

Snowflake #6736

Table of Content

[Table of Content](#)

[Sources](#)

[Failed task in Schema - Failed](#)

[With Task Name in text - Failed](#)

[With Task Name in regex - Failed](#)

Sources

Failed task in Schema - Failed

With Task Name in text - Failed

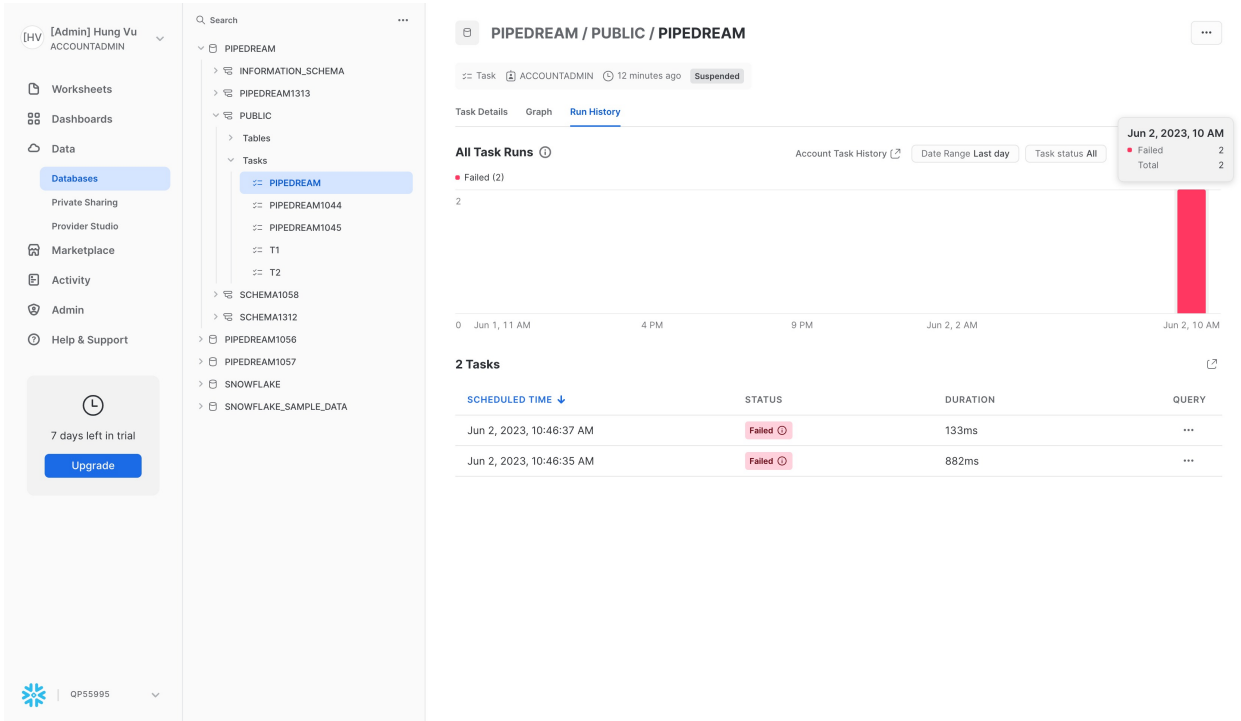
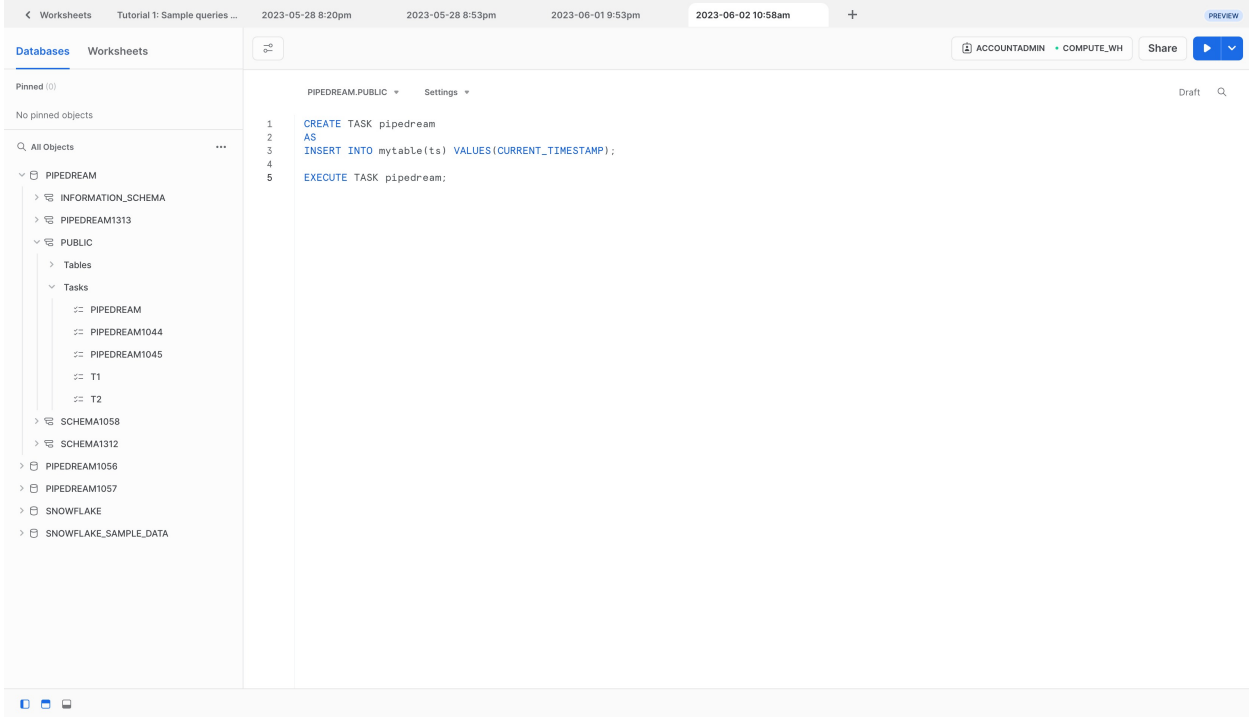
On Pipedream, I created a task with the `Task Name = pipedream`

