

Flow in VantageCloud Lake: Cookbook to Make it Work in AWS

I wrote this document in February 2024. At the time, Teradata had online documentation about [Flow](#). It was the first I was going to use the service, I needed to [quickly upload a file into a database](#) and I got stuck. So, I noted everything I did so I could repeat my steps. Here you have my lessons learnt.

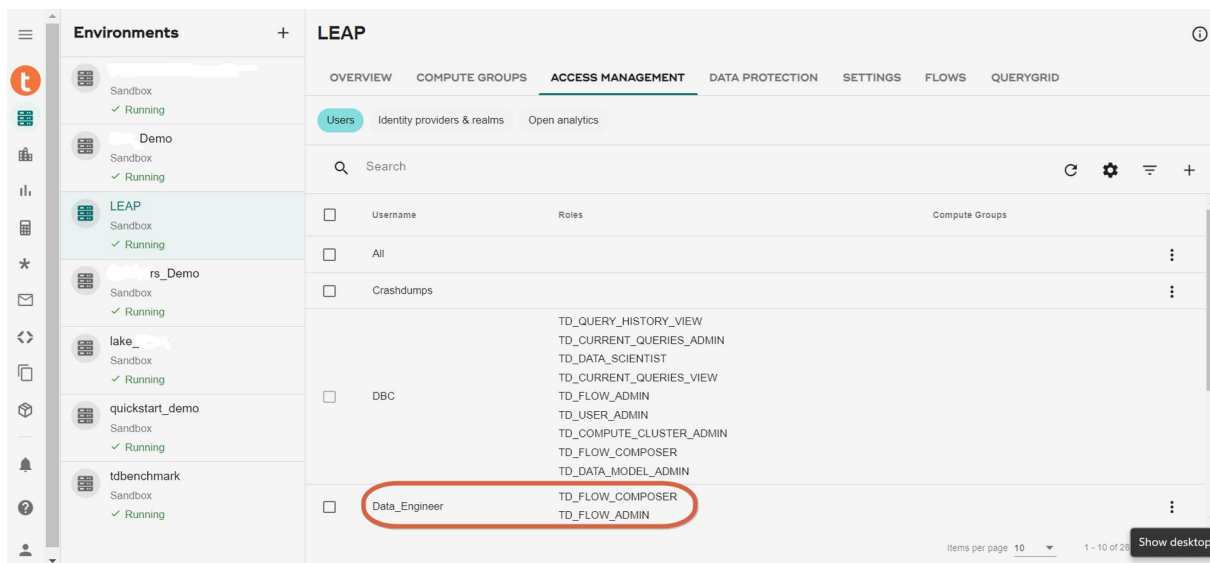
Table of Contents

Set up the Flow database user – One-time task.....	1
Grant permissions to Lake on the AWS S3 bucket where I keep my files – One time task.....	2
AWS Account ID where my AWS S3 bucket is.....	3
AWS Account ID where my Lake environment is.....	3
Create an AWS IAM role in the account where you have your bucket.....	5
Role ARN.....	10
Create a flow – Whenever you need it.....	11

Set up the Flow database user – One-time task

Flow needs a database user to run the jobs. To that end, I created the user Data_Engineer in my environment and assigned the TD_FLOW_COMPOSER and TD_FLOW_ADMIN roles to it.





Additionally, I must grant permissions to the Flow database user (Data_Engineer, in my case) to load the target database.

Note that every flow will create its error tables within the Flow database user

```
grant SELECT, UPDATE, DELETE, INSERT, CREATE TABLE
on <target-database>1
to <flow-database-user>;
```

```
grant EXECUTE
on <flow-database-user>
to <target-database>
with grant option;
```

