INDIGO-DataCloud: Project Achievements

by the **INDIGO-DataCloud Collaboration**

,
D. Salomoni[1], I. Campos[2], L. Gaido[3], J. Marco de Lucas[2], P. Solagna[14], J. Gomes[9],
L. Matyska[13], P. Fuhrman[5], M. Hardt[8], G. Donvito[4], L. Dutka[13], M. Plociennik[7], R.
Barbera[11], C. Aiftimiei[1], I. Blanquer[6], A. Ceccanti[1], M. David[9], A. López-García[2],
G. Moltó[6], P. Orviz[2], Z. Sustr[13], M. Viljoen[14], F. Aguilar[2], L. Alves[9], A. Bonvin[16],
R. Bruno[11], D. Davidovic[18], M. Fargetta[11], Y. Chen[14], S. Fiore[10], Z. Kurkcuoglu[16],
L. Lloret[2], J. Martins[9], A. Nuzzo[10], P. Nassisi[10], J. Pina[9], E. Sciacca[17], D. Spiga[12],
M. Tangaro[4], M. Urbaniak[7], B. Wegh[8], and T. Zok[7]

[1]INFN Division CNAF (Bologna), Italy
[2]IFCA, Consejo Superior de Investigaciones Cientificas-CSIC, Santander (Spain)
[3]INFN Division of Torino, Italy
[4]INFN Division of Bari, Italy
[5]Deutsches Elektronen Synchrotron (DESY), Germany
[6]Universidad Politecnica de Valencia, Spain
[7]PSNC IBCh PAS, Poland
[8]Karlsruhe Institute of Technology (KIT), Germany
[9]Laboratory of Instrumentation and Experimental Particle Physics (LIP), Portugal
[10]Fondazione Centro Euro-Mediterraneo sui Cambiamenti Climatici , Lecce, Italy
[11]INFN Division of Catania, Italy
[12]INFN Division of Perugia, Italy
[13]Cyfronet AGH, Krakow, Poland
[14]European Grid Initiative Foundation, EGI.eu, Amsterdam (Netherlands)
[15]CESNET, Prague, Czech Republic
[16]University of Utrecht, The Netherlands
[15]Istituto Nazionale di Astrofisica, Italy
[15]Ruder Boskovic Institute, Zagreb (Kroatia)

November 13, 2017

Abstract

This paper describes the achievements of the H2020 project INDIGO-DataCloud. The project has provided e-infrastructures with tools, applications and cloud framework enhancements to manage the demanding requirements of scientific communities, either locally or through enhanced interfaces. The middleware developed allows to federate hybrid resources, to easily write, port and run scientific applications to the cloud. Our developments are freely downloadable as open source components, and are already being integrated into many scientific applications.

1

1 Introduction

INDIGO-DataCloud started in April 2015, with the purpose of developing a modular architecture and software components to improve how scientific work is supported at the edge of computing services development.

The project main goal has been to deliver a Cloud platform addressing the specific needs of scientists in a wide spectrum of disciplines, engaging public institutions and private companies. It aimed at being as inclusive as possible, developing open source software exploiting existing solutions, adopting and enhancing state of the art technologies, connecting with other initiatives and with leading commercial providers.

The project categorized user community requests, and identified common requirements, classifying them into the Computational, Storage and Infrastructural areas. This process was reviewed across the entire lifetime of the project, which allowed the update of existing requirements as well as the insertion of new ones, thus driving the technological developments of the project

Following this requirement analysis, the project identified the technology gaps linked to concrete use cases, which were subsequently used to validate the technical implementations. Section 2 describes the software developments that took place in the form of tools, applications and cloud framework enhancements to manage the demanding requirements of modern scientific communities at the level of the computing resource centers (i.e. at the Infrastructure as a Service, IaaS, level).

This process resulted in the definition of the INDIGO Technical Architecture: a modular framework, fully based on open standards, covering all areas of the Cloud stack (IaaS, PaaS, SaaS). Section 3 and 4 review the project architecture and the user oriented interfaces respectively.

In order to create production-quality software, the project defined procedures and quality metrics, which were followed by, and automatically checked for, all the INDIGO components. A comprehensive process to package and issue the INDIGO software was also defined. As an outcome of this, INDIGO delivered two main software releases (the first in August 2016, the second in April 2017), each followed by several minor updates.

The description of our software release process can be found in Section 5. The INDIGO products are documented through two service catalogues: a short one, with a high-level description of the INDIGO solutions, and a longer version, with details for each released component and with reports of sample applications.

The latest INDIGO release consists of about 40 open modular components, 50 Docker containers, 170 software packages, all supporting up-to-date open operating systems. This result was accomplished reusing and extending open source software and –whenever applicable –contributing code to upstream projects.

For testing and validation purposes, two large distributed testbeds were created: one for supporting development activities, and another for pre-production applications, open to all the communities participating to the project. These testbeds let the communities integrate INDIGO components into many scientific applications, which are progressively being deployed in production over

public or private infrastructures. Basic and advanced examples of deployment of scientific applications are presented in Section 6.

2 Developing for the Infrastructure as a Service (IaaS) Layer

INDIGO-DataCloud has provided e-infrastructures with tools, applications and cloud framework enhancements to manage the demanding requirements of modern scientific communities, either locally or through enhanced interfaces enabling those infrastructures to become part of a federation or to connect to federated platform layers (PaaS).

In this section we will describe the highlights of this development work, which was needed to properly interface with the resource centers. This work has focussed on virtualizing local compute, storage and networking resources (IaaS) and on providing those resources in a standardized, reliable and performing way to remote customers or to higher level federated services, building virtualized site independent platforms.

The IaaS resources are provided by large resource centers, typically engaged in well-established European e-infrastructures. The e-infrastructure management bodies, or the resource centers themselves will select the components they operate, and INDIGO will have limited influence on that process.

Therefore, INDIGO has concentrated on a selection of the most prominent exisitng components and has further developed the appropriate interfaces to high-level services based on standards. We have also developed new components where we felt a full functionality was completely missing. Figure 1 shows a very schematic view of the interrelation among those components.

The contribution of INDIGO to enhance the flexibility to access resources in Cloud and HPC infrastructures will be of paramount importance to enable transparent execution of applications across systems promoting the development of the future European Open Science Cloud (EOSC) ecosystem [1].

As we describe below new components are provided, or already existing components are improved in the areas of computing, storage, networking and Authorization and Authentication Infrastructure (AAI). For almost all components, we succeeded in committing modifications to the corresponding upstream software providers and by that, significantly contributed to the sustainability model of the software.

2.1 Support to transparent execution of containers

INDIGO has provided support for container execution, both interactively and through batch systems, via trusted container repositories and using open standard interfaces.

The key middleware developed for this purpose is **udocker**[1]. Udocker en-

---

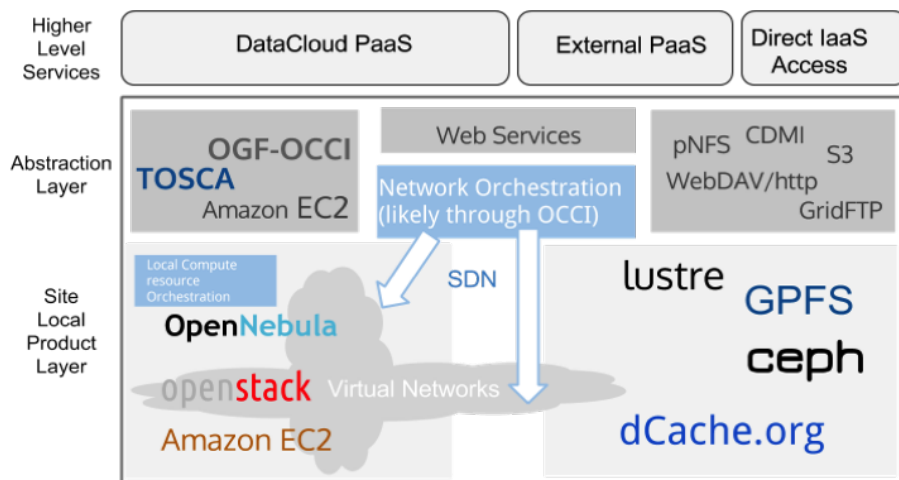[1]https://github.com/indigo-dc/udocker

Figure 1: Components at the Infrastructure as a Service level and their relation with the INDIGO developments

ables the execution of Docker[6] containers without Docker with three different engines based on PRoot[7], runC[8] and a Fakechroot[9]. It can be used to execute containers in batch or interactive systems where Docker cannot be used or is unavailable. None of the udocker engines requires root privileges being therefore adequate for deployment and use by end-users. By using udocker it is possible to use GPU computing and MPI without software installation or system administration intervention, making it adequate to execute containers in batch systems and HPC clusters. udocker grew to become a tool that integrates several existing technologies to run Docker containers in unprivileged mode.

Since its first release in June 2016 udocker expanded quickly in the open source community as can be seen from the github metrics (number of stars, forks or number of external contributions). It has been adopted by a number software projects to substitute of Docker. Among them openmole, bioconda, common-work language (cwl) or SCAR - Serverless Container-aware ARchitectures.

As an example, udocker is being used with great success to execute code produced by the MasterCode collaboration[10]. The MasterCode collaboration is concerned with the investigation of Supersymmetric models that go beyond the current status of the Standard Model of particle physics. It involves teams from CERN, DESY, Fermilab, SLAC, CSIC, INFN, NIKHEF, Imperial College London, King's College London, the Universities of Amsterdam, Antwerpen, Bristol, Minnesota and ETH Zurich

INDIGO has also developed **bdocker**[2]. It enables execution of containers in batch systems by enabling execution of containers using Docker under the

---

[2] https://github.com/indigo-dc/bdocker

control of bdocker. It has been implemented and tested with SoGE. Provides the advantage of supporting the native Docker as execution method.

2.2 Development of advanced Scheduling Technologies

INDIGO pushed the state of the art in scheduling technologies by implementing preemptible-instances on top of the OpenStack[5] cloud management framework, **opie**[3]. Openstack preemptible instances is the materialization of the preemptible instances extensión, serving as a reference implementation.

