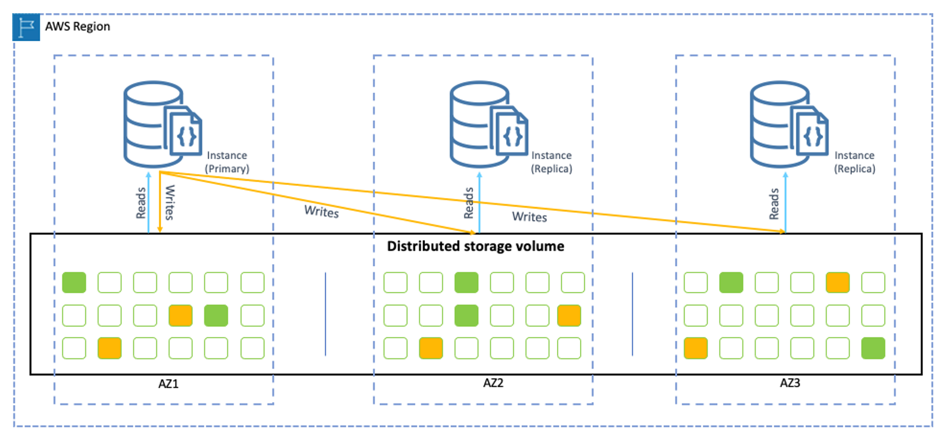
# Lab 5: Working with Amazon DocumentDB databases

