How to deploy containers on INFN-CLOUD

Corso *Docker e orchestrazione di container (II Edizione)* - Feb 7-11 2022

Marica Antonacci (INFN BA)

What is INFN-Cloud?

INFN Cloud is an internal project which aims to

- manage a (large) fraction of the INFN resources in a sustainable and optimized way;
- make different INFN communities able to access resources, regardless of the availability of local and dedicated hardware (including special hw like GPUs), of the availability of IT skilled people;
- focus on high-level added value services, not on “infrastructures”, to support:
  - ➢ Scientific Computing
  - ➢ Development and R&D, testing of new services
  - ➢ Training activities
  - ➢ Support to INFN data centers (for example for backups of services, etc)

INFN Cloud is built on top of INFN experiences, know-how and solutions developed during several projects and initiatives.

The INFN Cloud architecture

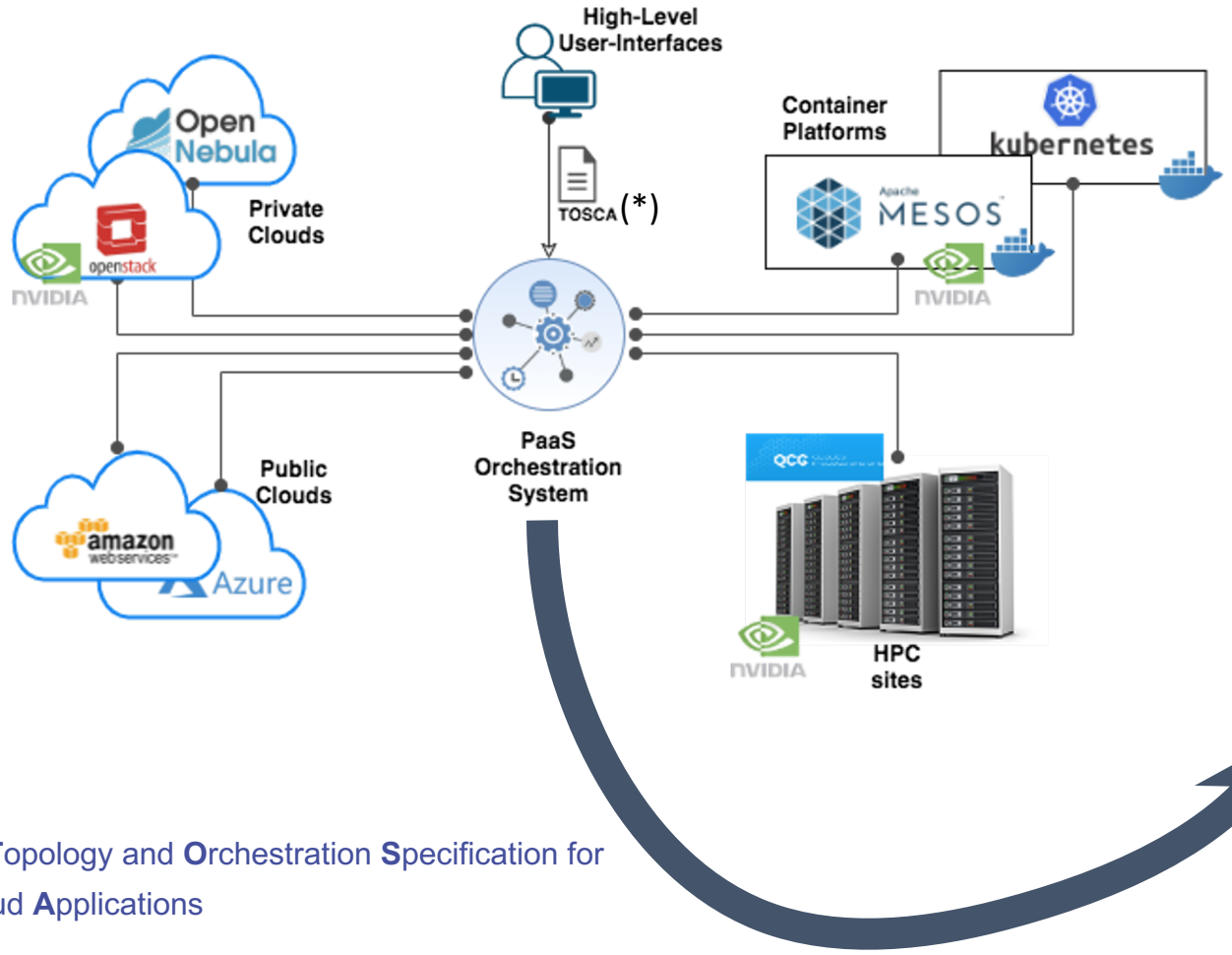
Architecturally INFN Cloud is a **federation** of existing infrastructures

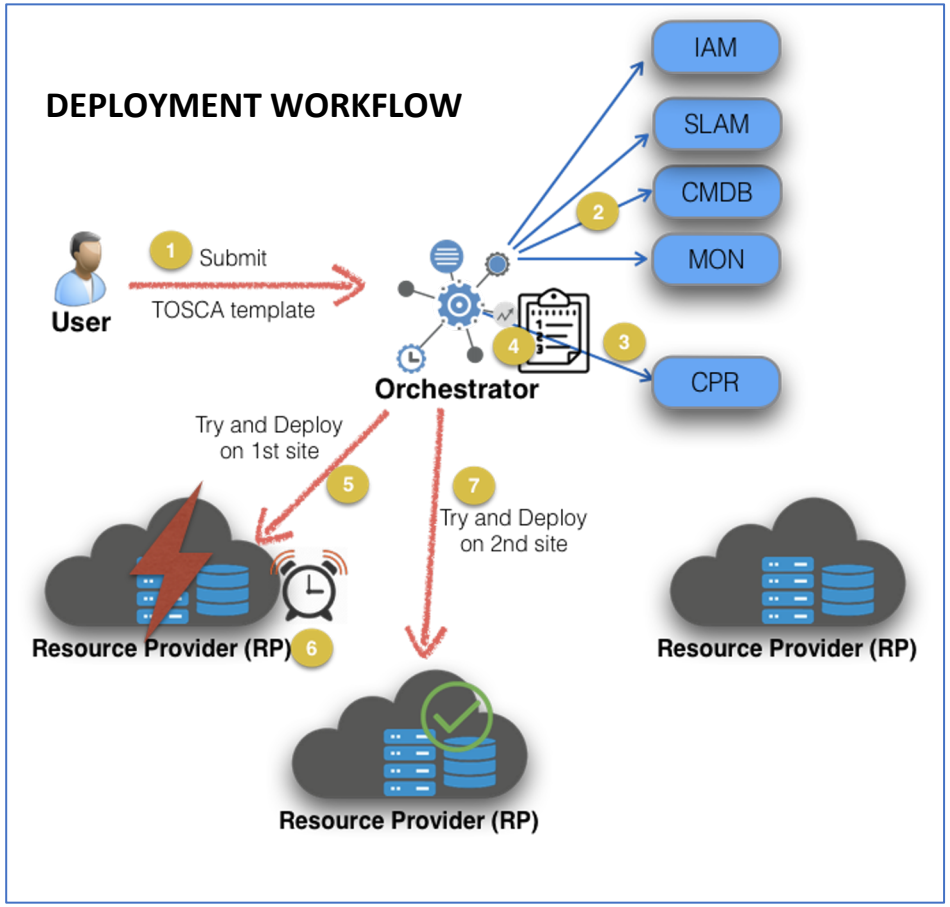
- ❏ **the INFN Cloud backbone**, that consists of two tightly coupled federated sites: BARI and CNAF
- ❏ **a scalable set of satellite sites**, geographically distributed across Italy, and loosely coupled.
  - Currently Cloud@CNAF, CloudVeneto and ReCaS-Bari are federated with the backbone

Key enabling factors for the federation:

- ❏ leverage the same authentication/authorization layer based on **INDIGO-IAM**
- ❏ agree on a consistent set of policies and **participation rules** (user management, SLA, security, etc.)
- ❏ transparent and dynamic orchestration of the resources across all the federated infrastructures through the **INDIGO PaaS Orchestrator**

PaaS Orchestration System (from 10Km)





(*) Topology and **O**rchestration **S**pecification for **C**loud **A**pplications

Ref: [TOSCA Simple Profile in YAML Version 1.1]

The INFN-Cloud services

Virtual Machines (VM) possibly with external volume for storing data.

Docker containers

Pre-configured environment for **data analytics**

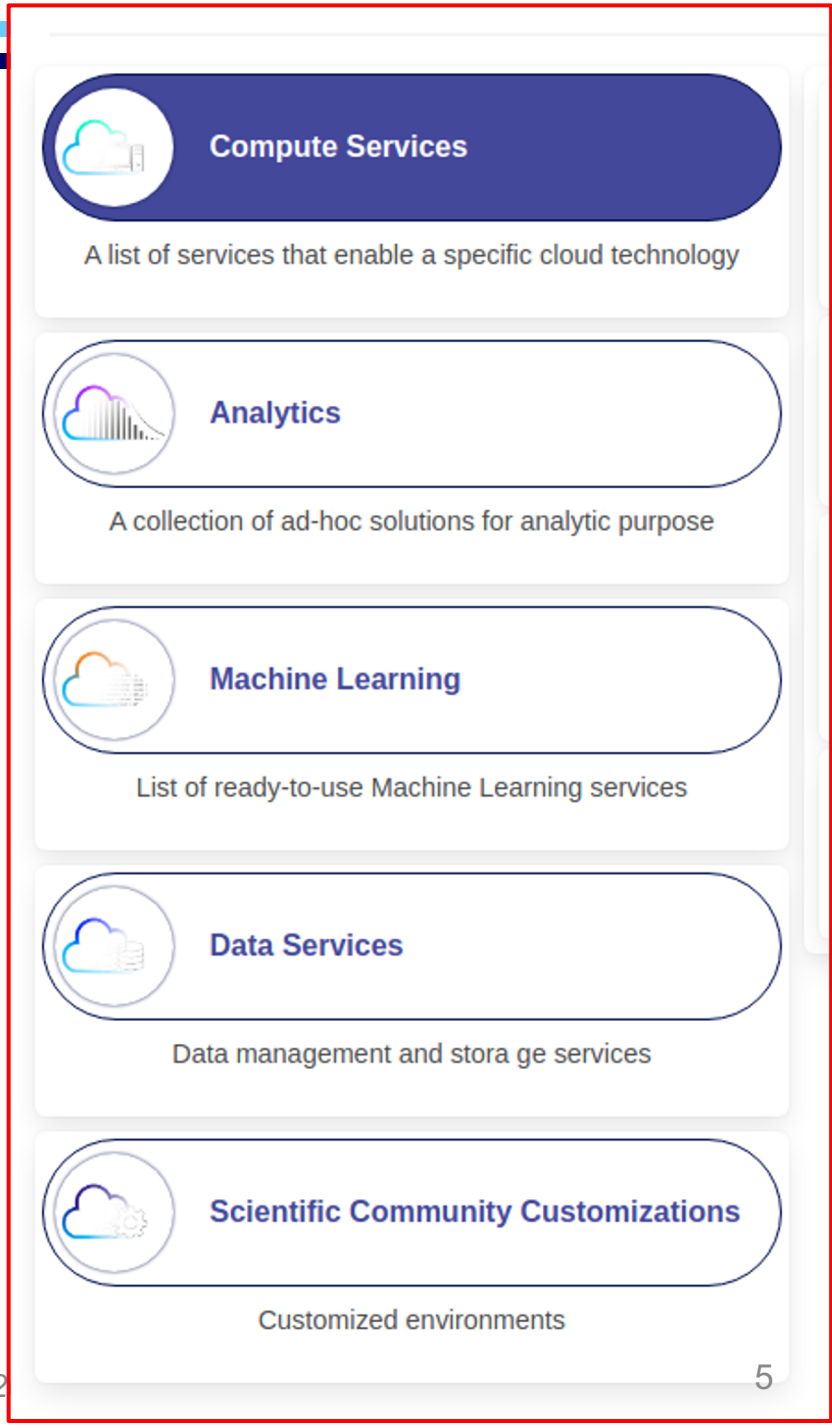
- Spark e/o ElasticSearch e Kibana, R, etc..

