Work Package 4, Milestone 4.2: First deployment of the consolidated platform

Besides operating and maintaining the existing application portals and infrastructure, the consolidation work of WP4 was done in three main directions: agreement on common job submission mechanism, virtualization of portals using emerging cloud orchestration technology, and virtualization of the applications themselves, making them ready to be deployed in the cloud. The work progress is tracked at the WP4 wiki pages http://internal-wiki.west-life.eu/w/index.php?title=WP4. This document is a snapshot of the information available there at the time of this milestone, i.e. October 2016.

Common Job Submission

General Schema

The general schema of application portal architecture, which submits the payload of extensive calculations to the Grid, was stabilized over several years in the family of originally WeNMR portals. It is proven to work, even in the intrinsically faulty and not fully reliable distributed environment. Details are described at http://www.wenmr.eu/wenmr/automated-grid-submission-and-polling-daemons. The essential components of the architecture are:

- **Pool of job working directories**, where the user inputs and calculation parameters are stored, and the complete information on job status is maintained in a set of control files.
- **Web input form** which takes the input files and parameters from the user and it performs the following actions synchronously:
  - authenticate and authorize the user with usage of the AAI services
  - store accounting data
  - perform essential sanity checks on input data
  - create the job working directory, where are all the data stored
- **Job controller** (cron job) which is responsible for following the job lifecycle. It cycles through all active job directories in the pool, spawns the pre- and post- processing tasks when appropriate, and it updates the job status, which is available to the user through the web interface in turn.
- **Grid manager** (cron job) is responsible for interaction with the payload submission interface. It formats the job specification and intputs according to the needs of the used submission engine, and it calls the appropriate interfaces. Then the status of the job is monitored periodically, and it is mapped to the internal status of the portal jobs. On successful termination the job outputs are gathered.

Legacy Grid Interface

The original implementation of the schema described in the previous section interfaces the EGI grid infrastructure resources through gLite WMS service (Grid job scheduler). The portals are configured with a static list of WMS instances which take care of distributing the jobs to EGI grid resources sites. The Grid manager component is implemented on top of gLite “user interface” commands.

DIRAC Interface

Software package Dirac (http://diracgrid.org/) is a software framework for distributed computing. It builds common interface for accessing heterogeneous resources. Dirac has been chosen for role of West-Life job dispatcher in the first deployment of consolidated architecture. Details are discussed in Deliverable D4.1, the main reasons for this decision were ability to submit jobs to both grid and cloud part of EGI infrastructure with uniform interface to the user/portal developer, and foreseen sustained support from the DIRAC community.

The DIRAC-based job submission follows exactly the general schema describe above. The only component which requires modification is the Grid manager.

Besides submitting legacy grid jobs to EGI grid infrastructure, the DIRAC submission also allows to run jobs as virtual machines in the EGI Federated Cloud.

Currently, the HADDOCK portal uses DIRAC submission in production.

Application Software Deployment

Typically, application software must be available at the grid worker nodes (or cloud virtual machines) which execute the job payload.

In the grid submission scenario, sites supporting an application are tagged in the grid information system (BDII) and it is the task of WMS to match job requirements with those tags. Typically, CERN VMFS distributed filesystem, developed mostly for this purpose, is used to deploy specific software packages at the sites.

The recent developments leverage the user-space restricted Docker environment UDOCKER (https://www.indigo-datacloud.eu/userspace-container-support) which enables container deployment without superuser privileges, hence it can be used in both legacy grid jobs and virtual machines.

User Identities, Authentication and Authorization

The project deliverable D4.2 “Common security model design” defined a revision of the AAI architecture in the environment of the project. The goal of the architecture is to harmonize existing approaches to AAI and arrive to a common solution that can be generally applicable to all West-Life systems. Since the existing services must continue to work, the introduction of new AAI is split into several steps that will allow for a gradual transition.

This section summarizes the first implementation of AAI components that follow the architecture. We started with focus on provisioning of authentication services and integration of AAI with end services.

