Migrating Your Legacy On-Prem SQL Server Databases to RDS

AWS offers many paths to migrating your on-premise SQL Server databases to AWS Relational Database Service (RDS). **AWS Database Migration Service** (DMS) is our primary migration choice, but in some cases your database migration is homogeneous and may be in the GBs not the TBs. In this scenario, DMS may be unnecessary overhead and cost compared to some alternative solutions, such as leveraging **S3, RDS** and **PowerShell** to migrate your data to AWS RDS.

The scenario here is you upload your SQL Server .BAK database backup files to S3, followed by executing CloudFormation and PowerShell scripts and being up and running in AWS RDS with minimal downtime of your database.



You will want to perform the migration during a period of inactivity or minimal database traffic. Below are the key aspects you need to factor in when deciding on your downtime window.

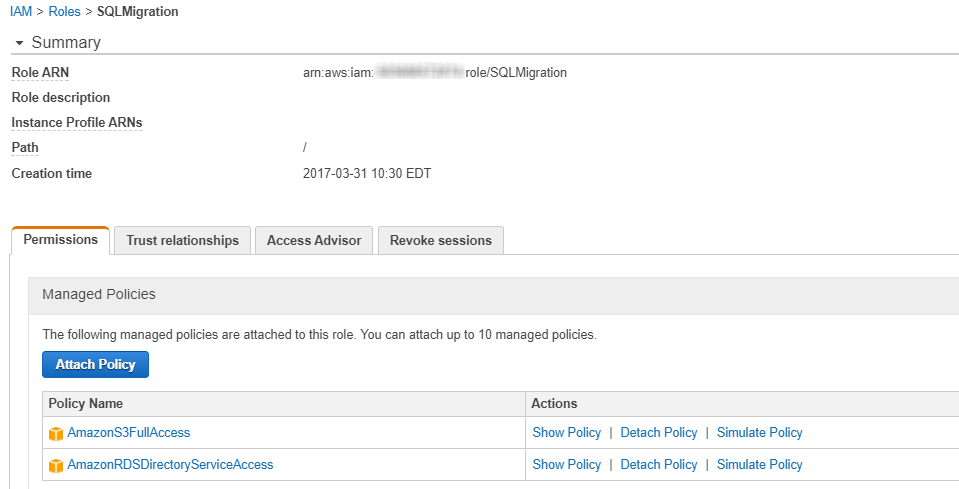
* **Network bandwidth**: Are you using AWS Direct Connect, VPN, etc. It is recommended to simulate an S3 upload of your .BAK file to get an estimate of your upload times.
* **Backup size:** The backup size may impact whether you need to strip the backup of into smaller chunks to improve the upload completion time and success.
  + **Uploading files greater than 5GB would require a multi-part upload. Please review this page for** [**further**](http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadingObjects.html) **details.**

# provision An Iam role with the correct policies

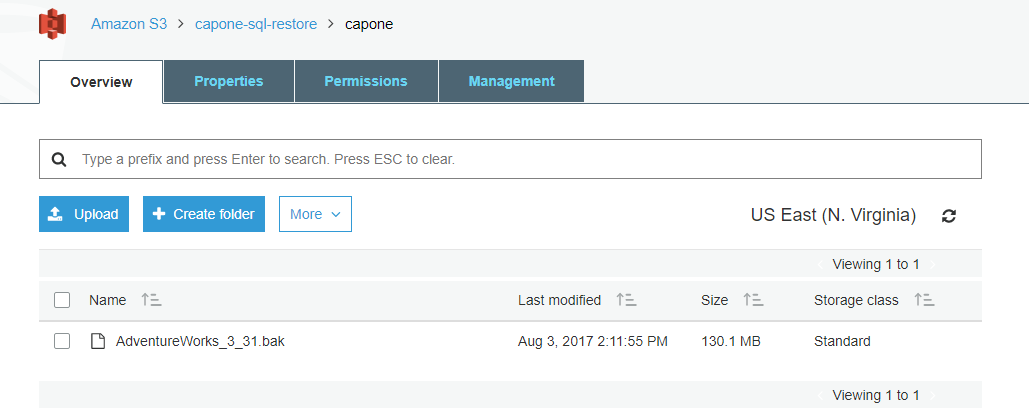
For this example my CloudFormation will create an IAM role named **SQLMigration**, which will be used to allow the RDS instance to be able to access the S3 bucket holding my the SQL Server backup (.bak) file.

