Quick start

This guide aims to provide a quick and easy way to get started with RisingWave.

Step 1: Start RisingWave

> **INFO**
>
> The following options start RisingWave in the standalone mode. In this mode, data is stored in the file system and the metadata is stored in the embedded SQLite database. See About RisingWave standalone mode for more details.
>
> For extensive testing or single-machine deployment, consider starting RisingWave via Docker Compose. For production environments, consider RisingWave Cloud, our fully managed service, or deployment on Kubernetes using the Operator or Helm Chart.

Script installation

Open a terminal and run the following `curl` command.

```
curl https://risingwave.com/sh | sh
```

To start a RisingWave instance, run the following command.

```
risingwave
```

Docker

Ensure Docker Desktop is installed and running in your environment.

```
docker run -it --pull=always -p 4566:4566 -p 5691:5691
risingwavelabs/risingwave:latest single_node
```


Homebrew

Ensure Homebrew is installed, and run the following commands:

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

Step 2: Connect to RisingWave

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
```

Step 3: Insert some data

RisingWave supports both direct data insertion and streaming data ingestion from sources like message queues and database change streams.

To keep things simple, we'll demonstrate the approach of direct data insertion. Let's create a table and insert some data.

For instance, we can create a table named `exam_scores` to store information about examination scores.

Create the table

```
CREATE TABLE exam_scores (
  score_id int,
  exam_id int,
  student_id int,
  score real,
  exam_date date
);
```

Insert five rows of data

Ask AI

```
INSERT INTO exam_scores (score_id, exam_id, student_id, score, exam_date)
VALUES
```

```
(1, 101, 1001, 85.5, '2022-01-10'),
(2, 101, 1002, 92.0, '2022-01-10'),
(3, 101, 1003, 78.5, '2022-01-10'),
(4, 102, 1001, 91.2, '2022-02-15'),
(5, 102, 1003, 88.9, '2022-02-15');
```

Step 4: Analyze and query data

As an example, let's create a materialized view to calculate the average score for each examination, and then view the current result of the materialized view.

Create a materialized view

```
CREATE MATERIALIZED VIEW average_exam_scores AS
SELECT
    exam_id,
    AVG(score) AS average_score,
    COUNT(score) AS total_scores
FROM
    exam_scores
GROUP BY
    exam_id;
```

Query the current result

```
SELECT * FROM average_exam_scores;
------
 exam_id |   average_score   | total_scores
---------+-------------------+--------------
     102 | 90.05000000000001 |            2
     101 | 85.33333333333333 |            3
(2 rows)
```

As new data is received, the `average_exam_scores` materialized view will be automatically updated. RisingWave performs incremental computations in the background to keep the results up to date.

Now, let's insert five additional rows of data into the `exam_scores` table and query the latest result from the `average_exam_scores` materialized view. This will provide us with the updat[illegible] for each examination.

Ask AI

Insert more data

```
INSERT INTO exam_scores (score_id, exam_id, student_id, score, exam_date)
VALUES
  (11, 101, 1004, 89.5, '2022-05-05'),
  (12, 101, 1005, 93.2, '2022-05-05'),
  (13, 102, 1004, 87.1, '2022-06-10'),
  (14, 102, 1005, 91.7, '2022-06-10'),
  (15, 102, 1006, 84.3, '2022-06-10');
```

Query the latest result

```
SELECT * FROM average_exam_scores;
------
 exam_id |   average_score   | total_scores
---------+-------------------+--------------
     101 |             87.74 |            5
     102 | 88.64000000000001 |            5
(2 rows)
```

About RisingWave standalone mode

RisingWave standalone mode is a simplified deployment mode for RisingWave. It is designed to be minimal, easy to install, and configure.

Unlike other deployment modes, for instance Docker Compose or Kubernetes, RisingWave standalone mode starts the cluster as a single process. This means that services like `compactor`, `frontend`, `compute` and `meta` are all embedded in this process.

For state store, we will use the embedded `LocalFs` Object Store, eliminating the need for an external service like `minio` or `s3`; for meta store, we will use the embedded `SQLite` database, eliminating the need for an external service like `etcd`.

By default, the RisingWave standalone mode will store its data in `~/risingwave`, which includes both `Metadata` and `State Data`.

For a batteries-included setup, with `monitoring` tools and external services like `kafka` fully included, you can use Docker Compose instead. If you would like to set up these external services manually, you may check out RisingWave's Docker Compose, and run these services using the same configurations.

Ask AI

Configure RisingWave standalone mode

The instance of RisingWave standalone mode can run without any configuration. However, there are some options available to customize the instance.

The main options which new users may require would be the state store directory (`--state-store-directory`) and in-memory mode (`--in-memory`).

`--state-store-directory` specifies the new directory where the cluster's `Metadata` and `State Data` will reside. The default is to store it in the `~/risingwave` folder.

```
# Reconfigure RisingWave to be stored under 'projects' folder instead.
risingwave --state-store-directory ~/projects/risingwave
```

`--in-memory` will run an in-memory instance of RisingWave, both `Metadata` and `State Data` will not be persisted.

```
risingwave --in-memory
```

You can view other options with:

```
risingwave single --help
```

Monitoring RisingWave standalone mode with Grafana and Prometheus

To monitor your standalone cluster, you may wish to integrate metrics monitoring with Grafana and Prometheus.

First install Grafana and Prometheus.

Next, clone the RisingWave repository, it contains various configuration files.

Start the RisingWave standalone cluster.

Make sure you're in the `RisingWave` directory.

Start your prometheus instance:

```
prometheus --config.file=./standalone/prometheus.yaml --web.list
address=0.0.0.0:9500
```

Then start the Grafana instance:

```
grafana server --config ./standalone/grafana.ini
```

Next, add the Prometheus Data Source on the Grafana Dashboard: http://localhost:3001/connections/datasources/prometheus.

```
name: risedev-prometheus
Prometheus Server URL: http://localhost:9500
```

Finally, add the User and Dev Dashboard: http://localhost:3001/dashboard/import. The file paths are `grafana/risingwave-dev-dashboard.json`, `grafana/risingwave-user-dashboard.json`.

With that you can now monitor your standalone cluster with Grafana and Prometheus.

What's next?

Congratulations! You've successfully started RisingWave and conducted some initial data analysis. To explore further, you may want to:

- Check out the ready-to-run examples:
  - Example A: Ingest data from Kafka
  - Example B: Ingest data from Postgres CDC
- See this GitHub directory for ready-to-run demos and integration examples.
- Read our documentation to learn about how to ingest data from data streaming sources, transform data, and deliver data to downstream systems.

*Last updated on **Sep 30, 2024***