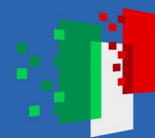




Finanziato
dall'Unione europea
NextGenerationEU



Ministero
dell'Università
e della Ricerca



Italiadomani
PIANO NAZIONALE
DI RIPRESA E RESILIENZA



terabit

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

Workshop sul Calcolo nell'INFN – Palau | 20-24 Maggio 2024

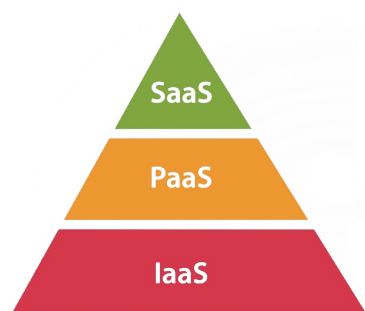
The INFN Cloud infrastructure



INFN decided to implement a **national Cloud computing infrastructure** for research

- as a **federation** of existing distributed infrastructures
- as an “user-centric” infrastructure which makes 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, e.g. INDIGO DataCloud

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



e.g. Notebook as a Service

e.g. Virtual Machine, Docker compose

e.g. Start & Stop, Hostname choice

Backbone

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

Federated Clouds

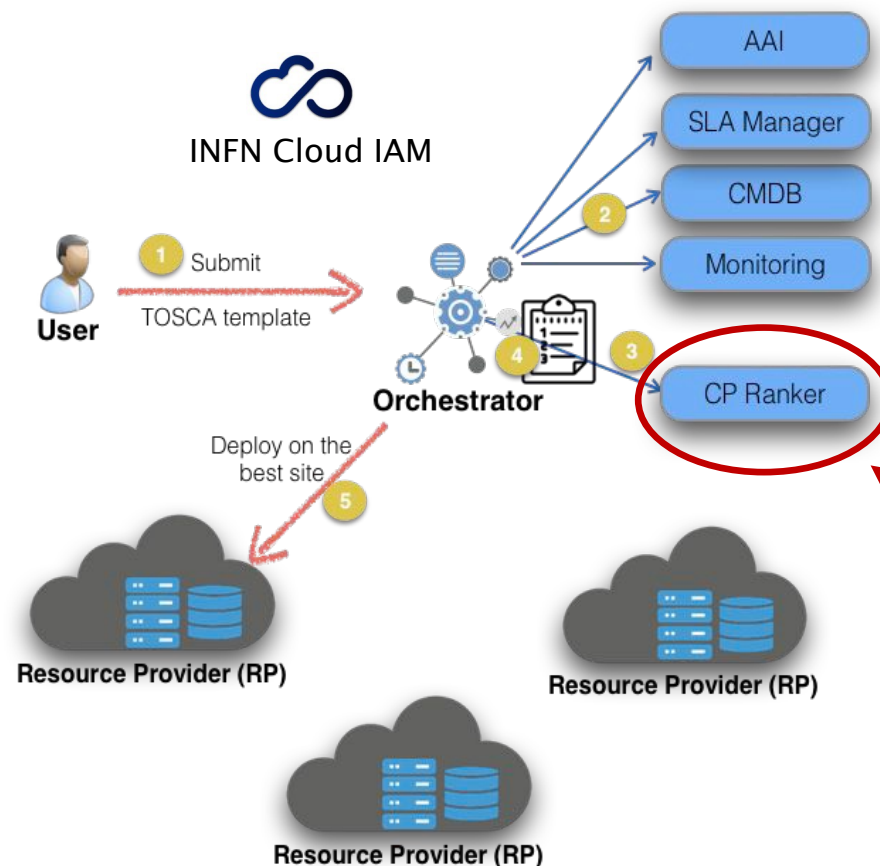
~ 3955 vCPU
~ 84 TB RAM
~ 343 TB Storage net



The INDIGO PaaS Orchestration system of INFN Cloud

The federative middleware of INFN Cloud is based on the **INDIGO PaaS orchestration system**, consisting of interconnected open-source microservices

- The **Orchestrator** receives high-level deployment requests in the form of TOSCA templates and coordinates the process of creating deployments
- The Orchestrator interacts with the provider services through the **Infrastructure Manager (IM)** for deploying complex and customized virtual infrastructures on the IaaS platforms made available by the federated providers