The screenshot displays the Pipedream interface for a component named "Failed Task in Schema". The component is configured to run on a timer and emit events when a task fails in a database schema. The configuration details are as follows:

- Account:** Snowflake (Snowflake #1)
- Timer:** Every 15 minutes
- Database:** PIPEDREAM
- Schema:** PUBLIC
- Task Name:** pipedream

The interface also shows a "RUN NOW" button and a section for integrations, including workflows and APIs. The workflow section includes a "CREATE WORKFLOW" button and a "View Tutorial (1 minute)" link. The API section lists "REST" and "SSE" options.

Then on Snowflake, I created a task Pipedream and execute that task, I see that it failed 2 times



But on Pipedream source, I see no event emitted

The screenshot displays the Pipedream interface for a component named "Failed Task in Schema". The component is configured to run on a timer every 15 minutes. The configuration details are as follows:

- Account:** Snowflake (Snowflake #1)
- Timer:** Every 15 minutes
- Database:** PIPEDREAM
- Schema:** PUBLIC
- Task Name:** pipedream

The interface shows 0 events over the last 24 hours and a "Waiting for event..." message. The "INTEGRATIONS" section includes "Workflows" and "APIs".

Source log

```

2023-06-02T10:59:29 end
▶ 2023-06-02T10:59:29 db.set db["lastMaxTimestamp"] ⇐ 1685678368994
▶ 2023-06-02T10:59:29 log Raw results: []
▶ 2023-06-02T10:59:23 log Max ts of last run: 1685678175616
▶ 2023-06-02T10:59:23 db.get db["lastMaxTimestamp"] ⇒ 1685678175616
▶ 2023-06-02T10:59:23 start { "timestamp": 1685678362, "timezone_utc": { "date": { "day": 2, "month": 6, "year": 2023 }, "iso8601": { "date": "2023-06-02", "time": "03...
2023-06-02T10:56:16 end
▶ 2023-06-02T10:56:16 db.set db["lastMaxTimestamp"] ⇐ 1685678175616
▶ 2023-06-02T10:56:16 log Raw results: []
▶ 2023-06-02T10:56:09 log Max ts of last run: 1685678089333
▶ 2023-06-02T10:56:09 db.get db["lastMaxTimestamp"] ⇒ 1685678089333
▶ 2023-06-02T10:56:09 start { "timestamp": 1685678169 }
2023-06-02T10:54:49 end
▶ 2023-06-02T10:54:49 db.set db["lastMaxTimestamp"] ⇐ 1685678089333
▶ 2023-06-02T10:54:49 log Raw results: []
▶ 2023-06-02T10:54:39 log Max ts of last run: 1685677676619
▶ 2023-06-02T10:54:39 db.get db["lastMaxTimestamp"] ⇒ 1685677676619
▶ 2023-06-02T10:54:39 start { "timestamp": 1685678067 }
2023-06-02T10:47:57 end
▶ 2023-06-02T10:47:57 db.set db["lastMaxTimestamp"] ⇐ 1685677676619
▶ 2023-06-02T10:47:57 log Raw results: []
▶ 2023-06-02T10:47:50 log Max ts of last run: 1685677662243
▶ 2023-06-02T10:47:50 db.get db["lastMaxTimestamp"] ⇒ 1685677662243
▶ 2023-06-02T10:47:50 start { "timestamp": 1685677670 }
2023-06-02T10:47:42 end
▶ 2023-06-02T10:47:42 db.set db["lastMaxTimestamp"] ⇐ 1685677662243
▶ 2023-06-02T10:47:42 log Raw results: []
