

## **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:

The screenshot displays the AWS Management Console interface for creating a new database. The top navigation bar shows the AWS logo, a search bar, and the user's profile (adityadikonda) in the Asia Pacific (Mumbai) region. The breadcrumb trail indicates the path: Aurora and RDS > Create database.

The main content area is titled "Create database" and includes an "Info" link. It is divided into several sections:

- Choose a database creation method:** Two options are available: "Standard create" (selected) and "Easy create".
- Engine options:** A grid of database engines is shown, including Aurora (MySQL Compatible), Aurora (PostgreSQL Compatible), MySQL (selected), PostgreSQL, MariaDB, and Oracle.
- MySQL details (right sidebar):** A summary of MySQL features, including support for database size up to 64 TiB, General Purpose, Memory Optimized, and Burstable Performance instance classes, automated backup and point-in-time recovery, and up to 15 Read Replicas per instance.
- Edition:** "MySQL Community" is selected.
- Engine version:** A dropdown menu shows "MySQL 8.0.40".
- Hide filters:** Two filter options are present: "Show only versions that support the Multi-AZ DB cluster" and "Show only versions that support the Amazon RDS Optimized Writes".
- Enable RDS Extended Support:** A checkbox option for extended support.
- Templates:** Three templates are listed: "Production", "Dev/Test", and "Free tier" (selected).

The bottom of the console shows the "CloudShell" button, a "Feedback" link, and the footer with copyright information (© 2025, Amazon Web Services, Inc. or its affiliates) and links to Privacy, Terms, and Cookie preferences.

3

4

The image shows two screenshots of the AWS Management Console. The top screenshot is the 'Create database' page for Amazon RDS, and the bottom screenshot is the 'Instances' page for Amazon EC2.

**Top Screenshot: Amazon RDS Create database**

**IAM role**  
The following service-linked role is used for publishing logs to CloudWatch Logs.  
RDS service-linked role

**Additional configuration**  
Database options, encryption turned on, backup turned on, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

**Estimated monthly costs**  
The Amazon RDS Free Tier is available to you for 12 months. Each calendar month, the free tier will allow you to use the Amazon RDS resources listed below for free:

- 750 hrs of Amazon RDS in a Single-AZ db.t2.micro, db.t3.micro or db.t4g.micro Instance.
- 20 GB of General Purpose Storage (SSD).
- 20 GB for automated backup storage and any user-initiated DB Snapshots.

[Learn more about AWS Free Tier.](#)  
When your free usage expires or if your application use exceeds the free usage tiers, you simply pay standard, pay-as-you-go service rates as described in the [Amazon RDS Pricing page.](#)

**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.

**Bottom Screenshot: Amazon EC2 Instances**

**Instances (1/1)**  
Last updated less than a minute ago

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IPv
t1224	i-09c449653e518cac4	Running	t2.micro	Initializing	View alarms	ap-south-1a	ec2-3-108

