Deliverable 9.2.5

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	D9.2.5 Portal Virtual Machine Image that is capable of integrating other PhenoMeNal-VMIs (in local federated clouds) and make all functionality available via command-line, Web-APIs and graphical user interfaces.
Delivery Date	M24
Work Package leader	IPB
Contributing Partners	IPB, EMBL-EBI, UU
Authors	Ken Haug, Sijin He, Pablo Moreno, Steffen Neumann, Daniel Schober

Abstract
The PhenoMeNal Portal contains the app-library, documentation and allows users to deploy the PhenoMeNal application stack to public or private OpenStack environments or the commercial cloud providers Google GCE and Amazon AWS. The production container images are publicly available on https://portal.phenomenal-h2020.eu/.

Contents

1 EXECUTIVE SUMMARY

The PhenoMeNal Portal contains the app-library and allows users to deploy the PhenoMeNal Workflow management infrastructure to public or private cloud providers. The detailed scope and required functionalities were determined in a previous design workshop and were already described in Deliverable 6.1. Initially, the portal was developed as a static website (implementation details described in Deliverable 6.2), and has since been developed further. We have now worked on packaging the PhenoMeNal Portal so that it can easily be installed on any Kubernetes cluster, including the easily installable Minikube[1]. This means that an administrator within a clinical institution can run the PhenoMeNal portal on top of a Minikube installation (which is simple to setup on a single machine).

In this deliverable, we describe the current operational status of the virtual research environment (VRE) portal and the set of infrastructure and GUI containers that constitute both the frontend as well as the backend engines.

2 CONTRIBUTION TOWARDS PROJECT OBJECTIVES

This deliverable has contributed to the following 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.

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

Objective 9.3 Add quality control and quality assurance to pipelines to ensure high quality and reliable data, keep an audit trail of intermediate steps and results.

[1] https://github.com/kubernetes/minikube

3 DETAILED REPORT OF THE DELIVERABLE

3.1 Overview and role of the portal

The PhenoMeNal Portal consists of the documentation (wiki), the app-library and the frontend UI application to provide the ability for users to trigger the installation of a PhenoMeNal VRE cloud deployment on the supported cloud providers.

The help pages at http://portal.phenomenal-h2020.eu/help are categorised into end user focused documentation and -tutorials, and technical documentation targeting developers and clinical IT managers. Documentation pages are created using markdown format on the project's GitHub wiki pages, publicly available at https://github.com/phnmnl/phenomenal-h2020/wiki, which simplifies content creation and community contributions via pull requests. The Portal help section mirrors this contents with regular periodicity.

The App Library showcases all the existing PhenoMeNal packaged tools that are available through a PhenoMeNal VRE deployment. The content is dynamically parsed from the README.md files created in the different PhenoMeNal containers on GitHub. The PhenoMeNal Portal App Library component consumes these periodically and produces the corresponding HTML output so that it can be viewed within the Portal.

The Portal (or CRE Portal) allows users to deploy the PhenoMeNal VRE (branded as PhenoMeNal Cloud Research Environment, CRE[2]) infrastructure to public or private cloud providers. After the initial authentication step through the Elixir AAI Single Sign-On (SSO) system, the user is presented with the ability to deploy to the two major commercial cloud providers (Amazon AWS and Google GCE) and to local (i.e. behind-clinical-firewall) OpenStack installations. The interface allows the user to secure the environment provisioned with username and password, in addition to manage and delete existing PhenoMeNal VRE deployments. Figure 1 shows the functionality diagrammatically for a user with AWS credentials.

[2] VRE and CRE are basically used synonymous, with the term 'VRE' appearing in the grant application and deliverables (for consistency) and the term 'CRE' appearing in the implemented portal infrastructure.

