

Name: Meet Duseja

Batch: T12

Roll-No: 27

## **EXPERIMENT 9**

### **Aim:**

To understand Docker Architecture and Container Life Cycle, install Docker and execute docker commands to manage images and interact with containers.

### **Theory:**

Docker is a popular platform that enables developers to build, package, and deploy applications as lightweight, portable, and self-sufficient containers. These containers encapsulate all the necessary dependencies and libraries required for an application to run, ensuring consistency across different environments. Here is a theoretical overview of Docker:

#### **Containerization:**

Docker utilizes containerization technology to create isolated environments for applications. Containers are lightweight, standalone, and executable packages that include everything needed to run an application, such as code, runtime, system tools, libraries, and settings. This isolation ensures that applications run consistently across different environments, from development to production.

#### **Docker Engine:**

At the core of Docker is the Docker Engine, which is responsible for building, running, and managing containers. It consists of the Docker daemon, which manages containers, images, networks, and volumes, and the Docker client, which allows users to interact with the daemon through the Docker API.

#### **Docker Images:**

Docker images are read-only templates used to create containers. They contain the application code, runtime, libraries, dependencies, and other files needed to run the application. Images are built using Dockerfiles, which are text files that define the steps needed to create the image.

#### **Docker Containers:**

Containers are instances of Docker images that are running as isolated processes on a host machine. They are lightweight, portable, and can be easily started, stopped, moved, and deleted. Containers provide a consistent environment for applications to run, regardless of the underlying infrastructure.

#### **Benefits of Docker:**

**Portability:** Docker containers can run on any platform that supports Docker, making it easy to deploy applications across different environments.

