INFN CLOUD DASHBOARD OVERVIEW
AND
SERVICE IMPLEMENTATION STRATEGY

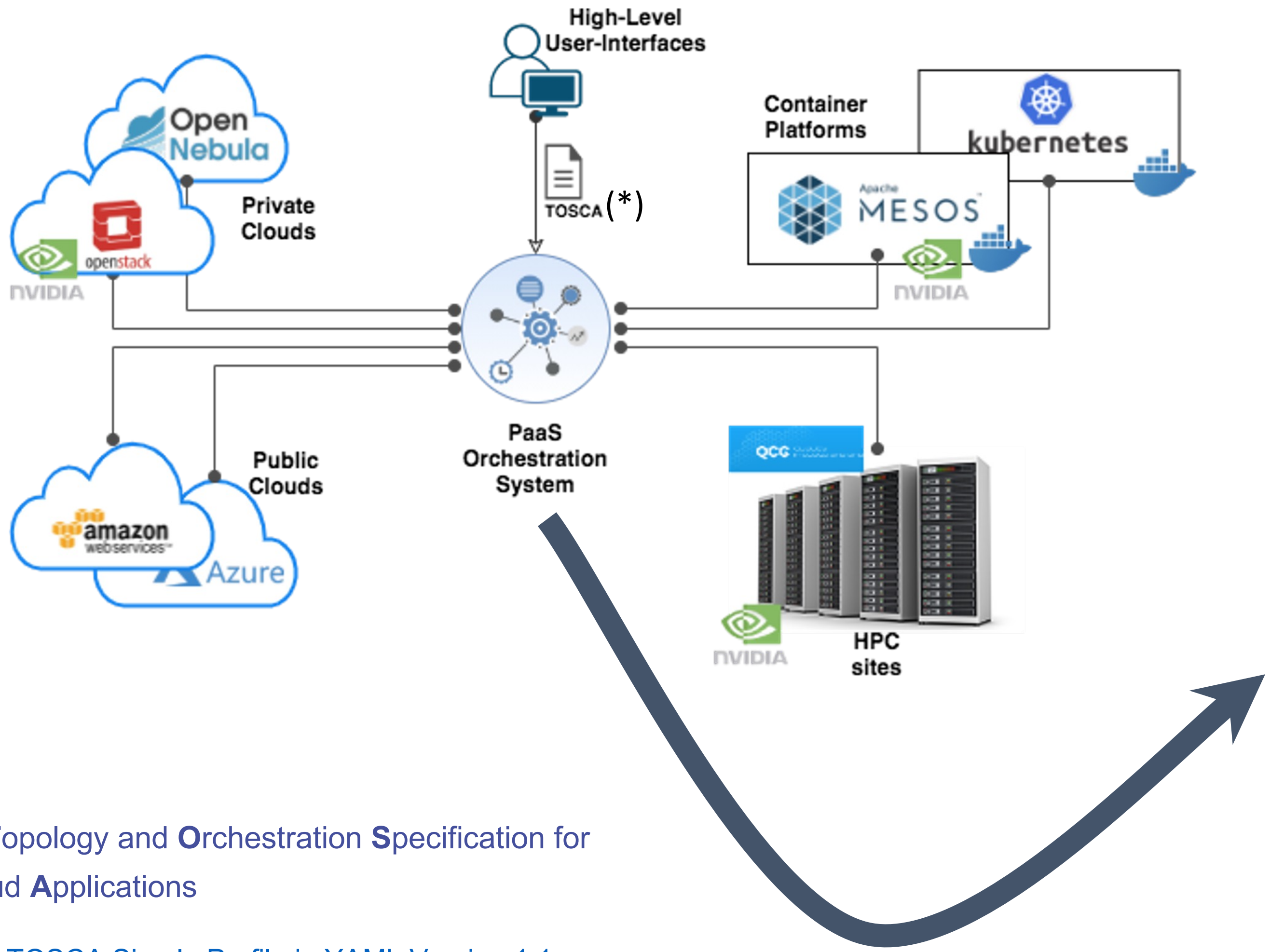
Marica Antonacci (INFN BA)

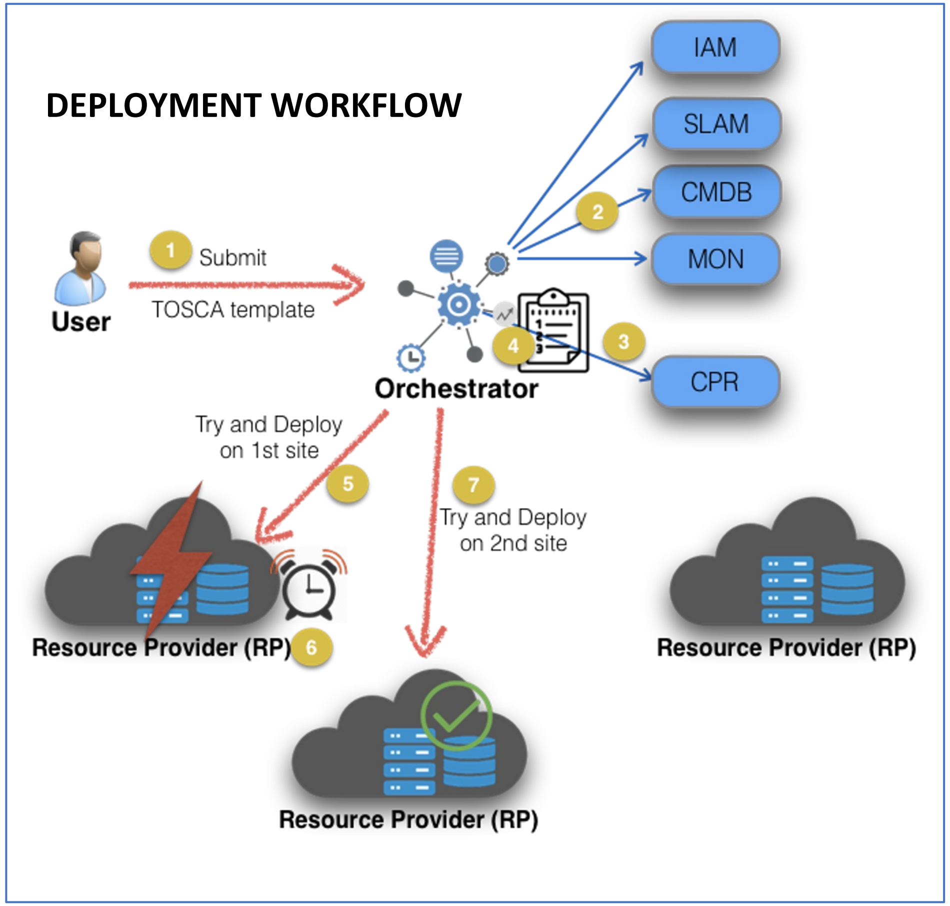
Corso utenti INFN Datacloud
16-18 July 2024

The INFN Cloud

- **INFN Cloud aims to offer a full set of high-level cloud services to INFN user communities**
  - the service catalogue is not static: new applications are included through a defined “on-boarding” process for new use-cases
- **Architecturally INFN Cloud is a federation of existing infrastructures**
  - the *INFN Cloud backbone*, consists of two tightly coupled federated sites: BARI and CNAF
  - a scalable set of satellite sites, geographically distributed across Italy, and loosely coupled.
- **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)





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

Ref: TOSCA Simple Profile in YAML Version 1.1

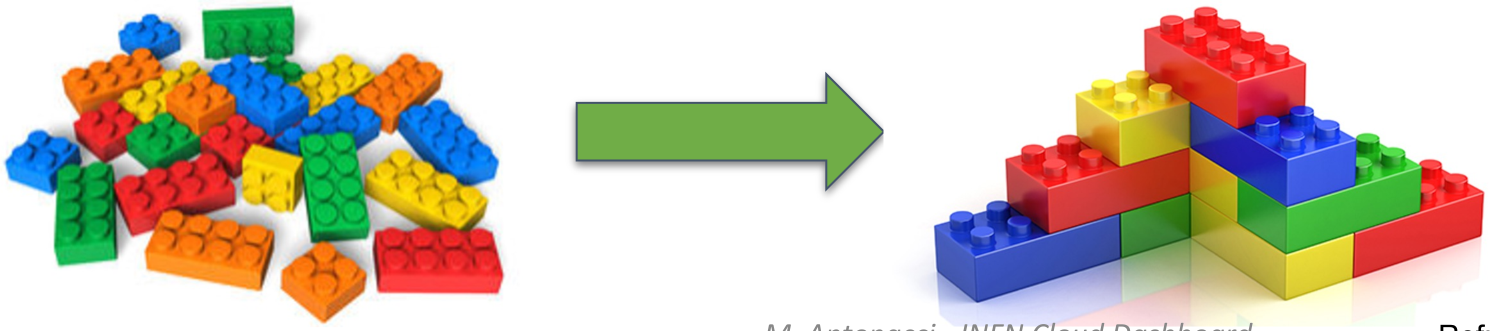
The INFN Cloud services

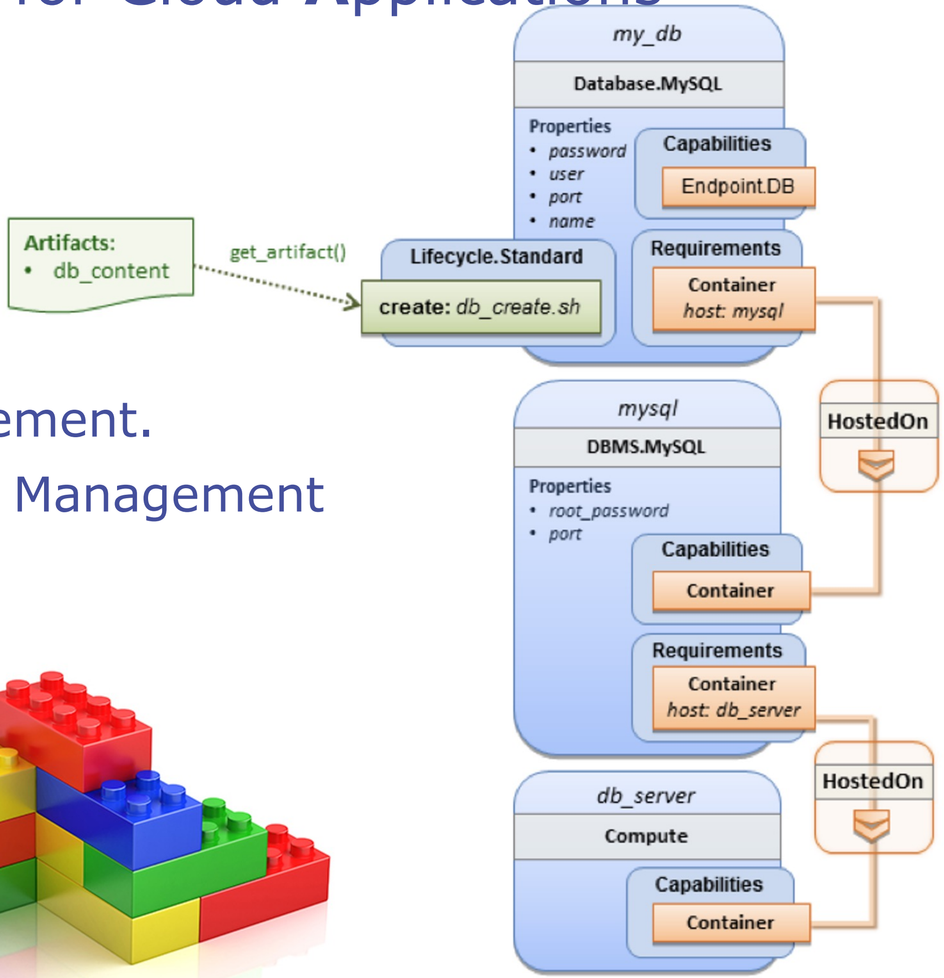
- The INFN Cloud services are based on **modular components and span the IaaS, PaaS and SaaS models** for both computing and data.
- All services are described by **TOSCA templates** (which can refer internally to other components such as Ansible playbooks, HELM charts, etc.).
- The services can be **deployed** via the INFN Cloud Dashboard or via a command line interface:
  - **Automatically** by the INFN Cloud Orchestrator on one of the federated Cloud infrastructures, depending on resource availability and policies.
  - **Manually** by a user on a specific federated Cloud infrastructure.

TOSCA

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

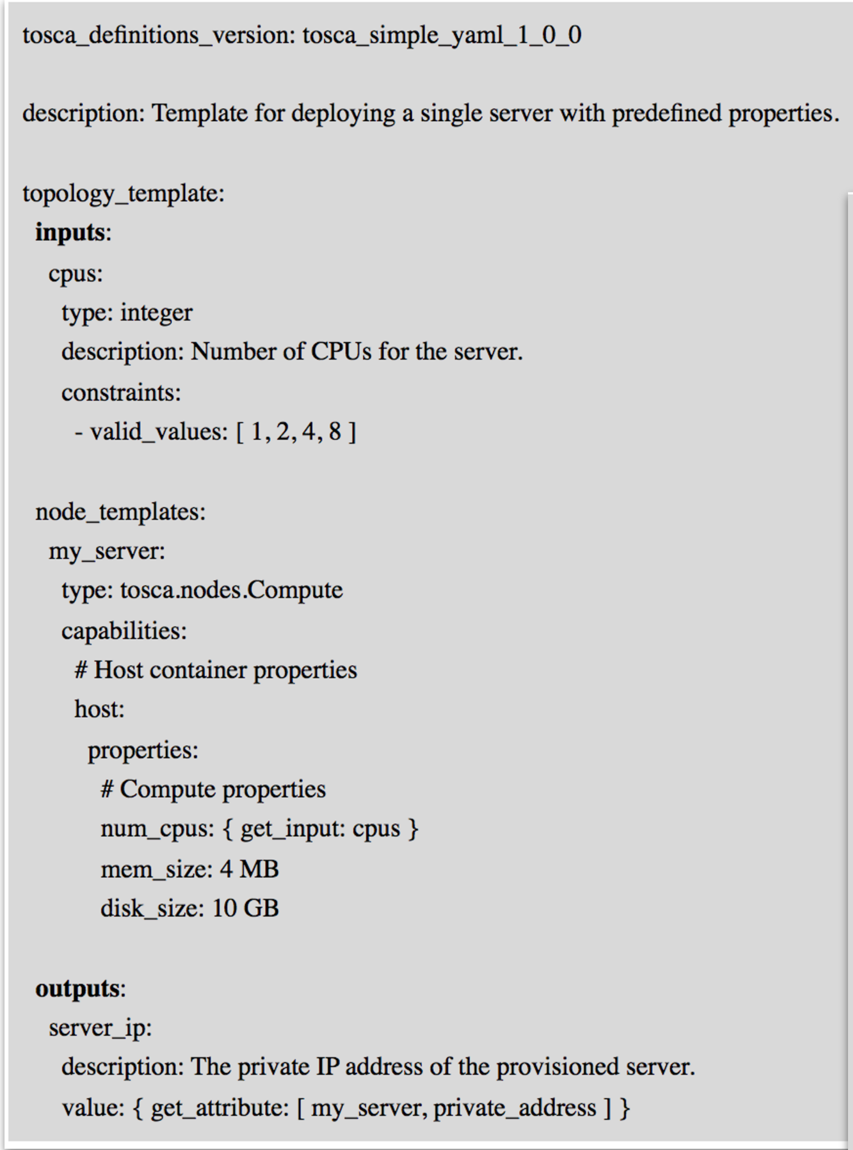
- Goals:
  - Automated Application Deployment and Management.
  - Portability of Application Descriptions and Their Management
  - Interoperability and Reusability of Components



