AI-based approach for provider selection in the INDIGO PaaS Orchestration system of INFN Cloud

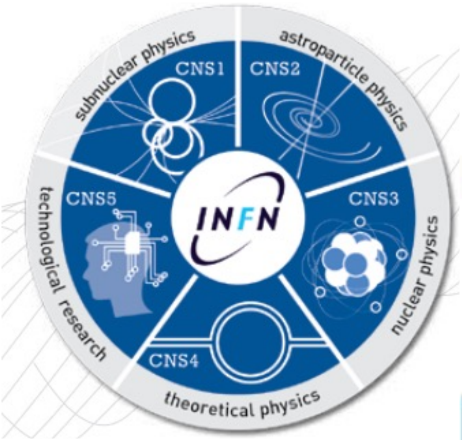
Luca Giommi – INFN CNAF

A. Costantini – INFN CNAF

F. Debiase, M. Antonacci, G. Savarese , G. Vino, G. Donvito – INFN Bari

1st AI-INFN User Forum

INFN and its facilities



- INFN manages and supports the **largest public computing infrastructure for scientific research** spread throughout the country

- INFN has been running for more than 20 years a **distributed infrastructure** which currently offers about 150K CPU cores, 120 PB of enterprise-level disk space and 120 PB of tape storage, serving more than 40 international scientific collaborations

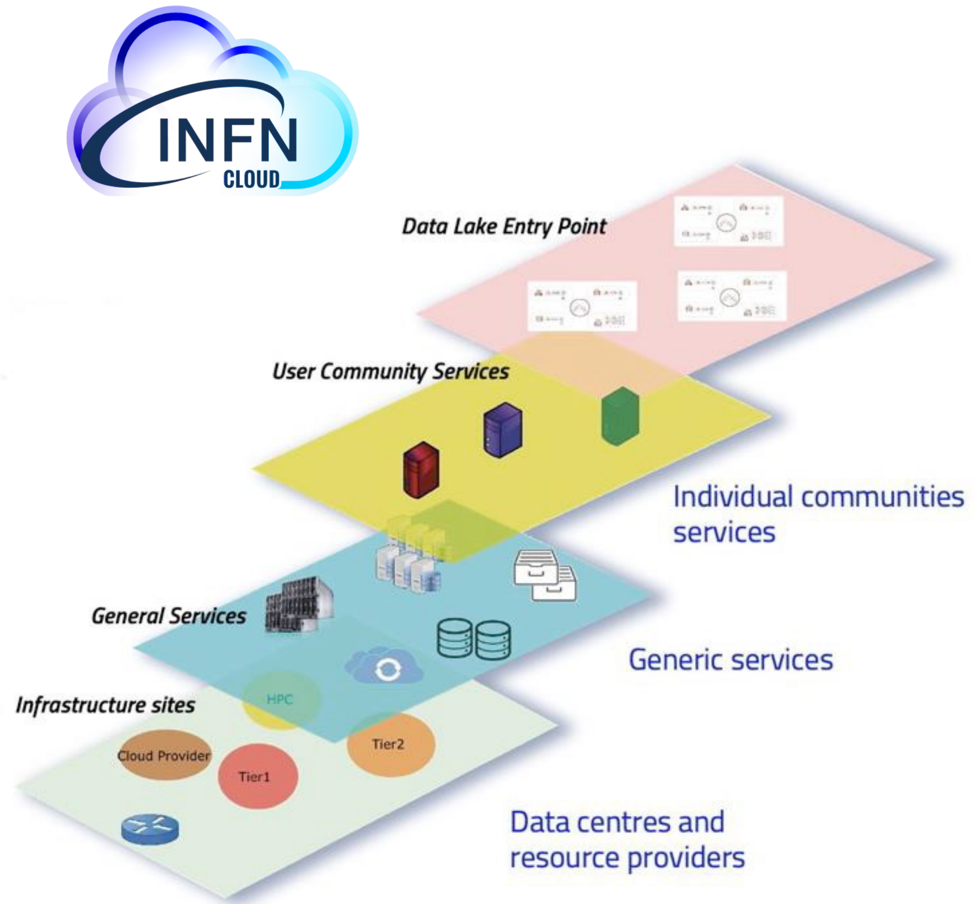
- INFN was one of main promoters of the GRID project to address LHC computing needs. Since then INFN has been participating to **WLCG** that includes more than 170 sites around the world, loosely organized in a tiered model.
  - In Italy, there are the Tier-1 at CNAF, Bologna and 9 Tier-2 centers

