BUAN 6335.501/Spring 2023

Data Analytics Organization Design Problem:

Group 2 - Surya Saketh Akella, Mira Radhakrishnan, Pavithra Rajasekaran, Raghuraman Shankar, Yamini Veepuri, Jaswanth Yarlagadda

Objective

In the rapidly changing world of payment acquiring and processing companies, it's clear that our current data analytics architecture is falling short. It's outdated, lacks standardization, and hinders our ability to introduce new products. This poses challenges for critical functions like investor reporting, regulatory compliance, and analyzing customer transactions.

To address these challenges and reach our full potential, we have devised a unified data analytics approach that can support our internal and external teams. Our goal is to establish a single, reliable source of truth for investor reporting, regulatory compliance, and analyzing customer transactions. We also want to drive data standardization across our subsidiaries and divisions.

Our data sources

Payment acquiring and processing companies may collect upstream data from various sources, including:

- Point of Sale (POS) terminals: These are devices that enable customers to make payments using credit or debit cards at physical retail locations.
- E-commerce endpoints: These are web-based platforms that allow customers to make payments online.
- Mobile devices: Customers can make payments using mobile devices through mobile payment apps such as Apple Pay or Google Wallet.
- Automated Clearing House (ACH): ACH is an electronic network for financial transactions in the United States.
- Payment gateways: These are third-party services that enable businesses to accept and process payments from various sources, including credit and debit cards, bank transfers, and digital wallets.
- Payment processors: These are companies that handle the processing of payments between merchants and banks, including authorization, settlement, and reconciliation.
- Issuing banks: These are banks that issue credit or debit cards to customers.
- Acquiring banks: These are banks that provide merchant accounts to businesses, enabling them to accept payments from customers.
- Payment networks: These are companies that provide the infrastructure for processing payments between banks, merchants, and customers, such as Visa, Mastercard, American Express, and Discover.
- Loyalty and rewards programs: Payment acquiring and processing companies may also collect data from loyalty and rewards programs to gain insights into customer behavior and preferences.

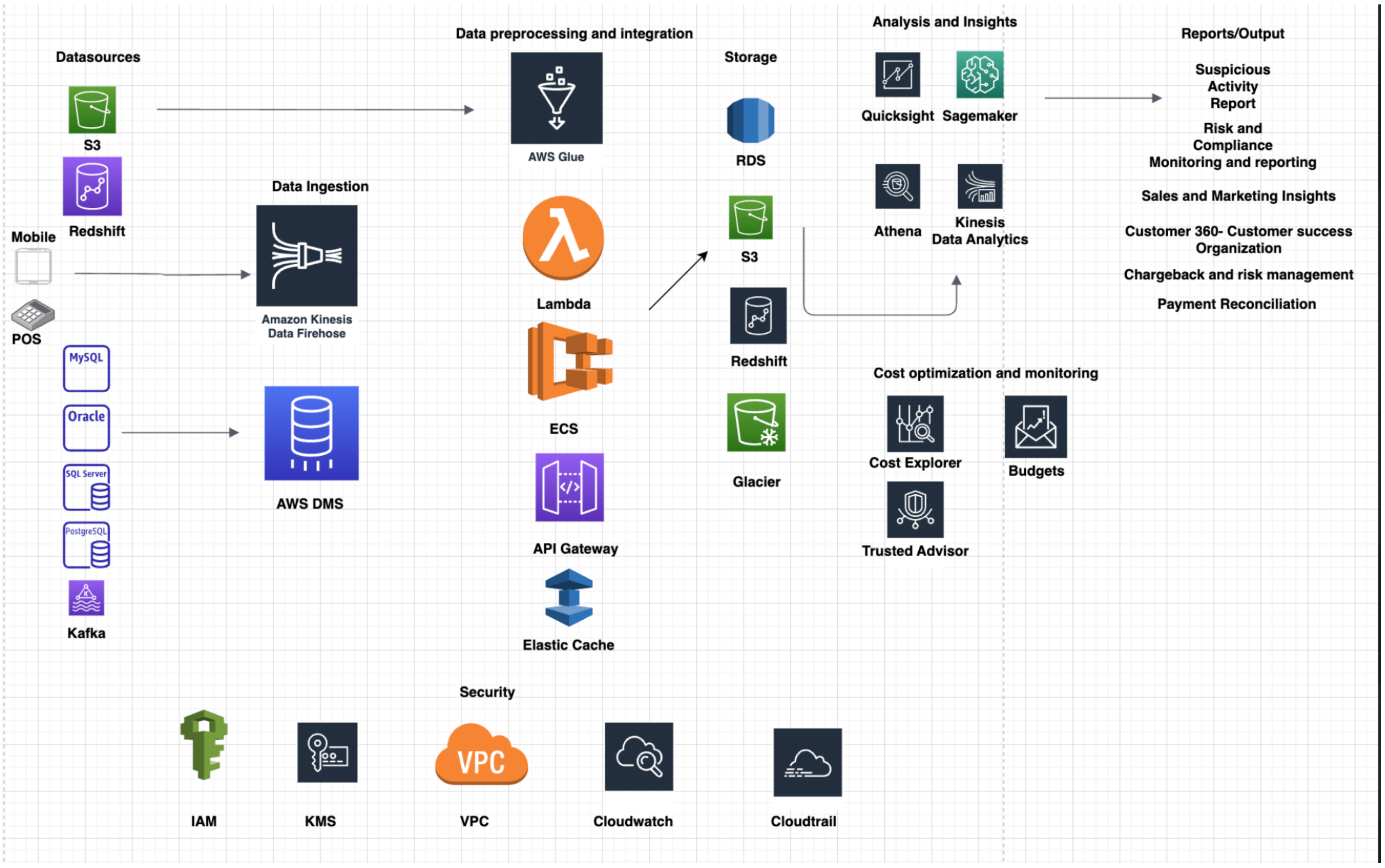
Our redesigned data analytics architecture will prioritize data security, scalability, and cost optimization. We aim to eliminate duplicate data reporting, simplify data integration, and provide seamless access to a unified data architecture. Additionally, we will empower downstream reporting and business systems through microservices, catering to the needs of developers, finance teams, analysts, and data scientists. We will focus on efficient storage usage and support historical and incremental data models, enabling smooth data preparation for downstream processes. This will ensure consistency and remove irregularities.

