



Amazon Web Services Data Engineering Immersion Day

Database Migration Services Instructor Lab Setup
March 2020

Table of Contents

| | |
|--|----------|
| <i>Limit Instruction:</i> | 2 |
| <i>Introduction</i> | 2 |
| <i>Create the Instructor Environment</i> | 3 |
| <i>Changing RDS Security Group</i> | 6 |
| <i>Access Database from SQL Client (Optional)</i> | 8 |
| <i>Generate and Replicate the CDC Data (Optional)</i> | 9 |

Limit Instruction:

This immersion day required each student to have their own account. If you are sharing single account with multiple students by creating a multiple IAM users, Account can hit following default service limit:

- VPC – VPCs per Region 5
- Glue - Number of crawlers per account 50
- Glue - Number of concurrent jobs runs per account 50
- Glue - Maximum DPU's used by a role at one time 300
- S3 – Number of buckets per account 100
- Athena - Number of DDL queries you can submit at the same time 20
- Athena - Number of DML queries you can submit at the same time 20
- RDS – Make sure you have enough disk space available in your RDS instance, if want to run DMS Change Data Capture (CDC) as generating large amount of data can exhaust RDS disk space.
- DMS - Make sure you have enough disk space available in your DMS replication instance, if want to run DMS Change Data Capture (CDC) as transferring large amount of CDC data can exhaust disk space.

Introduction

*****Make sure you select the us-east-1 (Virginia) region*****

The Database Migration Services (DMS) hands-on lab provide a scenario, where participant learns to hydrate Amazon S3 data lake with a relation database. To achieve that, participants need a source endpoint and this guide helps instructors set up a PostgreSQL database with public endpoint as the source database.

In this lab, you will complete the following tasks using AWS CloudFormation template:

1. Create the source database environment.
2. Hydrate the source database environment.
3. Update the source database environment to demonstrate CDC (Change Data Capture) replication within DMS.
4. Create Lambda function to trigger CDC data which will be replicated to Amazon S3 by DMS CDC endpoint.

Relevant information about this lab:

- Expected setup time: 20 minutes
- Source database name: sportstickets
- Source schema name: dms_sample

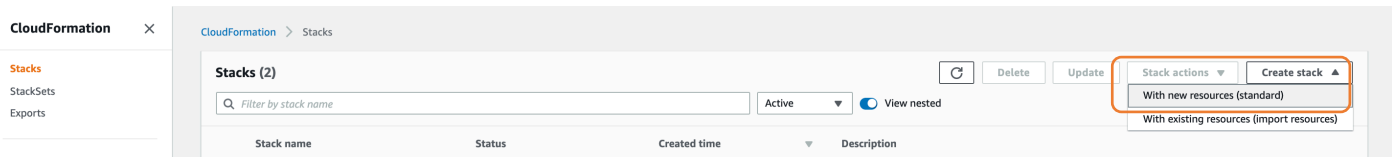
Instructor will provide source database details to participants during main lab to configure source endpoint.

Labs are also available in GitHub - <https://github.com/aws-samples/data-engineering-for-aws-immersion-day>

Create the Instructor Environment

In this section, you are going to create a PostgreSQL RDS instance as data source for AWS Data Migration Service to consume by lab attendees for data migration to Amazon S3 data lake.

1. Sign in to the Console where you will host the source database environment.
2. Navigate to the **AWS CloudFormation** page.
3. Launch a new stack with the AWS CloudFormation template **DMSLab_instructor_CFN.json** provided with your lab package. Make sure to select us-east-1 (Virginia) region.
 - a. On top right corner, Click on **"Create Stack"** and select **"With new resources"**.



- b. In **"Create Stack"** Page, select **"Template is ready"** and for template source, select **"Upload a template file"**.
- c. Locate the **DMSLab_Instructor_CFN.json** template from your local machine.
- d. Click **Next**.

Database Migration Services Instructor Environment for the Lab

The screenshot shows the 'Create stack' wizard in the AWS CloudFormation console. The breadcrumb trail is 'CloudFormation > Stacks > Create stack'. The left sidebar shows four steps: Step 1: Specify template (selected), Step 2: Specify stack details, Step 3: Configure stack options, and Step 4: Review. The main content area is titled 'Create stack'. It has three sections: 1. 'Prerequisite - Prepare template' with three radio buttons: 'Template is ready' (selected), 'Use a sample template', and 'Create template in Designer'. 2. 'Specify template' with a description: 'A template is a JSON or YAML file that describes your stack's resources and properties.' It has a 'Template source' section with two options: 'Amazon S3 URL' and 'Upload a template file' (selected). Below this is an 'Upload a template file' section with a 'Choose file' button and the filename 'n_DMSLab_instructor_CFN.json'. Below that is an 'S3 URL' field with the value 'https://s3-external-1.amazonaws.com/cf-templates-1anfeo75ta7xu-us-east-1/2020066tdv-n_DMSLab_instructor_CFN.json' and a 'View in Designer' button. At the bottom right are 'Cancel' and 'Next' buttons.

- e. In Specify stack details, provide a name for **Stack Name** as “**dmslab-instructor**”.

