PaaS-level Components

Alfonso Pérez, Miguel Caballer, Germán Moltó, Marica Antonacci, Lukasz Dutka

Universitat Politècnica de València, INFN-Bari, Cyfronet AGH

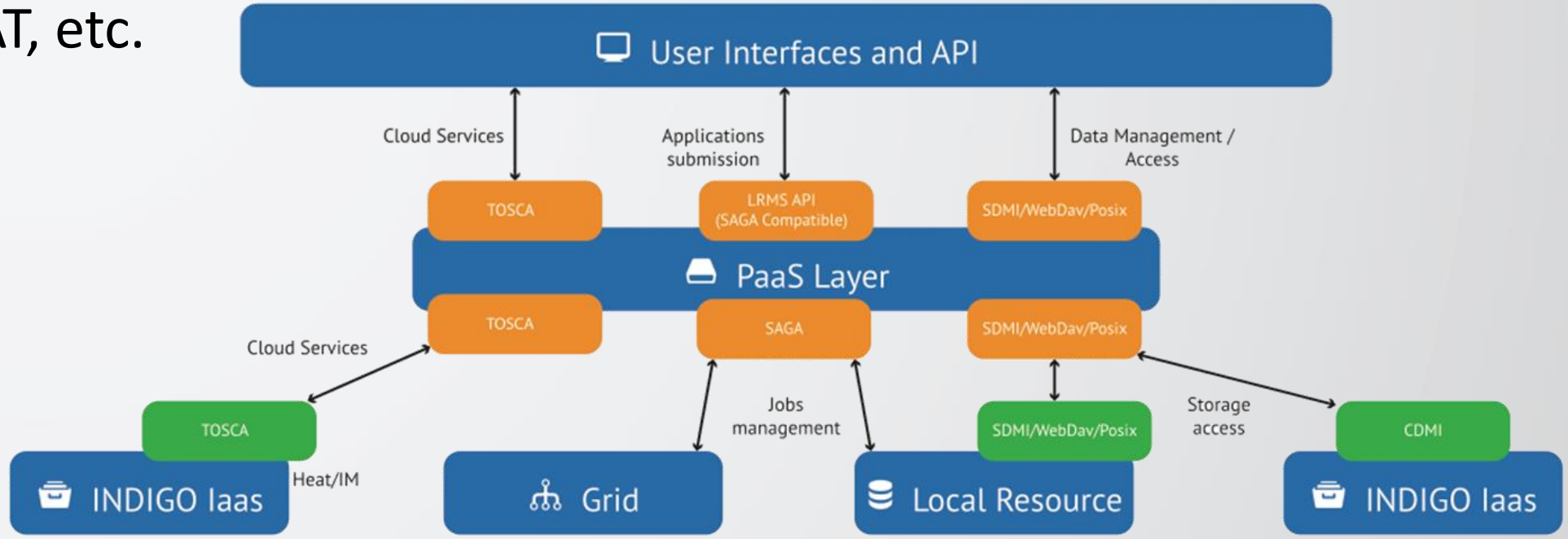
INDIGO-DataCloud is co-founded by the Horizon 2020Framework Programme

Index

Index

INDIGO advanced PaaS Platform

- INDIGO implements an advanced PaaS layer allowing scientific communities to exploit, in a powerful and high-level way, several heterogeneous computing and data e-infrastructure such as: IaaS Cloud, Helix Nebula, EGI Grid, EGI Federated Cloud, PRACE, HPC, EUDAT, etc.



INDIGO PaaS: Key features

- Improved capabilities in the geographical exploitation of Cloud resources.
- Standard interface to access PaaS services.
  - INDIGO will use the TOSCA standard
- Support for data requirements in Cloud resource allocations.
  - Resources can be allocated where data is stored.
- Integrated use of resources coming from both public and private Cloud infrastructures

INDIGO PaaS: Key features

- Distributed data federations supporting legacy applications as well as high level capabilities for distributed QoS and Data Lifecycle Management.
  - This includes for example remote Posix access to data.
- Transparent client-side import/export of distributed Cloud data.
  - This supports dropbox-like mechanisms for importing and exporting data from/to the Cloud. That data can then be easily ingested by Cloud applications through the INDIGO unified data tools.
- Support for distributed data caching mechanisms and integration with existing storage infrastructures.

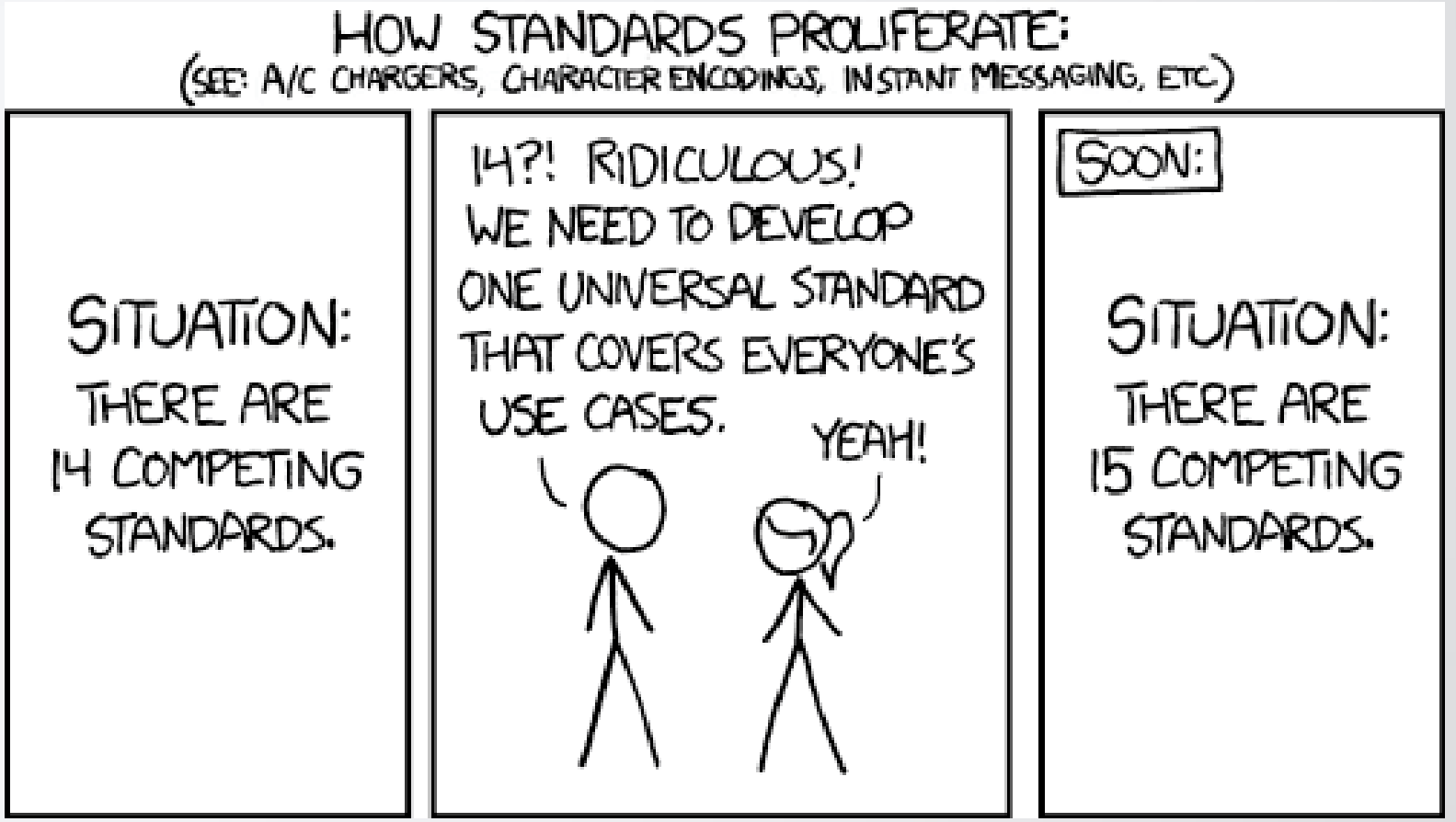
INDIGO PaaS: Key features

- Deployment, monitoring and automatic scalability of existing applications.
- Integrated support for high-performance Big Data analytics.
- Support for dynamic and elastic clusters of resources.
  - batch systems on-demand (such as HTCondor or Torque)
  - extensible application platforms (such as Apache Mesos) capable of supporting both application execution and instantiation of long-running services.

Index

- Indigo PaaS Overview
- **TOSCA**
- IM
- CLUES
- Orchestrator
- Cloud Provider Ranker
- Monitoring Service
- SLAM
- Accounting Service
- CMDB
- Managed Services/Application (MSA) Deployment Service
- OneData

Standards


HOW STANDARDS PROLIFERATE:
(SEE: A/C CHARGERS, CHARACTER ENCODINGS, INSTANT MESSAGING, ETC)
SITUATION: THERE ARE 14 COMPETING STANDARDS.
14?! RIDICULOUS! WE NEED TO DEVELOP ONE UNIVERSAL STANDARD THAT COVERS EVERYONE'S USE CASES.
YEAH!
SOON:
SITUATION: THERE ARE 15 COMPETING STANDARDS.

