tinybird

Events API

The Events API enables high-throughput streaming ingestion into Tinybird from an easy-to-use HTTP API.

This page gives examples of how to use the Events API to perform various tasks. For more information, read the Events API Reference docs.

Send individual JSON events

You can send individual JSON events to the Events API by including the JSON event in the Request body.

For example, to send an individual JSON event using cURL:

SENDING INDIVIDUAL JSON EVENTS

```
curl \
-H "Authorization: Bearer <DATASOURCE:APPEND token>" \
-d '{"date": "2020-04-05 00:05:38", "city": "Chicago"}' \
'https://api.tinybird.co/v0/events?name=events_test'
```

The `name` parameter defines the name of the Data Source in which to insert events. If the Data Source does not exist, Tinybird creates the Data Source by inferring the schema of the JSON.

The Auth Token used to send data to the Events API needs the appropriate scopes. To append data to an existing Data Source, the `DATASOURCE:APPEND` scope is required. If the Data Source does not already exist, the `DATASOURCE:CREATE` scope is required to create the new Data Source.

> You can define the schema upfront using the CLI. However, a quick way to bootstrap this is by sending a single event over the Events API first and allowing Tinybird to create the schema for you automatically. Then, go to the Data Source schema page in the UI and download the schema as a text file. Edit the text file to adjust the schema if required, then drag & drop the file back on to the UI. The UI then creates the Data Source for you.

Send batches of JSON events

Sending batches of events enables you to achieve much higher total throughput than sending individual events.

You can send batches of JSON events to the Events API by formatting the events as NDJSON (newline delimited JSON). Each individual JSON event should be separated by a newline (`\n`) character.

SENDING BATCHES OF JSON EVENTS

```
curl \
-H "Authorization: Bearer <import_token>" \
-d $'{"date": "2020-04-05 00:05:38", "city": "Chicago"}\n{"date": "2020-04-05 00
'https://api.tinybird.co/v0/events?name=events_test'
```

The `name` parameter defines the name of the Data Source in which to insert events. If the Data Source does not exist, Tinybird creates the Data Source by inferring the schema of the JSON.

> The Auth Token used to send data to the Events API must have the appropriate scopes. To append data to an existing Data Source, the `DATASOURCE:APPEND` scope is required. If the Data Source does not already exist, the `DATASOURCE:CREATE` scope is required to create the new Data Source.

Limits

The Events API will deliver an out of the box capacity of:

- Up to 1000 requests/second
- Up to 20MB/s
- Up to 10MB per request

Throughput above these limits is offered as best-effort.

> The Events API is able to scale beyond these limits. If you are reaching these limits, contact support@tinybird.co.

Compression

NJDSON events sent to the Events API can be compressed with Gzip. However, it is only recommended to do this when necessary, such as when you have big events that are grouped

into large batches. Compressing events adds overhead to the ingestion process, which can introduce latency, although it is typically minimal.

Here is an example of sending a JSON event compressed with Gzip from the command line:

```
echo '{"timestamp":"2022-10-27T11:43:02.099Z","transaction_id":"8d1e1533-6071-4b

curl \
    -X POST 'https://api.tinybird.co/v0/events?name=gzip_events_example' \
    -H "Authorization: Bearer <AUTH_TOKEN>." \
    -H "Content-Encoding: gzip" \
    --data-binary @body.gz
```

Write acknowledgements

When you send data to the Events API, you usually recieve a `HTTP202` response, which indicates that the request was successful - however it does not confirm that the data has been committed into the underlying database. This is useful when guarantees on writes are not strictly necessary. Typically, it should take under 2 seconds to recieve a response from the Events API in this case.

```
curl \
    -X POST 'https://api.tinybird.co/v0/events?name=events_example' \
    -H "Authorization: Bearer <AUTH_TOKEN>" \
    -d $'{"timestamp":"2022-10-27T11:43:02.099Z"}'

< HTTP/2 202
< content-type: application/json
< content-length: 42
<
{"successful_rows":2,"quarantined_rows":0}
```

However, if your use case requires absolute gauarantees that data is committed, use the `wait` parameter.

The `wait` parameter is a boolean that accepts a value of `true` or `false`. A value of `false` is the default behavior, equivalent to ommitting the parameter entirely.

Using `wait=true` with your request will ask the Events API to wait for acknowledgement that the data you sent has been committed into the underlying database. You will recieve a `HTTP200` response that confirms data has been committed.

Note that adding `wait=true` to your request can result in a slower response time, and we recommend having a time-out of at least 10 seconds when waiting for the response.

For example:

```
curl \
    -X POST 'https://api.tinybird.co/v0/events?name=events_example&wait=true' \
    -H "Authorization: Bearer <AUTH_TOKEN>" \
    -d $'{"timestamp":"2022-10-27T11:43:02.099Z"}'

< HTTP/2 200
< content-type: application/json
< content-length: 42
<
{"successful_rows":2,"quarantined_rows":0}
```

It is good practice to log your requests to, and responses from, the Events API. This will help to give you visibility into any failures for reporting or recovery.

Integrations

Amazon S3

Kafka Data Streams

Google Cloud Storage

Google BigQuery

Snowflake

Confluent

Use cases

In-Product Analytics

Operational Intelligence

Realtime Personalization

Anomaly Detection & Alerts

Usage Based Pricing

Sports Betting/Gaming

Smart Inventory Management

Serverless ClickHouse

Spain
Calle del Dr. Fourquet, 27
28012 Madrid

USA
41 East 11th Street 11th floor
New York, NY 10003