[](http://aws.amazon.com/)

Amazon Web Services

Data Engineering Immersion Day

Database Migration Services Instructor Setup Instructions

Jun 2019

Table of Contents

[Limit Instruction: 2](#_Toc10041279)

[Introduction 2](#_Toc10041280)

[Create the Instructor Environment 3](#_Toc10041281)

[Access Database from SQL Client (Optional) 6](#_Toc10041282)

[Create the Change Data Capture Environment (Optional) 8](#_Toc10041283)

[Generate the CDC Data (Optional) 13](#_Toc10041284)

[Appendix: AWS CloudFormation Template 15](#_Toc10041285)

# 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 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:

1. Create the source database environment.
2. Hydrate the source database environment.
3. Update the source database environment to demonstrate CDC replication within DMS.

Relevant information about this lab:

* Expected setup time: 45 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.

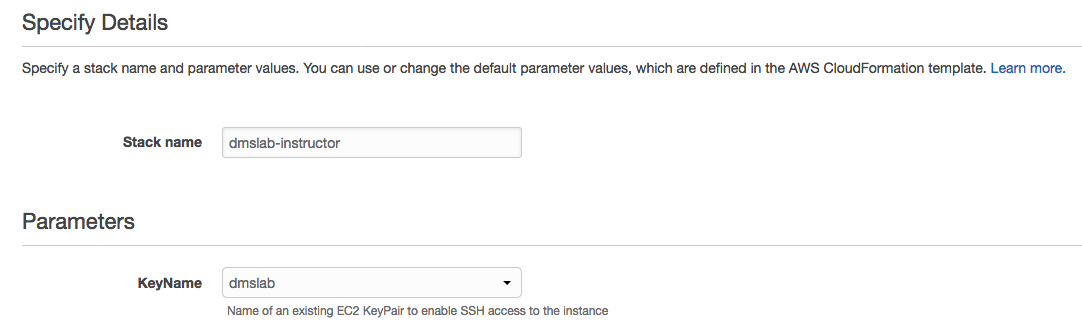
# Create the Instructor Environment

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.

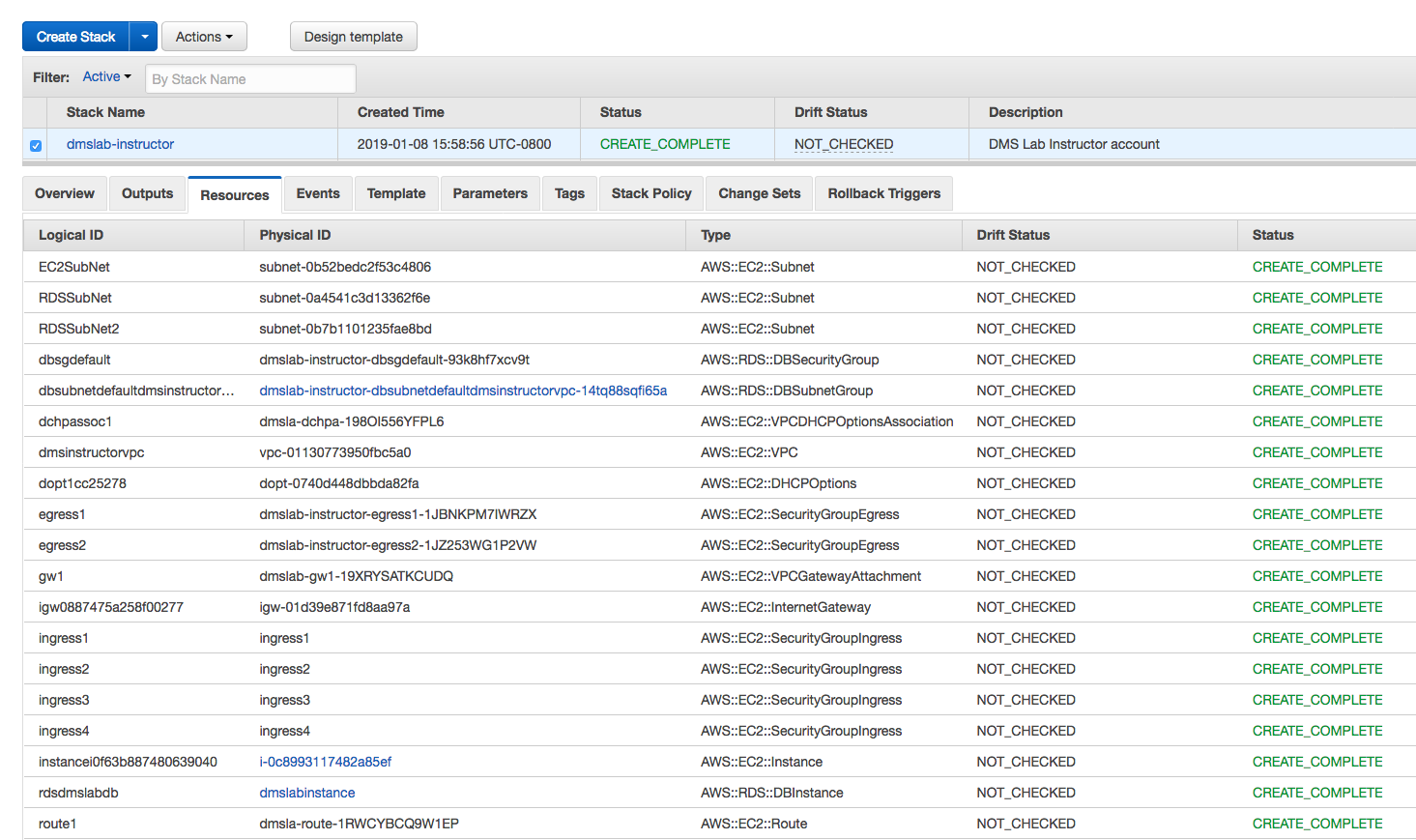
Alternatively, You can follow instruction in [Appendix : AWS CloudFormation Template](#_Appendix_:_AWS) to create your AWS CloudFormation template for this lab.