For this demonstration I have given my role full policy rights for the following pre-defined policies, for greater security you should limit these policies in your own environments.

* AmazonS3FullAccess
* AmazonRDSDirectoryServicesAccess



# Upload your backup to s3

For this demonstration I have uploaded the sample AdventureWorks database. Ensure you upload your backup to a PRIVATE bucket, and we recommend encrypting your backup file(s).

#### To continue, the AWS Tools for Windows PowerShell must be installed on your windows computer (details can be found [here](https://aws.amazon.com/powershell/)). Connect to your AWS account using an authorized user and execute the command below to restore your database from S3.

#### PREREQUISITES:

* Backup uploaded to an accessible S3 bucket
* An IAM User which has required access to your AWS Account

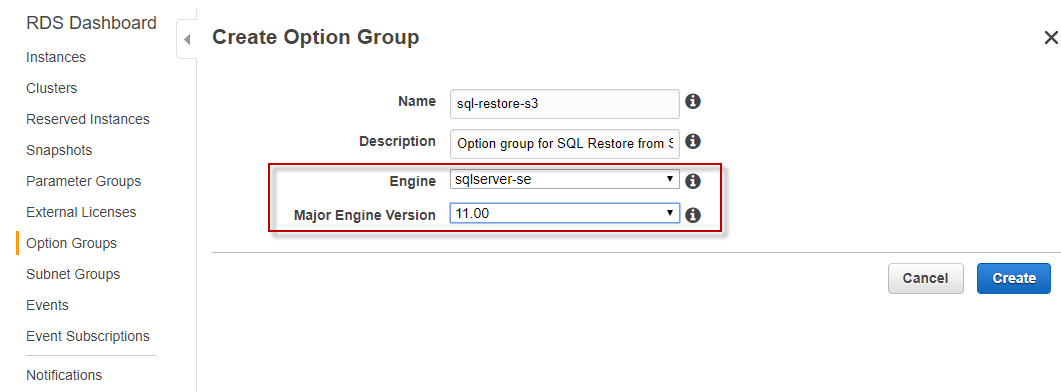
PowerShell Command:

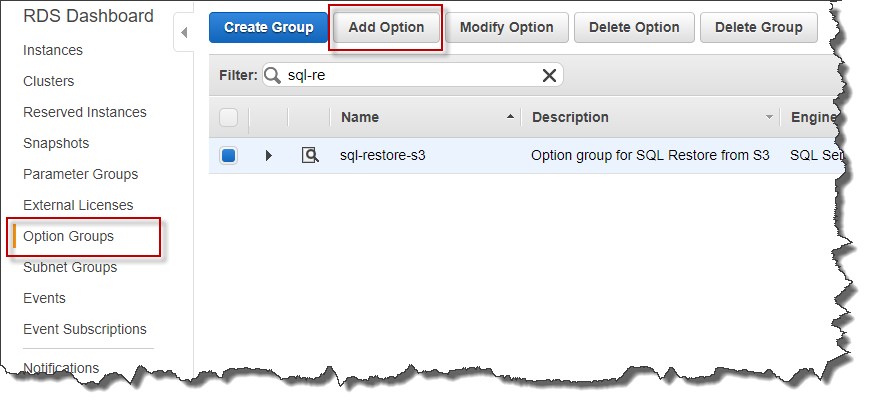


# Create an Option Group

CloudFormation will also create a key component of this solution with the RDS **Option Group**.

