Data Engineering Day 20

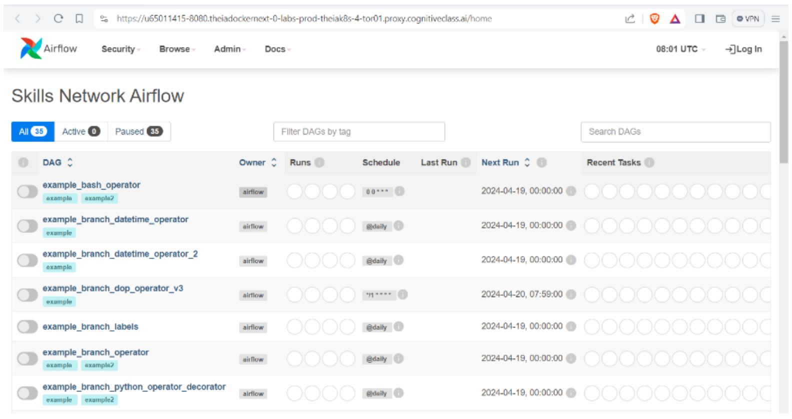
The credit for this course goes to Coursera. Click More

Another link : Azure data Engineer

ETL and Data Pipelines with Shell, Airflow and Kafka

# Getting started using Apache Airflow UI: Apache airflow docs

- The figure below represents the UI of Apache airflow.



DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
example_bash_operator (example, example2)	airflow		0 0 * * *		2024-04-19, 00:00:00	
example_branch_datetime_operator (example)	airflow		@daily		2024-04-19, 00:00:00	
example_branch_datetime_operator_2 (example)	airflow		@daily		2024-04-19, 00:00:00	
example_branch_dop_operator_v3 (example)	airflow		*/1 * * * *		2024-04-20, 07:59:00	
example_branch_labels	airflow		@daily		2024-04-19, 00:00:00	
example_branch_operator (example, example2)	airflow		@daily		2024-04-19, 00:00:00	
example_branch_python_operator_decorator (example, example2)	airflow		@daily		2024-04-19, 00:00:00	

- to perform etl tasks in airflow, we need to create DAGs and peform shell scriptings for automating the tasks. First of all
- create a file **my_first_dag.sh** copy and paste the code provided below.

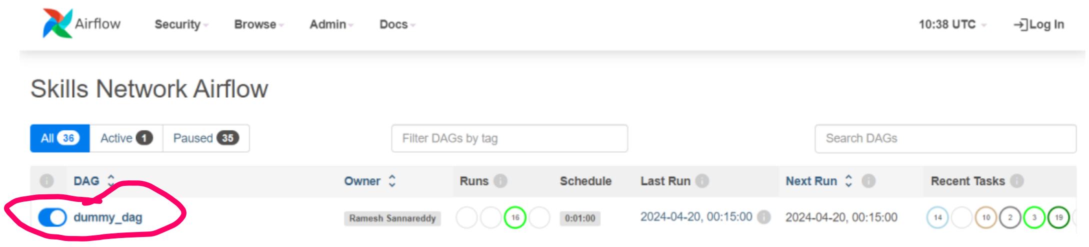
```
- #! /bin/bash
  echo "extract_transform_and_load"

  cut -d ":" -f1,3,6 /etc/passwd  >
  /home/project/airflow/dags/extracted-date.txt
  tr ":" "," < /home/project/airflow/dags/extracted-data.txt >
  /home/project/airflow/dags/transformed-data.csv
```

