Noname manuscript No.
(will be inserted by the editor)

Orchestrating complex application architectures in heterogeneous clouds

Miguel Caballer · Sahdev Zala · Álvaro López García · Germán Moltó · Pablo Orviz Fernández · Mathieu Velten

Received: date / Accepted: date

Abstract Private cloud infrastructures are now widely deployed and adopted across technology industries and research institutions. Although cloud computing has emerged as a reality, it is now known that a single cloud provider cannot fully satisfy complex user requirements. This has resulted in a growing interest in developing hybrid cloud solutions that bind together distinct and heterogeneous cloud infrastructures. In this paper we describe the orchestration approach for heterogeneous clouds that has been implemented and used within the INDIGO-DataCloud project. This orchestration model uses existing open-source software like OpenStack and leverages the OASIS Topology and Specification for Cloud Applications (TOSCA) open standard as the modelling language. Our approach uses virtual machines and Docker containers in an homogeneous and transparent way providing consistent application deployment for the users. This approach is illustrated by means of two different use cases in different scientific communities, implemented using the INDIGO-DataCloud solutions.

Keywords Cloud-Computing · Heterogeneous-Cloud · Multi-Cloud · Open-Source · TOSCA

A. López García · P. Orviz Fernández
Instituto de Física de Cantabria.
Centro Mixto CSIC - UC. Santander, Spain.
E-mail: aloga@ifca.unican.es

Miguel Caballer · Germán Moltó
Instituto de Instrumentación para Imagen Molecular (I3M).
Centro Mixto CSIC - Universitat Politècnica de València. Valencia, Spain.

Sahdev Zala
IBM Raleigh. North Carolina. USA.

Mathieu Velten
European Organization for Nuclear Research (CERN).
Geneva, Switzerland.

1 Introduction

The scientific exploitation of cloud resources is nowadays a reality. Large collaborations, small groups and individual scientists have incorporated the usage of cloud infrastructures as an additional way of obtaining computing resources for their research. However, in spite of this large adoption, cloud computing still presents several functionality gaps that make difficult to deliver its full potential, specially for scientific usage [74,75]. One of the most prominent challenges is the lack of elasticity and transparent interoperability and portability across different cloud technologies and infrastructures [70,47, 21]. It is absolutely needed to provide users with seamless dynamic elasticity over a large pool of computing resources across multiple cloud providers.

Commercial providers can create this illusion of infinite resources (limited by the amount of money that users can afford to pay), but this is not true in scientific datacenters, where resources tend to be more limited or are used in a more saturated regime [75,79]. In this context, it is a clear requirement that a user application must be capable of spanning over several different and heterogeneous infrastructures as a way to obtain the claimed flexibility and elasticity. Orchestrating multiple IaaS (Infrastructure as a Service) resources requires deep knowledge on the infrastructures being used, something perceived as too low level by normal scientific users. However, it is possible to hide this complexity moving the user interaction up in the cloud model stack. This way users do not deal anymore with infrastructure resources, but rather interact with PaaS (Platform as a Service) or SaaS (Software as a Service) resources. In these cases, the orchestration complexity is carried out by the middleware layer that provides

the platform or software as a service, thus the low level details can be hidden to the users.

In this work we will describe how the INDIGO-DataCloud project [30] is overcoming this limitation by providing a mechanism to orchestrate computing resources across heterogeneous cloud infrastructures. We will also thoroughly describe how this solution is being exploited to deliver the execution of complex scientific applications to the final users. INDIGO-DataCloud is an European Union's Horizon 2020 funded project that aims at developing a data and computing platform targeting scientific communities, deployable on multiple hardware and provisioned over hybrid (private or public) e-infrastructures. INDIGO-DataCloud is helping application developers, e-infrastructures, resource providers and scientific communities to overcome current challenges in the cloud computing, storage and network areas, being the orchestration across heterogeneous providers one of the project's main objectives.

The remainder of this paper is structured as follows. In Section 2 we describe related work in the area. In Section 3 we describe the INDIGO-DataCloud overall approach including the technology choices that the project has made. In Section 4 we include a high-level architectural description of the orchestration technique that INDIGO-DataCloud is implementing. Section 5 contains some selected use cases, in order to illustrate the architecture previously described. Finally, we present our conclusions and the future work in Section 6.

2 Background and related work

The usage and promotion of open standards (being TOSCA [43] one of them) in the cloud as a way to obtain more interoperable, distributed and open infrastructures is a topic that has been already discussed [80, 46]. Major actors [40] agree that these principles should drive the evolution of cloud infrastructures (specially scientific clouds [45]) as the key to success over closed infrastructures.

As a matter of fact, the European Commission recommended, back in 2004, the usage of Open Standards in its "European Interoperability Framework for pan-European eGovernment Services" [25]. In the same line, the United States (US) National Institute of Standards and Technology (NIST) has encouraged US national agencies to specify cloud computing standards in their public procurement processes [7]. Similarly, the United Kingdom Government provided a set of equivalent principles in 2014 [82].

However, providers and users perceive that lower level (i.e. infrastructure provision and management) standards hinder the adoption of cloud infrastructures [9]. Cloud technologies and frameworks tend to have a fast development pace, adding new functionalities as they evolve, whereas standards' evolution is sometimes not as fast as the underlying technologies. This has been perceived as a negative fact limiting the potential of a given cloud infrastructure, that sees its functionality and flexibility decreased. On top of this, infrastructure management is also perceived too low level when moving to more service-centric approaches that require not only the deployment and management of services, but also all their operational concerns (like fault or error handling, auto-scaling, etc.) [44].

In this service-centric context, cloud orchestration is being considered more and more important, as it will play the role needed to perform the abstractions needed to deploy complex service architectures for a wide range of application domains, such as e-government, industry and science. Cloud orchestration involves the automated arrangement, coordination and management of cloud resources (i.e. compute, storage and network) to meet to the user's needs and requirements [8]. Those requirements normally derive from the user demand of delivering a service (such as a web service where there is a need to orchestrate and compose different services together), performing a business logic or process, or executing a given scientific workflow.

Cloud orchestration within science applications has been tackled before by several authors, in works related to specific scientific areas such as bioinformatics and biomedical applications [41], neuroscience [79], phenomenology physics [9], astrophysics [77], environmental sciences [18], engineering [39], high energy physics [81], etc. This has been also addressed in more generic approaches, not bounded to a scientific discipline [84, 38,15]. However, these works tend to be, in general, too tied to a given type of application and workload, and they need to be generalized in order to be reused outside their original communities.

Several open source orchestration tools and services exist in the market, but most of them come with the limitation of only supporting their own Cloud Management Platforms (CMPs) as they are developed within those project ecosystems. As an example we can cite some of them: OpenStack Heat [61] and its YAML-based Domain Specific Language (DSL) called Heat Orchestration Template (HOT) [59], native to OpenStack [60]. OpenNebula [58] also provides its own JSON-based multi-tier cloud application orchestration called OneFlow [57]. Eucalyptus [17] supports orchestration via its implementation of the AWS CloudFormation [4] web service. All of them are focused on the their own CMPs and furthermore they rely on their own DSL languages (open or proprietary ones such as CloudFormation).

