

# Modernization of Applications on AWS- Full Migration vs Integration

## Intro

Many of our customers and large organizations are trying to modernize archaic applications and architectures. The main drive behind these initiatives is to:

- 1) Reduce cost – sustaining architectures that are based on licensing models cost a lot of money.
- 2) Reduce fault tolerance and efficiency – archaic technologies that are far behind from the distributing computing of today, have complex and costly setups to allow high availability and fault tolerance.
- 3) Scalability – many old school databases struggle to scale automatically according to consumption demand often occurring unnecessary costs to customers or failing to meet demand.
- 4) New Requirements – as business needs grow, there is a need to introduce new functionality to increase overall operational efficiency and meet new requirements. One example of that is Machine Learning use cases as many organizations are trying to exploit their data assets to realise data insights.

Two very common approaches to achieve Modernization of Applications are:

- Fully migrate your application to quickly reduce cost, increase fault tolerance and efficiency
- Integrate your archaic applications with new capability that is easily realised on AWS Cloud to increase operational efficiency

In this white paper and use case example we will showcase both of these scenarios and how you can easily migrate your entire application on AWS Cloud or introduce a new capability by integrating with AWS Cloud Services.

## Our Scenario

A customer has an application running on MS SQL Server 2016 Enterprise. Due to constant upgrade requests and concerns over cost, high availability and new functional requirements, the customer has made a decision to either migrate the whole application or at least introduce the new functionality by integrating to AWS Cloud Services. Our customer has requested from AWS Professional Services to provide a POC that can prove that both approaches are valid.

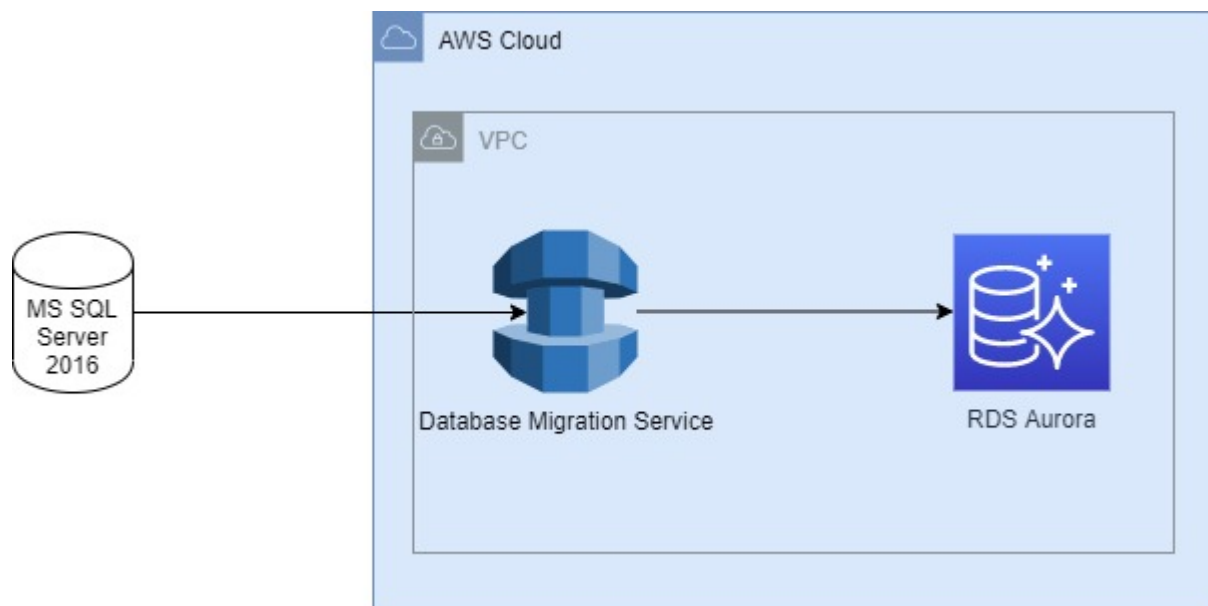
## Approach 1 – Full Migration

As explained above, one of the key drivers for full migration is reducing cost, scalability, fault tolerance and efficiency. In order to achieve that our customer has selected Database Migration Service (DMS) to move their data to the cloud and RDS Aurora as their target Database.

With AWS DMS, the source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. The AWS Database Migration Service can migrate your data to and from most widely used commercial and open-source databases. AWS

Database Migration Service supports homogeneous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle or Microsoft SQL Server to Amazon Aurora. With AWS Database Migration Service, you can continuously replicate your data with high availability and consolidate databases into a petabyte-scale data warehouse by streaming data to Amazon Redshift or RDS and Amazon S3. AWS DMS is a low cost service (as you only pay for the compute resources used during the migration process and any additional log storage), its fast and easy to set-up and its very reliable as it can be Highly Available (through Multi-AZ mode) and its self-healing.

Amazon Aurora is a MySQL and PostgreSQL-compatible relational database built for the cloud that combines the performance and availability of traditional enterprise databases with the simplicity and cost-effectiveness of open source databases. Amazon Aurora is up to five times faster than standard MySQL databases and three times faster than standard PostgreSQL databases. It provides the security, availability, and reliability of commercial databases at 1/10th the cost. Amazon Aurora is fully managed by Amazon Relational Database Service (RDS), which automates time-consuming administration tasks like hardware provisioning, database setup, patching, and backups. Amazon Aurora features a distributed, fault-tolerant, self-healing storage system that auto-scales up to 128TB per database instance. It delivers high performance and availability with up to 15 low-latency read replicas, point-in-time recovery, and continuous backup to Amazon S3, and replication across three Availability Zones (AZs).



