What’s new in the INDIGO PaaS

Marica Antonacci (INFN BA)

Giacinto Donvito (INFN BA), Michele Perniola (INFN BA), Andrea Ceccanti (CNAF)

CCR Workshop 2021
May 24-28 - Online

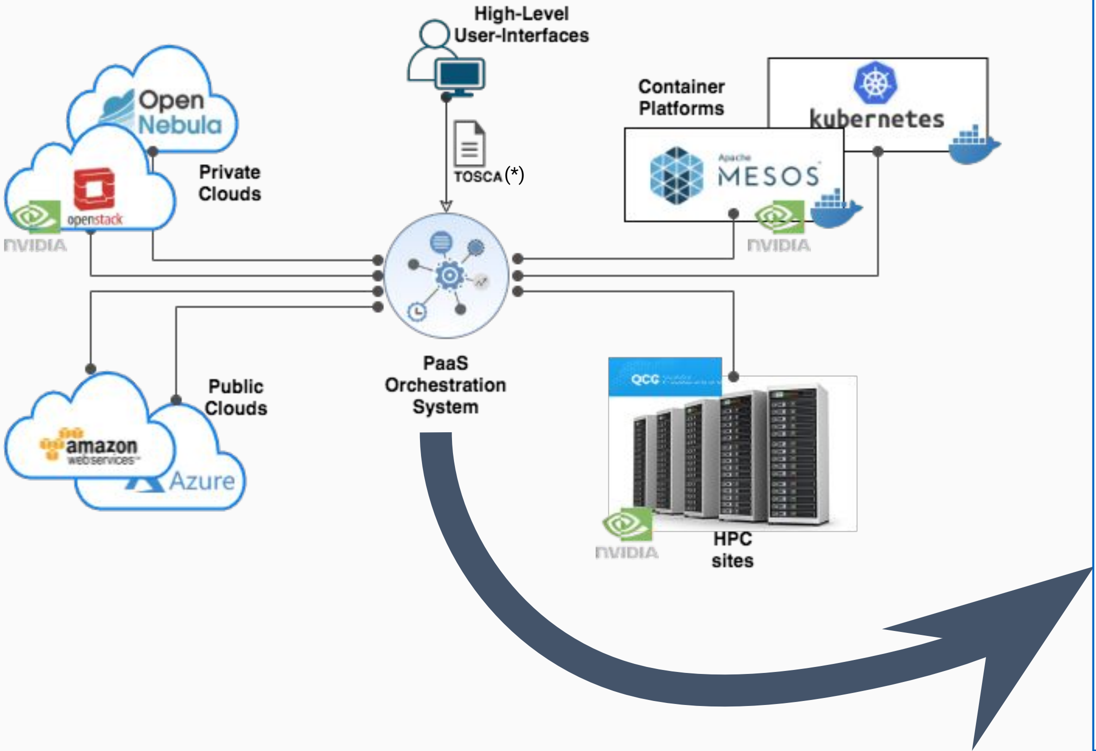
Outline

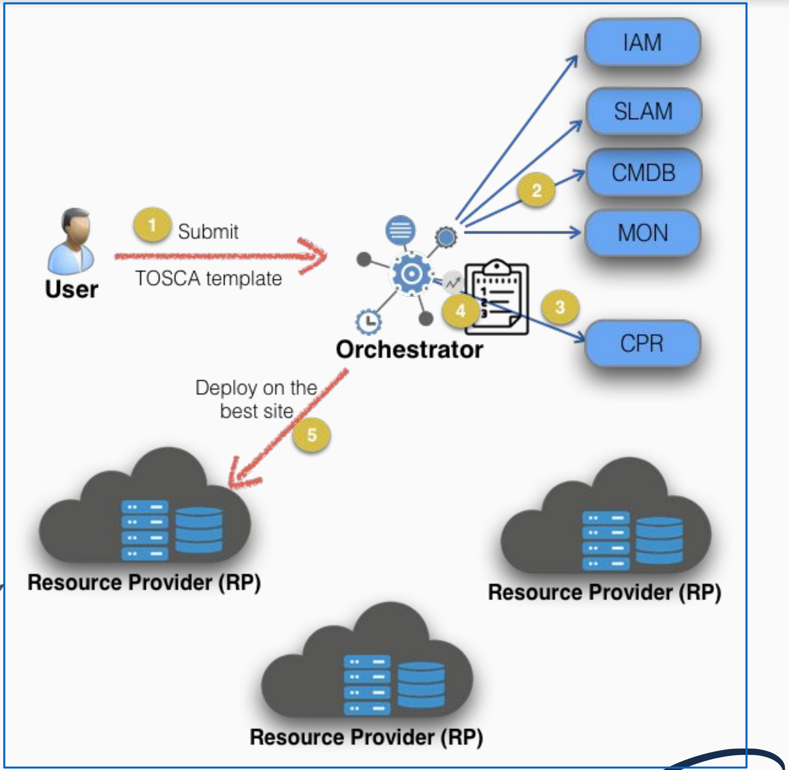
- ❑ INDIGO PaaS Overview
- ❑ Orchestrator architecture
- ❑ New functionalities
- ❑ Work in progress
- ❑ Conclusions

