Efficient management of INDIGO-IAM clients and S3 buckets via INDIGO PaaS Orchestrator in INFN Cloud

Luca Giommi,[a,*] **Marica Antonacci,**[b] **Enrico Vianello,**[a] **Roberta Miccoli,**[a] **Federica Agostini,**[a] **Federico Fornari,**[a] **Gioacchino Vino,**[b] **Giovanni Savarese,**[b] **Giacinto Donvito**[b] **and Alessandro Costantini**[a]

[a]*INFN CNAF,*
Viale Berti Pichat 6/2, Bologna, Italy
[b]*INFN Sezione di Bari,*
Via Giovanni Amendola 173, Bari, Italy
E-mail: luca.giommi@cnaf.infn.it, marica.antonacci@ba.infn.it, enrico.vianello@cnaf.infn.it, roberta.miccoli@cnaf.infn.it, federica.agostini@cnaf.infn.it, federico.fornari@cnaf.infn.it, gioacchino.vino@ba.infn.it, giovanni.savarese@ba.infn.it, giacinto.donvito@ba.infn.it, alessandro.costantini@cnaf.infn.it

The National Institute for Nuclear Physics (INFN) has been operating and supporting Italy's largest research and academic distributed infrastructure for several decades. In March 2021, INFN launched the "INFN Cloud" initiative which provides a federated cloud infrastructure and a customizable service portfolio tailored for the scientific communities supported by the institute. The federation middleware of the INFN Cloud platform is built upon the INDIGO PaaS Orchestration system, which consists of interconnected open-source microservices. Among them, there is the INDIGO PaaS Orchestrator which receives high-level deployment requests from users and coordinates the deployment process over IaaS platforms.

In this work, we address an issue within INFN Cloud concerning the proliferation of INDIGO Identity and Access Management (INDIGO-IAM) clients and S3 buckets. Specifically, these resources are created during the on-demand deployment of most high-level services with no control from the INDIGO PaaS Orchestrator. This results in a scenario where, upon deployment deletion or failure, the related resources are not removed. This leads to an increasing number of unused INDIGO-IAM clients and buckets, consequently causing a decrease in the performance of the services hosting such resources. Our proposed solution involves delegating the management of such resources to the INDIGO PaaS Orchestrator, offering the users enough flexibility without losing control over them.

International Symposium on Grids and Clouds (ISGC2024)
24 -29 March, 2024
Academia Sinica Computing Centre (ASGC), Institute of Physics, Academia Sinica Taipei, Taiwan

[*]Speaker

1. Introduction

The Italian National Institute for Nuclear Physics (Istituto Nazionale di Fisica Nucleare, INFN) is the coordinating institution for nuclear, particle, theoretical, and astroparticle physics in Italy. It promotes, coordinates, and carries out scientific research as well as the technological development necessary for the activities in these sectors. Moreover, INFN manages and supports the largest public computing infrastructure for scientific research spread throughout the country, serving more than 40 international scientific collaborations.

In March 2021, INFN launched the "INFN Cloud" initiative aimed at providing a federated cloud infrastructure and a customizable service portfolio tailored for the scientific communities supported by the institute. The portfolio assortment comprises standard Infrastructure as a Service (IaaS) options and more advanced Platform as a Service (PaaS) and Software as a Service (SaaS) solutions, all crafted to suit the distinct needs of specific communities. Built upon the INDIGO PaaS Orchestration system, the federation middleware of the INFN Cloud platform consists of interconnected open-source microservices aimed at processing deployment requests from users and coordinating the deployment process over the federated IaaS platforms.

Most of the services in the INFN Cloud portfolio require the creation of INDIGO-IAM clients [1] or S3 buckets. Currently, this creation is performed through Ansible recipes [2] and the information about such resources remains confined to the deployed service. This results in a scenario where, upon deployment deletion or failure, the related INDIGO-IAM clients or S3 buckets are not removed leading to an increasing amount of unused resources. In this paper, the solution and the related implementation to address this problem are presented and discussed. The paper is organized as follows: the INFN Cloud ecosystem is laid out in Section 2, the PaaS Orchestration layer is described in Section 3, the problem we face is described in Section 4, whereas the new approach adopted is presented in Section 5 and Section 6.

2. The INFN Cloud ecosystem

To optimize the use of available resources and expertise, INFN decided to implement a national Cloud infrastructure for research:

- as a federation of existing distributed infrastructures, extending them if necessary in a transparent way to private and commercial providers,
- as a "user-centric" infrastructure that makes available a dynamic set of services tailored to specific use cases to the final users,
- as a solution that leverages the outcomes of several national and European cloud projects in which INFN actively participated in the past, e.g. the INDIGO DataCloud project [3].

The infrastructure of INFN Cloud is based on a core backbone, connecting the large data centers of CNAF and Bari at high speed which hosts the INFN Cloud core services, and on a set of loosely coupled distributed and federated sites connected to the backbone, which currently are: Cloud-CNAF, Recas-Bari, Cloud-Veneto, Cloud-INFN-Catania, Cloud-Ibisco-Napoli. Currently, the backbone has around 2000 vCPUs, 15 TB of RAM, 600 TB of net storage, and 320 TB of object storage, whereas the federated clouds have around 3160 vCPUs, 75 TB of RAM, and 334 TB of net storage.

The portfolio of the offered services ranges from Software as a Service (SaaS) to Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) solutions, giving a different level of service management to users.

