TOSCA-based orchestration of complex clusters at the IaaS level

IOP Conf. Series: Journal of Physics: Conf. Series **898** (2017) 082036 doi:10.1088/1742-6596/898/8/082036

TOSCA-based orchestration of complex clusters at the IaaS level

M Caballer[1], G Donvito[2], G Moltó[1], R Rocha[3] and M Velten[3]

1 Instituto de Instrumentación para Imagen Molecular (I3M). Centro mixto CSIC Universitat Politècnica de València - CIEMAT, camino de Vera s/n, 46022 Valencia, Spain
2 Istituto Nazionale di Fisica Nucleare (INFN), Via Giovanni Amendola, 173, 70126 Bari, Italy
3 European Organization for Nuclear Research (CERN), CH-1211 Geneva, Switzerland

Abstract. This paper describes the adoption and extension of the TOSCA standard by the INDIGO-DataCloud project for the definition and deployment of complex computing clusters together with the required support in both OpenStack and OpenNebula, carried out in close collaboration with industry partners such as IBM. Two examples of these clusters are described in this paper, the definition of an elastic computing cluster to support the Galaxy bioinformatics application where the nodes are dynamically added and removed from the cluster to adapt to the workload, and the definition of an scalable Apache Mesos cluster for the execution of batch jobs and support for long-running services. The coupling of TOSCA with Ansible Roles to perform automated installation has resulted in the definition of high-level, deterministic templates to provision complex computing clusters across different Cloud sites.

1. Introduction

INDIGO-DataCloud is an European Union's Horizon 2020 funded project whose ultimate goal is to provide a sustainable European software infrastructure for science, spanning multiple computer centers and existing public clouds. The participating sites form a set of heterogeneous cloud infrastructures with different Cloud Management Platforms (CMP), some running OpenNebula, some running OpenStack. INDIGO-DataCloud is introducing innovative advancements at the layer of IaaS (Infrastructure as a Service), e.g., by introducing and supporting containers in the aforementioned CMPs, at the layer of PaaS (Platform as a Service), e.g., by creating SLA-based orchestration components that support deployments on multi-Clouds and, finally, at the layer of SaaS (Software as a Service), e.g., by developing high-level REST and graphical user interfaces to facilitate the usage of computing infrastructures for different scientific communities. Moreover from the point of view of the storage INDIGO-DataCloud pushes forward and greatly simplify the management of heterogeneous storage resources in the cloud environment.

There was the need to find a common denominator for the deployment of both the required PaaS services and the end user application architecture, which typically involve customized virtual infrastructures. In this context TOSCA represents a standard approach to provide descriptions of applications architectures to be deployed on a cloud and was adopted and extended by INDIGO-DataCloud to support the requirements coming from scientific communities.

In the context of this work, we describe how TOSCA is employed to describe complex clusters of applications and services and to provide a way to express their automatic configuration via Ansible recipes so that they can deployed at an IaaS cloud site. In particular, we focus on two different examples of complex clusters: i) customized elastic virtual computing clusters where the number of nodes can dynamically grow and shrink according to the workload and ii) deployment of elastic Apache Mesos clusters supporting the Chronos framework, to perform batch execution of jobs and the Marathon framework, to manage the execution of long-running services, both of them encapsulated as Docker containers.

After the introduction, the remainder of the paper is structured as follows. First, section 2 provides a brief introduction to the TOSCA standard. Second, section 3 describes the motivation to adopt TOSCA as the standard to describe complex application architectures in INDIGO-DataCloud focusing on its current usage to orchestrate complex clusters. Third, section 4 describes different examples of defining complex clusters using TOSCA, both involving scientific applications employed by user communities as well as to perform support for the execution of batch jobs and the deployment of long-running services. Finally, section 5 summarizes the paper and points to future work.