▶ 2023-06-02T10:47:37 log Max ts of last run: 1685677652701
▶ 2023-06-02T10:47:37 db.get db["lastMaxTimestamp"] ⇒ 1685677652701
▶ 2023-06-02T10:47:37 start { "timestamp": 1685677656 }
2023-06-02T10:47:33 end
▶ 2023-06-02T10:47:33 db.set db["lastMaxTimestamp"] ⇐ 1685677652701
▶ 2023-06-02T10:47:33 log Raw results: []
▶ 2023-06-02T10:47:27 log Max ts of last run: 1685677569571
▶ 2023-06-02T10:47:27 db.get db["lastMaxTimestamp"] ⇒ 1685677569571
▶ 2023-06-02T10:47:27 start { "timestamp": 1685677646 }
2023-06-02T10:46:10 end
▶ 2023-06-02T10:46:10 db.set db["lastMaxTimestamp"] ⇐ 1685677569571
▶ 2023-06-02T10:46:10 log Raw results: []
▶ 2023-06-02T10:46:02 log Max ts of last run: 1685677556119
▶ 2023-06-02T10:46:02 db.get db["lastMaxTimestamp"] ⇒ 1685677556119
▶ 2023-06-02T10:46:02 start { "timestamp": 1685677561 }
2023-06-02T10:45:56 end
▶ 2023-06-02T10:45:56 db.set db["lastMaxTimestamp"] ⇐ 1685677556119
▶ 2023-06-02T10:45:56 log Raw results: []
▶ 2023-06-02T10:45:50 log Max ts of last run: 1685677504780
▶ 2023-06-02T10:45:50 db.get db["lastMaxTimestamp"] ⇒ 1685677504780
▶ 2023-06-02T10:45:50 start { "timestamp": 1685677549 }
2023-06-02T10:45:05 end
▶ 2023-06-02T10:45:05 db.set db["lastMaxTimestamp"] ⇐ 1685677504780
▶ 2023-06-02T10:45:05 log Raw results: []
▶ 2023-06-02T10:44:59 log Max ts of last run: 1685677481314
▶ 2023-06-02T10:44:59 db.get db["lastMaxTimestamp"] ⇒ 1685677481314
▶ 2023-06-02T10:44:59 start { "timestamp": 1685677497 }
2023-06-02T10:44:41 end
▶ 2023-06-02T10:44:41 db.set db["lastMaxTimestamp"] ⇐ 1685677481314
▶ 2023-06-02T10:44:41 log Raw results: []
▶ 2023-06-02T10:44:29 log Max ts of last run: 1685591069290
▶ 2023-06-02T10:44:29 db.get db["lastMaxTimestamp"] ⇒ null
▶ 2023-06-02T10:44:29 start { "timestamp": 1685677460, "timezone_utc": { "date": { "day": 2, "month": 6, "year": 2023 }, "iso8601": { "date": "2023-06-02", "time": "03...

```

Expected Behavior:

- There should be new event emitted for new task failed event matched the Task name `pipedream`

With Task Name in regex - Failed

On Pipedream, I created a task with the `Task Name = pipedream.*` which I expected to match `pipedream`, `pipedream1044`, `pipedream1045`

The screenshot shows the Pipedream interface for a workflow titled "Failed Task in Schema". The workflow is configured to emit new events when a task fails in a database schema. The configuration details are as follows:

- Account:** Snowflake (Snowflake #1)
- Timer:** Every 15 minutes
- Database:** PIPEDREAM
- Schema:** PUBLIC
- Task Name:** pipedream.*

The workflow is currently in a "Waiting for event..." state. The interface also shows a sidebar with navigation options like Workflows, Sources, Accounts, Data Stores, Components, Settings, Explore, Community, Help & Docs, and Careers. At the bottom, there is a "DELETE ALL EVENTS" button.