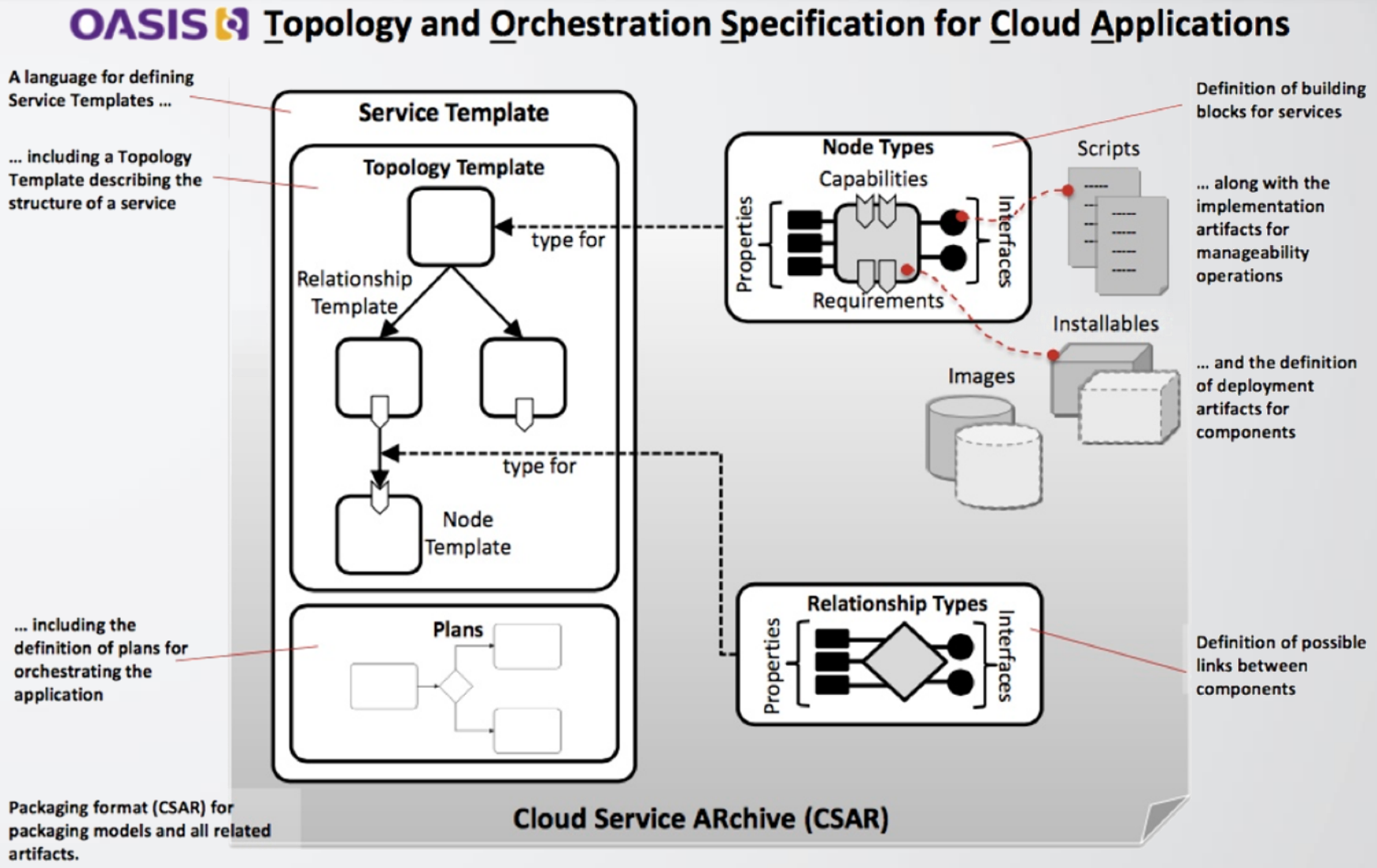
TOSCA

INDIGO - DataCloud

- Topology and Orchestration Specification for Cloud Applications

- 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

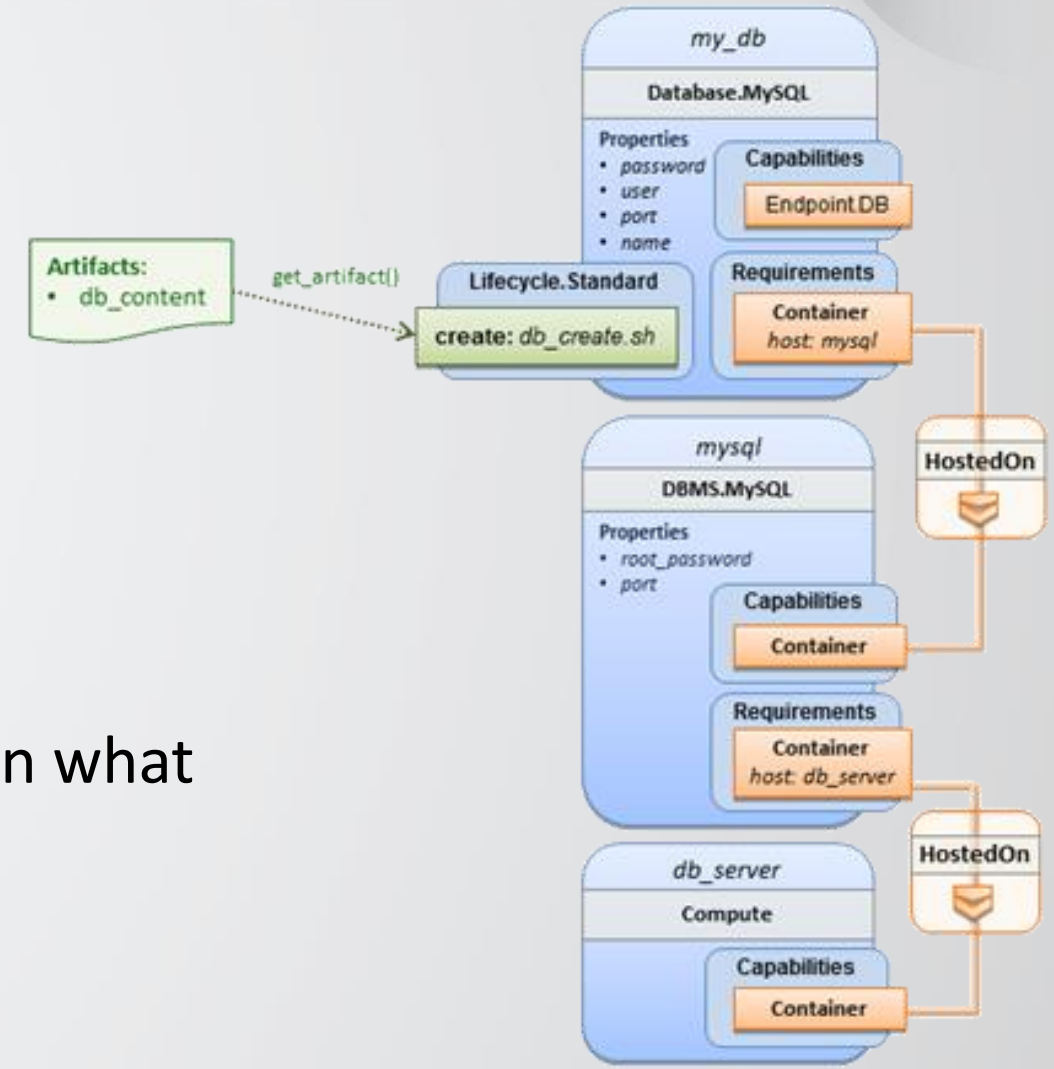
TOSCA in a nutshell



TOSCA Topology

- 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

INDIGO - DataCloud

- 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

Node Type Examples

- Defines properties as YAML maps
- Might defines capabilities (What it can provide to other nodes)

```
tosca.nodes.DBMS
  derived_from: tosca.nodes.SoftwareComponent
  properties:
    dbms_root_password:
      type: string
      description: the root password for the DBMS service
    dbms_port:
      type: integer
      description: the port the DBMS service will listen to for data and requests
  capabilities:
    host:
      type: Container
      containee_types: [ tosca.nodes.Database ]
```

Node Type Examples

INDIGO - DataCloud

- Might Define Requirements (what it needs from other nodes)

```
tosca.nodes.Database:
  derived_from: tosca.nodes.Root
  properties:
    db_user:
      type: string
      description: user account name for DB administration
    db_password:
      type: string
      description: the password for the DB user account
    db_port:
      type: integer
      description: the port the underlying database service will listen to data
    db_name:
      type: string
      description: the logical name of the database
  requirements:
    - host: tosca.nodes.DBMS
  capabilities:
    - database_endpoint: tosca.capabilities.DatabaseEndpoint
```

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)

Topology Template Example

INDIGO - DataCloud

```
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 ] }
```

INDIGO custom types

INDIGO - DataCloud

- Extend the TOSCA normative types
  - E.g.: tosca.nodes.indigo.GalaxyPortal, tosca.nodes.indigo.GalaxyWN

```
tosca.nodes.indigo.GalaxyPortal:
  derived_from: tosca.nodes.WebServer
  properties:
    admin:
      type: string
      description: email of the admin user
      default: admin@admin.com
      required: false
    admin_api_key:
      type: string
      description: key to access the API with admin role
      default: not_very_secret_api_key
      required: false
    user:
      type: string
      description: username to launch the galaxy daemon
      default: galaxy
      required: false
    install_path:
      type: string
      description: path to install the galaxy tool
      default: /home/galaxy/galaxy
      required: false
  requirements:
    - lrms:
        capability: tosca.capabilities.indigo.LRMS
        node: tosca.nodes.indigo.LRMS.FrontEnd
        relationship: tosca.relationships.HostedOn
  artifacts:
    galaxy_role:
      file: indigo-dc.galaxy
      type: tosca.artifacts.AnsibleGalaxy.role
  interfaces:
    Standard:
      configure:
        implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/master/artifacts/galaxy/galaxy_install.yml
        inputs:
          galaxy_install_path: { get_property: [ SELF, install_path ] }
          galaxy_user: { get_property: [ SELF, user ] }
          galaxy_admin: { get_property: [ SELF, admin ] }
          galaxy_admin_api_key: { get_property: [ SELF, admin_api_key ] }
          galaxy_lrms: { get_property: [ SELF, lrms, type ] }
```

Ansible playbook

A Sample tosca template: kepler

INDIGO - DataCloud

```
tosca_definitions_version: tosca_simple_yaml_1_0

imports:

 - indigo_custom_types: https://raw.githubusercontent.com/indigo-dc/tosca-types/master/custom_types.yaml

description: TOSCA template for deploying an instance for Kepler

topology_template:

 inputs:

  number_cpus:

   type: integer

   description: number of cpus required for the instance

   default: 1

  memory_size:

   type: string

   description: ram memory required for the instance

   default: 1 GB

 node_templates:

  kepler:

   type: tosca.nodes.indigo.Kepler

   requirements:

    - host: kepler_server
```

Computing requirements

```
kepler_server:
    type: tosca.nodes.indigo.Compute
    capabilities:
     endpoint:
      properties:
       network_name: PUBLIC
       ports:
        vnc_port:
         protocol: tcp
         source: 5900
     host:
      properties:
       num_cpus: { get_input: number_cpus }
       mem_size: { get_input: memory_size }
     os:
      properties:
       type: linux
       distribution: ubuntu
       version: 14.04
       image: one://onecloud.i3m.upv.es:2633/67

 outputs:
  instance_ip:
   value: { get_attribute: [ kepler_server, public_address, 0 ] }
  instance_creds:
   value: { get_attribute: [ kepler_server, endpoint, credential, 0 ] }
```

Network requirements

OS requirements

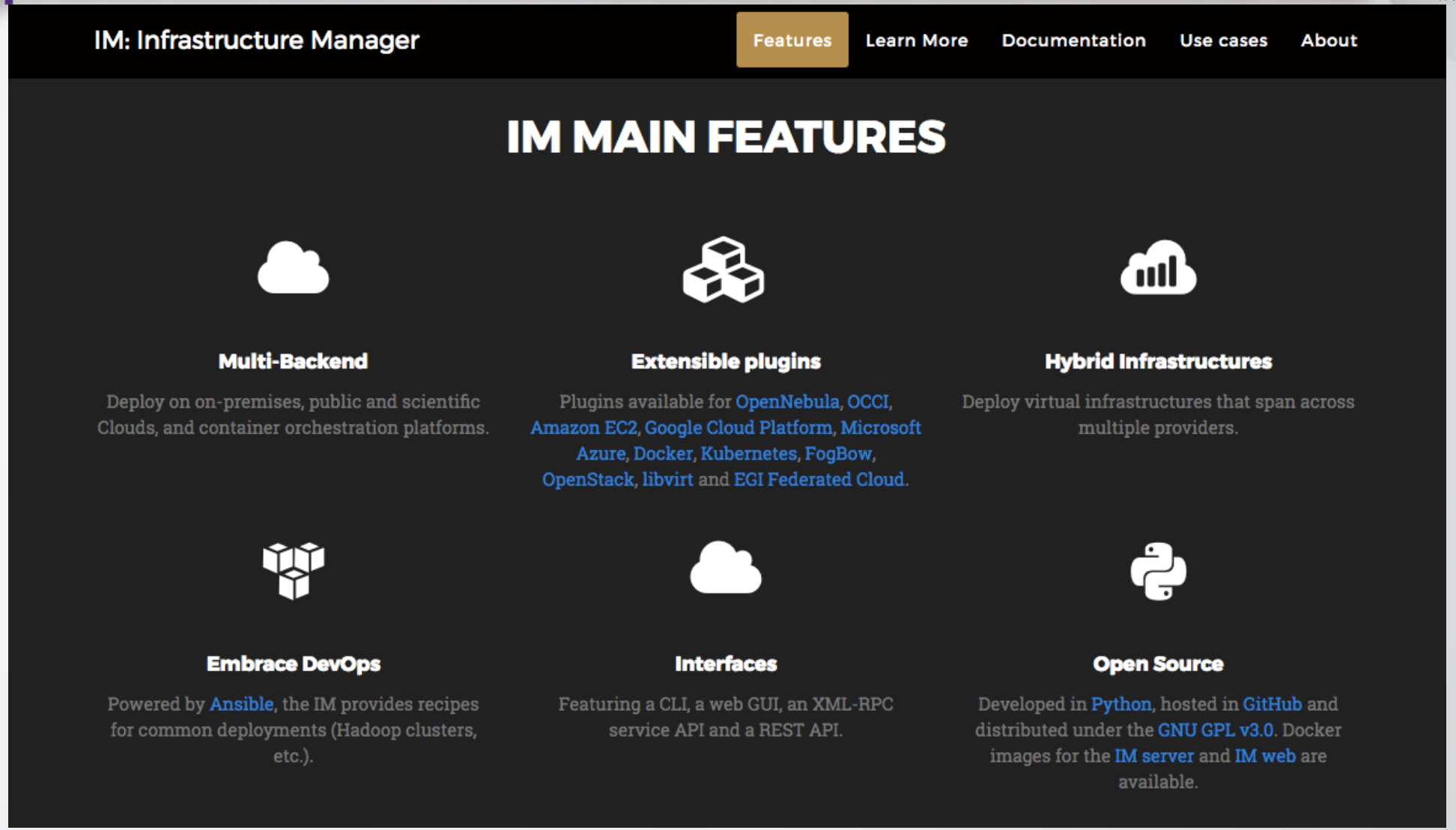
Image registered in the Cloud site

Index

- Indigo PaaS Overview
- TOSCA
- **IM**
- CLUES
- Orchestrator
- Cloud Provider Ranker
- Monitoring Service
- SLAM
- Accounting Service
- CMDB
- Managed Services/Application (MSA) Deployment Service
- OneData