Preemptible instances differ from regular ones in that they are subject to be terminated by a incoming request for provision of a normal instance. If bidding is in place, this special type of instance could also be stopped by a higher priority preemptible instance (higher bid). Not all the applications are suitable for preemptible execution, only fault-tolerant ones can withstand this type of execution. On the other side they are highly affordable VMs that allow providers to optimize the usage of their available computing resources (i.e. backfilling).

The opie package provides a set of pluggable extensions for OpenStack Compute (nova) making possible to execute preemptible instances using a modified filter scheduler. This solution has gained great interest from the scientific community and commercial partners, and is under discussions to be introduced in the upstream OpenStack scheduler.

The project has also developed support for advanced scheduling policies such as intelligent job allocation based on fair-share algorithms, deployment of tailored clusters using the **Infrastructure Manager**[4] (IM), which has extended its capabilities in the framework of the project.

IM is a tool that deploys complex and customized virtual infrastructures on IaaS Cloud deployments (such as AWS, OpenStack, etc.). It eases the access and the usability of IaaS clouds by automating the VMI (Virtual Machine Image) selection, deployment, configuration, software installation, monitoring and update of the virtual infrastructure. It supports APIs from a large number of virtual platforms, making user applications cloud-agnostic.

It is used by the INDIGO Orchestrator (see section 3) to contact Cloud sites to finally deploy the VMs/containers.

In OpenStack based IaaS private clouds the computing and storage resources are statically partitioned among projects. A user typically is member of one project, and each project has its own fixed quota of resources defined by the cloud administrator. A user request is rejected if the project quota has been already reached, even if unused resources allocated to other projects would be available.

This rigid resource allocation model strongly limits the global efficiency of the data centres, which aim to fully utilize their resources for optimizing costs. In the traditional computing clusters the utilization efficiency is maximized through the use of a batch system with sophisticated scheduling algorithms plugged in.

---

[3]https://github.com/indigo-dc/opie
[4]https://github.com/indigo-dc/im

Synergy[5] is an advanced service interoperable with the OpenStack components, which implements a new resource provisioning model based on pluggable scheduling algorithms. It allows to maximize the resource usage, at the same time guaranteeing a fair distribution of resources among users and groups. The service also provides a persistent queuing mechanism for handling those user requests exceeding the current overall resource capacity. These requests are processed according to a priority defined by the scheduling algorithm, when the required resources become available.

2.3 Development of Authorization and Authentication Infrastructures

INDIGO has provided the necessary components to offer a commonly agreed Authentication and Authorization Infrastructure (AAI). The INDIGO **IAM**[6] (Identity and Access Management service) provides user identity and policy information to services so that consistent authorization decisions can be enforced across distributed services.

IAM has a big impact on the end-user experience. It provides a layer where identities, enrollment, group membership and other attributes and authorization policies on distributed resources can be managed in a homogeneous way, supporting the federated authentication mechanisms supported by the INDIGO AAI.

To develop IAM, the INDIGO AAI team was the first to use OpenID Connect (OIDC) on the SP-IdP proxy and INDIGO has made contributions to the upstream components whenever needed to enable OpenID Connect (namely in OpenStack Keystone and Apache Libcloud). See Figure 2

IAM is used internally to the PaaS layer in order to deal with the authorization of each user to the services, but also in order to deal with group membership and role management for each user. Users may present themselves with any of the three supported token types (X.509, OpenID Connect, SAML) and are able to access resources that support any of them.

All the interaction across all the services will require an authentication based on a well-scoped token. The IAM service also provides the basic for building the Token Translation Service.

WaTTS[7] is the INDIGO Token Translation Service. WaTTS allows using any legacy service with federated identities, such as eduGain or Google. For this, WaTTS accepts federated identities (via OpenID Connect) and uses a plugin scheme to generate credentials for your service. This allows the provision of services that do not normally support federated identities to federated users.

---

[5]https://github.com/indigo-dc/synergy-service
[6]https://github.com/indigo-dc/iam
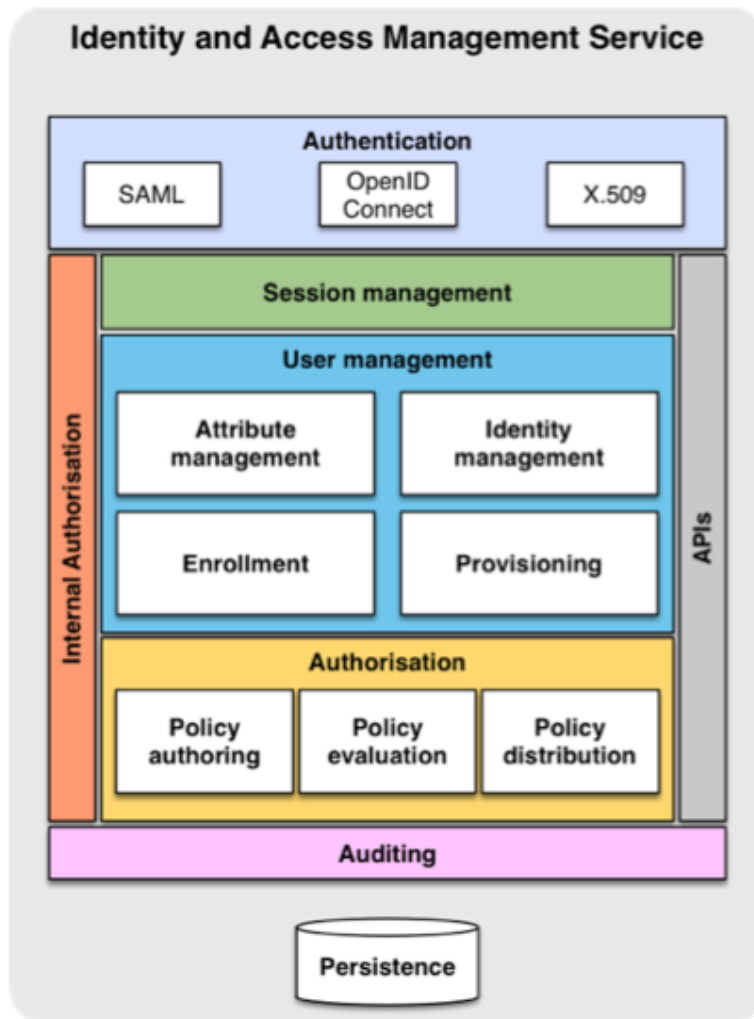[7]https://github.com/indigo-dc/tts

Figure 2: Architecture of the INDIGO Identity and Access management Service

2.4 Virtual Networks

The Virtual Networking Task in INDIGO-DataCloud focused on controlling the computer network environment that interconnects other virtual resources, namely virtual machines and storage services, in the cloud. This work has been addressed in two distinguished areas

1. Intra-cloud networking, i.e., making sure the immediate network environment (public/private accessibility, address space, etc.) within a single provider's cloud site can be controlled by the user in a uniform and standardized way.

2. Inter-cloud networking, i.e., spanning networks between separate cloud sites, potentially geographically distant, so that a user's custom networking environment can be set up in accordance with their needs even if their resources are allocated in separate cloud sites.

Inter-site networking became a basis for the development of the INDIGO Virtual Router component that can be used to span a virtual network across multiple provider sites in geographically distant areas.

For OpenNebula[4] sites the solution consists in a **Network Orchestrator Wrapper** (NOW) [8] and a corresponding backend in the rOCCI-server. NOW enforces site-wide policy and network configuration by making sure that only LANs designated by site administrators are made available to users, and that users cannot reuse LANs assigned to others while they remain reserved. NOW has been released with INDIGO, and the backed has been provided as a contribution to upstream rOCCI-server distribution.

For Open Stack, the OpenStack OCCI Interface (OOI) has been extended with support for advanced networking functions provided by OpenStack's Neutron component. The contribution was accepted upstream and distributed with OOI.

The verification of networking features in the OCCI gateway to Amazon's EC2 interface was performed and minor adjustments were made to make sure that the model of setting up and using local virtual networks was the same regardless of which of the relevant cloud management frameworks (OpenNebula, OpenStack, Amazon Web Services) being used.

3   Architecture of the platform as a service

Generally speaking, a Platform as a Service (PaaS) is a software suite, which is able to receive programmatic resource requests from end users, and execute these requests provisioning the resources on some e-infrastructures. We can see already many examples in the industrial sector, in which open source PaaS solutions (eg. OpenShift[14] or Cloudfoundry[15]) are being deployed to support the work of companies in different sectors.

The case of supporting scientific users is more complex in general than supporting commercial activities, because of the heterogeneous nature of the infrastructures at the IaaS level (i.e. the resource centers) and of the inherent
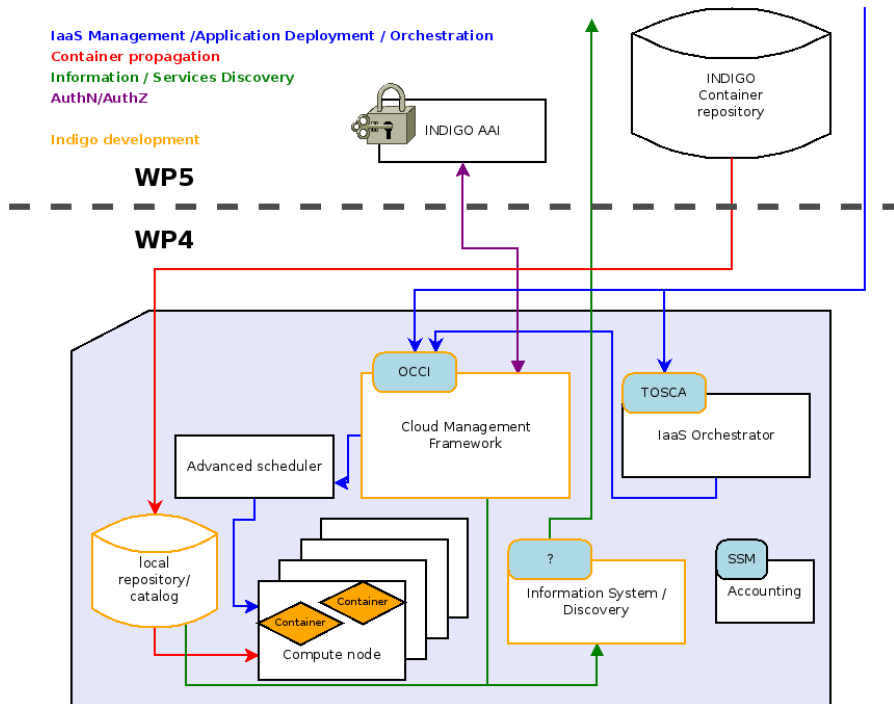
---

[8]https://github.com/indigo-dc/now

Figure 3: Interaction between the IaaS and PaaS layers

complexity of the scientific work requirements. The key point is to find the right agreement to unify interfaces between the PaaS and IaaS levels.

