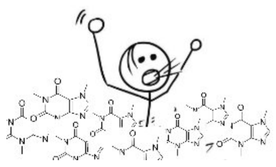
MetaboCloud : A catalog of microservices hosted on a Cloud infrastructure and addressing issues linked to FAIR principles and open science.

Faustine SOUC[1], *Thomas BELLEMBOIS*[2], *Nils PAULHE*[1], *Érina POINT*[1], *Antoine MAHUL*[2], *Nadia GOUÉ*[2] *and Franck GIACOMONI*[1]

[1] *Université Clermont Auvergne, INRAE, UNH, Plateforme d'Exploration du Métabolisme, MetaboHUB Clermont, F-63000 Clermont-Ferrand, France*
[2] *Université Clermont Auvergne, AuBi Platform and Mésocentre Clermont Auvergne, F-63000 Clermont-Ferrand, France*
Corresponding Authors : franck.giacomoni@inrae.fr and nadia.goue@uca.fr

CONTEXT



Metabolomics

Challenges to overcome

- Massive and complex data manipulation
- Molecule annotation

METABOCLOUD

Catalog of bioinformatics tools in the form of microservices hosted on a Cloud infrastructure

Proof of concept

Recipe to share with the community

Meeting **FAIR** (Findable, Accessible, Interoperability, Reusable) principles

Integrate best practices in term of code and deployment

Challenges to overcome

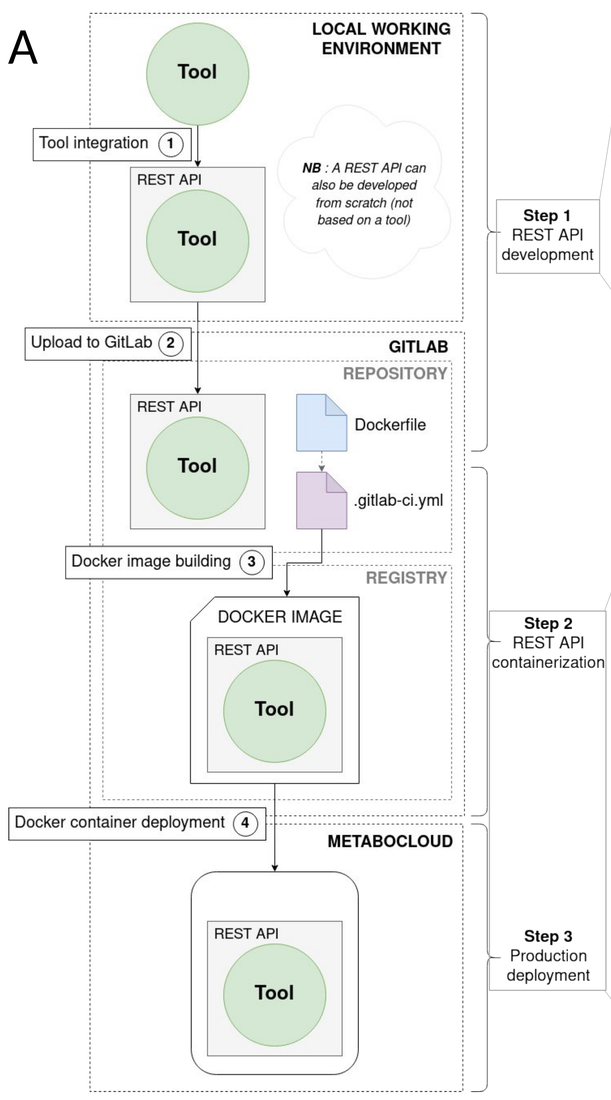
- Scalability
- Performance
- Interfacing with web portal

