

Contents

Introduction	2
Overview	2
The Cockroach Cluster – The Target for my DMS Migration	6
Create a User and Password	6
Create a DMS database.....	7
Create a ca.pem file for SSL communication in DMS.....	7
Create the Aurora Database Instance – The Source Database for my DMS Migration	7
Aurora Parameter Group	7
Create the DB Subnet Group	9
Create the Aurora PostgreSQL Database	10
Configure Secrets Manager.....	12
Store a Secret for Aurora	12
Store a Secret for CockroachDB	15
Create the IAM Policy and Role	18
Create a Policy.....	18
Create an IAM Role	19
Create a Windows EC2 Instance (Optional) and Install DBeaver.....	20
Connect to CockroachDB	21
Connect to Aurora PostgreSQL	23
Configure DMS and Start On Going Replication	25
Create the PostgreSQL Source Table	25
Create the CockroachDB Target Table	25
Create the Certificate.....	26
Create the Subnet groups	28
Create the Replication instance	29
Create the Endpoints	30
Create the Source Endpoint.....	31
Create the Target Endpoint.....	33
Create the Database migration task	34

Introduction

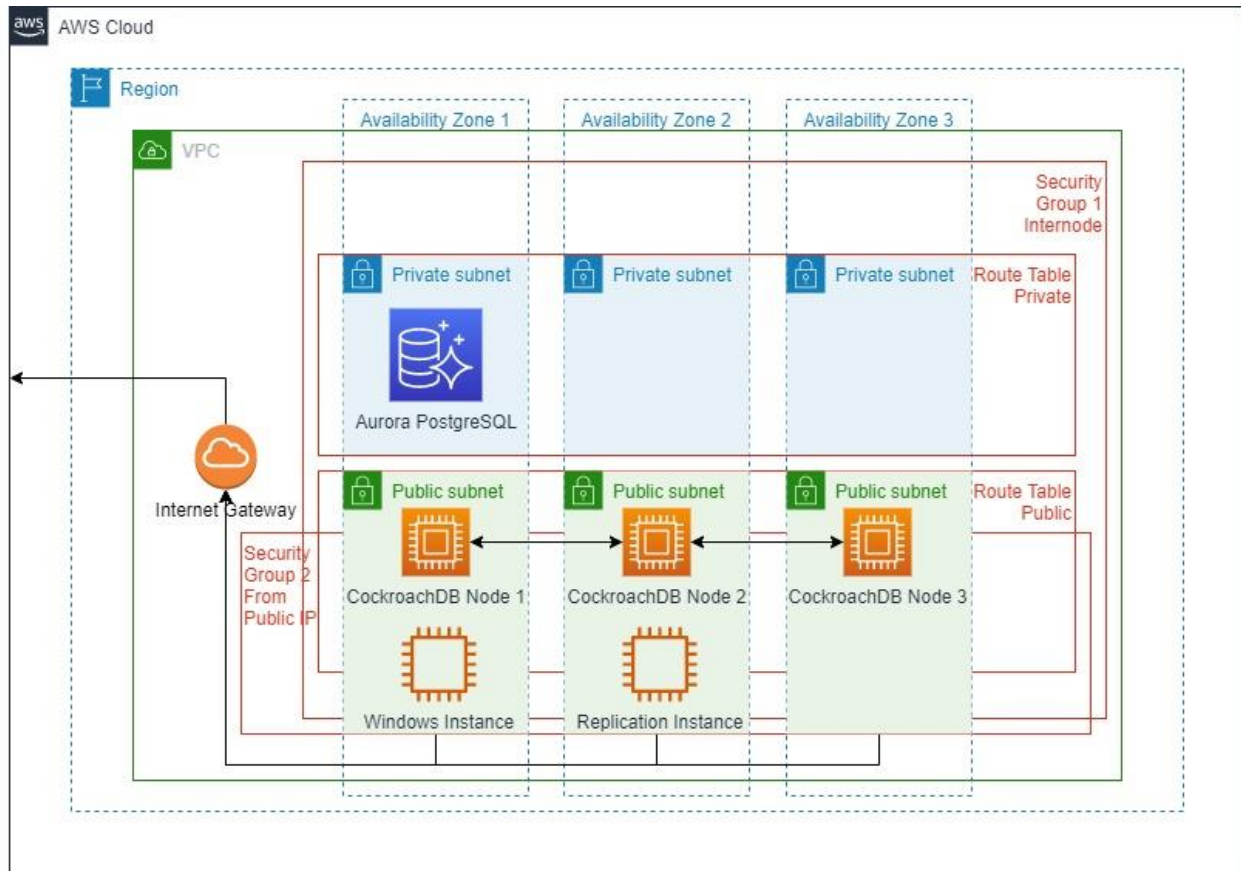
So many exciting things are always happening at Cockroach Labs, thanks in large part to our amazing engineering staff. Our release of 22.1 promises to continue to push our Distributed SQL forward with many new features and improvements. One of the most exciting things for me in 22.1 is the Preview Release of CockroachDB's integration with AWS Data Migration Service (DMS)! Stay tuned, many more exciting advancements with DMS are already in the works.

In this blog, I'm going to walk through (in tedious detail), how to set up SSL DMS between Aurora Postres and CockroachDB.

(By the way, unlike most of my peers at Cockroach Labs, I am working on a Windows Desktop, so you may have to map some of what I'm doing to your OS.)

Overview

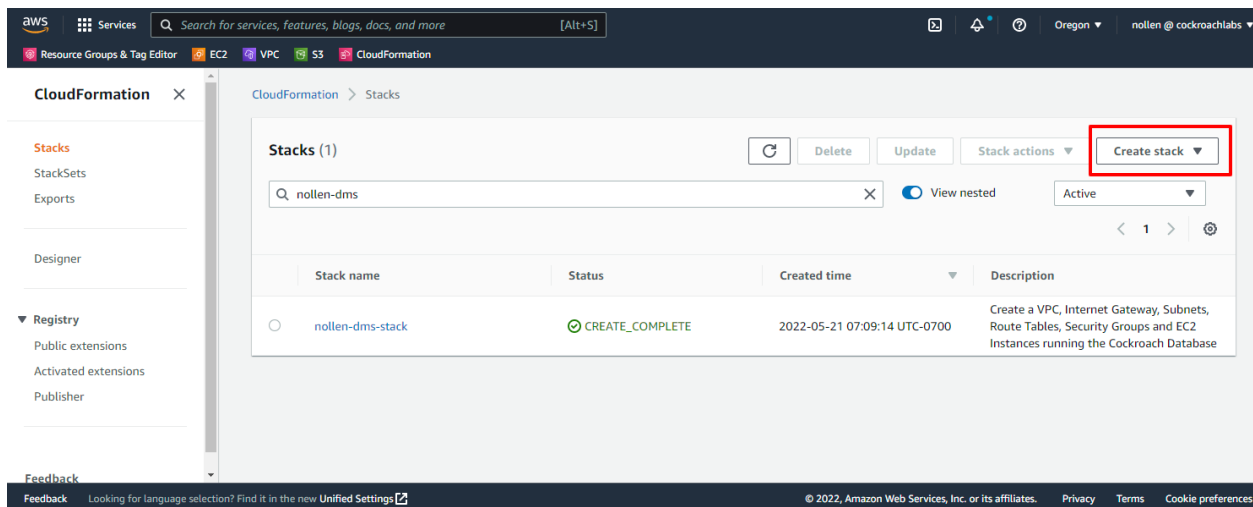
Let's start with an overview of what we're going to build:



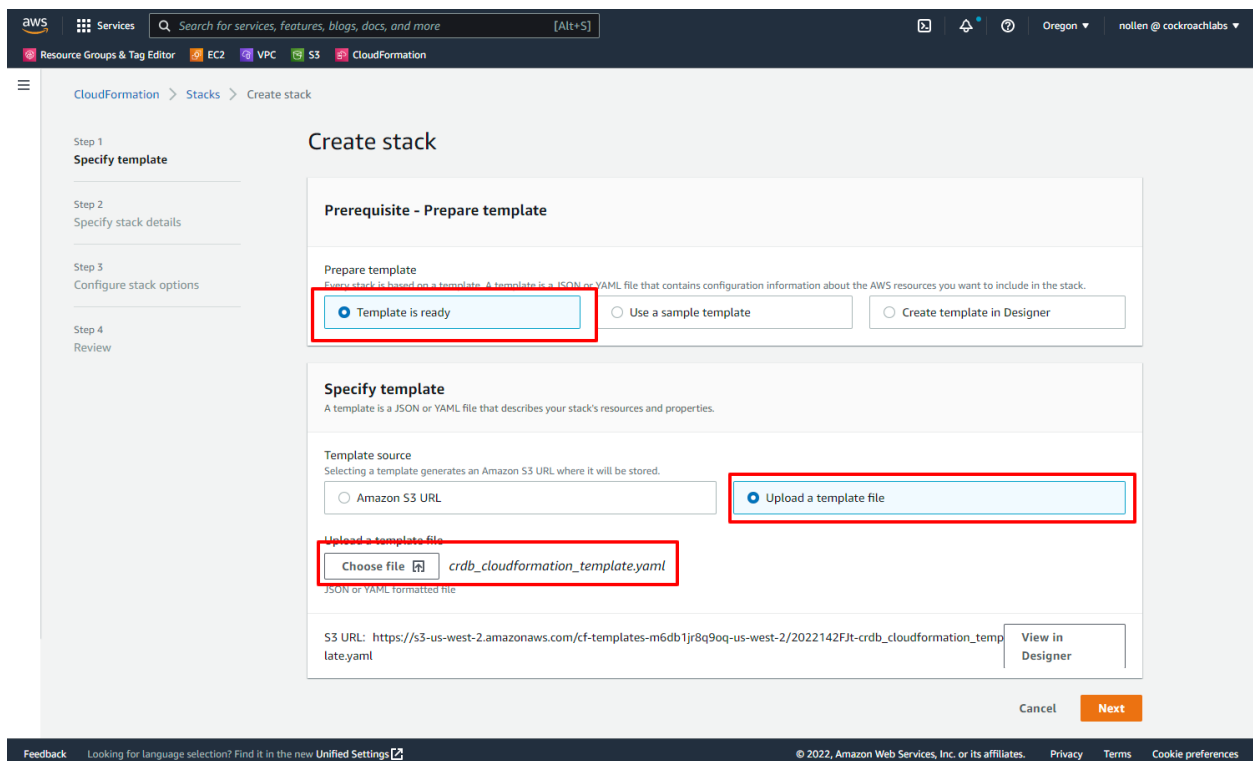
Our architecture includes a VPC where we will run both our Aurora Postgres Database in a private subnet along with a 3 node CockroachDB cluster in public subnets (please note, the CockroachDB could also be in the private subnets which in a production setting would be much more advisable). I'm using AWS Linux

In addition to the databases, I am also including a Windows server so that I can run things like DBeaver and connect to both databases as well the DMS Replication Instance. Since I placed the Aurora database in a private subnet, I will only be able to connect to the database from within the VPC so a Windows instance in the VPC make that very easy.

I have an AWS CloudFormation Script available [here](#), which will create VPC, Internet Gateway, Subnets, RouteTables, Security Groups EC2 Instances and the CockroachDB Cluster including all the necessary certs and locality settings – basically everything you see in the diagram above except for the Aurora Database, Windows Instance and DMS Replication Instance. To use the CloudFormation template, check out the [github](#) link. Briefly, navigate to the CloudFormation page in AWS, select create stack [With new resources (standard)]:



On the next page, choose to use the `crdb_cloudformation_template.yaml` available in the github repo.

