Healthcare Data Lake

Kendal, Joseph
University of Bristol
jk17246@bristol.ac.uk

Sherred, Jago
University of Bristol
j.sherred.2019@bristol.ac.uk

Benson, Luke
University of Bristol
wr19606@bristol.ac.uk

Liu, Anna
University of Bristol
gf19916@bristol.ac.uk

Cismaru, Armand
University of Bristol
fz19792@bristol.ac.uk

November 9, 2020

Abstract

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 is designed to explore the benefits of cloud technologies to produce a prototype secure, scalable health data storage platform that can underpin local healthcare analytics.

1 Overview

1.1 Client

1.2 Domain

1.3 Project

1.4 Vision

The Healthcare Data Lake Project is envisioning a future integrated data storage solution, one that is scalable and portable, while using the latest cloud-based technologies. Starting with a prototype, the final scope is to create, alongside the Simulation and Analytics Projects, a system that is going to change how data is handled and used in the healthcare system. There is the real possibility that this three projects will represent the cornerstone of a future solution used and developed extensively by the NHS, which will bring immediate aid to the average medical worker and improve the quality of the service provided to the patients.

2 Stakeholder and Requirements Analysis

2.1 Primary Stakeholder and User Story

Philip Harfield at Bristol, North Somerset and South Gloucestershire CCG (BNSSG). Philip Harfield is our client, and this software is being developed for him at BNSSG. The user story for this stakeholder is that BNSSG require a piece of software that can be used to allow long-term healthcare data analytics from multiple data sources to inform clinical and strategic decisions. The reason for this is understanding the longitudinal health of a patient allows understanding of the merits of previous clinical decisions taken. In addition it can be used to inform strategic commissioning decisions using data on how effective services offered to patients have been.

2.1.1 Flow steps

Considering the user story above, we can breakdown the story into a sequence of steps of user flow.

1. Local healthcare services provide data to the data lake.
2. Data is transformed and loaded into the data lake.
3. The data is catalogued in the data lake and stored in structured data marts as required.
4. The data lake allows access to a data analytics environment.
5. An analytics environment can run queries on the data lake and report results to analytics environment.
6. Clinical and strategic decisions can be taken based on analysis.

We can also identify alternative or exceptional flows.
Exceptional Flow:

1. Local healthcare services provide data to the data lake.
2. Data is not provided in an acceptable format to the API.
3. The healthcare service receives an error message to provide data in standard format.

2.1.2 Atomic Requirements

We can breakdown these steps into atomic requirements of the software.

1. An API takes in data from local healthcare providers as a HL7 FHIR message. We have chosen HL7 FHIR as it is the UK standard for transferring healthcare messages.
2. These data messages are stored in a cloud solution in a well structured data model.
3. Ingested data is catalogued.
4. An ETL tool is used to curate data marts.
5. These data marts can be queried by the analytics environment.

There are also additional requirements specified for the software:

1. Medical data is to be stored independently from pseudonymised patient identifiers.
2. Provide a user console to monitor automated ETL jobs.
3. Provide full audit trails.

2.2 Additional stakeholders and User Stories

This software will provide services to a number of local healthcare organisations such as NHS trusts and the Healthier Together STP and such all these additional users are secondary stakeholders to this project. These organisations will need to be able to provide healthcare information to the software which will need to be able to load and store the data for future analysis.

2.2.1 Flow Steps

Considering the user story above, we can breakdown the story into a sequence of steps of user flow.

1. Healthcare provider (e.g. Hospital Trust) provide data to the data lake by a HL7 FHIR message.
2. The data is stored in the data lake.

3 Personal Data, Privacy, Security and Ethics Management

3.1 GDPR

3.2 Security

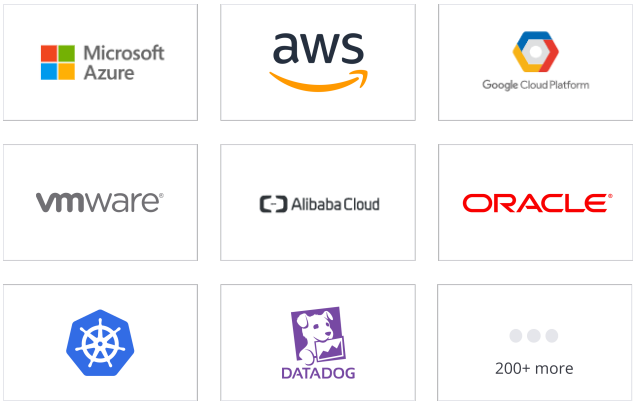
3.3 Ethics

4 Architecture

4.1 Introduction

We propose a modular, (cloud)platform-independent solution that offers high scalability and performance at a low cost for maintenance, development and deployment. The key to achieving this is leveraging the practises of infrastructure-as-code (IaC), serverless architectures and open standards. Therefore, development expense is focused on providing the most value, security and flexibility to the user.

IaC By building infrastructure through extensible configuration files, we make it easy to build, test, secure, update and rollback versions of architecture which can combine multiple cloud or on-premise services. Terraform is an IaC framework that supports all major cloud providers in addition to self-hosted options such as Kubernetes. This makes it a popular choice for a modern infrastructure team that seeks to avoid vendor lock-in and easily protect the security of it with robust tests and auditing.



