Project Objective

Develop a Data Pipeline Architecture using AWS services to store data from OpenFisheries, to be used for front-end Business Intelligence purposes.

OpenFisheries Data

The dataset is to be assessed using RESTful API the endpoint:
https://www.openfisheries.org/api/landings.json [aggregated fish data]

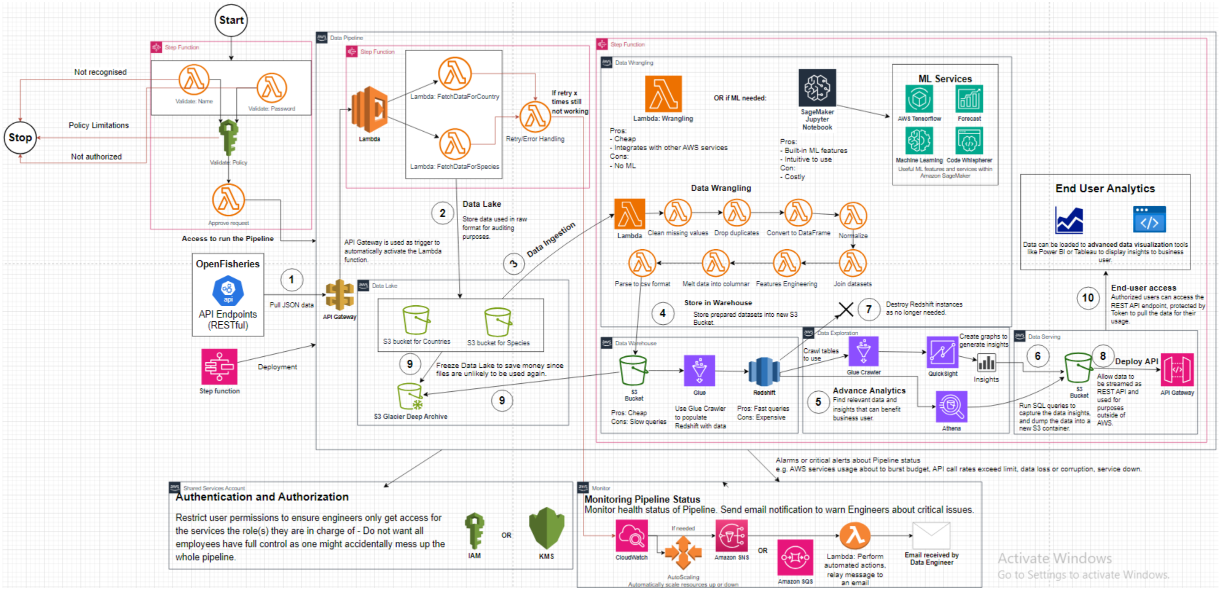
http://openfisheries.org/api/landings/countries.json [list of countries]
http://openfisheries.org/api/landings/species.json [list of species]

Tasks

Our data engineering solution needs to:

- Clean and fix any potential data quality issues from the raw form.
- Transform and prepare the data into fact and dimension tables format to be stored in a Data Warehouse.
- Ensure the data is usable for purposes of Business Intelligence, such as data visualizations and machine learning to forecast future trends.
- Be a deployable API so that the relevant data can be accessed and used by analysts to conduct data analytics and visualizations.

Data Pipeline Architecture


Start
Stop
Step Function
Not recognised
Validate: Name
Validate: Password
Policy Limitations
Validate: Policy
Not authorized
Approve request
Access to run the Pipeline
OpenFisheries
API Endpoints (RESTful)
1
Pull JSON data
API Gateway
Step function
Deployment
Data Pipeline
Step Function
Lambda
Lambda: FetchDataForCountry
Lambda: FetchDataForSpecies
Retry/Error Handling
If retry x times still not working
2
Data Lake
Store data used in raw format for auditing purposes.
API Gateway is used as trigger to automatically activate the Lambda function.
3
Data Ingestion
Data Lake
S3 bucket for Countries
S3 bucket for Species
9
Freeze Data Lake to save money since files are unlikely to be used again.
S3 Glacier Deep Archive
9
Step Function
Data Wrangling
Lambda: Wrangling
Pros:
- Cheap
- Integrates with other AWS services
Cons:
- No ML
OR if ML needed:
SageMaker Jupyter Notebook
Pros:
- Built-in ML features
- Intuitive to use
Con:
- Costly
ML Services
AWS Tensorflow
Forecast
Machine Learning
Code Whisperer
Useful ML features and services within Amazon SageMaker
Data Wrangling
Lambda
Clean missing values
Drop duplicates
Convert to DataFrame
Normalize
Join datasets
Features Engineering
Melt data into columnar
Parse to csv format
4
Store in Warehouse
Store prepared datasets into new S3 Bucket.
7
Destroy Redshift instances as no longer needed.
Data Warehouse
S3 Bucket
Pros: Cheap
Cons: Slow queries
Glue
Use Glue Crawler to populate Redshift with data
Redshift
Pros: Fast queries
Cons: Expensive
Data Exploration
Crawl tables to use
Glue Crawler
QuickSight
Create graphs to generate insights
Insights
5
Advance Analytics
Find relevant data and insights that can benefit business user.
Athena
Data Serving
6
S3 Bucket
Run SQL queries to capture the data insights, and dump the data into a new S3 container.
8
Deploy API
Allow data to be streamed as REST API and used for purposes outside of AWS.
API Gateway
End User Analytics
Data can be loaded to advanced data visualization tools like Power BI or Tableau to display insights to business user.
10
End-user access
Authorized users can access the REST API endpoint, protected by Token to pull the data for their usage.
Shared Services Account
Authentication and Authorization
Restrict user permissions to ensure engineers only get access for the services the role(s) they are in charge of - Do not want all employees have full control as one might accidentally mess up the whole pipeline.
IAM
OR
KMS
Alarms or critical alerts about Pipeline status
e.g. AWS services usage about to burst budget, API call rates exceed limit, data loss or corruption, service down.
Monitor
Monitoring Pipeline Status
Monitor health status of Pipeline. Send email notification to warn Engineers about critical issues.
CloudWatch
If needed
AutoScaling
Automatically scale resources up or down
Amazon SNS
OR
Amazon SQS
Lambda: Perform automated actions, relay message to an email
Email received by Data Engineer

