
Laniakea Documentation

Release 2.0.0

Marco Antonio Tangaro

May 16, 2022

1	Overview	3
2	Service architecture	5
3	ELIXIR-IIB: The Italian Infrastructure for Bioinformatics	7
4	INDIGO-DataCloud	9
4.1	The ELIXIR-IIB use case in INDIGO	11
4.2	References	11
5	Launch Galaxy	13
5.1	Galaxy express	13
5.2	Galaxy live build	13
5.3	Instantiate Galaxy	14
5.4	Galaxy access	17
6	Launch Galaxy Docker	19
6.1	Instantiate Galaxy	19
6.2	Galaxy access	21
6.3	References	24
7	Launch Galaxy cluster	25
7.1	Galaxy cluster	25
7.2	Galaxy elastic cluster	26
7.3	Instantiate Galaxy	26
7.4	Galaxy access	28
8	Manage an encrypted instance	31
8.1	Retrieve the encrypted storage passphrase	31
8.2	Restart Galaxy on an encrypted instance	32
8.3	Command line interface: luksctl	32
9	Create SSH Keys	33
9.1	Create your SSH key with Laniakea	33
9.2	Remove the SSH key from Laniakea	35
9.3	How to create SSH keys on Linux or macOS	35
9.4	How to create SSH keys on Windows	38

10 Virtual hardware presets	39
10.1 Laniakea@ReCaS	39
11 Galaxy Flavours	41
11.1 Galaxy minimal	41
11.2 Galaxy CoVaCS	41
11.3 Galaxy GDC Somatic Variant	41
11.4 Galaxy RNA workbench	42
11.5 Galaxy Epigen	42
12 Submit your flavour	43
12.1 Tool list configuration options	46
12.2 Conda support	46
12.3 References	49
13 Reference Data	51
13.1 data.galaxyproject.org	52
13.2 elixir-italy.covacs.refdata	52
13.3 elixir-italy.galaxy.refdata	52
13.4 Supplementary information	52
14 Galaxy production environment	59
14.1 OS support	59
14.2 PostgreSQL	59
14.3 NGINX	61
14.4 uWSGI	62
14.5 Proftpd	63
14.6 Supervisord	65
14.7 Paths	68
14.8 Enable Dockerized tools support in job_conf.xml	68
15 Galaxy Docker instance	69
15.1 Configuration files	69
15.2 CVMFS configuration	70
15.3 Galaxy docker usage	70
15.4 Galaxy Docker usage tutorial	71
16 Cluster configuration	73
16.1 job_conf.xml configuration	73
16.2 Shared file system	74
16.3 Network configuration	75
16.4 Worker nodes SSH access	75
16.5 Worker nodes deployment on elastic cluster	76
16.6 References	76
17 Authentication	79
17.1 Registration	79
17.2 Login	84
18 Frequently Asked Questions	87
18.1 How to manually recover Galaxy after VM reboot	87
18.2 I'm unable to create users from admin panel	87
19 The encryption layer	89
19.1 The encryption strategy	89

19.2	Storage encryption workflow	91
19.3	File System Encryption Test	91
19.4	Fast-luks script	93
19.5	Luksctl: LUKS volumes management	93
19.6	LUKSctl: APIs	95
19.7	Cryptsetup hints	98
19.8	References	99
20	Galaxyctl: Galaxy management	101
20.1	Galaxyctl basic usage	102
20.2	Logging	102
20.3	Advanced options	103
20.4	Configuration file	103
20.5	Features	104
21	Laniakea Ansible Roles	109
21.1	indigo-dc.galaxycloud	109
21.2	indigo-dc.galaxycloud-os	109
21.3	indigo-dc.galaxycloud-tools	110
21.4	indigo-dc.galaxycloud-refdata	110
21.5	indigo-dc.galaxycloud-fastconfig	110
21.6	indigo-dc.galaxycloud_docker	110
21.7	indigo-dc.cvmfs-client	111
21.8	indigo-dc.cvmfs-server	111
22	TOSCA templates	113
22.1	Custom types	114
22.2	Galaxy template	124
22.3	Galaxy cluster template	127
23	Build CVMFS server for reference data	131
23.1	Create CernVM-FS Repository	131
23.2	Client configuration	132
23.3	Populate a CernVM-FS Repository (with reference data)	132
23.4	Reference data download	133
23.5	References	134
24	Vault configuration	135
24.1	Vault main concepts	135
24.2	Vault authentication and authorization flow	135
24.3	Vault passphrase storage flow	137
24.4	Passphrase path on Vault	137
25	Laniakea Dashboard	139
25.1	Configuration	140
26	Laniakea installation	167
26.1	Services end-points	169
26.2	Service installation	170
27	GitHub repository	227
28	DockerHub repository	229
29	Support	231

30 Cite	233
31 Licence	235
32 Galaxy tutorials	237
33 Indices and tables	239



Laniakea provides the possibility to automate the creation of Galaxy-based virtualized environments through an easy setup procedure, providing an on-demand workspace ready to be used by life scientists and bioinformaticians.

Galaxy is a workflow manager adopted in many life science research environments in order to facilitate the interaction with bioinformatics tools and the handling of large quantities of biological data.

Once deployed each Galaxy instance will be fully customizable with tools and reference data and running in an insulated environment, thus providing a suitable platform for research, training and even clinical scenarios involving sensible data. Sensitive data requires the development and adoption of technologies and policies for data access, including e.g. a robust user authentication platform.

For more information on the Galaxy Project, please visit the <https://galaxyproject.org>

Laniakea has been developed by ELIXIR-IIB, the italian node of ELIXIR, within the INDIGO-DataCloud project (H2020-EINFRA-2014-2) which aims to develop PaaS based cloud solutions for e-science.

Note: Laniakea is in fast development. For this reason the code and the documentation may not always be in sync. We try to make our best to have good documentatation

Galaxy is a workflow manager adopted in many life science research environments in order to facilitate the interaction with bioinformatics tools and the handling of large quantities of biological data. Through a coherent work environment and an user-friendly web interface it organizes data, tools and workflows providing reproducibility, transparency and data sharing functionalities to users.

Currently, Galaxy instances can be deployed in three ways, each one with pros and cons: public servers, local servers and commercial cloud solutions. In particular, the demand for cloud solutions is rapidly growing (over 2400 Galaxy cloud servers launched in 2015), since they allow the creation of a ready-to-use galaxy production environment avoiding initial configuration issues, requiring less technical expertise and outsourcing the hardware needs. Nevertheless relying on commercial cloud providers is quite costly and can pose ethical and legal drawbacks in terms of data privacy.

ELIXIR-IIB in the framework of the INDIGO-DataCloud project is developing a cloud Galaxy instance provider, allowing to fully customize each virtual instance through a user-friendly web interface, overcoming the limitations of others galaxy deployment solutions. In particular, our goal is to develop a PaaS architecture to automate the creation of Galaxy-based virtualized environments exploiting the software catalogue provided by the INDIGO-DataCloud community (www.indigo-datacloud.eu/service-component).

Once deployed each Galaxy instance will be fully customizable with tools and reference data and running in an insulated environment, thus providing a suitable platform for research, training and even clinical scenarios involving sensible data. Sensitive data requires the development and adoption of technologies and policies for data access, including e.g. a robust user authentication platform.

The system allows to setup and launch a virtual machines configured with the Operative System (CentOS 7 or Ubuntu 14.04/16.04) and the auxiliary applications needed to support a Galaxy production environment such as PostgreSQL, Nginx, uWSGI and Proftpd and to deploy the Galaxy platform itself. It is possible to choose between different tools preset, or **flavors**: basic Galaxy or Galaxy configured with a selection of tools for NGS analyses already installed and configured (e.g. SAMtools, BamTools, Bowtie, MACS, RSEM, etc...) together with reference data for many organisms.

Service architecture

The web front-end is designed to grant user friendly access to the service, allowing to easily configure and launch each Galaxy instance through the indigo_fgw portal.

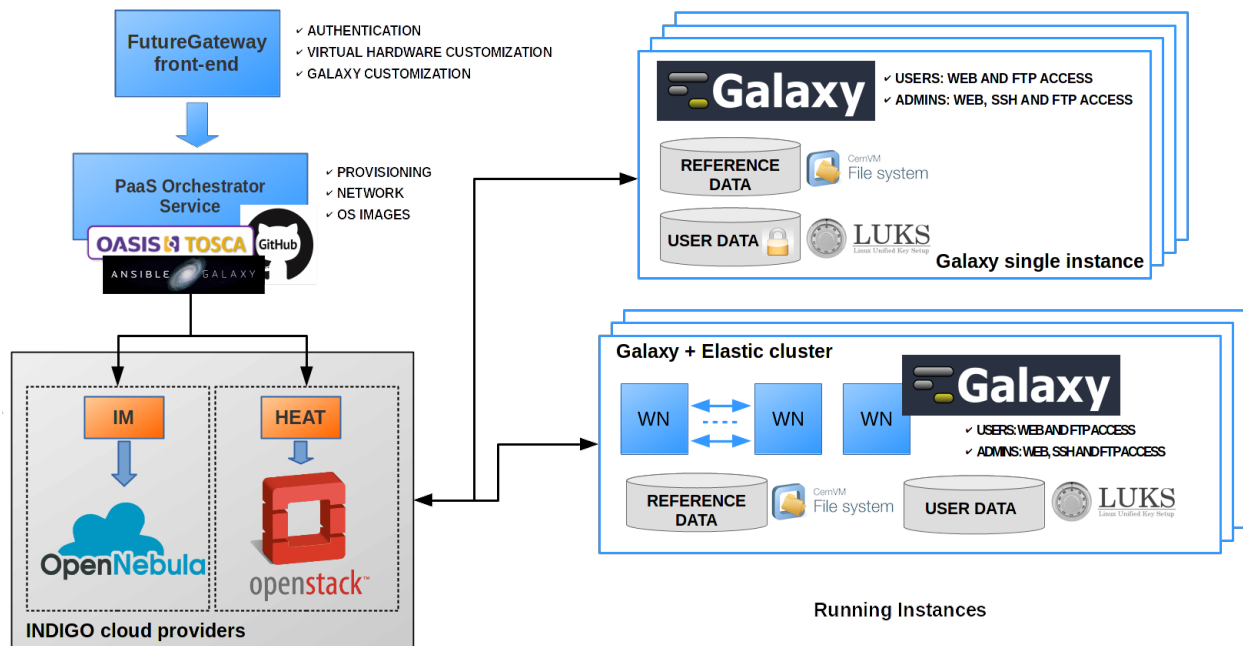


Fig. 1: Laniakea architecture

All the required components to automatically setup Galaxy instances (Galaxy and all its companion software) are deployed using the indigo_orchestrator and the indigo_im services, based on the TOSCA orchestration language. The service is compatible with both OpenNebula and OpenStack, its deployment on different e-infrastructures. Moreover, it supports both VMs and Docker containers, leaving the selection of the virtual environment to the service providers. This effectively removes the need to depend on particular configurations (e.g. OpenStack, OpenNebula or other private cloud solution like Amazon or Google).

Persistent storage is provided to store users and reference data and to install and run new (custom) tools and workflows. Data security and privacy are granted through the INDIGO indigo_onedata component which, at the same time, allow for transparent access to the storage resources through token management. Data encryption implemented at file system level protects user's data from any unauthorized access.

Automatic elasticity, provided using the indigo_clues service component, enables dynamic cluster resources scaling, deploying and powering-on new working nodes depending on the workload of the cluster and powering-off them when no longer needed. This provides an efficient use of the resources, making them available only when really needed.

ELIXIR-IIB: The Italian Infrastructure for Bioinformatics



ELIXIR-IIB (elixir-italy.org) is the Italian Node of ELIXIR (elixir-europe.org) and collects most of the leading Italian institutions in the field of bioinformatics, including a vast and heterogeneous community of scientists that use, develop and maintain a large set of bioinformatics services. It represents the Italian Node of ELIXIR, an European research infrastructure which goal is to integrate research data from all over Europe and ensure a seamless service provision easily accessible by the scientific community.

ELIXIR-IIB is also one of the scientific communities providing use cases to the INDIGO-Datacloud project (H2020-EINFRA-2014-2) which aims to develop PaaS based cloud solutions for e-science.

For a complete overview of ELIXIR-IIB related projects and services, please visit: <http://elixir-italy.org/en/>



INDIGO - DataCloud

The **INDIGO-DataCloud** project (H2020-EINFRA-2014-2) aims to develop an open source computing and data platform, targeted at multi-disciplinary scientific communities, provisioned over public and private e-infrastructures.

In order to exploit the full capabilities of current cloud infrastructures, supporting complex workflows, data transfer and analysis scenarios, the INDIGO architecture is based on the analysis and the realization of use cases selected by different research communities in the areas of High Energy Physics, Bioinformatics, Astrophysics, Environmental modelling, Social sciences and others.

INDIGO released two software release:

Release	Code name	URL
First release	MIDNIGHT-BLUE	https://www.indigo-datacloud.eu/news/first-indigo-datacloud-software-release-out
Second re-release	ELEC-TRICINDIGO	https://www.indigo-datacloud.eu/news/electricindigo-second-indigo-datacloud-software-release

The INDIGO-DataCloud releases provide open source components for:

1. **IaaS layer:** increase the efficiency of existing Cloud infrastructures based on OpenStack or OpenNebula through advanced scheduling, flexible cloud/batch management, network orchestration and interfacing of high-level Cloud services to existing storage systems.

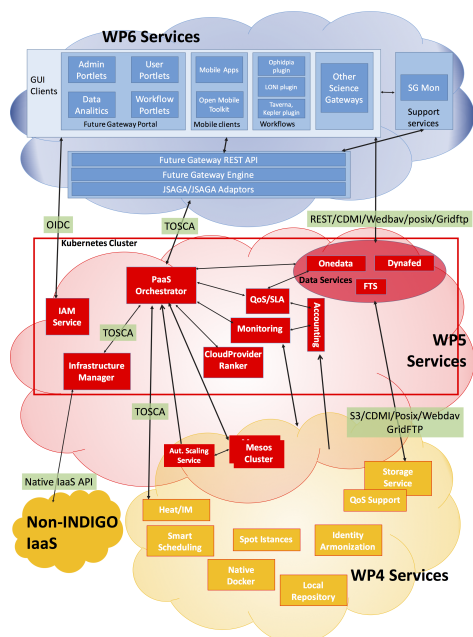


Fig. 1: The INDIGO-DataCloud architecture

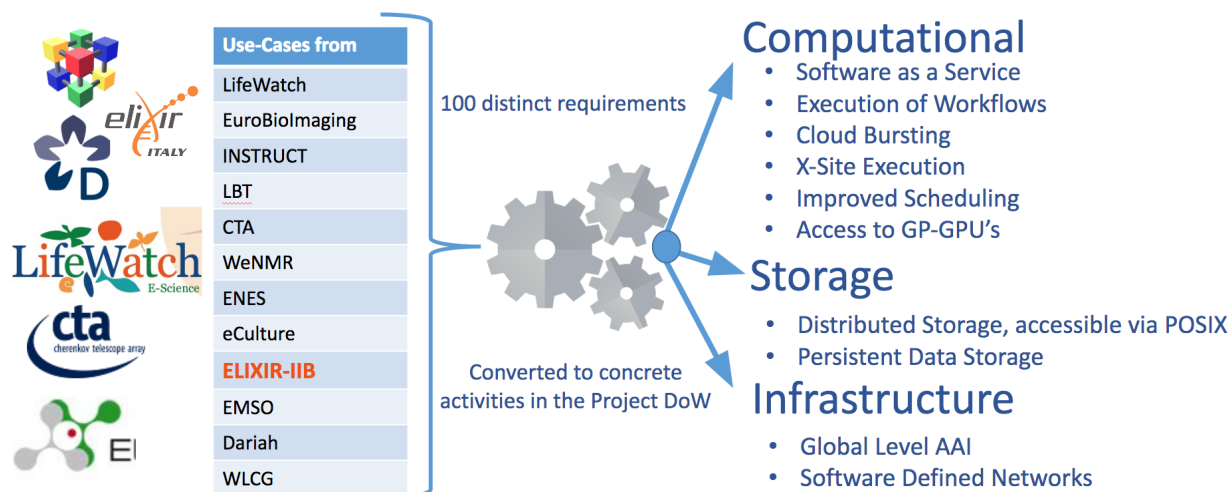


Fig. 2: The INDIGO-DataCloud communities

2. **PaaS layer:** easily port applications to public and private Clouds using open programmable interfaces, user-level containers, and standards-based languages to automate definition, composition and embodiment of complex set-ups.
3. **Identity and Access Management:** manage access and policies to distributed resources.
4. **FutureGateway:** a programmable scientific portal providing easy access to both the advanced PaaS features provided by the project and to already existing applications.
5. **Data Management and Data Analytics Solutions:** distribute and access data through multiple providers via virtual file systems and automated replication and caching.

For a complete list of INDIGO-DataCloud services, please visit: <https://www.indigo-datacloud.eu/service-component>

4.1 The ELIXIR-IIB use case in INDIGO

ELIXIR-IIB in the framework of the INDIGO-DataCloud project is developing a cloud Galaxy instance provider, allowing to fully customize each virtual instance through a user-friendly web interface, overcoming the limitations of others galaxy deployment solutions. In particular, our goal is to develop a PaaS architecture to automate the creation of Galaxy-based virtualized environments exploiting the software catalogue provided by the INDIGO-DataCloud community.

1. All Galaxy required components automatically deployed (**INDIGO PaaS Orchestrator** and the **Infrastructure Manager**):
 - Galaxy
 - PostgreSQL
 - NGINX
 - uWSGI
 - Proftpd
 - Galaxy tools (from ToolShed)
 - Reference Data
2. User friendly access, allowing to easily configure and launch a Galaxy instance (**INDIGO FutureGateway portal**)
3. Authentication (**Identity and Access Management** and **FutureGateway**)
4. Persistent storage, data security and privacy (**Onedata** or IaaS block storage with filesystem encryption).
5. Cluster support with automatic elasticity (**INDIGO CLUES**).

4.2 References

INDIGO services

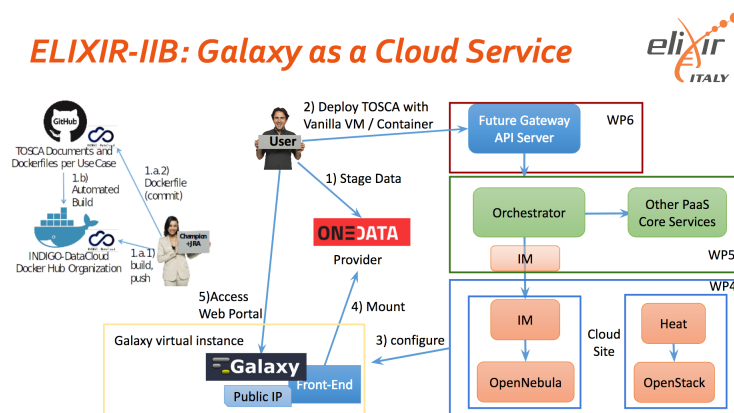


Fig. 3: ELIXIR-IIB use case in INDIGO architecture for single Galaxy instances deployment.

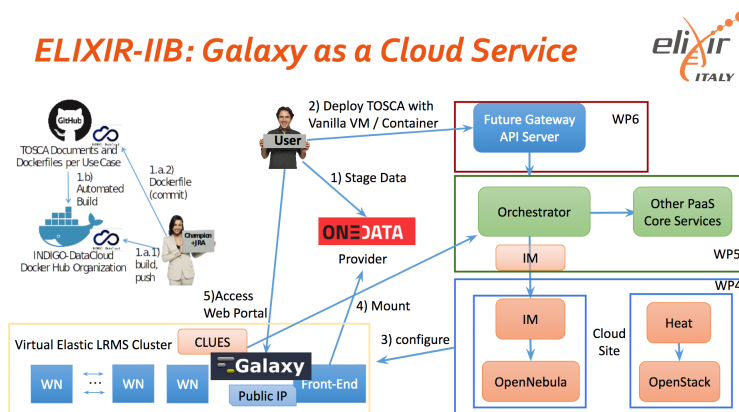


Fig. 4: ELIXIR-IIB use case in INDIGO architecture for Galaxy with cluster support deployment

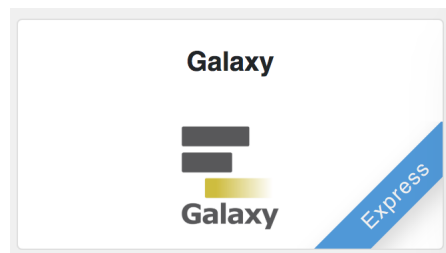
Launch Galaxy

The Laniakea dashboard tiles allow user to deploy a standard [Galaxy production environment](#) through two methods: Galaxy express and Galaxy live build.

See also:

To login to the Laniakea dashboard visit the section: [Authentication](#).

5.1 Galaxy express

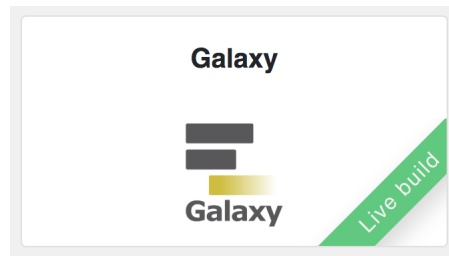


The Galaxy express instantiate a CentOS 7 Virtual Machine with Galaxy, all its companion software and the set of tools that come with the selected flavour. Once deployed each Galaxy instance can be further customized with additional tools and reference data.

This version is usually quite reliable and work well for most users.

5.2 Galaxy live build

The Galaxy live build allows to setup and launch a virtual machine configured with the Operative System CentOS 7 and the auxiliary applications needed to support a Galaxy production environment such as PostgreSQL, Nginx, uWSGI and Proftpd and to deploy the Galaxy platform itself and the tools that come with the selected flavour.



This version is recommended for those users which want to be sure to have the latest available version of each tool.

Warning: In fact, each tool is downloaded from the repositories and configured on the fly. Depending on the number of the tools to be installed the deployment process may take time a variable amount of time.

5.3 Instantiate Galaxy

Enter the Galaxy express or Galaxy live build configuration section. The configuration options are the same.

Provide a description for your instance using the `Instance description` field, which will identify your Galaxy in the **Deployments page**, once your request is submitted.

Two panels allows to configure the virtual hardware and the Galaxy instance respectively.

5.3.1 Virtual hardware configuration



1. Select your instance flavour (virtual CPUs and the memory size). More information on available virtual hardware presets can be found here: [Virtual hardware presets](#).
2. Copy & Paste your SSH key, to login in the Galaxy instance or configure it in the [Create SSH Keys](#) page.
3. Laniakea provides the possibility to encrypt the storage volume associated with the virtual machine on-demand, to protect user data.

To enable storage encryption set the switch to **ON**.

Warning: Only the external volume where Galaxy data are stored is encrypted, not the Virtual Machine root disk.

The storage will be encrypted with a strong alphanumerical passphrase. More information on this topic can be found here:

- [Manage an encrypted instance](#)
 - [The encryption layer](#)
4. Finally, it is possible to select the user storage volume size.

 Laniakea Dashboard [Deployments](#) [Advanced](#) [Users](#) [Documentation](#)  Marco Tangaro

Galaxy

Description: Deploy Galaxy on a single Virtual Machine from a VM image (FAST). The basic configuration includes CentOS 7, the selected Galaxy flavour, companion software and reference data. Configure, click on the "Submit" button, wait for the confirmation e-mail(s) and log in to your new Galaxy instance. If after some hours you do not receive any e-mail please be sure to check your SPAM BOX.

Instance description

Virtual hardware [Galaxy](#) [Advanced](#)

Instance flavour

Medium (2 cpu, 4 GB RAM, 20 GB disk)

CPUs, memory size (RAM), root disk size

Galaxy instance SSH public key

Paste here your SSH public key or configure a default key

Enable encryption



☐ Off

Encrypt instance external storage

Storage volume size

50 GB

Select storage size

 **Laniakea Dashboard** [Deployments](#) [Advanced ▾](#) [Users](#) [Documentation](#)  [Marco Tangaro ▾](#)

Galaxy

Description: Deploy Galaxy on a single Virtual Machine from a VM image (FAST). The basic configuration includes CentOS 7, the selected Galaxy flavour, companion software and reference data. Configure, click on the "Submit" button, wait for the confirmation e-mail(s) and log in to your new Galaxy instance. If after some hours you do not receive any e-mail please be sure to check your SPAM BOX.

Instance description

[Virtual hardware](#) [Galaxy](#) [Advanced](#)

Galaxy version

▾

Galaxy release 18.05 recommended

Instance description

Set Galaxy Brand

Galaxy administrator e-mail

Type a valid e-mail address.

Galaxy flavours

▾

Load Galaxy tools preset

Reference data repository

▾

Select reference data repository

5.3.2 Galaxy configuration

1. Select the Galaxy version, the instance administrator e-mail and the Galaxy brand tag (the top-left name in the Galaxy home page).
2. Provide a valid e-mail address as Galaxy administrator credential.

Note: A notification mail will be sent to this e-mail address once the deployment is done.

3. Select the Galaxy flavour among those available (see section [Galaxy Flavours](#)).
4. Select Galaxy reference dataset. The default should be the best choice for most users (see section [Reference Data](#)).
5. Finally, `SUBMIT` your request.

5.4 Galaxy access

Once your Galaxy instance is ready, a confirmation e-mail is sent to the Laniakea user and to the galaxy administrator email, if different, with the Galaxy URL and user credentials.

Warning: If you don't receive the e-mail:

1. Check you SPAM mail directory
2. Chek mail address spelling
3. Wait 15 minutes more.


The instance information are also available in the **Deployments** page of the dashboard:

The galaxy administrator password and the API key are automatically set during the instatiation procedure and are the same for each instance:

```
User: administrator e-mail
Password: galaxy_admin_password
API key: ADMIN_API_KEY
```

Warning: Change the Galaxy password and API key as soon as possible!

Warning: The anonymous login is disabled by default.


 Laniakea Dashboard

Deployments


Advanced ▾

Users

Documentation

 Marco Tangaro ▾




My deployments

 Refresh

[+ New deployment](#)

Show entries

Search:

Instance name	Status	Creation time	Galaxy flavour	VM flavour	Endpoint	Actions
galaxy docker test	CREATE_COMPLETE	2019-10-07 10:10:00	bgruening/galaxy-stable:19.01	medium	http://90.147.75.27	 Delete ▾
galaxy express test	CREATE_COMPLETE	2019-10-07 09:21:00	galaxy-minimal	medium	http://90.147.75.219/galaxy	 Delete ▾
elastic cluster test	UPDATE_COMPLETE	2019-10-04 18:32:00	galaxy-minimal	cluster	http://90.147.102.53/galaxy	 Delete ▾

Showing 1 to 3 of 3 entries

Previous

1

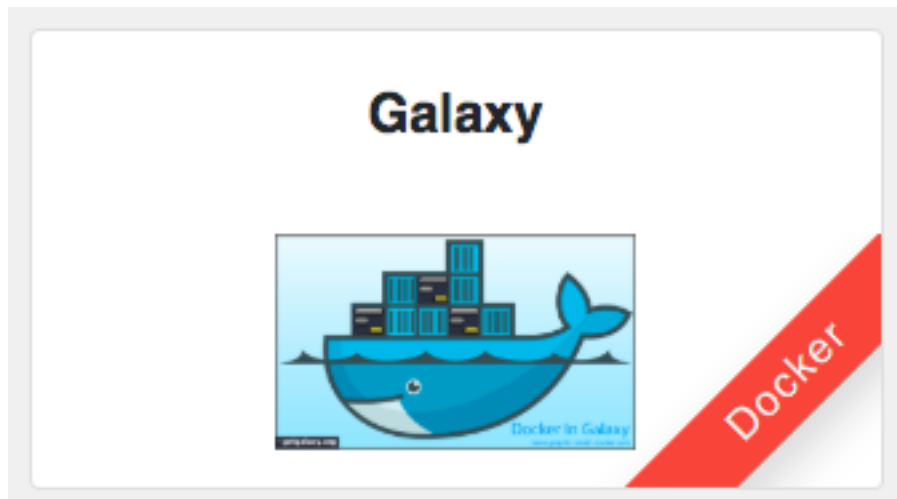
Next

Launch Galaxy Docker

The Laniakea dashboard tiles allow user to deploy Galaxy through its official [Docker image](#).

See also:


To login to the Laniakea dashboard visit the section: [Authentication](#).



The Galaxy Docker instantiate an Ubuntu 16.04 Virtual Machine with the Galaxy official Docker. Once deployed each Galaxy instance can be further customized with additional tools and reference data.

6.1 Instantiate Galaxy

Enter the Galaxy Docker configuration section.


 Laniakea Dashboard

Deployments

Advanced ▾

Users

Documentation

 Marco Tangaro ▾

Galaxy

Description: Deploy Galaxy docker image on a single Virtual Machine.

Instance description

Virtual hardware [Galaxy](#) [Advanced](#)

Instance flavour

Medium (2 cpu, 4 GB RAM, 20 GB disk) ▾

CPU, memory size (RAM), root disk size

Galaxy instance SSH public key

Paste here your SSH public key or configure a default key

Enable encryption

☐ Off

Encrypt instance external storage

Storage volume size

100 GB ▾

Select storage size

Provide a description for your instance using the `Instance description` field, which will identify your Galaxy in the **Deployments** page, once your request is submitted.

Two panels allows to configure the virtual hardware and the Galaxy instance respectively.

6.1.1 Virtual hardware configuration

1. Select your instance flavour (virtual CPUs and the memory size). More information on available virtual hardware presets can be found here: [Virtual hardware presets](#).
2. Copy & Paste your SSH key, to login in the Galaxy instance or configure it in the [Create SSH Keys](#) page.
3. Laniakea provides the possibility to encrypt the storage volume associated with the virtual machine on-demand, to protect user data.

To enable storage encryption set the switch to **ON**.

Warning: Only the external volume where Galaxy data are stored is encrypted, not the Virtual Machine root disk.

The storage will be encrypted with a strong alphanumerical passphrase. More information on this topic can be found here:

- [Manage an encrypted instance](#)
- [The encryption layer](#)

4. Finally, it is possible to select the user storage volume size.

6.1.2 Galaxy configuration

1. Select the instance administrator e-mail and the Galaxy brand tag (the top-left name in the Galaxy home page).
2. Provide a valid e-mail address as Galaxy administrator credential.

Note: A notification mail will be sent to this e-mail address once the deployment is done.



3. Select the Galaxy flavour among those available.
4. Select Galaxy reference dataset. The default should be the best choice for most users (see section [Reference Data](#)).
5. Finally, **SUBMIT** your request.

6.2 Galaxy access

Once your Galaxy instance is ready, a confirmation e-mail is sent to the Laniakea user and to the galaxy administrator email, if different, with the Galaxy URL and user credentials.

Warning: If you don't receive the e-mail:

1. Check you SPAM mail directory


Laniakea Dashboard
[Deployments](#)
[Advanced ▾](#)
[Users](#)
[Documentation](#)

 Marco Tangaro ▾

Galaxy

Description: Deploy Galaxy docker image on a single Virtual Machine.

Instance description

[Virtual hardware](#)
[Galaxy](#)
[Advanced](#)

Instance description

Set Galaxy Brand

Galaxy administrator e-mail

Type a valid e-mail address.

Galaxy flavours

Load Galaxy tools preset

Reference data repository

Select reference data repository

2. Chek mail address spelling
3. Wait 15 minutes more.

The instance information are also available in the **Deployments** page of the dashboard:

My deployments

Refresh + New deployment

Show 10 entries Search:

Instance name	Status	Creation time	Galaxy flavour	VM flavour	Endpoint	Actions
galaxy docker test	CREATE_COMPLETE	2019-10-07 10:10:00	bgruening/galaxy-stable:19.01	medium	http://90.147.75.27	Delete
galaxy express test	CREATE_COMPLETE	2019-10-07 09:21:00	galaxy-minimal	medium	http://90.147.75.219/galaxy	Delete
elastic cluster test	UPDATE_COMPLETE	2019-10-04 18:32:00	galaxy-minimal	cluster	http://90.147.102.53/galaxy	Delete

Showing 1 to 3 of 3 entries Previous 1 Next

The galaxy administrator password and the API key are automatically set during the instantiation procedure and are the same for each instance:

```
User: administrator e-mail
Password: galaxy_admin_password
API key: ADMIN_API_KEY
```

Warning: Change the Galaxy password and API key as soon as possible!

Warning: The anonymous login is disabled by default.

6.3 References

Official Galaxy Docker slides

Launch Galaxy cluster

Galaxy serves tools which may require a wide range of computing resources to properly work. To account this, the Laniakea dashboard tiles allow user to deploy a standard [Galaxy production environment](#) connected to a [compute cluster](#).

See also:

To login to the Laniakea dashboard visit the section: [Authentication](#).

7.1 Galaxy cluster

The Galaxy cluster instantiate a Galaxy server and the worker nodes.



7.1.1 Galaxy cluster Express

The Galaxy cluster Express instantiate a CentOS 7 Virtual Machine with Galaxy, all its companion software and the set of tools that come with the selected flavour. Once deployed each Galaxy instance can be further customized with additional tools and reference data.

This version is usually quite reliable and work well for most users.

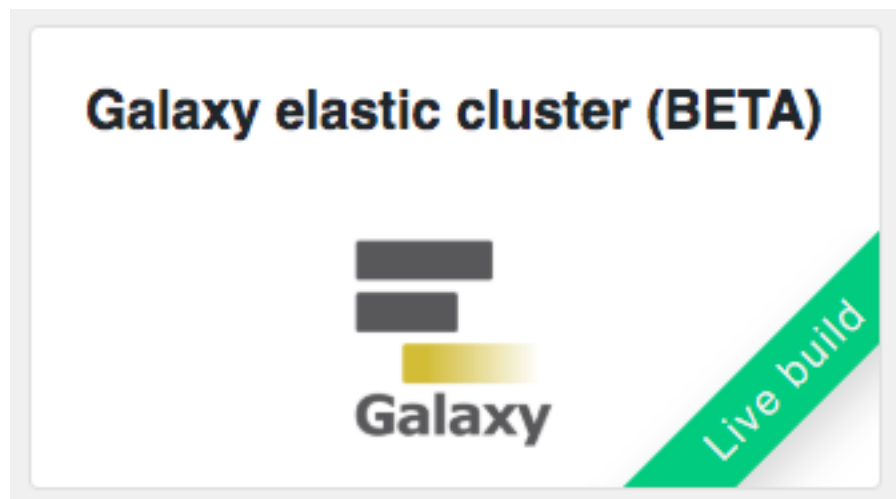
7.1.2 Galaxy cluster Live Build

The Galaxy cluster Live Build allows to setup and launch a virtual machine configured with the Operative System CentOS 7 and the auxiliary applications needed to support a Galaxy production environment such as PostgreSQL, Nginx, uWSGI and Proftpd and to deploy the Galaxy platform itself and the tools that come with the selected flavour.

This version is recommended for those users which want to be sure to have the latest available version of each tool.

7.2 Galaxy elastic cluster

The Galaxy elastic cluster section allows to deploy a Galaxy Server with automatic elasticity support for worker nodes deployment. Automatic elasticity enables dynamic cluster resources scaling, deploying and powering on new working nodes depending on the workload of the cluster and powering-off them when no longer needed. This provides an efficient use of the resources, making them available only when really needed.



Warning: Currently, this feature is under beta testing. Galaxy and tools are installed on-the-fly starting from a bare CentOS 7 image. The whole process, i.e. install Galaxy and tools, may take time. We will soon add the possibility to exploit images with tools to speed-up the configuration

Warning: Each node takes 12 minutes or more to be instantiated. Therefore, the job needs the same time to start. On the contrary, if the node is already deployed, the job will start immediately.

7.3 Instantiate Galaxy

Enter the Galaxy cluster (Express or Live BUild) or Galaxy elastic cluster configuration section. The configuration options are the same.

Provide a description for your instance using the `Instance description` field, which will identify your Galaxy in the **Deployments page**, once your request is submitted.

Two panels allows to configure the virtual hardware and the Galaxy instance respectively.

Galaxy cluster

Description: Deploy Galaxy from a VM image with cluster support (FAST). The basic configuration includes CentOS 7, SLURM, the selected Galaxy flavour, companion software and reference data. Configure, click on the "Submit" button, wait for the confirmation e-mail(s) and log in to your new Galaxy instance. If after some hours you do not receive any e-mail please be sure to check your SPAM BOX.

Instance description

Virtual hardware [Galaxy](#) [Advanced](#)

Instance flavour

CPUs, memory size (RAM), root disk size

Worker nodes number

Number of worker nodes in the cluster

Worker nodes flavour

CPUs, memory size (RAM), root disk size

Galaxy instance SSH public key

Paste here your SSH public key or configure a default key

Enable encryption

☐ Off

Encrypt instance external storage

Storage volume size

Select storage size

1. Select the instance flavour (virtual CPUs and the memory size) for your Front node, i.e. the Galaxy server. More information on available virtual hardware presets can be found here: [Virtual hardware presets](#).
2. Select the number of Virtual Worker Nodes of your Cluster and the instance flavor, (virtual CPUs and RAM) for each worker node. More information on available virtual hardware presets can be found here: [Virtual hardware presets](#).
3. Copy & Paste your SSH key, to login in the Galaxy instance or configure it in the [Create SSH Keys](#) page.
4. Laniakea provides the possibility to encrypt the storage volume associated with the virtual machine on-demand, to protect user data.

To enable storage encryption set the switch to **ON** .

Warning: Only the external volume where Galaxy data are stored is encrypted, not the Virtual Machine root disk.

The storage will be encrypted with a strong alphanumerical passphrase. More information on this topic can be found here:

- [Manage an encrypted instance](#)
- [The encryption layer](#)

5. Finally, it is possible to select the user storage volume size.
 1. Select the Galaxy version, the instance administrator e-mail and the Galaxy brand tag (the top-left name in the Galaxy home page).
 2. Provide a valid e-mail address as Galaxy administrator credential.

Note: A notification mail will be sent to this e-mail address once the deployment is done.

3. Select the Galaxy flavour among those available (see section [Galaxy Flavours](#)).
4. Select Galaxy reference dataset. The default should be the best choice for most users (see section [Reference Data](#)).
5. Finally, **SUBMIT** your request.



7.4 Galaxy access

Once your Galaxy instance is ready, a confirmation e-mail is sent to the Laniakea user and to the galaxy administrator email, if different, with the Galaxy URL and user credentials.

Warning: If you don't receive the e-mail:

1. Check you SPAM mail directory
2. Chek mail address spelling
3. Wait 15 minutes more.

The instance information are also available in the **Deployments** page of the dashboard:

 Laniakea Dashboard [Deployments](#) [Advanced](#) [Users](#) [Documentation](#)  Marco Tangaro

Galaxy cluster

Description: Deploy Galaxy from a VM image with cluster support (FAST). The basic configuration includes CentOS 7, SLURM, the selected Galaxy flavour, companion software and reference data. Configure, click on the "Submit" button, wait for the confirmation e-mail(s) and log in to your new Galaxy instance. If after some hours you do not receive any e-mail please be sure to check your SPAM BOX.

Instance description

[Virtual hardware](#) [Galaxy](#) [Advanced](#)

Galaxy version

Galaxy release 18.05 recommended

Instance description

Set Galaxy Brand

Galaxy administrator e-mail


Type a valid e-mail address.

Galaxy flavours

Load Galaxy tools preset

Reference data repository

Select reference data repository


Laniakea Dashboard
[Deployments](#)
[Advanced](#)
[Users](#)
[Documentation](#)
Marco Tangaro

My deployments

Refresh
+ New deployment

Show 10 entries
Search:

Instance name	Status	Creation time	Galaxy flavour	VM flavour	Endpoint	Actions
galaxy docker test	CREATE_COMPLETE	2019-10-07 10:10:00	bgruening/galaxy-stable:19.01	medium	http://90.147.75.27	Delete
galaxy express test	CREATE_COMPLETE	2019-10-07 09:21:00	galaxy-minimal	medium	http://90.147.75.219/galaxy	Delete
elastic cluster test	UPDATE_COMPLETE	2019-10-04 18:32:00	galaxy-minimal	cluster	http://90.147.102.53/galaxy	Delete

Showing 1 to 3 of 3 entries

Previous
1
Next

The galaxy administrator password and the API key are automatically set during the instantiation procedure and are the same for each instance:

```
User: administrator e-mail
Password: galaxy_admin_password
API key: ADMIN_API_KEY
```

Warning: Change the Galaxy password and API key as soon as possible!

Warning: The anonymous login is disabled by default.

Manage an encrypted instance

Laniakea provides the possibility to encrypt the storage volume associated to the virtual machine on-demand.

A detailed description of Laniakea encryption strategy is reported here: [The encryption layer](#).

Warning: Only the external volume, where Galaxy data are stored, is encrypted, not the Virtual Machine root disk. The encryption layer should be secure enough to protect data uploaded from users to the Galaxy instance from any unwanted attention. However, users must be aware that the responsibility of correctly handling any sensitive data they upload to Laniakea falls on them and that the administrators of the Laniakea service can not be considered responsible for any data breach that may happen due to negligence by Galaxy users or the action of external malicious attackers.

8.1 Retrieve the encrypted storage passphrase

Cryptographic keys should never be transmitted in the clear. For this reason Laniakea encrypt your storage with a strong alphanumerical random passphrase.

This passphrase can be easily retrieved thorough the dashboard.

Warning: If you require the storage encryption, please retrieve your passphrase as soon as possible and keep it secret.

1. Connect to the dashboard and click on the name of your encrypted instance.
2. In the overview tab, click on `Retrieve LUKS passphrase` button.
3. Copy your passptase.

8.2 Restart Galaxy on an encrypted instance

In case of reboot of your virtual instance, the encrypted storage cannot be automatically enabled again, since the encryption passphrase is needed. The user intervention is needed.

It is possible to do this through the dashboard.

1. Connect to the dashboard and click on the name of your encrypted instance.
2. In the overview tab, the button `Unlock` and `mount volume` is available only if the encrypted storage is not mounted. Click it to unlock
3. It is now possible to restart Galaxy. The button `Try to restart Galaxy` will be enabled only if the encrypted storage is correctly mounted, avoiding to start Galaxy without user data.

Note: If the automatic procedure does not work, please have a look here: [Frequently Asked Questions](#)

8.3 Command line interface: luksctl

To easily the encrypted storage management a python script, `luksctl`, is installed.

By default its configuration file is stored in `/etc/luks/luks-cryptdev.ini`.

Warning: Please don't change it unless you know what you're doing.

Note: The script requires superuser rights.

Here the list of the currently available commands:

Action	Command	Description
Open	<code>sudo luksctl open</code>	Open the encrypted device, requiring your passphrase.
Close	<code>sudo luksctl close</code>	Close and umount the encrypted device
Status	<code>sudo luksctl status</code>	Check device status

Create SSH Keys

SSH keys allow you to establish a secure connection between your computer and Galaxy.

Generating a key pair provides you with two long string of characters: a public and a private key. Laniakea upload the public key on the Galaxy server and then unlock it by connecting to it with a client that already has the private key. When the two match up, the system unlocks without the need for a password. You can increase security even more by protecting the private key with a passphrase.

Warning: Laniakea requires **ONLY** a SSH public key to instatiate Galaxy and grant you the access on the Virtual Machine.

9.1 Create your SSH key with Laniakea

During the Galaxy instance configuration procedure a SSH public key has to be mandatorly provided. This field, in fact, is required and without the SSH key you won't be able to submit your deployment.

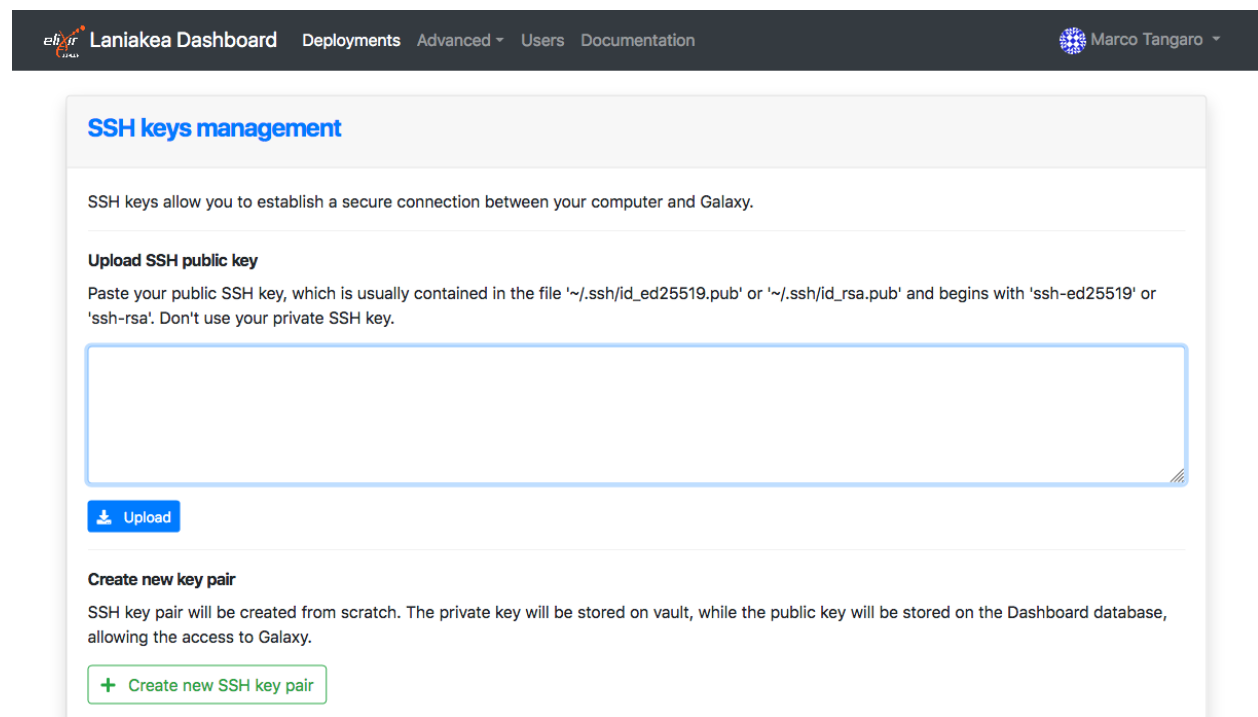
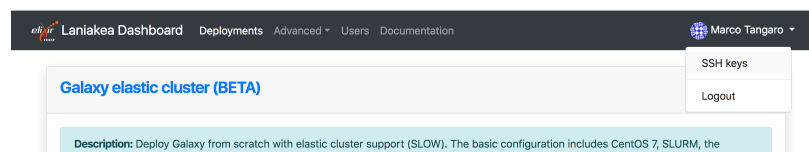
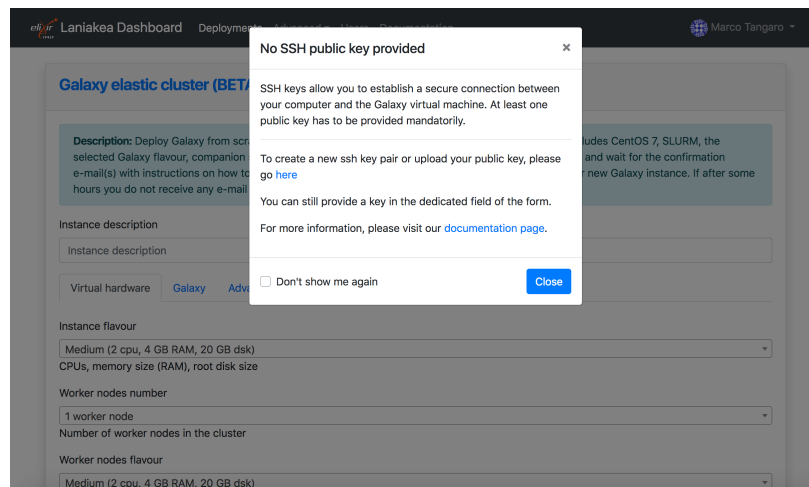
Warning: FOR SECURITY REASONS THE SSH KEY OF A VIRTUAL INSTANCE CANNOT BE CHANGED FROM THE LANIAKEA DASHBOARD AFTER ITS DEPLOYMENT. IF NEEDED, AND IF YOU KNOW WHAT YOU ARE DOING, IT CAN STILL BE MODIFIED ACCESSING DIRECTLY THE INSTANCE VIA SSH.

