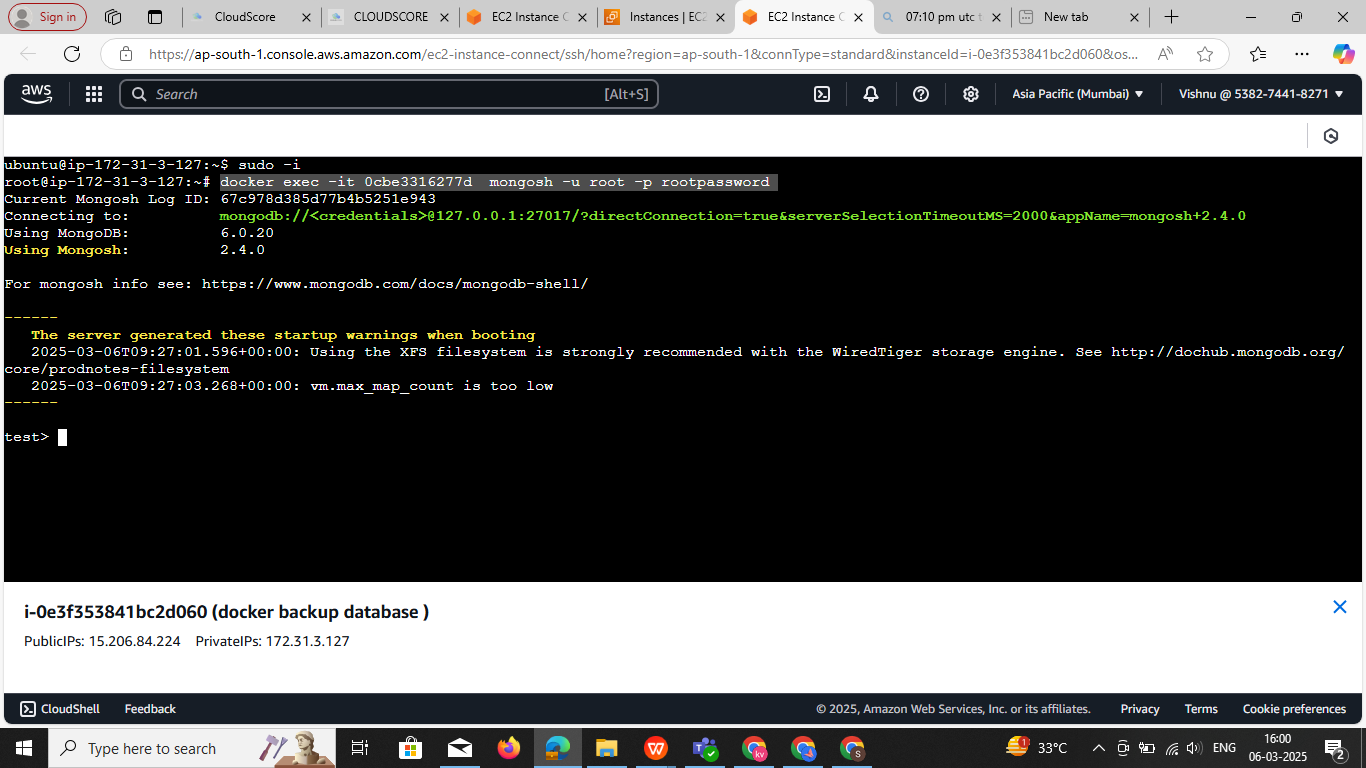
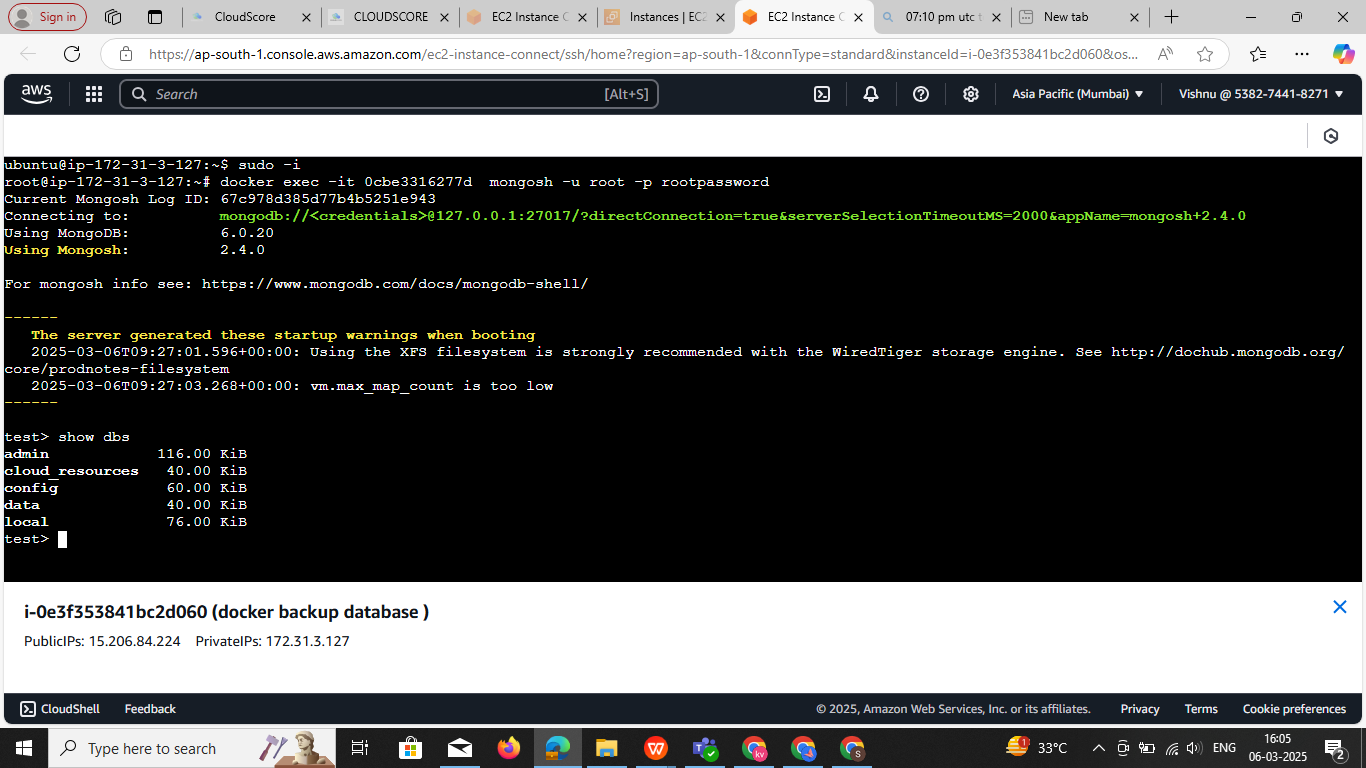
## ****Step 1: Log In to MongoDB****

