

**WELCOME TO
PROJECT PERSENTATION**

**Automate Database
Backup and Restoration on
AWS**



Employ your cloud computing skills in an AWS environment to automate the process of backing up a database to S3 and restoring it when needed.

Cloud computing has become more important than ever with the ever growing usage of the internet and the many applications being launched for different purposes.

Purpose/Problem

With all the applications requiring a lot of data to be backed up on the cloud, along with the security and speed of access, automating this process is extremely important.

Connected RDS instance to MySQL workbench, and Deploy database to AWS S3 Bucket.

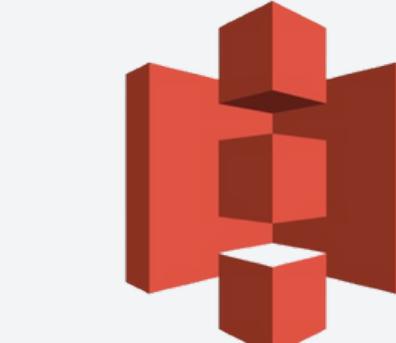
Wrote Optimized python script and Dockerfile , to push container image to AWS ECR .

Launched Lambda function to execute code and establish AWS Cloudwatch rule to schedule background task to automate process

Technology:



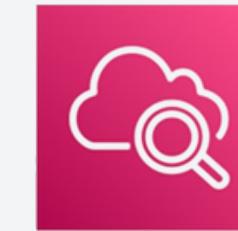
Amazon
RDS



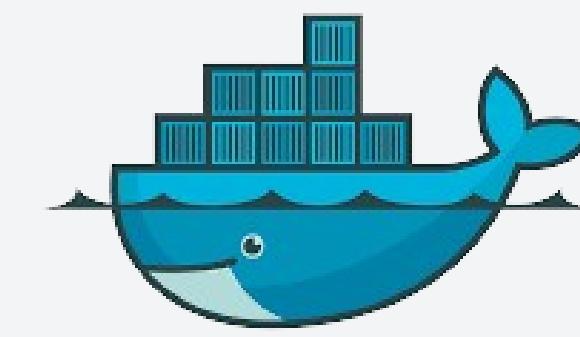
Amazon S3



python™



Amazon CloudWatch



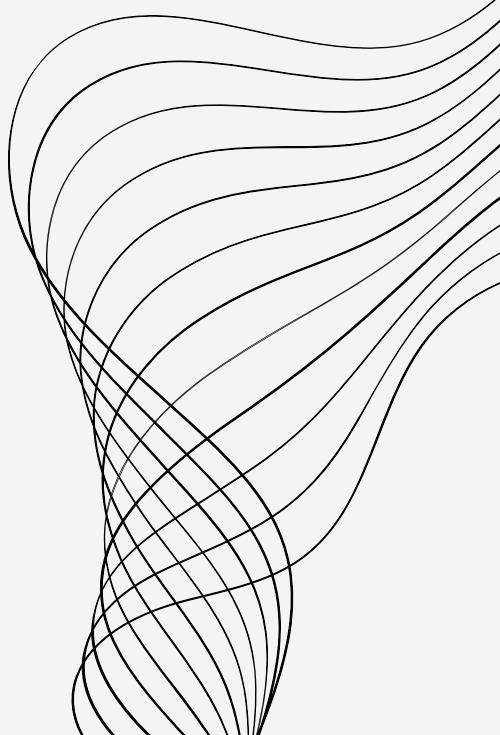
docker



Amazon Elastic
Container Registry



AWS Lambda



Create an RDS instance on AWS

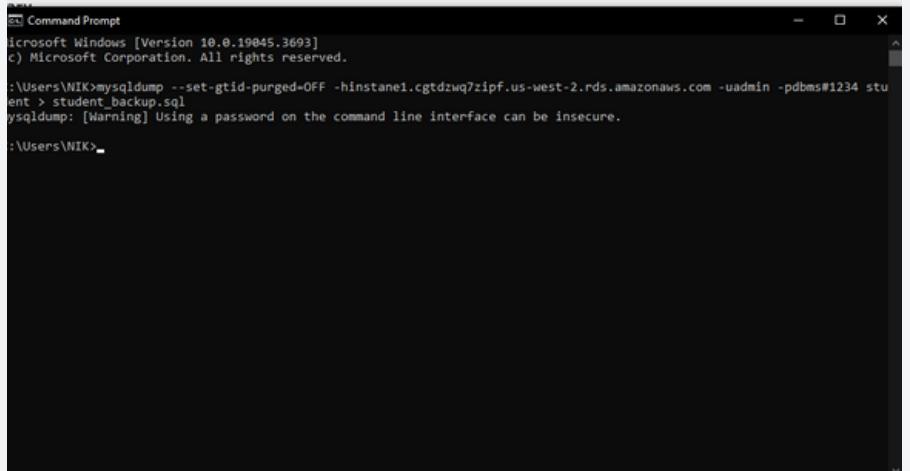
The screenshot shows the Amazon RDS Dashboard. On the left, there's a sidebar with options like Dashboard, Databases, Query Editor, and Performance Insights. The main area is titled 'Databases (1)' and shows a table with one row for 'instance1'. The details for 'instance1' are: Status: Available, Engine: MySQL Community, Region & AZ: us-west-2c, Size: db.t3.micro, CPU: 3 Actions, Current activity: 3.87%, and 0 Connections.

