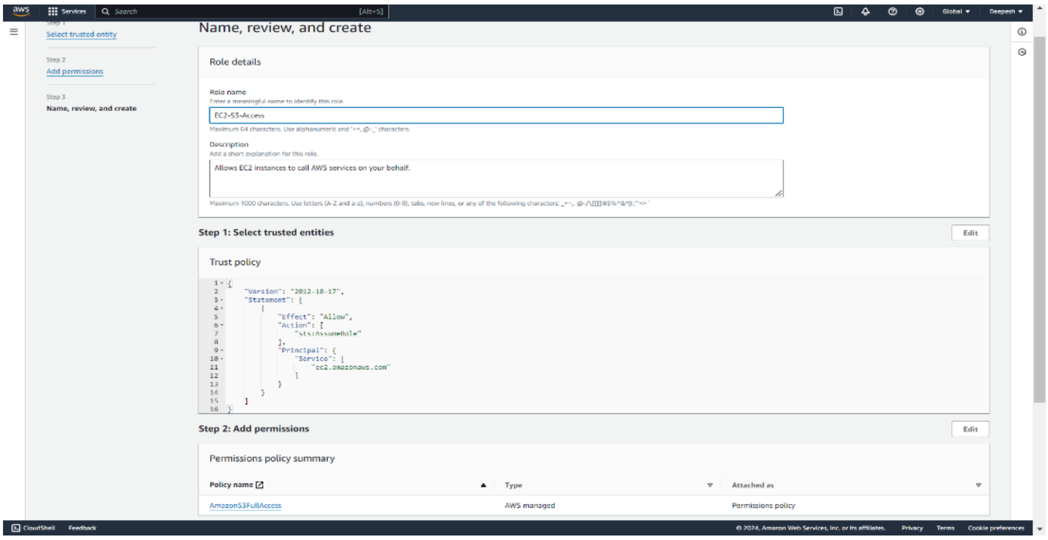
Backup Scripting with S3 Synchronization

1. Create an IAM Role which allows access to S3, and attach with your aws instance
2. Login to the console (CLI) of the server and configure a MariaDB server.
3. Write a shell script for taking up the backups of databases to a directory
4. Configure aws-cli within the server and synchronize the backup directory with any of the aws S3 buckets.
5. Verify the backups in S3

STEPS

1. Login to AWS account
2. Create an IAM Role: -
   a. Go to IAM Console: Navigate to the IAM service in the AWS Management Console.
   b. Create IAM Role:
      - Click on "Roles" in the left-hand menu.
      - Click "Create role".
      - Choose "AWS service" as the type of trusted entity, and select "EC2" as the service that will use this role.
      - Attach policies: Search for "AmazonS3FullAccess" and select it
      - Name the role appropriately (e.g., EC2-S3-Access).
   c. Attach IAM Role to EC2 Instance:
      - Once the role is created, go to your EC2 instances.
      - Select the instance you want to attach the role to.
      - Click on actions- Security- Modify IAM role
      - Choose the IAM role you created and attach it to the instance.



Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.
EC2-S3-Access
Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description
Add a short explanation for this role.
Allows EC2 instances to call AWS services on your behalf.
Maximum 1000 characters. Use letters (A-Z and a-z), numbers (0-9), tabs, new lines, or any of the following characters: _+=,. @-/\[{}]!#$%^&*();:"'<>`