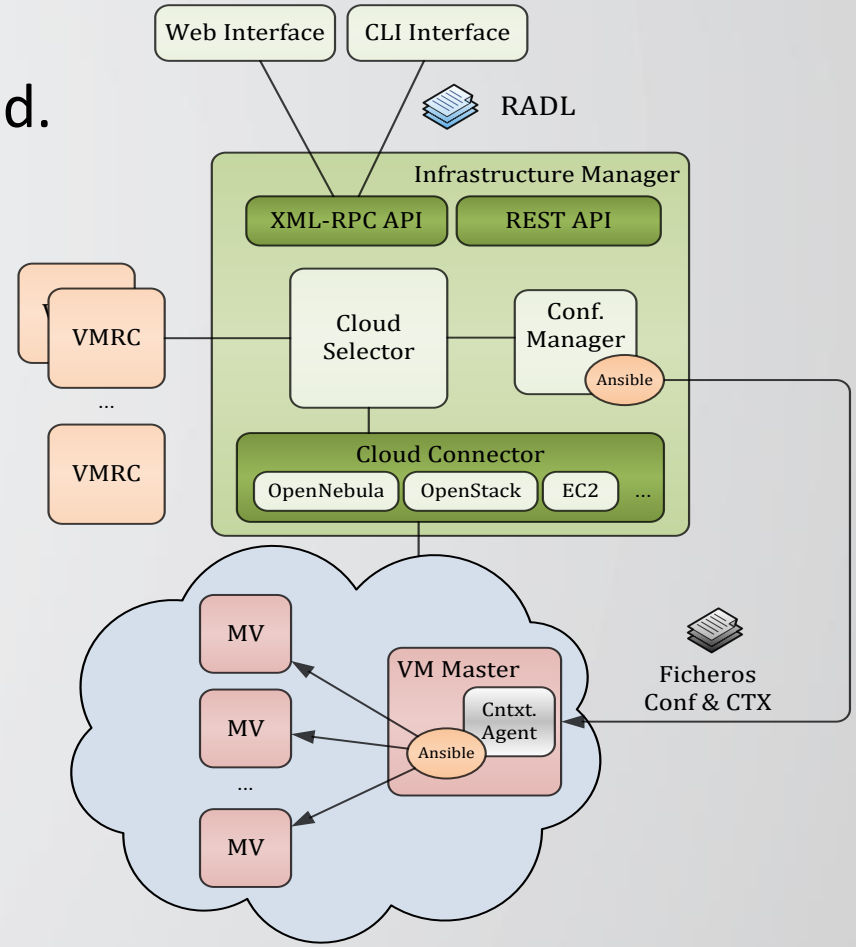
Landing page

Features


IM: Infrastructure Manager
Features
Learn More
Documentation
Use cases
About
IM MAIN FEATURES
Multi-Backend
Deploy on on-premises, public and scientific Clouds, and container orchestration platforms.
Extensible plugins
Plugins available for OpenNebula, OCCI, Amazon EC2, Google Cloud Platform, Microsoft Azure, Docker, Kubernetes, FogBow, OpenStack, libvirt and EGI Federated Cloud.
Hybrid Infrastructures
Deploy virtual infrastructures that span across multiple providers.
Embrace DevOps
Powered by Ansible, the IM provides recipes for common deployments (Hadoop clusters, etc.).
Interfaces
Featuring a CLI, a web GUI, an XML-RPC service API and a REST API.
Open Source
Developed in Python, hosted in GitHub and distributed under the GNU GPL v3.0. Docker images for the IM server and IM web are available.

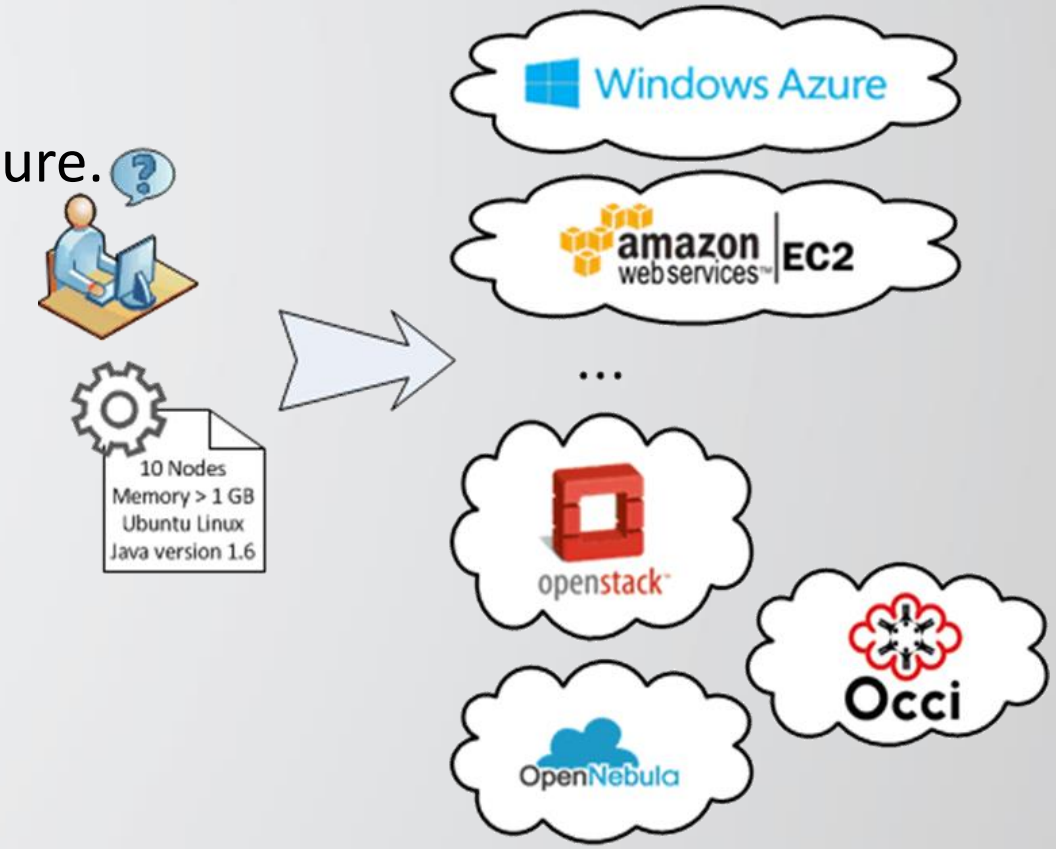
Introduction

- General platform to deploy on demand customizable virtual computing infrastructures.
  - With the precise software configuration required.
  - Allow to deploy any kind of complex infrastructure.
  - Share Infrastructure descriptions.
  - No need of pre-baked VMIs.
  - The same complex infrastructure can be deployed both on-premises and in a public Cloud.



Cloud providers

- It supports a wide range of cloud providers and other computing back-ends :
  - **Public**: Amazon Web Services (AWS), Google Cloud Platform (GCP), Microsoft Azure.
  - **On-premises**: OpenNebula, OpenStack, libvirt.
  - **Federated**: EGI FedCloud (OCCI), FogBow.
  - **Containers**: Docker, Kubernetes
  - The list above can be easily extended by plugins.



Main features

INDIGO - DataCloud

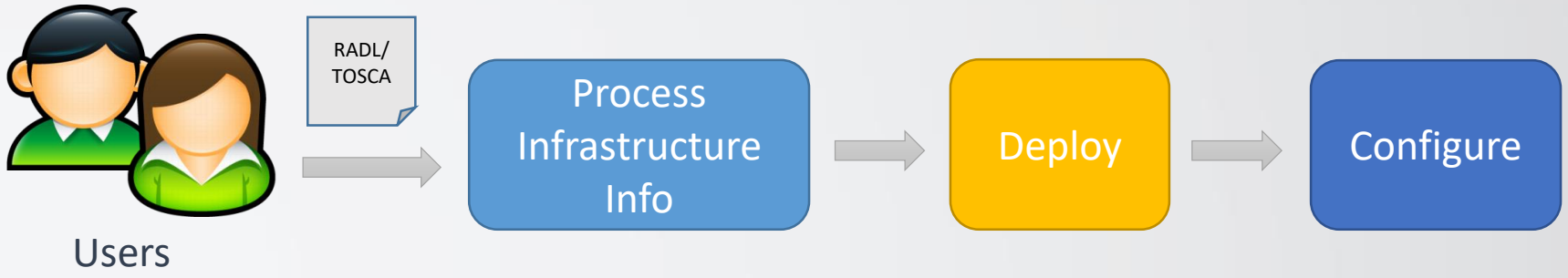
- It features **DevOps** capabilities.
  - Based on Ansible.
  - Provides recipes for common deployments.
  - Also supporting cloud-init scripts.

ANSIBLE

- IM works as a service that offers several **interfaces**:
  - XML-RPC and REST APIs.
  - Command-line application.
  - Web-based GUI.
- It is distributed under a **GNU** GPL v3.0 open source license and its source code is available on **GitHub**.

https://github.com/grycap/im

Working Scheme (I)



- The user can provide an RADL or TOSCA documents as input to the IM, describing the infrastructure:
  - **RADL**:
    - Resource and Application Description Language.
    - High level Language to define virtual infrastructures and Specify VM requirements.
  - **TOSCA**:
    - OASIS Standard
    - Open standard language to model application architectures to be deployed on a Cloud.