SaaS solutions, e.g. Notebook as a Service, are centralized services with no customization needed. PaaS solutions, e.g. creation of a Virtual Machine (VM) or Kubernetes [4] cluster, are on-demand services that users can configure, e.g. choosing the operating system and the flavor of the VM. IaaS solutions, e.g. start and stop of a VM, are operations at the infrastructure level that users can do. Users can access SaaS services and PaaS services managed by other system administrator users. System administrator users can instantiate and destroy PaaS services hosted on INFN Cloud, give other INFN Cloud users access to these services, keep deployments updated with the latest security patches, and access IaaS solutions. All the services are made available to users through a user-friendly interface, i.e. the INFN Cloud Dashboard [5], see Fig. 1. Moreover, client interfaces are offered for advanced users, including REST APIs, CLI, and Python bindings, which allow them to interact with the platform programmatically.

The services in INFN Cloud are described through an Infrastructure as Code paradigm, via a combination of:

- Topology and Orchestration Specification for Cloud Applications (TOSCA) templates [6], to model an application stack,
- Ansible roles [7], to manage the automated configuration of virtual environments,
- Docker containers [8], to encapsulate high-level application software and runtime,
- Helm charts [9], to manage the deployment of applications in Kubernetes clusters.

This approach allows for the reduction of manual processes and increases flexibility and portability across environments. The INFN Cloud service catalogue is a graphical representation of the TOSCA templates that have been developed by extending the INDIGO-DataCloud custom types. A Lego-like approach has been followed, building on top of reusable components and exploiting the TOSCA service composition pattern. The main objectives of this approach are to build added value services on top of IaaS and PaaS infrastructures and to lower the entry barrier for non-skilled scientists. An example of the combination of TOSCA template and type, Ansible role, and Docker containers used for the creation of the "Jupyter with persistence for Notebooks" service is shown in Figure 2.

3. The PaaS Orchestration layer

The federation middleware of the INFN Cloud platform is built upon the INDIGO PaaS Orchestration system, which consists of interconnected open-source microservices [11] (see Figure 3).

These microservices create an abstraction layer that allows access to compute and storage resources made available by various and heterogeneous providers. These providers can include