**i-09c449653e518cac4 (t1224)**

**Details** | Status and alarms | Monitoring | Security | Networking | Storage | Tags

**Instance summary**

<b>Instance ID</b> i-09c449653e518cac4	<b>Public IPv4 address</b> 3.108.67.99   <a href="#">open address</a>	<b>Private IPv4 addresses</b> 172.30.0.203
<b>IPv6 address</b> -	<b>Instance state</b> Running	<b>Public IPv4 DNS</b> ec2-3-108-67-99.ap-south-1.compute.amazonaws.com

```
aws [Alt+S]
[ec2-user@ip-172-30-0-203 ~]$ sudo yum install -y docker
Amazon Linux 2023 Kernel Livepatch repository 115 kB/s | 15 kB 00:00
Dependencies resolved.
=====
Package Architecture Version Repository Size
=====
Installing:
docker x86_64 25.0.8-1.amzn2023.0.1 amazonlinux 44
Installing dependencies:
containerd x86_64 1.7.25-1.amzn2023.0.1 amazonlinux 36
iptables-libs x86_64 1.8.8-3.amzn2023.0.2 amazonlinux 401
iptables-nft x86_64 1.8.8-3.amzn2023.0.2 amazonlinux 183
libcgroup x86_64 3.0-1.amzn2023.0.1 amazonlinux 75
libnetfilter_conntrack x86_64 1.0.8-2.amzn2023.0.2 amazonlinux 58
libnftnl x86_64 1.0.1-19.amzn2023.0.2 amazonlinux 30
libnftnl x86_64 1.2.2-2.amzn2023.0.2 amazonlinux 84
pigz x86_64 2.5-1.amzn2023.0.3 amazonlinux 83
runc x86_64 1.2.4-1.amzn2023.0.1 amazonlinux 3.4
Transaction Summary
=====
CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws [Alt+S]
runc x86_64 1.2.4-1.amzn2023.0.1 amazonlinux 3.4
Transaction Summary
=====
Install 10 Packages
Total download size: 84 M
Installed size: 319 M
Downloading Packages:
(1/10): iptables-libs-1.8.8-3.amzn2023.0.2.x86_64.rpm 7.3 MB/s | 401 kB 00:00
(2/10): iptables-nft-1.8.8-3.amzn2023.0.2.x86_64.rpm 7.0 MB/s | 183 kB 00:00
(3/10): libcgroup-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 1.3 MB/s | 58 kB 00:00
(5/10): libnftnl-1.0.1-19.amzn2023.0.2.x86_64.rpm 1.3 MB/s | 30 kB 00:00
(6/10): libnftnl-1.2.2-2.amzn2023.0.2.x86_64.rpm 3.8 MB/s | 84 kB 00:00
(7/10): pigz-2.5-1.amzn2023.0.3.x86_64.rpm 2.7 MB/s | 83 kB 00:00
(8/10): runc-1.2.4-1.amzn2023.0.1.x86_64.rpm 24 MB/s | 3.4 MB 00:00
(9/10): containerd-1.7.25-1.amzn2023.0.1.x86_64.rpm 44 MB/s | 36 MB 00:00
