INDIGO - DataCloud

RIA-653549

The INDIGO-DataCloud Data & Computing Platform for Scientific Communities

Davide Salomoni
INDIGO-DataCloud Project Coordinator
davide.salomoni@cnaf.infn.it

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

Talk Outline

- Cloud computing, recap
- The INDIGO-DataCloud Project
- Some background technologies
- INDIGO-DataCloud main technology advancements
- The INDIGO PaaS
- INDIGO-DataCloud use case examples
- Conclusions

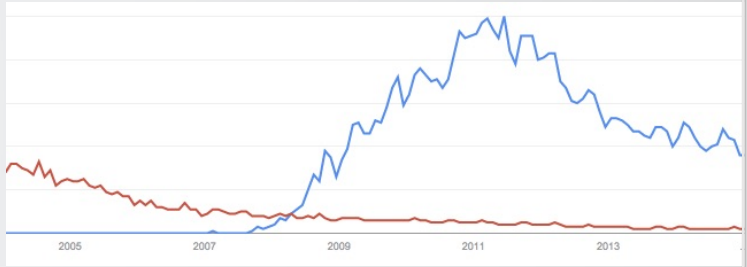
Cloud Computing

INDIGO - DataCloud

- The canonical definition of “Cloud Computing” comes from the US National Institute of Standards and Technology (NIST) (see http://goo.gl/eBGBk)

- In a nutshell, Cloud Computing deals with:

Google search trends of “Grid Computing” vs. “Cloud Computing” (from Google Trends)

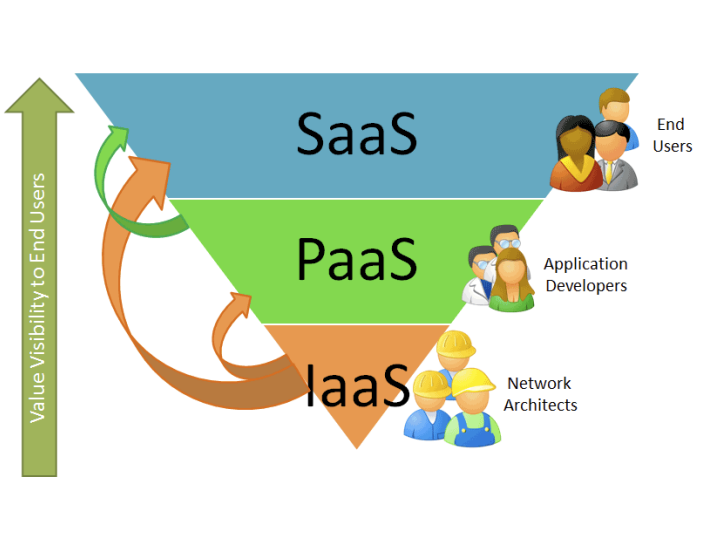


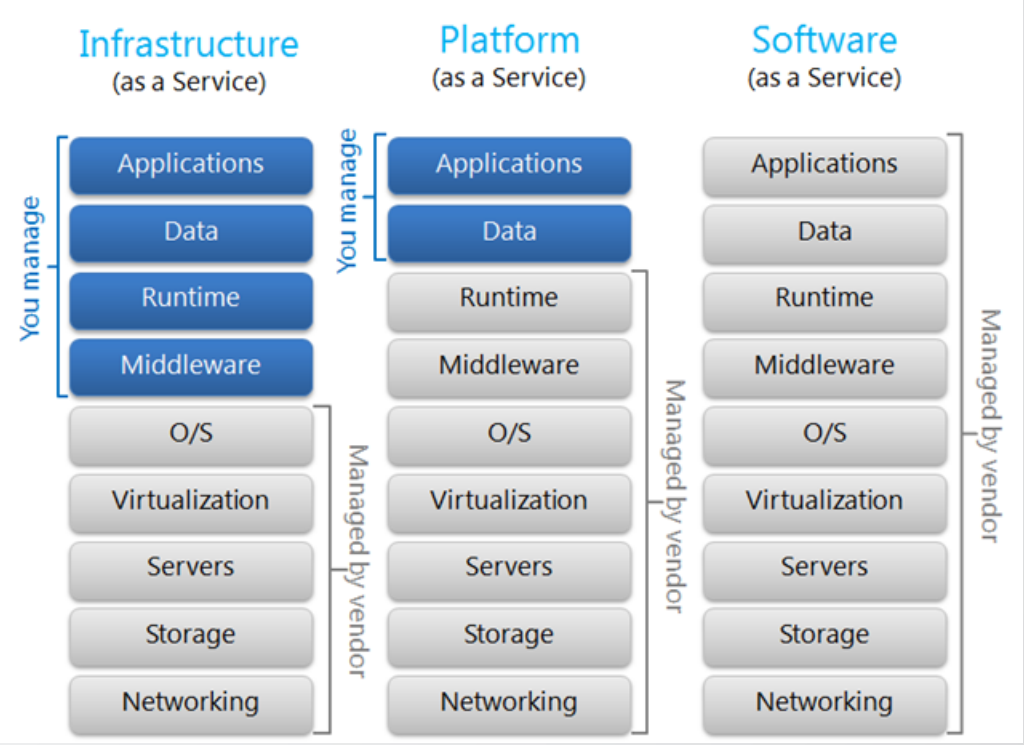
1 Provisioning
2 of information and communication technology
3 as a service

The 5 Cloud postulates

1. **Self-service, on-demand**
2. **Access through the network**
3. **Resource sharing**
4. **Elasticity (with *infinite resources*)**
5. **Pay-per-use**

What counts, at the end, *are the applications*.



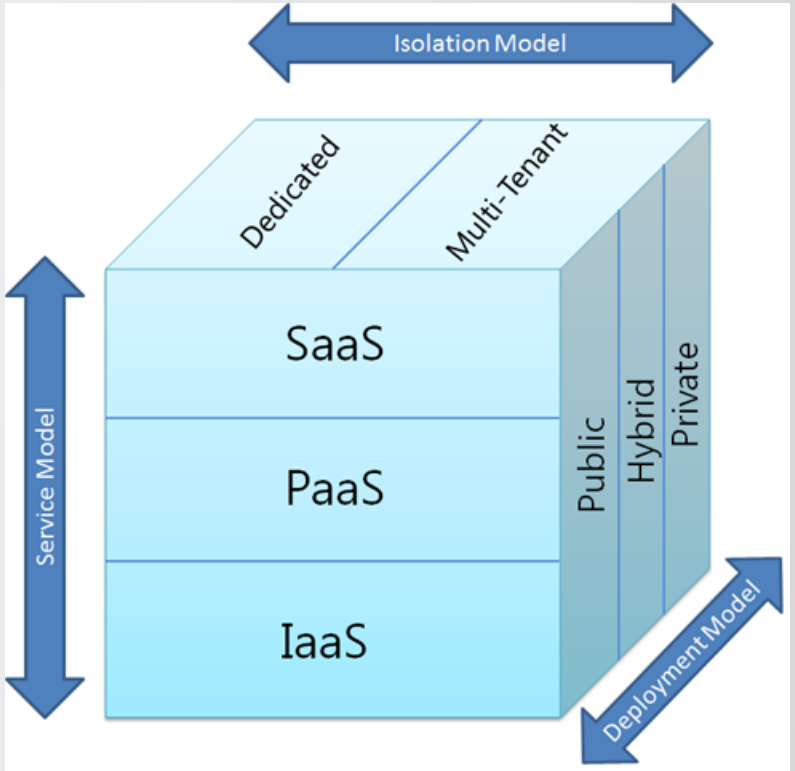


The emphasis on “service”

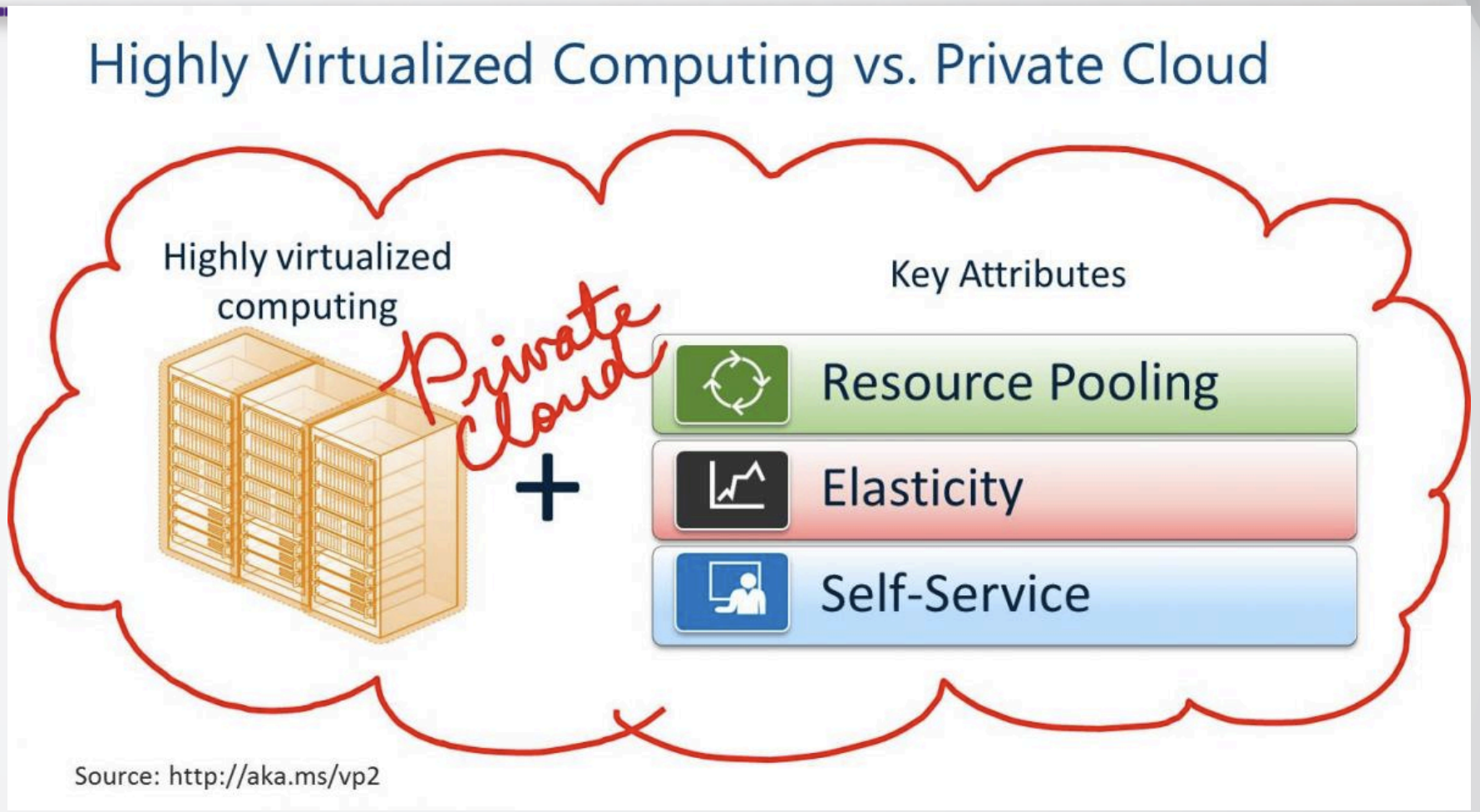
- In the Cloud Computing definition (“Provisioning of information and communication technology as a service”), **services** – their ease of use, flexibility, reliability, etc. – toward users/customers are the key point.
- Indeed, Cloud computing can be (and typically is) modelled around *services* linked primarily to:
  - Infrastructure (**IaaS** → Infrastructure as a Service)
  - Platform (**PaaS** → Platform as a Service)
  - Software (**SaaS** → Software as a Service)