NOTICE THAT IF YOU LOSE THE PRIVATE KEY CORRESPONDING TO THE PUBLIC ONE ON THE VM HOSTING YOUR GALAXY INSTANCE, IT WILL BECOME UNACCESSIBLE FOREVER.

An example of using interpreted text

For this reason the Laniakea dashboard provides a menu to upload/create the user public (and private) key, in the top left user menu.

This will load the SSH management page, which will allow you to upload a SSH public key or generate a SSH key pair.



We recommend you to manually generate your SSH key pair and then upload the SSH public key on Laniakea. Paste your public Key in the text box

SSH keys management

SSH keys allow you to establish a secure connection between your computer and Galaxy.

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.

```
AAAAB3NzaC1yc2EAAAADAQABAAQDy787GZIVdHW7QV+Wu2q9q5k5CiTOq04ENioVig88IIvGNqi8qiX+3fhZx/w2hhlz6AePrYu8CfVPpICRd
SMjP46av53V1M7r0+yqJvuk1PC2f
/rSoEL95TvaelV28+5WY4MC58UvYuewuhIHcbfPiXhf3NEE3scd38GXCYKLhAP28mUQ950Ar4SoWv4irv21maJwkwn5AYXcy1yrbBZtaTbQELV
Pa/E6X9j+k29bn32ITmmtKBA3ne/QIFRaaYI3XggvMXhhSSiYsJUdISOjUTriB2DraHsxMGfOPjmPXkivrXp9MfOzjMg10fb7K2Mda8u
/ujK/dvx3BnhISlpn marco@marco-Latitude-3440
```

Upload

Create new key pair

SSH key pair will be created from scratch. The private key will be stored on vault, while the public key will be stored on the Dashboard database, allowing the access to Galaxy.

+ Create new SSH key pair

and press the upload button.

If you don't have a public key, it is possible to create a SSH key pair, i.e. a public and a private key.

Warning: The private key is not exploited by Laniakea. Is only generated and uploaded on Vault for security. Please download it. The Laniakea team will not be held liable for lost data due to hardware failure, virus, spyware, corruption or any other situation.

And then retrieve it with the `Retrieve SSH private key` button.

Once the public SSH key is available on the Dashboard the service will recognize it and it no longer needs to be loaded.


9.2 Remove the SSH key from Laniakea


It is possible to delete the SSH key (pair) from Laniakea with `Delete` button.

Warning: The key will not be removed from the virtual instances where it has been inserted. Once removed, if not saved elsewhere, and if no different keys were added, you will not be able to access the instances.

9.3 How to create SSH keys on Linux or macOS

<https://www.digitalocean.com/docs/droplets/how-to/add-ssh-keys/create-with-openssh/>


Laniakea Dashboard
Deployments
Advanced
Users
Documentation


Marco Tangaro


SSH keys management

SSH keys allow you to establish a secure connection between your computer and Galaxy.

Your SSH key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDy7B7GZIVdHW7QV+Wu2q9q5k5CtOq04ENioVig88IIlVGNqi8qiX+3fhZx/w2hhlz6AePrYu8CfVPpICRd
SMjP46av53V1M7r0+yqJvuk1PC2f
/rSoEL95TvaeiV28+5Wy4MC58UvYuewuhIHcbfPiXhF3NEE3scd38GXCYKlHAP28mUQ950Ar4SoWv4irv21maJwkwn5AYXcy1yrbBZtaTbQELV
Pa/E6X9j+k29bn32ITmmtKBA3ne/QlFRaaYI3XggvMXhhSSiYsJUdlSOjUTriB2DraHsxMGfOPjmPXkvrXp9MfOzjMg10fb7K2Mda8u
```


Delete
Retrieve SSH private key




Create new key pair

SSH key pair will be created from scratch. The private key will be stored on vault, while the public key will be stored on the Dashboard database, allowing the access to Galaxy.

+ Create new SSH key pair


Laniakea Dashboard
Deployments
Advanced
Users
Documentation


Marco Tangaro

SSH keys management

SSH keys allow you to establish a secure connection between your computer and Galaxy.

Your SSH key:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAQDy7B7GZIVdHW7QV+Wu2q9q5k5CtOq04ENioVig88IIlVGNqi8qiX+3fhZx/w2hhlz6AePrYu8CfVPpICRd
SMjP46av53V1M7r0+yqJvuk1PC2f
/rSoEL95TvaeiV28+5Wy4MC58UvYuewuhIHcbfPiXhF3NEE3scd38GXCYKlHAP28mUQ950Ar4SoWv4irv21maJwkwn5AYXcy1yrbBZtaTbQELV
Pa/E6X9j+k29bn32ITmmtKBA3ne/QlFRaaYI3XggvMXhhSSiYsJUdlSOjUTriB2DraHsxMGfOPjmPXkvrXp9MfOzjMg10fb7K2Mda8u
```

Delete
Retrieve SSH private key



SSH Private Key

```
-----BEGIN PRIVATE KEY-----
MIIEvwiBADANBgkqhkiG9w0BAQEFAASCBAkwggSIAgEAAoI
BAQDPiI20kmqOCqhV
x2TMtMTfH70B7cvtUvEv/fibsuOS27h86Hi4KdaKAfxWawpN
eOKksPUwHu2c1TIQ
Z7PC9ebXiAzjbXsGMik6th29peYQTgfN1Bz8UZUmuceD7M
mjYn6oPVk6C2E4LXp8
7ldhHD8htK+uQloWjLi1NgQlInf+BGXlxiH0tg2rjve18uT7qoF1
pSTDGMbSsuT2
HBeWHRjukZ8nHU7U+eYwLhAWyG2sOqcyz+qfUj/t2lFyTnsG
F1X/ETgEd2hwm3Bv
NOWVcEhfKeZKi2BTsYM2+jGCVgFO0XST6xBAEUPZVySLA
pOT1bBiv47vdBvCQzhI
YRPkBVXBAGMBAECggEAPWW1qwHVvAezHQ1L6LNhupB
thfD3b1lmQAhmT8lwQbX2
sh0j+XeHHHTR7c7k0V59zjL7lizQJqF9vyeuqnKMxB5fkbIFuK
OGRvQxgDVgBlrP
PDkQCiUodrhnKcQXMTqklajwiJdtP21U2zguJLByjMojXu4KM
LmYuyMrvsCftumV
sQRHgjajWjbVpJF4oGgfEEUrzUw1kG/fgmLNobYQr9hWqh
```

Close
Copy to clipboard

36

Chapter 9. Create SSH Keys

 Laniakea Dashboard [Deployments](#) [Advanced](#) [Users](#) [Documentation](#)  Marco Tangaro

Galaxy elastic cluster (BETA)

Description: Deploy Galaxy from scratch with elastic cluster support (SLOW). The basic configuration includes CentOS 7, SLURM, the selected Galaxy flavour, companion software and reference data. Configure, click on the "Submit" button and wait for the confirmation e-mail(s) with instructions on how to provide your passphrase (if encryption is enabled) and log in to your new Galaxy instance. If after some hours you do not receive any e-mail please be sure to check your SPAM BOX.

Instance description

Virtual hardware [Galaxy](#) [Advanced](#)

Instance flavour

Medium (2 cpu, 4 GB RAM, 20 GB dsk) ▼
CPUs, memory size (RAM), root disk size

Worker nodes number

1 worker node ▼
Number of worker nodes in the cluster

Worker nodes flavour

Medium (2 cpu, 4 GB RAM, 20 GB dsk) ▼
CPUs, memory size (RAM), root disk size

Galaxy instance SSH public key

Leave blank this field to load your default SSH public key
Paste here your SSH public key or configure a default key

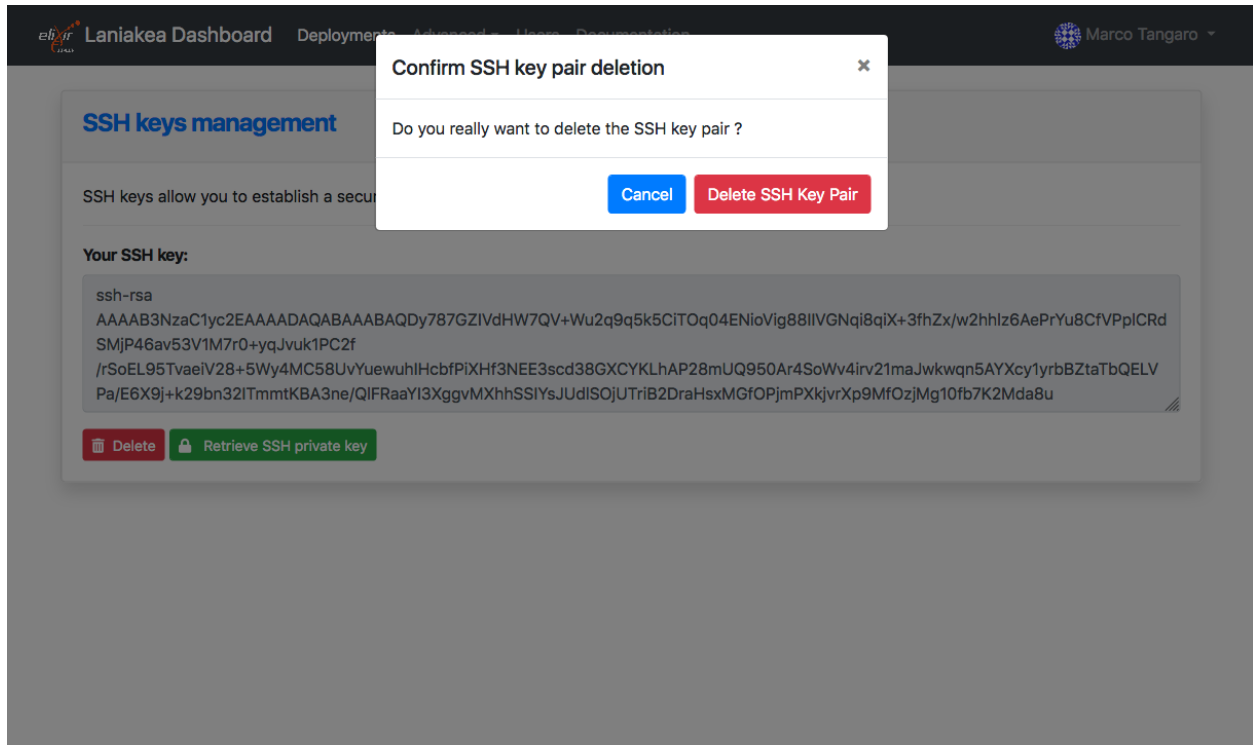
Enable encryption

☐ Off

Encrypt instance external storage

Storage volume size

50 GB ▼
Select storage size



9.4 How to create SSH keys on Windows

<https://docs.microsoft.com/en-us/azure/virtual-machines/linux/ssh-from-windows>

CHAPTER 10

Virtual hardware presets

Each cloud provider enable a set of Image Flavor, defined in terms of Virtual CPUs (VCPUS), Memory, Disk, etc.

10.1 Laniakea@ReCaS

Currently, the following pre-sets are available at ReCaS-Bari facility:

Name	VCPUs	RAM	Disk	Enabled
small	1	2 GB	20 GB	No
medium	2	4 GB	20 GB	No
large	4	8 GB	20 GB	Yes
xlarge	8	16 GB	20 GB	Yes
xxlarge	16	32 GB	20 GB	No

Note: New flavors can be assigned to particular projects.

Note: The storage associated to each instance is configured separately.

Each Galaxy instance is customizable, through the web front-end, with different sets of pre installed tools (e.g. SAM-tools, BamTools, Bowtie, MACS, RSEM, etc...), exploiting CONDA as default dependency resolver. New tools are automatically installed using the official GalaxyProject python library [Ephemeris](#).

Currently the following Galaxy flavours are available on Laniakea

11.1 Galaxy minimal

Description Galaxy production-grade server (Galaxy, PostgreSQL, NGINX, proFTPD, uWSGI).

Reference data repository [usegalaxy.org](#) **Galaxy reference data** **CVMFS repository**

11.2 Galaxy CoVaCS

Description Workflow for genotyping and variant annotation of whole genome/exome and target-gene sequencing data.

For more information on CoVaCs Flavour visit this page: [galaxy_covacs](#).

Reference data repository [ELIXIR-IT Galaxy CoVaCS reference data](#) **CVMFS repository**

Reference <https://www.ncbi.nlm.nih.gov/pubmed/29402227>

11.3 Galaxy GDC Somatic Variant

Description Port of the Genomic Data Commons (GDC) pipeline for the identification of somatic variants on whole exome/genome sequencing data.

For more information on GDC Somatic Variant visit this page: [galaxy_gdc](#).

Reference data repository [usegalaxy.org](#) **Galaxy reference data** **CVMFS repository**

Reference <https://gdc.cancer.gov/node/246>

11.4 Galaxy RNA workbench

Description More than 50 tools for RNA centric analysis.

Reference data repository usegalaxy.org **Galaxy reference data CVMFS repository**

Reference <https://www.ncbi.nlm.nih.gov/pubmed/28582575>

11.5 Galaxy Epigen

Description Based on Epigen project.

Reference data repository usegalaxy.org **Galaxy reference data CVMFS repository**

Reference [Galaxy Epigen server](#)

11.5.1 Create new Galaxy flavours

New flavors can be created through yaml recipes with the list of tools. A tool list example can be found [here](#).

For more information on how to create a flavour visit this page: *[Submit your flavour](#)*.

11.5.2 References

[Galaxy flavors](#)

[Ephemeris](#)

[Ephemeris documentation](#)

[Conda for Galaxy tools dependencies](#)

CHAPTER 12

Submit your flavour

Note: To follow this procedure basic knowledge of Git is needed. If you feel unsure you can contact us using our support mail address (laniakea.helpdesk@gmail.com) and we will be happy to assist you in creating your flavour.

New flavours can be easily added to Laniakea through a Pull Request on our [GitHub page](#).

In this step will be described how to make a Pull Request to the Laniakea GitHub repository to create a new flavour.

1. Fork the Laniakea GitHub [Galaxy flavours repository](#).
2. Clone the forked repository:

```
git clone https://github.com/<user-name>/Galaxy-flavours.git
```

3. Create a new directory with the name of your flavour. For example, `galaxy-testing` in this case.

```
mkdir galaxy-testing
```

4. To create a new Galaxy flavour, a tool list file, written in YAML syntax, has to be provided. The examples directory provides some samples.

Move in the flavour directory:

```
cd galaxy-testing
```

Edit your tool list file with your favourite text editor adding the following default configuration lines:

```
---

api_key: admin
galaxy_instance: http://localhost:8080
install_resolver_dependencies: true
install_tool_dependencies: false
```

Then, add your tool list. For each tool to install, name, owner and tool_panel_section_label, which labels the tools section in the right Galaxy panel, have to be provided:

```
tools:

- name: fastqc
  owner: devteam
  tool_panel_section_label: "tools"

- name: bowtie2
  owner: devteam
  tool_panel_section_label: "tools"

- name: bowtie_wrappers
  owner: devteam
  tool_panel_section_label: "tools"

- name: sam_to_bam
  owner: devteam
  tool_panel_section_label: "tools"

- name: bam_to_sam
  owner: devteam
  tool_panel_section_label: "tools"
```

In this case the resulting Galaxy tools section will be:

5. If you don't need to add one or more workflows to your flavor, move to the next step.

Create a new directory in your flavour directory:

```
mkdir workflow
```

For example, in our galaxy-testing flavour we have:

```
~/Galaxy-flavours/galaxy-testing$ ls

tool-list.yaml  workflow
```

Navigate in this directory and copy here your Galaxy workflows with .ga extension.

6. We are now ready to create a Pull Request. Add your files to your GitHub repository. For example, for our testing flavour:


```
cd galaxy-testing



$ git add tool-list.yaml workflow/Galaxy-Workflow-test.ga


$ git commit -m "add galaxy-testing flavour"
[master 2bc262d] add galaxy-testing flavour
 2 files changed, 30 insertions(+)
 create mode 100644 galaxy-testing/tool-list.yaml
 create mode 100644 galaxy-testing/workflow/Galaxy-Workflow-test.ga

$ git push
Username for 'https://github.com': mtangaro
Password for 'https://mtangaro@github.com':
Counting objects: 3, done.
Compressing objects: 100% (3/3), done.
```


(continues on next page)

 **Galaxy / testing**

Tools  



Get Data
Send Data
Collection Operations
Lift-Over
Text Manipulation
Convert Formats
Filter and Sort
Join, Subtract and Group
Fetch Alignments/Sequences
Operate on Genomic Intervals
Statistics
Graph/Display Data
Phenotype Association
tools
[FastQC](#) Read Quality reports
[Map with Bowtie for Illumina](#)
[BAM-to-SAM](#) convert BAM to SAM
[SAM-to-BAM](#) convert SAM to BAM
Workflows
All workflows

He
To c

Take

Gal
The

(continued from previous page)

```
Writing objects: 100% (3/3), 356 bytes | 0 bytes/s, done.
Total 3 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/mtangaro/Galaxy-flavours.git
   be92a03..2bc262d  master -> master
```

7. Finally, from GitHub it is possible to create a Pull Request to the Laniakea repository:

We will review and test your flavour and enable it on Laniakea.

These changes must be merged to the main branch of the [Galaxy flavours repository](#). The merge will be done once the flavour has been enabled on Laniakea.

Warning: To enable this changes on Laniakea requires at least 1 working day.

12.1 Tool list configuration options

Keys	Re-quired	Default value	Description
name	yes		This is is the name of the tool to install
owner	yes		Owner of the Tool Shed repository from where the tools is being installed
tool_panel_section_id	yes, if not specified		ID of the tool panel section where you want the tool to be installed. The section ID can be found in Galaxy's shed_tool_conf.xml config file. Note that the specified section must exist in this file. Otherwise, the tool will be installed outside any section.
tool_panel_section_label	yes, if not specified		Display label of a tool panel section where you want the tool to be installed. If it does not exist, this section will be created on the target Galaxy instance (note that this is different than when using the ID). Multi-word labels need to be placed in quotes. Each label will have a corresponding ID created; the ID will be an all lowercase version of the label, with multiple words joined with underscores (e.g., 'BED tools' -> 'bed_tools').
tool_shed_url		https://toolshed.g2.bx.psu.edu/	The URL of the Tool Shed from where the tool should be installed.
revisions		latest	A list of revisions of the tool, all of which will attempt to be installed.
install_tool_dependencies		True	True or False - whether to install tool dependencies or not.
install_repository_dependencies		True	True or False - whether to install repo dependencies or not, using classic toolshed packages

12.2 Conda support

Conda is a package manager like apt-get, yum, pip, brew or guix and it is, currently, used as default dependency resolver in Galaxy.

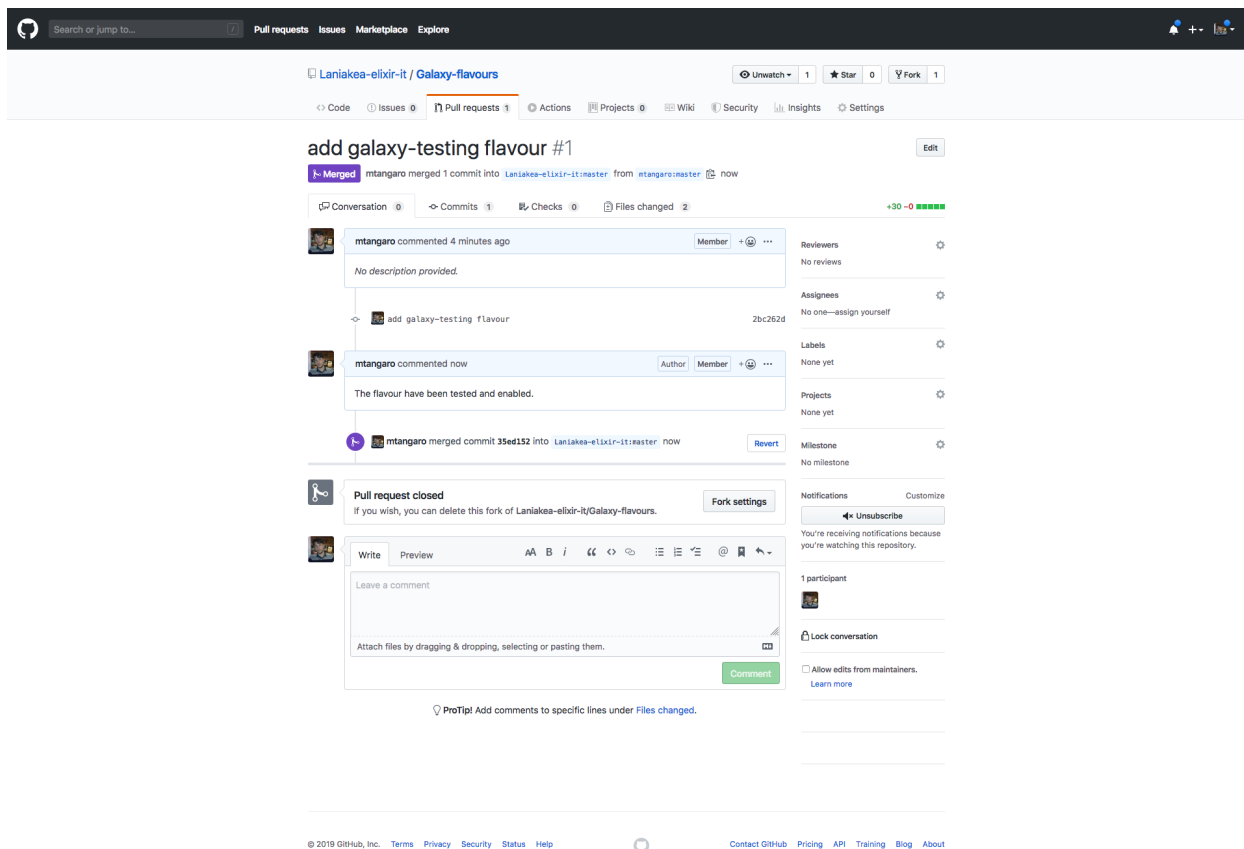
Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

Commits on Dec 19, 2019

2bc262d

Showing 2 changed files with 30 additions and 0 deletions

No commit comments for this range



12.3 References

Galaxy flavors

Ephemeris

Ephemeris documentation

Conda for Galaxy tools dependencies

CHAPTER 13

Reference Data

Many Galaxy tools rely on the presence of reference data, such as alignment indexes or reference genome sequences, to efficiently work. A complete set of Reference Data, able to work with most common tools for NGS analysis is available for each Galaxy instance deployed.

The reference data are available for many species and shared among all the instances, avoiding unnecessary and costly data duplication, exploiting a [CernVM-FS \(CVMFS\)](#) repository.

Laniakea automatically configures Galaxy to properly use them.

By default Laniakea exploits the [usegalaxy.org reference data](#), but for specific needs, e.g. new tools, it is possible to enable, using the Laniakea DASHBOARD, different repositories:

Map with Bowtie for Illumina (Galaxy Version 1.2.0) Options

Will you select a reference genome from your history or use a built-in index?

Use a built-in index

Built-ins were indexed using default options

Select a reference genome

Arabidopsis thaliana (TAIR10)

Is

Arabidopsis thaliana (TAIR9)

Drosophila melanogaster (dm3)

Homo sapiens (hg18)

Homo sapiens (hg19)

Homo sapiens (hg38)

Mus musculus (mm10)

Mus musculus (mm9)

Sa

Saccharomyces cerevisiae (sacCer3)

Yes No

Suppress the header in the output SAM file (--sam-nohead)

Yes No

Bowtie produces SAM with several lines of header information by default

Execute

What it does

Bowtie is a short read aligner designed to be ultrafast and memory-efficient. It is developed by Ben Langmead and Cole Trapnell. Please cite: Langmead B, Trapnell C, Pop M, Salzberg SL. Ultrafast and memory-efficient alignment of short DNA sequences to the human genome. Genome Biology 10.R25.

Know what you are doing

There is no such thing (yet) as an automated gearshift in short read mapping. It is all like stick-shift driving in San Francisco. In other words = running

Fig. 1: Reference data indexes available for bowtie

13.1 data.galaxyproject.org

Description The **usegalaxy.org** CVMFS repository hosts more than 4 TB of reference data. There are two primary directories in the reference data repository:

- **/managed**: Data generated with Galaxy Data Managers, organized by data table (index format), then by genome build.
- **/byhand**: Data generated prior to the existence/use of Data Managers, manually curated.

Currently, the Laniakea instances are preconfigured to mount `/byhand` data. More information can be found [here](#).

For GDC Somatic Variant flavour (`/user_documentation/galaxy/galaxy_gdc`) Galaxy is configured to use also an additional `gdc_tool_data_table_conf.xml`, which can be found [here](#).

13.2 elixir-italy.covacs.refdata

Description This repository hosts specific reference data for CoVaCS pipeline, Laniakea configure the CoVaCS flavours to consume these data.

Reference data cvmfs	Details
cvmfs repository name	elixir-italy.covacs.refdata
cvmfs server url	90.147.75.251
cvmfs config file	elixir-italy.covacs.refdata.conf
cvmfs key file	elixir-italy.covacs.refdata.pub
cvmfs proxy url	DIRECT
galaxy tool data table	tool-data-table.xml

13.3 elixir-italy.galaxy.refdata

Description This repository is recommended only for testing tools and is currently not available on the Laniakea Dashboard. It is used for those tools need to ship reference data still not in the Galaxy Official CVMFS.

Reference data cvmfs	Details
cvmfs repository name	elixir-italy.galaxy.refdata
cvmfs server url	90.147.102.186
cvmfs config file	elixir-italy.galaxy.refdata.conf
cvmfs key file	elixir-italy.galaxy.refdata.pub
cvmfs proxy url	DIRECT
galaxy tool data table	tool-data-table.xml

13.4 Supplementary information

13.4.1 ELIXIR-Italy CVMFS documentation

ELIXIR-Italy maintain two CVMFS repository, exploited by Laniakea.

CVMFS	Flavours supported	folder tree
elixir-italy.covacs.refdata	galaxy CoVaCS	tree structure
elixir-italy.galaxy.refdata	galaxy Epigen, galaxy RNA-workbench, Galaxy GDC Somatic Variant Calling	tree structure

A complete list of the reference data, with download link, is available [here](#).

Default folders structure

The basic structure of the CVMFS repositories is the same. The repository directories are referred to the model organism genome different assemblies:

- at10
- at9
- dm2
- dm3
- dm6
- hg18
- hg19
- hg38
- mm10
- mm8
- mm9
- sacCer1
- sacCer2
- sacCer3

Inside each assembly directory there is the `genome.fa` and the `refseq gtf` and `gff` downloaded from UCSC and the tools indexes:

bwa

It has been created using the default command

```
$ bwa index -a bwtsw genome.fa
```

bowtie2

It has been created using the default command

```
$ bowtie2-build
```

bowtie

Created using the default command

```
$ bowtie-build
```

rsem

Created using the default command

```
$ rsem-prepare-reference --gtf (.gtf) --transcript-to-gene-map (table.txt) --bowtie (.  
↪ fa) <assembly-name>
```

Additional folders

The two repositories hosts also specific directories:

elixir-italy.covacs.refdata

annovar_db

Hosts the databases needed to perform CoVaCS pipeline downloaded from annovar repository using the `annotate_variation.pl` perl script.

bed_file_covacs

Hosts the bed files needed to perform CoVacs pipeline, the same bed files were present in the CINECA implementation of the CoVaCS pipeline.

location

Hosts the `.loc` file and the `tool_data_table.xml` file that will be used by galaxy CoVaCS flavours.

elixir-italy.galaxy.refdata

rRNAdatabase

Location of ribosomal RNA for sortmeRNA tool in galaxy RNA workbench flavour.

index_GATK_bundle

Location of genome indices for GATK tools for hg38 and hg19 assembly downloaded from GATK ftp bundle (<https://software.broadinstitute.org/gatk/download/bundle>).

location

Hosts the `.loc` file and the `tool_data_table.xml` file that will be used by galaxy RNA workbench, galaxy EPIGEN and galaxy GDC Somatic Variant flavours

CVMFS server details

Since, cvmfs relies on OverlayFS or AUFS as default storage driver and Ubuntu 16.04 natively supports OverlayFS, it is used as default choice to create and populate the cvmfs server.

A resign script is located in `/usr/local/bin/Cvmfs-stratum0-resign` and the corresponding weekly cron job is set to `/etc/cron.d/cvmfs_server_resign`.

Log file is located in `/var/log/Cvmfs-stratum0-resign.log`.

13.4.2 Manage CVMFS

The CernVM-File System (conversely cvmfs) provides a scalable, reliable and low-maintenance software distribution service. It was developed to assist High Energy Physics (HEP) collaborations to deploy software on the worldwide distributed computing infrastructure used to run data processing applications.

CernVM-FS is implemented as a POSIX read-only file system in user space (a FUSE module). When initially mounted, CVMFS does not consume any local disk space on the client (in this case, your Galaxy server). Instead, as files are accessed, they are pulled from the server to a local disk-based cache of a configurable size. The reference data files and directories are hosted on standard web servers and mounted on `/cvmfs` directory

For example, listing the CVMFS `elixir-italy.galaxy.refdata` will results in:

```
$ ls -l /cvmfs/elixir-italy.galaxy.refdata/
total 60
drwxr-xr-x. 5 cvmfs cvmfs 4096 May 21 20:10 at10
drwxr-xr-x. 5 cvmfs cvmfs 4096 May 21 20:10 at9
drwxr-xr-x. 3 cvmfs cvmfs 4096 May 21 20:10 dm2
drwxr-xr-x. 7 cvmfs cvmfs 4096 May 21 20:11 dm3
drwxr-xr-x. 7 cvmfs cvmfs 4096 May 21 20:15 hg18
drwxr-xr-x. 7 cvmfs cvmfs 4096 May 21 18:36 hg19
drwxr-xr-x. 7 cvmfs cvmfs 4096 May 21 20:18 hg38
drwxr-xr-x. 7 cvmfs cvmfs 4096 May 21 20:22 mm10
drwxr-xr-x. 3 cvmfs cvmfs 4096 May 21 20:22 mm8
drwxr-xr-x. 7 cvmfs cvmfs 4096 May 21 20:25 mm9
-rw-r--r--. 1 cvmfs cvmfs   57 May 21 18:31 new_repository
drwxr-xr-x. 3 cvmfs cvmfs 4096 May 21 20:25 sacCer1
drwxr-xr-x. 3 cvmfs cvmfs 4096 May 21 20:25 sacCer2
drwxr-xr-x. 7 cvmfs cvmfs 4096 May 21 20:25 sacCer3
-rw-r--r--. 1 cvmfs cvmfs    0 May 21 18:31 test-content
```

Note: The files hosted on a CVMFS repository are pulled from the server only if required, resulting in an empty directory if the file are not required. For example, just listing the directory content will cause the files to be mounted.

Cvmfs client setup

CVMFS is installed by default on each Galaxy instance (CentOS 7 or Ubuntu 16.04). The public key is installed in `/etc/cvmfs/keys/`. The `/etc/cvmfs/default.local` file is also already configured. The `cvmfs_config probe` command mount the cvmfs volume to `/cvmfs`.

Description	Command
check configuration	<code>cvmfs_config chksetup</code>
mount volume	<code>cvmfs_config probe</code>
umount volume	<code>cvmfs_config umount <refdata_repository_name></code>
reload repository	<code>cvmfs_config reload <refdata_repository_name></code>

Note: If mount fails, try to restart autofs with `sudo service autofs restart`.

Note: CVMFS commands require root privileges

The CVMFS repository can be mount also using the `mount` command to a specific mount point:

```
$ sudo mount -t cvmfs elixir-italy.galaxy.refdata /refdata/elixir-italy.galaxy.refdata
CernVM-FS: running with credentials 994:990
CernVM-FS: loading Fuse module... done

$ ls /refdata/elixir-italy.galaxy.refdata/
at10  at9  dm2  dm3  hg18  hg19  hg38  mm10  mm8  mm9  new_repository  sacCer1  ↵
↪sacCer2  sacCer3  test-content
```

Troubleshooting

After an instance reboot, CVMFS is automatically restarted. If this does not happen:

```
$ sudo cvmfs_config_probe
Probing /cvmfs/elixir-italy.galaxy.refdata... Failed!
```

A reload of the config could be able to fix the [problem](#):

```
$ sudo cvmfs_config reload elixir-italy.galaxy.refdata
Connecting to CernVM-FS loader... done
Entering maintenance mode
Draining out kernel caches (60s)
Blocking new file system calls
Waiting for active file system calls
Saving inode tracker
Saving chunk tables
Saving inode generation
Saving open files counter
Unloading Fuse module
Re-Loading Fuse module
Restoring inode tracker... done
Restoring chunk tables... done
Restoring inode generation... done
Restoring open files counter... done
Releasing saved glue buffer
Releasing chunk tables
Releasing saved inode generation info
Releasing open files counter
Activating Fuse module
```

If the file system appears to be hanging, it might have been interrupted during a reload operation. Try to run `sudo cvmfs_config killall` and then again `sudo cvmfs_config probe`.

References

[CernVM-FS](#)

[CVMFS documentation](#)

[Debugging CVMFS](#)

13.4.3 References

[Galaxyproject CVMFS](#)

[CernVM-FS](#)

[CVMFS documentation](#)

[Debugging CVMFS](#)

Galaxy production environment

Laniakea allows to setup and launch a virtual machine (VM) configured with the Operative System (CentOS 7 or Ubuntu 16.04) and the auxiliary applications needed to support a Galaxy production environment such as PostgreSQL, Nginx, uWSGI and Proftpd and to deploy the Galaxy platform itself. A common set of Reference data is available through a CernVM-FS volume. Once deployed each Galaxy instance can be further customized with tools and reference data.

The Galaxy production environment is deployed according to Galaxy official documentation: <https://docs.galaxyproject.org/en/latest/admin/production.html>.

14.1 OS support

CentOS 7 is our default distribution, Given its adherence to Standards and the length of official support (CentOS-7 updates until June 30, 2024, <https://wiki.centos.org/FAQ/General#head-fe8a0be91ee3e7dea812e8694491e1dde5b75e6d>). CentOS 7 and Ubuntu 16.04 are both supported.







Warning: Selinux is by default disabled on CentOS.

14.2 PostgreSQL

PostgreSQL packages coming from PostgreSQL official repository are installed:

Note: Current installed PostgreSQL is: PostgreSQL 9.6

Distribution	Repository
Centos	https://wiki.postgresql.org/wiki/YUM_Installation
Ubuntu	https://wiki.postgresql.org/wiki/Apt

	<ul style="list-style-type: none"> • Users and groups management • System updates • Services management
	<ul style="list-style-type: none"> • Galaxy updates • Galaxy.ini configuration • Virtual environments setup
	<ul style="list-style-type: none"> • Galaxy database management • Separated galaxy tools database
	<ul style="list-style-type: none"> • Web server/web service interface
	<ul style="list-style-type: none"> • NGINX web server + upload module
	<ul style="list-style-type: none"> • FTP instance access • FTP data upload (> 2 GB)

On CentOS 7 the default pgdata directory is `/var/lib/pgsql/9.6/data`. The `pg_hba.conf` configuration is modified allowing for password authentication. On CentOS we need to exclude CentOS base and updates repo for PostgreSQL, otherwise dependencies might resolve to the postgresql supplied by the base repository.

On Ubuntu default pgdata directory is `/var/lib/postgresql/9.6/main`, while the configuration files are stored in `/etc/postgresql/9.6/main`. There's no need to modify the HBA configuration file since, by default, it is allowing password authentication.

PostgreSQL start/stop/status is entrusted to Systemd on CentOS 7 and Ubuntu Xenial.

Distribution	Command
CentOS 7	<code>sudo systemctl start/stop/status postgres-9.6</code>
Ubuntu Xenial	<code>sudo systemctl start/stop/status postgresql</code>

14.2.1 Galaxy database configuration

Two different database are configured to track data and tool shed install data, e.g. allowing to bootstrap fresh Galaxy instance with pretested installs. The database passwords are randomly generated and the password can be retrieved in the `galaxy.yml` file.

Galaxy database is named `galaxy` and is configured in the `galaxy.yml` file:

```
database_connection = postgresql://galaxy:gtLxNnH7DpISmI5FXeeI@localhost:5432/galaxy
```

The shed install tool database is named `galaxy_tools` and is configured as:

```
install_database_connection = postgresql://galaxy:gtLxNnH7DpISmI5FXeeI@localhost:5432/
➔galaxy_tools
```

14.2.2 PostgreSQL troubleshooting

With the recents update (October 2019) the package python2-psycopg2 requires postgresql12-libs, resulting in a broken environment since the package is not available. We avoid this behaviour excluding python python2-psycopg2 update in `/etc/yum.conf` file with the line `exclude=python2-psycopg2`. If you need to update it, just remove it from the exclude line in `/etc/yum.conf`.

14.2.3 Docker configuration

On Docker container PostgreSQL cannot be managed through systemd/upstart, since there's no init system on CentOS and Ubuntu docker images. Therefore, the system is automatically configured to run postgresql using `supervisord`.

14.3 NGINX

To improve Galaxy performance, NGINX is used as web server. The official Galaxy nginx packages are used by default (built in upload module support).

Distribution	Repository
Centos	https://depot.galaxyproject.org/yum/
Ubuntu	ppa:galaxyproject/nginx

Moreover, on Ubuntu, we need to prevent NGINX to be updated by apt default packages. For this purpose the pin priority of NGINX ppa packages is raised, by editing `/etc/apt/preferences.d/galaxyproject-nginx-pin-700` (more on apt pinning at: <https://wiki.debian.org/AptPreferences>).

NGINX is configured following the official Galaxy wiki: <https://galaxyproject.org/admin/config/nginx-proxy/>.

NGINX is started, usually using systemd:

```
$ sudo systemctl start nginx
```

14.3.1 NGINX options

NGINX options are listed here: <https://www.nginx.com/resources/wiki/start/topics/tutorials/commandline/>

To start/stop/status NGINX with systemd:

Distribution	Command
CentOS 7	<code>sudo systemctl start/stop/status nginx</code>
Ubuntu Xenial	<code>sudo systemctl start/stop/status nginx</code>

14.3.2 NGINX troubleshooting

Running NGINX on CentOS through systemd could lead to this error in `/var/log/nginx/error.log`, which can prevent Galaxy web page loading:

```
2017/08/24 08:22:32 [crit] 3320#0: *7 connect() to 127.0.0.1:4001 failed (13:
↪Permission denied) while connecting to upstream, client: 192.167.91.214, server:
↪localhost, request: "GET /galaxy HTTP/1.1", upstream: "uwsgi://127.0.0.1:4001",
↪host: "90.147.102.159"
```

This is related to SELinux policy on CentOS.

Warning: You should avoid to modify SELinux policy, since you can still use NGINX command line options.

Anyway, the problem is that selinux deny socket access. This results in a generic access denied error in NGINX's log, the important messages are actually in selinux's audit log. To solve this issue, you can run the following commands as superuser.

```
# show the new rules to be generated
grep nginx /var/log/audit/audit.log | audit2allow

# show the full rules to be applied
grep nginx /var/log/audit/audit.log | audit2allow -m nginx

# generate the rules to be applied
grep nginx /var/log/audit/audit.log | audit2allow -M nginx

# apply the rules
semodule -i nginx.pp
```

Then restart NGINX.

You may need to generate the rules multiple times (likely four times to fix all policies), trying to access the site after each pass, since the first selinux error might not be the only one that can be generated.

Further readings

[NGINX documentation](#)

[StackOverflow post](#)

[Blog post](#)

14.4 uWSGI

uWSGI (<https://uwsgi-docs.readthedocs.io/en/latest>) is used as interface between the web server (i.e. NGINX) and the web application (i.e. Galaxy). Using uWSGI for production servers is recommended by the Galaxy team: <https://galaxyproject.org/admin/config/performance/scaling/>

uWSGI configuration is embedded in the galaxy.yml file (\$HOME/galaxy/config/galaxy.yml), and by default foresee 4 handler configuration. The number of processes (i.e. uWSGI workers) is set to `number_of_virtual_cpus - 1`. This configuration should be fine for most uses. Nevertheless, there's no golden rule to define the workers number. It is up to the end-user to configure it depending on your needs. The same goes for the number of job handlers (4 by default).

uWSGI socket and stats server are, by default, listening on `127.0.0.1:4001` and `127.0.0.1:9191`, respectively. More on the uWSGI stats server here: <http://uwsgi-docs.readthedocs.io/en/latest/StatsServer.html?highlight=stats%20server>.

```
enable-threads: true
socket: 127.0.0.1:4001
manage-script-name: True
stats: 127.0.0.1:9191
logto: /var/log/galaxy/uwsgi.log
no-orphans: true
```


14.5 Proftpd

To allow user to upload files (> 2GB) through FTP, Proftpd is installed and configured on each Galaxy server, according to: <https://galaxyproject.org/admin/config/upload-via-ftp/>

Proftpd configuration file is located at `/etc/proftpd.conf` on CentOS and `/etc/proftpd/proftpd.conf` on Ubuntu.

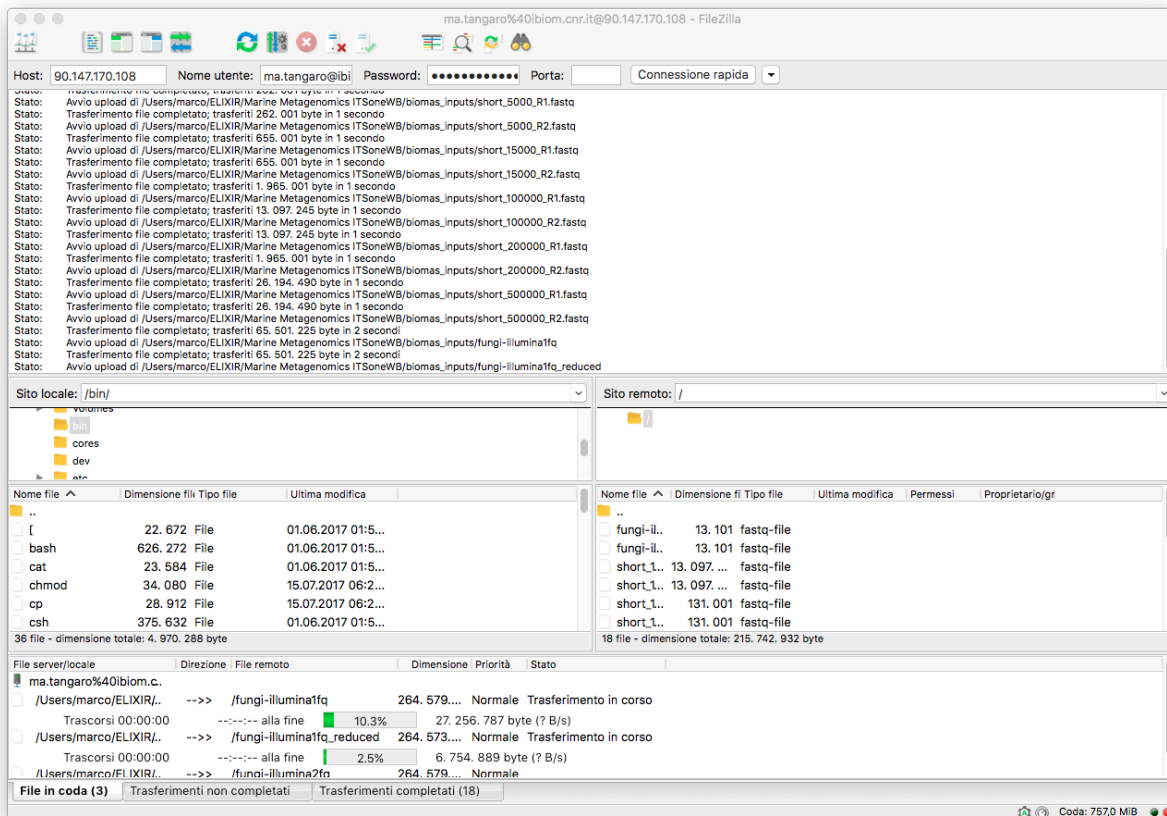
To grant a user access to read emails and passwords from the Galaxy database, a separate user is created for the FTP server which has permission to SELECT from the `galaxy_user` table and nothing else.

Proftpd is listening on port 21. FTP protocol is not encrypted by default, thus any usernames and passwords are sent over clear text to Galaxy.

14.5.1 How to use FTP through FileZilla

Open FileZilla and configure it with:

1. Host: Galaxy ip address (e.g. 90.147.170.108), without the `/galaxy`.
2. User name: your e-mail address on Galaxy.
3. Password: your password on Galaxy.
4. Port: 21



14.5.2 How to use FTP through command line

To install FTP command line client, type `sudo yum install ftp` on CentOS or `sudo apt-get install ftp` on Ubuntu.

