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.

```
User Browser
      |
      v
Frontend EC2 (Public Subnet - Nginx UI)
      |
      v
Application Load Balancer (Public Subnet)
      |
      v
Backend EC2 (Private Subnet - Node.js API)
      |
      v
Amazon RDS MySQL (Private Subnet)
```

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.

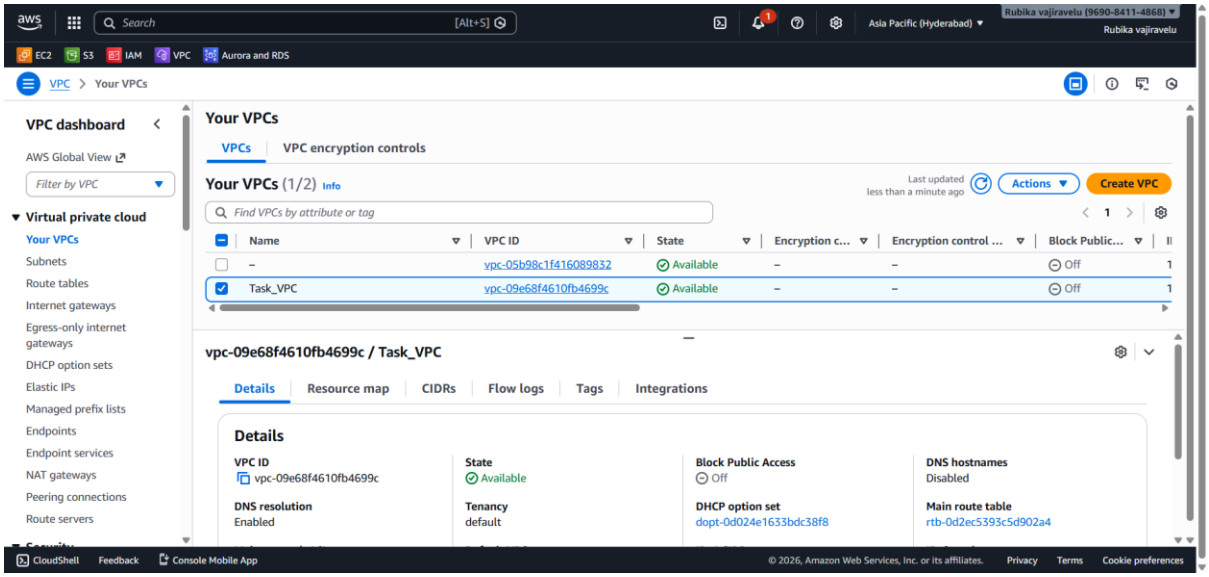
Step 5: Backend Processing

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

Step 6: Database Interaction

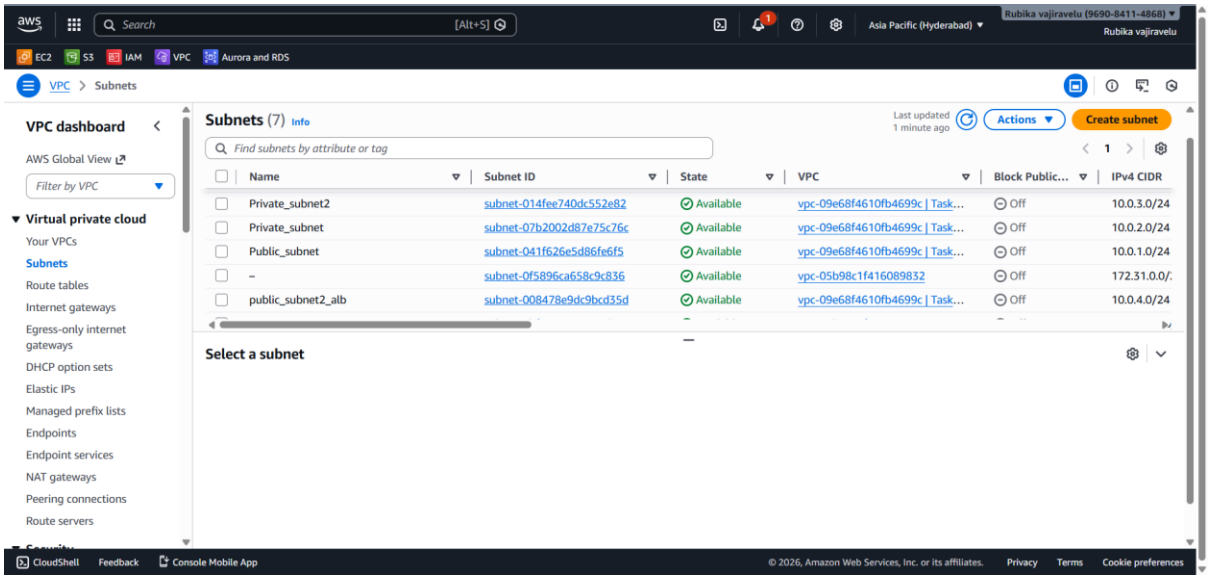
The backend securely connects to the Amazon RDS MySQL database located in a private subnet and stores or retrieves task data.

Creation of VPC:

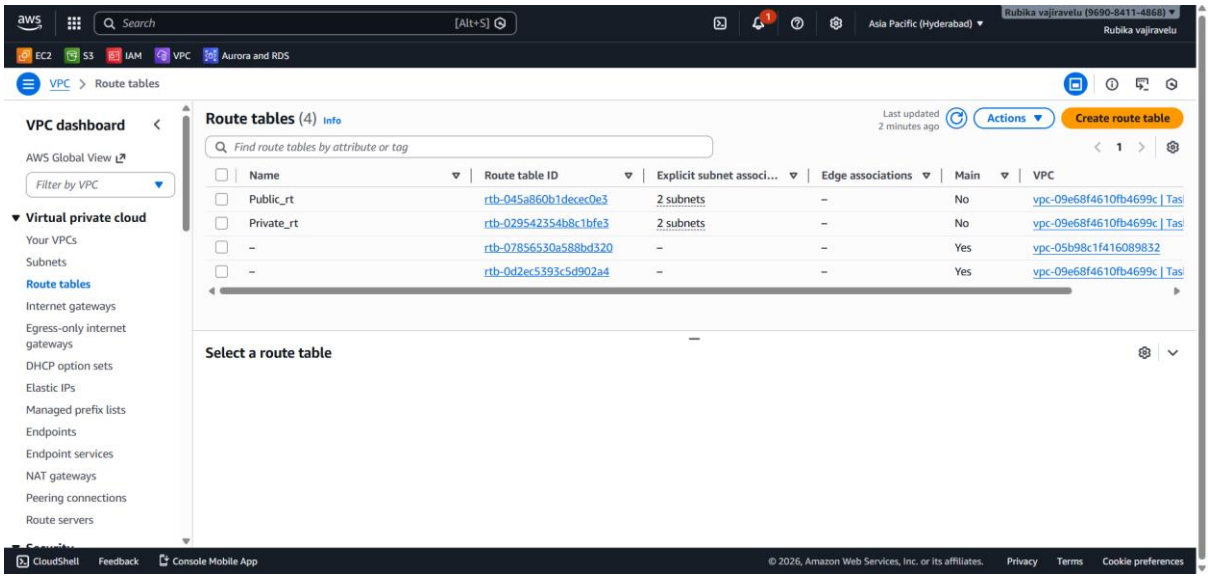


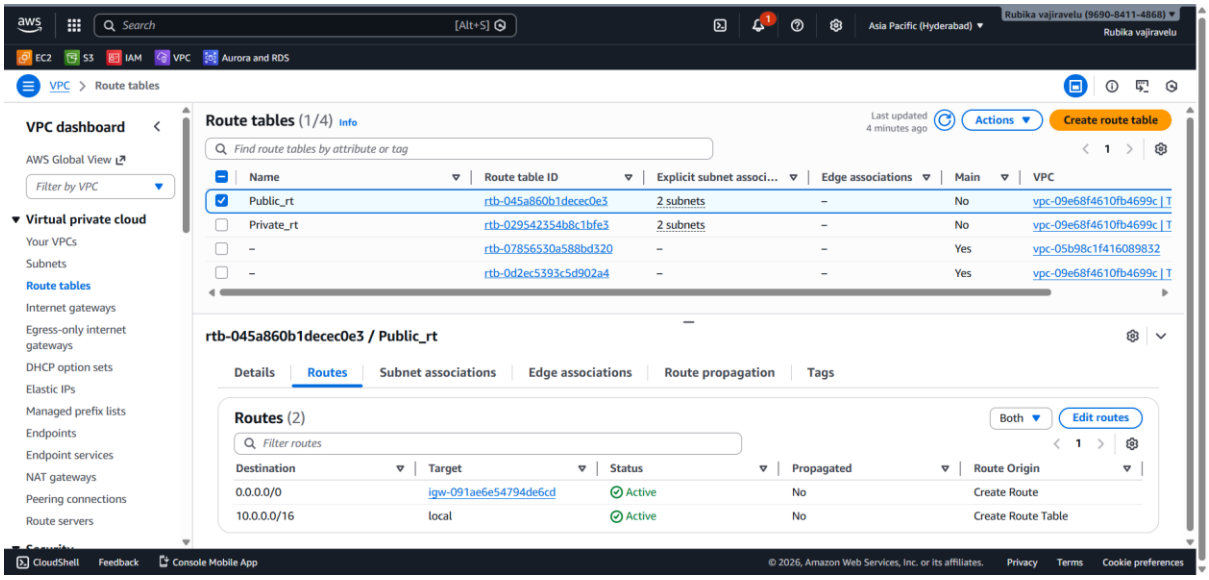
CIDR: 10.0.0.0/16

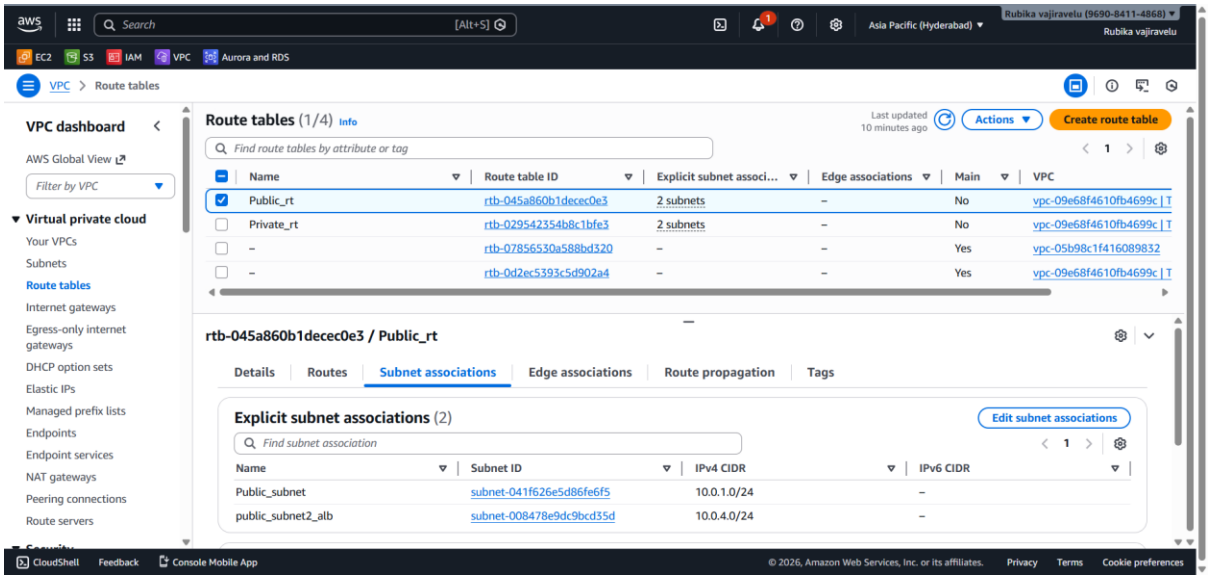
Creation of subnet:

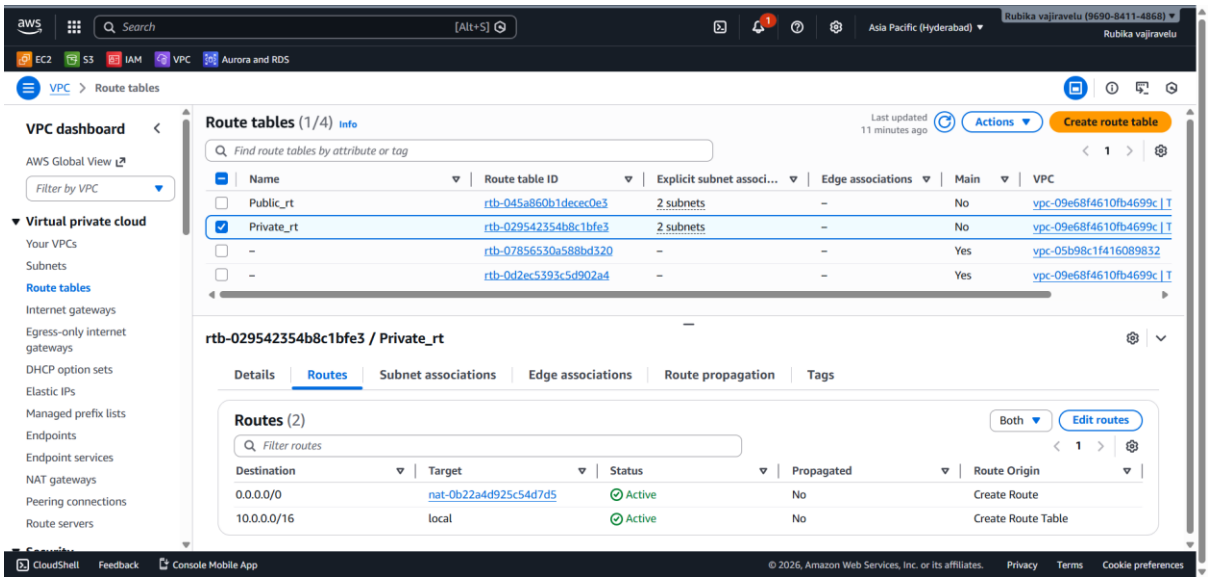


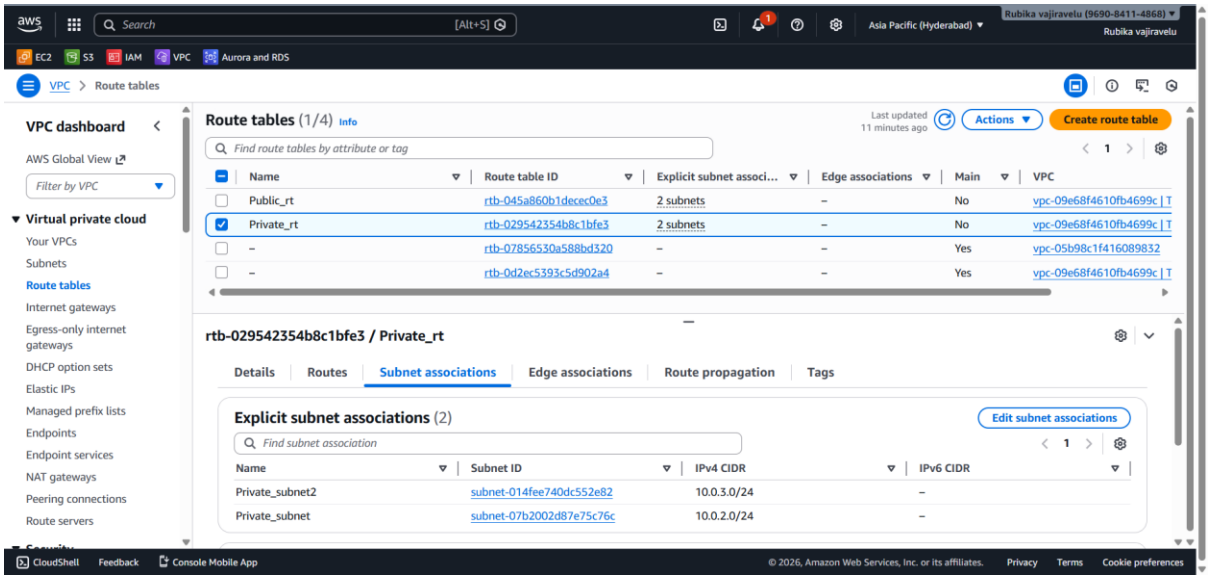
Creation of route table:



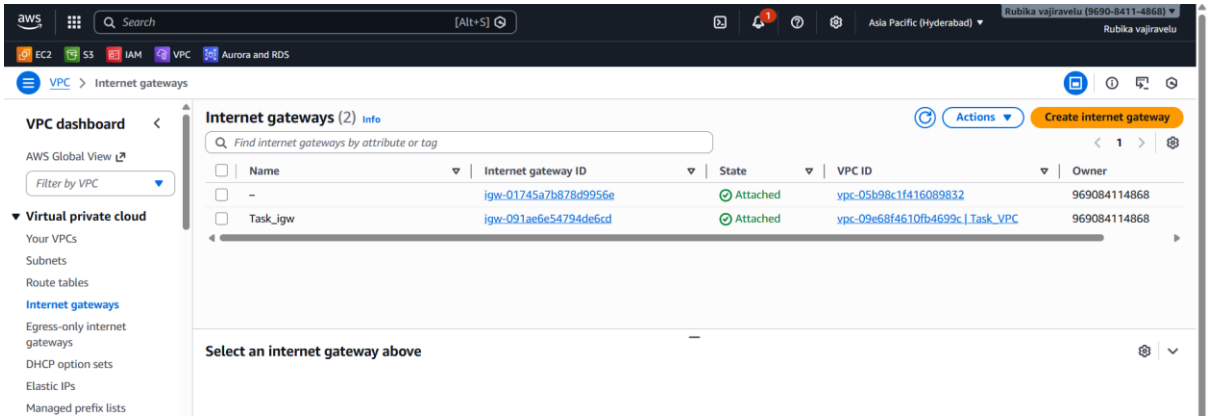

Search [Alt+S]
Asia Pacific (Hyderabad)
Rubika vajiravelu (9690-8411-4868)
Rubika vajiravelu
EC2 S3 IAM VPC Aurora and RDS
VPC > Route tables
VPC dashboard
AWS Global View
Filter by VPC
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Route servers
Route tables (1/4) Info
Last updated 4 minutes ago
Actions
Create route table
Find route tables by attribute or tag
Name | Route table ID | Explicit subnet associ... | Edge associations | Main | VPC
Public_rt | rtb-045a860b1decec0e3 | 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-045a860b1decec0e3 / Public_rt
Details Routes Subnet associations Edge associations Route propagation Tags
Routes (2)
Both
Edit routes
Filter routes
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
CloudShell Feedback Console Mobile App
© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences


Search [Alt+S]
Asia Pacific (Hyderabad)
Rubika vajiravelu (9690-8411-4868)
Rubika vajiravelu
EC2 S3 IAM VPC Aurora and RDS
VPC > Route tables
VPC dashboard
AWS Global View
Filter by VPC
Virtual private cloud
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
Route servers
Route tables (1/4) Info
Last updated 10 minutes ago
Actions
Create route table
Find route tables by attribute or tag
Name | Route table ID | Explicit subnet associ... | Edge associations | Main | VPC
Public_rt | rtb-045a860b1decec0e3 | 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-045a860b1decec0e3 / Public_rt
Details Routes Subnet associations Edge associations Route propagation Tags
Explicit subnet associations (2)
Edit subnet associations
Find subnet association
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 | –
CloudShell Feedback Console Mobile App
© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

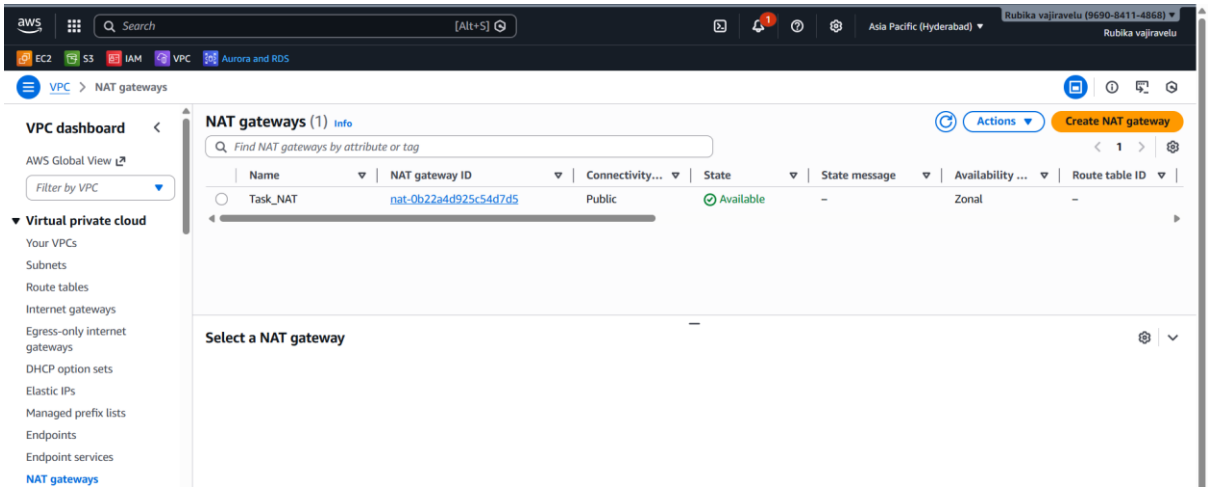

Search [Alt+S]
Asia Pacific (Hyderabad)
Rubika vajiravelu (9690-8411-4868)
Rubika vajiravelu
EC2 S3 IAM VPC Aurora and RDS
VPC > Route tables
VPC dashboard
AWS Global View
Filter by VPC
Virtual private cloud
Your VPCs
Subnets
Route tables
Internet gateways
Egress-only internet gateways
DHCP option sets
Elastic IPs
Managed prefix lists
Endpoints
Endpoint services
NAT gateways
Peering connections
Route servers
Route tables (1/4) Info
Last updated 11 minutes ago
Actions
Create route table
Find route tables by attribute or tag
Name | Route table ID | Explicit subnet associ... | Edge associations | Main | VPC
Public_rt | rtb-045a860b1decec0e3 | 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
Details Routes Subnet associations Edge associations Route propagation Tags
Routes (2)
Both
Edit routes
Filter routes
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
CloudShell Feedback Console Mobile App
© 2026, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences



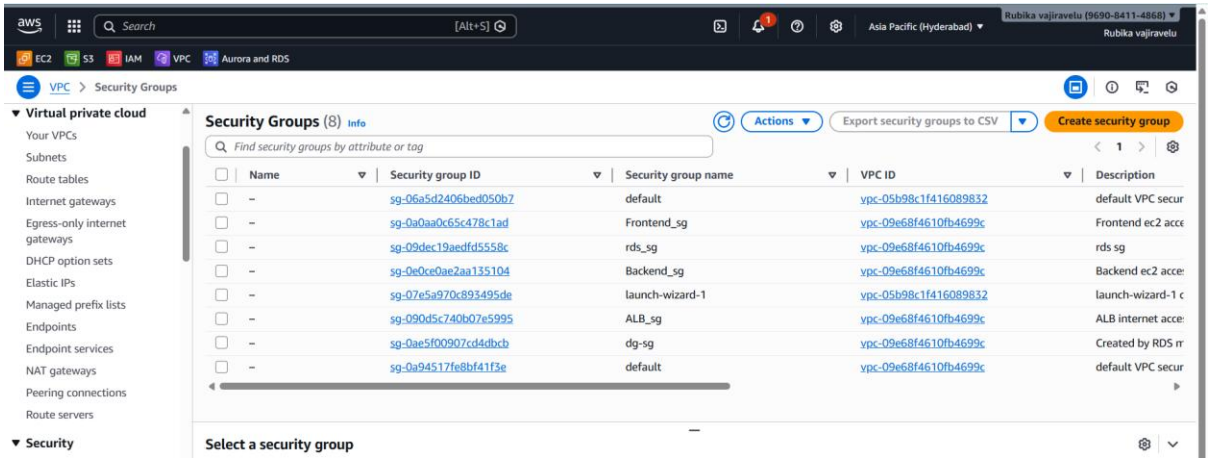
Creation of internet gateway:



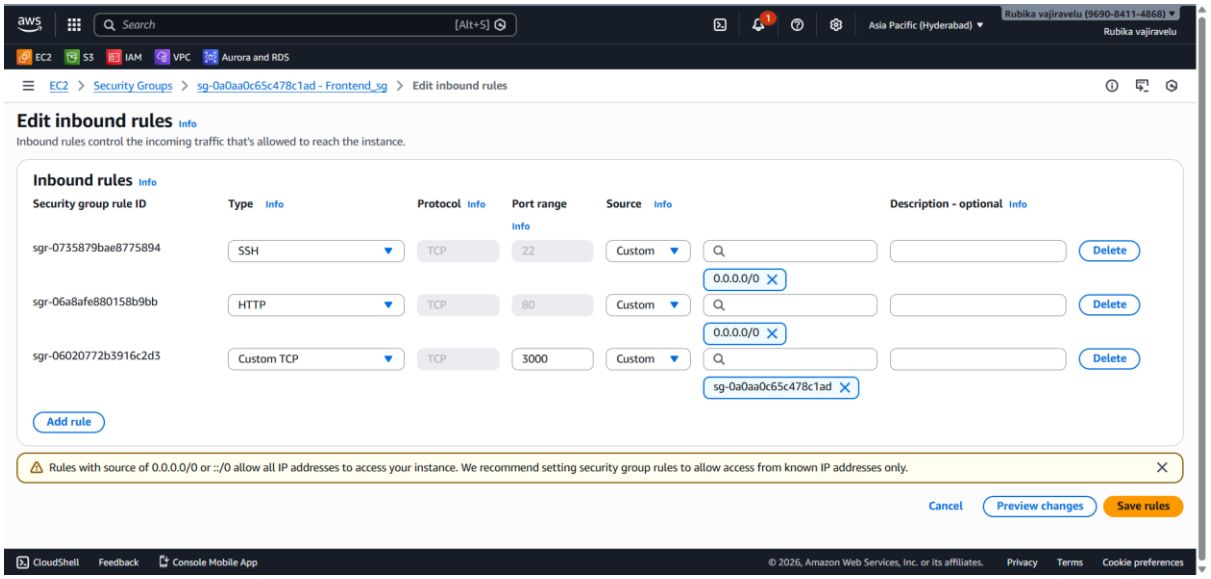
Creation of NAT gateway:



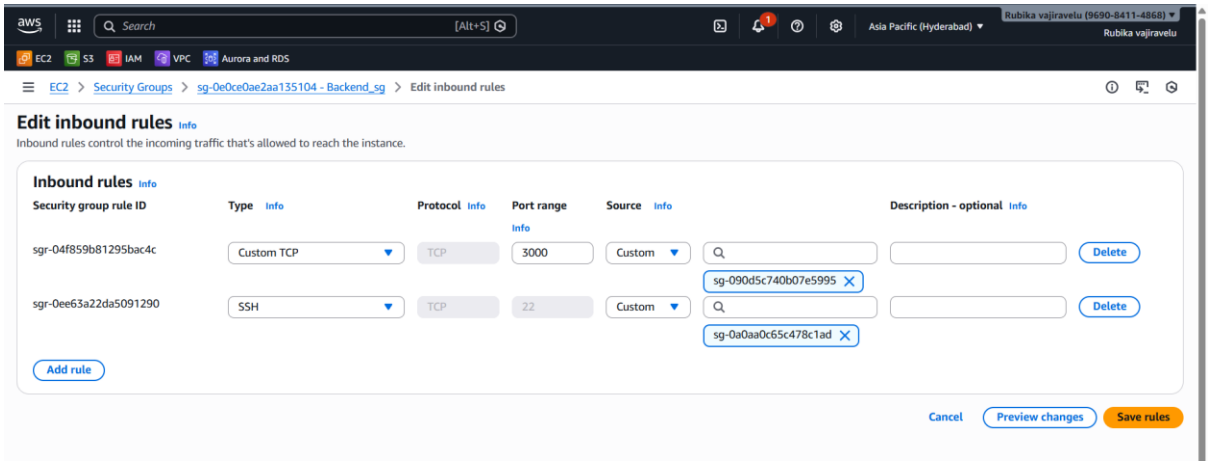
Creation of security group:



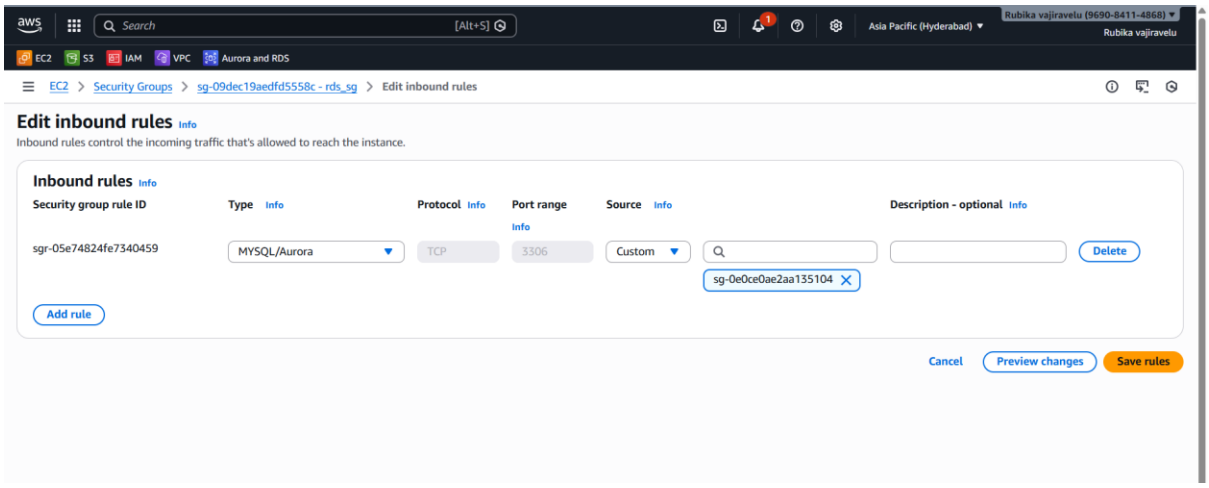
Inbound rules of front_end sg:



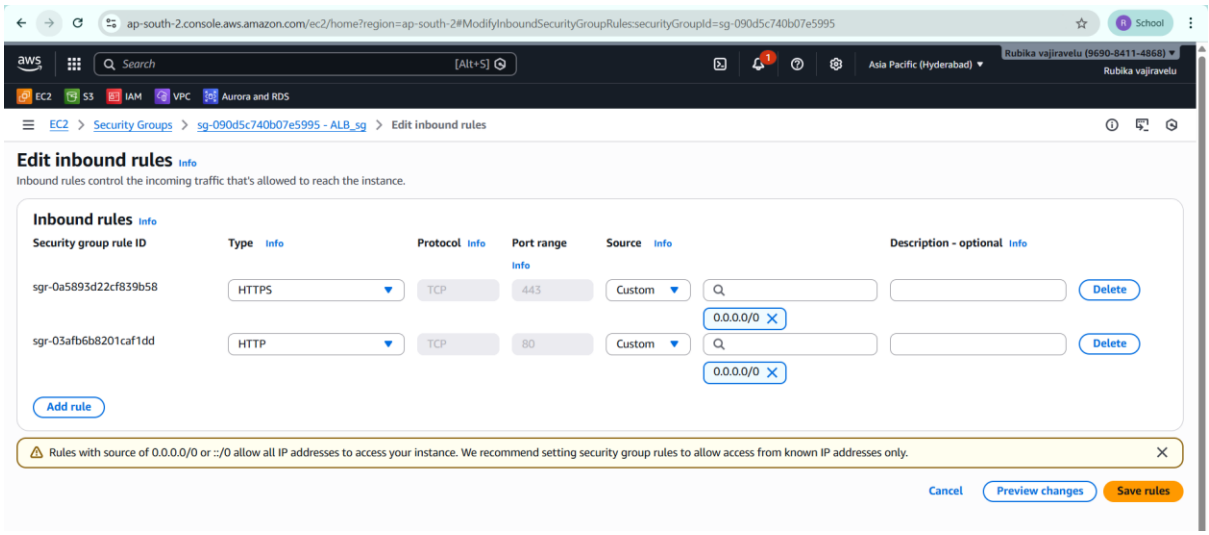
Inbound rules of back_end sg:



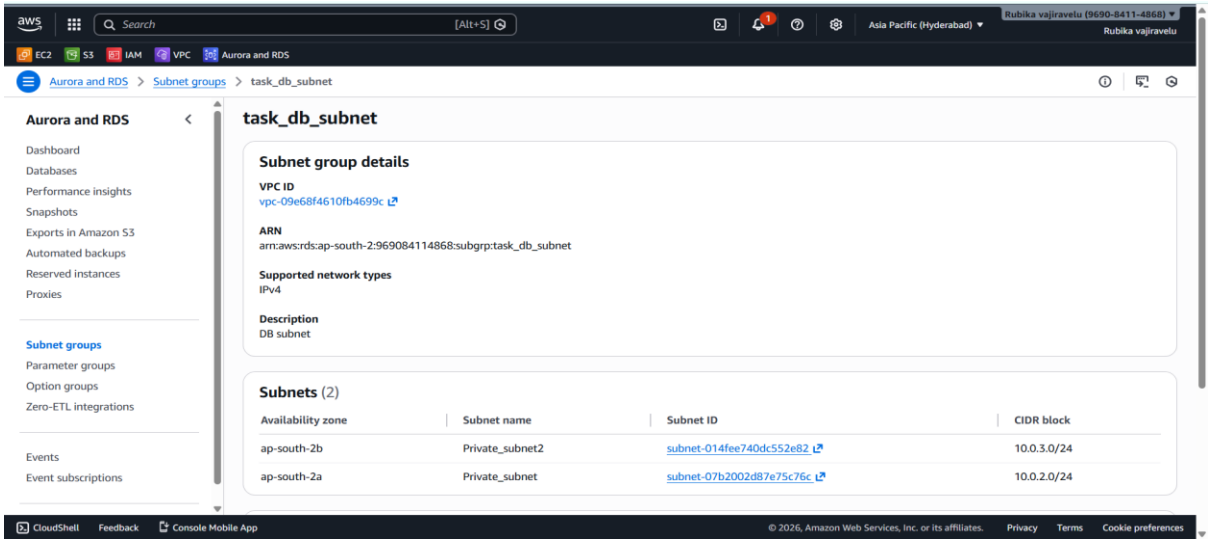
Inbound rules of rds:



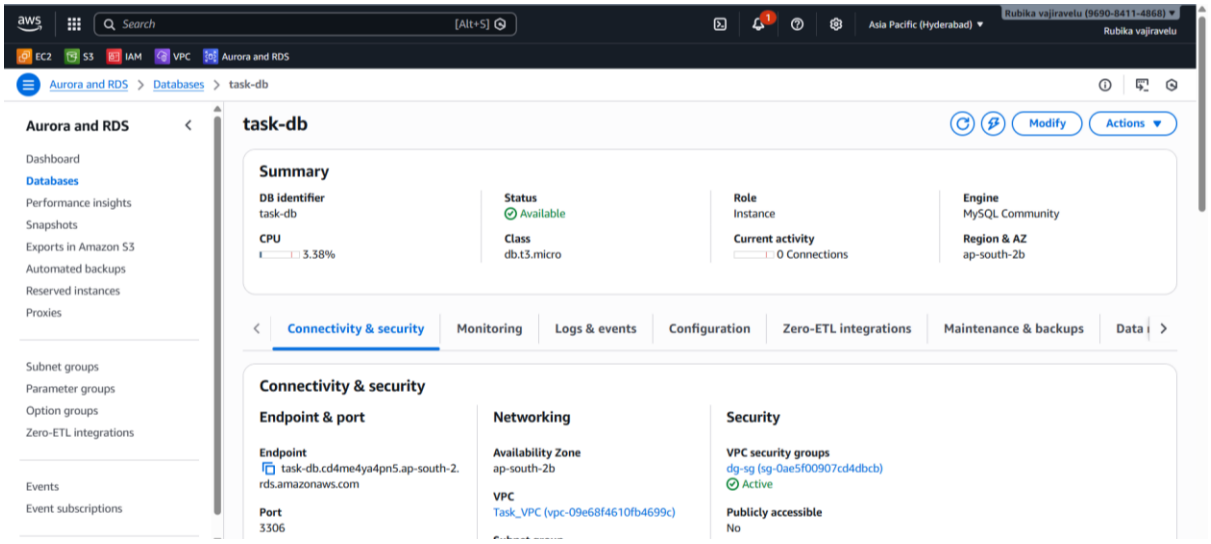
Inbound rules of alb:



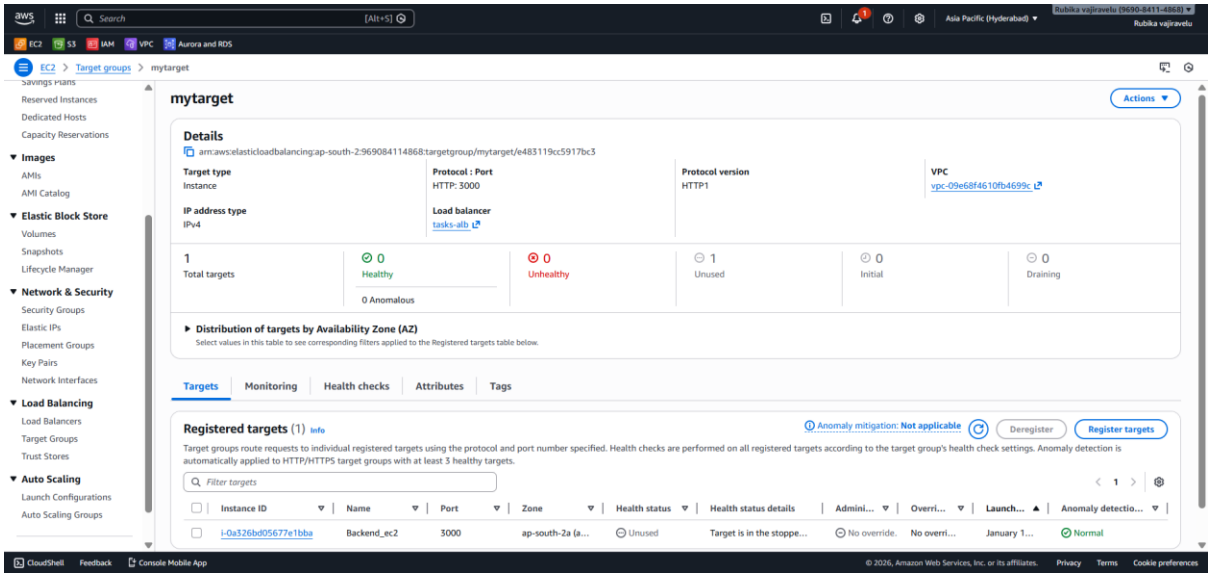
Creation of RDS subnet group: (grouping 2 private subnet)