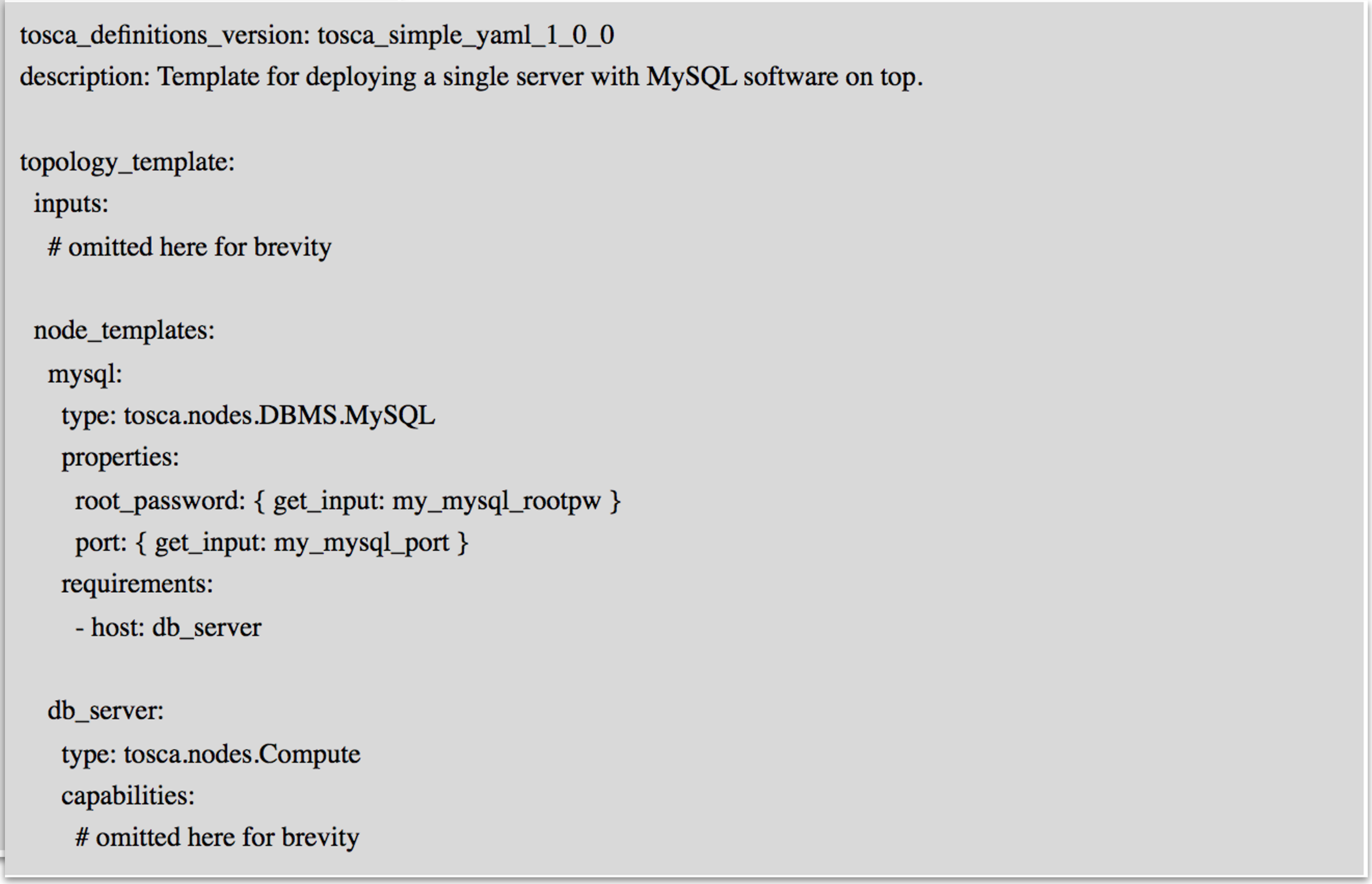


Ref: TOSCA Simple Profile in YAML Version 1.1

Template example



```
tosca_definitions_version: tosca_simple_yaml_1_0_0

description: Template for deploying a single server with predefined properties.

topology_template:
 inputs:
  cpus:
   type: integer
   description: Number of CPUs for the server.
   constraints:
    - valid_values: [ 1, 2, 4, 8 ]

 node_templates:
  my_server:
   type: tosca.nodes.Compute
   capabilities:
    # Host container properties
    host:
     properties:
      # Compute properties
      num_cpus: { get_input: cpus }
      mem_size: 4 MB
      disk_size: 10 GB

 outputs:
  server_ip:
   description: The private IP address of the provisioned server.
   value: { get_attribute: [ my_server, private_address ] }
```



```
tosca_definitions_version: tosca_simple_yaml_1_0_0
description: Template for deploying a single server with MySQL software on top.

topology_template:
 inputs:
  # omitted here for brevity

 node_templates:
  mysql:
   type: tosca.nodes.DBMS.MySQL
   properties:
    root_password: { get_input: my_mysql_rootpw }
    port: { get_input: my_mysql_port }
   requirements:
    - host: db_server

  db_server:
   type: tosca.nodes.Compute
   capabilities:
    # omitted here for brevity
```

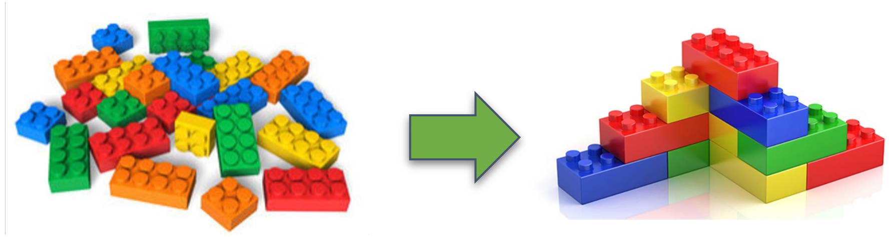
The service catalogue

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

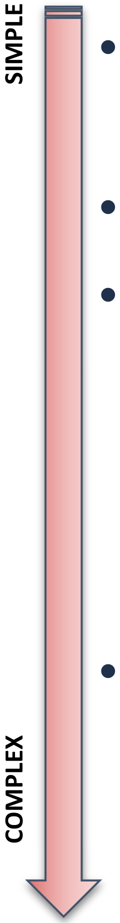
Main objectives:

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



Which services are available?

General purpose services:



- Virtual Machine with or without external block storage, eventually equipped with docker engine and docker-compose, on top of which dockerized services can be automatically started;
- data analytics and visualization environments based on Elasticsearch and Kibana;
- file sync & share solution based on OwnCloud/NextCloud with 1) replicated backend storage on the S3-compliant Object Storage provided by the INFN Cloud infrastructure; 2) automatic configuration for enabling INDIGO IAM OpenID Connect authentication; 3) pre-installed and configured backup cron jobs for safely storing configuration and data on the Object Storage for future restore in case of disaster; 4) integrated application and backup monitoring based on Nagios.
- web-based multi-user interactive development environment for notebooks, code and data built on JupyterLab and enhanced with 1) persistent storage areas for storing results and notebooks for future re-use; 2) a monitoring system based on Prometheus and Grafana for collecting relevant metrics;

Which services are available? (2)

K8s-based services:

- HTCondor on-demand clusters
- Spark clusters, integrated with Jupyter

Experiment-specific services:

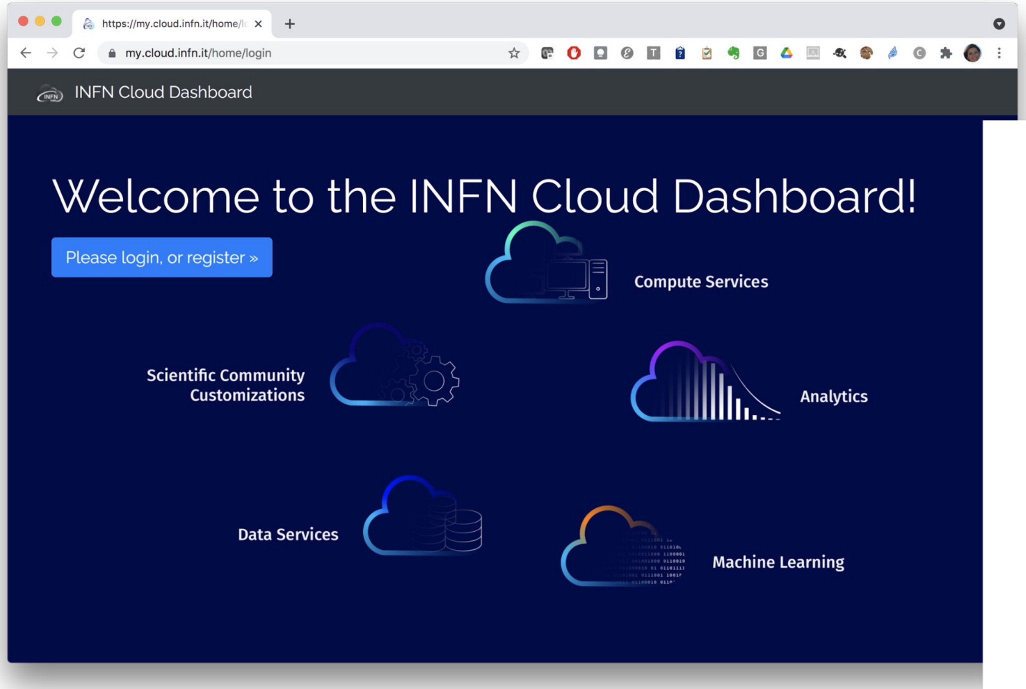
- CYGNO experiment, studying Dark Matter and Neutrinos
- AI INFN, a INFN-funded project aiming at lowering the potential barriers for accessing specialized hardware for the exploitation of Machine Learning techniques.

The service catalogue can be easily extended with the simple addition/customization of TOSCA templates.

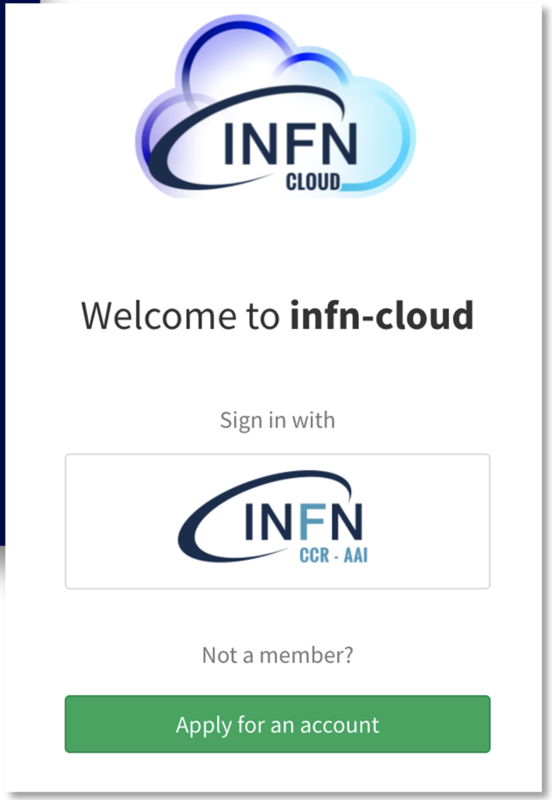
The INFN Cloud Dashboard

https://my.cloud.infn.it

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


https://my.cloud.infn.it/home/l
my.cloud.infn.it/home/login
INFN Cloud Dashboard
Please login, or register »
Compute Services
Scientific Community Customizations
Analytics
Data Services
Machine Learning

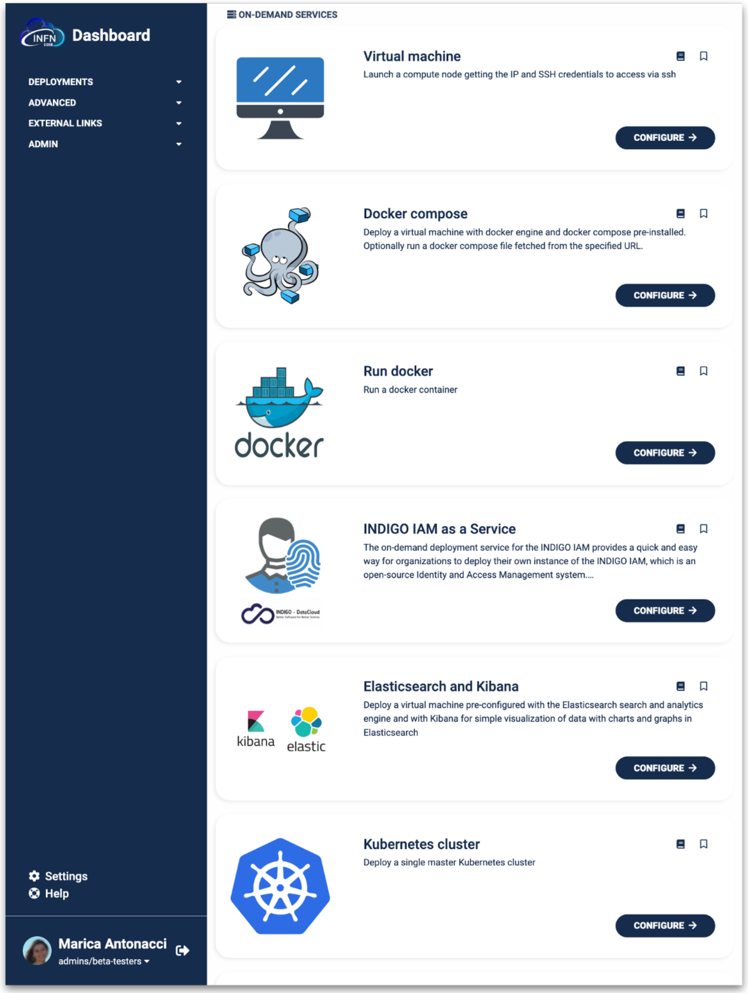
Welcome to the INFN Cloud Dashboard!



Welcome to **infn-cloud**

Sign in with

Not a member?