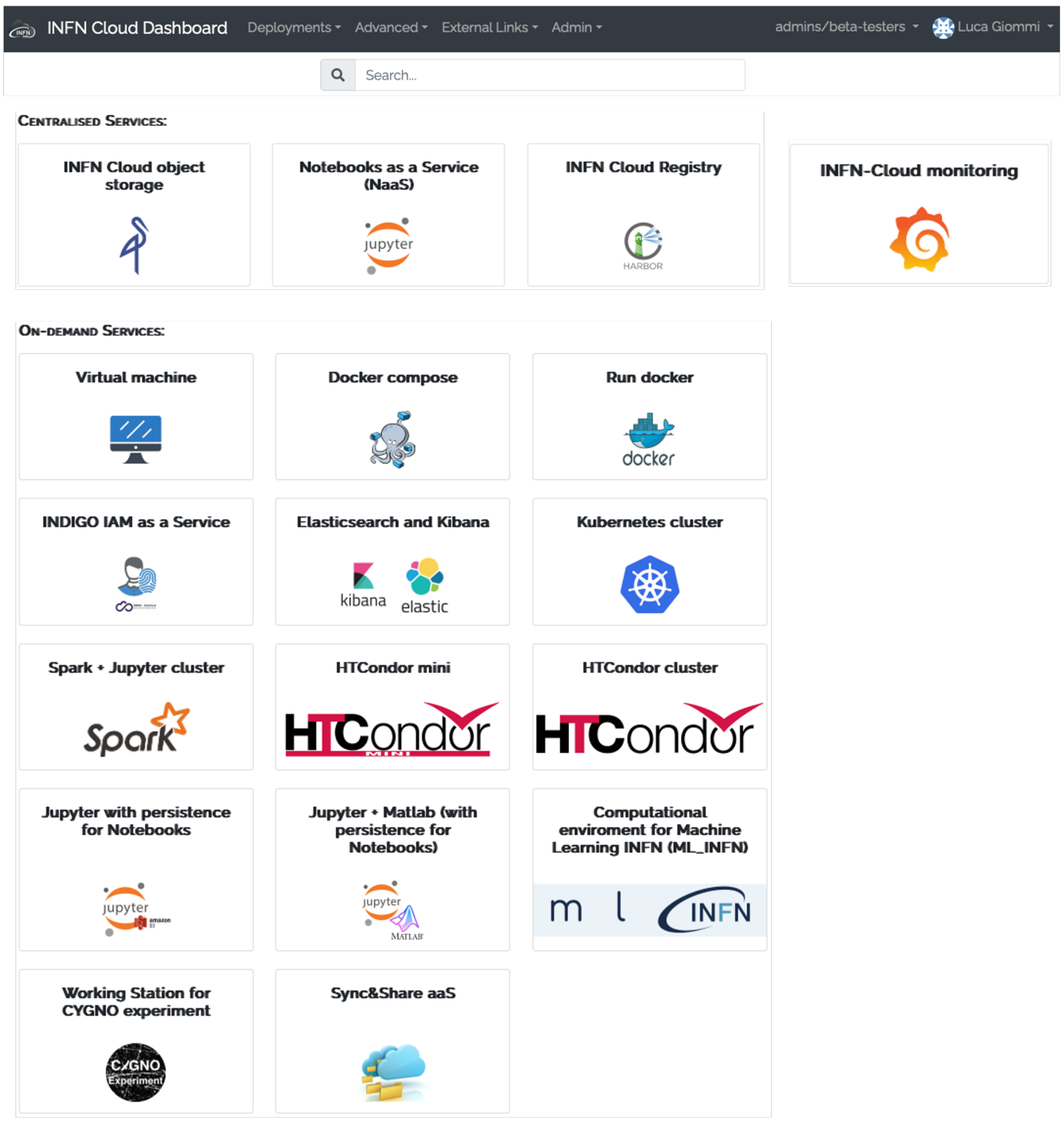
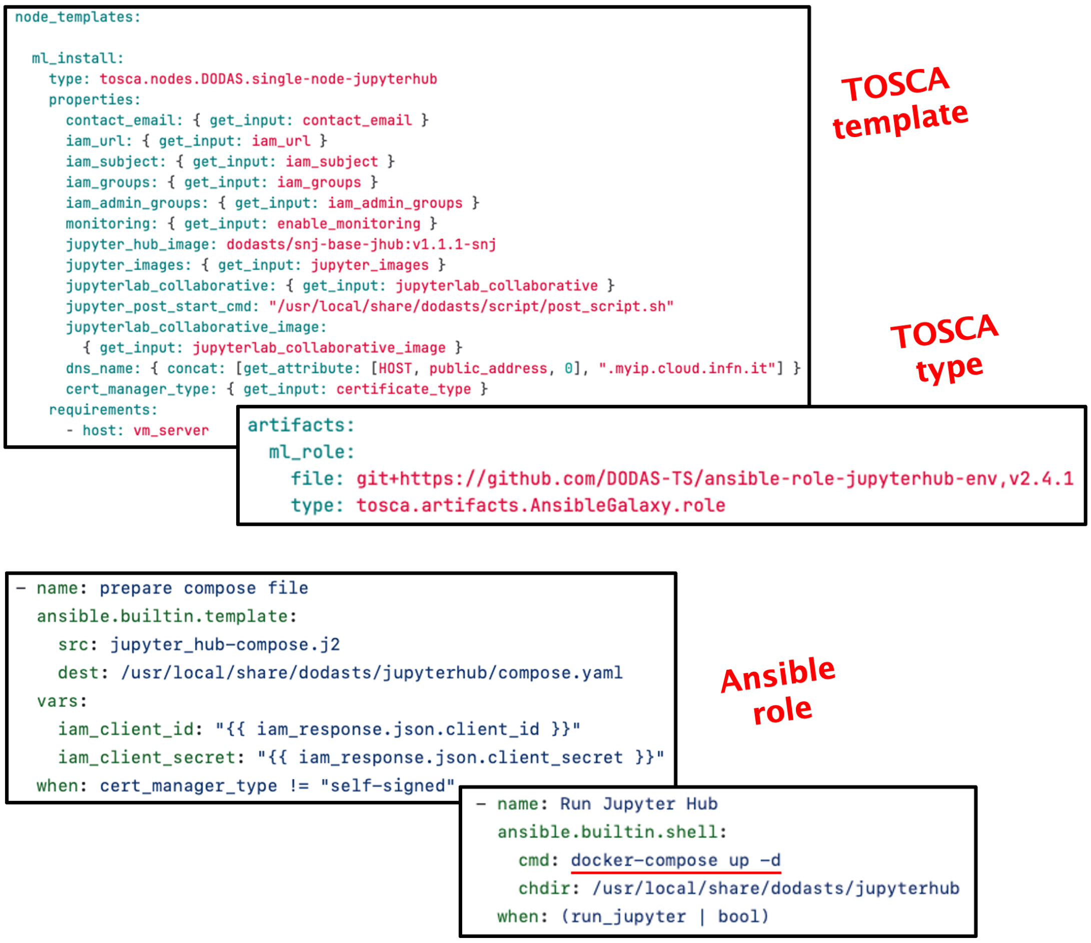


Figure 1: The INFN Cloud dashboard. It allows users to access centralized services, instantiate PaaS services independently, and access IaaS solutions. The PaaS services available in the dashboard depend on the user group.

public and private clouds (OpenStack, AWS, Azure, etc.), container orchestration platforms such as Kubernetes, and other types of infrastructures. Currently in INFN Cloud, all the federated providers make available an instance of OpenStack, while the backbone is a single OpenStack instance with two regions, i.e. CNAF and Bari.

