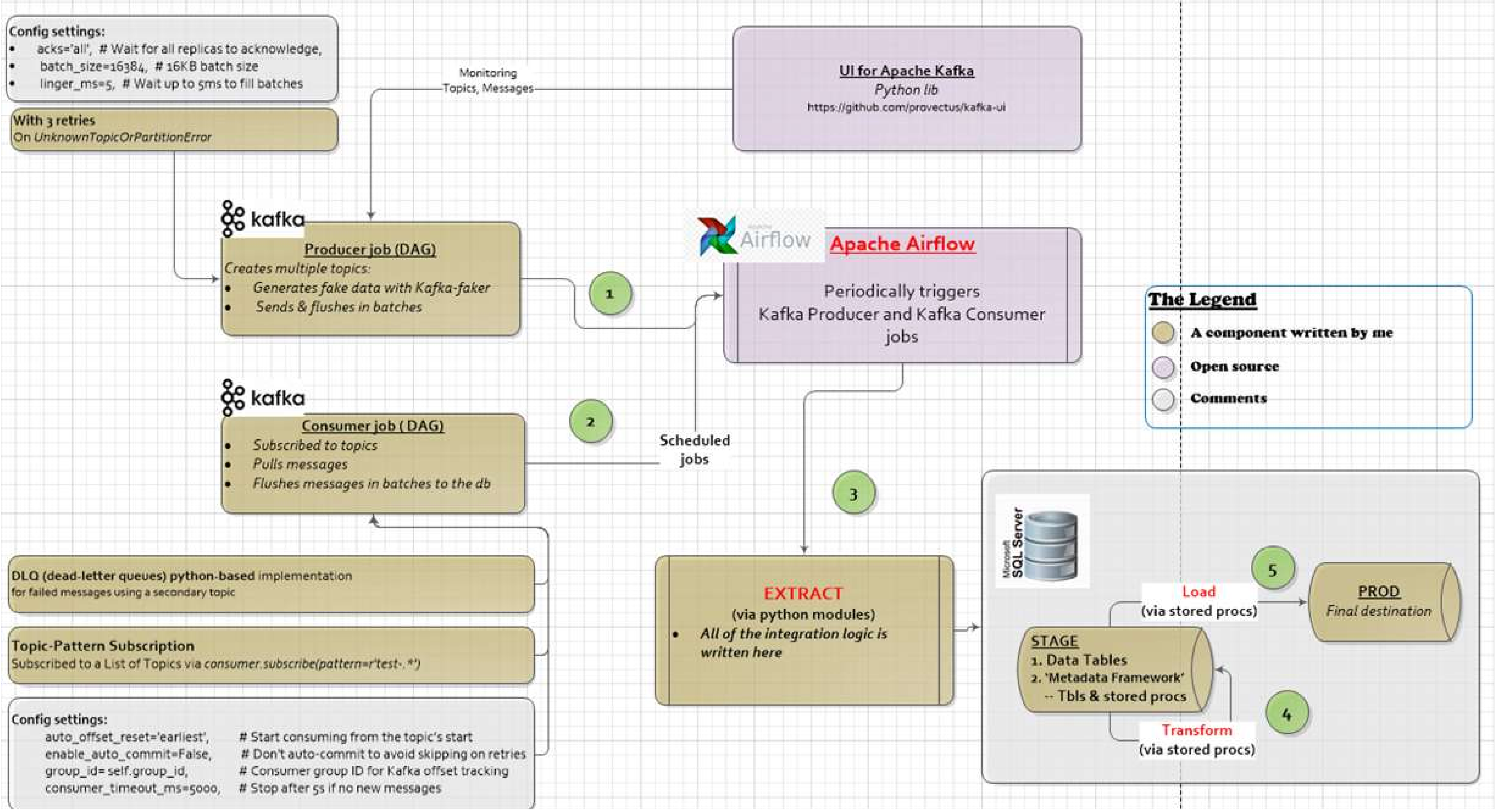
Kafka-python Pipelines with Apache Airflow

(*Proof of concept*) Last Edit 5/22/2025



Config settings:

```
•     acks='all',  # Wait for all replicas to acknowledge,
•      batch_size=16384,  # 16KB batch size
•      linger_ms=5,  # Wait up to 5ms to fill batches
```

