CHAPTER 4 Data Ingestion : Extracting Data

Saturday, 20 September 2025 18:48

The ELT pattern is ideal design for data pipelines built for data analysis, data science and data products.

The Extraction + Loading part comes under Data Ingestion

This chapter deals with extraction of data from variety of source systems as well as data comes in many forms. So all which will be addressed while ingesting it.

Setting up Python Environment

- Create a virtual environment whenever you want to deal with scope as specific user and not globally.

  Python3 -m venv env

- Activate the virtual environment created above

  Source env/bin/activate

- Use pip to install libraries as it is the most shipped with python distributions
- Install configparser using pip, which will be used to read configuration information you will add to a file later.
- Create an empty file called pipeline.conf - the code samples in this chapter are added to it

  Touch pipeline.conf

Setting up Cloud File Storage

- Create a S3 Bucket (Simple Storage Device) in AWS
- Install boto3

  Pip install boto3

- We will use boto3 to interact with our S3 bucket and also creating IAM user getting configuration keys and storing them in a private file so that python scripts can read from them

- Create an S3 bucket

  Mydatapipelinebucket1

- Create an IAM user

  Data_pipeline_readwrite

- Give IAM roles and permissions of S3FullAccess and copy the credentials and paste it in pipeline.conf file

```
[aws_boto_credentials]
access_key = AKIAWQ5RFA7OCQNK22KQ
secret_key = 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
bucket_name = mydatapipelinebucket1
account_id = 448658737116
```

EXTRACTING DATA FROM MYSQL DATABASE

Extraction of data can be done in two ways

- **Full or Incremental extraction using SQL**
- **Binary Log (binlog) replication**

Binary log extraction is complex to implement and better suited to cases where the data volume of changes in source table is high or there is need for frequent data ingestion from MySQL source.

MySQL Workbench

Use this syntax to create table Orders in database -> datapipeline and insert some values

CREATE TABLE Orders (OrderId int, OrderStatus varchar(30), LastUpdated timestamp); INSERT INTO Orders VALUES(1,'Backordered', '2020-06-01 12:00:00'); INSERT INTO Orders VALUES(1,'Shipped', '2020-06-09 12:00:25'); INSERT INTO Orders VALUES(2,'Shipped', '2020-07-11 3:05:00'); INSERT INTO Orders VALUES(1,'Shipped', '2020-06-09 11:50:00'); INSERT INTO Orders VALUES(3,'Shipped', '2020-07-12 12:00:00');

- When we need to ingest either all or subset of columns from a MySQL table into data warehouse or data lake we can do by
    - Full Extraction

      In full extraction every specified columns will be ingested into the destination

      SELECT * FROM Orders;

    - Incremental extraction

      In this only the records from source table that have changed or been added since the last run of the job are extracted

      SELECT * FROM Orders WHERE LastUpdate > {{ last_extraction_run}};

      When incremental extraction is carried out, the destination table will have both the original as well as updated table. Which is useful during analysis and modifications.

Both full and incremental extractions can be performed with use of SQL queries and python scripts, install pymsql library

FULL EXTRACTION

- Create a file extract_mysql_full.py
- Import libraries
    - Pymysql - establishes connection between python scripts in vscode and sql queries run in workbench
    - Csv - converts extracted data into flat files for ease the extraction of files into s3
    - Boto3 - helps us to interact to s3 from vscode
    - Configparser - reads configuration information from the pipeline.conf file (credentials file)

```
import pymysql
import csv
import boto3
import configparser

# initialize a connection to mysql database
parser = configparser.ConfigParser()
parser.read("pipeline.conf")
hostname = parser.get("127.0.0.1", "hostname")
port = parser.get("127.0.0.1", "port")
username = parser.get("127.0.0.1", "username")
dbname = parser.get("127.0.0.1", "database")
password = parser.get("127.0.0.1", "password")
conn = pymysql.connect(host=hostname,
 user=username,
 password=password,
 db=dbname,
 port=int(port))
```

```
 port=int(port))
if conn is None:
 print("Error connecting to the MySQL database")
else:
 print("MySQL connection established!")
```

- Perform a full extraction to extract entire data in a csv file format.

```
m_query = "SELECT * FROM Orders;"
local_filename = "order_extract.csv"
m_cursor = conn.cursor()
m_cursor.execute(m_query)
results = m_cursor.fetchall()
with open(local_filename, 'w') as fp:
 csv_w = csv.writer(fp, delimiter='|')
 csv_w.writerows(results)
 fp.close()
m_cursor.close()
conn.close()
```

- Establish aws connection using boto library and push the data into s3 bucket

```
# load the aws_boto_credentials values
parser = configparser.ConfigParser()
parser.read("pipeline.conf")
access_key = parser.get("aws_boto_credentials",
"access_key")
secret_key = parser.get("aws_boto_credentials",
"secret_key")
bucket_name = parser.get("aws_boto_credentials",
"bucket_name")
s3 = boto3.client('s3',
aws_access_key_id=access_key,
aws_secret_access_key=secret_key)
s3_file = local_filename
# uploads the full extraction csv data into aws s3 storage
s3.upload_file(local_filename, bucket_name,
s3_file)
```

Extracting Data from REST API's

```
import requests
import json
import configparser
import csv
import boto3

lat = 42.36
lon = 71.05
lat_log_params = {"lat": lat, "lon": lon}
api_response = requests.get(
    "http://api.open-notify.org/iss-now.json",
    params=lat_log_params
)
response_json = json.loads(api_response.content)
all_passes = []
current_pass = []

# store the lat/log from the request
current_pass.append(lat)
current_pass.append(lon)

# store the ISS current latitude/longitude and timestamp
current_pass.append(response_json['iss_position']['latitude'])
current_pass.append(response_json['iss_position']['longitude'])
current_pass.append(response_json['timestamp'])
all_passes.append(current_pass)
export_file = "export_file.csv"
with open(export_file, 'w') as fp:
    csvw = csv.writer(fp, delimiter='|')
    csvw.writerows(all_passes)

# establish s3 connection from vscode to send data
parser = configparser.ConfigParser()
parser.read("pipeline.conf")
access_key = parser.get("aws_boto_credentials", "access_key")
secret_key = parser.get("aws_boto_credentials", "secret_key")
bucket_name = parser.get("aws_boto_credentials", "bucket_name")
```

```
s3 = boto3.client(
    's3',
    aws_access_key_id=access_key,
    aws_secret_access_key=secret_key
)
s3.upload_file(export_file, bucket_name, export_file)
```

Now S3 will have both data from mysql as well as rest api into csv formats.

Extracting Data from Streaming Platforms