Kafka Topics Partition Count Recommender [MULTITHREADED] Tool

> **TL;DR:** ***End Kafka performance headaches.*** *This intelligent recommender dynamically evaluates your Kafka environment—leveraging either deep historical consumption patterns or aggregated insights from the Confluent Metrics API—to deliver precise, data-driven partition recommendations. By detecting whether topics' partitions are under- or over-provisioned, it empowers you to optimize resources, eliminate inefficiencies, and ensure your workloads scale seamlessly with demand.*

The **Kafka Cluster Topics Partition Count Recommender [MULTITHREADED] Tool** offers data-driven accuracy for Kafka topic sizing. By analyzing past consumption trends, that is, the average consumption records in bytes, it uses this information to determine consumer throughput. Then, over a rolling **n-day** period, it identifies the average consumption of records in bytes, scaling that number by **n-factor** to forecast future demand and calculate the required throughput. Next, it divides the required throughput by the consumer throughput and rounds the result to the nearest whole number to determine the optimal number of partitions. The result is an intelligent, automated recommendation system that ensures each Kafka topic has the appropriate number of partitions to handle current workload and support future growth effectively.

Table of Contents

1.0 To get started

Download ---> ***Configure*** ---> ***Run*** ---> ***Results***

1.1 Download the Tool

Clone the repo: `shell git clone https://github.com/j3-signalroom/kafka_cluster-topics-partition_count_recommender-tool.git`

Since this project was built using `uv`, please install it, and then run the following command to install all the project dependencies:

```
uv sync
```

1.1.1 Special Note on two custom dependencies

This project has *two custom dependencies* that we want to bring to your attention:

1. `cc-clients-python_lib`: *This library offers a simple way to interact with Confluent Cloud services, including the Metrics API. It makes it easier to send API requests and manage responses. It is used in this project to connect to the Confluent Cloud Metrics API and retrieve topic consumption metrics.*
2. `aws-clients-python_lib`: *This library is used to interact with AWS services, specifically AWS Secrets Manager in this case. It enables the tool to securely retrieve secrets stored in AWS Secrets Manager.*

1.2 Configure the Tool

Now, you need to set up the tool by creating a `.env` file in the root directory of your project. This file will store all the essential environment variables required for the tool to connect to your Confluent Cloud Kafka cluster and function correctly. Additionally, you can choose to use **AWS Secrets Manager** to manage your secrets.

1.2.1 Create a Dedicated Service Account for the Recommender Tool

The service account needs to have OrganizationAdmin, EnvironmentAdmin or CloudClusterAdmin role to provision Kafka cluster API keys and the MetricsViewer role to access the Metrics API for all clusters it has access to.

1. Use the Confluent CLI (Command-Line Interface) to create the service account:

   > **Note:** If you haven't already, install the Confluent CLI and log in to your Confluent Cloud account using `confluent login`. Moreover, the account you use to log in must have the OrganizationAdmin role to create the **Cloud API key in Step 5**.

   ```
   confluent iam service-account create <SERVICE_ACCOUNT_NAME> --description "<DESCRIPTION>"
   ```

   For instance, you run `confluent iam service-account create recommender-service-account --description "Service account for Recommender Tool"`, the output should resemble:

   ```
   +-------------+---------------------------------+
   | ID          | sa-abcd123                      |
   | Name        | recommender-service-account     |
   | Description | Service account for             |
   |             | Recommender Tool                |
   +-------------+---------------------------------+
   ```

2. Make note of the service account ID in the output, which is in the form `sa-xxxxxxx`, which you will assign the OrganizationAdmin, EnvironmentAdmin or CloudClusterAdmin role, and MetricsViewer role to in the next steps, and assign it to the `PRINCIPAL_ID` environment variable in the `.env` file.

3. Decide at what level you want to assign the OrganizationAdmin, EnvironmentAdmin or CloudClusterAdmin role to the service account. The recommended approach is to assign the role at the organization level so that the service account can provision API keys for any Kafka cluster in the organization. If you want to restrict the service account to only be able to provision API keys for Kafka clusters in a specific environment, then assign the EnvironmentAdmin role at the environment level. If you want to restrict the service account to only be able to provision API keys for a specific Kafka cluster, then assign the CloudClusterAdmin role at the cluster level.

   For example, to assign the EnvironmentAdmin role at the environment level:

   ```
   confluent iam rbac role-binding create --role EnvironmentAdmin --principal User:<SERVICE_ACCOUNT_ID> --
   environment <ENVIRONMENT_ID>
   ```

   Or, to assign the CloudClusterAdmin role at the cluster level:

   ```
   confluent iam rbac role-binding create --role CloudClusterAdmin --principal User:<SERVICE_ACCOUNT_ID> --cluster
   <KAFKA_CLUSTER_ID>
   ```

   For instance, you run `confluent iam rbac role-binding create --role EnvironmentAdmin --principal User:sa-abcd123 --environment env-123abc`, the output should resemble:

   ```
   +-----------+-----------------+
   | ID        | rb-j3XQ8Y       |
   | Principal | User:sa-abcd123 |
   | Role      | EnvironmentAdmin |
   +-----------+-----------------+
   ```

4. Assign the MetricsViewer role to the service account at the organization, environment, or cluster level, For example to assign the MetricsViewer role at the environment level:

   ```
   confluent iam rbac role-binding create --role MetricsViewer --principal User:<SERVICE_ACCOUNT_ID> --environment
   <ENVIRONMENT_ID>
   ```

For instance, you run `confluent iam rbac role-binding create --role MetricsViewer --principal User:sa-abcd123 --environment env-123abc`, the output should resemble:

```
+-----------+------------------+
| ID        | rb-1GgVMN        |
| Principal | User:sa-abcd123  |
| Role      | MetricsViewer    |
+-----------+------------------+
```

5. Create an API key for the service account:

```
confluent api-key create --resource cloud --service-account <SERVICE_ACCOUNT_ID> --description "<DESCRIPTION>"
```

For instance, you run `confluent api-key create --resource cloud --service-account sa-abcd123 --description "API Key for Recommender Tool"`, the output should resemble:

```
+------------+------------------------------------------------------------------+
| API Key    | 1WORLDABCDEF7OAB                                                 |
| API Secret | cfltabCdeFg1hI+/2j34KLMnoprSTuvxy/Za+b5/6bcDe/7fGhIjklMnOPQ8rT9U |
+------------+------------------------------------------------------------------+
```

6. Make note of the API key and secret in the output, which you will assign to the `confluent_cloud_api_key` and `confluent_cloud_api_secret` environment variables in the `.env` file. Alternatively, you can securely store and retrieve these credentials using AWS Secrets Manager.