INDIGO PaaS Overview

- ❑ 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 and Kubernetes clusters.
- ❑ The development started during the European H2020 project "**INDIGO-DataCloud**" and continued during the following projects DEEP-Hybrid DataCloud, eXtreme-DataCloud and EOSC-Hub
  - ❍ Evolving the functionalities to **TRL8**
  - ❍ Ensuring the **scalability** and **performance** of the developed solutions
  - ❍ Providing relevant contributions to the **EOSC**
- ❑ **Further improvements are being designed and implemented in the framework of the INFN Cloud project**

PaaS Orchestration System (from 10Km)





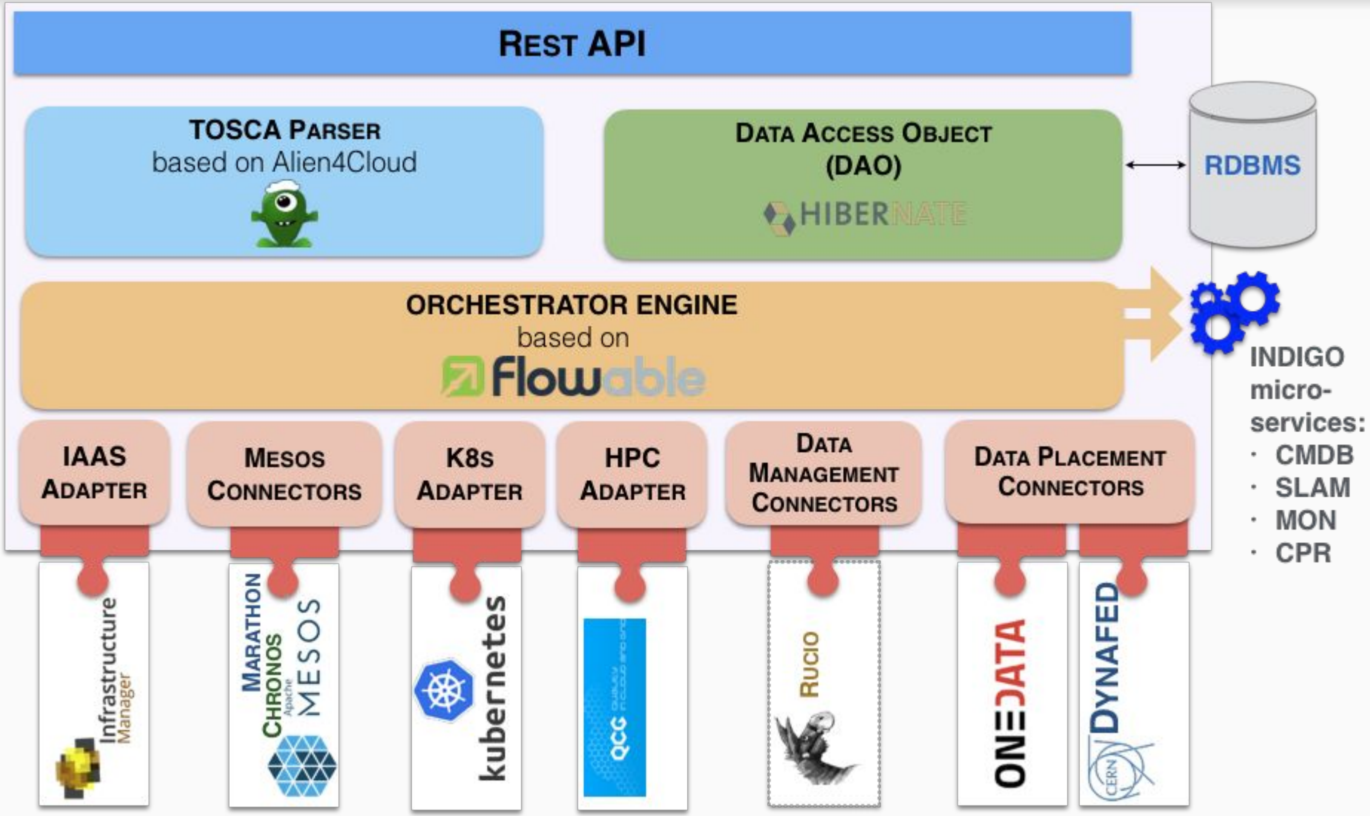
(*) **T**opology and **O**rchestration **S**pecification for **C**loud **A**pplications

Ref: TOSCA Simple Profile in YAML Version 1.1

Major improvements after INDIGO

- ❑ **New Workflow Manager based on Flowable (Orchestrator)**
  - ❍ More reliable and faster than the old engine based on jBPM
- ❑ **Enhanced TOSCA Parser based on Alien4Cloud (Orchestrator)**
  - ❍ Allows the dynamic import of the TOSCA types
- ❑ **Enhanced Information System (CMDB+CIP)**
  - ❍ New data hierarchy including “tenants”
- ❑ **Improved Monitoring zabbix-based probes - data reorganized in a coherent structure + status page for PaaS services**
- ❑ **Re-engineered Cloud Provider Ranker (CPR)**
- ❑ **Support for specialised hardware (GPUs, Infiniband)**
- ❑ **Support for hybrid deployments and network orchestration**
- ❑ **New plugins (HPC and Kubernetes) - prototype**
- ❑ **Integration with Rucio data orchestration - prototype**
- ❑ **Brand new Orchestrator dashboard**

PaaS Orchestrator high-level architecture



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.