ON-DEMAND SERVICES

Virtual machine
Launch a compute node getting the IP and SSH credentials to access via ssh
CONFIGURE →

Docker compose
Deploy a virtual machine with docker engine and docker compose pre-installed. Optionally run a docker compose file fetched from the specified URL.
CONFIGURE →

Run docker
Run a docker container
CONFIGURE →

INDIGO IAM as a Service
The on-demand deployment service for the INDIGO IAM provides a quick and easy way for organizations to deploy their own instance of the INDIGO IAM, which is an open-source Identity and Access Management system....
CONFIGURE →

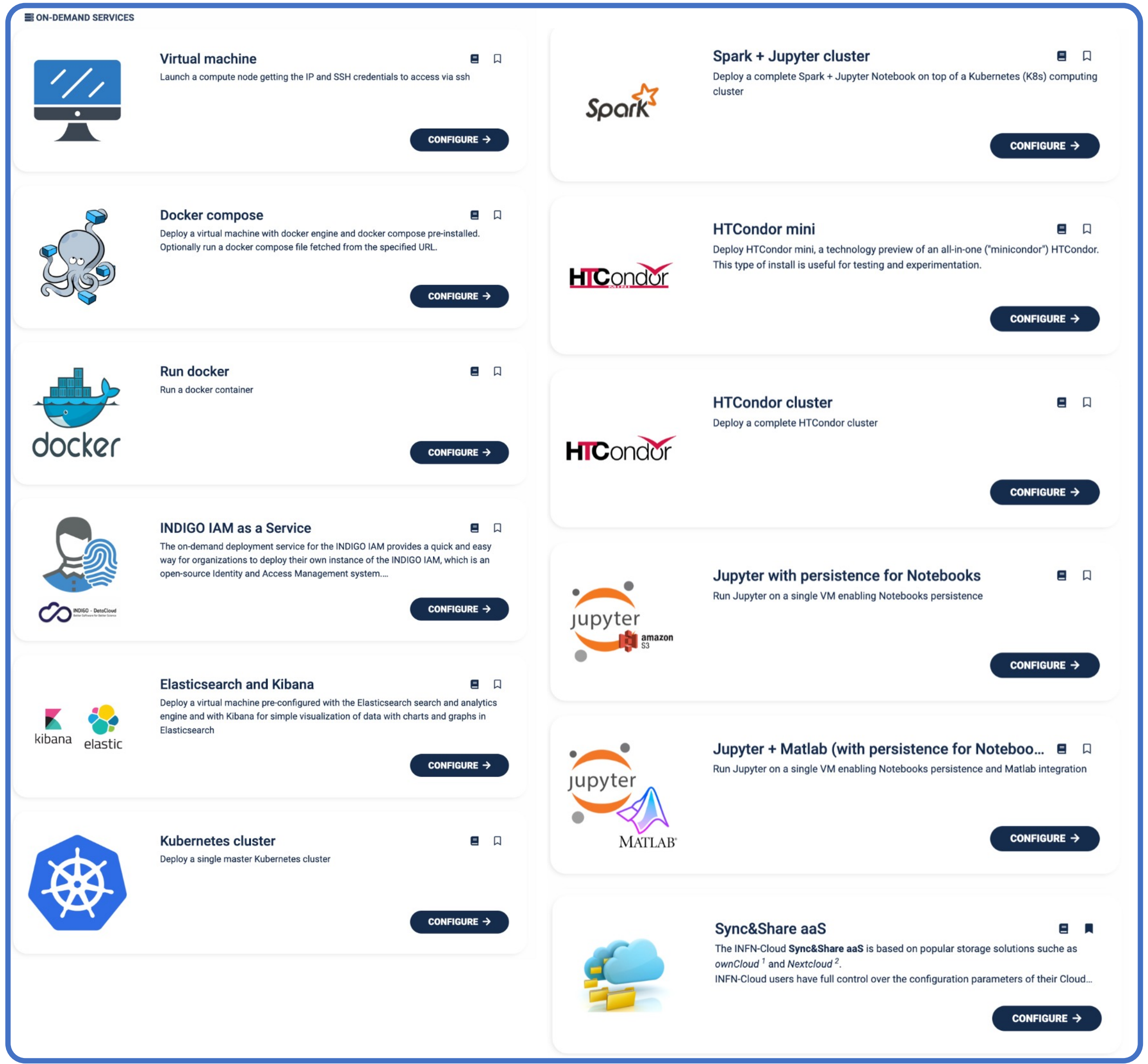
Elasticsearch and Kibana
Deploy a virtual machine pre-configured with the Elasticsearch search and analytics engine and with Kibana for simple visualization of data with charts and graphs in Elasticsearch
CONFIGURE →

Kubernetes cluster
Deploy a single master Kubernetes cluster
CONFIGURE →

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 INFN Cloud Dashboard



ON-DEMAND SERVICES

Virtual machine
Launch a compute node getting the IP and SSH credentials to access via ssh
CONFIGURE →

Docker compose
Deploy a virtual machine with docker engine and docker compose pre-installed. Optionally run a docker compose file fetched from the specified URL.
CONFIGURE →

Run docker
Run a docker container
CONFIGURE →

INDIGO IAM as a Service
The on-demand deployment service for the INDIGO IAM provides a quick and easy way for organizations to deploy their own instance of the INDIGO IAM, which is an open-source Identity and Access Management system....
CONFIGURE →

Elasticsearch and Kibana
Deploy a virtual machine pre-configured with the Elasticsearch search and analytics engine and with Kibana for simple visualization of data with charts and graphs in Elasticsearch
CONFIGURE →

Kubernetes cluster
Deploy a single master Kubernetes cluster
CONFIGURE →

Spark + Jupyter cluster
Deploy a complete Spark + Jupyter Notebook on top of a Kubernetes (K8s) computing cluster
CONFIGURE →

HTCondor mini
Deploy HTCondor mini, a technology preview of an all-in-one ("minicondor") HTCondor. This type of install is useful for testing and experimentation.
CONFIGURE →

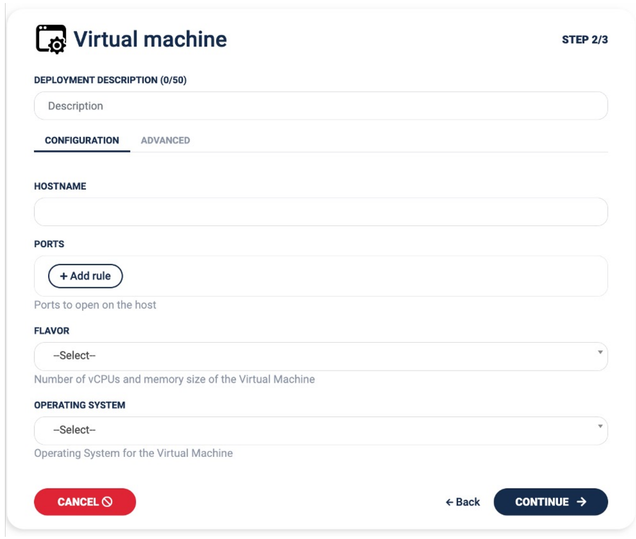
HTCondor cluster
Deploy a complete HTCondor cluster
CONFIGURE →

Jupyter with persistence for Notebooks
Run Jupyter on a single VM enabling Notebooks persistence
CONFIGURE →

Jupyter + Matlab (with persistence for Noteboo...
Run Jupyter on a single VM enabling Notebooks persistence and Matlab integration
CONFIGURE →

Sync&Share aaS
The INFN-Cloud **Sync&Share aaS** is based on popular storage solutions suche as *ownCloud* [1] and *Nextcloud* [2].
INFN-Cloud users have full control over the configuration parameters of their Cloud...
CONFIGURE →

The services are **easily customizable** and configurable directly by users



Virtual machine STEP 2/3

DEPLOYMENT DESCRIPTION (0/50)
Description

CONFIGURATION | ADVANCED

HOSTNAME

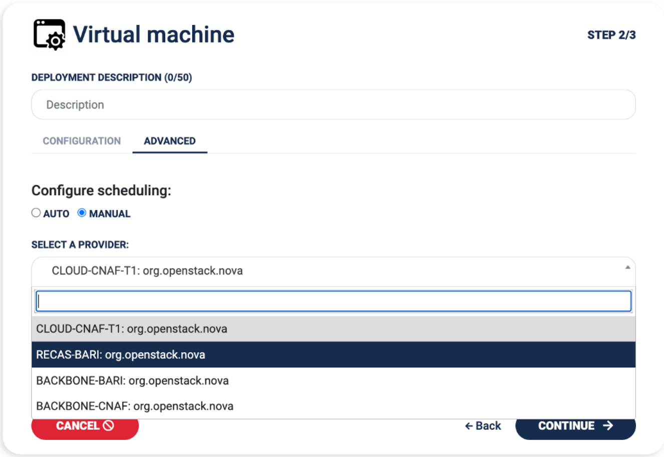
PORTS
+ Add rule
Ports to open on the host

FLAVOR
--Select--
Number of vCPUs and memory size of the Virtual Machine

OPERATING SYSTEM
--Select--
Operating System for the Virtual Machine

CANCEL ⊘ ← Back CONTINUE →

Transparent, multi-site **federation or site selection** made manually by the user



Virtual machine STEP 2/3

DEPLOYMENT DESCRIPTION (0/50)
Description

CONFIGURATION | ADVANCED

Configure scheduling:
○ AUTO ● MANUAL

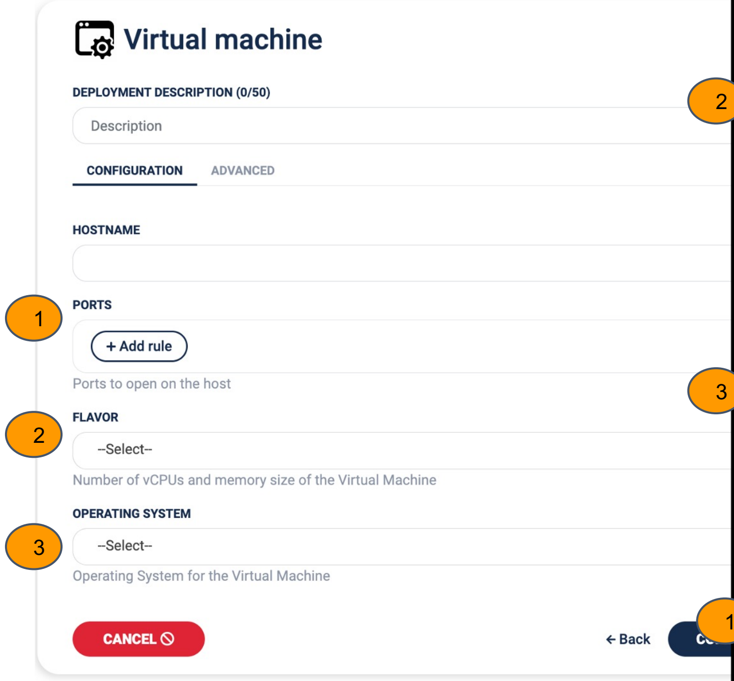
SELECT A PROVIDER:
CLOUD-CNAF-T1: org.openstack.nova

CLOUD-CNAF-T1: org.openstack.nova
RECAS-BARI: org.openstack.nova
BACKBONE-BARI: org.openstack.nova
BACKBONE-CNAF: org.openstack.nova

CANCEL ⊘ ← Back CONTINUE →

Marica Antonacci (marica.antonacci@ba.infn.it)

Service request customization



Virtual machine

DEPLOYMENT DESCRIPTION (0/50)

Description

CONFIGURATION ADVANCED

HOSTNAME

PORTS

+ Add rule

Ports to open on the host

FLAVOR

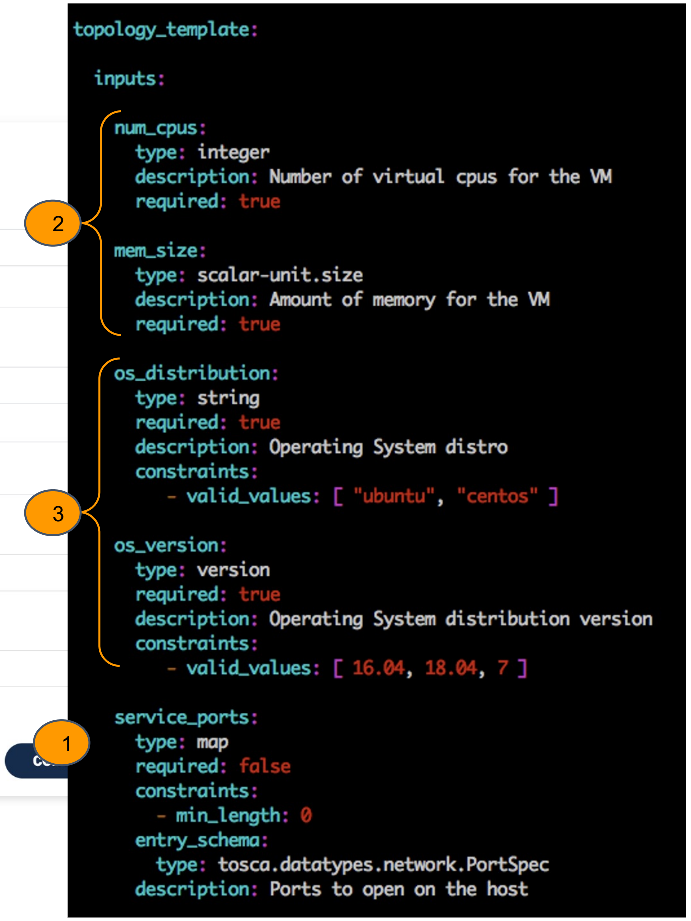
--Select--

Number of vCPUs and memory size of the Virtual Machine

OPERATING SYSTEM

--Select--

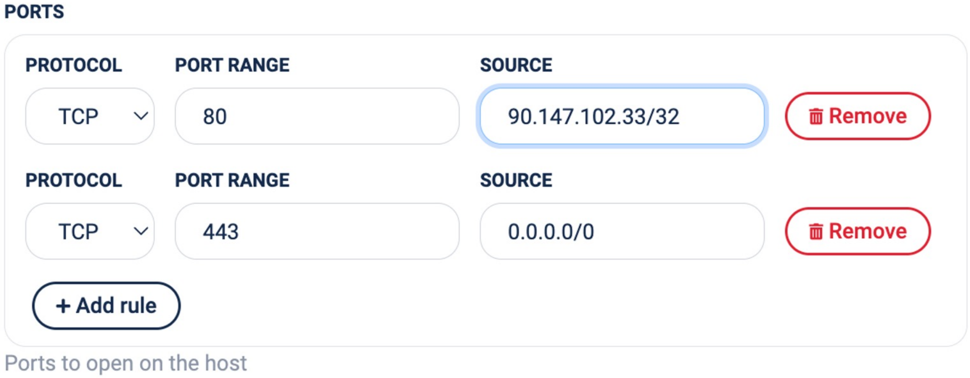
Operating System for the Virtual Machine



```
topology_template:

 inputs:

  num_cpus:
    type: integer
    description: Number of virtual cpus for the VM
    required: true

  mem_size:
    type: scalar-unit.size
    description: Amount of memory for the VM
    required: true

  os_distribution:
    type: string
    required: true
    description: Operating System distro
    constraints:
      - valid_values: [ "ubuntu", "centos" ]

  os_version:
    type: version
    required: true
    description: Operating System distribution version
    constraints:
      - valid_values: [ 16.04, 18.04, 7 ]

  service_ports:
    type: map
    required: false
    constraints:
     - min_length: 0
    entry_schema:
      type: tosca.datatypes.network.PortSpec
    description: Ports to open on the host
```

The configuration form allows the user to specify requirements for the deployment in a straightforward way

