

Software Engineering and Project Management Lab Experiment No: - 09

Aim: To Study and Implement a Container using Docker.

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 an open-source platform that enables developers to automate the deployment of applications inside lightweight, portable, and self-sufficient containers. Containers package the application along with its dependencies and environment, ensuring consistent behavior across different systems.

Key Features of Docker:

- ❖ **Lightweight:** Shares the host OS kernel, reducing overhead compared to VMs.
- ❖ **Portability:** Runs the same container on any environment (dev, test, prod).
- ❖ **Isolation:** Applications run in isolated environments with their own file systems and dependencies.
- ❖ **Fast startup:** Containers start in seconds, enabling rapid deployment.
- ❖ **Version control:** Docker images can be versioned and managed easily.
- ❖ **Microservices-friendly:** Perfect for deploying individual services in a microservices architecture.

Docker as a Containerization Tool

Docker uses containerization to simplify software development, testing, and deployment. It provides tools to build, ship, and run applications inside containers.

Key Docker Components:

Docker Engine Core service for building and running containers

Docker Image Read-only template used to create containers

Docker Container A runnable instance of a Docker image

Dockerfile A script with instructions to build a Docker image

Docker Hub Cloud-based registry to share and download images

Docker Compose Tool for defining and running multi-container apps

Docker Volume Persistent storage for containers

Docker Network Manage communication between containers

Demonstration of Docker (Theoretical Steps)

1. Install Docker
 - i. Download Docker Desktop from <https://www.docker.com>.
 - ii. Install and start Docker Engine on your system (Linux, Windows, or macOS).
 - iii. Verify installation by running:
docker --version
2. Pull a Docker Image
 - i. Use Docker Hub to pull a pre-built image:
docker pull nginx

Software Engineering and Project Management Lab Experiment No: - 09

Aim: To Study and Implement a Container using Docker.

3. Run a Docker Container

- i. Start a container using the pulled image:

```
docker run -d -p 8080:80 nginx
```

- ii. Access the web server by navigating to `http://localhost:8080` in your browser.

4. Build a Custom Docker Image

- i. Create a file named Dockerfile:

```
FROM python:3.10
```

```
COPY app.py /app/app.py
```

```
WORKDIR /app
```

```
CMD ["python", "app.py"]
```

- ii. Build the image using:

```
docker build -t my-python-app .
```

- iii. Run the container:

```
docker run -d -p 5000:5000 my-python-app
```

5. Manage Containers

- i. View running containers:

```
docker ps
```

- ii. Stop a container:

```
docker stop <container_id>
```

- iii. Remove a container:

```
docker rm <container_id>
```

6. Use Docker Compose (Optional)

- i. Create a `docker-compose.yml` file to define multiple services.

- ii. Start all services using:

```
docker-compose up
```

Use Case Example:

- ❖ Containerizing web applications for consistent deployment
- ❖ Running microservices architecture in isolated containers
- ❖ Developing and testing across different environments
- ❖ Deploying scalable applications on cloud or on-premises
- ❖ Creating reproducible development environments
- ❖ CI/CD pipeline integration for automated testing and delivery
- ❖ Education and experimentation with different tech stacks

Implementation:

Software Engineering and Project Management Lab Experiment No: - 09

Aim: To Study and Implement a Container using Docker.

[Aurora and RDS](#) > Create database

Create database [Info](#)


Choose a database creation method


☒ Standard create
You set all of the configuration options, including ones for availability, security, backups, and maintenance.


☐ Easy create
Use recommended best-practice configurations. Some configuration options can be changed after the database is created.


Engine options

Engine type [Info](#)

☐ Aurora (MySQL Compatible) 

☐ Aurora (PostgreSQL Compatible) 

☒ MySQL 

☐ PostgreSQL 

☐ MariaDB

☐ Oracle

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.

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

[Aurora and RDS](#) > Create database

Edition

☒ MySQL Community

Engine version [Info](#)

View the engine versions that support the following database features.