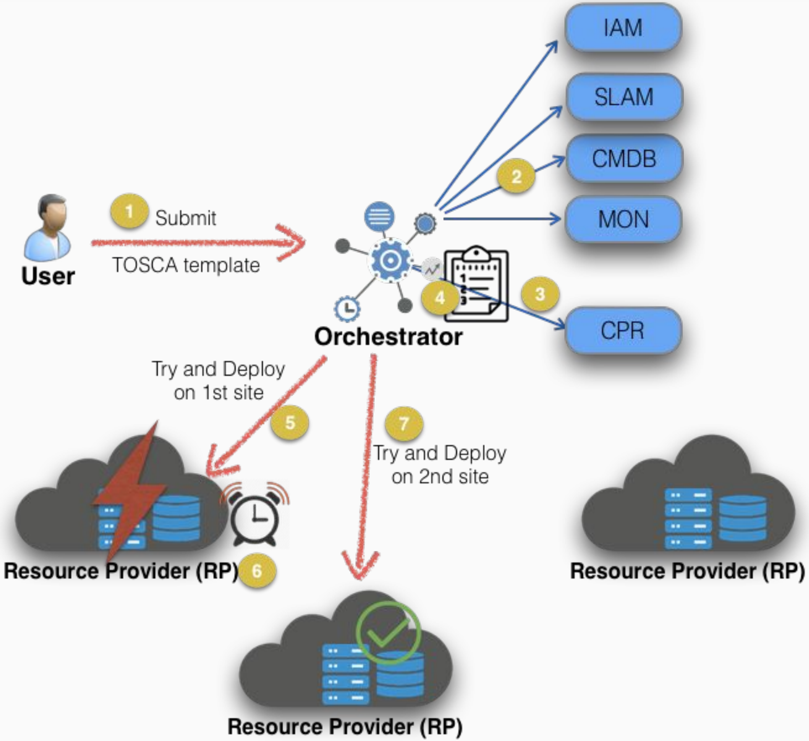
Example: the deployment fails due to a runtime error on the chosen site

The 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).



Support for specialized hardware requirements

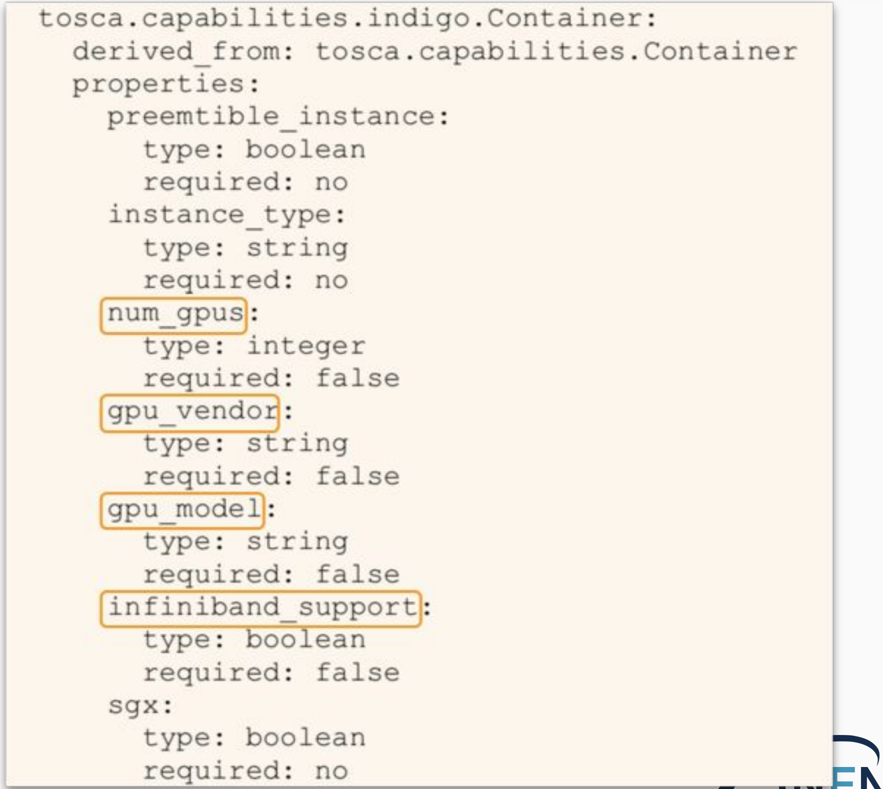
The PaaS layer allows to federate and access specialized hardware resources, mainly **GPUs** and **infiniband**, using high level interfaces.

Both complex clusters of virtual machines and containers can be allocated to sites providing specialized computing hardware devices needed for running time consuming workloads, like the deep learning applications.