1. Give stack name and Enter the Key Pair to use. Please make sure to create an Amazon EC2 Key pair if don’t have one in select us-east-1 (Virginia) region. Follow User guide [Amazon EC2 key pairs](https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html) to create a key pair.

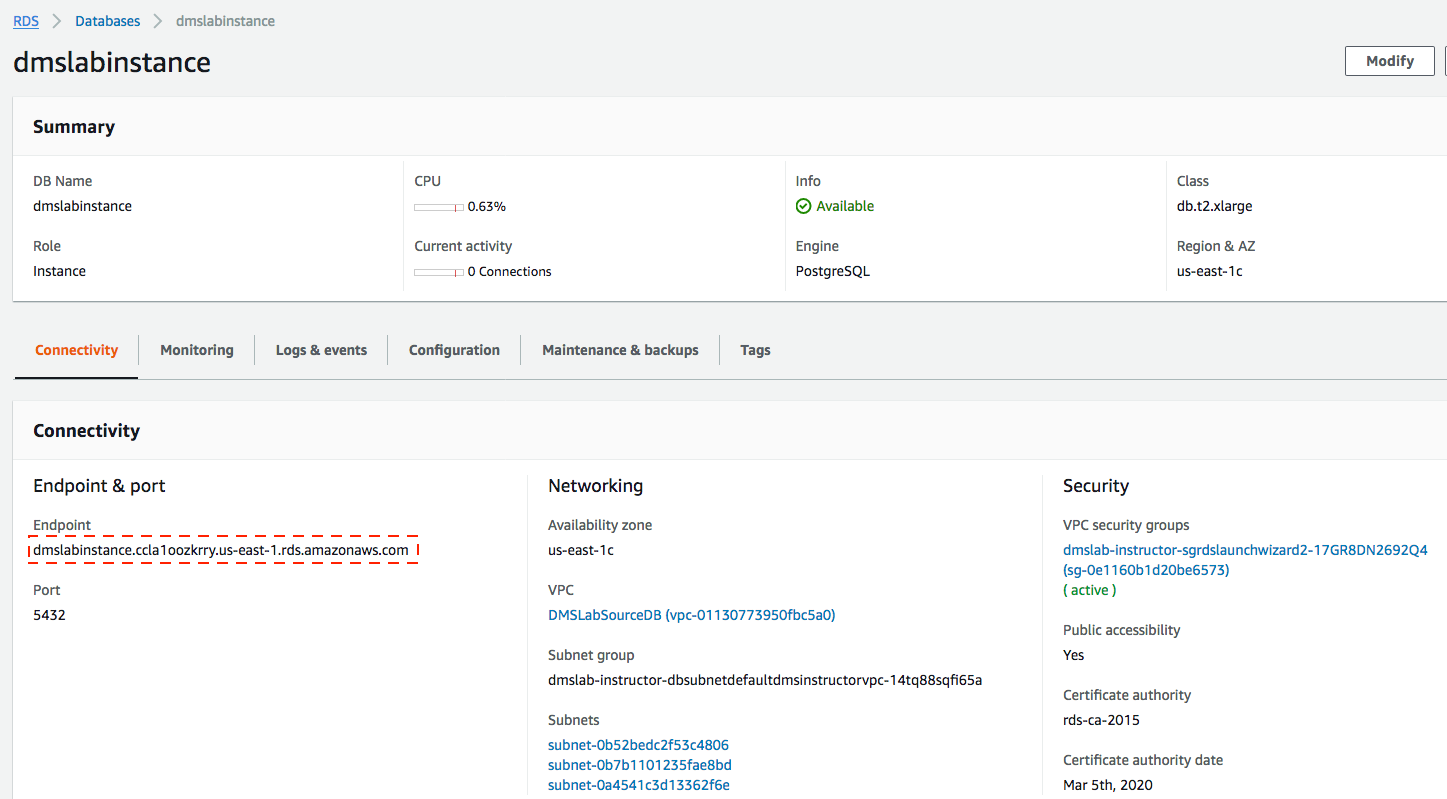


1. Enter a tag for the Name that identifies the resources as part of this lab.
2. Launch the stack. It may take 15 minutes for the stack to launch.

This stack creates a new VPC, Subnets, Security groups, EC2 instance, Route table, Routes, and an RDS Postgres instance. You can see all resources listed below:



1. Once the stack is launched, navigate to the Amazon Relational Database Service (Amazon RDS) page and select **Instances** > **dmslabinstance and** Copy the instance Endpoint information as shown in below screenshot



1. [SSH to the ec2 instance](https://docs.aws.amazon.com/quickstarts/latest/vmlaunch/step-2-connect-to-instance.html) created by this template and execute the following command(s) in sequence:

cd aws-database-migration-samples/PostgreSQL/sampledb/v1/

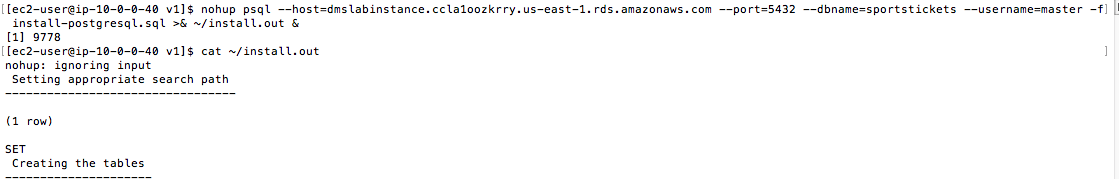
export PGPASSWORD=master123

nohup psql --host=<instance endpoint> --port=5432 --dbname=sportstickets --username=master -f install-postgresql.sql >& ~/install.out &

