**Use case document**

**17-09-2024:**

* **Research about company** – what ever company mentioned in profile, do research about that company. Is it a product based – what kind of product the company have , service based - what type of services it provides.
  + What is the company size, is it multiregional company, where it is located(US/UK based). What other locations the company have.(use LinkedIn or any job portal to research about company)
* **Different Domains** – identify which kind of company it is, healthcare, finance, retail, telecom, transport, banking.
* **Below information for banking domain**

Account Management: Users can view their current accounts, savings accounts, and transaction history.

Loans: Information on personal loans, home loans, and auto loans, including the ability to apply and check the status.

Credit Cards: Users can manage their credit cards, view balances, make payments, and check rewards or cashback options.

Mortgages: Options for mortgage products, mortgage calculators, and application processes.

Investments: Services related to investment accounts, stocks, bonds, and mutual funds.

Insurance Products: Information on various insurance products like home, auto, and life insurance.

Payment Services: Features like transferring money, setting up direct debits, and peer-to-peer payment options.

Financial Tools: Budgeting tools, financial insights, and spending analysis.

Customer Support: Access to chat support, FAQs, and other assistance options.

Understanding Banking Products and Domain Knowledge

Product Types: It's essential to know the different banking products available (e.g., loans, credit cards, mortgages) and how they function.

Financial Terms: Familiarity with terms like APR (Annual Percentage Rate), LTV (Loan-to-Value), and credit score is crucial.

Data Sources for Data Engineers

As a data engineer, you might gather data from various sources:

Legacy Systems: Older systems that contain historical customer data, transaction records, and loan histories.

Mobile and Web Applications: Data generated when customers interact with banking apps, such as login events, transaction data, and usage patterns.

APIs: Integration with third-party services or internal systems to gather real-time data, such as credit scoring services.

Databases: Structured data from relational databases (e.g., customer records, account details) and unstructured data from sources like logs and documents.

Data Warehouses: Centralized repositories where data from multiple sources is stored, allowing for analytics and reporting.

Example Use Case: Credit Risk Modeling

In building a credit risk model:

Data Collection: Historical data from legacy systems about previous loan performance, customer demographics, and behaviors.

Feature Engineering: Creating features based on customer location, income, credit history, and other relevant attributes.

Modeling: Using statistical methods or machine learning algorithms to predict the likelihood of a new customer defaulting on a loan.

**ETL – ELT:**

* In case of huge datasets will first extract and load it into Datawarehouse(ELT)
* The medallion architecture fits with ELT. Once you extract the data, will load the entire raw data without any transformation in bronze folder and then will do some basic transformation and load into silver layer and as per the requirements (file format)by stakeholders and end user will place in gold layer.
* ETL is suitable when you know what you have to do with the data, when your use case is defined, when business logic is given to you – will get data directly apply the business logic and load the data
* ELT is when you don’t know the business logic, as a data engineer bring the data into data lake first that where the injection layer created.
  + Bring the batch data as it is into raw layer, generally we have the data scientists, data analysts who will explore the data they find the use case then they will give the business logic. Then we will read data from raw layer apply the transformation logic and load the data into curated layer and further more will go to gold layer that can be utilized by data scientists, business analysts and application team based on the requirements.
* Data Injection : we have 3 parts in data injection
  + Batch processing
  + Real-time
  + Near real time
* **Batch processing** refers to the processing of large volumes of data in **chunks** or **batches**, usually at scheduled intervals (e.g., daily, hourly, or even weekly).ex : super market -Tesco is busy until 11pm it will be bsy so we don’t disturb the existing system- so will schedule daily job after 11(everyday will get the new data)
* **Real-time :** 
  + Real-time (or stream) processing refers to the continuous processing of data as it arrives in the system.
  + The data is processed immediately upon arrival, often with very low latency. Ex: flight status, stock market, credit card deduction(fraud deduction).
* Near real time : if it delayed more than 30sec or 1min upto 10min will consider near real time
* Atleast we need 2 use case – one for batch and one for real time
* **Batch processing :**
  + **Source:** RDBMS, files(xml, pdf, csv) location for files : (web scraping, Staging node, sftp, one drive)
  + **Extraction tools:** ADF(azure data factory), AWS DMS(database migration service), NIFI(Data Flow), Sqoop(lazy systems, google dataflow)
    - Copy command – redshift
    - Dist cp (HDFS)- To migrate data form one cluster to another
    - Aws data sync
* Once you extract the data will follow the medallion architecture
  + **Raw(Bronze)** : will keep data into raw layer. Will extract the data as it is and will bring data into bronze layer.
  + Before doing transformation we need to mention about the tool that you used for analytical purpose. (someone need to analyze data in source)
    - Analysis tools on BRaw layer(Athena with Glue crawler), Delta table in data brick.
  + Transformation (ETL tools): Glue, EMR, Data bricks, data proc, Spark. Once you do the ETL will load data into silver layer(curated layer)
  + Once the cleaned data is in curated layer, we have data warehousing gold layer for that we have Redshift, azure synaps analytics, Big Query, Snowflake, Hive(on-prem cluster)
  + Finally we have dashboard – BI tools, tableau, powerBI, aws - Quick sight.
* **Simple use case** 
  + Source postgres-> aws : dms-> glue -> load data into : redshift
* **CICD:**
  + CI - Github, bitbucket(version Control)
    - The first part is version control – we continuously integrate your code whatever changes we made whatever we do this is your github and big bucket
  + CD - Jenkins, git action, codepipeline, azure devops
  + Testing: within CICD testing is important - Unit test, integration test, regression test, load test
* **Then we need monitoring** : Cloudwatch, azure monitor
* **Orchestration**: Airflow, google composer, step function , adf ( for example we have so many jobs how you manage scheduling which job to run when for that we have airflow …)
* **Event Driven tools**: Lamda, azure function
* Sample usecase :