Moving from the CMP specific tools and focusing on other orchestration stacks we can find: Cloudify [13], which provides TOSCA-based orchestration across different Clouds, but is not currently able to deploy on OpenNebula sites, one of the main CMPs used within science clouds being supported by the project. Apache ARIA [6] is a very recent project, not mature enough and without support for OpenNebula. Project CELAR [10] used an old TOSCA XML version using SlipStream [78] as the orchestration layer (this project has no activity in the last years and SlipStream has the limitation of being open-core, thus not supporting commercial providers in the open-source version). CompatibleOne [83] provided orchestration capabilities based on the Open Cloud Computing Interface (OCCI) [54,49,50]. However the project has not been active in the last years. OpenTOSCA [67] currently only supports OpenStack and EC2 providers.

In contrast, the Infrastructure Manager (IM) [26] supports TOSCA-based deployments over a variety of cloud backends including OpenNebula and OpenStack, the two main CMPs targeted in the project; commercial cloud providers such as Microsoft Azure [51], Amazon Web Services (AWS) [3], Google Cloud Platform (GCP) [23] and Open Telekom Cloud (OTC) [56]; and the EGI Federated Cloud [16] a large-scale pan-european federated IaaS Cloud to support scientific research.

As we can see, CMP-agnostic tools tend to move away from specific DSLs and to utilize open standards such as OCCI or TOSCA. Although both may seem suitable for orchestration purposes, OCCI is an standard focused on all kind of management tasks [46], whereas TOSCA is a standard designed specifically to model cloud-based application architectures. Choosing TOSCA as the description language for an orchestration tool is a reasonable choice, as we will later describe in Section 3.1.

3 INDIGO-DataCloud vision

The project's design specification [27] has put the focus not only on evolving available open-source cloud components, but also on developing new solutions to cope with the project targets, introducing as a result innovative advancements at the layer of IaaS, e.g. by implementing advanced scheduling strategies based on fair share or preemptible instances [22], at the layer of PaaS, e.g. by creating SLA-based orchestration components that support deployments on multi-Clouds [76] and, finally, at the layer of SaaS, e.g. by developing high-level REST and graphical user interfaces to facilitate the usage of computing infrastructures for different scientific communities [71].

The heterogeneity in the IaaS platforms has been addressed by the adoption of the two leading open-source CMPs, *OpenStack* [60] and *OpenNebula* [58]. *OpenStack* is a major open-source collaboration that develops a cloud operating system that controls large pools of compute, storage, and networking resources throughout a datacenter. *OpenNebula* provides a simple but feature-rich and flexible solution for the comprehensive management of virtualized data centers to enable private, public and hybrid IaaS clouds. Both are enterprise-ready solution that include the functionality needed to provide an on-premises (private) cloud, and to offer public cloud services. However, the disparity in the maturity level of the key components needed by INDIGO-DataCloud project have brought along multiple developments in the codebase of both CMPs, in many cases resulting in upstream contributions [65,63].

3.1 TOSCA

The interoperability required to orchestrate resources either in *OpenStack* or *OpenNebula* from the PaaS layer has been provided by the usage of the TOSCA (Topology and Specification for Cloud Applications) [43] open standard. TOSCA is a Domain Specific Language (DSL) to describe cloud application architectures, developed by the OASIS [55] nonprofit consortium and supported by several companies as contributors, reviewers, implementers or users. These companies include, AT&T, Bank of America, Brocade, Cisco, Fujitsu, GigaSpaces, Huawei, IBM, Intel, Red Hat, SAP, VMWare, Vnomic and ZTE corporation. The usage of TOSCA was already made practical by OpenStack projects like TOSCA Parser [66] and Heat Translator [62]. Both projects are easy to consume via Python Package Index (PyPI) [73] packages or directly from the master branch of the source code.

INDIGO-DataCloud adopted the TOSCA language as it allows to define interoperable descriptions of cloud applications, services, platforms, data and infrastructure along with their requirements, capabilities, relationship and policies. TOSCA enables portability and automated management across multiple clouds regardless of the underlying platform or infrastructure and is supported by a large and growing number of international industry leaders.

TOSCA uses the concept of *service templates* to describe cloud application architectures as a *topology template*, which is a graph of *node types* (used to describe the possible building blocks for constructing a service template) and *relationship types* (used to define lifecycle operations to implement the behavior an orches-

tration engine can invoke when instantiating a service template).

Three additional open software components are taking part on the orchestration solution in the INDIGO-DataCloud project: *TOSCA Parser* is an OpenStack open-source tool to parse documents expressed using the TOSCA Simple Profile in YAML [69]. *Heat Translator* translates non-Heat templates (e.g. TOSCA templates) to the native OpenStack's orchestration language HOT (Heat Orchestration Template). Last but not least, the *Infrastructure Manager* (IM) [8] is a TOSCA compliant orchestrator, which relies on the *TOSCA Parser*, that enables the deployment and configuration of the virtual infrastructures over a large set of different on-premises CMPs (e.g. OpenNebula and OpenStack) and public Cloud providers.

3.2 TOSCA Parser

The TOSCA Parser is an OpenStack project, although it is a general purpose tool, not restricted to be used within an OpenStack environment. The TOSCA Parser is a Python library able to read TOSCA simple YAML templates, TOSCA Cloud Service Archive (CSAR) and TOSCA Simple Profile for Network Functions Virtualization (NFV), creating in-memory graphs of TOSCA nodes and their relationship, as illustrated in Figure 1.

3.3 Heat Translator

The OpenStack Heat Translator project enables integration of TOSCA into an OpenStack cloud. With Heat Translator a user can translate TOSCA templates to the OpenStack native HOT language. These templates can then be automatically deployed in an OpenStack cloud (Figure 2).

The Heat Translator project can be used directly from the OpenStack command line and web user interface, and is well integrated into the OpenStack ecosystem (Figure 3). It uses various OpenStack projects for translation purposes (like image and instance type mapping) and it is also consumed by other OpenStack official projects like the OpenStack NFV Orchestration project, Tacker [64].

Listing 1 shows a TOSCA document that, when passed to the Heat Translator, results in the HOT output shown in Listing 2.

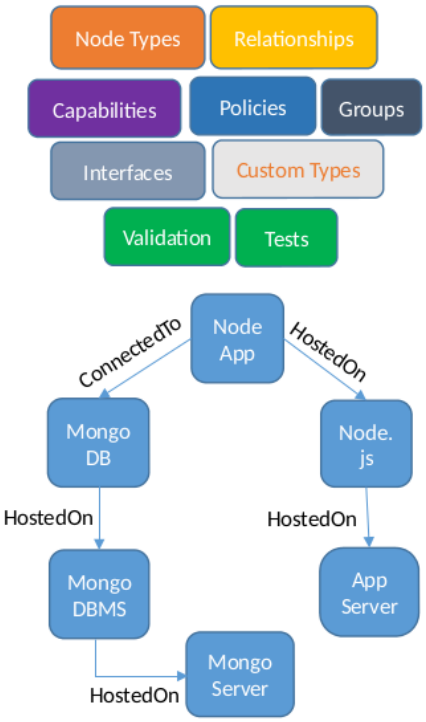

Fig. 1 The in-memory representation of TOSCA nodes.

