Complete Project Report: AWS Database Migration using Terraform

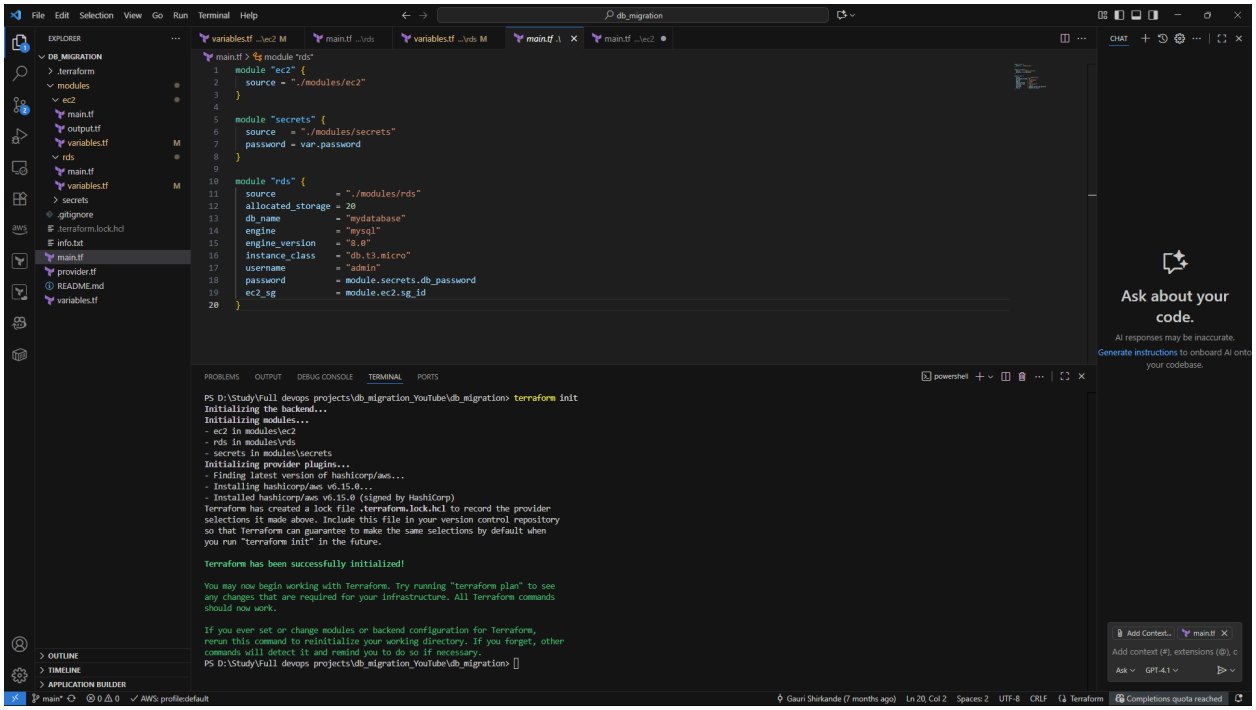
Project Goal: To provision a cloud infrastructure environment using Infrastructure as Code (Terraform) and execute a successful logical database migration from a self-managed EC2 instance (Source) to a managed AWS RDS instance (Target).

1. Infrastructure as Code (IaC) Setup and Deployment

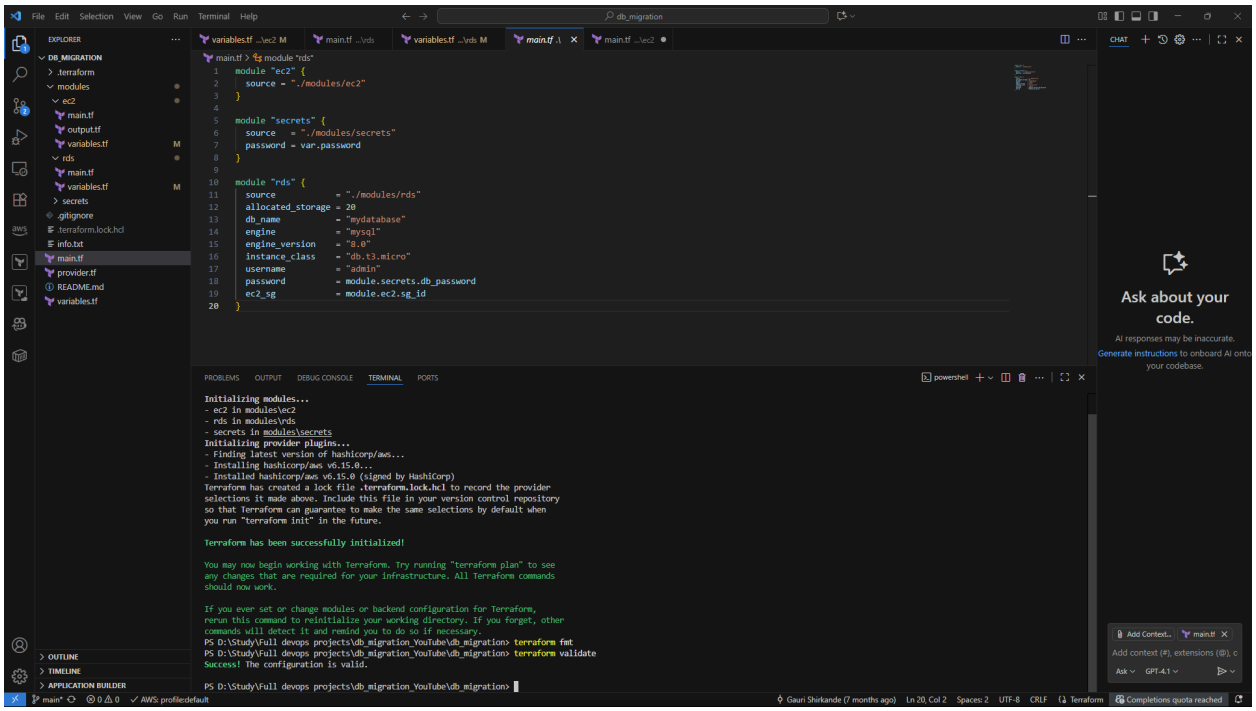
This phase focused on defining and deploying the required AWS resources—an EC2 instance, an RDS MySQL database, and secure credential storage—using Terraform.

1.1 Project Initialization and Configuration Review

Step 1: Initialize the Terraform Backend and Modules
The project was initialized using terraform init to download the necessary AWS provider and modules, confirming the project's readiness for deployment.