The PaaS Orchestration system offers automatic selection of the best provider based on computing and storage needs versus the providers' capabilities. The core component of such a system is the INDIGO PaaS Orchestrator [12] which takes the information about the user's request in the form of a TOSCA template and interacts with the other microservices to get information about the available providers and their capabilities. These auxiliary services are the Service Level Agreement



```
node_templates:

  ml_install:
    type: tosca.nodes.DODAS.single-node-jupyterhub
    properties:
      contact_email: { get_input: contact_email }
      iam_url: { get_input: iam_url }
      iam_subject: { get_input: iam_subject }
      iam_groups: { get_input: iam_groups }
      iam_admin_groups: { get_input: iam_admin_groups }
      monitoring: { get_input: enable_monitoring }
      jupyter_hub_image: dodasts/snj-base-jhub:v1.1.1-snj
      jupyter_images: { get_input: jupyter_images }
      jupyterlab_collaborative: { get_input: jupyterlab_collaborative }
      jupyter_post_start_cmd: "/usr/local/share/dodasts/script/post_script.sh"
      jupyterlab_collaborative_image:
        { get_input: jupyterlab_collaborative_image }
      dns_name: { concat: [get_attribute: [HOST, public_address, 0], ".myip.cloud.infn.it"] }
      cert_manager_type: { get_input: certificate_type }
    requirements:
      - host: vm_server
```

```
artifacts:
  ml_role:
    file: git+https://github.com/DODAS-TS/ansible-role-jupyterhub-env,v2.4.1
    type: tosca.artifacts.AnsibleGalaxy.role
```

```
- name: prepare compose file
  ansible.builtin.template:
    src: jupyter_hub-compose.j2
    dest: /usr/local/share/dodasts/jupyterhub/compose.yaml
  vars:
    iam_client_id: "{{ iam_response.json.client_id }}"
    iam_client_secret: "{{ iam_response.json.client_secret }}"
  when: cert_manager_type != "self-signed"
```

```
- name: Run Jupyter Hub
  ansible.builtin.shell:
    cmd: docker-compose up -d
    chdir: /usr/local/share/dodasts/jupyterhub
  when: (run_jupyter | bool)
```

Figure 2: Combination of TOSCA template and type, Ansible role, and Docker containers (run using a Docker compose file created from a Jinja template [10] called *jupyter_hub-compose.j2*) used for the creation of Jupyter with persistence for Notebooks service.

Manager (SLAM), the Configuration Management DataBase (CMDB), and the Monitoring System (which is not currently available in INFN Cloud). Then the Cloud Provider Ranker (CPR) uses this information to generate a ranked list of candidate providers for the deployment. The best provider is placed at the top of the list, and the deployment request will initially be directed to it. Should there be a failure, a retry strategy is activated, and after that the deployment is rescheduled on the subsequent available provider in the list.

Another key component is the Authentication and Authorization Infrastructure service realised through the INDIGO Identity and Access Management (INDIGO-IAM) [1] tool which manages the authentication and authorization at all cloud levels of INFN Cloud (IaaS, PaaS, and SaaS). To access the Orchestrator APIs, a user must be authenticated and presented with an access JSON Web Token (JWT) [13] issued by the INDIGO-IAM service of INFN Cloud. Such a token has the *groups* claim with the list of INDIGO-IAM groups to which the user belongs. Each group represents a scientific community, a collaboration, or an experiment that can access a specific set of services and is mapped onto dedicated OpenStack projects of the federated clouds. Once the orchestrator

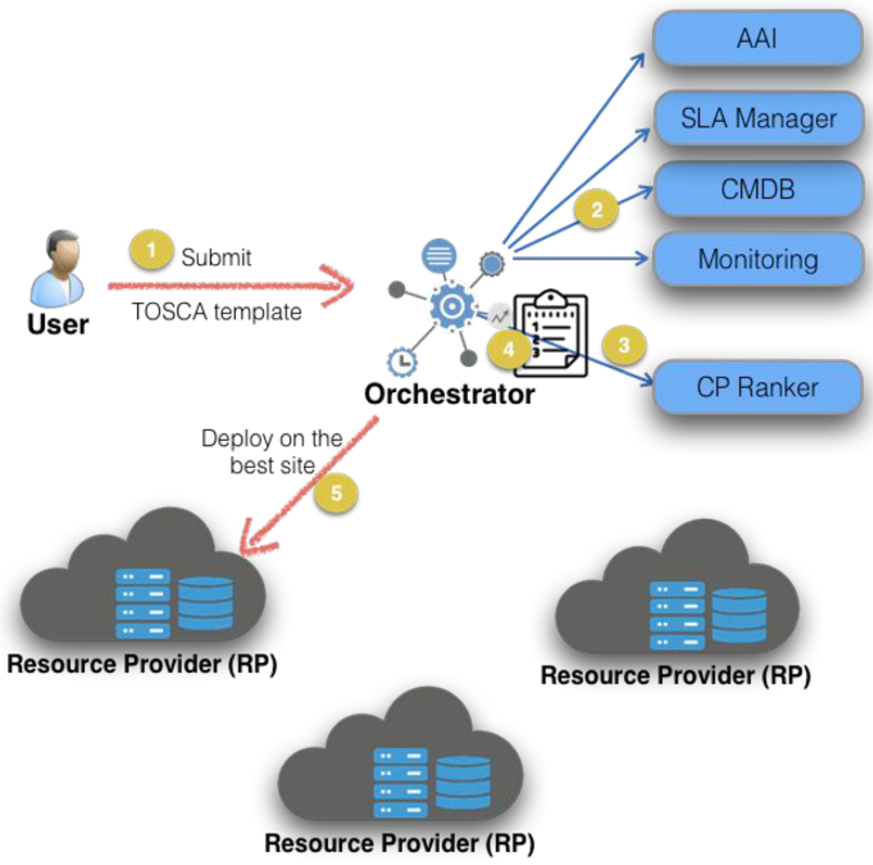


Figure 3: High-level architecture and workflow of the PaaS Orchestration system.

receives a deployment request from the user in terms of a TOSCA template, it manages to obtain a token with user information to access services at the IaaS level that also allows to keep track of the resources that have been created for that specific user. The Orchestrator modifies the TOSCA template submitted by the user and sends it to the Infrastructure Manager (IM) service [14] (the low-level orchestrator, at the IaaS level) that contacts the OpenStack sites for the provisioning of the resources for the deployment.