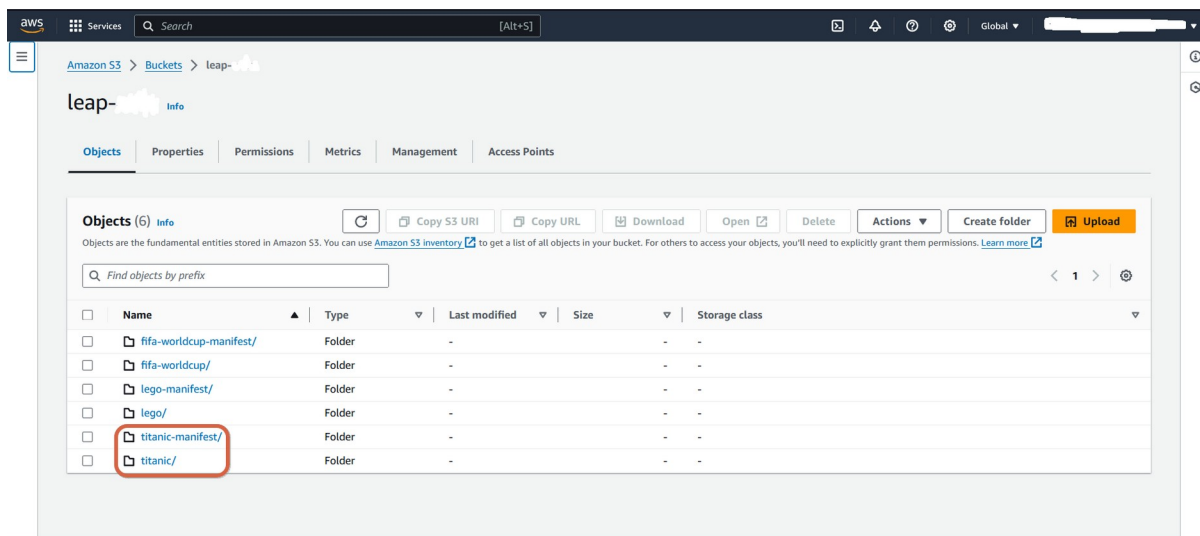
Note that every flow will create its error tables within the Flow database user, so it must have perm space. The error tables names are <flow-name>_error and <flow-name>_nos_error.

Grant permissions to Lake on the AWS S3 bucket where I keep my files – One time task

I have one bucket in AWS S3 where I have all the files I need to upload once or recurrently. I must configure the security to allow Lake to read the files in any folders when needed. I only need to configure the security in the bucket once, and then I'll keep my different workloads in separate folders. I used the titanic folder and its content to write this document. Note: I have also created a separate manifest folder to store a file Flow uses to select the files to read.

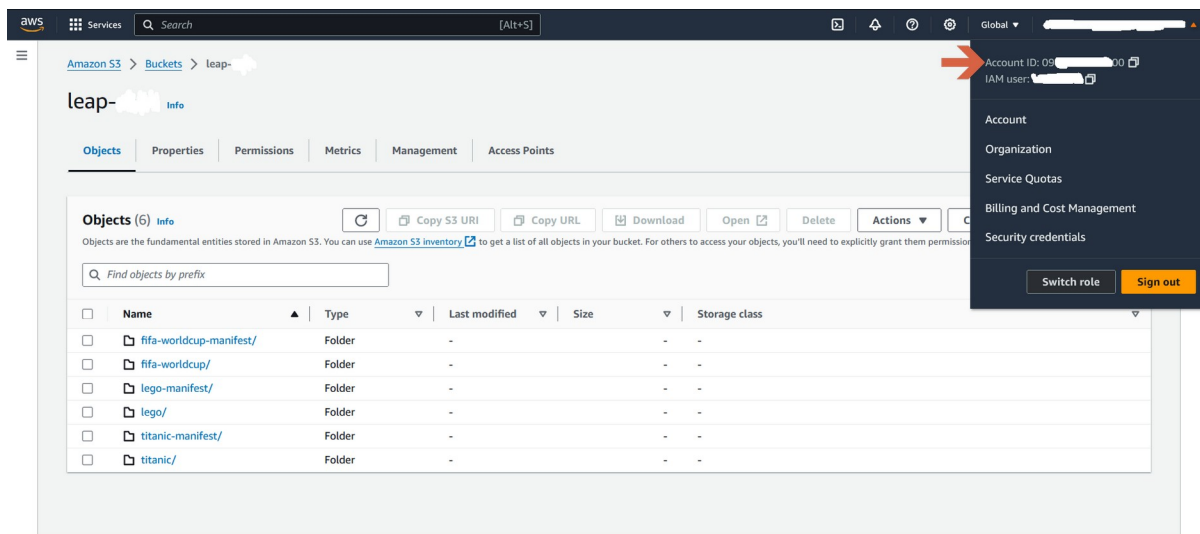
¹ In this document, all code is in Courier New. When highlighted in green, you should replace it with the appropriate value for your case.





