

Protrac to Smartabase data migration process flow

Step 1) Understand the data available in the Protrac database

Write a python script to connect to the Protrac database instance and using SQL write a query that lists all the tables and views within the database.

Save the list as a csv file or as a pdf so it can be used as a master reference document or data dictionary.

Services/ Software Used: Jupyter Notebook, Python.

Step 2) Determine what data you want to extract from the Protrac database

Using the list created, identify which tables and views have the data you wish to extract from the database.

Services/ Software Used:

Step 3) Add any new tables and views to the existing data extraction process

Using AWS DMS that is already configured and connected to the PgAdmin database, add any additional tables or views that you want extracted to the Client's data lake and set the Client Data Lake S3 location for where the data will be saved.

Source: PostgresDB

Target: S3

Services/ Software Used:

Step 4) Map the data tables and fields in Protrac to the Smartabase data structures that have been created

Create a data mapping table that lists the tables, views, and fields in Protrac to their matching tables and fields in Smartabase including any relationships or any data transformations required.

Services/ Software Used:

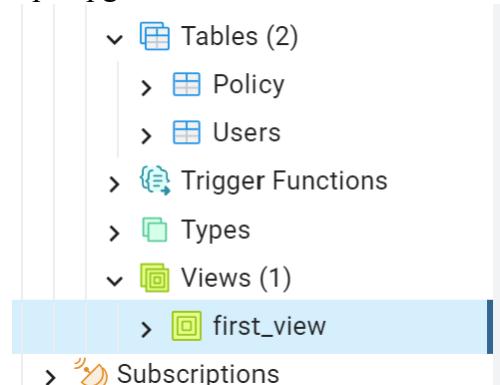
Step 5) Copy the Protrac data to Smartabase

- i. Obtain a long living Smartabase API token
- ii. Create a Python application that iterates through the mapping document created in step 4 and with each step, performs any data transformations and data joins using the data stored in the Swans Data Lake and maps it to the Smartabase structure. The application then connects to the Smartabase API using the token and calls the appropriate API's that will push the mapped data into Smartabase.

Services/ Software Used:

Step 1) Understand the data available in the Protrac database

Open pgAdmin and check all the tables and views within the PostgreSQL database.



Open Jupyter notebooks, create a .ipynb file to create a .csv file containing the list all the tables and views within the database.

Python Script:

```
pip install psycopg2-binary
pip install tablib
pip install reportlab
```

```
import psycopg2
import csv
from tablib import Dataset
from reportlab.lib.pagesizes import letter
from reportlab.pdfgen import canvas

# Database connection parameters
db_params = {
    'host': 'localhost',
    'database': 'schneider_db',
    'user': 'postgres',
    'password': 're*****',
}

def get_tables_and_views(connection):
    query = """
        SELECT table_name
    
```

```

        FROM information_schema.tables
        WHERE table_schema = 'public'
"""

with connection.cursor() as cursor:
    cursor.execute(query)
    tables = cursor.fetchall()
return tables

def save_to_csv(tables, filename='tables_list.csv'):
    with open(filename, 'w', newline='') as csvfile:
        csv_writer = csv.writer(csvfile)
        csv_writer.writerow(['Table/View'])
        csv_writer.writerows(tables)

def main():
    try:
        # Connect to the PostgreSQL database
        connection = psycopg2.connect(**db_params)

        # Retrieve tables and views
        tables = get_tables_and_views(connection)

        # Save to CSV
        save_to_csv(tables)

        print("Tables and views list saved successfully.")
        print(tables)

    except psycopg2.Error as e:
        print(f"Error: {e}")

    finally:
        if connection:
            connection.close()

if __name__ == "__main__":
    main()

```

OUTPUT:

```

Tables and views list saved successfully.
[('Policy',), ('Users',), ('first_view',)]

```

.CSV file content

jupyter tables_list.csv :

File Edit View Language

```
1 Table/View
2 Policy
3 Users
4 first_view
5
```

Note: The Python script should be running on the same local system where the pgAdmin (PostgreSQL) DBMS is.

Step 2) Determine what data you want to extract from the Protrac database

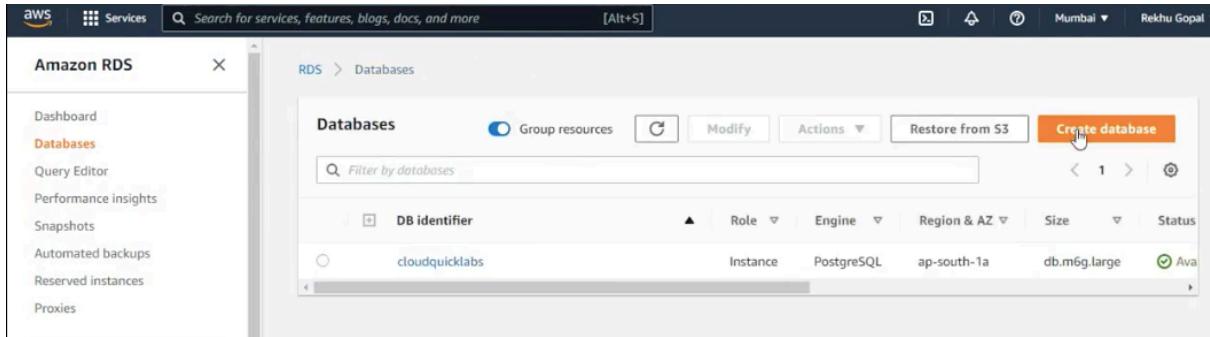
Using the list created in .CSV file above, identify which tables and views have the data you wish to extract from the database.

