tinybird

S3 Sink

Tinybird's S3 Sink allows you to offload data to Amazon S3, either on a pre-defined schedule or on demand. It's good for a variety of different scenarios where Amazon S3 is the common ground, for example:

- You're building a platform on top of Tinybird, and need to send data extracts to your clients on a regular basis.
- You want to export new records to Amazon S3 every day, so you can load them into Snowflake to run ML recommendation jobs.
- You need to share the data you have in Tinybird with other systems in your organization, in bulk.

> The Tinybird S3 Sink feature is available for Professional and Enterprise plans. If you are on a Build plan but want to access this feature, you can upgrade to Professional directly from your account Settings, or contact us at support@tinybird.co.

About Tinybird's S3 Sink

How it works

Tinybird's S3 Sink is fully managed and requires no additional tooling. You create a new connection to an Amazon S3 bucket, then choose a Pipe whose result gets written to Amazon S3. Tinybird provides you with complete observability and control over the executions, resulting files, their size, data transfer, and more.

Why S3?

Amazon S3 is very commonly used in data. Almost every service offers a way to import data from Amazon S3 (or S3-compatible storage). This Sink Connector enables you to use Tinybird's analytical capabilities to transform your data and provide it for onward use via Amazon S3 files.

Sink Pipes

The Sink connector is built on Tinybird's Sink Pipes, an extension of the Pipes concept, simila Copy Pipes. Sink Pipes allow you to capture the result of a Pipe at a moment in time, and store

the output. Currently, Amazon S3 is the only service Tinybird's Sink Pipes support. Sink Pipes can be run on a schedule, or executed on demand.

Supported regions

The Tinybird S3 Sink feature only supports exporting data to the following AWS regions:

- `us-east-*`
- `us-west-*`
- `eu-central-*`
- `eu-west-*`
- `eu-south-*`
- `eu-north-*`

Prerequisites

To use the Tinybird S3 Sink feature, you should be familiar with Amazon S3 buckets and have the necessary permissions to set up a new policy and role in AWS.

Scheduling considerations

The schedule applied to a Sink Pipe doesn't guarantee that the underlying job executes immediately at the configured time. The job is placed into a job queue when the configured time elapses. It is possible that, if the queue is busy, the job could be delayed and executed after the scheduled time.

To reduce the chances of a busy queue affecting your Sink Pipe execution schedule, we recommend distributing the jobs over a wider period of time rather than grouped close together.

For Enterprise customers, these settings can be customized. Reach out to your Customer Success team or email us at support@tinybird.co.

Set up

The setup process involves configuring both Tinybird and AWS:

1. Create your Pipe and promote it to Sink Pipe
2. Create the AWS S3 connection
3. Choose the scheduling options
4. Configure destination path and file names
5. Preview and trigger your new Sink Pipe

Using the UI

Step 1: Create a Pipe and promote it to Sink Pipe

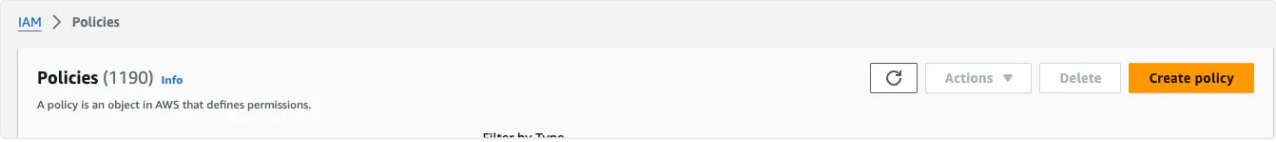
In the Tinybird UI, create a Pipe and write the query that produces the result you want to export. In the top right "Create API Endpoint" menu, select "Create Sink". In the modal, choose the destination (Amazon S3), and enter the bucket name and region.

Follow the step-by-step process on the modal to guide you through the AWS setup steps, or use the docs below.