By achieving these goals, we expect to unlock significant business value, improve operational efficiency, and enable data-driven decision-making.

In this project report, we will delve into the details of our proposed data analytics architecture. We will discuss the strategies and solutions we have implemented to achieve,

- Unified data access and storage,
- Scalable data ingestion and extraction,
- Data security,
- Data Integration flexibility,
- Optimal storage usage,
- Cost efficiency, and
- Overall scalability

Our Data Architecture:



Sample Project Plan

Phase 1: Project Initiation and Requirements Gathering

- Define project goals, scope, and success criteria.
- Identify key stakeholders and establish communication channels.
- Conduct workshops and interviews to gather requirements for unified data access, scalable ingestion/extraction, data security, integration flexibility, optimal storage usage, cost efficiency, and overall scalability.
- Document the gathered requirements and obtain stakeholder approval.

Phase 2: Solution Design and Architecture

- Develop a high-level solution design that addresses the identified requirements.
- Identify the appropriate AWS services and tools to meet the desired outcomes.
- Design the data access and storage architecture, considering scalability, security, and integration flexibility.
- Determine the data ingestion and extraction mechanisms and define the necessary workflows.
- Design data security measures, including access controls, encryption, and compliance requirements.
- Establish storage optimization techniques, such as compression, partitioning, and data lifecycle management.
- Create a detailed solution architecture document for reference and review.

Phase 3: Infrastructure Setup and Configuration

- Provision the required AWS resources, including compute instances, storage services, and networking components.
- Configure data access mechanisms, such as setting up virtual private networks (VPNs) or secure direct connections.
- Implement data ingestion and extraction processes, including real-time streaming and batch processing.
- Set up security measures, such as implementing encryption, establishing identity and access management policies, and enabling audit logging.
- Configure storage optimization techniques based on the identified requirements.

Phase 4: Data Migration and Integration

- Perform data migration from existing sources to the new unified data storage solution.
- Develop and execute data integration processes to consolidate and transform data from various sources.
- Implement data quality checks and cleansing processes to ensure data accuracy and consistency.
- Establish data governance practices and documentation to maintain data integrity and compliance.

Phase 5: Testing and Quality Assurance

- Conduct thorough testing of the implemented solution, including functional, integration, performance, and security testing.
- Identify and resolve any issues or bugs identified during testing.
- Validate the solution against the defined success criteria and obtain stakeholder sign-off.

Phase 6: Deployment and Training

- Plan the deployment strategy, considering any downtime or migration requirements.
- Execute the deployment plan and ensure a smooth transition to the new solution.
- Provide training and documentation to end-users, administrators, and other relevant stakeholders on using and managing the implemented solution effectively.

