Community Requirements

Dr. Isabel Campos,
CSIC – Spanish National Research Council,
isabel.campos@csic.es

as INDIGO-Datacloud Deputy Coordinator

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

Content of the presentation

INDIGO - DataCloud

- **INDIGO-Datacloud approach & implementation**
- **Designing a Platform as a Service for research**
- **Infrastructure to support Advanced Services**
- **Requirements from Research Communities:**
  - **Requirements Gathering Methodology**
- **Initial feedback and proposed solutions**

An H2020 project approved in January 2015 in the EINFRA-1-2014, 11.1M€, 30 months (until September 2017)
Who: **26 European partners** in 11 European countries
What: develop an open source Cloud platform for computing and data ("DataCloud").
For: multi-disciplinary scientific communities
Where: deployable on hybrid (public or private) Cloud infrastructures

INDIGO-Datacloud approach

Imagine that Computing & Storage resources are presented as a big pool, in which, the user never really knows which machine is being used

- **This approach is viable for many applications** (not for all)
  - Eg. when knowing the architecture of the processor is vital to achieve a very high target performance.

- **At the Software level it is already a reality**
  - Virtualization or Containers technology make the underlying OS irrelevant for the user.

  - If you do not mind loosing performance "too much", a virtual machine runs everywhere with the software you need

  - If you are more nervous about performance, you have to work in the direction of containers technology: encapsulate your application in a container (eg. using docker, or liblxc from "first principles".

INDIGO-Datacloud approach

What would researchers need in order to work in such an environment?

Login interface: from a simple shell, to a sophisticated portal

- Highly user dependent

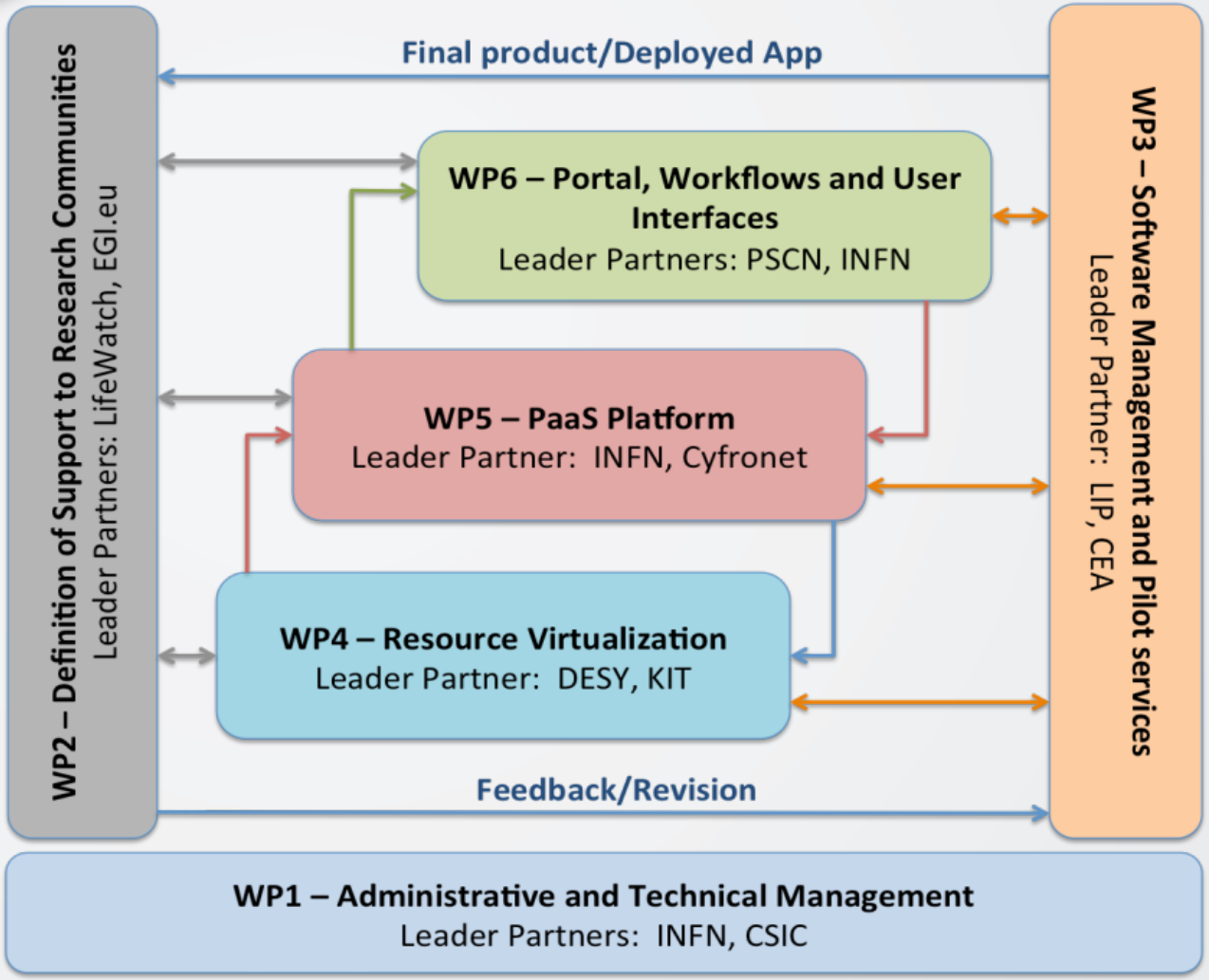
The login interface would need to **expose advanced capabilities to the user**: **Platform as a Service (PaaS)**

- We would like the user to be able to deploy a small cluster on demand
- Interact with Containers and Virtual Machines repositories
- Interact with data repositories in a user-friendly way

The **underlying computing, storage and network infrastructure, would need to support the interaction** of the users via such tools