- ❑ The information system (Cloud Info Provider + CMDB) has been extended in order to collect information about the availability of GPUs and infiniband support at the sites
  - ❍ This information is read and consumed by the Orchestrator to select the best site where the resources will be allocated
- ❑ The TOSCA model for compute and container nodes has been extended in order to include these requirements

Support for specialized hardware requirements (2)

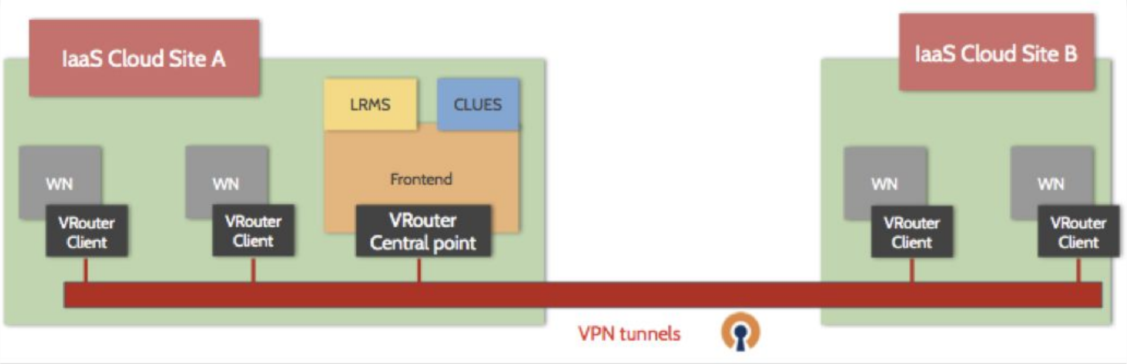
The Orchestrator has been enhanced in order to match the TOSCA requirements with the available resources at the sites (improved ranking algorithm and scheduling), thanks to the fine-grained description provided by the Information System



