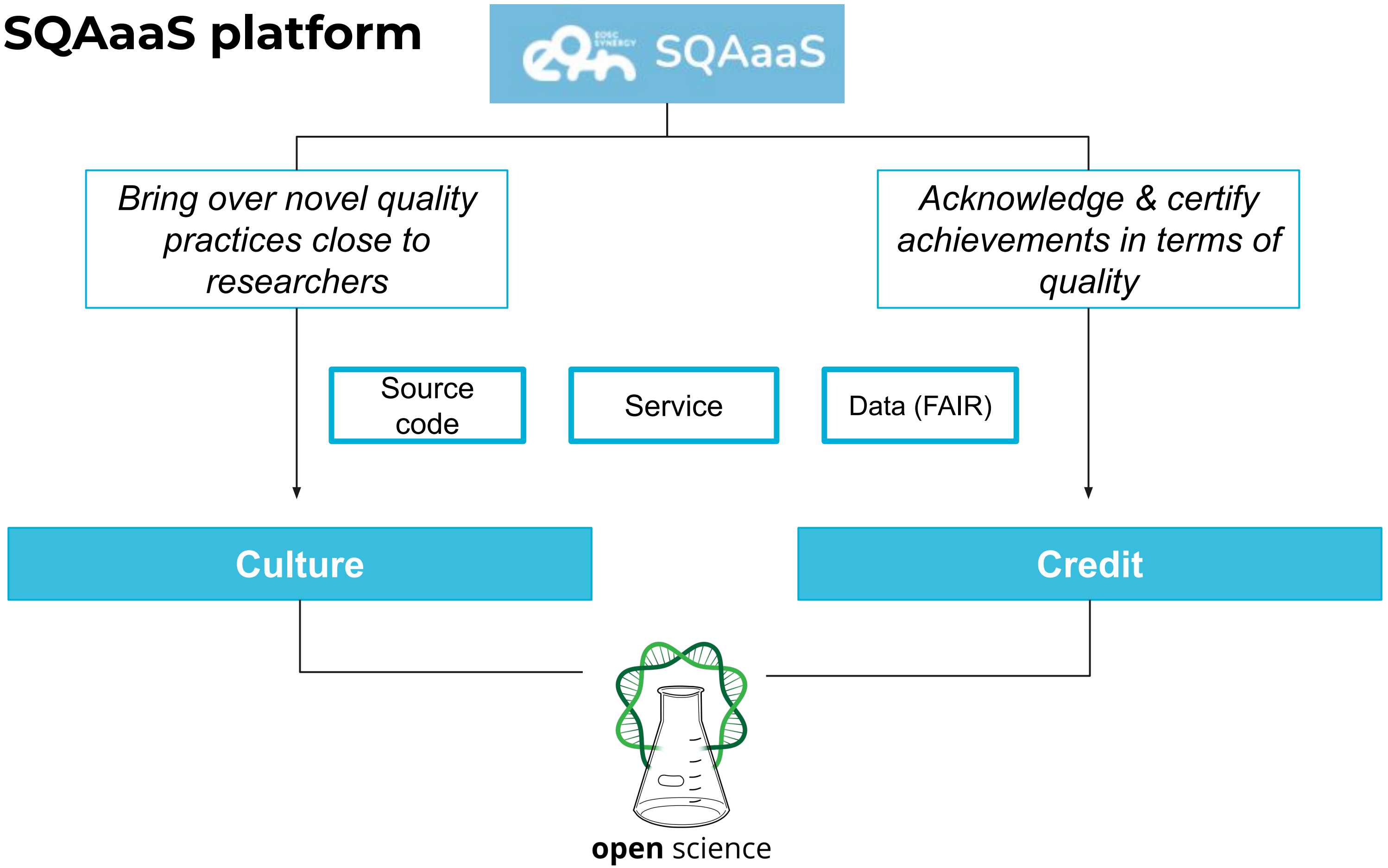
The SQA as a Service (SQAaaS) platform

Adopt and get recognition for quality practices in software development

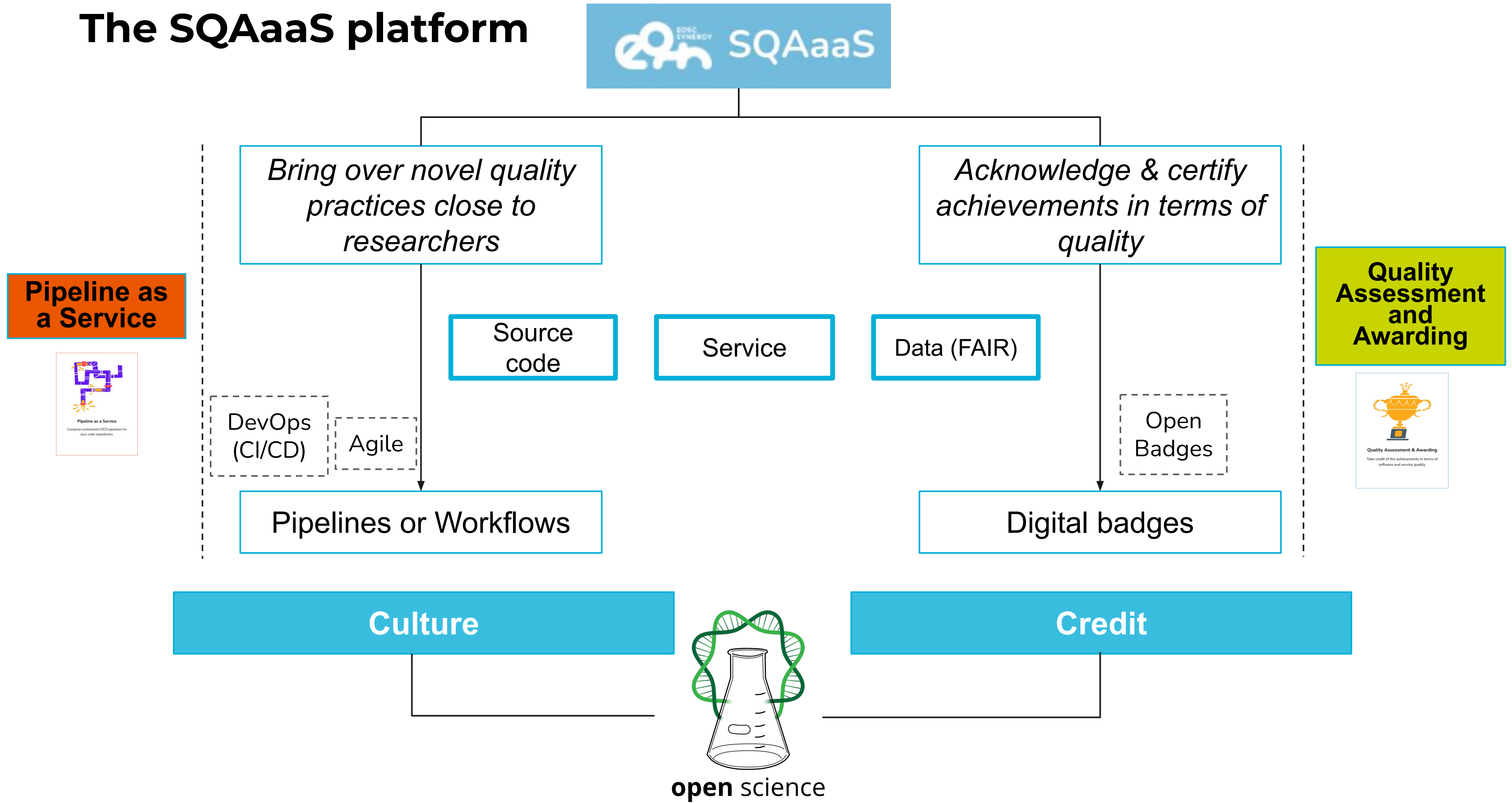
Pablo Orviz
orviz@ifca.unican.es
IFCA-CSIC

Samuel Bernardo
samuel@lip.pt
LIP

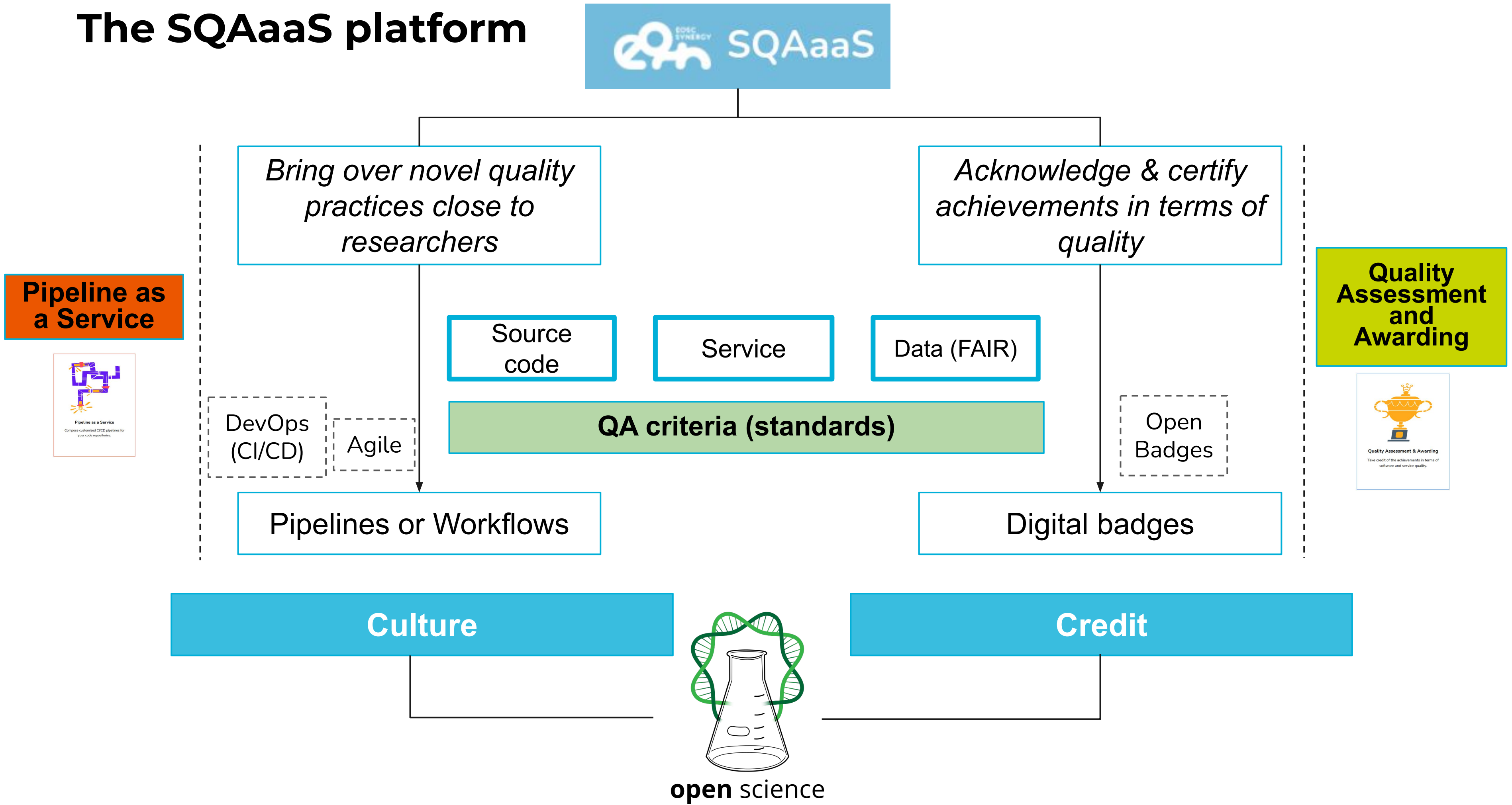
The SQAaaS platform


SQAaaS
EOSC SYNERGY
Bring over novel quality practices close to researchers
Acknowledge & certify achievements in terms of quality
Source code
Service
Data (FAIR)
Culture
Credit
open science

The SQAaaS platform


EOSC SYNERGY SQAaaS
Bring over novel quality practices close to researchers
Acknowledge & certify achievements in terms of quality
Pipeline as a Service
Pipeline as a Service
Compose customized CI/CD pipelines for your code repositories.
Source code
Service
Data (FAIR)
Quality Assessment and Awarding
Quality Assessment & Awarding
Take credit of the achievements in terms of software and service quality.
DevOps (CI/CD)
Agile
Open Badges
Pipelines or Workflows
Digital badges
Culture
Credit
open science

The SQAaaS platform


EOSC SYNERGY SQAaaS
Bring over novel quality practices close to researchers
Acknowledge & certify achievements in terms of quality
Pipeline as a Service
Pipeline as a Service
Compose customized CI/CD pipelines for your code repositories.
Source code
Service
Data (FAIR)
DevOps (CI/CD)
Agile
QA criteria (standards)
Open Badges
Quality Assessment and Awarding
Quality Assessment & Awarding
Take credit of the achievements in terms of software and service quality.
Pipelines or Workflows
Digital badges
Culture
Credit
open science

Quality standards

Source code

10.20350/digitalCSIC/12543

A set of Common Software Quality Assurance Baseline Criteria for Research Projects

A DOI-citable version of this manuscript is available at http://hdl.handle.net/10261/160086.

- **Code Accessibility**
- **Code Workflow**
- **Code Management**
- **Code Review**
- **Licensing**
- **Code Metadata**
- **Documentation**
- **Code Style**
- **Unit Testing**
- **Security**
- **Automated Deployment**
- **Semantic Versioning**
- **Test Harness**
- **Test-Driven Development**
- **Automated Delivery**

Services

10.20350/digitalCSIC/12533

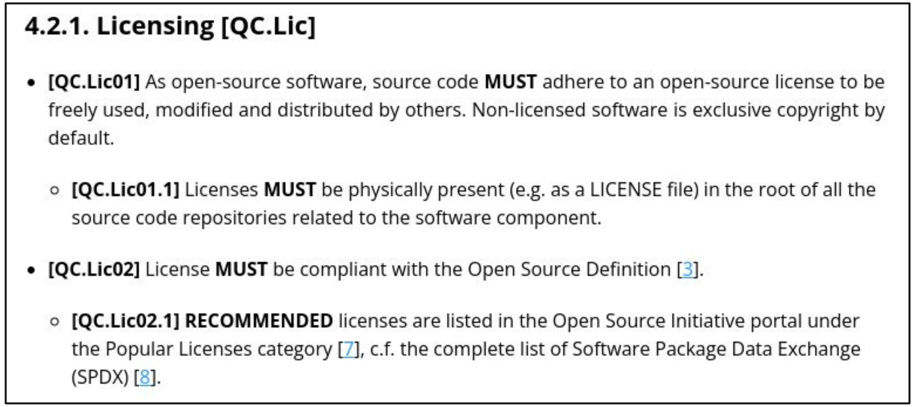
A set of Common Service Quality Assurance Baseline Criteria for Research Projects

A DOI-citable version of this manuscript is available at http://hdl.handle.net/.

This manuscript was automatically generated on 29-04-2020.