For example:

nohup psql --host=dmslabinstance.ccla1oozkrry.us-east-1.rds.amazonaws.com --port=5432 --dbname=sportstickets --username=master -f install-postgresql.sql >& ~/install.out &

You will get a process id upon successful submission:



To see how your job is doing you can observe install.out file by giving command

cat ~/install.out

Note:

* + 1. It may include messages about non-existing table, but you should not see any errors and the background process will end when complete. You can check whether the process is still running with the following command.

ps -aef | grep psql

Wait 15 to 20 minutes for the install to complete.

The github repository for aws-database-migration-samples is located here:

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

You can read though the documentation to better understand the source database environment.

# Access Database from SQL Client (Optional)

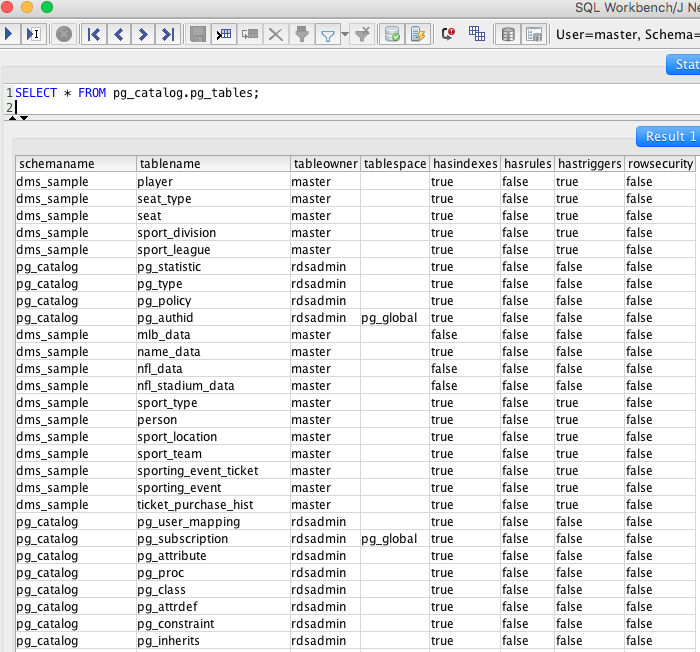
You can follow below instruction to setup SQL Workbench to access your Postgre Database from SQL client:

<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;

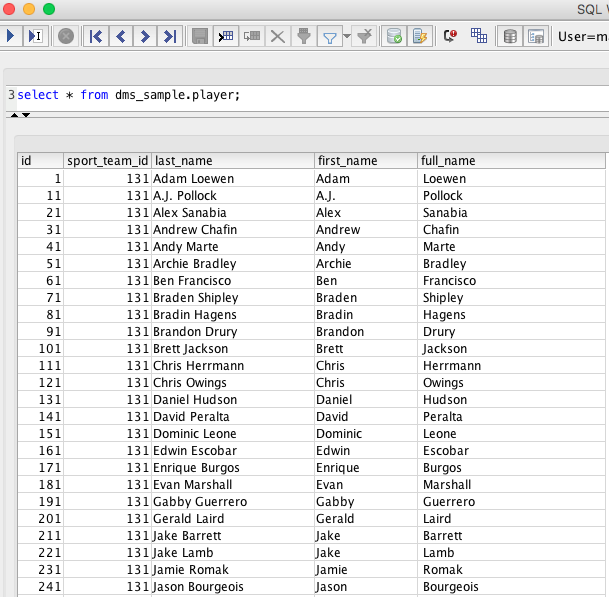


Use following query to analyze a table

select \* from schemaname.tablename;

For example:

select \* from dms\_sample.player;

