Software and Services Quality-as-a-Service

Jorge Gomes on behalf of EOSC-Synergy WP3

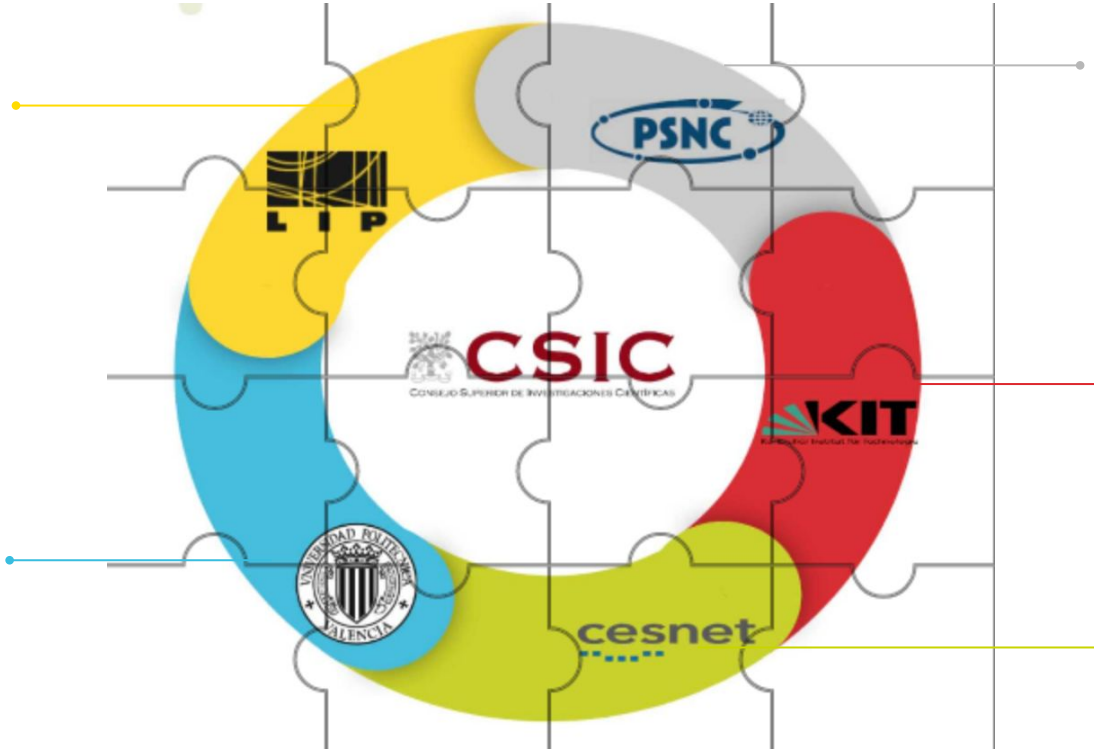
EOSC-SYNERGY receives funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 857647

Promoting EOSC High Quality Services

Software & Services quality as a service, FAIRness evaluation and quality certification badges

Thematic Services Integration

10 thematic services addressing 4 scientific areas (Earth Observation, Environment, Biomedicine and Astrophysics)



Skills development

Environment for tutorials with a dedicated MOOC platform, courses methodology and a Hackaton as a service platform.

Capacity Expansion at the Infrastructure level

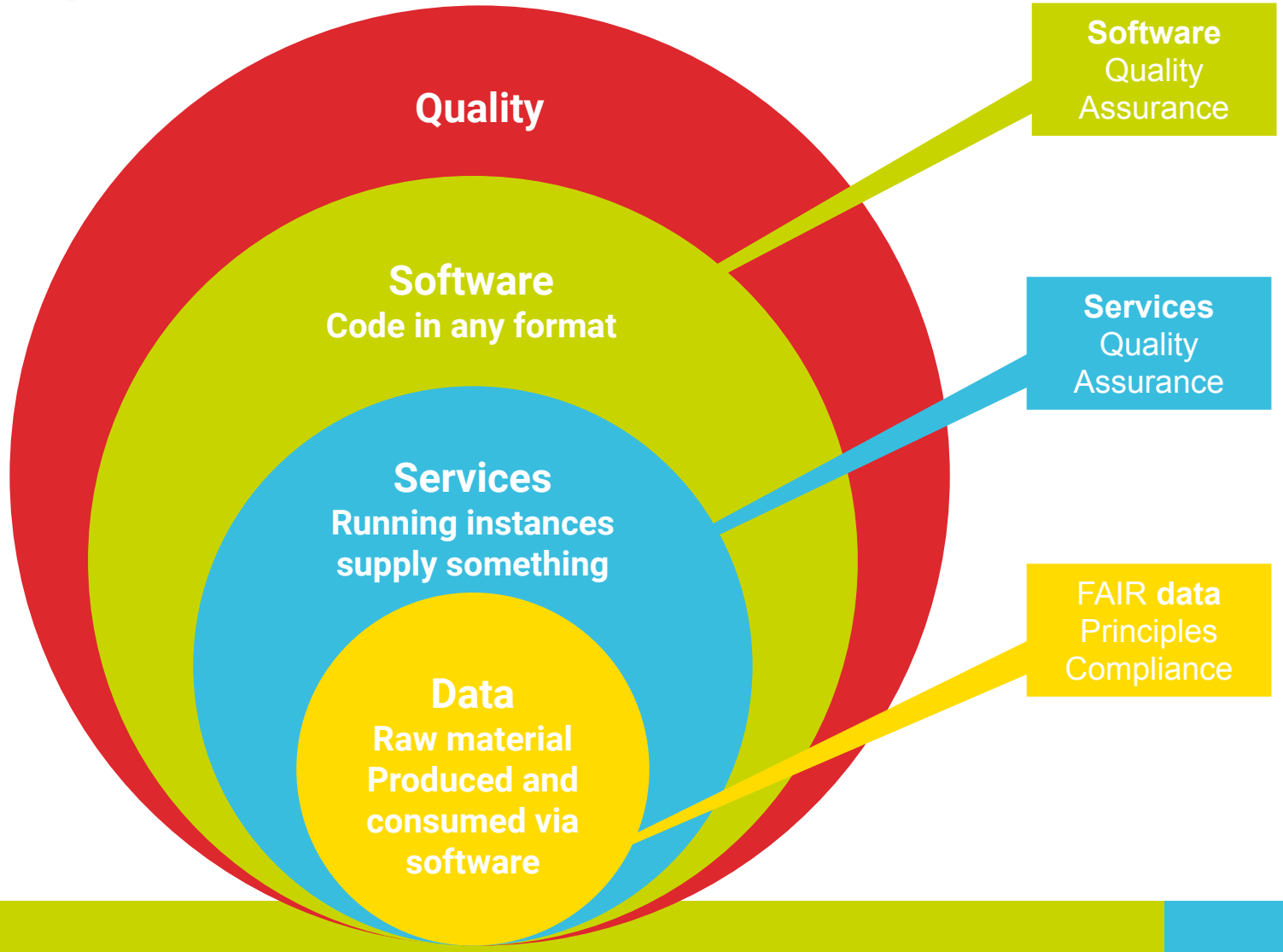
Integration of services and resources from the RIs of the consortium partners.