Phase 7: Monitoring and Optimization

- Set up monitoring and alerting mechanisms to track system performance, data access, and security.
- Continuously monitor the solution's scalability, resource utilization, and cost efficiency.
- Identify opportunities for optimization, such as refining data ingestion/extraction processes, tuning storage configurations, or implementing automation.

Phase 8: Ongoing Support and Maintenance

- Establish ongoing support processes and documentation for system maintenance, upgrades, and issue resolution.
- Conduct regular reviews and assessments of the implemented solution to ensure it aligns with evolving business needs and technology advancements.

Unified Data Storage and Access:

As the data flows in from various systems and databases, processing millions of transactions and customer data daily becomes a cumbersome activity. These financial transactions generate vast amounts of data that need to be collected, stored, and analyzed. The current system works inefficiently with such huge amounts of data leading to duplication of data and discrepancies between datasets, leading to inaccurate analysis and decisions. Furthermore, it can create a lack of transparency and accountability, making it difficult to identify the source of errors in the data.

One way to overcome these challenges is having a unified data architecture. Unified data storage and access is a critical component of modern business intelligence and analytics. The ability to store, process and analyze data from various sources in a unified way provides a single source of truth for the stakeholders in an organization. With the increasing volume and complexity of the data generated through financial transactions and acquisitions, having a unified approach for data storage and access is essential to drive insights, optimize processes, and improve decision-making. By consolidating data into a centralized location, businesses can gain a holistic view of their operations, customer behavior, and market trends, enabling them to identify new revenue opportunities, mitigate risks, and enhance their competitive position.

AWS offers a variety of services and tools that can be used to achieve a unified data storage and access solution. Data must be stored in a centralized data lakehouse. Data lakehouses are becoming increasingly popular as a modern approach to storing and accessing large volumes of data. They combine the best features of data warehouses and data lakes to create a unified platform for data storage, processing, and analysis.

A data lakehouse in AWS can provide several benefits:

1. Flexibility: One of the primary advantages of using a data lakehouse is the ability to store both structured and unstructured data. Payment processing and acquiring companies have diverse data sources, including transaction logs, clickstreams, social media, and more. These data sources often vary in structure and schema, making it challenging to store and process them in traditional data warehouses. A data lakehouse can handle this diversity by storing data in its raw, unprocessed form and using schema-on-read to transform the data into a structured format when it is accessed.

2. Scalability: As data grows exponentially every day, a scalable data storage solution is needed that can accommodate the increasing volume of data. AWS provides an elastic infrastructure that can easily scale up or down based on the demand for processing and storage resources. Additionally, the use of serverless technology, such as AWS Lambda, can help reduce costs by only processing data when it is needed. Amazon S3 provides highly scalable, durable, and secure object storage that can store petabytes of data. Amazon Redshift, on the other hand, is a cloud-based data warehouse that can handle large volumes of structured and semi-structured data for analytics and reporting purposes. By using these tools, we can easily store and analyze vast amounts of data without having to worry about scalability issues.

3. Governance: A data lakehouse can help improve data governance and security. With a unified data storage system, we can have better control over their data, reducing the risk of data silos and inconsistencies. AWS provides a range of security and compliance services, such as AWS Identity and Access Management (IAM), which enables fine-grained control over access to data. IAM can be used to define roles and permissions for different users or groups, ensuring that only authorized personnel have access to sensitive data. In addition to IAM, AWS offers a range of compliance certifications, such as PCI-DSS and HIPAA, which provide assurance that data is being stored and processed in compliance with regulatory standards. These certifications are critical for payment processing and acquiring data that need to comply with strict regulations around data privacy and security.

4. Cost-effective: A data lakehouse approach can be more cost-effective compared to a traditional data warehouse approach because it allows for a more flexible and scalable data storage architecture. With a data lakehouse, data can be stored in its raw and unprocessed form, which reduces the cost of data transformation and processing. Additionally, data lakehouses use distributed storage systems such as Amazon S3, which are designed to scale horizontally and provide cost savings through economies of scale. With S3, you only pay for what you use, which means you can store large amounts of data cost-effectively and pay only for the storage and retrieval you need.

Additionally, the above listed AWS tools and services also aid in implementing the principles of CAP theorem in the data lakehouse architecture. CAP theorem is an important consideration when building a cloud-based data architecture, particularly when designing distributed systems where data is replicated across multiple nodes. The theorem states that in a distributed system, it is impossible to simultaneously achieve all three of the following guarantees: consistency, availability, and partition tolerance.