- **Automated Deployment**
- **API Testing**
- **Integration Testing**
- **Functional tests**
- **Performance tests**
- **Security**
- **Documentation**
- **Policies**
- **Support**
- **Monitoring**
- **Metrics**

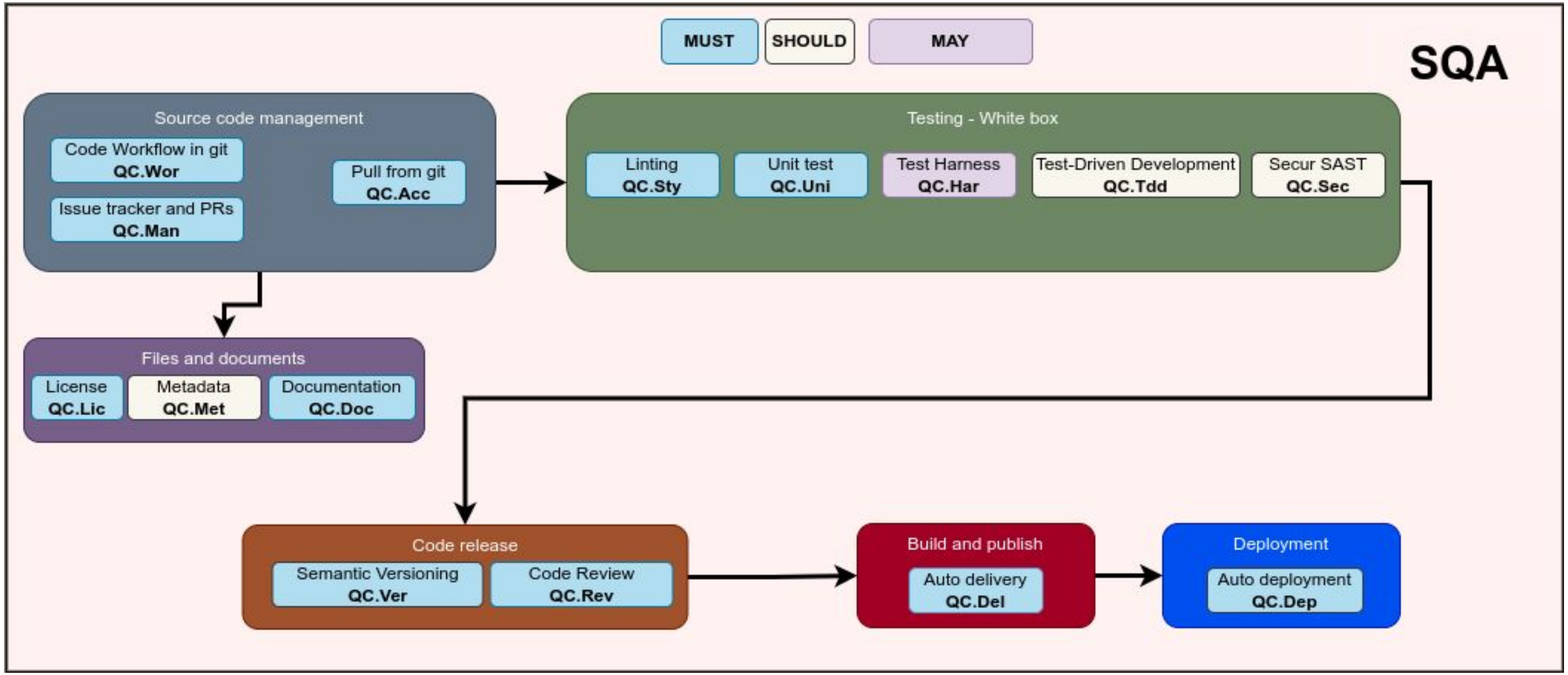
- Documents are openly **managed through github repositories**:
  - https://github.com/indigo-dc/sqa-baseline
  - https://github.com/EOSC-synergy/service-qa-baseline



4.2.1. Licensing [QC.Lic]

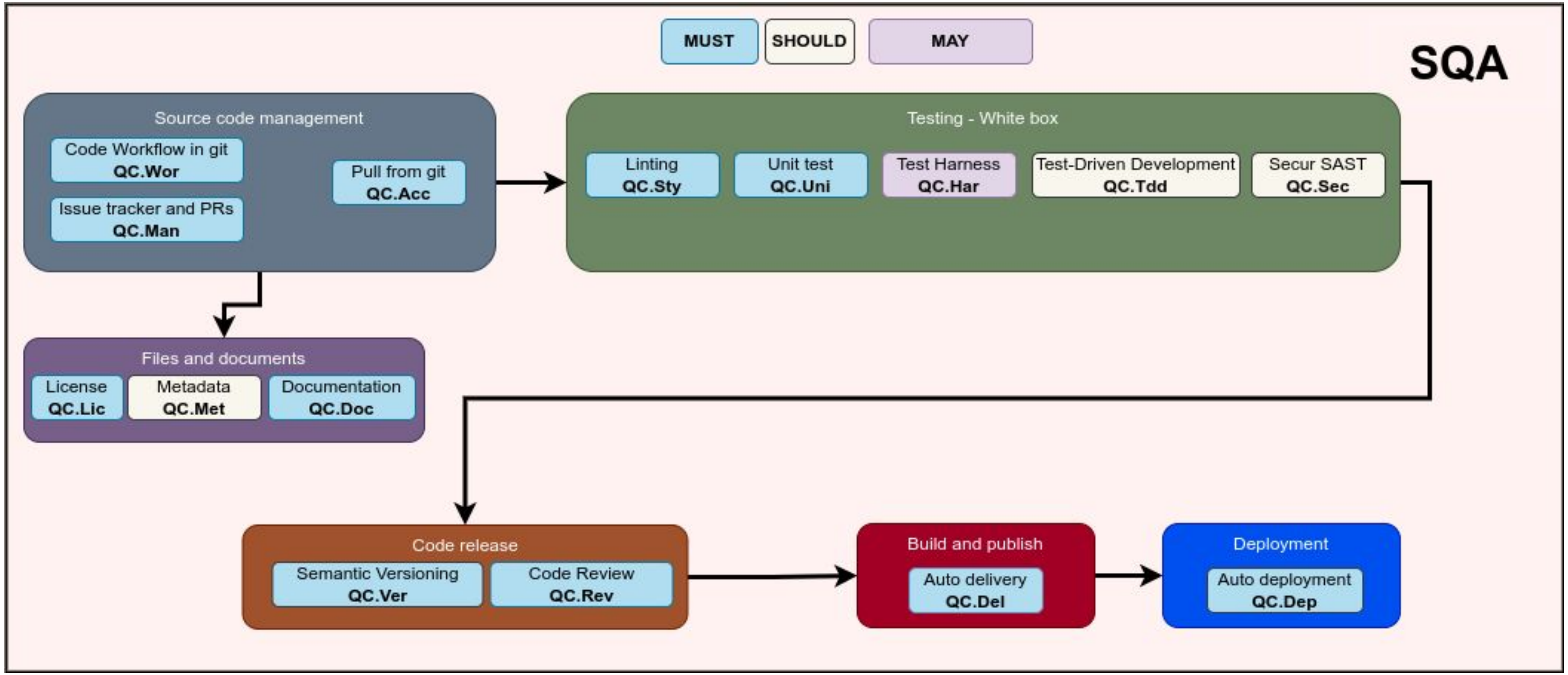
- **[QC.Lic01]** As open-source software, source code **MUST** adhere to an open-source license to be freely used, modified and distributed by others. Non-licensed software is exclusive copyright by default.
  - **[QC.Lic01.1]** Licenses **MUST** be physically present (e.g. as a LICENSE file) in the root of all the source code repositories related to the software component.
- **[QC.Lic02]** License **MUST** be compliant with the Open Source Definition [3].
  - **[QC.Lic02.1] RECOMMENDED** licenses are listed in the Open Source Initiative portal under the Popular Licenses category [7], c.f. the complete list of Software Package Data Exchange (SPDX) [8].

Quality standards: criticality

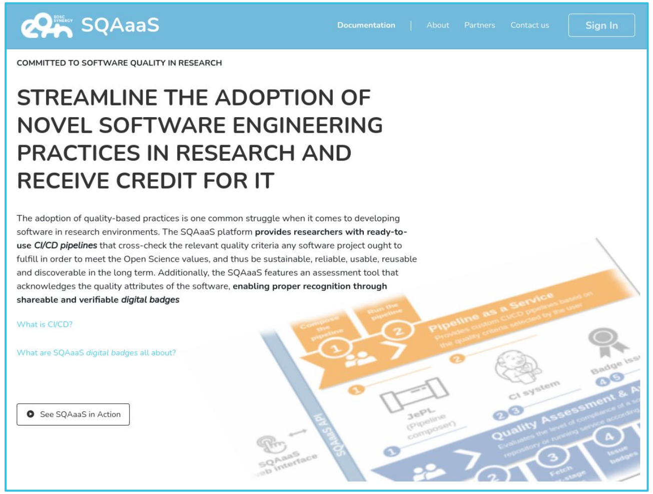


Quality standards: criticality

Covers 3/6 categories from the EOSC Task Force: "Ensure Research Software Quality" (EOSC-SWRelMan, EOSC-Test, EOSC-SrvOps)



The SQAaaS portal



COMMITTED TO SOFTWARE QUALITY IN RESEARCH

STREAMLINE THE ADOPTION OF NOVEL SOFTWARE ENGINEERING PRACTICES IN RESEARCH AND RECEIVE CREDIT FOR IT

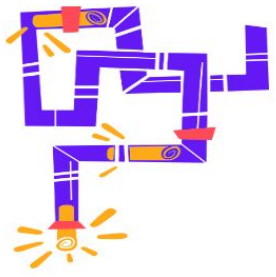
The adoption of quality-based practices is one common struggle when it comes to developing software in research environments. The SQAaaS platform **provides researchers with ready-to-use *CI/CD pipelines*** that cross-check the relevant quality criteria any software project ought to fulfill in order to meet the Open Science values, and thus be sustainable, reliable, usable, reusable and discoverable in the long term. Additionally, the SQAaaS features an assessment tool that acknowledges the quality attributes of the software, **enabling proper recognition through shareable and verifiable *digital badges***

What is CI/CD?

What are SQAaaS *digital badges* all about?

https://sqaaas.eosc-synergy.eu

01 Pipeline as a Service



Pipeline as a Service

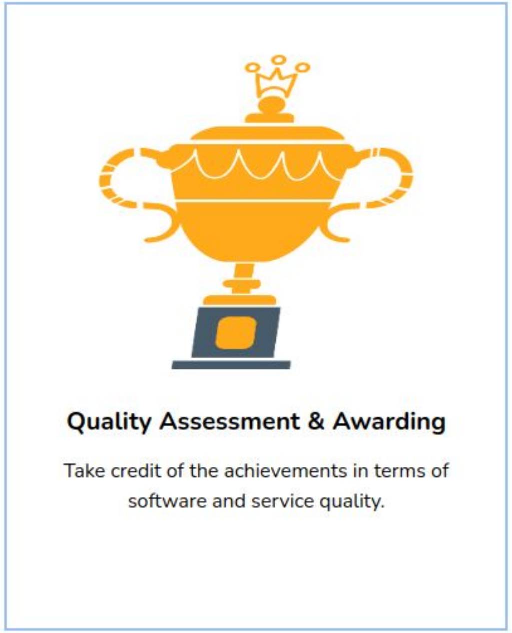
Compose customized CI/CD pipelines for your code repositories.

02 Quality Assessment & Awarding

Quality Assessment & Awarding

Take credit of the achievements in terms of software and service quality.

Quality assessment & awarding


Quality Assessment & Awarding

Take credit of the achievements in terms of software and service quality.

Why assessment is important?

Adopt quality conventions to **ease the adoption and sustainability** of the digital object (source code, service and/or data)

Types i) **Source code,** ii) (web) **service** iii) **Data FAIRness**

What awarding gives?

Digital badges contributes to reputation-building and crediting. They include metadata with references to the associated assessment results, improving the reusability and reproducibility of the awarded software releases.

SQAaaS & Digital Twins

Quality assessment and awarding

Event-driven V&V

- In response to *events* generated by code platforms (e.g. `push`, `pull request`, `tag creation`)
- Requires: integration with code platforms (GitHub Actions, GitLab CI)
- Suitable for: V&V of the whole workflow

As step within a DT workflow

- WfMSs trigger SQAaaS as part of the workflow execution
- Requires: integration with WfMS solution
- Suitable for: more granularity on the V&V work

Event-driven (workflow) validation in SQAaaS

Quality assessment and awarding

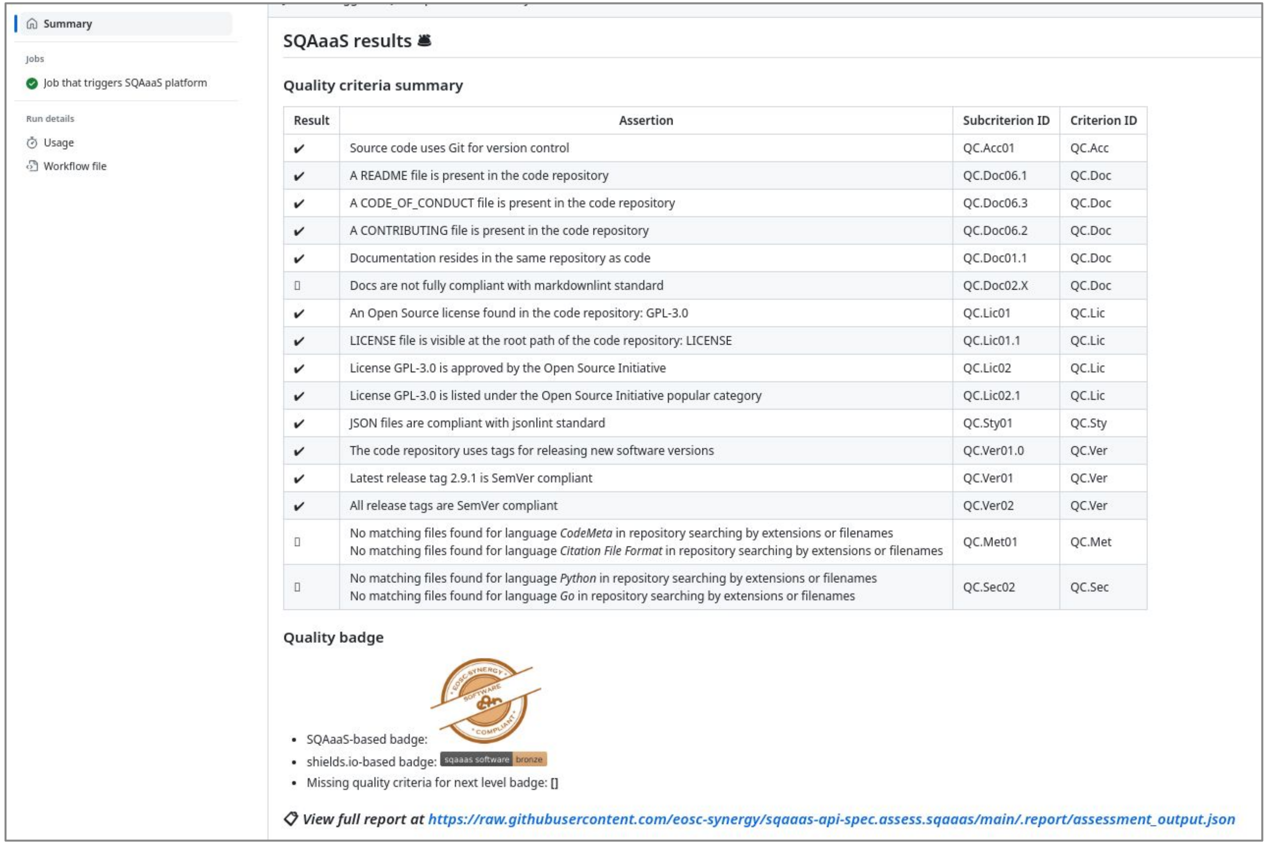
EOSC-synergy / **sqaaas-gh-action**

sqaaas-gh-action Public

About

GitHub action to trigger QA assessments in SQAaaS platform

- Limited (so far) to ***fixed QA assessments***, i.e. source code
- Will be extended to support the concept of ***custom assessments***



