WELCOME TO

PROJECT PERSENTATION

Automate Database Backup and Restoration on AWS

Purpose/Problem

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.

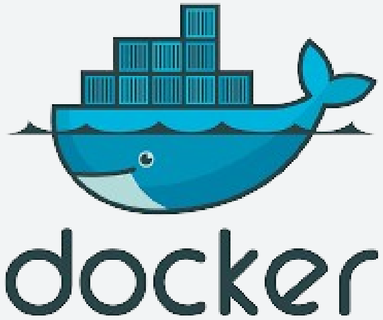
Cloud computing has become more important than ever with the ever growing usage of the internet and the many applications being launched for different purposes.
With all the applications requiring a lot of data to be backed up on the cloud, along with the security and speed of access, automating this process is extremely important.

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

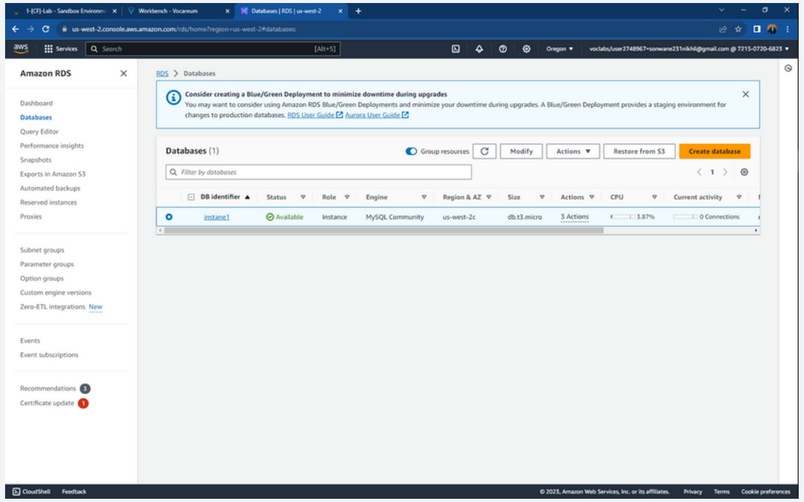
Wrote Optimized python script and Dockerfile , to push container imageto AWS ECR .

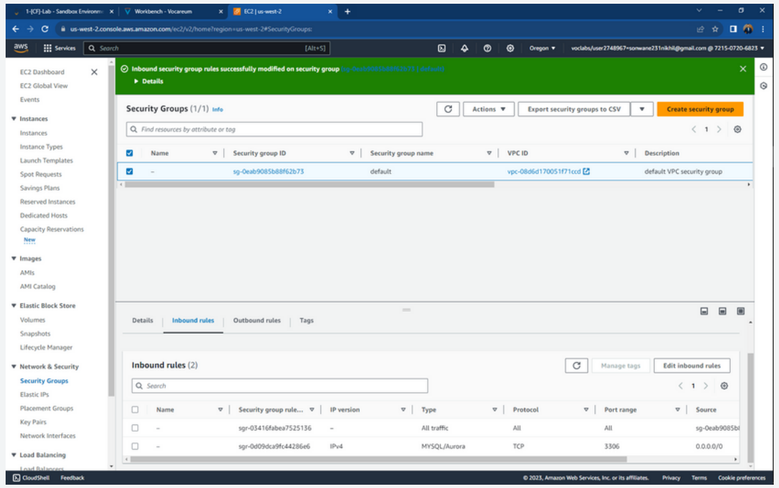
Launched Lambda function to execute code and establish AWS Cloudwatch rule to schedule background task to automate process

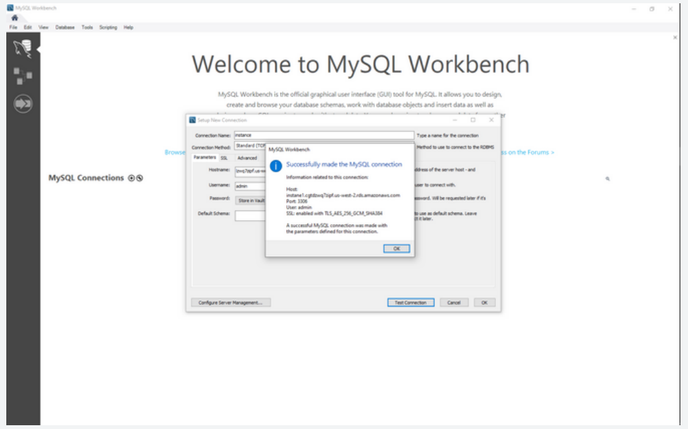
Technology:



Create an RDS instance on AWS

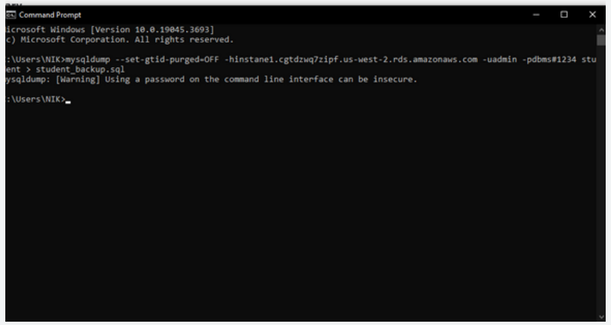




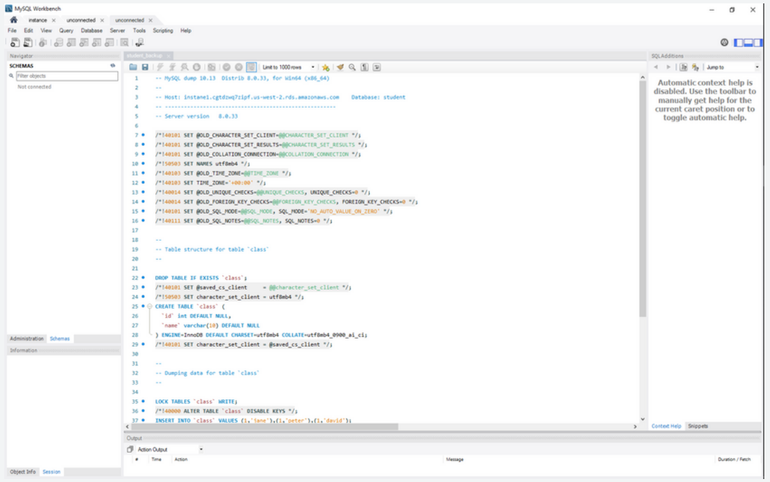


Security Groups: Allow inbound traffic on MySQL's port (usually 3306) from trusted IP addresses or CIDR blocks.

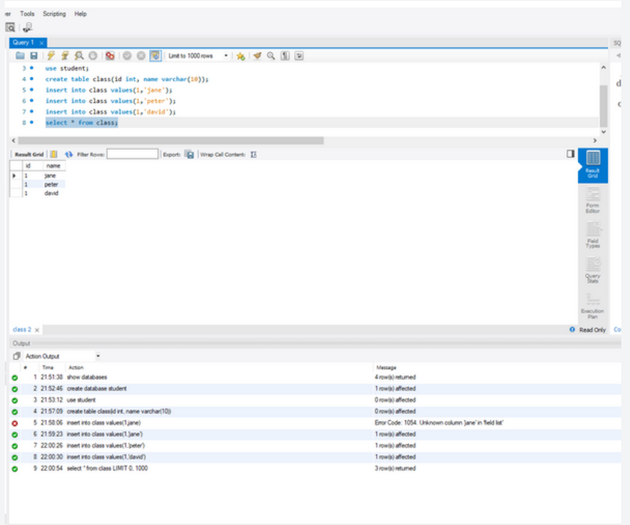
Using tools like MySQL Workbench, DBeaver, or the mysql command-line client, connect to your RDS instance using the endpoint provided in the RDS dashboard and the master username/password you set up earlier.



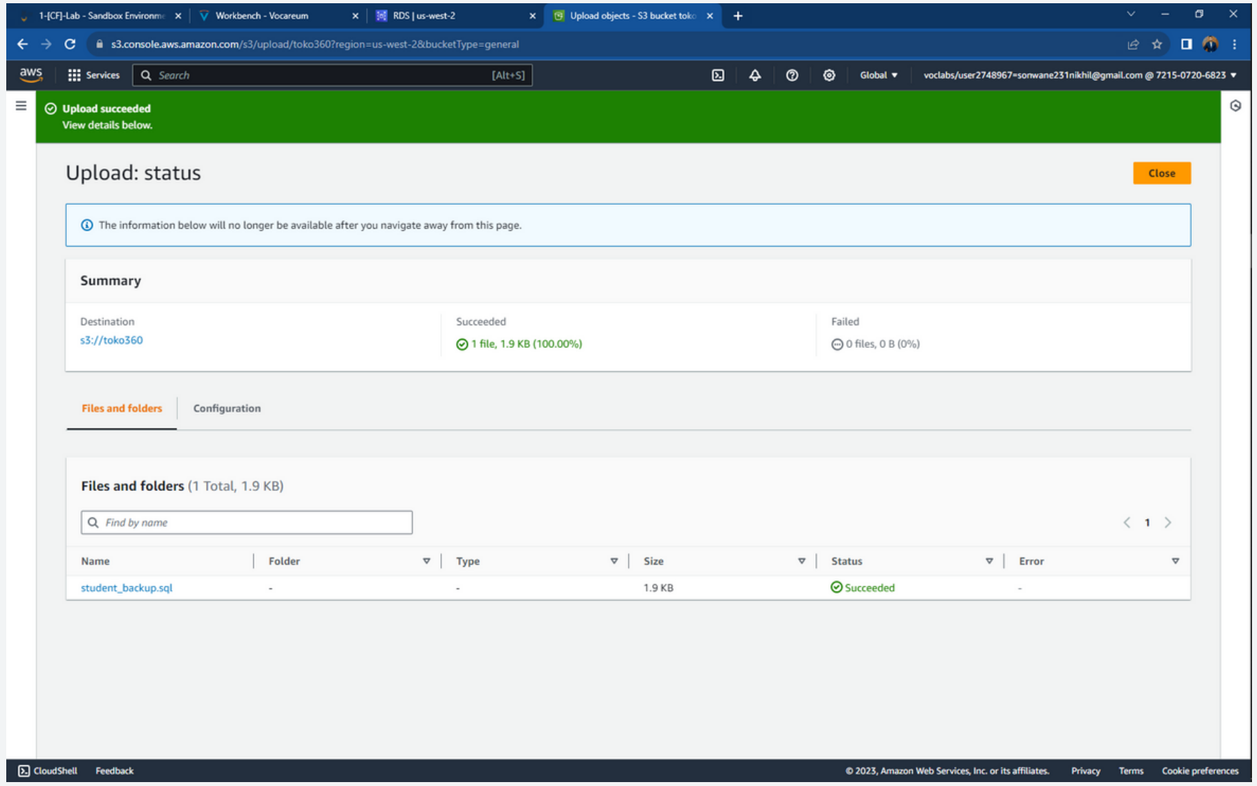
creating the mysqldump for backup file student sql database > name of back file name



