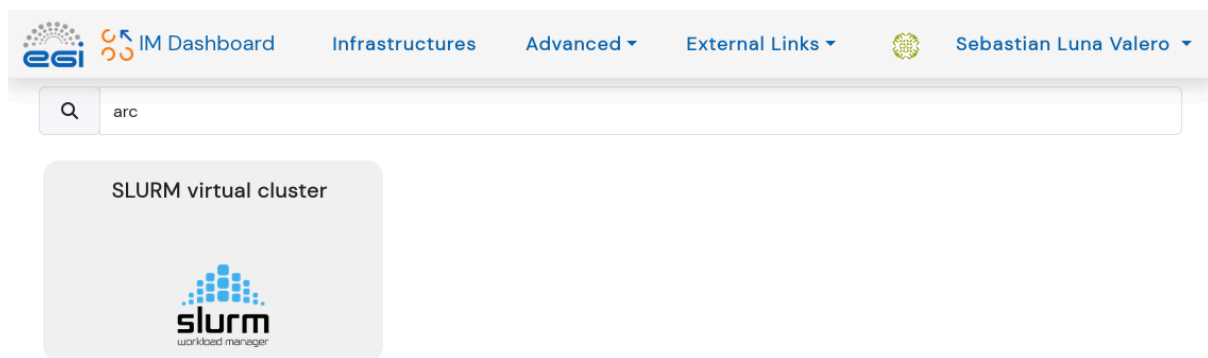


Figure 3.2.2.1. Select the Slurm option in Infrastructure Manager

After selecting the option "Slurm" in IM, the user is presented with several options to deploy services on top of Slurm. Figure 3.2.2.2 below shows the ARC option selected. The user needs to click on "Add" and then "Configure".



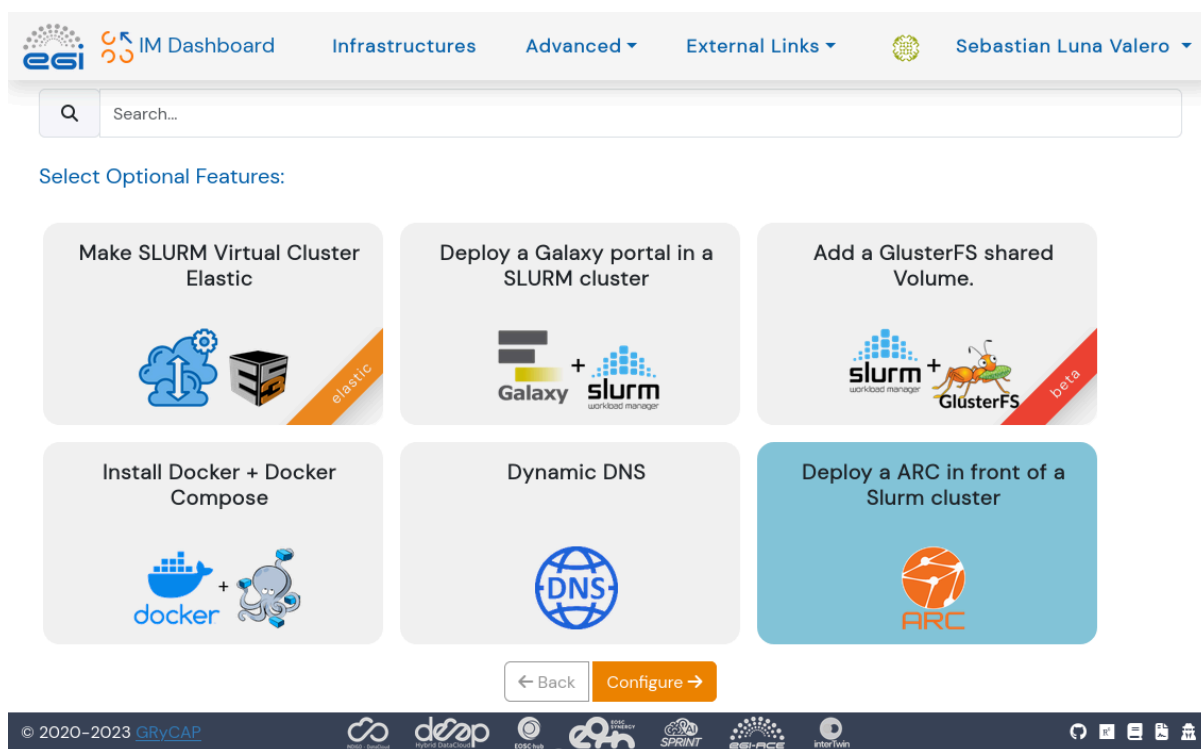


Figure 3.2.2.2. Select the ARC option in Infrastructure Manager

In the configuration page for the deployment in IM (see Figure 3.2.2.3) the user needs to enter:

- Number of CPUs, RAM for the front-end node of the Slurm cluster
- Number of CPUs, RAM for each worker node of the Slurm cluster
- Version of Slurm to install
- Timezone for ARC
- Cloud Provider and Virtual Machine Image



SLURM virtual cluster + ARC

Description: Deploy a SLURM Virtual Cluster.
Deploy a ARC alongside a Slurm cluster

Infrastructure Name
arc

FE Node Features WNs Features SLURM Features **ARC Features**

Cloud Provider Selection
Europe/Oslo

See https://en.wikipedia.org/wiki/List_of_tz_database_time_zones

Submit Cancel

Figure 3.2.2.3. Configure the deployment of ARC with Slurm

All going well the deployment completes as shown below in Figure 3.2.2.4. Infrastructure Manager shows: 1) the name of the deployment, 2) a unique ID for it, 3) the cloud where the deployment happened (with additional information about the specific site and VO, in the case of the EGI Federated Cloud), 4) status of the deployment, 5) information about the number of Virtual Machines deployed, and 6) the Actions menu.

| Name | Infrastructure uuid | Cloud Type | Cloud Info | Status | VMs | Actions |
|------|--------------------------------------|------------|---|------------|-------|---------|
| arc | 48159948-168c-11ef-91ef-5a3710828b91 | ESI | Site: CESNET-MCC VO: training.egi.eu | configured | 0 1 2 | Outputs |

Figure 3.2.2.4. Status of deployment in Infrastructure Manager

Clicking on the Outputs button (see Figure 3.2.2.4) in Infrastructure Manager provides the following information: username for SSH, public SSH key and public IP address to connect via SSH. For more information about possible actions, please refer to the documentation³³.

³³ [List of actions for deployments with Infrastructure Manager](#)



In order to allow ARC to be deployable with the Infrastructure Manager two steps had to be carried out: 1) make sure there is a suitable Ansible role³⁴, and 2) TOSCA types and templates are defined³⁵ to tight everything together for Infrastructure Manager.

To submit jobs to the ARC site through a Galaxy instance, the Galaxy sys-admin can enable the ARC job-runner, and the end-user can configure his user preferences with the ARC endpoint in order to submit jobs to this remote ARC site, as was discussed in Section 3.2.1. Assuming the ARC admin has configured ARC to trust tokens submitted by EGI Check-in, the user can log into Galaxy with EGI Check-in and will then automatically be using an access token from EGI Check-in for authentication towards the ARC remote site.

3.3. DIRAC

DIRAC³⁶ is a meta-scheduler that allows users to execute many computing jobs (in the order of thousands) across multiple sites geographically distributed in different locations, initially developed in the context of the CERN LHCb experiment.

In DIRAC every High Throughput Computing cluster (or site) is called a Compute Element (CE). Through the Pilot Factory DIRAC sends Pilot Jobs to every compute node available across all sites for each configured community (Virtual Organisation in DIRAC terms). The goal of a Pilot Job is to grab a computing node and present its properties (number of cores, RAM available, operating system, etc.) to the DIRAC Matcher. Once the Pilot Job has occupied a computing node it becomes a Pilot Resource.

On the other hand, user jobs are queued in DIRAC's Central Task Queue. The DIRAC Matcher inspects and marks jobs in the Central Task Queue as eligible and then all Pilot Resources compete to pull eligible jobs for their execution. DIRAC's pull scheduling policy allows for the optimal allocation of thousands of user jobs across the available computing resources and is widely used at EGI. The goal of this task is to write a Galaxy JobRunner that allows Galaxy to submit jobs to DIRAC to bring together the best of both worlds and expand the computational capacity of Galaxy instances with a wide range of distributed High Throughput Computing clusters.

