Deliverable 9.2.3

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	9
Work Package Title	WP9 Tools, Workflows, Audit and Data Management
Deliverable Title	D9.2.3 *Services* Virtual Machine Image to facilitate the PhenoMeNal toolsets and pipelines, both locally and in the grid.
Delivery Date	M16
Work Package leader	IPB
Contributing Partners	IPB, ICL, UU, EMBL-EBI, UL, CRS4, UOXF
Authors	Steffen Neumann, Noureddin Sadawi, Jianliang Gao, Daniel Schober, Christoph Ruttkies, Kristian Peters, Philippe Rocca-Serra, David Johnson, Alejandra Gonzalez-Beltran, Reza Salek, Michael van Vliet, Luca Pireddu, Ola Spjuth, Stephanie Herman, Pablo Moreno.
Abstract Description of the architecture of the service VMIs with user-facing web interfaces	

and how service VMIs performing the container orchestration are installed in the Virtual Research Environment (VRE). Stable releases are planned bi-annually, the first one in 2017.02. The chosen cloud and software architectures are well-aligned with emerging bioinformatics efforts for use in national and European scientific cloud infrastructures.

CONTENTS

1. EXECUTIVE SUMMARY

We here report on the harmonised ecosystem of interacting services necessary to facilitate core PhenoMeNal toolsets and pipelines, both locally and in the cloud. In this report we describe the architecture of the service Virtual Machine Images (VMIs) with the user-facing web interfaces, where we offer Galaxy as a visual workflow engine and Jupyter for interactive explorative analysis in R and Python. We also describe how the service VMIs performing the container orchestration are installed in the Virtual Research Environment (VRE) in an automated fashion, shielding the users and local administrators from the complexity of building their own cloud infrastructure.

The VMIs described here build on the work from deliverables, 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), D9.2.1 (*PhenomeNal-Preprocess* Virtual Machine Image to enable data producers to locally process raw data into standards formats supported in PhenoMeNal) and D9.2.2 (*PhenoMeNal-Data* Virtual Machine Image to enable sharing and dissemination of standardised and processed omics data to participating online repositories, like MetaboLights), which produced the Continuous Integration system, Virtual Machines Images (VMIs) for compute infrastructure, data standardisation and storage repository upload, respectively. Furthermore, we have created a VRE release plan scheduling the public releases of the PhenoMeNal infrastructure with dependencies for the first two use cases data processing workflows in February 2017. In conjunction with the compute VMIs due in D9.2.4, we have all the required components and scheduled the release process to launch the first release of PhenoMeNal Alanine in 2017.02.

2. WORK TOWARDS PROJECT OBJECTIVES

A summary of work towards the project objectives:

Objective 9.1: “Specify and integrate software pipelines and tools utilised in the PhenoMeNal e-Infrastructure into VMIs, adhering to data standards developed in WP8 and supporting the interoperability and federation middleware developed in WP5. Most tools will be already available (see table 1.1) and we will develop new applications to complete ‘missing links’ in pipelines. Although two explicit releases for VMIs are listed

as deliverables below, we will use public repositories and continuous integration to always provide development snapshots of the infrastructure VMIs."

- We have created cloud-ready VMIs for the Galaxy and Jupyter user facing web applications.
- All the VMIs are available on the PhenoMeNal public container registry.
- We have created a release plan with four defined stable releases during the project runtime that include the tools that have passed a multi-layer testing procedure.

Objective 9.2: "Develop methods to scale-up software pipelines for high-throughput analysis, supporting distributed execution on e.g. local clusters, private clouds, federated clouds, or GRIDs."

- We have created cloud-ready VMIs for the Galaxy and Jupyter user facing web applications.
- All the VMIs are available on the PhenoMeNal public container registry.
- All of the software packaged in containers has been tested to run on scalable infrastructure on the Google Cloud Platform (GCP), Amazon Web Services (AWS) and three different OpenStack installations (EMBL-EBI Embassy cloud, the German scientific de.NBI cloud[1] and the commercial Swedish operator CityCloud)

3. DETAILED REPORT ON THE DELIVERABLE

In this document we report on the delivery of the Services Virtual Machine Images (VMIs) to facilitate the PhenoMeNal toolsets and pipelines, as shown in the overview in Figure 1. In particular, we describe:

1. The service VMIs with the user-facing web interfaces, where we offer Galaxy as a workflow engine and Jupyter for interactive explorative analysis in R and Python;
2. The service VMIs performing the container orchestration, which are installed in the Virtual Research Environment (VRE);
3. The containerization conventions and testing of the tools required in the use case workflows;
4. The release process for the VRE.

