PROCESS

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777533.

PROviding Computing solutions for ExaScale ChallengeS

D7.1	Beta release of the Data service		
Project:	PROCESS H2020 – 777533	Start / Duration:	01 November 2017 36 Months
Dissemination[1]:	Public	Nature[2]:	R+O
Due Date:	31st January 2020	Work Package:	WP 7
Filename[3]	PROCESS_D7.1_Beta_release_of_the_Data_service_v1.0.docx		

ABSTRACT

Following project month 15 and 27 we continued to iterate upon the state of the platform previously described in D5.2 (Alpha release of the Data Service), to further extend data components but at the same time smoothly integrate them with compute components using the orchestration platform.

The demonstration presented in the deliverable focuses on the augmented version of the infrastructure components jointly utilized to enable execution of Use Cases from the IEE, which is now responsible for scheduling both data transfers (as needed, using the LOBCDER component) as well as running computations on this data using the relevant compute components, such as Rimrock (for HPC), Cloudify (for Cloud) or dedicated APIs like the one used to execute the Copernicus Use Case (UC 5) on the LRZ infrastructure. Of course, the core concept of auto-scalable data micro-infrastructure as described in the D5.2 is present in the toolchain and managed internally by LOBCDER, as described earlier.

[1] PU = Public; CO = Confidential, only for members of the Consortium (including the EC services).
[2] R = Report; R+O = Report plus Other. Note: all "O" deliverables must be accompanied by a deliverable report.
[3] eg DX.Y_name to the deliverable_v0xx. v1 corresponds to the final release submitted to the EC.

Deliverable Contributors:	Name	Organisation	Role / Title
Deliverable Leader[4]	Marian Bubak	AGH	Coordinator
Contributing Authors[5]	Maximilian Höb, Jan Schmidt	LMU	Writers
	Adam Belloum, Reggie Cushing, Onno Valkering	UvA	Writers
	Jason Maassen, Souley Madougou	NLESC	Writers
	Mara Graziani, Henning Müller	HES-SO	Writers
	Balazs Somoskoi, Jörg Pancake-Steeg	LSY	Writers
	Martin Bobák, Ondrej Habala, Martin Šeleng, Viet Tran	UISAV	Writers
	Piotr Nowakowski, Bartosz Wilk, Jan Meizner	AGH	Writers
Reviewer(s)[6]	Maximilian Höb	LMU	Reviewer
	Martin Bobák, Ondrej Habala	UISAV	Reviewer
	Piotr Nowakowski	AGH	Reviewer
Final review and approval	Maximilian Höb	LMU	Coordinator

Document History

Release	Date	Reasons for Change	Status[7]	Distribution
0.0	2019-11-21	Structure of the deliverable fixed	Draft	
0.1	2019-11-22	First draft of the chapters	Draft	
0.2	2019-12-06	Added more preliminary content	Draft	
0.3	2019-12-20	Technology part updated	Draft	
0.4	2020-01-10	Use case part updated	Draft	
0.5	2020-01-20	First draft of the complete document	Draft	
0.6	2020-01-29	Draft completed for internal review	In Review	
0.7	2020-01-30	Reviewed Draft	In Review	
1.0	2020-01-31	Final Version	Released	Public

[4] Person from the lead beneficiary that is responsible for the deliverable.
[5] Person(s) from contributing partners for the deliverable.
[6] Typically, person(s) with appropriate expertise to assess the deliverable quality.
[7] Status = “Draft”; “In Review”; “Released”.

Table of Contents

Executive Summary

This deliverable follows upon the work performed in the analysis and design phase of the PROCESS platform, including its data service components described in previous deliverables – particularly D4.1, D4.2, D5.1 and D5.2. The data-related components described therein were further extended and integrated with compute components – also previously described in the relevant deliverables (D6.1 and D6.2, presenting the first and second prototype of those components respectively).

Section 1 describes the current status of the PROCESS platform from the infrastructure perspective focusing on updates of the relevant components. Section 2 focuses on the current requirements for the data service from the perspective of the Use Case providers. Section 3 presents an overview of the demonstration of the current status of the platform. The deliverable is concluded in Section 4 which also outlines future work which needs to be performed prior to the final release of the platform. Additionally, the document contains Appendices which outline currently available functionality of the platform on the example of four selected Use Cases.

List of Figures

1 Overview

In this section we present the updated version of the description of the PROCESS data services. A detailed description of each of the components was already provided in D5.2 and this section only describes updates which have been put in place since the release of that deliverable.

1.1 DISPEL

DISPEL installations are distributed across the infrastructure, bringing computations and especially data preprocessing to the data itself, in order to decrease network transfer requirements. In the first phase of the project we concentrated on modifying the existing tools to fit PROCESS architecture and data platform. In the second phase of the project, reported here, we began integration of DISPEL services with DataNet, the PROCESS' metadata storage. Using DataNet as a central metadata storage, individual DISPEL instances can have access to each other's outputs, further decreasing data processing requirements and network load. Once certain data is prepared by one instance, other instances will find their metadata in DataNet, so they don't need to collate it again when needed.

1.2 Cloudify

In the beta release of the Data Service, Cloudify provides support for WebDAV storage, which enables better integration with the data infrastructure. Jobs running in the cloud can have direct access to input and output data stored on external storage resources, via the WebDAV protocol. This will enable jobs running in the cloud to be fully integrated with the data infrastructure provided by LOBCDER.

1.3 DataNet

During the beta implementation phase, basic scalability of DataNet repositories was implemented on top of a Docker SWARM cluster. Scalable repositories allow for metadata database evolution in terms of volume. A NoSQL database, utilizing so-called shards, provides efficiency of read and write transactions in the growing metadata dataset.

It has also been decided that DataNet would be used for storage and analysis of the computation and Data Transfers lifecycle, to collect proper metrics required for validation of the further prototypes and the final release. Adequate solutions for these needs have been designed.

1.4 LOBCDER

To extend the programmability of LOBCDER into runtime, preparations have been made to support reactive and event-driven execution. Central to this is a message bus, which can be fed with events from internal and external (e.g. a programmed driver) sources. Services in the micro-infrastructure can independently react to events. This decentralized interplay of services and events shapes the dynamic, programmable, execution. In order to research and experiment with such programmability, a separate framework has been created. This framework is initially based on the imperative programming paradigm. It features a domain-specific language (DSL) that has been designed such that it is quick to write and easy to debug. We're investigating the adaptation of this framework to an event-driven programming paradigm in order to establish integration with LOBCDER. The final goal of this runtime programmability is to effortlessly support new and/or changing requirements of use cases without the need for the development and incorporation of use case specific services.

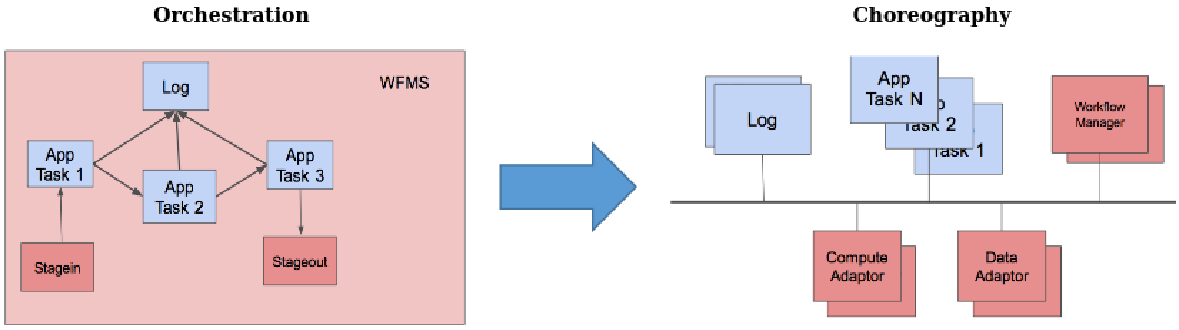

Figure 1 Dynamically established execution at runtime

In Figure 1 execution is no longer predetermined but is dynamically established at runtime. This is made possible by a message bus that allows services to interact between themselves (orchestration vs choreography).