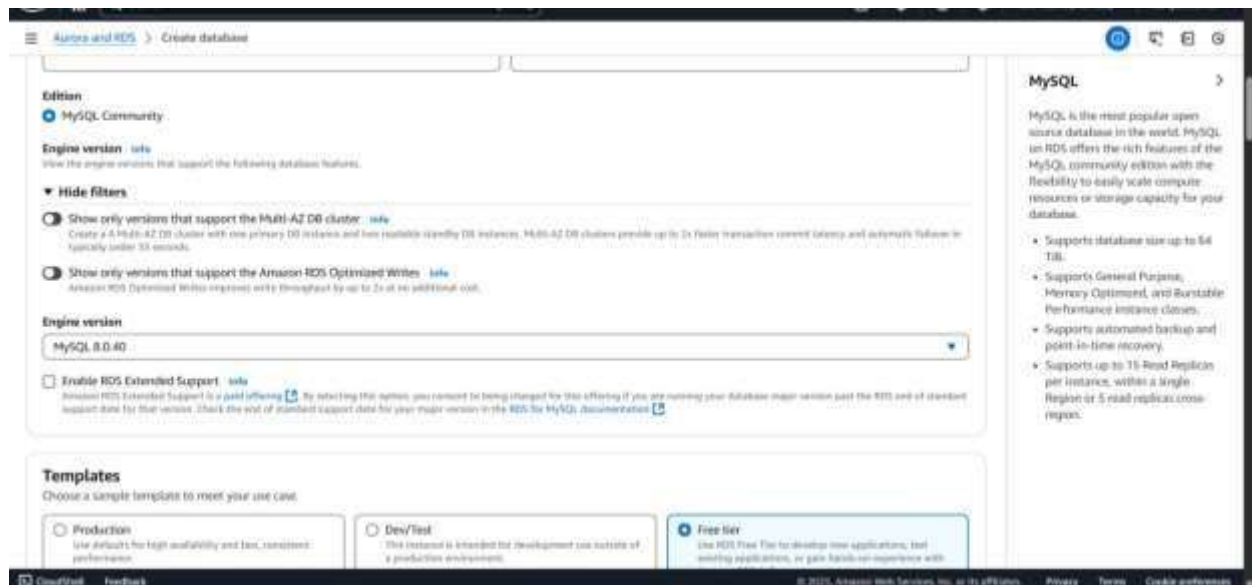
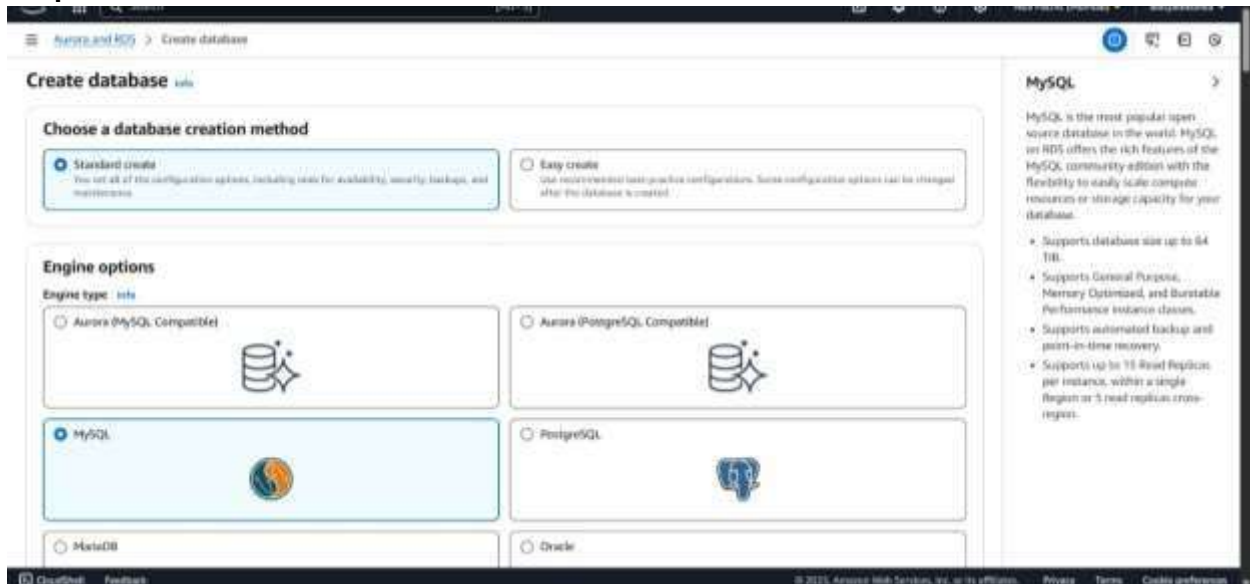
**Efficiency:** Containers share the host OS kernel, reducing overhead and improving resource utilization.

**Isolation:** Containers provide a level of isolation that helps prevent conflicts between applications and dependencies.

**Scalability:** Docker enables easy scaling of applications by quickly spinning up additional containers.

**Consistency:** Docker ensures that applications run the same way in development, testing, and production environments.

## Output:



**Settings**

**DB instance identifier** [info](#)  
Type a name for your DB instance. This name must be unique across all DB instances owned by your AWS account in the current AWS Region.

The DB instance identifier is case-insensitive, but is stored as all lowercase (set in "mysqlinstanceid"). Constraints: 3 to 63 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

▼ **Credentials Settings**

**Master username** [info](#)  
Type a login ID for the master user of your DB instance.

Can't be alphanumeric characters. The first character must be a letter.

**Credentials management**  
You can use AWS Secrets Manager or manage your master user credentials.

☐ Managed in AWS Secrets Manager - **most secure**  
AWS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

☒ Self managed  
Create your own password or have AWS create a password that you manage.

☐ Auto-generate password  
Amazon RDS will generate a password for you, or you can specify your own password.

**Master password** [info](#)

**Password strength** [info](#)

**MySQL**

MySQL is the most popular open-source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

© 2015, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

**Instance configuration**

The DB instance configuration options below are limited to those supported by the engine that you selected above.