The screenshot shows the 'Specify stack details' wizard in the AWS CloudFormation console. The title is 'Specify stack details'. It has two main sections: 1. 'Stack name' with a text input field containing 'dmslab-instructor'. Below the field is a note: 'Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-)'. 2. 'Parameters' with a description: 'Parameters are defined in your template and allow you to input custom values when you create or update a stack.' Below this is a box titled 'No parameters' with the text 'There are no parameters defined in your template'. At the bottom right are 'Cancel', 'Previous', and 'Next' buttons.

- f. Click on **Next**.
- g. In review page, review all the details, scroll down and check the box to acknowledge the policy and then click on **Create Stack**.

Database Migration Services Instructor Environment for the Lab

► Quick-create link

Capabilities

The following resource(s) require capabilities: [AWS::IAM::Role]

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. [Learn more](#)

☒ I acknowledge that AWS CloudFormation might create IAM resources.

Cancel Previous Create change set **Create stack**

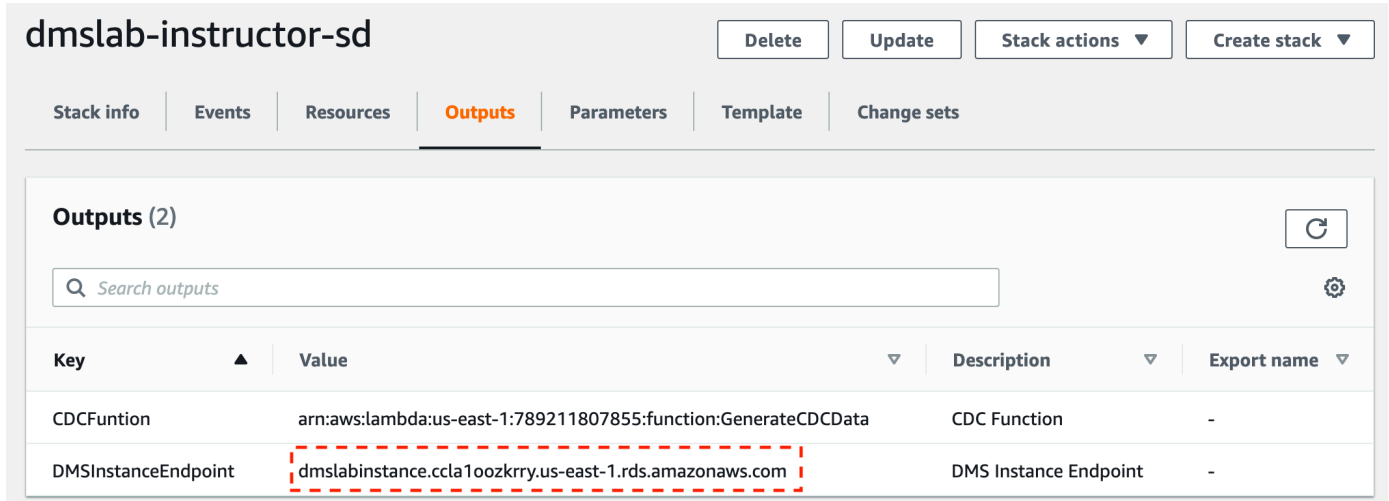
- h. Launch the stack. It may take 15 minutes for the stack to launch. This stack creates a new VPC, Subnets, Security groups, EC2 instance, Route table, Routes, and an RDS Postgres instance.

NOTE: Please make sure the Postgres database is fully populated before proceed with the DMS lab. It takes 15 to 20 minutes to finish, after the stack is launched.

You can see all resources listed below:

| | | | | | | | | | | |
|---------------------------------|--|----------------------------|-----------------|---------------|----------|-------------|--------|--------|-----------------|----------------|
| dmslab-instructor | | | | | | | Delete | Update | Stack actions ▼ | Create stack ▼ |
| Stack info | Events | Resources | Outputs | Parameters | Template | Change sets | | | | |
| Resources (27) | | | | | | | | | | |
| Q Search resources | | | | | | | | | | |
| Logical ID | Physical ID | Type | Status | Status reason | | | | | | |
| EC2SubNet | subnet-0b46150fc43e400bc | AWS::EC2::Subnet | CREATE_COMPLETE | | | | | | | |
| GenerateCDCData | GenerateCDCData | AWS::Lambda::Function | CREATE_COMPLETE | | | | | | | |
| LambdaExecutionRole | dmslab-instructor-LambdaExecutionRole-1QS0V5OCLPR09 | AWS::IAM::Role | CREATE_COMPLETE | | | | | | | |
| RDSSubNet | subnet-0477e0e0071e80331 | AWS::EC2::Subnet | CREATE_COMPLETE | | | | | | | |
| RDSSubNet2 | subnet-00dea43618c4868d8 | AWS::EC2::Subnet | CREATE_COMPLETE | | | | | | | |
| dbpgdataengdmsgroup | dmslab-instructor-dbpgdataengdmsgroup-1pbby1tnpdgq | AWS::RDS::DBParameterGroup | CREATE_COMPLETE | | | | | | | |
| dbsgdefault | dmslab-instructor-dbsgdefault-1p5usgck1gq0a | AWS::RDS::DBSecurityGroup | CREATE_COMPLETE | | | | | | | |
| dbsubnetdefaultdmsinstructorvpc | dmslab-instructor-dbsubnetdefaultdmsinstructorvpc-13e6pv5p7lbvyr | AWS::RDS::DBSubnetGroup | CREATE_COMPLETE | | | | | | | |

