Audibene

Mai Elzayat

June, 2024

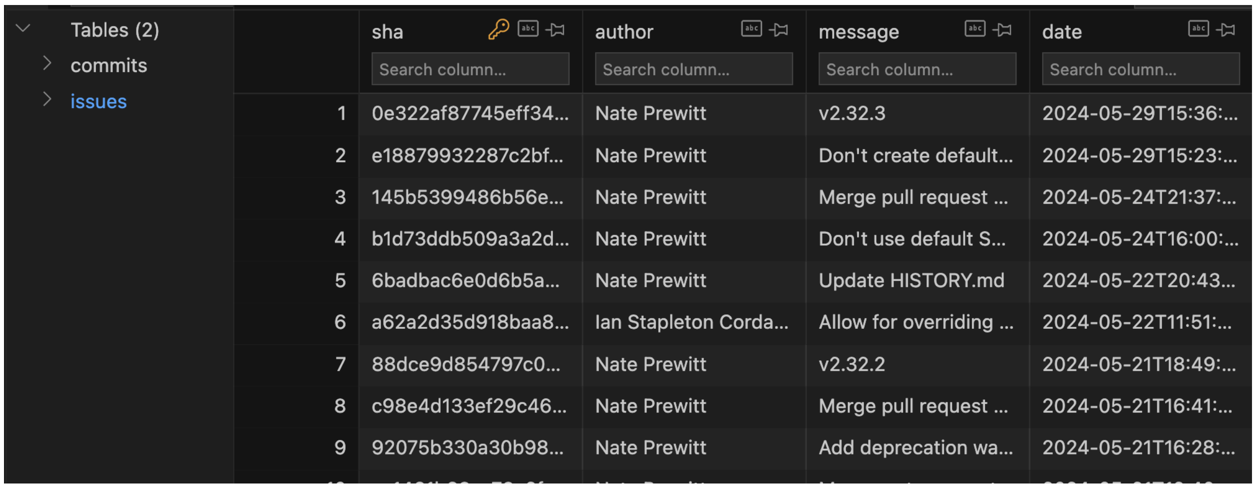
Overview:

- This document further explains and illustrates the logic behind the steps done in the project as well as provides two system architectures for the last part (system design) alongside the benefits and challenges of using the specified tech stack.
- The README file provides step by step instructions for setting up and running the project
- The output result sqlite and github data generated by the python script and dbt models are included in the repo to show the outputs of the task, however you can ignore them when running the project to have fresh results generated out of the scripts.

Dataset and Python Code

- The GitHub access token is used to authenticate with the GitHub API and avoid rate limiting. It can be stored as an environment variable or in a separate file, depending on the project's requirements. We are using environment variables for simplicity.
- Please make sure to create a GitHub API access token, you can follow this link for reference.
- The project follows best practices of code separation. The `utils.py` script contains reusable functions for fetching data from the GitHub API, creating a SQLite connection, creating tables, and inserting data into the database.
- The `api_data.py` script uses the functions from `utils.py` to fetch (commits and issues) data from the GitHub API and store it in the SQLite database

api_data.py output:



Tables (2)
commits
issues

	sha	author	message	date
1	0e322af87745eff34...	Nate Prewitt	v2.32.3	2024-05-29T15:36:...
2	e18879932287c2bf...	Nate Prewitt	Don't create default...	2024-05-29T15:23:...
3	145b5399486b56e...	Nate Prewitt	Merge pull request ...	2024-05-24T21:37:...
4	b1d73ddb509a3a2d...	Nate Prewitt	Don't use default S...	2024-05-24T16:00:...
5	6badbac6e0d6b5a...	Nate Prewitt	Update HISTORY.md	2024-05-22T20:43:...
6	a62a2d35d918baa8...	Ian Stapleton Corda...	Allow for overriding ...	2024-05-22T11:51:...
7	88dce9d854797c0...	Nate Prewitt	v2.32.2	2024-05-21T18:49:...
8	c98e4d133ef29c46...	Nate Prewitt	Merge pull request ...	2024-05-21T16:41:...
9	92075b330a30b98...	Nate Prewitt	Add deprecation wa...	2024-05-21T16:28:...

