Healthcare Data Simulators

Simulating the Regional Healthcare Landscape

Contributors:

Vlad Andrei Bucur
George Edward Nechitoaia
Victor Traistaru
Ena Balatinac
Victor Kingi

Abstract Description

Digital healthcare provided by the NHS in England typically operates in silos. GPs have electronic systems to manage patient care which are distinct from hospital systems which are distinct from the ambulance service, 111, mental health services etc. Each data owner has a wealth of data that, if combined, would generate a more valuable resource than it does in isolation. While there are solutions to integrate this data for direct care purposes, there is no centralised solution to use this data to inform future care or service provisioning. This project will create a set of configurable ‘healthcare data simulators’ that generate as-live data to simulate the regional healthcare landscape.

1 OVERVIEW

1.1 PROJECT ON A LARGER SCALE

In a bigger picture, Healthcare Data Simulators is first part of 3-part solution for the currently uncentralized data system that NHS in England has. Combining the origins of data would inform clinical decision making by offering more sophisticated insights into the patient's longitudinal health on arrival and understanding the merits of previous clinical decisions taken.

1.2 CLIENTS

Our primary client is **Dr. Philip Harfield,** the Shared Data & Planning Programme Manager at NHS Healthier Together Sustainability Transformation Partnership (STP) Bristol, North Somerset and South Gloucestershire (BNSSG) & Health Informatics who is associated with Bristol Biomedical Research Centre and University of Bristol Medical School.
Our client can also be considered other 2 teams in this NHS project, especially **Heathcare Datalake Team** who will use our product to simulate data that they need in their own part.

1.3 VISION FOR THE PRODUCT SOLUTION

Our main idea for this project is building a desktop app based in java that will deploy data generation within open source programme **Synthea^{TM}** Patient Generator to generate as-live data to simulate the regional healthcare landscape. Such generated data will be forwarded into **Lyniate Rhapsody** which is used to integrate given data within complicated environment. Such processed date will then be put into **AWS Gateway API** in form of **HL7 FHIR** message. In the process **AWS Cognito** will be used to handle authentication and Split data into patient sensitive identifiers and anonymous medical data. After that data is ready to be ingested by DataLake team.

2 REQUIREMENTS

2.1 CLIENT

Our client is **Dr. Philip Harfield**, the Shared Data & Planning Programme Manager at NHS Healthier Together Sustainability Transformation Partnership (STP) Bristol, North Somerset and South Gloucestershire (BNSSG) & Health Informatics. He is also working within Bristol Biomedical Research Centre and University of Bristol Medical School.

BNSSG requires a software that can be used to perform data analytics from multiple sources with the purpose of understanding the merits of previous clinical decisions taken and more advanced insights into the longitudinal health of the patient.

2.2 USER STORY

Considering the scenario presented above the main user story can be adapted to:

*As a **Data Analyst**, I want to **get a centralised patient data system** so that **I can use this data for further analysis in order to help and inform future care or service provisioning.***

Our project is designed to simulate more than one healthcare provider category and provide source data to the Data Lake project team.

*As **the Data Lake team,** I want to **get the simulated patient data** so I can **store it in structured data marts.***

Considering the user **story**, we can divide it into the following **sequence of steps**:

1. *The user sends a request to the system to generate several healthcare provider categories.*
2. *Data is generated and split for privacy reasons.*
3. *The data is integrated in a data centralised system.*
4. *The data is ready to be provided to the Data Lake team.*

Exceptional Flow

1. *The local machine is not powerful enough to use the app due to its technical specifications.*
2. *The app freezes.*
3. *An error message pop up is displayed.*

The above steps can be breakdown into **atomic requirements** of the software:

1. *Synthea Patient Generator is configured to simulate more than one healthcare provider category (e.g. Acute hospital, primary care, 111)*
2. *The data generated is split into patient sensitive identifiers and anonymous medical data*
3. *The data is integrated to a centralised data storage using the commercial Lyniate Rhapsody.*
4. *Authenticate with AWS Cognito and use the AWS Gateway API to provide data to the Data Lake team.*

2.3 ADDITIONAL REQUIREMENTS:

1. *Simulate at least 4 modes of data transport including: Message broker technology, scheduled SFTP, HTTPS, and data federation.*
2. *Conform to healthcare standards: HL7 v2.7 and HL7 FHIR. Other non-standard data should also be simulated.*

3 PERSONAL DATA, PRIVACY, SECURITY AND ETHICS MANAGEMENT

Ethics pre-approval was applied for on 23rd November 2020 11:28.