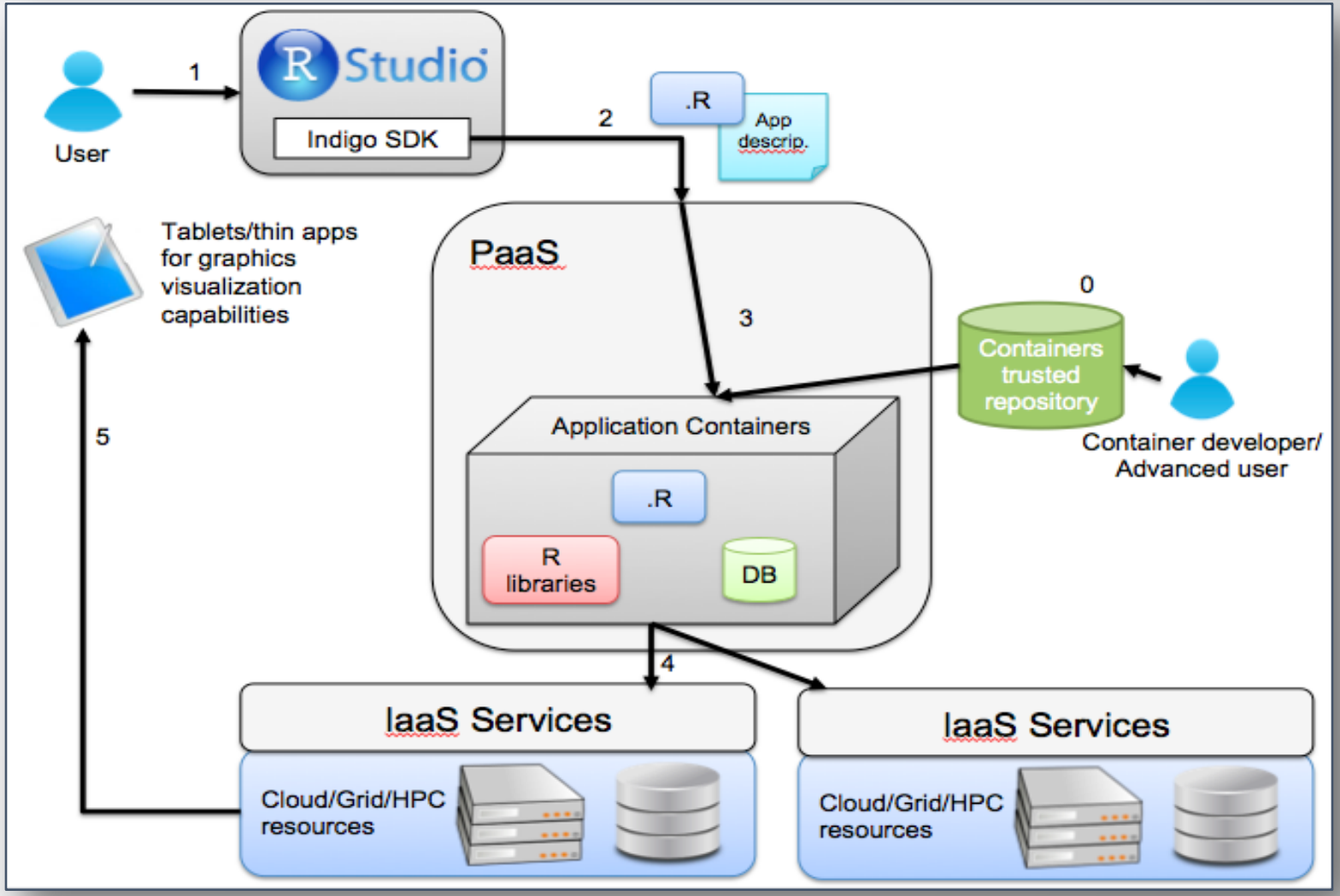
- The solutions cannot be “invasive” at the level of Infrastructure administration (or else will not be adopted)

INDIGO-Datacloud: project implementation



- ✓ **Community Requirements (WP2)**
- ✓ **Portal deployment → user access (WP6)**
- ✓ **Platform as a Service design (WP5)**
- ✓ **Infrastructure Oriented (WP4)**
- ✓ **Software management and pilot services (WP3)**

INDIGO-Datacloud approach



Building a Platform as a Service layer for researchers

INDIGO - DataCloud

- **The PaaS layer is composed of four main areas:**
  - the PaaS core services,
  - a unified federated virtual storage,
  - a federated AAI service and
  - a service for geographically deploying both user applications and services.

- **The INDIGO PaaS core will be based on a microservice architecture**

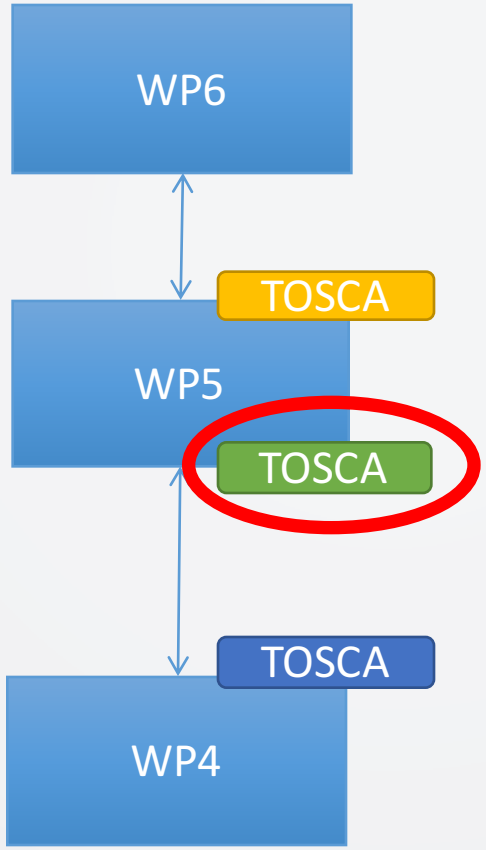
http://microservices.io

  - Microservices consist of a set of narrowly focused, independently deployable services, typically implemented using container-embedded applications, exposed by RESTful interfaces.
  - Microservices are designed to be highly scalable, highly available and targeted for the use in cloud environments.