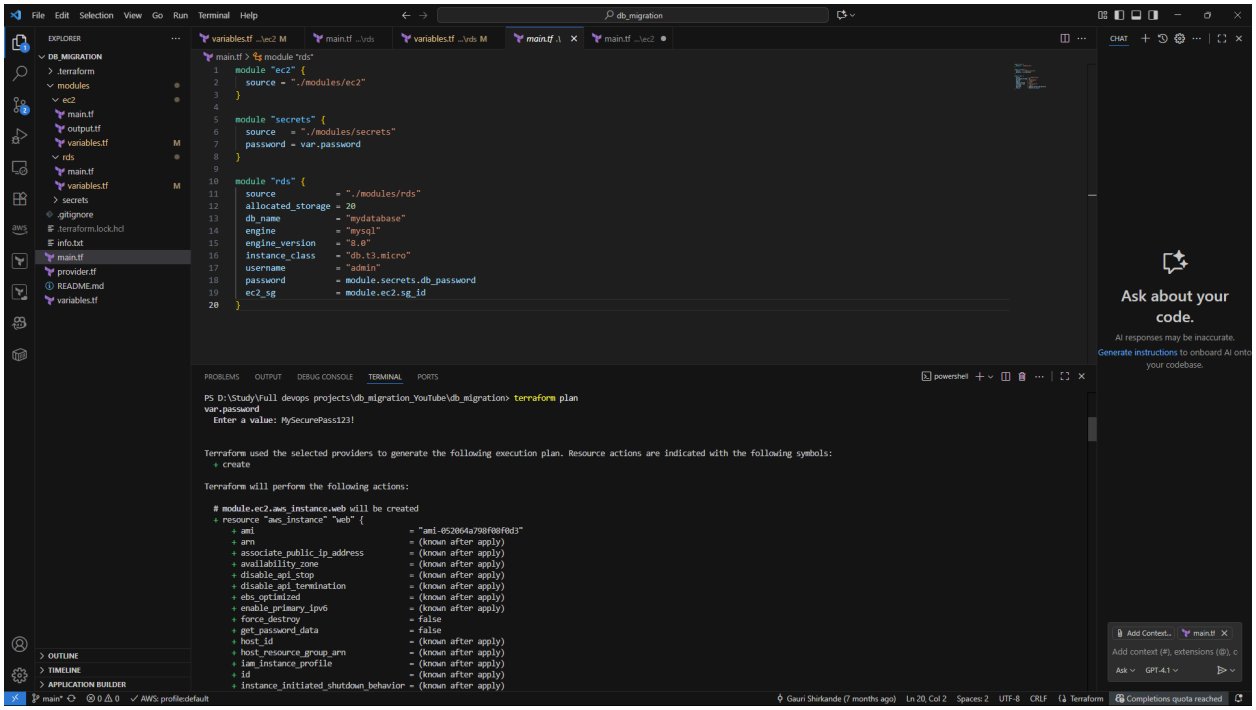


Step 2: Review the Core Terraform Configuration
The core configuration, defined in main.tf, was reviewed. It shows modular calls for the EC2 instance, RDS database, and Secrets Manager, ensuring a clean and reusable structure.

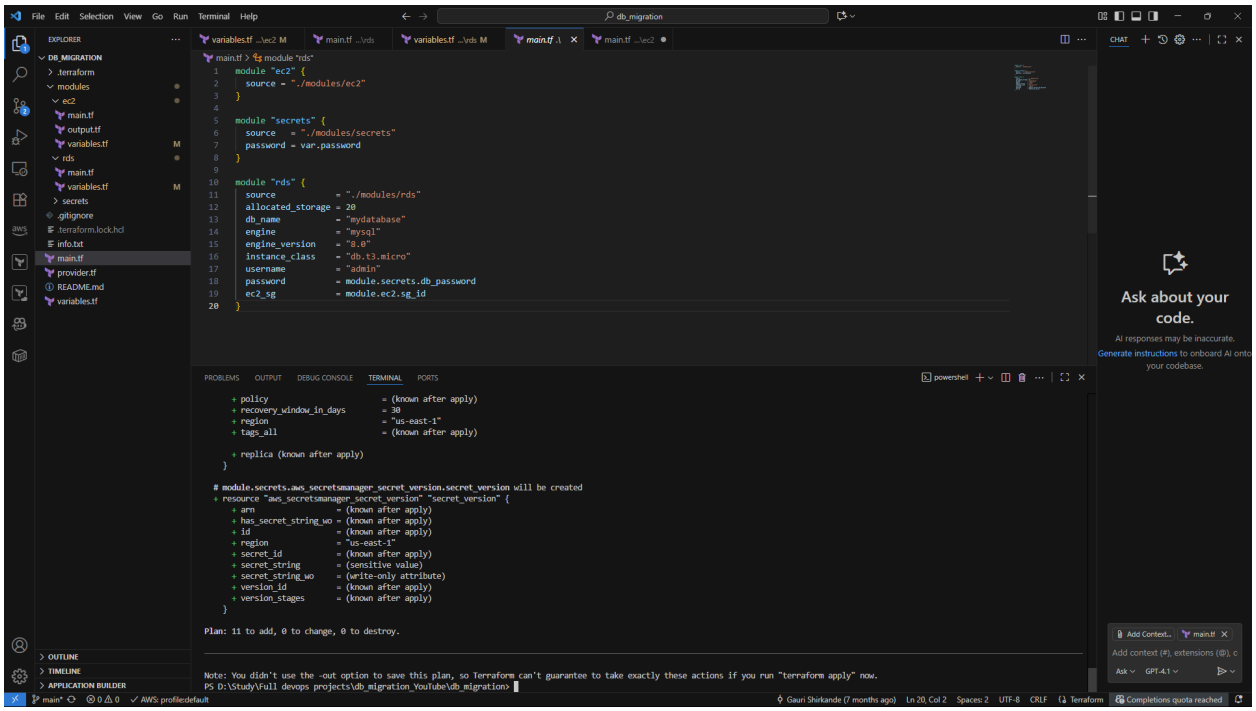


1.2 Deployment Planning and Execution

Step 3: Execute terraform plan (EC2 review)
The terraform plan command was executed. The output was reviewed to confirm the planned creation of the EC2 instance and its associated resources.

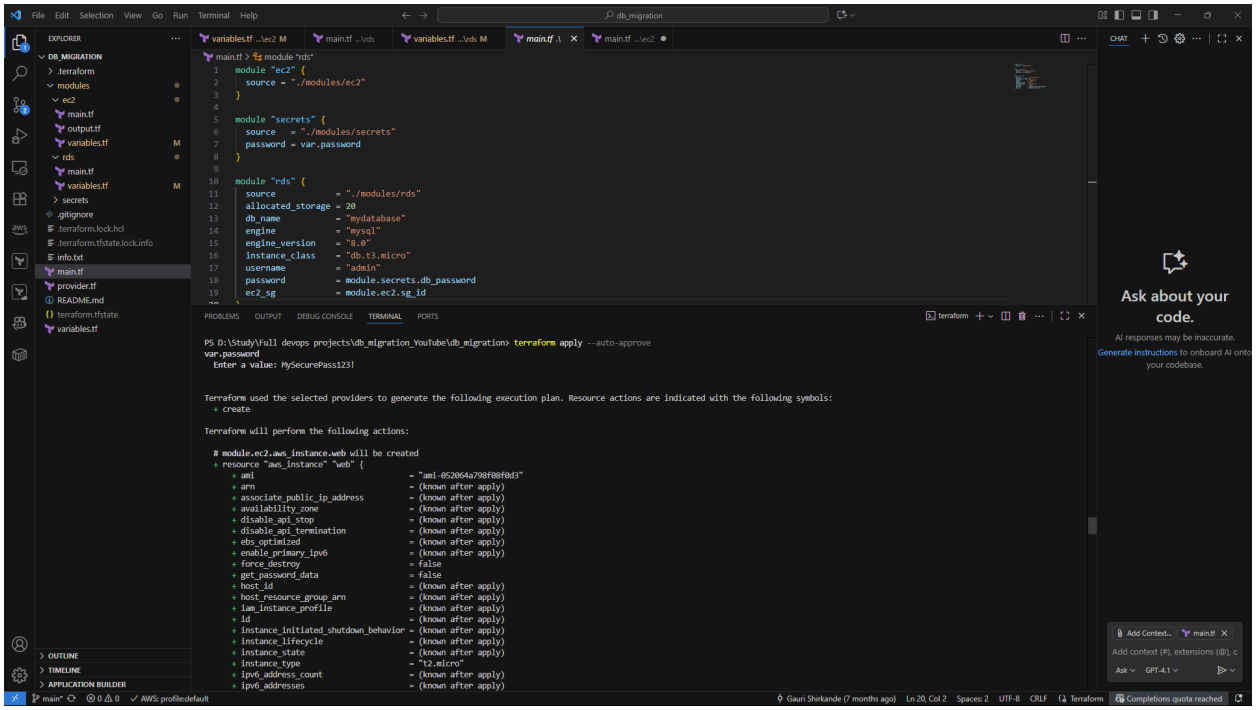


Step 4: Review the terraform plan (Secrets Manager review)
The plan output was further reviewed to confirm the planned creation of the AWS Secrets Manager resource, which securely stores the database credentials.