[1] http://www.denbi.de/localmedia/documents/quarterly_newsletter_05.pdf

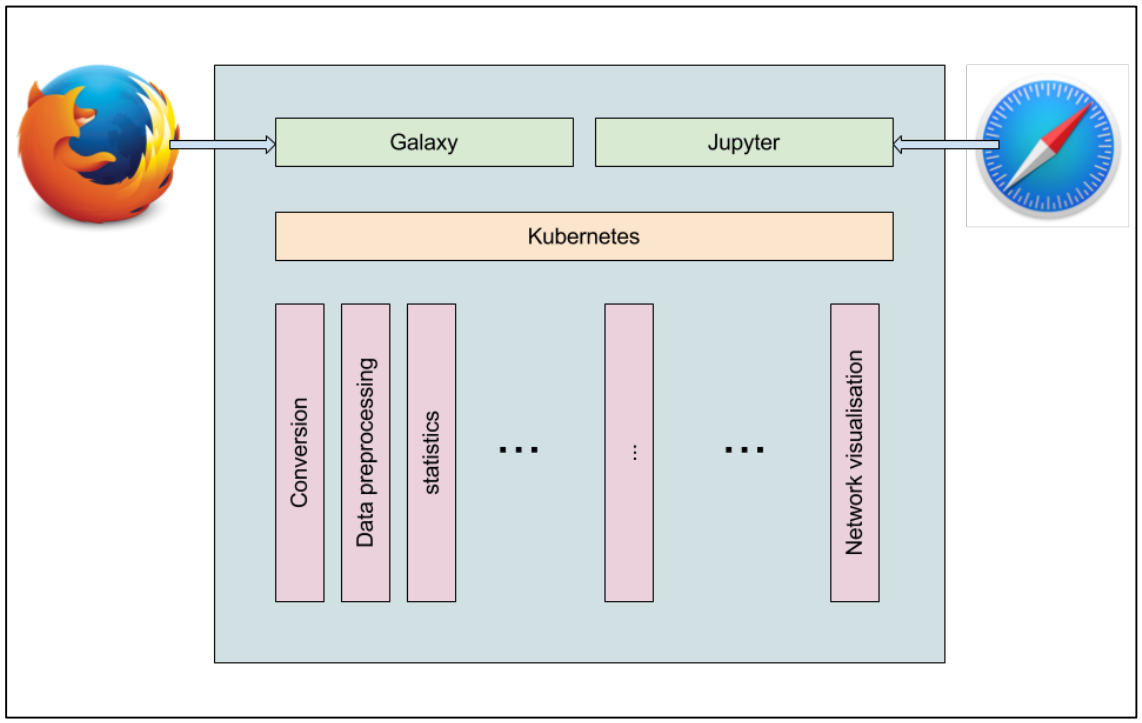

Figure 1: Architectural overview. Shown in green are the user-facing service VMIs with the web interfaces to design and run workflows (Galaxy) as well as programmatic access to specific research questions (Jupyter). As container orchestration service we use kubernetes, which is shown in orange. This central component is being used to control the life cycle of both the service VMIs and the compute VMIs shown in pink (the latter will be described in detail in D9.2.4).

3.1. Service VMIs with User web interfaces

A central component of the PhenoMeNal platform are the web interface VMIs that are deployed in cloud environments by bioinformaticians and local cloud administrators for the actual end-users, e.g., clinicians, biochemists. In PhenoMeNal we chose **Galaxy** to be the web interface VMI for visual workflow development and **Jupyter**[2] to be the web interface VMI for programmatic access to the containerized tools (see more details in the following sections). Both Galaxy and Jupyter are long-living service VMIs and are distinct from the short-living compute service VMIs – e.g., defined data analysis tools running particular jobs. By contrast, the latter are only destined to run until they have finished the assigned computational task.

[2] "Jupyter." http://jupyter.org/. Accessed 16 Dec. 2016.

Galaxy for visual workflow development

In PhenoMeNal we chose the Galaxy environment because it allows users to create and manipulate workflows in a very flexible way. Galaxy is a web application that provides a graphical user interface which can be easily used and customised by the end user. Galaxy is a well established Bioinformatics Workflow environment with a rich community of users and developers supporting it. According to the recent survey carried out in the field of Metabolomics and reported in Deliverable 8.1, it was by far the most well known workflow environment among practitioners in the field.

Within PhenoMeNal, we deploy a Galaxy instance in the cloud environment of each VRE. To achieve that, it is vital to design and create the appropriate infrastructure. The underlying technical infrastructure is described in the section "Service VMIs performing the container orchestration", where we describe how we facilitate running and accessing the compute VMIs (which represent the individual tools and components of the workflows, due in D9.2.4).

Jupyter for programmatic access to containerised tools

Jupyter Notebook[3] is a web application that allows users to create and share documents – called *notebooks* – that contain live code, equations, visualizations and explanatory text. Compared to Galaxy, Jupyter is a more flexible tool that provides a mixed command line and graphical interface, allowing advanced users to programmatically leverage the platform. For instance, it is simple with Jupyter to generate interactive results and publication-ready graphics in downstream analyses.

The scripting/coding nature of Jupyter allows users to run workflows of containerized tools programmatically, through the Kubernetes[4] REST API. To demonstrate its potential within PhenoMeNal, we have created an OpenMS[5] preprocessing workflow using the PhenoMeNal OpenMS container combined with an R-based downstream analysis.