Step 3) Add any new tables and views to the existing data extraction process

Using AWS DMS that is already configured and connected to the Protrac database, add any additional tables or views that you want extracted to the Client's data lake and set the Client Data Lake S3 location for where the data will be saved.

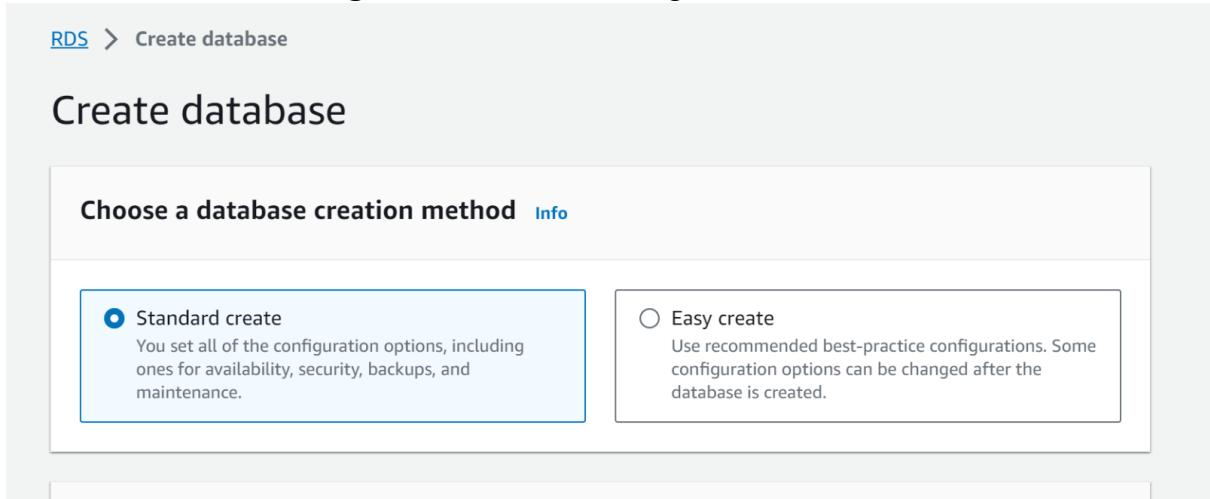
Source: PostgresDB

Target: S3



The screenshot shows the AWS RDS console with the 'Databases' page selected. On the left, there is a sidebar with links like Dashboard, Databases (which is highlighted), Query Editor, Performance insights, Snapshots, Automated backups, Reserved instances, and Proxies. The main area has a table titled 'Databases' with columns: DB identifier, Role, Engine, Region & AZ, Size, and Status. One row is visible: 'cloudquicklabs' (Instance: PostgreSQL, Region: ap-south-1a, Size: db.m6g.large, Status: Available). At the top right of the main area, there is a prominent orange 'Create database' button.

Go to **RDS** & start **creating a database** as following:



The screenshot shows the 'Create database' wizard. The first step is 'Choose a database creation method'. It offers two options: 'Standard create' (selected) and 'Easy create'. The 'Standard create' section includes a note: 'You set all of the configuration options, including ones for availability, security, backups, and maintenance.' The 'Easy create' section includes a note: 'Use recommended best-practice configurations. Some configuration options can be changed after the database is created.'

Engine options

Engine type [Info](#)

Aurora (MySQL Compatible)



Aurora (PostgreSQL Compatible)



MySQL



MariaDB



PostgreSQL



Oracle

ORACLE®

Microsoft SQL Server



IBM Db2

IBM Db2

Engine version [Info](#)

View the engine versions that support the following database features.

[▼ Hide filters](#)

Show versions that support the Multi-AZ DB cluster [Info](#)

Create a Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.

Engine Version

PostgreSQL 14.10-R1



Templates

Choose a sample template to meet your use case.

Production

Use defaults for high availability and fast, consistent performance.

Dev/Test

This instance is intended for development use outside of a production environment.

Free tier

Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with Amazon RDS.
[Info](#)

Availability and durability

Deployment options [Info](#)

The deployment options below are limited to those supported by the engine you selected above.

Multi-AZ DB Cluster

Creates a DB cluster with a primary DB instance and two readable standby DB instances, with each DB instance in a different Availability Zone (AZ). Provides high availability, data redundancy and increases capacity to serve read workloads.

Multi-AZ DB instance

Creates a primary DB instance and a standby DB instance in a different AZ. Provides high availability and data redundancy, but the standby DB instance doesn't support connections for read workloads.

Single DB instance

Creates a single DB instance with no standby DB instances.

Settings

DB instance identifier [Info](#)

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

database-1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ Credentials Settings

Master username [Info](#)

Type a login ID for the master user of your DB instance.

postgres

Public access [Info](#)

Yes

RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. Choose one or more VPC security groups that specify which resources can connect to the database.

No

RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

VPC security group (firewall) [Info](#)

Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

Choose existing

Choose existing VPC security groups

Create new

Create new VPC security group

Existing VPC security groups

Choose one or more options ▾

default 

Availability Zone [Info](#)

No preference ▾

RDS Proxy

RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.

Create an RDS Proxy [Info](#)

RDS automatically creates an IAM role and a Secrets Manager secret for the proxy. RDS Proxy has additional costs. For more information, see [Amazon RDS Proxy pricing](#).