Hide filters

☒ Show only versions that support the Multi-AZ DB cluster [Info](#)
Create a Multi-AZ DB cluster with one primary DB instance and two readable standby DB instances. Multi-AZ DB clusters provide up to 2x faster transaction commit latency and automatic failover in typically under 35 seconds.

☒ Show only versions that support the Amazon RDS Optimized Writes [Info](#)
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

Engine version

MySQL 8.0.40

☐ Enable RDS Extended Support [Info](#)
Amazon RDS Extended Support is a paid offering. By selecting this option, you consent to being charged for this offering if you are running your database major version past the RDS end of standard support date for that version. Check the end of standard support date for your major version in the [RDS for MySQL documentation](#).

Templates

Choose a sample template to meet your use case.

☐ Production
Use defaults for high availability and fast, consistent performance.

☐ Dev/Test
This instance is intended for development use outside of a production environment.

☒ Free tier
Use RDS Free Tier to develop new applications, test existing applications, or gain hands-on experience with

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.

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

Software Engineering and Project Management Lab Experiment No: - 09

Aim: To Study and Implement a Container using Docker.

[Aurora and RDS](#) > Create database

Settings

DB instance identifier [Info](#)
Type a name for your DB instance. The 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 (as in "mydbinstance"). Constraints: 1 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.

1 to 16 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**
RDS generates a password for you and manages it throughout its lifecycle using AWS Secrets Manager.

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

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

Master password [Info](#)

Password strength [Strong](#)

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.

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

[Aurora and RDS](#) > Create database

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 Writes** [Info](#)
Amazon RDS Optimized Writes improves write throughput by up to 2x at no additional cost.

☐ **Include previous generation classes**

☐ Standard classes (includes m classes)

☐ Memory optimized classes (includes r and x classes)

☒ **Burstable classes (includes t classes)**

▼
2 vCPUs 1 GiB RAM Network: Up to 2,085 Mbps

Storage

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

▼
Baseline performance determined by volume size

Allocated storage [Info](#)

GIB

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.

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

Software Engineering and Project Management Lab Experiment No: - 09

Aim: To Study and Implement a Container using Docker.

Aurora 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.

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.

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

Aurora 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 automatically creates an IAM role and a Secrets 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 validating 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.
rds-ca-rsa2048-g1 (default)
Expiry: May 20, 2061
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 UAT X Remove
Q UAT X
[Add new tag](#)
You can add up to 49 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.

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

Software Engineering and Project Management Lab Experiment No: - 09

Aim: To Study and Implement a Container using Docker.

Aurora and 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.](#)

You are responsible for ensuring that you have all of the necessary rights for any third-party products or services that you use with AWS services.

Cancel Create database

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.

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

EC2 > Instances

Instances (1/1) info
Last updated less than a minute ago
Connect Instance state Actions Launch instances

Find Instance by attribute or tag (case-sensitive) All states

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone	Public IP
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 info

Instance ID i-09c449653e518cac4	Public IPv4 address 3.108.67.99 open address	Private IPv4 addresses 172.30.0.203
IPv6 address -	Instance state Running	Public IPv4 DNS ec2-3-108-67-99-ap-south-1.compute.amazonaws.com

Aim: To Study and Implement a Container using Docker.

```

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

