EOSC-hub

eosc-hub.eu

@EOSC_eu

TOSCA Orchestration

Marica Antonacci - INFN BARI

Istanziazione e utilizzo di batch system on demand su infrastrutture Cloud, 25-28 November 2019

EOSC-hub receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 777536.

Outline

- What is TOSCA
- TOSCA in a nutshell
- Portability and TOSCA Orchestrators
- INDIGO PaaS Orchestration

TOSCA

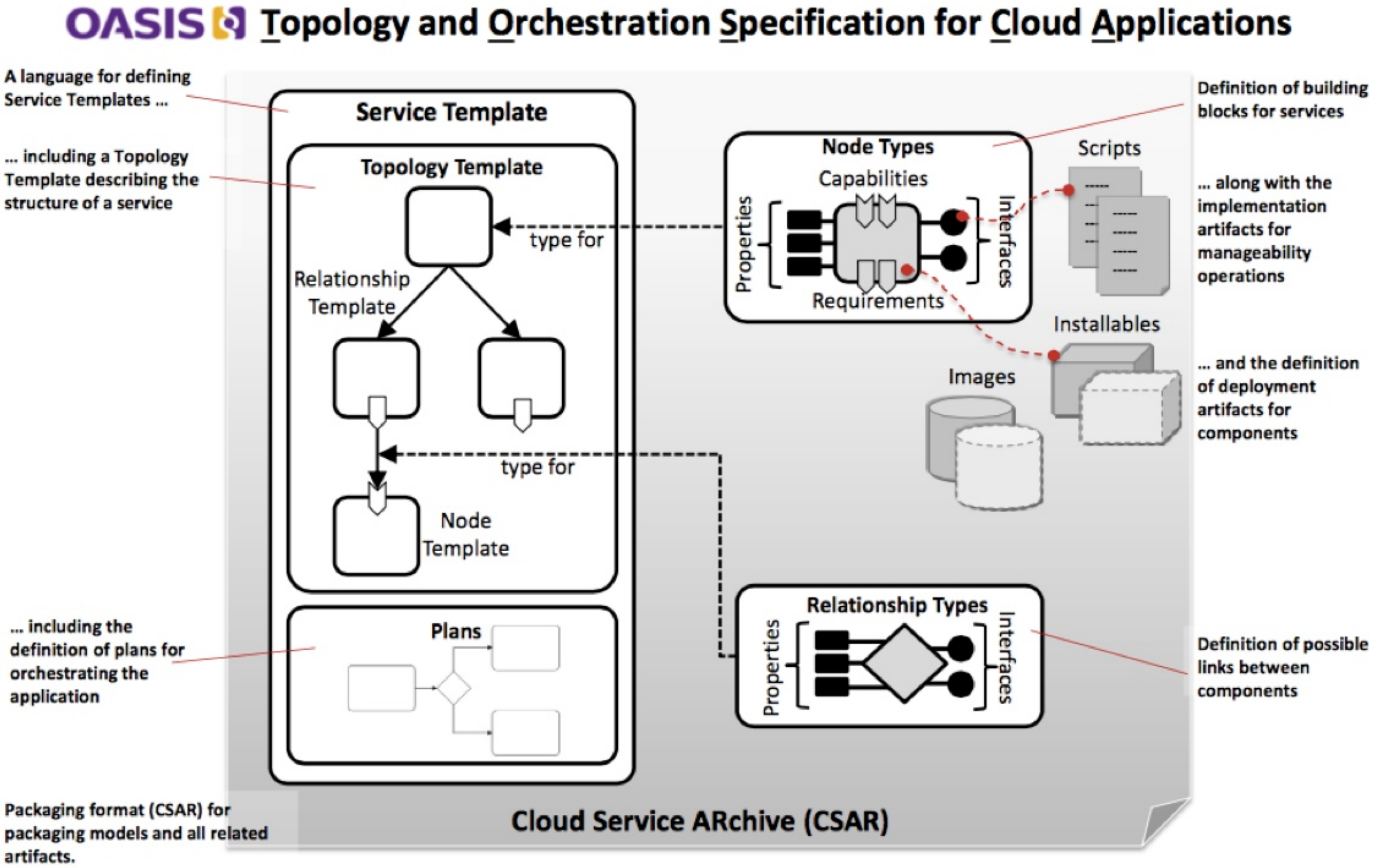
- **T**opology and **O**rchestration **S**pecification for **C**loud **A**pplications
- Standardizes the language to describe
  - The structure of an ITService (its **topology** model)
  - How to orchestrate operational behavior (plans such as build, deploy, patch, shutdown, etc.)
    - Leveraging the BPMN standard
  - Declarative model that spans applications, virtual and physical infrastructure

TOSCA: Goals

- Automated Application Deployment and Management
- Portability of Application Descriptions and their Management
- Interoperability and Reusability of Components

TOSCA in a nutshell

OASIS Topology and Orchestration Specification for Cloud Applications



A language for defining Service Templates …

… including a Topology Template describing the structure of a service

… including the definition of plans for orchestrating the application

Packaging format (CSAR) for packaging models and all related artifacts.

Definition of building blocks for services

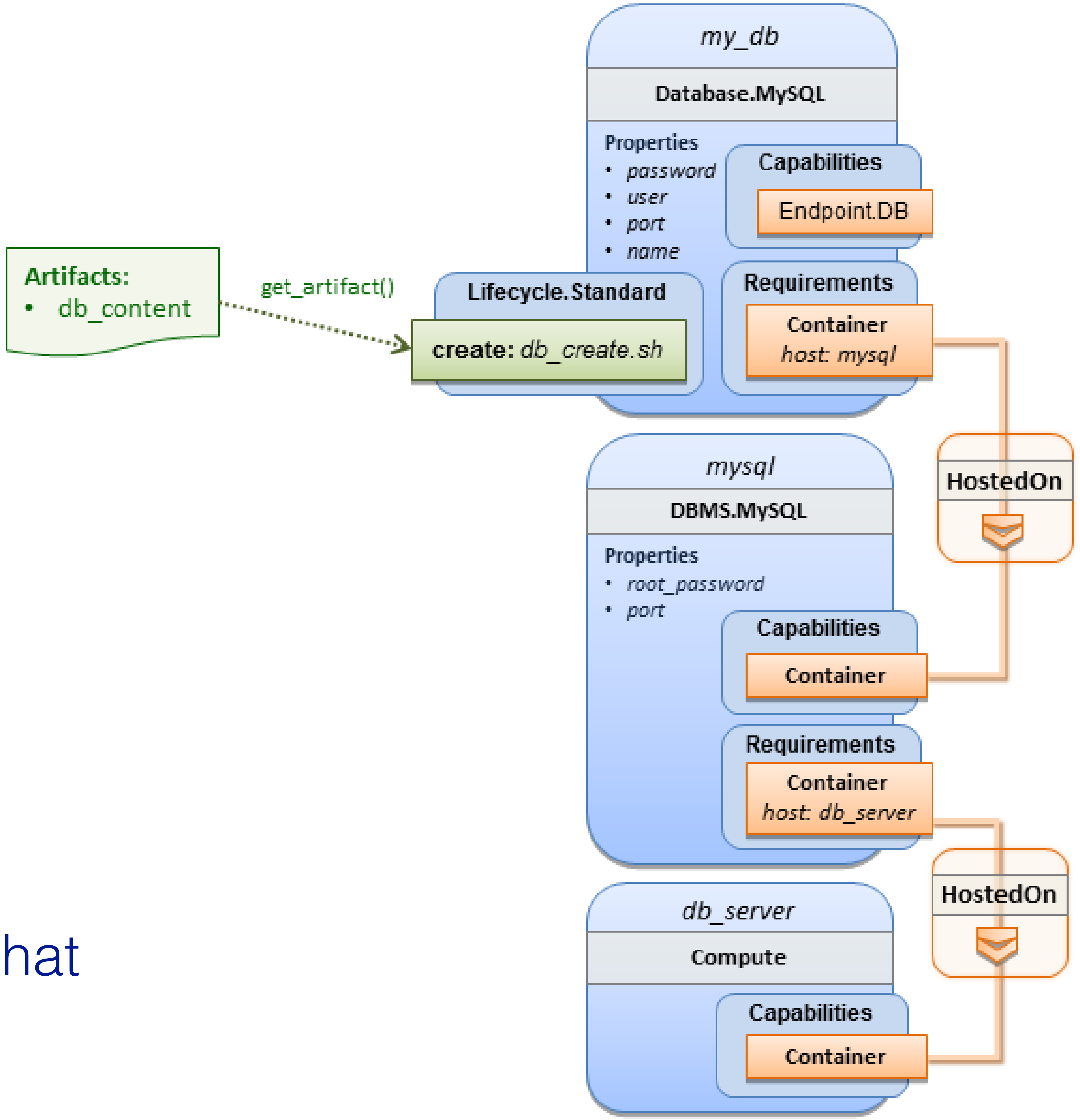
… along with the implementation artifacts for manageability operations

… and the definition of deployment artifacts for components

Definition of possible links between components

TOSCA: Infrastructure as a Code

- **3 layers**
  - **Infrastructure**
    (Cloud or DC objects)
  - **Platform or Middleware**
    (App containers)
  - **Application modules, schemas and configurations**
- **Relationships** between components:
  - What's hosted on what or installed on what
  - What's connected to what



TOSCA Topology

- Components in the topology are called **Nodes**
- Each Node has a **Type** (e.g. Host, BD, Web server).
  - The Type is abstract and hence portable
  - The Type defines Properties and Interfaces
- An Interface is a set of hooks (named Operations)
- Nodes are connected to one another using **Relationships**
- Both Node Types and Relationship Types can be **derived**

