How to orchestrate services in the EOSC Compute Platform with the INDIGO PaaS

Marica Antonacci, Giacinto Donvito
INFN - National Institute for Nuclear Physics (Italy)

EGI Webinar

27 October 2021

Outline

- INDIGO PaaS Overview
- What is TOSCA
- Orchestrator high level architecture
- PaaS main features
- User interfaces
- Live demo

INDIGO PaaS origins

The development of the INDIGO PaaS started during the **European H2020 project** "**INDIGO-DataCloud**" aimed to develop an open source Cloud platform for computing and data ("DataCloud") tailored to science but applicable to other domains as well

- 26 partners coordinated by INFN (11.1M€, 30 months - April 2015 to September 2017)

and continued during the two spin-off projects DEEP-Hybrid DataCloud, eXtreme-DataCloud and EOSC-Hub (2017-2020)

- Evolving the functionalities to **TRL8**
- Ensuring the scalability and performance of the developed solutions
- Providing relevant contributions to the **European Open Science Cloud**

Current exploitation and usage

Istituto Nazionale di Fisica Nucleare

Several on-going projects rely on the INDIGO PaaS as federation and orchestration tool: EGI-ACE, C-SCALE, IoTwins, ...

It is also used in production services like

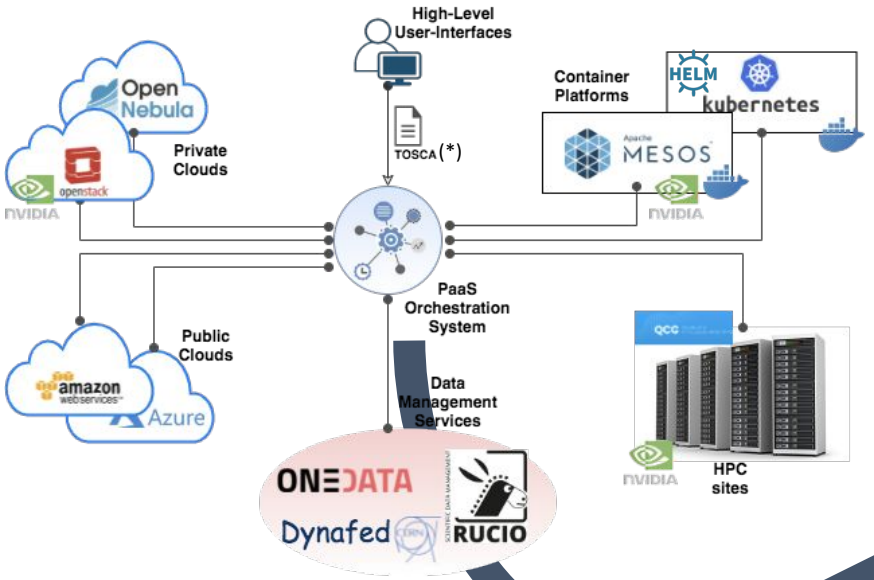
- **Laniakea@ReCaS**, the service that allows to perform the automatic deployment of virtual Galaxy environments for life science (Elixir-Italy)
- **DODAS (Dynamic On Demand Analysis Service)**, a service for generating an on-demand container-based HTCondor cluster
- The **INFN Cloud Infrastructure** is exploiting the INDIGO PaaS capabilities for federating the resources provided by the two large data centers of CNAF and Bari, and other satellite sites.

