**Dynamic Airspace Monitoring: Real-Time Visualization**

**Using Snowflake, Amazon Kinesis Data Firehose, and Streamlit**

In this project, real-time commercial flight data over the San Francisco Bay Area, sourced from the [OpenSky](https://opensky-network.org/) Network, is utilized to demonstrate an innovative solution integrating Snowflake, [Amazon Kinesis Data Firehose (ADF)](https://aws.amazon.com/firehose/), and Streamlit. This combination facilitates efficient data handling, analysis, and visualization in real time, which is essential for dynamic and data-intensive applications like air traffic monitoring.

**Project Overview:**

The San Francisco Bay Area, a major hub for both commercial and private aviation, presents complex challenges in air traffic management. Effective monitoring of this airspace requires the real-time processing and visualization of large volumes of flight data. This project harnesses the power of the OpenSky Network to capture detailed flight data, including aircraft positions, altitudes, and speeds.

**Data Integration and Processing:**

Amazon Kinesis Data Firehose is used for the real-time ingestion of flight data. It captures, transforms, and reliably loads streaming data into data lakes and analytics platforms, minimizing latency. Snowflake provides a scalable cloud data warehouse, offering the computational power needed to analyze large datasets rapidly.

**Visualization with Streamlit:**

Streamlit is incorporated as the front-end visualization tool, enabling the creation of interactive dashboards that display live flight data. This allows users to explore data through intuitive interfaces and gain insights into air traffic trends, patterns, and anomalies.

**Architecture Highlights:**

1. ***Data Capture:*** Live flight data for all aircraft over the San Francisco Bay Area is captured via the OpenSky API.
2. ***Data Ingestion and Analysis:***

* *Amazon Kinesis Data Firehose:* Streams data in real time, applying necessary transformations before loading it into Snowflake.
* *Snowflake:* Stores the processed data and performs complex analytical queries without sacrificing performance.

1. ***Visualization:***

* *Streamlit:* Provides a user-friendly web app that visualizes real-time data with interactive graphs and maps, enhancing the data interaction experience and providing actionable insights at a glance.

THE ARCHITECTURE DIAGRAM

A diagram of a diagram of a computer

Description automatically generated with medium confidence

**Operational Efficiency:**

The seamless integration of Kinesis Data Firehose with Snowflake optimizes data flow from ingestion to analysis, supporting data volume scalability and reducing the need for manual interventions. Streamlit’s integration further enables the dynamic visualization of this data, promoting better user engagement and faster decision-making processes.

**Providing a Linux Jumphost in AWS**

* A Linux EC2 instance (jumphost) will be provisioned in the subnet of an AWS VPC. The Linux jumphost will host the data producer that ingests real-time flight data into the Firehose delivery stream.
* The data producer calls the data sources' REST API and receives time-series data in JSON format. This data is then ingested into the Firehose delivery stream and delivered to a Snowflake table.

1. **CREATING THE EC2 INSTANCE:**

An EC2 instance was successfully launched using AWS CloudFormation to serve as a jumphost for processing and analyzing real-time flight data captured over the San Francisco Bay Area. This instance facilitates the initial data capture and preprocessing steps integral to the subsequent analysis in Snowflake and visualization in Streamlit.

**Details of Implementation:**

**Launch:**

The EC2 instance was initiated through AWS CloudFormation, tailored to efficiently handle and process real-time data.

**Region:**

The AWS region selected for deploying the EC2 instance was us-west-2 (Oregon). This region is optimal for the data sources and the services used in this project.

**Subnet Configuration:**

*Subnet Selection****:*** An existing subnet within the VPC was selected for deploying the EC2 instance. The choice between public or private subnets was made based on the desired network layout and accessibility considerations, ensuring optimal connectivity and security.

*Security Group:* The default security group of the VPC was utilized for this instance. This group was configured to ensure robust security and efficient traffic management, which is crucial for the integrity and reliability of the data processing tasks.

