Incubation Lab – Star Project Simulation

**Business Requirement:**

A multinational company wants to create a central storage repository (i.e. Data Lake) that will hold big data in raw as well as processed format. This data will be further consumed by different users for a variety of business use cases. As a data engineer, you need to create an end-to-end ETL pipeline that will first ingest raw data from the landing zone to a raw zone in an as-is format. Then, the data will be processed and moved to a staging zone. The processing will include masking and cast transformation of some fields. The name of the fields that will be transformed will be read from a configuration file. The pipeline should be triggered on the upload of raw data in the landing zone. Finally, create a lookup dataset for unmasked and masked data with SCD2 type implementation.

Stores transformed data

Stores raw data with restricted access

Stores raw data

**Staging Zone**

**Raw Zone**

**Landing Zone**

**Getting Ready:**

**You should be ready with the following:**

* AWS – User Account
* Schema to prepare raw data for ingestion
* Access to [draw.io](https://www.draw.io/)
* Please go through this link- [What is Data Lake?](https://www.talend.com/resources/what-is-data-lake/)

**Milestone 1:**

**Goal:** Understand the **business requirement** and create a high-level **architecture** on AWS

You need to **perform** the following **steps**:

* Please go through the business requirement
* Identify the appropriate AWS services
* Understand the data schema
* Create a high-level architectural diagram using [draw.io](https://www.draw.io/)
* The suggested solution should be scalable and cost-effective

**Architecture Diagram:**

Diagram

Description automatically generated

**Milestone 2:**

**Goal**: Create **S3** buckets for **Data Lake**

You need to **perform** the following **steps**:

* Create three **S3** buckets for landing zone, raw zone, and staging zone.
* Enable appropriate **life-cycle management** on these buckets.
* Read & Write access for raw bucket should be limited to a service account for programmatic access only. This is for GDPR compliance. Use **IAM**
* Enable **versioning** on these buckets.
* **Prepare sample data** in parquet format by referring to the schema.

**Milestone 3:**

**Goal:** Createan **EMR-Spark** Job to perform data transformation

You need to **perform** the following **steps**:

* Create a **spark job** using the programming language you learned.
* This spark job will read data from the raw zone and after the transformation put the data in the staging zone.
* Implement the logic to mask critical fields
* Implement the logic to cast some fields.
* Create an **EMR cluster** that will be used to submit this spark job.
* Spark configuration should be **configurable.** The configuration should be read from a file.
* Manually **execute** the spark job.
* The EMR cluster should terminate after the job execution is done. Use cost-saving measures.
* Please refer [to EMR Tutorial](https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-gs.html)

**Milestone 4:**

**Goal:** Implement **Livy** to interact with an **EMR** cluster over a **REST interface**

You need to **perform** the following **steps**:

* Understand the purpose and usage of Livy
* Please go through this [What is Livy?](https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-livy.html)
* Implement Livy to interact with the EMR cluster to easily submit the spark job

**Milestone 5:**

**Goal:** Setup **Airflow** on **EC2**

You need to **perform** the following **steps**:

* Create an **EC2** machine in the appropriate region and availability zone
* Please check [Airflow](–%20https:/airflow.apache.org/) documentation for the details
* Please check the minimum hardware requirement for airflow installation
* Install Apache **Airflow** on EC2.
* Test the installation with a sample workflow.

**Milestone 6:**

**Goal:** Createa high-level **DAG** in **Airflow**

You need to **perform** the following **steps**:

* Create a DAG which will include the following steps:

1. Read config file
2. Copy raw data as it is from the landing zone and copy to raw zone
3. Pre-validation
4. Submit EMR-Spark with Livy (to transform)
5. Copy transformed data to a staging zone
6. Post-validation

* Validation implementation to be done in Milestone 8.
* Note: There will be a total of two configuration files. One for storing spark configuration and the other one will store masking column names.

**Milestone 7:**

**Goal:** Create **Lambda** function to **trigger DAG**

You need to **perform** the following **steps**:

* Write a lambda function in the programming language you learned
* This lambda function will be triggered as soon as raw data is uploaded to the landing zone
* Assign an appropriate IAM role to this lambda function.
* This lambda function will trigger the Airflow DAG

**Milestone 8:**

**Goal:** Improve DAG by adding **validations & lookup data set**

You need to **perform** the following **steps**:

* Implement pre-validation and post validation steps.
* Perform following validations - Data availability check, count check, data type check
* Create a lookup dataset for unmasked and masked data with SCD2 type implementation.

“We have to maintain the SCD2 implementation only for PII columns i.e., advertising\_id as a lookup store.

For example, if we are receiving the advertising\_id as 123 upon masking it will be transformed as #fj#11a, we have to create a store with these two columns, and if tomorrow if we are receiving the same value 123 then it doesn't have to be updated on the store, since we already maintaining its masked data. Only for the new advertising\_id which has to be captured on the lookup store.”

**Milestone 9:**

**Goal**: Implement a basic **deployment pipeline**

You need to **perform** the following **steps**:

* Code check-in should automatically deploy the latest code.

**Milestone 10:**

**Goal**: Peer review of code, perform unit testing, and document the results

You need to **perform** the following **steps**:

* Interchange the code and perform peer review
* End to End Black box testing
* Document the results
* Note down the defects

**Possible Extensions:**

* Data Security. Identify PII information and means to encrypt/secure this data. Either create separate clear data Vs masked data or think about dynamic data masking.
* Data Ingestion from RDBMS, file systems. Can be an enhancement to this.
* Performance Optimization - Try Spark program optimizations – different join types for high volume data handling, partitioning, etc. At least few datasets should be high volume  ! 1M++
* Logging – How to enable CloudWatch logs for different applications, how to monitor etc.
* Getting comfortable with Unix commands and some basic scripting is important. So maybe some housekeeping programs can be tried out with Unix scripts
* Basic CloudOps – IAM, permissions, role, etc. This can be a demo session

**Reference Document:**



**Schema Reference:**

**Datasets**

1. **Actives:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Names** | **Data Type** | **PII column?** | **Partition Columns** | **Transformations** |
| advertising\_id | string | Yes | No | None |
| city | string | No | No |  |
| location\_category | string | No | No |  |
| location\_granularities | string | No | No |  |
| location\_source | array[string] | No | No | Convert to a comma-separated string |
| state | string | No | No |  |
| timestamp | bigint | No | No |  |
| user\_id | string | Yes | No |  |
| user\_latitude | double | No | No | convert to decimal with 7 precision |
| user\_ longitude | double | No | No | convert to decimal with 7 precision |
| month | string | No | Yes |  |
| date | date | No | Yes |  |

1. **Viewership:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Column Names** | **Data Type** | **PII Column?** | **Partition Column** | **Transformations** |
| advertising\_id | string | Yes | No |  |
| channel\_genre | string | No | No |  |
| channel\_name | string | No | No |  |
| city | string | No | No |  |
| device | string | No | No |  |
| device\_type | string | No | No |  |
| duration | integer | No | No |  |
| grid\_id | string | No | No |  |
| language | string | No | No |  |
| location\_category | string | No | No |  |
| location\_granularities | string | No | No |  |
| location\_source | Array[String] | No | No | Convert to a comma-separated string |
| record\_timestamp | bigint | No | No |  |
| show\_genre | string | No | No |  |
| show\_name | string | No | No |  |
| state | string | No | No |  |
| user\_lat | double | No | No | Convert to decimal with 7 precision |
| user\_long | double | No | No | Convert to decimal with 7 precision |
| month | string | No | Yes |  |
| date | date | No | Yes |  |