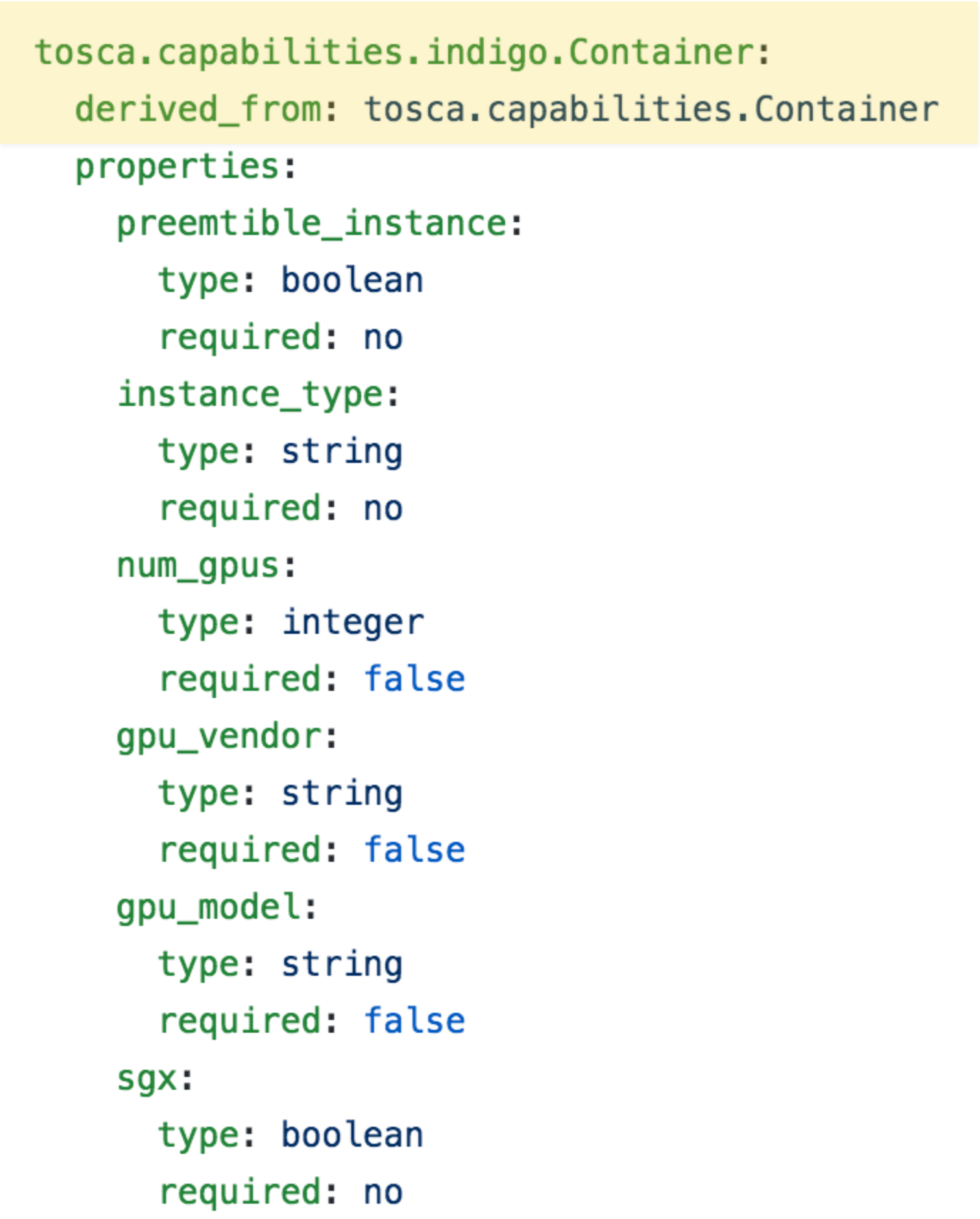
Normative Types

- The **TOSCA Simple Profile in YAML** specifies a rendering of TOSCA to provide a more accessible syntax and a more concise expressiveness of the TOSCA DSL
- It provides a rich set of **base** types (node types and relationship types): e.g. ‘Compute’ node type
- Some **non-normative types** are provided as well but implementations of this specification are not required to support these types for conformance.

Custom types

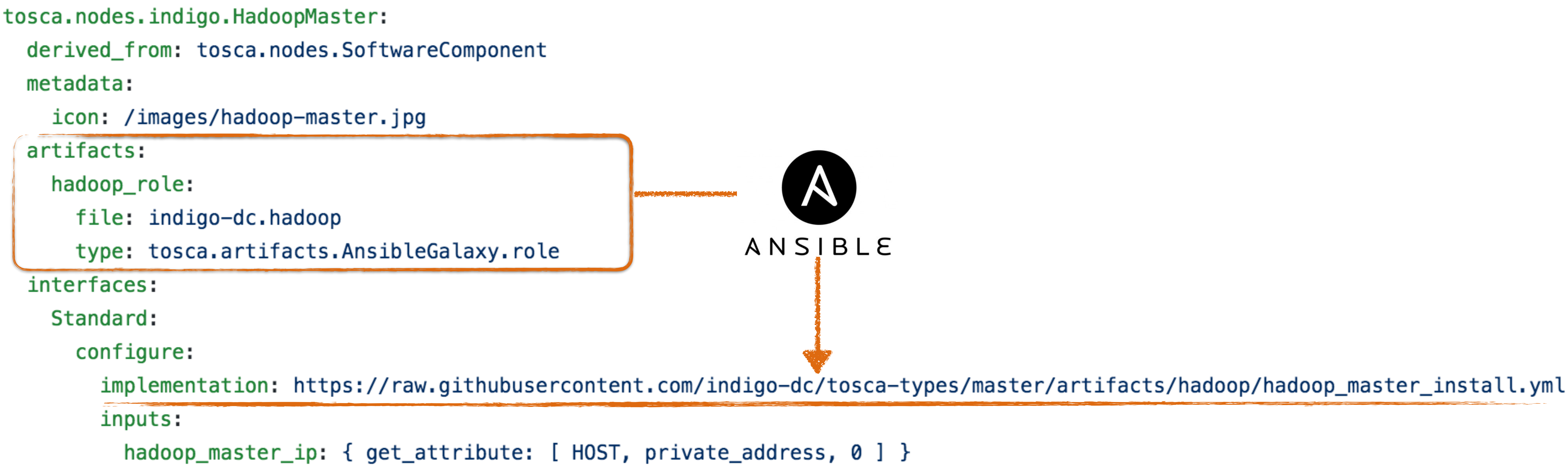
- TOSCA is highly versatile
  - You can define custom types for nodes, relationships, and capabilities —> can be used in different domains
  - **indigo custom types**

https://github.com/indigo-dc/tosca-types



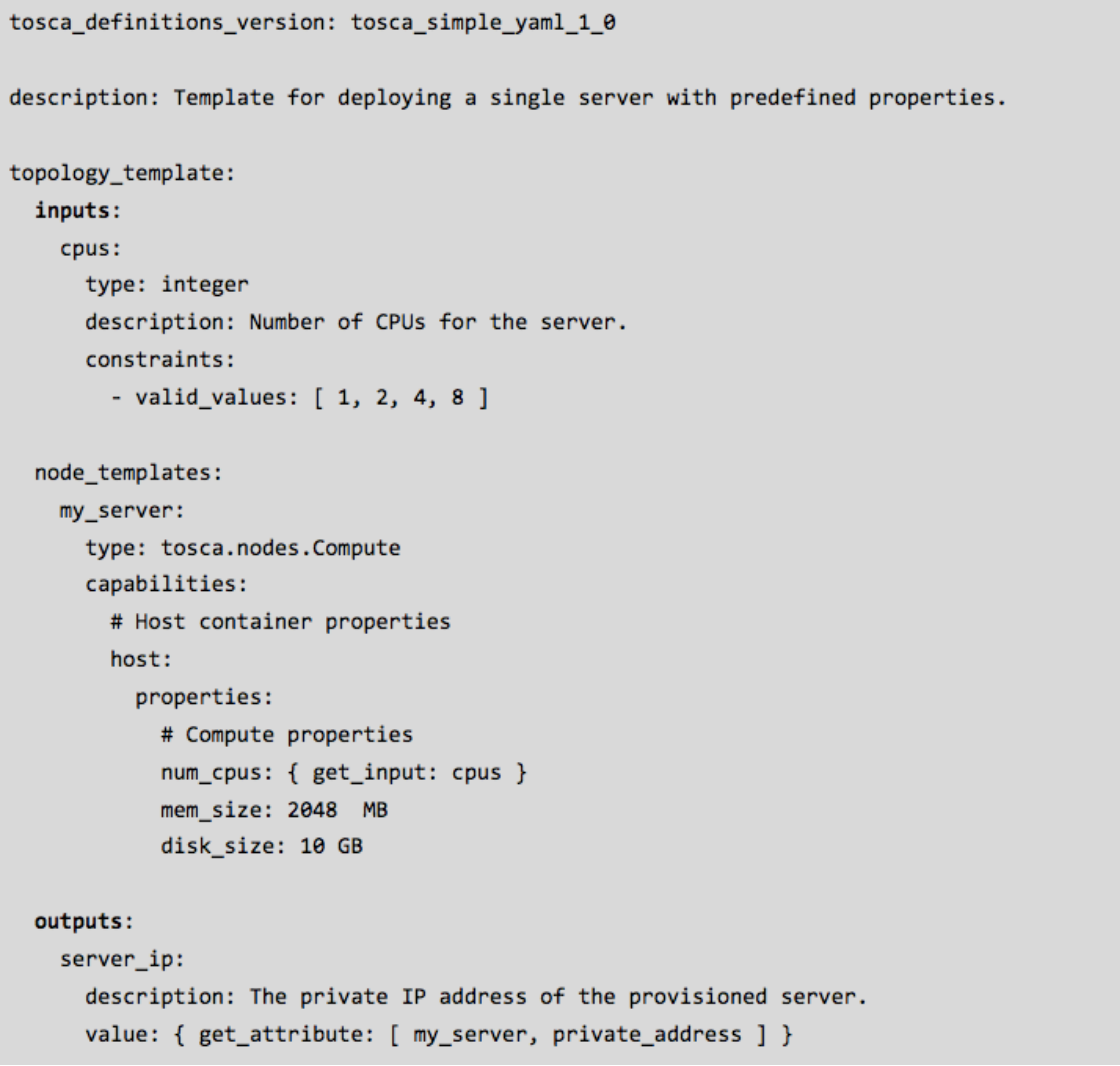
```
tosca.capabilities.indigo.Container:
  derived_from: tosca.capabilities.Container
  properties:
    preemtible_instance:
      type: boolean
      required: no
    instance_type:
      type: string
      required: no
    num_gpus:
      type: integer
      required: false
    gpu_vendor:
      type: string
      required: false
    gpu_model:
      type: string
      required: false
    sgx:
      type: boolean
      required: no
```