To establish a connection with Galaxy Proftpd server, you can use your Galaxy username and password, in addition to the server IP address you're connecting to (e.g. 90.147.102.82). To open a connection in Terminal type the following command, replacing the IP address with your server IP address:

```
$ ftp 90.147.102.82
Connected to 90.147.102.82.
220 ProFTPD 1.3.5e Server (galaxy ftp server) [::ffff:90.147.102.82]
Name (90.147.102.82:marco):
```

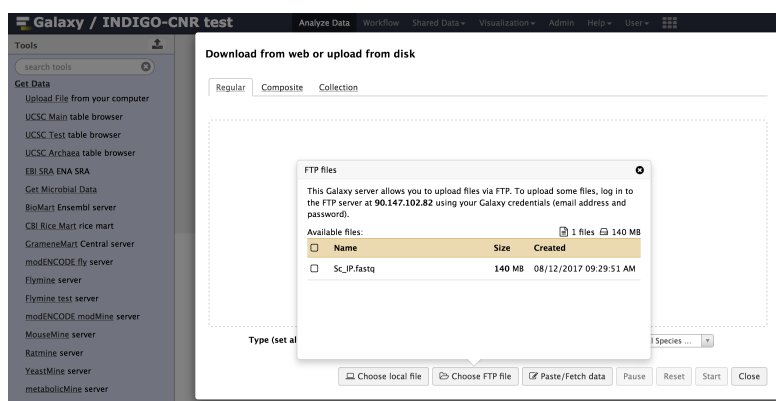
Then login with your Galaxy credentials, typing your Galaxy e-mail address and password:

```
$ ftp 90.147.102.82
Connected to 90.147.102.82.
220 ProFTPD 1.3.5e Server (galaxy ftp server) [::ffff:90.147.102.82]
Name (90.147.102.82:marco): ma.tangaro@gmail.com
331 Password required for ma.tangaro@gmail.com
Password:
```

To upload file to your Galaxy remote directory:

```
ftp> put Sc_IP.fastq
local: Sc_IP.fastq remote: Sc_IP.fastq
229 Entering Extended Passive Mode (|||30023|)
150 Opening BINARY mode data connection for Sc_IP.fastq
8% |*****
↪ | 12544 KiB   23.84 KiB/s   1:31:23 ETA
```

Then you will find it on Galaxy:



Here's a list of the basic commands that you can use with the FTP client.

Command	Description
ls	Is the current directory on the remote machine.
cd	to change directory on the remote machine.
pwd	to find out the pathname of the current directory on the remote machine.
delete	to delete (remove) a file in the current remote directory (same as rm in UNIX).
mkdir	to make a new directory within the current remote directory.
rmdir	to remove (delete) a directory in the current remote directory.
get	to copy one file from the remote machine to the local machine
	get ABC DEF copies file ABC in the current remote directory to (or on top of) a file named DEF in your current local directory.
	get ABC copies file ABC in the current remote directory to (or on top of) a file with the same name, ABC, in your current local directory.
mget	to copy multiple files from the remote machine to the local machine; you are prompted for a y/n answer before transferring each file.
put	to copy one file from the local machine to the remote machine.
mput	to copy multiple files from the local machine to the remote machine; you are prompted for a y/n answer before transferring each file.
quit	to exit the FTP environment (same as bye).

14.6 Supervisord

Supervisor is a process manager written in Python, which allows its users to monitor and control processes on UNIX-like operating systems. It includes:

1. Supervisord daemon (privileged or unprivileged);
2. Supervisorctl command line interface;
3. INI config format;
4. [program:x] defines a program to control.

Supervisord requires root privileges to run.

Galaxy supervisord configuration is located [here](#) and [here](#).

A configuration running the Galaxy server under uWSGI has been installed on `/etc/supervisord.d/galaxy_web.ini` on CentOS, while it is located on `/etc/supervisor/conf.d/galaxy.conf` on Ubuntu. The options `stopasgroup = true` and `killasgroup = true` ensure that the SIGINT signal, to shutdown Galaxy, is propagated to all uWSGI child processes (i.e. to all uWSGI workers).

PYTHONPATH is not specified in this configuration since it was conflicting with Conda.

To manage Galaxy through supervisord:

Action	Command
Start Galaxy	<code>sudo supervisorctl start galaxy:</code>
Stop Galaxy	<code>sudo supervisorctl stop galaxy:</code>
Restart Galaxy	<code>sudo supervisorctl restart galaxy:</code>
Galaxy status	<code>sudo supervisorctl status galaxy:</code>

```
$ supervisorctl help
```

```
default commands (type help <topic>):
```

```
=====
```

```
add      clear  fg        open  quit    remove  restart  start  stop  update
avail    exit   maintail  pid   reload  reread  shutdown status  tail  version
```

```
$ sudo supervisorctl status galaxy:
```

```
galaxy:galaxy_web      RUNNING    pid 9030, uptime 2 days, 21:19:28
galaxy:handler0        RUNNING    pid 9031, uptime 2 days, 21:19:28
galaxy:handler1        RUNNING    pid 9041, uptime 2 days, 21:19:27
galaxy:handler2        RUNNING    pid 9046, uptime 2 days, 21:19:26
galaxy:handler3        RUNNING    pid 9055, uptime 2 days, 21:19:25
```

galaxy_web.ini file configuration:

```
[program:galaxy_web]
command      = /home/galaxy/galaxy/.venv/bin/uwsgi --virtualenv /home/galaxy/
↳ galaxy/.venv --ini-paste /home/galaxy/galaxy/config/galaxy.ini --pidfile /var/log/
↳ galaxy/uwsgi.pid
directory    = /home/galaxy/galaxy
umask        = 022
autostart    = true
autorestart  = true
startsecs   = 20
user         = galaxy
environment  = PATH="/home/galaxy/galaxy/.venv/bin:/usr/local/sbin:/usr/local/bin:/
↳ usr/sbin:/usr/bin:/sbin:/bin"
numprocs     = 1
stopsignal   = INT
startretries = 15
stopasgroup  = true
killasgroup  = true

[program:handler]
command      = /home/galaxy/galaxy/.venv/bin/python ./lib/galaxy/main.py -c /home/
↳ galaxy/galaxy/config/galaxy.ini --server-name=handler%(process_num)s --log-file=/
↳ var/log/galaxy/handler%(process_num)s.log
directory    = /home/galaxy/galaxy
process_name  = handler%(process_num)s
numprocs     = 4
umask        = 022
autostart    = true
autorestart  = true
startsecs   = 20
user         = galaxy
startretries = 15

[group:galaxy]
programs = handler, galaxy_web
```

Finally, a systemd script has been installed to start/stop Supervisor on `/etc/systemd/system/supervisord.service`.

Action	Command
Start	<code>sudo systemctl start supervisord.service</code>
Stop	<code>sudo systemctl stop supervisord.service</code>
Restart	<code>sudo systemctl restart supervisord.service</code>
Status	<code>sudo systemctl status supervisord.service</code>

```
$ sudo systemctl status supervisord.service
supervisord.service - Supervisor process control system for UNIX
Loaded: loaded (/etc/systemd/system/supervisord.service; disabled; vendor preset:
↳ disabled)
Active: active (running) since Sat 2017-08-12 08:48:33 UTC; 9s ago
Docs: http://supervisord.org
Main PID: 12204 (supervisord)
CGroup: /system.slice/supervisord.service
├─12204 /usr/bin/python /usr/bin/supervisord -n -c /etc/supervisord.conf
├─12207 /home/galaxy/galaxy/.venv/bin/uwsgi --virtualenv /home/galaxy/
↳ galaxy/.venv --ini-paste /home/galaxy/galaxy/config/galaxy.ini --pidfile /var/log/
↳ galaxy/uwsgi.pid
├─12208 /home/galaxy/galaxy/.venv/bin/python ./lib/galaxy/main.py -c /home/
↳ galaxy/galaxy/config/galaxy.ini --server-name=handler0 --log-file=/var/log/galaxy/
↳ handler0.log
├─12209 /home/galaxy/galaxy/.venv/bin/python ./lib/galaxy/main.py -c /home/
↳ galaxy/galaxy/config/galaxy.ini --server-name=handler1 --log-file=/var/log/galaxy/
↳ handler1.log
├─12210 /home/galaxy/galaxy/.venv/bin/python ./lib/galaxy/main.py -c /home/
↳ galaxy/galaxy/config/galaxy.ini --server-name=handler2 --log-file=/var/log/galaxy/
↳ handler2.log
└─12211 /home/galaxy/galaxy/.venv/bin/python ./lib/galaxy/main.py -c /home/
↳ galaxy/galaxy/config/galaxy.ini --server-name=handler3 --log-file=/var/log/galaxy/
↳ handler3.log

Aug 12 08:48:33 galaxy-indigo-test supervisord[12204]: 2017-08-12 08:48:33,805 CRIT_
↳ Supervisor running as root (no user in config file)
Aug 12 08:48:33 galaxy-indigo-test supervisord[12204]: 2017-08-12 08:48:33,805 WARN_
↳ Included extra file "/etc/supervisord.d/galaxy_web.ini" during parsing
Aug 12 08:48:34 galaxy-indigo-test supervisord[12204]: 2017-08-12 08:48:34,564 INFO_
↳ RPC interface 'supervisor' initialized
Aug 12 08:48:34 galaxy-indigo-test supervisord[12204]: 2017-08-12 08:48:34,564 CRIT_
↳ Server 'unix_http_server' running without any HTTP authentication checking
Aug 12 08:48:34 galaxy-indigo-test supervisord[12204]: 2017-08-12 08:48:34,565 INFO_
↳ supervisord started with pid 12204
Aug 12 08:48:35 galaxy-indigo-test supervisord[12204]: 2017-08-12 08:48:35,569 INFO_
↳ spawned: 'galaxy_web' with pid 12207
Aug 12 08:48:35 galaxy-indigo-test supervisord[12204]: 2017-08-12 08:48:35,573 INFO_
↳ spawned: 'handler0' with pid 12208
Aug 12 08:48:35 galaxy-indigo-test supervisord[12204]: 2017-08-12 08:48:35,576 INFO_
↳ spawned: 'handler1' with pid 12209
Aug 12 08:48:35 galaxy-indigo-test supervisord[12204]: 2017-08-12 08:48:35,581 INFO_
↳ spawned: 'handler2' with pid 12210
Aug 12 08:48:35 galaxy-indigo-test supervisord[12204]: 2017-08-12 08:48:35,584 INFO_
↳ spawned: 'handler3' with pid 12211
```

14.7 Paths

User data are automatically stored to the “/export” directory, where an external (standard block storage) volume is mounted.

All Galaxy job results are stored in this directory through galaxy.yml (galaxy.ini on galaxy < 18.01) configuration file. For instance, the files directory is located:

```
# Dataset files are stored in this directory.
file_path = /export/galaxy/database/files
```

while the job working directory is located:

```
# Each job is given a unique empty directory as its current working directory.
# This option defines in what parent directory those directories will be
# created.
job_working_directory = /export/job_work_dir
```

Here is the list of Galaxy database path directories:

```
file_path = /export/galaxy/database/files
job_working_directory = /export/job_work_dir
new_file_path = /export/galaxy/database/tmp
template_cache_path = /export/galaxy/database/compiled_templates
citation_cache_data_dir = /export/galaxy/database/citations/data
citation_cache_lock_dir = /export/galaxy/database/citations/lock
whoosh_index_dir = /export/galaxy/database/whoosh_indexes
object_store_cache_path = /export/galaxy/database/object_store_cache
cluster_file_directory = /export/galaxy/database/pbs"
ftp_upload_dir = /export/galaxy/database/ftp
```

14.8 Enable Dockerized tools support in job_conf.xml

Different job_conf.xml configurations to exploit Dockerized tools can be [here](#).

Galaxy Docker instance

The Laniakea Galaxy Docker application runs a Galaxy Docker container inside a Centos 7 virtual machine. The [Official Galaxy Docker image](#) is used. Currently, Laniakea supports the following Docker images:

- [bgruening/galaxy-stable](#)
- [laniakeacloud/galaxy-covacs](#)
- [laniakeacloud/galaxy-gdc_somatic_variant](#)
- [bgruening/galaxy-rna-workbench](#)
- [laniakeacloud/galaxy-epigen](#)

Note: Docker is configured to install all docker-engine files on `/export`, i.e. in the external storage.

15.1 Configuration files

The Docker configuration is slightly customized to make the Galaxy experience as similar as possible to the one on the virtual machine.

- `/etc/galaxy/.myenv.sh`: file with the environment variables of the Docker container.

The customized variables are:

`GALAXY_CONFIG_TOOL_DATA_TABLE_CONFIG_PATH`: `tool_data_table_conf.xml` specific for the galaxy flavour (see section [Galaxy Flavours](#))

`GALAXY_CONFIG_ADMIN_USERS`: `admin_user` - the email selected in the laniakea dashboard

`GALAXY_CONFIG_BRAND`: Galaxy brand - the **Instance description** inserted in the laniakea dashboard

`GALAXY_CONFIG_REQUIRE_LOGIN`: `true` - avoid anonymous login.

`GALAXY_CONFIG_ALLOW_USER_CREATION`: `true` - allow user creation.

`GALAXY_CONFIG_ALLOW_USER_IMPERSONATION: false` - allow user impersonation.

`GALAXY_CONFIG_NEW_USER_DATASET_ACCESS_ROLE_DEFAULT_PRIVATE: true` - By default, users' data will be public, but setting this to `True` will cause it to be private.

`GALAXY_CONDA_PREFIX: path to _conda prefix`

`GALAXY_CONFIG_CONDA_AUTO_INIT: true` - conda auto-start

`GALAXY_CONFIG_CONDA_AUTO_INSTALL: true` - conda auto-install

- `/etc/galaxy/tool_data_tables:` directory with the `tool_data_table_conf.xml` files. A detailed description of Laniakea Galaxy flavours configuration for the reference data is here: [Galaxy Flavours](#).

15.2 CVMFS configuration

The CVMFS repository selected in the Laniakea dashboard is automatically configured and mounted inside the docker directory `/cvmfs`. The corresponding configuration files are in the directory `/etc/cvmfs`.

15.3 Galaxy docker usage

15.3.1 Galaxy docker logs

SSH login in the virtual machine and type:

```
$ sudo docker logs --tail 200 -f galaxydocker
```

15.3.2 Enter in the Docker

In order to access to the Galaxy container, SSH login in the virtual machine and execute the following command:

```
$ sudo docker exec -it galaxydocker bash
```

15.3.3 Main directories in the Docker

Main Galaxy directories inside the Docker container are in `/export`:

- `ftp:` `/export/ftp`
- `database:` `/export/database`
- `conda:` `/export/tool_deps/_conda`

15.3.4 Check Galaxy configuration

In order to see the Galaxy Docker configuration, SSH login in the virtual machine and execute the following command:

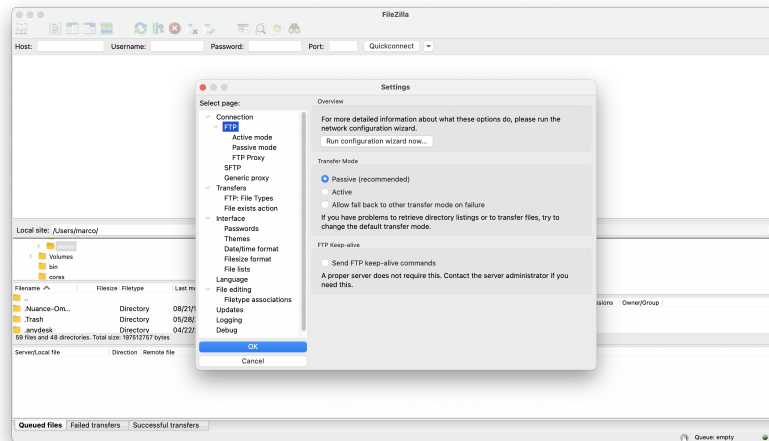
```
$ sudo docker exec -it galaxydocker echo $GALAXY_CONFIG
```


15.3.5 Data upload: FTP

Of course, the Galaxy Docker container allows user to upload data through FTP.

The procedure is similar to that described in the Proftpd section here: /user_documentation/galaxy_production_environment/galaxy_production_environment_configuration.rst.

Moreover, you need to enable FTP Passive mode. Go to Settings..., then to FTP and flag Passive (recommended), as shown in the following picture.



For those using the command line tool, you can enable/disable the passive mode by typing `passive`. First connect to the server then type:

```
passive
```

and you will be in passive mode.

15.4 Galaxy Docker usage tutorial

CHAPTER 16

Cluster configuration

Laniakea provides the possibility to instantiate Galaxy with [SLURM](#) as Resource Manager and to customize the number of virtual worker nodes and the worker nodes and front-end server virtual hardware, e.g. vCPUs and memory.

Furthermore, automatic elasticity, provided using [CLUES](#), enables dynamic cluster resources scaling, deploying and powering on new working nodes depending on the workload of the cluster and powering-off them when no longer needed. This provides an efficient use of the resources, making them available only when really needed.

Conda packages used to solve Galaxy tools dependencies are stored in `/export/tool_deps/_conda` directory and shared between front and worker nodes.

16.1 job_conf.xml configuration

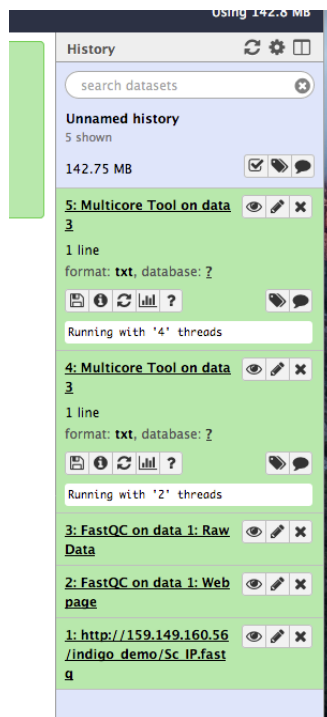
SLURM has been configured following the [GalaxyProject](#) tutorial.

In particular the number of tasks per nodes, i.e. the `$GALAXY_SLOTS`, is set at `--ntasks=2` by default.

Moreover, to allow SLURM restart on elastic cluster, the number of connection retries has been set to 100.

```
<?xml version="1.0"?>
<job_conf>
  <plugins>
    <plugin id="local" type="runner" load="galaxy.jobs.runners.
↪local:LocalJobRunner" workers="2"/>
    <plugin id="slurm" type="runner" load="galaxy.jobs.runners.
↪drmaa:DRMAAJobRunner" workers="100">
      <param id="drmaa_library_path">/usr/local/lib/libdrmaa.so</param>
      <param id="internalexception_retries">100</param>
    </plugin>
  </plugins>
  <handlers default="handlers">
    <handler id="handler0" tags="handlers"/>
    <handler id="handler1" tags="handlers"/>
    <handler id="handler2" tags="handlers"/>
  </handlers>
</job_conf>
```

(continues on next page)



(continued from previous page)

```

    <handler id="handler3" tags="handlers"/>
</handlers>
<destinations default="slurm">
  <destination id="slurm" runner="slurm" tags="mycluster" >
    <param id="nativeSpecification">--nodes=1 --ntasks=2</param>
  </destination>
  <destination id="local" runner="local">
    <param id="local_slots">2</param>
  </destination>
</destinations>
<tools>
  <tool id="upload1" destination="local"/>
</tools>
<limits>
  <limit type="registered_user_concurrent_jobs">1</limit>
  <limit type="unregistered_user_concurrent_jobs">0</limit>
  <limit type="job_walltime">72:00:00</limit>
  <limit type="output_size">268435456000</limit>
</limits>
</job_conf>

```

16.2 Shared file system

Current cluster configuration foresee two paths shared between front and worker nodes:

1. `/home` where Galaxy is installed.
2. `/export` where Galaxy input and output datasets are stored. Here is also mounted the external (encrypted) storage volume, allowing to share it among worker nodes.

Note: The NFS exports configuration file is: `/etc/exports`

For example, listing the mount points in the worker nodes:

```
$ df -h
Filesystem                Size      Used Avail Use% Mounted on
devtmpfs                  1.9G         0   1.9G   0% /dev
tmpfs                     1.9G         0   1.9G   0% /dev/shm
tmpfs                     1.9G       17M   1.9G   1% /run
tmpfs                     1.9G         0   1.9G   0% /sys/fs/cgroup
/dev/vda1                 20G       2.3G   18G  12% /
172.30.66.154:/home       20G       3.9G   17G  20% /home
172.30.66.154:/export    47G      537M   44G   2% /export
tmpfs                     379M         0   379M   0% /run/user/1000
cvmfs2                    4.0G        68K   4.0G   1% /cvmfs/data.galaxyproject.org
```

Note: The CVMFS repository is mounted on each node of the cluster.

16.3 Network configuration

The front node, hosting Galaxy and SLURM, is deployed with a public IP address. Moreover, a private net is created among front and worker nodes. The worker nodes are not exposed to the internet, but reachable only from the front node, because they connected only with the private network.

Instances

Instances											
<div> <div>Instance Name =</div> <div>Filter</div> <div>Launch Instance</div> <div>Delete Instances</div> <div>More Actions</div> </div>											
<input type="checkbox"/>	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/>	lrms_server-157096049010	CentOS 7 1907 base 0.1-2nic	public_net 90.147.170.108 private_net 172.30.66.154	medium	-	Active	nova	None	Running	7 hours, 21 minutes	Create Snapshot
<input type="checkbox"/>	lrms_wn-157096048446	CentOS 7 1907 base 0.1-2nic	172.30.66.153	medium	-	Active	nova	None	Running	7 hours, 21 minutes	Create Snapshot
<input type="checkbox"/>	lrms_wn-157096048446	CentOS 7 1907 base 0.1-2nic	172.30.66.152	medium	-	Active	nova	None	Running	7 hours, 21 minutes	Create Snapshot
<input type="checkbox"/>	lrms_wn-157096048446	CentOS 7 1907 base 0.1-2nic	172.30.66.151	medium	-	Active	nova	None	Running	7 hours, 21 minutes	Create Snapshot

16.4 Worker nodes SSH access

It is possible to SSH login to each deployed worker node from the front node, i.e. the Galaxy server.

The SSH public key is available at `/var/tmp/.im/<deployment_uuid>/ansible_key`. The `deployment_uuid` is a random string which identifies your deployment and is the only directory in the path `/var/tmp/.im`. For examples:

```
# cd /var/tmp/.im/748ee382-ed9f-11e9-9ace-fa163eefe815/
(.venv) [root@slurmserver 748ee382-ed9f-11e9-9ace-fa163eefe815]# ll ansible_key
ansible_key      ansible_key.pub
```



The list of the worker nodes ip address is in the `Output values` tab of the deployment, as `wn_ips`:

Finally, you can connect to worker nodes as:

```
ssh -i ansible_key cloudadm@<wn_ip_address>
```

where `wn_ip_address` is the worker node ip address

16.5 Worker nodes deployment on elastic cluster

Warning: Each node takes 12 minutes or more to be instantiated. Therefore, the job needs the same time to start. On the contrary, if the node is already deployed, the job will start immediately.

This is due to:


1. Virtual Machine configuration
2. CernVM-FS configuration
3. SLURM installation and configuration

During the worker node deployment and delete procedure the Dashboard will show the status `UPDATE_IN_PROGRESS`:

When the worker node is up and running or once it is deleted the Dashboard will show the status `UPDATE_COMPLETE`:

16.6 References

Connecting Galaxy to a compute cluster


 Laniakea Dashboard

Deployments


Advanced ▾

Users

Documentation

 Marco Tangaro ▾



My deployments

 Refresh

[+ New deployment](#)


Show entries

Search:

Instance name	Status	Creation time	Galaxy flavour	VM flavour	Endpoint	Actions
History test 2	CREATE_COMPLETE	2019-10-09 16:33:00	galaxy-epigen	medium	http://90.147.75.159/galaxy	 Delete ▾
elastic cluster test	UPDATE_IN_PROGRESS	2019-10-04 18:32:00	galaxy-minimal	cluster	http://90.147.102.53/galaxy	 Delete ▾

Showing 1 to 2 of 2 entries

Previous **1** Next


 Laniakea Dashboard

Deployments


Advanced ▾

Users

Documentation

 Marco Tangaro ▾



My deployments

 Refresh

[+ New deployment](#)

Show entries

Search:

Instance name	Status	Creation time	Galaxy flavour	VM flavour	Endpoint	Actions
History test 2	CREATE_COMPLETE	2019-10-09 16:33:00	galaxy-epigen	medium	http://90.147.75.159/galaxy	 Delete ▾
elastic cluster test	UPDATE_COMPLETE	2019-10-04 18:32:00	galaxy-minimal	cluster	http://90.147.102.53/galaxy	 Delete ▾

Showing 1 to 2 of 2 entries

Previous **1** Next

SLURM main commands

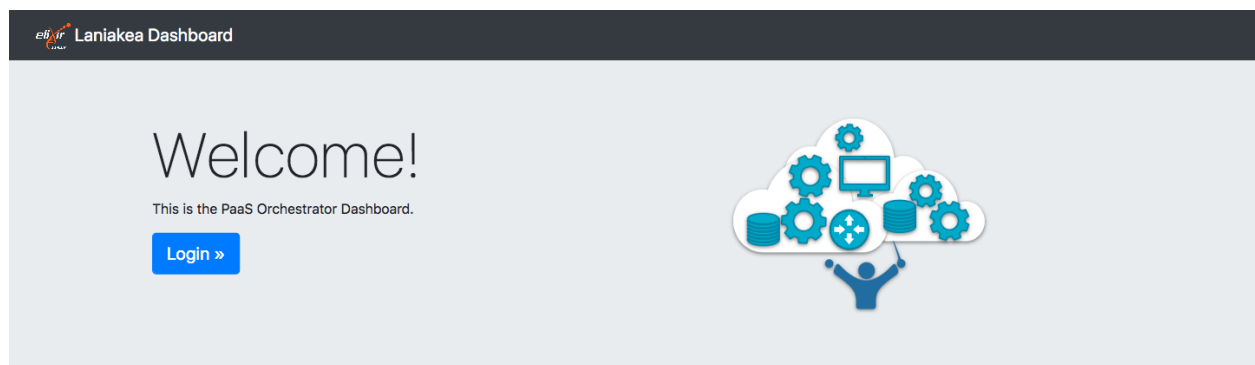
Sbatch commands

CHAPTER 17

Authentication

Currently, the authentication system relies on INDIGO-AAI.

To login into the portal, select the `Sign in` section on top-right:



17.1 Registration

It is needed to register to the portal at the first login. Register with your preferred username or using Google authentication.

Fill the registration form using a valid e-mail address:



Welcome to **recas-bari**

Sign in

[Forgot your password?](#)

 Sign in with Google

Sign in with INFN AAI

Register a new account



Register at **recas-bari**

You have been successfully authenticated with **Google**, but your credentials are **not** yet linked to an **recas-bari** account.

To proceed with the registration please fill in your personal information below.

To abort this registration click [here](#).

Given name

laniakea.testuser

Family name

laniakea-elixir-it

Email

laniakea.testuser@gmail.com

Username

Choose a username

Please choose a username

Notes

and accept the usage policy to complete the registration:

compliance with the law and in accordance with the security directions provided by Computing and Networking Service. They are required to ensure the privacy of processed personal data by proper observance of the rules available at the following web page: www.infn.it/privacy/; take into account the guidelines provided by the Computing and Networking Service concerning the selection of computing devices to use, especially if they concern security-related features. They shall prefer systems and procedures that offer the highest levels of protection; be responsible for the data and for the software they install on the computers entrusted to them: they are required to examine software carefully and in advance and do not install any software with no regular licenses; regularly update the software installed on the computers entrusted to them; protect from unauthorized access data used and/or stored in the computers and systems they are allowed to access; carefully evaluate the reliability of external services, including cloud services, in terms of security, storage and data confidentiality; follow the Computing and Networking Service recommendations concerning the regular backup of data and used programmes; protect their account avoiding to choose obvious passwords and in the event of multiple authentication systems by using different passwords for each system, not share their passwords, nor allow even occasional use by anyone other than the account holder; immediately notify any incidents, suspected abuses and security breaches to their contact person and to the Computing and Networking Service; use updated anti-virus software where operating systems require that. They shall take care to scan all software and files exchanged over the network and all removable media they use; not maintain unused remote connections nor leave their resources unattended with unprotected open connections. I hereby declare of having read and understood and to accept the Acceptable Use Policy described in the present document. Moreover, I declare that any violations of national or international laws and of the terms attached to the present document, linked to the ReCaS-Bari computing resources assigned to me, will be my sole responsibility.

By submitting this registration request, you agree to the terms of this organization Acceptable Usage Policy (AUP).

[Register](#) [Reset Form](#)

A confirmation e-mail is the sent your e-mail address:



Request submitted successfully

Your registration request has been submitted successfully.

An email with a confirmation link is being sent to the email address provided in the registration form. Check your mail!

[Back to Login Page](#)

You don't need to answer to this mail, just follow the instructions, going to the link in the e-mail.

Once confirmed, your request has to be approved by the site administrators. This usually does not require too much time.

Once your request is approved, you will be notified by mail and asked to insert your password.

Finally at the first login you have to allow the Laniakea portal to acquire your login information:

Confirm your recas-bari registration request Posta in arrivo ✕



iam@ba.infn.it
a me ▾

🌐 inglese ▾ > italiano ▾ [Traduci messaggio](#)

Dear laniakea.testuser laniakea-elixir-it,

you have requested to be a member of recas-bari.

In order for the registration to proceed, please confirm this request by going to the following URL:

<https://iam.recas.ba.infn.it/registration/verify/2550a255-0db1-4f86-a6c1-6b44076a4718>

The recas-bari registration service



Request confirmed successfully

Your registration request has been confirmed successfully, and is now waiting for administrator approval. As soon as your request is approved you will receive a confirmation email.

[Back to Login Page](#)

Your recas-bari account is now active Posta in arrivo ✕



iam@ba.infn.it
a me ▾

🌐 inglese ▾ > italiano ▾ [Traduci messaggio](#)

Dear laniakea.testuser laniakea-elixir-it,

your registration request has been approved.

You can set your password by following this link:

<https://iam.recas.ba.infn.it/iam/password-reset/token/af06b392-ea0a-4db3-9331-176e194e6090>

The recas-bari registration service



Set your password

[Save](#)



Your password has been reset successfully!

[Back to Login Page](#)

RECAPS IAM Test Instance for RECAPS-BARI

mtangaro_elixdraai

Approval Required for *dashboard_test*

Caution:

This client was dynamically registered .
It has **never** been approved previously.

[more information](#)

You will be redirected to the following page if you click Approve:

<https://cloud-90-147-170-32.cloud.ba.infn.it/login/iam/authorized>

Access to:

- ☒ log in using your identity
- ☒ basic profile information
- ☒ email address
- ☒ physical address
- ☒ telephone number
- ☒ offline access

Remember this decision:

- ☒ remember this decision until I revoke it
- ☐ remember this decision for one hour
- ☐ prompt me again next time

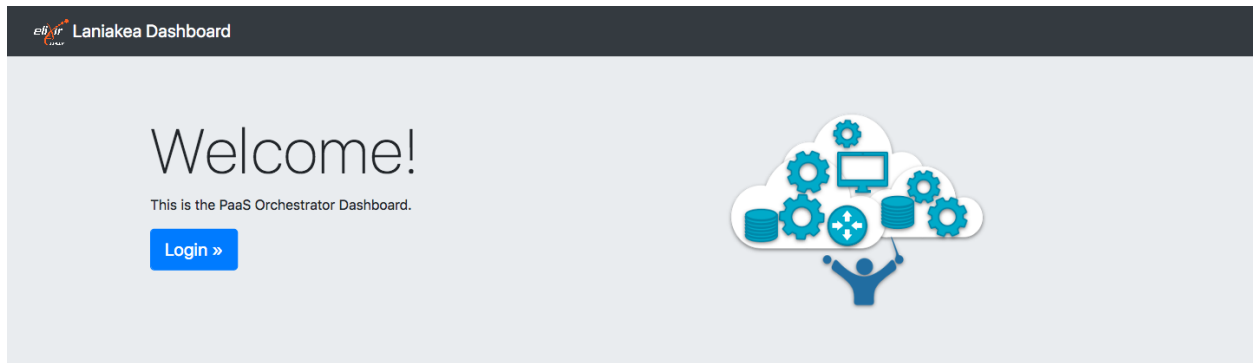
Do you authorize " *dashboard_test* " ?

Authorize

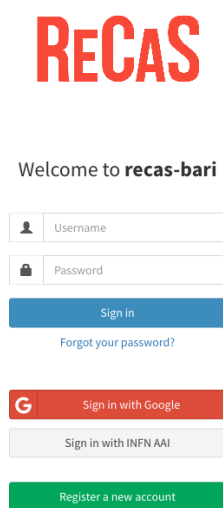
Deny

17.2 Login

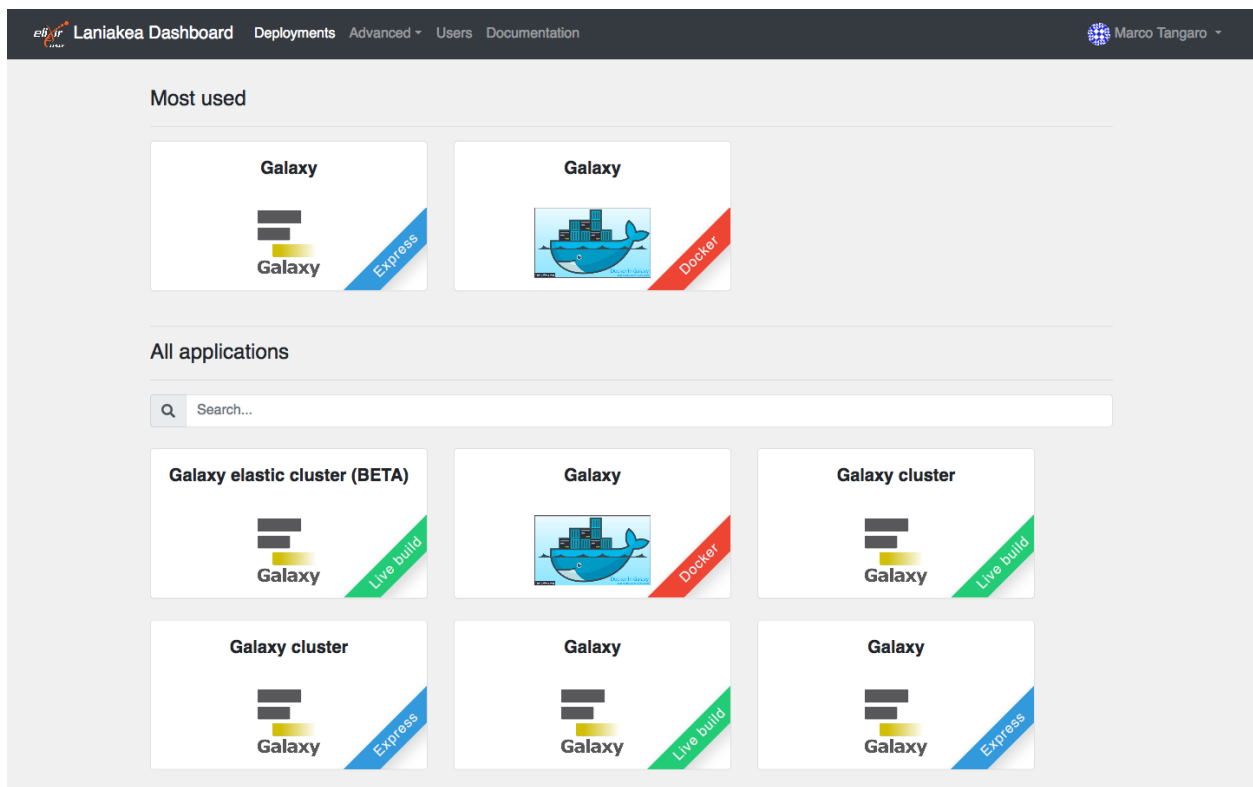
To login into the portal, select the `Sign in` section on top-right:



Then insert your credentials or login using another authentication provider, you used during the registration procedure, like Google.



Finally, you can access the dashboard and instantiate Galaxy:



Frequently Asked Questions

Laniakea FAQs.

18.1 How to manually recover Galaxy after VM reboot

`galaxy_restart`

18.2 I'm unable to create users from admin panel

`galaxy_user_create`

The encryption layer

While the adoption of a distributed environment for data analysis makes data difficult to be tracked and identified by a malevolent attacker, full data anonymity and isolation is still not granted.

The user data privacy is granted through LUKS storage encryption as a service: data are isolated from any other instance on the same platform and from the cloud service administrators. In the past version, users were required to insert a password to encrypt/decrypt data directly on the virtual instance during its deployment, through SSH connection.

In the second Laniakea release the encryption procedure has been completely re-worked and automated in order to simplify the user experience: now the user can encrypt storage on-demand, using a strong random alphanumerical passphrase, without having to interact with the Galaxy instance through SSH. This has been achieved integrating the key management system Hashicorp Vault (vaultproject.io) to store encryption keys, which are shown in the Laniakea Dashboard only if explicitly requested by the user.

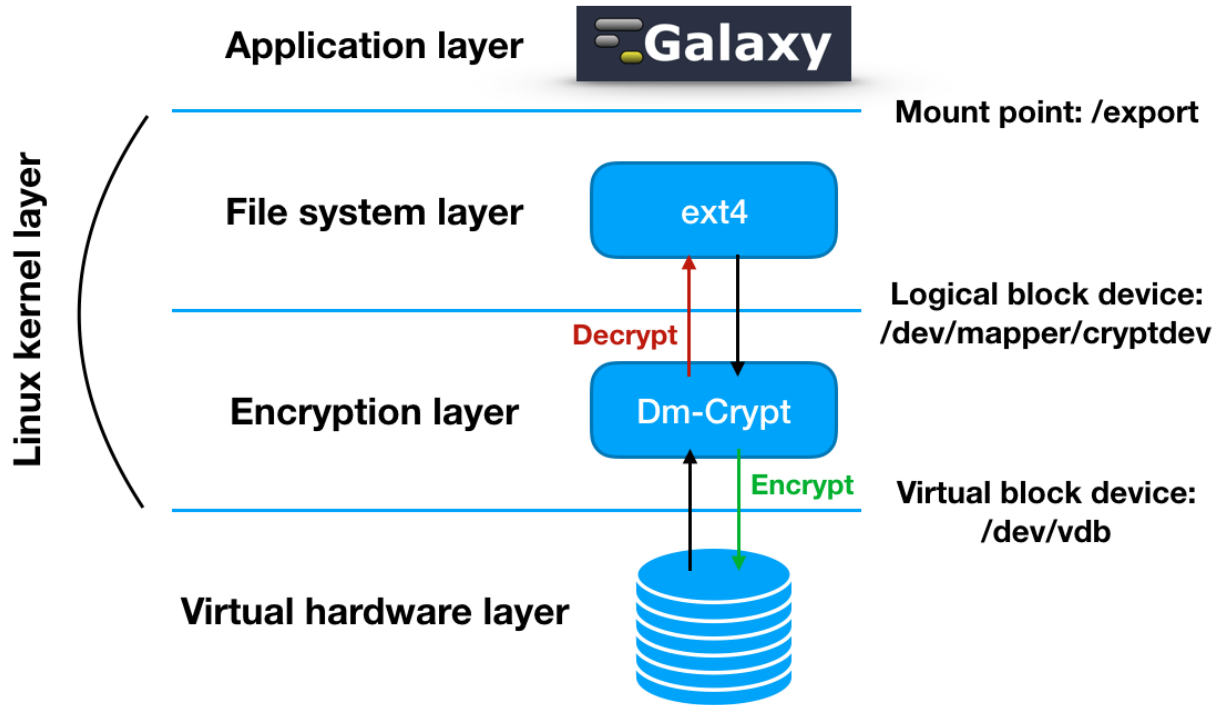
Disk encryption ensures that files are stored on disk in an encrypted form: the files only become available to the operating system and applications in readable when the volume is unlocked by a trusted user. The adopted block device encryption method, operates below the filesystem layer and ensures that everything is written to the block device (i.e. the external volume) is encrypted.

The encryption layer sits between the physical disk and the file system and Galaxy is unaware of storage encryption. Galaxy exploits a specific mount point in order to store and retrieve files. Files are encrypted when stored to disk and decrypted when read.

19.1 The encryption strategy

Device mapper is the Linux kernel driver for volume management and provides transparent encryption of devices through the Linux kernel crypto API, using its device mapper crypt (dm-crypt) module. Dm-crypt is commonly used through Cryptsetup [cryptsetup], a command line interface to dm-crypt, allowing user to setup a new encrypted block device in /dev, specifying the encryption mode, the cipher and the key. Then the device can be formatted with a file system (e.g. ext4), mounted like any other partition and used as persistent storage.

Cryptsetup supports different encryption modes, like plain dm-crypt [cryptsetup] and LUKS volumes [LUKS_web, LUKS_spec] already included in the Linux kernel, but also Loop-AES [loopaes] and TrueCrypt/VeraCrypt [vera]



requiring extra modules installation.

We restricted our choice to dm-crypt usage, which exploits Linux kernel built-in APIs, avoiding the installation of any additional external package other than cryptsetup. In particular, the LUKS encryption grants better usability and flexibility to end users without neglecting data security. Unlike others encryption modes, LUKS stores all dm-crypt setup information in the partition header at the beginning of the block device itself, allowing for multiple passphrases that can be changed and/or revoked anytime. It provides robustness against low-entropy passphrases attack using salting and iterated PBKDF2 passphrase hashing.

Cryptsetup allows for different ciphers usage. A cipher consists of three parts: a block cipher, i.e. it is the encryption algorithm, which operate on fixed-length blocks of data; a block cipher mode of operation, which describes how to repeatedly apply a cipher single block operation to data larger than cipher block size and an Initialization Vector (IV) generator, used to randomize the output of the encryption algorithm, ensuring that the same data are encrypted differently with the same key.

LUKS default cipher is aes-xts-plain64, i.e. AES as block cipher, XTS as mode of operation and plain64 as IV generator. The Advanced Encryption Standard (AES) [AES] is a symmetric-key algorithm, i.e. the same key is used either to encrypt and decrypt data, applying several substitution and permutation rounds to plaintext block to produce encrypted blocks. The Xor encrypt xor Tweakable block Cipher (XTS) mode of operation [XTS1, XTS2] is intended specifically to encrypt data on a block-structured storage device, e.g. disk sectors. The mode works with AES as underlying block cipher which is applied two times to each data chunk: the plain text block is combined with the tweak value, i.e. the plain64 IV, encrypted with AES. Then the block is AES encrypted with the key. Finally, the result is combined again with the tweak value before storing the cipher block.

These options represent the current standard on storage encryption and their modification is strongly discouraged, unless user requires particular configurations. For this reason, even if the Laniakea encryption layer can in theory accept user-defined configuration, e.g. different ciphers, we did not expose these options in the user-interface.

19.2 Storage encryption workflow

When the storage encryption is required by the user the following workflow is triggered:

1. All required software are installed, e.g. cryptsetup.
2. A strong alphanumerical passphrase is generated (100 characters long).
3. The storage is encrypted. Laniakea adopts, by default, xts-aes-plain64 cipher with 256 bit keys and sha256 hashing algorithm.

```
# Defaults values

cipher_algorithm='aes-xts-plain64'
keysize='256'
hash_algorithm='sha256'
device='/dev/vdb'
cryptdev='crypt'
mountpoint='/export'
filesystem='ext4'
```

4. The passphrase is uploaded on Vault, allowing user to retrieve it through the Laniakea dashboard.
5. Once the LUKS partition is created, it is unlocked.

The unlocking process will map the partition to a new device name using the device mapper. This alerts the kernel that device is actually an encrypted device and should be addressed through LUKS using the `/dev/mapper/<cryptdev_name>` so as not to overwrite the encrypted data. `cryptdev_name` is random generated to avoid accidental overwriting.

6. The volume is mounted, by default, on `/export`, with standard `ext4` filesystem and Galaxy is configured to store here datasets.

19.3 File System Encryption Test

Test executed to ensure LUKS volume encryption.

1. Create two volumes, here named `vol1`, `vol2`.
2. Attach each one to the instance (here listed as `/dev/vdd` and `/dev/vde`) and mount them respectively to `/export` and `/export1`.

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
...
/dev/vdd        976M  2.6M  907M   1% /export
/dev/vde        976M  2.6M  907M   1% /export1
```

3. Encrypt `/export`, i.e. `/dev/vdd` using `fast_luks` (`/export` is the default value).

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
...
/dev/vde        976M  2.6M  907M   1% /export1
/dev/mapper/jtdehex 990M  2.6M  921M   1% /export
```

Ensure that `/export` has the same permissions of the other two volumes.

```
drwxr-xr-x.  3 centos centos 4096 Nov  9 10:27 export
drwxr-xr-x.  3 centos centos 4096 Nov  9 10:27 export1
```

4. Put the same file on both volumes:

```
$ echo "encryption test" > /export/test.txt
$ echo "encryption test" > /export1/test.txt
```

5. Umount all the volumes and luksClose the encrypted one:

```
$ sudo cryptsetup luksClose /dev/mapper/jtdehex
```

6. Create the volume binary image using dd:

```
sudo dd if=/dev/vdd of=/home/centos/vdd_out
2097152+0 records in
2097152+0 records out
1073741824 bytes (1.1 GB) copied, 21.809 s, 49.2 MB/s

$ sudo dd if=/dev/vde of=/home/centos/vde_out
2097152+0 records in
2097152+0 records out
1073741824 bytes (1.1 GB) copied, 21.3385 s, 50.3 MB/s
```

7. HexDump the binary image with xdd:

```
$ xxd vdd_out > vdd.txt
$ xxd vde_out > vde.txt
```

As output you should have:

```
$ ls -ltrh
-rw-r--r--. 1 root root 1.0G Nov  9 11:19 vdd_out
-rw-r--r--. 1 root root 1.0G Nov  9 11:22 vde_out
-rw-rw-r--. 1 centos centos 4.2G Nov  9 11:32 vdd.txt
-rw-rw-r--. 1 centos centos 4.2G Nov  9 11:36 vde.txt
```

8. Grep non-zero bytes and search for the test.txt file content encryption test:

```
$ grep -v "0000 0000 0000 0000 0000 0000 0000 0000" vde.txt > grep_vde.txt
$ grep "encryption test" grep_vde.txt
8081000: 656e 6372 7970 7469 6f6e 2074 6573 740a  encryption test.

$ grep -v "0000 0000 0000 0000 0000 0000 0000 0000" vdd.txt > grep_vdd.txt
$ grep "encryption test" grep_vdd.txt
$
```

Note: It is possible to see the test.txt file content only on the un-encrypted volume.

Moreover, the output file grep_vde.txt is 73 kb while the encrypted one, grep_vdd.txt (138 MB), is very large:

```
-rw-rw-r--. 1 centos centos 73K Nov  9 11:46 grep_vde.txt
-rw-rw-r--. 1 centos centos 138M Nov  9 11:58 grep_vdd.txt
```

We also tried to open the volume when active (LUKS volume opened and mounted, Galaxy running) in the Virtual Machine, using the cloud controller (as administrator).