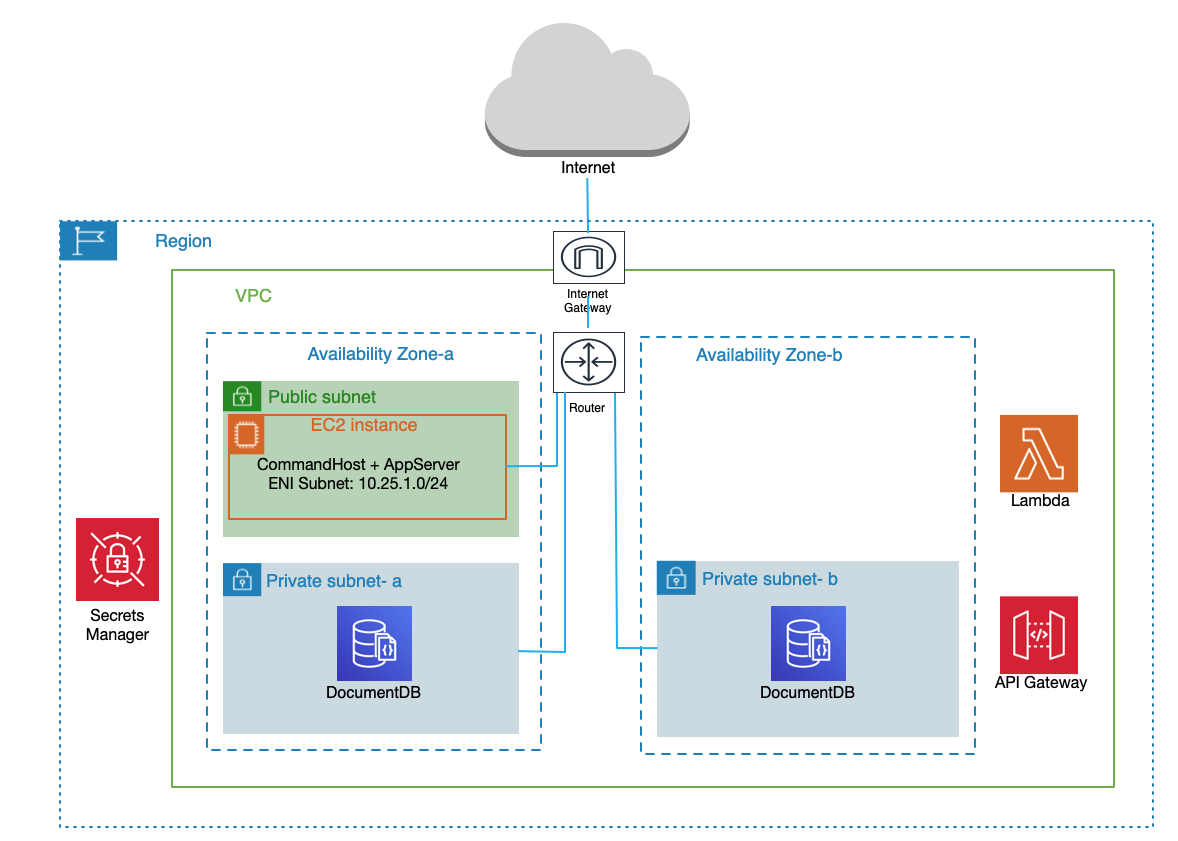
Objs-

1. Evaluate relational and nonrelational data.
2. Load and verify data in Amazon DocumentDB.
3. Run complex queries.
4. Monitor health and performance of Amazon DocumentDB.
5. Backup DocumentDB cluster using manual snapshot.

Nonrelational Databases (NoSQL): These use different ways to store and manage data, such as documents, key-value pairs, or graphs. They are designed to handle large volumes of data, be fast, and offer flexibility in data models.

DocDB is NRDB





Public Subnet:

The EC2 instance can communicate with the internet through the internet gateway.

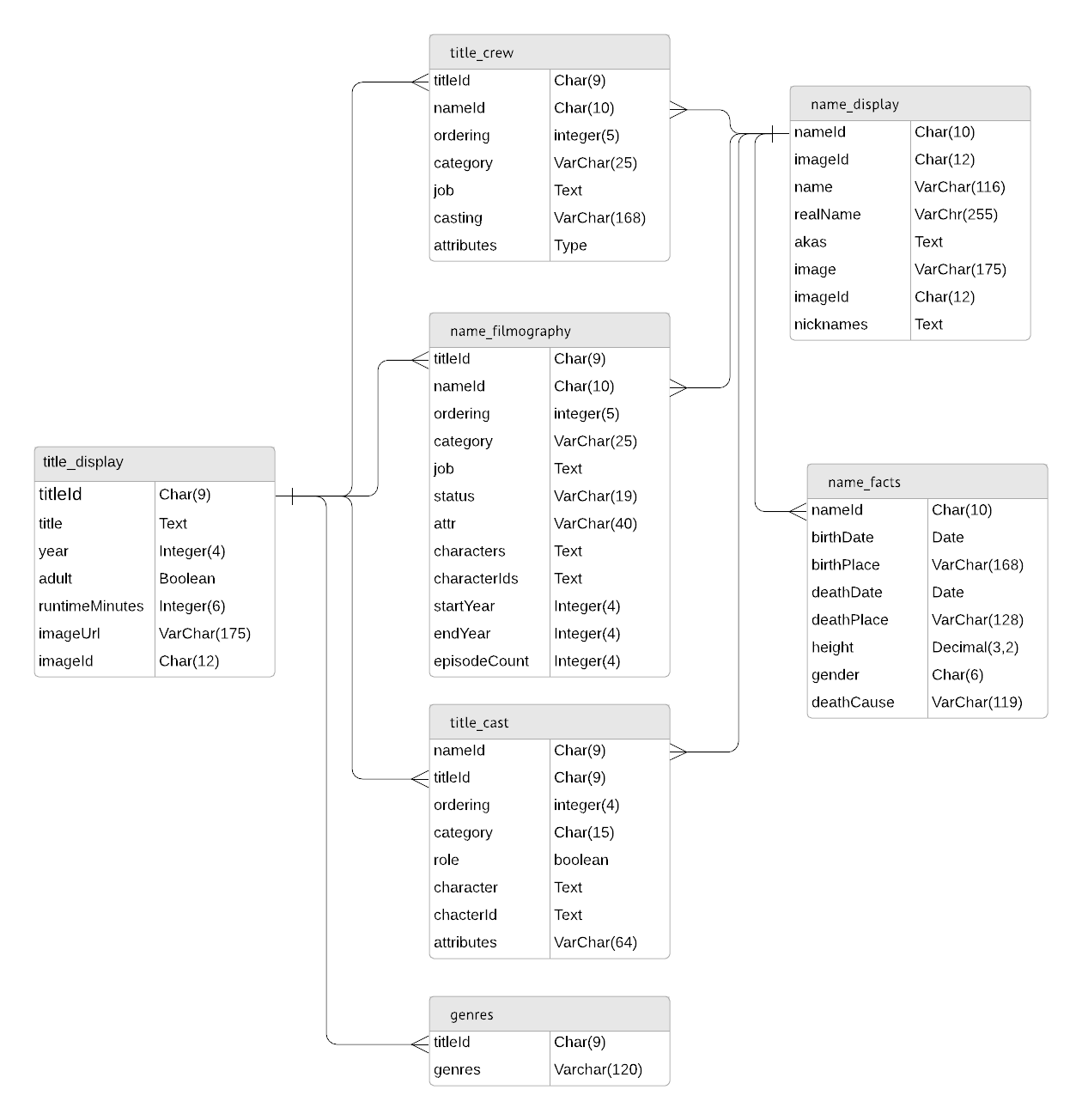
The instance can also interact with the DocumentDB instances in the private subnets.

