Quick start

This guide is designed to help you get up and running with RisingWave quickly and easily. In this guide, we will walk you through the common tasks of using RisingWave.

Install and start RisingWave

Ensure you have Homebrew installed and then run the following commands:

```
brew tap risingwavelabs/risingwave
brew install risingwave
risingwave playground
```

INFO

This method launches RisingWave in playground mode, where data is temporarily stored in memory. The service is designed to automatically terminate after 30 minutes of inactivity, and any data stored will be deleted upon termination. We recommend using this method for quick tests only.

Other installation options are available. If you intend to deploy RisingWave to production environments, use Kubernetes Operator or RisingWave Cloud, our fully managed service.

Connect to RisingWave

After RisingWave is up and running, connect to it via the Postgres interactive terminal `psql`. Ensure you have `psql` installed in your environment. To learn about how to install it, see Install `psql` without PostgreSQL.

Open a new terminal window and run:

```
psql -h localhost -p 4566 -d dev -U root
```

Notes about the `psql` options:

- The `-h` option is used to specify the host name or IP address of the PostgreSQL server to connect to.
- The `-p` option is used to specify the port number that the server is listening on.
- The `-d` option is used to specify the name of the database to connect to.
- The `-U` option is used to specify the name of the database user to connect as.
- By default, the PostgreSQL server uses the `root user` account to authenticate connections to the `dev` database. Note that this user account does not require a password to connect.

Create a table

As RisingWave is a database, you can directly create a table and insert data into it. For example, let's create a table to store data about web page visits.

```
CREATE TABLE website_visits (
  timestamp timestamp,
  user_id varchar,
  page_id varchar,
  action varchar
);
```

Insert data

You can get data into RisingWave in two ways, directly inserting data and consuming data from streaming data sources.

Inserting data into RisingWave is the same as inserting data in any SQL database. Let's insert 5 rows of data to table `website_visits`.

```
INSERT INTO website_visits (timestamp, user_id, page_id, action) VALUES
  ('2023-06-13T10:00:00Z', 'user1', 'page1', 'view'),
  ('2023-06-13T10:01:00Z', 'user2', 'page2', 'view'),
  ('2023-06-13T10:02:00Z', 'user3', 'page3', 'view'),
  ('2023-06-13T10:03:00Z', 'user4', 'page1', 'view'),
  ('2023-06-13T10:04:00Z', 'user5', 'page2', 'view');
```

Connect to a source

The most common way for getting streaming data into RisingWave is through upstream sources such as message queues or Change Data Capture streams. For streaming data ingestion, you need use the `CREATE SOURCE` command to connect to a source first.

Let's assume that you have entered five rows of data in the same schema as table `website_visits` into the `test` topic in Kafka:

```
{"timestamp": "2023-06-13T10:05:00Z", "user_id": "user1", "page_id": "page1",
"action": "click"}
{"timestamp": "2023-06-13T10:06:00Z", "user_id": "user2", "page_id": "page2",
"action": "scroll"}
{"timestamp": "2023-06-13T10:07:00Z", "user_id": "user3", "page_id": "page1",
"action": "view"}
{"timestamp": "2023-06-13T10:08:00Z", "user_id": "user4", "page_id": "page2",
"action": "view"}
{"timestamp": "2023-06-13T10:09:00Z", "user_id": "user5", "page_id": "page3",
"action": "view"}
```

You can now connect to the topic from RisingWave by running the following command:

```
CREATE SOURCE IF NOT EXISTS website_visits_stream (
 timestamp timestamp,
 user_id varchar,
 page_id varchar,
 action varchar
 )
WITH (
 connector='kafka',
 topic='test',
 properties.bootstrap.server='localhost:9092',
 scan.startup.mode='earliest'
 ) FORMAT PLAIN ENCODE JSON;
```

Note that after the source is created, data is not automatically ingested into RisingWave. You need to create a materialized view to start the data movement.

RisingWave supports ingesting data from sources including mainstream message queues and databases. For supported sources and formats, see Supported sources and Supported formats.

Transform data with materialized views

In RisingWave, data are joined and transformed via materialized views. You do not need to set up processing jobs or pipelines.

A materialized views can be created on tables, sources, or joined data between tables and sources.

Let's create a materialized view to get the total page visits, unique visitors, and the last visit time for each page based on the data in source `website_visits_stream`.

```
CREATE MATERIALIZED VIEW visits_stream_mv AS
SELECT page_id,
count(*) AS total_visits,
count(DISTINCT user_id) AS unique_visitors,
max(timestamp) AS last_visit_time
FROM website_visits_stream
GROUP BY page_id;
```

Query data

Use the `SELECT` command to query data in a table or materialized view.

For example, let's see the latest results of the `visits_stream_mv` materialized view:

```
SELECT * FROM visits_stream_mv;

 page_id | total_visits | unique_visitors |   last_visit_time
---------+--------------+-----------------+---------------------
 page2   |            2 |               2 | 2023-06-13 10:08:00
 page1   |            2 |               2 | 2023-06-13 10:07:00
 page3   |            1 |               1 | 2023-06-13 10:09:00
(3 rows)
```

As new data comes in, the results in `visits_stream_mv` will be automatically updated. Behind the scenes, RisingWave performs incremental computations when new data comes in.

For example, if you enter five more rows of data into the `test` topic:

```
{"timestamp": "2023-06-13T10:10:00Z", "user_id": "user1", "page_id": "page3",
"action": "scroll"}
{"timestamp": "2023-06-13T10:11:00Z", "user_id": "user2", "page_id": "page1",
"action": "click"}
{"timestamp": "2023-06-13T10:12:00Z", "user_id": "user3", "page_id": "page2",
"action": "scroll"}
{"timestamp": "2023-06-13T10:13:00Z", "user_id": "user4", "page_id": "page3",
"action": "view"}
```

```
{"timestamp": "2023-06-13T10:14:00Z", "user_id": "user5", "page_id": "page1",
"action": "click"}
```

The results will be automatically updated:

```
SELECT * FROM visits_stream_mv;

 page_id | total_visits | unique_visitors |   last_visit_time
---------+--------------+-----------------+---------------------
 page2   |            3 |               3 | 2023-06-13 10:12:00
 page3   |            3 |               3 | 2023-06-13 10:13:00
 page1   |            4 |               4 | 2023-06-13 10:14:00
(3 rows)
```

Sink data out of RisingWave

Data in tables and materialized views are stored in RisingWave. You can sink data out of RisingWave and into Kafka topics or databases.

To sink data out of RisingWave, you need to create a sink using the `CREATE SINK`. A sink can be created from an existing table, source, or materialized view, or an ad-hoc `SELECT` query.

Let's sink all data from `visits_stream_mv` to a Kafka topic:

```
CREATE SINK sink1 FROM visits_stream_mv
WITH (
connector='kafka',
type='append-only',
force_append_only='true',
properties.bootstrap.server='localhost:9092',
topic='sink1'
);
```

*Last updated on **Oct 9, 2023***

0 comments

Write Preview Aa

Sign in to comment

Help us make this doc better! File an issue Edit this page

Was this page helpful?

Yes No

Powered by Happy React