Test executed on the cloud controller:

```
# rbd map volume-3bedc7bc-eaed-466f-9d55-f2c29b44a7b2 --pool volumes
/dev/rbd0

# lsblk -f
NAME                FSTYPE      LABEL UUID                                MOUNTPOINT
sda
|-sda1              ext4                db06fc46-7231-4189-ba2b-0b0117049680    /boot
|-sda2
|-sda5              swap                e5b98538-8337-4e25-8f82-f97f04258716    [SWAP]
`-sda6              LVM2_member        n4SAgY-GRNy-4Fl2-ROoQ-rRIf-bdBP-QC1B6s
  `--vg00-root      ext4                1e3f1ff1-8677-4236-8cb4-07d5cad32441    /
rbd0                crypto_LUKS        c4bee3b9-e0dc-438e-87ae-2a3e491081c0

# mount /dev/rbd0 /mnt/
mount: unknown filesystem type 'crypto_LUKS'
```

It is not possible to mount the volume without the user password.

19.4 Fast-luks script

The `fast-luks` bash script is responsible for Laniakea Storage encryption. It parse common cryptsetup parameters to encrypt the volume. For this reason it checks for cryptsetup and dm-setup packages and it install cryptsetup, if not installed.

The default encryption parameters are:

```
cipher_algorithm: aes-xts-plain64
keysize: 256
hash_algorithm: sha256
device: /dev/vdb
cryptdev: crypt [this is randomly generated]
mountpoint: /export
filesystem: ext4
```

From version v3.0.1 Hashicorp Vault support has been integrated. It exploits a Vault token with the right write policy only, which can be used only one time and for a limited time duration (currently configured to expire after 12 hours), to store user secret passphrases. A temporary python virtual environment is created allowing fast-luks to store secrets on vault and then it is deleted.

The `fast-luks` script is automatically downloaded in `/home/galaxy/laniakea_utils/fast-luks`.

Full documentation on fast-luks script is hosted [here](#).

Note: The script requires superuser rights.

19.5 Luksctl: LUKS volumes management

Luksctl is a python script allowing to easily Open/Close and Check LUKS encrypted volumes, parsing dmsetup and cryptsetup commands. It's source code is located on [Laniakea GitHub](#).

Note: The script requires superuser rights.

Module	Action	Description
luksctl	open	Open and mount the encrypted storage
	close	Umount and close the encrypted storage
	status	Show the encrypted storage status

19.5.1 Dependencies

Since the script is going to parse cryptsetup, dmsetup and mount/umount commands, all of them are required

```
cryptsetup
dmsetup
```

19.5.2 Open LUKS volumes

To open LUKS volume, call: `luksctl open`, which will require your LUKS decrypt password:

```
$ sudo luksctl open
Enter passphrase for /dev/disk/by-uuid/9bc8b7c6-dc7e-4aac-9cd7-8b7258facc75:
Name:                ribqvkjj
State:               ACTIVE
Read Ahead:          8192
Tables present:      LIVE
Open count:           1
Event number:         0
Major, minor:        252, 1
Number of targets: 1
UUID: CRYPT-LUKS1-9bc8b7c6dc7e4aac9cd78b7258facc75-ribqvkjj

Encrypted volume: [ OK ]
```

19.5.3 Close LUKS volumes

To Close LUKS volume, call `luksctl close`:

```
$ sudo luksctl close
Encrypted volume umount: [ OK ]
```

19.5.4 LUKS volumes status

To check if LUKS volume is Open or not call `luksctl status`


```
$ sudo luksctl status
Name:                ribqvkjj
State:               ACTIVE
Read Ahead:          8192
Tables present:      LIVE
Open count:          1
Event number:         0
Major, minor:        252, 1
Number of targets:   1
UUID: CRYPT-LUKS1-9bc8b7c6dc7e4aac9cd78b7258facc75-ribqvkjj

Encrypted volume: [ OK ]
```

19.6 LUKSctl: APIs

A set of RESTFul APIs is distributed with LUKSctl. It is written using python Flask micro framework and Gunicorn. It's source code is located on [Laniakea GitHub](#).

A systemd unit file is used for start/stop/restart the API.

Moudule	Action	Description
luksctl-api	status	Show status
	stop	Stop the API
	start	Start the API.
	restart	Restart the API.

Note: LUKSctl-api is configured to listen on 5000 port.

```
$ sudo systemctl status luksctl-api
luksctl-api.service - Gunicorn instance to serve luksctl api server
   Loaded: loaded (/etc/systemd/system/luksctl-api.service; enabled; vendor preset: ↪
↪disabled)
   Active: active (running) since Fri 2019-10-25 14:23:06 UTC; 1 day 17h ago
 Main PID: 19972 (gunicorn)
    CGroup: /system.slice/luksctl-api.service
            └─19972 /home/luksctl_api/luksctl_api/venv/bin/python /home/luksctl_api/
↪luksctl_api/venv/bin/gunicorn --workers 2...
            └─19995 /home/luksctl_api/luksctl_api/venv/bin/python /home/luksctl_api/
↪luksctl_api/venv/bin/gunicorn --workers 2...
            └─19997 /home/luksctl_api/luksctl_api/venv/bin/python /home/luksctl_api/
↪luksctl_api/venv/bin/gunicorn --workers 2...

Oct 25 14:23:06 slurmserver systemd[1]: Started Gunicorn instance to serve luksctl_
↪api server.
```

(continues on next page)

(continued from previous page)

```
Oct 25 14:23:07 slurmsrver gunicorn[19972]: [2019-10-25 14:23:07 +0000] [19972]
↳[INFO] Starting gunicorn 19.9.0
Oct 25 14:23:07 slurmsrver gunicorn[19972]: [2019-10-25 14:23:07 +0000] [19972]
↳[INFO] Listening at: https://0.0.0.0:...19972)
Oct 25 14:23:07 slurmsrver gunicorn[19972]: [2019-10-25 14:23:07 +0000] [19972]
↳[INFO] Using worker: sync
Oct 25 14:23:07 slurmsrver gunicorn[19972]: [2019-10-25 14:23:07 +0000] [19995]
↳[INFO] Booting worker with pid: 19995
Oct 25 14:23:07 slurmsrver gunicorn[19972]: [2019-10-25 14:23:07 +0000] [19997]
↳[INFO] Booting worker with pid: 19997
Oct 26 07:55:37 slurmsrver sudo[24629]: luksctl_api : TTY=unknown ; PWD=/home/
↳luksctl_api/luksctl_api ; USER=root ; C...status
Oct 27 07:48:04 slurmsrver sudo[21947]: luksctl_api : TTY=unknown ; PWD=/home/
↳luksctl_api/luksctl_api ; USER=root ; C...status
Hint: Some lines were ellipsized, use -l to show in full.
```

It used to connect the Laniakea Dashboard to the encrypted instances, allowing end-user to perform some actions, e.g. to mount and enable the LUKS storage volume, without accessing the Virtual Machine with SSH.

Currently, supported APIs are:

19.6.1 Volume Status

A GET request is used to check the status of the encrypted volume and show it in the Dhasboard. If the volume is open and mounted it return mounted, othrewise it return umounted. If the API is not available, an unavailable status is showed.

Example request:

```
$ curl -k -i -X GET 'https://90.147.75.173:5000/luksctl_api/v1.0/status'
HTTP/1.1 200 OK
Server: gunicorn/19.9.0
Date: Sun, 27 Oct 2019 08:02:54 GMT
Connection: close
Content-Type: application/json
Content-Length: 27

{"volume_state":"mounted"}
```

19.6.2 Volume Open

A POST request can be used to open and mount the encrypted volume in case of VM reboot. To prevent unwanted restart, the API check if the volume is already mounted. If yes it return mounted, otherwise it run luksctl open command.

Example request:

```
curl -k -X POST 'https://<vm_ip_address>:5000/luksctl_api/v1.0/open' -H 'Content-
↳Type: application/json' -d '{ "vault_url": vault_url, "vault_token": wrapping_read_
↳token, "secret_root": vault_secrets_path, "secret_path": secret_path, "secret_key":
↳user_key }'
```

API configuration

To perform the LUKSctl API, Laniakea creates a `luksctl_api` user on the Virtual Machine, and install the LUKSctl on its home directory. This user can only run the LUKS commands as super user, for security reasons. Finally, to secure API communications, a self signed SSL certificate is created and installed.

The LUKSctl API currently support both single VMs and Cluster. Moreover, if the encrypted volume is used to host the Docker Engine files, it can be configured to correctly manage this scenario. This is managed using a json configuration file `config.json`.

Note: Laniakea provides automatic configuration for LUKSctl APIs.

Single VM

Description This is the default API configuration.

config.json

```
{
  "INFRASTRUCTURE_CONFIGURATION": "single_vm"
}
```

Docker

Description The Docker engine files are installed on the encrypted storage, so the Docker daemon needs to be restarted after LUKS volume mount. If `VIRTUALIZATION_TYPE` is set at `docker` after LUKS volume mount, the Docker daemon is restarted.

config.json

```
{
  "INFRASTRUCTURE_CONFIGURATION": "single_vm",
  "VIRTUALIZATION_TYPE": "docker"
}
```

Cluster

Current cluster configuration foresees a NFS between front and worker nodes. If the Front End and/or the Worker Nodes are restarted, once the encrypted volume is opened and mounted, the NFS has to be restarted. If the cluster support is enabled in the API configuration file, after LUKS volume mount, the API contacts each worker nodes, via API, and restart the NFS module.

Front End configuration

Description To enable API cluster support the variable `INFRASTRUCTURE_CONFIGURATION` has to be set at `cluster` on the front end and the worker nodes list has to be provided.

config.json

```
{
  "INFRASTRUCTURE_CONFIGURATION": "cluster",
  "WN_IPS": ["127.0.0.1"]
}
```

Worker Nodes(s) configuration

Description On each worker node, the API needs the list of the NFS shared directories. This list is required to check if all directories have been properly mounted.

config.json

```
{
  "NFS_MOUNTPOINT_LIST": ["/home", "/export"]
}
```

19.7 Cryptsetup hints

The cryptsetup action to set up a new dm-crypt device in LUKS encryption mode is luksFormat:

```
cryptsetup -v --cipher aes-xts-plain64 --key-size 256 --hash sha 256 --iter-time 2000_
↪--use-urandom --verify-passphrase luksFormat crypt --batch-mode
```

where crypt is the new device located to /dev/mapper/crypt.

To open and mount to /export an encrypted device:

```
cryptsetup luksOpen /dev/vdb crypt
mount /dev/mapper/crypt /export
```

To show LUKS device info:

```
dmsetup info /dev/mapper/crypt
```

To unmount and close an encrypted device:

```
umount /export
cryptsetup close crypt
```

To force LUKS volume removal:

```
dmsetup remove /dev/mapper/crypt
```

Note: Run as root.

19.7.1 Change LUKS password

LUKS provides 8 slots for passwords or key files. First, check, which of them are used:

```
cryptsetup luksDump /dev/<device> | grep Slot
```

where the output, for example, looks like:

```
Key Slot 0: ENABLED
Key Slot 1: DISABLED
Key Slot 2: DISABLED
Key Slot 3: DISABLED
Key Slot 4: DISABLED
Key Slot 5: DISABLED
Key Slot 6: DISABLED
Key Slot 7: DISABLED
```

Then you can add, change or delete chosen keys:

```
cryptsetup luksAddKey /dev/<device> (/path/to/<additionalkeyfile>)

cryptsetup luksChangeKey /dev/<device> -S 6
```

As for deleting keys, you have 2 options:

1. delete any key that matches your entered password:

```
cryptsetup luksRemoveKey /dev/<device>
```

2. delete a key in specified slot:

```
cryptsetup luksKillSlot /dev/<device> 6
```

19.8 References

1. [LUKS](#)
2. [Disk encryption archlinux wiki page](#)
3. [Dm-crypt archlinux wiki page](#)
4. [LUKS how-to](#)
5. [Original LUKS script](#) (Credits to John Troon for the original script)

Galaxyctl: Galaxy management

Galaxyctl is a python script collection used for Galaxy management, to properly check uWSGI Stats and to correctly retrieve Galaxy and uWSGI workers status. It's source code is located on [Laniakea GitHub](#).

Note: Since the script parse `supervisorctl` or `systemd` commands, it needs to be run as superuser.

Moudule	Action	Description
galaxy	status	Show galaxy status
	stop	Stop Galaxy. <code>--force</code> check uwsgi master process. If it is still running, after galaxy stop, it is killed.
	start	Start Galaxy. <code>--force</code> force galaxy to start by restarting it. <code>--retry</code> option allow to specify number of tentative retart (default 5). <code>--timeout</code> allow to customize uWSGI stats server wait time. These options are used during galaxy instantiation and you should not use them on production.
	restart	Restart Galaxy. <code>--force</code> force galaxy to start by restarting it. <code>--retry</code> option allow to specify number of tentative retart (default 5). <code>--timeout</code> allow to customize uWSGI stats server wait time. These options are used during galaxy instantiation and you should not use them on production.
	startup	This method is used only to run galaxy for the first time and you shoud not use it in production. <code>--retry</code> option allow to specify number of tentative retart (default 5). <code>--timeout</code> allow to customize uWSGI stats server wait time.

20.1 Galaxyctl basic usage

The script requires superuser commands to be used. Its basic commands are:

Action	Command
Start Galaxy	<code>sudo galaxyctl start galaxy</code>
Stop Galaxy	<code>sudo galaxyctl stop galaxy</code>
Restart Galaxy	<code>sudo galaxyctl restart galaxy</code>
Check Galaxy Status	<code>sudo galaxyctl status galaxy</code>

20.2 Logging

Logs are stored in `/var/log/galaxy/galaxyctl.log` file.

20.3 Advanced options

20.3.1 stop

To stop galaxy:

```
sudo galaxyctl stop galaxy
```

The script check the uWSGI Stats server to retrieve workers PID and their status. If, after uWSGI stop, workers are still up and running, they are killed, allowing Galaxy to correctly start next time. The `--force` options allow to kill uwsgi master process if it is still alive after galaxy stop (in case of uwsgi FATAL error or ABNORMAL TERMINATION). Please check galaxy logs before run `--force` option.

20.3.2 start

To start Galaxy:

```
sudo galaxyctl start galaxy
```

Once Galaxy started, galaxyctl waits and check the uWSGI Stats server. Since it is the last software loaded, this ensure that Galaxy has correctly started. The script also check that at least 1 uWSGI worker has correctly started and it is accepting requests.

If no workers are available you have to restart Galaxy. Galaxyctl is able to automatically restart galaxy if the option `--force` is specified, restarting it until the workers are correctly loaded The number of retries is set, by default, to 5. It can be customized using `--retry` option, e.g. `--retry 10`. These options were not designed for production, but are used only during VMs instantiation phase to ensure Galaxy can correctly start.

20.3.3 restart

To restart Galaxy:

```
sudo galaxyctl restart galaxy
```

The options `--force`, `--timeout` and `--retry` are available for restart command too.

20.3.4 Galaxy first start

Galaxy takes longer to start the first time. Since the uWSGI stats server is the last software component started, the script waits to ensure that Galaxy has correctly started. Then uWSGI workers are checked to ensure Galaxy is accepting requests. If not, uWSGI is restarted. Currently, before rise an error, the script try to restart galaxy 5 times, while the waiting time is set to 600 seconds. The command used in `/usr/local/bin/galaxy-startup` script, is

```
galaxyctl startup galaxy -c /home/galaxy/galaxy/galaxy.ini -t 600
```

20.4 Configuration file

Supervisord and systemd/upstart are supported to start/stop/restart/status Galaxy. The init system can be set using the variables `init_system`: two values are, currently, allowed: `supervisord` and `init`

init_system	Explanation
supervi-sord	Supervisord is current default, it is mandatory for docker container, since there's no systemd on docker images.
init	CentOS 7 and Ubuntu 16.04 use systemd, while Ubuntu 14.04 is using upstart.

Through `galaxyctl_libs.DetectGalaxyCommands` method the script automatically retrieves the right command to be used and it is compatible with both CentOS 7 and Ubuntu 16.04.

If Supervisord is used to manage Galaxy (which is our default choice), configuration files have to be specified using the variable `supervisord_conf_file` On CentOS:

```
supervisord_conf_file = '/etc/supervisord.conf'
```

while on Ubuntu:

```
supervisord_conf_file = '/etc/supervisor/supervisord.conf'
```

Galaxyctl needs `galaxy.yml` to retrieve uWSGI stats server information, through the variable:

```
galaxy_config_file = '/home/galaxy/galaxy/config/galaxy.yml'
```

20.5 Features

20.5.1 Galaxyctl: libraries

Galaxyctl is a python script collection for Galaxy management (first start, stop/start/restart/status).

Note: Galaxyctl requires superuser privileges.

Note: Current version: v2.0.0

Script	Description
galaxyctl_libs	Python libraries for uWSGI socket and stats server management, LUKS volume and Onedata space management.
galaxyctl	Galaxy management script. It integrates Luksctl and Onedatactl commands.

Galaxyctl_libs is composed by several modules.

Dependencies

Galaxyctl_libs depends on uWSGI for Galaxy management (i.e. currently no run.sh support). Moreover `lsof` is needed to check listening ports.

```
uwsgi
lsof
```

DetectGalaxyCommands

Parse galaxy Stop/Start/Restart/Status commands. Currently it supports supervisor or systemd/upstart

UwsgiSocket

Get uWSGI socket from galaxy.ini config file (e.g. 127.0.0.1:4001) and using `lsof` return uWSGI master PID.

```
master_pid, stderr, status = UwsgiSocket(fname='/home/galaxy/galaxy/config/galaxy.ini
↪').get_uwsgi_master_pid()
```

UwsgiStatsServer

Read uWSGI stats server json. The stats server is the last software which uWSGI run during galaxy start procedure. When the stats server is ready, galaxy is ready to accept requests. Stats server address and port can be specified, but the class is able to read galaxy.ini file to recover stats informations. Reading Stats json the class is able to detect if uWSGI workers accept requests or not.

In-puts	Description
server	uWSGI stats server address, e.g. 127.0.0.1
port	uUWSG stats server port, e.g. 9191
time-out	Wait time, in seconds, for the Stats server start. If galaxy is starting, 300 seconds as timeout is ok, while if galaxy is already running 5 seconds are enough.
fname	Galaxy config file, e.g. /home/galaxy/galaxy/config/galaxy.ini

GetUwsgiStatsServer

To connect to running uWSGI stats server call:

```
stats = UwsgiStatsServer(timeout=300, fname='/home/galaxy/galaxy/config/galaxy.ini')
socket = stats.GetUwsgiStatsServer()
```

GetUwsgiStatsServer

To check if at least one uWSGI workers accept requests, call:

```
stats = UwsgiStatsServer(timeout=300, fname='/home/galaxy/galaxy/config/galaxy.ini')
status = stats.GetUwsgiStatsServer('/home/galaxy/galaxy/config/galaxy.ini')
```

GetBusyList

To get the list of busy uWSGI workers:

```
stats = UwsgiStatsServer(timeout=5, fname='/home/galaxy/galaxy/config/galaxy.ini')
busy_list = stats.GetBusyList()
```

20.5.2 Galaxyctl: APIs

A set of RESTful APIs is distributed with Galaxyctl. It is written using python Flask micro framework and Gunicorn. A systemd unit file is used for start/stop/restart the API.

Moudule	Action	Description
galaxyctl-api	status	Show status
	stop	Stop the API
	start	Start the API.
	restart	Restart the API.

Note: Galaxyctl-api is configured to listen on 5001 port.

```
$ sudo systemctl status galaxyctl-api
galaxyctl-api.service - Gunicorn instance to serve luksctl api server
  Loaded: loaded (/etc/systemd/system/galaxyctl-api.service; enabled; vendor preset:
↳ disabled)
  Active: active (running) since Wed 2019-10-09 16:49:57 UTC; 2 weeks 2 days ago
  Main PID: 15648 (gunicorn)
  CGroup: /system.slice/galaxyctl-api.service
          └─15648 /home/galaxy/.galaxyctl/api/venv/bin/python /home/galaxy/.
↳ galaxyctl/api/venv/bin/gunicorn --workers 2 --b...
          └─15662 /home/galaxy/.galaxyctl/api/venv/bin/python /home/galaxy/.
↳ galaxyctl/api/venv/bin/gunicorn --workers 2 --b...
          └─15663 /home/galaxy/.galaxyctl/api/venv/bin/python /home/galaxy/.
↳ galaxyctl/api/venv/bin/gunicorn --workers 2 --b...

Oct 09 16:49:57 vnode-0.localdomain systemd[1]: Started Gunicorn instance to serve
↳ luksctl api server.
Oct 09 16:49:58 vnode-0.localdomain gunicorn[15648]: [2019-10-09 16:49:58 +0000]
↳ [15648] [INFO] Starting gunicorn 19.9.0
Oct 09 16:49:58 vnode-0.localdomain gunicorn[15648]: [2019-10-09 16:49:58 +0000]
↳ [15648] [INFO] Listening at: http://0....5648)
Oct 09 16:49:58 vnode-0.localdomain gunicorn[15648]: [2019-10-09 16:49:58 +0000]
↳ [15648] [INFO] Using worker: sync
Oct 09 16:49:58 vnode-0.localdomain gunicorn[15648]: [2019-10-09 16:49:58 +0000]
↳ [15662] [INFO] Booting worker with pid: 15662
Oct 09 16:49:58 vnode-0.localdomain gunicorn[15648]: [2019-10-09 16:49:58 +0000]
↳ [15663] [INFO] Booting worker with pid: 15663
Hint: Some lines were ellipsized, use -l to show in full.
```

It used to connect the Laniakea Dashboard to the Galaxy instances, allowing end-user to perform some actions, e.g. to restart Galaxy, without accessing the Virtual Machine with SSH.

Currently, supported APIs are:

Restart Galaxy

A POST request is used to restart Galaxy if offline. To prevent unwanted restart, the API check if Galaxy is on line. If yes it return on-line else it run the galaxy-startup script. Also NGINX is restarted.

Example request:

```
$ curl 'http://<galaxy_ip_address>:5001/galaxyctl_api/v1.0/galaxy-startup' -i -X POST
↪-H 'Content-Type: application/json' -d '{"endpoint": "http://<galaxy_ip_address>/
↪galaxy"}'
```

Laniakea Ansible Roles

Ansible automates Galaxy installation and configuration using Ansible roles. These roles make extensive use of Ansible Modules, which are the ones that do the actual work in ansible, they are what gets executed in each playbook task. Furthermore, a python scripts collection for galaxy advanced configuration is used (run by ansible).

Note: All roles can be easily installed through `ansible-galaxy`.

21.1 indigo-dc.galaxycloud

Description Install Galaxy Production environment, i.e. Galaxy with all needed software, PostgreSQL, NGINX, Proftpd and uWSGI. The role also installs `Galaxyctl` and its API for Galaxy management.

Installation

```
# ansible-galaxy install indigo-dc.galaxycloud
```

Documentation <https://github.com/indigo-dc/ansible-role-galaxycloud>

21.2 indigo-dc.galaxycloud-os

Description This role provides storage encryption with aes-xts-plain64 algorithm using LUKS for Galaxy instances. The role installs and run `fast-luks` for storage encryption, and `LUKSctl` and `LUKSctl` APIs for storage management.

Installation

```
ansible-galaxy install indigo-dc.galaxycloud-os
```

Documentation <https://github.com/indigo-dc/ansible-role-galaxycloud-os>

21.3 indigo-dc.galaxycloud-tools

Description Automated installation of tools from a Tool Shed into Galaxy. The role use the path scheme from the `indigo-dc.galaxycloud` role. It creates a virtual environment, install ephemeris and invoke the install script to tools into Galaxy. The script stop Galaxy (if running), start a local Galaxy instance on `http://localhost:8080` and install tools. The list of tools to install is provided in `files/tool_list.yaml` file, hosted in the `external repository`. Workflows are also installed.

Installation

```
ansible-galaxy install indigo-dc.galaxycloud-tools
```

Documentation <https://github.com/indigo-dc/ansible-role-galaxycloud-tools>

21.4 indigo-dc.galaxycloud-refdata

Description The role provides reference data using the CernVM File System and the corresponding Galaxy configuration.

Installation

```
ansible-galaxy install indigo-dc.galaxycloud-refdata
```

Documentation <https://github.com/indigo-dc/ansible-role-galaxycloud-refdata>

21.5 indigo-dc.galaxycloud-fastconfig

Description Ansible role for Galaxy fast configuration on Virtual Machines with Galaxy and tools already inside, created using `indigo-dc.galaxycloud` role. The documentation on Galaxy Express services, which exploits this role, is: `/admin_documentation/indigo_paas_deploy/galaxy_vm`.

Installation

```
ansible-galaxy install indigo-dc.galaxycloud-fastconfig
```

Documentation <https://github.com/indigo-dc/ansible-role-galaxycloud-fastconfig>

21.6 indigo-dc.galaxycloud_docker

Description Run Galaxy Docker containers on a Centos7 (Ubuntu 16.04) virtual machine, creating Galaxy administrator user and mounting specific Cern VM file system. The Docker engine is installed and stored with docker images on the external volume (`/export`).

Installation

```
ansible-galaxy install indigo-dc.galaxycloud_docker
```

Documentation <https://github.com/indigo-dc/ansible-role-galaxycloud-docker>

21.7 indigo-dc.cvmfs-client

Description Ansible role to install CernVM-FS Client.

Installation

```
ansible-galaxy install indigo-dc.cvmfs-client
```

Documentation <https://github.com/indigo-dc/ansible-role-cvmfs-client>

21.8 indigo-dc.cvmfs-server

Description Ansible role to install CernVM FS Server.

Installation

```
ansible-galaxy install indigo-dc.cvmfs-server
```

Documentation <https://github.com/indigo-dc/ansible-role-cvmfs-server>

TOSCA templates



The [INDIGO PaaS Orchestrator](#) is the key software component of the INDIGO PaaS layer: it receives deployment requests from the user interface software layer and coordinates the deployment process over the IaaS platforms. The Orchestrator accepts the deployment requests written using the [TOSCA standard](#), allowing to deploy complex application using small building blocks, named node types, which exploit Ansible to install and configure the end-user applications or services, like Galaxy, on bare OS images. Therefore, to correctly orchestrate Galaxy deployment the following component are needed:

- [Ansible](#) roles to automate software installation and configuration (see section [Laniakea Ansible Roles](#))
- Custom types: define user configurable parameters, node requirements, call ansible playbooks.
- Artifact: define what to install and how to do it, through ansible role configuration.
- TOSCA template: the orchestrator interprets the TOSCA template and orchestrates the deployment.

Note: This section is not intended to be a complete guide to TOSCA types, but aims to describe the solutions adopted to deploy Galaxy in Laniakea.

22.1 Custom types

22.1.1 GalaxyPortal

Galaxy portal installation and configuration is entrusted to the GalaxyPortal custom type.

```
tosca.nodes.indigo.GalaxyPortal:
    derived_from: tosca.nodes.WebServer
```

It is composed by the following sections:

properties

Galaxy input parameters are listed in the **properties** section:

```
properties:
  admin_email:
    type: string
    description: email of the admin user
    default: admin@admin.com
    required: false
  admin_api_key:
    type: string
    description: key to access the API with admin role
    default: not_very_secret_api_key
    required: false
  user:
    type: string
    description: username to launch the galaxy daemon
    default: galaxy
    required: false
  install_path:
    type: string
    description: path to install the galaxy tool
    default: /home/galaxy/galaxy
    required: false
  export_dir:
    type: string
    description: path to store galaxy data
    default: /export
    required: false
  version:
    type: string
    description: galaxy version to install
    default: master
    required: false
  instance_description:
    type: string
    description: galaxy instance description
    default: "INDIGO Galaxy test"
  instance_key_pub:
    type: string
    description: galaxy instance ssh public key
    default: your_ssh_public_key
  flavor:
```

(continues on next page)

(continued from previous page)

```

type: string
description: name of the Galaxy flavor
required: false
default: galaxy-no-tools
reference_data:
  type: boolean
  description: Install Reference data
  default: true
  required: false

```

Note: The `export_dir` property is able to set Galaxy storage location. On single VMs it is set to `/export`, while on Cluster it has to be set to `/home/export`, allowing for data sharing.

requirements

The LRMS, e.g. local, torque, slurm, sge, condor, mesos, is specified in the **requirements** section:

```

requirements:
- lrms:
    capability: tosca.capabilities.indigo.LRMS
    node: tosca.nodes.indigo.LRMS.FrontEnd
    relationship: tosca.relationships.HostedOn

```

artifacts

The needed Ansible roles, installed using `ansible-galaxy`, are listed in the **artifacts** section:

```

artifacts:
  nfs_role:
    file: indigo-dc.nfs
    type: tosca.artifacts.AnsibleGalaxy.role
  galaxy_role:
    file: mtangaro.galaxycloud, master
    type: tosca.artifacts.AnsibleGalaxy.role

```

interfaces

The Ansible role is called with its input parameters:

```

interfaces:
  Standard:
    configure:
      implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/v3.0.1/
        ↪ artifacts/galaxy/galaxy_install.yml
    inputs:
      galaxy_install_path: { get_property: [ SELF, install_path ] }
      galaxy_user: { get_property: [ SELF, user ] }
      galaxy_admin: { get_property: [ SELF, admin_email ] }
      galaxy_admin_api_key: { get_property: [ SELF, admin_api_key ] }
      galaxy_lrms: { get_property: [ SELF, lrms, type ] }

```

(continues on next page)

(continued from previous page)

```
galaxy_version: { get_property: [ SELF, version ] }
galaxy_instance_description: { get_property: [ SELF, instance_description ] }
galaxy_instance_key_pub: { get_property: [ SELF, instance_key_pub ] }
export_dir: { get_property: [ SELF, export_dir ] }
galaxy_flavor: { get_property: [ SELF, flavor ] }
get_refdata: { get_property: [ SELF, reference_data ] }
```

The artifact, called in the implementation line, is located on github [tosca-types/artifacts/galaxy/galaxy_install.yml](#)

```
---
- hosts: localhost
  connection: local
  roles:
    - role: indigo-dc.galaxycloud
      GALAXY_VERSION: "{{ galaxy_version }}"
      GALAXY_ADMIN_EMAIL: "{{ galaxy_admin }}"
      GALAXY_ADMIN_API_KEY: "{{ galaxy_admin_api_key }}"
```

22.1.2 GalaxyPortalAndStorage

GalaxyPortalAndStorage custom type inherits its properties from GalaxyPortal and extends its functionalities for the **storage encryption**:

```
tosca.nodes.indigo.GalaxyPortalAndStorage:
  derived_from: tosca.nodes.indigo.GalaxyPortal
```

properties

The inputs needed to enable the storage encryption and the Hashicorp Vault key management are:

```
properties:
  storage_encryption:
    type: boolean
    description: Enable storage encryption using Vault to store secrets and LUKS to
    ↪encrypt
    default: false
    required: true
  vault_url:
    type: string
    description: Hashicorp Vault server url
    default: vault_url
    required: false
  vault_wrapping_token:
    type: string
    description: Vault Wrapping token to write secret
    default: not_a_valid_token
    required: false
  vault_secret_path:
    type: string
    description: Vault path to store secret
    default: path_to_secret
    required: false
```

(continues on next page)

(continued from previous page)

```

vault_secret_key:
  type: string
  description: Vault secret key name
  default: secret_key_name
  required: false
wn_ips:
  type: list
  entry_schema:
    type: string
  description: List of IPs of the WNs
  required: false
  default: []

```

artifacts

Here the indigo-dc.galaxycloud-os is the ansible role entrusted of file system encryption:

```

artifacts:
  nfs_role:
    file: indigo-dc.nfs
    type: toska.artifacts.AnsibleGalaxy.role
  galaxy_os_role:
    file: indigo-dc.galaxycloud-os
    type: toska.artifacts.AnsibleGalaxy.role
  galaxy_role:
    file: mtangaro.galaxycloud
    type: toska.artifacts.AnsibleGalaxy.role

```

interfaces

The Ansible role is called with its input parameters:

```

interfaces:
  Standard:
    configure:
      implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/v3.0.1/
↪artifacts/galaxy/galaxy_os_install.yml
    inputs:
      storage_encryption: { get_property: [ SELF, storage_encryption ] }
      vault_url: { get_property: [ SELF, vault_url ] }
      vault_wrapping_token: { get_property: [ SELF, vault_wrapping_token ] }
      vault_secret_path: { get_property: [ SELF, vault_secret_path ] }
      vault_secret_key: { get_property: [ SELF, vault_secret_key ] }
      wn_ips: { get_property: [ SELF, wn_ips ] }
      galaxy_install_path: { get_property: [ SELF, install_path ] }
      galaxy_user: { get_property: [ SELF, user ] }
      galaxy_admin: { get_property: [ SELF, admin_email ] }
      galaxy_admin_api_key: { get_property: [ SELF, admin_api_key ] }
      galaxy_lrms: { get_property: [ SELF, lrms, type ] }
      galaxy_version: { get_property: [ SELF, version ] }
      galaxy_instance_description: { get_property: [ SELF, instance_description ] }
      galaxy_instance_key_pub: { get_property: [ SELF, instance_key_pub ] }
      export_dir: { get_property: [ SELF, export_dir ] }

```

(continues on next page)

(continued from previous page)

```
galaxy_flavor: { get_property: [ SELF, flavor ] }
get_refdata: { get_property: [ SELF, reference_data ] }
```

The artifact includes indigo-dc.galaxycloud-os and indigo-dc.galaxycloud call.

```
---
- hosts: localhost
  connection: local
  roles:
    - role: indigo-dc.galaxycloud-os
      GALAXY_ADMIN_EMAIL: "{{ galaxy_admin }}"

    - role: indigo-dc.galaxycloud
      GALAXY_VERSION: "{{ galaxy_version }}"
      GALAXY_ADMIN_EMAIL: "{{ galaxy_admin }}"
      GALAXY_ADMIN_API_KEY: "{{ galaxy_admin_api_key }}"
      enable_storage_advanced_options: true # true only with indigo-dc.galaxycloud-os
```

Note: The option `enable_storage_advanced_options` has to be set to `true`, leaving storage configuration to indigo-dc.galaxycloud-os.

22.1.3 GalaxyShedTool

This custom type is used to install tools on Galaxy.

```
tosca.nodes.indigo.GalaxyShedTool:
  derived_from: tosca.nodes.WebApplication
```

properties

The inputs needed to install tools on Galaxy are:

```
properties:
  flavor:
    type: string
    description: name of the Galaxy flavor
    required: true
    default: galaxy-no-tools
  admin_api_key:
    type: string
    description: key to access the API with admin role
    default: not_very_secret_api_key
    required: false
  version:
    type: string
    description: galaxy version installed
    default: master
    required: false
  reference_data:
    type: boolean
    description: Install Reference data
```

(continues on next page)

(continued from previous page)

```
default: true
required: false
```

requirements

This custom types requires to be run on a Host with Galaxy already installed before tools installation.

```
requirements:
- host:
    capability: tosca.capabilities.Container
    node: tosca.nodes.indigo.GalaxyPortal
    relationship: tosca.relationships.HostedOn
```

Then the Indigo-dc.galaxy-tools role is installed:

```
artifacts:
  galaxy_role:
    file: indigo-dc.galaxy-tools, master
    type: tosca.artifacts.AnsibleGalaxy.role
```

interfaces

Finally, ansible is called:

```
interfaces:
  Standard:
    configure:
      implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/v3.0.1/
↳ artifacts/galaxy/galaxy_tools_configure.yml
    inputs:
      galaxy_flavor: { get_property: [ SELF, flavor ] }
      galaxy_admin_api_key: { get_property: [ HOST, admin_api_key ] }
      galaxy_version: { get_property: [ SELF, version ] }
      get_refdata: { get_property: [ SELF, reference_data ] }
```

to install tools:

```
---
- hosts: localhost
  connection: local
  roles:
    - { role: indigo-dc.galaxytools, GALAXY_VERSION: '{{ galaxy_version }}',
↳ when: galaxy_flavor != 'galaxy-no-tools' }
```

22.1.4 GalaxyReferenceData

The ReferenceData custom type configure Galaxy to retrieve the reference data from a CernVM-FS repository.

```
tosca.nodes.indigo.GalaxyReferenceData:
  derived_from: tosca.nodes.WebApplication
```

properties

The ReferenceData input parameters are:

```
properties:
  reference_data:
    type: boolean
    description: Install Reference data
    default: true
    required: true
  refdata_cvmfs_configuration:
    type: string
    description: Configure cvmfs or load preconfigured repository
    default: 'cvmfs_preconfigured'
    required: false
  refdata_cvmfs_repository_name:
    type: string
    description: CernVM-FS repository name
    default: 'elixir-italy.galaxy.refdata'
    required: false
  refdata_cvmfs_server_url:
    type: string
    description: CernVM-FS server, replica or stratum-zero
    default: 'server_url'
    required: false
  refdata_cvmfs_key_file:
    type: string
    description: CernVM-FS public key
    default: 'not_a_key'
    required: false
  refdata_cvmfs_proxy_url:
    type: string
    description: CernVM-FS proxy url
    default: 'DIRECT'
    required: false
  refdata_cvmfs_proxy_port:
    type: integer
    description: CernVM-FS proxy port
    default: 80
    required: false
  refdata_dir:
    type: string
    description: path to store galaxy reference data
    default: /cvmfs
    required: false
  flavor:
    type: string
    description: name of the Galaxy flavor
    required: true
    default: galaxy-no-tools
```

If `refdata_cvmfs_configuration` is set to `cvmfs` all the parameters are required to setup the CVMFS repository.

On the contrary, if `refdata_cvmfs_configuration` is set to `cvmfs_preconfigured` only `refdata_cvmfs_repository_name`, i.e. the name of the repository is needed, since all the needed parameters are retrieved from [GitHub](#).

requirements

Also in this case, Galaxy is required to install and configure reference data:

```
requirements:
- host:
    capability: tosca.capabilities.Container
    node: tosca.nodes.indigo.GalaxyPortal
    relationship: tosca.relationships.HostedOn
```

artifacts

The role is used to install cvmfs client.

```
artifacts:
  cvmfs_role:
    file: indigo-dc.cvmfs-client
    type: tosca.artifacts.AnsibleGalaxy.role
  galaxy_role:
    file: indigo-dc.galaxycloud-refdata
    type: tosca.artifacts.AnsibleGalaxy.role
```

interfaces

The Ansible role is called with the parameters:

```
interfaces:
  Standard:
    configure:
      implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/v3.0.1/
        ↪artifacts/galaxy/galaxy_refdata_configure.yml
      inputs:
        get_refdata: { get_property: [ SELF, reference_data ] }
        refdata_cvmfs_configuration: { get_property: [ SELF, refdata_cvmfs_
        ↪configuration ] }
        refdata_cvmfs_repository_name: { get_property: [ SELF, refdata_cvmfs_
        ↪repository_name ] }
        refdata_cvmfs_server_url: { get_property: [ SELF, refdata_cvmfs_server_url ] }
        refdata_cvmfs_key_file: { get_property: [ SELF, refdata_cvmfs_key_file ] }
        refdata_cvmfs_proxy_url: { get_property: [ SELF, refdata_cvmfs_proxy_url ] }
        refdata_cvmfs_proxy_port: { get_property: [ SELF, refdata_cvmfs_proxy_port ] }
        refdata_dir: { get_property: [ SELF, refdata_dir ] }
        galaxy_flavor: { get_property: [ SELF, flavor ] }
```

The role download from the [GitHub](#) repository all needed information to mount the CVMFS repository:

```
---
- hosts: localhost
  connection: local
  pre_tasks:
    - set_fact:
        galaxy_flavor: 'galaxy-no-tools'
        when: galaxy_flavor == 'galaxy-minimal'
    - name: Get reference data cvmfs key for on-the-fly configuration
      get_url:
```

(continues on next page)

(continued from previous page)

```

    url: 'https://raw.githubusercontent.com/indigo-dc/Reference-data-galaxycloud-
↪repository/master/cvmfs_server_keys/{{ refdata_cvmfs_key_file }}'
    dest: '/tmp'
    when: refdata_cvmfs_configuration == 'cvmfs'
  - name: Get reference data cvmfs key for preconfigured repository
    get_url:
      url: 'https://raw.githubusercontent.com/indigo-dc/Reference-data-galaxycloud-
↪repository/master/cvmfs_server_keys/{{ refdata_cvmfs_repository_name }}.pub'
      dest: '/tmp'
      when: refdata_cvmfs_configuration == 'cvmfs_preconfigured'
  - name: Get reference data cvmfs configuration for preconfigured repository
    get_url:
      url: 'https://raw.githubusercontent.com/indigo-dc/Reference-data-galaxycloud-
↪repository/master/cvmfs_server_config_files/{{ refdata_cvmfs_repository_name }}.conf
↪'
      dest: '/tmp'
      when: refdata_cvmfs_configuration == 'cvmfs_preconfigured'
  roles:
    - role: indigo-dc.galaxycloud-refdata

```

22.1.5 GalaxyPortalDocker

The role to deploy the Galaxy Official Docker is derived again from the GalaxyPortalAndStorage, allowing to configure the same options and to perform, also, the storage encryption.

```

tosca.nodes.indigo.GalaxyPortalDocker:
  derived_from: tosca.nodes.indigo.GalaxyPortalAndStorage

```

properties

The reference data are automatically configured, using CVMFS. Therefore the repository name is needed between the inputs.

```

properties:
  refdata_cvmfs_repository_name:
    type: string
    description: CernVM-FS repository name
    default: 'elixir-italy.galaxy.refdata'
    required: false

```

artifacts

The Docker engine has to be installed, alongside with the role to configure the Docker and the storage encryption.

```

artifacts:
  nfs_role:
    file: indigo-dc.nfs
    type: tosca.artifacts.AnsibleGalaxy.role
  galaxy_os_role:
    file: indigo-dc.galaxycloud-os
    type: tosca.artifacts.AnsibleGalaxy.role
  docker_role:

```

(continues on next page)

(continued from previous page)

```

file: indigo-dc.docker
type: tosa.artifacts.AnsibleGalaxy.role
galaxy_role_docker:
file: indigo-dc.galaxycloud_docker
type: tosa.artifacts.AnsibleGalaxy.role

```

interfaces

The Ansible role is called with the paramteres:

```

interfaces:
  Standard:
    configure:
      implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/v3.0.1/
↪artifacts/galaxy/galaxy_docker.yml
    inputs:
      storage_encryption: { get_property: [ SELF, storage_encryption ] }
      vault_url: { get_property: [ SELF, vault_url ] }
      vault_wrapping_token: { get_property: [ SELF, vault_wrapping_token ] }
      vault_secret_path: { get_property: [ SELF, vault_secret_path ] }
      vault_secret_key: { get_property: [ SELF, vault_secret_key ] }
      galaxy_install_path: { get_property: [ SELF, install_path ] }
      galaxy_user: { get_property: [ SELF, user ] }
      galaxy_admin: { get_property: [ SELF, admin_email ] }
      galaxy_admin_api_key: { get_property: [ SELF, admin_api_key ] }
      galaxy_lrms: { get_property: [ SELF, lrms, type ] }
      galaxy_version: { get_property: [ SELF, version ] }
      galaxy_instance_description: { get_property: [ SELF, instance_description ] }
      galaxy_instance_key_pub: { get_property: [ SELF, instance_key_pub ] }
      export_dir: { get_property: [ SELF, export_dir ] }
      galaxy_flavor: { get_property: [ SELF, flavor ] }
      get_refdata: { get_property: [ SELF, reference_data ] }
      refdata_cvmfs_repository_name: { get_property: [ SELF, refdata_cvmfs_
↪repository_name ] }

```

Finally, the galaxycloud_docker ansible role download and run the Galaxy Docker image.

```

---
- hosts: localhost
  connection: local
  roles:
    - role: indigo-dc.galaxycloud-os
      GALAXY_ADMIN_EMAIL: "{{ galaxy_admin }}"
      application_virtualization_type: 'docker'
      enable_reboot_scripts: false
      enable_customization_scripts: false

    - role: indigo-dc.galaxycloud_docker
      GALAXY_VERSION: "{{ galaxy_version }}"
      GALAXY_ADMIN_EMAIL: "{{ galaxy_admin }}"
      GALAXY_ADMIN_API_KEY: "{{ galaxy_admin_api_key }}"

```

22.2 Galaxy template

The orchestrator interprets the TOSCA template and orchestrate the Galaxy deployment on the virtual machine.

Galaxy template is located [here](#).

