NAR Genomics and Bioinformatics, 2024, **6**, lqae140
https://doi.org/10.1093/nargab/lqae140
Advance access publication date: 10 October 2024
Pipelines and Workflows for Biological Data Analysis

Dynamic configuration and data security for bioinformatics cloud services with the Laniakea Dashboard

Marco Antonio Tangaro[1,2,*,†], Marica Antonacci[2,†], Giacinto Donvito[2], Nadina Foggetti[1,2], Pietro Mandreoli[3], Daniele Colombo[3], Graziano Pesole[1,4,*] and Federico Zambelli[1,3,*]

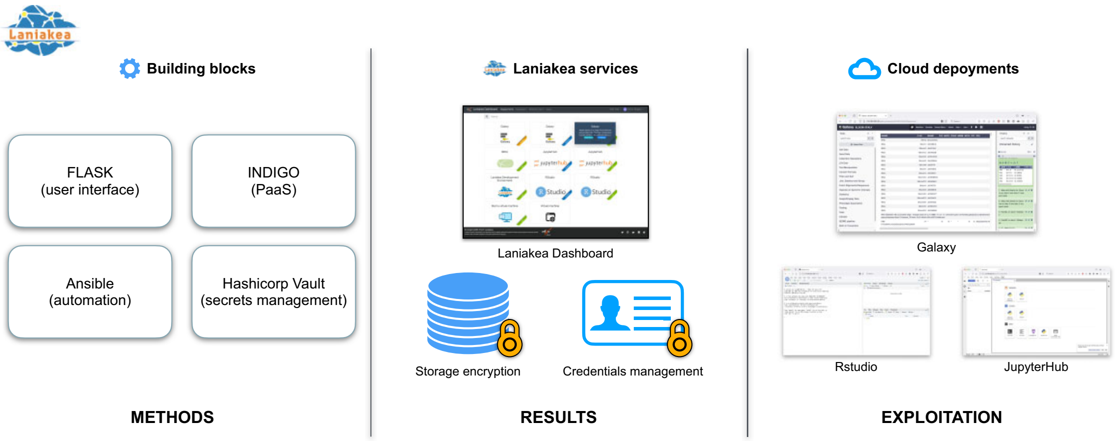
[1] Institute of Biomembranes, Bioenergetics and Molecular Biotechnologies, National Research Council (CNR), Via Giovanni Amendola 122/O, 70126 Bari, Italy
[2] National Institute for Nuclear Physics (INFN), Section of Bari, Via Orabona 4, 70126 Bari, Italy
[3] Department of Biosciences, University of Milan, Via Celoria 26, 20133 Milano, Italy
[4] Department of Biosciences, Biotechnologies and Environment, University of Bari Aldo Moro, Via Orabona 4, 70126 Bari, Italy

[*] To whom correspondence should be addressed. Tel: +39 02 50314923; Email: ma.tangaro@ibiom.cnr.it
Correspondence may also be addressed to Graziano Pesole. Email: graziano.pesole@uniba.it
Correspondence may also be addressed to Federico Zambelli. Email: federico.zambelli@unimi.it
[†] The first two authors should be regarded as Joint First Authors.
Present addresses:
Pietro Mandreoli, Human Technopole, Milan 20157, Italy.
Daniele Colombo, Human Technopole, Milan 20157, Italy.

Abstract

Technological advances in high-throughput technologies improve our ability to explore the molecular mechanisms of life. Computational infrastructures for scientific applications fulfil a critical role in harnessing this potential. However, there is an ongoing need to improve accessibility and implement robust data security technologies to allow the processing of sensitive data, particularly human genetic data. Scientific clouds have emerged as a promising solution to meet these needs. We present three components of the Laniakea software stack, initially developed to support the provision of private on-demand Galaxy instances. These components can be adopted by providers of scientific cloud services built on the INDIGO PaaS layer. The *Dashboard* translates configuration template files into user-friendly web interfaces, enabling the easy configuration and launch of on-demand applications. The *secret management* and the *encryption* components, integrated within the Dashboard, support the secure handling of passphrases and credentials and the deployment of block-level encrypted storage volumes for managing sensitive data in the cloud environment. By adopting these software components, scientific cloud providers can develop convenient, secure and efficient on-demand services for their users.

Graphical abstract



Introduction

High-throughput technologies have revolutionised fields such as genomics, proteomics, and metabolomics, enabling researchers to produce data at an unprecedented scale. However, the challenge lies in effectively managing, analysing, sharing, accessing, and integrating extensive and heterogeneous data

Received: June 29, 2024. **Revised:** August 30, 2024. **Editorial Decision:** September 19, 2024. **Accepted:** September 26, 2024

generated by multiple sources. Addressing these challenges is essential to harness their full value for the scientific community and society (1).

Managing such volumes of data poses problems that can be difficult for research groups or even institutions to address, impelling the establishment of robust local and national computational research infrastructures (2,3). Furthermore, the FAIRification process (4), which aims to make data available to all possible stakeholders across different domains, requires even greater coordinated efforts and broader partnerships (2,5).

For researchers, inadequate access to dedicated computational infrastructures can depend on several factors, such as the lack of local or remote computing resources. However, at least in developed countries, access to computational resources is more frequently hampered by usability barriers preventing researchers from taking advantage of existing infrastructural services (6,7). For example, infrastructures may not meet specific requirements, e.g. the technical and procedural measures needed to ensure the compliant handling of human genetic data under privacy laws and ELSI policies (8). Additionally, their use may be hindered by the need for specialised IT expertise, which can be uncommon among life sciences researchers, forcing them to outsource the analysis of their research data (9). Those and other infrastructural bottlenecks can lead to data under-exploitation, resulting in inefficiencies and increased costs for the community (10).

Several recent initiatives have aimed at improving access to scientific computing infrastructures, e.g. the development of HPC (11) and Cloud (12) services dedicated to various communities has bridged users closer to these technologies, allowing them to exploit extensive computational and storage resources by lowering the complexity barrier to their use. The *UseGalaxy* (13) public servers provide a notable case of those resource-access-democratisation efforts. The success of this approach is also made evident by the adoption of Galaxy beyond its initial bioinformatics community to include fields such as climate, astrophysics, imaging and others (13).