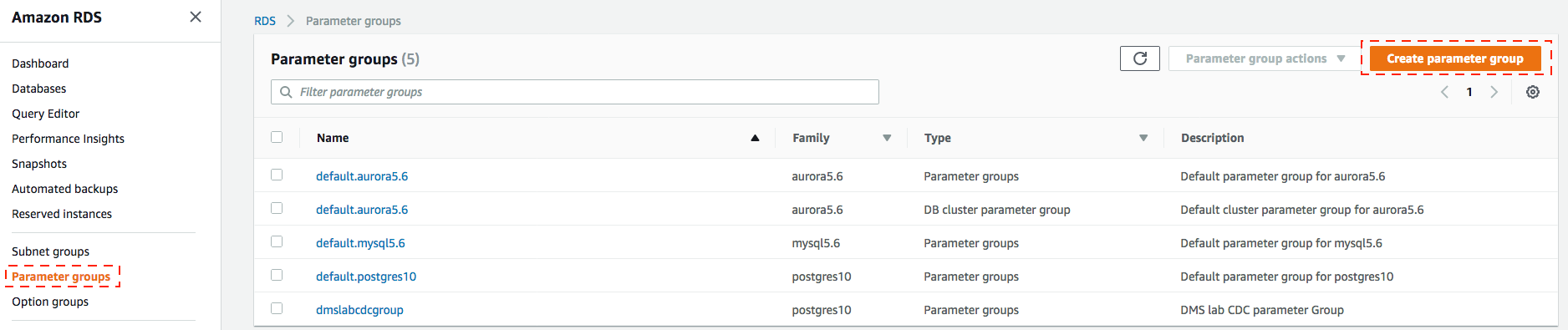


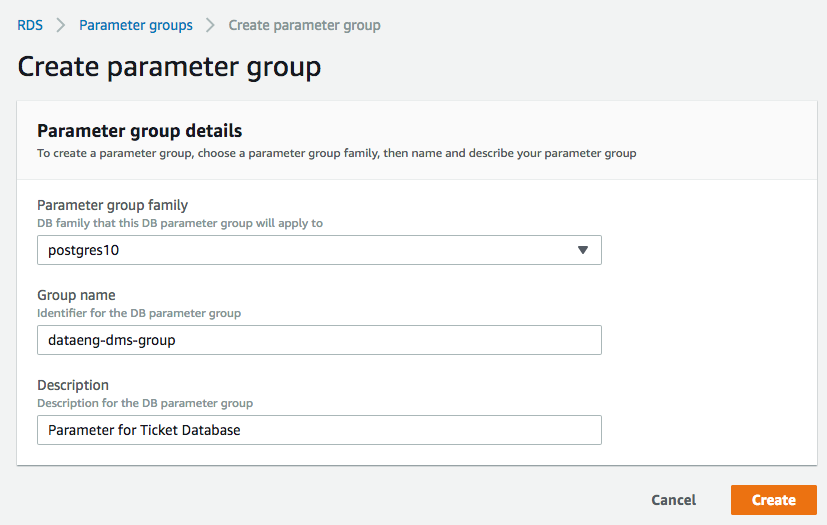
**Following sections are optional you only need to execute, if you want to show change data capture replication with DMS.**

# Create the Change Data Capture Environment (Optional)

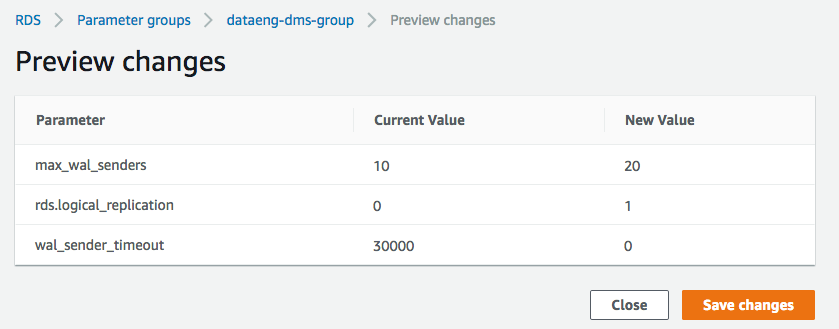
If you are planning to show ongoing CDC capability you should also set the following attributes:

1. Create a custom DB parameter group in RDS console for postgres10. Go to Amazon RDS Parameter groups and click on Create Parameter group button as shown below:

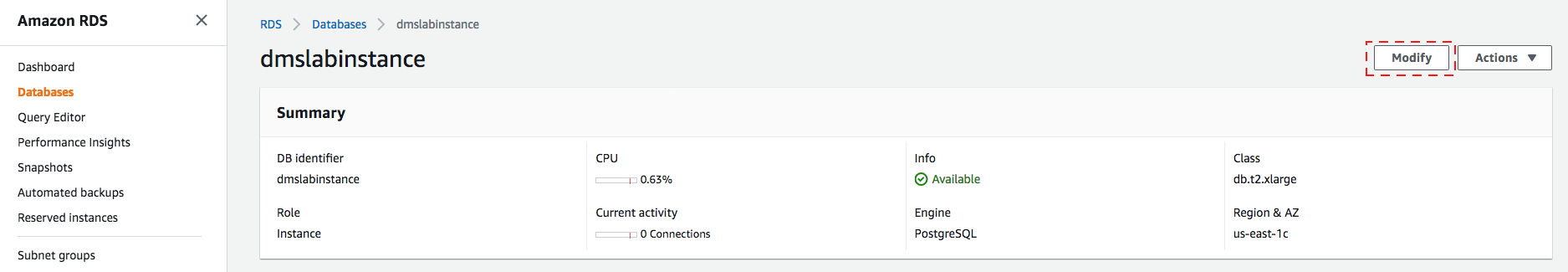




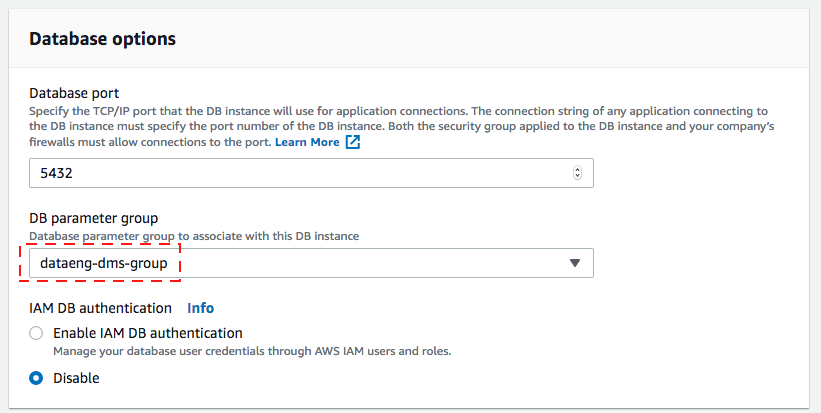
1. In your custom parameter group, you should:
   * + - 1. Set rds.logical\_replication to 1. This is a static parameter that requires a reboot of the DB instance for the parameter to take effect .
         2. Set the wal\_sender\_timeout parameter to 0. Setting this parameter to 0 prevents PostgreSQL from terminating replication connections that are inactive longer than the specified timeout.
         3. Increase max\_wal\_senders parameter from 10 to 20 to accommodate for Data Migration Service.