- checking the mandatory fields
- hiding the complexity of TOSCA
  - related fields are collapsed into a single input (e.g. num_cpu & mem_size into flavor)
  - complex TOSCA types are managed with dedicated Javascript functions (e.g. the ports specification)



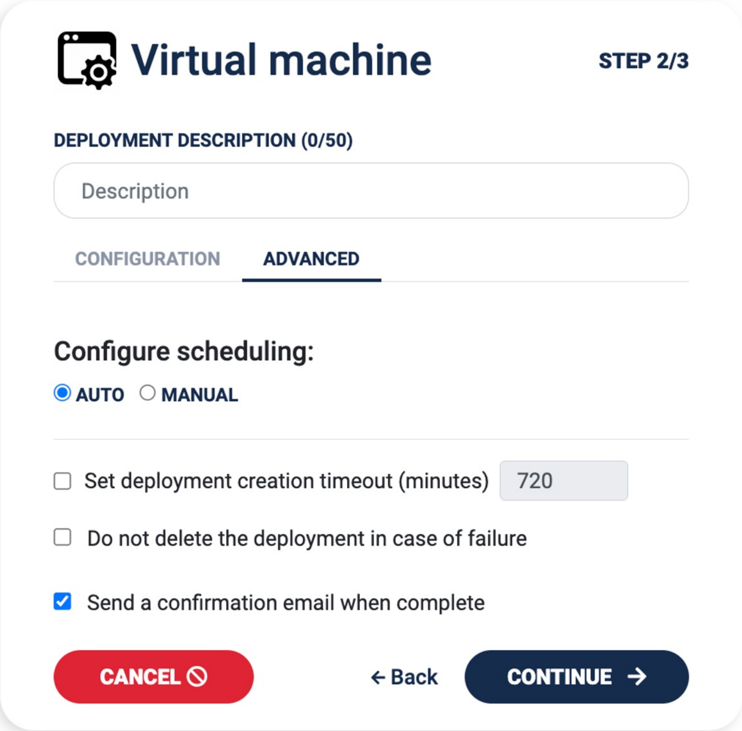
PORTS

PROTOCOL	PORT RANGE	SOURCE	
TCP	80	90.147.102.33/32	🗑 Remove
TCP	443	0.0.0.0/0	🗑 Remove

+ Add rule

Ports to open on the host

Advanced configurations



Virtual machine STEP 2/3

DEPLOYMENT DESCRIPTION (0/50)

Description

CONFIGURATION ADVANCED

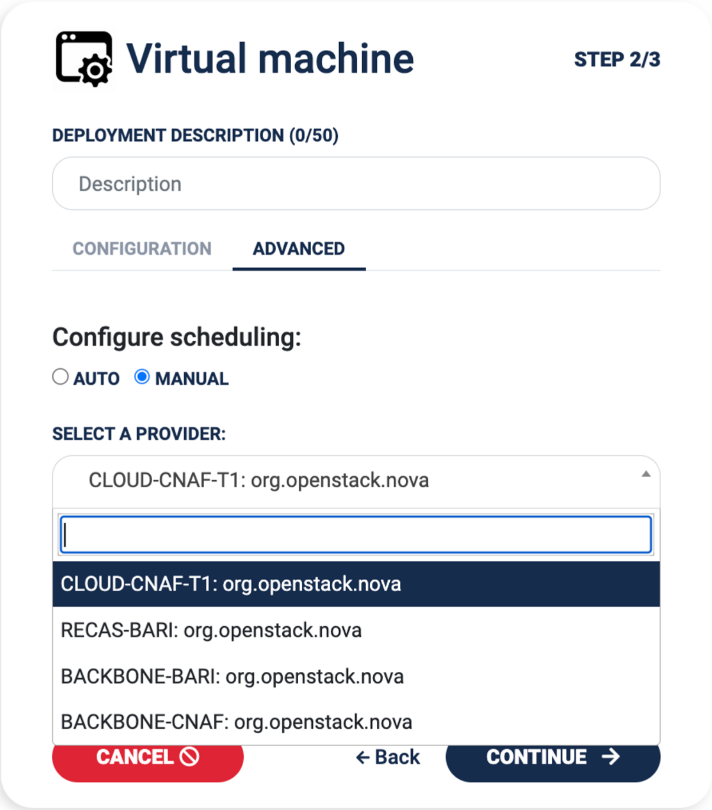
Configure scheduling:

● AUTO ○ MANUAL

□ Set deployment creation timeout (minutes) 720

□ Do not delete the deployment in case of failure

☑ Send a confirmation email when complete



Virtual machine STEP 2/3

DEPLOYMENT DESCRIPTION (0/50)

Description

CONFIGURATION ADVANCED

Configure scheduling:

○ AUTO ● MANUAL

SELECT A PROVIDER:

CLOUD-CNAF-T1: org.openstack.nova

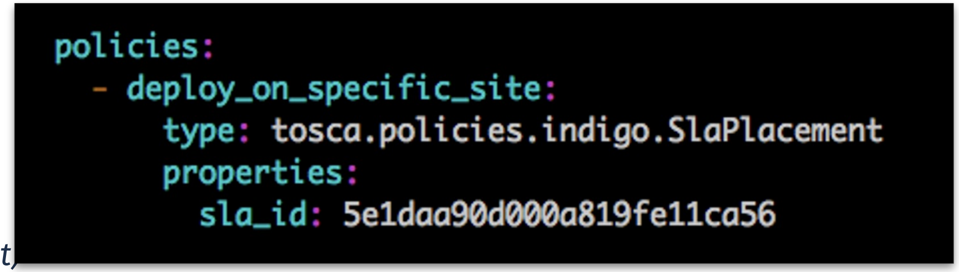
- CLOUD-CNAF-T1: org.openstack.nova
- RECAS-BARI: org.openstack.nova
- BACKBONE-BARI: org.openstack.nova
- BACKBONE-CNAF: org.openstack.nova

The dashboard allows also to bypass the automatic scheduling implemented by the Orchestrator: the user can choose a specific provider to send his/her deployment request to.

Under the hood:

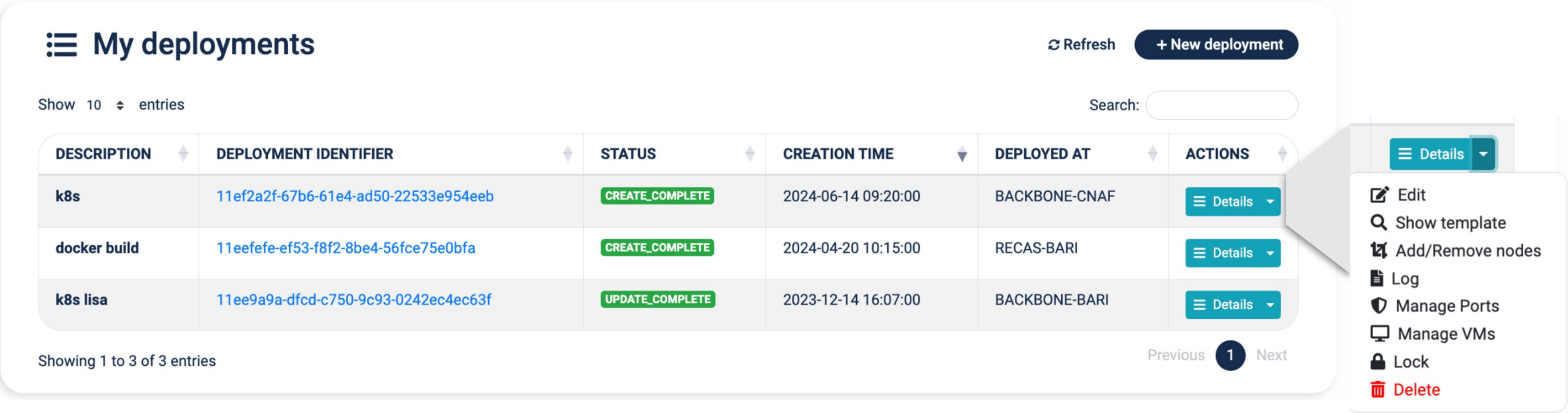
the drop-down menu is automatically created by the Dashboard interacting the SLA Manager Service to get the list of providers for the user;

before submitting the request to the Orchestrator, the Dashboard completes the TOSCA template including the proper SLA placement policy:



```
policies:
  - deploy_on_specific_site:
      type: tosca.policies.indigo.SlaPlacement
      properties:
        sla_id: 5e1daa90d000a819fe11ca56
```

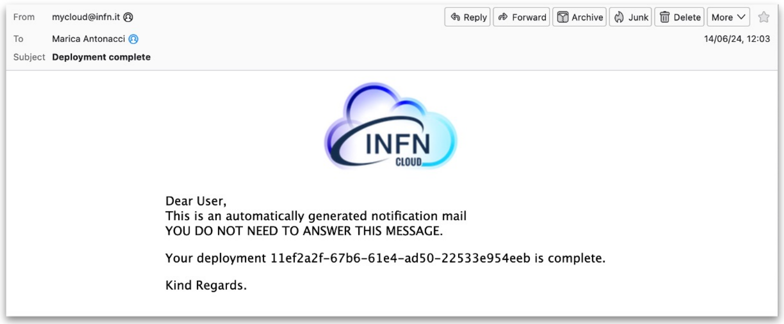
Deployment outputs and notifications



My deployments

DESCRIPTION	DEPLOYMENT IDENTIFIER	STATUS	CREATION TIME	DEPLOYED AT	ACTIONS
k8s	11ef2a2f-67b6-61e4-ad50-22533e954eeb	CREATE_COMPLETE	2024-06-14 09:20:00	BACKBONE-CNAF	Details
docker build	11eefefe-ef53-f8f2-8be4-56fce75e0bfa	CREATE_COMPLETE	2024-04-20 10:15:00	RECAS-BARI	Details
k8s lisa	11ee9a9a-dfcd-c750-9c93-0242ec4ec63f	UPDATE_COMPLETE	2023-12-14 16:07:00	BACKBONE-BARI	Details

Showing 1 to 3 of 3 entries

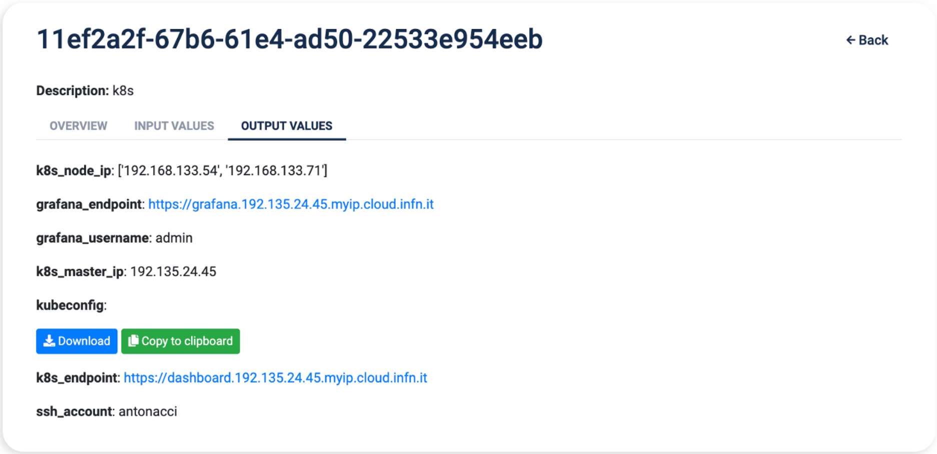


From mycloud@infn.it
To Marica Antonacci
Subject **Deployment complete**
14/06/24, 12:03

Dear User,
This is an automatically generated notification mail
YOU DO NOT NEED TO ANSWER THIS MESSAGE.

Your deployment 11ef2a2f-67b6-61e4-ad50-22533e954eeb is complete.

Kind Regards.



11ef2a2f-67b6-61e4-ad50-22533e954eeb

Description: k8s

k8s_node_ip: ['192.168.133.54', '192.168.133.71']

grafana_endpoint: https://grafana.192.135.24.45.myip.cloud.infn.it

grafana_username: admin

k8s_master_ip: 192.135.24.45

kubeconfig:

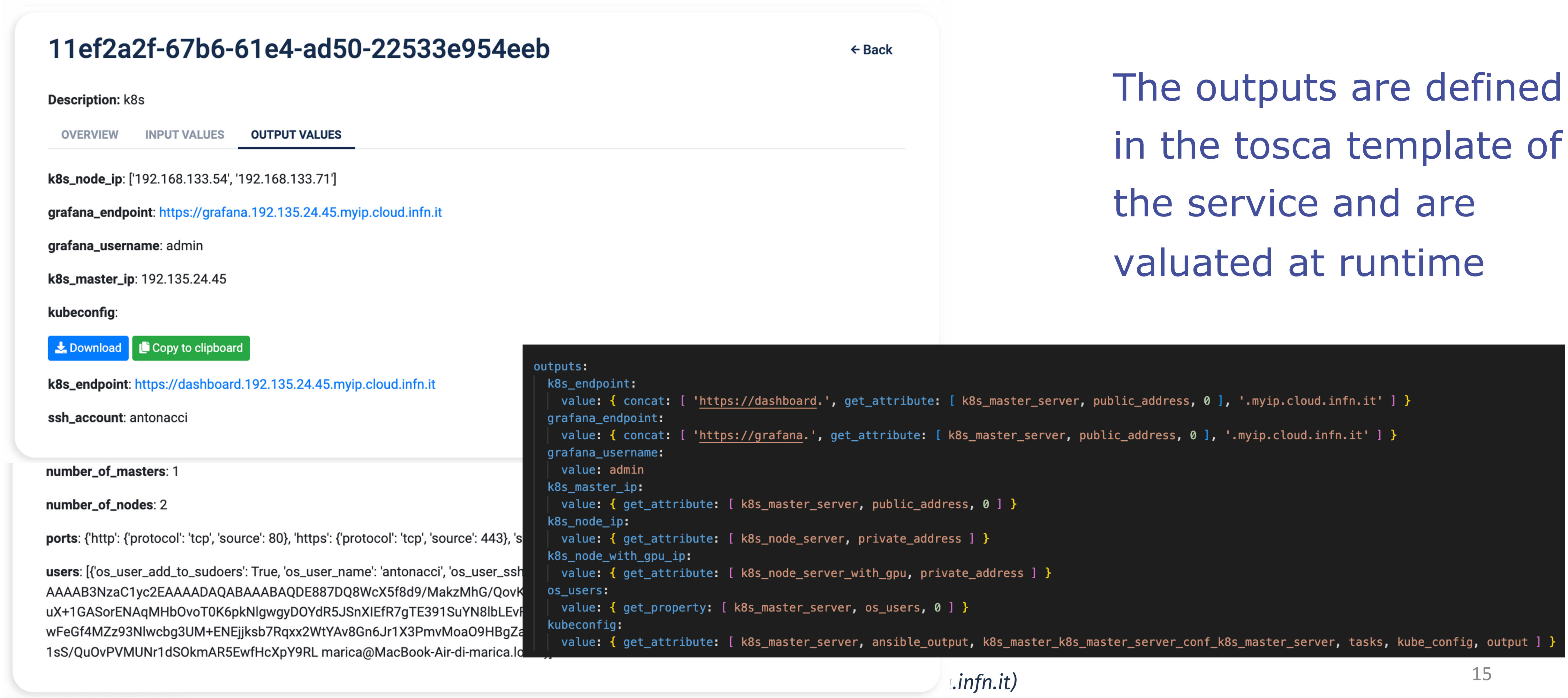
k8s_endpoint: https://dashboard.192.135.24.45.myip.cloud.infn.it

ssh_account: antonacci

A notification system is implemented in the Dashboard: the user receives an automatic email as soon as the deployment is ready.

