



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



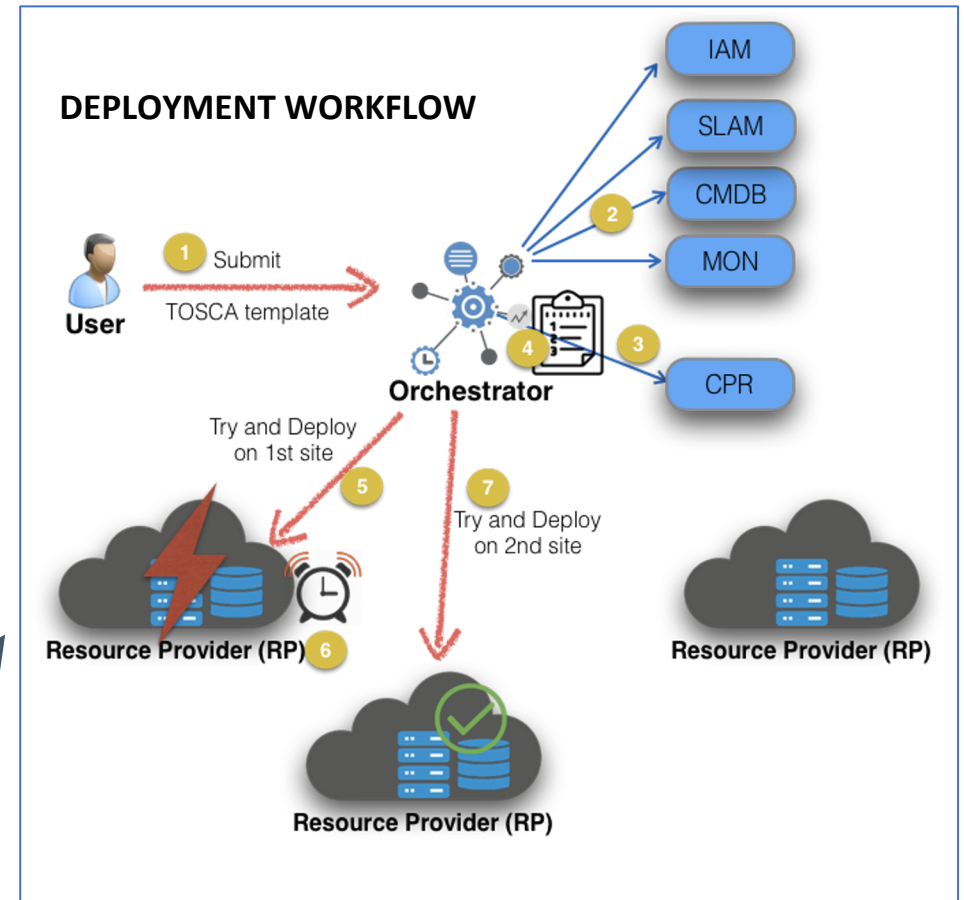
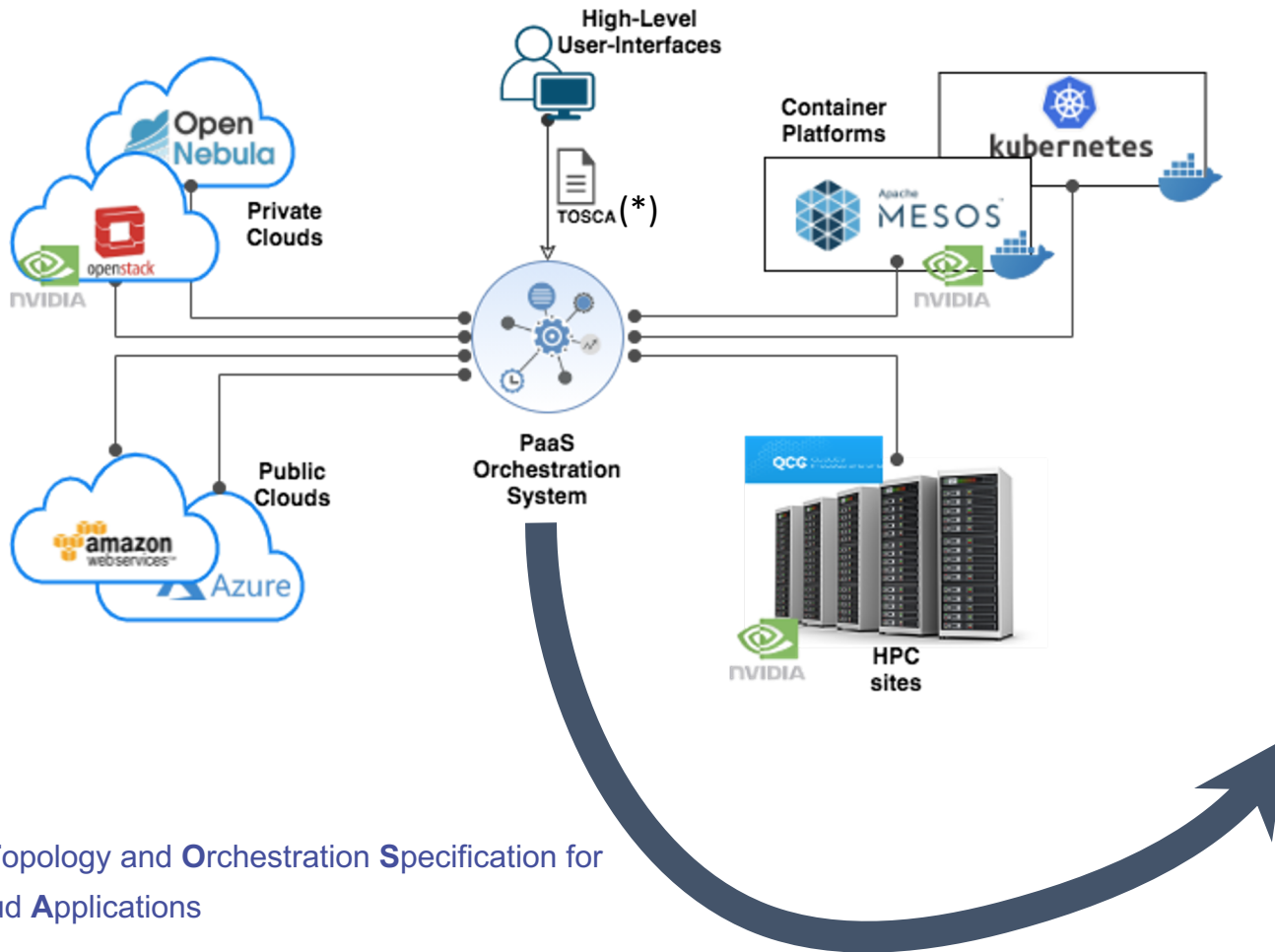
Architecturally INFN Cloud is a **federation** of existing infrastructures