SQAaaS results

Quality criteria summary

Result	Assertion	Subcriterion ID	Criterion ID
✔	Source code uses Git for version control	QC.Acc01	QC.Acc
✔	A README file is present in the code repository	QC.Doc06.1	QC.Doc
✔	A CODE_OF_CONDUCT file is present in the code repository	QC.Doc06.3	QC.Doc
✔	A CONTRIBUTING file is present in the code repository	QC.Doc06.2	QC.Doc
✔	Documentation resides in the same repository as code	QC.Doc01.1	QC.Doc
▯	Docs are not fully compliant with markdownlint standard	QC.Doc02.X	QC.Doc
✔	An Open Source license found in the code repository: GPL-3.0	QC.Lic01	QC.Lic
✔	LICENSE file is visible at the root path of the code repository: LICENSE	QC.Lic01.1	QC.Lic
✔	License GPL-3.0 is approved by the Open Source Initiative	QC.Lic02	QC.Lic
✔	License GPL-3.0 is listed under the Open Source Initiative popular category	QC.Lic02.1	QC.Lic
✔	JSON files are compliant with jsonlint standard	QC.Sty01	QC.Sty
✔	The code repository uses tags for releasing new software versions	QC.Ver01.0	QC.Ver
✔	Latest release tag 2.9.1 is SemVer compliant	QC.Ver01	QC.Ver
✔	All release tags are SemVer compliant	QC.Ver02	QC.Ver
▯	No matching files found for language *CodeMeta* in repository searching by extensions or filenames No matching files found for language *Citation File Format* in repository searching by extensions or filenames	QC.Met01	QC.Met
▯	No matching files found for language *Python* in repository searching by extensions or filenames No matching files found for language *Go* in repository searching by extensions or filenames	QC.Sec02	QC.Sec

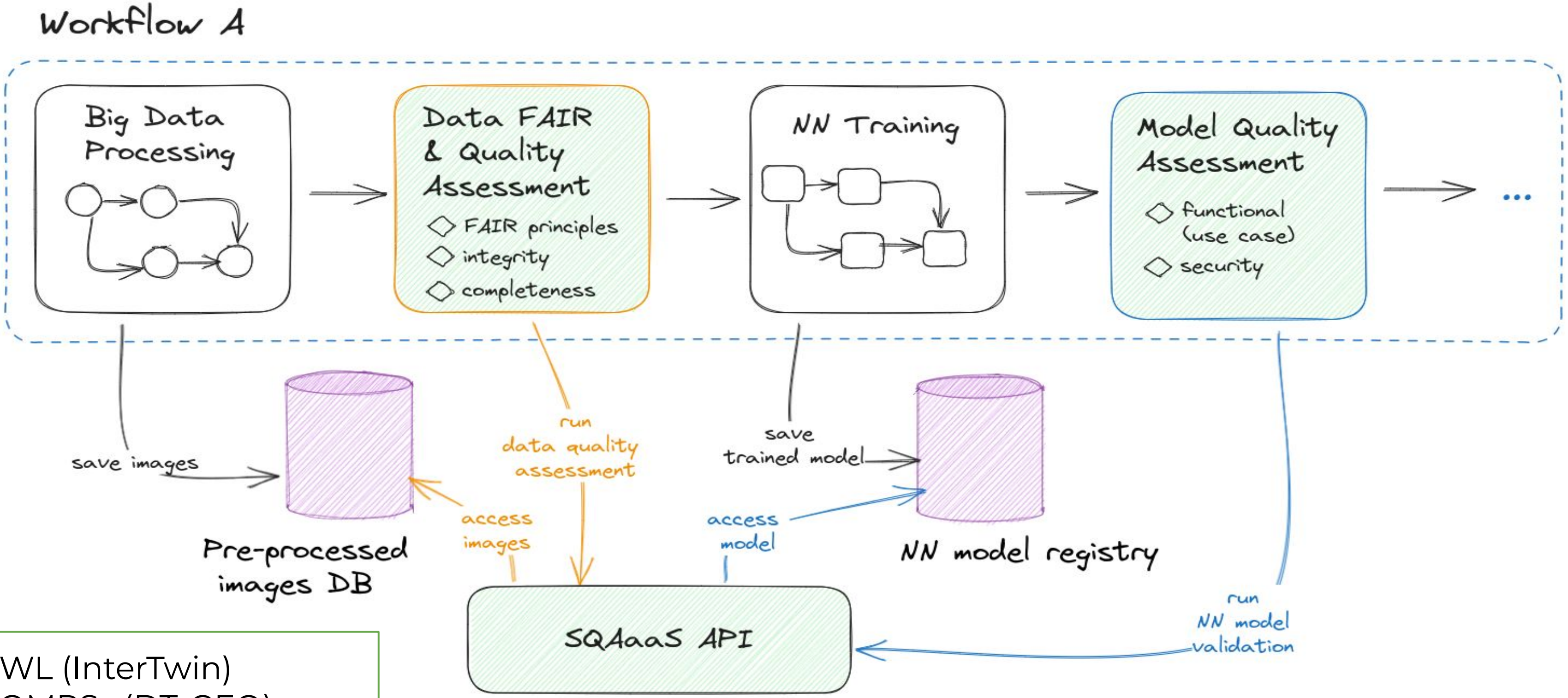
Quality badge

- SQAaaS-based badge:
- shields.io-based badge: sqaaas software bronze
- Missing quality criteria for next level badge: []

View full report at https://raw.githubusercontent.com/eosc-synergy/sqaaas-api-spec.assess.sqaaas/main/.report/assessment_output.json

SQAaaS validation within Workflow

Quality assessment and awarding

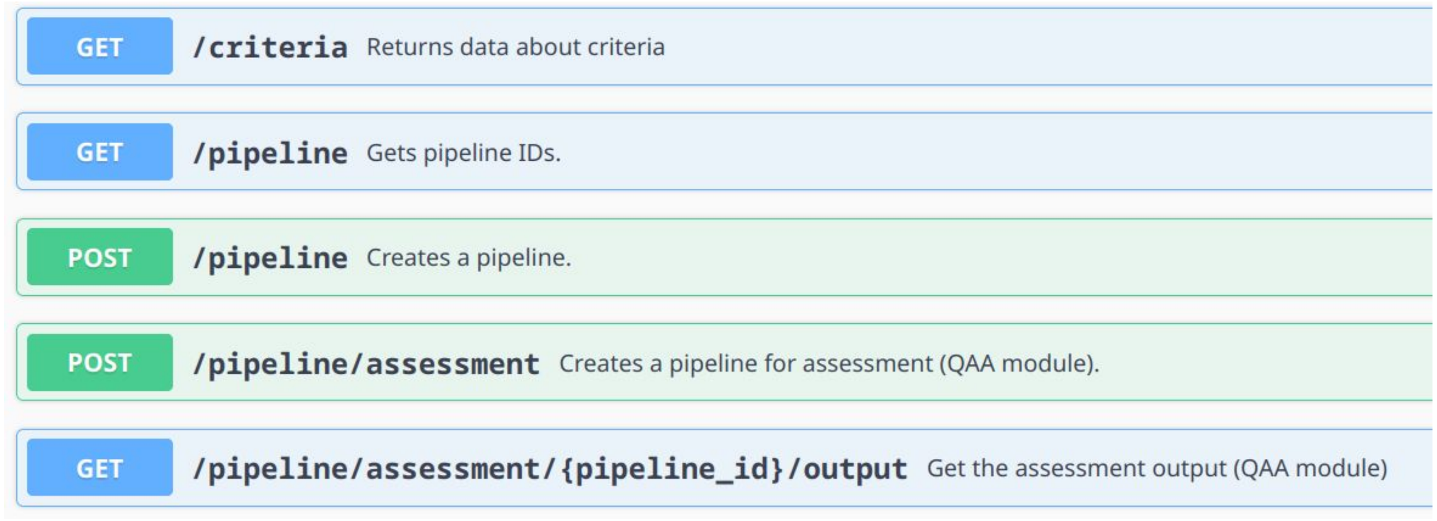


WfMS

- CWL (InterTwin)
- COMPSs (DT-GEO)
- Apache Airflow

New interface: **SQAaaS CLI**

- Interaction with SQAaaS API without the burden of required arguments encoding
- Better integration with CI automation services and workflow management systems
- Provides an implementation for the available SQAaaS API features
- Still in development and first release expected during October



GET **/criteria** Returns data about criteria

GET **/pipeline** Gets pipeline IDs.

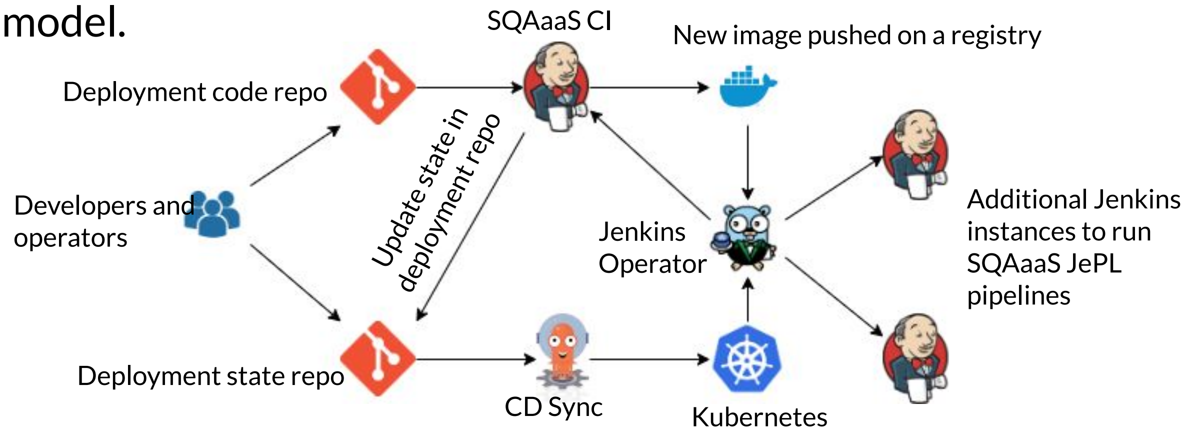
POST **/pipeline** Creates a pipeline.

POST **/pipeline/assessment** Creates a pipeline for assessment (QAA module).

GET **/pipeline/assessment/{pipeline_id}/output** Get the assessment output (QAA module)

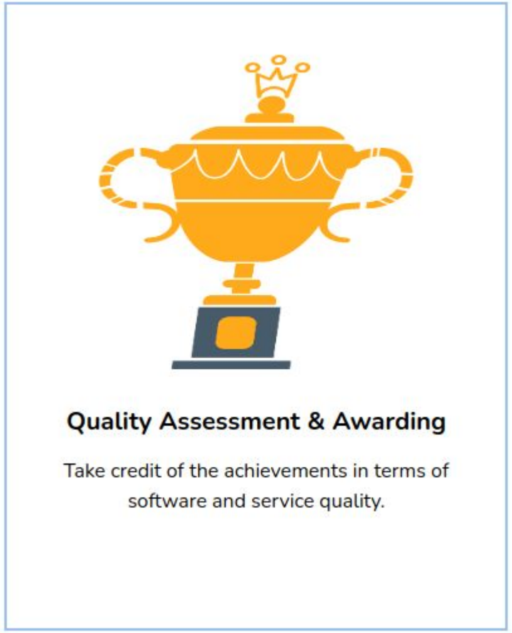
SQAaaS deployment automation

- GitOPS deployment model using ArgoCD implementation.
- Helm charts to deploy SQAaaS services over Kubernetes with code continuously tested over SQAaaS CI/CD.
- Kubernetes cluster management using Kubespray across multiple infrastructure platforms using Ansible and other supported provisioning tools.
- Jenkins Operator configuration as code approach to manage Jenkins pipeline system over Kubernetes, providing a solution for the integration with the GitOPS deployment model.


SQAaaS CI
New image pushed on a registry
Deployment code repo
Update state in deployment repo
Developers and operators
Jenkins Operator
Additional Jenkins instances to run SQAaaS JePL pipelines
Deployment state repo
CD Sync
Kubernetes

Quality assessment & awarding

Fixed assessments



Quality Assessment & Awarding

Take credit of the achievements in terms of software and service quality.

Demo

01 Source code assessment

02 Overview of service and FAIR assessment

Quality assessment of source code: our use case

From the EOSC-Synergy+KIT software stack:
https://github.com/EOSC-synergy/ibergrid-perf

EOSC PERFORMANCE

ibergrid-perf Public

forked from EOSC-synergy/eosc-perf

About

Performance of European Open Science Cloud

Let’s trigger QA assessment on the SQAaaS platform

Capabilities of QA assessments

QA assessment in SQAaaS relies on existing open-source tools

- **Only `git` is supported** as VCS tool
  - *Major requirement:* QA assessment is not performed for other VCSs
    - No real plans to support additional VCSs in the near future
- **Both public and private repositories** can be assessed
  - Private repositories are supported through personal tokens (GitHub, GitLab)
  - Work-in-progress: Vault secret management (best)
- **Programming languages:** `Python, Go(lang), Ruby, Java & Javascript`
  - Others/Misc: `JSON, Dockerfiles,` ..
  - Support for new programming languages upon internal roadmap/user request (e.g. C++)

sqaaas@ibergrid.eu

or through issues in:

- SQAaaS main repo https://github.com/EOSC-synergy/SQAaaS/issues/new/choose
- SQAaaS' tooling repo https://github.com/EOSC-synergy/sqaaas-tooling/issues

Supported tools: code management, style, security and documentation

Some QA criteria must be addressed by specific tools...