Estimated Monthly costs

DB instance	153.30 USD
Storage	23.00 USD
Total	176.30 USD

This billing estimate is based on on-demand usage as described in [Amazon RDS Pricing](#). Estimate does not include costs for backup storage, IOs (if applicable), or data transfer.

Estimate your monthly costs for the DB Instance using the [AWS Simple Monthly Calculator](#).

 You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

Cancel

Create database

How to connect RDS instance with pgAdmin client:

The screenshot shows the Amazon RDS console interface. On the left, there's a sidebar with links like Dashboard, Databases, Query Editor, etc. The main area shows a summary for a database named 'database-1'. The summary table includes columns for DB identifier, Status, Role, Engine, and Recommendations. Below the summary is a navigation bar with tabs: Connectivity & security (which is selected), Monitoring, Logs & events, Configuration, Maintenance & backups, Tags, and Recommendations.

DB identifier	Status	Role	Engine	Recommendations
database-1	Backing-up	Instance	PostgreSQL	
CPU	Class	Current activity	Region & AZ	
	db.m5d.large	0.00 sessions	us-east-1c	

This modal window displays connection details for the database-1 instance. It includes a warning message about viewing the password, a link to learn about connecting, and fields for Master username (postgres) and Master password (rex[REDACTED] Copy). The Endpoint is listed as database-1.cwe[REDACTED]1xd.us-east-1.rds.amazonaws.com with a Copy button. A Close button is at the bottom right.

This is the only time you can view this password. Copy and save the password for your reference. If you lose the password, you must modify your database to change it. You can use a SQL client application or utility to connect to your database.

[Learn about connecting to your database](#)

Master username
postgres

Master password
rex[REDACTED] [Copy](#)

Endpoint
database-1.cwe[REDACTED]1xd.us-east-1.rds.amazonaws.com [Copy](#)

[Close](#)

Amazon RDS

Databases

DB identifier: database-1

Status: Backing-up

CPU: 4.64%

Role: Instance

Engine: PostgreSQL

Region & AZ: us-east-1c

Connectivity & security

Endpoint & port

Endpoint: database-1.cweq0nrqo1xd.us-east-1.rds.amazonaws.com

Port: 5432

Networking

Availability Zone: us-east-1c

VPC: vpc-0771777d

Security

VPC security groups: default (sg-f63a50d9) (Active)

Publicly accessible: Yes

Creating required Inbound rules for RDS Postgres instance:

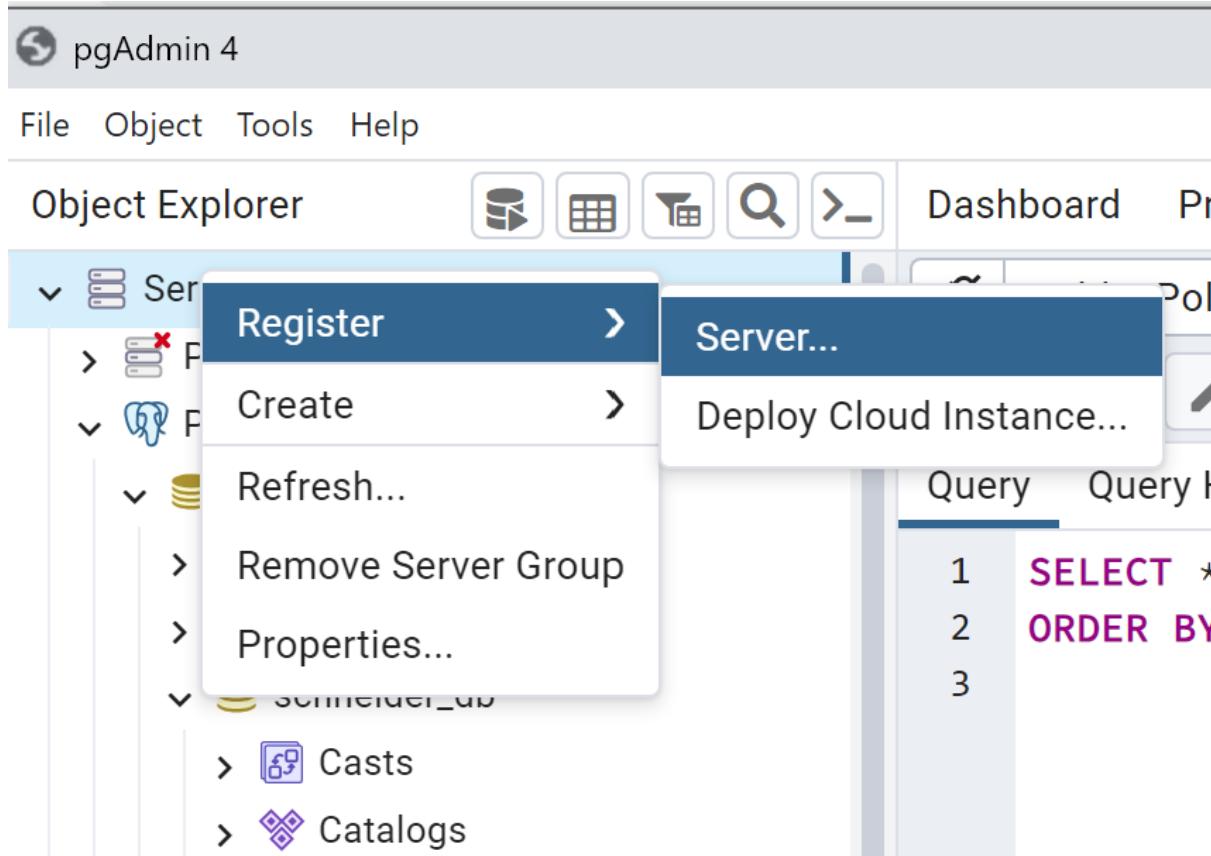
EC2 > Security Groups > sg-f63a50d9 - default > Edit inbound rules