Storage solutions: Object storage/posix, possibly connected to high level application layers;

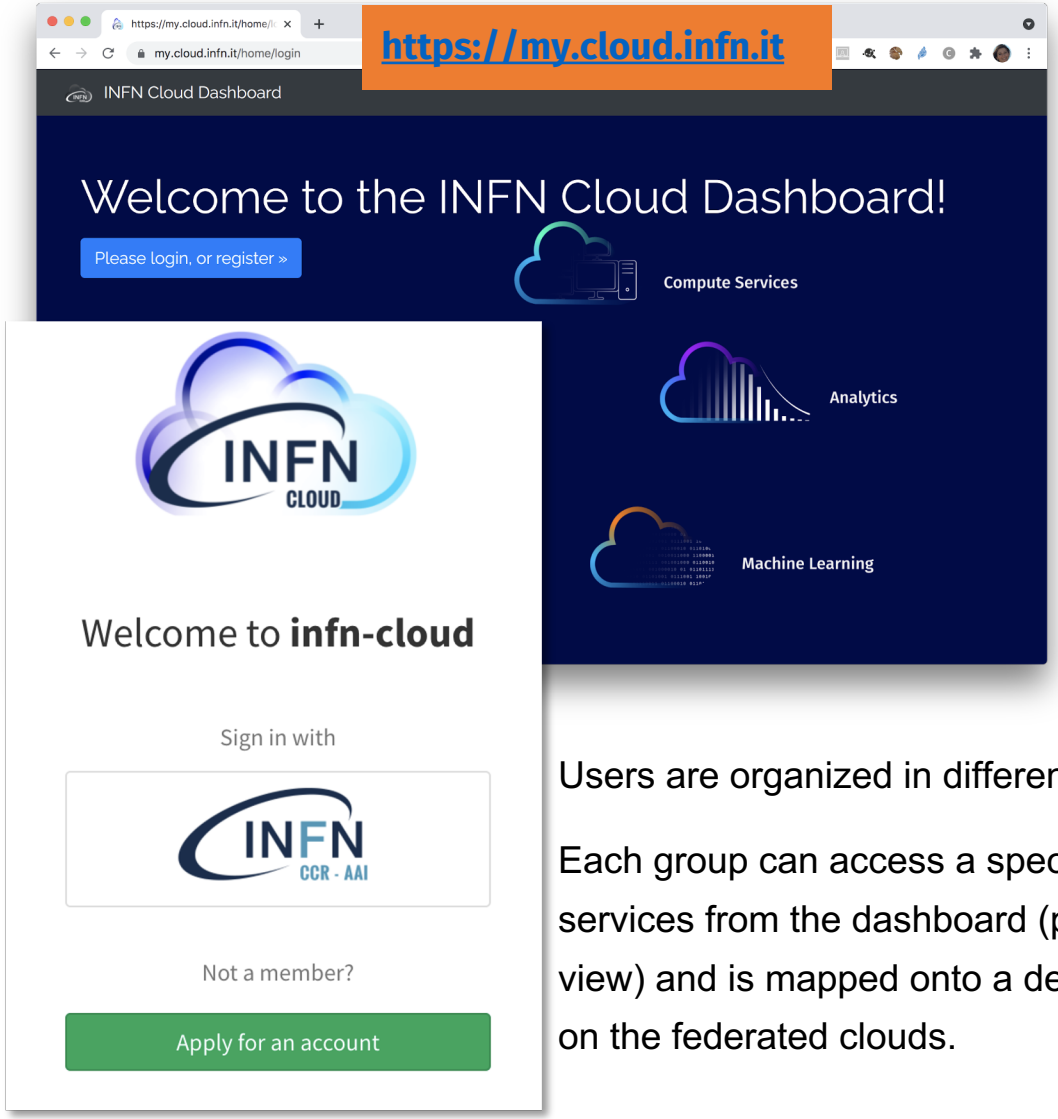
- Jupyter Notebooks with persistent storage (replicated)

Dynamic Clusters even designed and tuned taking into account the specific communities needs;

- HTCondor batch system; environment optimized for ML i.e. equipped with GPUs
- Container orchestrators such as K8s and Mesos

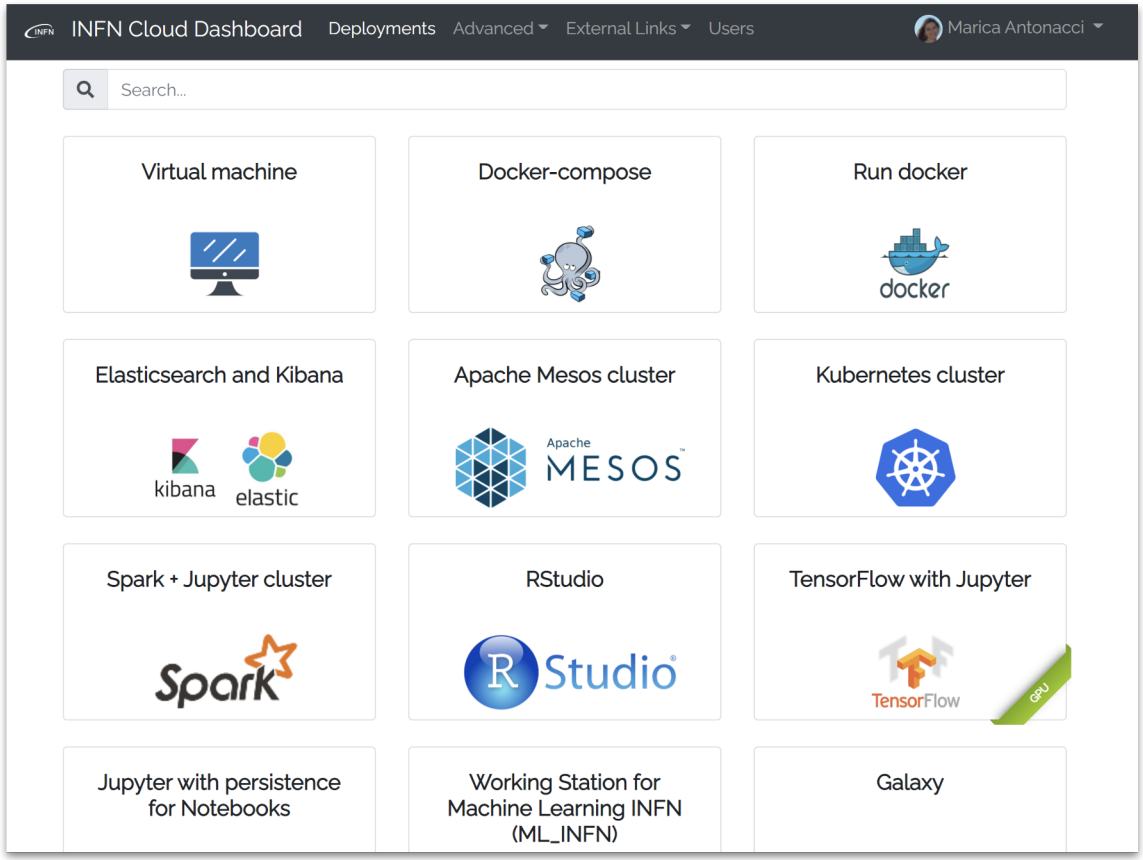


The INFN Cloud Dashboard



https://my.cloud.infn.it

INDIGO IAM manages the authentication/authorization through the whole stack (from PaaS to Iaas)



Users are organized in different IAM **groups**.

Each group can access a specific set of services from the dashboard (personalized view) and is mapped onto a dedicated tenant on the federated clouds.

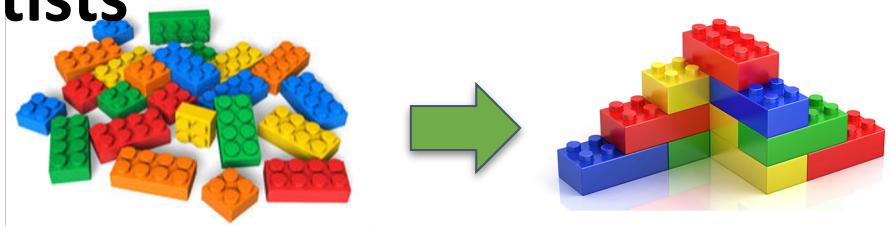
The service catalogue

The catalogue is a graphical representation of the TOSCA templates repository that we have been developing extending the INDIGO-DC custom types

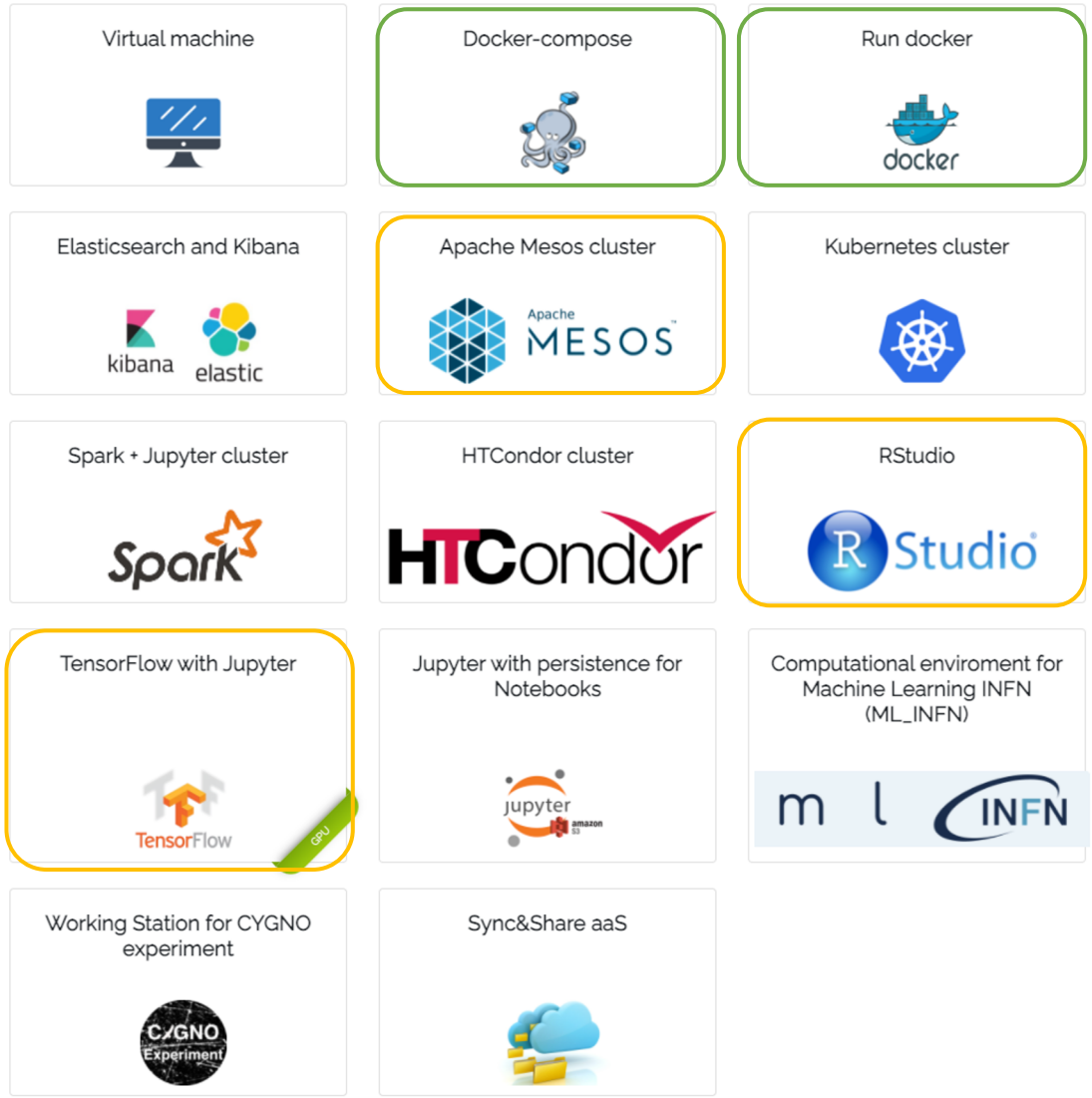
- Each card in the catalogue is associated to one or more templates
- We are following a **lego-like** approach, building on top of reusable components and exploiting the TOSCA service composition pattern

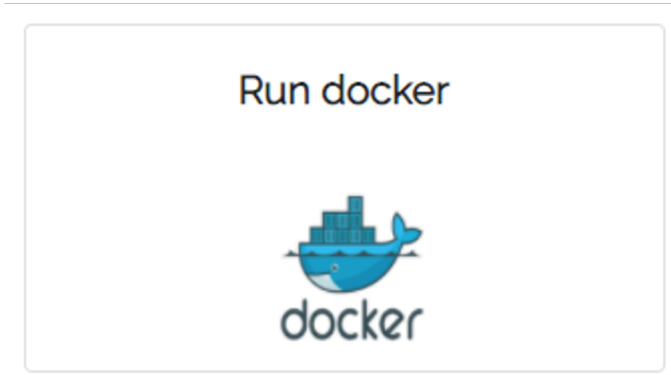
Main objectives:

#1 - build added value services on top of IaaS and PaaS infrastructures
#2 - lower the entry barrier for non-skilled scientists



