



Mastering the SQAaaS platform: a Software Quality Assurance as a Service tutorial

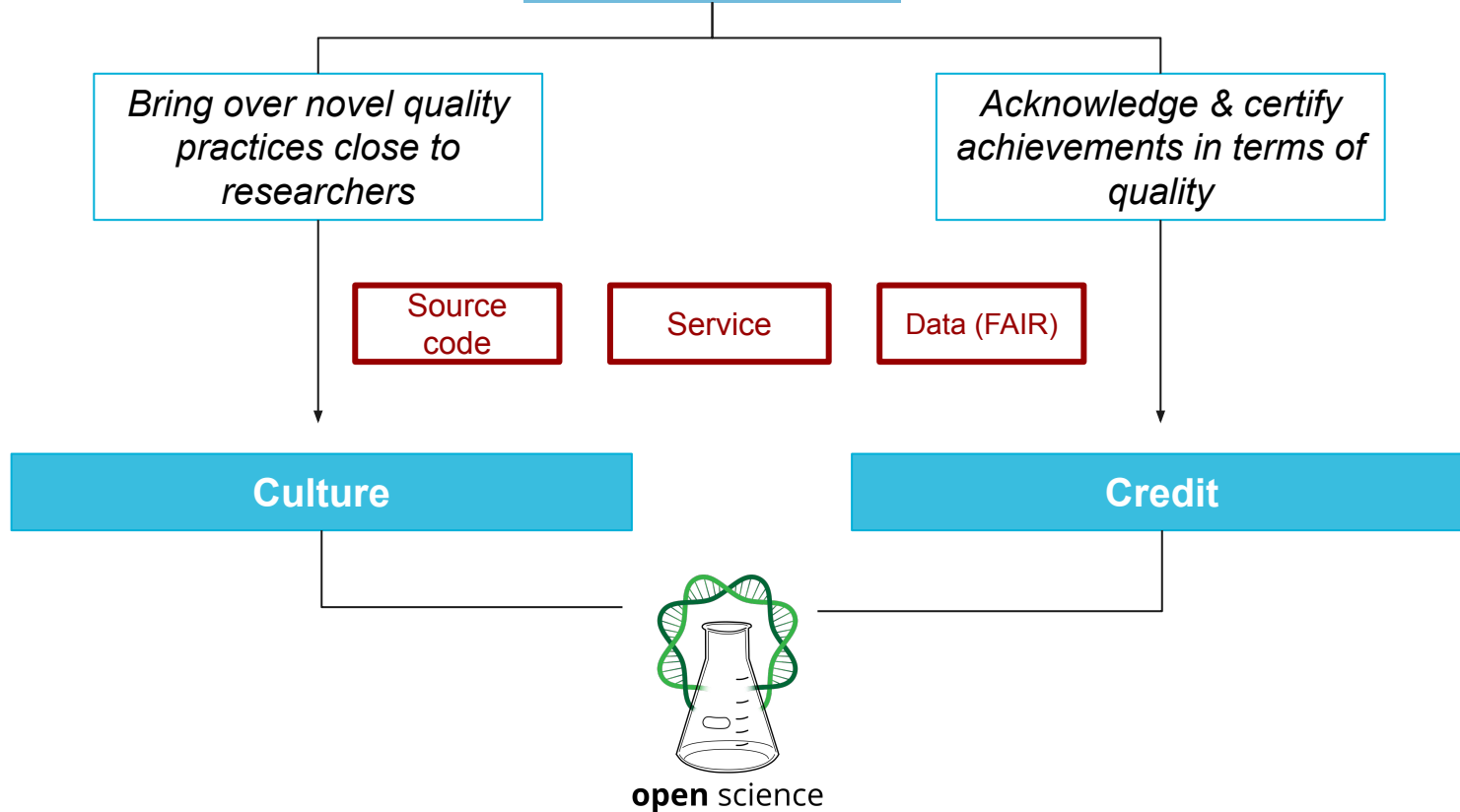
Pablo Orviz

orviz@ifca.unican.es
IFCA-CSIC

Samuel Bernardo

samuel@lip.pt
LIP

The SQAaaS platform



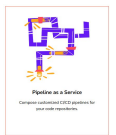
The SQAaaS platform



Bring over novel quality practices close to researchers

Acknowledge & certify achievements in terms of quality

Pipeline as a Service



DevOps (CI/CD)

Agile

Pipelines or Workflows

Source code

Service

Data (FAIR)

Open Badges

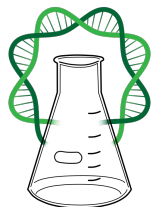
Digital badges

Quality Assessment and Awarding



Culture

Credit



open science

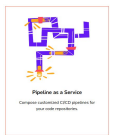
The SQAaaS platform



Bring over novel quality practices close to researchers

Acknowledge & certify achievements in terms of quality

Pipeline as a Service



DevOps (CI/CD)

Agile

Source code

Service

Data (FAIR)

QA criteria (standards)

Open Badges

Pipelines or Workflows

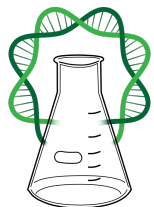
Digital badges

Quality Assessment and Awarding



Culture

Credit



open science

The SQAaaS platform: the Quality Assessment and Awarding

For this tutorial we will focus solely on the **Quality Assessment and Awarding** (SQAaaS building block) of **source code** (research object)



Quality Assessment & Awarding

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

Motivation

Stimulates the adoption of good practices to **ease the adoption and sustainability** of the source code

Delivers

Assessment data, including QA reports and build logs for the traceability of development life cycle.

Digital badges that contribute to reputation-building and crediting.

The SQAaaS platform: QA criteria & digital badges

	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://docs.sqaaaS.eosc-synergy.eu>

The QA criteria...

- Have different levels of criticality
 - *MUST/SHOULD/MAY* (software & services)
 - *Essential/Important/Useful* (FAIR)
- **Not all** of them can be **automatically evaluated**
- **Some require user feedback** for a successful automated evaluation

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

 [10.20350/digitalCSIC/12543](https://doi.org/10.20350/digitalCSIC/12543)

The SQAaaS platform: reporting & awarding



[Verify](#)

[Go to Badgr's award page](#)

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

[Get Badge Image](#)

[Get Badge Shield](#)

Code Accessibility		✓
✓ QC.Acc01	Is the source code managed with a Version Control System?	
✓	Source code uses Git for version control	i

Documentation		✓
✓ QC.Doc06.1	Is the software scope outlined in the code repository?	
✓	A README file is present in the code repository	i
✓ QC.Doc06.3	Does the project establish a code of conduct for its participants?	i
	contribute to the code?	i
	code?	i
	markup language?	i
✗	Docs are not fully compliant with markdownlint standard	i

Bronze badge



- ✓ Licensing
- ✓ Documentation
- ✓ Code Accessibility

Silver badge



- ✓ Versioning
- ✓ Code metadata

Gold badge












- ✓ Security
- ✗ Unit Testing
- ✗ Code Style

[Check complete SQAaaS report](#)

The SQAaaS platform: 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 	checkstyle 	stylelint 	jsonlint 	hadolint 
Security (QC.Sec)	bandit 	gosec 					