Use Case: ATM Optimization and Credit Risk Modeling

Objective: To optimize ATM locations based on usage patterns and enhance credit risk assessment for loan applications.

Data Sources

RDBMS (PostgreSQL):

Contains transactional data related to ATM usage, customer transactions, and historical loan performance.

File Storage:

Files are uploaded to a shared location (e.g., OneDrive or an internal file share) containing external data such as demographic information and economic indicators that may influence ATM usage and credit risk.

Data Ingestion and Processing Pipeline

Data Migration:

AWS Database Migration Service (DMS): Migrates data from PostgreSQL to a raw data layer in S3, preserving historical data.

File Ingestion:

AWS Lambda: Automatically triggers when new files are uploaded to the shared location, moving these files to the S3 raw data layer.

Data Storage Architecture

Raw Layer: All ingested data from PostgreSQL and external files is stored in a raw format in S3.

Data Transformation

ETL Process:

AWS Glue: Used for transforming data from the raw layer to a curated layer. This may include:

Aggregating ATM usage data by location.

Merging demographic data to analyze potential usage patterns.

Heavy Transformations:

Amazon EMR: If complex transformations are needed (e.g., large-scale joins or advanced analytics), EMR processes the data and loads it into the curated layer.

Curated Layer

The curated layer in S3 stores refined data ready for analysis, including:

Aggregated ATM usage statistics.

Enhanced customer profiles for credit risk assessment.

Data Warehouse

Amazon Redshift: The curated data is loaded into Redshift for structured querying and analysis, providing a robust environment for business intelligence.

Analytics and Modeling

Descriptive Analytics:

Tableau: Business analysts use Tableau to create interactive dashboards and visualizations based on the data stored in Redshift. This allows stakeholders to explore ATM usage trends, identify high and low usage ATMs, and make informed decisions regarding ATM placements.

Credit Risk Modeling:

AWS SageMaker: Data scientists access the curated layer data to build and train machine learning models for credit risk assessment. The models leverage:

Historical loan performance.

Customer demographic data.

Economic indicators.

Deployment and Monitoring

Model Deployment:

Amazon EKS (Elastic Kubernetes Service): Models are containerized using Docker and stored in Amazon ECR (Elastic Container Registry). They are deployed to EKS for real-time scoring of loan applications.

Monitoring and Alerts:

Amazon CloudWatch: Monitors the pipeline, sending alerts for any failures in data ingestion or processing. SQS (Simple Queue Service) can be used to manage messages between different components of the architecture, ensuring smooth data flow and processing.

CI/CD for Deployment

Continuous Integration and Deployment:

AWS CodePipeline: Implement CI/CD pipelines that automate the deployment of ETL jobs, model updates, and dashboard updates.

CodeCommit: Store the code for ETL processes and machine learning models, triggering builds and deployments on code changes.

CloudFormation: Use Infrastructure as Code (IaC) to manage and provision AWS resources consistently, ensuring that environments are reproducible and scalable.

Real Time

18-09-2024:

* Real time pipeline will include with the credit responder - when data scientist develop the model then will get the live stream data. When someone submit for the credit request, will get data using api. We need to choose kenisis, msk. Will get the live stream. We can use spark datastream for real time and then the data load will put into dynamo db. Dynamo db provides low latency, fast read fast write, scalability and then that will be fed into model for prediction and then we have the api gateway – api created and that will read with mobile development team and also will store data into s3 for historical use (for data scientist to rebuld the model)

**Use Case: Real-Time Credit Risk Assessment**

Objective: To provide real-time credit risk assessment for loan applications as customers submit their requests, ensuring timely and accurate decision-making.