Alongside these well-established multiuser environments, on-demand scientific services are also emerging. These services provide an alternative, more independent access model to computational resources, allowing users to autonomously configure, launch, use, and manage personal instances of data analysis applications (3). This *Platform as a Service* (PaaS) model enables service providers to more easily meet advanced requirements, such as customisation, reserved or privileged use of IT resources, and enhanced data security, by insulating each user or group of users (e.g. research teams) at the data and application levels.

For example, we released Laniakea (14), a software stack for cloud infrastructures that enables the on-demand provision of Galaxy environments. Services based on this platform can be used to deploy private or public production-grade Galaxy instances for data analysis, training or development, as described in (15).

Moreover, Laniakea streamlines the creation and management of encrypted storage volumes, improving security for those services dedicated to processing sensitive data, such as human genomic sequences, thus helping to balance the principles of Open Science with data privacy obligations, such as those mandated by GDPR (https://zenodo.org/records/6334878).

However, suitable access gateways and reliable IT infrastructures are essential to users' adoption of these PaaS solutions. Therefore, we describe the *Laniakea Dashboard*, along with the *secrets management* and *encryption* modules, as convenient building blocks for developing on-demand bioinformatics services based on the *INDIGO PaaS* (16).

Material and methods

The Laniakea software stack

The Laniakea software stack consists of three basic components: the *INDIGO Identity and Access Management (INDIGO IAM)* service, the *PaaS Orchestrator* and its *Dashboard*, each of which controls a specific step in the deployment of the on-demand service.

INDIGO IAM provides the authentication and authorisation infrastructure (AAI) for the Dashboard, PaaS services and IaaS (Infrastructure as a Service, i.e. the virtualised computing, storage, and networking) resources (16).

The PaaS Orchestrator (16) (Orchestrator from now on) deploys the virtual infrastructure required to support the application requested by the user. It utilises available cloud resources to install and configure the selected application(s) on the deployed virtual infrastructure. Upon successful deployment, the Orchestrator provides the user with the endpoint, such as a URL or IP address, where the application is accessible.

The Orchestrator supports deploying any application described by a suitable TOSCA template (http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html), i.e. YAML files, through its *RestFul API* or the CLI utility *orchent*. The templates are organised into three main sections:

- **Input parameters**: specifies the total requested virtual resources, such as the number of vCPUs, RAM and storage, and any configuration options for the application to be deployed.
- **Node template**: details how to allocate the virtual resources among the required compute nodes and specifies which software to install and configure using dedicated Ansible roles (https://ansible.com).
- **Output:** Defines the values to be returned, such as the application IP.

Finally, the Orchestrator Dashboard provides a graphical user interface to the Orchestrator, removing the burden of managing TOSCA templates and command line requests from end-users.

The Laniakea Dashboard

The *Laniakea Dashboard* (the Dashboard from now on) extends the functionalities of the Orchestrator Dashboard (https://zenodo.org/records/7883082) to include the secrets management and storage encryption modules and introduce further features.

The Dashboard converts the configuration options of every deployable application, formally described by a TOSCA template, into a user-friendly web interface presenting only the parameters relevant to that application. Each TOSCA template is associated with a YAML parameters file, which maps each input key to an HTML form input element, such as dropdown menus, text fields, and switch toggles, specifying the available options. Additionally, any TOSCA input the user cannot configure (e.g. infrastructure-specific endpoints, paths to storage volumes) is automatically populated according to the directives in the Dashboard configuration files. This automation enables the seamless execution of advanced tasks, such as managing SSH keys and injecting them into Virtual Machines (VMs) or handling encrypted storage volumes.

The Dashboard is built on the Flask micro-framework (https://flask.palletsprojects.com) and the Jinja template engine (https://jinja.palletsprojects.com) to generate the routing system and the web front-end. During start-up, the Dashboard creates an application tile for each TOSCA template, inspects the template's input section, collects additional input information from the YAML parameter file, and passes this data as variables to the Jinja templates. Once rendered, these templates generate the application-specific HTML submission forms.

Secrets management system and credential support

A secret is any configuration-related information users need to control access to, such as encryption passphrases and application credentials. To manage these secrets securely, Laniakea employs three services: Hashicorp Vault (https://www.vaultproject.io) for secure storage and access, the Orchestrator for enabling read/write operations during application installation and configuration, and the Dashboard, which provides the necessary information for the Orchestrator to access Vault.

HashiCorp Vault is configured to store passphrases in an encrypted form, using specific paths determined by the user's IAM identity and the current deployment. INDIGO IAM assigns a Universally Unique Identifier (UUID) to each user upon registration, while the Orchestrator does the same when deploying an application. The combination of these two UUIDs creates a unique path, ensuring that secrets from different applications do not overwrite each other. Only the owner of the path can read or write to it.

Furthermore, the system's overall security is significantly enhanced by using temporary single-use tokens valid only for a limited timeframe. This approach prevents sharing secrets as plain text among the microservices of the PaaS layer. Any attempt to reuse a token would result in deployment failures.

Two new TOSCA datatypes have been introduced to integrate HashiCorp Vault tokens: *VaultToken* and *VaultSecret*.

- **VaultToken:** it describes the token, i.e. the Vault endpoint, mount point, associated policy (i.e. read, write, delete, see Table 1), and its value.
- **VaultSecret:** it details the secret, such as its path on Vault, the key-value pair, and the action to perform (e.g. read or write).

Originally developed to manage volume encryption passphrases, the secret management system has been generalised to support various types of user secrets, such as application-specific credentials. Each TOSCA template input can be declared as a secret in the YAML parameters file and stored in Vault. Advanced users can also define additional secrets through the Dashboard interface (Supplementary Figure S1). Ansible roles can use those user-defined secrets to configure applications during deployment (see Supplementary S1).