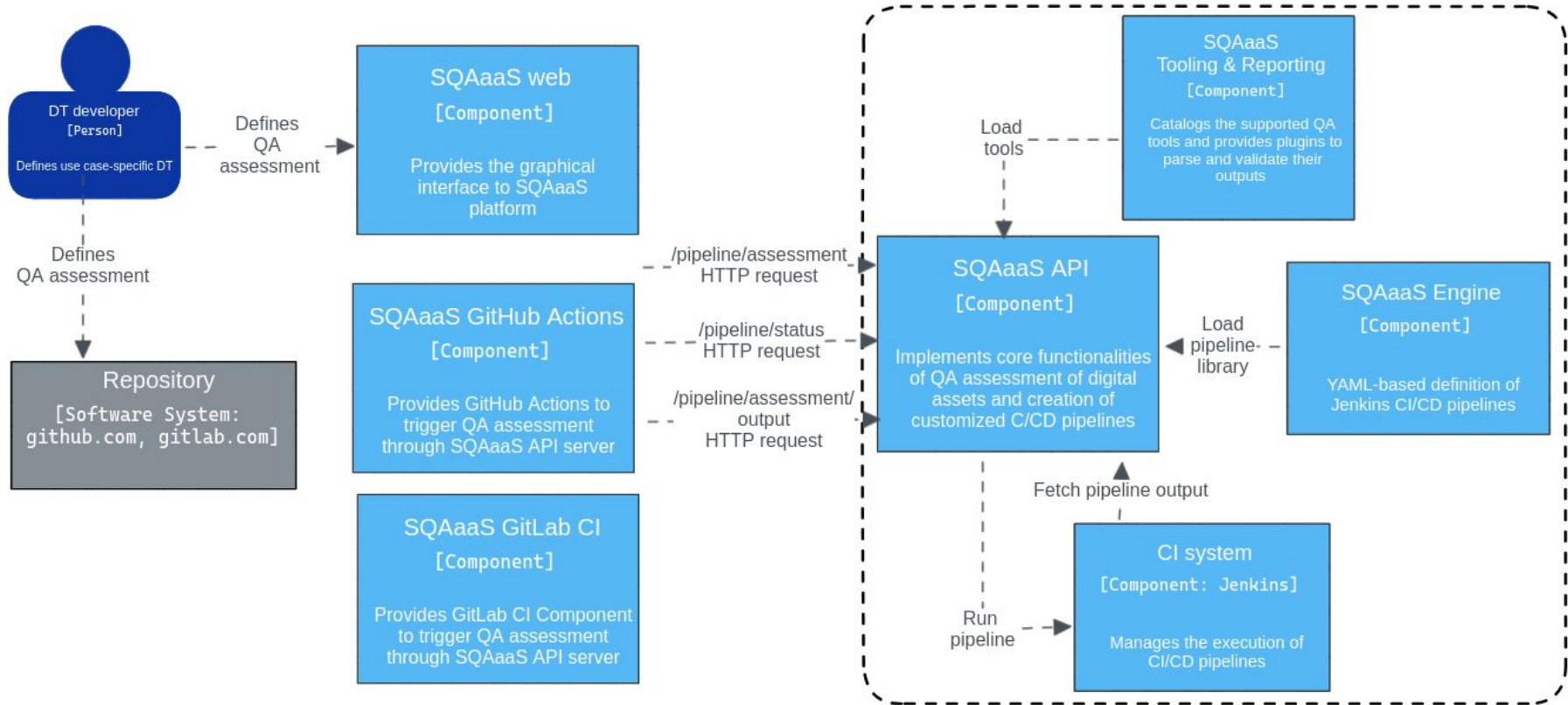
Documentation
(QC.Doc)

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

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, arguments, QA criterion, etc.
3. (optional) Create a plugin to parse the output of the tool

The SQAaaS platform: architecture & integrations





Demo: leveraging SQAaaS GitHub actions

SQAaaS demo: **our use case**

Lattice QCD use case from InterTwin project

https://github.com/jkomijani/normflow_/



The screenshot shows the GitHub repository page for 'normflow_'. The repository is public and has a 'Public' badge. It is categorized as 'sqaas software' with a 'silver' badge and is licensed under 'MIT'. The 'About' section describes the project as 'Normalizing flow for generating lattice field configurations'. The 'Languages' section shows a bar chart with Python at 100.0%.

About

Normalizing flow for generating lattice field configurations

Languages

- Python 100.0%

We'll use **SQAaaS GitHub Actions** to run a comprehensive QA assessment upon a new release of the software

SQAaaS demo: **about GitHub Actions**

- Automate the execution of software workflows to build, test & deploy code
- Uses YAML-based descriptions placed under `.github/workflows`
 - Most common properties: `name`, `on` (when to run the workflow), `jobs` (the actual work)
 - `jobs` property contains the workflow steps, which can be user-defined commands (`run`) or actions reused (`uses`)
 - Loads of possible actions to be reused: [GitHub Actions marketplace](#)
- More on [GitHub Actions docs](#)

```
YAML
```

```
name: GitHub Actions Demo
run-name: ${{ github.actor }} is testing out GitHub Actions
on: [push]
jobs:
  Explore-GitHub-Actions:
    runs-on: ubuntu-latest
    steps:
      - run: echo "👋 The job was automatically triggered by a ${{
github.event_name }} event."
      - run: echo "🌍 This job is now running on a ${{ runner.os }} server hosted
by GitHub!"
      - run: echo "📄 The name of your branch is ${{ github.ref }} and your
repository is ${{ github.repository }}."
      - name: Check out repository code
        uses: actions/checkout@v4
      - run: echo "📄 The ${{ github.repository }} repository has been cloned to
the runner."
      - run: echo "🗄 The workflow is now ready to test your code on the runner."
      - name: List files in the repository
        run: |
          ls ${{ github.workspace }}
      - run: echo "🎉 This job's status is ${{ job.status }}."
```

The screenshot displays the GitHub Actions interface. On the left, the 'Actions' tab is active, showing a list of workflows under the repository `.github/workflows/greet-everyone.yml`. The workflow `GitHub Actions Demo` is highlighted with a red box. On the right, the 'All workflows' tab shows 88 workflow runs. A specific run titled 'octocat is testing out GitHub Actions' is selected, showing it was triggered by 'octocat' on the 'octocat-patch-1' branch. Below this, a detailed log of the workflow run is shown, listing 10 steps that all completed successfully.

Step	Duration
Run echo "👋 This job is now running on a Linux server hosted by GitHub!"	0s
Run echo "🌍 The name of your branch is refs/heads/octocat-patch-1 and your repository is oct..."	0s
Check out repository code	1s
Run echo "📄 The octo-org/octo-repo repository has been cloned to the runner."	0s
Run echo "📄 The workflow is now ready to test your code on the runner."	0s
List files in the repository	0s
Run echo "🎉 This job's status is success."	0s
Post Check out repository code	0s
Complete job	0s

SQAaaS demo: **the SQAaaS GitHub Action/s**

Why this integration?

- GitHub Actions widespread use/popular technology for CI work
- To trigger SQAaaS assessment building block as certification tool (report, badges)
 - In response to events such as releases, pull requests, regular pushes..

How could one trigger SQAaaS from GitHub? The [sqaaas-assessment-action](#)



sqaaas-assessment-action Public

GitHub action to trigger QA assessments in SQAaaS platform

Python ☆ 0 GPL-3.0 0 2 1 Updated on Aug 14

This is the main SQAaaS GH action: triggers the default quality assessment of code in the SQAaaS platform

- Includes the criteria that can be checked without user's feedback

```
uses: eosc-synergy/sqaaas-assessment-action@v2
```

```
uses: eosc-synergy/sqaaas-assessment-action@v2
with:
  repo: 'https://github.com/eosc-synergy/sqaaas-assessment-action'
  branch: 'main'
```

SQAaaS demo: **the SQAaaS GitHub Action/s**

Why this integration?

- GitHub Actions widespread use/popular technology for CI work
- To trigger SQAaaS assessment building block as certification tool (report, badges)
 - In response to events such as releases, pull requests, regular pushes..