Code Management (QC.Man, QC.Acc)

git

Code Style and Security (QC.Sty, QC.Sec)

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

Documentation (QC.Doc)

	Markdown	reStructuredText
Documentation (QC.Doc)	`markdownlint`	`restructuredtext-lint`

QA badges for source code: QA criteria mappings

	Bronze	Silver	Gold
Accessibility (QC.Acc)	✓	✓	✓
Code Management (QC.Man)			✓
Code Metadata (QC.Met)		✓	✓
Code Style (QC.Sty)			✓
Code Workflow (QC.Wor)			✓
Delivery (QC.Del)			✓
Documentation (QC.Doc)	✓	✓	✓
Licensing (QC.Lic)	✓	✓	✓
Security Static Analysis (QC.Sec)			✓
Unit Testing (QC.Uni)			✓
Versioning (QC.Ver)		✓	✓

https://indigo-dc.github.io/sqa-baseline/

QA badges for services and data

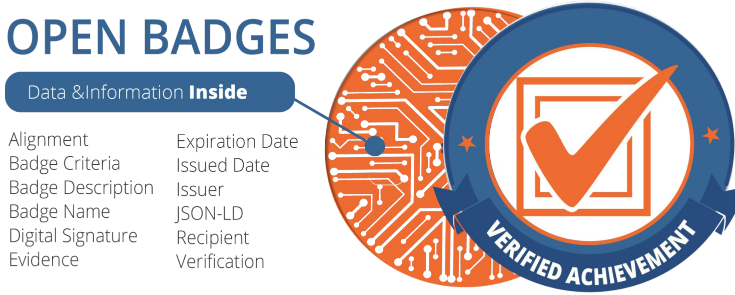
- Service assessment: only bronze badges
  - <u>Major requirement:</u> means for automated deployment (IM, Kubernetes, ..)
- FAIR assessment through FAIR-EVA tool
  - Added value: identify critical FAIR indicators

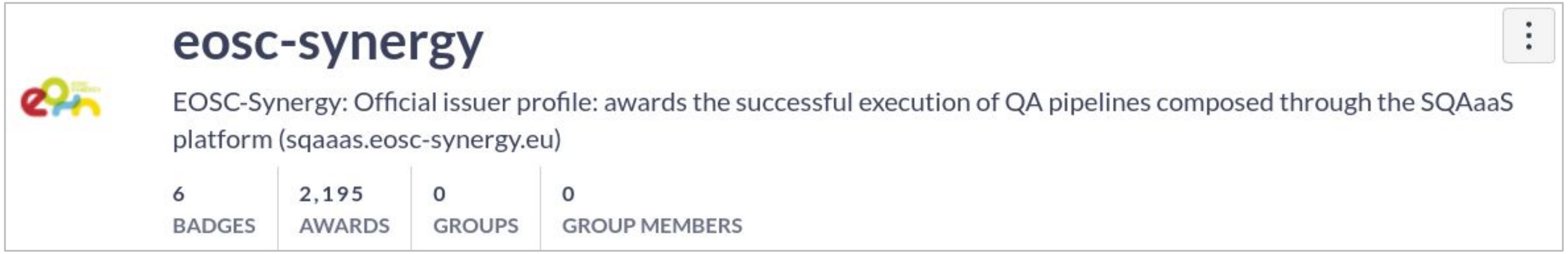
https://eosc-synergy.github.io/service-qa-baseline/

	Bronze	**Silver**	**Gold**
Deployment (SvcQC.Dep)	✓	✓	✓
API testing (SvcQC.API)			✓
Integration testing (SvcQC.Int)		✓	✓
Functional testing (SvcQC.Fun)			✓
Performance testing (SvcQC.Per)			✓
Documentation (SvcQC.Doc)	✓	✓	✓
Security Dynamic Analysis (SvcQC.Sec)			✓

About digital badges

- Digital badges issued by SQAaaS are based on **Open Badges specification**
- Provide additional metadata that is “baked” into the badge
    - In particular, *Evidence* property contains relevant info about the assessment process (such as links to build info, logs)
- Are shareable and verifiable
    - Issuer: eosc-synergy



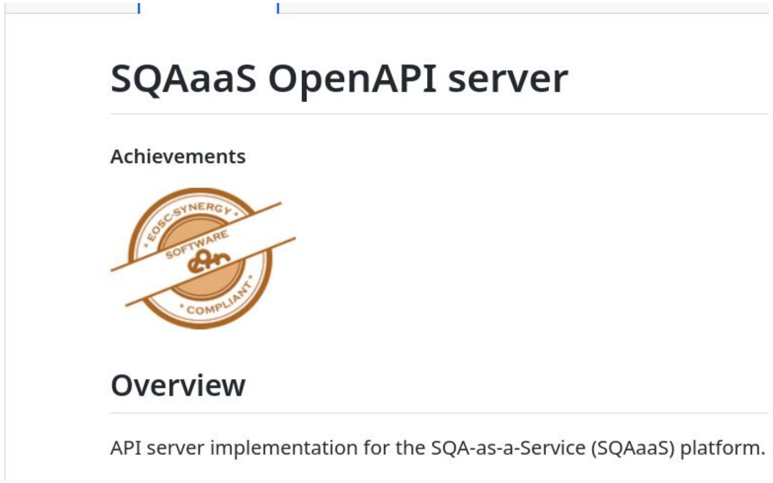

eosc-synergy

EOSC-Synergy: Official issuer profile: awards the successful execution of QA pipelines composed through the SQAaaS platform (sqaaas.eosc-synergy.eu)

6	2,195	0	0
BADGES	AWARDS	GROUPS	GROUP MEMBERS

Badges for source code: how can they be shared?

EOSC-Synergy badge image AND/OR shields.io badge image

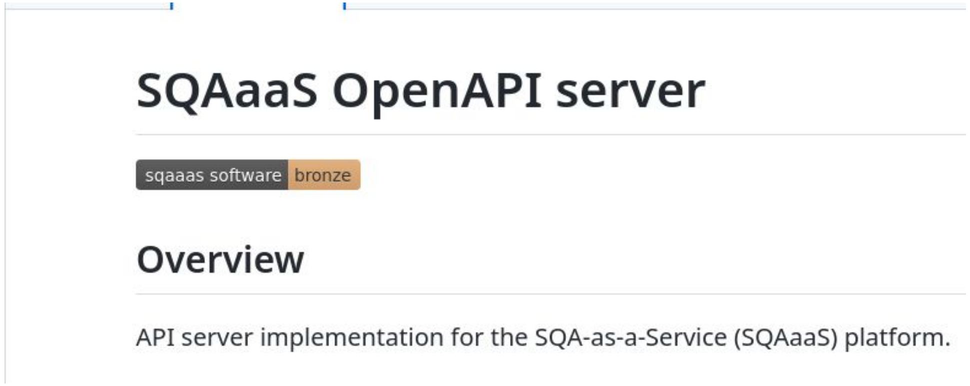


SQAaaS OpenAPI server

Achievements

Overview

API server implementation for the SQA-as-a-Service (SQAaaS) platform.

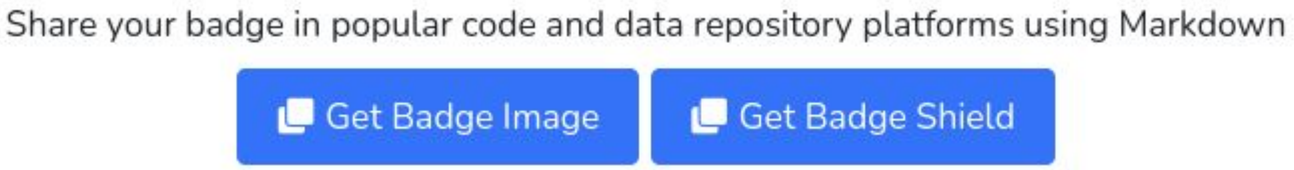


SQAaaS OpenAPI server

Overview

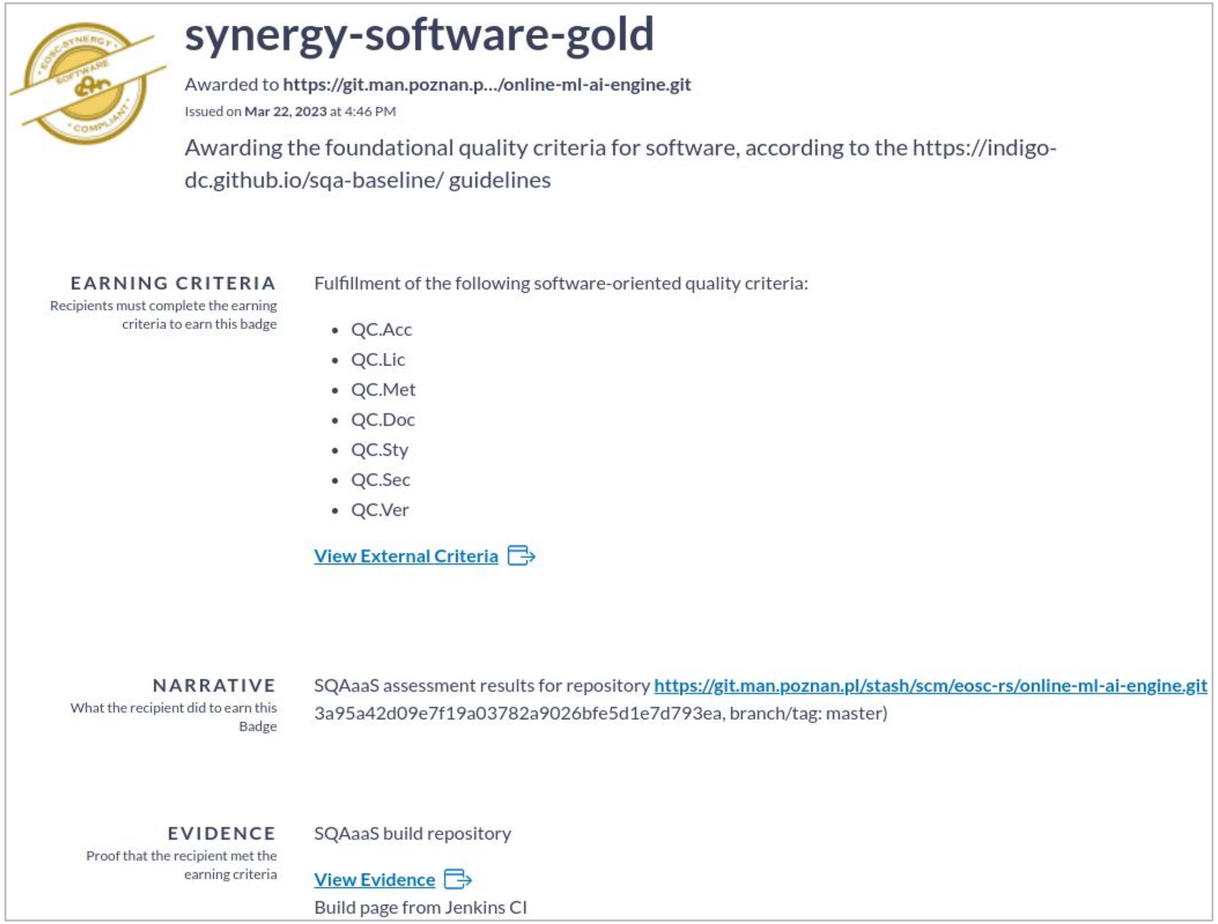
API server implementation for the SQA-as-a-Service (SQAaaS) platform.

Required Markdown code is generated to be copy/pasted in your README



Share your badge in popular code and data repository platforms using Markdown

About digital badges



synergy-software-gold

Awarded to **https://git.man.poznan.p.../online-ml-ai-engine.git**

Issued on **Mar 22, 2023** at 4:46 PM

Awarding the foundational quality criteria for software, according to the https://indigo-dc.github.io/sqa-baseline/ guidelines

EARNING CRITERIA
Recipients must complete the earning criteria to earn this badge

Fulfillment of the following software-oriented quality criteria:

- QC.Acc
- QC.Lic
- QC.Met
- QC.Doc
- QC.Sty
- QC.Sec
- QC.Ver

View External Criteria

NARRATIVE
What the recipient did to earn this Badge

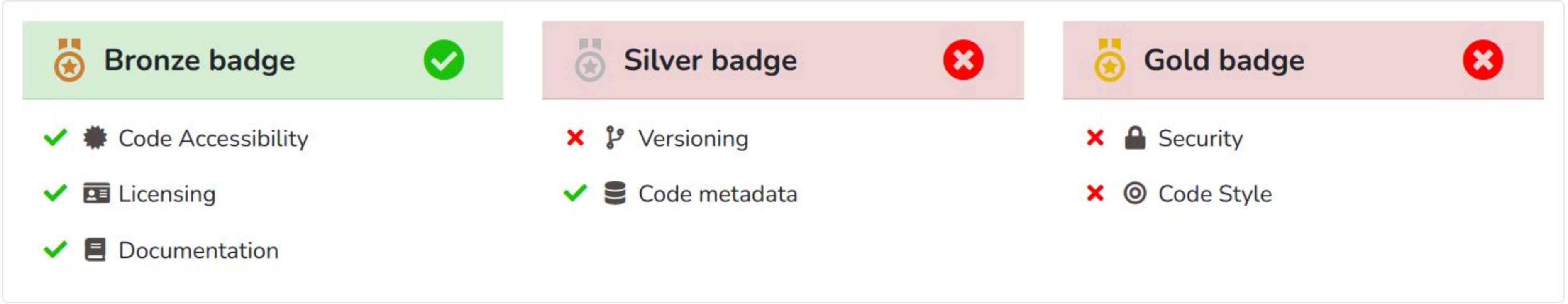
SQAaaS assessment results for repository https://git.man.poznan.pl/stash/scm/eosc-rs/online-ml-ai-engine.git 3a95a42d09e7f19a03782a9026bfe5d1e7d793ea, branch/tag: master)

EVIDENCE
Proof that the recipient met the earning criteria

SQAaaS build repository

View Evidence
Build page from Jenkins CI

Pipeline as a Service



Bronze badge ✓	Silver badge ✗	Gold badge ✗
✓ Code Accessibility	✗ Versioning	✗ Security
✓ Licensing	✓ Code metadata	✗ Code Style
✓ Documentation		

What is a CI/CD pipeline?

CI/CD pipelines automatize the CI/CD work so that it can be executed for every change in the code

CI/CD stands for Continuous Integration and Delivery

- Aims at improving the overall quality of the software during the development life cycle by its continuous (every change) verification and validation (V&V)
- meets both the functional (behavior-driven) and non-functional (usability-oriented) requirements

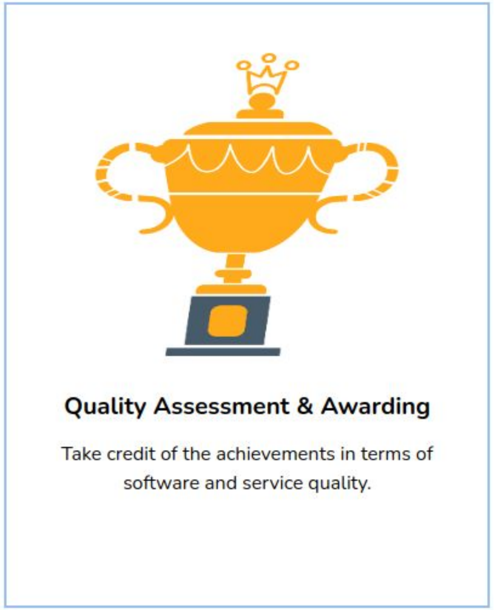
Demos

01 **Improve your software with SQAaaS assessment**

02 SQAaaS integration with git flow development process

03 JePL: improve your CI pipeline

01 Improve your software with SQAaaS assessment



Quality Assessment & Awarding

Take credit of the achievements in terms of software and service quality.