- ❑ **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 Orchestration Specification for Cloud Applications

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**

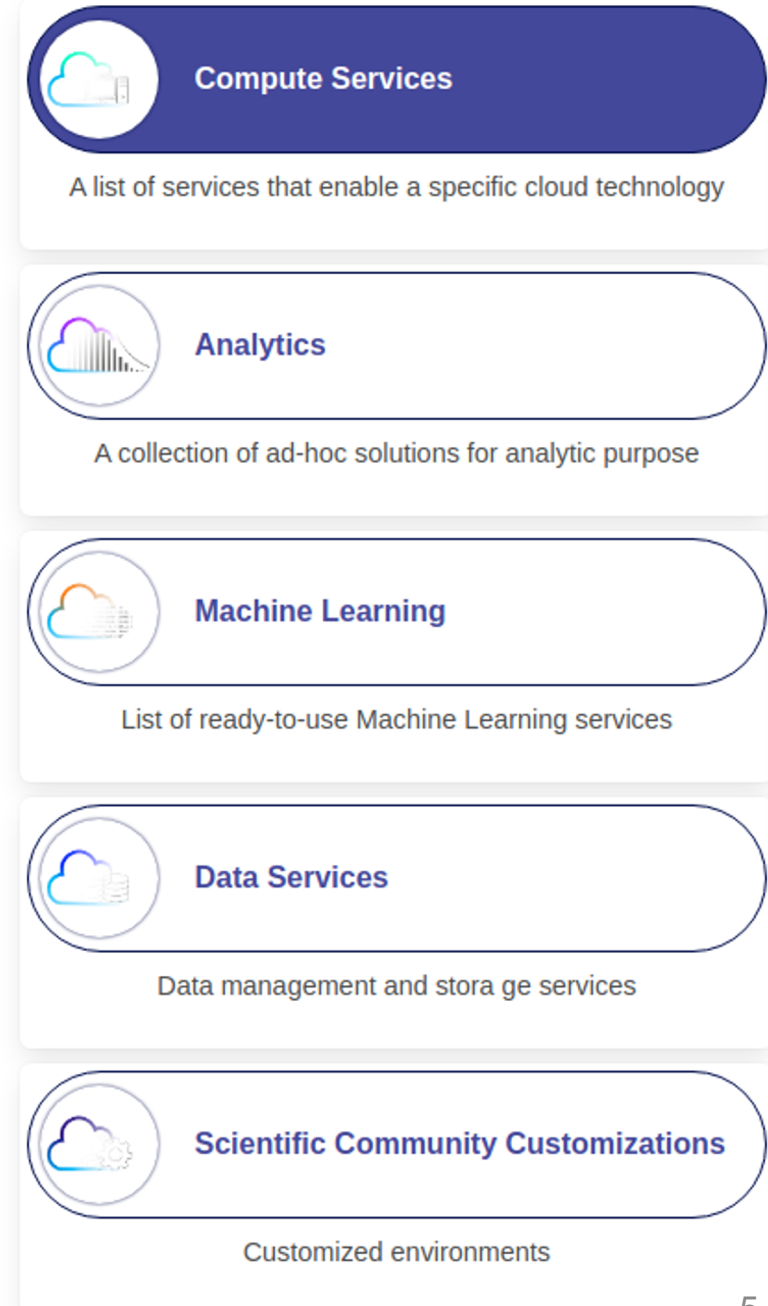
- Spark e/o Elasticsearch e Kibana, R, etc..

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;

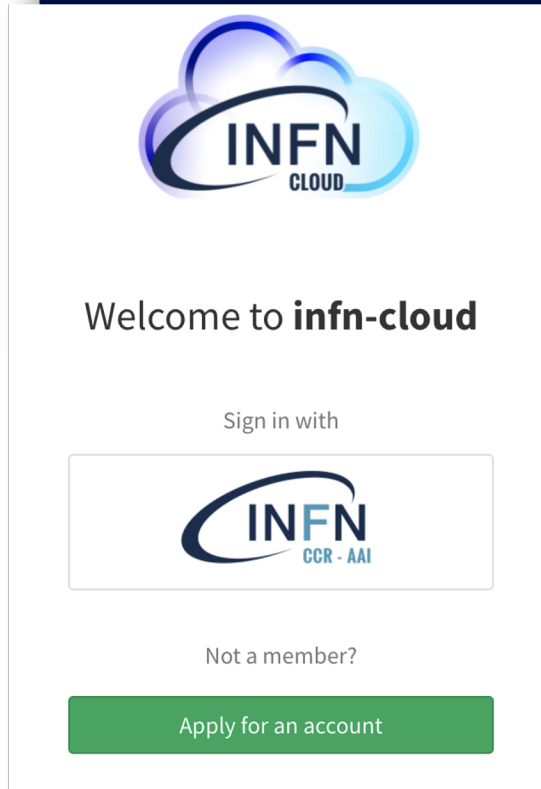
- HTCondor batch system; environment optimized for ML i.e. equipped with GPUs
- Container orchestrators such as K8s and Mesos