Serverless Serverless computing is a cloud computing execution model in which the cloud provider runs the server, and dynamically manages the allocation of machine resources. Pricing is based on the actual amount of resources consumed by an application, rather than on pre-purchased units of capacity [1]. The benefits of this approach are the huge reduction in infrastructure and development expense. Engineers can focus on shipping their microservices and have secure, scalable infrastructure taken care of for them.

The Serverless Framework is also an example of IaC and provides developers with the ability to develop and deploy their serverless application to different cloud providers or simply Kubernetes (using **Knative**) if they wanted to run it on-premise or avoid code exposure to a cloud native service such as AWS Lambda or Google Cloud Functions.

This also provides quicker developer on-boarding and flexibility as a broad array of popular languages are supported. Given that the Function as a Service (`FaaS`) model is centred around the

principles of loosely coupled microservices, this enables easy migration out of serverless architecture to a VM or container-based deployment. This may make sense in scenarios where there are cost savings to be made which typically applies to long-running, memory intensive tasks as opposed to short-lived and resource-light stateless services.

Provider	Supported Languages
AWS Lambda	Python, Java, Go, Node, .NET, Ruby or Custom runtime
Google Cloud Functions	Node, Python, Go and Java
Azure Functions	.NET, Node, Java, PowerShell and Python
IBM Cloud Functions	Node, Python, Swift, PHP, Go, Ruby, Java and .NET
Cloudflare Workers	JavaScript, TypeScript or Wasm-supported

Open standards Open Standards allow people to share all kinds of data freely and with perfect fidelity. They prevent lock-in and other artificial barriers to interoperability, and promote choice between vendors and technology solutions. [2]

OpenAPI The OpenAPI Specification (OAS) defines a standard, language-agnostic interface to RESTful APIs which allows both humans and computers to discover and understand the capabilities of the service without access to source code, documentation, or through network traffic inspection. When properly defined, a consumer can understand and interact with the remote service with a minimal amount of implementation logic. An OpenAPI definition can then be used by documentation generation tools to display the API, code generation tools to generate servers and clients in various programming languages, testing tools, and many other use cases. [3]

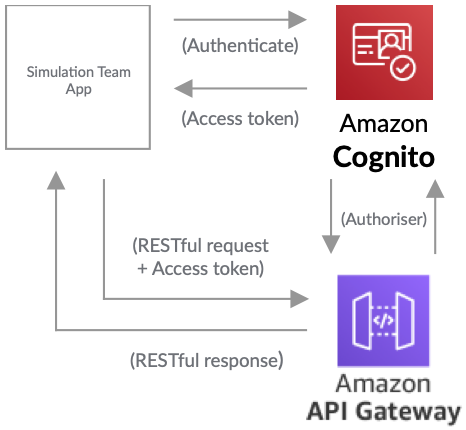
HL7 FHIR *Fast Healthcare Interoperability Resources* (`FHIR`, pronounced "fire") is a standard describing data formats and elements (known as "resources") and an [API] for exchanging electronic health records (`EHR`). The standard was created by the Health Level Seven International (`HL7`) health-care standards organization. One of its goals is to facilitate interoperation between legacy health care systems, to make it easy to provide health care information to health care providers and individuals on a wide variety of devices from computers to tablets to cell phones, and to allow third-party application developers to provide medical applications which can be easily integrated into existing systems. [4]

4.2 Data ingestion

4.2.1 API management

Authentication

Gateway



Example: AWS API Gateway

4.2.2 Microservices

Example: AWS Lambda

4.2.3 Metadata cataloguing

Glue

4.2.4 Schema-less object store

S3

4.3 Data warehousing

4.3.1 Federated querying

4.3.2 Common models

4.4 ETL & data marts

4.4.1 Scheduling

4.4.2 Developers

4.4.3 Console

4.5 Secure access

4.5.1 Encryption

4.5.2 IAM

4.5.3 Logging

5 Development Testing

5.1 Local environment

5.1.1 API development

Serverless The Serverless Framework enables developers to run an emulated environment locally instead of needing to deploy that serverless app for testing. This comes with over 1,000 plugins that can emulate services such as AWS Lambda and API Gateway.

Postman Postman is a very popular tool used by developers that want to conveniently test and build APIs in a collaborative way.

As we are implementing the OpenAPI specification, Postman enables the import of Swagger files to automatically generate a Postman environment. This makes developer on-boarding much easier as versioned mock endpoints and documentation are hosted for us as part of the DevOps deployment.

5.2 DevOps pipeline

5.2.1 GitHub

5.2.2 CircleCI

5.2.3 CodeDeploy

6 Release Testing

6.1 Staging

6.2 Production

7 OO Design & UML

References

[1] Ron Miller. *AWS Lambda Makes Serverless Applications A Reality*. TechCrunch. 24. URL: https://techcrunch.com/2015/11/24/aws-lamda-makes-serverless-applications-a-reality/.

[2] *Overview of Open Standards*. Free Software Foundation Europe. URL: https://fsfe.org/freesoftware/standards/def.en.html.

[3] *OpenAPI Specification*. Version 3.0.3. URL: https://swagger.io/specification/.

[4] *Fast Healthcare Interoperability Resources*. Wikipedia Foundation. Nov. 2016. URL: https://en.wikipedia.org/wiki/Fast_Healthcare_Interoperability_Resources.