Consistency:

- Amazon S3 (Simple Storage Service): AWS S3 offers strong data consistency across all regions, ensuring that data written to S3 is immediately available for subsequent reads. This helps maintain consistency when storing and accessing data within the data lakehouse.
- Amazon RDS (Relational Database Service): RDS supports ACID (Atomicity, Consistency, Isolation, Durability) transactions for relational databases, ensuring data consistency for critical transactional data within the data lakehouse.
- AWS Glue: AWS Glue provides data cataloging and metadata management capabilities, ensuring consistent data definitions and schemas across different sources and systems.

Availability:
Amazon S3: S3 provides high durability and availability for data storage, with built-in redundancy and data replication across multiple availability zones. This ensures that data within the data lakehouse remains highly available even in the event of hardware failures or other disruptions.

Partition Tolerance:

- Amazon S3: S3 is a distributed storage system designed to handle massive scalability and concurrent access. It can seamlessly handle data flows from multiple sources and allow partitioned access to data across different components or systems within the data lakehouse.
- AWS Glue: Glue supports distributed data processing, allowing large-scale data transformations and analytics across partitioned datasets.
- Amazon Kinesis: Kinesis enables real-time data streaming and processing, providing partition tolerance for high-speed data ingestion and processing scenarios.

By leveraging these AWS services and features, we can design and implement data lakehouse architectures that achieve the desired levels of consistency, availability, and partition tolerance while ensuring the scalability, reliability, and performance required for unified data storage and access needs.

Data ingestion:
The ingestion layer is responsible for bringing the data into data lake. It provides ability to connect to internal and external data sources. The data ingestion layer is a critical component of any modern data architecture, particularly for big data and streaming data workloads. It involves the process of collecting, processing, and storing data from various sources into a centralized repository, which can then be accessed and analyzed by other layers or systems within the architecture. It can ingest batch and streaming data to the storage layer.

The data ingestion layer plays a crucial role in any data analytics architecture, as it sets the foundation for subsequent layers such as data processing, storage, and analytics. The choice of tools and technologies for the data ingestion layer depends on various factors such as data volume, velocity, variety, and the business requirements of the organization.

We will use Amazon Kinesis firehose to stream the data to different destinations such as Amazon S3, Amazon Redshift and Amazon Elasticsearch service. Kinesis firehose can ingest data from variety of sources, including POS terminals, e commerce endpoints and mobile services. This service will allow us handle both batch and streaming data ingestion with ease and scale as per our requirements.

With Kinesis Firehose, you can easily configure data delivery to destinations such as S3, Redshift, or Elasticsearch. You can choose the frequency of data delivery and enable data compression and encryption for security. You can also configure transformation functions, such as data filtering, data format conversion, and data cleaning, using AWS Lambda functions.

Kinesis Firehose also integrates with AWS Identity and Access Management (IAM) to provide fine-grained access control and authorization. You can use IAM to create roles and policies that restrict access to specific Kinesis Firehose resources and operations.

Overall, Kinesis Firehose simplifies the process of loading streaming data into AWS services, reduces the operational overhead of managing streaming data, and provides a reliable and scalable data ingestion solution.

With AWS Database Migration Service, we can seamlessly migrate data from on-premises databases to AWS. It can also be used for continuous data replication between on-premises databases and AWS. We can use DMS to migrate data from various databases and messaging systems to AWS. DMS supports several database engines, including MySQL, Oracle, Cassandra, SQL server, and PostgreSQL. We can set up DMS to replicate data from on-premises databases to Amazon RDS or Amazon Aurora. Alternatively, we can use DMS to replicate data to Amazon S3, which can then be used by AWS Glue for further processing.

Data Processing and Integration:

AWS Lambda can be used to process and analyze data from these various sources by triggering events in response to changes in data. For example, Lambda can be used to process data from POS terminals, e-commerce endpoints, mobile devices, ACH, payment gateways, payment processors, issuing banks, acquiring banks, payment networks, and loyalty and rewards programs.

API Gateway can be used to create RESTful APIs that interact with downstream systems, such as finance, merchant, segmentation, sales/marketing, and customer success organizations. These APIs can be used to access and manipulate data from various sources and provide standardized data to downstream systems. For example, the API can be used to retrieve transaction data from the data lake and provide it to the finance team for financial reporting.

To integrate data from these various sources, AWS Lambda can be used to process and normalize data as it is ingested into the data lake. This can include cleaning up data, removing duplicates, and standardizing data formats. Once the data is in the data lake, it can be accessed via APIs to provide data to downstream systems. For example, data from POS terminals can be ingested into the data lake via Lambda functions that normalize the data and store it in the appropriate data store. APIs can then be used to provide access to this data to downstream systems, such as the sales/marketing team, who can use it to gain insights into customer behavior and preferences. Similarly, data from e-commerce endpoints can be ingested into the data lake via Lambda functions that normalize the data and store it in the appropriate data store. APIs can then be used to provide access to this data to downstream systems, such as the finance team, who can use it for financial reporting.

