INDIGO - DataCloud

RIA-653549

How the INDIGO-DataCloud computing platform aims at helping scientific communities

Giacinto DONVITO
INDIGO Technical Director
INFN Bari
giacinto.donvito@ba.infn.it

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

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.

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.**

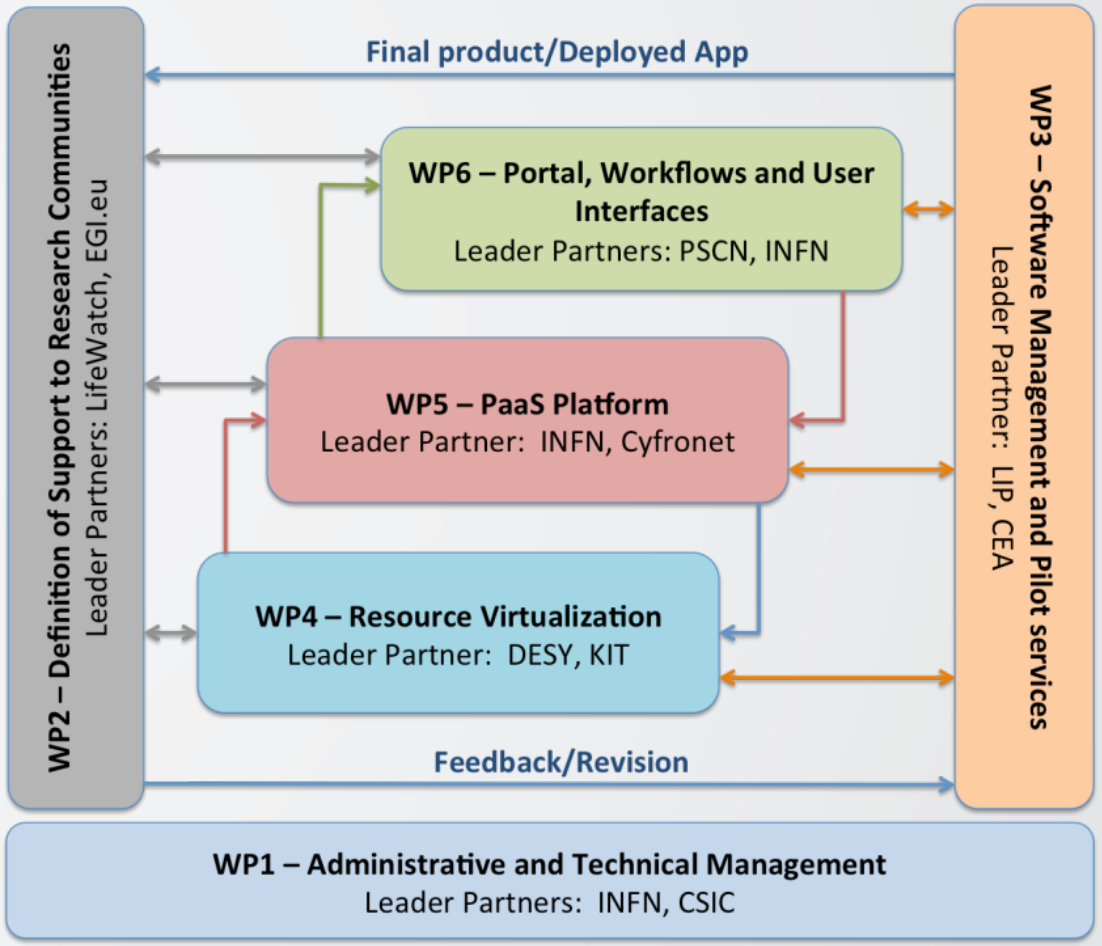
INDIGO Addresses Cloud Gaps

- **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.