Data Sources

API for Credit Requests:

Customers submit their credit applications through a mobile app or web interface, generating real-time request data via an API.

Data Ingestion and Processing Pipeline

Real-Time Data Ingestion:

Amazon MSK (Managed Streaming for Apache Kafka): Captures real-time credit application requests submitted via the API, streaming this data for further processing.

Real-Time Transformation:

Apache Flink or Spark Streaming: Processes the streaming data in real-time, transforming it as necessary (e.g., filtering, enriching with external data, or aggregating).

Data Storage

DynamoDB:

The transformed data is stored in DynamoDB, which provides low-latency access and high scalability for real-time applications.

Model Interaction

Machine Learning Model:

The real-time data in DynamoDB is fed into a pre-trained credit risk model developed by the data science team using AWS SageMaker.

The data science team manages the deployment and operationalization of the model using Docker and Kubernetes on Amazon EKS.

Prediction Flow

Prediction:

The credit risk model processes the incoming application data and generates predictions on the likelihood of loan repayment.

Results Storage:

The predictions are stored back in DynamoDB, allowing for immediate retrieval and response.

API Integration

API Gateway:

An API is created using Amazon API Gateway to serve the predictions to the mobile development team, enabling real-time updates to the mobile app.

Historical Data Storage

S3 for Historical Use:

The raw streaming data and prediction results are also stored in Amazon S3 for historical analysis and auditing. This data can be used later by data scientists to retrain and improve the credit risk model, ensuring its accuracy over time.

Overall Summary: Batch and Real-Time Data Pipelines for Barclays Bank

Batch Processing Pipeline

Objective: Optimize ATM locations and enhance credit risk assessment.

Data Sources:

RDBMS (PostgreSQL)

External files (e.g., OneDrive)

Key Tools:

AWS DMS: Data migration from PostgreSQL to S3.

AWS Lambda: File ingestion to S3.

AWS Glue: ETL for transforming raw data.

Amazon EMR: Heavy transformations if needed.

Amazon Redshift: Data warehousing for analysis.

Tableau: Descriptive analytics for interactive dashboards.

AWS SageMaker: Model training for credit risk.

Real-Time Processing Pipeline

Objective: Provide real-time credit risk assessment for loan applications.

Data Sources:

API for credit requests.

Key Tools:

Amazon MSK: Real-time data ingestion from API.

Apache Flink / Spark Streaming: Real-time data transformation.

DynamoDB: Low-latency storage for transformed data.

AWS SageMaker: Real-time model predictions.

Amazon API Gateway: Serve predictions to mobile applications.

Amazon S3: Store historical data for future model training

Overall Summary: Batch and Real-Time Data Pipelines for Barclays Bank

Batch Processing Pipeline

Objective: Optimize ATM locations and enhance credit risk assessment.

Data Sources:

RDBMS (e.g., Azure SQL Database)

External files (e.g., Blob Storage)

Key Tools:

Azure Data Factory: Data migration and orchestration.

Azure Functions: File ingestion to Blob Storage.

Azure Databricks: ETL for transforming raw data.

Azure Synapse Analytics: Data warehousing for analysis.

Power BI: Descriptive analytics for interactive dashboards.

Azure Machine Learning: Model training for credit risk.

Real-Time Processing Pipeline

Objective: Provide real-time credit risk assessment for loan applications.

Data Sources:

API for credit requests.

Key Tools:

Azure Event Hubs: Real-time data ingestion from API.

Apache Spark Structured Streaming: Real-time data transformation.

Azure Cosmos DB: Low-latency storage for transformed data.

Azure Machine Learning: Real-time model predictions.

Azure API Management: Serve predictions to mobile applications.

Azure Blob Storage: Store historical data for future model training.

Overall Summary: Batch and Real-Time Data Pipelines for Barclays Bank

Batch Processing Pipeline

Objective: Optimize ATM locations and enhance credit risk assessment.

Data Sources:

RDBMS (e.g., Cloud SQL)

External files (e.g., Google Cloud Storage)

Key Tools:

Google Cloud Dataflow: Data migration and orchestration.

Cloud Functions: File ingestion to Cloud Storage.

Dataproc: ETL for transforming raw data.

BigQuery: Data warehousing for analysis.

Looker: Descriptive analytics for interactive dashboards.

AI Platform: Model training for credit risk.

Real-Time Processing Pipeline

Objective: Provide real-time credit risk assessment for loan applications.

Data Sources:

API for credit requests.

Key Tools:

Google Cloud Pub/Sub: Real-time data ingestion from API.