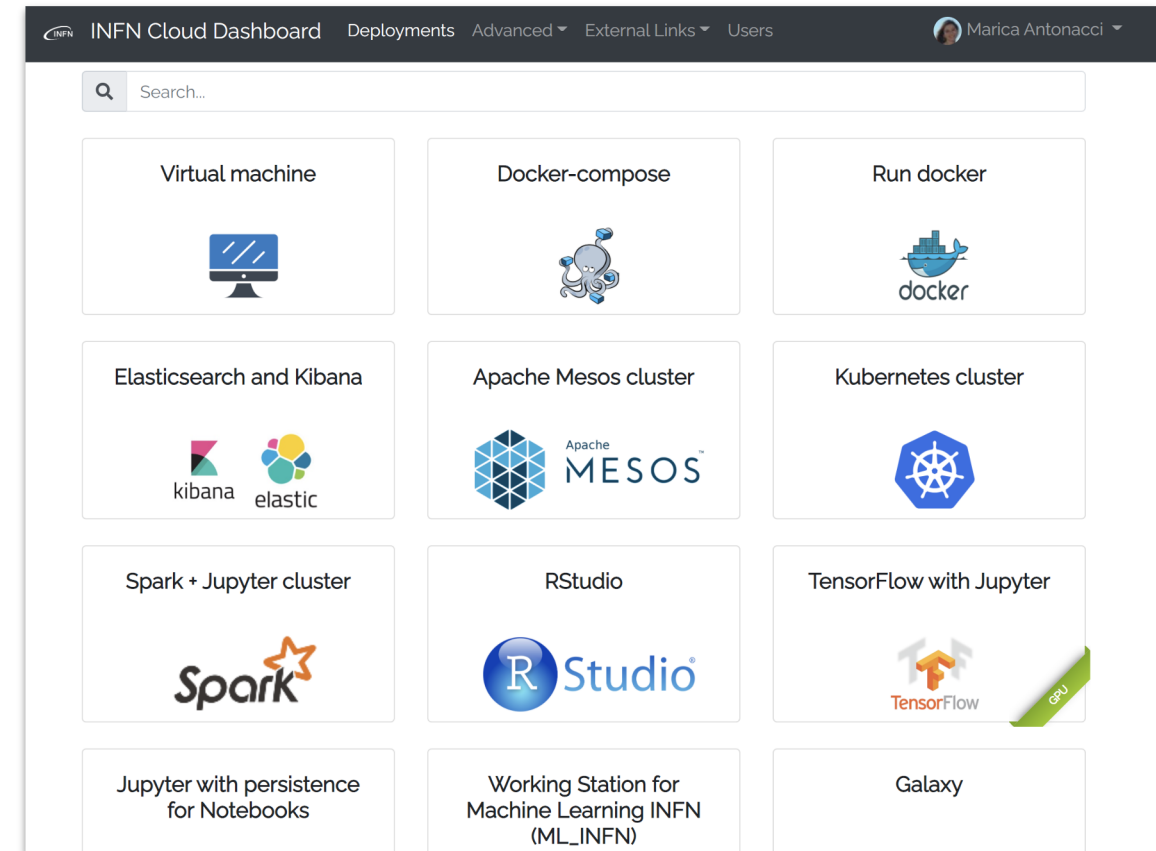
The INFN Cloud Dashboard



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



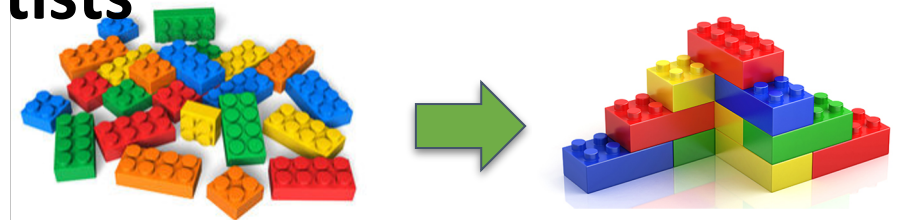
The catalogue is a graphical representation of the TOSCA templates repository that we have been developing extending the INDIGO-DC custom types

- 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

<p>Virtual machine</p> 	<p>Docker-compose</p> 	<p>Run docker</p> 
<p>Elasticsearch and Kibana</p> 	<p>Apache Mesos cluster</p> 	<p>Kubernetes cluster</p> 
<p>Spark + Jupyter cluster</p> 	<p>HTCondor cluster</p> 	<p>RStudio</p> 
<p>TensorFlow with Jupyter</p> 	<p>Jupyter with persistence for Notebooks</p> 	<p>Computational environment for Machine Learning INFN (ML-INFN)</p> 
<p>Working Station for CYGNO experiment</p> 	<p>Sync&Share aaS</p> 	

Run docker



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

Run docker

Description: Run a docker container

Deployment description

description

Configuration **Advanced**

num_cpus
1
Number of virtual cpus for the VM

mem_size
2
Amount of memory for the VM

docker_appname
nginx
Name to be assigned to the container

docker_image
nginx
Name of the image used to create the container

docker_tag
latest
Tag of the image used to create the container

ports_mapping

List of ports to publish from the container to the host. Use docker CLI syntax: 8000, or 9000:8000, where 8000 is a container port, 9000 host port

docker_command
Command to execute when the container starts

service_ports

Ports to open on the VM to access the service(s)

environment_variables

Environment variables (key,value pairs)

My deployments

Show 10 entries Search:

Description ↑↓	Deployment identifier ↑↓	Status ↑↓	Creation time ↑↓	Deployed at ↑↓	Actions ↑↓
nginx	11ebcf73-a1a1-dc3d-a7b8-0242699101a7	CREATE_COMPLETE	2021-06-17 13:55:00	RECAS-BARI	<input type="button" value="Details"/>

11ebcf73-a1a1-dc3d-a7b8-0242699101a7

Description: nginx

[Overview](#) [Input values](#) [Output values](#)

node_ip: 212.189.205.23

ssh_account: antonacci



How to su guides.cloud.infn.it



The sidebar of the INFN Cloud documentation website. At the top, it says 'INFN Cloud' with a home icon and the INFN CLOUD logo. Below the logo is a search bar labeled 'Search docs'. A 'TABLE OF CONTENTS' section lists various guides, with 'How To: Instantiate docker containers using docker run' selected and highlighted in a light blue box. At the bottom, there is a 'Read the Docs' icon and a version selector set to 'latest'.

Docs » How To: Instantiate docker containers using docker run

[View page source](#)

How To: Instantiate docker containers using docker run

How To: Deploy a MySQL Server application with Run docker

Author: Alessandro Costantini

Version: 1

Copyright: This document has been placed in the public domain.

Contents

- How To: Instantiate docker containers using docker run
 - 1. Prerequisites
 - 2. How to deploy a MySQL Server with Run docker
 - Step 1 - Connecting and authenticating to the INFN-CLOUD dashboard
 - Step 2 - Select and Configure the Run docker deployment
 - Step 3 - Submitting the Run docker Deployment
 - Step 4 - Operate with the deployed MySQL-Server application

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

1. Prerequisites

The user has to be registered in the IAM system for INFN-CLOUD <https://iam.cloud.infn.it/login>. Only registered users can login into the INFN-CLOUD dashboard <https://my.cloud.infn.it/login>.

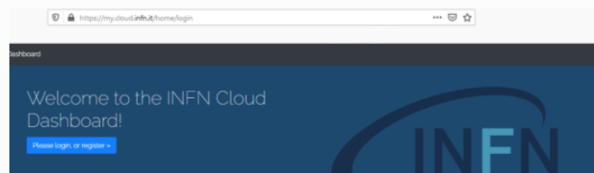
The access to the INFN-CLOUD dashboard enable the user to instantiate Docker Run.