This screenshot shows the AWS EC2 Security Groups page. It lists a single security group named 'default' with a VPC ID of 'vpc-08d0d170051f71cd'. Under the 'Inbound rules' tab, there is one rule: 'sgr-03416fabea7525136' which allows all traffic from the security group 'sg-Oeb9085b8bf62b73'. The rule is defined by the source security group 'sg-Oeb9085b8bf62b73' and the destination port range '0.0.0.0/0'.

A screenshot of the MySQL Workbench 'Setup New Connection' dialog. The connection is named 'instance' and uses the 'Standard (TCP)' method. The host is 'instance.cgtdzwq7zpf.us-west-2.rds.amazonaws.com' with port 3306. SSL is enabled with 'TLS_AES_256_GCM_SHA384'. The user is 'admin' and the password is 'Store in Vault'. The default schema is left empty. A message box says 'Successfully made the MySQL connection'.

Security Groups: Allow inbound traffic on MySQL's port (usually 3306) from trusted IP addresses or CIDR blocks.

Using tools like MySQL Workbench, DBeaver, or the mysql command-line client, connect to your RDS instance using the endpoint provided in the RDS dashboard and the master username/password you set up earlier.



creating the mysqldump for backup file student sql database > name of back file name

This screenshot shows MySQL Workbench. On the left, the 'Schemas' tree shows 'student'. In the center, the 'Query Editor' tab has a dump of the 'student' database. Below it, the 'Object Info' tab shows the structure of the 'class' table, which has columns 'id' (int, primary key) and 'name' (varchar(10)). The 'SQL Additions' panel shows various MySQL configuration statements.

backup file

This screenshot shows MySQL Workbench with two tabs: 'Query 1' and 'Query 2'. 'Query 1' contains a script to create a 'student' database and tables 'class' and 'values', and insert data into them. 'Query 2' shows the results of the 'class' table, which has three rows: 'jane', 'peter', and 'david'. The 'Output' tab at the bottom shows the execution log with various MySQL commands and their results.

Create an S3 Bucket

The screenshot shows the AWS S3 console with a green header bar indicating 'Upload succeeded'. Below it, a summary table shows one file uploaded successfully to 's3://toko360'. The main area displays a table of files and folders, with 'student_backup.sql' listed as 1.9 KB and 'Succeeded' status. The bottom navigation bar includes 'CloudShell' and 'Feedback'.

- Bucket name: toko360
- Upload the student_back.sql file in bucket

The screenshot shows the AWS S3 Lifecycle configuration page for bucket 'toko360'. A green header bar indicates 'The lifecycle configuration was updated. Lifecycle rule "backups3" was successfully added.' The main area shows a table of lifecycle rules, with one entry named 'backups3' which transitions objects from Standard to One Zone-IA, then to Glacier Flexible Retrieval, and finally to Glacier Deep Archive. The bottom navigation bar includes 'CloudShell' and 'Feedback'.

Adding Lifecycle Policies

- S3 Standard for 30 day
- S3 One Zone-Infrequent Access (S3 One Zone-IA) for 90 day
- S3 Glacier Instant Retrieval for 270 day
- Amazon S3 Glacier Deep Archive for 365 day