- i. Go to the **Outputs** tabs of AWS CloudFormation stack and note down the instance Endpoint information for your RDS endpoint, which will be similar to information shown in below screenshot



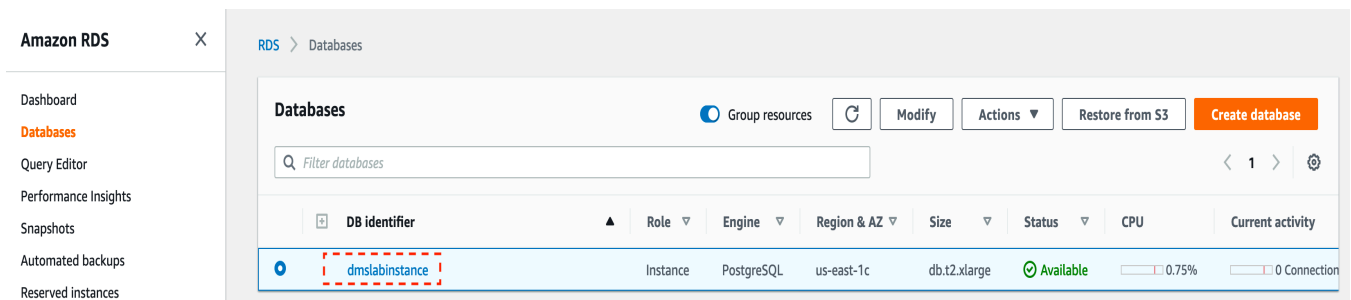
Changing RDS Security Group

Currently your RDS source end point is not open to connect to outside world for security reason. You need to open RDS security group to accept traffic from intended range of IP address. As it is difficult to determine range of IP address of workshop environment, so to have smooth experience of running lab you can temporally allow inbound traffic from all IP address (0.0.0.0/0 CIDR range).

Warning: It is not best practice to allow ALL CIDR range in your database security group. You should never apply open to all IP CIDR range while working on actual workload.

Follow below steps to open security group for students to connect with source RDS data base for DMS full data and CDC data dump:

1. Go to the RDS and double click on “dmslabinstance” **DB identifier** as shown below:



- Click **VPC security groups** under **Connectivity & security** tab as shown below:

The screenshot shows the AWS RDS console for a database instance named 'dmslabinstance'. The 'Connectivity & security' tab is selected. Under the 'Security' section, the 'VPC security groups' list is highlighted with a red dashed box. It contains one group: 'dmslab-instructor-sd-sgrds-launchwizard2-1TQEC430639QV (sg-0fa6619e6be612a98) (active)'.

- In Security group screen, Go to **Inbound** tab and click on **Edit** as shown below

The screenshot shows the AWS Security Groups console. The 'Inbound' tab is selected for the security group 'sg-0fa6619e6be612a98'. The 'Edit' button is highlighted with a red dashed box. Below the tabs, there is a table of inbound rules.

| Type | Protocol | Port Range | Source | Description |
|------------|----------|------------|--|-------------|
| PostgreSQL | TCP | 5432 | 72.21.196.67/32 | |
| PostgreSQL | TCP | 5432 | sg-0d1979886d7072628 (dmslab-instructo | |

- Update Inbound rule to "Anywhere" from hardcoded value "72.21.196.67/32", as shown in below screen. You can also update to your own IP address if want running both lab in same

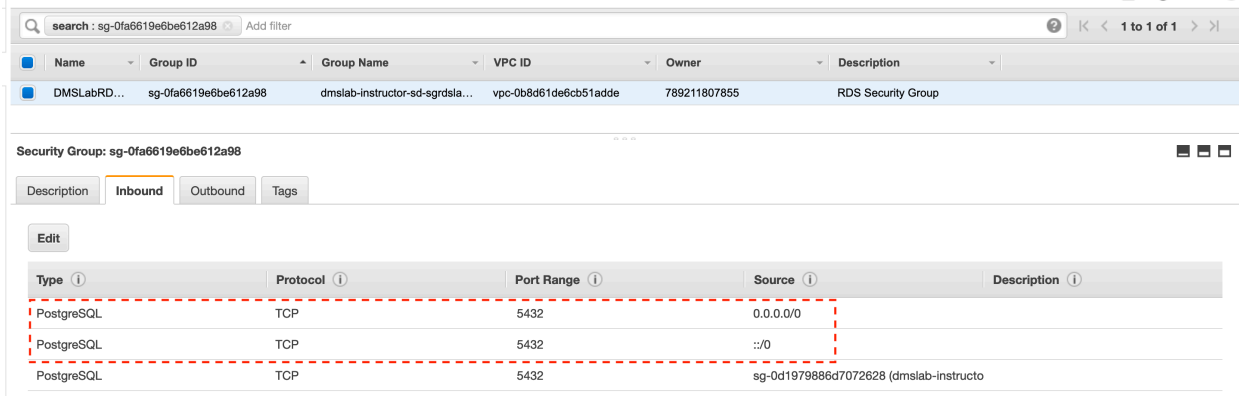
The screenshot shows the 'Edit inbound rules' dialog box. The 'Source' dropdown is open, and 'Anywhere' is selected, highlighted with a red dashed box. The table below shows the rules being edited.

