



# Amazon Web Services Data Engineering Immersion Day

---

Prelab1. Instructor Environment Setup

*July 2021*

## 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> .....	9
<i>Generate and Replicate the CDC Data (Optional)</i> .....	10

## 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 DPUs 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 appropriate AWS region and stick to it for whole of the workshop\*\*\***

The Database Migration Services (DMS) hands-on lab provide a scenario, where participant learns to hydrate Amazon S3 data lake with a relational 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 prelab, you will complete the following tasks:

1. Create a Postgres RDS source database environment.
2. Install the source database.

Once the full data replication is finished by DMS, go to the next step if the CDC lab is required:

3. Execute Lambda function to generate CDC data at the source database environment, to demonstrate CDC (Change Data Capture) replication within DMS.

Relevant information about this prelab:

- CloudFormation execution time: 15 minutes
- Source DB installation time: 20 mins

## Prelab1. Database Migration Services Instructor Environment Setup

In an instructor-led AWS event, participants can get the Postgres RDS database detail from an event dashboard. (The instructor is required to update the database information before each event)

The screenshot shows a section titled "RDS DB Info". Below it is a table with a single row. The first column contains the text "Outputs:" and the second column contains the text "No outputs defined". In the top right corner of the table, there is a blue button with a white "i" icon and the word "Readme". A red arrow points downwards towards this "Readme" button.

RDS DB Info	Readme
<b>Outputs:</b>	No outputs defined

The instructor setup is also available in our online workshop: <https://aws-dataengineering-day.workshop.aws/400/410-pre-lab-1.html>

## 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, for data migration to Amazon S3 data lake.

1. Launch the instructor [CloudFormation](#) stack. (*Make sure you select the correct AWS region*)
2. Under **Capabilities** select the checkbox for "**I acknowledge that AWS CloudFormation might create IAM resources**" and select "**Create stack**".

## Prelab1. Database Migration Services Instructor Environment Setup

**Quick create stack**

**Template**

Template URL  
[https://s3.amazonaws.com/aws-dataengineering-day.workshop.aws/DMSLab\\_instructor\\_CFN.yaml](https://s3.amazonaws.com/aws-dataengineering-day.workshop.aws/DMSLab_instructor_CFN.yaml)

Stack description  
DMS Lab Instructor account

**Stack name**

Stack name  
dmslab-instructor

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

**Parameters**

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

LatestAmiId  
/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86\_64-gp2

SourceBucket  
S3 bucket which contains the source object  
aws-dataengineering-day.workshop.aws

SourceKey  
S3 Key which contains the source object  
dmslambda.zip

**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](#) [Create change set](#) [Create 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.

**Warning:** make sure the Postgres database is fully populated before proceed to the DMS lab. It takes additional 20 minutes to finish, after the CloudFormation setup is completed.

## Prelab1. Database Migration Services Instructor Environment Setup

You can see all resources listed below:

dmslab-instructor						
Stack info		Events	Resources	Outputs	Parameters	Template
Resources (27)						
Search resources						
Logical ID	Physical ID	Type	Status	Status reason		
EC2SubNet	subnet-0b46150fc43e400bc	AWS::EC2::Subnet	<span>CREATE_COMPLETE</span>	-		
GenerateCDCData	GenerateCDCData	AWS::Lambda::Function	<span>CREATE_COMPLETE</span>	-		
LambdaExecutionRole	dmslab-instructor-LambdaExecutionRole-1QSOV5OCLPR09	AWS::IAM::Role	<span>CREATE_COMPLETE</span>	-		
RDSSubNet	subnet-0477e0e0071e80331	AWS::EC2::Subnet	<span>CREATE_COMPLETE</span>	-		
RDSSubNet2	subnet-00dea43618c4868d8	AWS::EC2::Subnet	<span>CREATE_COMPLETE</span>	-		
dbpgdataengdmsgroup	dmslab-instructor-dbpgdataengdmsgroup-1pbby1nnpdqg	AWS::RDS::DBParameterGroup	<span>CREATE_COMPLETE</span>	-		
dbsgdefault	dmslab-instructor-dbsgdefault-1p5usgck1gq0a	AWS::RDS::DBSecurityGroup	<span>CREATE_COMPLETE</span>	-		
dbsubnetdefaultdmsinst ructorvpc	dmslab-instructor-dbsubnetdefaultdmsin structorvpc-13e6pv5p7lbvy	AWS::RDS::DBSub netGroup	<span>CREATE_COMPLETE</span>	-		