More Cloud dimensions

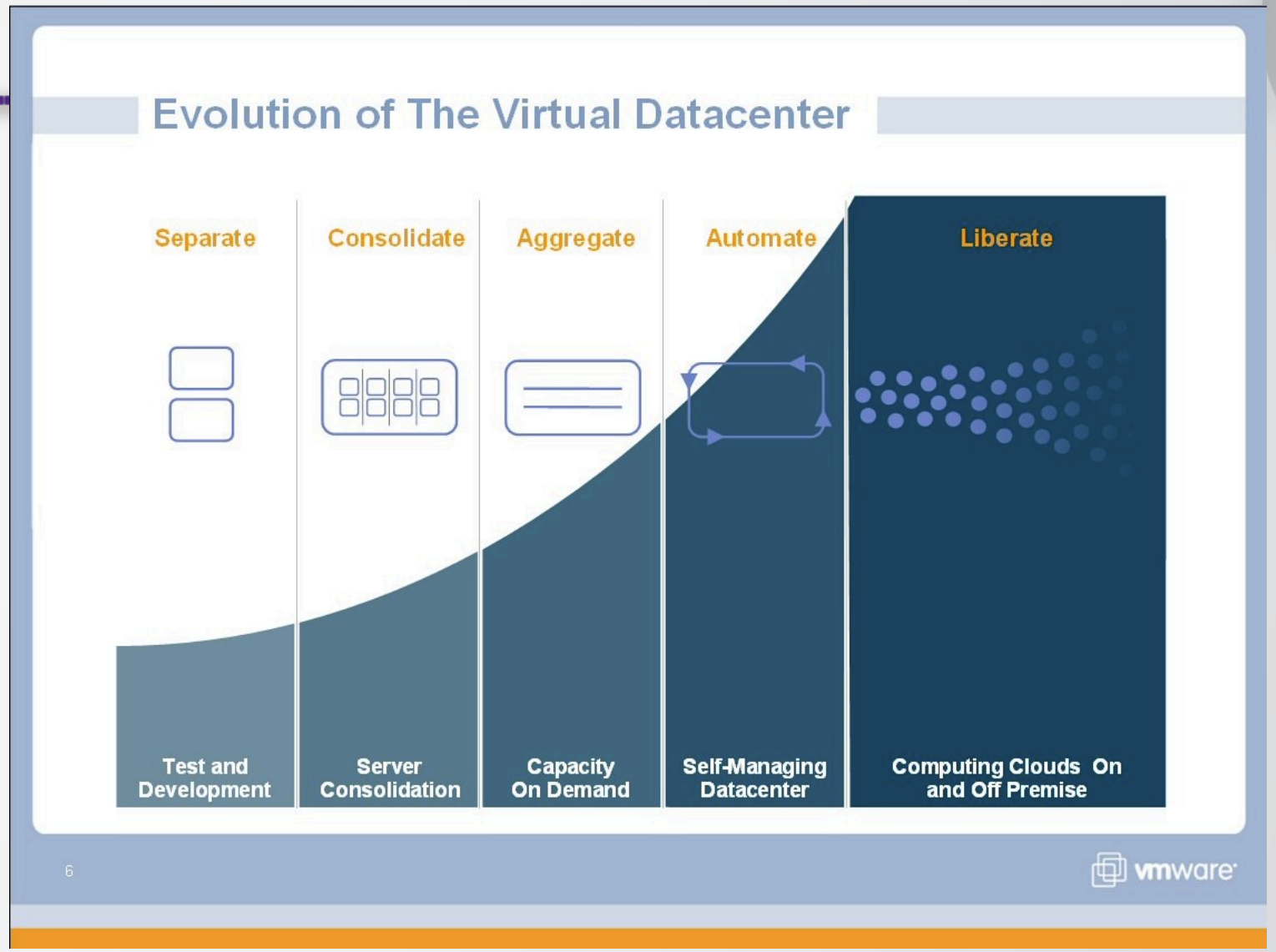
- Clouds may or may not use **virtualization technologies**
  - Useful e.g. for resource consolidation, cost/energy savings, isolation, flexibility
- Beyond the **service model**, Clouds can and should also be classified on their **isolation model** and **deployment model**.



Cloud Computing vs. Virtualization

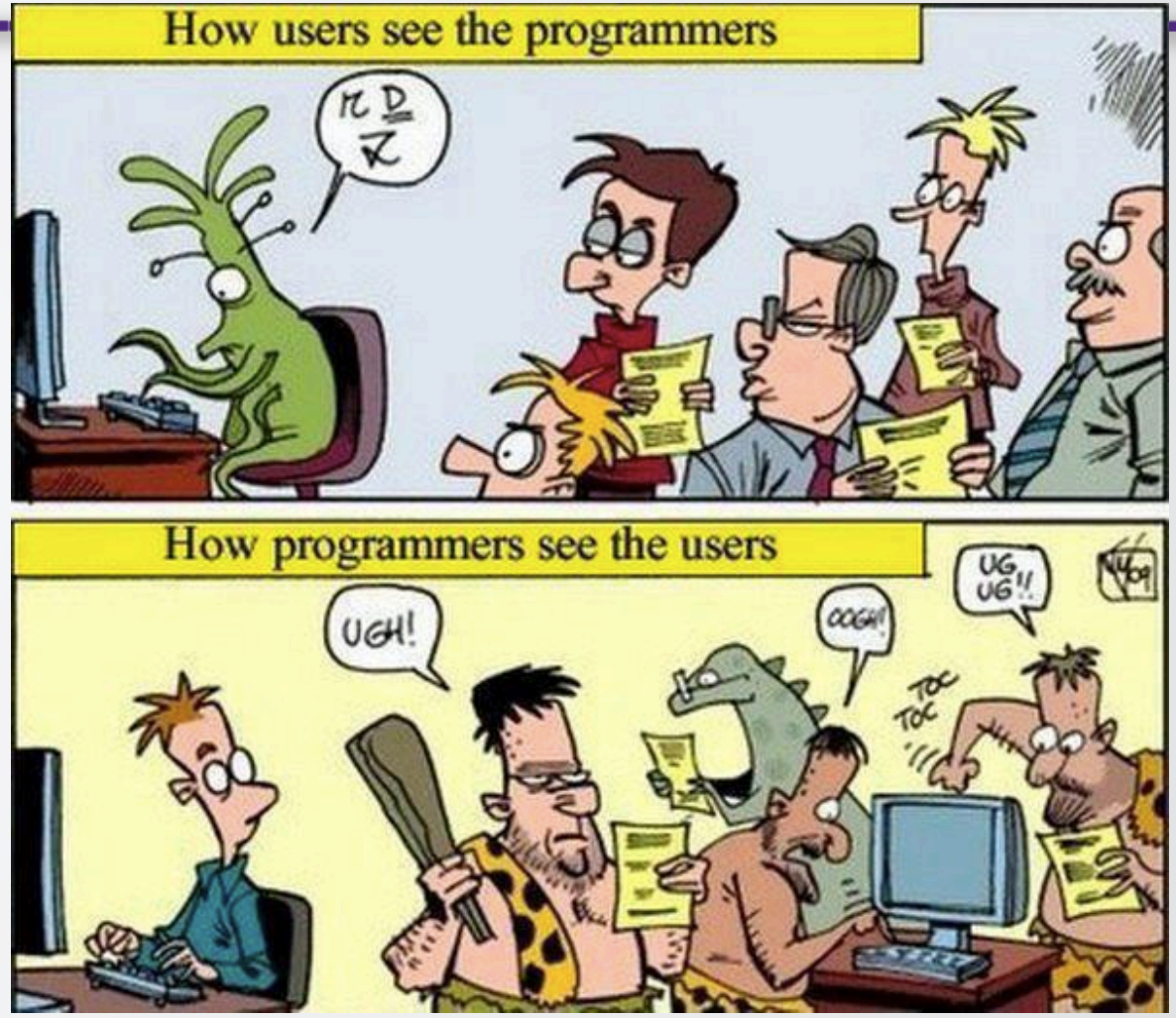


Toward the Cloud



The INDIGO-DataCloud Project

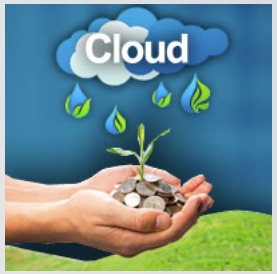
Something is missing…



Source: http://goo.gl/wT8XEq

What is missing?

1. Easy of access for both small and big communities or enterprises.
2. Financial and software.
3. Robustness (mitigation of vulnerabilities).
4. Scalable and modular architecture.
5. Open source software, independence from single providers or vendors.



From the Paper “Advances in Cloud”

- EC Expert Group Report on Cloud Computing, http://cordis.europa.eu/fp7/ict/ssai/docs/future-cc-2may-finalreport-experts.pdf

To reach the full promises of CLOUD computing, major aspects have not yet been developed and realised and in some cases not even researched. Prominent among these are **open interoperation across (proprietary) CLOUD solutions at IaaS, PaaS and SaaS levels**. A second issue is **managing multitenancy** at large scale and in heterogeneous environments. A third is **dynamic and seamless elasticity** from in- house CLOUD to public CLOUDs for unusual (scale, complexity) and/or infrequent requirements. A fourth is **data management in a CLOUD environment**: bandwidth may not permit shipping data to the CLOUD environment and there are many associated legal problems concerning security and privacy. All these challenges are opportunities towards a more powerful CLOUD ecosystem.

[…] **A major opportunity for Europe involves finding a SaaS interoperable solution across multiple CLOUD platforms.**

Another lies in migrating legacy applications without losing the benefits of the CLOUD, i.e. exploiting the main characteristics, such as elasticity etc.

Let's split the main points

- What is missing:
  - Open **interoperation** across (proprietary) CLOUD solutions at
    - IaaS,
    - PaaS,
    - and SaaS levels
  - Managing **multitenancy**
    - At large scale…
    - … and in heterogeneous environments
  - Dynamic and seamless **elasticity**
    - For both private and public cloud…
    - … and for complex or infrequent requirements
  - **Data management** in a Cloud environment
    - Due to technical…
    - … as well as to legal problems

Filling these gaps should lead to:

- **An interoperable SaaS solution covering both public and private Cloud solutions**
- **Migration of legacy applications to the Cloud**

INDIGO-DataCloud

- **An H2020 project** approved in January 2015 in the EINFRA-1-2014 call
  - 11.1M€, 30 months (**from April 2015 to September 2017**)
- **Who**: **26 European partners** in 11 European countries
  - Coordination by the Italian National Institute for Nuclear Physics (INFN)
  - Including developers of distributed software, industrial partners, research institutes, universities, e-infrastructures
- **What**: **develop an open source Cloud platform** for computing and data ("DataCloud") tailored to science.
- **For**: **multi-disciplinary scientific communities**
  - E.g. structural biology, earth science, physics, bioinformatics, cultural heritage, astrophysics, life science, climatology
- **Where**: deployable on **hybrid (public or private) Cloud infrastructures**
  - INDIGO = **IN**tegrating **D**istributed data **I**nfrastructures for **G**lobal Expl**O**itation
- **Why**: answer to the technological **needs of scientists** seeking to easily exploit distributed Cloud/Grid compute and data resources.

INDIGO Addresses Cloud Gaps

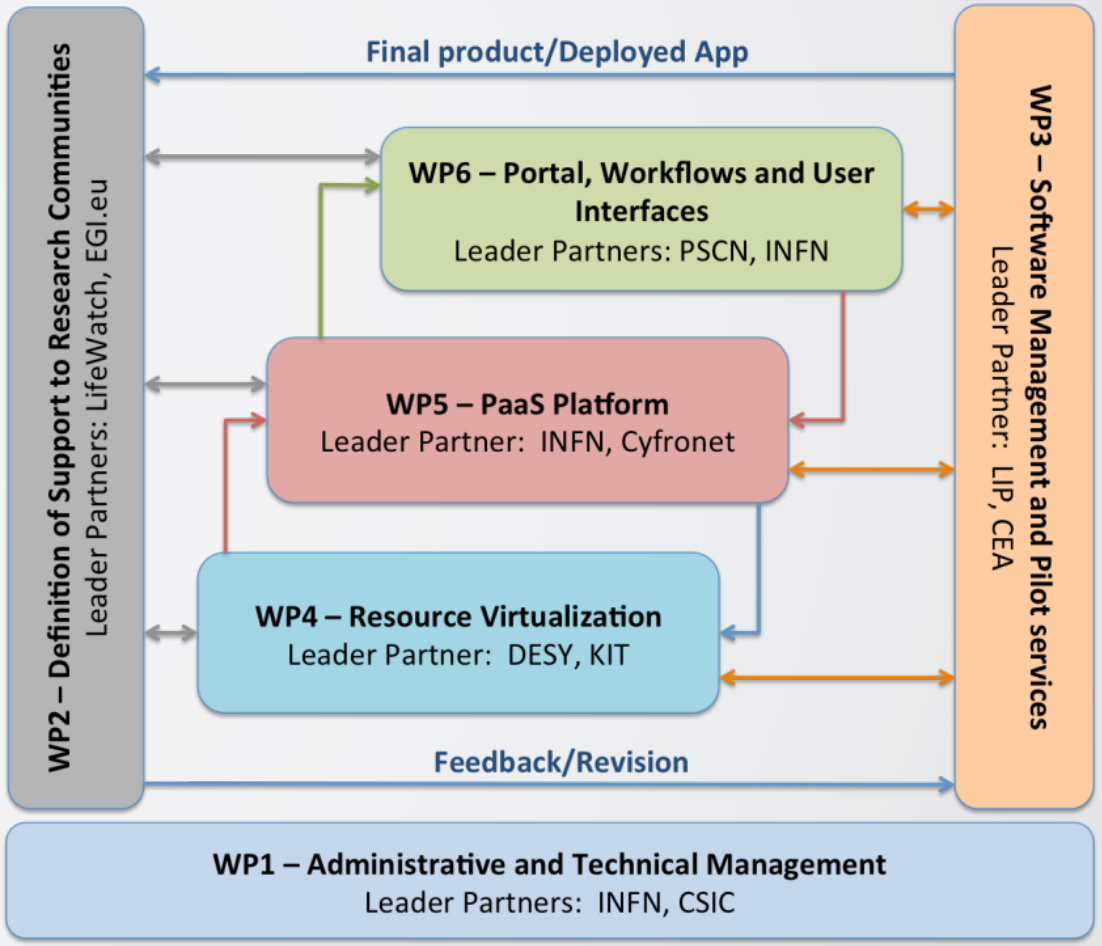
- **INDIGO focuses on use cases presented by its scientific communities** to address the gaps identified by the previously mentioned EC Report, with regard to:
  - Redundancy / reliability
  - Scalability (elasticity)
  - Resource utilization
  - Multi-tenancy issues
  - Lock-in
  - Moving to the Cloud
  - Data challenges: streaming, multimedia, big data
  - Performance
- **Reusing existing open source components** wherever possible and **contributing to upstream projects** (such as OpenStack, OpenNebula, Galaxy, etc.) for sustainability.