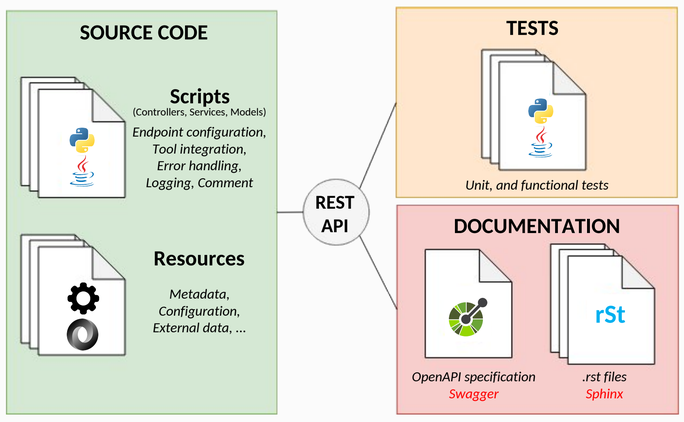
Bioinformatics

MATERIAL AND METHODS

The **recipe** established for MetaboCloud follows several steps (*A*) :

Step 1 : The REST API development (B)
Step 2 : The REST API containerization (C)
Step 3 : The deployment in production (C)





B **Step 1 :** REST API development

The REST API development stage itself comprises several components:

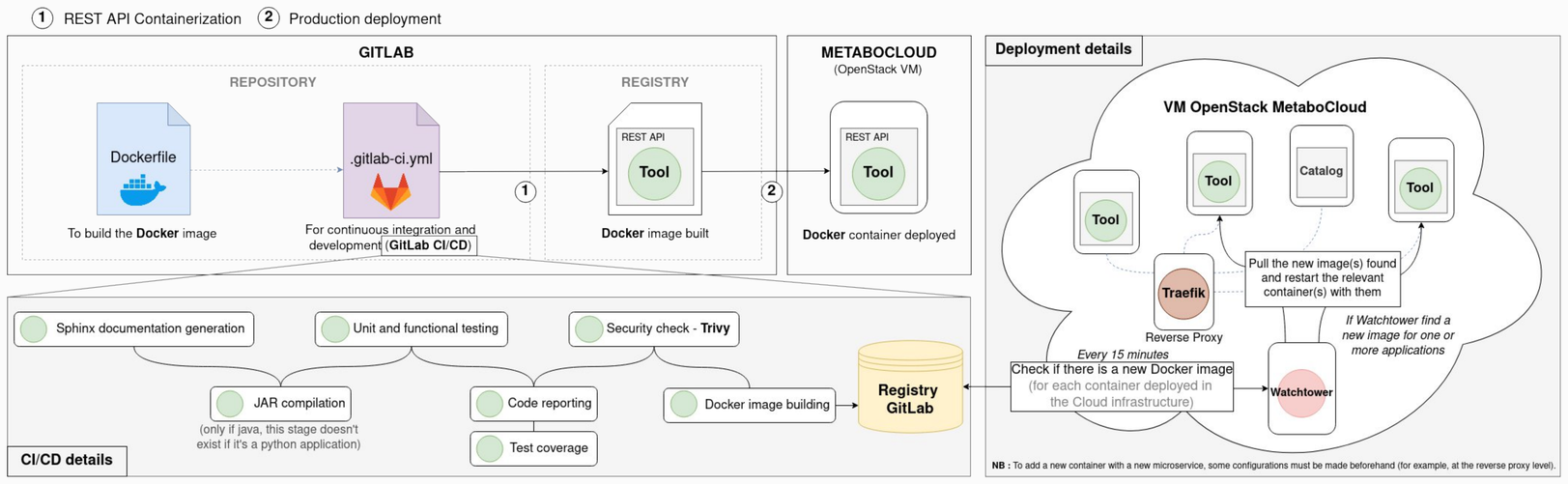
- ➜ **Source code implementation**, i.e. development of the application core (in MetaboCloud project, using Java or Python)
- ➜ **Test implementation**, with the aim of achieving test coverage of over 80% (monitored with SonarQube)
- ➜ **Drafting of documentation**, the first based on OpenAPI specifications (technical guide), the second more general using Sphinx (user guide)

2 Templates available

Java 17
- OpenAPI specification 3.0.3
- Sphinx 7.2.6
- Spring Boot 3.2.5
- Maven 3.6.3
- OpenAPI Generator 7.2.0

Python 3.10
- OpenAPI specification 3.0.3
- Sphinx 7.2.6
- Falcon 3.1.3

C **Step 2 & 3 :** REST API containerization & Production deployment



The REST API containerization stage is carried out using **Docker** technology within **GitLab's CI/CD** (Continuous Integration / Continuous Deployment) pipeline.