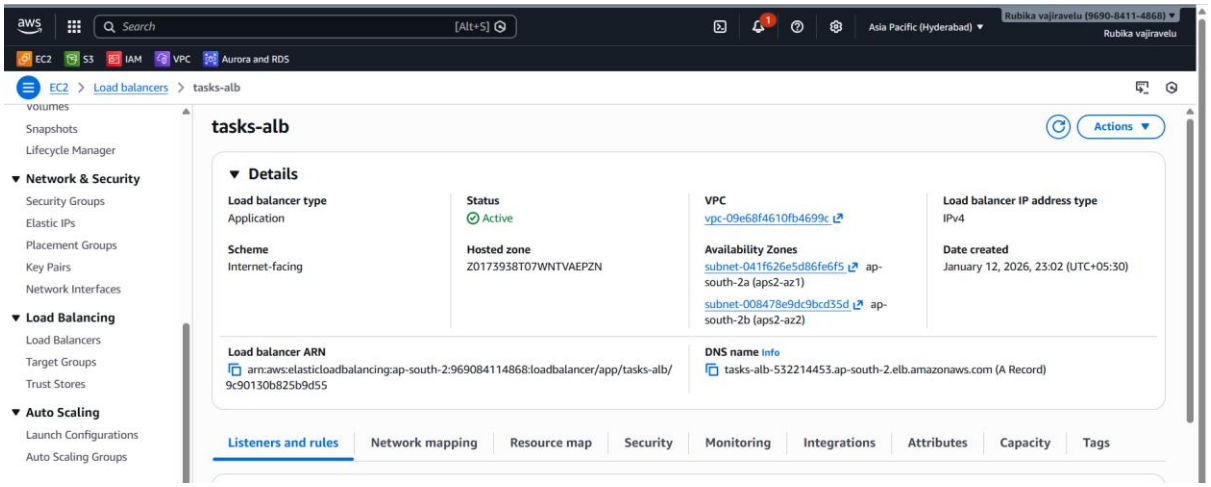
Creation of database:



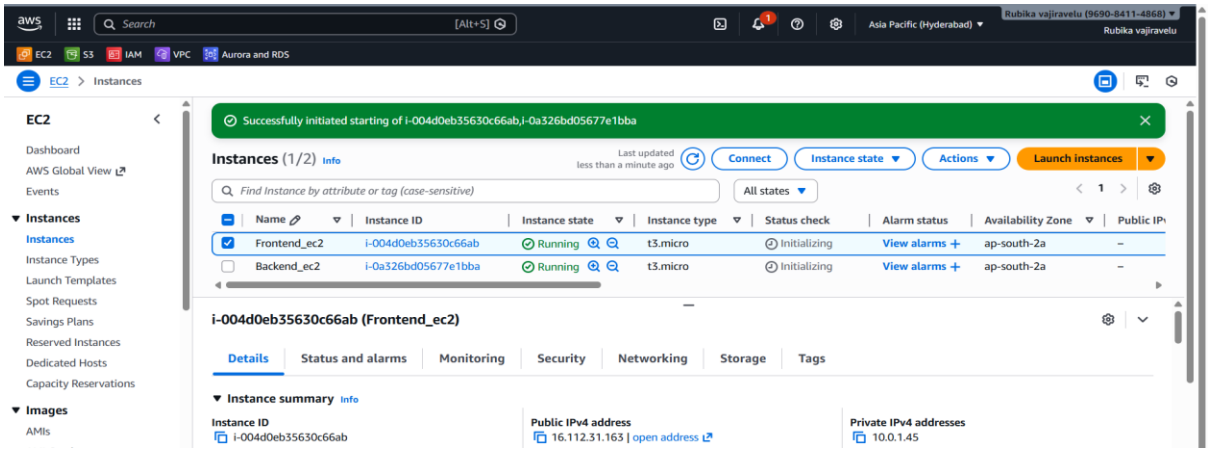
Creation of target group:

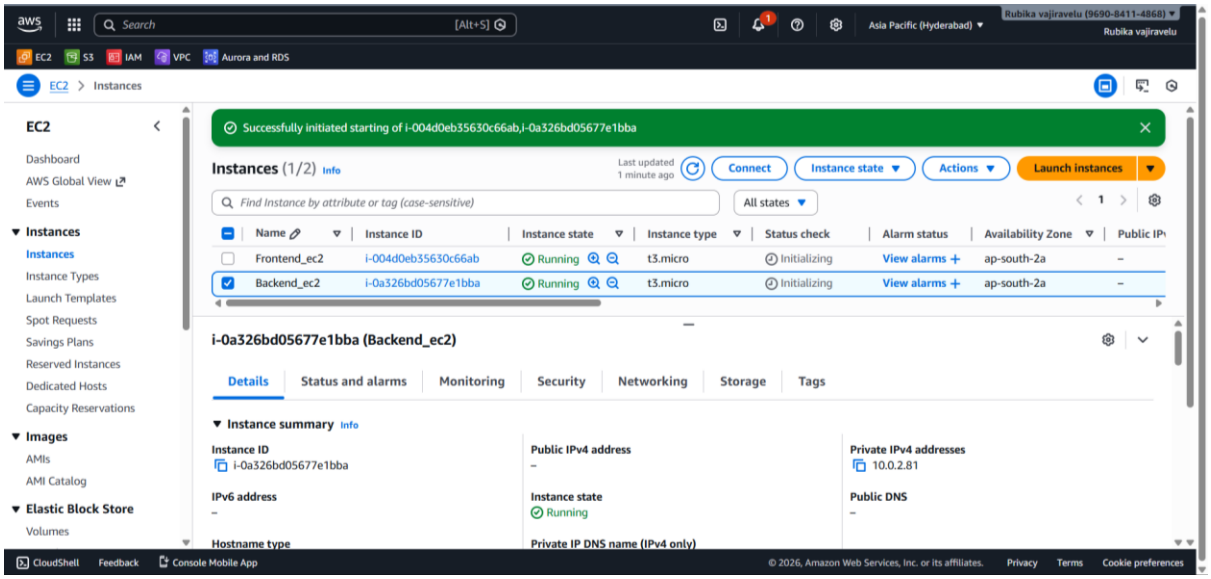


Creation of load balancer:



Creation of instance:

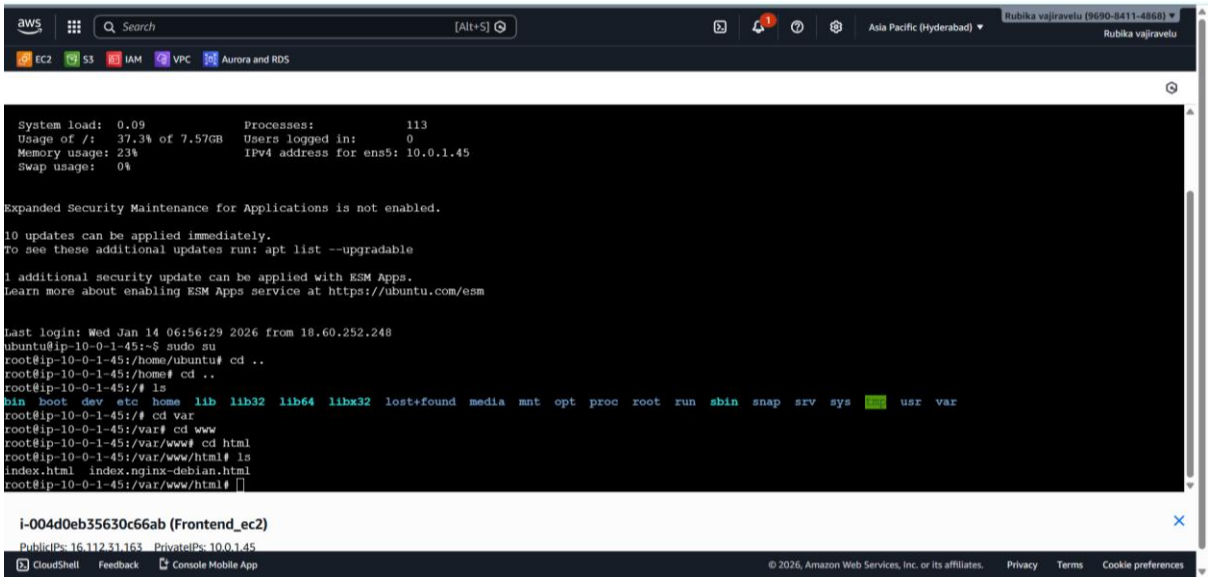


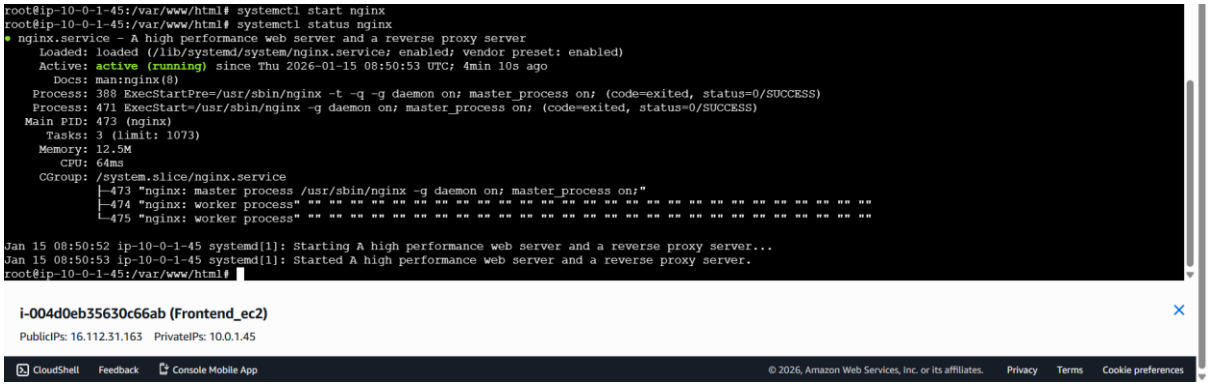


Frontend:

COMMANDS USED IN FRONTEND EC2:

Apt update -y
Apt install nginx
Var/www/html/ (create index.html file)
Systemctl start nginx

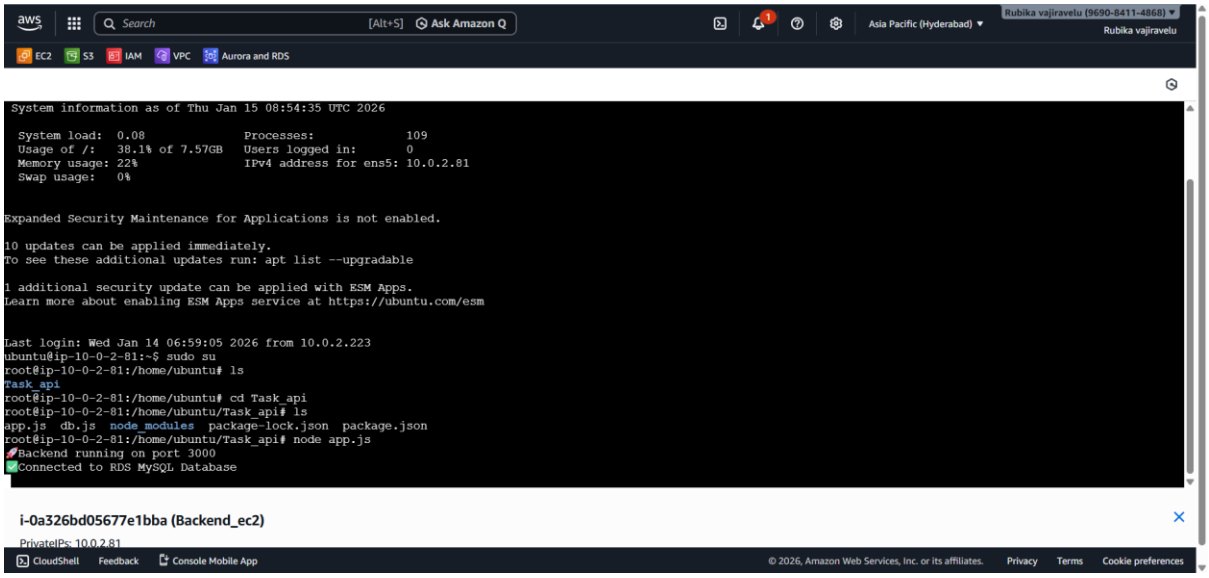




Backend:

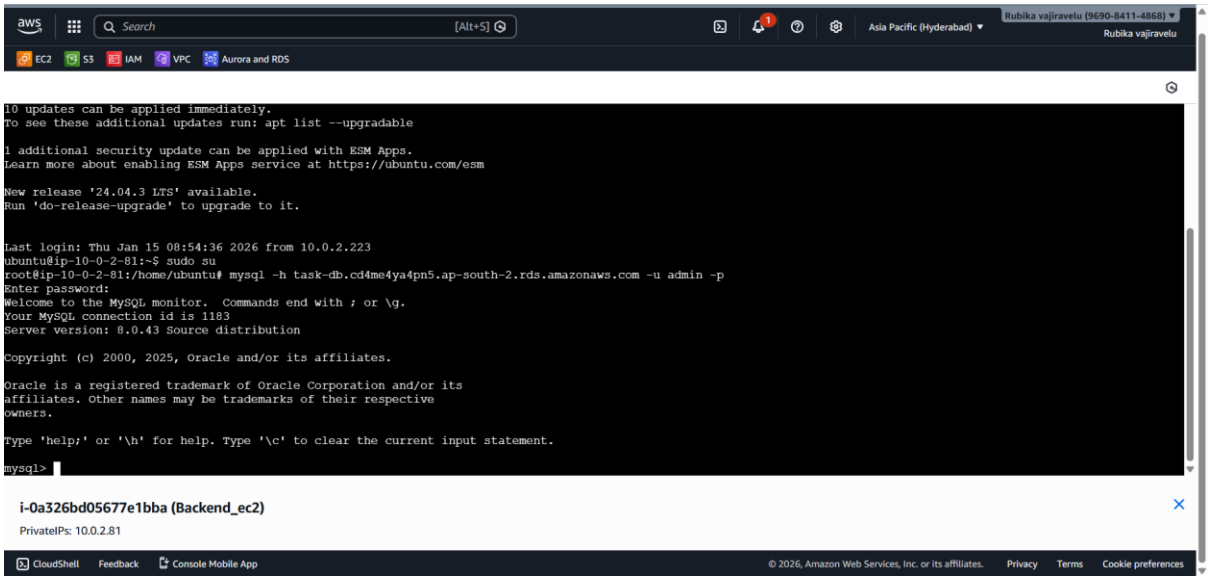
COMMANDS USED IN BACKEND EC2:

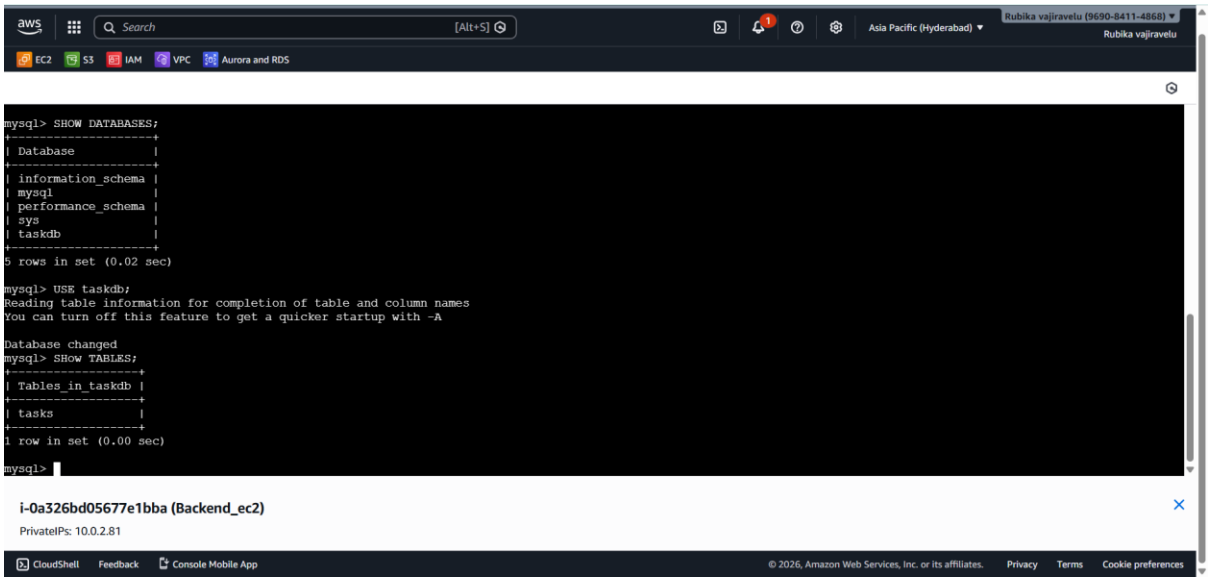
Mkdir Task_api(folder name)
Cd Task_api
apt install nodejs npm -y
npm install express mysql2 cors
vi app.js (nodejs code)
vi db.js (database code)
node app.js

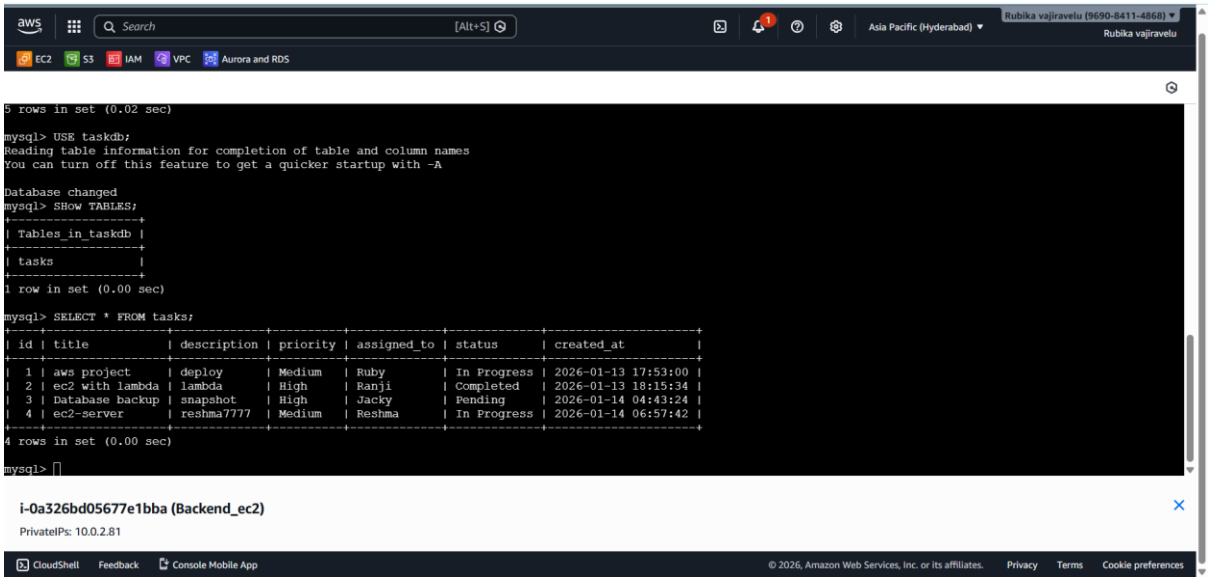


Connecting the database:

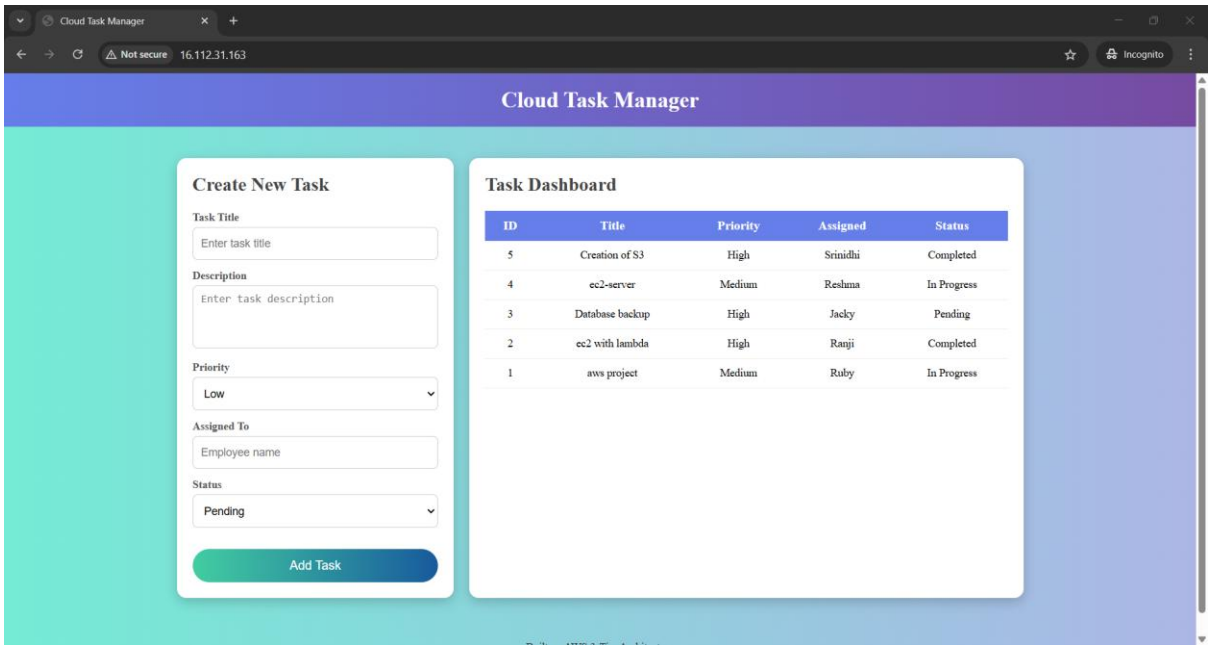
mysql -h task-db.cd4me4ya4pn5.ap-south-2.rds.amazonaws.com -u admin -p (Connect the mysql database)



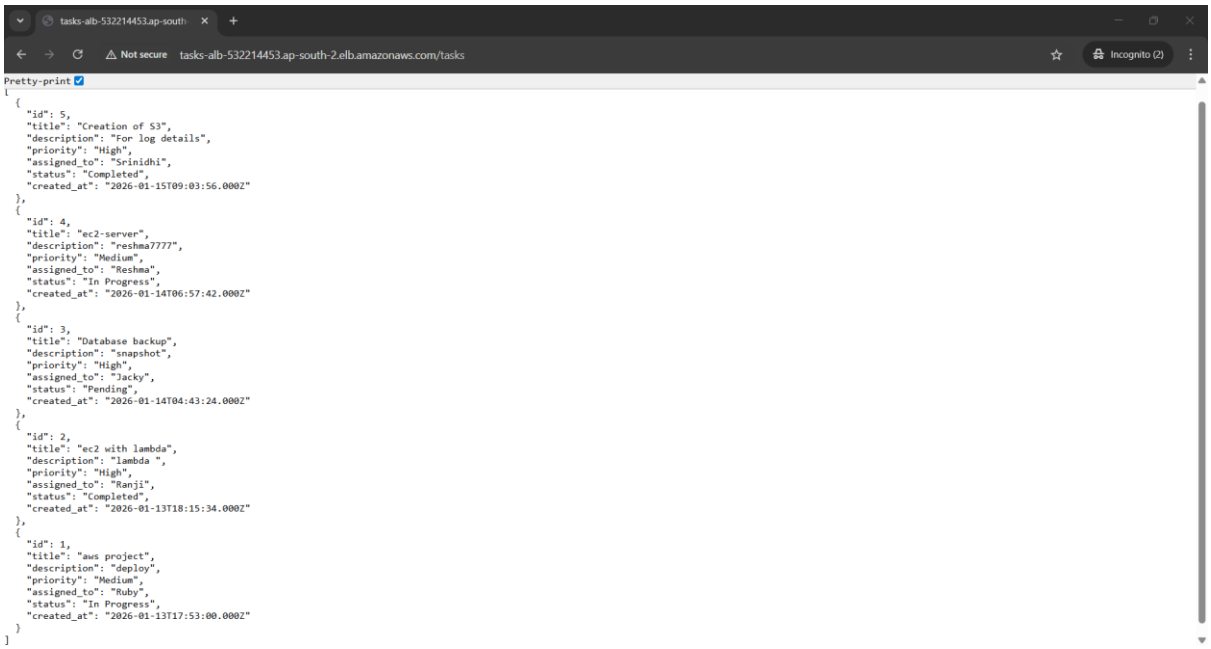

Search
[Alt+S]
Asia Pacific (Hyderabad)
Rubika vajiravelu (9690-8411-4868)
Rubika vajiravelu
EC2
S3
IAM
VPC
Aurora and RDS
mysql> SHOW DATABASES;
+--------------------+
| Database |
+--------------------+
| information_schema |
| mysql |
| performance_schema |
| sys |
| taskdb |
+--------------------+
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


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	reshma7777	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:



Output of backend: - alb:



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;
```