In Figure 3 we show the general interaction between the IaaS and PaaS layers.

INDIGO has provided a working PaaS Layer orchestrating heterogeneous computing and storage resources. Using PaaS Orchestrator together with the IM and TOSCA Templates, the end users are able to exploit computational resources without knowledge about the IaaS details.

In the following we describe the main technologies employed to build the PaaS.

3.1 PaaS layer and microservices architecture

The Paas layer should be able to hide complexity and federate resources for both Computing and Storage. For that we have applied the current technologies based on lightweight containers and related virtualization developments using on microservices.

Kubernetes[2], an open source platform to orchestrate and manage Docker containersis used to coordinate the microservices in the PaaS layer. Kubernetes is extremely useful for the monitoring and scaling of services, and to ensure their
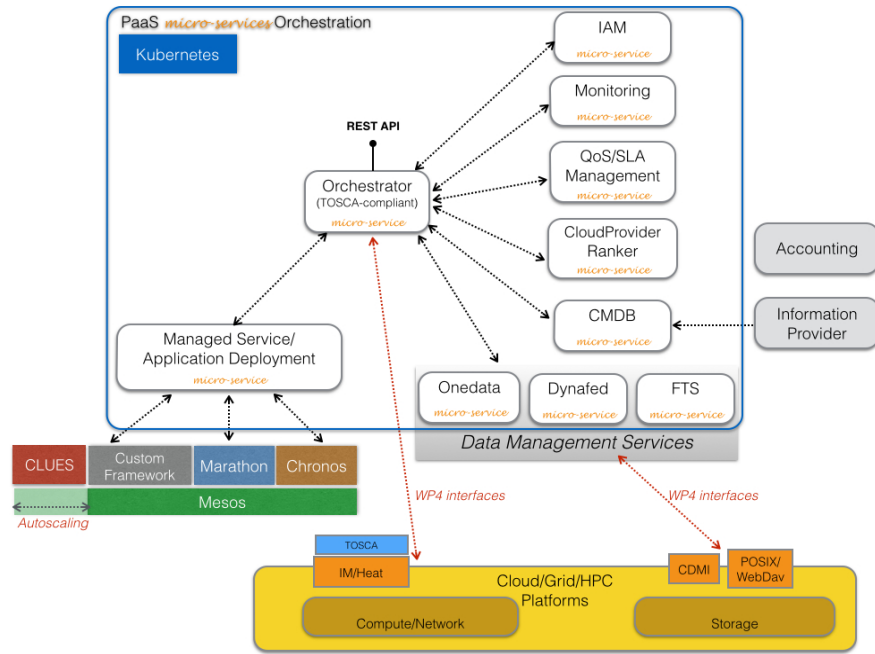
Figure 4: Architecture of the INDIGO Platform as a Service layer.

reliability. The PaaS manages the needed micro-services using Kubernetes, in order, for example, to select the right end-point for the deployment of applications or services. The Kubernetes solution is used in the PaaS layer as is provided by the community.

The microservices that compose the PaaS layer are very heterogeneous in terms of development: some of them are developed ad hoc, some others were already available and are used as they are, few others are deeply modified in order to implement new features within INDIGO.

The language in which the INDIGO PaaS receives end user requests is TOSCA[3]. TOSCA stands for Topology and Orchestration Specification for Cloud Applications. It is an OASIS specification for the interoperable description of application and infrastructure cloud services, the relationships between parts of these services, and their operational behaviour.

TOSCA has been selected as the language for describing applications, due to the wide-ranging adoption of this standard, and since it can be used as the orchestration language for both OpenNebula (through the IM) and OpenStack (through Heat).

The released INDIGO PaaS layer (see Figure 4) is able to provide automatic distribution of the application and/or services over a hybrid and heterogeneous set of IaaS infrastructures, on both private and public clouds.

The PaaS layer is able to accept a description of a complex set, or cluster,
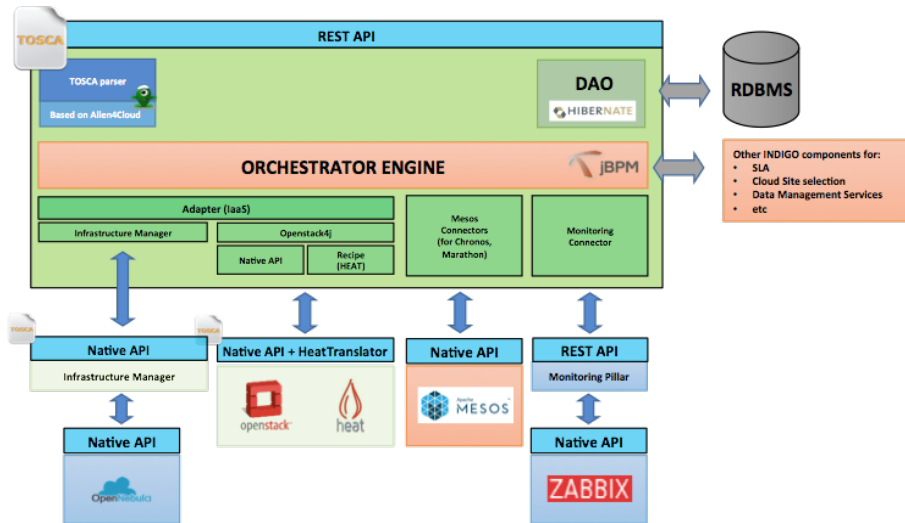
Figure 5: Architecture of the Orchestrator within the PaaS layer.

of services/applications by mean of TOSCA templates, and is able to provide the needed brokering features in order to find the best fitting resources. During this process, the PaaS layer is also able to evaluate data distribution, so that the resources requested by the users are chosen by the closeness to the storage services hosting the data requested by those specific applications/services.

3.1.1 The Orchestrator Engine

The INDIGO PaaS **Orchestrator**[9] is a component of the PaaS layer that allows to instantiate resources on Cloud Management Frameworks (like OpenStack and OpenNebula) and Mesos clusters.

It takes the deployment requests, expressed through templates written in TOSCA, and deploys them on the best cloud site available. In order to do that it gathers SLAs, monitoring info and other data from other platform services and it asks the **Cloud Provider Ranker**[10] for a list of the best cloud sites. Therefore the Orchestrator mission is to coordinated the deployment process over the IaaS platforms. See Figure 5 for an overview of the orchestrator architecture.

The Orchestrator is based on developments already done in other publicly funded projects, as the Italian PONs PRISMA[12], and Open City Platform[13]. During the INDIGO project, this component has been extended and enhanced to support the specific microservices building the INDIGO PaaS Layer. It delegates the actual deployment of resources to IM, OpenStack Heat or Mesos frameworks based on TOSCA templates and the ranked list of sites.

---

[9]https://github.com/indigo-dc/orchestrator
[10]https://github.com/indigo-dc/cloudproviderranker

11

A very innovative component is the Cloud Provider Ranker. This is a standalone REST web service, which ranks cloud providers on the basis of rules defined per user/group/use case, with the aim of fully decoupling the ranking logic from the business logic of the Orchestrator.

It allows the consumers of the service (one or more orchestrators) to specify preferences: if preferences have been specified, then they have absolute priority over any other provider information (like monitoring data); on the contrary, when preferences are not specified, for each provider the rank is calculated, by default, as the sum of SLA ranks and a combination of monitoring data, conveniently normalized with weights specified in the Ranker configuration file (with the last update the ranking algorithm can be customized to the specific needs).

This is a completely new service, fully implemented within the INDIGO project; it is based on an open source tool, Drools[11], in order to reduce the needed development effort, and to simplify the long-term support.

3.1.2 Data Management Services

Data management services are based on three open source components: OneData[16], DynaFed[17] and FTS3[18].

INDIGO has invested a substantial effort in the development of **OneData**[12], this is a global data management system aiming to provide easy access to distributed storage resources. The finall goal is supporting a wide range of use cases from personal data management to data-intensive scientific computations.

The OneData development roadmap includes several data management related functionalities, among the most relevant ones:

- Cloud based data access

- POSIX based data access

- Data location

- Data migration

- Metadata management

- ACL management

- QoS and policy management

The OneData roadmap includes also adding features to support federated HPC applications, allowing transparent access to storage resources from multiple data centers simultaneously. OneData automatically detects whether data is available on local storage and can be accessed directly, or whether it has to be fetched from remote sites in real time.

---

[11]https://www.drools.org
[12]https://github.com/indigo–dc/onedata

Support for federation in OneData can be achieved by the possibility of establishing a distributed provider registry, where various infrastructures can setup their own provider registry and build trust relationship between these instances, allowing users from various platforms to share their data transparently.

OneData provides an easy to use Graphical User Interface for managing storage Spaces, with customizable access control rights on entire data sets or single files to particular users or groups.

4   Interfacing with the users

Users typically do not access the PaaS core directly. They instead often use Graphical User Interfaces or simpler APIs (see Section 4). A user authenticated on the INDIGO Platform can access and customize a rich set of TOSCA-compliant templates through a GUI-based portlet.

The INDIGO repository provides a catalogue of pre-configured TOSCA templates to be used for the deployment of a wide range of applications and services, customizable with different requirements of scalability, reliability and performance.

In those templates a user can choose between two different examples of generic scenarios:

1. Deploy a customized virtual infrastructure starting from a TOSCA template that has been imported, or built from scratch: The user will be able to access the deployed customized virtual infrastructure and run/administer/manage applications running on it.

2. Deploy a service/application whose lifecycle will be directly managed by the PaaS platform: in this case the user will be returned the list of endpoints to access the deployed services.

INDIGO has provided the tools for a simple and effective end user experience, both for software developers and for researchers running the applications.

