Deliverable 5.2

Project ID	654241
Project Title	A comprehensive and standardised e-infrastructure for analysing medical metabolic phenotype data
Project Acronym	PhenoMeNal
Start Date of the Project	1st September 2015
Duration of the Project	36 Months
Work Package Number	5
Work Package Title	Operations and Maintenance of PhenoMeNal GRID/Cloud
Deliverable Title	D5.2 A beta-version of PhenoMeNal integration VMI capable of proof- of-concept integration with other VMIs. Initial services online supporting PhenoMeNal data standards.
Delivery Date	M12
Work Package leader	UU
Contributing Partners	UU, IPB, EMBL-EBI, ICL
Authors	Ola Spjuth, Pablo Moreno, Pierrick Roger, Etienne Thévenot, Kristian Peters, Steffen Neumann, Christoph Steinbeck, Ken Haug, Gianluigi Zanetti, Pedro de Atauri, Tim Ebbels, Jianliang Gao

Abstract:
This deliverable presents the PhenoMeNal VRE, which in the first 12 months of the project has reached proof-of-concept stage in terms of integration between components. We describe the architecture of the e-infrastructure, the workflow systems chosen, and case studies demonstrating the capabilities of integrating services.

Table of Contents

1. EXECUTIVE SUMMARY

The PhenoMeNal VRE has reached a stage where proof-of-concept integration between VMIs and containers is operational, and where initial services make it possible to carry out pieces of analysis workflows within the VRE. This report covers the current status of the architecture contextualization, the development lifecycle, and present demonstrators for analysis workflows in the form of case studies. We have chosen to provision virtual infrastructures using contextualization tools (MANTL, Terraform, and Ansible) and demonstrated them to instantiate PhenoMeNal VREs on local hardware, private cloud installations, and public cloud providers. For the microservice architecture we employ Docker containers to enable tool isolation, and orchestrate them using Kubernetes and Mesos frameworks. As our workflow system we have selected to work primarily with the graphical engine Galaxy, but also the Jupyter online notebook for text-based workflow authoring, which is in line with the current and anticipated use in metabolomics research in several large centers, as described in *D4.1 Report on requirements for relevant research centers producing and/or consuming metabolomics data with respect to computational aspects, data storage, and infrastructural needs*.

We present a number of case studies that demonstrate the capabilities of the VRE, which are:

I. R-based metabolomics workflow
II. Fluxomics Tools
III. Statistical analysis of the sacurine data set
IV. Individual tools tested
    a. IPO
    b. MS-Convert
    c. NMR-Convert
    d. BATMAN

We have also placed large focus on documenting the e-infrastructure development and guides for users, with the PhenoMeNal wiki (https://github.com/phnmnl/phenomenal-h2020/wiki) as the main point of documentation; which also is mirrored as read-only on the PhenoMeNal website (http://phenomenal-h2020.eu/home/wiki/).

2. CONTRIBUTION TOWARDS PROJECT OBJECTIVES

The deliverable has contributed towards the following objectives:

- **Objective 5.1:** Establishment of the PhenoMeNal e-infrastructure
- **Objective 5.2:** Operations and maintenance of the PhenoMeNal VRC portal
- **Objective 5.3:** Maintenance and provisioning of the PhenoMeNal services in the PhenoMeNal e-infrastructure

3. DETAILED REPORT ON THE DELIVERABLE

3.1. Introduction and overview

PhenoMeNal provides Virtual Research Environments (VRE) for interoperable and scalable metabolomics analysis. End-users, such as researchers and research teams, educators, SMEs, and any other type of user, will be able to create, on-demand and through a simple user interface, an environment of tools, services, data supporting their research needs. Hardware setup and software deployment required to operate these facilities are completely transparent to the VRE and hence the users can focus on the analysis and not the technicalities (see Figure 1).

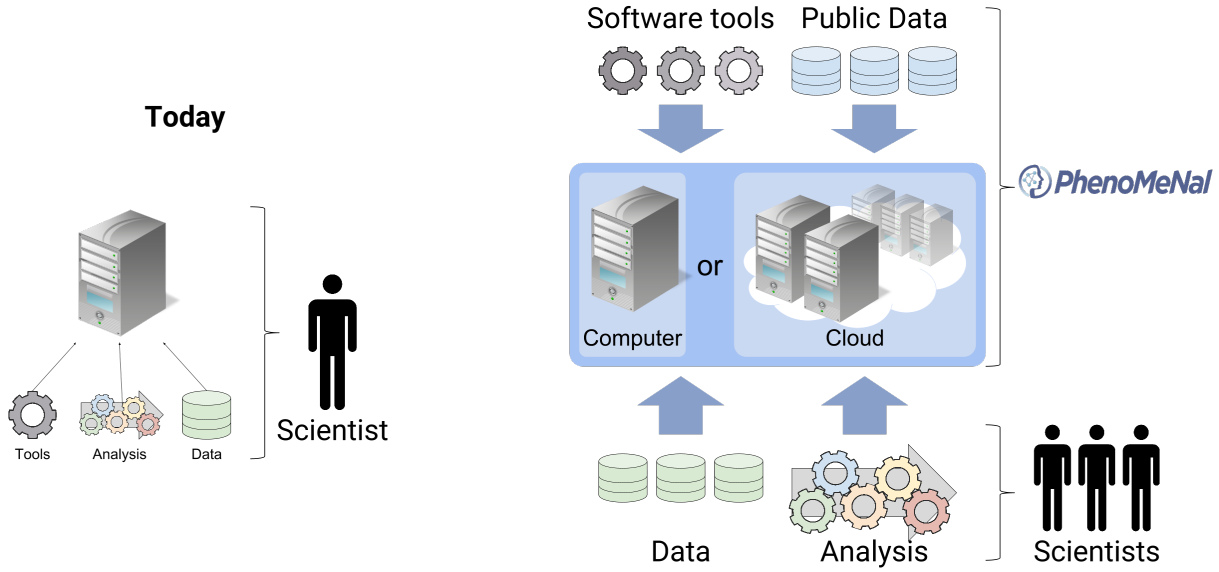

Figure 1: ***Responsibilities when carrying out contemporary metabolomics data analysis.*** *(Left:) Today's situation:* Scientists are responsible for everything, including the computer hardware, installing all necessary software, and carrying out the actual analysis. All execution is limited by the resources in the single computer.
(Right:) The PhenoMeNal approach: Software tools are available as containers without the need for installations, with data in agreed-upon interoperable file formats. The VRE can be started on single computers or on cloud resources, and the scientists benefit from only needing to deal with the analysis as the technical implementations are handled by the VRE.

3.2. PhenoMeNal Architecture

The PhenoMeNal Virtual Research Environment (VRE) consists of the following main components: 1) Software tools which are standardised and wrapped as software containers, 2) Standardised and interoperable data formats, 3) VRE contextualisation scripts to launch it on an Infrastructure-as-a-Service (IaaS) resources from public providers such as Google Cloud Platform, Amazon Web services; private OpenStack installations, or standalone computers.

PhenoMeNal implements a microservices architecture, where data analysis consists of connecting tools together to form an analysis pipeline. Since data formats are all agreed-upon and following open standards like mzML, nmrML and ISA-Tab, the communication and data sent between tools is simplified. Since tools are available as containers, they can be easily deployed without manual installation and dependency management, and containers can, in an elastic IT-environment, scale out to run analysis

in parallel on multiple compute nodes. All technical details are transparent for the metabolomics researcher.

The PhenoMeNal architecture and chosen implementations is depicted with a stack-based diagram in Figure 2. Every level of the stack builds on the one below it, and in turn supports the one above it. The arrows indicate a launch, from which software level they are used as well as what they implement. For example, Container Orchestration are run on the OS, and they manage the launch of Containers.

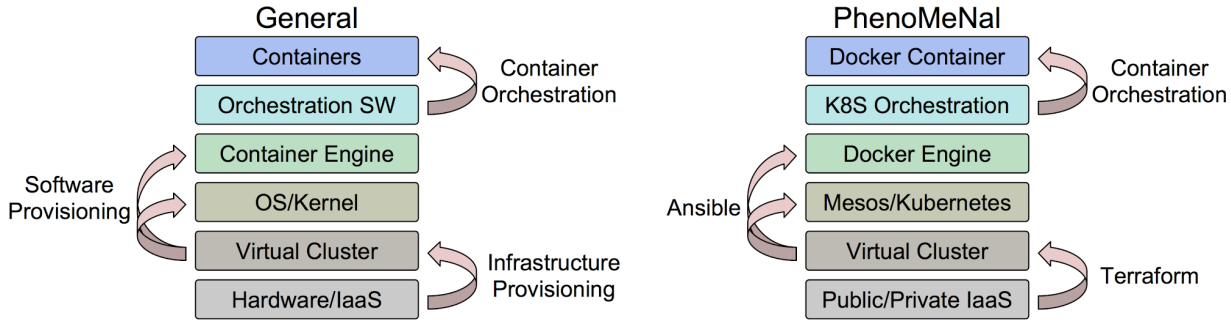

Figure 2: The PhenoMeNal architecture (right) with selected implementations, depicted as a stack diagram and aligned to general microservice-based architectures (left).

On the lowest level is the actual Hardware, be it a laptop on your desk or a virtual cloud running on a cluster miles away. The user makes use of *Provisioning Software* to prepare and equip the *Virtual Cluste*r with necessary software-layers. This often starts with a system kernel, which controls the very basics of the computer system. The kernel is the intermediary between the hardware (possibly virtual) and OS. It deals with resource management, load-balancing, runtime scheduling and more.

Every single node runs its own kernel and OS, with a Cluster OS layered on top as an abstraction-layer, making it appear as if all the nodes are part of one big computer. Combining the fundamental functions provided by the kernel with a Cluster OS of choice results in a virtual cluster with combined resources and the ability to split workloads between nodes as if they were all part of the same physical machine. The operating system then takes over and handles most of the communication.

With the operating system in place the desired services can be installed. In order to be able to mount and run containers containing microservices, a container engine is needed. The main function of it is supporting the launching, scaling, management and

termination of its auxiliary containers. It is through the container engines API that all container orchestration software operate.

Containers are pieces or parts of a program running within a closed virtual environment containing only the files needed for it to function. This makes a container entirely independent of the surrounding software environment, which is advantageous because it can be moved to and run on any operating system having the required container engine. In this use-case where microservices are wrapped up in software containers this means they are easy to add, remove and rearrange for the desired workflow.

The microservices run within these containers are all independent functions, usually from existing software packages. Containerizing these functions comes with several benefits, where their quick launch is one of the most important. This results in fast and simple scalability as required, since additional virtual nodes can be added to the virtual cluster, provisioned with all the software needed and then supplied with the necessary container. In a fraction of the time it would take to build, configure and install additional physical machines, a virtual cluster can accommodate for heavier workloads.

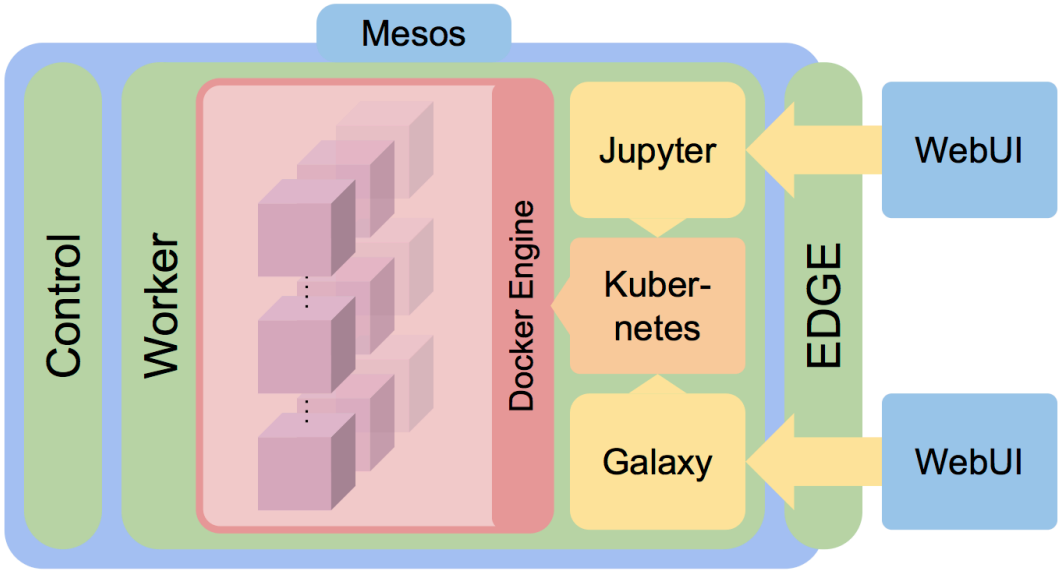

Figure 3: Overview of the interacting components inside a running PhenoMeNal VRE deployed using Mesos. The control nodes are redundant services enabling the functions of the systems in a fault-tolerant way. The Edge nodes manage the network connection with the user. PhenoMeNal currently uses Jupyter and Galaxy as graphical web front-ends with traffic routed through these Edge nodes. Workflow engines such as Galaxy, manage dependency graphs and communicates with Kubernetes that handles the orchestration of containers using the docker engine.

The PhenoMeNal stack

PhenoMeNal is built to run on your private machine as well as with any Infrastructure-as-a-Service-provider. It uses the MANTL suite of tools for most of its functions which means Terraform is the infrastructure builder of choice. It gives the user simple script-based control over the launch and management of their infrastructure. Ansible is used as provisioning software, installing both kernel, operating system and engines. Mesos and/or Kubernetes gathers the cluster of nodes to a single workspace and functions as its kernel and OS. Docker is the container-environment of choice and Ansible supplies its engine along with the dependencies. Kubernetes and Mesos functions overlap but within PhenoMeNal the main function of Kubernetes is container orchestration. The desired analysis functions are downloaded as small independent Docker containers and mounted through Kubernetes' orchestration tools. Figure 3 shows an overview of the interacting components inside a running VRE.