Table 1. Laniakea Vault tokens policies and how they are utilised by the platform

Policy	Used by	Description
Write	Ansible roles and pyLUKS	Write the volume encryption passphrase.
	Dashboard	Write the user application credentials.
Read	Dashboard	Users can read owned encryption passphrases through the Dashboard or consume them to perform mount action on the encrypted volume.
	Ansible roles	Read user credentials to configure applications.
Delete	Dashboard	The encryption passphrase and user custom credentials can be deleted once the associated VM is deleted.

Storage volume encryption system

The Laniakea encryption module exploits the Linux Unified Key Setup (LUKS) (17) specification for encrypting storage volumes at the block level, a well-established standard in Linux environments, supported by the Linux kernel through the *dm-crypt* module.

LUKS implements a robust approach against brute force attacks by adopting the Argon2id hash function (18). It encrypts the storage volume with a master key, which is itself encrypted using the user passphrase and an additional input known as cryptographic salt. Designed to be RAM-intensive, Argon2id provides resistance against massively parallel brute force attacks, such as those facilitated by modern GPUs. The anti-forensic splitter further enhances security by slitting the master key before storing it, preventing attacks aiming to recover deleted data. Metadata, including setup information, is stored in a header partition at the block device's beginning, allowing multiple passphrases to be issued, changed and revoked.

Once encrypted, the volume is initialised using the *ext4* file system and attached to the target VM as persistent storage.

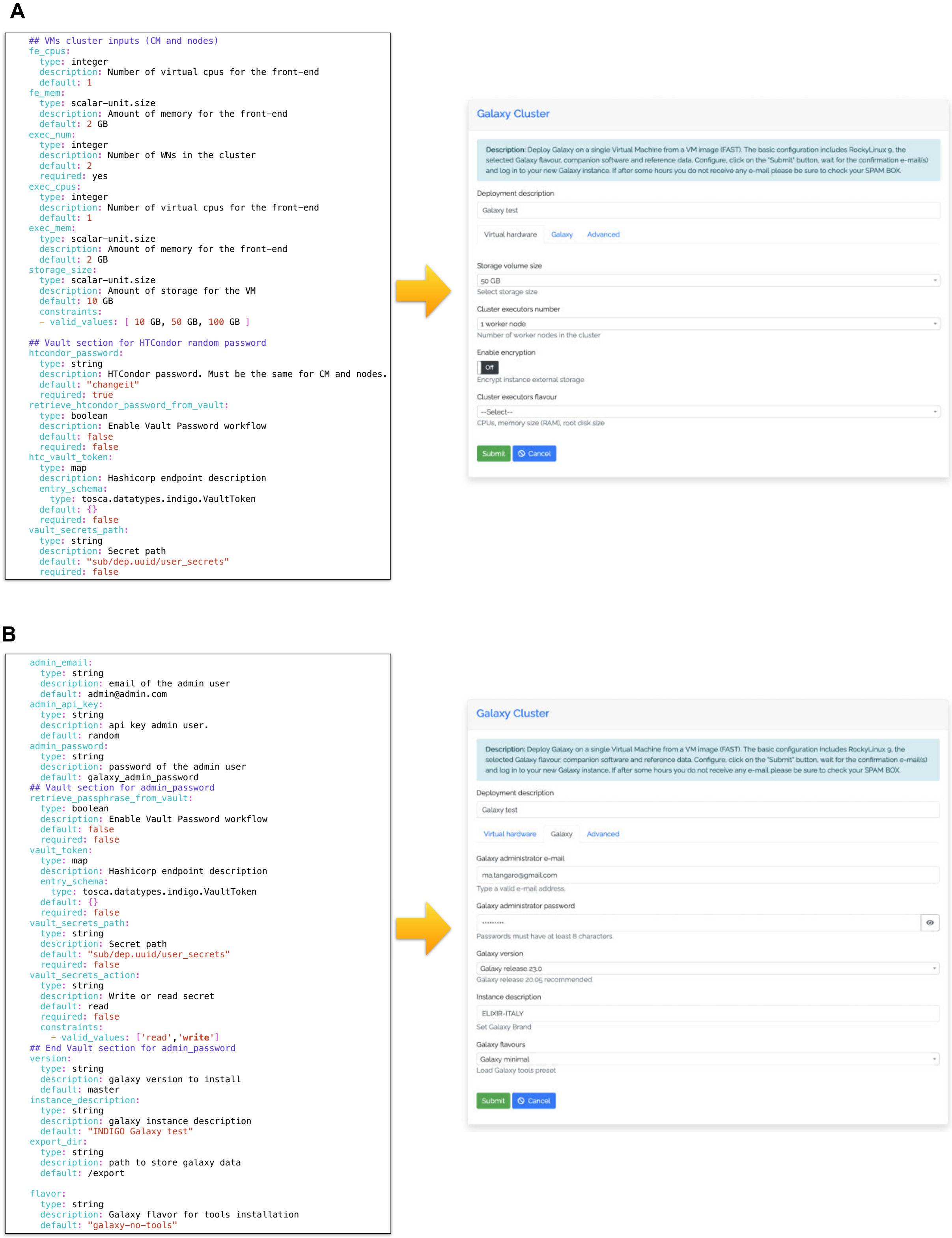
The encryption process and management of the storage volumes are controlled by the *pyLUKS* package (https://github.com/Laniakea-elixir-it/pyluks), consisting of three main applications:

- **fastluks:** Encrypts and initialises the storage devices.
- **luksctl:** Manages encrypted devices, performing operations such as mount/unmount and checking the volume status.
- **luksctl API:** an Application Programming Interface (API) that supports operations on encrypted storage from the Dashboard.

PyLUKS integrates Hashicorp Vault support through the *hvac* (https://hvac.readthedocs.io) package, enabling the *fastluks* module to store passphrases in Vault and the *luksctl* API to read them for mounting the encrypted storage volume.