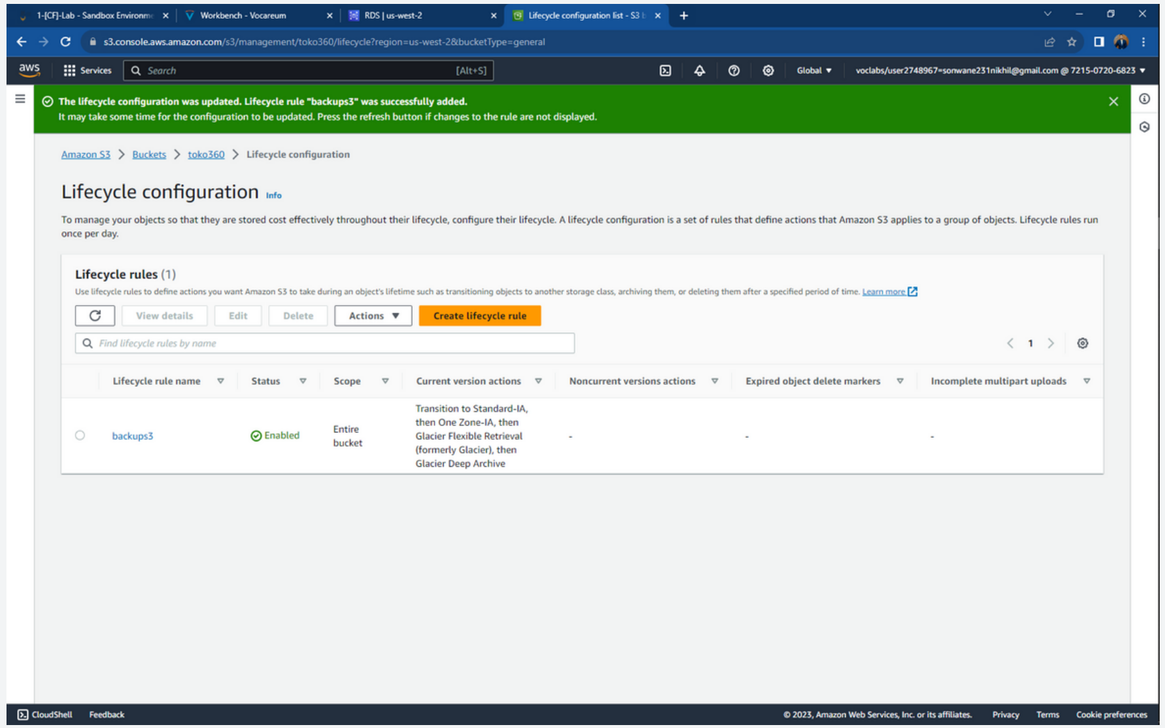
backup file



Create an S3 Bucket



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

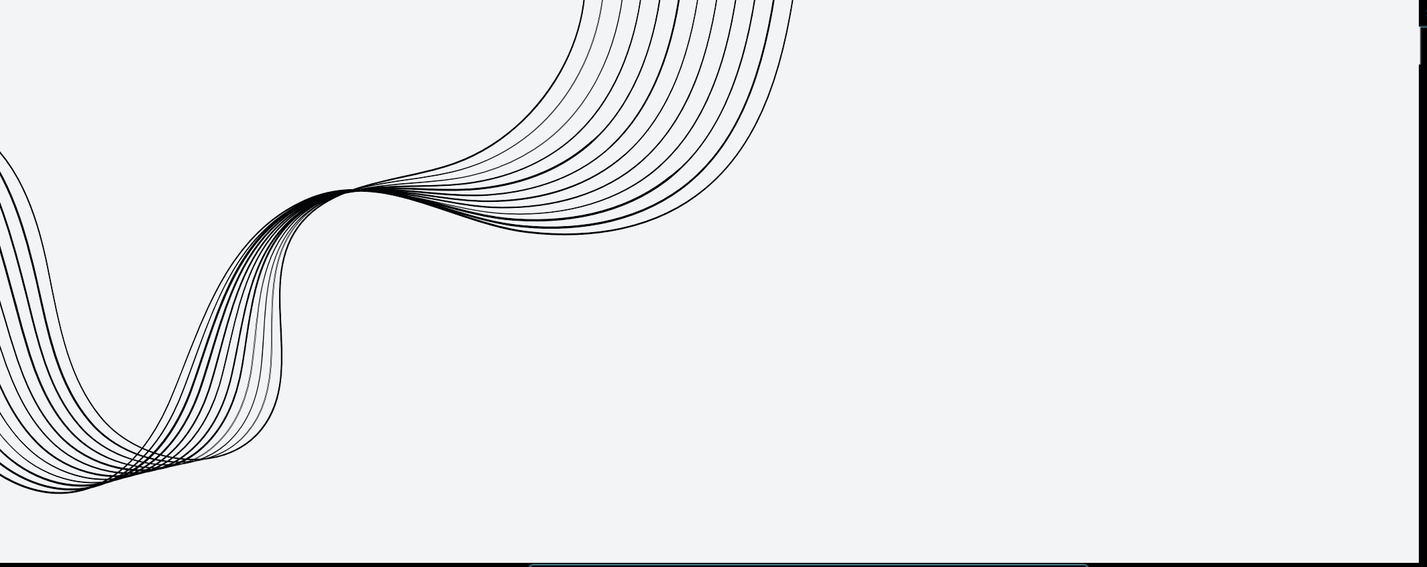


Adding Lifecycle Policies

- 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