Input parameters are needed for each custom type used in the template:

- Virtual hardware parameters:

```
number_cpus:
  type: integer
  description: number of cpus required for the instance
  default: 1
memory_size:
  type: string
  description: ram memory required for the instance
  default: 1 GB
storage_size:
  type: string
  description: storage memory required for the instance
  default: 10 GB
```

- Galaxy input paramters:

```
admin_email:
  type: string
  description: email of the admin user
  default: admin@admin.com
admin_api_key:
  type: string
  description: key to access the API with admin role
  default: not_very_secret_api_key
user:
  type: string
  description: username to launch the galaxy daemon
  default: galaxy
version:
  type: string
  description: galaxy version to install
  default: master
instance_description:
  type: string
  description: galaxy instance description
  default: "INDIGO Galaxy test"
instance_key_pub:
  type: string
  description: galaxy instance ssh public key
  default: your_ssh_public_key
export_dir:
  type: string
  description: path to store galaxy data
  default: /export
```

- Storage input parameters:

```
galaxy_storage_type:
  type: string
```

(continues on next page)

(continued from previous page)

```

    description: Storage type (Iaas Block Storage, Onedaata, Filesystem encryption)
    default: "IaaS"
userdata_provider:
    type: string
    description: default OneProvider
    default: "not_a_privder_url"
userdata_token:
    type: string
    description: Access token for onedata space
    default: "not_a_token"
userdata_space:
    type: string
    description: Onedata space
    default: "galaxy"

```

- Galaxy flavor input parameters:

```

flavor:
    type: string
    description: Galaxy flavor for tools installation
    default: "galaxy-no-tools"

```

- Reference data input parameters, for all possible options (CernVM-FS, Onedata and download).

```

reference_data:
    type: boolean
    description: Install Reference data
    default: true
refdata_dir:
    type: string
    description: path to store galaxy reference data
    default: /refdata
refdata_repository_name:
    type: string
    description: Onedata space name, CernVM-FS repository name or subdirectory
↳download name
    default: 'elixir-italy.galaxy.refdata'
refdata_provider_type:
    type: string
    description: Select Reference data provider type (Onedata, CernVM-FS or
↳download)
    default: 'onedata'
refdata_provider:
    type: string
    description: Oneprovider for reference data
    default: 'not_a_provider'
refdata_token:
    type: string
    description: Access token for reference data
    default: 'not_a_token'
refdata_cvmfs_server_url:
    type: string
    description: CernVM-FS server, replica or stratum-zero
    default: 'server_url'
refdata_cvmfs_repository_name:
    type: string
    description: Reference data CernVM-FS repository name

```

(continues on next page)

(continued from previous page)

```

    default: 'not_a_cvmfs_repository_name'
refdata_cvmfs_key_file:
    type: string
    description: CernVM-FS public key
    default: 'not_a_key'
refdata_cvmfs_proxy_url:
    type: string
    description: CernVM-FS proxy url
    default: 'DIRECT'
refdata_cvmfs_proxy_port:
    type: integer
    description: CernVM-FS proxy port
    default: 80

```

Input parameters are passed to the corresponding ansible roles, through custom type call:

```

galaxy:
    type: tosca.nodes.indigo.GalaxyPortalAndStorage
    properties:
        os_storage: { get_input: galaxy_storage_type }
        token: { get_input: userdata_token }
        provider: { get_input: userdata_provider }
        space: { get_input: userdata_space }
        admin_email: { get_input: admin_email }
        admin_api_key: { get_input: admin_api_key }
        version: { get_input: version }
        instance_description: { get_input: instance_description }
        instance_key_pub: { get_input: instance_key_pub }
        export_dir: { get_input: export_dir }
    requirements:
        - lrms: local_lrms

galaxy_tools:
    type: tosca.nodes.indigo.GalaxyShedTool
    properties:
        flavor: { get_input: flavor }
        admin_api_key: { get_input: admin_api_key }
    requirements:
        - host: galaxy

galaxy_refdata:
    type: tosca.nodes.indigo.GalaxyReferenceData
    properties:
        reference_data: { get_input: reference_data }
        refdata_dir: { get_input: refdata_dir }
        flavor: { get_input: flavor }
        refdata_repository_name: { get_input: refdata_repository_name }
        refdata_provider_type: { get_input: refdata_provider_type }
        refdata_provider: { get_input: refdata_provider }
        refdata_token: { get_input: refdata_token }
        refdata_cvmfs_server_url: { get_input: refdata_cvmfs_server_url }
        refdata_cvmfs_repository_name: { get_input: refdata_cvmfs_repository_name }
        refdata_cvmfs_key_file: { get_input: refdata_cvmfs_key_file }
        refdata_cvmfs_proxy_url: { get_input: refdata_cvmfs_proxy_url }
        refdata_cvmfs_proxy_port: { get_input: refdata_cvmfs_proxy_port }
    requirements:
        - host: galaxy

```

(continues on next page)

(continued from previous page)

```
- dependency: galaxy_tools
```

Note: Note that Reference data custom type needs Galaxy installed to the ost host : galaxy, but depends on galaxy tools dependency: galaxy_tools since it has to check installed and missing tools.

Finally we have virtual hardware customization:

```
host:
  properties:
    num_cpus: { get_input: number_cpus }
    mem_size: { get_input: memory_size }
```

Image selection:

```
os:
  properties:
    type: linux
    distribution: centos
    version: 7.2
    image: indigodatacloudapps/galaxy
```

And Storage configuration, which takes the export_dir input for the mount point and storage_size input allowing for storage size customization.

```
- local_storage:
  # capability is provided by Compute Node Type
  node: my_block_storage
  capability: tosca.capabilities.Attachment
  relationship:
    type: tosca.relationships.AttachesTo
  properties:
    location: { get_input: export_dir }
    device: hdb
```

```
my_block_storage:
  type: tosca.nodes.BlockStorage
  properties:
    size: { get_input: storage_size }
```

22.3 Galaxy cluster template

The ansible_galaxycloud role provides the possibility to instantiate Galaxy with SLURM as Resource Manager, just setting the galaxy_lrms variable to slurm.

This allows to instantiate Galaxy with SLURM cluster exploiting INDIGO custom types and ansible roles using INDIGO components:

- CLUES (INDIGO solution for automatic elasticity)
- Master node deployment with SLURM (ansible recipes + tosca types)
- Install Galaxy + SLURM support (already in our ansible role indigo-dc.galaxycloud)
- Worker node deployment

- Galaxy customization for worker nodes

The related toasca template is located [here](#).

The input parameters allow to customize the number of virtual nodes, nodes and master virtual hardware:

```
wn_num:
  type: integer
  description: Maximum number of WNs in the elastic cluster
  default: 5
  required: yes
fe_cpus:
  type: integer
  description: Numer of CPUs for the front-end node
  default: 1
  required: yes
fe_mem:
  type: scalar-unit.size
  description: Amount of Memory for the front-end node
  default: 1 GB
  required: yes
wn_cpus:
  type: integer
  description: Numer of CPUs for the WNs
  default: 1
  required: yes
wn_mem:
  type: scalar-unit.size
  description: Amount of Memory for the WNs
  default: 1 GB
  required: yes
```

Note: You can refer to *Galaxy template* section for galaxy input parameters.

The master node hosts Galaxy and Slurm controller:

```
elastic_cluster_front_end:
  type: toasca.nodes.indigo.ElasticCluster
  properties:
    deployment_id: orchestrator_deployment_id
    iam_access_token: iam_access_token
    iam_clues_client_id: iam_clues_client_id
    iam_clues_client_secret: iam_clues_client_secret
  requirements:
    - lrms: lrms_front_end
    - wn: wn_node

galaxy_portal:
  type: toasca.nodes.indigo.GalaxyPortal
  properties:
    admin_email: { get_input: admin_email }
    admin_api_key: { get_input: admin_api_key }
    version: { get_input: version }
    instance_description: { get_input: instance_description }
    instance_key_pub: { get_input: instance_key_pub }
  requirements:
    - lrms: lrms_front_end
```

(continues on next page)

(continued from previous page)

```
lrms_front_end:
  type: tosca.nodes.indigo.LRMS.FrontEnd.Slurm
  properties:
    wn_ips: { get_attribute: [ lrms_wn, private_address ] }
  requirements:
    - host: lrms_server

lrms_server:
  type: tosca.nodes.indigo.Compute
  capabilities:
    endpoint:
      properties:
        dns_name: slurmsserver
        network_name: PUBLIC
      ports:
        http_port:
          protocol: tcp
          source: 80
  host:
    properties:
      num_cpus: { get_input: fe_cpus }
      mem_size: { get_input: fe_mem }
  os:
    properties:
      image: linux-ubuntu-14.04-vmi
```

Then the worker nodes configuration (OS and virtual hardware):

```
wn_node:
  type: tosca.nodes.indigo.LRMS.WorkerNode.Slurm
  properties:
    front_end_ip: { get_attribute: [ lrms_server, private_address, 0 ] }
  capabilities:
    wn:
      properties:
        max_instances: { get_input: wn_num }
        min_instances: 0
  requirements:
    - host: lrms_wn

galaxy_wn:
  type: tosca.nodes.indigo.GalaxyWN
  requirements:
    - host: lrms_wn

lrms_wn:
  type: tosca.nodes.indigo.Compute
  capabilities:
    scalable:
      properties:
        count: 0
  host:
    properties:
      num_cpus: { get_input: wn_cpus }
      mem_size: { get_input: wn_mem }
  os:
```

(continues on next page)

(continued from previous page)

```
properties:
  image: linux-ubuntu-14.04-vmi
```

Note: Note that to orchestrate Galaxy with SLURM we do not need new TOSCA custom types or ansible roles. Everything is already built in INDIGO.

Build CVMFS server for reference data

This section gives a quick overview of the steps needed to create a new `cvmfs` repository to share reference data and activate it on the clients. The repository name used is `elixir-italy.galaxy.refdata`, but it can be replaced with the appropriate name.

All script needed to deploy a Reference data CernVM-FS Stratum 0 are located [here](#).

23.1 Create CernVM-FS Repository

The CernVM-FS (`cvmfs`) relies on OverlayFS or AUFS as default storage driver. Ubuntu 16.04 natively supports OverlayFS, therefore it is used as default, to create and populate the `cvmfs` server.

1. Install `cvmfs` and `cvmfs-server` packages.
2. Ensure enough disk space in `/var/spool/cvmfs` (>50GiB).
3. For local storage: ensure enough disk space in `/srv/cvmfs`.
4. Create a repository with `cvmfs_server mkfs`.

Warning:

- `/cvmfs` is the repository mount point, containing read-only union file system mountpoints that become writable during repository updates.
- `/var/spool/cvmfs` hosts the scratch area described here, thus might consume notable disk space during repository updates. When you copy your files to `/cvmfs/<your_repository_name>/`, they are stored in `/var/spool/cvmfs`, therefore you have ensure enough space to this directory.
- `/srv/cvmfs` is the central repository storage location. During the `cvmfs_server publish` procedure, your files will be moved and stored here. Therefore you have to ensure enough space here, too. This directory needs to have enough space to store all your `cvmfs` server contents.

Note: A complete set of reference data takes 100 GB. Our cvmfs server exploits two different volumes, one 100 GB volume mounted on `/var/spool/cvmfs` and one 200 GB volume for `/srv/cvmfs`.

- To **Create** a new repository:

```
cvmfs_server mkfs -w http://<stratum_zero>/cvmfs/elixir-italy.galaxy.refdata -o 
↪cvmfs elixir-italy.galaxy.refdata'
```

Replace `<stratum_zero>` with your domain or ip address.

- **Publish** your contents to the cvfms stratum zero server:

```
cvmfs_server transaction elixir-italy.galaxy.refdata
touch /cvmfs/elixir-italy.galaxy-refdata/test-content
cvmfs_server publish elixir-italy.galaxy.refdata
```

- **Periodically** resign the repository (at least every 30 days):

```
cvmfs_server resign elixir-italy.galaxy.refdata
```

A resign script is located in `/usr/local/bin/Cvmfs-stratum0-resign` and the corresponding weekly cron job is set to `/etc/cron.d/cvmfs_server_resign`.

Log file is located in `/var/log/Cvmfs-stratum0-resign.log`.

- Finally **restart** the apache2 daemon.

```
sudo systemctl restart apache2
```

The public key of the new repository is located in `/etc/cvmfs/keys/elixir-italy.galaxy.refdata.pub`

23.2 Client configuration

- Add the public key of the new repository to `/etc/cvmfs/keys/elixir-italy.galaxy.refdata.pub`
- Repository configuration:

```
$ cat /etc/cvmfs/config.d/elixir-italy.galaxy.refdata.conf
CVMFS_SERVER_URL=http://90.147.102.186/cvmfs/elixir-italy.galaxy.refdata
CVMFS_PUBLIC_KEY=/etc/cvmfs/keys/elixir-italy.galaxy.refdata.pub
CVMFS_HTTP_PROXY=DIRECT
```

23.3 Populate a CernVM-FS Repository (with reference data)

Content Publishing

1. `cvmfs_server transaction <repository name>`
2. Install content into `/cvmfs/<repository name>` (see [Reference data download](#) section)
3. `cvmfs_server publish <repository name>`

Note: `cvmfs_server publish` command will take time to move your contents from `/cvmfs` to `/srv/cvmfs`.

23.4 Reference data download

Reference data are available on Openstack Swift for public download. The list of reference data download link is [here](#)

Furthermore, to automatically download our reference data set it is possible to use python script `refdata_download.py`.

The package `python-pycurl` is needed to satisfy `refdata_download.py` requirements: on Ubuntu `sudo apt-get install python-pycurl`

23.4.1 Script usage

This script takes the `yml` files as input located in `Reference-data-galaxycloud-repository/lists/` directory.

Option	Description
<code>-i,</code> <code>--input .</code>	Input genome list in <code>yml</code> format
<code>-o,</code> <code>--outdir</code>	Destination directory. Default <code>/refdata</code>
<code>-s, --space</code>	Subdirectory name (for <code>cvmfs</code> and <code>onedata</code> spaces). Default <code>elixir-italy.galaxy.refdata</code>

```
/usr/bin/python refdata_download.py -i sacCer3-list.yml -o /refdata -s elixir-italy.
↳ galaxy.refdata
```

Available Reference data `yml` file:

- `at10-list.yml`
- `at9-list.yml`
- `dm2-list.yml`
- `dm3-list.yml`
- `hg18-list.yml`
- `hg19-list.yml`
- `hg38-list.yml`
- `mm10-list.yml`
- `mm8-list.yml`
- `mm9-list.yml`
- `sacCer1-list.yml`
- `sacCer2-list.yml`
- `sacCer3-list.yml`

It is possible to download automatically all reference data files using the bash script `refdata_download.sh`, which parse the python script, using as input the list file `Reference-data-galaxycloud-repository/lists/list.txt`

```
./refdata_download.sh list.txt
```

23.5 References

[CernVM-FS stratum Zero documentation](#)

[Nikhef wiki](#)

Vault configuration

Hashicorp Vault is a tool for securely accessing “secrets” and is exploited on Laniakea to store and manage user encryption passphrases.

A secret is everything you want to tightly control access to, such as encryption passphrases. Data stored on Vault are encrypted with 256 bit AES (Advanced Encryption Standard) cipher in the Galois Counter Mode (GCM) with a randomly generated nonce.

Laniakea by default exploits `kv-v2` secrets engine to store secrets within the configured physical storage for Vault.

24.1 Vault main concepts

1. Paths: everything in Vault is path based: users are able to write their secrets on a specific path, depending on their Identity.
2. Tokens are the core method for authentication within Vault. After the authentication on the Laniakea Dashboard, tokens are dynamically generated based on user identity.
3. Policies provide a declarative way to grant or forbid access to certain path and operations, controlling what the token holder is allowed to do within Vault.

A token generated with a specific policy allows to write/read/update a secret in a specific path.

24.2 Vault authentication and authorization flow

Laniakea exploits a set of four different policies for secrets management:

1. The first policy needed is named `kv-2` and is used to issue new tokens and grant permissions on the Vault UI.

```
# Manage tokens
path "auth/token/*" {
  capabilities = [ "create", "read", "update", "delete", "sudo" ]
}
```

(continues on next page)

(continued from previous page)

```
# Grant permissions on user specific path
path "secrets/data/{{identity.entity.aliases.<jwt_auth_accessor>.name}}/*" {
  capabilities = [ "read" ]
}

# For Web UI usage
path "secrets/metadata" {
  capabilities = ["list"]
}
```

- The **write_only.hcl** token is exploited by LUKS script on the Virtual machine during the encryption procedure to store passphrases on Vault.

```
# Grant permissions on user specific path
path "secrets/data/{{identity.entity.aliases.<jwt_auth_accessor>.name}}/*" {
  capabilities = [ "create" ]
}
```

The encryption script write the random generated passphrase on vault, in a path where only the user can access, since it depends on its identity.

- The Laniakea Dashboard can Read, if required by the user, after the authentication, the passphrase from Vault using the **read_only.hcl** policy.

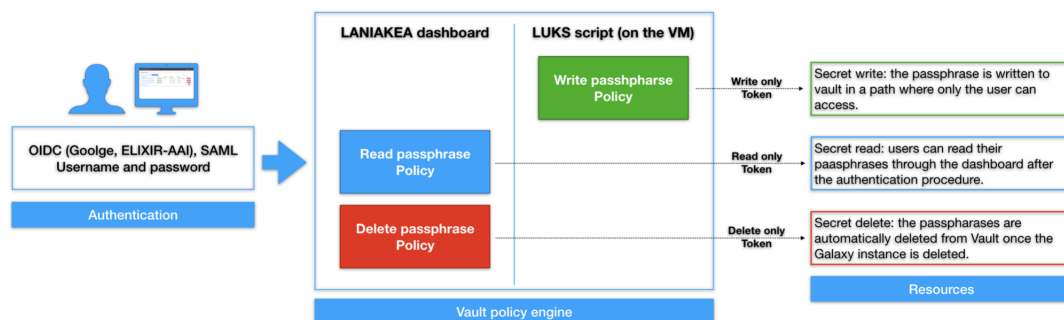
```
# Grant permissions on user specific path
path "secrets/data/{{identity.entity.aliases.<jwt_auth_accessor>.name}}/*" {
  capabilities = [ "read" ]
}
```

Users can read their passphrases through the dashboard after authenticating.

- Finally, the Laniakea Dashboard Deletes the passphrase from Vault, once the deployment is deleted using the **delete_only.hcl** policy.

```
# Permanently remove all versions and metadata for a key
path "secrets/metadata/{{identity.entity.aliases.auth_jwt_9144d398.name}}/*" {
  capabilities = ["delete"]
}
```

The passphrases are automatically deleted from Vault once the Galaxy instance is deleted.



24.3 Vault passphrase storage flow

On the Dashboard:

1. The dashboard exploits the JWT token (from IAM) to get Vault token using the `kv-2` policy. This token should not be revoked until the write procedure is finished, otherwise also the children token are revoked.
2. The vault token is used to get a **wrapping token** :
 - with `write_only` policy, i.e. the token can only write (not update) a new secret on vault.
 - it can be used only one time.
 - limited in time duration (currently configured to expire after 12 hours).

The wrapping token is sent to the VM, via TOSCA template, with the vault path where the secret has to be stored. These information are sent to the VM, all needed to store a secret on vault using `kv-v2`:

- The path of the secret: `secrets/<user_subject>/<deployment_uuid>`. This allows to have **user identity** and **deployment uuid** dependent path for every secret
- wrapping token
- key name: the kv secret has key and its value. The value, i.e. the encryption passphrase, is automatically filled by luks script (it is randomly generated).

On the Virtual machine:

1. The ansible role on the VM run the fast-luks script to encrypt storage.
2. The (alphanumeric) passphrase is randomly generated.
3. The wrapping token is unwrapped, thus obtaining the privileged token with write (only) permissions to the secrets path.
4. The passphrase is written to `secrets/<user_subject>/<deployment_uuid>` path.
5. The token used to write the passphrase is revoked.

Finally, if required, the dashboard create a `read_only` token to show the passphrase to the user.

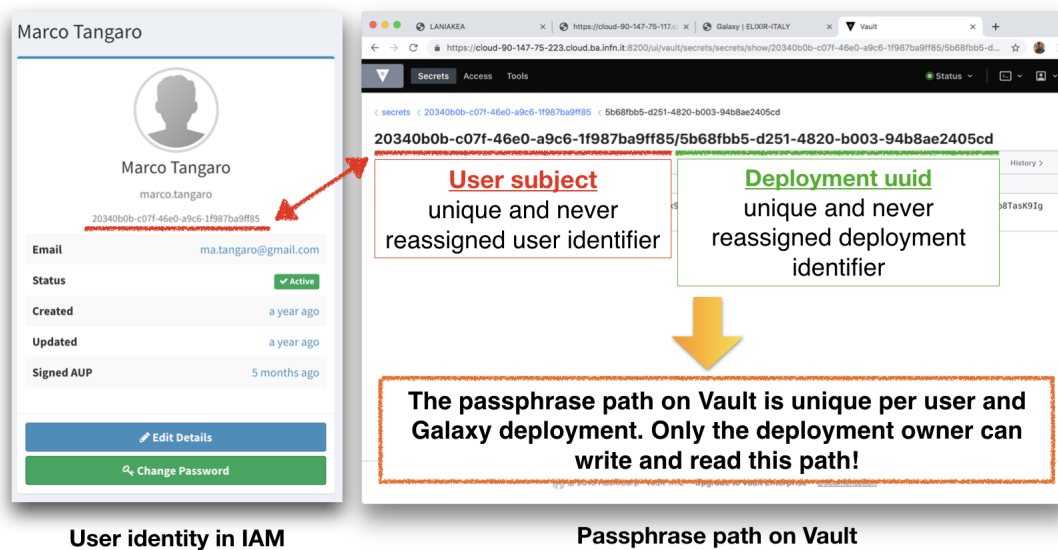
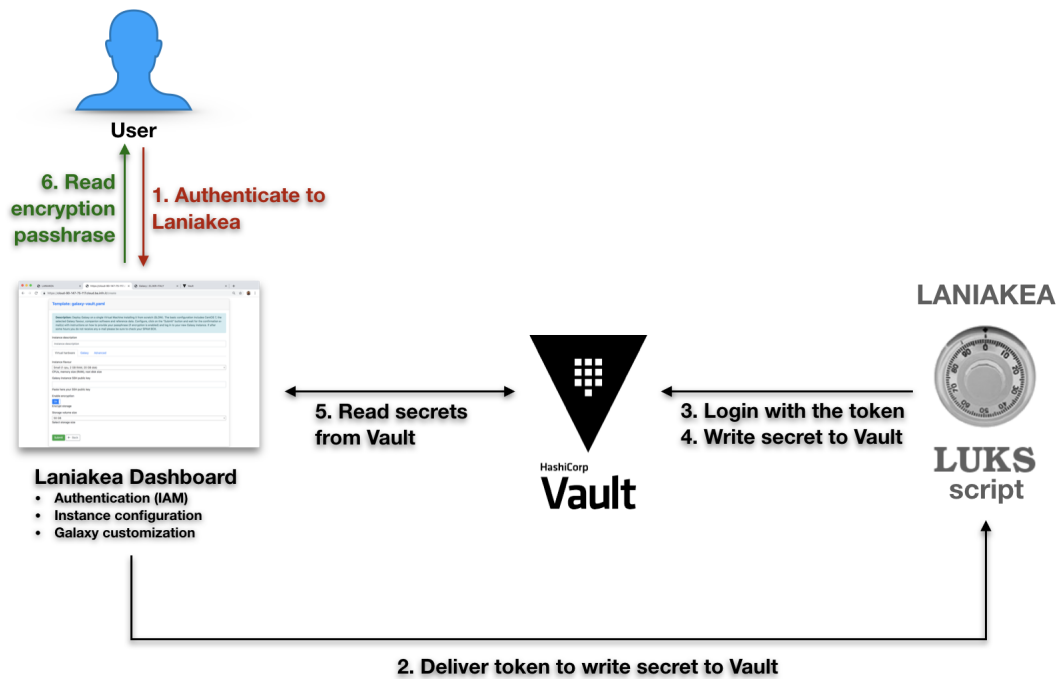
24.4 Passphrase path on Vault

Each passphrase is stored on vault on `/secrets` path. Each one depends on

1. User subject (issued by IAM): a unique and never reassigned user identifier

#. Deployment uuid (issued by the Dashboard): a unique and never reassigned deployment identifier.

This procedure results to have a passphrase path on Vault unique per user and Galaxy deployment. Only the deployment owner can write and read this path.



Laniakea Dashboard

The Laniakea Dashboard is the new, redesigned and reimplemented, user interface of Laniakea, developed using:

- [Flask](#) web micro-framework;
- [Jinja2](#) template engine;
- [Bootstrap 4](#) toolkit.

Lighter and more flexible than the previous interface, it has been integrated with Hashicorp Vault for user secrets management.

The Laniakea dashboard has, currently, two configuration files, in json format, which can be found in the `/etc/orchestrator-dashboard` directory: the `config.json` for the dashboard configuration and the `vault-config.json` specific for the Vault integration configuration.

Moreover, the TOSCA templates for each Laniakea application, with the corresponding parameters and metadata file can be found in `/opt/laniakea-dashboard-config`:

- `/opt/laniakea-dashboard-config/tosca-templates`: this directory collects the TOSCA templates of applications shown in the dashboard.
- `/opt/laniakea-dashboard-config/tosca-parameters`: this directory collects the parameters files corresponding to the TOSCA templates.
- `/opt/laniakea-dashboard-config/tosca-metadata`: this directory collects the metadata files corresponding to the TOSCA templates.

These paths can be configured in the `config.json` file.

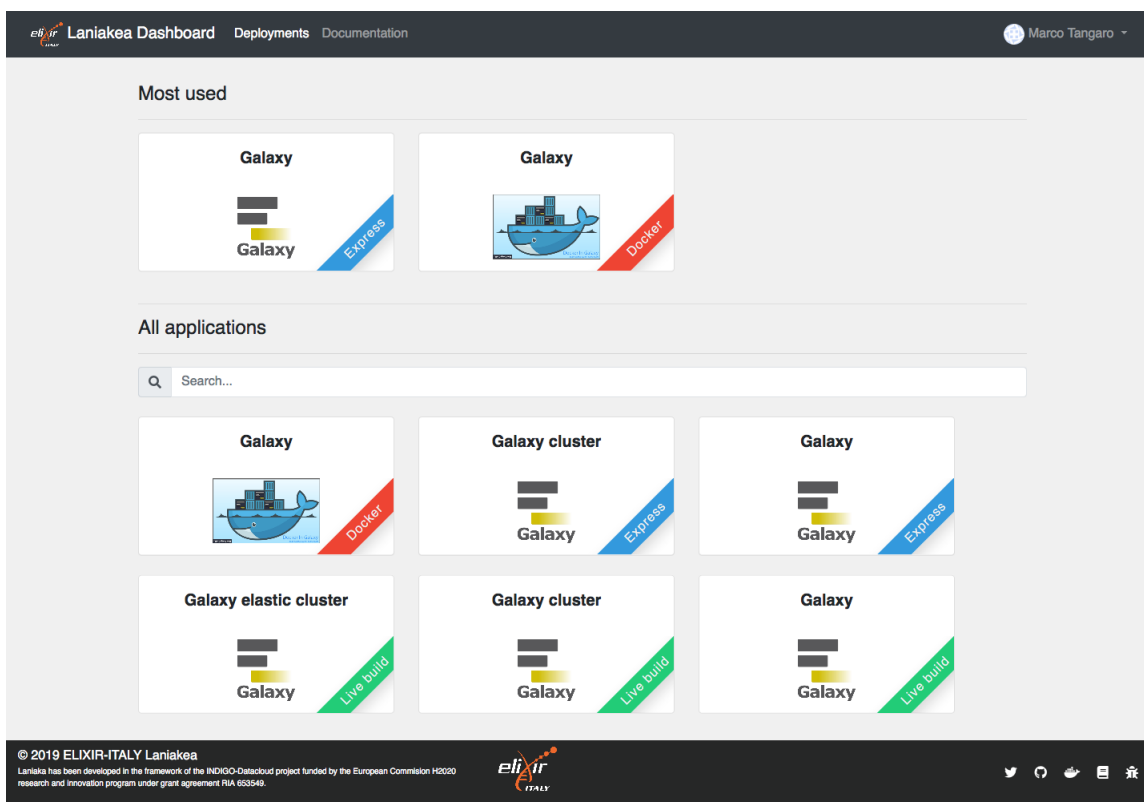
Warning: The Laniakea configuration files and templates are automatically configured by the installation procedure. Please modify them only if you know what you are doing!

25.1 Configuration

25.1.1 Overview

Home view

The home page tiles show the available applications. The goal of each tile is to quickly display each application, with its description and configuration button. Currently, the interface allows to pin three applications.



Deployments list

Each user can manage its instances. It is possible to view details, delete and access instances. Finally, using the menu in the action column, It is also possible to view logs and the template used for each instance.

Advanced options



If advanced options are enabled in the Dashboard configuration file, a new `Advanced` dropdown menu becomes available in the navbar,

showing available Service Level Agreement


and Dashboard settings.

Administration panel



For the Dashboard Administrator `Users` panel is available for advanced users management,

 Laniakea Dashboard [Deployments](#) [Advanced](#) [Users](#) [Documentation](#)  Marco Tangaro

My deployments







 Refresh [+ New deployment](#)



Show entries Search:

Instance name	Status	Creation time	Galaxy flavour	VM flavour	Endpoint	Actions
test	DELETE_IN_PROGRESS	2019-11-02 16:45:00	galaxy-minimal	medium	http://90.147.75.134/galaxy	 Delete
History test 2	CREATE_COMPLETE	2019-10-09 16:33:00	galaxy-epigen	medium	http://90.147.75.159/galaxy	 Delete


Showing 1 to 2 of 2 entries Previous **1** Next

© 2019 ELIXIR-ITALY Laniakea
Laniakea has been developed in the framework of the INDIGO-Datacloud project funded by the European Commission H2020 research and innovation program under grant agreement RIA 653549.



 Laniakea Dashboard [Deployments](#) [Advanced](#) [Users](#) [Documentation](#)  Marco Tangaro

Service Level Agreements







 Refresh


Show entries Search:


Site	Service Type	Start date	End date
RECAS-BARI	eu.egi.cloud.vm-management.openstack	2017-12-04T23:00:00.000Z	2018-08-31T22:00:00.000Z

Showing 1 to 1 of 1 entries Previous **1** Next

© 2019 ELIXIR-ITALY Laniakea
Laniakea has been developed in the framework of the INDIGO-Datacloud project funded by the European Commission H2020 research and innovation program under grant agreement RIA 653549.




Laniakea Dashboard
Deployments
Advanced ▾
Users
Documentation


Marco Tangaro ▾

Settings

Show
10 ▾
entries


Search:






Service	Endpoint
SLA Manager (SLAM)	https://cloud-90-147-170-175.cloud.ba.infn.it:8443/rest/slam
PaaS Orchestrator	https://cloud-90-147-75-155.cloud.ba.infn.it/orchestrator
Identity and Access Management (IAM)	https://iam.recas.ba.infn.it
Configuration Management DB (CMDB)	https://paas-orchestrator.cloud.ba.infn.it/cmdb
Infrastructure Manager (IM)	https://paas-orchestrator.cloud.ba.infn.it/im


Showing 1 to 5 of 5 entries


Previous
1
Next

© 2019 ELIXIR-ITALY Laniakea
Laniakea has been developed in the framework of the INDIGO-Datacloud project funded by the European Commission H2020 research and innovation program under grant agreement RIA 653549.




Laniakea Dashboard
Deployments
Advanced ▾
Users
Documentation


Marco Tangaro ▾

Galaxy

Description: Deploy Galaxy on a single Virtual Machine from a VM image (FAST). The basic configuration includes CentOS 7, the selected Galaxy flavour, companion software and reference data. Configure, click on the "Submit" button, wait for the confirmation e-mail(s) and log in to your new Galaxy instance. If after some hours you do not receive any e-mail please be sure to check your SPAM BOX.

Instance description

Instance description

Virtual hardware
Galaxy
Advanced

Configure scheduling:

☒ Auto
☐ Manual


☐ Do not delete the deployment in case of failure






☒ Send a confirmation email when complete

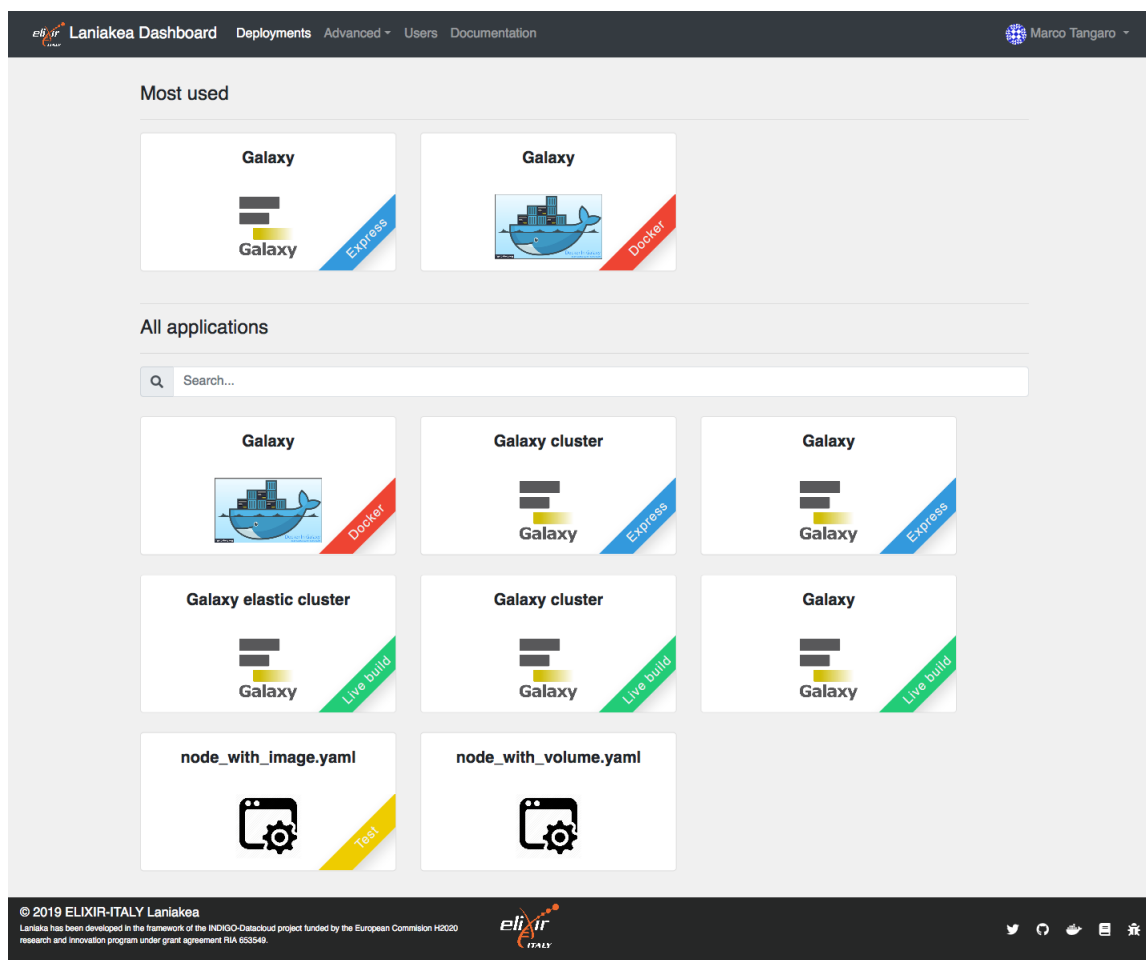
Submit

Cancel

© 2019 ELIXIR-ITALY Laniakea
Laniakea has been developed in the framework of the INDIGO-Datacloud project funded by the European Commission H2020 research and innovation program under grant agreement RIA 653549.





allowing to browse the Laniakea users,

Laniakea Dashboard Deployments Users Documentation Marco Tangaro

Users

Show 10 entries Search:

sub	Username	First Name	Last Name	Organisation	e-mail	Role		
0a17a932-afce-437c-b074-c00156a181d2	marco.tangaro	Marco	Tangaro	laniakea	ma.tangaro@ibiom.cnr.it	user		
5d13f594-b527-4f9f-80aa-af2a9c3b960f	ma.tangaro@gmail.com	Marco	Tangaro	laniakea	ma.tangaro@gmail.com	admin		
73f16d93-2441-4a50-88ff-85360d78c6b5	admin	Admin	User	laniakea	admin@iam.test	user		

Showing 1 to 3 of 3 entries Previous 1 Next

© 2019 ELIXIR-ITALY Laniakea
Laniakea has been developed in the framework of the INDIGO-Datacloud project funded by the European Commission H2020 research and innovation program under grant agreement RIA 653549.

user details:


and user deployment list. The Deployment details can be inspected. The cloud icon in the last icon shows if the deployment is connected to the INDIGO PaaS Orchestrator or not.


25.1.2 Basic configuration

The dashboard configuration file is located at `/etc/orchestrator-dashboard/config.json`, to make configuration changes.



```
{
  "IAM_CLIENT_ID": "my_client_id",
  "IAM_CLIENT_SECRET": "my_client_secret",
  "IAM_BASE_URL": "https://iam-test.indigo-datacloud.eu",
  "ORCHESTRATOR_URL": "https://indigo-paas.cloud.ba.infn.it/orchestrator",
  "SLAM_URL": "https://indigo-slam.cloud.ba.infn.it:8443",
  "CMDB_URL": "https://indigo-paas.cloud.ba.infn.it/cmdb",
  "IM_URL": "https://indigo-paas.cloud.ba.infn.it/im",
  "TOSCA_TEMPLATES_DIR": "/opt/tosca-templates",
  "TOSCA_PARAMETERS_DIR": "/opt/tosca-parameters",
  "TOSCA_METADATA_DIR": "/opt/tosca-metadata",
  "CALLBACK_URL": "https://my-orchestrator-dashboard.com/callback",
  "DB_HOST": "localhost",
  "DB_USER": "my-user",
  "DB_PASSWORD": "my-password",
  "DB_NAME": "orchestrator_dashboard",
  "DB_PORT": "3306",
  "MAIL_SERVER": "your.smtp.server.com",
  "MAIL_PORT": "25",
  "MAIL_SENDER": "test@orchestrator.com",
  "ADMINS": ["'admin@foo.it'", "'other_admin@test.it'"],
  "VAULT_URL": "https://my-vault-instance.com",
  "SUPPORT_EMAIL": "support@example.com",
  "EXTERNAL_LINKS": [ { "url": "https://indigo-paas.cloud.ba.infn.it/status-page",
    "menu_item_name": "Services status" } ],
  "menu_item_name": "Services status" } ],
```

(continues on next page)


Laniakea Dashboard
Deployments
Users
Documentation


Marco Tangaro

User data

 Update
 Back

sub

0a17a932-a1ce-437c-b074-c00156a181d2

First Name

Marco

Last Name

Tangaro

Username

marco.tangaro

e-mail

ma.tangaro@ibiom.cnr.it

Organisation

laniakea


Role






User


Active


Active

© 2019 ELIXIR-ITALY Laniakea
Laniakea has been developed in the framework of the INDIGO-Datacloud project funded by the European Commission H2020 research and innovation program under grant agreement RIA 653549.






Laniakea Dashboard
Deployments
Users
Documentation






















Marco Tangaro

Deployments for: Marco Tangaro

 Refresh
 Back

Show 10 entries


Search:






uuid	Status	Description	Created At	Updated At	Deployed At	Physical ID	Endpoint		
11e9fbc-86ca-3609-ac54-02420f5dce16	CREATE_COMPLETE	elastic cluster test	2019-10-31 16:36:00	2019-10-31 17:44:00	provider-RECAS-BARI	8e310778-fbfc-11e9-86dc-fa163eefe815	http://90.147.75.86/galaxy		
11e9fbed-6afe-43a7-ac54-02420f5dce16	DELETE_COMPLETE	elastic fixed	2019-10-31 14:47:00	2019-10-31 16:09:24	provider-RECAS-BARI	760090d8-fbed-11e9-aa4c-fa163eefe815	http://90.147.75.254/galaxy		
11e9fbe6-266f-c611-82e5-02420f5dce16	DELETE_COMPLETE	im v1.8.6.1 ans 2.3	2019-10-31 13:55:00	2019-10-31 14:12:59	provider-RECAS-BARI	31e83a10-fbe6-11e9-9442-fa163eefe815	http://90.147.75.199/galaxy		
11e9fbe2-c127-ea07-82e5-02420f5dce16	DELETE_COMPLETE	prova	2019-10-31 13:31:00	2019-10-31 13:46:48	provider-RECAS-BARI				
11e9fbde-338a-b2e9-82e5-02420f5dce16	DELETE_COMPLETE	im v1.8.6.1	2019-10-31 12:58:00	2019-10-31 13:32:12	provider-RECAS-BARI	39849a50-fbde-11e9-a5c8-fa163eefe815			
11e9fb15-9309-f90b-82e5-02420f5dce16	DELETE_COMPLETE	elastic cluster live	2019-10-30 13:02:00	2019-10-31 06:53:10	provider-RECAS-BARI	9fc319f0-fb15-11e9-ac2a-fa163eefe815	http://90.147.102.71/galaxy		
11e9fa64-a8e6-9d37-82e5-02420f5dce16	CREATE_FAILED	cluster express	2019-10-29 15:56:00	2019-10-29 15:56:00					
11e9fa64-b83e-3a24-82e5-02420f5dce16	DELETE_COMPLETE	elastic cluster live	2019-10-29 15:56:00	2019-10-30 13:03:21	provider-RECAS-BARI	bec7e40e-fa64-11e9-ac2a-fa163eefe815	http://90.147.170.217/galaxy		
11e9fa64-985c-2596-82e5-02420f5dce16	DELETE_COMPLETE	cluster live	2019-10-29 15:55:00	2019-10-30 13:03:51	provider-RECAS-BARI	a18e69f8-fa64-11e9-a613-fa163eefe815	http://90.147.170.203/galaxy		
11e9fa5b-4484-271c-a29e-02420f5dce16	DELETE_COMPLETE	centos 2	2019-10-29 14:49:00	2019-10-29 15:21:11	provider-RECAS-BARI	4d35f654-fa5b-11e9-ac2a-fa163eefe815			

Showing 1 to 10 of 132 entries

Previous
1
2
3
4
5
...
14
Next

© 2019 ELIXIR-ITALY Laniakea
Laniakea has been developed in the framework of the INDIGO-Datacloud project funded by the European Commission H2020 research and innovation program under grant agreement RIA 653549.



(continued from previous page)

```
"ENABLE_ADVANCED_MENU": "no",  
"LOG_LEVEL": "info"  
}
```

Configuration options

IAM_CLIENT_ID

Description: IAM client ID for the dashboard.

IAM_CLIENT_SECRET

Description: IAM client Secret for the dashaboard.

IAM_BASE_URL

Description: IAM url.

ORCHESTRATOR_URL

Description: Orchestrator url.

SLAM_URL

Description: SLAM url.

CMDB_URL

Description: CMDB url.

IM_URL

Description: IM url.

TOSCA_TEMPLATES_DIR

Description: Path of TOSCA tempaltes to be loaded.

Defaults: /opt/laniakea-dashboard-config/tosca-templates”,

TOSCA_PARAMETERS_DIR

Description: Path of TOSCA template parameters to create Dashboard configurable forms.

Defaults: /opt/laniakea-dashboard-config/tosca-parameters

TOSCA_METADATA_DIR

Description: Path of TOSCA template metadata with additional info (e.g. icon path).

Defaults: /opt/laniakea-dashboard-config/tosca-metadata

CALLBACK_URL

Description: Dashboard url for callback. Configure it as <dashboard url>/callback

Defaults: <https://my-orchestrator-dashboard.com/callback>

DB_HOST

Description: Database host. Configure it with the IP address of the Database host (do not leave localhost).

Defaults: localhost

DB_USER

Description: MySQL database user.

Defaults: orchestrator

DB_PASSWORD

Description: MySQL database password.

DB_NAME

Description: MySQL database name:

Defaults: orchestrator_dashboard

DB_PORT

Description: MySQL database port.

Defaults: 3306

MAIL_SERVER

Description: Mail server address allowing Dashboard notifications.

MAIL_PORT

Description: Mail server port.

Defaults: 25

MAIL_SENDER

Description: Mail sender of the notification mail.

Defaults: `Laniakea@elixir-italy.org`

ADMINS

Description: Dahsobard administrator users. Set this to a comma-separated list of valid Galaxy users (email addresses). These users will have access to the `Users` section of the dashboard.

VAULT_URL

Description: Vault url. This option enable vault support on Laniakea.

SUPPORT_EMAIL

Description: Support email, displayed on 500 error page.

Defaults: `laniakea.helpdesk@gmail.com`

EXTERNAL_LINKS

Description: create menu with external links, giving the url and the menu item name.

ENABLE_ADVANCED_MENU

Description: if yes, show advanced options in the navbar and the configurator form.

LOG_LEVEL

Description: Set log level.

Defaults: `info`

25.1.3 Vault configuration

The Vault support can be enabled editing the `/etc/orchestrator-dashboard/config.json` file, inserting the Vault url:

```
...  
"VAULT_URL": "https://<vault_host>:<vault_port>"
```

Vault fine tuning can be done through the `vault-config.json` file at `/etc/orchestrator-dashboard/vault-config.json`:

```
{
  "VAULT_BOUND_AUDIENCE": "orchestrator-dashboard",
  "VAULT_SECRETS_PATH": "secrets",
  "WRAPPING_TOKEN_TIME_DURATION": "1h",
  "READ_POLICY": "read_only",
  "READ_TOKEN_TIME_DURATION": "12h",
  "READ_TOKEN_RENEWAL_TIME_DURATION": "12h",
  "WRITE_POLICY": "write_only",
  "WRITE_TOKEN_TIME_DURATION": "12h",
  "WRITE_TOKEN_RENEWAL_TIME_DURATION": "12h",
  "DELETE_POLICY": "delete_only",
  "DELETE_TOKEN_TIME_DURATION": "12h",
  "DELETE_TOKEN_RENEWAL_TIME_DURATION": "12h"
}
```

Configuration options

VAULT_BOUND_AUDIENCE

Description: Vault is configured to exploits Json Web Token (JWT) for authentication. The role created on Vault (called laniakea) authorizes **only** JWT with the given subject (i.e. user identifier) and this audience claim and gives it the policy. This parameter allows the dashboard to retrieve a token with the right bound audience to login on Vault.

Default: orchestrator-dashboard

VAULT_SECRETS_PATH

Description: path on Vault where users secrets are stored.

Default: secrets/

WRAPPING_TOKEN_TIME_DURATION

Description: time duration of the wrapping token sent to the encryption script to upload secrets on Vault.

Default: 1h (1 hour)

READ_POLICY

Description: Secrets reading policy name. This policy has to be configured on Vault with the right permissions to read secrets.

Default: read_only

READ_TOKEN_TIME_DURATION

Description: time duration of the read token, to read secrets on vault

Default: 12h (12 hours)

READ_TOKEN_RENEWAL_TIME_DURATION

Description: renew time period of read token.

Default: 12h (12 hours)

WRITE_POLICY

Description: Secrets writing policy name: The correspondig policy has to be configured on Vault with the right permissions to write secrets.

Default: write_only

WRITE_TOKEN_TIME_DURATION

Description: time duration of the write token, to write secrets on vault

Default: 12h (12 hours)

WRITE_TOKEN_RENEWAL_TIME_DURATION

Description: renew time period of write token.

Default: 12h (12 hours)

DELETE_POLICY

Description: Secrets deletion policy name. This policy has to be configured on Vault with the right permissions to delete secrets.

Default: delete_only

DELETE_TOKEN_TIME_DURATION

Description: time duration of the delete token, to delete secrets on vault

Default: 12h (12 hours)

DELETE_TOKEN_RENEWAL_TIME_DURATION

Description: renew time period of delete token.