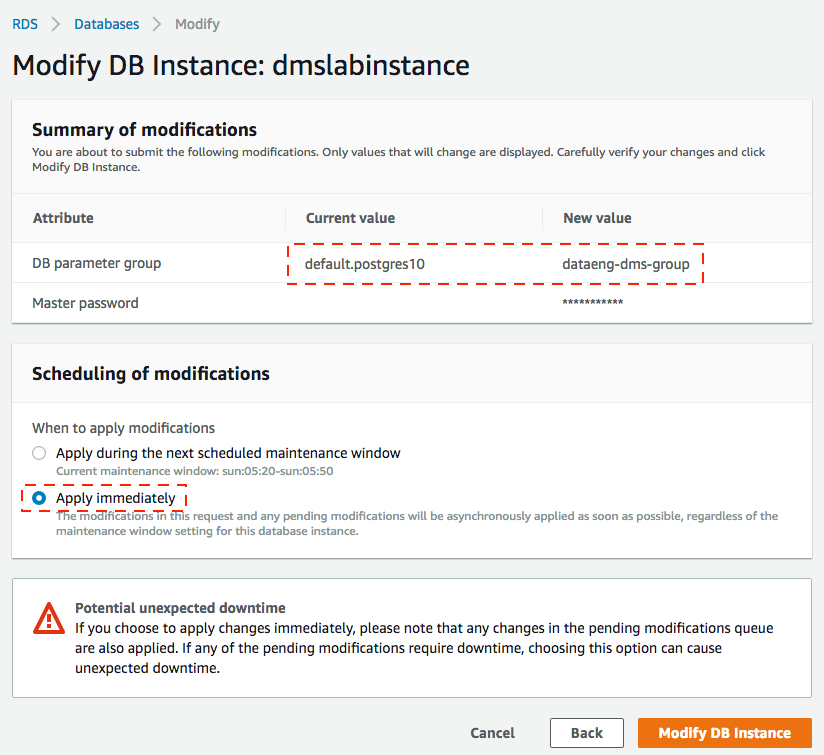
1. Go to your RDS Databases and click on modify.



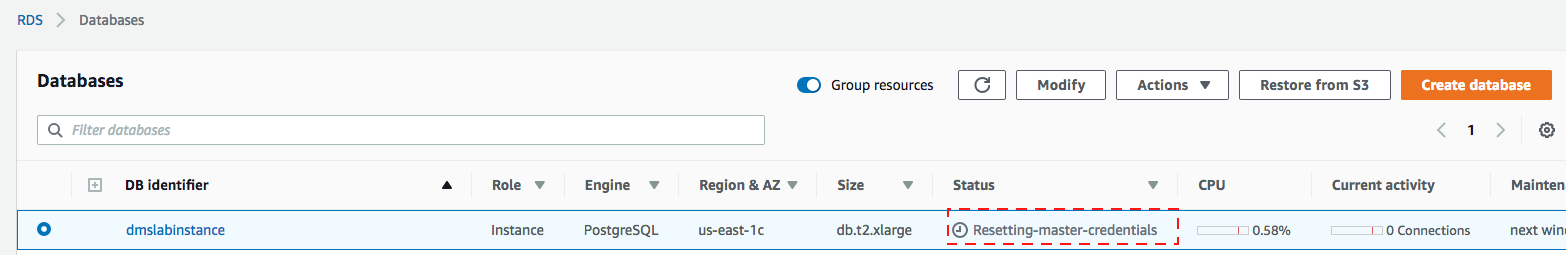
1. Scroll down to **Database options** section, Modify the RDS instance we created, and associate the custom parameter group with the RDS instead of the default parameter group, and click continue at bottom of screen.



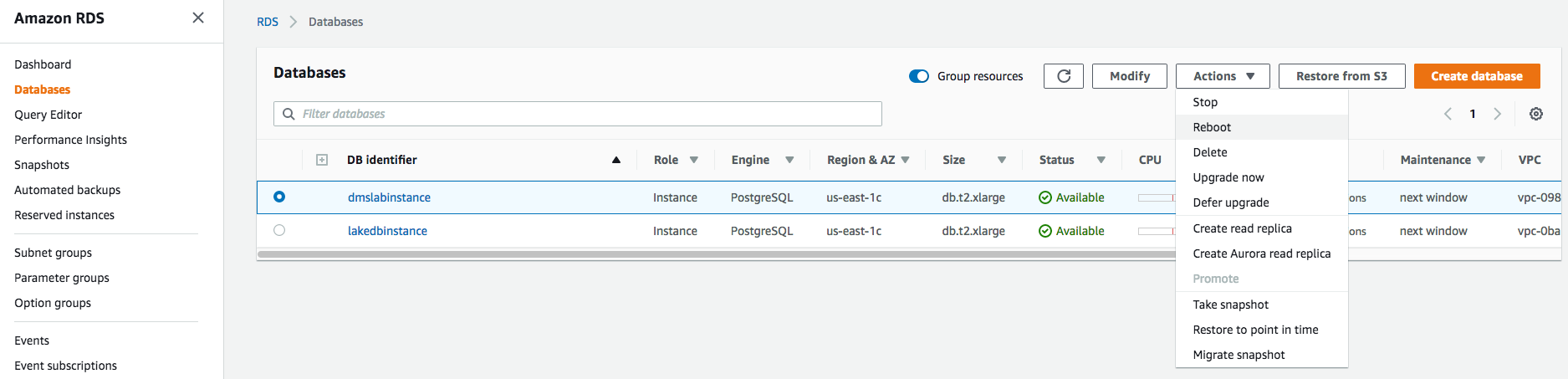
1. Review changes, select **Apply immediately**.



