

Extending SQAaaS
Updates and Upgrades

Iván Palomo Llavona

Ibergrid 2026

Index

- What is SQAaaS?
- How does it evaluate?
- Use cases
- Upgrades & maintenance
- Future developments



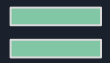
What is SQAaaS?

Software
Quality
Assurance
as a
Service

- Platform
- Bring novel quality practices
- Developing software in research environments
- Following quality conventions boosts



1. Adoption
2. Reusability
3. Sustainability



Steady Innovation

The different evaluation criteria

Accessibility	QC.Acc	Promote the accessibility of the source code as a public resource
Documentation	QC.Doc	Ensures the beneficial documents related with the code, both in terms of covering the target audience and relevant to allowing external contribution
FAIRness	QC.FAIR	Set of all criterion for FAIRness assessment, groups all the FAIR indicators which can be validated by the associated tools
Licensing	QC.Lic	Resolves the legal aspects of source code reuse through the presence of license files
Code Metadata	QC.Met	Promotes crediting and identification for software in research publications
Static Security	QC.Sec	Protects by helping detect unsecure patterns when writing code
Code style	QC.Sty	Fosters readability of the code by following style standards
Unit testing	QC.Uni	Specific tests to be performed to verify the code
Versioning	QC.Ver	Shows that the code has a proper versioning approach

Powered by open source tools

Code Style and Security (QC.Sty & QC.SEC)

	Python	Golang	Ruby	Java	JavaScript	JSON	Dockerfile
Code Style (QC.Sty)	flake8	staticcheck	rubocop RuboCop	checkstyle checkstyle	stylelint	jsonlint JSON	hadolint
Security (QC.Sec)	bandit Bandit	gosec					

Fair (QC.FAIR)

Fair Eva	F-UJI

Streamlined support for new tools (3 steps)

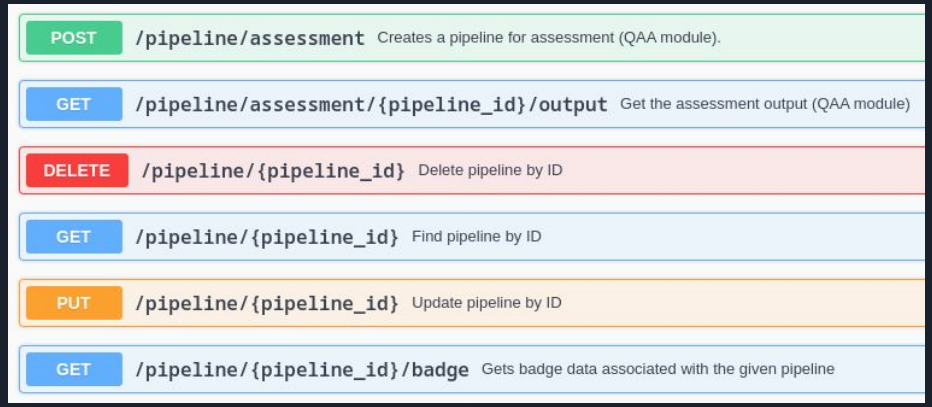
1. Set the environment for the tool (Docker image)
2. Add the tool details in SQAaaS tooling metadata: docker image, files to search, arguments, QA criterion, etc.
3. Create a plugin to parse the output of the tool