Then, the details about the deployed service can be accessed through the Dashboard.

Deployment details



11ef2a2f-67b6-61e4-ad50-22533e954eeb

Description: k8s

k8s_node_ip: ['192.168.133.54', '192.168.133.71']

grafana_endpoint: https://grafana.192.135.24.45.myip.cloud.infn.it

grafana_username: admin

k8s_master_ip: 192.135.24.45

kubeconfig:

k8s_endpoint: https://dashboard.192.135.24.45.myip.cloud.infn.it

ssh_account: antonacci

number_of_masters: 1

number_of_nodes: 2

ports: {'http': {'protocol': 'tcp', 'source': 80}, 'https': {'protocol': 'tcp', 'source': 443}, 's

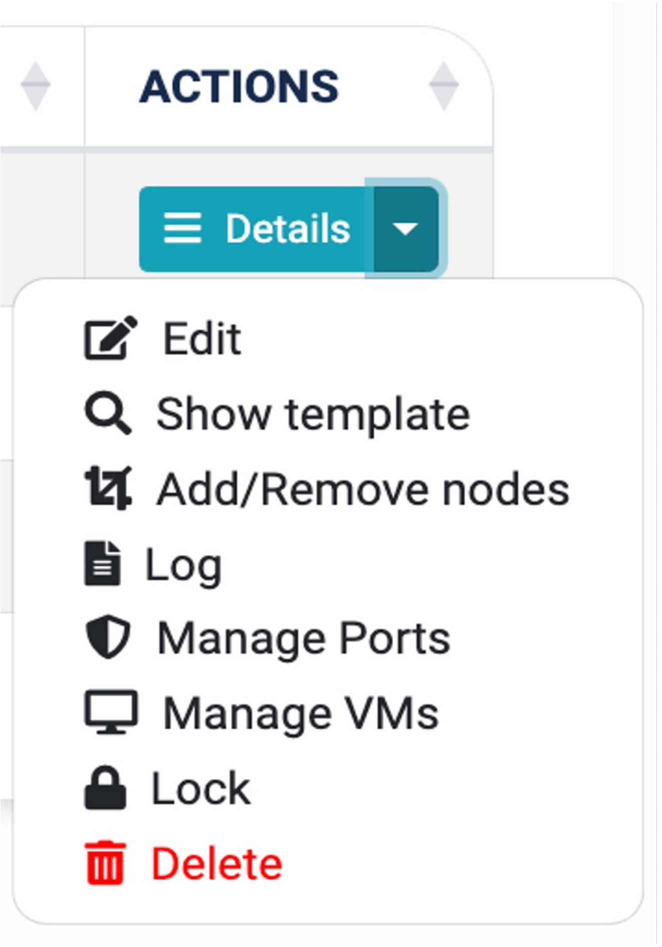
users: [{'os_user_add_to_sudoers': True, 'os_user_name': 'antonacci', 'os_user_ssh
AAAAB3NzaC1yc2EAAAADAQABAAABAQDE887DQ8WcX5f8d9/MakzMhG/QovK
uX+1GASorENAqMHbOvoT0K6pkNlgwgyDOYdR5JSnXIEfR7gTE391SuYN8lbLEvF
wFeGf4MZz93Nlwcbg3UM+ENEjjksb7Rqxx2WtYAv8Gn6Jr1X3PmvMoaO9HBgZa
1sS/QuOvPVMUNr1dSOkmAR5EwfHcXpY9RL marica@MacBook-Air-di-marica.lo

```
outputs:
  k8s_endpoint:
    value: { concat: [ 'https://dashboard.', get_attribute: [ k8s_master_server, public_address, 0 ], '.myip.cloud.infn.it' ] }
  grafana_endpoint:
    value: { concat: [ 'https://grafana.', get_attribute: [ k8s_master_server, public_address, 0 ], '.myip.cloud.infn.it' ] }
  grafana_username:
    value: admin
  k8s_master_ip:
    value: { get_attribute: [ k8s_master_server, public_address, 0 ] }
  k8s_node_ip:
    value: { get_attribute: [ k8s_node_server, private_address ] }
  k8s_node_with_gpu_ip:
    value: { get_attribute: [ k8s_node_server_with_gpu, private_address ] }
  os_users:
    value: { get_property: [ k8s_master_server, os_users, 0 ] }
  kubeconfig:
    value: { get_attribute: [ k8s_master_server, ansible_output, k8s_master_k8s_master_server_conf_k8s_master_server, tasks, kube_config, output ] }
```

The outputs are defined in the tosca template of the service and are valuated at runtime

.infn.it)

Menu “Actions”

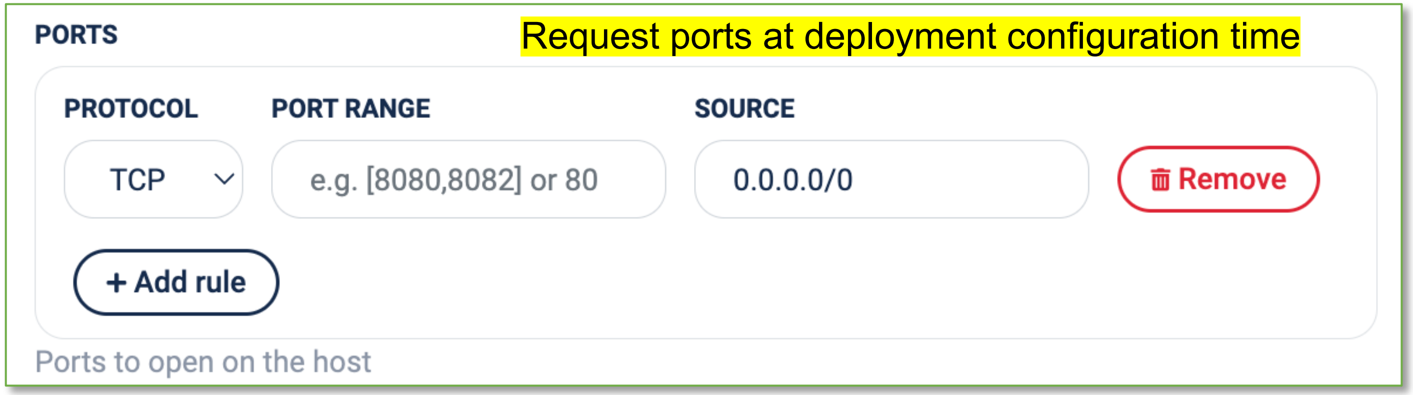


- **Edit**: modify the deployment description
- **Show template**: view the TOSCA template used to make the deployment
- **Add/Remove nodes**: add or remove nodes in a deployment (available only for clusters)
- **Log**: view the contextualization log (generated by the Infrastructure Manager)
- **Manage VMs**: get detailed information about the VMs of the deployment and start/stop them
- **Manage Ports**: modify the security group rules of the “main” machine
- **Lock**: protect deployment against delete operations
- **Delete**: remove the whole deployment

Ports management

- In both public and private network deployments, the main machine serves a dual purpose:
  - Provides SSH access, with port 22 open.
  - Exposes any deployed services, with open ports varying based on the services instantiated.
- In some cases, **the user can specify additional ports to be opened** through the service configuration form (e.g., for deploying a single virtual machine or a Kubernetes cluster) **when requesting the service deployment**.
- In all cases, **after the deployment is completed**, the user can modify the firewall rules of the main machine to restrict or permit traffic on specific ports and to/from specific IP ranges.

Ports management (2)



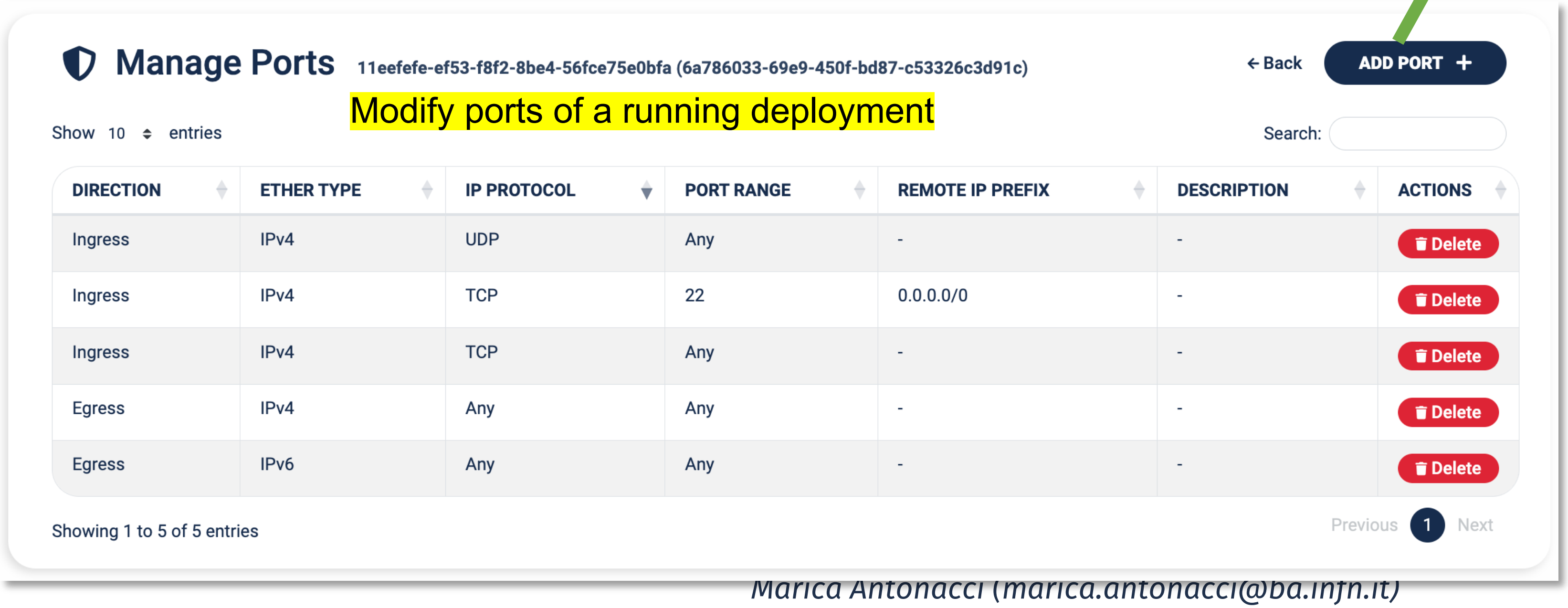
Request ports at deployment configuration time

PORTS

PROTOCOL	PORT RANGE	SOURCE	
TCP	e.g. [8080,8082] or 80	0.0.0.0/0	Remove

+ Add rule

Ports to open on the host



Modify ports of a running deployment

Manage Ports 11eefefe-ef53-f8f2-8be4-56fce75e0bfa (6a786033-69e9-450f-bd87-c53326c3d91c)

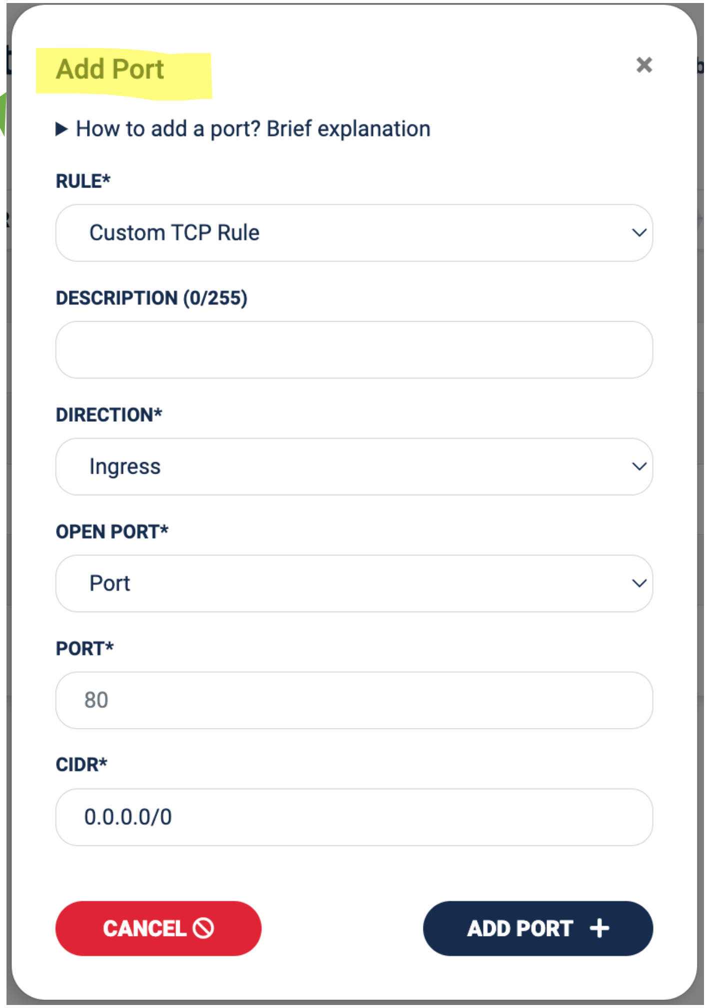
← Back ADD PORT +

Show 10 entries Search:

DIRECTION	ETHER TYPE	IP PROTOCOL	PORT RANGE	REMOTE IP PREFIX	DESCRIPTION	ACTIONS
Ingress	IPv4	UDP	Any	-	-	Delete
Ingress	IPv4	TCP	22	0.0.0.0/0	-	Delete
Ingress	IPv4	TCP	Any	-	-	Delete
Egress	IPv4	Any	Any	-	-	Delete
Egress	IPv6	Any	Any	-	-	Delete

Showing 1 to 5 of 5 entries

Previous 1 Next



Add Port ×

▶ How to add a port? Brief explanation

RULE*: Custom TCP Rule

DESCRIPTION (0/255):