indigo custom type example



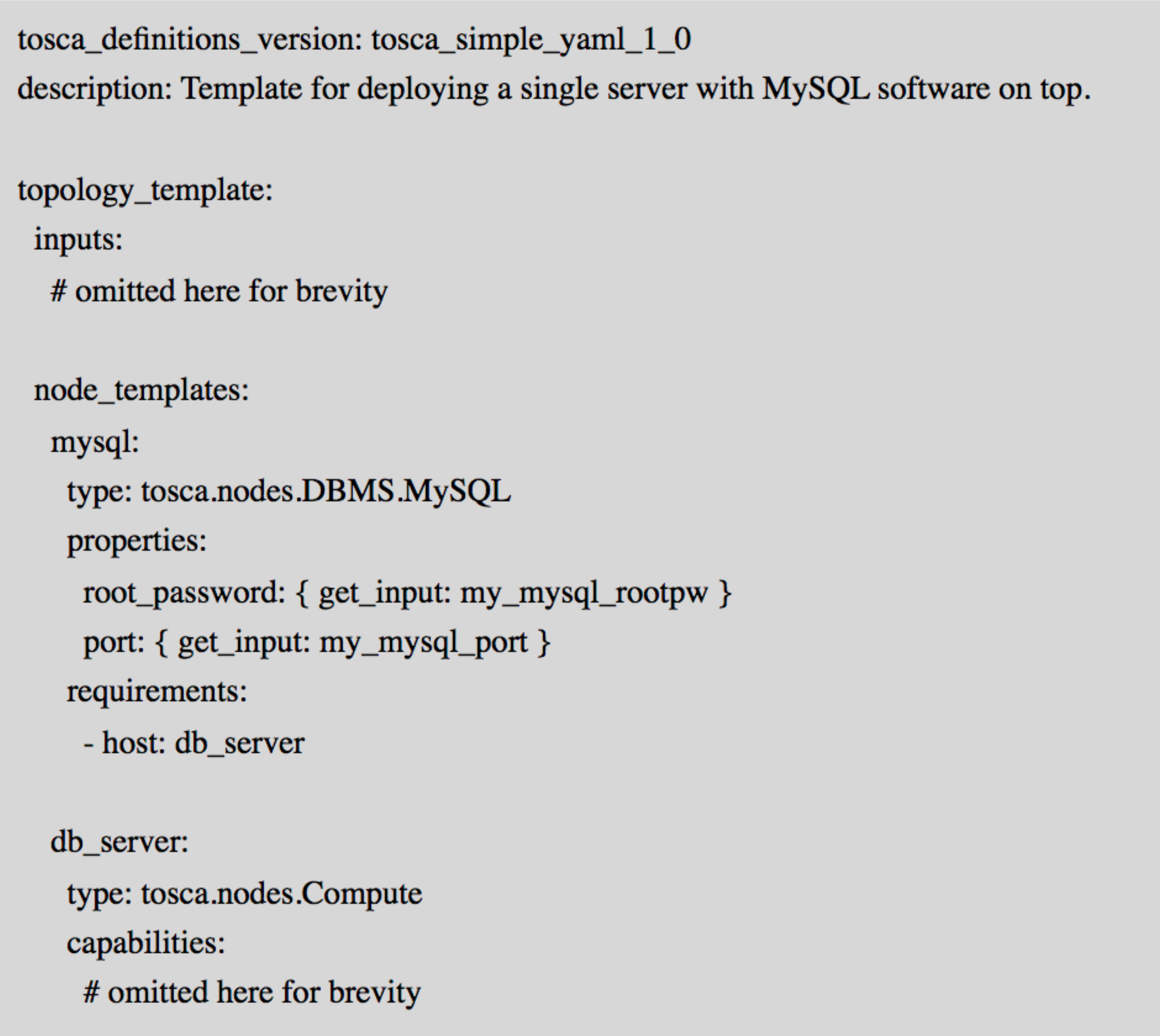
```
tosca.nodes.indigo.HadoopMaster:
  derived_from: tosca.nodes.SoftwareComponent
  metadata:
    icon: /images/hadoop-master.jpg
  artifacts:
    hadoop_role:
      file: indigo-dc.hadoop
      type: tosca.artifacts.AnsibleGalaxy.role
  interfaces:
    Standard:
      configure:
        implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/master/artifacts/hadoop/hadoop_master_install.yml
        inputs:
          hadoop_master_ip: { get_attribute: [ HOST, private_address, 0 ] }
```

Topology Template Example 1

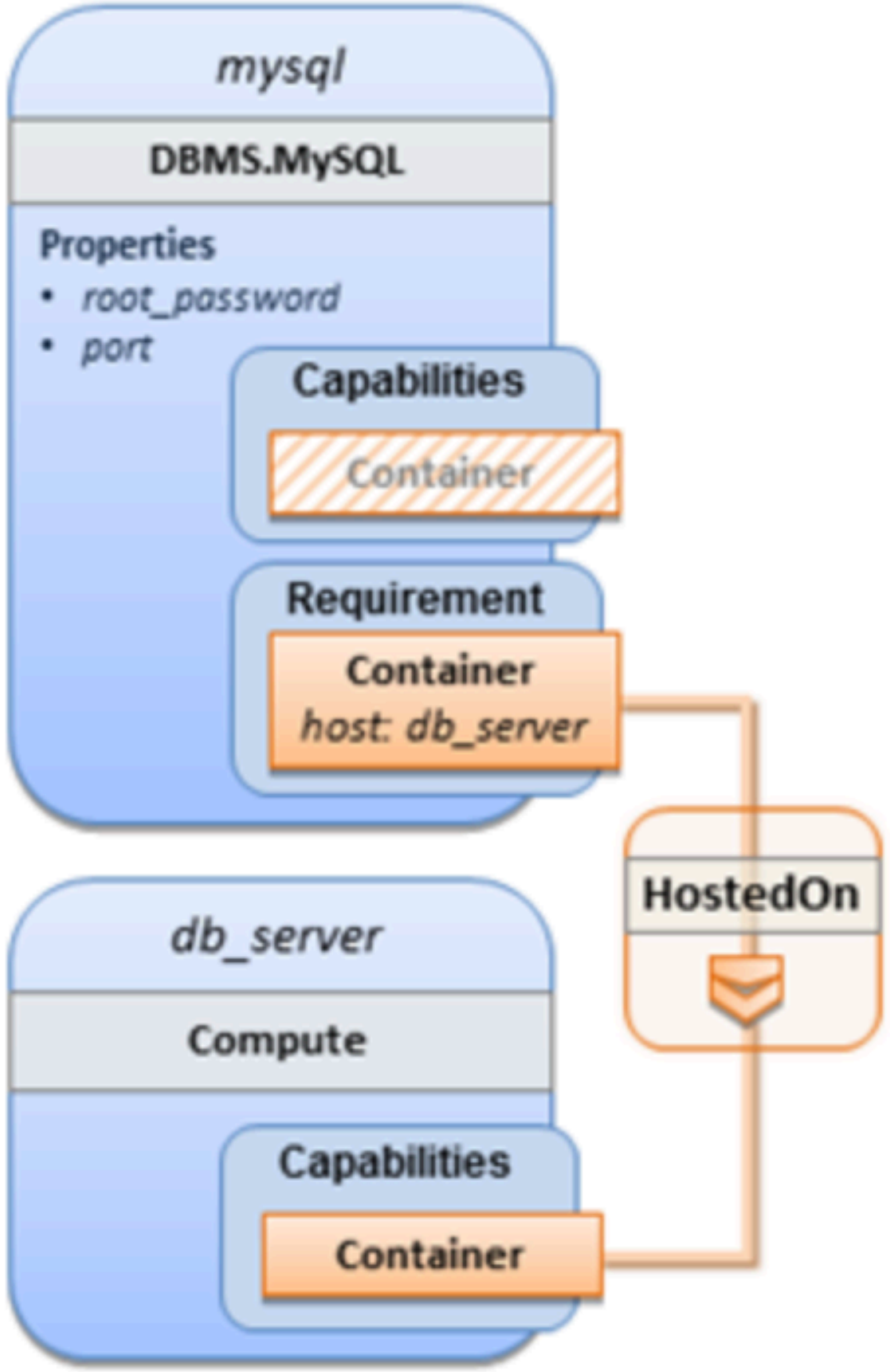


```
tosca_definitions_version: tosca_simple_yaml_1_0

description: Template for deploying a single server with predefined properties.

topology_template:
  inputs:
    cpus:
      type: integer
      description: Number of CPUs for the server.
      constraints:
        - valid_values: [ 1, 2, 4, 8 ]

  node_templates:
    my_server:
      type: tosca.nodes.Compute
      capabilities:
        # Host container properties
        host:
          properties:
            # Compute properties
            num_cpus: { get_input: cpus }
            mem_size: 2048  MB
            disk_size: 10 GB

  outputs:
    server_ip:
      description: The private IP address of the provisioned server.
      value: { get_attribute: [ my_server, private_address ] }
```

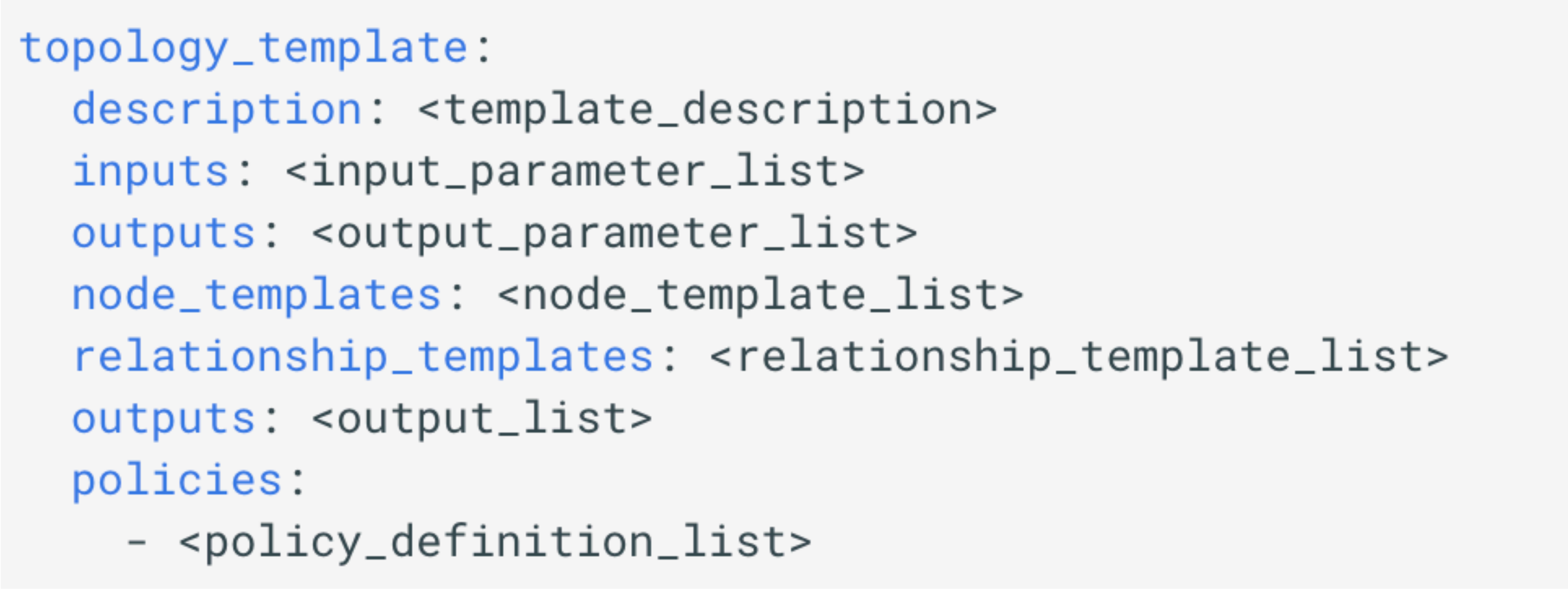
Topology Template Example 2



```
tosca_definitions_version: tosca_simple_yaml_1_0
description: Template for deploying a single server with MySQL software on top.

topology_template:
  inputs:
    # omitted here for brevity

  node_templates:
    mysql:
      type: tosca.nodes.DBMS.MySQL
      properties:
        root_password: { get_input: my_mysql_rootpw }
        port: { get_input: my_mysql_port }
      requirements:
        - host: db_server

    db_server:
      type: tosca.nodes.Compute
      capabilities:
        # omitted here for brevity
```



Simplified Topology Template Structure



```
topology_template:
  description: <template_description>
  inputs: <input_parameter_list>
  outputs: <output_parameter_list>
  node_templates: <node_template_list>
  relationship_templates: <relationship_template_list>
  outputs: <output_list>
  policies:
    - <policy_definition_list>
```

Node template

- An instance of a type (like Object to Class)
  - Has specific properties
  - Has artifacts:
    - What to install
    - How to install (mapped to interface hooks)
  - Has requirements and capabilities (or relationships)

Node template

A Node template is an instance of a specified Node Type and can provide customized properties, constraints or operations which override the defaults provided by its Node Type and its implementations.

```
<node_template_name>:
  type: <node_type_name>
  properties:
    <property_definitions>
  requirements:
    <requirement_definitions>
  capabilities:
    <capability_definitions>
  interfaces:
    <interface_definitions>
```