RADL Document

INDIGO - DataCloud

```
ansible <ansible_host_id> (<features>)

network <network_id> (<features>)

system <system_id> (<features>)

configure <configure_id> (<Ansible recipes>)

contextualize [max_time] ( system <system_id>

configure <configure_id> [step <num>] ... )

deploy <system_id> <num> [<cloud_id>]
```

The keywords ansible, network, system and configure assign some *features* or *recipes* to an identity <id>. The features are a list of constraints separated by and, and a constraint is formed by <feature name> <operator> <value>.

```
network net (inbound = 'no')

system node (
  cpu.arch = 'x86_64' and
  cpu.count = 1 and
  memory.size >= 512M and
  net_interface.0.connection = 'net' and
  disk.0.os.name = 'linux‘
 )

configure node (
@begin
---
 - tasks:
  - user: name=user1 password=1234
 @end
 )

contextualize (
  system node configure node
)

deploy node 1
```

Using IM: A simple node with Ansible


```
network publica (outbound = 'yes')
system front (
  instance_type = 'extra-large' and
  net_interface.0.connection = 'publica' and
  disk.0.os.name = 'linux' and
  disk.0.image.url = 'https://one.upv.es:9856/495' and
  disk.0.os.credentials.username = 'cloudadm' and
  disk.0.os.credentials.password = 'xxx'
)
configure front (
@begin
 - name: Install some packages
   action: yum pkg=pkg1,pkg2 state=installed
@end
)
```

A simple node with cloud-init

INDIGO - DataCloud

```
network publica (outbound = 'yes')
system front (
  instance_type = 'extra-large' and
  net_interface.0.connection = 'publica' and
  disk.0.os.name = 'linux' and
  disk.0.image.url =
  'https://fc-one.i3m.upv.es:11443/uuid_image_for_egi_centos_6_centos6kvm_im154_170' and
  disk.0.os.credentials.username = 'cloudadm' and
  disk.0.os.credentials.password = `xxx`
)
configure front (
@begin
  #!/bin/bash
  yum install -y pkg1 pkg2
@end
)
```

```
contextualize (
  system front configure front with cloud_init
)
```

A Sample tosca template: kepler

INDIGO - DataCloud

```
tosca_definitions_version: tosca_simple_yaml_1_0

imports:

 - indigo_custom_types: https://raw.githubusercontent.com/indigo-dc/tosca-types/master/custom_types.yaml

description: TOSCA template for deploying an instance for Kepler

topology_template:

 inputs:

  number_cpus:

   type: integer

   description: number of cpus required for the instance

   default: 1

  memory_size:

   type: string

   description: ram memory required for the instance

   default: 1 GB

 node_templates:

  kepler:

   type: tosca.nodes.indigo.Kepler

   requirements:

    - host: kepler_server
```

Computing requirements

```
kepler_server:
    type: tosca.nodes.indigo.Compute
    capabilities:
     endpoint:
      properties:
       network_name: PUBLIC
       ports:
        vnc_port:
         protocol: tcp
         source: 5900
     host:
      properties:
       num_cpus: { get_input: number_cpus }
       mem_size: { get_input: memory_size }
     os:
      properties:
       type: linux
       distribution: ubuntu
       version: 14.04
       image: one://onecloud.i3m.upv.es:2633/67

 outputs:
  instance_ip:
   value: { get_attribute: [ kepler_server, public_address, 0 ] }
  instance_creds:
   value: { get_attribute: [ kepler_server, endpoint, credential, 0 ] }
```

Network requirements

OS requirements

Image registered in the Cloud site

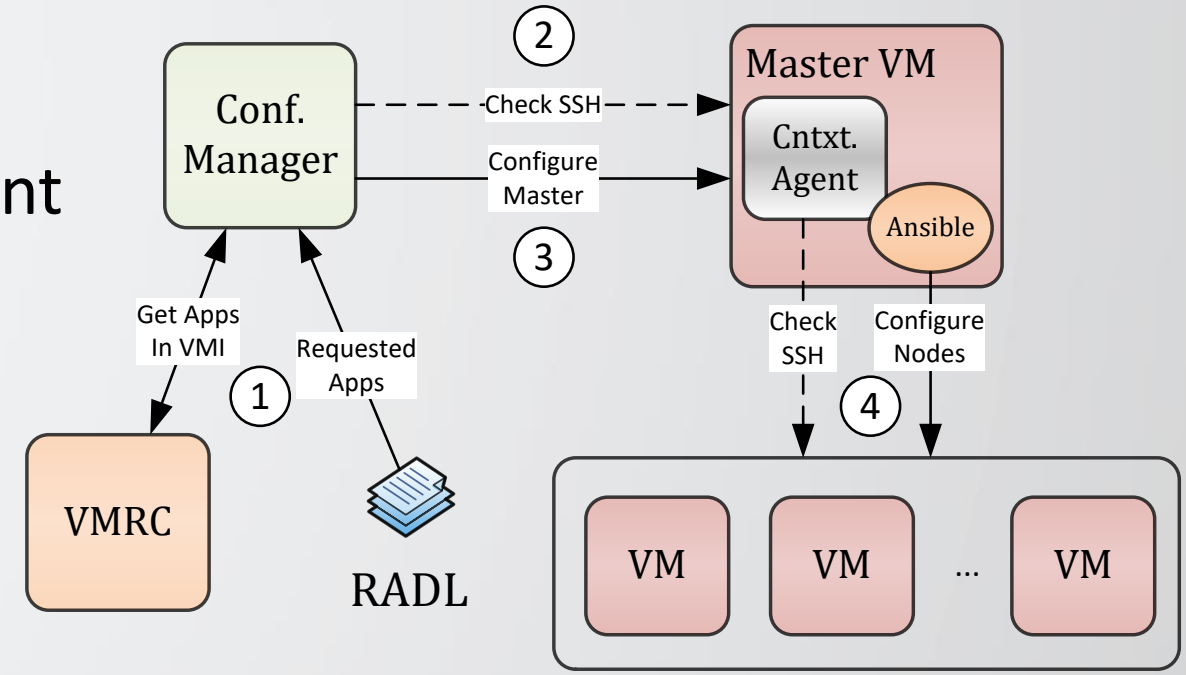
Working Scheme (II)

INDIGO - DataCloud

- The user specifies the image (or list of images) to use.
  - URI naming convention to abstract from cloud provider:
    - one://server:port/image-id
    - ost://server:port/ami-id
    - aws://region/ami-id
    - <site end-point>/<image-id>
  - The IM can also contact the VMRC (http://www.grycap.upv.es/vmrc) to get a list with the most suitable VMIs
    - In INDIGO-DataCloud, the image information is obtained from the CMDB.
- Then, the IM obtains the list of IaaS providers available to the user.
  - From the credentials provided by the user.
- Finally, it contacts the IaaS provider selected and deploys the infrastructure.

Contextualization process: with ansible

1. SSH connection to the Master VM
   - A GNU/Linux-based VM with a public IP
2. Configure Master VM
   - Install and configure Ansible
3. Launch Contextualization Agent
   - Check SSH from VMs
   - Call Ansible

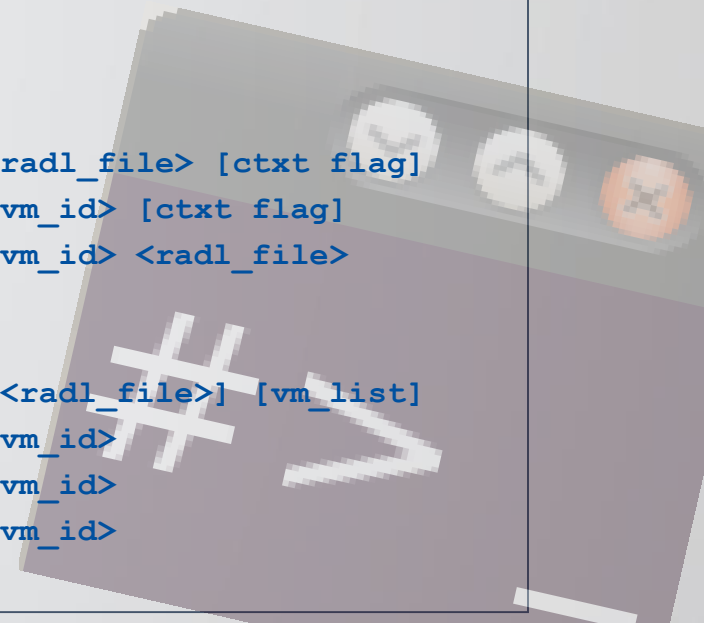


Client Tools

INDIGO - DataCloud

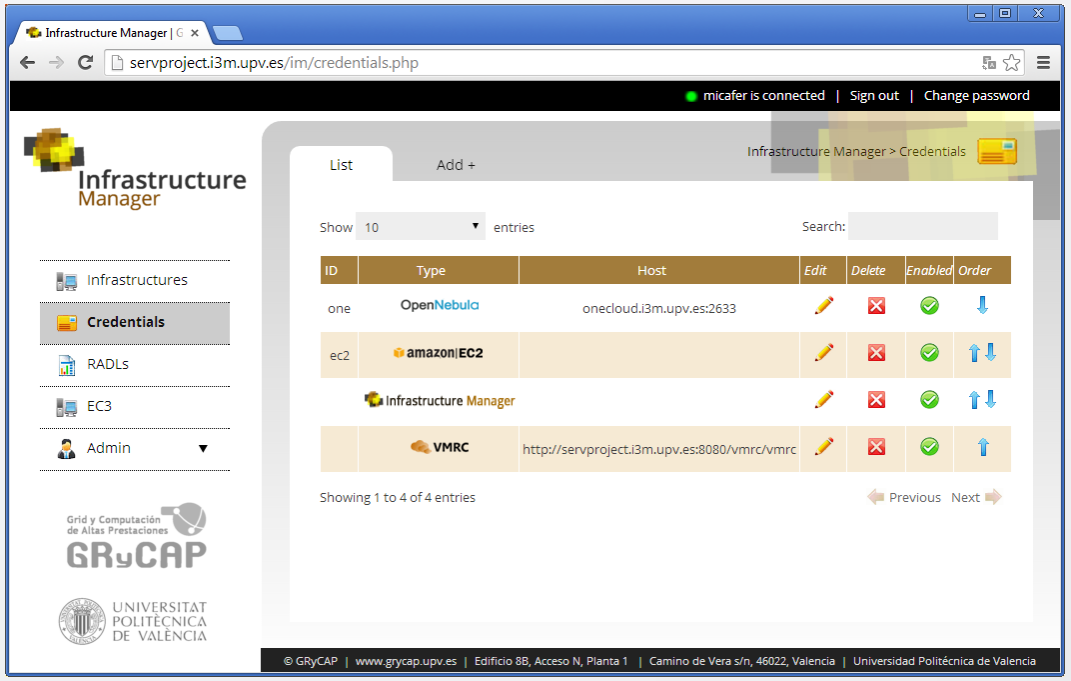
- Command Line Tool (CLI).

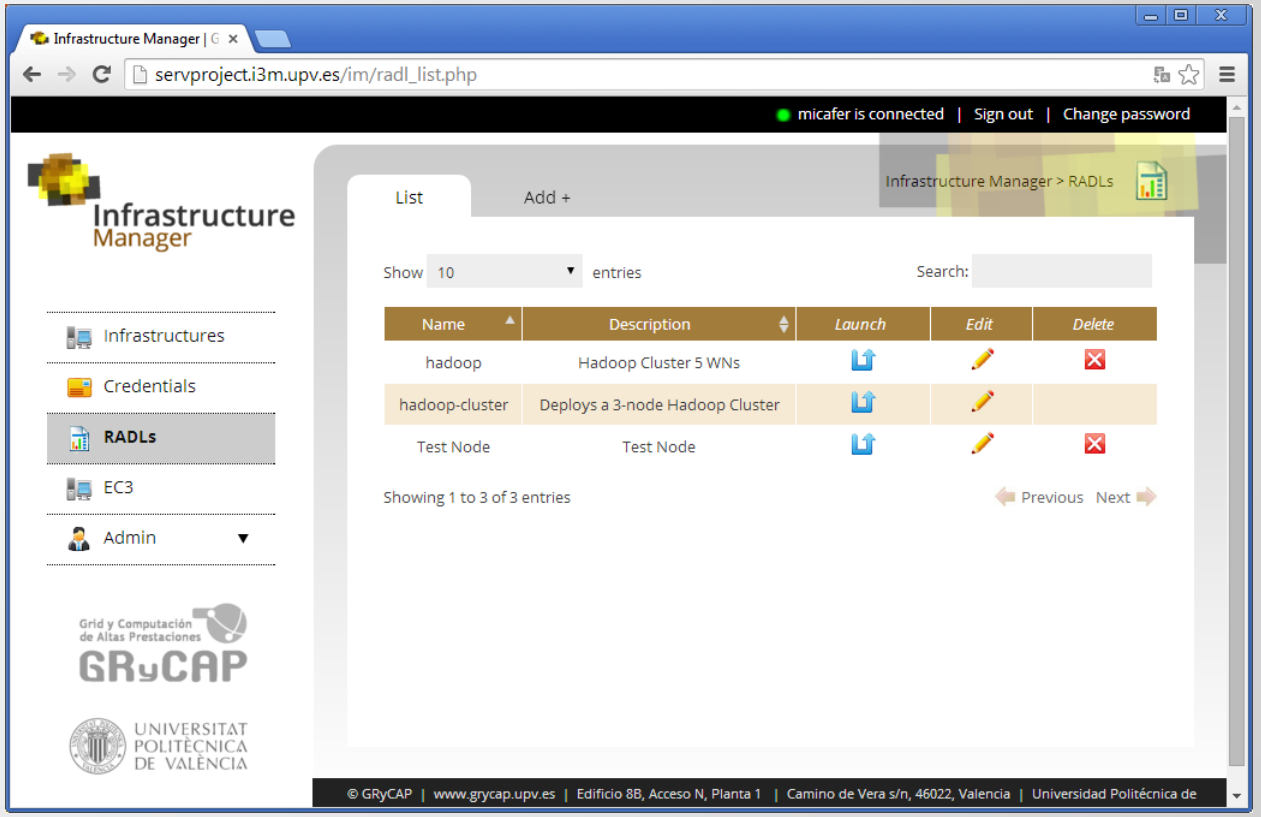
```
usage: client.py [-u|--xmlrpc-url <url>] [-a|--auth_file <filename>] operation op_parameters
  options:
    --version           show program's version number and exit
    -h, --help          show this help message and exit
    -a AUTH_FILE, --auth_file=AUTH_FILE
                        File with authentication data
    -u XML-RPC, --xmlrpc-url=XML-RPC
                        URL of the InfrastructureManager service.
Operations:
  list                                                  addresource     <inf_id> <radl_file> [ctxt flag]
  create          <radl_file>                           removeresource  <inf_id> <vm_id> [ctxt flag]
  destroy         <inf_id>                              alter           <inf_id> <vm_id> <radl_file>
  getinfo         <inf_id> [radl_attribute]             start           <inf_id>
  getradl         <inf_id>                              stop            <inf_id>
  getcontmsg      <inf_id>                              reconfigure     <inf_id> [<radl_file>] [vm_list]
  getstate        <inf_id>                              startvm         <inf_id> <vm_id>
  getvminfo       <inf_id> <vm_id> [radl_attribute]     stopvm          <inf_id> <vm_id>
  getvmcontmsg    <inf_id> <vm_id>                      sshvm           <inf_id> <vm_id>
                                                        getversion
```