How could one trigger SQAaaS from GitHub? The [sqaas-assessment-action](#)



The screenshot shows the GitHub repository page for 'sqaas-assessment-action'. The repository is public and is a Python-based action. It has 0 stars, 0 forks, 0 issues, 2 commits, and 1 pull request. It was last updated on August 14. The description reads: 'GitHub action to trigger QA assessments in SQAaaS platform'. There is a green waveform icon in the top right corner of the repository card.

However, some other criteria relate to checks that implies user's feedback...

- Frequent requirement in unit/functional/integration testing



The [sqaas-step-action](#) permits the definition of the work to be done (only for customizable criteria)

The screenshot shows the GitHub repository page for 'sqaas-step-action'. The repository is public and is a Python-based action. It has 0 stars, 0 forks, 0 issues, 1 commit, and 1 pull request. It was last updated on March 21. The description reads: 'GitHub action to define the work to be done within a criterion step'. There is a green waveform icon in the top right corner of the repository card.

SQAaaS demo: **the SQAaaS GitHub Action/s**

Why this integration?

- GitHub Actions widespread use/popular technology for CI work
- To trigger SQAaaS assessment building block as certification tool (report, badges)
 - In response to events such as releases, pull requests, regular pushes..

How could one trigger SQAaaS from GitHub? The [sqaas-assessment-action](#)

[sqaas-step-action](#) Public

GitHub action to define the work to be done within a criterion step

Python ☆ 0 GPL-3.0 0 1 1 Updated 2 minutes ago

```
uses: eosc-synergy/sqaas-step-action@v1
```

```
with:
```

```
  name: pytest-step
```

```
  container: myownpytestimage:latest
```

```
  tool: pytest
```

```
  test-path: ./tests
```

SQAaaS demo: **the SQAaaS GitHub Action/s**

Why this integration?

- GitHub Actions widespread use/popular technology for CI work
- To trigger SQAaaS assessment building block as certification tool (report, badges)
 - In response to events such as releases, pull requests, regular pushes..

How could one trigger SQAaaS from GitHub? The [sqaas-assessment-action](#)

[sqaas-step-action](#) Public

GitHub action to define the work to be done within a criterion step

Python 0 0 GPL-3.0 0 1 1 Updated 2 minutes ago

```
uses: eosc-synergy/sqaas-step-action@v1
with:
  name: pytest-step
  container: myownpytestimage:latest
  tool: pytest
  test-path: ./tests
```

[sqaas-assessment-action](#) Public

GitHub action to trigger QA assessments in SQAaaS platform

Python 0 0 GPL-3.0 0 2 1 Updated on Aug 14

```
uses: eosc-synergy/sqaas-assessment-action@v2
with:
  qc_uni_steps: pytest-step
```


SQAaaS demo: **the SQAaaS GitHub Action/s**

Putting the requirements all together..

We plan to trigger a SQAaaS certification for our use case `normflow_` that:

- #1. Use GitHub Actions solution:** *the repository is maintained through GitHub and already uses GH actions*
- #2. Extend the regular assessment with the unit testing check (pytest):** *opt for higher scores in terms of SQAaaS certification (up to golden badge)*

For the demo we will use a fork of the `normflow_` repository located under:

https://github.com/EOSC-Synergy-SQAaaS/normflow_ (branch **tests**)

SQAaaS demo: **the SQAaaS GitHub Action/s**

https://github.com/EOSC-Synergy-SQAaaS/normflow_ (branch **tests**)

- #1. Use GitHub Actions solution:** *the repository is maintained through GitHub and already uses GH actions*

[normflow_ / .github / workflows / sqaaas.yml](#) 

 jkomijani sqaaas.yml updated

Code Blame 11 lines (9 loc) · 231 Bytes

```
1   name: SQAaaS-Test
2
3   on: [push]
4
5   jobs:
6     sqaaas_job:
7       runs-on: ubuntu-latest
8       name: Job that triggers SQAaaS platform
9       steps:
10      - name: SQAaaS assessment step
11        uses: eosk-synergy/sqaaas-assessment-action@v2
```

SQAaaS demo: **the SQAaaS GitHub Action/s**

https://github.com/EOSC-Synergy-SQAaaS/normflow_ (branch **tests**)

#2. Extend the QA levels with unit testing check: *opt for higher scores in SQAaaS (up to golden badge)*

- Code already achieved SQAaaS (silver) badge  (click to report)
- Steps:
 1. Definition of **unit test execution (QC.Uni criterion)** ⇨ [sqaaaS-step-action](#)
 2. **Include the previous QC.Uni step** in the overall assessment ⇨ [sqaaaS-assessment-action](#)

SQAaaS demo: **the SQAaaS GitHub Action/s**

https://github.com/EOSC-Synergy-SQAaaS/normflow_ (branch **tests**)

#2. **Extend the QA levels with unit testing check:** *opt for higher scores in SQAaaS (up to golden badge)*

- Code already achieved SQAaaS (silver) badge  (click to report)
- Steps:
 1. Definition of **unit test execution (QC.Uni criterion)** ⇨ [sqaaaS-step-action](#)
 2. **Include the previous QC.Uni step in the overall assessment** ⇨ [sqaaaS-assessment-action](#)

↳ Proper environment with required dependencies to run the tests (pytest, pytorch + numpy)

```
uses: eosc-synergy/sqaaaS-step-action@v1
with:
  name: commands-step
  container: myownpytestimage:latest
  tool: commands
  commands: |
    echo "First command to run"
    echo "Second command to run"
```

SQAaaS demo: **the SQAaaS GitHub Action/s**

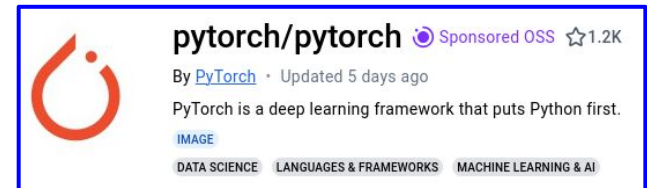
<https://github.com/EOSC-Synergy-SQAaaS/normflow> (branch **tests**)

#2. **Extend the QA levels with unit testing check:** *opt for higher scores in SQAaaS (up to golden badge)*

- Code already achieved SQAaaS (silver) badge  (click to report)
- Steps:
 1. Definition of **unit test execution (QC.Uni criterion)** ⇨ [sqaas-step-action](#)
 2. *Include the previous QC.Uni step in the overall assessment* ⇨ [sqaas-assessment-action](#)

↳ Proper environment with required dependencies to run the tests (pytest, pytorch + numpy)

```
uses: eosc-synergy/sqaas-step-action@v1
with:
  name: commands-step
  container: myownpytestimage:latest
  tool: commands
  commands: |
    echo "First command to run"
    echo "Second command to run"
```



*We'll use Docker Hub's
official pytorch image*

SQAaaS demo: the SQAaaS GitHub Action/s

<https://github.com/EOSC-Synergy-SQAaaS/normflow> (branch **tests**)

#2. Extend the QA levels with unit testing check: *opt for higher scores in SQAaaS (up to golden badge)*

- Code already achieved SQAaaS (silver) badge  (click to report)
- Steps:
 1. Definition of **unit test execution (QC.Uni criterion)** ⇨ [sqaaaS-step-action](#)
 2. *Include the previous QC.Uni step in the overall assessment* ⇨ [sqaaaS-assessment-action](#)

↳ Proper environment with required dependencies to run the tests (pytest, pytorch + numpy)

↳ We'll provide the set of commands with the [sqaaaS-step-action](#) (see [docs](#))

```
uses: eosc-synergy/sqaaaS-step-action@v1
with:
  name: commands-step
  container: myownpytestimage:latest
  tool: commands
  commands: |
    echo "First command to run"
    echo "Second command to run"
```

