



Technical Documentation

URED U: Final Phase

Prepared by: iOLAP

Document Status: FINAL

Version: 5.1.3 | DATE \@ "d MMMM yyyy" 14 September 2023



Contents

1 Introduction	2
2 Architecture overview	3
3 AWS S3	4
3.1 Data	4
3.2 Scripts	5
4 AWS DynamoDB	6
4.1 Overview	6
4.2 Table Attributes	6
4.3 Use Case: Efficient Glue Job Execution	6
4.4 Conclusion	7
5 AWS Redshift	7
6 AWS Glue	11
7 Secrets Manager	12
7.1 Overview	12
7.2 Stored Information	12
8 AWS Lambda	12
8.1 Overview	12
8.2 Function Configuration	12
8.3 Lambda Workflow	12
8.4 Usage	13
8.5 Security and Permissions	13
8.6 Conclusion	13
9 DevOps	13
9.1 CloudFormation	13






1 Introduction

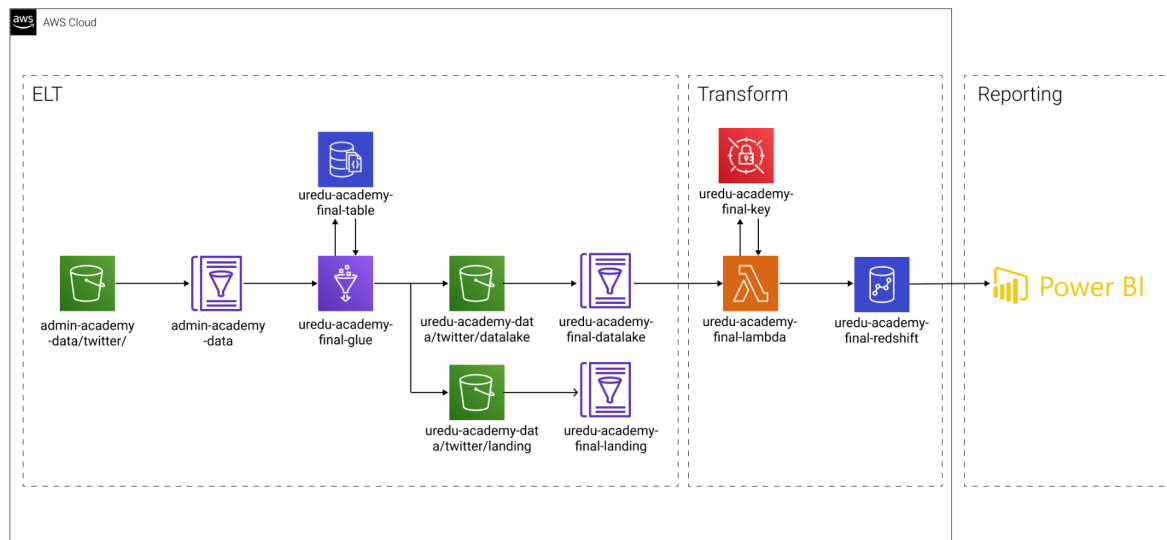
The goal of this document is to provide insight into the technical aspects of the teams Uredu final assignment.

The client, Societum News wants to develop engaging news articles and successful marketing strategies to enable growth of their company.

The research object is twitter data, with main focus on finding social groups and their behavior. After completing a model for social group behavior, we will be able to predict what kind of news will attract most of the public, when it should be posted and how regularly for maximizing engagement and what social group should be targeted for gathering the broadest audience.

2 Architecture overview

Service	Description
	AWS S3 (Simple Storage Service)
	AWS DynamoDB
	AWS Glue
	AWS Redshift
	AWS Lambda



This architecture is built using ETL standards where data is extracted from a source, transformed using Glue job and then loaded into the Data Lake.