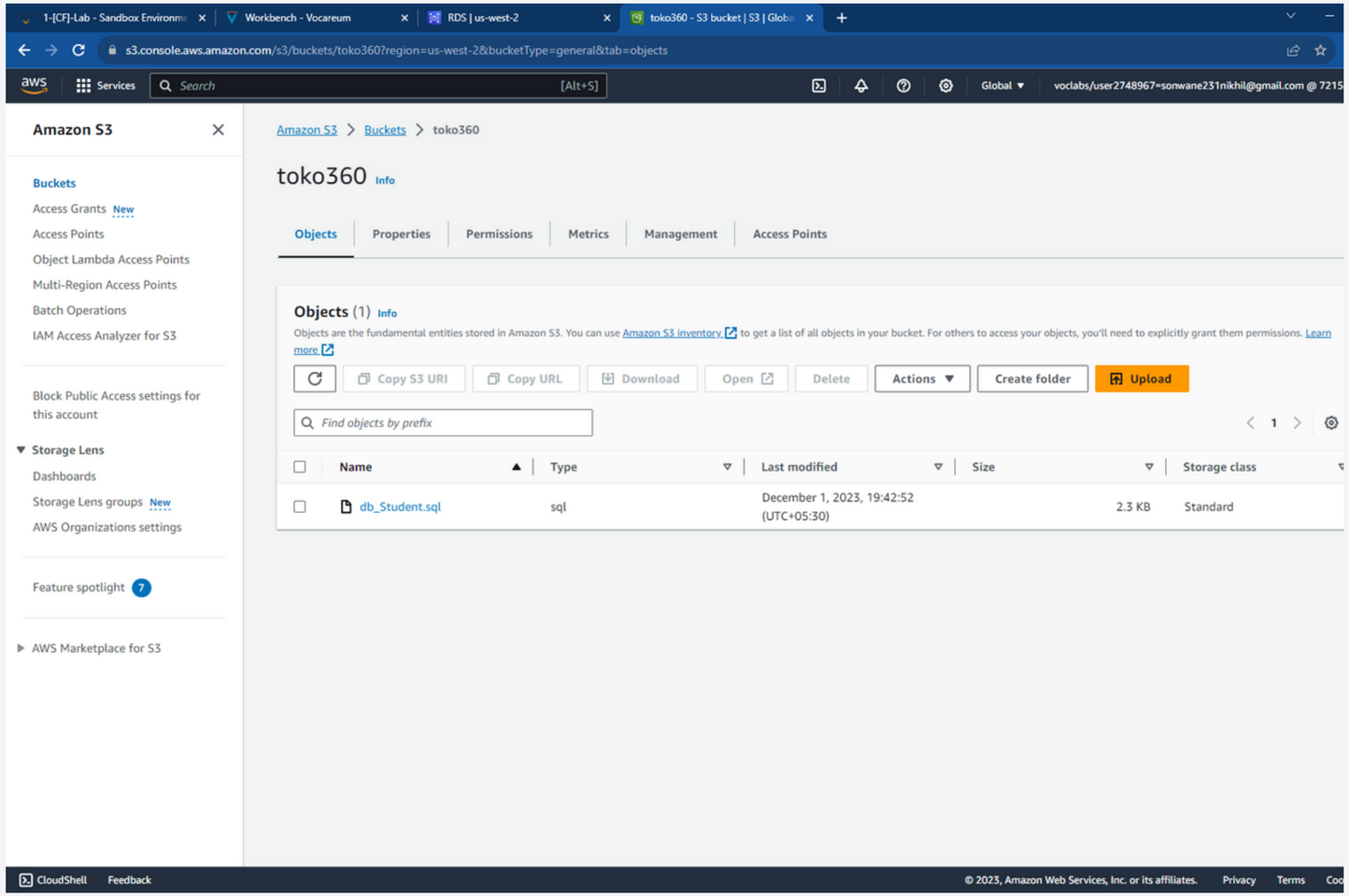
Write the Backup Script



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

DB_NAME = 'db_Student'
BACKUP_BD_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 = 'toko360'