Node Type

- Describes a Cloud or Software type (e.g. Server or Apache)

```
<node_type_name>:
  derived_from: <parent_node_type_name>
  description: <node_type_description>
  properties:
    <property_definitions>
  attributes:
    <attribute_definitions>
  requirements:
    - <requirement_definition_1>
    ...
    - <requirement_definition_n>
  capabilities:
    <capability_definitions>
  interfaces:
    <interface_definitions>
  artifacts:
    <artifact_definitions>
```

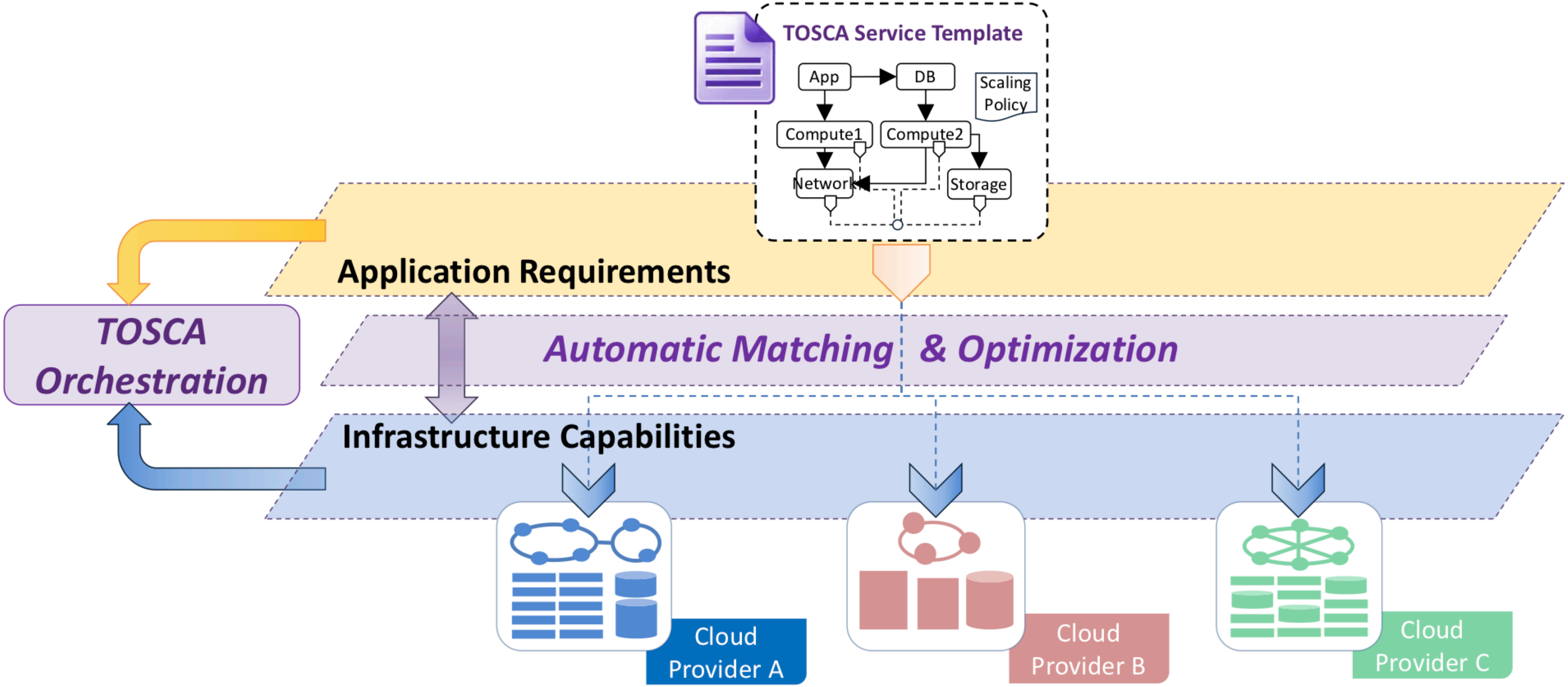
Relationship Type

The basic relationship types are:

- **dependsOn** – abstract type and its sub types:
- **hostedOn** – a node is contained within another
- **connectsTo** – a node has a connection configured to another

```
<relationship_type_name>:
  derived_from: <parent_relationship_type_name>
  description: <relationship_description>
  properties:
    <property_definitions>
  attributes:
    <attribute_definitions>
  interfaces:
    <interface_definitions>
  valid_target_types: [ <entity_name_or_type_1>, ..., <entity_name_or_type_n> ]
```

Portability & TOSCA Orchestrators



Known TOSCA Implementations

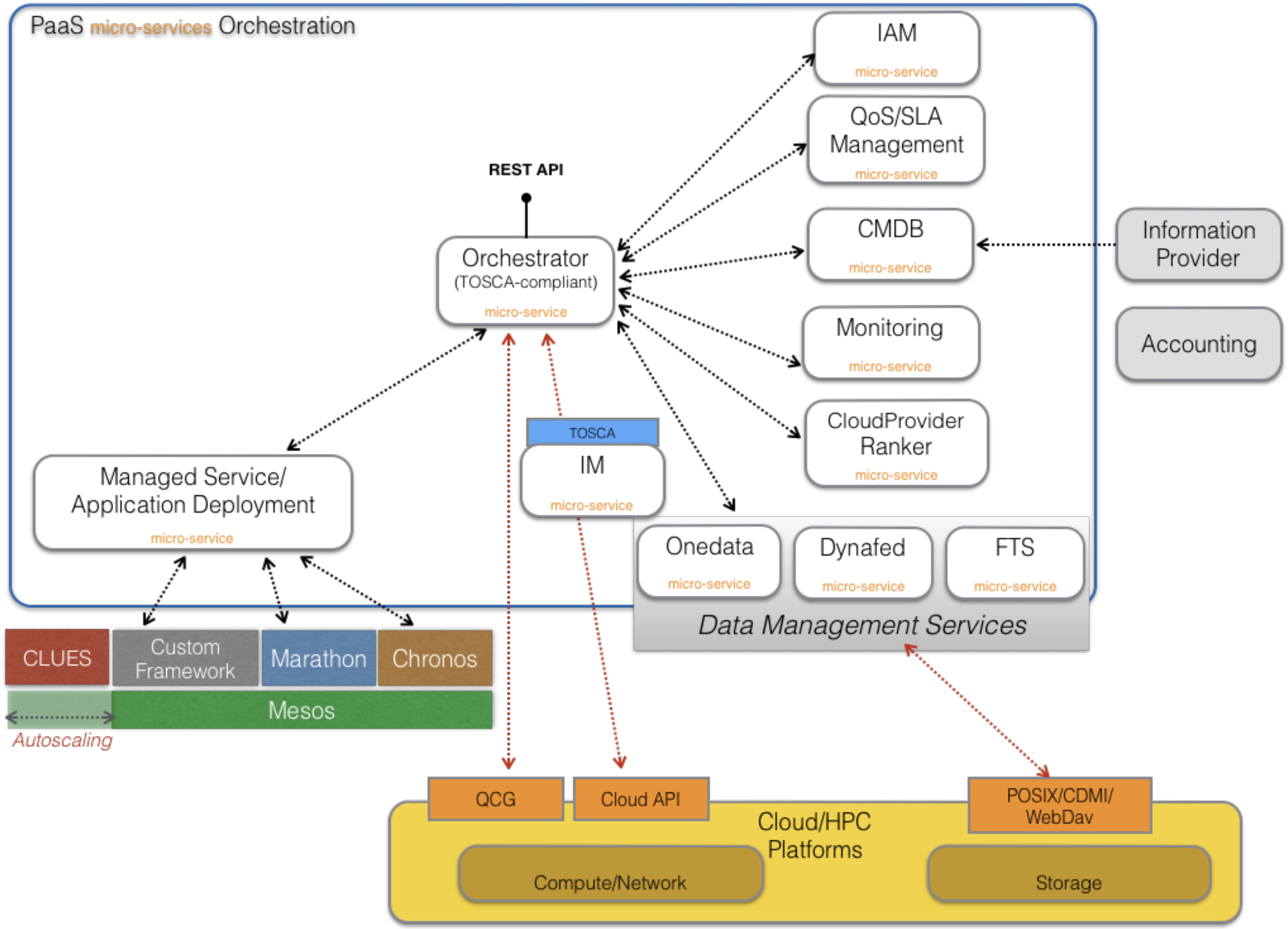
- Alien4Cloud
- Apache AriaTosca
- Celar
- Cloudify
- **INDIGO PaaS Orchestrator**
- Openstack Heat
- OpenTOSCA
- Puccini
- …

https://wiki.oasis-open.org/tosca/TOSCA-implementations

INDIGO PaaS Orchestration

- The **PaaS Orchestrator** is based on the developments carried out during the INDIGO-DataCloud project
  - advanced features and important enhancements are being implemented in the framework of three projects: DEEP-Hybrid DataCloud, eXtreme-DataCloud and EOSC-Hub
- It allows to coordinate 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.
- 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