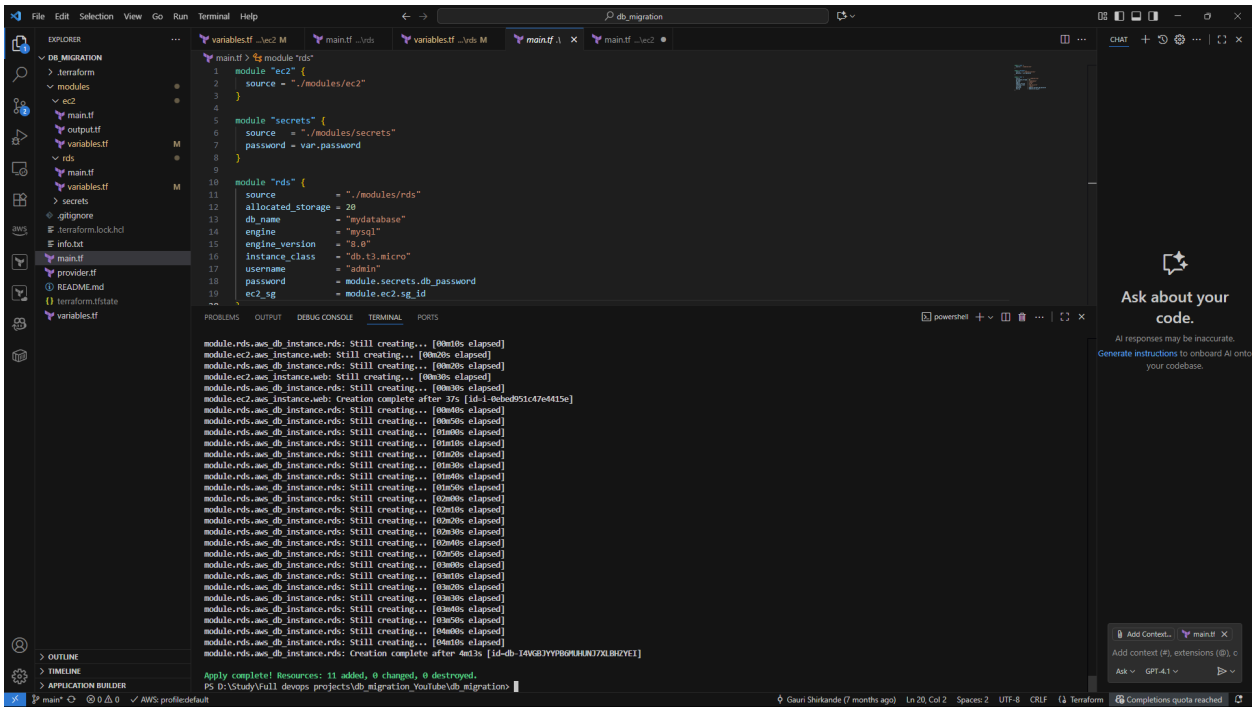
Step 5: Execute terraform apply

The terraform apply -auto-approve command was executed to start the resource provisioning in AWS, securely passing the required database password as a variable.



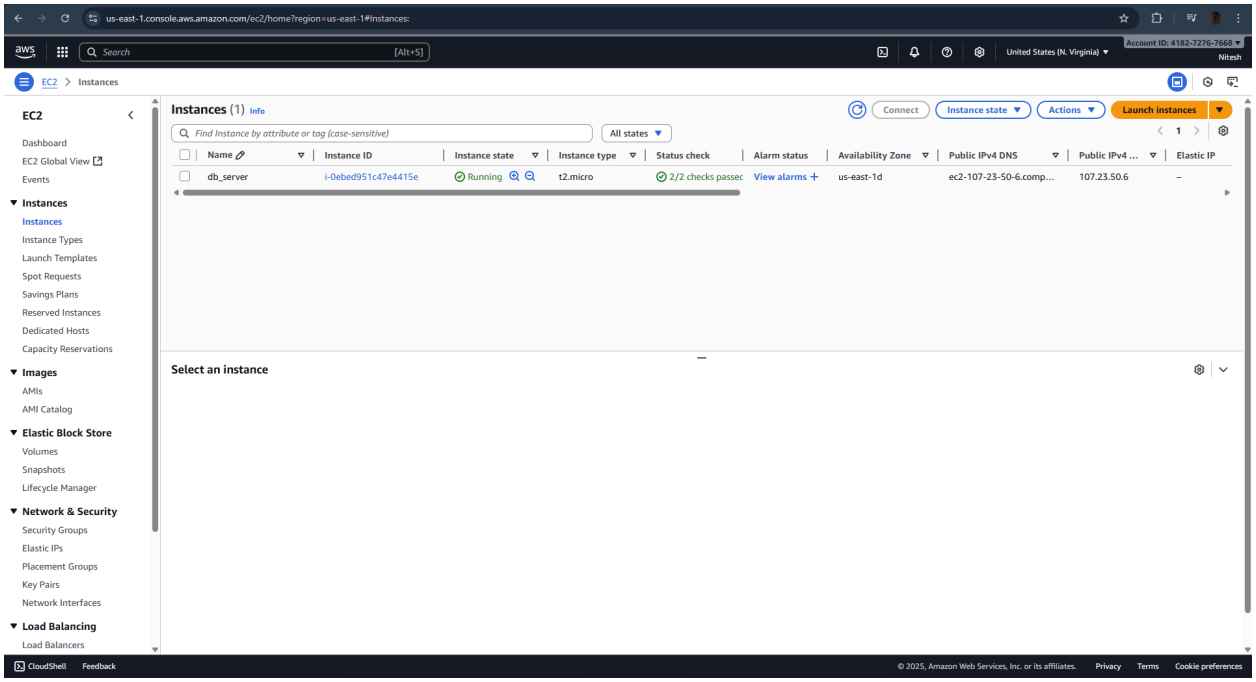
Step 6: Verify Deployment Completion

The console confirmed the successful completion of the deployment: "Apply complete! Resources: 11 added, 0 changed, 0 destroyed."