In the downstream analysis notebook, the most important feature is the interactivity, enabling parameters to be changed and code to be rerun and directly evaluated using visualizations and direct output. Default analysis workflows can be shared with colleagues and collaborators and further customized for individual studies, to promote

[3] "Jupyter." http://jupyter.org/. Accessed 16 Dec. 2016.
[4] "Kubernetes - Production-Grade Container Orchestration" http://kubernetes.io/. Accessed 20 Dec 2016
[5] "OpenMS." 17 Jun. 2016, http://www.openms.de/. Accessed 16 Dec. 2016.

and increase consistency between studies and decrease time spent on rewriting processing workflows.

To emphasize the convenience that Jupyter Notebook provides, we are currently working on a reproduction of the data processing and analysis done by C. Ranninger *et al.*[6] The preprocessing is done with the containerised OpenMS, using the same parameters that were used in the actual study. The whole workflow, preprocessing and downstream analysis will be deployed and controlled from the Jupyter interface. Furthermore, to demonstrate scalability, the same workflow will be run with a larger dataset, namely the one published by A. Ganna *et al.*[7] as shown in Figure 2.

[6] C. Ranninger *et al.* (2016). Improving global feature detectabilities through scan range splitting for untargeted metabolomics by high-performance liquid chromatography-Orbitrap mass spectrometry, *Analytica Chimica Acta*, 930, 13-22

[7] A. Ganna *et al.* (2014). Large-scale metabolomic profiling identifies novel biomarkers for incident coronary heart disease, *PLOS Genetics*, 10(12)

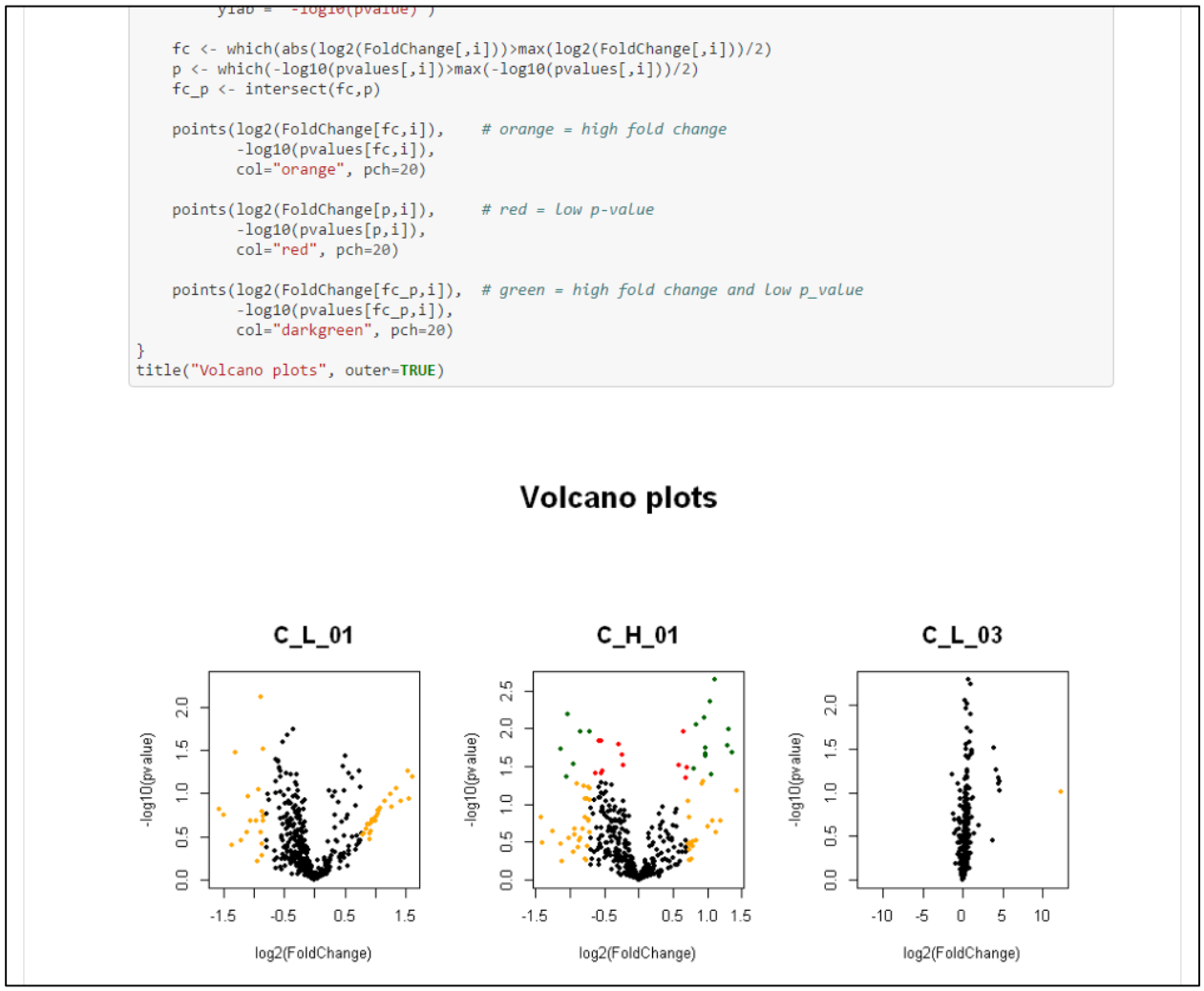

Figure 2. Screenshot from a Jupyter Notebook showing an excerpt of the source code for the creation of a volcano plot (above) and below the generated plots.