Step 2: Create the AWS S3 Connection

Step 2.1: Create the S3 access policy

First, create an IAM policy that grants the IAM role permissions to write to S3. Open the AWS console and navigate to IAM > Policies, then select “Create Policy”:



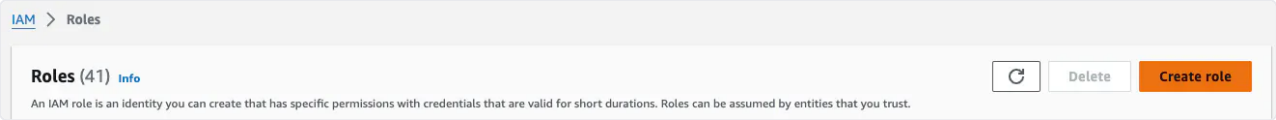
On the “Specify Permissions” screen, select “JSON” and paste the policy generated in the UI by clicking on the Copy icon. It'll look like something like this:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": "arn:aws:s3:::<bucket>/<prefix>/*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
```

Select “Next”, add a memorable name in the following dialog box (you’ll need it later!), and select “Create Policy”.

Step 2.2: Create the IAM role

In the AWS console, navigate to IAM > Roles and select “Create Role”:



On the “Select trusted entity” dialog box, select the "Custom trust policy” option. Copy the generated JSON and paste in into the Tinybird UI modal. Select "Next".

On the “Add permissions” screen, find the policy for S3 access you just created and tick the checkbox to the left of it. Select "Next" and give it a meaningful name and description. Confirm that the trusted entities and permissions granted are the expected ones, and select "Create Role".

> You’ll need the role’s ARN (Amazon Resource Name) in order to create the connection in the next step. To save you having to come back and look for it, go to IAM > Roles and browse the search box for the role you just created. Select it to open more role details, including the role's ARN. Copy it down somewhere you can find it easily again. It'll look like something like `arn:aws:iam::111111111111:role/my-awesome-role`.

Return to Tinybird's UI and enter the role ARN and Connection name in the modal. The Connection to AWS S3 is now created in Tinybird, and can be reused in multiple Sinks. 🎉 !

Step 3: Choose the scheduling options

You can configure your Sink to run "on demand" 9meaning you'll need to manually trigger it) or using a cron expression, so it runs automatically when needed.

Step 4: Configure destination path and file names

Enter the bucket URI where files will be generated (you can use subfolders), and the file name template. When generating multiple files, the Sink creates them using this template. You have multiple ways to configure this - see the File template section below.

Step 5: Preview and create

The final step is to check and confirm that the preview matches what you expect.

🎉 Congratulations! You've created your first Sink.

Trigger it manually using the "Run Sink now" option in the top right menu, or wait for the next scheduled execution.

> When triggering a Sink Pipe you have the option of overriding several of its settings, like format or compression. Refer to the Sink Pipes API spec for the full list of parameters.

Once the Sink Pipe is triggered, it creates a standard Tinybird job that can be followed via the `v0/jobs` API.

Using the CLI

Step 1: Create the AWS S3 Connection

To create a connection for an S3 Sink Pipe you need to use a CLI version equal to or higher than 3.5.0.

To start:

1. Run the `tb connection create s3_iamrole` command.
2. Copy the suggested policy and replace the two bucket placeholders with your bucket name.
3. Log into your AWS Console.
4. Create a new policy in AWS IAM > Policies using the copied text.

In the next step, you'll need the role's ARN (Amazon Resource Name) to create the connection. Go to IAM > Roles and browse the search box for the role you just created. Select it to open more role details, including the role's ARN. Copy it and paste it into the CLI when requested. It'll look like something like `arn:aws:iam::111111111111:role/my-awesome-role`.

Then, you will need to type the region where the bucket is located and choose a name to identify your connection within Tinybird. Once you have completed all these inputs, Tinybird will check access to the bucket and create the connection with the connection name you selected.

Step 2: Create S3 Sink Pipe

To create a Sink Pipe, create a regular .pipe and filter the data you want to export to your bucket in the SQL section as in any other Pipe. Then, specify the Pipe as a sink type and add the needed configuration. Your Pipe should have the following structure:

```
NODE node_0

SQL >
    SELECT *
    FROM events
    WHERE time >= toStartOfMinute(now()) - interval 30 minute)
```