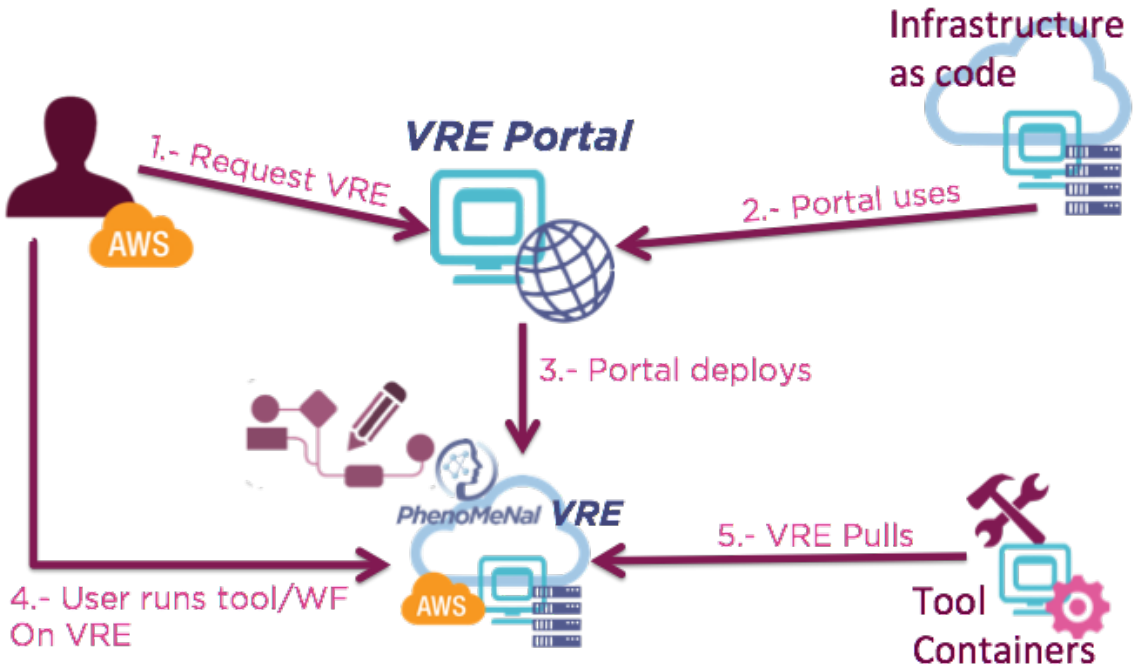

Figure 1: Involvement of the PhenoMeNal VRE Portal in the functional landscape of PhenoMeNal. The main functionality of the PhenoMeNal VRE is to allow an external user with cloud credentials to deploy a PhenoMeNal VRE cluster to the user's cloud provider.

3.2 Deployment scenarios

Depending on the circumstances, such as ELSI considerations, clinical institutions may or may not allow external entities to talk to their internal OpenStack installation. In the first case, where the institution exposes at least the REST API endpoints of their OpenStack installation, the external portal running at https://portal.phenomenal-h2020.eu/ can be used through the OpenStack deployment option (see Figure 2, top).

In the second case, where these REST API endpoints are not exposed, the local OpenStack installation can not be invoked through the external EBI TSI Portal, which executes the commands for triggering the deployments, meaning that the external PhenoMeNal portal cannot be used. For such a completely behind-firewall-portal solution we need to run a version of the PhenoMeNal Portal functionality from within the network of that OpenStack installation (see Figure 2, bottom).

This functionality has already been implemented through the existing command-line based `cloud-deploy-kubenow` installation, which has also been used for deployments in the german de.NBI cloud. For a more user-friendly option, we have started joint work with the TSI team at EBI to containerize the EBI TSI Portal for local deployment, to be able to fully run behind firewalls without resorting to the command line. Once this is in place, end users will be able to deploy a local instance of the PhenoMeNal VRE through this local portal in a secure manner, without any outbound network connection. Figure 2 below explains both use cases diagrammatically.