SQAaaS demo: the SQAaaS GitHub Action/s

https://github.com/EOSC-Synergy-SQAaaS/normflow_ (branch **tests**)

#2. Extend the QA levels with unit testing check: *opt for higher scores in SQAaaS (up to golden badge)*

- Code already achieved SQAaaS (silver) badge
- Steps:
 1. Definition of **unit test execution (QC.U)**
 2. *Include the previous QC.Uni step in*

↳ Modify existing job to add the step with the pytest definition under `.github/workflows/`

Note: the QC.Uni step **MUST** be declared within the same job ("sqaas_job") as the main SQAaaS assessment action, otherwise it will not be accessible

`normflow_ / .github / workflows / sqaas.yml`

Edit

Preview

```
1 name: SQAaaS-Test
2 on: [push]
3 jobs:
4   sqaas_job:
5     runs-on: ubuntu-latest
6     name: Job that triggers SQAaaS platform
7     steps:
8     - name: QC.Uni check
9       uses: eosc-synergy/sqaas-step-action@v1
10      with:
11        name: qc_uni_check
12        container: pytorch/pytorch
13        tool: commands
14        commands: |
15          pip install pytest
16          pytest -sv
17      - name: SQAaaS assessment step
18        uses: eosc-synergy/sqaas-assessment-action@v2
```

action

SQAaaS demo: the SQAaaS GitHub Action/s

https://github.com/EOSC-Synergy-SQAaaS/normflow_ (branch **tests**)

#2. Extend the QA levels with unit testing check: *opt for higher scores in SQAaaS (up to golden badge)*

- Code already achieved SQAaaS (silver) badge
- Steps:
 1. Definition of **unit test execution (QC.Uni)**
 2. *Include the previous QC.Uni step in*

↳ Add a new (job) step with the pytest definition in `.github/workflows/sqaas.yml`

Note: the QC.Uni step **MUST** be declared within the same job ("sqaas_job") as the main SQAaaS assessment action, otherwise it will not be accessible

Warning: Should we leave the GH workflow as it is now and the QC.Uni step would not be triggered by the sqaas-assessment-action

`normflow_ / .github / workflows / sqaas.yml`

Edit

Preview

```
1 name: SQAaaS-Test
2 on: [push]
3 jobs:
4   sqaas_job:
5     runs-on: ubuntu-latest
6     name: Job that triggers SQAaaS platform
7     steps:
8     - name: QC.Uni check
9       uses: eosco-synergy/sqaas-step-action@v1
10      with:
11        name: qc_uni_check
12        container: pytorch/pytorch
13        tool: commands
14        commands: |
15          pip install pytest
16          pytest -sv
17     - name: SQAaaS assessment step
18       uses: eosco-synergy/sqaas-assessment-action@v2
```

action

SQAaaS demo: **the SQAaaS GitHub Action/s**

<https://github.com/EOSC-Synergy-SQAaaS/normflow> (branch **tests**)

#2. Extend the QA levels with unit testing check: *opt for higher scores in SQAaaS (up to golden badge)*

- Code already achieved SQAaaS (silver) badge  (click to report)
- Steps:
 1. Definition of **unit test execution (QC.Uni criterion)** with **pytest** ⇨ [sqaaaS-step-action](#)
 2. **Include the previous QC.Uni step** in the overall assessment ⇨ [**sqaaaS-assessment-action**](#)

SQAaaS demo: the SQAaaS GitHub Action/s

https://github.com/EOSC-Synergy-SQAaaS/normflow_ (branch **tests**)

#2. Extend the QA levels with unit testing check: *opt for* `normflow_ / .github / workflows / sqaaas.yml`

- Code already achieved SQAaaS (silver) badge
- Steps:
 1. Definition of **unit test execution (QC.Uni)**
 2. **Include the previous QC.Uni step** in the c

↳ Explicitly provide the QC.Uni step as input for the main `sqaaas-assessment-action` ([docs](#))

Note: the QC.Uni step is accessible by using the value set in the **name** input `sqaaas-step-action`

```
1  name: SQAaaS-Test
2  on: [push]
3  jobs:
4    sqaaas_job:
5      runs-on: ubuntu-latest
6      name: Job that triggers SQAaaS platform
7      steps:
8        - name: QC.Uni check
9          uses: eossc-synergy/sqaaas-step-action@v1
10         with:
11           name: qc_uni_check
12           container: pytorch/pytorch
13           tool: commands
14           commands: |
15             pip install pytest
16             pytest -sv
17         - name: SQAaaS assessment step
18           uses: eossc-synergy/sqaaas-assessment-action@v2
19         with:
20           qc_uni_steps: qc_uni_check
```

SQAaaS demo: **the SQAaaS GitHub Action/s**

<https://github.com/EOSC-Synergy-SQAaaS/normflow> (branch **tests**)

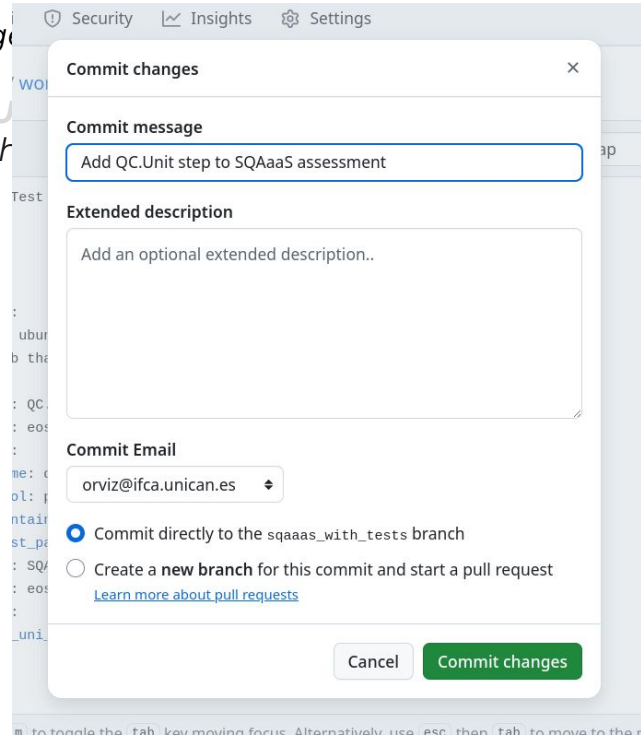
#2. **Extend the QA levels with unit testing check:** *opt for higher scores in SQAaaS (up to golden badge)*

- Code already achieved SQAaaS (silver) badge
- Steps:
 1. Definition of **unit test execution (QC.Uni)**
 2. **Include the previous QC.Uni step in the**

↳ Explicitly provide the QC.Uni step as input for the main `sqaas-assessment-action` ([docs](#))

Note: the QC.Uni step is accessible by using the value set in the **name** input `sqaas-step-action`

↳ We are all set ⇨ commit changes



SQAaaS demo: the SQAaaS GitHub Action/s

<https://github.com/EOSC-Synergy-SQAaaS/normflow> (branch **tests**)

#2. Extend the QA levels with unit testing check: *opt for higher scores in SQAaaS (up to golden badge)*

- Code already achieved SQAaaS (silver) badge  (click to report)
- Steps:
 1. Definition of **unit test execution (QC.Uni criterion)** with **pytest** ⇨ [sqaaaS-step-action](#)
 2. **Include the previous QC.Uni step** in the overall assessment ⇨ [sqaaaS-assessment-action](#)

↳ Explicitly provide the QC.Uni step as input for the main `sqaaaS-assessment-action` ([docs](#))