Testing of the system will be with simulated data only, rather than real patient data but a separate approval will be required for testing the software with actual patient data. Such as separate ethics (NHS REC review as it involves patients) and governance approvals (such as Health Research Authority HRA).

We can confirm that our client is getting ethics approval on the system before actual data is processed with it. A review of our testing strategy with our client will be included.

3.1 PROJECTS THAT INVOLVE PARTICIPANTS OR USE PERSONAL DATA:

a) Only straight forward projects
b) Adult in the UK, not from vulnerable population, for example, people affected by an illness or by economic disadvantage, or people recruited from self-help groups.
c) No data of racial or ethnic origin; religious or similar beliefs; membership of a trade union, physical or mental health or condition; sexual life or criminal history.
d) Data is anonymous at collection
e) No overt or covert observation of the participants
f) Participants not tricked or deceived in any way

4 ARCHITECTURE

4.1 INTRODUCTION

We propose the design of a client application which will:

1.Generate Data in HL7 FHIR standard

2.Integrate and Centralise Data

3.Use different data transfer protocols

4.Create a safe, fast, and efficient connection with web services

4.2 GENERATE DATA

SyntheaTM

We use open source SyntheaTM Patient Generator to generate as-live data to simulate the regional healthcare landscape. Specifically, the simulators will:

• Simulate more than one healthcare provider category (e.g. Acute hospital, primary care, 111).

• Conform to health care standards: HL7 v2.7 and HL7 FHIR. Other non-standard data should also be simulated.

Synthetic patients can be simulated with models of disease progression and corresponding standards of care to produce risk-free realistic synthetic health care records at scale.

The framework for the synthetic data generation process utilized by Synthea is based on the use of PADARSER, the Publicly Available Data Approach to the Realistic Synthetic EHR.35 The PADARSER framework, unlike EMERGE25 and medGAN,27assumes that access to the real EHR is impossible or undesirable, relying instead on publicly available datasets to populate the synthetic EHR. Figure 1 presents the PADARSER framework.

HL7 FHIR Fast Healthcare Interoperability Resources (FHIR, pronounced "fire") is a standard describing data formats and elements (known as "resources") and an application programming interface (API) for exchanging electronic health records (EHR). The standard was created by the Health Level Seven International (HL7) health-care standards organization.

4.3 DATA INTEGRATION AND CENTRALISATION

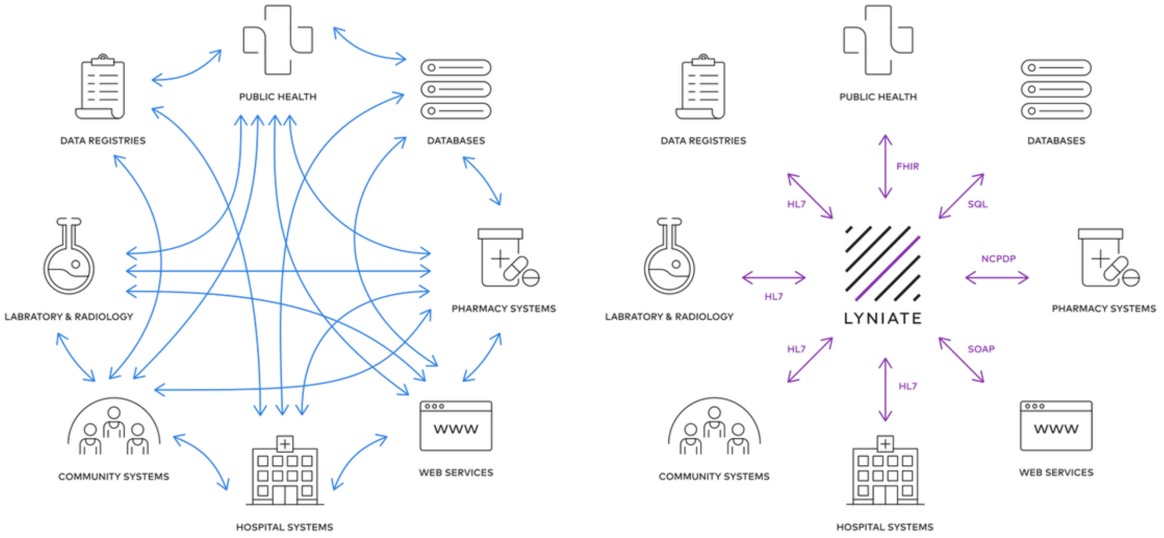
We propose to use Lyniate Rhapsody Data Centralisation and Integration Engine.

