BigData Concepts

08 January 2024 09:01

Apache Airflow

- Introduction to Data Warehousing and Data Lakes
- Designing Data warehousing for an ETL Data Pipeline
- Designing Data Lakes for an ETL Data Pipeline
- ETL vs ELT
- Fundamentals of Airflow
- Work management with Airflow
- Automating an entire Data Pipeline with Airflow

Data center
DAS (Direct Attached Storage) { internal or external } --> JBOD (just bunch of disks) --> RAID (Redundant Array of Independent Disks) / Disk Arrays --> Storage Arrays (ICDA -- Intelligently cached disk arrays)
SAN we use Storage arrays where all your data resides for all type of data related activity and we allows only SAN protocols (iscsi, FCP, iFCP, FCIP, FCoE) to communicate with the nodes present in the SAN

Designing data warehousing for ETL

1. Understanding business requirements and objectives, Identify key stakeholders, data sources and the types of analyses and reports that will be generated
2. Structuring the data in the source system and align it with business needs
    a. Identify entities and attributes
    b. Normalize or de-normalize the data based upon analysis requirements
    c. Define relationship between tables
    d. Prepare the echo systems to incorporate measures and dimensions for analytics
3. Choose Data warehouse technology
    a. On-premises solution --> present inhouse - locally all data is available to immediate access --> Traditional relations database - oracle, MS SQL, Teradata
    b. Cloud-Based Solutions --> data available over the internet by the respective cloud vendor --> Amazon Redshift, Google BigQuery, Azure synapse analytics
4. Data integration --> selecting the ETL tools { Apache Nifi, Cloud-based ETL services }
5. Partitioning and Indexing --> partition tables based on commonly used query patterns and hence in order to improve the performance we implement indexing
6. Scalability ---> we can use the cloud based solutions which can offer scalability on-demand and we can also consider parallel processing by spreading or partitioning the data across multiple nodes.
7. Performance optimization --> optimize the SQL queries
8. Data security --> RBAC , Encrypt the data at rest and in transit, audit trails
9. Governance and metadata mgmt ---> track about the origin , usage pattens , transformation logics, enforce data quality standards
10. Monitoring and logging --> monitor ETL job execution , failures and durations, log details information's for auditing and troubleshooting
11. HA & DR --> backup and restore strategies, offsite copes, implementing failover and redundancy for all critical components
    a. RPO --> Recovery Point Objective --> Amount of Data loss
    b. RTO --> Recovery Time Objective --> Downtime of a resource
    In a well-managed HA & DR setup the RPO and RTO will be zero. ie., zero % downtime and zero % data loss. { to active this we need to eliminate all type of SPOF (single point of failures)
12. Documentations - maintaining comprehensive document, including data models, ETL processes, configurations and security polices and etc.,
13. Testing --> Thoroughly test the entire ETL pipeline (perform unit testing , integration testing, performance testing, validating data accuracy and qualiy
14. Continuous Improvement --> optimize based on feedback, technology changes or based up on the business requiments

Designing data Lakes for ETL

1. Data lake architecture
    a. Raw data storage i.,e allocating raw storage - unprocessed data
    b. Data catalog --> metadata management
    c. Compute resources
    d. Security framework

2. Data lake technology
    a. On-premises solution --> HDFS or other on-premises storage solutions
    b. Cloud-Based Solutions --> data available over the internet by the respective cloud vendor --> Amazon S3, Google Cloud Storage, Azure Dat lake storage

3. Data integrations --> { Apache Nifi, Cloud-based ETL services, AWS Glue, Azure data factory } or implement CDC (changed data caoture)
4. Schema design --> We can choose the schema that best suits our requirements, and utilize the file formats (eg., parquet , avro, orc etc.,,)

Login as hadoop user and switch user as root
apt install python3.8-venv
exit

Now in your hadoop homedir
$ python3 -m venv airflow-env
$ source airflow-env/bin/activate

$ pip install apache-airflow
$ su - root
# apt install mysql-server
# pip3 install "apache-airflow[mysql]"

mysql>CREATE DATABASE airflow CHARACTER SET utf8 COLLATE utf8_unicode_ci;
mysql>create user 'airflow'@'localhost' identified by 'airflow';
mysql>grant all privileges on * . * to 'airflow'@'localhost';
mysql>flush privileges; mysql>quit

```
root@mainserver1:/tmp# mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 10
Server version: 8.0.35-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> create database airflow character set utf8 collate utf8_unicode_ci;
Query OK, 1 row affected, 2 warnings (0.03 sec)

mysql> create user 'airflow'@'localhost' identified by 'airflow';
Query OK, 0 rows affected (0.08 sec)

mysql> grant all privileges on *.*  to 'airflow'@'localhost';
Query OK, 0 rows affected (0.02 sec)

mysql> flush privileges;
Query OK, 0 rows affected (0.02 sec)

mysql> quit
Bye
```

$ mysql -u airflow -p

```
root@mainserver1:/tmp# exit
logout
hadoop@mainserver1:~$ mysql -u airflow -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 8.0.35-0ubuntu0.20.04.1 (Ubuntu)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| airflow            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.03 sec)

mysql>
```

Install the celery connector

pip3 install 'apache-airflow[celery]'
exit

$ cd ~/airflow-project

$ source airflow-env/bin/activate
$ airflow db init
$ Create and demo_dag.py script and place in the to the appropriate place holder
$ vim demo_dag.py

```
from datetime import datetime
from airflow import DAG
from airflow.decorators import task
from airflow.operators.bash import BashOperator

# A DAG represents a workflow, a collection of tasks
with DAG(dag_id="demo", start_date=datetime(2022, 1, 1), schedule="0 0 * * *") as dag:
    # Tasks are represented as operators
        hello = BashOperator(task_id="hello", bash_command="echo hello i am $HOSTNAME working")

        @task()
        def airflow():
            print("airflow is working fine")

    # Set dependencies between tasks
        hello >> airflow()
```

<save and exit>
<note: take care about indentations >

Now Activate the Dag and view the statistics