Steps

1.1 Review QC.Ver report

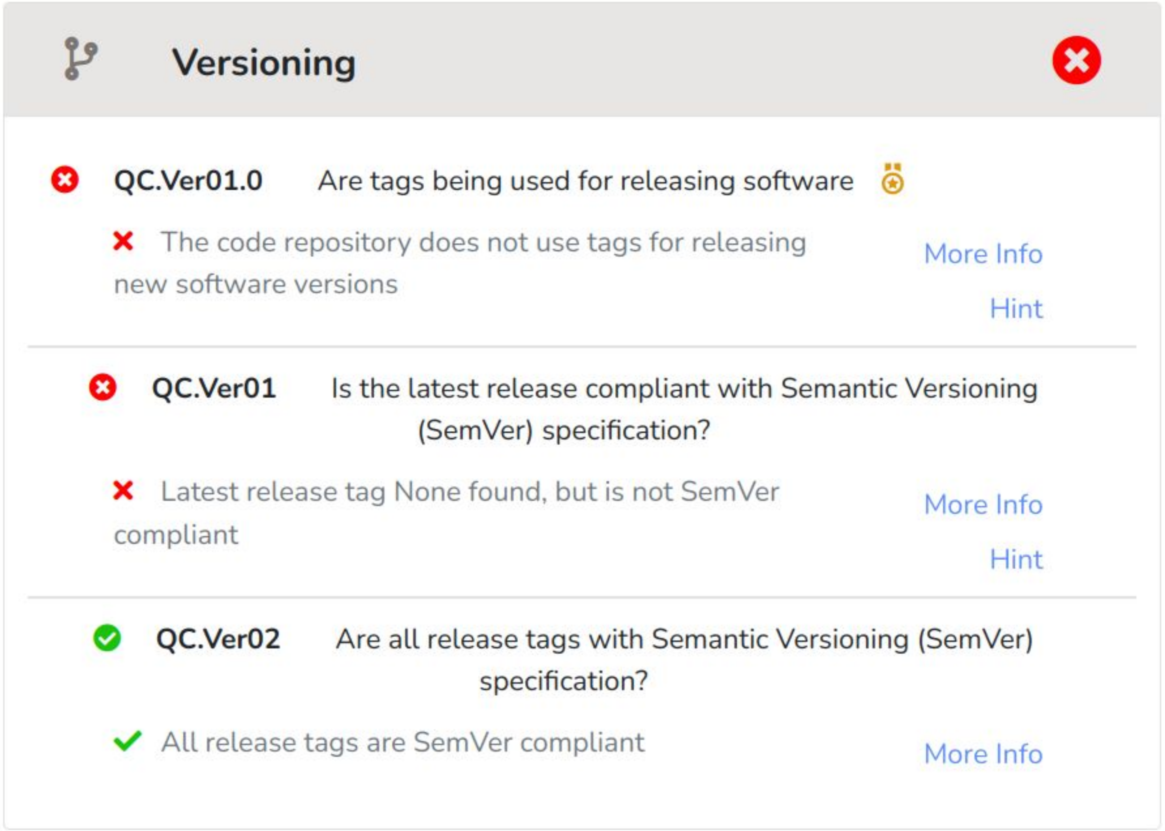
1.2 Correct the semantic versioning issue

1.3 Run SQAaaS assessment and check the silver badge

1.1 Review QC.Ver report

- Project has no tags yet
- We only need to create the first release tag

- Complying with semantic versioning, we are going to create the tag **1.0.0**



Versioning

QC.Ver01.0 Are tags being used for releasing software

✗ The code repository does not use tags for releasing new software versions

More Info

Hint

QC.Ver01 Is the latest release compliant with Semantic Versioning (SemVer) specification?

✗ Latest release tag None found, but is not SemVer compliant

More Info

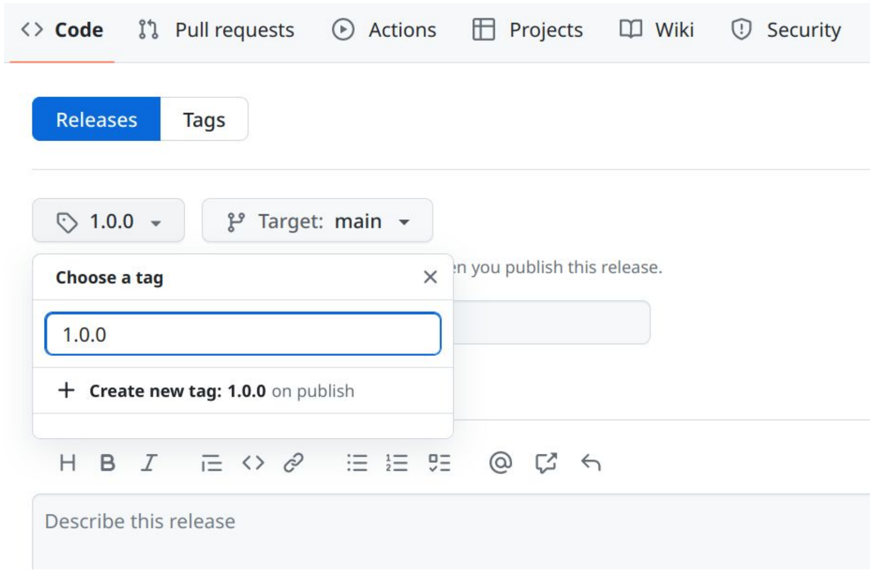
Hint

QC.Ver02 Are all release tags with Semantic Versioning (SemVer) specification?

✓ All release tags are SemVer compliant

More Info

1.2 Correct the semantic versioning issue



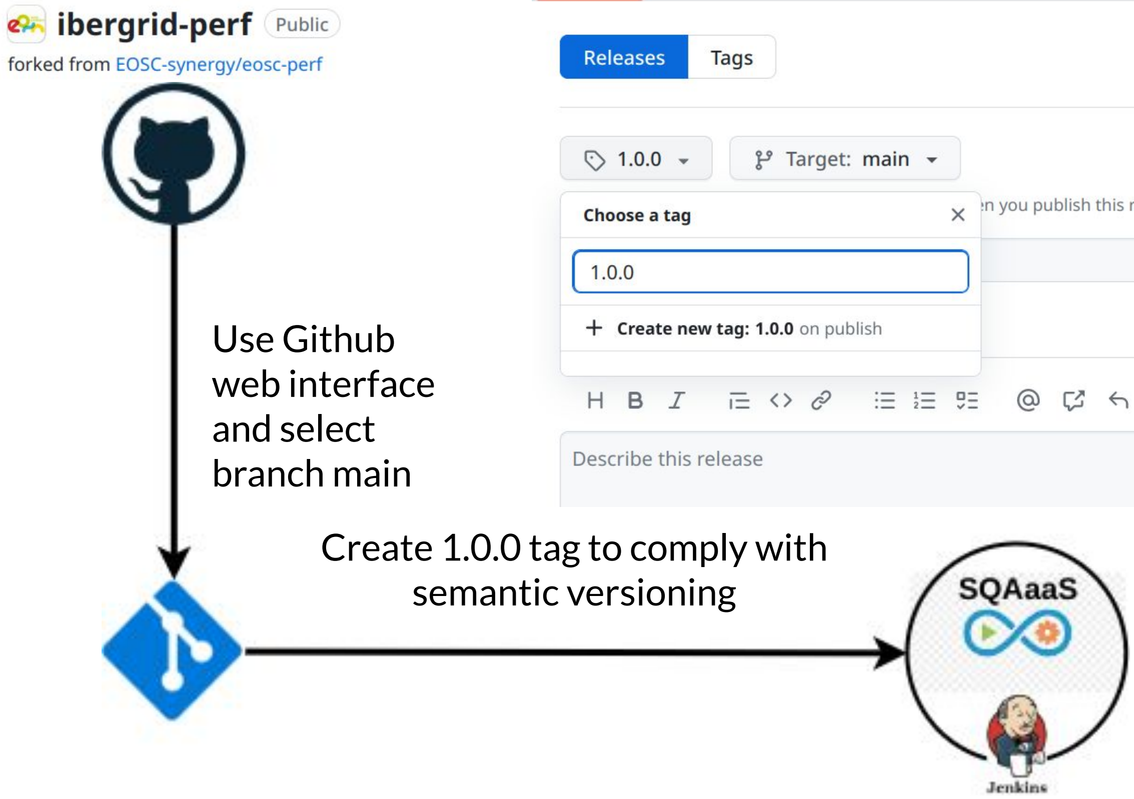
Choose a tag

1.0.0

+ Create new tag: 1.0.0 on publish

n you publish this release.

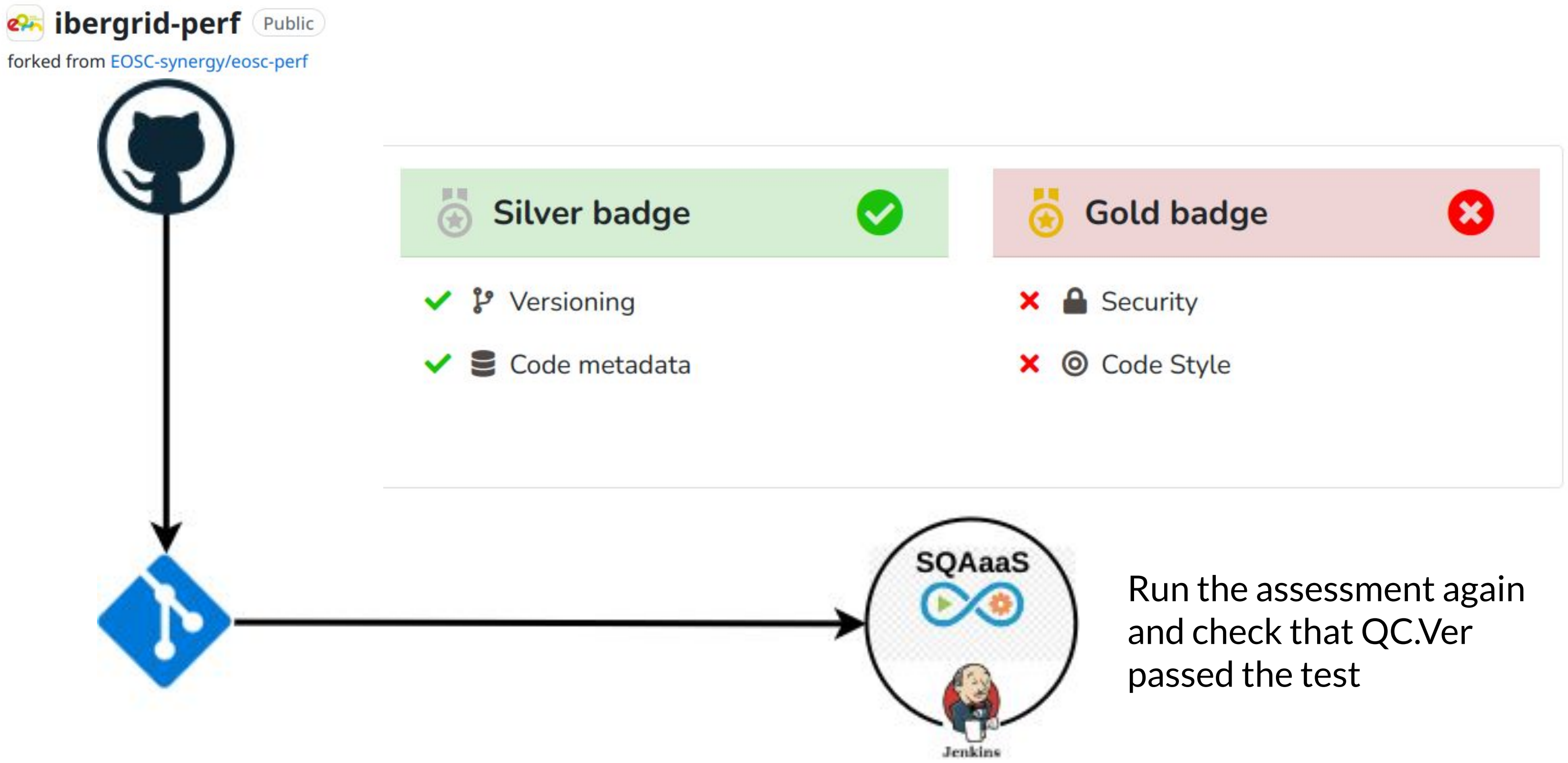
Describe this release



Use Github web interface and select branch main

Create 1.0.0 tag to comply with semantic versioning

1.3 Run SQAaaS assessment and check the silver badge

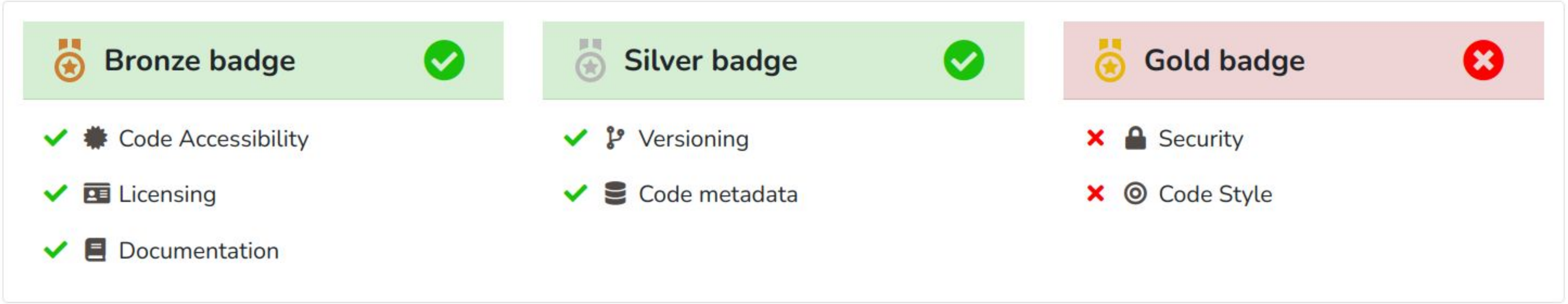

ibergrid-perf Public
forked from EOSC-synergy/eosc-perf

Silver badge ✓
- ✓ Versioning
- ✓ Code metadata

Gold badge ✗
- ✗ Security
- ✗ Code Style

Run the assessment again and check that QC.Ver passed the test

Pipeline as a Service



Bronze badge ✓	Silver badge ✓	Gold badge ✗
✓ Code Accessibility	✓ Versioning	✗ Security
✓ Licensing	✓ Code metadata	✗ Code Style
✓ Documentation		

What is a CI/CD pipeline?

CI/CD pipelines automatize the CI/CD work so that it can be executed for every change in the code

CI/CD stands for Continuous Integration and Delivery

- Aims at improving the overall quality of the software during the development life cycle by its continuous (every change) verification and validation (V&V)
- meets both the functional (behavior-driven) and non-functional (usability-oriented) requirements

Demos

01 Improve your software with SQAaaS assessment

02 **SQAaaS integration with git flow development process**

03 JePL: improve your CI pipeline

02 SQAaaS integration with git flow development process

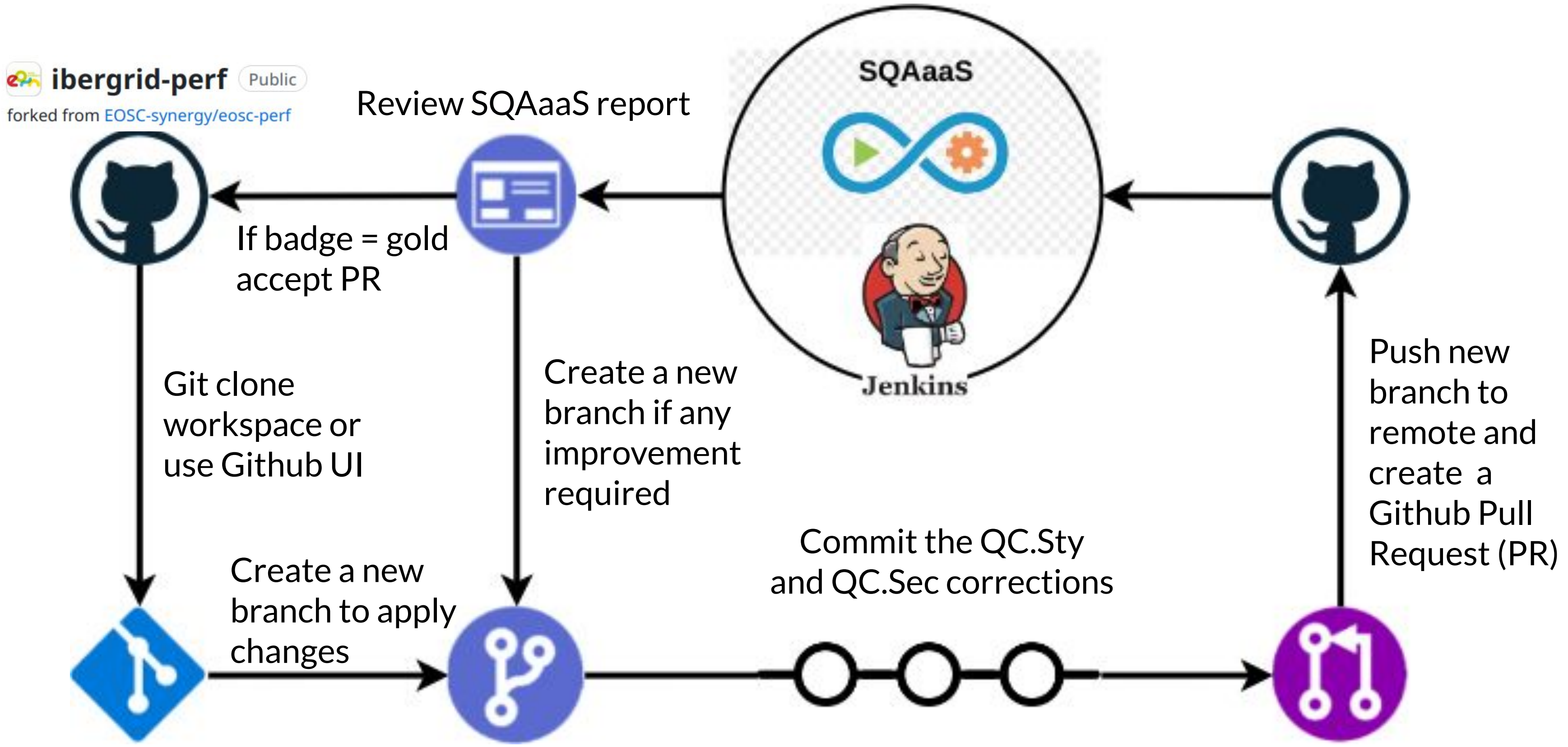
Pipeline as a Service

Compose customized CI/CD pipelines for your code repositories.