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

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

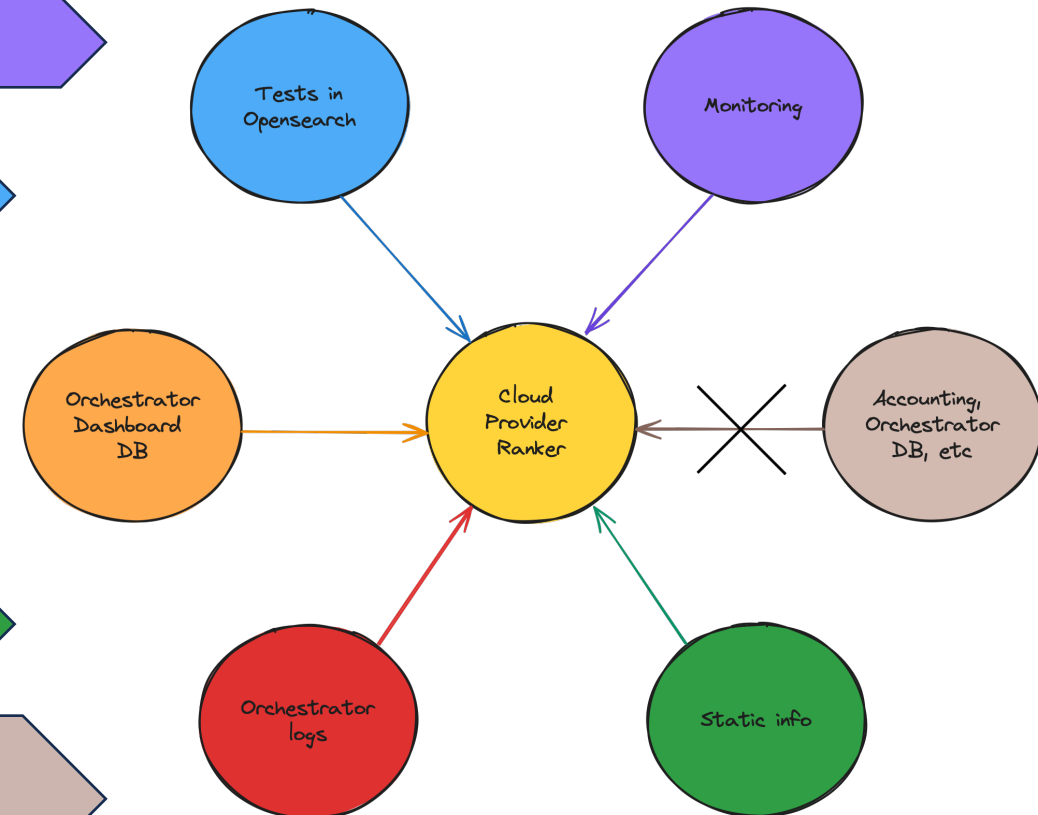
Tests in Opensearch: info about simple tests (e.g. VM creation) done on the providers

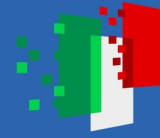
Orchestrator Dashboard DB: info about all deployments

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

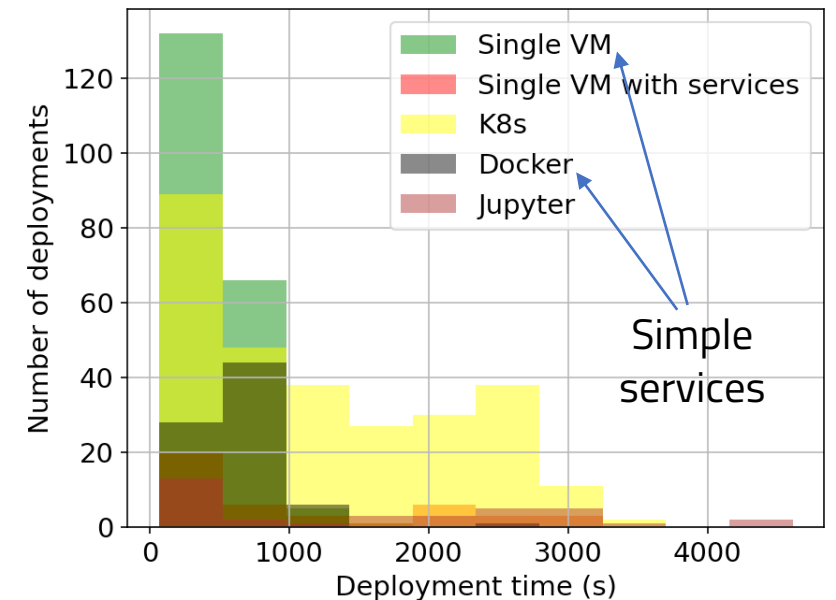




Results

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

Ranking	Feature	Importance
1	ram_diff	0.162
2	cpu_diff	0.147
3	failure_percentage	0.146
4	avg_deployment_time	0.120
5	storage_diff	0.114