Python Script

Write the Backup Script

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer (Left):** Shows a project structure under "PROJECT" containing files like .vscode, launch.json, Dockerfile, foo.csv, lambda_function.py, main.py, mysql.exe, mysqldump.exe, readme.txt, and rest.txt.
- Code Editor (Top Center):** Displays a Python script named "lambda_function.py" located at C:\...\Documents\lambda_function.py\lambda_handler. The code performs a MySQL backup to S3 and a restore from S3 back to the database.
- Terminal (Bottom):** Shows the output of running the script, including MySQL dump and restore logs, and a message indicating the screenshot was saved to OneDrive.

```
cursor.execute("show tables")
sql = "insert into tbl_Student values (1, 'NIKHIL', 'Aurangabad')"
cursor.execute(sql)
cursor.execute("select * from tbl_Student")
connection.commit()

print(cursor.fetchall())

command = f"mysqldump --host %s --user %s -p%s %s | aws s3 cp - s3://%s/%s" %(host, user, password, DB_NAME, S3_Bucket_Name, DB_NAME + '.sql')
subprocess.run(command, shell=True)
print("MySQL Database Backup Done ===>")

if not os.path.exists(f'/tmp/{DOWNLOAD_FILE}'):
    os.makedirs('/tmp')
    open(f'/tmp/{DOWNLOAD_FILE}', 'w').close()

S3_Client = boto3.client('s3')

S3_Client.download_file(S3_Bucket_Name, DB_NAME + '.sql', f'/tmp/{DOWNLOAD_FILE}')

cursor.execute("create database IF NOT EXISTS db_Student_backup")

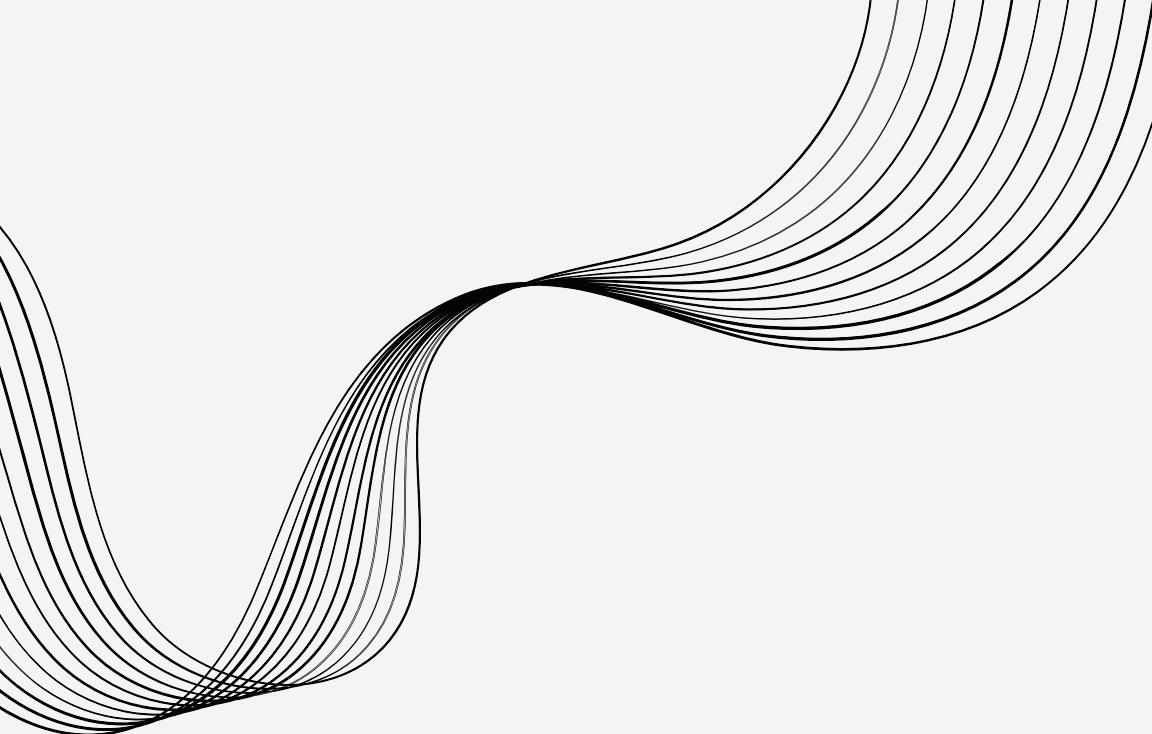
command = f"mysql -h {host}, -u {user}, -p{password} db_Student_backup < /tmp/{DOWNLOAD_FILE}"
output = subprocess.run(command, shell=True)

if output.returncode == 0:
    print("MySQL Database restore successfully ===>")
else:
    (f"MySQL Database restore failed with error code {output.returncode}")
return "MySQL Database Backup Done ===>"
lambda_handler(None,None)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

((1, 'NIKHIL', 'Aurangabad'), (1, 'NIKHIL', 'Aurangabad'), (1, 'NIKHIL', 'Aurangabad'))
mysqldump: [Warning] Using a password on the command line interface can be insecure.
Warning: A partial dump from a server that has GTIDs will by default include the GTIDs of all transactions, even those that changed suppressed parts of the database. If you don't want to restore GTIDs, pass --set-gtid-purged=OFF. To make a complete dump, pass --all-databases --triggers --routines --events.
Warning: A dump from a server that has GTIDs enabled will by default include the GTIDs of all transactions, even those that were executed during the dump. This might result in an inconsistent data dump.
In order to ensure a consistent backup of the database, pass --single-transaction or --lock-all-tables or --master-data.
MySQL Database Backup Done ===>
mysql: [Warning] Using a password on the command line interface can be insecure.
ERROR 2005 (HY000): Unknown MySQL server host 'database-1.cgtdzwq7zipf.us-west-2.rds.amazonaws.com,' (11001)
PS C:\Users\NIK\OneDrive\Desktop\project>

OneDrive
Screenshot saved
The screenshot was added to your OneDrive.



File Edit Selection View Go Run Terminal Help

project

EXPLORER

PROJECT

- .vscode
- launch.json
- Dockerfile
- foo.csv
- lambda_function.py
- main.py
- mysqlExe
- mysqldump.exe
- readme.txt
- rest.txt

main.py

```
1 import cursor
2 import boto3
3 import pymysql
4 import subprocess
5 import os
6
7 DB_NAME = 'db_Student'
8 BACKUP_DB_NAME = 'db_Student_backup'
9 TABLE = 'tbl_Student'
10 DOWNLOAD_FILE = 'download_backup.sql'
11
12 def lambda_handler(event, context):
13     print("MySQL Backup Function Started ===>")
14     #mysqldb.corlgu8ddbzz.us-west-2.rds.amazonaws.com
15
16     host = 'database-1.cgtdzwq7zipf.us-west-2.rds.amazonaws.com'
17     user = 'admin'
18     password = 'dbms#1234'
19     S3_Bucket_Name = 'toko360'
20
21     print("Trying to Connect With MySQL DataBase ===>")
22
23     connection = pymysql.connect(host=host, user=user, password=password)
24
25     print("MySQL Database Connected ===>")
26
27     cursor = connection.cursor()
28
29     sql = f"create database IF NOT EXISTS {DB_NAME}"
30     cursor.execute(sql)
31
32     cursor.execute("show databases")
33     cursor.fetchall()
34     connection.select_db(DB_NAME)
35
36     cursor.execute("CREATE TABLE IF NOT EXISTS tbl_Student (id int, name VARCHAR(255), address VARCHAR(255))")
37     cursor.execute("show tables")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

(1, 'NIKHIL', 'Aurangabad'), (1, 'NIKHIL', 'Aurangabad'), (1, 'NIKHIL', 'Aurangabad')

mysqldump: [Warning] Using a password on the command line interface can be insecure.