Let's go through our code example on how this would look like from a technical perspective. In order to emulate our scenario we have uploaded the Retail Dataset from Kaggle into an MS SQL Server 2016 Enterprise Edition database on EC2.

#### **Approach 1 – Guide:**

In order to follow this guide please use the code stored in this repo

<https://github.com/angeloschionis/SalesData/tree/master/sql-server-to-rds-via-DMS>

#### **Prerequisites:**

- 1) Create an MS SQL Instance on EC2 using an approach as described here:  
<https://youtu.be/9VAzYEKMmvU> .
- 2) Please load the data to your MS SQL instance from Kaggle here :  
<https://www.kaggle.com/manjeetsingh/retaildataset>
- 3) Configure your MS SQL Server according to the requirements and prerequisites listed here if you require CDC as part of the DMS task.  
[https://docs.aws.amazon.com/dms/latest/userguide/CHAP\\_Source.SQLServer.html](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.SQLServer.html)
- 4) DMS uses 2 IAM roles -1) dms-vpc-role 2) dms-cloudwatch-logs-role. Hence these roles need to be created before the actual DMS cloudformation stack is created. To create these roles, please follow the steps. **Please note that this is one time task and in case the roles are already created in your AWS account with necessary permissions, you don't need to create again.**

- Login to your AWS account
- Go to cloudformation service
- Go to create stack section and upload the template “dms-vpc-role.yaml” and click “Next”

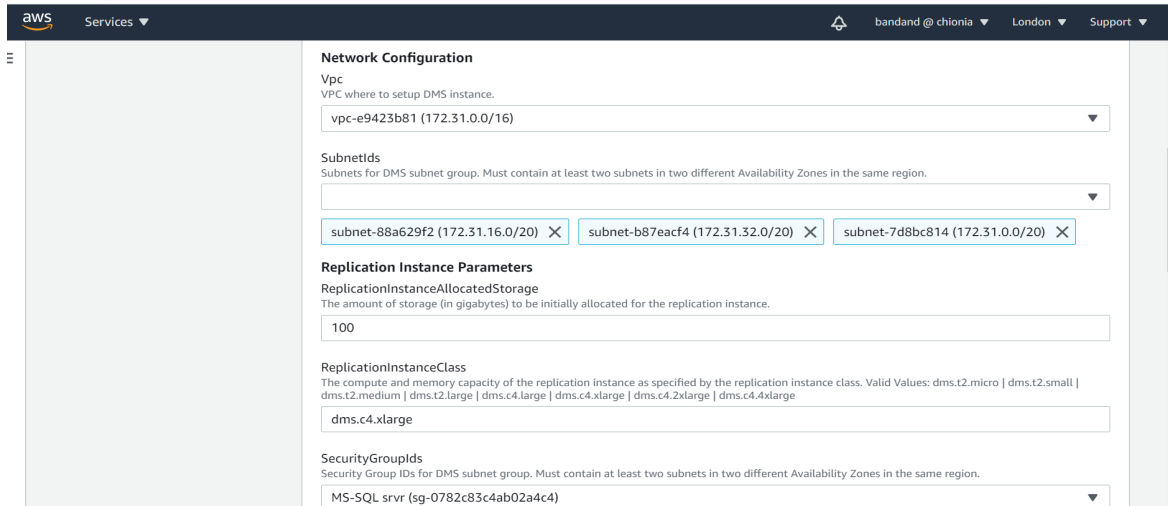
- Give a name to your stack and then again go to Next
- And finally create the stack.

- 5) The source sql server database credential needs to be stored in secret manager.
- 6) Create an AWS RDS Aurora cluster as per this documentation :  
<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.CreateInstance.html>
- 7) The target Aurora database credential needs to be stored in secret manager.

### Deployment of DMS Stack:

1) Please follow the steps to deploy the CloudFormation template “dms-sql-server-to-aurora.yaml”.

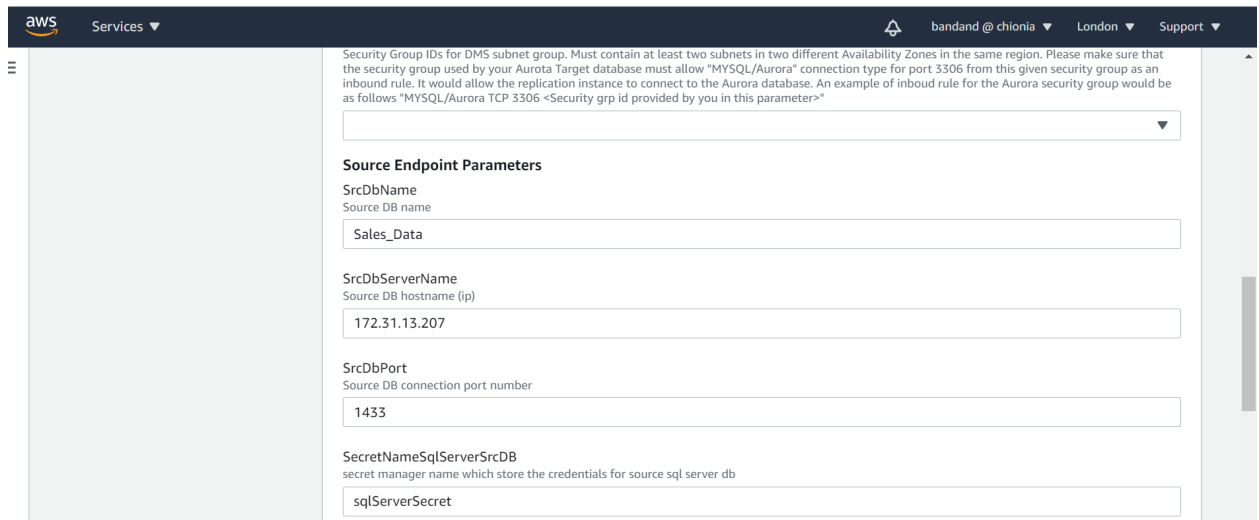
- Login to your AWS account and go to the CloudFormation service
- Upload the template “dms-sql-server-to-aurora.yaml” and follow the next steps
- Choose the network configuration like VPC, subnets etc. as per your environment.