Built using **Drawio**: https://app.diagrams.net/#G13J0cF2yyGpaR7hjSbrzBRWiW_HM7i7L0

Deployment tool: AWS CloudFormation Stack

Checklist (configurations of services)

Logging and Monitoring Mechanism for Compliance

Lambda Functions:

- Enable CloudWatch Logs for Lambda functions.
- Implement structured logging for compliance and auditing.
- Configure CloudWatch Alarms for critical Lambda metrics (duration, errors, invocations).

S3 Buckets:

- Enable CloudWatch Metrics for S3 buckets.
- Implement AWS CloudTrail for S3 bucket access monitoring.
- Configure S3 bucket access logging for compliance.
- Enable version control to track any changes made.

AWS Glue:

- Enable CloudWatch Logs for AWS Glue ETL jobs and CloudWatch Alarms for Glue job metrics.

Athena:

- Enable CloudWatch Metrics for Athena.
- Monitor query execution times and errors.
- Set up CloudWatch Alarms for Athena query metrics.

Amazon QuickSight:

- Enable QuickSight usage and access logs for auditing.
- Monitor QuickSight dashboard usage and access.

Access Control Management (IAM, Users, Roles, and Policies)

IAM Roles:

- Define IAM roles with least privilege for Lambda functions, Glue jobs, Athena, and QuickSight.
- Regularly review and update IAM policies based on evolving requirements.

Users and Groups:

- Organize IAM users into groups with distinct roles (admin, data analyst, etc.).
- Implement multi-factor authentication (MFA) for enhanced security.

IAM Policies:

- Clearly document IAM policies for different roles.
- Use IAM conditions to enforce security controls based on specific contexts.

Ideal Configurations for Each Component

S3 Bucket:

- Block public access to the S3 bucket.
- Implement versioning and logging for S3 bucket.
- Configure lifecycle policies for data retention.

AWS Glue:

- Set up AWS Glue crawlers to automatically infer the structure of the data.
- Utilize Glue triggers for job scheduling.
- Optimize Glue job configurations for resource utilization.

Amazon Athena:

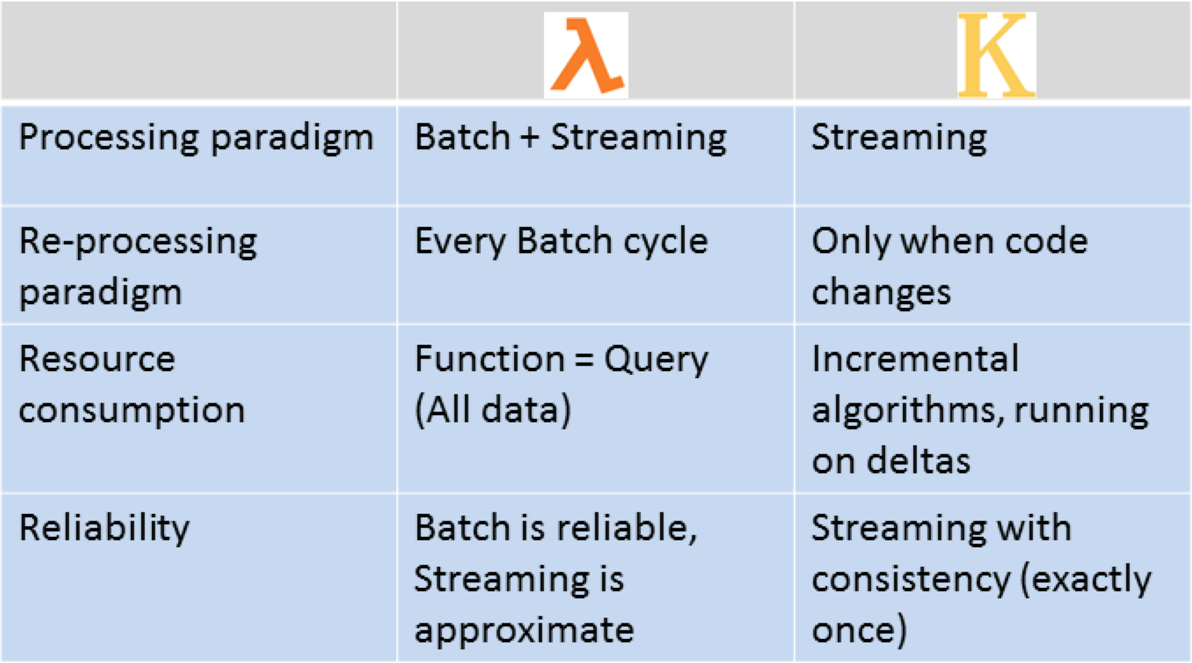
- Create Athena workgroups to organize and manage queries.
- Implement query execution time limits for resource optimization.
- Define query result location for efficient retrieval.