The Galaxy Job Runner³⁷ for DIRAC relies on the Python DIRAC Client³⁸, that uses the DIRAC API to the DIRAC server. Although DIRAC has support for EGI Check-in, users require a valid and recognised X.509 certificate for the authentication and submission of jobs. Obtaining and managing such certificates is not a trivial task for most users. The Job Runner for DIRAC is designed to avoid the need to manage certificates for final users and relies on a globally

³⁴ Ansible role to deploy ARC: <https://github.com/nordugrid/arc-ansible/>

³⁵ TOSCA types and templates defined for ARC: <https://github.com/grycap/tosca/pull/147>

³⁶ DIRAC: <https://dirac.readthedocs.io>

³⁷ <https://github.com/enolfc/galaxy/tree/dirac-jobrunner>

³⁸ Using DIRAC from Python:

<https://dirac.readthedocs.io/en/latest/UserGuide/Tutorials/UsingDIRACFromPython/index.html>



configured service identity that will have access to the DIRAC Pilot Job Factory of the community supported by the Galaxy instance. In the case of EuroScienceGateways, two Virtual Organisations were used for tests: biomed, a well established Virtual Organisation with access to 40 Million CPU Hours yearly in the EGI infrastructure for Life Sciences research and dteam, a VO for access to limited allocation for development purposes. For production usage, dedicated VOs can be created to support the execution of jobs of the Galaxy instances. As the JobRunner for DIRAC is configured globally, it can't be configured as a new endpoint in the user preferences menu (see the example of ARC in Section 3.2.1), but has to be managed by the Galaxy instance administrators instead.

DIRAC stages any input and output data needed for job execution as declared in the job definition. The Job Runner for DIRAC implementation includes as part of the input data: any needed tool files, the job input data and additional control files and scripts to ensure the job is executed as expected by Galaxy. The job input is transferred from the Galaxy instance to DIRAC on job submission and fetched by the Pilot Factory in turn once the job is allocated to it. Upon completion, the Pilot Job Factory will stage back the output to DIRAC and as soon as the Job Runner detects the job has finalised it will in turn transfer it to the Galaxy instance. The output includes any output files from the tool and several control files that mimic the structure of the local Job Runner to facilitate code reuse in the implementation of the new Runner. Large inputs and outputs not fitting the job sandbox can be handled by the DIRAC Storage Element framework, although this is not available in the current prototype.

Ongoing and future work

Similarly to the ARC job runner, the DIRAC Job Runner misses key functionality to be production ready and needs further work to be made generally available.

Most Galaxy tools require either a Galaxy installation or external binaries to be available. In the case of DIRAC, the installation of additional software at the remote computing resources is heavily restricted and currently only simple tools leveraging commonly available binaries are able to run without issues (e.g. tools using perl). The use of containers for the execution of tools would greatly simplify the execution of any tool as containers can be executed on any resources and can be dynamically pulled as needed during runtime.

The input staging capability of the Job Runner has not been extensively tested. Full path handling using the existing PULSAR as described above for ARC future work should be used instead. Similarly to the output handling.

For a complete Bring Your Own Compute experience, and allow for user-level configuration of the computing resources, DIRAC needs to support tokens instead of certificates. This is under implementation by the DIRAC development team and requires a considerable migration of the infrastructure to support the new authentication mechanism, which is not expected to happen in the short term. Hence the JobRunner will remain as an admin controlled feature for the time being.



4. *BYOS: Bring your Own Storage*

The Bring Your Own Storage (BYOS) feature in the Galaxy platform is an innovative concept similar to the Bring Your Own Compute. It aims to give users greater flexibility and capacity by leveraging external storage resources. BYOS allows users to integrate storage solutions that are not inherently part of Galaxy but to which they have access, thereby expanding their storage options.

Bring Your Own Storage was initially conceived to expand Galaxy's user storage capacity by connecting with external storage solutions in the cloud (Section 4.1). Thanks to the EuroScienceGateway project, Galaxy is also exploring an additional alternative using Onedata (Section 4.2).

4.1. *Cloud Storage (S3)*

A long-time goal of the core development team, since before the inception of the EuroScienceGateway project, was to enable users to bring their own storage to publicly hosted Galaxy instances. This feature has now been implemented by a larger community effort through a series of pull requests:

- Empower Users to Bring Their Own Storage:
<https://github.com/galaxyproject/galaxy/pull/18127>
 - This PR introduces the foundational functionality required for users to integrate their external storage systems with a Galaxy instance.
- Empower Users to Select Storage Destination:
<https://github.com/galaxyproject/galaxy/pull/14073>
 - This pull request made further enhancements, allowing users to choose their preferred storage destinations for different use cases.
- HashiCorp Vault abstraction for Galaxy:
<https://github.com/galaxyproject/galaxy/pull/12940>
 - Security and credential management are critical components for integrating external storage solutions. This pull request adds the vault abstraction for Galaxy, supports different vault backends, and ensures that users' storage credentials are securely managed.
- Storage dashboard: <https://github.com/galaxyproject/galaxy/pull/13113>, and <https://github.com/galaxyproject/galaxy/pull/17500>
 - These add the new storage dashboard feature to explore and visualize disk/quota usage. These also offer the tools to easily recover space or manage data.

Steps to add your own storage to Galaxy and visualize/explore the storage usage through a dashboard

1. Log into Galaxy
2. Navigate to User Preferences



**Funded by
the European Union**

3. Select Manage Your Storage Locations
4. Click on the Create button, select your desired storage location template from the available list (Figure 4.1.1), fill in your storage details (Figures 4.1.2 and 4.1.3), and submit the form.
5. Once the form is submitted, users have several options to select the new storage
 - a. At the tool execution level (Figure 4.1.4)
 - b. At the history level (Figure 4.1.5)
 - c. At the workflow invocation level (Figure 4.1.6)
 - d. Or as your preferred default storage for your user account (Figure 4.1.7)
6. The new storage dashboard (Figure 4.1.8) can be used to explore storage usage (Figure 4.1.9) and to perform storage management

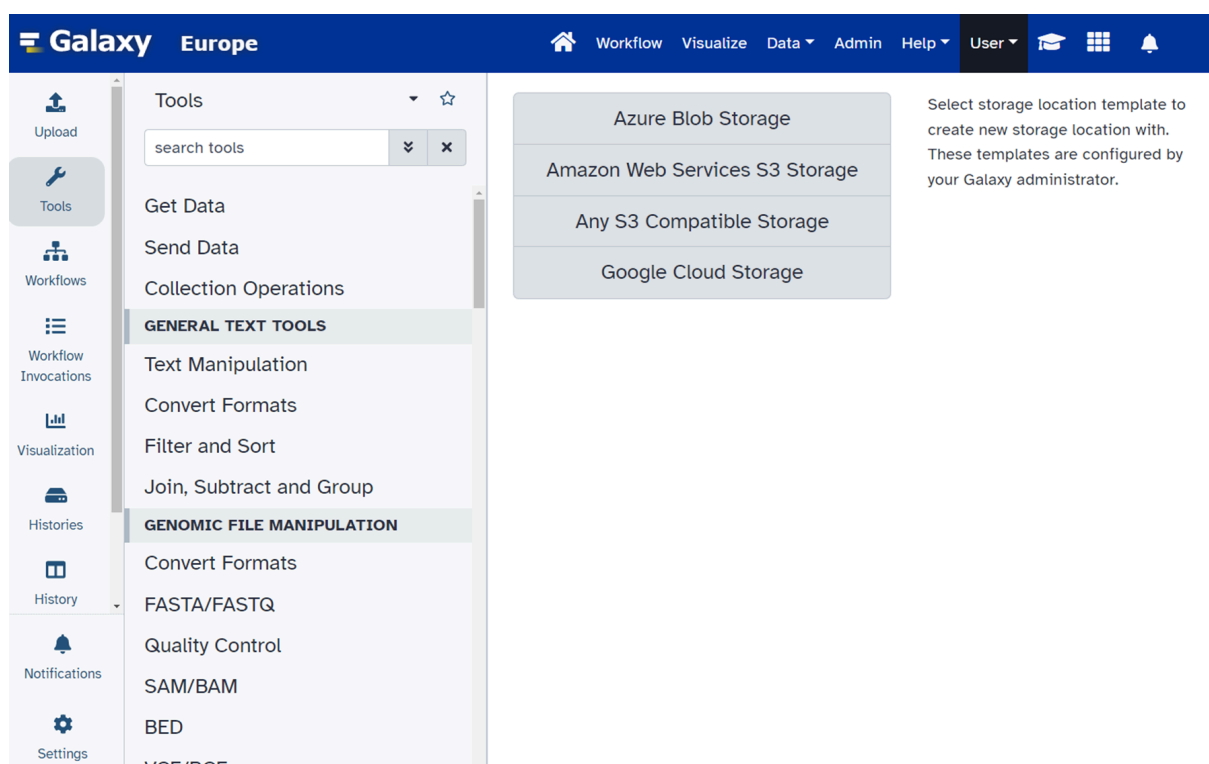


Figure 4.1.1: Multiple storage backend templates that can be leveraged to bring/add your storage



Create a new storage location for your data

Name *
Galaxy S3

Label this new storage location with a name.