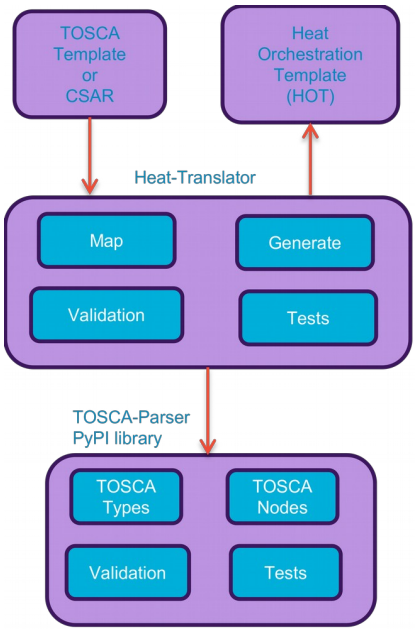

Fig. 2 Heat Translator Architecture.

Listing 1 TOSCA document equivalent to Figure 2.

```
tosca_definitions_version:
    tosca_simple_yaml_1_0

topology_template:
```

```
node_templates:
  my_server:
    type: tosca.nodes.Compute
    capabilities:
      host:
        properties:
```

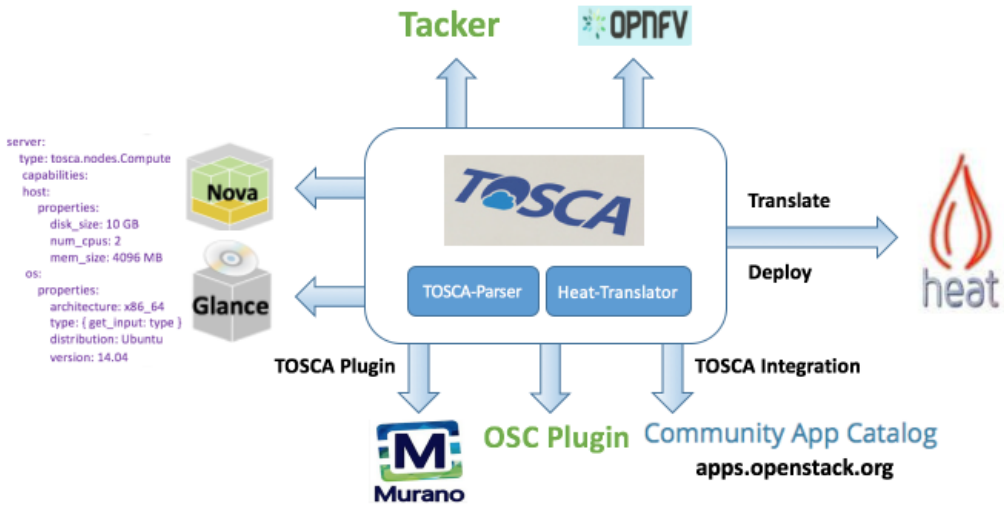

Fig. 3 Heat Translator OpenStack integration.

```
            num_cpus: 2
            disk_size: 10 GB
            mem_size: 512 MB
          os:
            properties:
              architecture: x86_64
              type: linux
              distribution: RHEL
              version: 6.5
```

Listing 2 HOT document equivalent to Listing 1.

```
heat_template_version: 2013-05-23

parameters: {}
resources:
  my_server:
    type: OS::Nova::Server
    properties:
      flavor: m1.medium
      image: rhel-6.5-test-image
      user_data_format: SOFTWARE_CONFIG
outputs: {}
```

3.4 Infrastructure Manager

The Infrastructure Manager (IM) [8] performs the orchestration, deployment and configuration of the virtual infrastructures and it was chosen within the project as it provided support for TOSCA based orchestration, as long as a wide variety of backends and infrastructures (as described in Section 2), specially those targeted by the project.

Figure 4 shows the scheme of the IM procedure. It receives the TOSCA template and contacts the cloud site using their own native APIs to orchestrate the deployment and configuration of the virtual infrastructure. The IM also uses the TOSCA parser to parse and load in memory the TOSCA documents received as input. Once the resources have been deployed and they are running, the IM selects one of them as the "master" node and installs and configures Ansible [24] to launch the contextualization agent that will configure all the nodes of the infrastructure. The master node requires a public IP accessible from the IM service and must be connected with the rest of nodes of the infrastructure (either via a public or private IP). Once the node is configured, the IM will launch the contextualization agent to configure all the nodes using the defined Ansible playbooks. Ansible was chosen over other DevOps tools such as Puppet, Chef or SaltStack due to the combination of the following features: i) YAML support, the same language used to define the TOSCA templates, ii) Ansible Galaxy [28], an online repository to share with the community the open-source Ansible roles created to dynamically install the services and end-user applications, iii) easy to install tool, and iv) agent-less architecture enabling the management of the nodes without requiring any pre-installed software (using standard SSH or WinRM connections).

4 TOSCA orchestration in INDIGO-DataCloud

INDIGO-DataCloud provides a comprehensive solution for deploying cloud applications in multiple CMPs that may need complex topologies and operational requirements, such as auto-scaling resources according to the application needs.

As explained in Section 3, this cloud orchestration scenario is perfect for using TOSCA to specify resources

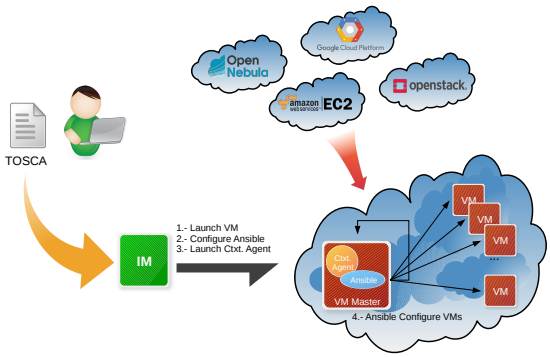

Fig. 4 The Infrastructure Manager (IM).

in heterogeneous environments, guiding the operation management throughout the application lifecycle. The tools described in the previous section provide the functionalities needed to cope with the orchestration needs of those cloud applications at the infrastructure and platform level.

Basically the implemented solution enables a user to deploy cloud applications over complex cloud infrastructures. The user interacts with a set of APIs or GUI based portlets that enable the definition of the relevant parameters for the application and infrastructure deployment. This is internally managed as a TOSCA document that is sent to the different components of the architecture to manage the lifecycle of the cloud topology: selecting the best image and cloud site to deploy the infrastructure and then contacting the TOSCA orchestration endpoint (IM or Heat) for the selected site.

Our approach proposes a new solution to deploy the final application at each resource provider by combining the usage of Docker containers and VMs in a transparent way for the user. It leverages Docker containers as the preferred underlying technology to encapsulate user applications [31], but in case that containers are not supported natively by the CMP or provider, it can also use virtual machines, achieving the same application deployment and execution environment. This process is being handled by the orchestration layer and is transparent for the user, resulting in the application being delivered to the user, regardless of the using Docker containers or VMs.

Once the TOSCA document is built upon the user requirements the system starts with the deployment of the application. In this TOSCA document, the user application is referenced, so that the INDIGO-DataCloud Orchestrator [32] can select the most suitable site for executing it. After the site is selected, the orchestrator can apply two different procedures for deploying it, based on the image availability at the selected site:

1. Whenever the requested image is registered at the cloud site, the configuration step is removed, so the user application is spawned right away without the need of any image contextualization.
2. For those cases where the pre-configured image is not at the local catalogue of the cloud site, the deployment of the application is performed on a vanilla virtual machine or Docker container using Ansible roles. The execution of the Ansible Role on the provisioned computing resource is performed by either the IM, on an OpenNebula site, or Heat, on an OpenStack site. Therefore the application deployment is automatically done, without any user intervention. The Ansible role deals with the installation and configuration of a given application, so every application supported in the project need to have its corresponding role online available before its actual instantiation.