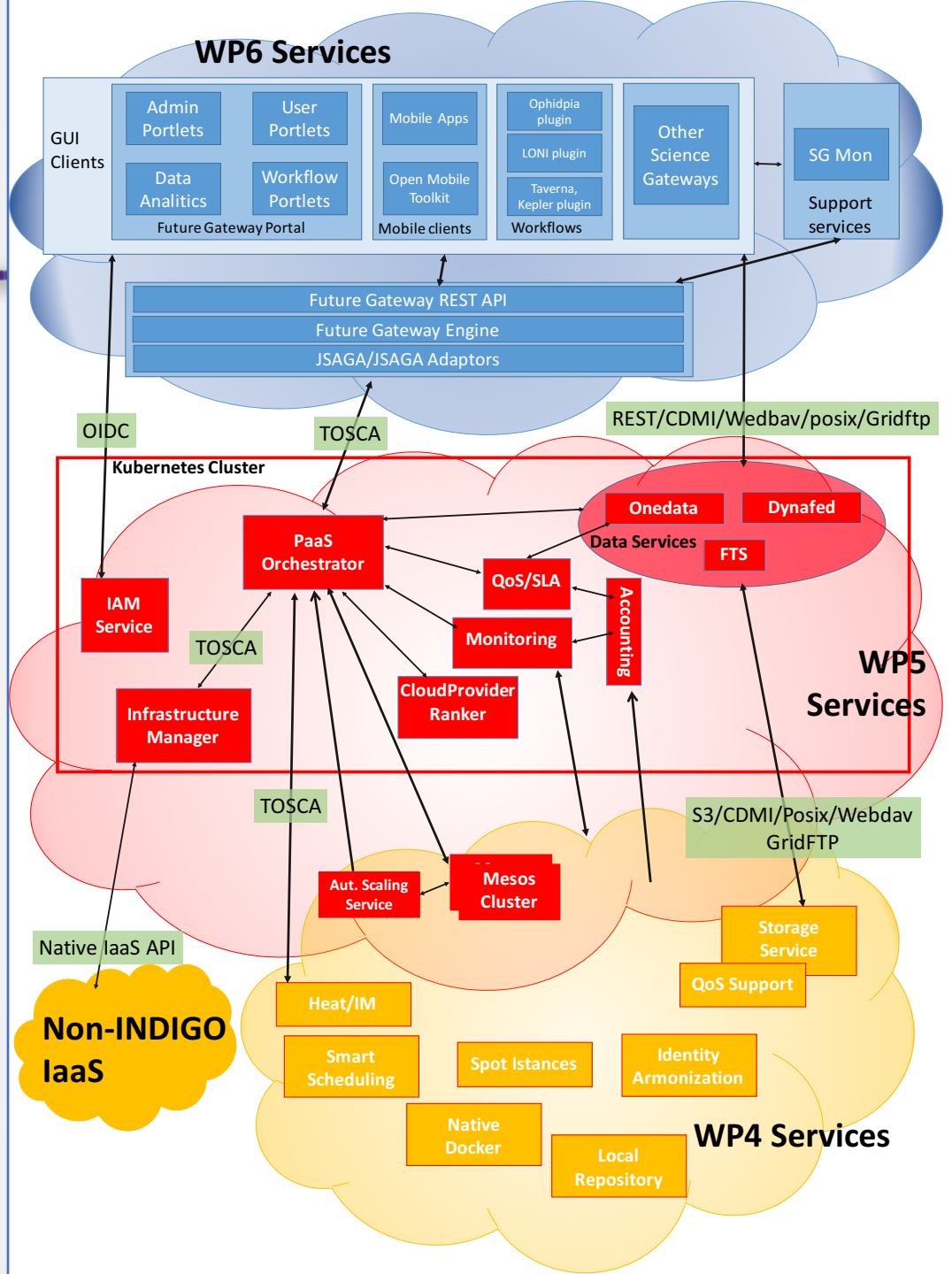
INDIGO and other European Projects

- The INDIGO services are being developed according to the requirements collected within many multidisciplinary scientific communities, such as **ELIXIR, WeNMR, INSTRUCT, EGI-FedCloud, DARIAH, INAF-LBT, CMCC-ENES, INAF-CTA, LifeWatch-Algae-Bloom, EMSO-MOIST, EuroBioImaging**. However, they are implemented so that they can be easily reused by other user communities.
- INDIGO has strong relationships with complementary initiatives, such as **EGI-Engage** on the operational side and **AARC** with respect to AuthN/AuthZ policies. Users of EC-funded initiatives such as **PRACE** and **EUDAT** are also expected to benefit from the deployment of INDIGO components in such infrastructures.
- Several **National/Regional infrastructures** are covered by the 26 INDIGO partners, located in 11 European countries.
- INDIGO is mentioned in the recent **Important Project of Common European Interest (IPCEI)** for the exploitation of HPC and HTC resources at national, regional and European levels.

Work Packages



INDIGO-DataCloud General Architecture



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)

- **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 to know where resources are located, because 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 will use 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)

- **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

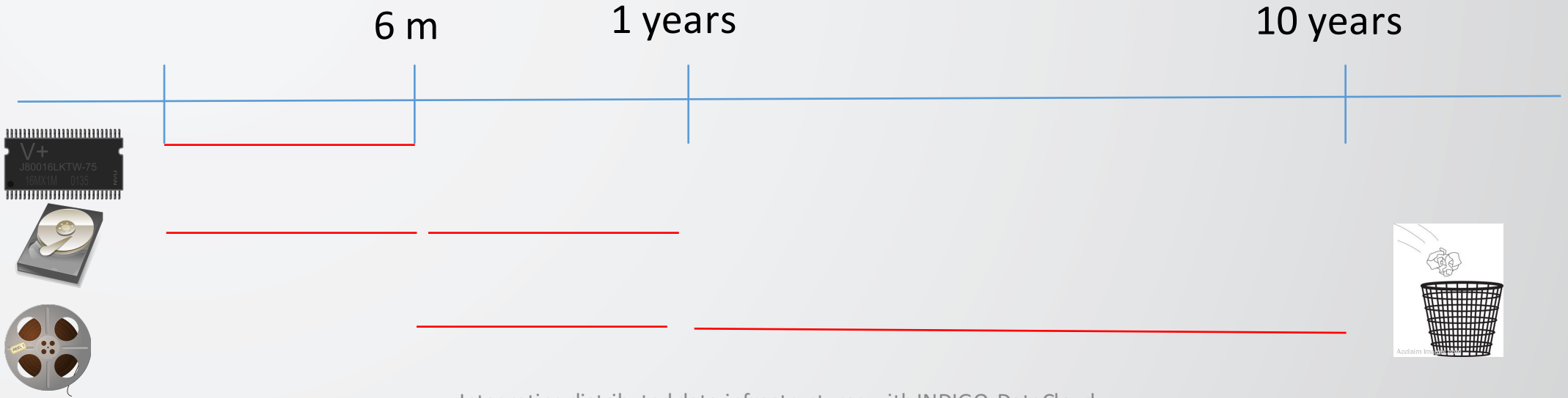
- Provide an advanced set of features that includes:
  - User authentication (supporting SAML, OIDC, X.509)
  - Identity harmonization (link heterogeneous AuthN mechanisms to a single VO identity)
  - Management of VO membership (i.e., groups and other attributes)
  - Management of registration and enrolment flows
  - Provisioning of VO structure and membership information to services
  - Management, distribution and enforcement of authorization policies