Mobile payments can also be processed via Lambda functions that process and normalize data as it is ingested into the data lake. This data can then be accessed via APIs to provide data to downstream systems. ACH transactions can be processed via Lambda functions that process and normalize data as it is ingested into the data lake. This data can then be accessed via APIs to provide data to downstream systems. Payment gateways, payment processors, issuing banks, acquiring banks, payment networks, and loyalty and rewards programs can also be processed via Lambda functions that process and normalize data as it is ingested into the data lake. This data can then be accessed via APIs to provide data to downstream systems.

Kinesis Data Firehose can invoke our Lambda function to transform incoming source data and deliver the transformed data to destinations. We can enable Kinesis Data Firehose data transformation when we create our delivery stream. This can be done in pre-processing before data is stored in Amazon S3 bucket.

As the data volume increases, it can become challenging to retrieve and process data in real-time, leading to increased response times and system latency. Hence, we can use Elastic Cache to cache data from the microservices used for downstream reporting or business systems. This can help in reducing the response time of these systems and improving their integration with other systems.

With the above-mentioned Lambda functions where all the normalizations and data transformation functions are written can be integrated in the ECS service platform to create the workflow.

Amazon Elastic Container Service (ECS) is a fully managed container service that makes it easy to run, stop, and manage Docker containers on a cluster. it can be used to build and run containerized data pre-processing workflows, which can offer several benefits over traditional on-premises or cloud-based data pre-processing solutions. There are several container-based data pre-processing platforms like Kubernetes, Apache airflow, AWS EKS services etc. On comparing all its features in terms of cost, scalability, upgradation to other platform in future Amazon ECS is found to be a better option.

Here are some ways that ECS can be better for data pre-processing:

- Scalability: ECS can easily scale up or down depending on the size and complexity of your data pre-processing workflows. You can use Amazon ECS to create a cluster of container instances and then run containerized data pre-processing tasks on the cluster. This allows you to easily add or remove compute resources as needed to handle larger or smaller data volumes.
- Flexibility: ECS supports a wide range of containerized data pre-processing tools and frameworks, including TensorFlow, PyTorch, Scikit-learn, and Apache Spark. You can use these tools to pre-process data in various ways, such as cleaning, normalization, transformation, and feature extraction. ECS also supports custom containers, which can be used to run any custom data pre-processing code.

- Integration: ECS integrates with other AWS services, such as Amazon S3, AWS Glue, AWS Lambda, and Amazon EMR, which can be used to store, process, and analyse data. For example, you can use ECS to pre-process data stored in Amazon S3 and then load the pre-processed data into an Amazon Redshift data warehouse for analysis.
- Cost-effectiveness: ECS offers a pay-as-you-go pricing model, which means you only pay for the compute resources you use. This can be more cost-effective than running data pre-processing workflows on dedicated on-premises or cloud-based infrastructure, which can be more expensive to set up and maintain.

Overall, ECS can offer several benefits for data pre-processing, including scalability, flexibility, integration, and cost-effectiveness. However, the choice of tool or service will depend on the specific needs and requirements of your data pre-processing workflows.

Where Kubernetes is also one of the other competitive options but in terms of cost it is cost effective and its complex. ECS can support all the necessary service and as Kubernetes the pipeline in ECS can be deployed in other platforms in future in its own way. Additionally, ECS offers cost savings with Spot Instances and Reserved Instances pricing options.

Optimal storage usage for different use cases:
Optimal storage usage varies depending on the specific use case and the type of data being stored. Few of the use cases are risk and compliance monitoring and reporting, fraud detection and prevention, sales/marketing insights and alerts, chargeback management, and customer 360 use cases for customer success etc.

1. The optimal storage usage for the above use cases can vary depending on the specific requirements of each use case. However, a data lakehouse architecture in AWS can provide a flexible and scalable solution for storing and processing large volumes of data across different use cases.
2. Risk and compliance monitoring and reporting: This use case involves the analysis of large amounts of transactional and other data to identify potential fraudulent or suspicious activity. Amazon Redshift may be appropriate for this use case due to its ability to handle complex queries and performant analytics. Redshift also has built-in support for data encryption and access controls, which are important for compliance purposes.
3. Finance: This involves the storage and analysis of financial data, such as revenue, expenses, and profits. A data lakehouse solution, such as Amazon S3 with AWS Glue and Amazon Athena, may be appropriate for this use case due to its ability to store and process large amounts of structured and unstructured data at scale. S3 also has built-in support for versioning and access controls, which are important for auditability and compliance purposes.
4. Merchant and Segmentation: This involves the segmentation of customers and merchants based on various criteria, such as transaction volume, location, and industry. A NoSQL database solution, such as Amazon DynamoDB, may be appropriate for this use case due to its ability to handle large amounts of unstructured and semi-structured data and perform fast queries on specific attributes.
5. Sales/marketing insights and alerts: This involves the storage and analysis of customer and sales data to identify trends, patterns, and anomalies. A data warehouse solution,

such as Amazon Redshift, may be appropriate for this use case due to its ability to perform complex analytics and support real-time reporting and dashboards.