- Sources
 - Data is stored in *admin-academy-data* AWS s3 bucket
 - It is available in *admin-academy-data* AWS Glue Database
- Extraction
 - Data is extracted from *admin-academy-data* AWS Glue Database with *uredu-academy-final-glue* AWS Glue Job as a PySpark DynamicFrame
- Transformation
 - Data is converted from PySpark DynamicFrame Apache Spark DataFrame
 - Multiple columns are exploded to convert lists to rows, strings are cleaned and tables are connected to denormalized table
 - Data is converted from Apache Spark DataFrame to PySpark DynamicFrame
- Data Lake
 - Data is saved in the *uredu-academy-final-data* AWS s3 bucket as a Data Lake

3 AWS S3

Processing big amounts of data coming in from various source systems requires a storage option where data can be stored before or after it's processed. Amazon S3 (Simple Storage Service) provides a scalable, low-cost storage option. In the scope of this project, we are using Amazon S3 to leverage any artifacts that require storage.

To support easier navigation through S3, we are using multiple buckets where each bucket stores specific artifacts. Naming convention for S3 buckets is defined as:

- `<user>-academy-<final/null>-<artifact description>-<alphanumeric hash1>`
- *s3 buckets:*
 - `admin-academy-data`
 - `uredu-academy-final-data`
 - `uredu-academy-final-scripts`

Artifacts that require storage are defined as:

1. Data – Raw (pending processing or uploaded manually), Data Lake data
2. Scripts – Glue jobs, SQL scripts

3.1 Data

To distinguish data in an intuitive way, we defined multiple buckets which store various data. Additionally, each bucket contains keys (folders) to further distinguish sources of the stored data:

- `admin-academy-data`
 - `tweeter/`
 - `tweets/`
 - `user_followers/`

¹ The purpose of alphanumeric hashes is to enforce globally unique bucket identifiers



- users/
- **uredu-academy-final-data**
 - tweeter/
 - landing/
 - tweets/
 - user_followers/
 - users/
 - datalake/
 - tweets/
 - user_followers/
 - user_mentions/
 - hashtags/
 - users/

3.2 Scripts

To distinguish scripts in an intuitive way, we defined a bucket to store various scripts. This bucket is not actively used by Developers, it is used as a source for AWS Glue deployments and a storage for AWS Redshift scripts. Each deployment will overwrite the scripts stored in this bucket, while AWS Glue only reads from this bucket.

Additionally, each bucket contains keys (folders) to further distinguish scripts between AWS Glue, AWS Lambda and their dependencies:

- **uredu-academy-final-scripts**
 - glue/
 - lambda/

4 AWS DynamoDB

Amazon DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. DynamoDB offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data import and export tools.

4.1 Overview

The "uredu-academy-final-table" DynamoDB table is designed to facilitate efficient data processing and ingestion for the purpose of minimizing redundant processing during Glue job execution. This table contains two key attributes: "name" and "ingestion_dttm."

4.2 Table Attributes

1. "name" (Partition Key)

- Description: This attribute serves as the partition key for the DynamoDB table. It represents the partition name.
- Data Type: String

2. "ingestion_dttm"

- Description: This attribute stores the timestamp when data was last ingested or processed.
- Data Type: String

4.3 Use Case: Efficient Glue Job Execution

The primary use case for the "uredu-academy-final-table" is to optimize Glue job execution by tracking the last ingested partition. By leveraging the "ingestion_dttm" attribute, Glue jobs can determine the timestamp of the most recent data ingestion. This information enables Glue jobs to avoid redundant processing of previously ingested data, thus improving efficiency and reducing processing costs.

The Glue job queries the "uredu-academy-final-table" DynamoDB table to retrieve the "ingestion_dttm" value, representing the timestamp of the last data ingestion.

Using this timestamp, the Glue job can efficiently identify and process only the new data partitions or records that have been ingested since the last execution.

After successfully processing the new data, the Glue job updates the "ingestion_dttm" attribute in the DynamoDB table with the current timestamp, marking the completion of the ingestion process.

By implementing this workflow, the "uredu-academy-final-table" DynamoDB table effectively serves as a checkpoint system for Glue job executions, ensuring that data processing is streamlined and prevents the reprocessing of already ingested data.

4.4 Conclusion

The "uredu-academy-final-table" DynamoDB table, with its "tweet_dt" and "ingestion_dttm" attributes, plays a crucial role in optimizing data processing workflows within the AWS Glue environment. It enables efficient tracking of data ingestion, reducing computational overhead and costs associated with redundant processing. This documentation serves as a reference for users and developers to leverage the table effectively in Glue job configurations.