A user application made available through the second approach will notably take more time to be deployed, when compared with the pre-configured image instantiation already described. However using Ansible roles at this stage allows a more flexible customization since they can be designed to support application installation on a wide set of platforms and operating system distributions. Therefore, they are not being constrained to a specific OS distribution, as it is the case of using pre-configured images.

The availability of the Ansible role for each supported application is taken for granted within the workflow, since the pre-configured images are also created from them in order to install the application in a Docker image that will be made available in Docker Hub [31]. Having a single, unified approach to describe the application installation and configuration steps, promotes re-usability and simplifies maintenance.

Applications being integrated in INDIGO-DataCloud require: i) an Ansible Role that performs the automated installation of the application together with its dependences on a specific Operating System flavour (or a subset of them); ii) an entry in Ansible Galaxy to easily install the Ansible Role; iii) a new TOSCA node type that defines the requirements for the application; iv) a TOSCA template that references the new node type and optionally specifies an existing Docker image in Docker Hub with the application already installed inside. This Docker image will be automatically registered in a Cloud site supporting a Docker-enabled CMP by means of the INDIGO RepoSync tool [33]. Notice that this process is just required once. The user would later just use the same TOSCA template to automatically provision instances of the application on-demand.

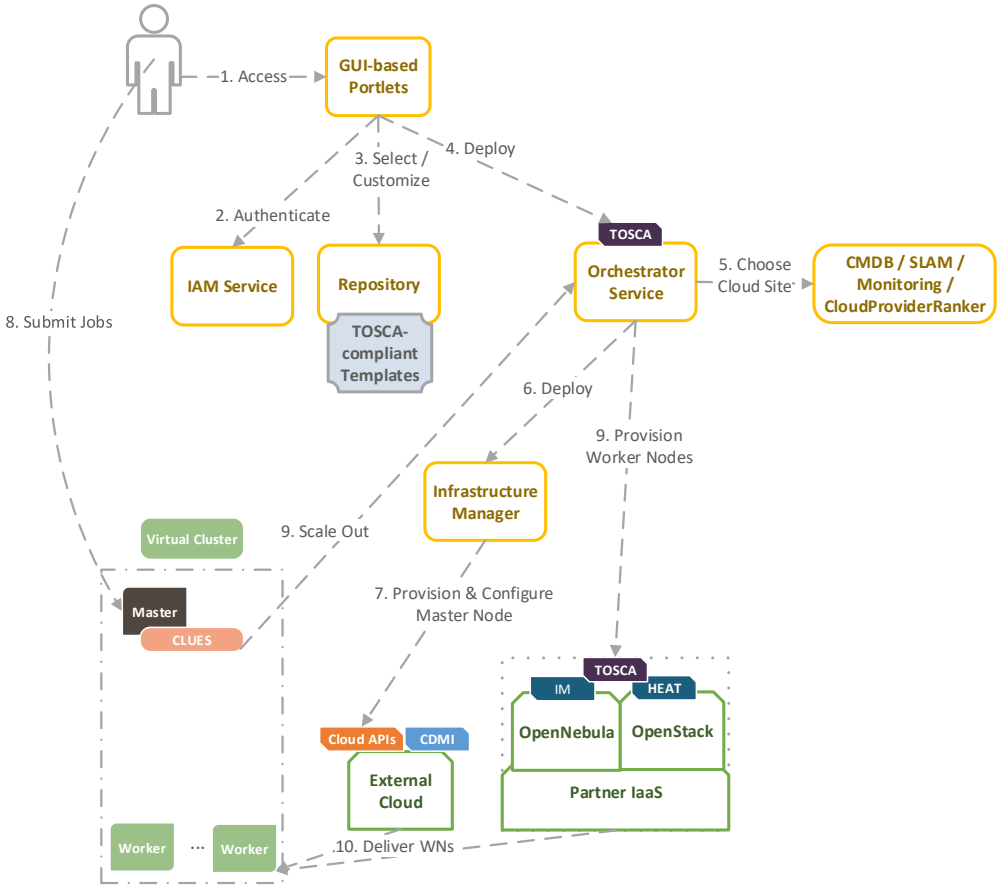

Fig. 5 Simplified architecture of the usage of TOSCA for the deployment of computing clusters in INDIGO-DataCloud

Figure 5 describes a simplification of the INDIGO-DataCloud architecture that enables the deployment of complex application layouts. In particular, the figure describes the workflow that involves both the TOSCA-based provision of the computing resources and their dynamic management for the specific case of a virtual elastic cluster.

The TOSCA document is obtained out of a TOSCA template, explicitly created for each particular application, filled with some runtime parameters. These parameters are provided, through the interaction with a high-level graphical user interface (GUI) (step 1). The GUI first asks for the user credentials by means of an identity access management service (step 2), then prompts for the selection of the TOSCA template (step 3). For the sake of example, we assume that the user wants to deploy a virtual elastic cluster.

The INDIGO-DataCloud orchestrator is the entry point to the PaaS layer, receiving TOSCA documents as input (step 4), to find the best match for the resource provisioning. The decision-making process is based on a SLA (Service Level Agreement) analysis and the assessment of the potential target providers availability (step 5), leveraging the INDIGO-DataCloud PaaS service stack [76,52]. One important feature of the orchestration system is that it supports hybrid deployments across different public or on-premises Cloud providers, making use of the VPN technology to establish secure and seamless connections among the compute nodes located in the different infrastructures (see an example TOSCA template for a hybrid deployment in [35]). The hybrid capabilities are particularly interesting for use cases involving virtual elastic clusters, allowing them to potentially scale out to access more servers than a single infrastructure could provide.

The CMP type of the selected cloud resource provider marks the remaining steps in the TOSCA orchestration workflow. Whenever an OpenStack cloud provider is selected, the orchestrator performs the interaction with the Heat service, preceded by the TOSCA template translation by means of the Heat Translator API. As described in Section 3, Heat Translator makes use of TOSCA Parser utility to load the TOSCA template,

ending up with a OpenStack's native HOT description of the stack to be deployed.

On the other hand, if the target cloud provider is based on OpenNebula framework or external public cloud platforms (such as Amazon AWS or Google Cloud), then the Orchestrator delegates on the IM to perform the provision and the configuration of the virtual infrastructure (step 6). The IM component will act as the unified TOSCA orchestrator, receiving the TOSCA template and acting as the orchestration layer with TOSCA support on top of the CMP.

The last steps in the workflow are related to the concrete example of deploying the virtual cluster. The orchestration layer —IM or OpenStack Heat— performs the automated deployment of the different tools to configure the virtual cluster along with CLUES [2] as the elasticity manager of the cluster (step 7). The user is then provided with the endpoints and credentials to access his virtual cluster. Once the user starts submitting jobs (step 8), CLUES automatically detects that additional resources are required, contacting the Orchestrator (step 9). It will then restart the provisioning process of step 5, resulting in new nodes dynamically added to the existing virtual cluster (step 10).

Figure 6 describes the relation among the Ansible roles and Docker images, as described in the TOSCA template [34]. The source code of the user applications are available in GitHub together with the Ansible roles that describe their installation and configuration process. Profiting from the GitHub and Docker Hub tight integration, automated builds of each application image are triggered once a new change is committed to its repository's default branch, thus making the last version of the application online available in Docker Hub registry. The Ansible roles are also centrally registered in the Ansible Galaxy online catalog. These Ansible roles are then referenced in the corresponding TOSCA types and used in the TOSCA templates so that applications can be automatically deployed on the provisioned virtual infrastructure.