- Go to the **Outputs** tab of AWS CloudFormation stack and you can find the Postgres RDS database endpoint, which will be similar to information shown in below screenshot

dmslab-instructor-sd						
Stack info		Events	Resources	Outputs	Parameters	Template
Outputs (2)						
Search outputs						
Key	Value		Description		Export name	
CDCFunction	arn:aws:lambda:us-east-1:unction:GenerateCDCData		CDC Function	-		
DMSInstanceEndpoint	dmslabinstance.ccla1oozkrry.us-east-1.rds.amazonaws.com		DMS Instance Endpoint	-		

## 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 temporarily 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. If you are in a self-paced workshop, the better secure way is to whitelist an IP address from DMS lab.**

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 Console](#) and double click on “dmslabinstance” DB identifier as shown below:

The screenshot shows the Amazon RDS Databases console. On the left, there's a sidebar with links: Dashboard, Databases (which is highlighted in orange), Query Editor, Performance Insights, Snapshots, Automated backups, and Reserved instances. The main area is titled 'Databases' and shows a table with one row for 'dmslabinstance'. The row is highlighted with a red dashed border. The columns include DB identifier, Instance, Engine, Region & AZ, Size, Status, CPU, and Current activity. The status is 'Available' with a green icon, and the CPU usage is 0.75%.

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

The screenshot shows the 'dmslabinstance' database details page. At the top, there's a summary section with various metrics like DB identifier, CPU usage, and engine type. Below this is the 'Connectivity & security' tab, which is highlighted in orange. This tab contains three sections: 'Endpoint & port', 'Networking', and 'Security'. In the 'Security' section, there's a list of 'VPC security groups' with one item highlighted by a red dashed box: 'dmslab-instructor-sg-sqrslaunchwizard2-1TQEC430639QV (active)'. Other items in the list are '(sg-0fa6619e6be612a98)' and 'Public accessibility Yes'.

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

## Prelab1. Database Migration Services Instructor Environment Setup

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

4. Update Inbound rule to “Anywhere” from hard coded value “**10.0.0.5/32**”, as shown in below screen. Make sure to remove the “Anywhere” inbound rule from security group, as soon as you are done with DMS lab.

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.

5. If you are in an AWS hosted event, proceed to **step 8**.
6. If you are running both instructor and student labs in a single AWS account, for example in a self-paced environment, replace the “Anywhere” rule by an IP address from student lab instead. In this example, we will allow AutoComplete DMS lab to access the RDS.

## Prelab1. Database Migration Services Instructor Environment Setup

The screenshot shows the AWS VPC Security Groups console. The top navigation bar includes 'Security Groups (1/10)', 'Info', 'Actions', and 'Create security group'. Below the search bar, there are columns for Name, Security group ID, Security group name, and VPC ID. A row for 'DMSLabRDS-SG' is selected, showing its details: sg-0f234a18e77dc1582, dmslab-instructor-sgrdslauchwizard2-1FNR..., and vpc-07233b6557617. Other rows include 'default' and 'auto-dmslab-sgdefault-1XZJQLPEVLA7E'. At the bottom, tabs for 'Details', 'Inbound rules' (which is selected), 'Outbound rules', and 'Tags' are visible. The 'Inbound rules' section shows two entries: one for PostgreSQL TCP port 5432 from IP 23.21.225.101/32, and another for PostgreSQL TCP port 5432 from security group sg-0478163b68b3d05b2 (dmslab-instructor-sgDMSLabSG-Y3OVJ46APB2J).

7. Go to [VPC NAT gateways Console](#), and look for the IP address you need to add to the RDS security group.

- If you are running DMS hands-on lab, note down the IP address tagged with "dmslab-student".