docker exec -it 0cbe3316277d mongosh -u root -p rootpassword



#### Show All Databases

show dbs



#### ****Explanation:****

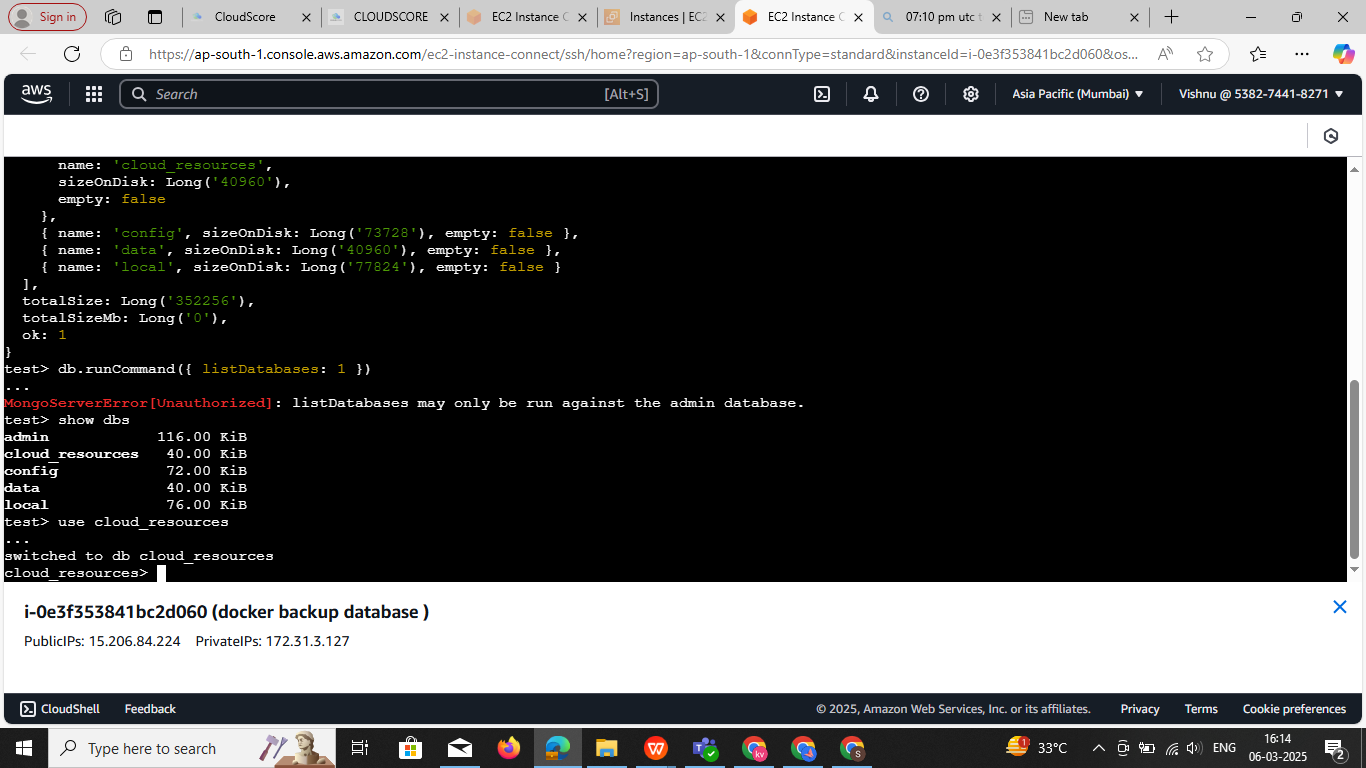
* Lists **all databases** in the MongoDB server.
* Displays **database names** along with their **size**.
* Only databases that contain at least **one collection** and **some data** are shown.

Switch to a Database (or Create a New One)

use cloud\_resources use <database\_name>

#### ****Explanation:****

* If cloud\_resources exists, it **switches** to that database.
* If cloud\_resources does **not exist**, it **creates** it (but it won’t appear in show dbs until you insert data).



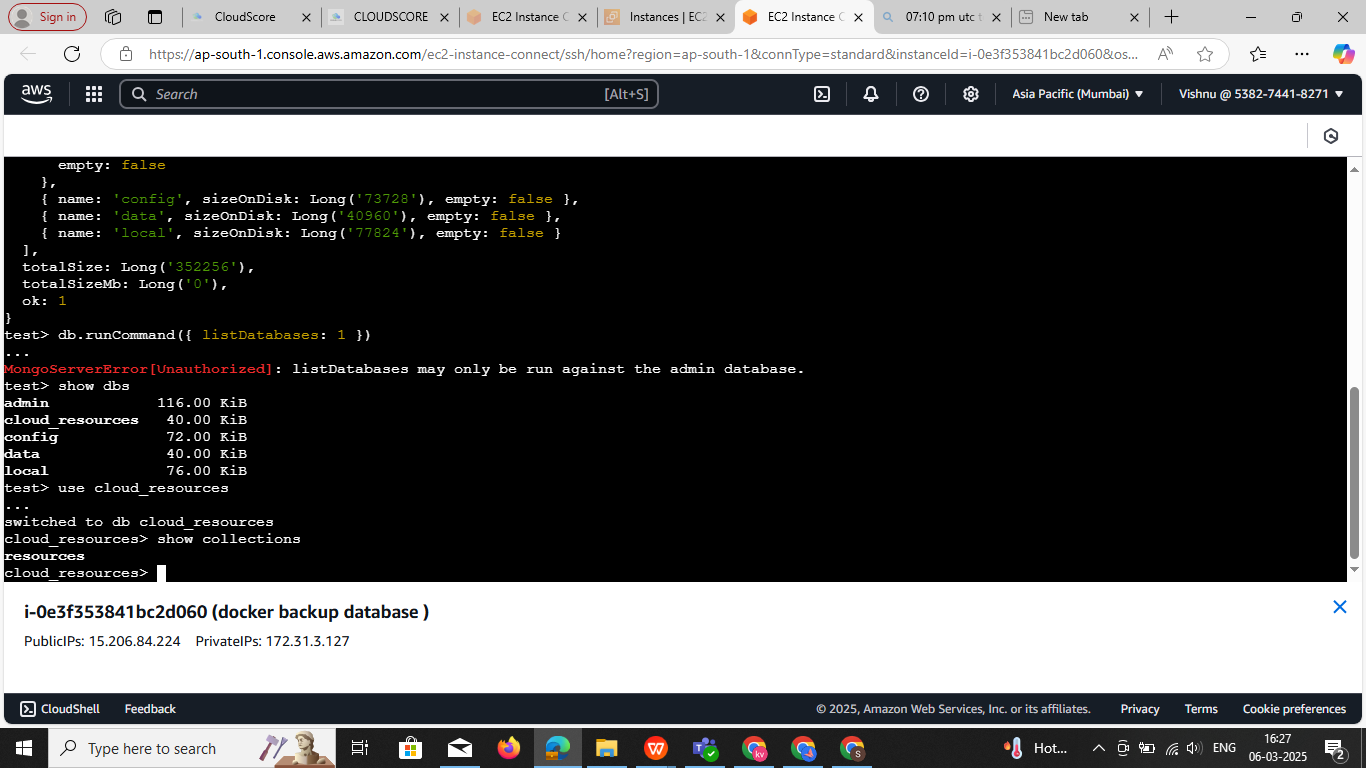
### ****3️⃣ Show Collections (List Tables)****