Warning: A partial dump from a server that has GTIDs will by default include the GTIDs of all transactions, even those that changed suppressed parts of the database. If you don't want TIDs, pass --set-gtid-purged=OFF. To make a complete dump, pass --all-databases --triggers --routines --events.

Warning: A dump from a server that has GTIDs enabled will by default include the GTIDs of all transactions, even those that were executed during its extraction and might not be represented in the dumped data. This might result in an inconsistent data dump.

In order to ensure a consistent backup of the database, pass --single-transaction or --lock-all-tables or --master-data.

MySQL Database Backup Done ===>

mysql: [Warning] Using a password on the command line interface can be insecure.

ERROR 2005 (HY000): Unknown MySQL server host 'database-1.cgtdzwq7zipf.us-west-2.rds.amazonaws.com', (11001)

PS C:\Users\NIK\OneDrive\Desktop\project> []

Ln 33, Col 1 Spaces:4 UTF-8 CRLF

TIMELINE lambda_function

- File Saved now
- File Saved 3 wks
- File Saved
- File Saved
- File Saved
- File Saved

File Edit Selection View Go Run Terminal Help

project

EXPLORER

PROJECT

- .vscode
- launch.json
- Dockerfile
- foo.csv
- lambda_function.py
- main.py
- mysqlExe
- mysqldump.exe
- readme.txt
- rest.txt