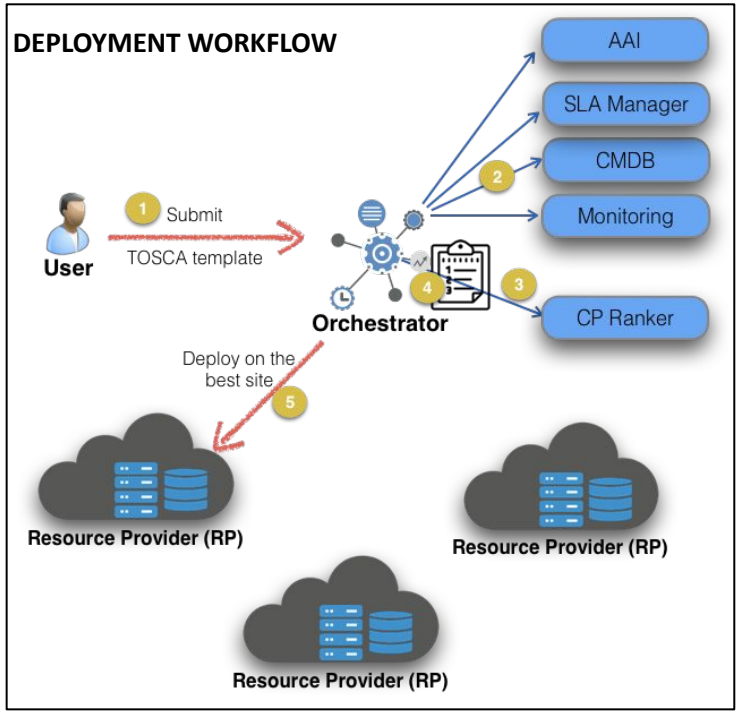
Foundations and key enablers

INFN CLOUD
Istituto Nazionale di Fisica Nucleare

- Develop and integrate **open-source** and **open standard-based** components to ensure portability and interoperability
- Adopt a **modular micro-service** architecture
- Use **Infrastructure as Code** (IaC), DevOps and **containers** to reduce manual processes and increase flexibility and portability across environments
- Leverage **federated Authentication and Authorization** technologies based on **OpenID-Connect** (supporting also legacy AAI solutions) like **INDIGO-IAM** and EGI Check-in
- Enable flexible **service composition** and re-use

INDIGO PaaS Orchestration System
High-level architecture





(*) Topology and Orchestration Specification for Cloud Applications Ref: TOSCA Simple Profile in YAML Version 1.1

The Orchestrator interacts with the provider services through:

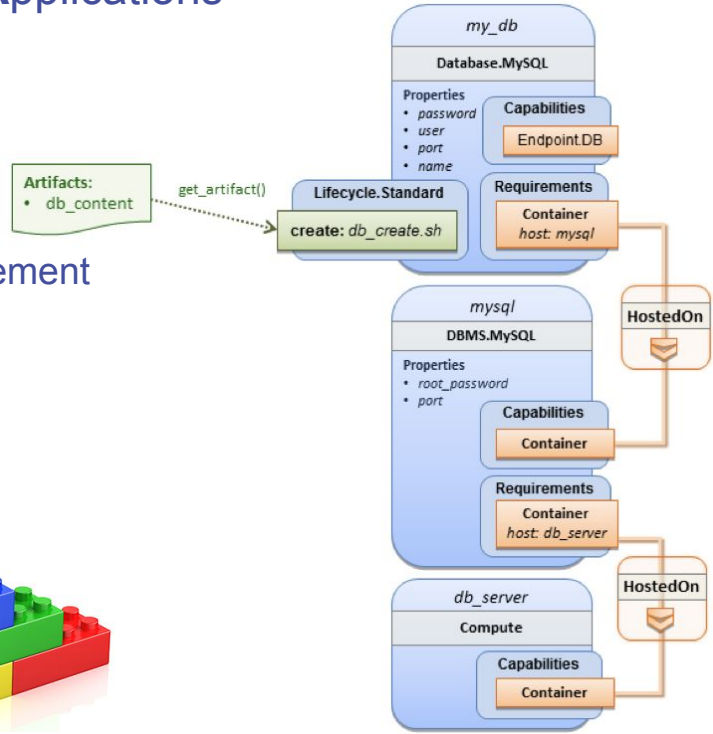
- the **Infrastructure Manager** for deploying complex and customized virtual infrastructures on multiple IaaS Cloud backends (Openstack, AWS, etc.)
- direct APIs for deploying dockerized workloads on container platforms or submitting HPC jobs