Step 1: Select trusted entities

Edit

Trust policy

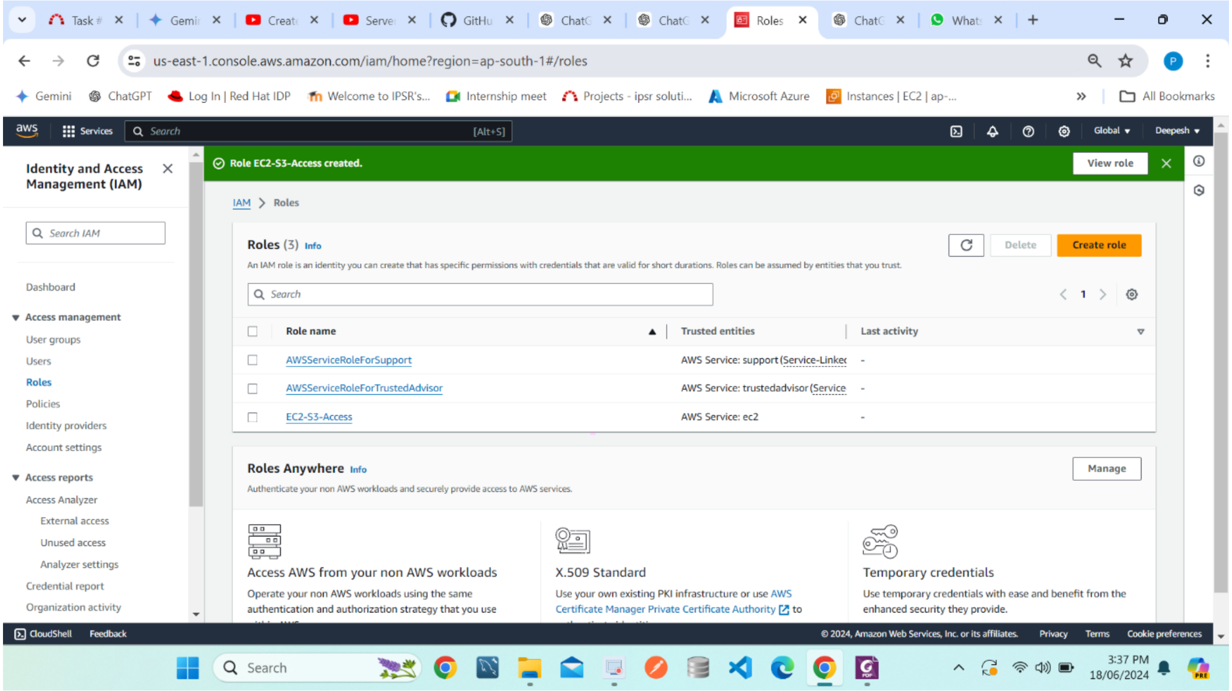
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Principal": {
                "Service": [
                    "ec2.amazonaws.com"
                ]
            }
        }
    ]
}
```

Step 2: Add permissions

Edit

Permissions policy summary

Policy name	Type	Attached as
AmazonS3FullAccess	AWS managed	Permissions policy



Role EC2-S3-Access created.

Roles (3) Info

An IAM role is an identity you can create that has specific permissions with credentials that are valid for short durations. Roles can be assumed by entities that you trust.

Role name	Trusted entities	Last activity
AWSServiceRoleForSupport	AWS Service: support (Service-Linkec	-
AWSServiceRoleForTrustedAdvisor	AWS Service: trustedadvisor (Service	-
EC2-S3-Access	AWS Service: ec2	-

Roles Anywhere Info

Authenticate your non AWS workloads and securely provide access to AWS services.

Access AWS from your non AWS workloads

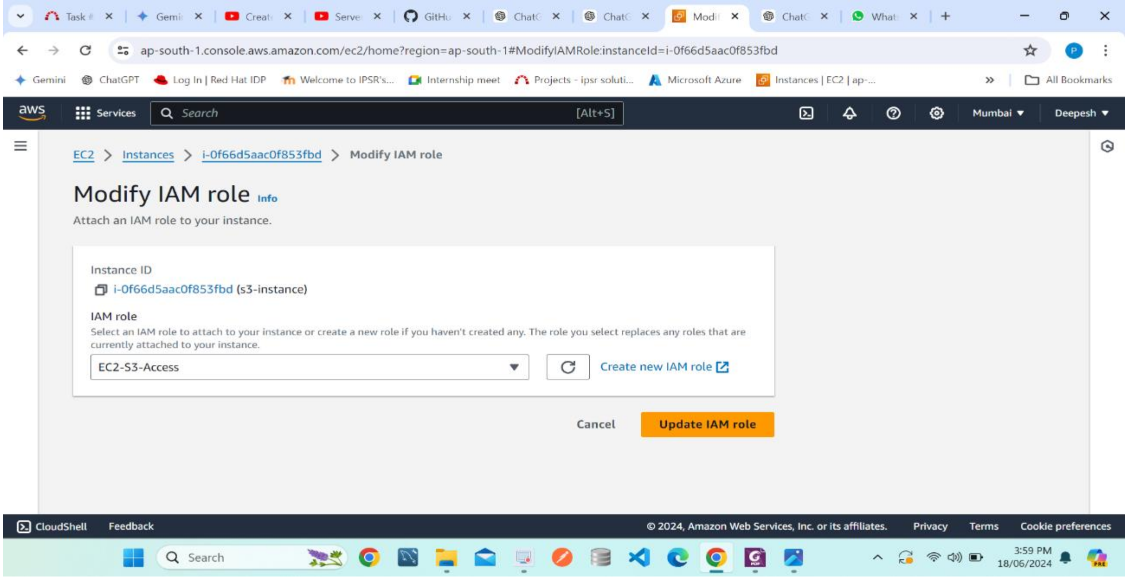
Operate your non AWS workloads using the same authentication and authorization strategy that you use

X.509 Standard

Use your own existing PKI infrastructure or use AWS Certificate Manager Private Certificate Authority to

Temporary credentials

Use temporary credentials with ease and benefit from the enhanced security they provide.


ap-south-1.console.aws.amazon.com/ec2/home?region=ap-south-1#ModifyIAMRole:instanceId=i-0f66d5aac0f853fbd
EC2 > Instances > i-0f66d5aac0f853fbd > Modify IAM role
Create new IAM role
Cancel
Update IAM role
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
3:59 PM
18/06/2024

Modify IAM role Info

Attach an IAM role to your instance.

Instance ID

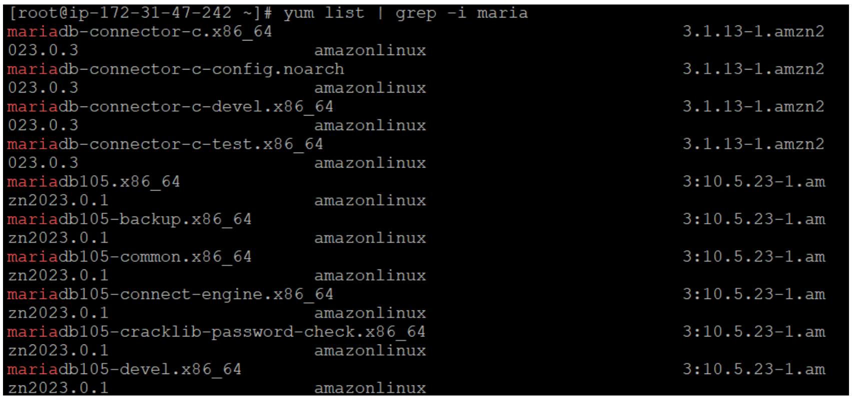
i-0f66d5aac0f853fbd (s3-instance)

IAM role

Select an IAM role to attach to your instance or create a new role if you haven't created any. The role you select replaces any roles that are currently attached to your instance.

EC2-S3-Access

3. Configure MariaDB Server



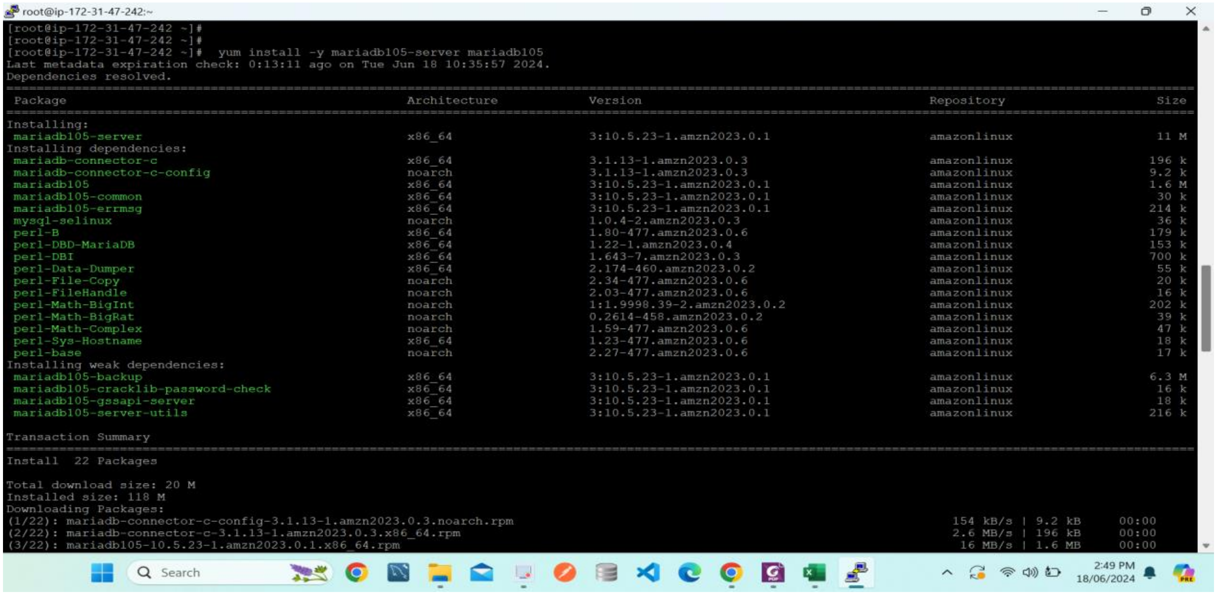
```
[root@ip-172-31-47-242 ~]# yum list | grep -i maria
mariadb-connector-c.x86_64                                     3.1.13-1.amzn2
023.0.3                         amazonlinux
mariadb-connector-c-config.noarch                              3.1.13-1.amzn2
023.0.3                         amazonlinux
mariadb-connector-c-devel.x86_64                               3.1.13-1.amzn2
023.0.3                         amazonlinux
mariadb-connector-c-test.x86_64                                3.1.13-1.amzn2
023.0.3                         amazonlinux
mariadb105.x86_64                                              3:10.5.23-1.am
zn2023.0.1                      amazonlinux
mariadb105-backup.x86_64                                       3:10.5.23-1.am
zn2023.0.1                      amazonlinux
mariadb105-common.x86_64                                       3:10.5.23-1.am
zn2023.0.1                      amazonlinux
mariadb105-connect-engine.x86_64                               3:10.5.23-1.am
zn2023.0.1                      amazonlinux
mariadb105-cracklib-password-check.x86_64                      3:10.5.23-1.am
zn2023.0.1                      amazonlinux
mariadb105-devel.x86_64                                        3:10.5.23-1.am
zn2023.0.1                      amazonlinux
```

a. Install mariadb

yum install -y mariadb105-server mariadb105

