
Laniakea Documentation

Release 0.1

Marco Antonio Tangaro

Aug 17, 2019

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	Get Galaxy Express	13
5.1	Instantiate Galaxy	13
5.2	Galaxy login	16
6	Get Galaxy	19
6.1	Instantiate Galaxy	19
6.2	Galaxy login	25
7	Get encrypted instance	27
7.1	Default encryption algorithm	30
8	Get Galaxy cluster	31
8.1	Instantiate Galaxy	31
8.2	Galaxy login	36
9	Create SSH Keys	39
9.1	How to create SSH keys on Linux or macOS	39
9.2	How to create SSH keys on Windows	39
10	Galaxy with Onedata	41
11	Virtual hardware presets	43
12	Laniakea options	45
12.1	job identifier	45
12.2	Virtual hardware configuration	45
12.3	Storage configuration	46
12.4	Galaxy configuration	46

12.5	Tools configuration	47
13	Galaxy production environment	49
13.1	OS support	49
13.2	PostgresSQL	49
13.3	NGINX	51
13.4	uWSGI	53
13.5	Proftpd	54
13.6	Supervisord	56
13.7	Galaxy init scripts	59
14	Galaxy instance isolation	61
14.1	Storage encryption	62
14.2	Storage encryption procedure	66
14.3	File System Encryption Test	68
14.4	Onedata (beta)	72
15	Galaxy ShedTools	75
15.1	Create and test Galaxy flavors	75
15.2	Conda support	76
15.3	References	77
16	Reference Data	79
16.1	Available reference data	79
16.2	CernVM-FS reference data	80
16.3	Onedata reference data (beta)	82
16.4	Reference data local download	83
17	Cluster support (SLURM)	85
17.1	Shared file system	85
17.2	Nodes deployment	86
17.3	SLURM main commands	86
18	Authentication	87
18.1	Registration	88
18.2	Login	88
19	Validation	95
20	Galaxyctl libraries	97
20.1	Dependencies	97
20.2	DetectGalaxyCommands	97
20.3	UwsgiSocket	98
20.4	UwsgiStatsServer	98
20.5	LUKSCtl	98
20.6	OneDataCtl	99
21	Galaxyctl: Galaxy management	101
21.1	Galaxyctl basic usage	103
21.2	Logging	103
21.3	Advanced options	103
21.4	LUKS module	104
21.5	Onedata module	104
21.6	Configuration files	105

22	Luksctl: LUKS volumes management	107
22.1	Dependencies	107
22.2	Open LUKS volumes	107
22.3	Close LUKS volumes	108
22.4	LUKS volumes status	108
23	Onedactl: Onedata spaces management	109
23.1	Options	109
23.2	The onedactl.ini file	109
23.3	Onedactl options	110
23.4	Oneclient	111
23.5	Troubleshooting	111
24	Frequently Asked Questions	113
24.1	Recover Galaxy after Virtual Machine reboot	113
25	Galaxycloud Ansible Roles	117
25.1	Ansible roles documentation	117
26	TOSCA templates	139
26.1	Custom types	139
26.2	Galaxy template	146
26.3	Galaxy cluster template	150
27	Build cvmfs server for reference data	153
27.1	Create CernVM-FS Repository	153
27.2	Client configuration	154
27.3	Populate a CernVM-FS Repository (with reference data)	154
27.4	Reference data download	155
27.5	References	156
28	INDIGO PaaS Orchestrator	157
28.1	Orchent	157
29	Infrastructure Manager	159
29.1	Deployments log	159
30	INDIGO FutureGateway	161
30.1	Portal configuration	162
30.2	Apache configuration	162
30.3	IAM integration	163
30.4	Administrator portlet	163
30.5	Portlet configuration	167
30.6	Update to Java 8 - Appendix A	167
30.7	Configure Apache for http - Appendix B	168
30.8	Import Signed CA - Appendix C	168
30.9	Import Signed CA in Java keystore - Appendix D	168
30.10	Create https certificate - Appendix E	169
30.11	Fix ghost deployment issue	169
30.12	Logs	170
30.13	References	170
31	ONEDATA	171
32	INDIGO CLUES	173
32.1	Check worker nodes status	173

32.2	Check worker nodes deployment	174
32.3	Troubleshooting	175
33	GitHub repository	177
34	DockerHub repository	179
35	Support	181
36	Cite	183
37	Licence	185
38	Galaxy tutorials	187
39	Indices and tables	189



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 FutureGateway* portal.

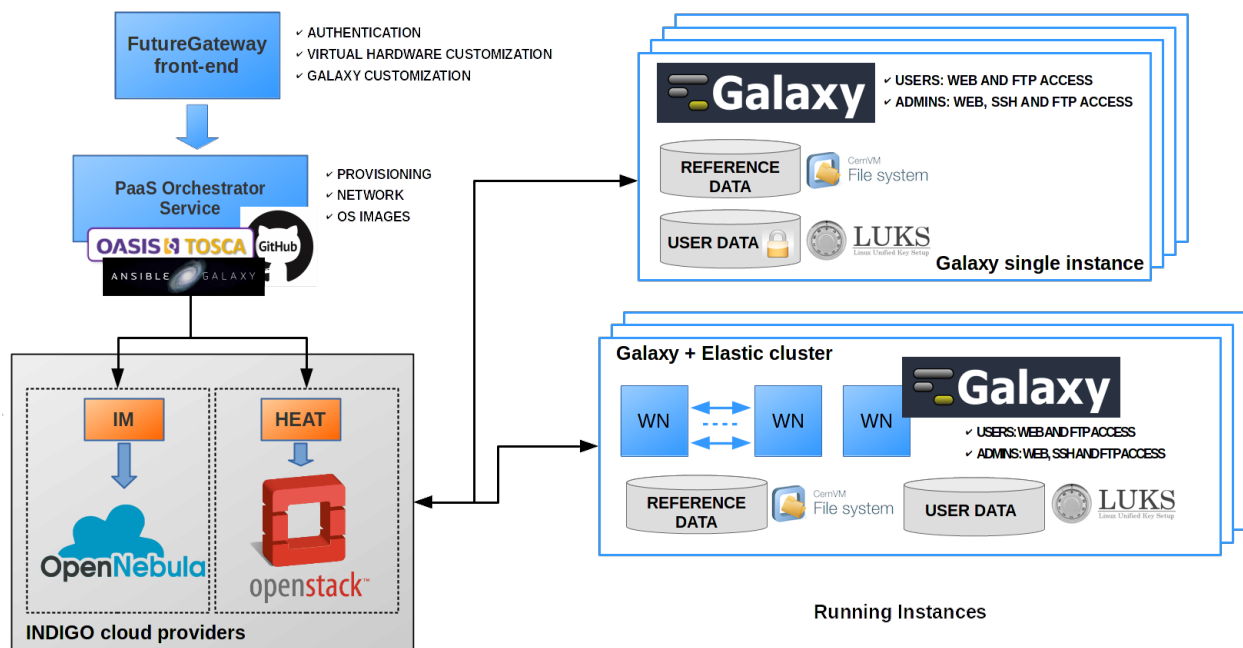


Fig. 1: Galaxy as a Cloud Service architecture

All the required components to automatically setup Galaxy instances (Galaxy and all its companion software) are deployed using the *INDIGO PaaS Orchestrator* and the *Infrastructure Manager* 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 *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.

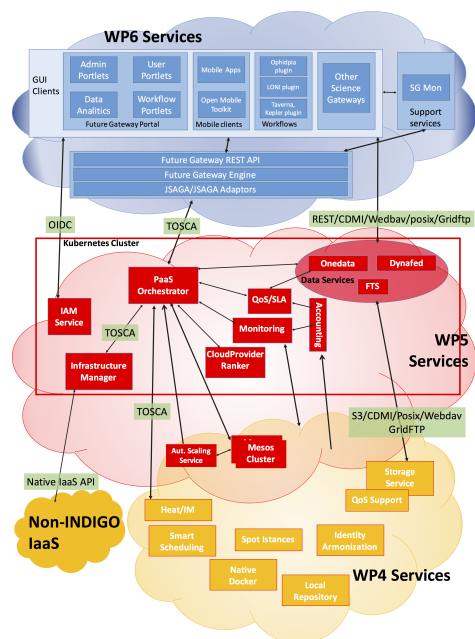


Fig. 1: The INDIGO-DataCloud architecture

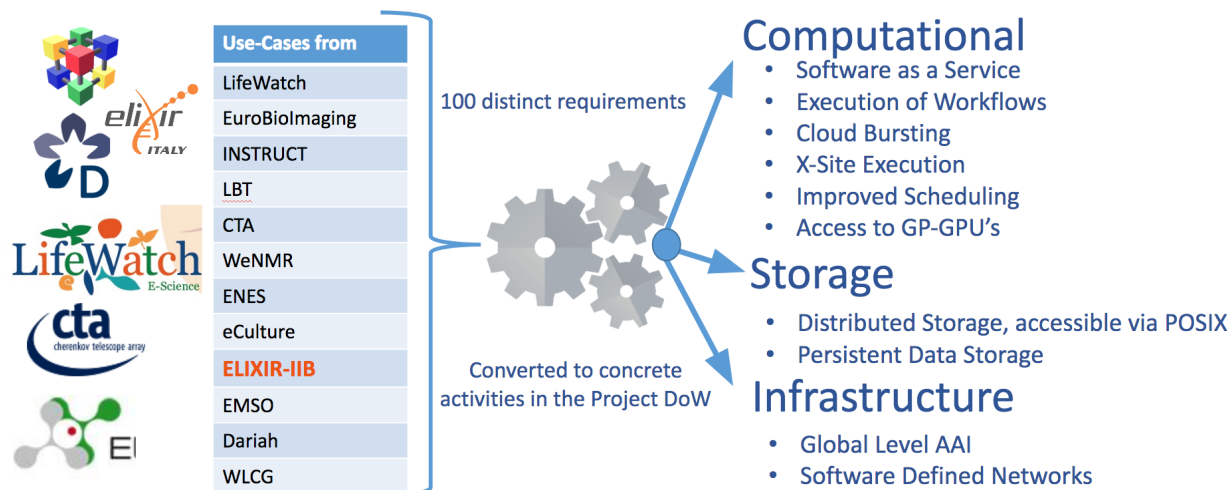


Fig. 2: The INDIGO-DataCloud communities

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: <https://www.indigo-datacloud.eu/service-component>

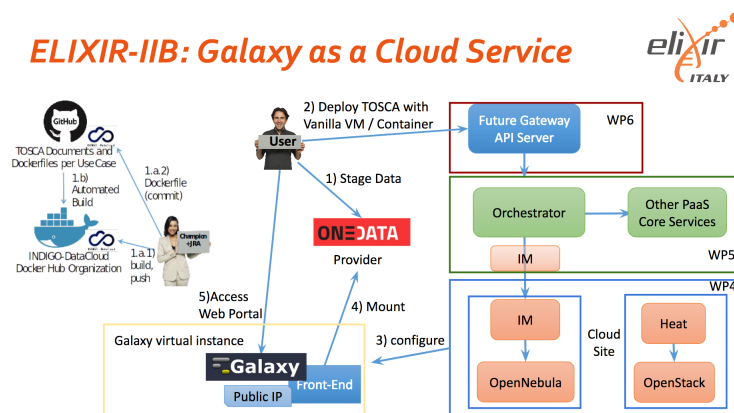


Fig. 3: ELIXIR-IIB use case in INDIGO architecture for single Galaxy instances deployment.

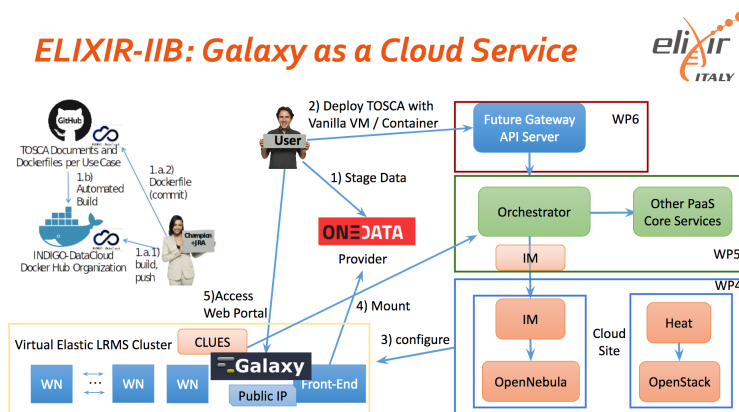


Fig. 4: ELIXIR-IIB use case in INDIGO architecture for Galaxy with cluster support deployment

Get Galaxy Express

The Galaxy Express section allows user to deploy a standard [Galaxy production environment](#).

The service instantiate a CentOS 7 Virtual Machine with Galaxy, all its companion software and tools already embedded. Once deployed each Galaxy instance can be further customized with tools and reference data.

See also:

For a detailed description of all Web UI options see section: [Laniakea options](#).

See also:

To login into the portal see section: [Authentication](#).

5.1 Instantiate Galaxy

1. Enter in the Galaxy Express section:
2. Describe your instance using the `Instance description` field, which will identify your Galaxy in the job list, once your request is submitted.
3. Select the Instance flavor, (virtual CPUs and RAM):

Currently, the following pre-sets are available, but not all of them are enabled.

Name	VCPUs	RAM	Total Disk	Root Disk
small	1	2 GB	20 GB	20 GB
medium	2	4 GB	20 GB	20 GB
large	4	8 GB	20 GB	20 GB
xlarge	8	16 GB	20 GB	20 GB
xxlarge	16	32 GB	20 GB	20 GB

4. Copy & Paste your SSH key, to login in the Galaxy instance:

[HOME](#)[Galaxy express](#)[Galaxy](#)[Galaxy cluster \(BETA\)](#)[Galaxy elastic cluster \(BETA\)](#)

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 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 configuration

Instance description

Virtual hardware

Galaxy

Instance flavor, i.e. CPUs, memory size (RAM), root disk size
large

SSH public key
Paste here your public key

Enable encryption
plain

Storage volume size
50 GB

Submit

No jobs available yet

[HOME](#)[Galaxy express](#)[Galaxy](#)[Galaxy cluster \(BETA\)](#)[Galaxy elastic cluster \(BETA\)](#)

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 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 configuration

Instance description

Virtual hardware

Galaxy

Instance flavor, i.e. CPUs, memory size (RAM), root disk size

xlarge

SSH public key


`c29bn32ITmmtKBA3ne/QIFRaaYI3XggvMXhhSSiYsJUdlSOjUTriB2DraHsxMGfOPjmPXkivrXp9MfOzjMg10fb7K2Mda8u/ujK/dvx3BnhlSlpn marco@marco-Latitude-3440`

Enable encryption

plain

Storage volume size

50 GB

Submit

No jobs available yet

5. Storage section allows to select the user storage volume size. The `Enable encryption` flag is explained here: *Get encrypted instance*.

Instance configuration

Instance description

Virtual hardware

Galaxy

Instance flavor, i.e. CPUs, memory size (RAM), root disk size

large

SSH public key

Paste here your public key

Enable encryption

plain

Storage volume size

✓ 50 GB

100 GB

200 GB

500 GB

Submit

No jobs available yet

6. Select the Galaxy version, the instance administrator e-mail and your custom Galaxy:

Warning: Please insert a vail mail address. No check is performed on its syntax, but entering an incorrect email address will cause deployment failure if the `encryption` option is set.

1. Select Galaxy tools pre-set:
2. and reference dataset:
3. Finally, `SUBMIT` your request:

5.2 Galaxy login

The galaxy administrator password is automatically generated during the instatiation procedure and is the same for each deployed instance:

```
User: galaxy administrator e-mail
Password: galaxy_admin_password
```

Warning: The anonymous login is by default disabled.

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

Instance configuration

Instance description

Virtual hardware

Galaxy

Galaxy version (release_18.05 recommended)

☒ release_18.05
☐ release_17.05

Instance description (Galaxy brand)

ELIXIR-ITALY

Galaxy administrator mail address

admin@elixir-italy.org

Galaxy flavor

galaxy-minimal

Reference data repository

elixir-italy.galaxy.refdata

Submit

No jobs available yet

Instance configuration

Instance description

Virtual hardware

Galaxy

Galaxy version (release_18.05 recommended)

release_18.05

Instance description (Galaxy brand)

ELIXIR-ITALY

Galaxy administrator mail address

admin@elixir-italy.org

Galaxy flavor

☒ galaxy-minimal
☒ galaxy-CoVaCS
☐ galaxy-GDC_Somatic_Variant
☐ galaxy-rna-workbench
☐ galaxy-epigen

elixir-italy.galaxy.refdata

Submit

No jobs available yet

Instance configuration

Instance description

Virtual hardware **Galaxy**

Galaxy version (release_18.05 recommended)

release_18.05

Instance description (Galaxy brand)

ELIXIR-ITALY

Galaxy administrator mail address

admin@elixir-italy.org

Galaxy flavor

galaxy-minimal

Reference data repository

- ☒ elixir-italy.galaxy.refdata
- ☒ elixir-italy.covacs.refdata
- ☐ data.galaxyproject.org

Submit

No jobs available yet

elixir

elixir-italy | Laniakea

HOME

Galaxy express

Galaxy

Galaxy cluster (BETA)

Galaxy elastic cluster (BETA)

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 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 configuration

Virtual hardware

Galaxy

Instance flavor, i.e. CPUs, memory size (RAM), root disk size

medium

SSH public key

Paste here your public key

Enable encryption

plain

Storage volume size

50 GB

Submit

Delete	Parameters	Submitted	Modified	Status	Description	Output	Advanced Info
✕	Show	2018-10-31T11:18:01Z	2018-10-31T11:56:31Z	DONE		http://90.147.75.22/galaxy	Show

1

The Galaxy section allows user to deploy a full [Galaxy production environment](#).

The service 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 selected Galaxt tools.

Warning: Everything is configured on the fly and, depending on the number of the tools to be installed may take time.

See also:

For a detailed descreption of all Web UI options see section: [Laniakea options](#).

See also:

To login into the portal see section: [Authentication](#).

6.1 Instantiate Galaxy

1. Enter the Galaxy section:
2. Describe your instance using the `Instance description` field, which will identify your Galaxy in the job list, once your request is submitted.
3. Select your instance flavour (virtual CPUs and the memory size):

Currently, the following pre-sets are available, but not all of them are enabled.

[HOME](#)[Galaxy express](#)[Galaxy](#)[Galaxy cluster \(BETA\)](#)[Galaxy elastic cluster \(BETA\)](#)

Deploy Galaxy on a single Virtual Machine installing it from scratch (SLOW).
The basic configuration includes CentOS 7, 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 configuration

Instance description

Virtual hardware

Galaxy

Instance flavor, i.e. CPUs, memory size (RAM), root disk size
large

SSH public key
Paste here your public key

Enable encryption
plain

Volume storage size
50 GB

Submit

No jobs available yet

20

Chapter 6. Get Galaxy

Name	VCPUs	RAM	Total Disk	Root Disk
small	1	2 GB	20 GB	20 GB
medium	2	4 GB	20 GB	20 GB
large	4	8 GB	20 GB	20 GB
xlarge	8	16 GB	20 GB	20 GB
xxlarge	16	32 GB	20 GB	20 GB

- Copy & Paste your SSH key, to login in the Galaxy instance:

[HOME](#)
[Galaxy express](#)
[Galaxy](#)
[Galaxy cluster \(BETA\)](#)
[Galaxy elastic cluster \(BETA\)](#)

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 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 configuration

Instance description

Virtual hardware

Galaxy

Instance flavor, i.e. CPUs, memory size (RAM), root disk size

xlarge

SSH public key

c29bn32TmmtKBA3ne/QIFRaaY13XggyMXhhSSiYsJUdISOjUTriB2DraHsxMGfOPjmPXkivrXp9MfOzjMg10fb7K2Mda8u/ujK/dvx3BnhISipn marco@marco-Latitude-3440

Enable encryption

plain

Storage volume size

50 GB

Submit

No jobs available yet

- Storage section allows to select the user storage volume size. The Enable encryption flag is explained here: *Get encrypted instance*.
- Select the Galaxy version, the instance administrator e-mail and your custom Galaxy brand:

Warning: Please insert a valid email address. No check is performed on its syntax, but entering an incorrect email address will cause deployment failure if the encryption option is set.

- Select Galaxy tools pre-set:
- and reference dataset:
- Finally, SUBMIT your request:

Instance configuration

Instance description

Virtual hardware

Galaxy

Instance flavor, i.e. CPUs, memory size (RAM), root disk size

large

SSH public key

Paste here your public key

Enable encryption

plain

Storage volume size

- ✓ 50 GB
- 100 GB
- 200 GB
- 500 GB

Submit

No jobs available yet

Instance configuration

Instance description

Virtual hardware

Galaxy

Galaxy version (release_18.05 recommended)

- ✓ release_18.05
- release_17.05

Instance description (Galaxy brand)

ELIXIR-ITALY

Galaxy administrator mail address

admin@elixir-italy.org

Galaxy flavor

galaxy-minimal

Reference data repository

elixir-italy.galaxy.refdata

Submit

No jobs available yet

Instance configuration

Instance description

Virtual hardware **Galaxy**

Galaxy version (release_18.05 recommended)

release_18.05

Instance description (Galaxy brand)

ELIXIR-ITALY

Galaxy administrator mail address

admin@elixir-italy.org

Galaxy flavor

- ✓ galaxy-minimal
- galaxy-CoVaCS
- galaxy-GDC_Somatic_Variant
- galaxy-rna-workbench
- galaxy-epigen

elixir-italy.galaxy.refdata

Submit

No jobs available yet

Instance configuration

Instance description

Virtual hardware **Galaxy**

Galaxy version (release_18.05 recommended)

release_18.05

Instance description (Galaxy brand)

ELIXIR-ITALY

Galaxy administrator mail address

admin@elixir-italy.org

Galaxy flavor

galaxy-minimal

Reference data repository

- ✓ elixir-italy.galaxy.refdata
- elixir-italy.covacs.refdata
- data.galaxyproject.org

Submit

No jobs available yet


elixir-italy | Laniakea beta

[HOME](#)
[Galaxy express](#)
[Galaxy](#)
[Galaxy cluster \(BETA\)](#)
[Galaxy elastic cluster \(BETA\)](#)

Deploy Galaxy on a single Virtual Machine installing it from scratch (SLOW).