AWS Account ID where my AWS S3 bucket is

Before granting permissions to Lake on the AWS S3 bucket, I need to annotate and keep handy the AWS Account ID of the account where I keep my AWS S3 bucket. See below where you can find it.



AWS Account ID where my Lake environment is

I also need to know the AWS Account ID of my Lake environment. You can find it in the "Create Flow" screen.



Environments

LEAP

Sandbox

✓ Running

LEAP

OVERVIEW

FLOWSS

↺

≡

+

Name	Status	User	Last action (UTC)	Last commit (UTC) ↓	
titanic8	✓ Completed	DATA_ENGINEER	Create by DATA_ENGINEER 02/27/2024 17:46	02/27/2024 17:47	⋮
titanic6	✓ Completed	DATA_ENGINEER	Create by DATA_ENGINEER 02/27/2024 17:13	02/27/2024 17:15	⋮
titanic	! Failed	DATA_ENGINEER	Delete by DATA_ENGINEER 02/27/2024 14:21		⋮
titanic1	! Failed	DATA_ENGINEER	Delete by DATA_ENGINEER 02/27/2024 14:50		⋮
titanic2	! Failed	DATA_ENGINEER	Delete by DATA_ENGINEER 02/27/2024 15:22		⋮
titanic3	! Failed	DATA_ENGINEER	Delete by DATA_ENGINEER 02/27/2024 15:23		⋮

Items per page 10 9 of 9 |< < > >|

Create flow

Enter required settings

Name *

0/27

Description

Scroll down

0/512

Load Options

Once

Source and Target Authorization

The following fields will be used to authorize against all sources. By creating this flow I am allowing Teradata to write to the target table.

[View documentation to learn how to configure an AWS role to use with flow](#)



Create flow

Enter required settings

SAVE

CREATE

The following fields will be used to authorize against all sources. By creating this flow I am allowing TeraData to write to the target table.

[View documentation to learn how to configure an AWS role to use with flow](#)

AWS account ID

027057685614

AWS role ARN *

AWS External ID *

Sources and Targets

Max sources per flow is 5

No Sources

ADD SOURCE

Create an AWS IAM role in the account where you have your bucket

The documentation explains [how to create this role](#), but I'll do it step by step in this section.

In the AWS Console, go to IAM and create a role as follows:



CELIA MURIEL

You don't need to choose any policy in the next screen. Just click on "Next".

On the following screen, name the role and click on “Create role”.



This screenshot shows the 'Name, review, and create' page in the AWS IAM console. It is divided into three main steps: Step 1: Select trusted entities, Step 2: Add permissions, and Step 3: Add tags. Step 1 is currently active, showing a 'Role details' section with a 'Role name' field (marked with a red circle 1) and a 'Role ARN' field. Below this is a 'Trust policy' section with a JSON policy document. Step 2 shows a table for 'Permissions and policy summary'. Step 3 shows an 'Add tags' section. At the bottom right, there are buttons for 'Cancel', 'Create role', and 'Save' (marked with a red circle 2).

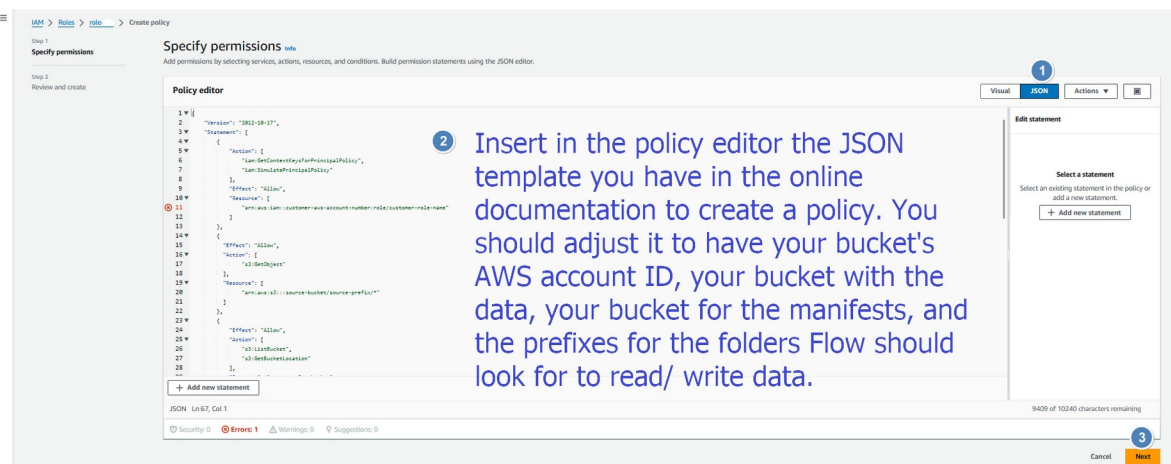
Now, go to the list of roles, search for the one you have just created, and open it.