Laniakea Utils API

The Laniakea Utils API (https://github.com/Laniakea-elixir-it/laniakea-utils) is a package that mediates the interaction between the Dashboard and deployed applications. It enables remote execution of predefined operations, such as restarting an application or handling failures, which would otherwise require command-line execution via SSH. The API accepts



```
## VMs cluster inputs (CM and nodes)
fe_cpus:
  type: integer
  description: Number of virtual cpus for the front-end
  default: 1
fe_mem:
  type: scalar-unit.size
  description: Amount of memory for the front-end
  default: 2 GB
exec_num:
  type: integer
  description: Number of WNs in the cluster
  default: 2
  required: yes
exec_cpus:
  type: integer
  description: Number of virtual cpus for the front-end
  default: 1
exec_mem:
  type: scalar-unit.size
  description: Amount of memory for the front-end
  default: 2 GB
storage_size:
  type: scalar-unit.size
  description: Amount of storage for the VM
  default: 10 GB
  constraints:
  - valid_values: [ 10 GB, 50 GB, 100 GB ]

## Vault section for HTCondor random password
htcondor_password:
  type: string
  description: HTCondor password. Must be the same for CM and nodes.
  default: "changeit"
  required: true
retrieve_htcondor_password_from_vault:
  type: boolean
  description: Enable Vault Password workflow
  default: false
  required: false
htc_vault_token:
  type: map
  description: Hashicorp endpoint description
  entry_schema:
    type: tosca.datatypes.indigo.VaultToken
  default: {}
  required: false
vault_secrets_path:
  type: string
  description: Secret path
  default: "sub/dep.uuid/user_secrets"
  required: false
```

Galaxy Cluster

Description: Deploy Galaxy on a single Virtual Machine from a VM image (FAST). The basic configuration includes RockyLinux 9, the selected Galaxy flavour, companion software and reference data. Configure, click on the "Submit" button, wait for the confirmation e-mail(s) and log in to your new Galaxy instance. If after some hours you do not receive any e-mail please be sure to check your SPAM BOX.