The basic configuration includes CentOS 7, 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 configuration

Instance description

[Virtual hardware](#)
Galaxy

Instance flavor, i.e. CPUs, memory size (RAM), root disk size

large

SSH public key

Paste here your public key

Enable encryption

plain

Volume storage size

50 GB

[Submit](#)

Delete	Parameters	Submitted	Modified	Status	Description	Output	Advanced Info
+	Show	2018-12-13T10:32:50Z	2018-12-13T11:18:50Z	DONE		http://90.147.75.76/galaxy	Show

1

6.2 Galaxy login

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

```
User: your user e-mail  
Password: galaxy_admin_password  
API key: ADMIN_API_KEY
```

Warning: The anonymous login is by default disabled.

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

CHAPTER 7

Get encrypted instance

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

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

To encrypt the external volume storage flag `enable_encryption` in `Enable encryption` box.

Instance configuration

Instance description

Virtual hardware

Galaxy

Instance flavor, i.e. CPUs, memory size (RAM), root disk size

large

SSH public key

Paste here your public key

Enable encryption

✓ plain

encryption

Volume storage size

50 GB

Submit

No jobs available yet

Cryptographic keys should never be transmitted in the clear. For this reason during Galaxy deployment user intervention is required.

Data privacy is granted through LUKS storage encryption: user will be required to insert a password to encrypt/decrypt data directly on the virtual instance during its deployment, avoiding any interaction with the cloud administrator(s).

An e-mail is sent to instance administrator the e-mail address configured in the `Galaxy Configuration` section.

Instance configuration

Instance description

Virtual hardware

Galaxy

Galaxy version (release_18.05 recommended)
release_18.05

Instance description (Galaxy brand)
ELIXIR-ITALY

Galaxy administrator mail address 
user.name@elix

Galaxy flavor
galaxy-minimal

Reference data repository
elixir-italy.galaxy.refdata

Submit

No jobs available yet

Warning: Make sure you have entered a valid mail address!

The e-mail is sent you only when the system is ready to accept your password and contains all the instructions to correctly encrypt/decrypt your system. The e-mail subject is `[ELIXIR-ITALY] Laniakea storage encryption`, sent by `Laniakea@elixir-italy.org`

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

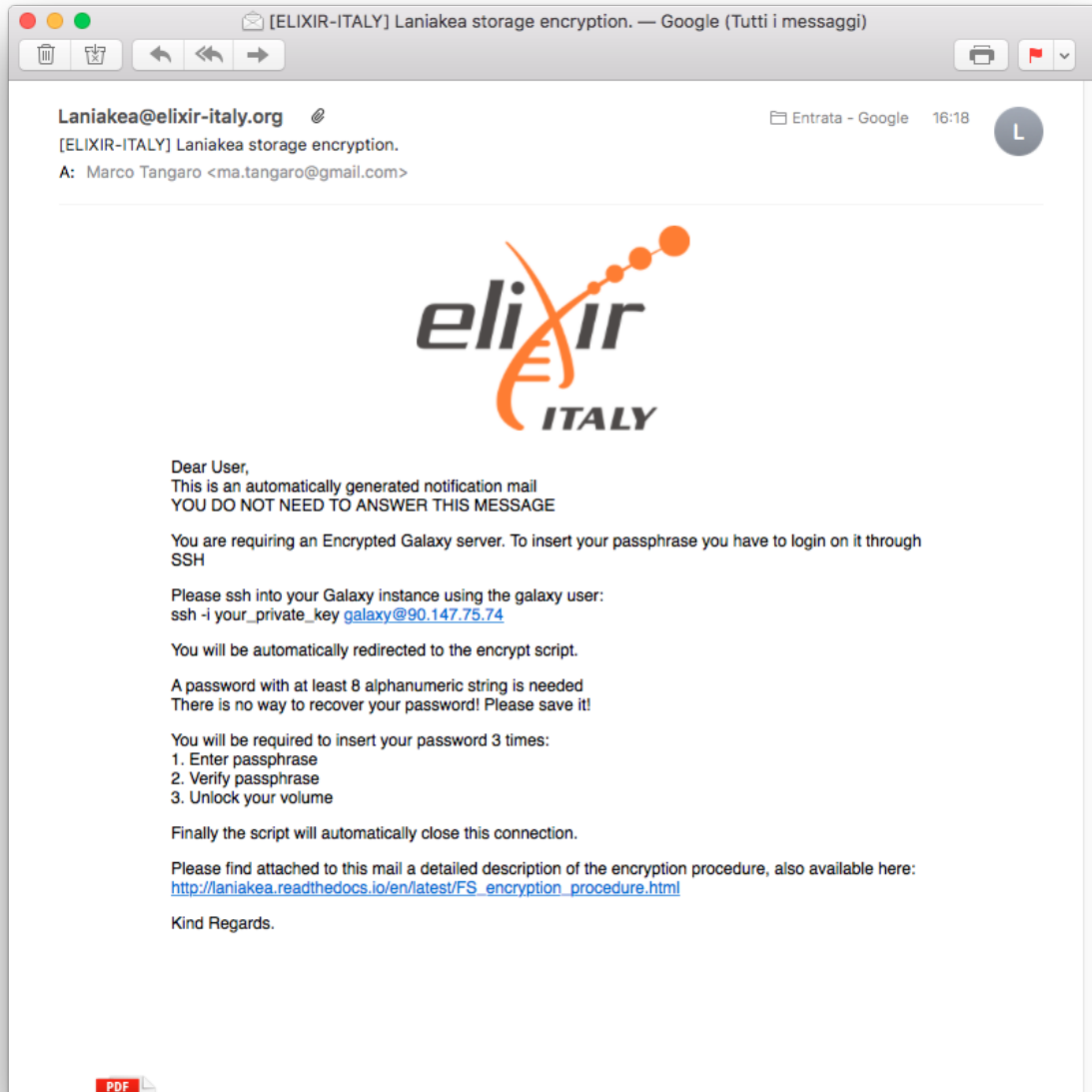
1. Check you SPAM mail directory
2. Check mail address spelling
3. Wait 15 minutes more.

Once the e-mail is arrived you can follow the step by step guide to encrypt your volume: [Storage encryption procedure](#).

User is only asked to insert their alphanumeric password 3 times:

1. Set password
2. Confirm password
3. Open LUKS volume.

After the password injection the logout is automatic: the encryption procedure continues in background.



7.1 Default encryption algorithm

The default LUKS configuration are:

```
Cipher: aes-xts-plain64  
Key size: 256 bit  
Hash Algorithm for key derivation: sha256
```

See also:

For a detailed description of all Web UI options see section: [Laniakea options](#).

Get Galaxy cluster

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.

It is possible to deploy a Galaxy cluster, using SLURM as Resource Manager, with or without automatic elasticity support.

1. Galaxy cluster section: all working nodes are deployed with the Galaxy server and are always available.
2. Galaxy elastic cluster section: 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.

The two sections provide the same configuration options.

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.

See also:

For a detailed description Galaxy elastic cluster see section *Cluster support (SLURM)*.

For a detailed description of all Web UI options see section: *Laniakea options*.

To login into the portal see section: *Authentication*.

8.1 Instantiate Galaxy

1. Enter in the `Galaxy cluster` section:

[HOME](#) [Galaxy express](#) [Galaxy](#) [Galaxy cluster \(BETA\)](#) [Galaxy elastic cluster \(BETA\)](#)

Deploy Galaxy from scratch with 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 configuration

Instance description

Virtual hardware

Galaxy

Galaxy server flavor, i.e. CPUs, memory size (RAM), root disk size
medium

Worker nodes number
1

Worker nodes flavor, i.e. CPUs, memory size (RAM), root disk size
large

SSH public key
Paste here your public key

Enable encryption
plain

Volume storage size
50 GB

Submit

No jobs available yet

- Describe your instance using the `Instance description` field, which will identify your Galaxy in the job list, once your request is submitted.
- Select the number of Virtual Worker Nondes of your Cluster:

Instance configuration

Instance description

Virtual hardware Galaxy

Galaxy server flavor, i.e. CPUs, memory size (RAM), root disk size

medium

Worker nodes number

1
2
3

Worker nodes flavor, i.e. CPUs, memory size (RAM), root disk size

large

SSH public key

Paste here your public key

Enable encryption

plain

Volume storage size

50 GB

Submit

No jobs available yet

- Select the Instance flavor, (virtual CPUs and RAM) for your Front node, i.e. the Galaxy server, and for each Worker Node:

Currently, the following pre-sets are available:

Name	VCPUs	RAM	Total Disk	Root Disk
small	1	2 GB	20 GB	20 GB
medium	2	4 GB	20 GB	20 GB
large	4	8 GB	20 GB	20 GB
xlarge	8	16 GB	20 GB	20 GB
xxlarge	16	32 GB	20 GB	20 GB

- Copy & Paste your SSH key, to login in the Galaxy instance:
- Storage section allows to select the IaaS storage volume size. The `Storage encryption` option is explained here: [Get encrypted instance](#).
- Select the Galaxy version, the instance administrator e-mail and your custom Galaxy brand:

Warning: Please insert a vail mail address. No check is performed on its syntax, but entering an incorrect email address will cause deployment failure if the `encryption` option is set.

- Select Galaxy tools pre-set:

Instance configuration

Instance description

Virtual hardware

Galaxy

Galaxy server flavor, i.e. CPUs, memory size (RAM), root disk size

medium

Worker nodes number

1

Worker nodes flavor, i.e. CPUs, memory size (RAM), root disk size

large

SSH public key

c29bn32ITmmtKBA3ne/QlFRaaYl3XggvMXhhSSlYsJUdISOjUTriB2DraHsxMGfOPjmPXkivrXp9MfOzjMg10fb7K2Mda8u/ujK/dvx3BnhlSlpn marco@marco-Latitude-3440

Enable encryption

plain

Volume storage size

50 GB

Submit

No jobs available yet

Instance configuration

Instance description

Virtual hardware

Galaxy

Galaxy server flavor, i.e. CPUs, memory size (RAM), root disk size

medium

Worker nodes number

1

Worker nodes flavor, i.e. CPUs, memory size (RAM), root disk size

large

SSH public key

Paste here your public key

Enable encryption

plain

Volume storage size

✓ 50 GB
100 GB
200 GB
500 GB

Submit

No jobs available yet

Instance configuration

Instance description

Virtual hardware

Galaxy

Galaxy version (release_18.05 recommended)

☒ release_18.05
☐ release_17.05

Instance description (Galaxy brand)

ELIXIR-ITALY

Galaxy administrator mail address

admin@elixir-italy.org

Galaxy flavor

galaxy-minimal

Reference data repository

elixir-italy.galaxy.refdata

Submit

No jobs available yet

Instance configuration

Instance description

Virtual hardware

Galaxy

Galaxy version (release_18.05 recommended)

release_18.05

Instance description (Galaxy brand)

ELIXIR-ITALY

Galaxy administrator mail address

admin@elixir-italy.org

Galaxy flavor

☒ galaxy-minimal
☒ galaxy-CoVaCS
☐ galaxy-GDC_Somatic_Variant
☐ galaxy-rna-workbench
☐ galaxy-epigen

elixir-italy.galaxy.refdata

Submit

No jobs available yet

2. and reference dataset:

Instance configuration

Instance description

Virtual hardware

Galaxy

Galaxy version (release_18.05 recommended)

release_18.05

Instance description (Galaxy brand)

ELIXIR-ITALY

Galaxy administrator mail address

admin@elixir-italy.org

Galaxy flavor

galaxy-minimal

Reference data repository

✓ elixir-italy.galaxy.refdata

elixir-italy.covacs.refdata

data.galaxyproject.org

Submit

No jobs available yet

3. Finally, SUBMIT your request:

8.2 Galaxy login

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

```
User: your user e-mail
Password: galaxy_admin_password
API key: ADMIN_API_KEY
```

Warning: The anonymous login is by default disabled.

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


elixir-italy | Laniakea beta

HOME
Galaxy express
Galaxy
Galaxy cluster (BETA)
Galaxy elastic cluster (BETA)

Deploy Galaxy from scratch with 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 configuration

Instance description

Virtual hardware
Galaxy

Galaxy server flavor, i.e. CPUs, memory size (RAM), root disk size

medium

Worker nodes number

1

Worker nodes flavor, i.e. CPUs, memory size (RAM), root disk size

large

SSH public key

Paste here your public key

Enable encryption

plain

Volume storage size

50 GB

[Submit](#)

Delete	Parameters	Submitted	Modified	Status	Description	Output	Advanced Info
	Show	2018-12-13T13:23:00Z	2018-12-13T14:18:02Z	DONE		http://90.147.102.120/galaxy	Show

1

Create SSH Keys

9.1 How to create SSH keys on Linux or macOS

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

9.2 How to create SSH keys on Windows

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

CHAPTER 10

Galaxy with Onedata

Warning: Galaxy with Onedata support is on-going and it is not currently available.

Virtual hardware presets

Each cloud provider enable a set of Image Flavor, defined in terms of Virtual CPUs (VCPUS), Memory, Disk, etc. Currently, the following pre-sets are available:

Name	VCPUS	RAM	Total Disk	Root Disk
small	1	2 GB	20 GB	20 GB
medium	2	4 GB	20 GB	20 GB
large	4	8 GB	20 GB	20 GB
xlarge	8	16 GB	20 GB	20 GB
xxlarge	16	32 GB	20 GB	20 GB

New flavors can be assigned to particular projects.

12.1 job identifier

Type: string

Description: Custom job identifier, it will identify your Galaxy in the job list.

Mandatory: No

12.2 Virtual hardware configuration

12.2.1 Instance flavor

Type: selectable menu (string)

Description: Select instance flavor in terms of virtual CPUs, memory size (RAM) and disk size.

Mandatory: Yes

Sections: **lgalaxy_expressl**, Galaxy. Available as Galaxy server flavor on cluster configuration tabs.

12.2.2 Worker node number

Type: selectable menu (int)

Description: Maximum worker nodes number.

Mandatory: Yes

Sections: Galaxy cluster, Galaxy elastic cluster.

12.2.3 Worker nodes flavor

Type: selectable menu (string)

Description: Select worker node flavor in terms of virtual CPUs, memory size (RAM) and disk size.

Mandatory: Yes

Sections: Galaxy cluster, Galaxy elastic cluster.

12.2.4 SSH public key

Type: string

Description: User personale SSH public key. Required to login through SSH in the Galaxy VM

Mandatory: Yes

Sections: all.

12.3 Storage configuration

12.3.1 Storage type

Type: selectable menu (string)

Description: Select common IaaS Block storage (IaaS) or Encrypted Storage (encryption).

Mandatory: Yes

Sections: all.

12.3.2 Storage size

Type: selectable menu (string)

Description: Select storage size.

Mandatory: Yes

Sections: all.

12.4 Galaxy configuration

12.4.1 Galaxy version

Type: selectable menu (string)

Description: Select Galaxy release version.

Mandatory: Yes

Sections: all.

12.4.2 Instance description

Type: string

Description: Customize Galaxy top-left main page badge.

Mandatory: No

Sections: all.

12.4.3 Galaxy administrator e-mail

Type: string

Description: Set galaxy administrator e-mail.

Mandatory: Yes

Sections: all.

12.4.4 Reference data

Type: selectable menu (string)

Description: Select reference data CernVM-FS provider.

Mandatory: Yes

Sections: all.

12.5 Tools configuration

12.5.1 Galaxy flavors

Type: selectable menu (string)

Description: Select tools presets to be installed on the server.

Mandatory: No

Sections: all.

CHAPTER 13

Galaxy production environment

The system allows to setup and launch a virtual machine (VM) 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. 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>.

13.1 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 (14.04 and 16.04) are both supported.

CentOS 7 and Ubuntu Xenial 16.04 exploit systemd as as init system, while Ubuntu Trusty 14.04 still uses upstart.

Warning: Selinux is by default disabled on CentOS.

13.2 PostgreSQL

PostgreSQL packages coming from PostgreSQL official repository are installed:

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

	• Users and groups management • System updates • Services management
	• Galaxy updates • Galaxy.ini configuration • Virtual environments setup
	• Galaxy database management • Separated galaxy tools database
	• Web server/web service interface
	• NGINX web server + upload module
	• FTP instance access • FTP data upload (> 2 GB)

Current stable PostgreSQL version is installed: PostgreSQL 9.6

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 and to Upstart for Ubuntu Trusty.

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

13.2.1 Galaxy database configuration

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

Galaxy database is named `galaxy` and is configured in the `galaxy.ini` 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
```

13.2.2 Docker

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.

13.3 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 the command line, from `/usr/sbin/nginx`:

```
$ sudo nginx
```

13.3.1 NGINX options

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

Option	Description
-?, -h	Print help.
-v	Print version.
-V	Print NGINX version, compiler version and configure parameters.
-t	Don't run, just test the configuration file. NGINX checks configuration for correct syntax and then try to open files referred in configuration.
-q	Suppress non-error messages during configuration testing.
-s signal	Send signal to a master process: stop, quit, reopen, reload. (version >= 0.7.53)
-p prefix	Set prefix path (default: <code>/usr/local/nginx/</code>). (version >= 0.7.53)
-c file-name	Specify which configuration file NGINX should use instead of the default.
-g directives	Set global directives. (version >= 0.7.4)

The main way to start/stop/reload nginx is through the `-s` command line option:

Action	Command
Start	sudo nginx
Stop	sudo nginx -s stop
Restart	First stop nginx then start it: sudo nginx -s stop; sudo nginx

Finally, to start/stop/status NGINX with systemd:

Distribution	Command
CentOS 7	sudo systemctl start/stop/status nginx
Ubuntu Xenial	sudo systemctl start/stop/status nginx
Ubuntu Trusty	sudo service nginx start/stop/status

13.3.2 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 polixy 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 dany 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 ran 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: <https://www.nginx.com/blog/nginx-se-linux-changes-upgrading-rhel-6-6/>
- StackOverflow post: <https://stackoverflow.com/questions/26334526/nginx-cant-access-a-uwsgi-unix-socket-on-centos-7>
- Blog post: <http://axilleas.me/en/blog/2013/selinux-policy-for-nginx-and-gitlab-unix-socket-in-fedora-19/>

13.4 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.ini file (\$HOME/galaxy/config/galaxy.ini), with 4 handler configuration. By default 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>.

UWSGI Galaxy Configuration:

```
[uwsgi]
master = True
processes = 1
socket = 127.0.0.1:4001
stats = 127.0.0.1:9191
pythonpath = /home/galaxy/galaxy/lib
pythonhome = /home/galaxy/galaxy/.venv
threads = 4
logto = /var/log/galaxy/uwsgi.log

# Job Handler(s)

[server:handler0]
use = egg:Paste@http
port = 8090
host = 127.0.0.1
use_threadpool = true
threadpool_workers = 5

[server:handler1]
use = egg:Paste@http
port = 8091
host = 127.0.0.1
use_threadpool = true
threadpool_workers = 5

[server:handler2]
use = egg:Paste@http
port = 8092
host = 127.0.0.1
use_threadpool = true
threadpool_workers = 5

[server:handler3]
use = egg:Paste@http
port = 8093
host = 127.0.0.1
use_threadpool = true
threadpool_workers = 5
```

13.5 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.

13.5.1 How to use FTP through FileZilla

You need to disable Passive (PASV) mode in FileZilla, since we are not going to open all passive ports.

1. Open FileZilla.
2. Click on Edit | Settings.
3. Open Connection menu on the left. Click on FTP menu.
4. Mark the Active radio button.
5. Click OK.

13.5.2 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 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|)
```

(continues on next page)

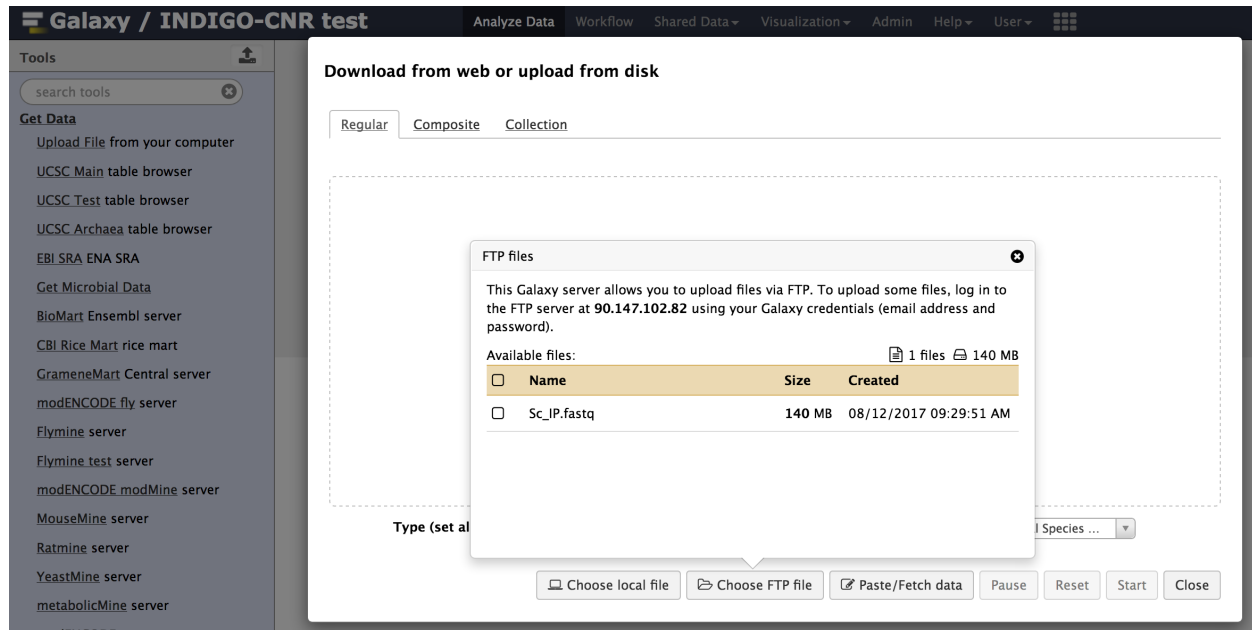
(continued from previous page)

```

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	to find out the pathname of 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 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	o 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).

13.6 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: https://docs.galaxyproject.org/en/master/admin/framework_dependencies.html?highlight=uwsgi#supervisor

and here: <https://galaxyproject.github.io/dagobah-training/2016-saltlakecity/002a-systemd-supervisor/systemd-supervisor.html#1>

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 running.

To manage Galaxy through supervisorctl:

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
```

(continues on next page)

(continued from previous page)

```
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
```

(continues on next page)

(continued from previous page)

```
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
```

13.7 Galaxy init scripts

Systemctl is the command line interface to systemd:

```
systemctl <start|stop|restart|...> <name>[.service]
systemctl <enable|disable> <name>[.service]
```

Since CentOS and Ubuntu Xenial 16.04 exploits systemd as init system, the Galaxy init script is located in `/etc/systemd/system/galaxy.service`.