| Type | Protocol | Port Range | Source | Description |
|------------|----------|------------|----------------------|----------------------------|
| PostgreSQL | TCP | 5432 | 0.0.0.0/0 :::/0 | e.g. SSH for Admin Desktop |
| PostgreSQL | TCP | 5432 | sg-0d1979886d7072628 | e.g. SSH for Admin Desktop |

NOTE: Any edits made on existing rules will result in the edited rule being deleted and a new rule created with the new details. This will cause traffic that depends on that rule to be dropped for a very brief period of time until the new rule can be created.

Buttons: Cancel, Save

- Click on **Save** and now everyone will be able to connect to source RDS instance for lab purpose to ingest data using DMS endpoint.



Note: Make sure to remove “Anywhere” inbound rule from security group as soon as you are done with DMS main lab.

Optionally, You can read though the documentation to better understand the source database environment. The GitHub repository for aws-database-migration-samples is located here:

<https://github.com/aws-samples/aws-database-migration-samples/tree/master/PostgreSQL/sampledbs/v1>

Access Database from SQL Client (Optional)

You can follow below instruction to setup SQL Workbench to access your Postgres Database from SQL client:

<https://aws.amazon.com/getting-started/tutorials/create-connect-postgresql-db/>

In SQL Workbench:

Run following query to find out all Schema and table created.

```
SELECT * FROM pg_catalog.pg_tables;
```

Ensure the following 2 functions exists. If anything is missing, check the solution at **Troubleshoot** section.

```
SELECT * FROM pg_stat_user_functions
WHERE funcname in ('generateticketactivity','generatetransferactivity')
```

Use following query to analyze a table

```
select * from schemaname.tablename;
```

For example:

```
select * from dms_sample.player;
```

Database Migration Services Instructor Environment for the Lab

The screenshot shows the SQL Workbench/J interface with two queries executed. The left pane shows the results of a query on the pg_catalog.pg_tables table, and the right pane shows the results of a query on the dms_sample.player table.

Query 1: `SELECT * FROM pg_catalog.pg_tables;`

| schemaname | tablename | tableowner | tablespace | hasindexes | hasrules | hastriggers | rowsecurity |
|------------|-----------------------|------------|------------|------------|----------|-------------|-------------|
| dms_sample | player | master | | true | false | true | false |
| dms_sample | seat_type | master | | true | false | true | false |
| dms_sample | seat | master | | true | false | true | false |
| dms_sample | sport_division | master | | true | false | true | false |
| dms_sample | sport_league | master | | true | false | true | false |
| pg_catalog | pg_statistic | rdadmin | | true | false | false | false |
| pg_catalog | pg_type | rdadmin | | true | false | false | false |
| pg_catalog | pg_policy | rdadmin | | true | false | false | false |
| pg_catalog | pg_authid | rdadmin | pg_global | true | false | false | false |
| dms_sample | mlb_data | master | | false | false | false | false |
| dms_sample | name_data | master | | true | false | false | false |
| dms_sample | nfl_data | master | | false | false | false | false |
| dms_sample | nfl_stadium_data | master | | false | false | false | false |
| dms_sample | sport_type | master | | true | false | true | false |
| dms_sample | person | master | | true | false | true | false |
| dms_sample | sport_location | master | | true | false | true | false |
| dms_sample | sport_team | master | | true | false | true | false |
| dms_sample | sporting_event_ticket | master | | true | false | true | false |
| dms_sample | sporting_event | master | | true | false | true | false |
| dms_sample | ticket_purchase_hist | master | | true | false | true | false |
| pg_catalog | pg_user_mapping | rdadmin | | true | false | false | false |
| pg_catalog | pg_subscription | rdadmin | pg_global | true | false | false | false |
| pg_catalog | pg_attribute | rdadmin | | true | false | false | false |
| pg_catalog | pg_proc | rdadmin | | true | false | false | false |
| pg_catalog | pg_class | rdadmin | | true | false | false | false |
| pg_catalog | pg_attrdef | rdadmin | | true | false | false | false |
| pg_catalog | pg_constraint | rdadmin | | true | false | false | false |
| pg_catalog | pg_inherits | rdadmin | | true | false | false | false |