Amazon QuickSight:

- Configure QuickSight user permissions based on roles.
- Implement data source encryption and access controls.
- Schedule refreshes for data consistency.

Design Pattern

Batch pipeline pattern is chosen since Openfisheries only updates annually at most since the data is about yearly fish catches. The API was last updated during 2018 since the data ends after 2018. Hence the data remain consistent throughout the years with no changes, making it unnecessary to waste resources maintaining a live connection to the API (streaming) constantly fetching the same data. Thus, Batch is chosen as the data will be transformed to be used for analytics and stored at the Pipeline's data warehouse – The Pipeline only runs once and stores data, then streams the stored data as an API for data analysts.



	λ	K
Processing paradigm	Batch + Streaming	Streaming
Re-processing paradigm	Every Batch cycle	Only when code changes
Resource consumption	Function = Query (All data)	Incremental algorithms, running on deltas
Reliability	Batch is reliable, Streaming is approximate	Streaming with consistency (exactly once)

Lambda Architecture is designed to handle both batch and streaming layers separately through its batch and speed layers. We will use the Lambda architecture and utilize its functions.

Here are some advantages of Lambda architecture:

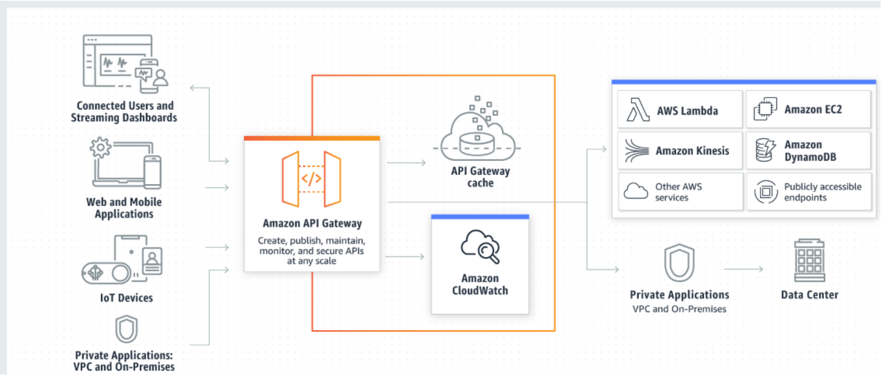
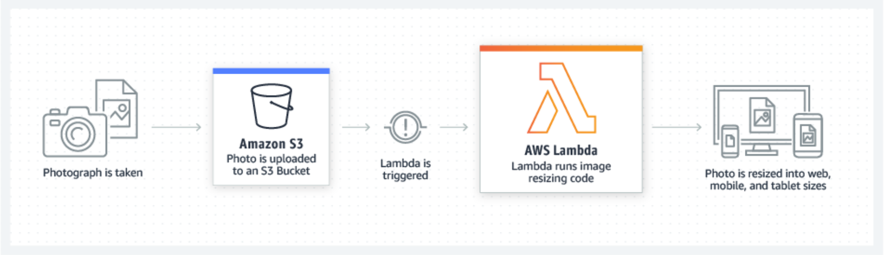
- **Scalability**: The Lambda Architecture is designed to be able to scale horizontally, allowing each layer in the architecture to be able to scale independently based on the data volume, velocity, and variety as well as to the needs of the stakeholders.
- **Fault-tolerance:** Lambda architecture is designed to be fault-tolerant, with multiple layers and systems working together to ensure that data is processed and stored reliably.
- **Flexibility:** Lambda architecture is flexible and can handle a wide range of data processing workloads, from historical batch processing to streaming architecture.
- **Cost effective:** Lambda architecture follows a pay-per-use format which can make it more cost-efficient when used wisely and efficiently. It also utilizes reduced server management and efficient data storage making it even more financially conserving
- **Performance:** Lambda architecture has low latency due to the serverless nature which prevents the need for a server startup time. Additionally, lambda function has parallelization ability which allows it run concurrently, making it efficient for large dataset.
- **Security:** Lambda architecture also has in-built security features like IAM and encryption to enhance data protection and access control.