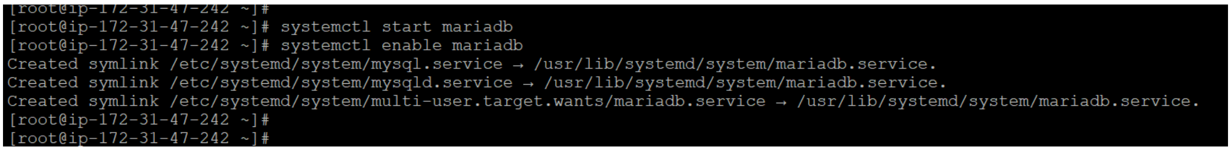
systemctl start mariadb

systemctl enable mariadb



root@ip-172-31-47-242:~

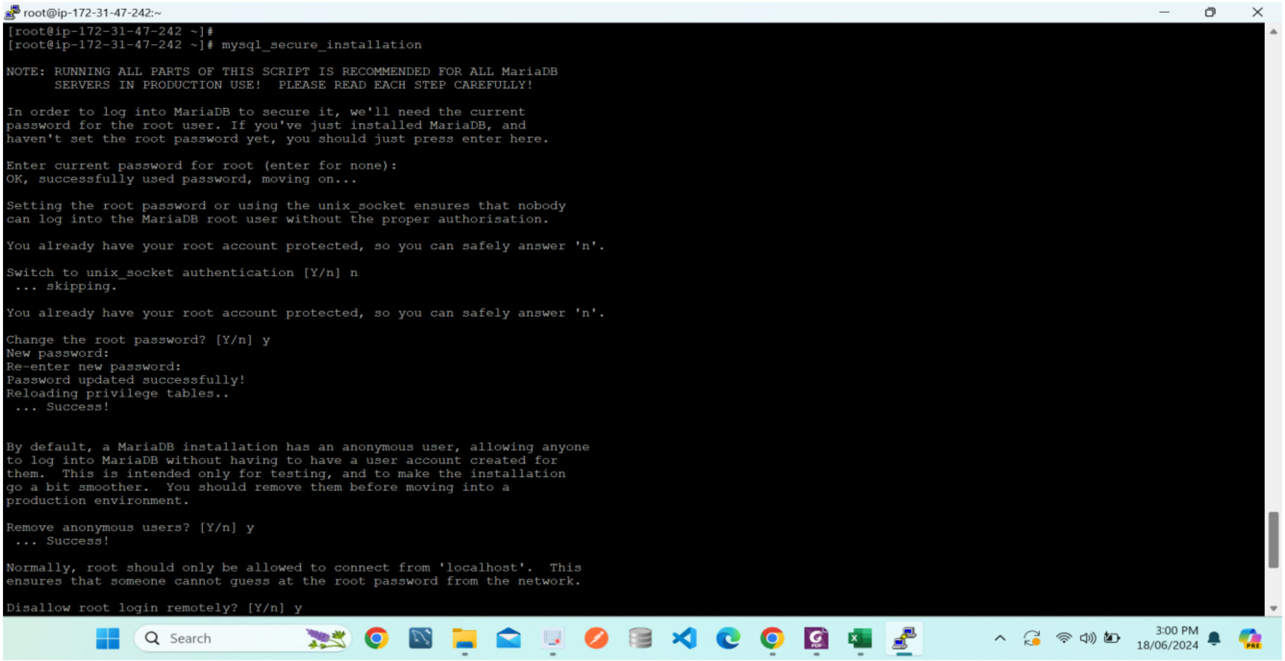
```
[root@ip-172-31-47-242 ~]#
[root@ip-172-31-47-242 ~]#
[root@ip-172-31-47-242 ~]#  yum install -y mariadb105-server mariadb105
Last metadata expiration check: 0:13:11 ago on Tue Jun 18 10:35:57 2024.
Dependencies resolved.
==========================================================================================================================
 Package                                 Architecture   Version                            Repository             Size
==========================================================================================================================
Installing:
 mariadb105-server                       x86_64         3:10.5.23-1.amzn2023.0.1           amazonlinux            11 M
Installing dependencies:
 mariadb-connector-c                     x86_64         3.1.13-1.amzn2023.0.3              amazonlinux           196 k
 mariadb-connector-c-config              noarch         3.1.13-1.amzn2023.0.3              amazonlinux           9.2 k
 mariadb105                              x86_64         3:10.5.23-1.amzn2023.0.1           amazonlinux           1.6 M
 mariadb105-common                       x86_64         3:10.5.23-1.amzn2023.0.1           amazonlinux            30 k
 mariadb105-errmsg                       x86_64         3:10.5.23-1.amzn2023.0.1           amazonlinux           214 k
 mysql-selinux                           noarch         1.0.4-2.amzn2023.0.3               amazonlinux            36 k
 perl-B                                  x86_64         1.80-477.amzn2023.0.6              amazonlinux           179 k
 perl-DBD-MariaDB                        x86_64         1.22-1.amzn2023.0.4                amazonlinux           153 k
 perl-DBI                                x86_64         1.643-7.amzn2023.0.3               amazonlinux           700 k
 perl-Data-Dumper                        x86_64         2.174-460.amzn2023.0.2             amazonlinux            55 k
 perl-File-Copy                          noarch         2.34-477.amzn2023.0.6              amazonlinux            20 k
 perl-FileHandle                         noarch         2.03-477.amzn2023.0.6              amazonlinux            16 k
 perl-Math-BigInt                        noarch         1:1.9998.39-2.amzn2023.0.2         amazonlinux           202 k
 perl-Math-BigRat                        noarch         0.2614-458.amzn2023.0.2            amazonlinux            39 k
 perl-Math-Complex                       noarch         1.59-477.amzn2023.0.6              amazonlinux            47 k
 perl-Sys-Hostname                       x86_64         1.23-477.amzn2023.0.6              amazonlinux            18 k
 perl-base                               noarch         2.27-477.amzn2023.0.6              amazonlinux            17 k
Installing weak dependencies:
 mariadb105-backup                       x86_64         3:10.5.23-1.amzn2023.0.1           amazonlinux           6.3 M
 mariadb105-cracklib-password-check      x86_64         3:10.5.23-1.amzn2023.0.1           amazonlinux            16 k
 mariadb105-gssapi-server                x86_64         3:10.5.23-1.amzn2023.0.1           amazonlinux            18 k
 mariadb105-server-utils                 x86_64         3:10.5.23-1.amzn2023.0.1           amazonlinux           216 k