(10/10): docker-25.0.8-1.amzn2023.0.1.x86_64.rpm 40 MB/s | 44 MB 00:01
--
Total 74 MB/s | 84 MB 00:01
Running transaction check
CloudShell Feedback © 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences
```

```
aws [Alt+S] Search [Alt+S] Asia Pacific (Mumbai) adityadikonda
```

```
Verifying : pigz-2.5-1.amzn2023.0.3.x86_64 9/1
Verifying : runc-1.2.4-1.amzn2023.0.1.x86_64 10/1

Installed:
containerd-1.7.25-1.amzn2023.0.1.x86_64      docker-25.0.8-1.amzn2023.0.1.x86_64      iptables-libs-1.8.8-3.amzn2023.0.2.x86_64
iptables-nft-1.8.8-3.amzn2023.0.2.x86_64      libcgrou-3.0-1.amzn2023.0.1.x86_64      libnetfilter_conntrack-1.0.8-2.amzn2023.0.2.x86_64
libnftnl-1.0.1-19.amzn2023.0.2.x86_64      libnftnl-1.2.2-2.amzn2023.0.2.x86_64      pigz-2.5-1.amzn2023.0.3.x86_64
runc-1.2.4-1.amzn2023.0.1.x86_64

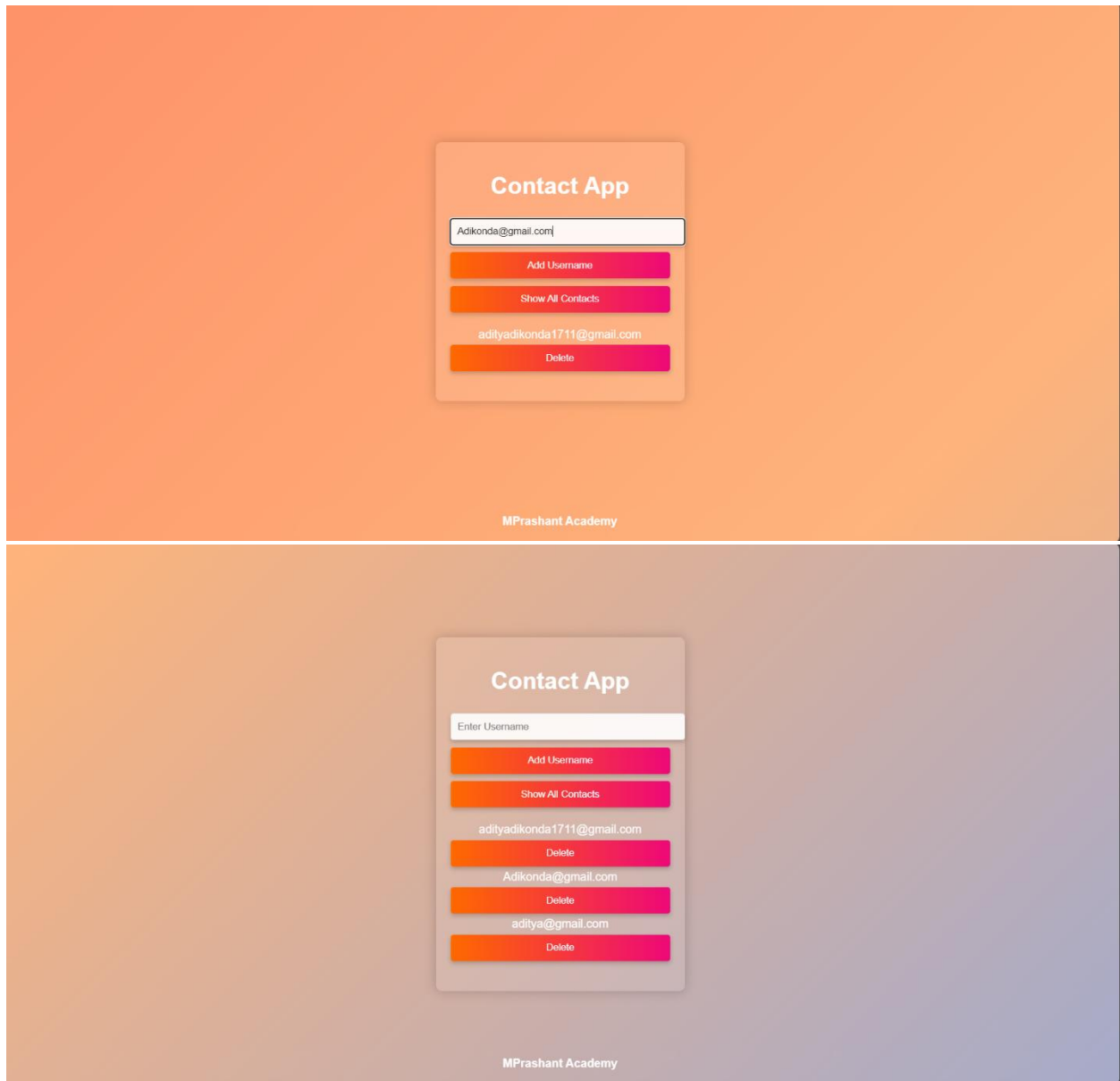
Complete!
[ec2-user@ip-172-30-0-203 ~]$ sudo systemctl start docker
[ec2-user@ip-172-30-0-203 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; disabled; preset: disabled)
   Active: active (running) since Tue 2025-04-01 13:34:08 UTC; 11s ago
   TriggeredBy: ● docker.socket
     Docs: https://docs.docker.com
    Process: 27033 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
    Process: 27034 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
   Main PID: 27035 (dockerd)
      Tasks: 7
     Memory: 28.1M
        CPU: 258ms
     CGroup: /system.slice/docker.service
             └─27035 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

apr 01 13:34:07 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="2025-04-01T13:34:07.656215534Z" level=info msg="Starting up"
apr 01 13:34:07 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:07.715129546Z" level=info msg="Loading containers: start."
apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.117426754Z" level=info msg="Loading containers: done."
apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.141150659Z" level=info msg="Docker daemon" commit=71907ca contai
```