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

Ubuntu Trusty 14.04 exploits Upstart as init system. Galaxy init file is located in `/etc/init.d/galaxy`.

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

Galaxy instance isolation

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.ini configuration file. For instance, 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
```

The service offers the possibility to store data exploiting file system encryption or Onedata.

Using file system encryption each Galaxy instance can be deployed as an insulated environment, i.e. data are isolated from any other instance on the same platform and from the cloud service administrators, opening to the adoption of Galaxy based cloud solutions even within clinical environments. Data privacy is granted through LUKS file system encryption: users will be required to insert a password to encrypt/decrypt data directly on the virtual instance during

its deployment, avoiding any interaction with the cloud administrator(s). The encryption procedure is described on [Storage encryption procedure](#).

Alternatively, Users' data access rights will be controlled through the OneData INDIGO component: [Onedata \(beta\)](#)

14.1 Storage encryption

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

For this reason, we have introduced the possibility for the user to isolate data through standard filesystem encryption, using the Linux device mapper encryption module, dm-crypt as encryption backend, cryptsetup, which is a command line interface to dm-crypt and LUKS (Linux Unified Key Setup) as encryption mode.

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.

Dm-crypt is the standard device-mapper encryption functionality provided by the Linux kernel. Dm-crypt management is entrusted to the cryptsetup userspace utility, using LUKS as default block-device encryption method. LUKS, is an additional layer which stores all the needed setup information for dm-crypt on the disk itself, abstracts partition and key management in an attempt to improve ease of use and cryptographic security

To automatically setup LUKS volumes on your Galaxy instances a bash script, named [fast-luks](#) has been created. The script has been integrated in the Galaxy instantiation procedure: if the File System Encryption option is selected through the dialogue window the users will be required to insert a password to encrypt/decrypt data on the virtual instance during its deployment, avoiding any interaction with the cloud administrator(s). For more details see: [Fast-luks script](#).

Note: During the anryption procedure an e-mail is sent to the Galaxy Instance Administrator, with the Virtual Machine IP address and a detailed description of the procedure to inject the encryption password in the system. The Galaxy installation procedure is paused until the password is correctly set (after 5 hours the system will return an error).

Warning: The system does not store your keys on its servers and cannot access your protected data unless you provide the key. This also means that if you forget or lose your key, there is no way to recover the key or to recover any data encrypted with the lost key.

14.1.1 Default configuration

Fast-luks, by default adopt `xts-aes-plain64` cipher with 256 bit keys and sha256 hashing algorithm.

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.

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

Defaults values

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

14.1.2 Password choice

During the encryption procedure (*Storage encryption procedure*), users are required to configure their encryption password. You must provide at least an alphanumeric 8-character long password. A random generated password is generated by the script, ONLY FOR EXAMPLE!

During the encryption procedure, the password will be required three times.

The script will query for passwords twice, to verify it. Then, you have to unlock your device, by inserting your password.

The procedure is described here: (*Storage encryption procedure*).

14.1.3 Luksctl

During the encryption procedure (*Storage encryption procedure*), `fast-luks` creates a configuration ini file, allowing Galaxy administrator to easily manage LUKS devices, through the `luksctl` script (*Luksctl: LUKS volumes management*).

By default the file is stored in `/etc/galaxy/luks-cryptdev.ini`.

The script requires superuser rights.

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

Galaxyctl can be used to parse `luksctl` commands:

Action	Command	Description
Open LUKS volume	<code>sudo galaxyctl open luks</code>	Open the encrypted device, requiring your passphrase.
Close LUKS volume	<code>sudo galaxyctl close luks</code>	Close the encrypted device
Check LUKS volume	<code>sudo galaxyctl status luks</code>	Check device status using <code>dm-setup</code>

14.1.4 Fast-luks script

The `fast-luks` script is located in `/usr/local/bin/fast-luks`.

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.

Typing `sudo fast-luks` the script will load defaults parameters and will LUKS format `/dev/vdb` device, otherwise different parameters can be specified.

NB: Run as root.

Argument	Defaults	Description
<code>-c, --cipher</code>	<code>aes-xts-plain64</code>	Set cipher specification string.
<code>-k, --keysize</code>	<code>256</code>	Set key size in bits.
<code>-a, --hash_algorithm</code>	<code>sha256</code>	For luksFormat action specifies hash used in LUKS key setup scheme and volumekey digest.
<code>-d, --device</code>	<code>/dev/vdb</code>	Set device to be mounted
<code>-e, --cryptdev</code>	<code>crypt</code>	Sets up a mapping <name> after successful verification of the supplied key (via prompting).
<code>-m, --mountpoint</code>	<code>/export</code>	Set mount point
<code>-f, --filesystem</code>	<code>ext4</code>	Set filesystem

```
$ sudo fast-luks --help
=====
                        ELIXIR-Italy
                    Filesystem encryption script

A password with at least 8 alphanumeric string is needed
There's no way to recover your password.
Example (automatic random generated passphrase):
                        PcHhaWx4

You will be required to insert your password 3 times:
  1. Enter passphrase
  2. Verify passphrase
  3. Unlock your volume

The connection will be automatically closed.

=====

fast-luks: a bash script to automate LUKS file system encryption.
usage: fast-luks [-h]

optionals arguments:
-h, --help            show this help text
-c, --cipher          set cipher algorithm [default: aes-xts-plain64]
-k, --keysize         set key size [default: 256]
-a, --hash_algorithm  set hash algorithm used for key derivation
-d, --device          set device [default: /dev/vdb]
-e, --cryptdev        set crypt device [default: cryptdev]
-m, --mountpoint      set mount point [default: /export]
-f, --filesystem      set filesystem [default: ext4]
--default             load default values
```

14.1.5 Cryptsetup howto

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:

NB: Run as root.

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
```

14.1.6 References

Disk encryption archlinux wiki page: https://wiki.archlinux.org/index.php/disk_encryption#Block_device_encryption_specific

Dm-crypt archlinux wiki page: https://wiki.archlinux.org/index.php/Dm-crypt/Device_encryption#Encryption_options_for_LUKS_mode

Original LUKS script: <https://github.com/JohnTroony/LUKS-OPs/blob/master/luks-ops.sh> (Credits to John Troon for the original script))

LUKS: <https://guardianproject.info/code/luks/>

LUKS how-to: <http://www.thegeekstuff.com/2016/03/cryptsetup-lukskey>

14.2 Storage encryption procedure

To encrypt the Virtual machine external volume follow this procedure.

14.2.1 Virtual Machine login

Log-in into your machine with:

```
ssh -i your_private_ssh_key.key galaxy@virtual.machine.ip.address
```

Typical IP addresses are: 90.147.170.xx, 90.147.102.xx or 90.147.75.xx and it is reported in the e-mail we sent you. You can copy and past the command from the mail the system send you.

Probably, you have to permanently accept the connection, typing “yes”.

and then enter your SSH passphrase.

14.2.2 Passphrase creation

You will be now prompted in the encryption script automatically. You will be required to insert an alphanumeric key, at least 8 characters. A key is automatically generated, as example, plase do not use if for production!

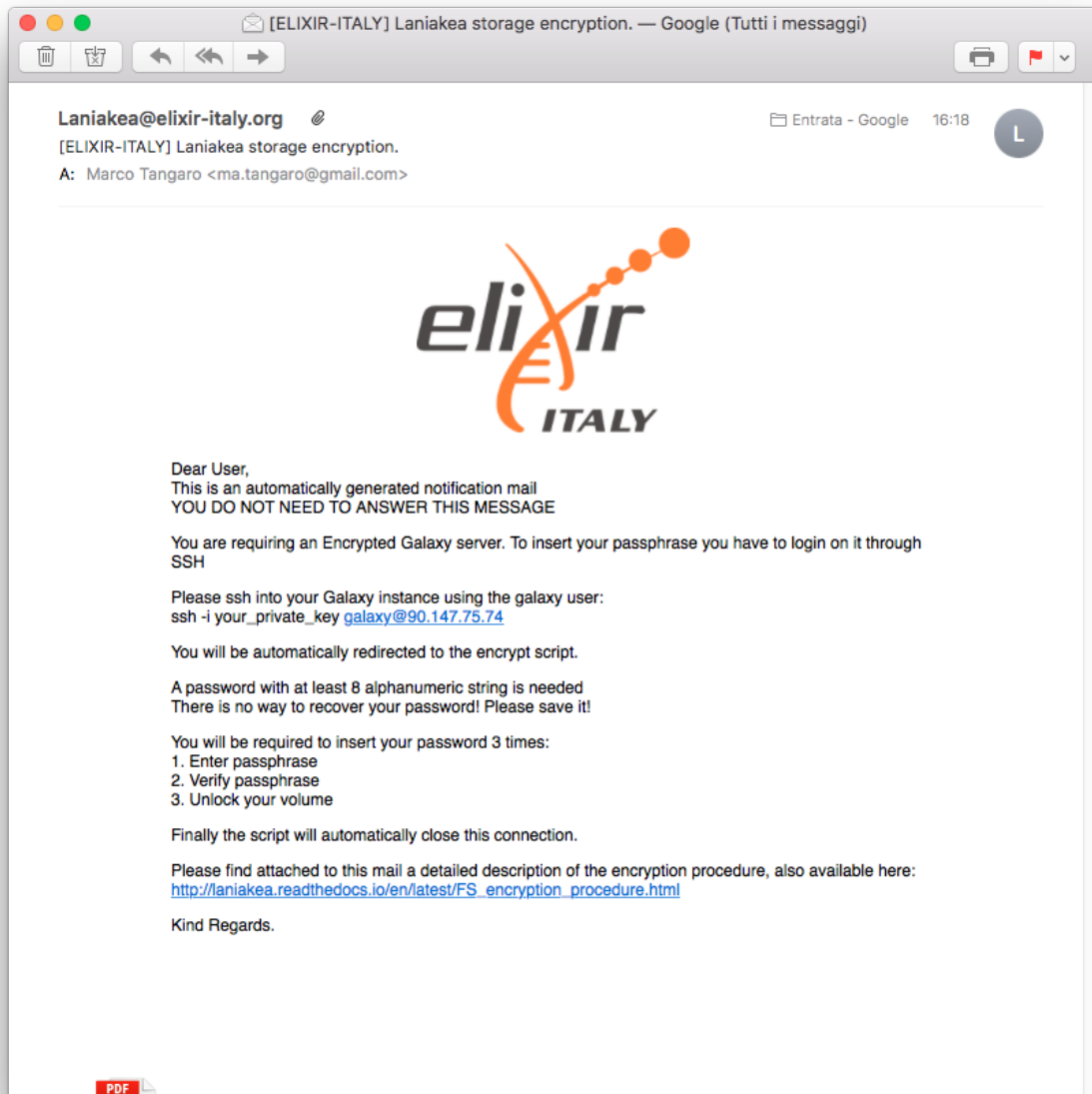
You have to type your password three times:

1. inject your password
2. Confirm your password
3. Unlock your encrypted volume

Insert your volume encrypt/decrypt password for the first time:

and confirm it:

If the passphrases don’t match, restart the procedure.





14.2.3 Unlock the volume

Unlock the encrypted volume typing again your password:

The volume will be now encrypted and you will be automatically log-out the VM, until Galaxy is installed.

14.3 File System Encryption Test

Test executed to ensure LUKS volume encryption.

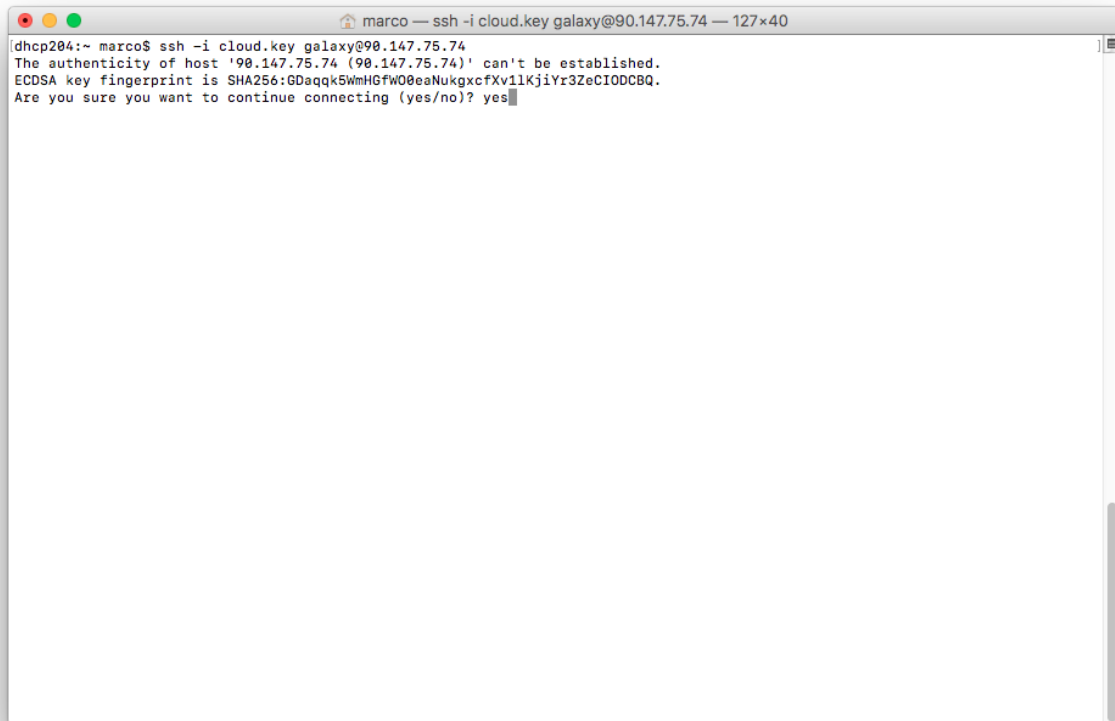
1. Create two volumes, here named vol1, vol2.
2. Attach each one to the instance (here listed as /dev/vdd and /dev/vde) and mount them respectively to /export and /export1.

```
$ 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
...
```

(continues on next page)



(continued from previous page)

```

/dev/vde          976M  2.6M  907M   1% /export1
/dev/mapper/jtedehex 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

```

- Put the same file on both volumes:

```

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

```

- Umount all the volumes and luksClose the encrypted one:

```

$ sudo cryptsetup luksClose /dev/mapper/jtedehex

```

- Create the volume binary image using dd:

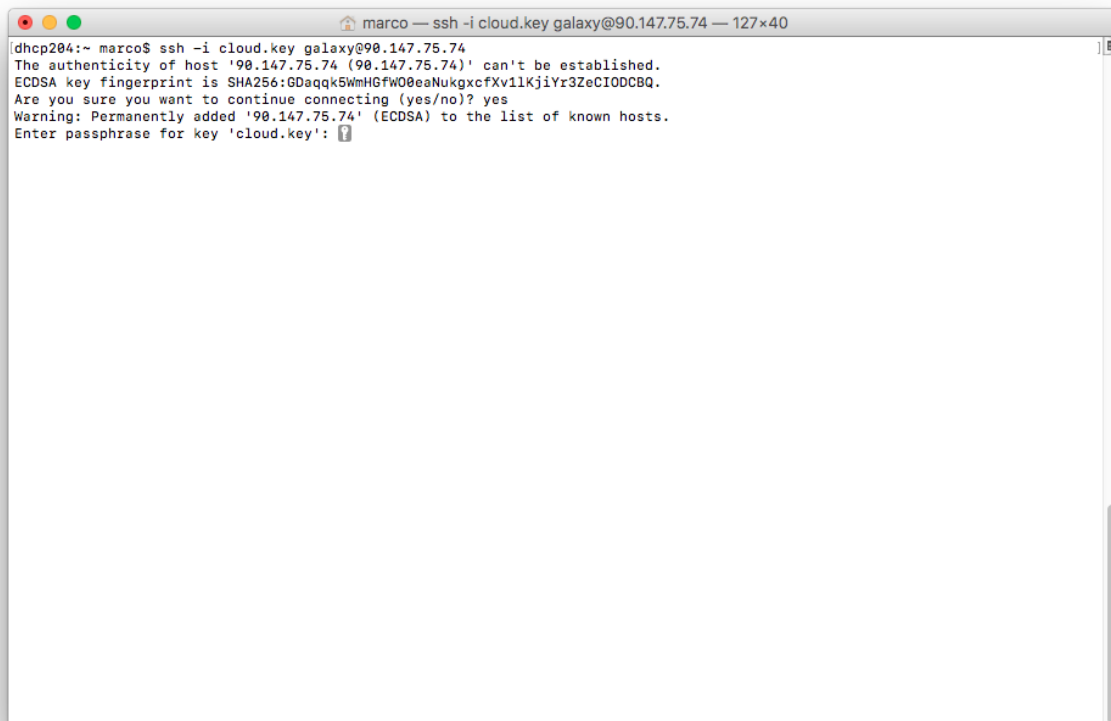
```

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

```

(continues on next page)



(continued from previous page)

```

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

```

(continues on next page)

```

marco — ssh -i cloud.key galaxy@90.147.75.74 — 127x40
dhcp204:~ marco$ ssh -i cloud.key galaxy@90.147.75.74
The authenticity of host '90.147.75.74 (90.147.75.74)' can't be established.
ECDSA key fingerprint is SHA256:GDaqk5WmHGfW00eaNukgxfXv1lKjiYr3ZeCIODCBQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '90.147.75.74' (ECDSA) to the list of known hosts.
Enter passphrase for key 'cloud.key':
=====
                ELIXIR-Italy
        Filesystem encryption script

A password with at least 8 alphanumeric string is needed
There's no way to recover your password.
Example (automatic random generated passphrase):
                FJujKDj9

You will be required to insert your password 3 times:
  1. Enter passphrase
  2. Verify passphrase
  3. Unlock your volume

=====

INFO 2018-09-14 14:22:41 [fast-luks-encryption] Check if the required applications are installed...
INFO 2018-09-14 14:22:41 [fast-luks-encryption] Start the encryption procedure.
Enter passphrase: █

```

(continued from previous page)

```

$ 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:

```

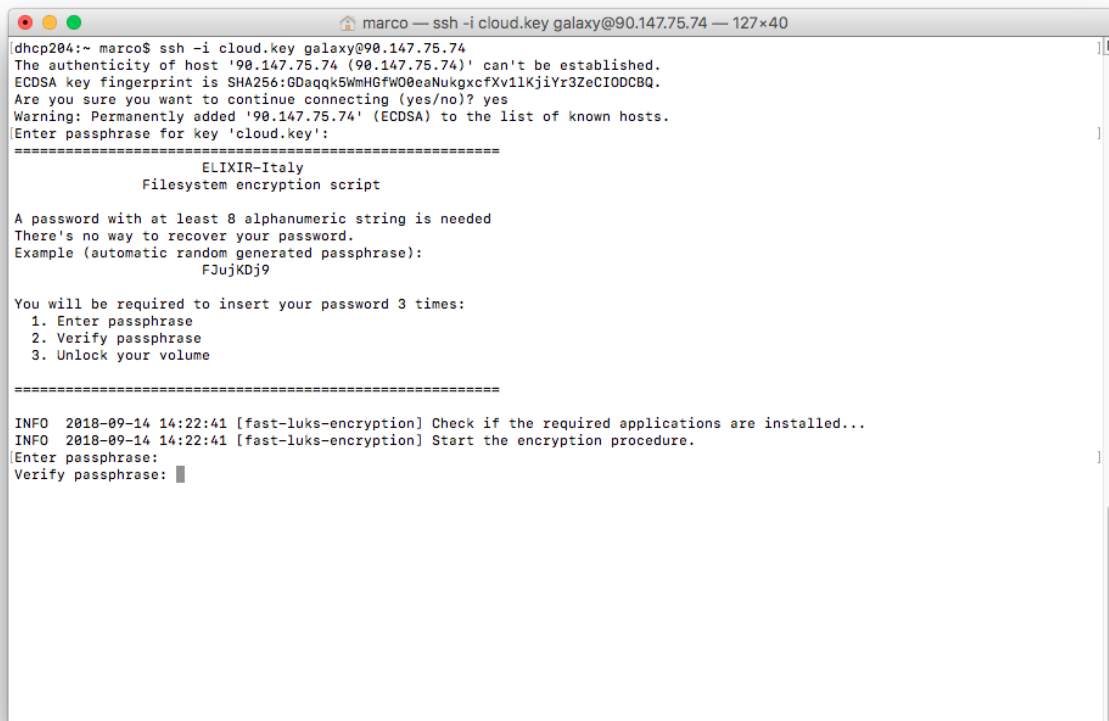
# rbd map volume-3bedc7bc-eaed-466f-9d55-f2c29b44a7b2 --pool volumes
/dev/rbd0

# 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]

(continues on next page)



```

marco — ssh -i cloud.key galaxy@90.147.75.74 — 127x40
dhcp204:~ marco$ ssh -i cloud.key galaxy@90.147.75.74
The authenticity of host '90.147.75.74 (90.147.75.74)' can't be established.
ECDSA key fingerprint is SHA256:6Daqqk5WmHGfW00eaNukgxcFv1lKjiYr3ZeCIODCBQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '90.147.75.74' (ECDSA) to the list of known hosts.
Enter passphrase for key 'cloud.key':
=====
ELIXIR-Italy
Filesystem encryption script

A password with at least 8 alphanumeric string is needed
There's no way to recover your password.
Example (automatic random generated passphrase):
    FJujKDj9

You will be required to insert your password 3 times:
  1. Enter passphrase
  2. Verify passphrase
  3. Unlock your volume

=====
INFO 2018-09-14 14:22:41 [fast-luks-encryption] Check if the required applications are installed...
INFO 2018-09-14 14:22:41 [fast-luks-encryption] Start the encryption procedure.
Enter passphrase:
Verify passphrase: █

```

(continued from previous page)

```

`-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'

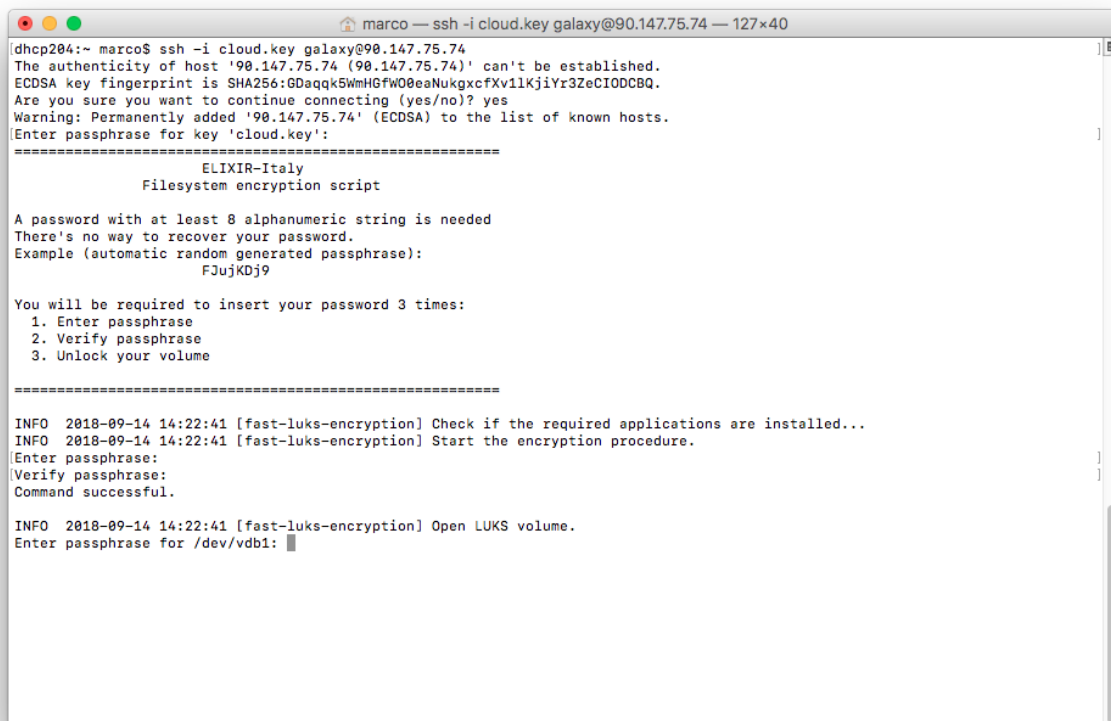
```

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

14.4 Onedata (beta)

Transparent access to the storage resources through token management (Onedata). Each Galaxy instance uses of OneData spaces to store data.

14.4.1 Get Onedata space support



```
marco — ssh -i cloud.key galaxy@90.147.75.74 — 127x40
dhcp204:~ marco$ ssh -i cloud.key galaxy@90.147.75.74
The authenticity of host '90.147.75.74 (90.147.75.74)' can't be established.
ECDSA key fingerprint is SHA256:GDaqqk5WmHGfw00eaNukgxcfXv1lKjiYr3ZeCIODCBQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '90.147.75.74' (ECDSA) to the list of known hosts.
Enter passphrase for key 'cloud.key':
=====
                ELIXIR-Italy
            Filesystem encryption script

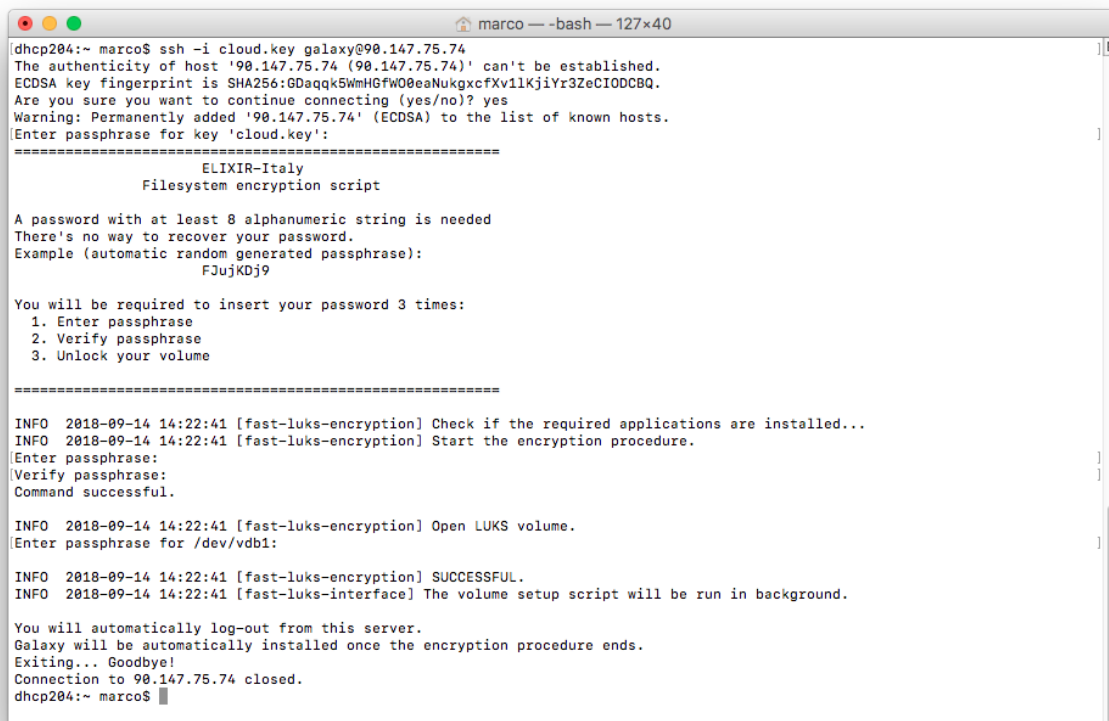
A password with at least 8 alphanumeric string is needed
There's no way to recover your password.
Example (automatic random generated passphrase):
                FJuJKDj9

You will be required to insert your password 3 times:
 1. Enter passphrase
 2. Verify passphrase
 3. Unlock your volume

=====

INFO 2018-09-14 14:22:41 [fast-luks-encryption] Check if the required applications are installed...
INFO 2018-09-14 14:22:41 [fast-luks-encryption] Start the encryption procedure.
Enter passphrase:
Verify passphrase:
Command successful.

INFO 2018-09-14 14:22:41 [fast-luks-encryption] Open LUKS volume.
Enter passphrase for /dev/vdb1: 
```



```
marco — -bash — 127x40
dhcp204:~ marco$ ssh -i cloud.key galaxy@90.147.75.74
The authenticity of host '90.147.75.74 (90.147.75.74)' can't be established.
ECDSA key fingerprint is SHA256:GDaqqk5WmHGfw00eaNukgxcfXv1lKjiYr3ZeCIODCBQ.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '90.147.75.74' (ECDSA) to the list of known hosts.
[Enter passphrase for key 'cloud.key':]
=====
                ELIXIR-Italy
            Filesystem encryption script

A password with at least 8 alphanumeric string is needed
There's no way to recover your password.
Example (automatic random generated passphrase):
                FJujKDj9

You will be required to insert your password 3 times:
 1. Enter passphrase
 2. Verify passphrase
 3. Unlock your volume

=====

INFO 2018-09-14 14:22:41 [fast-luks-encryption] Check if the required applications are installed...
INFO 2018-09-14 14:22:41 [fast-luks-encryption] Start the encryption procedure.
[Enter passphrase:]
[Verify passphrase:]
Command successful.

INFO 2018-09-14 14:22:41 [fast-luks-encryption] Open LUKS volume.
[Enter passphrase for /dev/vdb1:]

INFO 2018-09-14 14:22:41 [fast-luks-encryption] SUCCESSFUL.
INFO 2018-09-14 14:22:41 [fast-luks-interface] The volume setup script will be run in background.

You will automatically log-out from this server.
Galaxy will be automatically installed once the encryption procedure ends.
Exiting... Goodbye!
Connection to 90.147.75.74 closed.
dhcp204:~ marco$
```

CHAPTER 15

Galaxy ShedTools

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 automatically installed using the official GalaxyProject python library [Ephemeris](#).

Current possible pre-sets:

1. [galaxy-epigen](#): Galaxy ready for NGS analysis.
2. [galaxy-rna-workbench](#): Galaxy ready for RNA Sequencing analysis (original galaxy flavor [here](#)).
3. [galaxy-testing](#): Galaxy test recipe.

The corresponding recipe on github is downloaded and processed. It is possible to easily add new flavors, just adding new recipes on github.

15.1 Create and test Galaxy flavors

For each tool you want to install, you must provide tool name and owner and one between `tool_panel_section_id` and `tool_panel_section_label`.

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 tool_panel_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.		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 tool_panel_section_label 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').		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

For instance, this is the galaxy TESTING recipe:

```
---
api_key: <Admin user API key from galaxy_instance>
galaxy_instance: <Galaxy instance IP>

tools:
- name: fastqc
  owner: devteam
  tool_panel_section_label: 'Tools'
  install_resolver_dependencies: True
  install_tool_dependencies: False

- name: 'bowtie_wrappers'
  owner: 'devteam'
  tool_panel_section_label: 'Tools'
  install_resolver_dependencies: True
  install_tool_dependencies: False
```

15.2 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.

15.3 References

Galaxy flavors: <https://github.com/bgruening/docker-galaxy-stable#Extending-the-Docker-Image>

Ephemeris: <https://ephemeris.readthedocs.io/en/latest/>

Ephemeris documentation: <https://github.com/galaxyproject/ephemeris>

Conda for Galaxy tools dependencies: https://docs.galaxyproject.org/en/master/admin/conda_faq.html

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.

16.1 Available reference data

Reference data (e.g. genomic sequences) are available for many species and shared among all the instances, avoiding unnecessary and costly data duplication, exploiting the CernVM-FS filesystem or Onedata.

Until now, Galaxy administrators have been responsible for downloading, building and installing these important reference data. For example, to make the UCSC hg19 build of the human reference genome available to the Burrows-Wheeler Aligner (BWA) short-read mapper (Li and Durbin, 2009), a Galaxy administrator would need to (i) download the reference genome FASTA file, (ii) make it available as a reference genome via the 'all_fasta' table (optional), (iii) build BWA alignment indexes via proper command-line calls, (iv) register the location and availability of the indexes within the 'bwa_indexes' data table (by adding an additional entry to the tool-data/bwa_index.loc file on disk) and (v) finally, restart the Galaxy server.

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

```
Arabidopsis thaliana (TAIR9)
Arabidopsis thaliana (TAIR10)
Drosophila melanogaster (dm3)
Homo sapiens (hg18)
Homo sapiens (hg19)
Homo sapiens (hg38)
Mus musculus (mm9)
Mus musculus (mm10)
Saccharomyces cerevisiae (sacCer3)
```

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

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 bowite

16.1.1 Homo Sapiens - hg19

Available reference index	Details
bowtie	
bowtie2	
bwa	
liftover	
rsem	created with rsem v1.3.0, table and gtf file from ucsc (download date 11 March 2018)
.fa	
.2bit	
.gtf	download date: 11 March 2018

16.2 CernVM-FS reference data

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). The reference data Files and directories are hosted on standard web servers and mounted on `/refdata` directory:

```
$ ls -l /refdata/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
```

(continues on next page)

(continued from previous page)

```
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
```

16.2.1 Cvmfs client setup

Cvmfs is installed by default on each Galaxy instance (CentOS 7 or Ubuntu 16.04) if this is the reference data configuration provided by your service provider.

The `elixir-italy.galaxy.refdata.pub` 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`, therefore `mount` command is used to correctly mount the cvmfs volume to `/refdata`.

Description	Command
check configuration	<code>cvmfs_config chksetup</code>
mount volume	<code>mount -t cvmfs elixir-italy.galaxy.refdata /refdata/elixir-italy.galaxy.refdata</code>
umount volume	<code>cvmfs_config umount elixir-italy.galaxy.refdata</code>
reload repository	<code>cvmfs_config reload elixir-italy.galaxy.refdata</code>

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

Note: Cvmfs commands require root privileges

Cvmfs mount output:

```
$ 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
```

16.2.2 Cvmfs server location

Current cvmfs server configuration:

Reference data cvmfs	Details
cvmfs repository name	<code>elixir-italy.galaxy.refdata</code>
cvmfs server url	<code>90.147.102.186</code>
cvmfs key file	<code>elixir-italy.galaxy.refdata.pub</code>
cvmfs proxy url	<code>DIRECT</code>

16.2.3 Troubleshooting

Cvmfs not running, e.g. after reboot:

```
$ 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... Failed to initialize root file catalog (16 - file_
↪catalog failure)
```

A reload of the config is able to fix the problem: <https://wiki.chipp.ch/twiki/bin/view/CmsTier3/IssueCvmfsFailsToMount>

```
$ 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
```

16.2.4 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`.

16.2.5 Cvmfs references

CernVM-FS: <https://cernvm.cern.ch/portal/filesystem>

Cvmfs documentation: <http://cvmfs.readthedocs.io/en/stable/>

16.3 Onedata reference data (beta)

To Be Updated

16.4 Reference data local download

The reference data set can be downloaded on your machine. This option is not explicitly available, by default, on the service web interface to avoid unuseful replication and costly virtual space consumption.

Nevertheless, it is still possible to download them through ansible and automatically configure galaxy to use them: *Galaxycloud-refdata*.

Cluster support (SLURM)

The service provides support for virtual clusters through a dedicated section of the web front-end and allows to instantiate Galaxy with SLURM (slurm.schedmd.com) as Resource Manager and to customize the number of virtual nodes, nodes and master virtual hardware.

Automatic elasticity, provided using the CLUES INDIGO service component (*INDIGO 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.

Each node is configured according to the Galaxy tools installed on the VM as selected by the user during the configuration phase. All tools pre-set are tested to work with the galaxy elastic cluster.

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

The service is scalable and both users and service providers can chose among a full range of different computational capabilities: from limited ones to serve e.g. small research groups, Galaxy developers or for didactic and training purposes, to instances with elasticity cluster support to deliver enough computational power for any kind of bioinformatic analysis and number of users, opening the route for the migration of public Galaxy instances to this service.

17.1 Shared file system

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

1. `/home` where galaxy is installed.
2. `/export` where galaxy input and output datasets are hosted.

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

17.2 Nodes deployment

Warning: Worker node deployment takes 10 minutes! Then your job will run. If the node is already deployed the job starts immediately.

This is due to:

1. VM configuration
2. Tools dependencies installation
3. CernVM-FS configuration
4. SLURM installation and configuration

17.3 SLURM main commands

Please have a look here for a summary of SLURM main commands: <https://www.rc.fas.harvard.edu/resources/documentation/convenient-slurm-commands/>

CHAPTER 18

Authentication

Currently, the authentication system relies on INDIGO-AAI.

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

 elixir-italy | Laniakea

[Sign In](#)


LANIAKEA
Automatic deployment of virtual Galaxy environments for life science

Documentation: <http://laniakea.readthedocs.io>

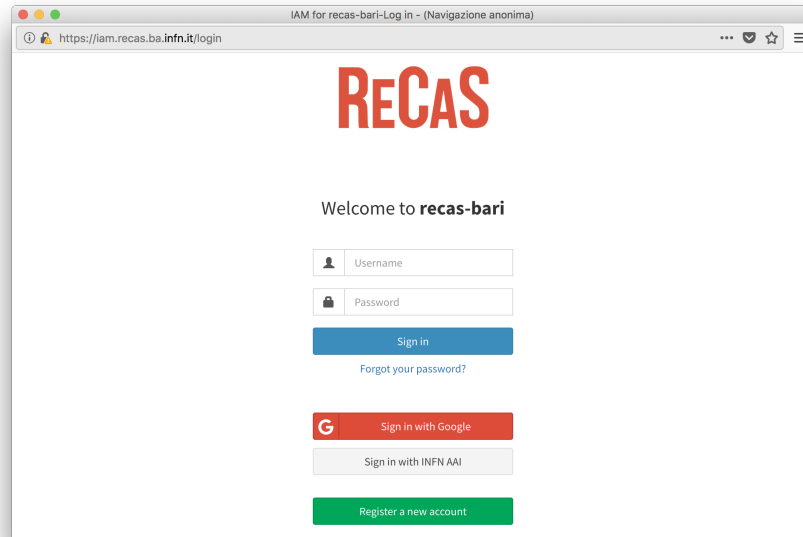
Laniakea closed beta program is now open. To require resources please contact us at laniakea.helpdesk@gmail.com to be included among early beta users. Thank you.

   UNIVERSITÀ
DEGLI STUDI
DI MILANO 

Powered By Liferay

18.1 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:

and accept the usage policy to complete the registration:

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

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:

18.2 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 portal sections and instantiate Galaxy:



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

Family name

Email

Username

Please choose a username

Notes

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).



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](#)

Confirm your recas-bari registration request Posta in arrivo x



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 x



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

Two password input fields with masked characters (dots). The second field has a green border, indicating it is the required field. Below the fields is a blue "Save" button.



Your password has been reset successfully!

[Back to Login Page](#)

ReCAS IAM for recas-bari

laniakea.testuser

Approval Required for *elixir-italy-science-gateway-testing*

This client was dynamically registered on January 29, 2018.

[more information](#)

You will be redirected to the following page if you click Approve:
https://elixir-italy-science-gateway.cloud.ba.infn.it/c/portal/iam_op

Access to:

- ☒ log in using your identity
- ☒ basic profile information
- ☒ email address
- ☒ 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 "elixir-italy-science-gateway-testing"?

Authorize

Deny

 elixir-italy | Laniakea

Sign In


LANIAKEA
Automatic deployment of virtual Galaxy environments for life science

Documentation: <http://laniakea.readthedocs.io>

Laniakea closed beta program is now open. To require resources please contact us at laniakea.helpdesk@gmail.com to be included among early beta users. Thank you.

 UNIVERSITÀ
DEGLI STUDI
DI MILANO

Powered By Liferay

IAM for recas-bari-Log in - (Navigazione anonima)

https://iam.recas.ba.infn.it/login



Welcome to **recas-bari**

Sign in

[Forgot your password?](#)

 Sign in with Google

Sign in with INFN AAI

Register a new account

 elixir-italy | Laniakea

HOME Galaxy express Galaxy Galaxy cluster (BETA) Galaxy elastic cluster (BETA)

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 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 configuration

Virtual hardware

Galaxy

Instance flavor, i.e. CPUs, memory size (RAM), root disk size

medium

SSH public key

Paste here your public key

Enable encryption

plain

Storage volume size

50 GB

Submit

Delete	Parameters	Submitted	Modified	Status	Description	Output	Advanced Info
	Show	2018-10-31T11:18:01Z	2018-10-31T11:56:31Z	DONE		http://90.147.75.22/galaxy	Show

1

CHAPTER 19

Validation

Warning: Validation package for automatic tools test is under development and it is not currently available.

