Service QA and Dataverse

Speaker:
Samuel Bernardo (LIP) on behalf of EOSC-Synergy

Collaborators:
Jorge Gomes (LIP), João Pina (LIP), Mário David (LIP), Ricardo Martins (LNEC), Alberto Azevedo (LNEC), Vyacheslav Tykhonov (DANS-KNAW), Pablo Orviz (CSIC), Isabel Campos (CSIC), Germán Moltó (UPV) and Miguel Caballer (UPV)

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

Outline

- **The SQA process in EOSC-Synergy**
- Worsica use case

Consortium

CSIC Consejo Superior de Investigaciones Científicas
LIP
EGI
KIT Karlsruher Institut für Technologie
PSNC
Universitat Politècnica de València
LNEC Laboratório Nacional de Engenharia Civil
bifi
INCD Infraestrutura Nacional de Computação Distribuída
FCT Fundação para a Ciência e a Tecnologia
CESGA
CYFRONET
Ciemat Centro de Investigaciones Energéticas, Medioambientales
cesnet
indra
DANS Data Archiving and Networked Services
LEGOS
BSC
Jisc

The Quality Baselines
SQA process in EOSC-Synergy

- **SQA baseline** doc is v3.2:
  - https://github.com/indigo-dc/sqa-baseline
  - https://indigo-dc.github.io/sqa-baseline/manuscript.pdf
    - doc in github
    - treated as code
    - discussions in “issues”
    - changes with PRs
    - autobuild:
      - when tag new release and pushed to “master”

- The criteria is design towards automation (DevOps) and we want to translate this into an SQAaaS.

Open

We accept contributions

http://hdl.handle.net/10261/160086

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

The Quality Baselines
SQA process in EOSC-Synergy

- **Service-QA** baseline doc is v1.0:
  - https://github.com/EOSC-synergy/service-qa-baseline
  - https://eosc-synergy.github.io/service-qa-baseline/manuscript.pdf
    - doc in github
    - treated as code
    - discussions in “issues”
    - changes with PRs
    - autobuild:
      - when tag new release and pushed to “master”

- The criteria is design towards automation and we want to translate this into an SQAaaS.