Alignment at the Policy Level

Collaboration with regional projects on landscaping activities, gap analysis and contribution to EOSC policies.

Quality for Software, Services & Data

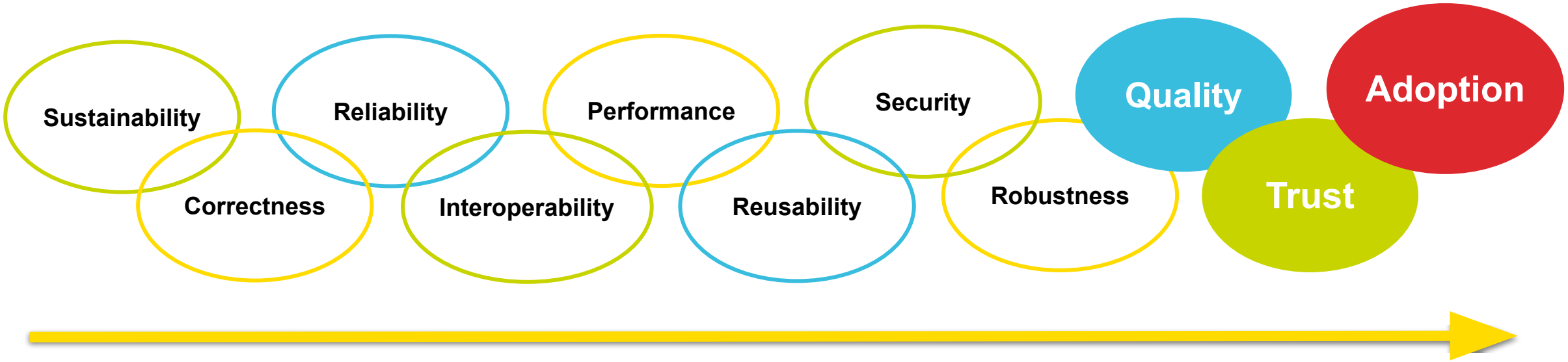
- Data is produced and consumed using software
- EOSC needs
  - **software**
  - **services**
  - **data**
- Quality must be transversal across EOSC software, services and data



Fostering Service Integration and Adoption

Quality based approach to service integration and adoption

A road towards EOSC adoption



Fostering adoption through quality

Quality is essential to foster adoption of EOSC services and data.

- **Software and Services**:
  - Promote adoption of best practices.
  - Automatically validate software and services quality
  - of both: thematic services and generic services.

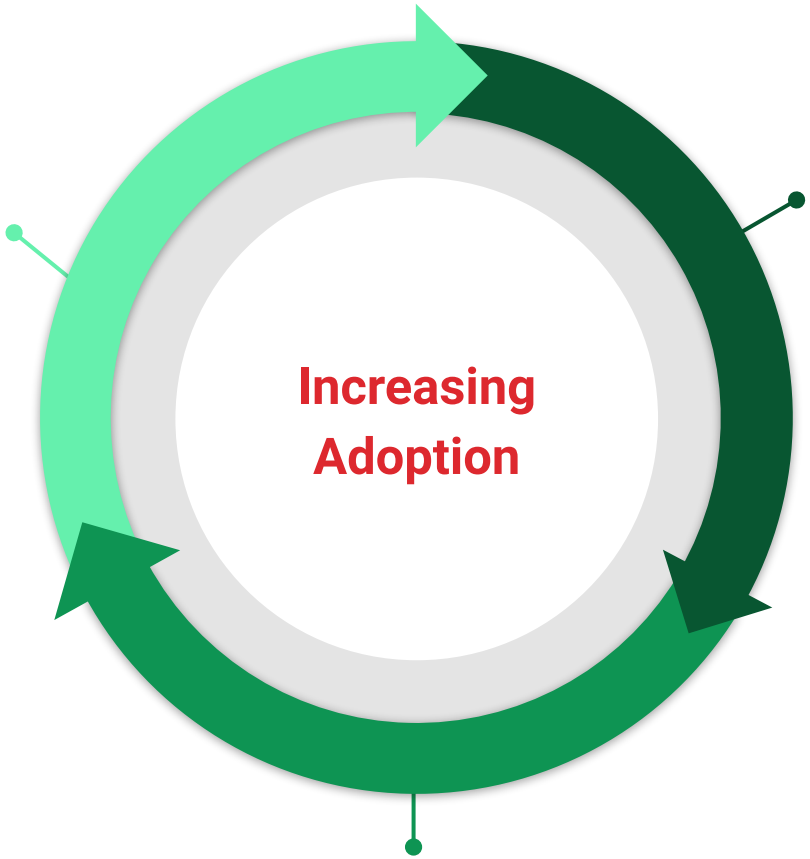
- **Data**:
  - Promote the adoption of FAIR data principles.
  - Leverage actionable features on data repositories to
  - analyse and validate FAIR compliance.