Database and backend service VMI

Several tools require accessing and performing complex searches on molecular databases like PubChem. For example, the Open Source MetFrag software performs these queries to annotate high-precision tandem mass spectra of metabolites, which is a first and critical step for the identification of a molecule's structure. Candidate molecules of different databases are fragmented in silico and matched against mass to charge values, and a score is calculated using the fragment peak matches. However, performing these database queries on public databases connected to the Internet is a major bottleneck for a tool like MetFrag due to load limits on the public service and data transfer speeds over the internet. Therefore, when they are required, making these data

resources a part of the PhenoMeNal cloud infrastructure is a key to reduce processing times.

We have created the “metfrag-cli”[8] VMI to integrate MetFrag in our platform (it is already available as Galaxy tool in our public Galaxy instance). To ensure it runs swiftly, we have implemented a service VMI facilitating the metfrag-cli VMI by creating local database mirrors of the PubChem[9] database, as shown in FIgure 3. PubChem contains structures and associated biological data on almost 93 million compounds. These VMIs consist of a database VMI running the PostgreSQL RDBMS, which has access to local storage providing the data resources, and a second backend VMI serving as a data importer that downloads all necessary datasets once and copies them via the database container to the distributed storage. Once the data import is finished, the metfrag-cli and other VMIs requiring database access are able to query data via the database VMI. The import VMI can then be used to keep the local database mirrors up to date by periodically fetching the updates from the online repositories. As the database storage is distributed within the PhenoMeNal cloud infrastructure, the same data can be queried from several instances of the database VMI.

[8] "GitHub - phnmnl/container-metfrag-cli: Command line interface of ...." 3 Nov. 2016, https://github.com/phnmnl/container-metfrag-cli. Accessed 16 Dec. 2016.
[9] "PubChem." https://pubchem.ncbi.nlm.nih.gov/. Accessed 16 Dec. 2016.

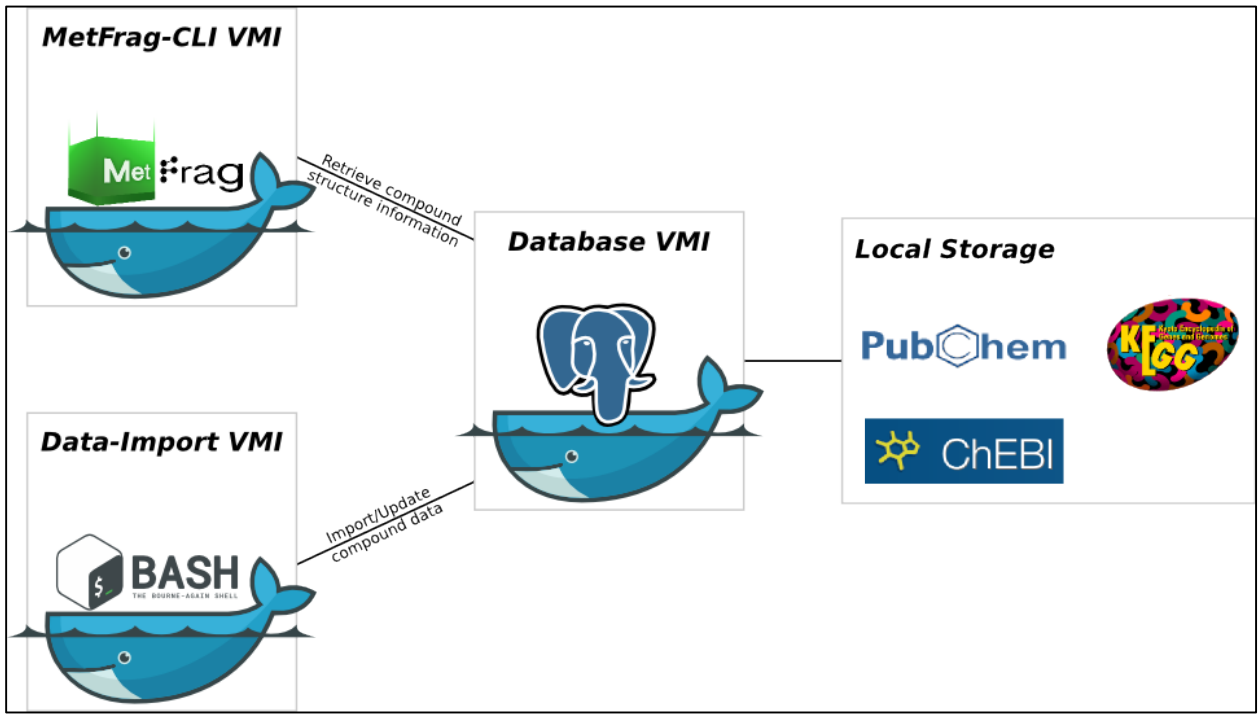

Figure 3: Architectural overview of database VMIs. The Database VMI is connected to a local storage located within the PhenoMeNal cloud infrastructure. Data are imported and updated via the Data-Import VMI connected to the Database VMI.