**DB instance class** [info](#)

▼ **Hide filters**

☒ Show instance classes that support Amazon RDS Optimized Windows [info](#)  
Amazon RDS Optimized Windows instances with throughput up to 30 GB/sec additional cost.

☒ Include previous generation classes

☐ Standard classes (includes all classes)

☐ Memory optimized classes (includes 1 and 2 classes)

☒ Burstable classes (includes 1 class)

2 vCPUs 1 GB RAM Network: up to 2.5GBbps

**Storage**

**Storage type** [info](#)  
Provisioned IOPS SSD (io2) storage volumes are now available.

Baseline performance determined by instance size

**Allocated storage** [info](#)  
 GB

**MySQL**

MySQL is the most popular open-source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

© 2015, Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie preferences](#)

Amazon and RDS > Create database

Database

☒ **Don't connect to an EC2 compute resource**  
Don't set up a connection to a compute resource for this database. You can manually set up a connection to a compute resource later.

☐ **Connect to an EC2 compute resource**  
Set up a connection to an EC2 compute resource for this database.

**Virtual private cloud (VPC)** info  
 Choose the VPC. The VPC defines the virtual networking environment for this DB instance.

Create new VPC

Only VPCs with a corresponding DB subnet group are listed.

After a database is created, you can't change its VPC.

**DB subnet group** info  
 Choose the DB subnet group. The DB subnet group defines which subnets and IP ranges the DB instance can use in the VPC that you selected.

Create new DB Subnet Group

**Public access** info

☒ **Yes**  
 RDS assigns a public IP address to the database. Amazon EC2 instances and other resources outside of the VPC can connect to your database. Resources inside the VPC can also connect to the database. (Choose one or more VPC security groups that specify which resources can connect to the database.)

☐ **No**  
 RDS doesn't assign a public IP address to the database. Only Amazon EC2 instances and other resources inside the VPC can connect to your database. Choose one or more VPC security groups that specify which resources can connect to the database.

**VPC security group (firewall)** info  
 Choose one or more VPC security groups to allow access to your database. Make sure that the security group rules allow the appropriate incoming traffic.

☐ Choose existing  
Choose existing VPC security groups.

☒ **Create new**  
Create new VPC security group.

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TiB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

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

Amazon and RDS > Create database

Availability Zone info

No preference

**RDS Proxy**  
 RDS Proxy is a fully managed, highly available database proxy that improves application scalability, resiliency, and security.

☐ **Create an RDS Proxy** info  
 RDS Proxy is available in all AWS regions and a Security Manager secret for the proxy. RDS Proxy has additional costs. For more information, see [Amazon RDS Proxy pricing](#).

**Certificate authority - optional** info  
 Using a server certificate provides an extra layer of security by confirming that the connection is being made to an Amazon database. It does so by checking the server certificate that is automatically installed on all databases that you provision.

rn-cs-rs2048-g1 (default)  
Expires May 31, 2026

If you don't select a certificate authority, RDS chooses one for you.