6. Customer 360 use cases for customer success organization: This involves the consolidation of customer data from various sources, such as CRM, support tickets, and transactional data, to provide a complete view of the customer. A data lakehouse solution, such as Amazon S3 with AWS Glue and Amazon Athena, may be appropriate for this use case due to its ability to store and process large amounts of structured and unstructured data and perform fast queries on specific attributes.
7. Fraud detection and prevention: A combination of structured and unstructured data can be stored in a data lakehouse. Structured data such as transactional data can be stored in a relational database such as Amazon RDS or Amazon Aurora, while unstructured data such as log data can be stored in a data lake using Amazon S3 and Amazon EMR for processing.
8. Payment reconciliation: This data ensures that payments are correctly matched and recorded for each transaction. This type of data can be handled by storing transactional data in a relational database and using a data warehouse such as Amazon Redshift for aggregating and analysing the data.
9. Chargeback and risk management: Chargebacks occur when a customer disputes a charge, and the payment companies need to process them. They use data to track chargeback rates, identify trends, and implement strategies to reduce them. This type of data can be handled by storing data from various sources such as transactional data, log data, and external data sources in a data lakehouse. This data can be processed using tools such as Amazon EMR, Amazon Athena, or Amazon Redshift Spectrum to provide insights into chargeback rates, identify trends, and implement strategies to reduce them.
10. Business intelligence and analytics: This can be achieved by storing data from various sources such as transactional data, customer behaviour data, and market trend data in a data lakehouse. The data can be processed using tools such as Amazon EMR, Amazon Athena, or Amazon Redshift to provide insights into business operations and identify opportunities for growth and optimization.
11. Payment gateway optimization: This can be achieved by storing transactional data in a relational database such as Amazon RDS or Amazon Aurora and using a data warehouse such as Amazon Redshift for aggregating and analysing the data. This can help identify and address bottlenecks and other issues in the payment gateway.

Data Security

Data security is a critical aspect of any data analytics organization's design and infrastructure, especially for a payments processing company handling sensitive financial transactions. A robust data security strategy should encompass various aspects, including access control, data anonymization, data encryption, secure data transfer, and compliance with relevant standards and regulations.

Components across all layers of our architecture protect data, and identities, and processes resources by natively using capabilities provided by the security and governance layer. Here are some of the ways in which we address data security while using Amazon Web Services (AWS) in our model:

1. **Authentication and authorization**

For our payment processing model, we rely on the robust security features provided by AWS Lake Formation for authentication and authorization. AWS Lake Formation offers a comprehensive permissions model that seamlessly integrates with AWS Identity and Access Management (IAM), enabling us to effectively control access to our data lakes and the associated metadata.

IAM is a crucial component that allows us to manage access to various AWS services and resources in a secure manner. It provides user-, group-, and role-level identity management, giving us granular control over access permissions for resources across all layers of our architecture. With IAM, we can configure fine-grained access control, support multi-factor authentication, and implement single sign-on functionality by integrating with corporate directories and popular identity providers such as Google, Facebook, and Amazon. In our payment processing model, we leverage IAM to create and manage AWS users and groups, and utilize permissions to define the precise access privileges for each entity, enabling us to enforce secure access to AWS resources.

Within the context of Lake Formation, we can effortlessly grant or revoke access at the database, table, or column level for IAM users, groups, or roles that are defined within our payment processing company's AWS account. By leveraging Lake Formation's integrated permissions model, we can ensure secure access to our payment processing data and metadata, providing a strong foundation for maintaining the confidentiality and integrity of sensitive information.

2. Encryption

Ensuring the security and confidentiality of sensitive data is of utmost importance to us. To achieve this, we rely on AWS Key Management Service (KMS) for the creation and management of cryptographic keys that are used for encryption and decryption purposes. AWS KMS offers a robust solution that allows us to maintain control over our encryption keys while benefiting from seamless integration with various AWS services throughout our architecture.