To extend the resource pool of LOBCDER we integrated LOBCDER with Cloudify to provision virtual resources on demand. With this integration, containers that make up a micro-infrastructure can be mapped to dedicated VMs. This is done during the initial steps of creating the micro-infrastructure with a new flag in the JSON description. With the new Cloudify integration, LOBCDER can now spin up new VMs as part of the micro-infrastructure description. Part of the description is a flag (*dedicatedNode*) to notify that the deployment should be mapped on a dedicated VM. In this case LOBCDER will first provision a VM through Cloudify and attach to the k8s cluster before mapping the deployment onto the new VM. Figure 2 illustrates the additional steps in the sequence of creating a micro-infrastructure from Figure 3 in D5.2. This allows for the resource pool to scale while providing more secure isolation for use cases.

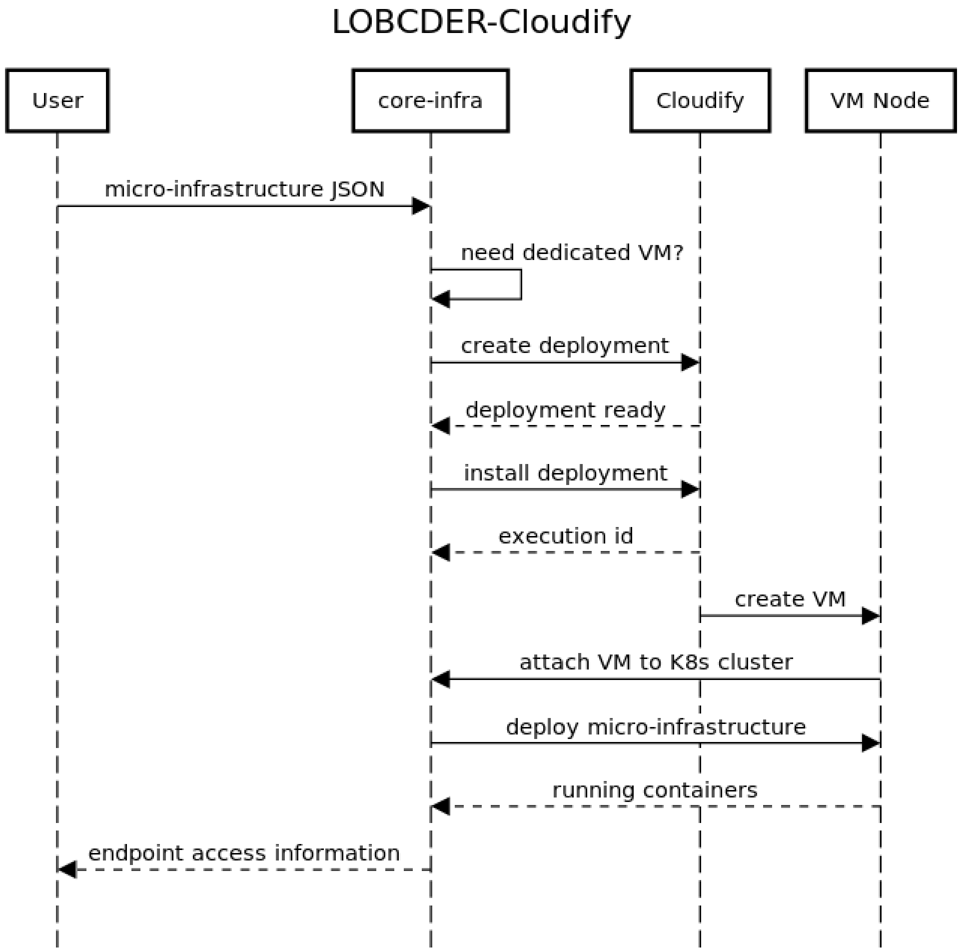

Figure 2 LOBCDER and Cloudify sequence to obtain new resources for user's micro-infrastructure. The rest of the sequence to generate a micro-infrastructure is described in D5.2.

Cloudify integration is effected through REST API web services. A Cloudify installation plus virtual resources are available at UISAV. The implementation approach for this integration involves:

- Preparation of a virtual machine with required software tools to allow the virtual machine to be able to attach to Kubernetes and run Docker containers.
- LOBCDER core-infra web service detects the *dedicatedNode* flag in the JSON submission to create a new micro-infrastructure. More details on how this was previously done can be found in D5.2.
- LOBCDER first checks if a VM with the same id is already running, in which case it merely submits containers to the preexisting node. If no running node is available, LOBCDER contacts Cloudify to boot a new VM.
- From the virtual machine a Cloudify blueprint is created. This blueprint can be called as a deployment through the REST API. The blueprint is parameterized with the following parameters:
    - *master*: the IP and port of the Kubernetes master node.
    - *token*: the Kuberntes security join token.
    - *discovery_ca*: hash of the public key of CA. Token-based discovery involves validation that the root CA public key matches this hash.
- Creating a new Kubernetes node involves installing the deployment through the REST API with the *deployment_id* from the previous step. Installation takes several minutes.
- LOBCDER waits for the VM to come up and periodically polls Kubernetes nodes to determine when the new node has appeared.
- Deleting the Kubernetes node can also be done through LOBCDER which results in uninstallation of the Cloudify deployment.

2 PROCESS data services from the user perspective

2.1 Data Services for Medical Use Case

The Data Services provided by PROCESS are used to meet the requirements of the Data Upload workflow as defined in D4.1. The datasets available on the local storage resources are uploaded to the HPC infrastructure by making use of the WebDAV and LOBCDER data services. Correct handling of the data and its distribution to CPU and GPU is ensured to avoid the bottlenecks of I/O operations and data decoding. Data download and transfer are readily available through WebDAV.

2.2 Data Services for LOFAR Use Case

UC#2 currently uses two data services provided by LOBCDER: the data staging service and the HPC-SSHFS data transfer service. We have been using these services for a couple of months on DAS5 and they have proven robust. However, during our technical meeting in Munich, we could not use them to stage and transfer observational data from the LTA to CoolMUC in Munich as this required addressing security-related issues specific to the MUC infrastructure. Furthermore, several other points need to be dealt with:

- Use of the above data services from any PROCESS computing infrastructure, including LISA, PROMETHEUS and CoolMUC
- Given the size of the input data, a network of data transfer nodes, or an equally performant high-speed transfer technology (FTS), would be very beneficial for horizontal scaling
- The end products of our use case are images; currently we use an ad-hoc solution to send them back to the UI. This solution is slated to be replaced with a LOBCDER access extension
- The final images of UC#2 are FITS images but given their large size, we convert them into JPEG before displaying them in the UI. DISPEL can be used to perform this conversion in situ, which would also improve transfer time

2.3 Data Services for UNISDR Use Case

The Data Services for UNISDR will be replaced with the appropriate set of data and exposed as the services using the PROCESS infrastructure in the future/final release of the platform.

2.4 Data Services for Ancillary Pricing Use Case

The work has been carried out to enable running mixed workflows featuring the part of the use case running in the cloud via Cloudify (as described in the previous release) and the second part on an HPC node. To this end, a solution was designed to enable moving data between Cloud and HPC sites by extending LOBCDER to allow integration with HDFS.

2.5 Data Services for Copernicus Use Case

UC#5 uses the LRZ Data Science Storage adaptor of LOBCDER to stage out results of the computation and make them available to the user via the LOBCDER WEBDAV endpoints through the IEE. This enables archiving results on the DSS as well as easy access for end users. Due to the proprietary nature of the use case, data stage-in cannot be handled by LOBCDER – however, this is only a legal restriction; not a technical one.

3 Demonstration scenarios

The second prototype was demonstrated using the 4 out of 5 Use Cases described below – namely the Medical Use Case (1), LOFAR Use Case (2), Ancillary Pricing Use Case (4) and

Copernicus Use Case (5). This section describes the demonstration scenario and the resulting screenshots are provided in the Appendix to this deliverable.

3.1 UC#1 scenario

The demonstrator highlights the process of preparing the computation for execution on the HPC infrastructure at Cyfronet. This process involves scheduling of the Use Case code, packed as a singularity container, on the infrastructure, using the Slurm scheduler. The demo covers the entire process including:

- Preparation of the computation and data
- Running the computation
- Collecting results

3.2 UC#2 scenario

Use Case 2 is demonstrated in a similar fashion to Use Case 1. The main difference involves the need for a pre-staging phase where a large amount of data is moved from the LTA to the proper HPC site. The infrastructure has also been prepared for tighter integration with the LOFAR portal which will be implemented before the release of the production prototype.