Open

We accept contributions

https://digital.csic.es/handle/10261/214441

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

Implementation of Quality Baselines

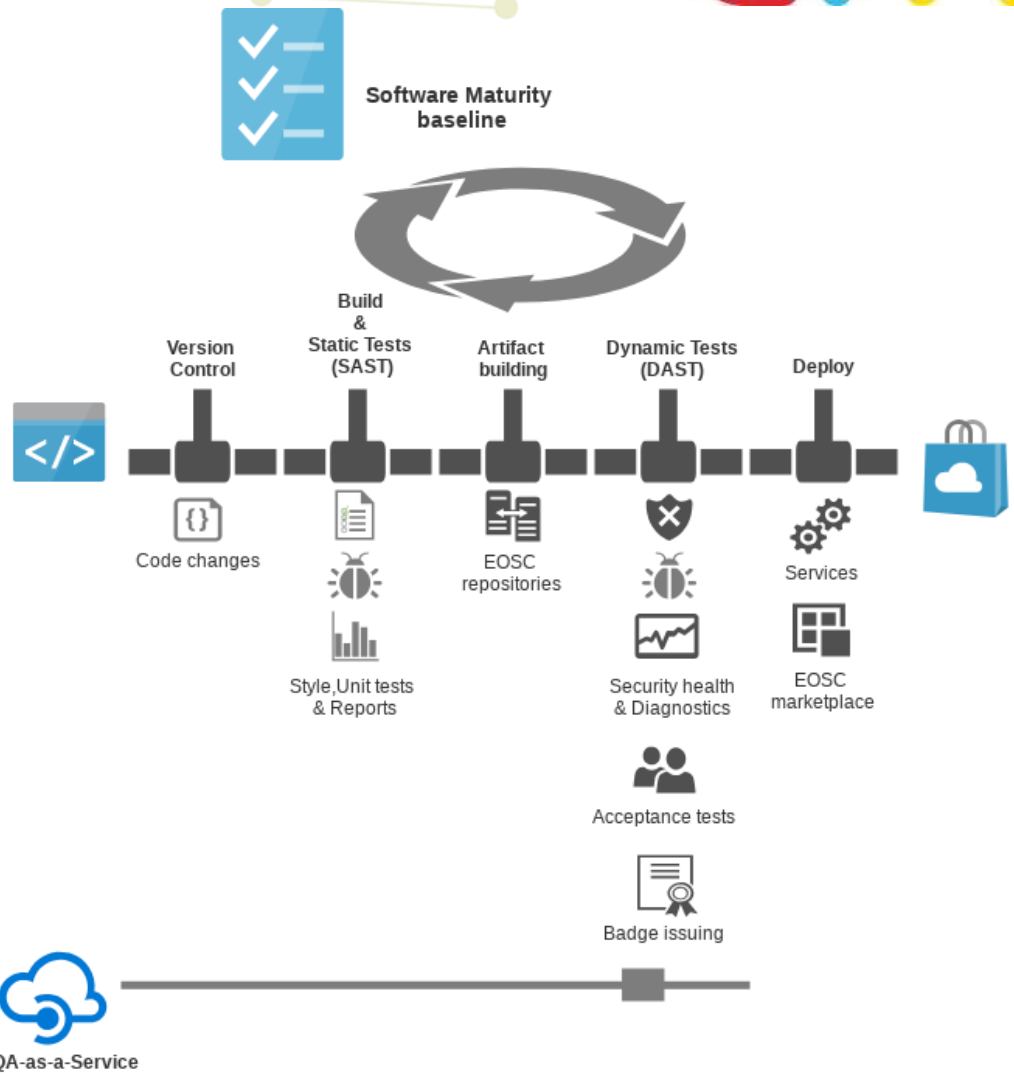
SQA process in EOSC-Synergy

Software quality: Jenkins pipelines

- Verify criteria
- Produce artefacts
- Issue badges

Service quality: extending Jenkins pipelines

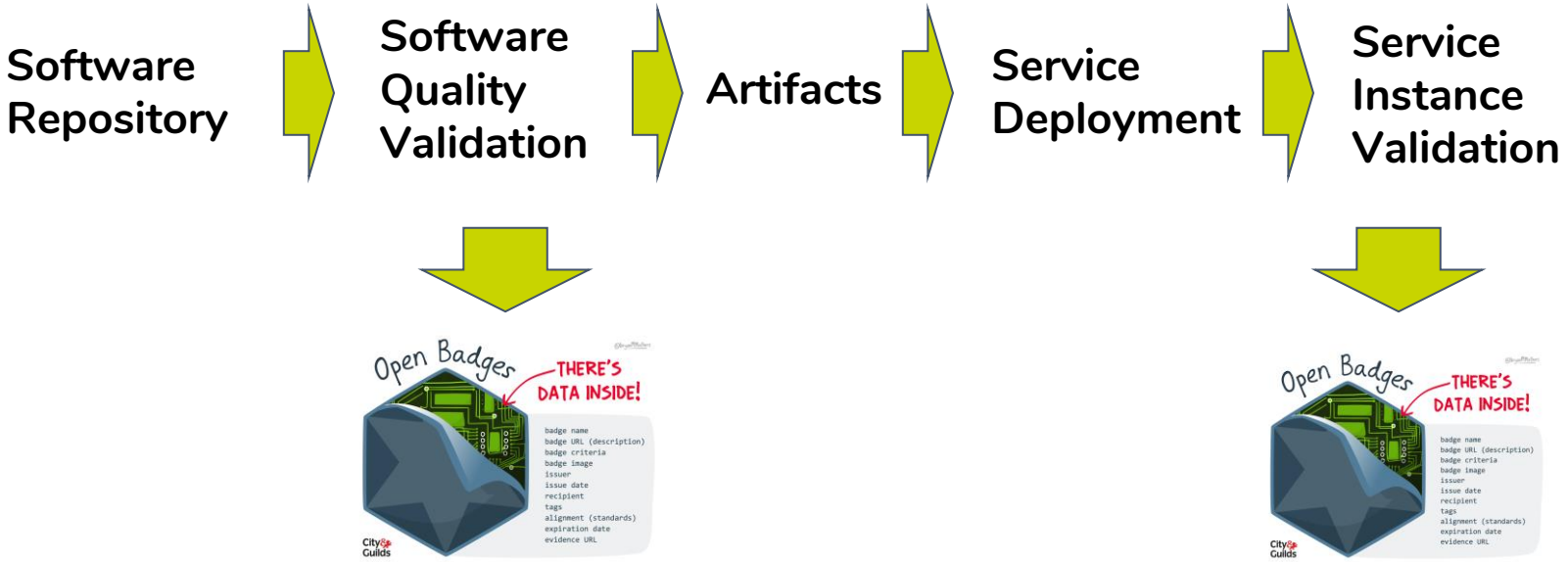
- Step beyond software quality
- Automated deployment
- Issue badges