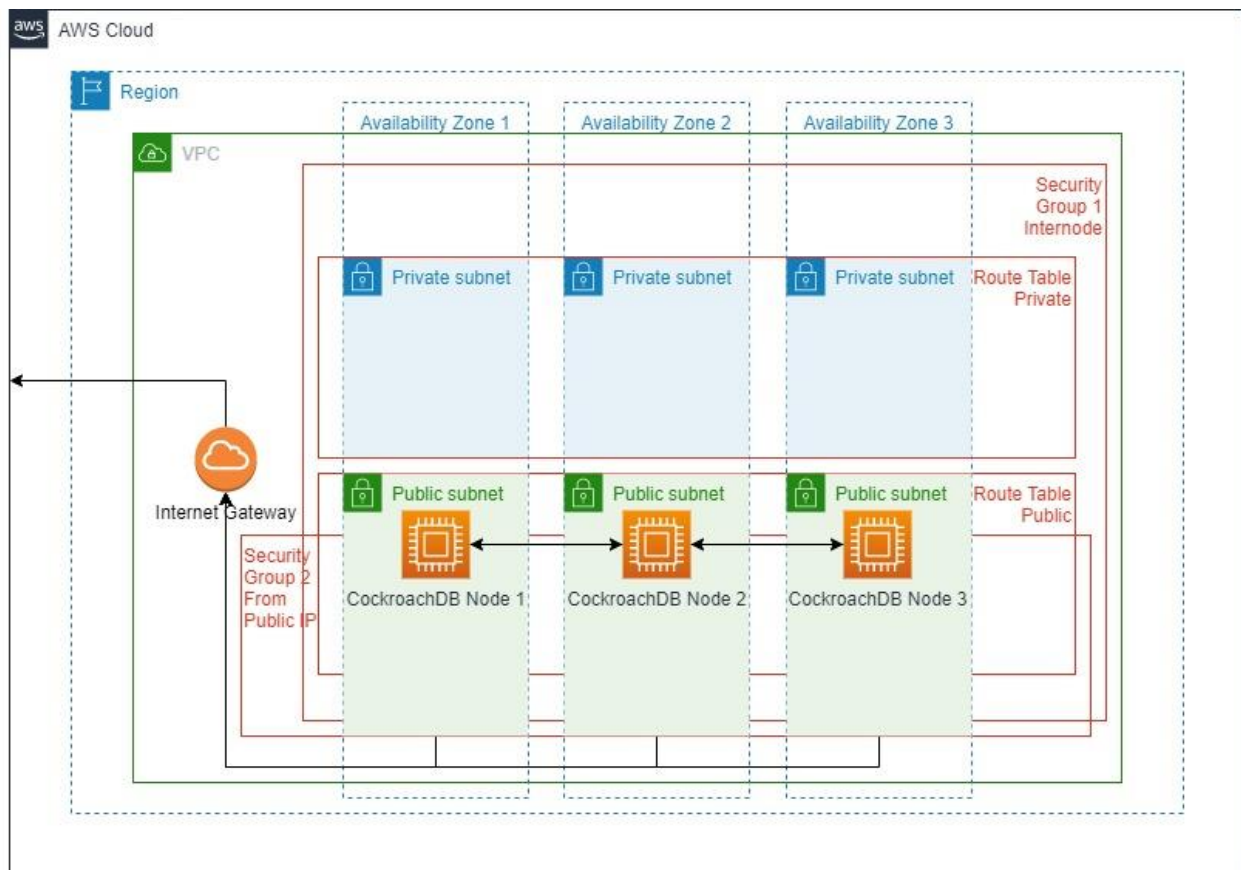


From there, you'll be able to input the parameters to create the architecture to your specification. The repo has information on the parameters, and what you'll need to set up before using the CloudFormation Template (such as a Key-Pair for the region in which you are working). Here are the parameters I used to create this CockroachDB Cluster:

CRDBAMIID	ami-00f7e5c52c0f43726
ClusterName	nollen-dms-target-cluster

CockroachVersion	21.2.10
ExistingJoinString	NONE
Installpsql	YES
InstanceType	m4.large
KeyPairName	nollen-cockroach-us-west-2-kp01
MyIP	72.132.195.192
NumberOfNodes	3
RunInit	YES
VpcAzs	us-west-2a,us-west-2b,us-west-2c
VpcCidrParameter	192.168.5.0/24
VpcName	nollen-dms-vpc-us-west-2

At this point, I'm going to assume you have the following:



- The VPC in the AWS region of your choice
- At least 3 Availability Zones (AZ) in the VPC
- 3 public and 3 private subnets (1 public and 1 private subnet in each AZ)
- Route Tables for the public and private subnets. The public route table should include an internet gateway so that instances in the public subnet can reach the internet).
- Security Groups. I have 2 security groups, one which allows all internode communicate within the VPC, and the other which allows only my IP to access instances in the public subnets on very

specific ports (3389 for RDP to my windows box, 8080 for access to the CockroachDB UI, 26257 for access to the CockroachDB and 22 for SSH access)

- CockroachDB Cluster: a 3 node CockroachDB Cluster running in 3 AZs

A few free tools that I will be using to access the different instances:

- putty
- winscp

The Cockroach Cluster – The Target for my DMS Migration

Here is a snapshot of my CockroachDB Cluster as created by my CloudFormation Template.

```
[ec2-user@ip-192-168-5-132 ~]$ cockroach node status
```

id	build	locality	is_available	is_live
1	v22.1.0-beta.4-234-g5c9bf148ea	region=us-west-2,zone=us-west-2c	true	true
2	v22.1.0-beta.4-234-g5c9bf148ea	region=us-west-2,zone=us-west-2b	true	true
3	v22.1.0-beta.4-234-g5c9bf148ea	region=us-west-2,zone=us-west-2a	true	true

(3 rows)

The database nodes are running Amazon Linux 2 Kernel 5.10 AMI 2.0.20211201.0 x86_64 HVM gp2 on m4.large instance types.

This is a 3 node cluster running CockroachDB Version 22.1.

Create a User and Password

Login to the cluster:

```
[ec2-user@ip-192-168-5-4 ~]$ cockroach sql
#
# Welcome to the CockroachDB SQL shell.
# All statements must be terminated by a semicolon.
# To exit, type: \q.
#
# Server version: CockroachDB CCL v22.1.0-rc.1 (x86_64-pc-linux-gnu, built
2022/05/09 14:20:38, go1.17.6) (same version as client)
# Cluster ID: d491e7b3-6cf7-43b2-a507-49ed1470710a
#
# Enter \? for a brief introduction.
#
root@:26257/defaultdb>
```

And create a user with admin privileges

```
root@:26257/defaultdb> create user ron with password "<redacted>";
CREATE ROLE
```

```
Time: 178ms total (execution 177ms / network 0ms)
root@:26257/defaultdb> grant admin to ron;
GRANT
```

```
Time: 92ms total (execution 92ms / network 0ms)
```

Create a DMS database

```
root@:26257/defaultdb> create database dms;  
CREATE DATABASE
```

```
Time: 22ms total (execution 21ms / network 0ms)
```

Create a ca.pem file for SSL communication in DMS.

After creating the Cockroach Cluster, you'll need to generate a ca.pem file for SSL communication in DMS. Unfortunately, DMS will not work with ".cert" (certificates), so we must convert the cert to a ".pem" or base-64 encoding of the certificate.

To create the ".pem", use putty (or the tool of your choice) to log onto one of the nodes of your CockroachDB cluster, navigate to the "certs" directory and issue the following:

```
[ec2-user@ip-192-168-5-4 certs]$ openssl x509 -in ca.crt -out ca.pem -outform PEM
```

Now, you'll need to use winscp (or the tool of your choice) to download that file. We'll upload this file to DMS later in order to establish SSL connectivity.

While you're at it, also download the ca.crt if you plan on using DBeaver to connect to your secure CockroachDB Cluster.

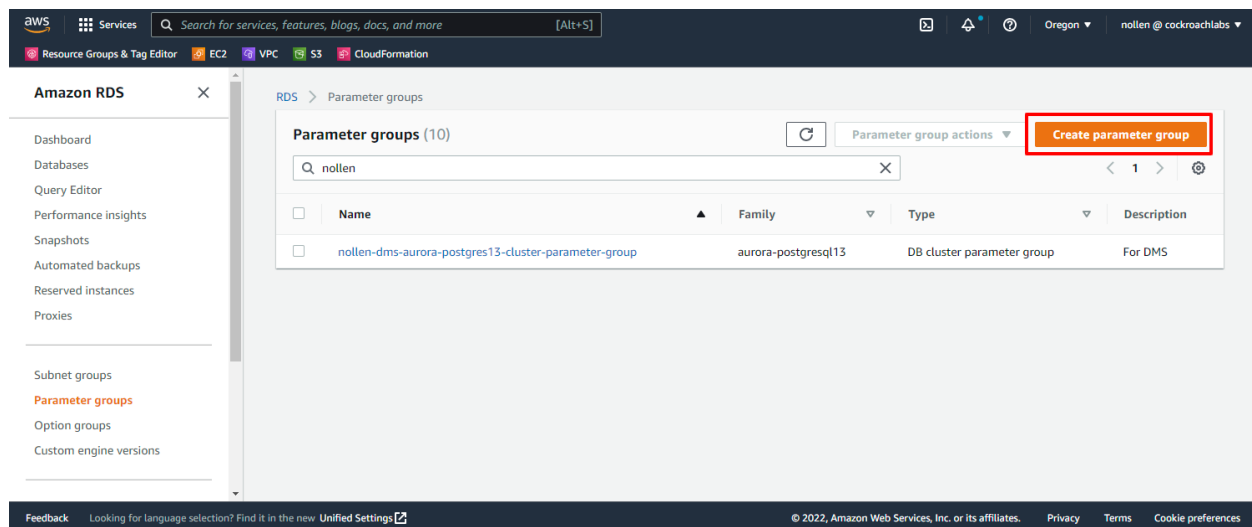
Create the Aurora Database Instance – The Source Database for my DMS Migration

Chances are you already have an Aurora PostgreSQL database, but there may be some modifications you may need to make to enable DMS to use the database as a source for a migration or replication. We'll walk through those steps now.