**Additional configuration**

**Tags - optional**  
 A tag consists of a case-sensitive key-value pair.

Key

Q ENV X Use URI X Remove

Add new tag

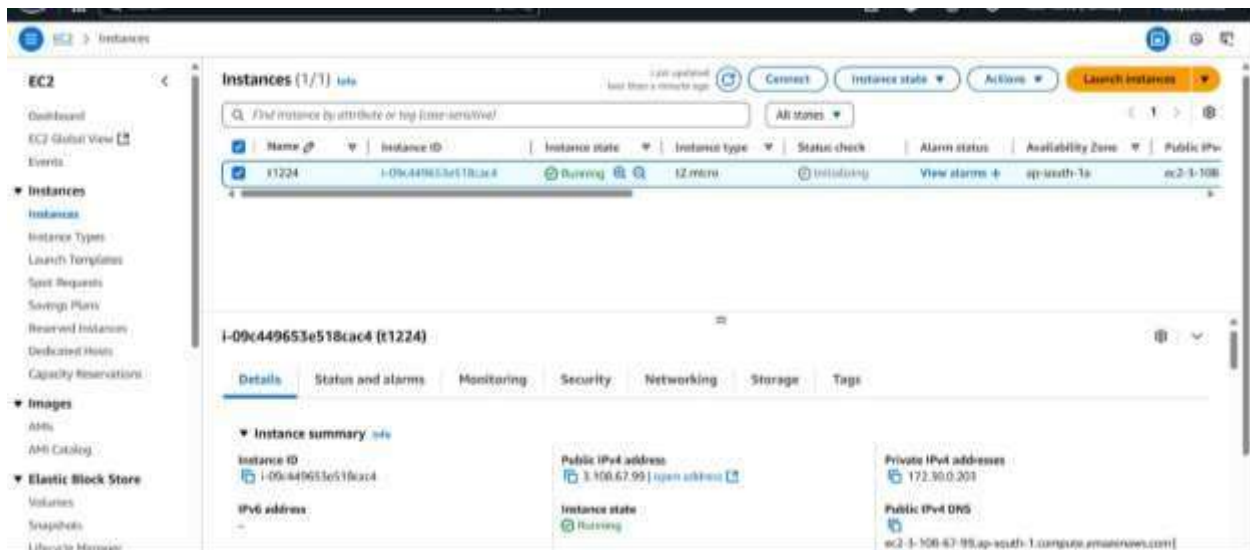
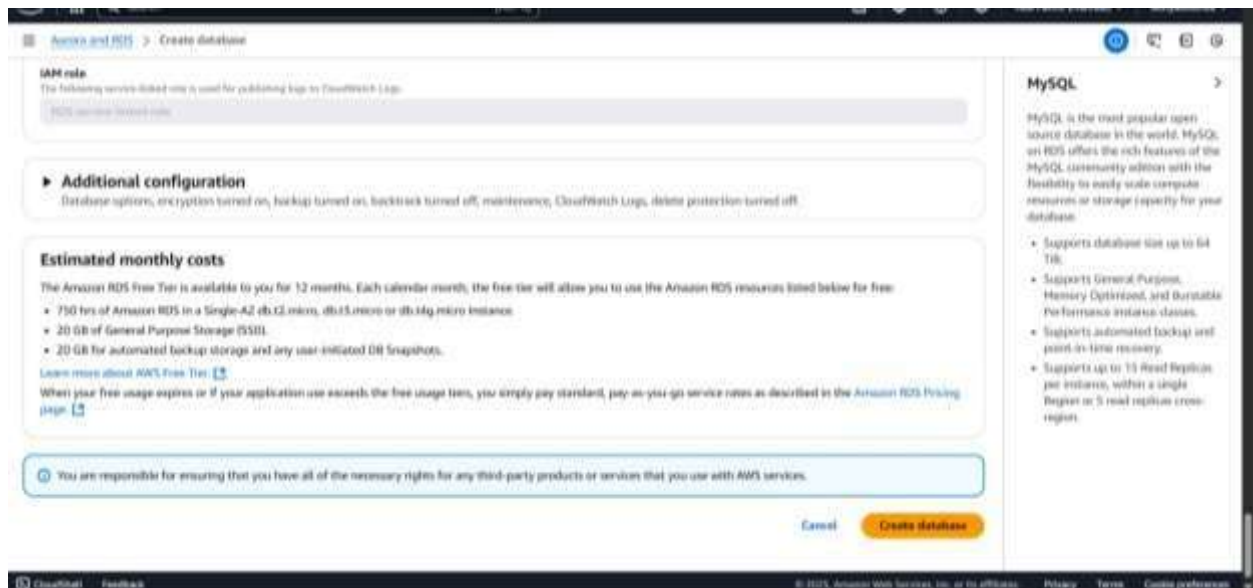
You can add up to 50 more tags.

MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 64 TiB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 15 Read Replicas per instance, within a single Region or 5 read replicas cross-region.

