Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
terabit

The INFN Cloud platform: state of the art and services implementation

Luca Giommi – INFN CNAF

G. Savarese, E. Serra, M. Perniola, M. Gattari, J. Gasparetto, G. Vino

M. Antonacci, A. Costantini, G. Donvito, E. Vianello, B. Martelli, C. Grandi

International Symposium on Grids & Clouds (ISGC) 2025 | 16-21 March 2025

INFN and its facilities

- 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

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

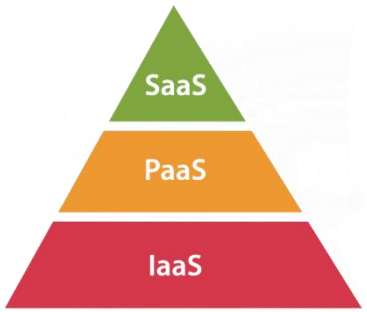
- INFN was one of main promoters of the GRID project to address LHC computing needs. Since then INFN has been participating to **WLCG** that includes more than 170 sites around the world, loosely organized in a tiered model.
  - In Italy, there are the Tier-1 at CNAF, Bologna and 9 Tier-2 centers

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
~ 10 PB Storage (RAW)
~ 6% SSD

Federated Clouds
~ 3100 vCPU
~ 15 TB RAM
~ 400 TB Storage net

The Infrastructure as Code paradigm

All PaaS services are defined using an **Infrastructure as Code** paradigm, based on a procedural paradigm that aims to reduce manual processes and increase flexibility and portability across environments, 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

```yaml
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
```

```yaml
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