Docker- & Mesos-based use cases



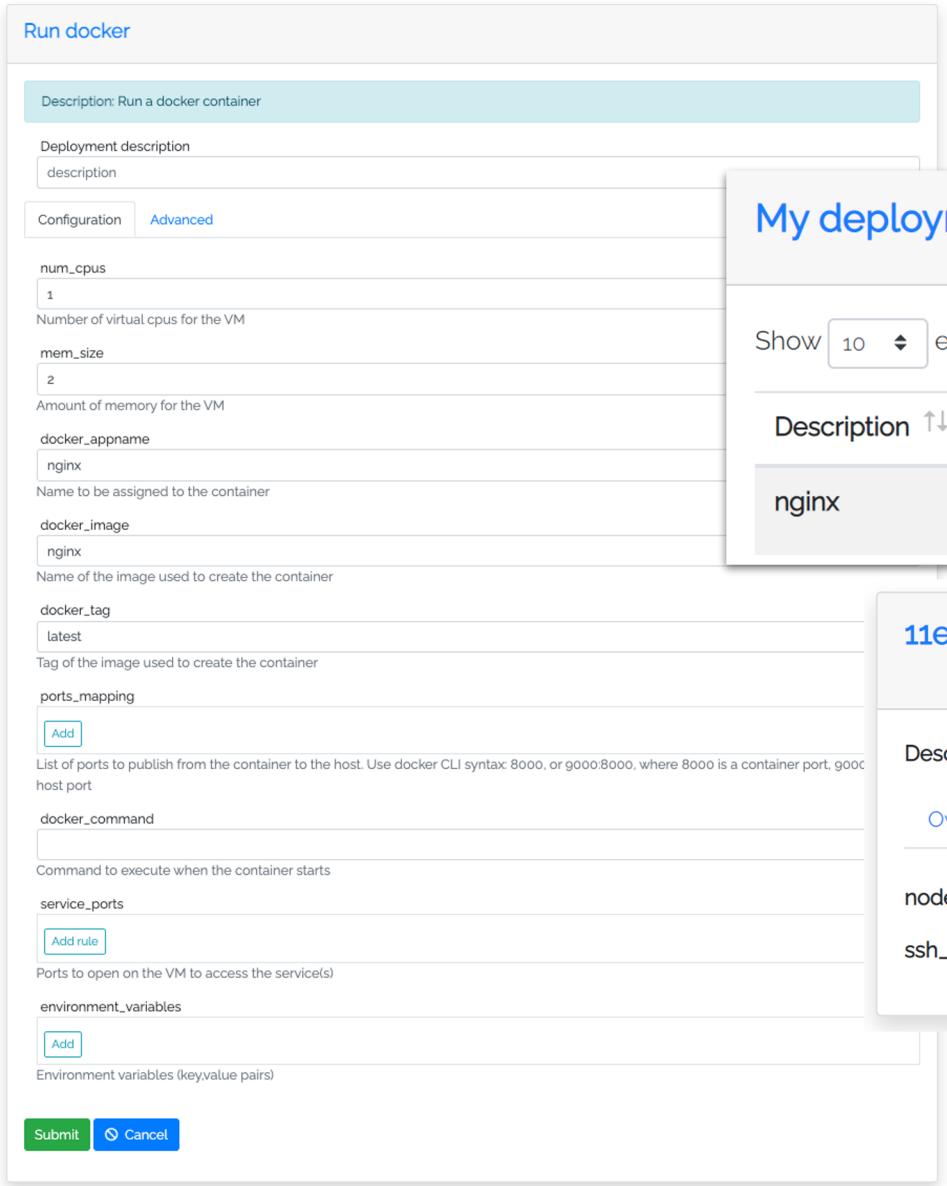


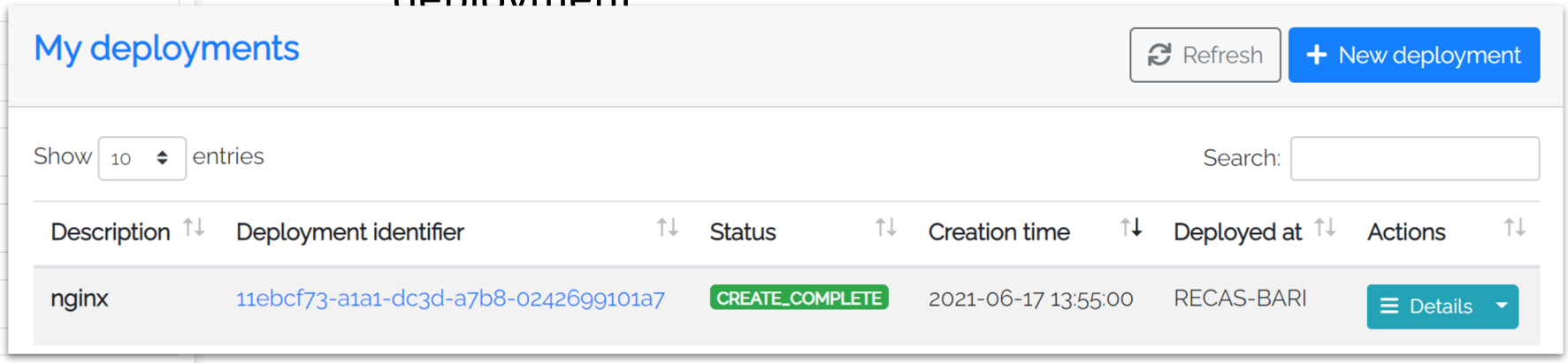
Docker run use-case

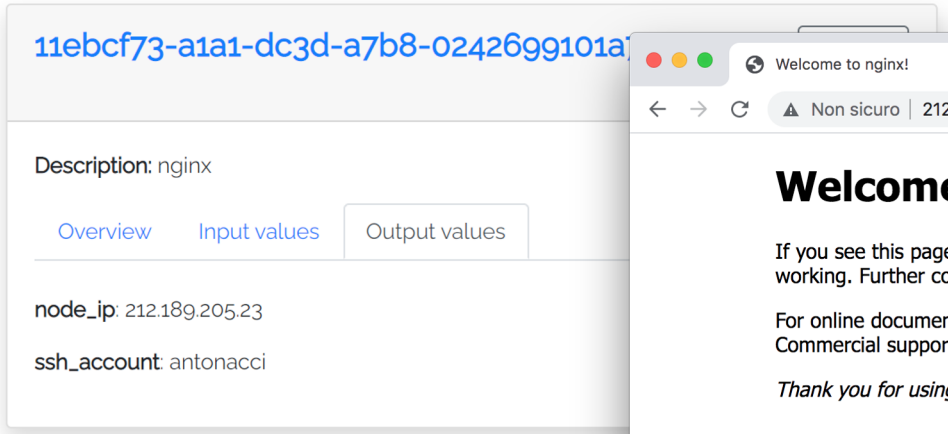
How to run a container on INFN Cloud

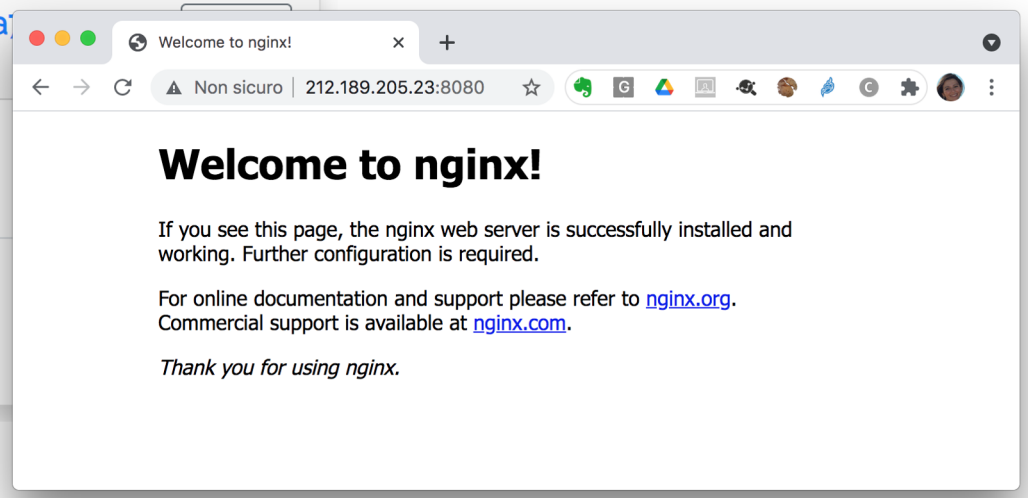
Configure your dockerized service

The configuration form allows you to customize your deployment

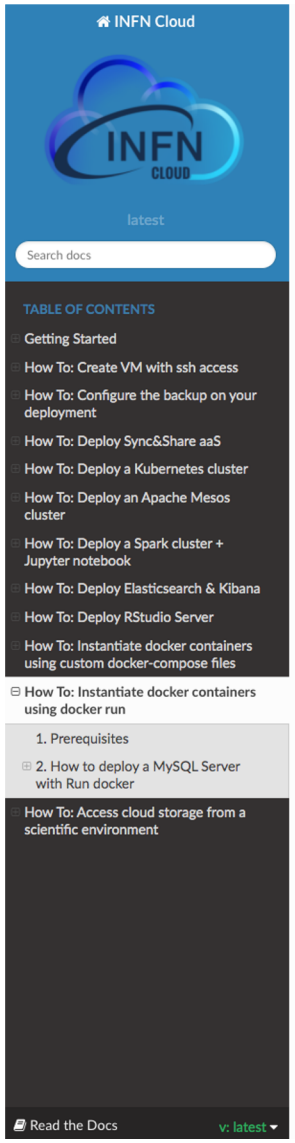


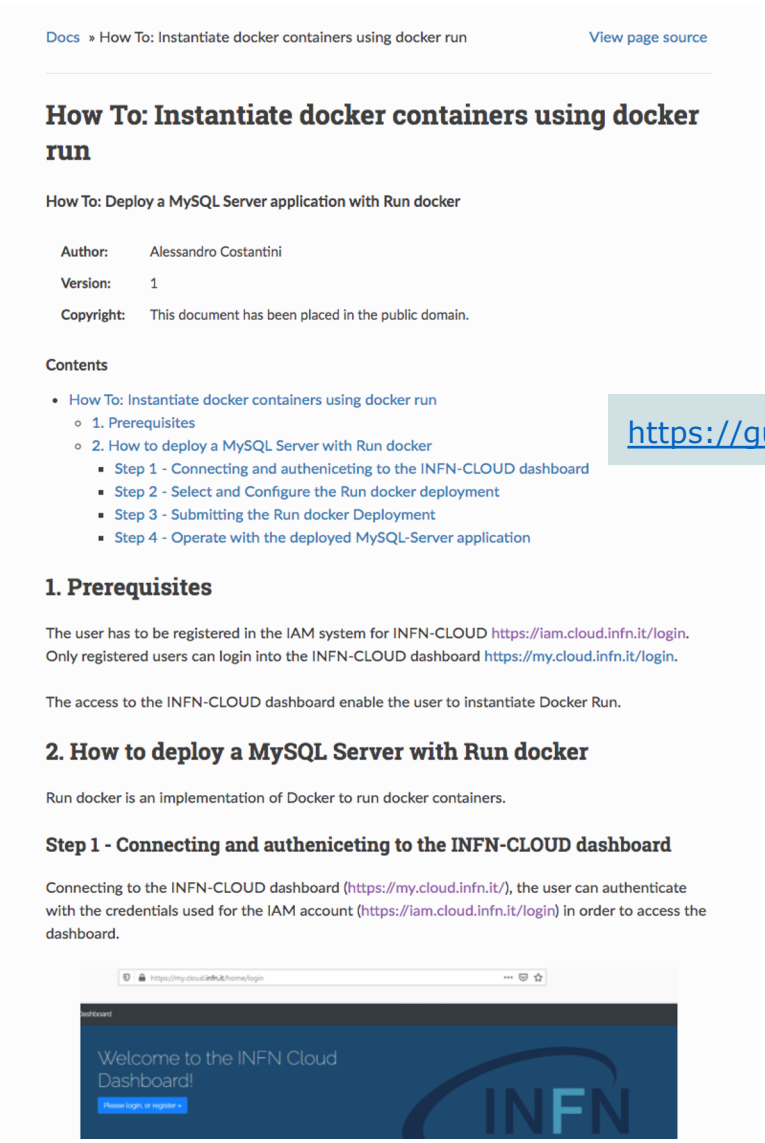




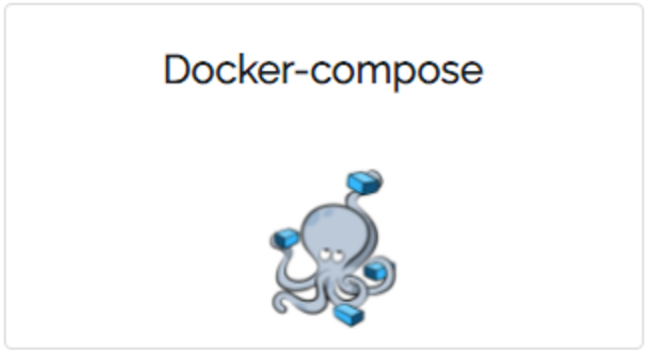


How to su guides.cloud.infn.it





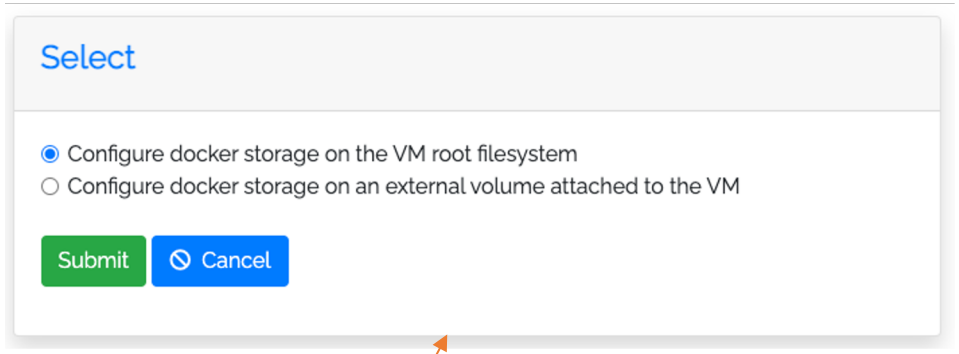
https://guides.cloud.infn.it/docs/users-guides/en/latest/users_guides/howto8.html



Docker-compose use-case

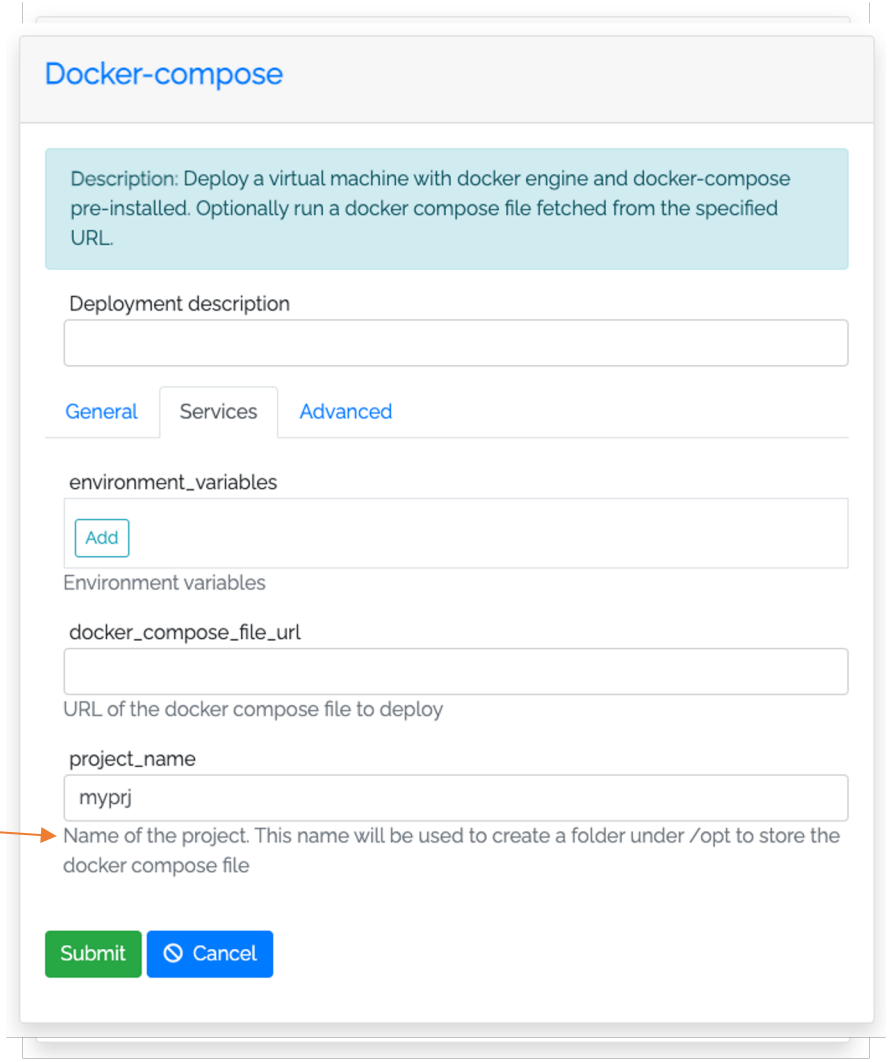
How to run a docker compose file fetched from a given URL