3.3. Workflow systems in PhenoMeNal as integrators of other VMIs and containers

There are many different workflow engines used in Bioinformatics, and we do not wish to limit PhenoMeNal to a particular framework/technology but strive to be workflow-agnostic at the tool level. However, we have chosen to implement two reference user interfaces; one with a graphical workflow designer (Galaxy) and one with a textual workflow designer (Jupyter).

Galaxy

Galaxy (https://galaxyproject.org/) is a workflow environment tool developed by a large Bioinformatics community, mostly by people working in the context of Next Generation Sequencing (NGS) tools, but lately also including communities in the Proteomics and Metabolomics areas, such as Galaxy-P, Workflow4Metabolomics and Galaxy-M. As a workflow environment, it allows researchers with no programming ability to concatenate common bioinformatics tools to create pipelines or workflows. Galaxy uses the original code and binaries of those bioinformatics tools developed elsewhere, and provides tool wrappers for them so that the Galaxy's UI and API can interact with those tools.
In a classical installation, most tools would be executed serially on the same machine where Galaxy is running. This approach does not scale for the purposes of PhenoMeNal. In order to enable scalable analysis on multiple compute nodes using microservices, the PhenoMeNal consortium has contributed to the Galaxy project the

ability to connect Galaxy to Kubernetes (for details of this contribution see https://github.com/galaxyproject/galaxy/pulls?q=author%3Apcm32). This is done by the Galaxy Kubernetes Runner, which translates Galaxy jobs to Kubernetes jobs, waits for their execution, handles errors and signals Galaxy when it should collect results. The main requirement posed by the Kubernetes Runner is to have availability of a shared file system between Galaxy and Kubernetes, where Galaxy deposits data files and Kubernetes Jobs/Pods can read them, and where in turn Kubernetes Jobs/Pods leave results that Galaxy can later collect. The Galaxy Kubernetes Runner aims to be the least disruptive as possible, requiring normally no modification of the Galaxy tools to be able to work through it. The configuration of tools for the Kubernetes Runner is done only in a central configuration files instead, where metadata to link the tool to a container is added.

The availability of the Galaxy Kubernetes Runner means that we can now run Galaxy to build workflows that use the tools that we have pre-packaged as containers in WP9, and these tools are launched within a single or multi-node VRE when the user executes those workflows.

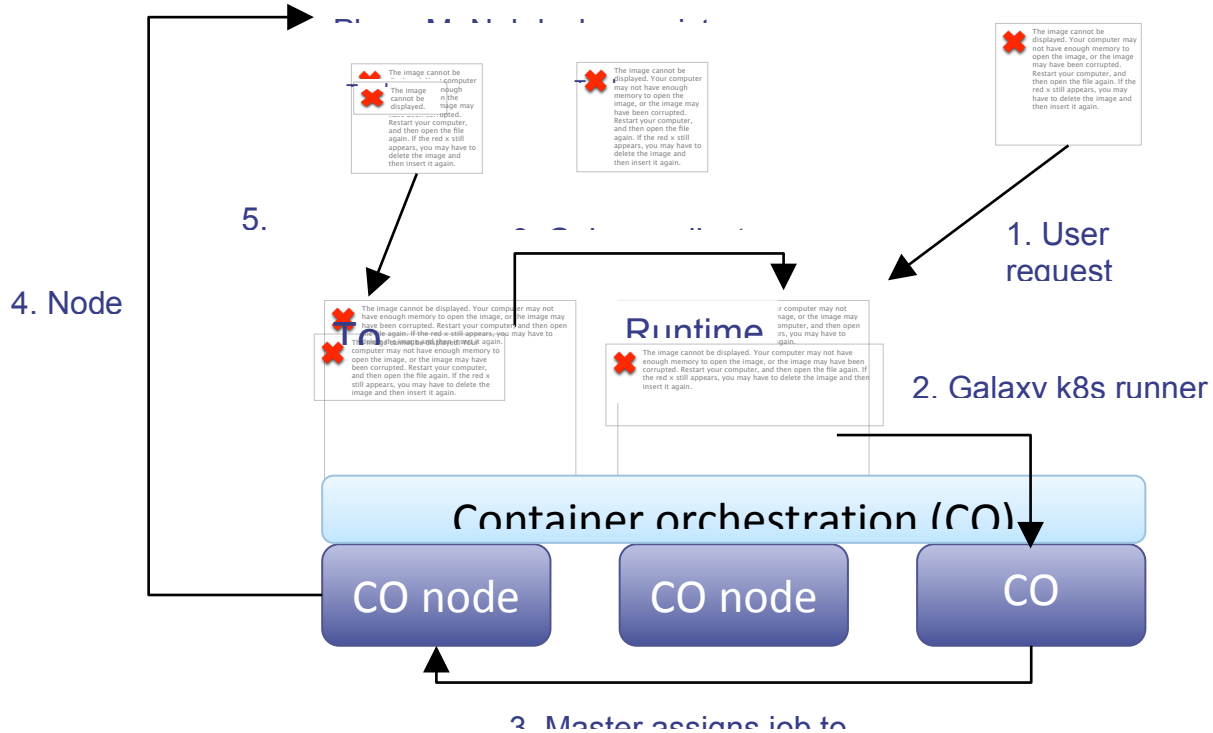

Figure 4: The flow implemented for deploying the Galaxy runtime into a Kubernetes (k8s) container orchestration (CO) system. Initially, (1) the user requests Galaxy (through its UI or API) to run a job with certain data. (2) Definitions added to our Galaxy instance allows the implemented k8s Runner for Galaxy to map the tool required in the job to a container. All this information is passed by the k8s runner for Galaxy to the master node of the CO in the form a of k8s Job API object using the pykube Python library to communicate. (3) The master node allocates the k8s Job to a node, according to availability of resources. (4) The node, using the Job definition, requests the required container (if not available) image from the PhenoMeNal docker registry. (5) The node, with the container obtained, runs the k8s Job, while the k8s Runner for Galaxy constantly queries to the master about the status of the job. (6) Once ready and signalled by the runner, Galaxy collects the results through the shared filesystem, once requests to the k8s master's REST API Endpoint shows that the job is done, and exposes them to the user.

Additionally, besides enabling Galaxy to communicate with the container orchestrator cluster, we have contributed the ability for Galaxy to run inside the container orchestrator, have PhenoMeNal tools provisioned to that installation (ie. the PhenoMeNal tools are available inside that instance of Galaxy) and send execution jobs from the inside to the Kubernetes container orchestrator through our pre-provisioned Galaxy docker image https://github.com/phnmnl/docker-galaxy-k8s-runtime.

Together, these developments mean that our only requirement to run Galaxy pre-provisioned with PhenoMeNal tools is to have a running Kubernetes installation with a shared file system for the nodes of that container orchestrator. Figure 4 explains the interaction between our installation of Galaxy, the container orchestrator and our containerised tools, available from the PhenoMeNal public docker registry. When the user requests a job to be executed on our deployed Galaxy instance, this installation has metadata for the tool that indicates to the Galaxy Kubernetes runner which container to fetch for the job and from where. That information is passed to the container orchestrator, this time as a Kubernetes Job object (translated from a Galaxy Job by the Kubernetes Galaxy Runner). The master node takes this specification, assigns it to a node, which in turns uses this object to know which container needs to be fetched from which docker registry, and which commands it should use for running. In the meantime, the Kubernetes Runner on Galaxy keeps asking for the status of the sent job to the master node, to trigger the collection of results by Galaxy when the job is done. Results are then presented to the user through the Galaxy interface. Tool containers and the Galaxy runtime container share an external file system through

Kubernetes' Persistent Volumes and Persistent Volume Claims API objects that allow them to have access to the same physical files.

Jupyter

Jupyter is a system to combine text (including e.g. mathematical equations) and code in an easy-to-read document that renders in a web browser (see Figure 5 for a screenshot). The code can be run directly from the notebook and display textual or graphical output, and there are a lot of kernels (i.e., backends) for many different types of programming languages and data analytics frameworks. The notebook itself is stored as a text file in JSON format, and the user simply needs to navigate to a URL using a web browser. Notebooks such as Jupyter have received a lot of attention lately for providing hands-on tutorial in e.g. programming and statistics, and is increasingly used in education as it gives students a complete working environment without the need to install anything on laptops or tablets.

Within PhenoMeNal, we use Jupyter as one of the ways of consuming the microservices developed within the consortium. When launching the VRE, users can open Jupyter and then either invoke services directly in an interactive fashion, or schedule long-running jobs using a workflow system of their own.

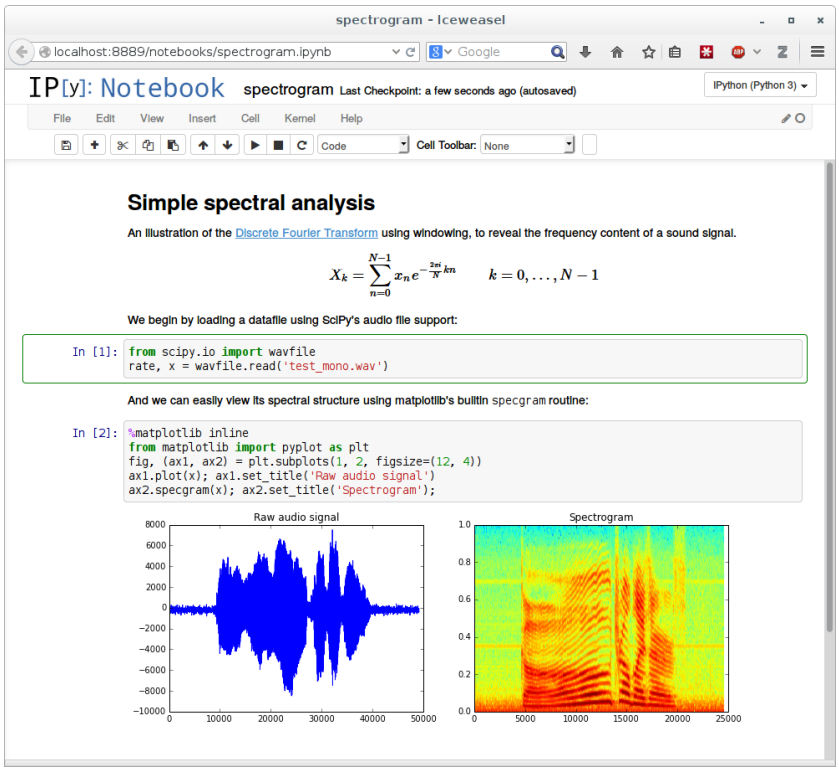

Figure 5: Screenshot of a Jupyter open notebook showing code for spectral analysis that is executed inline but remotely run on a server, and displaying graphical results directly in the browser.

3.4. Online Resources and services

Continuous integration system

PhenoMeNal hosts a Jenkins continuous integration system at https://phenomenal-h2020.eu/jenkins/ (see Figure 6 for a screenshot) which serves as an integration point where source code is collected, tools are built, containers are assembled, tests can be run to ensure correctness and interoperability, and where results can be pushed to public or private registries.

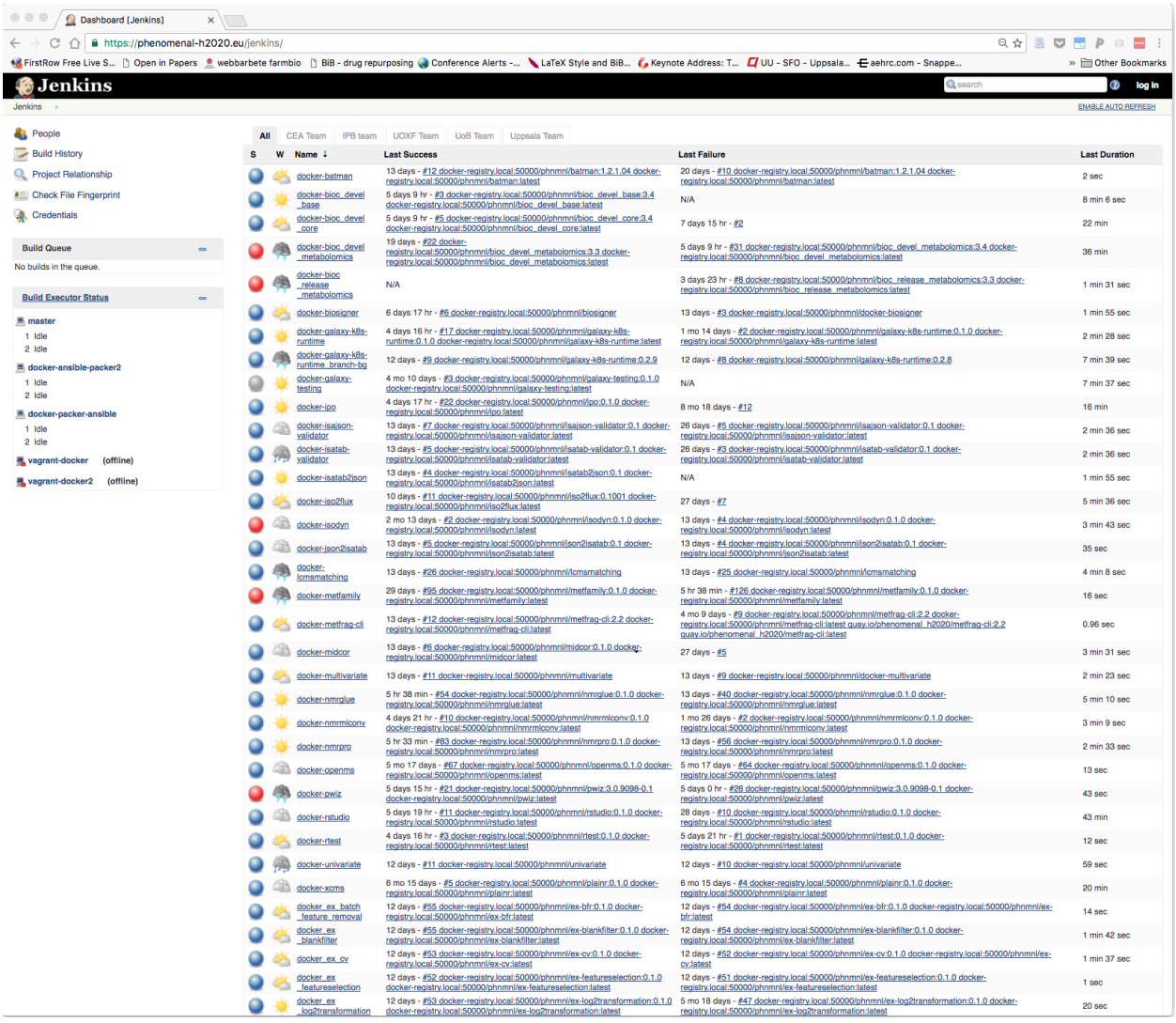

Figure 6: Screenshot from the PhenoMeNal continuous integration system Jenkins at https://phenomenal-h2020.eu/jenkins/.