DBT

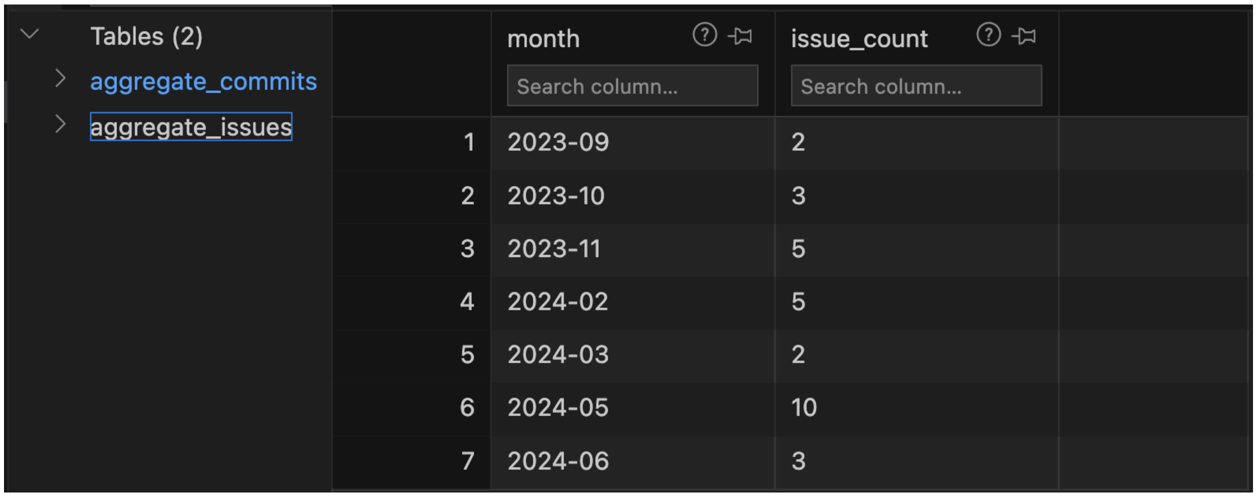
- Adaptors installed:
  - Dbt-core ‘*v=1.8.2*’
  - Dbt-sqlite ‘*v=1.1.0*’
- Project name init: “data-dbt-analytics”
- The project contains 5 main files as follows:

File name	Function
dbt_project.yml	Define project configs
aggregate_commits.sql	Model for github commits data
aggregate_issues.sql	Model for github issues data
src_github.yml	Define models sources and docs
dt_github.yml	Testing

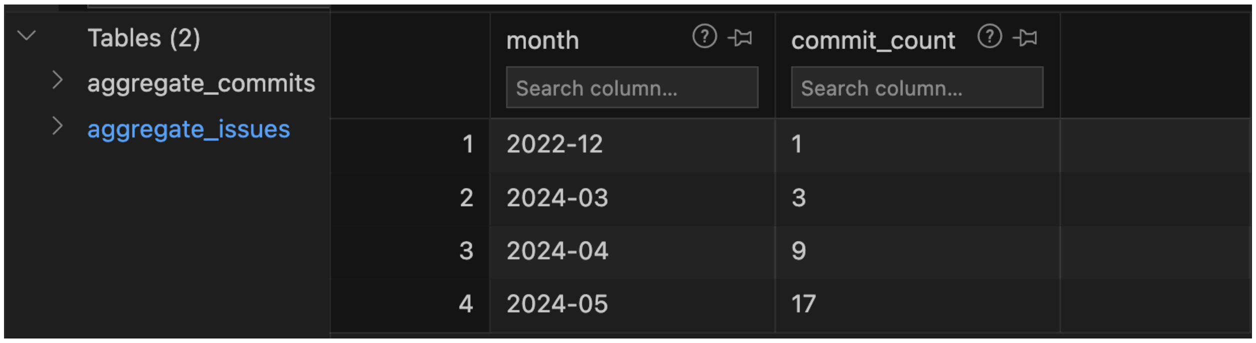
- The DBT models (`aggregate_commits.sql` and `aggregate_issues.sql`) aggregate the commit and issue data by month using the `strftime` function and the `GROUP BY` clause.

- As SQLite does not use schemas there was no need to use the `ref` and `source` functions since they require a schema to be able to run the model so it was more efficient to write a simple SQL query.
- Two yml files:
    - src_github.yml → sources the tables inside the sqlite db and provides overall documentation about the model
    - dt_github.yml → provides tests and more comprehensive documentation about the model data, columns description
- Data_schemas directory is needed for sqlite to save the generated models inside the dbt project.

DBT models output:



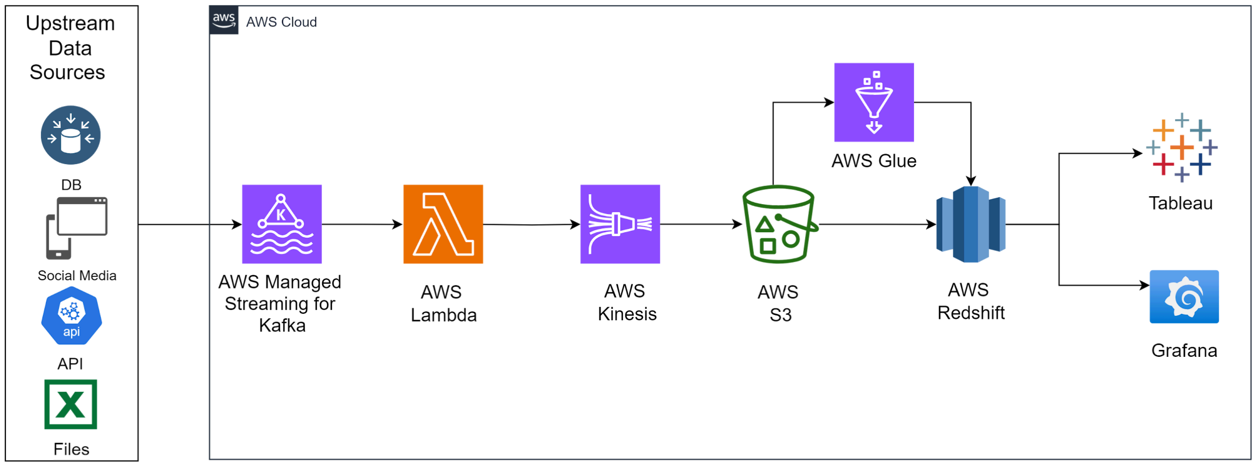
	month	issue_count
1	2023-09	2
2	2023-10	3
3	2023-11	5
4	2024-02	5
5	2024-03	2
6	2024-05	10
7	2024-06	3



	month	commit_count
1	2022-12	1
2	2024-03	3
3	2024-04	9
4	2024-05	17

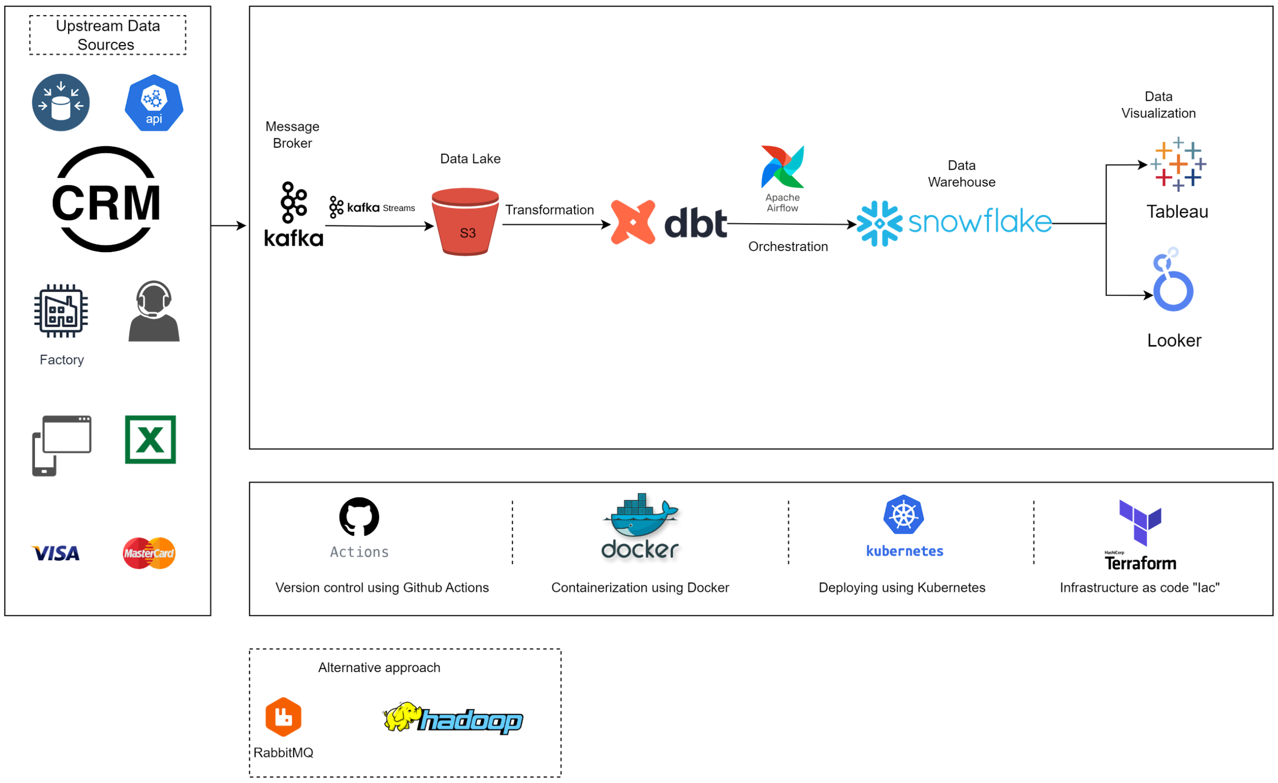
System Design

AWS Cloud Architecture:



Component	Description
Upstream Data Sources	Various data sources including databases, social media platforms, APIs, and flat files.
AWS Managed Streaming for Kafka	**Ingests real-time data** streams from upstream sources, providing high throughput and low latency.
AWS Lambda	Processes data streams from Kafka, applying transformations, filtering, and routing. Scales automatically.
AWS Kinesis	Further processes and manages streaming data, enabling real-time analytics. Ensures scalability and durability.
AWS S3	Stores processed data from Kinesis. Acts as a **data lake** with scalable and durable storage for both structured and unstructured data.
AWS Glue	Performs ETL operations on data stored in S3, preparing it for analysis. Loads **transformed data** into Redshift.
AWS Redshift	Stores structured data optimised for analytical queries. Supports complex queries on large datasets.
Tableau & Grafana	Connect to Redshift to provide interactive dashboards and visualisations. Enable users to derive insights and make data-driven decisions.

Cloud Stack Architecture:



Component	Description
Kafka Message Broker	Manages and routes messages from upstream data sources to downstream consumers, ensuring reliable message delivery.
Kafka Streaming	Processes real-time data streams from various data sources, enabling real-time analytics and event-driven applications.
AWS S3	Stores processed data from Kafka. Acts as a **data lake** with scalable and durable storage for both structured and unstructured data
DBT	- **Load:** data from S3 'Data lake' to Snowflake 'Data warehouse' - **Transformation:** of data inside data warehouse to produce data marts
Airflow	**Orchestration**: for DBT project and other use case pipelines, managing the execution order and dependencies.
Snowflake	**Data Warehouse:** where data is loaded using dbt and later transformed into layers such as: - Base / Staging - Intermediate - Core - Data Marts

Github Actions	- **DBT version control:** (Pre-hooks for sql fluff / dbt-commits) - **Docker ECR:** image build/linting - push to ECR - **Kubernetes:** application run
Terraform	**Infrastructure as code:** Deploying the infrastructure using terraform leveraging the use of states, and keeping history on of infrastructure status - Troubleshooting and restoring states at failure points.
Tableau & Looker	Connect to Snowflake to provide interactive dashboards and visualizations. Enable users to derive insights and make data-driven decisions.