Deployment description: Galaxy test

Virtual hardware | Galaxy | Advanced

Storage volume size: 50 GB
Select storage size

Cluster executors number: 1 worker node
Number of worker nodes in the cluster

Enable encryption: Off
Encrypt instance external storage

Cluster executors flavour: --Select--
CPUs, memory size (RAM), root disk size

Submit | Cancel

```
admin_email:
  type: string
  description: email of the admin user
  default: admin@admin.com
admin_api_key:
  type: string
  description: api key admin user.
  default: random
admin_password:
  type: string
  description: password of the admin user
  default: galaxy_admin_password
## Vault section for admin_password
retrieve_passphrase_from_vault:
  type: boolean
  description: Enable Vault Password workflow
  default: false
  required: false
vault_token:
  type: map
  description: Hashicorp endpoint description
  entry_schema:
    type: tosca.datatypes.indigo.VaultToken
  default: {}
  required: false
vault_secrets_path:
  type: string
  description: Secret path
  default: "sub/dep.uuid/user_secrets"
  required: false
vault_secrets_action:
  type: string
  description: Write or read secret
  default: read
  required: false
  constraints:
    - valid_values: ['read','write']
## End Vault section for admin_password
version:
  type: string
  description: galaxy version to install
  default: master
instance_description:
  type: string
  description: galaxy instance description
  default: "INDIGO Galaxy test"
export_dir:
  type: string
  description: path to store galaxy data
  default: /export

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

Galaxy Cluster

Description: Deploy Galaxy on a single Virtual Machine from a VM image (FAST). The basic configuration includes RockyLinux 9, the selected Galaxy flavour, companion software and reference data. Configure, click on the "Submit" button, wait for the confirmation e-mail(s) and log in to your new Galaxy instance. If after some hours you do not receive any e-mail please be sure to check your SPAM BOX.

Deployment description: Galaxy test

Virtual hardware | Galaxy | Advanced

Galaxy administrator e-mail: ma.tangaro@gmail.com
Type a valid e-mail address.

Galaxy administrator password: ••••••••••
Passwords must have at least 8 characters.

Galaxy version: Galaxy release 23.0
Galaxy release 20.05 recommended

Instance description: ELIXIR-ITALY
Set Galaxy Brand

Galaxy flavours: Galaxy minimal
Load Galaxy tools preset

Submit | Cancel

Figure 1. Two examples of the TOSCA template input (left) and the corresponding web interface panels as rendered by the Dashboard (right) for configuring a Galaxy instance backed by an HTCondor Cluster. (**A**) Input required for creating a VM cluster, including the number of worker nodes, their configuration, the storage volume size and the option for encryption. (**B**) The customisation options available for the application, such as administrator credentials (username and password), software version and the set of pre-installed bioinformatics tools.

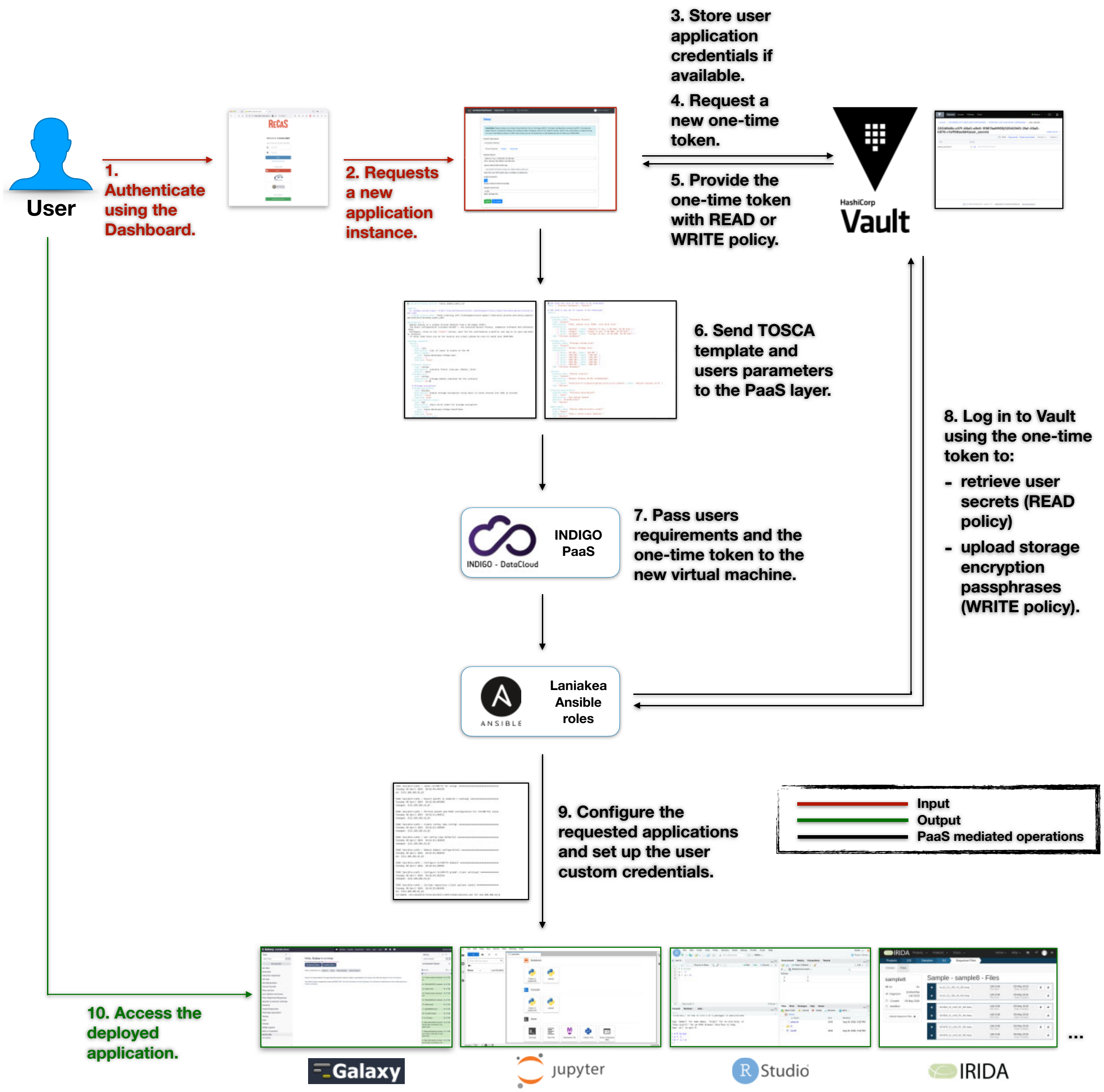

User
1. Authenticate using the Dashboard.
2. Requests a new application instance.
3. Store user application credentials if available.
4. Request a new one-time token.
5. Provide the one-time token with READ or WRITE policy.
HashiCorp Vault
6. Send TOSCA template and users parameters to the PaaS layer.
INDIGO PaaS
INDIGO - DataCloud
7. Pass users requirements and the one-time token to the new virtual machine.
8. Log in to Vault using the one-time token to:
- retrieve user secrets (READ policy)
- upload storage encryption passphrases (WRITE policy).
Laniakea Ansible roles
ANSIBLE
9. Configure the requested applications and set up the user custom credentials.
Input
Output
PaaS mediated operations
10. Access the deployed application.
Galaxy
jupyter
RStudio
IRIDA
...

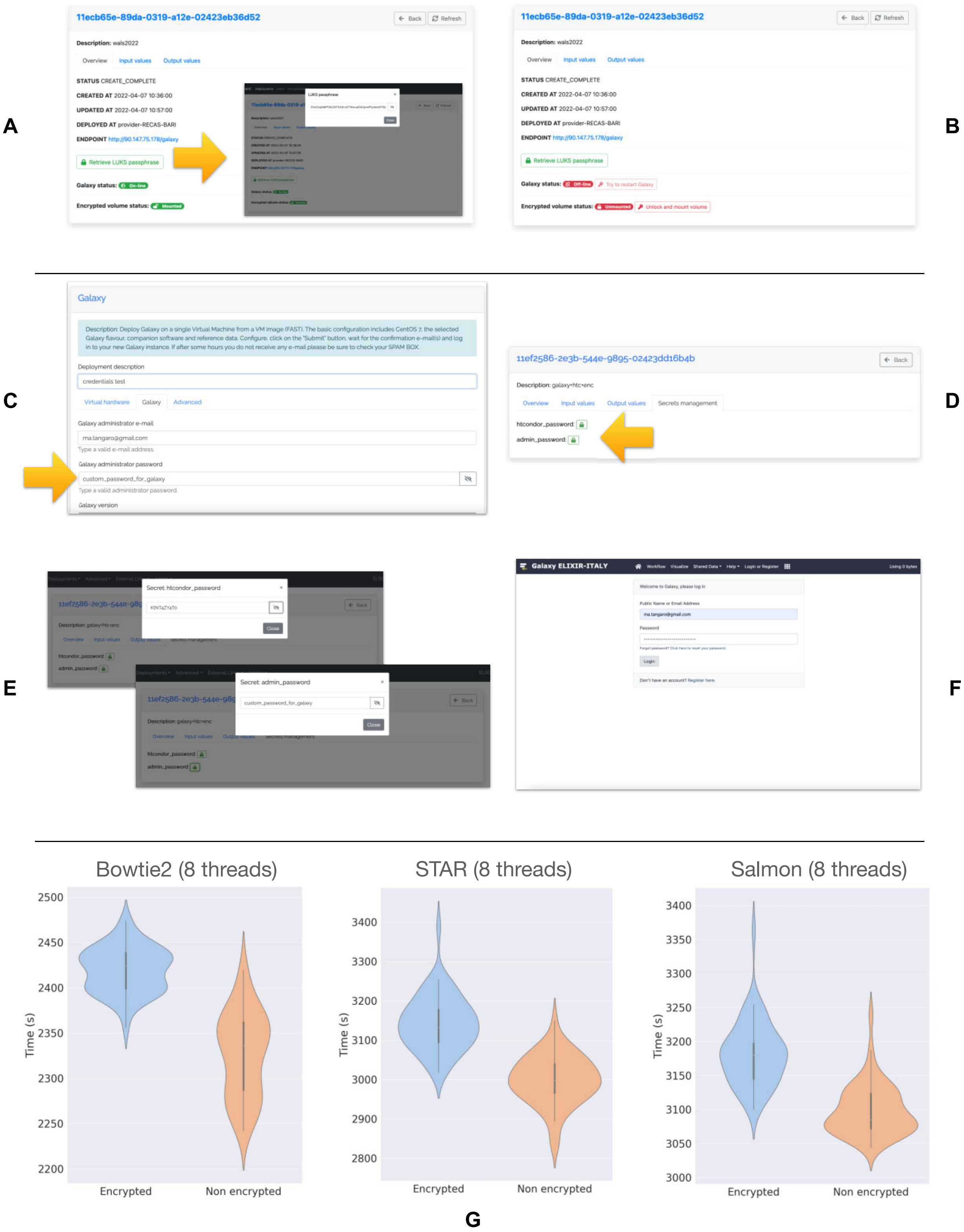
Figure 2. The secrets management workflow when deploying an application using an encrypted storage volume and private user credentials. The user authenticates and logs into the Laniakea Dashboard (1). The user configures the requested application and provides application-specific secrets, i.e. the credentials. (2–7) The Dashboard contacts Hashicorp Vault to store the secrets and retrieve a one-time token, sent through the Orchestrator service and any other user preference, to the VM created to host the application. (8) An Ansible role logs into Vault using the one-time token to upload the encryption passphrase using the write policy and retrieves the credentials using the read policy. (9) Ansible installs and configures the requested application to accept the user-supplied credentials. (10) The user can access the applications using the credentials. All trademarks, logos and brand names are the property of their respective owners.

connections exclusively from the user's Dashboard and utilises the *flaat* Python library (https://github.com/indigo-dc/flaat) to authenticate routes using IAM tokens, ensuring that only authorized actions are performed.

Results

The Laniakea Dashboard is designed to provide scientific cloud service providers with a comprehensive tool to assist users throughout the entire lifecycle of their deployed applications, including creation, management and deletion. Initially developed for deploying on-demand Galaxy instances and utilised by the *Laniakea@ReCaS* service (15), the implementation is flexible enough to support any similar service based on the INDIGO PaaS framework (16).