- Second step is to create a second file called **my_first_dag.py** and paste the code provided below.

```
-   # import the libraries
-
-   from datetime import timedelta
-   # the DAG object; we will need this to instantiate a DAG
-
-   from airflow import DAG
-
-   # operators; we need this to write tasks!
-
-   from airflow.operators.bash_operator import BashOperator
-
-   # this makes sheduleing esay
-
-   from airflow.utlis.dates import days_ago
-
-   # defining DAG arguments
-
-   # You can override them on a per-task basis during operator
    initializations
-
-   default_args = {
-       'owner': 'Ramesh Sannareddy',
-       'start_date': days_ago(0),
-       'email': ['ramesh@somemail.com'],
-       'email_on_failure': False,
-       'email_on_retry': False,
-       'retries': 1,
-       'retry_delay': timedelta(minutes=5),
-   }
-
-   #DAG arguments are like settings for the DAG.
-
-   #The above settings mention
-
-   #the owner name,
-   #when this DAG should run from: days_age(0) means today,
-   #the email address where the alerts are sent to,
-   #whether alert must be sent on failure,
-   #whether alert must be sent on retry,
-   #the number of retries in case of failure, and
-   #the time delay between retries.
-   # defining the DAG
-
-   # define the DAG
-
-   #A typical DAG definition block looks like below.
-   dag = DAG(
-       'my-first-dag',
-       default_args=default_args,
-       description='My first DAG',
-       schedule_interval=timedelta(days=1),
-   )
-   '''
-   Here we are creating a variable named dag by instantiating the
    DAG class with the following parameters.
-
-   sample-etl-dag is the ID of the DAG. This is what you see on
    the web console.
-
-   We are passing the dictionary default_args, in which all the
    defaults are defined.
-
-   description helps us in understanding what this DAG does.
-
-   schedule_interval tells us how frequently this DAG runs. In
    this case every day. (days=1).
-   '''
-


-   # define the task **extract_transform_and_load** to call shell
    script
-   #calling the shell script
-
-   extract_transform_load = BashOperator(
-       task_id="extract_transform_load",
-       bash_command="/home/project/airflow/dags/my_first_dag.sh ",
-       dag=dag,
-   )
-


    '''
    A task is defined using:

    A task_id which is a string and helps in identifying the task.
```

```
What bash command it represents. Here we are calling
the shell script extract_transform_load.shwhich we
previously defined
Which dag this task belongs to.
'''
# task pipeline
extract_transform_load

'''
When we execute the task extract_transform_and_load
the code in the shell script gets executed.
'''
```

- You will something like below like name of your dags, time, and tasks.