Scalability and Benefits:

Aspect	AWS Architecture	Cloud Stack Architecture
Scalability	- AWS Lambda: Auto-scales with incoming events.	- Kafka: Scales with brokers and partitions.
	- AWS Kinesis: Adds shards for horizontal scaling.	- DBT & Snowflake: Scales with Snowflake's resources.
	- AWS Redshift: Scales by adding nodes.	- Airflow: Scales by adding worker nodes.
		- GitHub Actions: Scales with CI/CD runners and Kubernetes.
Benefits	- AWS Lambda: Serverless, cost-efficient.	- Kafka: Real-time processing, low latency.
	- AWS Glue: Managed ETL service.	- DBT: Simplifies transformations, integrates with Snowflake.
	- AWS Redshift: High-performance data warehouse.	- Airflow: Flexible orchestration, integrates with various tools.
	- Tableau & Grafana: Powerful visualisation tools.	- Terraform: Infrastructure as code, supports multi-cloud.

Organisational and Technical challenges:

Technical Challenges:

- **Data Integration and Compatibility**: Ensuring that data from diverse sources (databases, social media, APIs, files) is compatible and can be integrated seamlessly.
    - **Solution**: Use data transformation tools and standardise data formats during ingestion.
- **Infrastructure Scalability**:
    - **Challenge**: Managing the increasing volume of data, varying data structures, and ensuring low-latency processing.
    - **Solution**: Implement scalable cloud-based services like AWS Managed Streaming for Kafka, AWS Lambda, and AWS Kinesis, which automatically scale with demand.
- **Data Storage and Management**:
    - **Challenge**: Efficiently storing and managing large volumes of structured and unstructured data.
    - **Solution**: Use AWS S3 as a scalable data lake and AWS Redshift for structured data warehousing.
- **Data Security and Privacy**:
    - **Challenge**: Ensuring data security and compliance with privacy regulations.
    - **Solution**: Implement strong security measures (encryption, access controls) and ensure compliance with regulations like GDPR.
- **Data Consistency and Quality**:
    - **Challenge**: Maintaining data consistency and quality across various sources and transformations.

    - **Solution**: Implement data validation and cleansing processes, monitoring and alerting for data quality regularly.

Organisational Challenges:

- **Change Management**:
    - **Challenge**: Managing resistance to change from employees accustomed to legacy systems.
    - **Solution**: Communicate the benefits of the new system clearly and involve key stakeholders in the implementation process.
- **Stakeholder Alignment**:
    - **Challenge**: Aligning various stakeholders (IT, data analysts, business units) with the project goals.
    - **Solution**: Foster collaboration through regular meetings, clear communication, and defining shared objectives.
- **Budget Constraints**:
    - **Challenge**: Managing the budget for implementing and maintaining the new system.
    - **Solution**: Plan a phased implementation to spread costs over time and demonstrate value early to secure additional funding if needed.
- **Data Governance**:
    - **Challenge**: Establishing data governance policies to manage data quality, security, and usage.
    - **Solution**: Develop and enforce comprehensive data governance frameworks and policies.
- **Project Management**:
    - **Challenge**: Coordinating multiple teams and tasks to ensure timely and successful project delivery.

    - **Solution**: Use agile project management methodologies and tools to track progress and facilitate communication.
- **User Adoption**:
    - **Challenge**: Ensuring that end-users adopt the new tools and processes.
    - **Solution**: Provide training, support, and create user-friendly documentation to facilitate adoption.