Hospital Quality Management Distributed Artifact Repository

SE Project MTech IT

Nupur Garg MT2014081

Setu Patani MT2014085

Shaily Sanghvi MT2014102

Dhruvik Shah MT2014110

Problem Statement

- Building a cloud-based repository for healthcare system.
- Repository should be able to receive data (HL7 Messages) from various health care devices and store it in repository.
- On the request by various devices or analytic application, provide appropriate data in Json or XML format.
- The technology choice includes Hadoop, NoSQL frameworks for scale and volume.

Approach

- To address the problem, we need to understand the medical records and its structure. Basic knowledge about the medical records and its interpretation is required. After evaluating various ways to represent medical records, we decided to use HL7 messages to represent medical data.
- To store these HL7 messages, after evaluating multiple database and properties, we found MongoDB best suitable option for our needs. It is NoSQL. Already MongoDB is preferred in health care sector for storage as the data is not structured.
- As we are building repository, as of now all the work done is cloud based. We have used Amazon Web Services (AWS) for the same. All the work has been deployed on AWS.
- As to deal with multiple incoming health messages, we have implemented queueing mechanism, Simple Queueing Service (SQS).

Following are the details of each parameters.

HL 7:

HL7 Messages are used to transfer electronic data between disparate healthcare systems. Each HL7 message sends information about a particular event such as a patient admission. An HL7 message consists of one or more segments. Each segment consists of one or more composites, also known as fields.

http://www.interfaceware.com/blog/understanding-hl7-messages/

Each HL7 message is of a particular message type. This HL7 message type indicates what health-related information is being provided in this message. The message type also determines what segments can be included as part of the message.

To determine the message type of an HL7 message, examine its MSH segment. The message type is normally the ninth field of this segment. For example, consider this MSH segment, which you have seen before:

```
MSH|^~\&|EPIC|EPICADT|SMS|SMSADT|199912271408|CHARRIS|ADT^A04|1817457|D|2.5|
```

Here, the HL7 message type is ADT^A04, which is “Register a Patient”.

http://www.interfaceware.com/blog/determining-the-hl7-message-type/

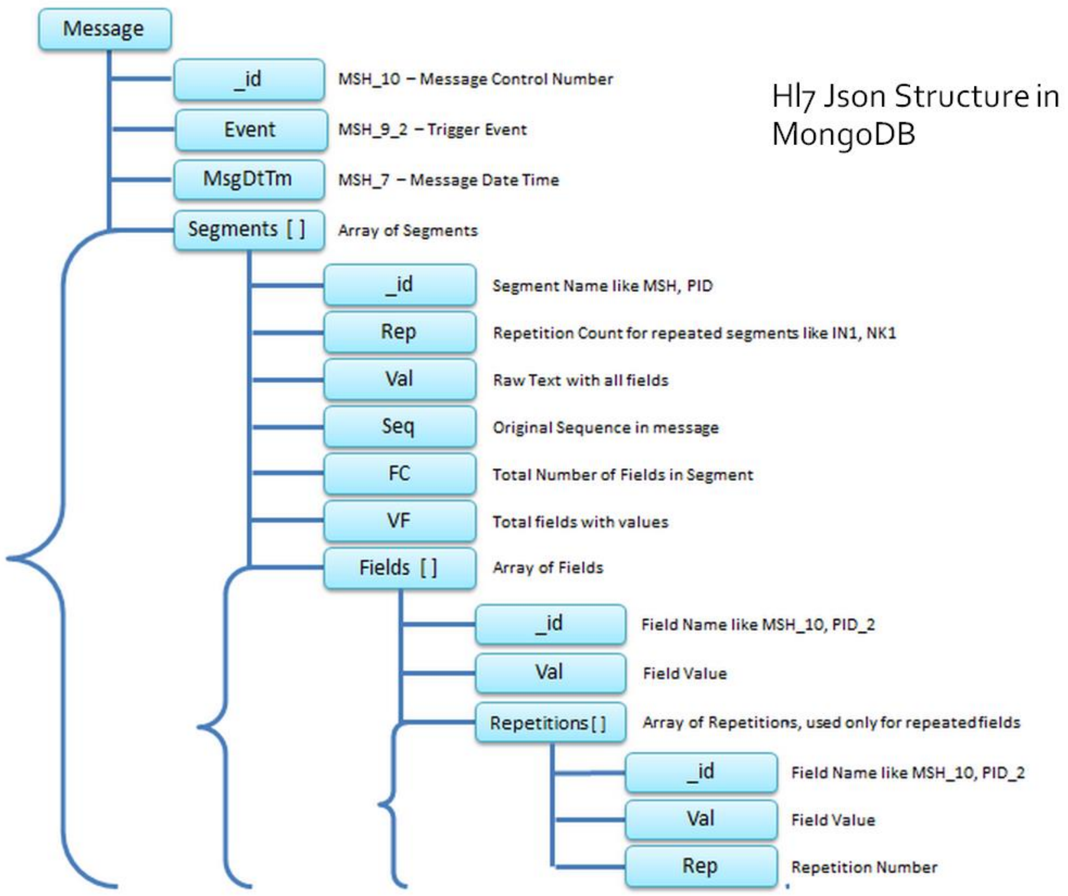
MongoDB:

MongoDB (from humongous) is a cross-platform document-oriented database. Classified as a NoSQL database, MongoDB eschews the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas (MongoDB calls the format BSON), making the integration of data in certain types of applications easier and faster. Released under a combination of the GNU Affero General Public License and the Apache License, MongoDB is free and open-source software.