As an example, we present in Figure 1 two web forms generated by the Dashboard for deploying and configuring a Galaxy instance supported by an HTCondor cluster alongside their corresponding TOSCA template input. The Dashboard renders only the parameters relevant to the user, while other parameters are managed according to the Dashboard



11ecb65e-89da-0319-a12e-02423eb36d52

Description: wals2022

STATUS CREATE_COMPLETE

CREATED AT 2022-04-07 10:36:00

UPDATED AT 2022-04-07 10:57:00

DEPLOYED AT provider-RECAS-BARI

ENDPOINT http://90.147.75.178/galaxy

Galaxy status: On-line

Encrypted volume status: Mounted

LUKS passphrase

11ecb65e-89da-0319-a12e-02423eb36d52

Description: wals2022

STATUS CREATE_COMPLETE

CREATED AT 2022-04-07 10:36:00

UPDATED AT 2022-04-07 10:57:00

DEPLOYED AT provider-RECAS-BARI

ENDPOINT http://90.147.75.178/galaxy

Galaxy status: Off-line

Encrypted volume status: Unmounted

Galaxy

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

Deployment description: credentials test

Galaxy administrator e-mail: ma.tangaro@gmail.com

Type a valid e-mail address

Galaxy administrator password: custom_password_for_galaxy

Type a valid administrator password

Galaxy version

11ef2586-2e3b-544e-9895-02423dd16b4b

Description: galaxy+htc+enc

htcondor_password:

admin_password:

Secret: htcondor_password

KINT4ZY4To

Secret: admin_password

custom_password_for_galaxy

Galaxy ELIXIR-ITALY

Welcome to Galaxy, please log in

Public Name or Email Address: ma.tangaro@gmail.com

Password: ••••••••••••••••••••••••••

Forgot password? Click here to reset your password.

Don't have an account? Register here.

Figure 3. (**A, B**) The Laniakea Dashboard instance management interface allows the user to check the status of Galaxy instances and associated encrypted storage volumes. It is possible to retrieve the encryption passphrase (A) at any moment. If Galaxy or the storage volume are offline, the user can restore them using the interface (B). (**C–F**) The Laniakea Dashboard interface is used to set up and retrieve custom user credentials for a Galaxy instance. The user sets the Galaxy administrator e-mail and password in the configuration tab (C). After the deployment, the password can be retrieved at any time (D, E). Finally, the credentials are used to access Galaxy as an administrator (F). (**G**) Results of the average job runtime test for Bowtie 2, STAR, and Salmon for encrypted (blue) and non-encrypted (orange) volumes. See Supplementary S2 for details on how the test was conducted.

configuration settings. This approach simplifies complex tasks, such as cluster configuration and storage encryption procedures.

The workflow for creating a new on-demand Galaxy instance with encrypted storage is outlined in Figure 2. After authentication, the user selects the desired virtual hardware, enables the encryption procedure by toggling a switch on the interface, configures the application, and submits the request. At this point, the Dashboard retrieves the appropriate token(s) from the Vault service along with any other required information from its configuration files. These, combined with the parameters set by the user through the web interface, complete the application-specific TOSCA template, which is then dispatched to the Orchestrator. Ansible roles subsequently use the data to encrypt the storage volume and install and customise the application at deployment time.

A one-time write token is retrieved from Vault and delivered to the Ansible role responsible for running pyLUKS, which performs the encryption using a strong alphanumeric random passphrase and securely stores it in Vault. The Vault 'write' policy prevents overwriting existing secrets, ensuring that passphrases cannot be accidentally overwritten. If the user has defined additional secrets through the Dashboard, one-time read tokens are fetched from Vault and delivered to the Ansible role, which uses them to configure the desired application.

Furthermore, Dashboard-generated front-ends enable users to perform scripted operations on applications after deployment by interacting directly with the host VM or its storage volume(s). Users can restart the application in case of problems, re-mount the (encrypted) storage volume, reboot the VM or switch it off (Figure 3A, B). The Dashboard communicates with the host virtual machine through the APIs installed on the latter, for example, the *luksctl* API provides web routes to check the status of the storage volume and mount it if necessary. The new Laniakea Utils API facilitates interaction with deployed applications, enabling the execution of predefined operations, such as restarting Galaxy once the storage volume has been correctly unlocked.

Finally, users can always retrieve their secrets, such as encryption passphrases (Figure 3A, B) and user credentials (Figure 3C–F) from Vault using the Dashboard, as they will be removed only when the owner deletes the associated VM.

Laniakea's encryption implementation ensures that the application layer on the VM, whether it is Galaxy, bioinformatics tools, or any other application, can perform transparent input/output operations on the volume, as the data are encrypted and decrypted on the fly at the level of the Linux kernel during read/write operations.