Then in the Permissions tab, open the drop-down menu “Add permissions” and click on “Create inline policy”.

This screenshot shows the 'Role role- created.' page in the AWS IAM console. The left sidebar shows the 'Identity and Access Management (IAM)' menu with options like Dashboard, Access management, Access reports, and Related consoles. The main content area shows the 'role-' role details. The 'Permissions' tab is selected, showing a table for 'Permissions policies (0)'. A red arrow points to the 'Add permissions' button, which has a dropdown menu with 'Attach policies' and 'Create inline policy' options.

On the following screen, choose the JSON format, copy the JSON template you will find in the online documentation to [create a policy](#), and correct it with the details of your account.





In my case, I want to have one bucket with several folders. In each folder, I store the data needed for every one of my workloads. Additionally, I'll have separate folders for the manifests. So, I use the same bucket name for the data and the manifests, and I don't use prefixes but "*" to include all folders within my bucket. You have my policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "iam:GetContextKeysForPrincipalPolicy",
        "iam:SimulatePrincipalPolicy"
      ],
      "Effect": "Allow",
      "Resource": [
        "arn:aws:iam::<bucket-aws-account-id>:role/<your-
role>"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::<your-bucket>/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
```




```
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::<your-bucket>",
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "*"
          ]
        }
      },
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:DeleteObject",
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::<your-bucket>/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::<your-bucket>",
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "*"
          ]
        }
      }
    }
  ]
}
```

Name the policy and click on “Create policy”.



The screenshot shows the 'Review and create' step in the AWS IAM console. The left sidebar indicates 'Step 1: Specify permissions' and 'Step 2: Review and create'. The main content area is titled 'Review and create' and includes a 'Policy details' section with a 'Policy name' field containing 'policy-' (marked with a red circle 1). Below this is a blue informational box stating that the policy must have an action with an applicable resource or condition. The 'Permissions defined in this policy' section shows a search bar and a table of permissions. The table has columns for Service, Access level, Resource, and Request condition. It lists two permissions: one for IAM with 'Limited: Read' access level and 'RoleName string like [role-leap]' resource, and another for S3 with 'Limited: Read, List, Write' access level and 's3:prefix string like [all]' resource. At the bottom right, there are 'Cancel', 'Previous', and 'Create policy' buttons, with the 'Create policy' button marked with a red circle 2.

Back on the role screen, go the “Trusted relationships” tab and click on “Edit trust policy”.

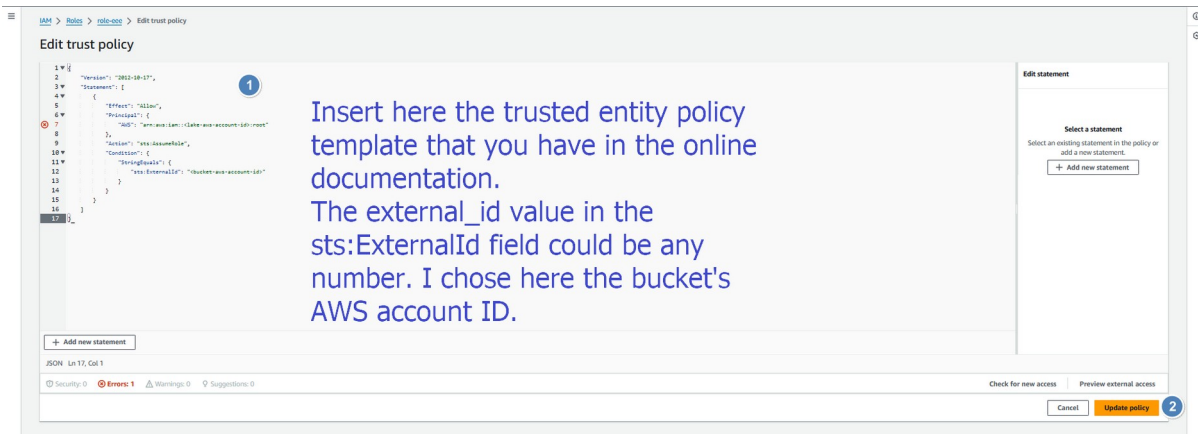
The screenshot shows the 'Trusted relationships' tab in the AWS IAM console for a role named 'role-'. The left sidebar shows the 'Identity and Access Management (IAM)' section with various navigation options. The main content area has a 'Summary' section with details like 'Creation date: February 27, 2024, 21:11 (UTC+01:00)' and 'ARN: arn:aws:iam::00role/role-eee'. Below this is the 'Trusted entities' section, which is marked with a red circle 1. It shows a list of trusted entities with a JSON policy document. The policy document is as follows:

```
1 {
2   "version": "2012-10-17",
3   "statement": [
4     {
5       "effect": "allow",
6       "principal": {
7         "arn": "arn:aws:iam::00role/role-eee"
8       },
9       "action": "sts:assumeRole",
10      "condition": {}
11    }
12  ]
13 }
```

At the bottom right of the 'Trusted entities' section, there is an 'Edit trust policy' button marked with a red circle 2.

Insert here the [trusted entity policy template](#) that you have in the online documentation.





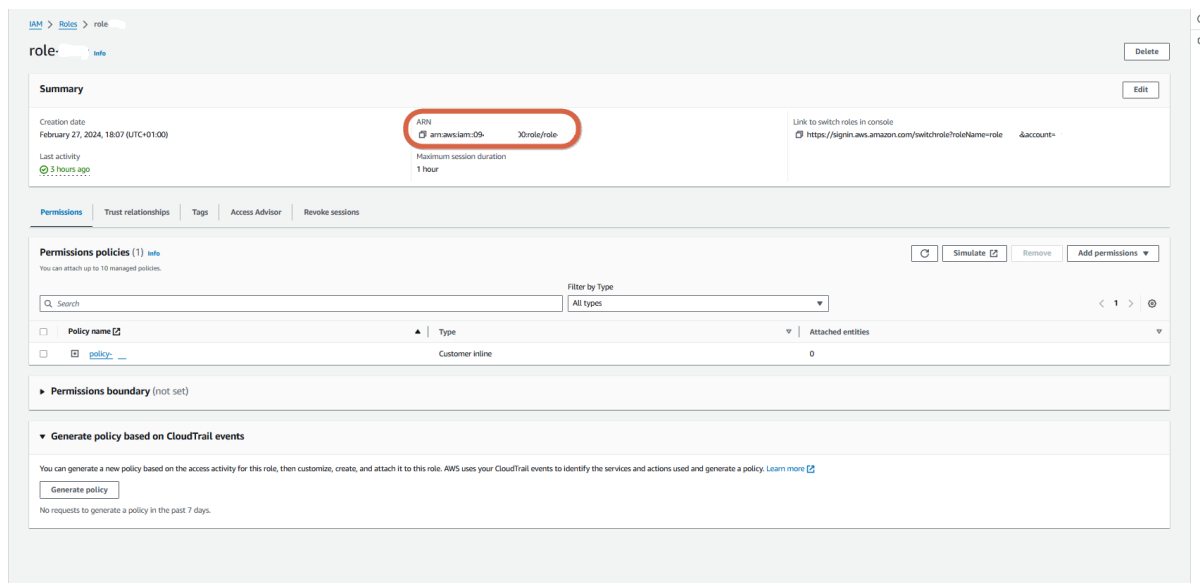
Below you have the trusted entity policy I used.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::<lake-aws-account-id>:root"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "<bucket-aws-account-id>"
        }
      }
    }
  ]
}
```