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

Companies in the healthcare industry are finding themselves hamstrung by rising costs, changing regulations and complex technological demands. Traditional systems are not only expensive but also poorly-equipped to handle the industry’s twenty-first century needs for scale, cost efficiency and flexibility. Companies in the healthcare industry – including insurers, hospitals and solutions providers – are building applications on MongoDB to reduce costs, meet compliance standards and improve healthcare outcomes.

https://www.mongodb.com/industries/healthcare


Hl7 Json Structure in MongoDB
Message
_id
MSH_10 – Message Control Number
Event
MSH_9_2 – Trigger Event
MsgDtTm
MSH_7 – Message Date Time
Segments [ ]
Array of Segments
_id
Segment Name like MSH, PID
Rep
Repetition Count for repeated segments like IN1, NK1
Val
Raw Text with all fields
Seq
Original Sequence in message
FC
Total Number of Fields in Segment
VF
Total fields with values
Fields [ ]
Array of Fields
_id
Field Name like MSH_10, PID_2
Val
Field Value
Repetitions [ ]
Array of Repetitions, used only for repeated fields
_id
Field Name like MSH_10, PID_2
Val
Field Value
Rep
Repetition Number

Amazon Web Services (AWS):

After evaluating multiple cloud service providers, we decided to go with the AWS. Following are some of the factors that made impact while selecting service provider,

- AWS has been prioritizing cloud data security over the years using the federal approach to security drills along with firm adherence to internationally recognized standards and protocols. These best practices ensure that information is protected in a systematic manner keeping everyone on the same page, irrespective of the geographic location.

- Elastic: Amazon EC2 Cloud enables user to increase or decrease capacity within minutes. One can commission one, two, three or even thousands of server instances simultaneously. Application can automatically scale itself up and down depending on its needs.

- Cost Effective: Amazon AWS account is free, and you only pay for what you use. It's much simpler, and more affordable, than buying and installing your own gear.

- Flexible: We can change the server size within one minute. This allows us to scale-up to a beefier specification with more memory, disk space or more processors if required, or downgrade as they see fit.

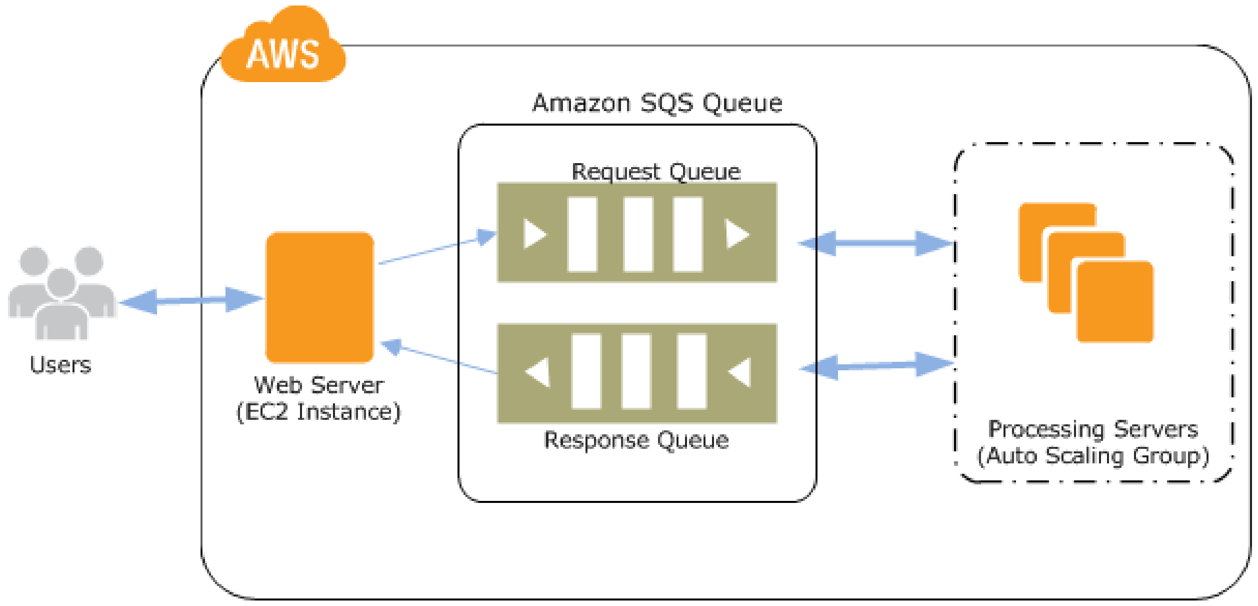
- Reliable: Amazon EC2 offers a highly reliable environment where replacement servers can be rapidly and predictably commissioned. The Amazon EC2 Service Level Agreement commitment is 99.95% availability for each Amazon EC2 Region.