With 3 retries
On *UnknownTopicOrPartitionError*

Monitoring
Topics, Messages

UI for Apache Kafka
Python lib
https://github.com/provectus/kafka-ui

kafka

Producer job (DAG)
Creates multiple topics:
- *Generates fake data with Kafka-faker*
- *Sends & flushes in batches*

1

Apache Airflow

Periodically triggers Kafka Producer and Kafka Consumer jobs

kafka

Consumer job (DAG)
- *Subscribed to topics*
- *Pulls messages*
- *Flushes messages in batches to the db*

2

Scheduled jobs

3

The Legend

●	**A component written by me**
●	**Open source**
○	**Comments**

DLQ (dead-letter queues) python-based implementation
for failed messages using a secondary topic

Topic-Pattern Subscription
Subscribed to a List of Topics via *consumer.subscribe(pattern=r'test-.*')*

Config settings:

```
        auto_offset_reset='earliest',       # Start consuming from the topic's start
        enable_auto_commit=False,           # Don't auto-commit to avoid skipping on retries
        group_id= self.group_id,            # Consumer group ID for Kafka offset tracking
        consumer_timeout_ms=5000,           # Stop after 5s if no new messages
```

EXTRACT
(via python modules)
- *All of the integration logic is written here*

5

Load
(via stored procs)

PROD
Final destination

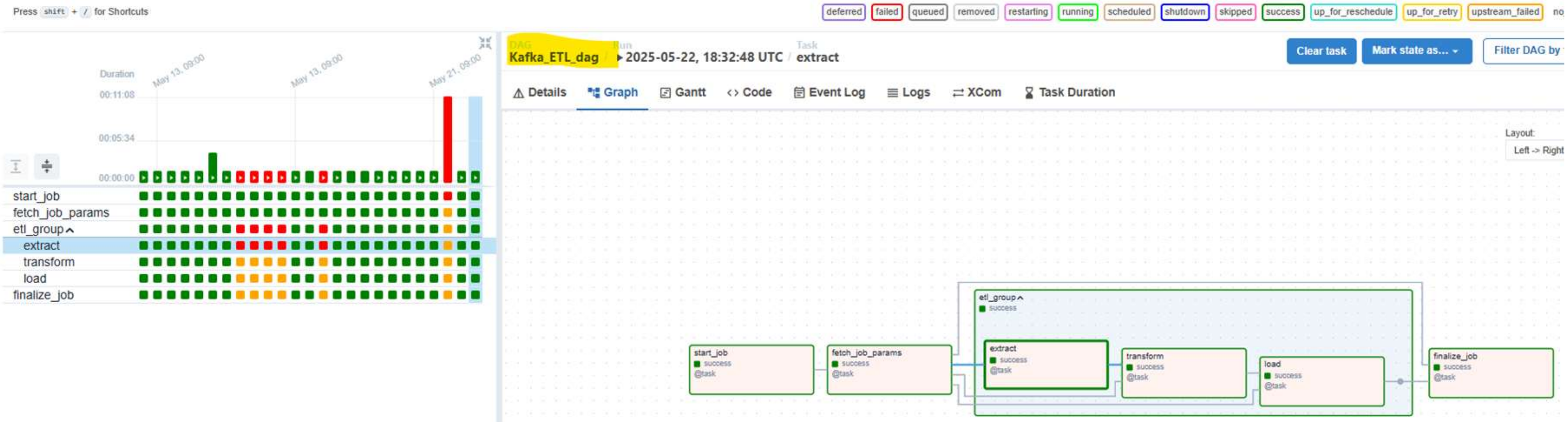
STAGE
1. Data Tables
2. 'Metadata Framework'
-- Tbls & stored procs

Transform
(via stored procs)