3.3 UC#4 scenario

This use case is deployed in the cloud. It is composed of two phases. In the first phase, data is generated and stored in HDFS. Subsequently, it is processed and stored in local WebDAV – which is part of the local data micro-infrastructure. It enables moving data for further access or processing e.g. on other types of e-infrastructures.

3.4 UC#5 scenario

In this scenario computations are only deployed on the LRZ Cluster – namely, CoolMUC. The operation is scheduled via a specialized API. The demo also includes preparation of the computation, execution of Use Case code, as well as obtaining access to results.

4 Conclusion and future work

In this deliverable we have described progress beyond the alpha stage of the platform, focusing on smooth integration with the rest of the PROCESS platform – particularly the orchestration component. Nearly all use cases now operate on various elements of the PROECESS platform. We have also begun work on strengthening integration across our core technologies (HPC/Cloud) and geographical locations where PROCESS resources reside.

In the future we will work to further stabilize the solution and make it as user friendly as possible. The most significant tasks for the data service platform will include:

- Further streamlining of data transfers across technologies and sites
- Support for updated versions of Use Cases, such as a multi-node MPI version of the Medical Use Case or scaled-up version of the LOFAR pipeline
- Improving the user interface to provide better integration with Science Gateways such as the LOFAR portal, including, of course, presentation of the resulting data to end users, taking into account physical restrictions imposed by the size of input and output data

Appendices – demo of the Beta release of the Data service

Appendix A: UC#1 scenario

The computation is set up by providing the required parameters, as visualized in Figure 3.

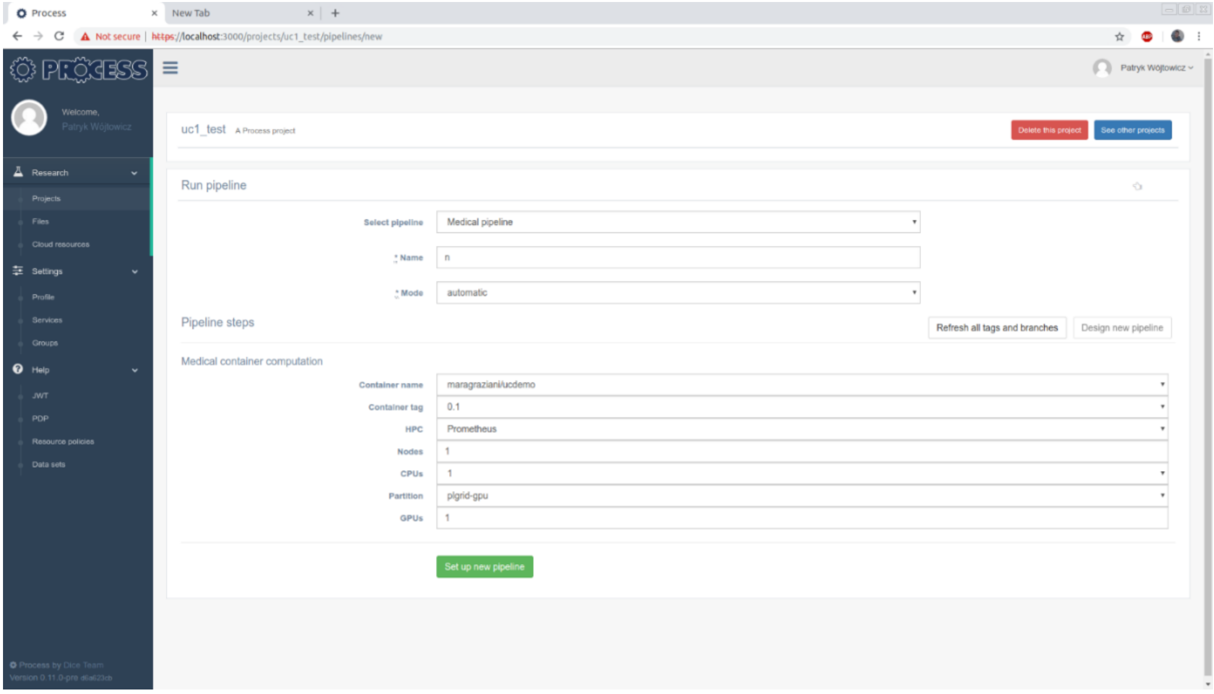

Figure 3 UC#1 – Configuration

Then pipeline is prepared for computation as shown in Figure 4.

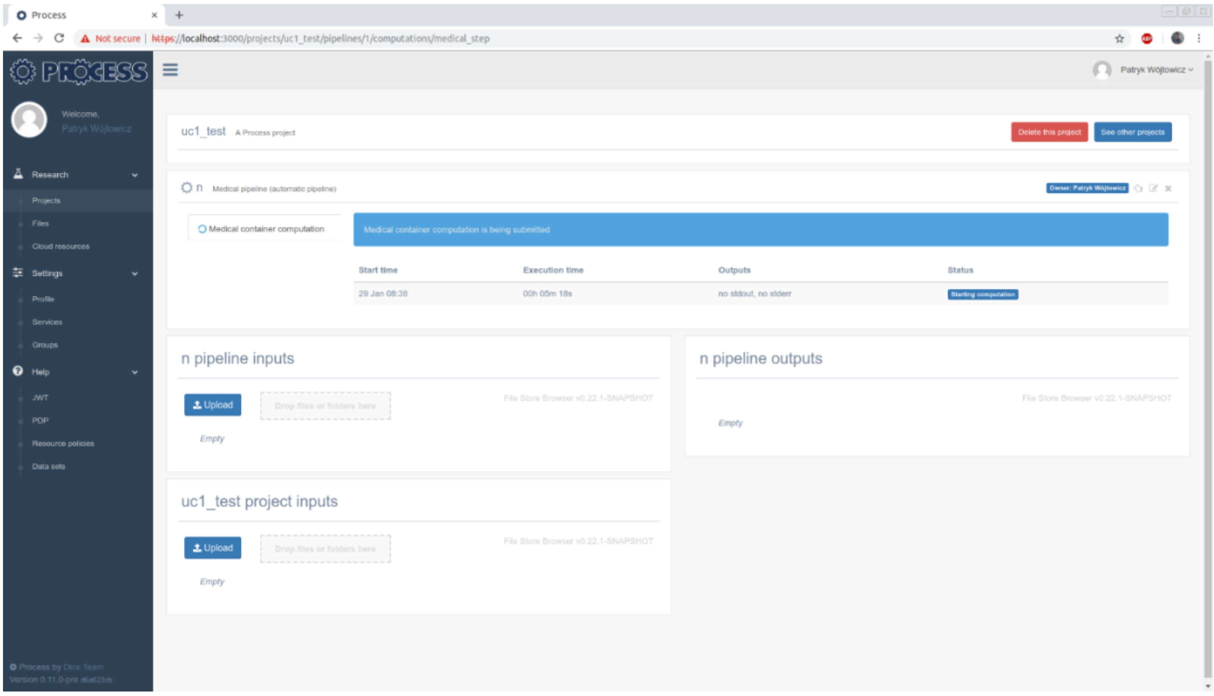

Figure 4 UC#1 – Preparation

The computation is scheduled and executed on the HPC cluster at Cyfronet, as shown in Figure 5.