Docker registry

PhenoMeNal hosts a docker registry to make containers publicly available for the research community. Currently we have 29 containers hosted on our docker registry at docker-registry.phenomenal-h2020.eu, listed in Table 1 below:

batman	galaxy-k8s-runtime	nmrglue
bioc_devel_base	ipo	nmrmlconv
bioc_devel_core	isajson-validator	nmrpro
biosigner	isatab-validator	pwiz
ex-bfr	isatab2json	rstudio
ex-blankfilter	iso2flux	rtest
ex-cv	json2isatab	univariate
ex-featureselection	lcmsmatching	
ex-log2transformation	metfrag-cli	
ex-merger	midcor	
ex-splitter	multivariate	

Table 1: List of docker containers in the PhenoMeNal docker registry at docker-registry.phenomenal-h2020.eu.

Any of these containers can be retrieved from any docker installation through the command:

```
docker pull docker-registry.phenomenal-h2020.eu/phnmnl/<container>
```

For instance:

```
docker pull docker-registry.phenomenal-h2020.eu/phnmnl/batman
```

Public Galaxy Instance

As an example of a VRE we have made the *PhenoMeNal Public Galaxy VRE* available at http://public.phenomenal-h2020.eu/. This instance of Galaxy (Figure 7) runs on top of a Kubernetes cluster of 3 nodes on the EBI EMBASSY Cloud (www.embassycloud.org,

running OpenStack). Each machine (8 cores, 16 GB RAM) runs a Kubernetes master container, a Kubernetes node container and the etcd node container on top of CoreOS. The Kubernetes cluster has access to a scalable GlusterFS shared file system, provided by 3 Ubuntu 16.04 VMs in the same tenancy.
The pre-provisioned PhenoMeNal Galaxy docker image (source available at https://github.com/phnmnl/docker-galaxy-k8s-runtime) is able to run inside a Kubernetes Replication Controller/Pod and communicates through the service account of Kubernetes with the master nodes to submit jobs to the cluster. This docker image contains as well all the tools that have been dockerized, "galaxified" and tested (currently manually, in the future via automatic integration tests in PhenoMeNal continuous integration system) with sample datasets to check that they work adequately.
Within this public instance, we provide shared workflows and data sets within Galaxy, that any user can try on the instance.

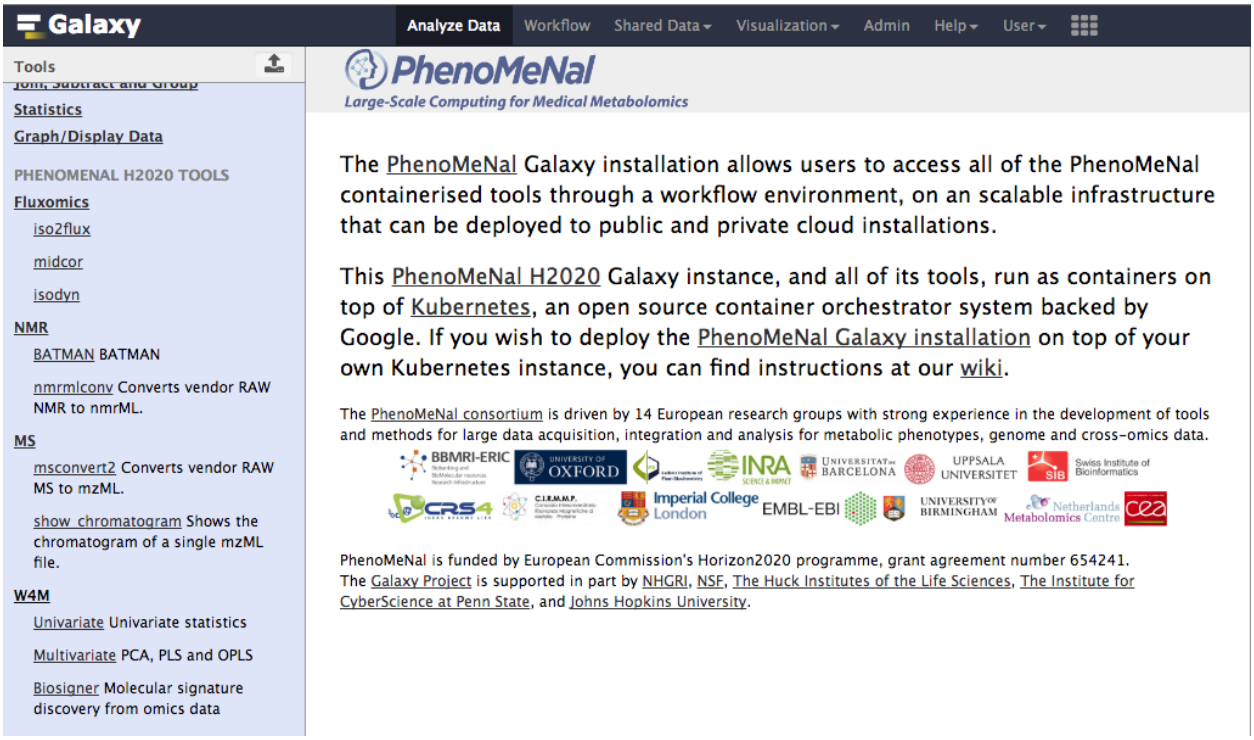

Figure 7: The initial page of the PhenoMeNal Public Galaxy VRE, with the currently available tools expanded on the left side.

In the future, we expect to have a selection step on the public VRE page, which should allow users to either launch Galaxy or Jupyter/iPython as runtimes to interact with the containerised PhenoMeNal tools.

3.5. Proof of concept demonstrators

Demonstrator 1: R-based metabolomics workflow

We containerized the necessary services (see Table 2) and developed a Jupyter notebook to demonstrate how an R-based metabolomics workflow from the Kultima group at the Department of Medical Sciences, Uppsala University (http://www.caramba.clinic/) could be executed within a PhenoMeNal VRE. As a job scheduler, we used the Chronos scheduler (https://mesos.github.io/chronos/) to construct a Direct Acyclic Graph (DAG) to define the workflow (see Figure 8). Each node in the DAG represents a microservice that performs a specific task. Once the DAG is properly setup, Chronos will figure out the dependencies between the various microservices, running them in the correct order, and keeping them alive only for the time they are needed. Furthermore, independent microservices are run in parallel.

The aim of the pipeline is to:

1. Remove contaminants present in blank samples;
2. Remove batch specific features;
3. Transform the intensities to the log2 base scale;
4. Perform variable selection based on coefficients of variation (CV) on metabolites across replicates.

Table 2: Containerized services used to carry out Demonstrator 1.

Service name	**Functionality**
Blank filter	*Contaminants Removal.* Remove the contaminants detected in blank samples (only DMSO) from all other samples
BatchfeatureRemoval	*Removal of batch specific feature.* Remove features that have a coverage of 80% within one batch, but not in any other.
log2transformation	Transform the data to log2 scale
Splitter	Splits data into subsets for parallel execution
CV	Calculate the coefficient of variation
Merger	Merges several files into one, columnwise
FeatureSelection	Extract stable features, based on the median coefficient of

	variation

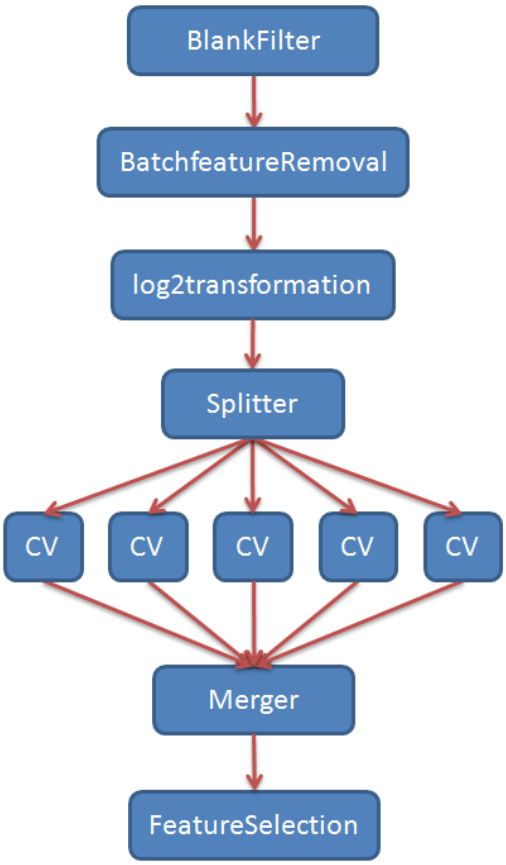

Figure 8: Visualization of the workflow in Demonstrator 1 for an R-based metabolomics workflow. Each box is a microservice that is executed in the VRE.

Demonstrator 2: Fluxomics Tools

The team of Prof. Marta Cascante at the University of Barcelona, one of the PhenoMeNal partners, develops tools and methods for the study of metabolic networks based on ^{13}C-tracer mass spectrometry. These approaches allow the researcher to follow fluxes of metabolic reactions in a reconstructed metabolic network. In close collaboration with them, we have created containers for their tools midcor, iso2flux and isodyn, and further, created wrappers for them to be available in Galaxy. These tools, together with the ProteoWizard container developed at IPB, permit the creation of two workflows in Galaxy:

- Stationary-state fluxomics: uses midcor and iso2flux
- Dynamic fluxomics: uses midcor and isodyn.

The first pipeline has been tested on Galaxy running on top of Kubernetes (the container orchestrator) with test data provided by the Cascante group. The second pipeline is awaiting some changes to midcor and isodyn regarding the standard used for tracer data, which is being currently discussed by members of U. of Barcelona Team, EBI and U. of Oxford, but each tool runs correctly on its own, requiring isodyn to incorporate some updates. As Figure 9 shows, tools are invoked through docker containers for each tool, which are pulled on request from the phenomenal docker registry. Table 3 shows the functionality of each of the tools used in the Fluxomics pipelines

Table 3: Containerised services for Demonstrator 2.

Service	**Functionality**
midcor	R-tool for primary ^{13}C data correction for natural isotope enrichment
iso2flux	Python-tool for steady state analysis of fluxes based on measured distributions of isotopologues (mass isotopomers)
isodyn	C-tool for dynamical analysis of fluxes based on time-series measured distributions of isotopologues

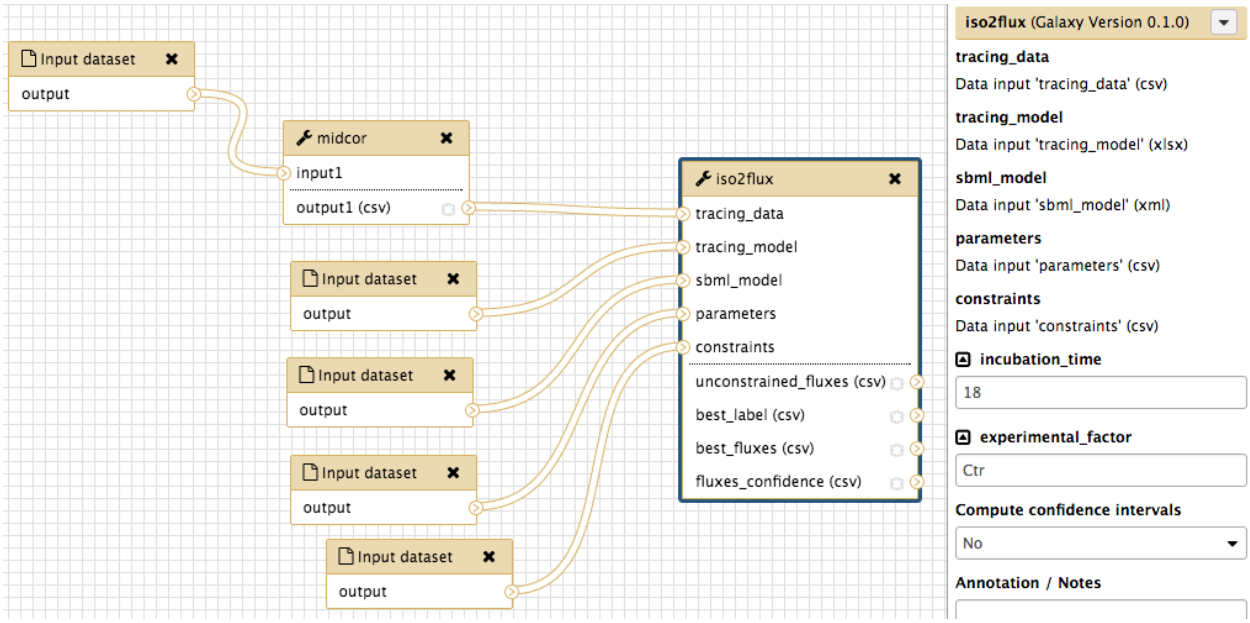

Figure 9: View of the Fluxomics workflow in Galaxy, in this case showing the tools midcor and iso2flux. Two trial datasets have been executed, running successfully both on local machine installations of the container orchestrator + Galaxy + tools stack, and on cloud deployments (EMBASSY Cloud Kubernetes deployment). To the right, the figure shows the settings used for the iso2flux module.