2. A Brief Introduction to TOSCA

TOSCA (Topology and Orchestration Specification for Cloud Applications) is an OASIS specification for the interoperable description of application and infrastructure cloud services, the relationships between parts of these services, and their operational behaviour. Originally, TOSCA templates were written in XML, but the TOSCA Simple Profile in YAML[7] enables the user to write TOSCA documents in YAML that are less verbose and more human-readable. This greately simplifies the creation of TOSCA templates. The TOSCA Simple Profile specification defines a set of normative types as a base type system that provides high-level abstractions for most cloud service and infrastructure components. In addition, new non-normative node types can be included to fit some particular needs.

Recently the TOSCA TC has approved the TOSCA Simple Profile for Network Functions Virtualization (NFV) [10]. This enables the specification of a NFV specific data model using the TOSCA language to deploy and operate Network Services and Virtual Network Functions (VNFs) on an NFV infrastructure platform.

Furthermore, the TOSCA TCs Container and Clustering ad hoc workgroup has been empowered to expand its focus to include how to define clusters of homogeneous and heterogeneous containers and provide standard capabilities that allow containers to describe abstract Compute, Network, and Storage requirements, as well as provide overall load balancing and scaling of both individual containers and the clusters themselves.

To sum up, TOSCA provides the language to create a TOSCA template, which is a YAML document that describes the architecture of an application to be deployed on a cloud site.

3. TOSCA in INDIGO-DataCloud

At the start of the INDIGO-DataCloud project an effort was made to evaluate the available orchestration options. The following were considered:

- HOT[4], the language behind OpenStack HEAT
- CloudFormation[1], the language used by AWS (Amazon Web Services)
- TOSCA[7]

The first two had the advantage of a larger user base, with multiple tools relying on them to accomplish more complex tasks. Indeed, Heat endeavours to provide compatibility with the AWS CloudFormation template format. One big disadvantage was clear: they are both tied to specific implementations, and not easily reusable in an heterogeneous environment. TOSCA on