Architecture Process

Step	Description
1. Pull data from OpenFisheries API	Use lambda function to extract the JSON data from REST API. Use API Gateway as a trigger to run the Step functions and activate the Lambda functions when the REST API receives an update.
2. Error Handling	If API endpoint cannot be reached or does not exist, retry for x specified tries. If still unable, CloudWatch activates Amazon SNS to send email to engineers about the issue. Else if data retrieval was successful, continue.
3. Dump data used in Data Lake	For data that are being used, dump the raw files into S3 bucket for future auditing purposes. The S3 Bucket will be frozen using S3 Glacier Deep Archive to conserve expenses since the Data Lake is unlikely to be used. This is done after the pipeline is done running as the last step of pipeline (using Step function).
4. Data Wrangling	Use Lambda function to extract data from Data Lake to perform data cleaning and transformation, to prepare data in a columnar format that can be stored in Data Warehouse. Alternatively, SageMaker can be used to perform Wrangling if ML is needed, since ML can be easily performed in SageMaker using Jupyter Notebook + ML Services in SageMaker.
5. Store wrangled data	Store in an S3 Bucket. This bucket will also be frozen as the last step of the pipeline.
6. Upload to Redshift (optional)	This is expensive and only needed if a large volume of data is in the pipeline or if complex SQL queries are needed to answer complicated business questions. Redshift is specifically designed for OLAP, hence optimized to handle large volumes of datasets and complex SQL queries with efficient runtime; better Velocity. Redshift is not compulsory, but utilizing the indexing strategies in Redshift can lead to better query performance. Glue is used to automatically infer the tables' structure based on the S3 files to create the database schema before Glue crawls the data from the S3 files and stores it in the Redshift database.
7. Exploratory Data Analysis (EDA)	Use Crawler to extract relevant data from Data Warehouse onto Athena and QuickSight (optional) to perform EDA and BI to find insights into the data and answer business questions. After completing EDA, preserve the insights by using SQL queries on Athena to select the relevant data and export it into a new S3 Bucket.

8. Store relevant data to S3 Bucket	Store SQL query results to answer business questions/contains insights into a csv file, then store these files into a S3 Bucket.
9. Destroy all Redshift instances (if used)	Crucial to use the Step function to delete all Redshift instances hosted after relevant data is extracted into S3 because Redshift is extremely costly and instances should not be left running when it's not used.
10. Deploy API	Use API Gateway to create an Endpoint for each file in the S3 Bucket that needs to be used for analysis. This enables analysts to access data to create visualizations to report findings and insights regarding business.
11. Archive Data Lake and Data Warehouse data	Since these files no longer need to be used at all, archive them to S3 Glacier Deep Archive to save money since it's cheaper than S3. Can also choose to delete these buckets, but it is good to keep them in an event where they are needed for traceability, etc.
12. Securing API Endpoint	Create authentication methods like Token or SSH for users to access the API. This ensures that only users authorized to access data can access the data, preventing other unauthorized guests from flooding the API calls and bursting the limit.
Orchestrate and Automate pipeline	Use Step functions to automate the pipeline such that it is reusable and can repeatedly be run simply.
Admin to run the pipeline	Ensure only authorized employees can run the pipeline.
Security and Accountability of Actions	Use IAM to manage and restrict permissions and access to services within the pipeline, to ensure the engineer only has access to the service they oversee.
Error handling and Pipeline Monitoring	Robust error-handling mechanisms, like try-catch loop and retrying, were in place to handle potential issues that may arise during the operation of the pipeline gracefully. If an issue or error occurs at any step of the pipeline, engineers will be notified with an alarm about the issue using Monitor measures to send email notifications to engineers. Engineers can monitor any potential arising issues in the services using CloudWatch. For instance, if the API endpoint is unable to be reached or is unable to fetch data, the Lambda function will retry again. If still unable to, engineers will be notified.
Scalability	Services used are well-prepared to be scalable for large volumes of data. The pipeline also has an additional measure of AutoScaling to automatically scale and adjust based on the volume and consumption detected in the pipeline.