APIs accessing the INDIGO PaaS layer are available, so that its features can be exploited by Portals, Desktop Applications and Mobile Apps. The final release of INDIGO-DataCloud software includes a large set of components to facilitate the development of Science Gateways and desktop/mobile applications, big data analytics and scientific workflows.

In Figures 6 and 7 we show how to launch an application using a movil platform developed in the project.

In particular, components directly related to end user interfaces are:

- The INDIGO FutureGateway (FG) framework, used to build powerful, customized, easy to use, science gateway environment and front-ends, on top of the INDIGO-DataCloud PaaS layer and integrated with data management services. The FG provides many capabilities, including:
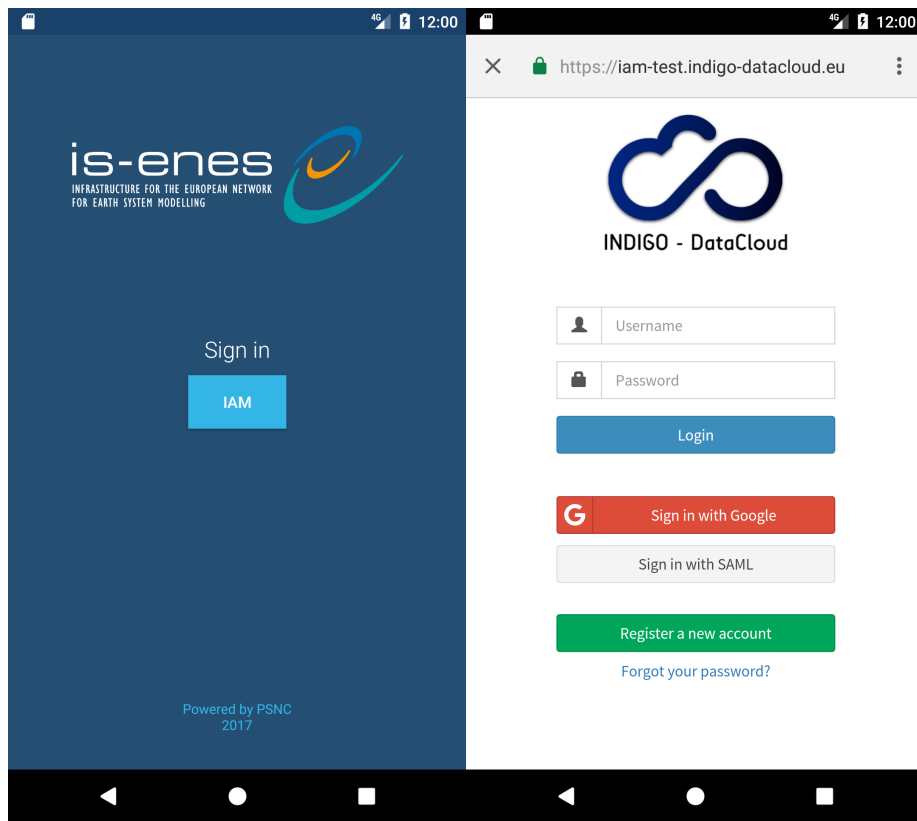
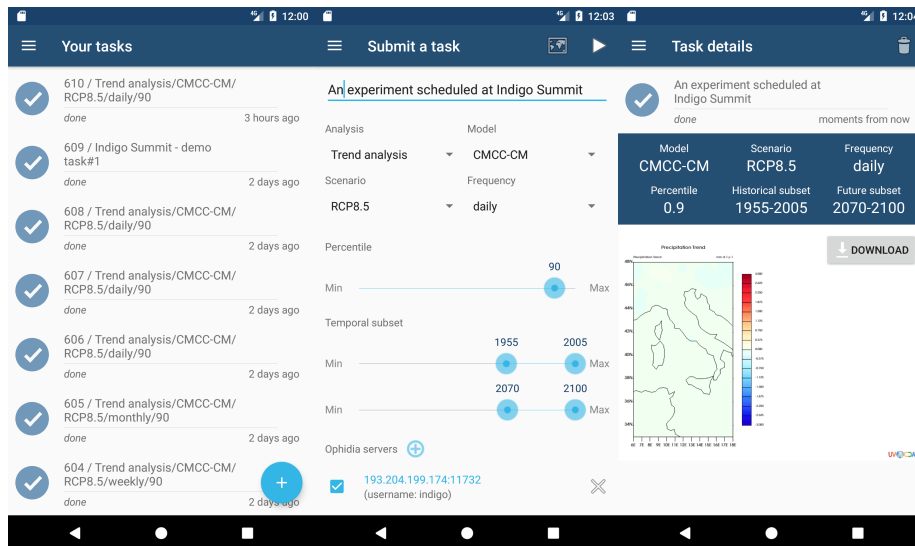Figure 6: Authentication of users on a mobile platform using IAM

Figure 7: Task submission and visualization of results on the mobile platform for ENES

- The FG API server, used to integrate third-party science gateways; the FG Liferay Portal, containing base portlets for the authentication, authorization and administration of running applications and deployments;

- Customizable Application Portlets, for user-friendly specification of the parameters used by TOSCA templates;

- A workflows monitoring portlet, used for monitoring task execution via integrated workflow systems, described below.

- An Open Mobile Toolkit as well as application templates for Android and iOS, simplifying the creation of mobile apps making use of the FG API Server.

- Support for scientific workflows, where the INDIGO components:

  - Provide dynamic scalable services in a Workflows as a Service model;
  - Implement modules and components enabling the usage of the PaaS layer (via FG API Server) for the main scientific workflow engines deployed by user communities (such as Kepler, Ophidia, Taverna,Pegasus);
  - Support a two-level (coarse and fine grained) workflow orchestration, essential for complex, distributed experiments involving (among others) parallel data analysis tasks on large volumes of scientific data.

- Key extensions to the Ophidia big data analytics framework (allowing to process, transform and manipulate array-based data in scientific contexts),

15

providing many new functionalities, including a set of new operators related to data import regarding heterogeneous data formats (e.g. SAC, FITS), a new OpenIDConnect interface and new workflow interface extensions.

- Enhancements of the jSAGA library through a "Resource Management API", complementing the standard Job/Data Management API. This allows to acquire and manage resources (compute, storage, network) and enables the wrapping of underlying technologies (cloud, pilot jobs, grid, etc.) by means of a single API, supporting asynchronous mode (task), timeout management, notification (metrics) and security context forwarding.

- Command-line clients for the PaaS layer to provide an easy way for users to interact with the Orchestrator or with WATTS:

  - **Orchent**[13]: a command-line application to manage deployments and their resources on the INDIGO-DataCloud Orchestrator;
  - **Wattson**[14]: a command-line client for the INDIGO Token Translation Service.

5 Software development and release process

The software development process in INDIGO has been supported by a continuous software improvement process that includes software quality assurance, software release management, maintenance, support services, and the pilot infrastructures needed for software integration and testing.

Following the successful examples of other software development projects like, EGEE I, II, III, and EMI, the development, maintenance and evolution of the software is implemented inside Product Teams (PTs). PTs are small teams of software developers, fully responsible for the development of a particular software product (or group of tightly related products) compliant with agreed sets of technical specification and acceptance criteria.

The software improvement cycle is outlined in Figure 8. It starts with user and infrastructure provider requirements and technical plans which guide the software development and related test plans.

All the software characteristics relevant for quality assurance and release preparation, are described in software specifications (blueprints) provided by the projects' Product Teams for each product going through the process. Software is then developed and continuously tested according to the blueprints and released according to a predefined schedule.

Preview releases are made available for evaluation by user communities and resource providers. Release candidates are subjected to integration testing, which may include the participation of user communities. Once the required

---

[13]https://github.com/indigo-dc/orchent
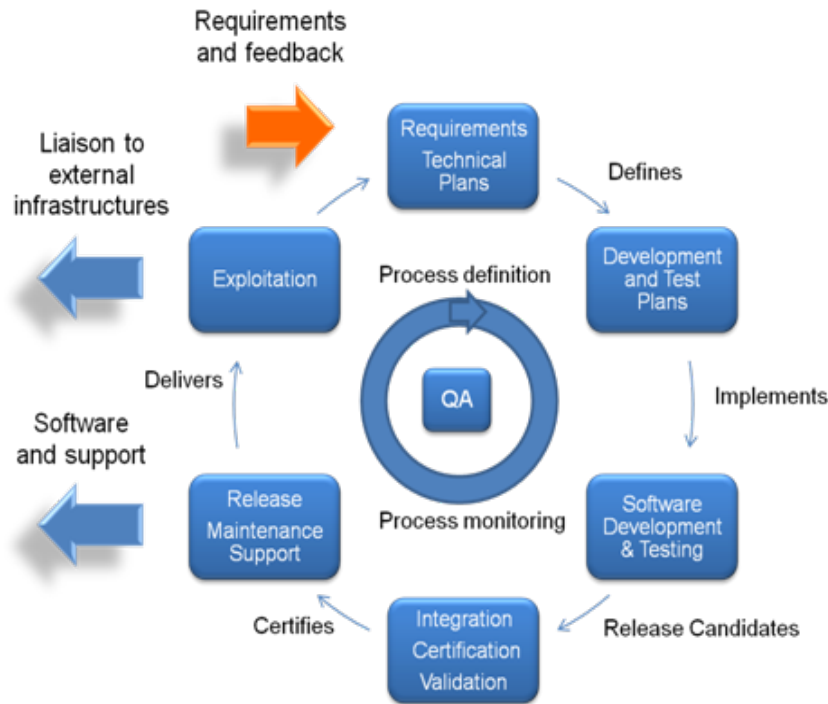[14]https://github.com/indigo-dc/wattson

Figure 8: Software lifecycle implementation

release quality is attained the software is made available to the general public through INDIGO-DataCloud repositories. INDIGO then provides support for the released software and manage bug fixes in close coordination with the developers.

5.1 Software lifecycle

Software quality assurance is performed mostly via automated actions. These actions are orchestrated by a Continuous Integration (CI) tool which checks out the source code from the source code repositories, perform any building steps and execute any required tests, and verifications. The actions themselves are implemented by the developers together with the quality assurance and release teams, and performs code style checking, unit testing, functional testing and, whenever possible, integration testing. Some manual steps, e.g. for documentation verification and code review, are also required.