```
TYPE sink
EXPORT_SERVICE s3_iamrole
EXPORT_CONNECTION_NAME "test_s3"
EXPORT_BUCKET_URI "s3://tinybird-sinks"
EXPORT_FILE_TEMPLATE "daily_prices" # Supports partitioning
EXPORT_SCHEDULE "*/5 * * * *" # Optional
EXPORT_FORMAT "csv"
EXPORT_COMPRESSION "gz" # Optional
```

Sink Pipe parameters

KEY	TYPE	DESCRIPTION
EXPORT_CONNECTION_NAME	string	Required. The connection name to the destination service. This the connection created in Step 1.
EXPORT_BUCKET_URI	string	Required. The path to the destination bucket. Example: `s3://tinybird-export`
EXPORT_FILE_TEMPLATE	string	Required. The target file name. Can use parameters to dynamically name and partition the files. See File partitioning section below. Example: `daily_prices_{customer_id}`
EXPORT_FORMAT	string	Optional. The output format of the file. Values: CSV, NDJSON, Parquet. Default value: CSV
EXPORT_COMPRESSION	string	Optional. Accepted values: `none`, `gz` for gzip, `br` for brotli, `xz` for LZMA, `zst` for zstd. Default: `none`
EXPORT_SCHEDULE	string	A crontab expression that sets the frequency of the Sink operation or the @on-demand string.

Once ready, push the datafile to your Workspace using `tb push` (or `tb deploy` if you are using version control) to create the Sink Pipe.

File template

The export process allows you to partition the result in different files, allowing you to organize your data and get smaller files. The partitioning is defined in the file template and based on the

values of columns of the result set.

Partition by column

Add a template variable like `{COLUMN_NAME}` to the file name. For instance, let's set the file template as `invoice_summary_{customer_id}.csv`.

Imagine your query schema and result for an export is like this:

CUSTOMER_ID	INVOICE_ID	AMOUNT
ACME	INV20230608	23.45
ACME	12345INV	12.3
GLOBEX	INV-ABC-789	35.34
OSCORP	INVOICE2023-06-08	57
ACME	INV-XYZ-98765	23.16
OSCORP	INV210608-001	62.23
GLOBEX	987INV654	36.23

With the given file template `invoice_summary_{customer_id}.csv` you'd get 3 files:

`invoice_summary_ACME.csv`

CUSTOMER_ID	INVOICE_ID	AMOUNT
ACME	INV20230608	23.45
ACME	12345INV	12.3
ACME	INV-XYZ-98765	23.16

`invoice_summary_OSCORP.csv`

CUSTOMER_ID	INVOICE_ID	AMOUNT
OSCORP	INVOICE2023-06-08	57
OSCORP	INV210608-001	62.23

`invoice_summary_GLOBEX.csv`

CUSTOMER_ID	INVOICE_ID	AMOUNT
GLOBEX	INV-ABC-789	35.34
GLOBEX	987INV654	36.23

Values format

> In the case of DateTime columns, it can be dangerous to partition just by the column. Why? Because you could end up with as many files as seconds, as they're the different values for a DateTime column. In an hour, that's potentially 3600 files.

To help partition in a sensible way, you can add a format string to the column name using the following placeholders:

PLACEHOLDER	DESCRIPTION	EXAMPLE
%Y	Year	2023
%m	Month as an integer number (01-12)	06
%d	Day of the month, zero-padded (01-31)	07
%H	Hour in 24h format (00-23)	14
%i	Minute (00-59)	45

For instance, for a result like this:

TIMESTAMP	INVOICE_ID	AMOUNT
2023-07-07 09:07:05	INV20230608	23.45
2023-07-07 09:07:01	12345INV	12.3
2023-07-07 09:06:45	INV-ABC-789	35.34
2023-07-07 09:05:35	INVOICE2023-06-08	57
2023-07-06 23:14:05	INV-XYZ-98765	23.16
2023-07-06 23:14:02	INV210608-001	62.23
2023-07-06 23:10:55	987INV654	36.23

Note that all 7 events have different times in the column timestamp. Using a file template like `invoices_{timestamp}` would create 7 different files.

If you were interested in writing one file per hour, you could use a file template like `invoices_{timestamp, '%Y%m%d-%H'}`. You'd then get only two files for that dataset:

`invoices_20230707-09.csv`

TIMESTAMP	INVOICE_ID	AMOUNT
2023-07-07 09:07:05	INV20230608	23.45
2023-07-07 09:07:01	12345INV	12.3
2023-07-07 09:06:45	INV-ABC-789	35.34
2023-07-07 09:05:35	INVOICE2023-06-08	57

`invoices_20230706-23.csv`

TIMESTAMP	INVOICE_ID	AMOUNT
2023-07-06 23:14:05	INV-XYZ-98765	23.16
2023-07-06 23:14:02	INV210608-001	62.23
2023-07-06 23:10:55	987INV654	36.23

By number of files

You also have the option to write the result into X files. Instead of using a column name, use an integer between brackets.

Example: `invoice_summary.{8}.csv`

This is convenient to reduce the file size of the result, especially when the files are meant to be consumed by other services, like Snowflake where uploading big files is discouraged.

The results are written in random order. This means that the final result rows would be written in X files, but you can't count the specific order of the result.

There are a maximum of 16 files.

Combining different partitions

It's possible to add more than one partitioning parameter in the file template. This is useful, for instance, when you do a daily dump of data, but want to export one file per hour.

Setting the file template as `invoices/dt={timestamp, '%Y-%m-%d'}/H{timestamp, '%H}.csv` would create the following file structure in different days and executions:

```
Invoices
├── dt=2023-07-07
│   └── H23.csv
│   └── H22.csv
│   └── H21.csv
│   └── ...
├── dt=2023-07-06
│   └── H23.csv
│   └── H22.csv
```

You can also mix column names and number of files. For instance, setting the file template as `invoices/{customer_id}/dump_{4}.csv` would create the following file structure in different days and executions:

```
Invoices
├── ACME
│   └── dump_0.csv
│   └── dump_1.csv
│   └── dump_2.csv
│   └── dump_3.csv
├── OSCORP
│   └── dump_0.csv
│   └── dump_1.csv
│   └── dump_2.csv
│   └── dump_3.csv
```

> Be careful with excessive partitioning. Take into consideration that the write process will create as many files as combinations of the values of the partitioning columns for a given result set.

Iterating a Sink Pipe

Sink Pipes can be iterated using version control just like any other resource in your Data Project. However, you need to understand how connections work in Branches and deployments in order to select the appropriate strategy for your desired changes.

Branches don't execute on creation by default recurrent jobs, like the scheduled ones for Sink Pipes (they continue executing in your Release as usual).

To iterate a Sink Pipe, create a new one (or recreate the existing one) with the desired configuration. The new Sink Pipe will start executing from the Branch too (without affecting the unchanged production resource). It will use the new configuration and export the new files into a **branch-specific folder** `my_bucket/<prefix>/branch_<branch_id>/` (the `<prefix>` is optional). This means you can test the changes without mixing the test resource with your production exports.

When you deploy that Branch, the specific folder in the path is automatically ignored and production continues to point to `my_bucket/prefix/` or the new path you changed.

To deploy configuration changes like the ones described above, use an alter strategy (format 0.0.1-1) to avoid having the same Sink Pipe exporting to production from different Releases if it's not needed.

Take into account that, for now, while you can change the Sink Pipe configuration using version control, new connections to S3 must be created directly in the Live Release.

There is an example about how to create a Pipe Sink to S3 with version control here.

Observability

Sink Pipes operations are logged in the tinybird.sinks_ops_log Service Data Source.

Data Transfer incurred by Sink Pipes is tracked in tinybird.data_transfer Service Data Source.

Limits & quotas

Limits

Sink Pipes have the following limits, depending on your billing plan:

PLAN	SINK PIPES PER WORKSPACE	EXECUTION TIME	FREQUENCY	MEMORY US QUERY
Pro	3	30 s	Up to every 10 min	10 GB
Enterprise	10	300 s	Up to every minute	10 GB

Billing

The 3 different Tinybird plans (Build, Pro, Enterprise) are explained in the Billing docs.

Tinybird uses two metrics for billing Sink Pipes: Processed Data and Data Transfer. A Sink Pipe executes the Pipe's query (Processed Data), and writes the result into a Bucket (Data Transfer). If the resulting files are compressed, Tinybird accounts for the compressed size.

Processed Data

Any Processed Data incurred by a Sink Pipe is charged at the standard rate for your account. The Processed Data is already included in your plan, and counts towards your commitment. If you're on an Enterprise plan, view your plan and commitment on the Organizations tab in the UI.

Data Transfer

Data Transfer depends on your environment. There are two scenarios:

- The destination bucket is in the **same** cloud provider and region as your Tinybird workspace: $0.01 / GB
- The destination bucket is in a **different** cloud provider or region as your Tinybird workspace: $0.10 / GB

Enterprise customers

We're including 50 GB for free for every Enterprise customer, so you can test the feature and validate your use case. After that, we're happy to set up a meeting to understand your use case and adjust your contract accordingly, to accommodate the necessary Data Transfer.

Next steps

- Get familiar with the Service Data Source and see what's going on in your account
- Deep dive on Tinybird's Pipes concept