Edit inbound rules

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-045b99da5005f0526	All TCP	TCP	0 - 65535	Cus... 0.0.0.0/0	0.0.0.0/0 X
sgr-045ff7d1097a6a9ef	All traffic	All	All	Cus... sg-f63a50d9	sg-f63a50d9 X

Go to pgAdmin, click Server



 Register - Server

General Connection Parameters SSH Tunnel Advanced

Name !

Server group  Servers | ▾

Background X

Foreground X

Connect now?

Comments

! 'Name' cannot be empty. X

i ? X Close Reset Save

Give name **Test**

 Register - Server

General Connection Parameters SSH Tunnel Advanced

Name

Server group  Servers

Background X

Foreground X

Go to Connection, paste Endpoint URL. Fill all details and click Save.

Register - Server

General Connection Parameters SSH Tunnel Advanced

Host name/address	database-1.cweq0nrqo1xd.us-east-1.rds.amazonaws.com
Port	5432
Maintenance database	postgres
Username	postgres
Kerberos authentication?	<input type="checkbox"/>
Password
Save password?	<input type="checkbox"/>
Role	
Service	

Buttons: Close Reset Save

You can see the Test connection created below

The screenshot shows the pgAdmin 4 interface. At the top is a toolbar with icons for File, Object, Tools, and Help. Below that is the Object Explorer title bar. The main pane displays a tree view under 'Servers (4)'. The 'PostgreSQL 14' server is selected, highlighted with a blue background. The other servers listed are 'PostgreSQL 12', 'PostgreSQL 16', and 'Test'. Each server entry has a small icon to its left.

As we can see above, our **Test** server is created

Now, as we need to do the **CDC (change data capture)** to create a **Parameter Group**

The screenshot shows the AWS Lambda 'Custom parameter groups' page. At the top, it says 'Custom parameter groups (0)' with a 'Create parameter group' button. Below is a search bar labeled 'Filter by custom parameter groups'. A table follows with columns: Name, Family, Type, Description, and ARN. The table is currently empty, displaying the message 'No parameter group found'.

Create parameter group

Parameter group details

Parameter group family
DB family that this DB parameter group will apply to

Type
Type for the DB parameter group

Group Name
Identifier for the DB parameter group

Description
Description for the DB parameter group

Cancel **Create**

Go into recently created Parameter Group

Parameter groups Info

Custom **Default**

Custom parameter groups (1)

<input type="checkbox"/>	Name	Family	Type	Description	ARN		Actions <small>▼</small>	Create parameter group
<input type="checkbox"/>	testdemo	aurora-postgresql14	DB instance parameter group	Test	arn:aws:rds:us-east-1:646975365807:pg:testdemo			

Open the **RDS Database** & attach the recently created **Parameter Group** to it.

RDS > Databases > Modify DB instance: database-1

Modify DB instance: database-1

Settings

DB engine version

Version number of the database engine to be used for this database

14.10

DB instance identifier [Info](#)

Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in the current AWS Region.

database-1

The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

Manage master credentials in AWS Secrets Manager

Manages master user credentials in Secrets Manager. RDS can generate a password for you and store it in Secrets Manager.

▼ Additional configuration

Database options, backup turned on, Performance Insights turned on, Enhanced Monitoring turned on, maintenance, CloudWatch Logs, delete protection turned off

Database options

DB parameter group [Info](#)

default.postgres14

default.postgres14

Enable automated backups

Creates a point-in-time snapshot of your database

Reboot the RDS Database

RDS > Databases

Databases

Group resources



Modify

Actions ▲

Restore from S3

Create database

Filter by databases

DB identifier

▲ Role ▼

cloudquicklabs

Instance

Stop temporarily

Reboot

Delete

Create read replica

Create Aurora read replica

Promote

Take snapshot

Restore to point in time

Migrate snapshot

< 1 >

Z ▾

Size

▼

Status

a

db.m6g.large

Available

Open pgAdmin & create the Table with SQL commands

Query Query History

```
1  create table demo (
2      industry_name_anzsic varchar(100)      not null,
3      rme_size_grp         varchar(100)      not null,
4      variables            varchar(100)      not null
5  )
6
7
8  INSERT INTO demo (industry_name_anzsic, rme_size_grp, variables)
9  VALUES ('value1', 10, 'value3');
10
11 INSERT INTO demo (industry_name_anzsic, rme_size_grp, variables)
12 VALUES ('value1', 20, 'value3');
13
14 INSERT INTO demo (industry_name_anzsic, rme_size_grp, variables)
15 VALUES ('value1', 30, 'value3');
```

Data Output Messages Notifications

INSERT 0 1

Query returned successfully in 83 msec.