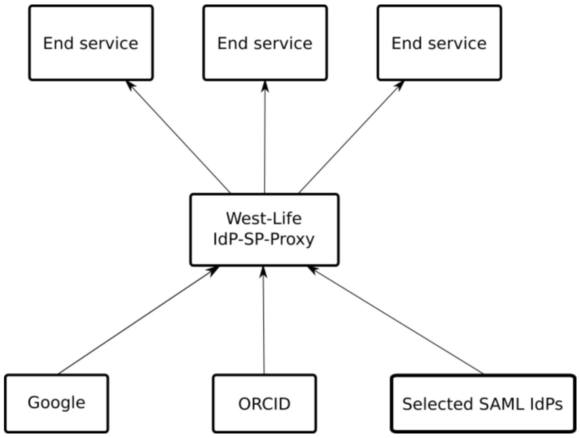
As a crucial component of the new AAI architecture we established an IdP-SP-Proxy service at the testbed. The proxy makes it possible to unify access to multiple identity providers and centralize handling of users' identitifiers and attributes. It removes the need for the service providers to establish links to individual identity providers used by the user community by relying on a single identity services. The centralized nature of the service also simplifies handling of users' attributes and linking multiple identities to a single user record.

The Proxy has been implemented using the SimpleSAMLPhp software. It was deployed on a dedicated virtual machine following standard installation guides. The Proxy was configured to support users' authentication to several identity providers. At the moment, it supports Google, ORCID, and SAML-based IdPs compatible with services used in current R&D identity federations. The functionality is demonstrated at http1s://cloud96b.cerit-sc.cz/simplesaml/module.php/core/authenticate.php?as=multi-auth -- a simple web page which performs the authentication and displays the retrieved user identity.

The Proxy service is not registered yet with any existing identity federation so it cannot leverage from access to EduGain directly. We started discussions about how the current Wenmr SSO systems and users' credentials could be used.

To the end services the Proxy service utilizes the standard SAML protocol. In order to demonstrate a possible integration, we configured an Apache web server to support SAML-based authentication using the auth_mod_mellon module. The service is linked directly to the Proxy, and is not aware of any authentication service interfaced by the proxy. With the authentication being implemented in Apache, the actual application does not need to implement its own authentication steps.

The implemented schema of AAI with the Proxy is depicted in the following picture.



The user is authenticated using the standard schema of federated authentication. When they access an end service first time, their browser is redirected to the Proxy. If they do not have any authentication session kept with the Proxy, they are offered a list of supported identity providers. After they select one, they get redirected again to the identity provider login portal where they present appropriate credentials. After the authentication is verified by the provider they are sent back to the Proxy and in turn to the end service.

To demonstrate integration of a real-world application we have adapted the GROMACS portal, which is used to construct and submit computational jobs. The existing portal requires users to authenticate using their username and password registered with the Wenmr SSO system. The need for supplying these credentials have been removed in our pilot instance and the portal was made to rely on federated authentication facilitated by the Proxy.

The GROMACS portal still needs to make callouts to the central Wenmr portal to keep track of users' activities and to check users' authorizations.

Portal Virtualization

Application web portals hide the complexity of use of the distributed grid infrastructure from the application user. Despite the fact, that this has been proven to work many times, there are also significant drawbacks of this approach. In particular, the setup of a single portal instance, which spawns grid jobs at many sites, becomes rather inflexible – its updates are difficult to perform due to the need of continuous operation serving its large user community. Moreover, it is impossible to meet user requirements once they become contradictory (e.g. the need of a new version of application software vs. requirement to keep strict backward compatibility with an older version of the same software).

It would be possible to deploy and maintain multiple instances of the same portal manually. However, such work is tedious and error prone. On the contrary, *portal virtualization* based on emerging cloud technologies is a promising alternative.

Portal virtualization allows to set up a dedicated instance of an application portal, not relying on a single centrally maintained portal.