1.2.2 Create the `.env` file

Create the `.env` file and add the following environment variables, filling them with your Confluent Cloud credentials and other required values:

```
# Set the flag to `True` to use the Confluent Cloud API key for fetching Kafka
# credentials; otherwise, set it to `False` to reference `KAFKA_CREDENTIALS` or
# `KAFKA_API_SECRET_PATHS` to obtain the Kafka Cluster credentials
USE_CONFLUENT_CLOUD_API_KEY_TO_FETCH_KAFKA_CREDENTIALS=<True|False>

# Environment and Kafka cluster filters (comma-separated IDs)
# Example: ENVIRONMENT_FILTER="env-123,env-456"
# Example: KAFKA_CLUSTER_FILTER="lkc-123,lkc-456"
ENVIRONMENT_FILTER=<YOUR_ENVIRONMENT_FILTER, IF ANY>
KAFKA_CLUSTER_FILTER=<YOUR_KAFKA_CLUSTER_FILTER, IF ANY>

# Environment variables credentials for Confluent Cloud and Kafka clusters
CONFLUENT_CLOUD_CREDENTIAL={"confluent_cloud_api_key":"<YOUR_CONFLUENT_CLOUD_API_KEY>", "confluent_cloud_api_secret":
"<YOUR_CONFLUENT_CLOUD_API_SECRETS>"}
KAFKA_CREDENTIALS=[{"kafka_cluster_id": "<YOUR_KAFKA_CLUSTER_ID>", "bootstrap.servers": "
<YOUR_BOOTSTRAP_SERVER_URI>", "sasl.username": "<YOUR_KAFKA_API_KEY>", "sasl.password": "<YOUR_KAFKA_API_SECRET>"}]

# Confluent Cloud principal ID (user or service account) for API key creation
# Example: PRINCIPAL_ID=u-abc123 or PRINCIPAL_ID=sa-xyz789
PRINCIPAL_ID=<YOUR_PRINCIPAL_ID>

# AWS Secrets Manager Secrets for Confluent Cloud and Kafka clusters
USE_AWS_SECRETS_MANAGER=<True|False>
CONFLUENT_CLOUD_API_SECRET_PATH={"region_name": "<YOUR_SECRET_AWS_REGION_NAME>", "secret_name": "
<YOUR_CONFLUENT_CLOUD_API_KEY_AWS_SECRETS>"}
KAFKA_API_SECRET_PATHS=[{"region_name": "<YOUR_SECRET_AWS_REGION_NAME>", "secret_name": "
<YOUR_KAFKA_API_KEY_AWS_SECRETS>"}]

# Topic analysis configuration
INCLUDE_INTERNAL_TOPICS=<True|False>
TOPIC_FILTER=<YOUR_TOPIC_FILTER, IF ANY>

# Minimum recommended partitions
MIN_RECOMMENDED_PARTITIONS=<YOUR_MIN_RECOMMENDED_PARTITIONS>

# Throughput and partition calculation configuration
REQUIRED_CONSUMPTION_THROUGHPUT_FACTOR=<YOUR_REQUIRED_CONSUMPTION_THROUGHPUT_FACTOR>

# Sampling configuration
USE_SAMPLE_RECORDS=<True|False>
SAMPLING_DAYS=<YOUR_SAMPLING_DAYS>
SAMPLING_BATCH_SIZE=<YOUR_SAMPLING_BATCH_SIZE>
SAMPLING_MAX_CONSECUTIVE_NULLS=<YOUR_SAMPLING_MAX_CONSECUTIVE_NULLS>
```

```
SAMPLING_TIMEOUT_SECONDS=<YOUR_SAMPLING_TIMEOUT_SECONDS>
SAMPLING_MAX_CONTINUOUS_FAILED_BATCHES=<YOUR_SAMPLING_MAX_CONTINUOUS_FAILED_BATCHES>

# Multithreading configuration
MAX_CLUSTER_WORKERS=<YOUR_MAX_CLUSTER_WORKERS>
MAX_WORKERS_PER_CLUSTER=<YOUR_MAX_WORKERS_PER_CLUSTER>

# Test environment variables
TEST_ENVIRONMENT_ID=<YOUR_TEST_ENVIRONMENT_ID>
TEST_KAFKA_TOPIC_NAME=<YOUR_TEST_KAFKA_TOPIC_NAME>
TEST_KAFKA_CLUSTER_ID=<YOUR_TEST_KAFKA_CLUSTER_ID>

# Kafka writer configuration
USE_KAFKA_WRITER=<True|False>
KAFKA_WRITER_TOPIC_NAME=<YOUR_KAFKA_WRITER_TOPIC_NAME>
KAFKA_WRITER_TOPIC_PARTITION_COUNT=<YOUR_KAFKA_WRITER_TOPIC_PARTITION_COUNT>
KAFKA_WRITER_TOPIC_REPLICATION_FACTOR=<YOUR_KAFKA_WRITER_TOPIC_REPLICATION_FACTOR>
KAFKA_WRITER_TOPIC_DATA_RETENTION_IN_DAYS=<YOUR_KAFKA_WRITER_TOPIC_DATA_RETENTION_IN_DAYS>
```

The environment variables are defined as follows:

Environment Variable Name	Type	Description	Example
`USE_CONFLUENT_CLOUD_API_KEY_TO_FETCH_KAFKA_CREDENTIALS`	Boolean	Set the flag to `True` to use the Confluent Cloud API key for fetching Kafka credentials; otherwise, set it to `False` to reference `KAFKA_CREDENTIALS` or `KAFKA_API_SECRET_PATHS` to obtain the Kafka Cluster credentials.	`True` or `False`
`ENVIRONMENT_FILTER`	Comma-separated String	A list of specific Confluent Cloud environment IDs to filter. When provided, only these environments will be used to fetch Kafka cluster credentials. Use commas to separate multiple environment IDs. Leave blank or unset to use all available environments.	`env-123,env-456`
`PRINCIPAL_ID`	String	Confluent Cloud principal ID (user or service account) for API key creation.	`u-abc123` or `sa-xyz789`
`KAFKA_CLUSTER_FILTER`	Comma-separated String	A list of specific Kafka cluster IDs to filter. When provided, only these Kafka clusters will be analyzed. Use commas to separate multiple cluster IDs. Leave blank or unset to analyze all available clusters.	`lkc-123,lkc-456`
`CONFLUENT_CLOUD_CREDENTIAL`	JSON Object	Contains authentication credentials for Confluent Cloud API access. Must include `confluent_cloud_api_key` and `confluent_cloud_api_secret` fields for authenticating with Confluent Cloud services.	`{"confluent_cloud_api_key": "CKABCD123456", "confluent_cloud_api_secret" "xyz789secretkey"}`
`KAFKA_CREDENTIALS`	JSON Array	Array of Kafka cluster connection objects. Each object must contain `sasl.username`, `sasl.password`, `kafka_cluster_id`, and `bootstrap.servers` for connecting to specific Kafka clusters.	`[{"sasl.username": "ABC123", "sasl.password": "secret123" "kafka_cluster_id": "lkc-abc123", "bootstrap.servers" "pkc-123.us-east-1.aws.confluent.cloud:9092"}`