Query 3: `select * from dms_sample.player;`

| id | sport_team_id | last_name | first_name | full_name |
|-----|---------------|-----------------|------------|-----------|
| 1 | 131 | Adam Loewen | Adam | Loewen |
| 11 | 131 | A.J. Pollock | A.J. | Pollock |
| 21 | 131 | Alex Sanabia | Alex | Sanabia |
| 31 | 131 | Andrew Chafin | Andrew | Chafin |
| 41 | 131 | Andy Marte | Andy | Marte |
| 51 | 131 | Archie Bradley | Archie | Bradley |
| 61 | 131 | Ben Francisco | Ben | Francisco |
| 71 | 131 | Braden Shipley | Braden | Shipley |
| 81 | 131 | Braden Hagens | Braden | Hagens |
| 91 | 131 | Brandon Drury | Brandon | Drury |
| 101 | 131 | Brett Jackson | Brett | Jackson |
| 111 | 131 | Chris Herrmann | Chris | Herrmann |
| 121 | 131 | Chris Owings | Chris | Owings |
| 131 | 131 | Daniel Hudson | Daniel | Hudson |
| 141 | 131 | David Peralta | David | Peralta |
| 151 | 131 | Dominic Leone | Dominic | Leone |
| 161 | 131 | Edwin Escobar | Edwin | Escobar |
| 171 | 131 | Enrique Burgos | Enrique | Burgos |
| 181 | 131 | Evan Marshall | Evan | Marshall |
| 191 | 131 | Gabby Guerrero | Gabby | Guerrero |
| 201 | 131 | Gerald Laird | Gerald | Laird |
| 211 | 131 | Jake Barrett | Jake | Barrett |
| 221 | 131 | Jake Lamb | Jake | Lamb |
| 231 | 131 | Jamie Romak | Jamie | Romak |
| 241 | 131 | Jason Bourgeois | Jason | Bourgeois |

Following sections are optional you only need to execute, if you want to show change data capture replication with DMS.

Generate and Replicate the CDC Data (Optional)

Warning: This step is not required at your initial lab environment setup.

Once the full table load - DMS lab is completed, you can start to generate extra transactions in source database to demonstrate DMS CDC (Change Data Capture) functionality.

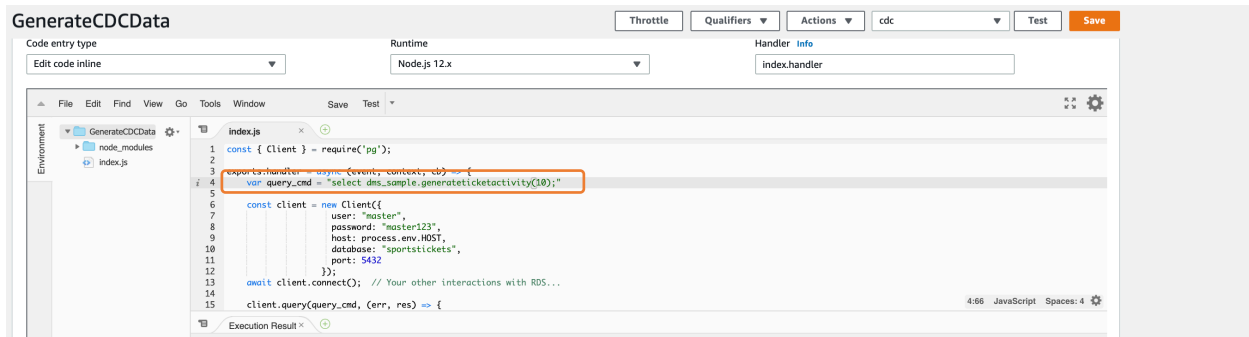
Navigate to Lambda console and you will see a pre-built Lambda function named **"GenerateCDCData"**.

The screenshot shows the AWS Lambda console interface. The left sidebar contains navigation links: Dashboard, Applications, Functions (selected), and Layers. The main content area shows the 'Functions (1)' list. A search bar is present with the text 'Filter by tags and attributes or search by keyword'. Below the search bar, there is a table with the following data:

| Function name | Description | Runtime | Code size | Last modified |
|-----------------|-------------------------------|--------------|-----------|---------------|
| GenerateCDCData | Function to generate CDC data | Node.js 12.x | 208.6 kB | 19 hours ago |

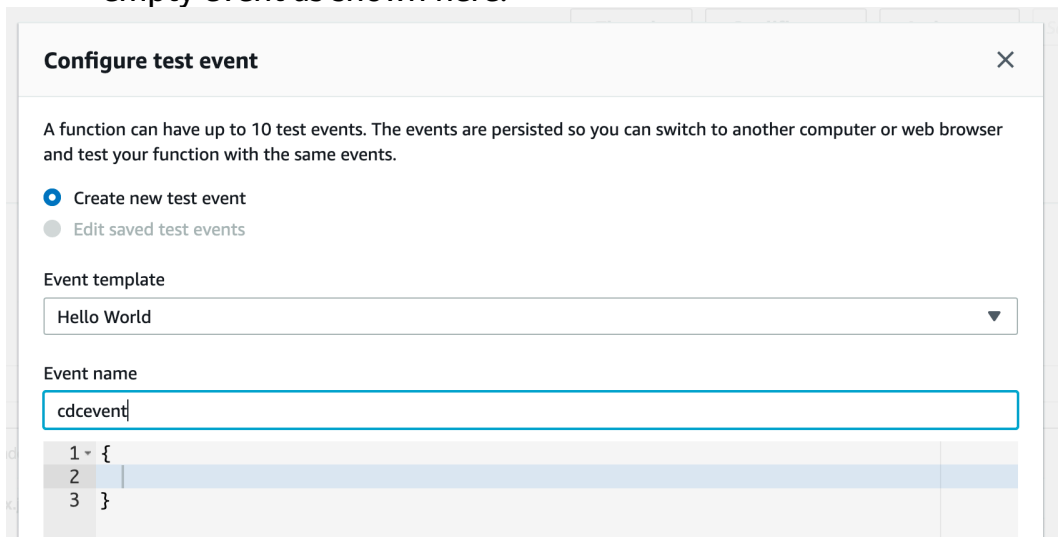
Database Migration Services Instructor Environment for the Lab

1. Click on the function and scroll down. You will see the code for this function. Copy the below query and paste it in the placeholder (value) of this code line:
" var query_cmd= "<insert-SQL-query-here>" "
2. Run this query first: **select dms_sample.generateticketactivity(10);**



This query will generate 10 ticket sales in batches of 1-6 tickets to randomly selected people for a random price (within a range.) A record of each transaction is recorded in the **ticket_purchase_hist** table.