Github link:

https://github.com/RekhuGopal/PythonHacks/blob/main/AWS_RDS_Data_Migration_CDC_to_S3_Using_DMS/Create_Update.sql

Now, query the recently created tables

pgAdmin 4

File Object Tools Help

Object Explorer Dashboard Properties SQL Statistics Dependencies Dependents postgres < >

Event Triggers
Extensions
Foreign Data Wrappers
Languages
Publications
Schemas (1)
public
Aggregates
Collations
Domains
FTS Configurations
FTS Dictionaries
FTS Parsers
FTS Templates
Foreign Tables
Functions
Materialized Views
Operators
Procedures
Sequences
Tables (4)
dpstgresqlcdc
dpstgresqlcdc
dpstgresqlcdc
employees
Trigger Functions
Types
Views
Subscriptions
rdsadmin
Login/Group Roles

public.dpstgresqlcdc/postgres@Test

No limit

Query History

1 `SELECT * FROM public.dpstgresqlcdc`

Data Output Messages Notifications

	industry_name_anzsic	rme_size_grp	variables
1	ANZ	15	A
2	CBA	10	B
3	westpac	20	C
4	ANZ	15	A
5	CBA	10	B
6	westpac	20	C

Now, create a **subnet group** with all the AZs available

AWS Services Search for services, features, blogs, docs, and more [Alt+S] Mumbai Reku Gopal

AWS DMS

DMS Studio New

Dashboard Database migration tasks Replication instances Endpoints Certificates Subnet groups

DMS > Subnet groups

Subnet groups (1)

Create subnet group

Name	Status	VPC ID
rdsposgresscdc	Complete	vpc-9c193af4

Amazon RDS

RDS > Subnet groups > default-vpc-0771777d

default-vpc-0771777d

Subnet group details

VPC ID	vpc-0771777d
ARN	arn:aws:rds:us-east-1:646975365807:subgrp:default-vpc-0771777d
Supported network types	IPv4
Description	Created from the RDS Management Console

Subnets (6)

Availability zone	Subnet ID	CIDR block
us-east-1a	subnet-d20bfc8d	172.31.32.0/20
us-east-1b	subnet-82e903e4	172.31.0.0/20
us-east-1c	subnet-8e40b1af	172.31.80.0/20
us-east-1e	subnet-da95abe4	172.31.48.0/20
us-east-1f	subnet-3e7fd330	172.31.64.0/20
us-east-1d	subnet-b2368cff	172.31.16.0/20

Now, create a Replication Instance

AWS DMS

DMS > Replication instances

Upgrades to versions 3.4.7 and higher
 You have 1 instance that uses AWS DMS version 3.4.7. Upgrades to AWS DMS versions 3.4.7 and higher require that you configure AWS DMS to use VPC endpoints or use public routes. This requirement applies to source and target endpoints for these data stores: S3, Kinesis, Secrets Manager, DynamoDB, Amazon Redshift, and OpenSearch Service.
[Learn more](#)

Replication instances (1)

Name	Status	VPC	Class	Engine version	Availability zone	Public
mypostgresqldemo	Available	vpc-9c193af4	dms.t3.medium	3.4.7	ap-south-1b	Yes

Replication instances (1)

Name	Status	VPC	Class	Engine version	Availability zone	Network type
testswans	Available	vpc-0771... dms.t3.m...	3.5.2	us-east-1f	IPv4	

AWS DMS

DMS > Replication instances > testswans

testswans

Status: Available | Class: dms.t3.medium | Engine version: 3.5.2 | Associated migration tasks: 1

Overview details | CloudWatch metrics | Migration tasks | Log management | Tags

Details

Basic configuration	Connectivity and security	Maintenance
ARN: arn:aws:dms:us-east-1:646975365807:rep:test-swans-er	Network type: IPv4	Minor version automatic upgrade: Yes
Instance class: dms.t3.medium	VPC: vpc-0771777d	Pending modify values: -
Status: View details	Replication subnet group: View details	Maintenance window: Jan 21 2024 05:01 (UTC-11:00) - 05:31

Now, create Source & Target Endpoints

AWS DMS

DMS > Endpoints

Endpoints (2)

	Name	Type	Status	Engine	Server name	Port
<input type="checkbox"/>	database-1	Source	Active	PostgreSQL	database-1.cweq0nrqo1xd.us-east-1.rds.amazonaws.com	5432
<input type="checkbox"/>	test	Target	Active	Amazon S3	-	-

DMS > Endpoints > database-1

database-1

Actions ▾

Overview details Connections Schemas Endpoint settings Tags

Details	
Endpoint ARN	Status
arn:aws:dms:us-east-1:646975365807:endpoint:test-a-1	<input checked="" type="checkbox"/> Active
Endpoint engine	SSL mode
PostgreSQL	None
Endpoint type	Server name
Source	database-1.cweq0nrqo1xd.us-east-1.rds.amazonaws.com
User name	Port
postgres	5432
KMS ARN / ID	Database name
arn:aws:kms:us-east-1:646975365807:key/7b0e482e-242b-4ecc-957b-ae568c41c5c8	postgres

DMS > Endpoints > Create endpoint

Create endpoint [Info](#)

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 stores such as Amazon DynamoDB or Kinesis.

Select RDS DB instance
Choose this option if the endpoint is an Amazon RDS DB instance. It provides a list of available RDS Instances from the current region.

