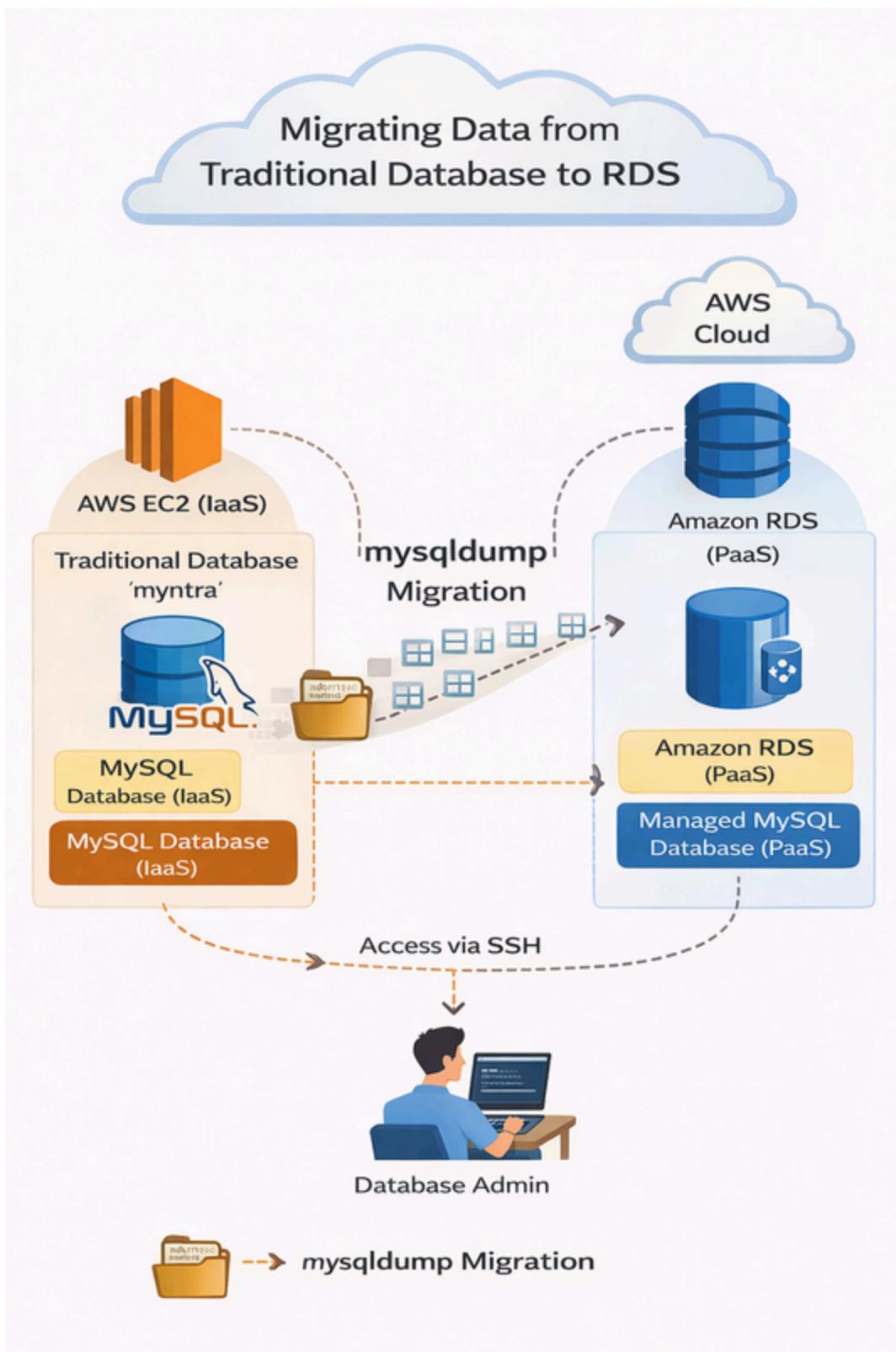


Migration of Traditional Database to AWS RDS (IaaS to PaaS)

Introduction

This project demonstrates the migration of data from a **traditional database hosted on an AWS EC2 instance (IaaS)** to **Amazon RDS (PaaS)**. The objective of the project is to understand how organizations can move from self-managed databases to managed database services provided by AWS for better scalability, availability, and reduced operational overhead.

In this project, a traditional database named `myntra` is hosted on an EC2 instance and migrated to **Amazon RDS**, showcasing the transition from **Infrastructure as a Service (IaaS)** to **Platform as a Service (PaaS)**.



Project Overview

- Traditional database hosted on **EC2 instance**
- Database name: **myntra**

- Target database: **Amazon RDS**
- Migration performed from **IaaS → PaaS**
- Data successfully transferred from EC2 database to RDS

Technologies and AWS Services Used

- **Amazon EC2** – Hosting traditional database (IaaS)
- **Amazon RDS** – Managed relational database service (PaaS)
- **MySQL / MariaDB** – Database engine
- **Amazon Linux** – Operating system
- **AWS Security Groups** – Network security
- **mysqldump** – Database migration tool
- **SSH** – Secure server access

Project Architecture

1. Traditional database **myntra** runs on an EC2 instance.
2. Amazon RDS instance is created with the same database engine.
3. Database data is exported from EC2 using dump utilities.
4. Exported data is imported into Amazon RDS.
5. Application can now connect to RDS instead of EC2 database.