Client-side Tools: Web

- Publicly-available web interface (also open-sourced):
  - Easily deploy infrastructures from a web browser
  - Share RADL/TOSCA documents.
  - http://servproject.i3m.upv.es/im/





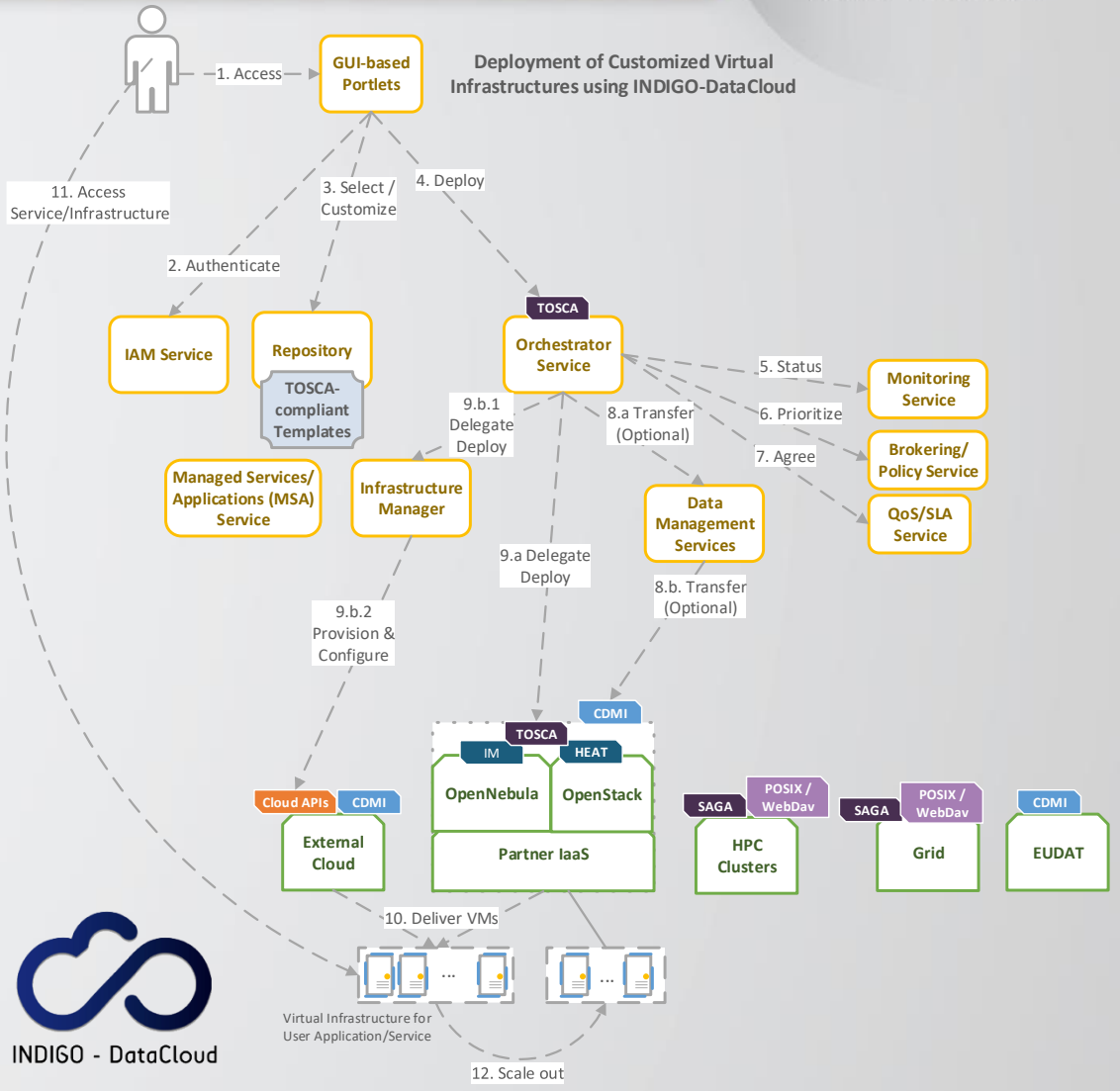
APIs to be consumed by clients

- **XML-RPC API**
  - API that follows the XML-RPC specification.
- **REST API**
  - IM Service can be accessed through a REST(ful) API
- More info: http://www.grycap.upv.es/im/documentation.php

HTTP method	/infrastructures	/infrastructures/<infId>	/infrastructures/<infId> /vms/<vmId>
GET	**List** the infrastructure IDs.	**List** the virtual machines in the infrastructure `infId`	**Get** information associated to the virtual machine `vmId` in `infId`.
POST	**Create** a new infrastructure based on the RADL posted	**Create** a new virtual machine based on the RADL posted.	
PUT			**Modify** the virtual machine based on the RADL posted.
DELETE		**Undeploy** all the virtual machines in the infrastructure.	**Undeploy** the virtual machine.

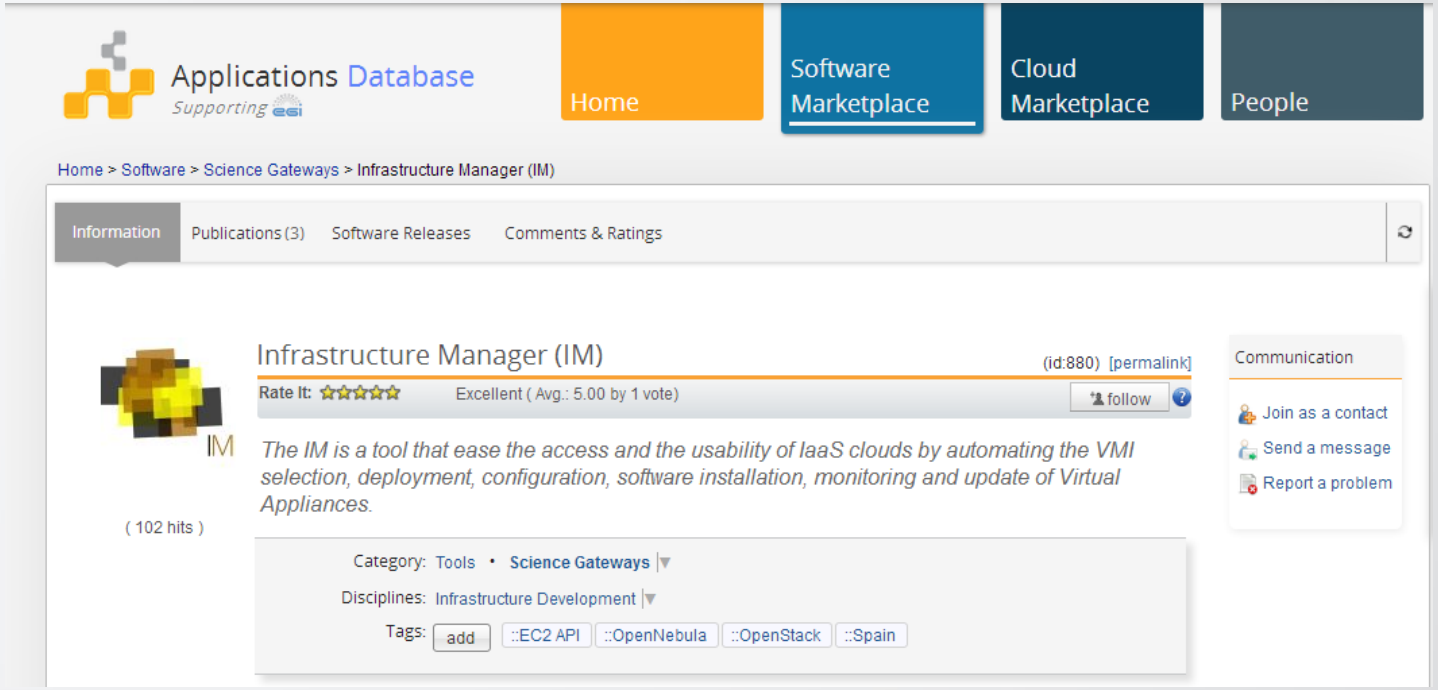
Where is the im used? - INDIGO-DataCloud

- In the **INDIGO-DataCloud** project:
  - IM is a **key component** of the architecture:
    - Used at the PaaS Core to provide deployment of infrastructures to Cloud sites external to INDIGO-DataCloud (including public Cloud sites).
    - Used at IaaS to provide TOSCA-based deployment of infrastructures for OpenNebula sites.



Where is the im used? - EGI

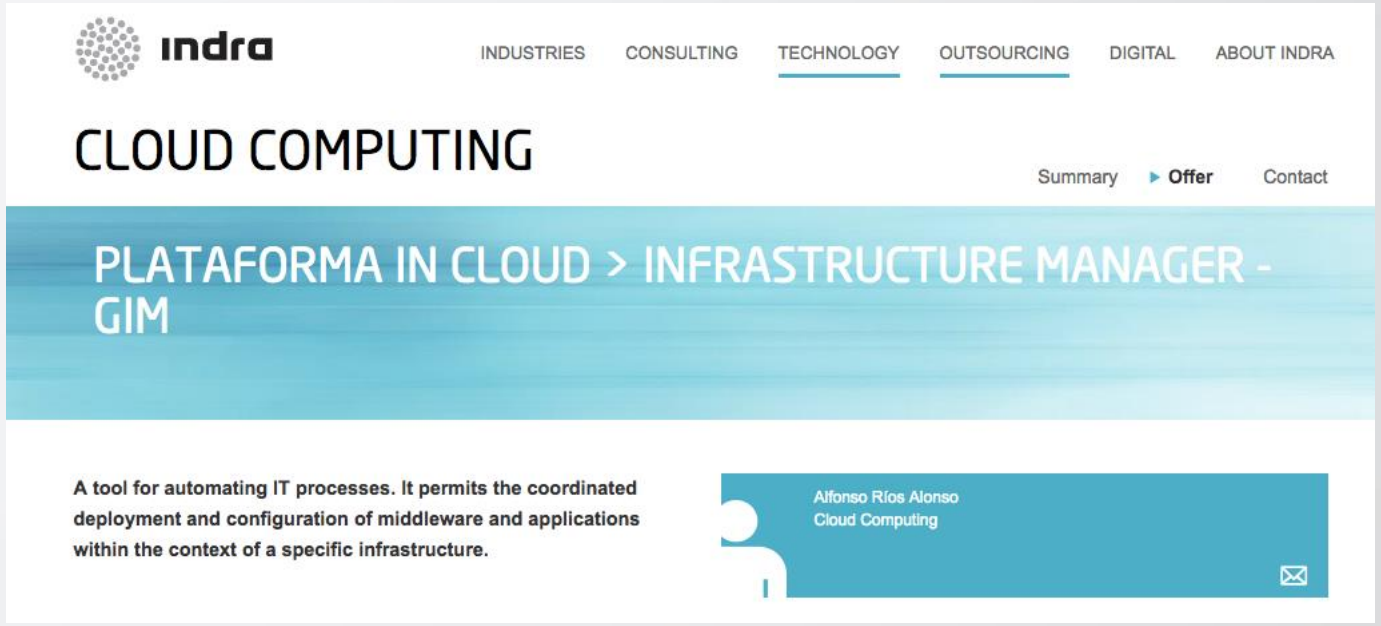
- The IM is used in the **VMOps Dashboard** of **EGI**.
  - As the OCCI communication layer to create VM topologies.
  - TOSCA compatible.
  - https://wiki.egi.eu/wiki/Federated_Cloud_AppDB_VMOps_Dashboard