Note: the QC.Uni step is accessible by using the value set in the **name** input `sqaaaS-step-action`

↳ We are all set ⇨ commit changes

↳ Since the GH workflow runs for every push, it will be executed immediately

Triggered via push 3 minutes ago

Status

 orviz pushed ⇨ 127e670 `sqaaaS_with_tests`

In progress

sqaaaS.yml

on: push

 Job that triggers SQAa... 3m 28s

SQAaaS demo: the SQAaaS GitHub Action/s

<https://github.com/EOSC-Synergy-SQAaaS/normflow> (branch **tests**)

#2. Extend the QA levels with unit testing check: *opt for*

- Code already achieved SQAaaS (silver) badge
- Steps:
 1. Definition of **unit test execution** (QC.Uni c
 2. **Include the previous QC.Uni step** in the ov

↳ Explicitly provide the QC.Uni step as input for the main `sqaas-assessment-action` ([docs](#))

Note: the QC.Uni step is accessible by using the value set in the **name** input `sqaas-step-action`

↳ We are all set ⇨ commit changes


↳ Since the GH workflow runs for every push, it will be executed immediately



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: MIT	QC.Lic01	QC.Lic
✓	LICENSE file is visible at the root path of the code repository: LICENSE	QC.Lic01.1	QC.Lic
✓	License MIT is approved by the Open Source Initiative	QC.Lic02	QC.Lic
✓	License MIT is listed under the Open Source Initiative popular category	QC.Lic02.1	QC.Lic
✓	Software metadata successfully validated	QC.Met01	QC.Met
✓	Source code files pass SAST checks performed by bandit tool	QC.Sec02	QC.Sec
✗	Python files are not fully compliant with flake8 (pycodestyle, pyflakes, mccabe) standard	QC.Sty01	QC.Sty
✓	Test cases are successfully passing using bash testing framework	QC.Uni01	QC.Uni
✓	The code repository uses tags for releasing new software versions	QC.Ver01.0	QC.Ver
✗	Latest release tag v1.1.0 found, but is not SemVer compliant	QC.Ver01	QC.Ver
✗	Not all release tags are SemVer compliant	QC.Ver02	QC.Ver

Quality badge



- SQAaaS-based badge: 
- shields.io-based badge: 
- Missing quality criteria for next level badge (gold): [QC.Sty](#)

🔗 View full report in the [SQAaaS platform](#)

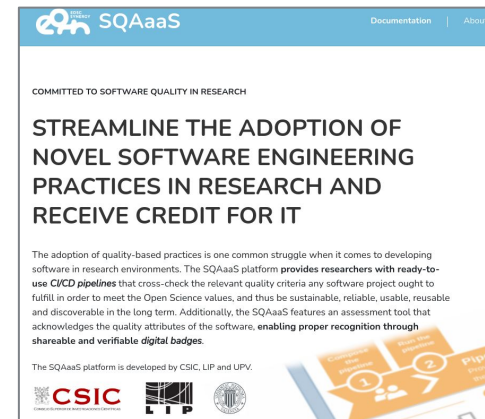


**Use case:
Workflow
validation in
DT-GEO project**

DT-GEO: Digital Asset validation

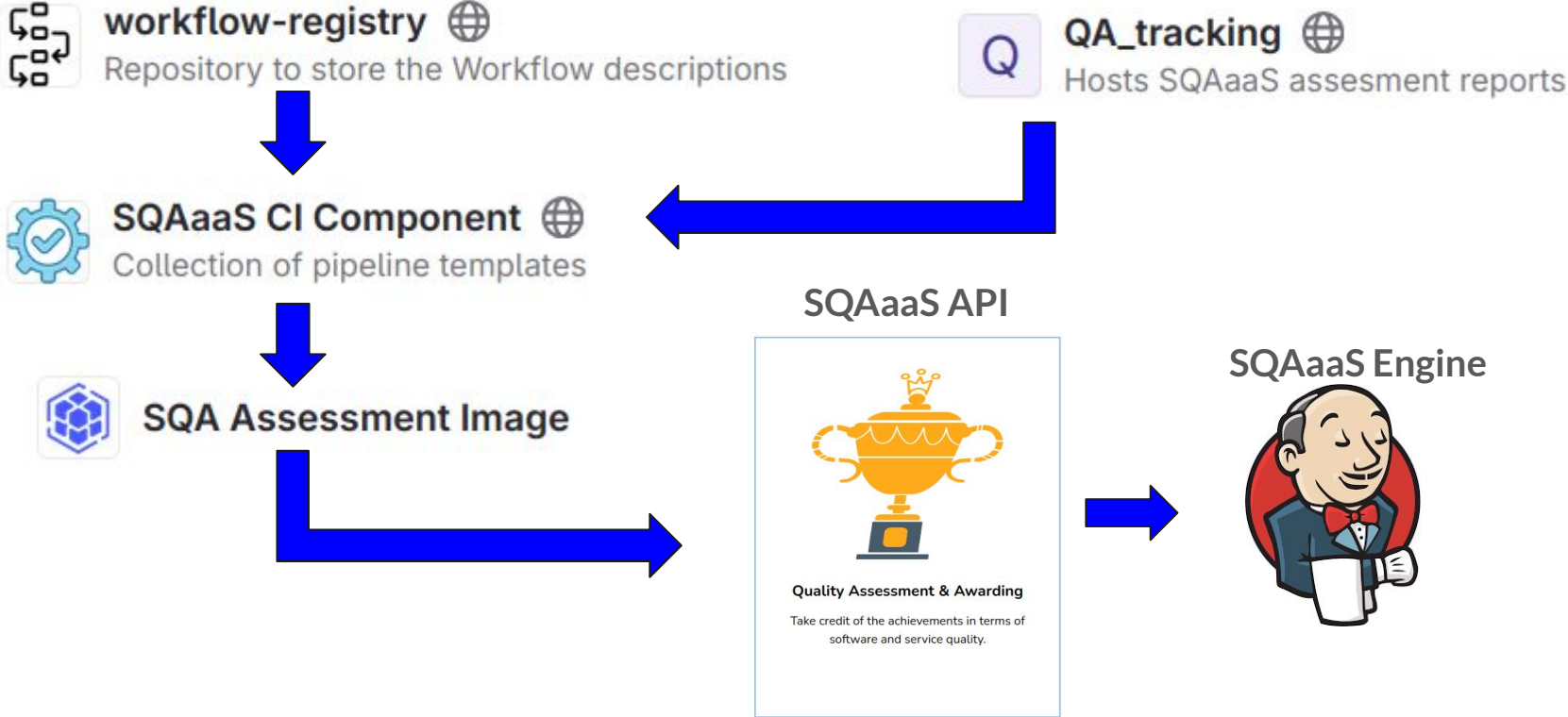
CI/CD for workflow validation

- ✓ GitLab's DT-GEO organisation: <https://gitlab.com/dtgeo>
 - [Workflow registry](#) (eFlows4HPC service) for the DTC workflow validation
 - [QA scheduled reports](#) for tracking DT-GEO digital assets through SQAaaS platform
- 🔄 Create Gitlab SQAaaS-based CI/CD pipelines
 - SQAaaS API client implementation using a Gitlab component
 - SQAaaS Engine generates the Jenkins pipeline
 - Trigger SQAaaS Jenkins PaC from git repository event:
<https://jenkins.eosc-synergy.eu>

