EOSC Technical Specification

PaaS Orchestration

Technical Area:	PaaS Solutions
Version:	1.2
Status:	Public
Document Link:	https://confluence.egi.eu/pages/viewpage.action?pageId=52598373

OSC-HUB RECEIVES FUNDING FROM THE EUROPEAN UNION'S HORIZON 2020 RESEARCH AND INNOVATION PROGRAMME UNDER RANT AGREEMENT NO. 777536.

DELIVERY SLIP

Date	*Name*	*Partner/Activity*
From:	Marica Antonacci	INFN
Moderated by:	Marica Antonacci	INFN
Reviewed by:	Alessandro Costantini Giacinto Donvito German Molto	INFN INFN UPV
Approved by:	TCOM	

DOCUMENT LOG

Issue	*Date*	*Comment*	*Author*
1.0	15/05/20	Document ready for review	Marica Antonacci (INFN)
1.1	16/05/20	Comments from reviewers addressed	Marica Antonacci (INFN)
1.2	17/05/20	Final version	Marica Antonacci (INFN)

TERMINOLOGY

https://wiki.eosc-hub.eu/display/EOSC/EOSC-hub+Glossary

Terminology/Acronym	*Definition*
AAI	Authentication & Authorization Infrastructure
HPC	High Performance Computing
PaaS	Platform as a Service
TOSCA	Topology and Orchestration Specification for Cloud Applications)

Table of Contents

Introduction 3
Adopted standards 3
High-level Service Architecture 4
Interoperability guidelines 5
Examples of solutions implementing this specification 7
Procedure to integrate a service with the EOSC Hub PaaS Orchestration 7
References 7

Introduction

The PaaS (Platform as a Service) solution adopted in this project allows the users to deploy virtualised computing infrastructures with complex topologies (such as clusters of virtual machines or applications packaged as Docker containers), by using standardized interfaces based on REST APIs and adopting the TOSCA (Topology and Orchestration Specification for Cloud Applications) templating language for the description of Cloud-based applications.

The PaaS layer features advanced federation and scheduling capabilities ensuring the transparent access to the different IaaS back-ends, including on-premises Cloud Management Frameworks such as OpenStack and OpenNebula, public Cloud providers such as Amazon Web Services and Microsoft Azure and finally, Container Orchestration Platforms such as Apache Mesos and Kubernetes.

The selection of the best cloud provider to fulfill the user request is performed considering criteria like the user's SLAs, the services availability and the data location.

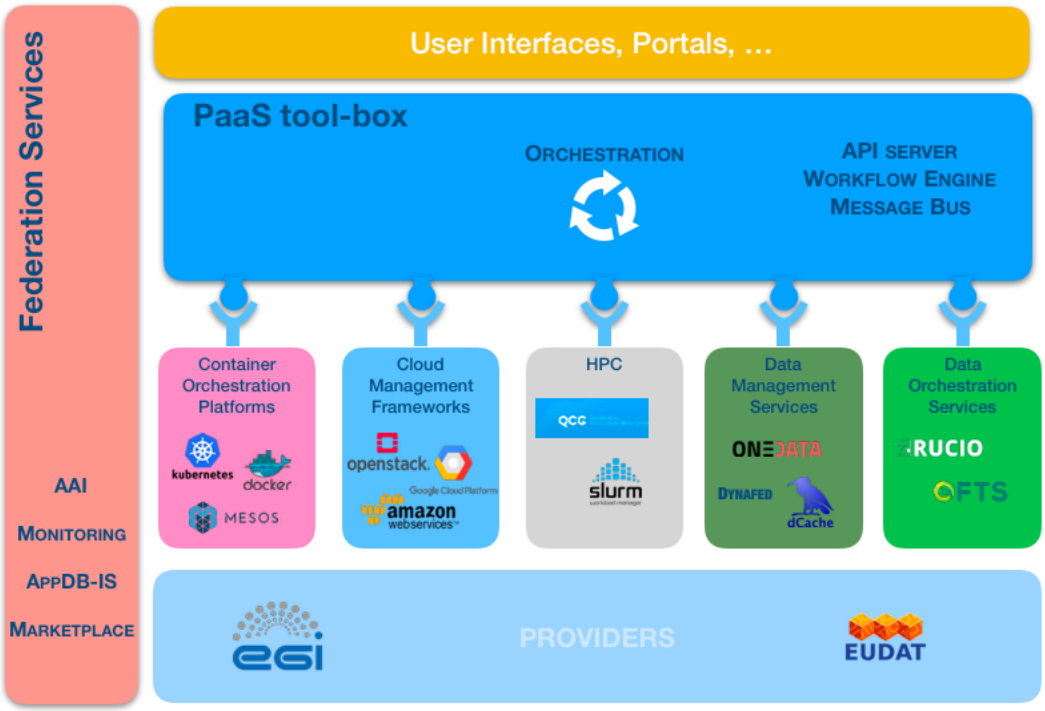
Adopted standards

Standard	Short description	References
TOSCA	OASIS open standard that defines the interoperable description of services and applications hosted on the cloud and elsewhere; including their components, relationships, dependencies, requirements, and capabilities, thereby enabling **portability** and **automated management** across cloud providers regardless of underlying platform or infrastructure.	http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.2/csprd01/TOSCA-Simple-Profile-YAML-v1.2-csprd01.html#_Toc503782167
Oauth2.0 Authorization Framework	The OAuth 2.0 authorization framework enables a third-party application to obtain limited access to an HTTP service, either on behalf of a resource owner by orchestrating an approval interaction between the resource owner and the HTTP service, or by allowing the third-party application to obtain access on	https://tools.ietf.org/html/rfc6750

	its own behalf.	
REST	REST, or REpresentational State Transfer, is an architectural style for providing standards between computer systems on the web, making it easier for systems to communicate with each other. REST-compliant systems, often called RESTful systems, are characterized by how they are stateless and separate the concerns of client and server.	https://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#relwwwrest

High-level Service Architecture

The high-level reference architecture is depicted in the following diagram.