Users First

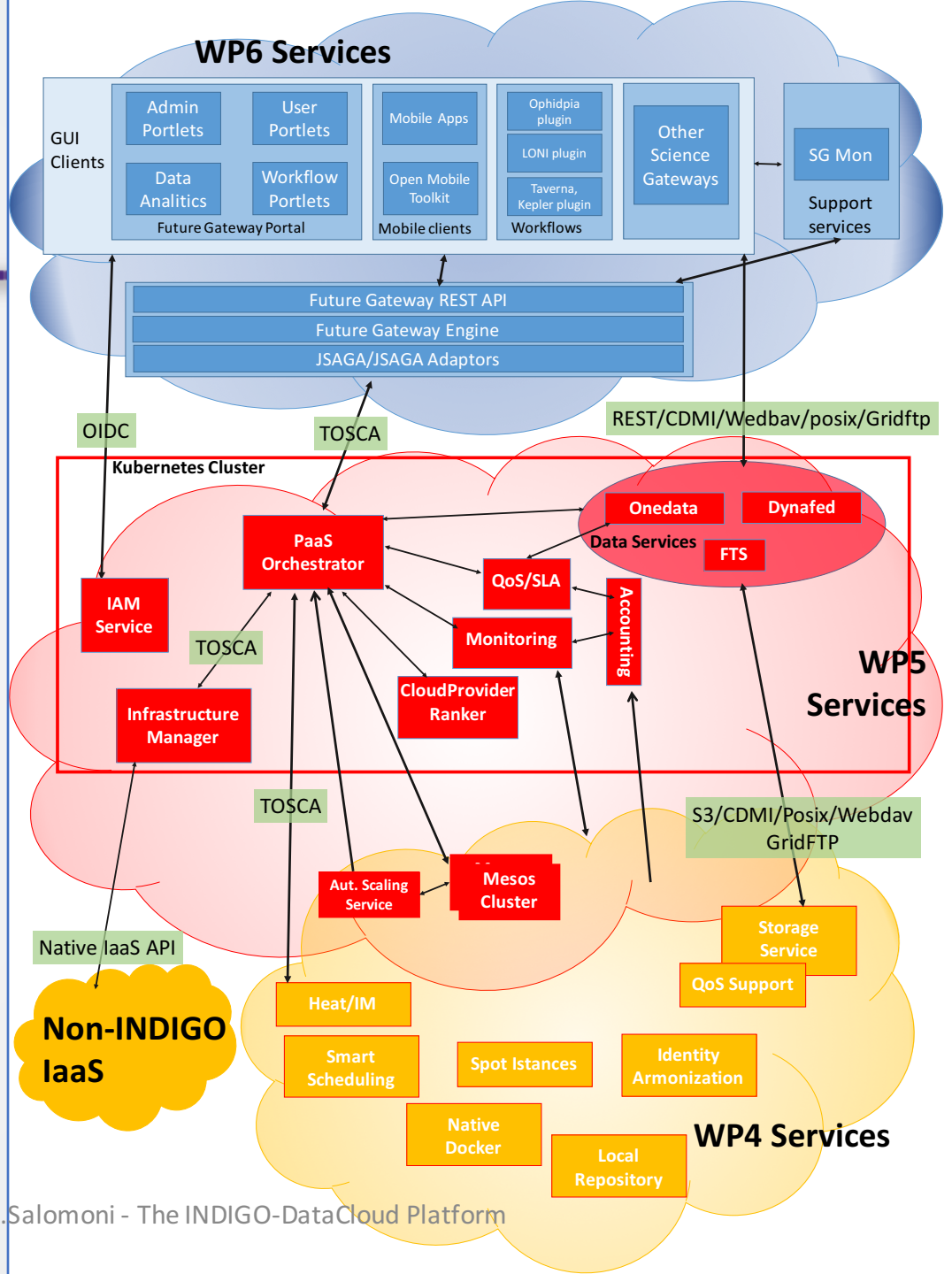
- **Requirements come from research communities**
  - "The proposal is oriented to support the use of different e-infrastructures by a wide-range of scientific communities, and aims to address a wide range of challenging requirements posed by leading-edge research activities conducted by those communities." (INDIGO DoW)
- **We gathered use cases** from 11 scientific communities.
  - LifeWatch, EuroBioImaging, INSTRUCT, LBT, CTA, WeNMR, ENES, eCulture Science Gateway, ELIXIR, EMSO, DARIAH.
- Starting from about 100 distinct requirements we derived a much shorter list, grouped **into 3 categories: Computational requirements, Storage requirements, Requirements on infrastructures**. Each requirement has an associated ranking (mandatory / convenient / optional).
- **See the INDIGO website**, Deliverable Area, for details (http://www.indigo-datacloud.eu)

#REQ	Description	Type	Rank	Proposed Improvement
CO#1	**Deployment of Interface SaaS**	Computing / PaaS	M	A mechanism to facilitate the deployment of a customised Haddock portal and backend in system in a panoply of infrastructures with minimal intervention.
CO#2	**Deployment of Customized computing back-ends as batch queues**	Computing / PaaS	M	Each instance may have an independent software configuration, potentially incompatible with other projects or specially tailored without side-effects.
CO#3	**Deployment of user-specific software**	Computing / PaaS	M	Manual installation may be cumbersome for large-scale application involving many computing resources or when requesting users to update VMIs. This should be automated.
CO#4	**Automatic elasticity of computing batch queues**	Computing / PaaS	M	When moving to the cloud, users should be provided with the exact number and size of resources they need. Overprovisioning will produce an undesirable cost or inability to serve other requests. On the other side, underprovisioning will lower QoS.
CO#5	**Terminal access to the resources.**	Computing / PaaS service	M	This feature must be linked to the AAI
CO#6	**Privileged access**	Computing / PaaS service	M	This feature must be linked to the AAI
CO#7	**Execution of workflows**	Computing / PaaS	M	Processing done on the cloud where the outputs of the processing are stored. Orchestration of complex pipelines.
CO#8	**Provenance information**	Computing / PaaS Service	C	Very important for revision of papers and project proposals.
CO#9	**Cloud bursting**	Computing /	M	Supplementing the computing capacity with special

INDIGO Work Packages

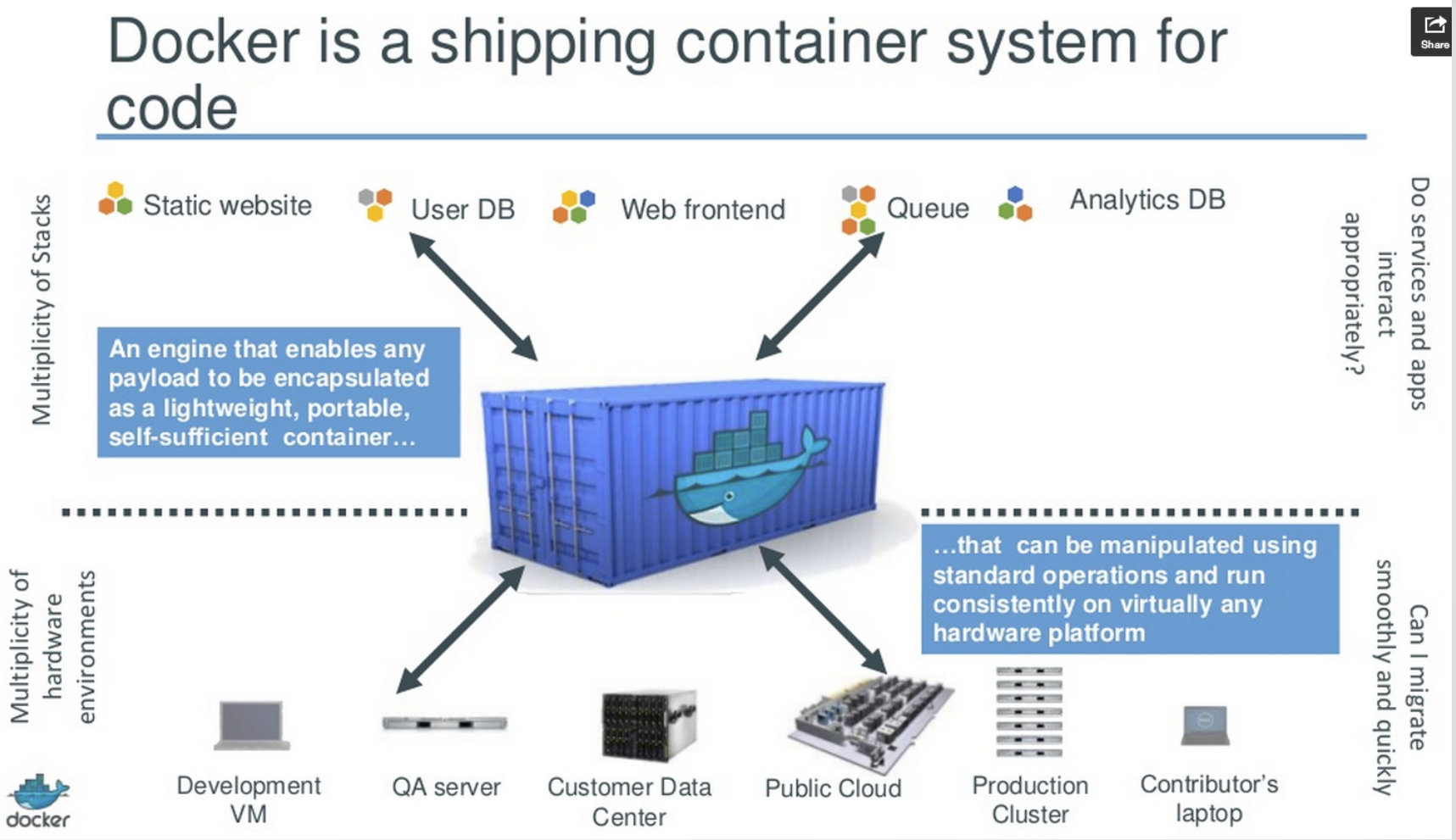


The INDIGO-DataCloud General Architecture



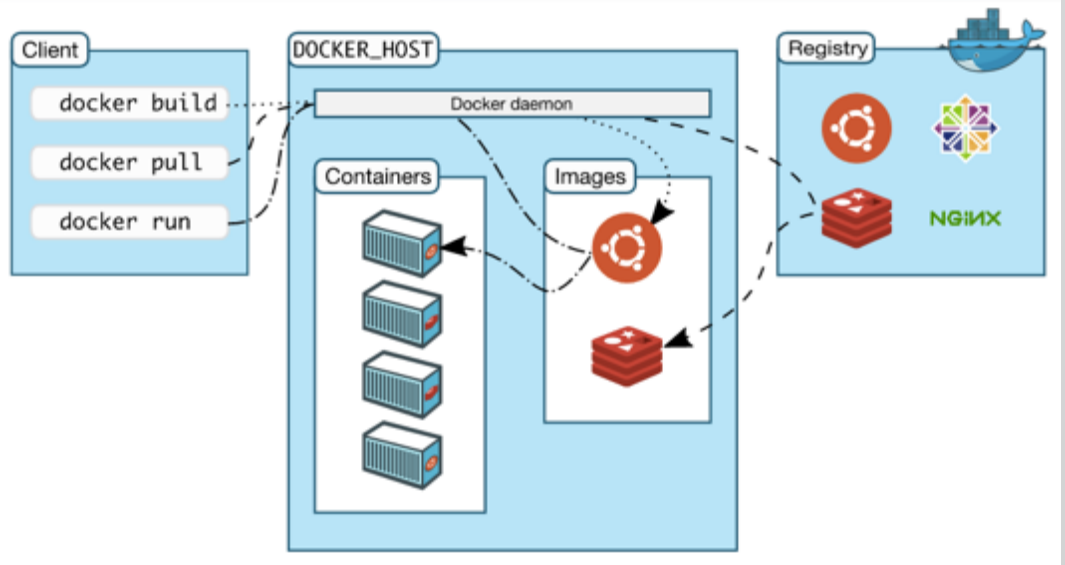
A couple of technologies used by INDIGO-DataCloud (more later)