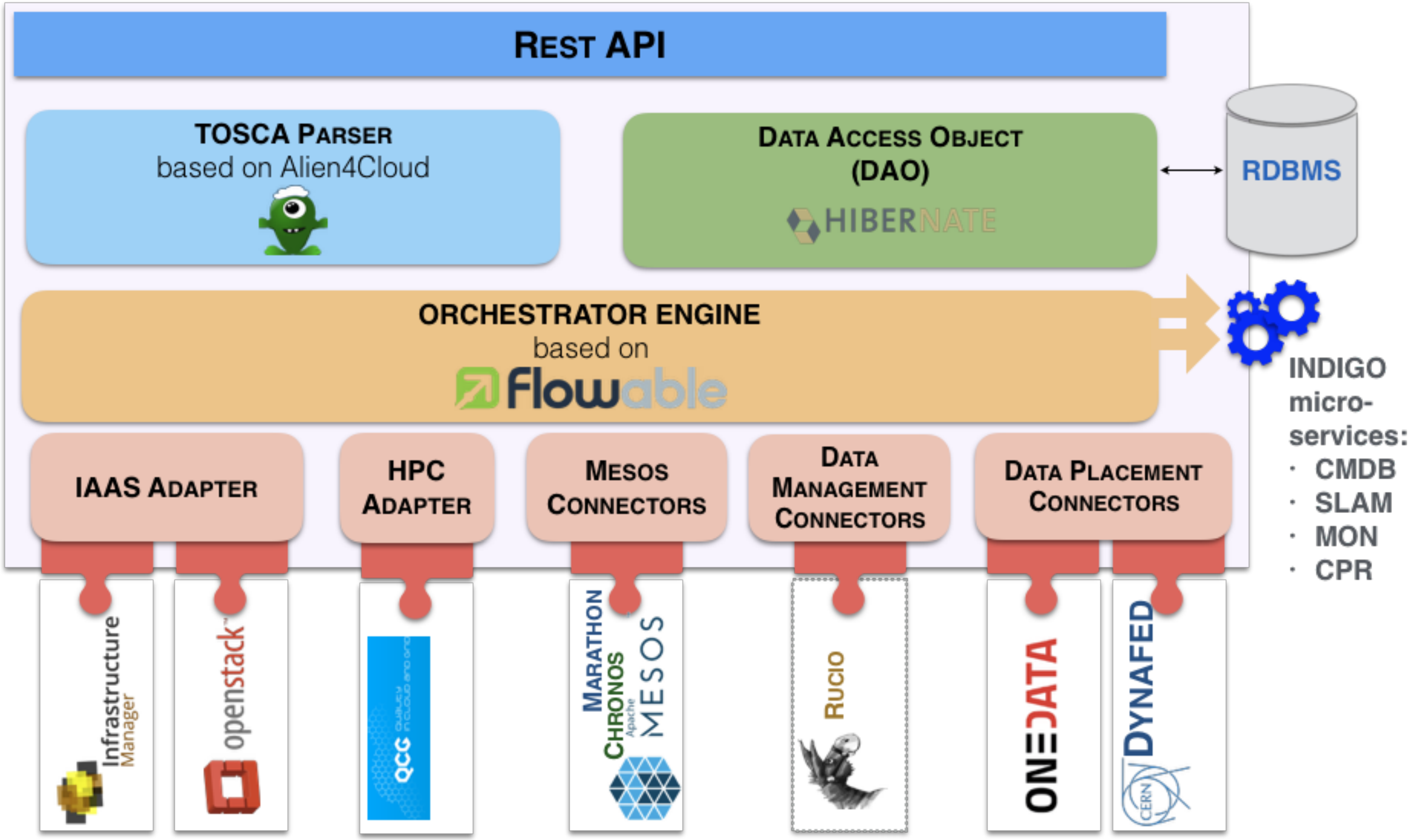
INDIGO Platform as a Service



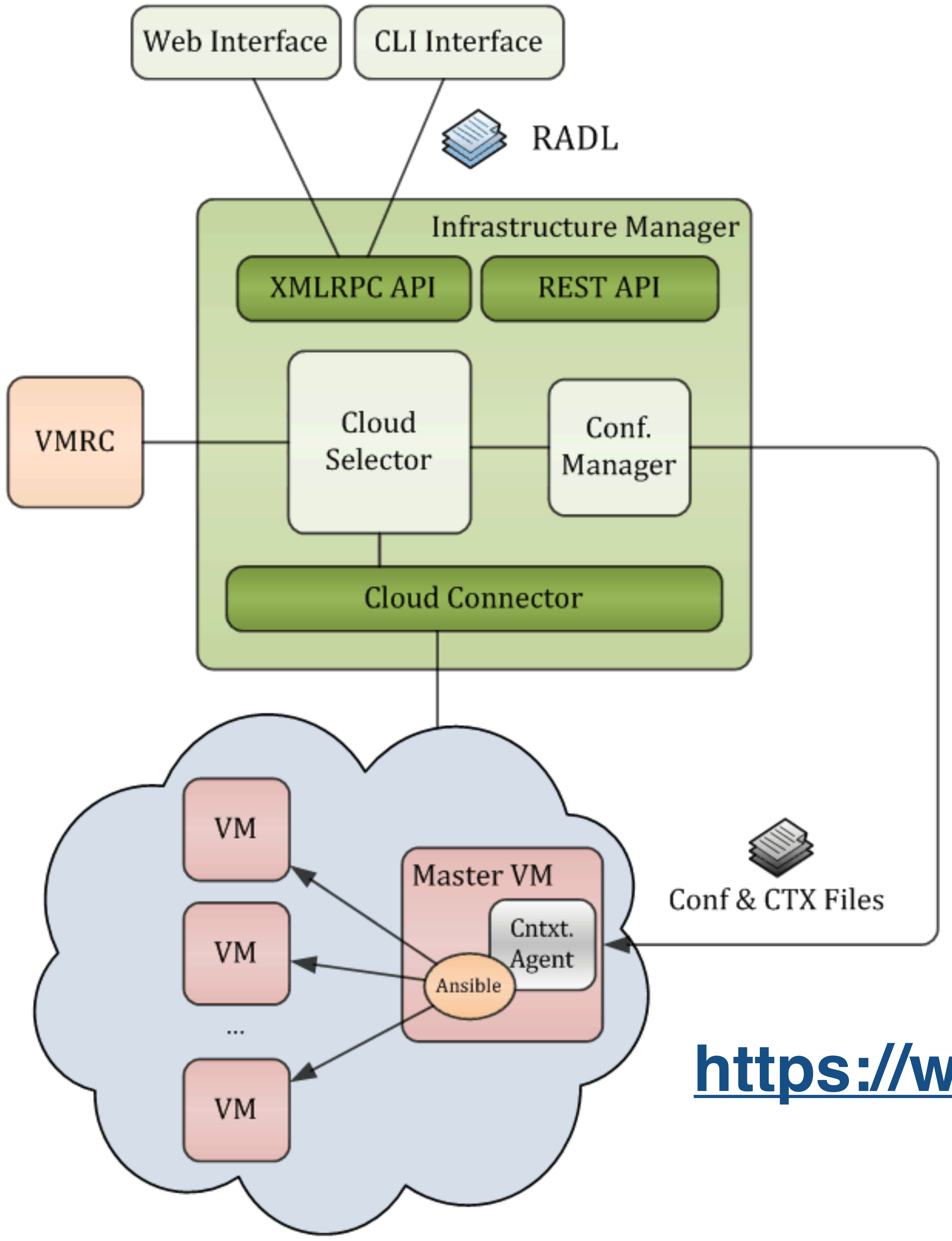
The deployment workflow

- The Orchestrator receives the deployment request (TOSCA template)
- The Orchestrator collects all the information needed to deploy the virtual infra/service/job consuming others PaaS μServices APIs:
  - **SLAM Service**: get the prioritized list of SLAs per user/group;
  - **Configuration Management DB**: get the the capabilities of the underlying IaaS platforms;
  - **Data Management Service**: get the status of the data files and storage resources needed by the service/application
  - **Monitoring Service**: get the IaaS services availability and their metrics;
  - **CloudProviderRanker Service** (Rule Engine): sort the list of sites on the basis of configurable rules;
- The orchestrator delegates the deployment to IM, Mesos or QCG-Computing based on the TOSCA template and the list of sites.
- Cross-site deployments are also possible.

PaaS Orchestrator Architecture



Infrastructure Manager architecture

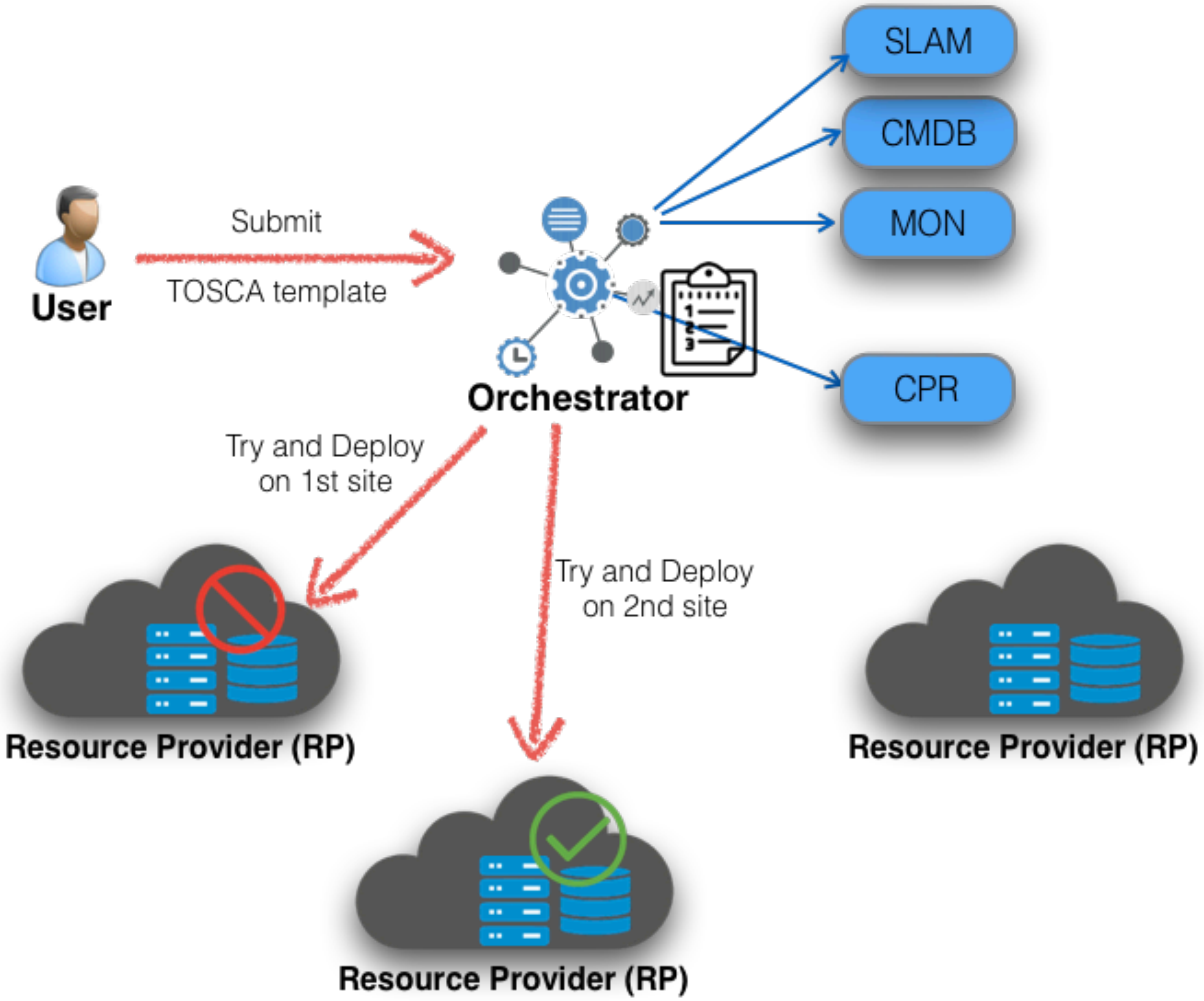


https://www.grycap.upv.es/im/index.php

Deployment retry strategy

- The Orchestrator implements a **trial-and-error** mechanism that allows to re-schedule the deployment on the next available cloud provider from the list of candidate sites.