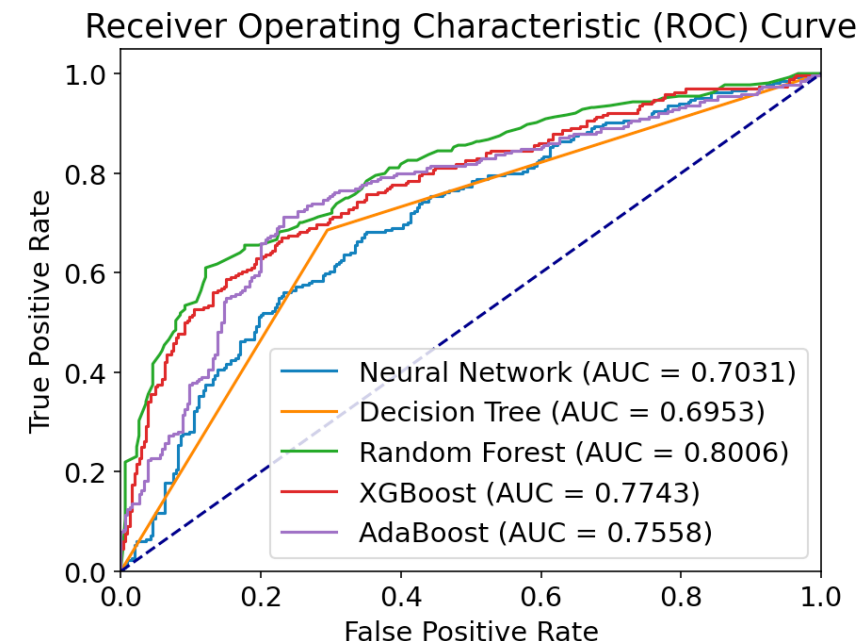


Results and future directions

- **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**
- Classification: compared different models with parameter tuning. Best **Random Forest**
- For the regression part 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

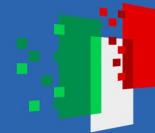




Finanziato dall'Unione europea
NextGenerationEU



Ministero dell'Università e della Ricerca



Italiadomani
PIANO NAZIONALE DI RIPRESA E RESILIENZA



Thank you

luca.giommi@cnaif.infn.it

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

Luca Giommi – INFN CNAF
A. Costantini – INFN CNAF
F. Debono, M. Antonacci, G. Severice, G. Vigna, B. Donato – INFN Bari
Workshop on CloudNet@INFN – Pisa | 22-23 June 2023

The INFN Cloud infrastructure

INFN decided to implement a national Cloud computing infrastructure for research and education as a federation of existing distributed infrastructures. It is an "open ecosystem" infrastructure which makes available to the final users a dynamic set of services tailored on specific use cases.

INFN Cloud was officially made available to users in March 2023.

The federative middleware of INFN Cloud is based on the INDIGO PaaS orchestration system, consisting of interconnected open-source microservices.

The Orchestrator receives high-level deployment requests in the form of TOSCA templates and coordinates the process of creating deployments across the infrastructure through the provider agents for deploying complex and customized virtual infrastructures on the host providers made available by the federated providers.

Problems addressed and solution with the use of Artificial Intelligence

In the default configuration, the Orchestrator determines the provider where to place the deployment request starting from an ordered list of providers, selected according to the group the user belongs to. This list is provided by the Cloud Manager (CM) service, which applies a ranking algorithm using a restricted set of metrics relating to the deployment and service level agreements (SLAs) defined for the providers. Then, the Orchestrator searches for requests to the first provider in the list in case of failure it asks to the next provider until the list is exhausted. Our work aims to improve the existing system by identifying and using more appropriate information with an approach based on Artificial Intelligence.

Machine Learning workflow

- 1) Identification of data sources
- 2) Identification of features
- 3) Data collection and dataset creation associated with 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

Verification of data sources

- Monitoring info about resource usage per group and provider over time
- Health Checks info about single hosts (e.g. VM ready state or the process)
- Performance info about the history of all deployments
- Small VM Submits, cancelling, etc.
- Monitoring info about the time taken about creating and deleting the group infrastructure
- Monitoring info about the SLAs defined for each deployment

Results and future directions

- 6 months of data used (06/2023 – 01/2024, 643 entities being fed)
- Different entities associated to different service deployment templates used grouping them according to their complexity
- Reduction in the number of features through data cleaning and feature engineering (e.g. ram_off + ipsize_ram - ram_used - requested_ram)
- Finaly used 7 features

Ranking	Feature	Importance
1	ram_off	0.192
2	cpu_off	0.187
3	disk_size_percentage	0.146
4	cpu_memory_usage	0.132
5	storage_off	0.116

► The model 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 old time windows with fixed size

► Classification compared different models (Support Vector, Decision Tree, Random Forest, XGBoost, AdaBoost) with parameter tuning (Best Random Forest)

► For the regression part call open for improvements

What's next?

- Try to improve the results (especially for regression) by creating a better dataset more extensive, balanced dataset and better definition of features
- Automating data collection and releasing the CM service through an online-learning model
- Plans for exploring Reinforcement Learning techniques