- The INDIGO PaaS will offer an **upper layer orchestration service for distributed applications using the TOSCA** language standard

  - **TOSCA = Topology and Orchestration Specification for Cloud Applications**

INDIGO – PaaS : use of Standards



- OASIS TOSCA (Topology and Orchestration Specification for Cloud Applications) 1.0 (11/2013)
  - Interoperable description of application and infrastructure cloud services, the relationships between parts of the service, and the operational behavior of these services (e.g., deploy, patch, shutdown) independent of the supplier creating the service, and any particular cloud provider or hosting technology.

Source: http://www.slideshare.net/OpenTOSCA/tosca-and-opentosca-tosca-introduction-and-opentosca-ecosystem-overview?ref=http://www.slideshare.net/OpenTOSCA/slideshelf

INDIGO - Platform as a Service

- We will be using the **TOSCA language also as a "glue" between the infrastructure and the user**
  - Eg. for the description of application requirements
- The **PaaS core services will be deployed as Docker containers too** using the open source solution "Kubernetes"
- Cloud infrastructures
  - We will be supporting interaction via OCCI (to keep standards) and also,
  - OpenStack interacts with TOSCA via its own orchestrator (Heat)
  - OpenNebula interacts with TOSCA via the IM (developed by one of our partner, the UPValencia).

INDIGO - The Infrastructure side

INDIGO - DataCloud

Optimizing the access and usage of virtualized resources on the Infrastructure provider level, towards the PaaS implemented in the project

- **Using the rigth interfaces**
  - Standards and its extensions: OCCI, TOSCA, CDMI, WebDAV
- **Responding to the gap analysis performed, some of them being:**
  - Unified handling of containers in Cloud computing infrastructures
  - Improvement of job schedulers providing fair share, and spot-instance mechanism
  - Mechanisms to steer quality of services data life cycle policies
  - Evaluation of Software Defined Networks together with Virtually managed services

Implementing Containers: Docker

- Docker is an Open Source Linux container engine.
  - first release: 3/2013
  - https://www.docker.io/
  - git repository at: https://github.com/dotcloud/docker.git

- **Docker is optimized for the deployment of applications**, as opposed to machines (system administration in general).

  - the liblxc helper scripts focus on containers as lightweight machines - basically servers that boot faster and need less RAM.
  - There is more to containers than just that.
  - It the near future it will change radically the way software is deployed

What are containers good for?

INDIGO - DataCloud

- Imagine a situation in which applications run on a sort of “cloud” where,
  - The user never actually knows anything about the machine that is being used.
  - A total encapsulation of the application would be necessary
    - No assumptions can be made regarding the OS, or the hardware

- A **system software layer** able to deploy such encapsulated applications would be extremely interesting
  - Actually is the key to the success of container technology in the scientific areas
  - They should be integrated in batch systems: docker cannot do that.
  - In INDIGO we are building such layer of system software.
  - INDIGO (both at PaaS and IaaS level) will leverage the official DockerHub repository in order to be as standard and open as possible.