In Figure 9 we depict the software lifecycle, release, maintenance andexploitation interdependencies, in this complex and diversified environment. Source code is made available through public repositories, housed externally (GitHub) increasing so the visibility and simplifying the path to exploitation beyond the project lifetime.
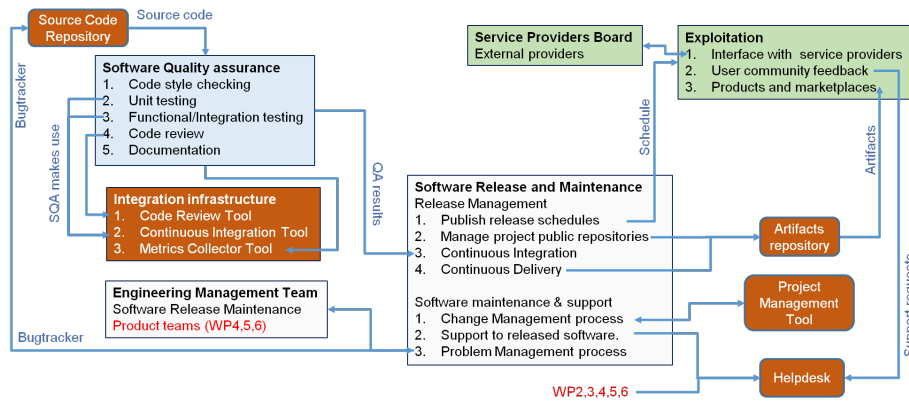
17

Figure 9: Software lifecycle, release, maintenance andexploitation interdependencies.

The INDIGO-DataCloud software is released under the Apache 2.0 software license and can be deployed on both public and private Cloud infrastructures. Release notes, installation and configuration guides,user and development manuals are made available on **GitBook**[15]. The software can be downloaded from our public repository[16].

A set of tools and services are needed to support the PTs, the Software Quality Assurance, the Continuous Integration and the software release and maintenance. See Figure 10.

- Project management service using **openproject.org**: It provides tools such as an issue tracker, wiki, a placeholder for documents and a project management timeline.

- Version Control System using **GitHub**: Service for configuration management and revision control for software source code.

- Continuous Integration service using **Jenkins**: Service to automate the building, packaging (where applicable) and execution of unit and functional tests of software components. Automatic source code style checking is also foreseen.

- Artifacts repositories for RPM and Debian packages, and Docker Hub for containers: In INDIGO-DataCloud there are two types of artifacts, packaged software and virtual images.

- Code review service using GitHub: Code review of software source code is one integral part of the SQA. This service facilitates the code review process. It records the comments and allows the reviewer to verify the software modification.

---

[15]https://indigo-dc.gitbooks.io/indigo-datacloud-releases
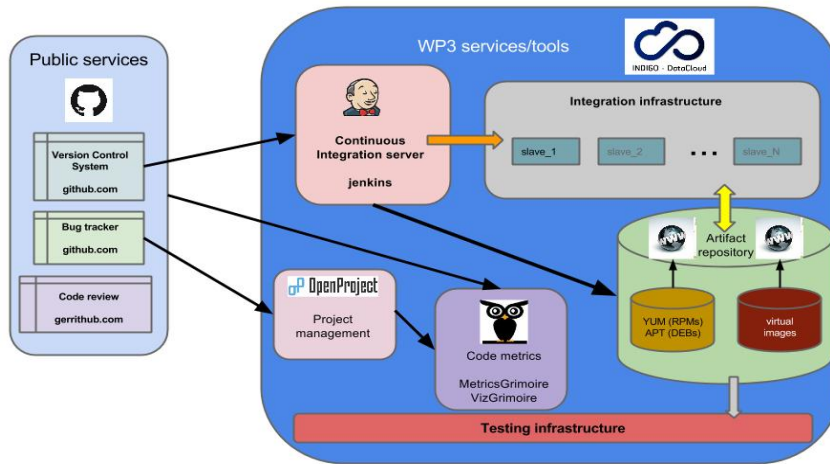[16]http://repo.indigo-datacloud.eu

Figure 10: Tools and services used to support the software development Product Teams.

- Code metrics services using **Grimoire**: To collect and visualize several metrics about the software components.

- Bug trackers using GitHub issues tracker: Services to track issues and bugs of INDIGO-DataCloud software components as well as services provided by WP3.

- Integration infrastructure: this infrastructure is composed of computing resources to support directly the Continuous Integration service. It's the place where building and packaging of software occurs as well as the execution of unit and functional tests. These resources are provided by INDIGO partners.

- Testing infrastructure: this infrastructure aims to provide several types of environment. A stable environment for users where they can preview the software and services developed by INDIGO-DataCloud, prior to its public release.

- Preview infrastructure: where the released artifacts are deployed and made available for testing and validation by the use-cases.

5.2   Services for continuous integration and SQA

The INDIGO-DataCloud SQA process is schematically shown in Figure 11. The process, in its different steps, reflects some of the main and important achievements of the software integration team, such as:
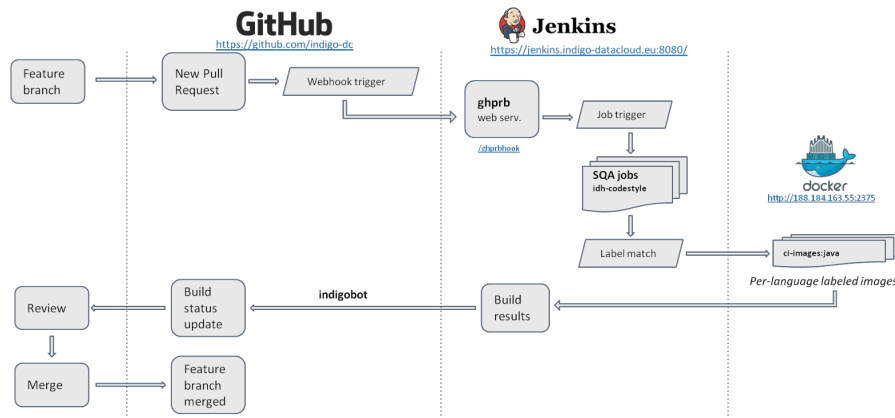
Figure 11: Software Quality Assurance process

- The pull request of a feature branch in GitHub triggers the execution of SQA jobs in the INDIGO-DataCloud Jenkins continuous integration system.

- The SQA jobs perform the code style verification and calculate unit and functional test coverage.

  - The tools necessary for tackling these tests are packaged in Docker images, available in DockerHub1.
  - Each test then initiates a new container that provides a clean environment for its execution.
  - This is an innovative approach that provides the flexibility needed to cope with the INDIGO-DataCloud software diversity.

- The results of the several SQA jobs are made available in the Jenkins service which notifies back to GitHub their exit status.

  - Only if the tests have succeeded, the source code is validated and is ready to be merged into the production branch.

- The last step in the workflow is the code review, where a human review of the change is performed. After code review the source code can be merged and becomes ready for integration and later release.

INDIGO-DataCloud provides a full set of development services for continuous integration and software quality assurance (version control system, code review, issue tracker, containers repository, continuous integration, artefacts repository, code metrics services, documentation repository). As a general rule, these services must be used by the all software products. However there are exceptions to this rule that fall into two main categories:

- mature products that have well established development services already in place;

- new software products that aim to contribute to existing projects/frameworks (such as OpenStack, OpenNebula and others) and that make use of dedicated development services made available by the corresponding projects.

For these cases where products are using external development services, the product teams must account that:

- continuous integration testing must satisfy the INDIGO-DataCloud SQA requirements;

- the product teams must provide metrics to WP3 whenever requested.

The INDIGO-DataCloud SQA team keeps track of the exceptions and continuous integration systems being used by the several products. This is a dynamic process. As products evolve and become accepted in their target projects they may move from the INDIGO-DataCloud development system to external development systems.

This is indeed one of the objectives of the SQA in INDIGO-DataCloud, improve the quality of the software, so that it can be more easily accepted in target projects ensuring a path for sustainability and exploitation.

5.3 DevOps approach in INDIGO

Progressive levels of automation were being adopted throughout the different phases of the INDIGO-DataCloud project software development and delivery processes. This evolution was intentionally marked by the commitment to DevOps principles. Starting with a continuous integration approach, achieved already in the early stages of the project, the second part of the project was devoted to the establishment of the next natural step in the DevOps practices: the continuous delivery (CD).

Continuous delivery adds, on top of the software development chain, a seamless manufacturing of software packages ready to be deployed into production services. Therefore, fast, frequent and small releases can be taken over thus promoting the reliability of the software.

In the INDIGO-DataCloud scenario, the continuous delivery adoption translates into the definition of pipelines. A pipeline is a serial execution of tasks that encompasses in the first place the SQA jobs (CI phase) and adds as the second part (CD phase) the building and deployment testing of the software packages created. The pipeline only succeeds if each task is run to completion, otherwise the process is stopped and set as a build failure.

As a result, in one pipeline execution, the source code is validated and packaged automatically, with the only manual intervention of approving and marking the resulting packages as production-ready. As shown in Figure 12, these software pipelines where defined in the Jenkins CI service and differ in the number
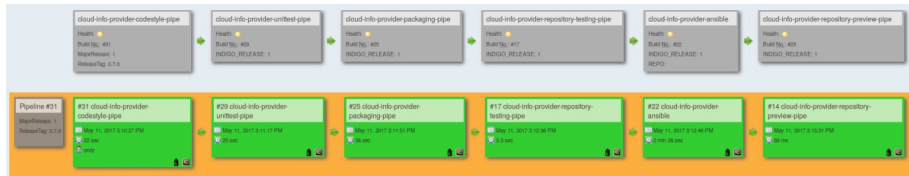
Figure 12: DevOps pipeline to distribute Ubuntu Xenial and CentOS7 packages for cloud-info-provider product

of tasks depending on the packaging type being produced: tarballs, RPM/DEB packages and/or Docker images.

5.3.1   DevOps adoption from user communities

The experience gathered throughout the project with regards to the adoption of different DevOps practices is not only useful and suitable for the software related to the core services in the INDIGO-DataCloud solution, but also applicable to the development and distribution of the applications coming from the user communities.

As an example, two applications DisVis and PowerFit, were integrated into a similar CI/CD pipeline described above. As it can be seen in the Figure 12, with this pipeline in place the application developers were provided with both a means to validate the source code before merging and the creation of a new versioned Docker image, automatically available in the INDIGO-DataCloud's catalogue for applications i.e. DockerHub's `indigodatacloudapps` repository.