The screenshot shows the AWS CloudFormation console interface. The top navigation bar includes the AWS logo, 'Services', and user information. The main content area displays the 'Network Configuration' section for a CloudFormation template. It includes the following fields:

- Vpc:** VPC where to setup DMS instance. Value: vpc-e9423b81 (172.31.0.0/16).
- SubnetIds:** Subnets for DMS subnet group. Must contain at least two subnets in two different Availability Zones in the same region. Values: subnet-88a629f2 (172.31.16.0/20), subnet-b87eac4 (172.31.32.0/20), subnet-7d8bc814 (172.31.0.0/20).
- ReplicationInstanceAllocatedStorage:** The amount of storage (in gigabytes) to be initially allocated for the replication instance. Value: 100.
- ReplicationInstanceClass:** The compute and memory capacity of the replication instance as specified by the replication instance class. Valid Values: dms.t2.micro | dms.t2.small | dms.t2.medium | dms.t2.large | dms.c4.large | dms.c4.xlarge | dms.c4.2xlarge | dms.c4.4xlarge. Value: dms.c4.xlarge.
- SecurityGroupIds:** Security Group IDs for DMS subnet group. Must contain at least two subnets in two different Availability Zones in the same region. Value: MS-SQL srvr (sg-0782c83c4ab02a4c4).

- Put the source sql server database details like its host ip, port number, name and the secret of the secret manager storing its username and password.



The screenshot shows the AWS CloudFormation console interface. The top navigation bar includes the AWS logo, 'Services', and user information. The main content area displays the 'Source Endpoint Parameters' section for a CloudFormation template. It includes the following fields:

- Security Group IDs:** Security Group IDs for DMS subnet group. Must contain at least two subnets in two different Availability Zones in the same region. Please make sure that the security group used by your Aurora Target database must allow "MySQL/Aurora" connection type for port 3306 from this given security group as an inbound rule. It would allow the replication instance to connect to the Aurora database. An example of inbound rule for the Aurora security group would be as follows "MySQL/Aurora TCP 3306 <Security grp id provided by you in this parameter>".
- SrcDbName:** Source DB name. Value: Sales\_Data.
- SrcDbServerName:** Source DB hostname (ip). Value: 172.31.13.207.
- SrcDbPort:** Source DB connection port number. Value: 1433.
- SecretNameSqlServerSrcDB:** secret manager name which store the credentials for source sql server db. Value: sqlServerSecret.

- Put the target Aurora RDS details and migration task configurations like which particular source schema and table data you want to migrate. In case you want to migrate all, you may put %.

aws Services

bandand @ chionia London Support

sqlServerSecret

**Target Endpoint Parameters**

TgtDbPort  
Target DB connection port number

3306

TgtDbAuroraEndPointArn  
Target DB endpoint ARN. If you go to the Aurora database from Console, it will be available in the Connectivity & security section.

database-1.cluster-cqk0hofejoyr.eu-west-2.rds.amazonaws.com

SecretNameAuroraTgtDB  
secret manager name which store the credentials for source sql server db

auroraRdsServerSecret

**Replication Task Parameters**

SrcDbSchemaName  
DB schema name, which will be migrated. You may put % in case not to specify any particular schema

%

SrcDbTableName  
DB table name, which will be migrated. You may put % in case not to specify any particular table

%

DmsReplicationTaskMigrationType  
The migration type. Valid values: full-load | cdc | full-load-and-cdc

full-load

- 2) Then follow the instructions in the console and create the stack
- 3) This template will deploy the below resources
  - a) Source endpoint to connect to MS-SQL server
  - b) Target endpoint to AWS Aurora Database
  - c) Replication Instance
  - d) Data Migration Task
- 4) As part of security measurement, followings have been considered
  - Both DMS Replication instance and Aurora target database are within VPC.
  - Both source and target Database credentials have been stored in secret manager
  - Logging and Monitoring data is being stored as cloud watch logs
- 5) After the stack creation is complete, data migration task would be created in the DMS and would be in "Ready" state. Please start the task, it would be up and running and bring the data from source database to Aurora RDS.

aws Services

bandand @ chionia London Support

**AWS DMS**

Dashboard

Migration

Database migration tasks

Resource management

Replication instances

Endpoints

Certificates

Subnet groups

Events

Event subscriptions

What's new 1

Notifications

DMS > Database migration tasks

Database migration tasks (2)

Find database migration tasks

	Identifier	Status	Progress	Type
<input checked="" type="checkbox"/>	dmsreplicationtasksqlserveraurora-hzzzothyl61hgqzw	Load complete, replication ongoing	100%	Full load, cdc
<input type="checkbox"/>	dmsreplicationtasksqlservers3-iv06qwpk1y1k3sm1	Load complete, replication ongoing	100%	Full load, cdc

- 6) Data would be loaded to corresponding target Aurora database. We connected to the Query section of the Aurora database and ran the select query for the table migrated from the source sql server database to target Aurora RDS.