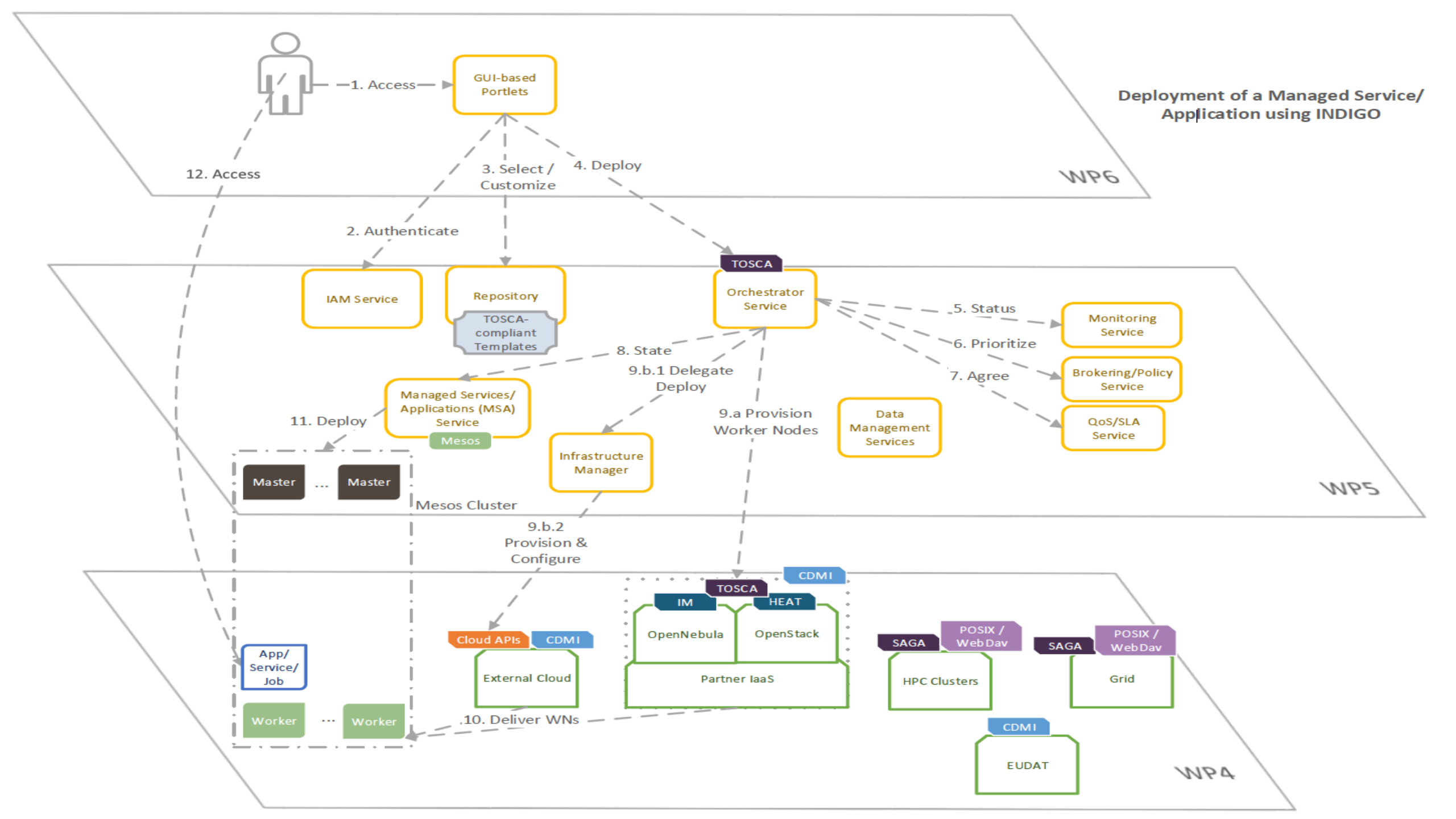
INDIGO: containers support

How to provide users with support for execution of "containerized" applications ?

- **Develop / extend container support**
  - OpenStack (extending functionalities of nova-docker)
  - OpenNebula (develop support for docker)
- **Integration of trusted repositories** for containers
- **Extend relevant standard interfaces (OCCI).**
- Integration of container execution in
  - **Batch systems**
  - **Specialized hardware (Infiniband and GPGPUs)**

Global Architecture

Quick look


Deployment of a Managed Service/ Application using INDIGO
1. Access
GUI-based Portlets
12. Access
3. Select / Customize
4. Deploy
WP6
2. Authenticate
IAM Service
Repository
TOSCA-compliant Templates
TOSCA
Orchestrator Service
5. Status
6. Prioritize
7. Agree
Monitoring Service
Brokering/Policy Service
QoS/SLA Service
8. State
9.b.1 Delegate Deploy
Managed Services/ Applications (MSA) Service
Mesos
11. Deploy
Infrastructure Manager
9.a Provision Worker Nodes
Data Management Services
Master
...
Master
Mesos Cluster
WP5
9.b.2 Provision & Configure
CDMI
TOSCA
IM
HEAT
OpenNebula
OpenStack
Partner IaaS
Cloud APIs
CDMI
External Cloud
SAGA
POSIX / WebDav
HPC Clusters
SAGA
POSIX / WebDav
Grid
CDMI
EUDAT
App/ Service/ Job
Worker
...
Worker
10. Deliver WNs
WP4

INDIGO Research Communities

- A work package (WP2, Networking Activity) represents the interest of Research Communities to ensure that their requirements will be satisfied by the project products

- **Keep the focus also on big data research use and management through a dedicated task** oriented to track the different needs at the data life-cycle, following the reference models used by the different research communities

INDIGO - Community requirements

Research Community	Case Study/Application	
LifeWatch	**Monitoring and Modelling Algae Bloom in a Water Reservoir**	
	TRUFA (Transcriptomes User-Friendly Analysis)	
EuroBioImaging	**Medical Imaging Biobanks**	
INSTRUCT	**Molecular dynamics simulations**	
LBT CTA	**Astronomical Data Archives** **Archive System for the Cherenkov Telescope Array**	
WeNMR	**HADDOCK portal**	
ENES	**Climate models inter comparison data analysis**	
Galleries, Libraries, Archives, Museums	**eCulture science Gateway**	
ELIXIR	**Galaxy as a Cloud service**	
EMSO	**MOIST-multidisciplinary oceanic information system**	
DARIAH	**Big Data in Arts and Humanities**	
EGI Virtual Teams Competence Centres	**Chipster**	**BILS**
	READemption	**Human Brain Project**
	JAMS	**BBMRI-ERIC CC**
	HAPPI	**DARIAH CC**
	INERTIA	**EPOS CC**
	DRIHM	**Disaster Mitigation**
	CANFAR	**LoFAR**

Requirements gathering

INDIGO - DataCloud

- Analyze the use cases proposed by the communities participating to the consortium. Capture the requirements for efficiently running the applications and workflows on Cloud, Grid or HPC infratructures

Caution! Impendance mismatch, ICT experts vs. Researchers

- Capture requirements generated by user communities not part of the project (such as the EGI Federated Cloud users)
- Liaise with the INFRADEV-4 projects to enable synergies between the projects, and interoperability between the INDIGO output and the VRE to be deployed by the e-INFRA-9 projects.

