A black and white logo

Description automatically generated with medium confidence

**Distributed data research**

OFS Platform

By Mohammed Al Harbi

Contents

[Introduction 3](#_Toc121292344)

[What and why distributed data storage? 3](#_Toc121292345)

[Microservices data 4](#_Toc121292346)

[Data processing and requirements 6](#_Toc121292347)

[Data storages 7](#_Toc121292348)

[Conclusion 8](#_Toc121292349)

# Introduction

In this research document we will get an idea regard the definition distributed data and explain why we need it in OFS. Afterwards, we will have a look at what each microservice need to store and what the data processing are, with this information we should have the context to be able to list the data requirements for OFS system. Finally, we will look at various data storages which help us achieve the defined data requirements.

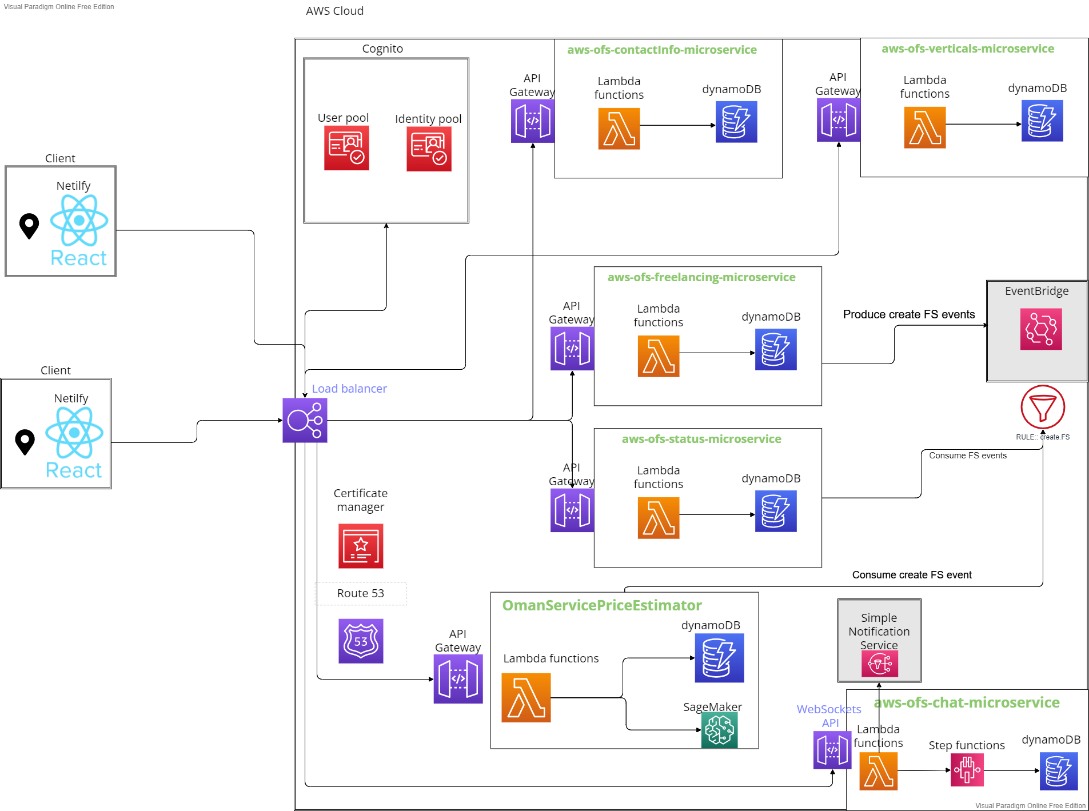
# What and why distributed data storage?

Distributed data is essentially having different data files in 2 or more differently located databases that way, combining all the data files result in a full overview data.

Back in time, monolithic applications from data storage perspective were very simple; one standard relational data base which contains all data. With the introduction of microservices things got a little more complex. Essentially separating concerns in the architectural level allowed for many benefits such as growing the developer team, allowing for multitude of programming languages but it came with challenges. One of those challenges are handling of data. Having multiple microservices which could be written in totally different programming languages using different ORMS (Object relational mappings) frameworks or pure SQL statements required for multiple databases as having one database for all microservices simply will not help business take advantages of actual microservices benefit which lean more towards anonymous teams work; therefore as a first step in microservices approach the databases should distributed to be one database per microservice.

# Microservices data

Oman freelancing services have multiple microservices find below screenshot:



* + - 1. AWS-OFS-VERTICAL-MICROSERVICE: Data attributes
* ID: Unique string, clarifies the id of the vertical in the system.
* Name: String, clarifies the name of the vertical uniquely.
  + - 1. AWS-OFS-FREELANCING-MICROSERVICE: Data attributes
* ID: Unique string, clarifies the id of the freelancing in the system.
* FS: JSON payload, clarifies the name of the vertical uniquely.