The screenshot shows the Amazon RDS Query Editor interface. On the left is a navigation sidebar with options like Dashboard, Databases, Query Editor (selected), Performance Insights, Snapshots, Automated backups, Reserved instances, Proxies, Subnet groups, Parameter groups, Option groups, Events, Event subscriptions, Recommendations, and Certificate update. The main area displays a SQL query in a text editor:

```
1 # select * from information_schema.tables;
2 # Press run and see the current database tables below
3
4 select * from dbo.'Features data set' limit 10;
```

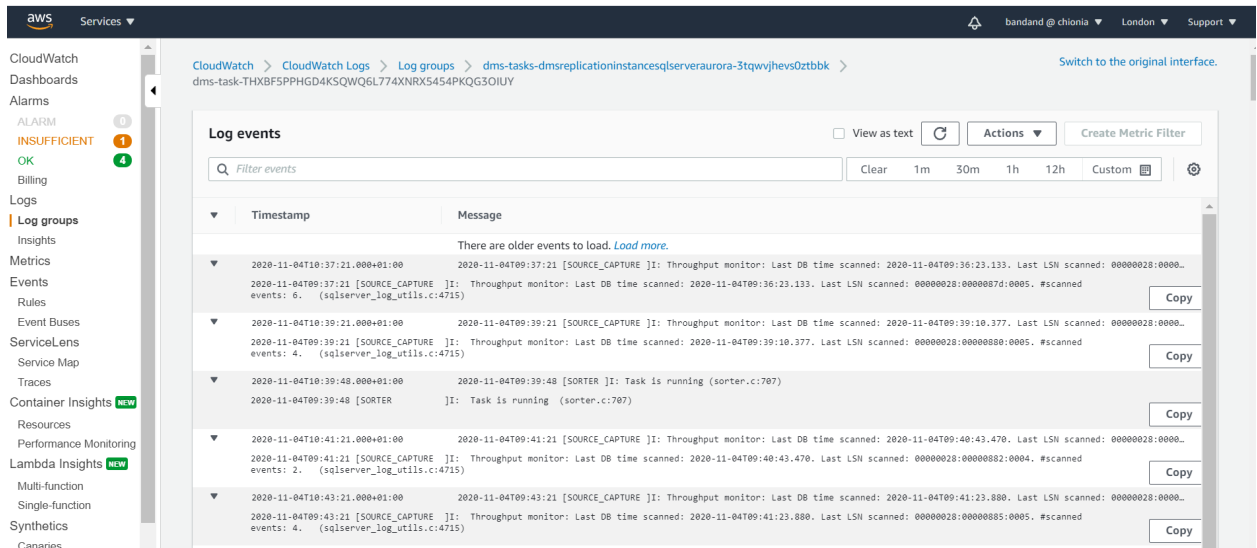
Below the query editor are buttons for Run, Save, Clear, and Change database. The Output section shows 'Result set 2 (10)' with a search bar and an 'Export to csv' button. A table of 10 rows is displayed with columns: Store, Date, Temperature, Fuel\_Price, MarkDown1, MarkDown2, MarkDown3, MarkDown4, MarkDown5, CPI, and Unemployment. The first two rows are visible:

Store	Date	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
1	05/02/2010	42.31	2.572	NA	NA	NA	NA	NA	211.0963582	8.10
1	12/02/2010	38.51	2.548	NA	NA	NA	NA	NA	211.2421698	8.10

- 7) The log for data migration task would be available in cloudwatch log. The link for the cloudwatch log group is available inside the database migration task.

The screenshot shows the AWS DMS Overview details page for a database migration task. The left sidebar includes options like Dashboard, Migration, Database migration tasks (selected), Resource management, Replication instances, Endpoints, Certificates, Subnet groups, Events, Event subscriptions, What's new (1), and Notifications. The main area has tabs for Overview details (selected), Table statistics, CloudWatch metrics, Mapping rules, Premigration assessments (New), Assessment results, and Tags. The Overview details section shows the Basic configuration for a task named 'arn:aws:dms:eu-west-2:386900942011:task:THXBFSPPHGD4KSQWQ6L774XNRX5454PKQG3OIUY'. The Progress bar is at 100% with the status 'Load complete, replication ongoing'. The Created date is '11/2/2020, 11:15:46 AM GMT+0100'. The Stopped status is '-'. The Change data capture (CDC) section shows 'Change data capture (CDC) start position' as '-' and 'Change data capture (CDC) recovery checkpoint' as '-'. On the right, the Replication instance is 'dmsreplicationinstancesqlserveraurora-3tqwvjhevs0ztbbk', the Last failure message is '-', and the Started date is '11/2/2020, 11:19:04 AM GMT+0100'. There are links for 'Migration task logs Info' and 'View CloudWatch logs'.