**Stack Configuration:**

The CloudFormation stack was named appropriately to align with the project's naming conventions and to facilitate easy identification and management within the AWS environment.

**Execution:**

The setup process was completed by executing the stack creation, which involved provisioning the EC2 instance with the configuration settings.

1. **CONFIGURATION OF LINUX SESSION AND SHELL PREFERENCES ON EC2 INSTANCE**

To configure the shell and session timeout settings for the EC2 instance launched via AWS CloudFormation, ensuring optimal performance and security for running real-time data ingestion tasks.

**Procedures Executed:**

* **AWS Systems Manager Setup:**
  + Accessed the AWS Systems Manager console within the us-west-2 region, aligning with the EC2 instance setup.
  + Navigated to Session Manager from the left pane to configure instance settings.
* **Shell Configuration:**
  + In the Preferences tab, the Edit button was clicked to modify session settings.
  + Under General preferences, the idle session timeout was set to 60 minutes to prevent premature terminations of idle sessions, which could disrupt ongoing tasks or analyses.
  + Scrolled to the Linux shell profile section and set the default shell to /bin/bash by entering /bin/bash in the provided field. This ensures that bash is used for session commands, leveraging its advanced features and compatibility with most data processing scripts.
* **Saving Preferences:**
  + Clicked the Save button to apply the new configuration settings.

1. **Connecting to the Linux EC2 Instance**

To establish a secure and stable connection to the EC2 instance for real-time data processing activities.

**Procedure Executed:**

* Returned to the Session tab and initiated a new session by clicking the Start session button.
* Located the EC2 instance named -jumphost under Target instances, which was provisioned by the CloudFormation template.
* Selected the -jumphost instance and successfully started the session, thereby accessing the Linux console of the EC2 instance.

1. **Creation of Key-Pair for Snowflake Authentication**

To generate a secure key-pair for authenticating with Snowflake, ensuring encrypted communication and data transfer.

**Procedures Executed:**

1. **Key-Pair Generation:**
   * Navigated to the home directory ($HOME) on the EC2 instance.
   * Executed the command to generate a private key:

During this process, an encryption password was prompted and set, which is critical for later stages of key usage.

1. **Public Key Creation:**
   * Generated the corresponding public key using:

Re-entered the previously set encryption password to authorize the public key generation.

1. **Key Formatting:**
   * Extracted and formatted the public key to remove headers, footers, and newlines, making it suitable for configuration inputs:
   * Applied the same formatting to the private key:

The EC2 instance is now fully configured with optimal session and shell settings, and a secure key-pair has been generated and formatted appropriately for Snowflake authentication. These steps are critical in establishing a secure and efficient environment for real-time data processing and analysis using AWS and Snowflake technologies. This documentation ensures transparency, security, and reproducibility of the setup within the project framework.

**Preparing the Snowflake cluster for streaming**

To create and configure a dedicated Snowflake environment that includes the necessary user roles, database, and warehouse for the project. This setup enables the secure and efficient handling of streaming data from the EC2 instance.

**Procedures Executed:**

* 1. **Initial Configuration**
* Logged into the Snowflake account using an account with an ACCOUNTADMIN role to ensure sufficient privileges for the setup tasks.
* Set multiple variables to streamline the creation of user, database, and warehouse.
  1. **Role and User Creation**
  2. **Database and Warehouse Steup**
  3. **Granted Permissions**
  4. **Account Identifier Retrieval**
  5. **Public key configuration**
  6. **Schema Creation**

The Snowflake environment is now fully configured with all necessary components properly set up: a user, role, database, warehouse, and schema, along with secure RSA public key authentication for the user. This configuration ensures that the project is ready for secure and efficient real-time data processing and analysis. The setup procedures are documented to provide clarity and support any required audits or future modifications.

**Creating an ADF Delivery stream**

To establish a delivery stream in Amazon Kinesis Data Firehose that securely streams real-time data into Snowflake, utilizing Amazon PrivateLink for secure data transfer.