3.2. Service VMIs performing the container orchestration

The process of installing and configuring a complete PhenoMeNal infrastructure with all required resources (Galaxy and Jupyter on top of multiple Kubernetes nodes, network topology and storage volumes etc.) on a given cloud provider would be cumbersome if done manually. Instead, we are providing a completely automated installation procedure, which allows the end user to have the defined interface service VMIs running through a cloud provider only minutes after providing its cloud credentials on top of the scalable Kubernetes cluster.

Installation of the service VMIs in a cloud infrastructure

The first phase consists of instantiating the resources on the cloud providers. For this step, there are cloud-provider-specific automation engines like AWS CloudFormation or OpenStack Heat, but to avoid a vendor lock-in, we are using the Open Source project Terraform[10] to define the required infrastructure: VMs, networks, firewalls and storage volumes can easily be defined, leaving to Terraform the task to understand

[10] https://www.terraform.io/

dependencies between all these resources and the order in which they must be created. At the time of writing, Terraform supports all the major public cloud providers (AWS, Google Cloud, Azure, Rackspace, as well as OpenStack).

After a set of resources is created by Terraform, the second phase consists in the installation and configuration of the underlying components of the PhenoMeNal-specific software stack. Here, we are using the Open Source Ansible framework[11] to customise the operating system and install the container engine (Docker), container orchestrator (Kubernetes), required overlay network layer (currently flannel, others available) and shared file system provision (currently GlusterFS) for running a scalable PhenoMeNal VRE. The shared filesystem is created spanning over several VMs with the Open Source GlusterFS[12] network filesystem. This storage is then used by both the PhenoMeNal services – e.g. Galaxy – and the compute nodes, to exchange input and output data sets. This part is known as the "software" provisioning layer, and as mentioned, is executed through Ansible. At the end of this "software" provisioning process, the interface service VMIs (Galaxy and Jupyter) are started on top of Kubernetes for the user. We use the Helm[13] framework to install all Kubernetes managed applications. We have contributed Ansible roles and variable definitions[14] to the upstream project so that Ansible can install and communicate with a Helm daemon running on a provisioned Kubernetes cluster.

To start the interface service VMIs, Helm downloads and configures the containerised components – the upper layers of the PhenoMeNal specific software stack. Helm was chosen because it deploys the Service VMIs automatically and encapsulates their deployment, making this process independent of whatever was used to deploy the underlying Kubernetes cluster.

Helm uses uses Go templates to make parametrized deployments of complex Kubernetes API object constructs and uses YAML with placeholders for variables that can be set on deployment time. This strategy facilitates the deployment process. For instance, for the PhenoMeNal Galaxy VRE, two Kubernetes Pod objects (one for Galaxy, one for PostgreSQL), two Kubernetes Service objects (to access Galaxy and to access PostgreSQL from Galaxy), a Kubernetes Secret object (for storing encrypted database access), a Kubernetes Configuration Map object (which stores database settings) and Ingresses objects are necessary. Each of these contain a number of variables, from user names to ports or volumes (storage) preferences, that can vary

[11] https://www.ansible.com/
[12] https://www.gluster.org/
[13] https://github.com/kubernetes/helm
[14] https://github.com/kubernetes-incubator/kargo/issues/661

from a local deployment meant for testing compared to a production deployment at a cloud provider (and from cloud provider to cloud provider). Manual Kubernetes object deployment would require a number of steps to create all these objects, and changing YAML files for each different type of installation. The Helm based installation takes care of all of this through a single call, including variable setup for the different scenarios.

In order to run the specified two phases of the provisioning, we initially selected the deployment project MANTL, which is unfortunately not actively maintained anymore by its main contributor, Cisco Systems. Hence, we currently use the PhenoMeNal in-house developed KubeNow[15], using the Kubernetes community-developed KubeAdm[16] administration interface. KubeAdm's deployment enables the creation of immutable images and is considered to be very performant in large deployments (hundreds of nodes). KubeNow merges some ideas (such as edge nodes or the embedded usage of reverse proxies) from the previously used MANTL.

We are also investigating Kargo, which has support for complex cloud environments and High Availability, as it manages the whole complexity of a Kubernetes deployment. Both KubeNow and Kargo make use of the same tool sets for the provisioning (Terraform and Ansible), as described above. Kargo[17] is a community developed project, recently incorporated as an official Kubernetes project, and receives a lot of attention from the community and also companies like OpenStack, Mirantis and CoreOS.

Kargo was already used at the beginning of PhenoMeNal (in a non-automated way) to provision the existing PhenoMeNal public instances available at the EBI EMBASSY Cloud, but was missing important features required in the PhenoMeNal deployment use case. In particular, Kargo did initially not allow to provision clusters with a single public IP, nor did have built-in support for the required shared file system or the Helm package manager. These abilities were contributed by PhenoMeNal developers to the community maintained Kargo project.