Virtuous cycle

Improved Software Quality

Developers/Integrators

Increase software quality
Adhere to software development best practices



Providers/Operations

Increase service quality
Adhere to service delivery best practices

Increased Usage

Improved Services Quality

Users

More aware of EOSC services quality
Build trust and increase adoption

Provide incentives to all parties

- Developers, Providers incentives to improve quality
  - Establish foundations for an EOSC-ready stamp (**quality certification**)
  - Reward quality achievements for: software, services and data (**quality visibility**)

- Users incentives to adopt EOSC
  - Increased visibility of EOSC services and data (**increased visibility**)
  - Availability of mature validated software, services and data (**increased trust**)

EOSC-Synergy Digital Badges

A process to foster quality and adoption

Push the EOSC state-of-the-art in software and services life-cycle

Facilitate the integration of EOSC services

Services Thematic or Generic	Quality Baselines	SQAaaS Platform	Quality Badges	Trusted Integrated Services
• Software source code • For EOSC services • Being integrated or being maintained	• Best practices • Specifications • Guide development and integration • Baseline for quality verification	• Framework to perform validation • Implements the quality baselines • Enables on-demand verification	• Reward quality • Make quality attributes visible • Increase trust • Promote adherence	• More dependable • Easier to maintain • Easier to support • Easier to use • More reliable • Safer
Thematic Services internal and external	**Software baseline Services baseline FAIR principles**	**Jenkins Pipeline Library SQAaaS FAIR metadata validation**	**Digital badges OpenBadges**	**EOSC ready services Increased satisfaction**

Quality baseline criteria

- Definition of quality requirements:
  - Generic definition not tied to any particular implementation
  - Aimed at automated validation
- Aims to improve:
  - Reliability, sustainability, and reusability of software and services.
- Two sets of quality criteria have been developed:
  - **Software - improve EOSC software quality**
  - **Services - improve EOSC services quality**
- In practice is implemented by JePL and SQAaaS
  - Already being used by several thematic services


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.

This manuscript (permalink) was automatically generated from indigo-dc/sqa-baseline@a9c34fa on April 29, 2020.

Authors

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

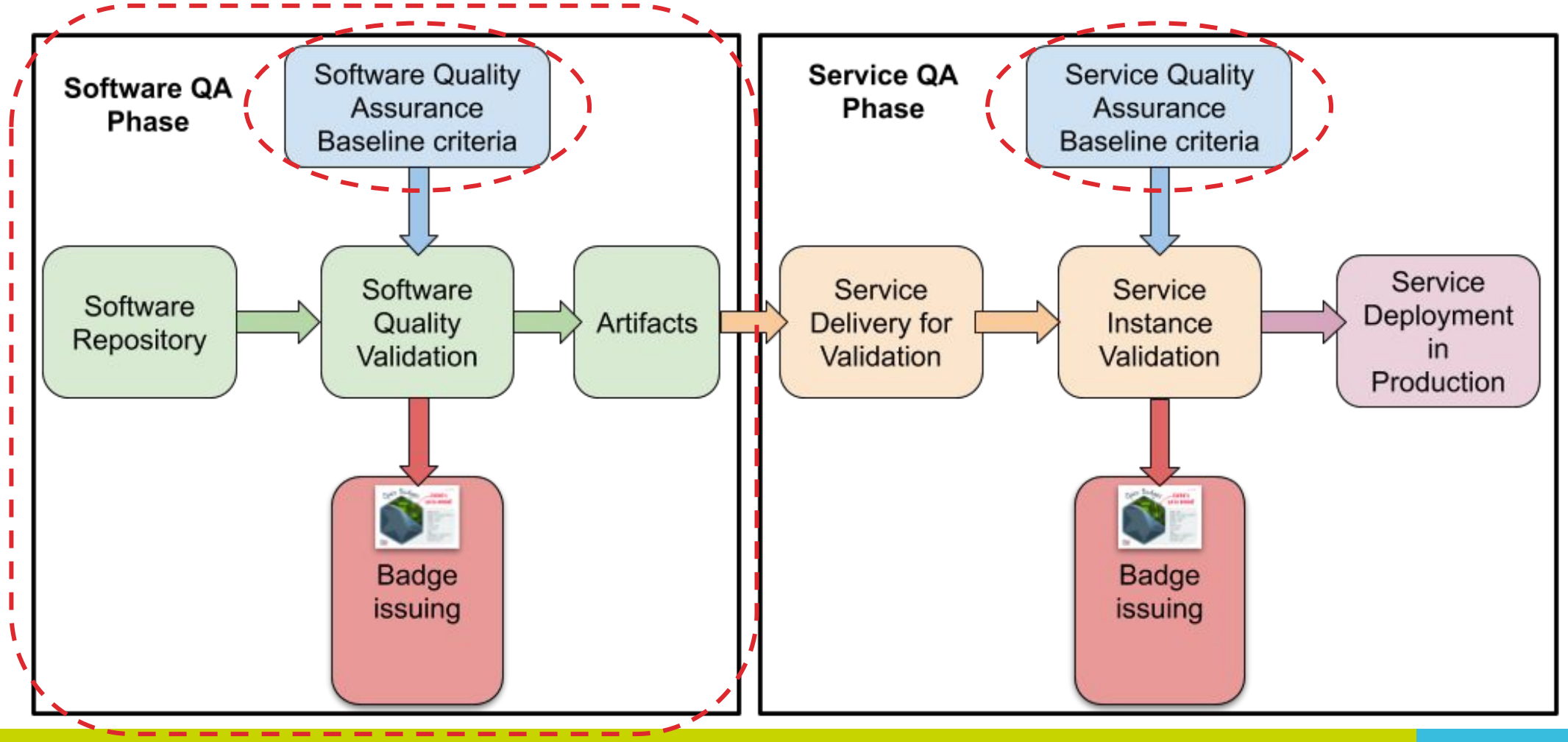
A DOI-citable version of this manuscript is available at http://hdl.handle.net/.

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

Authors

- Pablo Orviz 0000-0002-2473-6405 · orviz Spanish National Research Council (CSIC); Institute of Physics of Cantabria (IFCA)
- Mario David 0000-0003-1802-5356 · mariojmdavid
- Jorge Gomes 0000-0002-9142-2596 · jorge-lip
- Joao Pina 0000-0001-8959-5044 · jopina
- Samuel Bernardo 0000-0002-6175-4012 · samuelbernardolip
- Isabel Campos 0000-0002-9350-0383 · isabel-campos-plasencia Spanish National Research Council (CSIC); Institute of Physics of Cantabria (IFCA)

Two sets of Quality Criteria for: software and services

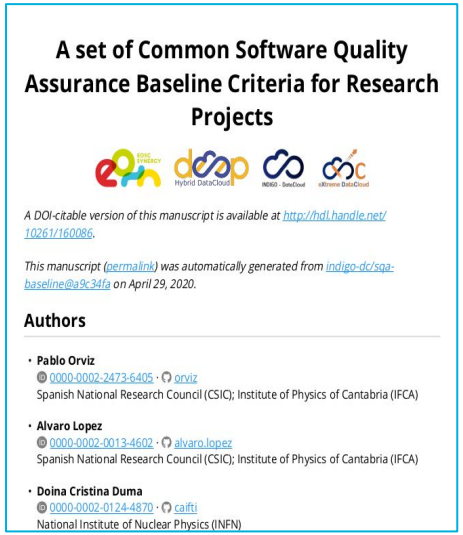


Software Quality Baseline

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

- http://dx.doi.org/10.20350/digitalCSIC/12543
- Initially created in the INDIGO-DataCloud project.
- Used by several projects.