https://appdb.egi.eu/store/software/infrastructure.manager.im

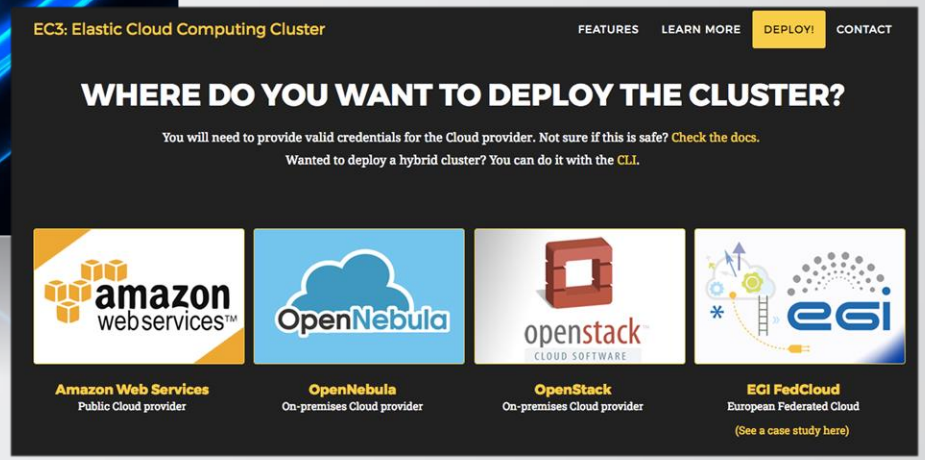
Where is the im used? - INDRA

- The **INDRA** company has integrated the IM in their GPaaS Cloud platform.
  - It permits the coordinated deployment and configuration of middleware and applications within the context of a specific infrastructure.
  - http://www.indracompany.com/en/node/64122



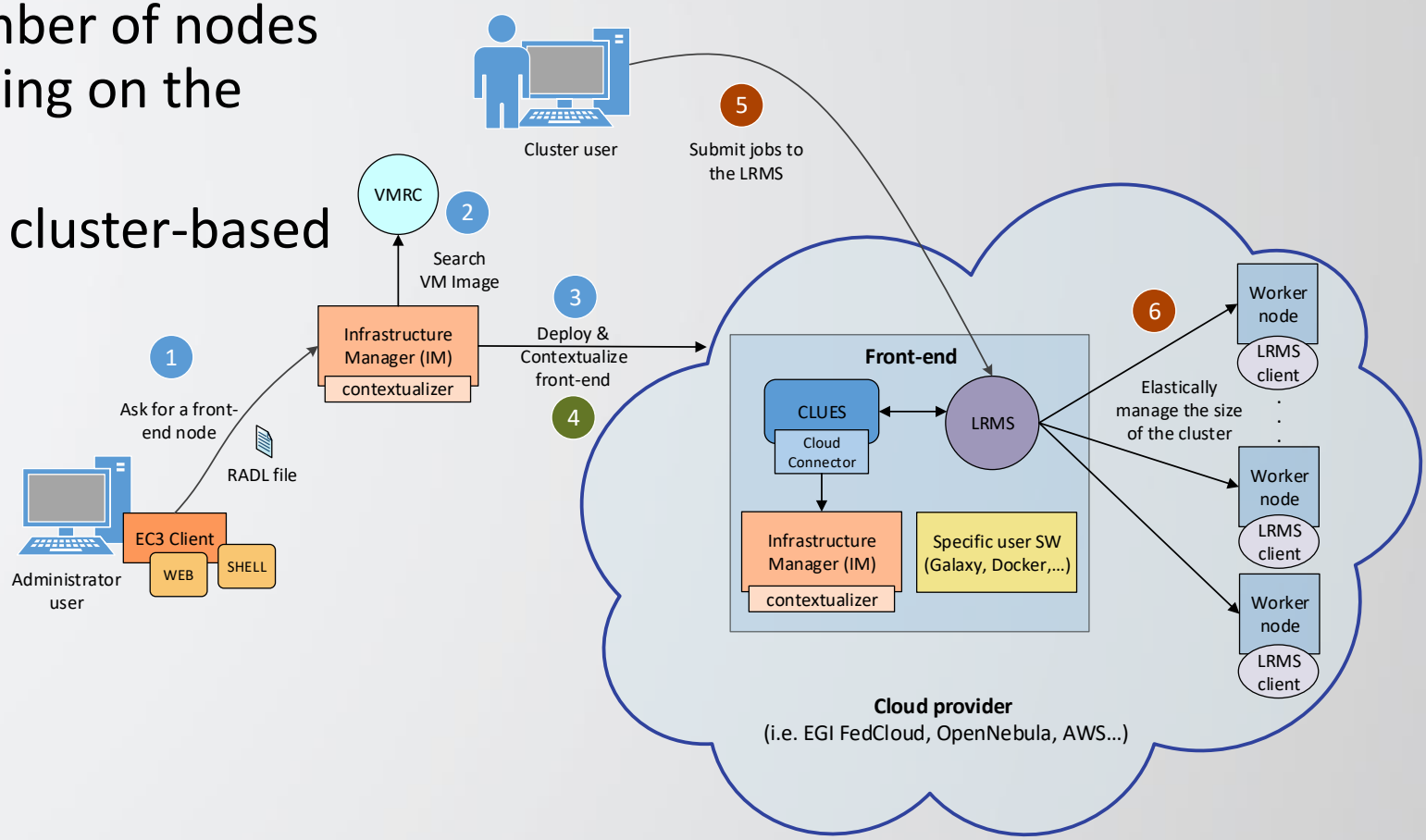
Where is the im used? - EC3

- IM is a key component in **EC3**, a tool to deploy virtual hybrid elastic clusters that is integrated in the **EGI Access** service (for the long-tail of science).
  - EC3: http://www.grycap.upv.es/ec3/
  - EGI Access: https://access.egi.eu/start



Where is the im used? - EC3

- The IM provisions the front-end node of a virtual cluster where the number of nodes can grow and shrink (depending on the workload).
  - Cost and energy-efficient cluster-based computing.



Index

- Indigo PaaS Overview
- TOSCA
- IM
- **CLUES**
- Orchestrator
- Cloud Provider Ranker
- Monitoring Service
- SLAM
- Accounting Service
- CMDB
- Managed Services/Application (MSA) Deployment Service
- OneData

Automatic Scaling Service

- Extends EC3 CLUES adding the interfaces required to interact with the INDIGO Orchestrator
  - Documentation: http://www.grycap.upv.es/ec3
- Implements the elasticity rules considering the state of the virtual cluster.
- The virtual cluster will deploy additional worker nodes as required, and integrate them on the LRMS without user intervention, in order to cope with increased workload of jobs. Worker nodes will be terminated when they are no longer required.
- Plugins are available for: SLURM, Torque/PBS, HTCondor, Mesos

Index

- Indigo PaaS Overview
- TOSCA
- IM
- CLUES
- **Orchestrator**
- Cloud Provider Ranker
- Monitoring Service
- SLAM
- Accounting Service
- CMDB
- Managed Services/Application (MSA) Deployment Service
- OneData

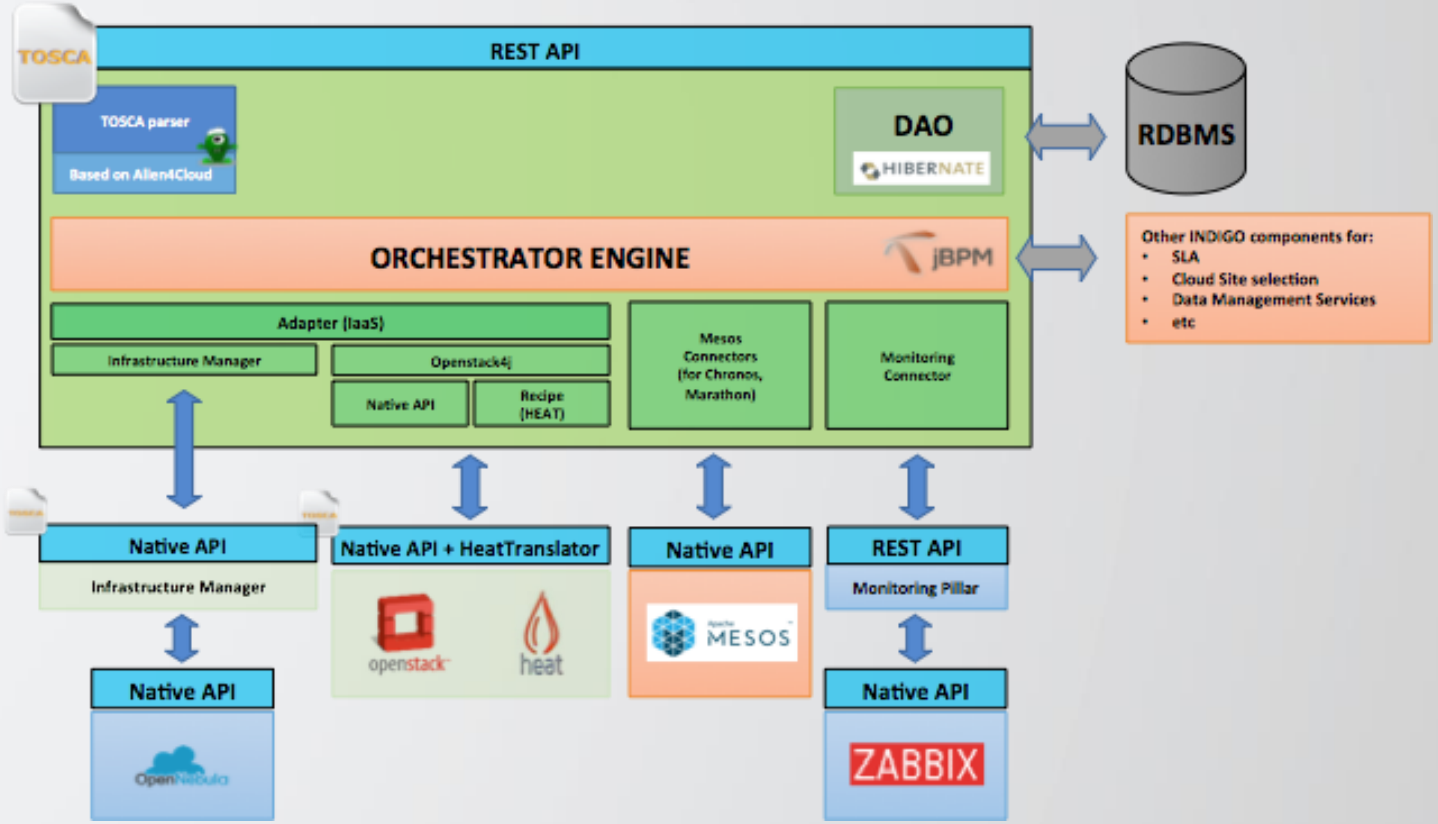
Orchestrator Service

INDIGO - DataCloud

- The Orchestrator **coordinates the deployment process** over the IaaS platforms
- The Orchestrator collects all the information needed to deploy a service consuming others PaaS µServices APIs:
  - ***Monitoring Service***: get the capabilities of the underlying IaaS platforms and their resource availability;
  - ***QoS/SLA Service***: get the prioritized list of SLA per user/group
  - ***CloudProviderRanker*** (Rule Engine) Service: sort the list sites on the basis of rules defined per user/group/use-case;
  - ***Data Management Service***: get the status of the data files and storage resources needed by the service/application
- The orchestrator **delegates** the deployment to **IM**, **HEAT** or **Mesos** based on the TOSCA template and the list of sites.
- **Cross-site deployments** will also be possible.