Environment Variable Name	Type	Description	Example
`USE_AWS_SECRETS_MANAGER`	Boolean	Controls whether to retrieve credentials from AWS Secrets Manager instead of using direct environment variables. When `True`, credentials are fetched from AWS Secrets Manager using the paths specified in other variables.	`True` or `False`
`CONFLUENT_CLOUD_API_SECRET_PATH`	JSON Object	AWS Secrets Manager configuration for Confluent Cloud credentials. Contains `region_name` (AWS region) and `secret_name` (name of the secret in AWS Secrets Manager). Only used when `USE_AWS_SECRETS_MANAGER` is `True`.	`{"region_name": "us-east-1", "secret_name": "confluent-cloud-api-credentials"}`
`KAFKA_API_SECRET_PATHS`	JSON Array	Array of AWS Secrets Manager configurations for Kafka cluster credentials. Each object contains `region_name` and `secret_name` for retrieving cluster-specific credentials from AWS Secrets Manager.	`[{"region_name": "us-east-1" "secret_name": "kafka-cluster-1-creds"}, {"region_name": "us-east-1", "secret_name": "kafka-cluster-2-creds"}]`
`INCLUDE_INTERNAL_TOPICS`	Boolean	Determines whether Kafka internal topics (system topics like `__consumer_offsets`, `_schemas`) are included in the analysis and reporting. Set to `False` to exclude internal topics and focus only on user-created topics.	`True` or `False`
`TOPIC_FILTER`	Comma-separated String	A list of specific topic names or part of topic names to analyze. When provided, only these topics will be included in the analysis. Use commas to separate multiple topic names. Leave blank or unset to analyze all available topics.	`user-events,order-processing,payment-notifications`
`MIN_RECOMMENDED_PARTITIONS`	Integer	The minimum number of partitions to recommend for any topic, regardless of calculated needs. This ensures that topics have a baseline level of parallelism and fault tolerance.	`6`, `12`
`MIN_CONSUMPTION_THROUGHPUT`	Integer	The minimum required consumption throughput for any topic, regardless of calculated needs. This ensures that topics have a baseline level of performance.	`10485760` (10 MB/s)
`REQUIRED_CONSUMPTION_THROUGHPUT_FACTOR`	Float/Integer	Multiplier applied to current peak consumption rates for capacity planning and future demand forecasting. A value of `3` means planning for 3x the current peak throughput (300% of current load).	`3` (for 300%), `2.5` (for 250%)
`USE_SAMPLE_RECORDS`	Boolean	Enables record sampling mode for analysis instead of processing all records. When `True`, only a subset of records is analyzed for performance optimization. Recommended for large topics or initial analysis.	`True` or `False`

Environment Variable Name	Type	Description	Example
`SAMPLING_BATCH_SIZE`	Integer	Maximum number of records to sample per topic when `USE_SAMPLE_RECORDS` is `True`. Controls the sample size for analysis to balance accuracy with performance. Larger values provide more accurate analysis but slower processing.	`1000`, `10000`
`SAMPLING_MAX_CONSECUTIVE_NULLS`	Integer	Maximum number of consecutive null records encountered during sampling before stopping the sampling process for a topic. Helps to avoid excessive polling when there are no new records.	`10`, `50`
`SAMPLING_TIMEOUT_SECONDS`	Float	Maximum time (in seconds) to wait for records during sampling before stopping the sampling process for a topic. Prevents long waits when there are no new records.	`1.0`, `2.5`
`SAMPLING_MAX_CONTINUOUS_FAILED_BATCHES`	Integer	Maximum number of continuous failed batches encountered during sampling before stopping the sampling process for a topic. Helps to avoid excessive retries when there are persistent issues.	`3`, `5`
`SAMPLING_DAYS`	Integer	Time window (in days) for record sampling, creating a rolling window that looks back from the current time. Defines how far back to sample records for analysis. **Note**: Topics with retention periods shorter than this value will use their maximum available retention period instead.	`7` (last week), `30` (last month)
`MAX_CLUSTER_WORKERS`	Integer	Maximum number of concurrent worker threads to analyze multiple Kafka clusters in parallel. Helps to speed up analysis when working with multiple clusters.	`2`, `4`
`MAX_WORKERS_PER_CLUSTER`	Integer	Maximum number of concurrent worker threads to analyze multiple topics within a single Kafka cluster in parallel. Helps to speed up analysis for clusters with many topics.	`4`, `8`
`TEST_ENVIRONMENT_ID`	String	Confluent Cloud environment ID used for testing connectivity.	`env-abc123`
`TEST_KAFKA_TOPIC_NAME`	String	Kafka topic name used for testing connectivity.	`test-topic`
`TEST_KAFKA_CLUSTER_ID`	String	Kafka cluster ID used for testing connectivity.	`lkc-abc123`
`USE_KAFKA_WRITER`	Boolean	Enables the Kafka writer functionality to create a test topic and produce messages to it for connectivity testing. When `True`, the tool will attempt to create the specified test topic and produce messages to it.	`True` or `False`
`KAFKA_WRITER_TOPIC_NAME`	String	Name of the Kafka topic to be created by the Kafka writer for connectivity testing. Only used if `USE_KAFKA_WRITER` is `True`.	`connectivity-test-topic`

Environment Variable Name	Type	Description	Example
KAFKA_WRITER_TOPIC_PARTITION_COUNT	Integer	Number of partitions for the Kafka writer test topic. Only used if USE_KAFKA_WRITER is True.	3, 6
KAFKA_WRITER_TOPIC_REPLICATION_FACTOR	Integer	Replication factor for the Kafka writer test topic. Only used if USE_KAFKA_WRITER is True.	3
KAFKA_WRITER_TOPIC_DATA_RETENTION_IN_DAYS	Integer	Data retention period (in days) for the Kafka writer test topic. Only used if USE_KAFKA_WRITER is True.	0 (Infinite), 7

1.2.3 Using the AWS Secrets Manager (optional)

If you use **AWS Secrets Manager** to manage your secrets, set the USE_AWS_SECRETS_MANAGER variable to True and the tool will retrieve the secrets from AWS Secrets Manager using the names provided in CONFLUENT_CLOUD_API_KEY_AWS_SECRETS and KAFKA_API_KEY_AWS_SECRETS.

The code expects the CONFLUENT_CLOUD_API_KEY_AWS_SECRETS to be stored in JSON format with these keys:

- confluent_cloud_api_key
- confluent_cloud_api_secret

The code expects the KAFKA_API_KEY_AWS_SECRETS to be stored in JSON format with these keys:

- kafka_cluster_id
- bootstrap.servers
- sasl.username
- sasl.password

1.3 Run the Tool

Navigate to the Project Root Directory

Open your Terminal and navigate to the root folder of the kafka_cluster-topics-partition_count_recommender-tool/ repository that you have cloned. You can do this by executing:

```
cd path/to/kafka_cluster-topics-partition_count_recommender-tool/
```

> Replace path/to/ with the actual path where your repository is located.