5 Use Cases

This section illustrates two different use cases that have been implemented within INDIGO-DataCloud: A single node based application (Powerfit) and an elastic Mesos cluster. Both examples use new TOSCA types added by the INDIGO-DataCloud project to extend TOSCA Simple Profile in YAML Version 1.0.

5.1 Powerfit

The Powerfit [85] use case provides an example of a single-node application workflow. The reduced version of the TOSCA template required to deploy the application is shown in Listing 3 (the full example is available at *tosca-types* GitHub repository [37]).

In this use case the user selects a node of the type `tosca.nodes.indigo.Powerfit`, defined as shown in Listing 4. The definition includes the Ansible role needed to install and configure the application [29]. This is the same role used to build the Docker image specified in the template, i.e., `indigodatacloudapps/powerfit`). The deployment of this application follows the steps shown in section 4, i.e. using the pre-installed Docker image whenever it is locally available, or otherwise using a vanilla VM or Docker container.

Listing 3 A modified excerpt of the Powerfit TOSCA template.

```
tosca_definitions_version:
    tosca_simple_yaml_1_0

imports:
  - indigo_types: indigo-dc/tosca-types/master/
     custom_types.yaml

topology_template:

  node_templates:

    powerfit:
      type: tosca.nodes.indigo.Powerfit
      requirements:
        - host: p_server

    p_server:
      type: tosca.nodes.indigo.Compute
      capabilities:
        ...
        os:
          properties:
            type: linux
            distribution: ubuntu
            version: 14.04
            image: indigodatacloudapps/powerfit

    ...
```

Listing 4 tosca.nodes.indigo.Powerfit node type definition.

```
tosca.nodes.indigo.HaddockApp:
derived_from: tosca.nodes.SoftwareComponent
properties:
  haddock_app_name:
    type: string
    description: Haddocking application
    required: true
    constraints:
    - valid_values: [ disvis, powerfit ]
artifacts:
```

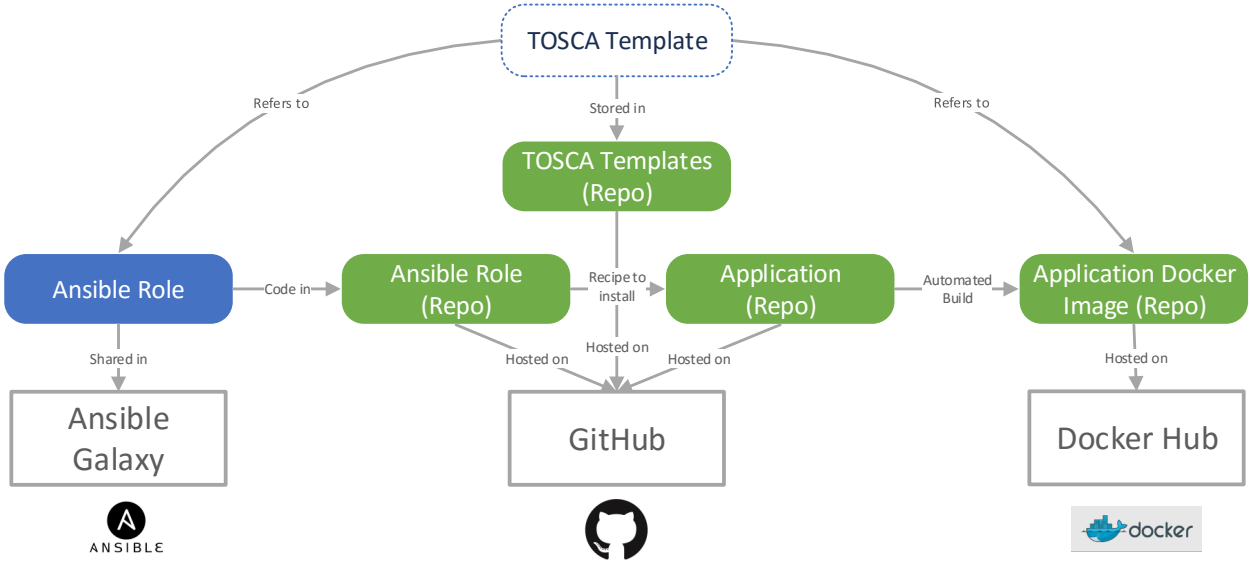

Fig. 6 Relation among Ansible Roles, Docker images and TOSCA templates.

```
  galaxy_role:
    file: indigo-dc.disvis-powerfit
    type: tosca.artifacts.AnsibleGalaxy.role
interfaces:
  Standard:
    configure:
      implementation: https://raw.
          githubusercontent.com/indigo-dc/
          tosca-types/master/artifacts/haddock
          /haddock_install.yml
      inputs:
        haddock_app_name: {get_property:[SELF,
            haddock_app_name]}

tosca.nodes.indigo.Powerfit:
derived_from: tosca.nodes.indigo.HaddockApp
properties:
  haddock_app_name:
    type: string
    required: true
    default: powerfit
    constraints:
      - equal: powerfit
```

This example demonstrates the capability to extend TOSCA with additional non-normative types that best match the application requirements. These new types also link to the appropriate automated deployment procedures (i.e. Ansible roles), previously tested to guarantee the success under different environments. Having custom types for different applications simplifies the TOSCA templates, enhancing their readability while preventing users from introducing changes that could affect the deterministic behaviour of the application deployment.

5.2 Elastic Mesos Cluster

Using virtual Apache Mesos clusters [5] in the cloud enables scientific communities to get access to customized cluster-based computing on-demand, to address both the execution of batch jobs and long-running services via Chronos [12] and the Marathon [48] framework, respectively. The project made available TOSCA templates to support the deployment of different types of customized virtual elastic computing clusters.

These clusters are elastic since, initially, only the front-end nodes are deployed. Moreover, unlike traditional computing clusters based on more traditional Local Resource Management Systems (LRMS) —like SGE, Torque or HTCondor—, Mesos provides with a high-availability mode that features multiple masters and load balancers.

The Mesos masters are customized with the required scientific applications, specific for a given user community. They are also configured with CLUES support, the elasticity management system for clusters introduced in Section 4. CLUES monitors the state of the job queue to detect when additional Mesos slaves are required to be deployed in order to cope with the number of pending jobs. The cluster is then dynamically adapted to a given workload, constrained by the maximum number of slaves specified in the TOSCA document as *"max_instances"*. CLUES was extended within the INDIGO-DataCloud project to provision additional nodes from the PaaS Orchestrator, as well as to intro-

duce elasticity for Apache Mesos Clusters and HTCondor batch resources.

An example of a TOSCA-based description for these virtual elastic computing clusters is available in the *tosca-types* GitHub repository [36], and summarized in Listing 5. The TOSCA template provides a description of the elastic cluster in terms of the computing requirements for all the Mesos cluster nodes (master, load balancer and slave nodes). It also specifies the maximum number of slaves to launch (5 in this example). For the sake of simplicity, some information has been omitted in the TOSCA template depicted.