INDIGO initial gap analysis: selected topics

INDIGO - DataCloud

- Support **federated identities** and provide privacy and distributed authorization in open Cloud platforms
- The barriers that limit the adoption of true PaaS solutions, such as the use of custom, non-interoperable interfaces and the limited availability of APIs for **technology-independent storage access**.
- The lack of **flexible data sharing** between groups members and the difficulty in obtaining easy access to data generated by collaborating users **working with different infrastructures or sites**
- Static allocation and partitioning of **both storage and computing resources** in data centers

INDIGO – Methodology to gather requirements - I

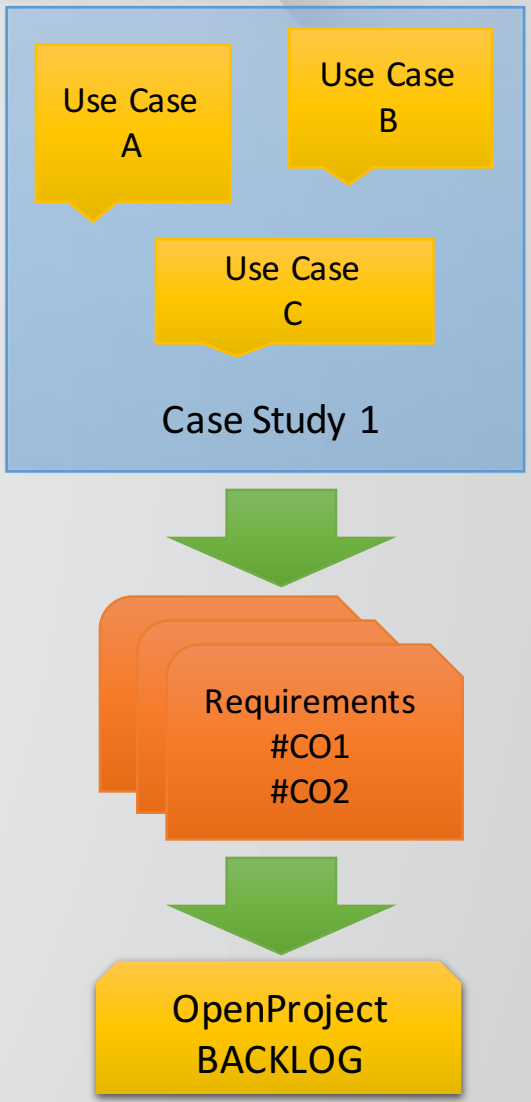
Highly criticized

and refined

in several iterations

A template was designed to gather information from communities

- Based on case studies: implementation of the research method involving a detailed examination of the subject and the contextual conditions
- Focus on case studies that are representative both of the research challenge and complexity but also of the possibilities offered by INDIGO solution to it
- A Case Study is ideally based on a set of Use Cases
- Use cases are useful to capture the Requirements that will be handled by the INDIGO software developed in JRA workpackages, and tracked by the backlog system from our OpenProject tool.
- The template serves as a structured framework with guiding questions concerned by INDIGO development workpackages



INDIGO – Methodology to gather requirements - II

- **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 https://www.indigo-datacloud.eu/pages/components/deliverables.html