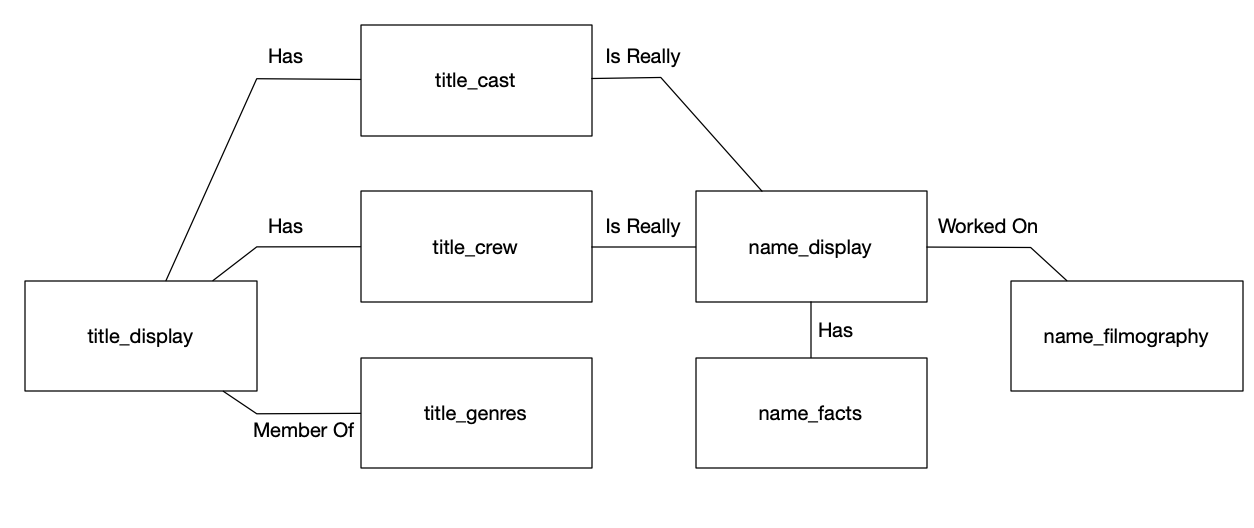
High avlbl-ty due to 2 AZ

Load Data: Use the EC2 instance to import data into DocumentDB.