main.py

```
24
25     print("MySQL Database Connected ===>")
26
27     cursor = connection.cursor()
28
29     sql = f"create database IF NOT EXISTS {DB_NAME}"
30     cursor.execute(sql)
31
32     cursor.execute("show databases")
33     cursor.fetchall()
34     connection.select_db(DB_NAME)
35
36     cursor.execute("CREATE TABLE IF NOT EXISTS tbl_Student (id int, name VARCHAR(255), address VARCHAR(255))")
37     cursor.execute("show tables")
38
39     sql = "insert into tbl_Student values (1, 'NIKHIL', 'Aurangabad')"
40     cursor.execute(sql)
41
42     cursor.execute("select * from tbl_Student")
43     connection.commit()
44
45     print(cursor.fetchall())
46
47     command = f"mysqldump --host %s --user %s -p%s %s | aws s3 cp - s3://%s/%s" %(
48         host, user, password, DB_NAME, S3_Bucket_Name, DB_NAME + '.sql')
49     subprocess.run(command, shell=True)
50     print("MySQL Database Backup Done ===>")
51
52     if not os.path.exists(f'/tmp/{DOWNLOAD_FILE}'):
53         os.makedirs('/tmp')
54         open(f'/tmp/{DOWNLOAD_FILE}', 'w').close()
55
56     S3_Client = boto3.client('s3')
57
58     S3_Client.download_file(S3_Bucket_Name, DB_NAME + '.sql', f'/tmp/{DOWNLOAD_FILE}')
59
60     cursor.execute("create database IF NOT EXISTS db_Student backup")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

(1, 'NIKHIL', 'Aurangabad'), (1, 'NIKHIL', 'Aurangabad'), (1, 'NIKHIL', 'Aurangabad')

mysqldump: [Warning] Using a password on the command line interface can be insecure.

Warning: A partial dump from a server that has GTIDs will by default include the GTIDs of all transactions, even those that changed suppressed parts of the database. If you don't want TIDs, pass --set-gtid-purged=OFF. To make a complete dump, pass --all-databases --triggers --routines --events.

Warning: A dump from a server that has GTIDs enabled will by default include the GTIDs of all transactions, even those that were executed during its extraction and might not be represented in the dumped data. This might result in an inconsistent data dump.

In order to ensure a consistent backup of the database, pass --single-transaction or --lock-all-tables or --master-data.

MySQL Database Backup Done ===>

mysql: [Warning] Using a password on the command line interface can be insecure.

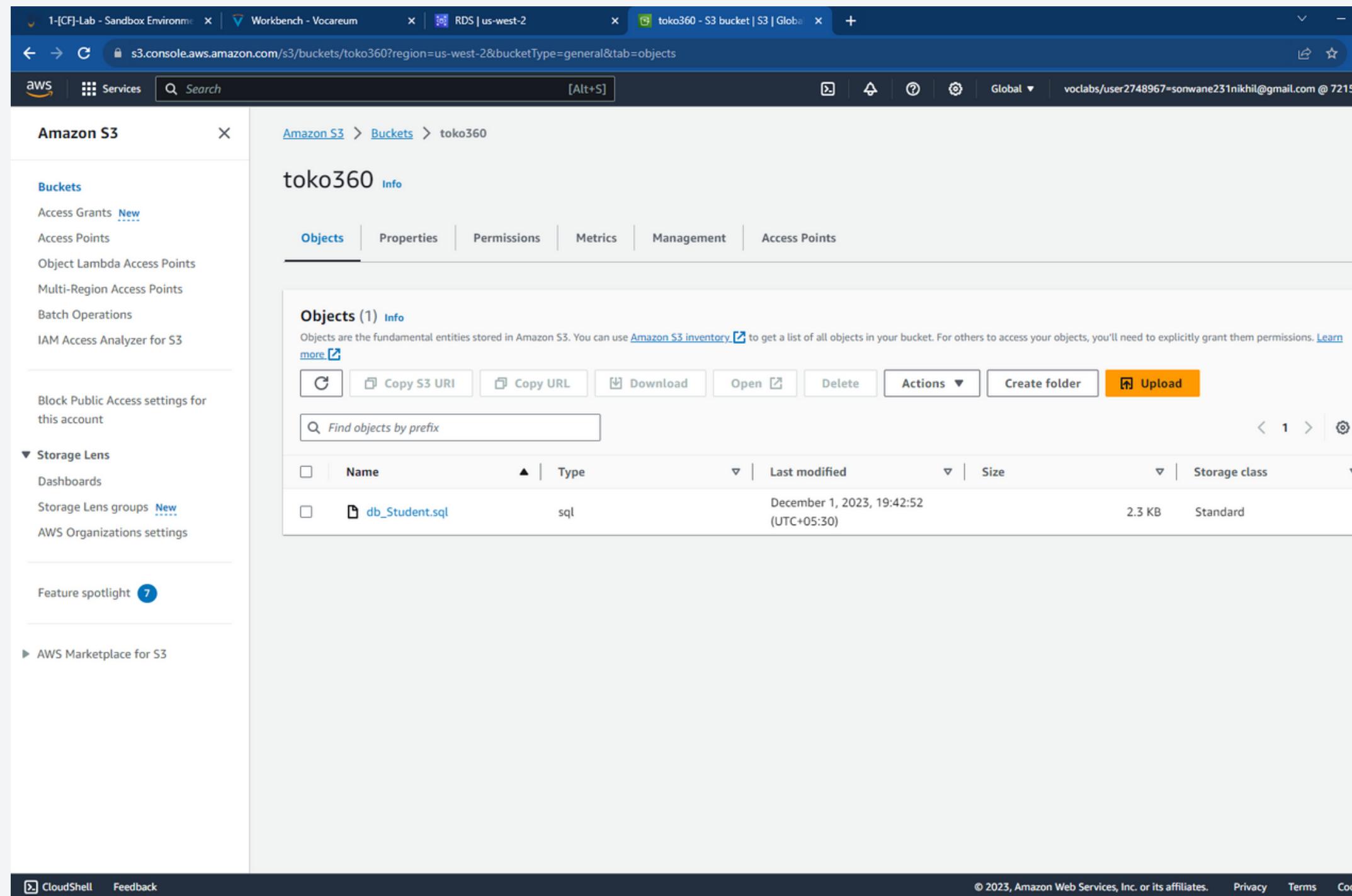
ERROR 2005 (HY000): Unknown MySQL server host 'database-1.cgtdzwq7zipf.us-west-2.rds.amazonaws.com', (11001)

PS C:\Users\NIK\OneDrive\Desktop\project> []

TIMELINE lambda_function

- File Saved now
- File Saved 3 wks
- File Saved
- File Saved
- File Saved
- File Saved

Write a python script for backup in s3:

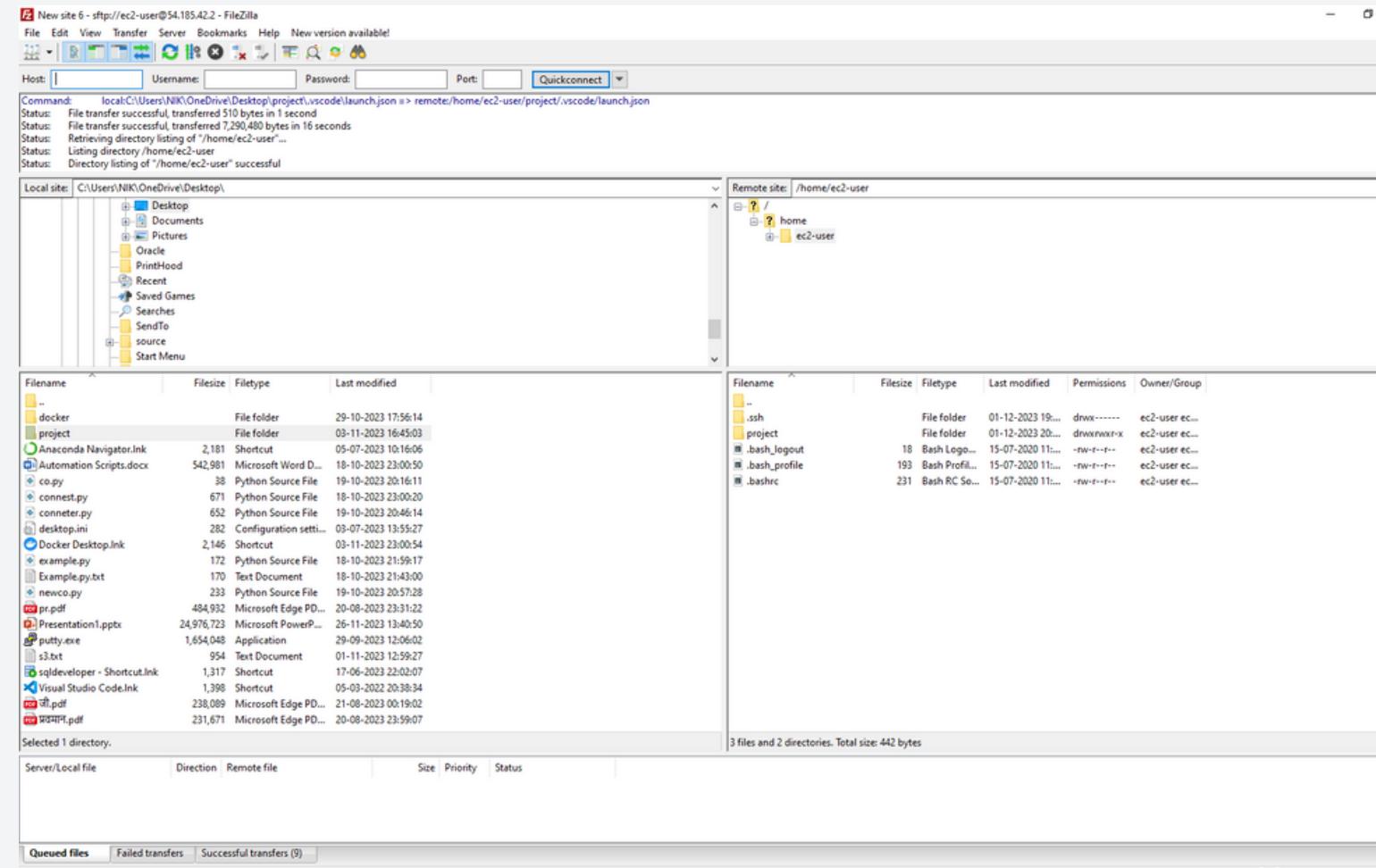


The screenshot shows the AWS S3 console interface. The left sidebar lists various services like Amazon S3, RDS, and Workbench. The main content area shows the 'toko360' bucket. The 'Objects' tab is selected, displaying one object: 'db_Student.sql'. The object details show it is a SQL file (sql) uploaded on December 1, 2023, at 19:42:52 (UTC+05:30), with a size of 2.3 KB and a storage class of Standard.

Name	Type	Last modified	Size	Storage class
db_Student.sql	sql	December 1, 2023, 19:42:52 (UTC+05:30)	2.3 KB	Standard

db_Student.sql
bucket name is
toko360

Package and Upload Your Script to Lambda



Filezilla using to file upload to cloud

Dockerfile

The screenshot shows a terminal window in a code editor (VS Code) with a Dockerfile open. The Dockerfile contains commands to copy files from the host to the container, set environment variables, and run a command. The terminal output shows the Docker build process, including the creation of a build context and the resulting Docker image. The final command shown is "docker push 721507206823.dkr.ecr.us-west-2.amazonaws.com/docker-lambda:latest", which pushes the image to the specified AWS Lambda repository.