Apache Beam: Real-time data transformation.

Cloud Firestore: Low-latency storage for transformed data.

AI Platform: Real-time model predictions.

Cloud Endpoints: Serve predictions to mobile applications.

Google Cloud Storage: Store historical data for future model training.

Overall Summary: Batch and Real-Time Data Pipelines for Barclays Bank (On-Premises)

Batch Processing Pipeline

Objective: Optimize ATM locations and enhance credit risk assessment.

Data Sources:

RDBMS (e.g., PostgreSQL)

External files (e.g., local file systems)

Key Tools:

Apache Sqoop: Data migration from RDBMS to Hadoop.

Apache NiFi: Data ingestion from external files into HDFS.

Apache Hive: ETL for transforming raw data.

Cloudera Data Warehouse: Data warehousing for analysis.

Tableau: Descriptive analytics for interactive dashboards.

Apache Spark: Model training for credit risk.

Real-Time Processing Pipeline

Objective: Provide real-time credit risk assessment for loan applications.

Data Sources:

API for credit requests.

Key Tools:

Apache Kafka: Real-time data ingestion from API.

Apache Flink: Real-time data transformation.

HBase: Low-latency storage for transformed data.

Apache Spark: Real-time model predictions.

REST API: Serve predictions to mobile applications.

HDFS: Store historical data for future model training.

Azure and GCP

Company Overview: A company specializing in manufacturing and selling high-quality kitchen appliances (e.g., ovens, refrigerators, dishwashers) is looking to leverage data analytics to enhance their operations, marketing strategies, and customer insights.

Objectives

Sales Analysis: Understand sales performance across different regions and product lines.

Customer Insights: Analyze customer demographics and purchasing behavior to tailor marketing strategies.

Inventory Management: Optimize inventory levels based on sales forecasts and seasonal trends.

Data Sources

Sales Data: Transaction data from the e-commerce platform and retail partners.

Customer Data: Demographic and behavioral data collected through the website and customer interactions.

Inventory Data: Data from supply chain and warehouse management systems.

Data Pipeline with Snowflake and dbt

Data Ingestion:

Snowpipe: Automatically loads data from the e-commerce platform and other sources into Snowflake in real-time, ensuring up-to-date sales and inventory information.

Data Transformation:

dbt (data build tool): Used for transforming data already loaded into Snowflake. The existing compute resources of Snowflake allow for cost-effective transformations, eliminating the need for expensive ETL tools.

Examples of Transformations:

Creating a consolidated sales table that aggregates daily sales by region and product category.

Developing a customer segmentation model based on purchasing behavior and demographics.

Generating forecasts for inventory needs using historical sales data.

Data Modeling:

dbt enables the creation of data models that can be easily maintained and reused across the organization. This ensures consistency in reporting and analytics.

Analytics and Reporting:

BI Tools (e.g., Tableau or Looker): Connect to Snowflake to visualize the transformed data. Dashboards can be created to monitor sales performance, customer insights, and inventory levels.

Cost Optimization:

By leveraging Snowflake’s capabilities and using dbt for transformations, the company can minimize costs associated with data processing while maximizing performance and efficiency.

Team Structure

Team Size: The team typically consisted of 6-10 members, including:

Data Engineers

Data Scientists

Business Analysts

Product Owners

QA Engineers

Specific Role

Role: As a Data Engineer, my responsibilities included:

Designing and implementing data pipelines.

Managing data ingestion, transformation, and storage using tools like Snowflake and dbt.

Collaborating with data scientists to prepare data for modeling.

Ensuring data quality and integrity.

Activities and Collaboration

Agile Methodology: We followed Agile practices, specifically Scrum, which involved:

Sprint Planning: Collaboratively defining the scope and tasks for each sprint.

Daily Standups: Brief meetings to discuss progress, roadblocks, and plans for the day.

Retrospectives: Reflecting on the sprint to identify what went well and areas for improvement.

Collaboration Tools:

Jira: Used for task management and tracking progress. We employed Kanban boards to visualize the workflow.

Confluence: Documented project details, decisions, and shared knowledge among team members.

GitHub/GitLab: Managed code repositories, conducted code reviews, and tracked changes.

Dispute Resolution and Communication

Conflict Management: We fostered an open communication culture, encouraging team members to voice concerns early. In case of disputes:

Discussed issues openly during daily standups or scheduled meetings.

Engaged in collaborative problem-solving to reach consensus.

Task Management:

Tasks were assigned during sprint planning. If I anticipated that a task wouldn’t be completed on time, I communicated this early to the team, providing updates on progress and any blockers.

We encouraged proactive communication to adjust priorities or redistribute tasks as needed, ensuring that project timelines remained realistic.