```

```

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

```


Software Engineering and Project Management Lab Experiment No: - 09

Aim: To Study and Implement a Container using Docker.

```
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      libcgrouper-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

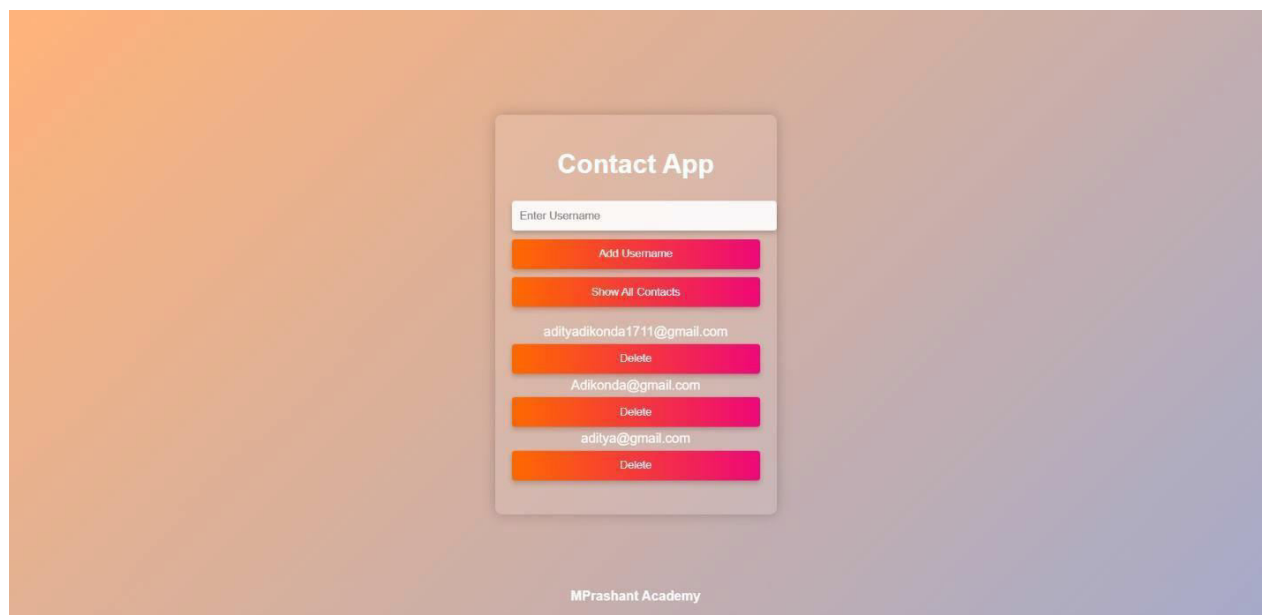
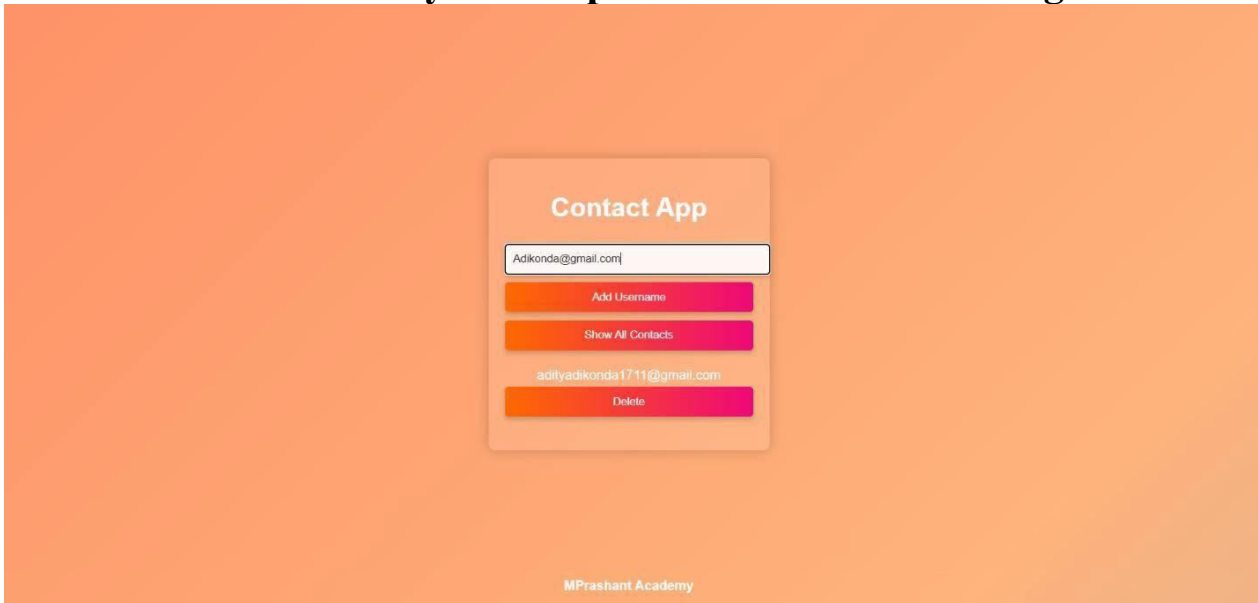
Apr 01 13:34:07 ip-172-30-0-203.ap-south-1.compute.internal systemd[1]: Starting docker.service - Docker Application Container Engine...
Apr 01 13:34:07 ip-172-30-0-203.ap-south-1.compute.internal dockerd[27035]: time="2025-04-01T13:34:07.656215534Z" level=info msg="Starting up"
Apr 01 13:34:07 ip-172-30-0-203.ap-south-1.compute.internal 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" containerd=1.7.25-1.amzn2023.0.1.x86_64 docker=25.0.8-1.amzn2023.0.1.x86_64 iptables=1.8.8-3.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 libnetfilter_conntrack=1.0.8-2.amzn2023.0.2.x86_64 libcgrouper=3.0-1.amzn2023.0.1.x86_64 pigz=2.5-1.amzn2023.0.3.x86_64 runc=1.2.4-1.amzn2023.0.1.x86_64
```

```
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" containerd=1.7.25-1.amzn2023.0.1.x86_64 docker=25.0.8-1.amzn2023.0.1.x86_64 iptables=1.8.8-3.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 libnetfilter_conntrack=1.0.8-2.amzn2023.0.2.x86_64 libcgrouper=3.0-1.amzn2023.0.1.x86_64 pigz=2.5-1.amzn2023.0.3.x86_64 runc=1.2.4-1.amzn2023.0.1.x86_64
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
b253aeaafaa7: Pull complete
8d2201bd995c: Pull complete
1de76e268b10: Pull complete
49a8df589451: Pull complete
ff51ee005dea: Pull complete