2. How to deploy a MySQL Server with Run docker

Run docker is an implementation of Docker to run docker containers.

Step 1 - Connecting and authenticating to the INFN-CLOUD dashboard

Connecting to the INFN-CLOUD dashboard (<https://my.cloud.infn.it/>), the user can authenticate with the credentials used for the IAM account (<https://iam.cloud.infn.it/login>) in order to access the dashboard.



Docker-compose



Docker-compose use-case

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

Configure your service

Select

Configure docker storage on the VM root filesystem
 Configure docker storage on an external volume attached to the VM

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

Docker-compose

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

Deployment description

General Services Advanced

environment_variables

Environment variables

docker_compose_file_url




URL of the docker compose file to deploy

project_name

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

Environment variables management

environment_variables

Key	Value	
DB_USER	wp	
DB_ROOT_PASSWORD	1234qwer	
DB_USER_PASSWORD	3456erty	

Environment variables

- 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

Description: Run a docker compose file fetched from the specified URL

Deployment description
wordpress

General | **Ports** | Advanced

service_ports

Protocol	Port Range	Source	
TCP ▾	80	0.0.0.0/0	Remove
TCP ▾	443	0.0.0.0/0	Remove

[Add rule](#)

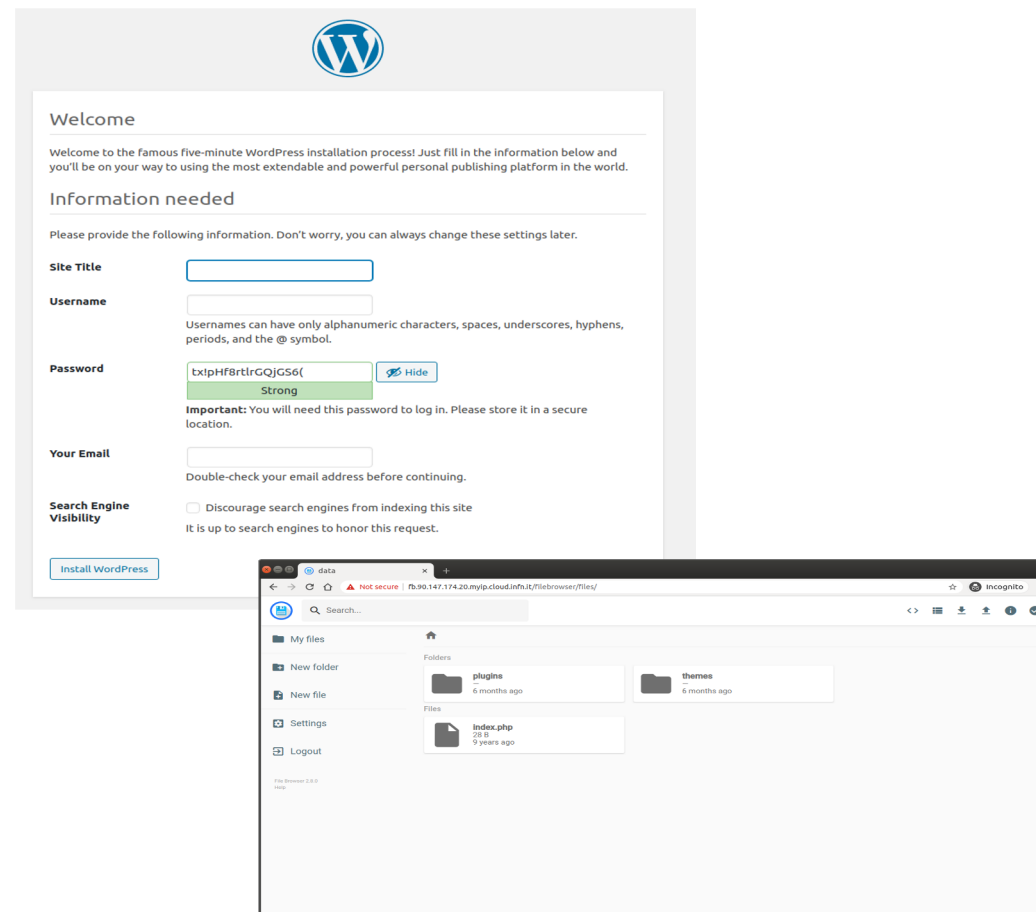
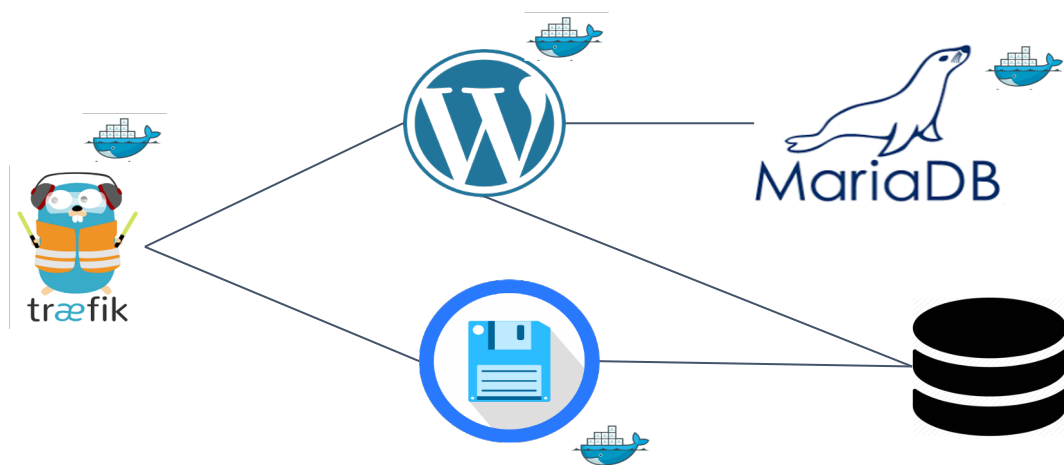
Ports to open to access the service(s)

[Submit](#) [Cancel](#)

Docker compose example

<https://baltig.infn.it/inf-n-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



INFN Cloud

latest

TABLE OF CONTENTS

- Getting Started
- How To: Create VM with ssh access
- How To: Configure the backup on your deployment
- How To: Deploy Sync&Share aaS
- How To: Deploy a Kubernetes cluster
- How To: Deploy an Apache Mesos cluster
- How To: Deploy a Spark cluster + Jupyter notebook
- How To: Deploy Elasticsearch & Kibana
- How To: Deploy RStudio Server