Description - optional
My object store for all my data analysis needs in Galaxy

Provide some notes to yourself about this storage location - perhaps to remind you how it is configured, where it stores the data, etc..

Access Key ID
galaxy_eu_001

The less secure part of your access tokens or access keys that describe the user that is accessing the data. The [Amazon documentation] calls these an "access key ID", the CloudFlare documentation describes these as `aws_access_key_id`. Internally to Galaxy, we often just call this the `access_key`.

Bucket
galaxy_eu_001

The bucket to store your datasets in. How to setup buckets for your storage will vary from service to service but all S3 compatible storage services should have the concept of a bucket to namespace a grouping of your data together with.

S3-Compatible API Endpoint
https://usegalaxy-eu-byos.cloud

If the documentation for your storage service has something called an `endpoint_url`, For instance, the CloudFlare documentation describes its endpoints as `https://<accountid>.r2.cloudflarestorage.com`. Here you would substitute your CloudFlare account ID into the endpoint url and use that value. So if your account ID was `galactian`, you would enter `galactian.r2.cloudflarestorage.com`. The MinIO documentation describes the endpoint URL for its Play service as `https://play.min.io:9000`, this whole value would be entered here.

Secret Access Key
.....

The secret key used to connect to the S3 compatible storage with for the given access key.

The [Amazon documentation] calls these an "secret access key" and the CloudFlare documentation describes these as `aws_secret_access_key`. Internally to Galaxy, we often just call this the `secret_key`.

Submit

Figure 4.1.2: An example showing the form to add a generic S3 storage endpoint. In this example, we are adding an S3 storage system named Galaxy S3.

Updated storage location Galaxy S3

Name **Description** **Type** **From Template**

| Name | Description | Type | From Template |
|-----------|--|------|---------------------------|
| Galaxy S3 | My object store for all my data analysis needs in Galaxy | S3 | Any S3 Compatible Storage |

+ Create

Figure 4.1.3: Showing a list of object stores that were added.



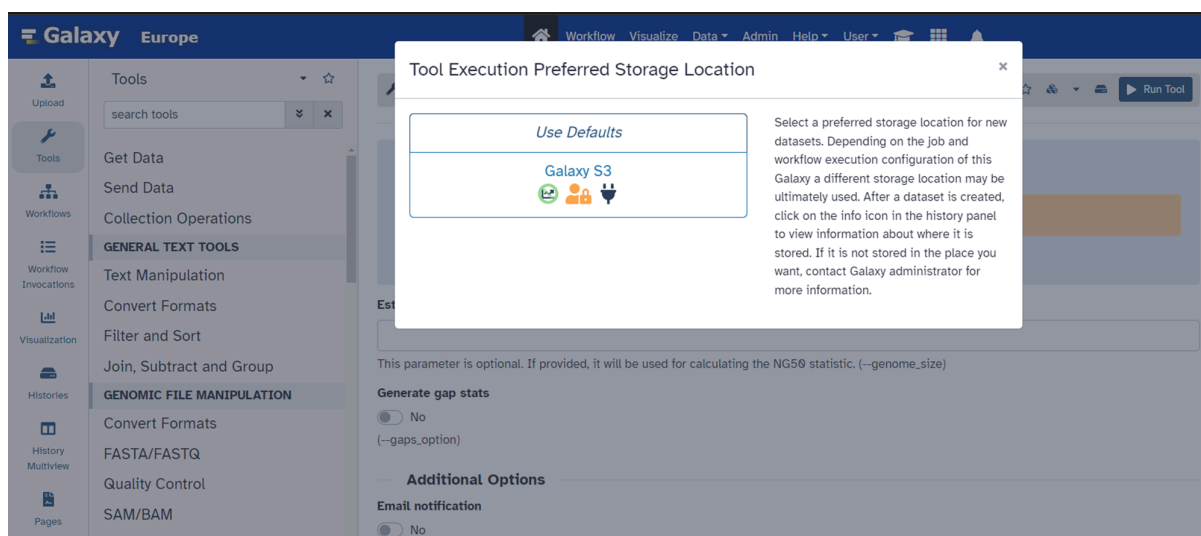


Figure 4.1.4: An example of selecting a dedicated storage endpoint for a tool execution.

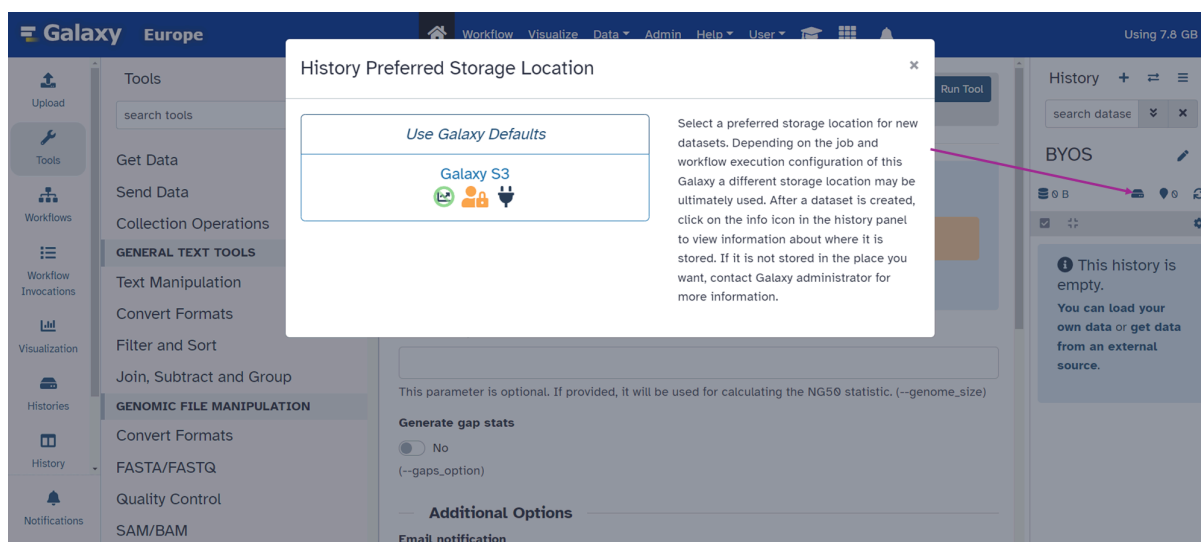


Figure 4.1.5: An example of selecting a dedicated storage endpoint at the history level.

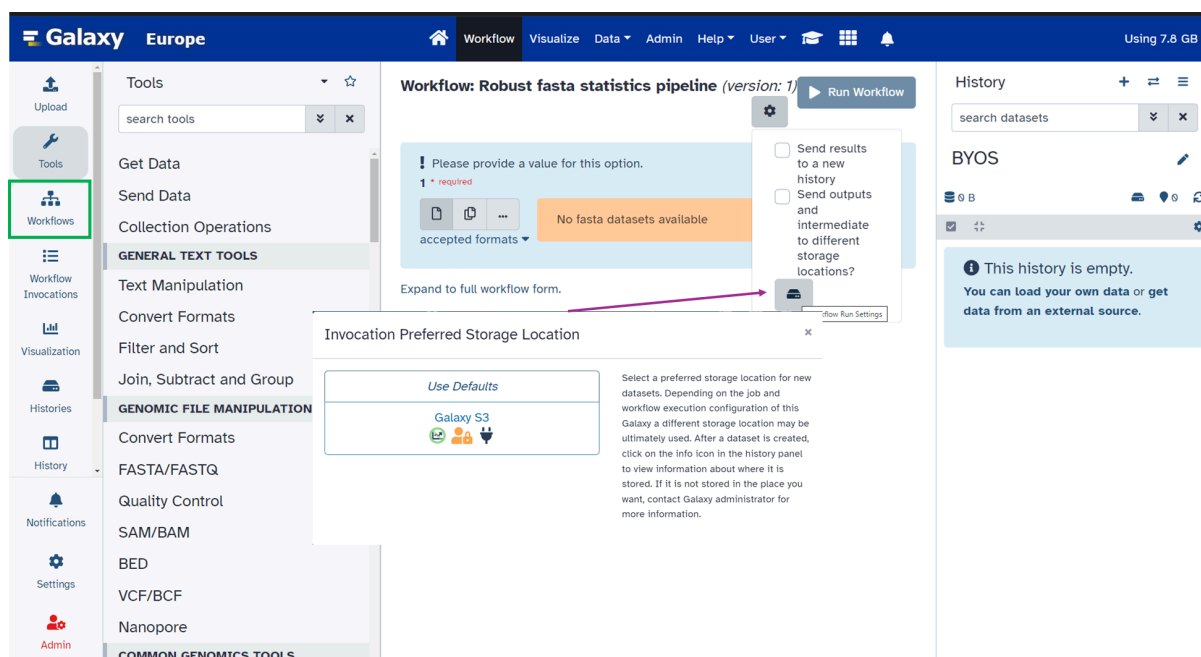


Figure 4.1.6: An example of selecting a dedicated storage endpoint at the workflow level.