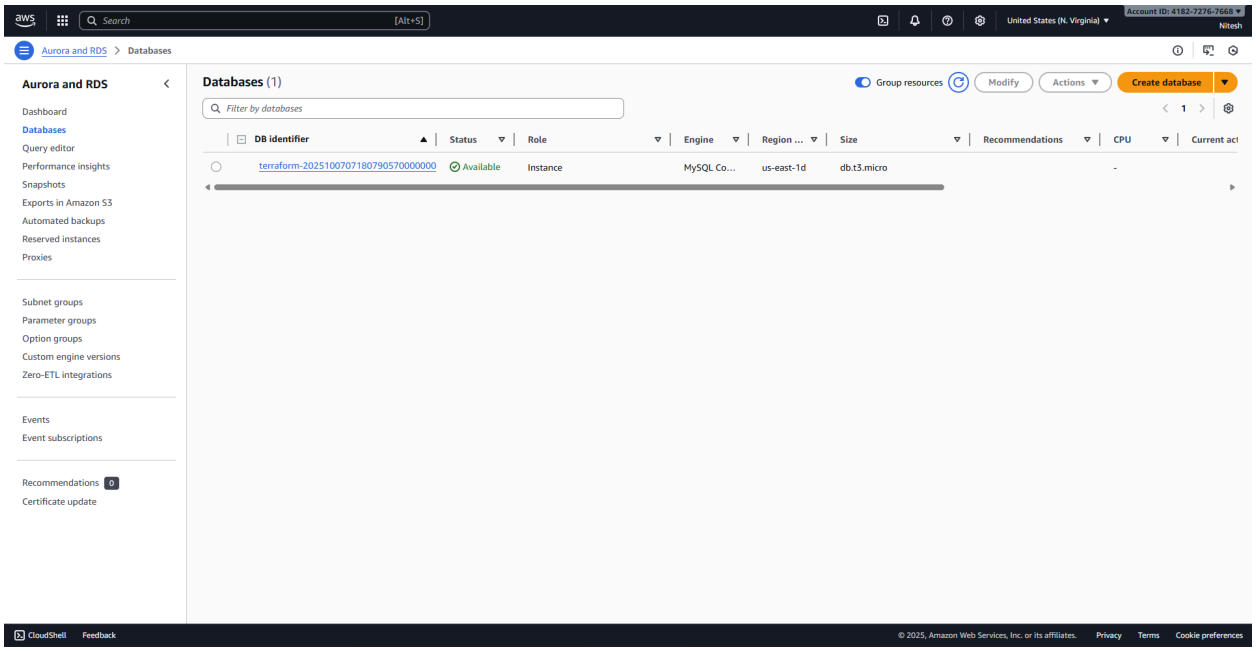


1.3 AWS Console Verification

Step 7: Verify the EC2 instance status
The AWS Management Console was checked to confirm that the EC2 instance ($db_server) was in the Running state.



Step 8: Verify the RDS MySQL instance status
The AWS Management Console was checked to confirm that the RDS MySQL instance was in the Available state.



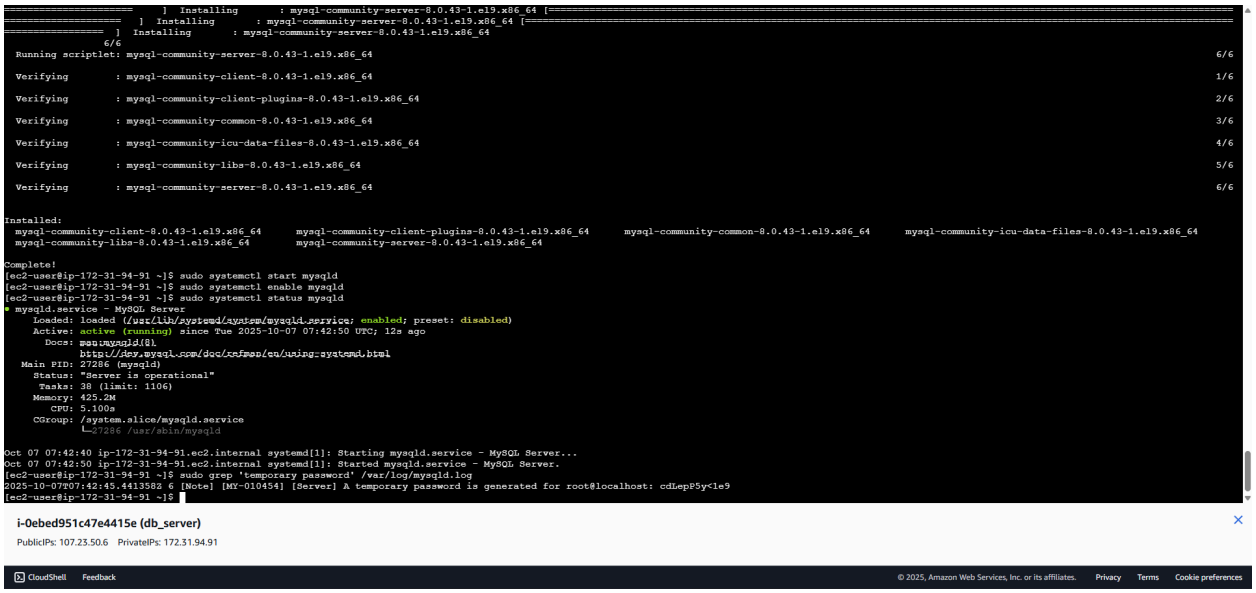
2. Source Database Preparation on EC2

The provisioned EC2 instance was prepared as the source database environment.

2.1 MySQL Installation and Configuration

Step 9: Install MySQL Server and Client
Commands were executed via SSH to download the MySQL repository and install the server and client components on the EC2 instance.

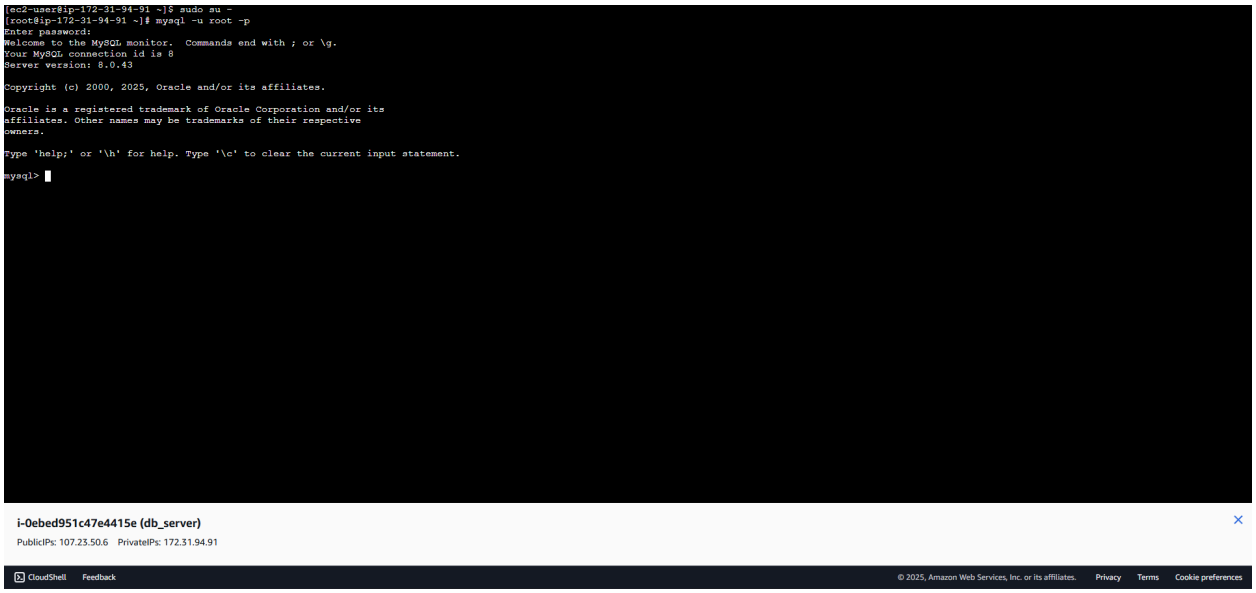
Step 10: Confirm Service Status and Retrieve Password
The mysqld service was confirmed to be running, and the initial temporary root password was retrieved from the MySQL log file.



2.2 Source Database Setup

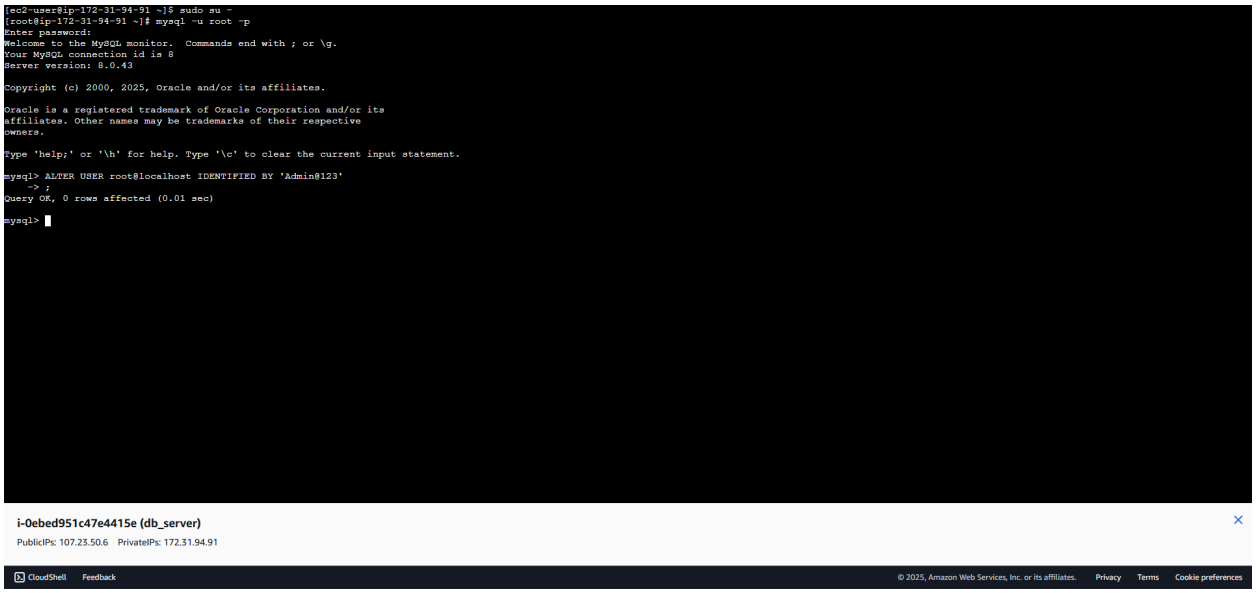
Step 11: Log into the MySQL Monitor

The temporary password was used to log into the MySQL monitor to begin the configuration process.



Step 12: Set a Secure Root Password

A secure, permanent password was set for the root user to secure the source database instance.

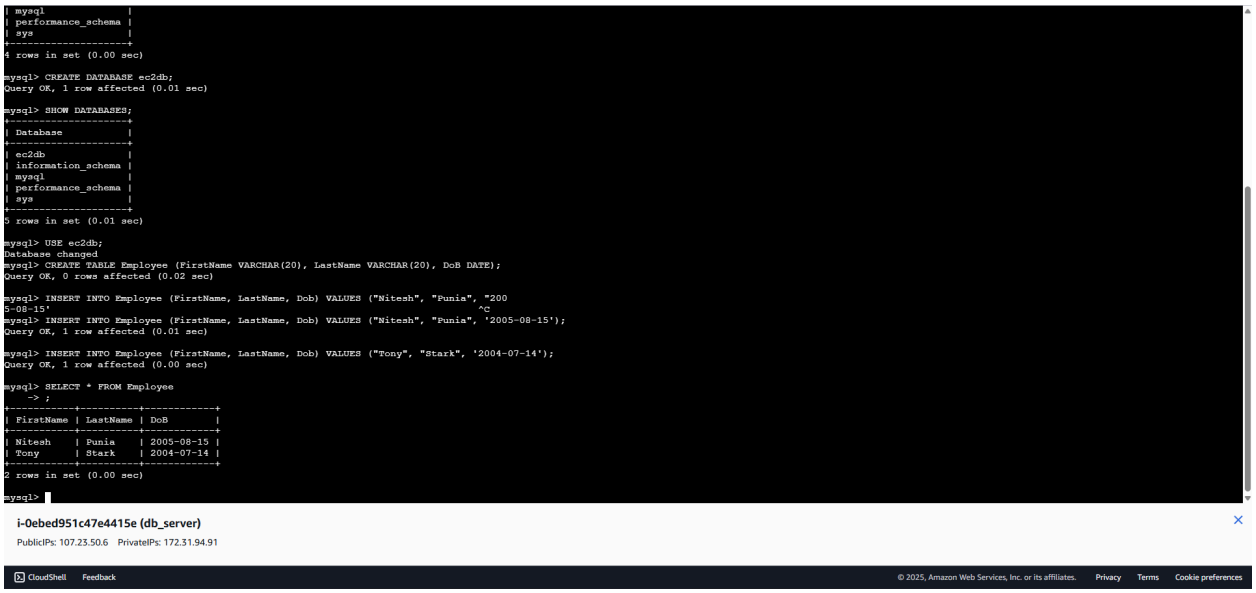


Step 13: Create the Source Database

The source database, named ec2db, was created and its existence was verified.

Step 14: Populate Test Data

A sample Employee table was created, and two test records were inserted and verified, creating the data payload for the migration.

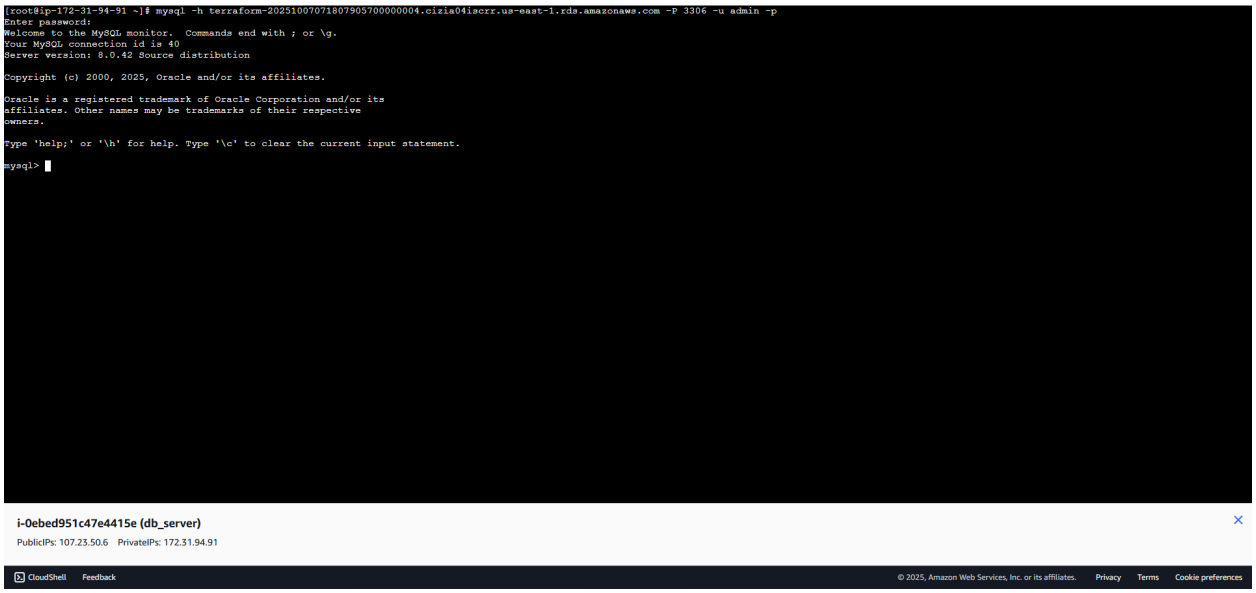


3. Database Migration and Verification

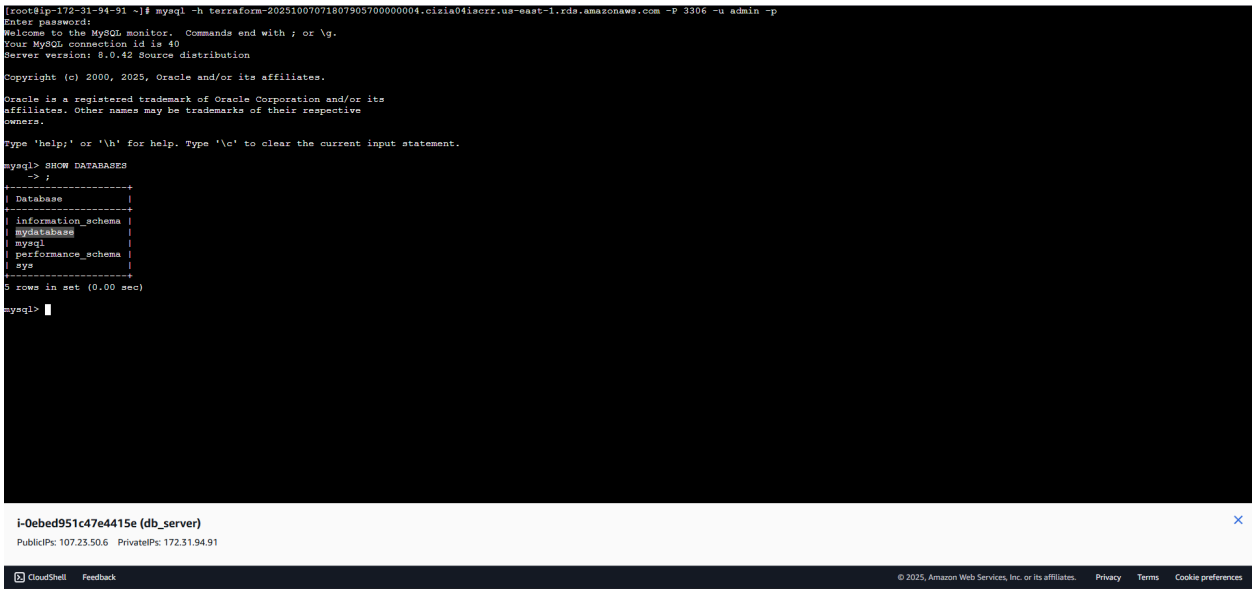
This phase involved data export from the source, import to the target, and validation.

3.1 Target Connectivity and Readiness Check

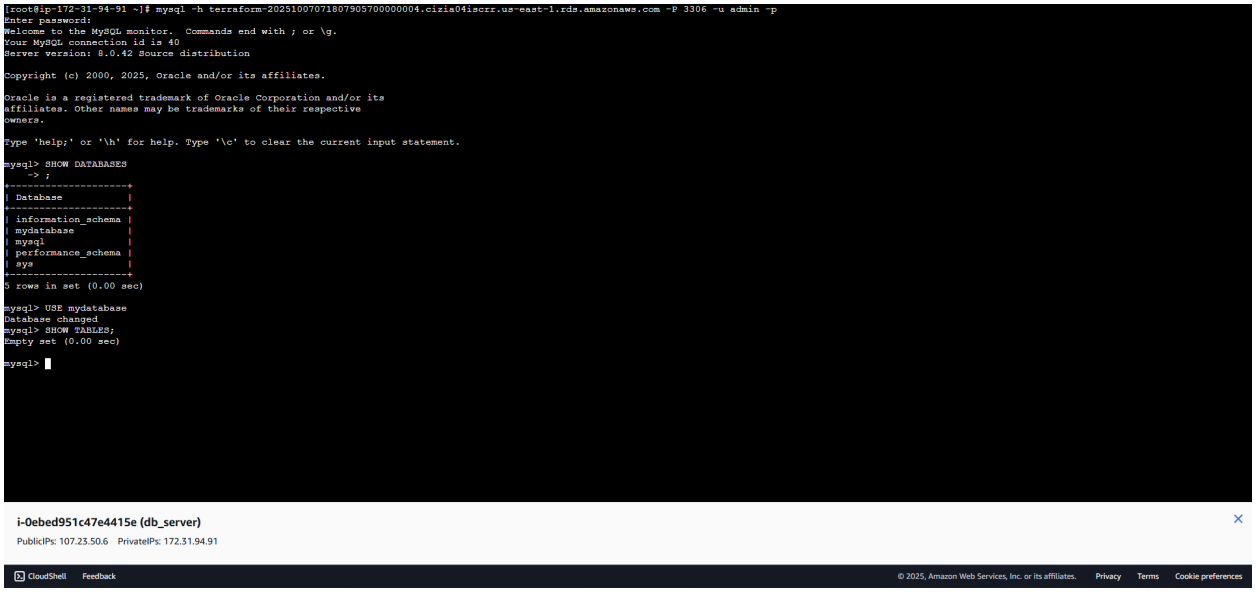
Step 15: Connect to the Target RDS Endpoint
A connection was successfully established from the EC2 source instance to the RDS target endpoint, confirming network security group rules were correct.



Step 16: Confirm Target Database Existence
The RDS databases were listed, confirming the presence of the Terraform-created target, mydbdatabase.

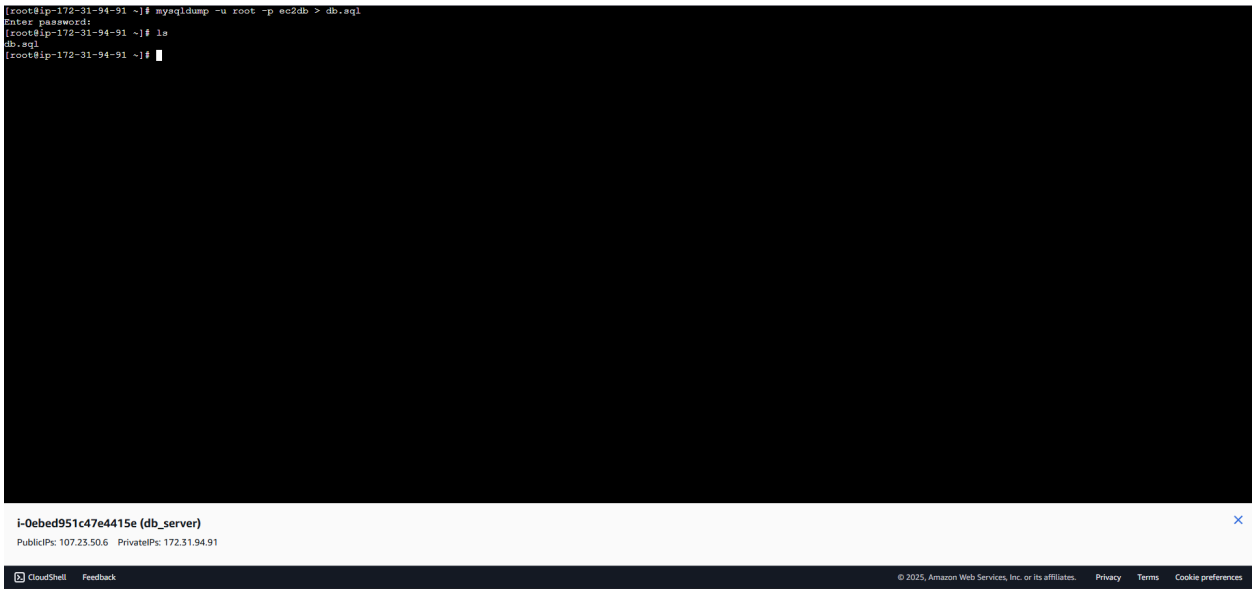


Step 17: Confirm Target Database is Empty
The target database was inspected (SHOW TABLES) to confirm it was empty prior to the migration process.