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



```

[ec2-user@ip-172-30-0-203 ~]$ sudo yum install -y docker
Amazon Linux 2023 Kernel livepatch repository
115 MB/s | 35 kB | 00:00
Dependencies resolved.

```

Package	Architecture	Version	Repository	Size
Installing:				
docker	x86_64	25.0.0-1.amzn2023.0.1	amazonlinux	44
Installing dependencies:				
containersd	x86_64	1.7.25-1.amzn2023.0.1	amazonlinux	36
iptables-libe	x86_64	1.8.6-3.amzn2023.0.2	amazonlinux	401
iptables-soft	x86_64	1.8.6-3.amzn2023.0.2	amazonlinux	183
libgroup	x86_64	3.0-1.amzn2023.0.1	amazonlinux	75
libnetfilter_conntrack	x86_64	1.0.8-2.amzn2023.0.2	amazonlinux	50
libnetfilter_log	x86_64	1.0.1-19.amzn2023.0.2	amazonlinux	30
libnetfilter_tcp	x86_64	1.2.2-2.amzn2023.0.2	amazonlinux	84
pkg	x86_64	2.6-1.amzn2023.0.1	amazonlinux	83
run	x86_64	1.2.4-1.amzn2023.0.1	amazonlinux	2.4

```

Transaction Summary