4

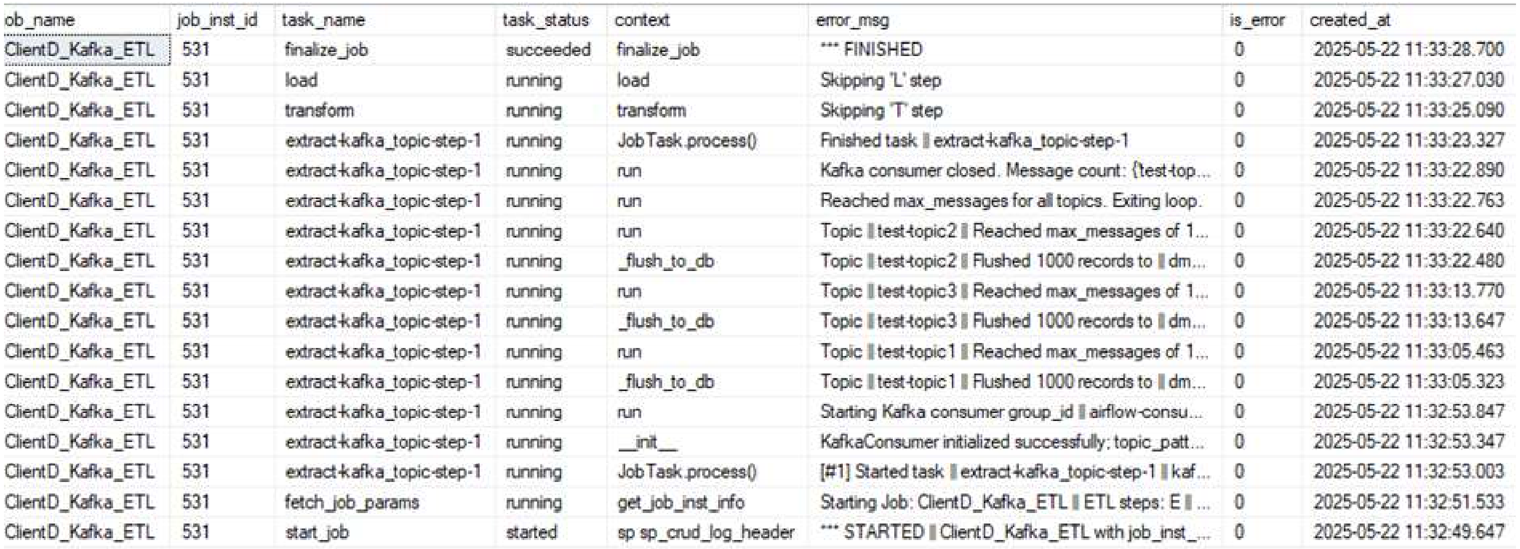
Kafka
Consumer

1. Kafka Consumer DAG



DAG Kafka_ETL_dag / Run ▸ 2025-05-22, 18:32:48 UTC / Task extract

2. Database log for Kafka Consumerr DAG

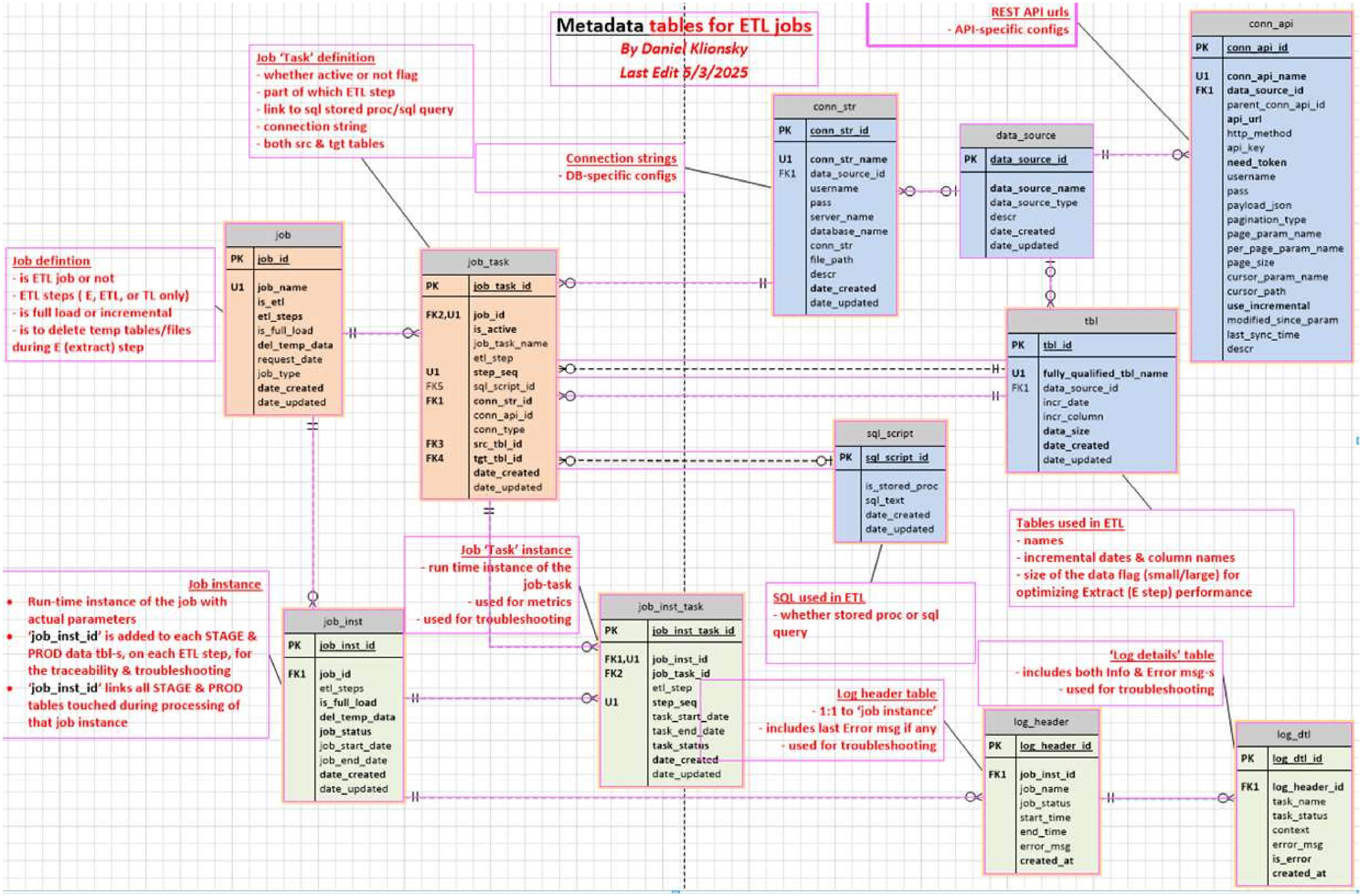


ob_name	job_inst_id	task_name	task_status	context	error_msg	is_error	created_at
ClientD_Kafka_ETL	531	finalize_job	succeeded	finalize_job	*** FINISHED	0	2025-05-22 11:33:28.700
ClientD_Kafka_ETL	531	load	running	load	Skipping 'L' step	0	2025-05-22 11:33:27.030
ClientD_Kafka_ETL	531	transform	running	transform	Skipping 'T' step	0	2025-05-22 11:33:25.090
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	JobTask.process()	Finished task ▮ extract-kafka_topic-step-1	0	2025-05-22 11:33:23.327
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	run	Kafka consumer closed. Message count: {'test-top...	0	2025-05-22 11:33:22.890
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	run	Reached max_messages for all topics. Exiting loop.	0	2025-05-22 11:33:22.763
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	run	Topic ▮ test-topic2 ▮ Reached max_messages of 1...	0	2025-05-22 11:33:22.640
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	_flush_to_db	Topic ▮ test-topic2 ▮ Flushed 1000 records to ▮ dm...	0	2025-05-22 11:33:22.480