Storage Quality of Service and the Cloud

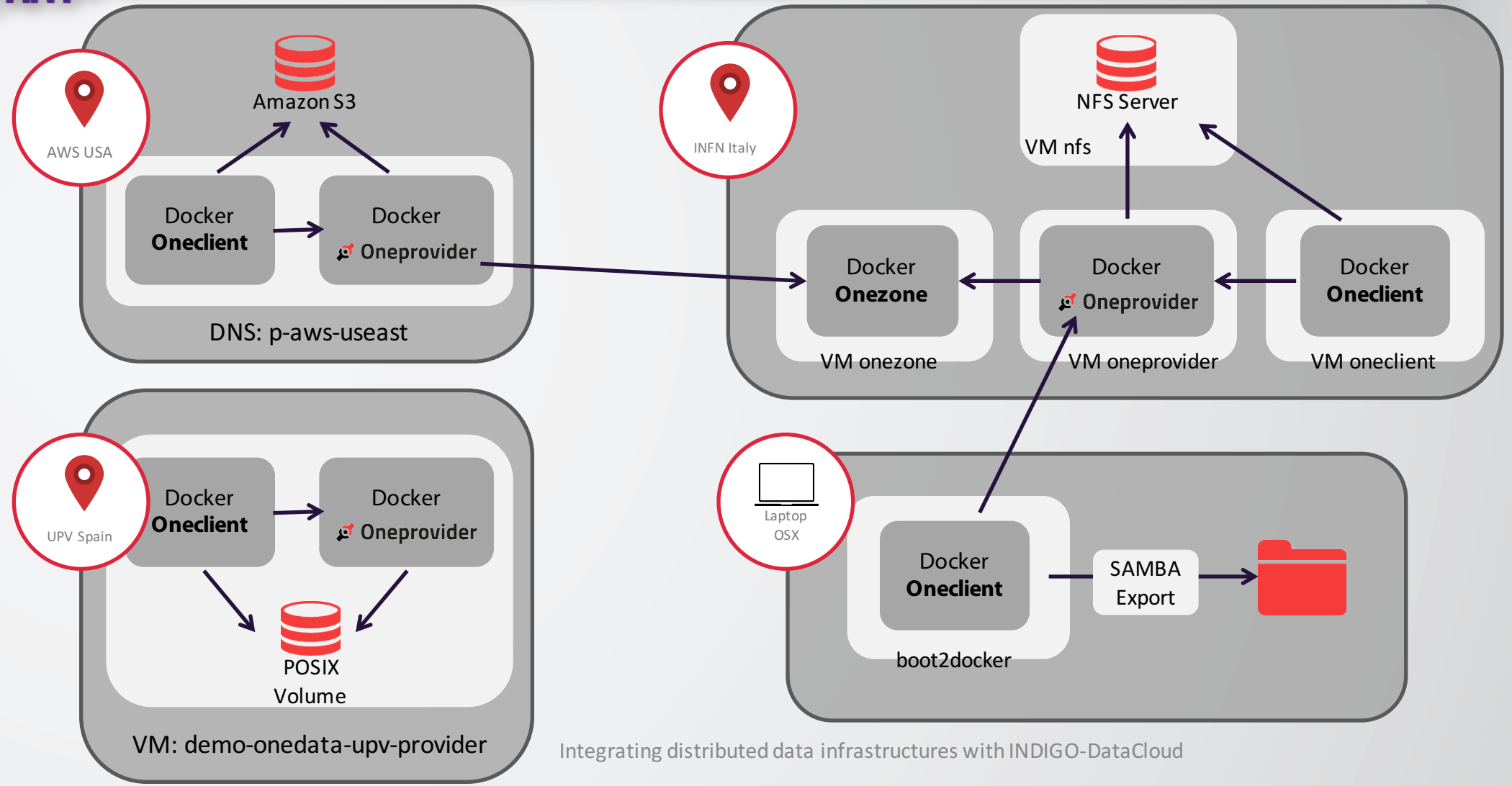
Amazon	S3		Glacier
Google	Standard	Durable Reduces Availability	Nearline
HPSS/GPFS	Corresponds to the HPSS Classes (customizable)		
dCache	Resilient	disk+tape	TAPE