1. click on **Modify DB Instance** button.
2. Wait until status change to available.



1. Click on the Actions drop down at reboot the db instances as shown below:



Wait until status change to available.

1. Now SSH to your ec2 instance and run following:

psql --host=<instanceaddress> --port=5432 --dbname=sportstickets --username=master

For example : nohup psql --host=dmslabinstance.ccla1oozkrry.us-east-1.rds.amazonaws.com --port=5432 --dbname=sportstickets --username=master

enter the password “master123” when prompted, then you can run the following SQL script to create the wrappers needed for DMS CDC replication:

BEGIN;

CREATE SCHEMA IF NOT EXISTS fnRenames;

CREATE OR REPLACE FUNCTION fnRenames.pg\_switch\_xlog() RETURNS pg\_lsn AS $$

SELECT pg\_switch\_wal(); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_xlog\_replay\_pause() RETURNS VOID AS $$

SELECT pg\_wal\_replay\_pause(); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_xlog\_replay\_resume() RETURNS VOID AS $$

SELECT pg\_wal\_replay\_resume(); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_current\_xlog\_location() RETURNS pg\_lsn AS $$

SELECT pg\_current\_wal\_lsn(); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_is\_xlog\_replay\_paused() RETURNS boolean AS $$

SELECT pg\_is\_wal\_replay\_paused(); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_xlogfile\_name(lsn pg\_lsn) RETURNS TEXT AS $$

SELECT pg\_walfile\_name(lsn); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_last\_xlog\_replay\_location() RETURNS pg\_lsn AS $$

SELECT pg\_last\_wal\_replay\_lsn(); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_last\_xlog\_receive\_location() RETURNS pg\_lsn AS $$

SELECT pg\_last\_wal\_receive\_lsn(); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_current\_xlog\_flush\_location() RETURNS pg\_lsn AS $$

SELECT pg\_current\_wal\_flush\_lsn(); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_current\_xlog\_insert\_location() RETURNS pg\_lsn AS $$

SELECT pg\_current\_wal\_insert\_lsn(); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_xlog\_location\_diff(lsn1 pg\_lsn, lsn2 pg\_lsn) RETURNS NUMERIC AS $$

SELECT pg\_wal\_lsn\_diff(lsn1, lsn2); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_xlogfile\_name\_offset(lsn pg\_lsn, OUT TEXT, OUT INTEGER) AS $$

SELECT pg\_walfile\_name\_offset(lsn); $$ LANGUAGE SQL;

CREATE OR REPLACE FUNCTION fnRenames.pg\_create\_logical\_replication\_slot(slot\_name name, plugin name,

temporary BOOLEAN DEFAULT FALSE, OUT slot\_name name, OUT xlog\_position pg\_lsn) RETURNS RECORD AS $$

SELECT slot\_name::NAME, lsn::pg\_lsn FROM pg\_catalog.pg\_create\_logical\_replication\_slot(slot\_name, plugin,

temporary); $$ LANGUAGE SQL;

ALTER user master SET search\_path to fnRenames, pg\_catalog, "$user", public;

COMMIT;

Details on the above script can be found here , You can also copy from below docs and change user name : <https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.PostgreSQL.html#CHAP_Source.PostgreSQL.v10>)

# Generate the CDC Data (Optional)

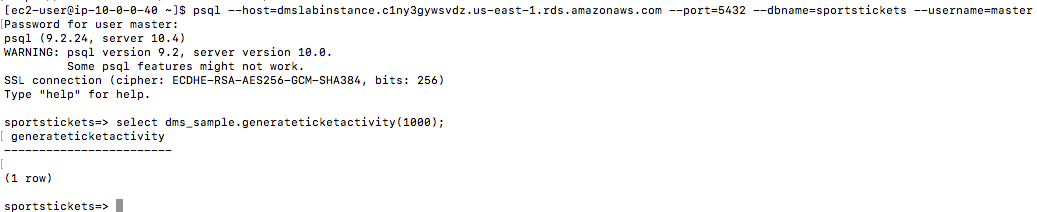
When you want to generate transactions to demonstrate DMS CDC (Change Data Capture) functionality you can execute the following commands:

psql --host=<instanceaddress> --port=5432 --dbname=sportstickets --username=master

enter the password “master123” when prompted, then you can execute the following within the psql command prompt (sportstickets=>)

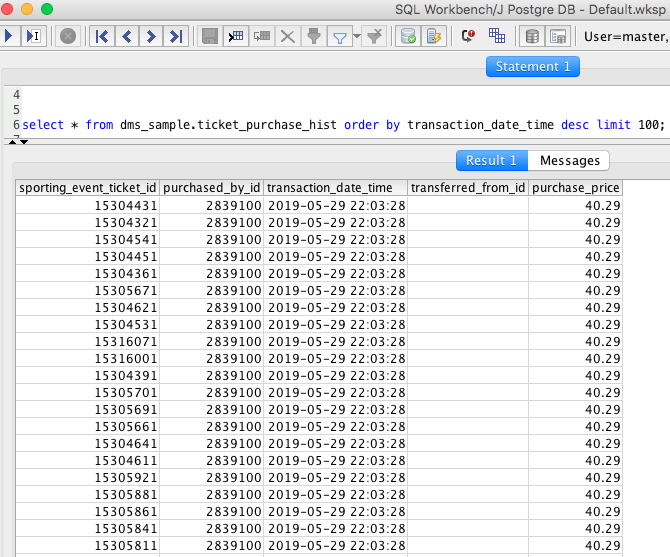
The following will generate 1000 ticket sales in batches of 1-6 tickets to randomly selected people for a random price (within a range.) A record of each transaction is recorded in the ticket\_purchase\_hist table:

select dms\_sample.generateticketactivity(1000);