3. Click on **Save** and then click on **Test** to run the function. You can create an empty event as shown here:



You will see no error in lambda log

Database Migration Services Instructor Environment for the Lab

GenerateCDCData Throttle Qualifiers Actions missingfunc Test Save

Environment: GenerateCDCData, node_modules, index.js

```
1 const { Client } = require('pg');
2
3 exports.handler = async (event, context, cb) => {
4   var query_cmd = "select dms_sample.generateTicketActivity(10);";
5
6   const client = new Client({
7     user: "master",
8     password: "master123",
9     host: process.env.HOST,
10    database: "sportstickets",
11    port: 5432
12  });
13
14  await client.connect(); // Your other interactions with RDS...
15}
```

4:68 JavaScript Spaces: 4

Execution Result: Status: **Succeeded** Max Memory Used: 70 MB Time: 651.14 ms

Response:

```
{
  "statusCode": 200,
  "body": "{\"command\":\"SELECT\\\",\\\"rowCount\\\":1,\\\"oid\\\":null,\\\"rows\\\":[{\\\"generateTicketActivity\\\":\\\"\\\"}],\\\"fields\\\":[{\\\"generateTicketActivity\\\":\\\"\\\"}]}\""}
}
```

Request ID: "30827ad1-21c8-4b1f-a4db-1b4aaa381da4"

Function logs:

```
START RequestId: 30827ad1-21c8-4b1f-a4db-1b4aaa381da4 Version: $LATEST
END RequestId: 30827ad1-21c8-4b1f-a4db-1b4aaa381da4
REPORT RequestId: 30827ad1-21c8-4b1f-a4db-1b4aaa381da4 Duration: 651.14 ms Billed Duration: 700 ms Memory Size: 128 MB
```

- Once you've sold some tickets you can run the `generateTransferActivity` procedure. The following will transfer tickets from the owner to another person. The whole "batch" of tickets purchased is transferred 80% of the time and 20% of the time an individual ticket is transferred.

Run this query next in the lambda function:

`select dms_sample.generateTransferActivity(10);`

Click on **Save** and then click on **Test** to run the function.

GenerateCDCData Throttle Qualifiers Actions missingfunc Test Save

Environment: GenerateCDCData, node_modules, index.js

```
1 const { Client } = require('pg');
2
3 exports.handler = async (event, context, cb) => {
4   var query_cmd = "select dms_sample.generateTransferActivity(10);";
5
6   const client = new Client({
7     user: "master",
8     password: "master123",
9     host: process.env.HOST,
10    database: "sportstickets",
11    port: 5432
12  });
13
14  await client.connect(); // Your other interactions with RDS...
15}
```

4:56 JavaScript Spaces: 4

Execution Result: Status: **Succeeded** Max Memory Used: 70 MB Time: 4299.49 ms

Response:

```
{
  "statusCode": 200,
  "body": "{\"command\":\"SELECT\\\",\\\"rowCount\\\":1,\\\"oid\\\":null,\\\"rows\\\":[{\\\"generateTransferActivity\\\":\\\"\\\"}],\\\"fields\\\":[{\\\"generateTransferActivity\\\":\\\"\\\"}]}\""}
}
```

Request ID: "ef8f710f-1ad2-4de5-8a0c-4fcaba539d6c"

Function logs:

```
START RequestId: ef8f710f-1ad2-4de5-8a0c-4fcaba539d6c Version: $LATEST
END RequestId: ef8f710f-1ad2-4de5-8a0c-4fcaba539d6c
REPORT RequestId: ef8f710f-1ad2-4de5-8a0c-4fcaba539d6c Duration: 4299.49 ms Billed Duration: 4300 ms Memory Size: 128 MB
```

Note:

When enabling CDC functionality in DMS, only one DMS instance/task should activate "Ongoing replication" to avoid conflicts.

When replicating to multiple targets, the processing to fan out the updates should begin with the Amazon S3 bucket, that is the target of the DMS task responsible for Ongoing replication. The process should not begin with the source database, as only one CDC process should be tracking and setting the last committed transaction that was replicated.

Troubleshoot

1. Failed to run Lambda function 'GenerateCDCData'.

GenerateCDCData

Throttle

Qualifiers ▼

Actions ▼

test

This function belongs to an application. [Click here](#) to manage it.

Execution result: failed ([logs](#))

▼ Details

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

```

{
  "errorType": "error",
  "errorMessage": "function dms_sample.generateticketactivity(integer) does not exist",
  "trace": [
    "error: function dms_sample.generateticketactivity(integer) does not exist",
    "    at Parser.parseErrorMessage (/var/task/node_modules/pg-protocol/dist/parser.js:278:15)",
    "    at Parser.handlePacket (/var/task/node_modules/pg-protocol/dist/parser.js:126:29)",
    "    at Parser.parse (/var/task/node_modules/pg-protocol/dist/parser.js:39:38)",
    "    at Socket.<anonymous> (/var/task/node_modules/pg-protocol/dist/index.js:8:42)",
    "    at Socket.emit (events.js:310:20)",
  ]
}

```

Cause

The source DB **sportstickets** setup is interrupted. Some database objects, such as the function **generateticketactivity()** will be missing.

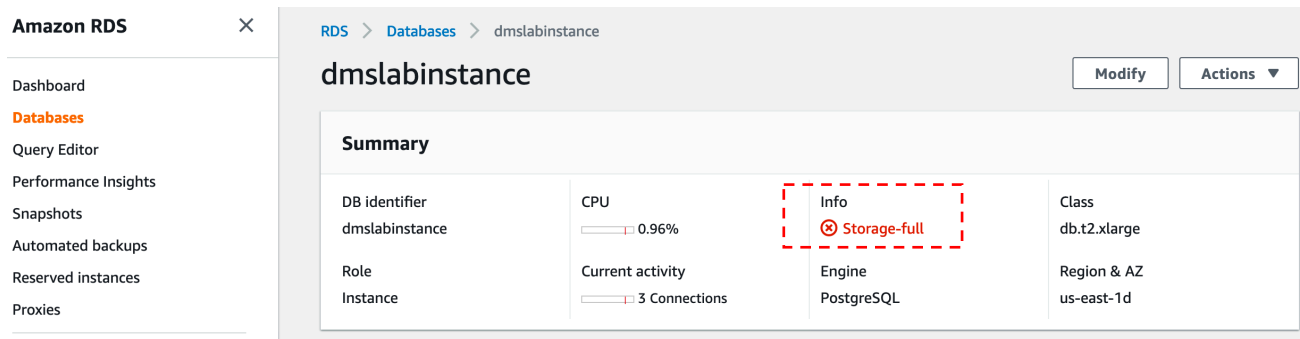
Resolution

Go to EC2 console, reboot the instance **DMSLabEC2**. It will reload the DB and create any objects that were missing. Due to the [re-run issue](#), the table **sporting_event_ticket** will be doubled in size at each reboot. You can manually drop

the table via the following query, before the reboot. Then wait for 20 minutes before checking the missing DB object again.

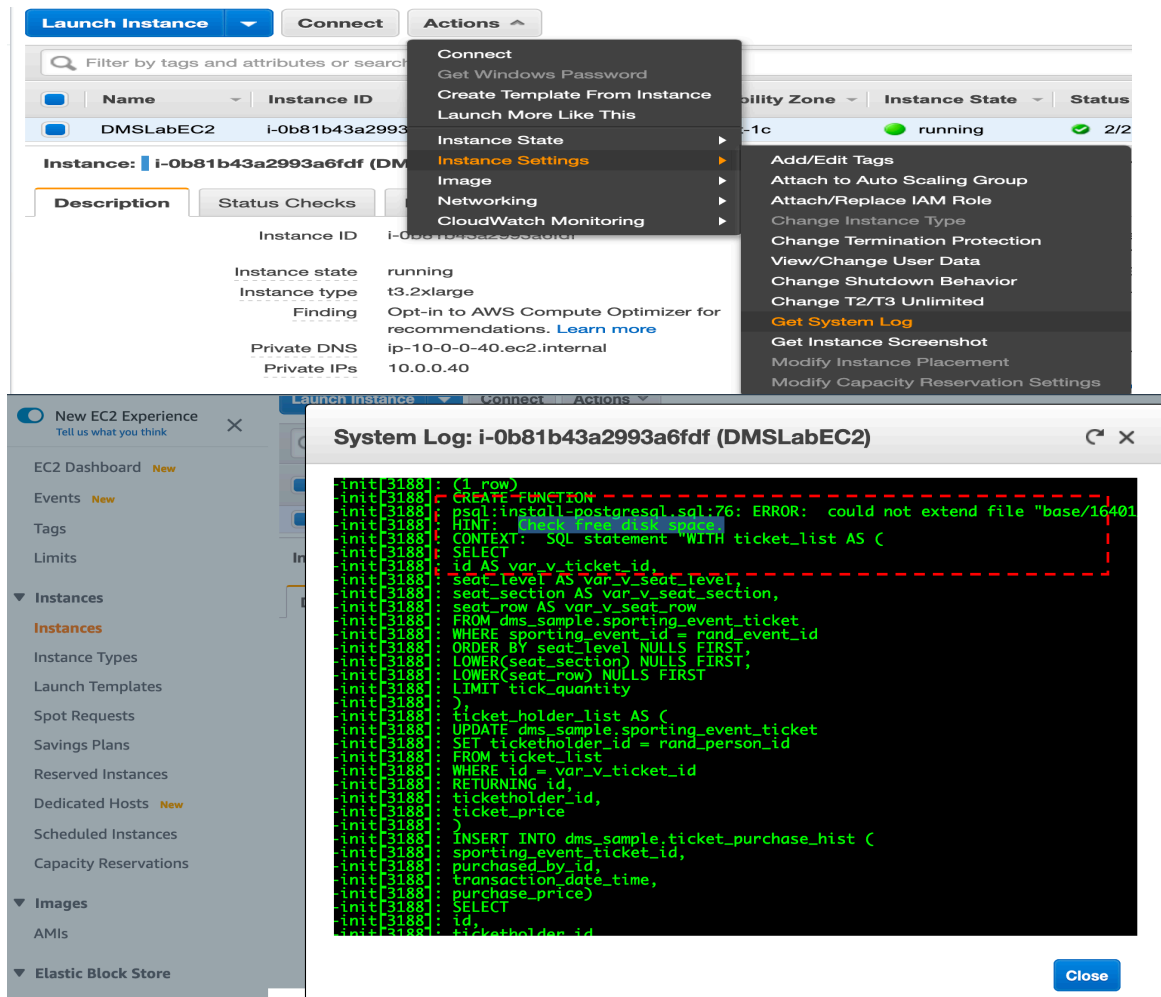
`DROP TABLE dms_sample.sporting_event_ticket CASCADE`