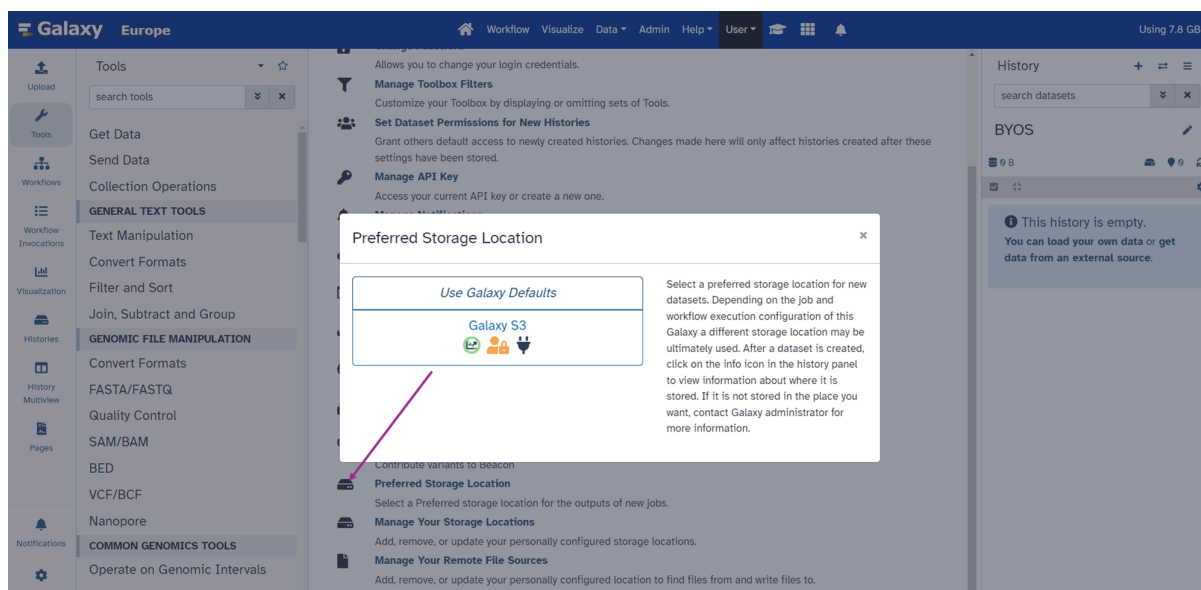


Figure 4.1.7: An example of selecting a dedicated storage endpoint as the user's default preferred storage location.



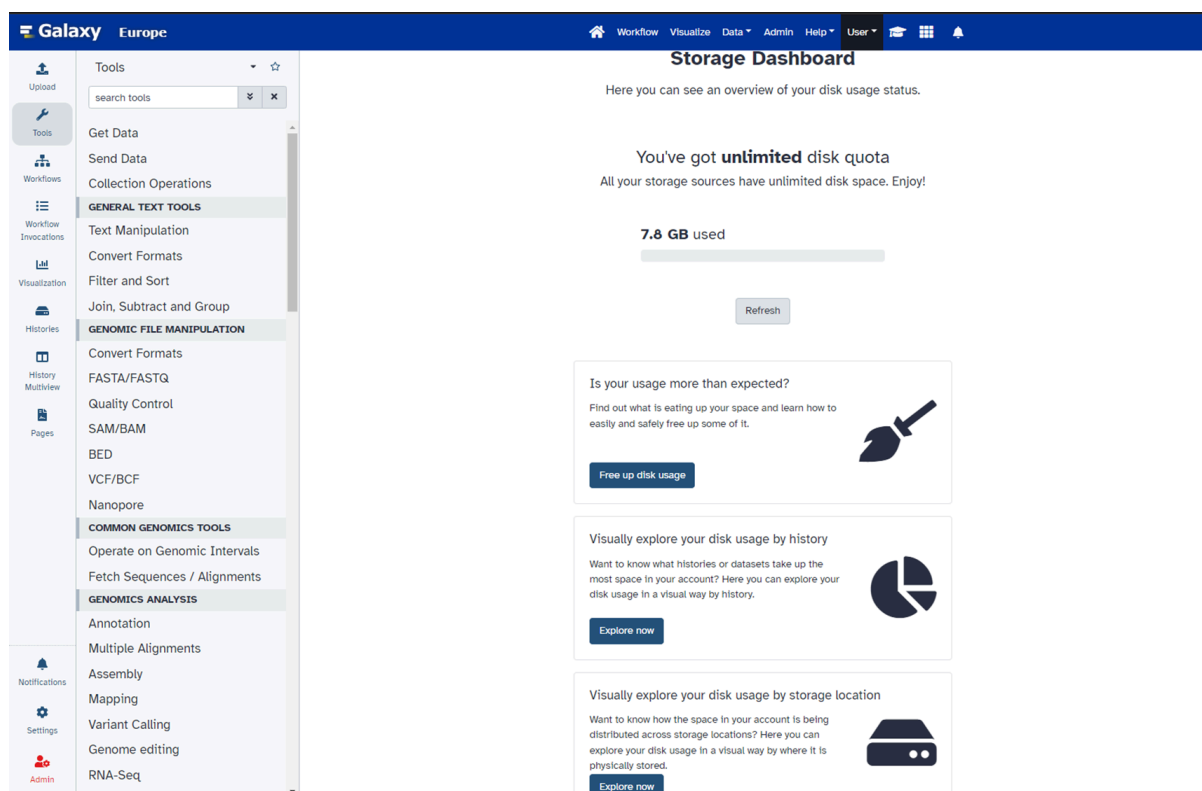


Figure 4.1.8: The new storage dashboard allows you to explore, visualize, and clean up or manage your storage.

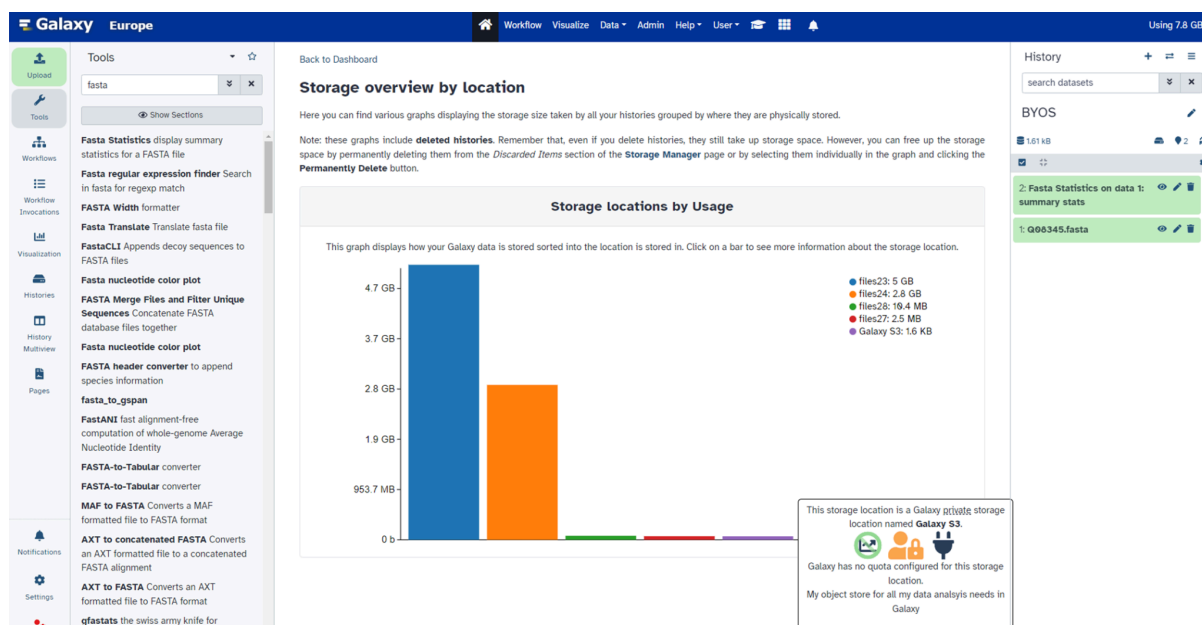


Figure 4.1.9: An example showing the disk usage separated by the storage location.