Testing module for Galaxy workflows: <https://github.com/phnmnl/wft4galaxy>

CHAPTER 20

Galaxyctl libraries

Galaxyctl is a python script collection for Galaxy management (first start, stop/start/restart/status). Moreover it is possible to manage, through specific script, LUKS volumes and Onedata spaces.

Galaxyctl requires superuser privileges.

Current version: 0.0.1

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.
luksctl	LUKS volume script management
onedatactl	Onedata spaces for user data and reference data management.

Galaxyctl_libs is composed by several modules.

20.1 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
```

20.2 DetectGalaxyCommands

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

20.3 UwsgiSocket

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

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

20.4 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

20.4.1 GetUwsgiStatsServer

To connect to running uWSGI stats server call:

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

20.4.2 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')
```

20.4.3 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 LUKSctl

Read LUKS ini file, usually stored on `/etc/galaxy/luks-cryptdev.ini`, for LUKS volume management. Open, Close and Status commands are managed through `luksctl` script.

20.6 OneDataCtl

Reads Onedata ini file, usually stored on `/etc/galaxy/onedatactl.ini`, for Onedata space management: both user data and reference data.

20.6.1 mount_space

To mount onedata space (userdata or refdata), call:

```
onedata = OneDataCtl('/etc/galaxy/onedatactl.ini', 'userdata')
onedata.mount_space()
```

20.6.2 umount_space

To umount onedata space, call:

```
onedata = OneDataCtl('/etc/galaxy/onedatactl.ini', 'userdata')
onedata.umount_space()
```

Galaxyctl: Galaxy management

Galaxyctl is a python script collection used for Galaxy management. In particular it exploits `galaxyctl_libs` to properly check uWSGI Stats to correctly retrieve Galaxy and uWSGI workers status. Moreover the script allow to manage, using the same command, luks volume and onedata spaces, parsing `luksctl` and `onedatctl` commands.

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.
luks	status	Show LUKS volume status
	open	LUKSOpen volume
	close	LUKSClose volume
onedata	status	Show onedata space status. <code>--userdata</code> allow to check user data space if mounte through oneclient. <code>--refdata</code> allow to check reference data onedata space if mounted through oneclient.
	mount	Mount onedata space using oneclient. Use <code>--userdata</code> or <code>--refdata</code> to mount user data and reference data volumes.
	umount	Umount onedata space using furmount.

21.1 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>

21.2 Logging

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

21.3 Advanced options

21.3.1 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.

21.3.2 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.

21.3.3 restart

To restart Galaxy:

```
sudo galaxyctl restart galaxy
```

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

21.3.4 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 it ensure that Galaxy has correctly started. Then uWSGI workers are checked to ensure Galaxy is acceptin 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
```

21.4 LUKS module

By parsing luksctl script and using galaxyctl_libs, galaxyctl is able to manage LUKS volumes

Action	Command
Open LUKS volume	<code>sudo galaxyctl open luks</code>
Close LUKS volume	<code>sudo galaxyctl close luks</code>
Check LUKS volume	<code>sudo galaxyctl status luks</code>

In particular to unlock you LUKS volume:

```
sudo galaxyctl open luks
```

Then you will be asket to insert your LUKS passphrase. For instance:

```
(.venv) [galaxy@galaxy-indigo-test ~]$ sudo galaxyctl open luks
Enter passphrase for /dev/disk/by-uuid/42aaf979-6351-44e9-97ee-19e7f8c5e9f6:
```

21.5 Onedata module

By parsing onedatactl script and using galaxyctl_libs, galaxyctl is able to manage onedata user data and reference data spaces.

Data	Action	Command
User data	mount	<code>sudo galaxyctl mount onedata --userdata</code>
	umount	<code>sudo galaxyctl umount onedata --userdata</code>
	status	<code>sudo galaxyctl status onedata --user-data</code>
Reference data	mount	<code>sudo galaxyctl mount onedata --ref-data</code>
	umount	<code>sudo galaxyctl umount onedata --userdata</code>
	status	<code>sudo galaxyctl status onedata --ref-data</code>

The options `--userdata` and `--refdata` are mutually exclusive.

21.6 Configuration files

Supervisord and systemd/upstart are supported to start/stop/restart/status Galaxy. The init system can be set using the variable `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 retrieve the right command to be used and it is compatible with both CentOS 7 and Ubuntu (14.04 and 16.04).

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

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

while on Ubuntu:

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

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

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

For LUKS volume configuration, the script reads our custom `luks-cryptdev.ini` file (stored in `/etc/galaxy/` and needs `luksctl` script path (usually stored in `/usr/local/bin`) to load methods

```
luks_config_file = '/etc/galaxy/luks-cryptdev.ini'
luksctl_path = '/usr/local/bin'
```

Finally, for onedata spaces management, `onedatactl.ini` file (stored in `/etc/galaxy`) and `onedatactl` path (usually `/usr/local/bin`) are needed:

```
onedatactl_config_file = '/etc/galaxy/onedatactl.ini'
onedatactl_path = '/usr/local/bin'
```

Luksctl: LUKS volumes management

Luksctl is a python script allowing to easily Open/Close and Check LUKS encrypted volumes, parsing dmsetup and cryptsetup commands.

The script requires superuser rights.

Action	Command
Open	sudo luksctl open
Close	sudo luksctl close
Status	sudo luksctl status

22.1 Dependencies

Since the script is going to parse cryptsetup and dmsetup commands, both are required

cryptsetup

22.2 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
```

(continues on next page)

(continued from previous page)

```
Number of targets: 1
UUID: CRYPT-LUKS1-9bc8b7c6dc7e4aac9cd78b7258facc75-ribqvkjj

Encrypted volume: [ OK ]
```

22.3 Close LUKS volumes

To Close LUKS volume, call `luksctl close`:

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

22.4 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 ]
```

Onedactl: Onedata spaces management

Onedactl is a python script to mount Onedata spaces, for user data and reference data. This script parse oneclient options using a configuration file (onedactl.ini) to retrieve the required onedata space information (e.g. oneprovider, token and mountpoint).

23.1 Options

Parsing all oneclient option is out of the purpose of this script. Therefore the script provides only few options to allow user to easily mount their data hosted on onedata

Option	Description
<code>mount</code>	Mount user data or reference data space.
<code>umount</code>	Umount user data or reference data space.
<code>-t --token</code>	Set access token.
<code>-H --provider</code>	Set onedata provider.
<code>-m --mountpoint</code>	Set mountpoint.
<code>-c config_file</code>	Load configuration file.
<code>--insecure</code>	Allow insecure mount.
<code>--nonempty</code>	Mount space on non empty space.
<code>--version</code>	Show galaxyctl_libs version.

23.2 The onedactl.ini file

Onedactl.ini file is located in `/etc/galaxy/onedactl.ini`. The ini file has two sections: `[userdata]` for user data management and `refdata` for reference data management.

23.2.1 mountpoint

By enabling this option onedata space will be used to store user data and it will be mounted on the specified directory, e.g. `/export` This argument is required.

```
mountpoint = /export
```

23.2.2 provider

Insert Onedata provider for user data, e.g. `oneprovider2.cloud.ba.infn.it` This argument is required.

```
provider = oneprovider2.cloud.ba.infn.it
```

23.2.3 token

Insert Onedata token for user data. This argument is required.

```
token = ↵  
↪MDAxNWxvY2F0OaW9uIG9uZXpvmUKMDAzYmlkZW50aWZpZXIgeExqMi00xdFN3YVp1VWIxM1dFSzRoNEdkb2x3cXVwTnpSaGZ0
```

23.2.4 insecure

Allow insecure connection. This is an optional argument.

```
insecure = False
```

23.2.5 nonempty

Allow to mount space on non empty directory. This is an optional argument

```
nonempty = False
```

23.3 Onedatactl options

The script allow to mount two different volumes.

23.3.1 mount

For user data, call:

```
sudo onedatactl mount userdata
```

For reference data, call:

```
sudo onedatactl mount refdata
```

23.3.2 umount

For user data, call:

```
sudo onedatactl umount userdata
```

For reference data, call:

```
sudo onedatactl umount refdata
```

23.3.3 status

For user data, call:

```
sudo onedatactl status userdata
```

For reference data, call:

```
sudo onedatactl status refdata
```

23.4 Oneclient

Onedatactl parse oneclient command using the information stored in `/etc/galaxy/onedatactl.ini` file. Therefore oneclient can be used to perform the same actions.

The basic command line syntax to mount spaces using a specific Oneprovider is:

```
oneclient -H <PROVIDER_HOSTNAME> -t <ACCESS_TOKEN> <MOUNT_POINT>
```

In order to unmount your spaces, type:

```
oneclient -u MOUNT_POINT
```

The complete Oneclient documentation is located here: https://onedata.org/docs/doc/using_onedata/oneclient.html

23.5 Troubleshooting

If you are connecting to a provider service which does not have a globally trusted certificate, you will have to use `--insecure` option.

Frequently Asked Questions

Laniakea FAQs.

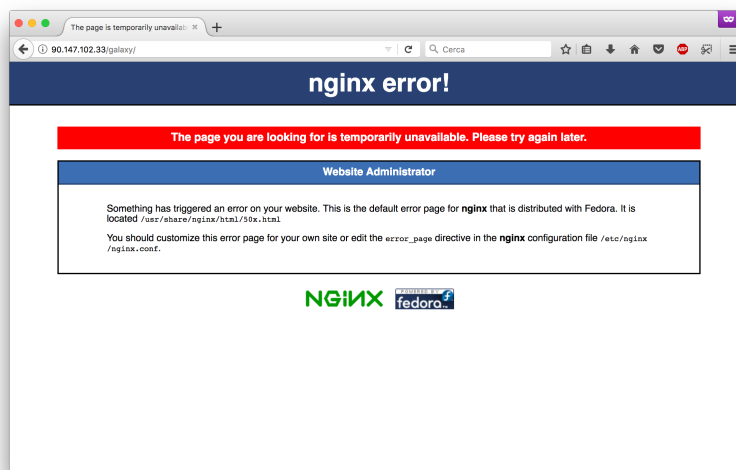
24.1 Recover Galaxy after Virtual Machine reboot

How to correctly restart Galaxy after a reboot of the Virtual Machine ?

After the boot procedure of your VM at least these services should already be up and running:

1. PostgreSQL
2. Proftpd
3. NGINX

Trying to connect to your Galaxy instance IP with a web browser you should see:



24.1.1 Step 1: Unlock encrypted storage

If your instance is not mounted on encrypted storage please skip this and go to [Step 2: One-command procedure](#)

To unlock the volume connect with SSH and type the command `sudo /usr/local/bin/luksctl open` followed by your passphrase.

```
$ sudo /usr/local/bin/luksctl open
Enter passphrase for /dev/disk/by-uuid/4ba96890-f914-46aa-b7a6-7e71f0846f43:

Name:                jzwoejuw
State:                ACTIVE
Read Ahead:          8192
Tables present:      LIVE
Open count:           1
Event number:         0
Major, minor:         252, 0
Number of targets:    1
UUID: CRYPT-LUKS1-4ba96890f91446aab7a67e71f0846f43-jzwoejuw

Encrypted volume: [ OK ]
```

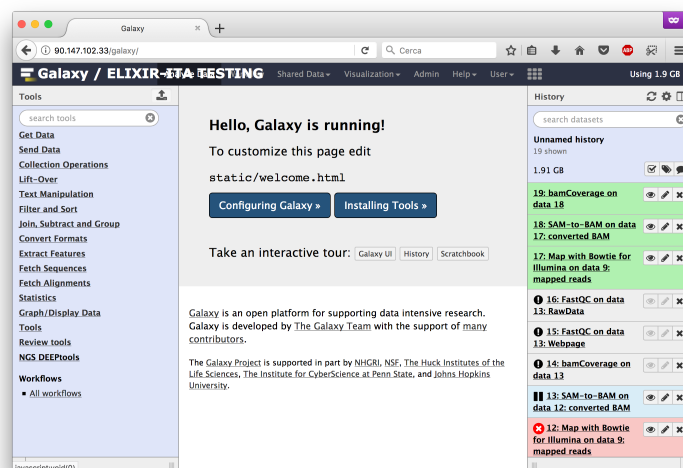
24.1.2 Step 2: One-command procedure

To start Galaxy you can just copy and paste this:

```
$ sudo su -c "wget -O - https://raw.githubusercontent.com/Laniakea-elixir-it/Scripts/
↪master/galaxy/recover.sh | bash" root

Probing /cvmfs/elixir-italy.galaxy.refdata... OK
Contacting Galaxy (wait for 10 seconds)...
Galaxy server on-line: [ OK ]
```

Warning: This may require few minutes, but should safely restart your Galaxy.



24.1.3 Advanced Step-by-step procedure

Warning: Please follow this instructions only if you know what you are doing!

Mount reference data

After encryption storage unlock (see section *Step 1: Unlock encrypted storage*), if you have it, you need now to mount your reference datasets:

```
$ sudo systemctl restart autofs
```

```
$ sudo cvmfs_config killall

Terminating cvmfs_config processes... OK
Terminating cvmfs2 processes... OK
Unmounting stale mount points... OK
Cleaning up run-time variable data... OK
Reloading autofs... OK
```

```
$ sudo cvmfs_config probe

Probing /cvmfs/elixir-italy.galaxy.refdata... OK
```

Start Galaxy

Finally, to correctly start all Galaxy services run: `sudo /usr/local/bin/galaxy-startup`

```
$ sudo /usr/local/bin/galaxy-startup

Loading Galaxy environment
Exporting environment variables
Check if galaxy is up and running, to avoid unuseful restart
status: 502
Galaxy unreachable
Check is supervisord is running
Starting the Galaxy production environment
/usr/lib/python2.7/site-packages/supervisor/options.py:383:
↳PkgResourcesDeprecationWarning: Parameters to load are deprecated. Call .resolve_
↳and .require separately.
return pkg_resources.EntryPoint.parse("x="+spec).load(False)

Galaxy start: [ OK ]
```

Galaxycloud Ansible Roles

Ansible automates Galaxy (postgresql, NGINX, uWSGI, proftpd) installation and configuration using YAML syntax.

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).

All roles can be easily installed through `ansible-galaxy`, for instance:

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

25.1 Ansible roles documentation

25.1.1 Galaxycloud

Install Galaxy Production environment. This role has been specifically developed to be used for the ELIXIR-IIB use case in the INDIGO-DataCloud project.

Requirements

This ansible role supports CentOS 7, Ubuntu 14.04 Trusty and Ubuntu 16.04 Xenial

Note: Minimum ansible version: 2.1.2.0

Role Variables

Path

`galaxy_instance_description`: set Galaxy brand

galaxy_user: set linux user to launch the Galaxy portal (default: galaxy).

GALAXY_UID: set user UID (default: 4001).

galaxy_FS_path: path to install Galaxy (default: /home/galaxy).

galaxy_directory: Galaxy directory (usually galaxy or galaxy-dist, default galaxy).

galaxy_install_path: Galaxy installation directory (default: /home/galaxy/galaxy).

galaxy_config_path: Galaxy config pat location.

galaxy_config_file: Galaxy primary configuration file.

galaxy_venv_path: Galaxy virtual environment directory (usually located to <galaxy_install_path>/venv).

galaxy_custom_config_path: Galaxy custom configuration files path (default: /etc/galaxy).

galaxy_custom_script_path: Galaxy custom script path (default: /usr/local/bin).

galaxy_log_path: log file directory (default: /var/log/galaxy).

galaxy_instance_url: instance url (default: http://<ipv4_address>/galaxy/).

galaxy_instance_key_pub: instance ssh public key to configure <galaxy_user> access.

galaxy_lrms: enable Galaxy virtual elastic cluster support. Currently supported local and slurm (default: local, possible values: local, slurm).

main options

GALAXY_VERSION: set Galaxy version (e.g. master, release_17.01, release_17.05...).

create_galaxy_admin: if true the administrator user will be created (default: true).

GALAXY_ADMIN_USERNAME: Galaxy administrator username.

GALAXY_ADMIN_PASSWORD: Galaxy administrator password.

GALAXY_ADMIN_API_KEY: Galaxy administrator API_KEY. <https://wiki.galaxyproject.org/Admin/API>. Please note that this key acts as an alternate means to access your account, and should be treated with the same care as your login password. To be changed by the administrator.(default value: GALAXY_ADMIN_API_KEY)

GALAXY_ADMIN_EMAIL: Galaxy administrator e-mail address

Galaxy configuration

export_dir: Galaxy userdata are stored here (default: /export).

tool_deps_path: change tool dependency directory (default: {{ export_dir }}/tool_deps)

use_conda: enable Conda (default: true).

job_work_dir: change job_working_dir path. Due to a current limitation in conda, the total length of the conda_prefix and the job_working_directory path should be less than 50 characters! (default: {{ export_dir }}/job_work_dir).

conda_prefix: change conda prefix directory (default: {{ export_dir }}/_conda).

conda_channels: change conda channels (default: iuc,bioconda,r,defaults,conda-forge).

update_ucsc: update UCSC genome database (default: ``true). A monthly cron job is added to keep update ucsc genome db.

`fast_update`: force database update by copying cached files (default: `true`).

`use_pbkdf2`: enable pbkdf2 cryptography (default: `true`).

Postgres database details

`postgresql_version`: set postgres version to be installed (current default: 9.6).

`galaxy_db_dir`: change galaxy database directory to store jobs results (default: `{{export_dir}}/galaxy/database`).

`galaxy_db_port`: set postgres port (default: 5432).

`galaxy_db_passwd`: set database password. By default it is generated a random password 20 characters long.

`set_pgsql_random_password`: if set to `false` the role takes the password specified through `galaxy_db_passwd` variable (default: `true`).

NGINX

`nginx_upload_store_path`: set nginx upload dataset directory (default: `{{galaxy_db_dir}}/tmp/nginx_upload_store`).

https mode

The Galaxy portal runs through an *nginx* http proxy by default. The following variables enable you to set nginx in https mode:

```
nginx_https: true
ssl_cert: /etc/certs/cert.pem
ssl_key: /etc/certs/key.pem
ssl_dhparam: /etc/certs/dhparam.pem
```

If `nginx_https` is set to `true`, the other ssl variables are required. You can either request a signed trusted certificate or generated self-signed certificate. An ansible role to generate self-signed certificate can be found in <https://galaxy.ansible.com/LIP-Computing/ssl-certs/>

PROFTPD

`proftpd_welcome`: set proftpd welcome message (default: `galaxy ftp server`).

`proftpd_conf_path`: set proftpd configuration file path.

`proftpd_db_user`: set proftpd database user (default: `galaxyftp`).

`proftpd_db_passwd`: set postgresql database password. By default it is generated a random password 20 characters long.

`proftpd_files_path`: set proftpd upload directory (default: `{{galaxy_db_dir}}/ftp`).

`proftpd_ftp_port`: set proftpd port (default: 21).

`proftpd_passive_port_low`: set passive port range minimum (default: 30000).

`proftpd_passive_port_high`: set passive port range maximum (default: 40000).

`set_proftpd_random_password`: if set to `false` the role takes the password specified through `proftpd_db_passwd` variable (default: `true`).

Init system

Currently this role support `supervisord` and `systemd/upstart` to start Galaxy services. `init_type`: if set to `supervisord`, it use to manage Galaxy. If set to `init` `systemd/upstart` is used to start Galaxy.

It is possible to exploit `supervisord` to manage `postgreSQL`, `NGINX` and `proftpd` setting to `true` the following variables. To run this role on docker container you have to set them to `true`. `supervisor_manage_postgres`: enable `supervisord` `postgresql` management (default: `false`).

`supervisor_manage_nginx`: enable `supervisord` `nginx` management (default: `false`).

`supervisor_manage_proftpd`: enable `supervisord` `proftpd` management (default: `false`).

Advanced storage configuration

`enable_storage_advanced_options`: this option, `false` by the default, has to be set to `true` only if you run the ansible role `indigo-dc.galaxycloud-os`, for advanced path configuration, `onedata` and filesystem encryption support. More details here: [Galaxycloud-os](#) (default: `false`).

Example Playbook

Including an example of how to use your role (for instance, with variables passed in as parameters) is always nice for users too:

```
- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud
      GALAXY_ADMIN_EMAIL: "admin@elixir-italy.org"
      GALAXY_ADMIN_USERNAME: "admin"
      GALAXY_VERSION: "release_17.05"
      galaxy_instance_key_pub: "your_public_key"
      galaxy_instance_description: "INDIGO-CNR test"
```

Install Galaxy setting `postgresql` passwords:

```
- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud
      GALAXY_ADMIN_EMAIL: "admin@elixir-italy.org"
      GALAXY_ADMIN_USERNAME: "admin"
      GALAXY_VERSION: "release_17.05"
      galaxy_instance_key_pub: "your_public_key"
      galaxy_instance_description: "INDIGO-CNR test"
      set_pgsql_random_password: false
      galaxy_db_passwd: 'galaxy'
      set_proftpd_random_password: false
      proftpd_db_passwd: 'galaxy'
```

Setup Galaxy Docker container. The role, using `ansible`, automatically recognize the virtual platform (virtual machine or Docker container).