Run following query in SQL Workbench to see changes in table:

select \* from dms\_sample.ticket\_purchase\_hist order by transaction\_date\_time desc limit 100;



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.

select dms\_sample.generatetransferactivity(100);

**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.

# Appendix: AWS CloudFormation Template

The AWS CloudFormation template is below. This template only works in the us-east-1 region.

Copy and paste this template into an instructor\_dmslab.json file on your computer and save it. Select that file in AWS CloudFormation for Step 3.

{

"AWSTemplateFormatVersion": "2010-09-09",

"Parameters" : {

"KeyName": {

"Description" : "Name of an existing EC2 KeyPair to enable SSH access to the instance",

"Type": "AWS::EC2::KeyPair::KeyName",

"ConstraintDescription" : "must be the name of an existing EC2 KeyPair in us-east-1 only."

}

},

"Resources": {

"dmsinstructorvpc": {

"Type": "AWS::EC2::VPC",

"Properties": {

"CidrBlock": "10.0.0.0/24",

"InstanceTenancy": "default",

"EnableDnsSupport": "true",

"EnableDnsHostnames": "true",

"Tags": [

{

"Key": "Name",

"Value": "DMSLabSourceDB"

}

]

}

},

"RDSSubNet": {

"Type": "AWS::EC2::Subnet",

"Properties": {

"CidrBlock": "10.0.0.0/28",

"AvailabilityZone": "us-east-1d",

"VpcId": {

"Ref": "dmsinstructorvpc"

},

"Tags": [

{

"Key": "Name",

"Value": "DMSLabRDS1"

}

]

}

},

"EC2SubNet": {

"Type": "AWS::EC2::Subnet",

"Properties": {

"CidrBlock": "10.0.0.32/28",

"AvailabilityZone": "us-east-1c",

"VpcId": {

"Ref": "dmsinstructorvpc"

},

"Tags": [

{

"Key": "Name",

"Value": "DMSLabEC2"

}

]

}

},

"RDSSubNet2": {

"Type": "AWS::EC2::Subnet",

"Properties": {

"CidrBlock": "10.0.0.16/28",

"AvailabilityZone": "us-east-1b",

"VpcId": {

"Ref": "dmsinstructorvpc"

},

"Tags": [

{

"Key": "Name",

"Value": "DMSLabRDS2"

}

]

}

},

"igw0887475a258f00277": {

"Type": "AWS::EC2::InternetGateway",

"Properties": {

"Tags": [

{

"Key": "Name",

"Value": "DMSLabIGW"

}

]

}

},

"dopt1cc25278": {

"Type": "AWS::EC2::DHCPOptions",

"Properties": {

"DomainName": "ec2.internal",

"DomainNameServers": [

"AmazonProvidedDNS"

]

}

},

"rtb0c3fae104a7b64456": {

"Type": "AWS::EC2::RouteTable",

"Properties": {

"VpcId": {

"Ref": "dmsinstructorvpc"

},

"Tags": [

{

"Key": "Name",

"Value": "DMSLabRT"

}

]

}

},

"instancei0f63b887480639040": {

"Type": "AWS::EC2::Instance",

"Properties": {

"DisableApiTermination": "false",

"InstanceInitiatedShutdownBehavior": "stop",

"EbsOptimized": "true",

"ImageId": "ami-04681a1dbd79675a5",

"InstanceType": "t3.2xlarge",

"KeyName": {"Ref" : "KeyName" },

"UserData" : {"Fn::Base64" : {"Fn::Join" : ["", [

"#!/bin/bash -xe\n",

"yum install -y postgresql\n",

"yum install -y git\n",

"yum update -y\n",

"cd /home/ec2-user\n",

"git clone https://github.com/aws-samples/aws-database-migration-samples.git\n"

]]}},

"Monitoring": "false",

"Tags": [

{

"Key": "Name",

"Value": "DMSLabEC2"

}

],

"NetworkInterfaces": [

{

"DeleteOnTermination": "true",

"Description": "Primary network interface",

"DeviceIndex": 0,

"SubnetId": {

"Ref": "EC2SubNet"

},

"PrivateIpAddresses": [

{

"PrivateIpAddress": "10.0.0.40",

"Primary": "true"

}

],

"GroupSet": [

{

"Ref": "sgDMSLabSG"

}

],

"AssociatePublicIpAddress": "true"

}

]

}

},

"rdsdmslabdb": {

"Type": "AWS::RDS::DBInstance",

"Properties": {

"AllocatedStorage": "20",

"AllowMajorVersionUpgrade": "false",

"AutoMinorVersionUpgrade": "true",

"DBInstanceClass": "db.t2.xlarge",

"DBInstanceIdentifier": "dmslabinstance",

"Port": "5432",

"PubliclyAccessible": "true",

"StorageType": "gp2",

"BackupRetentionPeriod": "7",

"MasterUsername": "master",

"MasterUserPassword": "master123",

"PreferredBackupWindow": "04:00-04:30",

"PreferredMaintenanceWindow": "sun:05:20-sun:05:50",

"DBName": "sportstickets",

"Engine": "postgres",

"EngineVersion": "10.4",

"LicenseModel": "postgresql-license",

"DBSubnetGroupName": {

"Ref": "dbsubnetdefaultdmsinstructorvpc"