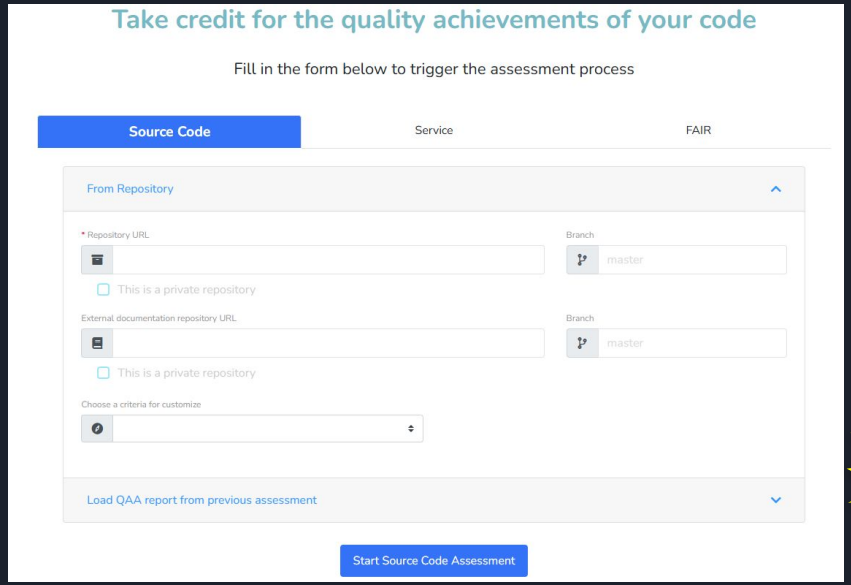
NEW!

Accessing SQAaaS Quality assessment

↘ **SQAaaS** can be accessed in diferent ways according to your needs

- Programmatic QA assessment (RESTful API) and awarding of digital research content
- A Web Interface to simplify usage: SQAaaS Web
- GitHub actions or Gitlab-CI and Gitlab Components



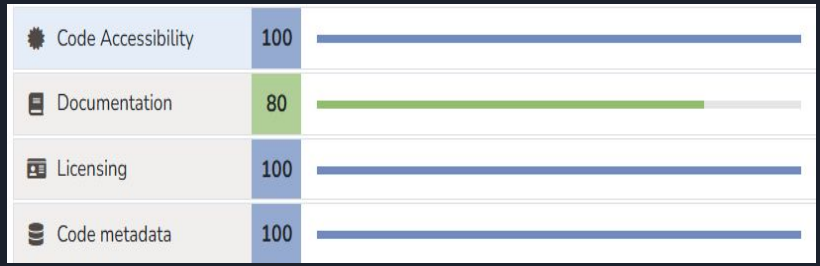


NEW!

QA criteria & digital badges



	Bronze	Silver	Gold
Accessibility (QC.Acc)	✔	✔	✔
Code Metadata (QC.Met)		✔	✔
Code Style (QC.Sty)			✔
Documentation (QC.Doc)	✔	✔	✔
Licensing (QC.Lic)	✔	✔	✔
Security Static Analysis (QC.Sec)			✔
Unit Testing (QC.Uni)			✔
Versioning (QC.Ver)		✔	✔



The QA criteria...

- Have **different test with different levels** of criticality
  - *MUST/SHOULD/MAY*
- **Not all** of them can be **automatically evaluated. Some require user feedback** for a successful automated evaluation (Unit testing)

Use Cases

The SQAaaS platform is in use in different contexts:

- **EuroHPC CoE flagship codes**
- **Digital Twin projects**











Now what has changed, how have we evolved?



We followed our user's needs, and we adapted to the updates in the tools we were employing

ENVRI-Hub

Upgrades

Gitlab CI/Components

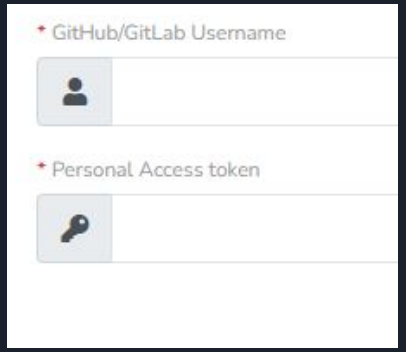
Private repositories

Jupyter notebooks

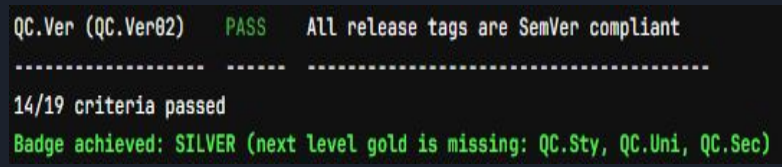
General

QC ID	Result	Explanation
QC.Acc (QC.Acc01)	PASS	Source code uses Git for version control
QC.Doc (QC.Doc06.1)	PASS	A README file is present in the code repository
QC.Doc (QC.Doc06.3)	PASS	A CODE_OF_CONDUCT file is present in the code repository
QC.Doc (QC.Doc06.2)	PASS	A CONTRIBUTING file is present in the code repository
QC.Doc (QC.Doc01.1)	PASS	Documentation resides in the same repository as code