Then enter the following command below to run the tool:

```
uv run python src/thread_safe_tool.py
```

If USE_SAMPLE_RECORDS environment variable is set to True, the tool will sample records from each topic to calculate the average record size in bytes. For example, below is a screenshot of the tool running successfully:

```
2025-09-30 10:35:43 - INFO - main - Retrieving the Confluent Cloud credentials from the .env file.
2025-09-30 10:35:45 - INFO - main -
================================================================================================
2025-09-30 10:35:45 - INFO - main - MULTITHREADED KAFKA CLUSTER ANALYSIS STARTING
2025-09-30 10:35:45 - INFO - main - ------------------------------------------------------------------------------------------
--------------------
2025-09-30 10:35:45 - INFO - main - Number of Kafka clusters to analyze: 1
2025-09-30 10:35:45 - INFO - main - Max concurrent Kafka clusters: 4
2025-09-30 10:35:45 - INFO - main - Max concurrent topics per cluster: 8
2025-09-30 10:35:45 - INFO - main - Analysis method: Record sampling
2025-09-30 10:35:45 - INFO - main -
================================================================================================
2025-09-30 10:35:51 - INFO - __log_initial_parameters -
================================================================================================
2025-09-30 10:35:51 - INFO - __log_initial_parameters - INITIAL ANALYSIS PARAMETERS
2025-09-30 10:35:51 - INFO - __log_initial_parameters - ------------------------------------------------------------
------------------------------------
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Analysis Timestamp: 2025-09-30T10:35:51.001810
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Using Confluent Cloud API Key to fetch Kafka credential: True
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Environment Filter: None
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Kafka Cluster Filter: None
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Principal ID Filter: sa-j5zz1w8
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Kafka Cluster ID: lkc-782no1
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Max worker threads: 8
```

```
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Connecting to Kafka cluster and retrieving metadata...
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Found 2 topics to analyze
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Excluding internal topics
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Required consumption throughput factor: 10.0
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Minimum required throughput threshold: 10.0 MB/s
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Topic filter: None
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Default Partition Count: 6
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Using sample records for average record size calculation
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Sampling batch size: 10000 records
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Sampling days: 1 days
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Sampling max consecutive nulls: 50 records
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Sampling timeout: 2.0 seconds
2025-09-30 10:35:51 - INFO - __log_initial_parameters - Sampling max continuous failed batches: 5 batches
2025-09-30 10:35:51 - INFO - __log_initial_parameters -
================================================================================================
2025-09-30 10:35:51 - INFO - analyze_all_topics - Created the lkc-782no1-recommender-1759242951-detail-report.csv
file
2025-09-30 10:35:51 - INFO - analyze_topic - [Thread-6117666816] Analyzing topic stock_trades with 1-day rolling
window (from 2025-09-29T14:35:51+00:00)
2025-09-30 10:35:51 - INFO - analyze_topic - [Thread-6134493184] Analyzing topic stock_trades_with_totals with 1-day
rolling window (from 2025-09-29T14:35:51+00:00)
2025-09-30 10:35:54 - INFO - __sample_record_sizes - [Thread-6117666816] Partition 000 of 006: using effective batch
size 5000 (requested: 10000, optimal: 5000)
2025-09-30 10:35:54 - INFO - __sample_record_sizes - [Thread-6117666816]     Sampling from partition 000 of 006:
offsets [55341, 104255)
2025-09-30 10:35:54 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 1: 5000 valid records (0
errors/nulls), progress: 10.2%, running avg: 83.41 bytes
2025-09-30 10:35:55 - INFO - __sample_record_sizes - [Thread-6134493184] Partition 000 of 006: using effective batch
size 5000 (requested: 10000, optimal: 5000)
2025-09-30 10:35:55 - INFO - __sample_record_sizes - [Thread-6134493184]     Sampling from partition 000 of 006:
offsets [31735, 59864)
2025-09-30 10:35:55 - WARNING - __sample_record_sizes - [Thread-6134493184] Failed to seek for
stock_trades_with_totals 000 of 006: KafkaError{code=_STATE,val=-172,str="Failed to seek to offset 31735: Local:
Erroneous state"}
2025-09-30 10:35:55 - INFO - __sample_record_sizes - [Thread-6134493184] Partition 001 of 006: using effective batch
size 5000 (requested: 10000, optimal: 5000)
2025-09-30 10:35:55 - INFO - __sample_record_sizes - [Thread-6134493184]     Sampling from partition 001 of 006:
offsets [32292, 63567)
2025-09-30 10:35:55 - WARNING - __sample_record_sizes - [Thread-6134493184] Failed to seek for
stock_trades_with_totals 001 of 006: KafkaError{code=_STATE,val=-172,str="Failed to seek to offset 32292: Local:
Erroneous state"}
2025-09-30 10:35:55 - INFO - __sample_record_sizes - [Thread-6134493184] Partition 002 of 006: using effective batch
size 5000 (requested: 10000, optimal: 5000)
2025-09-30 10:35:55 - INFO - __sample_record_sizes - [Thread-6134493184]     Sampling from partition 002 of 006:
offsets [32609, 61883)
2025-09-30 10:35:55 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 2: 5000 valid records (0
errors/nulls), progress: 20.4%, running avg: 83.40 bytes
2025-09-30 10:35:55 - WARNING - __sample_record_sizes - [Thread-6134493184] Failed to seek for
stock_trades_with_totals 002 of 006: KafkaError{code=_STATE,val=-172,str="Failed to seek to offset 32609: Local:
Erroneous state"}
2025-09-30 10:35:56 - INFO - __sample_record_sizes - [Thread-6134493184] Partition 003 of 006: using effective batch
size 5000 (requested: 10000, optimal: 5000)
2025-09-30 10:35:56 - INFO - __sample_record_sizes - [Thread-6134493184]     Sampling from partition 003 of 006:
offsets [31156, 59407)
2025-09-30 10:35:56 - WARNING - __sample_record_sizes - [Thread-6134493184] Failed to seek for
stock_trades_with_totals 003 of 006: KafkaError{code=_STATE,val=-172,str="Failed to seek to offset 31156: Local:
Erroneous state"}
2025-09-30 10:35:56 - INFO - __sample_record_sizes - [Thread-6134493184] Partition 004 of 006: using effective batch
size 5000 (requested: 10000, optimal: 5000)
2025-09-30 10:35:56 - INFO - __sample_record_sizes - [Thread-6134493184]     Sampling from partition 004 of 006:
offsets [33656, 62008)
2025-09-30 10:35:56 - WARNING - __sample_record_sizes - [Thread-6134493184] Failed to seek for
stock_trades_with_totals 004 of 006: KafkaError{code=_STATE,val=-172,str="Failed to seek to offset 33656: Local:
Erroneous state"}
2025-09-30 10:35:56 - INFO - __sample_record_sizes - [Thread-6134493184] Partition 005 of 006: using effective batch
size 5000 (requested: 10000, optimal: 5000)
2025-09-30 10:35:56 - INFO - __sample_record_sizes - [Thread-6134493184]     Sampling from partition 005 of 006:
offsets [34106, 63261)
2025-09-30 10:35:56 - WARNING - __sample_record_sizes - [Thread-6134493184] Failed to seek for
stock_trades_with_totals 005 of 006: KafkaError{code=_STATE,val=-172,str="Failed to seek to offset 34106: Local:
Erroneous state"}
2025-09-30 10:35:56 - WARNING - __sample_record_sizes - [Thread-6134493184] No records sampled from topic
'stock_trades_with_totals'
2025-09-30 10:35:56 - INFO - update_progress - Progress: 1 of 2 (50.0%) topics completed
2025-09-30 10:35:57 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 3: 5000 valid records (0
errors/nulls), progress: 30.7%, running avg: 83.42 bytes
2025-09-30 10:35:58 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 4: 5000 valid records (0
errors/nulls), progress: 40.9%, running avg: 83.43 bytes
2025-09-30 10:35:58 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 5: 5000 valid records (0
errors/nulls), progress: 51.1%, running avg: 83.43 bytes
2025-09-30 10:35:59 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 6: 5000 valid records (0
```

```
errors/nulls), progress: 61.3%, running avg: 83.43 bytes
2025-09-30 10:36:00 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 7: 5000 valid records (0
errors/nulls), progress: 71.6%, running avg: 83.44 bytes
2025-09-30 10:36:02 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 8: 5000 valid records (0
errors/nulls), progress: 81.8%, running avg: 83.44 bytes
2025-09-30 10:36:03 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 9: 5000 valid records (0
errors/nulls), progress: 92.0%, running avg: 83.44 bytes
2025-09-30 10:36:03 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 10: 3914 valid records (1
errors/nulls), progress: 100.0%, running avg: 83.44 bytes
2025-09-30 10:36:03 - INFO - __sample_record_sizes - [Thread-6117666816] Partition 002 of 006: using effective batch
size 5000 (requested: 10000, optimal: 5000)
2025-09-30 10:36:03 - INFO - __sample_record_sizes - [Thread-6117666816]     Sampling from partition 002 of 006:
offsets [27502, 52242)
2025-09-30 10:36:03 - WARNING - __sample_record_sizes - [Thread-6117666816] Failed to seek for stock_trades 002 of
006: KafkaError{code=_STATE,val=-172,str="Failed to seek to offset 27502: Local: Erroneous state"}
2025-09-30 10:36:03 - INFO - __sample_record_sizes - [Thread-6117666816] Partition 004 of 006: using effective batch
size 5000 (requested: 10000, optimal: 5000)
2025-09-30 10:36:03 - INFO - __sample_record_sizes - [Thread-6117666816]     Sampling from partition 004 of 006:
offsets [27597, 52015)
2025-09-30 10:36:03 - WARNING - __sample_record_sizes - [Thread-6117666816] Failed to seek for stock_trades 004 of
006: KafkaError{code=_STATE,val=-172,str="Failed to seek to offset 27597: Local: Erroneous state"}
2025-09-30 10:36:03 - INFO - __sample_record_sizes - [Thread-6117666816] Partition 005 of 006: using effective batch
size 5000 (requested: 10000, optimal: 5000)
2025-09-30 10:36:03 - INFO - __sample_record_sizes - [Thread-6117666816]     Sampling from partition 005 of 006:
offsets [82379, 156373)
2025-09-30 10:36:04 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 1: 5000 valid records (0
errors/nulls), progress: 6.8%, running avg: 83.62 bytes
2025-09-30 10:36:07 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 2: 5000 valid records (0
errors/nulls), progress: 13.5%, running avg: 83.78 bytes
2025-09-30 10:36:07 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 3: 5000 valid records (0
errors/nulls), progress: 20.3%, running avg: 83.90 bytes
2025-09-30 10:36:08 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 4: 5000 valid records (0
errors/nulls), progress: 27.0%, running avg: 84.02 bytes
2025-09-30 10:36:09 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 5: 5000 valid records (0
errors/nulls), progress: 33.8%, running avg: 84.11 bytes
2025-09-30 10:36:10 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 6: 5000 valid records (0
errors/nulls), progress: 40.5%, running avg: 84.19 bytes
2025-09-30 10:36:12 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 7: 5000 valid records (0
errors/nulls), progress: 47.3%, running avg: 84.27 bytes
2025-09-30 10:36:12 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 8: 5000 valid records (0
errors/nulls), progress: 54.1%, running avg: 84.33 bytes
2025-09-30 10:36:13 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 9: 5000 valid records (0
errors/nulls), progress: 60.8%, running avg: 84.39 bytes
2025-09-30 10:36:14 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 10: 5000 valid records (0
errors/nulls), progress: 67.6%, running avg: 84.44 bytes
2025-09-30 10:36:15 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 11: 5000 valid records (0
errors/nulls), progress: 74.3%, running avg: 84.49 bytes
2025-09-30 10:36:16 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 12: 5000 valid records (0
errors/nulls), progress: 81.1%, running avg: 84.53 bytes
2025-09-30 10:36:16 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 13: 5000 valid records (0
errors/nulls), progress: 87.8%, running avg: 84.57 bytes
2025-09-30 10:36:18 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 14: 5000 valid records (0
errors/nulls), progress: 94.6%, running avg: 84.61 bytes
2025-09-30 10:36:19 - INFO - __sample_record_sizes - [Thread-6117666816]       Batch 15: 3994 valid records (1
errors/nulls), progress: 100.0%, running avg: 84.64 bytes
2025-09-30 10:36:19 - INFO - __sample_record_sizes - [Thread-6117666816] Final average: 84.64 bytes from 122908
records
2025-09-30 10:36:19 - INFO - update_progress - Progress: 2 of 2 (100.0%) topics completed
2025-09-30 10:36:19 - INFO - __log_summary_stats -
================================================================================================
2025-09-30 10:36:19 - INFO - __log_summary_stats - ANALYSIS SUMMARY STATISTICS
2025-09-30 10:36:19 - INFO - __log_summary_stats - ------------------------------------------------------------------
----------------------------------
2025-09-30 10:36:19 - INFO - __log_summary_stats - Elapsed Time: 0.01 hours
2025-09-30 10:36:19 - INFO - __log_summary_stats - Total Topics: 2
2025-09-30 10:36:19 - INFO - __log_summary_stats - Active Topics: 2
2025-09-30 10:36:19 - INFO - __log_summary_stats - Active Topics %: 100.0%
2025-09-30 10:36:19 - INFO - __log_summary_stats - Total Partitions: 12
2025-09-30 10:36:19 - INFO - __log_summary_stats - Total Recommended Partitions: 16
2025-09-30 10:36:19 - INFO - __log_summary_stats - Non-Empty Topics Total Partitions: 12
2025-09-30 10:36:19 - INFO - __log_summary_stats - RECOMMENDED Increase in Partitions: 33.3%
2025-09-30 10:36:19 - INFO - __log_summary_stats - Total Records: 734875
2025-09-30 10:36:19 - INFO - __log_summary_stats - Average Partitions per Topic: 6
2025-09-30 10:36:19 - INFO - __log_summary_stats - Average Partitions per Active Topic: 6
2025-09-30 10:36:19 - INFO - __log_summary_stats - Average Recommended Partitions per Topic: 8
2025-09-30 10:36:19 - INFO - __log_summary_stats -
================================================================================================
2025-09-30 10:36:19 - INFO - _analyze_kafka_cluster - KAFKA CLUSTER lkc-782no1: TOPIC ANALYSIS COMPLETED
SUCCESSFULLY.
2025-09-30 10:36:19 - INFO - _analyze_kafka_cluster - Kafka API key QCU6SGWGYS5L6SWV for Kafka Cluster lkc-782no1
```

```
deleted successfully.
2025-09-30 10:36:19 - INFO - main - SINGLE KAFKA CLUSTER ANALYSIS COMPLETED SUCCESSFULLY.
```