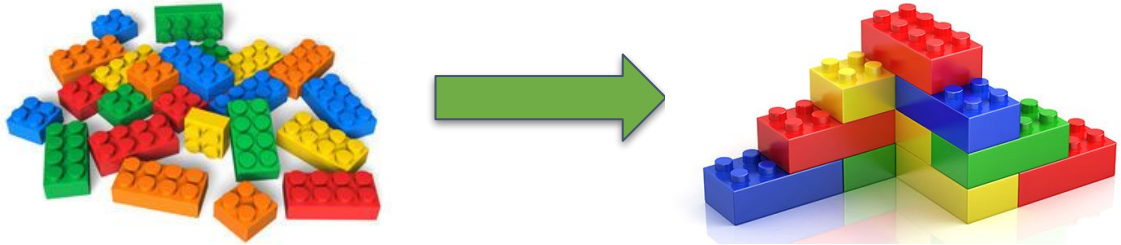
TOSCA

INFN CLOUD
Istituto Nazionale di Fisica Nucleare

Topology and **O**rchestration **S**pecification for **C**loud **A**pplications

- Goals:
  - Automated Application Deployment and Management
  - Portability of Application Descriptions and Their Management
  - Interoperability and Reusability of Components





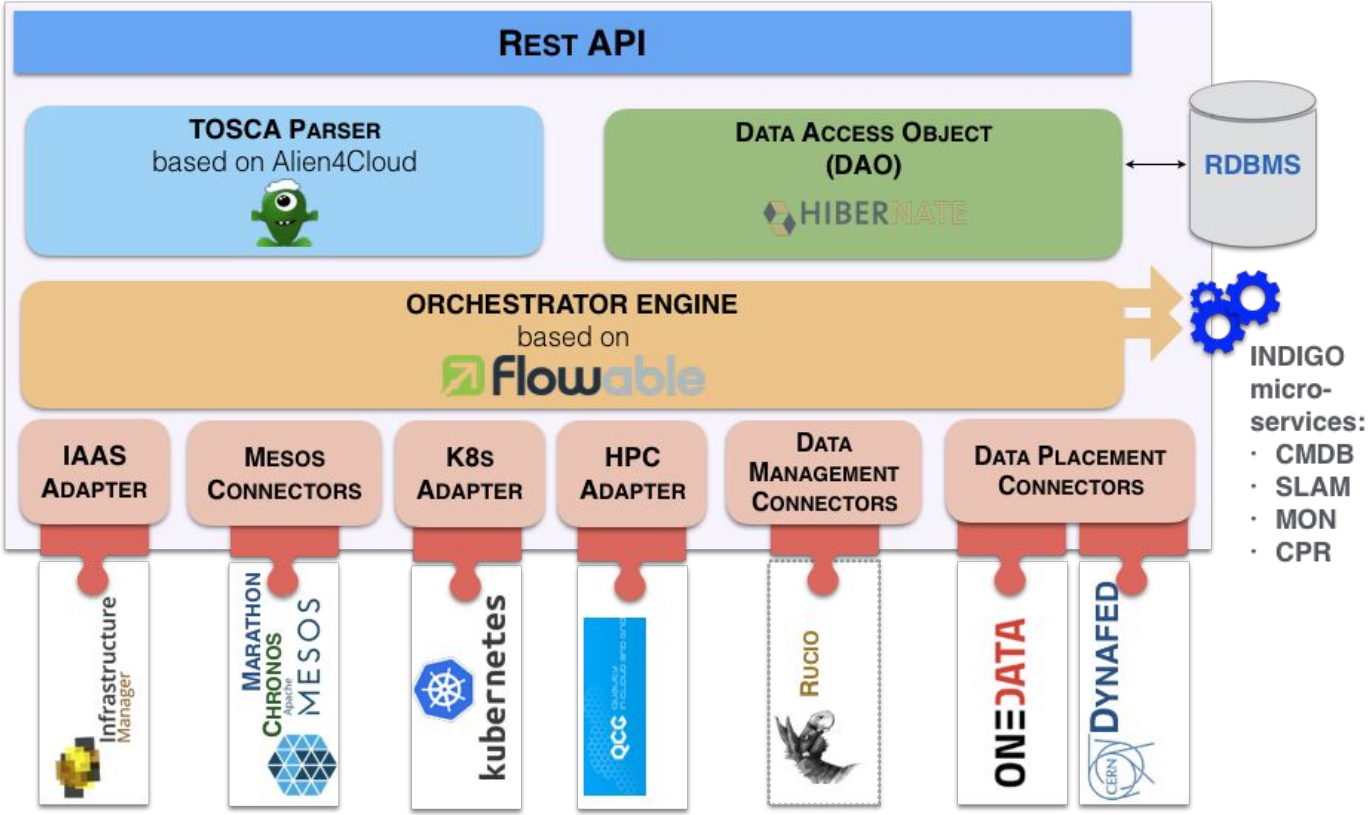
Ref: TOSCA Simple Profile in YAML Version 1.1