Docker


Docker is a shipping container system for code
Share
Multiplicity of Stacks
Static website
User DB
Web frontend
Queue
Analytics DB
Do services and apps interact appropriately?
An engine that enables any payload to be encapsulated as a lightweight, portable, self-sufficient container…
…that can be manipulated using standard operations and run consistently on virtually any hardware platform
Multiplicity of hardware environments
Can I migrate smoothly and quickly
docker
Development VM
QA server
Customer Data Center
Public Cloud
Production Cluster
Contributor's laptop

Docker: Build-Ship-Run

- Docker is an open-source engine to easily create lightweight, portable, self-sufficient containers from any application.
- The same container that a developer builds and test on a laptop can run at scale, in production, on VMs, OpenStack/OpenNebula[1] clusters, public clouds and more.
- Docker features:
  - versioning (git-like)
  - component re-use
  - sharing (public repository)



[1]: thanks to ONEDock, an INDIGO-DataCloud development

Dockerfiles

- A **Dockerfile** is a script, composed of various commands (instructions) and arguments listed successively to automatically perform actions on a base image in order to create (or form) a new one.

```
FROM ubuntu
MAINTAINER Romin Irani (email@domain.com)
RUN apt-get update
RUN apt-get install -y nginx
ENTRYPOINT ["/usr/sbin/nginx","-g","daemon off;"]
EXPOSE 80
```

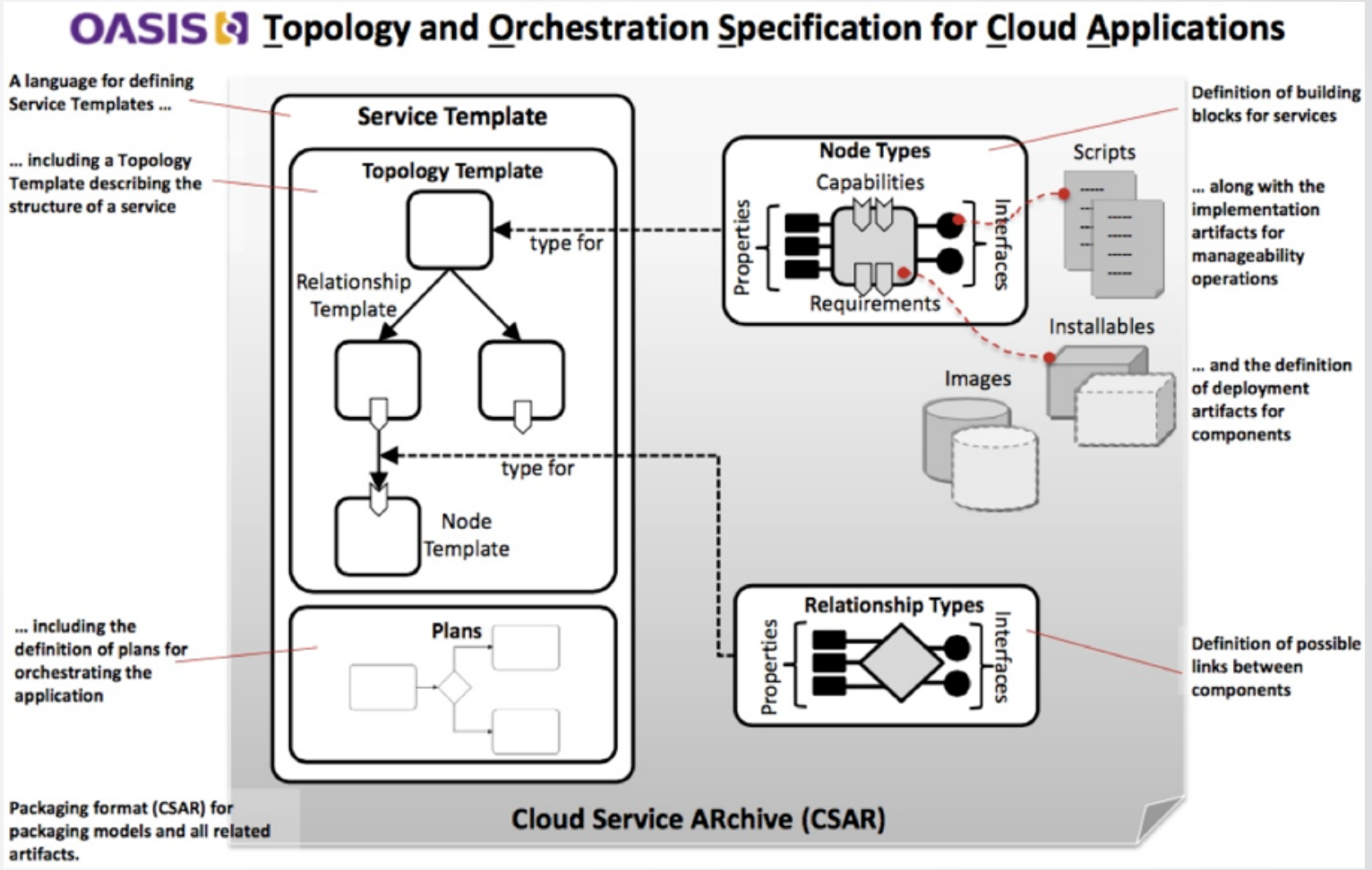
Docker: ship and run

- Ship a Docker Image
  - *docker push indigodatacloudapp/ambertools*
- Fetch a Docker Image
  - *docker pull indigodatacloudapp/ambertools*
    - download docker image from Docker Hub to a local Docker repository
- Run a docker container
  - *docker run […] indigodatacloudapp/ambertools*
    - creates a docker container out of the docker image