The main features of this work include a graphical way for users to add their own storage and select their desired storage depending on the use case. It can be at the tool execution, history, workflow invocation, or as a default preferred storage location, and a way for secrets of the



object stores to be stored securely. Galaxy admins can now add metadata to the managed object stores that can visually explain to users the type of storage, quota, speed, location, accessibility, storage policies, etc. Additionally, an object store templating system allows admins to offer users production templates to configure their object stores (from a set of given providers, e.g., Generic S3, Azure, AWS, GCP, Figure 4.1.1) with a set of required variables (Figure 4.1.2). These features are now available in the Galaxy release 24.1 and have been deployed at <https://usegalaxy.eu>.

4.1.1. MinIO

Users with access to cloud resources in the EGI Federated Cloud can also benefit from the available TOSCA template in Infrastructure Manager (or IM) for the automated deployment of MinIO³⁹. Please refer to Section 3.1.2 to learn more details about Infrastructure Manager and how to configure the credentials. The steps below assume the following: 1) the user has created an EGI Check-in account, and 2) is a member of a Virtual Organization with cloud resources to deploy MinIO.

Another prerequisite for the deployment of MinIO with IM is the use of some sort of Dynamic DNS service. EGI also offers a Dynamic DNS service for free for users with an EGI Check-in account. Please checkout the documentation⁴⁰ for further details. Using a Dynamic DNS service the user needs to register two DNS hostnames as a prerequisite for the automatic deployment of MinIO with IM: one DNS hostname for the MinIO Console⁴¹, and another one for the MinIO API endpoint. See a tutorial⁴² if you need further help.

These are the steps for the user:

- Visit <https://im.egi.eu/> and login using EGI Check-in.
- Select “Deploy a VM” and click “Configure” (Figure 4.1.1.1)
- Select “MinIO” and click “Add” (Figure 4.1.1.2)
- Configure deployment details for MinIO and click “Submit” (Figure 4.1.1.3)

IM will deploy a Virtual Machine with Docker⁴³ and MinIO as a containerised service. Therefore, the first step for the user is to select “Deploy a VM” in IM (see Figure 4.1.1.1).

³⁹ MinIO: <https://min.io/>

⁴⁰ EGI Dynamic DNS: <https://docs.egi.eu/users/compute/cloud-compute/dynamic-dns/>

⁴¹ MinIO Console: <https://github.com/minio/console>

⁴² EGI Dynamic DNS tutorial: <https://youtu.be/waylAJ4p-LA>

⁴³ Docker: <https://www.docker.com/>



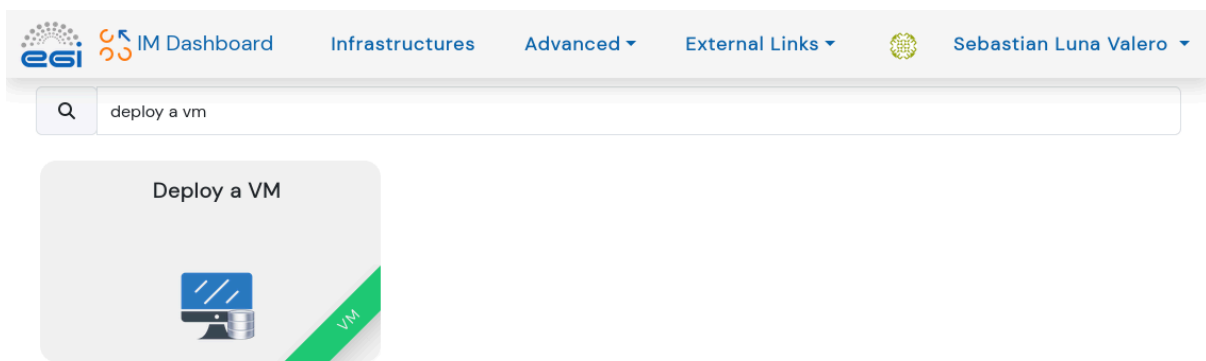


Figure 4.1.1.1. Select “Deploy a VM” and click “Configure”.

After selecting the option “Deploy a VM” in IM, the user is presented with several options to deploy services on top of a Virtual Machine. Figure 4.1.1.2 below shows the filtered output when looking for MinIO. The user needs to click on “Add” and then “Configure”.

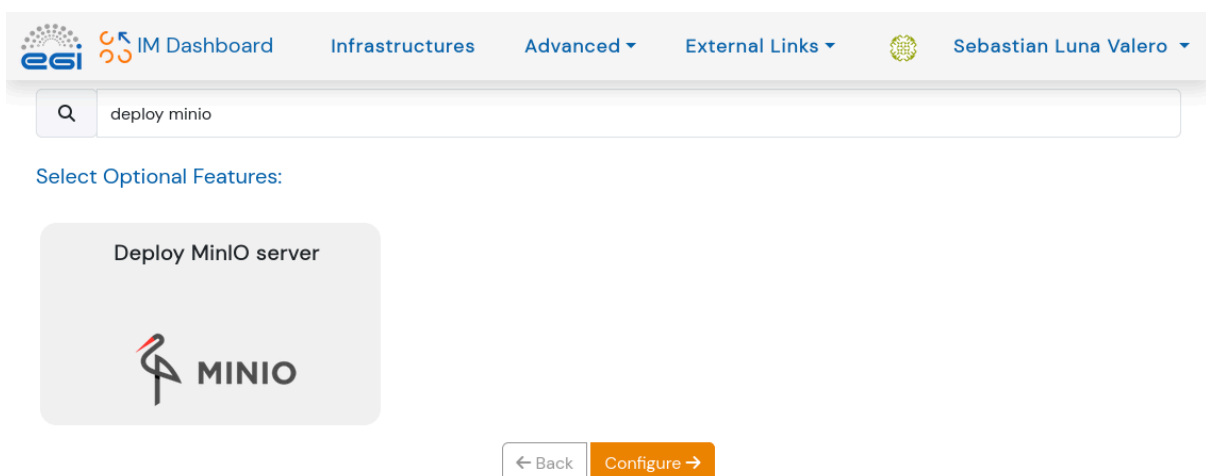


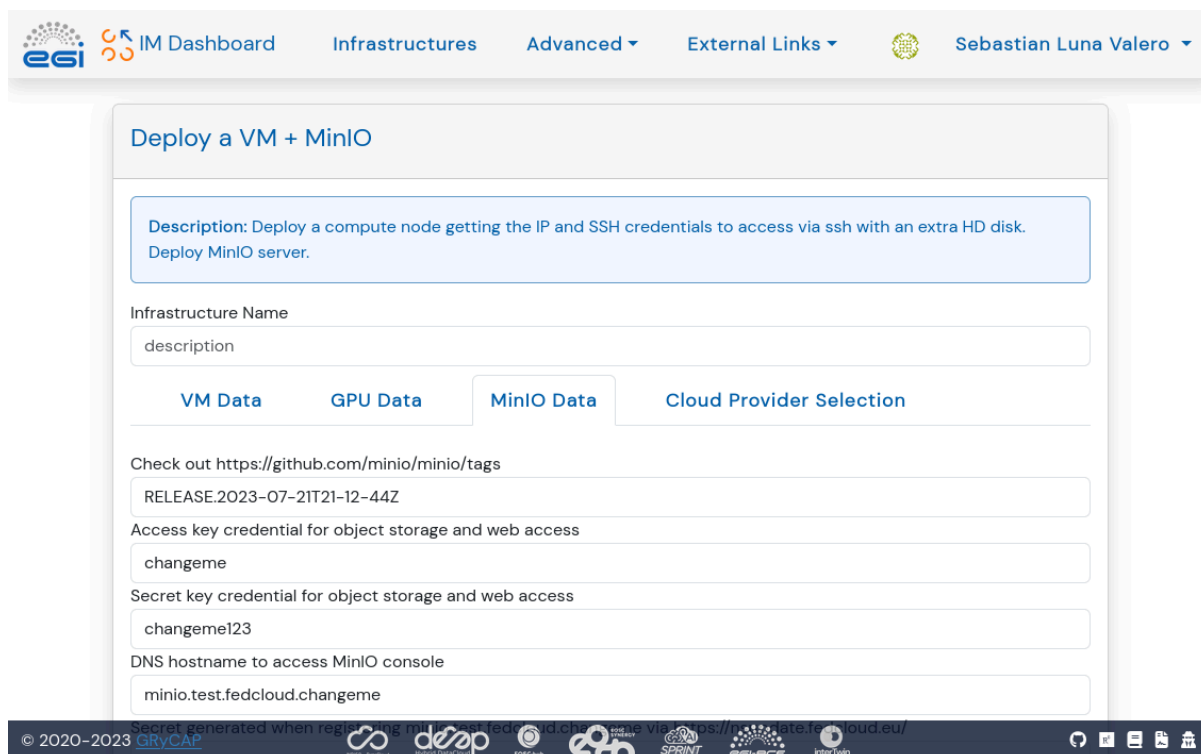
Figure 4.1.1.2. Select “MinIO”, click “Add”, and “Configure”.