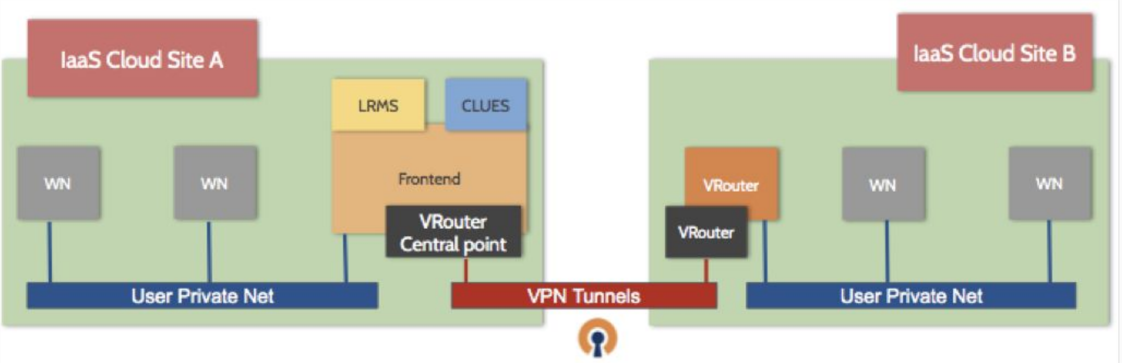
```
tosca.capabilities.indigo.Container:
  derived_from: tosca.capabilities.Container
  properties:
    preemtible_instance:
      type: boolean
      required: no
    instance_type:
      type: string
      required: no
    num_gpus:
      type: integer
      required: false
    gpu_vendor:
      type: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

Virtual Networking Orchestration

Scenario I:
exploits private networks already existing at the sites



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



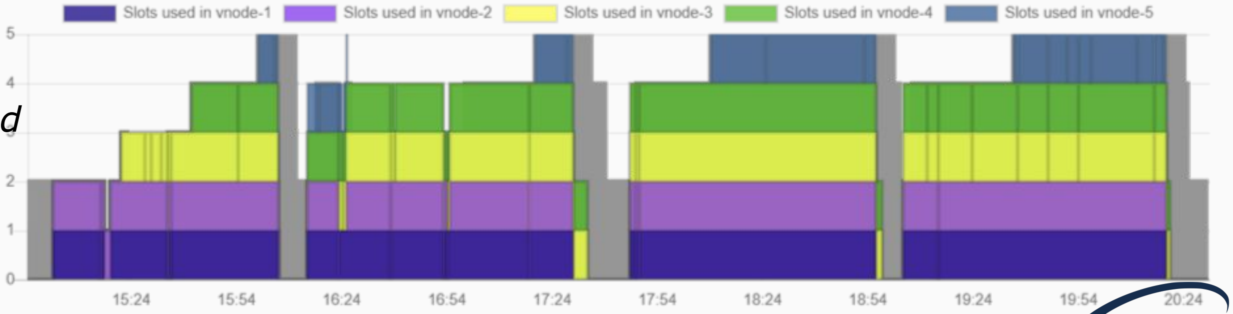
Autoscaling clusters with L2 isolated networks

Slurm cluster deployed on CESNET cloud and AWS

- ❑ L2 networks created at both sites and interconnected through vRouter components
- ❑ CLUES (Elasticity Manager) used to add/remove nodes in the cluster depending on the workload

Reference Test: 3676 audio files from the Urban Sound Dataset classified with the Audio Classifier model from the DEEP Open Catalog using udocker

"Deployment of Elastic Virtual Hybrid Clusters Across Cloud Sites" published on Journal of Grid Computing



Cluster usage evolution

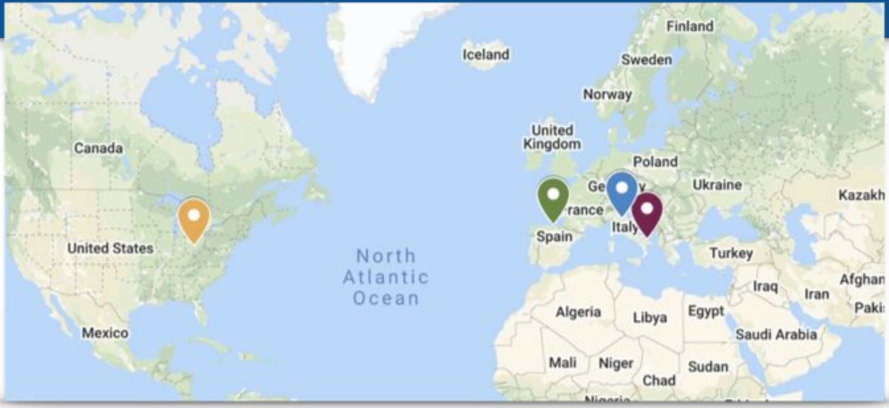
Scaling across multiple cloud providers

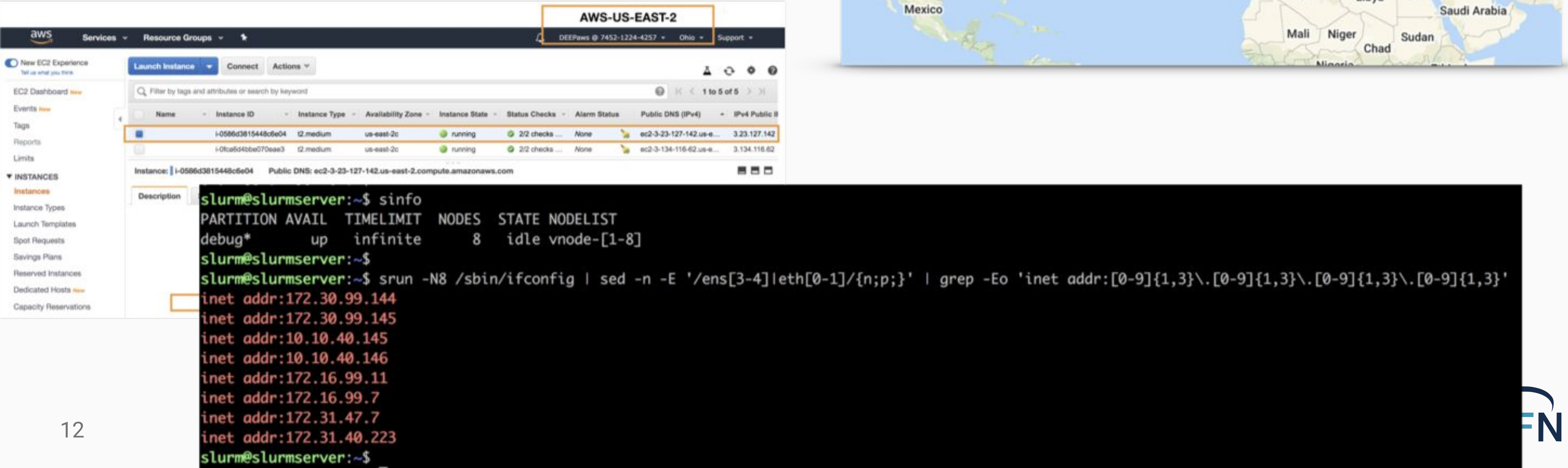
Reference test: Slurm cluster

with nodes on 4 sites

including AWS

- INFN-BARI
- INFN-CNAF
- IFCA-LCG2
- AWS-US-EAST-2 (Ohio)





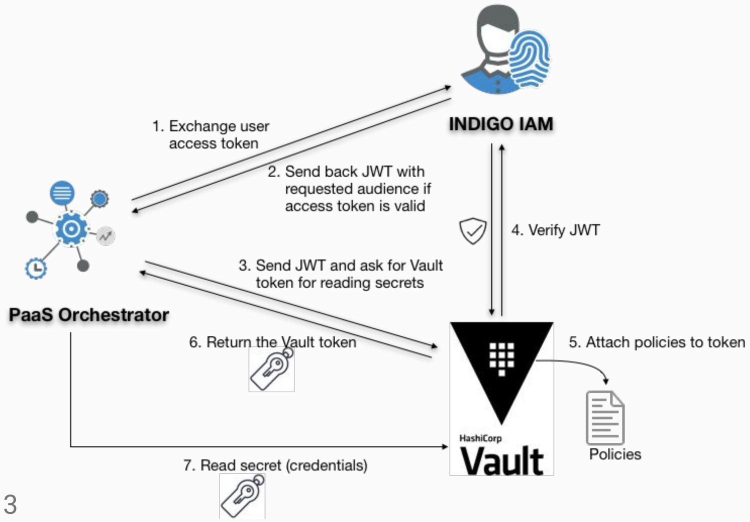
Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public I
	i-0586d3815448c6e04	t2.medium	us-east-2c	running	2/2 checks ...	None	ec2-3-23-127-142.us-e...	3.23.127.142
	i-0fca6d4bbe070eae3	t2.medium	us-east-2c	running	2/2 checks ...	None	ec2-3-134-116-62.us-e...	3.134.116.62

Instance: i-0586d3815448c6e04 Public DNS: ec2-3-23-127-142.us-east-2.compute.amazonaws.com