Architecture Components

Service	Reason
API Gateway	Based on official Amazon documentation:  Connected Users and Streaming Dashboards Web and Mobile Applications IoT Devices Private Applications: VPC and On-Premises Amazon API Gateway Create, publish, maintain, monitor, and secure APIs at any scale API Gateway cache Amazon CloudWatch AWS Lambda Amazon EC2 Amazon Kinesis Amazon DynamoDB Other AWS services Publicly accessible endpoints Private Applications VPC and On-Premises Data Center How it can be applied for our Pipeline: 1. Configure a trigger to automate and run the Lambda function pulling data from OpenFisheries Rest API endpoint. 2. Publish our API to let users perform Data Analytics on the OpenFisheries using the data stored at Redshift. Creating an API fulfils the project requirements for actual implementation where data can be streamed to end-users just like how we accessed it in OpenFisheries, except that the data is formatted in columnar and prepared to be used for analytical purposes. Kinesis Data Firehose was also considered as a service for data capture and ingestion. It provides simplicity, streamlining the pipeline, and is optimized for efficient ingestion of data streams. However, Kinesis Data Firehose was not selected as the security consideration required to implement our own security measures for authenticating and authorizing GitHub API access within its configuration and lacks the centralized management and monitoring capabilities of API Gateway. The optimization of ingesting data streams does not prove to be beneficial as OpenFisheries only updates annually at most.
AWS Lambda	 Photograph is taken Amazon S3 Photo is uploaded to an S3 Bucket Lambda is triggered AWS Lambda Lambda runs image resizing code Photo is resized into web, mobile, and tablet sizes AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. The layer usually contains library dependencies, a custom runtime, or configuration files. AWS Lambda can be used for a wide range of tasks, including data transformation, real-time stream processing, and event-driven computing. Furthermore, it allows the user to write custom code to process data, giving the user more control over the handling of incoming data.

<table>
<tr>
<td></td>
<td>

Moreover, Lambda scales automatically in response to the number of incoming events, and only charges based on the number of incoming events and the execution time of your functions.

Pushing Lambda to S3 instead of using Kinesis Data Firehose:
Given our data engineering solution, using AWS Lambda over Kinesis Data Firehose is preferred despite Kinesis having parallel processing, voluminous data handling, and just-in-time data transformation. Lambda functions can be used for complex data transformations, filtering, and enrichment. This allows us to write custom code to parse, clean, and format incoming data from various sources, while Kinesis Data Firehose only offers basic data transformation capabilities, such as compression and encryption. It is also not as flexible as Lambda for complex data transformations.

Since our focus will be data transformations and performing advanced analytics on the countries and species dataset in JSON files on the Openfisheries API, AWS Lambda would be a better choice, given that it provides greater flexibility and control over data processing.

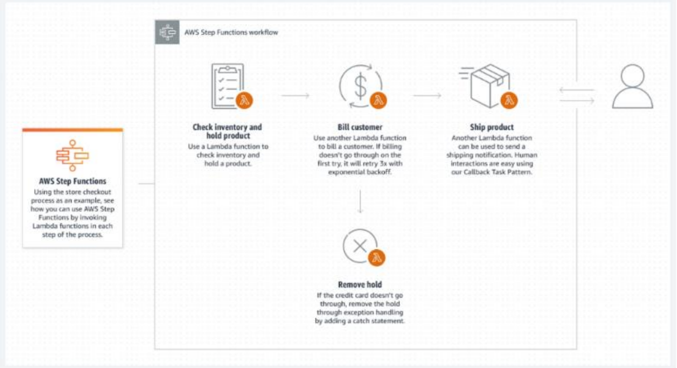
Furthermore, our primary focus will not be on ingesting large volumes of streaming or batch data. Therefore, despite Firehose enabling us to batch the data, which collects incoming data until a threshold is reached and writing it to S3, Lambda allows us to write to S3 directly. This makes the process more seamless as there will be less time delay when data is stored into S3.