```
- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud
      GALAXY_ADMIN_EMAIL: "admin@elixir-italy.org"
      GALAXY_ADMIN_USERNAME: "admin"
      GALAXY_VERSION: "release_17.05"
      galaxy_instance_key_pub: "your_public_key"
      galaxy_instance_description: "INDIGO-CNR test"
      supervisor_manage_postgres: "True"
      supervisor_manage_nginx: "True"
      supervisor_manage_proftpd: "True"
```

License

Apache Licence v2

References

Galaxy: <https://galaxyproject.org/>

Apache licence: <http://www.apache.org/licenses/LICENSE-2.0>

25.1.2 Galaxycloud-os

This role provides advanced storage options for Galaxy instances.

Warning: Run `indigo-dc.galaxycloud-os` before `indigo-dc.galaxycloud`, setting the variable `enable_storage_advanced_options` to `true`.

It is possible to select three different storage options using the `os_storage` ansible role variable.

Storage provider	Description
iaas	IaaS block storage volume is attached to the instance and Galaxy is configured.
onedata	Onedata space is mounte through oneclient and Galaxy is configured.
encryption	IaaS block storage volume is encrypted with aes-xts-plain64 algorithm using LUKS.

Path configuration for Galaxy is then correctly set, depending on the storage solution selected, replacing the `indigo-dc.galaxycloud` path recipe (with the `enable_storage_advanced_options` set to `true`).

The role exploits the `galaxyctl_libs` (see *Galaxyctl libraries*) for LUKS and onedata volumes management .

LUKS encryption

For a detailed description of LUKS encryption used and scripts, see section *Storage encryption*.

Dependencies

For LUKS encryption the ansible role install `cryptsetup`.

For onedata reference data provider, the role depends on `indigo-dc.oneclient` role, to install oneclient:

```
- hosts: servers
  roles:
    - role: indigo-dc.oneclient
      when: os_storage == 'onedata'
```

Variables

The Galaxy path variables are the same of indigo-dc.galaxycloud.

Path

galaxy_user: set linux user to launch the Galaxy portal (default: galaxy).

GALAXY_UID: set user UID (default: 4001).

galaxy_FS_path: path to install Galaxy (default: /home/galaxy).

galaxy_directory: Galaxy directory (usually galaxy or galaxy-dist, default galaxy).

galaxy_install_path: Galaxy installation directory (default: /home/galaxy/galaxy).

export_dir: Galaxy userdata are stored here (default: /export).

galaxy_custom_config_path: Galaxy custom configuration files path (default: /etc/galaxy).

galaxy_custom_script_path: Galaxy custom script path (default: /usr/local/bin).

galaxy_log_path: log file directory (default: /var/log/galaxy).

galaxy_instance_key_pub: instance ssh public key to configure <galaxy_user> access.

galaxy_lrms: enable Galaxy virtual elastic cluster support. Currently supported local and slurm (default: local, possible values: local, slurm).

Main options

GALAXY_ADMIN_EMAIL: Galaxy administrator e-mail address

Isolation specific vars

os_storage: takes three possible values:

1. IaaS: standard IaaS block storage volume.
2. onedata: Onedata space is mounted for user data.
3. download: IaaS block storage volume encrypted with aes-xts-plain64 is mounted.

Onedata

onedata_dir: onedata mountpoint. (default: /onedata).

Note: Once onedata space is mounted, files existing before mount operation, will not be available until volume umount. For this reason we set it to /onedata to a differet path.

onedatactl_config_file: set onedatactl config file (default: {{ galaxy_custom_config_path }}/onedatactl.ini).

userdata_oneprovider: set onedata oneprovider.

userdata_token: set onedata access token.

userdata_space: set space name.

Encryption

luks_lock_dir: set luks lock file directory (default: /var/run/fast_luks).

luks_success_file: set success file. It signals to ansible to proceed (default: /var/run/fast-luks.success).

luks_log_path: set LUKS log path (default: /var/log/galaxy).

luks_config_file: set luksctl configuration file (default: /etc/galaxy/luks-cryptdev.ini).

wait_timeout: time to wait encryption password (default: 5 hours).

mail_from: set mail from field (default: GalaxyCloud@elixir-italy.org).

mail_subject: with the instructions to access and encrypt the volume is sent to the user (default: [ELIXIR-ITALY] GalaxyCloud encrypt password).

LUKS specific variables

cipher_algorithm: set cipher algorithm (default: aes-xts-plain64).

keysize: set key size (default: 256).

hash_algorithm: set hash algorithm (default: sha256).

device: set device to mount (default: /dev/vdb)

cryptdev: set device mapper name (default: /dev'crypt).

mountpoint: set mount point. Usually the same of export_dir (default: {{ export_dir }}).

filesystem: set file system (default: ext4).

Create block file:

1. https://wiki.archlinux.org/index.php/Dm-crypt/Device_encryption
2. https://wiki.archlinux.org/index.php/Dm-crypt/Drive_preparation
3. https://wiki.archlinux.org/index.php/Disk_encryption#Preparing_the_disk

Before encrypting a drive, it is recommended to perform a secure erase of the disk by overwriting the entire drive with random data.

To prevent cryptographic attacks or unwanted file recovery, this data is ideally indistinguishable from data later written by dm-crypt.

paranoic_mode: to enable block storage low level deletion set to true (default: false).

Example Playbook

IaaS configuration:

```
- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud-os
      os_storage: 'IaaS'
      GALAXY_ADMIN_EMAIL: "admin@server.com"
      galaxy_instance_key_pub: '<your_ssh_public_key>'

    - role: indigo-dc.galaxycloud
      GALAXY_ADMIN_EMAIL: "admin@server.com"
      GALAXY_ADMIN_USERNAME: "admin"
      GALAXY_VERSION: "release_17.05"
      galaxy_instance_key_pub: "<your_ssh_public_key>"
      enable_storage_advanced_options: true
```

Onedata configuration:

```
- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud-os
      os_storage: 'onedata'
      GALAXY_ADMIN_EMAIL: "admin@server.com"
      userdata_provider: 'oneprovider2.cloud.ba.infn.it'
      userdata_token: '<your_access_token>'
      userdata_space: '<your_onedata_space>'
      galaxy_instance_key_pub: '<your_ssh_public_key>'

    - role: indigo-dc.galaxycloud
      GALAXY_ADMIN_EMAIL: "admin@server.com"
      GALAXY_ADMIN_USERNAME: "admin"
      GALAXY_VERSION: "release_17.05"
      galaxy_instance_key_pub: "<your_ssh_public_key>"
      enable_storage_advanced_options: true
```

LUKS configuration:

```
- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud-os
      os_storage: 'encryption'
      GALAXY_ADMIN_EMAIL: "admin@server.com"
      galaxy_instance_key_pub: '<your_ssh_public_key>'

    - role: indigo-dc.galaxycloud
      GALAXY_ADMIN_EMAIL: "admin@server.com"
      GALAXY_ADMIN_USERNAME: "admin"
      GALAXY_VERSION: "release_17.05"
      galaxy_instance_key_pub: "<your_ssh_public_key>"
      enable_storage_advanced_options: true
```

References

Galaxy: <https://galaxyproject.org/>

Apache licence: <http://www.apache.org/licenses/LICENSE-2.0>

25.1.3 Galaxycloud-tools

This Ansible role is for automated installation of tools from a Tool Shed into Galaxy. The role use the path scheme from the role `indigo-dc.galaxycloud`

When run, this role will create 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 on an external repository: <https://github.com/indigo-dc/Galaxy-flavors-recipes>.

The role automatically clone this repository to install tools.

Requirements

This ansible role supports CentOS 7, Ubuntu 14.04 Trusty and Ubuntu 16.04 Xenial

Note: Minimum ansible version: 2.1.2.0

Role Variables

Path

`galaxy_instance_description`: set Galaxy brand

`galaxy_user`: set linux user to launch the Galaxy portal (default: `galaxy`).

`GALAXY_UID`: set user UID (default: 4001).

`galaxy_FS_path`: path to install Galaxy (default: `/home/galaxy`).

`galaxy_directory`: Galaxy directory (usually `galaxy` or `galaxy-dist`, default `galaxy`).

`galaxy_install_path`: Galaxy installation directory (default: `/home/galaxy/galaxy`).

`galaxy_config_path`: Galaxy config pat location.

`galaxy_config_file`: Galaxy primary configuration file.

`galaxy_venv_path`: Galaxy virtual environment directory (usually located to `<galaxy_install_path>/venv`).

`galaxy_custom_config_path`: Galaxy custom configuration files path (default: `/etc/galaxy`).

`galaxy_custom_script_path`: Galaxy custom script path (default: `/usr/local/bin`).

`galaxy_log_path`: log file directory (default: `/var/log/galaxy`).

`galaxy_instance_url`: instance url (default: `http://<ipv4_address>/galaxy/`).

`galaxy_instance_key_pub`: instance ssh public key to configure `<galaxy_user>` access.

Main options

`GALAXY_ADMIN_API_KEY`: Galaxy administrator API_KEY. <https://wiki.galaxyproject.org/Admin/API>. Please note that this key acts as an alternate means to access your account, and should be treated with the same care as your login password. To be changed by the administrator.(default value: `GALAXY_ADMIN_API_KEY`)

`galaxy_tools_tool_list_files`: a list of yml files that list the tools to be installed.

`galaxy_tools_base_dir`: base dir to install installation script (default: `/tmp`).

`galaxy_flavor`: galaxy flavor to install. Each flavor corresponds to a directory hosted here: <https://github.com/indigo-dc/Galaxy-flavors-recipes> (default: `galaxy-no-tools`).

`lock_file_path`: add lock file to avoid role re-run during recipe update. To re-run the role remove the lock file `{{lock_file_path}}/indigo-dc.galaxycloud-tools.lock` (default: `/var/run`).

`install_workflows`: install workflows (default: ```false`).

`install_data_libraries`: install data libs (default: ```false`).

`install_interactive_tours`: enable interactive tours installation (default: ```false`).

`export_dir`: Galaxy userdata are stored here (default: `/export`).

This role exploits a lite version of galaxy to install data-libraries. It install dataset in `/home/galaxy/galaxy/database/files/000` by defaults. If the default galaxy database directory is different you have two options. By default the role read `galaxy.ini` and move datasets to `file_path` dir:

```
move_datasets: true
set_dataset_dest_dir: false
```

Otherwise you can set the destination directory. `Dataset_dest_dir` must exist, since the role will not create it.

```
move_datasets: true
set_dataset_dest_dir: true
dataset_dest_dir: '/path/to/dir'
```

Defaults values:

`move_datasets: true`

`set_dataset_dest_dir: true`

`dataset_dest_dir: /path/to/dir`

`add_more_assets`: add custom resources (i.e. visualisations plugins, custom web pages, etc.). Since there is no a standard way to retrieve and install visualisation plugin, we keep this recipes external and implement a common interface to insall these resources (default: `false`).

Create bootstrap user

if an apy key is not present on galaxy, a new user is created to istall tools and removed. This is a very basic implementation. Advanced one is located here: https://raw.githubusercontent.com/indigo-dc/ansible-galaxy-tools/master/files/manage_bootstrap_user.py Currently, to create it, few informations are needed: - galaxy installation path - `galaxy_database_connection` - and `pbkdf2` enabled

`create_bootstrap_user: false`

`galaxy_database_connection: postgresql://galaxy:galaxy@localhost:5432/galaxy`

```
use_pbkdf2: true
bootstrap_user_mail: admin@server.com
bootstrap_user_name: admin
bootstrap_user_password: password
```

By default, the api key is random-generated, overwriting the `galaxy_admin_api_key` variable assignment. You can set it to a defined value, by setting this `create_random_api_key` to `false`. `create_random_api_key: true`

Example Playbook

Including an example of how to use your role (for instance, with variables passed in as parameters) is always nice for users too:

```
- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud-tools
      galaxy_flavor: "galaxy-rna-workbench"
      galaxy_admin_api_key: "ADMIN_API_KEY"
      when: galaxy_flavor != "galaxy-no-tools"
```

Example Sources list

The role takes a sources list as input. Sources list recipe is used to describe Galaxy resources, tools, reference data, workflow, data libraries and/or visualization plugin to install.

The role always install tools:

```
- name: "Set {{galaxy_flavor}} resources"
  set_fact:
    galaxy_tools_tool_list_files:
      - '{{galaxy_tools_base_dir}}/Galaxy-flavors-recipes/{{galaxy_flavor}}/tool-list-
        ↳example.yml'
```

You can enable workflows installation setting the variable `install_workflows` to `true`, then insert the directory containing workflows:

```
# set path to workflow files
# ephemeris takes the workflow path to install workflows
- name: "Set {{galaxy_flavor}} workflows resources"
  set_fact:
    install_workflows: true
    galaxy_tools_workflow_list_path:
      - '{{galaxy_tools_base_dir}}/Galaxy-flavors-recipes/{{galaxy_flavor}}/workflow'
```

The same goes for data libraries. You have to enable the installation, setting `install_data_libraries` to `true`, then the yaml recipe path

```
# set yaml recipes
# ephemeris takes single files as argument
- name: "Set {{galaxy_flavor}} data library resources"
  set_fact:
    install_data_libraries: true
```

(continues on next page)

(continued from previous page)

```
galaxy_tools_data_library_list_files:
  - '{{galaxy_tools_base_dir}}/Galaxy-flavors-recipes/{{galaxy_flavor}}/library_
↪data.yaml'
```

To enable tours set `install_interactive_tours` to `true` and the tours path:

```
# set galaxy tours path
# the whole dir is copied to galaxy/config/plugins/tours/
- name: "Set {{galaxy_flavor}} tours resources"
  set_fact:
    install_interactive_tours: true
    galaxy_tools_interactive_tour_list_path:
      - '{{galaxy_tools_base_dir}}/Galaxy-flavors-recipes/{{galaxy_flavor}}/tours'
```

Finally, it is possible to install external resources, like visualisation plugins, setting `add_more_assets` to `true`:

```
# set more resources to be installed
# like visualisation plugins.
# since there is no a standard way to retrieve and install
# visualisation plugin, we keep this recepie external.
- name: "Install visualisation plugins"
  set_fact:
    add_more_assets: true
    galaxy_tools_assets_recipe_list_files:
      - '{{galaxy_tools_base_dir}}/Galaxy-flavors-recipes/{{galaxy_flavor}}/
↪visualisations.yml'
```

Example Tool list

For each tool you want to install, you must provide tool name and owner and one between `tool_panel_section_id` and `tool_panel_section_label` in the yaml tool list.

```
---
api_key: <Admin user API key from galaxy_instance>
galaxy_instance: <Galaxy instance IP>

tools:
- name: fastqc
  owner: devteam
  tool_panel_section_label: 'Tools'
  install_resolver_dependencies: True
  install_tool_dependencies: False

- name: 'bowtie_wrappers'
  owner: 'devteam'
  tool_panel_section_label: 'Tools'
  install_resolver_dependencies: True
  install_tool_dependencies: False
```

License

Apache Licence v2

25.1.4 Galaxycloud-tooldeps

This role fix tools installation done using the role `indigo-dc.galaxycloud-tools`, taking into account the Galaxy version.

Example Playbook

Including an example of how to use your role (for instance, with variables passed in as parameters) is always nice for users too:

```
::
```

- hosts: servers roles:
 - role: indigo-dc.galaxycloud-tooldeps

License

Apache Licence v2

25.1.5 Galaxycloud-refdata

Reference data ansible role for `indigo-dc.galaxycloud`. The role provides reference data and the corresponding galaxy configuration.

Currently, three reference data source are supported:

Provider	Description
cvmfs	CernVM-FS repository is used to provide reference data. It is mounted to <code>/refdata</code> .
onedata	Onedata space hosting reference data is mounted to <code>/refdata</code> .
down-load	Reference data are downloaded in <code>/refdata</code> (requires >100GB free space available on <code>/refdata</code> directory).

Moreover, this role, exploiting the python library `Ephemeris`, is able to check which tools have been installed through `indigo-dc.galaxy-tools` ansible role, and returns

- the list of installed tools stored in `/var/log/galaxy/galaxy-installed-tool-list.yml` in yaml format
- the list of missing tools stored in `/var/log/galaxy/galaxy-missing-tool-list.yml` in yaml format

Note: This option has been introduced for galaxy tools automatic deployment. If you need to install and configure reference data, you can disable it using `galaxy_flavor: "galaxy-no-tools"`.

Requirements

When a CernVM-FS server is used, the role run the `indigo-dc.cvmfs-client` ansible role as dependency to install and configure the `cvmfs` client.

If the role use `onedata` to provide reference data, `onedata` command line tool `oneclient` needs to be installed on your system. In this case, the role is going to depend on `indigo-dc.oneclient` role and it will install `oneclient` automatically.

Finally, if the download option is selected, the role exploits a python script to download the reference data, which depends on python-pycurl (which is automatically installed).

Role Variables

`galaxy_flavor`: if different from ‘galaxy-no-tools’ the role will check if all tools installed using <https://github.com/indigo-dc/ansible-galaxy-tools> have been correctly installed. Possible `galaxy_flavor` values with the corresponding recipes are reported here: [Galaxy ShedTools](#) (default: `galaxy-no-tools`).

`get_refdata`: enable reference data configuration. If set to `false` this variable disable reference data configuration (default: `true`).

`refdata_provider_type`: takes three possible values:

1. `cvmfs`: CernVM-FS repository with reference data is mounted
2. `onedata`: Onedata space with reference data is mounted
3. `download`: Reference data download

`refdata_repository_name`: onedata space, CernVM-FS repository name or subdirectory to download local reference data.

cvmfs variables

`refdata_cvmfs_server_url`: set CernVM-FS server (stratum 0 or Replica) address without ‘`http://`’ string, e.g. single ip address.

`refdata_cvmfs_repository_name`: set a different `cvmfs` repository name, overwriting the default option, which point to `refdata_repository_name` (e.g. `elixir-italy.galaxy.refdata`).

`refdata_cvmfs_key_file`: SSH public key to mount the repository

`refdata_cvmfs_proxy_url`: proxy address (default `DIRECT`).

`refdata_cvmfs_proxy_port`: proxy port (default 80).

onedata variables

`refdata_provider`: set reference data oneprovider (e.g. `oneprovider2.cloud.ba.infn.it`).

`refdata_token` set reference data access token (e.g. `MDAxNWxvY2F00aW9uIG9uZXpzbmUKMDAzYmlkZW500aWZpZXIgeEx`).

`refdata_space`: set reference data space name.

download

```
at10: false # A. thaliana (TAIR 10)
at9: false # A. thaliana (TAIR 9)
dm2: false # D. melanogaster (dm2)
dm3: false # D. melanogaster (dm3)
hg18: false # H. sapiens (hg18)
hg19: false # H. sapiens (hg19)
hg38: false # H. sapeins (hg38)
mm10: false # M. musculus (mm10)
mm8: false # M. musculus (mm9)
```

(continues on next page)

(continued from previous page)

```
mm9: false # M. musculus (mm8)
sacCer1: false # S. cerevisiae (sacCer1)
sacCer2: false # S. cerevisiae (sacCer2)
sacCer3: true # S. cerevisiae (sacCer3)
```

Select which reference data genome has to be downloaded.

Dependencies

For cvmfs server reference data provider, the role depends on indigo-dc.cvmfs-client role, which takes as input parameters the CernVM-FS server location details (stratum 0 address, public key and mount point).

```
- hosts: servers
  roles:
    - role: indigo-dc.cvmfs-client
      server_url: '90.147.102.186'
      repository_name: 'elixir-italy.galaxy.refdata'
      cvmfs_public_key: 'elixir-italy.galaxy.refdata.pub'
      proxy_url: 'DIRECT'
      proxy_port: '80'
      cvmfs_mountpoint: '/refdata'
      when: refdata_provider_type == 'cvmfs'
```

For onedata reference data provider, the role depends on indigo-dc.oneclient role:

```
- hosts: servers
  roles:
    - role: indigo-dc.oneclient
      when: refdata_provider_type == 'onedata'
```

Example Playbook

- Configure Galaxy with CernVM-FS reference data volume.

```
- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud-refdata
      galaxy_flavor: 'galaxy-no-tools'
      get_refdata: true
      refdata_provider_type: 'cvmfs'
      refdata_cvmfs_server_url: '90.147.102.186'
      refdata_cvmfs_repository_name: 'elixir-italy.galaxy.refdata'
      refdata_cvmfs_key_file: 'elixir-italy.galaxy.refdata'
      refdata_cvmfs_proxy_url: 'DIRECT'
```

- Configure Galaxy with Onedata space for reference data.

```
- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud-refdata
      galaxy_flavor: "galaxy-no-tools"
      get_refdata: true
      refdata_provider: 'oneprovider2.cloud.ba.infn.it'
```

(continues on next page)

(continued from previous page)

```

refdata_token:
→ 'MDAxNWxvY2F0OaW9uIG9uZXPvbmUKMDAzYmlkZW50OaWZpZXIgeExqMi00xdFN3YVp1VWIxM1dFSzRoNEdkb2x3cXVwTm
→ '
refdata_space: 'elixir-italy.galaxy.refdata'

```

- Download (all available) reference data. You can select which one download.