5 AWS Redshift

Amazon Redshift is a fully managed, data warehouse service in the cloud. It uses SQL to analyze structured and semi-structured data across data warehouses, operational databases, and data lakes. In scope of this project Amazon Redshift is used to create views. Amazon Redshift uses the Amazon Redshift Spectrum processing engine to make the data available as schemas in Redshift and query the data directly from the AWS Glue Data Catalog.

The next list of schemas is used in Amazon Redshift:

- uredu-academy-final-landing
- uredu-academy-final-datalake
- uredu-academy-final-schema

Schemas are located in the admin-academy-redshift workgroup, inside twitter_redshift_db database. The next image shows the layout of the schemas in Amazon Redshift:

External schema uredu-academy-final-landing consists of raw and untransformed data. This schema is used only to archive the original database. In this scheme, there are three tables:

- tweets
- users
- user_followers

user_followers columns, descriptions and their data types:

Column	Description	Data Type
user_id_str	String user identifier	string
followers_list	List of users user with user_id_str follows	datetime64[ns, UTC]

tweets columns, descriptions and their data types:



Column	Description	Data Type
id	Numeric Tweet identifier	int64
created_at	Date and time of Tweet creation	datetime64[ns, UTC]
user_id	Tweet author	int64
full_text	Tweet full text	string
hashtags	Hashtags used in Tweet	object
user_mentions	Users mentioned in Tweet	object
is_retweet	Tweet is a Retweet	bool
retweet_created_at	Date and time of Retweet creation	datetime64[ns, UTC]
retweet_timedelta_sec	Time elapsed between original Tweet and Retweet creation	Int64 (nullable)
retweet_from_user_id	Retweet author	Int64 (nullable)
retweet_from_tweet_id	Numeric Retweet identifier	Int64 (nullable)
is_reply	Tweet is a reply to another Tweet	bool
in_reply_to_status_id	Original Tweet ID if this Tweet is a reply	Int64 (nullable)
in_reply_to_user_id	Original Tweet's User ID if this Tweet is a reply	Int64 (nullable)
is_quote_status	Tweet is a quote	bool
favorite_count	Number of times a Tweet was favorited (liked)	int64
possibly_sensitive	Tweet contains sensitive content	bool
langid	Tweet language	string

users columns, descriptions and their data types:

Column	Description	Data Type
user_id	Numeric user identifier	int64
created_at	The UTC datetime that the user account was created on Twitter	datetime64[ns, UTC]
screen_name	Alphabetic user identifier	string
location	User submitted location	string
description	User submitted description	string
protected	User allows public access to his profile	boolean
verified	User is verified by Twitter (Verified Accounts)	boolean
followers_count	Number of accounts that follow the user	int64
friends_count	Number of accounts that the user follows	int64
listed_count	Number of public lists that this user is a member of	int64
favourites_count	The number of Tweets this user has liked in the account's lifetime	int64
statuses_count	The number of Tweets (including retweets) issued by the user	int64
is_croatian	Custom value indicating the user's location is Croatian	bool
clean_location	Custom value, aims to pinpoint a Croatian location	string

External schema uredu-academy-final-datalake consists of transformed data. Data from this schema is moved to uredu-academy-final-schema automatically inside the Lambda function. In this schema, there are six tables:



- user_followers table with unnested followers_list column
- user table
- tweets table without hashtags and user_mentions columns
- hashtags table
- user_mentions
- tweet_groups

user_mentions columns, descriptions and their data types:

Column	Description	Data Type
user_id	Numeric user identifier	int64
followers	All user followers	int64

This table is essential for identifying influencers and user groups within the social media platform. By tracking user mentions in tweets, we can identify popular individuals and groups that receive mentions, helping us understand trends in user engagement and conversations.

hashtags columns, descriptions and their data types:

Column	Description	Data Type
id	Numeric tweet identifier	int64
hashtag	Hashtag used in a tweet with a specific id	string

The hashtags table is crucial for tracking trends on the platform. It allows us to identify trending topics and understand user interests by analyzing the hashtags used in tweets. This information can be used to improve content recommendation algorithms and discover what is currently popular among users.

user_mentions columns, descriptions and their data types:

Column	Description	Data Type
id	Numeric user identifier	int64
user_mentions	Name of user mentioned in a tweet	string

This table is used for identifying influencers and groups. It is possible that more people will tag popular news channels and people as they have a better chance of them seeing the tweet.

hashtag_groups columns, descriptions and their data types:

Column	Description	Data Type
id	Numeric user identifier	int64
group_name	Name of a group for a selected hashtag	string

This table is used for grouping and getting a better insight into hashtags users use. Many different hashtags can represent the same interest group so it is important that they are properly grouped.

Along with tables from users-academy-final-datalake, users-academy-final-schema consists of views, too. Views used are the following:

- hashtag_group_users__v
- hashtag_group_tweets__v
- users_joined_dates__v
- group_counts__v

hashtag_group_users__v columns, descriptions and their data types:

Column	Description	Data Type
user_id	Numeric user identifier	int64
group_name	Name of group the user belongs to	string

The hashtag_group_users__v view is used to group users based on the interests they tweet about. It matches user IDs with the groups they are associated with, helping us identify users who share common interests. This information can be used to recommend users to follow or connect with others who have similar interests.

hashtag_group_tweets__v columns, descriptions and their data types:

Column	Description	Data Type
id	Numeric tweet identifier	int64
group_name	Name of group the tweet belongs to	string

The hashtag_group_tweets__v view is used for grouping tweets by their content. It matches tweet IDs with the hashtags present in the tweets, allowing us to categorize tweets based on their content and the user groups they are associated with. This information can be used for content analysis and filtering based on user engagement metrics.

users_joined_dates__v columns, descriptions and their data types:

Column	Description	Data Type
user_id	Numeric user identifier	int64
group_name	Name of user mentioned in a tweet	string
min	Date of user joining the group	datetime64[ns, UTC]

The users_joined_dates__v is used for finding the join date for each user and each group the user joined. It matches every user's ID with all the names of the groups he belongs to and the minimum date for each of the groups joined. This information can be used for content analysis and filtering based on user engagement metrics.

group_counts__v columns, descriptions and their data types:

Column	Description	Data Type
group_name	Name of group the tweet belongs to	string
tweet_count	Count of all unique tweets inside the group	int64
retweet_user_count	Count of all unique users that were retweeted in the group	int64
user_count	Count of all unique users that are active in the group	int64
custom_metric	Metric defined by $\text{retweet_user_count} * \text{user_count} / \text{tweet_count}$	int64

The group_counts__v view is used for basic statistical insight for each group. It matches tweet IDs with the group_name present in the hashtag_group_tweets__v, allowing us to count unique users, retweeted users and tweets that belongs to each group. Column custom_metrics is defined to provide insight to groups that have more action and reaction on each tweet.

6 AWS Glue

Glue Jobs are an essential part of our ELT process. They are used to *Extract* data from source systems, transform the data, and load it to the Data Lake.

The Glue job is using an extension of the PySpark Python dialect for scripting ELT jobs. Glue job is loading the data from admin-academy-twitter database and it's storing transformed data in S3 bucket uredu-academy-final-data.

Data Source: Glue DataCatalog

Data Target: S3 bucket

Job summary:

1. Data Processing
 - a. Before the data is being loaded, Glue Job checks if the data was processed. Which will reduce our runtime
2. Data Extraction
 - a. Glue job is extracting raw data from Glue DataCatalog
3. Data Transformation
 - a. Splitting full_text column
 - i. full_text column contains duplicated text, to address the duplicated text Glue Job splits the text so the duplication is removed
 - b. Unnesting user_mentions, hashtags and followers columns
 - i. Unnesting of the user_mentions, hashtags and followers columns provides structured data for the analysis
 - c. Data Cleaning
 - i. Removing empty and NaN values from user_mentions, hashtags and followers columns
 - ii. The columns user_mentions, hashtags and followers are removed and loaded as separate tables in the Glue DataCatalog

7 Secrets Manager

7.1 Overview

AWS Secrets Manager is a service designed to securely store sensitive information, such as Amazon Redshift database connection credentials. It simplifies and enhances security for applications, services, and resources by providing a centralized and secure location for storing secrets.