Pre-lab -

Convert relational data to a nonrelational JSON object



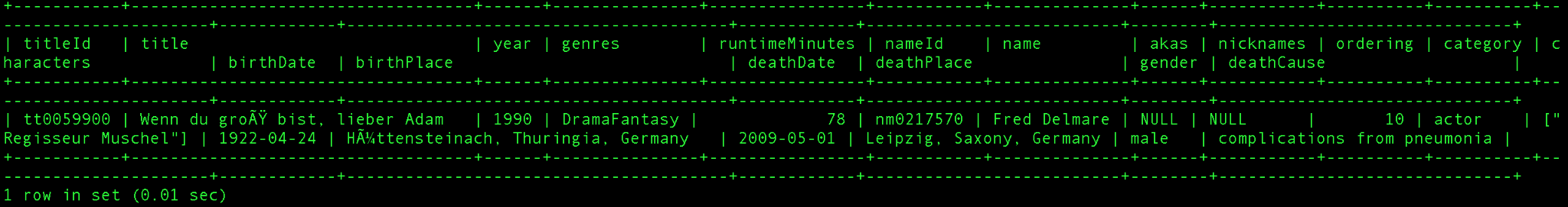


Nonrelational databases rely upon access patterns to determine the way data is organized.

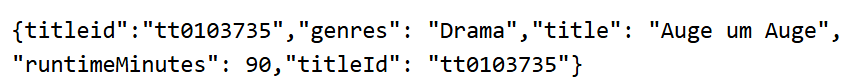
There are three basic steps to converting relational data into nonrelational data:

1. Create a flat file from the relational data.
2. Write a program to convert the flat file data into JSON format.
3. Import the JSON data into the nonrelational database of your choice.

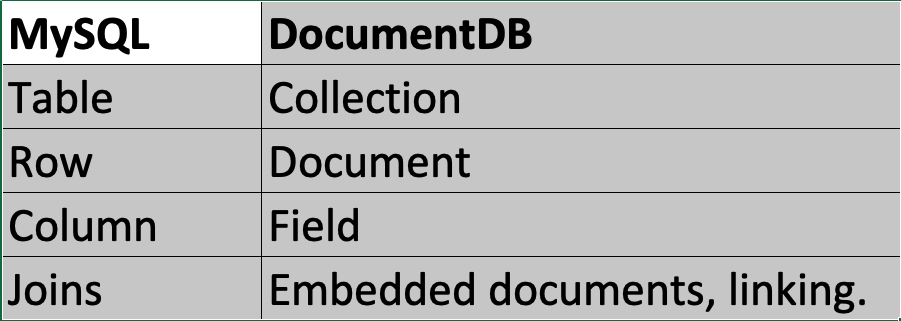
Example of 1st step (sql to one line data)



Example of 2nd



differences-



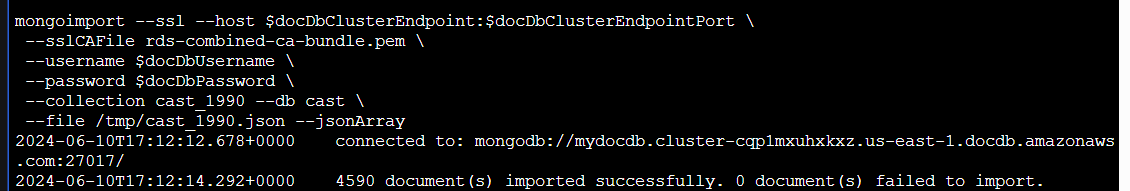
**Task 1: Import and validate data in Amazon DocumentDB**

Import flatfile (pre-created) into DB using two ways-single and batch

1.1 connect to CLI using URL given

1.2 **Import data using the mongoimport command**

This command imports data from the JSON file `/tmp/cast\_1990.json` into the `cast\_1990` collection in the `cast` database of a MongoDB cluster using SSL for a secure connection. The necessary credentials and parameters are sourced from the `set-creds.sh` script.



Running the command imports the cast\_1990.json file to Amazon DocumentDB as a collection of database cast.