Birth of INFN Cloud

- To support and evolve use cases that could not easily exploit the Grid paradigm, for many years several INFN sites have been investing in **Cloud computing** infrastructures
  - heterogeneous in hardware, software and cloud middleware

- 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 an "user-centric" infrastructure making available to the final users a dynamic **set of services** tailored on specific use cases
  - leveraging the outcomes of several national and European cloud projects where INFN actively participated

- INFN Cloud was officially made available to users in **March 2021**



Proposed architecture for ICSC and TeRABIT projects

Resources in INFN Cloud

The infrastructure is based on a core **backbone** connecting the large data centers of CNAF and Bari, and on a set of loosely coupled distributed and federated sites connected to the backbone

- Backbone sites are high speed connected and host the INFN Cloud core services
- **Federated clouds**: Cloud@CNAF, CloudVeneto, Cloud@ReCaS-Bari, Cloud-CT, Cloud-IBISCO-Na. Coming soon: LNGS, Milano, HTC in Tier-2s, HPC bubbles

Backbone

~ 2000 vCPU
~ 15 TB RAM
~ 1.6 PB Storage (RAW)
> 600 TB Storage net,
~ 10% SSD, ~ 320 TB for object storage

Federated Clouds

~ 3955 vCPU
~ 84 TB RAM
~ 343 TB Storage net

Portfolio of services

SaaS

- Notebook as a Service
- INFN Cloud Registry (Harbor)
- INFN Cloud object storage (Minio)
- INFN Cloud monitoring (Grafana)

PaaS

- Virtual Machine
- Docker Compose
- Run Docker
- INDIGO IAM as a Service
- Elasticsearch & Kibana
- Kubernetes cluster
- Spark + Jupyter cluster
- HTCondor (mini or cluster)
- Jupyter (w/o Matlab) with persistence
- Sync & Share
- ML_INFN working station
- CYGNO working station

IaaS

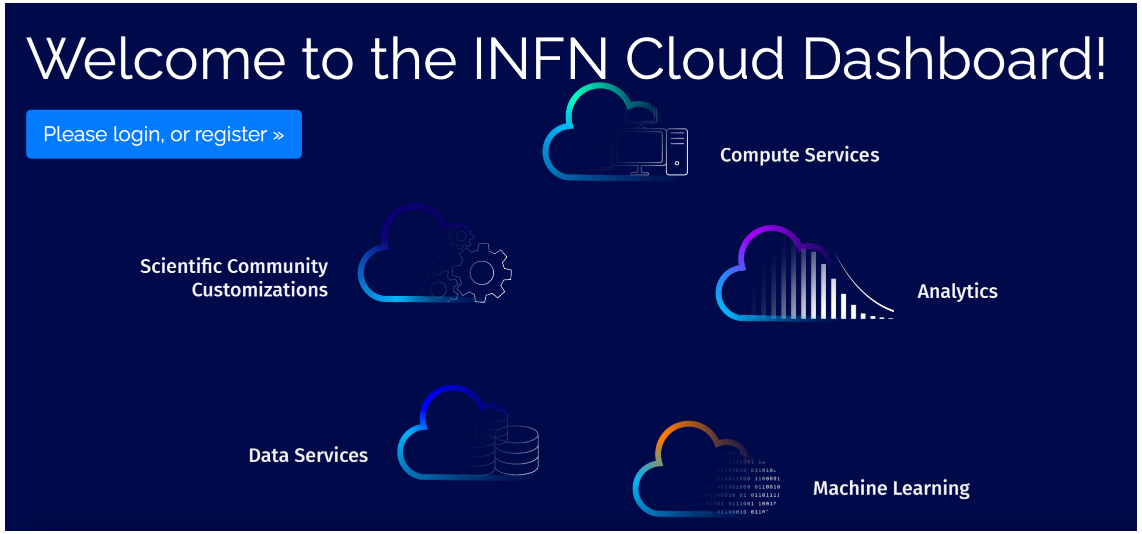
- Start & Stop
- Hostname choice
- Open ports