RDS Instance
Instances available only for current user and region

database-1

Endpoint configuration

Endpoint identifier [Info](#)
A label for the endpoint to help you identify it.

database-1

Source engine
The type of database engine this endpoint is connected to. [Learn more](#)

PostgreSQL

Access to endpoint database | [Info](#)

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:secret::123456789012

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::123456789012

Secure Socket Layer (SSL) mode | [Info](#)
The type of Secure Socket Layer enforcement

none

Database name

postgres

▼ Endpoint settings

DMS > Endpoints > test	
test	
Overview details Connections Endpoint settings Tags	
Details	
Endpoint ARN	Status
arn:aws:dms:us-east-1:646975365807:endpoint:NO77U2A3SKCSZ4L432PC6RGPU7USKUNG5KI2WRY	 Active
Endpoint engine	SSL mode
Amazon S3	None
Endpoint type	Extra connection attributes
Target	bucketName=dmsprotractest;compressionType=NONE;csvDelimiter=,;csvRowDelimiter=\n;datePartitionEnabled=false;
Amazon Resource Name (ARN) for service access role	Bucket name
arn:aws:iam::646975365807:role/DMSProtrac	dmsprotractest

Endpoint configuration

Endpoint identifier | [Info](#)
A label for the endpoint to help you identify it.

Descriptive Amazon Resource Name (ARN) - optional
A friendly name to override the default DMS ARN. You cannot modify it after creation.

Target engine
The type of database engine this endpoint is connected to. [Learn more](#)

Amazon Resource Name (ARN) for service access role
Role that can access target.

Bucket name
The name of an Amazon S3 bucket where DMS will read the files from

Bucket folder
The Amazon S3 bucket path where the CSV files can be found

Copy the S3 Bucket name, that you created:

Amazon S3 X [Amazon S3](#) > Buckets

Buckets

- Access Grants
- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- IAM Access Analyzer for S3

Block Public Access settings for this account

▼ Storage Lens

- Dashboards
- Storage Lens groups
- AWS Organizations settings

Feature spotlight

Account snapshot
Last updated: Jan 15, 2024 by Storage Lens. Metrics are generated every 24 hours. Metrics don't include directory buckets. [Learn more](#)

Total storage	Object count	Average object size
89.6 GB	478.7 k	196.3 KB

You can enable advanced metrics in the "default-account-dashboard" configuration.

General purpose buckets (169) [Info](#)
Buckets are containers for data stored in S3. [Learn more](#)

Name	AWS Region	Access	Creation date
dmsprotractest	US East (N. Virginia) us-east-1	Objects can be public	January 16, 2024, 00:12:37 (UTC+11:00)

We need to create or attach an Amazon Resource Name(ARN), to create go to IAM Roles:
As we created one below

Identity and Access Management (IAM)

Search IAM

- Dashboard
- Access management**
 - User groups
 - Users
 - Roles**
 - Policies
 - Identity providers
 - Account settings
- Access reports**
 - Access Analyzer
 - External access
 - Unused access
 - Analyzer settings
 - Credential report
 - Organization activity
 - Service control policies (SCPs)

DMSProtrac Info

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

Summary

Creation date: January 16, 2024, 00:03 (UTC+11:00) ARN: arn:aws:iam::646975365807:role/DMSProtrac

Last activity: 11 hours ago Maximum session duration: 1 hour

Permissions | Trust relationships | Tags | Access Advisor | Revoke sessions

Permissions policies (2) Info

You can attach up to 10 managed policies.

Filter by Type		
Search	All types	
<input type="checkbox"/> Policy name	Type	Attached entities
<input type="checkbox"/> AdministratorAccess	AWS managed - job function	35
<input type="checkbox"/> AmazonS3FullAccess	AWS managed	30

Get the ARN of this Role and paste above.

Finally, we can create our Data Migration Task:

AWS DMS

DMS Studio New

- Dashboard
- Database migration tasks**
- Replication instances
- Endpoints
- Certificates
- Subnet groups
- Events
- Event subscriptions

DMS > Database migration tasks

Database migration tasks (1)

Create task

Identifier	Status	Progress	Type	Source	Target	Replication instance
migratedb	Load complete	100%	Full load	cloudquicklabs	ddctargetendpoint	mypostgresqldemo

AWS DMS

- Dashboard
- Discover**
 - Data collectors
 - Inventory
- Assess**
 - Recommendations New
- Convert and migrate**
 - Migration projects New
 - Instance profiles New
 - Data providers New
- Migrate data**
 - Replication instances
 - Endpoints
- Database migration tasks**
- Serverless replications

DMS > Database migration tasks

New feature announcements

Consider the following new features to simplify and accelerate your migrations:

- Fleet Advisor recommendations - Generate right-sizing recommendations for migrating to Amazon Aurora or Amazon Relational Database Service (RDS). [Learn more](#)
- DMS schema conversion - Automatically convert source database schemas to a format that is compatible with the target database. [Learn more](#)
- Homogenous data migrations - Leverage built-in native database tooling in DMS for easy and performant like-to-like migrations. [Learn more](#)

Database migration tasks (1)

Create task

Identifier	Status	Progress	Type	Source	Target	Replication instance
dmsprotrec	Load complete	100%	Full load	database-1	test	testswans

AWS DMS

Create database migration task [Info](#)

Task configuration

Task identifier: test

Descriptive Amazon Resource Name (ARN) - optional
A friendly name to override the default DMS ARN. You cannot modify it after creation.
Friendly-ARN-name

Replication instance: testswans - vpc-0771777d

Source database endpoint: database-1

Target database endpoint: test

Migration type [Info](#): Migrate existing data and replicate ongoing changes

AWS DMS

migration task. [Info](#)