**Procedure Executed:**

1. *Accessing the ADF Console:*

Navigated to the Amazon Kinesis Data Firehose (ADF) console and initiated the creation of a new delivery stream.

1. *Configuring the Source:*

For the source of the data, "Direct PUT" was selected. This choice allows data to be sent directly to the Firehose without going through another AWS service first.

1. *Setting the Destination:*

Snowflake was selected as the destination for the data stream to leverage its capabilities for large-scale data analysis and management.

1. *Naming the Delivery Stream:*

A descriptive name was provided for the Firehose stream to easily identify its purpose and association with the Snowflake destination.

1. *Specifying Snowflake Details:*

The Snowflake account URL was set to a PrivateLink endpoint, enhancing security by ensuring that data travels within the AWS network and not over the public internet. This URL was obtained by running a specific SQL command in Snowflake that fetches the PrivateLink URL.

1. *Authentication Configuration:*

The username STREAMING\_USER and the corresponding RSA private key were specified for authenticating with Snowflake. The private key was retrieved from the EC2 instance where it was stored, and the passphrase used during its creation was entered to complete the setup.

1. *Role and Schema Settings:*

A custom Snowflake role ADF\_STREAMING\_RL was chosen to define permissions specifically tailored for the delivery stream's operations.

The Snowflake database ADF\_STREAMING\_DB, schema ADF\_STREAMING\_SCHEMA, and the destination table ADF\_STREAMING\_TBL were defined. These settings ensure that the data is correctly routed within Snowflake.

1. *Data Loading Configuration:*

Configured the data loading options to use JSON keys as table column names, which organizes the incoming JSON data into structured columns within the specified Snowflake table.

1. *S3 Backup Setup:*

An existing S3 bucket was selected for backing up logs and error messages. This step is crucial for maintaining data integrity and troubleshooting potential issues in the data streaming process.

1. *Finalizing the Stream Creation:*

Completed the setup by reviewing all configurations and launching the delivery stream. The process typically takes around 5 minutes, after which the stream is active and ready to receive data.

The Amazon Kinesis Data Firehose delivery stream is now fully operational, securely streaming real-time data into Snowflake. This setup not only enhances data transfer efficiency but also strengthens the overall security posture by using Amazon PrivateLink. The detailed and strategic configuration of the delivery stream ensures that the data is appropriately managed and stored in Snowflake, ready for real-time analysis and visualization using Streamlit.

**Ingest and Query Data in Snowflake**

**Snowflake Table Creation and Data Ingestion**

To create a destination table in Snowflake to receive real-time data from an Amazon Kinesis Data Firehose delivery stream and transform this raw data into a structured format for analysis.

**Steps Executed:**

1. *Destination Table Setup:*

A table named ADF\_STREAMING\_TBL was created within Snowflake under the database ADF\_STREAMING\_DB and schema ADF\_STREAMING\_SCHEMA. This table was designed to store raw flight data with all fields initially set as VARCHAR or NUMBER types to accommodate incoming JSON formatted data.

1. *Data Streaming:*

A Python script was utilized on the EC2 instance to fetch real-time flight data and stream it directly into the ADF\_STREAMING\_TBL table via the configured delivery stream. This script facilitated the continuous ingestion of data, populating the table with fresh data as it was generated by the source.

1. *Initial Data Query:*

To confirm that data was successfully streamed into Snowflake, a simple SQL query was executed to retrieve records from ADF\_STREAMING\_TBL. The successful execution of this query verified that the delivery stream was correctly configured, and that data was populating the table.

**Transformation of Raw Data into Structured View**

To convert the raw data stored in ADF\_STREAMING\_TBL into a more structured and analytically useful format using a view within Snowflake.

**Steps Executed:**

1. *View Creation:*

A view named flights\_vw was created to transform the data from ADF\_STREAMING\_TBL. This transformation included converting field types from strings to more specific data types like integers and floats where appropriate and adjusting timestamps to a specific timezone.