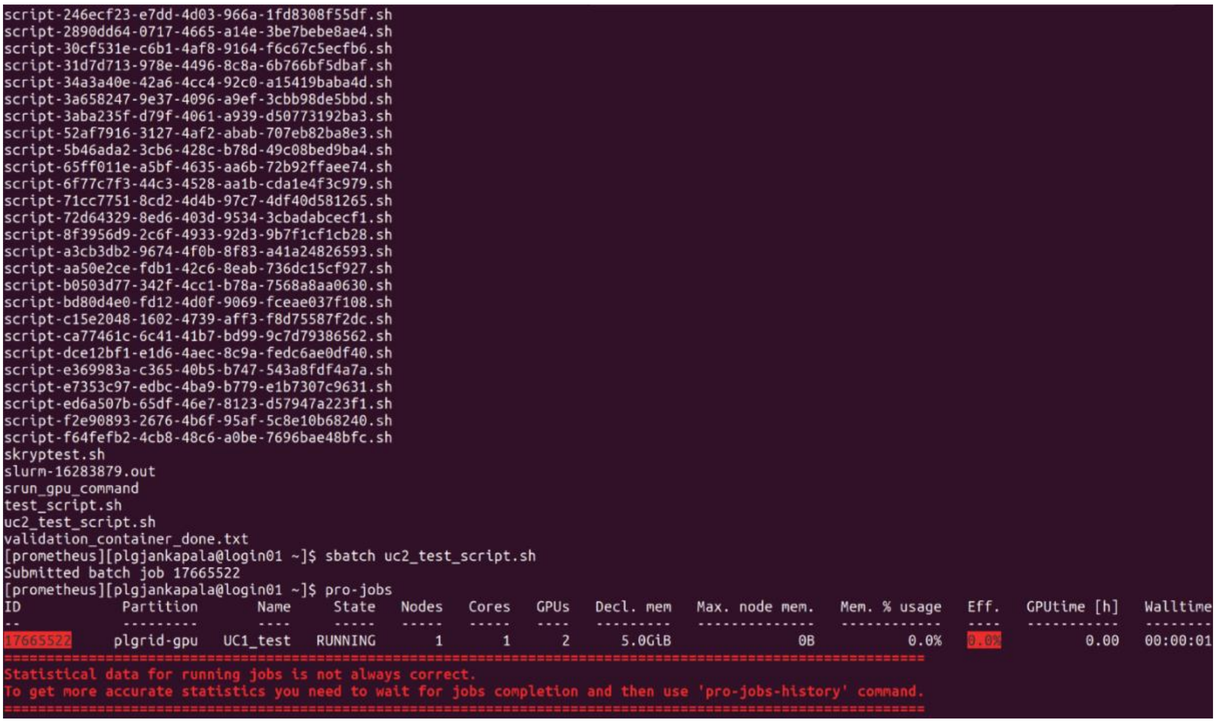

Figure 5 UC#1 – Execution on HPC

When the computation finishes, it is presented in the IEE Web interface as shown in Figure 6.

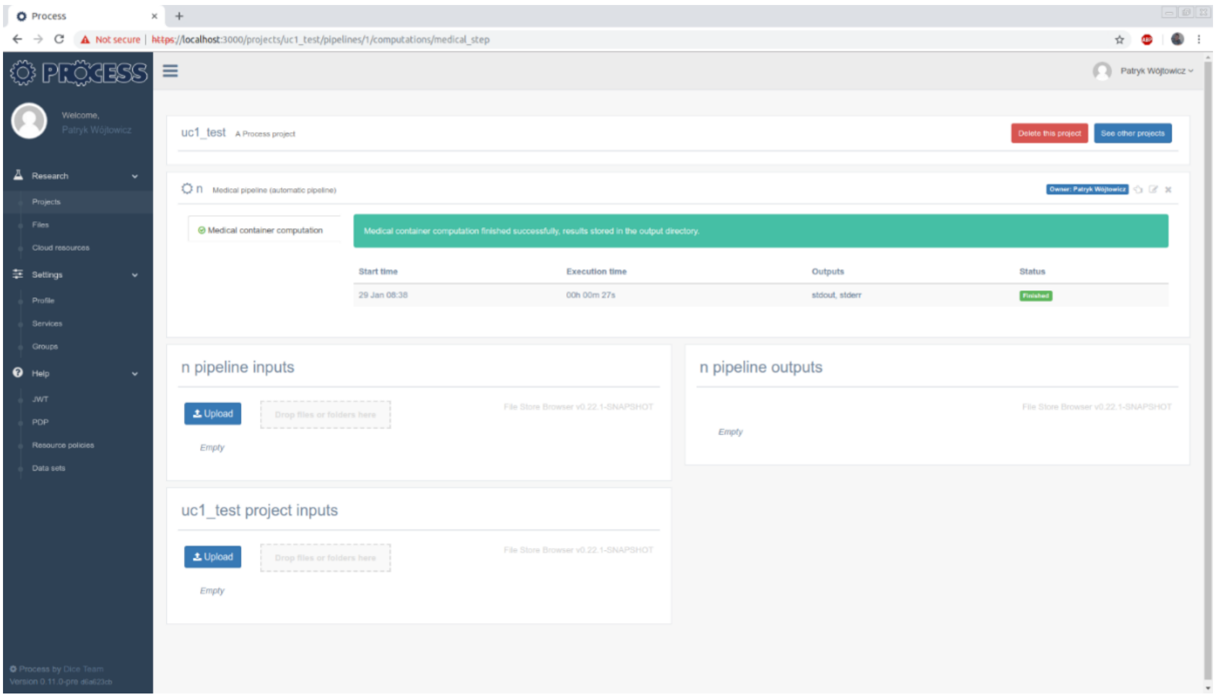

Figure 6 UC#1 – Completion

The resulting data is generated, as shown in Figure 7, and may be downloaded by the user.

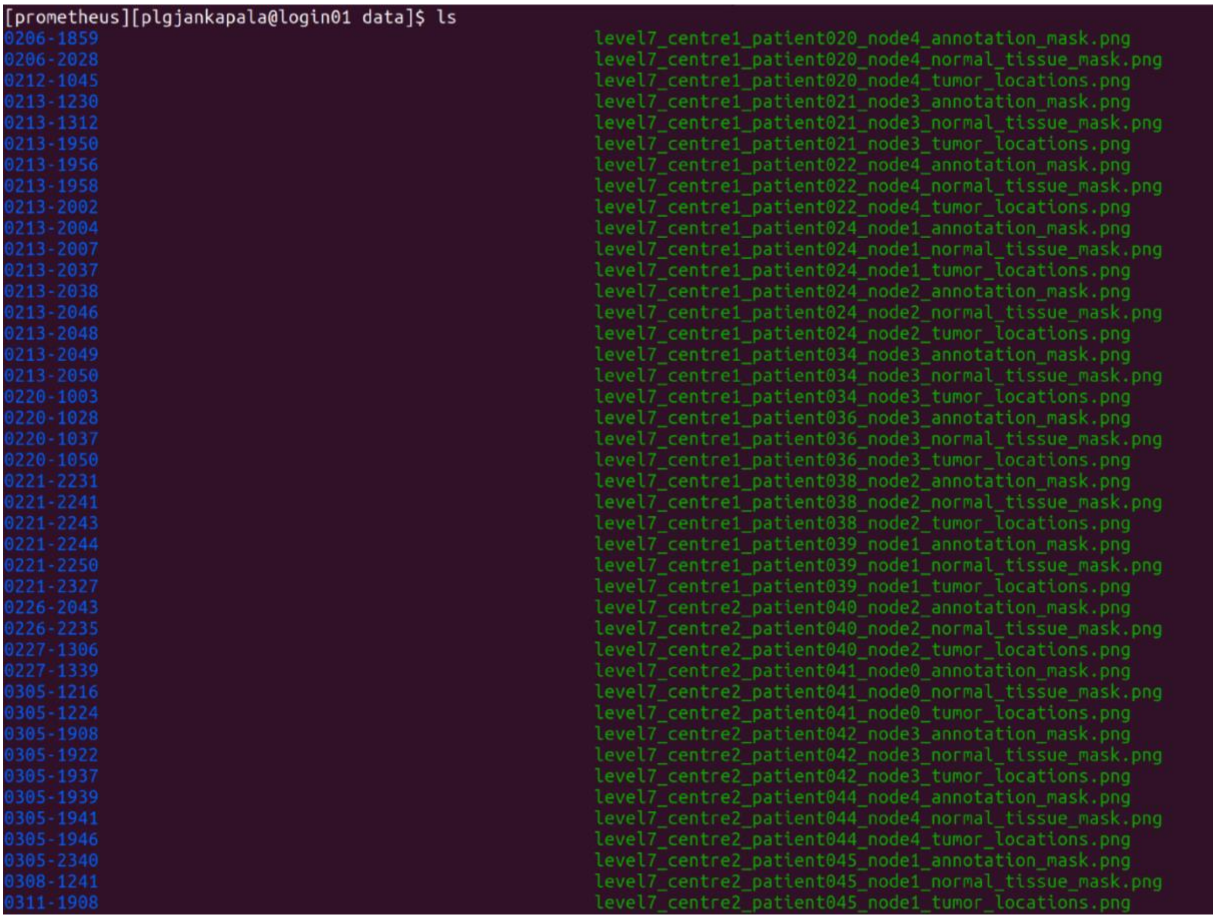

Figure 7 UC#1 – Outputs

The final state is shown in Figure 8.