```
main.py
lambda_function.py C:\Users\NIK\OneDrive\Documents> Dockerfile
lambda_function.py
readme.txt
rest.txt

EXPLORER PROJECT
vscode
launch.json
Dockerfile
foo.csv
lambda_function.py
main.py
mysql.exe
mysqldump.exe
readme.txt
rest.txt

C: > Users > NIK > OneDrive > Documents > lambda_function.py > lambda_handler
37 cursor.execute('show tables')
38
39 sql = "insert into tbl_Student values (1, 'NIKHIL', 'Aurangabad')"
cursor.execute(sql)
40
41 cursor.execute("select * from tbl_Student")
connection.commit()
42
43
44
45 print(cursor.fetchall())
46
47 command = f"mysqldump --host %s --user %s -p%{s} %s | aws s3 cp - s3://%s/%s" %(
| host, user, password, DB_NAME, S3_Bucket_Name, DB_NAME + '.sql')
48 subprocess.run(command, shell=True)
49 print("MySQL Database Backup Done =====")
50
51 if not os.path.exists(f'/tmp/{DOWNLOAD_FILE}'):
52     os.makedirs('/tmp')
53     open(f'/tmp/{DOWNLOAD_FILE}', 'w').close()
54
55 S3_client = boto3.client('s3')
56