```
slurm@slurmserver:~$ sinfo
PARTITION AVAIL  TIMELIMIT  NODES  STATE NODELIST
debug*       up   infinite      8   idle vnode-[1-8]
slurm@slurmserver:~$
slurm@slurmserver:~$ srun -N8 /sbin/ifconfig | sed -n -E '/ens[3-4]|eth[0-1]/{n;p;}' | grep -Eo 'inet addr:[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}'
inet addr:172.30.99.144
inet addr:172.30.99.145
inet addr:10.10.40.145
inet addr:10.10.40.146
inet addr:172.16.99.11
inet addr:172.16.99.7
inet addr:172.31.47.7
inet addr:172.31.40.223
slurm@slurmserver:~$
```

Improved support for public clouds

- ❑ Integrated Hashicorp Vault for storing (public) cloud credentials
  - ❍ No need to provide the credentials in the TOSCA template (as done previously)
  - ❍ Authentication managed through IAM


INDIGO IAM

PaaS Orchestrator

1. Exchange user access token

2. Send back JWT with requested audience if access token is valid

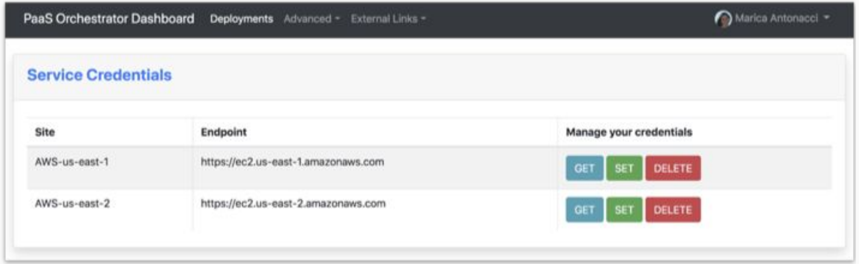
3. Send JWT and ask for Vault token for reading secrets

4. Verify JWT

5. Attach policies to token

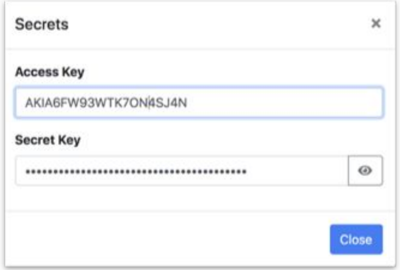
6. Return the Vault token

7. Read secret (credentials)



Service Credentials

Site	Endpoint	Manage your credentials
AWS-us-east-1	https://ec2.us-east-1.amazonaws.com	GET SET DELETE
AWS-us-east-2	https://ec2.us-east-2.amazonaws.com	GET SET DELETE


Secrets ×

Access Key: AKIA6FW93WTK7ON4SJ4N

Secret Key: ••

Support for multiple OIDC identity providers

The PaaS layer has been initially integrated only with the **INDIGO IAM** solution (developed during the same INDIGO-DataCloud project)

This integration remains the reference implementation for managing the users and the authentication/authorization flows throughout the whole stack (from PaaS to IaaS)

- ❑ Some details can be found in this presentation from the IAM Users Workshop (27-28 Jan 2021)

Anyway the PaaS has been adapted in order to support multiple and different OIDC Identity Providers, e.g. EGI Checkin

Multi-tenancy support

Initially all the authenticated users were considered part of the same group (organization) → flat structure →same roles, users mapped on the same IaaS project, etc.

A **finer-grained control** has been implemented later - based on the group membership attribute provided in the user oauth token:

- The Orchestrator uses this information to authorize the user to perform admin operations
- At IaaS level the user is mapped onto a dedicated project (Openstack tenant, Kubernetes namespace, etc.)
- Multiple groups membership is managed at the Orchestrator level (the user can specify the current “active group”)

Integration with HPC resources

Seamless access to HPC resources using cloud interfaces

- ❑ New TOSCA type to model the HPC Job
- ❑ New **adapter** implemented in the Orchestrator allows to submit jobs to **QCG gateways**
  - ❍ The user can submit and monitor his/her job through the Orchestrator

QCG-Computing is an open architecture implementation of SOAP Web service for multi-user access and policy-based job control routines by various queuing and batch systems managing local computational resources. This key service in QCG is using Distributed Resource Management Application API (DRMAA) to communicate with the underlying queuing systems.

Kubernetes provider integration

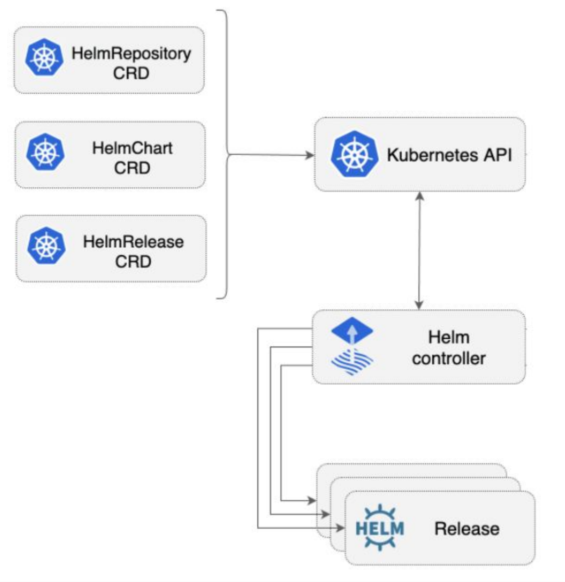
Integration requirements:

- The k8s cluster must support one of the OIDC IdP trusted by the PaaS
  - Fully tested with INDIGO IAM
- RBAC enabled and proper namespaces created
- FluxCD Helm operator installed

Deployment Workflow:

The Orchestrator interacts with the k8s APIs to create and monitor the helm release (through the helm operator)

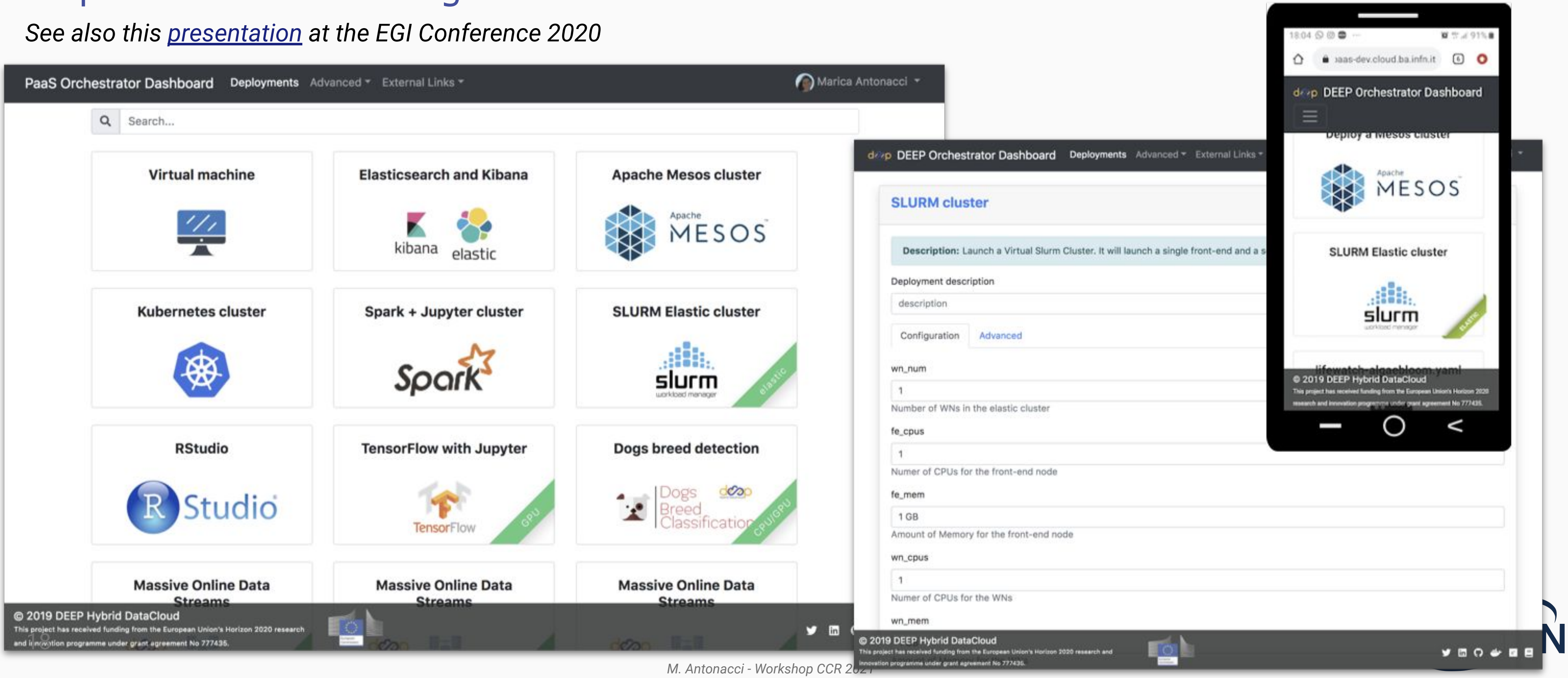
The deployment is created in the proper namespace (depending on the user group) and is tagged with a unique deployment id generated by the Orchestrator.



Improved high-level interfaces

Empower users lowering the access barriers

See also this presentation at the EGI Conference 2020



PaaS Orchestrator Dashboard Deployments Advanced ▾ External Links ▾ Marica Antonacci ▾

Search...

Virtual machine

Elasticsearch and Kibana

Apache Mesos cluster

Kubernetes cluster

Spark + Jupyter cluster

SLURM Elastic cluster

RStudio

TensorFlow with Jupyter

Dogs breed detection

Massive Online Data Streams

Massive Online Data Streams

Massive Online Data Streams

© 2019 DEEP Hybrid DataCloud
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777435.

DEEP Orchestrator Dashboard Deployments Advanced ▾ External Links ▾

SLURM cluster