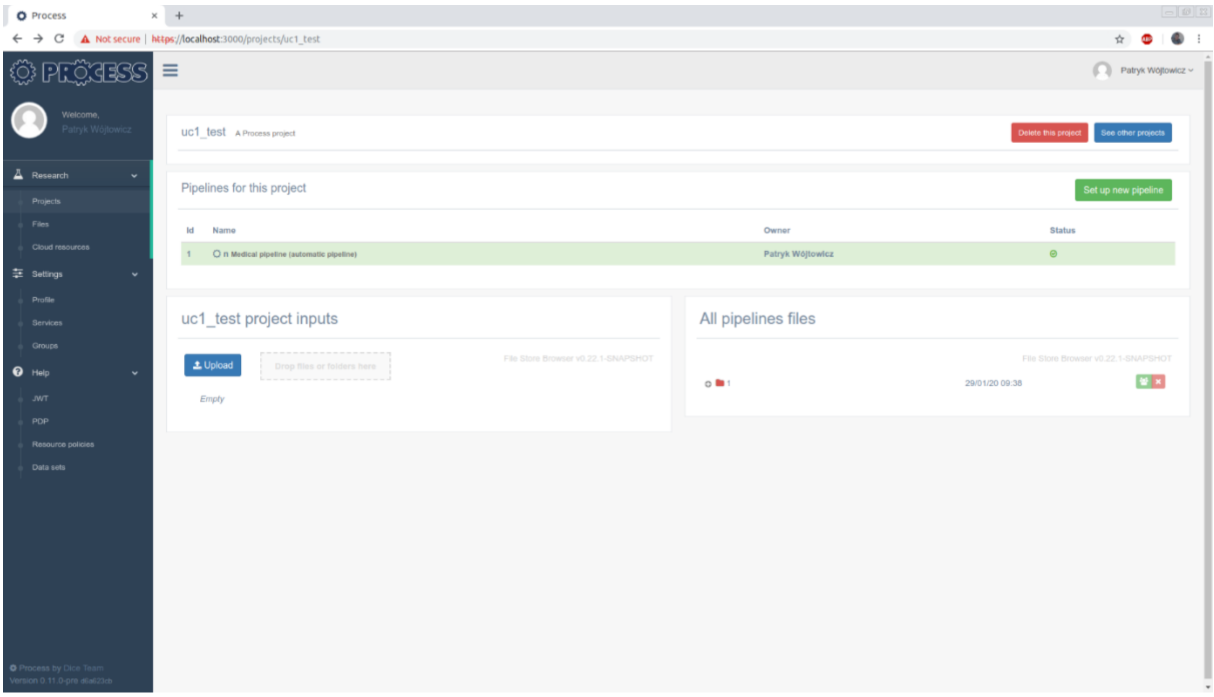

Figure 8 UC#1 – Final state

Appendix B: UC#2 scenario

Much like UC#1, the pipeline is also configured prior to launch, as shown in Figure 9.

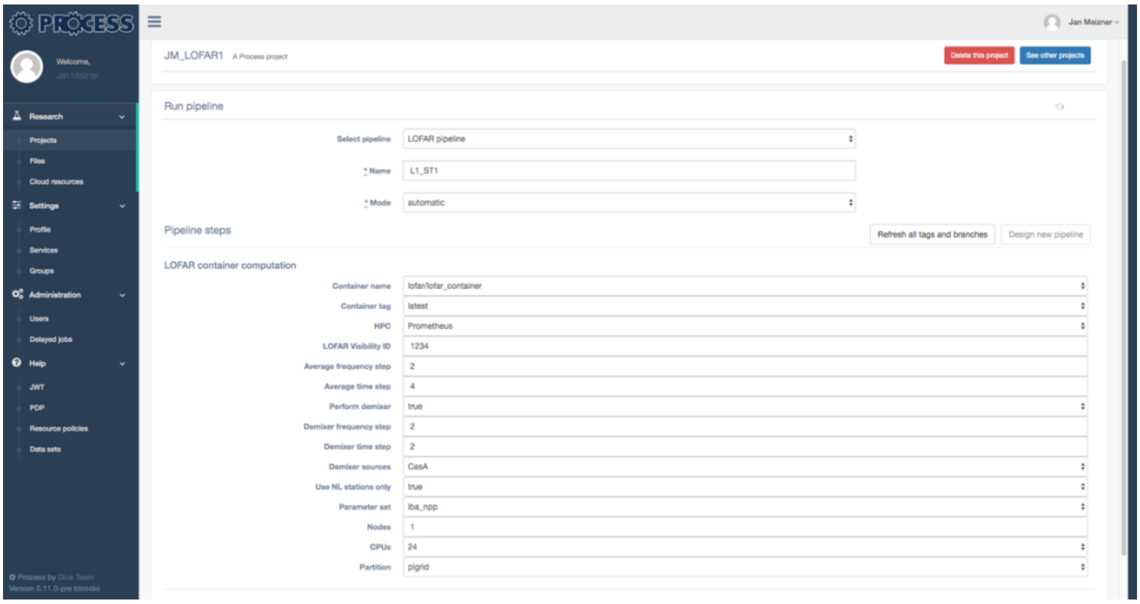

Figure 9 UC#2 – Configuration

Figure 10 shows a pipeline which is ready for execution.

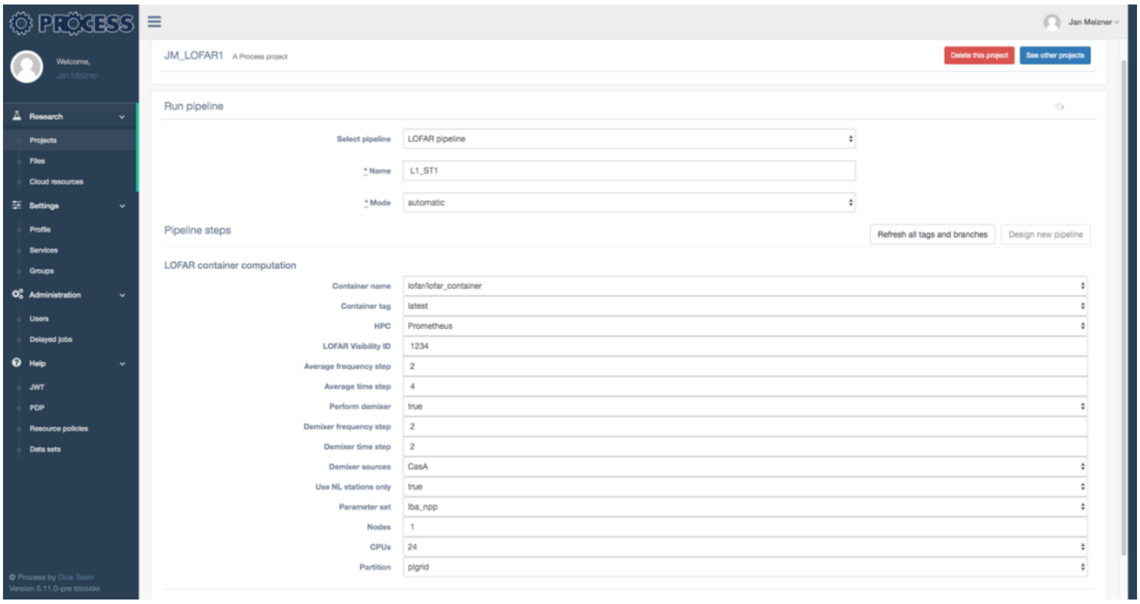

Figure 10 UC#2 – Ready for execution

The pipeline is subsequently run on HPC resources, as shown in Figure 11.