Template examples

```
tosca_definitions_version: tosca_simple_yaml_1_0_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: 4 MB
            disk_size: 10 GB

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

```
tosca_definitions_version: tosca_simple_yaml_1_0_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
```

PaaS Orchestrator high-level architecture



INDIGO PaaS - Main features

- **Enabling the federation of distributed and heterogeneous compute environments: clouds, docker orchestration platforms, HPC systems**
  - Further integrations can be easily included implementing new plugins (adapters)
- **Smart scheduling → Automatic selection of the provider**
  - First level based on the deployment type (cloud, Mesos, Kubernetes, HPC)
  - Second level based on SLAs and compute/storage requirements vs provider capabilities
    - Support for specialized hardware (GPU, Infiniband)
    - Data location
  - Best provider selection based on:
    - Resource quotas (SLA)
    - Monitoring data
  - Automatic retry of failed deployments
- **Support for hybrid deployments and network orchestration**
- **Client interfaces for advanced users** (REST APIs, CLI, python bindings) **and end-users** (web dashboard - no skills required)

Supported TOSCA types and templates

The INDIGO PaaS Orchestrator supports **TOSCA Simple Profile in YAML** (version 1.0)

Custom types have been developed during the INDIGO/DEEP/XDC/EOSC-Hub/… projects and are available on github: https://github.com/indigo-dc/tosca-types

Artifacts are mainly **ansible roles** and **docker images**.

Specific custom types have been implemented for modeling

- dockerized services/jobs to be deployed on Mesos clusters
- HPC jobs to be submitted through QCG Gateways
- Helm charts to be installed on Kubernetes clusters

Ready-to-use templates:

From simple to complex:

- Virtual machine provisioning (with or without additional block storage)
- Docker-based use-cases
- Cluster-based use-cases (Mesos, K8s, Spark, Galaxy)
- Applications (SaaS): Rstudio, Jupyter, Tensorflow, etc.

New services (i.e. TOSCA templates) can be easily implemented and included in the service catalogue

Orchestrator APIs

INFN CLOUD
Istituto Nazionale di Fisica Nucleare

- **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

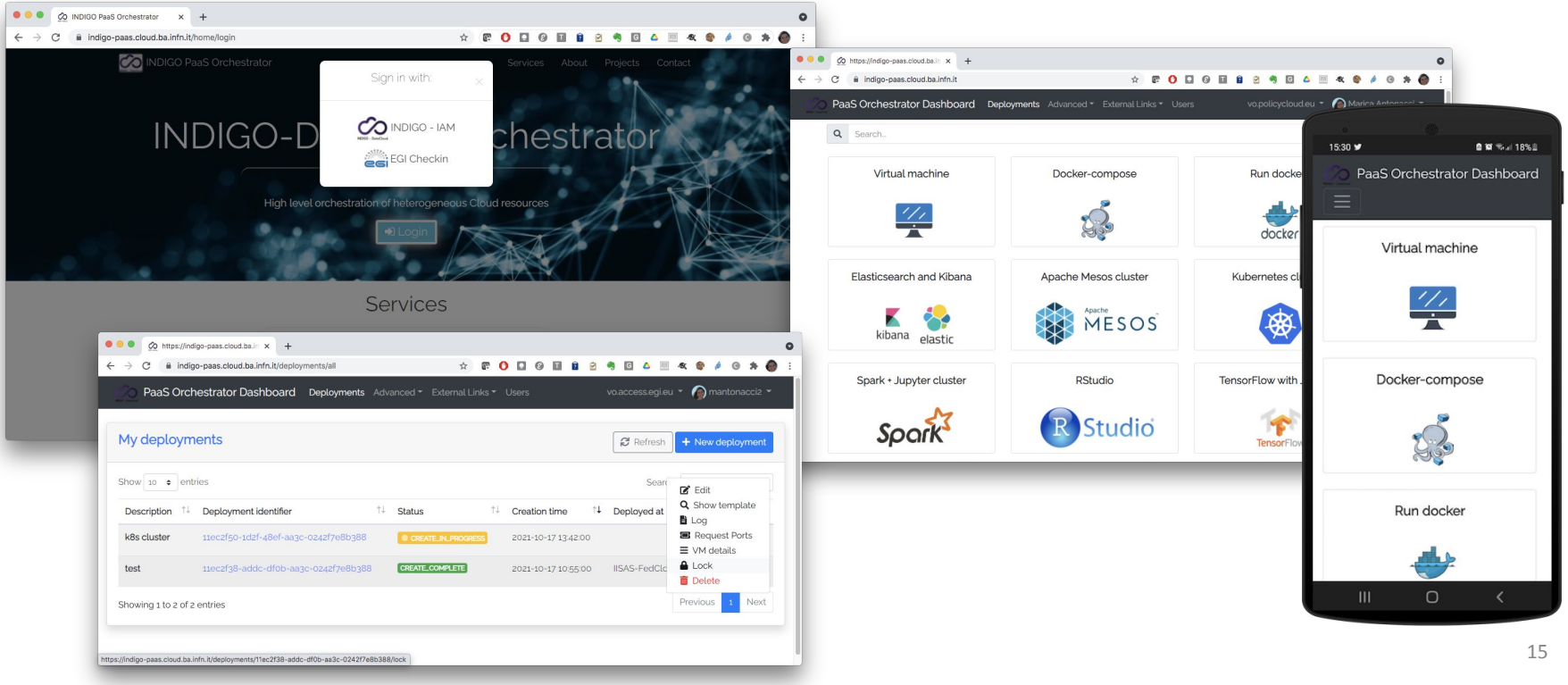
Orchent: The Orchestrator CLI