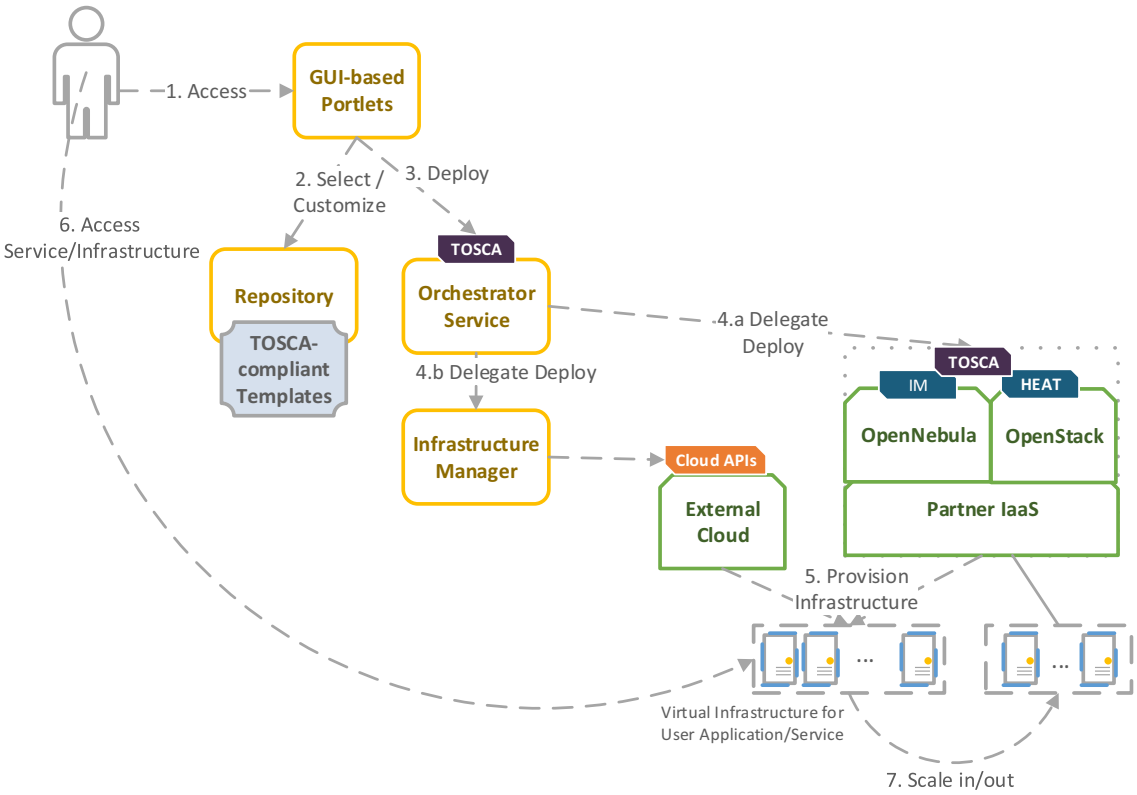

Figure 1. Simplified architecture of the usage of TOSCA for the deployment of computing clusters in INDIGO-DataCloud

the other hand is an open standard, with existing support from popular frameworks such as Cloudify[2] and OpenStack and benefits from a growing support in different communities. This is clear from the strong interest from the networking community regarding Network Functions Virtualization (NFV)[10] in a separate profile of the specification, as stated earlier.

The same evaluation also revealed two existing codebases that could be reused in our project: the TOSCA Parser[6] and the HEAT Translator[5]. Both are open sourced under the umbrella of the OpenStack project, but integration in other environments was easy with a couple of small prototypes created to prove this point. This filled in the requirement of the INDIGO-DataCloud to support multiple cloud providers.

The decision was thus to take TOSCA as the viable common denominator for the definition of both topologies and end user applications.

3.1. TOSCA Templates in INDIGO-DataCloud

A TOSCA Template is a text document written in YAML that describes an architecture of an application to be deployed on a Cloud site. In addition, in INDIGO-DataCloud, TOSCA templates are also employed to describe jobs to be executed via Chronos as well as long-running services executed via Marathon, both on an Apache Mesos cluster, though in this particular work we exclusively focus on the definition of complex clusters.

Figure 1 describes a simplification of the architecture of components employed to deploy complex clusters. For the sake of brevity, data management, authentication and autorization and other services required for orchestration (such as monitoring and SLA assessment, among

others) have been omitted from the diagram. The INDIGO-DataCloud PaaS Orchestrator accepts TOSCA templates as input. There are two main publicly available sources of TOSCA templates for INDIGO-DataCloud: i) the *tosca-templates* GitHub repository, which contains the authoritative templates employed to support different use cases and deploy required infrastructure used by the PaaS (e.g. a Mesos cluster). and ii) the *tosca-types/examples* directory of the corresponding GitHub repository, which contains additional TOSCA templates used for examples, training, testing as well as incubating TOSCA templates for uses cases before they are assessed.

The very same TOSCA template should be employed to: i) deploy an application by spawning an instance of a Docker container out of a Docker image available in Docker Hub on either a Cloud site managed via OneDock, in the case of OpenNebula, and nova-docker in the case of OpenStack, which both allow to natively deploy Docker containers (Docker images will have to be registered in the site), and, ii) deploy an application by spawning a vanilla VM or Docker container on a Cloud site and installing the application by means of the corresponding Ansible Role that defines how to install a particular application. An Ansible Role represents a recipe that describes the installation process of a particular application a specific platform or a set of them.

Having a single, unified approach to describe application installation enables to reuse this role in order to either deploy the application as part of the creation of a Docker image or to deploy the application on a running Docker container or Virtual Machine, thus simplifying maintenance.

3.2. TOSCA-based Elastic Computing Clusters

Computing clusters are a widely-known computing facility and virtual computing clusters in the Cloud have enabled scientific communities to access customized cluster-based computing on-demand. INDIGO-DataCloud supports the deployment of customized virtual elastic computing clusters defined in TOSCA templates and automatically provisioned by the PaaS Orchestrator on the available Cloud infrastructure.