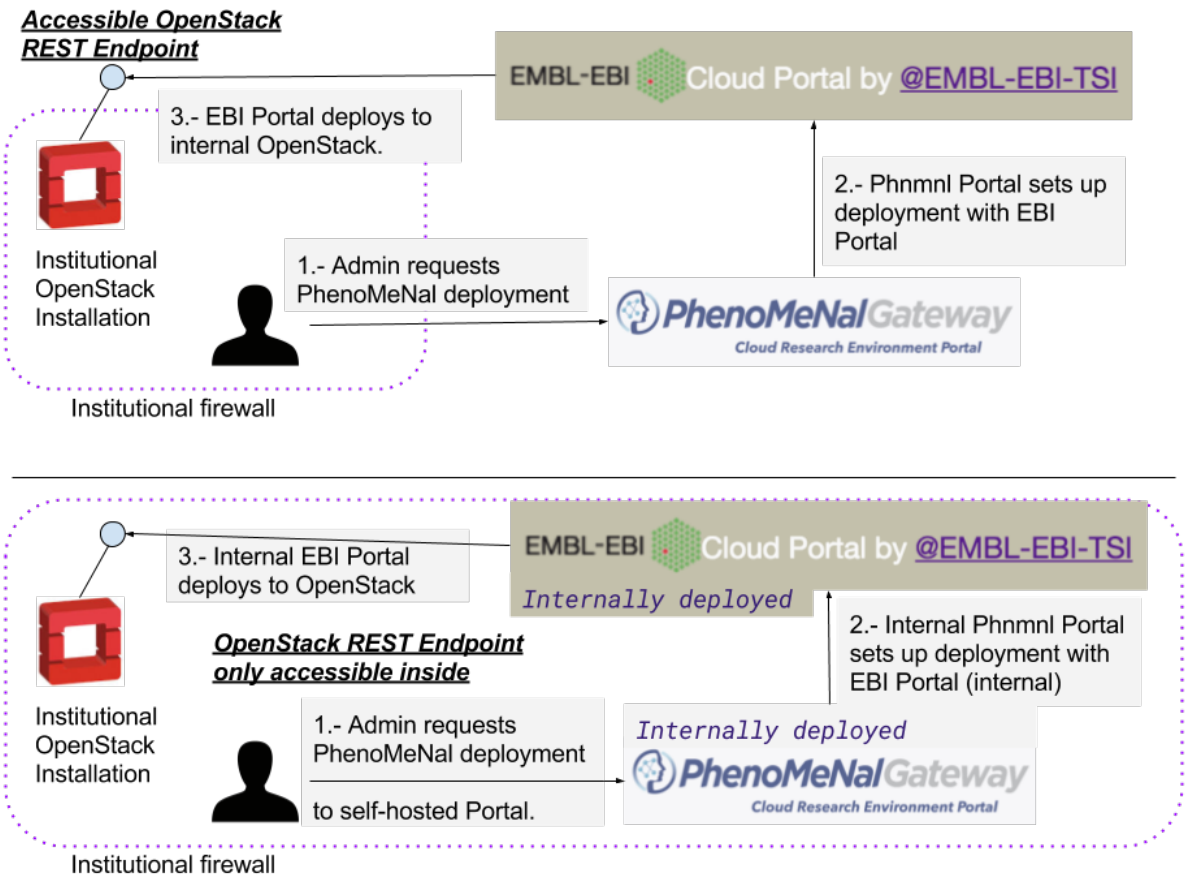

Figure 2: Deployment scenarios based on the networking policy of an institution. **Top**: simplest case, in which the OpenStack REST Endpoint is accessible from outside the institutional firewall, enabling both existing PhenoMeNal Portal and EBI TSI Portal to be used. **Bottom**: OpenStack REST endpoint not accessible from outside the firewall, which requires internally deployed versions of PhenoMeNal Portal and EBI TSI Portal. This second use case currently requires command line usage, but we expect to have a fully deployable EBI Portal in the coming months, which will complete our ability to run the PhenoMeNal Portal locally inside the user premises and in a bring-the-compute-to-the-data fashion (see section on ELSI considerations below).

3.3 Deployment of the portal through Helm charts into Kubernetes clusters

The PhenoMeNal portal helm chart, available at https://github.com/phnmnl/helm-charts/, simplifies the deployment of the following different containers required for the portal:

- container-phenomenal-portal: for the front-end of the portal, including SSO authentication, cloud deployment and facades for the other functionalities. Build history of this container is available at https://phenomenal-h2020.eu/jenkins/job/container-phenomenal-portal/

- container-phenomenal-portal-wiki: the backend functionality for the documentation. Build history of this container is available at https://phenomenal-h2020.eu/jenkins/job/container-phenomenal-portal-wiki/
- container-phenomenal-portal-app-library: backend functionality for the App Library, which showcases the PhenoMeNal containers. Build history of this container is available at: https://phenomenal-h2020.eu/jenkins/job/container-phenomenal-portal-app-library/
- container-portal-metadata-backend: metadata backend to store user settings and status, backed by a MySQL database which is deployed through the stable/mysql helm chart. Build history of this container is available at: https://phenomenal-h2020.eu/jenkins/job/container-portal-metadata-backend/ .

Figure 3 below shows how the different components listed here work together for the operation of the portal. In terms of release, the portal containers follow the same release procedure as used for any other PhenoMeNal container: separate branches (development, release and master) produce different tagged container images (development images, release candidate and release images), all handled by our continuous integration system based on Jenkins.