How To: Instantiate docker containers using custom docker-compose files

- Prerequisites
- How to deploy and access docker-compose

How To: Instantiate docker containers using docker run

How To: Access cloud storage from a scientific environment

Read the Docs v: latest

Docs » How To: Instantiate docker containers using custom docker-compose files

[View page source](#)

How To: Instantiate docker containers using custom docker-compose files

Table of Contents

- How To: Instantiate docker containers using custom docker-compose files
 - Prerequisites
 - How to deploy and access docker-compose
 - Step 1 - Connecting and authenticating to the INFN-CLOUD dashboard
 - Step 2 - Select and Configure the docker-compose deployment
 - Step 3 - Submitting the Docker-compose deployment
 - Step 4 - Access your application

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

Prerequisites

Make sure you are registered to the IAM system for INFN-CLOUD <https://iam.cloud.infn.it/login>. Only registered users can login into the INFN-CLOUD dashboard <https://paas.cloud.infn.it/login>.

Access to the INFN-CLOUD dashboard enables users to instantiate the "docker compose" deployment.

How to deploy and access docker-compose

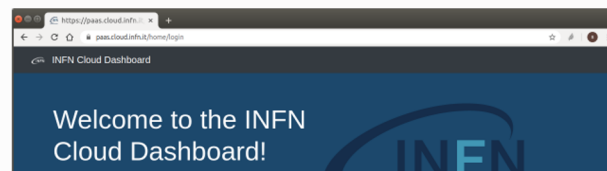
Docker-compose allows you to instantiate fully functional production level services by using a set of coordinated dockers.

The public IP Address of the VM hosting the docker containers is available to the docker-compose file as an environment variable: HOST_PUBLIC_IP

Step 1 - Connecting and authenticating to the INFN-CLOUD dashboard

Connect to the INFN-CLOUD dashboard (<https://paas.cloud.infn.it/>).

You can authenticate with the credentials used for the IAM account (<https://iam.cloud.infn.it/login>) in order to access the dashboard.



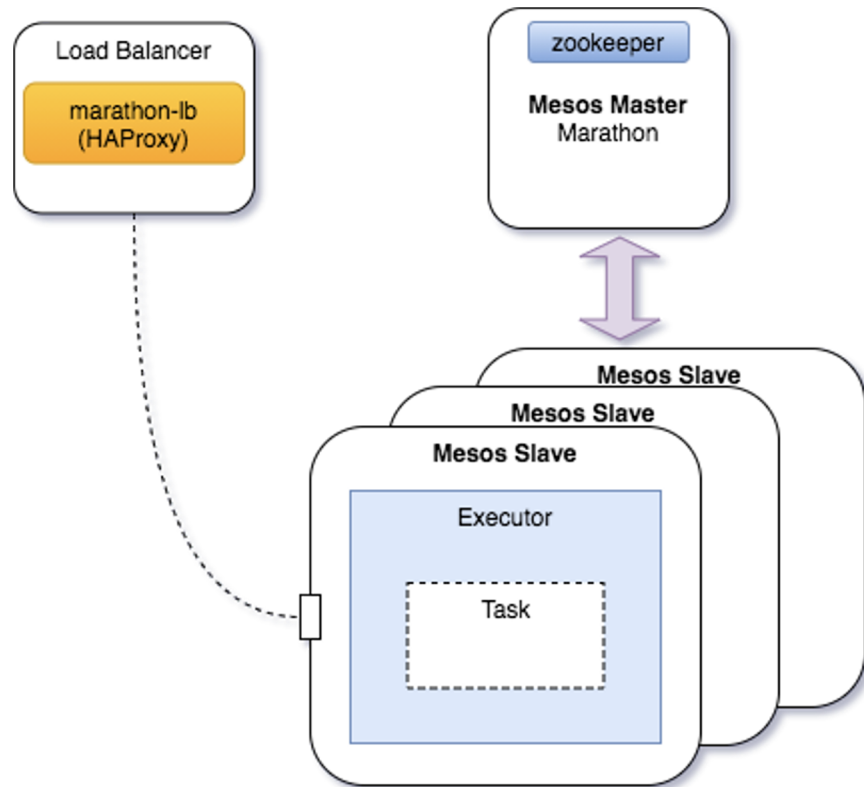
Apache Mesos cluster



Mesos use-case

How to deploy a complete Mesos cluster

Cluster architecture



Apache Mesos cluster

Description: Apache Mesos abstracts CPU, memory, storage, and other compute resources away from machines (physical or virtual), enabling fault-tolerant and elastic distributed systems to easily be built and run effectively

Deployment description

mesos cluster

Configuration [Advanced](#)

mesos_password

.....

Admin password for accessing Mesos HTTP service

marathon_password

.....

Admin password for accessing Marathon HTTP service

slave_num

2

Number of slave nodes in the cluster

admin_email

antonacci@inf.it

Admin email address

master_flavor

medium: 2 VCPUs, 4 GB RAM

Number of vCPUs and memory size of the Master Virtual Machine

slave_flavor

large: 4 VCPUs, 8 GB RAM

Number of vCPUs and memory size of each Slave Virtual Machine

Submit

Cancel

Deployment outputs

11ec2d02-2c66-02ae-edef-0242699101a7

← Back

Description: mesos cluster

Overview

Input values

Output values

mesos_lb_ip: ['90.147.75.69']

mesos_endpoint: <https://90.147.75.68.myip.cloud.infn.it:5050>

marathon_endpoint: <https://90.147.75.68.myip.cloud.infn.it:8443>

mesos_master: ['90.147.75.68']

ssh_account: antonacci

Mesos/Marathon



Apache MESOS Frameworks Agents Roles Offers Maintenance IndigoCluster

Master 67907213-52e8-4c2e-851e-61031f26f144

Cluster: IndigoCluster
Leader: 192.168.100.44:5050
Version: 1.9.0
Built: a year ago by ubuntu
Started: 12 hours ago
Elected: 12 hours ago

Leading Master Log: [Download](#) [View](#)

Agents

Activated	2
Deactivated	0
Unreachable	0

Tasks

Staging	0
Starting	0
Running	0
Unreachable	0
Killing	0
Finished	0
Killed	0
Failed	0
Lost	0

Resources

	CPUs	GPUs	Mem	Disk
Total	4	0	5.7 GB	28.6 GB
Allocated	0	0	0 B	0 B
Offered	0	0	0 B	0 B
Idle	4	0	5.7 GB	28.6 GB

Active Tasks

Framework ID	Task ID	Task Name	Role	State	Health	Started ▼	Host
No active tasks.							