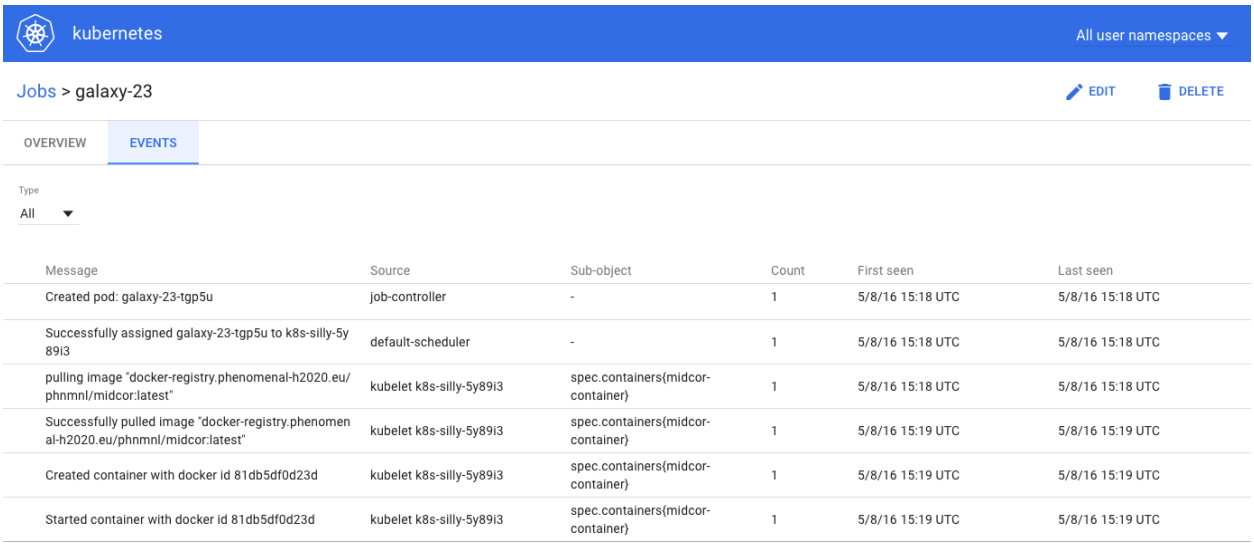

Figure 10: A screenshot of the Kubernetes dashboard showing the events associated to the run of the midcor job for one of the executions of the data sets on the EMBASSY Cloud. It can be seen that the job depends on docker image phnmnl/midcor:latest from the PhenoMeNal docker registry.

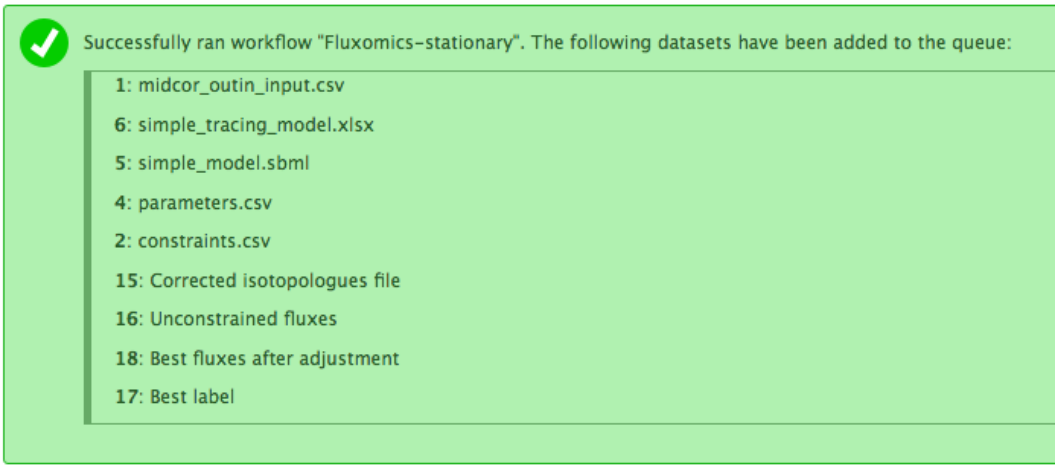

Figure 11: After a successful run of the workflow, Galaxy incorporates the outputs “Corrected isotopologues file”, “Unconstrained fluxes”, “Best fluxes after adjustment” and “Best label” into the history, for the user to be able to inspect and download those files.

The stationary flux workflow has been successfully tested on two local Kubernetes installations (Linux and Mac) and on a Kubernetes cloud deployment on the EMBASSY Cloud.

Data used were sample data sets obtained from experimentalist working at Professor Cascante’s lab. In summary: data acquisition was performed by a gas-chromatography/mass-spectrometry (GC/MS) detection analyses (Agilent 7890A GC / Agilent 5975C MS) by measuring stable isotope propagation from labelled glucose ([1,2-$^{13}C_2$]-glucose) to glycogen, RNA ribose, lactate and glutamate in Human Umbilical Vein Endothelial Cells (HUVECs) incubated for 40 hours under normoxic and hypoxic conditions.

The Fluxomics tools are available at the PhenoMeNal Public Galaxy instance, the workflow shared (Shared data menu, workflows item) to be accessible for users within the instance (login with user: *test-user* password: *galaxy*) and the data files shared as well (Shared data menu, histories).

Demonstrator 3: Statistical analysis of the *sacurine* data set

Characterization of the physiological variations of the metabolome in biofluids is critical for biomarker discovery, to avoid confounding effects in cohort studies. In this study, conducted by the CEA partner (from the MetaboHUB French Infrastructure for Metabolomics), urine samples from 183 adults were analyzed by liquid chromatography

coupled to high-resolution mass spectrometry (LC-HRMS). After pre-processing of the raw files, a total of 258 metabolites were identified at confidence levels provided by the metabolomics standards initiative (MSI) levels 1 (directly identified via reference standards) or 2 (identified by similarity using in-house and public databases). To study the physiological variations of these metabolites with age, body mass index, and gender, a dedicated statistical workflow was developed by E. Thévenot's team, combining univariate and multivariate statistics (Thévenot et al, 2015; Rinaudo et al, 2016). Raw files (in both Thermo proprietary and mzML open formats) are publicly available on the Workflow4Metabolomics computational infrastructure (Giacomoni et al, 2015) and the complete statistical workflow is publicly available with reference W4M00001b_sacurine-complete.

Table 4: Tools for the Sacurine data analysis

Service	**Functionality**
Univariate	Univariate hypothesis testing with multiple testing correction
Multivariate	Principal Component Analysis (PCA) and (Orthogonal) Partial Least Squares (OPLS) regression and classification
Biosigner	Feature selection methodology applied to the Partial Least Squares - Discriminant Analysis (PLS-DA), Random Forest, and Support Vector Machines (SVM), respectively

To demonstrate that this statistical workflow could also be run in the cloud environment from PhenoMeNal, the three modules (Table 4) were packaged in docker containers. The workflow (Figure 12) was then successfully run on two local installations of Kubernetes and one cloud installation (EMBASSY Cloud, Figure 13).

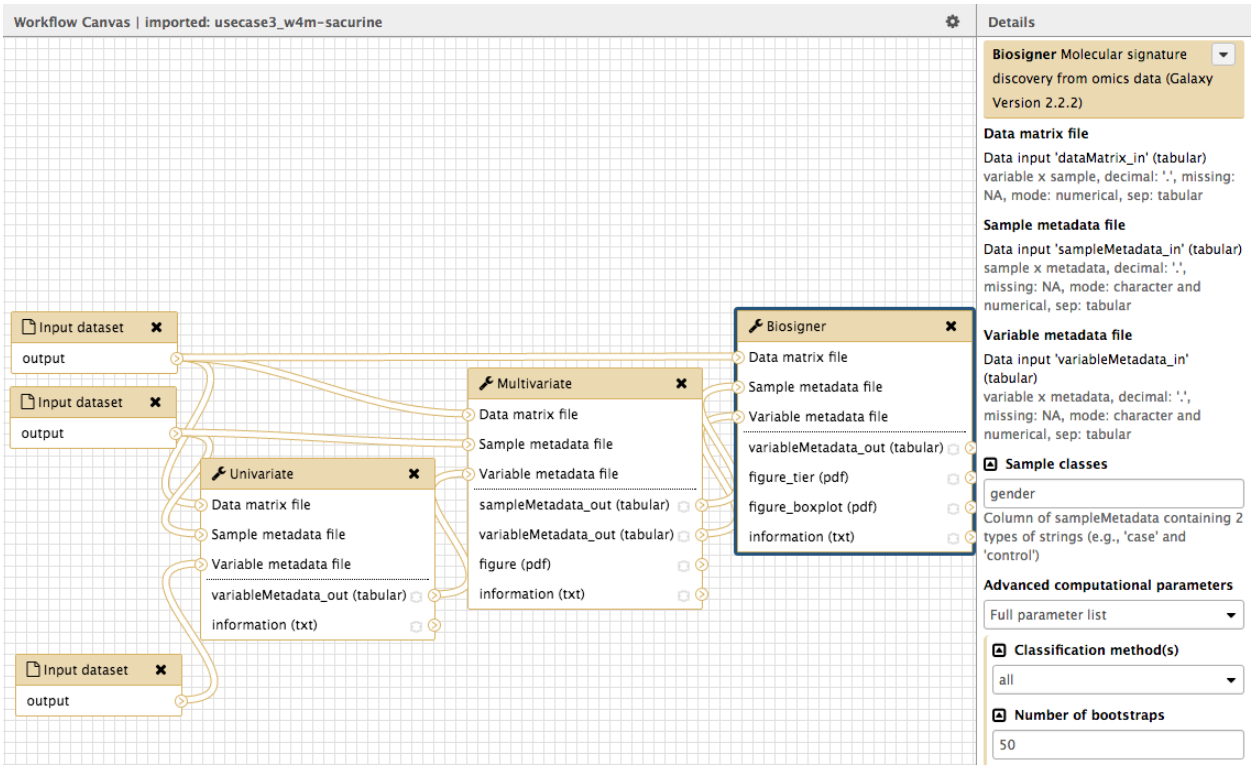

Figure 12: The statistical workflow from Workflow4Metabolomics for the analysis of the Sacurine data set was successfully packaged in docker containers and run on the EBI EMBASSY cloud installation.

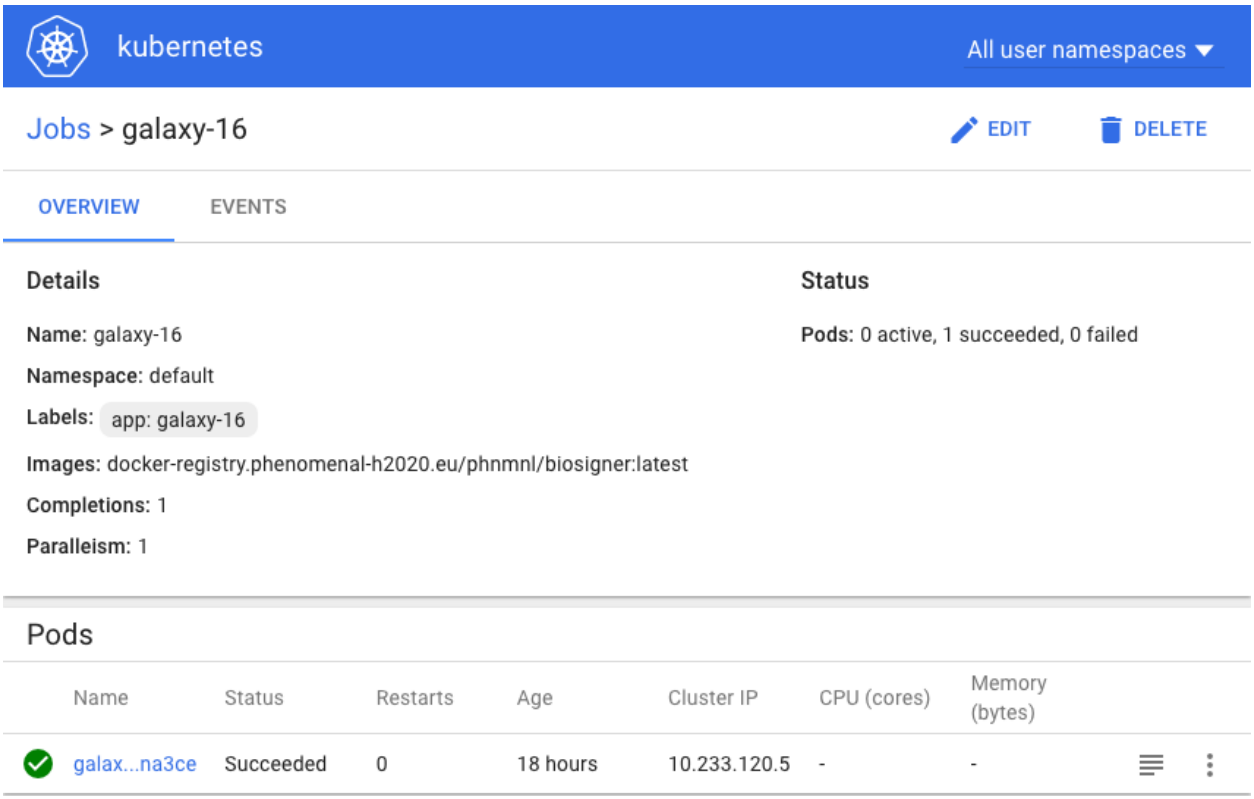

Figure 13: Screenshot of the Kubernetes dashboard showing a past successful execution of the last step of the pipeline, Biosigner, as part of a series of jobs that executed to run the complete presented workflow.