```

```

run
x86_64
1.2.4-1.amzn2023.0.1
amazonlinux
2.4
Transaction Summary

```

Package	Architecture	Version	Repository	Size
Install 10 Packages				
Total download size: 94 M				
Installed size: 319 M				
Downloading Packages:				
1/10   iptables-libe-1.8.6-3.amzn2023.0.2.x86_64.rpm				7.3 MB/s   401 kB   00:00
2/10   iptables-soft-1.8.6-3.amzn2023.0.2.x86_64.rpm				7.0 MB/s   183 kB   00:00
3/10   libgroup-3.0-1.amzn2023.0.1.x86_64.rpm				3.4 MB/s   75 kB   00:00
4/10   libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64.rpm				1.5 MB/s   50 kB   00:00
5/10   libnetfilter_log-1.0.1-19.amzn2023.0.2.x86_64.rpm				1.3 MB/s   30 kB   00:00
6/10   libnetfilter_tcp-1.2.2-2.amzn2023.0.2.x86_64.rpm				3.8 MB/s   84 kB   00:00
7/10   pkg-2.6-1.amzn2023.0.1.x86_64.rpm				2.7 MB/s   83 kB   00:00
8/10   run-1.2.4-1.amzn2023.0.1.x86_64.rpm				24 MB/s   2.4 kB   00:00
9/10   containersd-1.7.25-1.amzn2023.0.1.x86_64.rpm				44 MB/s   36 kB   00:00
10/10   docker-25.0.0-1.amzn2023.0.1.x86_64.rpm				60 MB/s   44 kB   00:01
Total				
				74 MB/s   94 kB   00:01

```
Verifying : pipx-2.5-1.amzn2023.0.1.x86_64 5/1
Verifying : rpm-1.2.4-1.amzn2023.0.1.x86_64 10/1

Installed:
  containers-1.7.25-1.amzn2023.0.1.x86_64      docker-25.0.3-1.amzn2023.0.1.x86_64      iptables-libs-1.8.6-2.amzn2023.0.2.x86_64
  iptables-soft-1.8.6-3.amzn2023.0.2.x86_64    libgroup-3.0-1.amzn2023.0.1.x86_64      libnetfilter_conntrack-1.0.9-2.amzn2023.0.2.x86_64
  libnftables-1.0.1-19.amzn2023.0.2.x86_64    libnftnl-1.2.3-2.amzn2023.0.2.x86_64    pipx-2.5-1.amzn2023.0.1.x86_64
  rpm-1.2.4-1.amzn2023.0.1.x86_64

Complete!
mc2-user@ip-172-30-0-203 ~$ sudo systemctl start docker
mc2-user@ip-172-30-0-203 ~$ sudo systemctl status docker
* docker.service - Docker Application Container Engine
   Loaded: loaded /usr/lib/systemd/system/docker.service; disabled; preset: disabled
   Active: active (running) since Tue 2025-04-01 13:34:00 UTC; 11s ago
   TriggeredBy: * docker.socket
   Docs: https://docs.docker.com
   Process: 27031 ExecStartPre=/bin/sh -p /run/docker code=exited, status=0/SUCCESS
   Process: 27034 ExecStartPre=/usr/libexec/docker/docker-setup-runtime.sh code=exited, status=0/SUCCESS
   Main PID: 27035 (dockerd)
   Tasks: 7
   Memory: 28.1M
   CPU: 25ms
   CGroup: /system.slice/docker.service
           └─27035 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nfile=32768:65536

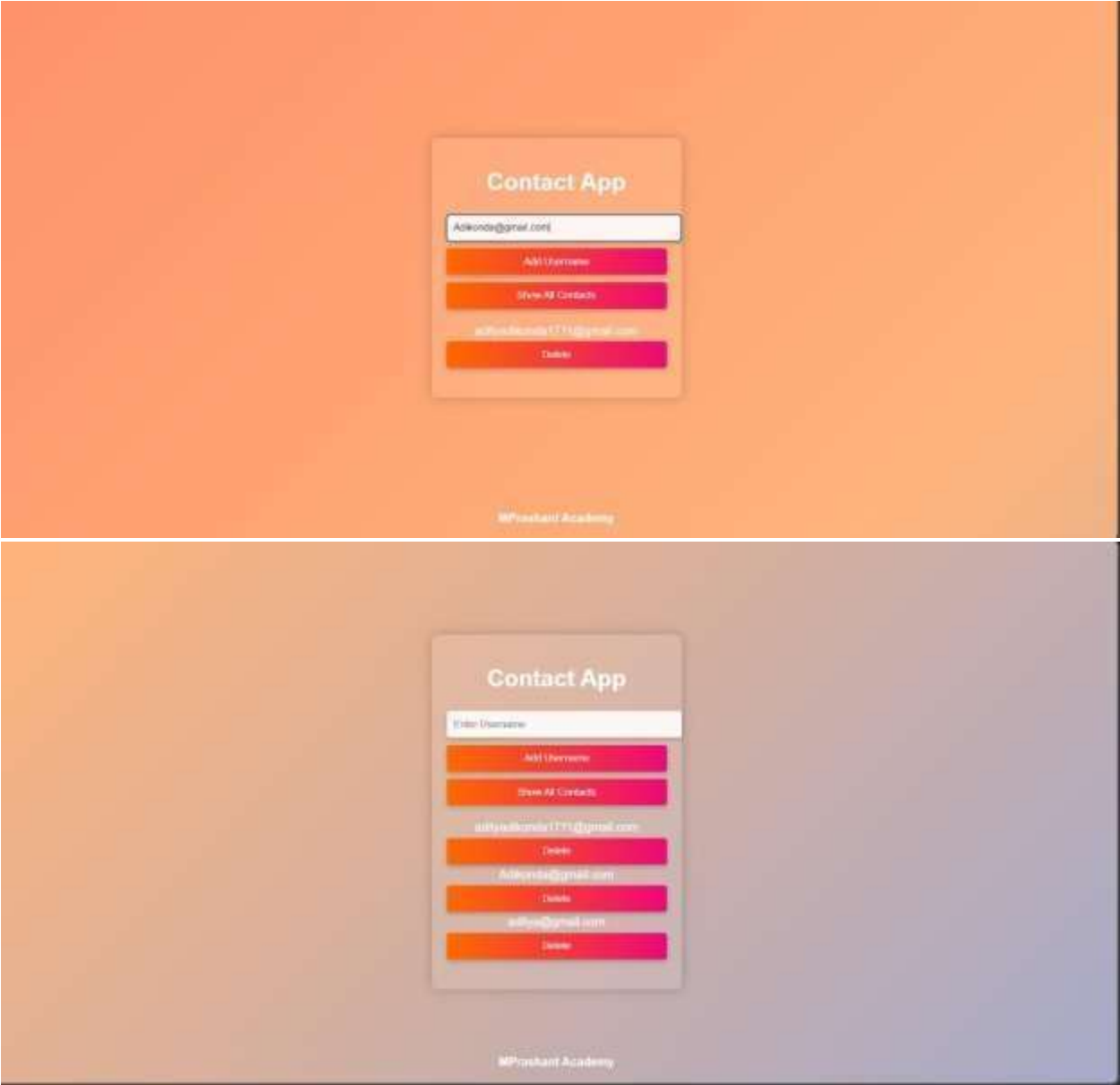