Configure your service

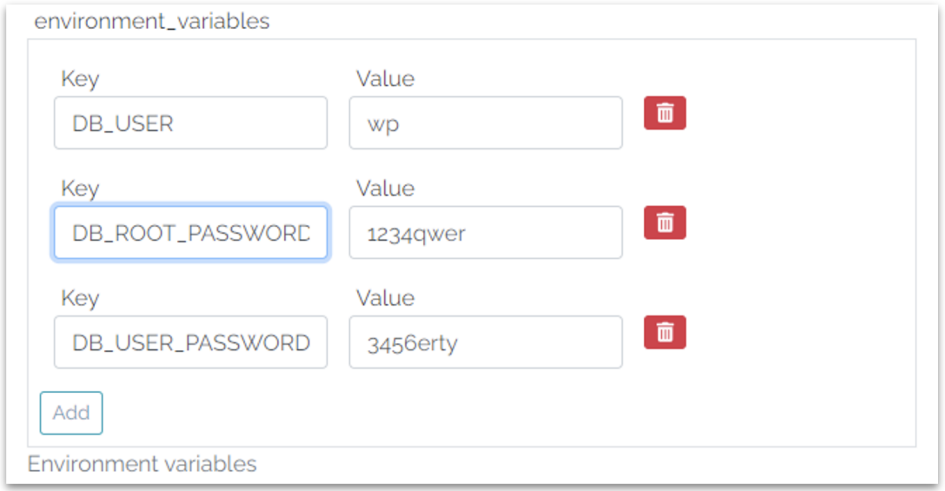


You can choose to

- Put the docker storage on a separate volume
- Configure the machine with only docker and docker-compose or provide a docker compose file URL to start your services



Environment variables management

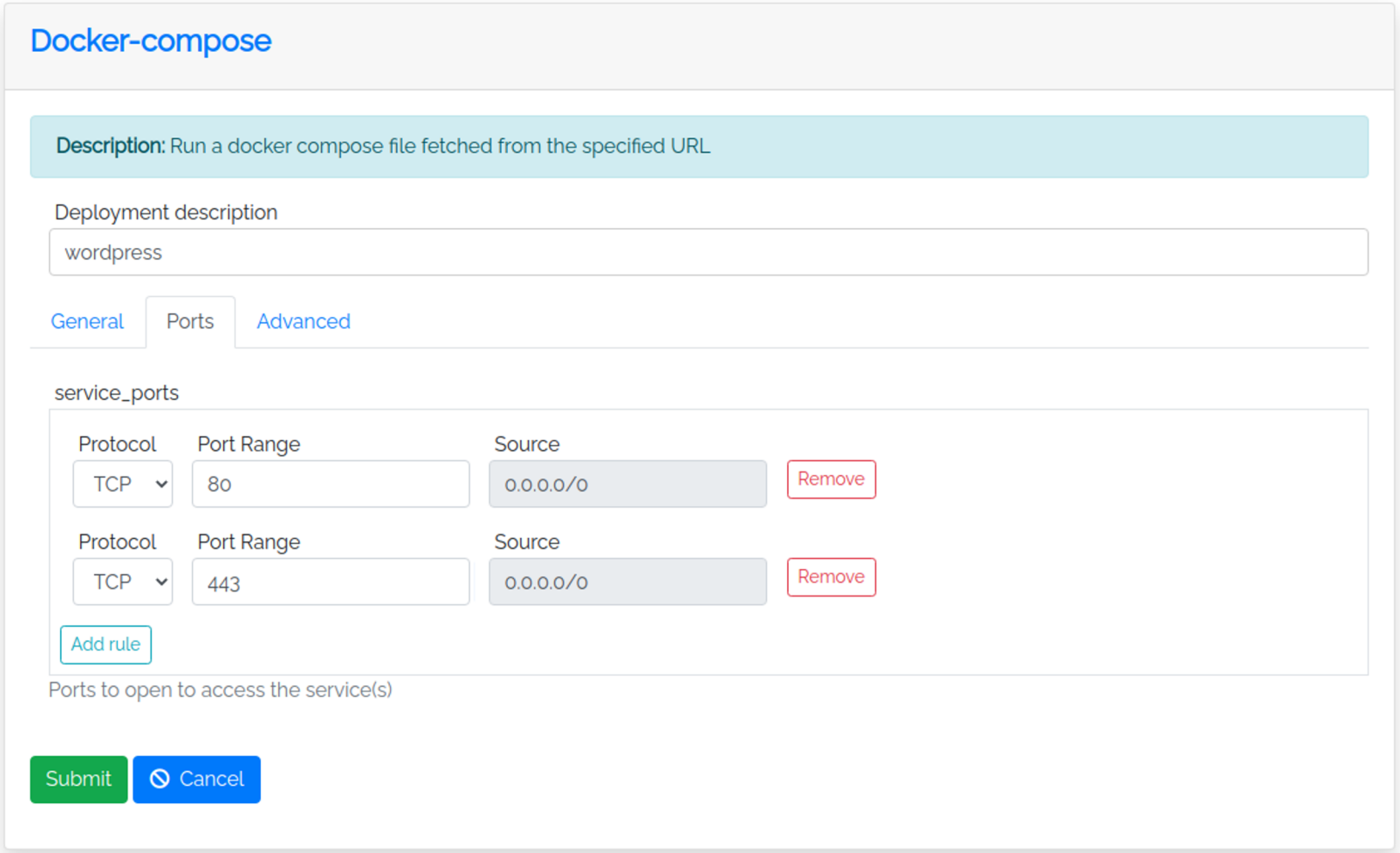


- The special variable *HOST_PUBLIC_IP* is made available by the PaaS system and contains the public IP assigned to the VM
- This env variable can be used as a normal env variable inside the user docker compose file

```
services:
  ........
  app:
    depends_on:
      - db
    image: wordpress
    container_name: app
    volumes:
      - wp-content:/var/www/html/wp-content
    environment:
      - WORDPRESS_DB_HOST=db:3306
      - WORDPRESS_DB_USER=${DB_USER}
      - WORDPRESS_DB_PASSWORD=${DB_USER_PASSWORD}
      - VIRTUAL_HOST=wp.${HOST_PUBLIC_IP}.myip.cloud.infn.it
    expose:
      - 80
```

Ports management

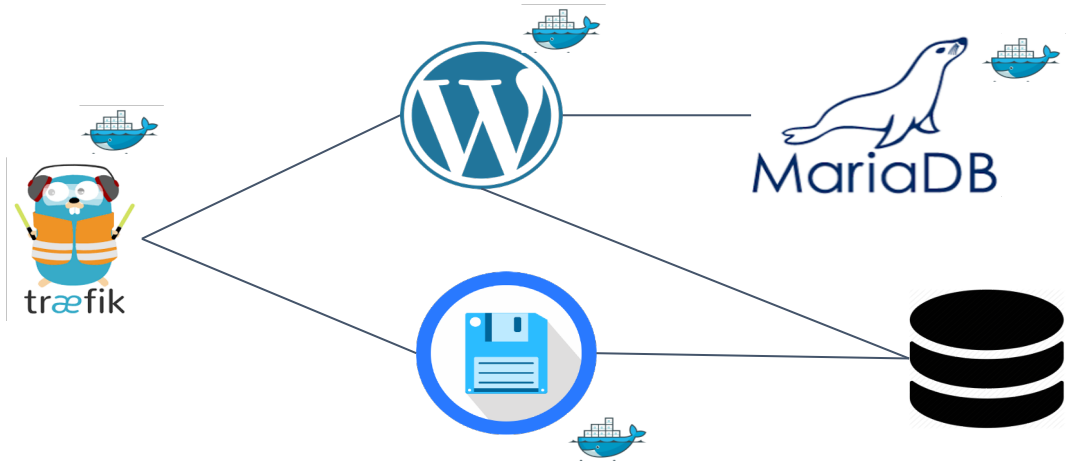
You can define the set of ports that must be automatically opened on the server in order to access your services

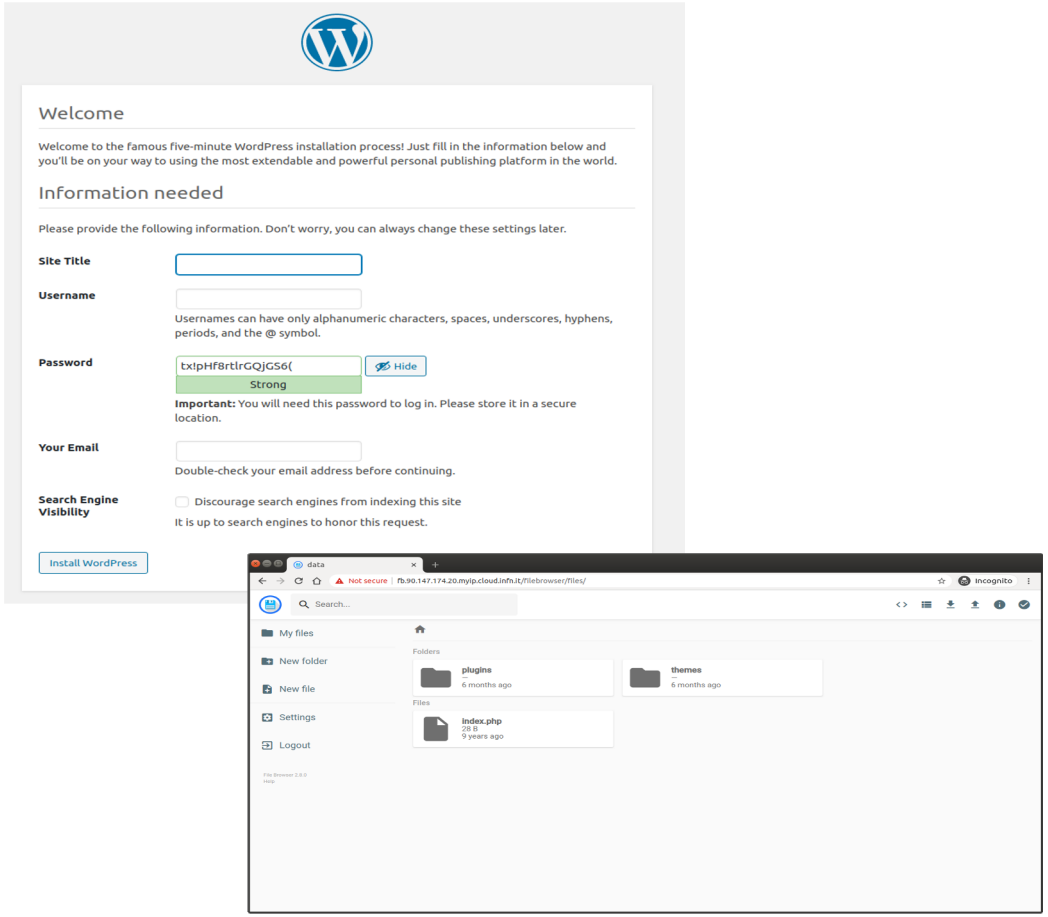


Docker compose example

https://baltig.infn.it/infn-cloud/apps/-/blob/master/compose-example/docker-demo.yaml

Author: Stefano Stalio (LNGS)





DNS @INFN Cloud

INFN Cloud provides a DNSaaS mechanism that associates a DNS name to each VM public IP

$ host **wp.90.147.174.132.myip.cloud.infn.it**
wp.90.147.174.132.myip.cloud.infn.it has address 90.147.174.132

This mechanism is based on xip.io (wildcard DNS) and is exploited for the automatic generation of ssl certificates (e.g. with letsencrypt)

```
services:
 db:
  image: mariadb
  container_name: db
  volumes:
   - db:/var/lib/mysql
  environment:
   - MYSQL_ROOT_PASSWORD=${DB_ROOT_PASSWORD}
   - MYSQL_DATABASE=wordpress
   - MYSQL_USER=${DB_USER}
   - MYSQL_PASSWORD=${DB_USER_PASSWORD}
  expose:
   - 3306
 app:
  depends_on:
   - db
  image: wordpress
  container_name: app
  volumes:
   - wp-content:/var/www/html/wp-content
  environment:
   - WORDPRESS_DB_HOST=db:3306
   - WORDPRESS_DB_USER=${DB_USER}
   - WORDPRESS_DB_PASSWORD=${DB_USER_PASSWORD}
   - VIRTUAL_HOST=wp.${HOST_PUBLIC_IP}.myip.cloud.infn.it
  expose:
   - 80
```