AWS KMS enables us to create and manage both symmetric and asymmetric customer-managed encryption keys. These keys are used to encrypt data at various stages of our data processing pipeline, providing an additional layer of protection for sensitive information. AWS services across our architecture natively integrate with AWS KMS, allowing us to encrypt data within our data lake seamlessly.

Additionally, detailed audit trails generated by AWS CloudTrail allow us to monitor and track access to the encryption keys, maintaining a comprehensive record of key usage.

In the context of our payment processing model, encryption is crucial throughout the entire data lifecycle. It begins with the secure transmission of payment data from customers to our systems, where the sensitive information is encrypted using AWS KMS-managed keys. This encryption ensures that the data remains protected during transit. Furthermore, within our data lake, all stored payment data, including customer information, transaction details, and financial records, is encrypted using encryption keys managed by AWS KMS. This encryption provides an additional layer of defence against unauthorized access and helps safeguard the confidentiality and integrity of the data.

By leveraging AWS KMS for encryption within our model, we can maintain a high level of security for sensitive payment data, ensuring compliance with industry regulations and instilling trust in our customers.

3. Network protection

In our payment processing model, we prioritize the security and integrity of our network infrastructure. To achieve this, we leverage Amazon Virtual Private Cloud (Amazon VPC), which allows us to provision a logically isolated section within the AWS Cloud environment. This isolation ensures that our network remains segregated from the internet and other AWS customers, providing an additional layer of protection for our payment processing operations.

With Amazon VPC, we have the flexibility to choose our own IP address range, create subnets, and configure route tables and network gateways according to our specific requirements. This level of control enables us to design a private VPC environment that aligns with our security needs and network architecture. By launching resources from other layers of our architecture within this private VPC, we can ensure that all traffic to and from these resources is protected within this isolated network environment.

The use of Amazon VPC grants us comprehensive control over our virtual networking environment, including resource placement, connectivity, and security. We can strategically position our resources within the VPC to optimize performance and security. Additionally, we have the ability to establish secure connectivity options, such as Virtual Private Network (VPN) connections or AWS Direct Connect, to securely connect our on-premises infrastructure with the payment processing resources deployed in the VPC.

By leveraging Amazon VPC in our payment processing model, we can establish a robust network protection framework that safeguards sensitive payment data and infrastructure. The isolation provided by VPC ensures that our network remains secure from external threats, while the ability to configure routing tables and gateways allows us to efficiently manage network traffic. This network protection mechanism, combined with other security measures implemented throughout our architecture, helps us maintain a secure and trusted environment for processing payments and safeguarding customer data.

4. Monitoring and logging

Maintaining comprehensive logs and monitoring metrics is vital for security and operational analysis. To achieve this, we rely on AWS CloudWatch, which enables us to store detailed logs and monitor metrics across all layers of our architecture.

AWS CloudWatch empowers us to analyze logs, visualize monitored metrics, define monitoring thresholds, and receive alerts when those thresholds are exceeded. This functionality allows us to proactively identify any anomalies or potential security incidents within our payment processing environment. By leveraging CloudWatch, we can gain valuable insights into the performance and health of our infrastructure, ensuring the smooth operation of payment processing services.

In addition to CloudWatch, we also utilize AWS CloudTrail to store extensive audit trails of user and service actions within our payment processing architecture. CloudTrail provides a comprehensive event history of all activities occurring within our AWS account, including actions performed through the AWS Management Console, AWS SDKs, command line tools, and other AWS services. This event history simplifies security analysis, tracks changes to resources, and facilitates troubleshooting activities. CloudTrail plays a crucial role in detecting any unusual or unauthorized activity within our AWS accounts, enhancing our ability to identify and respond to potential security threats swiftly.

By leveraging CloudWatch and CloudTrail within our payment processing model, we establish a robust logging and monitoring framework. This framework enables us to conduct detailed analysis, track changes, and troubleshoot issues effectively. It provides us with the necessary visibility and insights to ensure the security and smooth operation of our payment processing infrastructure, helping us to maintain the confidentiality, integrity, and availability of sensitive payment data.