DIRECTION*: Ingress

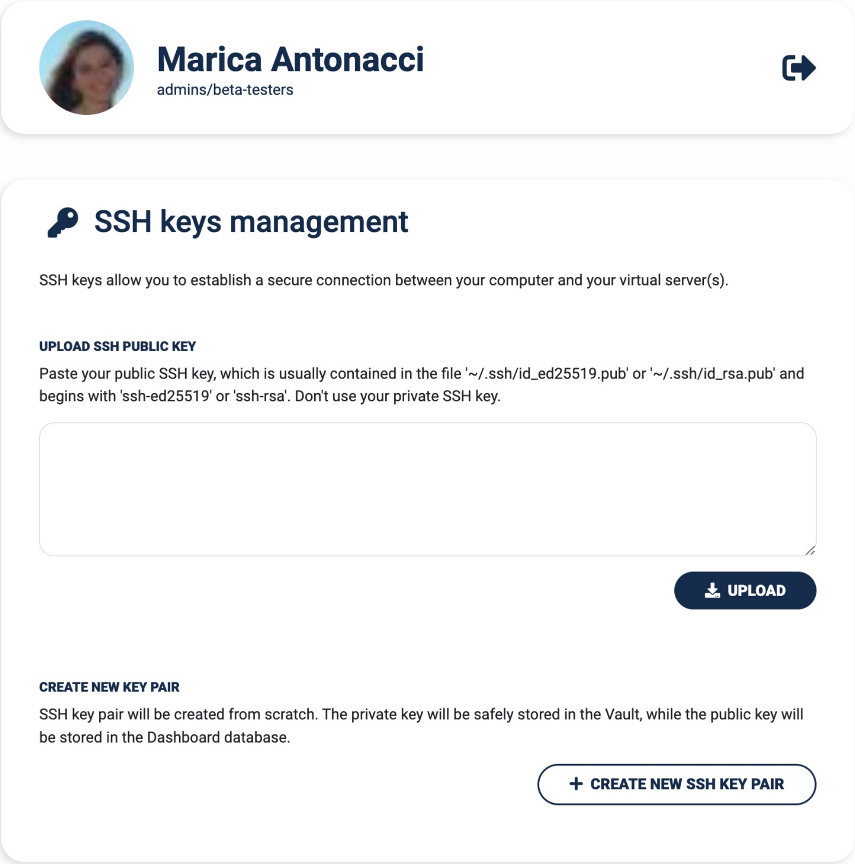
OPEN PORT*: Port

PORT*: 80

CIDR*: 0.0.0.0/0

CANCEL ⃠ ADD PORT +

Access all your VMs with your username and ssh key



Marica Antonacci

admins/beta-testers

SSH keys management

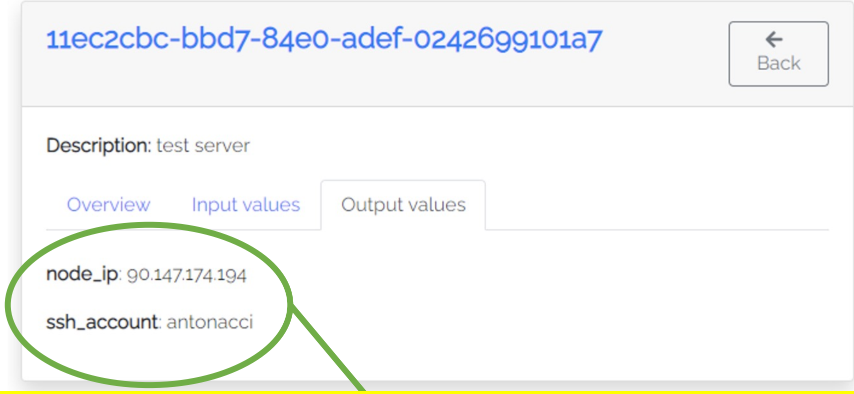
SSH keys allow you to establish a secure connection between your computer and your virtual server(s).

UPLOAD SSH PUBLIC KEY

Paste your public SSH key, which is usually contained in the file '~/.ssh/id_ed25519.pub' or '~/.ssh/id_rsa.pub' and begins with 'ssh-ed25519' or 'ssh-rsa'. Don't use your private SSH key.

CREATE NEW KEY PAIR

SSH key pair will be created from scratch. The private key will be safely stored in the Vault, while the public key will be stored in the Dashboard database.



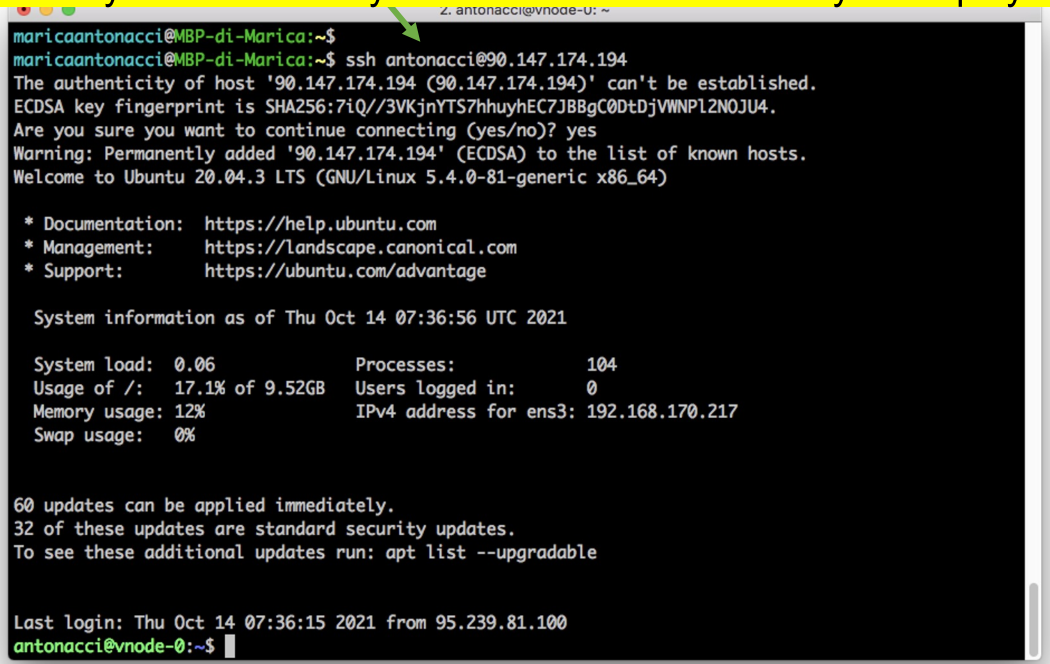
11ec2cbc-bbd7-84e0-adef-0242699101a7

Description: test server

node_ip: 90.147.174.194

ssh_account: antonacci

The SSH key is automatically installed on all the VMs of your deployments



2. antonacci@vnode-0: ~

```
maricaantonacci@MBP-di-Marica:~$
maricaantonacci@MBP-di-Marica:~$ ssh antonacci@90.147.174.194
The authenticity of host '90.147.174.194 (90.147.174.194)' can't be established.
ECDSA key fingerprint is SHA256:7iQ//3VKjnYTS7hhuyhEC7JBBgC0DtDjVWNPl2NOJU4.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '90.147.174.194' (ECDSA) to the list of known hosts.
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-81-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Thu Oct 14 07:36:56 UTC 2021

  System load:  0.06              Processes:             104
  Usage of /:   17.1% of 9.52GB   Users logged in:       0
  Memory usage: 12%               IPv4 address for ens3: 192.168.170.217
  Swap usage:   0%


60 updates can be applied immediately.
32 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


Last login: Thu Oct 14 07:36:15 2021 from 95.239.81.100
antonacci@vnode-0:~$
```

The dashboard is integrated with an instance of Hashicorp Vault to store secrets, such as the private SSH key. This is particularly useful if the user chooses to create a key pair directly from the dashboard instead of uploading a pre-generated public key.

Service implementation strategy details

INFN Cloud services implementation strategy

The employed strategy is based on the **Infrastructure as Code paradigm**.

Users describe "**What**" is needed rather than "**How**" a specific service or functionality should be implemented.

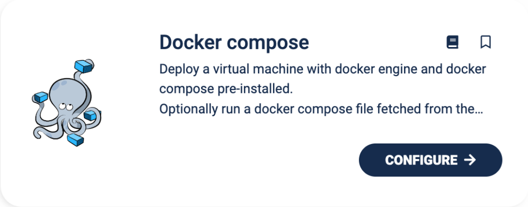
The adopted technologies enable a Lego-like approach: services can be composed and modules reused to create the desired infrastructure.

TOSCA is used to model the topology of the whole application stack

Ansible is used to automate the configuration of the virtual environments

Docker is used to encapsulate the high-level application software and runtime

Docker compose base implementation



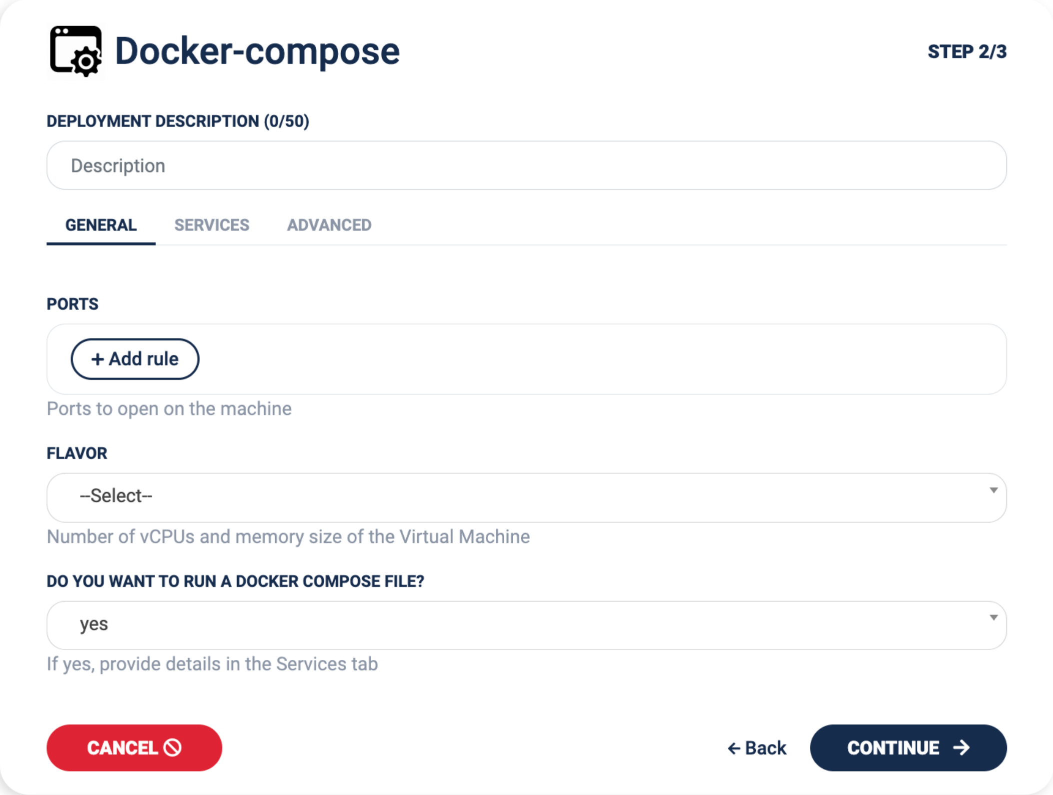
Docker compose

Deploy a virtual machine with docker engine and docker compose pre-installed.
Optionally run a docker compose file fetched from the...

CONFIGURE →

Let’s have a look at the TOSCA template

https://baltig.infn.it/infn-cloud/tosca-templates/-/blob/master/docker/docker_compose.yaml



Docker-compose

STEP 2/3

DEPLOYMENT DESCRIPTION (0/50)

Description

GENERAL SERVICES ADVANCED

PORTS

+ Add rule

Ports to open on the machine

FLAVOR

--Select--

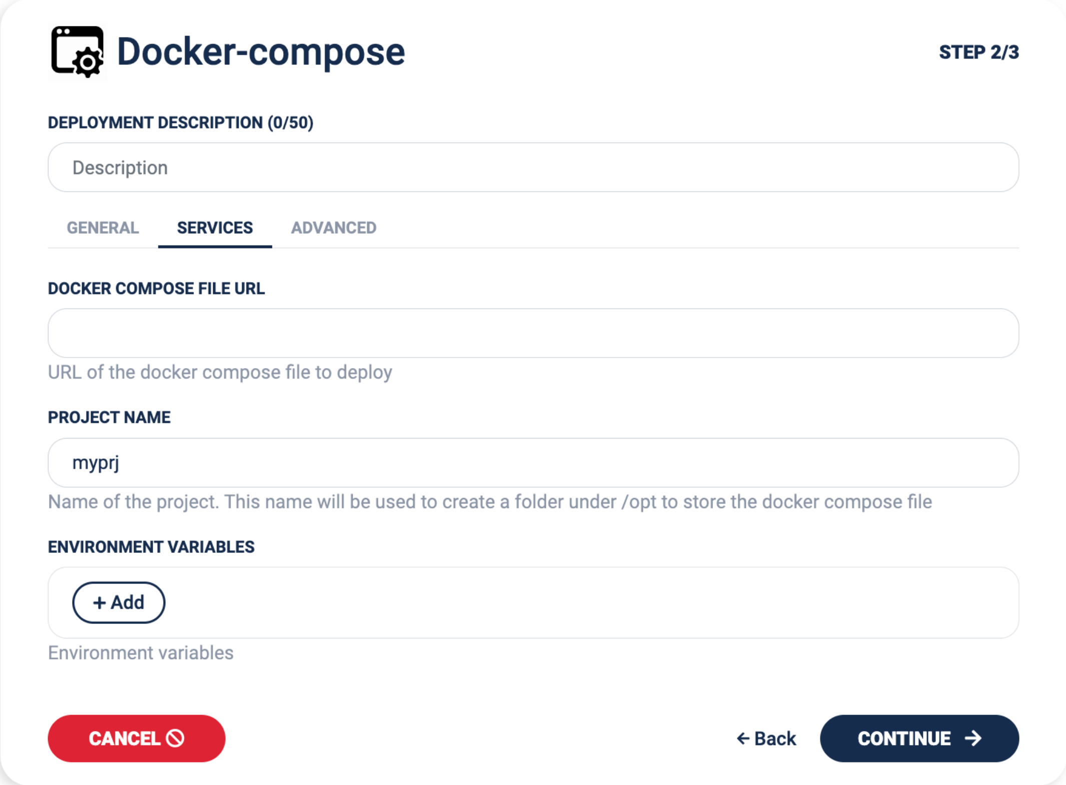
Number of vCPUs and memory size of the Virtual Machine

DO YOU WANT TO RUN A DOCKER COMPOSE FILE?

yes

If yes, provide details in the Services tab

CANCEL ⃠ ← Back CONTINUE →



Docker-compose

STEP 2/3

DEPLOYMENT DESCRIPTION (0/50)

Description

GENERAL SERVICES ADVANCED

DOCKER COMPOSE FILE URL

URL of the docker compose file to deploy

PROJECT NAME

myprj

Name of the project. This name will be used to create a folder under /opt to store the docker compose file