TOSCA

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

TOSCA in a nutshell



TOSCA topology template example

```
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-DataCloud Main Technology Advancements

The INDIGO solutions

- **The INDIGO architecture can be seen as providing:**
  - **Site-level solutions**
  - **Data solutions**
  - **Automated solutions**
  - **User-level solutions**
- All of them integrate in a consistent global framework. Frequently a given solution spans multiple INDIGO WPs.
- There are many details “behind the scenes”. But let’s focus here on a bird’s eye view from a practical perspective.

Site-level solutions

- New scheduling algorithms for open source Cloud frameworks.
- Full support for containers.
- Dynamic partitioning of batch vs. Cloud resources.
- Storage QoS and data lifecycle support.
- Support for external infrastructures.
- Automated synchronization of dockerhub repos with the local repository of open source Cloud frameworks.

Data solutions

- Integrated local and remote Posix access for all types of resources (bare metal, virtual machines, containers).
- Transparent mapping of object storage (e.g. Ceph, S3) to Posix.
- Transparent data caching and replicas.
- Transparent gateway to existing filesystems (e.g. GPFS, Lustre).
- Webdav, GridFTP, CDMI access.
- Dropbox-like functionalities (target: September 2016).
- Linux, Mac OS, Windows desktop support.

Automated solutions

They are typically based on TOSCA templates used to specify resource requirements (sample templates for common use cases are provided by the project).

- Selection of resources across multiple Cloud providers (e.g. depending on data location or resource requirements).
- Support for application requirements in Cloud resource allocations (e.g. for what regards InfiniBand or GPUs).
- Dynamic instantiation, automated monitoring and elasticity of long-running services.
- Dynamic instantiation, automated monitoring and elasticity of batch systems, front-ends included.
- Support for custom frameworks for porting arbitrary applications to the Cloud, with automated monitoring and scalability.
- Support for big data analysis applications (Ophidia, Spark).
- Mesos clusters transparently spanning multiple data centers (not in the first release).

User-level solutions

- Customizable / programmable portal (gateway) engine integrated with the features mentioned in the previous slides.
- Sample portals delivered for selected applications ("all-in-one", "plug and play" bundles).
- Mobile toolkit to access INDIGO features on mobile devices.
- AAI architecture integrated at all levels supporting X.509, SAML, OpenID Connect.

IaaS Features (1)

- **Improved scheduling for allocation of resources** by popular open source Cloud platforms, i.e. OpenStack and OpenNebula.
  - Enhancements will address both better scheduling algorithms and support for spot-instances. The latter are in particular needed to support allocation mechanisms similar to those available on public clouds such as Amazon and Google.
  - We will also support dynamic partitioning of resources among “traditional batch systems” and Cloud infrastructures (for some LRMS).
- **Support for standards in IaaS resource orchestration engines** through the use of the TOSCA standard.
  - This overcomes the portability and usability problem that ways of orchestrating resources in Cloud computing frameworks widely differ among each other.
- **Improved IaaS orchestration capabilities** for popular open source Cloud platforms, i.e. OpenStack and OpenNebula.
  - Enhancements will include the development of custom TOSCA templates to facilitate resource orchestration for end users, increased scalability of deployed resources and support of orchestration capabilities for OpenNebula.

IaaS Features (2)

INDIGO - DataCloud

- **Improved QoS capabilities of storage resources**.
  - Better support of high-level storage requirements such as flexible allocation of disk or tape storage space and support for data life cycle. This is an enhancement also with respect to what is currently available in public clouds, such as Amazon Glacier and Google Cloud Storage.
- **Improved capabilities for networking support**.
  - Enhancements will include flexible networking support in OpenNebula and handling of network configurations through developments of the OCCI standard for both OpenNebula and OpenStack.
- **Improved and transparent support for Docker containers**.
  - Introduction of native container support in OpenNebula, development of standard interfaces using the OCCI protocol to drive container support in both OpenNebula and OpenStack.

PaaS Features (1)

- **Improved capabilities in the geographical exploitation of Cloud resources**.
  - End users need not know where resources are located, since the INDIGO PaaS layer is hiding the complexity of both scheduling and brokering.
- **Standard interface to access PaaS services**.
  - Currently, each PaaS solution available on the market is using a different set of APIs, languages, etc. INDIGO uses the TOSCA standard to hide these differences.
- **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**.
  - The INDIGO resource orchestrator is capable of addressing both types of Cloud infrastructures through TOSCA templates handled at either the PaaS or IaaS level.

PaaS Features (2)

- **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.
- **Integrated IaaS and PaaS support in resource allocations**.
  - For example, storage provided at the IaaS layer is automatically made available to higher-level allocation resources performed at the PaaS layer.
- **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 storage solutions are capable of providing efficient access to data and of transparently connecting to Posix filesystems already available in data centers.

PaaS Features (3)

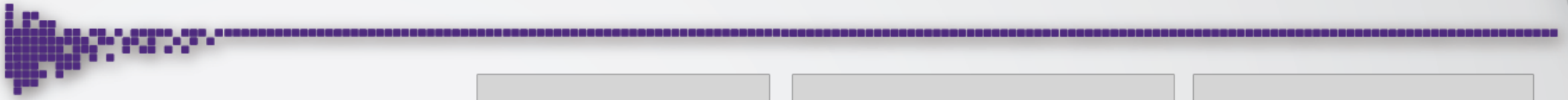
INDIGO - DataCloud

- **Deployment, monitoring and automatic scalability of existing applications**.
  - For example, existing applications such as web front-ends or R-Studio servers can be automatically and dynamically deployed in highly-available and scalable configurations.
- **Integrated support for high-performance Big Data analytics**.
  - This includes custom frameworks such as Ophidia (providing a high performance workflow execution environment for Big Data Analytics on large volumes of scientific data) as well as general purpose engines for large-scale data processing such as Spark, all integrated to make use of the INDIGO PaaS features.
- **Support for dynamic and elastic clusters of resources**.
  - Resources and applications can be clustered through the INDIGO APIs. This includes for example batch systems on-demand (such as HTCondor or Torque) and extensible application platforms (such as Apache Mesos) capable of supporting both application execution and instantiation of long-running services.

AAI Features

- INDIGO provides an advanced **set of AAI features** that includes:
  - User authentication (supporting SAML, OIDC, X.509)
  - Identity harmonization (link heterogeneous AuthN mechanisms to a single VO identity)
  - Management of VO membership (i.e., groups and other attributes)