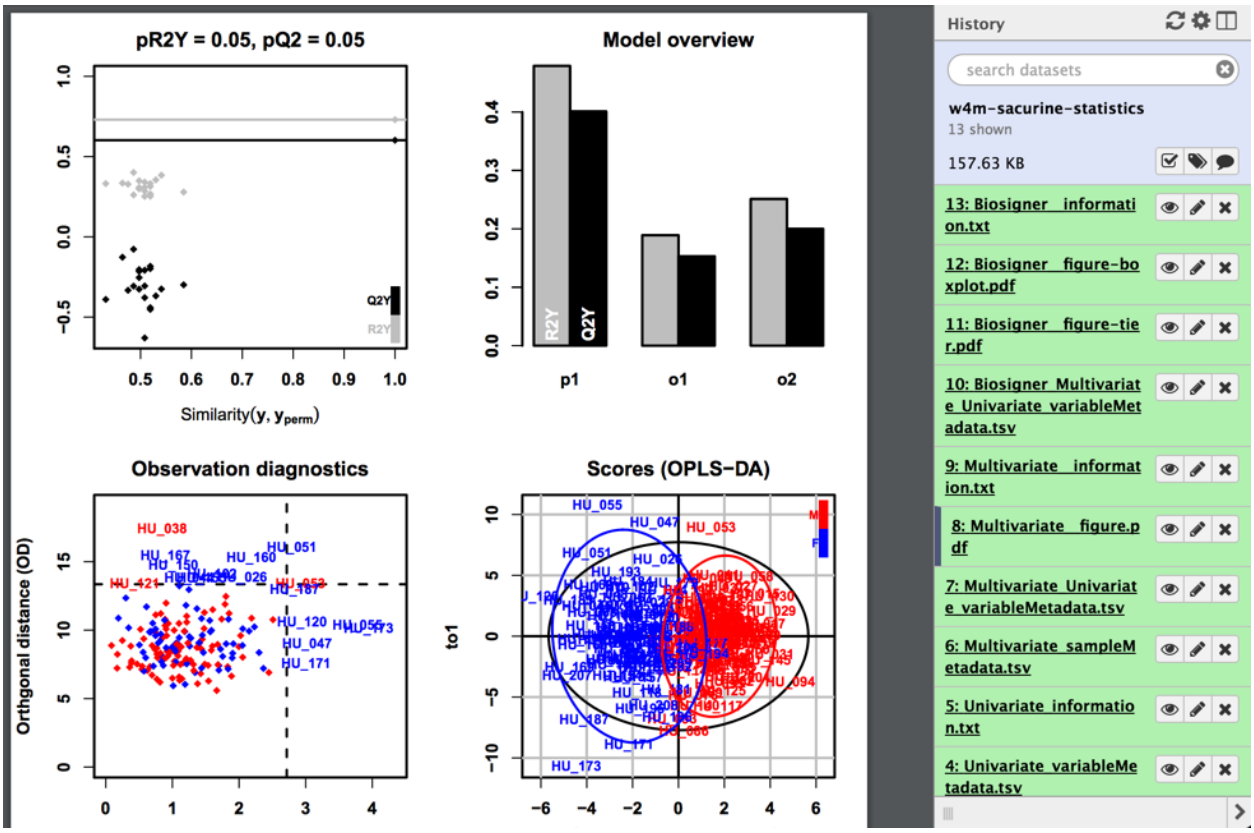

Figure 14: Example of Orthogonal Partial Least-Square - Discriminant Analysis (OPLS-DA) with the Multivariate module (*sacurine* data set) run on the Kubernetes installation on the EMBASSY Cloud. To the right, multiple output files can be seen listed; to the left, the different plots produced by the Multivariate module.

The *sacurine* demonstrators tools are available at the PhenoMeNal Public Galaxy instance, the workflow shared (Shared data menu, workflows item) to be accessible for users within the instance (login with user: *test-user* password: *galaxy*) and the data files shared as well (Shared data menu, histories).

Demonstrator 4: Isotopologue Parameter Optimization

Isotopologue Parameter Optimization (IPO) is a Tool for automated optimization of XCMS parameters for LC-MS. XCMS is a widely used tool for peak picking and retention time correction, but requires a number of parameters to be set, and suboptimal settings might produce biased results. IPO optimizes XCMS peak picking parameters by using natural, stable ^{13}C isotopic peaks to calculate a peak picking score. Retention time correction is optimized by minimizing relative retention time differences within peak groups. Grouping parameters are optimized by maximizing the number of peak groups that show one peak from each injection of a pooled sample. The different parameter

settings are achieved by design of experiments, and the resulting scores are evaluated using response surface models. IPO and XCMS developers are both external to PhenoMeNal.

Given that XCMS is computationally very intensive, so it is IPO, which runs XCMS several times for the design of experiment runs. IPO was tested with 4 complete LC-MS mzML data files from a MetaboLights study (MTBLS234: Automated Label-free Quantification of Metabolites from Liquid Chromatography–Mass Spectrometry Data), with a size each of roughly 200 MB. This was used to stress test the system and see how parallelization impacted on memory usage for such an intensive tool on a real data usage scenario (IPO is only normally run with sample of a few files from the a study, as done here). For this sake, IPO was containerised into a docker container and run on top of Kubernetes, with different settings of parallelization for its two main steps.

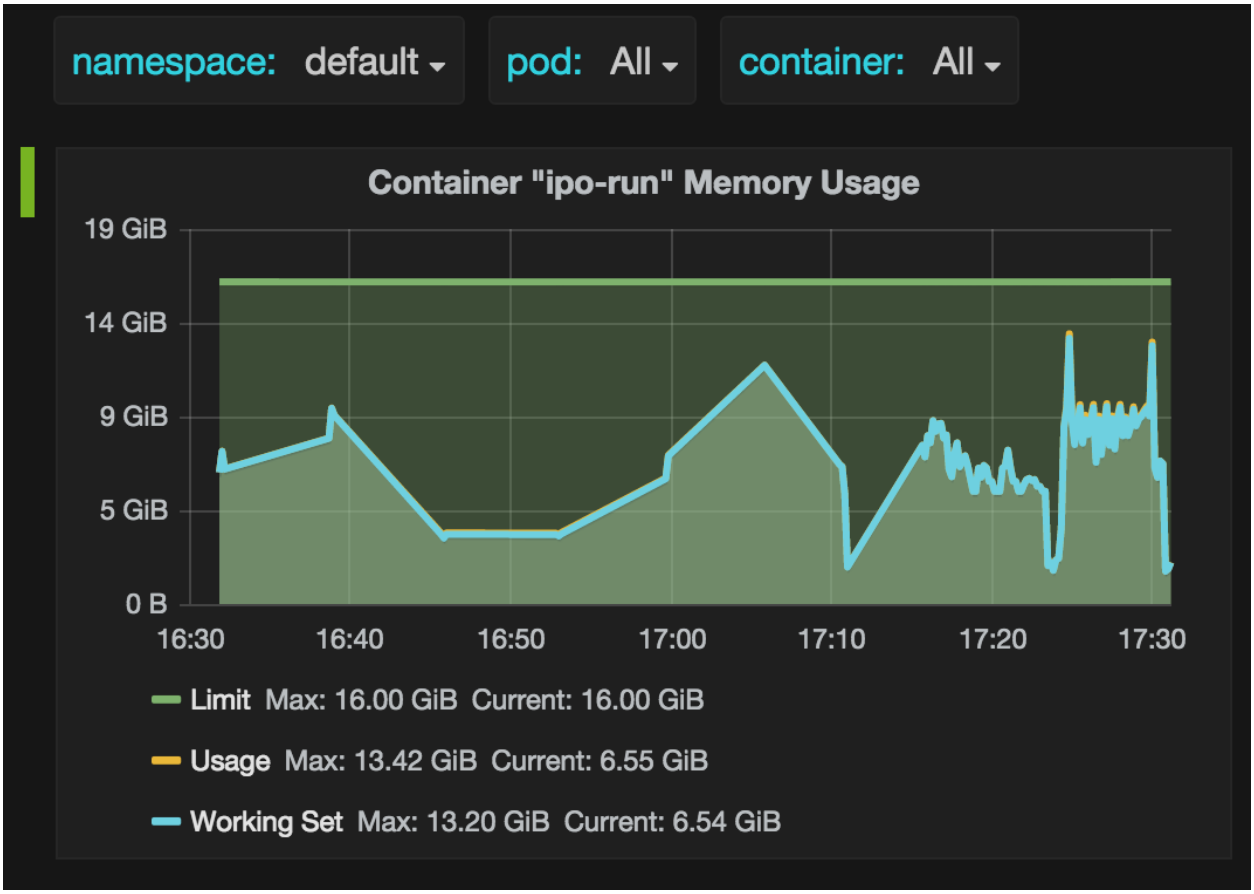

Figure 15: Profiling of memory usage for the containers to monitor the resource utilisation. Shown here is the IPO container while processing MetaboLights study MTBLS234 on an adequate number of threads, such that memory did not go past the

limit. This was monitored on top of a Kubernetes cloud installation, using Graphana/InfluxDB parallely to the execution of the IPO container (see Figure 15).

Main lessons learned were that the retention time design of experiments part (second part of the process) shouldn't be run with too many parallel threads, as its memory requirements grow relatively quickly. It was also detected that IPO was using a sub-optimal parallelization method in R (PSOCK), and an alternative was added through a this pull request https://github.com/rietho/IPO/pull/35. It will be also the case that very intensive tools might require provisioning of larger nodes in the PhenoMeNal VRE, or provide further options to the user to control number of threads to avoid getting out of memory errors.

Demonstrator 5: MS-Convert

Only a few open tools support the proprietary formats used natively by the mass spectrometry vendor software. As a result, the open mzML data format[1] was created and is now supported by a large number of community-developed metabolomics software[2]. Thus, the first step in a metabolomics data processing workflow with Open Source tools is the conversion to an open raw data format. One of the main routes to mzML-formatted data is using Open Source converter msconvert developed by the Proteowizard team[3], which is one of the reference implementations for mzML. In PhenoMeNal, the mzML data format forms the fundament on which all other mass spectrometry workflows will be built upon. That is why the successful conversion from a RAW vendor format to mzML is of utmost importance for the subsequent steps in any of the workflows.

We successfully deployed and run msconvert on MS raw data from Bruker instruments within the VRE on Embassy Cloud. It now can be used by all workflow tools that depend on and use the mzML data format (Figure 16). The tool can be tried out on the public PhenoMeNal Galaxy on http://public.phenomenal-h2020.eu/ with the "neg-MM8_1-A,1_01_376" test raw data set. Invisible to the actual user is the underlying cloud infrastructure. Starting the conversion does lift up a job, which is essentially a docker container, which is deployed in the EBI Embassy Cloud. This effort is being accomplished by using the kubernetes implementation of Galaxy (see above).

[1] Martens et al. (2010): mzML—a Community Standard for Mass Spectrometry Data. Molecular & Cellular Proteomics, 10. doi:10.1074/mcp.R110.000133

[2] Rocca-Serra, P., Salek, R.M., Arita, M. et al. (2016): Data standards can boost metabolomics research, and if there is a will, there is a way. Metabolomics 12: 14. doi:10.1007/s11306-015-0879-3

[3] Chambers, M. C. et al. (2012): A cross-platform toolkit for mass spectrometry and proteomics. Nature Biotechnology 30: 918-920. doi:10.1038/nbt.2377

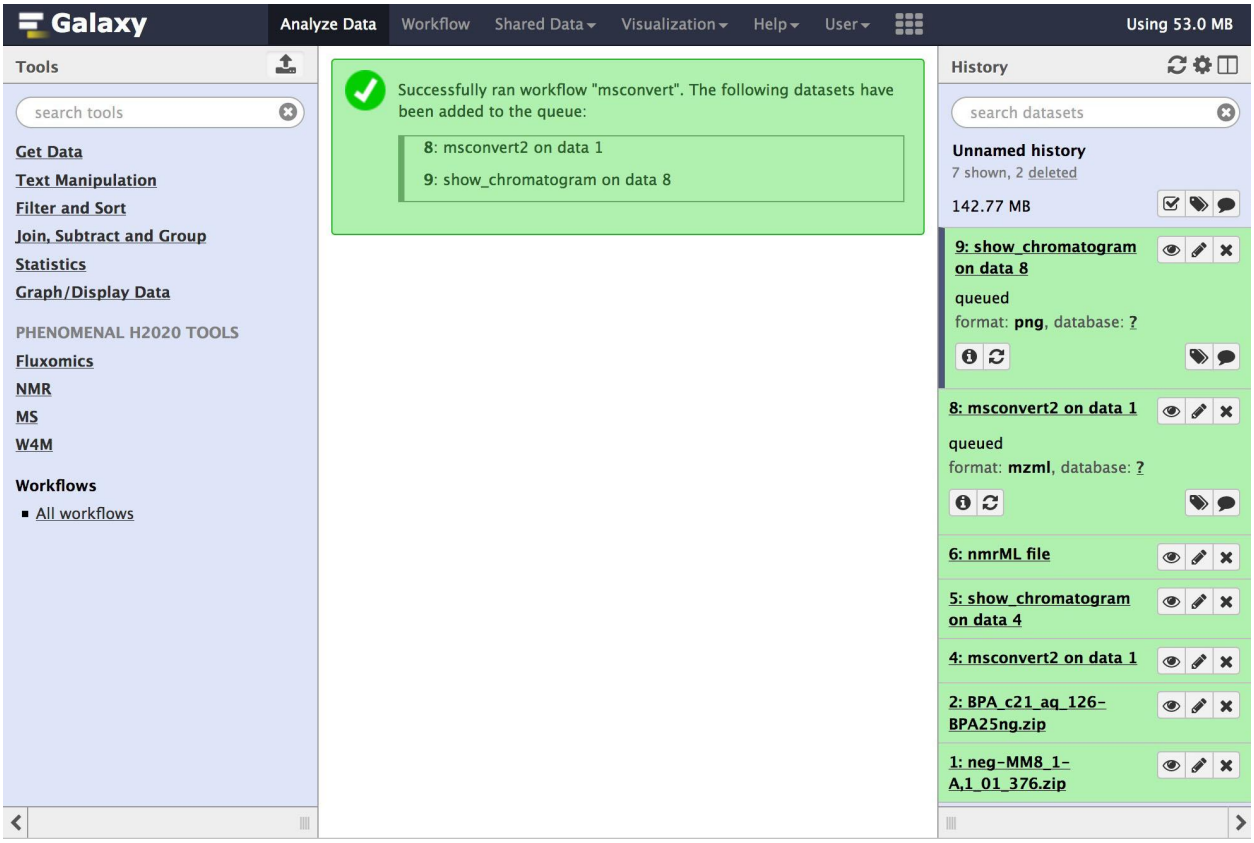

Figure 16: Screenshot of Galaxy running the D9.2.1 Preprocess VMI pwiz (tool msconvert and show_chromatogram are connected in a workflow in order to demonstrate that both tools can be deployed and run successfully in the PhenoMeNal eInfrastructure). The underlying complexity of the eInfrastructure (deploying and running containers in the cloud or local workstation) is hidden from the end user.