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



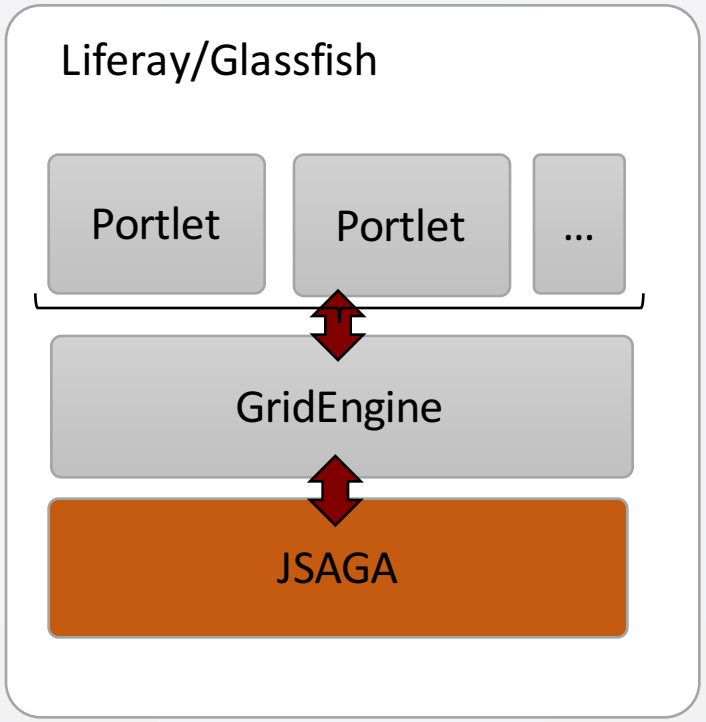
Frontend Services/Toolkit

Integration schemas

- We provide the graphical user interfaces in the form of the scientific gateways and workflows and the way to access the INDIGO PaaS services and software stack, and allow define and set up the on-demand infra for the WP2 use cases.
  - Setting up whole use case infrastructure: The administrator will be provided with the ready to use receipts that he will be able to customize. The final users will be provided with the service end-points 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 INDIGO whole software stack.
  - Use of the INDIGO tools and portals, including the FutureGateway, Scientific Workflows Systems, Big Data Analytics Frameworks (like Ophidia), Mobile Applications. In this scenario the final users as well as domain administrators will use the GUI tools. The administrator will use it as described in first case. In addition domain specific users will be provided with specific portlets/workflows/apps that will allow graphical interaction with their applications run via INDIGO software stack.

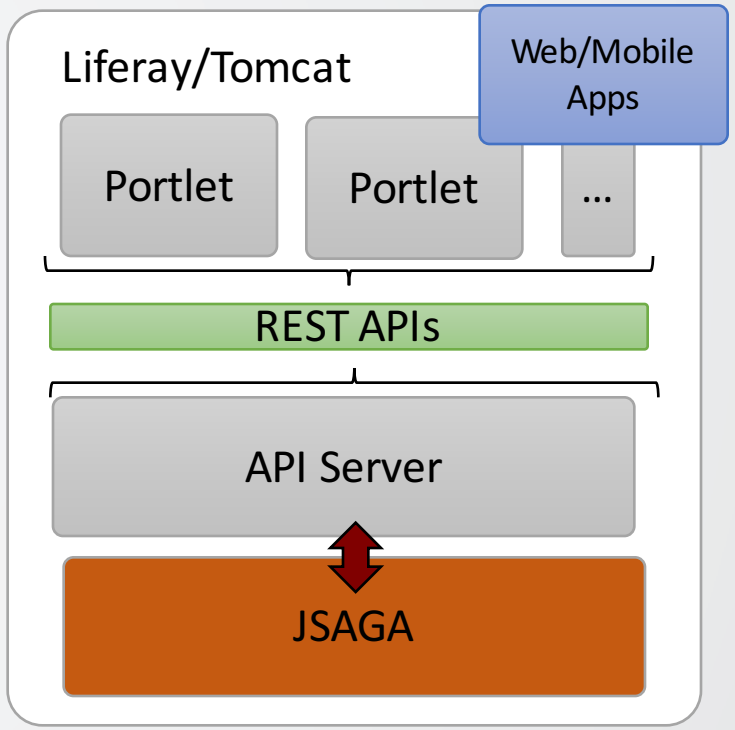
From CSGF to FutureGateway

Classic **CSGF** (before INDIGO)



Comunication Portlet-GridEngine-JSAGA only possible with JAVA libraries

FutureGateway Approach (**INDIGO**)

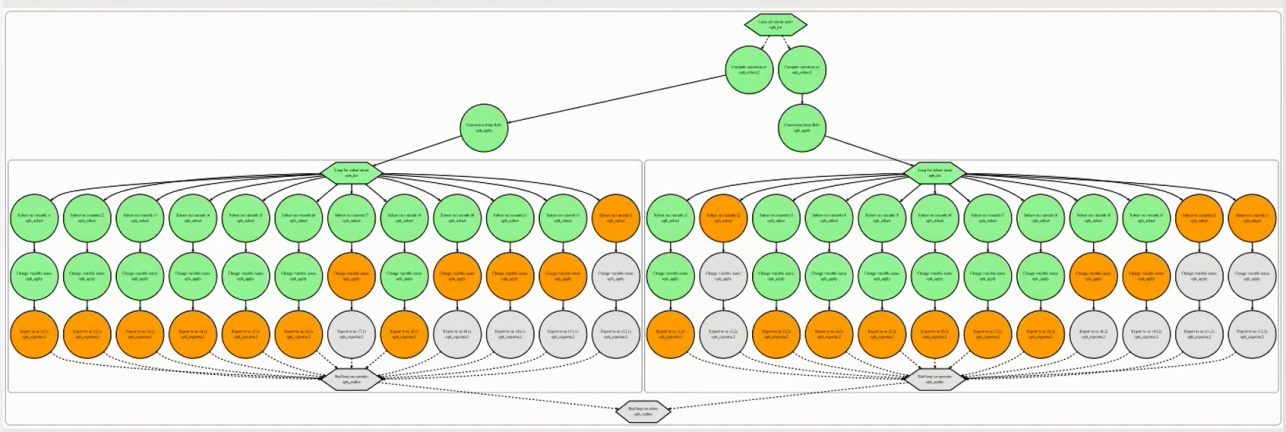


Comunication Portlet-API Server via REST APIs, this allows to serve external applications
The API Server interacts via JAVA libraries to JSAGA