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
=> => transferring context: 28
Dockerfile:3
-----
1 | ARG FUNCTION_DIR="/home/app"
2 |
3 | >>> MAINTAINER Nikhil
4 |
5 | FROM python:slim-buster
-----
ERROR: failed to solve: no build stage in current context
PS C:\Users\NIK\OneDrive\Desktop\project> docker push 721507206823.dkr.ecr.us-west-2.amazonaws.com/docker-lambda:latest
The push refers to repository [721507206823.dkr.ecr.us-west-2.amazonaws.com/docker-lambda]
5f70bf18a086: Pushed
35a0dfe93a97: Pushed
63553a5dcdef: Pushed
5c7d871cf19f: Pushed
0966a1e695e7: Pushed
e17640af2cd6: Pushed
87f39d58eefa: Pushed
195eb5416ef8: Pushed
7936e682fe33: Pushed
5b438b538e1e: Pushed
ae2d5769c5e: Pushed
e2ef8a51359d: Pushed
latest: digest: sha256:cfff1c8c0acf2dd49edf32091f15874ddf2f7d8b56eaef301474babdb86464a39 size: 2839
PS C:\Users\NIK\OneDrive\Desktop\project>
```

Docker name: docker_lambda

The screenshot shows the AWS ECR console with the URL <https://us-west-2.console.aws.amazon.com/ecr/repositories/private/721507206823/docker-lambda?region=us-west-2>. The left sidebar shows navigation options like Private registry, Public registry, and Repositories. The main area displays the 'docker-lambda' repository with one image listed:

Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status	Vulnerabilities
latest	Image	December 01, 2023, 20:27:15 (UTC+05:5)	214.33	Copy URI	sha256:cff1c8c0acf2dd49...	-	-

The screenshot shows the AWS ECR console with the URL <https://us-west-2.console.aws.amazon.com/ecr/repositories?region=us-west-2>. A green success message at the top says 'Successfully created repository docker-lambda'. The main area shows the 'Private repositories (1)' table:

Repository name	URI	Created at	Tag immutability	Scan frequency	Encryption type	Pull through cache
docker-lambda	721507206823.dkr.ecr.us-west-2.amazonaws.com/docker-lambda	December 01, 2023, 19:45:42 (UTC+05:5)	Disabled	Manual	AES-256	Inactive

deploy docker image

upload

Function overview [Info](#)

EventBridge (CloudWatch Events) (2)

[+ Add trigger](#)[+ Add destination](#)

Description

Last modified
14 minutes ago

Function ARN

arn:aws:lambda:us-west-2:721507206823:function:upload

Function URL [Info](#)

Image

Test

Monitor

Configuration

Aliases

Versions

Metrics

Logs

Traces

[View CloudWatch logs](#)[View X-Ray traces](#)CloudWatch Logs [Info](#)

Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables list the most recent and most expensive function invocations across all function activity. To view logs for a specific function version or alias, visit the **Monitor** section at that level.

1h 3h 12h 1d 3d 1w Custom UTC timezone C ▾

Recent invocations

#	Timestamp	RequestID	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB	MemoryUsedInMB
► 1	2023-11-04T12:44:58.061Z	9f7d3c45-ae66-49ed-8b4d-c235a87a5207	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6940.32	6941.0	256	66
► 2	2023-11-04T12:44:30.443Z	h8facb18-521e-423c-943e-6087bd56cc71	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6964.24	6965.0	256	66

Cloudwatch and output

aws Services Search [Alt+S] Oregon voclabs/user2748967+sonwane231nkhil@gmail.com @ 7215-0720-6823

CloudWatch Logs [Info](#)

Lambda logs all requests handled by your function and automatically stores logs generated by your code through Amazon CloudWatch Logs. To validate your code, instrument it with custom logging statements. The following tables list the most recent and most expensive function invocations across all function activity. To view logs for a specific function version or alias, visit the **Monitor** section at that level.

1h 3h 12h 1d 3d 1w Custom UTC timezone C ▾

Recent invocations

#	Timestamp	RequestID	LogStream	DurationInMS	BilledDurationInMS	MemorySetInMB	MemoryUsedInMB
► 1	2023-11-04T12:44:58.061Z	9f7d3c45-ae66-49ed-8b4d-c235a87a5207	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6940.32	6941.0	256	66
► 2	2023-11-04T12:44:30.443Z	b8facb18-521e-423c-943e-6087bd56cc71	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6964.24	6965.0	256	66
► 3	2023-11-04T12:43:10.007Z	66668eec-f208-468e-bb04-48c7c6b0b09	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6595.39	6596.0	256	66
► 4	2023-11-04T12:42:59.965Z	2a5ce4ed-5307-4824-a806-01d9d24aebce	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6709.93	6710.0	256	66
► 5	2023-11-04T12:42:58.001Z	71698470-3ef4-4998-907a-c7803c3690da	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6979.90	6980.0	256	66
► 6	2023-11-04T12:42:09.761Z	b8facb18-521e-423c-943e-6087bd56cc71	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6990.02	6991.0	256	66
► 7	2023-11-04T12:41:57.702Z	66668eec-f208-468e-bb04-48c7c6b0b09	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6906.39	6907.0	256	66
► 8	2023-11-04T12:41:06.560Z	2a5ce4ed-5307-4824-a806-01d9d24aebce	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6923.66	6924.0	256	66
► 9	2023-11-04T12:40:58.061Z	b8facb18-521e-423c-943e-6087bd56cc71	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	7103.67	7104.0	256	66

Most expensive invocations in GB-seconds (memory assigned * billed duration)

#	Timestamp	RequestID	LogStream	BilledDurationInMS	MemorySetInMB	BilledDurationInGBSeconds
► 1	2023-11-04T12:39:59.901Z	2a5ce4ed-5307-4824-a806-01d9d24aebce	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	7114.0	256	1.7785
► 2	2023-11-04T12:40:58.061Z	b8facb18-521e-423c-943e-6087bd56cc71	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	7104.0	256	1.776
► 3	2023-11-04T12:42:09.761Z	b8facb18-521e-423c-943e-6087bd56cc71	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6991.0	256	1.7477
► 4	2023-11-04T12:42:08.001Z	71698470-3ef4-4998-907a-c7803c3690da	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6980.0	256	1.745
► 5	2023-11-04T12:44:30.443Z	b8facb18-521e-423c-943e-6087bd56cc71	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6965.0	256	1.7412
► 6	2023-11-04T12:44:58.061Z	9f7d3c45-ae66-49ed-8b4d-c235a87a5207	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6941.0	256	1.7352
► 7	2023-11-04T12:41:06.560Z	2a5ce4ed-5307-4824-a806-01d9d24aebce	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6924.0	256	1.731
► 8	2023-11-04T12:41:57.702Z	66668eec-f208-468e-bb04-48c7c6b0b09	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	6907.0	256	1.7268
► 9	2023-11-04T12:40:58.061Z	b66e718d-c068-48a8-bd94-c411abef10d6	2023/11/04/[\$LATEST]ce201e9a013547ea8002503936039471	13466.0	128	1.6832

Python Script : lambda_function.py