Listing 5 A modified excerpt of the virtual elastic Mesos cluster TOSCA template.

```
tosca_definitions_version:
    tosca_simple_yaml_1_0

imports:
  - indigo_types: indigo-dc/tosca-types/master/
     custom_types.yaml

topology_template:

  node_templates:
    elastic_cluster_front_end:
      type: tosca.nodes.indigo.ElasticCluster
      requirements:
        - lrms: mesos_master
        - wn: mesos_slave

    mesos_master:
      type: tosca.nodes.indigo.LRMS.FrontEnd.
          Mesos
      properties:
        marathon_password: marathon_password
        chronos_password: chronos_password
      requirements:
        - host: master_server

    mesos_slave:
      type: tosca.nodes.indigo.LRMS.WorkerNode.
          Mesos
      capabilities:
        wn:
          properties:
            max_instances: 5
            min_instances: 0
      properties:
        master_ips: {get_attribute:[
            master_server,public_address]}
      requirements:
        - host: mesos_slave_server

    mesos_load_balancer:
      type: tosca.nodes.indigo.
          MesosLoadBalancer
      properties:
        master_ips: {get_attribute:[
            master_server,public_address]}
      requirements:
        - host: lb_server

    master_server:
      type: tosca.nodes.indigo.Compute
      capabilities:
        endpoint:
          properties:
            dns_name: mesosserverpublic
            network_name: PUBLIC
        scalable:
          properties:
            count: 1

    mesos_slave_server:
      type: tosca.nodes.indigo.Compute

    lb_server:
      type: tosca.nodes.indigo.Compute
      capabilities:
        endpoint:
          properties:
            network_name: PUBLIC
        scalable:
          properties:
            count: 1

  outputs:
    mesos_lb_ips:
      value: { get_attribute: [ lb_server,
          public_address ] }
    mesos_master_ips:
      value: { get_attribute: [ master_server,
          public_address ] }
```

6 Conclusions and future work

This paper has described the challenges of orchestrating computing resources in heterogeneous clouds and the approach carried out in the INDIGO-DataCloud project to overcome them, with discussion of real life use cases from scientific communities.

The TOSCA open standard has been adopted for the description of the application layouts. Different examples have been shown ranging from a simple single-node application to an elastic Apache Mesos cluster. For this, an orchestration approach based on prioritizing cloud sites with existing pre-configured Docker images is employed, while being able to dynamically deploy the applications on cloud sites supporting only vanilla VMs or Docker images. By adopting a configuration management solution based on Ansible roles to carry out both the deployment of the application and the creation of the pre-configured Docker images, a single consistent unified approach for application delivery is employed.

By using TOSCA to model the user's complex application architectures it is possible to obtain repeatable and deterministic deployments. User's can port their virtual infrastructures between providers transparenty obtaining the same expected topology.

The time required to deploy a virtual infrastructure is strictly dominated by the time required to provision the underlying computational resources and the time to configure them. Therefore, provisioning from a Cloud site that already supports the pre-configured Docker images requested by the user is considerably much faster than having to boot up the Virtual Machines from another Cloud site and perform the whole application installation and its dependencies. The overhead introduced by the PaaS layer is negligible compared to the time to deploy an infrastructure, since it just requires a reduced subset of invocations among the different microservices.

As it can be seen from the use cases described in Section 5, several non-normative TOSCA node types were introduced in the context of INDIGO-DataCloud project to support both different user applications and specific services to be used within the deployed applications. Indeed, the extensibility of the TOSCA language and the ability of the underlying TOSCA parser to process these new elements facilitates the procedure of adopting TOSCA as the definition language to perform the orchestration of complex infrastructures across multiple Clouds.

It is important to point out that a key contribution of the INDIGO-DataCloud Orchestration system, with respect to other orchestration platforms, is the implementation of hybrid orchestration to satisfy demands of dynamic or highly changeable workloads, such as the virtual elastic cluster use case presented.

The approach described here is being used by several user communities that have been engaged within the project [14,72,20,68,1,19,11,53,42]. The developed solutions have also resulted in community and upstream code contributions to major open source solutions like OpenStack and OpenNebula.

Future work includes supporting different complex application architectures used by scientific communities. For example, this work will include architectures for Big Data processing in order to automatically provision virtual computing clusters to process large volumes of data using existing frameworks such as Hadoop and Spark). Regarding the tools that have been described, the OpenStack Heat Translator is expected to evolve into a service that could be deployed along with the OpenStack Orchestration (Heat) service, enabling the direct submission of TOSCA documents to the endpoint that this service will provide.

Acknowledgements The authors want to acknowledge the support of the INDIGO-Datacloud (grant number 653549) project, funded by the European Commission's Horizon 2020 Framework Programme.

References

1. Aguilar Gómez, F., de Lucas, J.M., García, D., Monteoliva, A.: Hydrodynamics and water quality forecasting over a cloud computing environment: Indigo-datacloud. In: EGU General Assembly Conference Abstracts, vol. 19, p. 9684 (2017)
2. de Alfonso, C., Caballer, M., Alvarruiz, F., Hernández, V.: An energy management system for cluster infrastructures. Computers & Electrical Engineering **39**(8), 2579–2590 (2013). URL http://www.sciencedirect.com/science/article/pii/S0045790613001365
3. Amazon Web Services (AWS): Amazon Web Services (AWS) (2017). URL https://aws.amazon.com/
4. Amazon Web Services (AWS): CloudFormation (2017). URL https://aws.amazon.com/cloudformation/
5. Apache Software Foundation: Apache Mesos (2017). URL http://mesos.apache.org/
6. ARIA: ARIA (2017). URL http://ariatosca.incubator.apache.org/
7. Bumpus, W.: NIST Cloud Computing Standards Roadmap. Tech. rep., National Institute of Standards and Technology (NIST) (2013). DOI 10.6028/NIST.SP.500-291r2
8. Caballer, M., Blanquer, I., Moltó, G., de Alfonso, C.: Dynamic Management of Virtual Infrastructures. Journal of Grid Computing **13**(1), 53–70 (2015). DOI 10.1007/s10723-014-9296-5
9. Campos Plasencia, I., Fernández-del Castillo, E., Heinemeyer, S., López García, Á., Pahlen, F., Borges, G.: Phenomenology tools on cloud infrastructures using OpenStack. The European Physical Journal C **73**(4), 2375 (2013). DOI 10.1140/epjc/s10052-013-2375-0. URL http://link.springer.com/10.1140/epjc/s10052-013-2375-0
10. Celar: Celar (2017). URL http://www.cloudwatchhub.eu/celar
11. Chen, Y., de Lucas, J.M., Aguilar, F., Fiore, S., Rossi, M., Ferrari, T.: Indigo: Building a datacloud framework to support open science. In: EGU General Assembly Conference Abstracts, vol. 18, p. 16610 (2016)
12. Chronos: Chronos (2017). URL https://mesos.github.io/chronos/
13. Cloudify: Cloudify (2017). URL http://getcloudify.org
14. Davidović, D., Cetinić, E., Skala, K.: European research area and digital humanities
15. Distefano, S., Serazzi, G.: Performance Driven WS Orchestration and Deployment in Service Oriented Infrastructure. Journal of Grid Computing **12**(2), 347–369 (2014). DOI 10.1007/s10723-014-9293-8
16. EGI FedCloud: EGI FedCloud (2017). URL https://www.egi.eu/federation/egi-federated-cloud/
17. Eucalyptus: Eucalyptus (2017). URL https://www.eucalyptus.com/
18. Fiore, S., D'Anca, A., Palazzo, C., Foster, I., Williams, D.N., Aloisio, G.: Ophidia: Toward big data analytics for eScience. Procedia Computer Science **18**, 2376–2385 (2013). DOI 10.1016/j.procs.2013.05.409. URL http://dx.doi.org/10.1016/j.procs.2013.05.409
19. Fiore, S., Palazzo, C., D'Anca, A., Elia, D., Londero, E., Knapic, C., Monna, S., Marcucci, N.M., Aguilar, F., Płóciennik, M., et al.: Big data analytics on large-scale scientific datasets in the indigo-datacloud project. In: Proceedings of the Computing Frontiers Conference, pp. 343–348. ACM (2017)