Software Quality as a Service (SQAaaS)

SQA process in EOSC-Synergy

- **On-demand quality assessment for:** Service software repositories and Service Instance
- **Making use of:** Quality criteria and Verification mechanisms
- **Requires:** User interface and further automation

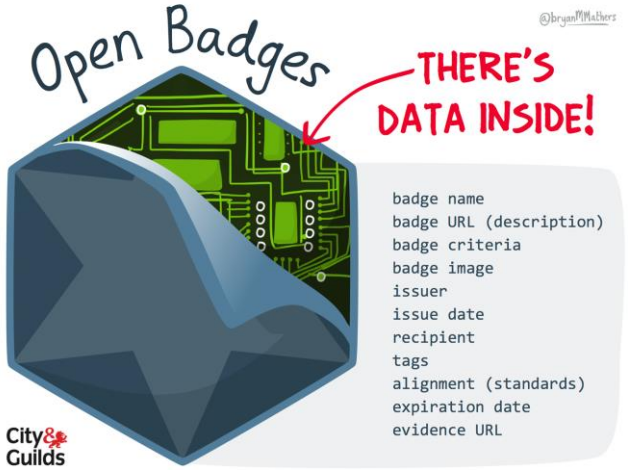


Software Quality as a Service
SQA process in EOSC-Synergy

Two main outcomes:

1. The ***Online Quality Assessment*** checks compliance of a *uniquely identified version of the source code* with regards to the quality baselines:
   a. Provide a comprehensive report (per-requirement analysis)
   b. Quality badges will be issued to recognize the achievements
2. The ***Pipeline as a Service*** compose Jenkins pipelines according to the set of software quality criteria selected by the user:
   a. Provides a library, coined jenkins-pipeline-library, to be used by the Jenkinsfiles

Digital badges: why creating?
SQA process in EOSC-Synergy



It is about Quality **recognition and trust:**

→ Engaging software developers in EOSC ecosystem

A proper recognition for software & services:

- Compliant with the quality levels defined in (EOSC)
- Issuing digital badges such that
  - can be Automatically verified,
  - Cannot be tampered,
  - Represent the achievement made by software developers and service integrators.
- Increases users trust in the software and services quality and maintenance promoting adoption

Outline

- The SQA process in EOSC-Synergy
- **Worsica use case**

Worsica use case

Worsica provides access to customized remote sensing services based on Copernicus data.

Current services

- Coastline water-land interface
- Inland water detection
- Water leak detections on irrigation networks

Worsica use case

The **developed main services** in WORSICA are:

- Web portal platform
- Processing engine

Programming language: **python**

Additional required open source services:

- **Celery** (task queue)
- **Postgis** (database)
- **RabbitMQ** (message broker)

Service QA Integration

Questions arises:

- **Test the service**
- **Deliver software** in an unknown environment
- **Data FAIRness** complaint

Service QA answer:

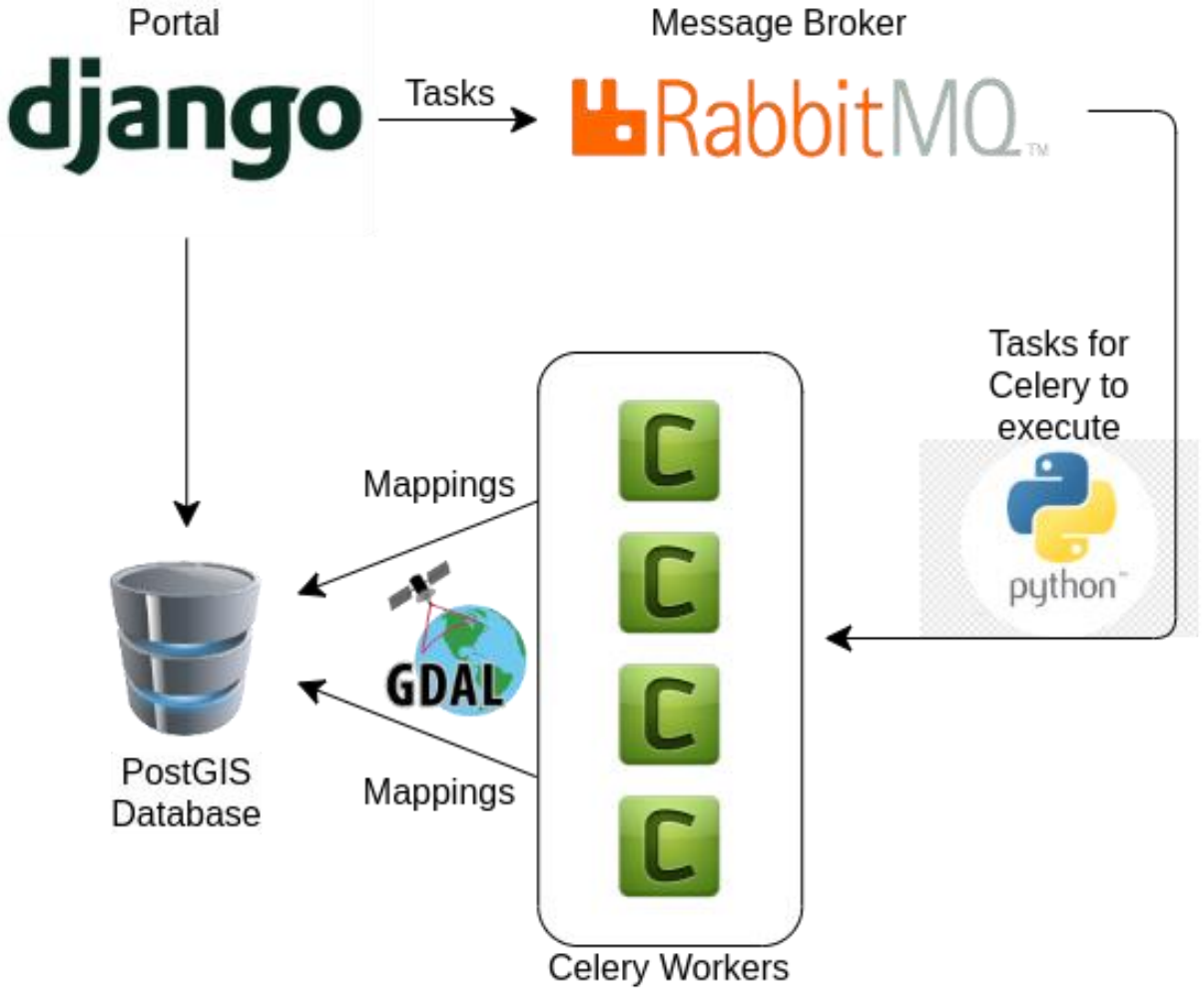
- **Tox automation tool** as a build helper to run the required tests
- Review service software and **create docker images** to automate the deployment
- **Keep data outside service** in Dataverse repository, with regular database snapshots, assuring FAIR principles

Service QA Integration

SQAaaS with Jenkins automation server:

- Require **docker compose** to deploy the service and their dependent services (multiple docker images)
- **Automated code fetching from github** repositories, selecting the branch or tag as required
- **Define the test environments** in tox for style check, unit test, coverage and security
- **Pipeline have already defined the stages** and will import for each one the required environment from tox file in expected docker container