    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")
```

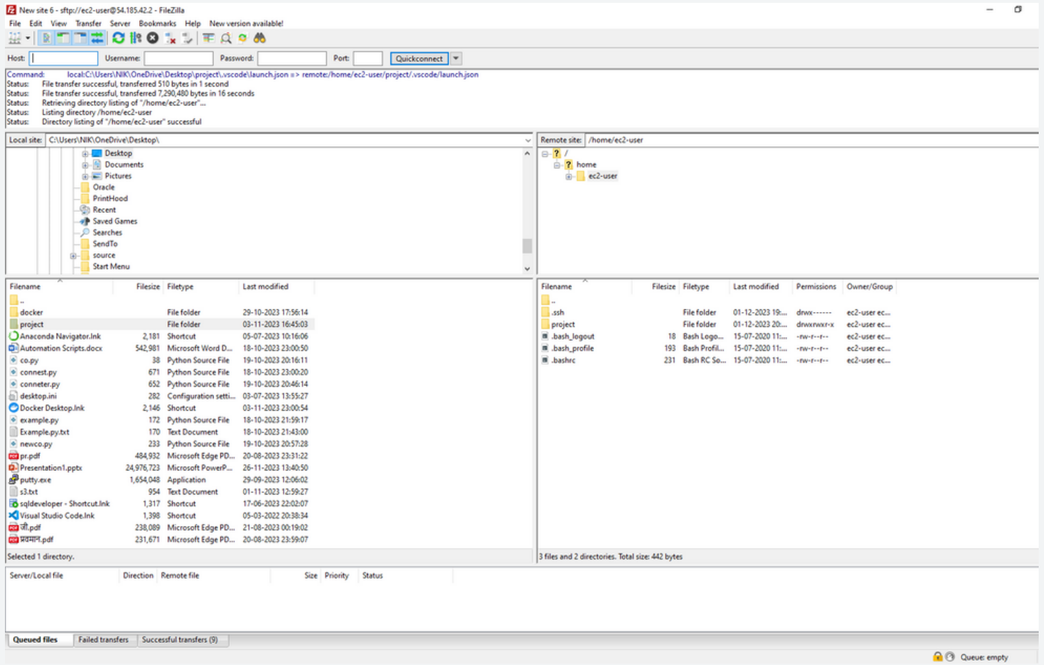
```
((1, 'NIKHIL', 'Aurangabad'), (1, 'NIKHIL', 'Aurangabad'), (1, 'NIKHIL', 'Aurangabad'))
mysqldump: [Warning] Using a password on the command line interface can be insecure.