If `USE_SAMPLE_RECORDS` is set to `False`, the tool will use the Confluent Cloud Metrics API to retrieve the average and peak consumption in bytes over a rolling seven-day period. For example, below is a screenshot of the tool running successfully:

```
2025-09-30 10:40:30 - INFO - main - Retrieving the Confluent Cloud credentials from the .env file.
2025-09-30 10:40:32 - INFO - main -
================================================================================================
2025-09-30 10:40:32 - INFO - main - MULTITHREADED KAFKA CLUSTER ANALYSIS STARTING
2025-09-30 10:40:32 - INFO - main - ------------------------------------------------------------------------------------
------------------
2025-09-30 10:40:32 - INFO - main - Number of Kafka clusters to analyze: 1
2025-09-30 10:40:32 - INFO - main - Max concurrent Kafka clusters: 4
2025-09-30 10:40:32 - INFO - main - Max concurrent topics per cluster: 8
2025-09-30 10:40:32 - INFO - main - Analysis method: Metrics API
2025-09-30 10:40:32 - INFO - main -
================================================================================================
2025-09-30 10:40:38 - INFO - __log_initial_parameters -
================================================================================================
2025-09-30 10:40:38 - INFO - __log_initial_parameters - INITIAL ANALYSIS PARAMETERS
2025-09-30 10:40:38 - INFO - __log_initial_parameters - ------------------------------------------------------------
------------------------------------
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Analysis Timestamp: 2025-09-30T10:40:38.342392
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Using Confluent Cloud API Key to fetch Kafka credential: True
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Environment Filter: None
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Kafka Cluster Filter: None
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Principal ID Filter: sa-j5zz1w8
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Kafka Cluster ID: lkc-782no1
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Max worker threads: 8
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Connecting to Kafka cluster and retrieving metadata...
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Found 2 topics to analyze
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Excluding internal topics
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Required consumption throughput factor: 10.0
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Minimum required throughput threshold: 10.0 MB/s
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Topic filter: None
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Default Partition Count: 6
2025-09-30 10:40:38 - INFO - __log_initial_parameters - Using Metrics API for average record size calculation
2025-09-30 10:40:38 - INFO - __log_initial_parameters -
================================================================================================
2025-09-30 10:40:38 - INFO - analyze_all_topics - Created the lkc-782no1-recommender-1759243238-detail-report.csv
file
2025-09-30 10:40:40 - INFO - analyze_topic_with_metrics - [Thread-6149812224] Confluent Metrics API - For topic
stock_trades, the average bytes per record is 157.28 bytes/record for a total of 365356 records.
2025-09-30 10:40:40 - INFO - update_progress - Progress: 1 of 2 (50.0%) topics completed
2025-09-30 10:40:40 - INFO - analyze_topic_with_metrics - [Thread-6166638592] Confluent Metrics API - For topic
stock_trades_with_totals, the average bytes per record is 112.34 bytes/record for a total of 365350 records.
2025-09-30 10:40:40 - INFO - update_progress - Progress: 2 of 2 (100.0%) topics completed
2025-09-30 10:40:40 - INFO - __log_summary_stats -
================================================================================================
2025-09-30 10:40:40 - INFO - __log_summary_stats - ANALYSIS SUMMARY STATISTICS
2025-09-30 10:40:40 - INFO - __log_summary_stats - ------------------------------------------------------------------
----------------------------------
2025-09-30 10:40:40 - INFO - __log_summary_stats - Elapsed Time: 0.00 hours
2025-09-30 10:40:40 - INFO - __log_summary_stats - Total Topics: 2
2025-09-30 10:40:40 - INFO - __log_summary_stats - Active Topics: 2
2025-09-30 10:40:40 - INFO - __log_summary_stats - Active Topics %: 100.0%
2025-09-30 10:40:40 - INFO - __log_summary_stats - Total Partitions: 12
2025-09-30 10:40:40 - INFO - __log_summary_stats - Total Recommended Partitions: 20
2025-09-30 10:40:40 - INFO - __log_summary_stats - Non-Empty Topics Total Partitions: 12
2025-09-30 10:40:40 - INFO - __log_summary_stats - RECOMMENDED Increase in Partitions: 66.7%
2025-09-30 10:40:40 - INFO - __log_summary_stats - Total Records: 730706
2025-09-30 10:40:40 - INFO - __log_summary_stats - Average Partitions per Topic: 6
2025-09-30 10:40:40 - INFO - __log_summary_stats - Average Partitions per Active Topic: 6
2025-09-30 10:40:40 - INFO - __log_summary_stats - Average Recommended Partitions per Topic: 10
2025-09-30 10:40:40 - INFO - __log_summary_stats -
================================================================================================
2025-09-30 10:40:40 - INFO - _analyze_kafka_cluster - KAFKA CLUSTER lkc-782no1: TOPIC ANALYSIS COMPLETED
SUCCESSFULLY.
2025-09-30 10:40:41 - INFO - _analyze_kafka_cluster - Kafka API key NAFW22BXFUA6I62N for Kafka Cluster lkc-782no1
deleted successfully.
2025-09-30 10:40:41 - INFO - main - SINGLE KAFKA CLUSTER ANALYSIS COMPLETED SUCCESSFULLY.
```