#### ****Command:****

show collections

#### ****Explanation:****

* Lists **all collections** (equivalent to tables in SQL) inside the current database.



**step4:** 1commands is used in **MongoDB** to retrieve and display all documents from the servers collection in a more readable format.

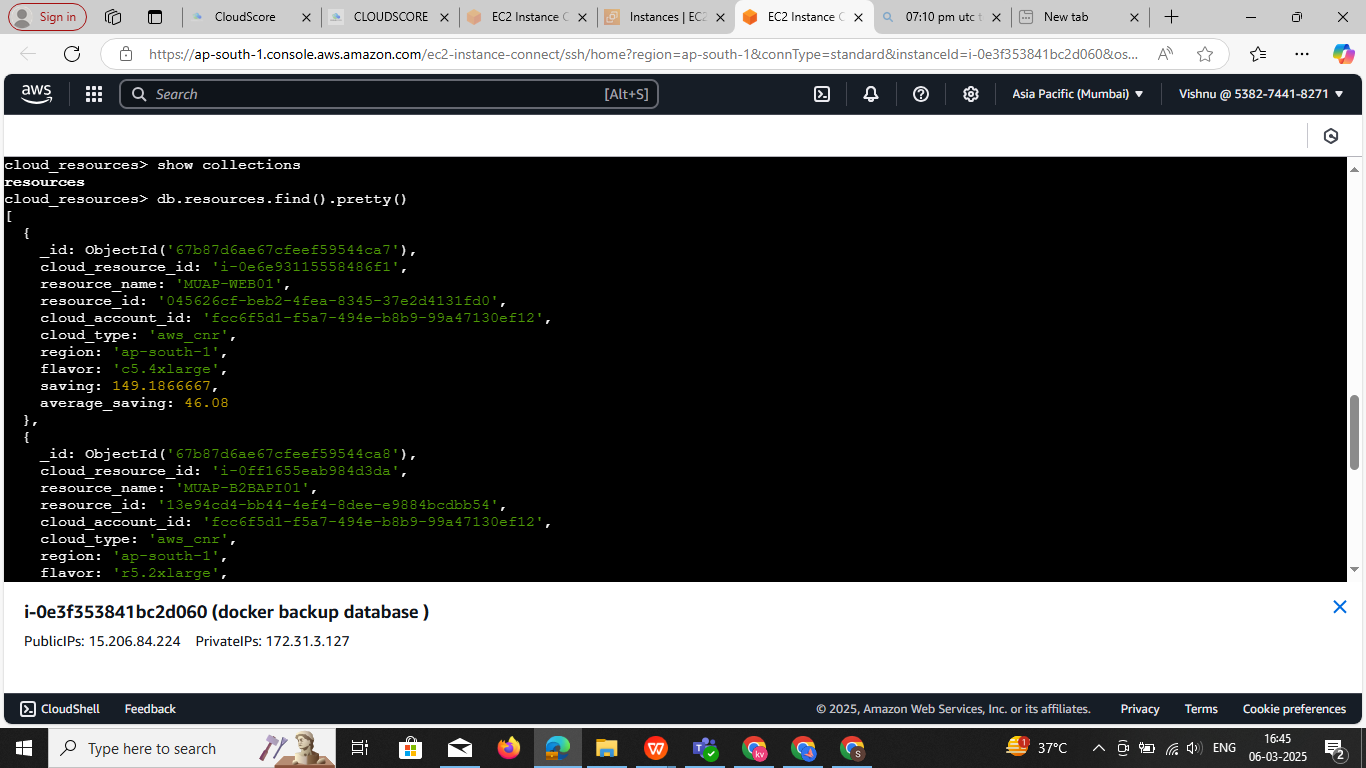
1 db.servers.find().pretty()   
 2 db.getMongo().getDBs() This will also show **empty databases** with an "empty": true field.

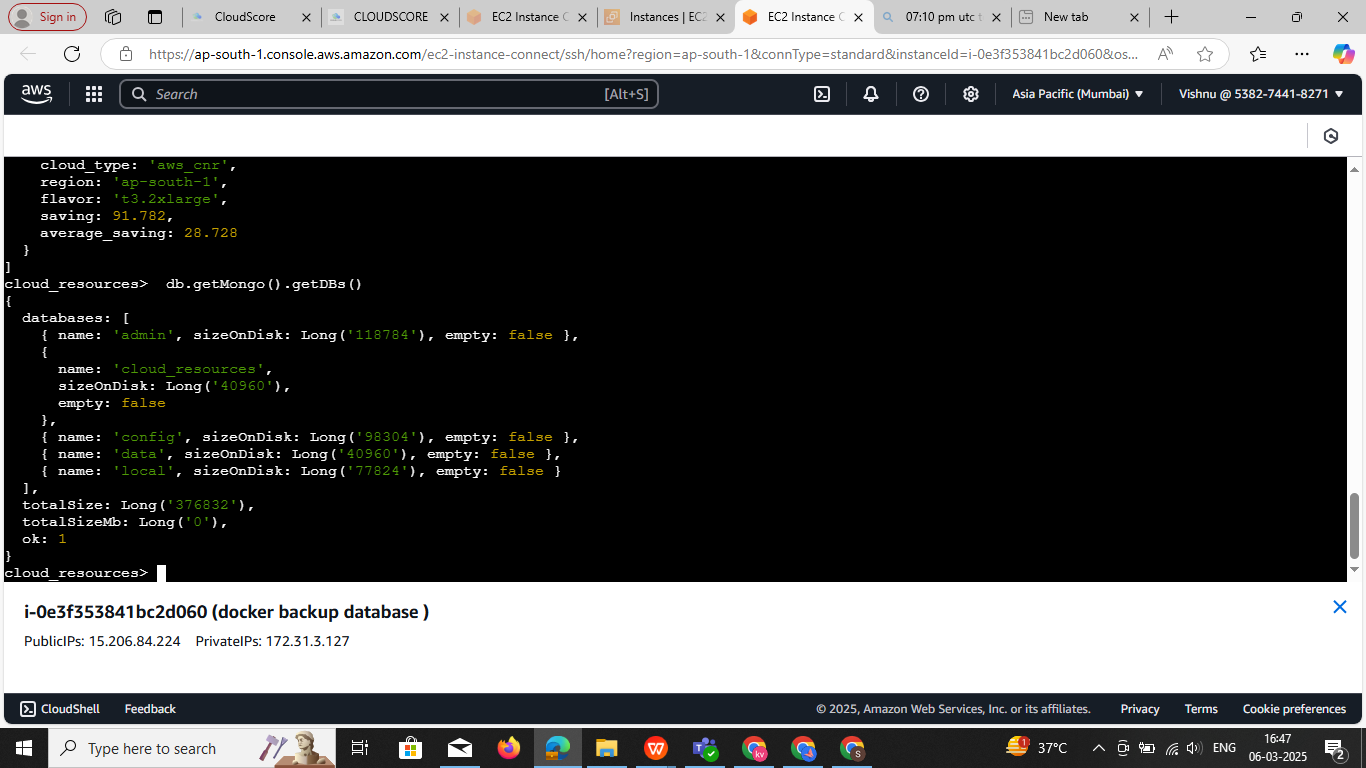
### Breakdown of the command:

**db.servers.find()**

* db refers to the current database.
* servers is the collection from which data is being retrieved.
* find() retrieves all documents from the collection (similar to SELECT \* FROM servers in SQL).

**.pretty()**Formats the output to make it more human-readable (with indentation and line breaks). Without .pretty(), MongoDB returns the result in a compact JSON format, which can be harder to read.



**db.getMongo().getDBs()**  This will also show **empty databases** with an "empty": true field.

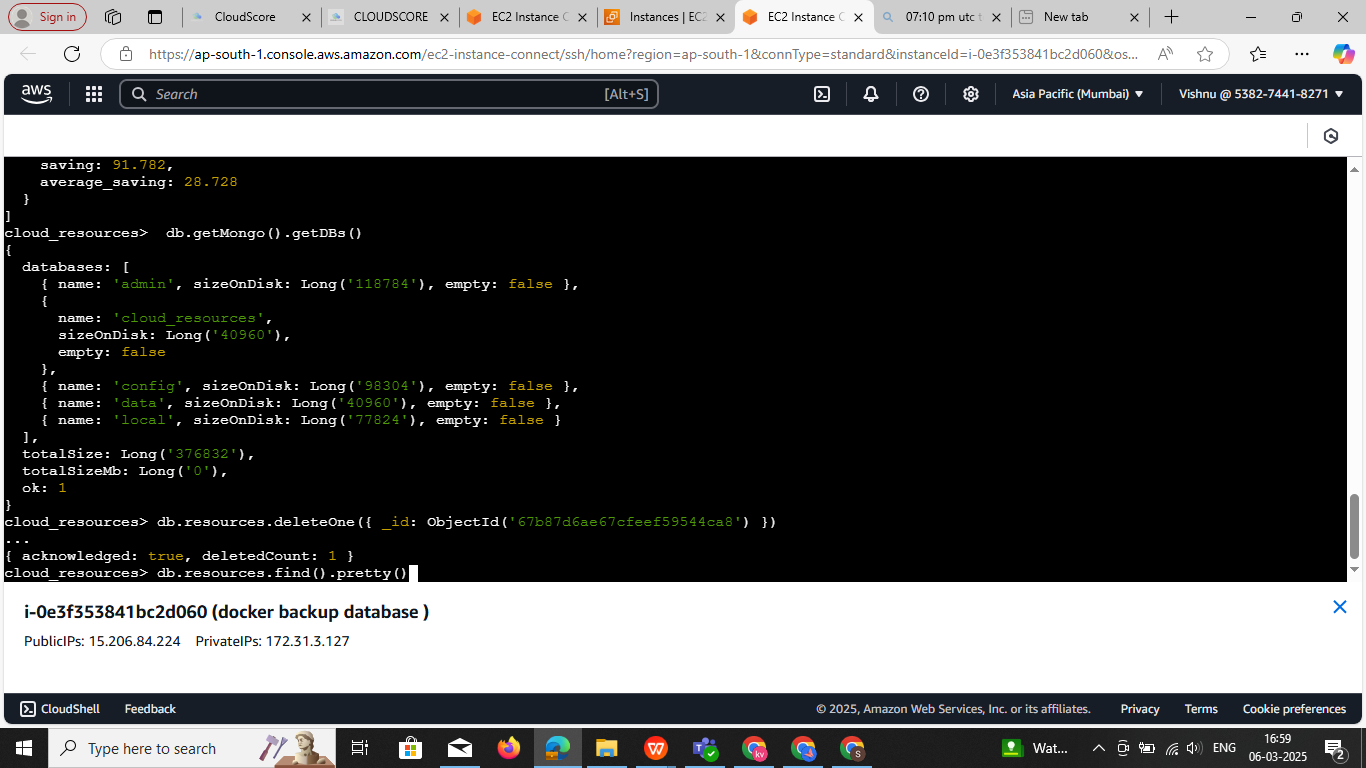
### Step5: ****Deleting a Single Record in MongoDB & Multiple record****

#### ****1️⃣ Delete a Record by**** \_id

To delete a specific document from the resources collection, use the \_id field.

**Example:** Delete the record with \_id: ObjectId('67b87d6ae67cfeef59544ca8')

db.resources.deleteOne({ \_id: ObjectId('67b87d6ae67cfeef59544ca8') })



### ****Delete All Documents from the**** resources ****Collection****

To delete all records in the resources collection **without deleting the collection itself**, use:

db.resources.deleteMany({}) This removes **all documents** but keeps the collection structure.

### ****Drop the Entire Collection (If Needed)****If you also want to delete the **collection itself**, use:

db.resources.drop() This **completely removes** the resources collection, including its indexes.

### ****Important Notes****

* deleteMany({}) **only deletes the documents**, but the collection remains.
* drop() **deletes the collection entirely**, so you'll have to recreate it if needed.