- Based on widely adopted best practices and first-hand experience.
- Developed through discussions aimed and DevOps.
- Focus on software and continuous integration (CI)



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.

This manuscript (permalink) was automatically generated from indigo-dc/sqa-baseline@a9c34fa on April 29, 2020.

Authors

- **Pablo Orviz**
  0000-0002-2473-6405 · orviz
  Spanish National Research Council (CSIC); Institute of Physics of Cantabria (IFCA)
- **Alvaro Lopez**
  0000-0002-0013-4602 · alvaro.lopez
  Spanish National Research Council (CSIC); Institute of Physics of Cantabria (IFCA)
- **Doina Cristina Duma**
  0000-0002-0124-4870 · caifti
  National Institute of Nuclear Physics (INFN)

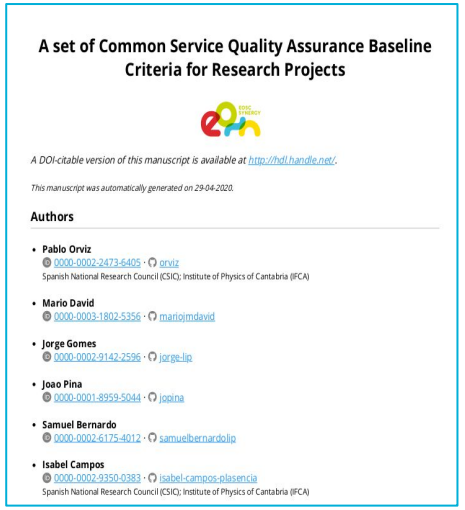
- Code Accessibility
- Licensing
- Code Workflow
- Code Management
- Code Style
- Code metadata
- Unit Testing
- Functional Testing
- Integration Testing
- Documentation
- Security
- Code Review
- Automated Deployment

Service Quality Baseline

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

- http://dx.doi.org/10.20350/digitalCSIC/12533
- New criteria that is appropriate for services (long running instances of software that supply something)

- Resulted from discussion with developers, infrastructure managers and IT service mgmt.
- Focus on services and continuous delivery (CD)
- Will the focus for the 2nd project period



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.

Authors

- Pablo Orviz
  0000-0002-2473-6405 · orviz
  Spanish National Research Council (CSIC); Institute of Physics of Cantabria (IFCA)
- Mario David
  0000-0003-1802-5356 · mariojmdavid
- Jorge Gomes
  0000-0002-9142-2596 · jorge-lip
- Joao Pina
  0000-0001-8959-5044 · jopina
- Samuel Bernardo
  0000-0002-6175-4012 · samuelbernardolip
- Isabel Campos
  0000-0002-9350-0383 · isabel-campos-plasencia
  Spanish National Research Council (CSIC); Institute of Physics of Cantabria (IFCA)

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

Example of software quality criteria

Unit Testing [QC.Uni]

Unit testing evaluates all the possible flows in the internal design of the code, so that its behaviour becomes apparent. It is a key type of testing for early detection of failures in the development cycle.

- **[QC.Uni01]** Minimum acceptable code coverage threshold SHOULD be 70%.
  - **[QC.Uni01.1]** Unit testing coverage SHOULD be higher for those sections of the code identified as critical by the developers, such as units part of a security module.
  - **[QC.Uni01.2]** Unit testing coverage MAY be lower for external libraries or pieces of code not maintained within the product's code base.
- **[QC.Uni02]** Units SHOULD reside in the repository code base but separated from the main code.
- **[QC.Uni03]** Unit testing coverage MUST be checked on change basis.
- **[QC.Uni04]** Unit testing coverage MUST be automated.
  - **[QC.Uni04.1]** When working on automated testing, the use of testing doubles is RECOMMENDED to mimic a simplistic behaviour of objects and procedures.

SQAaaS platform: concept and goals

1. Exploit the **2 baselines of QA criteria**
   - Software (CI) https://github.com/indigo-dc/sqa-baseline
   - Services (CD) https://github.com/EOSC-synergy/service-qa-baseline

2. Solution to assess the QA criteria

   ⇝ **SQA as a Service** (aka SQAaaS)

   - Under development → github.com/eosc-synergy
   - Based on **DevOps** practices
   - Using **CI/CD pipelines**
   - First prototype of the SQAaaS platform implemented
   - The core library JePL is already PRODUCTION quality

DevOps and CI/CD pipelines

- **DevOps** => movement
  - Improve efficiency reduce time from software development to production
  - Requires heavy automation of the development and deployment process
- **CI/CD** => method
  - Method to release software more frequently
  - Relying on automation
- **Jenkins** => tool
  - Open source CI/CD system for automation to build software, test & deploy
- **Jenkins Pipeline** => implements a process from testing to deployment
  - Allows building the software in a reliable and repeatable manner
  - Allows to automate the multiple stages of the process from testing to deployment

What is a CI/CD pipeline

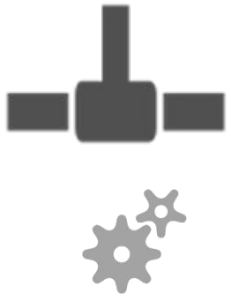
Software Source Code

Development

Version Control

Track code changes

Build



Produce executables

Static Testing SAST

Style checks
Unit tests
Integration tests

Produce Artifacts

Produce release

Dynamic Testing DAST

Security tests
Health tests
Diagnostics

Quality Badge Issuance

Certify quality attributes

EOSC Marketplace

Quality verified

SQAaaS platform: in short

Facilitate the assessment of the quality of research software.

Dynamic composition and execution of CI/CD pipelines and analysis of the results.