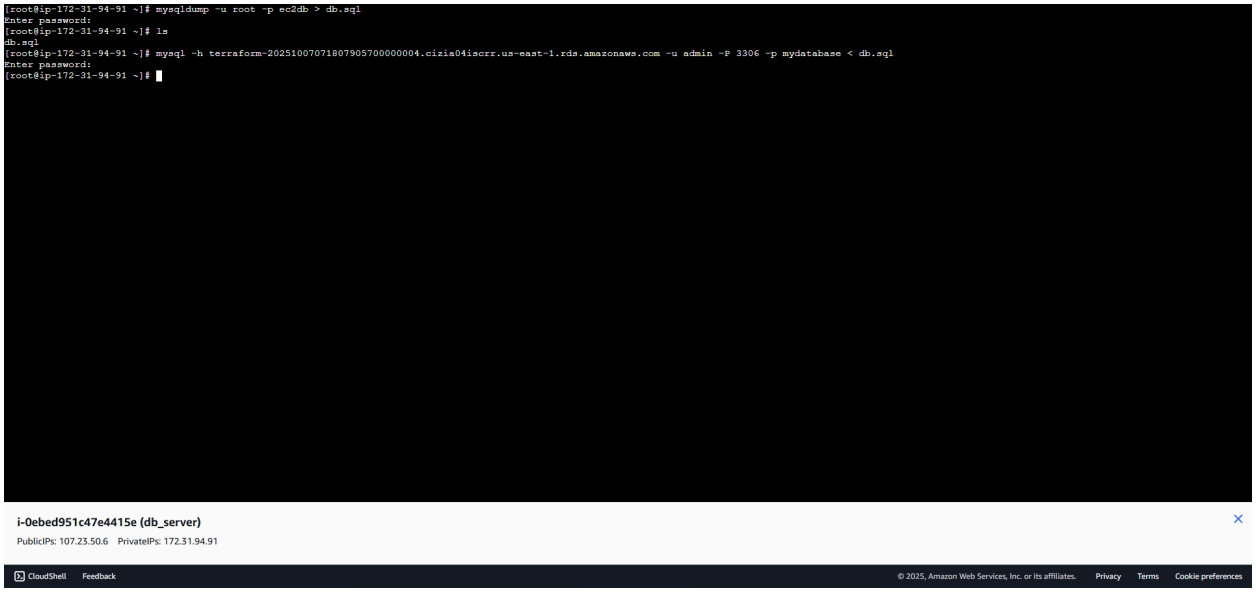


3.2 Data Export and Import

Step 18: Execute Data Dump (Export)
The mysqldump utility was used to export the entire ec2db schema and data into a file named db.sql on the EC2 file system.

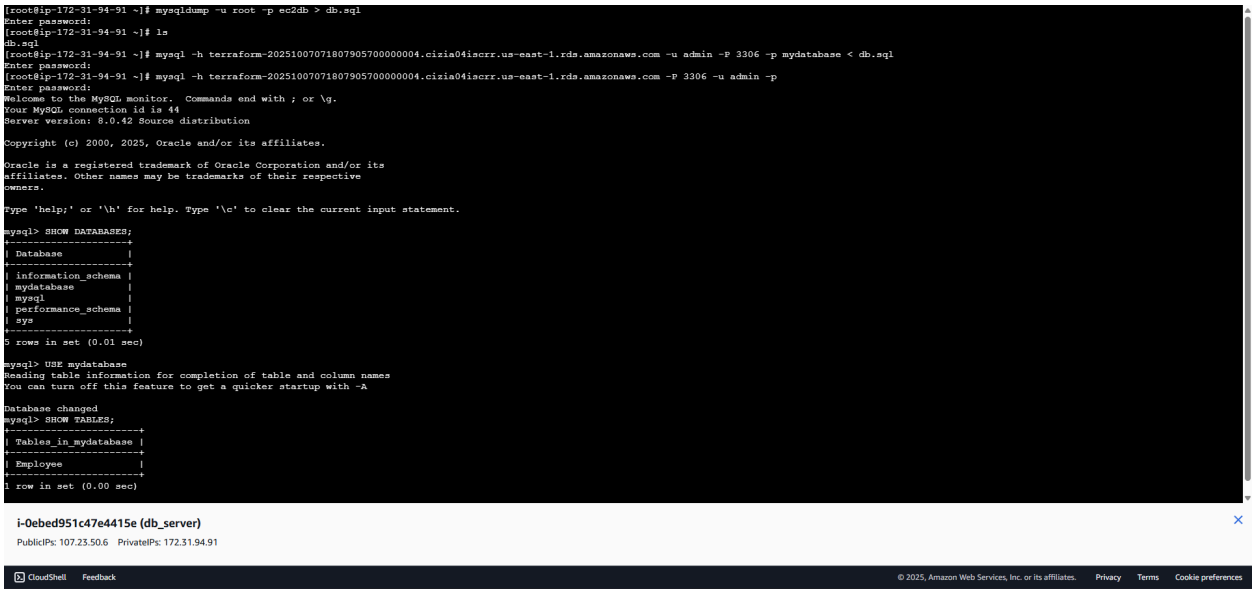


Step 19: Execute Data Load (Import)
The MySQL client was used to import the db.sql file into the target RDS database (mydbdatabase), completing the logical migration.

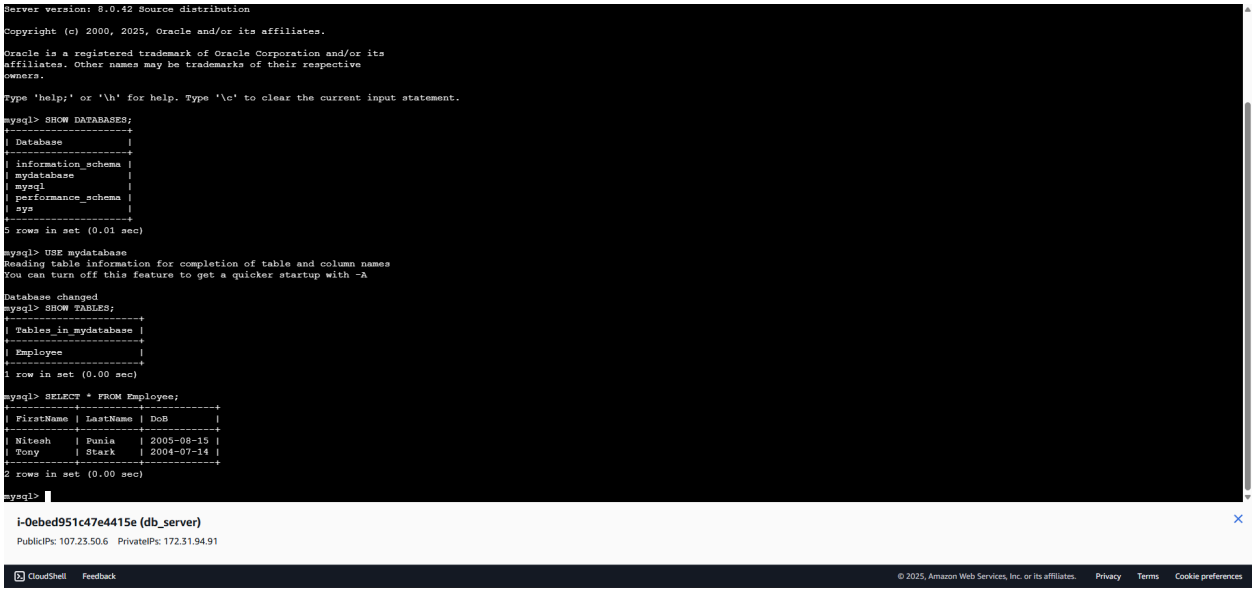


3.3 Post-Migration Validation

Step 20: Verify Table Existence in Target RDS
The target RDS database was checked, confirming the successful creation of the Employee table.