4. INDIGO-IAM clients management: the past

Most of the INFN Cloud services require the creation of an INDIGO-IAM client, which is an application registered on an Authorization Server (INDIGO-IAM of INFN Cloud in our case). The client interacts with the Authorization Server to manage users' authentication and access permissions. It is any entity that requires delegated access to the user's resources protected by the Authorization Server and this is done by requesting an access token defined within OAuth 2.0 [15] protocol by authenticating with the Authorization Server following the OpenID Connect [16] specification.

Let's take the "Jupyter with persistence for Notebooks" as an example. When the service is deployed on INFN Cloud, the Jupyter endpoint is given as an output. When clicking on it, the user is asked to log in by providing the INFN Authentication and Authorisation Infrastructure (AAI) credentials. Once provided, the user is then asked for approval so that the created INDIGO-IAM client can access the user's personal information. Once authorized, INDIGO-IAM returns an access token that will be used internally by the application for its scopes (e.g. ensuring that a user's private space is not accessible to other users). See Figure 4 for an overview of user interaction with the INDIGO-IAM service and the INDIGO-IAM client when connecting to the deployed service.

A reduction in the performance of the INDIGO-IAM service of INFN Cloud led to the launch of an internal analysis at INFN which allowed the cause to be identified in a growing number of INDIGO-IAM clients. Currently, an INDIGO-IAM client is created when creating and configuring some services, e.g. in Ansible roles or in Docker images. For example, in the "Jupyter with persistence for Notebooks" service, an INDIGO-IAM client with a client name equal to *jh-client* is created in a task of an Ansible role invoked by the TOSCA template of the service. A high number of clients with this name are currently in the INDIGO-IAM database but most of them are not used anymore by the deployments that required their creation as these deployments have been canceled in the meantime. The reason is that when a user triggers the deletion of a deployment, all the resources related to it are deleted apart from the INDIGO-IAM client. Indeed, since the INDIGO PaaS Orchestrator does not create the INDIGO-IAM client, it does not have the necessary information to delete it.

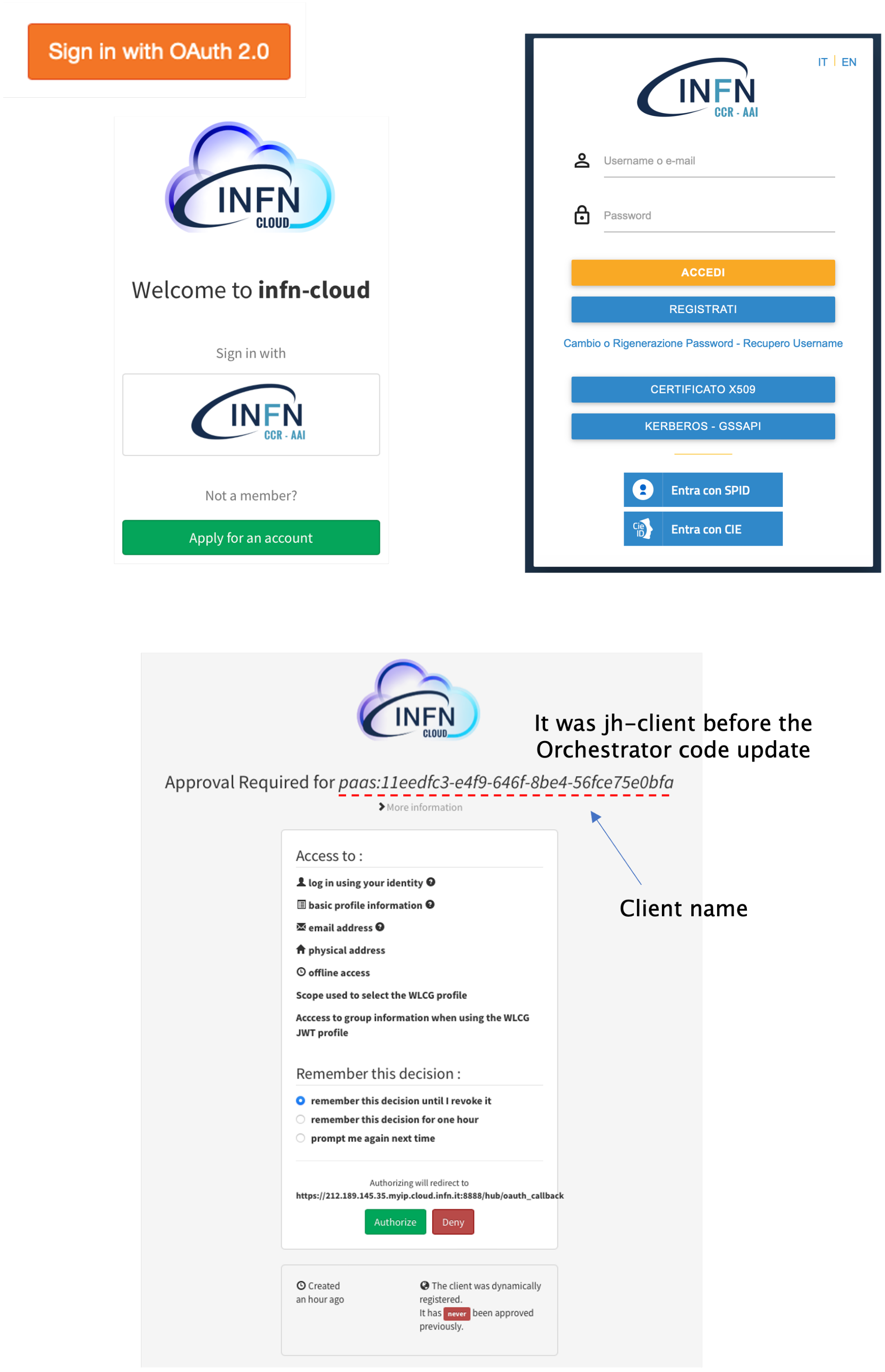
5. A new approach in the INDIGO-IAM clients management