Steps

- 2.1 SQAaaS CI/CD and Git Flow integration
- 2.2 Review QC.Sty and QC.Sec report
- 2.3 Create a pipeline configuration to support the development
- 2.4 Merge the results into the production branch
- 2.5 Run the SQAaaS assessment and check the gold badge

2.1 SQAaaS CI/CD and Git Flow integration

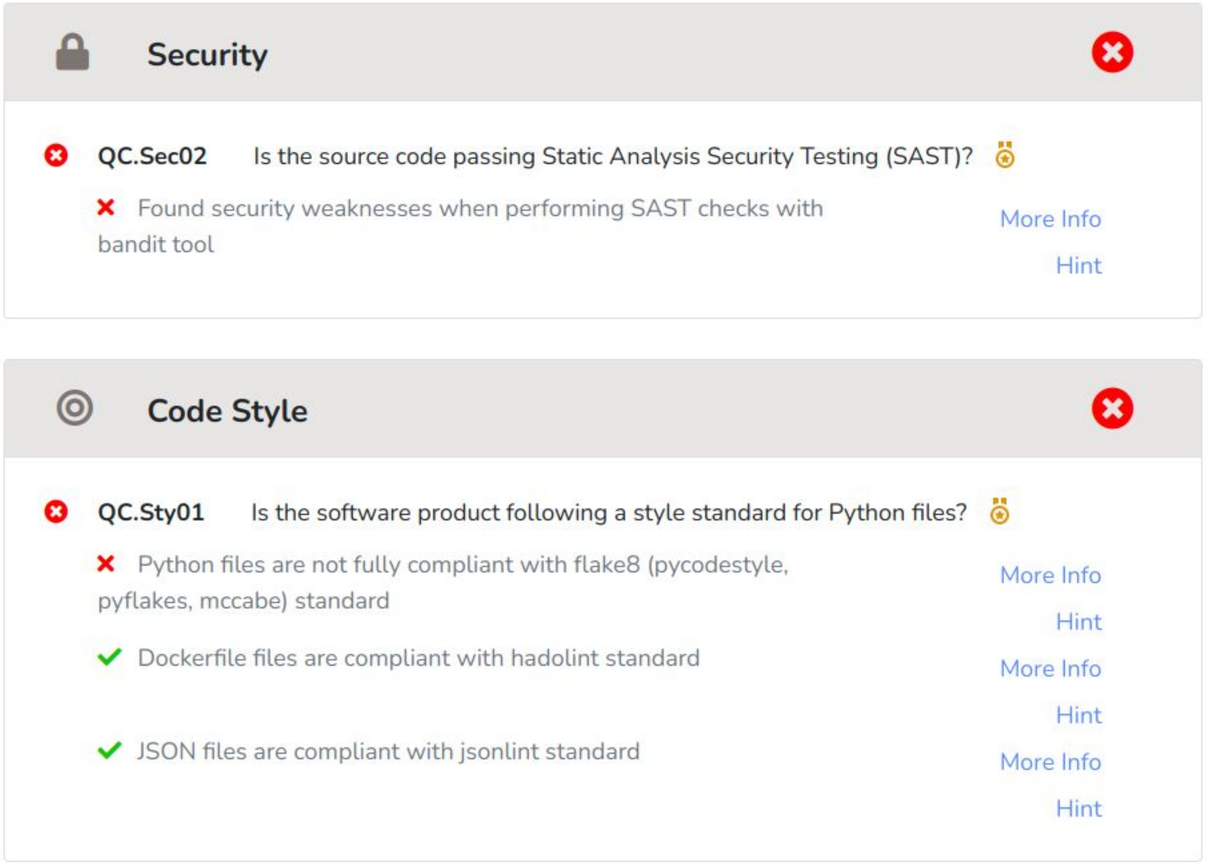

ibergrid-perf Public
forked from EOSC-synergy/eosc-perf
Review SQAaaS report
SQAaaS
Jenkins
If badge = gold accept PR
Git clone workspace or use Github UI
Create a new branch if any improvement required
Push new branch to remote and create a Github Pull Request (PR)
Commit the QC.Sty and QC.Sec corrections
Create a new branch to apply changes

2.2 Review QC.Sty and QC.Sec report

Bandit tool found an issue:
- **filename**: "./scripts/sandbox.py"
- **issue_text**: "Requests call with verify=False disabling SSL certificate checks, security issue."

Flake8 tool found issues:
- ./scripts/sandbox.py:9:80: **E501 line too long**
- ./scripts/sandbox.py:27:98: **E741 ambiguous variable name 'l'**
- ./scripts/sandbox.py:47:5: **F841 local variable 'user' is assigned to but never used**

Here we will ignore this issues within flake8


Security

✗ **QC.Sec02** Is the source code passing Static Analysis Security Testing (SAST)?

✗ Found security weaknesses when performing SAST checks with bandit tool

More Info
Hint

Code Style

✗ **QC.Sty01** Is the software product following a style standard for Python files?

✗ Python files are not fully compliant with flake8 (pycodestyle, pyflakes, mccabe) standard

More Info
Hint

✓ Dockerfile files are compliant with hadolint standard

More Info
Hint

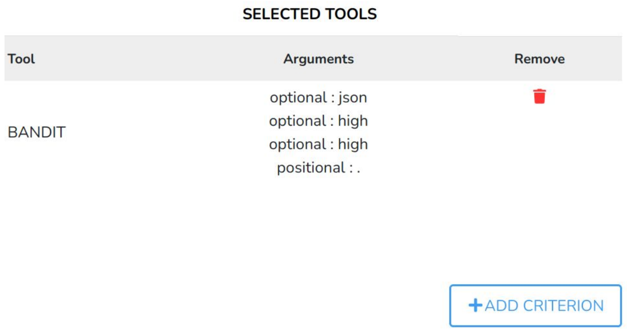
✓ JSON files are compliant with jsonlint standard

More Info
Hint

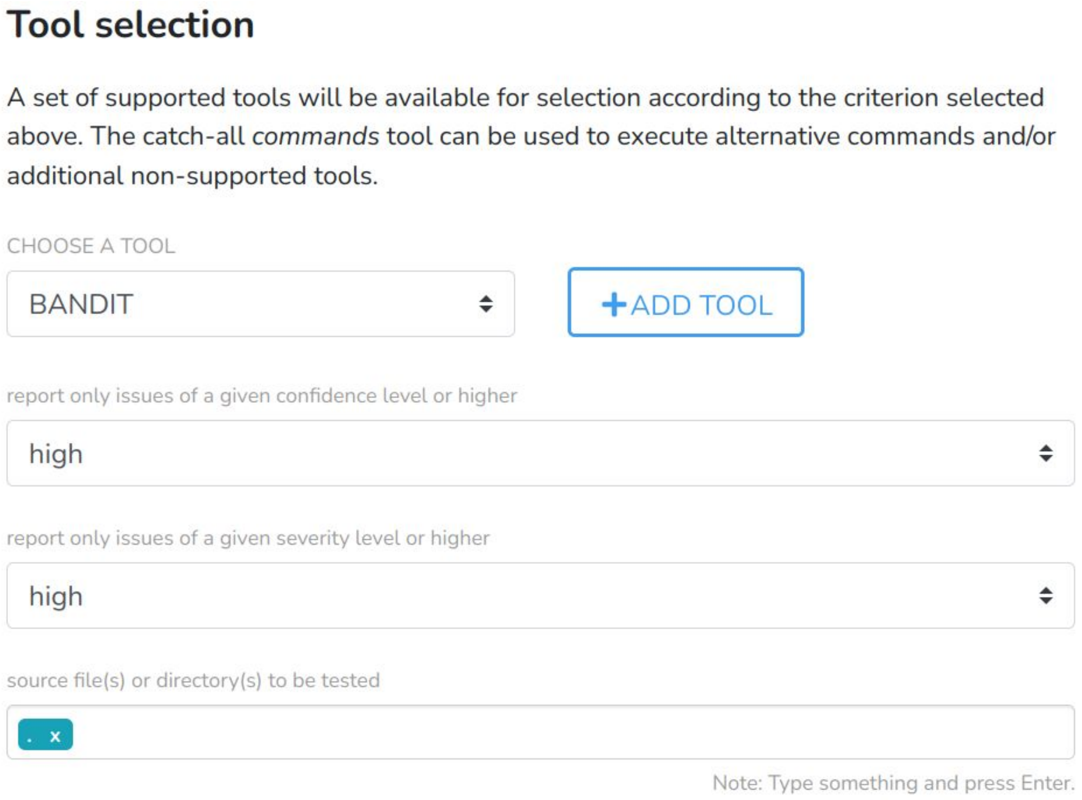
2.3 Create a pipeline configuration to support the development

QC.Sec:

- Add Bandit tool with same options as applied in the assessment test


SELECTED TOOLS

Tool	Arguments	Remove
BANDIT	optional : json optional : high optional : high positional : .	🗑


Tool selection

A set of supported tools will be available for selection according to the criterion selected above. The catch-all *commands* tool can be used to execute alternative commands and/or additional non-supported tools.

CHOOSE A TOOL: BANDIT

report only issues of a given confidence level or higher: high

report only issues of a given severity level or higher: high

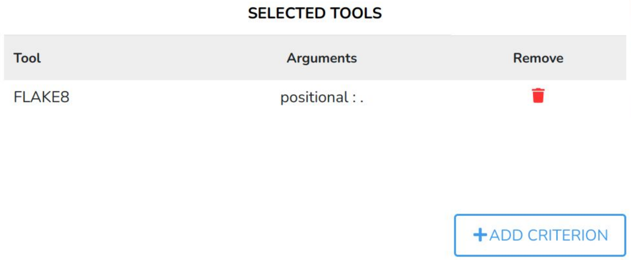
source file(s) or directory(s) to be tested: . x

Note: Type something and press Enter.

2.3 Create a pipeline configuration to support the development

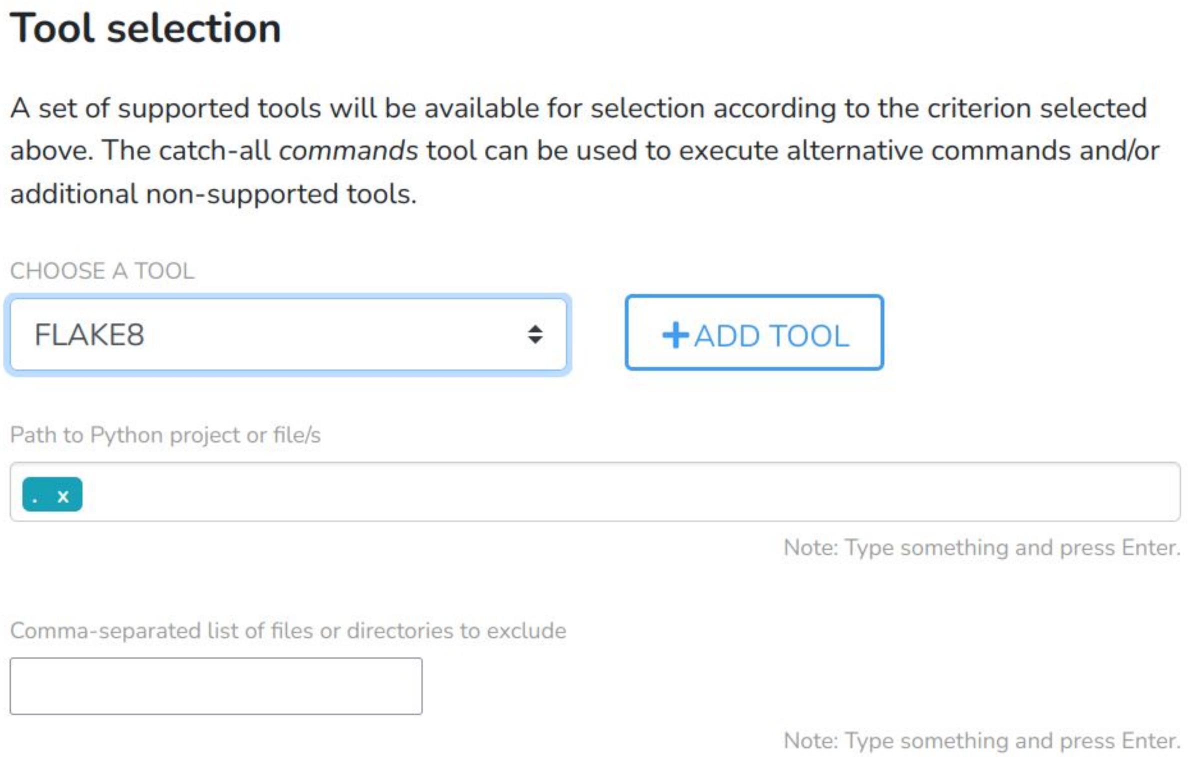
QC.Sty:

- Add flake8 tool with same options as applied in the assessment test



SELECTED TOOLS

Tool	Arguments	Remove
FLAKE8	positional : .	🗑



Tool selection

A set of supported tools will be available for selection according to the criterion selected above. The catch-all *commands* tool can be used to execute alternative commands and/or additional non-supported tools.

CHOOSE A TOOL: FLAKE8

Path to Python project or file/s: . x

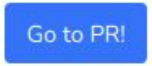
Note: Type something and press Enter.