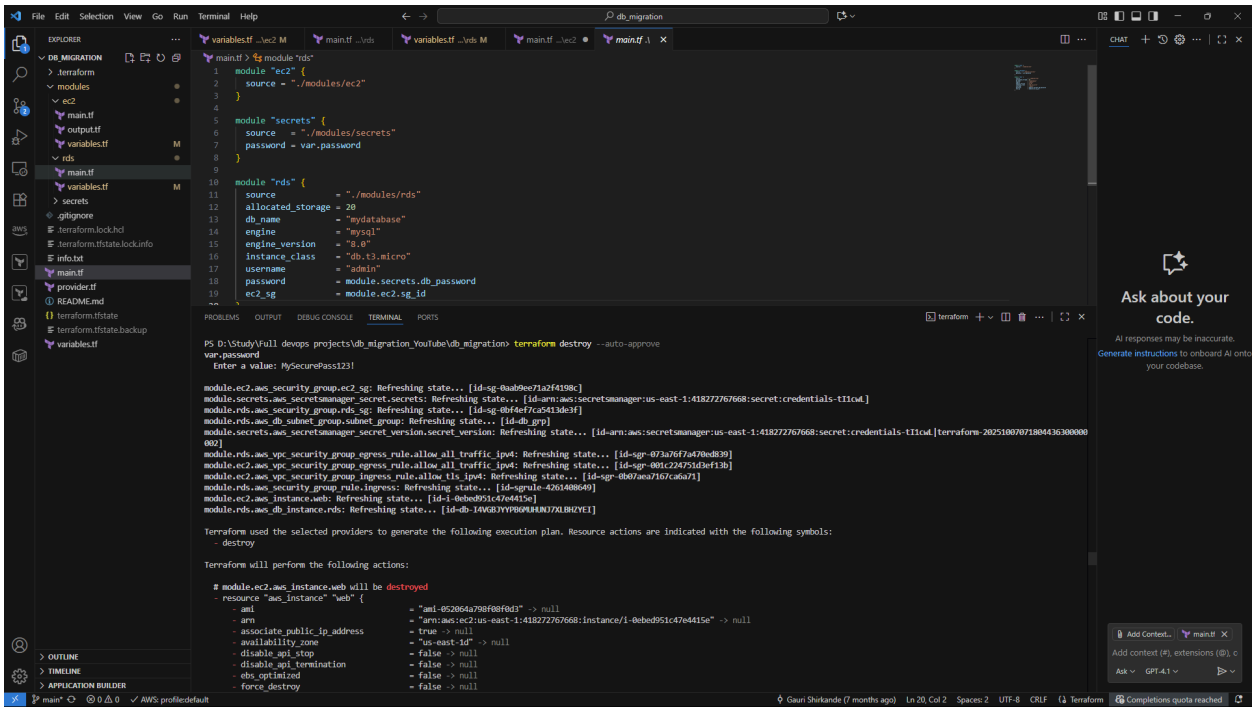
Step 21: Verify Data Integrity in Target RDS
A SELECT query was executed on the Employee table in RDS, confirming that all source data rows were successfully migrated and intact.



4. Infrastructure Teardown

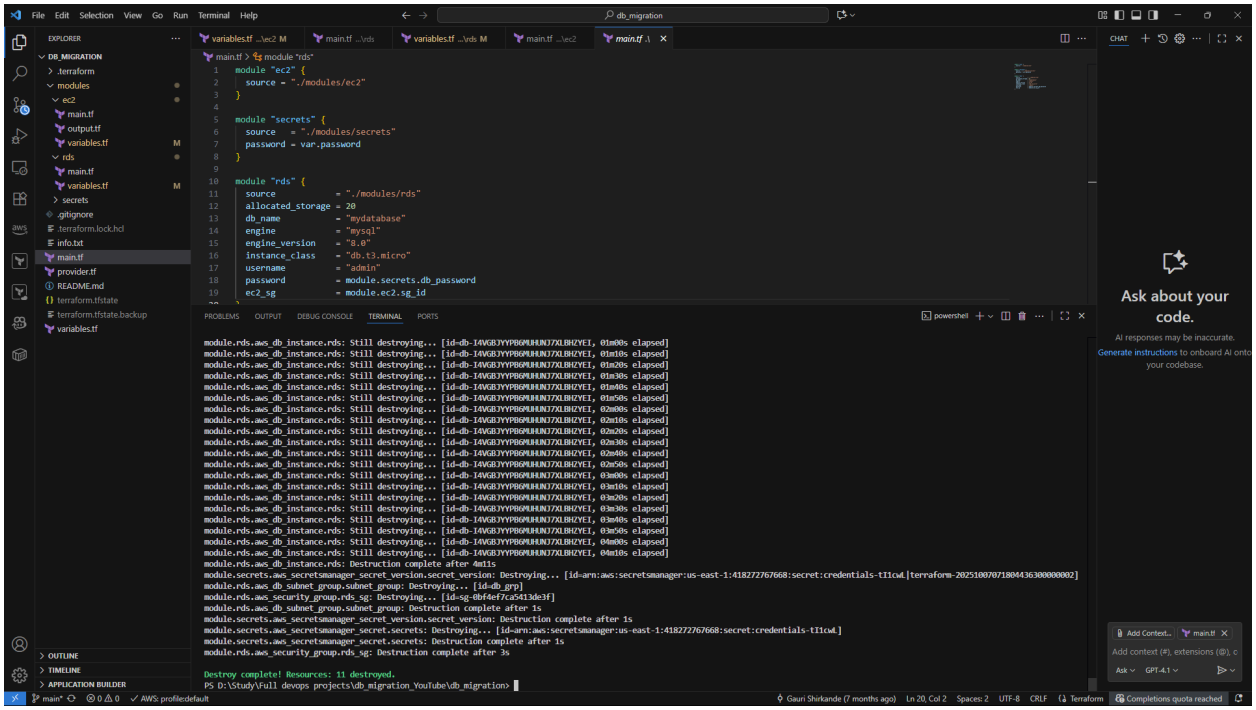
To manage cloud costs, the entire environment was systematically destroyed using Terraform.

Step 22: Execute terraform destroy
The terraform destroy -auto-approve command was executed to initiate the resource removal process.



Step 23: Verify Teardown Completion

The console confirmed the final completion of the process: "Destroy complete! Resources: 11 destroyed."



Conclusion

This project successfully demonstrated a robust, automated methodology for deploying complex cloud infrastructure using Terraform and executing a critical data migration task. All

phases, from infrastructure provisioning and security configuration to data transfer and environment cleanup, were completed efficiently and verified at each stage.