- Example: deployment fails because of exceeding the quota on the chosen site



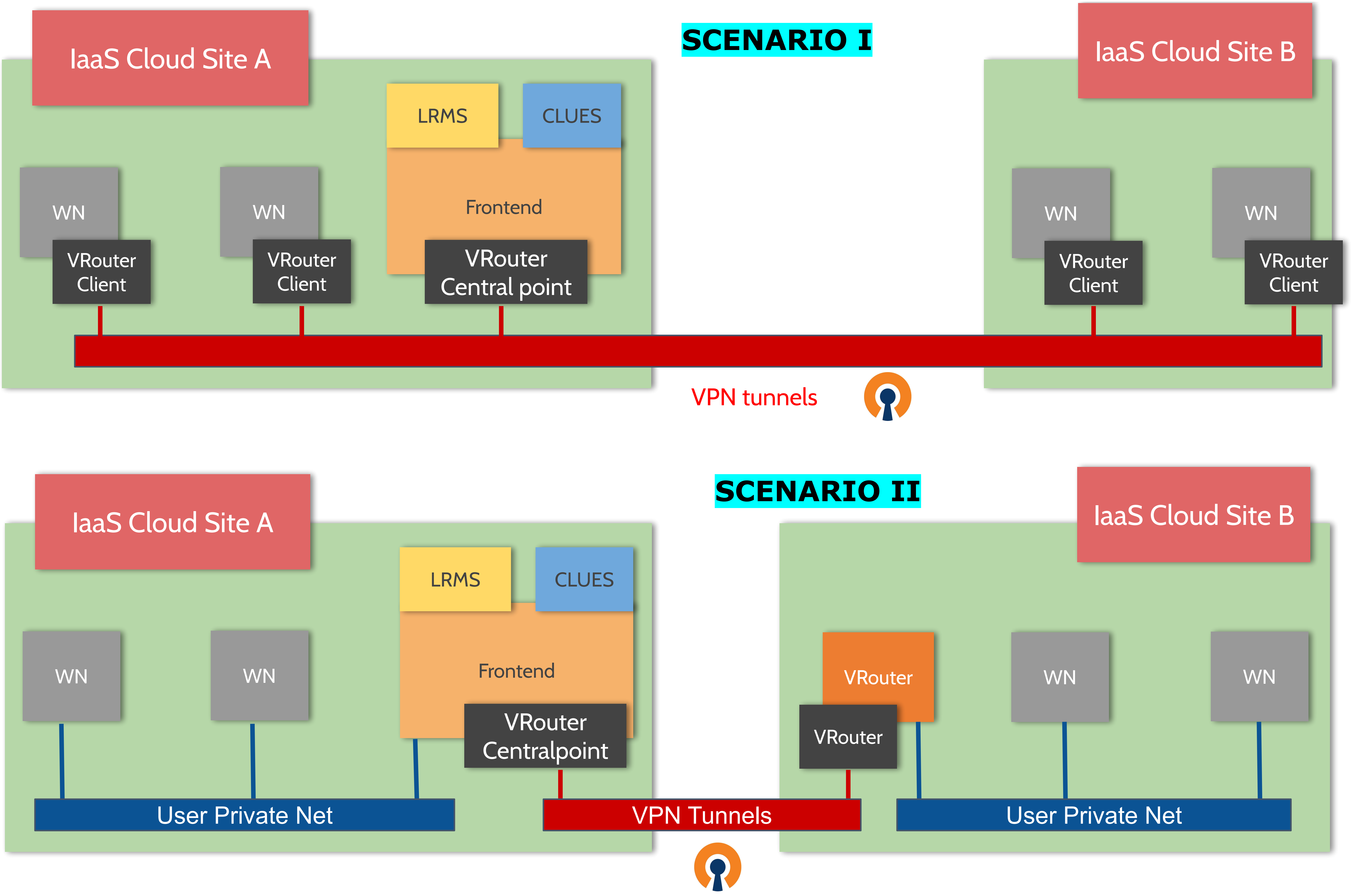
GPU scheduling and HPC integration

- The PaaS Orchestrator supports the deployment of virtual machines and containers that need to **access specialised hardware devices**, namely GPUs, to provide the processing power required by tasks like Machine Learning algorithms
  - the GPU requirements (num, vendor, model) can be specified in the TOSCA template
  - the Orchestrator automatically selects the sites/services that provide the needed capabilities (flavors, gpu support)
- The Orchestrator includes a plugin for **submitting jobs to HPC** facilities
  - exploits the **QCG-Computing** service (PSNC) that exposes REST APIs to submit jobs to the underlying batch systems

Support for hybrid deployments of elastic clusters

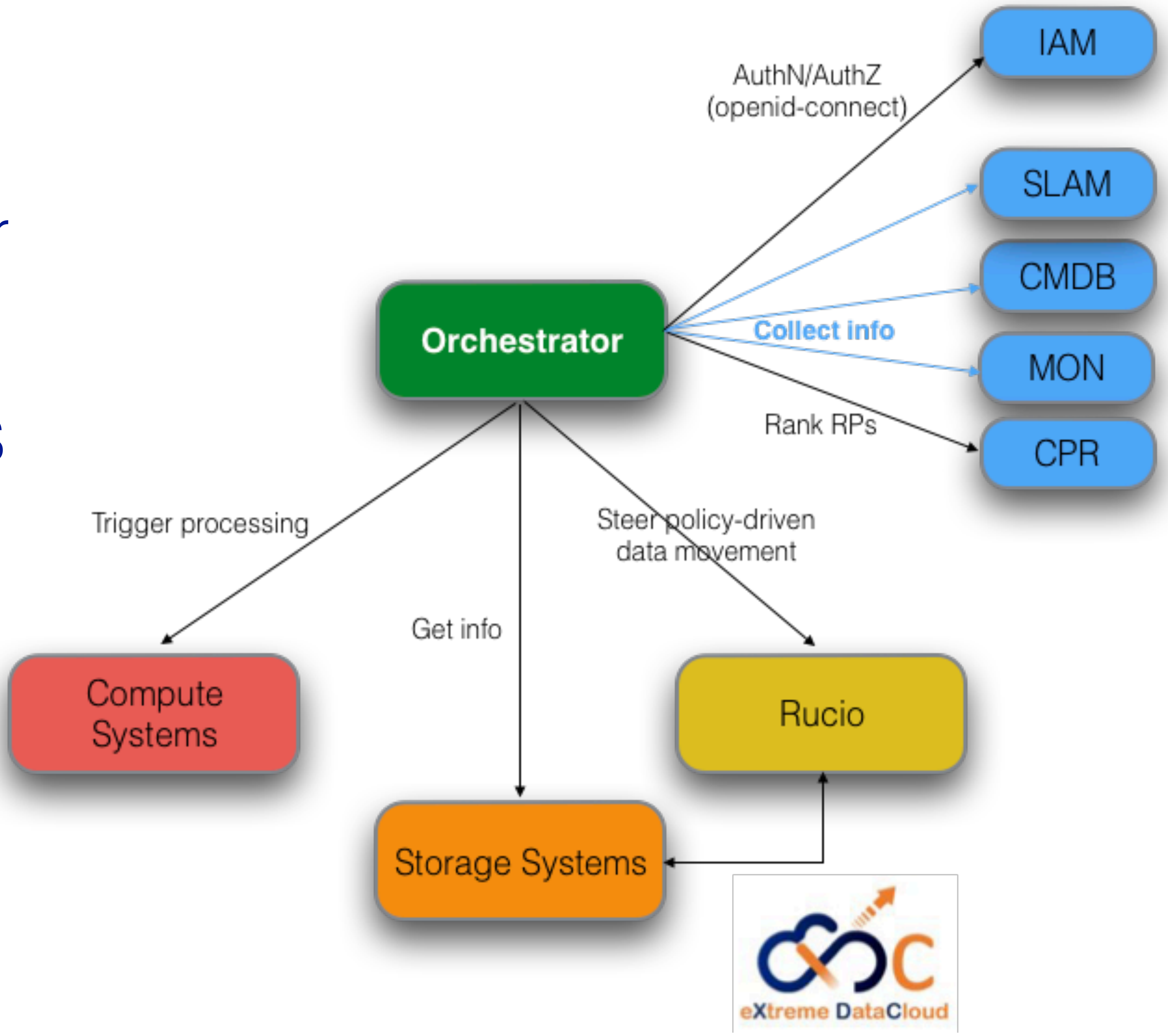
- Scenario I:
  exploits L2 network provided by the sites

- Scenario II:
  dedicated private nets are automatically provisioned



Further features and enhancements

- The PaaS Orchestrator has been enhanced to schedule the processing jobs near the data
- The PaaS Orchestrator is being extended in order to:
- Integrate a data management policy engine (QoS and Data Life Cycle)
  - move data between distributed storages
  - specify different QoS for replicas
  - Support workflows for data pre-processing at ingestion



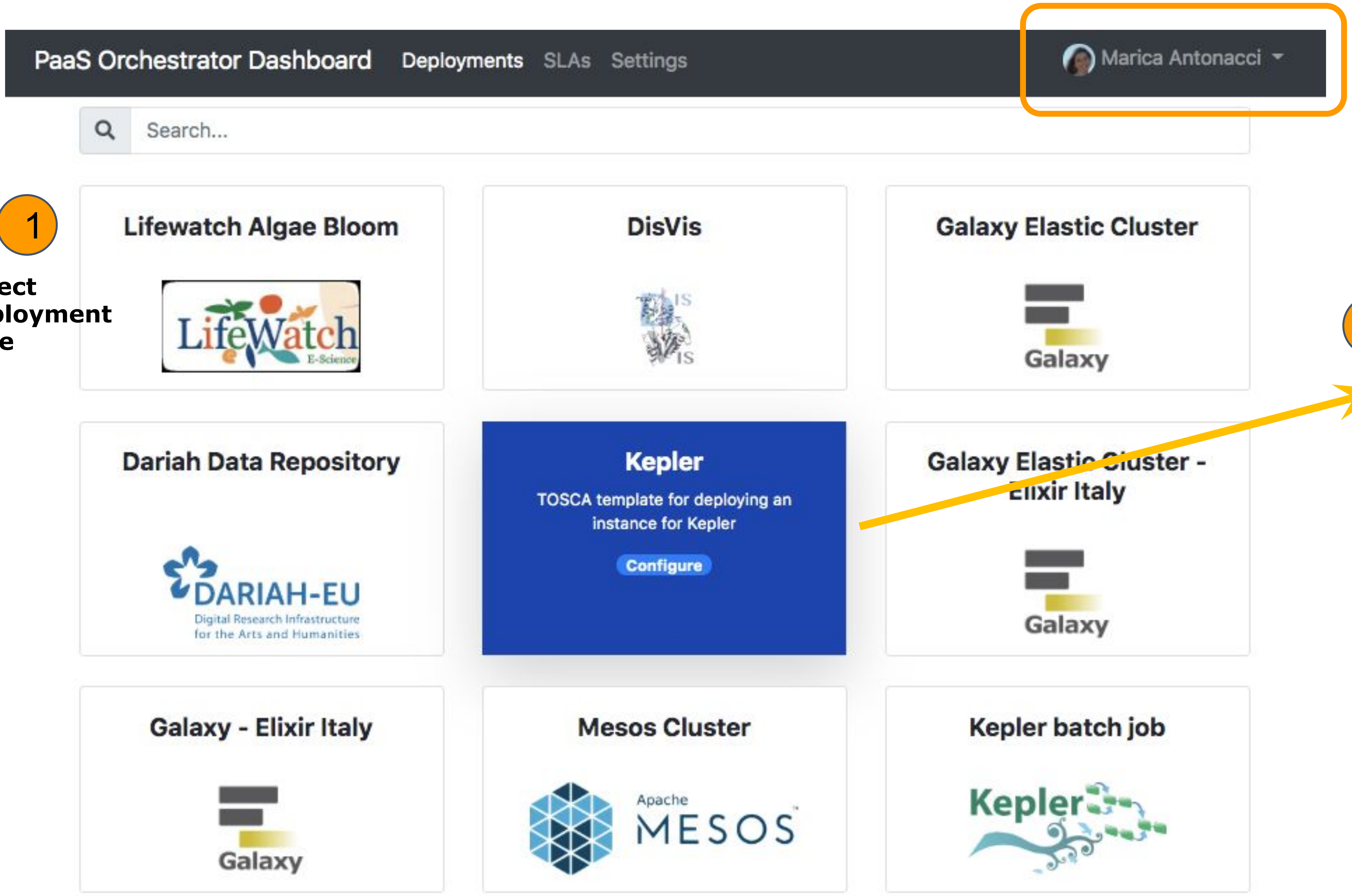
Orchestrator REST APIs

- **Create a deployment:**
  - POST request to /deployments - parameters:
    - template: string containing a TOSCA YAML-formatted template
    - parameters: the input parameters of the deployment (map of strings)
- **Get deployment details**
  - GET request to /deployments:
    - curl 'http://localhost:8080/deployments/<uuid>'
- **Delete deployment**
  - DELETE request
    - curl ‘http://localhost:8080/deployments/<uuid>'
- Documentation: http://indigo-dc.github.io/orchestrator/restdocs/#overview

Orchent: the Orchestrator CLI

```
export ORCHENT_TOKEN=<your access token>
export ORCHENT_URL=<orchestrator_url>

usage: orchent <command> [<args> ...]

Commands:
  help [<command>...]
    Show help.

  depls
    list all deployments

  depshow <uuid>
    show a specific deployment

  depcreate [<flags>] <template> <parameter>
    create a new deployment

  depupdate [<flags>] <uuid> <template> <parameter>
    update the given deployment

  deptemplate <uuid>
    show the template of the given deployment

  depdel <uuid>
    delete a given deployment

  resls <depployment uuid>
    list the resources of a given deployment
```

Orchestrator dashboard



PaaS Orchestrator Dashboard · Deployments · SLAs · Settings · Marica Antonacci

Search...