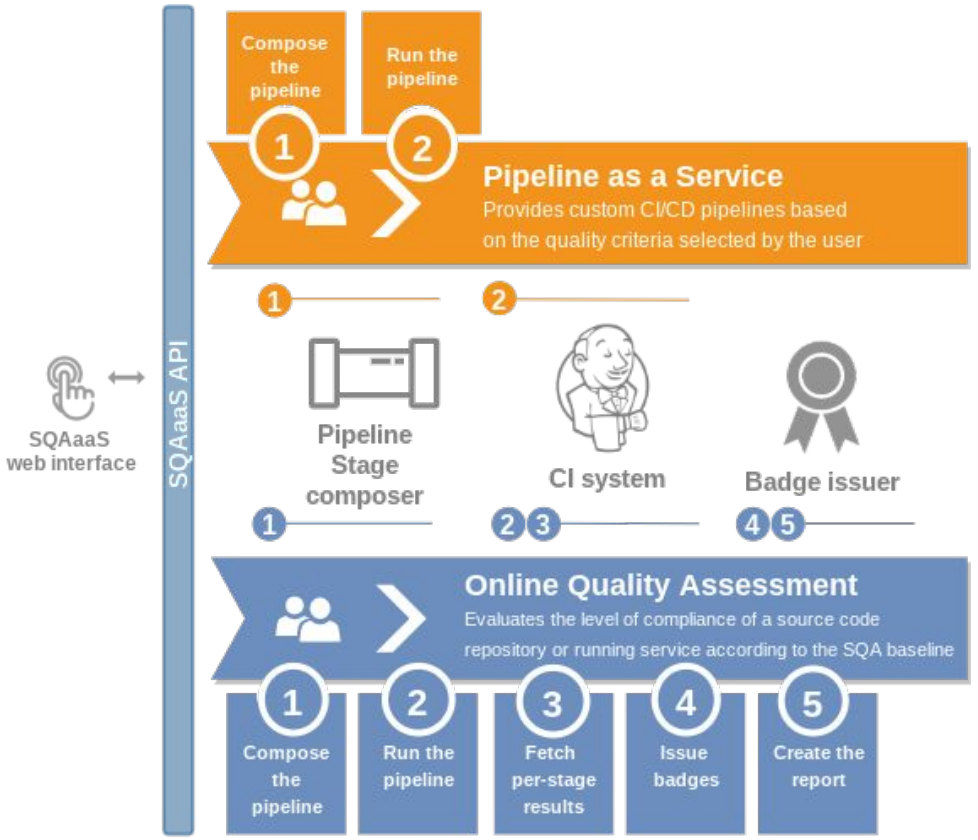
Two usage scenarios:

A. **Pipeline as a Service**

B. **SQA Assessment**

Components:

- *JePL*
- *SQAaaS Web & API*
- *Jenkins CI*
- *Badge issuer*



JePL (Jenkins Pipeline Library)

What?

- Core component of the SQAaaS platform, leverages Jenkins
  - Can be used independently from SQAaaS
  - Audience → EOSC service developers, any computational scientist or RSE

Why?

- Facilitates adoption of DevOps practices→ Improve softw. & serv. QA in research environments
  - Compliance with software & services baselines requires CI/CD pipelines!

How?

- Using (human-readable) YAML format, instead of Jenkins PaC specific language
  - Only a minimal Jenkinsfile is required
- Uses containers and Docker Compose to orchestrate required services

Where?

- https://github.com/indigo-dc/jenkins-pipeline-library

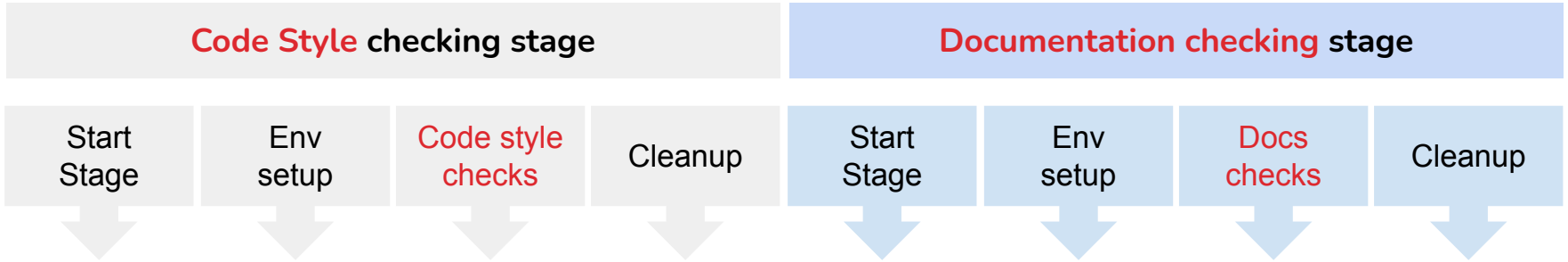
SQAaaS: JePL (Jenkins Pipeline Library)

- Already supports the most important software quality criteria:
  - style checking (qc_style)
  - unit tests (qc_unit)
  - functional tests (qc_functional)
  - documentation (qc_doc).
- Requires three configuration files that must be created:
  - config.yml : the quality verification stages
  - docker-compose.yml : containers to be deployed
  - Jenkinsfile : starting point invokes the JePL library
- The latest JePL release is 2.1.0
  - already tested by multiple use cases

JePL YAML Example

```yaml
config:
 project_repos:
   o3skim:
     repo: 'https://git.scc.kit.edu/synergy.o3as/o3skim.git'
     branch: master

sqa_criteria:
 qc_style:
   repos:
     o3skim:
       container: testing
       tox:
         tox_file: /o3skim/tox.ini
         testenv:
           - stylecheck

 qc_functional:
   repos:
     o3skim:
       container: testing
       tox:
         tox_file: /o3skim/tox.ini
         testenv:
             - functional

 qc_security:
   repos:
     o3skim:
       container: testing
       tox:
         tox_file: /o3skim/tox.ini
         testenv:
           - bandit

 qc_doc:
   repos:
     o3skim:
       container: testing
       tox:
         tox_file: /o3skim/tox.ini
         testenv:
           - docs
```

SQAaaS pipelines



SQA baseline dynamic stages: qc-style	Environment Setup	qc_style jpl-validator	Docker Compose cleanup	SQA baseline dynamic stages: qc-doc	Environment Setup	qc_doc jpl-validator	Docker Compose cleanup
13s	220ms	6s	17s	9s	0ms	37s	0ms
13s	232ms	7s	16s	12s	162ms	37s	16s