On snowflake, I created 2 tasks and execute them

Worksheets Tutorial 1: Sample queries... 2023-05-28 8:20pm 2023-05-28 8:53pm 2023-06-01 9:53pm 2023-06-02 10:58am 2023-06-02 11:04am PREVIEW

Databases Worksheets

ACCOUNTADMIN COMPUTE_WH Share

PIPEDREAM.PUBLIC Settings Latest Version

```

1 CREATE TASK pipedream1044
2 AS
3 INSERT INTO mytable(ts) VALUES (CURRENT_TIMESTAMP);
4 ---
5 CREATE TASK pipedream1045
6 AS
7 INSERT INTO mytable(ts) VALUES (CURRENT_TIMESTAMP);
8 ---
9 CREATE TASK pipedream
10 AS
11 INSERT INTO mytable(ts) VALUES (CURRENT_TIMESTAMP);
12
13 EXECUTE TASK pipedream;
14 EXECUTE TASK pipedream1044;
15 EXECUTE TASK pipedream1045;
16

```

Results Chart

status
1 Task PIPEDREAM1045 is scheduled to run immediately.

Query Details

- Query duration: 83ms
- Rows: 1
- Query ID: 01acb0b4-3200-bf72-Q...
- status: 100% filled

I see that they all have failed tasks

[Admin] Hung Vu ACCOUNTADMIN

Worksheets Dashboards Data Databases Private Sharing Provider Studio Marketplace Activity Admin Help & Support

7 days left in trial Upgrade

QP55995

PIPEDREAM / PUBLIC / PIPEDREAM

Task ACCOUNTADMIN 19 minutes ago Suspended

Task Details Graph Run History

Account Task History Date Range Last day Task status All

All Task Runs

Failed (3)

2

Jun 2, 2023, 10 AM

SCHEDULED TIME	STATUS	DURATION	QUERY
Jun 2, 2023, 11:04:41 AM	Failed	1.1s	...
Jun 2, 2023, 10:46:37 AM	Failed	133ms	...
Jun 2, 2023, 10:46:35 AM	Failed	882ms	...

[HV] [Admin] Hung Vu
ACCOUNTADMIN

- Worksheets
- Dashboards
- Data
 - Databases**
 - Private Sharing
 - Provider Studio
- Marketplace
- Activity
- Admin
- Help & Support

7 days left in trial
[Upgrade](#)

QP55995

Search

- PIPEDREAM
 - INFORMATION_SCHEMA
 - PIPEDREAM1313
 - PUBLIC
 - Tables
 - Tasks
 - PIPEDREAM
 - PIPEDREAM1044**
 - PIPEDREAM1045
 - T1
 - T2
 - SCHEMA1058
 - SCHEMA1312
 - PIPEDREAM1056
 - PIPEDREAM1057
 - SNOWFLAKE
 - SNOWFLAKE_SAMPLE_DATA

PIPEDREAM / PUBLIC / PIPEDREAM1044

Task ACCOUNTADMIN 19 minutes ago **Suspended**

Task Details Graph **Run History**

Account Task History [Date Range Last day](#) Task status All

All Task Runs ⓘ

Failed (3)

3 Tasks

SCHEDULED TIME ↓	STATUS	DURATION	QUERY
Jun 2, 2023, 11:04:42 AM	Failed ⓘ	584ms	...
Jun 2, 2023, 10:47:45 AM	Failed ⓘ	437ms	...
Jun 2, 2023, 10:45:18 AM	Failed ⓘ	618ms	...

[HV] [Admin] Hung Vu
ACCOUNTADMIN

- Worksheets
- Dashboards
- Data
 - Databases**
 - Private Sharing
 - Provider Studio
- Marketplace
- Activity
- Admin
- Help & Support

7 days left in trial
[Upgrade](#)

QP55995

Search

- PIPEDREAM
 - INFORMATION_SCHEMA
 - PIPEDREAM1313
 - PUBLIC
 - Tables
 - Tasks
 - PIPEDREAM
 - PIPEDREAM1044
 - PIPEDREAM1045**
 - T1
 - T2
 - SCHEMA1058
 - SCHEMA1312
 - PIPEDREAM1056
 - PIPEDREAM1057
 - SNOWFLAKE
 - SNOWFLAKE_SAMPLE_DATA

PIPEDREAM / PUBLIC / PIPEDREAM1045

Task ACCOUNTADMIN 20 minutes ago **Suspended**

Task Details Graph **Run History**

Account Task History [Date Range Last day](#) Task status All

All Task Runs ⓘ

Failed (3)

3 Tasks

SCHEDULED TIME ↓	STATUS	DURATION	QUERY
Jun 2, 2023, 11:04:43 AM	Failed ⓘ	403ms	...
Jun 2, 2023, 10:47:47 AM	Failed ⓘ	362ms	...
Jun 2, 2023, 10:45:19 AM	Failed ⓘ	446ms	...