</td>
</tr>
<tr>
<td>

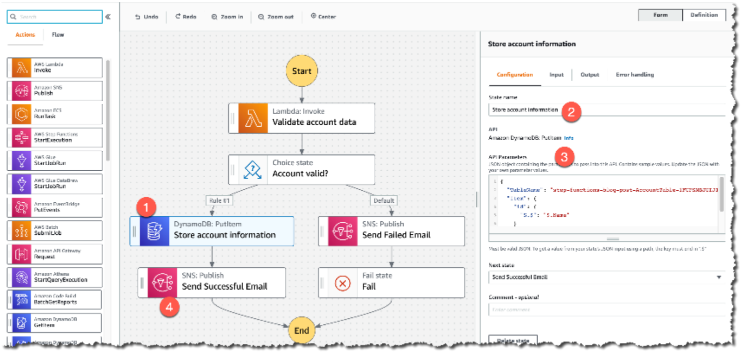
Step Functions

AWS Step Functions

</td>
<td>

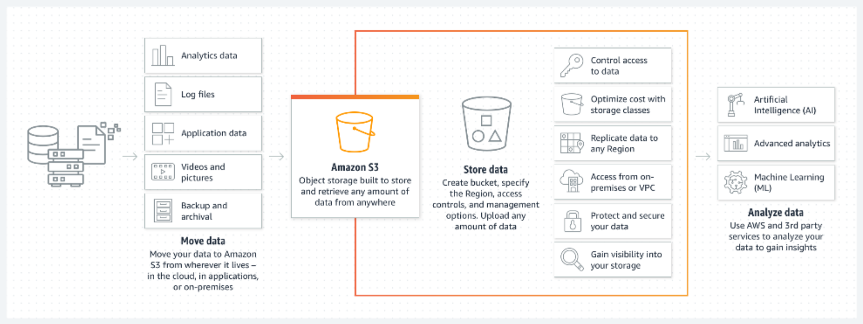

AWS Step Functions workflow
Check inventory and hold product
Use a Lambda function to check inventory and hold a product.
Bill customer
Use another Lambda function to bill a customer. If billing doesn't go through on the first try, it will retry 3x with exponential backoff.
Ship product
Another Lambda function can be used to send a shipping notification. Human interactions are easy using our Callback Task Pattern.
AWS Step Functions
Using the store checkout process as an example, see how you can use AWS Step Functions by invoking Lambda functions in each step of the process.
Remove hold
If the credit card doesn't go through, remove the hold through exception handling by adding a catch statement.

AWS Step Functions is a visual workflow service that helps developers use AWS services to build distributed applications, automate processes, orchestrate microservices, and create data and machine learning (ML) pipelines; In simple terms, Step Functions is used to orchestrate and automate a data pipeline. Below is a sample of how Step Functions is used:



</td>
</tr>
</table>

S3



Analytics data
Log files
Application data
Videos and pictures
Backup and archival
Move data
Move your data to Amazon S3 from wherever it lives – in the cloud, in applications, or on-premises
Amazon S3
Object storage built to store and retrieve any amount of data from anywhere
Store data
Create bucket, specify the Region, access controls, and management options. Upload any amount of data
Control access to data
Optimize cost with storage classes
Replicate data to any Region
Access from on-premises or VPC
Protect and secure your data
Gain visibility into your storage
Artificial Intelligence (AI)
Advanced analytics
Machine Learning (ML)
Analyze data
Use AWS and 3rd party services to analyze your data to gain insights

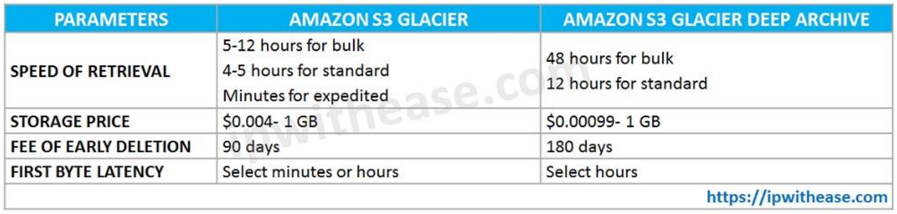
Amazon S3 (Simple Storage Service) is an object storage service created to store and retrieve any amount of data from anywhere. It offers high durability of 99.99%, along with industry-leading scalability, availability, performance, and security at very low costs. We can utilize S3 to securely store any volume of data for different purposes, including backups, archives and big data analytics.

S3 has reliable Security as S3 buckets are initially accessible only to their creator, except for IAM policy grants. Access permissions can be set at various levels (file, bucket, IAM), ensuring precise control and preventing unauthorized access. S3 has very low costs since we pay for only what we use, with prices starting at $0.022 per GB and even lower at $0.0125 per GB for infrequent access.

Comparing S3 with Google Cloud Storage, S3 is better due to the numerous data migration options available, where migration can be done via rsync or glacier interface. This results in a transition that will minimize disruptions to businesses, and beneficial for large data transfers. Automatic import and export features of S3 save costs and time. S3's latency is better than Google Cloud Storage, because a new HTTP connection needs to be established for each file resulting in speeds three times slower than Amazon S3.

Glacier

Like S3 but for non-active files to save money since it is an archive repository. It is a low-cost storage service designed for long-term data archiving and backup. It is optimized for data that is infrequently accessed but may need to be retained for compliance, regulatory, or business reasons.



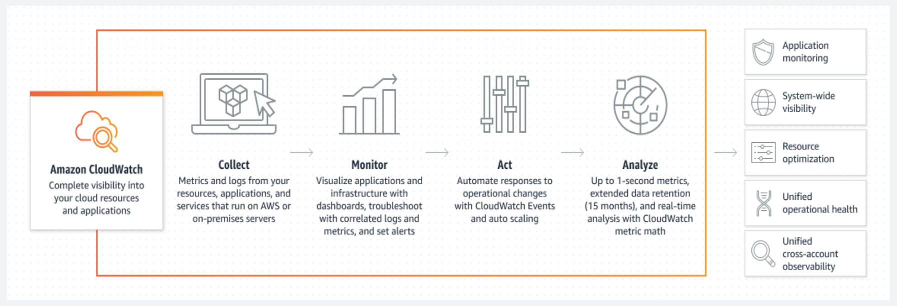
PARAMETERS	AMAZON S3 GLACIER	AMAZON S3 GLACIER DEEP ARCHIVE
SPEED OF RETRIEVAL	5-12 hours for bulk 4-5 hours for standard Minutes for expedited	48 hours for bulk 12 hours for standard
STORAGE PRICE	$0.004- 1 GB	$0.00099- 1 GB
FEE OF EARLY DELETION	90 days	180 days
FIRST BYTE LATENCY	Select minutes or hours	Select hours