Role ARN

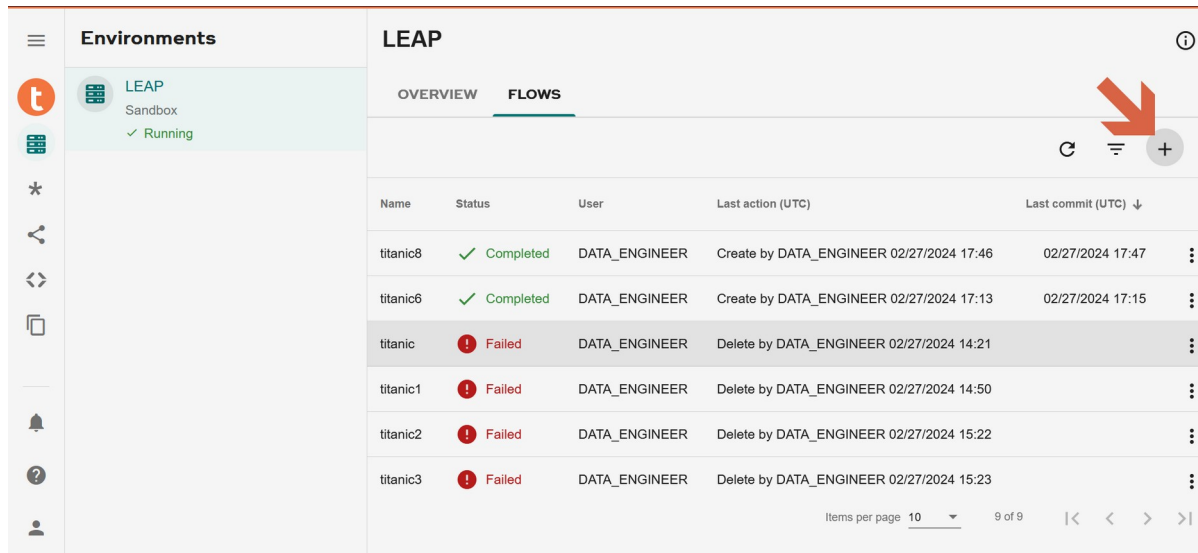
Take a note on the role ARN name you have just created, as you will need it to create flows.





Create a flow – Whenever you need it

Teradata online documentation explains in detail [how to create a flow](#) and what every field means. I'll review the process here as an example.




Create flow ×

Unsaved changes ? **SAVE** **CREATE**

Name *
titanic9 1

8/27


Description
0/512 

Load Options
Once ▼

Source and Target Authorization

The following fields will be used to authorize against all sources. By creating this flow I am allowing Teradata to write to the target table.

[View documentation to learn how to configure an AWS role to use with flow](#)

AWS account ID 027057685614 

AWS role ARN *
<your-role-ARN> 2

AWS External ID *
<bucket-aws-account-id> 3

Sources and Targets +
Max sources per flow is 5

No Sources
ADD SOURCE 4



← Source details

Enter or update settings

Foreign table name *

titanic9

1

S3 bucket path URI *

s3://leap- /titanic/

2

S3 Manifest bucket path URI *

s3://leap- /titanic-manifest/

3

Format

CSV

☒ Headers ☐ Quoted

Delimiter *

,

Compression

None

Targets

4 +

No Targets

ADD TARGET

Advanced Options

▼

Unsaved changes

SAVE

CREATE

Name *

titanic9

8/27

Description

0/512

Load Options

Once

Source and Target Authorization

The following fields will be used to authorize against all sources. By creating this flow I am allowing Teradata to write to the target table.

[View documentation to learn how to configure an AWS role to use with flow](#)