```

- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud-refdata
      galaxy_flavor: 'galaxy-no-tools'
      get_refdata: true
      refdata_repository_name: 'elixir-italy.galaxy.refdata'
      refdata_provider_type: 'download'
      atl10: true # A. thaliana (TAIR 10)
      at9: true # A. thaliana (TAIR 9)
      dm2: true # D. melanogaster (dm2)
      dm3: true # D. melanogaster (dm3)
      hg18: true # H. sapiens (hg18)
      hg19: true # H. sapiens (hg19)
      hg38: true # H. sapeins (hg38)
      mm10: true # M. musculus (mm10)
      mm8: true # M. musculus (mm9)
      mm9: true # M. musculus (mm8)
      sacCer1: true # S. cerevisiae (sacCer1)
      sacCer2: true # S. cerevisiae (sacCer2)
      sacCer3: true # S. cerevisiae (sacCer3)

```

References

Galaxy project: <https://galaxyproject.org>

CernVM-FS: <http://cvmfs.readthedocs.io/en/stable/index.html>

Onedata: <https://groundnuty.gitbooks.io/onedata-documentation/content/index.html/>

25.1.6 Galaxycloud-fastconfig

Ansible role for Galaxy fast configuration on Virtual Manachines with Galaxy already installed using indigo.dc-galaxycloud role.

Current indigo-dc.galaxycloud (and then Galaxy) configuration is the following:

```

- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud
      GALAXY_ADMIN_EMAIL: "admin@elixir-italy.org"
      GALAXY_ADMIN_USERNAME: "admin"
      GALAXY_VERSION: "release_17.05"
      galaxy_instance_key_pub: "ssh-rsa ..."
      set_pgsql_random_password: false # postgres password is fixed: galaxy
      set_proftpd_random_password: false # proftpd database password is fixed: galaxy
      galaxy_db_dir: '/home/galaxy/galaxy/database'
      tool_deps_path: '/home/galaxy/tool_deps'

```

(continues on next page)

(continued from previous page)

```
conda_prefix: '/home/galaxy/tool_deps/_conda'
job_work_dir: 'database/jobs_directory'
```

Final Galaxy configuration, i.e. galaxycloud + galaxycloud-fastconfig is the same of galaxycloud standalone.

Example Playbook

Including an example of how to use your role (for instance, with variables passed in as parameters) is always nice for users too:

```
- hosts: servers
  roles:
    - role: indigo-dc.galaxycloud-fastconfig
      GALAXY_ADMIN_EMAIL: "mymail@example.com"
      GALAXY_ADMIN_USERNAME: "myuser"
      galaxy_instance_description: "mygalaxy"
      galaxy_instance_key_pub: "ssh-rsa ..."
```

License

Apache Licence 2

25.1.7 Galaxy-tools (deprecated)

This ansible role has been cloned from the official [galaxyproject.galaxy-tools](#), with small changes.

This Ansible role is for automated installation of tools from a Tool Shed into Galaxy.

When run, this role will create an execution environment, install ephemeris and invoke the shed-install command to install desired tools into Galaxy. The list of tools to install is provided in files/tool_list.yaml file.

Example Playbook

To use the role, you need to create a playbook and include the role in it.

```
- hosts: localhost
  gather_facts: False
  connection: local
  vars:
    galaxy_tools_tool_list_files: [ "files/sample_tool_list.yaml" ]
    galaxy_tools_galaxy_instance_url: http://127.0.0.1:8080/
    # galaxy_tools_api_key: <API key for Galaxy admin user>
  roles:
    - indigo-dc.galaxy-tools
```

Example Tool list

For each tool you want to install, you must provide tool name and owner and one between tool_panel_section_id and tool_panel_section_label in the yaml tool list.

```
---
api_key: <Admin user API key from galaxy_instance>
galaxy_instance: <Galaxy instance IP>

tools:
- name: fastqc
  owner: devteam
  tool_panel_section_label: 'Tools'
  install_resolver_dependencies: True
  install_tool_dependencies: False

- name: 'bowtie_wrappers'
  owner: 'devteam'
  tool_panel_section_label: 'Tools'
  install_resolver_dependencies: True
  install_tool_dependencies: False
```

Variables

Required variables

Only one of the two variables is required (if both are set, the API key takes precedence and a bootstrap user is not created):

`galaxy_tools_api_key`: the Galaxy API key for an admin user on the target Galaxy instance (not required if the bootstrap user is being created)

`galaxy_tools_admin_user_password`: a password for the Galaxy bootstrap user (required only if `galaxy_install_bootstrap_user` variable is set)

Optional variables

See `defaults/main.yml` for the available variables and their defaults.

Control flow variables

The following variables can be set to either yes or no to indicate if the given part of the role should be executed:

`galaxy_tools_install_tools`: (default: yes) whether or not to run the tools installation script

`galaxy_tools_create_bootstrap_user`: (default: no) whether or not to create a bootstrap Galaxy admin user

`galaxy_tools_delete_bootstrap_user`: (default: no) whether or not to delete a bootstrap Galaxy admin user

References

`galaxyproject.galaxy-tools`: <https://github.com/galaxyproject/ansible-galaxy-tools>

`Ephemeris`: <https://github.com/galaxyproject/ephemeris>

`galaxy-tools-playbook`: <https://github.com/afgane/galaxy-tools-playbook>

25.1.8 Cvmfs-server

Ansible role to install CernVM FS Server.

This role has been create to be general, but it is used to create Galaxy Reference Data read-only repository. To populate Stratum Zero repository with Reference data see section [Build cvmfs server for reference data](#).

Requirements

This ansible role is compatible with both CentOS 7 and Ubuntu 16.04 Xenial.

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.

Python is required on host to run ansible: `sudo apt-get install python`

The apt ansible module requires the following packages on host to run:

- python-apt (python 2)

Variables

`repository_name`: set the cvmfs repository name (default: `elixir-italy.galaxy.refdata`).

`stratum_zero`: set your domain or your ip address. (default: `{{ ansible_default_ipv4.address }}`).

`repository_url`: set the cvmfs repository url (default: `http://{{ stratum_zero }}/cvmfs/{{ repository_name }}`).

Example Playbook

The role is able to detect the server ip address automatically, through ansible. To customize your server, set your repository name.

```
- hosts: servers
  roles:
    - role: indigo-dc.cvmfs-server
      repository_name: '<your_repository_name>'
```

Development

- S3 support and squid proxy server support is on-going.
- Replica server support is on-going.

References

Official cvmfs documentation: <http://cvmfs.readthedocs.io/en/stable/cpt-repo.html>

NIKHEF documentation: https://wiki.nikhef.nl/grid/Adding_a_new_cvmfs_repository

To Install CernVM FS Client: <https://github.com/indigo-dc/ansible-role-cvmfs-client>

To Create a CernVM Replica: <https://github.com/mtangaro/ansible-role-cvmfs-replica> (on-going)

25.1.9 Cvmfs_client

Ansible role to install CernVM-FS Client.

Requirements

Python is required on host to run ansible.

The apt ansible module requires the following packages on host to run:

- python-apt (python 2)

Variables

`server_url`: set cvmfs server url (e.g. ip address or domain).

`repository_name`: set cvmfs server repository name (default: `elixir-italy.galaxy.refdata`).

`cvmfs_server_url`: sert cvmfs server complete url (default: `'http://{{ server_url }}/cvmfs/{{ repository_name }}'`).

`cvmfs_public_key_path`: set path for cvmfs keys (default: `/etc/cvmfs/keys`).

`cvmfs_public_key`: set cvfms public key, usually `<repository_name.pub>` (defatul: `{{ repository_name }}.pub`).

`cvmfs_public_key_list_files`: list of `*.pub` files with the key to the repository to be mounted.

`public_key_src_path`: set cvmfs public key temporary path (default: `/tmp`).

`proxy_url`: set proxy name (default: `DIRECT`).

`proxy_port`: set proxy port (default: `80`).

`cvmfs_http_proxy`: set proxy complete url (default: `http://{{ proxy_url }}:{{ proxy_port }}`).

`cvmfs_mountpoint`: set cvmfs mount point (default: `/cvmfs`, for reference data `/refdata`). If set to `/cvmfs` the role will use `cvmfs_config probe` to mount the repository.

`add_fstab_entry`: add fstab entry to automatically mount the repository (default: `true`).

Example Playbook

The role takes as input parameters the CernVM-FS server location details (stratum 0 address, public key and mount point).

```
- hosts: servers
  roles:
    - role: indigo-dc.cvmfs-client
      server_url: '90.147.102.186'
      repository_name: 'elixir-italy.galaxy.refdata'
      cvmfs_public_key: 'elixir-italy.galaxy.refdata.pub'
      proxy_url: 'DIRECT'
      proxy_port: '80'
      cvmfs_mountpoint: '/refdata'
      when: refdata_provider_type == 'cvmfs'
```

References

Official cvmfs documentation: <http://cvmfs.readthedocs.io/en/stable/cpt-repo.html>

NIKHEF documentation: https://wiki.nikhef.nl/grid/Adding_a_new_cvmfs_repository



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. Then the *Infrastructure Manager* deploys the customized virtual infrastructure on IaaS Cloud deployment.

The *INDIGO PaaS Orchestrator* takes the deployment requests written in *TOSCA YAML Simple Profile v1.0*. To correctly orchestrate Galaxy deployment the following components are needed:

- Ansible roles: automate software installation and configuration (see section *Galaxycloud Ansible Roles*)
- Custom type: 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.

26.1 Custom types

26.1.1 GalaxyPortal

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

1. Galaxy instance input parameters:

```
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
```

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.

2. Specify LRMS, e.g. local, torque, slurm, sge, condor, mesos:

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

3. Ansible role installation through ansible-galaxy:

```
artifacts:
```

(continues on next page)

(continued from previous page)

```
galaxy_role:
  file: indigo-dc.galaxycloud
  type: tosa.artifacts.AnsibleGalaxy.role
```

4. Ansible role call with input parameters:

```
implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/master/
↳ 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 ] }
    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 ] }
```

5. The artifact is located on github [tosca-types/artifacts/galaxy/galaxy_install.yml](https://raw.githubusercontent.com/indigo-dc/tosca-types/master/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 }}"
```

26.1.2 GalaxyPortalAndStorage

GalaxyPortalAndStorage custom type inherits its properties from GalaxyPortal and extends its functionalities with onedata support and file system encryption:

1. Storage options properties:

```
os_storage:
  type: string
  description: Storage type (IaaS Block Storage (default), Onedaata, Filesystem,
↳ encryption)
  default: "IaaS"
  required: true
token:
  type: string
  description: Access token for onedata space
  default: "not_a_token"
  required: false
provider:
  type: string
  description: default OneProvider
  default: "not_a_provider_url"
  required: false
space:
```

(continues on next page)

(continued from previous page)

```

type: string
description: Onedata space
default: "galaxy"
required: false

```

2. Ansible roles: oneclient role is needed to install oneclient, while indigo-dc.galaxycloud-os is entrusted of file system encryption:

```

oneclient_role:
  file: indigo-dc.oneclient
  type: tosca.artifacts.AnsibleGalaxy.role
galaxy_os_role:
  file: indigo-dc.galaxycloud-os
  type: tosca.artifacts.AnsibleGalaxy.role
galaxy_role:
  file: indigo-dc.galaxycloud
  type: tosca.artifacts.AnsibleGalaxy.role

```

3. Ansible role call with input parameters:

```

implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/master/
↳artifacts/galaxy/galaxy_os_install.yml
inputs:
  os_storage: { get_property: [ SELF, os_storage ] }
  userdata_token: { get_property: [ SELF, token ] }
  userdata_oneprovider: { get_property: [ SELF, provider ] }
  userdata_space: { get_property: [ SELF, space ] }
  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 ] }

```

4. 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`.

26.1.3 GalaxyShedTool

1. Ansible Galaxy tools properties:

```
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
```

2. Galaxy is required:

```
::
```

requirements:

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

3. Indigo-dc.galaxy-tools role installation:

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

4. Ansible role call. Instance IP address is needed to install tools:

```
implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/master/
↳ artifacts/galaxy/galaxy_tools_configure.yml
inputs:
  galaxy_flavor: { get_property: [ SELF, flavor ] }
  galaxy_admin_api_key: { get_property: [ HOST, admin_api_key ] }
  instance_public_ip: { get_attribute: [ HOST, public_address, 0 ] }
```

5. The artifact checks if Galaxy is on-line before installing tools and downloads yaml tool list according to `galaxy_flavor` variable value.

```
---
- hosts: localhost
  connection: local
  gather_facts: False
  pre_tasks:
    - name: Wait Galaxy is up
      uri: url="http://{{ instance_public_ip }}/galaxy/"
      register: result
      until: result.status == 200
      retries: 30
      delay: 10
      when: galaxy_flavor != 'galaxy-no-tools'
    - name: Get tool-list
      get_url: url="https://raw.githubusercontent.com/indigo-dc/Galaxy-flavors-
↳ recipes/master/galaxy-flavors/{{ galaxy_flavor }}-tool-list.yml" dest="/tmp/"
```

(continues on next page)

(continued from previous page)

```

    when: galaxy_flavor != 'galaxy-no-tools'
vars:
  galaxy_tools_tool_list_files: [ "/tmp/{{ galaxy_flavor }}-tool-list.yml" ]
  galaxy_tools_galaxy_instance_url: "http://{{ instance_public_ip }}/galaxy/"
  galaxy_tools_api_key: "{{ galaxy_admin_api_key }}"
roles:
  - { role: indigo-dc.galaxy-tools, when: galaxy_flavor != 'galaxy-no-tools' }

```

Note: `gather_facts: False` is needed to properly set ansible variables.

26.1.4 GalaxyReferenceData

The ReferenceData custom type supports CernVM-FS, Onedata reference data volumes and reference data downloads.

1. ReferenceData input parameters:

```

properties:
  reference_data:
    type: boolean
    description: Install Reference data
    default: true
    required: true
  flavor:
    type: string
    description: name of the Galaxy flavor
    required: true
    default: galaxy-no-tools
  refdata_repository_name:
    type: string
    description: Onedata space name, CernVM-FS repository name or subdirectory
    ↳download name
    default: 'elixir-italy.galaxy.refdata'
    required: false
  refdata_provider_type:
    type: string
    description: Select Reference data provider type (Onedata, CernVM-FS or
    ↳download)
    default: 'onedata'
    required: false
  refdata_provider:
    type: string
    description: Oneprovider for reference data
    default: 'not_a_provider'
    required: false
  refdata_token:
    type: string
    description: Access token for reference data
    default: 'not_a_token'
    required: false
  refdata_cvmfs_server_url:
    type: string
    description: CernVM-FS server, replica or stratum-zero
    default: 'server_url'

```

(continues on next page)

(continued from previous page)

```

    required: false
  refdata_cvmfs_repository_name:
    type: string
    description: Reference data CernVM-FS repository name
    default: 'not_a_cvmfs_repository_name'
    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: /refdata
    required: false

```

2. Galaxy is required to install and configure reference data:

```

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

```

3. Cvmfs role is used to install cvmfs client if cvmfs is selected to provide reference data. Oneclient role is used to provide onedata support.

artifacts:

oneclient_role: file: indigo-dc.oneclient type: tosca.artifacts.AnsibleGalaxy.role

cvmfs_role: file: indigo-dc.cvmfs-client type: tosca.artifacts.AnsibleGalaxy.role

galaxy_role: file: indigo-dc.galaxycloud-refdata type: tosca.artifacts.AnsibleGalaxy.role

4. Ansible role call with paramteres:

```

implementation: https://raw.githubusercontent.com/indigo-dc/tosca-types/master/
↳ artifacts/galaxy/galaxy_redfata_configure.yml
inputs:
  get_refdata: { get_property: [ SELF, reference_data ] }
  galaxy_flavor: { get_property: [ SELF, flavor ] }
  refdata_repository_name: { get_property: [ SELF, refdata_repository_name ] }
  refdata_provider_type: { get_property: [ SELF, refdata_provider_type ] }
  refdata_provider: { get_property: [ SELF, refdata_provider ] }
  refdata_token: { get_property: [ SELF, refdata_token ] }
  refdata_cvmfs_server_url: { get_property: [ SELF, refdata_cvmfs_server_url ] }

```

(continues on next page)

(continued from previous page)

```

refdata_cvmfs_repository_name: { get_property: [ SELF, refdata_cvmfs_repository_
↪name ] }
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 ] }

```

5. Cvmfs public key is downloaded to mount cvmfs volume:

```

---
- hosts: localhost
  connection: local
  pre_tasks:
    - name: Get refdata-list
      get_url:
        url: 'https://raw.githubusercontent.com/indigo-dc/Reference-data-
↪galaxycloud-repository/master/cvmfs_server_keys/{{ refdata_cvmfs_key_file }}'
        dest: '/tmp'
  roles:
    - role: indigo-dc.galaxycloud-refdata

```

26.2 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:

```

(continues on next page)

(continued from previous page)

```

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
  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:

```

(continues on next page)

(continued from previous page)

```

    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
    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:

```

(continues on next page)

(continued from previous page)

```

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
    - 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:

```

(continues on next page)

(continued from previous page)

```
location: { get_input: export_dir }
device: hdb
```

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

26.3 Galaxy cluster template

The *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 tosca 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: tosca.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: tosca.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

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: slurms_server
        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
```

(continues on next page)

(continued from previous page)

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

Note: Note that to orchestrate Galaxy with SLURM we do not need new TOSCA custom types or anible 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](#).

27.1 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`

27.2 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
```

27.3 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`.

27.4 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`

27.4.1 Script usage

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

Option	Description
<code>-i,</code> <code>--input .</code>	Input genome list in <code>yaml</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 `yaml` 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
```

27.5 References

CernVM-FS stratum 0 documentation: <http://cvmfs.readthedocs.io/en/stable/cpt-repo.html>

Nikhef wiki: https://wiki.nikhef.nl/grid/Adding_a_new_cvmfs_repository

INDIGO PaaS Orchestrator

PaaS Orchestrator is the core component of the INDIGO PaaS layer. It collects high-level deployment requests from the software layer, and coordinates the resource or service deployment over IaaS platforms.

Web site: <https://www.indigo-datacloud.eu/paas-orchestrator>

Official GitBook documentation: <https://www.gitbook.com/book/indigo-dc/indigo-paas-orchestrator/details>

GiHub: <https://github.com/indigo-dc/orchestrator>

28.1 Orchest

Orchest is the indigo command line client.

Orchest: <https://github.com/indigo-dc/orchest>

Infrastructure Manager

The Infrastructure Manager (IM) is a tool that deploys complex and customized virtual infrastructures on IaaS Cloud deployments (such as AWS, OpenStack, etc.).

Official GitBook documentation: <https://www.gitbook.com/book/indigo-dc/im/details>

29.1 Deployments log

Deployment logs are available in `/var/tmp/.im/<im-id>/<deployment_ip>/ctxt_agent.log`. For instance:

```
# 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!!

INDIGO FutureGateway

Warning: FutureGateway deployment requires `https` and `java 8`. Currently the script exploits `openjdk 7`.
See section *Update to Java 8 - Appendix A*.

To correctly setup the FGW portal follow the instruction in the Ubuntu LTS 14.04 Server section [here](#) as super user:

```
# IP=$(ifconfig | grep -A 2 eth0 | grep inet\ addr | awk -F':' '{ print $2 }' | awk '
→{ print $1 }' | xargs echo)

# echo "$IP    futuregateway" >> /etc/hosts

# adduser --disabled-password --gecos "" futuregateway

# mkdir -p /home/futuregateway/.ssh

# chown futuregateway:futuregateway /home/futuregateway/.ssh

# wget https://github.com/indigo-dc/PortalSetup/raw/master/Ubuntu_14.04/fgSetup.sh

# chmod +x fgSetup.sh

# cat /dev/zero | ssh-keygen -q -N ""

# cat /root/.ssh/id_rsa.pub >> /home/futuregateway/.ssh/authorized_keys

# echo "#FGSetup remove the following after installation" >> /etc/sudoers

# echo "ALL    ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers
```

You can edit the `fgSetup.sh` script to point to specific FGW release or branch.

We are currently using master branch.

```
GITBASE=https://github.com/indigo-dc # GitHub base repository ↵
↪endpoint
GITBASERAW=https://raw.githubusercontent.com/indigo-dc # GitHub base for raw content
GITPORTALSETUP_NAME="PortalSetup" # PortalSetup git path name
GITPORTALSETUP_CLONE="PortalSetup.git" # PortalSetup clone name
GITPORTALSETUP_TAG="master" # PortalSetup tag name
GITFGAPISERVER_NAME="fgAPIServer" # fgAPIServer git path name
GITFGAPISERVER_CLONE="fgAPIServer.git" # fgAPIServer clone name
GITFGAPISERVER_TAG="master" # fgAPIServer tag name
GITFGAPISERVERDAEMON_NAME="APIServerDaemon" # APIServerDaemon git path name
GITFGAPISERVERDAEMON_CLONE="APIServerDaemon.git" # APIServerDaemon clone name
GITFGAPISERVERDAEMON_TAG="master" # APIServerDaemin clone tag ↵
↪name
```

Then:

```
# ./fgSetup.sh futuregateway futuregateway <your ssh port> $(cat /root/.ssh/id_rsa.
↪pub)
```

The ssh port is, usually, the 22.

Warning: FutureGateway token authentication requires https. Here, we are going to use Let's Encrypt certificates, as example. A quick guide is available [here](#).

30.1 Portal configuration

Start the portal:

```
# /etc/init.d/futuregateway start
```

The portal will available at `http(s)://<your_ip_address>:8080`

Note: FGW (re)start take a while!

Login with the mail configured during the wizard and `test` as password. Then set your new password and recovery question.

30.2 Apache configuration

https is mandatory for FutureGateway Token authentication.

Enable `http_proxy` and `ssl` modules on `apache2`

```
a2enmod ssl
a2enmod proxy_http
```

Port 443 must be opened.

In `/etc/apache2/sites-available/` create your `futuregateway.conf` file, setting

```

ServerName <your_server_name>

...