Lifewatch Algae Bloom

DisVis

Galaxy Elastic Cluster

Dariah Data Repository

Kepler

TOSCA template for deploying an instance for Kepler

Configure

Galaxy Elastic Cluster - Elixir Italy

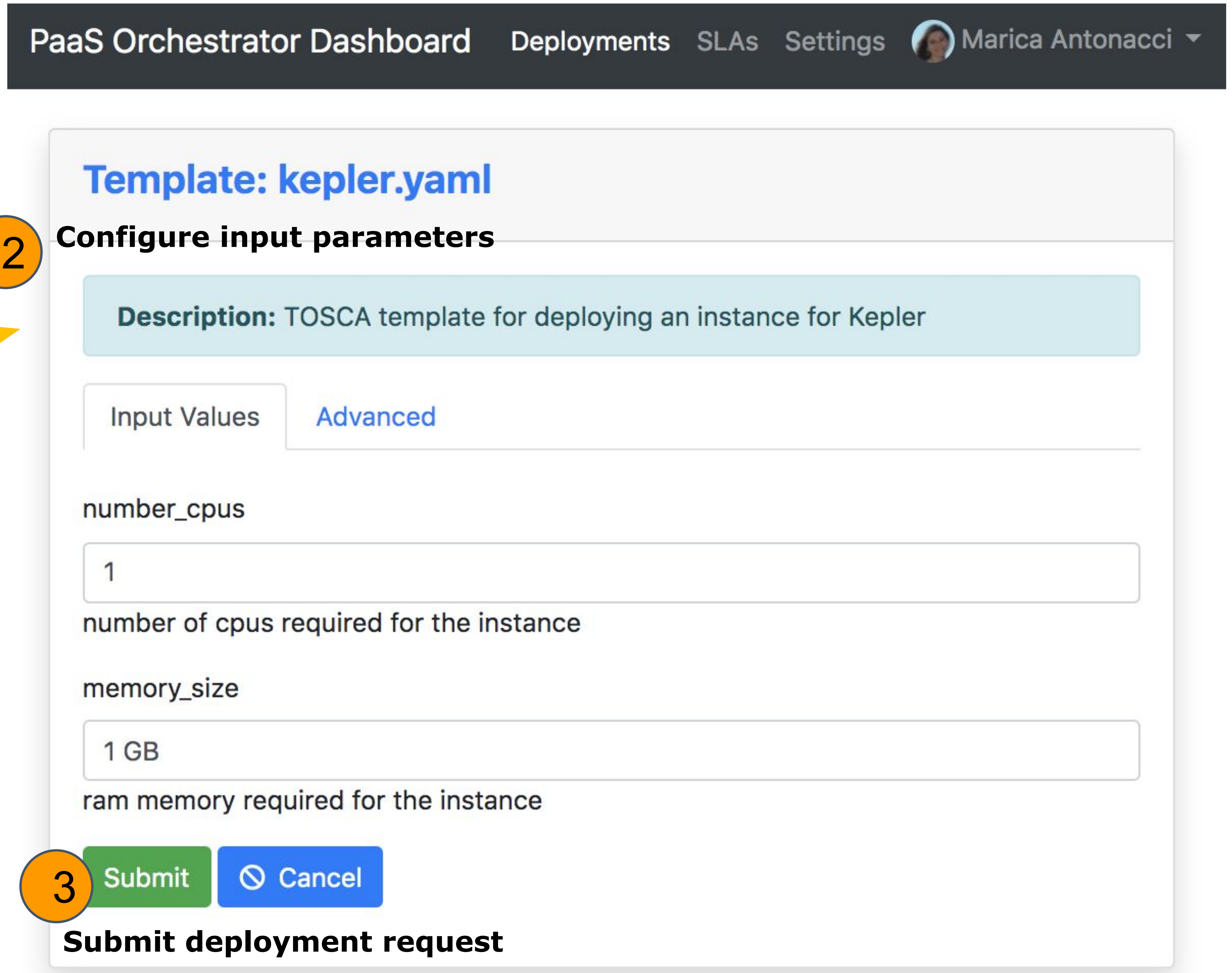
Galaxy - Elixir Italy

Mesos Cluster

Kepler batch job

Authentication via INDIGO IAM

1 Select deployment type



PaaS Orchestrator Dashboard · Deployments · SLAs · Settings · Marica Antonacci

Template: kepler.yaml

2 Configure input parameters

Description: TOSCA template for deploying an instance for Kepler

Input Values | Advanced

number_cpus

1

number of cpus required for the instance

memory_size

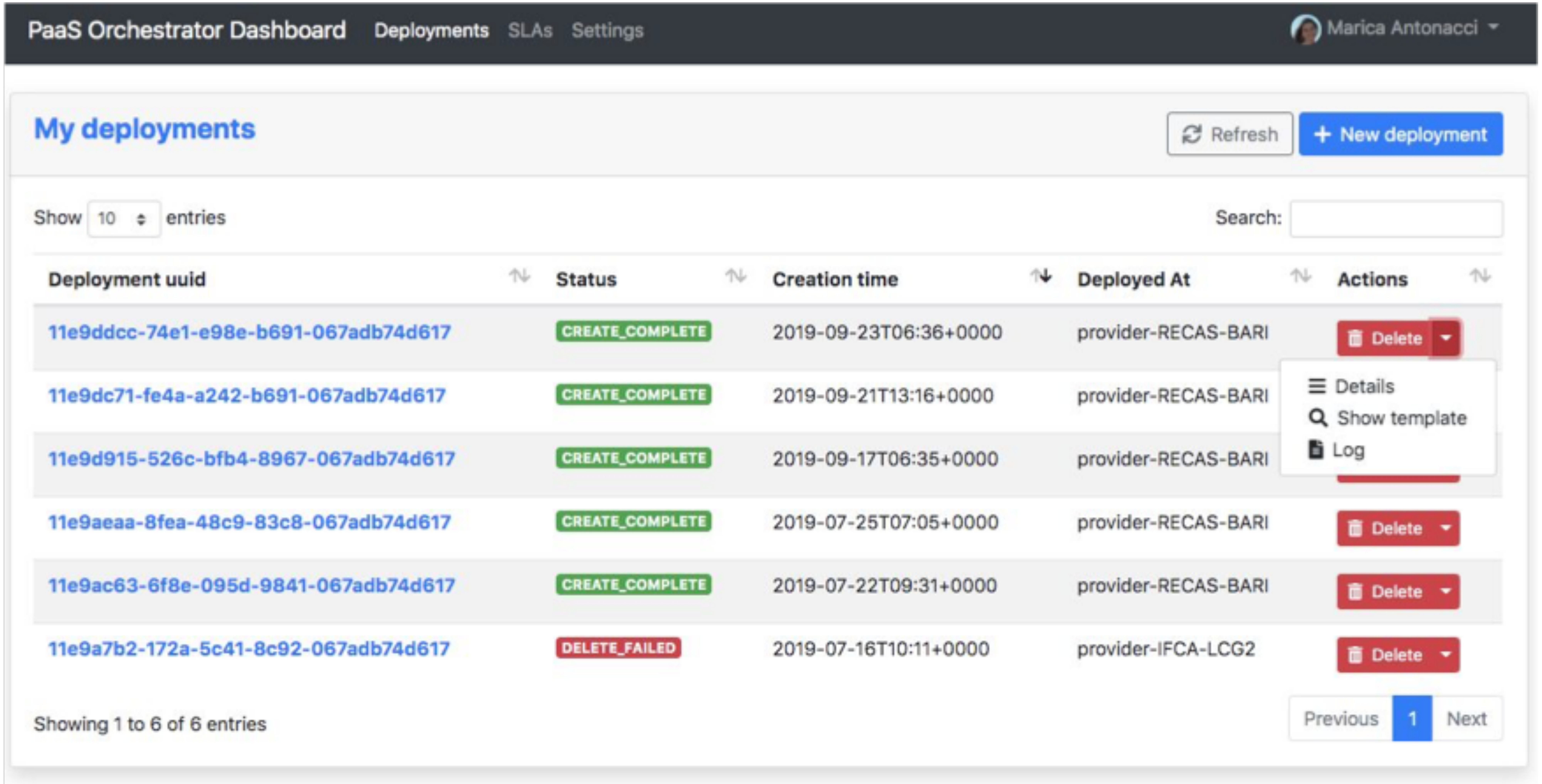
1 GB

ram memory required for the instance

3 Submit · Cancel

Submit deployment request

List your deployments

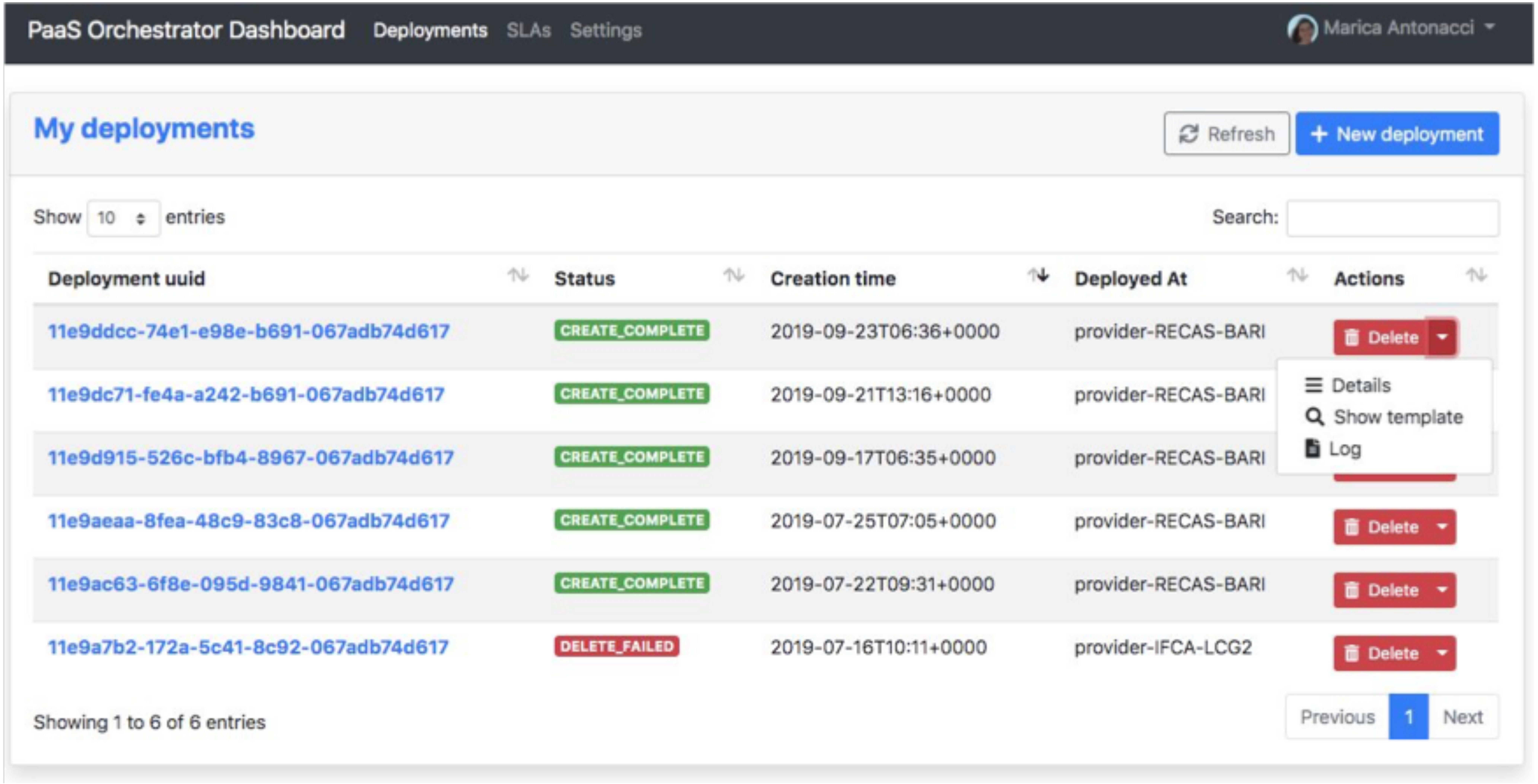


My deployments

Deployment uuid	Status	Creation time	Deployed At	Actions
11e9ddcc-74e1-e98e-b691-067adb74d617	CREATE_COMPLETE	2019-09-23T06:36+0000	provider-RECAS-BARI	Delete
11e9dc71-fe4a-a242-b691-067adb74d617	CREATE_COMPLETE	2019-09-21T13:16+0000	provider-RECAS-BARI	
11e9d915-526c-bfb4-8967-067adb74d617	CREATE_COMPLETE	2019-09-17T06:35+0000	provider-RECAS-BARI	
11e9aeaa-8fea-48c9-83c8-067adb74d617	CREATE_COMPLETE	2019-07-25T07:05+0000	provider-RECAS-BARI	Delete
11e9ac63-6f8e-095d-9841-067adb74d617	CREATE_COMPLETE	2019-07-22T09:31+0000	provider-RECAS-BARI	Delete
11e9a7b2-172a-5c41-8c92-067adb74d617	DELETE_FAILED	2019-07-16T10:11+0000	provider-IFCA-LCG2	Delete

Showing 1 to 6 of 6 entries

Get deployments details and outputs

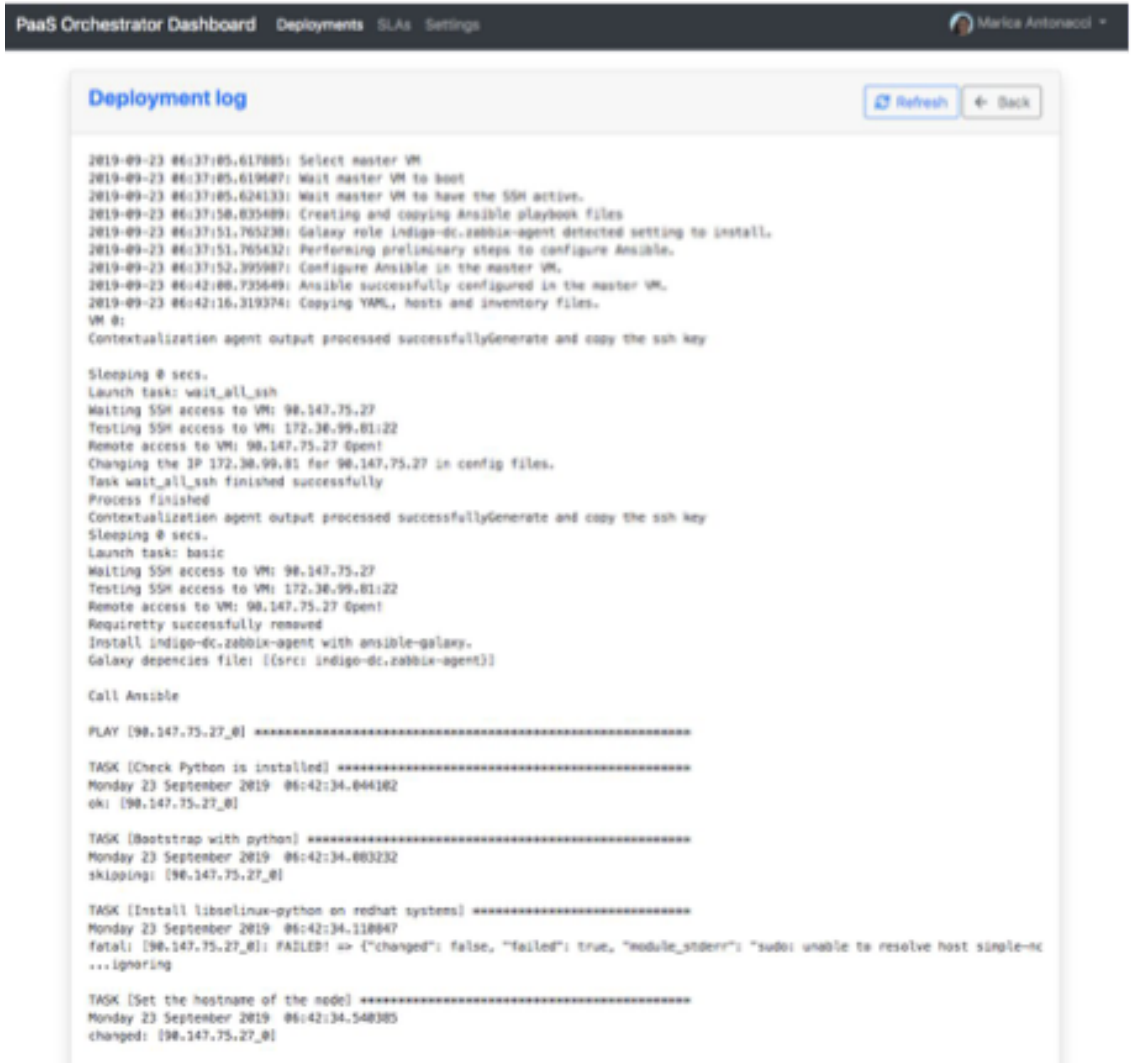


My deployments

Deployment uuid	Status	Creation time	Deployed At	Actions
11e9ddcc-74e1-e98e-b691-067adb74d617	CREATE_COMPLETE	2019-09-23T06:36+0000	provider-RECAS-BARI	Delete
11e9dc71-fe4a-a242-b691-067adb74d617	CREATE_COMPLETE	2019-09-21T13:16+0000	provider-RECAS-BARI	
11e9d915-526c-bfb4-8967-067adb74d617	CREATE_COMPLETE	2019-09-17T06:35+0000	provider-RECAS-BARI	
11e9aeaa-8fea-48c9-83c8-067adb74d617	CREATE_COMPLETE	2019-07-25T07:05+0000	provider-RECAS-BARI	Delete
11e9ac63-6f8e-095d-9841-067adb74d617	CREATE_COMPLETE	2019-07-22T09:31+0000	provider-RECAS-BARI	Delete
11e9a7b2-172a-5c41-8c92-067adb74d617	DELETE_FAILED	2019-07-16T10:11+0000	provider-IFCA-LCG2	Delete

Showing 1 to 6 of 6 entries

View the deployment log



Deployment log

```
2019-09-23 06:37:05.617885: Select master VM
2019-09-23 06:37:05.619607: Wait master VM to boot
2019-09-23 06:37:05.624133: Wait master VM to have the SSH active.
2019-09-23 06:37:50.835409: Creating and copying Ansible playbook files
2019-09-23 06:37:51.765230: Galaxy role indigo-dc.zabbix-agent detected setting to install.
2019-09-23 06:37:51.765432: Performing preliminary steps to configure Ansible.
2019-09-23 06:37:52.395987: Configure Ansible in the master VM.