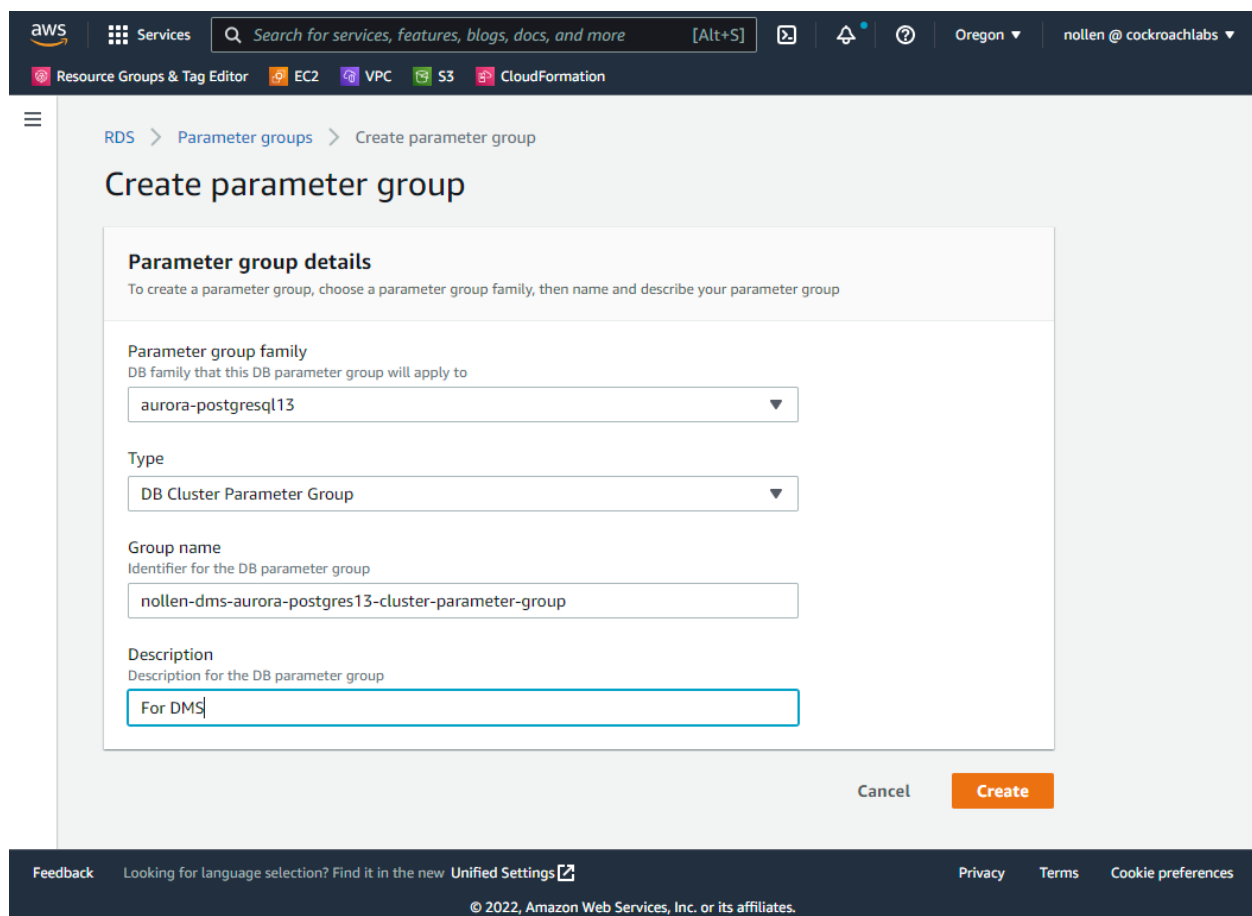
Aurora Parameter Group

In order for Aurora to be a source for DMS, we need to enable replication. To do that, we'll need to create a custom cluster parameter group. AWS Documentation on the requirements are available [here](#).

We'll create the parameter group by navigating to AWS RDS, choose "Parameter Groups" from the list on the left and then "Create Parameter Group":

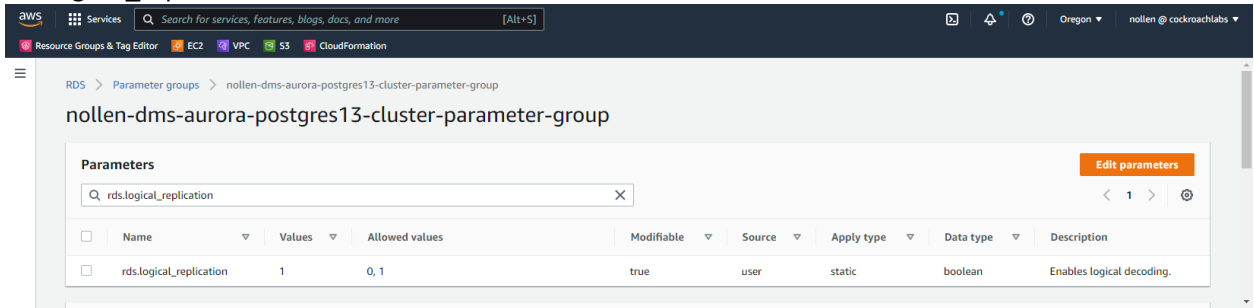


We need to create a parameter group for the version of Aurora PostgreSQL we going to create and the type is “DB Cluster Parameter Group”:

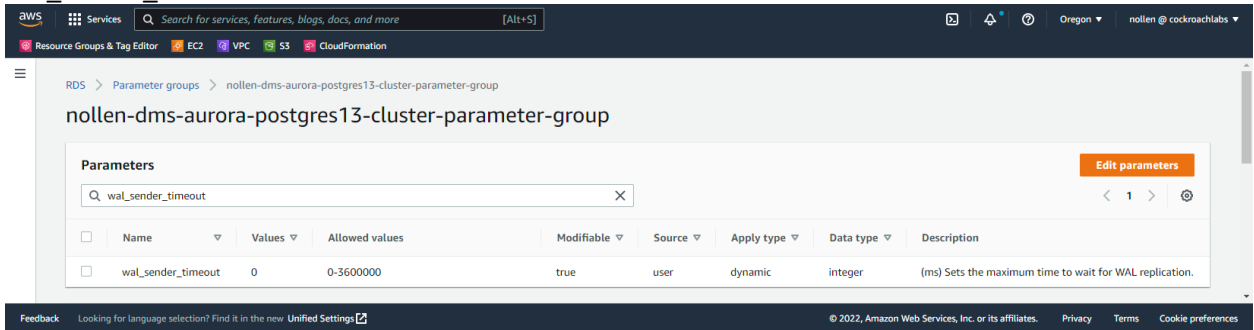


There are at least 2 parameters we need to change:

- `rds.logical_replication` which must be set to “1”



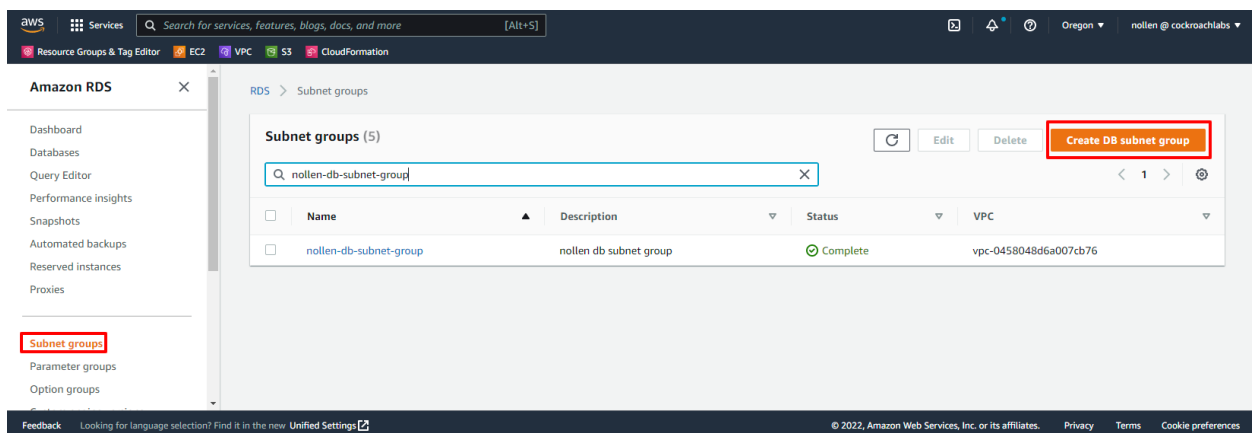
- `wal_sender_timeout` which should be set to 0 in order to disable it.



Again, please reference the AWS Documentation mentioned above for the background on these parameters and why they must be set to the values shown here.

Create the DB Subnet Group

Prior to creating the Aurora PostgreSQL database, we should also create a DB Subnet Group. To do that, navigate to RDS, select “Subnet groups” from the left hand panel and select “Create DB subnet group”.



When creating your subnet groups, but sure to select the VPC we are using for this implementation, select multiple Availability Zones and then select the Private Subnets from the Drop Down List Box. This will ensure that the Aurora Database and any replicas will be place in the correct subnets.

The screenshot shows the AWS Management Console interface for the 'nollen-dms-subnet-group'. The left sidebar contains the 'AWS DMS' menu with options like Dashboard, Database migration tasks, Replication instances, Endpoints, Certificates, Subnet groups (selected), Events, Event subscriptions, and Notifications. The main content area shows the details of the 'nollen-dms-subnet-group'. The 'Details' section includes the ARN (arn:aws:dms:us-west-2:541263489771:subgrp:nollen-dms-subnet-group), Status (Complete), VPC (vpc-0458048d6a007cb76), and Description (nollen-dms-subnet-group). The 'Subnets' section shows a list of three subnets: subnet-0d04652ed28189508 (Active, us-west-2c), subnet-0fe83cc1bd45bf17d (Active, us-west-2b), and subnet-04da8d0cdf533c8c3 (Active, us-west-2a). The 'Tags' section shows one tag with the key 'owner' and value 'nollen'.

Name	Status	Availability Zone
subnet-0d04652ed28189508	Active	us-west-2c
subnet-0fe83cc1bd45bf17d	Active	us-west-2b
subnet-04da8d0cdf533c8c3	Active	us-west-2a

Key	Value
owner	nollen

Create the Aurora PostgreSQL Database

I won't go into too much depth on creating the RDS Aurora PostgreSQL database other than to highlight a couple of important things (a summary of my selections when I created the database are shown below).

In the "Engine options" be sure to select "Amazon Aurora" and from the Edition Radio buttons be sure to select "Amazon Aurora PostgreSQL – Compatible Edition". For the Version, select "Aurora PostgreSQL (Compatible with PostgreSQL13.4)". I would suspect that other versions (10 and higher) would also work.

In the "Connectivity" section, be sure to select the correct VPC (the one we're currently working in) and in the "Subnet group", select the DB subnet group we created in the above step. For "Public Access" select "No". For VPC security Group, select "Choose existing" and use the 2 security groups we've been using throughout this blog.

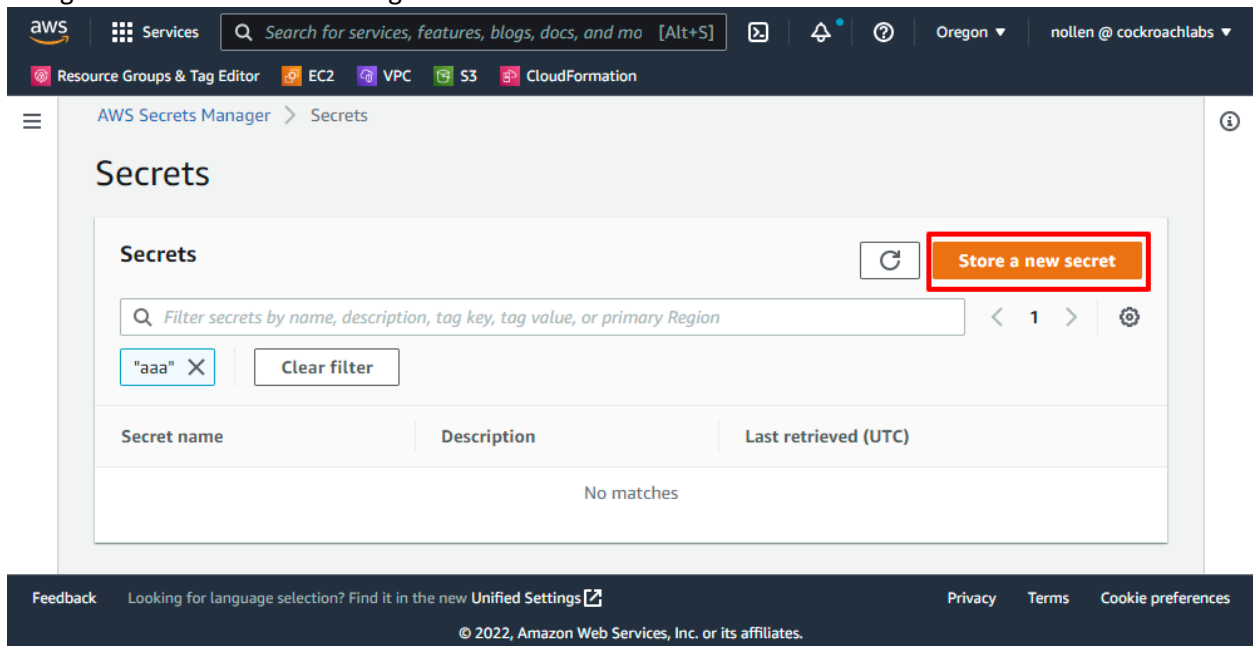
Under "Additional Configuration" you can select "defaultdb" as the "initial database name" to make things easier.