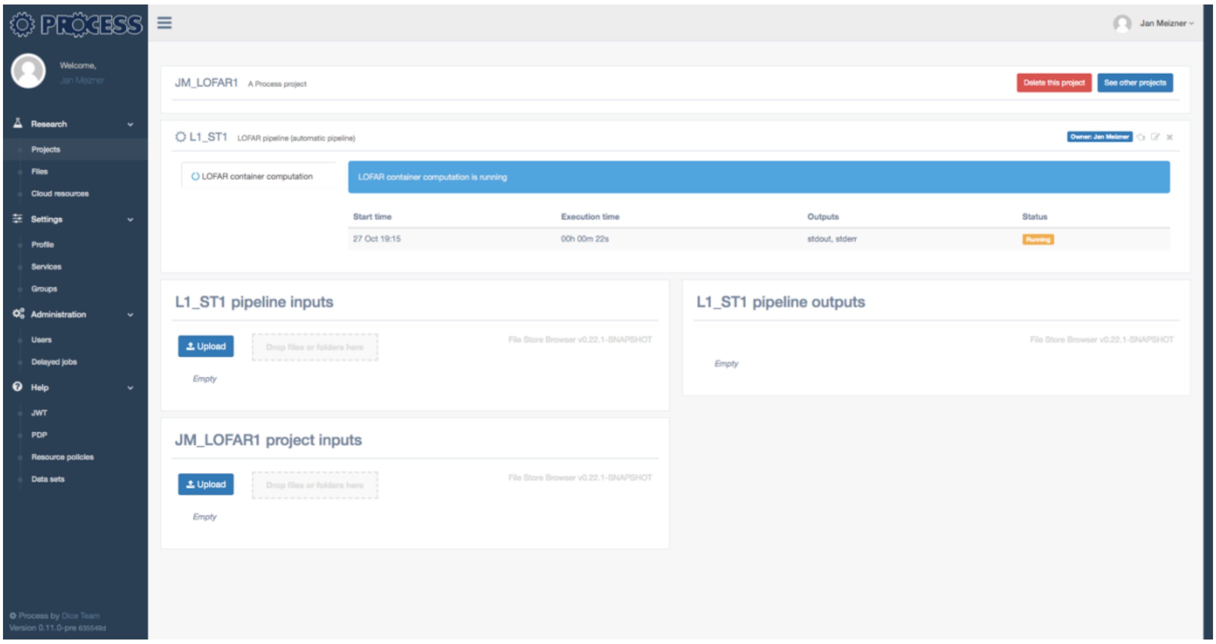

Figure 11 UC#2 – Execution on HPC

Finally, the finished pipeline is shown in Figure 12.

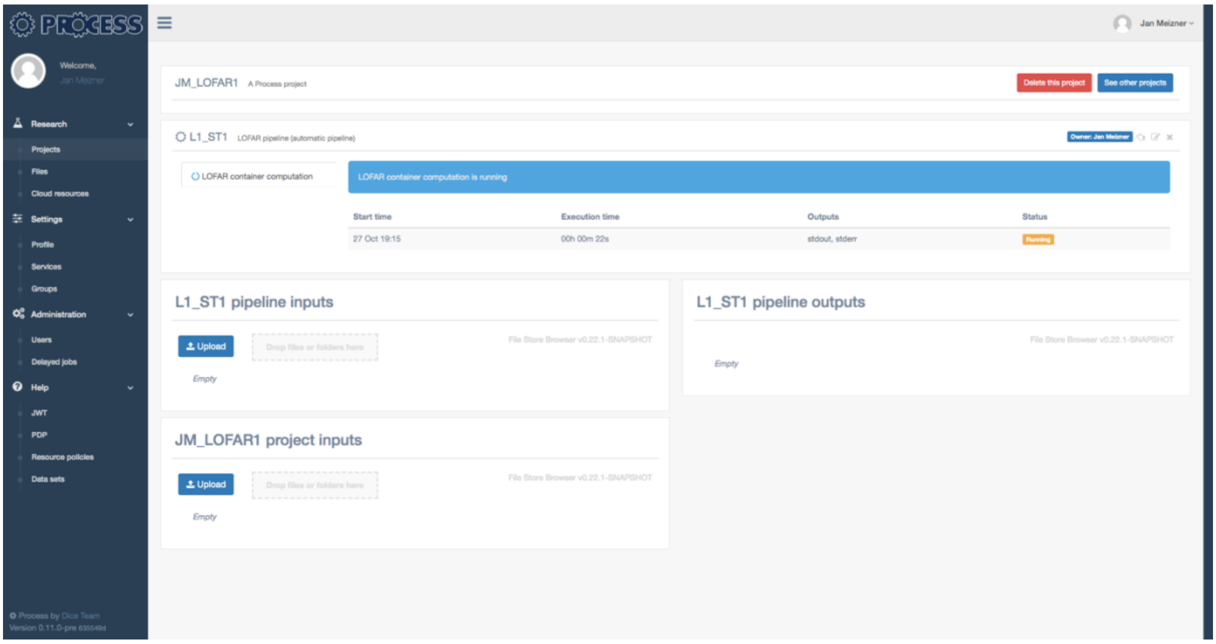

Figure 12 UC#2 – Completion

Appendix C: UC#4 scenario

UC#4 utilizes a different infrastructure, as it is reliant on cloud resources and scheduled via Cloudify. However, the workflow is similar to the previous two use cases thanks to the unification provided by the IEE platform. In the first step, the computation is preconfigured as shown in Figure 13.

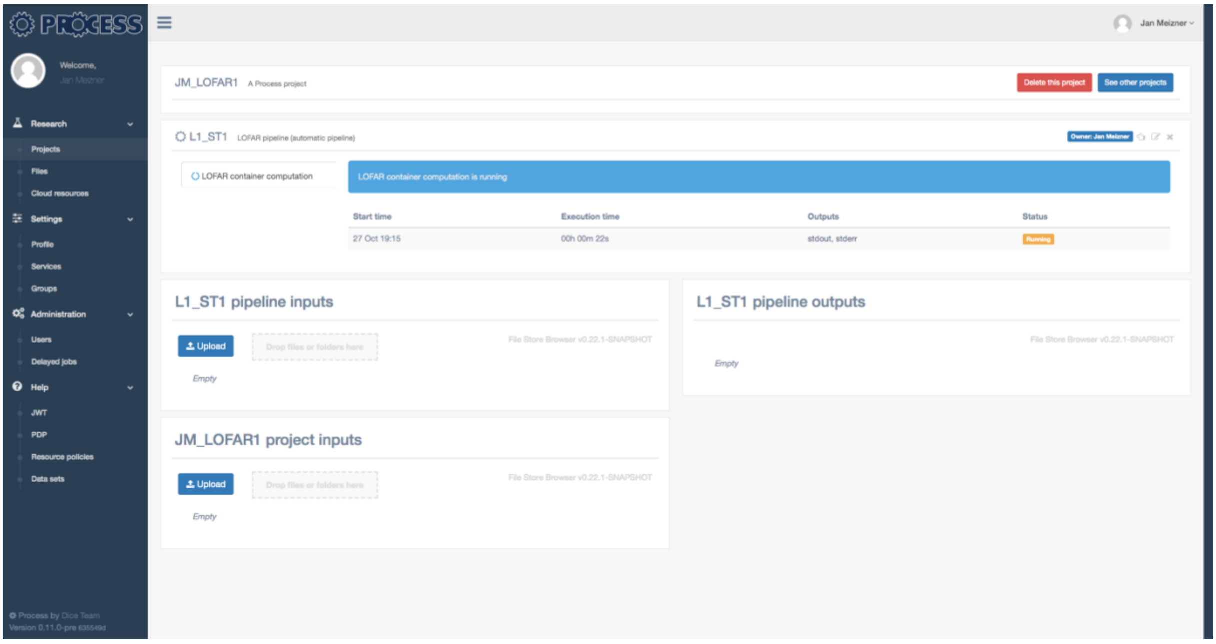

Figure 13 UC#4 – Initial configuration

Subsequent steps are similar to those which form part of the other use cases: they involve preparation (Figure 14), execution (Figure 15) and completion (Figure 16).