Skills Network Airflow

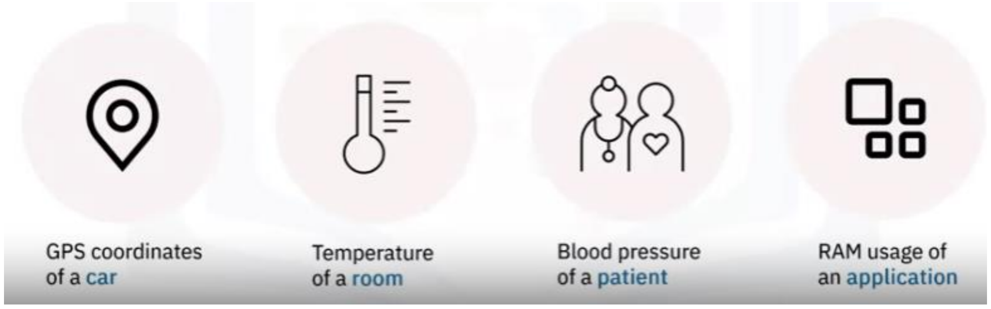
All 36 | Active 1 | Paused 35

Filter DAGs by tag

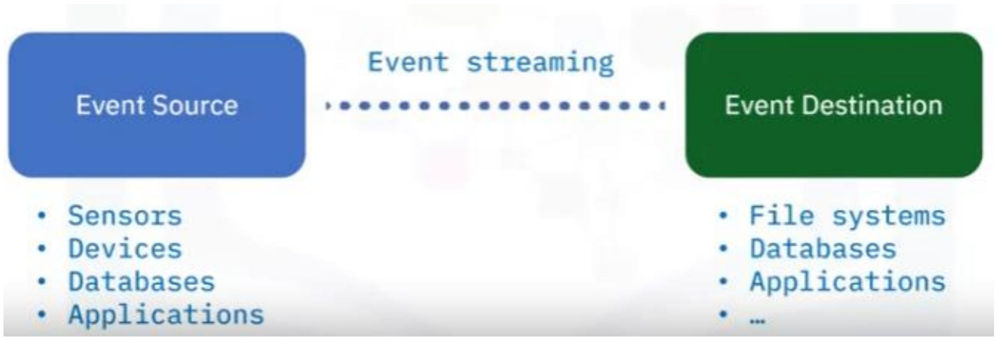
Search DAGs

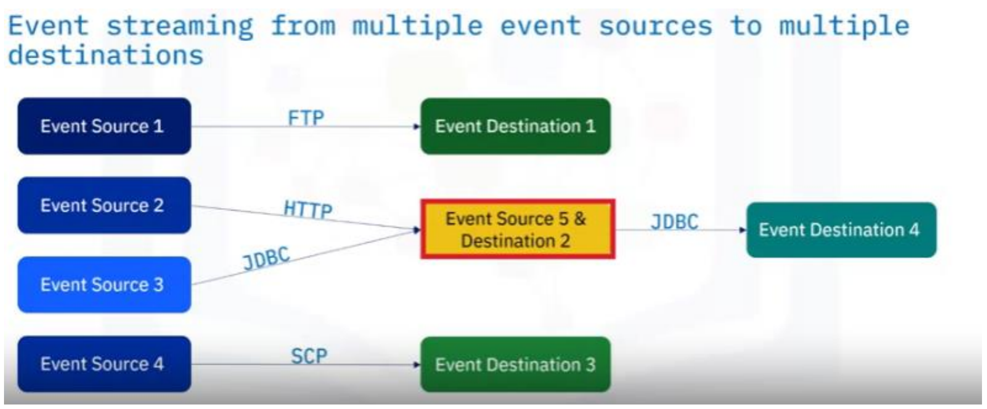
DAG	Owner	Runs	Schedule	Last Run	Next Run	Recent Tasks
dummy_dag	Ramesh Sannareddy	16	0:01:00	2024-04-20, 00:15:00	2024-04-20, 00:15:00	14 10 2 3 19

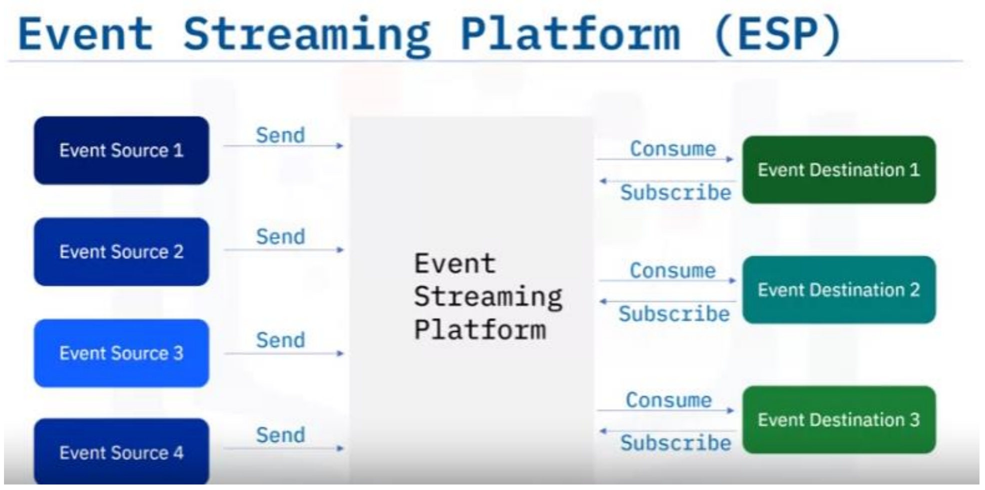
- **Distributed Event Streaming Platform Components:**
    - Event is a type of data which describes an entity observable state updates over time.



- Common formats of events are primitive, a key value pair, a key value pair with time stamp.
- Event streaming means transformations of events from one destination to another.