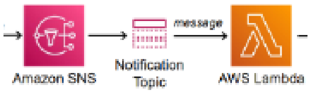
https://ipwithease.com

Glue

AWS Glue is a fully managed ETL service that simplifies data preparation and loading for analytics. With a few clicks in the AWS Management Console, we can create and run ETL. Glue automatically discovers and stores the associated metadata in the AWS Glue Data Catalog, making it easily available to search and query for ETL. Benefits of Glue include helping to automate data discovery process by crawling data sources. Identifying the schema as data discovery can be time-consuming, especially with large amounts of data across multiple sources. Data cleaning and

	preparation is made easy as Glue helps clean and prepare data with its built-in transformations. Glue helps migrate on-premises data to Amazon S3 as part of cloud migration strategies, avoiding the need to rewrite ETL jobs.
Data Wrangling	For Data Wrangling, Lambda is preferred as it's a versatile service within AWS that can easily be integrated into a STEP function to orchestrate and automate the pipeline's data flow. To set up the lambda function to wrangle, write code to perform tasks like data cleaning and parsing it into csv format, before performing transformations to prepare the datasets into fact and dimension table format. An alternative to consider *when Machine Learning is needed* is SageMaker, as this service contains a Jupyter Notebook with ML features like Code Whisperer for a pleasant experience in performing ML. Localhost (on user's PC) Jupyter Notebook can also be used, but using this option would make the wrangling process hard to be integrated and flow with the data pipeline.
Redshift Amazon **Redshift**	A Data Warehouse is needed to store data in columnar format to be usable for data analysis like running SQL queries and create visualizations to answer business questions logically and intuitively. The services considered were between RDS and Redshift as the data should be Structured to be used for analysis. Hence, Document databases (JSON) like DynamoDB were not considered. Both RDS and Redshift are reliable and scalable SQL relational databases. Redshift is twice as expensive as RDS, but Redshift is specifically designed for OLAP and optimized for LARGE datasets (with parallel processing capabilities), and since the pipeline should be prepared to support large amounts of data when deployed (based on project specs), Redshift is chosen as it delivers better performance for complex queries and runs faster when populated with extensive data. Redshift stores data in columnar format (enabling fact and dim tables) and integrates well with BI visualization tools, while RDS is designed for traditional OLTP (row-based) to store transactional data. Therefore, RDS not ideal for a pipeline meant for analytics but Redshift is meant for analytics.
RedShift API (How the data will be accessed to be used for analytics)	Users can access their Amazon RedShift Database using the built in Amazon Redshift Data API. Using this API, users can access Amazon Redshift data with web services–based applications, including AWS Lambda and AWS Cloud9. The Data API doesn't require a persistent connection to your database. Instead, it provides a secure HTTP endpoint and integration with AWS SDKs. Users then can use the endpoint to run SQL statements without managing connections. Users can then call the data API using the AWS Command Line Interface (AWS CLI), from their code, or using the query editor in the Amazon Redshift console.

API Gateway	To allow users to access data through an API Gateway connected to a Redshift database, there must be some configurations made first. A user must first be authorized. The user can be authorized to access the Data API by adding a managed policy, which is predefined by AWS Identity and Access Management (IAM), to that user. It is recommended to attach permission policies to an IAM role before assigning it to users and groups as needed. After, we need to ensure that the Redshift database is accessible and properly configured to interact with the API Gateway. After that, the user can extract the data from the database using the API gateway.
Cloudwatch and AutoScaling (monitor status and health of pipeline) Amazon CloudWatch	 Amazon CloudWatch Complete visibility into your cloud resources and applications Collect Metrics and logs from your resources, applications, and services that run on AWS or on-premises servers Monitor Visualize applications and infrastructure with dashboards, troubleshoot with correlated logs and metrics, and set alerts Act Automate responses to operational changes with CloudWatch Events and auto scaling Analyze Up to 1-second metrics, extended data retention (15 months), and real-time analysis with CloudWatch metric math Application monitoring System-wide visibility Resource optimization Unified operational health Unified cross-account observability CloudWatch is needed to optimize the performance of the data architecture by identifying and addressing performance bottlenecks in data processing, workflows, databases, and storage by collecting and visualizing metrics, such as CPU utilization, memory usage, and disk I/O and ensuring that they are running smoothly. AWS CloudWatch allows the setting up of alerts by using CloudWatch Alarms, based on our predefined thresholds which enables a quick response to any possible issues. By leveraging the metrics and alarms created by CloudWatch, our data pipeline can dynamically adapt to varying workloads which optimizes our resource allocation. This enhances our performance while contributing to saving costs efficiently by managing resources in response to our analyst's actual demand With the use of AWS AutoScaling and AWS CloudWatch Alarms, they can be used to help optimize resource usage and scale infrastructure based on user demand which helps save money.
QuickSight	QuickSight excels as a visualization tool due to its straightforward method of use, scalability, and comprehensive data integration capabilities. Users benefit from its intuitive user interface, fast learning curve, and ability to handle large datasets without performance degradation, due to its serverless architecture and in-memory engine.