Apr 01 13:34:03 ip-172-30-0-203.ap-south-1.compute.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Apr 01 13:34:07 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:07.456215534E" level=info msg="Starting up"
Apr 01 13:34:07 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:07.715129540E" level=info msg="Loading containers: start."
Apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.117426764E" level=info msg="Loading containers: done."
Apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.181350659E" level=info msg="Docker daemon" commit=7196ca contai
```

```
Tasks: 7
Memory: 28.1M
CPU: 25ms
CGroup: /system.slice/docker.service
           └─27035 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nfile=32768:65536

Apr 01 13:34:03 ip-172-30-0-203.ap-south-1.compute.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Apr 01 13:34:07 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:07.456215534E" level=info msg="Starting up"
Apr 01 13:34:07 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:07.715129540E" level=info msg="Loading containers: start."
Apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.117426764E" level=info msg="Loading containers: done."
Apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.181350659E" level=info msg="Docker daemon" commit=7196ca contai

Apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.181367909E" level=info msg="Daemon has completed initialization"
Apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.182701250E" level=info msg="API listen on /run/docker.sock"
Apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal systemd[1]: Started docker.service - Docker Application Container Engine.

mc2-user@ip-172-30-0-203 ~$ sudo docker pull philippaul/node-mysql-app:02
02: Pulling from philippaul/node-mysql-app
ff1d7041c74: Pull complete
b53aafca57: Pull complete
bd2201bd999c: Pull complete
4de76e26b1d: Pull complete
09andf5b8551: Pull complete
251aed0954e4: Pull complete
f12ed43c3229: Pull complete
8c0c2f24a8d: Pull complete
5d7a8e8132: Pull complete
255ca9a95db0: Pull complete
66a162df3db1: Pull complete
75b1a7bce97: Pull complete
f7978b845b1: Pull complete
Digest: sha256:f761cfd42a7f44806424030378b438c8ef3f8930682cd43f395bf7e42946b
Status: Downloaded newer image for philippaul/node-mysql-app:02
docker.io/philippaul/node-mysql-app:02
mc2-user@ip-172-30-0-203 ~$
```





```
awscli [AWS]
Search

[Alt+Q]

Fetch the logs of a container