#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

Common Requirements Analysis

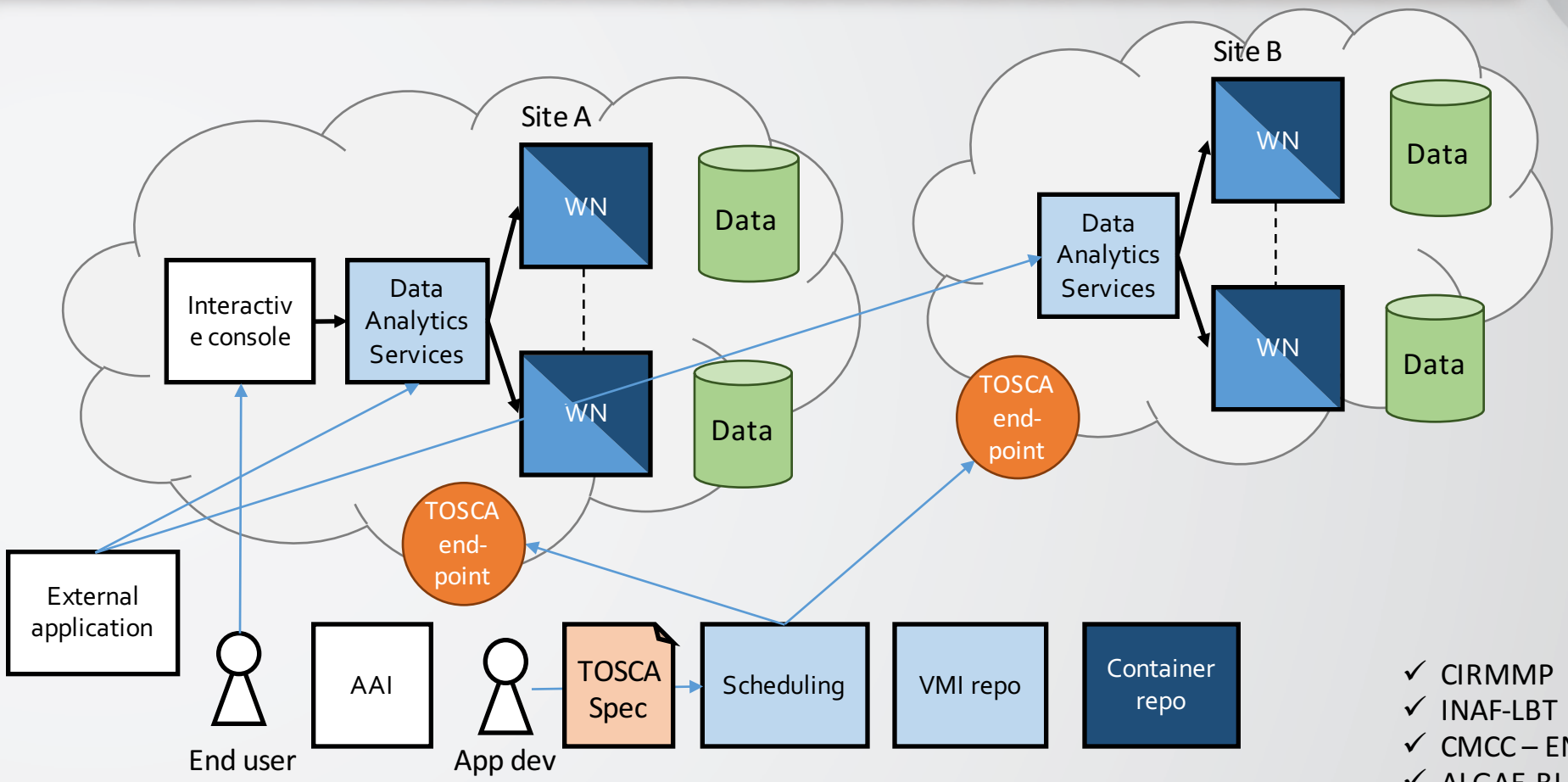
#	Requirement	Requirement	Rank	Proposed improvement	Potential solution for INDIGO (User	Pot	EuB	Lif	ELI	Had	CIR	Fe	DA	INA	CMC	C
	Deployment of Interface SaaS	Computing / PaaS	Mandatory	A mechanism to facilitate the deployment of a customised Haddock	The portal could be instantiated by means of a set of containers and/or specific base		M	C	M	M	M			C	C	N
	Deployment of Customized computing back-ends as batch queues	Computing / PaaS	Mandatory	Each instance may have an independent software configuration,	A devops tool integrated with the deployment service to install and configure		M	M	M	M	M	C		C	M	C
	Deployment of user-specific software	Computing / PaaS	Mandatory	Manual installation may be cumbersome for large-scale application involving many computing	Ability of a user to easily construct a software installation and configuration specification (e.g. TOSCA) for their own		M		M						C	
	Automatic elasticity of computing batch queues	Computing / PaaS	Mandatory	When moving to the cloud, users should be provided with the exact	Monitoring services may be integrated with the deployment, which will trigger the		M	M	M	M	M			C	M	
	Terminal access to the resources.	Computing / PaaS service	Mandatory	This feature must be linked to the AAI	This will require ssh ports to be open and direct access to the VMs. The massive		M		M	M	M				M	
	Privileged access	Computing / PaaS service	Mandatory	This feature must be linked to the AAI	A single special user in the "sudo" group.		C		M	M	M				M	
	Execution of workflows	Computing / PaaS	Mandatory	Processing done on the cloud where the outputs of the processing are	Workflow engine can be deployed as any other application. Back-end could be a		M		C	C		O			M	N
	Provenance information	Computing / PaaS Service	Convenient	Very important for revision of papers and project proposals.	Repository of data and software that could be deployed or inspected on demand.		C									
	Cloud bursting	Computing / PaaS Service	Mandatory	Supplementing the computing capacity with special instances	Automatic contextualization and configuration will enhance the		C	C	C	M					M	
	Data-aware scheduling	Computing / PaaS Service	Convenient	Currently storage and computing are highly coupled.	This will affect the scheduling. Moving computing to data. Maybe the use of				C			C			M	
	Provisioning of efficient Big Data Analysis solutions exploiting server-side and declarative approaches	Computing / Storage / PaaS Service	Mandatory		Currently it uses a hierarchical set of databases that are coordinated through distributed memory parallel computing										M	
	Execution across multiple centres.	Computing / PaaS Service	Mandatory	Interesting when exhausting resource capabilities of one deployment or when	Task T5.3 in INDIGO deals with the geographic scheduling of workloads,										M	
	On-line processing of data	PaaS	Mandatory	Special management of post-processing jobs that could be sent to	Despite that this may look similar to any other processing, two aspects need to be		C	M			M			M	M	
	Special hw configuration - MPI, multicore, GPGPU	Compute / PaaS	Mandatory	More flexibility in the way the requirements are defined and the	Three main issues must be analysed here (not all for the User Cases selected)c: 1) The		C	C			M				M	

User Community Computing Portal Service (~SaaS)

INDIGO - DataCloud

- A user community **has an application** (or set of them) that can be accessed through a portal and **requires a batch queue as back-e**nd.
- **Unpredictable workload** and user access profile.
- The application consists on two main parts: the portal / scientific Gateway and the processing working nodes
  - Working nodes should scale-up and down according to the workload (automatically done by the infrastructure).
  - Cloud-bursting to external infrastructures may be requested.
  - Portal services should also adapt to workload.
  - Users can access reference data and provide their own local data.
- **Requested by** the use cases from:
  - ELIXIR, Haddock, CIRMMP, FedCloud, DARIAH, INAF-LBT, CMCC – ENES, CTA, ALGAE – BLOSSOM, INGV - MOIST

Software as a Service “portal”