The deployment of Docker containers in production on the MetaboCloud **OpenStack** VM is automated thanks to the use of **Watchtower.**

RESULTS

Three microservices are currently available via the web portal MetaboCloud.

- ➜ Goslin[1] ("**G**rammar **O**n **S**uccinct **LI**pid **N**omenclature") microservice developed in python 3.10 from **pygoslin 2.1.0**
- ➜ CDK[2] ("**C**hemistry **D**evelopment **K**it") microservice developed in java 17 from **CDK 2.9**
- ➜ InChI[3] microservice developed in java 17 from the binary program **InChI 1.06**

Code source

Goslin

Translate lipid names into standardized names

Documentation

	GET /about	GET /grammars	GET /validate
Input :	∅	∅	Lipid name
Output :	Metadata	Grammars list	Lipid name standardization

InChI

Produce InChI/InChI Key of chemical compounds

Documentation

	GET /about	POST /generation
Input :	∅	InChI/MOL/SDF
Output :	Metadata	Generate the compound InChI/InChI Key

CDK

Compute chemical properties, depict or convert a molecule

Documentation

	GET /about	POST /properties	POST /depiction/{format}	POST /conversion/{format}
			format = png or svg	*format = inchi, inchikey, mol or sdf*
Input :	∅	InChI/MOL/SDF	InChI/MOL/SDF	InChI/MOL/SDF
Output :	Metadata	Chemical properties	Visual representation of the compound	Convert to another format

Catalog

https://metabocloud.mesocentre.uca.fr

Application building the MetaboCloud web portal, which list all available microservices with their metadata and documentation.

They can be used in a variety of ways. Either integrated into an application, used in batch mode with scripts, or via their associated web component.

Web Components
Ready-to-use client to insert into a HTML page
Documentation

Batch Mode
Single or combined use of microservices via scripts

Application
Construction of an application based on microservices architecture

CONCLUSIONS

The MetaboCloud microservices infrastructure is based on :

- ➜ Bioinformatic tools
- ➜ From scratch API development if tools are not available
- ➜ Quality testing and documentation
- ➜ An advanced CI/CD work environment managing the construction of a docker image
- ➜ Taken all together in an OpenStack cloud technology environment

Findable ➜ Web portal associated with metadata (FAIRChecker evaluation)
Accesible ➜ Metadata and documentation for each microservice
Interoperable ➜ Can be queried from any programming language, work environment, ...
Reusable ➜ Version control through containerization

WHAT'S NEXT?

- Development of new microservices
- Maintaining current microservices
- Hackathons directed to developpers/users
- Current development : a microservice to retrieve PNG and MOL stored in a S3 storage

References: [1] Kopczynski D, *et al*. **2022** *Anal Chem*. Apr 26;94(16):6097−101. [2] Willighagen EL, et al. **2017** *J Cheminformatics*. Dec;9(1):33. [3] Heller SR, *et al*. **2015** *J Cheminformatics*. May 30;7:23. [4] Emami Khoonsari P, *et al*. **2019** *Bioinformatics*. Oct 1;35(19):3752-3760.
Acknowledgments: The master internship was supported by Clermont Auvergne University. This work is supported by the French Ministry of Research and National Research Agency as part of the French metabolomics and fluxomics infrastructure (MetaboHUB-ANR-INBS-0010).

faustine.souc@inrae.fr

Centre Auvergne Rhône Alpes
Site de Theix
63122 Saint Genès Champanelle