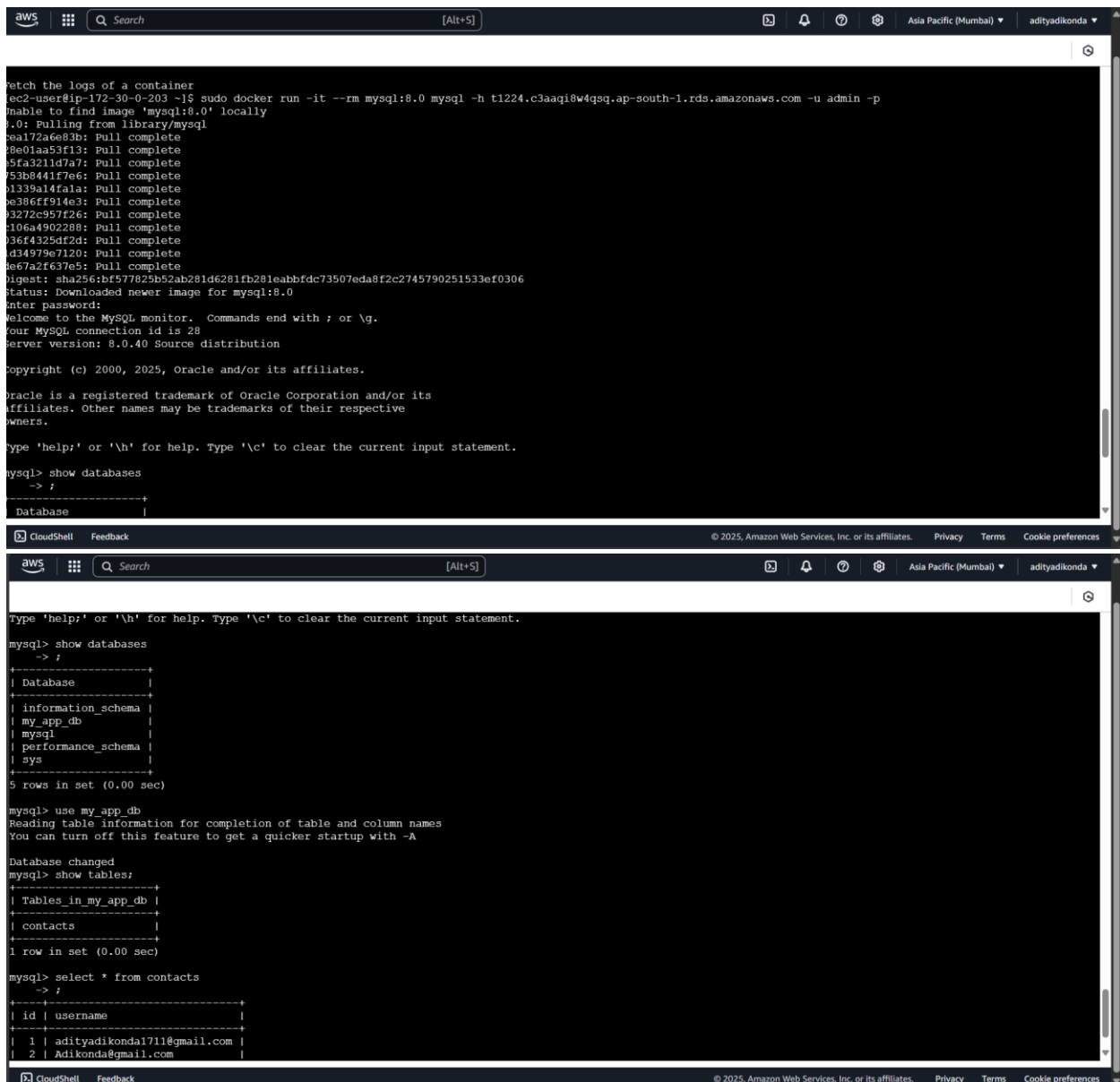
```
Tasks: 7
Memory: 28.1M
CPU: 258ms
CGroup: /system.slice/docker.service
        └─27035 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

apr 01 13:34:07 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="2025-04-01T13:34:07.656215534Z" level=info msg="Starting up"
apr 01 13:34:07 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:07.715129546Z" level=info msg="Loading containers: start."
apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.117426754Z" level=info msg="Loading containers: done."
apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.141150659Z" level=info msg="Docker daemon" commit=71907ca contai
apr 01 13:34:08 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:08.141367009Z" 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="2025-04-01T13:34:08.182701850Z" 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.
lines 1-22/22 (END)
[ec2-user@ip-172-30-0-203 ~]$ sudo docker pull philippaul/node-mysql-app:02
02: Pulling from philippaul/node-mysql-app
2ff1d7c41c74: Pull complete
b253aaefaea7: Pull complete
bd2201bd995c: Pull complete
1de76e268b10: Pull complete
49a8df589451: Pull complete
5f51ee005dea: Pull complete
5f32ed3c3f27: Pull complete
0c8cc2f24a4d: Pull complete
0d27a8e86132: Pull complete
b36ca9a95db0: Pull complete
16a182df3db1: Pull complete
f5b1a7ebae97: Pull complete
ff7978b844b1: Pull complete
Digest: sha256:f7c1cffb42a2f4a40b626b0d03f8b83bbc8ef3f88d0682cd43f395bf9e42966b
Status: Downloaded newer image for philippaul/node-mysql-app:02
docker.io/philippaul/node-mysql-app:02
[ec2-user@ip-172-30-0-203 ~]$
```

```
aws [Alt+S] Search [Alt+S] Asia Pacific (Mumbai) adityadikonda
```