AWS account ID

027057685614

AWS role ARN *

<your-role-ARN>

AWS External ID *

<bucket-aws-account-id>

Sources and Targets

+

Max sources per flow is 5

→

titanic9

S3

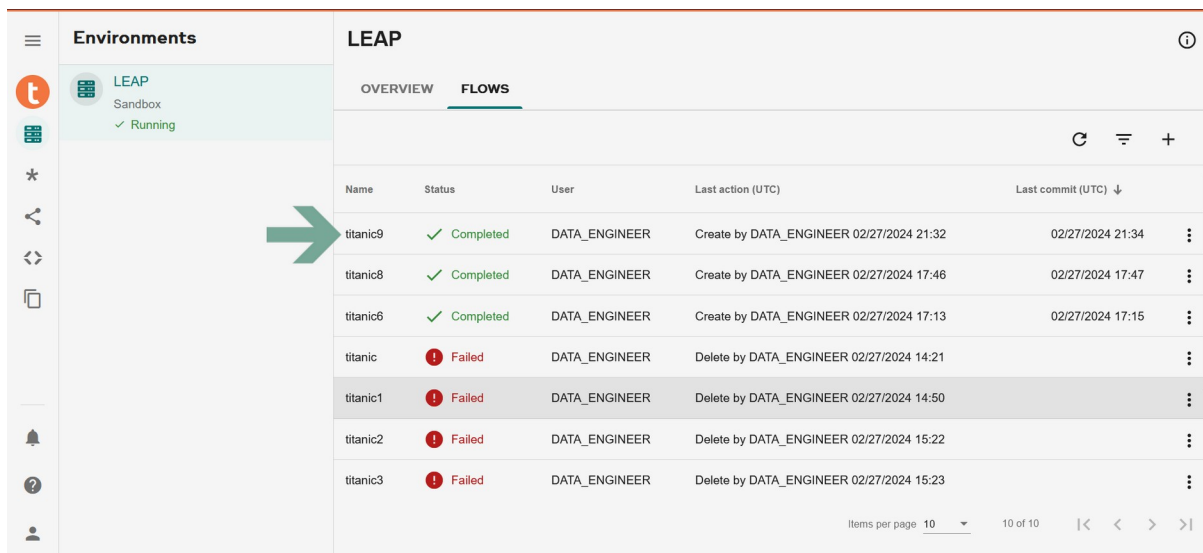
You can choose any name for the foreign table that Flow will use during the load. I used the same name of the flow to help me keep track of my tests.

Flow will load all files in this folder

3

4





Environments		LEAP				
<div>LEAP Sandbox ✓ Running</div>		OVERVIEW FLOWS				
		Name	Status	User	Last action (UTC)	Last commit (UTC) ↓
		titanic9	✓ Completed	DATA_ENGINEER	Create by DATA_ENGINEER 02/27/2024 21:32	02/27/2024 21:34
		titanic8	✓ Completed	DATA_ENGINEER	Create by DATA_ENGINEER 02/27/2024 17:46	02/27/2024 17:47
		titanic6	✓ Completed	DATA_ENGINEER	Create by DATA_ENGINEER 02/27/2024 17:13	02/27/2024 17:15
		titanic	❌ Failed	DATA_ENGINEER	Delete by DATA_ENGINEER 02/27/2024 14:21	
		titanic1	❌ Failed	DATA_ENGINEER	Delete by DATA_ENGINEER 02/27/2024 14:50	
		titanic2	❌ Failed	DATA_ENGINEER	Delete by DATA_ENGINEER 02/27/2024 15:22	
		titanic3	❌ Failed	DATA_ENGINEER	Delete by DATA_ENGINEER 02/27/2024 15:23	

Items per page 10 10 of 10



The screenshot displays the VantageCloud Lake Editor interface. On the left, a sidebar shows a tree of objects including TDMaps, TDSYSFLOW, TD_ANALYTICS_DB, TD_METRIC_SVC, TD_MLDB, TD_OFSDb, TD_OTFDB, TD_SYSFNLIB, TD_SYSXML, TD_VAL, Titanic, and passengers. The main editor area shows a SQL script in a file named 'Script-1' with the following content:

```
1 select *
2 from Titanic.passengers
```

Below the script, the 'Results' section shows a table with 6 columns: passengerid, survived, pclass, name, sex, and age. The table contains 4 rows of data:

passengerid	survived	pclass	name	sex	age
23	1	3	"McGowan, Miss. Anna ""..."	female	1
24	1	1	"Sloper, Mr. William Thom..."	male	2
25	0	3	"Palsson, Miss. Torborg D..."	female	8
26	1	3	"Asplund, Mrs. Carl Osca..."	female	3

The interface also includes a 'RUN' button, a 'LEAP Data_Engineer' dropdown, and a 'No Compute Group' dropdown. The bottom status bar shows 'Items per page: 50' and '1 - 50 of 891'.