```
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
```

Installation guide:
https://indigo-dc.gitbooks.io/orchent/content/admin.html
User guide:
https://indigo-dc.gitbooks.io/orchent/content/user.html

The PaaS Orchestrator dashboard

Empower users lowering the access barriers

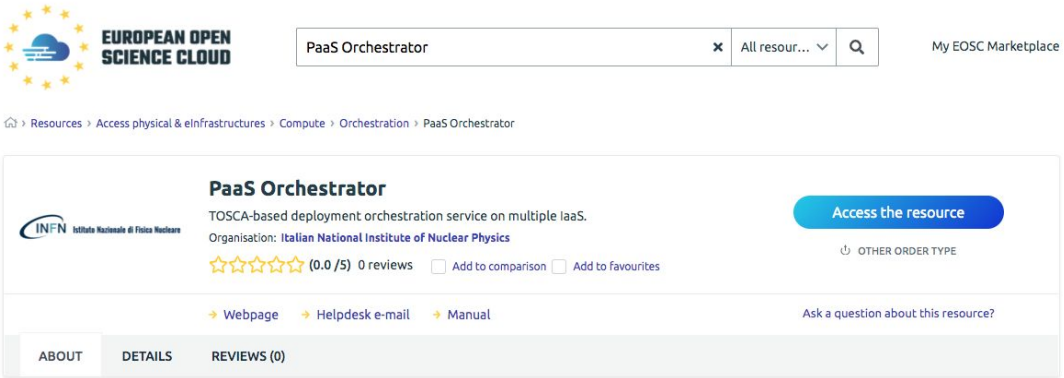


Live demonstration about how the INDIGO PaaS works:

- deploying a complete kubernetes cluster on a cloud provider (Openstack)
- deploying a Tensorflow docker container with GPU access

Thank you for your attention!

https://marketplace.eosc-portal.eu/services/paas-orchestrator



For more information and technical support mail to: indigo-paas-support@lists.infn.it

Backup slides