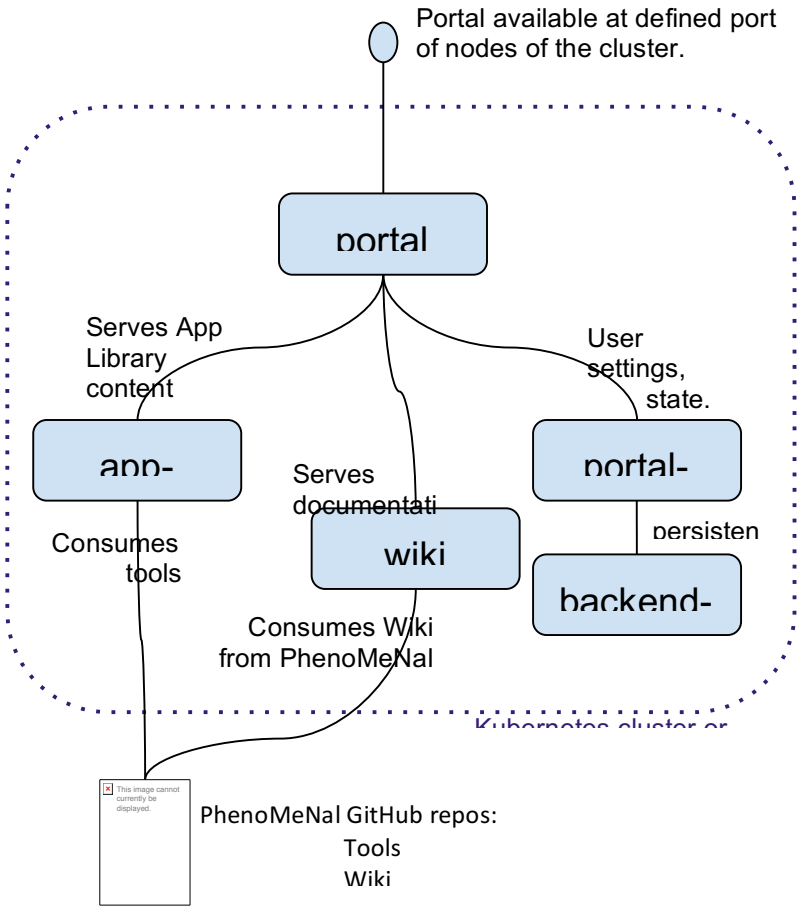

Figure 3: The different components of PhenoMeNal Portal in terms of containers, and how they interact to provide the required VRE functionality.

For deploying the PhenoMeNal Portal on a Kubernetes cluster or a Minikube installation (on a single machine), a configuration file (referred to as config_file.yaml later) like this is written:

```
galaxy_api_key: <galaxy-api-key-for-access-from-portal>
galaxy_url: "https://publicdev.phenomenal-h2020.eu"
image:
  repository: container-registry.phenomenal-h2020.eu/phnmnl/phenomenal-portal
  tag: latest
service:
  externalPort: 30758
backend:
  image:
    tag: latest
wiki:
  image:
    tag: latest
app_library:
  image:
    tag: latest
  resources:
    limits:
      cpu: 1
      memory: 1Gi
    requests:
      cpu: 100m
      memory: 128Mi
mysql:
  mysqlPassword: <a-password-for-mysql>
  persistence:
     enabled: true
     existingClaim: devel-portal-pvc
     subPath: mysql-devel
```

After having that config file, the PhenoMeNal helm chart repo is added to the machine with access to a Kubernetes cluster (ie. can run kubectl towards the cluster) and with helm (https://github.com/kubernetes/helm) installed:

```
$ helm repo add phenomenal https://phnmnl.github.io/helm-charts/
```

Then it can be invoked to launch the portal through:

```
$ helm install -f config_file.yaml phenomenal/portal
```

The helm install line above will deploy all the components of the portal, and requires that a Persistent Volume Claim called “devel-portal-pvc” (the name can be changed in the config file) exists to store the MySQL database content for the backend. These components can be listed through the kubernetes dashboard graphically, or through the following two command calls:

```
$ helm list
NAME                    REVISION    UPDATED                     STATUS    CHART           NAMESPACE
contrasting-dragonfly   1           Sun Aug 20 17:16:39 2017    DEPLOYED  portal-1.1.2    devel

$ kubectl get pods
NAME                                                READY     STATUS    RESTARTS   AGE
contrasting-dragonfly-app-library-473572749-vvj22   1/1       Running   0          19h
contrasting-dragonfly-backend-1359285988-50ck0      1/1       Running   0          19h
contrasting-dragonfly-mysql-4207991844-l323c        1/1       Running   0          19h
contrasting-dragonfly-wiki-1799232970-ghlqn         1/1       Running   0          19h
phenomenal-portal-3774936324-795fs                  1/1       Running   0          19h
```

Technical documentation of the PhenoMeNal Portal infrastructure can be found at https://github.com/phnmnl/phenomenal-h2020/wiki/PhenoMeNal-Portal-Infrastructure

The setup described above is fully operational and was used to deploy both the current versions of https://portal.phenomenal-h2020.eu/ and https://portaldev.phenomenal-h2020.eu/.