- The same REST APIs could be used by Mobile Apps
- Those APIs make easier the interaction with the PaaS layer
- Those REST APIs provide an easy exploitation of INDIGO Capabilities to non-INDIGO Applications

Ophidia framework

- **Ophidia** is a big data analytics framework for eScience
  - Primarily used for the analysis of climate data, exploitable in multiple domains
  - “Datacube” abstraction and OLAP-based approach for big data
  - Support for array-based data analysis and scientific data formats
  - Parallel computing techniques and smart data distribution methods
  - ~100 array-based primitives and ~50 datacube operators
    - i.e.: data sub-setting, data aggregation, array-based transformations, datacube roll-up/drill-down, data cube import, etc.



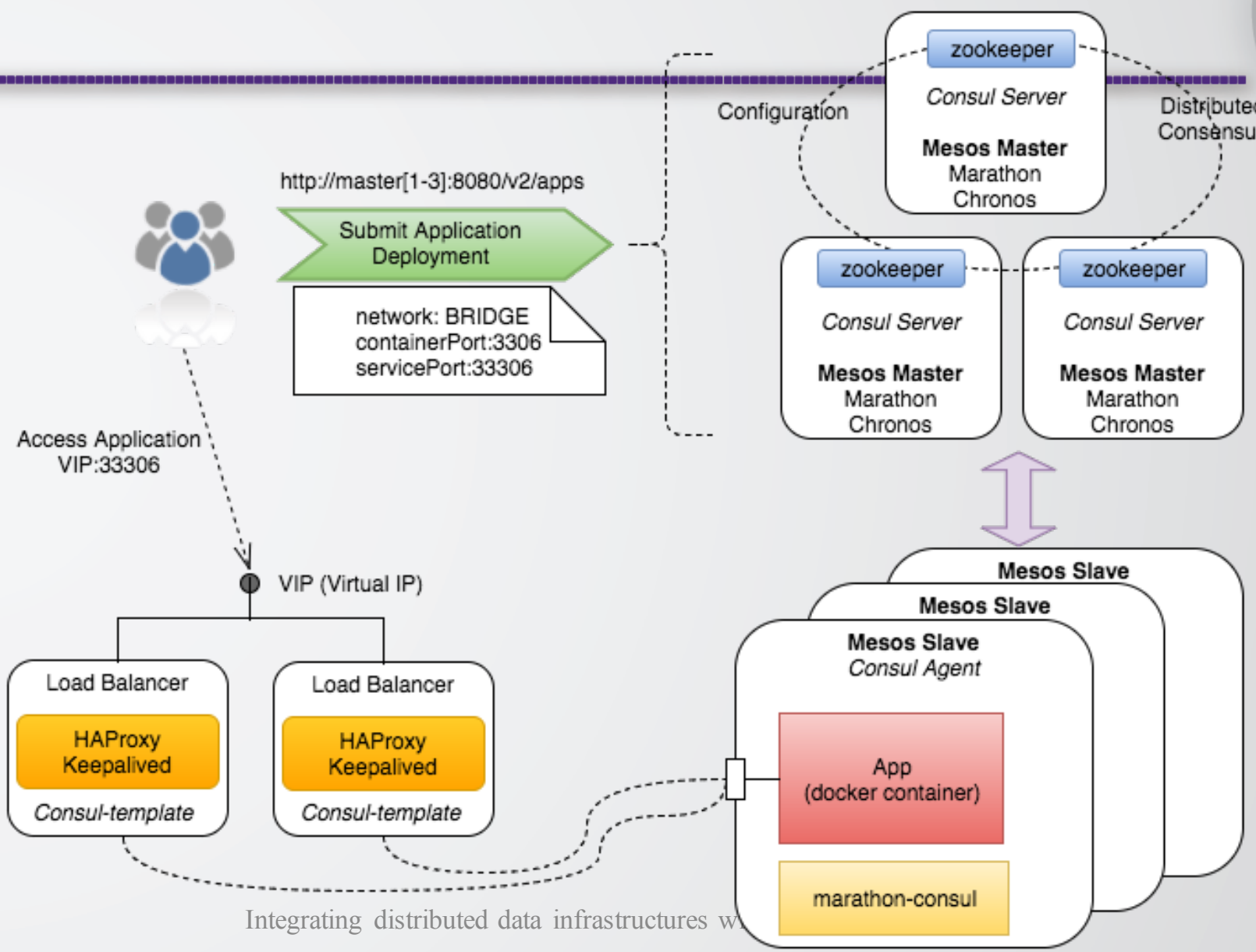
INDIGO module for Kepler

- The Kepler scientific workflow system is an open source tool that enables creation, execution and sharing of workflows across a broad range of scientific and engineering disciplines.
- First version of the INDIGO module delivered, gradually added new functionalities available for the users.
- INDIGO module based on the FutureGateway API
- At the moment, it is possible to build workflows that define task, prepares inputs and triggers execution. While a task is executed within INDIGO's infrastructure, it is possible to check its status.
- FutureGateway API client: https://github.com/indigo-dc/indigoclient
- Kepler based actors: https://github.com/indigo-dc/indigokepler