▼ where schema name is like 'public' and Source table name is like '%', include

Schema: Enter a schema

Source name: Use the % character as a wildcard
public

Source table name: Use the % character as a wildcard
%

Action: Choose "Include" to migrate your selected objects, or "Exclude" to ignore them during the migration.
Include

Source filters [Info](#) Add column filter

Now, Restart the Data Migration Task again

DMS > Database migration tasks

New feature announcements

Consider the following new features to simplify and accelerate your migrations:

- Fleet Advisor recommendations - Generate right-sizing recommendations for migrating to Amazon Aurora or Amazon Relational Database Service (RDS). [Learn more](#)
- DMS schema conversion - Automatically convert source database schemas to a format that is compatible with the target database. [Learn more](#)
- Homogenous data migrations - Leverage built-in native database tooling in DMS for easy and performant like-to-like migrations. [Learn more](#)

Database migration tasks (1/1)

Identifier	Status	Progress	Type	Replication instance	Start
dmsprotrec	Load complete	100%	Full	testswans	Ja

Actions ▾ **Quick view and compare** **Create task**

[Learn more](#)

Start task dmsprotrec

Resume
Resume the full load task from the last stopped point.

Restart
Restart the full load task from the beginning.

Cancel **Start task**

DMS > Database migration tasks

Database migration tasks (1/1)

Identifier	Status	Progress	Type	Source	Target	Replication instance	St
dmsprotrec	Load complete	100%	Full load	database-1	test	testswans	Ja

Finally, the load is completed

At the end, we can check the S3 Bucket to see if the Tables are migrated or not.

Amazon S3 > Buckets > dmsprotractest > public/

public/

Objects

Properties

Objects (4) Info

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your permissions. [Learn more](#)



Copy S3 URI



Copy URL



Download

Open

Delete



Find objects by prefix

Name

▲

Type

▼

Last modified

dspostgresqlcdc/

Folder

-

dspostgresqlcdc2/

Folder

-

dspostgresqlcdc3/

Folder

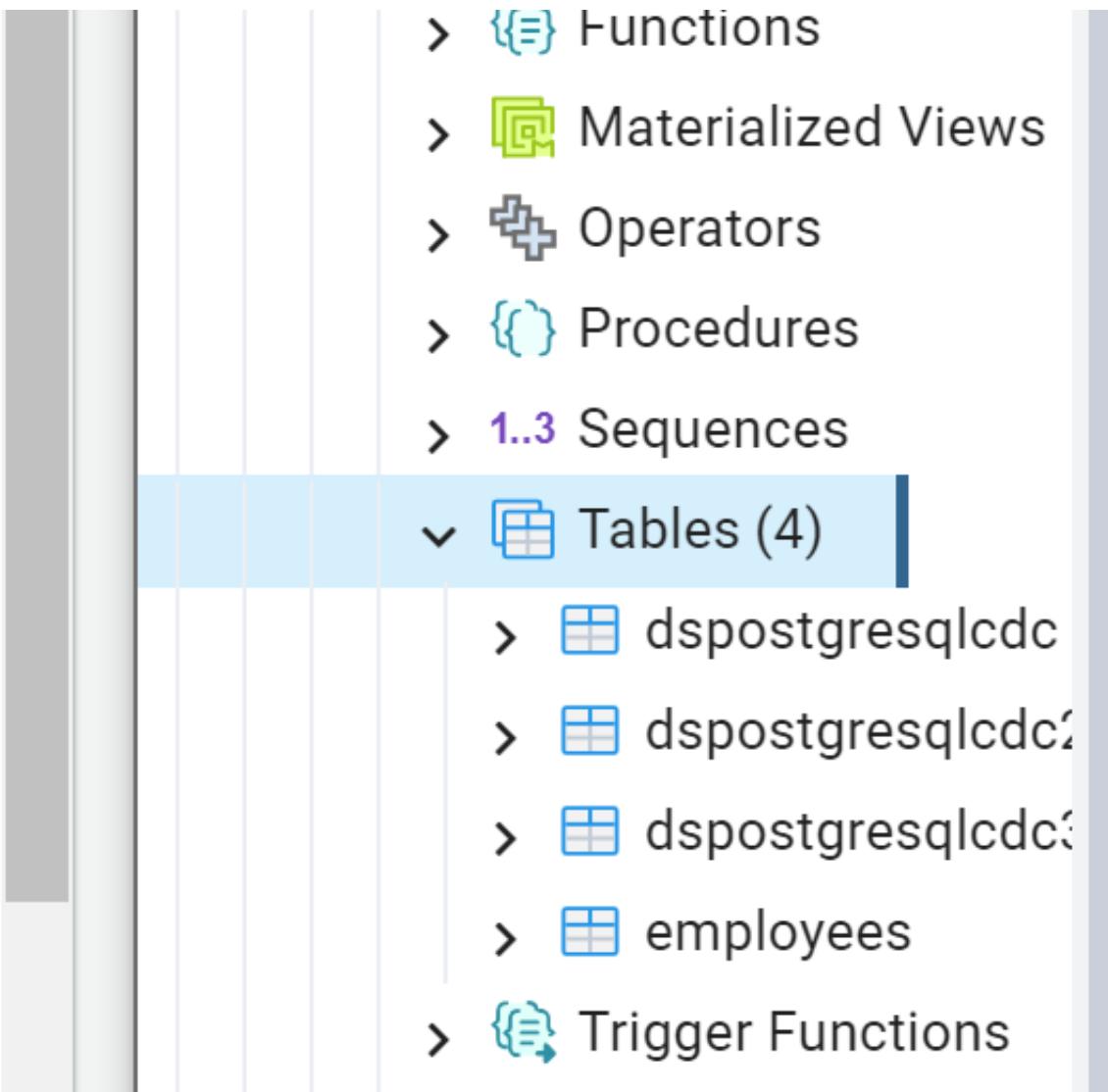
-

employees/

Folder

-

Also, cross-check with pgAdmin tables



Hence, the tables have been migrated to S3 from pgAdmin.