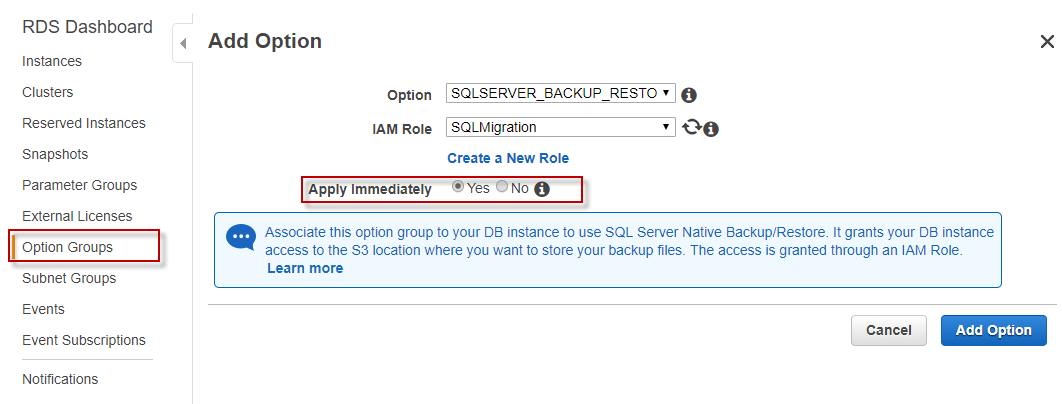
IMPORTANT: Ensure you select the correct Engine and Version as this is tied to the version of RDS SQL Server that is targeted.



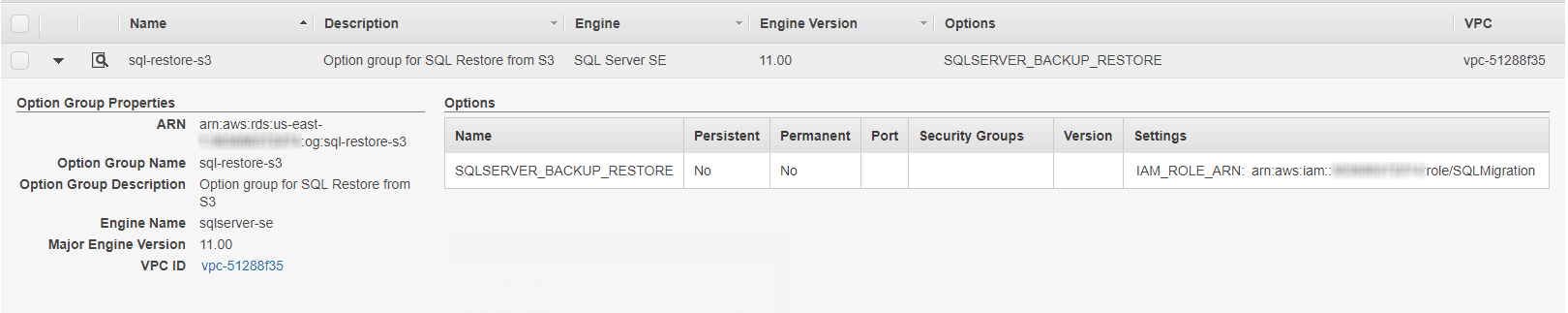


Select the Option: SQLSERVER\_BACKUP\_RESTORE, for the IAM Role created earlier.

IMPORTANT: It is recommended to select **Yes** to **Apply Immediately** to avoid having to wait to perform the AWS S3 restore before the next restart of your database server.