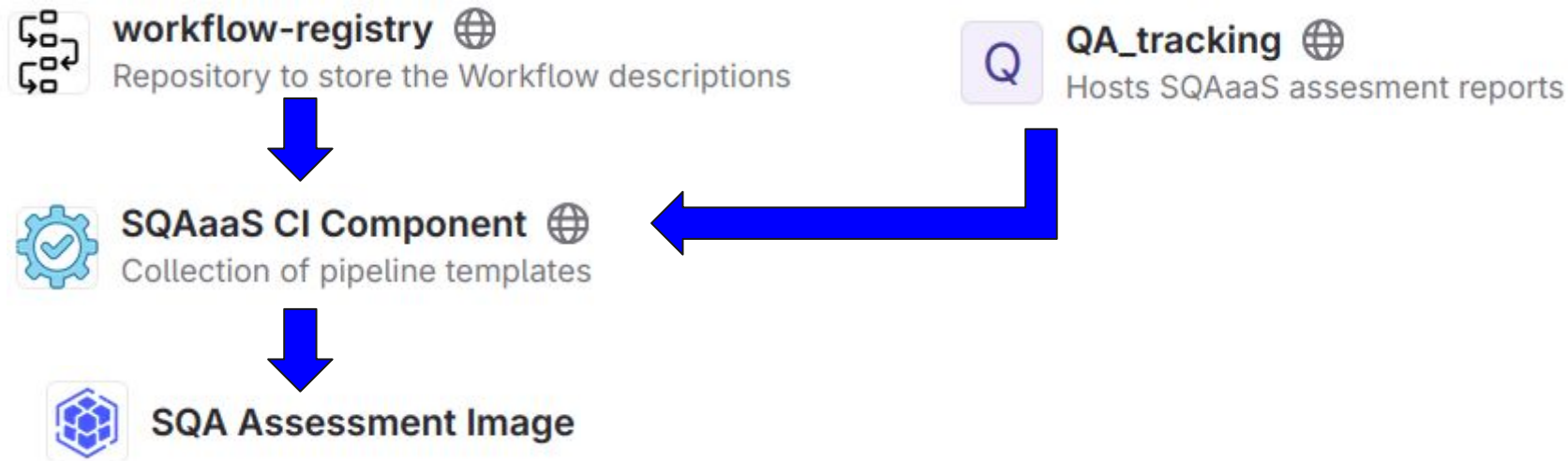


The screenshot shows the SQAaaS website homepage. At the top, there is a blue header with the SQAaaS logo on the left and 'Documentation' and 'About' links on the right. Below the header, the main content area has a white background. It starts with the tagline 'COMMITTED TO SOFTWARE QUALITY IN RESEARCH'. The main heading reads 'STREAMLINE THE ADOPTION OF NOVEL SOFTWARE ENGINEERING PRACTICES IN RESEARCH AND RECEIVE CREDIT FOR IT'. Below this, a paragraph explains the platform's purpose: '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.' At the bottom, it states 'The SQAaaS platform is developed by CSIC, LIP and LPV.' and includes logos for CSIC, LIP, and LPV. On the right side of the page, there is a partial view of a graphic showing a pipeline diagram with steps 1 and 2.

Gitlab pipeline using SQAaaS



The Gitlab pipeline components



Gitlab pipeline for Workflow validation



workflow-registry 

Repository to store the Workflow descriptions

include:

- component: `$CI_SERVER_FQDN/dtgeo/workflow-management-system/workflow-registry/DTC@$CI_COMMIT_REF_NAME`
- component: `$CI_SERVER_FQDN/dtgeo/workflow-management-system/workflow-registry/CIC@$CI_COMMIT_REF_NAME`

inputs:

`cic_builder_job_script: ./bin/cic_builder.sh`

depends_on:

- `collect_dtc_info`

- component: `$CI_SERVER_FQDN/dtgeo/metadata/sqa-assessment-ci-template/SQA-step.gitlab-ci@main`
- component: `$CI_SERVER_FQDN/dtgeo/metadata/sqa-assessment-ci-template/SQA.gitlab-ci@main`

inputs:

`subfolder: $INPUT_SUBFOLDER`

depends_on:

- `collect_dtc_info`

SQAaaS integration implemented using Gitlab components

Gitlab pipeline for Workflow validation



workflow-registry

Repository to store the Workflow descriptions

<https://gitlab.com/dtgeo/workflow-management-system/workflow-registry>

- Retrieve DTC configurations for the workflow: DTC component
- Container image creation: CIC component



SQAaaS CI Component

Collection of pipeline templates

<https://gitlab.com/dtgeo/metadata/sqa-assessment-ci-template>

- Config generator job definition: SQA-step component
- SQAaaS API assess job: SQA component



SQA Assessment Image

<https://gitlab.com/dtgeo/sqa-assessment-image>

- Config generator: step image
- Trigger SQAaaS API assess: assess image

Gitlab pipeline for QA reporting



QA_tracking

Hosts SQAaaS assesment reports

```
include:  
  - component: gitlab.com/dtgeo/sqa-assessment-ci-template/SQA.gitlab-ci@main  
  inputs:  
    reporting_folder: $LOCAL_FOLDER  
    depends_on: []
```

```
report-analysis:  
  stage: register  
  image: docker:latest  
  variables:  
    GIT_STRATEGY: clone  
  script:  
    - echo ${INPUT_REPORT_FILE_JSON}  
    - git status  
    - git checkout -b $CI_COMMIT_BRANCH
```

All 10 Active Inactive		
Description	Interval	Target
[SS7303] test	19 13 * * 5 Europe/ Madrid	🔗 main
[SS7302] eGSIM	27 20 * * 5 Europe/ Madrid	🔗 main

Gitlab pipeline using SQAaaS



QA_tracking

Hosts SQAaaS assesment reports

https://gitlab.com/dtgeo/metadata/QA_reports_dtgeo

- Scheduled pipelines to keep tracking QA analysis
- Generate the QA reports
- Store the reports in Git