Deployment retry strategy

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

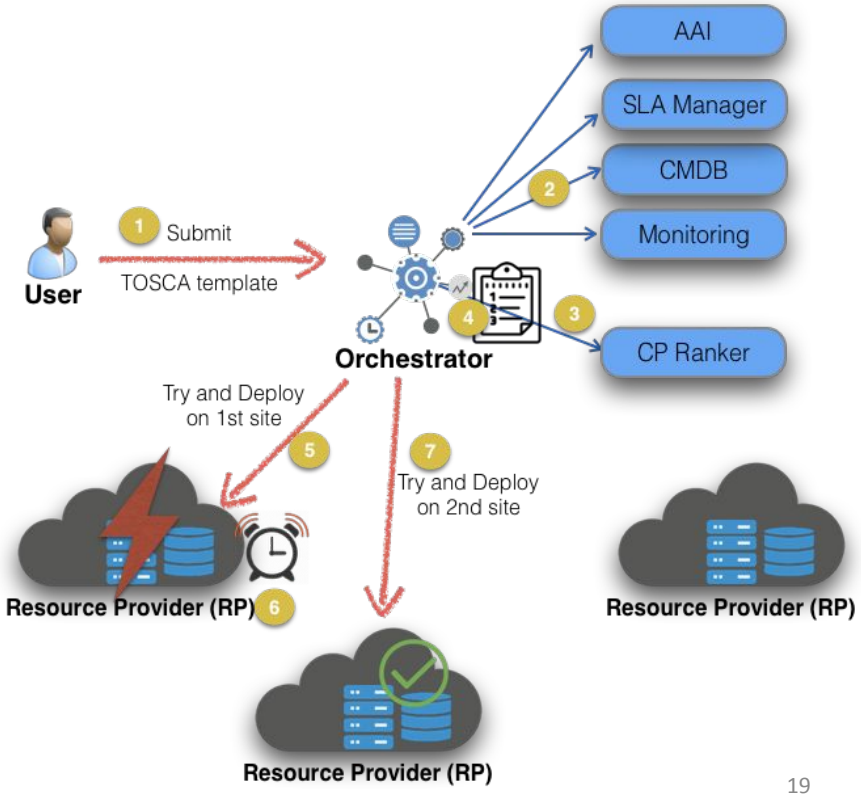
Example: the deployment fails due to a runtime error on the chosen site.

The implemented mechanism is able to address also the **timeout** in the deployment creation.

The user can specify

○ the maximum time for the single trial at each provider;

○ the overall maximum time for the deployment creation (including the possible retries).

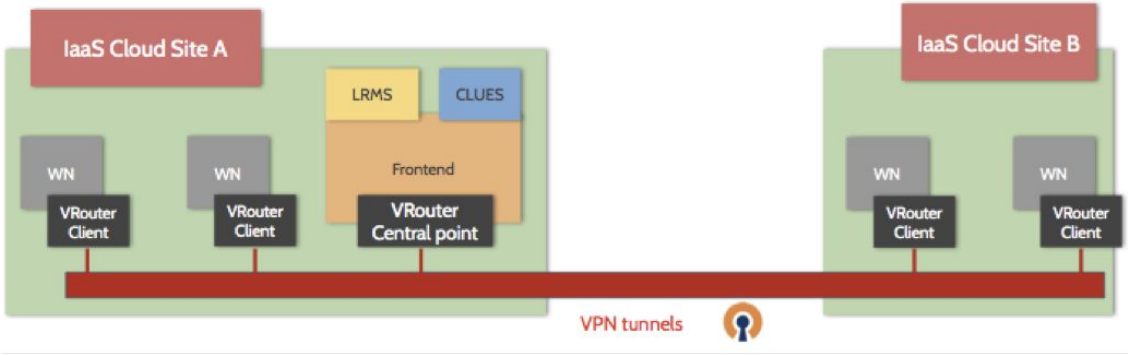


Virtual Networking Orchestration

INFN CLOUD
Istituto Nazionale di Fisica Nucleare

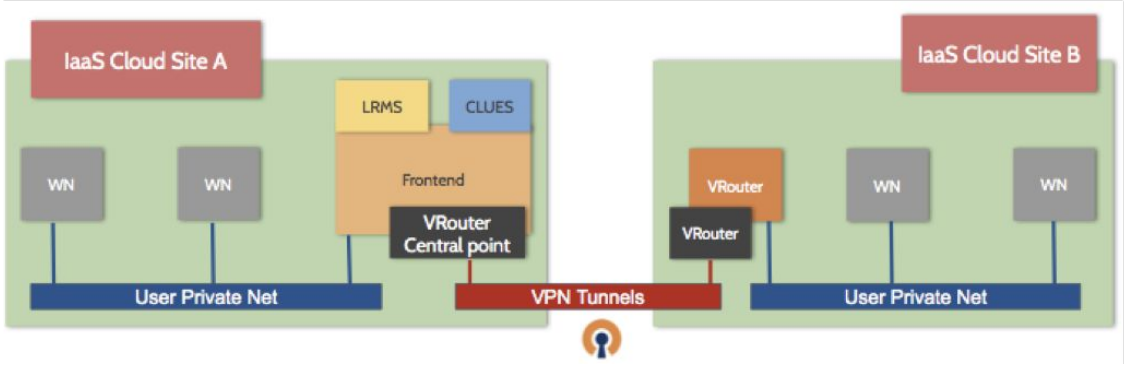
Scenario I:

exploits private networks already existing at the sites



Scenario II:

a dedicated private network is created for the deployment in both sites

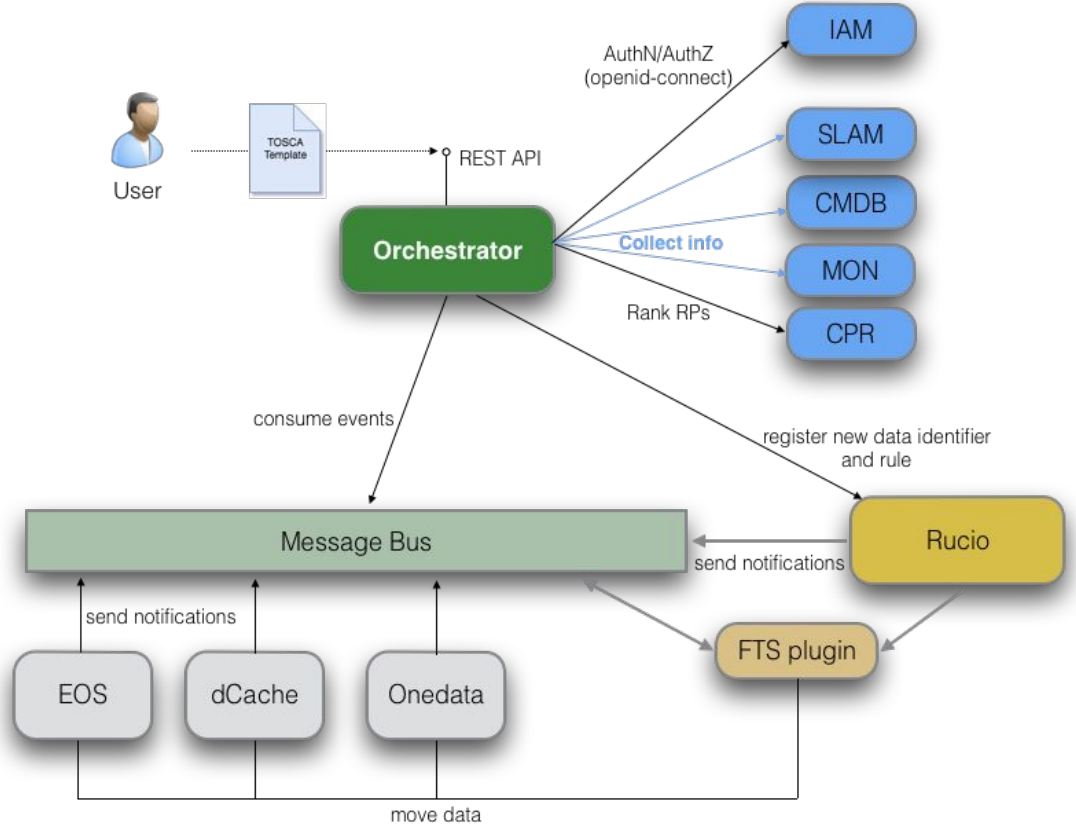


Data Placement and Orchestration

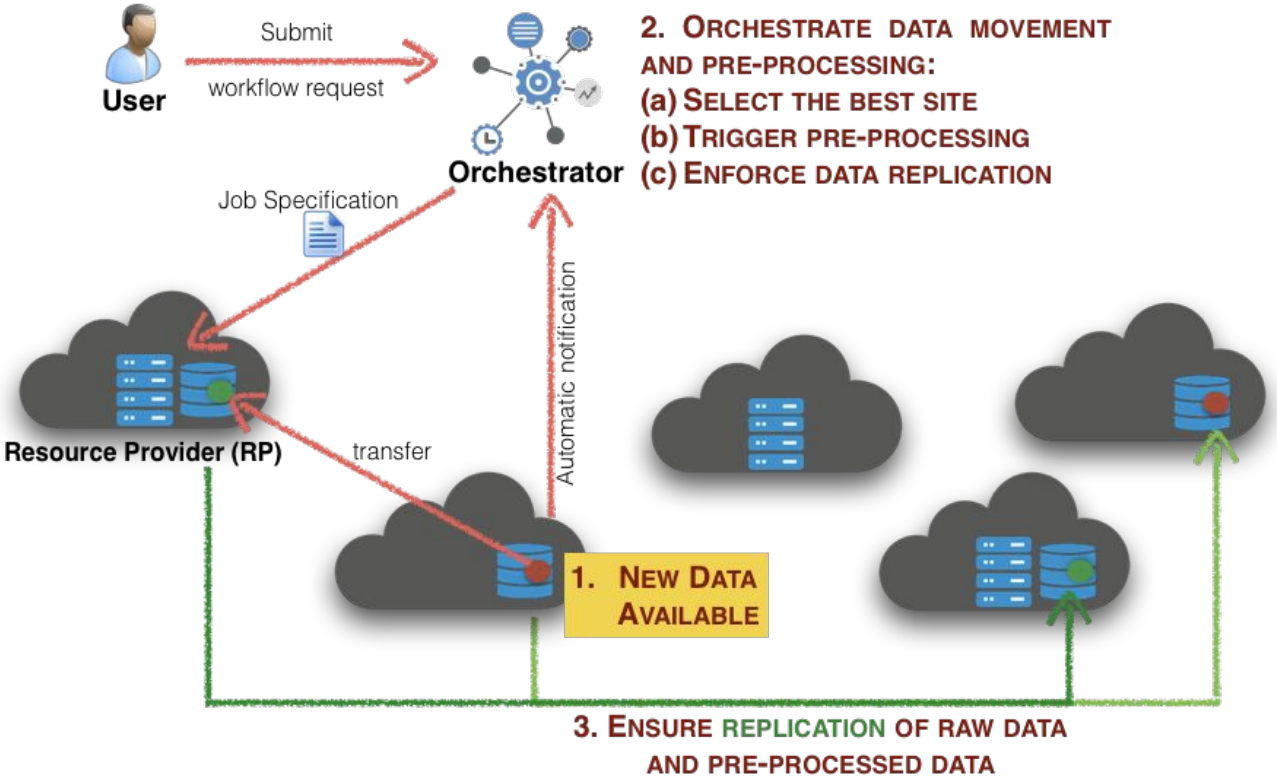
- **Data-aware scheduling**: the INDIGO Orchestrator is able to select automatically the best computing site based on the user data location
  - Data placement plugins available for Onedata and Dynafed

- **Data movement and orchestration**: Rucio, the data management system developed by Cern (initially for the ATLAS experiment), has been integrated in the INDIGO Orchestrator as a plugin to be used to steer the data movement
  - support workflows for data pre-processing at ingestion

Data Orchestration with Rucio



Use-case: Data Pre-processing at ingestion



Resources

- INDIGO-DataCloud: a Platform to Facilitate Seamless Access to E-Infrastructures: https://link.springer.com/content/pdf/10.1007/s10723-018-9453-3.pdf
- PaaS Orchestrator docs: https://indigo-dc.gitbook.io/indigo-paas-orchestrator/
- PaaS Orchestrator API docs: https://indigo-dc.github.io/orchestrator/
- PaaS Orchestrator code repo: https://github.com/indigo-dc/orchestrator
- INDIGO TOSCA types:
  - https://github.com/indigo-dc/tosca-types
  - Example templates: https://github.com/indigo-dc/tosca-types/tree/master/examples
- INDIGO Ansible Roles: https://galaxy.ansible.com/indigo-dc/
- Docker images: https://hub.docker.com/u/indigodatacloud/