The novelty introduced in the pipeline above is the validation of the user application. Once the application is deployed as a Docker container, and subsequently uploaded to `indigodatacloudapps` repository, it is instantiated in a new container to be validated. The application is then executed and the results compared with a set of reference outputs. Thus this pipeline implementation goes a step forward by testing the application execution for the last available Docker image in the catalogue.

5.4   INDIGO-1 release (`MidnightBlue`)

The first INDIGO-DataCloud major release - INDIGO-1 (codename `MidnightBlue`) was released 1st of August 2016. The public availability was announced through:

- General announcements
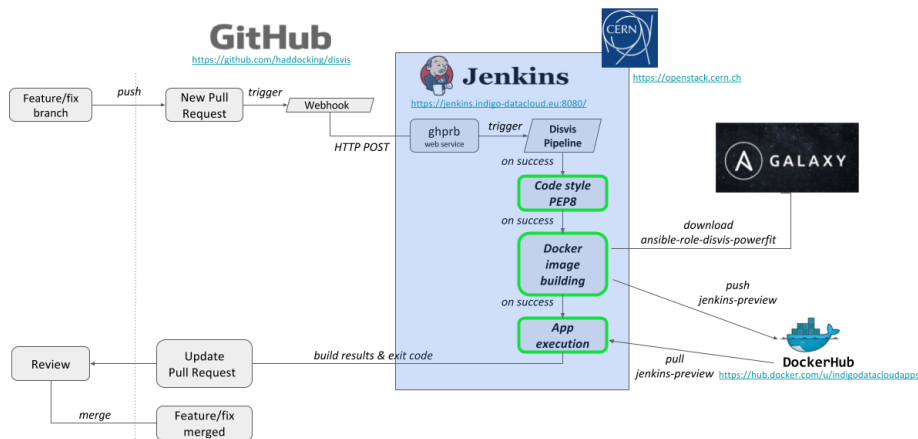
- On the project News:

  https://www.indigo-datacloud.eu/news/first-indigo-datacloud-software-release-out

- INDIGO-DataCloud RSS Feed:
  http://repo.indigo-datacloud.eu/INDIGODataCloudNews.rss.xml

Figure 13: DisVis development workflow using a CI/CD approach

Release version	Indigo-1
Release name	**MidnightBlue**
Release date	8th August 2016
Release repository	http://repo.indigo-datacloud.eu
Release documentation	https://www.gitbook.com/book/indigo-dc/indigo-datacloud-releases/details
Software license	Apache 2.0
Supported Operating Systems	Linux (CentOS 7 and Ubuntu 14.04)
Supported cloud frameworks	OpenStack Liberty and OpenNebula 4.14
Packages	208 RPMs, DEBs and tarballs 40 Docker containers
Release announcements	http://repo.indigo-datacloud.eu/INDIGONews.rss.xml e-mail: indigo-announce@lists.indigo-datacloud.eu

Figure 14: INDIGO-1 release fact sheet

- indigo-all mailing-list:
  https://lists.indigo-datacloud.eu/sympa/arc/indigo-all/2016-08/msg00000.html

In table 14 we show the INDIGO-1 fact sheet.

5.4.1 Contributions to Upstream Distribution

Many of the INDIGO-DataCloud solutions released through INDIGO-1 were also contributed to the respective upstream distributions like:

- OpenStack
  - Changes/contribution done already merged upstream
    * Nova Docker

23

- ∗ Heat Translator (INDIGO-DataCloud is 3rd overall contributor and core developer)
- ∗ TOSCA parser (INDIGO-DataCloud is 2nd overall contributor and core developer)
- ∗ OpenID Connect CLI support
- ∗ OOI: OCCI implementation for OpenStack
  - – Changes/contribution under discussion to be merged upstream OpenStack Preemptible Instances support (extensions)

- OpenNebula
  - – Changes/contribution done already merged upstream
    - ∗ ONEDock

- Changes/contribution done already merged upstream for:
  - – Infrastructure Manager
  - – CLUES
  - – OneData
  - – jSAGA Adaptors
  - – MitreID connect library

5.5 INDIGO-2 release (`ElectricIndigo`)

The second INDIGO-DataCloud major release - INDIGO-2 (codename `ElectricIndigo`) was made publicly available on April 14th 2017. The release was announced through:

- General announcements
  - – On the project news:
    https://www.indigo-datacloud.eu/news/electricindigo-second-indigo-datacloud-software-release

- INDIGO-DataCloud RSS Feed:
  http://repo.indigo-datacloud.eu/INDIGODataCloudNews.rss.xml

In table 15 we show the INDIGO-2 release fact sheet.

5.6 Software Stage Rollout

The INDIGO-DataCloud Staged Rollout is a procedure based on EGI[19] Staged Rollout whereby releases of the supported middleware are first deployed and tested by Service Providers who have agreed to be Early Adopter before being widely disseminated to all Service Providers.

This procedure permits testing the artifacts in a production environment and exposing the software to real endusers.

Release version	Indigo-2
Release name	**ElectricIndigo**
Release date	14th April 2017
Release repository	http://repo.indigo-datacloud.eu
Release documentation	https://www.gitbook.com/book/indigo-dc/indigo-datacloud-releases/details
Software license	Apache 2.0
Supported Operating Systems	Linux (CentOS 7 and Ubuntu 16.04)
Supported cloud frameworks	OpenStack Newton and OpenNebula 5.2170
Packages	413 RPMs, DEBs and tarballs 40 Docker containers
Release announcements	http://repo.indigo-datacloud.eu/INDIGONews.rss.xml e-mail: indigo-announce@lists.indigo-datacloud.eu

Figure 15: INDIGO-2 release fact sheet

This environment is also more heterogeneous than what is possible during the certification and verification phases. The procedure allows for potential issues to be discovered and workarounds to be swiftly added.

The Staged rollout takes place after the products are made available in the INDIGO-DataCloud repositories and constitute a last step of verification before the products are incorporated in other distributions such as EGI CMS and UMD. Users and site administrators unrelated to INDIGO-DataCloud can participate in the process.

5.7   Software maintenance strategy and updates

The software maintenance activity focus on the timely release of corrective (bug fixes) and adaptive (new/versions of operating systems and/or cloud management frameworks) changes as well as perfective and preventive ones, following the defined procedures.

The software support activities concentrate on following closely the different user communities and e-infrastructure sites that use and deploy projects solutions, ensuring that all their requests for support, for changes/improvements are addressed and correctly redirected to the right development teams, in case proven to be bugs in the released software.

Let us see with an example how the maintenance process of INDIGO-1 has smoothly worked out towards the release of INDIGO-2. Updates continued to be provided on a monthly manner taking in consideration that:

- After the release of the first INDIGO major release, MidgnightBlue, on August 2016, from the point of view of the schedule the activity entered in the Full Support period, period in which the developers were supposed to fix all the notified issues, according to their priority, and also to add missing or new features and improvements.

- The full support period ended February 2017, so that no new features were added to INDIGO-1, but moved, eventually to INDIGO-2 release. During
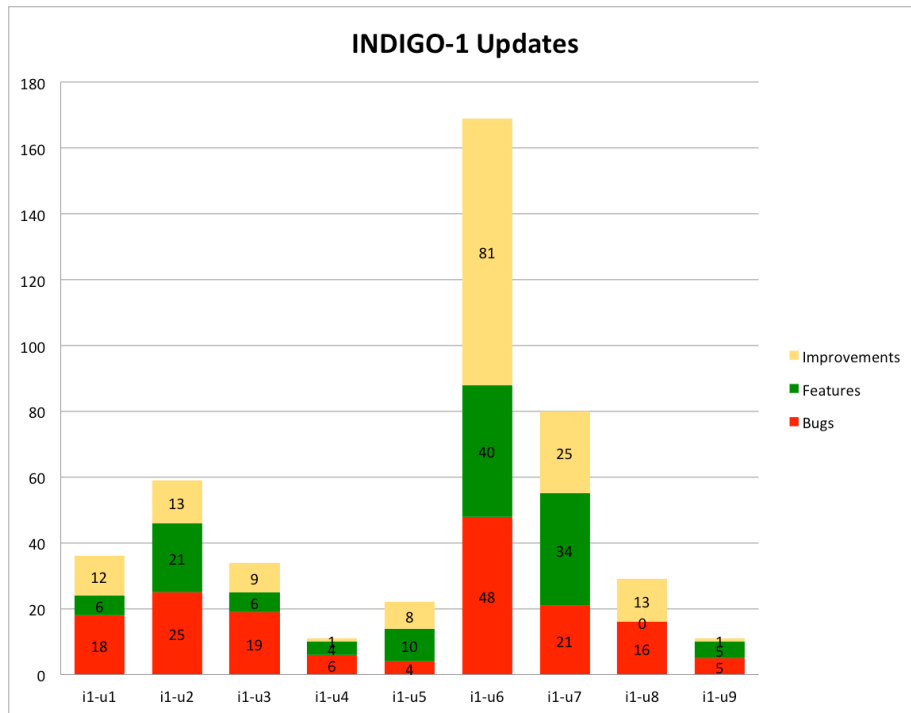
Figure 16: Updates and maintenance of INDIGO-1 (`MidnightBlue`)

this Standard Support period the developers were due to release only bug fixes.

- The Standard Support period ended at the end of March 2017, in concomitance with the release of the projects' second major release. No more changes, bug fixes were introduced. Just support for fixing security vulnerabilities for 2 more months, until the end of May 2017.

- After May 2017 - INDIGO-1 repositories are closed, and the only supported components will be the ones present in the INDIGO-2 release. And so the MidnightBlue release reached the EndOfLife.

During the full support period PTs managed to fix ~ 98 issues and to add ~ 226 new features & improvements to the released services & components, as detailed in Figures 16 and 17.