**BACKUP AND RESTORE script & manual back and resore**

#!/bin/bash

# Configuration

CONTAINER\_NAME="mongodb"

USERNAME="root"

PASSWORD="rootpassword"

AUTH\_DB="admin"

MONGO\_HOST="localhost"

MONGO\_PORT="27017"

# Backup retention (days)

RETENTION\_DAYS=7

# Email configuration (Set your email for alerts)

EMAIL="your\_email@example.com"

# Get date and time dynamically

TODAY=$(date +"%d-%m-%Y")

TIMESTAMP=$(date +"%H%M%S")

# Backup directory

BACKUP\_DIR="/root/pod/$TODAY"

# Log file

LOG\_FILE="/var/log/mongo\_backup.log"

# Ensure backup directory exists

mkdir -p "$BACKUP\_DIR"

log\_and\_alert() {

    echo "$(date '+%Y-%m-%d %H:%M:%S') - $1" | tee -a "$LOG\_FILE"

    echo "$1" | mail -s "MongoDB Backup Alert" $EMAIL

}

full\_backup() {

    log\_and\_alert "🔹 Taking full backup..."

    BACKUP\_PATH="$BACKUP\_DIR/full-$TIMESTAMP"

    if ! docker exec $CONTAINER\_NAME mongodump --host=$MONGO\_HOST --port=$MONGO\_PORT --username=$USERNAME --password=$PASSWORD --authenticationDatabase=$AUTH\_DB --out=/data/mongodump; then

        log\_and\_alert "❌ Full backup failed!"

        exit 1

    fi

    docker cp $CONTAINER\_NAME:/data/mongodump $BACKUP\_PATH

    log\_and\_alert "✅ Full backup saved at: $BACKUP\_PATH"

}

incremental\_backup() {

    log\_and\_alert "🔹 Taking incremental backup (Oplog)..."

    BACKUP\_PATH="$BACKUP\_DIR/oplog-$TIMESTAMP"

    if ! docker exec $CONTAINER\_NAME mongodump --username $USERNAME --password $PASSWORD -d local -c oplog.rs --out /data/oplog; then

        log\_and\_alert "❌ Incremental backup failed!"

        exit 1

    fi

    docker cp $CONTAINER\_NAME:/data/oplog $BACKUP\_PATH

    log\_and\_alert "✅ Incremental backup (oplog) saved at: $BACKUP\_PATH"

}

restore\_backup() {

    log\_and\_alert "🔹 Restoring full backup..."

    read -p "Enter full backup path: " FULL\_PATH

    read -p "Enter oplog backup path (optional, press Enter to skip): " OPLOG\_PATH

    if [ ! -d "$FULL\_PATH" ]; then

        log\_and\_alert "❌ Error: Full backup path does not exist!"

        exit 1

    fi

    docker cp "$FULL\_PATH" $CONTAINER\_NAME:/data/mongodump

    docker exec $CONTAINER\_NAME mongorestore --host=$MONGO\_HOST --port=$MONGO\_PORT -u $USERNAME -p $PASSWORD --authenticationDatabase=$AUTH\_DB --dir=/data/mongodump --drop

    if [ -n "$OPLOG\_PATH" ] && [ -d "$OPLOG\_PATH" ]; then

        docker cp "$OPLOG\_PATH" $CONTAINER\_NAME:/data/oplog

        docker exec $CONTAINER\_NAME mongorestore --oplogReplay /data/oplog

    fi

    log\_and\_alert "✅ Restore completed!"

}

cleanup\_old\_backups() {

    log\_and\_alert "🔹 Deleting backups older than $RETENTION\_DAYS days..."

    find /root/pod/ -type d -mtime +$RETENTION\_DAYS -exec rm -rf {} \;

    log\_and\_alert "✅ Old backups deleted!"

}

case "$1" in

    full) full\_backup ;;

    incr) incremental\_backup ;;

    restore) restore\_backup ;;

    cleanup) cleanup\_old\_backups ;;

    \*) echo "Usage: $0 {full|incr|restore|cleanup}" ;;