## SSL directives
SSLEngine on
SSLCertificateFile /path/to/cert.pem
SSLCertificateKeyFile /path/to/key.pem
SSLCertificateChainFile /path/to/chain.pem

...

```

then enable FGW:

```
a2ensite futuregateway.conf
```

and reload apache:

```
# service apache2 reload
```

Add to FGW configuration file `portal-ext.properties` the following lines:

```

web.server.protocol=https
web.server.https.port=443

```

and restart FGW:

```
# /etc/init.d/futuregateway restart
```

Note: To create your signed certificate with Let's Encrypt: <https://github.com/maricaantonacci/slam/blob/master/gitbook/create-custom-keystore.md>

30.3 IAM integration

Iam portlets for the FGW portal are available on github: <https://github.com/mtangaro/fgw-elixir-italy/tree/master/iam-modules>

Put the portlets in the `/home/futuregateway/FutureGateway/deploy/`, FGW will upload them automatically, moving them in `/home/futuregateway/FutureGateway/osgi/modules/`.

You can follow this instructions to set it up: <https://github.com/indigo-dc/LiferayPlugIns/blob/master/doc/admin.md>.

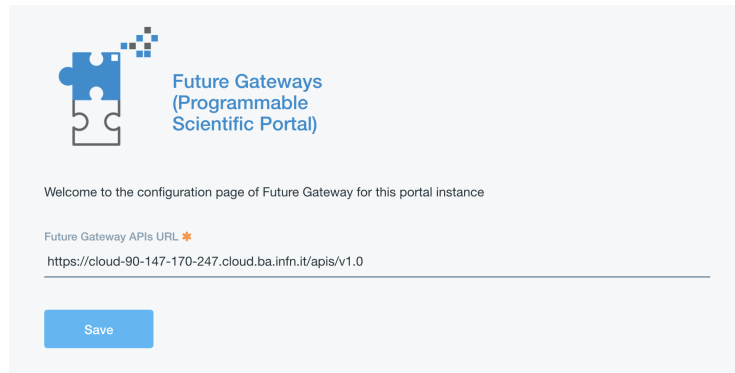
The option `javascript.fast.load=false` has to be set in `/home/futuregateway/FutureGateway/portal-ext.properties`.

30.4 Administrator portlet

The administrator portlet is here: <https://github.com/mtangaro/fgw-elixir-italy/tree/master/admin-modules>

Once uploaded, the Future Gateway APIs URL is `https://hostname/apis/v1.0`.

The next thing is the configuration of PTV (Portal Token Validator). This is a service which FG API server uses for token validation.

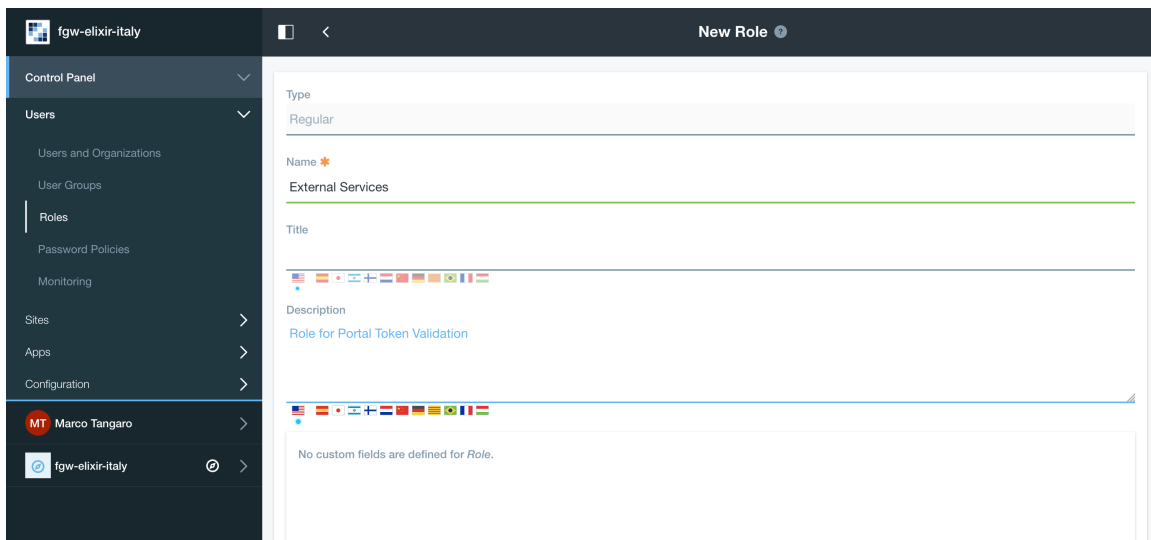


Specific Liferay user and role are needed to exploit PTV.

Note: This step is not mandatory, since you can still configure your portal owner for token validation, since it has all required permissions and is registered to IAM.

But, if you change IAM client you have to create a new user and change the PTV configuration. This configuration avoid this.

Create a new Role named `External Services` and give it IAM token permissions:



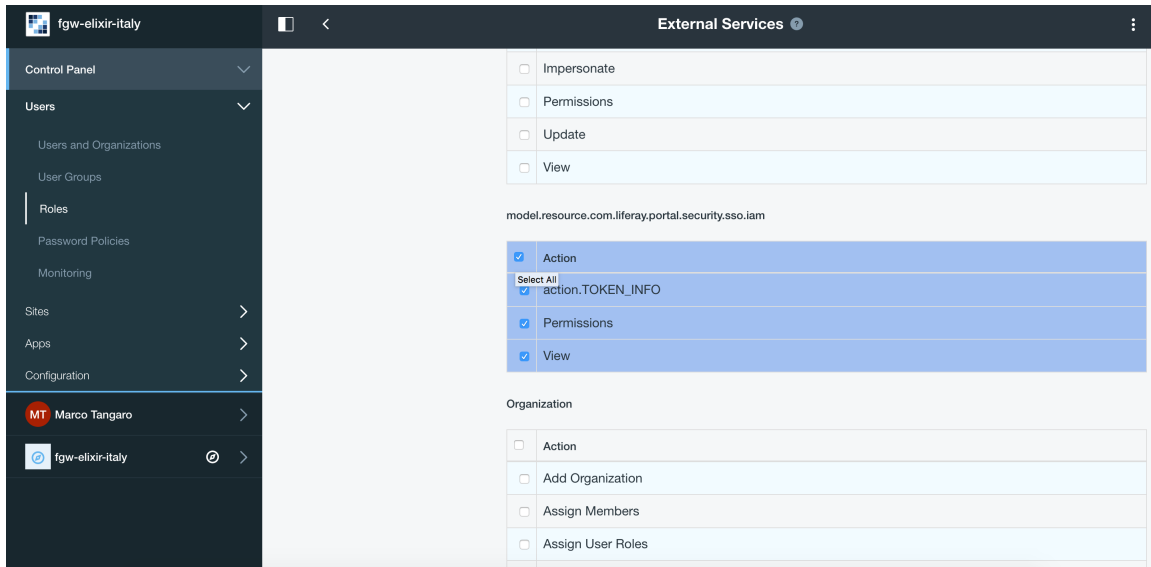
Create a new user (not a IAM user, just register it using the Sign-in liferay module and enable it).

Then assign the new Role `External Services` to the new user: Users and Organizations -> User Information -> Roles -> Select `External Services` and save.

To configure PTV you have to modify `FutureGateway/fgAPIServer/fgapiserver.conf` [1] by the following options:

```
fgapisrv_ptvendpoint= https://hostname/api/jsonws/iam.token/get-token-info
fgapisrv_ptvuser      = [...]
fgapisrv_ptvpass      = [...]
```

Moreover you have to configure `FutureGateway/apache-tomcat-8.0.36/webapps/APIServerDaemon/WEB-INF/classes/it/inf/ct/ToscaIDC.properties` [2] with:



```
fgapisrv_frontend    = https://hostname/apis/v1.0
fgapisrv_ptvtokensrv= https://hostname/api/jsonws/iam.token/get-token
fgapisrv_ptvendpoint= https://hostname/api/jsonws/iam.token
fgapisrv_ptvuser     = [...]
fgapisrv_ptvpass     = [...]
```

ptvuser and ptpass corresponds to user email and password of a FGW user with the right permissions for token validations.

Warning: After changing [1] restart of Apache # `service apache2 restart`, and after [2] restart of Tomcat # `service futuregateway restart`.

To validate if your PTV service is working, you can do the following:

1. Visit <https://jwt.io> and copy-paste your IAM token. Token is stored in Your User Name -> Account Settings -> Miscellaneous -> Iamaccess token

In the decoded payload, you will find your subject:

```
321f0ea3-4aab-46f7-accf-f645cd9d3629
```

2. Use the PTV web service directly:

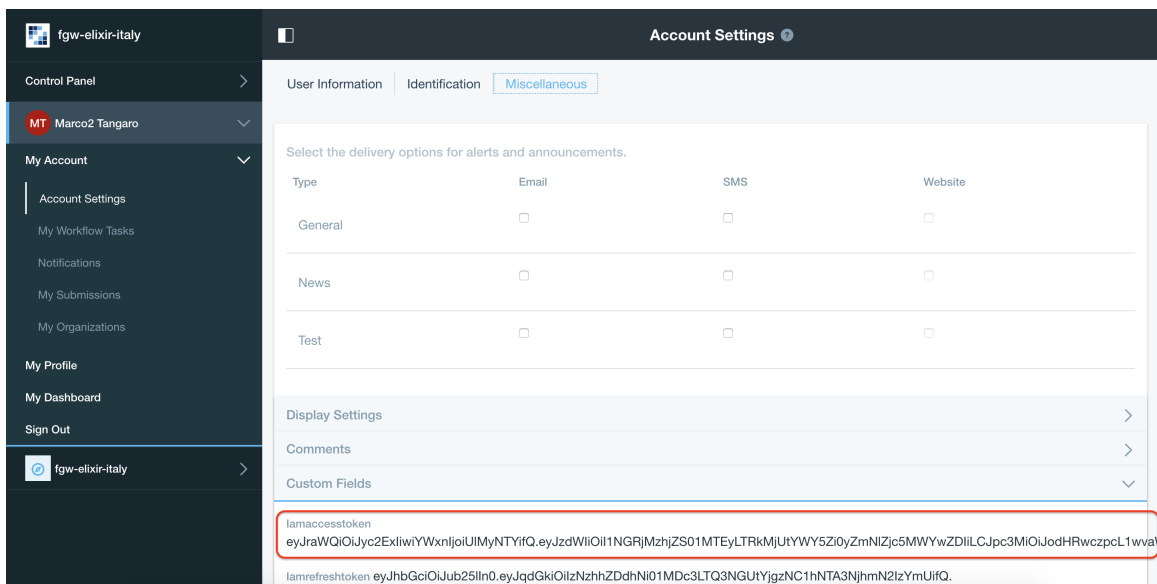
```
$ export PTV_USER= ***
$ export PTV_PASS= ***
$ export SUBJECT=321f0ea3-4aab-46f7-accf-f645cd9d3629

$ curl -u "$PTV_USER:$PTV_PASS" \
  -d "subject=$SUBJECT" \
  https://hostname/api/jsonws/iam.token/get-token
```

To test if the FGW API server is authenticating you correctly, you can do the following:

```
$ curl https://hostname/apis/v1.0/applications
```

This should show '401 Unauthorized', so do the following:

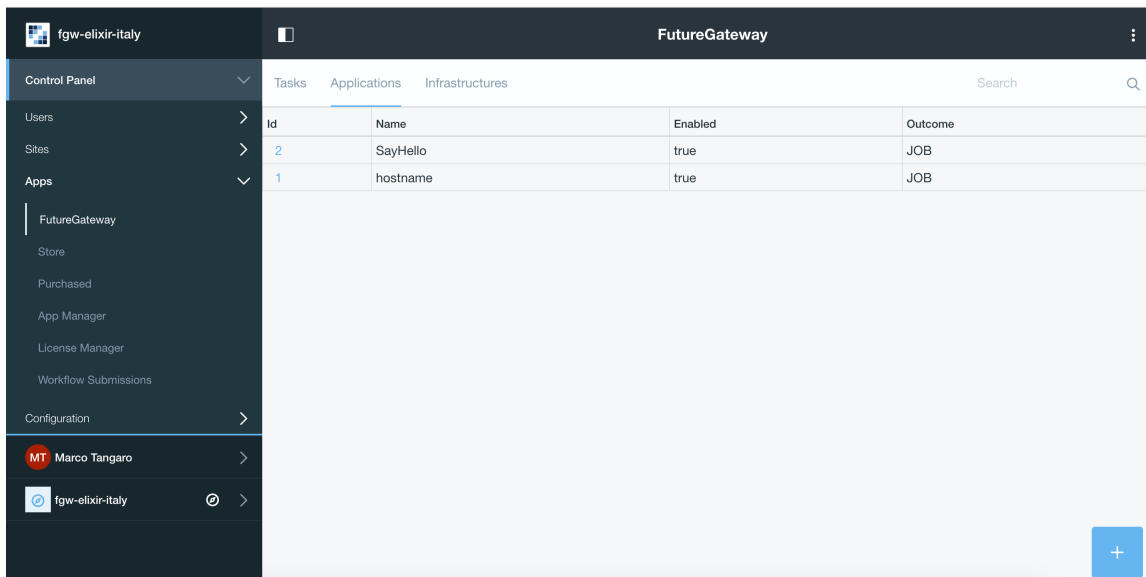


```
export TOKEN = IAM_token_from_FGW_portal

$ curl -H "Authorization: Bearer $TOKEN" https://hostname.cloud.ba.infn.it/apis/v1.0/
↪applications
```

If FG API server is configured correctly, you will get JSON description of your FG applications.

Finally, going in the FutureGateway admin portlet you should see:



30.5 Portlet configuration

30.5.1 Create build environment

To correctly build FutureGateway portlets we recommends to use Ubuntu 16.04 Java 8 and gradle are needed:

```
# apt-get install gradle
```

Install Blade cli: https://dev.liferay.com/develop/tutorials/-/knowledge_base/7-0/installing-blade-cli

The linux version of the liferay portal is available here: <https://sourceforge.net/projects/lportal/files/Liferay%20Workspace/1.5.0.1/LiferayWorkspace-1.5.0.1-linux-x64-installer.run>

```
$ chmod +x LiferayWorkspace-1.5.0.1-linux-x64-installer.run
$ ./LiferayWorkspace-1.5.0.1-linux-x64-installer.run
```

Answer [2] Don't initialize Liferay Workspace directory and continue the installation.

30.5.2 Build portlets

Next you should use some code lines like below:

```
blade init liferay-workspace
cd ./liferay-workspace
git clone https://github.com/indigo-dc/LiferayPlugIns modules/
cd ./modules
git checkout remotes/origin/nonofficial # to build nonofficial portlets
blade gw clean jar
```

Newly created portlets are in ./modules/LIB_NAME/build/libs.

Next you need copy created jars to ~/FutureGateway/deploy and portlets are available on the your website.

30.6 Update to Java 8 - Appendix A

```
sudo apt-get purge openjdk*
sudo add-apt-repository ppa:webupd8team/java
sudo apt-get update
sudo apt-get install oracle-java8-installer
```

log-out and log-in to refresh environment variables.

30.7 Configure Apache for http - Appendix B

Enable http_proxy on apache2

```
a2enmod proxy_http
```

In /etc/apache2/sites-available/ create your `futuregateway.conf` file, setting

```
ServerName <your_serve_name>
...
```

then enable FGW:

```
a2ensite futuregateway.conf
```

and reload apache:

```
# service apache2 reload
```

30.8 Import Signed CA - Appendix C

To import the SSL certificate you have to

1. Install the ca-certificates package:

```
# yum install ca-certificates
```

2. Add the certificate as a new file to

```
# cp path/to/goagent/local/CA.crt /usr/local/share/ca-certificates/cert.crt
```

3. Use command:

```
# update-ca-certificates
```

30.9 Import Signed CA in Java keystore - Appendix D

If IAM is under https but Tomcat log (`$CATILINA_HOME/webapps/APIServerDaemon/WEB-INF/logs/APIServerDaemon.log`) is showing:

```
...
23:03:55,675 ERROR [http-nio-8080-exec-13][IAMEndpoints:69] IAM Configuration URL
↪ 'https://indigoiam.cloud.ba.infn.it/.well-known/openid-configuration' is not_
↪ reachable
...
```

you have to import your https certificate in java keystore [*]:

```
# apt-get install ca-certificates-java

# keytool -import -file /path/to/crt/file/file.crt -storepass changeit -keystore
↪ $JAVA_HOME/lib/security/cacerts -alias mycert
```

To list java certificates:

```
$ keytool -list -storepass changeit -keystore $JAVA_HOME/lib/security/cacerts
```

You can install it in `/etc/ssl/certs` to make curl work, too.

[*] <http://www.thinkplexx.com/learn/howto/security/tools/understanding-java-keytool-working-with-crt-files-fixing-certificate-problem>

30.10 Create https certificate - Appendix E

You can install the certbot tool on your machine (1st approach) or you can use the docker image certbot/certbot (2nd approach).

```
Install certbot tool (https://certbot.eff.org/#ubuntuxenial-other)
$ sudo apt-get install software-properties-common
$ sudo add-apt-repository ppa:certbot/certbot
$ sudo apt-get update
$ sudo apt-get install certbot
$ sudo certbot certonly --standalone -d $HOSTNAME
```

The certificates should be in:

```
$ sudo ls /etc/letsencrypt/live/$HOSTNAME
cert.pem chain.pem fullchain.pem privkey.pem README
```

or

```
$ sudo docker run -it --rm -p 80:80 -p 443:443 -v /etc/letsencrypt:/etc/letsencrypt/ _
↳ certbot/certbot certonly --standalone -d $HOSTNAME
The certificates should be in:
$ sudo ls /etc/letsencrypt/live/$HOSTNAME
cert.pem chain.pem fullchain.pem privkey.pem README
```

30.11 Fix ghost deployment issue

Open mysql FGW database:

```
mysql -h localhost -P 3306 -u fgapiserver -pfgapiserver_password fgapiserver

mysql> select f1.id, f2.id, f1.name from fg_user f1 join fg_user f2 on f1.name = f2.
↳ name where f1.id <> f2.id;
+----+-----+-----+
| id | id | name |
+----+-----+-----+
| 6 | 5 | c0b907df-43a3-4c8a-952a-2b5ca56ec43e |
| 5 | 6 | c0b907df-43a3-4c8a-952a-2b5ca56ec43e |
| 8 | 7 | 9c3c7f53-7279-4008-82be-60600418c884 |
| 7 | 8 | 9c3c7f53-7279-4008-82be-60600418c884 |
+----+-----+-----+
4 rows in set (0.00 sec)
```

Suppose you want to delete user `id = 6`, which is showing you double entries (note that in this particular case we deleted both `id=6` and `id=8` which were our double entries, keeping `id=5` and `id=7`):

```
mysql> delete from fg_user_group where user_id = 6;
Query OK, 1 row affected (0.04 sec)

mysql> delete from fg_user where id=6;
Query OK, 1 row affected (0.01 sec)
```

Then you can add unique constraint to prevent double entries:

```
mysql> alter table fg_user add unique(name);
Query OK, 0 rows affected (0.50 sec)
Records: 0 Duplicates: 0 Warnings: 0
```

30.12 Logs

You can easily access to logs with symlinks:

```
ln -s /home/futuregateway/FutureGateway/apache-tomcat-8.0.36/webapps/APIServerDaemon/
↪WEB-INF/logs/APIServerDaemon.log logs-apiserverdaemon.log
ln -s /home/futuregateway/FutureGateway/fgAPIServer/fgapiserver.log logs-fgapiserver.
↪log
ln -s /home/futuregateway/FutureGateway/apache-tomcat-8.0.36/logs/catalina.out logs-
↪tomcat-catalina.out
```

30.13 References

GitBook: <https://www.gitbook.com/book/indigo-dc/futuregateway/details>

CHAPTER 31

ONEDATA

<https://onedata.org/#/home>

Documentation: <https://onedata.org/#/home/documentation/index.html>

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.

Official GitBook documentation: <https://www.gitbook.com/book/indigo-dc/clues-indigo/details>

32.1 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
* Show the status of the platform
Usage: status
```

(continues on next page)

(continued from previous page)

```

* 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

```

32.2 Check worker nodes deployment

Worker node deployment log are available to: `/var/log/clues2/clues2.log`

32.3 Troubleshooting

32.3.1 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:_
↪eyJraWQiOiJyc2ExIiwiaWwiYXNjaWoiUlMyNTYifQ.
↪eyJzdWIiOiI3REU4Qjg4MC1DNEQwLTQ2RkEtQjQxMS0wQTlCREI3OUYzOTYiLCJpc3MiOiJodHRwczpcL1lwvaW
↪FtLXRlc3QuaW
↪QqjYzVs0h5kuqoBZQf5PPcYrsRJksTFyZ05Zpx8xPcfjruWHwwOnw9knQq8Ex3lwAXgi5qxdmqBDi4EIZA0aofSpir1M7K6fCB
↪M_btm4nTbUvTSaUAfjki41DnPoEjLqXTTy8XLPUrCSmHVeqvSHHFipeSkP9OxKltlUadPc"}
```

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`.

CHAPTER 33

GitHub repository

<https://github.com/Laniakea-elixir-it>

CHAPTER 34

DockerHub repository

<https://hub.docker.com/r/laniakeacloud>

CHAPTER 35

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](#).

CHAPTER 36

Cite

CHAPTER 37

Licence

GNU General Public License v3.0+ (<https://www.gnu.org/licenses/gpl-3.0.txt>)

CHAPTER 38

Galaxy tutorials

Galaxy training network: <https://galaxyproject.org/teach/gtn/>

Galaxy For Developers: <https://crs4.github.io/Galaxy4Developers/>

CHAPTER 39

Indices and tables

- `genindex`
- `modindex`
- `search`