20. Fiore, S., Płóciennik, M., Doutriaux, C., Palazzo, C., Boutte, J., Żok, T., Elia, D., Owsiak, M., D'Anca, A., Shaheen, Z., et al.: Distributed and cloud-based multi-model analytics experiments on large volumes of climate change data in the earth system grid federation eco-system. In: Big Data (Big Data), 2016 IEEE International Conference on, pp. 2911–2918. IEEE (2016)
21. Galante, G., Erpen de Bona, L.C., Mury, A.R., Schulze, B., da Rosa Righi, R.: An Analysis of Public Clouds Elasticity in the Execution of Scientific Applications: a Survey. Journal of Grid Computing pp. 1–24 (2016). DOI 10.1007/s10723-016-9361-3. URL http://dx.doi.org/10.1007/s10723-016-9361-3
22. Garcia, A.L., Zangrando, L., Sgaravatto, M., Llorens, V., Vallero, S., Zaccolo, V., Bagnasco, S., Taneja, S., Pra, S.D., Salomoni, D., Donvito, G.: Improved Cloud resource allocation: how INDIGO-DataCloud is overcoming the current limitations in Cloud schedulers (2017). URL http://arxiv.org/abs/1707.06403
23. Google Cloud Platform (GCP): Google Cloud Platform (GCP) (2017). URL https://cloud.google.com/
24. Hochstein, L. (ed.): Ansible: Up and Running, Automating Configuration Management and Deployment the Easy Way. O'Reilly Media (2014)
25. Idabc: European Interoperability Framework for pan-European eGovernment Services. European Commission **version 1**, 1–25 (2004). DOI 10.1109/HICSS.2007.68
26. IM: IM (2017). URL http://www.grycap.upv.es/im
27. INDIGO-DataCloud: D1.8 - General Architecture. Tech. rep., INDIGO-DataCloud Consortium (2015)
28. INDIGO-DataCloud: Ansible Galaxy repository for INDIGO-DataCloud (2017). URL https://galaxy.ansible.com/indigo-dc/
29. INDIGO-DataCloud: Disvis/Powerfit Ansible Role in Ansible Galaxy (2017). URL https://galaxy.ansible.com/indigo-dc/disvis-powerfit/
30. INDIGO-DataCloud: INDIGO-DataCloud (2017). URL https://www.indigo-datacloud.eu/
31. INDIGO-DataCloud: INDIGO-DataCloud DockerHub application repository (2017). URL https://hub.docker.com/u/indigodatacloudapps/
32. INDIGO-DataCloud: INDIGO-DataCloud PaaS Orchestrator (2017). URL https://github.com/indigo-dc/orchestrator
33. INDIGO-DataCloud: INDIGO-DataCloud RepoSync (2017). URL https://github.com/indigo-dc/java-reposync
34. INDIGO-DataCloud: INDIGO-DataCloud TOSCA templates (2017). URL https://github.com/indigo-dc/tosca-templates
35. INDIGO-DataCloud: TOSCA Across Clouds (2017). URL https://github.com/indigo-dc/tosca-types/blob/master/examples/web_mysql_tosca_across_clouds.yaml
36. INDIGO-DataCloud: TOSCA template for deploying an Elastic Mesos Cluster (2017). URL http://github.com/indigo-dc/tosca-types/blob/master/examples/mesos_elastic_cluster.yaml
37. INDIGO-DataCloud: TOSCA template for Powerfit application (2017). URL https://github.com/indigo-dc/tosca-types/blob/master/examples/powerfit.yaml
38. Kacsuk, P., Kecskemeti, G., Kertesz, A., Nemeth, Z., Kovács, J., Farkas, Z.: Infrastructure Aware Scientific Workflows and Infrastructure Aware Workflow Managers in Science Gateways. Journal of Grid Computing pp. 641–654 (2016). DOI 10.1007/s10723-016-9380-0. URL http://link.springer.com/10.1007/s10723-016-9380-0
39. Korambath, P., Wang, J., Kumar, A., Hochstein, L., Schott, B., Graybill, R., Baldea, M., Davis, J.: Deploying kepler workflows as services on a cloud infrastructure for smart manufacturing. Procedia Computer Science **29**, 2254–2259 (2014)
40. Koski, K., Hormia-Poutanen, K., Chatzopoulos, M., Legré, Y., Day, B.: Position Paper: European Open Science Cloud for Research. Tech. Rep. October, EUDAT, LIBER, OpenAIRE, EGI, GÉANT, Bari (2015)
41. Krieger, M.T., Torreno, O., Trelles, O., Kranzlm??ller, D.: Building an open source cloud environment with auto-scaling resources for executing bioinformatics and biomedical workflows. Future Generation Computer Systems **67**, 329–340 (2017). DOI 10.1016/j.future.2016.02.008
42. Kurkcuoglu Soner, Z., Bonvin, A.: Science in the clouds: Virtualizing haddock, powerfit and disvis using indigo-datacloud solutions (2016)
43. Lipton, P.C.T., Moser, S.I., Palma, D.V., Spatzier, T.I.: Topology and Orchestration Specification for Cloud Applications. Tech. rep., OASIS Standard (2013)
44. Liu, C., Mao, Y., Van der Merwe, J., Fernandez, M.: Cloud resource orchestration: A data-centric approach. In: Proceedings of the biennial Conference on Innovative Data Systems Research (CIDR), pp. 1–8. Citeseer (2011)
45. López García, Á., Fernández-del Castillo, E.: Analysis of Scientific Cloud Computing requirements. In: Proceedings of the IBERGRID 2013 Conference, p. 147 158 (2013)
46. López García, Á., Fernández-del Castillo, E., Orviz Fernández, P.: Standards for enabling heterogeneous IaaS cloud federations. Computer Standards & Interfaces **47**, 19–23 (2016). DOI 10.1016/j.csi.2016.02.002. URL http://dx.doi.org/10.1016/j.csi.2016.02.002http://www.sciencedirect.com/science/article/pii/S0920548916300022
47. Lorido-Botran, T., Miguel-Alonso, J., Lozano, J.A.: A Review of Auto-scaling Techniques for Elastic Applications in Cloud Environments. Journal of Grid Computing **12**(4), 559–592 (2014). DOI 10.1007/s10723-014-9314-7
48. Marathon: Marathon (2017). URL https://mesosphere.github.io/marathon/
49. Metsch, T., Edmonds, A.: Open cloud computing interface-infrastructure. Tech. rep., Open Grid Forum (2010)
50. Metsch, T., Edmonds, A.: Open cloud computing interface-RESTful HTTP rendering. Tech. rep., Open Grid Forum (2011)
51. Microsoft Azure: Microsoft Azure (2017). URL https://azure.microsoft.com/
52. Molt, G., Caballer, M., Prez, A., d. Alfonso, C., Blanquer, I.: Coherent application delivery on hybrid distributed computing infrastructures of virtual machines and docker containers. In: 2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP), pp. 486–490 (2017). DOI 10.1109/PDP.2017.29
53. Monna, S., Marcucci, N.M., Marinaro, G., Fiore, S., D'Anca, A., Antonacci, M., Beranzoli, L., Favali, P.: An emso data case study within the indigo-dc project. In: EGU General Assembly Conference Abstracts, vol. 19, p. 12493 (2017)
54. Nyrén, R., Metsch, T., Edmonds, A., Papaspyrou, A.: Open cloud computing interfacecore. Tech. rep., Open Grid Forum (2010)