Transaction Summary
==========================================================================================================================
Install  22 Packages

Total download size: 20 M
Installed size: 118 M
Downloading Packages:
(1/22): mariadb-connector-c-config-3.1.13-1.amzn2023.0.3.noarch.rpm                  154 kB/s | 9.2 kB     00:00
(2/22): mariadb-connector-c-3.1.13-1.amzn2023.0.3.x86_64.rpm                         2.6 MB/s | 196 kB     00:00
(3/22): mariadb105-10.5.23-1.amzn2023.0.1.x86_64.rpm                                  16 MB/s | 1.6 MB     00:00
```



```
[root@ip-172-31-47-242 ~]#
[root@ip-172-31-47-242 ~]# systemctl start mariadb
[root@ip-172-31-47-242 ~]# systemctl enable mariadb
Created symlink /etc/systemd/system/mysql.service → /usr/lib/systemd/system/mariadb.service.
Created symlink /etc/systemd/system/mysqld.service → /usr/lib/systemd/system/mariadb.service.
Created symlink /etc/systemd/system/multi-user.target.wants/mariadb.service → /usr/lib/systemd/system/mariadb.service.
[root@ip-172-31-47-242 ~]#
[root@ip-172-31-47-242 ~]#
```

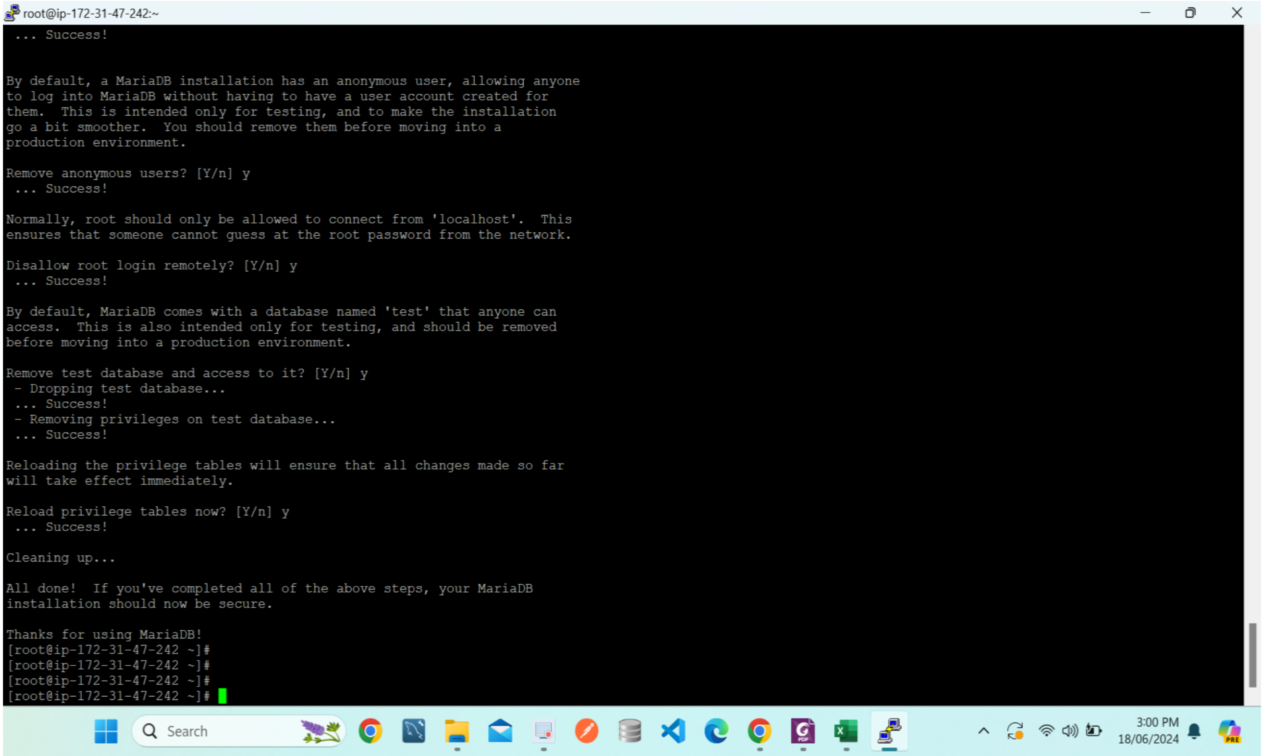
b. Secure MariaDB installation:

mysql_secure_installation (current password for root none so press enter)



root@ip-172-31-47-242:~

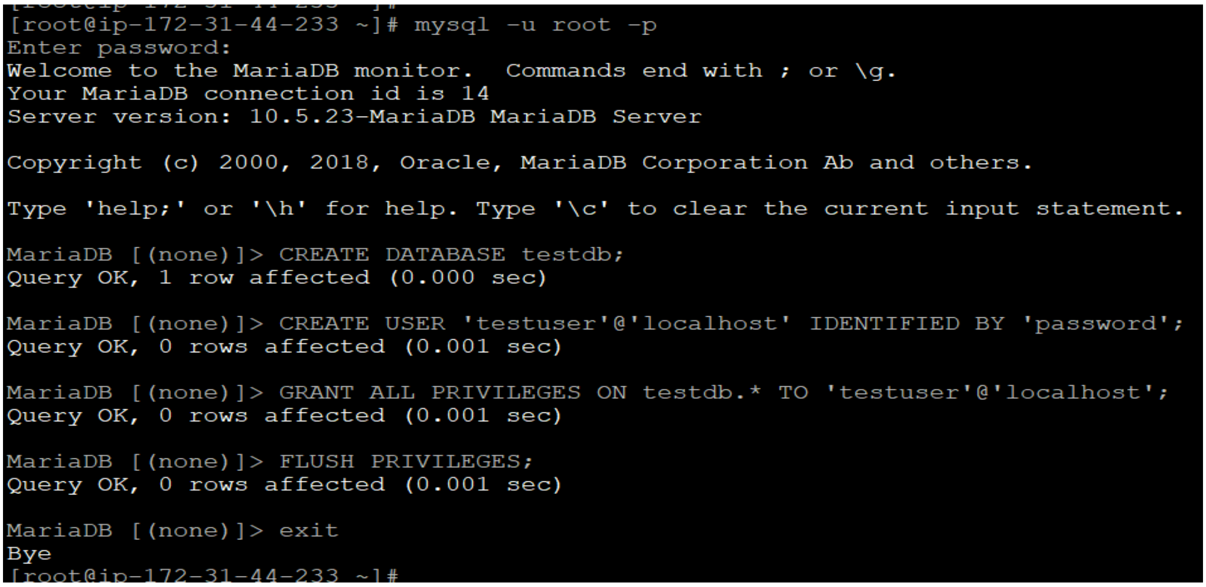
```
[root@ip-172-31-47-242 ~]#
[root@ip-172-31-47-242 ~]# mysql_secure_installation

NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user. If you've just installed MariaDB, and
haven't set the root password yet, you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password or using the unix_socket ensures that nobody
can log into the MariaDB root user without the proper authorisation.

You already have your root account protected, so you can safely answer 'n'.

Switch to unix_socket authentication [Y/n] n
 ... skipping.

You already have your root account protected, so you can safely answer 'n'.

Change the root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
```



```
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] y
 ... Success!

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] y
 ... Success!

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] y
 - Dropping test database...
 ... Success!
 - Removing privileges on test database...
 ... Success!

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
[root@ip-172-31-47-242 ~]#
[root@ip-172-31-47-242 ~]#
[root@ip-172-31-47-242 ~]#
[root@ip-172-31-47-242 ~]#
```

c. Create a test database and user:

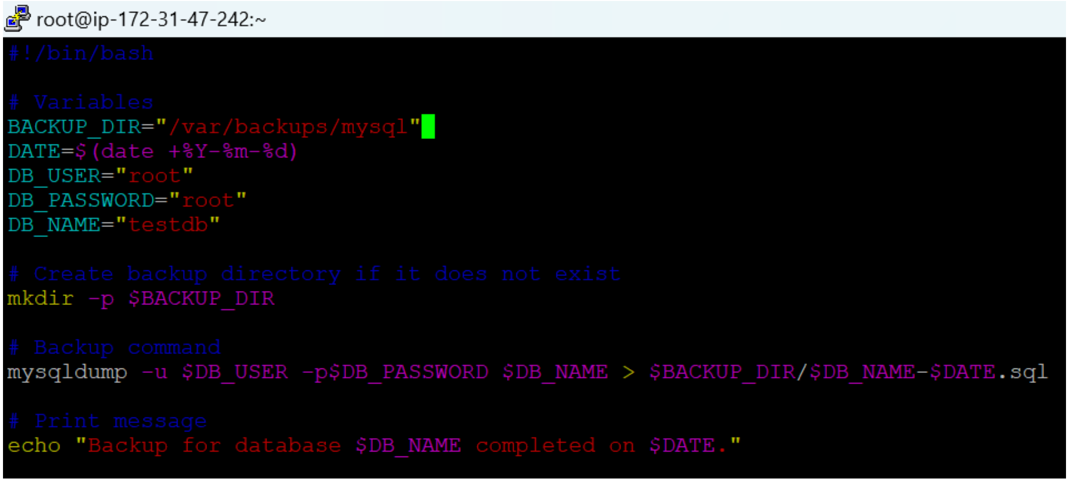
```
# mysql -u root -p
# CREATE DATABASE testdb;
# CREATE USER 'testuser'@'localhost' IDENTIFIED BY 'password';
# GRANT ALL PRIVILEGES ON testdb.* TO 'testuser'@'localhost';
# FLUSH PRIVILEGES;
# EXIT;
```



```
[root@ip-172-31-44-233 ~]# mysql -u root -p
Enter password:
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 14
Server version: 10.5.23-MariaDB MariaDB Server

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> CREATE DATABASE testdb;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> CREATE USER 'testuser'@'localhost' IDENTIFIED BY 'password';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> GRANT ALL PRIVILEGES ON testdb.* TO 'testuser'@'localhost';
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.001 sec)