Default: 12h (12 hours)

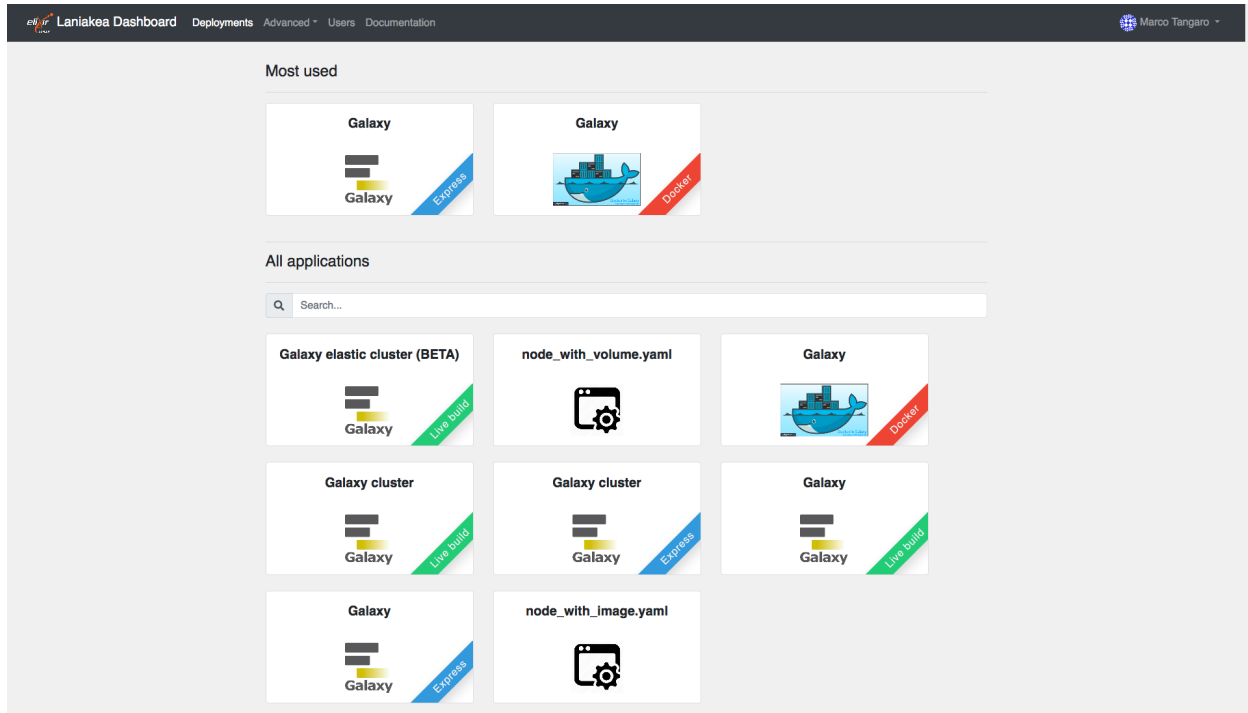
25.1.4 Add new applications

The PaaS Layer accepts deployment requests in the form of TOSCA Templates (see section *TOSCA templates*): a document (YAML syntax) describing the infrastructure to deploy, e.g. the virtual hardware and the software to be installed and configured. Galaxy TOSCA tempaltes are installed during Laniakea installation procedure automaticall on `/opt/laniakea-dashboard-config/tosca-templates`

To add new TOSCA applications copy your tosa template in `/opt/laniakea-dashboard-config/tosca-templates/` and restart the dashboard:

```
# cp tosa_example.yml /opt/laniakea-dashboard-config/tosca-templates/
# docker restart orchestrator-dashboard
```

New applications will be then displayed in the **All applications** section of the dashboard home page.



The Dashboard parses the TOSCA document automatically and renders the user interface with user friendly forms. This allows to extend Laniakea functionalities just adding new templates without any code modification.

For example, the input field in the TOSCA template to select the instance flavour in terms of vCPUs, RAM and disk storage is:

```
instance_flavor:
  type: string
  description: instance flavor (num_cpu, memory, disk)
  default: small
```

where the default value `small` corresponds to a VM with 1 CPU and 2 GB of RAM.

The user input field automatically rendered as text field on the dashboard, allowing the user to modify the flavour modifying the value:

Note: The dashboard automatically renders **all** the entries in the input section of the tosa templates as text fields in the tab `Input values`, for user configuration.

TOSCA templates inputs and outputs name are arbitrary and can be customized. The dashboard support some keywords to enable special features like the SSH key injection and Galaxy restart. Currently available keywords are listed below.



Supported inputs

`instance_key_pub`: user SSH public key is available in the dashboard through the **SSH keys** page (see section [../qs_key_pair](#)). If configured, the public key is automatically assigned to a TOSCA template input value with this name if the input form is left empty. Otherwise, the value inserted in the input form will be assigned to `instance_key_pub` input.

Note: Laniakea exploits this feature to automatically set user public key on Galaxy instances.

`admin_email`: if present among the inputs, this field is automatically filled with user e-mail address.

Supported outputs

`endpoint`: if the endpoint output is present, it is displayed in the **deployments** page of the dashboard, in the endpoint column as clickable url.

`node_ip`: if available among the output values of the single node Galaxy instance, it is consumed by the dashboard as base url to contact the instance APIs to restart the encrypted storage and Galaxy if needed,

`cluster_ip`: if available among the output values of a Galaxy cluster, it is consumed by the dashboard as base url to contact the instance APIs to restart the encrypted storage, the NFS between the nodes and Galaxy.

25.1.5 Application launcher forms customization

The dashboard automatically renders **all** the entries in the input section of the tosca templates as text fields, for user configuration. Despite this allows to easily increase Laniakea applications, it may be necessary to make available to users only some fields to be configured and only some options defined by the service provider.

For this reason we extended the TOSCA templates inputs to create configurable forms. This creates a flexible web interface, allowing straightforward customisation of the user experience through human readable YAML configuration files, which can be easily adapted adding new functionalities to the user interface (e.g. adding a dropdown menu, text fields, toggles...) based on the Laniakea administrator requirements.

To enable configurable forms a parameter file, corresponding to the TOSCA template, is needed. To be automatically parsed by the dashboard the file needs the same name of the TOSCA template file with the extension `.parameters.yaml`. For example if the TOSCA template is named `galaxy.yaml` the corresponding parameters file has to be named `galaxy.parameters.yaml` and has to be placed in `/opt/laniakea-dashboard-config/tosca-parameters`.

Note: The parameters directory can be modified in the dashboard configuration file `config.json` (see section [Basic configuration](#)).

Once added the parameters file, the dashboard needs to be restart to make changes effective.

The dashboard reads the content of this directory and automatically associate to each TOSCA template the corresponding parameters file, if existing.

Note: If the parameters file is available, only the inputs present within it will be shown on the dashboard user interface, allowing to select which TOSCA template input to customize and show.

For example, referring again to the input field to configure the VM virtual hardware, named `instance_flavor`, we have the following TOSCA template input:

```
instance_flavor:
  type: string
  description: instance flavor (num_cpu, memory, disk)
  default: small
```

Rendered as an input text field:

Instance description

Instance description

Input Values Advanced

instance_flavor

small

instance flavor (num_cpu, memory, disk)

storage size

The value `small`, which corresponds to a VM with 1 CPU and 2 GB of RAM, will be displayed as default value in an input text field, allowing the user to modify it and change the VM configuration.

This requires the user to know the hardware presets available on the infrastructure, their names and, above all, it would allow them to choose any possible presets knowing their names.

It is possible to customize this input value inserting an entry **with the same name** in the **YAML** parameters file.

For the `instance_flavor` input, for example, we will have as parameter file input:

```
instance_flavor:
  display_name: "Instance flavour"
  tag_type: "select"
  description: "CPUs, memory size (RAM), root disk size"
  constraints:
    - { value: "medium", label: "Medium (2 cpu, 4 GB RAM, 20 GB dsk)" }
    - { value: "large", label: "Large (4 cpu, 8 GB RAM, 20 GB dsk)" }
    - { value: "xlarge", label: "xLarge (8 cpu, 16 GB RAM, 20 GB dsk)" }
  tab: "Virtual hardware"
```

Which is rendered as a dropdown menu on the dashboard:

File structure

The YAML parameter file has two sections: `tabs` and `inputs`.

tabs

Description This section is optional, if set creates the listed tabs instead of the `Input values` one. It is possible to display each input in the desired tab, using the option `tab` in the input section. If not specified, all inputs will be displayed by default in the `Input values` tab, as default behaviour.

Example

```
# Set here the list of the tabs to be displayed
tabs: [ "tab_1", "tab_2"]
...
```

inputs

Description The list of the inputs is mandatory. Each input must have the same name of the corresponding TOSCA template input value, to be correctly associated.

Example

```
# Set here the list of the tabs to be displayed
tabs: [ "tab_1", "tab_2"]

# Set here a new set of inputs to be displayed
inputs:

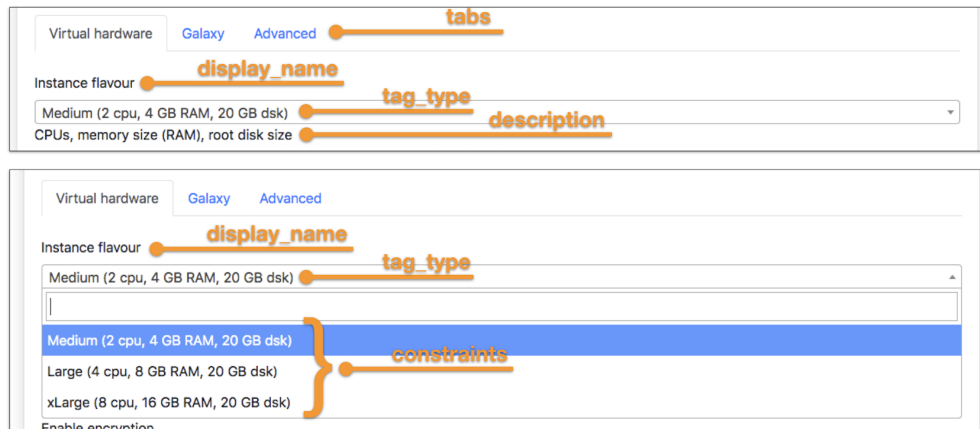
  first_input:
    display_name: "<name to be displayed>"
    tag_type: "<specific tag type for this input>"
    description: "<description to displayed>"
    tab: "tab_1"

  another_input:
    display_name: "<name to be displayed>"
    tag_type: "<specific tag type for this input>"
    description: "<description to displayed>"
    tab: "tab_2"

  ...
```

Input parameters options

Each entry in the YAML parameters file can be customized in order to simplify the user interaction with the UI.



The Laniakea dashboard supports the following options.

display_name

Description The name that will be displayed in the form.

Example

```
input_name: value
display_name: <name_to_be_displayed>
...
```

tag_type

Description Set the tag to be used in the form to generate dropdown menu, radio button... Currently, the following tags are available: text, hidden, email, password, select, radio, ssh_pub_key_type.

More on the available tag types can be found in the section: [Available tag types](#).

Example

```
input_name: value
display_name: <name_to_be_displayed>
tag_type: <selected_tag_type>
...
```

description

Description Override the description present in the tosa template input field.

Example

```
input_name: value
display_name: <name_to_be_displayed>
tag_type: <selected_tag_type>
description: <custom_description_of_the_input>
...
```

placeholder

Description The placeholder attribute specifies a short hint that describes the expected value of an input field/text area. It is available for the following tag_types: text, email, password, ssh_pub_key_type.

Example

```
input_name: value
display_name: <name_to_be_displayed>
tag_type: <selected_tag_type>
description: <custom_description_of_the_input>
placeholder: <custom_placeholder_of_the_input>
...
```

constraints

Description The constraint option is used to define the possible options to choose from. For instance, for select tag type it is possible to specify the selectable values.

It is possible to configure a value attribute, which is the value assigned to the input after the selection, and a label attribute to display.

Example

```
input_name: value
display_name: <name_to_be_displayed>
tag_type: <selected_tag_type>
description: <custom_description_of_the_input>
constraints:
  - { value: "<value_attribute>", label: "<displayed_label>" }
  - { value: "<value_attribute>", label: "<displayed_label>" }
  - { value: "<value_attribute>", label: "<displayed_label>" }
  ...
...
```

required

Description When present it specifies that the input field must be mandatorily filled out before submitting the form.

Example

```
input_name: value
display_name: <name_to_be_displayed>
tag_type: <selected_tag_type>
description: <custom_description_of_the_input>
constraints:
  - { value: "<value_attribute>", label: "<displayed_label>" }
  - { value: "<value_attribute>", label: "<displayed_label>" }
  - { value: "<value_attribute>", label: "<displayed_label>" }
  ...
required: <yes_or_no>
```

tab

Description The tab where the input must be shown.

Example

```
input_name: value
display_name: <name_to_be_displayed>
tag_type: <selected_tag_type>
description: <custom_description_of_the_input>
constraints:
  - { value: "<value_attribute>", label: "<displayed_label>" }
  - { value: "<value_attribute>", label: "<displayed_label>" }
  - { value: "<value_attribute>", label: "<displayed_label>" }
  ...
required: <yes_or_no>
tab: <custom_tab>
```

Available tag types

The Laniakea dashboard currently supports the following tag_types allowing to differentiate user interactions with the UI.

text

Description Defines a one-line text input field.

Example

```
input_example:
display_name: "Text input example"
tag_type: "text"
description: "Input description"
default: "default_value"
tab: "tab_2"
```

Text input example

default_value

Input description

hidden

Description Define an hidden input. The user will not see any entry in the configuration form. Despite this, the dashboard will automatically assign a value to this input.

For example the token to write secrets to vault is assigned with this system, without the user noticing.

Warning: If defined in the tabs section, the tab field is required.

Example

```
input_example:
  tag_type: "hidden"
  default: hidden_default_value
  tab: "tab_1" # Hidden fields needs a tab, if tabs are defined.
```

email

Description The email tag defines a field for an e-mail address. The input value is automatically validated to ensure it is a properly formatted e-mail address.

Example

```
email_input_example:
  display_name: "user e-mail"
  tag_type: "email"
  description: "Type a valid e-mail address."
  tab: "tab_1"
  required: yes
```

user e-mail

Enter your email

Type a valid e-mail address.

password

Description Defines a password field, i.e. a text field with hidden input.

Example

```
password_input_example:
  display_name: "Password input example"
  tag_type: "password"
  description: "Password description"
  default: "default_value"
  tab: "tab_1"
```

Password input example

.....

Password description

select

Description Create drop down list of options, which appears when clicking on form element and allows the user to choose one of the options. The options are described using the constraint attribute.

Example

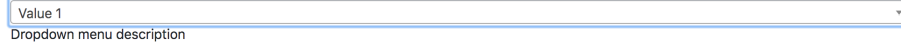
```
input_example:
  display_name: "Dropdown menu example"
  tag_type: "select"
  description: "Dropdown menu description"
```

(continues on next page)

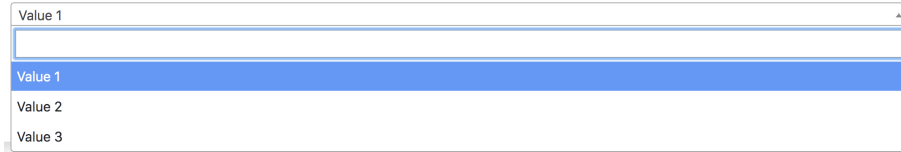
(continued from previous page)

```
constraints:
  - { value: "value1", label: "Value 1" }
  - { value: "value2", label: "Value 2" }
  - { value: "value3", label: "Value 3" }
tab: "tab_1"
```

Dropdown menu example



Dropdown menu example

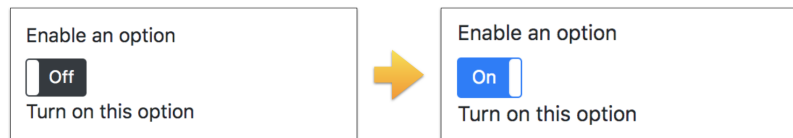


toggle

Description Create a On/Off toggle. On values can be set in the constraints option.

Example

```
input_example:
  display_name: "Enable an option"
  tag_type: "toggle"
  description: "Turn on this option"
  constraints:
    - { value: "True", label: "On" }
  tab: "tab_1"
```



radio

Description Create a radio button to select one of many choices.

Example

```
input_example:
  display_name: "Radio buttons example"
  tag_type: "radio"
  description: "Radio buttons description"
  constraints:
    - { value: "value1", label: "Value 1" }
    - { value: "value2", label: "Value 2" }
    - { value: "value3", label: "Value 3" }
  tab: "tab_1"
```

radio buttons example

- ☒ Value 1
- ☐ Value 2
- ☐ Value 3

Radio buttons description

ssh_pub_key_type

Description Special tag for ssh public key input. It is a `text` field to insert a SSH public key. If the ssh public key is set in the corresponding page (see section `./qs_key_pair`) a placeholder is shown to remember te possibility to load the default key. If no ssh public key is set, nothing is displayed as placeholder.

Warning: The input option has to be mandatorily named `instance_key_pub` in both TOSCA template and parameter file.

Example

```
instance_key_pub:
  display_name: "Insert instance SSH public key"
  tag_type: "ssh_pub_key_type"
  description: "Paste here your SSH public key or configure a default key
↵"
  placeholder: 'Leave blank this field to load your default SSH public_
↵key'
  tab: "tab_1"
  required: yes
```

Insert instance SSH public key

Leave blank this field to load your default SSH public key

Paste here your SSH public key or configure a default key

SSH public key loaded on Laniakea Dashboard

Insert instance SSH public key

Paste here your SSH public key or configure a default key

No SSH public key loaded on Laniakea Dashboard

Supported inputs

instance_flavor_fe

If an input with the same name is used in the TOSCA template, this variable does not trigger any special action. If not, the corresponding menu accepts couples of **number of CPUs** and **RAM size** in the form of python dictionary: {'<tosca_template_cpu_num>':'2', '<tosca_template_mem_size>':'4 GB'}. instance_flavor_fe is commonly used for front-end inputs.

tosca_template_cpu_num and toska_template_mem_size are the corresponding inputs in the TOSCA template. For example, if in the TOSCA template you have:

```
...
topology_template:
  inputs:

    fe_cpus:
      type: integer
      description: Numer of CPUs for the front-end node
      default: 1
      required: yes

    fe_mem:
      type: scalar-unit.size
      description: Amount of Memory for the front-end node
      default: 1 GB
      required: yes

...
```

The corresponding entry in the parameter file will be:

```
instance_flavor_fe:
  display_name: "Front End instance flavour"
  tag_type: "select"
  description: "CPUs, memory size (RAM), root disk size"
  constraints:
    - { value: '{"fe_cpus':'2', 'fe_mem':'4 GB'}", label: "Medium (2 cpu, 4 GB RAM, ↵
↵20 GB dsk)" }
    - { value: '{"fe_cpus':'4', 'fe_mem':'8 GB'}", label: "Large (4 cpu, 8 GB RAM, 20 ↵
↵GB dsk)" }
    - { value: '{"fe_cpus':'8', 'fe_mem':'16 GB'}", label: "xLarge (8 cpu, 16 GB RAM, ↵
↵20 GB dsk)" }
```

instance_flavor_wn

If an input with the same name is used in the TOSCA template, this variable does not trigger any special action. If not, the corresponding menu accepts couples of **number of CPUs** and **RAM size** in the form of python dictionary: {'<tosca_template_cpu_num>':'2', '<tosca_template_mem_size>':'4 GB'}. instance_flavor_wn is commonly used for front-end inputs.

tosca_template_cpu_num and toska_template_mem_size are the corresponding inputs in the TOSCA template. For example, if in the TOSCA template you have:

```
...
topology_template:
  inputs:
```

(continues on next page)

(continued from previous page)

```

wn_cpus:
  type: integer
  description: Numer of CPUs for the WNs
  default: 1
  required: yes

wn_mem:
  type: scalar-unit.size
  description: Amount of Memory for the WNs
  default: 1 GB
  required: yes
...

```

The corresponding entry in the parameter file will be:

```

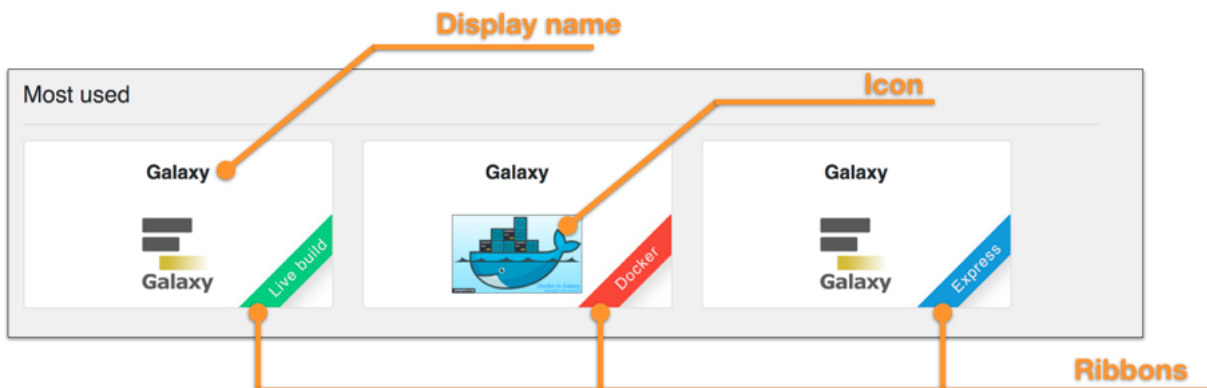
instance_flavor_wn:
  display_name: "Worker Node nstance flavour"
  tag_type: "select"
  description: "CPUs, memory size (RAM), root disk size"
  constraints:
    - { value: "'wn_cpus':'2', 'wn_mem':'4 GB'", label: "Medium (2 cpu, 4 GB RAM, ↵
↵20 GB dsk)" }
    - { value: "'wn_cpus':'4', 'wn_mem':'8 GB'", label: "Large (4 cpu, 8 GB RAM, 20 ↵
↵GB dsk)" }
    - { value: "'wn_cpus':'8', 'wn_mem':'16 GB'", label: "xLarge (8 cpu, 16 GB RAM, ↵
↵20 GB dsk)" }

```

Note: For the full list of supported tag types, see section: *Available tag types*.

25.1.6 Application metadata

The Laniakea dashboard needs some additional information to further customize each application, e.g. the image to show in the home page for each application.



To add metadata information, corresponding to the TOSCA template, a metadata file is needed. To be automatically parsed by the dashboard the file needs the same name of the TOSCA template file with the extension `.metadata.yaml`. For example if the TOSCA template is named `galaxy.yaml` the corresponding metadata file

has to be named `galaxy.metadata.yaml` and has to be placed in `/opt/laniakea-dashboard-config/tosca-metadata`.

Note: The metadata directory can be modified in the dashboard configuration file `config.json` (see section [Basic configuration](#)).

Once added the metadata file, the dashboard needs to be restart to make changes effective.

The dashboard reads the content of this directory and automatically associate to each TOSCA template the corresponding metadata file, if existing.

Metadata file structure

The YAML metadata file has only one section: `metadata`. For example:

```
metadata:
  icon: https://galaxyproject.org/images/galaxy-logos/galaxy_project_logo_square.png
  display_name: "Galaxy"
  virtualization_type: "Docker"
  pinned: 'yes'
  pin_order: 0
```

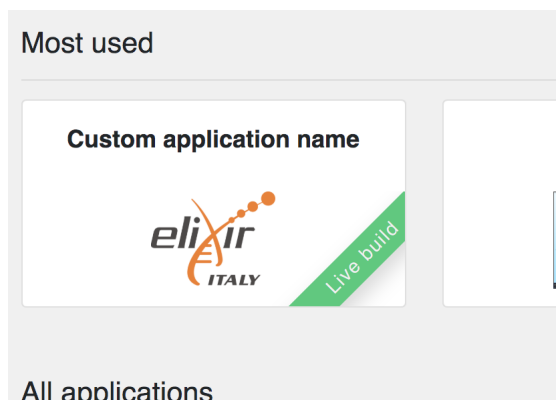
Supported options

icon

Documentation Define the image/icon loaded in the application tile. If no image URL is provided, the Dashboard loads this [icon](#).

Example

```
metadata:
  icon: https://elixir-europe.org/system/files/elixir_node_italy.png
  ...
```

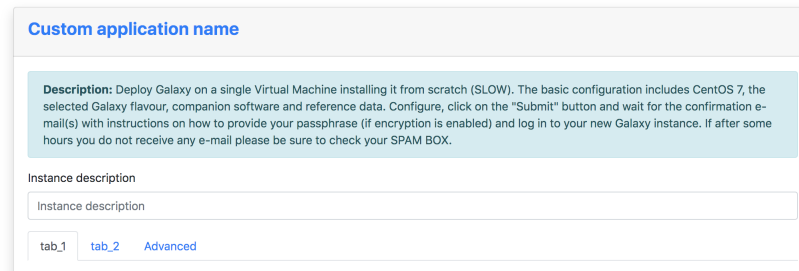


display_name

Documentation Define the name of the application shown in the Dashboard home page and in the configuration form.

Example

```
metadata:
  icon: https://elixir-europe.org/system/files/elixir_node_italy.png
  display_name: "Custom application name"
  ...
```

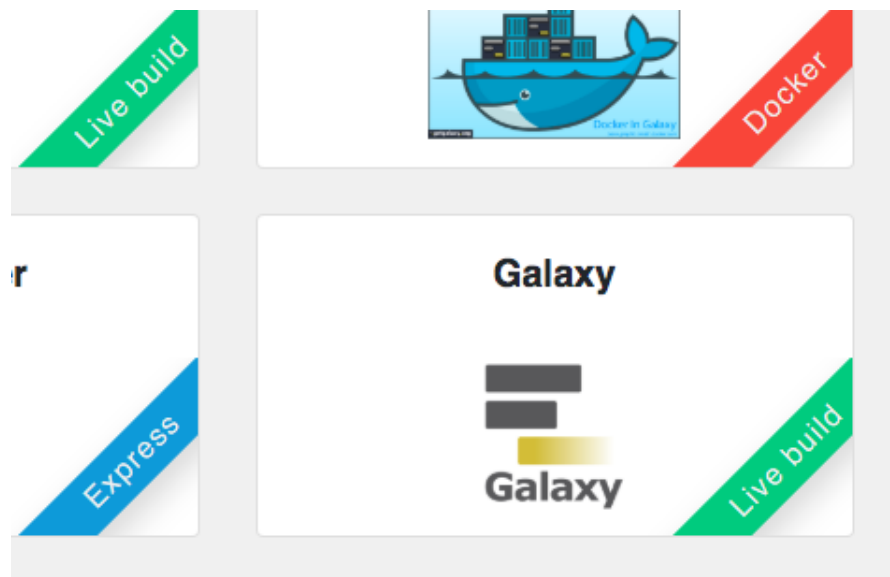


ribbon

Documentation Enable the ribbon on bottom right corner of each tile if True.

Example

```
metadata:
  icon: https://elixir-europe.org/system/files/elixir_node_italy.png
  display_name: "Custom application name"
  ribbon: true
  ribbon_tag: "Test"
  ribbon_color: "yellow"
  ...
```



ribbon_tag

Documentation Define the name to be shown within the colored ribbon on the bottom right corner of the tile. Currently, we adopted three values:

Express: for those applications already installed in the image used to create the Virtual Instance, to speed-up the deployment.

Docker: for those applications run using a Docker container.

Live build: for those applications installed on a bare OS image from scratch.

Example

```
metadata:
  icon: https://elixir-europe.org/system/files/elixir_node_italy.png
  display_name: "Custom application name"
  ribbon: true
  ribbon_tag: "Test"
  ribbon_color: "yellow"
  ...
```

ribbon_color

Documentation Define the color of the ribbons. Possible colors are: white, black, grey, blue, green, turquoise, purple, red, orange, yellow.

Example

```
metadata:
  icon: https://elixir-europe.org/system/files/elixir_node_italy.png
  display_name: "Custom application name"
  ribbon: true
  ribbon_tag: "Test"
  ribbon_color: "yellow"
  ...
```

pinned

Description Define the three applications which can be displayed in the Most used top row.

Example

```
metadata:
  icon: https://elixir-europe.org/system/files/elixir_node_italy.png
  display_name: "Custom application name"
  virtualization_type: "Live build"
  pinned: 'yes'
  ...
```

pin_order

Description Define the order of the three pinned application: 0 for the first place, 1 for the second and 2 for the third.

Example

```
metadata:  
  icon: https://elixir-europe.org/system/files/elixir_node_italy.png  
  display_name: "Custom application name"  
  virtualization_type: "Live build"  
  pinned: 'yes'  
  pin_order: '0'
```

Laniakea installation

Laniakea relies on the INDIGO-DataCloud [software catalogue](#). The Fig. 1 shows the deployment strategy to be followed to install Laniakea.

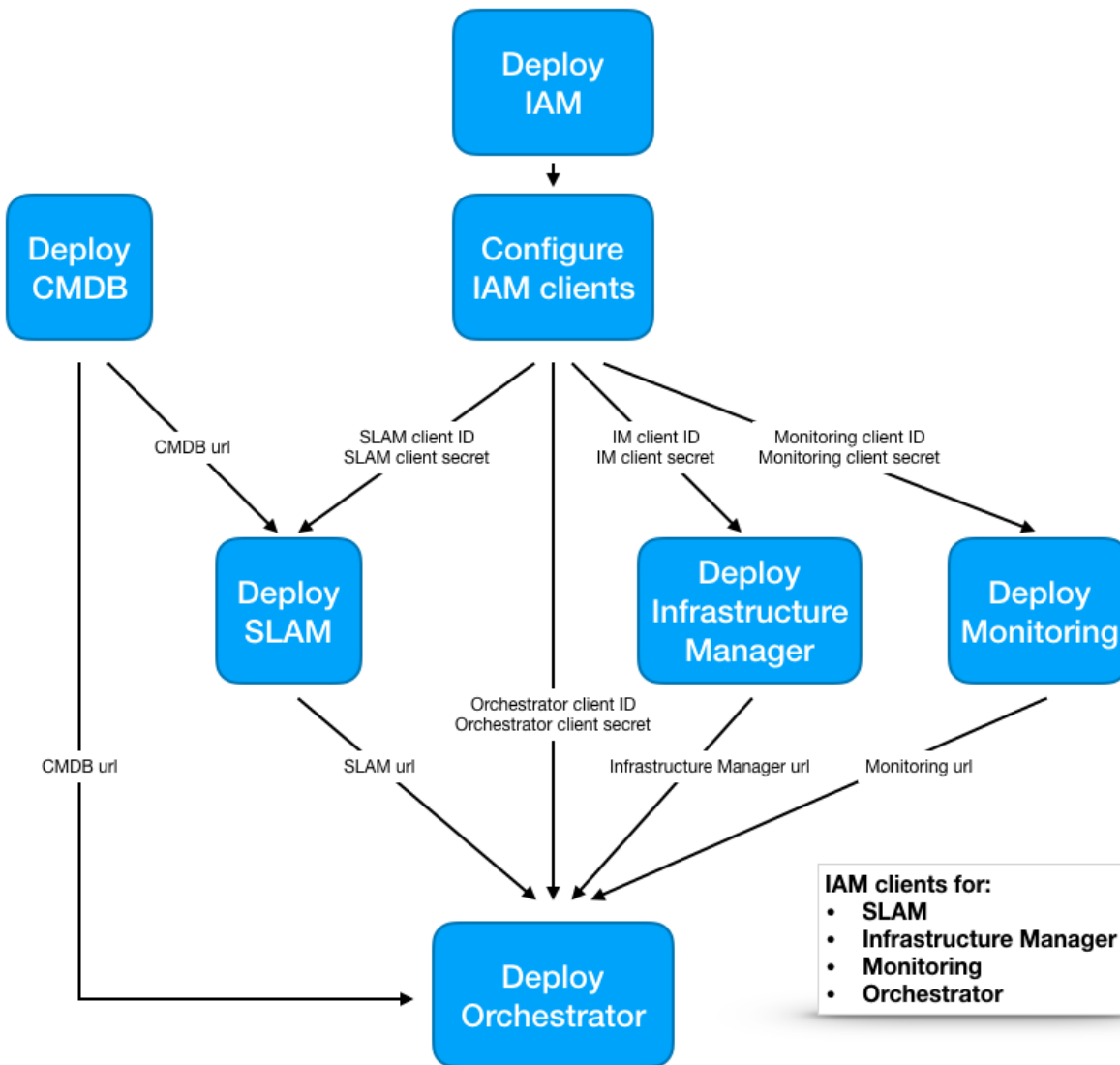
Fig 1: PaaS component architecture scheme

We tested our deployment on OpenStack [Mitaka](#) and [Stein](#), using Ubuntu 16.04 as default OS.

Docker containers are used to provide the INDIGO microservices: each INDIGO component is installed using its official Docker container and run on a specific Virtual Machine.

Tab. 1 shows the VMs tha has to be created, their requirements and the corresponding ports configuration needed to install Laniakea.

Please create the needed VMs with the following configuration:



INDIGO Component	RAM	vCPU	Ports	Network
Proxy server	2 GB	1	22, 443, 8080	public IP private IP
Identity and Access Manager (IAM)	4 GB	2	22, 443	public IP
Infrastructure Manager (IM)	4 GB	2	22, 8800	private IP
Change Management Database (CMDB), Cloud Provider Ranker (CPR)	4 GB	2	22, 443, 5984, 8080, 8081	private IP
Service Level Agreement Manager (SLAM)	2 GB	1	22, 8443, 443	public IP
PaaS Orchestrator	4 GB	2	22, 443	private IP
HashiCorp Vault and Dashboard	4 GB	2	22, 8200, 8250, 443	public IP

In particular we highlight in the table the VM Network configuration, i.e. if the VM needs a public IP address to be accessed from outside or a private IP address is enough.

Instances

Instance Name

Filter

Launch Instance

Delete Instances

More Actions

<input type="checkbox"/>	Instance Name	Image Name	IP Address	Size	Key Pair	Status	Availability Zone	Task	Power State	Time since created	Actions
<input type="checkbox"/>	test-dashboard	RECAS Ubuntu 16.04 LTS [Daily Build 20181114]	90.147.170.32	medium	mtangaro-key	Active	nova	None	Running	0 minutes	Create Snapshot
<input type="checkbox"/>	test-orchestrator	RECAS Ubuntu 16.04 LTS [Daily Build 20181114]	172.30.127.19	medium	mtangaro-key	Active	nova	None	Running	14 minutes	Create Snapshot
<input type="checkbox"/>	test-slam	RECAS Ubuntu 16.04 LTS [Daily Build 20181114]	90.147.75.149	small	mtangaro-key	Active	nova	None	Running	17 minutes	Create Snapshot
<input type="checkbox"/>	test-cmdb-cpr	RECAS Ubuntu 16.04 LTS [Daily Build 20181114]	172.30.127.18	medium	mtangaro-key	Active	nova	None	Running	19 minutes	Create Snapshot
<input type="checkbox"/>	test-im	RECAS Ubuntu 16.04 LTS [Daily Build 20181114]	172.30.127.17	medium	mtangaro-key	Active	nova	None	Running	20 minutes	Create Snapshot
<input type="checkbox"/>	test-lam	RECAS Ubuntu 16.04 LTS [Daily Build 20181114]	90.147.170.28	medium	mtangaro-key	Active	nova	None	Running	21 minutes	Create Snapshot
<input type="checkbox"/>	test-paas	RECAS Ubuntu 16.04 LTS [Daily Build 20181114]	public_net 90.147.75.119 private_net 172.30.127.16	small	mtangaro-key	Active	nova	None	Running	23 minutes	Create Snapshot

Fig 2: INDIGO PaaS VMs view on OpenStack

26.1 Services end-points

Once installed the services will be available at the following endpoint:

Table 1: Services end-points

Service	end-point
IAM	<a href="https://<iam_vm_dns_name>/">https://<iam_vm_dns_name>/
SLAM	<a href="https://<slam_vm_dns_name>:8443/auth">https://<slam_vm_dns_name>:8443/auth
Proxy	<a href="https://<proxy_vm_dns_name>">https://<proxy_vm_dns_name>
CMDB	<a href="https://<proxy_vm_dns_name>/couch/_utils/database.html?indigo-cmdb-v2">https://<proxy_vm_dns_name>/couch/_utils/database.html?indigo-cmdb-v2
IM	<a href="https://<proxy_vm_dns_name>/im">https://<proxy_vm_dns_name>/im
CPR	<a href="https://<proxy_vm_dns_name>/cpr">https://<proxy_vm_dns_name>/cpr
Orchestrator	<a href="https://<proxy_vm_dns_name>/orchestrator">https://<proxy_vm_dns_name>/orchestrator
Dashboard	<a href="https://<dashboard_vm_dns_name>">https://<dashboard_vm_dns_name>

26.2 Service installation

26.2.1 Prerequisites

The installation procedure exploits Ansible to deploy all the INDIGO services.

A Virtual Machine with ansible is mandatory for this purpose, which we will refer to as **control machine** in the following. This VM will be used to run the installation procedure of each INDIGO component on the remote VMs.

Here, we will exploit the same VM we will use to deploy the Proxy Server.

VM configuration

OS	Ubuntu 16.04
vCPUs	2
RAM	4 GB
Network	Public and private IP address.

This VM will be used as `control machine` VM to run Ansible and will also serve as host for the proxy server.

Warning: All the command will be run on this `control machine` VM!

Ansible installation

Ansible can be easily installed following the [documentation](#).

We tested the whole procedure using Ansible 2.8.3 with Ubuntu 16.04.

Configuration

1. Clone the [indigopaas-deploy repository](#), the collection of recipes to install INDIGO-DataCloud PaaS micro-services

```
# git clone -b v1.0 https://github.com/indigo-dc/indigopaas-deploy.git

# cd indigopaas-deploy

# mkdir ansible/inventory
```

2. Create the file `indigopaas-deploy/ansible/inventory/inventory` and set the IP of the virtual machines for each service as shown in the following:

```
[iam]
<iam_vm_public_ip>

[im]
<im_vm_private_ip>

[cmdb]
<cmdb_vm_private_ip>

[cpr]
<cpr_vm_private_ip>

[slam]
<slam_vm_public_ip>

[proxy]
<proxy_vm_private_ip>

[orchestrator]
<orchestrator_vm_private_ip>

[vault]
<vault_vm_public_ip>

[orchestrator-dashboard]
<dashboard_vm_public_ip>
```

Warning: CMDB and CPR will be host on the same Virtual Machine in this guide.

Warning: Vault and the Orchestrator Dashboard will be host on the same Virtual Machine in this guide.

3. Create the `group_vars` directory in `indigopaas-deploy/ansible/inventory/`

```
# cd indigopaas-deploy/ansible/inventory

# mkdir group_vars
```

This directory will be populated with the YAML files with the configuration variables for each indigo component, with the following structure:

```
group_vars/
├── cmdb.yml
└── iam.yml
```

```
|— im.yaml
|— orchestrator.yaml
|— proxy.yaml
|— slam.yaml
```

SSH key pair configuration

To run Ansible on remote hosts we need to configure an SSH connection on each VM.

You can create a new SSH key

```
# ssh-keygen -t rsa -b 4096
```

The default vaules should be ok.

Then you can distribute your new key copying and pasting the public key, i.e. the content of the file `.ssh/id_rsa.pub`, to `/root/.ssh/authorized_keys` on each virtual machine allowing ansible to to execute indigopaas-deploy roles.

Warning: The Ansible roles will install all the services over HTTPS protocol using Let's Eencrypt certificates.

26.2.2 Identity Access Manager (IAM)

The INDIGO Identity and Access Management (IAM) is an Authentication and Authorisation Infrastructure (AAI) service which manages users credentials and attributes, like group membership, and authorization policies to access the resources.

Note: Current IAM version: v1.5.0.rc2

Note: After IAM installation it is needed to configure the Cloud provider identity service to accept the INDIGO IAM OpenID Connect authentication. For Openstack Keystone this is a standard configuration and the documentation can be found [here](#). Furthermore, to enable more OpenID Connect providers configured in the apache `mod_auth_openidc` module used by Keystone, in order to not change Keystone configuration, it is possible to exploit the [ESACO plugin](#). At the moment, for example, it is used with OpenStack at ReCaS-Bari datacenter. An example of integration is available [here](#).

VM configuration

Create VM for IAM. The VM should meet the following minimum requirements:

OS	Ubuntu 16.04
vCPUs	2
RAM	4 GB
Network	Public IP address.

Warning: All the command will be run on the control machine.

Enable Google Authentication

To enable Google authentication access to [Google developers console](#) and create and configure a new credential project.

1. Create Credentials > OAuth Client ID
2. Application Type: Web Application
3. Name: Set a custom Service Provider (SP) name
4. Authorized JavaScript origins: [https://<iam_vm_dns_name>](#).
5. Authorized redirect URIs: [https://<iam_vm_dns_name>/openid_connect_login](#)
6. Create the client
7. Copy your client ID and client secret

Create the file `indigopaas-deploy/ansible/application-oidc.yml`, copying and pasting the client ID, client Secret and the IAM url

```
oidc:
  providers:
  - name: google
    issuer: https://accounts.google.com
    client:
      clientId: <iam_google_client_id>
      clientSecret: <iam_google_client_secret>
      redirectUri: https://<iam_url>/openid_connect_login
      scope: openid,profile,email,address,phone
    loginButton:
      text: Google
      style: btn-social btn-google
      image:
        fa-icon: google
```

Enable ELIXIR-AAI Authentication

To enable you need to request a valid client ID and client Secret. Please read the corresponding [documentation](#).

Then create the file `indigopaas-deploy/ansible/application-oidc.yml`, copying and pasting the client ID, client Secret and the IAM url:

```
oidc:
  providers:
  - name: elixir-aai
    issuer: https://login.elixir-czech.org/oidc/
    client:
      clientId: <iam_elixiraaai_client_id>
      clientSecret: <iam_elixiraaai_client_secret>
      redirectUri: https://<iam_fqdn>/openid_connect_login
      scope: openid,groupNames,bona_fide_status,forwardedScopedAffiliations,email,
↪profile
    loginButton:
```

(continues on next page)

(continued from previous page)

```

text:
style: no-bg
image:
  url: https://raw.githubusercontent.com/Laniakea-elixir-it/ELIXIR-AAI/master/
↪login-button-orange.png
  size: medium

```

Installation

In the following, both Google and ELIXIR-AAI authentication methods will be enabled. To achieve this the `indigopaas-deploy/ansible/application-oidc.yml` with Google and ELIXIR-AAI corresponding clients ID and clients Secret, looks like:

```

oidc:
  providers:
    - name: google
      issuer: https://accounts.google.com
      client:
        clientId: <iam_google_client_id>
        clientSecret: <iam_google_client_secret>
        redirectUri: https://<iam_fqdn>/openid_connect_login
        scope: openid,profile,email,address,phone
      loginButton:
        text: Google
        style: btn-social btn-google
        image:
          fa-icon: google
    - name: elixir-aai
      issuer: https://login.elixir-czech.org/oidc/
      client:
        clientId: <iam_elixiraaai_client_id>
        clientSecret: <iam_elixiraaai_client_secret>
        redirectUri: https://<iam_fqdn>/openid_connect_login
        scope: openid,groupNames,bona_fide_status,forwardedScopedAffiliations,email,
↪profile
      loginButton:
        text:
        style: no-bg
        image:
          url: https://raw.githubusercontent.com/Laniakea-elixir-it/ELIXIR-AAI/master/
↪login-button-orange.png
        size: medium

```

Create the file `indigopaas-deploy/ansible/inventory/group_vars/iam.yml` with the following configured values:

```

iam_fqdn: <iam_vm_dns_name>
iam_mysql_root_password: *****
iam_organization_name: '<your_organization_name>'
iam_logo_url: <logo_url>
iam_account_linking_disable: true
iam_mysql_image: "mysql:5.7"
iam_image: indigoiam/iam-login-service:v1.5.0.rc2-SNAPSHOT-latest
iam_notification_disable: true
iam_notification_from: 'iam@{{iam_fqdn}}'

```

(continues on next page)

(continued from previous page)

```
iam_enable_oidc_auth: true
iam_application_oidc_path: "/root/indigopaas-deploy/ansible/application-oidc.yml"
iam_admin_email: '<valid_email_address>'
```

Warning: Set also your custom mysql password with: `iam_mysql_root_password`.

Warning: Please provide a valid e-mail address, which is mandatory for Let's Encrypt certificate creation.

It is possible to enable mail notification adding the following parameters:

```
iam_notification_disable: false
iam_notification_from: 'laniakea-alert@example.com'
iam_notification_admin_address: <valid_email_address>
iam_mail_host: <mail_server_address>
```

This is needed to allow user registration, e.g. to enable confirmation e-mails.

Run the role using the `ansible-playbook` command:

```
# cd indigopaas-deploy/ansible
# ansible-playbook -i inventory/inventory playbooks/deploy-iam.yml
```

Note: Default administrator credentials:

```
username: admin
password: password
```

Fig.2: IAM login page

Video tutorial

IAM test

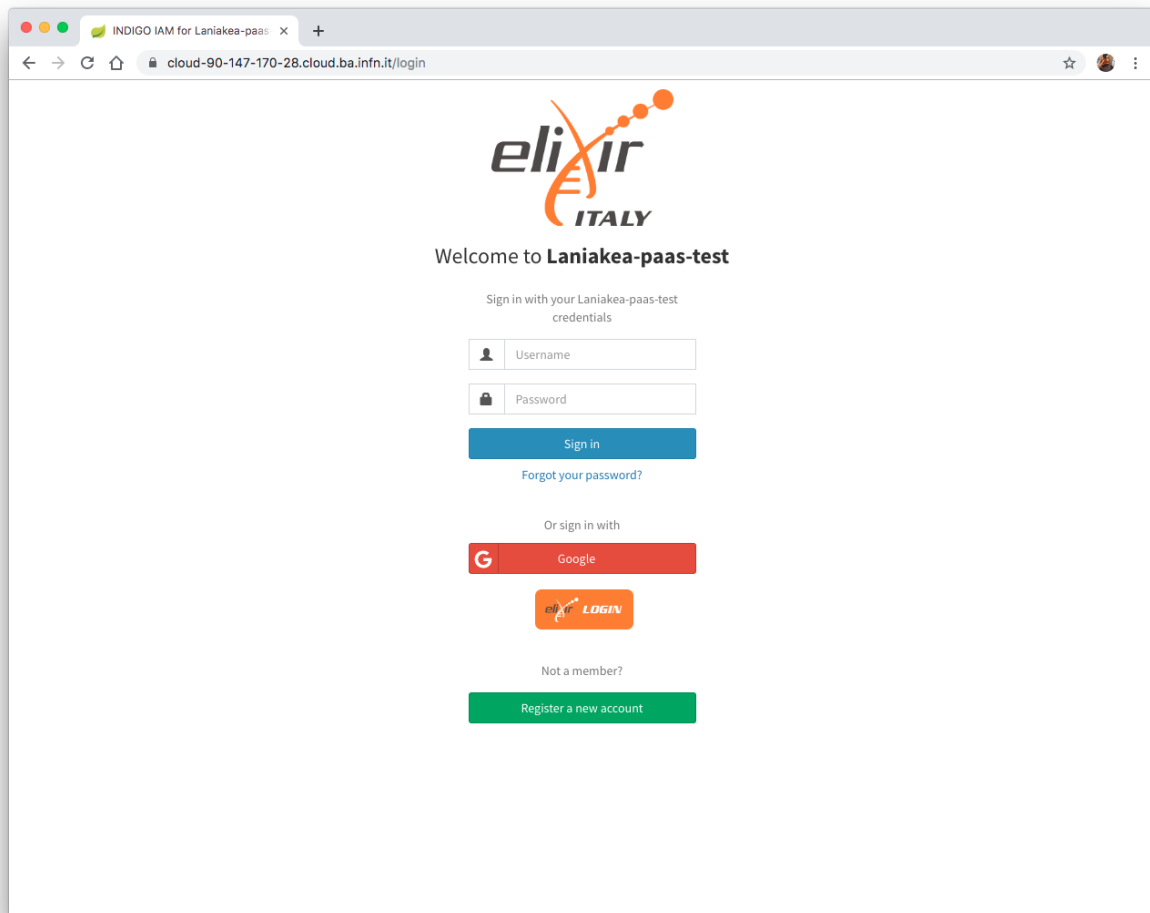
Basic IAM tests.