1. *Data Enrichment:*

Additional calculations were performed as part of the view creation, including:

* The generation of geographic hashes (Geohashes) for each record to facilitate location-based querying and visualization.
* The calculation of distances from a fixed point (in this case, San Francisco Airport) to provide insights into each flight's proximity to this location at the time of data capture.

1. *Structured Data Query:*

Post-view creation, a query was executed against flights\_vw to display the newly structured data. This query confirmed that the view not only presented the data in a more useful format but also ensured that all transformations and calculations were performed accurately.

The Snowflake setup for handling real-time data through an Amazon Kinesis Data Firehose delivery stream was successfully implemented. This setup included the creation of a raw data table and a structured view that transformed this data into a format suitable for advanced analytical tasks. The process was documented thoroughly to ensure clarity and facilitate future audits or modifications.

**Streamlit for Visualizing Real-Time Flight Data**

To develop a Streamlit application that connects to Snowflake, retrieves real-time flight data, and visualizes the number of arrivals by destination airport using an interactive bar chart.

**Tools and Libraries Used:**

* **Python**: Programming language used for scripting the application.
* **Streamlit**: Open-source app framework for Machine Learning and Data Science teams.
* **Pandas**: Data analysis library used to manipulate data and perform calculations.
* **Snowflake** Connector for Python: Library used to query data from Snowflake directly from Python.
* **Matplotlib**: Plotting library used for creating static, interactive, and animated visualizations in Python.

**Implementation Details:**

1. *Environment Setup:*

* Installed Python and Streamlit along with Pandas and the Snowflake connector to handle data retrieval and operations efficiently.
* Configured the Streamlit environment to run the application script.

1. *Connecting to Snowflake:*

Established a connection to Snowflake using credentials and connection parameters specific to the project's Snowflake setup:

**User: STREAMING\_USER**

**Password: Confidential**

**Role: ADF\_STREAMING\_RL**

**Account: Specific to the project (e.g., gncpega-sfb41498)**

**Warehouse, Database, and Schema were all specified according to the project structure.**

1. *Data Retrieval:*

* Executed a SQL query through the Snowflake connector to fetch real-time data related to flight arrivals, focusing on the destination airport and the timestamp of arrival.
* The data retrieved was loaded into a Pandas DataFrame to utilize powerful data manipulation capabilities provided by the library.

1. *Data Transformation:*

* Converted the timestamps from the SQL query results to Python datetime objects for accurate time-series manipulation and visualization.

1. *Visualization with Streamlit and Matplotlib:*

* Utilized Matplotlib to create a bar chart representing the number of arrivals at each destination airport.
* Customized the chart with specific colors for each airport, set up axis labels, and formatted the title and tick parameters to enhance readability and aesthetic appeal.
* Integrated the visualization into the Streamlit application to provide an interactive user interface that displays the bar chart dynamically based on the real-time data.

1. *Streamlit Application Interface:*

* The application displays a bar chart titled "Arrivals by Destination Airport", which dynamically updates as new data streams in.
* The chart uses a custom color scheme to differentiate between airports and includes formatted labels and titles for clarity.
* Users can interact with the data through the Streamlit interface, which refreshes to reflect the most current data available in Snowflake.

The Streamlit application successfully visualizes real-time flight data stored in Snowflake, providing insightful and interactive data presentations that enhance decision-making processes. The setup effectively demonstrates the integration of Python scripting, Snowflake data handling, and Streamlit visualization, serving as a robust model for similar real-time data visualization needs.

**Conclusion:**

This project exemplifies the powerful synergy between Snowflake, Amazon Kinesis Data Firehose, and Streamlit, illustrating a comprehensive real-time data processing and visualization system. The architecture not only boosts operational efficiencies but also supports proactive decision-making and maintains high safety standards in the heavily trafficked airspace of the San Francisco Bay Area. This case study serves as a benchmark for leveraging advanced cloud and visualization technologies to enhance real-time analytical capabilities in air traffic monitoring and other critical fields.