Warning: A partial dump from a server that has GTIDs will by default include the GTIDs of all transactions, even those that 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 repres
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>
```

Write a python script for backup in s3:



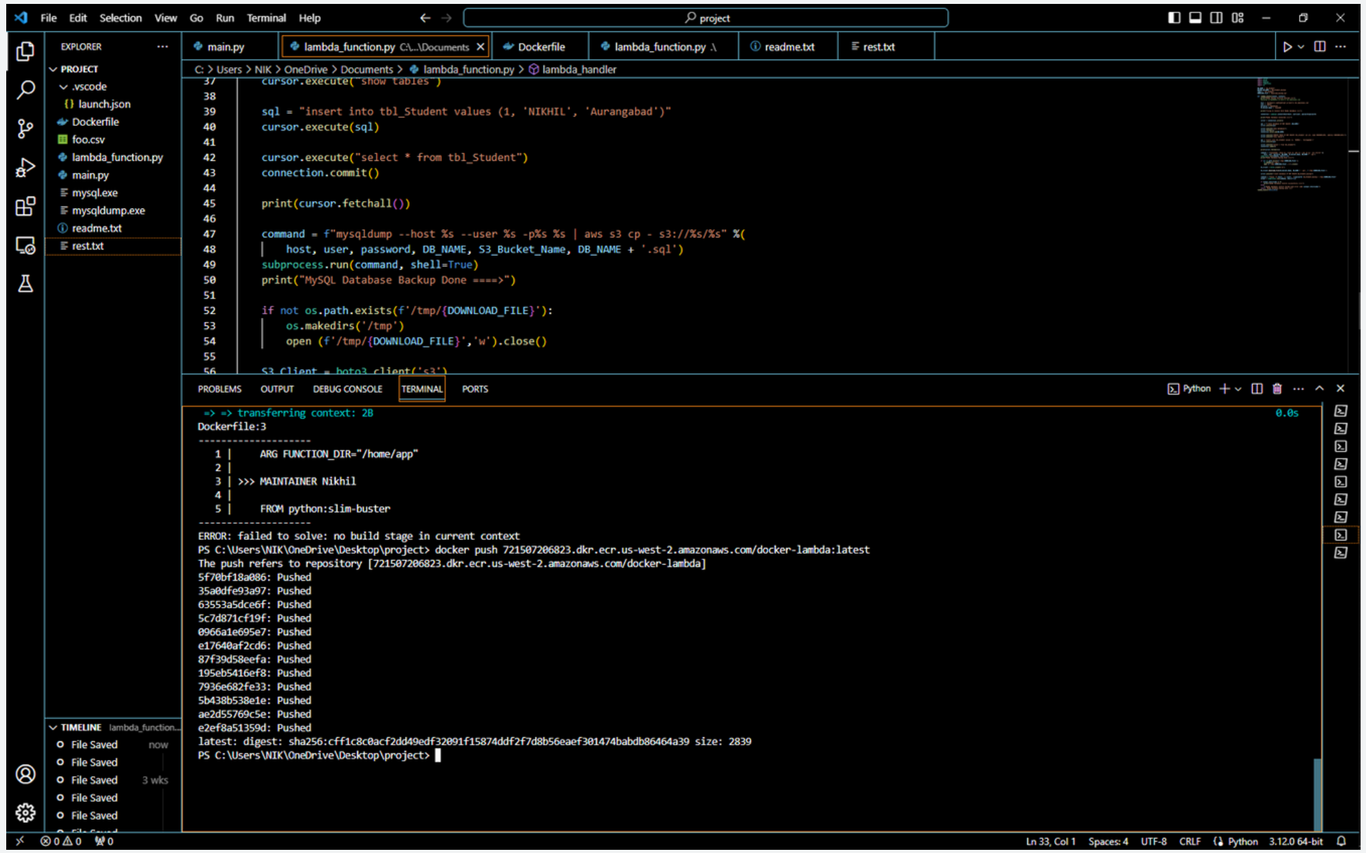