SQAaaS CI Component

Collection of pipeline templates

<https://gitlab.com/dtgeo/metadata/sqa-assessment-ci-template>

- SQAaaS API assess job: SQA component

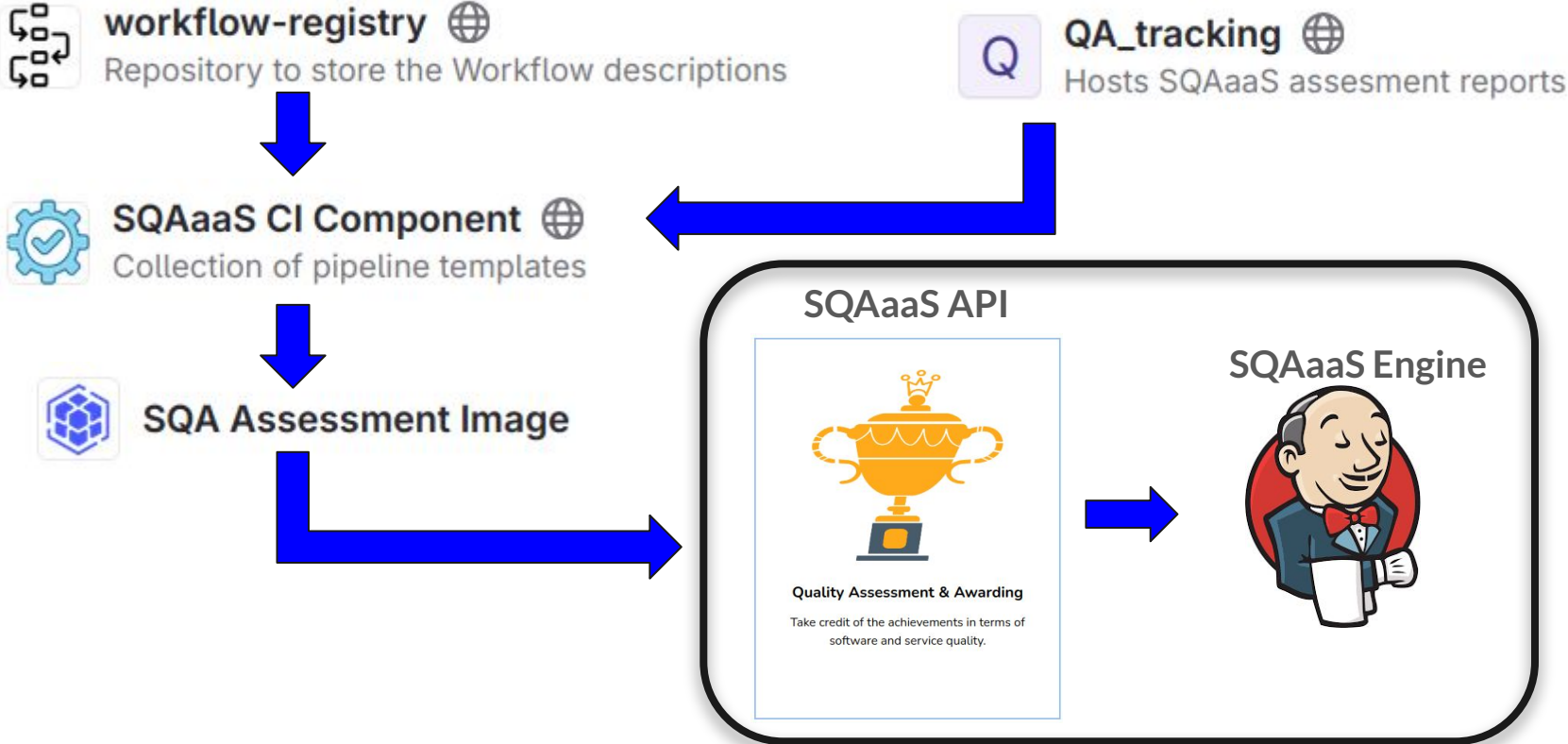


SQA Assessment Image

<https://gitlab.com/dtgeo/sqa-assessment-image>

- Trigger SQAaaS API assess: assess image

SQAaaS runtime



SQAaaS runtime

Resources Harvester:

- Use Jenkins Swarm Plugin
 - Accessing computing clusters
 - Accessing virtualization resources
- Use Jenkins Kubernetes Plugin
 - Require access to the Kubernetes cluster API from Jenkins Controller
 - Run dynamic agents in the cluster


SQAaaS runtime

Security and network isolation:

- Jenkins Agents can be completely isolated behind a firewall and use a proxy to connect to the Jenkins Controller
- Jenkins Controller can have access restricted behind a proxy
- Access resources that are not accessible from the Internet
 - Code is loaded to the target infrastructure
 - Create the pipeline using the pulled code from Git
 - Build or pull the required container images