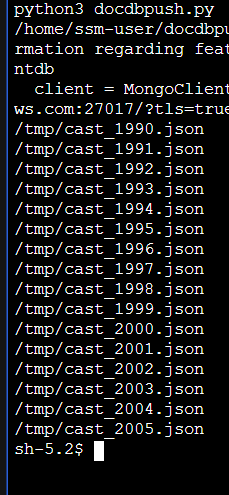
1.3 **Using python script**

Just run this command

cd ~

python3 docdbpush.py

The actual file is inbuilt into the CLI environment



**All files imported**

1.4 CONNECT TO THE AMAZON DOCUMENTDB SHELL

source ./set-creds.sh && \

mongosh --tls --host $docDbClusterEndpoint:$docDbClusterEndpointPort \

--tlsCAFile rds-combined-ca-bundle.pem \

--username $docDbUsername \

--password $docDbPassword

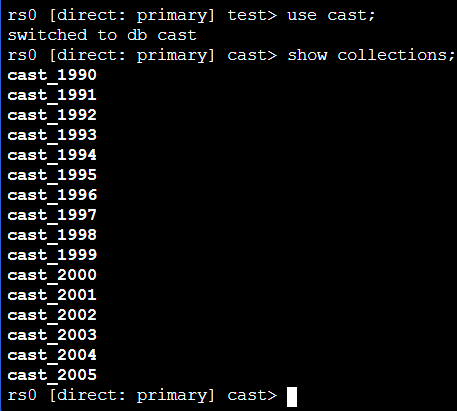
This command sources the set-creds.sh script to load necessary environment variables, then uses mongosh to securely connect to a MongoDB cluster using TLS, with the provided host, port, username, password, and CA file for SSL certificate verification.

1.5 using this



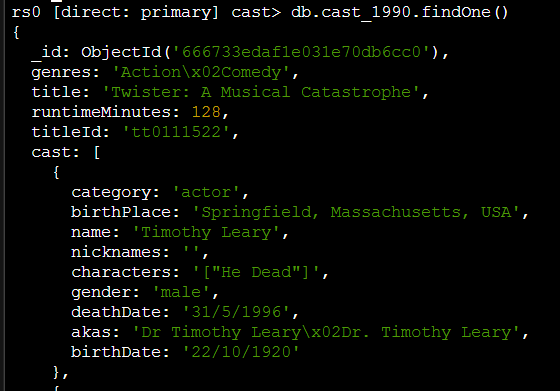
We can verify that data has been imported into cast DB

1.6 view collections inside DB



1.7 To view the first document in the cast\_1990 collection, run the following command:

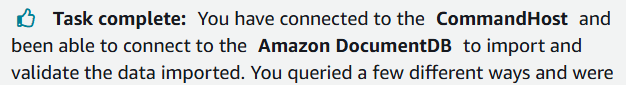
db.cast\_1990.findOne()



1.8 To count the number of documents in the cast\_1990 collection where the cast name is Sylvester Stallone, run the following command:

db.cast\_1990.find({"cast.name":"Sylvester Stallone"}).count()