SSL Terminator & Load-balancer

- You can use Traefik as load balancer and SSL terminator. https://traefik.io/traefik/
- Traefik is able to renew letsencrypt certificates

```
services:
  load_balancer:
    image: traefik
    container_name: traefik
    volumes:
      - letsencrypt:/letsencrypt
      - /var/run/docker.sock:/var/run/docker.sock:ro
    ports:
      - "80:80"
      - "443:443"
    command:
      - "--api.insecure=true"
      - "--providers.docker=true"
      - "--providers.docker.exposedbydefault=false"
      - "--entrypoints.web.address=:80"
      - "--entrypoints.websecure.address=:443"
      - "--certificatesresolvers.myhttpchallenge.acme.httpchallenge=true"
      - "--
certificatesresolvers.myhttpchallenge.acme.httpchallenge.entrypoint=web
"
      - "--
certificatesresolvers.myhttpchallenge.acme.email=${CONTACT_EMAIL}"
      - "--
certificatesresolvers.myhttpchallenge.acme.storage=/letsencrypt/acme.js
on"
```

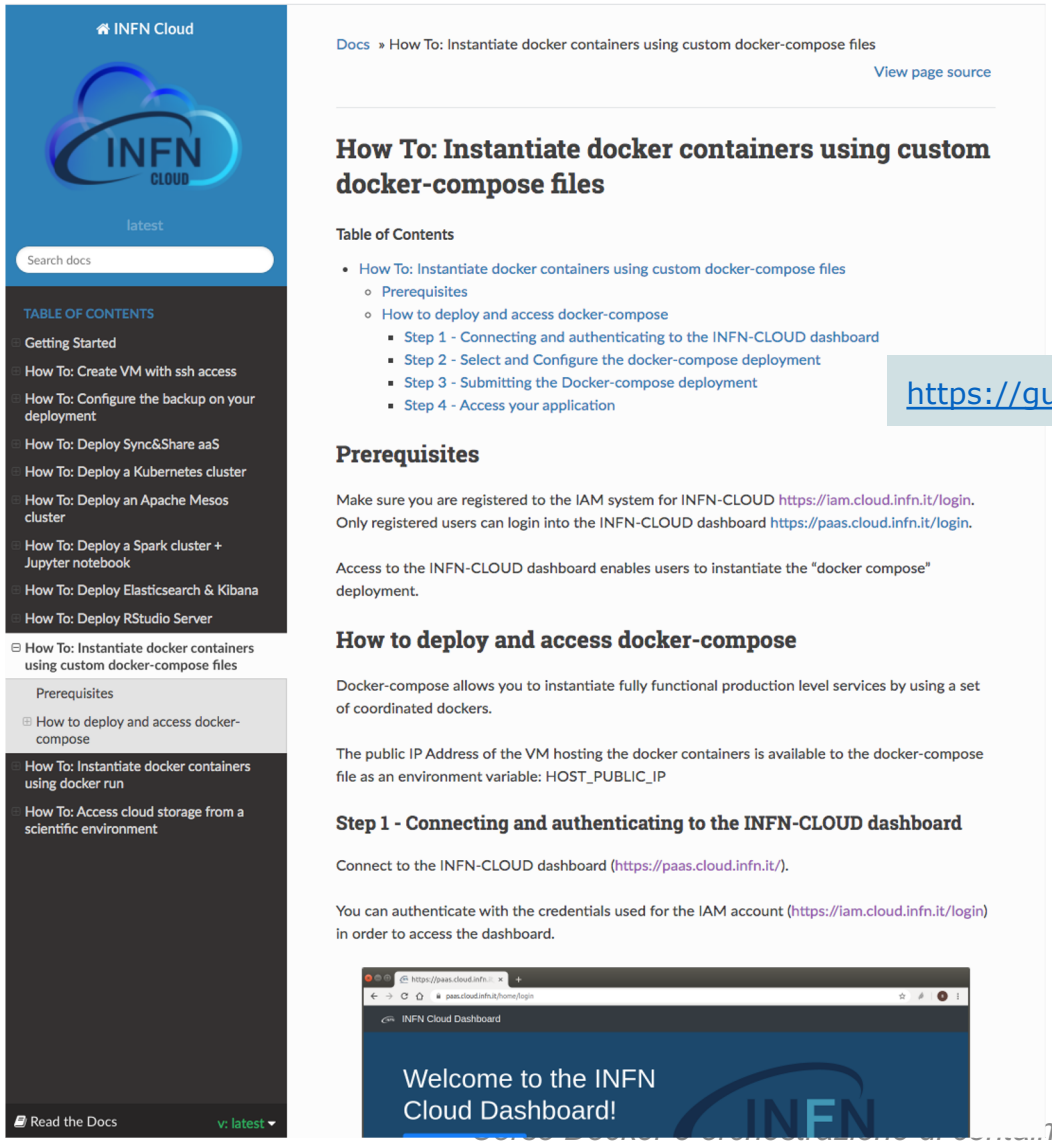
Traefik configuration

Traefik is automatically configured through the labels* exposed by the containers

(*) "A label is a **key=value** *pair that applies metadata to a container."*

```
services:
  app:
   depends_on:
     - db
   image: wordpress
   container_name: app
   volumes:
     - wp-content:/var/www/html/wp-content
   environment:
     - WORDPRESS_DB_HOST=db:3306
     - WORDPRESS_DB_USER=${DB_USER}
     - WORDPRESS_DB_PASSWORD=${DB_USER_PASSWORD}
     - VIRTUAL_HOST=wp.${HOST_PUBLIC_IP}.myip.cloud.infn.it
   expose:
     - 80
   labels:
     - "traefik.enable=true"
     - "traefik.http.middlewares.app-redirect-ssl.redirectscheme.scheme=https"
     - "traefik.http.routers.app-nossl.middlewares=app-redirect-ssl"
     - "traefik.http.routers.app-
nossl.rule=Host(`wp.${HOST_PUBLIC_IP}.myip.cloud.infn.it`)"
     - "traefik.http.routers.app-nossl.entrypoints=web"
     -
"traefik.http.routers.app.rule=Host(`wp.${HOST_PUBLIC_IP}.myip.cloud.infn.it`)"
     - "traefik.http.routers.app.entrypoints=websecure"
     - "traefik.http.routers.app.tls.certresolver=myhttpchallenge"
     - "traefik.http.routers.app.tls=true"
```

How to su guides.cloud.infn.it

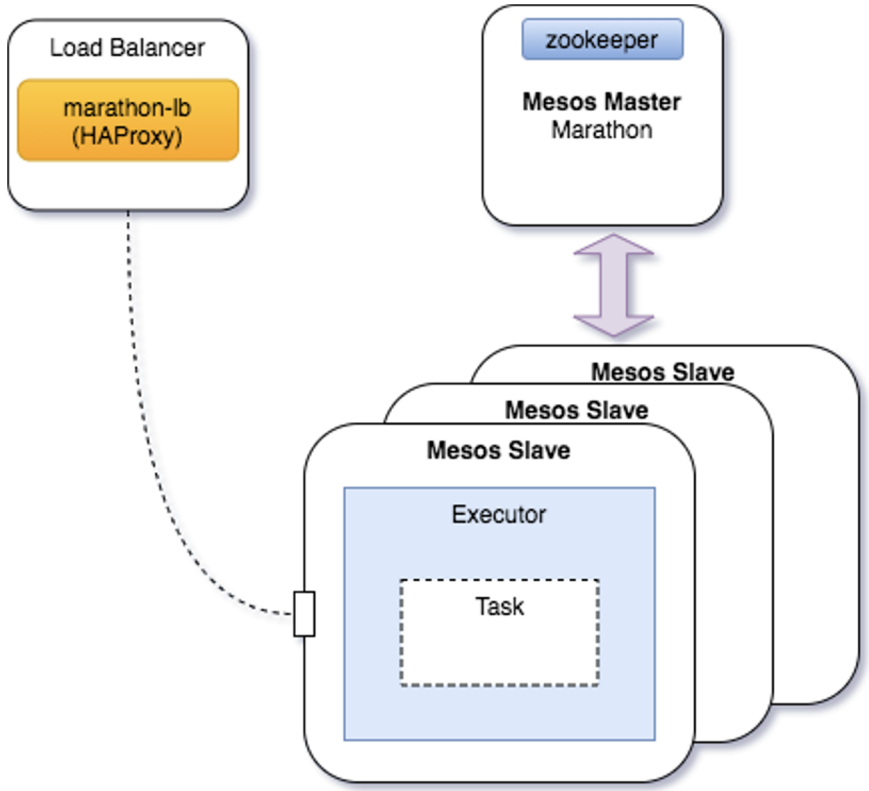


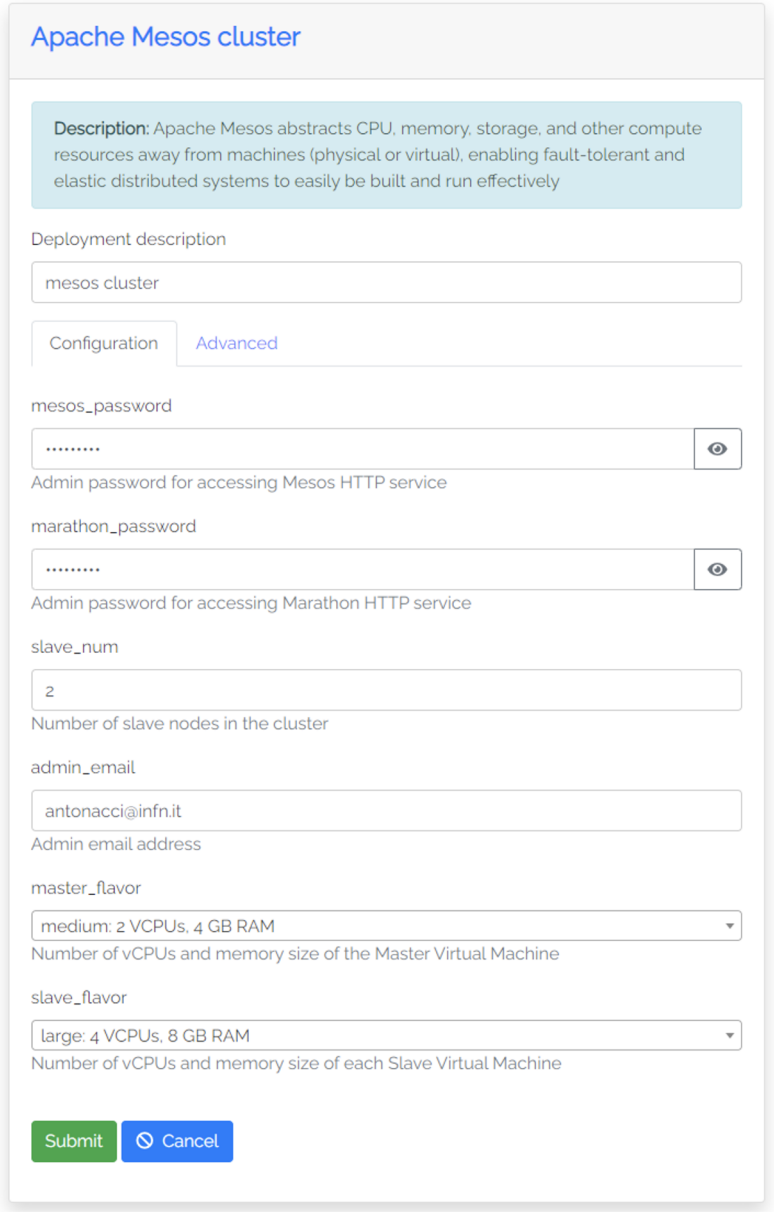
https://guides.cloud.infn.it/docs/users-guides/en/latest/users_guides/howto7.html