Kargo deals with the complexity of setting up a production grade Kubernetes cluster, yet this complexity makes the production of an immutable image more complicated. A Kargo deployment of about 4 nodes, which should suffice for many PhenoMeNal use cases, takes around 15 minutes, and this cluster can later be expanded. It is expected that the newer Ansible 2.2 reduces this further. Kargo can also be deployed on top of bare metal running a variety of Linux distributions (Ubuntu, Debian, Fedora, CentOS,

[15] https://github.com/kubenow/KubeNow
[16] http://kubernetes.io/docs/admin/kubeadm/
[17] https://github.com/kubernetes-incubator/kargo

RHEL and CoreOS currently). This scenario is not supported in PheNoMeNal, but would be possible for institutions that might prefer in-house bare metal instead of on-premises or public cloud installations.

For the first PhenoMeNal release we will use KubeNow. For future releases we will need to consider the trade-off between requirements and complexity. Not depending on a single software solution that might be abandoned is important for meeting the sustainability goals of PhenoMeNal in the long term.

Within PhenoMeNal, such deployments have been demonstrated on the Google Cloud Platform (GCP), Amazon Web Services (AWS) and three different OpenStack installations (EMBL-EBI Embassy cloud, the German scientific de.NBI cloud[18] and the commercial Swedish operator CityCloud). PhenoMeNal is currently in contact with the Amazon Web Service team for sponsorship to regularly demonstrate and test the installation in their environment.

The installation configurations for the two phases are developed in the repository https://github.com/phnmnl/cloud-deploy-kubenow. The structure is such that the description can be readily used by the EBI Cloud Portal developed by the EMBL-EBI Technology and Science Integration (TSI) team, and thus the upcoming version of the PhenoMeNal Cloud Research Environment features a user-friendly portal-guided installation option.

Local OpenStack installations behind strict firewall configurations require that the installation be initiated from the inside. For this case, a scripted installation is available to invoke the two phases (Terraform, Ansible and Helm). A graphical interface equivalent to the EBI Cloud Portal for such walled-garden installations is planned for later in the project.

Installation of downscaled service VMIs on a local computer

For smaller compute requirements, the software stack can also be installed on a powerful workstation or even on a laptop. For developers, the upper layers of the PhenoMeNal-specific software stack (Galaxy, Jupyter and the containerised tools) can be installed using the Minikube environment, which configures all components optimizing them for local development. In PhenoMeNal we use Minikube to lift up all the required Kubernetes components for a working local installation for tool development purposes. The actual deployment of the individual PhenoMeNal VMIs is still performed

[18] http://www.denbi.de/localmedia/documents/quarterly_newsletter_05.pdf

with Helm[19], i.e. we use Helm to deploy the Service VMIs into both local or cloud-provisioned Kubernetes cluster.

The Helm package management thus encapsulates the deployment of our service VMIs and makes them independent of the process by which the required Kubernetes cluster is provisioned; so whether the end-user prefers Minikube, a PhenoMeNal-based provisioned Kubernetes cluster, turn-key Kubernetes on GCE or AWS, or any method of provisioning such a cluster, the PhenoMeNal service VMIs can be deployed without distinctions. A QuickStart guide for the Helm-based installation is available from the PhenoMeNal Wiki: https://github.com/phnmnl/phenomenal-h2020/wiki/QuickStart-Installation-for-Local-PhenoMeNal-Workflow.

3.3. Guidelines and conventions for testing and streamlining releases

The development and operation of an infrastructure like PhenoMeNal requires automated testing of all components and processes. A dedicated workshop at the EBI in November 2016 was organised to discuss and formulate standards and conventions for testing containers, as well as for streamlining current and future releases. As an outcome we created guidelines to ensure we meet these standards and conventions with the goal of guaranteeing the sustainability and longevity of containers and the whole technical infrastructure: (https://github.com/phnmnl/phenomenal-h2020/wiki/Dockerfile-Guide):

- Naming scheme: container-app-name, phnmnl/app-name
- Versioning scheme: app-name:software_version:build_version
- Defining best practices for continuous integration

We have created improved Dockerfile definitions and Jenkins job templates for building/testing/storing those new images, and we are updating current projects to meet the new standards. Implementing the new integration conventions requires a lot of effort and coordination between the different WPs.

During the workshop we defined testing guidelines at different levels[20]:

Tool Unit testing refers to unit tests for individual tools. Especially for tools developed within the PhenoMeNal consortium, we increase the test coverage. These are light-weight tests that make sure that binaries can run or are present within the created container. Failure at these testing level means that the container built is not pushed to our container registry, and hence only previously passing versions are available for usage until these test are satisfied at the current version.

[19] https://github.com/kubernetes/helm
[20] https://github.com/phnmnl/phenomenal-h2020/wiki/Container-testing-guides

Container testing: refers to the test of a tool in its runtime environment – the packaged container running within the context of a container orchestrator (Kubernetes) to process representative subsamples of real data sets. Within the PhenoMeNal infrastructure we are using containerization to manage software (tools) and their dependencies as complete packages. This packaging is of great help in achieving platform independence and reproducibility of analyses. However, to ensure their proper operation, these containers need to be tested in an scenario that it is as close as possibly to where they will be ultimately used: a VRE which runs on top of a Kubernetes cluster. We have implemented automated container testing for the tools we integrated in our platform, and we integrated this testing with Jenkins as part of our global project test management portal. The standard tests implemented for our containers verify the correctness of container definitions, as implemented by the developers, by rebuilding containers and running them on known inputs to verify successful execution and the correctness of the output. The tests are automatically run whenever new changes are committed to the container specification, and previous Tool Unit testing is passed.