2019-09-23 06:42:00.735649: Ansible successfully configured in the master VM.
2019-09-23 06:42:16.319374: Copying YAML, hosts and inventory files.
VM 0:
Contextualization agent output processed successfullyGenerate and copy the ssh key

Sleeping 0 secs.
Launch task: wait_all_ssh
Waiting SSH access to VM: 90.147.75.27
Testing SSH access to VM: 172.30.99.81:22
Remote access to VM: 90.147.75.27 Open!
Changing the IP 172.30.99.81 for 90.147.75.27 in config files.
Task wait_all_ssh finished successfully
Process finished
Contextualization agent output processed successfullyGenerate and copy the ssh key
Sleeping 0 secs.
Launch task: basic
Waiting SSH access to VM: 90.147.75.27
Testing SSH access to VM: 172.30.99.81:22
Remote access to VM: 90.147.75.27 Open!
Requiretty successfully removed
Install indigo-dc.zabbix-agent with ansible-galaxy.
Galaxy depencies file: [{src: indigo-dc.zabbix-agent}]

Call Ansible

PLAY [90.147.75.27_0] ********************************************************

TASK [Check Python is installed] *********************************************
Monday 23 September 2019  06:42:34.044102
ok: [90.147.75.27_0]

TASK [Bootstrap with python] *************************************************
Monday 23 September 2019  06:42:34.083232
skipping: [90.147.75.27_0]

TASK [Install libselinux-python on redhat systems] ***************************
Monday 23 September 2019  06:42:34.110047
fatal: [90.147.75.27_0]: FAILED! => {"changed": false, "failed": true, "module_stderr": "sudo: unable to resolve host simple-nc
...ignoring

TASK [Set the hostname of the node] ******************************************
Monday 23 September 2019  06:42:34.540385
changed: [90.147.75.27_0]
```

Useful for debugging purposes.

You can also download the log file clicking on the ‘download’ button at the end of the page.

INDIGO Resources

- https://github.com/indigo-dc/tosca-templates
- https://github.com/indigo-dc/tosca-types

- https://galaxy.ansible.com/indigo-dc/

- https://hub.docker.com/u/indigodatacloud/dashboard/

References

- **TOSCA Simple Profile in YAML Version 1**
  http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.0/csprd02/TOSCA-Simple-Profile-YAML-v1.0-csprd02.html

- **Cloud Portability, Lifecycle Management and more**
  https://www.slideshare.net/CloudStandardsCustomerCouncil/oasis-tosca-cloud-portability-and-lifecycle-management

Thank you!

Hands-on Outline

- Goal: submit a simple TOSCA template to the PaaS Orchestrator
- The template describes an Elasticsearch + Kibana instance
  - Can be an all-in-one installation (single VM) or
  - Can be deployed on two different machines
  - A dependency must be declared between the elasticsearch (SW Component) node and the kibana (SW Component) node

https://maricaantonacci.github.io/TOSCA-Hands-on/