},

"VPCSecurityGroups": [

{

"Ref": "sgrdslaunchwizard2"

}

],

"Tags": [

{

"Key": "workload-type",

"Value": "other"

}

]

}

},

"dbsubnetdefaultdmsinstructorvpc": {

"Type": "AWS::RDS::DBSubnetGroup",

"Properties": {

"DBSubnetGroupDescription": "Created from the RDS Management Console",

"SubnetIds": [

{

"Ref": "RDSSubNet"

},

{

"Ref": "EC2SubNet"

},

{

"Ref": "RDSSubNet2"

}

]

}

},

"sgDMSLabSG": {

"Type": "AWS::EC2::SecurityGroup",

"Properties": {

"GroupDescription": "launch-wizard-6 created 2018-08-29T15:10:01.302-04:00",

"VpcId": {

"Ref": "dmsinstructorvpc"

}

}

},

"sgrdslaunchwizard2": {

"Type": "AWS::EC2::SecurityGroup",

"Properties": {

"GroupDescription": "Created from the RDS Management Console: 2018/08/29 18:14:15",

"VpcId": {

"Ref": "dmsinstructorvpc"

},

"Tags": [

{

"Key": "Name",

"Value": "DMSLabRDS-SG"

}

]

}

},

"dbsgdefault": {

"Type": "AWS::RDS::DBSecurityGroup",

"Properties": {

"GroupDescription": "default"

}

},

"gw1": {

"Type": "AWS::EC2::VPCGatewayAttachment",

"Properties": {

"VpcId": {

"Ref": "dmsinstructorvpc"

},

"InternetGatewayId": {

"Ref": "igw0887475a258f00277"

}

}

},

"subnetroute1": {

"Type": "AWS::EC2::SubnetRouteTableAssociation",

"Properties": {

"RouteTableId": {

"Ref": "rtb0c3fae104a7b64456"

},

"SubnetId": {

"Ref": "RDSSubNet2"

}

}

},

"subnetroute2": {

"Type": "AWS::EC2::SubnetRouteTableAssociation",

"Properties": {

"RouteTableId": {

"Ref": "rtb0c3fae104a7b64456"

},

"SubnetId": {

"Ref": "RDSSubNet"

}

}

},

"subnetroute3": {

"Type": "AWS::EC2::SubnetRouteTableAssociation",

"Properties": {

"RouteTableId": {

"Ref": "rtb0c3fae104a7b64456"

},

"SubnetId": {

"Ref": "EC2SubNet"

}

}

},

"route1": {

"Type": "AWS::EC2::Route",

"Properties": {

"DestinationCidrBlock": "0.0.0.0/0",

"RouteTableId": {

"Ref": "rtb0c3fae104a7b64456"

},

"GatewayId": {

"Ref": "igw0887475a258f00277"

}

},

"DependsOn": "gw1"

},

"dchpassoc1": {

"Type": "AWS::EC2::VPCDHCPOptionsAssociation",

"Properties": {

"VpcId": {

"Ref": "dmsinstructorvpc"

},

"DhcpOptionsId": {

"Ref": "dopt1cc25278"

}

}

},

"ingress1": {

"Type": "AWS::EC2::SecurityGroupIngress",

"Properties": {

"GroupId": {

"Ref": "sgDMSLabSG"

},

"IpProtocol": "tcp",

"FromPort": "22",

"ToPort": "22",

"CidrIp": "0.0.0.0/0"

}

},

"ingress2": {

"Type": "AWS::EC2::SecurityGroupIngress",

"Properties": {

"GroupId": {

"Ref": "sgrdslaunchwizard2"

},

"IpProtocol": "tcp",

"FromPort": "5432",

"ToPort": "5432",

"SourceSecurityGroupId": {

"Ref": "sgDMSLabSG"

},

"SourceSecurityGroupOwnerId": "649225637812"

}

},

"ingress3": {

"Type": "AWS::EC2::SecurityGroupIngress",

"Properties": {

"GroupId": {

"Ref": "sgrdslaunchwizard2"

},

"IpProtocol": "tcp",

"FromPort": "5432",

"ToPort": "5432",

"CidrIp": "72.21.196.67/32"

}

},

"ingress4": {

"Type": "AWS::EC2::SecurityGroupIngress",

"Properties": {

"GroupId": {

"Ref": "sgrdslaunchwizard2"

},

"IpProtocol": "tcp",

"FromPort": "5432",

"ToPort": "5432",

"CidrIp": "0.0.0.0/0"

}

},

"egress1": {

"Type": "AWS::EC2::SecurityGroupEgress",

"Properties": {

"GroupId": {

"Ref": "sgDMSLabSG"

},

"IpProtocol": "-1",

"CidrIp": "0.0.0.0/0"

}

},

"egress2": {

"Type": "AWS::EC2::SecurityGroupEgress",

"Properties": {

"GroupId": {

"Ref": "sgrdslaunchwizard2"

},

"IpProtocol": "-1",

"CidrIp": "0.0.0.0/0"

}

}

},

"Description": "DMS Lab Instructor account",

"Metadata": {

"AWS::CloudFormation::Designer": {

"a79fb943-c167-4e59-8eda-911d4acc331f": {

"size": {

"width": 60,

"height": 60

},

"position": {

"x": 810,

"y": 390

},

"z": 1,

"embeds": []

}

}

}

}