SQAaaS is a complete integrated solution to ease the creation and execution of quality assurance pipelines hiding the complexity from the researchers and developers

SQAaaS: use cases already using JePL

- Internal JePL usage from SQAaaS services themselves
  - JePL schema validator (validates JSON schema & builds validator's Docker image)
  - SQAaaS Web (builds & publishes production Web)
  - SQAaaS API (validates OpenAPI spec, builds & publishes API docs

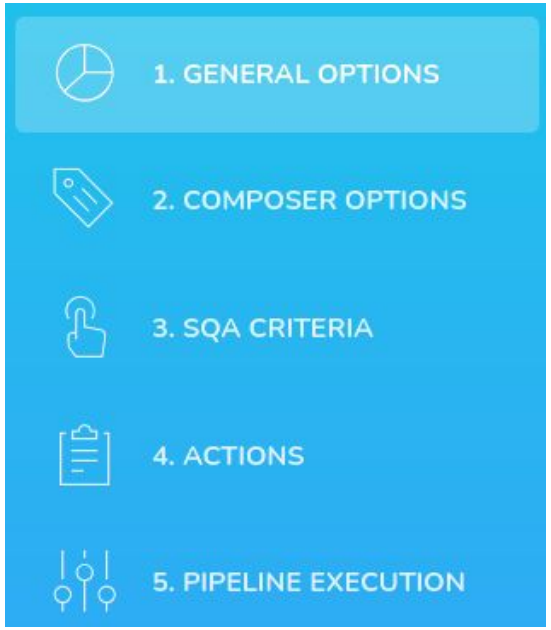
- Thematic services with ready pipelines
  - **WORSICA** https://jenkins.eosc-synergy.eu/job/WORSICA/
  - **O3AS** https://jenkins.eosc-synergy.eu/job/eosc-synergy-org/ (o3* projects)
  - **SAPS** https://jenkins.eosc-synergy.eu/job/eosc-synergy-org/ (saps-* projects)
  - **LAGO** https://jenkins.eosc-synergy.eu/job/eosc-synergy-org/job/onedataSim/
  - **OpenEBench** https://jenkins.eosc-synergy.eu/job/eosc-synergy-org/job/bench_event_api/

- More than 20 thematic service repositories are already using JePL

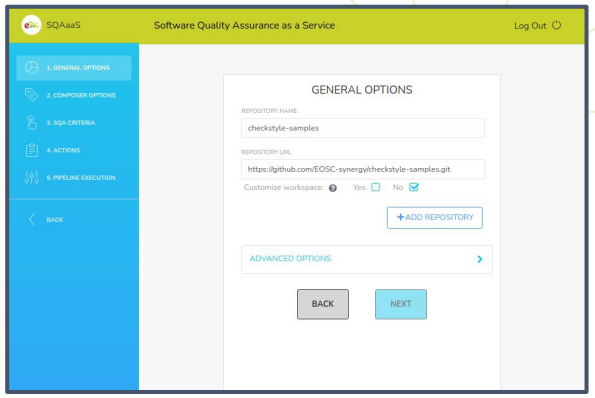
SQAaaS: Web

Pipeline as a Service

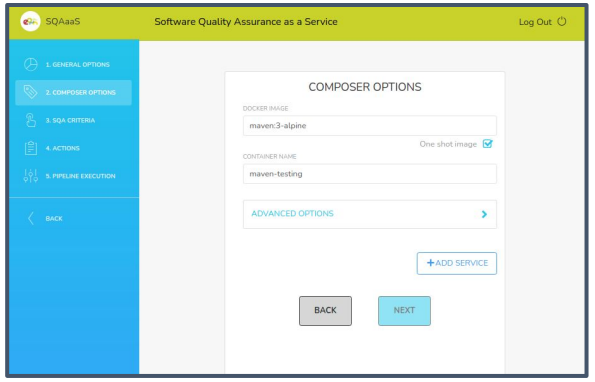
Custom CI/CD pipelines based according to the quality criteria from the Software as a Service.



