Week 13: Cloud and ETL

This week will cover the transformation and load phase of the data pipeline (the T,L in ETL). These stages convert data from one format to another and store it. The purpose is to extract business intelligence from raw data.

Facebook stores it’s social graph in a large set of MySQL databases. From that, a pipeline reads MySQL's binlogs, which are published to a message service. Subscribers then can read changes to the social graph from the message service. Results are stored in data warehouses. There, it can be analyzed. This resembles an ELT pattern.

Amazon uses a data pipeline to determine how to schedule truck deliveries to warehouses. Machine learning algorithms generate datasets that predict the frequency and content of deliveries. There are multiple such algorithms. Their predictions cover different periods of time, and are delivered in csv format. The data pipeline accepts the ML algorithm's results and chooses how to aggregate them after validating their contents. The data is transformed into an SQL compatible dataset. End users receive completed predictions. This resembles an ETL pattern.

The Facebook example depicts a "streaming" service, where data is continuously ingested. The Amazon example depicts a "batched" service, where data is periodically generated at scheduled intervals.

Additional factors

1. Validation- each transformation of the data should validate it to be sure it is not corrupted and is clean. There should be no duplicate records. Fields (such as dates) should follow an expected format. The same number of records should be present before and after each transformation. The input dataset may have other requirements (non-empty, or not too big).
2. Monitoring - the "health" of the data pipeline can be observed.

This week's lab will cover how to store data in s3. You will be given account credentials for this so do not need to create your own AWS account. In the homework, this will run within airflow.

Also in the homework, you will compare open source tools with their equivalents in the AWS ecosystem.

Lab 1

Last week's lab configured dags in airflow in a hacky way. When helm is used, the right way to store dags is covered in the documentation.

```
helm show values apache-airflow/airflow -n airflow > values.yaml
```

Create a git token, then put it into a kubernetes secret.

```
kubectl create secret generic airflow-ssh-git-secret
--from-file=gitSshKey=/Users/dlambrig/git-token -n airflow
```

In values.yaml, go to the section "dags:". Then go to "gitSync:".

```
    enabled: false
    repo: https://github.com/dlambrig/umlF21.git
    branch: main
    rev: HEAD
    subPath: ""
    sshKeySecret: airflow-ssh-git-secret
```

After saving the values file, update the helm configuration.

```
helm upgrade --install airflow apache-airflow/airflow -n airflow -f values.yaml --debug
```

In a similar manner it is possible to use a persistent volume. Create a pv and pvc as done last week, and populate it using a centos container.

```
dags:
  persistence:
    enabled: true
    existingClaim: my-dags-pvc
    accessMode: ReadOnlyMany
```

Lab 2

In this lab, you will use the AWS account provided to the class to access the AWS S3 storage service. Access will be done in three ways:

1. console
2. CLI
3. REST API

With these tools you should be able to do the homework. In the homework, you will store the CSV file in AWS in an s3 bucket. This will be one of the jobs in the DAG.

First, get into the aws console. Go to this class's aws site. The username is: **test** and password is: **umlnumber1**.

From there you can explore the s3 "UML" bucket. There are other buckets, but only "UML" is accessible. This is the one you will use in this lab and in the homework.

You can also examine other AWS services, such as code pipeline and data pipeline. The account does not give you access to use those services, but you can read documentation and look at the user interface. These services perform the same functions as the open source tools we have used in the class. In the homework you will contrast those services with airflow and Jenkins.

Next, install the aws cli on your PC. Instructions for Windows, Mac, and Linux are here.

Then run "aws configure" and enter the access key, secret key, us-east-1 (default region), and "None" for output format.

The access key id is AKIAQEIRHNZHLVJ5LYMI
The secret access key is 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

You should now be able to run the aws cli.

In s3, everyone in the class uses the same bucket. A bucket is like a folder in a file system. We will put objects into the bucket. An object is any file, such as our CSV file. You need to give objects a name. Use your username as part of the object name, so you do not conflict with other student's work.

List buckets:

```
dlambrig@Dans-MacBook-Pro-2 ch04 % aws s3 ls
2018-03-04 12:53:47 UML
2019-06-25 15:14:31 dlambrig-kubernetes-s3
2013-11-08 22:18:25 karthikm
2021-03-07 22:26:19 uml2
2021-11-13 19:50:18 umlf21-pipeline-bucket
```

Write to a bucket:

```
aws s3 cp mysql_binlog.py s3://UML
```

List objects in a bucket:

```
dlambrig@Dans-MacBook-Pro-2 ch04 % aws s3 ls UML
2021-11-28 19:24:30       2384 mysql_binlog.py
2021-11-28 19:06:50        159 order_extract.csv
```

Delete object in a bucket:

```
dlambrig@Dans-MacBook-Pro-2 ch04 % aws s3 rm  s3://UML/mysql_binlog.py
delete: s3://UML/mysql_binlog.py
```

Finally, access AWS using the API from a python program.

Pull the latest changes from the class github repository using "git pull".

Modify file pipeline.conf to contain the following:

```
[aws_boto_credentials]
aws_access_key_id = AKIAQEIRHNZHLVJ5LYMI
aws_secret_access_key = 9WHsqOnVPa3x9LKt/EDO6mlwhH5Ke7/DBstDuRlG
bucket_name = umlf21-pipeline-bucket
account_id = 009162944078
```

You should be able to run the python program in your centos container. You may need to import more python libraries from your container.

```
kubectl exec -it task-pv-pod -n airflow -- /bin/bash
# pip3 install boto3
# pip3 install psycopg2-binary
```

Once the python libraries are downloaded, run the python program.

It will put the csv file into S3, where you will be able to view it using the aws cli.

```
[root@task-pv-pod opt]# python3 extract_mysql_incremental.py
MySQL connection established!
[root@task-pv-pod opt]# ls *csv*
order_extract.csv
[root@task-pv-pod opt]# rm -f *csv*
[root@task-pv-pod opt]# exit
exit
dlambrig@Dans-MacBook-Pro-2 umlF21 % aws s3 ls UML
2021-11-28 19:39:51        159 order_extract.csv
```