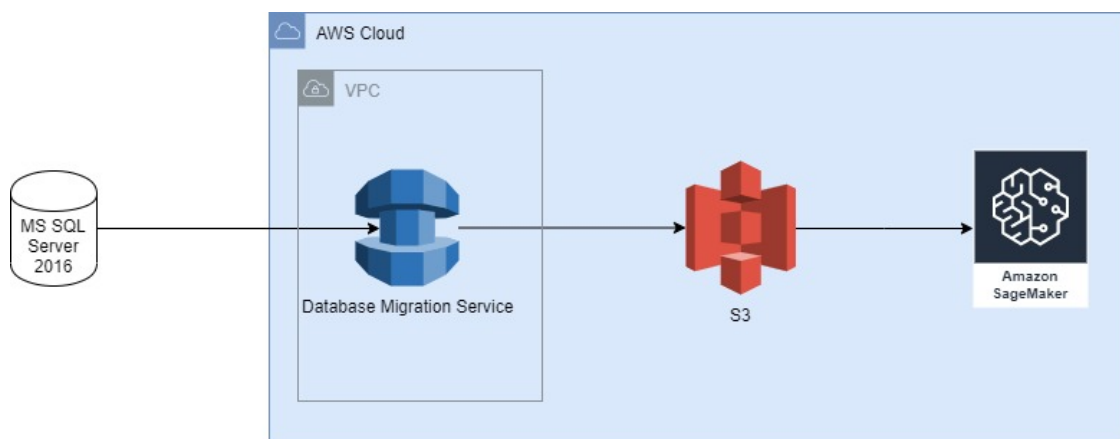
After we click the link for cloudwatch log, it would take us to the details of the logs within cloudwatch



## Approach 2 – Integrating with AWS Cloud Services

In this approach we need to showcase how our customer can integrate with AWS Cloud Services to provide a new functionality. Our customer has realised that the sales data that are holding in their MS SQL Server can be used to predict sales for the next years. Thus we will take an approach that will help us extract data from our warehouse and exploit them in such manner.

We will achieve our goal by using AWS DMS as well S3 to store our data and AWS SageMaker to gain some valuable insights. AWS SageMaker is a fully managed service that provides every developer and data scientist with the ability to build, train, and deploy machine learning (ML) models quickly. SageMaker removes the heavy lifting from each step of the machine learning process to make it easier to develop high quality models.



### Approach 2- Guide:

In order to follow this guide please use the code stored in this repo

<https://github.com/angeloschionis/SalesData/tree/master/sql-server-to-s3-and-sagemaker-via-DMS>

## Prerequisites:

- 1) Create an MS SQL Instance on EC2 using an approach as described here: <https://youtu.be/9VAzYEKMmvU> .
- 2) Please load the data on your MS SQL instance from Kaggle here : <https://www.kaggle.com/manjeetsingh/retaildataset>
- 3) To enable ongoing replication i.e. CDC or Full Load + CDC, from the MS-SQL server database via DMS, please follow the configuration/set for the MS-SQL server database as per the link - [https://docs.aws.amazon.com/dms/latest/userguide/CHAP\\_Source.SQLServer.html](https://docs.aws.amazon.com/dms/latest/userguide/CHAP_Source.SQLServer.html)
- 4) DMS uses 2 IAM roles -1) dms-vpc-role 2) dms-cloudwatch-logs-role. Hence these roles need to be created before the actual DMS cloudformation stack is created. To create these roles, please follow the steps. **Please note that this is one time task and in case the roles are already created in your AWS account with necessary permissions, you don't need to create again.**

- Login to your AWS account

- Go to cloudformation service

- Go to create stack section and upload the template “dms-vpc-role.yaml” and click “Next”

- Give a name to your stack and then again go to Next
- And finally create the stack.

- 5) Please make sure that your target s3 bucket has the necessary bucket policy to allow the role that DMS would use to connect to S3 bucket. An example of s3 bucket policy is mentioned below.

```
{
```



```

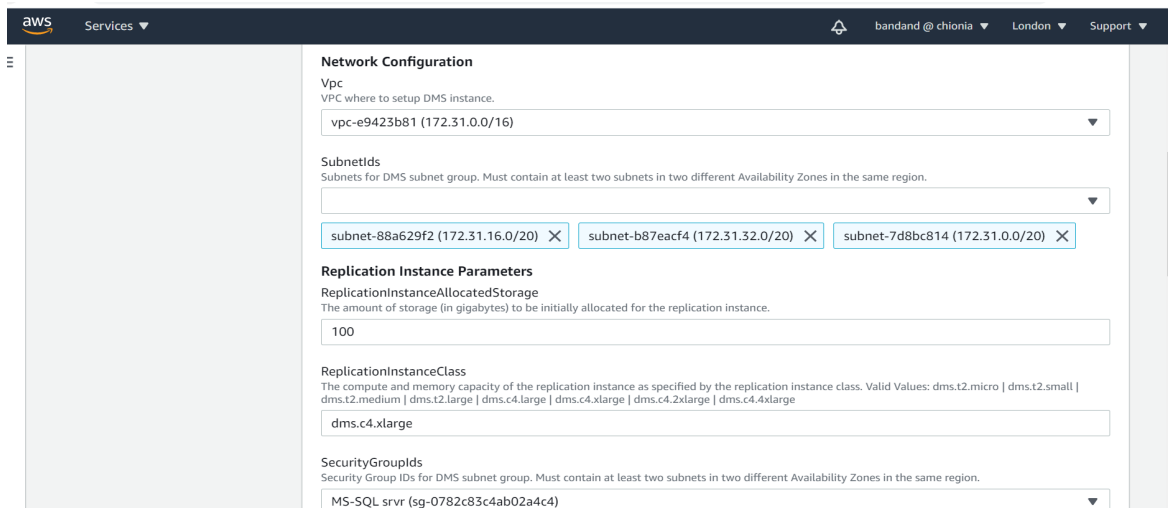
"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "allowDMSRoleToaccessS3Bucket",
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::386900942011:role/dms-to-access-s3"
    },
    "Action": [
      "s3:PutObject",
      "s3:PutObjectTagging",
      "s3:DeleteObject"
    ],
    "Resource": "arn:aws:s3:::sales-data-sql-server-via-dms/*"
  }
]
}

```

- 6) Replace the roles that DMS takes to connect to S3 and the target S3 bucket ARN to reflect your environment.
- 7) The source sql server database credentials is stored in secret manager.