7.2 Stored Information

AWS Secrets Manager is commonly used to store the following connection details for Amazon Redshift:

- **user:** The username required to authenticate and access the Redshift database.
- **password:** The password associated with the specified username.
- **host:** The hostname or endpoint of the Amazon Redshift cluster.
- **port:** The port number to connect to the Redshift cluster (typically 5439 for Redshift).
- **dbname:** The name of the specific database within the Redshift cluster.

8 AWS Lambda

8.1 Overview

The AWS Lambda function used serves the purpose of automating data ingestion into Amazon Redshift by connecting to the database and executing SQL queries. The function is designed to streamline the data loading process, making it more efficient and less error-prone.

8.2 Function Configuration

Environment Variables

- **secretName:** The name of the AWS Secrets Manager secret containing Redshift database connection credentials.

Dependencies

- **Boto3:** The AWS SDK for Python is used to interact with AWS services, specifically AWS Secrets Manager.
- **Psycopg2:** A PostgreSQL adapter for Python is used to connect to and interact with the Amazon Redshift database.

8.3 Lambda Workflow

Secret Retrieval: The Lambda function begins by retrieving the database connection credentials (username, password, host, etc.) from AWS Secrets Manager using the provided secret name.

Database Connection: Using the acquired credentials, the function establishes a connection to the Amazon Redshift cluster.

Schema and Table Creation: The function creates external schemas and tables in Amazon Redshift for data ingestion and storage. This includes schemas such as "uredu-academy-final-datalake," "uredu-academy-final-landing," and "uredu-academy-final-schema."

Data Ingestion: The function executes SQL queries to populate the created tables with data from the "uredu-academy-final-datalake" schema. It iterates through a list of tables ("users," "tweets," "user_followers," "hashtags," "user_mentions") and transfers data into corresponding tables within the "uredu-academy-final-schema."

8.4 Usage

It is designed to automate the process of data ingestion and schema creation, reducing manual effort and ensuring data consistency in Amazon Redshift.

8.5 Security and Permissions

To ensure secure access to AWS resources, the Lambda function should have appropriate IAM roles and permissions. The role should allow access to AWS Secrets Manager for secret retrieval and access to Amazon Redshift for schema and table creation and data ingestion.

8.6 Conclusion

The AWS Lambda function simplifies and automates the data ingestion process into Amazon Redshift. By connecting to the database, creating schemas and tables, and populating them with data, it enhances the efficiency of data processing workflows, making them more reliable and cost-effective.

9 DevOps

This section provides DevOps technical documentation for the Uredu final project. DevOps aims to shorten the system development life cycle and provide continuous delivery with high software quality.

In this project DevOps provides continuous integration, continuous delivery, data integrity and system security.

9.1 CloudFormation

CloudFormation is used to deploy resources using predefined templates. This allows us to deploy resources consistently across environments.

Templates used in this project are stored in the code repository **uredu-academy-final** (cicd folder). All templates are in YAML format.

Following table maps used templates with the deployed stack in the correspondent environment.



Template	Cloudformation Stack	AWS Environment	Note
uredu-academy-final: cicd/codepipeline.yml	uredu-academy-final-codepipeline	Production	Pipeline that is triggered with every push to origin main
uredu-academy-final: cicd/glue.yml	uredu-academy-final-glue	Production	Deployed by pipeline
uredu-academy-final: cicd/dynamodb.yml	uredu-academy-final-dynamodb	Production	Deployed by pipeline
uredu-academy-final: cicd/s3.yml	uredu-academy-final-s3	Production	Deployed by pipeline
uredu-academy-final: cicd/lambda.yml	uredu-academy-final-lambda	Production	Deployed by pipeline
uredu-academy-final: cicd/secretsmanager.yml	uredu-academy-final-secretsmanager	Production	Deployed by pipeline