Step 4) Map the data tables and fields in Protrac to the Smartabase data structures that have been created

Source table	Source field	Smartabase table	Smartabase field	Data transformations	Mapping rules	Other notes
table1	field 1	table1	field 1		needs to be lowercase and special characters removed	
table1	field 2	table1	field 4		make	
table1	field 3	table1	field 3 +5			
table1	field 4	table1	field 4			
table1	field 5	table1	field 5			
table1	field 6	table1	field 6			
table1	field 7	table1	field 7			

Source table	Source field	Smartabase table	Smartabase field
table1	field 1	table1	field 1
table1	field 2	table1	field 4
table1	field 3	table1	field 3 +5
table1	field 4	table1	field 4
table1	field 5	table1	field 5

Data transformations	Mapping rules	Other notes
	needs to be lowercase and special characters removed make	source.table1 needs to be joined to source.table2 by fields table1.field3 and tabl2.field5

Step 5) Copy the Protrac data to Smartabase

To achieve the described process in a Python application, you would typically use libraries such as `pandas` for data manipulation, `boto3` for interacting with Amazon S3, and `requests` for making API calls to Smartabase. Below is a basic example of how you might structure this application:

1. Install required libraries:

```
| pip install pandas openpyxl boto3 requests |
```

2. Create a Python script (`mapping_application.py`):

```
import pandas as pd
import boto3
import requests
import json

# Load the mapping document (assuming it's an Excel file)
mapping_document_path = 'mapping_document.xlsx'
mapping_df = pd.read_excel(mapping_document_path)

# Function to perform data transformations
def perform_data_transformations(data):
    # Implement your data transformations logic here
    # Example: Convert date formats, clean data, etc.
    return data

# Function to perform data joins
def perform_data_joins(data1, data2):
    # Implement your data join logic here
    # Example: Use pandas merge or join functions
    merged_data = pd.merge(data1, data2, on='common_column', how='inner')
    return merged_data

# Function to fetch data from S3 Data Lake
def fetch_data_from_s3(bucket_name, object_key):
    s3 = boto3.client('s3')
    response = s3.get_object(Bucket=bucket_name, Key=object_key)
    data = pd.read_csv(response['Body'])
    return data

# Connect to the Smartabase API
def connect_to_smartabase(api_url, api_key):
    auth_url = f"{api_url}/auth/token"
    headers = {
        'Authorization': f'Bearer {api_key}',
        'Content-Type': 'application/x-www-form-urlencoded',
    }

    response = requests.post(auth_url, headers=headers)

    if response.status_code == 200:
        access_token = response.json().get('access_token')
        return access_token
    else:
```

```

print(f"Authentication failed. Status code: {response.status_code}")
return None

# Function to push data to Smartabase
def push_data_to_smartabase(api_url, endpoint, access_token, data):
    push_url = f"{api_url}/{endpoint}"
    headers = {
        'Authorization': f'Bearer {access_token}',
        'Content-Type': 'application/json',
    }

    response = requests.post(push_url, headers=headers, data=json.dumps(data))

    if response.status_code == 200:
        print("Data pushed to Smartabase successfully.")
    else:
        print(f"Failed to push data to Smartabase. Status code: {response.status_code}")
        print(response.text)

# Iterate through the mapping document
for index, row in mapping_df.iterrows():
    # Fetch data from S3 Data Lake
    s3_bucket = 'your_s3_bucket_name'
    s3_key = 'your_data_file.csv'
    s3_data = fetch_data_from_s3(s3_bucket, s3_key)

    # Perform data transformations and joins
    transformed_data = perform_data_transformations(s3_data)
    joined_data = perform_data_joins(transformed_data, other_data)

    # Map data to Smartabase structure (assuming there is a mapping logic)
    mapped_data = {
        'smartabase_field1': joined_data['data1_column'],
        'smartabase_field2': joined_data['data2_column'],
        # Add more fields as needed
    }

    # Connect to Smartabase API
    smartabase_api_url = "https://api.smartabase.com"
    smartabase_api_key = "your_api_key" # Replace with your Smartabase API key
    access_token = connect_to_smartabase(smartabase_api_url, smartabase_api_key)

    if access_token:

```

```
# Push mapped data to Smartabase
push_data_to_smartabase(smartabase_api_url, 'your_api_endpoint',
access_token, mapped_data)
```

Replace placeholders like `'your_s3_bucket_name'`, `'your_data_file.csv'`, `'your_api_key'`, and `'your_api_endpoint'` with your actual S3 bucket name, data file in S3, Smartabase API key, and API endpoint.

This script assumes that your mapping document is stored in an Excel file (`mapping_document.xlsx`) and your data is in a CSV file in the S3 Data Lake. Customize the data transformation, data join, and mapping logic based on your specific requirements. Always ensure proper error handling and validation in a production environment.