### **Deployment of CloudFormation stack**

- 1) Please follow the steps to deploy the cloudformation template "*dms-sql-server-s3-parametrized.yaml*".
  - Login to your AWS account and go to the cloudformation service
  - Upload the template "*dms-sql-server-s3-parametrized.yaml*" and follow the next steps
  - Choose the network configuration like VPC , subnets etc as per your environment.



**Network Configuration**

**Vpc**  
VPC where to setup DMS instance.  
vpc-e9423b81 (172.31.0.0/16)

**SubnetIds**  
Subnets for DMS subnet group. Must contain at least two subnets in two different Availability Zones in the same region.  
subnet-88a629f2 (172.31.16.0/20) × subnet-b87eacf4 (172.31.32.0/20) × subnet-7d8bc814 (172.31.0.0/20) ×

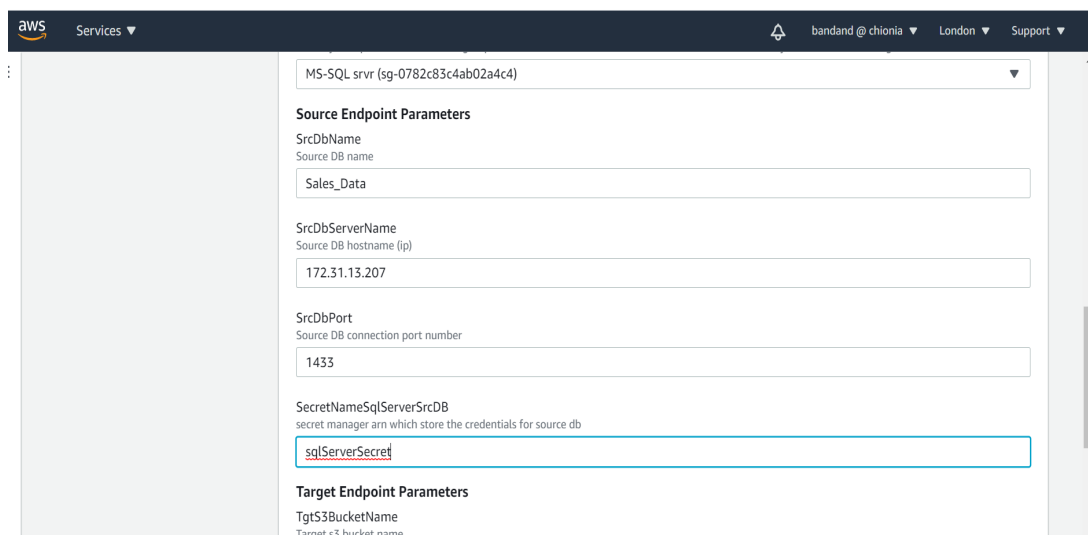
**Replication Instance Parameters**

**ReplicationInstanceAllocatedStorage**  
The amount of storage (in gigabytes) to be initially allocated for the replication instance.  
100

**ReplicationInstanceClass**  
The compute and memory capacity of the replication instance as specified by the replication instance class. Valid Values: dms.t2.micro | dms.t2.small | dms.t2.medium | dms.t2.large | dms.c4.large | dms.c4.xlarge | dms.c4.2xlarge | dms.c4.xlarge  
dms.c4.xlarge

**SecurityGroupIds**  
Security Group IDs for DMS subnet group. Must contain at least two subnets in two different Availability Zones in the same region.  
MS-SQL srvr (sg-0782c83c4ab02a4c4)

- Put the source sql server database details like its host ip, port number, name and the secret of the secret manager storing its username and password.



**Source Endpoint Parameters**

**SrcDbName**  
Source DB name  
Sales\_Data

**SrcDbServerName**  
Source DB hostname (ip)  
172.31.13.207

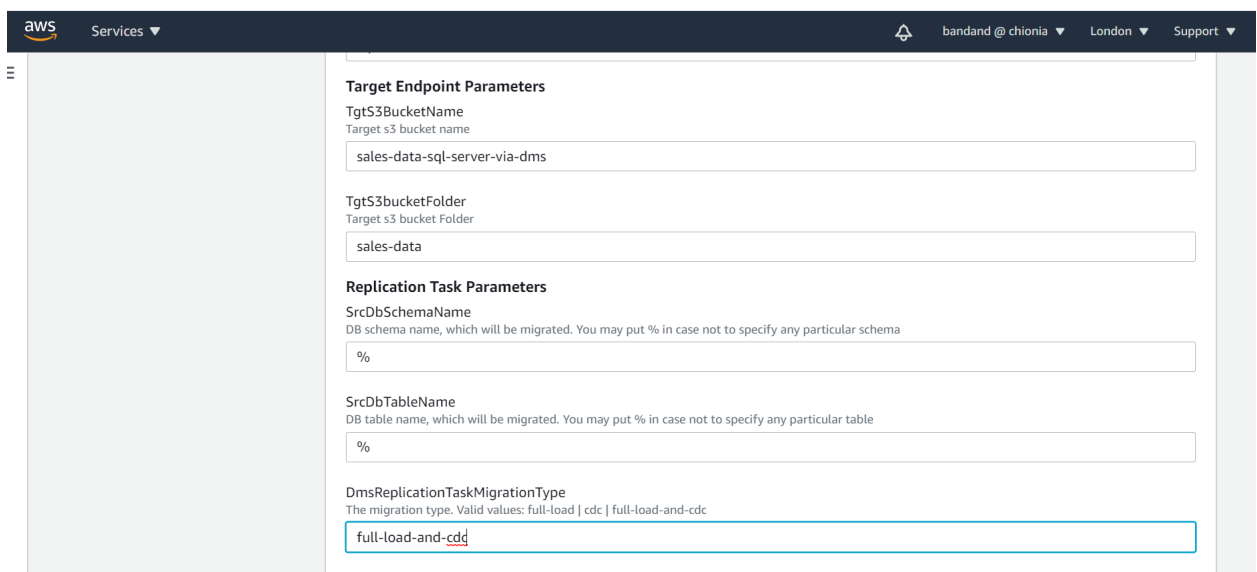
**SrcDbPort**  
Source DB connection port number  
1433