Unreachable Tasks

Framework ID	Task ID	Task Name	Role
No unreachable tasks.			

Completed Tasks

Framework ID	Task ID	Task Name	Role	State	Star
No completed tasks.					

MARATHON Applications Deployments

Search all application

Create Group Create Application

Applications

Name	CPU	Memory	Status	Running Instances	Health
No Applications Created					

Do more with Marathon by creating and organizing your applications.

Create Application

- Running
- Deploying
- Suspended
- Delayed
- Waiting

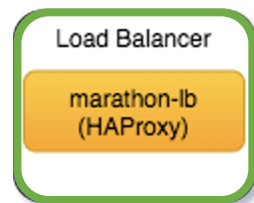
HEALTH

- Healthy
- Unhealthy
- Unknown

RESOURCES

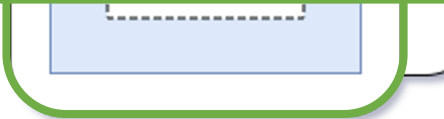
- Volumes

TOSCA Template: nodes



```
mesos_master:  
  type: toasca.nodes.indigo.MesosMaster  
  properties:  
    mesos_masters_list: { get_attribute: [ HOST, private_address ] }  
    mesos_password: { get_input: mesos_password }  
    marathon_password: { get_input: marathon_password }  
    chronos_password: { get_input: chronos_password }  
  else  
    r_server
```

```
mesos_load_balancer:  
  type: toasca.nodes.indigo.MesosLoadBalancer  
  properties:  
    master_ips: { get_attribute: [ mesos_master_server, private_address ] }  
    marathon_password: { get_input: marathon_password }  
    enable_consul_sd: false  
  requirements:  
    - host: mesos_lb_server
```



```
Slave  
  properties:  
    master_ips: { get_attribute: [ mesos_master_server, private_address ] }  
    front_end_ip: { get_attribute: [ mesos_master_server, private_address, 0 ] }  
    enable_consul_sd: false  
  requirements:  
    - host: mesos_slave_server
```

https://baltig.infn.it/inf-n-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



INFN Cloud

Istituto Nazionale di Fisica Nucleare

latest

Search docs

TABLE OF CONTENTS

- Getting Started
- How To: Create VM with ssh access
- How To: Configure the backup on your deployment
- How To: Deploy the Cloud Storage Service
- How To: Deploy a Kubernetes cluster
- How To: Deploy an Apache Mesos cluster**
- Prerequisites
- Apache Mesos cluster configuration
- Deployment result
- How To: Deploy a Spark cluster + Jupyter notebook
- How To: Deploy Elasticsearch & Kibana
- How To: Deploy RStudio application
- How To: Instantiate docker containers using custom docker-compose files
- How To: Instantiate docker containers using docker run
- How To: Access cloud storage from a scientific environment

Docs » How To: Deploy an Apache Mesos cluster

[View page source](#)

How To: Deploy an Apache Mesos cluster

Table of Contents

- [How To: Deploy an Apache Mesos cluster](#)
 - [Prerequisites](#)
 - [Apache Mesos cluster configuration](#)
 - [Basic configuration](#)
 - [Advanced configuration](#)
 - [Deployment result](#)
 - [Troubleshooting](#)

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

Prerequisites

The user has to be registered in the IAM system for INFN-CC <https://iam.cloud.infn.it/login>. Only registered users can login into the INFN-CC cloud dashboard <https://paas.cloud.infn.it/login>. (Through the INFN-CC cloud dashboard users can instantiate in a simple way infrastructures that cover different use cases, since the simple VM up to cluster for big data processing.)

Apache Mesos cluster configuration

After the login to the dashboard, selecting the "Apache Mesos cluster" button, then press configure. The configuration menu is shown.

Parameters are splitted in two pages: "Configuration" and "Advanced"

Basic configuration

The user has to fill the correct parameters like the description of the cluster, the mesos password for admin user of the cluster, the marathon password for admin user of marathon component and the chronos admin password for chronos component. Other parameters has a default loaded: 1 master with 2 cpus and 4GB RAM and 2 slaves both with 1 CPU and 2GB RAM each one. By default the provider where the cluster will be instantiated is automatically selected by PaaS system so the Orchestrator will deploy the cluster where bet fit you previous choice.

When all parameters has been set the use can submit the cluster.

WILL BE AVAILABLE SOON



RStudio use-case

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

Configure the service



RStudio

RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports [...]

[Read More](#) [Configure](#)

1

Click on *Configure* button

RStudio

Description: RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

Deployment description

description

Configuration [Advanced](#)

cpus

1.0

Amount of CPUs for this service

mem

1 GB

Amount of Memory for this service

rstudio_password

Password for user with username rstudio

[Submit](#) [Cancel](#)

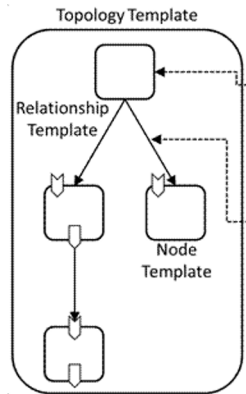
2

Set a few config parameters

3

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

3

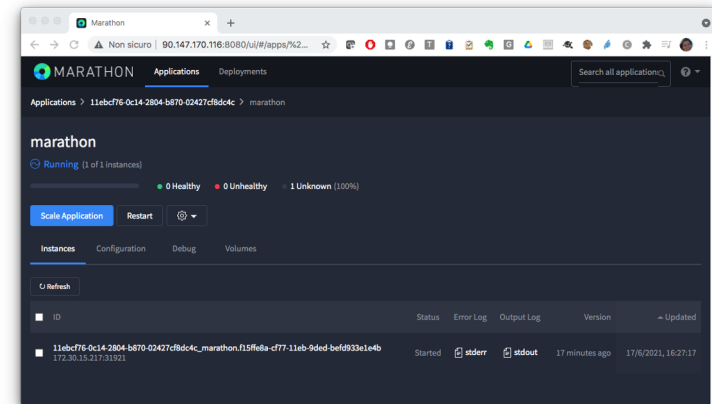
The template is translated into the Marathon application definition (json)