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	run	Topic ▮ test-topic3 ▮ Reached max_messages of 1...	0	2025-05-22 11:33:13.770
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	_flush_to_db	Topic ▮ test-topic3 ▮ Flushed 1000 records to ▮ dm...	0	2025-05-22 11:33:13.647
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	run	Topic ▮ test-topic1 ▮ Reached max_messages of 1...	0	2025-05-22 11:33:05.463
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	_flush_to_db	Topic ▮ test-topic1 ▮ Flushed 1000 records to ▮ dm...	0	2025-05-22 11:33:05.323
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	run	Starting Kafka consumer group_id ▮ airflow-consu...	0	2025-05-22 11:32:53.847
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	__init__	KafkaConsumer initialized successfully; topic_patt...	0	2025-05-22 11:32:53.347
ClientD_Kafka_ETL	531	extract-kafka_topic-step-1	running	JobTask.process()	[#1] Started task ▮ extract-kafka_topic-step-1 ▮ kaf...	0	2025-05-22 11:32:53.003
ClientD_Kafka_ETL	531	fetch_job_params	running	get_job_inst_info	Starting Job: ClientD_Kafka_ETL ▮ ETL steps: E ▮ ...	0	2025-05-22 11:32:51.533
ClientD_Kafka_ETL	531	start_job	started	sp sp_crud_log_header	*** STARTED ▮ ClientD_Kafka_ETL with job_inst_...	0	2025-05-22 11:32:49.647