Configure Secrets Manager

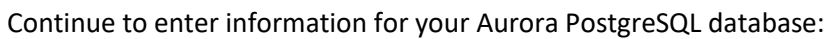
To make configuration with DMS easier, we're going to create a couple of secrets in Secret manager for connecting to our databases (both Aurora and CockroachDB).

Store a Secret for Aurora

Navigate to "AWS Secrets Manager" and select "Store a new Secret"



Enter the information requested on page 1 of "Store a New Secret":



Continue to enter information for your Aurora PostgreSQL database:

The screenshot shows the AWS console interface for the 'Store a new secret' wizard in the Secrets Manager service. The breadcrumb trail is 'AWS Secrets Manager > Secrets > Store a new secret'. The left sidebar shows the progress: Step 1 (Choose secret type), Step 2 (Configure secret - active), Step 3 (Configure rotation - optional), and Step 4 (Review). The main content area is titled 'Store a new secret' and contains several sections: 'Secret name and description' with a 'Secret name' field containing '/nollen/dms/aurora' and a 'Description' field containing 'Aurora secret'; 'Tags - optional' with an 'Add' button; 'Resource permissions - optional' with an 'Edit permissions' button; and 'Replicate secret- optional' with a 'Replicate secret' button. At the bottom, there are 'Cancel', 'Previous', and 'Next' buttons. The footer includes a feedback link, a language selection notice, and copyright information for Amazon Web Services, Inc. or its affiliates.

aws Services Search for services, features, blogs, docs, and more [Alt+S] Oregon nollen@cockroachlabs

Resource Groups & Tag Editor EC2 VPC S3 CloudFormation

Step 1 Choose secret type

Step 2 **Configure secret**

Step 3 Configure rotation - optional

Step 4 Review

AWS Secrets Manager > Secrets > Store a new secret

Store a new secret

Secret name and description [Info](#)

Secret name
A descriptive name that helps you find your secret later.

Secret name must contain only alphanumeric characters and the characters /_+-.@-

Description - optional

Maximum 250 characters.

Tags - optional








Resource permissions - optional [Info](#)
Add or edit a resource policy to access secrets across AWS accounts.

Replicate secret- optional
Create read-only replicas of your secret in other Regions. Replica secrets incur a charge. [Learn more](#) in the User Guide.

Feedback Looking for language selection? Find it in the new Unified Settings [\[?\]](#) © 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

You can leave the defaults on page 3 and select “next”. And then click “store” on the “Review” page.

When the secret is created, it will automatically contain all of the necessary connection information including the Username and Password you entered:

Secret value Info Retrieve and view the secret value.		<button>Close</button>	<button>Edit</button>
Key/value	Plaintext		
Secret key	Secret value		
username	 postgres		
password	 		
engine	 postgres		
host	 nollen-dms-source-aurora-db.cluster-c24vvqv1mozy.us-west-2.rds.amazonaws.com		
port	 5432		
dbClusterIdentifier	 nollen-dms-source-aurora-db		

Secrets manager was able to get this information for you directly from RDS without you having to enter it.

Store a Secret for CockroachDB

The process for storing a Secret for CockroachDB will basically be the same as above, except that you'll need to provide the Server Address, Database name and Port. To get started, select "Credentials for other database" under "Secret Type", choose PostgreSQL as the database and then enter the requested information. DMS will use this to create a connection to your Cockroach Database.

aws Services Search for services, features, blogs, docs, and more [Alt+S] Oregon nollen@cockroachlabs

Resource Groups & Tag Editor EC2 VPC S3 CloudFormation

Step 1 Choose secret type
Step 2 Configure secret
Step 3 Configure rotation - optional
Step 4 Review

Store a new secret

Secret type Info

☐ Credentials for Amazon RDS database ☐ Credentials for Amazon DocumentDB database ☐ Credentials for Amazon Redshift cluster

☒ Credentials for other database ☐ Other type of secret
API key, OAuth token, other.

Credentials Info

User name
ron

Password
.....

☐ Show password

Encryption key Info
You can encrypt using the KMS key that Secrets Manager creates or a customer managed KMS key that you create.
aws/secretsmanager

Database Info

☐ MariaDB ☐ MySQL ☒ PostgreSQL ☐ ORACLE DATABASE ☐ SQL Server

Server address
192.168.5.4

Database name
dms

Port
26257

Cancel **Next**

Feedback Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

This is what my completed secret for CockroachDB looks like:

Services

Search for services, features, blogs, docs, and more

[Alt+S]

Resource Groups & Tag Editor

EC2

VPC

S3

CloudFormation

Oregon

nollen @ cockroachlabs

AWSSecretsManagerSecrets/nollenr/dms/cockroachdb

Replicate secret to other Regions

Secret details

Actions

Encryption key

aws/secretsmanager

Secret name

/nollenr/dms/cockroachdb

Secret ARN

arn:aws:secretsmanager:us-west-2:541263489771:secret:/nollenr/dms/cockroachdb-SCpuIM

Secret description

-

Tags

Edit tags

Find by key or value...

Key	Value
owner	nollen

Secret value

Close

Edit

Retrieve and view the secret value.

Key/value

Plaintext

Secret key	Secret value
username	ron
password	
engine	postgres
host	192.168.5.4
port	26257
dbname	dms

Rotation configuration

Edit rotation

Rotation status

Disabled

Rotation schedule

-

Resource permissions - optional

Edit permissions

Add or edit a resource policy to access secrets across AWS accounts.

Sample code

Use these code samples to retrieve the secret in your application.

Java

JavaV2

JavaScript

C#

Python3

Ruby

Go

```
1 // Use this code snippet in your app.
2 // If you need more information about configurations or implementing the sample code, visit the AWS docs:
3 // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/java-dg-samples.html#prerequisites
4
5 public static void getSecret() {
6     String secretName = "arn:aws:secretsmanager:us-west-2:541263489771:secret:/nollenr/dms/cockroachdb-SCpuIM";
7     String region = "us-west-2";
8
9     // Create a Secrets Manager client
10    AWSSecretsManagerClientBuilder standard()
11    .withRegion(region)
12    .build();
13
14    // In this sample we only handle the specific exceptions for the 'GetSecretValue' API.
15    // See https://docs.aws.amazon.com/secretsmanager/latest/apireference/API_GetSecretValue.html
16    // We rethrow the exception by default.
17 }
```

Sample code

Use these code samples to retrieve the secret in your application.

Java

JavaV2

JavaScript

C#

Python3

Ruby

Go

```
1 // Use this code snippet in your app.
2 // If you need more information about configurations or implementing the sample code, visit the AWS docs:
3 // https://docs.aws.amazon.com/sdk-for-java/v1/developer-guide/java-dg-samples.html#prerequisites
4
5 public static void getSecret() {
6     String secretName = "arn:aws:secretsmanager:us-west-2:541263489771:secret:/nollenr/dms/cockroachdb-SCpuIM";
7     String region = "us-west-2";
8
9     // Create a Secrets Manager client
10    AWSSecretsManagerClientBuilder standard()
11    .withRegion(region)
12    .build();
13
14    // In this sample we only handle the specific exceptions for the 'GetSecretValue' API.
15    // See https://docs.aws.amazon.com/secretsmanager/latest/apireference/API_GetSecretValue.html
16    // We rethrow the exception by default.
17 }
```

Feedback

Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Create the IAM Policy and Role

We need to create an IAM Policy and Role in order for DMS to be able to use the secrets we just created.

Create a Policy

Navigate to Identity and Access Management (IAM) and select “Policies” from the “Access Management” section on the left. Choose create policy

A policy (which I named “nollen-read-secrets-policy”) which works is:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:GetResourcePolicy",
        "secretsmanager:GetSecretValue",
        "secretsmanager:DescribeSecret",
        "secretsmanager:ListSecretVersionIds"
      ],
      "Resource": [
        "arn:aws:secretsmanager:us-west-2:541263489771:secret:/nollen/dms/aurora-WaQe5W",
        "arn:aws:secretsmanager:us-west-2:541263489771:secret:/nollenr/dms/cockroachdb-SCpuiM"
      ]
    },
    {
      "Sid": "VisualEditor1",
      "Effect": "Allow",
      "Action": "secretsmanager:GetRandomPassword",
      "Resource": "*"
    }
  ]
}
```

The screenshot shows the AWS IAM console interface. At the top, there's a navigation bar with the AWS logo, 'Services' menu, a search bar, and user information 'nollen @ cockroachlabs'. Below this is a sub-header 'IAM > Policies'. The main content area is titled 'Policies (1028) Info' and includes a description: 'A policy is an object in AWS that defines permissions.' There are buttons for 'Create Policy' and 'Actions'. A search bar contains the text 'Filter policies by property or policy name and press enter' with '1 match' results. Below the search bar, there are filter tags for 'nollen' and 'nollen-read', and a 'Clear filters' button. A table lists the policy 'nollen-read-secrets-policy' with columns for Policy name, Type (Customer managed), Used as (Permissions policy (1)), and Description (This policy will be attached). The policy details are shown below the table, including a 'Copy' button and an 'Edit' button. The JSON policy document is displayed in a code editor with line numbers 1 through 25. The policy allows actions like 'secretsmanager:GetResourcePolicy', 'secretsmanager:GetSecretValue', 'secretsmanager:DescribeSecret', and 'secretsmanager:ListSecretVersionIds' on specific resources, and also allows 'secretsmanager:GetRandomPassword' on all resources.

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "VisualEditor0",
6       "Effect": "Allow",
7       "Action": [
8         "secretsmanager:GetResourcePolicy",
9         "secretsmanager:GetSecretValue",
10        "secretsmanager:DescribeSecret",
11        "secretsmanager:ListSecretVersionIds"
12      ],
13      "Resource": [
14        "arn:aws:secretsmanager:us-west-2:541263489771:secret:/nollen/dms/aurora-WaQe5W",
15        "arn:aws:secretsmanager:us-west-2:541263489771:secret:/nollenr/dms/cockroachdb-SCpuiM"
16      ]
17    },
18    {
19      "Sid": "VisualEditor1",
20      "Effect": "Allow",
21      "Action": "secretsmanager:GetRandomPassword",
22      "Resource": "*"
23    }
24  ]
25 }
```

Create an IAM Role

Now we need to attach the policy we just created to a role. To do that, navigate to IAM and select "Roles" from the panel on the left. Choose "Create role" (which I named "nollen-read-secrets-role") and attach the policy we created above.