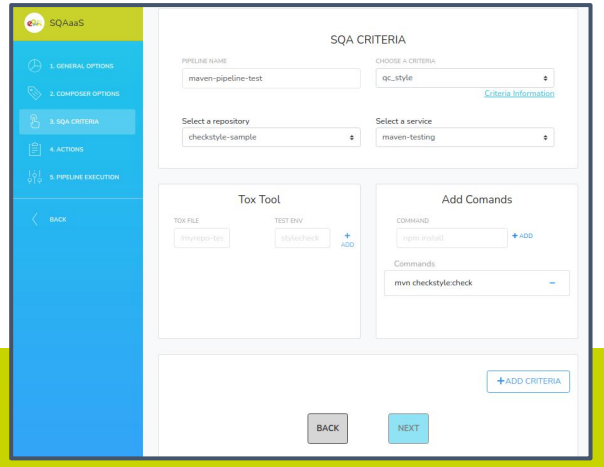
1. General options panel



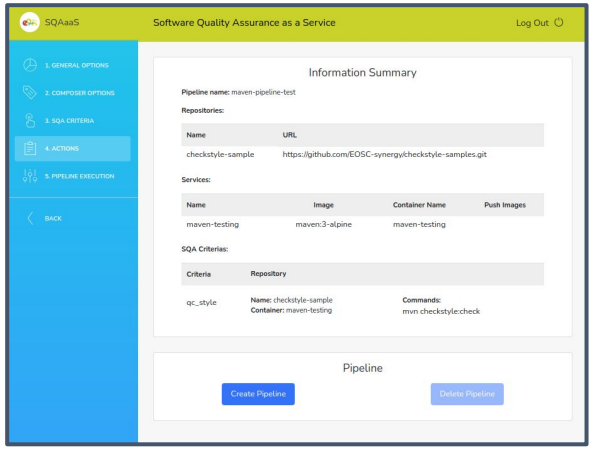
2. Composer Options panel



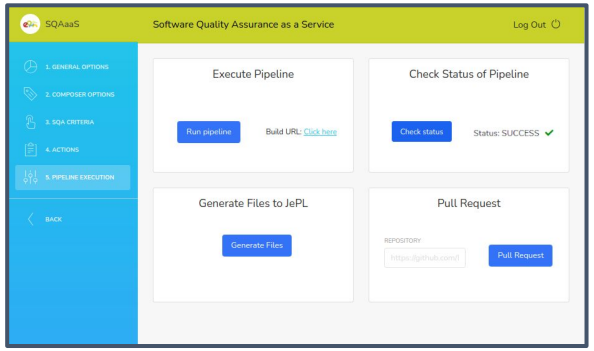
3. SQA Criteria panel



4. Actions panel



5. Pipeline execution panel

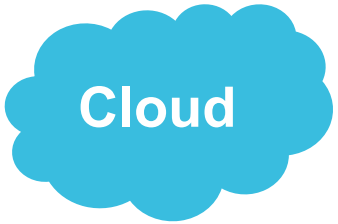


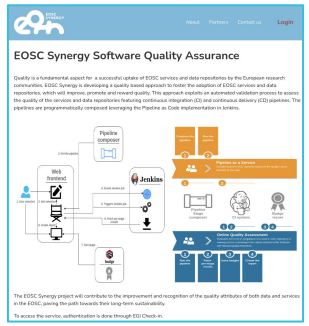
Underneath uses JePL, provides simple interface:

A. **Pipeline as a Service: finishing 1st prototype**
B. Full validation with badges: under development

SQAaaS: private deployment

- Aim ⇒ allow end users to have their SQAaaS deployment
  - Facilitates SQAaaS testing & promotes adoption
  - Important for closed/private environments
  - Follows WP2 infrastructure approach





SQAaaS platform

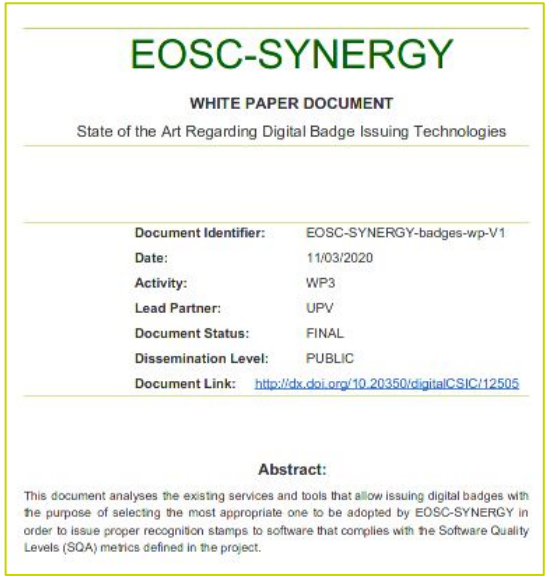
https://github.com/EOSC-synergy/SQAaaS
https://operatorhub.io/operator/jenkins-operator

Deployment in any cloud

Deployment on existing k8s

SQAaaS: digital badges

- Reward adherence to the quality baseline criteria.
- The badges can be added to repositories:
  - to increase visibility;
  - provide a verifiable method to assess the quality achievements.
- Whitepaper on digital badges:
  - *OpenBadges* specification;
  - *Badgr* will be the technology to issue digital badges.
- EOSC-Synergy Badges endpoint
  - https://badges.eosc-synergy.eu/



EOSC-SYNERGY

WHITE PAPER DOCUMENT

State of the Art Regarding Digital Badge Issuing Technologies

Document Identifier:	EOSC-SYNERGY-badges-wp-V1
Date:	11/03/2020
Activity:	WP3
Lead Partner:	UPV
Document Status:	FINAL
Dissemination Level:	PUBLIC
Document Link:	http://dx.doi.org/10.20350/digitalCSIC/12505

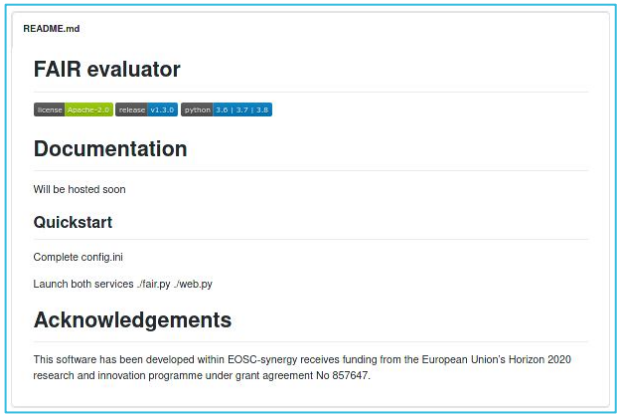
Abstract:

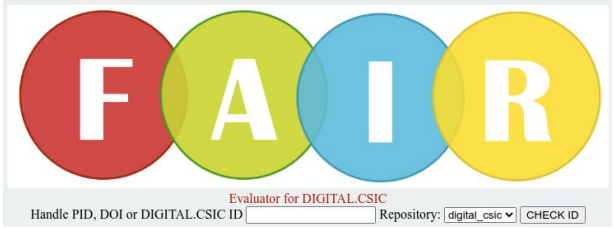
This document analyses the existing services and tools that allow issuing digital badges with the purpose of selecting the most appropriate one to be adopted by EOSC-SYNERGY in order to issue proper recognition stamps to software that complies with the Software Quality Levels (SQA) metrics defined in the project.

http://dx.doi.org/10.20350/digitalCSIC/12505

EOSC-Synergy FAIR Evaluator

- Open Source tool for FAIR evaluation of digital objects
- Oriented to:
  - researchers and repository administrators;
  - to get feedback on FAIR compliance level of research data;
  - for institutional/multidisciplinar repositories.
- Implements RDA indicators
- Open approach using:
  - JePL, CI/CD, Dockerized, OpenAPI.
- Testing to DSpace-CRIS at DIGITAL.CSIC