55. OASIS: Organization for the Advancement of Structured Information Standards (OASIS) (2015). URL https://www.oasis-open.org
56. Open Telekom Cloud (OTC): Open Telekom Cloud (OTC) (2017). URL https://cloud.telekom.de/en/
57. OpenNebula: OneFlow (2017). URL http://docs.opennebula.org/5.2/advanced_components/application_flow_and_auto-scaling/index.html
58. OpenNebula Project: OpenNebula (2017). URL https://www.opennebula.org
59. OpenStack Foundation: Heat Orchestration Template (HOT) Guide (2017). URL https://docs.openstack.org/heat/latest/template_guide/hot_guide.html
60. OpenStack Foundation: OpenStack (2017). URL https://www.openstack.org
61. OpenStack Foundation: Openstack Heat (2017). URL http://wiki.openstack.org/wiki/Heat
62. OpenStack Foundation: OpenStack Heat Translator (2017). URL https://github.com/openstack/heat-translator
63. OpenStack Foundation: OpenStack heat-translator project contribution statistics (2017). URL http://stackalytics.com/?release=all&metric=commits&module=heat-translator
64. OpenStack Foundation: OpenStack Tacker (2017). URL https://wiki.openstack.org/wiki/Tacker
65. OpenStack Foundation: OpenStack tosca-parser project contribution statistics (2017). URL http://stackalytics.com/?release=all&metric=commits&module=tosca-parser
66. OpenStack Foundation: TOSCA Parser (2017). URL https://github.com/openstack/tosca-parser
67. OpenTOSCA: OpenTOSCA (2017). URL http://www.opentosca.org/
68. Owsiak, M., Plociennik, M., Palak, B., Zok, T., Reux, C., Di Gallo, L., Kalupin, D., Johnson, T., Schneider, M.: Running simultaneous kepler sessions for the parallelization of parametric scans and optimization studies applied to complex workflows. Journal of Computational Science **20**, 103–111 (2017)
69. Palma, D., Rutkowski, M., Spatzier, T.: TOSCA Simple Profile in YAML Version 1.1. Tech. rep., OASIS Standard (2016). URL http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.1/TOSCA-Simple-Profile-YAML-v1.1.html
70. Petcu, D.: Consuming Resources and Services from Multiple Clouds: From Terminology to Cloudware Support. Journal of Grid Computing **12**(2), 321–345 (2014). DOI 10.1007/s10723-013-9290-3
71. Plóciennik, M., Fiore, S., Donvito, G., Owsiak, M., Fargetta, M., Barbera, R., Bruno, R., Giorgio, E., Williams, D.N., Aloisio, G.: Two-level dynamic workflow orchestration in the INDIGO DataCloud for large-scale, climate change data analytics experiments. Procedia Computer Science **80**, 722–733 (2016). DOI 10.1016/j.procs.2016.05.359
72. Płóciennik, M., Fiore, S., Donvito, G., Owsiak, M., Fargetta, M., Barbera, R., Bruno, R., Giorgio, E., Williams, D.N., Aloisio, G.: Two-level dynamic workflow orchestration in the indigo datacloud for large-scale, climate change data analytics experiments. Procedia Computer Science **80**, 722–733 (2016)
73. Python: Python Package Index (PyPI) (2017). URL https://pypi.python.org/pypi
74. Ramakrishnan, L., Jackson, K.R., Canon, S., Cholia, S., Shalf, J.: Defining future platform requirements for e-Science clouds. Proceedings of the 1st ACM symposium on Cloud computing - SoCC '10 p. 101 (2010). DOI 10.1145/1807128.1807145. URL http://portal.acm.org/citation.cfm?doid=1807128.1807145
75. Ramakrishnan, L., Zbiegel, P.T.T.T.: Magellan: experiences from a science cloud. In: Proceedings of the 2nd international workshop on Scientific cloud computing, pp. 49–58 (2011). URL http://dl.acm.org/citation.cfm?id=1996119
76. Salomoni, D., Campos, I., Gaido, L., Donvito, G., Antonacci, M., Fuhrman, P., Marco, J., Lopez-Garcia, A., Orviz, P., Blanquer, I., et al.: Indigo-datacloud: foundations and architectural description of a platform as a service oriented to scientific computing. arXiv preprint arXiv:1603.09536 (2016)
77. Sánchez-Expósito, S., Martín, P., Ruiz, J.E., Verdes-Montenegro, L., Garrido, J., Sirvent, R., Falcó, A.R., Badia, R.M., Lezzi, D.: Web Services as Building Blocks for Science Gateways in Astrophysics. Journal of Grid Computing **14**(4), 673–685 (2016). DOI 10.1007/s10723-016-9382-y. URL http://dx.doi.org/10.1007/s10723-016-9382-y
78. SlipStream: SlipStream (2017). URL http://sixsq.com/products/slipstream/
79. Stockton, D.B., Santamaria, F.: Automating NEURON Simulation Deployment in Cloud Resources. Neuroinformatics **15**(1), 51–70 (2017). DOI 10.1007/s12021-016-9315-8. URL http://dx.doi.org/10.1007/s12021-016-9315-8
80. Teckelmann, R., Reich, C., Sulistio, A.: Mapping of cloud standards to the taxonomy of interoperability in iaas. In: Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on, pp. 522–526. IEEE (2011)
81. Toor, S., Osmani, L., Eerola, P., Kraemer, O., Lindén, T., Tarkoma, S., White, J.: A scalable infrastructure for CMS data analysis based on OpenStack Cloud and Gluster file system. Journal of Physics: Conference Series **513**(6), 062,047 (2014). DOI 10.1088/1742-6596/513/6/062047. URL http://stacks.iop.org/1742-6596/513/i=6/a=062047?key=crossref.84033a04265ce343371c7f38064e7143
82. UK Government Cabinet Office: Open Standards Principles (2015). URL https://www.gov.uk/government/publications/open-standards-principles/open-standards-principles
83. Yangui, S., Marshall, I.J., Laisne, J.P., Tata, S.: Compatibleone: The open source cloud broker. Journal of Grid Computing **12**(1), 93–109 (2014)
84. Zhao, Y., Li, Y., Raicu, I., Lu, S., Tian, W., Liu, H.: Enabling scalable scientific workflow management in the Cloud. Future Generation Computer Systems **46**(November), 3–16 (2015). DOI 10.1016/j.future.2014.10.023
85. van Zundert, G., Trellet, M., Schaarschmidt, J., Kurkcuoglu, Z., David, M., Verlato, M., Rosato, A., Bonvin, A.: The DisVis and PowerFit Web Servers: Explorative and Integrative Modeling of Biomolecular Complexes. Journal of Molecular Biology **429**(3), 399–407 (2013). URL http://www.sciencedirect.com/science/article/pii/S0022283616305277