Orchestrator Service

- Built in **JBPM 6.1** (long-running workflow)
- Exposes **RESTful APIs**
- Supports the TOSCA Simple Profile in **YAML Version 1.0** specification



Index

- Indigo PaaS Overview
- TOSCA
- IM
- CLUES
- Orchestrator
- **Cloud Provider Ranker**
- Monitoring Service
- SLAM
- Accounting Service
- CMDB
- Managed Services/Application (MSA) Deployment Service
- OneData

Cloud Provider Ranker Service

- Provides information that will be consumed by the Orchestrator in order to properly coordinate the deployment of the required resource on the sites.
  - A WEB Service providing REST APIs to rank Cloud Providers described by a JSON blob containing:
    - Total VirtualCPUs, total VirtualRAM
    - Total Virtual Ephemeral Disk (space for instances)
    - Total Virtual Disk (block storage, e.g. Cinder)
    - In use VCPU, in use VRAM, in use VDISK, in use VEphDisk
- Ranking algorithm implemented using the largely diffused **Drools Rule Engine** runtime framework

Index

- Indigo PaaS Overview
- TOSCA
- IM
- CLUES
- Orchestrator
- Cloud Provider Ranker
- **Monitoring Service**
- SLAM
- Accounting Service
- CMDB
- Managed Services/Application (MSA) Deployment Service
- OneData

Monitoring Service

INDIGO - DataCloud

- The Monitoring service provides a comprehensive REST API to gather information about
  - the **PaaS Core Services**
    - Using *Heapster* in the *Kubernetes* cluster.
  - the **customized virtual infrastructures**
    - Using *Zabbix agents* deployed inside VMs/containers
  - the **state of the sites** (leveraging EGI FedCloud approach)

- Monitored information has to be exposed via a REST API to be consumed by other services

Index

- Indigo PaaS Overview
- TOSCA
- IM
- CLUES
- Orchestrator
- Cloud Provider Ranker
- Monitoring Service
- **SLAM**
- Accounting Service
- CMDB
- Managed Services/Application (MSA) Deployment Service
- OneData

QoS/SLA Service

INDIGO - DataCloud

- Allows the **handshake** between a **user** and a **site** on a given **SLA**
- Provides the Orchestrator/Ranker with the useful information for taking the decision on tasks scheduling according to the agreed and valid SLAs
- Describes the **QoS** that a specific user/group has both over a given site or generally in the PaaS as a whole; this includes a priority of a given users, the capability to access to different QoS at each site (Gold, Silver, Bronze services)

Index

- Indigo PaaS Overview
- TOSCA
- IM
- CLUES
- Orchestrator
- Cloud Provider Ranker
- Monitoring Service
- SLAM
- **Accounting Service**
- CMDB
- Managed Services/Application (MSA) Deployment Service
- OneData

Accounting Service

INDIGO - DataCloud

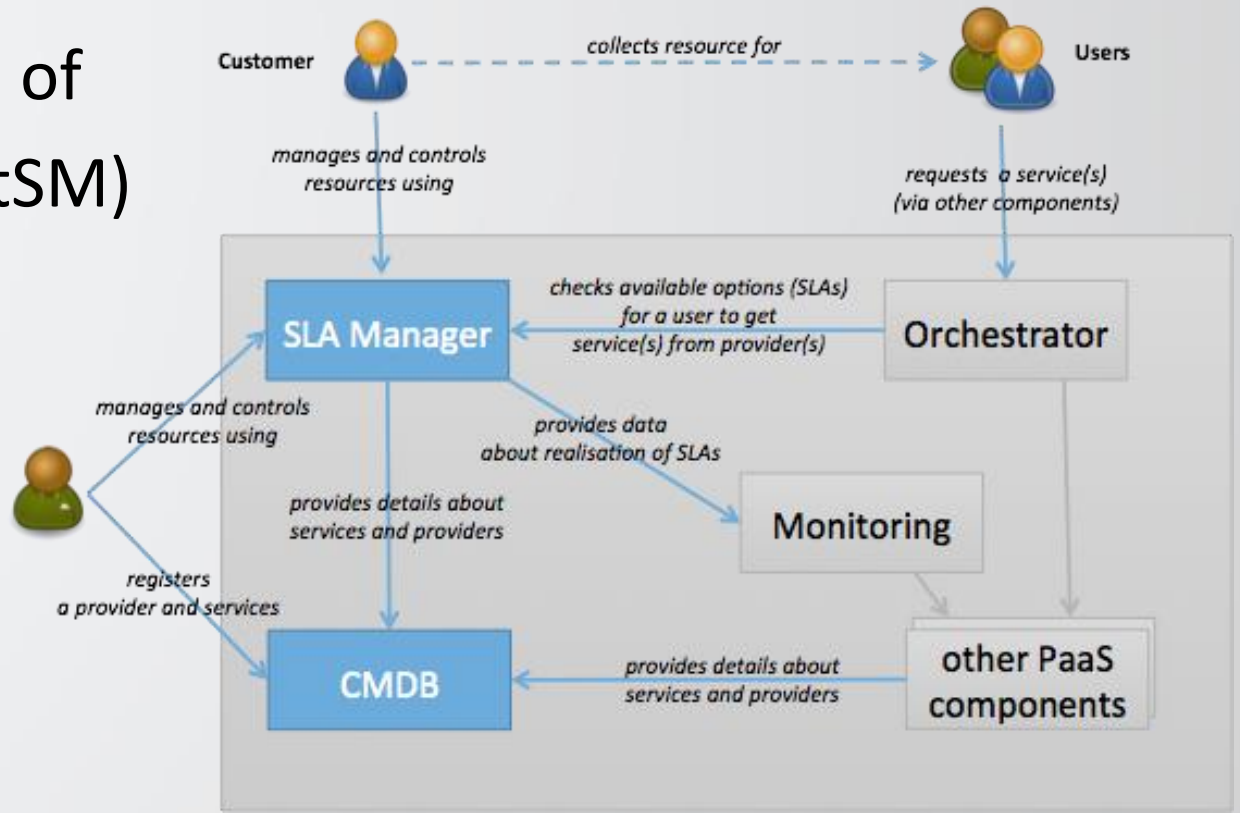
- Accounts for resource usage on the INDIGO PaaS and provides that data to other INDIGO-DataCloud services
  - QoS/SLA service will use information gathered by the Accounting service (and the monitoring pillar) to monitor SLA violation
- Usage data is extracted from the system where the resources are used and sent to a central repository
- The repository aggregates the data from across the infrastructure to produce totals based on a number of fields - such as user, site, month, year, etc.

Index

CMDB Service

- Indigo Configuration Management DB (CMDB)
  - Need to know what are INDIGO providers and services
  - need to register specific data
- Idea comes from good practices of

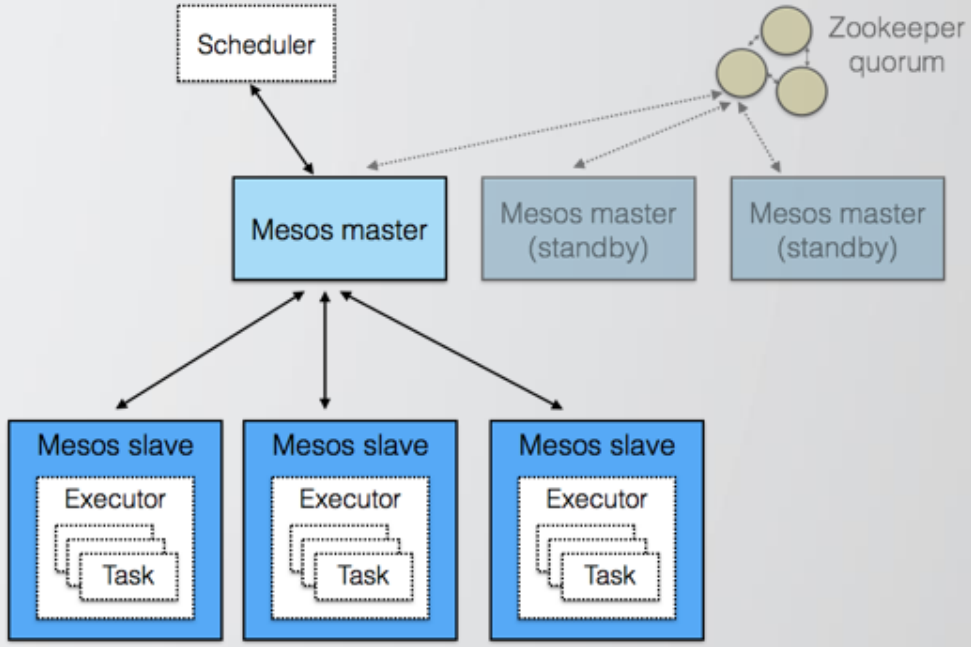
  IT Service Management (ITIL, FitSM)



Index

Managed Services/Application (MSA) Deployment Service

- This service is in charge of scheduling, spawning, executing and monitoring applications and services on a distributed infrastructure.
- The core of this component consists of an elastic Apache Mesos cluster with slave nodes dynamically provisioned and distributed on the IaaS sites.
- Apache Mesos provides efficient resource isolation and sharing across distributed applications (frameworks).