The screenshot shows the AWS VPC NAT gateways console. The top navigation bar includes 'NAT gateways (1/2)', 'Info', 'Actions', and 'Create NAT gateway'. Below the search bar, there are columns for Name, NAT gateway ID, State, State message, and Elastic IP address. Two NAT gateways are listed: 'NatGateway' (nat-095cf3cdd2514f3e7) with state 'Available' and IP 184.73.4.41, and another 'NatGateway' (nat-04b101c8c741c31a0) with state 'Available' and IP 23.21.225.101. At the bottom, tabs for 'Details', 'Monitoring', and 'Tags' are visible. The 'Tags' section shows two entries: 'aws:cloudformation:stack-name' with value 'dmslab-student' and 'aws:cloudformation:logical-id' with value 'NatGateway'.

- Or if you are running the AutoComplete DMS Lab, copy the IP address tagged with "auto-dmslab".

## Prelab1. Database Migration Services Instructor Environment Setup

NAT gateways (1/2) [Info](#)

Name	NAT gateway ID	State	State message	Elastic IP address
NatGateway	nat-095cf3cdd2514f3e7	Available	-	184.73.4.41
NatGateway	nat-04b101c8c741c31a0	Available	-	23.21.225.101

Tags

Key	Value
aws:cloudformation:stack-name	auto-dmslab
aws:cloudformation:logical-id	NatGateway

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

Security Group: sg-0fa6619e6be612a98

Type	Protocol	Port Range	Source	Description
PostgreSQL	TCP	5432	0.0.0.0/0	
PostgreSQL	TCP	5432	::/0	
PostgreSQL	TCP	5432	sg-0d1979886d7072628 (dmslab-instruc...	

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

Optionally, you can read through 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:

## Prelab1. Database Migration Services Instructor Environment Setup

<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 **Troubleshooting** 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;
```

The screenshot shows the SQL Workbench interface with two queries and their results. The top query is:

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

The results for this query are:

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	rdsadmin		true	false	false	false
pg_catalog	pg_type	rdsadmin		true	false	false	false
pg_catalog	pg_policy	rdsadmin		true	false	false	false
pg_catalog	pg_authid	rdsadmin	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	rdsadmin		true	false	false	false
pg_catalog	pg_subscription	rdsadmin	pg_global	true	false	false	false
pg_catalog	pg_attribute	rdsadmin		true	false	false	false
pg_catalog	pg_proc	rdsadmin		true	false	false	false
pg_catalog	pg_class	rdsadmin		true	false	false	false
pg_catalog	pg_attrdef	rdsadmin		true	false	false	false
pg_catalog	pg_Constraint	rdsadmin		true	false	false	false
pg_catalog	pg_inherits	rdsadmin		true	false	false	false

The bottom query is:

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

The results for this query are:

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	Brady Hagens	Brady	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.**

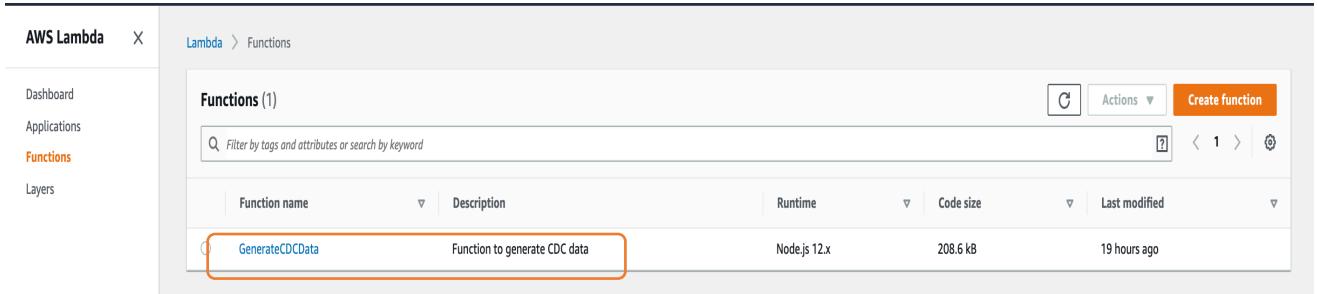
### Generate and Replicate the CDC Data (Optional)

## Prelab1. Database Migration Services Instructor Environment Setup

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

Once the full data replication in 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**”.

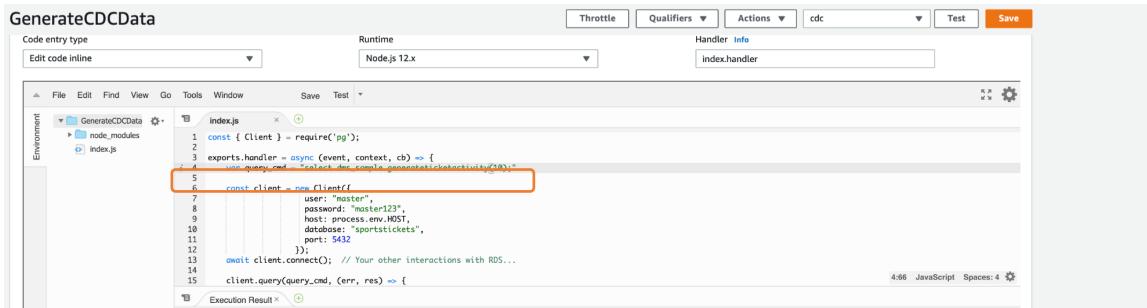


The screenshot shows the AWS Lambda console interface. On the left, there's a sidebar with links for Dashboard, Applications, Functions (which is highlighted in orange), and Layers. The main area is titled "Functions (1)" and contains a table with one row. The row for "GenerateCDCData" is highlighted with an orange border. The table columns include Function name, Description, Runtime, Code size, and Last modified. The "GenerateCDCData" entry has "Function to generate CDC data" in the Description column, "Node.js 12.x" in the Runtime column, "208.6 kB" in the Code size column, and "19 hours ago" in the Last modified column.

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);**

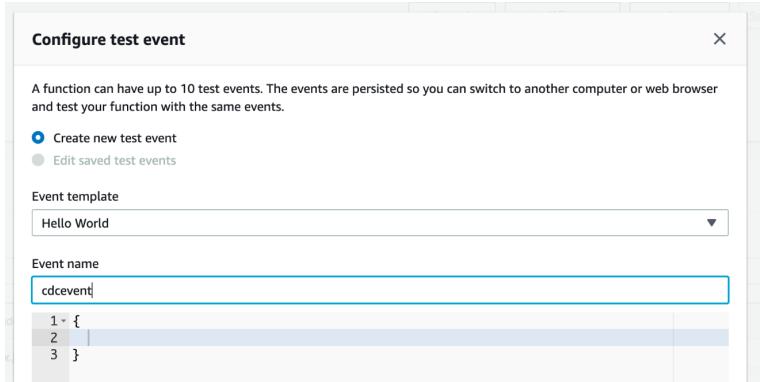


The screenshot shows the AWS Lambda function editor for the "GenerateCDCData" function. At the top, there are settings for Code entry type (Edit code inline), Runtime (Node.js 12.x), Handler (index.handler), and a Save button. Below that is a toolbar with Throttle, Qualifiers, Actions, and Test buttons. The main area is a code editor with an "index.js" file open. The code defines a handler function that runs a PostgreSQL query. A specific line of code, `const query\_cmd = "select dms\_sample.generateticketactivity(10);";`, is highlighted with an orange box. To the right of the code editor, there's a status bar showing "4:66 JavaScript Spaces: 4".

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:

## Prelab1. Database Migration Services Instructor Environment Setup



You will see no error in lambda log

```
const Client = require('pg');
exports.handler = async (event, context, cb) => {
  var query_cmd = "select dms_sample.generateticketactivity(10);"
  const client = new Client({
    user: "master",
    password: "master123",
    host: process.env.HOST,
    database: "sportstickets",
    port: 5432
  });
  await client.connect(); // Your other interactions with RDS...
}

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

- 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.