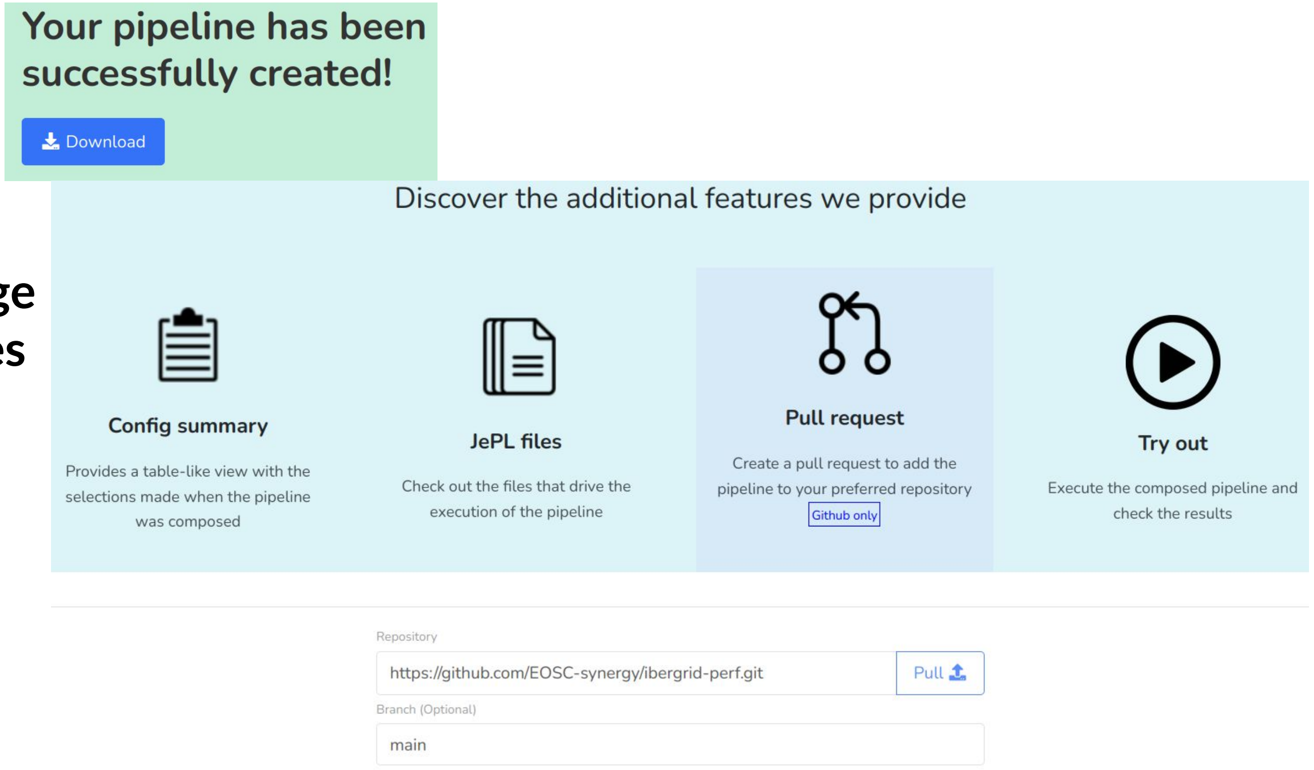
Comma-separated list of files or directories to exclude:

Note: Type something and press Enter.

2.3 **Create a pipeline configuration to support the development**

Download pipeline configuration files
or
Create a Github Pull Request (PR) to merge the configuration files




Your pipeline has been successfully created!

Discover the additional features we provide

Config summary
Provides a table-like view with the selections made when the pipeline was composed

JePL files
Check out the files that drive the execution of the pipeline

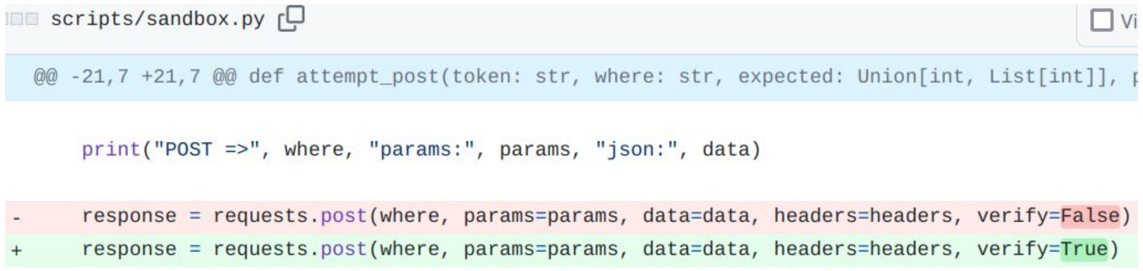
Pull request
Create a pull request to add the pipeline to your preferred repository
Github only

Try out
Execute the composed pipeline and check the results

Repository
https://github.com/EOSC-synergy/ibergrid-perf.git

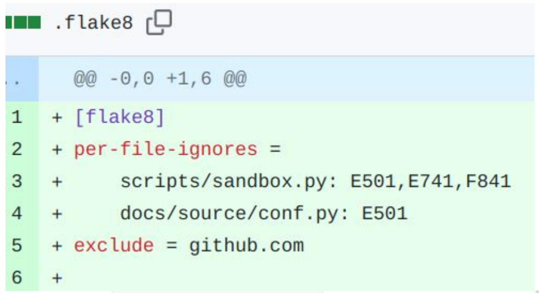
Branch (Optional)
main

2.4 Merge the results into the production branch



scripts/sandbox.py

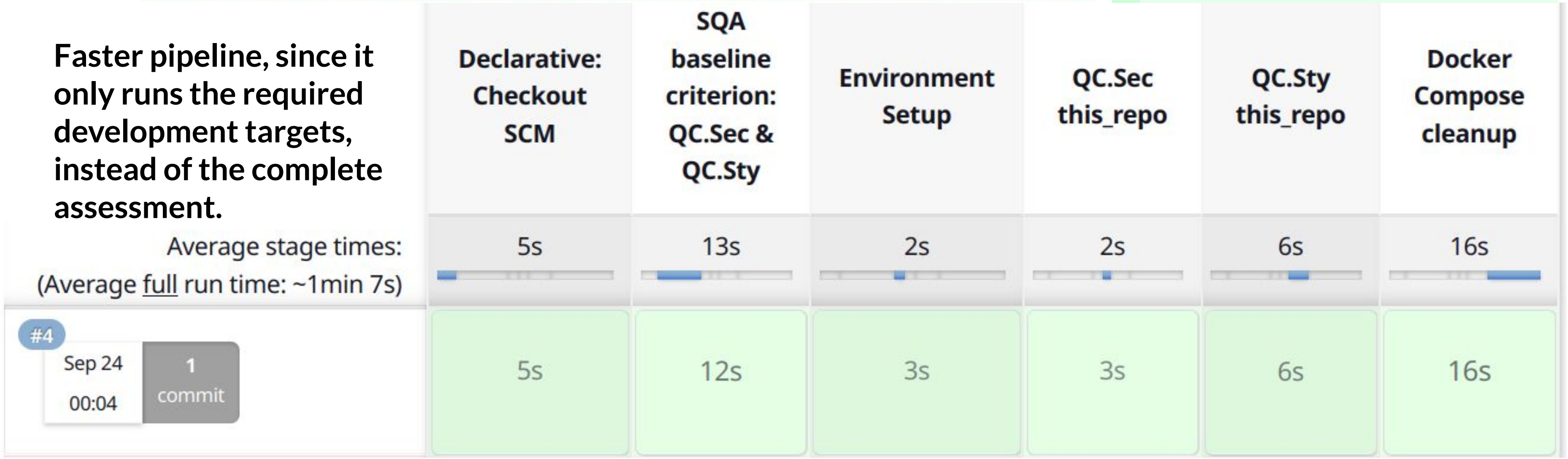
```diff
@@ -21,7 +21,7 @@ def attempt_post(token: str, where: str, expected: Union[int, List[int]], p

      print("POST =>", where, "params:", params, "json:", data)

-     response = requests.post(where, params=params, data=data, headers=headers, verify=False)
+     response = requests.post(where, params=params, data=data, headers=headers, verify=True)
```



.flake8

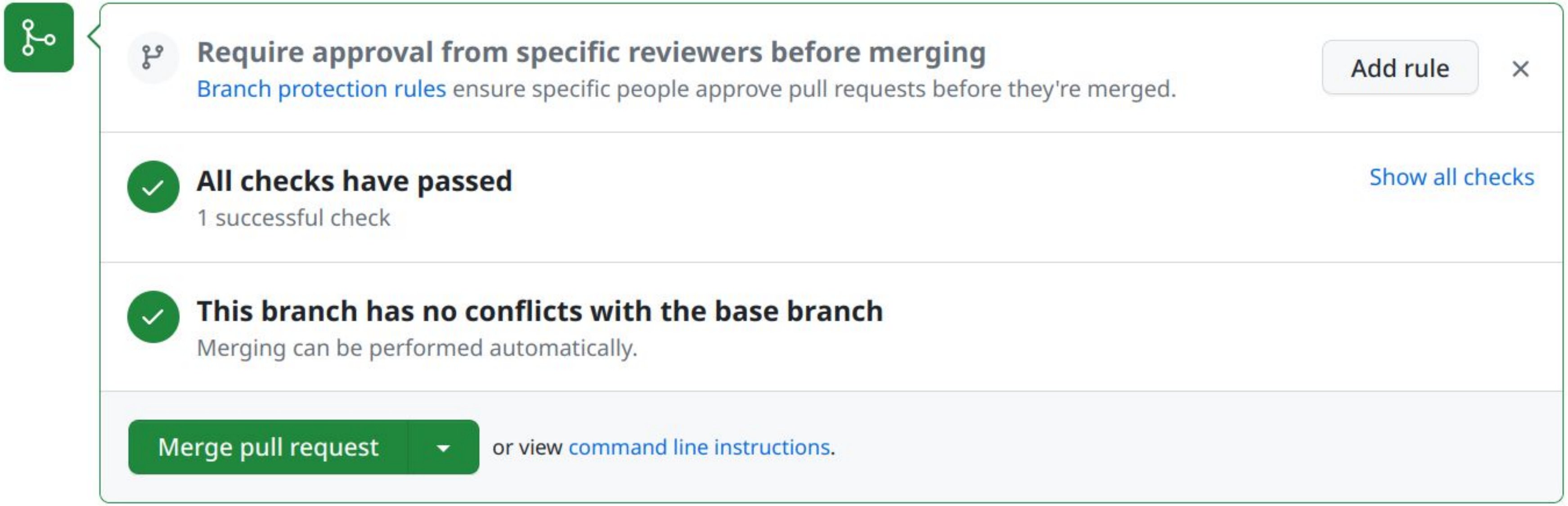
```diff
@@ -0,0 +1,6 @@
+ [flake8]
+ per-file-ignores =
+     scripts/sandbox.py: E501,E741,F841
+     docs/source/conf.py: E501
+ exclude = github.com
+
```

Faster pipeline, since it only runs the required development targets, instead of the complete assessment.



	Declarative: Checkout SCM	SQA baseline criterion: QC.Sec & QC.Sty	Environment Setup	QC.Sec this_repo	QC.Sty this_repo	Docker Compose cleanup
Average stage times: (Average full run time: ~1min 7s)	5s	13s	2s	2s	6s	16s
#4 Sep 24 00:04 1 commit	5s	12s	3s	3s	6s	16s

2.4 Merge the results into the production branch


Require approval from specific reviewers before merging

Branch protection rules ensure specific people approve pull requests before they're merged.

All checks have passed

1 successful check

This branch has no conflicts with the base branch

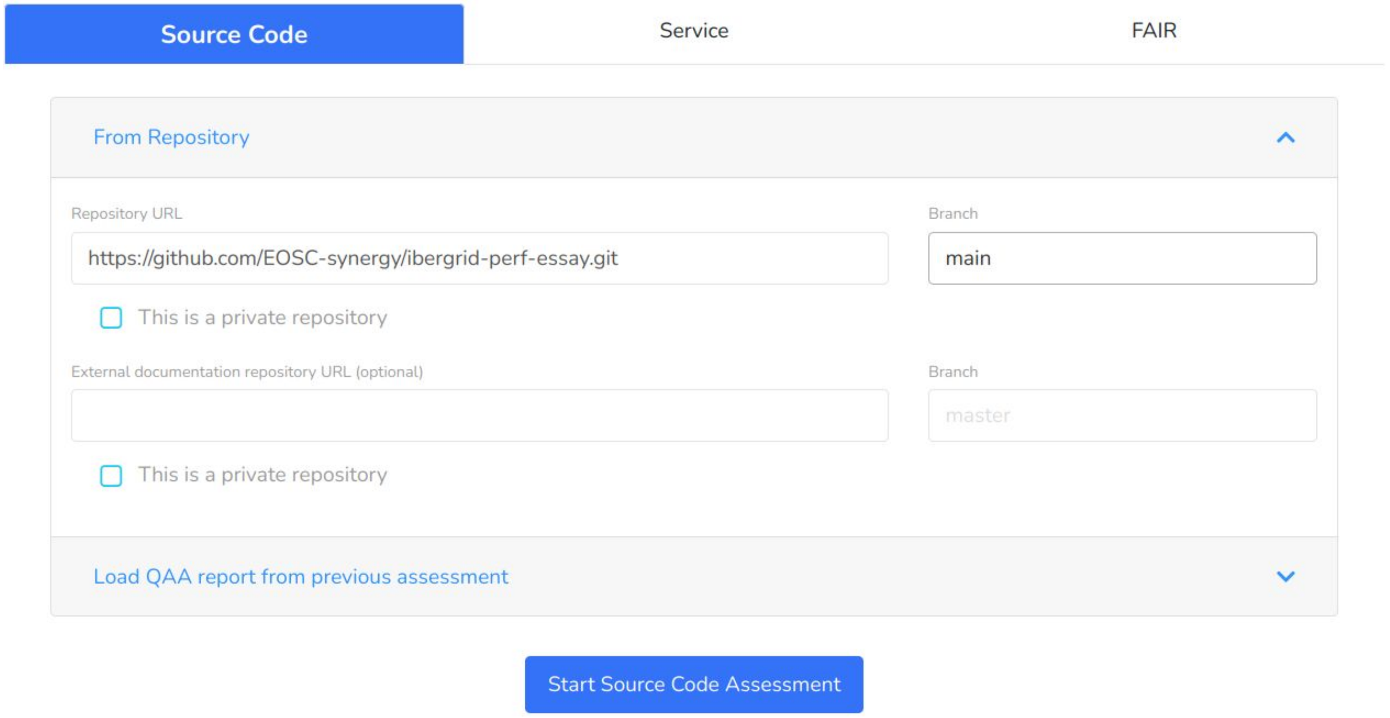
Merging can be performed automatically.

or view command line instructions.