	The platform seamlessly connects directly with other AWS services, making it convenient in this scenario of the pipeline construction and ensures data security through built-in controls. Collaboration features enable real-time sharing and exploration of insights, while its cost-effective pricing model, machine learning-powered insights and mobile accessibility.
SNS and Lambda (notify issues with Pipeline)  	Amazon Simple Notification Service (SNS) is a communication service providing message delivery between publishers and consumers using logical access points known as a topic. Benefits of SNS include a wide range of supported endpoint types, the First In, First Out (FIFO) feature for a strict messaging order and to prevent message duplication. SNS is lightweight, fully managed message queue and topic service that can scale almost infinitely while providing simple and easy to use APIs. In our context, we will be connecting Lambda functions downstream from the SNS. Using this approach, AWS Lambda can invoke with the payload of the published message and send the message to other AWS services. Lambda can also remedy the issue by triggering functions to perform automated actions based on application logs or metrics. An alternative we have considered is Amazon Simple Query Service (SQS). It was not selected as our notification system as the messages are not delivered instantly like SNS and requires additional code and infrastructure for message processing.
IAM (Control access of pipeline)	For ensuring data security and compliance with regulations such as GDPR, the chosen Identity and Access Management (IAM) solution plays a crucial role in providing access permissions and supporting data governance. The selected IAM system offers complete control over individual user roles, is a mature and well-integrated access control service, and facilitates audit trails through integration with AWS CloudTrail for logging and compliance. It is cost-effective and supports offline access management. However, for larger-scale pipelines, it may be complex, utilizing resource-based policies. Additionally, the IAM system is considered more robust and feature-rich compared to the alternative resource-based policy and service catalog options. The resource-based policy has a declarative approach for easy auditing but may be prone to human error, while the service catalog offers standardized deployment with straightforward user access but is less flexible and requires a more complicated initial setup. The choice depends on the specific needs and priorities of the data security and governance requirements.
KMS (Protection of data within pipeline)	Amazon Key Management Service (KMS) offers a robust encryption solution for data at rest and in transit, which ensures its security throughout the data engineering pipeline. With KMS, we can encrypt data in storage services like S3 or Redshift, as well as encrypt data in transit between services using managed keys, safeguarding it from interception. The service

AWS KMS	provides a centralized key management service, streamlining key creation, access control, and minimizing the risk of key misplacement or leakage. It also provides granular access controls that allow for detailed permissions, reducing the risk of unauthorized access to sensitive data. KMS supports key rotation and maintains an audit trail of key usage activities for security monitoring. Lastly, it also helps to regulate data laws such as the personal data protection act (PDPA) in Singapore.

Cost Explorer was used to compare services to determine which is more cost-effective for the project.

Deployment Method: AWS step function

1. Define the entire pipeline as code to ensure consistency and version control.
2. Declare the infrastructure to handle underlying provision and configuration to make deployment declarative and manageable.
3. Use parameters in the template to control configuration such as database name, resource size, and API keys -> Use step set to manage deployment across multiple AWS accounts.
4. Upon failure, CloudFormation will allow rollback to the previous state.
5. Finally, CloudFormation will be integrated with the Continuous integration and continuous delivery (CI/CD) pipeline and will trigger deployments automatically upon a change of code or scheduled intervals.

Contingency alternatives (Lambda)

Possible issues	Contingency plan
Duplicated data	- Identify types of duplicates & impacts (i.e. complete duplication where all values are the same or partial duplication) - Identify volume & frequency of duplicates - Data governance & compliance (i.e. form Regulations or policies dictating handling of duplicates) - Drop duplicated rows
Missing value in data	- Consider the type of missing data: Missing Completely At Random (MCAR) vs. Missing At Random (MAR) vs. Missing Not At Random (MNAR) requires different approaches. - Choose the appropriate imputation method such as K-Nearest Neighbour (KNN) which predicts missing values based on similar data points OR Simple imputation method of Mean/Median/Mode OR Impute NaN or 0 OR Drop rows with missing values if not suitable imputation. - Deep learning methods: Utilize autoencoders or other neural networks to impute missing values by learning from the data distribution.
Data inconsistencies	- Define inconsistencies (i.e. invalid format, misspelling) - Fix the inconsistency with the appropriate method.
Anomalies	- Identify anomalies by checking unique values and spotting illogical values. - Drop anomalous records OR fix with appropriate methods like imputation.
Outliers	- Calculate standard deviation and z-scores: Identify data points deviating significantly from the mean (beyond a predefined threshold) as potential outliers. - Similar to missing value imputation, different levels of imputation such as KNN or Mean/Median/Mode can be used to fill the outlier values.