Since introducing the storage encryption layer could impact the host VM's performance and users' daily operations, we decided to measure any footprint it may have. We benchmarked the job runtime using an encrypted volume and compared it to those obtained with an unencrypted one. For our test, we used Galaxy to run two popular reads mapping tools, *Bowtie2* (19) and *STAR* (20), as well as the quasi-mapping quantification function of *Salmon* (21). The rationale of this choice is that those tools involve a substantial number of concurrent multithreaded operations, assessing the effect of the encryption layer on the performance of the virtual storage device. Our results show that the performance impact of employing the encryption layer is limited, ranging approximately from 2.5% to 5% or less of the run times (Figure 3G, Supplementary S2 and Supplementary Table S1).

Conclusions and outlook

Although high throughput biomolecular techniques are increasingly being adopted also outside of research laboratories, e.g. in clinical, food safety, animal health, environmental monitoring, and other settings, the analysis of large quantities of data may remain challenging due to the scarcity of specialised personnel or the inadequacy of the local IT infrastructure: the Laniakea software components we described can help to address those challenges when coupled with adequate cloud resources.

The Dashboard presents the configuration options for on-demand applications through a web front-end, complementing INDIGO IAM and the Orchestrator to enable providers to adapt their services to the specificities of the local cloud platform. In particular, the Dashboard flexibility enables the seamless introduction of new options and applications by simply editing TOSCA templates: the interface dynamically adjusts to reflect these modifications. From the user's perspective, a uniform front-end for all the applications and their configuration options streamlines access and operations, reducing the barriers associated with installing and configuring complex applications.

Our volume performance test indicates that the trade-off between significantly enhanced data security and a slight reduction in throughput is acceptable in most scenarios. This supports the adoption of block-level storage encryption in routine bioinformatics workflows where stringent data security measures are essential. In fact, the storage volume encryption and secret management systems implemented in the Dashboard can promote the adoption of the on-demand application model in scientific domains where robust data protection is crucial, including personalised medicine, biobanking, clinical trials and drug discovery, genomic data analysis and GWAS studies. Typical use cases are provided by the variant detection, annotation and prioritisation methods and the bacterial genotypization tools described in (15,22): private Galaxy instances have been deployed with Laniakea and are used daily by institutions like hospitals and food-safety authorities that operate on sensitive data and require tools and reference data availability, proper compute resources, data isolation, and ease of use (23–25).

The Laniakea components we describe can also support deploying distributed computing environments, allowing remote IT resources to be deployed and insulated from external access. For example, the Pulsar application (https://github.com/galaxyproject/pulsar), which makes possible to route Galaxy jobs on a remote virtual cluster, can be used in conjunction with a virtual cluster deployed using Laniakea, obtaining data insulation from the infrastructure through storage encryption.

A further refinement of the data security layer would be to support with Laniakea the automatic encapsulation of the cluster within a virtual private network (VPN). Finally, also user applications could be deployed within a VPN, thus minimising the risks posed by attacks directed against the hosting VM.

Data availability

Data for the storage encryption performance test can be found at https://doi.org/10.5281/zenodo.11684814.

Supplementary data

Supplementary Data are available at NARGAB Online.

Acknowledgements

The authors thank ELIXIR-Italy and ReCaS-Bari for providing computing and bioinformatics facilities.

The authors thank Maria Rosa Mirizzi, Barbara De Marzo and Laura Marra for their administrative support.

Author contributions: Marco Antonio Tangaro: Investigation, Methodology, Software, Writing-original draft. Marica Antonacci: Investigation, Methodology, Software, Resources. Giacinto Donvito: Project administration, Resources. Nadina Foggetti: Project administration. Pietro Mandreoli: Methodology, Software. Daniele Colombo: Methodology, Software. Graziano Pesole: Project administration, Funding acquisition. Zambelli: Investigation, Project administration, Supervision, Writing-review & editing.

Funding

ELIXIR-IIB and ReCaS-Bari internal funding, the European Commission Horizon Europe research and innovation program under grant agreements with ID 101057388 (EuroScienceGateway), ID 101058386 (interTwin) and 101058593 (AI4EOSC); National Recovery and Resilience Plan (PNRR) projects ELIXIRNextGenIT [R0000010]; High-Performance Computing, Big Data e Quantum Computing Research Centre and TeRABIT. Funding for open access charge: National Research Council.

Conflict of interest statement

None declared.

Disclaimer

This article is an extended and revised version of the conference paper https://doi.org/10.1007/978-3-031-25380-5_14 published in the Springer Nature Computer Science book series.