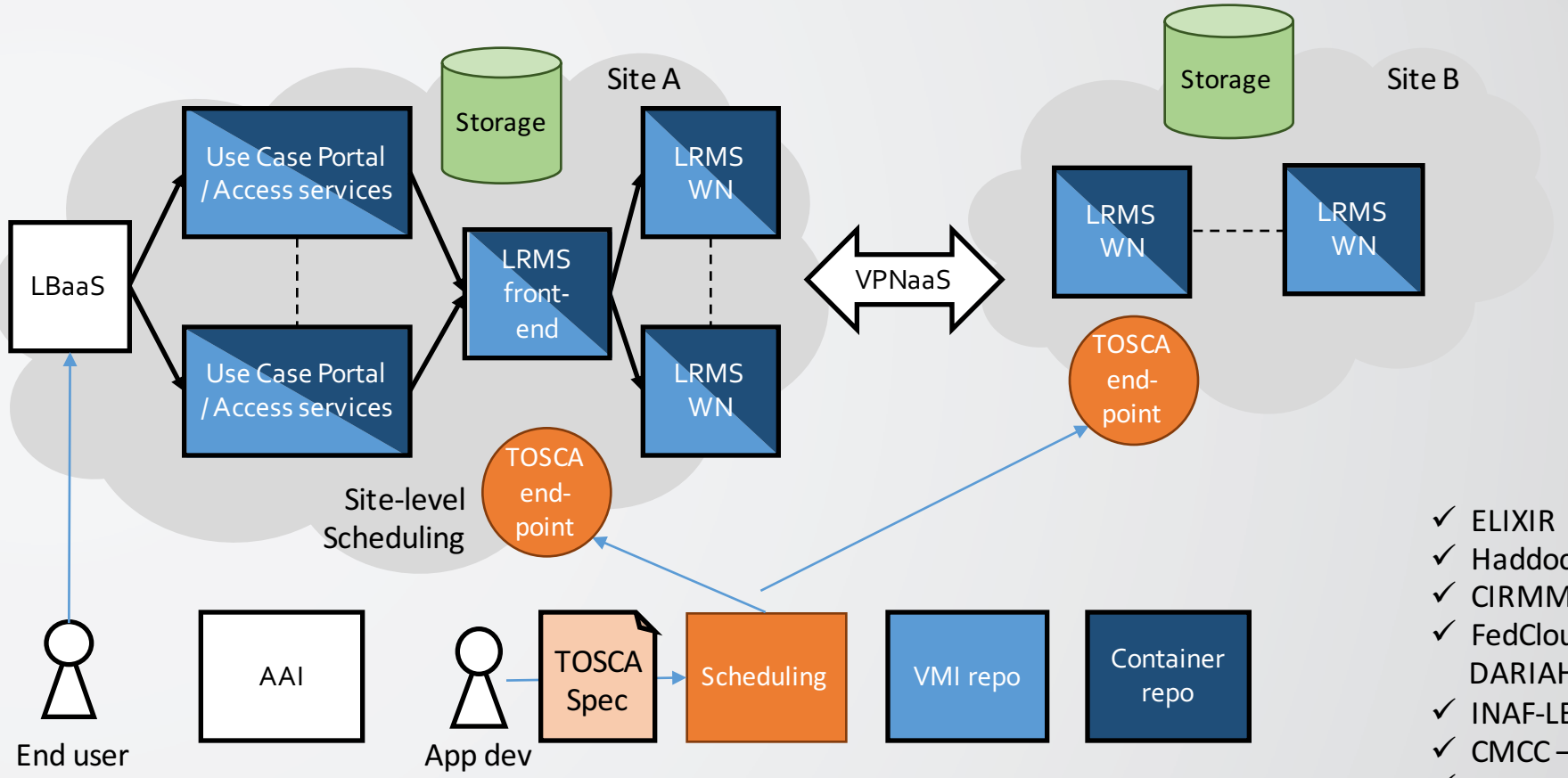
- ✓ CIRMMP
- ✓ INAF-LBT
- ✓ CMCC – ENES
- ✓ ALGAE-BLOSSOM
- ✓ INGV - MOIST

Data Analysis Service (~PaaS)

INDIGO - DataCloud

- A user community **has a coordinated set of data repositories and software services to access, process and inspect them**
  - Processing is **interactive**, requiring accessing a console deployed on data's premises.
- The application consists on **a console** / Scientific Gateway **that interacts with the data**
  - E.g "R", Python, OPHIDIA
  - It can be a complementary scenario from the previous one.
  - It can expose programmatic services.
- **Requested by** use cases from:
  - CIRMMP, INAF-LBT, CMCC – ENES, ALGAE – BLOSSOM, INGV – MOIST.

… Scientific Computational Portal “as a Service”



- ✓ ELIXIR
- ✓ Haddock
- ✓ CIRMMP
- ✓ FedCloud
  DARIAH
- ✓ INAF-LBT
- ✓ CMCC – ENES
- ✓ CTA
- ✓ ALGAE-BLOSSOM
  INGV - MOIST

Work ahead

- (The very hard task of) Collecting and **consolidating** evolving user requests
- Creation of a new **sustainable cloud competence in Europe for PaaS**, for both the scientific and industrial sectors, similar to what OpenStack and OpenNebula have done for IaaS
- **Many** technology gaps
  - For example: storage QoS, PaaS standardization, distributed AuthZ, static allocation of hardware resources, data sharing, customizable application portals