Metadata tables for ETL jobs

By Daniel Klionsky

Last Edit 5/3/2025



REST API urls
- API-specific configs

Job 'Task' definition
- whether active or not flag
- part of which ETL step
- link to sql stored proc/sql query
- connection string
- both src & tgt tables

Connection strings
- DB-specific configs

Job defintion
- is ETL job or not
- ETL steps (E, ETL, or TL only)
- is full load or incremental
- is to delete temp tables/files during E (extract) step

conn_api

PK	conn_api_id
U1	conn_api_name
FK1	data_source_id
	parent_conn_api_id
	api_url
	http_method
	api_key
	need_token
	username
	pass
	payload_json
	pagination_type
	page_param_name
	per_page_param_name
	page_size
	cursor_param_name
	cursor_path
	use_incremental
	modified_since_param
	last_sync_time
	descr

conn_str

PK	conn_str_id
U1	conn_str_name
FK1	data_source_id
	username
	pass
	server_name
	database_name
	conn_str
	file_path
	descr
	date_created
	date_updated

data_source

PK	data_source_id
	data_source_name
	data_source_type
	descr
	date_created
	date_updated

job

PK	job_id
U1	job_name
	is_etl
	etl_steps
	is_full_load
	del_temp_data
	request_date
	job_type
	date_created
	date_updated

job_task

PK	job_task_id
FK2,U1	job_id
	is_active
	job_task_name
	etl_step
U1	step_seq
FK5	sql_script_id
FK1	conn_str_id
	conn_api_id
	conn_type
FK3	src_tbl_id
FK4	tgt_tbl_id
	date_created
	date_updated

tbl

PK	tbl_id
U1	fully_qualified_tbl_name
FK1	data_source_id
	incr_date
	incr_column
	data_size
	date_created
	date_updated

sql_script

PK	sql_script_id
	is_stored_proc
	sql_text
	date_created
	date_updated

Tables used in ETL
- names
- incremental dates & column names
- size of the data flag (small/large) for optimizing Extract (E step) performance

Job 'Task' instance
- run time instance of the job-task
- used for metrics
- used for troubleshooting

Job instance
- Run-time instance of the job with actual parameters
- 'job_inst_id' is added to each STAGE & PROD data tbl-s, on each ETL step, for the traceability & troubleshooting
- 'job_inst_id' links all STAGE & PROD tables touched during processing of that job instance

SQL used in ETL
- whether stored proc or sql query

job_inst

PK	job_inst_id
FK1	job_id
	etl_steps
	is_full_load
	del_temp_data
	job_status
	job_start_date
	job_end_date
	date_created
	date_updated

job_inst_task

PK	job_inst_task_id
FK1,U1	job_inst_id
FK2	job_task_id
	etl_step
U1	step_seq
	task_start_date
	task_end_date
	task_status
	date_created
	date_updated

'Log details' table
- includes both Info & Error msg-s
- used for troubleshooting

Log header table
- 1:1 to 'job instance'
- includes last Error msg if any
- used for troubleshooting

log_header

PK	log_header_id
FK1	job_inst_id
	job_name
	job_status
	start_time
	end_time
	error_msg
	created_at