Use cases examples

Service Deployment and application execution

We are now working on adding a Calico network configuration



Application execution to the PaaS Layer

- The INDIGO approach to the application distribution and execution is:
  - Based on Docker
  - Exploits Mesos+Chronos
  - All the application executions are described exploiting a TOSCA Templates via simple APIs or Portlets
  - The input/output are automatically managed by the PaaS layer (via Onedata and external endpoints)
  - Dependencies, retry on failures supported by means of Chronos
  - Geographical data-aware scheduling provide by INDIGO PaaS orchestrator

```
chronos_job_1:
    type: tosca.nodes.indigo.Container.Application.Docker.Chronos
    properties:
      schedule: 'R0/2015-12-25T17:22:00Z/PT5M'
      description: 'Execute app'
      command: /bin/bash run.sh
      uris: []
      retries: 3
      environment_variables:
        INPUT_ONEDATA_SPACE: { get_input: InputOnedataSpace }
        INPUT_PATH: { get_input: InputPath }
        ....
    artifacts:
      image:
        file: indigodatacloud/ambertools_app
        type: tosca.artifacts.Deployment.Image.Container.Docker
    requirements:
      - host: docker_runtime1

 chronos_job_upload:
    type: tosca.nodes.indigo.Container.Application.Docker.Chronos
    properties:
      schedule: 'R0/2015-12-25T17:22:00Z/PT5M'
      description: 'Upload output data'
      command: /bin/bash run.sh
      retries: 3
      environment_variables:
        PROVIDER_HOSTNAME: <ONEDATA_PROVIDER_IP>
        ONEDATA_TOKEN: <ROBOT Token>
        ONEDATA_SPACE: <path>
        INPUT_FILENAME:  <input filename>
        OUTPUT_FILENAME:  <input filename --> coincindes with amber-job-01 OUTPUT_FILENAME>
        OUTPUT_PROTOCOL: http(s)|ftp(s)|S3|Swift|WebDav
        OUTPUT_URL:  <output URL>
        OUTPUT_CREDENTIALS:  <e.g. username:password>
    artifacts:
      image:
        file: indigodatacloud/jobuploader
        type: tosca.artifacts.Deployment.Image.Container.Docker
    requirements:
      - host: docker_runtime1
                        - job_predecessor: chronos_job_1

  docker_runtime1:
    type: tosca.nodes.indigo.Container.Runtime.Docker
    capabilities:
      host:
        properties:
          num_cpus: 0.5
          mem_size:  512 MB
```

```
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 examples for specifying a Chronos Job that runs an application using Onedata storage.
inputs:
 input_onedata_token:
  type: string
  description: User token required to mount the user's INPUT Onedata space
  required: yes
 output_onedata_token:
  type: string
  description: User token required to mount the user's OUTPUT Onedata space. It can be the same as the input token
  required: yes
 # data_locality:
 #  type: boolean
 #  description: Flag that controls the INPUT data locality: if yes the orchestrator will select the best provider, if no the user has to specify the provider to be used
 #  required: yes
 input_onedata_providers:
  type: list
  description: List of favorite Onedata providers to be used to mount the Input Onedata space. If not provided, data locality algo will be applied.
  entry_schema:
   type: string
  default: ['']
  required: no
 output_onedata_providers:
  type: list
  description: List of favorite Onedata providers to be used to mount the Output Onedata space. If not provided, the same provider(s) used to mount the input space will be used.
  entry_schema:
   type: string
  default: ['']
  required: no
 input_onedata_space:
  required: yes
 output_path:
  type: string
  description: Path to the output data inside the Output Onedata space
  required: yes
 output_filenames:
  type: list
  description: List of filenames generated by the application run
  entry_schema:
   type: string
  required: yes
 cpus:
  type: float
  description: Amount of CPUs for this job
  required: yes
 mem:
  type: float
  description: Amount of Memory (MB) for this job
  required: yes
topology_template:
 node_templates:
  chronos_job:
   type: tosca.nodes.indigo.Container.Application.Docker.Chronos
   properties:
    schedule: 'R0/2015-12-25T17:22:00Z/PT5M'
    name: 'JOB_ID_TO_BE_SET_BY_THE_ORCHESTRATOR'
    description: 'Execute app'
    command: '/bin/bash run.sh'
    uris: []
    retries: 3
    environment_variables:
     INPUT_ONEDATA_TOKEN: { get_input: input_onedata_token }
     OUTPUT_ONEDATA_TOKEN: { get_input: output_onedata_token }
     INPUT_ONEDATA_PROVIDERS: { get_input: input_onedata_providers }
     OUTPUT_ONEDATA_PROVIDERS: { get_input: output_onedata_providers }
     INPUT_ONEDATA_SPACE: { get_input: input_onedata_space }
     ....
   artifacts:
    image:
     file: indigodatacloud/ambertools_app
```