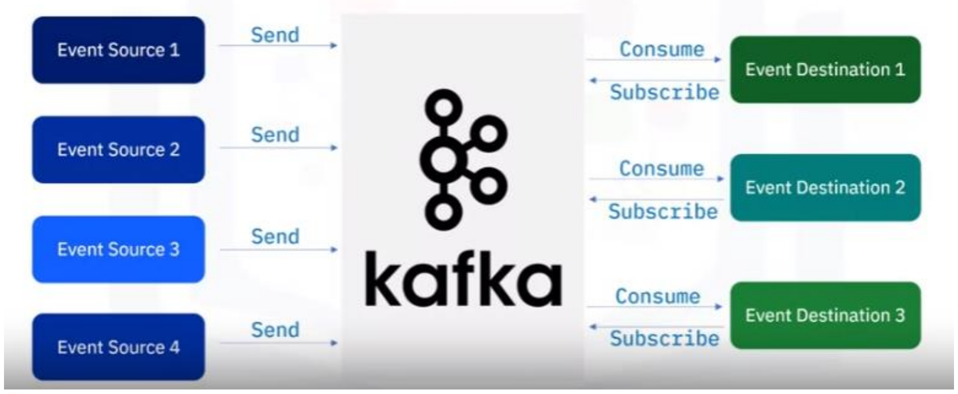




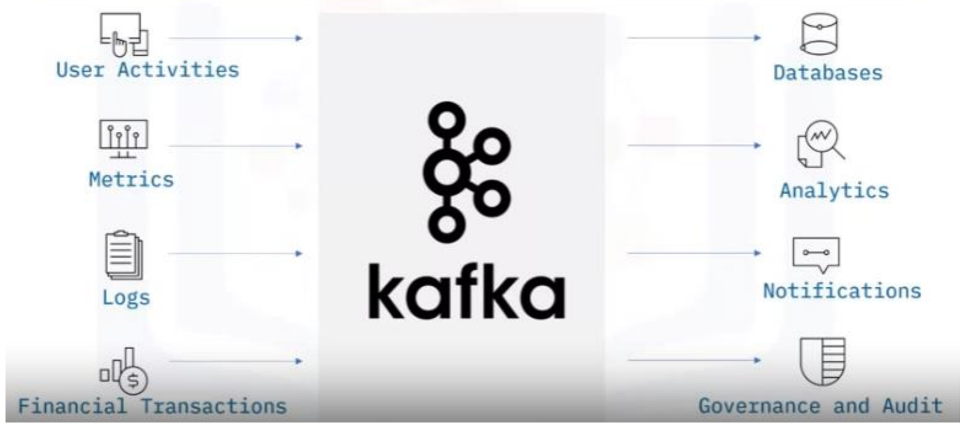


- **Apache Kafka Overview:**
  - Kafka is used to track user activities such as keyboard inputs, mouse, or cursor moments.
  - Used for sensor reading, GPS, hardware, and software monitoring. Monitoring
  - Used to store logs and financial transitions, data storages, analytics, notifications such as emails.