Mesos use-case

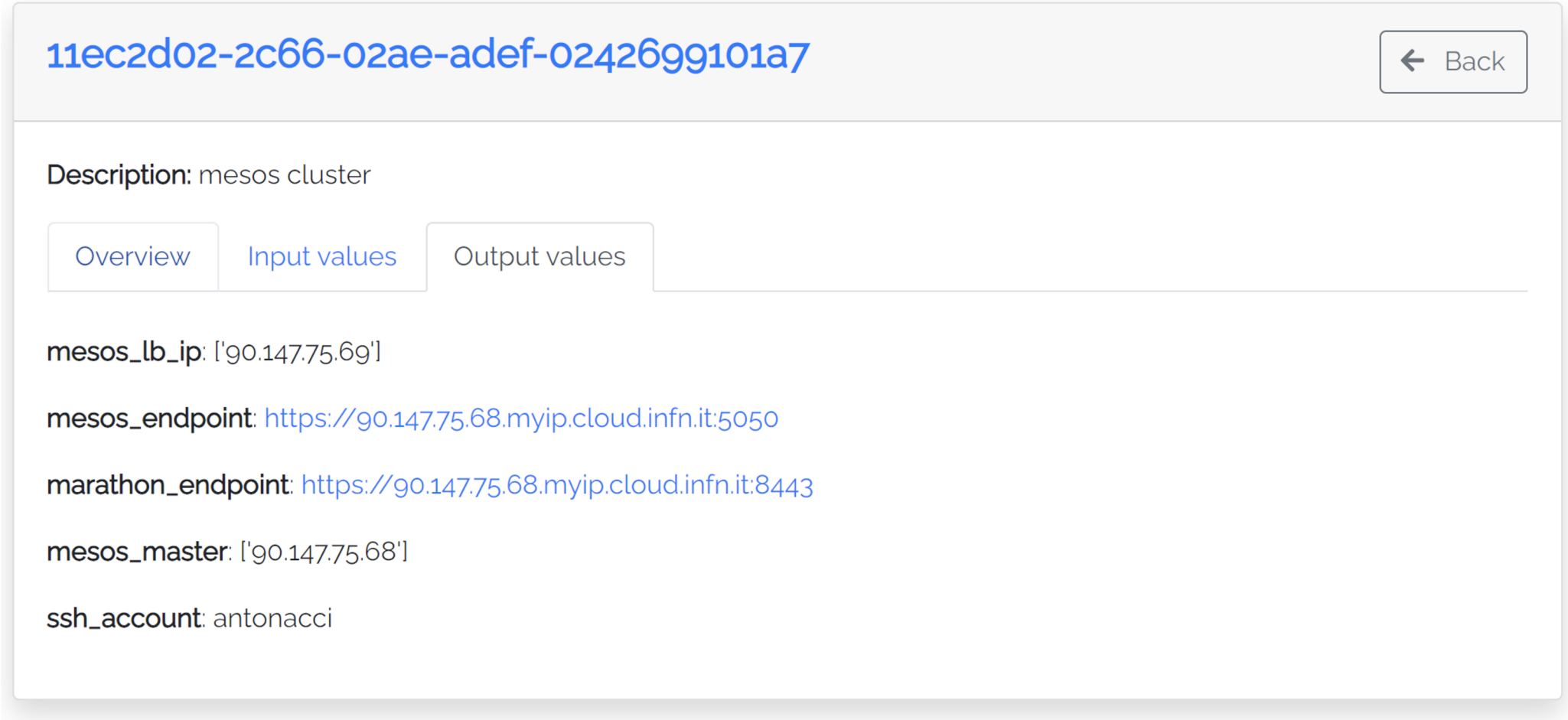
How to deploy a complete Mesos cluster

Cluster architecture

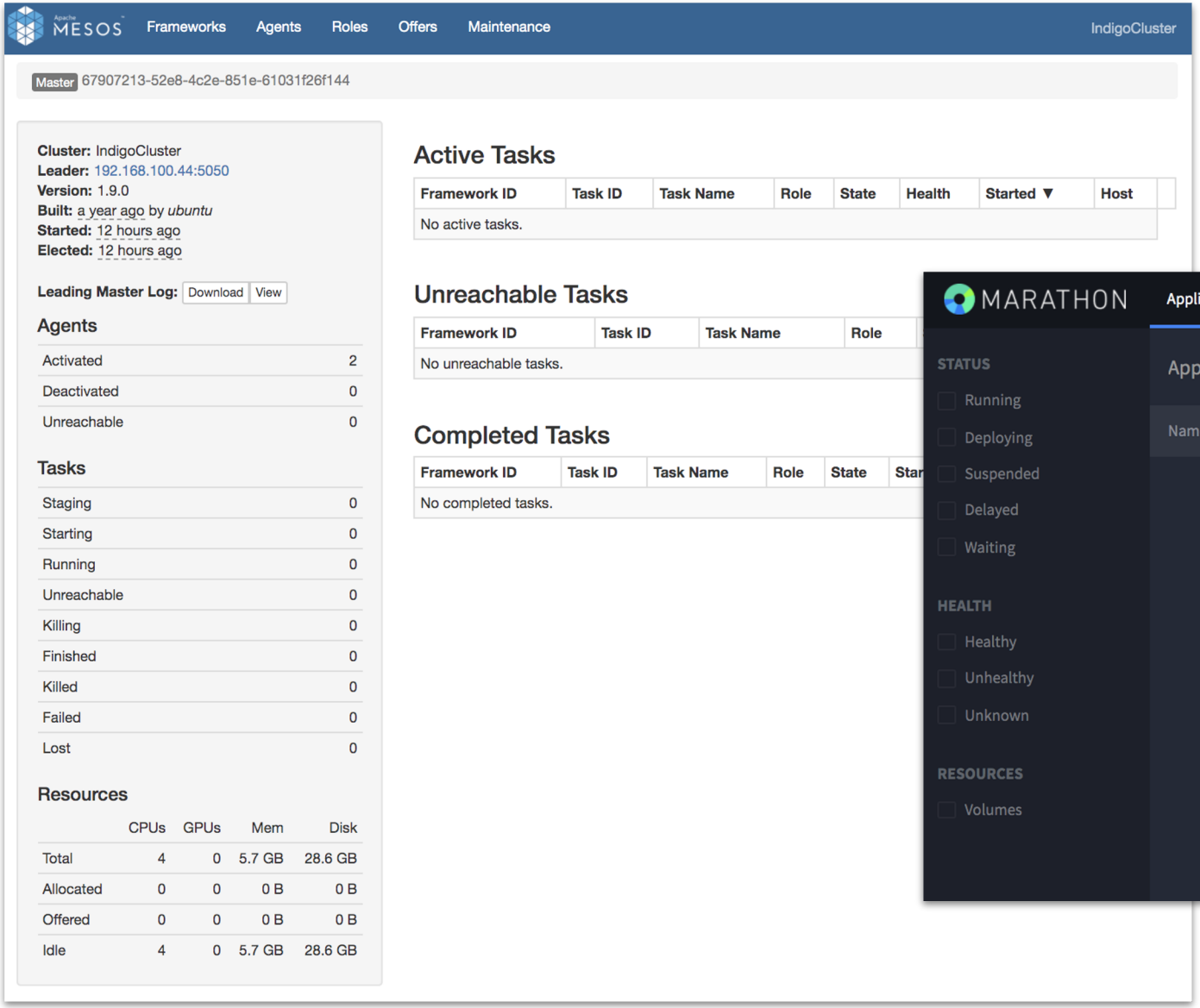


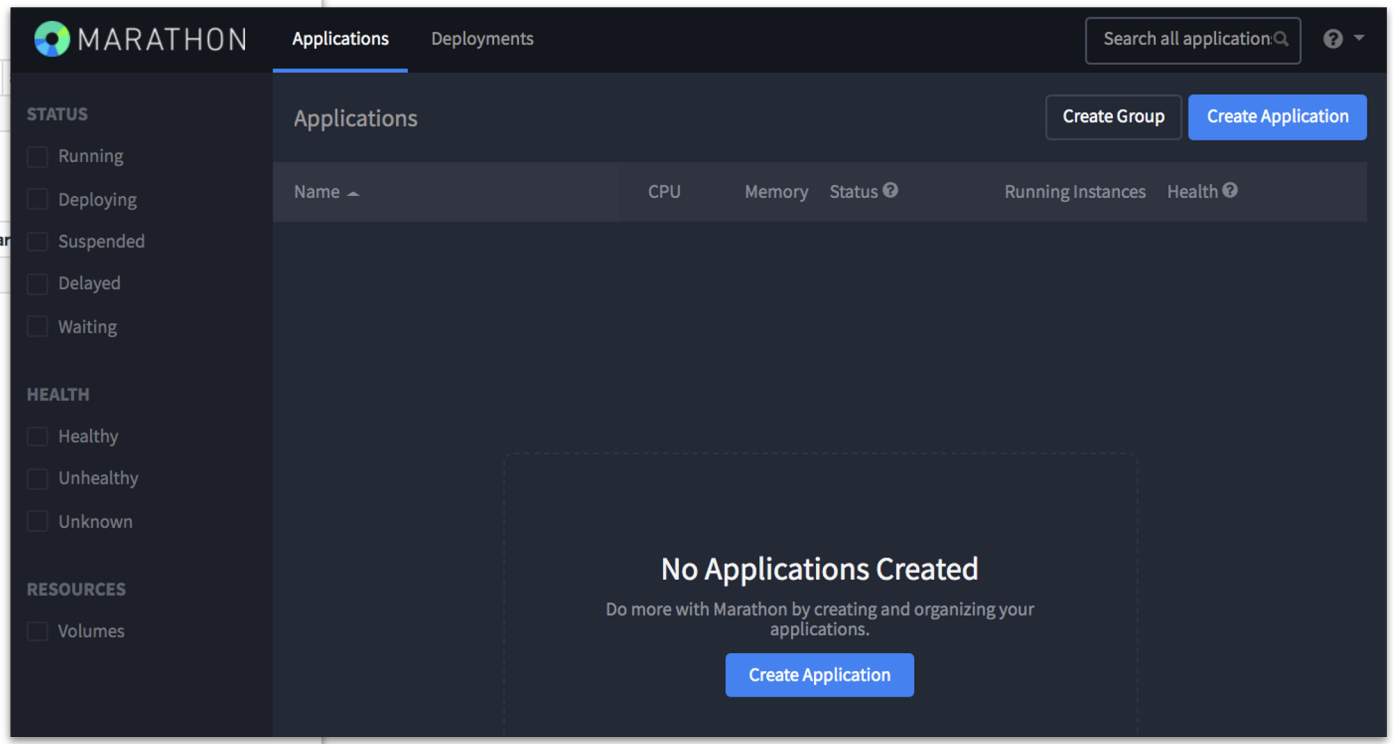


Deployment outputs

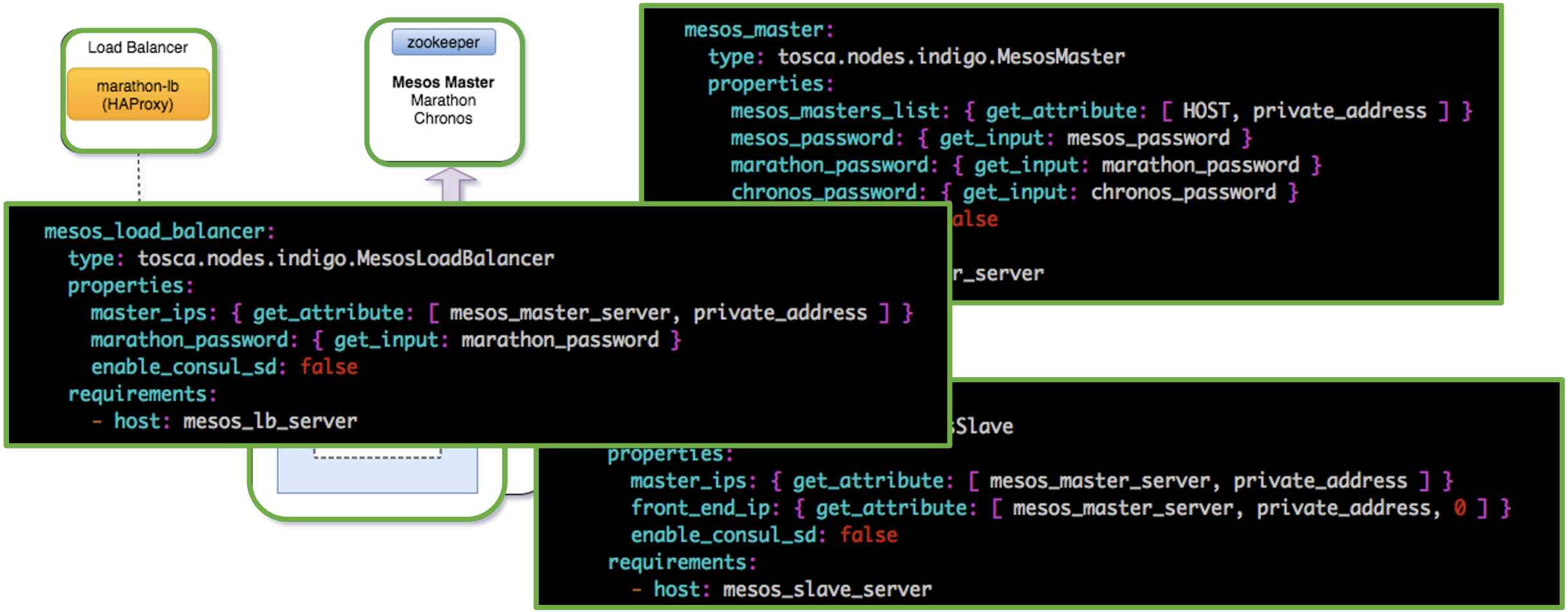


Mesos/Marathon





TOSCA Template: nodes



https://baltig.infn.it/infn-cloud/tosca-templates/-/blob/master/mesos/mesos_cluster.yaml

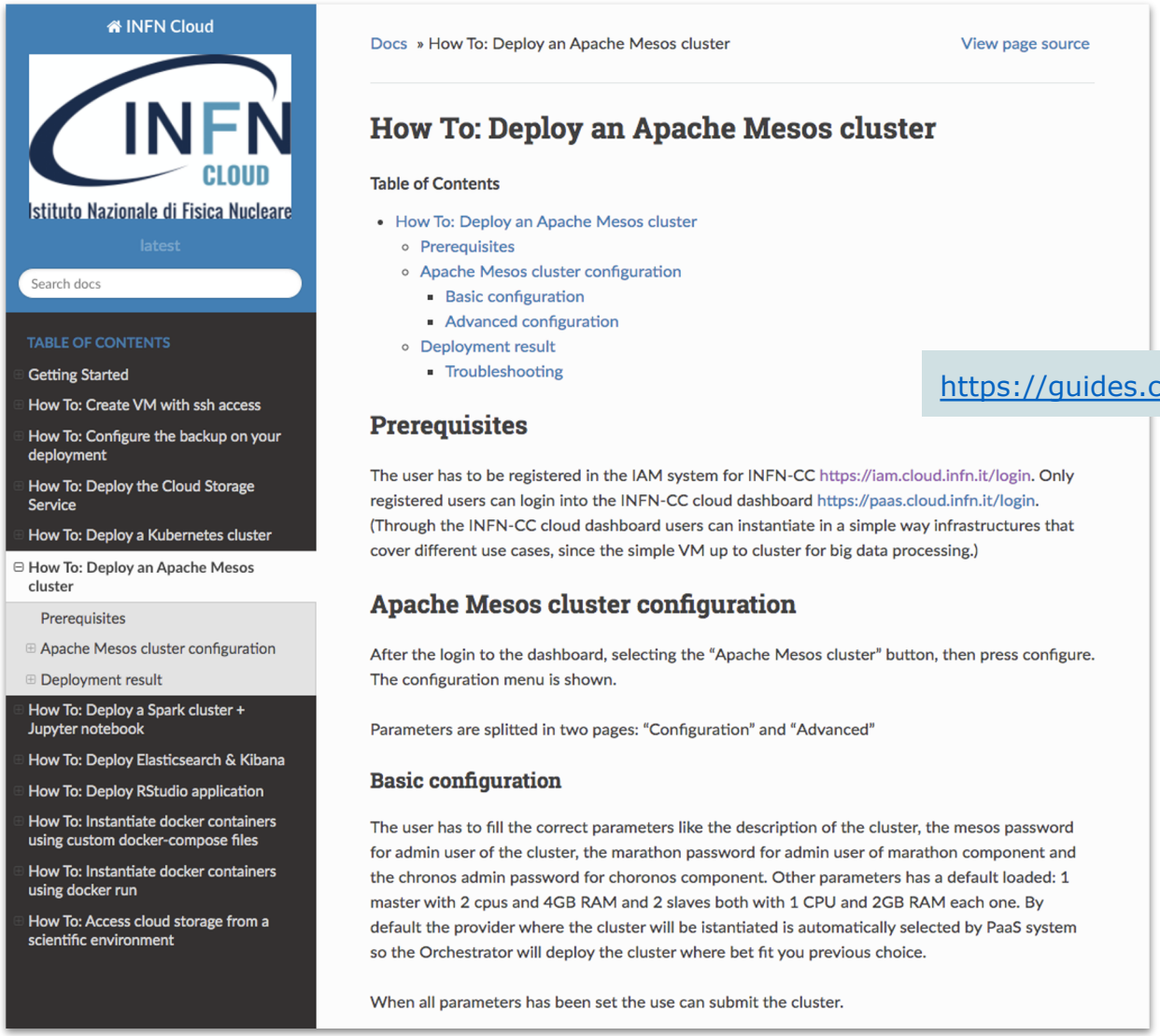
TOSCA artifacts: ruoli ansible

I principali ruoli utilizzati per configurare il cluster sono:

- indigo-dc.docker
- indigo-dc.mesos (master/slave)
- indigo-dc.marathon
- indigo-dc.marathon-lb

Sono tutti pubblicati sull'hub **Ansible Galaxy**
https://galaxy.ansible.com/indigo-dc

How to @ guides.cloud.infn.it



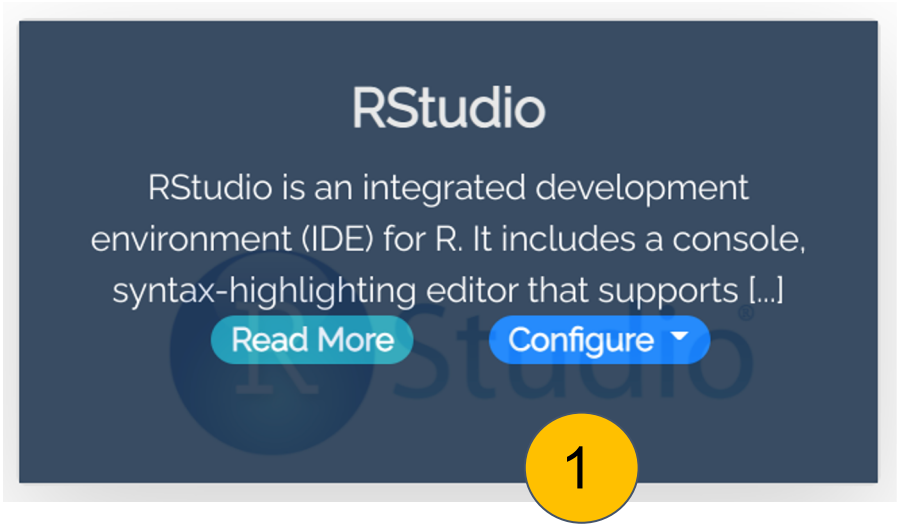
https://guides.cloud.infn.it/docs/users-guides/en/latest/users_guides/howto3.html

WILL BE AVAILABLE SOON

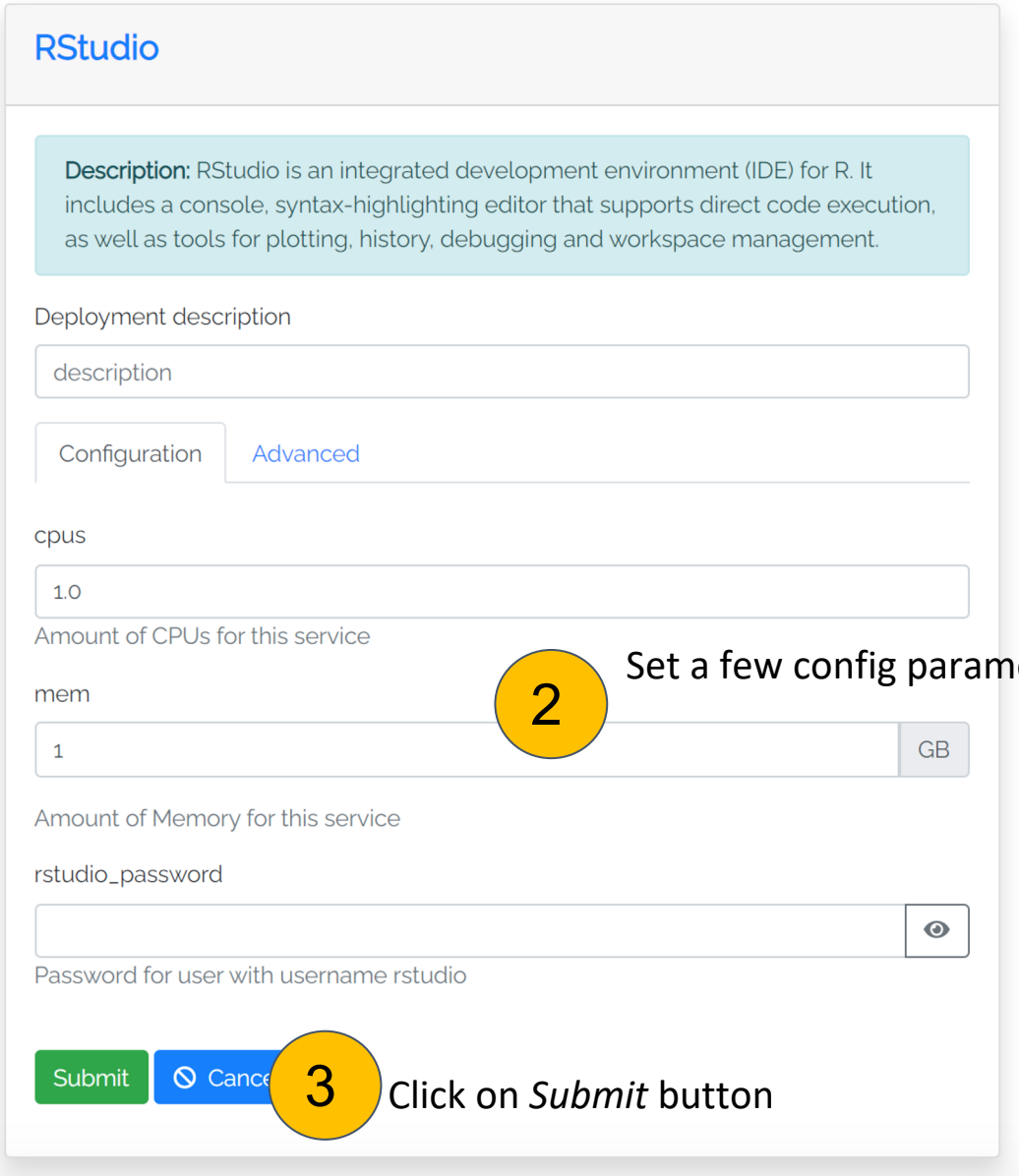
RStudio use-case

How to start an integrated development environment (IDE) for R programming

Configure the service



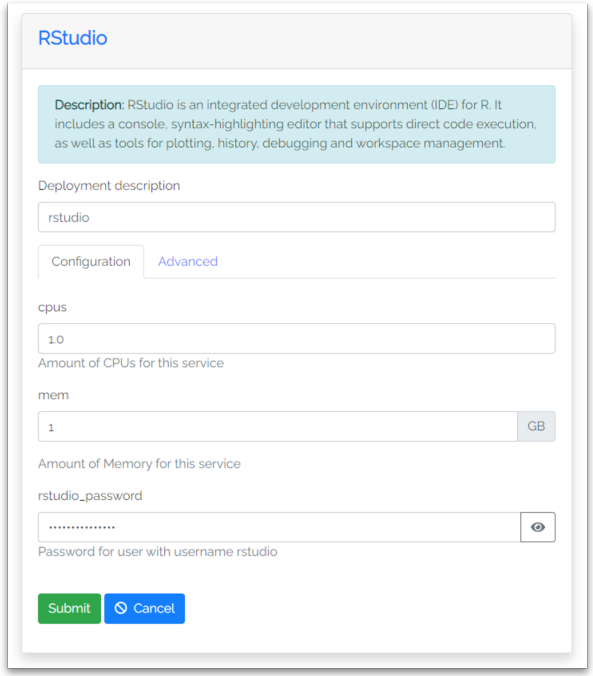
Click on *Configure* button

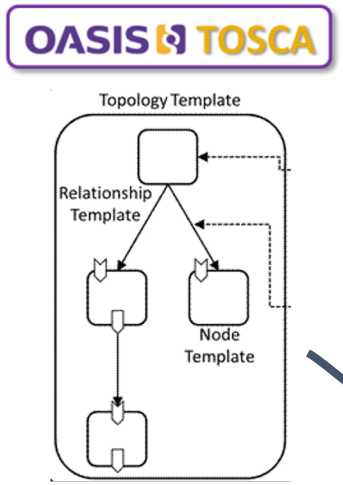


Set a few config parameters

Click on *Submit* button

Under the hood





1 A TOSCA template is used to render the config form (inputs)

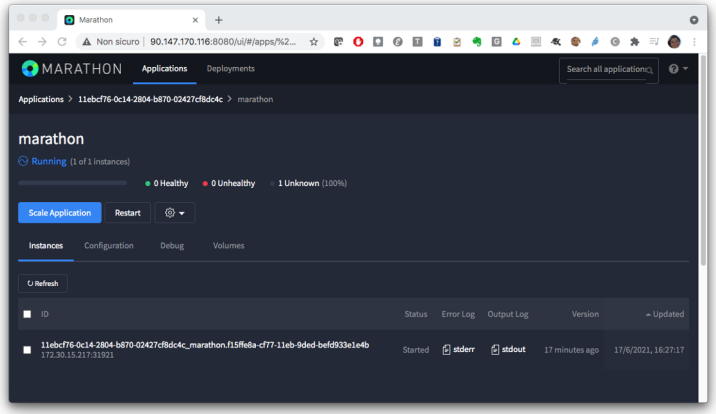
2 The template is sent to the PaaS Orchestrator

INDIGO PaaS Orchestrator

The template is translated into the Marathon application definition (json)

3

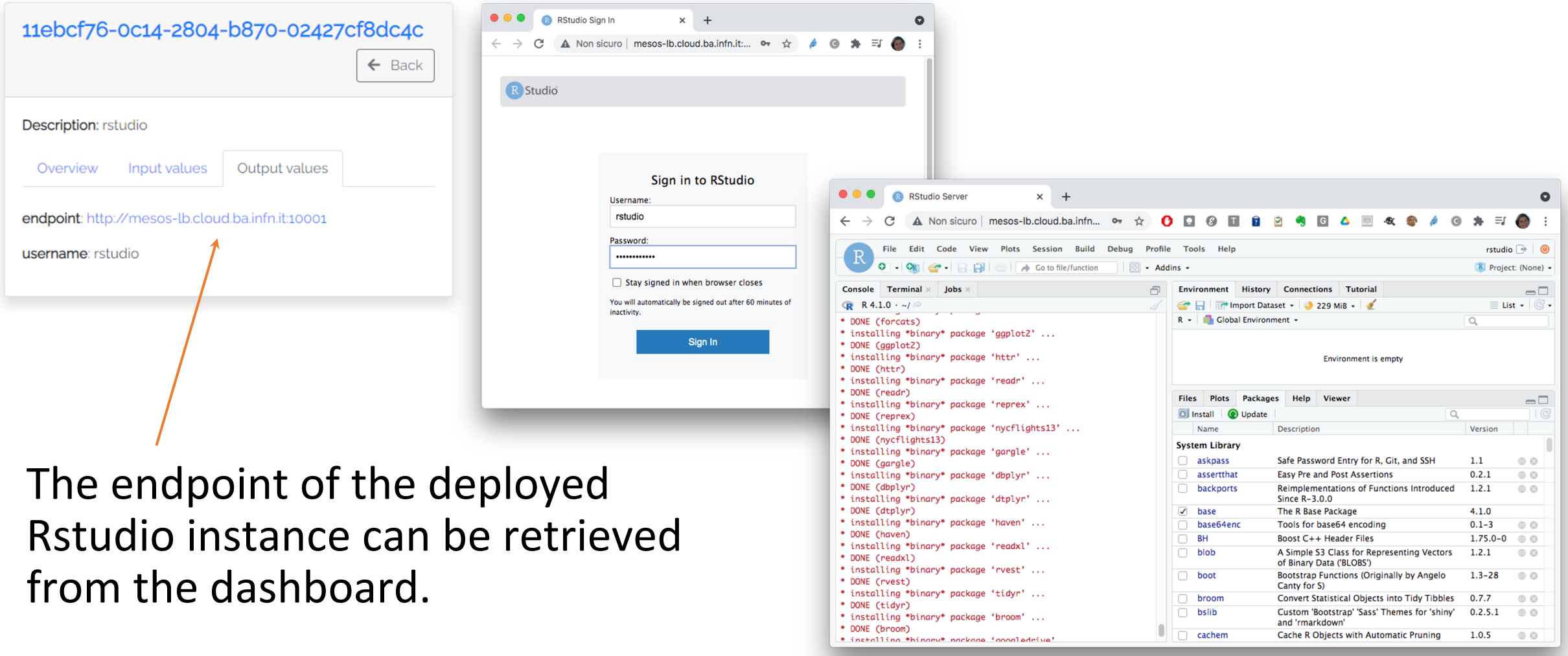
4 The request is submitted to Marathon/Mesos cluster