The behavior described in Section 4 is not desirable, as resources that are no longer used remain in the INDIGO-IAM service. Therefore we decided to adopt a new strategy by implementing the following steps:

- introduction of a new TOSCA type [17] that identifies an INDIGO-IAM client (see Figure 5),
- modification of the TOSCA templates of services that require an INDIGO-IAM client (see Figure 6), e.g. the "Jupyter with persistence for Notebooks" service as shown in Figure 4,
- modification of the code of the INDIGO PaaS Orchestrator to manage the creation and deletion of INDIGO-IAM clients,
- adaptation of the Ansible roles for the services using the new configuration, e.g. they no longer have to create a new INDIGO-IAM client but use the client previously created by the INDIGO PaaS Orchestrator.

In particular when creating a deployment of a service requiring an INDIGO-IAM client (see Figure 7):

1. the user submits to the INDIGO PaaS Orchestrator the TOSCA template,
2. the INDIGO PaaS Orchestrator sends a request to the INDIGO-IAM service for the creation of an INDIGO-IAM client, with *client_name* in the form *paas:{deployment_uuid}* where



Approval Required for *paas:11eedfc3-e4f9-646f-8be4-56fce75e0bfa*

Access to :

- log in using your identity
- basic profile information
- email address
- physical address
- offline access
- Scope used to select the WLCG profile
- Acccess to group information when using the WLCG JWT profile

Remember this decision :

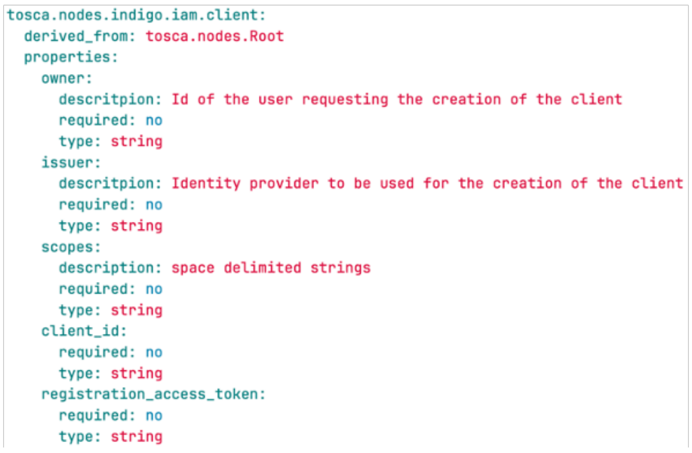
- remember this decision until I revoke it
- remember this decision for one hour
- prompt me again next time

Authorizing will redirect to
https://212.189.145.35.myip.cloud.infn.it:8888/hub/oauth_callback

Created an hour ago

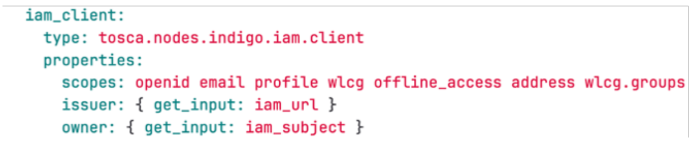
The client was dynamically registered. It has never been approved previously.

Figure 4: Overview of user interaction with the INDIGO-IAM service of INFN Cloud and the INDIGO-IAM client in a Jupyter with persistence for Notebooks service deployed in INFN Cloud. See text for more details.



```
tosca.nodes.indigo.iam.client:
  derived_from: tosca.nodes.Root
  properties:
    owner:
      descritpion: Id of the user requesting the creation of the client
      required: no
      type: string
    issuer:
      descritpion: Identity provider to be used for the creation of the client
      required: no
      type: string
    scopes:
      description: space delimited strings
      required: no
      type: string
    client_id:
      required: no
      type: string
    registration_access_token:
      required: no
      type: string
```

Figure 5: Definition of the TOSCA type for the INDIGO-IAM client.



```
  iam_client:
    type: tosca.nodes.indigo.iam.client
    properties:
      scopes: openid email profile wlcg offline_access address wlcg.groups
      issuer: { get_input: iam_url }
      owner: { get_input: iam_subject }
```

Figure 6: Part of a TOSCA template where the creation of an INDIGO-IAM client is defined.

deployment_uuid is the UUID that the INDIGO PaaS Orchestrator has associated with the deployment (by using this form it is easy to trace an INDIGO-IAM client from the deployment that requested its creation),

3. the INDIGO PaaS Orchestrator gets back information about the created INDIGO-IAM client,

4. the INDIGO PaaS Orchestrator saves the information of *client_id* and *registration_access _token* in its database,

5. the INDIGO PaaS Orchestrator submit to the IM the modified TOSCA template, containing the information of the created INDIGO-IAM client.

When deleting a deployment of a service that required the creation of an INDIGO-IAM client (see Figure 8), the procedure is as follows:

1. the user submits to the INDIGO PaaS Orchestrator the request for deployment deletion,

2. the INDIGO PaaS Orchestrator retrieves from its database the information of *client_id* and *registration_access_token* of the created INDIGO-IAM client,

3. the INDIGO PaaS Orchestrator sends to the INDIGO-IAM service the request for the INDIGO-IAM client deletion,

4. the INDIGO PaaS Orchestrator submits to the IM the request for deployment deletion.

The solution just presented offers flexibility to users, enabling them to:

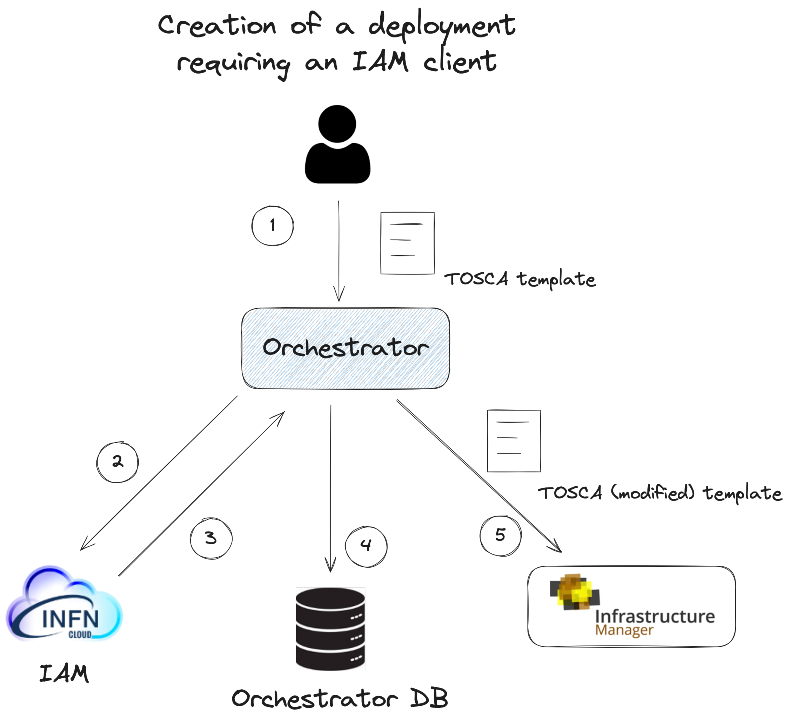


Figure 7: Overview of how the INDIGO PaaS Orchestrator manages the creation of INDIGO-IAM clients. See text for more details.

- create multiple INDIGO-IAM clients,
- select the identity provider, i.e. use another INDIGO-IAM service instead of the INFN Cloud one,
- define scopes,
- assign the client owner, which allows the user to customize the INDIGO-IAM client configuration at a later time.

6. A similar problem: the management of S3 buckets

A similar problem to what has been described in Section 4 concerns the proliferation of S3 buckets. Some services that integrate S3 storage as a backend (such as Sync&Share as a Service), usually write a lot of data into S3 buckets. These buckets are created in the Swift of the INFN Cloud backbone through an Ansible role (see Figure 9) and when the deletion of the deployment is triggered, the associated buckets (and their contents) persist, resulting in a continuous increase in disk space consumption.

To address this problem, we found it convenient to delegate both the creation and the deletion of buckets to the INDIGO PaaS Orchestrator, similarly to what has been described in Section 5. In this case, as we have to manage the AWS access and secret keys of each bucket, particular precautions have been taken to preserve such user-related information. For this reason, we decided to use the

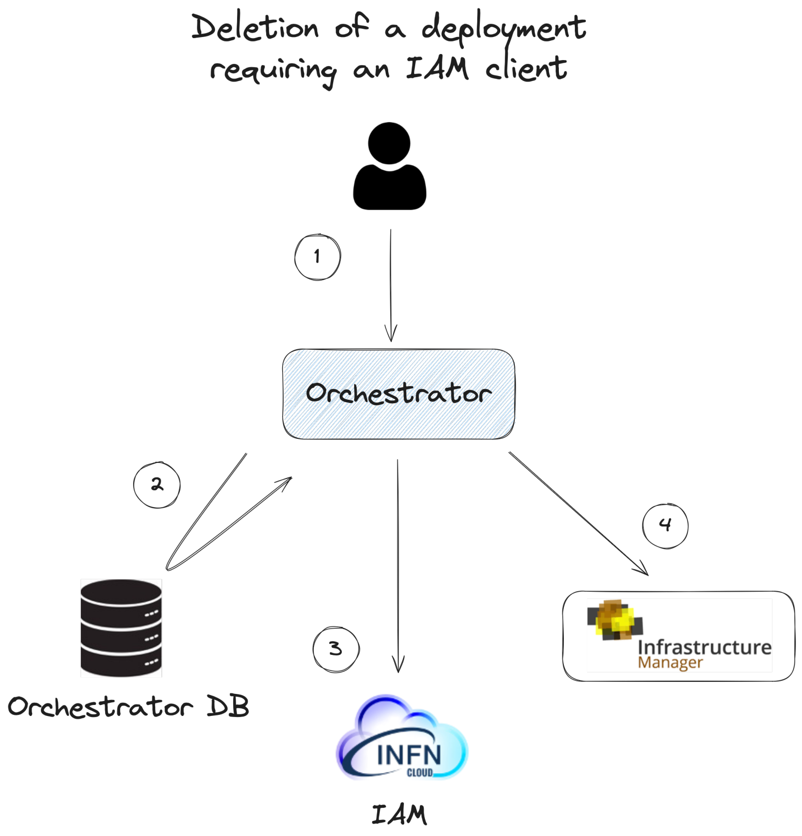
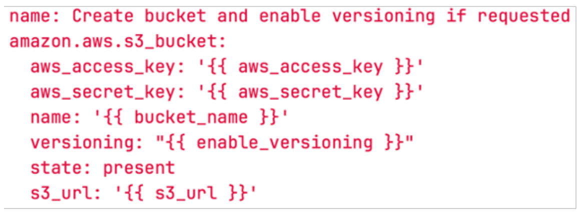