Rhapsody

Lyniate Rhapsody is a customizable, all-inclusive interoperability solution that enables reliable data integration within complex healthcare environments.

Benefits of using Rhapsody are:

- It is built for Healthcare
- It has purpose-built solution for FHIR
- It supports Data Acquisition (large amounts of data from multiple sources)



The old way

The Lyniate way

Alternatives: AWS Glue, Azure DataBricks

4.4 Data Transfer Protocols

Message broker technology is an intermediary computer program module that translates a message from the formal messaging protocol of the sender to the formal messaging protocol of the receiver. Message brokers are elements in telecommunication or computer networks where software applications communicate by exchanging formally defined messages.

SFTP works over the Secure Shell (SSH) data stream to establish a secure connection and provide organizations with a higher level of file transfer protection. This is because SFTP uses encryption algorithms to securely move data to your server and keep files unreadable during the process, and authentication prevents unauthorized file access during the operation.

HTTPS is used for secure communication over a computer network, and is widely used on the Internet. In HTTPS, the communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, Secure Sockets Layer (SSL).

Data federation technology can be used in place of a data warehouse to save the cost of creating a permanent, physical relational database. It can also be used as an enhancement to add fields or attributes that are not supported by the data warehouse application programming interface (API).

4.5 DATA INGESTION

The system will authenticate and create a RESTful endpoint for HL7 FHIR messages.

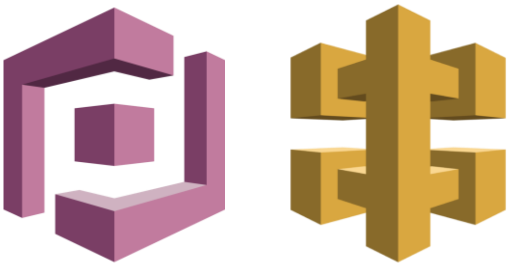
HL7 FHIR endpoint describes the technical details of a location that can be connected to for the delivery/retrieval of information. Sufficient information is required to ensure that a connection can be made securely, and appropriate data transmitted as defined by the endpoint owner.

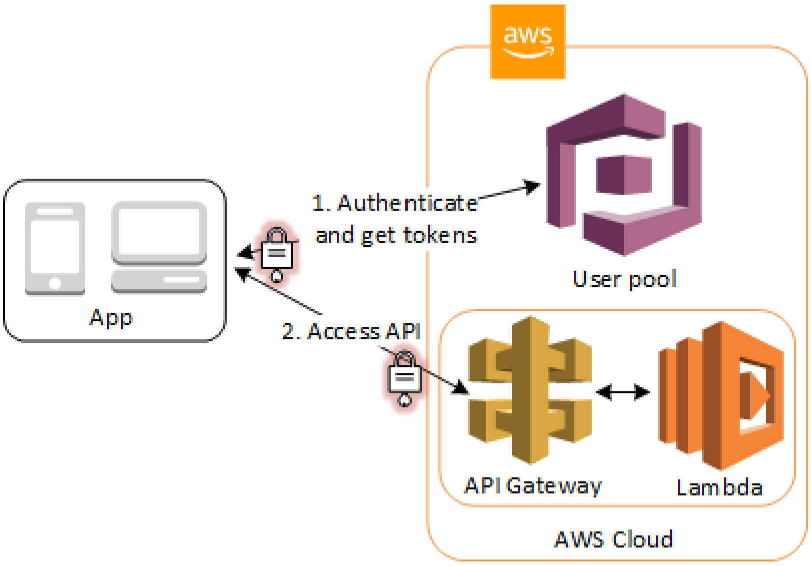
RESTful API is an architectural style for an application program interface (API) that uses HTTP requests to access and use data. That data can be used by using the CRUD approach: create, read, update, and delete.