Achieving cost efficiency without reducing the effectiveness of the use cases
While designing a data architecture, it is important to keep in mind the needs of our organization and create a budget to estimate the costs that will be involved. Some examples are as below,

- Infrastructure costs: This includes the cost of servers, storage, networking, and other related hardware and software. We need to carefully consider the scale and complexity of our data architecture to determine the appropriate infrastructure resources and how much they will cost.
- Data processing costs: Depending on the volume and complexity of our data, we need to use additional processing resources such as compute clusters or dedicated data processing engines. These resources will increase our data processing costs, and we need to balance the processing requirements with associated costs.
- Data storage costs: Data storage costs depend on the amount of data that needs to be stored and the required level of redundancy. We must carefully consider the type of data storage we need and select cost-effective options, such as cloud storage or object storage services.
- Maintenance and management costs: Maintenance and management costs include expenses like monitoring, backup and recovery, and security. These costs can significantly impact the overall cost of the data architecture.
- Scalability and flexibility: When designing a data architecture, we should consider how scalable and flexible it is to accommodate future growth and changes in business requirements. The cost of scalability and flexibility should be weighed against the long-term benefits of being able to adapt to changing business needs.

How to optimize costs while designing our data architecture?

Start with a cost optimization plan: Before designing the data architecture, we will design a simulation of the data architecture that we need to achieve our goals. A cost optimization plan will include specific goals for cost reduction and outline the strategies we will use to achieve those goals.

Implement Data life-cycle management
Data lifecycle management (DLM) is a policy-based approach to managing the flow information from creation to storage to when it becomes obsolete and is deleted.

DLM products automate lifecycle management processes. They typically organize data into separate tiers according to specified policies. They also automate data migration from one tier to another based on those criteria. As a rule, newer data and data that must be accessed more frequently is stored on faster and more expensive storage media, while less critical data is stored on cheaper, slower media.

Payment processing companies have a lot of regulations with respect to data retention policies and might span over several years. This policy is designed to ensure that the company complies with applicable regulations, such as those related to data security, privacy, and financial reporting. While this data may not be useful for purposes other than reporting, it is important to design a robust policy to save costs.

To implement data lifecycle management in AWS, we will use Amazon S3 lifecycle policies, which allows us to define rules for transitioning objects between different storage classes based on their age or access patterns. For example, we will set a policy to move data from the S3 Standard storage class to the S3 Infrequent Access (IA) or S3 One Zone-Infrequent Access (One Zone-IA) storage classes after a certain number of days.

In addition to S3 lifecycle policies, AWS also provides Amazon Glacier, which is a low-cost storage service for data archiving and long-term backup. We can use Amazon Glacier to store data that is rarely accessed but still needs to be retained for compliance or business purposes. We will use Amazon Glacier lifecycle policies to automate the transition of data between different Glacier storage classes based on their retention requirements.

Data compression, sharding and partitioning
Data sharding and partitioning are techniques used to horizontally partition large data sets into smaller, more manageable parts. Sharding is commonly used in distributed database systems to improve performance and scalability by breaking up large databases into smaller, more manageable pieces that can be stored on multiple servers. Partitioning, on the other hand, involves dividing data within a single database into smaller logical segments.

Data compression and partitioning can help us reduce storage costs and improve query performance. AWS provides tools like Amazon S3 Select, Amazon Redshift Spectrum, and Amazon Athena, which can help us query and analyze compressed and partitioned data.

Use cost optimization tools: AWS provides various cost optimization tools such as AWS Cost Explorer, AWS Budgets, and AWS Trusted Advisor. These tools can help monitor and control AWS spending, identify areas for cost savings, and optimize resource utilization.

Use spot instances: Spot instances are spare AWS compute capacity that can be purchased at a significantly lower price than on-demand instances. You can use spot instances for non-critical workloads such as data processing or analytics, which can help you reduce costs.

Risks associated with native-cloud model
While moving to a fully cloud-based AWS model for data processing offers many benefits, there are also several risks that should be considered:

Data security: Storing sensitive data in the cloud can pose security risks. Companies must ensure they have robust security measures in place to protect against data breaches and other security threats.

Dependence on third-party vendors: A fully cloud-based model means that companies will rely heavily on AWS services and their performance. Any outage or service disruption on AWS's part can impact business operations.

Latency and network issues: Cloud-based processing can introduce latency and network-related issues, affecting data processing and response times. This can be especially problematic for real-time processing and applications with strict latency requirements.

Vendor lock-in: Migrating data and applications from one cloud provider to another can be complex and costly. Therefore, companies must consider the risk of vendor lock-in when moving to a fully cloud-based model.

Cost management: While AWS offers cost-efficient pricing models, cloud-based data processing can still result in unexpected expenses if not managed effectively. Companies must have robust cost management strategies to avoid overpaying for cloud services.

Conclusion

Overall, this native-cloud architecture provides a scalable and cost-effective solution for managing and analyzing large amounts of financial transaction data. The use of AWS services ensures that the architecture is cloud-native and can be easily integrated with other AWS services. The inclusion of data security and access management services ensures that data is secure and compliant with regulations.