1.3.1 Did you notice we prefix `uv run` to `python src/thread_safe_tool.py`?

You maybe asking yourself why. Well, `uv` is an incredibly fast Python package installer and dependency resolver, written in **Rust**, and designed to seamlessly replace `pip`, `pipx`, `poetry`, `pyenv`, `twine`, `virtualenv`, and more in your workflows. By prefixing `uv run` to a command, you're ensuring that the command runs in an

optimal Python environment.

Now, let's go a little deeper into the magic behind `uv run`:

- When you use it with a file ending in `.py` or an HTTP(S) URL, `uv` treats it as a script and runs it with a Python interpreter. In other words, `uv run file.py` is equivalent to `uv run python file.py`. If you're working with a URL, `uv` even downloads it temporarily to execute it. Any inline dependency metadata is installed into an isolated, temporary environment—meaning zero leftover mess! When used with `-`, the input will be read from `stdin`, and treated as a Python script.
- If used in a project directory, `uv` will automatically create or update the project environment before running the command.
- Outside of a project, if there's a virtual environment present in your current directory (or any parent directory), `uv` runs the command in that environment. If no environment is found, it uses the interpreter's environment.

So what does this mean when we put `uv run` before `python src/thread_safe_tool.py`? It means `uv` takes care of all the setup—fast and seamless—right in your local environment. If you think AI/ML is magic, the work the folks at Astral have done with `uv` is pure wizardry!

Curious to learn more about Astral's `uv`? Check these out:

- Documentation: Learn about `uv`.
- Video: `uv` IS THE FUTURE OF PYTHON PACKING!.

If you have Kafka connectivity issues, you can verify connectivity using the following command:

1.3.2 Troubleshoot Connectivity Issues (if any)

To verify connectivity to your Kafka cluster, you can use the `kafka-topics.sh` command-line tool. First, download the Kafka binaries from the Apache Kafka website and extract them. Navigate to the `bin` directory of the extracted Kafka folder. Second, create a `client.properties` file with your Kafka credentials:

```
# For SASL_SSL (most common for cloud services)
security.protocol=SASL_SSL
sasl.mechanism=PLAIN
sasl.jaas.config=org.apache.kafka.common.security.plain.PlainLoginModule required \
  username="<YOUR_KAFKA_API_KEY>" \
  password="<YOUR_KAFKA_API_SECRET>";

# Additional SSL settings if needed
ssl.endpoint.identification.algorithm=https
```

Finally, run the following command to list all topics in your Kafka cluster:

```
./kafka-topics.sh --list --bootstrap-server <YOUR_BOOTSTRAP_SERVER_URI> --command-config ./client.properties
```

If the connection is successful, you should see a list of topics in your Kafka cluster. If you encounter any errors, double-check your credentials and network connectivity.