OAuth 2.0 is the industry-standard protocol for authorization. OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows for web applications, desktop applications, mobile phones, and living room devices. This specification and its extensions are

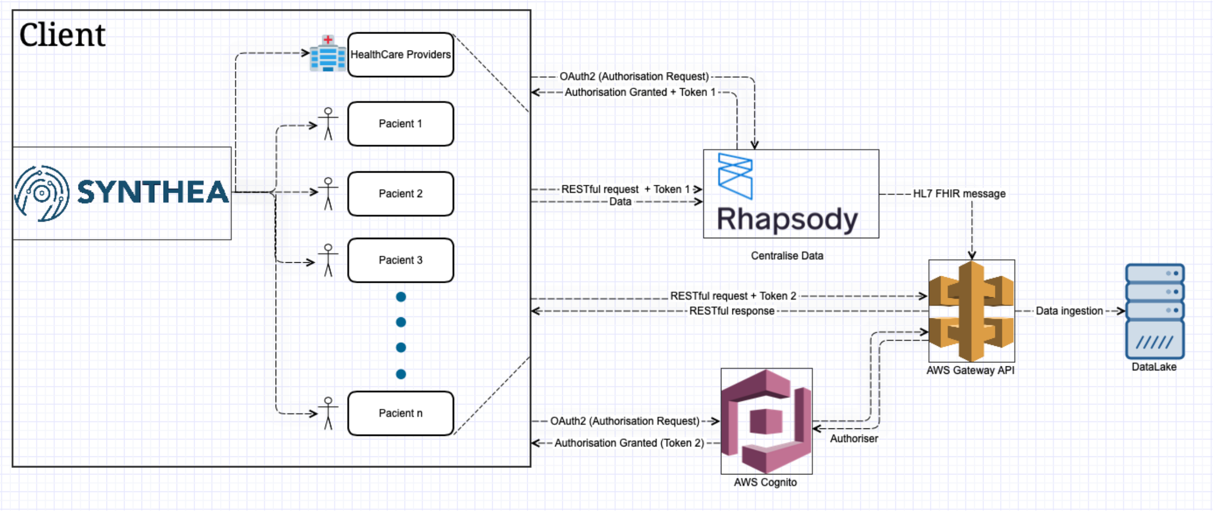
Authentication will be secured by using Amazon Cognito. The system will use a secure Token to access the API Gateway to create a safe and recognized connection with the HealthCare Lake/database infrastructure. The API Gateway will run a RESTful API and a HL7 FHIR message for ingestion into data lake. Amazon Cognito will verify the token and continue with the data transfer.

Gateway Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. Using API Gateway, RESTful APIs enables real-time two-way communication applications. API Gateway supports containerized and serverless workloads, as well as web applications.





AWS Cognito + AWS Gateway API



1st Architecture Idea Diagram

Token 1 - Authentication with Rhapsody

Token 2 - Authentication with DataLake

5 VERIFICATION AND VALIDATION- OVERVIEW

5.1 MOTIVATION

The data simulation solution developed by the "Healthcare Data Simulators" team is an integrating part of the whole prototype system proposed by the NHS. Not only does our team have to work closely with the "Data Lake" team bouncing ideas off about a shared rest API from which the "Data Lake" should be able to correctly fetch out data generated by Synthea, but we should also keep an eye on the whole system on which our colleagues from "Healthcare Analytics Team" work as well. The general purpose of these projects is to ensure a better future by making data which General Practitioners, hospitals and other health services use available all the time enabling for more precise treatment for patients in a much smaller amount of time. Another purpose is to change how data is handled and used in the healthcare system.

5.2 REQUIREMENTS

To guarantee this, apart from the actual development of the java desktop application that is supposed to envisage the data, we need a form of verification and validation. As we cannot work with live patient data due to the legislation surrounding sensitive patient data, we are going to use **Synthea**, a **Synthetic Patient Population Simulator engine**. Furthermore, our system is going to be used in compliance with the *NHS Digital GDPR compliance* and the liability for the personal data stored falls onto the respective primary stakeholder, Philip Harfield at Bristol, North Somerset and South Gloucestershire CCG (BNSSG). Our team developing the data simulation is responsible for creating a robust, defect free application generating data under the **HL7 FHIR** (Fast Healthcare Interoperability Resources – pronounced "fire") standards. **FHIR** provides an alternative to document-centric approaches by directly exposing discrete data elements as services. The idea of working with synthetic, realistic (but not real data) arises from the fact that the identity of the patients must not be uncovered, preserving the security of this private information.