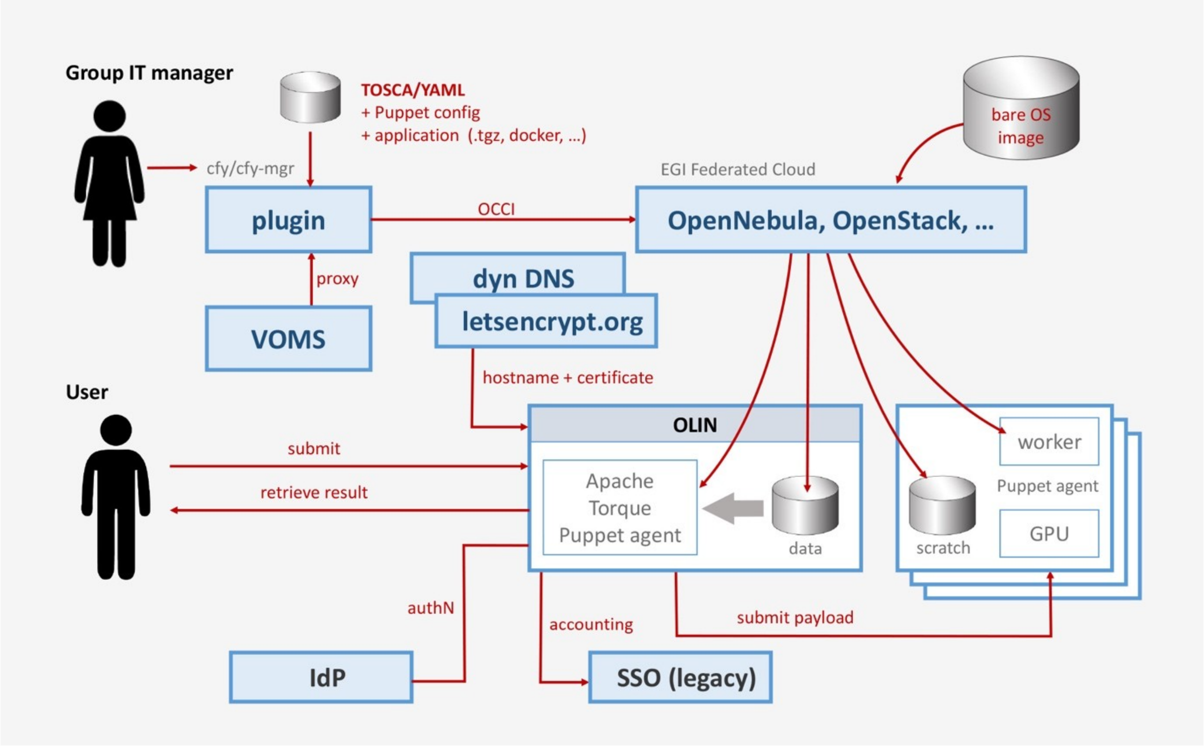
The lifecycle starts with a user group/community negotiating access to resources at a cloud site. Then the group IT manager uses a cloud orchestrator to spawn a portal instance.

Each portal is described as a set of mutually interacting nodes. The portal typically includes the web front end server, auxiliary database server, local batch system server and several local worker nodes). The whole portal can be deployed in the cloud environment in multiple instances using cloud orchestration tools. The reason for several portal instanaces can be for example isolation of user groups, running multiple versions simultaneously, load balancing etc.

The users access this portal through web browsers as before.

The instances of portals spawned in this way are not expected to be long-lived. Instead, new fresh versions are created when e.g. new functionality is added. Therefore, we emphasize on automated implementation of the setup and maintenance of the entire virtualized portal. This approach is called *Infrastructure as a Code*.
In PY1 we provide a proof of concept implementation with orchestration tools Cloudify (http://getcloudify.org) and Infrastructure Manager (http://www.grycap.upv.es/im/). We plan to adopt more complex solutions by INDIGO-Datacloud project.
Details on the choice of the tools are described in Deliverable D4.1

The following diagram shows the overall schema of portal virtualization prototype deployment.