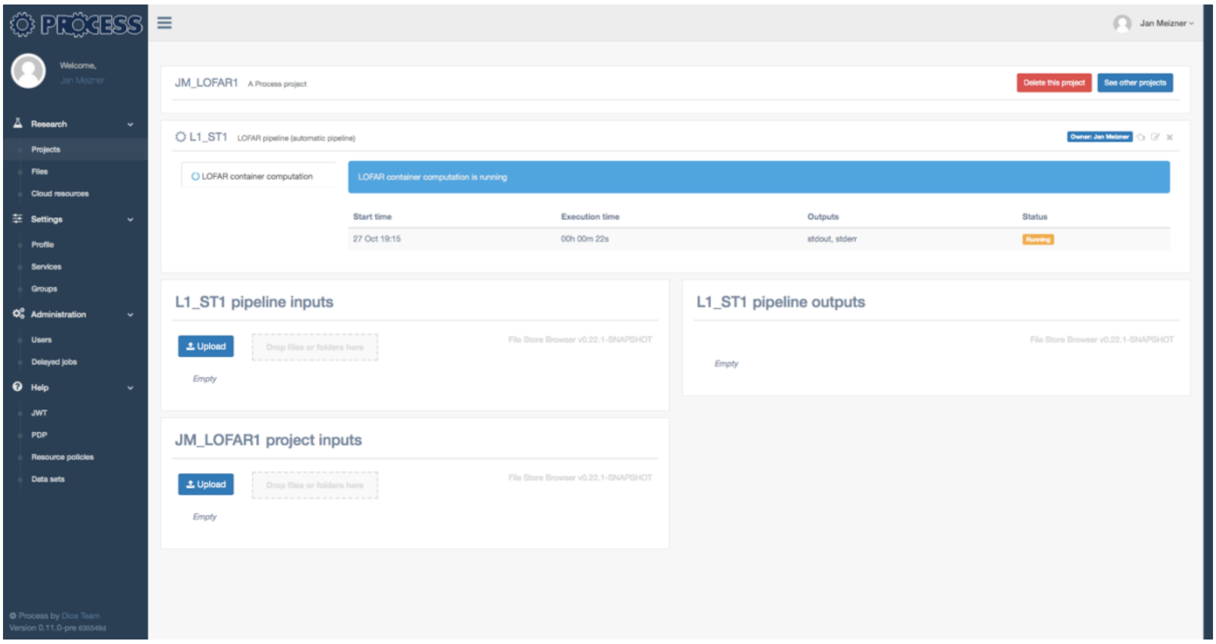

Figure 14 UC#4 – Preparation

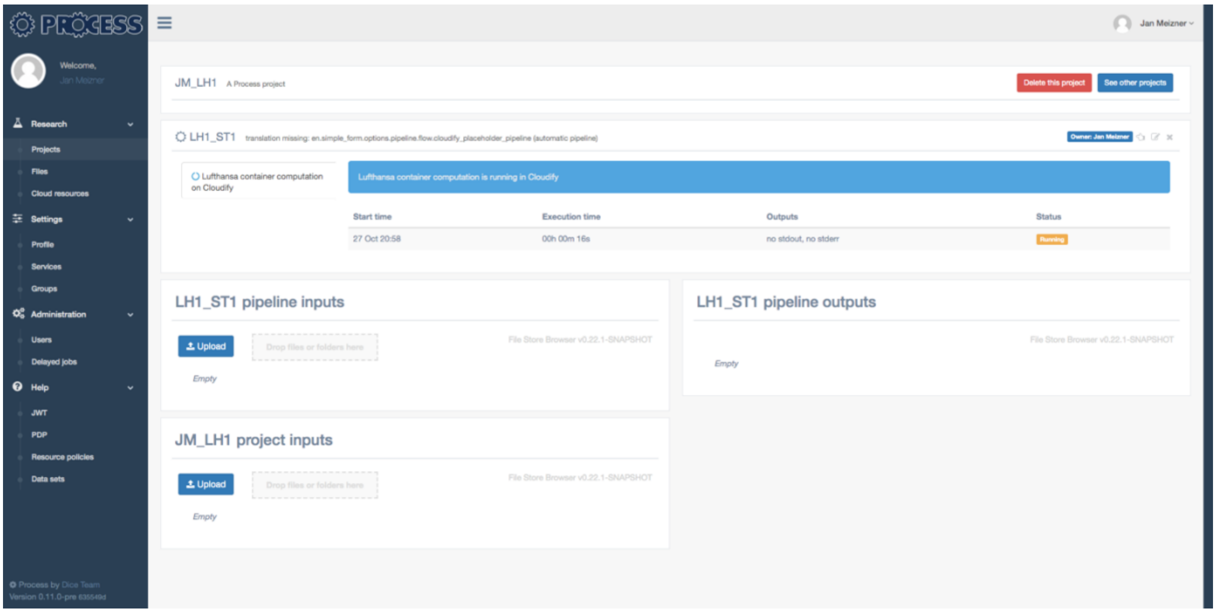

Figure 15 UC#4 – Execution

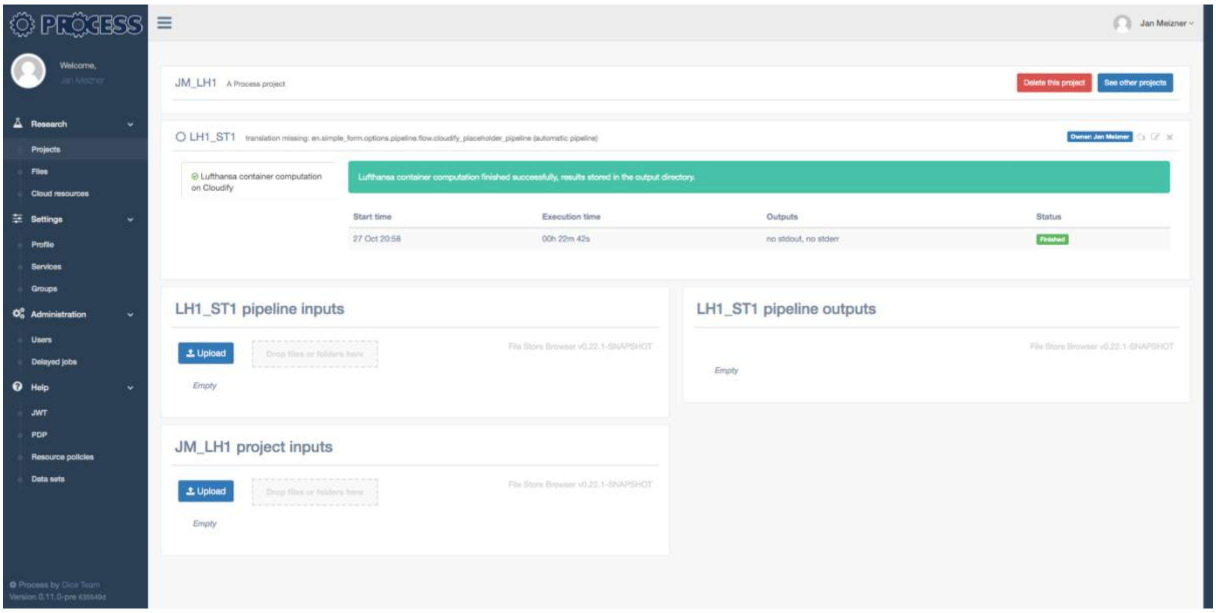

Figure 16 UC#4 – Completion

Appendix D: UC#5 scenario

The final use case presented here is based on yet another type of infrastructure – a cluster located at LRZ. As in UC#4, our goal was to provide similar user experience regardless of this heterogeneity. To this end, execution of this use case follows similar steps:

- Configuration – Figure 17
- Execution – Figure 18
- Completion – Figure 19

Finally, the generated results may be retrieved, as shown in Figure 20.