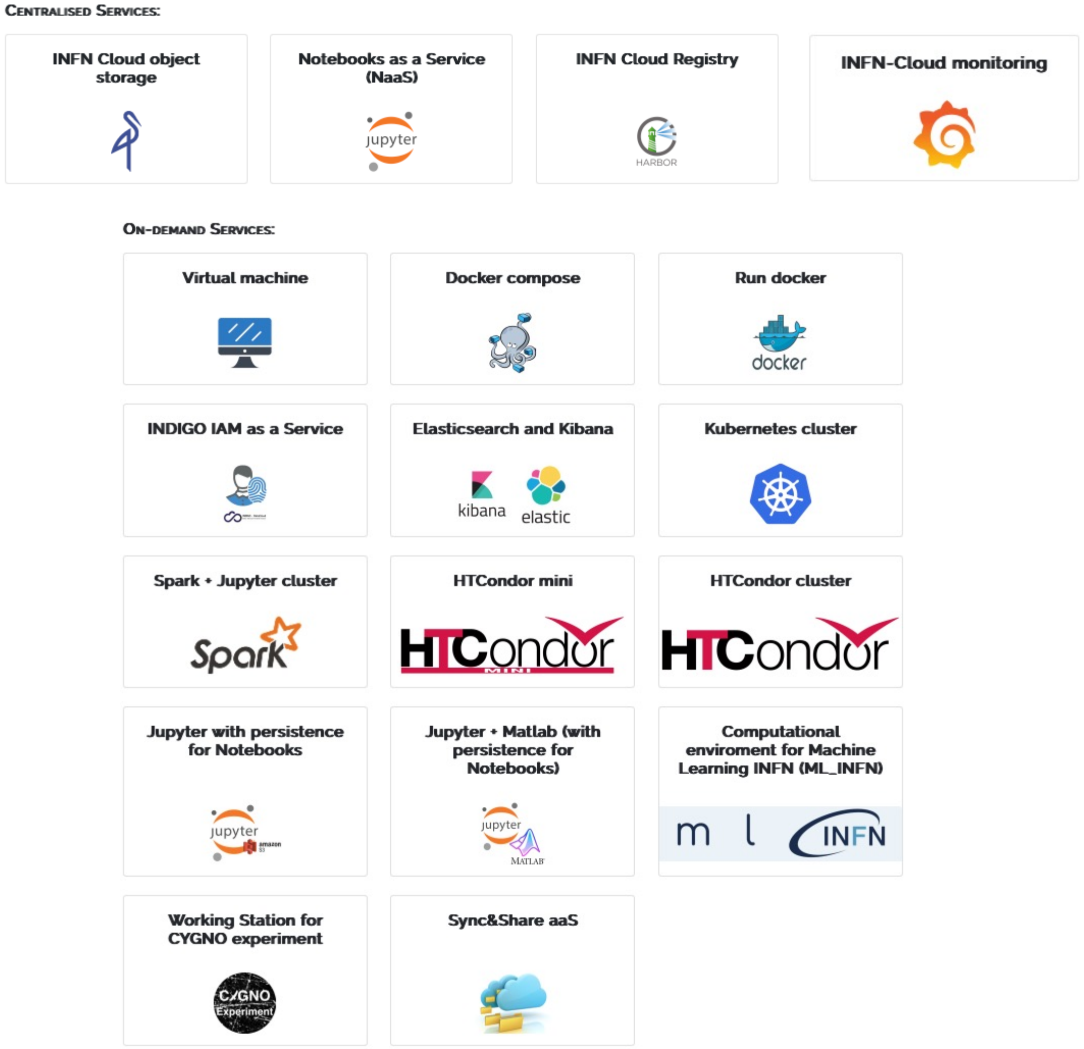
The INFN Cloud dashboard

https://my.cloud.infn.it

It allows users to:

- ➢ access centralized services
- ➢ instantiate PaaS services independently





The Infrastructure as Code paradigm

All services are described through an **Infrastructure as Code** paradigm via a combination of:

- **TOSCA** (**T**opology and **O**rchestration **S**pecification for **C**loud **A**pplications) templates, to model an application stack
- **Ansible** roles, to manage the automated configuration of virtual environments
- **Docker** containers, to encapsulate high-level application software and runtime
- **Helm** charts, to manage the deployment of an application in Kubernetes clusters

It allows to reduce manual processes and increase flexibility and portability across environments

```
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],
      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
```

TOSCA

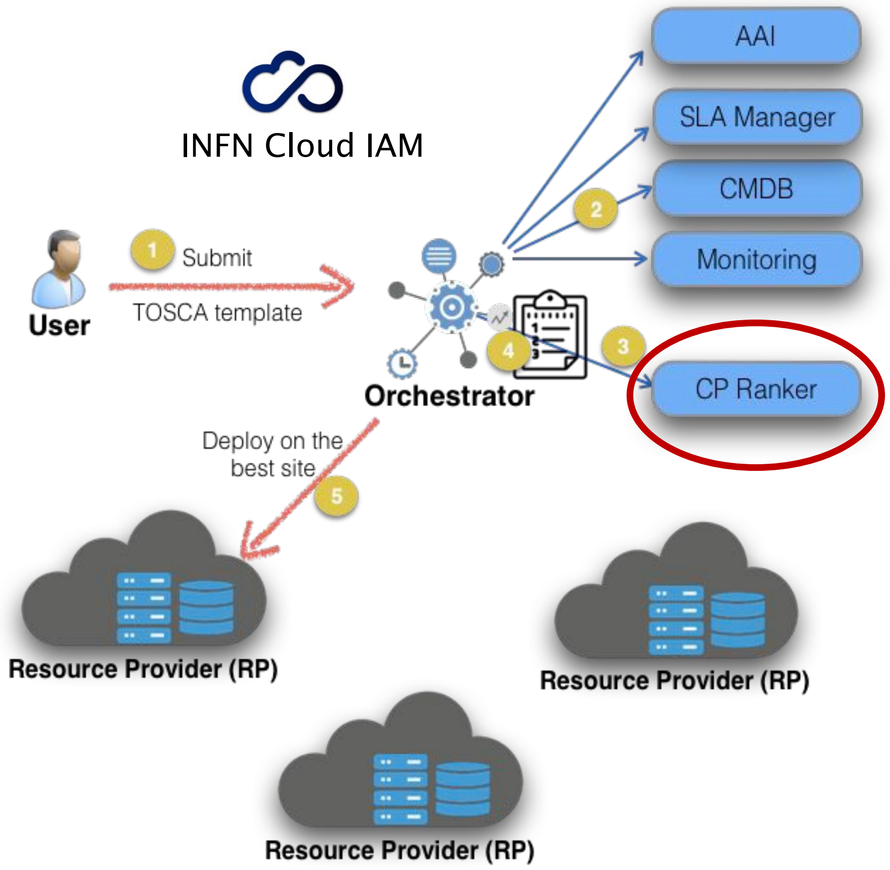
Ref: [TOSCA Simple Profile in YAML Version 1.1]

```
- 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)
```

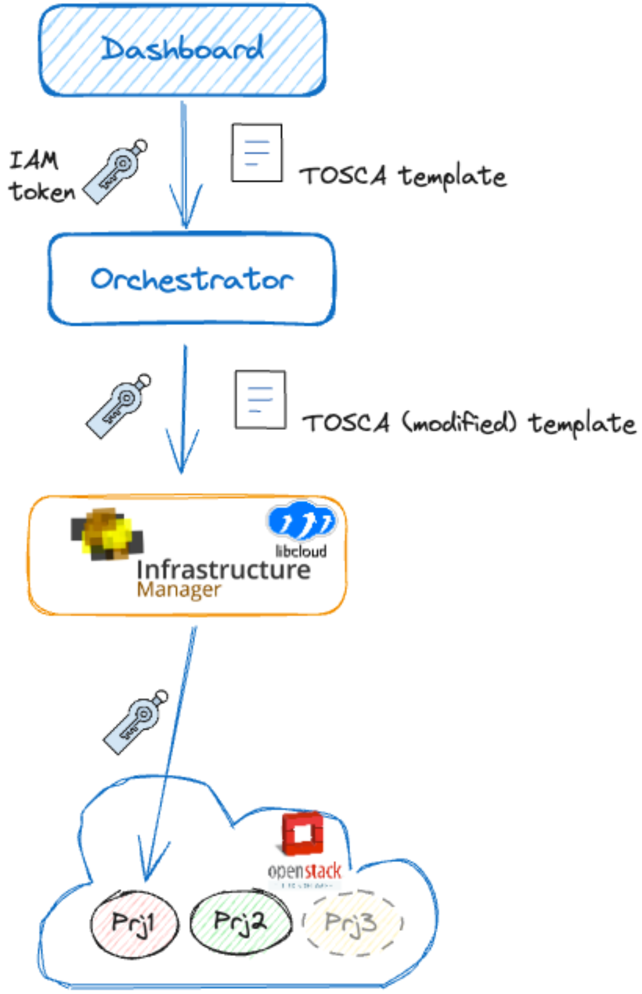
Ansible

The PaaS Orchestration system



Gives a ranked list of providers. **We want to add ML here**

The Orchestrator interacts with the provider services through the **Infrastructure Manager (IM)** for deploying complex and customized virtual infrastructures on on the IaaS platforms made available by the federated providers

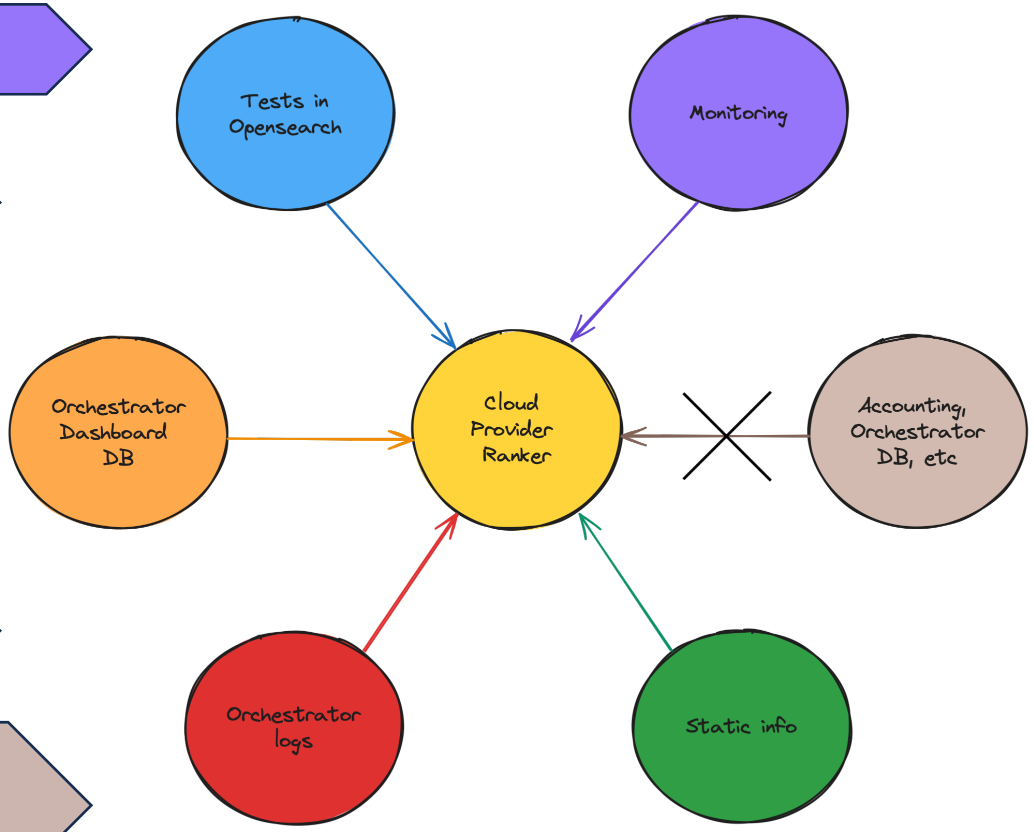


Machine Learning workflow

1) Identification of data sources
2) Identification of features
3) Data collection and dataset creation: associate info from each source with each deployment
4) Data exploration
5) Data cleaning
6) Data transformation and feature engineering
7) Model and training design
8) Performance evaluation

Identification of data sources

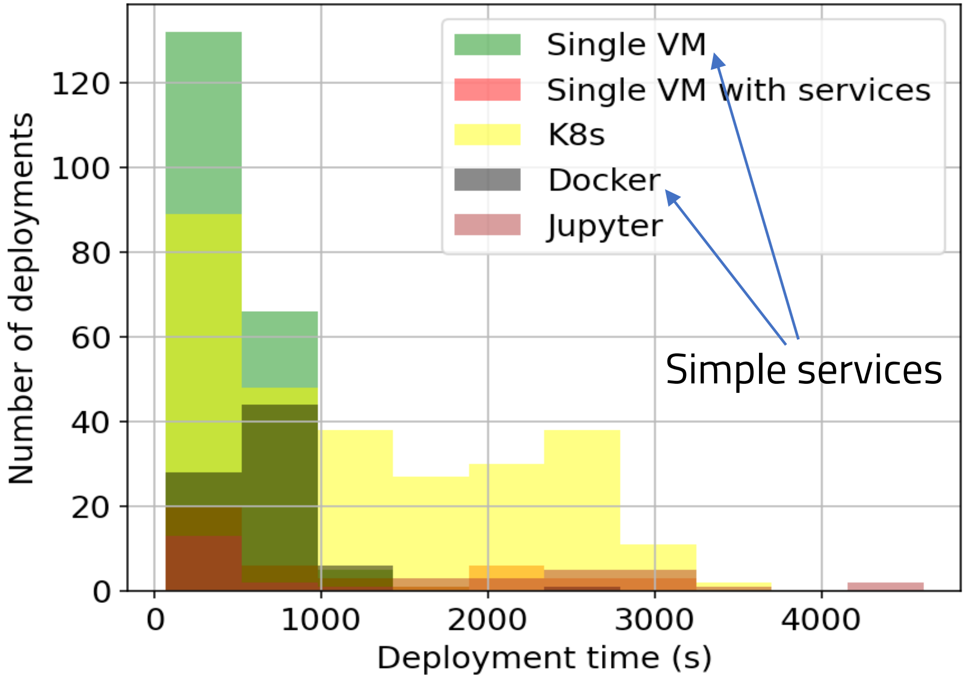
- Monitoring: info about resource usage per group and provider over time
- Tests in Opensearch: info about simple tests (e.g. VM creation) done on the providers
- Orchestrator Dashboard DB: info about all deployments
- Orchestrator logs: info about the history of all deployments
- Static info: badwidth, overbooking, etc
- Accounting: aggregated info in time slots about resource usage per group and provider
  Orchestrator DB: info about not deleted deployments



Info about data

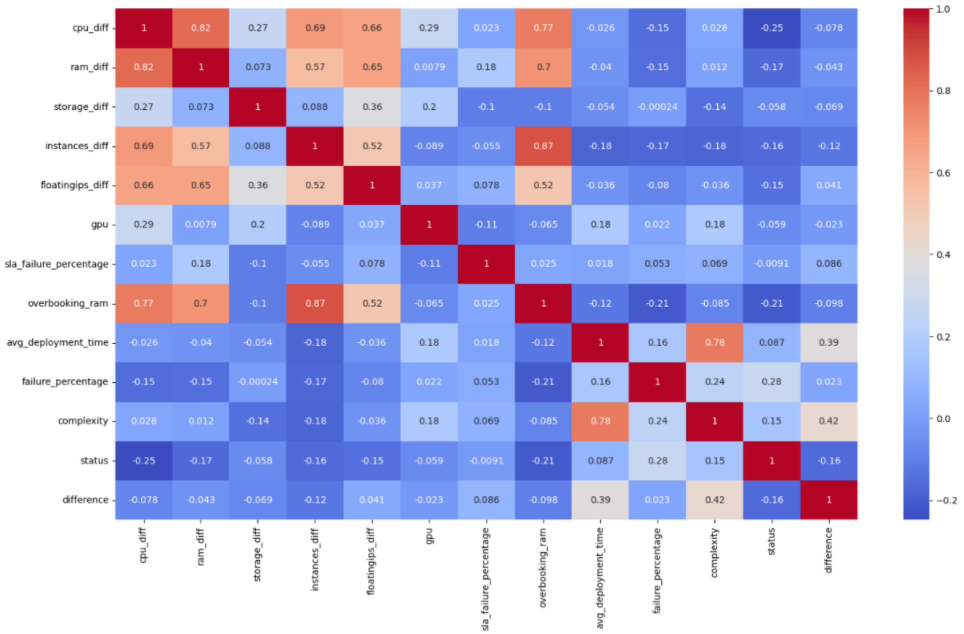
- **6 months of data** used: 08.2023 – 01.2024, **643 entries** (**very few!**)
- Different entries associated to different service deployments/templates: tried **grouping** them according to their **complexity**

Type of service	# of create complete	# of create failed
Single VM	137	67
Single VM with services	23	19
K8S	159	124
Docker	41	38
Jupyter	25	10
All	343	300



Reduction in features number

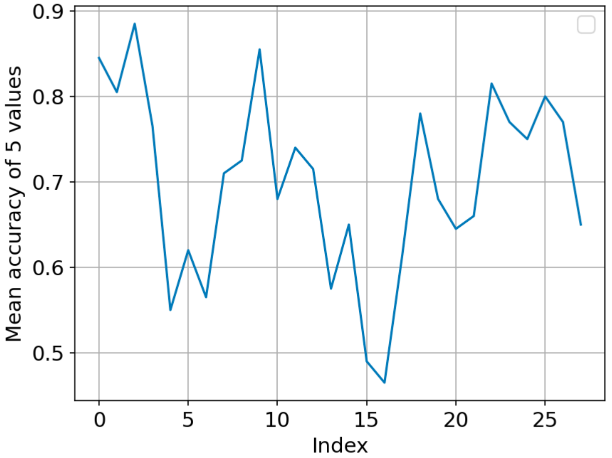
- **Reduction in the number of features** through data cleaning and feature engineering, e.g. ram_diff = (quota_ram – ram_used) – requested_ram
- Finally used **11 features**

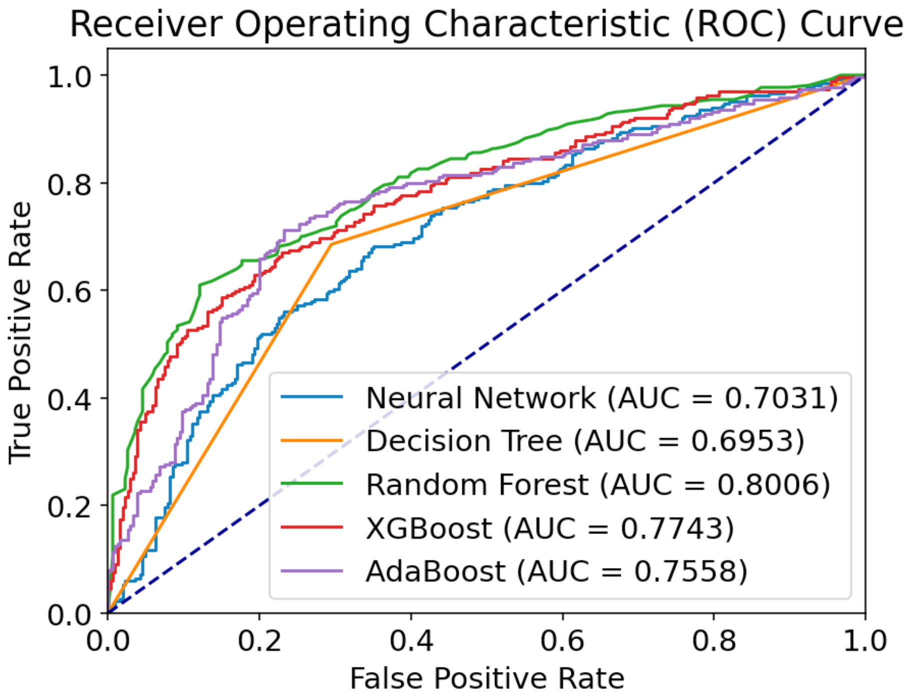


Feature	Importance
ram_diff	0.162
cpu_diff	0.147
failure_percentage	0.146
avg_deployment_time	0.120
storage_diff	0.114
instances_diff	0.098
sla_failure_percentage	0.093
floatingips_diff	0.088
complexity	0.022
overbooking_ram	0.006
gpu	0.003

Model and training design and performance evaluation

- **Two models** to create:
  - **classification** for success/failure of a deployment
  - **regression** for creation/failure time of a deployment
- Defined training procedure using data of recent and **sliding time windows with fixed size** (75 entries for training and 10 for test)
- Classification: compared different models with parameter tuning. Best **Random Forest**





Mean accuracy using RandomForest: 0.692

Regression models and future directions

Regression part

- Tried linear, ridge, lasso, polynomial, SVM, decision tree, KNN, random forest, MLP regression
- Used MSE, RMSE, MAE error and R2 score as metrics
- Message: still room for improvements

What's next?

- Trying to improve the results (especially for regression) by **creating a better dataset**: more statistics, balanced dataset and better definition of failures
- **Automatizing data collection** and redesigning the CPR service through an **online-learned** model
- Plans for exploring **Reinforcement Learning** techniques

Thank you

luca.giommi@cnaf.infn.it