  - Management of registration and enrolment flows
  - Provisioning of VO structure and membership information to services
  - Management, distribution and enforcement of authorization policies
  - A Token Translation Service (TTS), creating credentials for services that do not natively support OpenID Connect. Services that do not support OpenID Connect are for example ssh, S3 storage, OpenNebula.

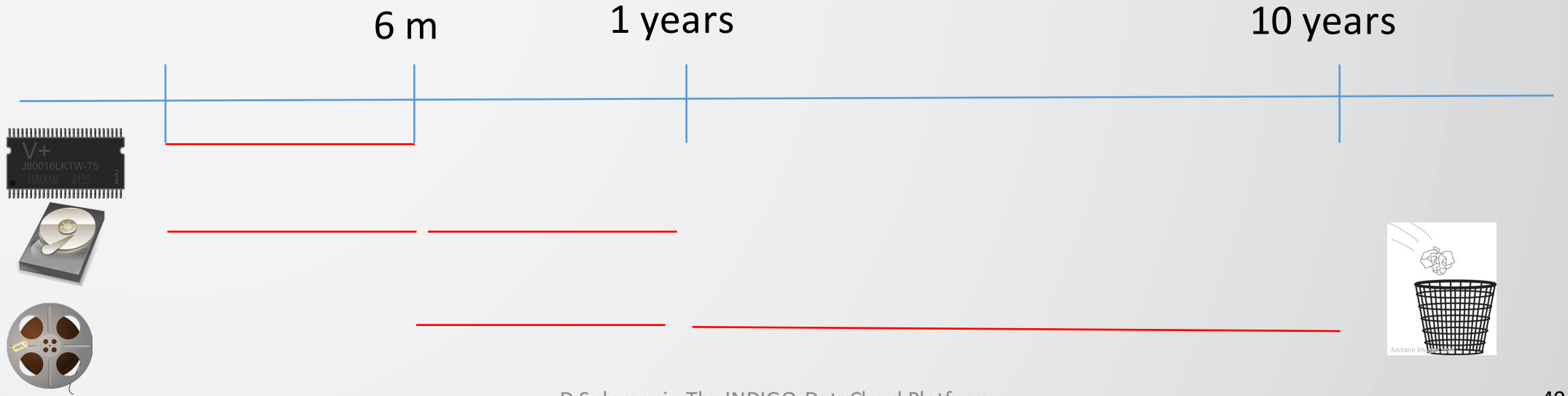
Storage Quality of Service and the Cloud



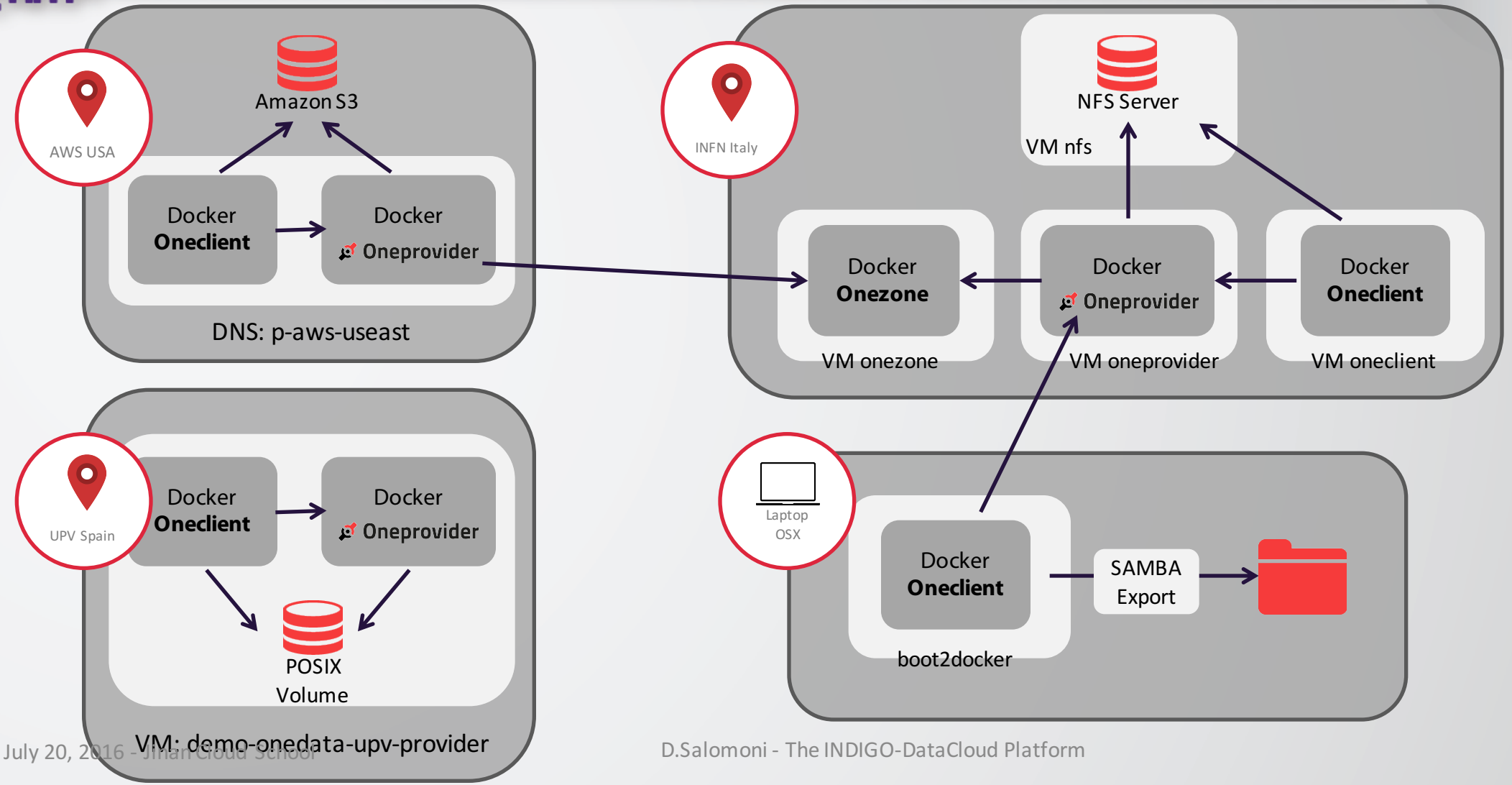
Amazon	S3		Glacier
Google	Standard	Durable Reduces Availability	Nearline
HPSS/GPFS	Corresponds to the HPSS Classes (customizable)		
dCache	Resilient	disk+tape	TAPE

The next step: Data Life Cycle

- **Data Life Cycle** is just the time dependent change of
  - Storage Quality of Service
  - Ownership and Access Control (PI Owned, no access, Site Owned, Public access)
  - Payment model: Pay as you go ; Pay in advance for rest of lifetime.
  - Maybe other things



Data Federation through INDIGO Onedata

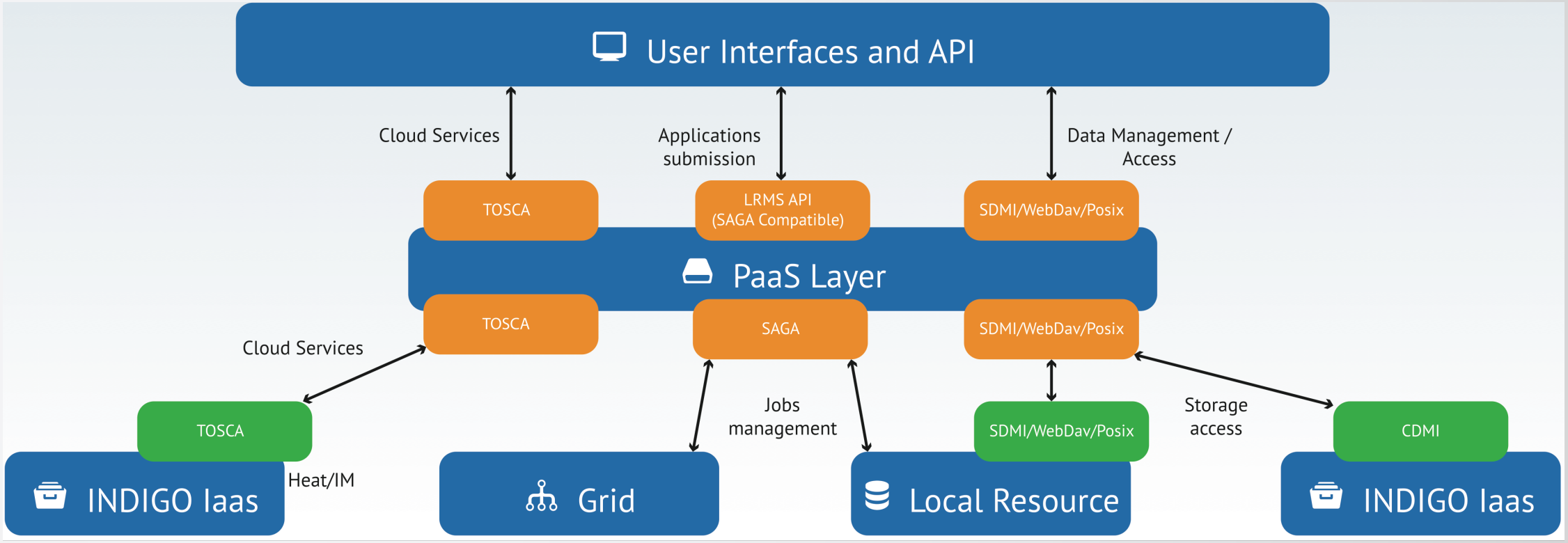


Front-end integration schemas

- We provide graphical user interfaces in the form of **scientific gateways and workflows** through the INDIGO FutureGateway (FG). The FG can directly access the INDIGO PaaS services and software stack and allows to define and set up on-demand infrastructures for the use cases presented by our scientific communities.
  - Setting up whole use case infrastructure: administrators will be provided with **ready-to-use receipts** that he will be able to customize. Final users will be provided with the service endpoints and will not be aware of the backend.
  - Use the INDIGO features from their own Portals: user communities, having their **own Scientific Gateway setup**, can exploit the FutureGateway REST API to deal with the entire INDIGO software stack.
  - Use of the INDIGO tools and portals, including the FutureGateway, Scientific Workflows Systems, Big Data Analytics Frameworks (such as Ophidia), Mobile Applications or Kepler extensions. In this scenario, final users as well as domain administrators will use GUI tools. Administrators will use it as described in the first case. In addition, domain-specific users will be provided with specific portlets/workflows/apps to allow graphical interaction with their applications run via the INDIGO software stack.

The INDIGO PaaS

The INDIGO PaaS

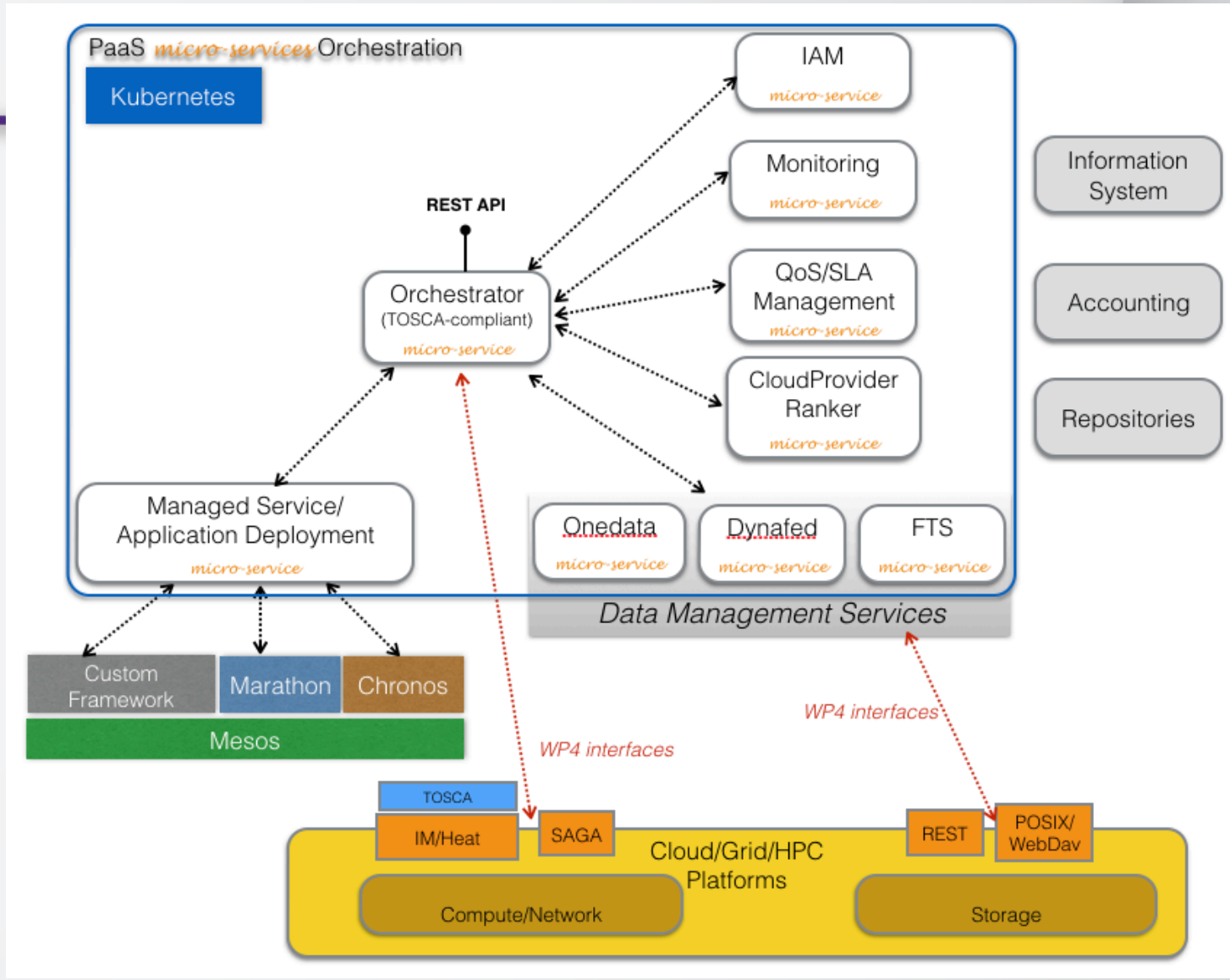


The INDIGO PaaS Architecture

The INDIGO PaaS core is built upon a set of services (exposing REST interfaces) that are:

- Deployed
- Scaled
- Managed
- Upgraded
- Monitored
- Self-healed

through **Kubernetes** (http://kubernetes.io), an open source system for managing containerized applications across multiple hosts in a cluster.



The INDIGO Orchestrator Service

- The Orchestrator **coordinates the deployment process** over the IaaS platforms.
- It collects all the information needed to deploy a service consuming other 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.
- It exposes **REST APIs** and supports the **TOSCA Simple Profile in YAML Version 1.0** specification.

The INDIGO 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.
  - It's 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 are implemented using the largely diffused **Drools Rule Engine** runtime framework

The INDIGO Monitoring Service

- The Monitoring service provides a comprehensive REST API to gather information about:
  - The **PaaS Core Services**
    - Using *Heapster* in the *Kubernetes* cluster.
  - **Customized virtual infrastructures** not using directly the PaaS
    - Using *Zabbix* agents deployed inside VMs/containers
  - The **state of the sites** (leveraging the EGI FedCloud approach)

The INDIGO QoS/SLA Service

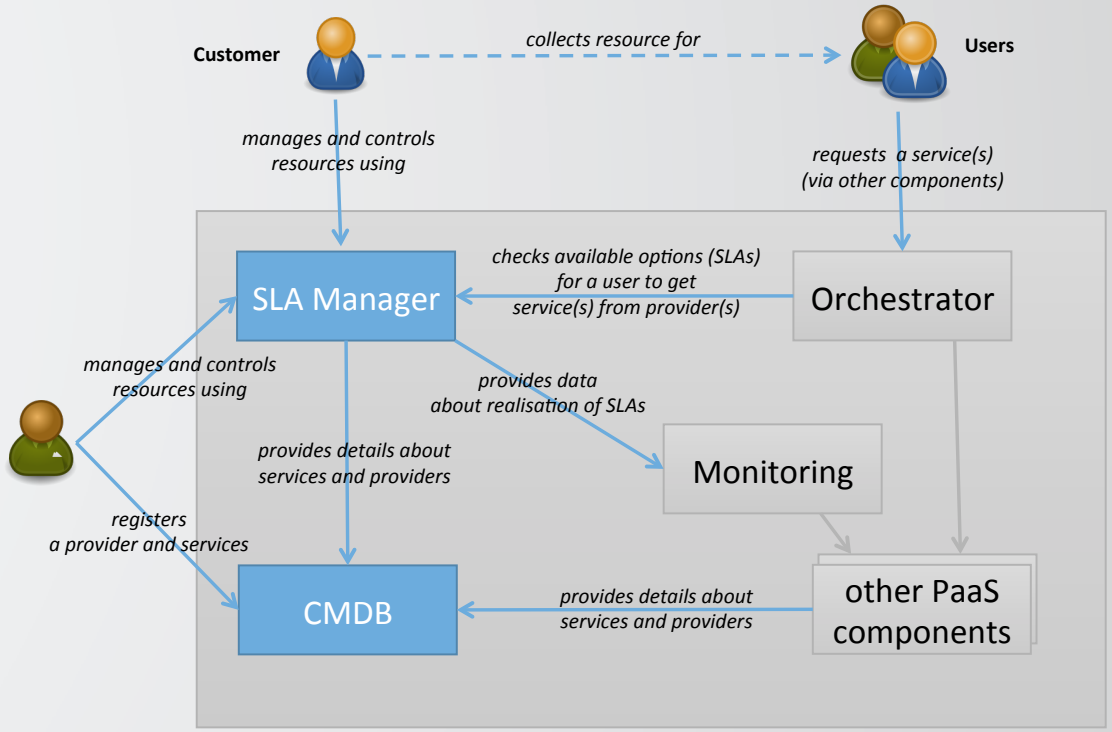
- Allows the **handshake** between a **user** and a **site** on a given **SLA**
- Provides the Orchestrator/Ranker with useful information for taking decisions on tasks scheduling according to agreed and valid SLAs
- Describes the **QoS** that a specific user/group has both over a given site or generally 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).

The INDIGO Accounting Service

- 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.

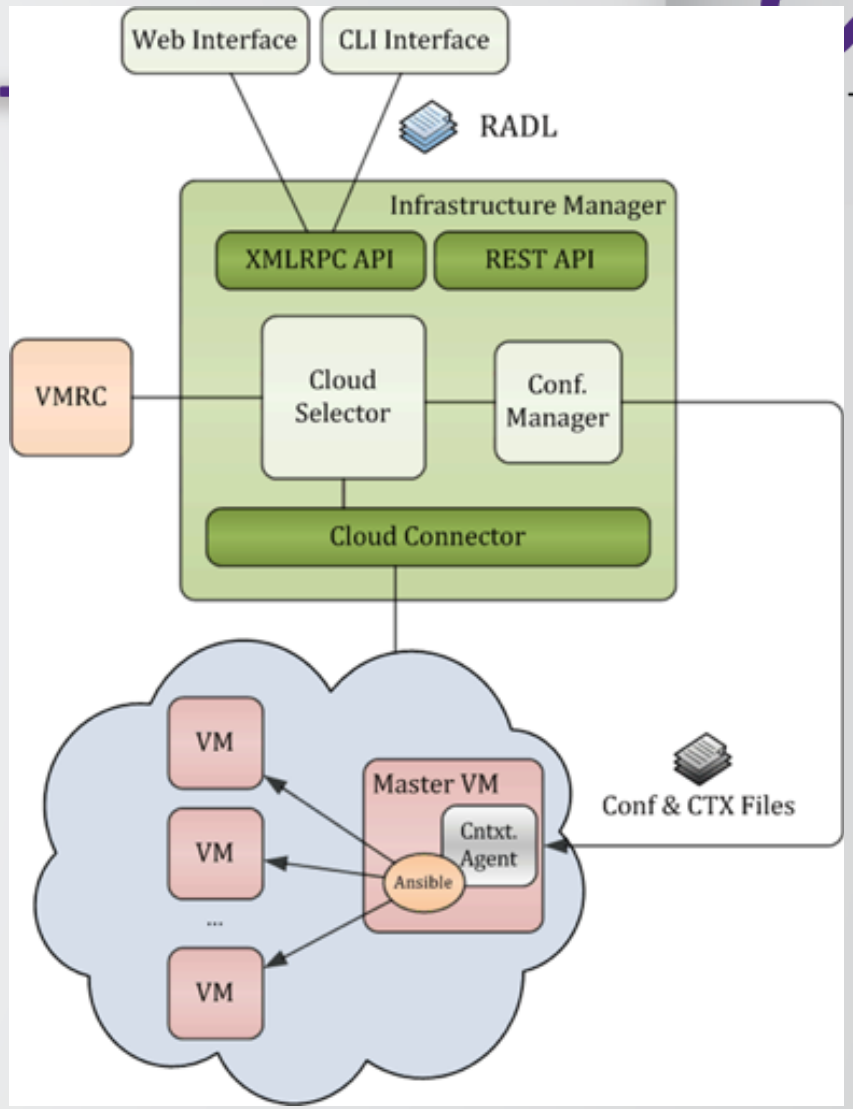
The INDIGO CMDB Service

- This is the **INDIGO Configuration Management DB** (CMDB)
  - It needs to know what are INDIGO providers and services
  - It needs to register specific data (not covered by the GocDB)
- The idea comes from good practices of IT Service Management (ITIL, FitSM)



The INDIGO Infrastructure Manager (IM) Service

- The IM is a service for the whole orchestration of virtual infrastructures and applications deployed on them, including resource provisioning, deployment, configuration, re-configuration and termination.
- A configuration manager based on Ansible configures the VMs deployed by the cloud connector and installs the necessary software.
- IM supports APIs from a large number of virtual platforms, making user applications cloud-agnostic.
- IM has been extended in INDIGO to support the TOSCA Simple Profile in YAML Version 1.0 for infrastructure description.
- Documentation: http://www.grycap.upv.es/im

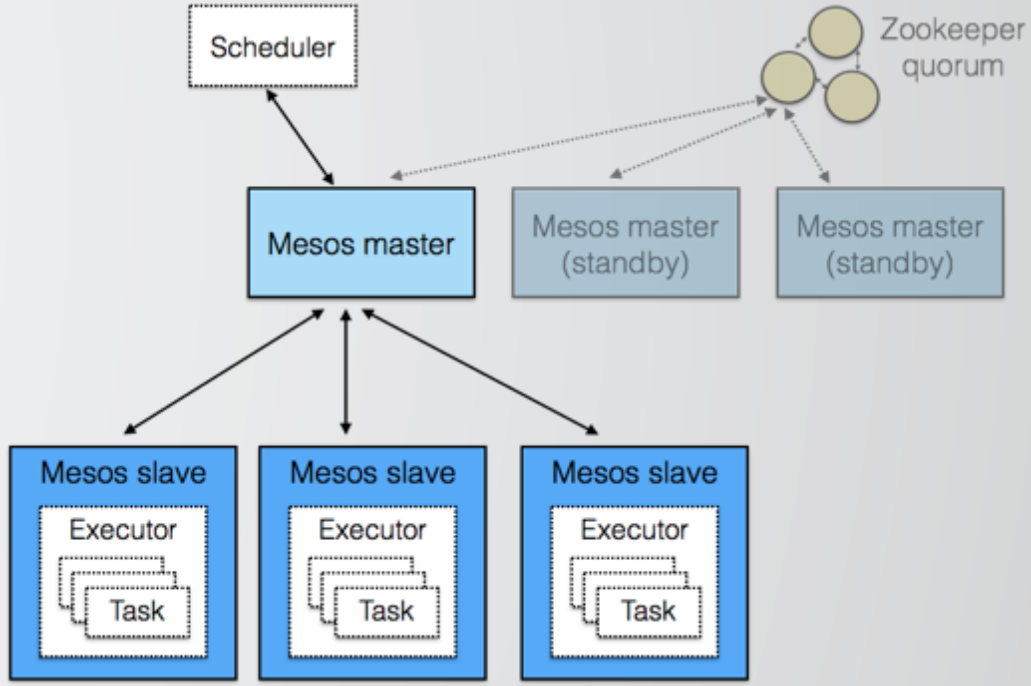


The INDIGO Automatic Scaling Service

- It extends the EC3 CLUES software adding the interfaces required to interact with the INDIGO Orchestrator
  - Documentation: http://www.grycap.upv.es/ec3
- It implements the elasticity rules considering the state of the virtual cluster.
- The virtual cluster will deploy additional worker nodes as required, and integrate them with 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

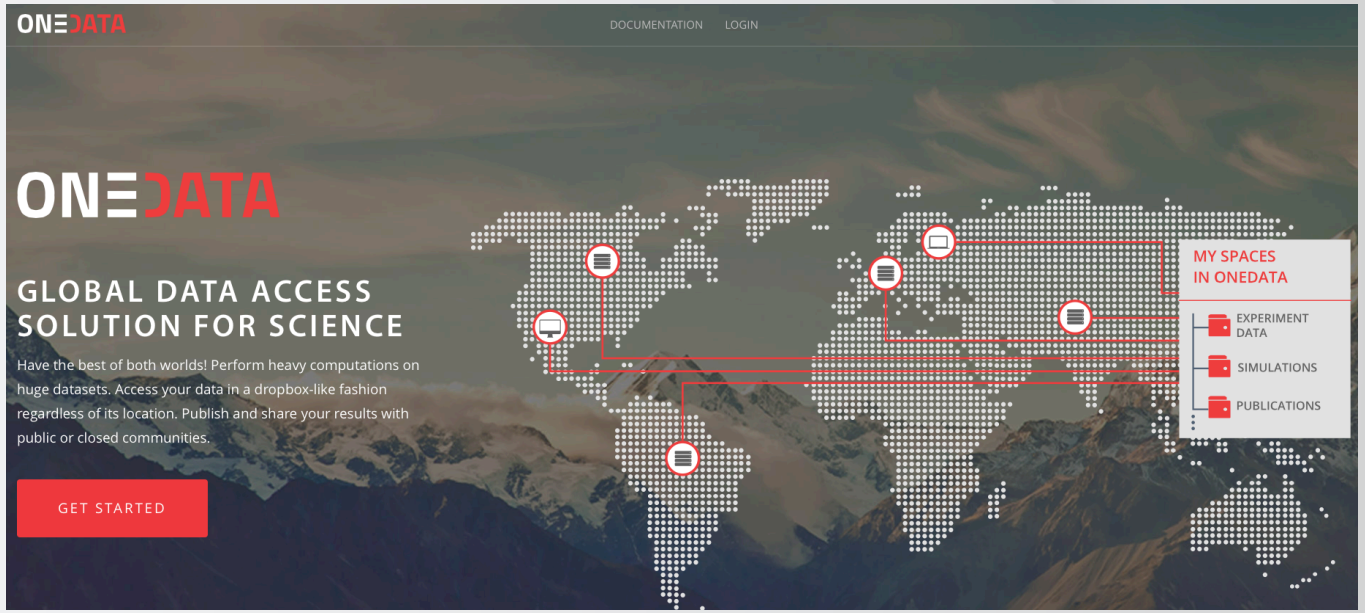
The INDIGO Managed Services/Application (MSA) Deployment Service

- 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).



The INDIGO Data Services

- Unified vision of geographically distributed data sets
- Data affinity
  - Computation jobs started on resources close to data.
- Federated data access
  - Interoperability and Open Data
- Optimization and Data on the fly



- Technologies used, integrated and in some cases extended:
  - Onedata, https://onedata.org
  - Dynafed, http://lcgdm.web.cern.ch/dynafed-dynamic-federation-project
  - FTS, http://egee-jra1-dm.web.cern.ch/egee-jra1-dm/FTS/

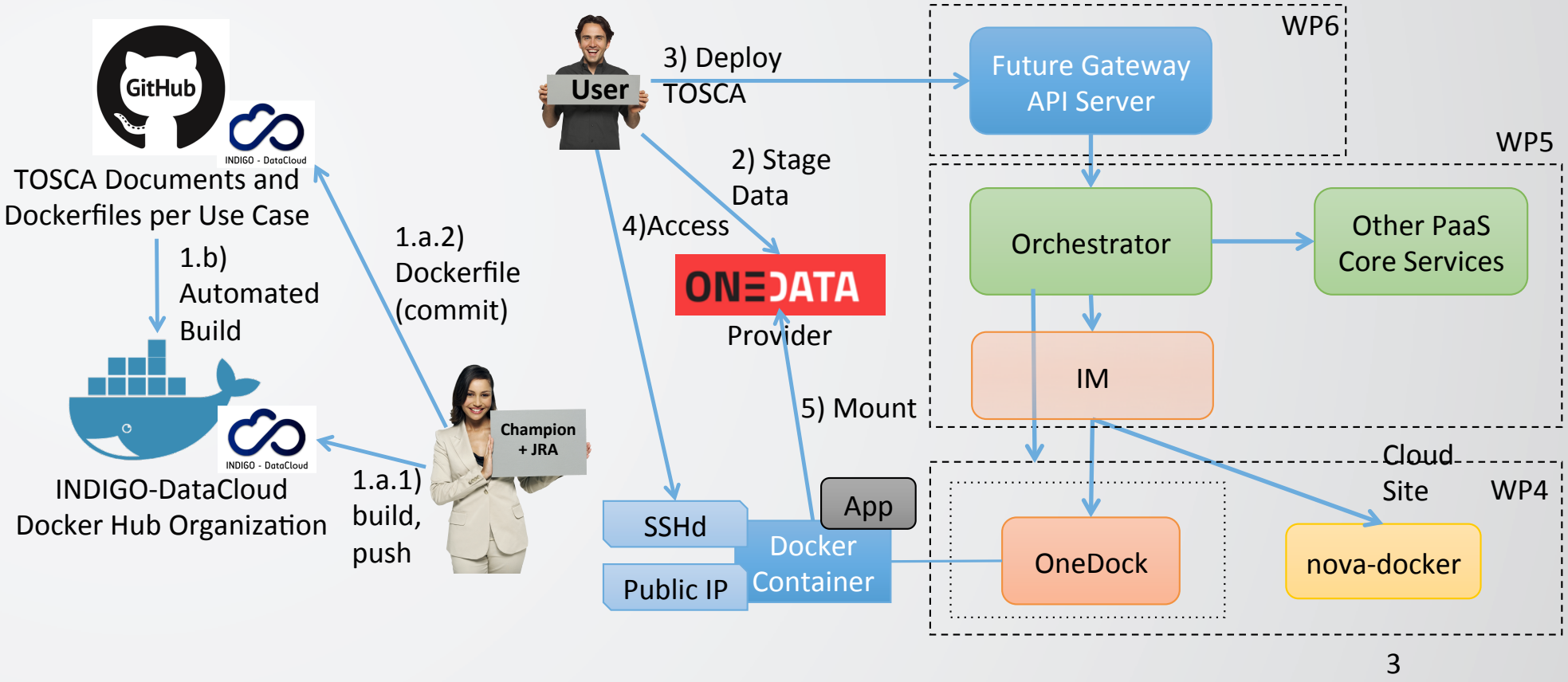
INDIGO-DataCloud
Use Case Examples

UC #1: Interactive usage of a Docker container with *ssh*

INDIGO - DataCloud

- A Docker container is instantiated automatically after a simple request on the web portal from an end-user.
  - This will exploit a **TOSCA Template** through the INDIGO orchestrator.
- The container has a public IP address and the user (or the portal) can directly get access to it.
- Users can mount a local **or remote** posix filesystem through INDIGO **Onedata**.
- The application in the Docker container is able to simply read the files provided via web browser by the end user and to write posix files that are available to users via web browsers.
- The same Docker container could be used to execute a large list of applications in a batch-like behaviour.