These clusters are elastic since, initially, only the front-end node is deployed, which is customized with the required scientific applications, depending on the target user community. This front-end node is also configured with support for CLUES [8], an elasticity management system for clusters that supports several Local Resource Management Systems (LRMS) (e.g. SLURM, PBS/Torque). CLUES monitors the state of the job queue in order to detect when additional working nodes are required to be deployed in order to cope with the number of pending jobs. This way, the cluster can dynamically grow and shrink in terms of the number of working nodes to adapt to the workload (as visually depicted in step 7 in Figure 1). In INDIGO-DataCloud, CLUES was adapted to provision additional nodes from the PaaS Orchestrator, as well as to introduce elasticity for Apache Mesos Clusters and HTCondor batch resources.

The usage of TOSCA templates to describe complex computing clusters provides a deterministic, reproducible approach to provide cluster-based computing to a wide variety of scientific communities.

3.3. TOSCA Workflow in INDIGO-DataCloud

As described earlier, the INDIGO-DataCloud project is providing a set of TOSCA templates and Ansible Roles to describe useful stacks for scientific purpose. These templates should be deployed on any of the Cloud providers supported by the project. For the sake of the discussion, we differentiate between OpenStack sites and other Cloud sites.

For the first case, the Heat component of OpenStack is used in order to deploy a software stack and manage the lifetime of its various components (mainly VMs in our case but basically all OpenStack primitives are supported by Heat like volumes, networks, etc). Unfortunately Heat does not natively understand TOSCA templates. However, a translator is available under

the OpenStack umbrella to convert a template from the TOSCA language to HOT. We used this translator as a base and improved it upstream to properly support our templates. This work includes Ansible script support, Ansible Galaxy Role as TOSCA artifact, querying and matching the available flavors, images and networks, scalable server support among other features and fixes. Currently this translator is a client side implementation but there is work in progress to provide a pluggable component to the Heat server directly, using the same principles as used for the CloudFormation compatibility layer.

To contextualize a VM with Ansible and other scripts an agent is needed inside the used image to communicate with the Heat server. Contextualization can also directly happen through the cloud-init metadata information. However, this mechanism is way less flexible since we cannot communicate back any value from inside the VM to pass it to another component.

In case of deploying on OpenNebula sites or public Clouds, such as Amazon Web Services, the Infrastructure Manager (IM) [11] is used to perform the orchestration, deployment and configuration of the virtual infrastructures. There is one IM instance in each cloud provider of the INDIGO-DataCloud platform, that acts as the TOSCA runtime for the site, in a similar way as Heat works for OpenStack sites (see Figure 1 for details). This IM instance directly receives the TOSCA template and contacts the cloud site using their own native APIs to deploy and configure the virtual infrastructure. Once the resources have been deployed and they are running the IM selects one of them as the "master" node and installs Ansible [9] and configures it 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 will the rest of nodes of the infrastructure (either via a public or private IP). Once the node is configured the contextualization agent will configure all the nodes using the defined Ansible playbooks.

In an IaaS context both Heat and IM are very useful to ease portable provisioning of resources and deployment of services and applications on dynamically instantiated clusters. One of the advantages of using TOSCA and the IM/Heat approach is that the INDIGO PaaS layer can easily exploit it across different IaaS implementations, increasing the portability of the cluster definitions, and implementing the provisioning of the required services across multiple IaaS infrastructures through the INDIGO orchestrator.

A similar approach is used in case of accessing external providers. For example public clouds (Amazon Web Services, Google Cloud Platform, Microsoft Azure) or some federated infrastructures as EGI FedCloud sites [3]. In this case, a special instance of the IM is provided by the INDIGO PaaS core to access these providers.

4. Examples of TOSCA-based Complex Clusters in INDIGO-DataCloud

This section describes two representative definition examples of complex clusters. The first one involves the deployment if a virtual elastic cluster supporting the Galaxy bioinformatics application. The second one exemplifies the deployment of an elastic Apache Mesos cluster.