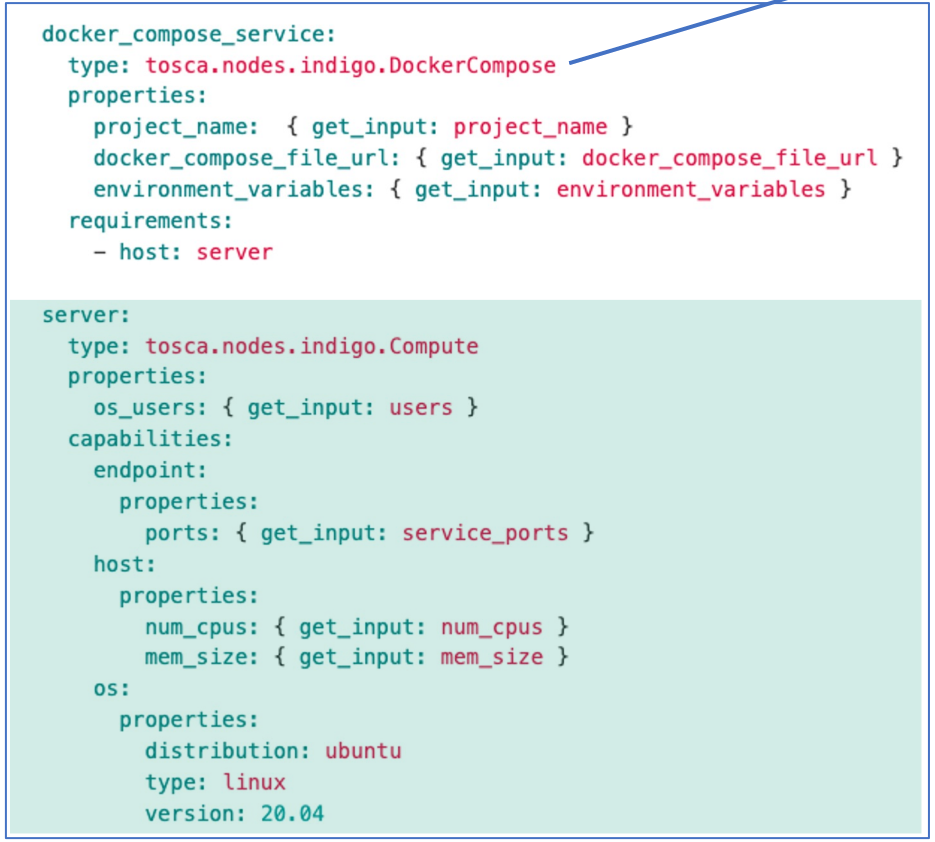
ENVIRONMENT VARIABLES

+ Add

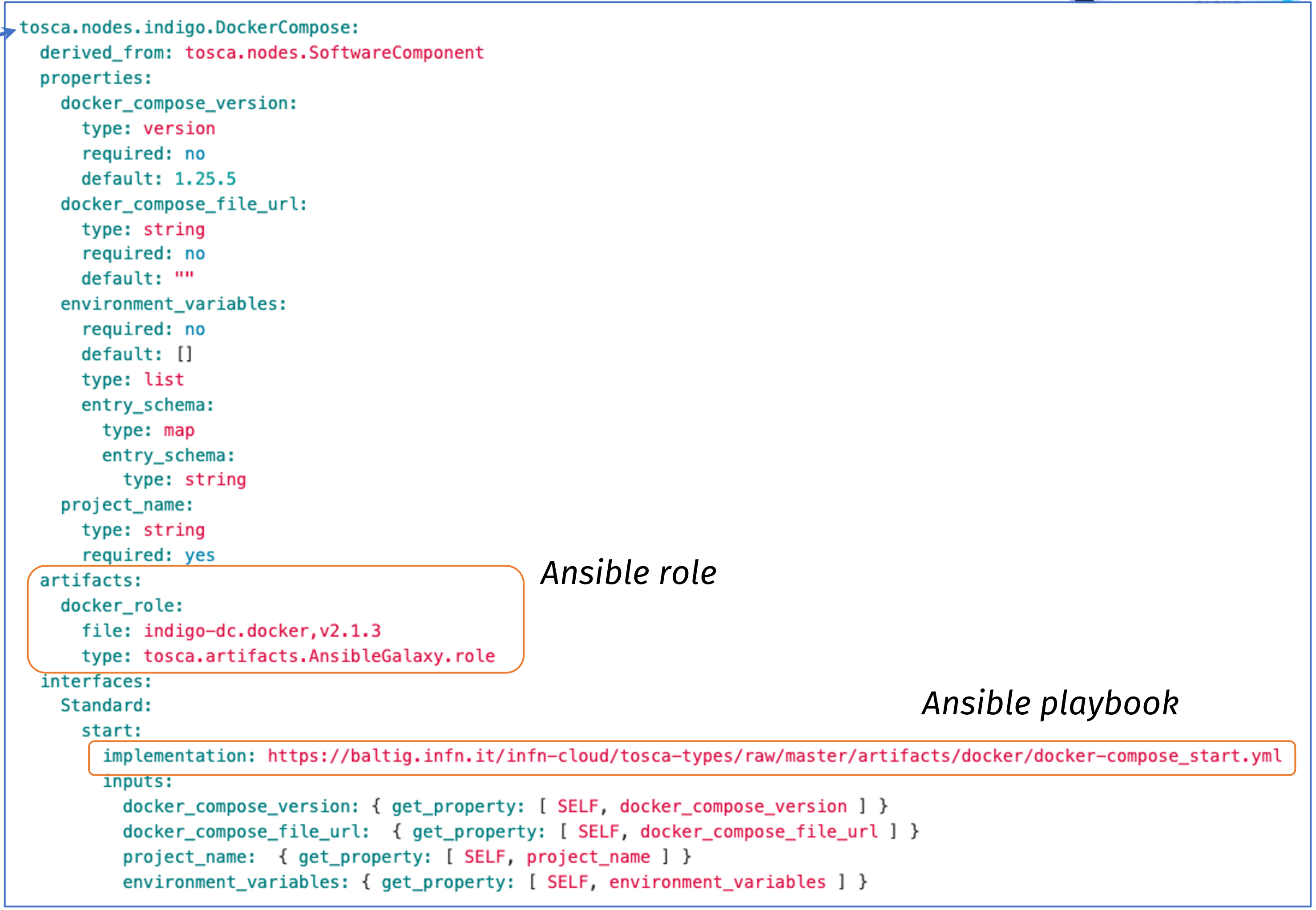
Environment variables

CANCEL ⃠ ← Back CONTINUE →

TOSCA definition



```
docker_compose_service:
  type: tosca.nodes.indigo.DockerCompose
  properties:
    project_name:  { get_input: project_name }
    docker_compose_file_url: { get_input: docker_compose_file_url }
    environment_variables: { get_input: environment_variables }
  requirements:
    - host: server

server:
  type: tosca.nodes.indigo.Compute
  properties:
    os_users: { get_input: users }
  capabilities:
    endpoint:
      properties:
        ports: { get_input: service_ports }
    host:
      properties:
        num_cpus: { get_input: num_cpus }
        mem_size: { get_input: mem_size }
    os:
      properties:
        distribution: ubuntu
        type: linux
        version: 20.04
```



```
tosca.nodes.indigo.DockerCompose:
  derived_from: tosca.nodes.SoftwareComponent
  properties:
    docker_compose_version:
      type: version
      required: no
      default: 1.25.5
    docker_compose_file_url:
      type: string
      required: no
      default: ""
    environment_variables:
      required: no
      default: []
      type: list
      entry_schema:
        type: map
        entry_schema:
          type: string
    project_name:
      type: string
      required: yes
  artifacts:
    docker_role:
      file: indigo-dc.docker,v2.1.3
      type: tosca.artifacts.AnsibleGalaxy.role
  interfaces:
    Standard:
      start:
        implementation: https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/docker/docker-compose_start.yml
        inputs:
          docker_compose_version: { get_property: [ SELF, docker_compose_version ] }
          docker_compose_file_url:  { get_property: [ SELF, docker_compose_file_url ] }
          project_name:  { get_property: [ SELF, project_name ] }
          environment_variables: { get_property: [ SELF, environment_variables ] }
```

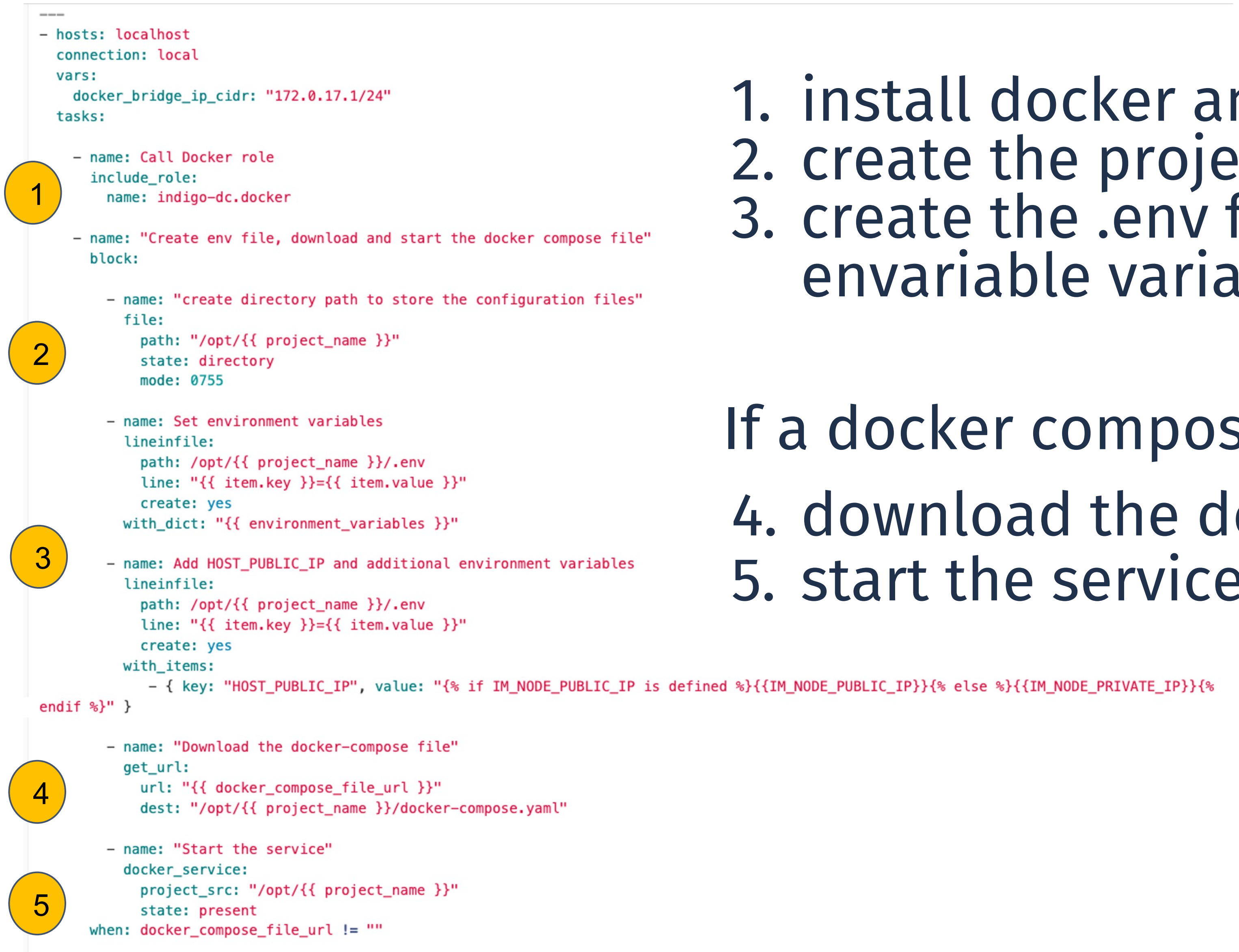
Ansible role

Ansible playbook

https://baltig.infn.it/infn-cloud/tosca-types/-/blob/master/tosca_types/infrastructure/docker_types.yaml

Marica Antonacci (marica.antonacci@ba.infn.it)

The playbook



```
---
- hosts: localhost
  connection: local
  vars:
    docker_bridge_ip_cidr: "172.0.17.1/24"
  tasks:

    - name: Call Docker role
      include_role:
        name: indigo-dc.docker

    - name: "Create env file, download and start the docker compose file"
      block:

        - name: "create directory path to store the configuration files"
          file:
            path: "/opt/{{ project_name }}"
            state: directory
            mode: 0755

        - name: Set environment variables
          lineinfile:
            path: /opt/{{ project_name }}/.env
            line: "{{ item.key }}={{ item.value }}"
            create: yes
          with_dict: "{{ environment_variables }}"

        - name: Add HOST_PUBLIC_IP and additional environment variables
          lineinfile:
            path: /opt/{{ project_name }}/.env
            line: "{{ item.key }}={{ item.value }}"
            create: yes
          with_items:
            - { key: "HOST_PUBLIC_IP", value: "{% if IM_NODE_PUBLIC_IP is defined %}{{IM_NODE_PUBLIC_IP}}{% else %}{{IM_NODE_PRIVATE_IP}}{%
endif %}" }

        - name: "Download the docker-compose file"
          get_url:
            url: "{{ docker_compose_file_url }}"
            dest: "/opt/{{ project_name }}/docker-compose.yaml"

        - name: "Start the service"
          docker_service:
            project_src: "/opt/{{ project_name }}"
            state: present
      when: docker_compose_file_url != ""
```

1. install docker and compose
2. create the project dir
3. create the .env file with all the envariable variables

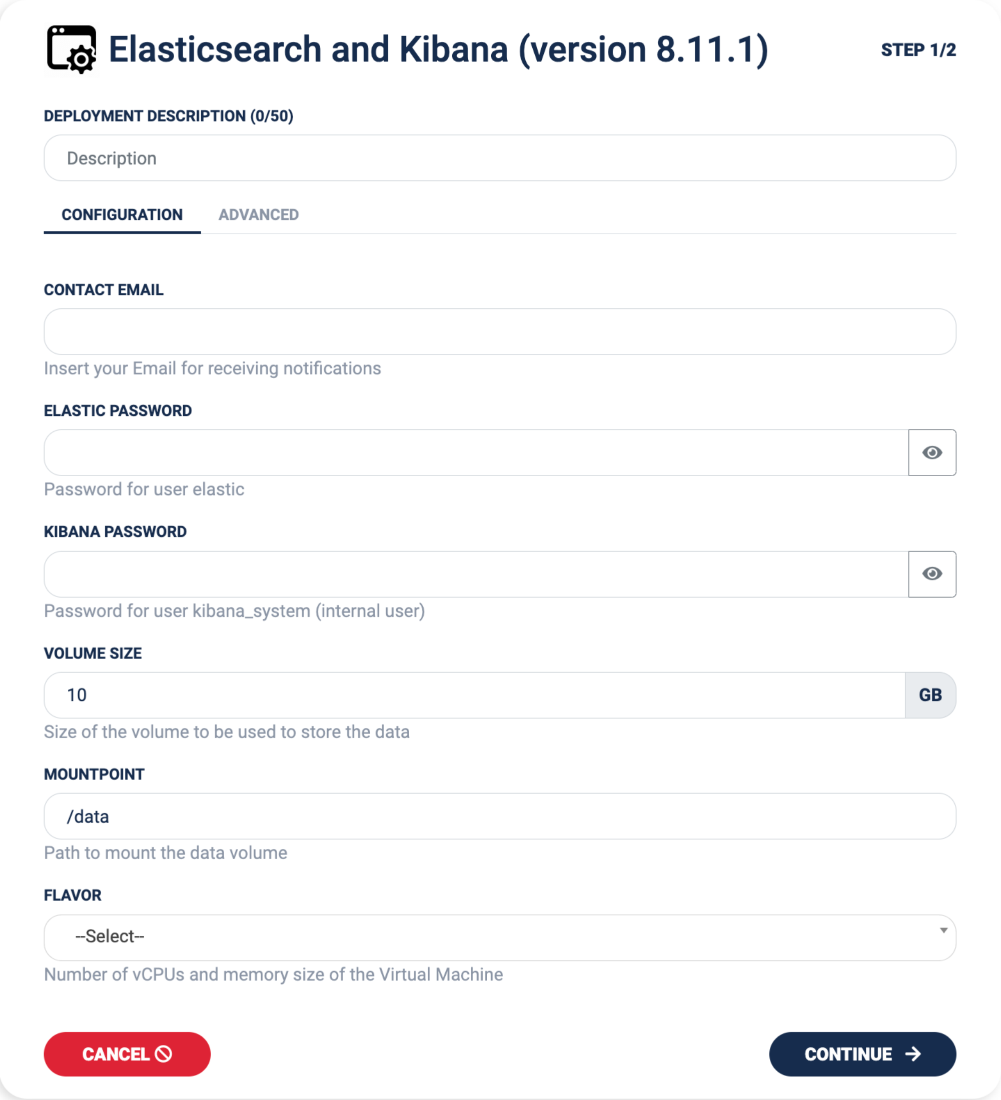
If a docker compose file url is defined:

4. download the docker compose file
5. start the services

EK services implementation

The elasticsearch + kibana (EK) service has been implemented extending the basic docker compose service, deriving the custom type from ***tosca.nodes.indigo.DockerCompose***

EK service implementation



Elasticsearch and Kibana (version 8.11.1)

STEP 1/2

DEPLOYMENT DESCRIPTION (0/50)

Description

CONTACT EMAIL

Insert your Email for receiving notifications

ELASTIC PASSWORD

Password for user elastic

KIBANA PASSWORD

Password for user kibana_system (internal user)

VOLUME SIZE

10 GB

Size of the volume to be used to store the data

MOUNTPOINT

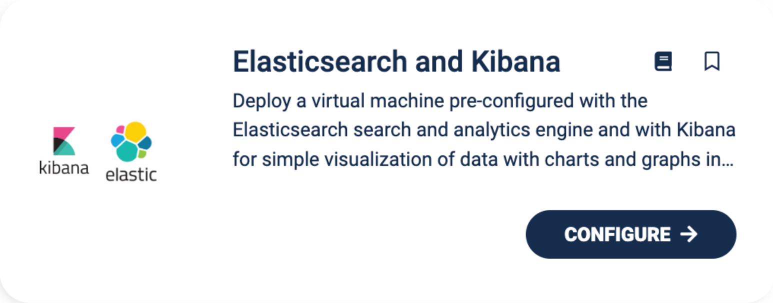
/data

Path to mount the data volume

FLAVOR

--Select--

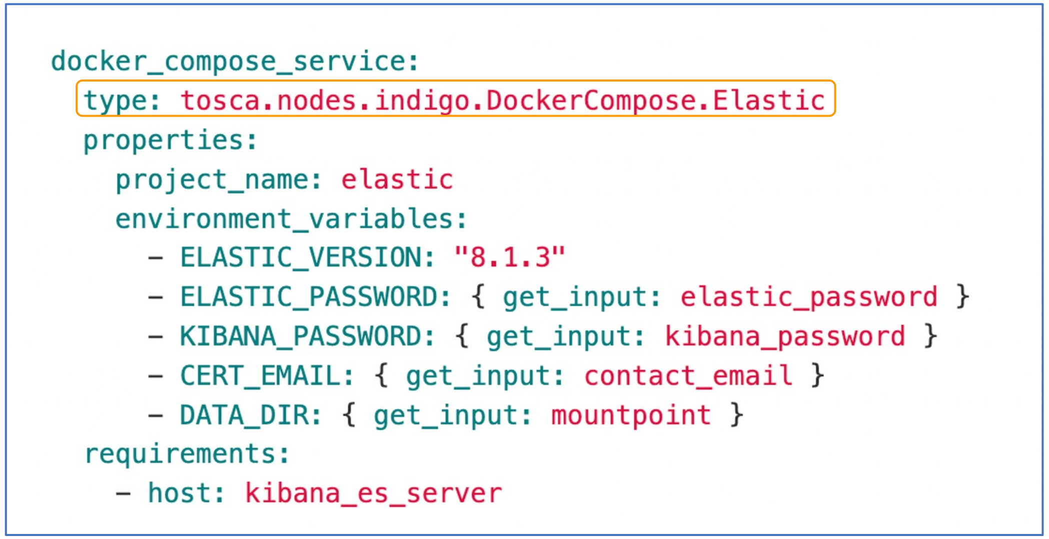
Number of vCPUs and memory size of the Virtual Machine



Elasticsearch and Kibana

Deploy a virtual machine pre-configured with the Elasticsearch search and analytics engine and with Kibana for simple visualization of data with charts and graphs in...

TOSCA template:
https://baltig.infn.it/infn-cloud/tosca-templates/-/blob/master/single-vm/elasticsearch_kibana.yaml



```
docker_compose_service:
  type: tosca.nodes.indigo.DockerCompose.Elastic
  properties:
    project_name: elastic
    environment_variables:
      - ELASTIC_VERSION: "8.1.3"
      - ELASTIC_PASSWORD: { get_input: elastic_password }
      - KIBANA_PASSWORD: { get_input: kibana_password }
      - CERT_EMAIL: { get_input: contact_email }
      - DATA_DIR: { get_input: mountpoint }
  requirements:
    - host: kibana_es_server
```

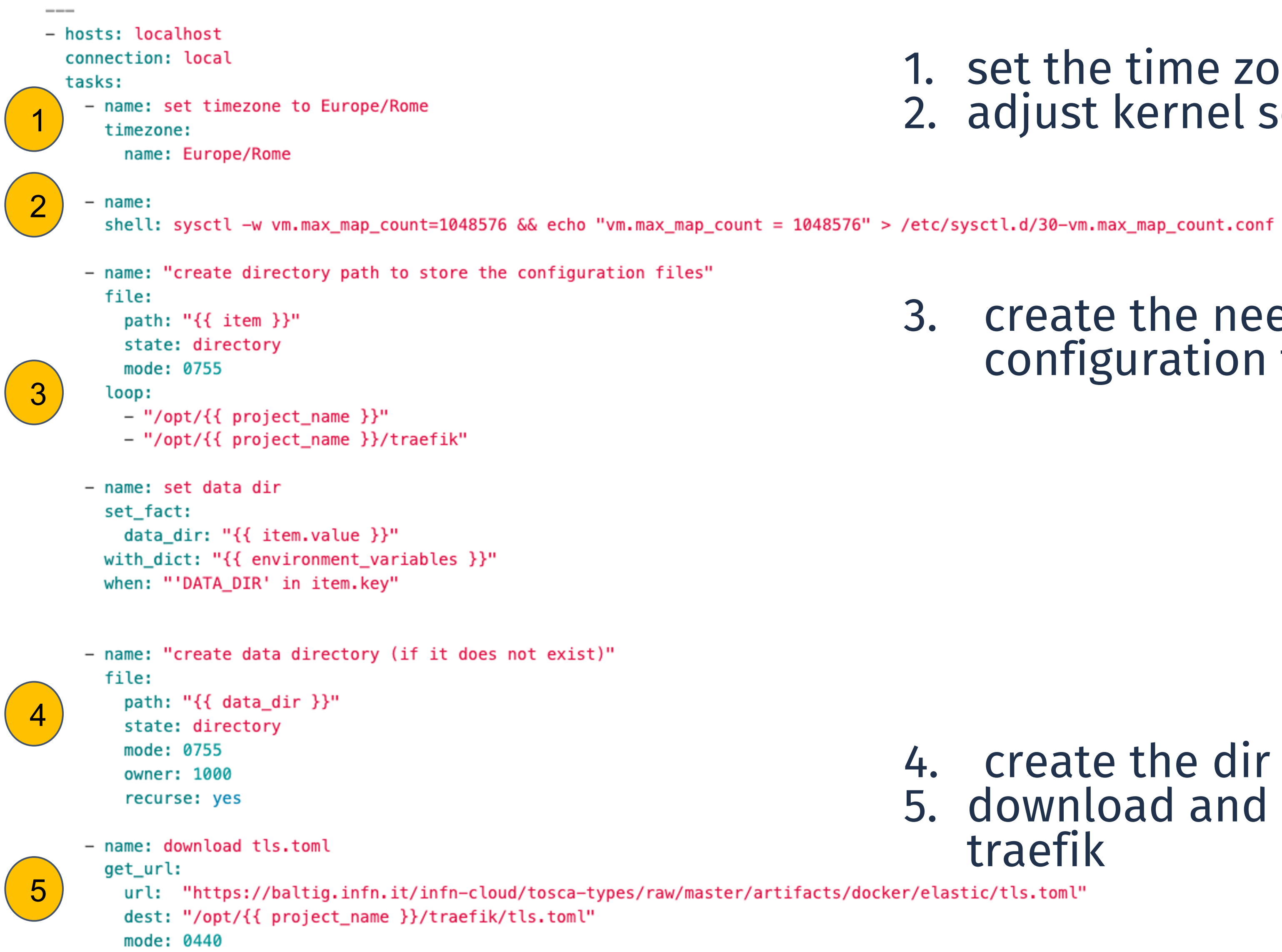
Derived type

```
tosca.nodes.indigo.DockerCompose.Elastic:
  derived_from: tosca.nodes.indigo.DockerCompose
  properties:
    docker_compose_file_url:
      type: string
      default: https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/docker/elastic/docker-compose.yml
  artifacts:
    docker_role:
      file: indigo-dc.docker,v2.1.3
      type: tosca.artifacts.AnsibleGalaxy.role
  interfaces:
    Standard:
      configure:
        implementation: https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/docker/elastic/configure.yml
        inputs:
          project_name:  { get_property: [ SELF, project_name ] }
          environment_variables: { get_property: [ SELF, environment_variables ] }
      start:
        implementation: https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/docker/docker-compose_start.yml
        inputs:
          docker_compose_version: { get_property: [ SELF, docker_compose_version ] }
          docker_compose_file_url:  { get_property: [ SELF, docker_compose_file_url ] }
          project_name:  { get_property: [ SELF, project_name ] }
```

The property *docker_compose_file_url* is overridden providing the default docker compose file. All other properties are inherited by the parent type

The interfaces are specialised too in order to perform custom preliminary configurations (see next slide)

Customized playbook



```
---
- hosts: localhost
  connection: local
  tasks:
    - name: set timezone to Europe/Rome
      timezone:
        name: Europe/Rome

    - name:
      shell: sysctl -w vm.max_map_count=1048576 && echo "vm.max_map_count = 1048576" > /etc/sysctl.d/30-vm.max_map_count.conf

    - name: "create directory path to store the configuration files"
      file:
        path: "{{ item }}"
        state: directory
        mode: 0755
      loop:
        - "/opt/{{ project_name }}"
        - "/opt/{{ project_name }}/traefik"

    - name: set data dir
      set_fact:
        data_dir: "{{ item.value }}"
      with_dict: "{{ environment_variables }}"
      when: "'DATA_DIR' in item.key"


    - name: "create data directory (if it does not exist)"
      file:
        path: "{{ data_dir }}"
        state: directory
        mode: 0755
        owner: 1000
        recurse: yes

    - name: download tls.toml
      get_url:
        url:  "https://baltig.infn.it/infn-cloud/tosca-types/raw/master/artifacts/docker/elastic/tls.toml"
        dest: "/opt/{{ project_name }}/traefik/tls.toml"
        mode: 0440
```

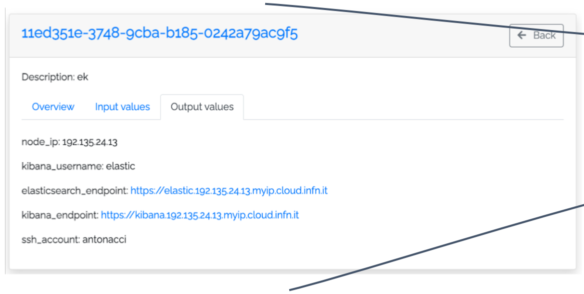
1. set the time zone
2. adjust kernel settings (see doc)

3. create the needed dirs to host configuration files

4. create the dir to store the collected data
5. download and install the TLS settings for traefik

The docker compose file

https://elastic.<IP>.myip.cloud.infn.it



11ed351e-3748-9cba-b185-0242a79ac9f5

Description: ek

Overview | Input values | Output values

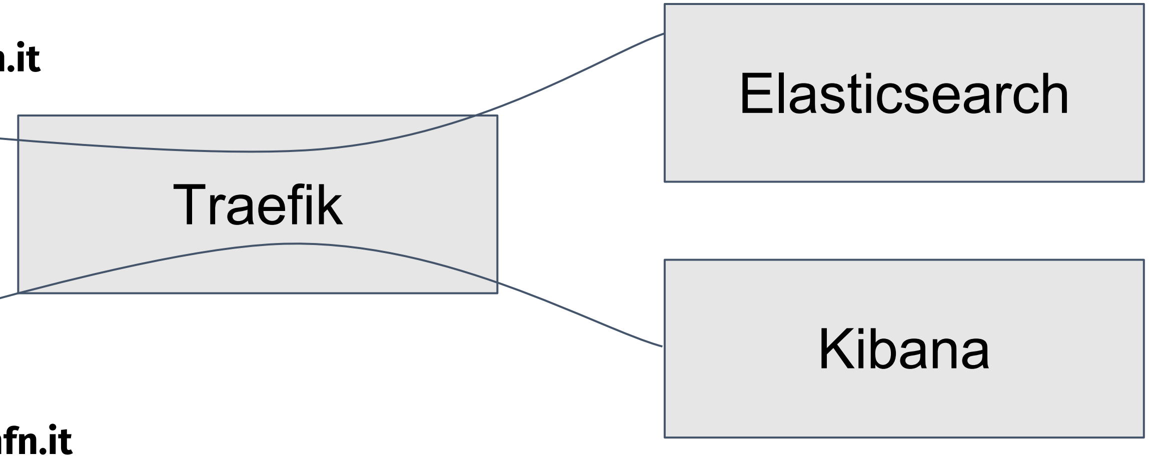
node_ip: 192.135.24.13

kibana_username: elastic

elasticsearch_endpoint: https://elastic.192.135.24.13.myip.cloud.infn.it

kibana_endpoint: https://kibana.192.135.24.13.myip.cloud.infn.it

ssh_account: antonacci



https://kibana.<IP>.myip.cloud.infn.it

Traefik terminates the SSL connections: it is configured to use an ACME provider (Let's Encrypt) for automatic certificate generation.

https://baltig.infn.it/infn-cloud/tosca-types/-/blob/master/artifacts/docker/elastic/compose.yml

Conclusions

- The INFN Cloud PaaS Dashboard makes it easy to discover, select, configure and request the deployment of services that fit the needs and requirements of the INFN research communities.
- New applications and services are continuously included in the catalogue and the Dashboard is enriched with new functionalities to support them.
- Both the addition of a new service in the marketplace and the federation of a new resource provider are quite simple processes, thanks to the flexibility and extensibility of the PaaS architecture and implementation.

Thank you

for your attention!