In the configuration page for the deployment in IM (see Figure 4.1.1.3) the user needs to enter:

- The release tag from <https://github.com/minio/minio/tags> for the deployment.
- The desired *access* and *secret key* for MinIO.
- Dynamic DNS configuration for both the Console and the API endpoint.
- An email address used by certbot⁴⁴ when registering the TLS certificates.

⁴⁴ certbot: <https://certbot.eff.org/>





Deploy a VM + MinIO

Description: Deploy a compute node getting the IP and SSH credentials to access via ssh with an extra HD disk. Deploy MinIO server.

Infrastructure Name
description

VM Data **GPU Data** **MinIO Data** **Cloud Provider Selection**

Check out <https://github.com/minio/minio/tags>

RELEASE.2023-07-21T21-12-44Z

Access key credential for object storage and web access
changeme

Secret key credential for object storage and web access
changeme123

DNS hostname to access MinIO console
minio.test.fedcloud.changeme

© 2020-2023 GRV CAP

Figure 4.1.1.3. Configure deployment details for MinIO and click “Submit”.

Once all the parameters are configured for MinIO and the VM, the user needs to choose a cloud provider for the deployment. The next step is the selection of cloud credentials and the Virtual Machine Image for the deployment. Please refer to Section 3.1.2 to see an example.

All going well the deployment completes successfully and IM will show the URL for both the MinIO Console and the API endpoint that should be added to Galaxy to connect this external object store.

See the discussion⁴⁵ on GitHub to add the TOSCA templates and the Ansible playbooks to deploy MinIO with docker-compose on top of a Virtual Machine with IM.

4.2. Onedata

Onedata⁴⁶ is a global data management system providing easy access to distributed storage resources. With Onedata, users can access, store, process and publish data in a unified logical namespace spanning computing centres and storage providers worldwide. Onedata focuses on instant, transparent access to distributed data sets, without unnecessary staging and migration, allowing access to the data directly from a local computer or a remote server.

The Onedata team has been working on integrating Onedata with Galaxy on the data layer. There are two aspects to that; firstly, Onedata can act as an external source of datasets to be

⁴⁵ Discussion [on GitHub](#) for the automated deployment of MinIO with IM.

⁴⁶ Onedata: <https://onedata.org>



imported into Galaxy for processing (Galaxy Files Source Plugin). Secondly, Onedata can serve as a storage backend for storing the physical representation of Galaxy datasets (Galaxy Object Store). Both features, which are described in detail below, have already been implemented and integrated into the Galaxy codebase. They are based on the same common, lightweight Python library called *OnedataFileRESTClient*⁴⁷. The library has been created for the ESG project but is fundamentally a fully functional Onedata data access client.

The work done within the ESG project revolves mostly around the EGI DataHub service. It's essentially an instance of the Onedata ecosystem integrated closely with the EGI Check-In services and embracing its VOs and group structures. The EGI DataHub currently brings together 19 data providers, distributed around Europe, to cater for many scientific projects coordinated by EGI. Nevertheless, the Onedata integration in Galaxy is universal and can be used to connect it to any Onedata ecosystem.

4.2.1. Onedata — Galaxy File Source Plugin

The generic concept of a Files Source Plugin in Galaxy allows plugging in arbitrary data storage services as sources from which datasets can be imported into Galaxy. It reuses the Python's *pyfilesystem* abstraction so that any specific *pyfilesystem* plugin implementation can be straightforwardly plugged into Galaxy.

The Onedata team has implemented a new lightweight *pyfilesystem* plugin called *fs.onedatarestfs*⁴⁸, which uses *OnedataFileRESTClient* behind the scenes. It was integrated into Galaxy, along with the code and templates responsible for plugin configuration and corresponding user preferences — consult the PR⁴⁹ for details.

The Onedata files source plugin can be marked as *writable*, allowing the export of Galaxy datasets into an external data storage service.

The first version of the files source plugin has turned out to be unresistant to misbehaving data providers, causing some of the users to be unable to access and import some data collections into Galaxy. The Onedata team has prepared a subsequent version⁵⁰ that mitigates these problems. It includes mechanisms for dynamic selection of suitable data providers with the ability to fall back to another one in case of problems.

⁴⁷ OnedataFileRESTClient repository: <https://github.com/onedata/onedatafilerestclient>

⁴⁸ OnedataRESTFS repository: <https://github.com/onedata/onedatarestfs>

⁴⁹ Pull request with the files source plugin: <https://github.com/galaxyproject/galaxy/pull/16690>

⁵⁰ Pull request with improvements: <https://github.com/galaxyproject/galaxy/pull/18372>



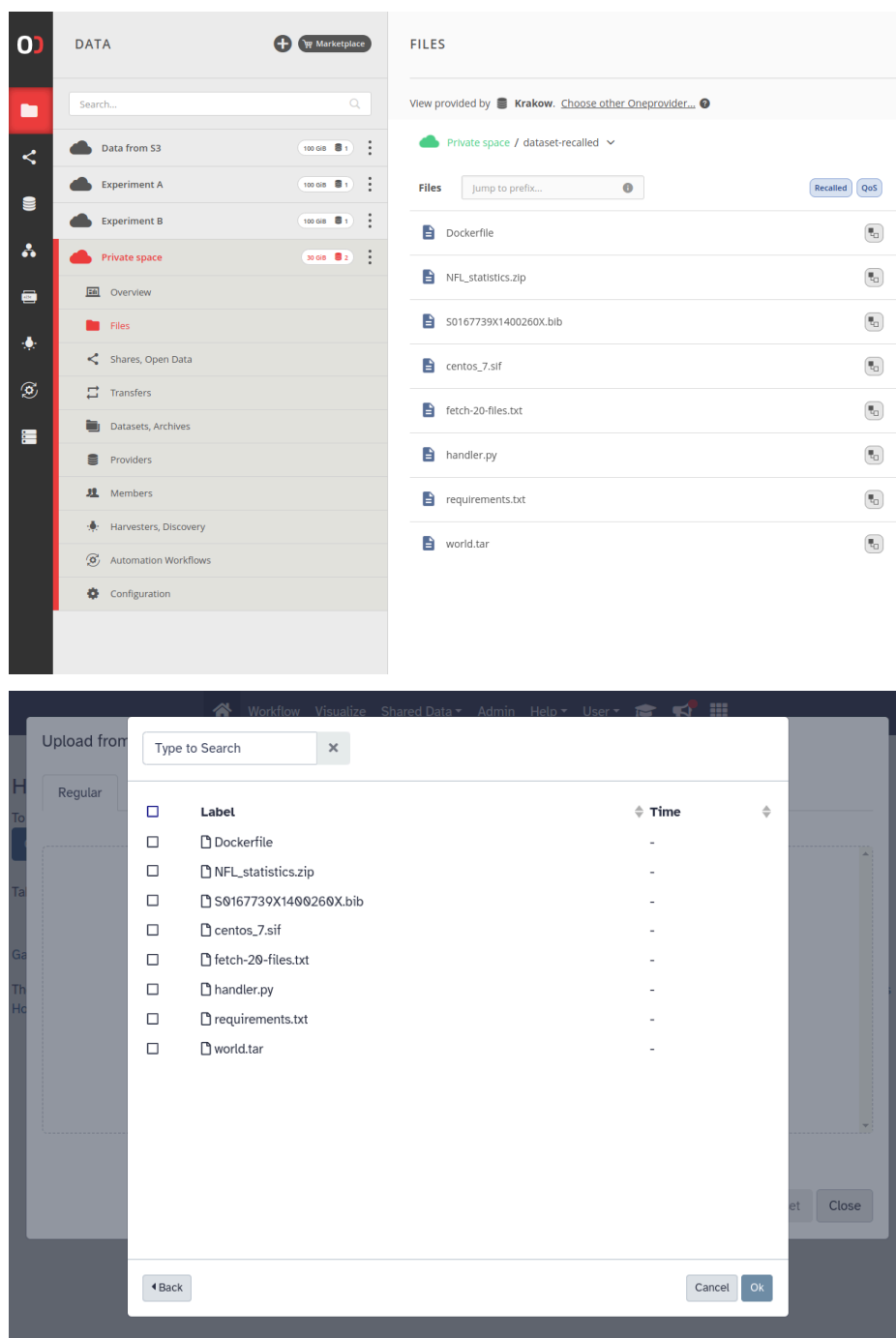


Figure 4.2.1.1. User data as seen in the Onedata interface and during import to Galaxy.