Workflow testing: refers to testing complete Galaxy workflows in headless mode to verify their correct execution. Our guidelines define a testing model for workflow workflows where test cases a composed of input datasets, workflow parameters, and the corresponding expected output datasets; this model is analogous to typical component integration testing. We have developed a workflow test harness (https://github.com/phnmnl/wft4galaxy) that uses this model to automate workflow testing. Using our testing harness, the author of a workflow can define test cases in a simple configuration file (in YAML format) which references the prepared input and output dataset files and defines any relevant workflow parameters, see Figure 4 for an example. The workflow testing software takes care of all the details pertaining to the execution of the workflow (e.g., instantiating the workflow on the infrastructure, uploading and downloading test datasets, etc.) and verifying that the observed and expected results match. Following our accepted software containerization strategy, a Docker container with the testing harness has been created, allowing it to be used without any previous software installation. Moreover, our workflow testing harness is well integrated with the Jenkins testing platform used by PhenoMeNal. By combining these two tools, we are able to completely automate the testing of standard workflows that are maintained in our (and others') public repositories and go to form a PhenoMeNal workflow library.

```
### Global settings ###
enable_logger: True

### Workflow tests ###

workflows:
  # test case "multivariate"
  multivariate:
    file: "multivariate/workflow.ga"
    inputs:
      "DataMatrix": "multivariate/dataMatrix.tsv"
    expected:
      sampleMetadata_out: "multivariate/sampleMetadata_out"
      variableMetadata_out: "multivariate/variableMetadata_out"
```

Figure 4: Example of a workflow test definition. The file defines a test case for a workflow with one input dataset and two output datasets.

Infrastructure testing: refers to testing the deployment of the whole PhenoMeNal stack on a local workstation, local OpenStack installation, or public cloud provider.

This comprehensive testing architecture on all levels is novel in Bioinformatics and sets PhenoMeNal apart from other cloud projects. Although testing adds to the complexity of building VMIs, it allows us to promptly find software bugs that would otherwise remain hidden. Our testing scheme also facilitates the collaboration between the various Bioinformaticians and software / tool developers and meets the Sustainability goals as defined in the Release plan (see below).

3.4. Release plan and process for the PhenoMeNal Research Environment

A PhenoMeNal (stable) release comprises two sets of components for deploying and running a PhenoMeNal VRE:

1. The core PhenoMeNal cloud infrastructure contains sets of scripts that are responsible for installing PhenoMeNal core components on top of existing local, cloud or OpenStack environments and configured to orchestrate and deploy the PhenoMeNal VMIs in an automated way.
2. The PhenoMeNal VMIs which deliver services and workflows. Here, we subsume all the individual tools that are encapsulated in containers which can be deployed in the PhenoMeNal infrastructure.

The installation and provisioning was described in detail in the chapter “Service VMIs performing the container orchestration” above, in the following we describe the release process and tests for inclusion of the individual tools and VMIs.

PhenoMeNal VREs can be deployed from the either the development builds and as stable versions. The stable versions are available as bi-annual releases (Table 1), while development versions can be deployed any time from the latest builds from our continuous integration (CI) service.

Release Date	Codename	Version	Supported
February 2017	Alanine	2017.02	+1 year
August 2017	Betaine	2017.08	+1 year
February 2018	Cysteine	2018.02	+1 year
August 2018	Dopamine	2018.08	+1 year
February 2019	Epinephrine	2019.02	+1 year

Table 1: The current (stable) release plan. The maintenance of the 2018.08 version and release of 2019.02 will be handled under the sustainability plan.

Development builds, also referred to as “bleeding edge”, are for testing and getting access to the latest and greatest that PhenoMeNal has to offer, and we have started to reach out to the metabolomics developer community to also incorporate tools developed outside of PhenoMeNal. We encourage tool developers that want to integrate their tools into PhenoMeNal to use the development version and provide valuable feedback as input for the upcoming stable releases. While we keep all builds of the stable release, we only keep the last successful builds of the development branch. Thus, development builds are not intended to be used in production environments.

Stable builds are a selection of the tools that are considered production ready. These tools have passed all of our automated tests that are part of our testing framework. For a tool at a specified version to be included in a PhenoMeNal release, it needs to pass its Tool Unit tests, its Container tests and must not cause a failure for the Workflow tests in which it participates. That means they are tested properly, are maintainable, and have been well documented. An important aspect of a stable release is reproducibility. A user should be able to deploy an instance of a stable PhenoMeNal VRE on his/her preferred provider (or even locally for that matter), and be able to run individual tools or complete workflows resulting in identical results for the same input data. We accomplish this by storing all the release container images, properly version tagged, at each stable release in our container registry.