The above statistics are based on the information provided by the Product Teams in the release notes of each updated service or component. To conclude, we present in Table 18 the updated timelines of the INDIGO - Datacloud Maintenance and Support activities.
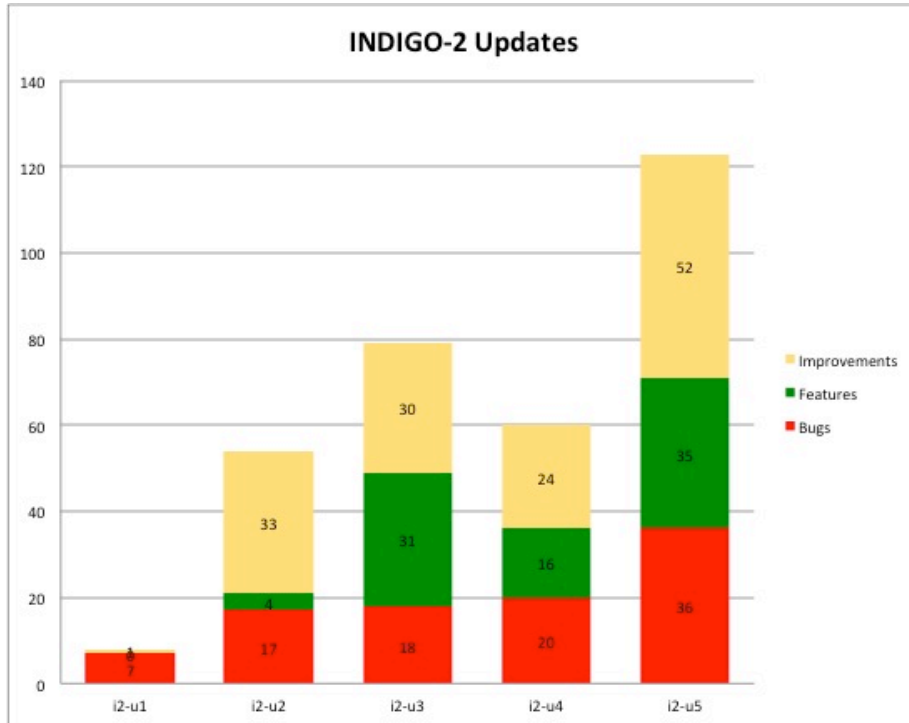
Figure 17: Updates and maintenance of INDIGO-2 (`ElectricIndigo`)

		Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec	Jan
INDIGO-1	Full updates																		
	Standard updates																		
	Security updates																		
INDIGO-2	Full updates																		
	Standard updates																		
	Security updates																		

	Release Date	End of Full Updates	End of Standard Updates	End of Security Updates & EOL
INDIGO-1 MidnightBlue	08/08/2016	31/01/2017	31/03/2017	**31/05/2017**
INDIGO-2 EletricIndigo	14/04/2017	30/09/2017	*30/11/2017*	*31/01/2018*

Figure 18: Support timeline for the INDIGO releases

6 Examples of implementation of results towards research communities

In what follows we try to provide some basic information that may be useful for promoting the use of INDIGO solutions towards the Research Communities.

Based on the described architecture we will introduce the basic ideas on how to develop, deploy and support applications in the Cloud framework, exploiting the different service layers, and introducing generic examples that may make easier the use of INDIGO solutions.

6.1 Understanding services in the Cloud Computing framework

Figure 19 provides the description of how an application can be built using a service oriented architecture in the Cloud, using INDIGO solutions.

This layered scheme includes different elements, that are managed by different actors, that must be minimally understood in order to design, develop, test, deploy and put in production an application.

The lowest layer, Infrastructure as a Service, provides a way to access to the basic resources that the application will use, like computing, storage, network, etc. These resources are physically in a "site", typically a computing center, either in a research centre, or in a "cloud provider" (like for example commercial cloud services), and are handled by the system managers at those sites, that install a IaaS solution compatible with the INDIGO software stack (e.g. OpenStack, OpenNebula, Google Compute Engine, Microsoft Azure, Amazon EC2).

Accessing through a web interface to a proprietary dashboard, like for example Horizon for OpenStack, a "user" that is granted access to a pool of resources at a site (for example 150 cores in "virtual" machines with 600GB RAM, 3 TB of storage and 1GB network interface) can "launch" a virtual machine, for example a server with 2 cores, 4GB RAM, 100GB of storage and with a certain Linux flavour installed.

The user can then get access in "console" mode to this machine, for example through a protocol like ssh, and employ it for the tasks required. For example, he can execute a simple script, or can install a web server, etc. When the work is finished the machine can be stopped by the user, liberating so the resources. This is a very basic mode of accessing Cloud services, which shares analogies with the usual access to classical computing services, like for example any remote server or a cluster.

A different way to interact with IaaS services is to use the existing APIs to manage the resources using the web services protocol. Such invocation of services can be made from any program or application, from example from a python script, and even through a web interface.

Many applications require the setup, launching and interconnection of several (IaaS) services implemented in different virtual machines, and managed
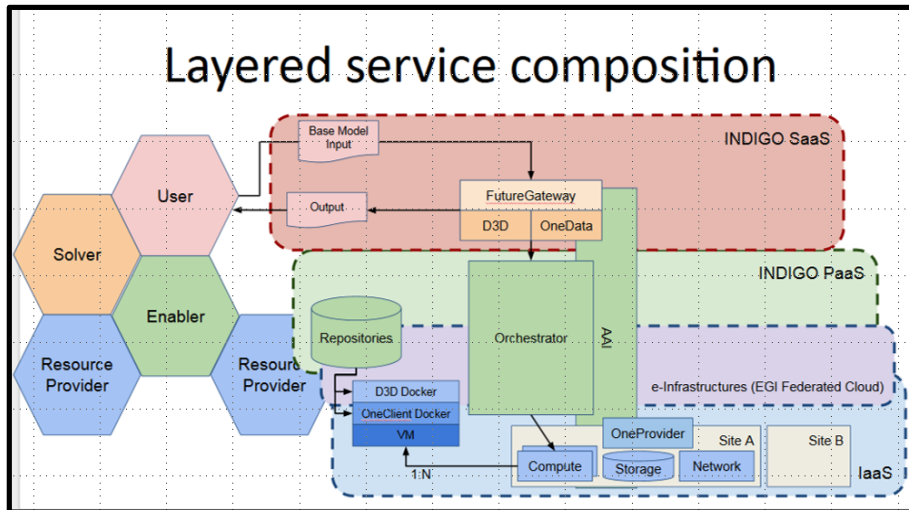
28

Figure 19: Service composition in the Service Oriented Architecture

under a single control, as a "Platform". The Platform as a Service (PaaS) layer enables this "orchestration" of IaaS services, and in the case of a "Federated Cloud" they might even be located in different sites.

For example, an Apache Web may be launched in site 1, and display the output of a simulation running in a cluster launched on demand in site 2. The Apache Web service may be better supported with a pool of resources using another cloud-oriented solution like Marathon/Mesos. Launching an application in this context is a non-trivial process, that requires identifying the available resources and launching them via the IaaS services. INDIGO is supporting the TOSCA standard to prepare a template that can be used to automatize this selection and orchestration of services.

However, most researchers are mainly interested in "running" a final application, either accessing it as an existing service, or eventually requesting resources and launching the application themselves, to use it afterwards. This possibility is enabled via the Software as a Service (SaaS) layer.

Before continuing to provide different examples and templates based on the previous discussion, we need a few words on the organization from the point of view of the research community.

6.2   Building basic applications using INDIGO solutions

In what follows we present below several simple examples of basic, but generic, applications exploiting INDIGO solutions.

- The first generic example is how to build an application encapsulated as a container able to run in an HPC system, using INDIGO solution udocker

29

udocker is a tool to run containers in user space, without Docker and without privileges, without sysadmin assistance. In particular, udocker empowers users to run applications encapsulated in Docker containers, but it can be used to run any container that does not require admin privileges. Examples and documentation can be found at https://github.com/indigo-dc/udocker.

udocker is also a very good solution for research communities that want to migrate towards a cloud-based framework using containers, but keep exploiting resources like grid-enabled clusters or even supercomputers. The support to run parallel applications over Infiniband has been completed, opening a very interesting path for further exploitation. udocker is used by the Case Studies on Structural Biology (Powerfit and Disvis) exploiting grid resources, on Phenomenology in Particle Physics, and recently for Lattice QCD on supercomputers. Also, the TRUFA genomic pipeline exploits this solution, and it is being extended to similar applications in the area that require the integration of legacy libraries. The use of udocker does not require a special effort on adaptation.

- The second example is how to build an application encapsulated as a container and launch it in the Cloud, from a web interface, using the INDIGO solutions FutureGateway and PaaS Orchestrator

This second example, compared to the first one, shows the evolution required to move an application to the Cloud arena: the application must be encapsulated into a container, as before, but to launch this container the cloud resources must be allocated, the user must authenticate and get the access granted. If different services are required, they must be orchestrated.

The way to express these requirements, using an open standard, is a TOSCA template. FutureGateway offers a user-friendly web-based interface to customize the TOSCA template, authenticate the user, select the container to be executed, interact with the Orchestrator to allocate the required cloud resources and launch the application.

This very basic example of usage of INDIGO solutions is shown in Figure 20. A user is able to create, using a Dockerfile a description of the application;

The building the container can be fully automatized using GitHub and Docker Hub in such a way that a change in the Dockerfile, inmediately triggers a rebuilt of the application container.

Using the Future Gateway portal or the command line tool Orchent, the user can submit a TOSCA template to the Orchestrator, which in turn will request and allocate the resources at the IaaS level by asking the Infrastructure Manager to do so.

The user may wish to connect to the container that has been launched via the orchestrator using the `ssh` command. Once in the container it is
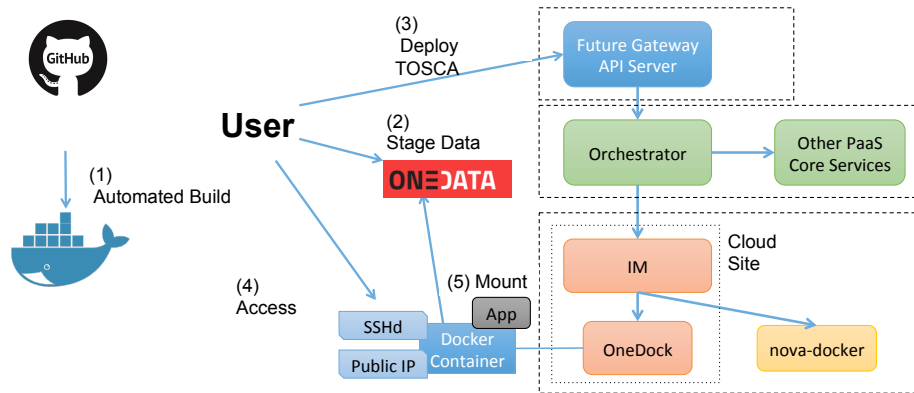
Figure 20: Basic execution of containers using INDIGO solutions

possible to mount remote data repositories or stage the output data using ONEDATA, available at the IaaS layer.