From TOSCA to Marathon App definition in json

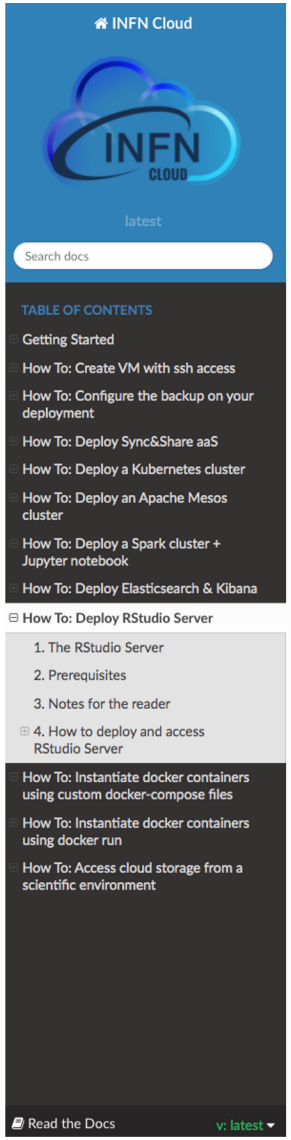
```yaml
topology_template:

  inputs:

    cpus:
      type: float
      description: Amount of CPUs for this service
      required: yes
      default: 1.0

    mem:
      type: scalar-unit.size
      description: Amount of Memory for this service
      required: yes
      default: 1 GB

    rstudio_password:
      type: string
      description: Password for user with username rstudio
      required: yes

  node_templates:

    marathon:
      type: tosca.nodes.indigo.Container.Application.Docker.Marathon
      properties:
        environment_variables:
          PASSWORD: { get_input: rstudio_password }
        uris: []
      artifacts:
        image:
          file: rocker/rstudio
          type: tosca.artifacts.Deployment.Image.Container.Docker
      requirements:
        - host: docker_runtime

    docker_runtime:
      type: tosca.nodes.indigo.Container.Runtime.Docker
      capabilities:
        host:
          properties:
            num_cpus: { get_input: cpus}
            mem_size: { get_input: mem}
            publish_ports:
              - protocol: tcp
                source: 8787
            volumes: [ { concat: [ 'marathon:', '/home/rstudio' , ':rw:dvdi:rexray'] } ]
```

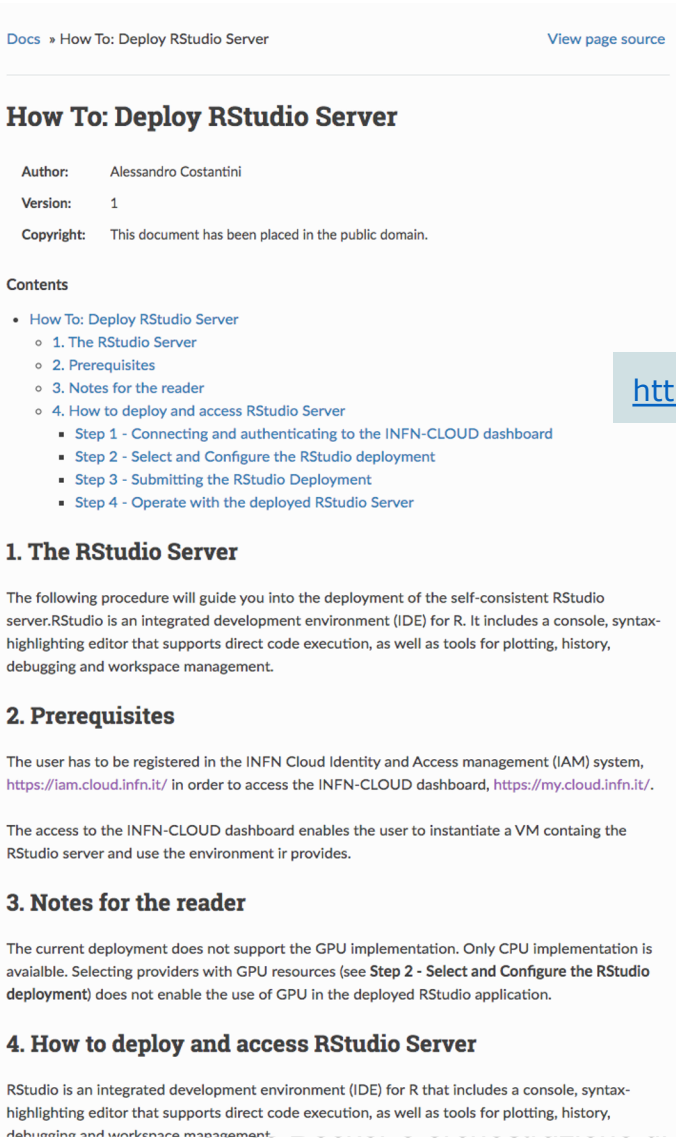
Access your environment



The endpoint of the deployed Rstudio instance can be retrieved from the dashboard.

How to @ guides.cloud.infn.it





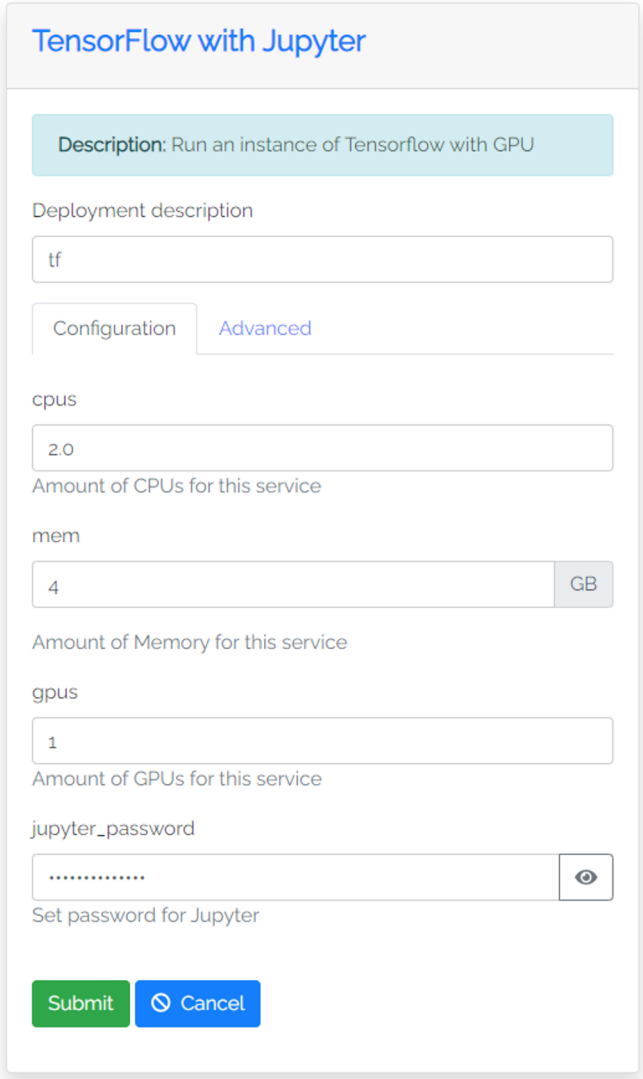
https://guides.cloud.infn.it/docs/users-guides/en/latest/users_guides/howto6.html

WILL BE AVAILABLE SOON

Tensorflow with GPU(s) use-case

How to start a Tensorflow container using GPU and with Jupyter access

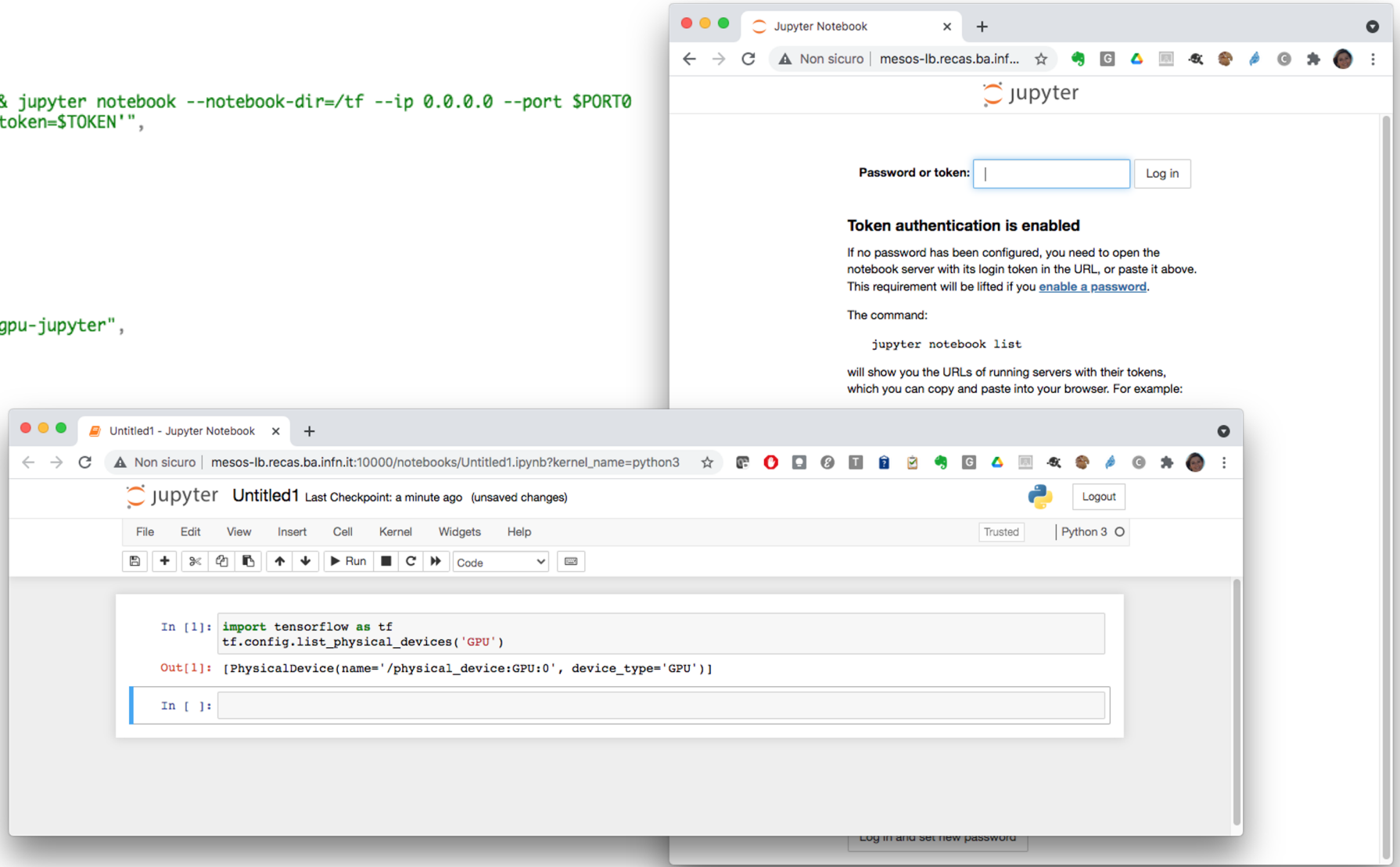
Configure the service




```
node_templates:

  tensorflow:
    type: tosca.nodes.indigo.Container.Application.Docker.Marathon
    properties:
      environment_variables:
        TOKEN: { get_input: jupyter_password }
      uris: []
      command: "bash -c 'source /etc/bash.bashrc && jupyter notebook --notebook-dir=/tf --ip 0.0.0
          .0 --port $PORT0 --no-browser --allow-root --NotebookApp.token=$TOKEN'"
    artifacts:
      image:
        file: tensorflow/tensorflow:2.4.2-gpu-jupyter
        type: tosca.artifacts.Deployment.Image.Container.Docker
    requirements:
      - host: docker_runtime

  docker_runtime:
    type: tosca.nodes.indigo.Container.Runtime.Docker
    capabilities:
      host:
        properties:
          num_cpus: { get_input: cpus}
          mem_size: { get_input: mem}
          num_gpus: { get_input: gpus}
          publish_ports:
             - protocol: tcp
               source: 8888
          volumes: [ { concat: [ 'tensorflow:', '/tf/notebooks' , ':rw:dvdi:rexray'] } ]
```

GPU-powered Jupyter Notebook

```
{
  "id": "tensorflowapp",
  "cmd": "bash -c 'source /etc/bash.bashrc && jupyter notebook --notebook-dir=/tf --ip 0.0.0.0 --port $PORT0
    --no-browser --allow-root --NotebookApp.token=$TOKEN'",
  "instances": 1,
  "cpus": 2,
  "mem": 4096,
  "gpus": 1,
  "uris": [],
  "constraints": [],
  "container": {
    "type": "MESOS",
    "docker": {
      "image": "tensorflow/tensorflow:2.4.2-gpu-jupyter",
      "privileged": false
    },
    "volumes": []
  },
  "env": {},
  "labels": {
    "HAPROXY_GROUP": "external"
  },
  "portDefinitions": [
    {
      "port": 0,
      "protocol": "tcp",
      "labels": {}
    }
  ],
  "networks": [
    {
      "mode": "HOST",
      "labels": {}
    }
  ]
}
```



Conclusions

The goal of INFN Cloud is to provide end-users with compute and storage services by offering

- a **portfolio of technical solutions** already developed but extensible – continuously evolving following a **user driven development approach**
- technical support for the end user applications migration to a cloud-based environment
- **transparent** solutions hiding the resources allocation complexity in a **federation of distributed clouds**

The high-level services shown in this presentation are part of the current portfolio:

- They provide a simple way to run docker containers on cloud resources
- Further (more complex) services have been built starting from these building blocks
- More advanced use-cases are shown in the next presentation.

Thank you

for your attention!

www.cloud.infn.it

For general communications email us at **cloud@lists.infn.it**

To ask for support write to our mailing list **cloud-support@infn.it**, integrated with our ServiceDesk