ec2-user@ip-172-30-0-203 ~$ sudo docker run -it --rm mysql:8.0 mysql -h ti224.c3aaq9w6pq.ap-south-1.rds.amazonaws.com -u admin -p
Unable to find image 'mysql:8.0' locally
.0: Pulling from library/mysql
ba37246e33e: Pull complete
8e01aa55f13: Pull complete
9fa3211d7a7: Pull complete
53e0441f7a6: Pull complete
d539a14fale: Pull complete
e336ff7314e2: Pull complete
3272c957f26: Pull complete
10644002388: Pull complete
136f43f54c2d: Pull complete
d54979e7120: Pull complete
a6f2a7f39b5: Pull complete
Digest: sha256:b4577815b32ab01d631fb261eabbb5c73507eda87c2745730251533ef030e
Status: Downloaded newer image for mysql:8.0
Enter password:
Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 28
Server version: 8.0.40 Source distribution

Copyright (c) 2000, 2020, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

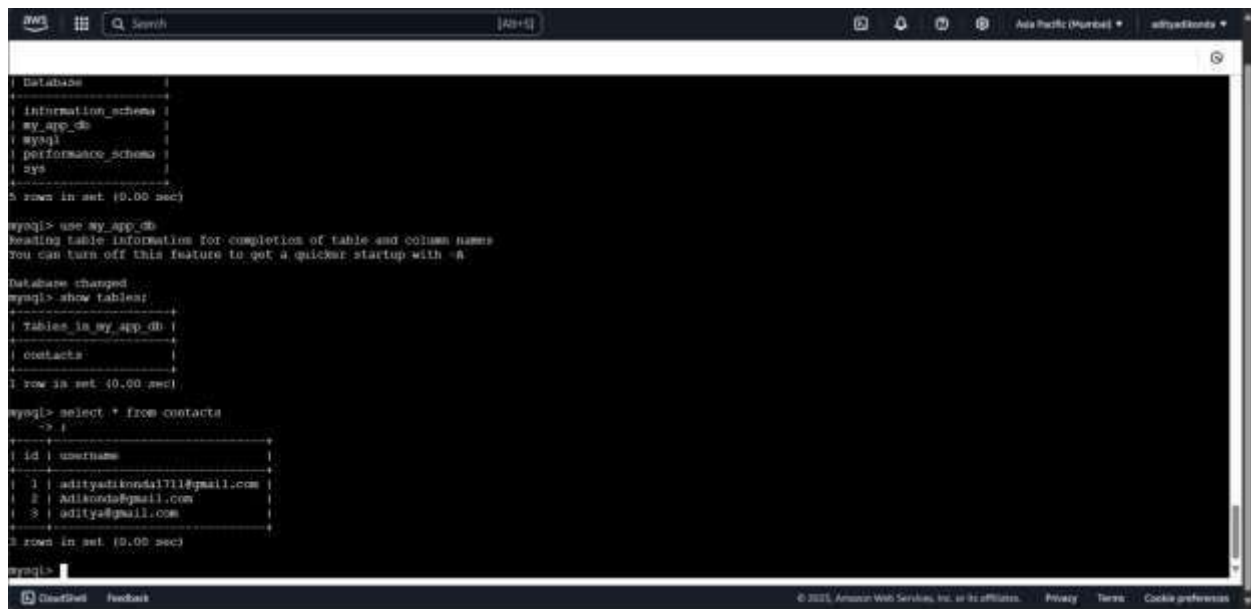
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> show databases;
+-----+
| Database |
+-----+
| information_schema |
| my_app_db |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)

mysql> use my_app_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_my_app_db |
+-----+
| contacts |
+-----+
1 row in set (0.00 sec)

mysql> select * from contacts;
+-----+
| id | username |
+-----+
| 1 | adityadikondal71@gmail.com |
| 2 | adityadikondal71@gmail.com |
+-----+
```



The screenshot shows an AWS CloudShell terminal window with a dark background. The terminal displays the following commands and output:

```
Database
+-----+
| information_schema |
| my_app_db          |
| mysql              |
| performance_schema |
| sys                |
+-----+
3 rows in set (0.00 sec)

mysql> use my_app_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----+
| Tables_in_my_app_db |
+-----+
| contacts             |
+-----+
1 row in set (0.00 sec)

mysql> select * from contacts
-> +-----+
| id | username |
+-----+
| 1  | adityatikondal711@gmail.com |
| 2  | Aditkonda@gmail.com        |
| 3  | aditya@gmail.com           |
+-----+
3 rows in set (0.00 sec)

mysql>
```

The bottom of the terminal window shows the AWS CloudShell interface with the AWS logo, a search bar, and the user's profile information (Asia Pacific (Mumbai) and adityatikonda).

## Conclusion:

Docker revolutionizes the software development and deployment process by providing a powerful platform for containerization. By encapsulating applications and their dependencies into lightweight, portable containers,