**SecretNameSqlServerSrcDB**  
secret manager arn which store the credentials for source db  
sqlServerSecret

**Target Endpoint Parameters**

**TgtS3BucketName**  
Target s3 bucket name

- Put the target s3 bucket details and migration task configurations like which particular source schema and table data you want to migrate. In case you want to migrate all, you may put %.



**Target Endpoint Parameters**

**TgtS3BucketName**  
Target s3 bucket name  
sales-data-sql-server-via-dms

**TgtS3bucketFolder**  
Target s3 bucket Folder  
sales-data

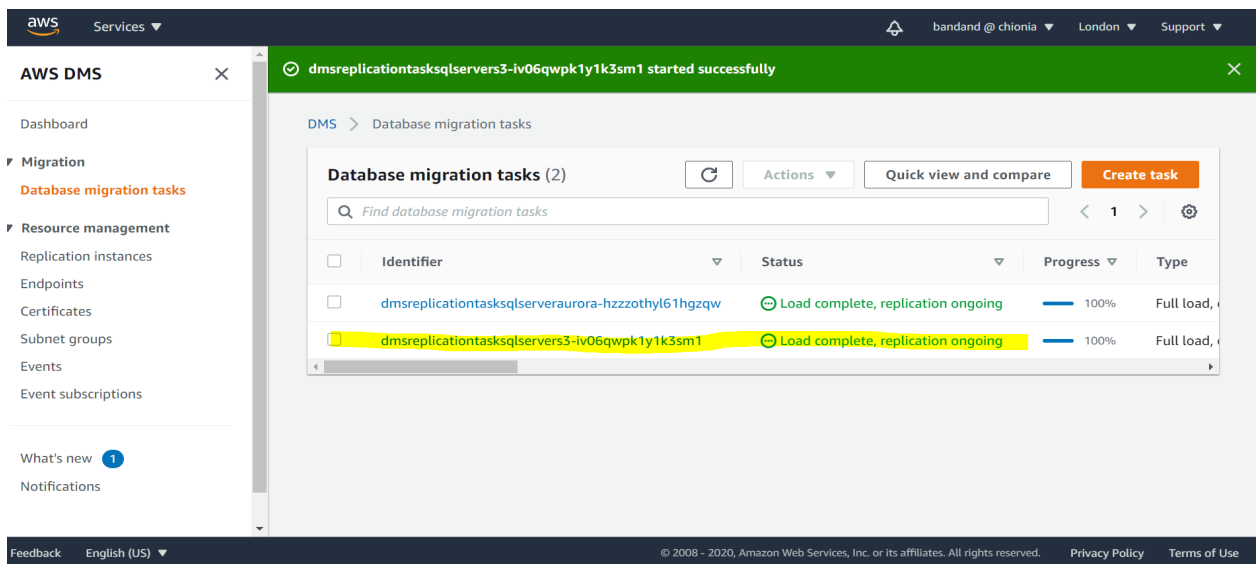
**Replication Task Parameters**

**SrcDbSchemaName**  
DB schema name, which will be migrated. You may put % in case not to specify any particular schema  
%

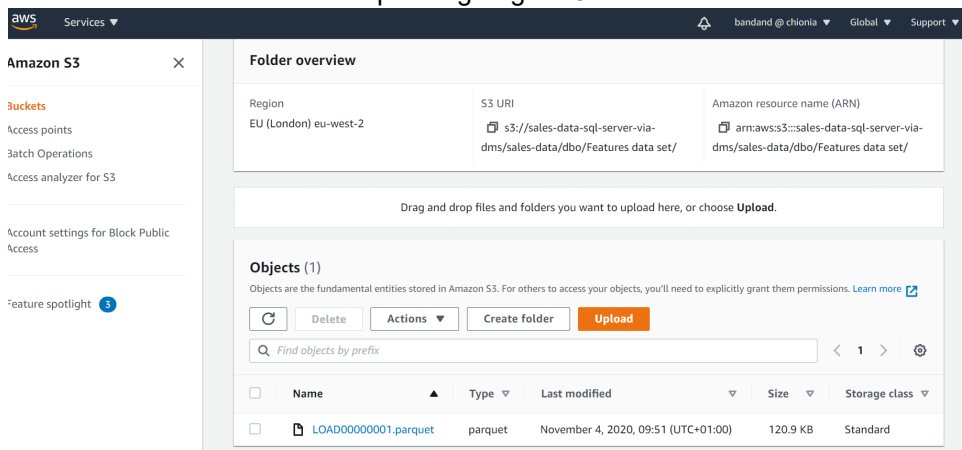
**SrcDbTableName**  
DB table name, which will be migrated. You may put % in case not to specify any particular table  
%