Test 1: login as admin

1. Login as admin

```
username: admin
password: password
```

Warning: Change the default password.



Test 2: Register a new user

1. Click Register a new account
2. Fill the form
3. Login as admin and accept the request
4. Login as new user

The full registration procedure is described in the [Authentication](#) section.

Test 3: Register using Google account (optional)

1. Sign-in with Google
2. Login as admin and accept the request
3. Login with Google

The full registration procedure is described in the [Authentication](#) section.

Create IAM Client

Registered clients allow to request and receive information about authenticated end-users. Each INDIGO service must authenticate to a dedicated IAM client using a **client id** and a **client secret**.

To create a IAM **client** or a **protectet resource**, please follow these instructions:

Create a IAM client or a protected resource

1. Login as administrator or registered user.
2. Click on **MitreID Dashboard** and then **Self-service client registration** for client creation or **Self-service protected resource registration** to register a new protected resource.
3. Click on *New client* and provides at least the the following parameters:

```
Client name = iam-client-name
redirect URI(s) = http(s)://<service_url>
```

Warning: The redirect URI(s) is required only for client creation.

4. In the **Access** tab configure your client as requested by your service, for example:

```
Scopes: openid, profile, email, address, phone, offline_access
Grant Types: authorization code, refresh
```

5. Save the client.
6. Save **Client ID**, **Client Secret** and **Registration Access Token** or the full output json in the **JSON** tab for future access.

The screenshot shows the 'IAM for Laniakea' Admin User interface. The top navigation bar includes a hamburger menu, the text 'IAM for Laniakea', and a user profile 'Admin User'. The left sidebar lists navigation options: Home, Users (5), Groups, Requests, Tokens (66), AUP, Client management, and MitreID Dashboard. The main content area is titled 'Admin User' and shows a user profile for 'Admin User' (VO administrator, admin) with email 'admin@iam.test' and status 'Active'. Below the profile are buttons for 'Edit Details' and 'Change Password'. To the right of the profile are four sections: 'Groups' (No groups found, + Add to group), 'Group requests' (No request found, + Add request), 'Linked accounts' (No linked accounts found, + Add account), and 'X.509 certificates' (No certificates found, + Add certificate). The bottom of the sidebar indicates 'IAM 1.5.0.rc2-SNAPSHOT (615a333)'.

The screenshot shows the 'INDIGO IAM for Laniakea' user interface. The top navigation bar includes the 'elixir ITALY' logo, the text 'INDIGO IAM for Laniakea', and a user profile 'admin'. The left sidebar lists navigation options categorized into three sections: ADMINISTRATIVE (Manage Clients, Whitelisted Clients, Blacklisted Clients, System Scopes, IAM Dashboard), PERSONAL (Manage Approved Sites, Manage Active Tokens, View Profile Information), and DEVELOPER (Self-service client registration, Self-service protected resource registration). The main content area features the 'elixir ITALY' logo and a 'Welcome!' message, stating 'This is the INDIGO Identity and Access Management (IAM) service.' The bottom of the page shows 'Powered by MITREid Connect' and '© 2016 INFN'.

INDIGO IAM for laniakea
admin

[Home](#) / [Self-service Client Registration](#) / [Register a new client](#)

New Client

Save
Cancel

Main
Access
Credentials
Crypto
Other
JSON

Client ID
Will be generated by the server when the client is saved

Client Secret
Will be generated by the server when the client is saved

Client Configuration URL
Will be generated by the server when the client is saved

Registration Access Token
Will be generated by the server when the client is saved

Client name

Human-readable application name

Redirect URI(s)

+

-

URIs that the client can be redirected to after the authorization page

Logo

URL that points to a logo image, will be displayed on approval page

Enter a logo URL

Terms of Service

URL for the Terms of Service of this client, will be displayed to the user

Policy Statement

URL for the Policy Statement of this client, will be displayed to the user

Home Page

URL for the client's home page, will be displayed to the user

Software ID

Identifier for the software in this client

Software Version

Version of the software in this client

Contacts
List of contacts for administrators of this client.

+

-

List of contacts for administrators of this client.

Software Statement

A software statement is issued by a trusted third party and locks certain elements of a client's registration

Save
Cancel

Powered by MITREid Connect
© 2016 INFN

INDIGO IAM for laniakea

admin

ADMINISTRATIVE

Manage Clients

Whitelisted Clients

Blacklisted Clients

System Scopes

IAM Dashboard

PERSONAL

Manage Approved Sites

Manage Active Tokens

View Profile Information

DEVELOPER

Self-service client registration

Self-service protected resource registration

Home / Self-service Client Registration / Register a new client

New Client

Save Cancel

Main

Access

Credentials

Crypto

Other

JSON

Scope

new scope

openid

profile

email

address

phone

offline_access

OAuth scopes this client is allowed to request

Grant Types

authorization code

client credentials

implicit

password

redelegation

refresh

device

token exchange

Response Types

code

token

id_token

token id_token

code id_token

code token

code token id_token

Subject Type

Public Pairwise

Sector Identifier URI

https://

Sector Identifier for JavaScript

Save

Cancel

Powered by MITREid Connect

© 2016 INFN

INDIGO IAM for laniakea
admin

[Home](#) / [Self-service Client Registration](#) / [Edit an existing client](#)

Edit Client

Save
Cancel
Delete

Main
Access
Credentials
Crypto
Other
JSON

Warning! You MUST protect your **Client ID**, **Client Secret** (if provided), and your **Registration Access Token**. If you lose your Client ID or Registration Access Token, you will no longer have access to your client's registration records and you will need to register a new client.

Client ID

4cda8565-8638-47c8-a68d-96510af8cc9b

Client Secret

RtG9X8_kZn1c-uWf1jcGvpFSXFue4bu_QWpU5FmJL0G0o4f40ounnk5hcAZkpQETBP00h_b7ikNkuv1JJ8z1Qw

Client Configuration URL

https://cloud-90-147-75-207.cloud.ba.infn.it/register/4cda8565-8638-47c8-a68d-96510af8cc9b

Registration Access Token

eyJraWQ1O1Jyc2ExIiwiaWxknIjo1U1MNTY1fQ.eyJhdWQiOi01I0Y2Rm00U2N504NjM4LTQ3YzgtYTY4ZC05NjUxMGFmOGNjOWI1L1Cjpc3M1O1JodHRwczpcL1l1wY2xvdWQ1OTAtMTQ3L1R1TiW55jbG91ZC51Y5SpbmZuLm10XC81L1CjYXQ1OjE1NzEyMzExMzI1IjQZWVlNDZlYWJkZS59.R0wVNaBqDsYm3mSgpyRjmfum_s7-38TYcezmSzvN1BzGkZnpG32H11q0omzaJ6gpmXcFnb8WHJFr78D-JPea_r9aIxCxdwDc q2hdZe_M34_eXQbNtYv0Xb153ALPe3MnIP1Z17Qvz8IKJ3pk1Uj6kmSnJ2981EsG1xAlWhwGW8

Client name

new_client

Human-readable application name

Redirect URI(s)

https://

https://localhost

URIs that the client can be redirected to after the authorization page

Logo

https://

URL that points to a logo image, will be displayed on approval page

Enter a logo URL

Terms of Service

https://

URL for the Terms of Service of this client, will be displayed to the user

Policy Statement

https://

URL for the Policy Statement of this client, will be displayed to the user

Home Page

https://

URL for the client's home page, will be displayed to the user

Software ID

software ID...

Identifier for the software in this client

Software Version

1.0...

Version of the software in this client

Contacts

List of contacts for administrators of this client.

new contact

admin@iam.test

List of contacts for administrators of this client.

Software Statement

eyJ0...

A software statement is issued by a trusted third party and locks certain elements of a client's registration

Save
Cancel
Delete

[illegible]

7. If you need Token Introspection and/or Token exchange, login as Administrator user, and through the **ADMINISTRATIVE, Manage Clients**, in the **Access** tab flag the needed options.

Obtaining an IAM access token

To get a valid IAM access token, please follow these instructions:

Obtaining an IAM access token

Prerequisites

1. Create a IAM client. The Redirect URI is not important, so you can exploit the IAM address itself.
2. Give the client the right **Scopes** and **Grant Types** as in the figure:
3. Save.
4. Save **Client ID**, **Client Secret** and **Registration Access Token** or the full output json in the **JSON** tab for future access.
5. Login as Administrator user and select from the left menu **Manage Clients**.
6. Select the client just created.
7. Navigate to the **Tokens** tab and set it as in the figure and save. In particular the **Device Code Timeout** should not be empty.
8. On any linux distribution, e.g. Ubuntu, Install jq:

```
# apt-get install jq
```

9. Download the following script:

```
wget https://raw.githubusercontent.com/Laniakea-elixir-it/Scripts/master/IAM/dc-  
->get-access-token.sh
```

10. Give dc-get-access-token.sh execution permissions:

```
chmod +x dc-get-access-token.sh
```

11. Create the file iam.rc with the following content:

```
IAM_DEVICE_CODE_CLIENT_ID="<get_iam_token_client_id>"  
IAM_DEVICE_CODE_CLIENT_SECRET="<get_iam_token_client_secret>"  
IAM_TOKEN_ENDPOINT="<iam_url>/token"  
IAM_DEVICE_CODE_ENDPOINT="<iam_url>/devicecode"
```

Get IAM access token

1. Run dc-get-access-token.sh script

```
# ./dc-get-access-token.sh
```

2. Open in a browser the URL obtained from the script and paste code:
3. Authorize the client to create a token:

INDIGO IAM for laniakea

admin

ADMINISTRATIVE

Manage Clients

Whitelisted Clients

Blacklisted Clients

System Scopes

IAM Dashboard

PERSONAL

Manage Approved Sites

Manage Active Tokens

View Profile Information

DEVELOPER

Self-service client registration

Self-service protected resource registration

Home / Manage Clients / Edit Client

Save

Cancel

Main

Access

Credentials

Tokens

Crypto

Other

Edit Client

Scope

new scope

+

openid

profile

email

address

phone

offline_access

scim:read

scim:write

registration:read

registration:write

scim

registration

OAuth scopes this client is allowed to request

Grant Types

authorization code

client credentials

password

implicit

redelegation

device

token exchange

Response Types

code

token

id_token

token id_token

code id_token

code token

code token id_token

Introspection

Allow calls to the Introspection Endpoint?

Subject Type

Public Pairwise

Sector Identifier URI

https://

Sector Identifier for JavaScript

Save

Cancel

Powered by MITREid Connect

© 2016 INFN

RECAS IAM Test Instance for RECAS-BARIadmin

ADMINISTRATIVE

Manage Clients

Whitelisted Clients

Blacklisted Clients

System Scopes

IAM Dashboard

PERSONAL

Manage Approved Sites

Manage Active Tokens

View Profile Information

DEVELOPER

Self-service client registration

Self-service protected resource registration

Home / Self-service Client Registration / Register a new client

New Client

SaveCancel

MainAccessCredentialsCryptoOtherJSON

Client IDWill be generated by the server when the client is saved

Client SecretWill be generated by the server when the client is saved

Client Configuration URLWill be generated by the server when the client is saved

Registration Access TokenWill be generated by the server when the client is saved

Client nameget_IAM_token

Human-readable application name

Redirect URI(s)

https://

https://cloud-90-147-75-207.cloud.ba.infn.it/

RECAS IAM Test Instance for RECAS-BARIadmin

ADMINISTRATIVE

Manage Clients

Whitelisted Clients

Blacklisted Clients

System Scopes

IAM Dashboard

PERSONAL

Manage Approved Sites

Manage Active Tokens

View Profile Information

DEVELOPER

Self-service client registration

Self-service protected resource registration

Home / Self-service Client Registration / Register a new client

New Client

SaveCancel

MainAccessCredentialsCryptoOtherJSON

Scope

new scope

openid

profile

email

address

phone

offline_access

OAuth scopes this client is allowed to request

Grant Types

authorization code

client credentials

implicit

password

redelegation


refresh

device

token exchange

26.2. Service installation

185

 INDIGO IAM for laniakea
 admin

ADMINISTRATIVE

Manage Clients

Whitelisted Clients

Blacklisted Clients

System Scopes

IAM Dashboard

PERSONAL

Manage Approved Sites

Manage Active Tokens

View Profile Information

DEVELOPER

Self-service client registration

Self-service protected resource registration

[Home](#) / [Manage Clients](#) / [Edit Client](#)

Edit Client

Save Cancel

[Main](#) [Access](#) [Credentials](#) [Tokens](#) [Crypto](#) [Other](#)

Access Token Timeout
☐ Access tokens do not time out

seconds

Enter this time in seconds, minutes, or hours.

ID Token Timeout

seconds

Enter this time in seconds, minutes, or hours.

Refresh Tokens
☒ Refresh tokens are issued for this client
This will add the offline_access scope to the client's scopes.

☒ Refresh tokens for this client are re-used

☒ Active access tokens are automatically revoked when the refresh token is used

☒ Refresh tokens do not time out

seconds

Enter this time in seconds, minutes, or hours.

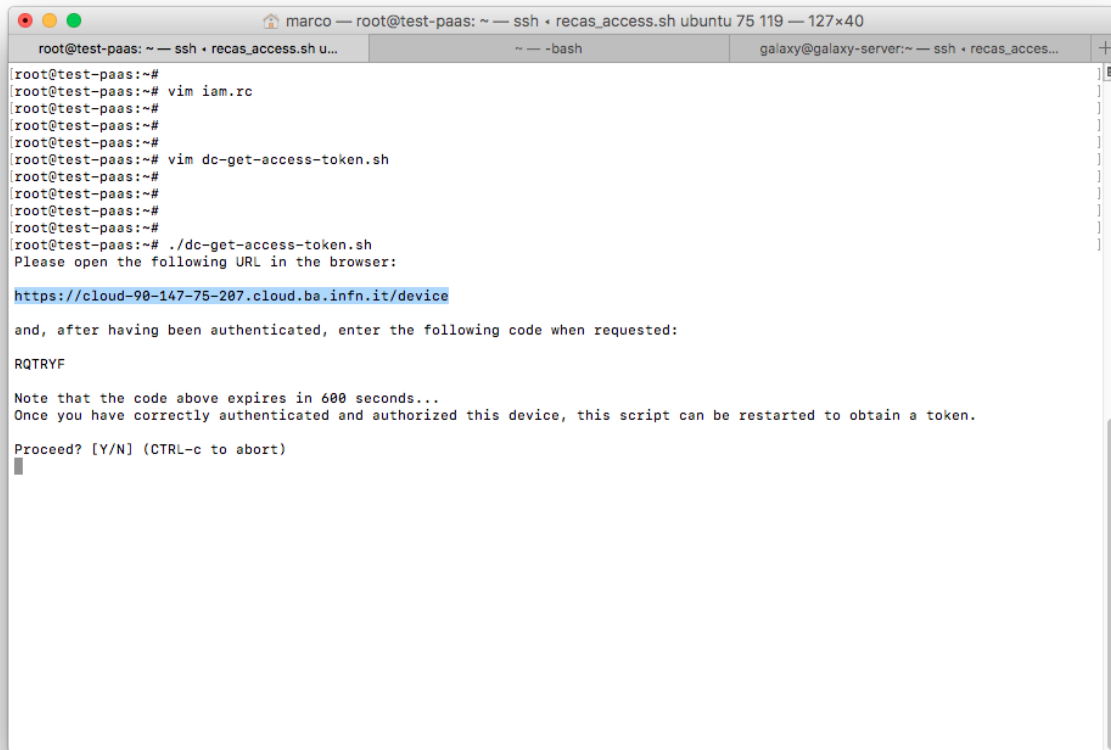
Device Code Timeout

seconds

Enter this time in seconds, minutes, or hours.

Save Cancel

 Powered by MITREid Connect 
© 2016 INFN



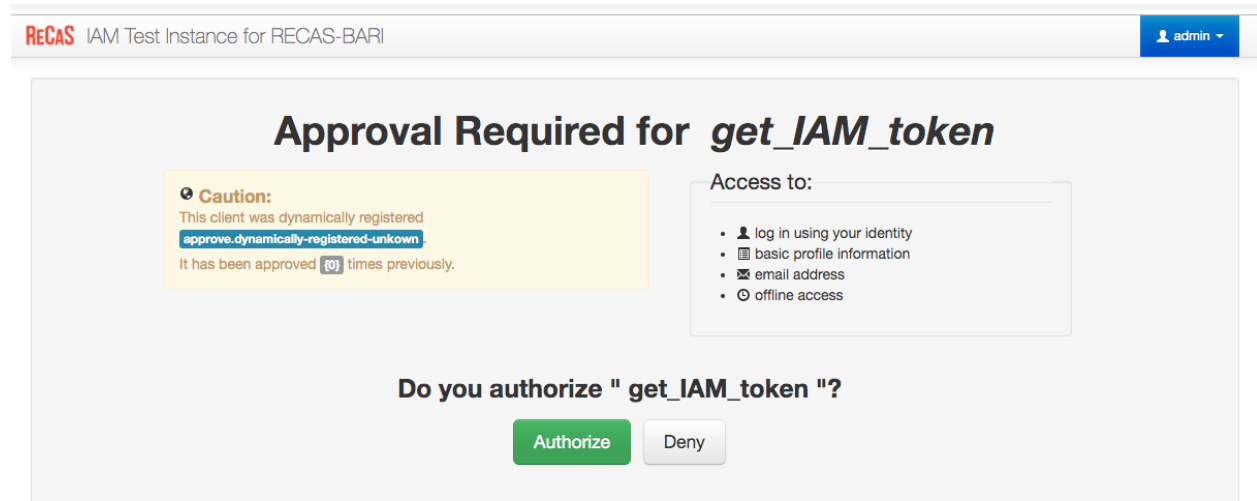
```
marco — root@test-paas: ~ — ssh • recas_access.sh ubuntu 75 119 — 127x40
root@test-paas: ~ — ssh • recas_access.sh u...  ~ — -bash  galaxy@galaxy-server:~ — ssh • recas_acces... +
root@test-paas:~#
root@test-paas:~# vim iam.rc
root@test-paas:~#
root@test-paas:~#
root@test-paas:~#
root@test-paas:~# vim dc-get-access-token.sh
root@test-paas:~#
root@test-paas:~#
root@test-paas:~#
root@test-paas:~#
root@test-paas:~# ./dc-get-access-token.sh
Please open the following URL in the browser:
https://cloud-90-147-75-207.cloud.ba.infn.it/device
and, after having been authenticated, enter the following code when requested:
RQTRYF
Note that the code above expires in 600 seconds...
Once you have correctly authenticated and authorized this device, this script can be restarted to obtain a token.
Proceed? [Y/N] (CTRL-c to abort)
█
```

ReCAS IAM Test Instance for RECAS-BARI admin

Enter Code

RQTRYF

Submit



4. Type ``Y` on the shell script and get your access token:

26.2.3 Proxy server

A proxy server is used to expose IM, CMDB, CPR and the PaaS Orchestrator.

VM configuration

The control machine can be used to run the proxy server. The VM should meet the following minimum requirements:

OS	Ubuntu 16.04
vCPUs	1
RAM	2 GB
Network	Public and private IP address.

Warning: All the command will be run from the control machine.

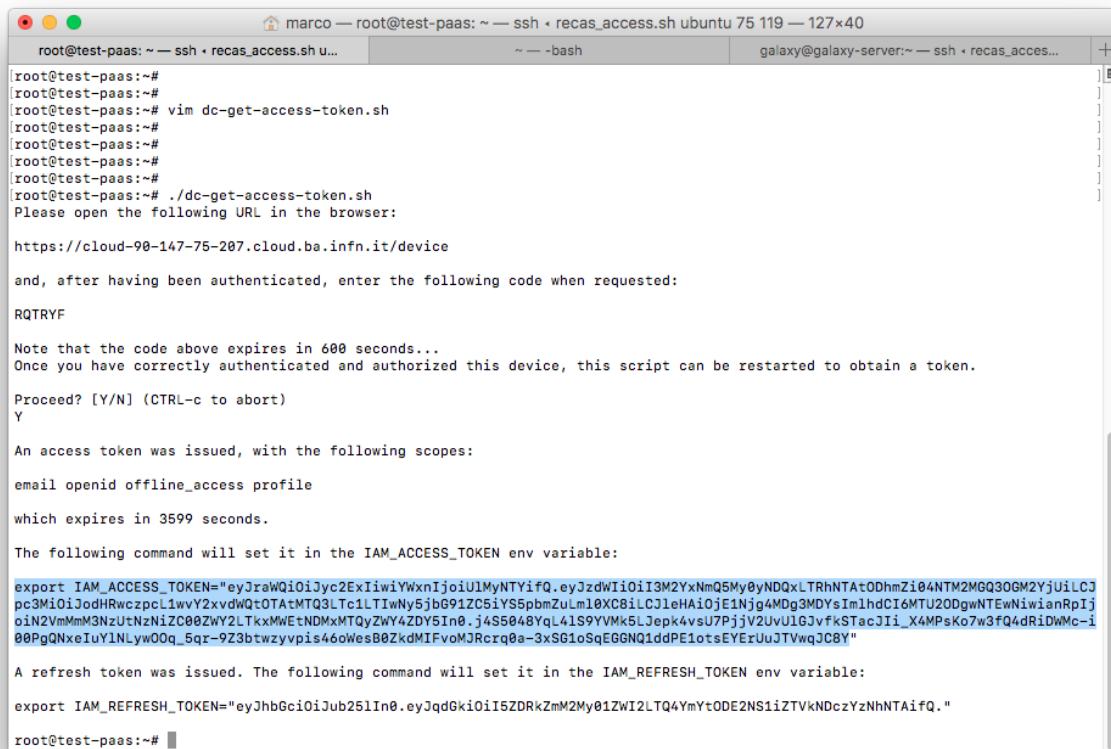
Installation

Create the file `indigopaas-deploy/ansible/inventory/group_vars/proxy.yaml` with the following configured values:

```
letsencrypt_email: "<valid_email_address>"
domain_name: "<proxy_vm_dns_name>"
```

Warning: Please provide a valid e-mail address, which is mandatory for Let's Encrypt certificate creation.

Run the role using `ansible-playbook`



```

root@test-paas: ~ — ssh • recas_access.sh ubuntu 75 119 — 127x40
root@test-paas: ~ — ssh • recas_access.sh u... ~ — -bash galaxy@galaxy-server: ~ — ssh • recas_acces...
root@test-paas:~#
root@test-paas:~#
root@test-paas:~# vim dc-get-access-token.sh
root@test-paas:~#
root@test-paas:~#
root@test-paas:~#
root@test-paas:~# ./dc-get-access-token.sh
Please open the following URL in the browser:

https://cloud-90-147-75-207.cloud.ba.infn.it/device

and, after having been authenticated, enter the following code when requested:

RQTRYF

Note that the code above expires in 600 seconds...
Once you have correctly authenticated and authorized this device, this script can be restarted to obtain a token.

Proceed? [Y/N] (CTRL-c to abort)
Y

An access token was issued, with the following scopes:

email openid offline_access profile

which expires in 3599 seconds.

The following command will set it in the IAM_ACCESS_TOKEN env variable:

export IAM_ACCESS_TOKEN="eyJraWQ1OjJyc2ExIiwiaWxzbnIjOiU1MyNTYifQ.eyJzdWUiOiI3M2YxNmQ5My0yNDQxLTRhNTAtODhmZi04NTM2MGM2YjU1LCJpc3MiOiJodHRwczpcL1wvY2xvdWQtOTA0MTQ3LTc1LTlIiwiaW5jbG91ZC51YS5pbmZuLm10XC8iLCJleHAiOiE1Njg4MDg3MDYsImh0dCI6MTU2ODgwNTEwNiwiianRpIjoiN2VmMmM3NzUtNzNlZC00ZWY2LTkxMwEtNDMxMTQyZWY4ZDY5In0.j4S5048YqL4LS9YVMk5LJepk4vsU7PjjV2UvU1G3vfkSTacJii_X4MPsK07w3fQ4dR1DwMc-i00PgQNxeIuY1Nlyw00q_5qr-9Z3btwzyvpis46oWesB0ZkdMIFvoMJRcrq0a-3xSG1oSqEGGNQ1ddPE1otsEYErUuJTVwqJC8Y"

A refresh token was issued. The following command will set it in the IAM_REFRESH_TOKEN env variable:

export IAM_REFRESH_TOKEN="eyJhbGciOiJub251In0.eyJqdGkiOiI5ZDRkZmM2My01ZWl2LTQ4YmYtODE2NS1iZTVkNDczYzNhNTAiLCJpc3MiOiJodHRwczpcL1wvY2xvdWQtOTA0MTQ3LTc1LTlIiwiaW5jbG91ZC51YS5pbmZuLm10XC8iLCJleHAiOiE1Njg4MDg3MDYsImh0dCI6MTU2ODgwNTEwNiwiianRpIjoiN2VmMmM3NzUtNzNlZC00ZWY2LTkxMwEtNDMxMTQyZWY4ZDY5In0.j4S5048YqL4LS9YVMk5LJepk4vsU7PjjV2UvU1G3vfkSTacJii_X4MPsK07w3fQ4dR1DwMc-i00PgQNxeIuY1Nlyw00q_5qr-9Z3btwzyvpis46oWesB0ZkdMIFvoMJRcrq0a-3xSG1oSqEGGNQ1ddPE1otsEYErUuJTVwqJC8Y"

root@test-paas:~#

```

```
# cd indigopaas-deploy/ansible
# ansible-playbook -i inventory/inventory playbooks/deploy-proxy.yml
```

Video tutorial

26.2.4 Infrastructure Manager (IM)

The **Infrastructure Manager (IM)** is used to deploy virtual infrastructures, e.g. Galaxy and the underlying virtual hardware.

Note: Current IM version: 1.8.6.1

VM configuration

Create VM for IM. The VM should meet the following minimum requirements:

OS	Ubuntu 16.04
vCPUs	2
RAM	4 GB
Network	Private IP address.

Warning: All the command will be run from the control machine VM.

IAM protected resource configuration

Register a new protected resource for IM on IAM:

1. Login on IAM as Administrator User.
2. Navigate to **MitreID Dashboard** and select from the left panel **Self-service protected resource registration**.
3. Create a **New Resource**.
4. Give it a name, e.g. `im_test`.
5. Keep the default configuration and Save.
6. Save **Client ID**, **Client Secret** and **Registration Access Token** or the full output json in the **JSON** tab for future access.
7. As Administrator user select from the left menu **Manage Clients**.
8. Select the client just created.
9. Navigate to the **Tokens** tab and set it as in the figure and save. In particular set:
 - Access Token Timeout: 3600
 - ID Token Timeout: 1800

INDIGO IAM for laniakea

admin

Home / Manage Clients / Edit Client

Edit Client

Save Cancel

Main Access Credentials Tokens Crypto Other

Access Token Timeout ☐ Access tokens do not time out

seconds

Enter this time in seconds, minutes, or hours.

ID Token Timeout seconds

Enter this time in seconds, minutes, or hours.

Refresh Tokens ☐ Refresh tokens are issued for this client

This will add the offline_access scope to the client's scopes.

Device Code Timeout seconds

Enter this time in seconds, minutes, or hours.

Save Cancel

Powered by MITREid Connect

© 2016 INFN

Installation

Create the file `indigopaas-deploy/ansible/inventory/group_vars/im.yaml` with the following configured values:

```
im_image_version: 1.8.6.1
im_repo_tag: vl.8.6
im_mysql_root_password: *****
im_mysql_password: *****
im_cfg_oidc_issuers: 'https://<iam_address>/'
im_cfg_oidc_client_id: '<im_client_id>'
im_cfg_oidc_client_secret: '<im_client_secret>'
im_cfg_ssh_reverse_tunnels: 'True'
im_ansible_version: '2.2.3.0'
```

Warning: Set also your custom mysql password with: `iam_mysql_root_password` and `im_mysql_password`.

Warning: Current supported Ansible version: 2.2.3.0

Run the role using the `ansible-playbook` command:

```
# cd indigopaas-deploy/ansible
# ansible-playbook -i inventory/inventory playbooks/deploy-im.yml
```

Video tutorial

IM configuration

In order to allow IM to distinguish private from public networks, IM needs to be properly configured. Edit the IM configuration file `/etc/im.cfg`, modifying the field `PRIVATE_NET_MASKS` with your favourite text editor, adding the network IP address. The IM will considers IPs not in these subnets as Public IPs.

```
...
PRIVATE_NET_MASKS = 10.0.0.0/8,172.16.0.0/12,192.168.0.0/16,169.254.0.0/16,100.64.0.0/
↪10,192.0.0.0/24,198.18.0.0/15,192.169.0.0/16
...
```

IM testing

1. Get IAM access token (see section *Obtaining an IAM access token*)
2. Download an IM toscatemplate

```
# mkdir im_test
# cd im_test
# wget https://raw.githubusercontent.com/Laniakea-elixir-it/IM-templates/develop/
↪node_with_image.yaml
```

3. Configure the image url as `ost://<keystone_url>/<image_id>`, as for example:

```
image: ost://cloud.recas.ba.infn.it/f38d4e87-cc7e-4035-921b-6b200a9ebaee
```

save and exit.

POST

The POST request instantiate a new deployment

```
curl -k -H 'Content-type: text/yaml' -H 'AUTHORIZATION: type = InfrastructureManager; ↵
↪username = mtangaro; token = eyJraWQiOiJyc2ExIiwiaWxnIjoilMyNTYifQ.
↪eyJzdWIiOiJhOGJjZmU0OS1hOWY3LTQzMDctYWl3YS0wMmMyYmMzZWUxMTgiLCJpc3MiOiJodHRwc3pcLlwwY2xvdWQtOTAtMTY-
↪OKqmt8NvUFWY22ui092yMPTIqCeGuyzjUfVAVw1lTeoZF-ea50RS91qSIHV8AW-
↪O1AZSg4tM5O4W49jVSzVzVq4gLJEMKhBojaJSe9tVf0HE2REcfCb1pYi70jLBhC2TF-
↪tiAmcb0ZywFcF3VEP8DhcPFrbJoiG0_q-vVtzcF4\nid = ost; type = OpenStack; host =
↪<keystone_url>; username = <username>; password = ***** ; tenant = <tenant_name>; ↵
↪service_region = <region>' -i -X POST https://cloud-90-147-75-119.cloud.ba.infn.it/
↪im/infrastructures --data-binary "@node_with_image.yaml"
```

HTTP/1.1 100 Continue

(continues on next page)

(continued from previous page)

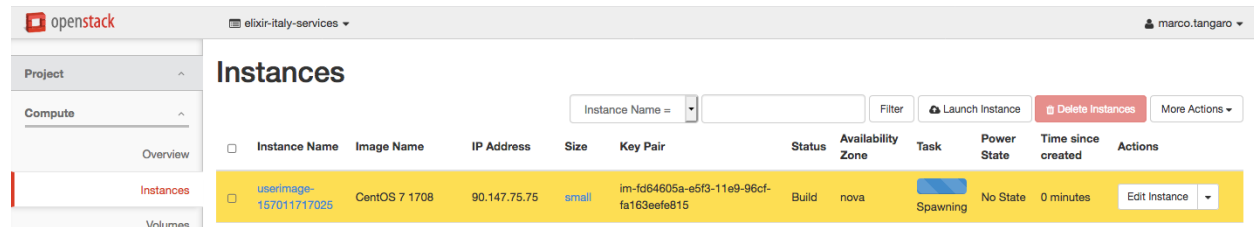
```

HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Thu, 03 Oct 2019 15:54:37 GMT
Content-Type: text/uri-list
Content-Length: 100
Connection: keep-alive
Infid: c90796fe-e5f5-11e9-930c-fa163eefe815

https://cloud-90-147-75-119.cloud.ba.infn.it/im/infrastructures/c90796fe-e5f5-11e9-
→930c-fa163eefe815

```

Where Infid, in this case a9feb488-e5f3-11e9-aafa-fa163eefe815, is the IM UUID of your deployment



GET

The GET request can be used to list the VMs associated to a deployment:

```

# curl -k -H 'Content-type: text/yaml' -H 'AUTHORIZATION: type =
→InfrastructureManager; username = mtangaro; token =
→eyJraWQiOiJyc2ExIiwiaWxnIjoiaUlMyNTYifQ.
→eyJzdWIiOiJhOGJjZmU0OS1hOWY3LTQzMdctYWVzYS0wMmMyYmMzZWUxMTgiLCJpc3MiOiJodHRwc2pLlWVY2xvdWQtOTAtMTQ.
→OKqmt8NvUFWY22ui092yMPTIqCeGuyzjUfVAW1lTeoZF-ea50RS91qSIHV8AW-
→01AZSg4tM5O4W49jVSzVzVq4gLJEMKhBojaJSe9tVf0HE2REcfCb1pYi70jLBhC2TF-
→tiAmcb0ZywFcF3VEP8DhcPFrbJoiG0_q-vVtzcF4\nid = ost; type = OpenStack; host =
→<keystone_url>; username = <username>; password = ***** ; tenant = <tenant_name>;
→service_region = <region>' -i -X GET https://cloud-90-147-75-119.cloud.ba.infn.it/
→im/infrastructures/c90796fe-e5f5-11e9-930c-fa163eefe815
HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Thu, 03 Oct 2019 18:49:43 GMT
Content-Type: text/uri-list
Content-Length: 106
Connection: keep-alive

https://cloud-90-147-75-119.cloud.ba.infn.it/im/infrastructures/c90796fe-e5f5-11e9-
→930c-fa163eefe815/vms/0

```

The GET request can be used to list all VMs information:

```

# curl -k -H 'Content-type: text/yaml' -H 'AUTHORIZATION: type =
→InfrastructureManager; username = mtangaro; token =
→eyJraWQiOiJyc2ExIiwiaWxnIjoiaUlMyNTYifQ.
→eyJzdWIiOiJhOGJjZmU0OS1hOWY3LTQzMdctYWVzYS0wMmMyYmMzZWUxMTgiLCJpc3MiOiJodHRwc2pLlWVY2xvdWQtOTAtMTQ.
→OKqmt8NvUFWY22ui092yMPTIqCeGuyzjUfVAW1lTeoZF-ea50RS91qSIHV8AW-
→01AZSg4tM5O4W49jVSzVzVq4gLJEMKhBojaJSe9tVf0HE2REcfCb1pYi70jLBhC2TF-
→tiAmcb0ZywFcF3VEP8DhcPFrbJoiG0_q-vVtzcF4\nid = ost; type = OpenStack; host =
→<keystone_url>; username = <username>; password = ***** ; tenant = <tenant_name>;
→service_region = <region>' -i -X GET https://cloud-90-147-75-119.cloud.ba.infn.it/
→im/infrastructures/c90796fe-e5f5-11e9-930c-fa163eefe815/vms/0

```

(continues on next page)

(continued from previous page)

```

HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Thu, 03 Oct 2019 18:52:38 GMT
Content-Type: text/plain
Content-Length: 2476
Connection: keep-alive

network public_net ( outports = '9001/tcp-9001/tcp,9000/tcp-9000/tcp' and
provider_id = 'public_net' and
outbound = 'yes' )
system simple_node (
instance_name = 'userimage-157011807495' and
cpu.arch = 'x86_64' and
disk.0.image.url = 'ost://cloud.recas.ba.infn.it/f38d4e87-cc7e-4035-921b-6b200a9eabee
↪' and
net_interface.0.ip = '90.147.75.76' and
memory.size = 2048M and
cpu.count = 1 and
disk.0.os.credentials.private_key = '-----BEGIN RSA PRIVATE KEY-----
MIIEQAIbAAKCAQEAMNLLui9dXce/1XAJ2linN5K4zrpqst7cAJmZwnbIrVqEiNa
q60MhINASHP5VR0HQpMqWuClDlDE09XGp6qGzPa1+Rfn894j5jd9X/H/HFbvMYN4
DFq5AF+Lwj0AkCQT4+R/9iYYJbjuZug3UflAspCYzg7Ht94lVRNAzhLCM++96kkO
j9jNxI5enX+MdKA0n1mOVhAyRi3wtfaQmhk2q47R1X9URqeE8UaZf6xL9KincVb/
X94Wnc0dtbQfyHsNWM/Oo78pkrSfKxUNHC18Em/ZfJ+ADm7u27rY+V2eiKK+kahm
8PCvOGO3qblBqwcnpUH/clVm5JGaiLal/keDlQIDAQABAoIBAAnjsjlVLVSRRY+5
VwitvwxwqTbvhytLEpWTWwjjiO726Za1VZA4tuntrQ5lQv1+e9L+LSyz+tdJK+U
qOtWtKx01qfMgY6ddHNEaf+YeGrMEWSB3nXmNQyaIkAqlGu/ee4IbmNuaaefRQYx
xsquN4qWotzKxg/W91F/EnWD2u3jXyxOAOmRFBY5y1pU9YhcDR8w46+ZyV7h04f8
hFbJILYA5kzmFtWHSUq5yGLlccddGSK40EGJNpni4gNh61D4DOD/yzCrgqL+th
wfwSMOVhxWPBKQlQDHqOyb2lTVc+5UeFBwb+3LbfCdJfA7Sfi4Dpygvv1FPELC1
ZGF1+0ECggCBAMUi+q15uresVXCyQrQ9HmZ2FcRwNc9BtB0ag5RuuuFNsh4suPcL
hxJVG35vTfRgf9USO2WzCrgiAHZij6yT/USIoAFOUvLrtg+T5abd6Fec3lrvXgsL
LLVXONPK0RVqKhTAgNzEAqGEOkd8Ew3WWH0Klrrw3uxp1sEO8I3kt8/RAoIagQDG
dImkibakryLFd833OWdG33C1WT0kgFRBerq8taHZjdBejze9n67LzJludW771qUQ
VCpH424lxP7qIT+hNs/pFXi9Sq/VBsbfehwPoetDgv0yKSP1mRHiKOvTu47hHdst
4q4iwxuYENLbjjESMKR2ngelpJMe2EUFURWHx87MhQKCAIAvp/QXqbzEmCmTc9SC
Q+AsftFmSoYHk2eaPYwfhWEyBB1SCBeyyRufB+n8l6WttQJSHPU08aJevwGFLzPy
UVhBkBG2HxwYU3kQrP0waKa5P1fvfdYrL0lgkVkpShFfbum7WIoOVGgaaZ+5Fjp4
9t8vYzbrSGO8nRloUFdAxdVcQKCAIEAspxsSmt8xjHhCR6MhfiAfK9wE3ZIGX
UNWA9hd9dSmJOY7oOlxykE2uRRiopv8Jy4fyBH9Fv/dm7oq9F/abYsVPwghT8wAG
N1VLq0Wq0TYvY9Rh58G3ti3dCsZd5vdXJhO3YNDzJAT/o+6xeg0L8zKC/ZL8UeWN
NxugpG/KSYECggCAbcJeVFjNQYEhroRg2dmVY/Y6cmndvCUudDs8hvtTmvWmFGri
7dY1T7ACdWAbFYh+Q1x2SswHAOXC+FYJ2HJ8InbKeRAlQ7KDgDsofPGRCTRUL9HO
mZQkIZqryAcSnC++OLNnbFGsTY4vhyotb3Igr/pC+6RSgqJFabFtA7Ttkgg=
-----END RSA PRIVATE KEY-----
' and
provider.host = 'cloud.recas.ba.infn.it' and
disk.0.free_size = 20G and
instance_id = 'c6e54ale-f2ce-4cd5-a38f-f26858d57d7c' and
instance_type = 'small' and
state = 'unconfigured' and
provider.port = 5000 and
provider.type = 'OpenStack' and
net_interface.0.connection = 'public_net' and
disk.0.os.name = 'linux' and
disk.0.os.credentials.username = 'cloudadm'

```

(continues on next page)

)

```
# curl -k -H 'Content-type: text/yaml' -H 'AUTHORIZATION: type =  
→InfrastructureManager; username = mtangaro; token =  
→eyJraWQiOiJyc2ExIiwiaWxnIjoiaUlMyNTYifQ.  
→eyJzdWIiOiJhOGJjZmU0OSIhOWY3LTQzMdctYWIZYS0wMmMyYmMzZWUxMTg1Cjpc3MiOiJodHRwczpcL1lwY2  
→OKqmt8NvUFWY22ui092yMPTIqCeGuyzjUfVAW1lTeoZF-ea50RS91qSIHV8AW-  
→01AZSg4tM504W49jVSzVzVq4gLJEMKhBojaJSe9tVf0HE2REcfCblpYi70jLBhC2TF-  
→tiAmcb0ZywFcF3VEP8DhcPFRbd_JoiG0_q-vVtzCF4\nid = ost; type = OpenStack; host =  
→<keystone_url>; username = <username>; password = ***** ; tenant = <tenant_name>;  
→service_region = <region>' -i -X DELETE https://cloud-90-147-75-119.cloud.ba.infn.  
→it/im/infrastructures/c90796fe-e5f5-11e9-930c-fa163eefe815
```

HTTP/1.1 200 OK
Server: nginx/1.10.3 (Ubuntu)
Date: Thu, 03 Oct 2019 15:43:52 GMT
Content-Type: text/plain
Content-Length: 0
Connection: keep-alive

```
export IAM_ACCESS_TOKEN="..."

curl -k -H 'Content-type: text/yaml' -H "Authorization: id = ost; type = OpenStack;_
→host = https://cloud.recas.ba.infn.it:5000/; username = laniakea; password = $IAM_
→ACCESS_TOKEN; tenant = oidc; auth_version = 3.x_oidc_access_token; service_region =_
→recas-cloud;\nuid = im; type = InfrastructureManager; token = $IAM_ACCESS_TOKEN" -i -
→X POST https://cloud-90-147-75-119.cloud.ba.infn.it/im/infrastructures --data-
→binary "@node_with_image.yaml"
```

```
export IAM_ACCESS_TOKEN="..."

curl -k -H 'Content-type: text/yaml' -H "Authorization: id = ost; type = OpenStack;
↪host = https://cloud.recas.ba.infn.it:5000/; username = laniakea; password = $IAM_
↪ACCESS_TOKEN; tenant = oidc; auth_version = 3.x_oidc_access_token; service_region =
↪recas-cloud;\nid = im; type = InfrastructureManager; token = $IAM_ACCESS_TOKEN" -i -
↪X GET https://cloud-90-147-75-119.cloud.ba.infn.it/im/infrastructures
```

DELETE

```
export IAM_ACCESS_TOKEN="..."

curl -k -H 'Content-type: text/yaml' -H "Authorization: id = ost; type = OpenStack;
↪host = https://cloud.recas.ba.infn.it:5000/; username = laniakea; password = $IAM_
↪ACCESS_TOKEN; tenant = oidc; auth_version = 3.x_oidc_access_token; service_region =
↪recas-cloud;\nid = im; type = InfrastructureManager; token = $IAM_ACCESS_TOKEN" -i -
↪X DELETE https://cloud-90-147-75-119.cloud.ba.infn.it/im/infrastructures/
↪<infrastructure_uuid>
```

FAQ

Where are the deployments log?

The deployment logs are available in `/var/tmp/im/<im-id>/<deployment_ip>/ctxt_agent.log`. For example:

```
# tail -f /var/tmp/.im/1b0e064c-9a29-11e7-9c45-300000000002/90.147.102.27_0/ctxt_
↪agent.log
```

Note: After each ansible role run, the log file is deleted!!

References

[IM configuration](#)

[IM APIs documentation](#)

26.2.5 CMDB and CPR

The Configuration Management DataBase (CMDB) is used to contain all the configuration items (CIs) that are valid to manage the infrastructure.

The Cloud Provider Ranker is a standalone REST WEB Service which ranks cloud providers.

CMDB and CPR are installed on the same machine.

Note: Current CMDB version: indigo_2

Note: Current CPR version: indigo_2

VM configuration

Create VM for CMDB and CPR. The VM should meet the following minimum requirements:

OS	Ubuntu 16.04
vCPUs	2
RAM	4 GB
Network	Private IP address.

Warning: All the command will be run from the control machine VM.

CMDB installation

Create the file `indigopaas-deploy/ansible/inventory/group_vars/cmdb.yaml` with the following configured values:

```
cmdb_crud_password: *****
cmdb_oidc_userinfo: https://<proxy_url>/userinfo
```

Run the role using the `ansible-playbook` command:

```
# cd indigopaas-deploy/ansible
# ansible-playbook -i inventory/inventory playbooks/deploy-cmdb.yml
```

CMDB installation video tutorial

CMDB configuration

The current version of CMDB is supporting set of configuration elements that are vital for INDIGO operations:

- providers: organizational entity that owns or operates the services;
- services (both computing and storage): main technical component description defining type and location of technical endpoints;
- images: local service metadata about mapping of INDIGO-wide names of images, which are necessary to translate TOSCA description into service specific request.

CMDB needs to be populated with IaaS provider, services and images information.

Warning: SSH on CMDB virtual machine.

1. Create a directory called **cmdb-data**

```
# mkdir cmdb-data
```

2. Create the file `cmdb-data/provider.json`

```
{
  "_id": "",
  "data": {
    "name": "",
    "country": "",
    "country_code": "",
    "roc": "",
    "subgrid": "",
    "giis_url": "",
    "owners": [ "" ]
  },
  "type": "provider"
}
```

The `_id` field identifies the Cloud Provider and can be set as preferred

Warning: The provider **owners** list requires at least a valid mail address, since this user has to be used for the resource negotiation procedure, during SLAM configuration (see section `slam`)

3. Create the file `cmdb-data/service.json`

```
{
  "_id": "",
  "data": {
    "service_type": "",
    "endpoint": "",
    "provider_id": "",
    "region": "",
    "sitename": "",
    "hostname": "",
    "type": "compute"
  },
  "type": "service"
}
```

Here the `_id` string identifies the service and can be set as preferred. On the contrary, the `provider_id` is the `_id` previously set in the `provider.json` file.

4. Create the file `cmdb-data/image.json`

```
{
  "type": "image",
  "data": {
    "image_id": "",
    "image_name": "",
    "architecture": "",
    "type": "linux",
    "distribution": "ubuntu",
    "version": "16.04",
    "service": ""
  }
}
```

where the `image_id` is the image ID on the Cloud Provider Manager, e.g. OpenStack.