5.3 TEST DRIVEN DEVELOPMENT (TDD)

We will adopt a **Test-Driven Development** strategy. This way we can focus on the mentioned requirements before writing the actual code. In our case the main features are spinning up a simulation of data, being able to generate different categories of health services for each patient, different treats of the patients, health history, more instances such as Primary care, Physio, Foster care etc. Finally, we need to make sure we correctly push data to the Data Lake team.

We ought to create tests based on these requirements having in view use cases and user stories. The users that come directly into use with our system are the colleagues from the

Data Lake team. They should be the first ones to test and integrate our product in their project.

At the moment we are still bouncing ideas off regarding what specific testing strategies we should follow, but the main ideas to focus on are Dependency Injection, working with objects that track calls by the tested code and raise exceptions if conditions are not met, checking the results for errors or anomalies when working with methodical, synthetic data, selecting test data for testing edge case and boundaries and defining an equivalence partitioning (in our case this could mean generating people of the same age, region, or treats – we should make sure on the other hand not to work with the same people).

5.4 Unit Testing

For the **Unit Testing** approach, we ought to use **JUnit** which allows us to create loops to generate tests for a big number of values. Furthermore, Junit is useful because it allows decoupling between the implementation and the testing files. "JUnit is a Java library for testing source code, which has advanced to the de-facto standard in unit testing. By using JUnit, you can assert that methods in your Java code work as designed, without the need to set up the complete application." [1]

5.5 Java Code Coverage

We could use **JaCoCo (Java Code Coverage**). This open source coverage technology allows simple use and integration with existing build scripts and tools, regression tests with full functional test coverage based on JUnit test cases or lightweight implementation with minimal dependencies on external libraries and system resources.[2]

5.6 Release (Integration) Testing

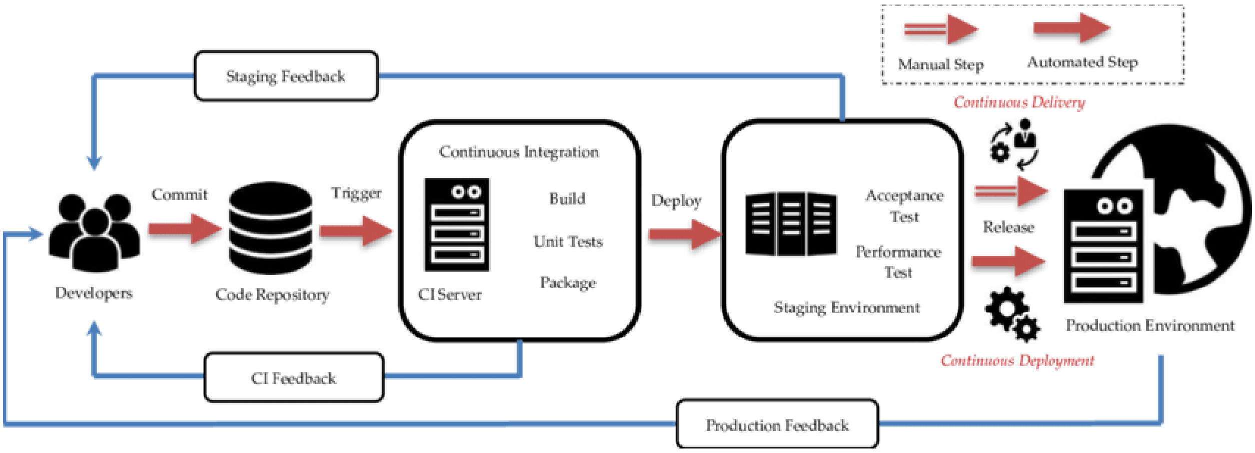
Lastly, we will test the robustness of our product relaying on **Release (Integration) Testing.** Release testing checks that the application fulfils the requirements and meets the expectations of the future usages. During this stage we would also test non-functional requirements. In our case these could be checking if our application handles well large amounts of data, or if it works error prone when ran on different OS or environments. Another check that should be done is related to the ability of dealing with large number of requests. Overall, this step means releasing the system as an integrated whole and is higher