The release process includes an alpha release where we freeze the collection of containers to undergo integration and usability testing. 2-4 weeks after the alpha release we do a beta release where any bugs found in previous versions should be fixed. The last step, before the actual release, is the RC (release candidate) where we check if the supporting documentation is sufficient and make the final preparations for the actual release.

We target to produce stable releases two times per year. Not all tools in a previous version will be guaranteed to be present in a new release. Tools can be replaced or obsoleted in newer versions to ensure that the individual tools work together in a workflow to produce the expected results.

Integration of tools into the build and release process

In the beginning of the incorporation of a tool in PhenoMeNal, the developers create a container that encapsulates the dependencies of the tool and the tool itself. This container must be made available through the public PhenoMeNal github repository (https://github.com/phnmnl), where the convention is to start the repository name with 'container-' and use the git branch 'develop' for main development.

In this repository we also require from the tool developers to supply several files and configurations, that are used to different aspects in the build and deployment process:

- A README.md that explains what the tool does and how it is used[21]. The Information is also picked up and presented in the PhenoMeNal AppDB[22]
- A container configuration (e.g. Dockerfile) to build the tool and dependencies and package into a container[23].
- Testing scripts (e.g. testRun.sh and required data) that are needed to run tests on the tool and the container[24]
- A tool definition/configuration file (e.g. galaxy-tool.xml) to make the tool available in the Galaxy workflow engine[25]

All the details and practices that should be followed by the tool developers are present in our dedicated Developer Documentation as part of the PhenoMeNal Wiki[26]

---

[21] https://github.com/phnmnl/phenomenal-h2020/wiki/The-Guideline-for-Container-GitHub-Respository-README.md-Creation

[22] http://portal.phenomenal-h2020.eu/app-library

[23] https://github.com/phnmnl/phenomenal-h2020/wiki/Dockerfile-Guide

[24] https://github.com/phnmnl/phenomenal-h2020/wiki/Container-testing-guides

[25] For more details, refer Galaxy Manual on writing Tool wrappers: https://wiki.galaxyproject.org/Admin/Tools/AddToolTutorial

In order to add the tool to the PhenoMeNal build process, it is required to to add a build job for the container to the PhenoMeNal Jenkins (http://phenomenal-h2020.eu/jenkins/) providing the continuous integration and deployment service. Job creation is based on a Jenkins job template[27] with several mandatory steps, including proper versioning and testing of the container and embedded tool. All the steps required are documented at https://github.com/phnmnl/phenomenal-h2020/wiki/Jenkins-Guide. This is being done by one of the PhenoMeNal members who will also provides stewardship of the integration of the tool in the PhenoMeNal. Once integrated, any change in the github repository will then trigger an automated build and test procedure. When both build and test finish successfully, the container image is stored in the PhenoMeNal container registry, ready for deployment. If the procedure is not successful, the owner of the failing tool container will be notified and be requested to fix the failure.

If the container passes all tests, it is considered to be included in the next stable release and changes are merged with the 'main' branch in the corresponding GitHub repository of the tool.

4. WORK PLAN

Structure and management of WP9 tasks

As for the rest of the project, all WP9 activities has been managed and tracked by our Pivotal Tracker project[28] and dedicated online hangouts.

Program code and VRE installation are managed in public repositories under https://github.com/phnmnl, together with issue tracking and commit histories.

Utilization of resources

Person Month (PM) contribution towards this deliverable

Partner	EMBL-EBI	ICL	IPB	UB	UOXF	UU	CEA	INRA	UL	CRS4
PMs	4	3	2	0.5	2	4	2	1	1	1

[26] https://github.com/phnmnl/phenomenal-h2020/wiki/Developer-Documentation
[27] http://phenomenal-h2020.eu/jenkins/job/container-template-project/ (Configuration details of the template can only be seen and changed with an account).
[28] http://phenomenal-h2020.eu/home/about/project-management-tool/

Next steps

The next step will be the delivery of the “D9.2.4 - Compute Virtual Machine Image to enable standardised compute capabilities” and complete the first release “Alanine”, and reach out to a broader metabolomics and medical user and developer community.

Updated releases will cover a larger number of tools and thus workflows required for clinical data processing, covering all steps in data management from preprocessing, data processing, statistical analysis, data reduction, publication and storage of the ultimate scientific results generated. We also aim to standardise data exchange formats not only for the input and metadata, but also within the workflows.

5. DELIVERY AND SCHEDULE

The deliverable is submitted on time.

6. CONCLUSION

In this report we described the architecture of the service VMIs with the user-facing web interfaces, where we offer Galaxy as a workflow engine and Jupyter for interactive explorative analysis in R and Python, and how the service VMIs performing the container orchestration are installed in the Virtual Research Environment (VRE) in an automated fashion, shielding the users and local administrators from the complexity of building their own cloud infrastructure. Together with the compute VMIs due in D9.2.4, we have all the required components and laid out the release process to launch the first release of PhenoMeNal Alanine in 2017.02.

The chosen cloud and software architectures are well-aligned with emerging bioinformatics infrastructures in other disciplines. This will facilitate the adoption by both developers and users, and paves the way for the use of PhenoMeNal in national and european scientific cloud infrastructures.