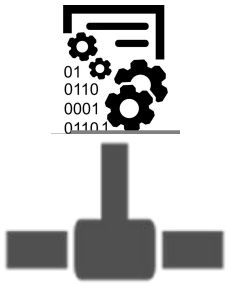


https://github.com/EOSC-synergy/FAIR_eva

SQAaaS: Integration of FAIR validation tests

Style checks
Unit tests
Integration tests

Repository
Deployment



Data
Ingestion
Test

FAIR
Verification
Tests

Quality Badge Issuance

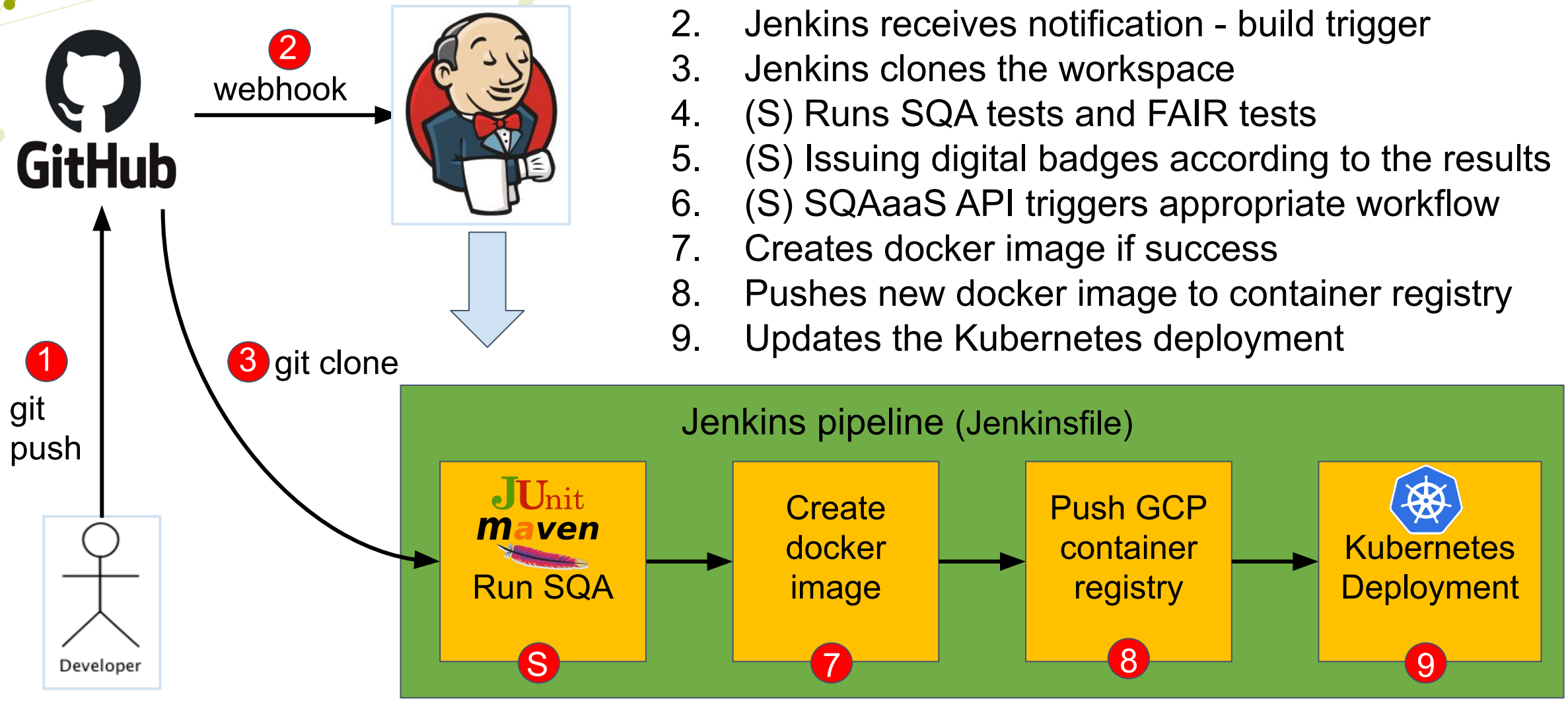
Certify
Quality
Attributes

Produce Artifacts

Produce
Release

WORSICA: quality checking pipeline

1. Developer pushes code to GitHub
2. Jenkins receives notification - build trigger
3. Jenkins clones the workspace
4. (S) Runs SQA tests and FAIR tests
5. (S) Issuing digital badges according to the results
6. (S) SQAaaS API triggers appropriate workflow
7. Creates docker image if success
8. Pushes new docker image to container registry
9. Updates the Kubernetes deployment

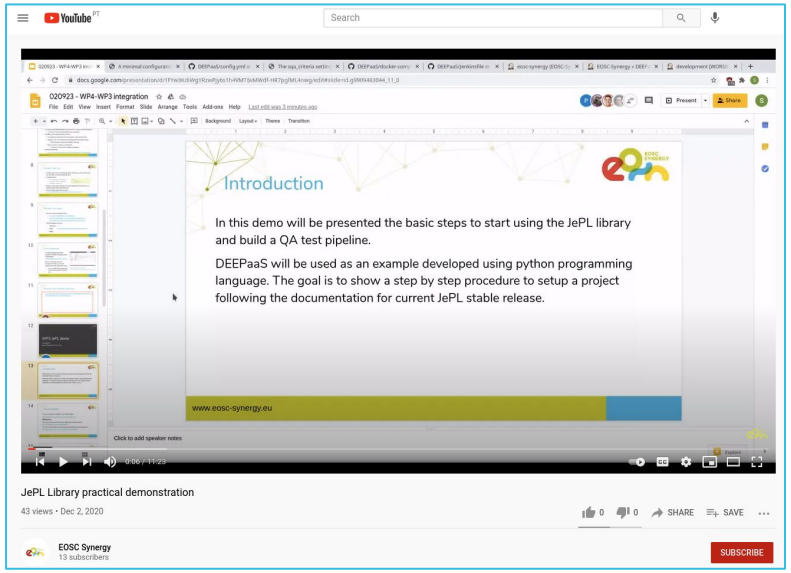


SQAaaS Benefits

- Quality assurance verification made easy
  - Simple via the SQAaaS Web
  - Advanced via JePL
- Facilitates adoption of best practices
  - Pragmatic approach to quality for **software**, **services** and **data**
- Built for automation, integrated with
  - Source code repos
  - Container engines
  - Jenkins
- From automation to quality badges
  - Build and use tailored quality pipelines
  - or use it to obtain quality certification

Thank you

For further information:



See the JePL demo in YouTube
Link in Indico

communications@eosc-synergy.eu

www.eosc-synergy.eu

FAIR conformance

- Data repository frameworks:
  - Dataverse , DSpace
- FAIR assessment
  - identify quality criteria for FAIR
  - develop FAIR verification tools
- Providing machine-actionable features
  - automated deployment of repositories
  - support for automated FAIR metadata checking
- Integrative approach, connecting the repositories with:
  - FAIR assessment tools, FAIR data point, other FAIR related software.