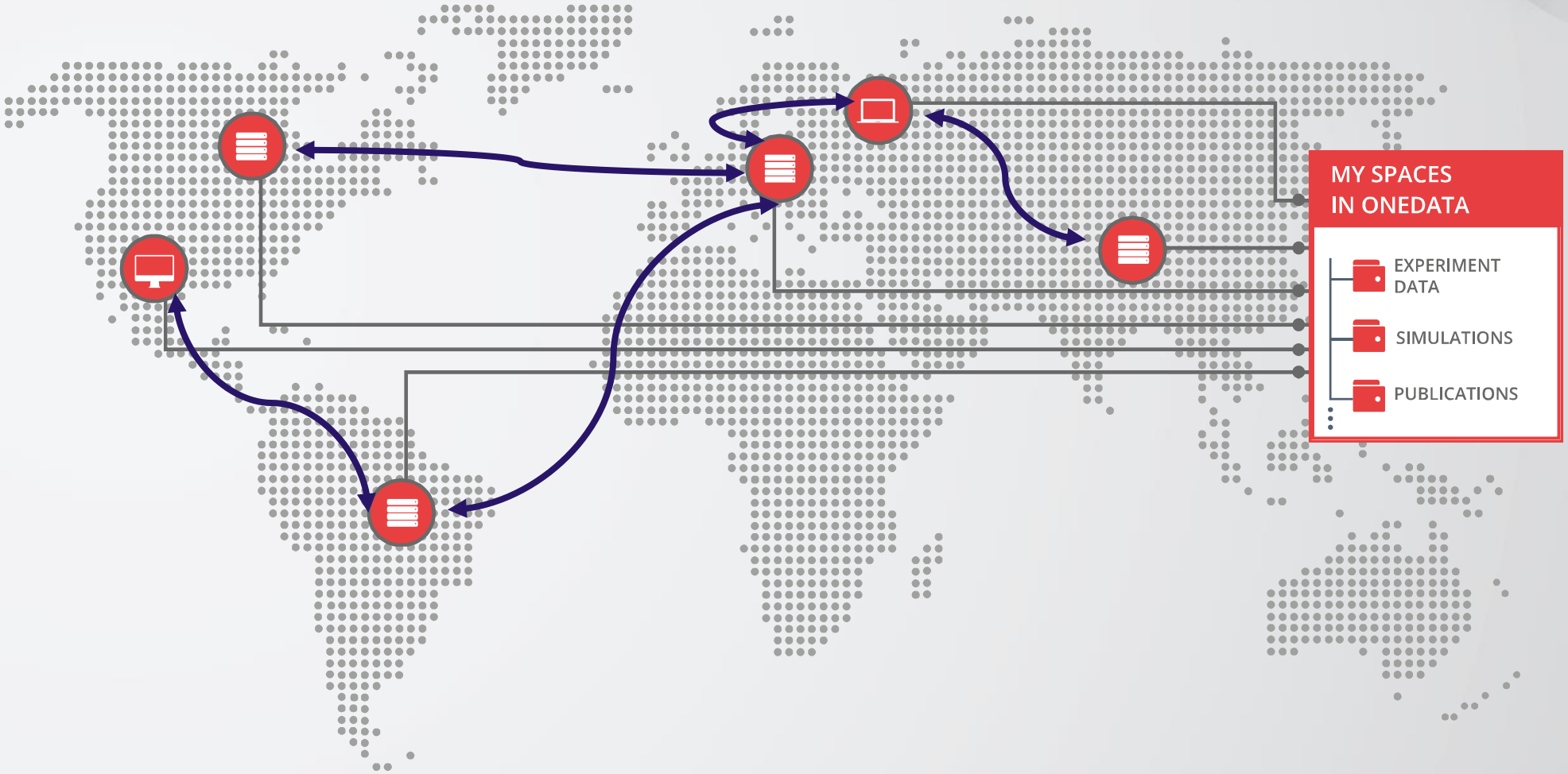


Index

- Indigo PaaS Overview
- TOSCA
- IM
- CLUES
- Orchestrator
- Cloud Provider Ranker
- Monitoring Service
- SLAM
- Accounting Service
- CMDB
- Managed Services/Application (MSA) Deployment Service
- **OneData**

Data in multi-cloud environments

INDIGO - DataCloud



Onedata – Features (I)

INDIGO - DataCloud

- Unified vision of geographically distributed data set
- Data affinity
  - Computation jobs started on resources close to data.
- Federated data access
  - Interoperability and OpenData
- Optimization and Data on the fly
  - when data is not staged

Onedata – Features (II)

INDIGO - DataCloud

- Multi-protocol transparent access to data in multi-cloud environments
  - Transparent access your data and create new one in multi-cloud environments
  - Care less about data locality, all your data are accessible wherever you go
  - Use many protocols to access the same data

- Heterogeneity of storage technologies
  - Use the same data access protocols (up to your choice) wherever you go
  - Pass-through problems of selection right storage technology to data centres operators
  - Avoid cloud vendor locking

Onedata – Features (III)

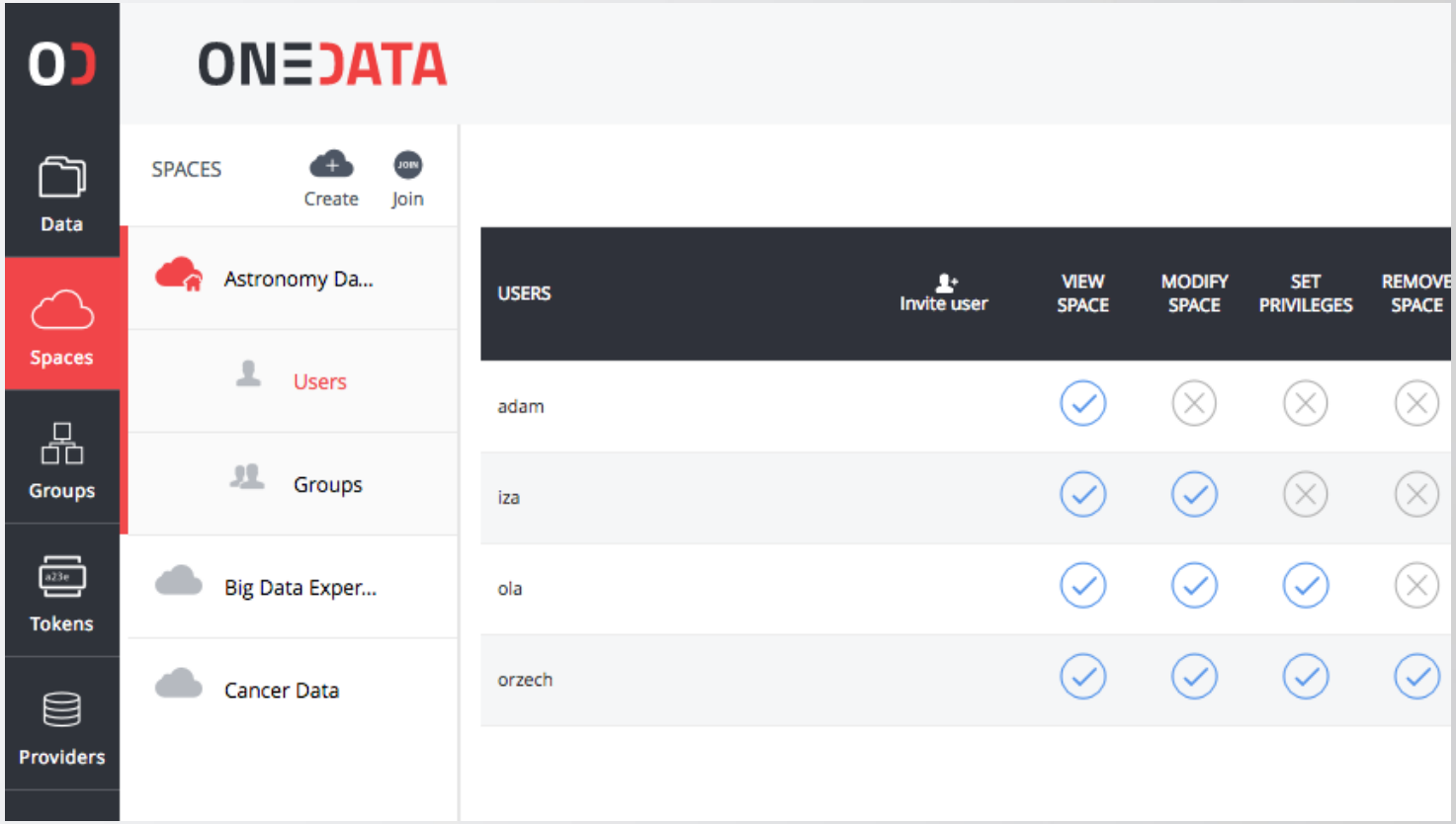
- Replica management
  - Replicate files on demand and on the fly without any additional effort
  - Migrate data between sites on demand with simple API interface
  - Easily check location of your data trough GUI or API

- Easy data sharing without borders
  - Share large scale data collection with other communities
  - Enable your data to be shared in cross-federation scenarios
  - Bring your data and tools as building blocks to European Open Science Cloud

Data sharing without borders

INDIGO - DataCloud

- Team-sharing
  - For groups
  - For individuals
  - Using tokens
- Cross-community data sharing
- Instant and ad-hoc data sharing
- *Thanks to effort supported by EGI Engage:*
  - *Open Data Publication*
  - *Handles (DOI) support*
  - *OAI-PMH*



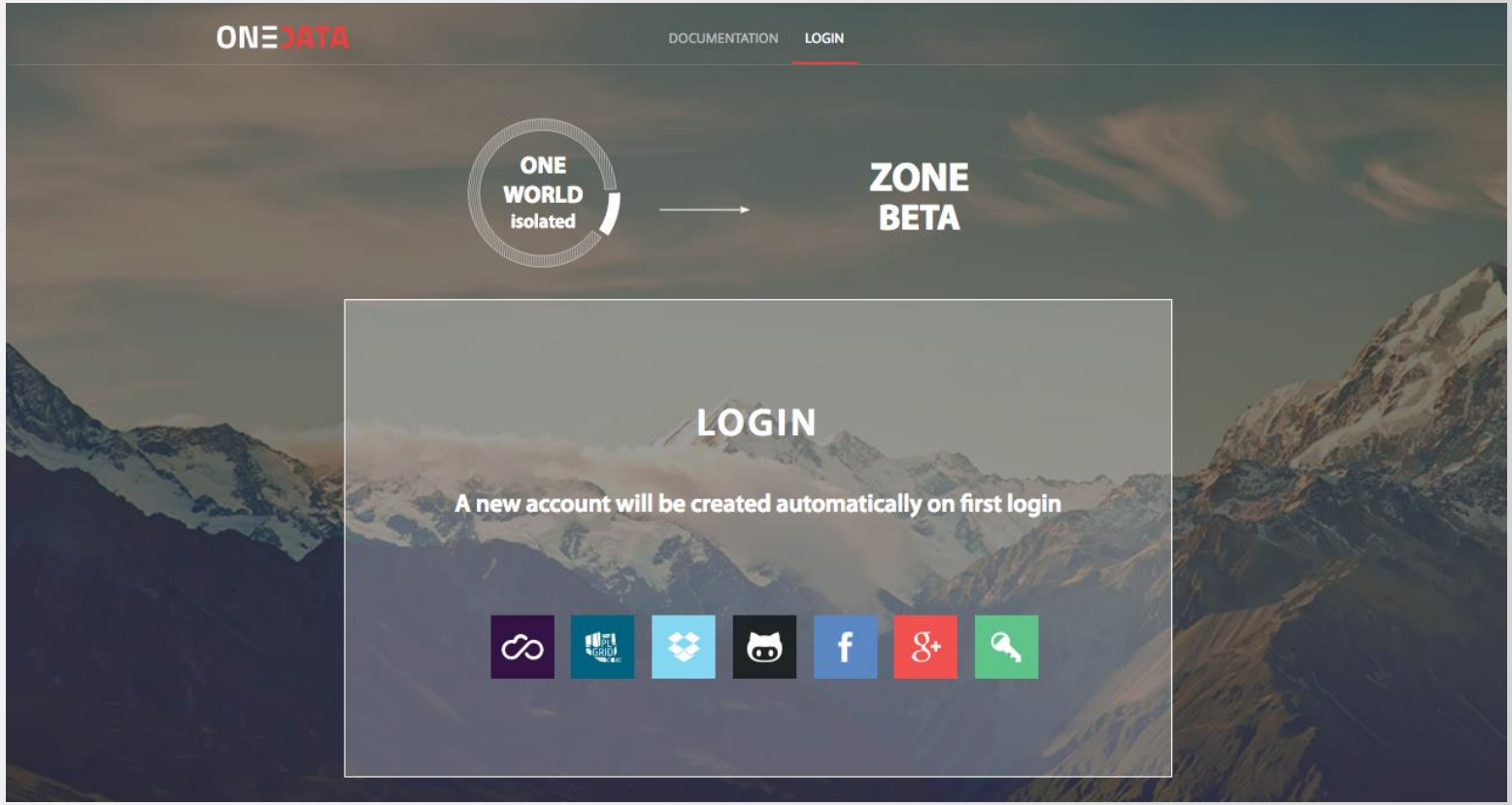
Onedata – Features (IV)

INDIGO - DataCloud

- Metadata management integrated with data management platform
  - Work with data and metadata in one system – avoiding problems of consistency
  - Monitor metadata data changes trough API in order to feed external custom systems
- Flexible authentication and authorization
  - Control who knows about your data
  - Control who can access data on a single file level
- Easy integration using API with external tools
  - Integrate external tools using rich API interfaces with data management platform building more complex environment for data processing

Flexible authentication and authorization

- Integrated with Indigo IAM
- Pluggable methods of authentication per zone
- Multi level of access control
- ACL on files and directories
- Group management
- Token based authentication (macaroons)
- X.509 in prep.



QUESTIONS?

INDIGO - DataCloud

Please visit:

www.indigo-datacloud.eu