Description: Launch a Virtual Slurm Cluster. It will launch a single front-end and a s

Deployment description: description

Configuration Advanced

wn_num: 1
Number of WNs in the elastic cluster

fe_cpus: 1
Numer of CPUs for the front-end node

fe_mem: 1 GB
Amount of Memory for the front-end node

wn_cpus: 1
Numer of CPUs for the WNs

wn_mem

© 2019 DEEP Hybrid DataCloud
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777435.

paas-dev.cloud.ba.infn.it

DEEP Orchestrator Dashboard

Deploy a Mesos cluster

SLURM Elastic cluster

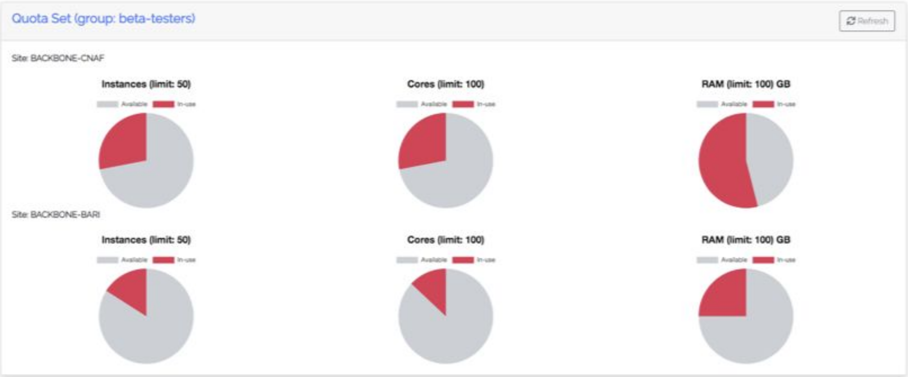
lifewatch-algaebloom.yaml

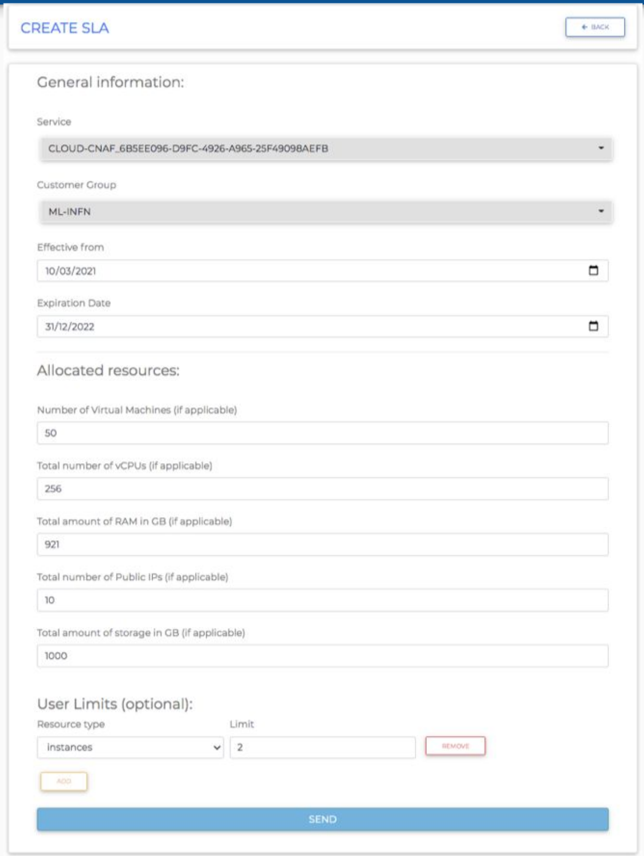
© 2019 DEEP Hybrid DataCloud
This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 777435.

A new tool for SLAs configuration - WIP

SLAT (Service Level Agreement Tool) is replacing the old SLAM (from Cyfronet)

- Development started during EOSC-Hub project and is continuing in INFN Cloud
    - In-house development allows us to control this important service
- New features are/will be available, e.g. per-group SLAs, user limits and quota management.





CREATE SLA

General information:

Service
CLOUD-CNAF_6B5EE096-D9FC-4926-A965-25F49098AEFB

Customer Group
ML-INFN

Effective from
10/03/2021

Expiration Date
31/12/2022

Allocated resources:

Number of Virtual Machines (if applicable)
50

Total number of vCPUs (if applicable)
256

Total amount of RAM in GB (if applicable)
921

Total number of Public IPs (if applicable)
10

Total amount of storage in GB (if applicable)
1000

User Limits (optional):

Resource type: instances
Limit: 2

Conclusions

- Several improvements and extensions have been implemented in the INDIGO PaaS during the last 3 years
  - Still working for providing a complete documentation (admin & user)
- The PaaS Orchestrator system has been included in the **EOSC Marketplace**
- It has been adopted and used in diverse **production** services, e.g. Laniakea@ReCaS (Elixir Italy)
- Several on-going projects rely on the INDIGO PaaS as federation and orchestration tool:
  - IoTwins, EGI-ACE, C-SCALE, ...
- **INFN Cloud** is exploiting the INDIGO PaaS capabilities for federating the resources provided by the backbone and the satellite sites

Thank you for your attention