```yaml
- 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"
```

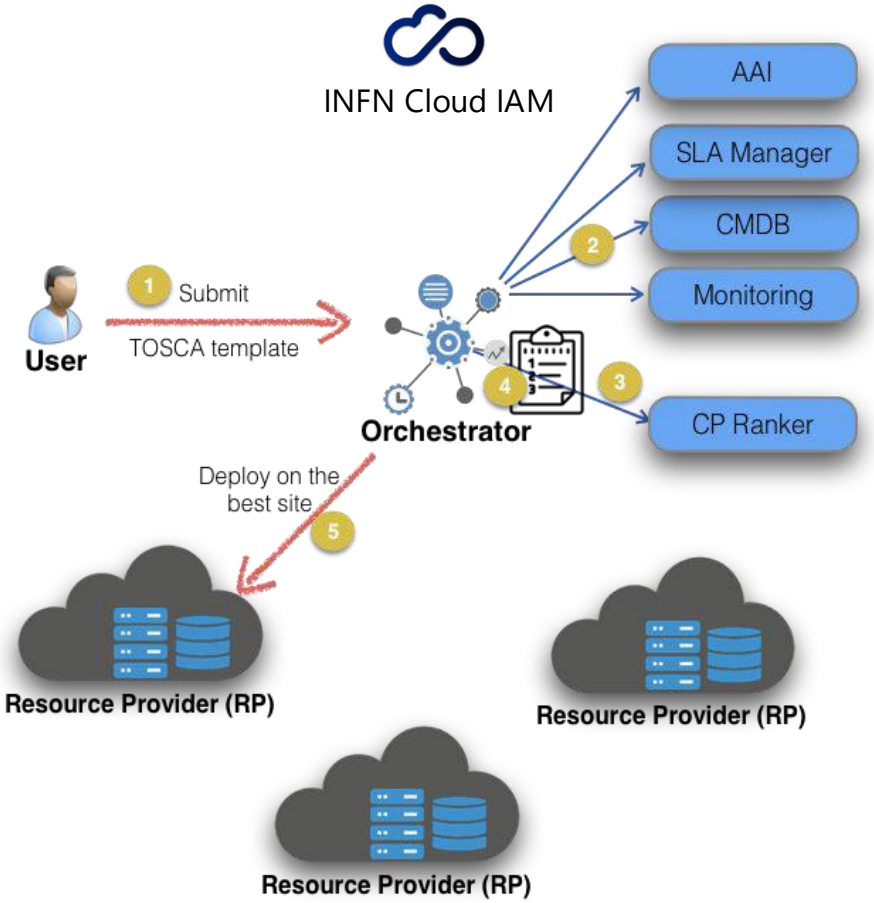
Ansible

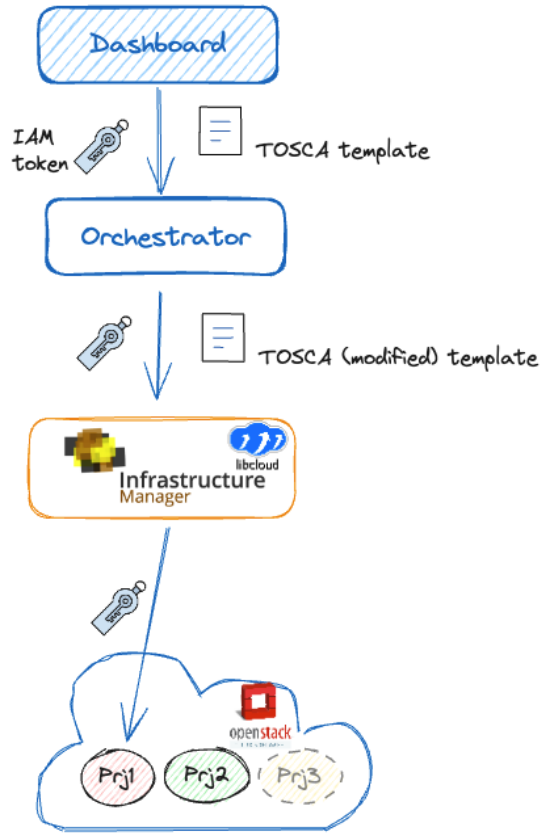
```yaml
- name: Run Jupyter Hub
  ansible.builtin.shell:
    cmd: docker-compose up -d
    chdir: /usr/local/share/dodasts/jupyterhub
  when: (run_jupyter | bool)
```

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 (currently based on OpenStack)

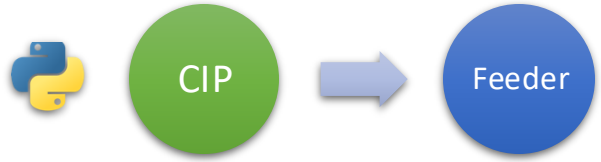




Introduction of the Federation Registry

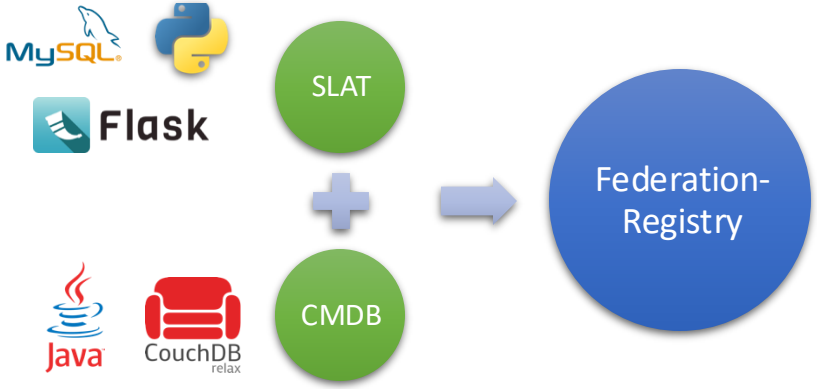
Federation Registry Feeder

- Written in Python
- Reads the configuration file of each provider
- Can use multiple IDP and through OIDC-agent retrieve the access token for the ops user
- For each project-region pair, it retrieves the available resources (flavors, images, networks, quotas and more) directly from the providers.
- Update the Federation Registry

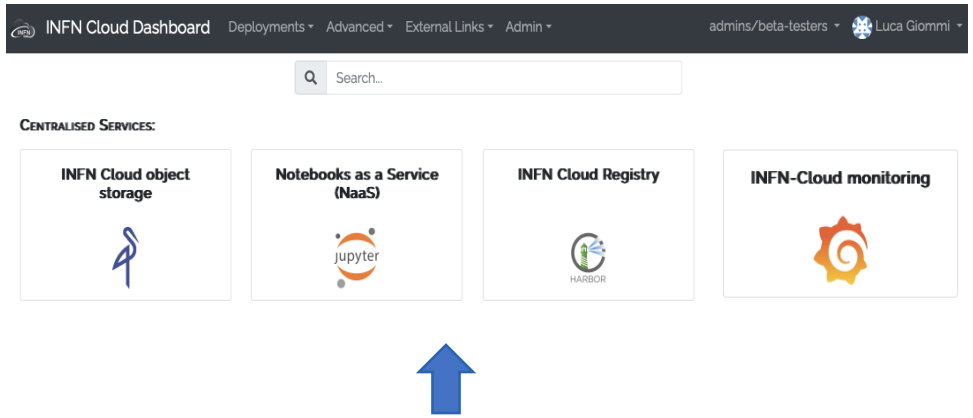


Federation Registry

- Written in Python
- Uses FastAPI for API definition
  - Automatic API documentation
- Uses the flaat library to secure the REST API. It uses OAuth for authentication and authorization