[1] http://www.methodsandtools.com/tools/tools.php?junit
[2] https://www.jacoco.org/jacoco/trunk/doc/mission.html

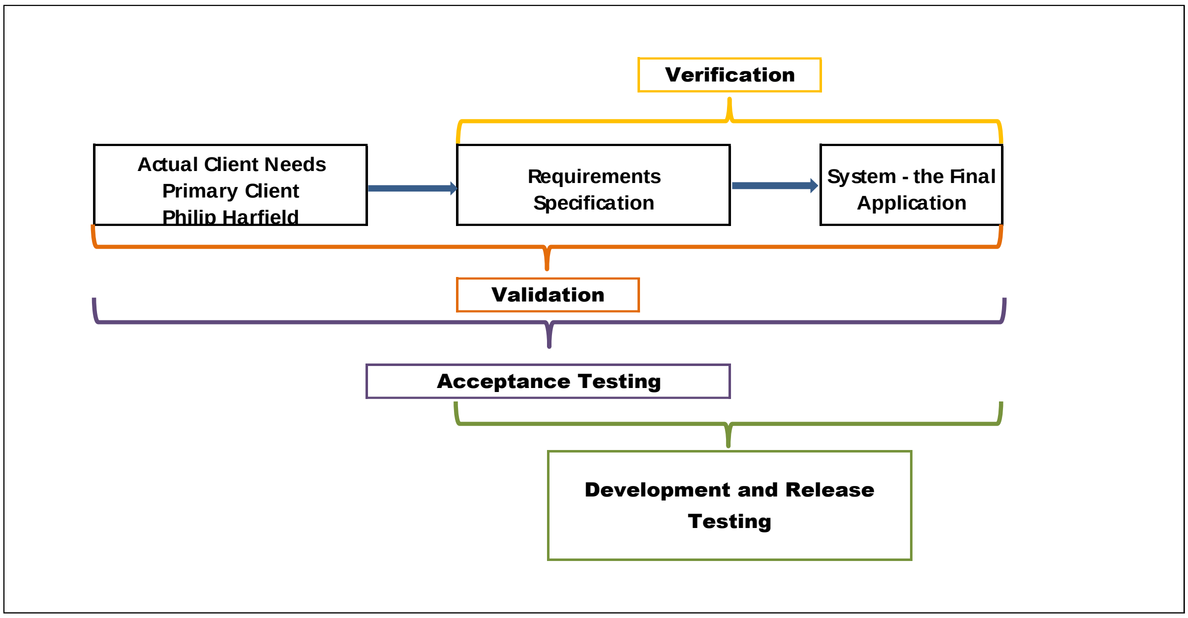
level than unit testing where we would test against bugs and most likely try to refactor code after base requirements are met.

5.7 Continuous Integration

Generally, we are keen on using **Continuous Integration**. “Continuous integration (CI) is a practice where a team of developers integrate their code early and often to the main branch or code repository. The goal is to reduce the risk of seeing “integration hell” by waiting for the end of a project or a sprint to merge the work of all developers. One of the primary benefits of adopting CI is that it will save you time during your development cycle by identifying and addressing conflicts early. It’s also a great way to reduce the amount of time spent on fixing bugs and regression by putting more emphasis on having a good test suite. Finally, it helps share a better understanding of the codebase and the features that you’re developing for your customers.”[3]



[3] https://www.atlassian.com/continuous-delivery/continuous-integration/how-to-get- to-continuous-integration


Verification
Actual Client Needs
Primary Client
Philip Harfield
Requirements Specification
System - the Final Application
Validation
Acceptance Testing
Development and Release Testing

6 REFERENCES

https://www.digitalocean.com/community/tutorials/an-introduction-to-oauth-2

https://www.lyniate.com/rhapsody/

https://synthetichealth.github.io/synthea/

https://restfulapi.net

https://aws.amazon.com/cognito/

https://aws.amazon.com/api-gateway/

https://www.hl7.org/fhir/

https://en.wikipedia.org/wiki/Message_broker

https://en.wikipedia.org/wiki/HTTPS

https://searchdatamanagement.techtarget.com/definition/data-federation-technology

http://www.methodsandtools.com/tools/tools.php?junit

https://www.jacoco.org/jacoco/trunk/doc/mission.html

https://www.atlassian.com/continuous-delivery/continuous-integration/how-to-get- to-continuous-integration