4.2.2. Onedata – Galaxy Object Store

Datasets in Galaxy are a logical concept that represents the data that is being processed using workflows (inputs and outputs). They are physically stored on a storage backend called an Object Store. Galaxy had already included support for different Object Stores, ranging from local POSIX disk storage to cloud storage solutions.



The Onedata team has implemented another Object Store subclass⁵¹ so that Onedata can be used to store the Galaxy data. It's based on the *OnedataFileRESTClient* library and is complemented by automated integration tests that start a minified, but fully functional Onedata environment to be checked against different Object Store operations. To make it possible, Onedata has been extended with a so-called *demo mode*⁵².

The introduction of the Onedata Object Store promises a powerful synergy with the distributed Pulsar network. Similarly to it, the Onedata ecosystem is distributed and the datasets are physically stored in different data providers. What's important, Onedata tracks the data distribution on a block-by-block basis and exposes this information through APIs. By introducing locality-aware scheduling logic into Galaxy (c.f. Figure 4.2.2.1), this knowledge can be used to minimize the data transfer costs (hence, the energy costs too) and execution delays. Locality-aware scheduling is still under development and the results will be presented in the upcoming deliverable D4.2.

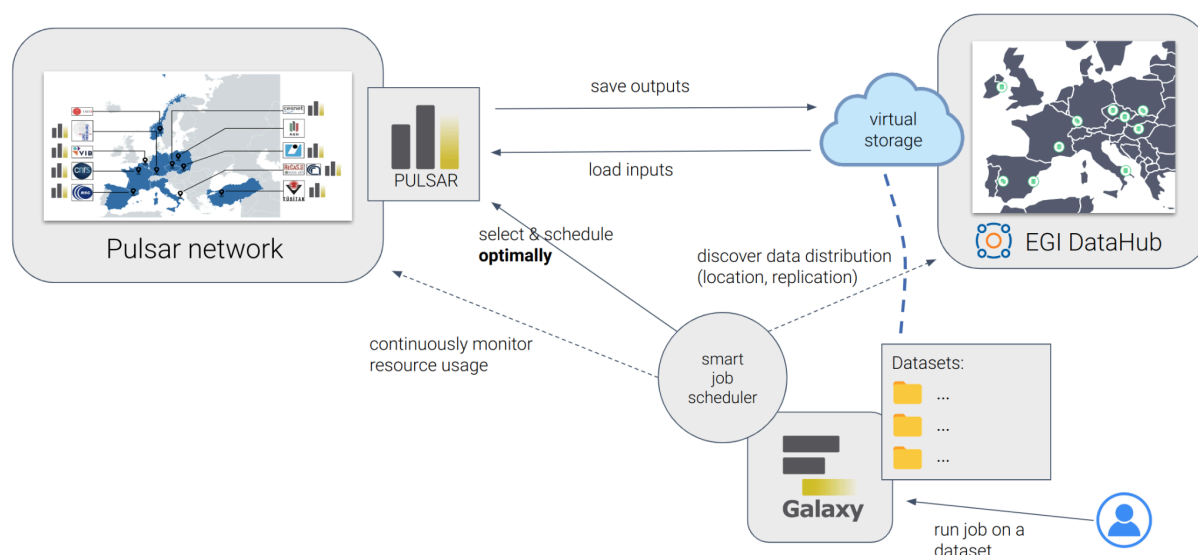


Figure 4.2.2.1. The synergy between Galaxy and Onedata distributed ecosystem.

Independently of the work on the Onedata Object Store, the generic Object Store layer in Galaxy has been refactored and reworked⁵³. The Onedata team has incorporated those changes and delivered a well-tested and improved Object Store version that takes advantage of the improvements regarding misbehaving data providers. It has been integrated into the Galaxy codebase⁵⁴.

⁵¹ Pull request with the Object Store: <https://github.com/galaxyproject/galaxy/pull/17540>

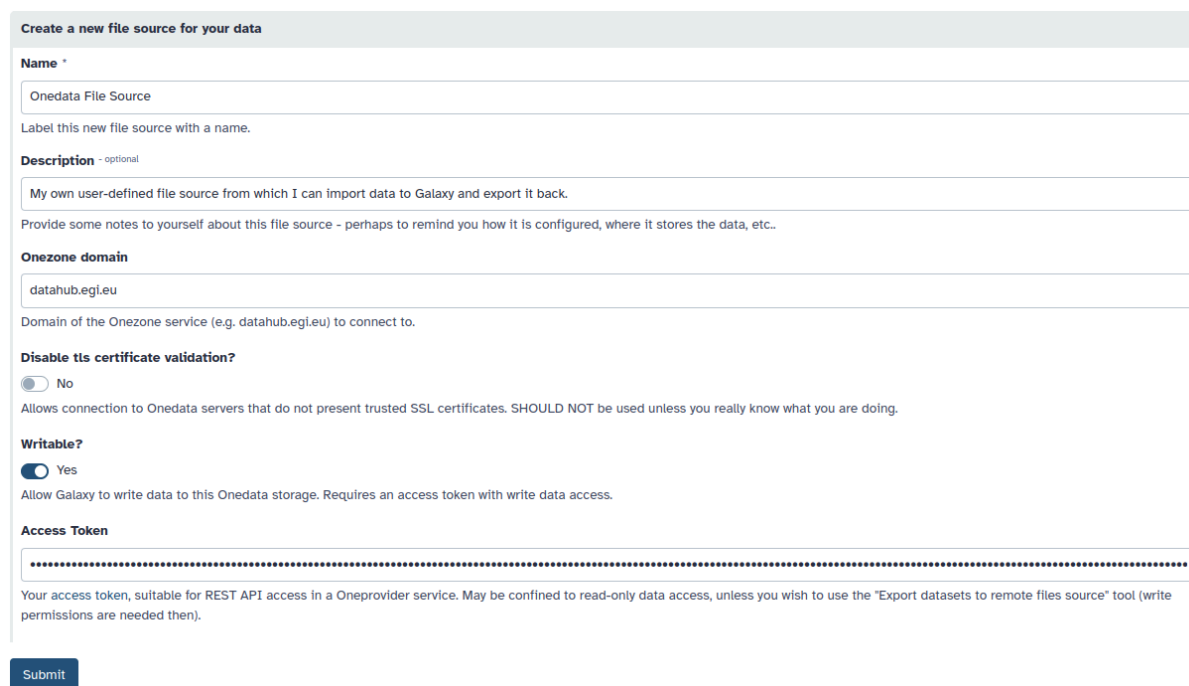
⁵² Demo mode docs: <https://onedata.org/#/home/documentation/topic/stable/demo-mode>

⁵³ Pull request with refactoring: <https://github.com/galaxyproject/galaxy/pull/18174>

⁵⁴ Pull request with final integration: <https://github.com/galaxyproject/galaxy/pull/18372>

4.2.3. Onedata – BYOS templates

Up until now, the features described above had to be configured by the Galaxy administrators. Thanks to the concept of Bring Your Own Storage, users are now empowered to configure their own Files Source Plugins and Object Stores that will be used to import and store their data. To make Onedata available as a BYOS option, the Onedata team has defined configuration templates and required models in the Galaxy project⁵⁵. Figures 4.2.3.1 and 4.2.3.2 show the BYOS configuration options for Onedata.



Create a new file source for your data

Name *

Onedata File Source

Label this new file source with a name.

Description - optional

My own user-defined file source from which I can import data to Galaxy and export it back.

Provide some notes to yourself about this file source - perhaps to remind you how it is configured, where it stores the data, etc..

Onezone domain

datahub.eui.eu

Domain of the Onezone service (e.g. datahub.eui.eu) to connect to.

Disable tls certificate validation?

☐ No

Allows connection to Onedata servers that do not present trusted SSL certificates. SHOULD NOT be used unless you really know what you are doing.

Writable?

☒ Yes

Allow Galaxy to write data to this Onedata storage. Requires an access token with write data access.