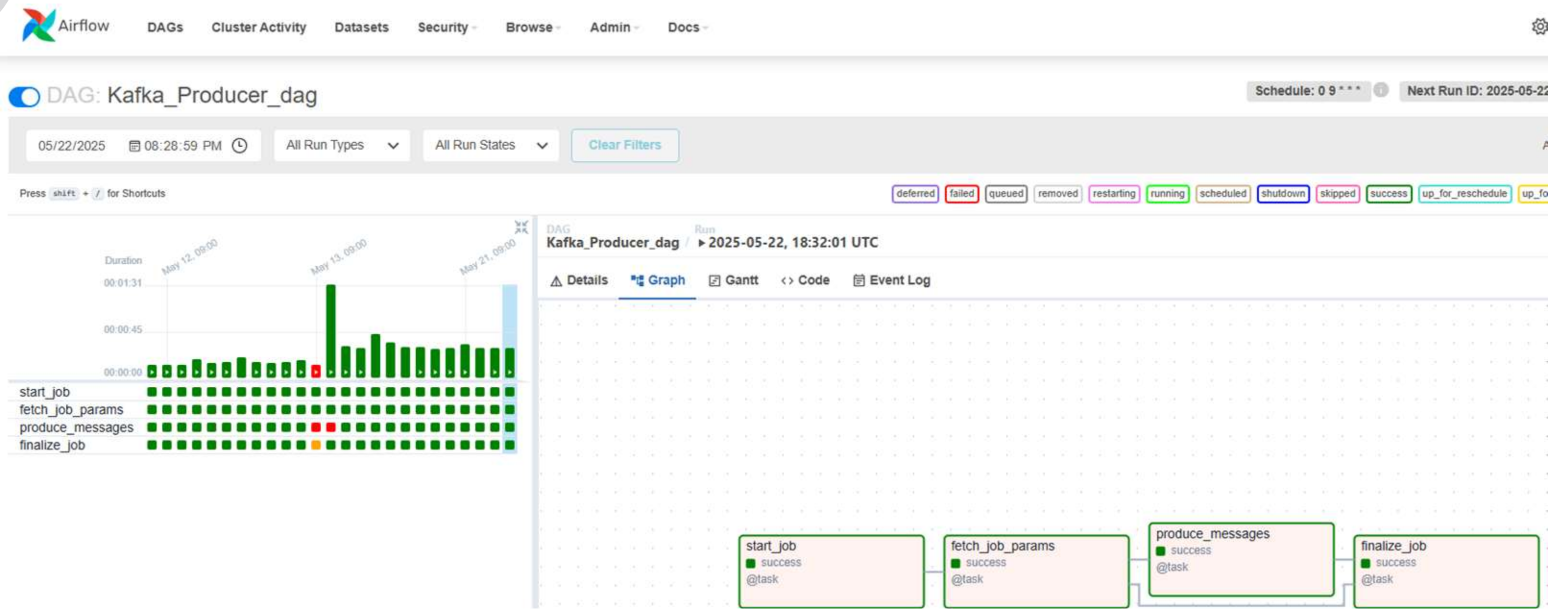
log_dtl

PK	log_dtl_id
FK1	log_header_id
	task_name
	task_status
	context
	error_msg
	is_error
	created_at