2. RDS source database is out of storage space.



The screenshot shows the Amazon RDS console for the instance 'dmslabinstance'. The 'Summary' tab is active, displaying various metrics. The 'Info' section, highlighted with a red dashed box, indicates a 'Storage-full' error. Other details include: DB identifier 'dmslabinstance', CPU usage at 0.96%, Current activity at 3 Connections, Engine 'PostgreSQL', Class 'db.t2.xlarge', and Region & AZ 'us-east-1d'.

Or you may see 'No Space left on device' error from DMSLabEC2 system log



The screenshot shows the AWS Management Console for the EC2 instance 'DMSLabEC2'. The 'Actions' menu is open, and the 'Get System Log' option is selected. The system log window displays the following error message:

```
init[3188]: (1 row)
init[3188]: CREATE FUNCTION
init[3188]: psql:install-postgresql.sql:76: ERROR: could not extend file "base/16401
init[3188]: HINT: Check free disk space
init[3188]: CONTEXT: SQL statement "WITH ticket_list AS (
init[3188]: SELECT
init[3188]: id AS var_v_ticket_id,
init[3188]: seat_level AS var_v_seat_level,
init[3188]: seat_section AS var_v_seat_section,
init[3188]: seat_row AS var_v_seat_row
init[3188]: FROM dms_sample.sporting_event_ticket
init[3188]: WHERE sporting_event_id = rand_event_id
init[3188]: ORDER BY seat_level NULLS FIRST,
init[3188]: LOWER(seat_section) NULLS FIRST,
init[3188]: LOWER(seat_row) NULLS FIRST
init[3188]: LIMIT tick_quantity
init[3188]: )
init[3188]: ticket_holder_list AS (
init[3188]: UPDATE dms_sample.sporting_event_ticket
init[3188]: SET ticketholder_id = rand_person_id
init[3188]: FROM ticket_list
init[3188]: WHERE id = var_v_ticket_id
init[3188]: RETURNING id,
init[3188]: ticketholder_id,
init[3188]: ticket_price
init[3188]: )
init[3188]: INSERT INTO dms_sample.ticket_purchase_hist (
init[3188]: sporting_event_ticket_id,
init[3188]: purchased_by_id,
init[3188]: transaction_date_time,
init[3188]: purchase_price)
init[3188]: SELECT
init[3188]: id,
init[3188]: ticketholder_id
```

Cause

Check the knowledge center [here](#)

Resolution

Increate the RDS instance disk size, as a quick fix.

RDS > Databases > dmslabinstance

dmslabinstance

Modify

Actions ▼

Summary

| | | | |
|---------------------------------|-----------------------------------|------------------------|---------------------------|
| DB identifier dmslabinstance | CPU 0.75% | Info ⊕ Storage-full | Class db.t2.xlarge |
| Role Instance | Current activity 3 Connections | Engine PostgreSQL | Region & AZ us-east-1d |

Modify DB Instance: dmslabinstance

Instance specifications

DB engine version
Version number of the database engine to be used for this instance.

PostgreSQL 11.5-R1 ▼

DB instance class
Contains the compute and memory capacity of the DB instance.

db.t2.xlarge — 4 vCPU, 16 GiB RAM ▼

Multi-AZ deployment
Specifies if the DB instance should have a standby deployed in another availability zone.

☐ Yes
☒ No

Storage type

General Purpose (SSD) ▼

Allocated storage

40 GiB

This instance supports multiple storage ranges between 20 and 65536 GiB. [See all](#)

Scheduling of modifications

When to apply modifications

☐ Apply during the next scheduled maintenance window
 Current maintenance window: sun:05:40-sun:05:50

☒ Apply immediately
 The modifications in this request and any pending modifications will be asynchronously applied as soon as possible, regardless of the maintenance window setting for this database instance.

⚠ **Potential unexpected downtime**

If you choose to apply changes immediately, please note that any changes in the pending modifications queue are also applied. If any of the pending modifications require downtime, choosing this option can cause unexpected downtime.

Cancel

Back

Modify DB Instance