6.3 Building advanced applications using INDIGO solutions

In this section, we present several simplified schemes, corresponding to different applications already implemented, with the idea that they can be more easily used as a guide to configure new applications.

The key, as stated before, is the composition of the template, written using the TOSCA language, specifying:

- the image of the application to be used, as a container, using docker technology.

- the resources (CPUs, storage, memory, network ports) required to support the execution

- the parameters required to configure INDIGO services used like, for example, OneData

- idem for additional cloud services (like Mesos/Marathon)

Examples of TOSCA templates can be found at https://github.com/indigo-dc/tosca-templates FutureGateway offers a friendly way to handle the TOSCA templates to launch the applications.

- A first example is the deployment, as a SaaS solution, of a digital repository. The scheme is presented in the Figure 21 below. The specific template for this application is available for reuse in the github repository of the project.
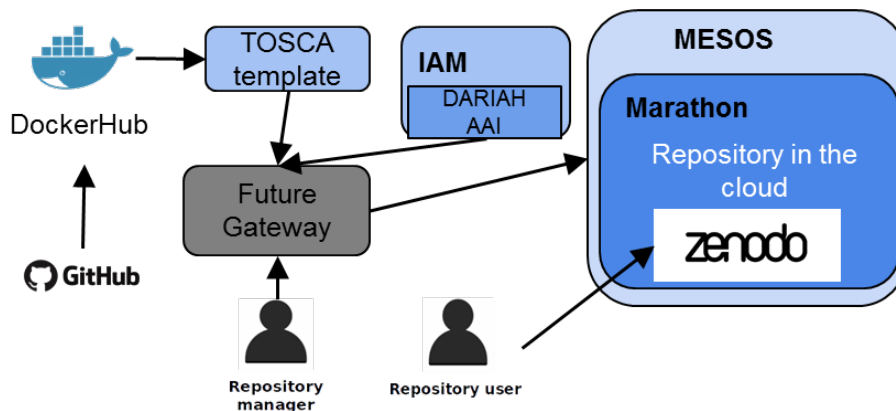
Figure 21: Deployment of a digital repository using INDIGO solutions

The repository manager, controlling the application, uses the FutureGateway to configure the application, based on the ZENODO software, that can be automatically scaled up and ensure its high availability using Cloud resources as needed, and also enabling the authentication and authorization mechanism for their research community, DARIAH, based on the INDIGO solution IAM.

All these details are transparent to the final user, who accesses the repository directly through its web interface, and benefits of the enhanced scalability and availability.

- A second example is the launch of a Virtual Elastic Cluster to support a data intensive system.The scheme is presented in Figure 22 below.

  The specific template for this advanced application is available for reuse in the github repository of the project.

  Galaxy is an open source, web-based platform for data intensive biomedical research. This application deploys a Galaxy instance provider platform, allowing to fully customize each virtual instance through a user-friendly web interface, ready to be used by life scientists and bioinformaticians.

  The front-end that will be in charge of managing the cluster elasticity can use a specified LRMS (selected among torque, sge, slurm and condor) workload.

  All these details are transparent to the final user, the researcher, who accesses the Galaxy instance directly through its web interface, and benefits of the enhanced scalability and availability.

  This complex template includes the configuration of the distributed storage based in OneData, the use of the encrypted files via LUKS, the deployment of elastic clusters using another INDIGO solution, CLUES, and
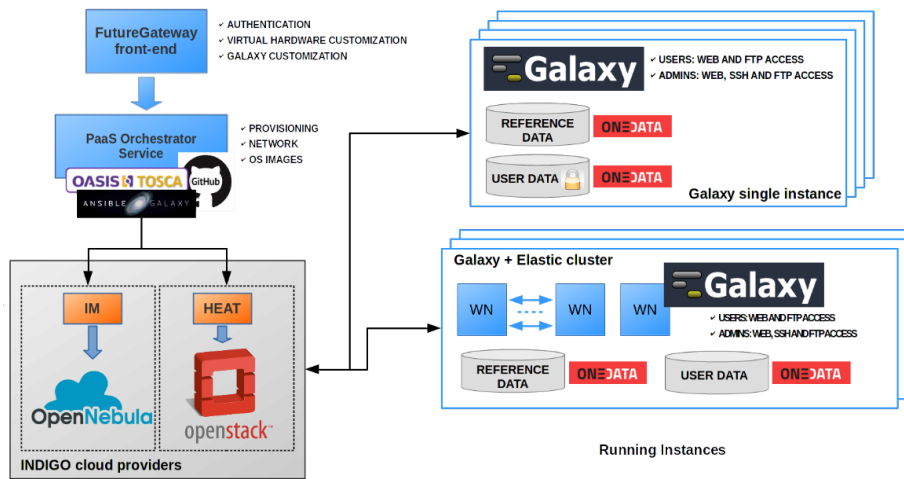
Figure 22: Launching of a Virtual Elastic Cluster using INDIGO solutions

the integration of the Authentication and Authorization mechanism, very relevant for this application area, using IAM.

7   Conclusions

Thanks to the new common solutions developed by the INDIGO project, teams of first-line researchers in Europe are using public and private Cloud resources to get new results in Physics, Biology, Astronomy, Medicine, Humanities and other disciplines.

INDIGO-developed solutions that have for instance enabled new advances in understanding how the basic blocks of matter (quarks) interact, using supercomputers, how new molecules involved in life work, using GPUs, or how complex new repositories to preserve and consult digital heritage can be easily built. The variety of the requirements coming from these so diverse user communities proves that the modular INDIGO platform, consisting of several state-of-the-art, production-level services, is flexible and general enough to be applied to all of them with the same ease of use and efficiency.

These services allow to federate hybrid resources, to easily write, port and run scientific applications to the cloud. They are all freely downloadable as open source components, and are already being integrated into many scientific applications, namely:

- High-energy physics: the creation of complex clusters deployed on several Cloud infrastructures is automated, in order to perform simulation and analysis of physics data for large experiments.

- Lifewatch: parameters from a water quality model in a reservoir are cali-

33

brated, using automated multiple simulations.

- Digital libraries: multiple libraries can easily access a cloud environment under central coordination but uploading and managing their own collections of digital objects. This allows them to consistently keep control of their collections and to certify their quality.

- Elixir: Galaxy, a tool often used in many life science research environments, is automatically configured, deployed on the Cloud and used to process data through a user-friendly interface.

- Theoretical HEP physics: the MasterCode software, used in theoretical physics, adopts INDIGO tools to run applications on Grids, Clouds and on HPC systems with aneffcient, simple-to-use, consistent interface.

- In DARIAH, a pan-european social and technical infrastructure for arts and humanities, the deployment of a self-managed, auto-scalable Zenodo-based repository in the cloud is automated.

- Climate change: distributed, parallel data analysis in the context of the Earth System Grid Federation (ESGF) infrastructure is performed through software deployed on HPC and cloud environments in Europe and in the US.

- Image analysis: in the context of EuroBioImaging, a distributed infrastructure for microscopy, molecular and medical imaging, INDIGO components are used to perform automatic and scalable analysis of bone density.

- Astronomical data archives: big data consisting of images collected by telescopes are automatically distributed and accessed via INDIGO tools.

The same solutions are also being explored by industry, to provide innovative services to EU companies: for example, modelling water reservoirs integrating satellite information, improving security in cyberspace, or assisting doctors in diagnostics through medical images.

The outcomes of INDIGO-DataCloud will persist, and also be extended, after the end of the project because. INDIGO was one of the three coordinating projects (together with EGI and EUDAT) in the preparation of the EOSC-hub proposal submitted to the EC H2020 EINFRA-12 call; the main goal of this recently approved project is to contribute to the EOSC implementation by enabling seamless and open access to a system of research data and services provided across nations and multiple disciplines.

Many INDIGO components will find place in the unified service catalogue provided by EOSC-hub. In addition INDIGO nominates the Technology Coordinator of the entire project, in recognition of the impressive work we have been doing in these 30 months.

Two additional Horizon 2020 projects were also approved (`DEEP HybridDataCloud` and `eXtreme DataCloud`), that will continue to develop and enhance INDIGO components to provide new and exciting innovative services.

INDIGO solutions are also being intensively tested in other projects, such as HelixNebula ScienceCloud. To conclude, the 30 months of the INDIGO-DataCloud project were totally exciting and ripe with results, well beyond what was originally foreseen. We believe that the foundations laid by INDIGO will continue to find proper development and adoption in a wide variety of fields, public and private, at the service of science and for the benefit of the overall general public.

Acknowledgments

References

[1] See https://ec.europa.eu/research/openscience

[2] See http://www.infoq.com/articles/scaling-docker-with-kubernetes

[3] For a definition of the standard see: http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html

[4] See http://opennebula.org/

[5] See https://www.openstack.org/

[6] S. Hykes (Docker Inc) See software description and downloads in http://www.docker.com

[7] See https://proot-me.github.io/

[8] See https://github.com/opencontainers/runc

[9] See https://github.com/dex4er/fakechroot

[10] K. J. de Vries, PhD thesis *Global Fits of Supersymmetric Models after LHC Run 1* (2015), available on the Imperial College website: `http://hdl.handle.net/10044/1/27 056`.

[11] Description of the cloud-based Docker repository service can be found in: https: //www.docker.com/products/docker-hub

[12] PRISMA: http://www.ponsmartcities-prisma.it/

[13] Opencityplatform: http://www.opencityplatform.eu

[14] See https://www.openshift.com/

[15] See https://www.cloudfoundry.org/

[16] See https://onedata.org/

[17] See http://lcgdm.web.cern.ch/dynafed-dynamic-federation-project

[18] See https://svnweb.cern.ch/trac/fts3

[19] For more details on the European Grid Infrastructure and software context around it see http://www.egi.eu