MariaDB [(none)]> exit
Bye
[root@ip-172-31-44-233 ~]#
```

4. Write a Shell Script for Backups

   # vi backup.sh

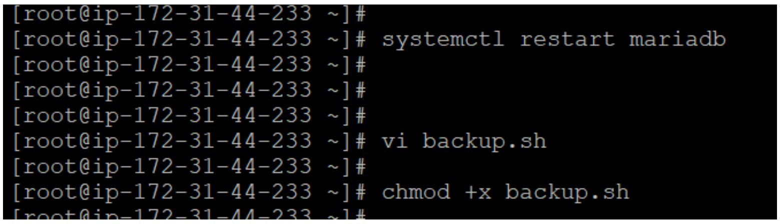


root@ip-172-31-47-242:~

```
#!/bin/bash

# Variables
BACKUP_DIR="/var/backups/mysql"
DATE=$(date +%Y-%m-%d)
DB_USER="root"
DB_PASSWORD="root"
DB_NAME="testdb"

# Create backup directory if it does not exist
mkdir -p $BACKUP_DIR

# Backup command
mysqldump -u $DB_USER -p$DB_PASSWORD $DB_NAME > $BACKUP_DIR/$DB_NAME-$DATE.sql

# Print message
echo "Backup for database $DB_NAME completed on $DATE."
```

5. Make sure the script is executable before running it:

   # chmod +x backup.sh



```
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]# systemctl restart mariadb
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]# vi backup.sh
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]# chmod +x backup.sh
[root@ip-172-31-44-233 ~]#
```

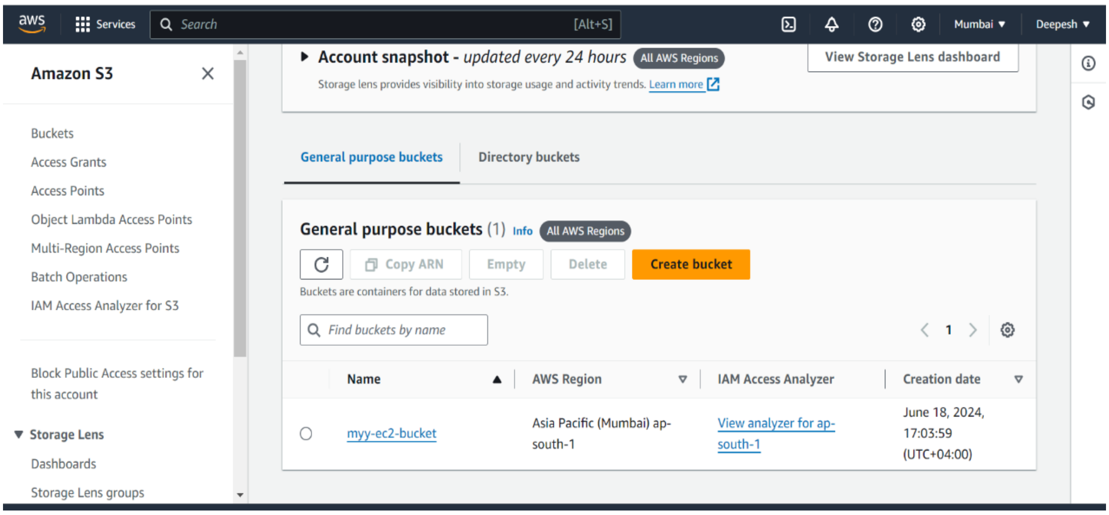
6. S3 bucket Creation

   a. Navigate to the S3 service: In the console, type "S3" in the search bar and select "S3"from the list.

   b. Create a New Bucket: Click the “Create bucket” button.

   c. Provide a unique name for your bucket (e.g., myy-ec2-bucket).

   d. Select the AWS Region where you want the bucket to be created and Click “Create bucket” at the bottom of the page



▶ Account snapshot - *updated every 24 hours* All AWS Regions

Storage lens provides visibility into storage usage and activity trends. Learn more

General purpose buckets (1)

Buckets are containers for data stored in S3.

	Name ▲	AWS Region ▽	IAM Access Analyzer	Creation date ▽
○	myy-ec2-bucket	Asia Pacific (Mumbai) ap-south-1	View analyzer for ap-south-1	June 18, 2024, 17:03:59 (UTC+04:00)

7. Configure AWS CLI and Sync Backups to S3

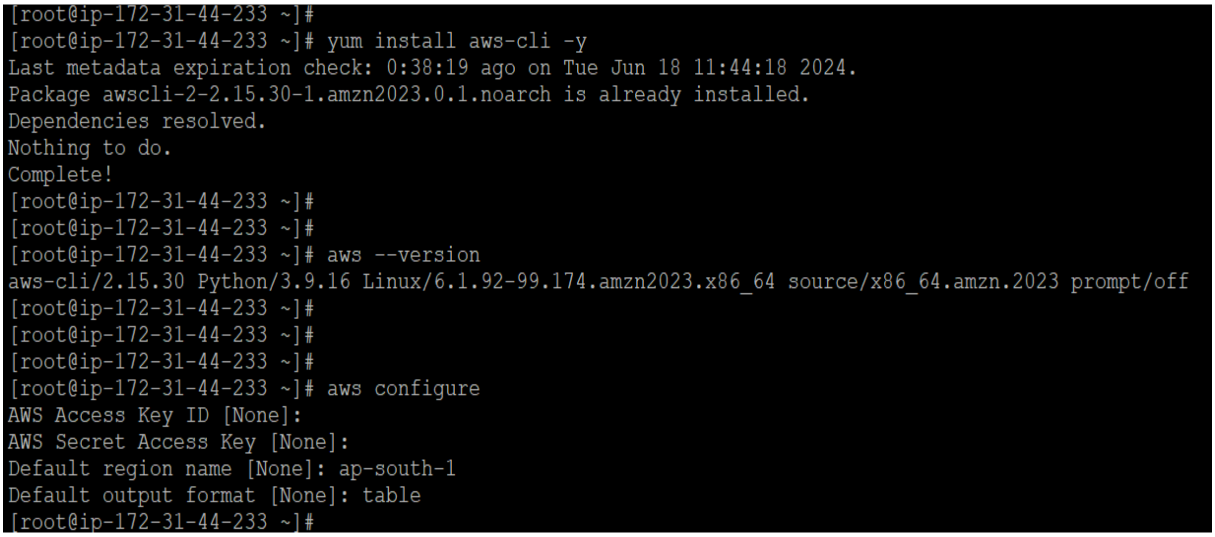
   a. Install AWS CLI:

      # yum install aws-cli -y

   b. Configure AWS CLI with the IAM user's credentials:

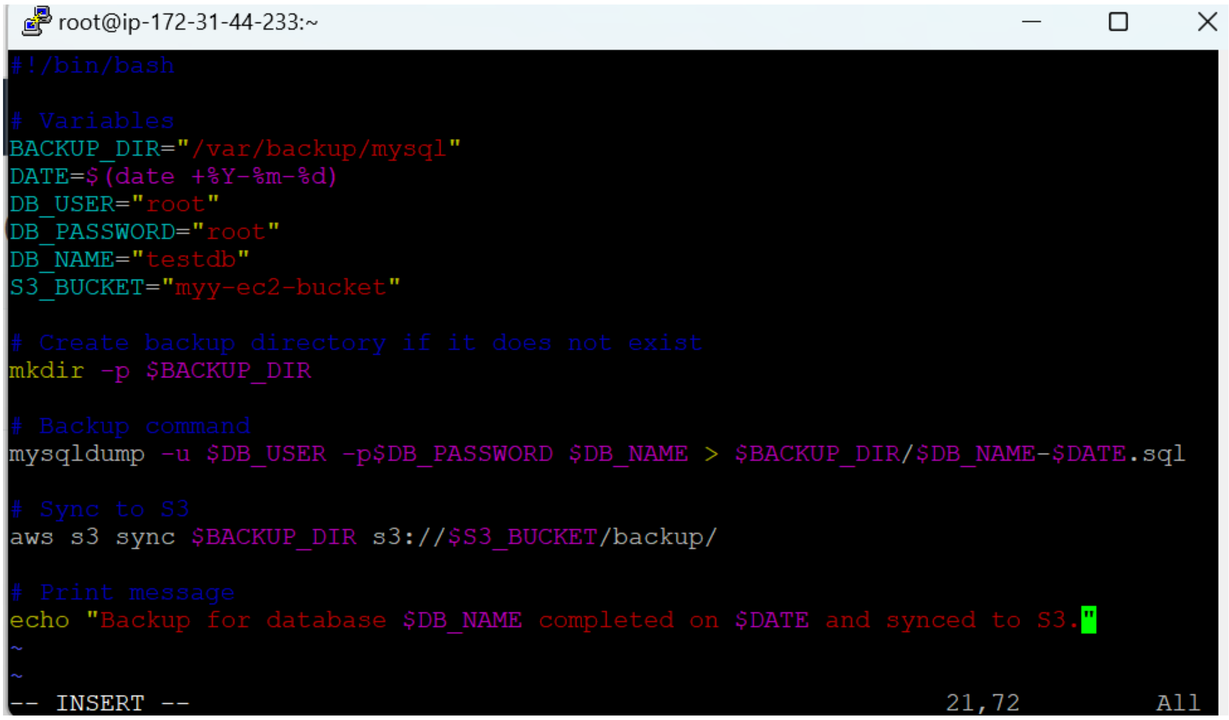
      # aws configure

      Enter the Access Key ID (blank), Secret Access Key(blank), default region name(ap-south-1), and default output format(table).