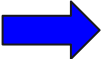
SQAaaS Engine using Slurm

SQAaaS API





Quality Assessment & Awarding


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



SQAaaS Engine



Jenkins Controller



Jenkins Swarm Plugin

<https://github.com/jenkinsci/swarm-plugin>

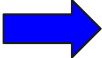
Setup Swarm Agent



ANSIBLE



Slurm Jobs



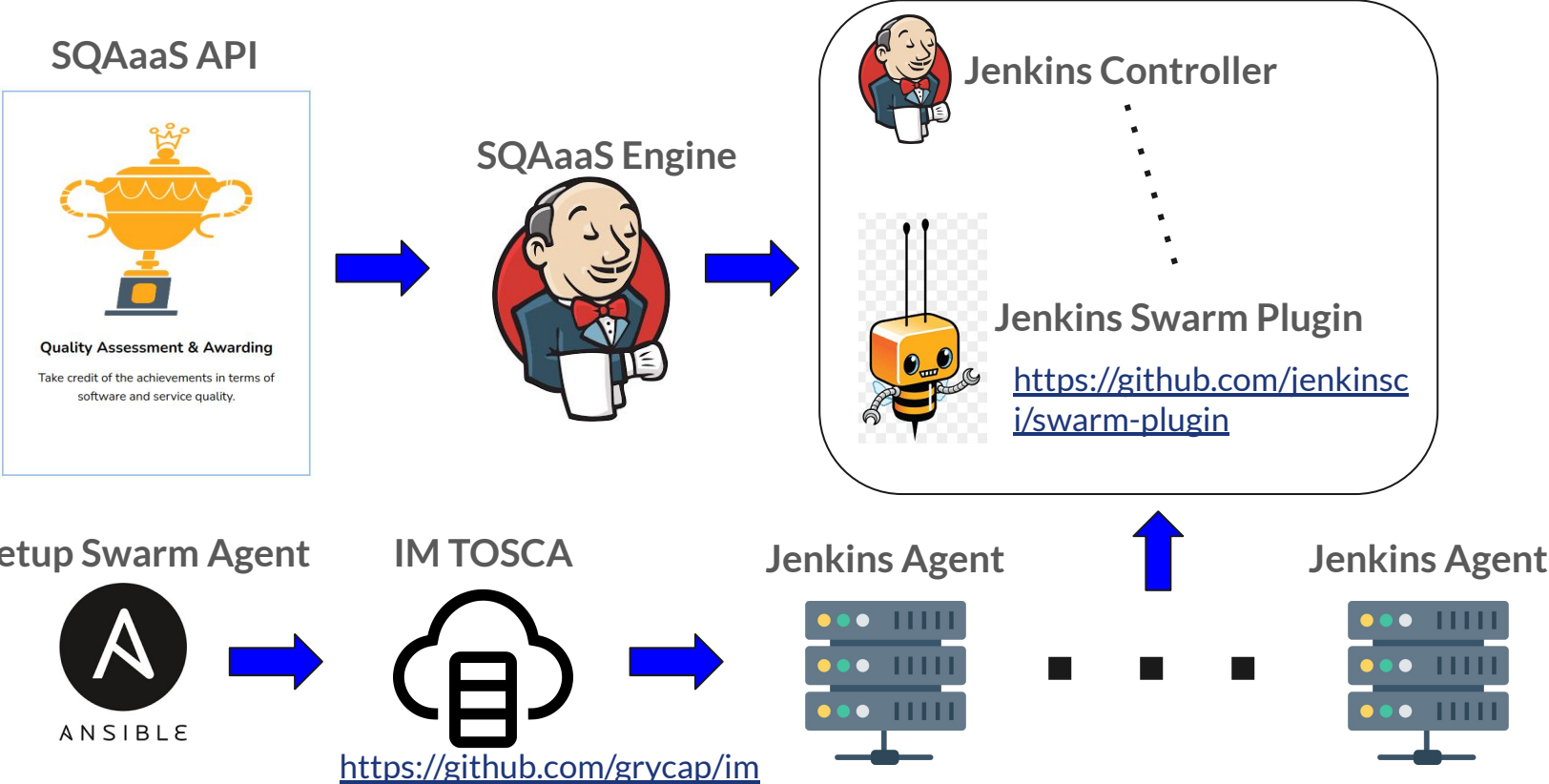
Jenkins Agent



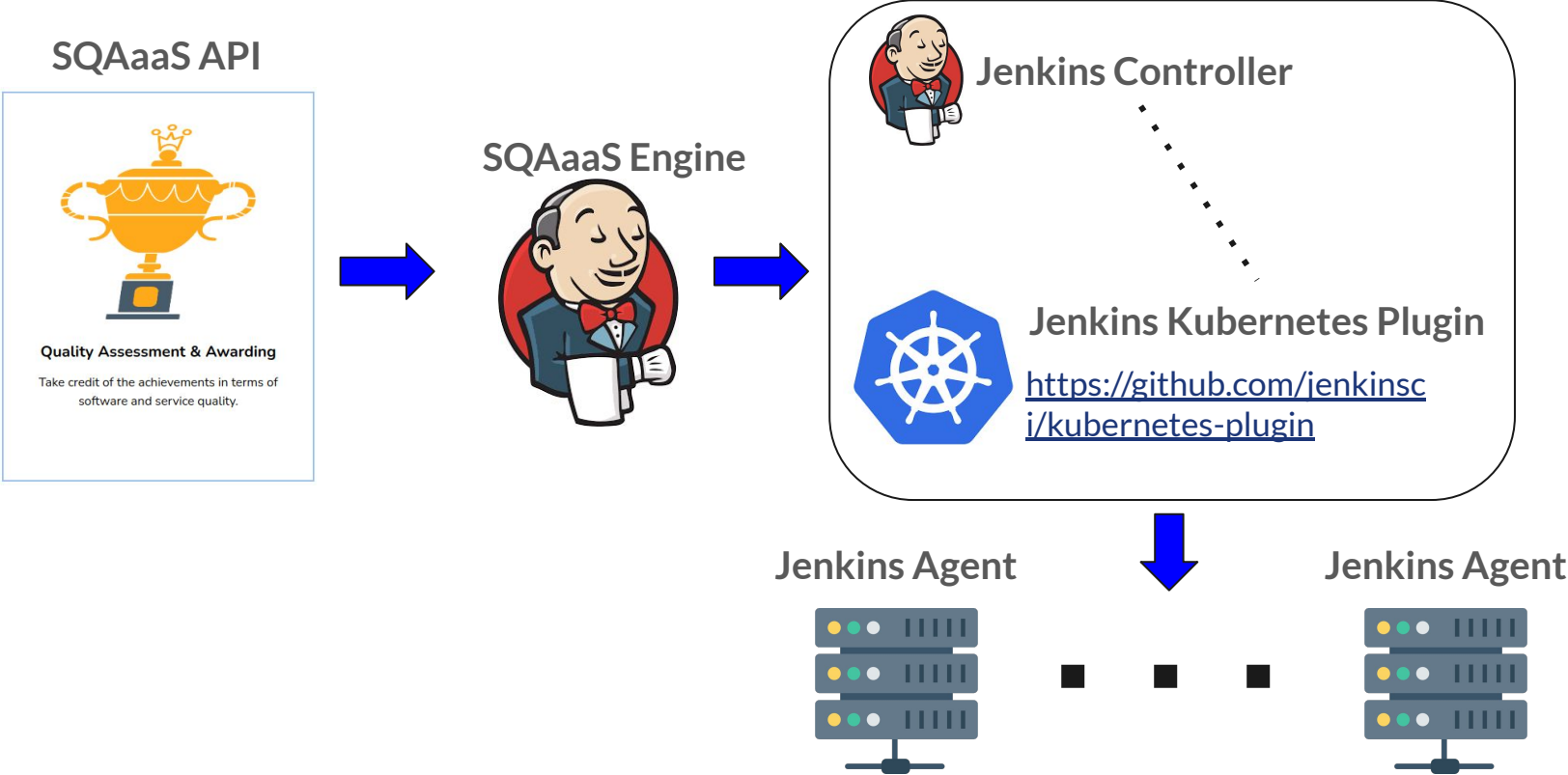
Jenkins Agent



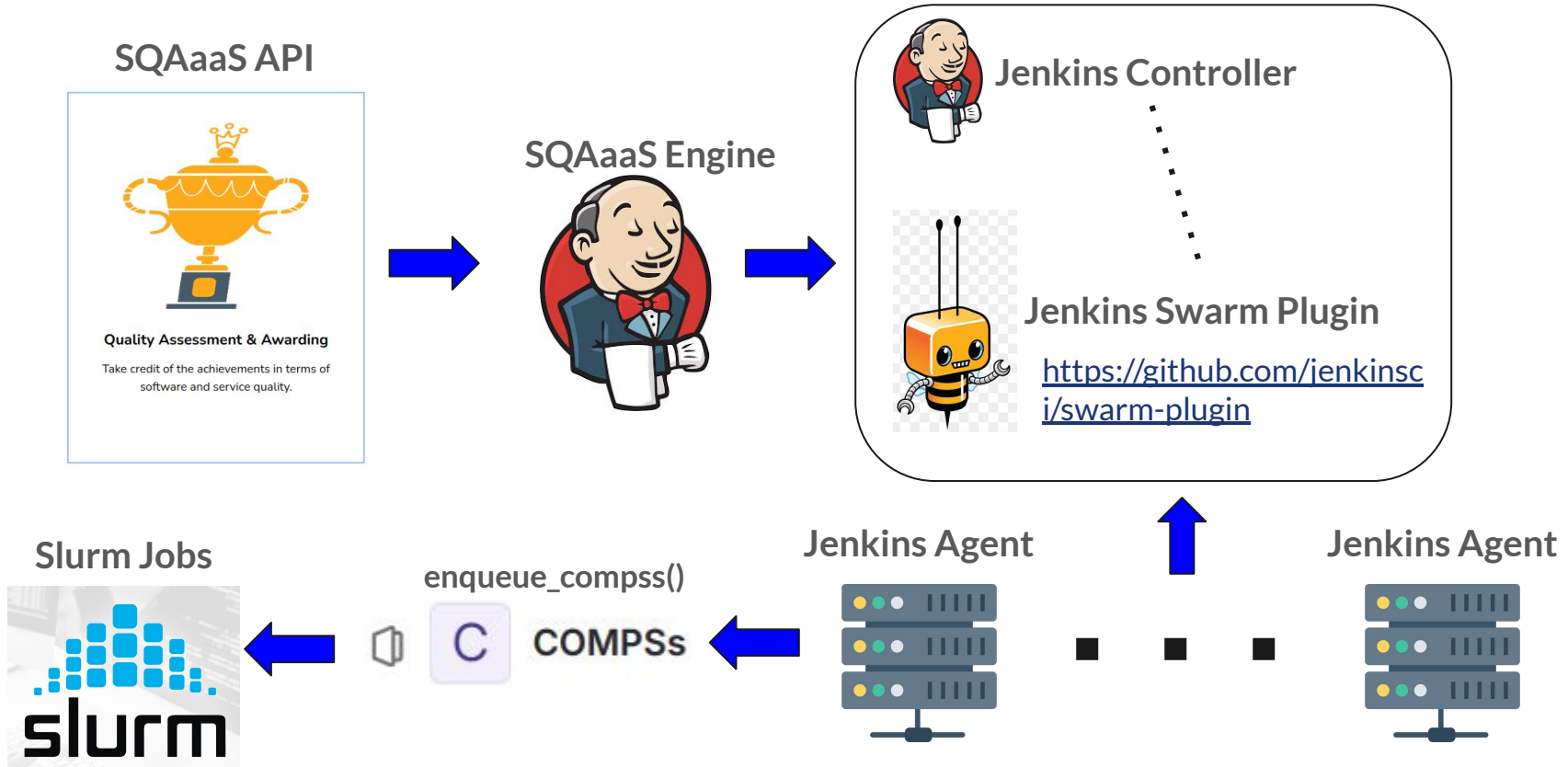
SQAaaS Engine using Virtualization



SQAaaS Engine using Kubernetes



Workflow Validation with SQAaaS





**Thank you for your
attention**

Q&A time

Pablo Orviz

orviz@ifca.unican.es
IFCA-CSIC

Samuel Bernardo

samuel@lip.pt
LIP