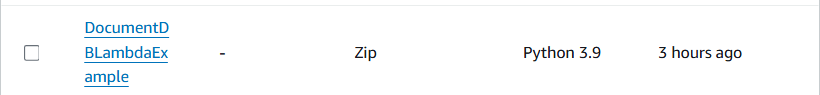




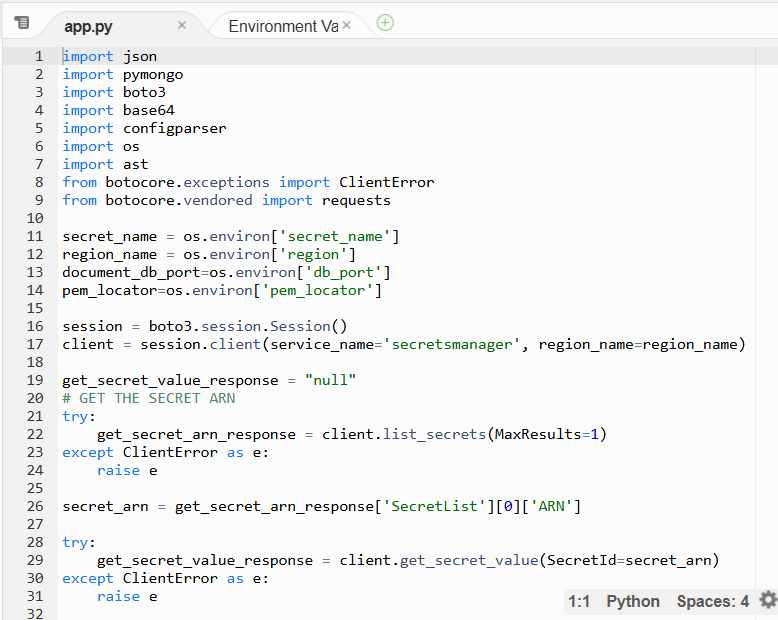
**Task 2: Use API Gateway and Lambda to read data from Amazon DocumentDB**

In this task, you configure a Lambda function to use Secrets Manager and API Gateway to read data from Amazon DocumentDB without needing a provisioned web server.

2.1 open lambda in console, and select this function (pre-created || on the main screen itself)



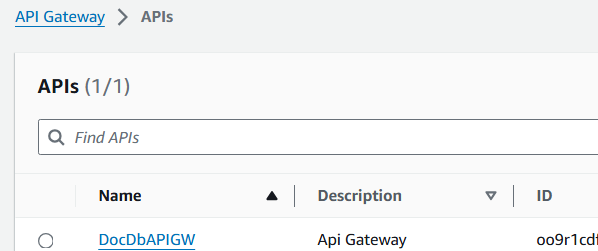
2.2 this is the code of the func



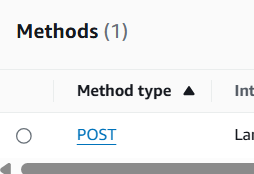
This function fetches credentials from Secrets Manager and uses those credentials to access Amazon DocumentDB.

The function expects an actor’s name to be supplied in the API POST request. The function reads this and uses it to count the number of times the actor is a member of a cast between 1990 and 2005.

2.3 Open API gateway in console and select this



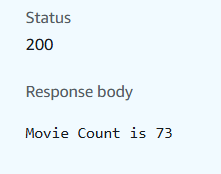
2.4 open POST method



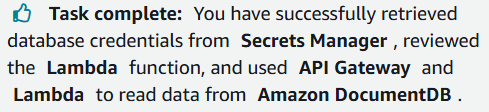
2.5 in the test section; enter the req. Body, and run