The screenshot displays the AWS IAM console interface. The top navigation bar includes the AWS logo, a search bar, and various service icons. The left-hand navigation menu is expanded, showing 'Identity and Access Management (IAM)' with sub-sections like 'Access management' and 'Access reports'. The main content area is titled 'nollen-read-secrets-role' and includes a 'Delete' button. Below the title, a description states: 'Allows Database Migration Service to call AWS services on your behalf.' The 'Summary' section shows the role's creation date (May 21, 2022, 14:01 UTC-07:00) and last activity (2 hours ago). The 'Permissions' tab is selected, showing a table of permissions policies. The table has columns for 'Policy name', 'Type', and 'Description'. One policy is listed: 'nollen-read-secrets-policy' (Customer managed). Below the table, there is a section for 'Permissions boundary - (not set)' and a 'Generate policy based on CloudTrail events' section.

Identity and Access Management (IAM)

Search IAM

Dashboard

Access management

- User groups
- Users
- Roles**
- Policies
- Identity providers
- Account settings

Access reports

- Access analyzer
- Archive rules
- Analizers
- Settings
- Credential report
- Organization activity
- Service control policies (SCPs)

nollen-read-secrets-role Delete

Allows Database Migration Service to call AWS services on your behalf.

Summary Edit

Creation date
May 21, 2022, 14:01 (UTC-07:00)

ARN
[arn:aws:iam::541263489771:role/nollen-read-secrets-role](#)

Last activity
2 hours ago

Maximum session duration
1 hour

Permissions | Trust relationships | Tags (1) | Access Advisor | Revoke sessions

Permissions policies (1)
You can attach up to 10 managed policies.

Simulate Remove Add permissions

<input type="checkbox"/>	Policy name ↗	Type	Description
<input type="checkbox"/>	nollen-read-secrets-policy	Customer managed	This policy will be attached to a policy that will be used by t...

Permissions boundary - (not set)
Set a permissions boundary to control the maximum permissions this role can have. This is not a common setting but can be used to delegate permission management to others.

Set permissions boundary

Generate policy based on CloudTrail events

You can generate a new policy based on the access activity for this role, then customize, create, and attach it to this role. AWS uses your CloudTrail events to identify the services and actions used and generate a policy. [Learn more](#)

Share your [feedback](#) and help us improve the policy generation experience.

Generate policy

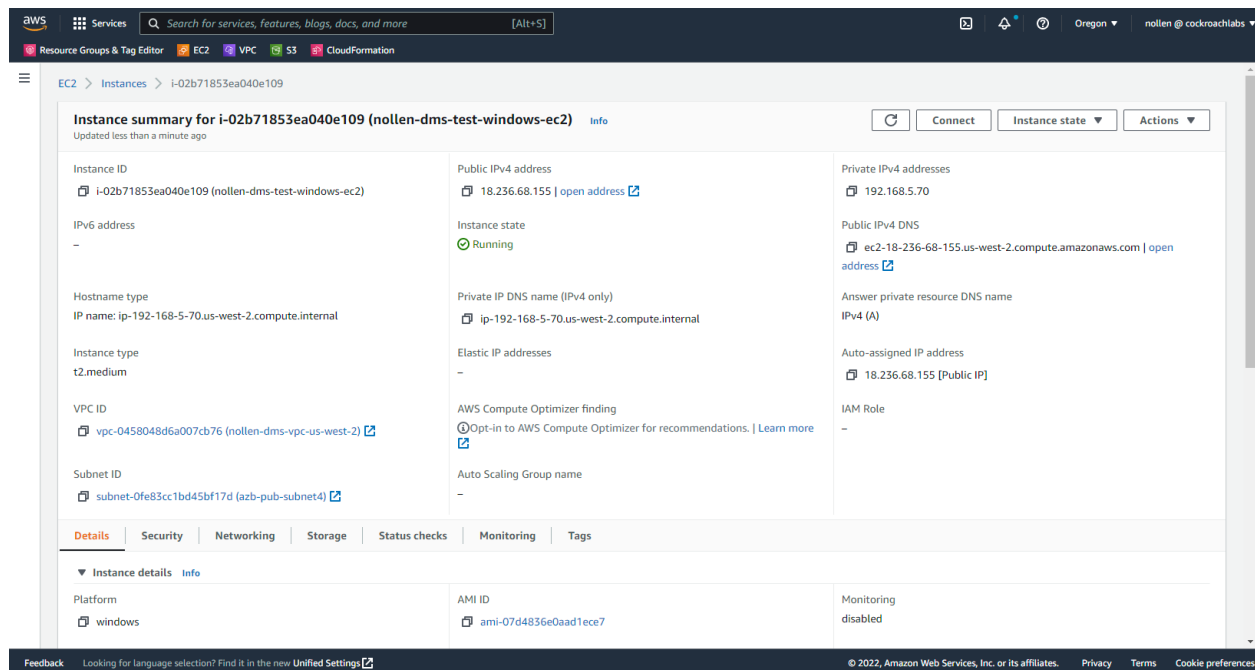
No requests to generate a policy in the past 7 days.

[Feedback](#) Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Create a Windows EC2 Instance (Optional) and Install DBeaver.

I created a windows instance in a public subnet (in any of the AZs of my VPC). I use this instance to install DBeaver and access my databases. I won't go into the details of creating an EC2 instance here, but below are some of the details of my instance:



Once I had my Windows instance and connected to it via RDP (use the “Connect” button on the EC2 Instance Screen to get the Username and Password), I installed DBeaver.

Connect to CockroachDB

I’m not going to go into installing DBeaver, but I think it might be helpful for me to explain how I connect to my Cockroach Instance from the utility.


When you create a connection in DBeaver for CockroachDB, be sure to select “Cockroach” as the database type.

In the “Server” section, you can choose any one of the 3 private IP addresses of the CockroachDB nodes; be sure to use Port 26257 and the DMS database.

Use the username and password we created above in the “Authentication” section.

Connection "dms" configuration

Connection settings
CockroachDB connection settings




▼ Connection settings
Initialization
Shell Commands
Client identification
Transactions
General
Metadata
Errors and timeouts
Data editor
SQL Editor

Main CockroachDB Driver properties SSH Proxy SSL

Server
Host: 192.168.5.4 Port: 26257
Database: dms

Authentication
Authentication: Database Native
Username: ron
Password: Save password locally

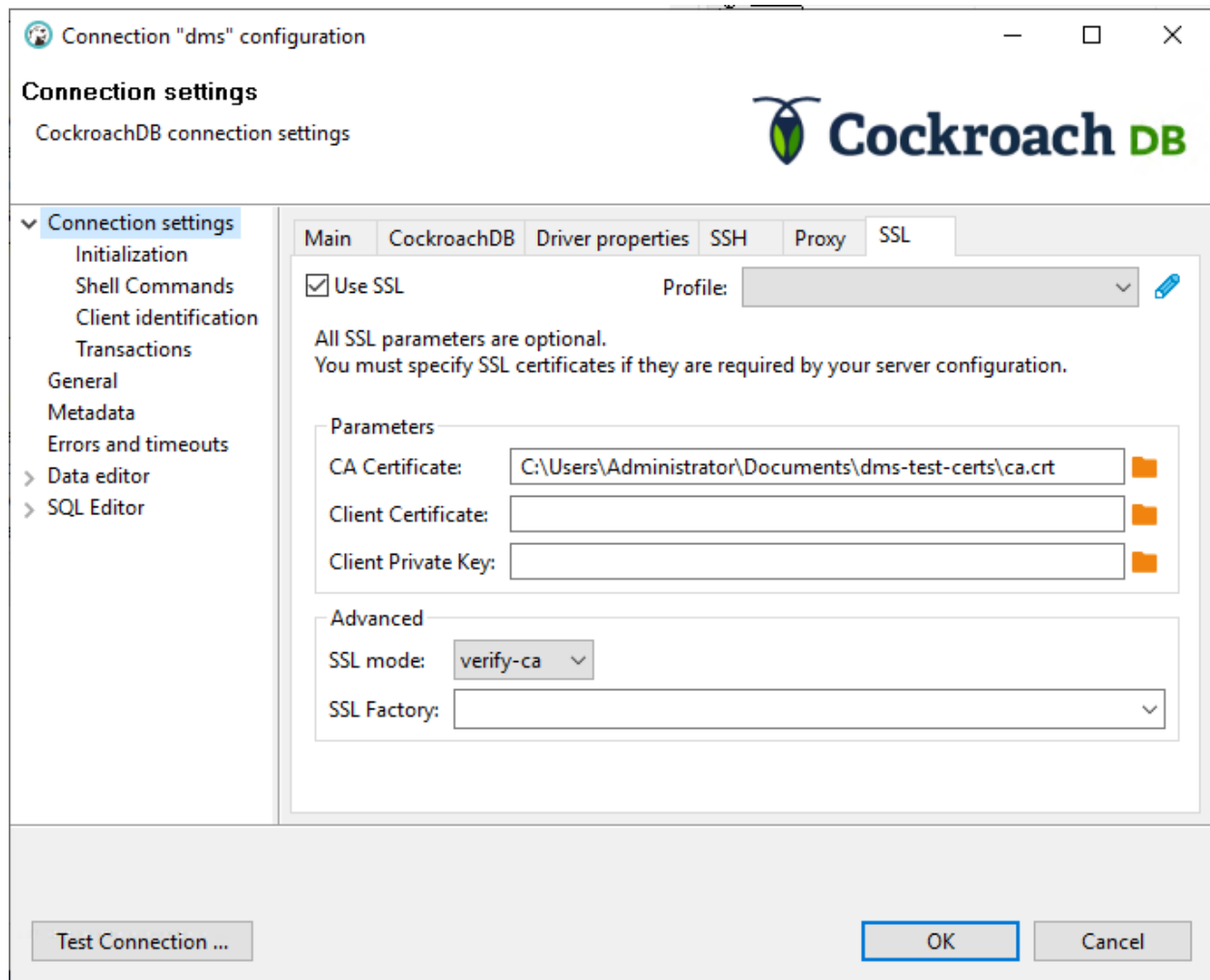
Advanced
Session role: Local Client: PostgreSQL Binaries

 You can use variables in connection parameters.

Driver name: CockroachDB [Edit Driver Settings](#)

Test Connection ... OK Cancel

On the SSL Tab of the connection modal, be sure to use the ca.crt file we downloaded earlier as the parameter for "CA Certificate". In the "advanced" section, choose "verify-ca" from the "SSL mode:" drop down list box.



Be sure to test your connection.

[Connect to Aurora PostgreSQL](#)

To connect to the Aurora PostgreSQL database, you'll use the host provided by AWS when you created the instance and port 5432. You'll also use the master user and password you supplied when you created the instance.

Connection "Aurora DMS DB" configuration

Connection settings

PostgreSQL connection settings

PostgreSQL

Connection settings

- Initialization
- Shell Commands
- Client identification
- Transactions
- General
- Metadata
- Errors and timeouts
- > Data editor
- > SQL Editor

Main PostgreSQL Driver properties SSH Proxy SSL

Server

Host: nollen-dms-source-aurora-db.cluster-c24vvqv1mozy.us-west-2.rds.amazonaws.com Port: 5432

Database: defaultdb

Authentication

Authentication: Database Native

Username: postgres

Password: ☒ Save password locally

Advanced

Session role: Local Client: PostgreSQL Binaries

i You can use variables in connection parameters.