{JSON}

```
{
  "type": "DOCKER",
  "docker": {
    "forcePullImage": true,
    "image": "rocker/rstudio",
    "parameters": {},
    "privileged": false
  },
  "volumes": [
    {
      "containerPath": "/home/rstudio",
      "external": {
        "name": "11ebc7f6-8c14-2804-b870-02427cf8dc4c-marathon",
        "provider": "dvc",
        "options": {
          "dvdL/driver": "rexray"
        }
      }
    }
  ],
  "node": "RM",
  "portMappings": [
    {
      "containerPort": 8787,
      "hostPort": 0,
      "protocol": "tcp",
      "servicePort": 10001
    }
  ]
}
```

4

The request is submitted to Marathon/Mesos cluster



From TOSCA to Marathon App definition in json

```
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' ] } ]
```

```
{
  "id": "/11ebcf76-0c14-2804-b870-02427cf8dc4c",
  "apps": [
    {
      "id": "marathon",
      "instances": 1,
      "cpus": 1.0,
      "mem": 1000.0,
      "uris": [],
      "constraints": [],
      "container": {
        "type": "DOCKER",
        "docker": {
          "image": "rocker/rstudio",
          "network": "BRIDGE",
          "forcePullImage": true,
          "portMappings": [
            {
              "containerPort": 8787,
              "protocol": "tcp",
              "labels": {}
            }
          ],
          "privileged": false
        },
        "volumes": [
          {
            "external": {
              "name": "11ebcf76-0c14-2804-b870-02427cf8dc4c-marathon",
              "provider": "dvdi",
              "options": {
                "dvdi/driver": "rexray"
              }
            },
            "containerPath": "/home/rstudio",
            "mode": "RW"
          }
        ]
      },
      "env": {
        "PASSWORD": "marathon@123456789"
      },
      "labels": {
        "origin": "https://indigo-paas.cloud.ba.infn.it/orchestrator",
        "HAPROXY_GROUP": "external",
        "created_by": "2230db43-9929-4ec7-bba0-98731d511058@https://iam-test.indigo-datacloud.eu/"
      }
    }
  ]
}
```

Access your environment



11ebcf76-0c14-2804-b870-02427cf8dc4c

← Back

Description: rstudio

Overview Input values Output values

endpoint: <http://mesos-lb.cloud.ba.infn.it:10001>

username: rstudio

RStudio Sign In

Non sicuro | mesos-lb.cloud.ba.infn.it...

Studio

Sign in to RStudio

Username:

Password:

Stay signed in when browser closes

You will automatically be signed out after 60 minutes of inactivity.

Sign In

RStudio Server

Non sicuro | mesos-lb.cloud.ba.infn.it...

RStudio

File Edit Code View Plots Session Build Debug Profile Tools Help

rstudio Project: (None)

Console Terminal Jobs

```
R 4.1.0 ~-/>
* DONE (forcats)
* installing *binary* package 'ggplot2' ...
* DONE (ggplot2)
* installing *binary* package 'httr' ...
* DONE (httr)
* installing *binary* package 'readr' ...
* DONE (readr)
* installing *binary* package 'reprex' ...
* DONE (reprex)
* installing *binary* package 'nycflights13' ...
* DONE (nycflights13)
* installing *binary* package 'gargle' ...
* DONE (gargle)
* installing *binary* package 'dbplyr' ...
* DONE (dbplyr)
* installing *binary* package 'dtplyr' ...
* DONE (dtplyr)
* installing *binary* package 'haven' ...
* DONE (haven)
* installing *binary* package 'readxl' ...
* DONE (readxl)
* installing *binary* package 'rvest' ...
* DONE (rvest)
* installing *binary* package 'tidyr' ...
* DONE (tidyr)
* installing *binary* package 'broom' ...
* DONE (broom)
* installing *binary* package 'janitor' ...
* DONE (janitor)
```

Environment History Connections Tutorial

R - Global Environment

Environment is empty

Files Plots Packages Help Viewer

Name	Description	Version
System Library		
<input type="checkbox"/> askpass	Safe Password Entry for R, Git, and SSH	1.1
<input type="checkbox"/> assertthat	Easy Pre and Post Assertions	0.2.1
<input type="checkbox"/> backports	Reimplementations of Functions Introduced Since R-3.0.0	1.2.1
<input checked="" type="checkbox"/> base	The R Base Package	4.1.0
<input type="checkbox"/> base64enc	Tools for base64 encoding	0.1-3
<input type="checkbox"/> BH	Boost C++ Header Files	1.75.0-0
<input type="checkbox"/> blob	A Simple S3 Class for Representing Vectors of Binary Data ('BLOBS')	1.2.1
<input type="checkbox"/> boot	Bootstrap Functions (Originally by Angelo Canty for S)	1.3-28
<input type="checkbox"/> broom	Convert Statistical Objects into Tidy Tibbles	0.7.7
<input type="checkbox"/> bslib	Custom 'Bootstrap' 'Sass' Themes for 'shiny' and 'rmarkdown'	0.2.5.1
<input type="checkbox"/> cachem	Cache R Objects with Automatic Pruning	1.0.5

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

How to @ guides.cloud.infn.it



INFN Cloud

latest

Search docs

TABLE OF CONTENTS

- Getting Started
- How To: Create VM with ssh access
- How To: Configure the backup on your deployment
- How To: Deploy Sync&Share aaS
- How To: Deploy a Kubernetes cluster
- How To: Deploy an Apache Mesos cluster
- How To: Deploy a Spark cluster + Jupyter notebook
- How To: Deploy Elasticsearch & Kibana

How To: Deploy RStudio Server

- 1. The RStudio Server
- 2. Prerequisites
- 3. Notes for the reader
- 4. How to deploy and access RStudio Server

- How To: Instantiate docker containers using custom docker-compose files
- How To: Instantiate docker containers using docker run
- How To: Access cloud storage from a scientific environment

Read the Docs v: latest

Docs » How To: Deploy RStudio Server

[View page source](#)

How To: Deploy RStudio Server

Author: Alessandro Costantini
Version: 1
Copyright: This document has been placed in the public domain.

Contents

- How To: Deploy RStudio Server
 - 1. The RStudio Server
 - 2. Prerequisites
 - 3. Notes for the reader
 - 4. How to deploy and access RStudio Server
 - Step 1 - Connecting and authenticating to the INFN-CLOUD dashboard
 - Step 2 - Select and Configure the RStudio deployment
 - Step 3 - Submitting the RStudio Deployment
 - Step 4 - Operate with the deployed RStudio Server

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

1. The RStudio Server

The following procedure will guide you into the deployment of the self-consistent RStudio server. RStudio is an integrated development environment (IDE) for R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

2. Prerequisites

The user has to be registered in the INFN Cloud Identity and Access management (IAM) system, <https://iam.cloud.infn.it/> in order to access the INFN-CLOUD dashboard, <https://my.cloud.infn.it/>.