The whole environment is managed by either **Cloudify CLI** (simple, one-shot deployment) or with **Cloudify Manager**. All the required configuration files and recipes (Tosca/YAML, Puppet, ...) are stored in a single directory tree. A prototype for Gromacs portal can be found at https://github.com/vholer/cloudify-gromacs (no guarantee to work out of box yet).
Cloudify uses simple OCCI plugin (https://github.com/vholer/cloudify-occi-plugin-experimental) to access **EGI Federated Cloud** resources. The operations are authenticated and authorized with VOMS proxy of the relevant virtual organization.
According to the TOSCA definition several nodes are spawned. We define an **OLIN node** (phonetic abbreviated spelling of "all in one") which runs

Apache server, the portal logic according to the common schema described above, and **Torque** batch system server to manage the **worker nodes**. The workers are minimalistic installation of Torque mom daemon, the main application code, and its dependencies. The submission to the grid is replaced with submission to local (to the portal) Torque, which is much simpler and more reliable. However, it is still possible to keep the DIRAC submission when submission to remote nodes is required.

We expect the cloud orchestrator to manage the size of the worker node pool dynamically, according to the actual load of the portal (TBD).

The nodes (both OLIN and workers) are instantiated using minimalistic generic VM images with bare operating system, as they are available in EGI VM repository and supported by virtually all EGI Federated Cloud sites. On start (during the cloud contextualization phase) **Puppet agent** is installed on the node, and it takes cares of reliable installation and configuration of all the required software (e.g. the whole web server environment, the portal logic itself, Torque, application and its dependencies) according to the recipes which are part of the virtualized portal definition. We prefer this approach to the alternative of preparing “fat”, fully installed VM images in advance, mostly because of flexibility. There is no need to register new images with EGI and to propagate them to sites repeatedly.

Currently we deploy the application as a tarball. Docker containers could be used in future, if it is the preferred way of distributing specific applications.

Besides the nodes (virtual machines) the cloud orchestration also creates required storage volumes and attaches them to the nodes. The current prototype (Gromacs usecase) uses dedicated storage of two types – the job pool at the OLIN node and scratch space at the workers. Inputs and outputs are staged in/out with the Torque jobs. In deployments which require POSIX access to shared disk space (e.g. Scipion) we expect to expert such filesystem either from extended OLIN or a dedicated storage node.

The OLIN node, which exposes the web front end to the users, uses **dynamic DNS** service to associate defined hostname with the IP address assigned to it by the cloud. Because encrypted communication with the user is required, the node also requests an X.509 certificate from an **online certification authority** service to be used by used by the web server. In order to improve users’ experience, we use the *Let’s encrypt* service (http://letsencrypt.org) that issues certificates accepted by mainstream web browsers.

Users are authenticated using West-life **Identity-service provider proxy** (described above), **authorization** and **accounting** is done by remote calls to the SSO service (legacy WeNMR at the moment).

Application Cloud Deployment

In parallel with the portal virtualization activity described above, experimental work to set up several applications in the cloud directly was performed. We summarize the experience here.

DisVis and PowerFit

Reliable bioinformatics and computational methods are needed for complementing experimental techniques, to study structure-dynamics-interactions of biomolecular machines at atomistic level. Accordingly, DisVis and PowerFit are software packages developed at Bonvin lab (http://www.bonvinlab.org) for visualization and quantification of the accessible interaction space of distance restrained binary biomolecular complexes and for automatic rigid body fitting of biomolecular structures in Cryo-EM densities, respectively. These softwares are Python-based, capable of harnessing multiple CPUs and GPGPU, and have similar complex dependencies on various packages making them rather difficult to install for end users. Using INDIGO-DataCloud (https://www.indigo-datacloud.eu/) tools, we aim at virtualizing DisVis and PowerFit to decrease the dependency on local hardware and also facilitate deployment, configuration and customization of these software packages.