UC #1: Interactive usage of a Docker container with *ssh* - Overview



UC #2: A web portal that exploits a batch system to run applications

- A user community maintains a “vanilla” version of a portal using Galaxy, a computing image, plus some specific recipes to customize software tools and data
  - Portal and computing are part of the same image that can take different roles.
  - Customization may include creating special users, copying (and registering in the portal) reference data, installing (and again registering) processing tools.
  - Typically web portal image also has a batch queue server installed.
- All the running instances share a common directory.
- Different credentials: end-user and application deployment.

UC#2: Galaxy in the cloud

- Galaxy can be installed on a dedicated machine or as a front/end to a batch queue.
- Galaxy exposes a web interface and executes all the interactions (including data uploading) as jobs in a batch queue.
- It requires a shared directory among the working nodes and the front/end.
- It supports a separate storage area for different users, managing them through the portal.

UC #2: A web portal that exploits a batch system to run applications

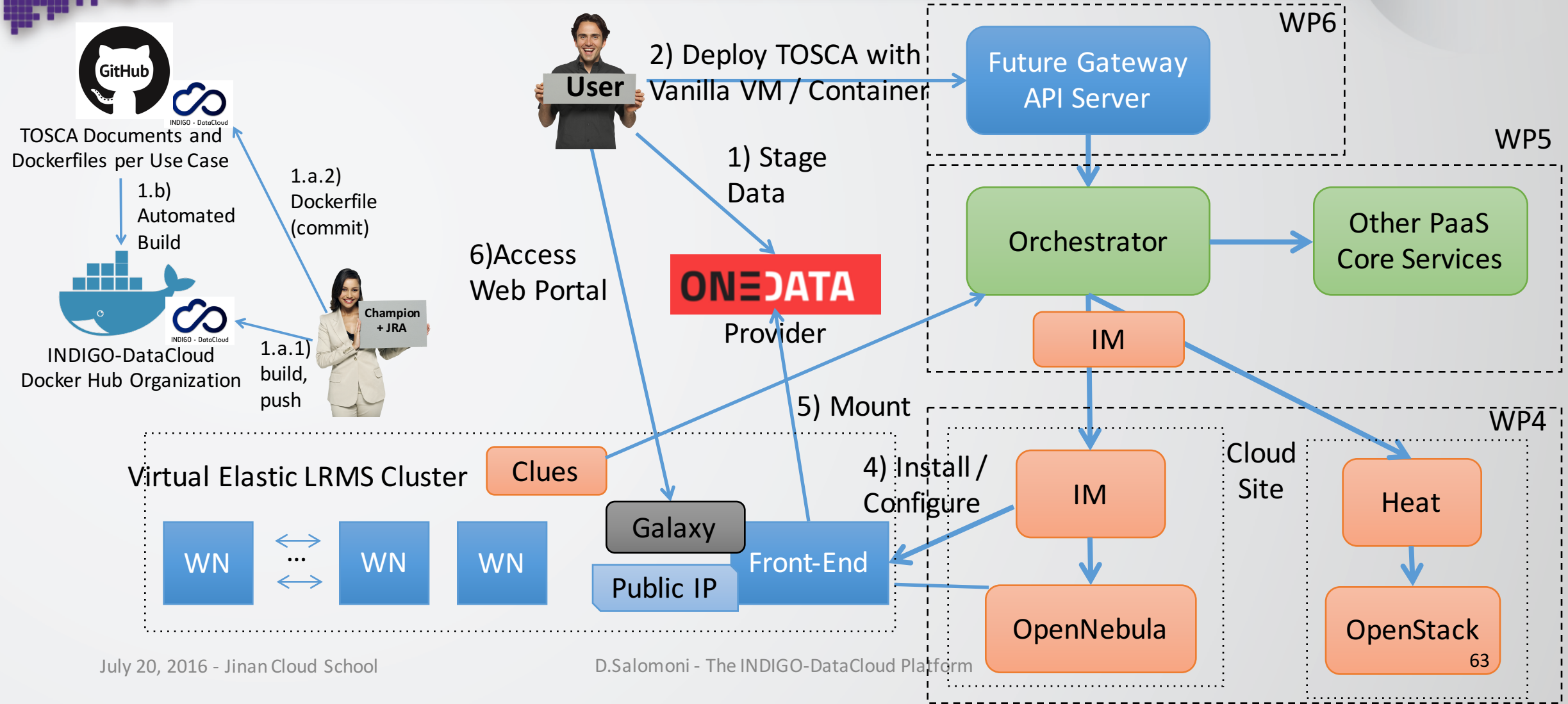
INDIGO - DataCloud

1) The web portal is instantiated, installed and configured automatically exploiting Ansible recipes and TOSCA Templates.
2) A remote posix share is automatically mounted on the web portal using Onedata
3) The same posix share is automatically mounted also on worker nodes using Onedata
4) End-users can see and access the same files via simple web browsers or similar.
5) A batch system is dynamically and automatically configured via TOSCA Templates
6) The portal is automatically configured in order to execute job on the batch cluster
7) The batch cluster is automatically scaled up & down looking at the job load on the batch system.

UC#2: Lifecycle

- Preliminary
  - The use case administrator creates the “vanilla” images of the portal+computing image.
  - The use case administrator, with the support of INDIGO experts, writes the TOSCA specification of the portal, queue, computing configuration.
- Group-specific
  - The use case administrator, with the support of INDIGO experts, writes specific modules for portal-specific configurations.
  - The use case administrator deploys the virtual appliance.
- Daily work
  - Users Access the portal as if it was locally deployed and submit Jobs to the system as they would have been provisioned statically.

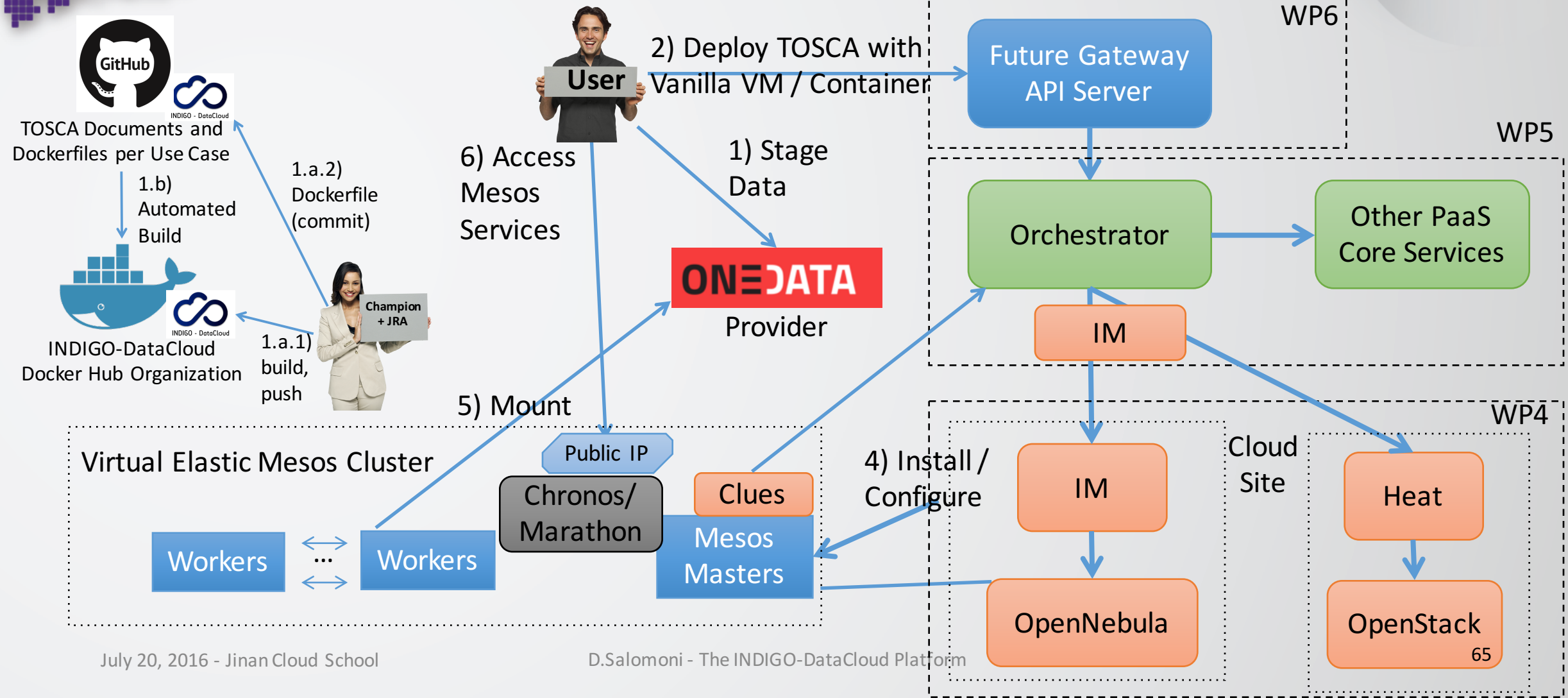
UC #2: A web portal that exploits a batch system to run applications - Overview



UC#3: A possible Phenomenal-INDIGO integration scenario

- Phenomenal is a European project already exploiting Apache Mesos for data processing and analysis pipelines applied to molecular phenotype data, generated by metabolomics applications.
- INDIGO is able to provide a customizable environment where a complex cluster could be deployed in an automatic way:
  - Using a specific TOSCA Template written with the help of INDIGO PaaS developers
- INDIGO can provide to Phenomenal:
  - (Automatic) Resource provisioning exploiting any kind of cloud environment (private or public)
    - Reacting on the monitoring the status of the services instantiated
  - Advanced and flexible AAI solutions
  - Advanced and flexible data management solutions
  - Advanced scheduling across cloud providers based on:
    - SLA/QoS, Data location, availability monitoring and ranked with highly flexible rules
  - An easy to use web interface for both end users and service admin/developers

UC#3: Phenomenal exploiting INDIGO



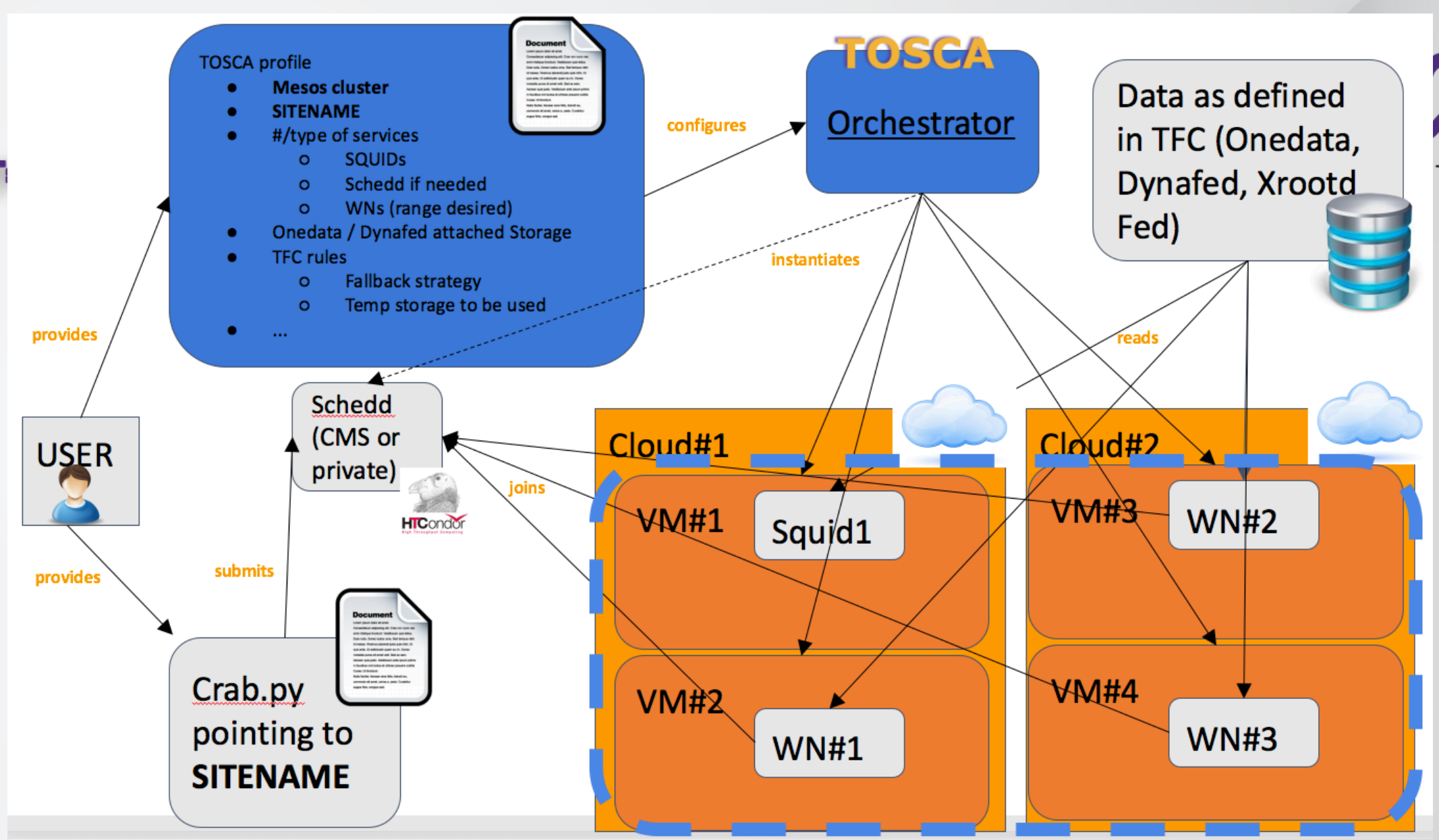
UC#4: enhancing CMS analysis workflows

- Our initial target is end user analysis@CMS, focusing in the area of the resource usage simplification, seen from multiple perspectives:
  - **Sites**:
    - Easy solution for dynamic/elastic T2 extensions on “opportunistic”/stable resources
    - Friendly procedure to instantiate a spot ‘Tier3 like’
  - **Users**:
    - Allow the generation of a ephemeral T3 on demand, or a personal T3 to serve a group of collaborators or even a single power user - all to be used via standard CRAB
  - **Collaboration**:
    - All this includes (see it as a by-product): an approach to the opportunistic computing
    - This might be extremely useful for small sites/campus, co-funded computing centers/ multidisciplinary centers, etc.

UC#4: how to achieve the goal

Automatically create a dynamic Mesos cluster to instantiate/manage CMS flavored WNs (and needed services)

- The plan is to use VM plus Docker in order to
  - Manage credential lifetime/translation/renewal through INDIGO Identity Access Management service
  - Mount posix remote file-system using Oneclient (from Onedata)
    - exploiting the INDIGO data solution for private / site storage
    - We plan to start with Dynafed; Xrootd is automatically supported via AAA
  - Execute condor_startd to connect with a CMS condor Pool (a-la HLT)
    - We'd like to exploit also the possibility to run a own condor_schedd (see next slide)
- The plan foresees to **scale resources dynamically through Marathon**
- **The generation of the Mesos cluster and the described setup will be automated**
  - Defining Tosca templates to be managed by INDIGO PaaS Orchestrator
  - Single YAML file describing the setup: Squid, Schedd, WNs in varying numbers, on-demand CMS Site name, ...


TOSCA profile
Mesos cluster
SITENAME
#/type of services
SQUIDs
Schedd if needed
WNs (range desired)
Onedata / Dynafed attached Storage
TFC rules
Fallback strategy
Temp storage to be used
...
configures
TOSCA
Orchestrator
Data as defined in TFC (Onedata, Dynafed, Xrootd Fed)
instantiates
reads
provides
USER
Schedd (CMS or private)
HTCondor
joins
Cloud#1
VM#1
Squid1
VM#2
WN#1
Cloud#2
VM#3
WN#2
VM#4
WN#3
provides
submits
Crab.py pointing to SITENAME

UC#5: running Docker containers without Docker ☺

INDIGO - DataCloud

- Adoption of Docker is being very slow in HPC centers
- Thus the typical situation is that Docker is not installed and one cannot run containers without some support from the system software.
- In general, Docker adoption will be slow in any computing farm or interactive linux system shared by many users.
  - It will take time for sysadmins to overcome the concerns of their security teams.
  - It is yet another service to maintain…
  - …. you name it.

UC#5: INDIGO udocker

- **A tool to execute content of docker containers in user space** when docker is not available
  - enables download of docker containers from dockerhub
  - enables execution of docker containers by non-privileged users
- **It can be used to execute the content of docker containers in Linux batch systems and interactive clusters managed by others**
- **A wrapper around other tools to mimic docker capabilities**
  - current version uses **proot** to provide a chroot like environment without privileges (it runs on CentOS 6, CentOS 7, Fedora, Ubuntu)
- **More info and downloads at:**
  - https://www.gitbook.com/book/indigo-dc/udocker/details
  - https://indigo-dc.gitbooks.io/udocker/content/doc/user_manual.html

UC#5: INDIGO udocker

- Examples:

```
# download, but could also import or load a container exported/save by docker
$ udocker.py  pull   ubuntu:latest
$ udocker.py  create  --name=myubuntu   ubuntu:latest

# make the host homedir visible inside the container and execute something
$ udocker.py  run   -v $HOME    myubuntu   /bin/bash   <<EOF
cat   /etc/lsb-release
ls  -l  $HOME
EOF
```

udocker is NOT an alternative to docker: we need the container image built by docker.

It is a tool to handle and run containers with regular user privileges and/or when docker is not available for some reason: **it is very convenient to access clusters and Grid resources**

UC#5: INDIGO udocker

- Everything is stored in the user home dir or some other location
- Container layers are download to the user home
- Directory trees can be created/extracted from these container layers
- proot uses the debugger ptrace mechanism to change pathnames and execute transparently inside a directory tree
- No impact on read/write or execution, only impact on system calls using pathnames (ex. open, chdir, etc)

- Does not require installation of software in the host system:
  - udocker is a python script
  - proot is statically compiled

Conclusions

- The **first public INDIGO release** will be out at the beginning of August 2016.
- The first prototype is already available for internal evaluation.
- Several concrete use cases are currently being implemented by the INDIGO scientific communities.
- A lot of important developments are being carried on with the original developers community so that code maintenance is not (only) in our hands .

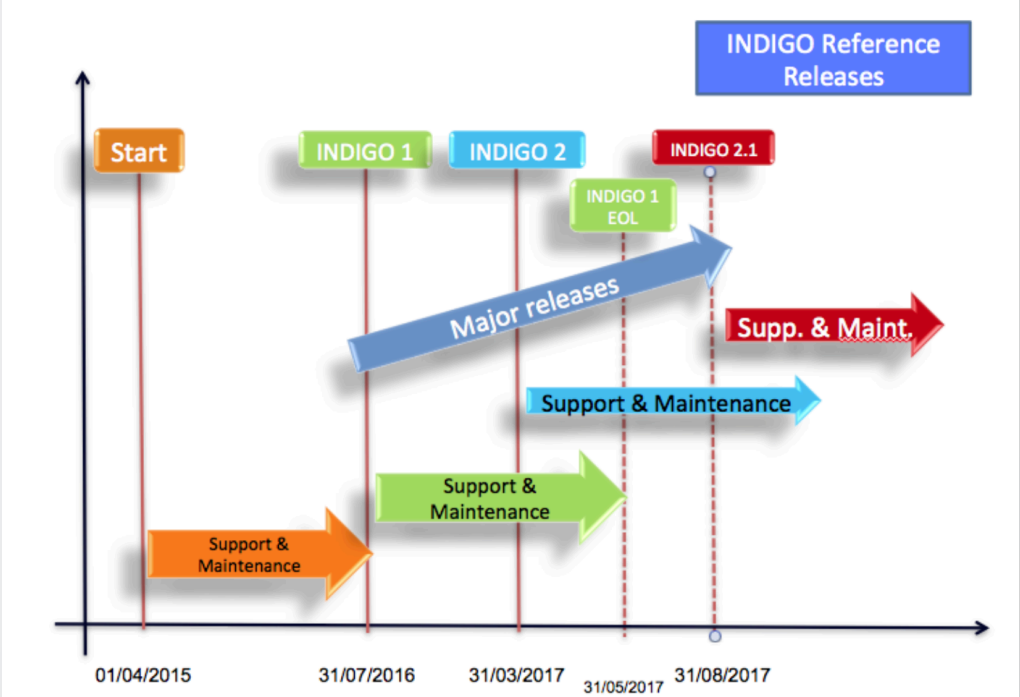


Figure 8: INDIGO-DataCloud Release Timeline

Thank you

INDIGO - DataCloud

https://www.indigo-datacloud.eu

Better Software for Better Science.