```
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]# yum install aws-cli -y
Last metadata expiration check: 0:38:19 ago on Tue Jun 18 11:44:18 2024.
Package awscli-2-2.15.30-1.amzn2023.0.1.noarch is already installed.
Dependencies resolved.
Nothing to do.
Complete!
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]# aws --version
aws-cli/2.15.30 Python/3.9.16 Linux/6.1.92-99.174.amzn2023.x86_64 source/x86_64.amzn.2023 prompt/off
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]# aws configure
AWS Access Key ID [None]:
AWS Secret Access Key [None]:
Default region name [None]: ap-south-1
Default output format [None]: table
[root@ip-172-31-44-233 ~]#
```

   c. Update the backup script to sync with S3:



```
#!/bin/bash

# Variables
BACKUP_DIR="/var/backup/mysql"
DATE=$(date +%Y-%m-%d)
DB_USER="root"
DB_PASSWORD="root"
DB_NAME="testdb"
S3_BUCKET="myy-ec2-bucket"

# Create backup directory if it does not exist
mkdir -p $BACKUP_DIR

# Backup command
mysqldump -u $DB_USER -p$DB_PASSWORD $DB_NAME > $BACKUP_DIR/$DB_NAME-$DATE.sql

# Sync to S3
aws s3 sync $BACKUP_DIR s3://$S3_BUCKET/backup/

# Print message
echo "Backup for database $DB_NAME completed on $DATE and synced to S3."
~
~
-- INSERT --                                                    21,72         All
```

8. Test the Backup Script

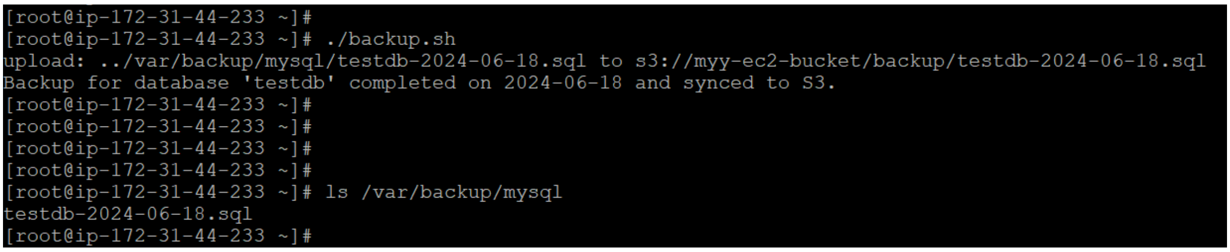
   a. Run the Backup Script:

      - Execute the script manually:

        #./backup.sh

   b. Check the backup directory to ensure a backup file with the current date is created:

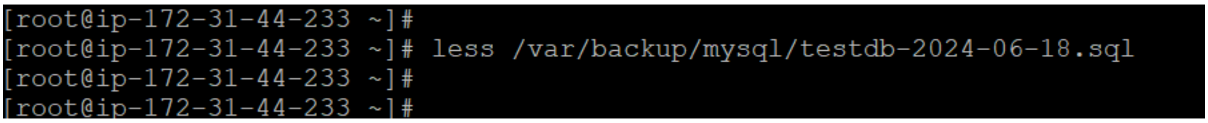
      # ls /var/backup/mysql