High-level Architecture for the PaaS orchestration

The architecture can be broken down into the following main categories of components:

- Core components:
  - *API server*, providing REST endpoints to submit and handle the deployment requests;
  - *Workflow Engine*, that manages the deployment workflow;
  - *Message Bus*, providing a way of integrating services loosely and based on notifications (events).
- Plugins
  - *Cloud connectors*, implementing the interfaces with the relevant Cloud Management Frameworks.
  - *Container orchestration connectors*, implementing the interfaces that abstract the interaction with the relevant container orchestration platforms, e.g. Mesos, Kubernetes.
  - *HPC integration connectors*, implementing the interfaces to interact with the HPC services; the envisage interaction is based on REST APIs provided by gateway hosted by the HPC site, e.g. using QCG APIs [R1] or SLURM APIs [R2].
  - *Storage services connectors*, implementing the interfaces to interact with the relevant storage management and orchestration services; the interaction is based on REST APIs provided by the storage services themselves.

Moreover, the following dependencies towards integration with the Federation Services are envisaged:

- EOSC-hub AAI, to ensure the federated access to the services and resources;
- AppDB-IS or AMS (optional): information published by the providers (e.g. available images, flavors, networking configurations, service endpoints, etc.) can be used by the PaaS tools;
- EOSC-Hub Monitoring (optional): information about the health status of the services can be usefully exploited by the PaaS orchestrators in order to select the best sites for scheduling the user requests;

Marketplace (optional): information collected in the Marketplace can be consumed by the PaaS tools.

Interoperability guidelines

The adoption of the TOSCA standard can help to reach a good level of interoperability among different services in this area. However, this is a necessary but not a sufficient condition since the full interoperability would require the adoption of the same TOSCA custom types (in addition to the normative ones) and of the same REST API specifications.

Currently there is not an official standard for the PaaS orchestration APIs, but we propose as reference the APIs implemented by the INDIGO PaaS Orchestrator:

https://indigo-dc.github.io/orchestrator/restdocs/

The API is organized around REST and has predictable resource-oriented URLs, accepts JSON-encoded request bodies, returns JSON-encoded responses, and uses standard HTTP response codes, authentication, and verbs.

Authentication is performed via OAuth2 bearer token.

The core resource is Deployment that represents a TOSCA template deployment.

The next table shows the main endpoints to manage the deployment objects:

Endpoint	**Description**
GET /deployments	Retrieve the deployments list
POST /deployments	Create a new deployment
GET /deployments/:id	Retrieve the details of a deployment that has previously been created (e.g. the status of the deployment, the endpoint of the deployed service and the credentials to access it, etc.)
GET /deployments/:id/log	Retrieve the log of a deployment that has previously been created
GET /deployments/:id/extrainfo	Retrieve additional information of a deployment that has previously been created
GET /deployments/:id/template	Retrieve the template of a deployment that has previously been created
PUT /deployments/:id	Update a deployment that has previously been created
DELETE /deployments/:id	Delete a deployment that has previously been created

Examples of solutions implementing this specification

- INDIGO PaaS Orchestrator: https://github.com/indigo-dc/orchestrator
    - The PaaS Orchestrator allows coordinating the provisioning of virtualized compute and storage resources on different Cloud Management Frameworks (like OpenStack, OpenNebula, AWS, etc.) and the deployment of dockerized services and jobs on Mesos clusters. Moreover, recently a plugin for the HPC integration has been added in order to submit and manage batch jobs through QCG gateways.
    - The PaaS orchestrator features advanced federation and scheduling capabilities ensuring the transparent access to heterogeneous cloud environments and the selection of the best resource providers based on criteria like user's SLAs, services availability and data location.
- Infrastructure Manager (IM): https://www.grycap.upv.es/im/index.php
    - IM is a tool that deploys complex and customized virtual infrastructures on multiple back-ends. The IM automates the Virtual Machine Image (VMI) selection, deployment, configuration, software installation, monitoring and update of virtual infrastructures. It supports a wide variety of back-ends, thus making user applications Cloud agnostic. In addition, it features DevOps capabilities, based on Ansible to enable the installation and configuration of all the user required applications providing the user with a fully functional infrastructure.

Procedure to integrate a service with the EOSC Hub PaaS Orchestration

As already stated in the previous paragraph, the PaaS Orcehstration specification has been implemented in different solutions and related services. For the adoption of the technical specification, a resource/service provider has to refer to the integration procedure of the specific service.

References

[R1] http://www.qoscosgrid.org/trac/qcg-computing

[R2] https://slurm.schedmd.com/api.html