- Uses Neo4j as graph database
  - Efficient horizontal scaling to handle high-throughput and very large data sets

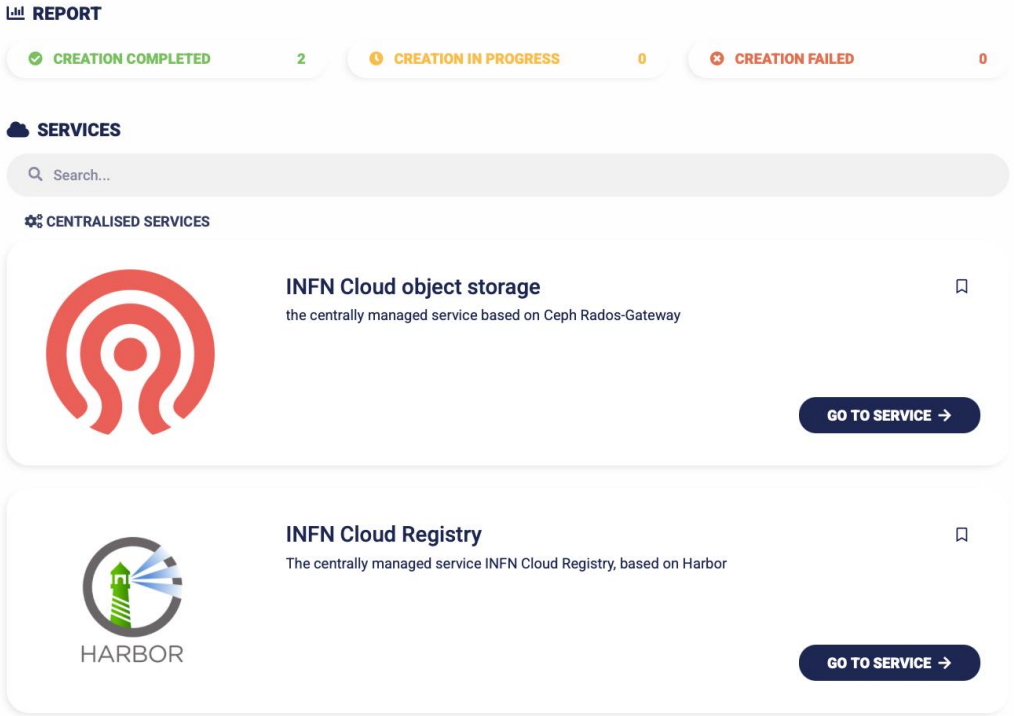


The PaaS Orchestrator Dashboard



Old style

New style



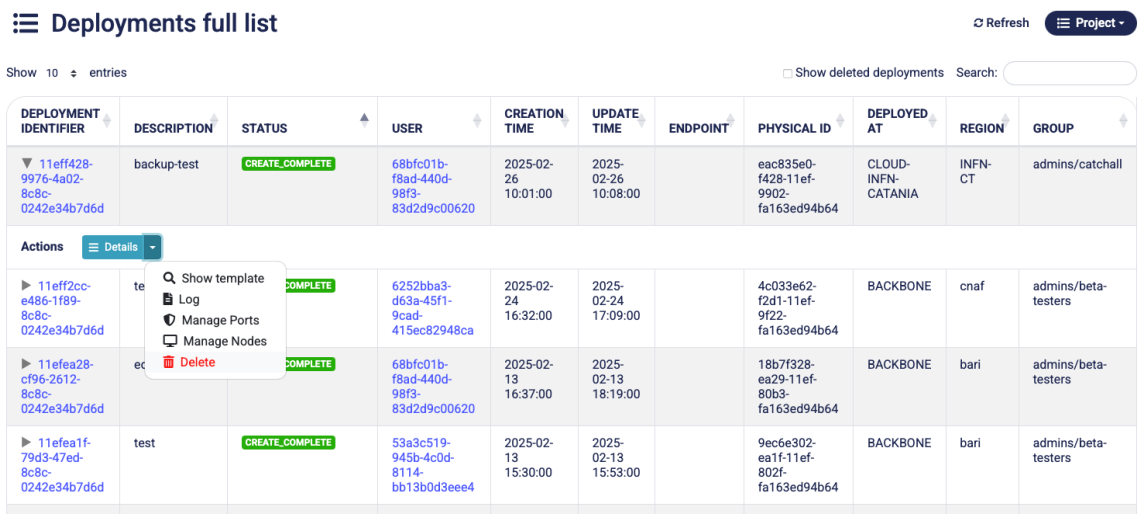
https://my.cloud.infn.it

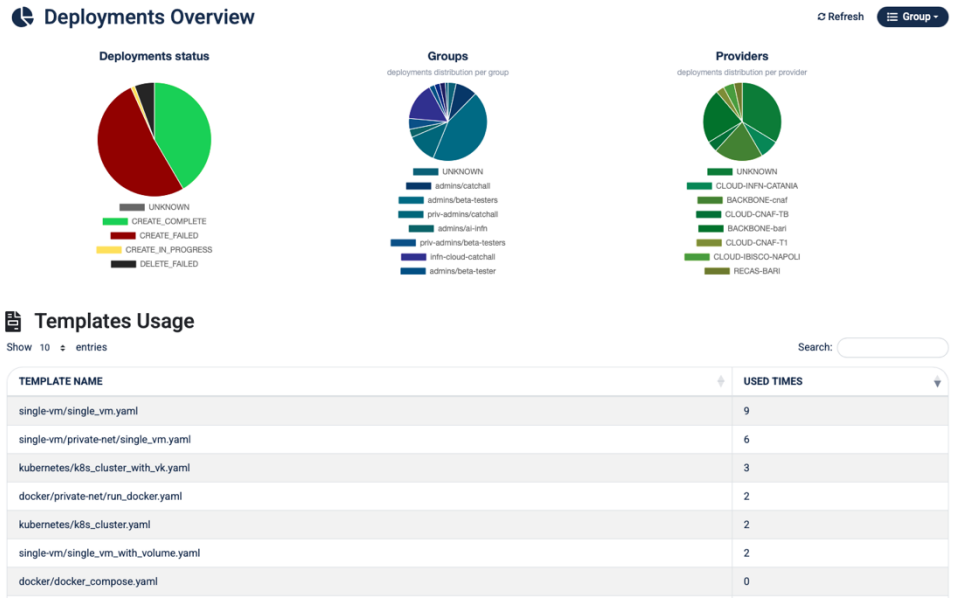
Updates on the production services

- **Transition from SLAT + CMDB + CIP to Federation Registry + Feeder v1.0.1**
- **Orchestrator v4.0.1**
  - Integration with the Federation Registry
  - Dependencies moved to CNAF Nexus
- **Orchestrator Dashboard v4.3.0**
  - Integration with the Federation Registry
  - Fix for port management related to pre-configured rules
  - Reset of deployments in inconsistent states
  - Retry of failed deployments
  - Check on S3 bucket names
- **Tosca Templates v1.1.1**
  - Migration of PaaS services to Debian 12
  - Integration with the Federation Registry

Updates and new features in testing phase

- Admins can access other users' deployments, view logs, and delete them if needed
- Admins can view the full list of deployments of all users