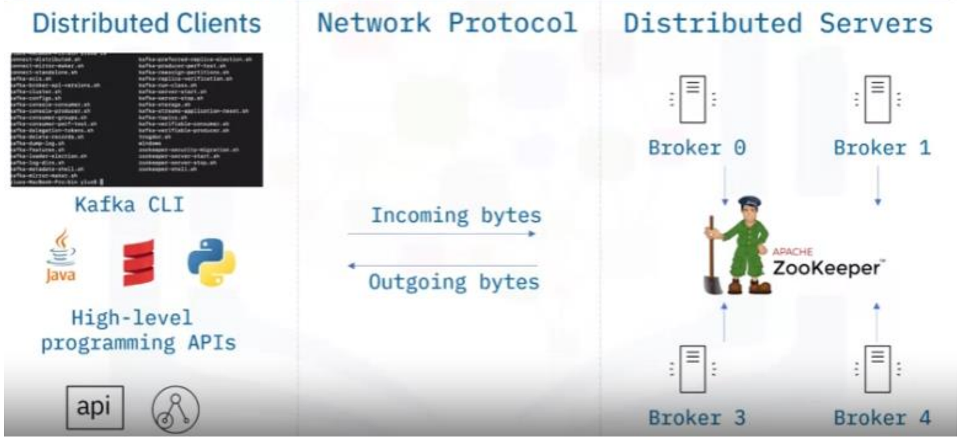
Apache Kafka



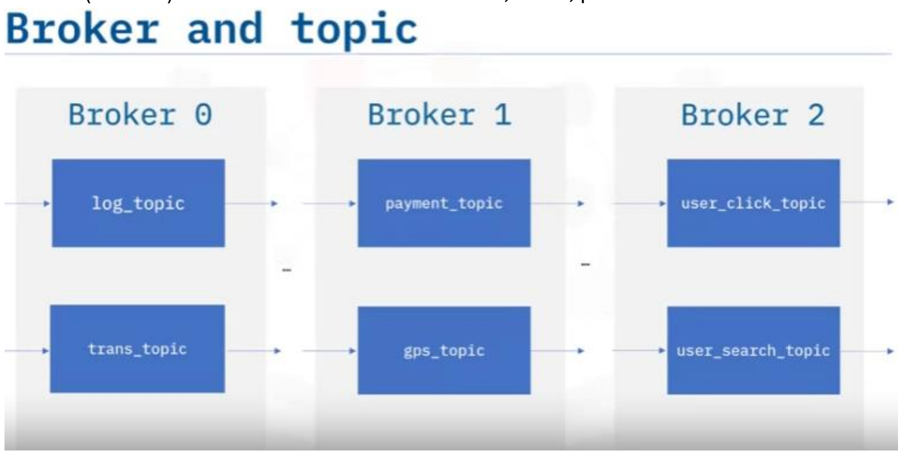
Common use cases



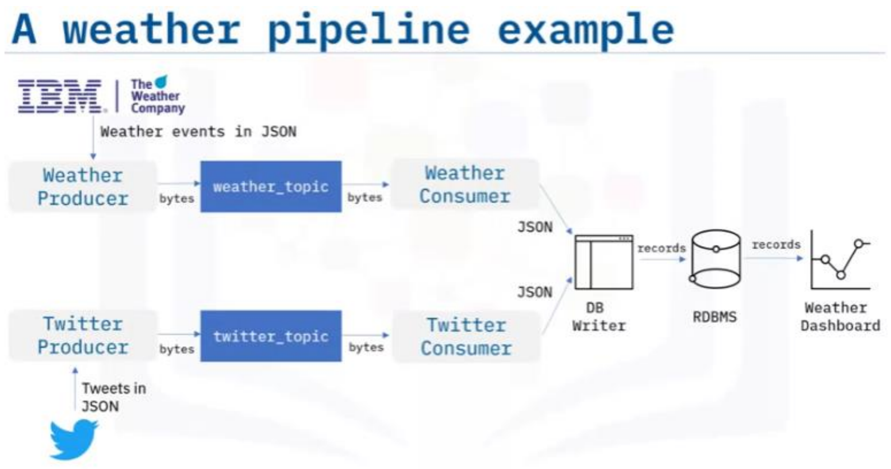
Kafka architecture



- **Building Event Streaming Pipelines using Kafka:**
  - Brokers(servers): the dedicated server to receive, store, process and distribute events.



  - Topics: containers or a database of the systems
  - Replicants: duplicate partitions into different brokers.
  - Partition: divide topics or a database into different brokers or a different server.
  - Producers: Kafka client’s applications to publish events into topic.



A weather pipeline example
IBM | The Weather Company
Weather events in JSON
Weather Producer
bytes
weather_topic
bytes
Weather Consumer
JSON
JSON
DB Writer
records
RDBMS
records
Weather Dashboard
Twitter Producer
bytes
twitter_topic
bytes
Twitter Consumer
Tweets in JSON