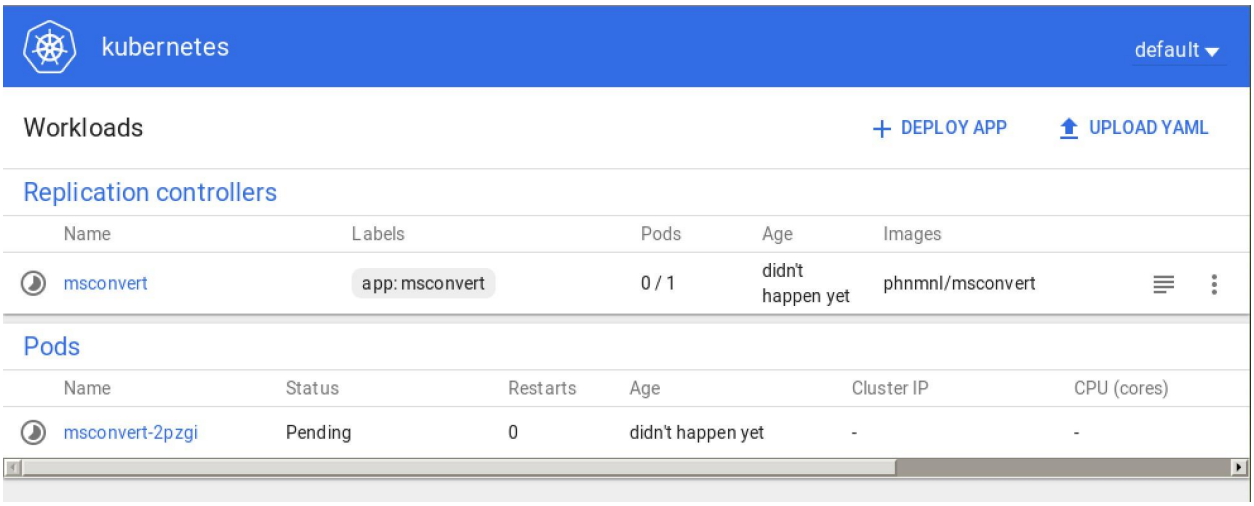

Figure 17: Screenshot of msconvert that is being deployed in a local kubernetes infrastructure environment. This screenshot demonstrates that from Galaxy we can now deploy and run container anywhere within the PhenoMeNal eInfrastructure.

Demonstrator 6: nmrML-Conversion

For NMR, the open nmrML data format is being developed in WP8, and it is supported by a growing number of community-developed metabolomics projects. The main route to nmrML-formatted data is using the Open Source converter nmrmlconv as part of the nmrML package[4].

We successfully deployed and run nmrmlconv on part data from the NMR Mus musculus data set[5] (which is part of Workflow4Metabolomics) within the VRE on Embassy Cloud. It now can be used by all workflow tools that depend on and use the nmrML data format. In an early effort, we succeeded in connecting the output of nmrmlconv to BATMAN (Figure 18). There is still work to be done to make the interplay between these tools smoothly and to deploy across the Embassy Cloud.

nmrmlconv can be tried out on the public PhenoMeNal Galaxy on http://public.phenomenal-h2020.eu/ with the “BPA_c21_aq_126-BPA25ng” test raw data set. Invisible to the actual user is the underlying cloud infrastructure. Starting the conversion does lift up a job, which is essentially a docker container deployed in the EBI Embassy Cloud. This effort is being accomplished by using the kubernetes implementation of Galaxy (see above).

[4] https://github.com/nmrML/nmrML
[5] http://workflow4metabolomics.org/node/48

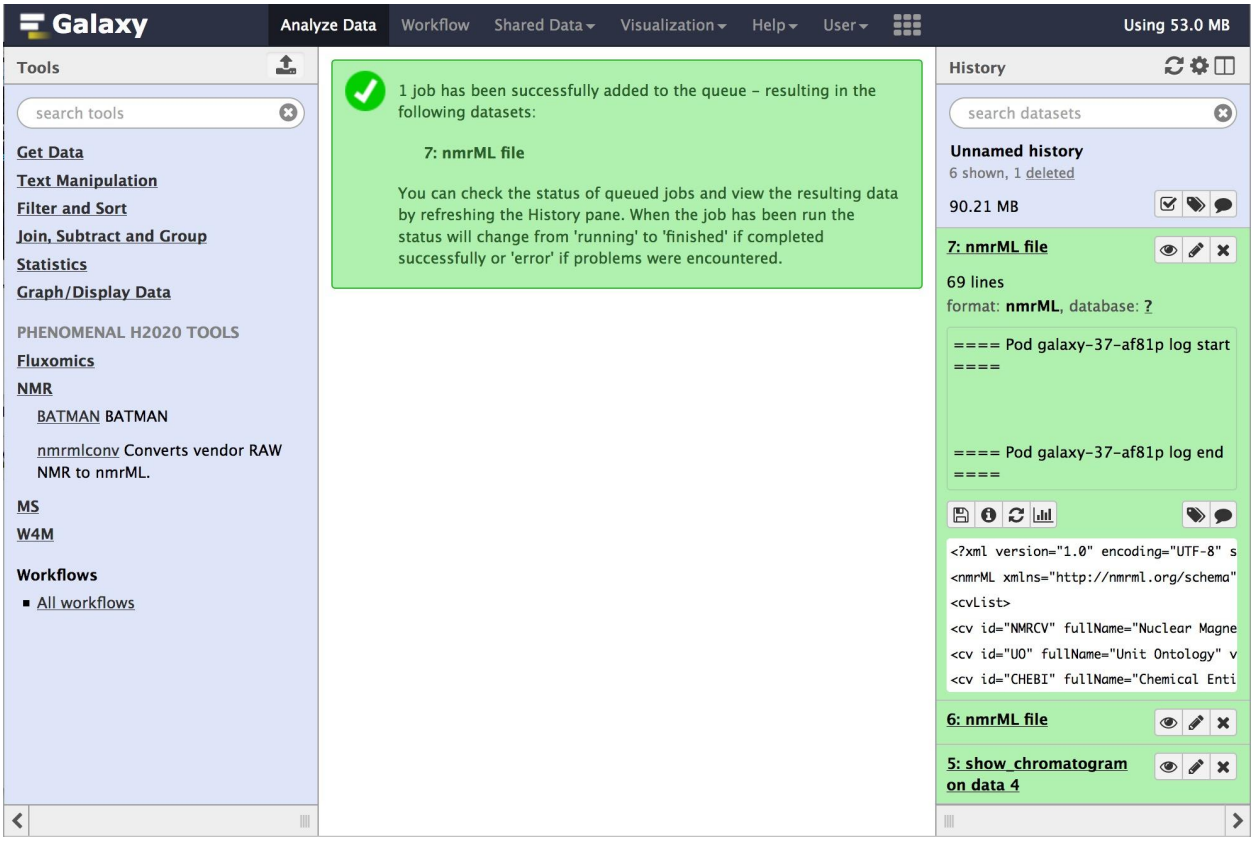

Figure 18: Screenshot of Galaxy running the tool nmrmlconv successfully. The underlying complexity of the eInfrastructure is hidden from the end user. However, the user can see whether the nmrmlconv tool has been successfully deployed in the cloud or local cloud environment by looking at the Galaxy logs (at the right hand side of the screenshot the text “pod galaxy-<id>-<hash>” indicates that the tool has been run as a container).

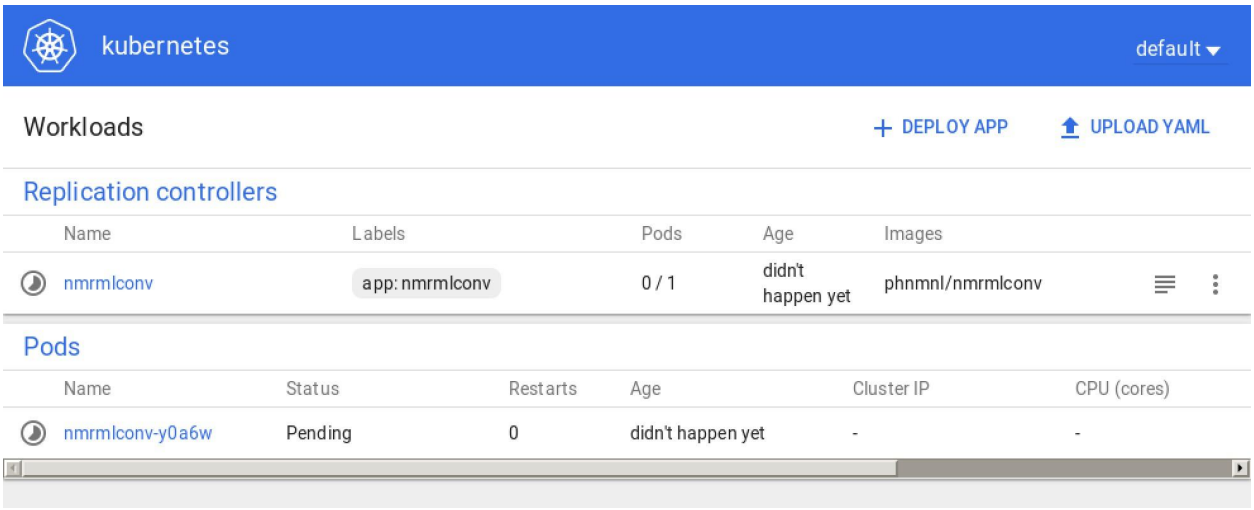

Figure 19: Screenshot of nmrmlconv that is being deployed in a local kubernetes environment. Currently, from Galaxy we can deploy and run nmrmlconv anywhere within the PhenoMeNal eInfrastructure.

Demonstrator 7: BATMAN NMR

BATMAN is short for Bayesian AuTomated Metabolite Analyser for NMR spectra. BATMAN deconvolutes resonance peaks from NMR spectra of complex mixtures and obtains relative concentration estimates for the corresponding metabolites automatically. This is achieved through a database of spectral profiles for known metabolites and a Bayesian Markov Chain Monte Carlo algorithm. Users have the options to specify the multiplet ppm position, position shift range, peak width range and so on. Parallel processing is available if processing many spectra. The algorithm is computationally intensive and thus is a primary candidate to benefit from use on distributed systems or cloud environments.

When the input spectra data are processed, BATMAN generates the output files including the logs of the processing and provides the resulting plots of peak fitting results. An example is shown below.

Currently, BATMAN takes input NMR data in tabulated ".txt", due to limitation of Galaxy platform. For non-dockerized BATMAN, it can take R data or Bruker (manufacturer proprietary) format files as well. . For .txt format, the filename of the NMR raw data is suggested to be (but not limited to) "NMRdata.txt". The other 3 input files as BATMAN running options must follow the naming rules (NOTE: case sensitive) as: "batmanOptions.txt", "multi_data_user.csv", and "metabolitesList.csv". The data and input files need to be prepared locally and uploaded to Galaxy before running BATMAN.