1.3.3 Running the Tool's Unit Tests (i.e., PyTests)

To ensure the tool is functioning as expected, you can run the provided unit tests. These tests cover various aspects of the tool's functionality, such as Kafka credential fetching.

Navigate to the Project Root Directory

Open your Terminal and navigate to the root folder of the `kafka_cluster-topics-partition_count_recommender-tool/` repository that you have cloned. You can do this by executing:

```
cd path/to/kafka_cluster-topics-partition_count_recommender-tool/
```

> Replace `path/to/` with the actual path where your repository is located.

Then enter the following commands below to run the test suites:

```
uv run pytest -s tests/test_fetch_kafka_credentials_via_confluent_cloud_api_key.py
```

```
uv run pytest -s tests/test_fetch_kafka_credentials_via_environment_variables.py
```

```
uv run pytest -s tests/test_metrics_client.py
```

```
uv run pytest -s tests/test_environment_client.py
```

```
uv run pytest -s tests/test_schema_registry_client.py
```

```
uv run pytest -s tests/test_iam_client.py
```

You should see output indicating the results of the tests, including any failures or errors. If all tests pass, it confirms that the tool is working correctly.

1.4 The Results

1.4.1 Detail and Summary Report written to CSV files

The tool automatically generates two comprehensive CSV reports for each Kafka Cluster that transform raw analysis into actionable insights:

- **Detail Report CSV.** For every topic analyzed, this report captures the topic's average consumer throughput (MB/s), its required throughput (MB/s), and a calculated recommended partition count, ensuring precise alignment between workload demand and partitioning strategy. Below is a screenshot of a sample detail report:

  ```
  method,topic_name,is_compacted,number_of_records,number_of_partitions,required_throughput,consumer_throughput,re
  commended_partitions,status
  sampling_records,stock_trades_with_totals,no,369990,6,0.0,0.0,6,active
  sampling_records,stock_trades,no,364885,6,294.51870118954434,29.451870118954435,10,active
  ```

- **Summary Report CSV.** Once all topics have been evaluated, this report consolidates the results into a high-level overview, providing a clear, data-driven snapshot of cluster-wide throughput patterns and partitioning recommendations. Below is a screenshot of a sample summary report:

  ```
  stat,value
  elapsed_time_hours,0.007820380065176222
  method,sampling_records
  required_consumption_throughput_factor,10
  minimum_required_throughput_threshold,10.0
  default_partition_count,6
  sampling_batch_size,10000
  sampling_days,1
  sampling_max_consecutive_nulls,50
  sampling_timeout,2.0
  sampling_max_continuous_failed_batches,5
  total_topics,2
  internal_topics_included,False
  topic_filter,None
  active_topic_count,2
  active_topic_percentage,100.0
  total_partitions,12
  total_recommended_partitions,16
  active_total_partition_count,12
  percentage_decrease,0.0
  percentage_increase,33.33333333333333
  total_records,734875
  average_partitions_per_topic,6.0
  active_average_partitions_per_topic,6.0
  average_recommended_partitions_per_topic,8.0
  ```

> The names of the CSV comprises of the `<KAFKA CLUSTER ID>-recommender-<CURRENT EPOCH TIME IN SECONDS WHEN THE TOOL STARTED>-detail-report.csv` and `<KAFKA CLUSTER ID>-recommender-<CURRENT EPOCH TIME IN SECONDS WHEN THE TOOL STARTED>-summary-report.csv`, respectively.

1.4.2 Detail Results Produced to Kafka

2.0 How the tool calculates the recommended partition count

The tool uses the Kafka `AdminClient` to retrieve all Kafka Topics (based on the `TOPIC_FILTER` specified) stored in your Kafka Cluster, including the original partition count per topic. Then, it iterates through each Kafka Topic, calling the Confluent Cloud Metrics RESTful API to retrieve the topic's average (i.e., the *Consumer Throughput*) and peak consumption in bytes over a rolling seven-day period. Next, it calculates the required throughput by multiplying the peak consumption by the `REQUIRED_CONSUMPTION_THROUGHPUT_FACTOR` (i.e., the *Required Throughput*). Finally, it divides the required throughput by the consumer throughput and rounds the result to the nearest whole number to determine the optimal number of partitions.

> **Note**: *This why the tool requires the Kafka API Key and Secret to connect to your Kafka Cluster via the AdminClient, and the Confluent Cloud API Key and Secret to connect to the Confluent Cloud Metrics API.*

For example, suppose you have a consumer that consumes at **25MB/s**, but the the consumer requirement is a throughput of **1.22GB/s**. How many partitions should you have?

To determine the number of partitions needed to support a throughput of **1.22GB/s** for a Kafka consumer that can only consume at **25MB/s**, you can calculate it as follows:

1. Convert the target throughput to the same units:
    - **1.22GB/s = 1250MB/s**
2. Divide the target throughput by the consumer's capacity:

$$\textbf{Partition Count} = \frac{Required\ Throughput}{Consumer\ Throughput} = \frac{1250\ MB/s}{25\ MB/s} = \mathbf{50}$$

3. Since you can only have a whole number of partitions, you should always round up to the nearest whole number:

$$\textbf{Partition Count} = \mathbf{50}$$

The **50 partitions** ensure that the consumer can achieve the required throughput of **1.22GB/s** while consuming at a rate of **25MB/s** per partition. This will allow the workload to be distributed across partitions so that multiple consumers can work in parallel to meet the throughput requirement.