ff32ed3c3f27: Pull complete
0c8cc2f24a4d: Pull complete
0d27a8e86132: Pull complete
b35ca9a95db0: Pull complete
46a182df3db1: Pull complete
f5b1a7ebee97: Pull complete
ff7978b84db1: Pull complete
Digest: sha256:f7c1cfff42a2f4a40b626b0d03f8b83bbcf3f88d0682cd43f395bf9e42966b
Status: Downloaded newer image for philippaul/node-mysql-app:02
docker.io/philippaul/node-mysql-app:02
[ec2-user@ip-172-30-0-203 ~]$
```


Software Engineering and Project Management Lab Experiment No: - 09

Aim: To Study and Implement a Container using Docker.



Software Engineering and Project Management Lab Experiment No: - 09

Aim: To Study and Implement a Container using Docker.

```
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
8.0: Pulling from library/mysql
2eal72a6e83b: Pull complete
88e01aa53f13: Pull complete
5fa321d7a7: Pull complete
f3b8441f7e6: Pull complete
d1339a14fa1a: Pull complete
e386ff914e3: Pull complete
93272c957f26: Pull complete
c106a4902288: Pull complete
036f4325df2d: Pull complete
d34979e7120: Pull complete
8e67a2f637e5: 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 |
+-----+
```

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

```
mysql> show tables;
+-----+
| Tables_in_my_app_db |
+-----+
| contacts |
+-----+
1 row 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>

Software Engineering and Project Management Lab Experiment No: - 09

Aim: To Study and Implement a Container using Docker.

Conclusion: We have successfully understood Docker Architecture and Container Life Cycle, installed Docker and executed docker commands to manage images and interact with containers.

LO Mapping: *LO is mapped*