Driver name: PostgreSQL [Edit Driver Settings](#)

Test Connection ... OK Cancel

On the PostgreSQL tab, I chose the following options:

Connection "Aurora DMS DB" configuration

Connection settings

PostgreSQL connection settings

PostgreSQL

Connection settings

- Initialization
- Shell Commands
- Client identification
- Transactions
- General
- Metadata
- Errors and timeouts
- > Data editor
- > SQL Editor

Main PostgreSQL Driver properties SSH Proxy SSL

Settings

☒ Show all databases

☒ Show template databases

☐ Show databases not available for connection

☒ Show database statistics

☐ Read all data types

SQL

Show \$\$ quote as: Code block

Show \$tagName\$ quote as: Code block

Performance

☐ Use prepared statements

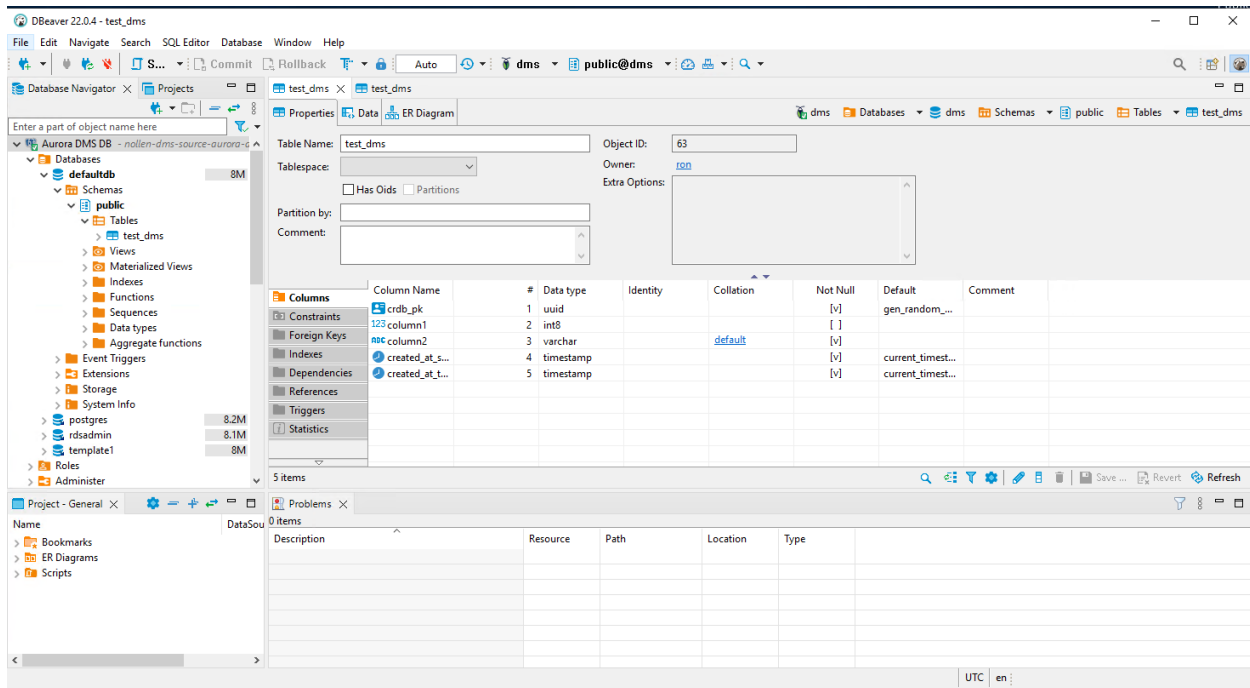
Test Connection ... OK Cancel

Configure DMS and Start On Going Replication

Create the PostgreSQL Source Table

Using DBeaver, I created the following table as my source in the “defaultdb” in Aurora PostgreSQL:

```
CREATE TABLE dms.public.test_dms (  
    column1 serial4 NOT NULL,  
    column2 varchar NOT NULL,  
    created_at_source timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,  
    CONSTRAINT test_dms_pk PRIMARY KEY (column1)  
);
```



Create the CockroachDB Target Table

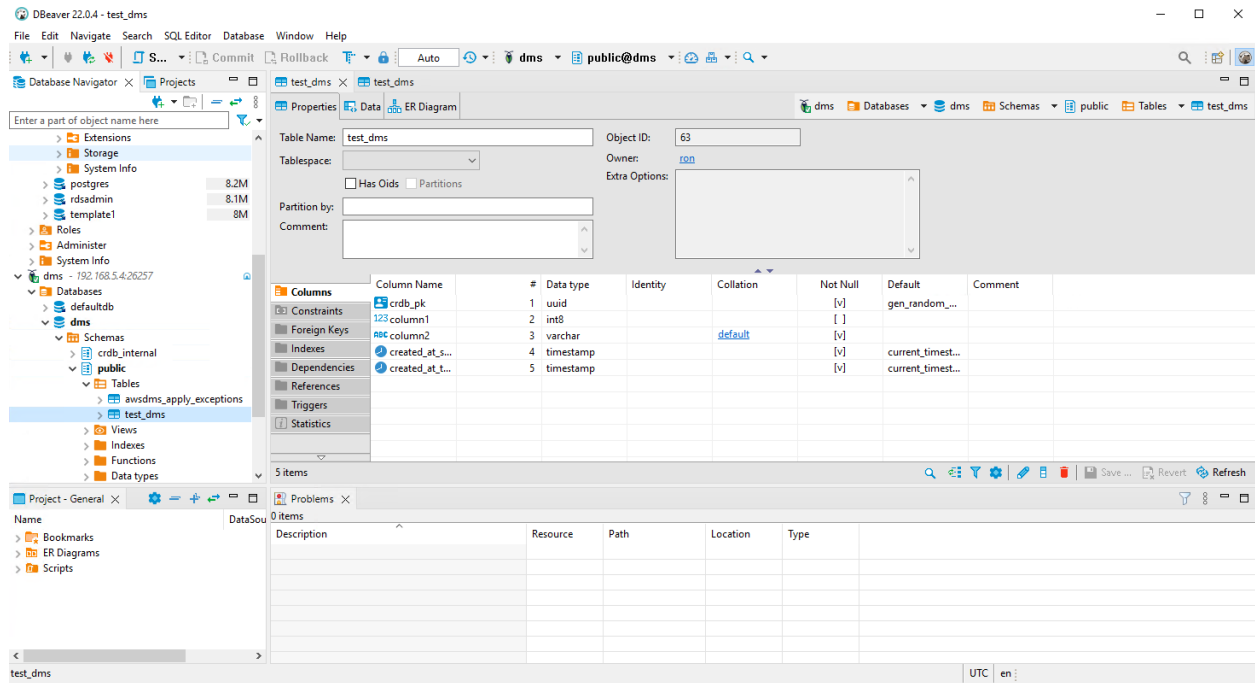
Using DBeaver, I created the following table as my target in the “dms” database of my CockroachDB Cluster

```
CREATE TABLE public.test_dms (  
    crdb_pk UUID NOT NULL DEFAULT gen_random_uuid(),  
    column1 INT8 NULL,  
    column2 VARCHAR NOT NULL,  
    created_at_source TIMESTAMP NOT NULL DEFAULT  
current_timestamp()::TIMESTAMP,  
    created_at_target TIMESTAMP NOT NULL DEFAULT  
current_timestamp()::TIMESTAMP,  
    CONSTRAINT test_dms_pkey PRIMARY KEY (crdb_pk ASC)  
);
```

A couple of important notes:

- The DMS does not need me to create any mapping from the PostgreSQL table to the CockroachDB table even though the tables are considerable different. (Gotta love that!)

- We did not have to create a table in the target database. DMS will create the table automatically on our behalf if it does not exist.



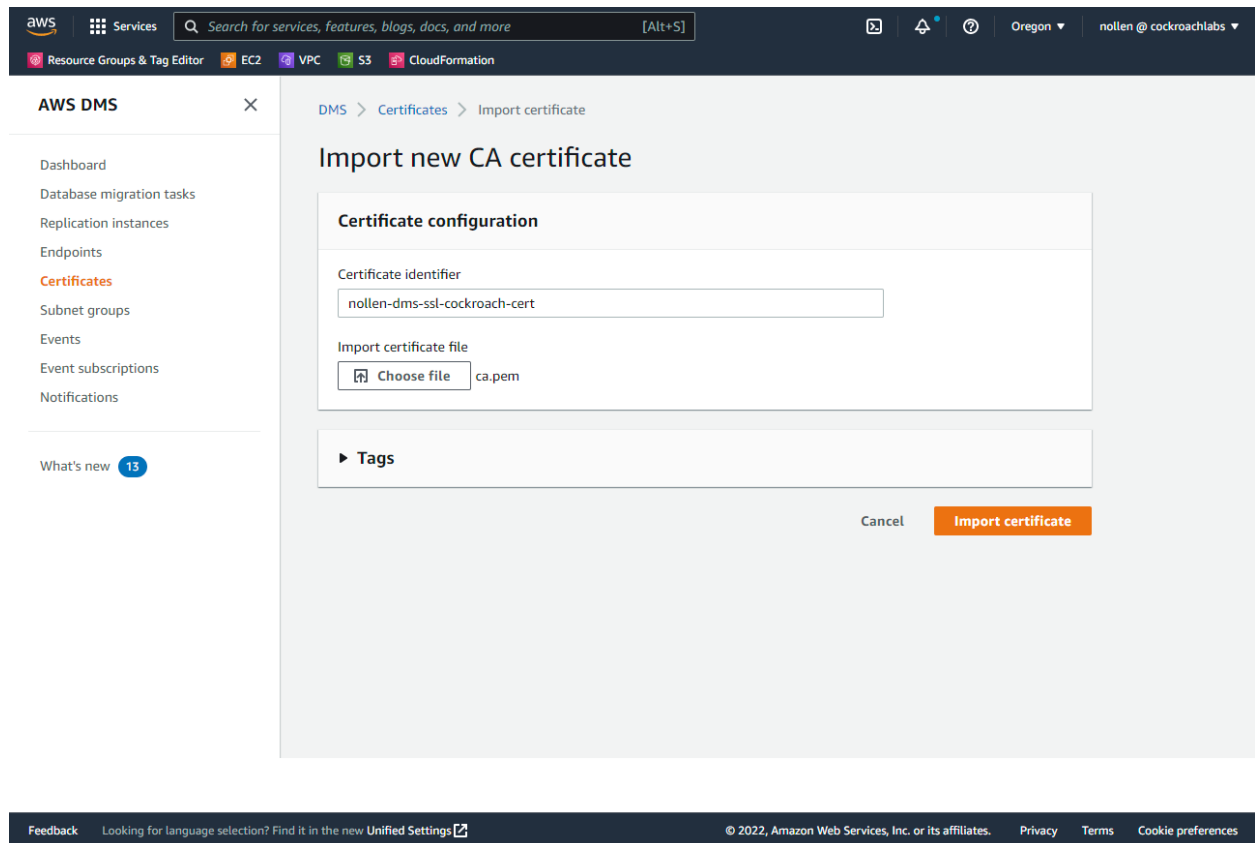
Create the Certificate

In order to have SSL communication to my CockroachDB, I'll need to upload the ".pem" cert I created earlier.