```
import cursor
import boto3
import pymysql
import subprocess
import os

DB_NAME = 'db_Student'
BACKUP_DB_NAME ='db_Student_backup'
TABLE = 'tbl_Student'
DOWNLOAD_FILE = 'download_backup.sql'

def lambda_handler(event, context):
    print("MySQL Backup Function Started ==>")
    #mysqlDb.corlgu8ddbzz.us-west-2.rds.amazonaws.com

    host = 'database-1.cgtdzwq7zipf.us-west-2.rds.amazonaws.com'
```

```
user = 'admin'  
password = 'dbms#1234'  
  
S3_Bucket_Name = 'toko2r6'  
  
print("Trying to Connect With MySQL DataBase ===>")  
connection = pymysql.connect(host=host, user=user,  
                             password=password)  
  
print("MySQL Database Connected ===>")  
cursor = connection.cursor()  
sql = f"create database IF NOT EXISTS {DB_NAME}"  
cursor.execute(sql)  
  
cursor.execute("show databases")  
cursor.fetchall()  
connection.select_db(DB_NAME)  
  
cursor.execute("CREATE TABLE IF NOT EXISTS  
tbl_Student (id int, name VARCHAR(255), address  
             VARCHAR(255))")  
  
cursor.execute("show tables")
```

```
sql = "insert into tbl_Student values (1, 'nikhil',  
                                         'Aurangabad')"  
  
cursor.execute(sql)  
  
cursor.execute("select * from tbl_Student")  
connection.commit()  
  
print(cursor.fetchall())  
  
command = f"mysqldump --host %s --user %s -p%s  
           %s | aws s3 cp - s3://%s/%s" %(  
           host, user, password, DB_NAME, S3_Bucket_Name,  
           DB_NAME + '.sql')  
  
subprocess.run(command, shell=True)  
  
print("MySQL Database Backup Done ===>")  
  
if not os.path.exists(f'/tmp/{DOWNLOAD_FILE}'):  
    os.makedirs('/tmp')  
    open (f'/tmp/{DOWNLOAD_FILE}', 'w').close()  
  
S3_Client = boto3.client('s3')
```

```
S3_Client.download_file(S3_Bucket_Name, DB_NAME +  
    '.sql', f'/tmp/{DOWNLOAD_FILE}')  
  
cursor.execute("create database IF NOT EXISTS  
    db_Student_backup")  
  
command = f"mysql -h {host}, -u {user}, -p{password}  
    db_Student_backup < /tmp/{DOWNLOAD_FILE}"  
  
output = subprocess.run(command, shell=True)  
  
if output.returncode == 0:  
    print("MySQL Database restore successfully ===>")  
else:  
    (f"MySQL Database restore failed with error code  
        {output.returncode}")  
  
return "MySQL Database Backup Done ===>"  
  
lambda_handler(None,None)
```

Dockerfile:

```
ARG FUNCTION_DIR="/home/app"  
  
#MAINTAINER Nikhil  
  
FROM python:slim-buster  
  
ARG FUNCTION_DIR  
RUN mkdir -p ${FUNCTION_DIR}  
COPY *.py ${FUNCTION_DIR}  
  
RUN apt-get update && \  
    apt-get install -y \  
    g++ \  
    make \  
    cmake \  
    unzip \  
    libcurl4-openssl-dev
```

```
RUN python3 -m pip install awslambdaric boto3 pymysql cursor
```

```
RUN python3 -m pip install \  
awscli
```

```
RUN apt-get update && apt-get install -y mariadb-client && \  
apt-get clean autoclean && \  
apt-get autoremove --yes &&\ \  
rm -rf /var/lib/{apt,dpkg,cache,log}/
```

```
WORKDIR ${FUNCTION_DIR}
```

```
ENTRYPOINT [ "/usr/local/bin/python3", "-m", "awslambdaric" ]
```

```
CMD [ "lambda_function.lambda_handler" ]
```