For this purpose, Infrastructure Manager (IM) was used to deploy a virtual machine on EGI Federated Cloud , with a bare OS image with Ubuntu 14.04 and Docker containers support (selected from EGI application catalog at https://appdb.egi.eu/). Docker is a convenient solution for packaging the applications together with their dependencies. Docker images for DisVis and also for PowerFit are available on Docker Hub within INDIGO-DataCloud project repository (https://hub.docker.com/r/indigodatacloudapps/).

Using IM web interface (http://servproject.i3m.upv.es/im/index.php), the user logins and sets his/her credentials according to the cloud provider that he/she chooses. In case of OCCI, the user should generate a grid certificate and upload it and also specify the host, which is the endpoint information obtained from EGI Application Database. As a side note, one can store his/her credentials not only for one but multiple resource providers if needed.

After setting the credentials, the user designs a Resource Application and Description Language (RADL) script according to the base image specifications he/she selects from EGI Application Database. It is also possible to launch already available scripts on RADL tab of IM. The design of RADL script for OCCI-EGI Federated Cloud case is rather simple; the user specifies the endpoint, the instance type, the template name, which can be obtained from image properties at EGI Application Database website (https://appdb.egi.eu/). Once the script is ready, the user launches the VM and wait for 5-10 minutes for it to be configured. When it’s configured, the user can access the VM by downloading the private key and using the IP available on IM. In VM, the user can run Docker containers for DisVis and PowerFit by pulling the images from Docker Hub. If the user does not need the VM anymore, he/she can delete it using IM.

Scipion

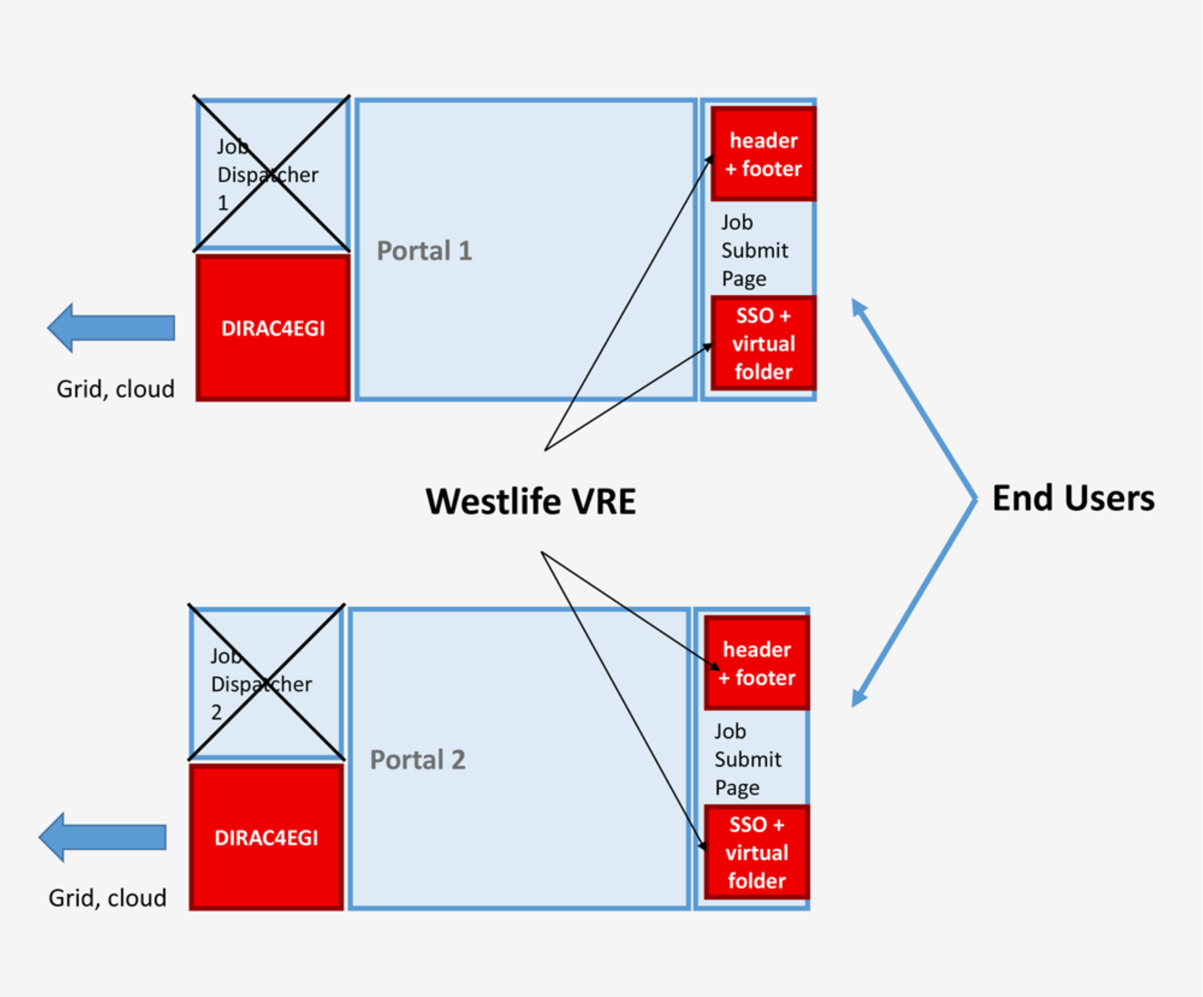
We have released a first version of ScipionCloud virtual appliance on the EGI AppDB that can be used to instantiate Scipion desktop on the EGI Federated Cloud. The appliance is available at https://appdb.egi.eu/store/vappliance/scipion.v1.0. This image provides Scipion 1.0.1 to be run on a single node plus remote desktop guacamole installed and configured to access the Virtual Machine through a Web Browser.

Commonly, applications developed for cloud have an optimized web front-end that works well with very low network resources. In the case of Scipion desktop and the EM packages it incorporates, their interfaces are based on X11, which was developed for local area network operation. Therefore, when accessing the machines remotely those interfaces feel too slow for interactive use. This is not a problem specific to EM software and different solutions have been developed to address this issue. In ScipionCloud we have tested VNC (www.realvnc.com), x2go (http://x2go.org) and Guacamole (www.guac-dev.org) and we have chosen the latest since it is the one that provides a level of interactive performance similar to a local desktop without requiring any software or plugin installation on the client side.

Additionally, some applications display 3D graphics based on OpenGL. When such graphics are rendered locally, GPUs can be used to accelerate this rendering, but in the cloud context the rendering is done in the cloud instance for which we provide non-accelerated OpenGL support that does not require a GPU.

Integration at the VRE Level

Virtual research environment (VRE) layer provides uniform interface for job submission at end-user access level. The general scheme is shown in the figure.



The principle of integration solution on VRE level is based on integration of job submit pages, which are part of all portal solutions. Job submit pages are interface used by end-user during job submission. Integration is based on inclusion of VRE integration components into this pages. The integration components implement integration functionality.
The key integration functionalities are following:

- **SSO** implements functionality of single sign-on and unified authentication and authorization. This functionality is based on architecture proposed in the deliverable 4.2 Common security model design.
- **Virtual folder** functionality provides users with uniform access to their data stored at different storage providers. Four data providers are supported since VRE prototype implementation. The supported ones are EUDAT (B2DROP), Dropbox, Google Drive and Amazon S3. This functionality is being developed as part of Work Package 6.

The job data received via the job submit page are transmitted to the internal workflow of the portal. If the internal workflow requires submission of jobs to external grid or cloud infrastructure, the DIRAC interface is used for jobs submission to EGI infrastructure.

For historical reasons a number of portals use different interface for submitting to the EGI grid infrastructure. Gradual substitution of these interfaces with unified interface based on DIRAC technology is planned.