Navigate to the AWS DMS page and from the left-hand panel, choose "Certificates" and then select "Import certificate".

The screenshot shows the AWS Management Console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and user information. Below this is a sidebar for 'AWS DMS' with a list of navigation items: Dashboard, Database migration tasks, Replication instances, Endpoints, Certificates (highlighted with a red box), Subnet groups, Events, Event subscriptions, and Notifications. At the bottom of the sidebar is a 'What's new' section with a blue badge showing '13'. The main content area is titled 'DMS > Certificates'. It features a 'Certificates (1)' header with a refresh icon, a 'Delete' button, and a red-outlined 'Import certificate' button. Below the header is a search bar labeled 'Find replication instance' and a filter bar with a text input containing 'aaa' and a 'Clear Filters' button. A table with columns for Name, Signing algorithm, Account Id, Key size, Valid from, Valid to, ARN, and TAG is visible, but it is currently empty. The footer of the console contains links for Feedback, language selection, and settings, along with copyright information and links for Privacy, Terms, and Cookie preferences.

On the next page, give your certificate a name and then select the “Choose file” button to select the ca.pem file.



Create the Subnet groups

Much like we created subnet groups for the Aurora database, we should create a subnet group for the DMS Replication Instance.

To create the subnet group, navigate to AWS DMS and select “Subnet groups” from the left hand side and select “Create subnet group”.

Be sure to choose the same VPC we’ve been working in throughout this blog, and then create the subnets from the VPC into which we could place the replication instance. I’m going to choose the public subnets from my VPC to create the subnet group.

Details

ARN	arn:aws:dms:us-west-2:541263489771:subgrp:nollen-dms-subnet-group	Status	Complete
Description	nollen-dms-subnet-group	VPC	vpc-0458048d6a007cb76

Subnets (3)

Name	Status	Availability Zone
subnet-0d04652ed28189508	Active	us-west-2c
subnet-0fe83cc1bd45bf17d	Active	us-west-2b
subnet-04da8d0cdf533c8c3	Active	us-west-2a

Tags (1)

Key	Value
owner	nollen

Create the Replication instance

DMS uses an EC2 instance to perform the replication between databases. In this step we create the replication instance.

Be sure to place the replication instance in the same VPC we've placed both of our databases. Note that it is possible to create a Multi-AZ replication instance configuration (for failovers), but for this demonstration, I'm going to leave that off. The subnet group that I created above included the public subnets so that I can access the instance if I need.

I also chose a small instance class for this demonstration (dms.t23.medium). Do not do this in any environment where you need substantial throughput (production or load testing). Choosing an instance class that is too small, along with storage is the Achilles heel of DMS.

aws Services secrets X

Resource Groups & Tag Editor EC2 VPC S3 CloudFormation

DMS > Replication instances > nollen-dms-replication-instance

nollen-dms-replication-instance

Refresh Actions

Replication instance summary

Class dms.t3.medium	Engine version 3.4.6	Status Available	Associated migration tasks 0
------------------------	-------------------------	---------------------	---------------------------------

Overview details CloudWatch metrics Migration tasks Log management Tags

Details

Basic configuration

ARN arn:aws:dms:us-west-2:541263489771:rep:VHXZRXFQWD3YFIYFIY5SVL5TI75S4XSPTCPJUNI	Status Available
Public IP address 52.34.119.69	Private IP address 192.168.5.11
Instance class dms.t3.medium	Free instance expiration N/A

Advanced security and network configuration

Multi AZ No	Availability zone us-west-2a
Publicly accessible Yes	Replication subnet group nollen-dms-subnet-group
Allocated storage (GiB) 50	VPC security group(s) sg-0b81b4ed1b0f3563d, sg-0569e3c6770808d07
Created May 21, 2022, 14:14:50 (UTC-07:00)	KMS key arn:aws:kms:us-west-2:541263489771:key/a3471208-77e8-41ad-bc9d-ea70d698c48c

Maintenance

Minor version automatic upgrade Yes	Maintenance window Tuesday at 05:41 (UTC) for 0.5 hour(s)
Pending modify values None	

Feedback Looking for language selection? Find it in the new Unified Settings

© 2022, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Create the Endpoints

Now, we need to create the endpoints (the target and source) for DMS. To get started, navigate to the AWS DMS page and then select “Endpoints” from the left hand menu.

When we’re done, we should have the following:

The screenshot shows the AWS DMS Endpoints console. The left sidebar contains navigation links: Dashboard, Database migration tasks, Replication instances, Endpoints (highlighted), Certificates, Subnet groups, Events, Event subscriptions, and Notifications. Below these is a 'What's new' section with a badge for 13 items.

The main content area is titled 'DMS > Endpoints'. It features a search bar with the text 'Find endpoint', a filter button labeled 'nollen', and a 'Clear Filters' button. There are also buttons for 'Refresh', 'Actions', and 'Create endpoint'.

The endpoints are listed in a table with the following columns: Name, Type, Status, Engine, Server name, and Port. Two endpoints are visible:

Name	Type	Status	Engine	Server name	Port
nollen-dms-source-aurora-db-instance-1	Source	Active	Amazon Aurora PostgreSQL	AWS Secrets Manager credentials	-
nollen-dms-target-cockroachdb	Target	Active	PostgreSQL	AWS Secrets Manager credentials	-

The footer of the console includes a 'Feedback' link, a language selection notice, a copyright notice for 2022 Amazon Web Services, Inc., and links for 'Privacy', 'Terms', and 'Cookie preferences'.

Create the Source Endpoint

Our source endpoint is going to be our Aurora PostgreSQL database that we created above. I'll be using the ARN of the secret we created above, along with the IAM role (and it's included policy) in this set up.

Once you've created the endpoint, be sure to test connectivity. Do not proceed until connectivity is successful.

The screenshot displays the AWS Management Console interface for the AWS DMS service. The left-hand navigation pane is open, showing various AWS services and a search bar. The 'Endpoints' section is highlighted. The main content area shows the 'Connections' tab for the endpoint 'nollen-dms-source-aurora-db-instance-1'. A table lists the connections, showing a successful connection to 'nollen-dms-replication-instance'.

Endpoint identifier	Replication instance	Status	Message
nollen-dms-source-aurora-db-instance-1	nollen-dms-replication-instance	successful	

Create the Target Endpoint

Creating the CockroachDB Target endpoint will include using the certificate we uploaded above. We'll chose "PostgreSQL as the Target Engine.

Services

[Alt+S]

Resource Groups & Tag Editor

EC2

VPC

S3

CloudFormation

AWS DMS

×

Dashboard

Database migration tasks

Replication instances

Endpoints

Certificates

Subnet groups

Events

Event subscriptions

Notifications

What's new 13

DMS > Endpoints > nollen-dms-target-cockroachdb > Modify endpoint

Modify endpoint

Endpoint type Info

☐ Source endpoint
A source endpoint allows AWS DMS to read data from a database (on-premises or in the cloud), or from other data source such as Amazon S3.

☒ Target endpoint
A target endpoint allows AWS DMS to write data to a database, or to other data source.

Endpoint configuration

Endpoint identifier Info
A label for the endpoint to help you identify it.
nollen-dms-target-cockroachdb

Target engine
The type of database engine this endpoint is connected to.
PostgreSQL

Access to endpoint database
☒ AWS Secrets Manager
☐ Provide access information manually

Secret ID
Amazon Resource Name (ARN) of the secret used to manage access to your endpoint database. If you need a new secret ARN, create it from [AWS Secrets Manager](#).
arn:aws:secretsmanager:us-west-2:541263489771:secret:nollen/dms/cockroachdb-SCpuiM

IAM role
IAM role that grants Amazon DMS permissions to access the specified secret (and any required KMS encryption key). For a new IAM role, create it from [IAM](#).
arn:aws:iam::541263489771:role/nollen-read-secrets-role

Secure Socket Layer (SSL) mode
The type of Secure Socket Layer enforcement
verify-ca

CA certificate
nollen-dms-ssl-cockroach-cert Add new CA certificate

Database name
dms

▶ Endpoint settings

Cancel

Save

Feedback

Looking for language selection? Find it in the new [Unified Settings](#)

© 2022, Amazon Web Services, Inc. or its affiliates.

Privacy

Terms

Cookie preferences

Be sure to test the connectivity to the CockroachDB instance, and fixing any problems, before going on to the next step.

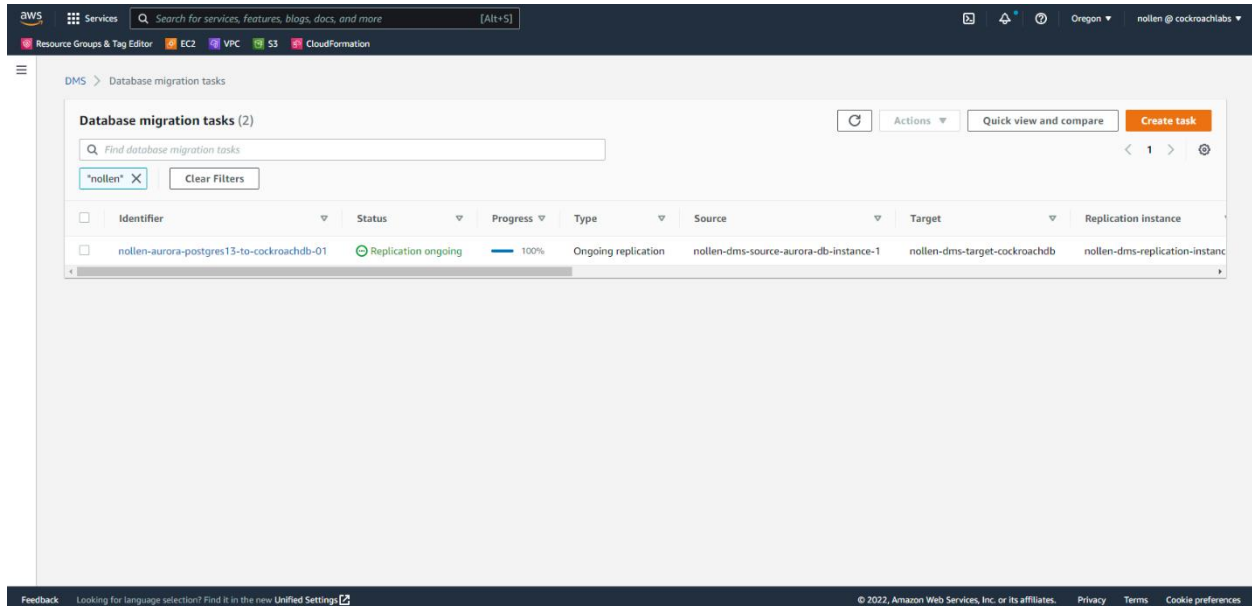
Create the Database migration task

At this point we've got:

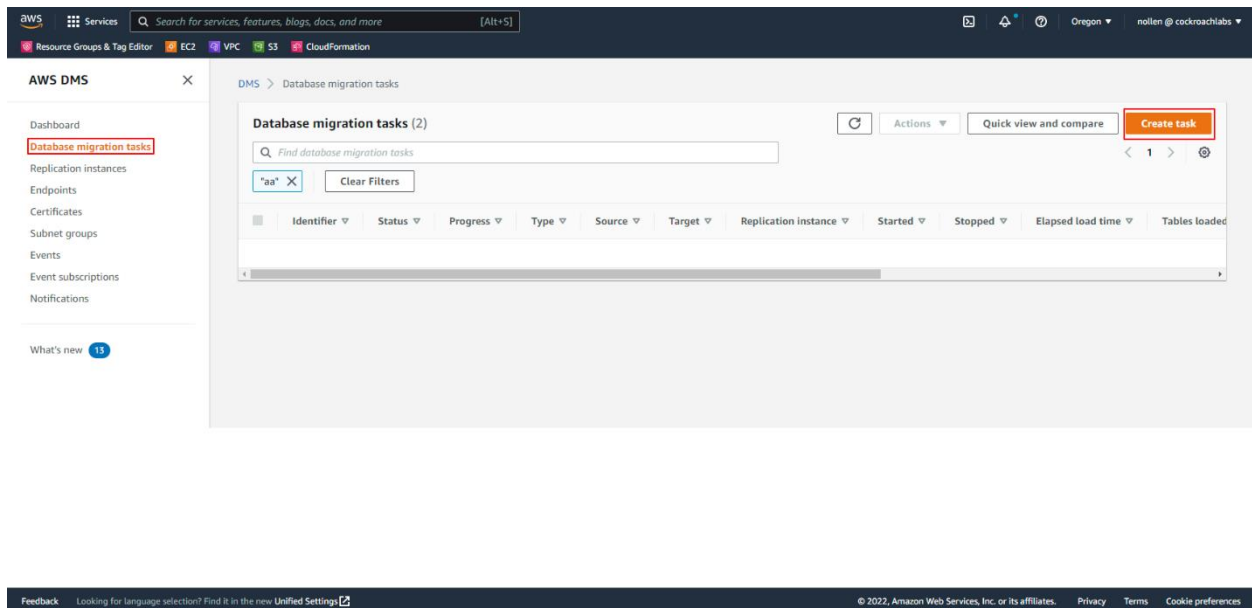
- A Source Aurora PostgreSQL Database configured with replication
- A 3 Node CockroachDB Cluster running in the same VPC as the Aurora Database
- DMS Endpoint for the Source Aurora Database
- DMS Endpoint for the Target Cockroach Database
- The source table created in the Aurora Database

- The target table create in the Cockroach Database (although this was not necessary)
- The DMS Replication instance is running

Now, we're ready to create the DMS task, which is the instruction set for the replication from the source to the target. When we've finished and the task is running it will look like this:



To create the task navigate to “AWS DMS”, select “Database migration tasks” from the left hand side and then choose “Create task”.



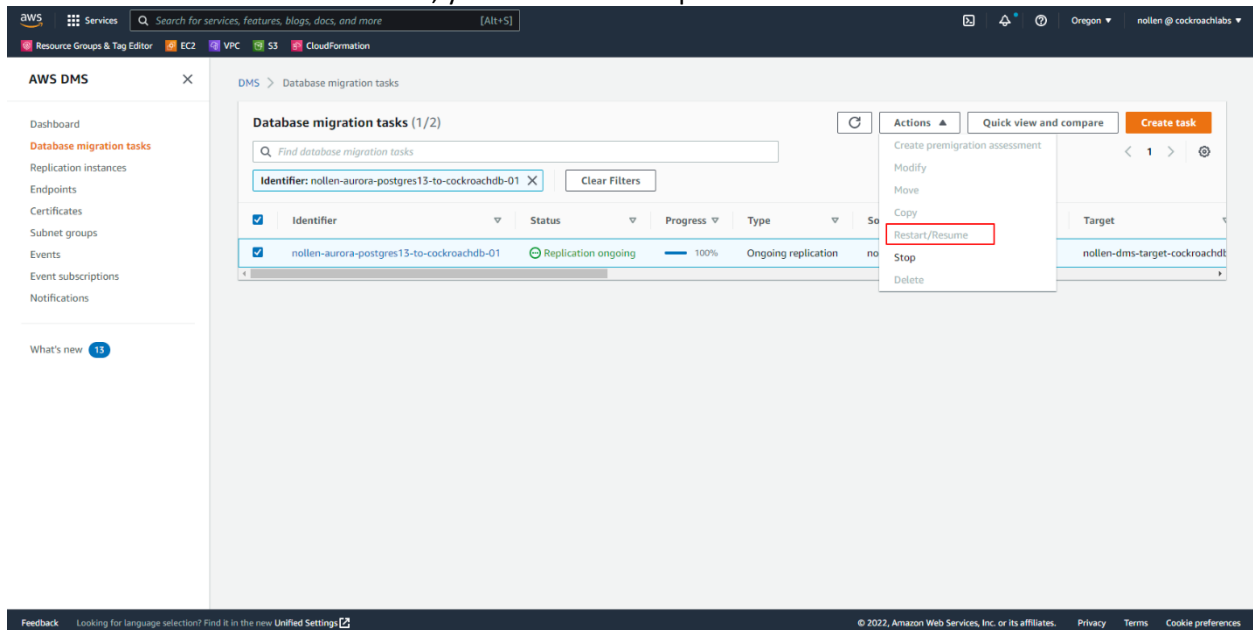
Basically, we've done all the hard work. In this step we're going to choose elements we've created in the steps above.

In my example below, I'm including CloudWatch logs (which we should modify after starting the replication so that logs are only kept for 5 or 7 days).

You also need to create at least one selection rule which tells DMS which objects to include in the replication task. In my example, I'm selecting all the tables in the public schema of the "defaultdb".

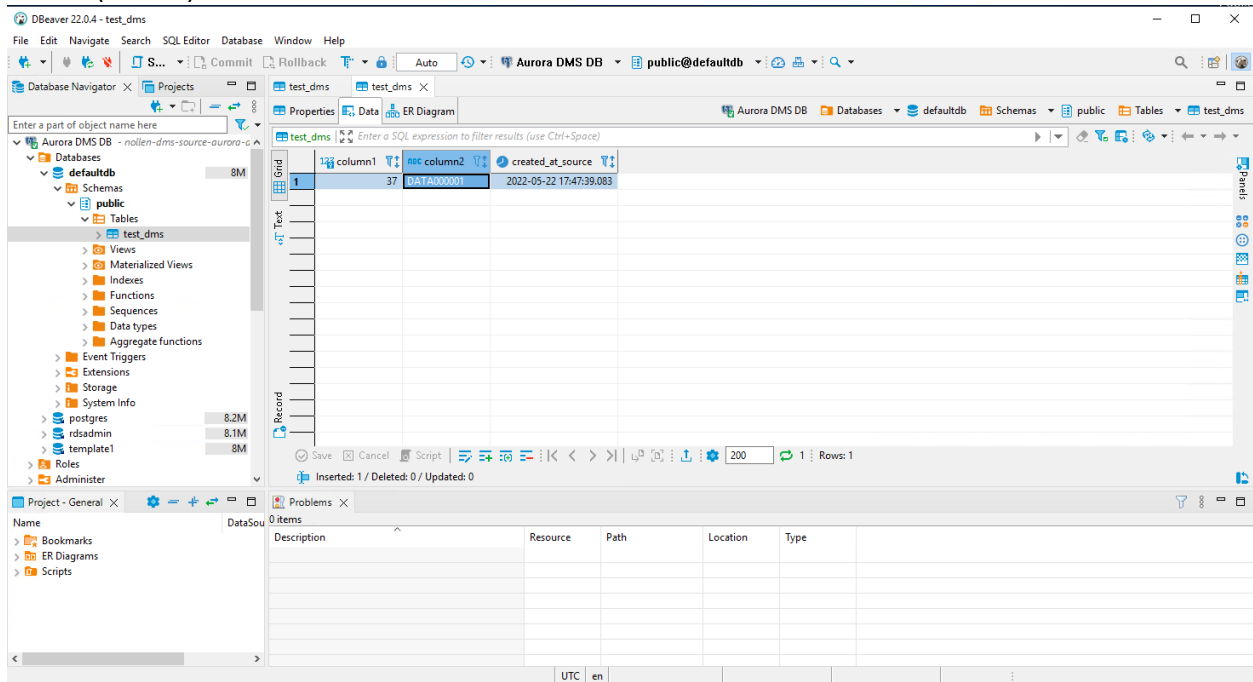
[illegible]

Now that the task has been created, you can start the replication!

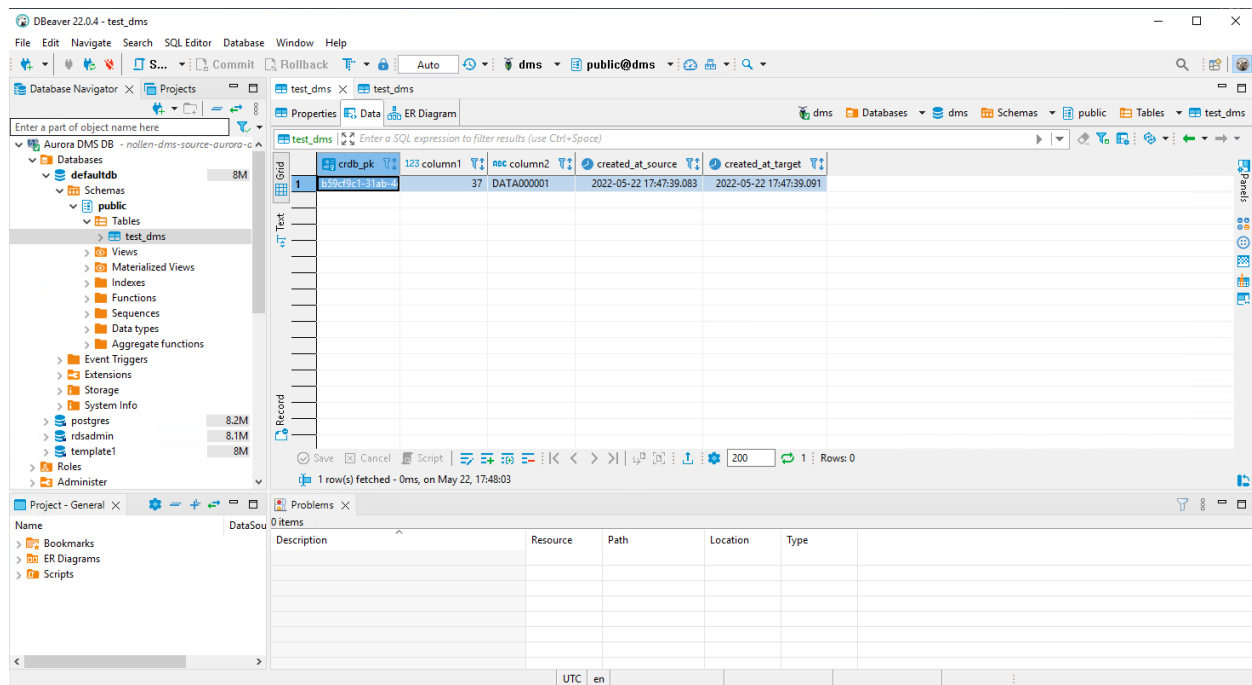


Once the task is running, you will be able to go to DBeaver and insert a row into the Aurora PostgreSQL database and see it replicated to CockroachDB

Source (Aurora)



Target (CockroachDB)



In our simple example, we can see it took about 8ms for the record to be replicated from Aurora to CockroachDB.

This was a very long DMS example and included things like Secrets, certification, subnet groups, etc. We could have made this much simpler, but leave it to you to carve out things that are not necessary for your implementation.

CHEERS!