Dataverse Repository

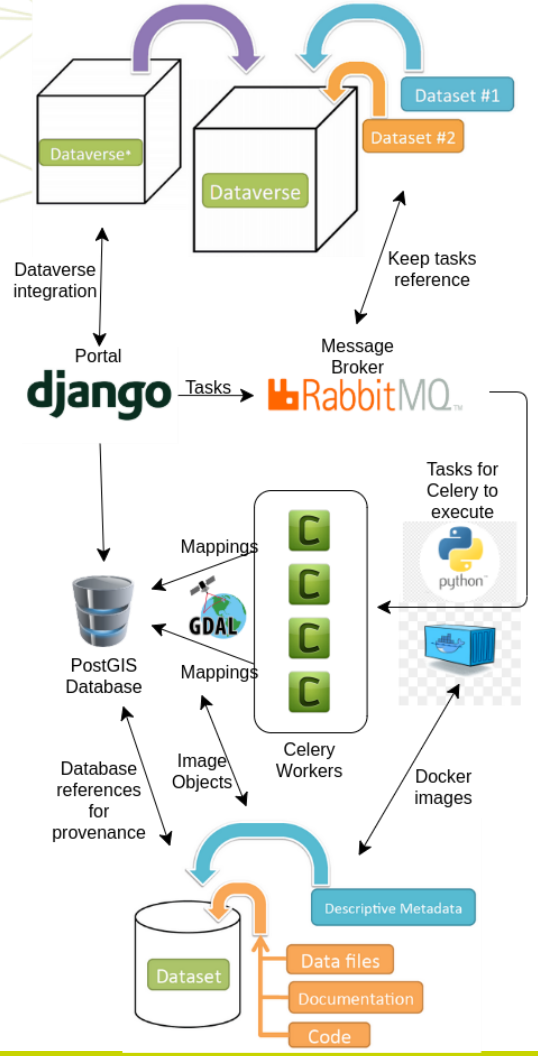


Initial service architecture:

- Data missing global unique identifier
- Data stored in multiple places internal to the services and not accessible
- Inexistent metadata detailed provenance association
- Data access not following vocabularies that apply FAIR principles

14

Dataverse Repository

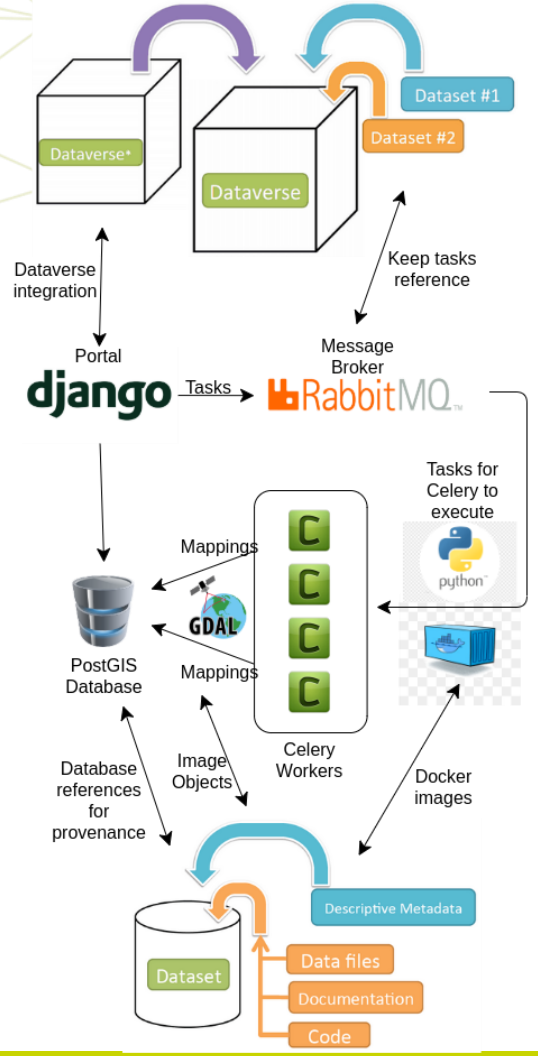


FAIR service architecture:

- Dataverse provides the repository solution that complies with the FAIR principles
- Define a dataverse and associate a persistent identifier namespace
- Associate metadata with the provided and produced data
- Use Data Commons to allow data sharing between all teams and projects
- Metadata is by default associated with CC0 Creative Commons license and publicly accessible

15

Dataverse Repository



Integrate code with Dataverse REST API:

- Very useful to implement in any language only being dependent with the provided interface without any library requirements
- Easy to maintain Worsica code in parallel with Dataverse service updates
- Current Dataverse REST API is very complete and allows to run all necessary operations
- Share sensitive data with confidence using DataTags System, that allows to use a set of security features and access requirements for file handling

16

Final remarks

Dataverse pros:

- provides a FAIR repository with a thorough REST interface
- open source software with Apache License v2.0
- allows to manage public and private data
- commons sharing along teams / projects

Dataverse cons:

- software integration for data management using Dataverse couldn’t be as quick as expected because of required learning curve
- an account and associated namespace must be acquired for a fee from a DOI or HDL provider for persistent identifiers be citable

17

Thank you

For further information:

communications@eosc-synergy.eu

www.eosc-synergy.eu