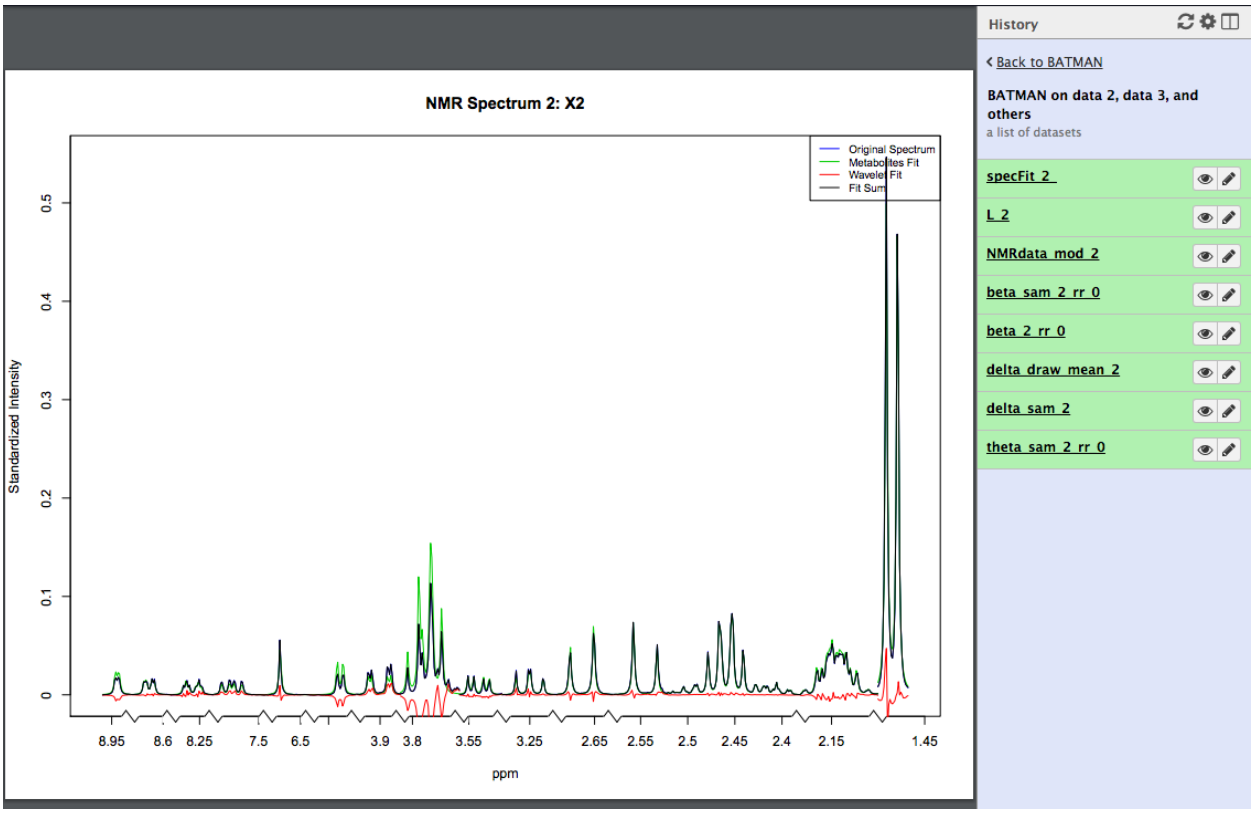

Figure 20: An example output of BATMAN (NMR spectra fitting) produced at the EMBASSY Cloud Galaxy installation, running BATMAN and Galaxy through containers.

BATMAN was successfully tested on two local Kubernetes installations (Windows and Mac) and in a cloud installation (EMBASSY Cloud), with Galaxy and BATMAN running as separate containers. Small sample datasets taken from a designed mixture trial data were used.

Figure 20 shows the example output plot “ppm” against “standardized intensity” of the listed/targeted metabolites from one spectrum. Three out of the four lines in the chart are of our interest. The blue line plots the original spectrum. The green line is the resulting metabolite fitting. The red line, the fit of the wavelet component of the model, can be thought of as indicating the residuals of the metabolite fit in general, the closer to zero the wavelet intensity, the better the metabolite fit. This demonstrates successful implementation of the BATMAN NMR tool within the Phenomenal architecture.

3.6. Documentation

An important part is to document the architecture, development, and usage of the VRE components. The PhenoMeNal wiki http://phenomenal-h2020.eu/home/wiki/ is the central location for technical documentation for users and developers. This is a read-only mode mirror to the source of the documentation on GitHub at https://github.com/phnmnl/phenomenal-h2020/wiki, where core developers and project members write the documentation articles.

We have generated detailed tutorials on how to execute the PhenoMeNal Galaxy VRE on top of a Kubernetes installation, both for users of PhenoMeNal Galaxy Tools and for PhenoMeNal tool developers, available at https://github.com/phnmnl/phenomenal-h2020/wiki/galaxy-with-k8s. This guide has been used to deploy Galaxy based VRE's on top of Kubernetes on the EBI EMBASSY Cloud, on EBI local machines (Mac), on Imperial College London (ICL) local machines (Windows), on CEA-Paris local machines (Mac) and on local machines at the University of Barcelona (Linux). We also have written guides on how to deploy Mesos-based VRE to OpenStack and public IAAS providers, and verified these on SNIC Science Cloud, EBI EMBASSY Cloud, and Google Cloud.

Interaction with external parties

For the design and implementation of PhenoMeNal e-infrastructure, WP5 has interacted with several organizations and projects. One important collaboration has been with the **MANTL project** for microservice architectures, which is funded by Cisco Cloud. We have been visited by MANTL lead developer (Steven Borrelli, Asteris, USA) who contributed to a workshop on microservice architectures in Uppsala on 2016-05-12 (see https://goo.gl/E2FqBS for the agenda). In the same event, project coordinator of EU H2020 **INDIGO Datacloud** participated. It has been very useful for the PhenoMeNal e-infrastructure design and project planning to interact with these two projects, and we have e.g. contributed back with code to the MANTL project. During another workshop in Uppsala, representatives of **EGI** participated (Enol Fernandez), and the result from this was (apart from informing developers about the offerings of EGI federated cloud) that Enol was able to bring back to EGI a list of use cases that PhenoMeNal would like to see in EGI. There is also large interaction with **Elixir Compute Platform (ECP)** in

particular in the area of Single Sign-On (Authentication, Authorisation and Profile Service) for interfacing with Elixir, EGI and for the development of common core components of the VRE Portal in WP6, and discussions on eInfra design has affected WP5 and D5.2.

We have also interacted with the H2020 projects **ToxBank** (www.toxbank.org) and **eNanoMapper** (http://www.enanomapper.net/), where the discussions on e-infrastructure based on PhenoMeNal experiences has lead to the recently granted H2020 EINFRA project **OpenRiskNet** (currently in grant agreement phase) for predictive toxicology using microservices. There will be much synergy between PhenoMeNal and OpenRiskNet during 2017 and 2018.

We also worked with the Galaxy project to meet requirements in the PhenoMeNal infrastructure. In particular, we were able to contribute the Kubernetes job runner, which is required to launch the microservice jobs in the containerized infrastructure. By submitting 7 pull requests (see https://github.com/galaxyproject/galaxy/pulls?q=author%3Apcm32), and thus upstreaming our changes, we reduce the burden of maintaining out-of-tree patches. We are also contributing towards the ansible-based provisioning of the Galaxy docker image, towards using parts of that for the second version of our Galaxy image, which is vital to move to better Galaxy provisioning (HTML cache, postgresql, etc) while at the same time keeping the docker image size low. This strategy also proved valuable to embedded system vendors contributing to the Linux kernel. Blörn Grüning (Galaxy core developer, Uni Freiburg, Germany) participated in these discussion. We are currently evaluating the “visual galaxy tours” as an instrument for the user training on Galaxy for Metabolomics in PhenoMeNal. PhenoMeNal developers are invited to the next Galaxy developer workshop 20./21.10.2016 in Freiburg/Germany.

During the PhenoMeNal annual consortium meeting there were discussions with Barend Mons of the European Open Science Cloud how PhenoMeNal fits in as a community demonstrator to drive innovation and demonstrate the utility of our e-infrastructure approach in metabolomics. PhenoMeNal representatives are now scheduled to attend future EOSC meetings, the next being in Krakow in September 2016.

3.7. Discussion

WP5 has progressed very well, but it has been challenging to work with state-of-the-art computational infrastructures in a rapidly moving software-defined contextualization and orchestration landscape. Some specific problems we have encountered so far are:

- Docker images can become very heavy relatively quickly, for which we have been slowly documenting and enforcing more good practices to avoid this. Careless image generation can easily make an image double the size (300 MB to 700 MB or more). Choosing the upstream image wisely, avoid distribution updates, use the same upstream image for as many containers as possible, and multi command runs executions which delete all that is not necessary for the following steps are some key strategies that allow to reduce file size of docker images. These best-practices have been documented in the PhenoMeNal wiki.
- High image size and poor default docker cleanup mechanisms quickly clogs infrastructure: docker slaves, machines in-charge of building docker images for the CI and more lately our docker registry. This has been more of an issue lately that we have so many people writing containers for so many tools. We can scale easily with more slaves and setting up scheduled clean ups of the slaves; docker registry needs scaling in terms of storage, which means that we will soon need to move this to a GlusterFS-backed storage or an object store directly backed by the EBI OpenStack installation (swift object store).
- Large docker images also makes deployment to the container orchestrator slower; again, another reason for improving the quality of our docker images.
- Integration of Galaxy to play nicely with tools orchestrated by Kubernetes imposes a minimal design constraints to containers:
    - Main scripts or programs to be invoked should be added to the PATH of the container and made executable. Execution shouldn't rely on the working directory of the docker container, but be independent of it.
    - Although still untested, containers should be able to allow any user to execute their main services, and not only root. In the production case, both Galaxy and the tools should run with a non-root user, but the same user. This needs to be thoroughly tested.
        - Solution for this is to agree on a common user id UID and make sure, as parts of the container test, that that user can run

successfully all the executables that a container wishes to expose as services.
    - These best-practices have been documented in the PhenoMeNal wiki.
- License issues of DLLs required for RAW formats conversions. Converting to open formats normally requires usage of proprietary libraries written by the vendors that have restrictive licenseses, which makes it problematic for open redistribution in the form of docker containers.
    - Solution would be to host these particular containers in a closed private docker registry (password protected), and then add the adequate secret to the container orchestrator cluster to be able to pull these images. However, the user would still need to say that they agree to the licenses (see ProteoWizard download page). This could be potentially done when the user deploys its VRE, so that the user says that it accepts all those needed licenses.

3.8. Risk assessment

A project at the forefront of e-infrastructure development is naturally associated with risks. Below we list a set of identified risks and how we plan to mitigate them.

Risk	Likelihood (1-3, 1 lowest)	Impact (1-3, 1 lowest)	Comment/plan
Cloud computing and microservices architectures will not be the next technology for building VREs	1	3	EU position papers[6] and roadmaps, the Elixir-Excelerate Compute Platform[7][8], and European Open Science Cloud[9] states that this type of architecture/technology is of highest interest for constructing VREs.

[6] E-infrastructures: making Europe the best place for research and innovation https://ec.europa.eu/futurium/en/content/e-infrastructures-making-europe-best-place-research-and-innovation

[7] ELIXIR Scientific Programme 2014-2018 https://www.elixir-europe.org/sites/default/files/documents/elixir_scientific_programme_final.pdf

[8] ELIXIR recommendations for a European Open Science Cloud https://ec.europa.eu/research/openscience/pdf/eosc-workshop-11-2015/elixir_24_november_2015.pdf

[9] European Open Science Cloud for Research, Position Paper: https://documents.egi.eu/public/RetrieveFile?docid=2637&version=1&filename=OSC_Position_Paper.pdf

Development and maintenance of VREs using the PhenoMeNal approach will be complicated.	1	2	There is very a large momentum in the entire IT-industry to move towards flexible virtual infrastructures, and frameworks and tools are maturing constantly, improving usage and development of VREs. Staff exchange meetings also act as training for used technologies.
PhenoMeNal eInfrastructure will be incompatible with other European initiatives (EGI, ECP, IINDIGO DataCloud etc)	1	2	Maintain tight communication with external parties to prevent diverging developments. PhenoMeNal is working closely with Elixir on core components for EGI integration, this collaboration will ensure common goals are meet
Tools in PhenoMeNal will be incompatible with each other	1	3	We will establish vigorous continuous integration tests and only publish tools “certified by PhenoMeNal” that pass them.
PhenoMeNal eInfra will be difficult and expensive to maintain	2	2	Building on open standards and open source, the tools in PhenoMeNal will be developed individually by groups. The minimal core of PhenoMeNal will be sustained as a focal point providing a continuous integration system to test/assess the tools, and with means to spawn VREs to execute them on IaaS.

3.9. Future roadmap

In the coming months we expect to increase substantially the amount of tools wrapped for Galaxy usage and tested within Kubernetes installations (both local and cloud based), as there are already 11 tools ready as docker containers that haven’t been yet incorporated into the workflow environment by the consortium developers. Having a tool containerised is the most significant pre-requisite to be able to run it on our micro-

services infrastructure. Following to that, writing a tool wrapper for Galaxy and testing it in the Galaxy PhenoMeNal runtime on top of Kubernetes is the next step. More containers will come as well in the following months for the different Metabolomics use cases (4) that we are working on.

The need to host provisionally containers that cannot be redistributed prior to licenses agreements by the user forces us to provide very soon a password protected private docker registry in addition to our already publicly available for download private docker registry.

We are in discussions as well to find ways of automating testing not only of our docker building process, as it is already to our Jenkins CI, but also testing that the containers can execute the task expected (Unit tests) and that they can work together (Integration tests). A workshop dedicated for testing of containers, workflows and virtual infrastructures is planned during fall 2016.

The Galaxy Kubernetes Runner has a number of improvements to be done: use PostgresSQL instead of SQLite, use a caching framework as uWSGI, provision Galaxy tools directly from the git repositories hosting those tool wrappers, use defined Kubernetes namespaces and improve security among other features.

The next challenge will be federation of data and computing in PhenoMeNal. The anticipated plan is to investigate data federation using iRODS, distributed container orchestration in Kubernetes, the new Pachyderm system for containerized analytics, and the use of Apache Spark for multi-data center analysis.

4. WORK PLAN

WP5 is dedicated to “Service activities” within the PhenoMeNal Work plan.