- Added *Usage statistics* section to see the deployments distribution by type, user group, and provider
- Lists of flavors/images that users can choose from are now retrieved from the providers, rather than being hardcoded
- Info about deleted deployments remains saved in the Dashboard DB



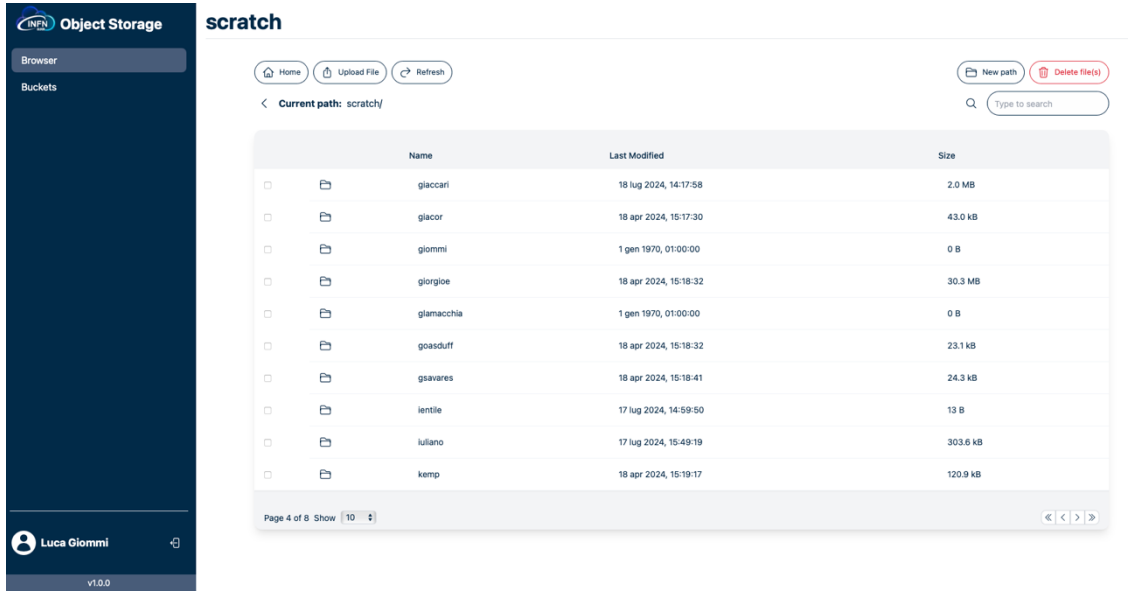


Renovation of the S3 Web App for the Object Storage Service

- The object storage service of INFN Cloud is now based on **CEPH / Rados Gateway** (previously MinIO)
  - Uses Open Policy Agent (OPA) for fine-grained Authorization
  - Two zones (CNAF and Bari) with independent Ceph Storage Clusters
  - Three instances of RADOS Gateway run in high availability within each zone

- Developed a new version (v1.0.0) of the **Web App** used as a GUI for the Object Storage Service
  - Built on Next.js and React.js
  - OIDC protocol with IAM to generate Json Web Token (JWT)
  - Uses IAM Access Token to perform STS with RGW
  - S3 operations using AWS SDK library
  - New graphical UI to be consistent with other INFN web applications