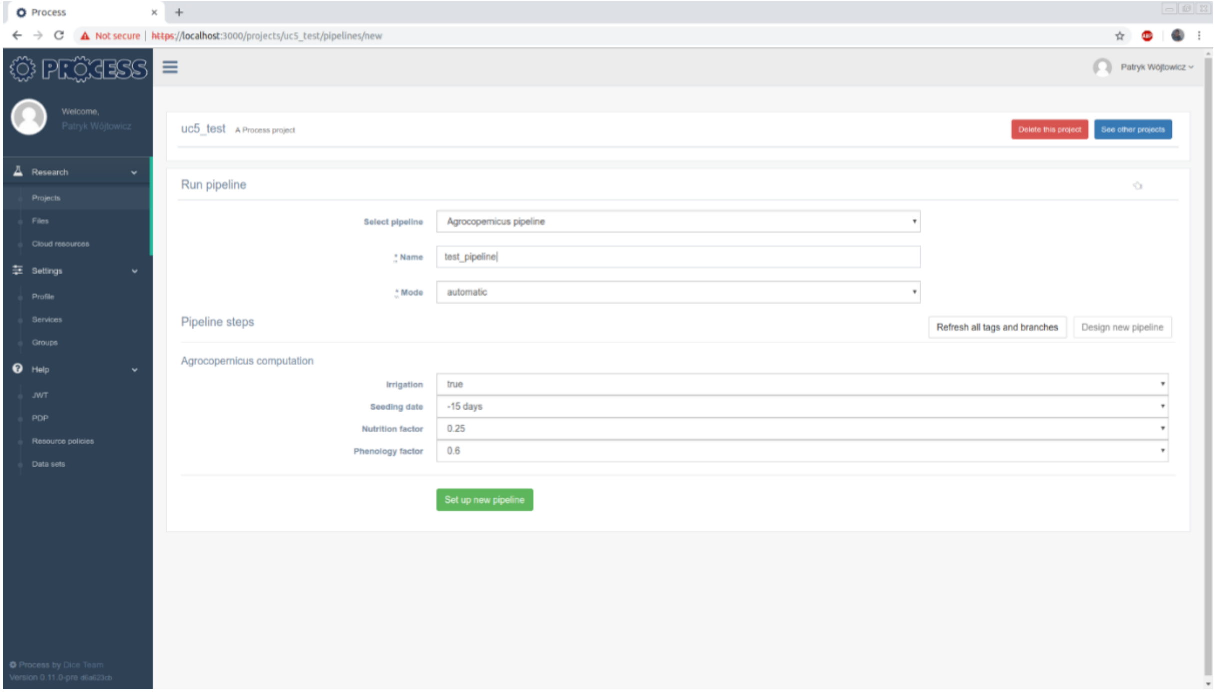

Figure 17 UC#5 – Configuration

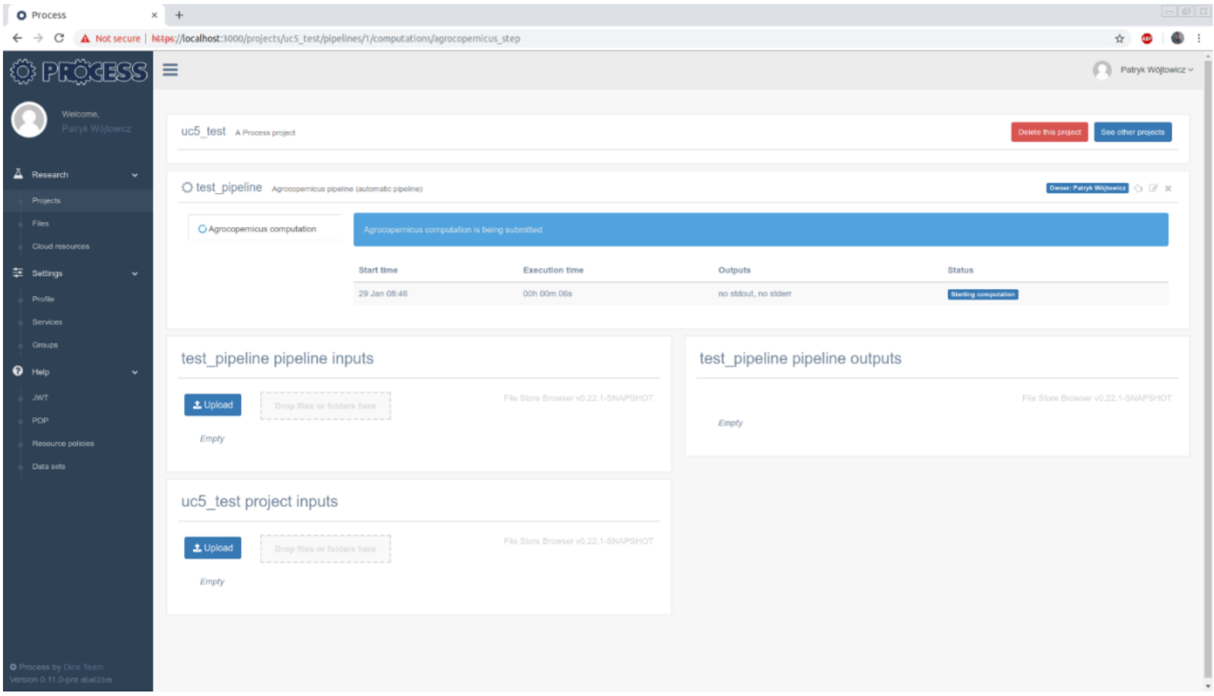

Figure 18 UC#5 – Execution

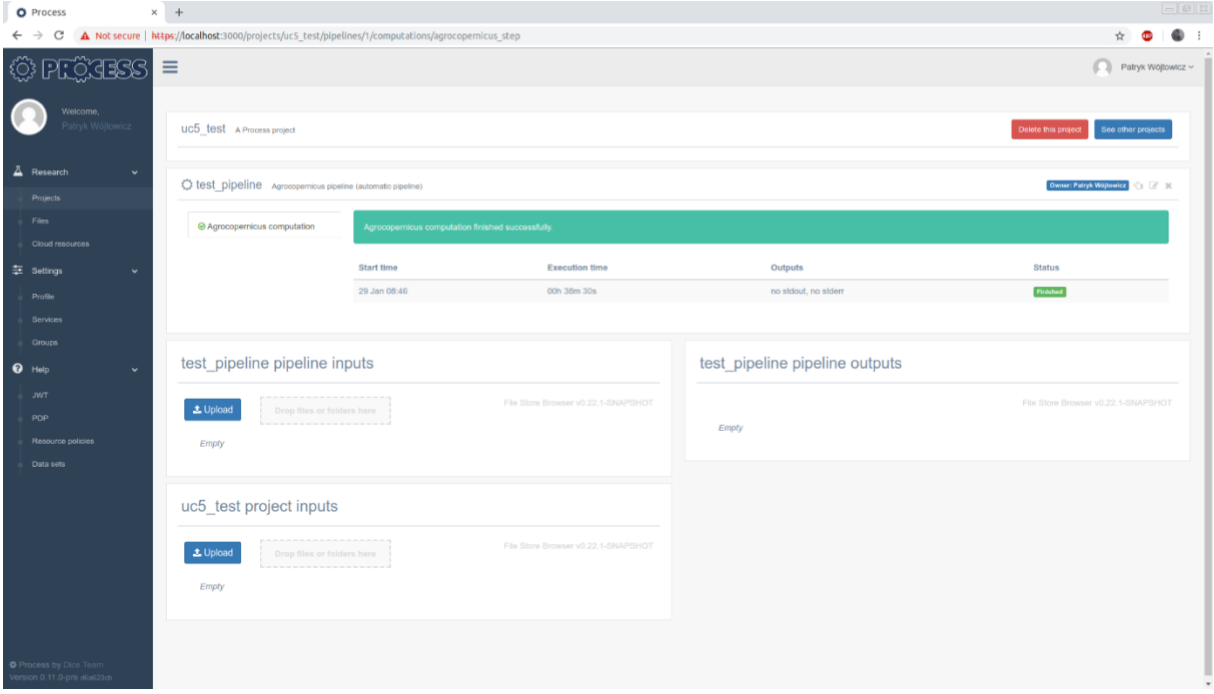

Figure 19 UC#5 – Completion

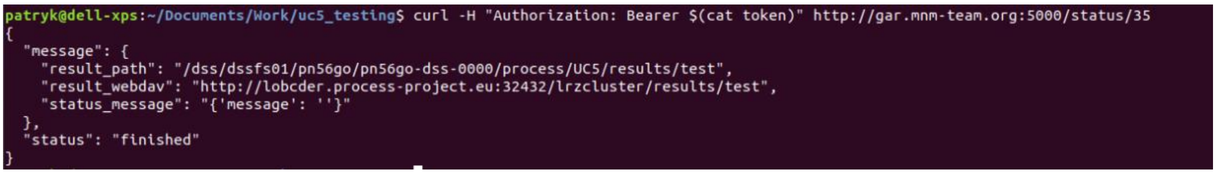

Figure 20 UC#5 – Final results