**Food Order App on AWS – Two-Tier Secure Setup**

---

## Step 0 – Prerequisites

- AWS account with a VPC containing public and private subnets.
- Security groups configured for EC2 and RDS.
- Private EC2 for backend (Flask + Python).
- Public EC2 for frontend + proxy (Flask + Python).
- RDS MySQL: database `mayilvaganan`, table `santhos`.

---

## Step 1 – RDS Setup

1. Launch RDS MySQL in a private subnet.
2. Database: `mayilvaganan`, User: `admin`, Password: `santhoskumar999431`.
3. Security group: allow MySQL port 3306 from private EC2 only.
4. Create table:

```
USE mayilvaganan;
CREATE TABLE IF NOT EXISTS santhos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_name VARCHAR(50) NOT NULL,
    item VARCHAR(100) NOT NULL
);
```

---

## Step 2 – Private EC2 Backend

### Install Python & Flask:

```
sudo yum update -y
sudo yum install python3 -y
python3 -m ensurepip
python3 -m pip install flask pymysql flask-cors
```

### app.py

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import pymysql

app = Flask(__name__)
CORS(app)

conn = pymysql.connect(
```

```
    host='database-1.cypu22wkeoit.us-east-1.rds.amazonaws.com',
    user='admin',
    password='santhoskumar999431',
    database='mayilvaganan'
)
cursor = conn.cursor()

cursor.execute("""
CREATE TABLE IF NOT EXISTS santhos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    user_name VARCHAR(50) NOT NULL,
    item VARCHAR(100) NOT NULL
)
""")
conn.commit()

@app.route('/api/order', methods=['POST'])
def order_food():
    data = request.json
    user = data.get('user')
    item = data.get('item')
    if not user or not item:
        return jsonify({"error": "User and item are required"}), 400
    cursor.execute("INSERT INTO santhos (user_name, item) VALUES (%s, %s)",
(user, item))
    conn.commit()
    return jsonify({"message": f"Order received from {user} for {item}"}),
201

@app.route('/api/orders', methods=['GET'])
def get_orders():
    cursor.execute("SELECT * FROM santhos")
    orders = cursor.fetchall()
    result = [{"id": o[0], "user": o[1], "item": o[2]} for o in orders]
    return jsonify(result)

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

**Run backend:**

```
nohup python3 app.py > app.log 2>&1 &
```

---

## Step 3 – Public EC2 Proxy

### Install Python & Packages:

```
sudo yum update -y
sudo yum install python3-pip -y
pip3 install flask requests flask-cors
```

### proxy.py

```
from flask import Flask, request, jsonify
from flask_cors import CORS
import requests

app = Flask(__name__)
CORS(app)

PRIVATE_BACKEND = "http://10.0.131.64:5000/api/order"  # Replace with your
private EC2 IP

@app.route("/api/order", methods=["POST"])
def proxy_order():
    try:
        data = request.json
        res = requests.post(PRIVATE_BACKEND, json=data, timeout=5)
        res.raise_for_status()
        return jsonify(res.json())
    except requests.exceptions.RequestException as e:
        return jsonify({"error": "Failed to reach private backend",
"details": str(e)}), 502
    except ValueError:
        return jsonify({"error": "Private backend returned invalid JSON"}),
502

if __name__ == "__main__":
    app.run(host="0.0.0.0", port=5000)
```

### Run proxy:

```
python3 proxy.py
```

- Ensure Security Group allows **TCP 5000** and **TCP 80**.

---

## Step 4 – Frontend (index.html)

Save in `/var/www/html/index.html`:

```
<!DOCTYPE html>
<html>
<head>
    <title>Food Order App</title>
    <style>
        body { font-family: Arial; background: #f0f0f0; text-align: center;
margin-top: 50px; }
        input { padding: 10px; margin: 5px; width: 200px; }
        button { padding: 10px 20px; background: #4CAF50; color: white;
border: none; cursor: pointer; }
        button:hover { background: #45a049; }
        #response { margin-top: 20px; font-weight: bold; color: blue; }
    </style>
</head>
<body>
    <h1>Food Order App</h1>
    <input type="text" id="user" placeholder="Your Name">
    <input type="text" id="item" placeholder="Food Item">
    <button onclick="sendOrder()">Order</button>
    <p id="response"></p>

    <script>
        function sendOrder() {
            const user = document.getElementById('user').value;
            const item = document.getElementById('item').value;
            if (!user || !item) { alert("Please enter name and food item");
return; }

            fetch('http://184.72.86.168:5000/api/order', {
                method: 'POST',
                headers: {'Content-Type': 'application/json'},
                body: JSON.stringify({user, item})
            })
            .then(res => res.json())
            .then(data => {
                document.getElementById('response').innerText = data.message
|| data.error;
                document.getElementById('user').value = '';
                document.getElementById('item').value = '';
            })
            .catch(err => console.error(err));
        }
    </script>
</body>
</html>
```

## Step 5 – Test End-to-End

1. Open browser: `http://184.72.86.168/index.html`
2. Enter name + food → click **Order**
3. Check **RDS** `santhos` **table** → order stored.

---

## Step 6 – Security & Notes

- Private EC2: only allows traffic from Public EC2.
- RDS: only allows traffic from Private EC2.
- CORS enabled in Flask for safe browser access.
- Optional: use Gunicorn + Nginx for production.

---

**Flow:**

```
Browser → Public EC2 (proxy.py) → Private EC2 (app.py) → RDS → Browser
```

This PDF-ready document contains all **codes and step-by-step instructions** to deploy a secure two-tier AWS Food Order App.