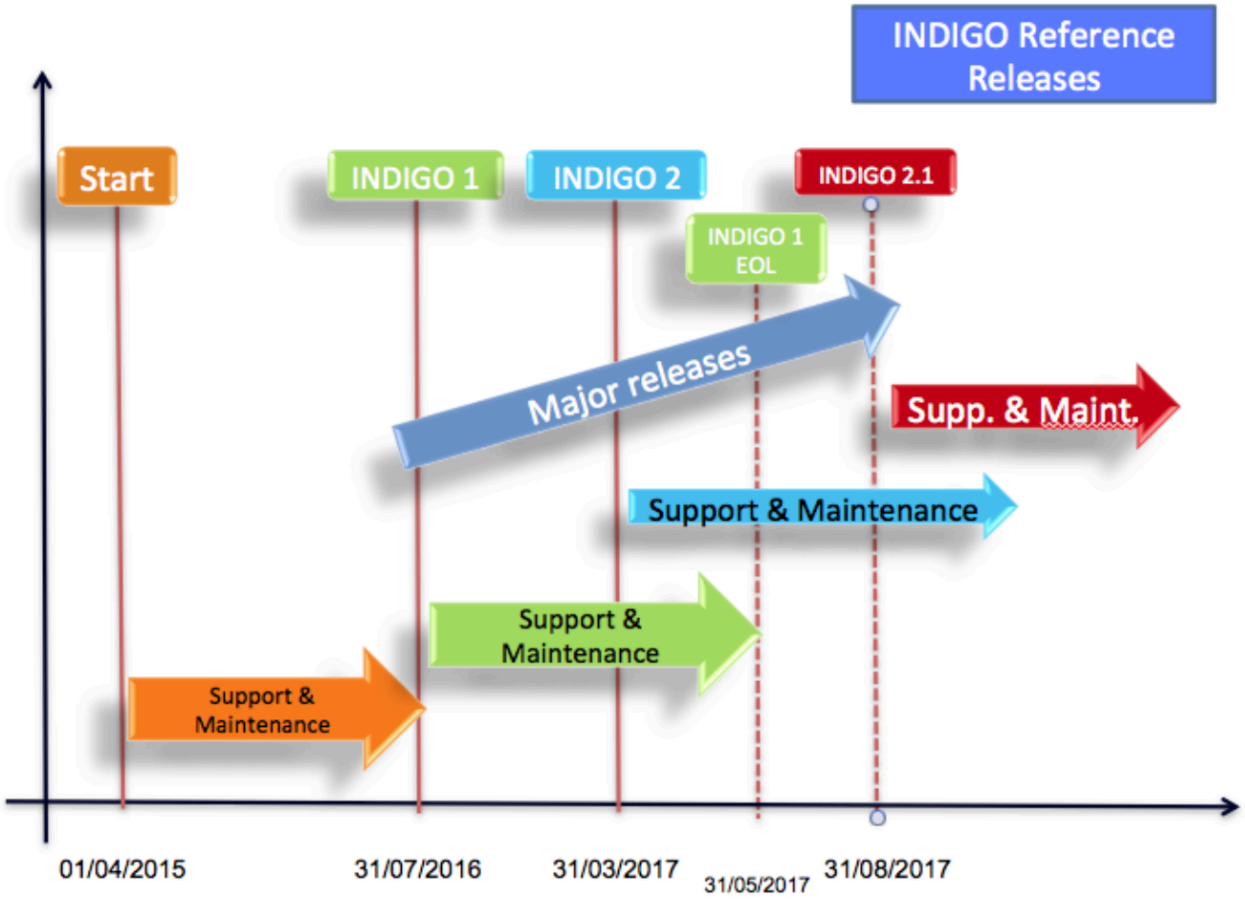


Figure 8: INDIGO-DataCloud Release Timeline

INDIGO - DataCloud

Thanks, Danke, Gracias!

Stay tunned on
www.indigo-datacloud.eu

Backup on Data

Unified Data Access

- Combining feature sets of products such as Onedata, FTS and DynaFed
- Data set registry - provides unified vision of geographically distributed data set
- API for data and metadata management including: registration, migration, replication, sharing
- Unified optimized data access for federated data based on APIs or POSIX
- Optimization of remote data access by remote block transfer on the fly and local caching
- Access Control Lists management for federated data
- Gateway to external data repositories
- Open data gateway

PaaS-level Unified Storage Interfaces

- Data access methods and protocols:
  - CDMI, Web GUI, WebDAV, S3
  - POSIX as a mounted virtual volume
- Data locations:
  (some standardization effort required here)
  - Onedata locations via CDMI
  - DynaFed location API or WebDAV?
- Data migrations and replications:
  (some standardization effort required here)
  - FTS data migration REST API
  - Onedata data migration REST API
  - Onedata CDMI-extended for replications based on metadata attributes

Backup on Containers

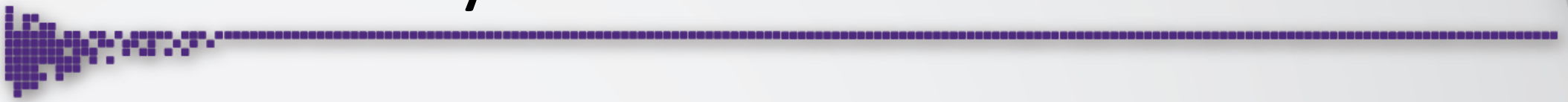
Containers - definition

INDIGO - DataCloud

A **“Linux Container”** is a technology provided by the Linux kernel, to “contain” a group of processes in an **independent execution environment**: this is called a **“container”**.

- They are internally realized using ***namespaces*** and ***cgroups***
  - Filesystem separation → Mount namespace (kernel 2.4.19)
  - Hostname separation → UTS namespace (kernel 2.6.19)
  - IPC separation → IPC namespece (kernel 2.6.19)
  - User (UID/GID) separation → User namespace (kernel 3.8)
  - Process table separation → PID namespace (kernel 2.6.24)
  - Network separation → Network Namespace (kernel 2.6.24)
  - Usage limit of CPU/Memory → Control Groups

Containers
The library API: liblxc



- The “low-level” foundations of the kernel are “wraped” in the ***liblxc*** library
  - It **encapsulates the capability** of using namespaces and cgroups in a user-friendly manner
  - One could work directly with the namespaces and cgroups tools described before
- Advanced tools such as **Docker** or **OpenVZ** use (to some extent) liblxc as well:
  - **Docker is an “applications oriented” implementation**
  - **OpenVZ is an “sysadmin oriented” implementation**

Containers versus Virtual Machines

We have available two types of virtualization:

- Virtual Machines with Hypervisors: (Xen, KVM) i.e. another OS instance by Virtualization
- Containers: Lightweight Process level handled by the linux kernel

Will Hypervisors, disappear from the infrastructures? probably not:

- Using VMs, one can run different kernels on different guests; with containers you must use the same “host kernel”.
- VM support non-Linux OS such as MS Windows, etc. You cannot run Windows in a container.

Containers advantages

- **Start-up and Shut-down time**: With process-level virtualization, starting and shutting down a container is faster than with VMs like Xen/KVM.
  - starting 4 Fedora containers in less then half a second on a laptop.
- **Density**: on a given server, you can deploy more Containers than VMs
  - Simply because containers do not exhaust resources as much
  - Higher revenues for cloud providers