4.1. Galaxy-based Elastic Computing Cluster

An example of TOSCA-based description of these virtual elastic computing clusters is available in the *tosca-templates* GitHub repository, summarized in figure 2. The TOSCA template provides a description of the elastic cluster in terms of the computing requirements for both the front-end node and the working nodes of the cluster, although for the sake of brevity, this information is omitted in the TOSCA template depicted. There are non-normative TOSCA types that have been defined to support the requirements for INDIGO-DataCloud, such as *tosca.nodes.indigo.ElasticCluster* and *tosca.nodes.indigo.GalaxyPortal*. These node types are described in the *tosca-types* GitHub repository, where a set of artifacts is defined to perform the installation of the software (SLURM, Galaxy, NFS, etc.) via the corresponding Ansible Roles.

```
tosca_definitions_version: tosca_simple_yaml_1_0
imports:
  - indigo_custom_types: indigo-dc/tosca-types/master/custom_types.yaml
topology_template:
  node_templates:
    elastic_cluster_front_end:
      type: tosca.nodes.indigo.ElasticCluster
      properties:
        deployment_id: orchestrator_deployment_id
      requirements:
        - lrms: lrms_front_end
        - wn: wn_node

    galaxy_portal:
      type: tosca.nodes.indigo.GalaxyPortal
      requirements:
        - lrms: lrms_front_end

    lrms_front_end:
      type: tosca.nodes.indigo.LRMS.FrontEnd.Slurm
      properties:
        wn_ips: { get_attribute: [ lrms_wn, private_address ] }
      requirements:
        - host: lrms_server

    lrms_server:
      type: tosca.nodes.indigo.Compute
      capabilities:
        endpoint:
          properties:
            dns_name: slurmserver
            network_name: PUBLIC
            ports:
              http_port:
                protocol: tcp
                source: 80
  outputs:
    galaxy_url:
      value:
        concat:
          - "http://"
          - get_attribute: [ lrms_server, public_address, 0 ]
          - "/galaxy"
```

Figure 2. A modified excerpt of the TOSCA template to describe a virtual elastic computing cluster to support the Galaxy bioinformatics application.

The output of the TOSCA template is the endpoint directly used by the end user to access the Galaxy portal by means of a web browser, where jobs are submitted to the SLURM queue as part of the operations performed while processing datasets in Galaxy.

4.2. Mesos-based Elastic Computing Cluster

Figure 3 includes a simplified TOSCA template to illustrate the various technical capabilities and integration that are supported by the Heat translator and the IM orchestrator. TOSCA is quite flexible and allows to define non normative types, a functionality employed in this

```
tosca_definitions_version: tosca_simple_yaml_1_0

artifact_types:
  tosca.artifacts.AnsibleGalaxy.role:
    derived_from: tosca.artifacts.Root

node_types:
  tosca.nodes.MesosSlave:
    derived_from: tosca.nodes.SoftwareComponent
    properties:
      master_ip:
        type: string
    artifacts:
      mesos_agent_role:
        file: indigo-dc.mesos
        type: tosca.artifacts.AnsibleGalaxy.role
    interfaces:
      Standard:
        create:
          implementation: mesos_slave_install.yml
          inputs:
            mesos_master_ips: [ { get_property: [ SELF, master_ip ] } ]

topology_template:
  node_templates:
    mesos_slave:
      type: tosca.nodes.MesosSlave
      properties:
        master_ip: 192.168.0.1
      requirements:
        - host: mesos_slave_server

    mesos_slave_server:
      type: tosca.nodes.Compute
      capabilities:
        scalable:
          type: tosca.capabilities.Scalable
```

Figure 3. A simplified example to deploy a scalable set of Mesos slaves provisioned with an Ansible role.

example: a new artifact type is defined to describe the needed Ansible Galaxy Role(s). To avoid any confusion, it is important to point out that Ansible Galaxy, an online repository of Ansible Roles, has nothing to do with the Galaxy bioinformatics application described in the previous example, despite using the same name.