esac

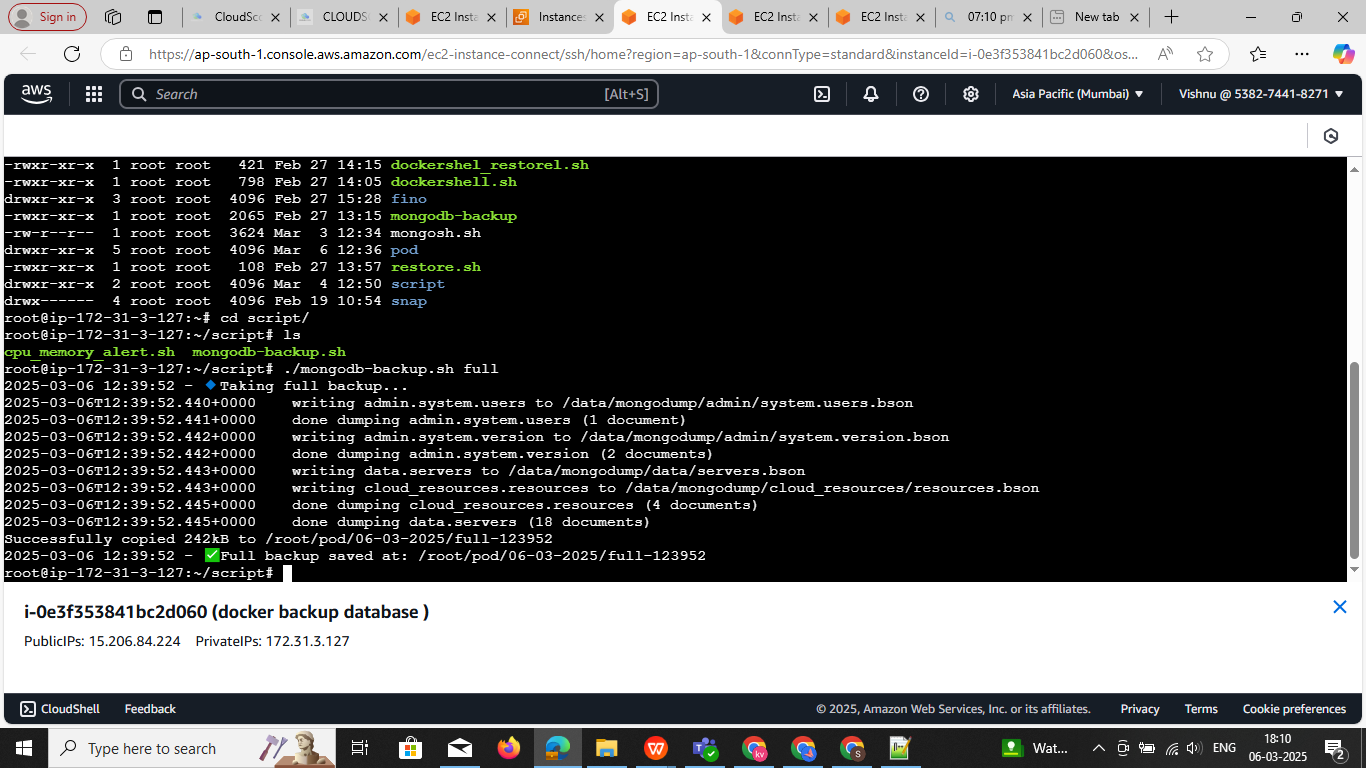
./mongodb-backup.sh full

./mongodb-backup.sh incr

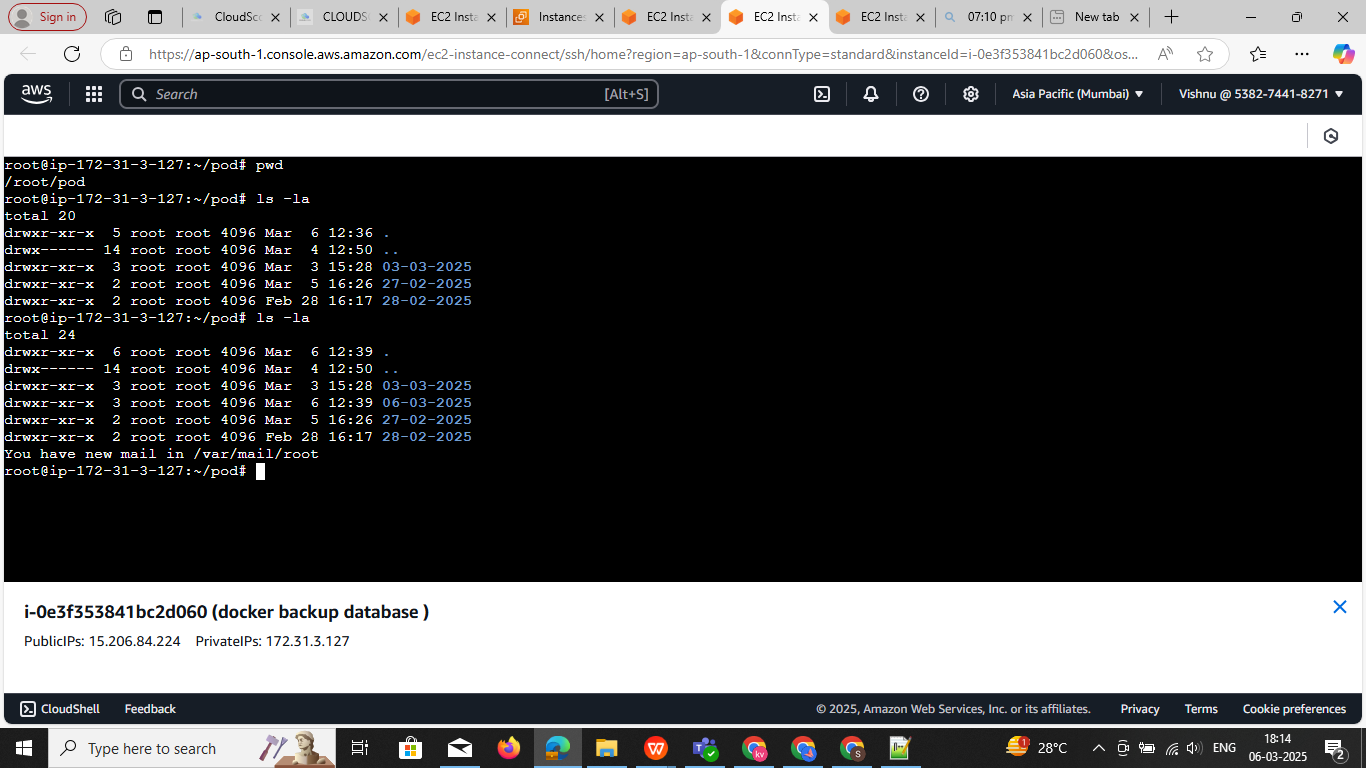
./mongodb-backup.sh restore

./mongodb-backup.sh cleanup

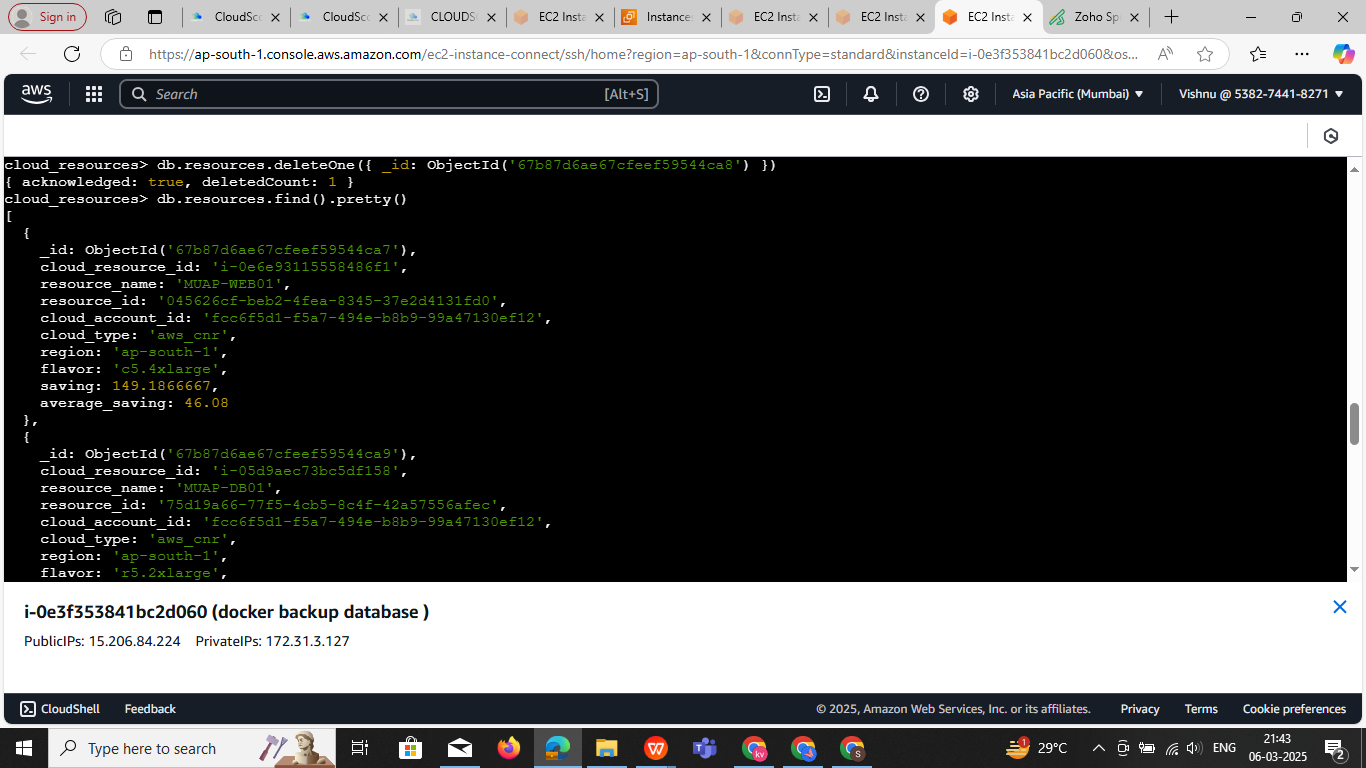
**Step5.1:** login to ec2 sudo - i --> change dir --> ./mongodb-backup.sh full