- Secure: The AWS cloud security infrastructure has been architected to be one of the most flexible and secure cloud computing environments available today. It provides an extremely scalable, highly reliable platform that enables customers to deploy applications and data quickly and securely.

Simple Queue Service:

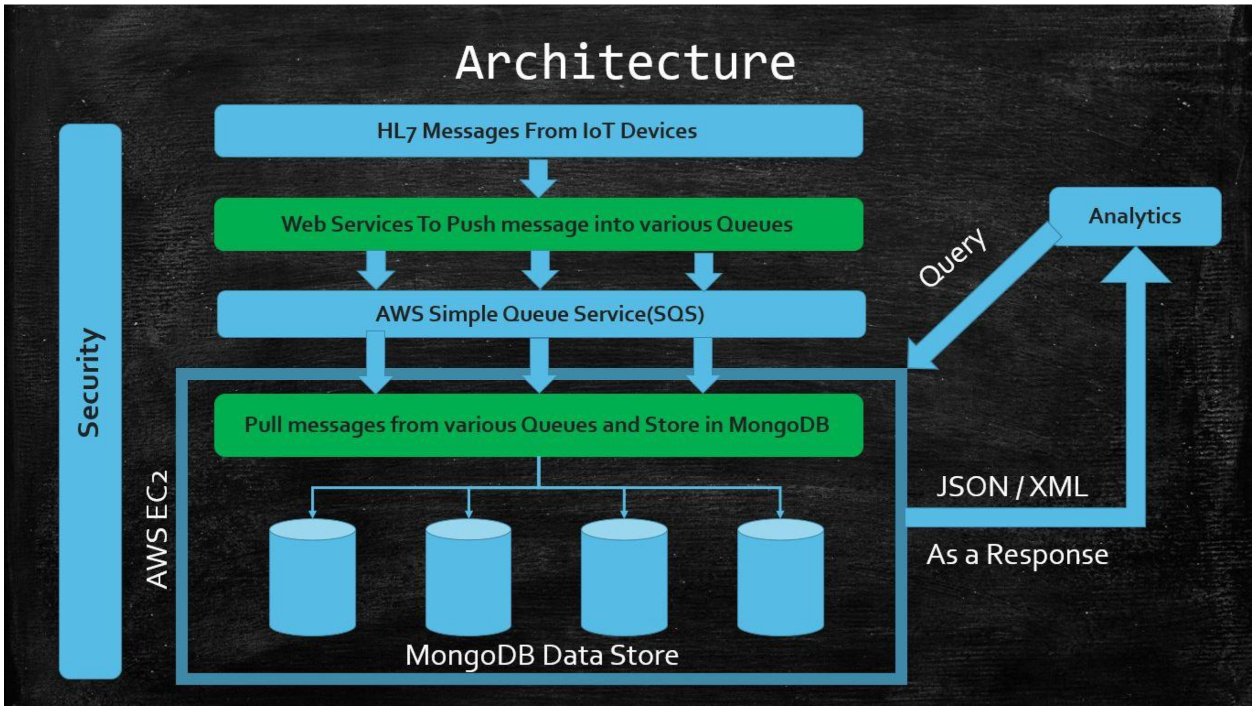
To deal with multiple incoming messages, we implemented SQS. SQS handles the queueing of messages and stores the messages that are yet to be processed.

Amazon Simple Queue Service (SQS) is a fast, reliable, scalable, fully managed message queuing service. SQS makes it simple and cost-effective to decouple the components of a cloud application. We can use SQS to transmit any volume of data, at any level of throughput, without losing messages or requiring other services to be always available.



Flow of Project

Architecture


Architecture
HL7 Messages From IoT Devices
Web Services To Push message into various Queues
AWS Simple Queue Service(SQS)
Pull messages from various Queues and Store in MongoDB
MongoDB Data Store
AWS EC2
Security
Analytics
Query
JSON / XML
As a Response

- First HL7 messages are sent by various devices.
- Those messages will be dealt by web services that pushes various messages to various queues on SQS based on type of the message.
    - This enables to differentiate among various type of messages before sending/pushing it to the queue. By doing this at this stage, we reduce the work and make it simplify for further level to manage various messages.
- Next will be SQS. This stores the messages sent by web services. This layer takes input from web service and provides input to next layer.
- Now Application will be running on the server that fetches the data from various queues and store in respective MongoDB storage.
    - By putting such kind of application makes this architecture **flexible**. Here we can modify how we store data based on our needs. To do so, we just need to make application store where want it to get stored, and application will do this for us.
    - By using such layer we can also store data based on its type, properties etc. in different kind of databases. All these can be taken care by application. So this kind of runtime changes can be made without affecting the rest of the architecture and its functions.
- At last we store data into database. As of now MongoDB is used. This architecture is capable to handle other database and also combination of multiple databases in single application.