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

- A user community maintains a “vanilla” version of portal and 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 Inspiration: Galaxy on 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.
- 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: A web portal that exploits a batch system to run applications

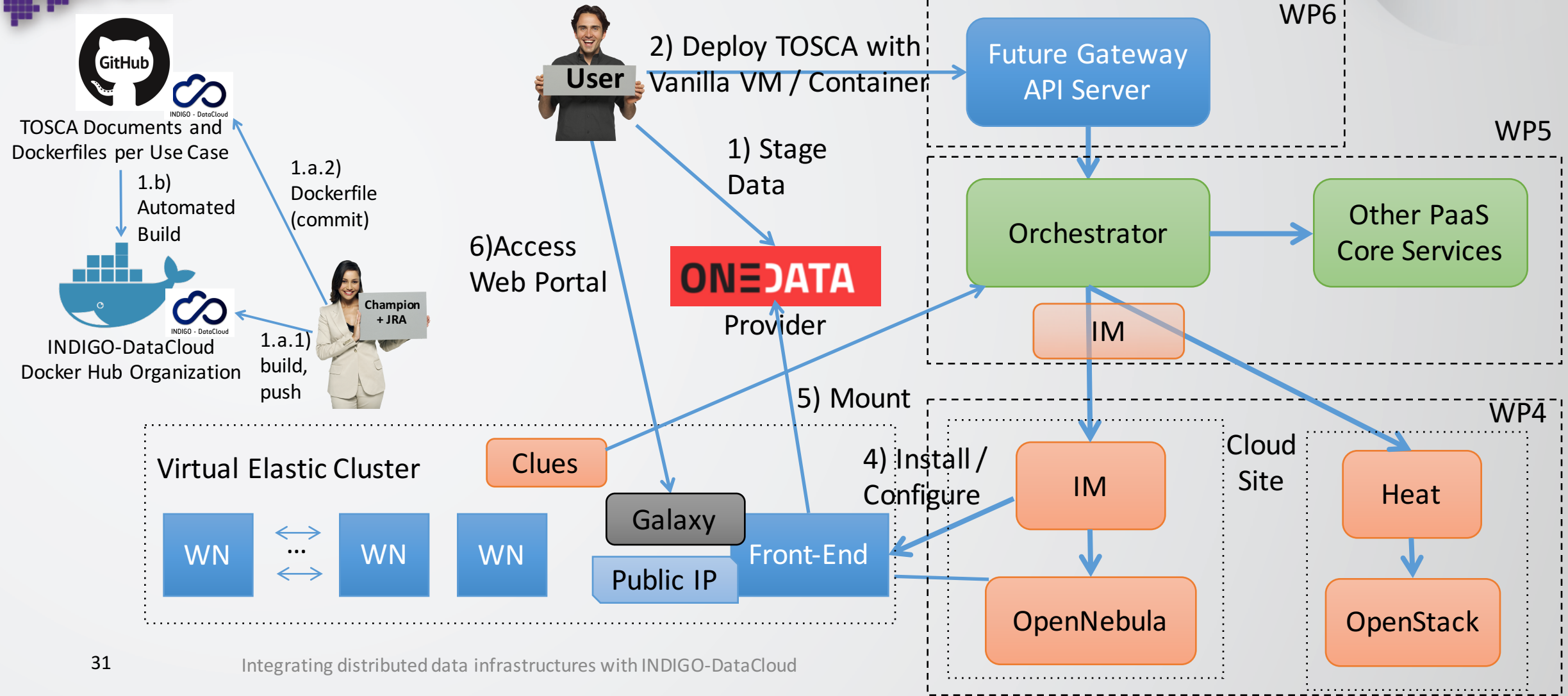
INDIGO - DataCloud

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: Use Case 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: A Graphic Overview



Detailed IaaS Status

nova-docker

- Docker driver for OpenStack Compute (nova).
  - 3rd party component, not from INDIGO, we have sent some patches and bugfixes.
  - Deployment scenario:
    - Install a package in the desired compute nodes.
    - Configure nova and glance to be aware of containers.
  - It will be packaged for Liberty.

ONEDock

- Docker driver for OpenNebula.
- It requires a working OpenNebula 4 installation.
- Deployment scenario:
  - Install and configure ONEDock in the frontend node.
  - Configure the compute nodes with Docker and ONE required config.

Repository Synchronization

INDIGO - DataCloud

- Sync between Docker Hub organization and local registry/catalog via webhooks.
- Modules for both OpenNebula and OpenStack.
- Standalone component with REST API.
- Deployment scenario:
  - Install a package and configure the webhooks.
  - Give access to the sync user to the registry/catalog.

opie

- Preemptible Instances extension for OpenStack Compute (nova)
- Needs a working OpenStack Compute (nova) environment.
- API, Scheduler and HostManager as pluggable external modules.
- Deployment scenario:
  - Installation of a package + a *manual* modification (i.e. applyinga patch) on the compute API.
  - Configure nova (i.e. nova.conf) to use the new HostManager and
  - PreemtibleFilterScheduler.
  - It will be released for Liberty.

Synergy

- Fair share scheduling support for OpenStack.
- Needs a working OpenStack Compute (nova) environment.
- External and standalone component, no modifications needed.
- Deployment scenario:
  - Install synergy as a separate service, configure it to interact with OpenStack.
  - It will be released for Liberty (for the 1st release).

uDocker

- Run containers in batch systems in userspace, making possible to download and run containers by non-privileged users.
- It does not require Docker at all.
- Deployment scenario:
  - No deployment. It is a simple python tool.
- This has been tested to run some containers in local HPC system (CSIC), with success, so it may be useful for EGI in the Grid context.