C

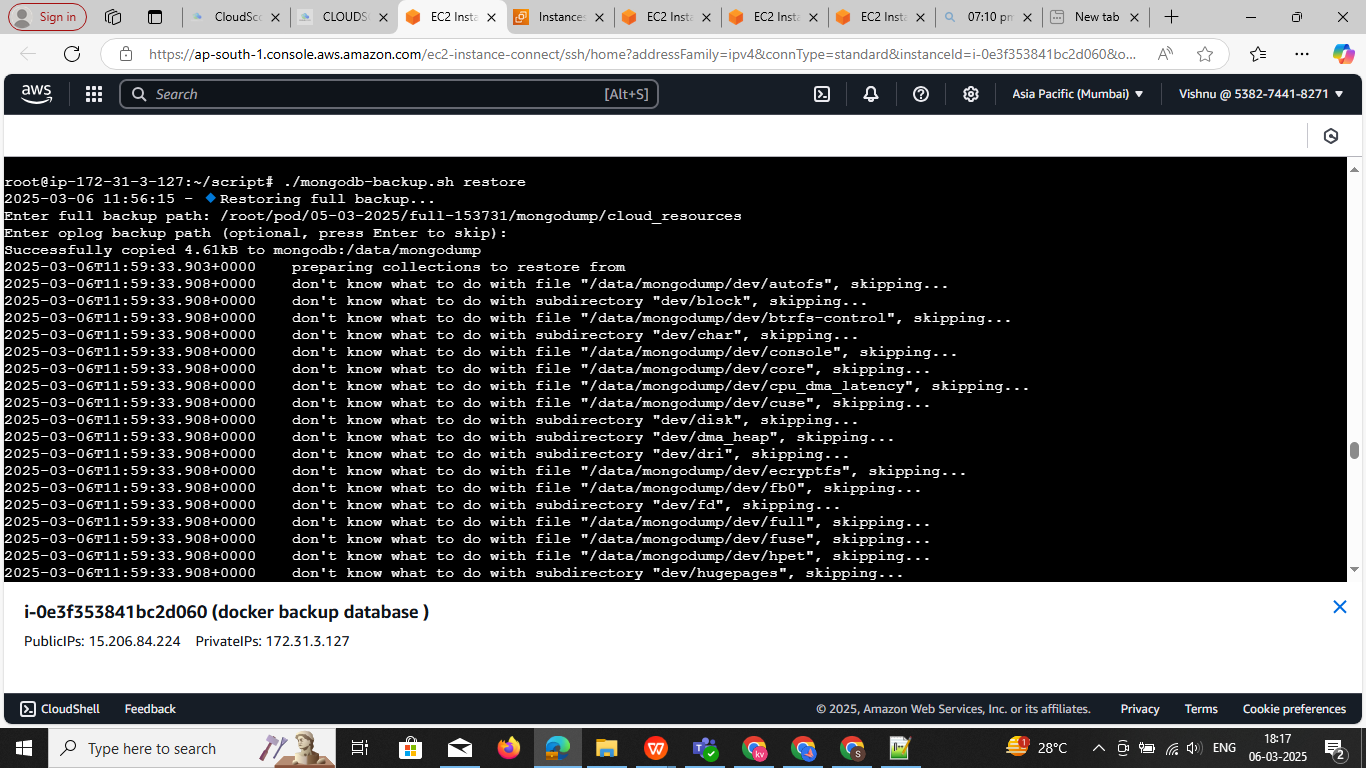
* check the dir is dump data is available are not