**DmsReplicationTaskMigrationType**  
The migration type. Valid values: full-load | cdc | full-load-and-cdc  
full-load-and-cdc

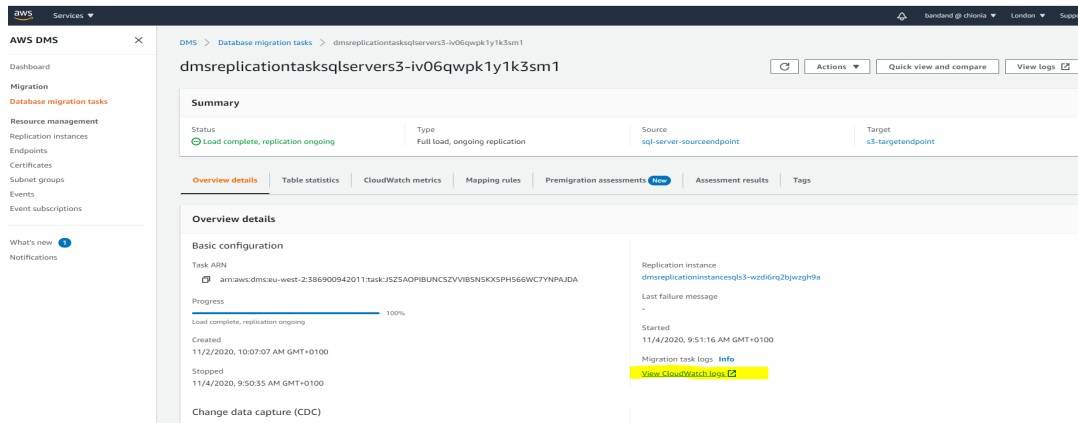
- 2) Then follow the instructions in the console and create the stack
- 3) This template will deploy the below resources
  - Source endpoint to connect to MS-SQL server
  - Target endpoint to S3
  - Service Role for DMS to use to load data to S3
  - Replication Instance
  - Data Migration Task
- 4) As per security best practices the following has been considered:
  - Data is encrypted in S3 via SSE\_S3
  - IAM roles created with minimum level of permission to be used by DMS service
  - Target S3 bucket has a policy to allow only limited access to a particular AWS principles ( here only the service role used by DMS).
  - Database credentials have been stored in secret manager
  - Logging and Monitoring data is being stored as cloudwatch logs
- 5) After the stack creation is complete, data migration task would be created in the DMS and would be in “Ready” state. Please start the task, it would be up and running and bring the data from source database to S3.



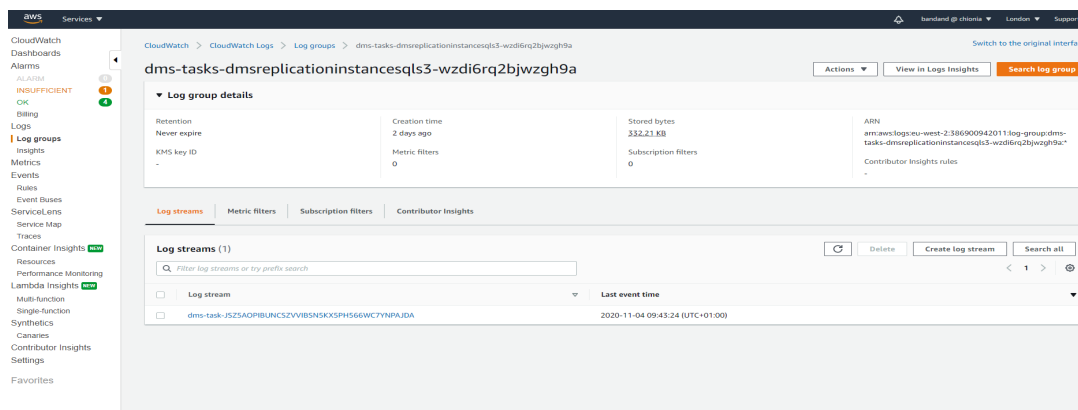
- 6) Data would be loaded to corresponding target S3 bucket.



- 7) The log for data migration task would be available in cloudwatch log. The link for the cloudwatch log group is available inside the database migration task.



After we click the link for cloudwatch log, it would take us to the details of the logs within cloudwatch



- 8) In similar fashion you can create the SageMaker Notebook required to analyse the data for our last step using the Sagemaker\_Notebook\_V1.yaml file.
- 9) Upload the data necessary for our analysis from your S3 target to SageMaker Notebook
- 10) Upload the retail-sales-forecast.ipynb on your Notebook to gain valuable insights from your data!

## Lessons Learned

From this example we can easily see that we can achieve all of the goals we set out in the start of our mission statement.

By fully migrating on AWS Cloud our customer has achieved scalability to petabytes while having a fully managed service to reduce cost on license and maintenance. By integrating with AWS Cloud Services our customer was able to gain new insights that otherwise would have taken a lot of money and effort to achieve.

There is no hard and fast rules on how you can achieve efficiency of your applications as you move to the cloud – but one thing is sure : either moving your entire application on AWS Cloud or just integrating with AWS Cloud services can prove to be AWSome! 😊