Note: The JSON payload contains the following attributes:

* + Vertical ID: Unique string, clarifies the id of the vertical in the system.
  + Name: String, clarifies the name of the freelancing service uniquely.
  + Address: String, allocate where the service address at.
  + Description: String gives a clear overview about the service and any additional remarks/information. (Can be used for AI NLP with SageMaker through data preparation)
  + Price: Integers, represents the price a freelancer offers for their services. (Can be used for AI NLP Evaluation)
    - 1. AWS-OFS-STATUS-MICROSERVICE: Data attributes
         * Fs\_id: Unique string, clarifies the id of the freelancing in the system.
         * scheduledTime: JSON payload, it contains the working days and the working hours. We can call it to indicate the availability of a service at any point.

Note: The JSON payload contains the following attributes:

* + WorkingDays: Array of strings, shows what the days of the week in which a service is available.
  + WorkingTimes: Array of strings, shows what the times of the day in which a service is available.
    - 1. AWS-OFS-ContactInfo-MICROSERVICE: Data attributes
         * Fs\_id: Unique string, clarifies the id of the freelancing in the system.
         * ContactInfo: JSON payload, it contains the working days and the working hours. We can call it to indicate the availability of a service at any point.

Note: The JSON payload contains the following attributes:

* + UserID: string, shows who the user given (user data is stored in Cognito which I do not need to manage.)
  + phoneNumber: String, shows the phone number of a user
    - 1. AWS-OFS-priceEstimator-MICROSERVICE: Data attributes
         * Fs\_id: Unique string, clarifies the id of the freelancing in the system.
         * FS\_info: JSON payload, it contains the description and freelancing price.

Note: The JSON payload contains the following attributes:

* + Description: String, gives overview about the freelancing service
  + price: Integer, gives the price of a freelancing service.
    - 1. AWS-OFS-Chat-MICROSERVICE: Data attributes
         * ID: Unique string, clarifies the id of the chat message in the system.
         * SenderID: unique string ID that matches users ID in cognito.
         * ReceiverID: unique string ID that receives messages using users ID in cognito.
         * Message: string, contains the chat message.

# Data processing and requirements

As for data processing model given the previous context a distributed data structure should be more than enough. I do not think that I should look for in memory databases since there is no need usage of real-time data processing.

However, the approach in which the data gets created is different than usual API call means create 1 entity, I first forward events to evenBridge and then those will be routed to a SQS Queue which parse those events and perform creates 500 resources in databases at a time. This allows more efficiency when scaling as requests do not need to wait databases to scale if auto scaling is configured.

Data requirements are the following:

* NoSQL database that allows to save JSON objects ideally since we do not really use SQL features.
* The database needs to be **available**; users need to be able to create freelancing services and read freelancing services all the time.
* The database needs to be **Partition Tolerance**; users need to be able get a response even if some distributed data instances are down.
* The database needs to support autoscaling.
* Security encryption

# Data storages

Below is a list of NoSQL databases with their key features as this is one of the data requirements that have been defined:

|  |  |
| --- | --- |
| Database | Key features |
| MongoDB | * real-time analytics * Indexing appropriately for better query executions * Replication for better data availability and stability * Load balancing * Supported intuitively by AWS cloud provider of my choice |
| Dynamodb | * Performance at scale * DynamoDB supports both key-value; have a flexible schema, so each row can have any number of columns at any point in time. * Automated global replication with global tables * Microsecond latency with DynamoDB Accelerator * Supported intuitively by AWS cloud provider of my choice * Data security & encryption |
| Apache Cassandra | * High Scalability. * Fast Writes. * Fault Tolerant. * Cassandra Query Language. * Tunable Consistency. |
| OrientDB | * Incremental backups. * Unmatched security. * 24x7 Support. * Query Profiler. * Distributed Clustering configuration. * Metrics Recording. * Live Monitor with configurable alerts. * Support graphs * OpenSource |
| RavenDB | * Graphical User Interface (GUI) * Voron Storage Engine. * Database Security. * ETL Replication and Pull Replication. * Memory Management System. |

# Conclusion

In this research document we saw the shift from standard monolithic architecture to microservices architecture came with challenges from data perspective and the need for distributed data. We look at how the data get processed in OFS and what each microservice needed the base schema to look like. Looking at the data requirement NoSQL database is a must since I want to be flexible with JSON objects I would like to save, may add a field or remove one as I see fit. Data encryption is another important point so hackers cannot benefit from data in those microservices, and user data security is handled by Cognito. I have selected DynamoDB for all microservices except for the PricingEstimator microservices as I will performing machine learning activities using NLP and so MongoDB is optimized for real-analytic data which is a better option in this case than DynamoDB.