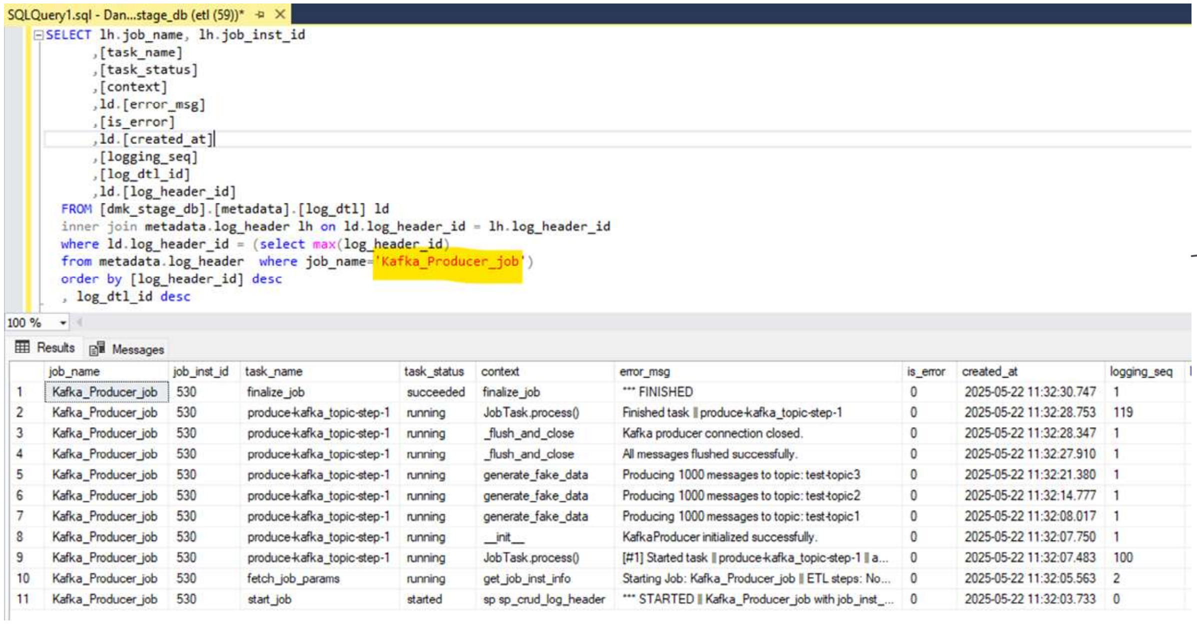
Kafka Producer

1. AirFlow Graph for Kafka Producer DAG

to generate sample messages



2. Database log for Kafka Producer DAG



```sql
SELECT lh.job_name, lh.job_inst_id
      ,[task_name]
      ,[task_status]
      ,[context]
      ,ld.[error_msg]
      ,[is_error]
      ,ld.[created_at]
      ,[logging_seq]
      ,[log_dtl_id]
      ,ld.[log_header_id]
  FROM [dmk_stage_db].[metadata].[log_dtl] ld
  inner join metadata.log_header lh on ld.log_header_id = lh.log_header_id
  where ld.log_header_id = (select max(log_header_id)
  from metadata.log_header  where job_name='Kafka_Producer_job')
  order by [log_header_id] desc
  , log_dtl_id desc
```

	job_name	job_inst_id	task_name	task_status	context	error_msg	is_error	created_at	logging_seq
1	Kafka_Producer_job	530	finalize_job	succeeded	finalize_job	*** FINISHED	0	2025-05-22 11:32:30.747	1
2	Kafka_Producer_job	530	produce-kafka_topic-step-1	running	JobTask.process()	Finished task ‖ produce-kafka_topic-step-1	0	2025-05-22 11:32:28.753	119
3	Kafka_Producer_job	530	produce-kafka_topic-step-1	running	_flush_and_close	Kafka producer connection closed.	0	2025-05-22 11:32:28.347	1
4	Kafka_Producer_job	530	produce-kafka_topic-step-1	running	_flush_and_close	All messages flushed successfully.	0	2025-05-22 11:32:27.910	1
5	Kafka_Producer_job	530	produce-kafka_topic-step-1	running	generate_fake_data	Producing 1000 messages to topic: test-topic3	0	2025-05-22 11:32:21.380	1
6	Kafka_Producer_job	530	produce-kafka_topic-step-1	running	generate_fake_data	Producing 1000 messages to topic: test-topic2	0	2025-05-22 11:32:14.777	1
7	Kafka_Producer_job	530	produce-kafka_topic-step-1	running	generate_fake_data	Producing 1000 messages to topic: test-topic1	0	2025-05-22 11:32:08.017	1
8	Kafka_Producer_job	530	produce-kafka_topic-step-1	running	__init__	KafkaProducer initialized successfully.	0	2025-05-22 11:32:07.750	1
9	Kafka_Producer_job	530	produce-kafka_topic-step-1	running	JobTask.process()	[#1] Started task ‖ produce-kafka_topic-step-1 ‖ a...	0	2025-05-22 11:32:07.483	100
10	Kafka_Producer_job	530	fetch_job_params	running	get_job_inst_info	Starting Job: Kafka_Producer_job ‖ ETL steps: No...	0	2025-05-22 11:32:05.563	2
11	Kafka_Producer_job	530	start_job	started	sp sp_crud_log_header	*** STARTED ‖ Kafka_Producer_job with job_inst_...	0	2025-05-22 11:32:03.733	0

Processing steps are logged here, in addition to AirFlow logging