2.6 o/p-



This shows the working of lambda and api gateway together



**Task 3: Monitor logs and server metrics**

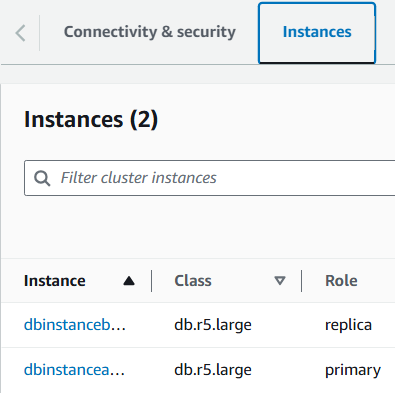
**USE THE AWS MANAGEMENT CONSOLE TO MONITOR**

3.1 open docDB in console and open clusters section

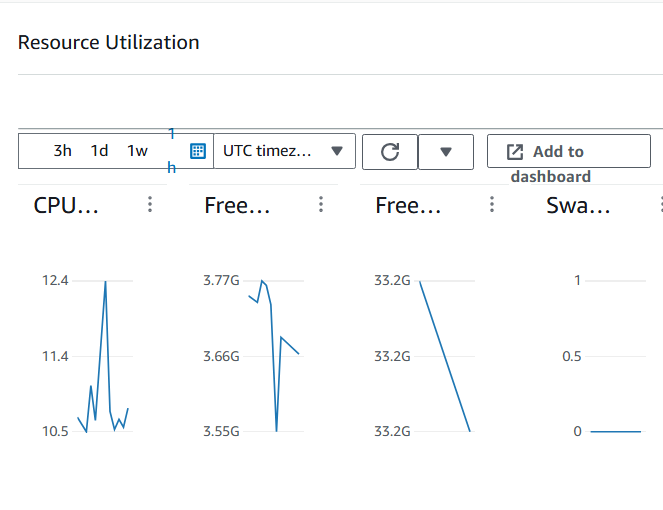
3.2 view the health

|  |  |
| --- | --- |

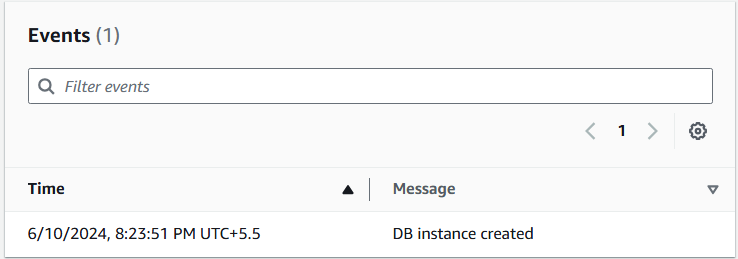
3.3 open the main cluster to view it’s instances



3.4 open the primary instance and view it’s resources



3.5 Choose the Events & tags tab, here you can find all recent instance-related events.



**USE THE AWS CLI TO CHECK CLUSTER AND INSTANCE STATUS**

3.6 use URL to open CLI

3.7 use this to check status

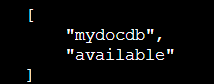
cd ~

aws docdb describe-db-clusters \

--db-cluster-identifier CLUSTER\_IDENTIFIER \

--query 'DBClusters[\*].[DBClusterIdentifier,Status]'

Replace CLUSTER\_IDENTIFIER with cluster name

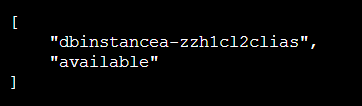


3.8 similarly for instance

aws docdb describe-db-instances \

--db-instance-identifier INSTANCE\_IDENTIFIER \

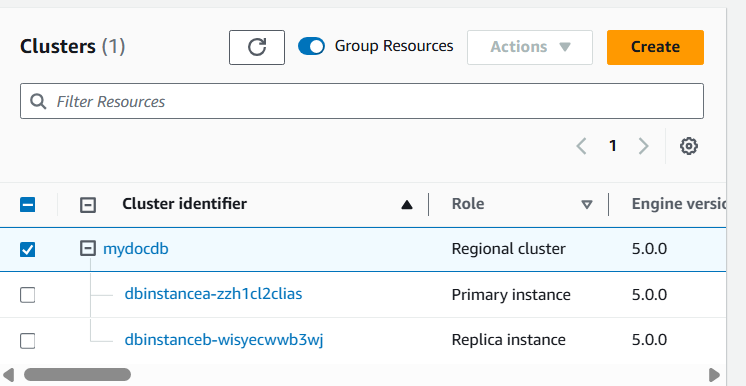
--query 'DBInstances[\*].[DBInstanceIdentifier,DBInstanceStatus]'



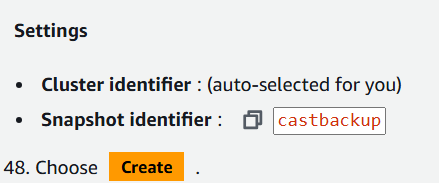
**Task 4: Creating a manual cluster snapshot** (creating backup)

Amazon DocumentDB continuously backs up your data to Amazon Simple Storage Service (Amazon S3) for up to 35 days. Amazon DocumentDB takes automatic snapshots of your data as part of this continuous backup process.

4.1 select your cluster (in docDB window) and choose actions



4.2 in the actions menu, choose **Take snapshot** and select as follows;



4.3 