4.1. Structure

The details about the objectives and the description of work as broken down into tasks in WP5 is described in detail in the document of work and Grant Agreement (GA) of the project. Below is the summary of the tasks including subtasks related towards this deliverable:

T5.1: Operation and Maintenance of the GRID/cloud infrastructure

From grant application: "In T5.1 all partners will contribute to the operation and upgrade of the middleware and the other necessary software tools that are needed to keep the GRID/cloud infrastructure and related functionalities fully available to users. Within this task, the partners will also perform hardware maintenance and upgrades as needed to sustain the services, on the basis of both the evolution of the technological requirements and the level of demand from the users. All partners will be responsible for the correct operation of the various elements deployed at their sites. In addition, UU will provide advice and indications to all the partners involved with respect to the technological solutions to be adopted at any specific point in time, both as a consequence of the availability of innovative solutions or due to specific requests from users or other stakeholders. UU will additionally be in charge of coordinating the deployment at the various sites, also by collaborating in testing possible alternative solutions."

This task was broken down in the following subtasks:

- Set up proxy server on EMBASSY Cloud
- Deploy jenkins appliance on openstack embassy cloud
- Automate proxy and wordpress appliances deployment on embassy cloud
- Deploy kubernetes (core-os cloud-init based) on EMBASSY Cloud tenancy
- Deploy kubernetes (MANTL based) on EMBASSY Cloud tenancy
- Set up glusterfs for MANTL based Kubernetes on EMBASSY Cloud
- Improve access to EMBASSY and k8s for collabs
- Update Jenkins to latest version and setup automatic updates
- Update jenkins slave docker daemon, delete intermediate images and automate this

T5.2: Operation and Maintenance of the PhenoMeNal VRC

From grant application: "In T5.2 the PhenoMeNal VRC (gateway/portal) will be installed, maintained operational and continuously improved as new relevant technologies and tools will become available. All partners will be responsible to contribute to the task, and tools should preferably be built and deployed in the Continuous Integration platform

(Jenkins) in T5.5. Partners UU, EMBL-EBI and IPB will coordinate the deployment of solutions at the various sites. This task will also include the installation and configuration of new applications and web portals to the gateway."

We have planned to implement this so that the PhenoMeNal VRE portal will be established (in WP6) to be able to launch VRE on a selected IaaS provider where the user has credentials. D5.2 concerns the proof-of-concept of the VRE without the portal, and has been broken down into the following subtasks:

- Investigation of StackStorm
- Explore cloud provisioning software
- Investigation of MesoSphere
- Background research on microservices frameworks
- Work on first prototype integration between galaxy and kubernetes
- Using Google Cloud with MANTL
- Plan and prepare k8s / galaxy uppsala practical
- Planning for workshop in Uppsala
- Tutorial for MANTL and MesoSphere
- Using OpenStack with MANTL
- Blog about e-infrastructures workshop
- First version of Wireframe / low fidelity mockups for VRE design
- Prioritised list of top VRE features per persona
- Investigate Jupyter and NextFlow
- Speeding up VI initialization and scaling
- Technical overall ethical requirements for phenomenal
- Documentation: Security in PhenoMeNal
- Container building streamlining
- Requirements for setting up VRE on local infrastructure
- Technical Proof-of-Concept manuscript
- Documentation: Deployment guide
- Documentation: Development pipeline
- Documentation of PH e-infrastructure architecture

T5.3: Provisioning of the PhenoMeNal Services

From grant application: "T5.3 covers the installation, operation and maintenance of services (e.g. as deployed publicly accessible VMIs) and other types of services to be deployed on the project grid/cloud. Each partner will act in identifying the needs for

services and will be responsible for formalizing them. Partner EMBL-EBI will coordinate the response of the project team by pooling together the various requests and identifying possible synergies. Partner UU will collaborate with the partner that identified the need, also in concert with the relevant stakeholders, to devise the best corresponding solution."

- Deploy Workflow4Metabolomics on openstack
- Sort out Jenkins privileges issues for building docker images/vagrant images
- Set up core-os based docker registry
- Second try at securing docker registry
- Test mass IPO with real data on kubernetes
- Debug IPO memory issue
- Microservices with Docker and R
- Microservices with Docker and OpenMS
- ISA converters microservices

4.2. Coordination and management of the activities

The WP5 is led by Uppsala University who is responsible for the coordination of the planning of work and related deliverables. The tasks for this deliverable were distributed between EMBL-EBI (WP1 and WP6 lead), IPB (WP9 lead) and UU (WP5 lead) and were monitored using dedicated Google hangouts. The progress was tracked using Pivotal Tracker- an Agile project management tool (see Figure 21). A complete list of the subtasks in WP5 is provided (see Table 1)

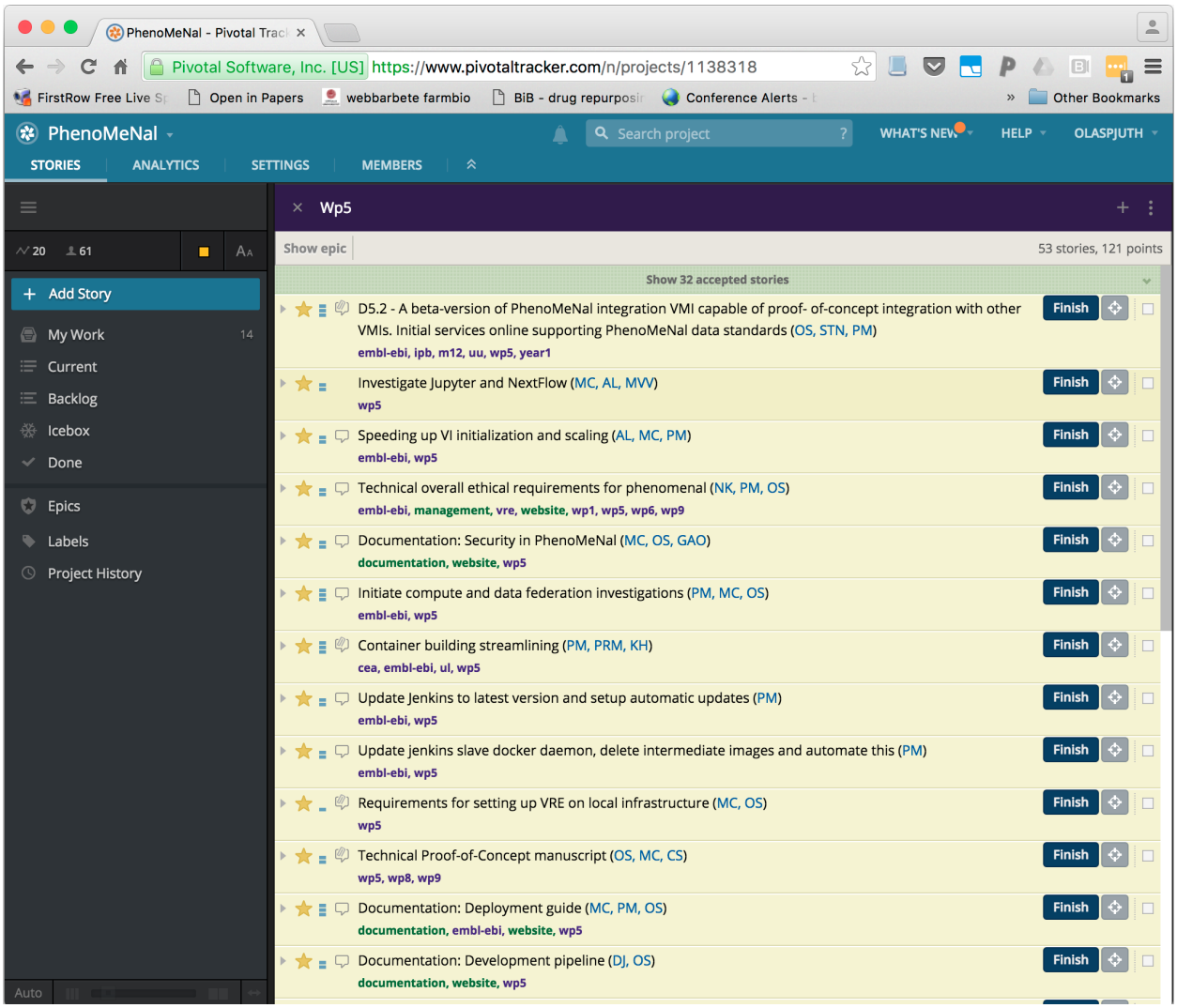

Figure 21: Screenshot from Pivotal Tracker- an Agile project management tool which is used in PhenoMeNal to manage tasks.

Title	Description
Investigation of StackStorm	
Deploy worflow4metabolomics on openstack	Deployment of workflow4metabolomics in openstack wasn't straightforward because it required to merge the multiple disk images that the VM fat image included. Took a few days. It is available when attaching the floating IP, but still routing is needed to avoid using the floating IP with this appliance only.
Set up proxy server on EMBASSY Cloud	
Deploy jenkins appliance on openstack embassy cloud	Set up a Jenkins appliance based on Ubuntu server 14.04 and docker container for jenkins. It should use an external non-ephemeral volume to keep jenkins user files. Use the openstack API and cloud-init to setup everything through shell scripts.
Automate proxy and wordpress appliances deployment on embassy cloud	Automate the deployment of both wordpress and proxy appliances through openstack API+cloud-init+shell. Wordpress appliance required a number of modifications to deal with the routing to a different URL and to use the phenomenal domain. This also includes wordpress using a non-ephemeral volume for posts (database) and for backups.
Explore cloud provisioning software	I was asked at the latest WP9+WP5 hangout to produce a document/diagram describing how multiple technologies interact: packer, vagrant, ansible, docker, kubernetes, cloud-init, slim OSs, etc.
Sort out Jenkins privileges issues for building docker images/vagrant images	Jenkins needs to build docker images, however this requires root access on the machine and doing docker within docker, since jenkins is running as a docker image. Root access means that anyone gaining access to jenkins could inject malicious code in a jenkins build and bring down the system.
Set up core-os based docker registry	This is where our CI pushes images, to be used later by Kubernetes to run jobs. Initially I tried to set up a secure registry using self signed certificates, however this failed, so in the end, initially, I set up an insecure docker registry.
Deploy kubernetes (core-os cloud-init based) on EMBASSY Cloud tenancy	Set up Kubernetes on the EMBASSY Cloud openstack tenancy, based on the core-os installation.
Investigation of MesoSphere	
Use cases at UU	
Deploy kubernetes (MANTL based) on EMBASSY Cloud tenancy	Set up Kubernetes on the EMBASSY Cloud openstack tenancy, based on the MANTL deployment.
Set up glusterfs for MANTL based Kubernetes on EMBASSY Cloud	We require a shared file system accesible from Kubernetes and the workflow environment.

Second try at securing docker registry	Having a secure docker registry allows any docker daemon to use it. This is relevant to be able to use our registry from different kubernetes instances without having to fiddle with the docker daemons in those nodes.
Background research on microservices frameworks	
Test mass IPO with real data on kubernetes	Run a real execution of mass ipo with metabolights data on the deployed kubernetes cluster. Memory problems found.
Work on first prototype integration between galaxy and kubernetes	We require a first prototype to understand how to send jobs from galaxy to kubernetes, its limitations, understand the galaxy approach to scheduling, etc.
Debug IPO memory issue	Running IPO on real data sets shows a big intake of memory.
Improve access to EMBASSY and k8s for collabs	Add open VPN and ssh access for partners.
Using Google Cloud with MANTL	
Plan and prepare k8s / galaxy uppsala practical	Tutorials/practicals/demos for the Uppsala workshop
Planning for workshop in Uppsala	
Tutorial for MANTL and MesoSphere	
Using OpenStack with MANTL	
Microservices with Docker and R	
Microservices with Docker and OpenMS	
Blog about e-infrastructures workshop	
ISA converters microservices	
First version of Wireframe / low fidelity mockups for VRE design	Focus on top 5 features for personas "bioinformatician" and "Clinician"
Prioritised list of top VRE features per persona	Identify a top list of features per persona. Please see included Google Doc. This will be the starting point for the initial mockups
Investigate Jupyter and NextFlow	
Speeding up VI initialization and scaling	Speeding up VI initialization and scaling (Ansible, Packer, local mirrors of repos)

Technical overall ethical requirements for phenomenal	What overall (30K feet overview) technical considerations/configurations in terms of security and ethics. Passwords, encryption, how to run pipelines safely. etc
Documentation: Security in PhenoMeNal	Internal verbose document, external short summary for website
Initiate compute and data federation investigations	Task to kick off this extensive sub-project.
Container building streamlining	Container Testing, statistics, keeping track of what we have - generate a web page etc. Interacts with WP6: VRE â□œAppStoreâ□□.
Update Jenkins to latest version and setup automatic updates	
Update jenkins slave docker daemon, delete intermediate images and automate this	Jenkins slave needed an update of docker to 1.11
Requirements for setting up VRE on local infrastructure	Create a document describing the requirements. Could be used to send to IT-providers at university/national organizations.
Technical Proof-of-Concept manuscript	Writing up of a Proof-of-Concept scientific manuscript.
Documentation: Deployment guide	Deployment guide, how-to (public cloud, openstack, server/VM). Will likely be a live document to be updated during PhenoMeNal lifetime. Also short summary for website.
Documentation: Development pipeline	Should produce internal documentation on development in PhenoMenal, and also graphical external material to be placed on website to explain for the world our strategy
Documentation of PH e-infrastructure architecture	Generate internal in-depth documentation and external one-page summary for website

Table 1. Pivotal tasks in WP5 that has lead up to deliverable D5.2.

Utilization of resources:

The total PMs (person months) utilised until M12 (inclusive)

Partners	**EMBL-EBI**	**ICL**	**UB**	**IPB**	**CIRMMP**	**UU**	**CEA**
PMs	5	2	1	9	1	12	4

5. DELIVER AND SCHEDULE

The deliverable was submitted on time.

6. CONCLUSION

With this deliverable, PhenoMeNal has established the VRE in proof-of-concept stage, with the integration of a number of services, demonstrated through several demonstrators. WP5 and project progress is hence on schedule.