3.4 OpenStack compatibility layer

While the resource naming (flavours, networks etc) available at Amazon AWS or Google GCE are known in advance, local OpenStack installations can be configured in different ways, not known to the deployment portal *a priori*. The PhenoMeNal portal now provides a feature that allows users to input their credentials to obtain the required OpenStack configuration metadata by calling OpenStack APIs. For the previous release, users were required to obtain this metadata information from OpenStack manually for OpenStack deployment. For this release, this manual process has been automated that can easily fetch metadata information from OpenStack. As a result, this feature can improve the user interface friendliness. We have contributed all the code necessary to be able to interact with OpenStack installations to the TSI API, so that also any application stack deployments beyond PhenoMeNal will be able to make use of this feature.

3.5 Packaging Application stacks for the EBI Cloud Portal

This section summarises the TSI packaging documentation[3], written by Dario Vianello et al. from the TSI group at the EBI. The document is currently being transferred into a publicly available document and also forms part of EMBL-EBI's ongoing DevOps training offerings.

The EBI Cloud Portal has been built to provide an App-Store-like experience when deploying application stacks to internal and public cloud providers, independently from their complexity. An application stack is defined through a git repository with the following layout:

```
.
├── .gitignore
├── README.md
├── aws
│   ├── ansible -> ../gcp/ansible/
│   ├── deploy.sh
```

[3] https://docs.google.com/document/d/1THnEt64EhBp4Dqh2E3nT8VKShDz-mQHw9MWnlLKeqko

```
│   ├── destroy.sh
│   ├── state.sh
│   └── terraform
├── gcp
│   ├── ansible
│   ├── deploy.sh
│   ├── destroy.sh
│   ├── state.sh
│   └── terraform
├── manifest.json
└── ostack
    ├── ansible -> ../gcp/ansible/
    ├── deploy.sh
    ├── destroy.sh
    ├── state.sh
    ├── terraform
    └── volume_parser.py
```

The core manifest.json file resides at the root of this structure, followed by folders storing code for each respective cloud provider. In this particular repo, the Ansible code is shared among the cloud providers via symbolic/soft links, but this is not a strict requirement. The cloud-deploy-kubenow[4] repository also described in detail in D9.3 follows this layout.

3.5.1 Portal usage

Before deployments can be made, a user first has to configure the intended portal repositories. These are application definitions which are later used for deployment purposes.

Steps:

1. User logs in to EBI Cloud Portal.
2. On the Dashboard user clicks on “Search Repositories” or selects “Repository” from the left pane menu.
3. Click on the “+” button on the right side of the screen.
4. On the “Add application screen” user pastes public repo URL and clicks “Add”.
5. The application is added to your repository.

These steps are already preconfigured and hidden from the user in the PhenoMeNal Portal. Other future projects adopting this approach with a single complex application stack, can start off the cloud-deploy-kubenow repository, and replace the container images that define their service landscape, e.g. with proteomics or transcriptomics specific applications.

3.6 ELSI considerations for the PhenoMeNal Portal

Clinical data managers need to be particularly aware of data security risks and even the term ‘cloud computing’ alone can raise doubts in anyone dealing with patient data in an ELSI compliant manner. Unless explicit patient consent was provided, clinical data managers are bound by contract to ensure complete privacy in order to protect patient rights and adhere to the country of origins ELSI laws and resulting restrictions. Hence the PhenoMeNal project allows for

[4] https://github.com/phnmnl/cloud-deploy-kubenow/

a solution that enables ELSI relevant data to stay and be processed within the safe bounds of a clinical firewall. This approach is here exemplified in a 'bring the compute to the data' philosophy and requires the clinicians to have access to a local cloud environment.

4 DELIVERY AND SCHEDULE

The deliverable was submitted on time.

5 CONCLUSION

The PhenoMeNal infrastructure is available to all end users from the VRE Portal. Users are given the choice of deploying to select OpenStack environments or commercial cloud providers, namely Google GCE and Amazon AWS. All documentation, videos and tutorials are available from the respective sections of the PhenoMeNal Portal.

The VRE Portal gives users access to all workflows and containerised tools therein. The Portal itself is also fully containerised and all container images are publicly available and fully deployable on a larger production Kubernetes cluster or a simpler Minikube installation.

The VRE Portal has been developed using thorough usability testing to ensure interactions and interfaces are as user centred as possible. A further full heuristic analysis of all available features is scheduled for completion in Q3 2017.

The public Portal is available in production use on https://portal.phenomenal-h2020.eu/. More adventurous users can test new pre-release features on https://portaldev.phenomenal-h2020.eu/