We then define a new node type to describe the necessary steps to have a configured Mesos slave. For that we first reference an Ansible Role artifact: it will then be interpreted by either the translator (converted to a HOT SoftwareConfig/Deployment with an inline script installing the role) or the IM (installed through SSH). This Ansible Role will then be used inside the Ansible playbook mesos_slave_install.yml. This playbook needs the list of IPs of the Mesos masters to register, hence we create an input property on the defined SoftwareComponent.

After defining those new types we enter the *topology_template* section, where we instantiate the MesosSlave software component defined earlier with an actual value for the master IP. In a full example this value would be directly retrieved from the output parameter of a Master

instantiation, similarly as what we can see in the previous figure 2.

We also instantiate a Compute node to host this software component with a scalable capability. For the translation to HOT this template will result in a set of SoftwareConfig/Deployment with dependencies (for example the role installation needs to happen before executing the playbook) and a Nova server. Those elements are then stored in a substack, that is referenced in a ResourceGroup to provide the scalability part. The passing of parameters is kept. On the IM side all of this happens directly through the IM, by launching new compute nodes via the IaaS API and contextualizing directly with SSH and Ansible.

5. Conclusion and Future Work

This paper has described the adoption of the TOSCA standard, for the description of application architectures to be deployed on a Cloud, in the INDIGO-DataCloud project focusing on the definition of complex computing clusters. Two examples of these complex clusters have been provided, an elastic cluster to support a bioinformatics application and the deployment of a scalable Mesos cluster for the execution of jobs and long-running services.

The extensibility features of the standard has allows to create additional non-normative types to support the specific requirements of the applications supported by the project, such as the automated elasticity required by the underlying virtual computing clusters. The usage of TOSCA templates, being a declarative language, coupled with the automated installation capabilities provided by Ansible Roles has paved the way to provide deterministic high-level declarations of complex clusters that can be deployed across multiple on-premises Clouds (OpenStack and OpenNebula) and public Clouds. By leveraging Heat, in the case of OpenStack and the Infrastructure Manager for OpenNebula and public Clouds, a wide variety of cloud provides on which to orchestrate complex virtual cluster is now possible.

Future work includes addressing other types of complex clusters, such as those required in Big Data involving distributed computing at scale and large volumes of data processing (e.g. Hadoop, Spark, etc.). Also, we plan to introduce support for hybrid deployments of these complex clusters across different Clouds in order to include multi-site orchestration for elastic computing clusters.

Acknowledgments

The authors would like to thank the European Commission for the financial support for project INDIGO-DataCloud (RIA 653549).

Bibliography

[1] Amazon cloud formation. `https://aws.amazon.com/cloudformation/`.
[2] Cloudify. `http://getcloudify.org/`.
[3] Egi federated cloud. `https://www.egi.eu/infrastructure/cloud/`. Accessed: 2017-01-13.
[4] Heat orchestration template. `https://docs.openstack.org/developer/heat/template_guide/hot_guide.html`.
[5] Heat translator. `https://docs.openstack.org/developer/heat-translator/`.
[6] Tosca parser. `https://wiki.openstack.org/wiki/TOSCA-Parser`.
[7] Palma D, Rutkowski M, and Spatzier T. TOSCA Simple Profile in YAML Version 1.0. Technical report, 2016.
[8] de Alfonso C, Caballer M, Alvarruiz F, and Hernández V. An energy management system for cluster infrastructures. *Computers & Electrical Engineering*, 39(8):2579–2590, 2013.
[9] Hochstein L. *Ansible: Up and Running, Automating Configuration Management and Deployment the Easy Way*. O'Reilly Media, 2014.
[10] S Li. TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0. Technical report, 2016.
[11] Caballer M, Blanquer I, Moltó G, and de Alfonso C. Dynamic management of virtual infrastructures. *Journal of Grid Computing*, 13(1):53–70, 2015.