But I don't see new event emitted

vunguyenh...
Failed Task in Schema
name slug: failed-task-in-schema-3 • created: 15 seconds ago

EVENTS
LOGS
CONFIGURATION

0 over last 24 hours

Waiting for event...

SNOWFLAKE: FAILED TASK IN SCHEMA
Emit new events when a task fails in a database schema

This component runs on a timer

Account: Snowflake (Snowflake #1)
Timer: Every 15 minutes
Database: PIPEDREAM
Schema: PUBLIC
Task Name: pipedream.*

[Edit code and configuration](#)

RUN NOW

INTEGRATIONS

Workflows

CREATE WORKFLOW
View Tutorial (1 minute)

- Trigger serverless workflows on each event
- Write Node.js and use any npm package, or use community-built actions
- Use OAuth tokens and API keys with managed auth for 200+ apps

APIs

Manage sources and consume event data in your app using [REST](#) or [SSE](#) (real-time) APIs.

[▶ REST](#)
[▶ SSE](#)

DELETE ALL EVENTS

Source log

2023-06-02T11:06:17	end
▶ 2023-06-02T11:06:17	db.set db["lastMaxTimestamp"] ⇐ 168567877487
▶ 2023-06-02T11:06:17	log Raw results: []
▶ 2023-06-02T11:06:07	log Max ts of last run: 1685678577727
▶ 2023-06-02T11:06:07	db.get db["lastMaxTimestamp"] ⇒ 1685678577727
▶ 2023-06-02T11:06:07	start { "timestamp": 1685678755 }
2023-06-02T11:02:58	end
▶ 2023-06-02T11:02:58	db.set db["lastMaxTimestamp"] ⇐ 1685678577727
▶ 2023-06-02T11:02:58	log Raw results: []
▶ 2023-06-02T11:02:48	log Max ts of last run: 1685677690934
▶ 2023-06-02T11:02:48	db.get db["lastMaxTimestamp"] ⇒ 1685677690934
▶ 2023-06-02T11:02:48	start { "timestamp": 1685678556, "timezone_utc": { "date": { "day": 2, "month": 6, "year": 2023 }, "iso8601": { "date": "2023-06-02", "time": "04..."
2023-06-02T10:48:11	end
▶ 2023-06-02T10:48:11	db.set db["lastMaxTimestamp"] ⇐ 1685677690934
▶ 2023-06-02T10:48:11	log Raw results: []
▶ 2023-06-02T10:48:01	log Max ts of last run: 1685591280656
▶ 2023-06-02T10:48:01	db.get db["lastMaxTimestamp"] ⇒ null
▶ 2023-06-02T10:48:01	start { "timestamp": 1685677669 }
2023-06-02T10:48:01	end
▶ 2023-06-02T10:48:01	db.set db["lastMaxTimestamp"] ⇐ 1685677681216
▶ 2023-06-02T10:48:01	log Raw results: []
▶ 2023-06-02T10:47:51	log Max ts of last run: 1685591271254
▶ 2023-06-02T10:47:51	db.get db["lastMaxTimestamp"] ⇒ null
▶ 2023-06-02T10:47:51	start { "timestamp": 1685677660 }
2023-06-02T10:47:52	end
▶ 2023-06-02T10:47:52	db.set db["lastMaxTimestamp"] ⇐ 1685677672074
▶ 2023-06-02T10:47:52	log Raw results: []
▶ 2023-06-02T10:47:42	log Max ts of last run: 1685591262073
▶ 2023-06-02T10:47:42	db.get db["lastMaxTimestamp"] ⇒ null
▶ 2023-06-02T10:47:42	start { "timestamp": 1685677652, "timezone_utc": { "date": { "day": 2, "month": 6, "year": 2023 }, "iso8601": { "date": "2023-06-02", "time": "03..."

Expected behavior:

- There should be new event emitted for new task failed event matched the Task name `pipedream`, `pipedream1044`, `pipedream1045`
- We should add example to the prop `Task Name` to instruct user how to input the regex correctly