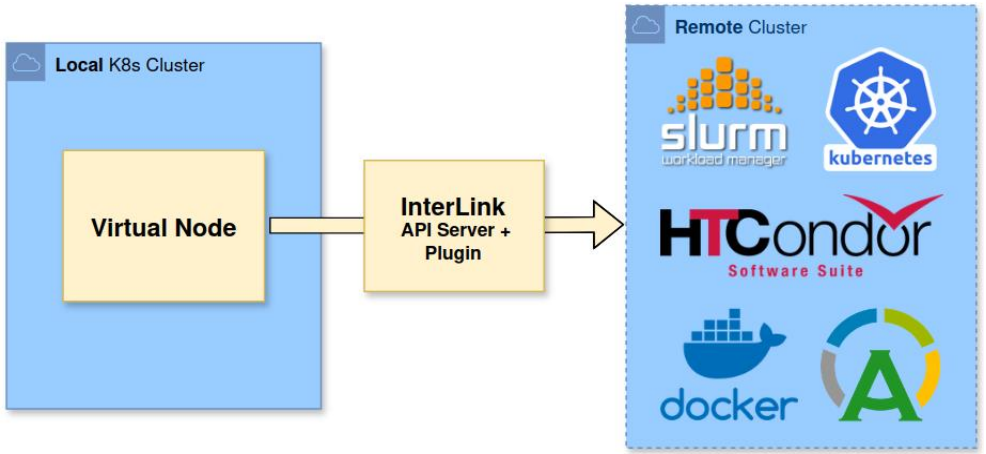
https://s3webui.cloud.infn.it



New PaaS service: Kubernetes cluster with an InterLink Virtual Node

- **It creates a Kubernetes Cluster that enables the transparent offloading of Kubernetes workloads to remote computation systems**
- It uses the interLink plugin developed within the interTwin project
  - **interTwin**: project funded by the EU for the development of an open-source platform, called Digital Twin Engine (DTE), to handle "digital twins" of selected scientific communities
  - **interLink**: allows transparent offloading of resources to heterogeneous computing providers

- Workload offloading
  - specify requirements, e.g. the number of GPUs
  - resources may not be available on the local cluster
  - workload can be opportunistically offloaded to a remote cluster where resources are available
- InterLink main components
  - **Virtual Node**: translate requests for a Kubernetes POD execution into a remote call to the InterLink API server.
  - **InterLink API Server**: a pluggable REST server that talks to the remote cluster



Thank you

luca.giommi@cnaf.infn.it