Figure 8: Overview of how the INDIGO PaaS Orchestrator manages the deletion of INDIGO-IAM clients. See text for more details.



```
name: Create bucket and enable versioning if requested
amazon.aws.s3_bucket:
  aws_access_key: '{{ aws_access_key }}'
  aws_secret_key: '{{ aws_secret_key }}'
  name: '{{ bucket_name }}'
  versioning: "{{ enable_versioning }}"
  state: present
  s3_url: '{{ s3_url }}'
```

Figure 9: Part of the Ansible role used for the Sync&Share as a Service where the S3 bucket is created by using the *amazon.aws.s3_bucket* Ansible module.

HashiCorp Vault [18] service already available within INFN Cloud platform to store them securely. The INDIGO PaaS Orchestrator will read these secrets directly from Vault and then create or delete the buckets together with the related deployment, avoiding loss of control over these resources.

7. Conclusions

INFN Cloud, which is based on a distributed and federated cloud infrastructure, provides its users with a set of services through the INDIGO PaaS Orchestration system, most of which require the creation of an INDIGO-IAM client or S3 bucket. In this paper, we presented a solution that has been designed and implemented to delegate the management of any INDIGO-IAM client and S3 bucket to the INDIGO PaaS Orchestrator, avoiding the unintended persistence of resources that are no longer associated with any deployment. The new approach involves the introduction of a new TOSCA type, the update of the TOSCA templates related to the services involved, the review of the PaaS Orchestrator code to create and delete INDIGO-IAM clients and S3 buckets, the interaction with Hashicorp Vault to manage secrets, and the adaptation of the Ansible recipes of the services to the new configuration. The presented solution for the management of INDIGO-IAM clients and S3 buckets is in production in INFN Cloud.

8. Acknowledgments

The work presented in this paper has been funded by the NextGenerationEU European initiative through the Italian Ministry of University and Research, PNRR Mission 4, Component 2 - TeRABIT: Investment 3.1, Project code IR0000022 - CUP I53C21000370006.

References

[1] *INDIGO-IAM*, `https://indigo-iam.github.io/v/v1.8.4/docs/`, accessed on 28/08/2024

[2] *Ansible*, `https://www.ansible.com`, accessed on 28/08/2024

[3] D. Salomoni, I. Campos, L. Gaido, G. Donvito, M. Antonacci, P. Fuhrman, J. Marco, A. Lopez-Garcia, P. Orviz, I. Blanquer et al. (2016), `1603.09536`

[4] *Kubernetes*, `https://kubernetes.io`, accessed on 28/08/2024

[5] *INFN Cloud dashboard*, `https://my.cloud.infn.it`, accessed on 28/08/2024

[6] *TOSCA Simple Profile in YAML Version 1.1*, `http://docs.oasis-open.org/tosca/TOSCA-Simple-Profile-YAML/v1.1/csprd01/TOSCA-Simple-Profile-YAML-v1.1-csprd01.html`, accessed on 28/08/2024

[7] *Ansible roles*, `https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_reuse_roles.html`, accessed on 28/08/2024

[8] *Docker containers*, `https://www.docker.com/resources/what-container/`, accessed on 28/08/2024

[9] *Helm charts*, `https://helm.sh/docs/topics/charts/`, accessed on 28/08/2024

[10] *Jinja*, `https://jinja.palletsprojects.com/en/3.1.x/`, accessed on 28/08/2024

[11] M. Antonacci, D. Salomoni, PoS **ISGC&HEPiX2023**, 020 (2023)

[12] *Orchestrator*, `https://github.com/indigo-paas/orchestrator`, accessed on 28/08/2024

[13] *JSON Web Tokens*, https://auth0.com/docs/secure/tokens/json-web-tokens, accessed on 28/08/2024

[14] *Infrastructure Manager*, https://github.com/grycap/im, accessed on 28/08/2024

[15] *OAuth 2.0*, https://www.rfc-editor.org/rfc/rfc6749, accessed on 28/08/2024

[16] *OpenID Connect*, https://openid.net/specs/openid-connect-core-1_0.html, accessed on 28/08/2024

[17] *TOSCA types used in INFN Cloud*, https://baltig.infn.it/infn-cloud/tosca-types, accessed on 28/08/2024

[18] *HashiCorp Vault*, https://www.vaultproject.io, accessed on 28/08/2024