Access Token

.....

Your access token, suitable for REST API access in a Oneprovider service. May be confined to read-only data access, unless you wish to use the "Export datasets to remote files source" tool (write permissions are needed then).

Submit

Figure 4.2.3.1. Configuring a user-defined Onedata Files Source (BYOS) in Galaxy UI.

⁵⁵ Pull request with Onedata BYOS templates: <https://github.com/galaxyproject/galaxy/pull/18457>



Create a new storage location for your data

Name *

Onedata Object Store

Label this new storage location with a name.

Description - optional

My own user-defined object store that will hold my Galaxy data, effectively increasing the quota I can use.

Provide some notes to yourself about this storage location - perhaps to remind you how it is configured, where it stores the data, etc..

Onezone Domain

datahub.eui.eu

Domain of the Onezone service (e.g. datahub.eui.eu) to connect to.

Disable tls certificate validation?

☐ No

Allows connection to Onedata servers that do not present trusted SSL certificates. SHOULD NOT be used unless you really know what you are doing.

Space Name

Experiment X

The name of the Onedata space where the Galaxy data will be stored. If there is more than one space with the same name, you can explicitly specify which one to select by using the format <space_name>@<space_id> (e.g. demo@7285220ecc636075ae5759aec7ad65d3cha8f9).

Galaxy root directory

galaxy-data

The relative directory path in the space at which the Galaxy data will be stored. If empty, the data will be stored in the space's root directory.

Access Token

.....

Your access token, suitable for REST API access in a Oneprovider service. Must allow both read and write data access.

Submit

Figure 4.2.3.2. Configuring a user-defined Onedata Object Store (BYOS) in Galaxy UI.

4.2.4. NextCloud/WebDAV via Onedata

Onedata allows the connection to different storage backends. One of the options is WebDAV⁵⁶. In this subsection we show how to connect Onedata via WebDAV with storage deployed in the EGI Federated Cloud.

To test the provisioning of WebDAV we opted for the installation of a dedicated NextCloud⁵⁷ instance which include, natively, a WebDAV interface. More in detail the installation of the service was done on the IISAS-FedCloud OpenStack environment. The NextCloud all-in-one⁵⁸ solution was used to simplify the deployment of the service. The machine created for the services has the following configuration:

- 4 vCPU
- 8 GB RAM
- 60 GB storage for system files, operating system and NextCloud
- 50 GB of additional storage provisioned as additional volume attached to the VM

NextCloud was configured and the **esg** user created for demonstration purposes. This however has shown to not support the SabreDAV or ModDAV supported by DataHub limiting

⁵⁶ WebDAV: <https://en.wikipedia.org/wiki/WebDAV>

⁵⁷ <https://nextcloud.com/>

⁵⁸ https://apps.nextcloud.com/apps/nextcloud_all_in_one and <https://github.com/nextcloud/all-in-one>



the access to NextCloud to read-only mode. Other tests with different servers are currently underway to demonstrate the access to WebDAV through DataHub.

For the installation of oneclient another VM has been provisioned with the following configuration:

- 4 vCPU
- 8 GB RAM
- 30 GB storage for system files, operating system and *oneclient*

The configuration followed the instruction on the EGI documentation pages⁵⁹. This configuration however does not allow the setup of a WebDAV backend so the initial configuration was done by setting up the local storage. After this initial setup in the storage backends section of the galaxy-webdav cluster page, a specific connection to the WebDAV endpoint of the NextCloud service was added. This includes information about the endpoint itself⁶⁰ and the esg user:

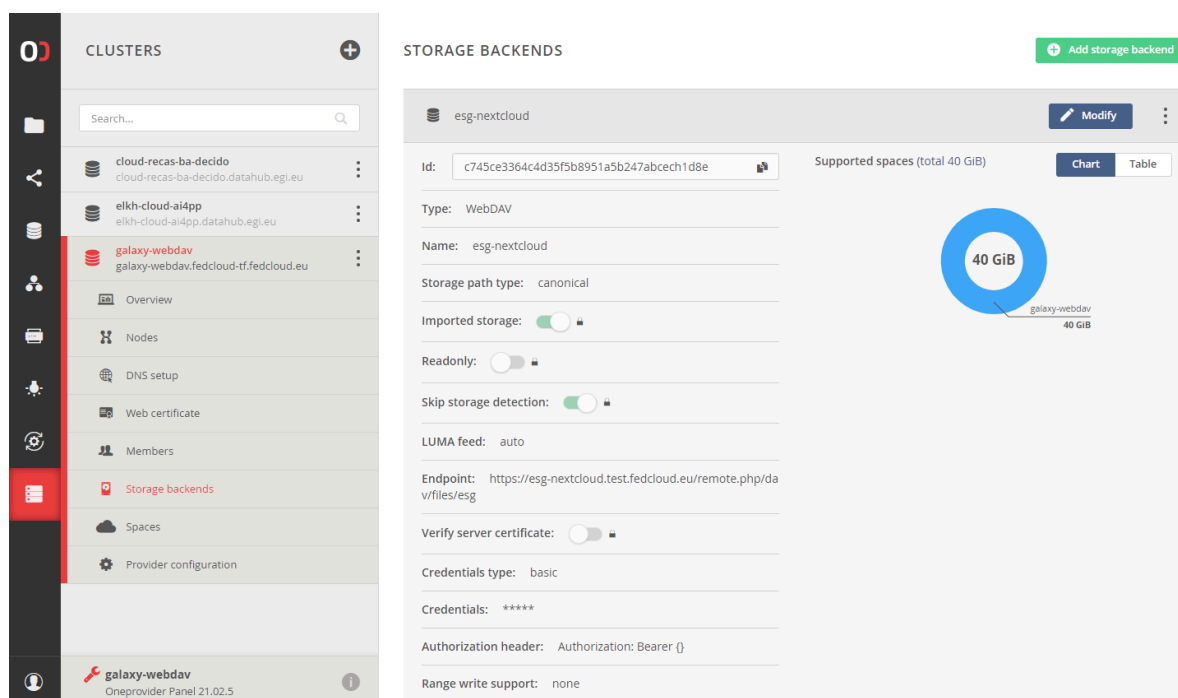


Figure 4.2.1.1. Configuration of the “galaxy-webdav” backend in Onedata

The WebDAV storage is now accessible through the DataHub service and can make available the WebDAV data as shown in the following screenshot:

⁵⁹ <https://docs.egi.eu/providers/datahub/oneprovider/>

⁶⁰ <https://esg-nextcloud.test.fedcloud.eu/remote.php/dav/files/esg>



The screenshot displays the 'galaxy-webdav' interface. On the left, a sidebar lists various data sources and tools. The main content area shows the 'galaxy-webdav' space details, including a file browser, information (Name, Creator, Created at, Shares, ID), space details (No details provided), providers (40 GiB), a world map, marketplace, members, and your membership.

As the last step of the configuration, the **vo.galaxy.eu** groups were added to grant access to the members of those VOs to the space created. This is shown in the following screenshot:

The screenshot displays the 'MEMBERS' page of the 'galaxy-webdav' interface. The left sidebar lists various data sources and tools. The main content area shows the 'MEMBERS' page with a list of groups and users. The 'GROUPS (5)' section lists 'vo.usegalaxy.eu', 'vo.usegalaxy.eu - members', 'egi-datahub-admins', 'vm_operator', and 'voms'. The 'USERS (11)' section lists 'Andrea Cristofori' (acristofori) as the owner. A table at the bottom shows privileges for 'Space management'.

| Privileges | Direct | Effective |
|------------------|-------------------------------------|-----------|
| Space management | <input checked="" type="checkbox"/> | 6/6 |



Funded by
the European Union

The configuration pages shown previously are part of the EGI DataHub⁶¹ service. More information on the usage of the service can be found on the EGI documentation pages⁶².

5. Summary

This deliverable presented the work carried out in the EuroScienceGateway project to make it easier for Galaxy users to connect their accounts in Galaxy to existing, externally managed compute and storage resources, also known as “Bring Your Own Infrastructure”. It covered the activities around integration of Galaxy with new Identity Providers, adding support for new compute and storage endpoints, and streamlining the way that users connect to them from Galaxy to exploit their benefits.

The EuroScienceGateway project has enhanced the capabilities of Galaxy in three areas: authentication and authorization infrastructure (AAI), computing power and storage capacity. These contributions improve the sustainability of Galaxy in the long run as the platform can now accommodate a larger number of users from across multiple disciplines with access to computing and storage resources that comes with its own funding model.

⁶¹ <https://datahub.egi.eu/>

⁶² <https://docs.egi.eu/users/data/management/datahub/>