2.1 End-to-End Tool Workflow

```
sequenceDiagram
    participant Main as Main Thread
    participant EC as EnvironmentClient
    participant AWS as AWS Secrets Manager
    participant KTA as KafkaTopicsAnalyzer
    participant TPE as ThreadPoolExecutor
    participant Worker as Worker Thread
    participant TA as TopicAnalyzer
    participant KC as Kafka Consumer
    participant MC as MetricsClient
    participant CSV as CSV Writer

    Main->>Main: Load environment variables
    Main->>Main: Read configuration settings

    alt Use AWS Secrets Manager
        Main->>AWS: get_secrets(region, secret_name)
        AWS-->>Main: Return credentials
    else Use environment variables
        Main->>Main: Read from .env file
    end

    alt Use Confluent Cloud API Key
        Main->>EC: Create EnvironmentClient
        Main->>EC: get_environment_list()
        EC-->>Main: Return environments
        Main->>EC: get_kafka_cluster_list(env_id)
        EC-->>Main: Return kafka clusters
        loop For each cluster
            Main->>EC: create_api_key(kafka_cluster_id, principal_id)
            EC-->>Main: Return API key pair
        end
    else Use existing credentials
        Main->>Main: Load kafka credentials from env/secrets
    end

    Main->>KTA: Create ThreadSafeKafkaTopicsAnalyzer
    Main->>KTA: analyze_all_topics()

    KTA->>KTA: __get_topics_metadata()
    KTA->>KTA: Get cluster metadata via AdminClient
    KTA->>KTA: Filter topics (internal, topic_filter)
    KTA->>KTA: Get topic configurations (retention, cleanup policy)

    KTA->>CSV: Create ThreadSafeCsvWriter
    CSV->>CSV: Initialize CSV file with headers

    KTA->>KWRITER: Create ThreadSafeKafkaWriter
    KWRITER->>KWRITER: Initialize Kafka producer for logging results

    alt Single cluster
        KTA->>KTA: _analyze_kafka_cluster() directly
```

```
    else Multiple clusters
        KTA->>TPE: Create ThreadPoolExecutor(max_cluster_workers)
        loop For each cluster
            KTA->>TPE: Submit _analyze_kafka_cluster task
        end
    end

    KTA->>TPE: Create ThreadPoolExecutor(max_workers_per_cluster)

    loop For each topic
        KTA->>TPE: Submit analyze_topic_worker task

        TPE->>Worker: Execute in worker thread
        Worker->>TA: Create ThreadSafeTopicAnalyzer

        alt Use sample records
            rect rgb(173, 216, 230)
                Worker->>TA: analyze_topic()
                TA->>KC: Create unique Consumer instance
                TA->>KC: get_watermark_offsets()
                KC-->>TA: Return low/high watermarks
                TA->>KC: offsets_for_times() for timestamp
                KC-->>TA: Return offset at timestamp

                loop For each partition
                    TA->>KC: assign([TopicPartition])
                    TA->>KC: seek(offset)
                    loop Batch processing
                        TA->>KC: poll(timeout)
                        KC-->>TA: Return record or None
                        TA->>TA: Calculate record size
                        TA->>TA: Update running totals
                    end
                end
                TA->>KC: close()
                TA-->>Worker: Return analysis result
            end

        else Use Metrics API
            rect rgb(255, 182, 193)
                Worker->>TA: analyze_topic_with_metrics()
                TA->>MC: Create MetricsClient
                TA->>MC: get_topic_daily_aggregated_totals(RECEIVED_BYTES)
                MC-->>TA: Return bytes metrics
                TA->>MC: get_topic_daily_aggregated_totals(RECEIVED_RECORDS)
                MC-->>TA: Return records metrics
                TA->>TA: Calculate avg bytes per record
                TA-->>Worker: Return analysis result
            end
        end

        Worker->>KTA: __process_and_write_result()
        KTA->>KTA: Calculate recommendations
        KTA->>CSV: write_row() [thread-safe]
        KTA->>KWRITER: write_result() [thread-safe]
        Worker-->>TPE: Return success/failure
    end

    TPE-->>KTA: All topic analysis complete
    KTA->>KTA: __calculate_summary_stats()
    KTA->>KTA: __write_summary_report()
    KTA->>KTA: __log_summary_stats()

    alt Confluent Cloud API cleanup
        loop For each created API key
            KTA->>EC: delete_api_key(api_key)
            EC-->>KTA: Confirm deletion
        end
    end

    KTA-->>Main: Return analysis success/failure
    Main->>Main: Log final results
    Main->>Main: Exit tool
```

3.0 Unlocking High-Performance Consumer Throughput

The throughput of a **Kafka consumer** refers to the rate at which it can read data from Kafka topics, typically measured in terms of **megabytes per second (MB/s)** or **records per second**. Consumer throughput depends on several factors, including the configuration of Kafka, the consumer tool, and the underlying infrastructure.

3.1 Key Factors Affecting Consumer Throughput

3.1.1 Partitions

- Throughput scales with the number of partitions assigned to the consumer. A consumer can read from multiple partitions concurrently, but the total throughput is bounded by the number of partitions and their data production rates.
- Increasing the number of partitions can improve parallelism and consumer throughput.

3.1.2 Consumer Parallelism

- A single consumer instance reads from one or more partitions, but it can be overwhelmed if the data rate exceeds its capacity.
- Adding more consumers in a consumer group increases parallelism, as Kafka reassigns partitions to balance the load.

3.1.3 Fetch Configuration

- `fetch.min.bytes`: Minimum amount of data (in bytes) the broker returns for a fetch request. Larger values reduce fetch requests but may introduce latency.
- `fetch.max.bytes`: Maximum amount of data returned in a single fetch response. A higher value allows fetching larger batches of messages, improving throughput.
- `fetch.max.wait.ms`: Maximum time the broker waits before responding to a fetch request. A higher value can increase batch sizes and throughput but may increase latency.

For more details, see the Confluent Cloud Client Optimization Guide - Consumer Fetching.

3.1.4 Batch Size

- Consumers process messages in batches for better efficiency. Larger batches reduce processing overhead but require sufficient memory.
- Configuration: `max.poll.records` controls the number of records fetched in a single poll.

3.1.5 Message Size

- Larger messages can reduce throughput if the network or storage systems are bottlenecks. Use compression (e.g., `lz4`, `snappy`) to optimize data transfer.

3.1.6 Network Bandwidth

- Network speed between Kafka brokers and consumers is critical. A consumer running on a limited-bandwidth network will see reduced throughput.

3.1.7 Deserialization Overhead

- The time required to deserialize records impacts throughput. Efficient deserialization methods (e.g., Avro, Protobuf with optimized schemas) can help.

3.1.8 Broker Load

- Broker performance and replication overhead impact the throughput seen by consumers. If brokers are under heavy load, consumer throughput may decrease.

3.1.9 Consumer Poll Frequency

- Consumers must frequently call `poll()` to fetch messages. If the consumer spends too much time processing messages between polls, throughput can drop.

3.1.10 System Resources

- CPU, memory, and disk I/O on the consumer's machine affect how fast it can process data.

3.2 Typical Consumer Throughput

- **Single Partition Throughput**: A single consumer reading from a single partition can typically achieve **10-50 MB/s** or higher, depending on record size, compression, and hardware.
- **Multi-Partition Throughput**: For a consumer group reading from multiple partitions, throughput can scale linearly with the number of partitions (subject to other system limits).

3.3 Seven Strategies to Improve Consumer Throughput

1. **Increase Partitions**: Scale partitions to allow more parallelism.
2. **Add Consumers**: Add more consumers in the consumer group to distribute the load.
3. **Optimize Fetch Settings**: Tune `fetch.min.bytes`, `fetch.max.bytes`, and `fetch.max.wait.ms`.
4. **Batch Processing**: Use `max.poll.records` to fetch and process larger batches.
5. **Compression**: Enable compression to reduce the amount of data transferred.
6. **Efficient SerDe (Serialization/Deserialization)**: Use optimized serializers and deserializers.
7. **Horizontal Scaling**: Ensure consumers run on high-performance hardware with sufficient network bandwidth.

By optimizing these factors, Kafka consumers can achieve higher throughput tailored to the specific use case and infrastructure.

4.0 Resources

4.1 Optimization Guides

- Optimize Confluent Cloud Clients for Throughput
- Choose and Change the Partition Count in Kafka

4.2 Confluent Cloud Metrics API

- Confluent Cloud Metrics API
- Confluent Cloud Metrics API: Metrics Reference
- Confluent Cloud Metrics

4.3 Confluent Kafka Python Client

- Confluent Kafka Python Client Documentation