AAI for OpenStack

- The OpenID Connect support has been improved in the Keystone authentication client libraries (it did not work out of the box).
- Deployment scenario:
  - Keystone server configuration needed.
  - Nothing required at the CLI, changes contributed upstream (although not released yet).

AAI for OpenNebula

- Based on the Token Translation Service (TTS).
- Deployment scenario:
  - No installation required at the site level, only configuration (similar to PERUN).

INDIGO FAQ

- How do INDIGO achieve resource redundancy and high availability?
  - This is achieved at multiple levels:
    - at the data level, redundancy can be implemented exploiting the capability of INDIGO's Onedata of replicating data across different data centers.
    - at the site level, it is possible to ask for copies of data to be for example on both disk and tape using the INDIGO QoS storage features.
    - for services, the INDIGO architecture uses Mesos and Marathon to provide automatic service high-availability and load balancing. This automation is easily obtainable for stateless services; for stateful services this is application-dependent but it can normally be integrated into Mesos through, for example, a custom framework (examples of which are provided by INDIGO).
- How do INDIGO achieve resource scalability?
  - First of all, we can distinguish between vertical (scale up) and horizontal (scale out) scalability. INDIGO provides both:
    - Mesos and Marathon handle vertical scalability by deploying Docker containers with an increasing amount of resources.
    - The INDIGO PaaS Orchestrator handles horizontal scalability through requests made at the IaaS level to add resources when needed.

INDIGO FAQ

- How do INDIGO achieve resource scalability?
  - The INDIGO software does this in a smart way, i.e. for example it does not look at CPU load only:
    - In the case of a dynamically instantiated LRMS, it checks the status of jobs and queues and accordingly adds or remove computing nodes.
    - In the case of a Mesos cluster, in case there are applications to start and there no free resources, INDIGO starts up more nodes. This happens within the limits of the submitted TOSCA templates. In other words, any given user stays within the limits of the TOSCA template he has submitted; this is true also for what regards accounting purposes.
- How do you know when and where resources are available?
  - We are extending the Information System available in the European Grid Infrastructure (EGI) to inform the INDIGO PaaS orchestrator about the available IaaS infrastructures and about the services they provide. It is therefore possible for the INDIGO orchestrator to optimally choose a certain IaaS infrastructure given, for example, the location of a certain dataset.

Conclusions

- First official release will be: 1st August
- The first prototype is already available:
  - Not all the services and features are available
  - This is for internal evaluation, but already some services could be tested
- A lot of important development are being carried on with the original developers community so that the code mantenance is not (only) in our hands

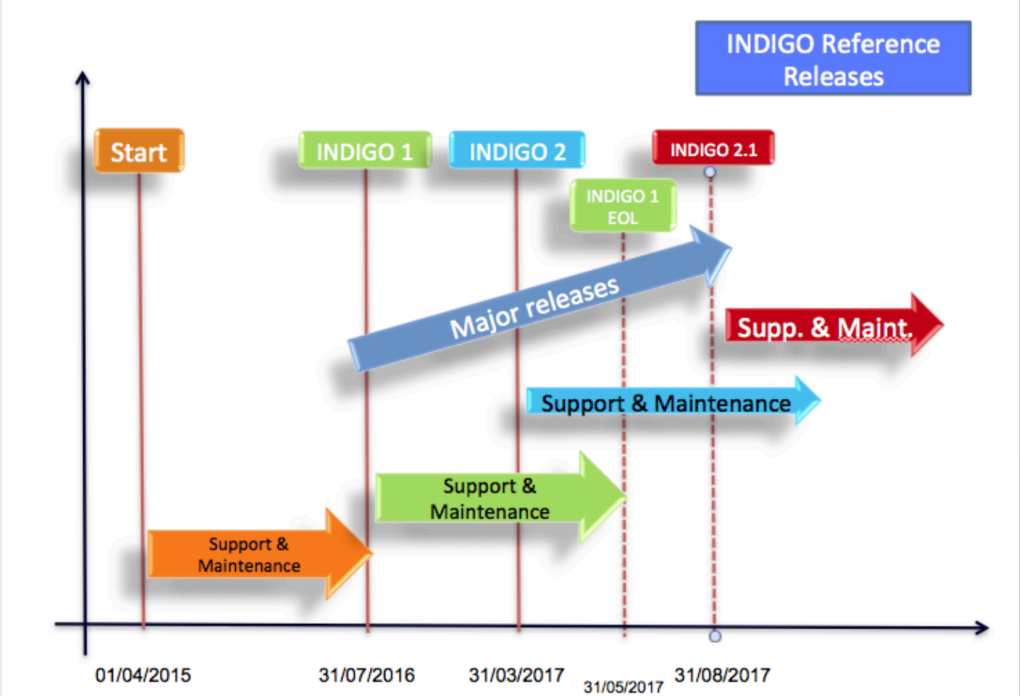


Figure 8: INDIGO-DataCloud Release Timeline

Thank you

INDIGO - DataCloud

https://www.indigo-datacloud.eu

Better Software for Better Science.