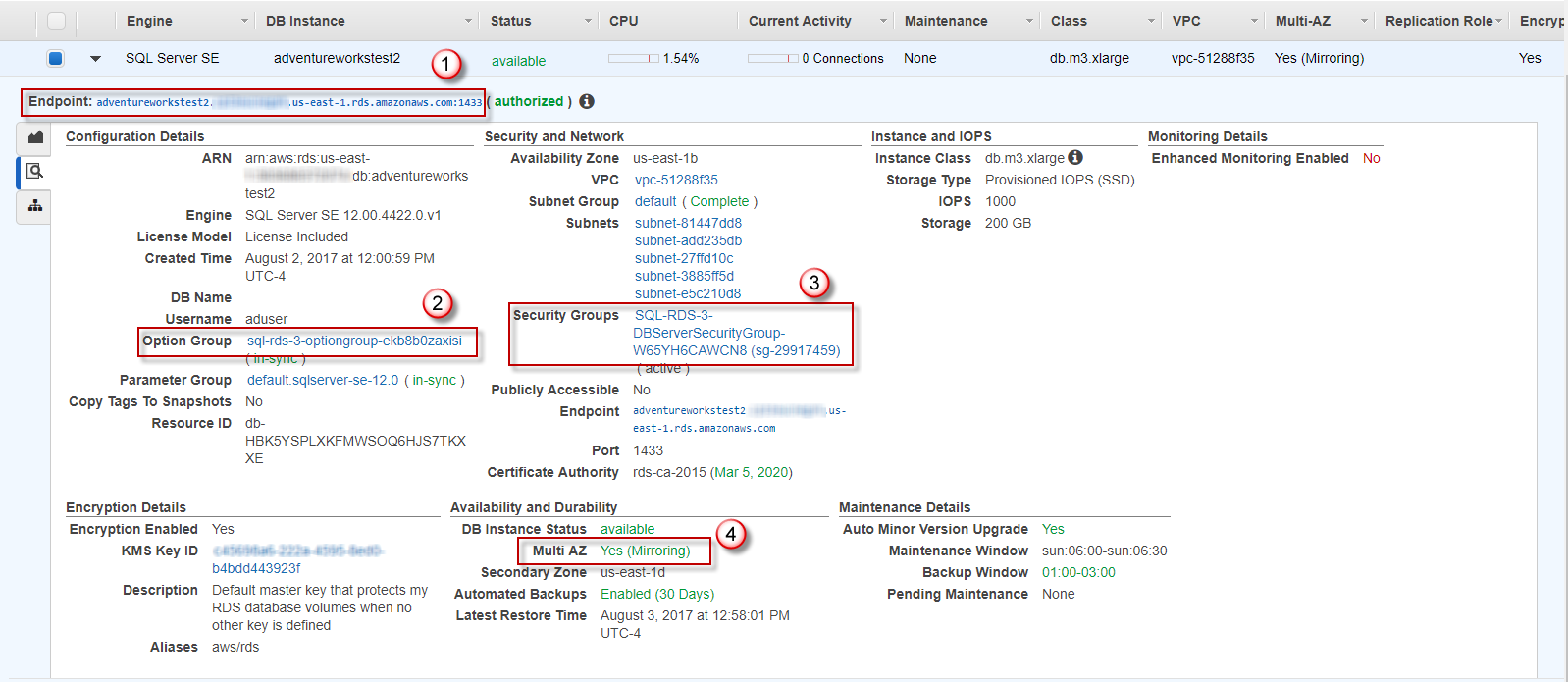
If successful you should see the role show up in Settings in the details under Options, by clicking the small arrow to the left of the Name.



# Launch the aws rds instance

Now launch the SQL Server RDS instance ensuring the Engine and Version match those assigned to the Option Group.

The image below highlights the key settings of the RDS SQL Server instance as shown in the AWS Console:

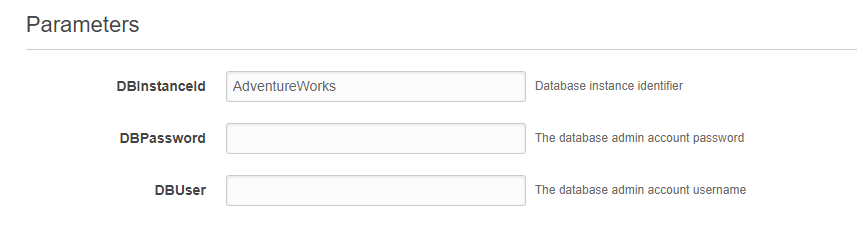


1. Endpoint: The host name and port for the RDS instance
2. Option Group: Used to allow S3 restores for a specified IAM role.
3. Security Groups: Ingress and Egress settings for the RDS instance
4. Multi AZ: For production instances a multi-AZ installation is essential. One could say the whole reason for moving to RDS. See [here](http://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html) for more details.

# Lets use the cloudformation to do all the above

Use the provided CloudFormation template simple-rds-restore-arn. This CloudFormation template automates the functionality as demonstrated manually in the earlier steps.

The template parameters are limited to the following:



Once the stack has been successfully created, the backup in S3 can be restored to RDS.

# RESTORE The Backup from S3

Using a PC with AWS Tools for Windows PowerShell installed, details can be found [here](https://aws.amazon.com/powershell/). Connect to your AWS account using an authorized user and execute the command below to restore your database from S3.

#### PREREQUISITES:

* Backup uploaded to an accessible S3 bucket
* An IAM User which has required access to your AWS Account

PowerShell Command:

