

Cloud-Based Task Management System Using AWS 3-Tier Architecture

Problem statement and solution:

Organizations and teams often struggle to track tasks, priorities, ownership, and progress using manual methods or locally hosted applications. These systems lack centralized access, real-time visibility, scalability, and secure data storage. As the number of users and tasks increases, traditional single-server systems face performance issues, data inconsistency, security risks, and downtime.

There is a need for a centralized, cloud-based system that allows users to create, store, and manage task information such as task title, description, priority, assigned person, and status in a reliable and scalable environment. The system must ensure secure access, high availability, controlled network communication, and persistent data storage.

This project addresses the problem by implementing a Cloud-Based Task Management System using AWS 3-Tier Architecture, where the frontend provides a user-friendly interface for task entry and visualization, the backend processes business logic, and the database securely stores task records in the cloud.

Aim of the project:

The aim of this project is to design, deploy, and validate a secure, scalable, and highly available Cloud-Based Task Management System using AWS 3-Tier Architecture. The project focuses on implementing best practices in cloud networking, security, and application deployment by separating the presentation layer, application layer, and database layer within a custom AWS Virtual Private Cloud (VPC). It aims to demonstrate real-world usage of AWS services such as EC2, Application Load Balancer, RDS, and secure networking components to deliver a reliable and enterprise-grade cloud solution.

Project description:

This project demonstrates the design and deployment of a highly available and secure cloud-native web application using AWS 3-Tier Architecture. The solution is built on a custom Virtual Private Cloud (VPC) with segmented public and private subnets to isolate application layers and enforce security best practices.

The presentation tier is hosted on an Amazon EC2 instance running Nginx, serving a responsive web interface accessible from the internet. User traffic is routed through an Application Load Balancer (ALB) to the application tier, where backend services run on private EC2 instances using Node.js and Express. This tier processes API requests and communicates securely with the database tier, which consists of an Amazon RDS MySQL instance deployed in a private subnet.

Network connectivity is controlled using route tables, Internet Gateway, NAT Gateway, and security groups, ensuring that only authorized traffic flows between tiers. Backend instances access the internet securely for updates via NAT Gateway without exposing private IPs publicly. The load balancer distributes traffic and provides fault tolerance and scalability.

AWS Services used:

Amazon VPC - Creates a private isolated network for deploying cloud resources securely.

Public Subnet - Hosts internet-facing resources like frontend EC2 and Load Balancer.

Private Subnet - Hosts backend EC2 and database securely without public exposure.

Internet Gateway (IGW) - Allows public subnet resources to access the internet.

NAT Gateway - Enables private instances to access the internet securely without being publicly exposed.

Route Tables - Controls how traffic flows between subnets and gateways.

Security Groups - Acts as a virtual firewall controlling inbound and outbound traffic.

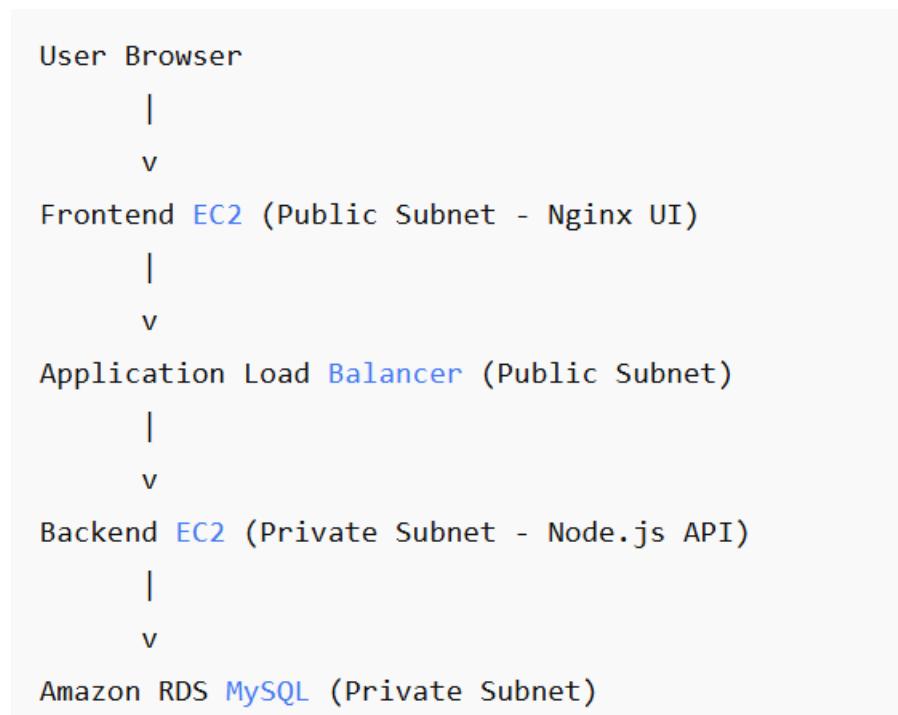
Amazon EC2 - Provides virtual servers to run frontend and backend applications.

Application Load Balancer (ALB) - Distributes incoming traffic across backend instances for scalability and high availability.

Target Group - Defines the backend instances that receive traffic from the load balancer.

Amazon RDS (MySQL) - Provides a managed relational database for storing application data securely.

EC2 Instance Connect Endpoint - Allows secure browser-based SSH access to private EC2 instances.



Step 1: User Access

The user opens the application using a browser and accesses the frontend through the public IP of the Frontend EC2 instance.

Step 2: Frontend Presentation Layer

The frontend EC2 instance hosts the HTML, CSS, and JavaScript UI using Nginx. The user enters task details such as title, description, priority, assigned person, and status.

Step 3: API Request Forwarding

When the user submits the form, the frontend sends an HTTP request to the backend API through the Application Load Balancer DNS endpoint.

Step 4: Load Balancer Routing

The Application Load Balancer receives the request and forwards it to a healthy backend EC2 instance based on target group health checks.

Step 5: Backend Processing

The backend EC2 instance processes the request using Node.js and Express, validates input data, and prepares SQL queries.

Step 6: Database Interaction

The backend securely connects to the Amazon RDS MySQL database located in a private subnet and stores or retrieves task data.

Creation of VPC:

The screenshot shows the AWS VPC dashboard with the following details:

- VPC ID:** `vpc-09e68f4610fb4699c`
- Name:** Task_VPC
- State:** Available
- Tenancy:** default
- Block Public Access:** Off
- DNS resolution:** Enabled
- DNS hostnames:** Disabled
- DHCP option set:** dopt-0d024e1633bdc38f8
- Main route table:** rtb-0d2ec5393c5d902a4

CIDR: 10.0.0.0/16

Creation of subnet:

The screenshot shows the AWS VPC Subnets page. The left sidebar is titled "Virtual private cloud" and includes sections for "Your VPCs", "Subnets", "Route tables", "Internet gateways", "Egress-only internet gateways", "DHCP option sets", "Elastic IPs", "Managed prefix lists", "Endpoints", "Endpoint services", "NAT gateways", "Peering connections", and "Route servers". The main content area is titled "Subnets (7) Info" and displays a table with the following data:

Name	Subnet ID	State	VPC	Block Public...	IPv4 CIDR
Private_subnet2	subnet-014fee740dc552e82	Available	vpc-09e68f4610fb4699c	Off	10.0.3.0/24
Private_subnet	subnet-07b2002d87e75c76c	Available	vpc-09e68f4610fb4699c	Off	10.0.2.0/24
Public_subnet	subnet-041f626e5d86fe6f5	Available	vpc-09e68f4610fb4699c	Off	10.0.1.0/24
-	subnet-0f5896ca658c9c836	Available	vpc-05b98c1f416089832	Off	172.31.0.0/16
public_subnet2_alb	subnet-008478e9dc9bcd35d	Available	vpc-09e68f4610fb4699c	Off	10.0.4.0/24

Below the table, there is a section titled "Select a subnet" with a dropdown menu.

Creation of route table:

The screenshot shows the AWS VPC Route tables page. The left sidebar is identical to the previous screenshot, showing the "Virtual private cloud" section with various networking components. The main content area is titled "Route tables (4) Info" and displays a table with the following data:

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC
Public_rt	rtb-045a860b1dece0e3	2 subnets	-	No	vpc-09e68f4610fb4699c
Private_rt	rtb-029542354b8c1bfe3	2 subnets	-	No	vpc-09e68f4610fb4699c
-	rtb-07856530a588bd320	-	-	Yes	vpc-05b98c1f416089832
-	rtb-0d2ec5393c5d902a4	-	-	Yes	vpc-09e68f4610fb4699c

Below the table, there is a section titled "Select a route table" with a dropdown menu.

Screenshot of the AWS VPC Route Tables page for the Public_rt route table.

Route tables (1/4) Info

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
Public_rt	rtb-045a860b1dece0e3	2 subnets	-	No	vpc-09e68f4610fb4699c T
Private_rt	rtb-029542354b8c1bfe3	2 subnets	-	No	vpc-09e68f4610fb4699c T
-	rtb-07856530a588bd320	-	-	Yes	vpc-05b98c1f416089832
-	rtb-0d2ec5393c5d902a4	-	-	Yes	vpc-09e68f4610fb4699c T

rtb-045a860b1dece0e3 / Public_rt

Routes (2)

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	igw-091ae6e54794de6cd	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table

Screenshot of the AWS VPC Route Tables page for the Public_rt route table.

Route tables (1/4) Info

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
Public_rt	rtb-045a860b1dece0e3	2 subnets	-	No	vpc-09e68f4610fb4699c T
Private_rt	rtb-029542354b8c1bfe3	2 subnets	-	No	vpc-09e68f4610fb4699c T
-	rtb-07856530a588bd320	-	-	Yes	vpc-05b98c1f416089832
-	rtb-0d2ec5393c5d902a4	-	-	Yes	vpc-09e68f4610fb4699c T

rtb-045a860b1dece0e3 / Public_rt

Subnet associations

Explicit subnet associations (2)

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
Public_subnet	subnet-041f626e5d86fe6f5	10.0.1.0/24	-
public_subnet2_alb	subnet-008478e9dc9bcd35d	10.0.4.0/24	-

Screenshot of the AWS VPC Route Tables page for the Private_rt route table.

Route tables (1/4) Info

Name	Route table ID	Explicit subnet associ...	Edge associations	Main	VPC
Public_rt	rtb-045a860b1dece0e3	2 subnets	-	No	vpc-09e68f4610fb4699c T
Private_rt	rtb-029542354b8c1bfe3	2 subnets	-	No	vpc-09e68f4610fb4699c T
-	rtb-07856530a588bd320	-	-	Yes	vpc-05b98c1f416089832
-	rtb-0d2ec5393c5d902a4	-	-	Yes	vpc-09e68f4610fb4699c T

rtb-029542354b8c1bfe3 / Private_rt

Routes (2)

Destination	Target	Status	Propagated	Route Origin
0.0.0.0/0	nat-0b22a4d925c54d7d5	Active	No	Create Route
10.0.0.0/16	local	Active	No	Create Route Table

The screenshot shows the AWS VPC Route Tables page. The left sidebar includes sections for VPC dashboard, Virtual private cloud (Your VPCs, Subnets, Route tables), and other services like EC2, S3, IAM, VPC, Aurora and RDS. The main content area displays a table of route tables:

Name	Route table ID	Explicit subnet assoc...	Edge associations	Main	VPC
Public_rt	rtb-045a860a1dece0e3	2 subnets	-	No	vpc-09e68f4610fb4699c T
Private_rt	rtb-029542354b8c1bfe3	2 subnets	-	No	vpc-09e68f4610fb4699c T
-	rtb-07856530a588bd320	-	-	Yes	vpc-05b98c1f416089832
-	rtb-0d2ec5393c5d902a4	-	-	Yes	vpc-09e68f4610fb4699c T

Below the table, a specific route table (rtb-029542354b8c1bfe3 / Private_rt) is selected, showing its details: Subnet associations, Edge associations, Route propagation, and Tags. The Subnet associations tab is active, displaying two explicit subnet associations:

Name	Subnet ID	IPv4 CIDR	IPv6 CIDR
Private_subnet2	subnet-014fee740dc552e82	10.0.3.0/24	-
Private_subnet	subnet-07b2002d87e75c76c	10.0.2.0/24	-

Creation of internet gateway:

The screenshot shows the AWS VPC Internet Gateways page. The left sidebar includes sections for VPC dashboard, Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways), and other services like EC2, S3, IAM, VPC, Aurora and RDS. The main content area displays a table of internet gateways:

Name	Internet gateway ID	State	VPC ID	Owner
-	igw-01745a7b878d9956e	Attached	vpc-05b98c1f416089832	969084114868
Task_igw	igw-091ae6e54794de6cd	Attached	vpc-09e68f4610fb4699c Task_VPC	969084114868

A "Select an internet gateway above" message is displayed below the table.

Creation of NAT gateway:

The screenshot shows the AWS VPC NAT Gateways page. The left sidebar includes sections for VPC dashboard, Virtual private cloud (Your VPCs, Subnets, Route tables, Internet gateways, Egress-only internet gateways), and other services like EC2, S3, IAM, VPC, Aurora and RDS. The main content area displays a table of NAT gateways:

Name	NAT gateway ID	Connectivity...	State	State message	Availability ...	Route table ID	Zonal
Task_NAT	nat-0b22a4d925c54d7d5	Public	Available	-	-	-	-

A "Select a NAT gateway" message is displayed below the table.

Creation of security group:

The screenshot shows the AWS VPC Security Groups page. The left sidebar lists various VPC-related resources like Subnets, Route tables, and Internet gateways. The main area displays a table of security groups with columns for Name, Security group ID, Security group name, VPC ID, and Description. The table contains eight entries, including default, Frontend_sg, rds_sg, Backend_sg, launch-wizard-1, ALB_sg, dg-sg, and another default entry.

Inbound rules of front_end sg:

The screenshot shows the 'Edit inbound rules' page for the 'Frontend_sg' security group. It lists three existing rules: one for SSH (TCP port 22), one for HTTP (TCP port 80), and one for Custom TCP (TCP port 3000) with source 'sg-0a0aa0c65c478c1ad'. A note at the bottom advises against using 0.0.0.0/0 or -/0 as it allows all IP addresses. Buttons for 'Add rule', 'Preview changes', and 'Save rules' are visible at the bottom right.

Inbound rules of back_end sg:

The screenshot shows the 'Edit inbound rules' page for the 'Backend_sg' security group. It lists two existing rules: one for Custom TCP (TCP port 3000) with source 'sg-090d5c740b07e5995' and one for SSH (TCP port 22) with source 'sg-0a0aa0c65c478c1ad'. A note at the bottom advises against using 0.0.0.0/0 or -/0 as it allows all IP addresses. Buttons for 'Add rule', 'Preview changes', and 'Save rules' are visible at the bottom right.

Inbound rules of rds:

The screenshot shows the 'Edit inbound rules' page for a security group rule ID 'sgr-05e74824fe7340459'. The rule is of type 'MySQL/Aurora' using 'TCP' on port '3306' from a custom source 'sg-0e0ce0ae2aa135104'. There is an 'Add rule' button and a 'Save rules' button at the bottom.

Inbound rules of alb:

The screenshot shows the 'Edit inbound rules' page for a security group rule ID 'sgr-0a5893d22cf839b58'. It contains two rules: one for HTTPS on port 443 from 0.0.0.0/0 and another for HTTP on port 80 from 0.0.0.0/0. A warning message at the bottom advises against using 0.0.0.0/0 or ::/0 as sources. There is an 'Add rule' button and a 'Save rules' button at the bottom.

Creation of RDS subnet group: (grouping 2 private subnet)

The screenshot shows the 'task_db_subnet' subnet group details. It includes a VPC ID 'vpc-09e68f4610fb4699c', an ARN 'arn:aws:rds:ap-south-2:969084114868:subgrp:task_db_subnet', and a description 'DB subnet'. The 'Subnets (2)' section lists two subnets: 'Private_subnet2' in 'ap-south-2b' and 'Private_subnet' in 'ap-south-2a', both with CIDR blocks '10.0.3.0/24' and '10.0.2.0/24' respectively. The left sidebar shows navigation options for Aurora and RDS.

Creation of database:

The screenshot shows the AWS Aurora and RDS console. In the left sidebar, under 'Databases', 'task-db' is selected. The main area displays the 'Summary' of the database 'task-db'. Key details include:

- DB identifier:** task-db
- Status:** Available
- Role:** Instance
- Engine:** MySQL Community
- Region & AZ:** ap-south-2b

The CPU usage is shown as 3.38%. Below the summary, there are tabs for 'Connectivity & security', 'Monitoring', 'Logs & events', 'Configuration', 'Zero-ETL integrations', 'Maintenance & backups', and 'Data'.

Creation of target group:

The screenshot shows the AWS EC2 Target Groups console. In the left sidebar, under 'Target groups', 'mytarget' is selected. The main area displays the 'Details' of the target group 'mytarget'. Key details include:

- Target type:** Instance
- Protocol:** Port
- Protocol version:** HTTP: 3000
- VPC:** vpc-09e68f4610fb4699c

The target group contains one target, which is healthy. The distribution of targets by Availability Zone (AZ) is shown in a table. Below the table, there are tabs for 'Targets', 'Monitoring', 'Health checks', 'Attributes', and 'Tags'.

Creation of load balancer:

The screenshot shows the AWS CloudWatch Metrics interface. At the top, there's a search bar and navigation links for EC2, S3, IAM, VPC, and Aurora and RDS. Below the navigation, there's a breadcrumb trail: EC2 > Load balancers > tasks-alb. On the left, a sidebar menu includes Network & Security, Load Balancing, and Auto Scaling sections. The main content area displays a table for the 'tasks-alb' load balancer, showing details like Status (Active), Scheme (Internet-facing), Hosted zone (Z0173938T07WNTVAEPZN), and Availability Zones (ap-south-2a, ap-south-2b). It also shows the Load balancer ARN and DNS name. Below the table are tabs for Listeners and rules, Network mapping, Resource map, Security, Monitoring, Integrations, Attributes, Capacity, and Tags.

Creation of instance:

The screenshot shows the AWS CloudWatch Metrics interface. At the top, there's a search bar and navigation links for EC2, S3, IAM, VPC, and Aurora and RDS. Below the navigation, there's a breadcrumb trail: EC2 > Instances. On the left, a sidebar menu includes Instances, Images, and Elastic Block Store sections. The main content area displays a table for instances, showing two entries: 'Frontend_ec2' (Instance ID: i-004d0eb35630c66ab, State: Running) and 'Backend_ec2' (Instance ID: i-0a326bd05677e1bba, State: Running). Below the table, there's a detailed view for the 'Backend_ec2' instance, showing its instance ID, Public IPv4 address (16.112.31.163), and Private IPv4 addresses (10.0.1.45).

Frontend:

COMMANDS USED IN FRONTEND EC2:

Apt update -y

Apt install nginx

Var/www/html/ (create index.html file)

Systemctl start nginx

```
aws CloudShell Search [Alt+S] Asia Pacific (Hyderabad) Rubika vajiravelu (9690-B411-4868) Rubika vajiravelu

EC2 S3 IAM VPC Aurora and RDS

System load: 0.09 Processes: 113
Usage of /: 37.3% of 7.57GB Users logged in: 0
Memory usage: 23% IPv4 address for ens5: 10.0.1.45
Swap usage: 0%


Expanded Security Maintenance for Applications is not enabled.

10 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

Last login: Wed Jan 14 06:56:29 2026 from 18.60.252.248
ubuntu@ip-10-0-1-45:~$ sudo su
root@ip-10-0-1-45:/home/ubuntu# cd ..
root@ip-10-0-1-45:/home# cd ..
root@ip-10-0-1-45:~/ ls
bin boot dev etc home lib lib32 lib64 libx32 lost+found media mnt opt proc root run sbin snap srv sys usr var
root@ip-10-0-1-45:~/ cd var
root@ip-10-0-1-45:/var# cd www
root@ip-10-0-1-45:/var/www# cd html
root@ip-10-0-1-45:/var/www/html# ls
index.html index.nginx-debian.html
root@ip-10-0-1-45:/var/www/html# [REDACTED]
```

i-004d0eb35630c66ab (Frontend_ec2)
PublicIPs: 16.112.31.163 PrivateIPs: 10.0.1.45
CloudShell Feedback Console Mobile App © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

```
root@ip-10-0-1-45:/var/www/html# systemctl start nginx
root@ip-10-0-1-45:/var/www/html# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2026-01-15 08:50:53 UTC; 4min 10s ago
       Docs: man:nginx(8)
   Process: 388 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Process: 471 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
 Main PID: 473 (nginx)
   Tasks: 3 (limit: 1073)
  Memory: 12.5M
    CPU: 64ms
   CGroup: /system.slice/nginx.service
           ├─473 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
           ├─474 "nginx: worker process" * * * * *
           └─475 "nginx: worker process" * * * * *

Jan 15 08:50:52 ip-10-0-1-45 systemd[1]: Starting A high performance web server and a reverse proxy server...
Jan 15 08:50:53 ip-10-0-1-45 systemd[1]: Started A high performance web server and a reverse proxy server.
root@ip-10-0-1-45:/var/www/html# [REDACTED]
```

i-004d0eb35630c66ab (Frontend_ec2)
PublicIPs: 16.112.31.163 PrivateIPs: 10.0.1.45
CloudShell Feedback Console Mobile App © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Backend:

COMMANDS USED IN BACKEND EC2:

Mkdir Task_api(folder name)

Cd Task_api

apt install nodejs npm -y

npm install express mysql2 cors

vi app.js (nodejs code)

vi db.js (database code)

node app.js

System information as of Thu Jan 15 08:54:35 UTC 2026

```
System load: 0.08      Processes: 109
Usage of /: 38.1% of 7.57GB  Users logged in: 0
Memory usage: 22%          IPv4 address for ens5: 10.0.2.81
Swap usage: 0%            
```

Expanded Security Maintenance for Applications is not enabled.

10 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at <https://ubuntu.com/esm>

```
Last login: Wed Jan 14 06:59:05 2026 from 10.0.2.223
ubuntu@ip-10-0-2-81:~$ sudo su
root@ip-10-0-2-81:/home/ubuntu# ls
Task_api
root@ip-10-0-2-81:/home/ubuntu# cd Task_api
root@ip-10-0-2-81:/home/ubuntu/Task_api# ls
app.js  db.js  node_modules  package-lock.json  package.json
root@ip-10-0-2-81:/home/ubuntu/Task_api# node app.js
Backend running on port 3000
Connected to RDS MySQL Database
```

i-0a326bd05677e1bba (Backend_ec2)

PrivateIP: 10.0.2.81

CloudShell Feedback Console Mobile App

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Connecting the database:

mysql -h task-db.cd4me4ya4pn5.ap-south-2.rds.amazonaws.com -u admin -p (Connect the mysql database)

```
10 updates can be applied immediately.
To see these additional updates run: apt list --upgradable
1 additional security update can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm

New release '24.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Thu Jan 15 08:54:36 2026 from 10.0.2.223
ubuntu@ip-10-0-2-81:~$ sudo su
root@ip-10-0-2-81:/home/ubuntu# mysql -h task-db.cd4me4ya4pn5.ap-south-2.rds.amazonaws.com -u admin -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1183
Server version: 8.0.43 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> 
```

i-0a326bd05677e1bba (Backend_ec2)

PrivateIP: 10.0.2.81

CloudShell Feedback Console Mobile App

© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws Search [Alt+S] Asia Pacific (Hyderabad) Rubika vajiravelu (9690-8411-4868) Rubika vajiravelu

EC2 S3 IAM VPC Aurora and RDS

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| mysql |
| performance_schema |
| sys |
| taskdb |
+-----+
5 rows in set (0.02 sec)

mysql> USE taskdb;
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_taskdb |
+-----+
| tasks |
+-----+
1 row in set (0.00 sec)

mysql> 
```

i-0a326bd05677e1bba (Backend_ec2)

PrivateIPs: 10.0.2.81

CloudShell Feedback Console Mobile App © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

aws Search [Alt+S] Asia Pacific (Hyderabad) Rubika vajiravelu (9690-8411-4868) Rubika vajiravelu

EC2 S3 IAM VPC Aurora and RDS

```
5 rows in set (0.02 sec)

mysql> USE taskdb;
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_taskdb |
+-----+
| tasks |
+-----+
1 row in set (0.00 sec)

mysql> SELECT * FROM tasks;
+----+----+----+----+----+----+----+
| id | title | description | priority | assigned_to | status | created_at |
+----+----+----+----+----+----+----+
| 1 | aws project | deploy | Medium | Ruby | In Progress | 2026-01-13 17:53:00 |
| 2 | ec2 with lambda | lambda | High | Ranji | Completed | 2026-01-13 18:15:34 |
| 3 | Database backup | snapshot | High | Jacky | Pending | 2026-01-14 04:43:24 |
| 4 | ec2-server | reshma777 | Medium | Reshma | In Progress | 2026-01-14 06:57:42 |
+----+----+----+----+----+----+----+
4 rows in set (0.00 sec)

mysql> 
```

i-0a326bd05677e1bba (Backend_ec2)

PrivateIPs: 10.0.2.81

CloudShell Feedback Console Mobile App © 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Output of frontend:

The screenshot shows a web application titled "Cloud Task Manager". On the left, there is a "Create New Task" form with fields for Task Title, Description, Priority (set to Low), Assigned To, and Status (set to Pending). A green "Add Task" button is at the bottom. On the right, there is a "Task Dashboard" table with the following data:

ID	Title	Priority	Assigned	Status
5	Creation of S3	High	Srinidhi	Completed
4	ec2-server	Medium	Reshma	In Progress
3	Database backup	High	Jacky	Pending
2	ec2 with lambda	High	Ranji	Completed
1	aws project	Medium	Ruby	In Progress

Output of backend: - alb:

```
Pretty-print 
[
  {
    "id": 5,
    "title": "Creation of S3",
    "description": "For log details",
    "priority": "High",
    "assigned_to": "Srinidhi",
    "status": "Completed",
    "created_at": "2026-01-15T09:03:56.000Z"
  },
  {
    "id": 4,
    "title": "ec2-server",
    "description": "reshma7777",
    "priority": "Medium",
    "assigned_to": "Reshma",
    "status": "In Progress",
    "created_at": "2026-01-14T06:57:42.000Z"
  },
  {
    "id": 3,
    "title": "Database backup",
    "description": "snapshot",
    "priority": "High",
    "assigned_to": "Jacky",
    "status": "Pending",
    "created_at": "2026-01-14T04:43:24.000Z"
  },
  {
    "id": 2,
    "title": "ec2 with lambda",
    "description": "Lambda",
    "priority": "High",
    "assigned_to": "Ranji",
    "status": "Completed",
    "created_at": "2026-01-13T18:15:34.000Z"
  },
  {
    "id": 1,
    "title": "aws project",
    "description": "deploy",
    "priority": "Medium",
    "assigned_to": "Ruby",
    "status": "In Progress",
    "created_at": "2026-01-13T17:53:00.000Z"
  }
]
```

Index.html:

```
<!DOCTYPE html>
<html>
<head>
<title>Cloud Task Manager</title>
<meta name="viewport" content="width=device-width, initial-scale=1.0">

<style>
  * {
```

```
    box-sizing: border-box;
}

body {
  margin: 0;
  font-family: "Times new roman", sans-serif;
  background: linear-gradient(to right, #74ebd5, #acb6e5);
  min-height: 100vh;
}

header {
  background: linear-gradient(to right, #667eea, #764ba2);
  padding: 18px;
  color: white;
  text-align: center;
  font-size: 28px;
  font-weight: bold;
}

.container {
  max-width: 1100px;
  margin: 30px auto;
  padding: 10px;
  display: grid;
  grid-template-columns: 1fr 2fr;
  gap: 20px;
}

.card {
  background: white;
  border-radius: 12px;
  padding: 20px;
  box-shadow: 0px 5px 15px rgba(0,0,0,0.2);
}

h2 {
  margin-top: 0;
  color: #444;
}

.form-group {
  margin-bottom: 12px;
}

label {
  font-size: 14px;
  font-weight: 600;
  color: #555;
```

```
display: block;
margin-bottom: 4px;
}

input, select, textarea {
width: 100%;
height: 42px;
padding: 8px 10px;
border-radius: 6px;
border: 1px solid #ccc;
font-size: 14px;
}

textarea {
resize: none;
height: 80px;
}

button {
margin-top: 15px;
padding: 12px;
width: 100%;
border: none;
border-radius: 25px;
font-size: 16px;
background: linear-gradient(to right, #43cea2, #185a9d);
color: white;
cursor: pointer;
transition: 0.3s;
}

button:hover {
opacity: 0.85;
}

table {
width: 100%;
border-collapse: collapse;
}

th {
background: #667eea;
color: white;
padding: 10px;
}

td {
padding: 10px;
```

```
text-align: center;
border-bottom: 1px solid #eee;
font-size: 14px;
}

tr:hover {
background: #f1f1f1;
}

footer {
text-align: center;
padding: 15px;
color: #333;
font-size: 14px;
}

@media(max-width: 900px) {
.container {
grid-template-columns: 1fr;
}
}

</style>
</head>

<body>

<header>
Cloud Task Manager
</header>

<div class="container">

<!-- Form Card -->
<div class="card">
<h2>Create New Task</h2>

<div class="form-group">
<label>Task Title</label>
<input id="title" placeholder="Enter task title">
</div>

<div class="form-group">
<label>Description</label>
<textarea id="description" placeholder="Enter task description"></textarea>
</div>

<div class="form-group">
<label>Priority</label>
```

```
<select id="priority">
<option>Low</option>
<option>Medium</option>
<option>High</option>
</select>
</div>

<div class="form-group">
<label>Assigned To</label>
<input id="assigned_to" placeholder="Employee name">
</div>

<div class="form-group">
<label>Status</label>
<select id="status">
<option>Pending</option>
<option>In Progress</option>
<option>Completed</option>
</select>
</div>

<button onclick="addTask()">Add Task</button>
</div>

<!-- Table Card -->
<div class="card">
<h2>Task Dashboard</h2>
<table>
<thead>
<tr>
<th>ID</th>
<th>Title</th>
<th>Priority</th>
<th>Assigned</th>
<th>Status</th>
</tr>
</thead>
<tbody id="taskTable"></tbody>
</table>
</div>

</div>

<footer>
  Built on AWS 3-Tier Architecture
</footer>

<script>
```

```

const API_URL = "http://tasks-alb-532214453.ap-south-2.elb.amazonaws.com/tasks"; // Replace this

function addTask() {
  const task = {
    title: document.getElementById("title").value,
    description: document.getElementById("description").value,
    priority: document.getElementById("priority").value,
    assigned_to: document.getElementById("assigned_to").value,
    status: document.getElementById("status").value
  };
  fetch(API_URL, {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify(task)
  }).then(() => {
    loadTasks();
    document.getElementById("title").value = "";
    document.getElementById("description").value = "";
    document.getElementById("assigned_to").value = "";
  });
}

function loadTasks() {
  fetch(API_URL)
    .then(res => res.json())
    .then(data => {
      const table = document.getElementById("taskTable");
      table.innerHTML = "";

      data.forEach(t => {
        table.innerHTML += `
          <tr>
            <td>${t.id}</td>
            <td>${t.title}</td>
            <td>${t.priority}</td>
            <td>${t.assigned_to}</td>
            <td>${t.status}</td>
          </tr>`;
      });
    });
}

loadTasks();
</script>

</body>
</html>

```

App.js:

```
const express = require("express");
const cors = require("cors");
const db = require("./db");

const app = express();
app.use(express.json());
app.use(cors());

/***
 * POST - Add New Task
 */
app.post("/tasks", (req, res) => {
  const { title, description, priority, assigned_to, status } = req.body;

  if (!title || !priority || !status) {
    return res.status(400).json({ error: "Missing required fields" });
  }

  const sql = `
    INSERT INTO tasks (title, description, priority, assigned_to, status)
    VALUES (?, ?, ?, ?, ?)
  `;

  db.query(
    sql,
    [title, description, priority, assigned_to, status],
    (err, result) => {
      if (err) {
        console.error("Insert error:", err);
        return res.status(500).json({ error: "Database insert failed" });
      }

      res.json({
        message: "✓ Task added successfully",
        id: result.insertId
      });
    }
  );
});

/***
 * GET - Fetch All Tasks
 */
app.get("/tasks", (req, res) => {
  const sql = "SELECT * FROM tasks ORDER BY created_at DESC";
})
```

```

db.query(sql, (err, results) => {
  if (err) {
    console.error("Fetch error:", err);
    return res.status(500).json({ error: "Database fetch failed" });
  }

  res.json(results);
});

});

```

// Server start

```

const PORT = 3000;
app.listen(PORT, () => {
  console.log(`🚀 Backend running on port ${PORT}`);
});

```

Db.js:

```

const mysql = require("mysql2");

// Database connection

const db = mysql.createConnection({
  host: "task-db.cd4me4ya4pn5.ap-south-2.rds.amazonaws.com",
  user: "admin",
  password: "Task#312004",
  database: "taskdb",
  port: 3306
});

// Connect to database

db.connect((err) => {
  if (err) {
    console.error("🔴 Database connection failed:", err);
    return;
  }

  console.log("✅ Connected to RDS MySQL Database");
});

module.exports = db;

```