2.5

Run the SQAaaS assessment and check the gold badge

Let’s trigger QA assessment on the SQAaaS platform


Source Code
Service
FAIR
From Repository
Load QAA report from previous assessment
Start Source Code Assessment

Repository URL: https://github.com/EOSC-synergy/ibergrid-perf-essay.git

Branch: main

This is a private repository

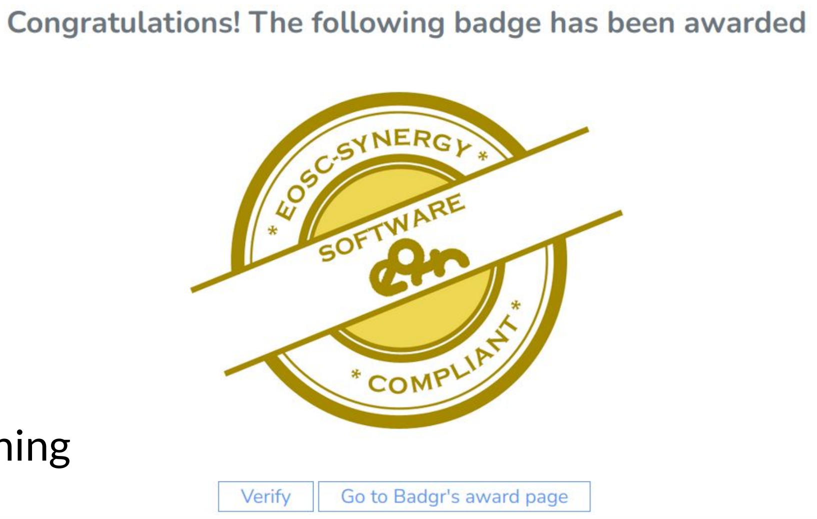
External documentation repository URL (optional):

Branch: master

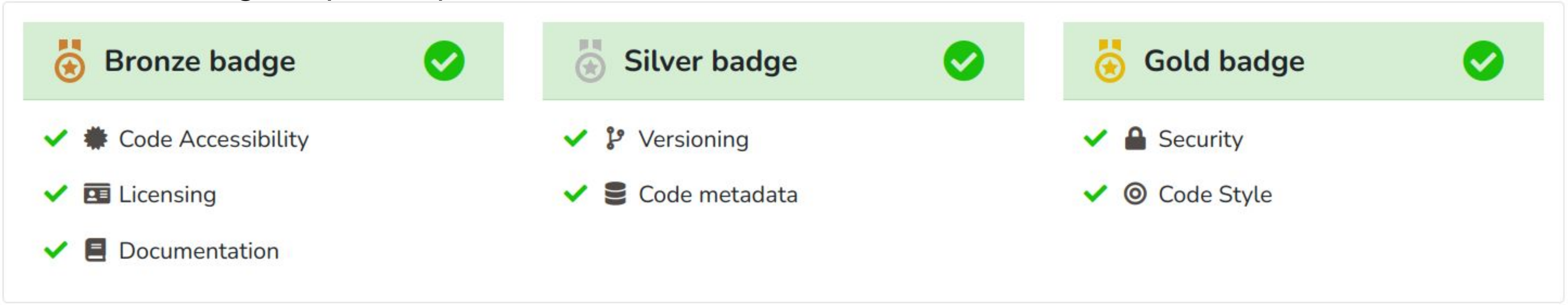
This is a private repository

2.5 Run the SQAaaS assessment and check the gold badge

- The SQAaaS assessment validated the software and updated the badge.
- The SQAaaS pipeline can be used in parallel to verify code quality with available tools following the git flow development.
- Verification and validation before pushing to target repository or branch.

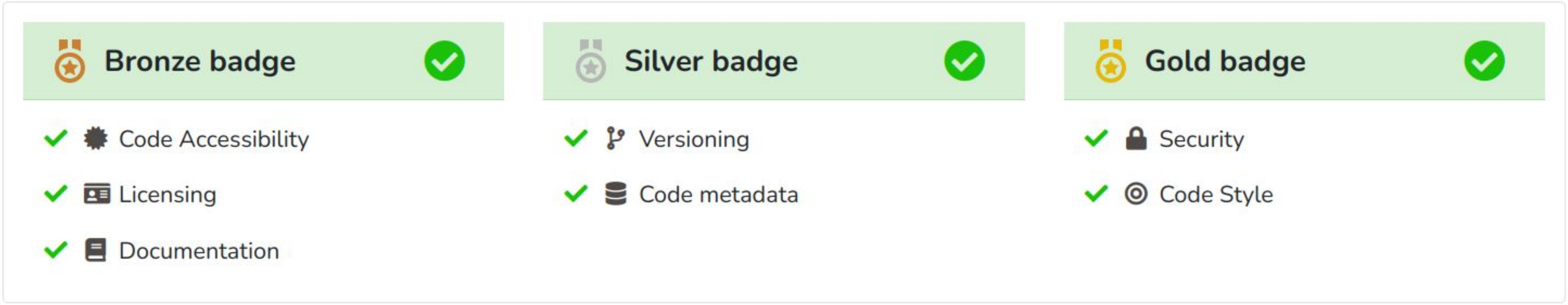


Congratulations! The following badge has been awarded



Bronze badge ✓	Silver badge ✓	Gold badge ✓
✓ Code Accessibility	✓ Versioning	✓ Security
✓ Licensing	✓ Code metadata	✓ Code Style
✓ Documentation		

Pipeline as a Service



Bronze badge ✓

- ✓ Code Accessibility
- ✓ Licensing
- ✓ Documentation

Silver badge ✓

- ✓ Versioning
- ✓ Code metadata

Gold badge ✓

- ✓ Security
- ✓ Code Style

What is JePL?

JePL is the core component of **SQAaaS**.

JePL translates the quality criteria into Jenkins pipelines using a shared library:

- Human readable YAML format configuration files to describe the software testing tools configuration and the composers that delivers the required execution environments for the supported platforms
- Dynamic pipelines generator over an automation system (Jenkins open source automation server)

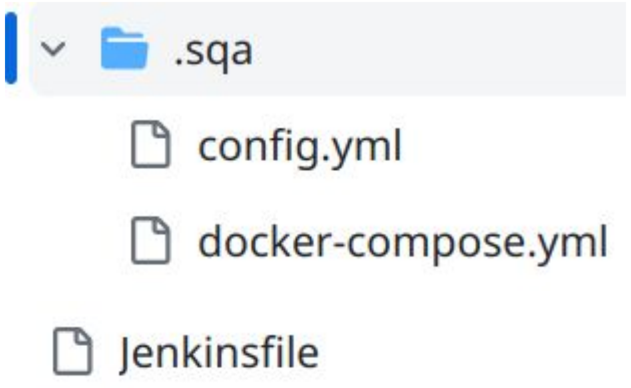
Demos

01 Improve your software with SQAaaS assessment

02 SQAaaS integration with git flow development process

03 **JePL: improve your CI pipeline**

03 JePL: improve your CI pipeline


.sqa
config.yml
docker-compose.yml
Jenkinsfile

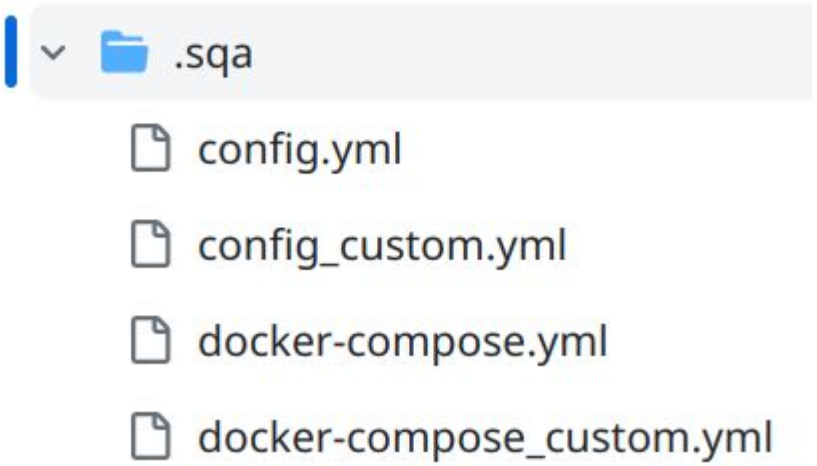
Steps

3.1 Add custom JePL configuration files

3.2 Edit Jenkinsfile to add custom configs

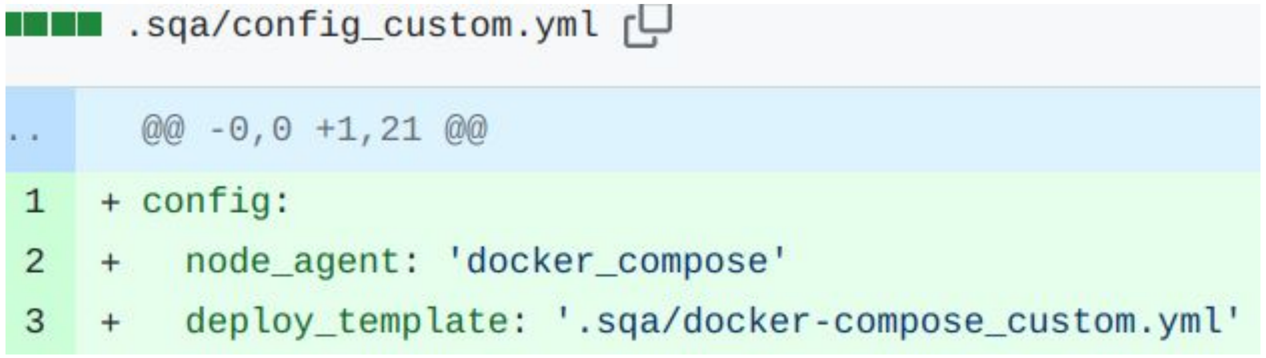
3.3 Pipeline automation with git flow operations

3.1 Add custom JePL configuration files



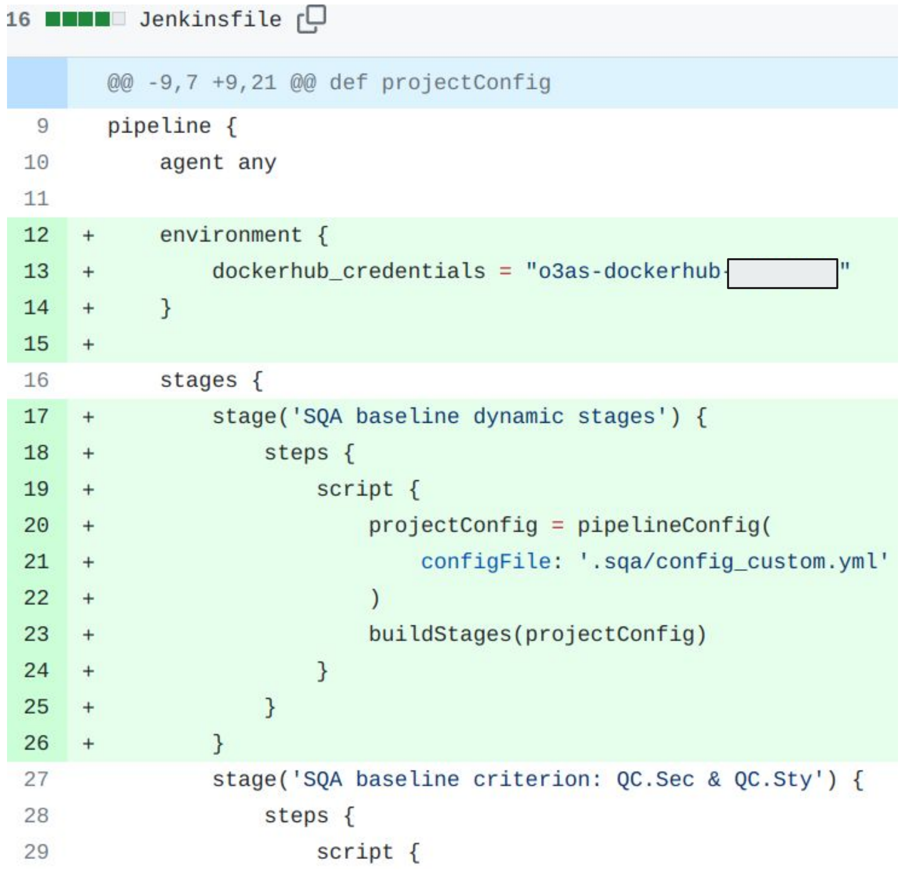
```
.sqa
    config.yml
    config_custom.yml
    docker-compose.yml
    docker-compose_custom.yml
```

- Get config.yml and docker-compose.yml from the initial repository state.
- Add configuration files with their name suffixed with “_custom”.
- Set deploy_template at config_custom.yml.



```
.sqa/config_custom.yml

@@ -0,0 +1,21 @@
+ config:
+   node_agent: 'docker_compose'
+   deploy_template: '.sqa/docker-compose_custom.yml'
```

3.2 Edit Jenkinsfile to add custom configs

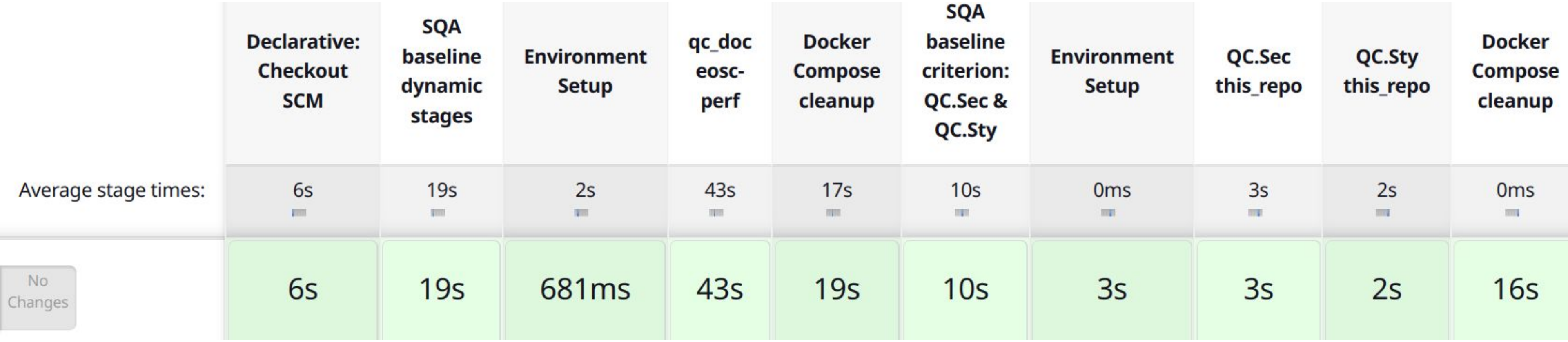


```
16 Jenkinsfile

@@ -9,7 +9,21 @@ def projectConfig
 pipeline {
     agent any

+    environment {
+        dockerhub_credentials = "o3as-dockerhub-[redacted]"
+    }
+
     stages {
+        stage('SQA baseline dynamic stages') {
+            steps {
+                script {
+                    projectConfig = pipelineConfig(
+                        configFile: '.sqa/config_custom.yml'
+                    )
+                    buildStages(projectConfig)
+                }
+            }
+        }
         stage('SQA baseline criterion: QC.Sec & QC.Sty') {
             steps {
                 script {
```

- Add a new stage with the JePL configuration file
- Also added an environment variable Jenkins side to pull private docker images

3.3 Pipeline automation with git flow operations

- Pipeline is automatically triggered with a new push into PR branch
- Processed pipeline with stages from both JePL configurations.
- If first custom configuration fails, the second will not run, sparing resources.



	Declarative: Checkout SCM	SQA baseline dynamic stages	Environment Setup	qc_doc eosc-perf	Docker Compose cleanup	SQA baseline criterion: QC.Sec & QC.Sty	Environment Setup	QC.Sec this_repo	QC.Sty this_repo	Docker Compose cleanup
Average stage times:	6s	19s	2s	43s	17s	10s	0ms	3s	2s	0ms
No Changes	6s	19s	681ms	43s	19s	10s	3s	3s	2s	16s

Takeaways

- SQAaaS fulfills a *twofold objective* wrt your digital objects:
    a. Take credit of quality achievements through metadata-empowered digital badges
    b. Build CI/CD environments for preserving quality attributes within digital object's lifecycle (development, staging & production), implemented through *git flow* model

- Next-up on SQAaaS:
    a. Evolving towards covering **workflow-based DT validation** (custom assessments)
    b. New **CLI interface**
    c. **On-premises automated deployment** (privacy, close-to-data assessment)

Thank you for your attention

Q&A time

Pablo Orviz
orviz@ifca.unican.es
IFCA-CSIC

Samuel Bernardo
samuel@lip.pt
LIP