Gitlab actions and github CI/ components

	GitHub	Gitlab
	https://github.com/EOSC-synergy/sqaaas-assessment-action	https://gitlab.com/sqaaas/sqaaas-gitlab-ci
Files	 sqaaasaction.yml	 gitlab_ci.yml
Process	Add the files and activate github actions	Add the files and activate the gitlab CI
Output		 

Jupyter notebooks evaluator



pynb lint

```
STATISTICS

 Cells                  Markdown usage          Code modularization
 Total cells: 22        Markdown titles: 18     Number of functions: 1
 Code cells: 13         Markdown lines: 22      Number of classes: 0
 Markdown cells: 9
 Raw cells: 0

LINTING RESULTS

(non-linear-execution)
     Notebook cells have been executed in a non-linear order.
     Recommendation: Re-run your notebook top to bottom to ensure it is
     reproducible.


(imports-beyond-first-cell)
     Import statements found beyond the first cell of the notebook.
     Recommendation: Move import statements to the first code cell to make your
     notebook dependencies more explicit.

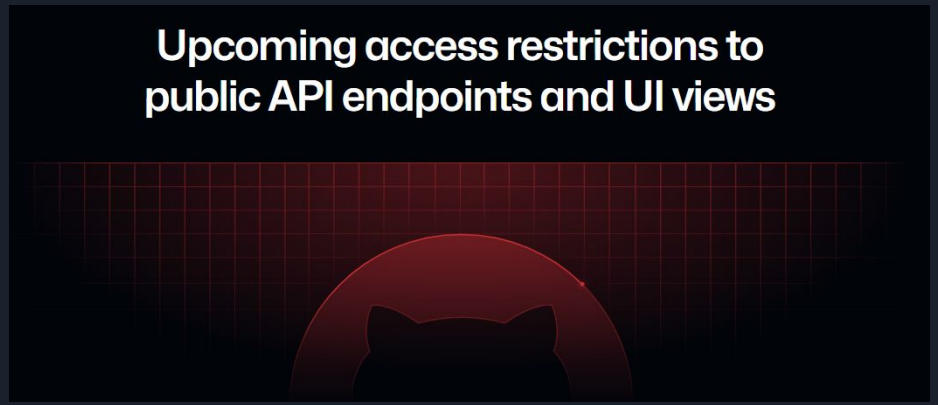

(missing-closing-MD-text)
     The final notebook cells (i.e., the last 3 cells in the notebook) contain
     no Markdown text.
     Recommendation: Conclude your notebook by describing what you have
     accomplished in one or more concluding Markdown cells.


(empty-cells)
     Empty cells are present in the notebook.
     Recommendation: Keep your notebook clean by deleting unused cells.
     Affected cells: indexes[20, 21]
```

Upgrades and maintenance

namegenerator -> randomgennames

This package has been removed from Pypi and cannot be used anymore.





Future developments

- Extend the SQAaaS platform to perform the quality validation of large-scale workflows handling the data volumes generated by HPC simulations

- Create a small badging platform to certificate the badges. Since the old one no longer is available

- Improve the SQAaaS engine, with a Kubernetes deployment (more on that in SQAaaS multiplatforms integration presentation later)

Next project

EARTH-AID



In conclusion

“Building software quality for researchers, focusing on threir needs”

Growing:

Gitlab-CI, jupyter notebooks, private evaluation

Maintaining:

Constant updates an improvements, that helps us navigate the new technologies

Future:

Evaluating EARTH- AID workflows and supporting our own badging

Any questions?

Thank you for your attention!

Iván Palomo Llavona
palomo@ifca.es