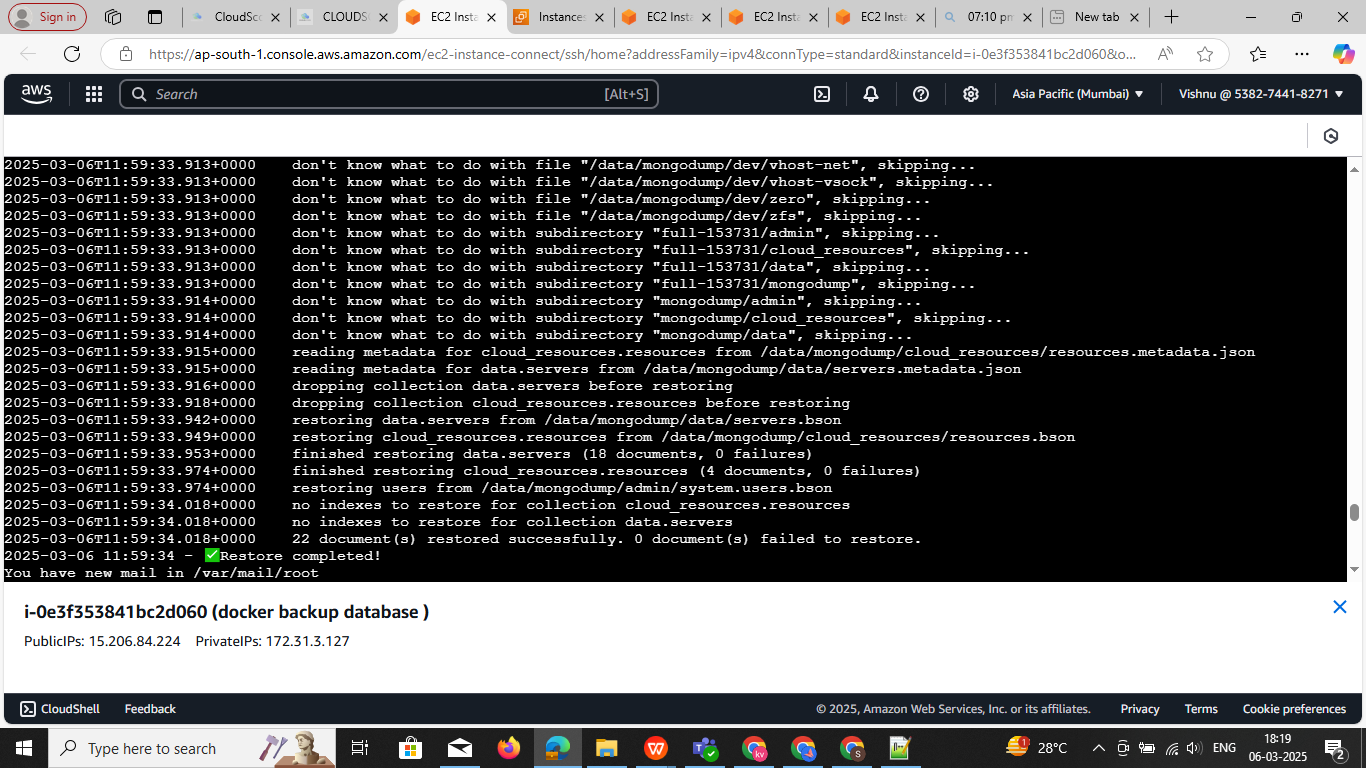


Delete the one of id I have 4 tables one of then delete it   
db.resources.deleteOne({ \_id: ObjectId('67b87d6ae67cfeef59544ca8') })



**step5.2** ./mongodb-backup.sh restore Now we need to restore the data

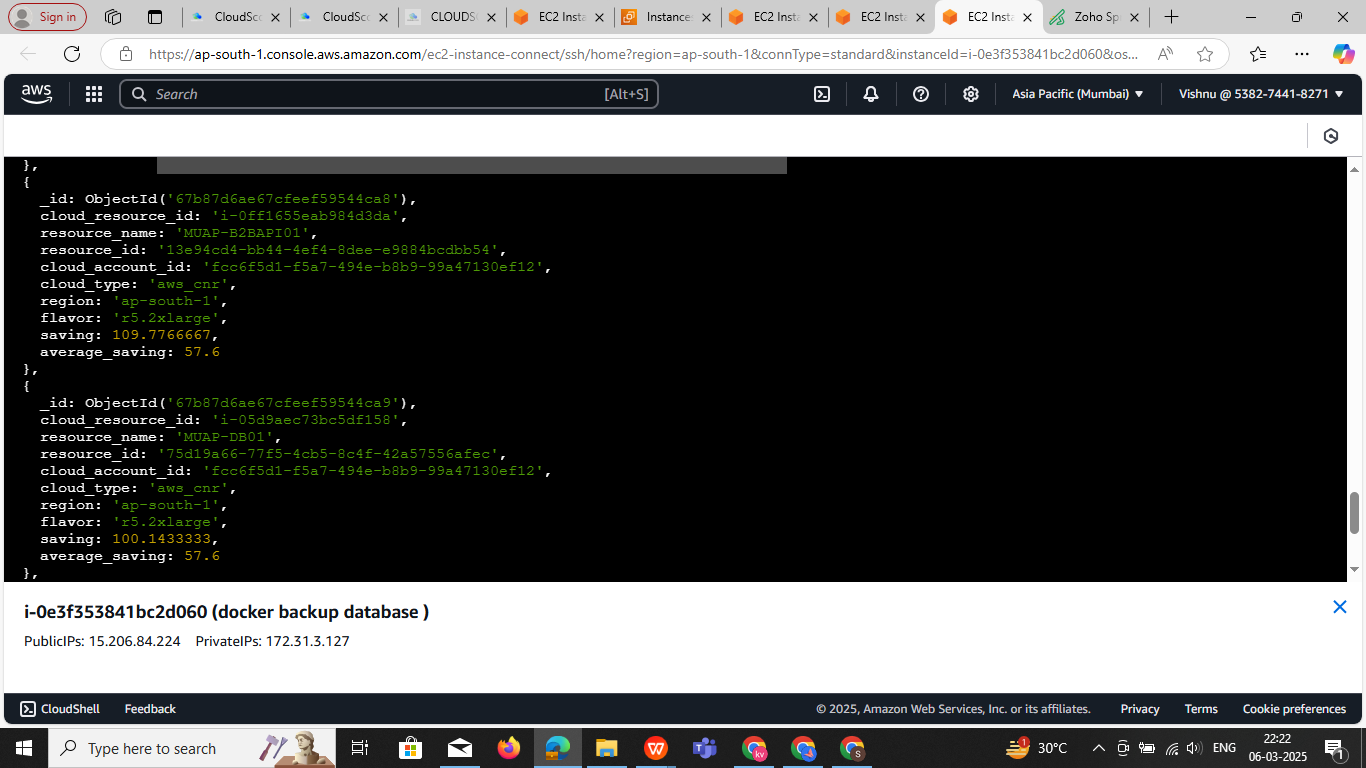




**Note:** now we need to check the database output how many restore

**Sep6:** now we need the check all To see the data inside your collection: I deleted recode db.resources.deleteOne({ \_id: ObjectId('67b87d6ae67cfeef59544ca8') }) to check the collection is restore

**db.servers.find().pretty()**



**Install the all Database set up**

**requirements t3x.large**

**30 gb storgare select AMI :Ubuntu**

**vi dockerdatabase-compose.yml**

Docker file

version: '3.8'

services:

mysql:

image: mysql:latest

restart: always

environment:

MYSQL\_ROOT\_PASSWORD: root

MYSQL\_DATABASE: mydb

MYSQL\_USER: myuser

MYSQL\_PASSWORD: mypassword

ports:

- "3306:3306"

mongodb:

image: mongo:latest

restart: always

environment:

MONGO\_INITDB\_ROOT\_USERNAME: admin

MONGO\_INITDB\_ROOT\_PASSWORD: admin

ports:

- "27017:27017"

influxdb:

image: influxdb:latest

restart: always

environment:

INFLUXDB\_ADMIN\_USER: admin

INFLUXDB\_ADMIN\_PASSWORD: admin

ports:

- "8086:8086"

minio:

image: minio/minio

restart: always

environment:

MINIO\_ROOT\_USER: admin

MINIO\_ROOT\_PASSWORD: admin123

ports:

- "9000:9000"

- "9001:9001"

command: server /data --console-address ":9001"

clickhouse:

image: clickhouse/clickhouse-server:latest

restart: always

ports:

- "8123:8123"

- "9000:9000"

redis:

image: redis:latest

restart: always

ports:

- "6379:6379"

### ****Run Docker Compose****

Since your file is not named docker-compose.yml, you must specify it using the -f flag:

**docker-compose -f dockerdatabase-compose.yml up -d**

**Explanation:**

* -f dockerdatabase-compose.yml → Specifies the file to use.
* up → Starts the services.
* -d → Runs in **detached mode** (in the background).