Week 8: Data pipelines with Redis, Python and PostgreSQL

Documentation of the Pipeline

This data pipeline aims to extract, transform, and load customer call log data from a CSV file into a PostgreSQL database. The pipeline is implemented in Python and utilizes the following technologies

i. **Pandas:-** to read and manipulate the data from the CSV file
ii. **Redis:-** to cache the data for faster retrieval
iii. **Psycopg2:-** to connect to the PostgreSQL database and execute queries

Pipeline Steps

The pipeline consists of three main steps

1. **Extract data:-** The data is extracted from the CSV file using the read_csv function from the Pandas library. The data is then cached in Redis for faster retrieval.
2. **Transform data:-** The data is retrieved from the Redis cache and transformed. This includes cleaning the data, structuring it into a more usable format, and formatting it appropriately.
3. **Load data:-** The transformed data is loaded into a PostgreSQL database. This includes creating a table to store the data and inserting the data into the table.

Best Practices

Here are three best practices used during the implementation of this pipeline

1. **Caching Data in Redis**

   By caching the data in Redis, the pipeline can retrieve the data much faster, reducing the overall runtime of the pipeline.

2. **Using Psycopg2 Cursor**

   By using the Psycopg2 cursor, the pipeline can execute multiple queries at once, improving the efficiency of data loading.

3. **Using Environment Variables for Credentials**

   Storing sensitive information like database credentials in environment variables instead of hard-coding them in the code adds an extra layer of security to the pipeline.

Recommendations for Deployment and Running the Pipeline with a Cloud-based Provider

Here are some recommendations for deploying and running this pipeline with a cloud-based provider

i. **Deploy to a Managed Service**
   Consider using a managed service like AWS Elastic Beanstalk or Heroku to deploy the pipeline. These services can handle deployment and scaling for you, freeing up time for development.
ii. **Use Docker Containers**
   Consider using Docker containers to package the pipeline and its dependencies into a single image. This makes it easy to deploy and run the pipeline on any cloud-based provider that supports Docker.

iii. Implement Continuous Integration and Deployment

Implement a continuous integration and deployment (CICD) pipeline to automate the deployment process. This can help catch errors early in the development process and make it easier to deploy updates to the pipeline.