```
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]# ./backup.sh
upload: ../var/backup/mysql/testdb-2024-06-18.sql to s3://myy-ec2-bucket/backup/testdb-2024-06-18.sql
Backup for database 'testdb' completed on 2024-06-18 and synced to S3.
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]# ls /var/backup/mysql
testdb-2024-06-18.sql
[root@ip-172-31-44-233 ~]#
```

   c. Check Backup File Content: - Inspect the backup file to ensure it contains SQL data, ensure it contains SQL commands and data from your database. Here no content in the database

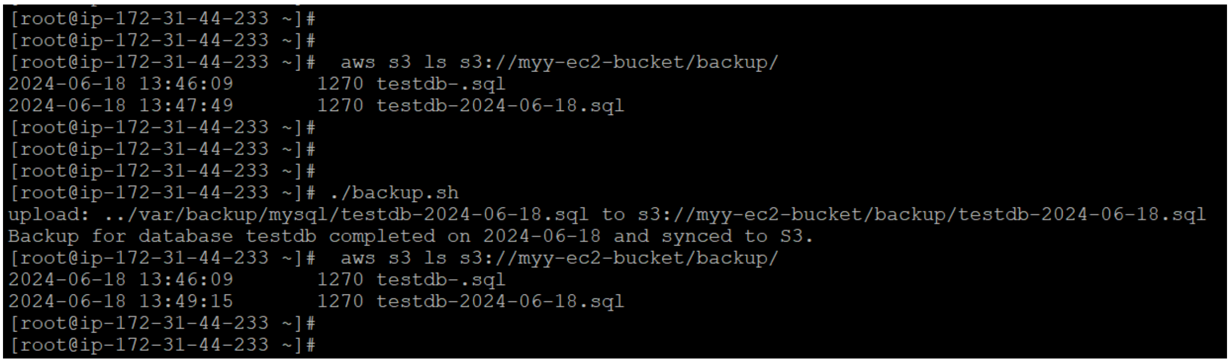
      # less /var/backup/mysql/testdb-2024-06-18.sql



```
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]# less /var/backup/mysql/testdb-2024-06-18.sql
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
```

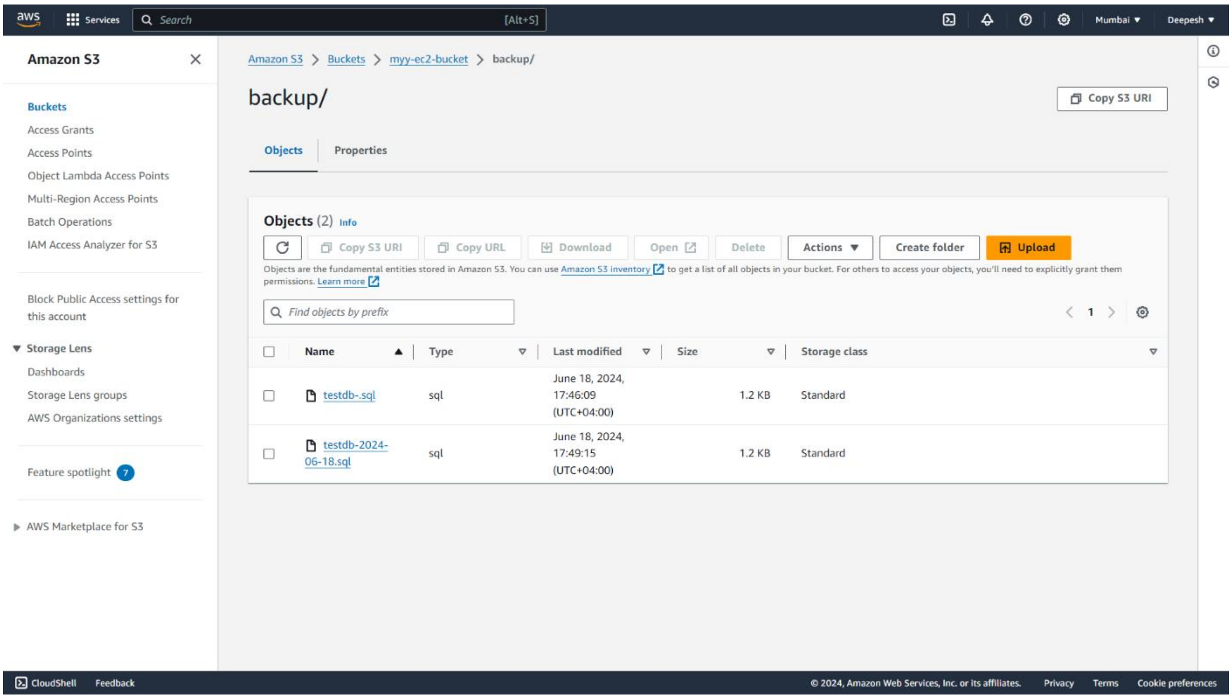
   d. Verify Files in S3: List the contents of the S3 bucket to confirm the files were uploaded. You should see the backup files listed.

      # aws s3 ls s3://myy-ec2-bucket/backup/



```
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#  aws s3 ls s3://myy-ec2-bucket/backup/
2024-06-18 13:46:09       1270 testdb-.sql
2024-06-18 13:47:49       1270 testdb-2024-06-18.sql
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]# ./backup.sh
upload: ../var/backup/mysql/testdb-2024-06-18.sql to s3://myy-ec2-bucket/backup/testdb-2024-06-18.sql
Backup for database testdb completed on 2024-06-18 and synced to S3.
[root@ip-172-31-44-233 ~]#  aws s3 ls s3://myy-ec2-bucket/backup/
2024-06-18 13:46:09       1270 testdb-.sql
2024-06-18 13:49:15       1270 testdb-2024-06-18.sql
[root@ip-172-31-44-233 ~]#
[root@ip-172-31-44-233 ~]#
```

e. View Backup Files in S3 Bucket



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

□	Name	Type	Last modified	Size	Storage class
□	testdb-.sql	sql	June 18, 2024, 17:46:09 (UTC+04:00)	1.2 KB	Standard
□	testdb-2024-06-18.sql	sql	June 18, 2024, 17:49:15 (UTC+04:00)	1.2 KB	Standard