References

1. Cremin,C.J., Dash,S. and Huang,X. (2022) Big data: historic advances and emerging trends in biomedical research. *Curr. Res. Biotechnol.*, **4**, 138–151.
2. Martin,C.S., Repo,S., Arenas Márquez,J., Blomberg,N., Lauer,K.B., Pérez Sitjà,X., Velek,P., Melo,A.M.P., Stansberg,C., De Leo,F., *et al.* (2021) Demonstrating public value to funders and other stakeholders—the journey of ELIXIR, a virtual and distributed research infrastructure for life science data. *Ann. Public Cooper. Econ.*, **92**, 497–510.
3. Navale,V. and Bourne,P.E. (2018) Cloud computing applications for biomedical science: a perspective. *PLoS Comput. Biol.*, **14**, e1006144.
4. Wilkinson,M.D., Dumontier,M., Aalbersberg,I..J., Appleton,G., Axton,M., Baak,A., Blomberg,N., Boiten,J.-W., da Silva Santos,L.B., Bourne,P.E., *et al.* (2016) The FAIR Guiding Principles for scientific data management and stewardship. *Sci. Data*, **3**, 160018.
5. David,R., Rybina,A., Burel,J., Heriche,J., Audergon,P., Boiten,J., Coppens,F., Crockett,S., Exter,K., Fahrner,S., *et al.* (2023) “Be sustainable”: eOSC-Life recommendations for implementation of FAIR principles in life science data handling. *EMBO J.*, **42**, e115008.
6. Dahlö,M., Scofield,D.G., Schaal,W. and Spjuth,O. (2018) Tracking the NGS revolution: managing life science research on shared high-performance computing clusters. *GigaScience*, **7**, giy028.
7. Harrow,J., Hancock,J., Elixir, -EXCELERATE Community and Blomberg,N. (2021) ELIXIR-EXCELERATE: establishing Europe’s data infrastructure for the life science research of the future. *EMBO J.*, **40**, e107409.
8. Starkbaum,J. and Felt,U. (2019) Negotiating the reuse of health-data: research, Big data, and the European General Data Protection Regulation. *Big Data Soc.*, **6**, 2053951719862594.
9. Smith,D.R. (2018) Bringing bioinformatics to the scientific masses. *EMBO Rep.*, **19**, e46262.
10. Prosperi,M., Min,J.S., Bian,J. and Modave,F. (2018) Big data hurdles in precision medicine and precision public health. *BMC Med. Inf. Decis. Making*, **18**, 139.
11. Castrignanò,T., Gioiosa,S., Flati,T., Cestari,M., Picardi,E., Chiara,M., Fratelli,M., Amente,S., Cirilli,M., Tangaro,M.A., *et al.* (2020) ELIXIR-IT HPC@CINECA: high performance computing resources for the bioinformatics community. *BMC Bioinf.*, **21**, 352.
12. Langmead,B. and Nellore,A. (2018) Cloud computing for genomic data analysis and collaboration. *Nat. Rev. Genet.*, **19**, 208–219.
13. The Galaxy Community (2024) The Galaxy platform for accessible, reproducible, and collaborative data analyses: 2024 update. *Nucleic Acids Res.*, **52**, W83–W94.
14. Tangaro,M.A., Donvito,G., Antonacci,M., Chiara,M., Mandreoli,P., Pesole,G. and Zambelli,F. (2020) Laniakea: an open solution to provide Galaxy “on-demand” instances over heterogeneous cloud infrastructures. *GigaScience*, **9**, giaa033.
15. Tangaro,M.A., Mandreoli,P., Chiara,M., Donvito,G., Antonacci,M., Parisi,A., Bianco,A., Romano,A., Bianchi,D.M., Cangelosi,D., *et al.* (2021) Laniakea@ReCaS: exploring the potential of customisable Galaxy on-demand instances as a cloud-based service. *BMC Bioinf.*, **22**, 544.
16. Salomoni,D., Campos,I., Gaido,L., de Lucas,J.M., Solagna,P., Gomes,J., Matyska,L., Fuhrman,P., Hardt,M., Donvito,G., *et al.* (2018) INDIGO-DataCloud: a platform to facilitate seamless access to E-infrastructures. *J. Grid Comput.*, **16**, 381–408.
17. Fruhwirth,C. (2005) New methods in hard disk encryption. Master’s thesis, Vienna University of Technology.
18. Biryukov,A., Dinu,D., Khovratovich,D. and Josefsson,S. (2021) Argon2 Memory-hard function for password hashing and proof-of-work applications internet engineering task force. https://www.rfc-editor.org/info/rfc9106.
19. Langmead,B. and Salzberg,S.L. (2012) Fast gapped-read alignment with Bowtie 2. *Nat. Methods*, **9**, 357–359.
20. Dobin,A., Davis,C.A., Schlesinger,F., Drenkow,J., Zaleski,C., Jha,S., Batut,P., Chaisson,M. and Gingeras,T.R. (2013) STAR: ultrafast universal RNA-seq aligner. *Bioinformatics*, **29**, 15–21.
21. Patro,R., Duggal,G., Love,M.I., Irizarry,R.A. and Kingsford,C. (2017) Salmon: fast and bias-aware quantification of transcript expression using dual-phase inference. *Nat. Methods*, **14**, 417–419.
22. Chiara,M., Mandreoli,P., Tangaro,M.A., D’Erchia,A.M., Sorrentino,S., Forleo,C., Horner,D.S., Zambelli,F. and Pesole,G. (2020) VINYL: variant prIoritizatioN bY survivaL analysis. *Bioinformatics*, **36**, 5590–5599.
23. Romano,A., Carrella,S., Rezza,S., Nia,Y., Hennekinne,J.A., Bianchi,D.M., Martucci,F., Zuccon,F., Gulino,M., Di Mari,C., *et al.* (2023) First report of food poisoning due to staphylococcal enterotoxin type B in Döner Kebab (Italy). *Pathogens*, **12**, 1139.

24. Meletiadis,A., Romano,A., Moroni,B., Di Nicola,M.R., Montemurro,V., Pitti,M., Pezzolato,M., Bozzetta,E., Sciuto,S. and Acutis,P.L. (2024) A case of food-borne salmonellosis in a corn snake (*Pantherophis guttatus*) after a feeder mouse meal. *Animals*, **14**, 1722.

25. Floris,I., Vannuccini,A., Ligotti,C., Musolino,N., Romano,A., Viani,A., Bianchi,D.M., Robetto,S. and Decastelli,L. (2024) Detection and characterization of zoonotic pathogens in game meat hunted in Northwestern Italy. *Animals*, **14**, 562.

Received: June 29, 2024. **Revised:** August 30, 2024. **Editorial Decision:** September 19, 2024. **Accepted:** September 26, 2024