## Prelab1. Database Migration Services Instructor Environment Setup

The screenshot shows the AWS Lambda function editor for a function named "GenerateCDCData". The "index.js" file contains the following code:

```
const { Client } = require('pg');
exports.handler = async (event, context, cb) => {
  var query_cmd = "select dms_sample.generatetransferactivity(10);"
  const client = new Client({
    user: "master",
    password: "master123",
    host: process.env.HOST,
    database: "sportstickets",
    port: 5432
  });
  await client.connect(); // Your other interactions with RDS...
}
```

The "Execution Result" tab shows the following output:

```
Status: Succeeded Max Memory Used: 70 MB Time: 4299.49 ms
```

Response:

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

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: 1
```

### 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.

## Troubleshooting

1. Failed to run Lambda function '**GenerateCDCData**'.

The screenshot shows the AWS Lambda console for the 'GenerateCDCData' function. At the top, there are tabs for 'Throttle', 'Qualifiers ▾', 'Actions ▾', and a dropdown set to 'test'. Below these, a message says 'This function belongs to an application. Click here to manage it.' A red box highlights the 'Execution result: failed (logs)' section. Underneath, a 'Details' link is shown. The main content area contains the error message: "The area below shows the result returned by your function execution. Learn more about returning results from your function." followed by a JSON log entry:

```
{  "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 database setup is interrupted. Some database objects, such as the function **generateticketactivity()** is 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 by the following script before each reboot. Then wait for 20 minutes before checking the missing DB object again.

```
DROP TABLE dms_sample.sporting_event_ticket CASCADE
```

## Prelab1. Database Migration Services Instructor Environment Setup

### 2. RDS source database is **out of storage space**.

The screenshot shows the Amazon RDS Databases page. On the left, there's a sidebar with links like Dashboard, Databases (which is selected and highlighted in orange), Query Editor, Performance Insights, Snapshots, Automated backups, Reserved instances, and Proxies. The main area is titled 'dmslabinstance' and has a 'Summary' section. It displays the DB identifier as 'dmslabinstance', CPU usage at 0.96%, a role of 'Instance', current activity with 3 connections, an engine of 'PostgreSQL', and a class of 'db.t2.xlarge'. A red dashed box highlights the 'Info' column which shows a warning icon and the text 'Storage-full'. Below the summary, there are sections for Engine, Region & AZ, and a detailed view of the instance.

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

The screenshot shows the AWS EC2 Instances page. On the left, there's a sidebar with links for New EC2 Experience, EC2 Dashboard, Events, Tags, Limits, Instances (selected and highlighted in orange), Images, and Elastic Block Store. The main area shows an instance named 'DMSLabEC2' with Instance ID 'i-0b81b43a2993a6fdf'. The instance state is 'running' and the type is 't3.2xlarge'. A red dashed box highlights the 'Status Checks' tab. An open Actions dropdown menu shows options like Connect, Create Template From Instance, Launch More Like This, Instance State, Instance Settings (selected and highlighted in orange), Image, Networking, CloudWatch Monitoring, Add/Edit Tags, Attach to Auto Scaling Group, Attach/Replace IAM Role, Change Instance Type, Change Termination Protection, View/Change User Data, Change Shutdown Behavior, Change T2/T3 Unlimited, Get System Log (highlighted with a red box), Get Instance Screenshot, Modify Instance Placement, and Modify Capacity Reservation Settings. Below the instance details, there's a 'System Log' window titled 'System Log: i-0b81b43a2993a6fdf (DMSLabEC2)'. The log output shows several PostgreSQL error messages related to storage issues, such as 'could not extend file "base/16401"' and 'CONTEXT: SQL statement "WITH ticket\_list AS ('. A red dashed box highlights the error text.

### Cause

Check the knowledge center [here](#)

### Resolution

## Prelab1. Database Migration Services Instructor Environment Setup

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

The screenshot shows the AWS RDS 'Databases' section with 'dmslabinstance' selected. A red dashed box highlights the 'Modify' button in the top right corner of the summary card. The summary card displays the following details:

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

The 'Modify DB Instance: dmslabinstance' page is open. A red dashed box highlights the 'Allocated storage' field under 'Instance specifications'. The current value is 40 GiB. A red dashed box also highlights the 'Scheduling of modifications' section, specifically the 'Apply immediately' option, which is selected. A warning message about potential downtime is shown in a callout box. The bottom right contains 'Cancel', 'Back', and 'Modify DB Instance' buttons, with 'Modify DB Instance' being highlighted by a red dashed box.