The **service** field has to be set with the `_id` set in the `service.json` file.

Note: The `image_name` field is the parameter which is used in the **image** field in the `tosca` template to identify the image to use (see section [Galaxy template](#))

5. Add providers, services and images to CMDB.

Create the file `cmdb-add-data.sh` with the content:

```
#!/bin/bash

source /etc/cmdb/.cmdbenv

if [[ -z "$CMDB_CRUD_USERNAME" ]]; then
echo ENV variable CMDB_USER not set
exit 1
fi

if [[ -z "$CMDB_CRUD_PASSWORD" ]]; then
echo ENV variable CMDB_PASSWORD not set
exit 1
fi

if [[ -z "$1" ]]; then
echo "
usage: $0 <json>
"
exit 1
fi
```

give it execution permissions:

```
chmod +x cmdb-add-data.sh
```

Finally you can upload informations to `cmdb` using `curl`:

```
curl -X POST http://cmdb:<cmdb_crud_password>@localhost:5984/indigo-cmdb-v2 -H
↪"Content-Type: application/json" -d@cmdb-data/provider.json

curl -X POST http://cmdb:<cmdb_crud_password>@localhost:5984/indigo-cmdb-v2 -H
↪"Content-Type: application/json" -d@cmdb-data/service.json

curl -X POST http://cmdb:<cmdb_crud_password>@localhost:5984/indigo-cmdb-v2 -H
↪"Content-Type: application/json" -d@cmdb-data/image.json
```

6. Check on CMDB couchDB if your configuration has been uploaded from your browser at the following end-point: `https://<proxy_url>/couch/_utils/database.html?indigo-cmdb-v2`

CMDB couchDB after the configuration process with provider, service and image.

Note: All CMDB image are listed at the address: https://<proxy_url>/cmdb/image/list?include_docs=true

CMBD configuration json example

These are the configuration files used for **Laniakea@ReCaS** service, the Laniakea installation at the ReCaS Datacenter:

provider.json

```
{
  "_id": "provider-RECAS-BARI",
  "data": {
    "name": "RECAS-BARI",
    "country": "Italy",
    "country_code": "IT",
    "roc": "NGI-IT",
    "subgrid": "",
    "giis_url": "ldap://cloud-bdii.recas.ba.infn.it:2170/GLUE2DomainID=RECAS-BARI,
    ↪o=glue",
    "owners": [ "*****" ]
  },
  "type": "provider"
}
```

service.json

```
{
  "_id": "service-RECAS-BARI-openstack",
  "data": {
    "service_type": "eu.egi.cloud.vm-management.openstack",
    "endpoint": "https://cloud.recas.ba.infn.it:5000/v3",
    "provider_id": "provider-RECAS-BARI",
    "region": "recas-cloud",
    "sitename": "RECAS-BARI",
    "hostname": "cloud.recas.ba.infn.it",
    "type": "compute"
  }
}
```

(continues on next page)

(continued from previous page)

```

    },
    "type": "service"
}

```

image.json

```

{
  "type": "image",
  "data": {
    "image_id": "8f667fbc-40bf-45b8-b22d-40f05b48d060",
    "image_name": "RECAS-BARI-ubuntu-16.04",
    "architecture": "x86_64",
    "type": "linux",
    "distribution": "ubuntu",
    "version": "16.04",
    "service": "service-RECAS-BARI-openstack"
  }
}

```

CMDB configuration video tutorial**CPR installation**

CPR does not need any configuration. Run the role using the `ansible-playbook` command:

```

# cd indigopaas-deploy/ansible

# ansible-playbook -i inventory/inventory playbooks/deploy-cpr.yml

```

CPR video tutorial**26.2.6 SLA Tool****26.2.7 PaaS Orchestrator**

PaaS Orchestrator is the core component of the PaaS layer. It collects high-level deployment requests from the software layer, and coordinates the resource or service deployment.

Note: Current Orchestrator version: 2.1.2-final

VM configuration

Create VM for IM. The VM should meet the following minimum requirements:

OS	Ubuntu 16.04
vCPUs	2
RAM	4 GB
Network	Private IP address.

IAM protected resource configuration for the Orchestrator

1. Login on IAM then **MitreID Dashboard** and select **Self-service protected resource registration** as Administrator user.
2. Select **New Resource** with the following parameters

```
Name: orchestrator_client

Scopes: openid, profile, offline_access
```

ReCAS IAM Test Instance for RECAS-BARI admin

Home / Self-service Protected Resource Registration / rsreg.new

[Save](#) [Cancel](#)

Main [Access](#) [Credentials](#) [JSON](#)

Client ID Will be generated by the server when the client is saved

Client Secret Will be generated by the server when the client is saved

Client Configuration URL Will be generated by the server when the client is saved

Registration Access Token Will be generated by the server when the client is saved

Client name

Human-readable application name

Logo

URL that points to a logo image, will be displayed on approval page

Terms of Service

URL for the Terms of Service of this client, will be displayed to the user

Policy Statement

URL for the Policy Statement of this client, will be displayed to the user

Home Page

URL for the client's home page, will be displayed to the user

☒ Application Type ☐ Native ☐ Web

Contacts

[+](#)

[-](#)

List of contacts for administrators of this client.

[Save](#) [Cancel](#)

Powered by MITREid Connect © 2016 INFN

3. Save the protected resource.

RECAS IAM Test Instance for RECAS-BAR! admin

Home / Self-service Protected Resource Registration / rsreg.new

Save Cancel

Main Access Credentials JSON

Scope

new scope

openid ☒

profile ☒

email ☐

address ☐

phone ☐

offline_access ☒

Scopes that this resource will be able to introspect tokens for.

Save Cancel

Powered by MITREid Connect

© 2016 INFN

4. Save **Client ID**, **Client Secret** and **Registration Access Token** or the full output json in the **JSON** tab for future access.
5. Edit the protected resource configuration page as Administrator user, through the **ADMINISTRATIVE**, **Manage Clients**
6. Enable **Token exchange** and Check the flag at **Introspection**:

Introspection Allow calls to the Introspection Endpoint?

7. Navigate to the **Tokens** tab and set:

- Access Token Timeout: 7200
- ID Token Timeout: 7200

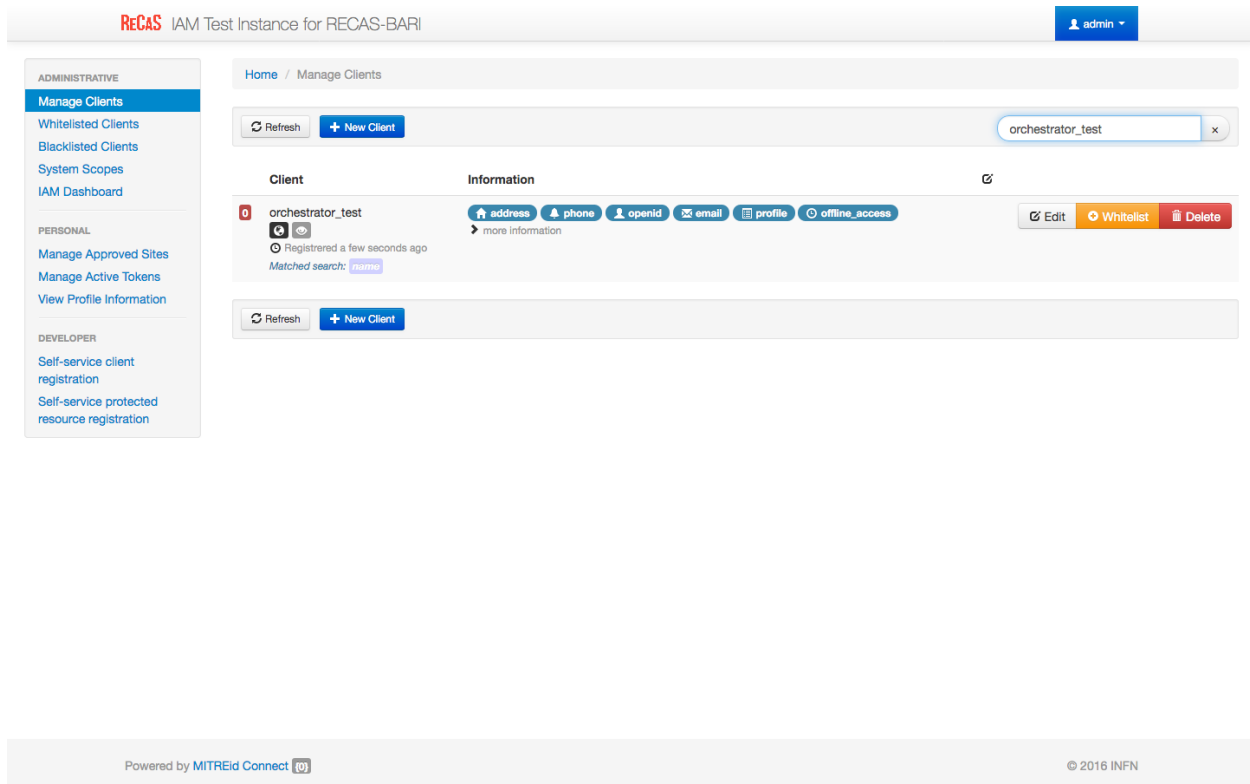
and flag:

- Refresh tokens are issued for this client
- Refresh tokens for this client are re-used
- Active access tokens are automatically revoked when the refresh token is used
- Refresh tokens do not time out

8. Save again the protected resource.

IAM protected resource configuration for CLUES

1. Login on IAM then **MitreID Dashboard** and select **Self-service protected resource registration** as Administrator user.



2. Select **New Resource** and set the following parameters

```
Name: clues_client
Scopes: openid, profile, email, address, phone, offline_access
```

3. Save the protected resource.
4. Save **Client ID**, **Client Secret** and **Registration Access Token** or the full output json in the **JSON** tab for future access.
5. Edit the protected resource configuration page as Administrator user, through the **ADMINISTRATIVE, Manage Clients**
6. Enable **Token exchange** and Check the flag at **Introspection**:
7. Navigate to the **Tokens** tab and set:
 - Access Token Timeout: 7200
 - ID Token Timeout: 7200
and flag:
 - Refresh tokens are issued for this client
 - Refresh tokens for this client are re-used
 - Active access tokens are automatically revoked when the refresh token is used
 - Refresh tokens do not time out
8. Save the protected resource again.

INDIGO IAM for laniakea

admin

ADMINISTRATIVE

Manage Clients

Whitelisted Clients

Blacklisted Clients

System Scopes

IAM Dashboard

PERSONAL

Manage Approved Sites

Manage Active Tokens

View Profile Information

DEVELOPER

Self-service client registration

Self-service protected resource registration

Home / Manage Clients / Edit Client

Save

Cancel

Main

Access

Credentials

Tokens

Crypto

Other

Scope

openid

☒

profile

☒

email

☐

address

☐

phone

☐

offline_access

☒

scim:read

☐

scim:write

☐

registration:read

☐

registration:write

☐

scim

☐

registration

☐

OAuth scopes this client is allowed to request

Grant Types

☐ authorization code

☐ client credentials

☐ password

☐ implicit

☐ redelegation

☐ device

☒ token exchange

Response Types

☐ code

☐ token

☐ id_token

☐ token id_token

☐ code id_token

☐ code token

☐ code token id_token

Introspection

Allow calls to the Introspection Endpoint? ☒

Subject Type

☐ Public ☐ Pairwise

Sector Identifier URI

Sector Identifier for JavaScript

Save

Cancel

Powered by MITREid Connect

© 2016 INFN

admin

ADMINISTRATIVE
Manage Clients
Whitelisted Clients
Blacklisted Clients
System Scopes
IAM Dashboard
PERSONAL
Manage Approved Sites
Manage Active Tokens
View Profile Information
DEVELOPER
Self-service client registration
Self-service protected resource registration

Home / Manage Clients / Edit Client

Edit Client

Save
Cancel

Main
Access
Credentials
Tokens
Crypto
Other

Access Token Timeout
☐ Access tokens do not time out

7200
seconds

Enter this time in seconds, minutes, or hours.

ID Token Timeout

7200
seconds

Enter this time in seconds, minutes, or hours.

Refresh Tokens
☒ Refresh tokens are issued for this client
This will add the offline_access scope to the client's scopes.

☒ Refresh tokens for this client are re-used
☒ Active access tokens are automatically revoked when the refresh token is used
☒ Refresh tokens do not time out

seconds

Enter this time in seconds, minutes, or hours.

Device Code Timeout

seconds

Enter this time in seconds, minutes, or hours.

Save
Cancel

Powered by MITREid Connect
© 2016 INFN

Orchestrator Installation

Create the file `indigopaas-deploy/ansible/inventory/group_vars/orchestrator.yaml` with the following configured values:

```
orchestrator_url: https://<proxy_dns_name>/orchestrator
orchestrator_image: indigodatacloud/orchestrator:2.1.2-final
orchestrator_mysql_root_password: *****
orchestrator_mysql_password: *****
orchestrator_im_url: https://<proxy_dns_name>/im
orchestrator_cmdb_url: https://<proxy_dns_name>/cmdb
orchestrator_slam_url: https://<slam_dns_name>:8443/rest/slam
orchestrator_cpr_url: https://<proxy_dns_name>/cpr
orchestrator_iam_issuer: https://<iam_dns_name>/
orchestrator_iam_client_id: <orchestrator_client_id>
orchestrator_iam_client_secret: <orchestrator_client_secret>
orchestrator_clues_iam_client_id: <clues_client_id>
orchestrator_clues_iam_client_secret: <clues_client_secret>
orchestrator_custom_types: https://raw.githubusercontent.com/Laniakea-elixir-it/indigopaas-resources/master/orchestrator/custom_types.yaml
disable_monitoring: True
```

Warning: SLAM and IAM are the only two services requiring a public IP, on the contrary all the others are behind the proxy.

The screenshot shows the RECAS IAM Test Instance for RECAS-BAR! interface. The top navigation bar includes the RECAS logo and the text "IAM Test Instance for RECAS-BAR!". A user menu in the top right shows "admin". The left sidebar contains a navigation menu with sections: ADMINISTRATIVE (Manage Clients, Whitelisted Clients, Blacklisted Clients, System Scopes, IAM Dashboard), PERSONAL (Manage Approved Sites, Manage Active Tokens, View Profile Information), and DEVELOPER (Self-service client registration, Self-service protected resource registration). The main content area shows the "Self-service Protected Resource Registration" page for "rsreg.new". It has tabs for Main, Access, Credentials, and JSON. The "Access" tab is active, showing a "Scope" section with a text input "new scope" and a list of scopes: openid, profile, email, address, phone, and offline_access, each with a checkbox. Below the list, it says "Scopes that this resource will be able to introspect tokens for." There are "Save" and "Cancel" buttons at the top and bottom of the form.

Warning: In this guide we avoid monitoring installation, leaving this job to the Cloud provider.

Run the role using the `ansible-playbook` command:

```
# cd indigopaas-deploy/ansible
# ansible-playbook -i inventory/inventory playbooks/deploy-orchestrator.yml
```

Video tutorial

FAQ

INDIGO PaaS Orchestrator

Orchent: the orchestrator CLI tool

Orchent is the indigo command line client.

Orchent: <https://github.com/indigo-dc/orchent>

INDIGO CLUES

CLUES is an elasticity manager system for HPC clusters and Cloud infrastructures that features the ability to power on/deploy working nodes as needed (depending on the job workload of the cluster) and to power off/terminate them when they are no longer needed.

ADMINISTRATIVE

Manage Clients

Whitelisted Clients

Blacklisted Clients

System Scopes

IAM Dashboard

PERSONAL

Manage Approved Sites

Manage Active Tokens

View Profile Information

DEVELOPER

Self-service client registration

Self-service protected resource registration

INDIGO IAM for laniakea

admin

Home / Manage Clients / Edit Client

Save Cancel

h1>Edit Client

MainAccessCredentialsTokensCryptoOther

Scope

new scope

openid

profile

email

address

phone

offline_access

scim:read

scim:write

registration:read

registration:write

scim

registration

OAuth scopes this client is allowed to request

Grant Types

authorization code

client credentials

password

implicit

redelegation

device

token exchange

Response Types

code

token

id_token

token id_token

code id_token

code token

code token id_token

Introspection

Allow calls to the Introspection Endpoint?

Subject Type

Public

Pairwise

Sector Identifier URI

https://

Sector Identifier for JavaScript

Save Cancel

Powered by MITREid Connect

© 2016 INFN

26.2. Service installation

209

admin

ADMINISTRATIVE
Manage Clients
Whitelisted Clients
Blacklisted Clients
System Scopes
IAM Dashboard
PERSONAL
Manage Approved Sites
Manage Active Tokens
View Profile Information
DEVELOPER
Self-service client registration
Self-service protected resource registration

Home / Manage Clients / Edit Client

Edit Client

Save
Cancel

Main
Access
Credentials
Tokens
Crypto
Other

Access Token Timeout
☐ Access tokens do not time out

7200
seconds

Enter this time in seconds, minutes, or hours.

ID Token Timeout

7200
seconds

Enter this time in seconds, minutes, or hours.

Refresh Tokens
☒ Refresh tokens are issued for this client
This will add the offline_access scope to the client's scopes.

☒ Refresh tokens for this client are re-used
☒ Active access tokens are automatically revoked when the refresh token is used
☒ Refresh tokens do not time out

seconds

Enter this time in seconds, minutes, or hours.

Device Code Timeout

seconds

Enter this time in seconds, minutes, or hours.

Save
Cancel

Powered by MITREid Connect
© 2016 INFN

Official GitBook documentation: <https://www.gitbook.com/book/indigo-dc/clues-indigo/details>

Check worker nodes status

To check worker node status:

```
# sudo clues status
node                                state  enabled  time stable  (cpu,mem) used  (cpu,
↪mem) total
-----
↪-----
vnode-1                             powon   enabled   00h02'54"    0,0.0           1,
↪1073741824.0
vnode-2                             off     enabled   00h41'00"    0,0.0           1,
↪1073741824.0
```

CLUES commands:

```
# clues --help
The CLUES command line utility

Usage: clues [-h]
↪ [status|resetstate|enable|disable|poweron|poweroff|nodeinfo|shownode|req_create|req_
↪wait|req_get]

[-h|--help] - Shows this help
```

(continues on next page)

(continued from previous page)

```

* Show the status of the platform
Usage: status

* Reset the state of one or more nodes to idle
Usage: resetstate <nodes>
<nodes> - names of the nodes whose state want to be reset

* Enable one or more nodes to be considered by the platform
Usage: enable <nodes>
<nodes> - names of the nodes that want to be enabled

* Disable one or more nodes to be considered by CLUES
Usage: disable <nodes>
<nodes> - names of the nodes that want to be disabled

* Power on one or more nodes
Usage: poweron <nodes>
<nodes> - names of the nodes that want to be powered on

* Power off one or more nodes
Usage: poweroff <nodes>
<nodes> - names of the nodes that want to be powered off

* Show the information about node(s), to be processed in a programmatically mode
Usage: nodeinfo [-x] <nodes>
[-x|--xml] - shows the information in XML format
<nodes> - names of the nodes whose information is wanted to be shown

* Show the information about node(s) as human readable
Usage: shownode <nodes>
<nodes> - names of the nodes whose information is wanted to be shown

* Create one request for resources
Usage: req_create --cpu <value> --memory <value> [--request <value>] [--count <value>]
--cpu|-c <value> - Requested CPU
--memory|-m <value> - Requested Memory
[--request|-r] <value> - Requested constraints for the nodes
[--count|-n] <value> - Number of resources (default is 1)

* Wait for a request
Usage: req_wait <id> [timeout]
<id> - Identifier of the request to wait
[timeout] - Timeout to wait

* Get the requests in a platform
Usage: req_get

```

Check worker nodes deployment

Worker node deployment log are available to: /var/log/clues2/clues2.log

Troubleshooting

Invalid Token

Symptoms: Galaxy jobs stuck in This job is waiting to run and stay gray in the Galaxy history.

The worker nodes are not correctly instantiated, due to an Invalid Token. Check `/var/log/clues2/clues2.log`:

```
urllib3.connectionpool;DEBUG;2017-10-31 10:52:33,288;"GET /orchestrator/deployments/
48126bd4-14d8-494d-970b-fb581a3e13b2/resources?size=20&page=0 HTTP/1.1" 401 None
[PLUGIN-INDIGO-ORCHESTRATOR];ERROR;2017-10-31 10:52:33,291;ERROR getting deployment
info: {"code":401,"title":"Unauthorized","message":"Invalid token:
eyJJraWQioiJyc2ExIiwiaWxnbGJvc3VybmVkeSI6ImMyNTYifQ.
eyJJzdWIioiIj3RU4Qjg4MC1DNEQwLTQ2RkEtQjQxMS0wQTtCREl3OUYzOTYiLCJpc3MiOiJodHRwczpcL1l
eQyJzVs0h5kuqoBZQf5PPcYrsRJsTFyZ05Zpx8xPcfjrUWHwwOnw9knQg8Ex3lwAxi5qxdmgBDi4EIZAG
M_btm4nTbUvTSAUafjki4lDnPoEjLqXTTy8XLPURcSMHvEqvSHHFipeSkp90xK1tlUadPc"} }
```

Solution:

1. Stop CLUES: `sudo systemctl stop cluesd`.
2. Edit the file `/etc/clues2/conf.d/plugin-ec3.cfg` and change the value of the `INDIGO_ORCHESTRATOR_AUTH_DATA` parameter with the new token.
3. Restart CLUES `sudo systemctl start cluesd`.
4. You also have to open the CLUES DB with `sqlite3` command: `sqlite3 /var/lib/clues2/clues.db` and delete old refreshed token: `DELETE FROM orchestrator_token;`. To exit from `sqlite` just type: `.exit`.

26.2.8 Hashicorp Vault

Vault is exploited as secrets management store, to store and manage encryption passphrases

Note: Current version: 1.1.2

VM configuration

Create a VM for Vault. The VM should meet the following minimum requirements:

OS	Ubuntu 16.04
vCPUs	2
RAM	4 GB
Network	Public IP address.

Warning: All the command will be run from the control machine VM.

Installation

Create the file `indigopaas-deploy/ansible/inventory/group_vars/vault.yaml` with the following configured values:

```
vault_fqdn: <dashboard_vm_dns_name>
vault_image_name: vault:1.1.2
vault_letsencrypt_email: "<valid_email_address>"
```

Warning: Depending on your Cloud Provider network configuration, the `vault_host` variable needs to be added and configured with the private ip address associated to the VM, for example when a floating IP is used.

In this case it is possible to set the IP address adding:

```
vault_host: '<vm_private_ip_address>'
```

Run the role using the `ansible-playbook` command:

```
# cd indigopaas-deploy/ansible
# ansible-playbook -i inventory/inventory playbooks/deploy-vault.yml
```

Installation video tutorial

Vault initialization

The Vault initialization can not be automated. To initialize it and get your root token for the initial configuration

1. Login on the VM hosting Vault:

```
ssh root@<vault_vm_ip_address>
```

2. Initialize Vault:

```
# docker exec -it vault vault operator init
Unseal Key 1: p7YF7vyLRrfeilwld/QusQ+UESJiGrhn1TwCsBAa7fKV
Unseal Key 2: OHoyPApMFuQTz9B20bmpJjzLgkCi2ELr+zKFdvKq8lmL
Unseal Key 3: xDRcbkOsYL9uswFzCdFqpxudgvZfVfAwFCkigYMMMCHt
Unseal Key 4: LJ0hHW5dsmbuFAnL+W/4NMtZUbuNkILFWXxL3zTYblzQ
Unseal Key 5: ZlOvJ7RvT+pUVtqB93RAQ8q1s8l04clGVFn+oi22x4rZ

Initial Root Token: s.YxsTl9H3f1qgAqH3cj4JAXR8

Vault initialized with 5 key shares and a key threshold of 3. Please securely
distribute the key shares printed above. When the Vault is re-sealed,
restarted, or stopped, you must supply at least 3 of these keys to unseal it
before it can start servicing requests.

Vault does not store the generated master key. Without at least 3 key to
reconstruct the master key, Vault will remain permanently sealed!

It is possible to generate new unseal keys, provided you have a quorum of
existing unseal keys shares. See "vault operator rekey" for more information.
```

3. Every initialized Vault server starts in the `sealed state`. Unsealing has to happen every time Vault starts. It can be done via the API and via the command line. To unseal the Vault, you must have the threshold number of unseal keys. In the output above, notice that the “key threshold” is 3. This means that to unseal the Vault, you need 3 of the 5 keys that were generated.

```
# docker exec -it vault vault operator unseal p7YF7vyLRrfeilw1D/
↪ QusQ+UESJiGrhn1TwCsBAa7fKV
```

Key	Value
---	-----
Seal Type	shamir
Initialized	true
Sealed	true
Total Shares	5
Threshold	3
Unseal Progress	1/3
Unseal Nonce	7a0891bb-7d0e-6efa-2081-9c60941f9a6d
Version	1.1.2
HA Enabled	false

```
# docker exec -it vault vault operator unseal_
↪ OHoyPAPmFuQTz9B20bmpJjzLgkCi2ELr+zKFdvKq8lmL
```

Key	Value
---	-----
Seal Type	shamir
Initialized	true
Sealed	true
Total Shares	5
Threshold	3
Unseal Progress	2/3
Unseal Nonce	7a0891bb-7d0e-6efa-2081-9c60941f9a6d
Version	1.1.2
HA Enabled	false

```
# docker exec -it vault vault operator unseal_
↪ xDRcbkOsYL9uswFzCdFqpxudgvZFVfAwFCKigYMMMCHt
```

Key	Value
---	-----
Seal Type	shamir
Initialized	true
Sealed	false
Total Shares	5
Threshold	3
Version	1.1.2
Cluster Name	vault-cluster-e6688ec2
Cluster ID	ccf2e852-69ca-bcd6-0079-6c820f9c0e67
HA Enabled	false

4. Finally, authenticate as the initial root token (it was included in the output with the unseal keys):

```
# docker exec -it vault vault login s.YxsTl9H3f1qgAqH3cj4JAXR8
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.
```

Key	Value
---	-----
token	s.YxsTl9H3f1qgAqH3cj4JAXR8
token_accessor	QEUBU4tepPWDatRu6jrnTbFW
token_duration	∞
token_renewable	false
token_policies	["root"]
identity_policies	[]
policies	["root"]

Warning: Save the unseal keys and the root token. Please read [Vault documentation](#).

Initialization video tutorial

References

[Vault documentation](#)

26.2.9 Laniakea Dashboard

The Laniakea Dashboard is built on top of the INDIGO Orchestrator Dashboard.

Note: Current Dashboard version: stable version

VM configuration

Create VM for Dashboard. The VM should meet the following minimum requirements:

OS	Ubuntu 16.04
vCPUs	2
RAM	4 GB
Network	Public IP address.

Warning: In this tutorial we will use the same VM for vault and the dashboard, being the two services strictly connected.

This is not required.

Warning: All the commands will be run from the control machine VM.

IAM client configuration

1. Login on IAM as Administrator User.
2. Navigate to **MitreID Dashboard** and select from the left panel **Self-service client registration**.
3. Create a **New client** and fill the form with the following parameters

```
Client name = dashboard_client
redirect URI(s) = https://<dashboard_vm_dns_name>/login/iam/authorized
```

4. In the Access tab select the following **Scopes**

RECAS

IAM Test Instance for RECAS-BARI

admin

ADMINISTRATIVE

Manage Clients

Whitelisted Clients

Blacklisted Clients

System Scopes

IAM Dashboard

PERSONAL

Manage Approved Sites

Manage Active Tokens

View Profile Information

DEVELOPER

Self-service client registration

Self-service protected resource registration

Home / Self-service Client Registration / Register a new client

New Client

Save

Cancel

Main

Access

Credentials

Crypto

Other

JSON

Client ID

Will be generated by the server when the client is saved

Client Secret

Will be generated by the server when the client is saved

Client Configuration URL

Will be generated by the server when the client is saved

Registration Access Token

Will be generated by the server when the client is saved

Client name

dashboard_test

Human-readable application name

Redirect URI(s)

https://

https://cloud-90-147-170-32.cloud.ba.infn.it/login/iam/authorized

URIs that the client can be redirected to after the authorization page

Logo

https://

URL that points to a logo image, will be displayed on approval page

Enter a logo URL

Terms of Service

https://

URL for the Terms of Service of this client, will be displayed to the user

Policy Statement

https://

URL for the Policy Statement of this client, will be displayed to the user

Home Page

https://

URL for the client's home page, will be displayed to the user

Software ID

software ID...

Identifier for the software in this client

Software Version

1.0...

Version of the software in this client

Contacts

List of contacts for administrators of this client.

new contact

admin@iam.test

List of contacts for administrators of this client.

Software Statement

eyJ0...

A software statement is issued by a trusted third party and locks certain elements of a client's registration

Save

Cancel

Powered by MITREid Connect

© 2016 INFN

```
Scopes: openid, profile, email, address, phone, offline_access
```

and for **Grant Types** select:

```
Grant types: authorization code
```

RECAS IAM Test Instance for RECAS-BARI admin

Home / Self-service Client Registration / Register a new client

New Client

Save Cancel

Main Access **Credentials** Crypto Other JSON

Scope +

- openid ☒
- profile ☒
- email ☒
- address ☒
- phone ☒
- offline_access ☒

OAuth scopes this client is allowed to request

Grant Types ☒ authorization code

- ☐ client credentials
- ☐ implicit
- ☐ password
- ☐ redelegation
- ☐ refresh
- ☐ device
- ☐ token exchange

ANY Response Types ☒ code

- ☐ token
- ☐ id_token
- ☐ token id_token
- ☐ code id_token
- ☐ code token
- ☐ code token id_token

Subject Type ☒ Public ☐ Pairwise

Sector Identifier URI

Sector Identifier for JavaScript

Save Cancel

Powered by MITREid Connect © 2016 INFN

5. Save.

6. Save **Client ID**, **Client Secret** and **Registration Access Token** or the full output json in the **JSON** tab for future access.

Installation

The Laniakea dashboard can be installed in three different ways: Stateless, with MySQL database and with MySQL and Vault integration.

The one with MySQL and Hashicorp Vault is the one used in Laniakea.

Install Laniakea dashboard (database and vault version)

Warning: Vault integration leverages on MySQL database. It can't work with dashboard stateless version

Update the dashboard IAM client configuration

To enable Vault integration the **token exchange** is needed. Therefore, edit the IAM client previously created for the dashboard.

Enable **token exchange** accessing to the client configuration page as Administrator user, through the **ADMINISTRATIVE, Manage Clients** and check the flag `token exchange` in the `Grant types` section.

IAM client configuration for Vault

Create another IAM client for Vault, to enable oidc integration to authenticate users.

1. Login on IAM then **MitreID Dashboard** and select **Self-service client registration** as Administrator user.
2. Click on **New client** with the following parameters:

```
Client name: vault_client

redirect URI(s): https://<dashboard_vm_dns_name>:8200/ui/vault/auth/oidc/oidc/
↪callback
                https://<dashboard_vm_dns_name>:8250/oidc/callback
```

3. In the Access tab select the following **Scopes**

```
Scopes: openid, profile, email, address, phone, offline_access
```

4. Save the client.
5. Save **Client ID**, **Client Secret** and **Registration Access Token** or the full output json in the **JSON** tab for future access.

Installation

Create the file `indigopaas-deploy/ansible/inventory/group_vars/orchestrator-dashboard.yaml` with the following configured values:

```
dashboard_fqdn: <dashboard_vm_dns_name>
dashboard_image_name: laniakeacloud/laniakea-dashboard

dashboard_iam_issuer: "https://<iam_address>/"
dashboard_iam_client_id: "<im_client_id>"
dashboard_iam_client_secret: "<iam_client_secret>"
dashboard_orchestrator_url: "https://<proxy_vm_dns_name>/orchestrator"
dashboard_slam_url: "https://<slam_vm_dns_name>:8443"
dashboard_cmdb_url: "https://<proxy_vm_dns_name>/cmdb"
dashboard_im_url: "https://<proxy_vm_dns_name>/im"

dashboard_tosca_template_repository_url: https://github.com/Laniakea-elixir-it/
↪laniakea-dashboard-config.git
```

(continues on next page)

RECAS
IAM Test Instance for RECAS-BARI
admin

ADMINISTRATIVE
Manage Clients
Whitelisted Clients
Blacklisted Clients
System Scopes
IAM Dashboard
PERSONAL
Manage Approved Sites
Manage Active Tokens
View Profile Information
DEVELOPER
Self-service client registration
Self-service protected resource registration

Home / Manage Clients / Edit Client

Edit Client

Save
Cancel

Main
Access
Credentials
Tokens
Crypto
Other

Scope

new scope

openid
☒
profile
☒
email
☒
address
☒
phone
☒
offline_access
☒
scim:read
☐
scim:write
☐
registration:read
☐
registration:write
☐
scim
☐
registration
☐

OAuth scopes this client is allowed to request

Grant Types
☒ authorization code
☐ client credentials
☐ password
☐ implicit
☐ redelegation
☐ device
☒ token exchange

Response Types
☒ code
☐ token
☐ id_token
☐ token id_token
☐ code id_token
☐ code token
☐ code token id_token

Introspection
Allow calls to the Introspection Endpoint? ☐

Subject Type
☐ Public ☐ Pairwise


Sector Identifier URI

https://

Sector Identifier for JavaScript

Save
Cancel

Powered by MITREid Connect
© 2016 INFN

Powered by MITREid Connect  © 2016 INFN

RECAS IAM Test Instance for RECAS-BARI admin

ADMINISTRATIVE

Manage Clients

Whitelisted Clients

Blacklisted Clients

System Scopes

IAM Dashboard

PERSONAL

Manage Approved Sites

Manage Active Tokens

View Profile Information

DEVELOPER

Self-service client registration

Self-service protected resource registration

Home / Self-service Client Registration / Register a new client

New Client

Save Cancel

Main Access Credentials Crypto Other JSON

Scope

new scope

openid

profile

email

address

phone

offline_access

OAuth scopes this client is allowed to request

Grant Types

authorization code

client credentials

implicit

password

redelegation

refresh

device

token exchange

Response Types

code

token

id_token

token id_token

code id_token

code token

code token id_token

Subject Type

Public

Pairwise

Sector Identifier URI

https://

Sector Identifier for JavaScript

Save Cancel

Powered by MITREid Connect © 2016 INFN

(continued from previous page)

```
dashboard_tosca_template_repository_dir: "/opt/laniakea-dashboard-config"
dashboard_tosca_templates_dir: "/opt/laniakea-dashboard-config/tosca-templates"
dashboard_tosca_parameters_dir: "/opt/laniakea-dashboard-config/tosca-parameters"
dashboard_tosca_metadata_dir: "/opt/laniakea-dashboard-config/tosca-metadata"
dashboard_administrators: "['<valid_email_address>']"
dashboard_support_email: "['<valid_email_address>']"

dashboard_letsencrypt_email: "<valid_email_address>"

dashboard_enable_db: True
dashboard_db_sql_file_url: "https://raw.githubusercontent.com/Laniakea-elixir-it/
↳ orchestrator-dashboard/laniakea-stable/utils/orchestrator_dashboard.sql"
dashboard_mysql_root_password: *****
dashboard_db_password: *****

dashboard_enable_vault: True
dashboard_vault_token: "<vault_valid_token>"
dashboard_vault_iam_client_id: "vault_iam_client_id"
dashboard_vault_iam_client_secret: "<vault_iam_client_secret>"
```

Warning: Depending on your Cloud Provider network configuration, the database IP address needs to be further configured, for example using the private ip address associated to the VM, when a floating IP is used.

In this case it is possible to set the database IP address adding:

```
dashboard_db_host: '<vm_private_ip_address>'
```

Warning: Set also your custom mysql password with: `dashboard_mysql_root_password` and `dashboard_mysql_password`.

Note: A valid token to create policies and enable OIDC authentication on vault is needed. Here, for simplicity we use the root token gathered in the Vault installation section [Hashicorp Vault](#).

Run the role using the `ansible-playbook` command:

```
# cd indigopaas-deploy/ansible
# ansible-playbook -i inventory/inventory playbooks/deploy-orchestrator-dashboard.yml
```

Video Tutorial

Post installation steps to enable the callback

If the callback is enabled, the PaaS Orchestrator (*PaaS Orchestrator*) needs to be configured accordingly.

In particular, the dashboard CA certificate has to be copied on the PaaS Orchestrator Virtual Machine in `/etc/orchestrator/trusted_certs`.

For Let's Encrypt certificates, those used in this wiki:

1. Connect through SSH to the Dashboard VM and copy the content of the file `/etc/letsencrypt/live/<orchestrator_dashboard_dns_name>/chain.pem`.
2. Connect through SSH to the PaaS Orchestrator VM and paste the `chain.pem` to `/etc/orchestrator/trusted_certs/dashboard-cert.pem`
3. Restart the PaaS Orchestrator with:

```
# docker restart orchestrator
```

4. Once the Orchestrator is started the chain file can be removed:

```
# rm /etc/orchestrator/trusted_certs/dashboard-cert.pem
```

Appendix A. Stateless version

This is a simple graphical User interface of the INDIGO PaaS orchestrator. The automated storage encryption will not work.

Install Laniakea dashboard (stateless version)

Create the file `indigopaas-deploy/ansible/inventory/group_vars/orchestrator-dashboard.yaml` with the following configured values:

```
dashboard_fqdn: <dashboard_vm_dns_name>
dashboard_image_name: indigodatacloud/orchestrator-dashboard

dashboard_iam_issuer: "https://<iam_address>/"
dashboard_iam_client_id: "<im_client_id>"
dashboard_iam_client_secret: "<iam_client_secret>"
dashboard_orchestrator_url: "https://<proxy_vm_dns_name>/orchestrator"
dashboard_slam_url: "https://<slam_vm_dns_name>:8443"
dashboard_cmdb_url: "https://<proxy_vm_dns_name>/cmdb"
dashboard_im_url: "https://<proxy_vm_dns_name>/im"

dashboard_tosca_template_repository_url: https://github.com/Laniakea-elixir-it/
↳ laniakea-dashboard-config.git
dashboard_tosca_template_repository_dir: "/opt/laniakea-dashboard-config"
dashboard_tosca_templates_dir: "/opt/laniakea-dashboard-config/tosca-templates"
dashboard_tosca_parameters_dir: "/opt/laniakea-dashboard-config/tosca-parameters"
dashboard_tosca_metadata_dir: "/opt/laniakea-dashboard-config/tosca-metadata"
dashboard_administrators: "['<valid_email_address>']"

dashboard_letsencrypt_email: "<valid_email_address>"
```

Run the role using the `ansible-playbook` command:

```
# cd indigopaas-deploy/ansible

# ansible-playbook -i inventory/inventory playbooks/deploy-orchestrator-dashboard.yaml
```

Appendix B. Database version

This version comes with a MySQL database support.

Install Laniakea dashboard (database version)

Create the file `indigopaas-deploy/ansible/inventory/group_vars/orchestrator-dashboard.yml` with the following configured values:

```
dashboard_fqdn: <dashboard_vm_dns_name>
dashboard_image_name: laniakeacloud/laniakea-dashboard:withDB

dashboard_iam_issuer: "https://<iam_address>/"
dashboard_iam_client_id: "<im_client_id>"
dashboard_iam_client_secret: "<iam_client_secret>"
dashboard_orchestrator_url: "https://<proxy_vm_dns_name>/orchestrator"
dashboard_slam_url: "https://<slam_vm_dns_name>:8443"
dashboard_cmdb_url: "https://<proxy_vm_dns_name>/cmdb"
dashboard_im_url: "https://<proxy_vm_dns_name>/im"

dashboard_tosca_template_repository_url: https://github.com/Laniakea-elixir-it/
↳ laniakea-dashboard-config.git
dashboard_tosca_template_repository_dir: "/opt/laniakea-dashboard-config"
dashboard_tosca_templates_dir: "/opt/laniakea-dashboard-config/tosca-templates"
dashboard_tosca_parameters_dir: "/opt/laniakea-dashboard-config/tosca-parameters"
dashboard_tosca_metadata_dir: "/opt/laniakea-dashboard-config/tosca-metadata"
dashboard_administrators: "['<valid_email_address>']"

dashboard_letsencrypt_email: "<valid_email_address>"

dashboard_enable_db: True
dashboard_db_sql_file_url: "https://raw.githubusercontent.com/Laniakea-elixir-it/
↳ orchestrator-dashboard/laniakea-stable/utils/orchestrator_dashboard.sql"
dashboard_mysql_root_password: *****
dashboard_db_password: *****
```

Warning: Depending on your Cloud Provider network configuration, the database IP address needs to be further configured, for example using the private ip address associated to the VM, when a floating IP is used.

In this case it is possible to set the database IP address adding:

```
dashboard_db_host: '<vm_private_ip_address>'
```

Warning: Set also your custom mysql password with: `dashboard_mysql_root_password` and `dashboard_mysql_password`.

Run the role using the `ansible-playbook` command:

```
# cd indigopaas-deploy/ansible
# ansible-playbook -i inventory/inventory playbooks/deploy-orchestrator-dashboard.yml
```

26.2.10 The last mile: applications configuration

By default, Laniakea is configured to run the following applications:

Galaxy live build

Description The Galaxy live build allows to setup and launch a virtual machine configured with the Operative System CentOS 7 and the auxiliary applications needed to support a Galaxy production environment such as PostgreSQL, Nginx, uWSGI and Proftpd and to deploy the Galaxy platform itself and the tools that come with the selected flavour.

This application can be deployed with cluster support, using SLURM as Resource Manager and with automatica elasticity support, with CLUES as elasticity manager.

Recommended images [CentOS-7-x86_64-GenericCloud-1907.qcow2](#)

Configuration galaxy_latest

Galaxy express

Description The Galaxy express instantiate a CentOS 7 Virtual Machine with Galaxy, all its companion software and the set of tools that come with the selected flavour. Once deployed each Galaxy instance can be further customized with additional tools and reference data.

This application can be deployed with cluster support, using SLURM as Resource Manager.

The default available flavours currently are:

- galaxy-minimal: Galaxy production-grade server (Galaxy, PostgreSQL, NGINX, proFTPD, uWSGI).
- galaxy-CoVaCS: workflow for genotyping and variant annotation of whole genome/exome and target-gene sequencing data (<https://www.ncbi.nlm.nih.gov/pubmed/29402227>).
- galaxy-GDC_Somatic_Variant: port of the Genomic Data Commons (GDC) pipeline for the identification of somatic variants on whole exome/genome sequencing data (<https://gdc.cancer.gov/node/246>).
- galaxy-rna-workbench: more than 50 tools for RNA centric analysis (<https://www.ncbi.nlm.nih.gov/pubmed/28582575>).
- galaxy-epigen: based on Epigen project (<http://www.epigen.it/>).

More information on Laniakea default Galaxy flavours can be found here: *Galaxy Flavours*.

Configuration galaxy_vm

Galaxy Docker

Description The Galaxy Docker instantiate an Ubuntu 16.04 Virtual Machine with the Galaxy official Docker. Once deployed each Galaxy instance can be further customized with additional tools and reference data.

Recommended images [Ubuntu 16.04 LTS cloud images](#)

Configuration galaxy_docker

Test applications

Description Two test recipes are shipped by default to test a simple Ubuntu or Centos deployment with or without storage volume

Recommended images CentOS-7-x86_64-GenericCloud-1907.qcow2 or Ubuntu 16.04 LTS cloud images

Configuration test_deployments

26.2.11 Updating Laniakea

The same ansible roles used to deploy Laniakea can be used to keep it up to date.

1. Update the indigopaas-deploy ansible roles:

```
cd indigopaas-deploy
git pull
```

1. All the services run inside docker container. Therefore, in most of cases, service update requires to re-create the Docker container with the updated image. The corresponding data are mounted inside the Docker container, thus avoiding any data loss during the update procedure.

The services docker images can be changed in the corresponding configuration file in `indigopaas-deploy/ansible/inventory/group_vars/<service>.yaml`.

2. Finally to update a service, just re-run the ansible role:

```
# cd indigopaas-deploy/ansible
# ansible-playbook -i inventory/inventory playbooks/deploy-<service>.yaml
```

Warning: INDIGO Software catalogue is actively developed. So the update procedure of Laniakea depends on the INDIGO services evolution. We will keep this page updated accordingly.

Note: All the (Galaxy) instances deployed with Laniakea are not influenced by the update procedure.

Current recommended configuration

Currently, the following versions of the INDIGO services are recommended:

Service	Version	Docker image
indigopaas-deploy	v1.0	—
IAM	1.5 rc2	indigoiam/iam-login-service:v1.5.0.rc2-SNAPSHOT-latest
IM	1.8.8.1	indigodatacloud/im:1.8.6.1
CMDB	indigo_2	indigodatacloud/cmdb:indigo_2
CPR	indigo_2	indigodatacloud/cloudproviderranker:indigo_2
SLAM	v2.0.0	indigodatacloud/slam:v2.0.0
Custom types	v3.0.1	—
Orchestrator	2.1.2-final	indigodatacloud/orchestrator:2.1.2-final
Vault	1.1.2	vault:1.1.2
Dashboard	laniakea-stable	laniakeacloud/laniakea-dashboard:stable

CHAPTER 27

GitHub repository

<https://github.com/Laniakea-elixir-it>

CHAPTER 28

DockerHub repository

<https://hub.docker.com/r/laniakeacloud>

CHAPTER 29

Support

If you need support please contact us to: laniakea.helpdesk@gmail.com

Software glitches and bugs can occasionally be encountered. The best way to report a bug is to open an issue on our [GitHub repository](#).

Cite

Marco Antonio Tangaro, Giacinto Donvito, Marica Antonacci, Matteo Chiara, Pietro Mandreoli, Graziano Pesole, Federico Zambelli, Laniakea: an open solution to provide Galaxy “on-demand” instances over heterogeneous cloud infrastructures, GigaScience, Volume 9, Issue 4, April 2020, giaa033, <https://doi.org/10.1093/gigascience/giaa033>

The paper is available [here](#).

CHAPTER 31

Licence

As an open source project Laniakea is made up of many pieces of software created by a range of individuals, teams, and companies. Laniakea is a collective work, and each piece of software within this work has its own license.

Your use of each piece of software is governed by the terms of its accompanying license. Redistribution of parts or the whole of Laniakea may require you to comply with additional license requirements.

CHAPTER 32

Galaxy tutorials

Galaxy training network: <https://galaxyproject.org/teach/gtn/>

Galaxy For Developers: <https://crs4.github.io/Galaxy4Developers/>

CHAPTER 33

Indices and tables

- `genindex`
- `modindex`
- `search`