The access to the INFN-CLOUD dashboard enables the user to instantiate a VM containing the RStudio server and use the environment it provides.

3. Notes for the reader

The current deployment does not support the GPU implementation. Only CPU implementation is available. Selecting providers with GPU resources (see **Step 2 - Select and Configure the RStudio deployment**) does not enable the use of GPU in the deployed RStudio application.

4. How to deploy and access RStudio Server

RStudio is an integrated development environment (IDE) for R that includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

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



TensorFlow with Jupyter

Description: Run an instance of Tensorflow with GPU

Deployment description

tf

Configuration [Advanced](#)

cpus

2.0

Amount of CPUs for this service

mem

4 GB

Amount of Memory for this service

gpus

1

Amount of GPUs for this service

jupyter_password

.....

Set password for Jupyter

```
node_templates:

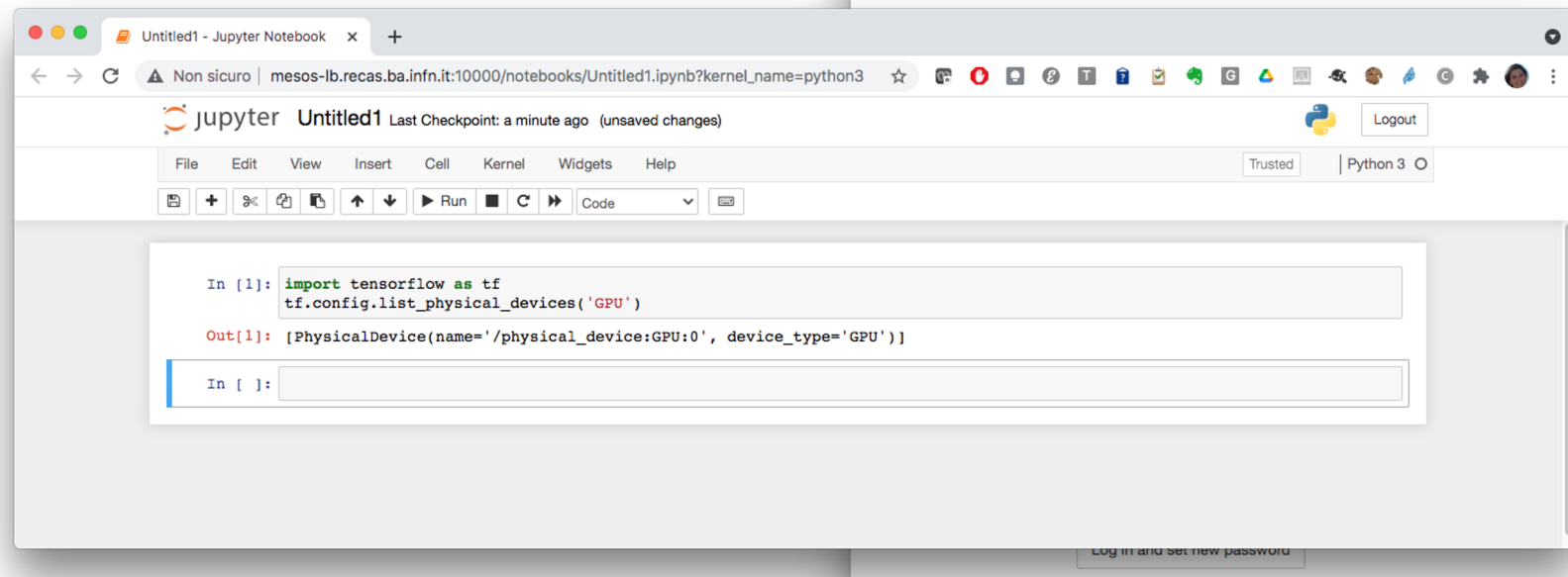
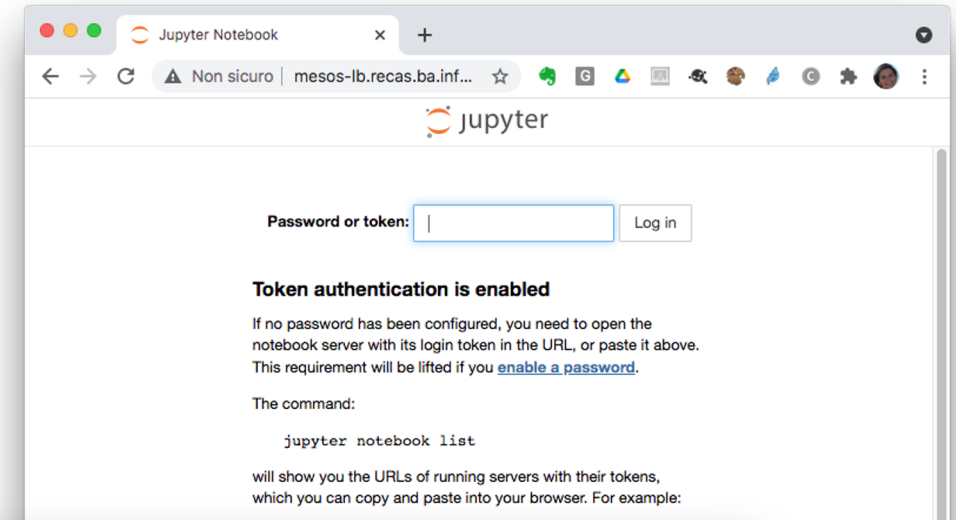
tensorflow:
  type: toscanodes.indigo.Container.Application.Docker.Marathon
  properties:
    environment_variables:
      TOKEN: { get_input: jupyter_password }
    uris: []
    command: "bash -c 'source /etc/bash.bashrc && jupyter notebook --notebook-dir=/tf --ip 0.0.0.0 --port $PORT0 --no-browser --allow-root --NotebookApp.token=$TOKEN'"
  artifacts:
    image:
      file: tensorflow/tensorflow:2.4.2-gpu-jupyter
      type: toscanodes.artifacts.Deployment.Image.Container.Docker
  requirements:
    - host: docker_runtime

docker_runtime:
  type: toscanodes.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](#)**