The screenshot shows two sequential terminal windows from the AWS CloudShell interface. The top window shows the process of pulling a MySQL 8.0 image and starting the MySQL monitor. The bottom window shows the execution of SQL commands to list databases, use a specific database, list tables, and query data from a table named 'contacts'.

```
Fetch the logs of a container
ec2-user@ip-172-30-0-203 ~]$ sudo docker run -it --rm mysql:8.0 mysql -h t1224.c3aaqi8w4qsq.ap-south-1.rds.amazonaws.com -u admin -p
Unable to find image 'mysql:8.0' locally
D: Pulling from library/mysql
ea172a6e83b: Pull complete
28e01aa53f13: Pull complete
55fa3211d7a7: Pull complete
753b8441f7e6: Pull complete
01339a14fala: Pull complete
6e386ff914e3: Pull complete
93272c957f26: Pull complete
106a4902288: Pull complete
036f4325df2d: Pull complete
d94979e7120: Pull complete
1e67a2f637e5: Pull complete
Digest: sha256:bf577825b52ab281d6281fb281eabbfd73507eda8f2c2745790251533ef0306
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, 2025, 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.

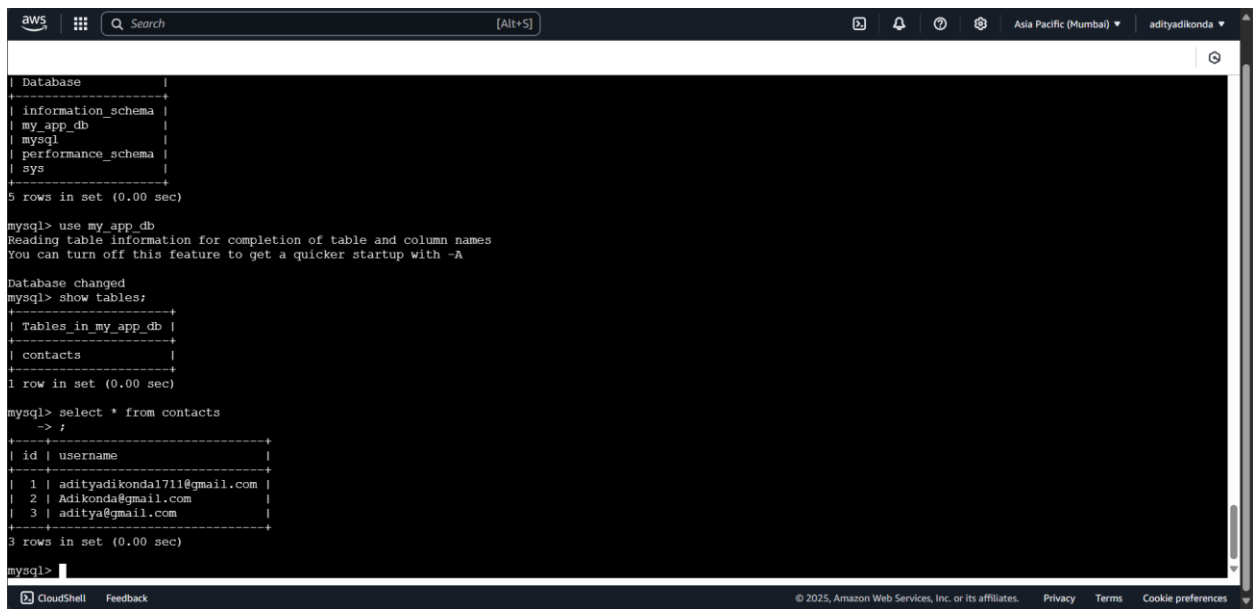
mysql> show databases
+-----+
| Database |
+-----+

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 | adityadikondal711@gmail.com |
| 2 | Adikonda@gmail.com |
+----+-----+
```



The screenshot shows an AWS CloudShell terminal window with a dark background. The terminal displays the output of several MySQL commands. The first command shows the database structure, including schemas like 'information\_schema', 'my\_app\_db', 'mysql', 'performance\_schema', and 'sys'. The second command switches to the 'my\_app\_db' database and shows its tables, which includes a table named 'contacts'. The third command queries the 'contacts' table, returning three rows of data with columns 'id' and 'username'.

```
aws
[Alt+S]
Search
Asia Pacific (Mumbai)
adityadikonda

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  | adityadikondal711@gmail.com          |
| 2  | Adikonda@gmail.com                  |
| 3  | aditya@gmail.com                    |
+-----+
3 rows in set (0.00 sec)

mysql>
```

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

### 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,