Step-by-Step Project Implementation

Step 1: Launch EC2 Instance for Traditional Database

- Launch an EC2 instance using Amazon Linux.
- Install database server (MySQL/MariaDB).
- Create database named **myntra**.

EC2 > Instances > Launch an instance

Launch an instance Info

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

Name and tags Info

Name
Tradional-DB Add additional tags

Application and OS Images (Amazon Machine Image) Info

An AMI contains the operating system, application server, and applications for your instance. If you don't see a suitable AMI below, use the search field or choose **Browse more AMIs**.

Search our full catalog including 1000s of application and OS images

Recents Quick Start

Amazon Ubuntu Windows Red Hat SUSE Linux Debian Search

CloudShell Feedback Console Mobile App

© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Summary

Number of instances Info
1

Software Image (AMI)
Amazon Linux 2023 AMI 2023.9.2... read more
ami-0623300d1b7caee89

Virtual server type (Instance type)
t3.micro

Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Cancel Launch instance Preview code

```
sudo yum update -y
sudo yum install mariadb-server -y
sudo systemctl start mariadb
sudo systemctl enable mariadb
```

Create database:

```
CREATE DATABASE myntra;
```

Step 2: Insert Sample Data

```
USE myntra;
create table user (id int, name varchar(100), city varchar(100), age int);

insert into user values (1,"Om","Nashik",22);
insert into user values(2,"Rohit","Pune",23);
insert into user values(3,"Nikhil","Mumbai",27);
```

```

ec2-user@Tradional-DB:~ Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> create database myntra;
Query OK, 1 row affected (0.000 sec)

MariaDB [(none)]> show database;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'database' at line 1
MariaDB [(none)]> show databases;
+-----+
| Database |
+-----+
| information_schema |
| myntra |
| mysql |
| performance_schema |
+-----+
4 rows in set (0.001 sec)

MariaDB [(none)]> use myntra;
Database changed
MariaDB [myntra]> create table user (id int, name varchar(100),city varchar(100),age int);
Query OK, 0 rows affected (0.008 sec)

MariaDB [myntra]> insert into user values (1,"Om","Nashik",22),(2,"Rohit","Pune",23),(3."Nikhil","Mumbai",27);
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '"Nikhil","Mumbai",27)' at line 1
MariaDB [myntra]> insert into user values (1,"Om","Nashik",22),(2,"Rohit","Pune",23),(3,"Nikhil","Mumbai",27);
Query OK, 3 rows affected (0.002 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [myntra]> select * from user;
+---+---+---+---+
| id | name | city | age |
+---+---+---+---+
| 1 | Om | Nashik | 22 |
| 2 | Rohit | Pune | 23 |
| 3 | Nikhil | Mumbai | 27 |
+---+---+---+---+
3 rows in set (0.000 sec)

MariaDB [myntra]>

```

Step 3: Create Amazon RDS Instance

- Go to **RDS Dashboard**.
- Create a new RDS instance (MySQL/MariaDB).
- Configure:
 - DB instance identifier
 - Username and password
 - VPC and security group
- Enable inbound access from EC2 security group.

The screenshot shows the 'Create database' wizard in the AWS RDS console. The top navigation bar includes 'Aurora and RDS > Databases > Create database'. Below this, there are four tabs for different database engines: MariaDB (selected), Oracle, Microsoft SQL Server, and IBM Db2. Each tab has a small icon and a brief description. Under the MariaDB tab, there are sections for 'DB instance size' (Production, Dev/Test, Free tier) and 'DB instance identifier' (with a note about case-insensitivity). The 'Free tier' section is highlighted with a blue border. At the bottom of the page are links for CloudShell, Feedback, Console Mobile App, and standard footer links for 2025, Privacy, Terms, and Cookie preferences.

Step 4: Export Database from EC2 (IaaS)

```
sudo mysqldump -u root -p myntra > myntra_bkp.sql
```

Step 5: Import Data into Amazon RDS (PaaS)

```
mysql -h <RDS-ENDPOINT> -u <username> -p myntra < myntra.sql>
sudo mysql -h rds-db.c16w68u6ormo.us-west-1.rds.amazonaws.com -u admin -p
```

Step 6: Verify Migration

- Login to RDS database.
- Verify tables and data.

```
USE myntra;
SELECT * FROM user;
```

Key Differences: IaaS vs PaaS

Feature	Traditional DB (EC2)	Amazon RDS
Server Management	Manual	AWS Managed
Backups	Manual	Automatic
Scaling	Manual	Automated
Patching	Manual	AWS Managed
Availability	User Managed	Built-in

Benefits of Migrating to Amazon RDS

- Reduced administrative overhead
- Automatic backups and patching
- High availability and fault tolerance
- Easy scalability
- Improved security

Conclusion

This project successfully demonstrates the migration of a traditional database named **myntra** from an **EC2 instance (IaaS)** to **Amazon RDS (PaaS)**. The migration highlights how organizations can modernize their infrastructure by adopting managed database services to improve reliability, scalability, and operational efficiency.

This approach is widely adopted in real-world cloud environments and forms a crucial step in cloud migration strategies.