db_Student.sql bucket name is toko360

Package and Upload Your Script to Lambda

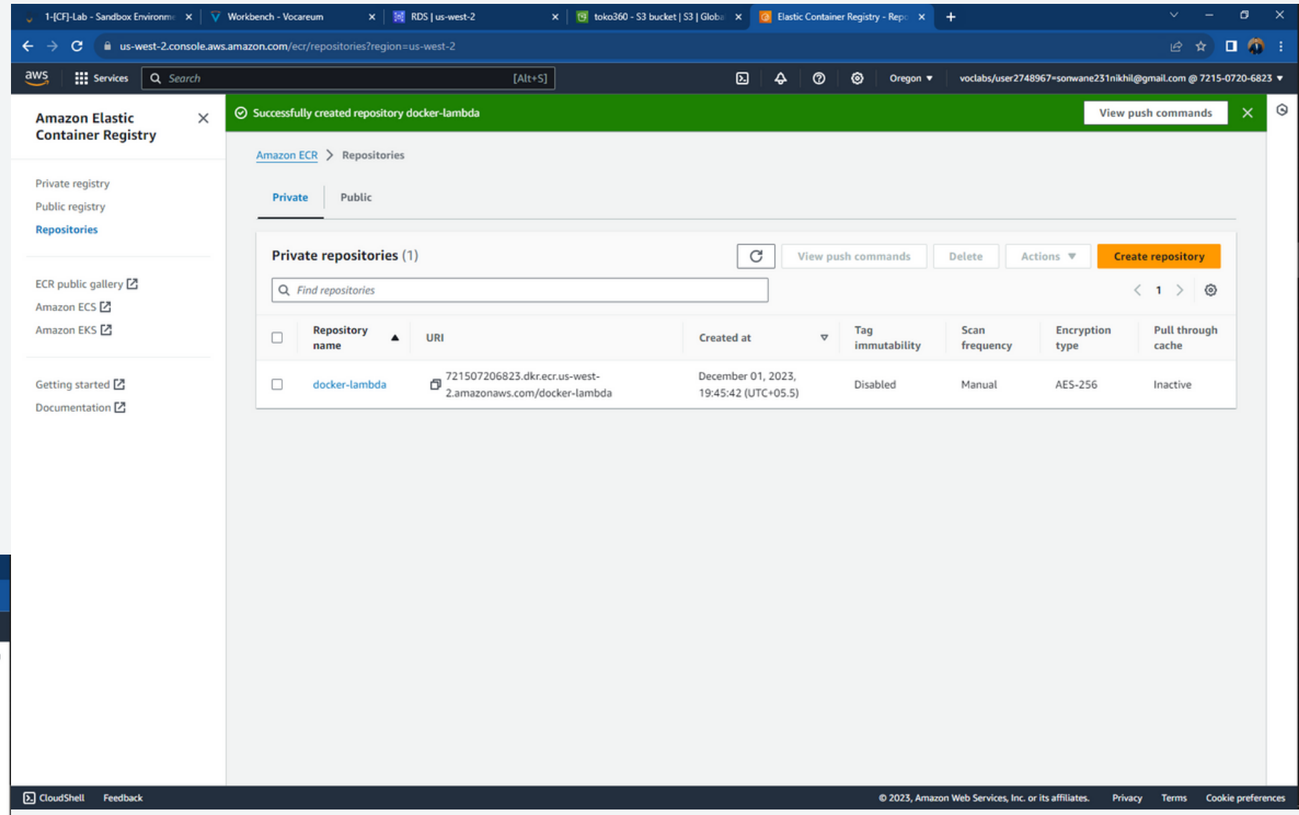


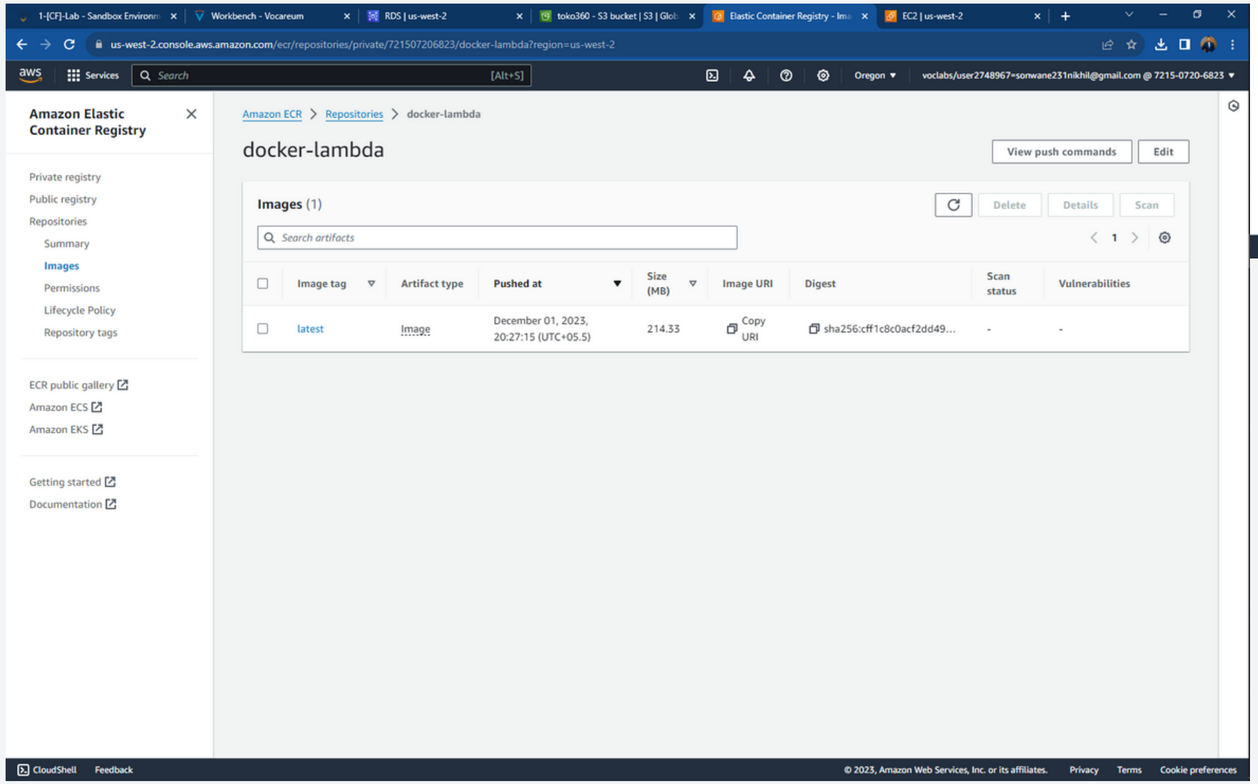
Filezilla using to file upload to cloud

Dockerfile

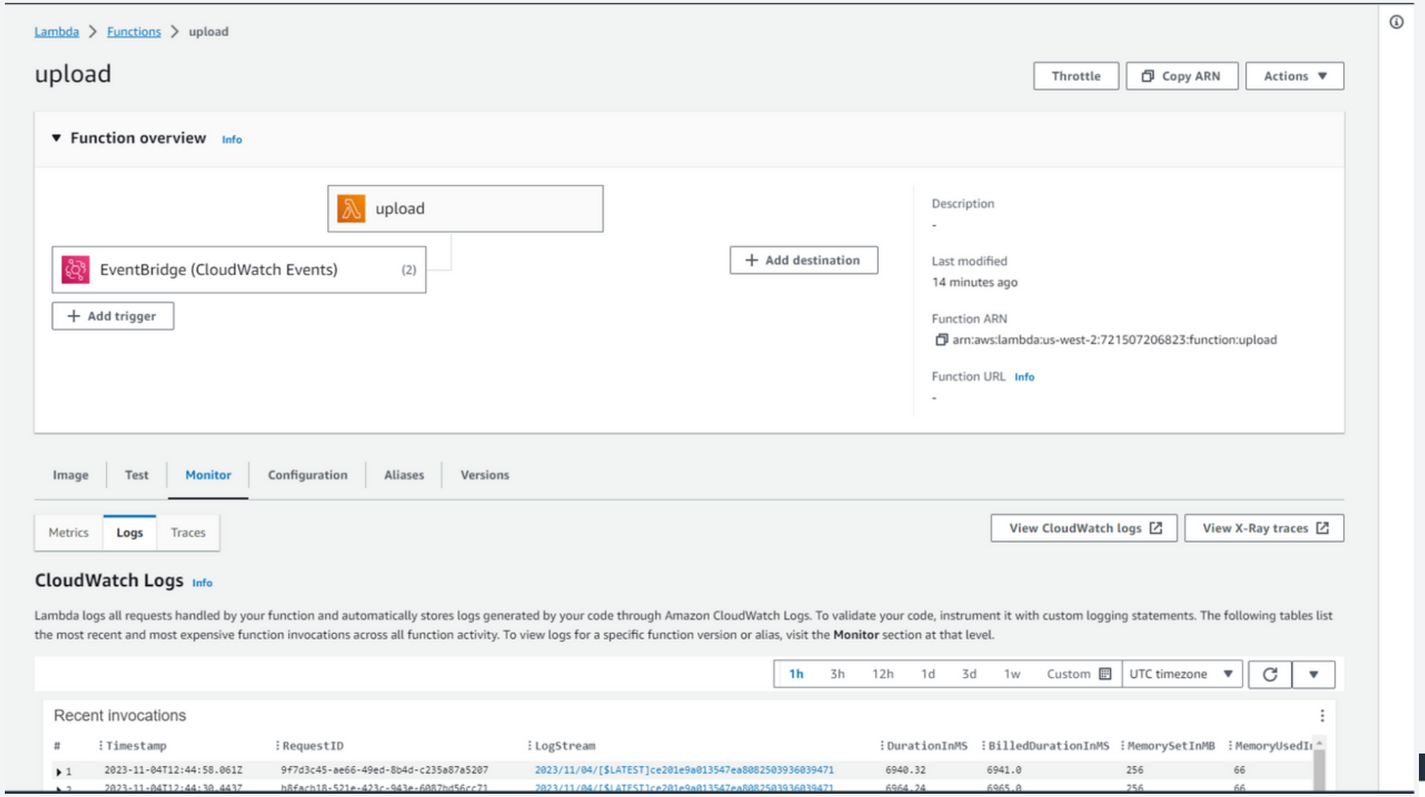


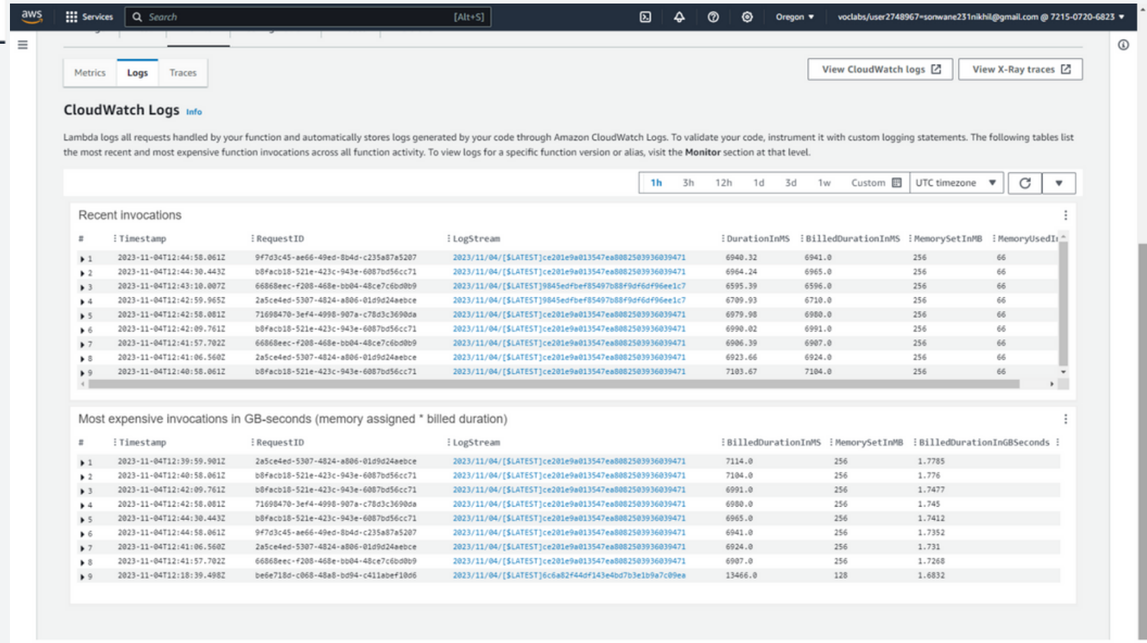
Docker name: docker_lambda





deploy docker image





Cloudwatch and output

Python Script : lambda_function.py

```
import cursor
import boto3
import pymysql
import subprocess
import os
DB_NAME = 'db_Student'
BACKUP_BD_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" ]
```