🚀 Flask + Express Deployment on AWS EC2 with Docker (Task 1 & Task 2)

This documentation explains the complete step-by-step process of deploying a Flask backend and an Express.js frontend application on AWS EC2 instances using Docker and Docker Compose.
It covers:

✅ Task 1: Deploy both Flask and Express in a single EC2 instance

✅ Task 2: Deploy Flask and Express on separate EC2 instances

🧩 Project Structure

```
aws-flask-express-deploy/
├── Backend/
│   ├── app.py
│   ├── Dockerfile
│   └── requirements.txt
├── Frontend/
│   ├── index.js
│   ├── package.json
│   ├── Dockerfile
│   └── docker-compose.yml
├── docker-compose.yml (for combined deployment)
└── readme.md
```

🥇 Task 1: Deploy Flask + Express in a Single EC2 Instance

1️⃣ Launch EC2 Instance

Choose Ubuntu 22.04 or 24.04 LTS.

Enable ports 22, 3000, 5000 in the security group.

SSH into your instance:

```
ssh -i "mohdsahal924.pem" ubuntu@<EC2_PUBLIC_IP>
```

2️⃣ Install Docker & Docker Compose

```
sudo apt update -y
sudo apt install -y docker.io docker-compose
sudo systemctl enable docker
sudo systemctl start docker
```

3️⃣ Clone or Upload Your Project

```
git clone <your-repo-url>
cd aws-flask-express-deploy
```

4️⃣ Build & Run Containers

Use docker-compose.yml with both frontend and backend services:

```
version: '3.8'
services:
 backend:
  build: ./Backend
  container_name: aws-flask-express-deploy_backend
  ports:
   - "5000:5000"
  restart: always

 frontend:
  build: ./Frontend
  container_name: aws-flask-express-deploy_frontend
  ports:
   - "3000:3000"
  environment:
   - BACKEND_URL=http://backend:5000
  depends_on:
   - backend
  restart: always
```

Then run:

```
sudo docker-compose up -d --build
```

Check status:

```
sudo docker ps
```

✅ Expected:

Backend: http://<EC2_PUBLIC_IP>:5000

Frontend: http://<EC2_PUBLIC_IP>:3000

5️⃣ Auto-Start on Reboot (Optional)

Create a systemd service:

```
sudo nano /etc/systemd/system/aws-flask-express.service
```

```
[Unit]
Description=Docker Compose Flask + Express App
After=docker.service
Requires=docker.service

[Service]
WorkingDirectory=/home/ubuntu/aws-flask-express-deploy
ExecStart=/usr/bin/docker-compose up -d
ExecStop=/usr/bin/docker-compose down
Restart=always
User=ubuntu

[Install]
WantedBy=multi-user.target
```

Enable service:

```
sudo systemctl daemon-reload
sudo systemctl enable aws-flask-express.service
sudo systemctl start aws-flask-express.service
```

Check:

```
sudo systemctl status aws-flask-express.service
```

✅ Task 1 Complete

Backend and frontend run together in a single EC2 instance.

🥈 Task 2: Deploy Flask & Express on Separate EC2 Instances

This demonstrates scaling and separation of services.

📍 Step 1: Backend EC2 Setup

Launch a new EC2 instance (Ubuntu).

SSH into it and install Docker:

```
sudo apt update -y
sudo apt install -y docker.io docker-compose
```

Clone or upload Backend/ folder.

Run backend container:

```
sudo docker-compose up -d --build
```

Verify:

```
curl http://localhost:5000
```

✅ Output:

```
{"message": "Hello from Flask backend!"}
```

Test public URL:

```
http://<BACKEND_PUBLIC_IP>:5000
```

📍 Step 2: Frontend EC2 Setup

Launch a second EC2 instance for the frontend.

Install Docker and Docker Compose:

```
sudo apt update -y
sudo apt install -y docker.io docker-compose
```

Copy frontend code to this instance using scp:

```
scp -i "mohdsahal924.pem" -r Frontend ubuntu@<FRONTEND_PUBLIC_IP>:~/
```

Edit docker-compose.yml for the frontend:

```
version: "3.8"
services:
  frontend:
    build: .
    container_name: frontend_app
    ports:
      - "3000:3000"
    environment:
      BACKEND_URL: "http://<BACKEND_PUBLIC_IP>:5000"
    restart: always
```

Build and run:

```
cd Frontend
sudo docker-compose up -d --build
```

Verify container:

```
sudo docker ps
```

✅ Output:

```
PORTS: 0.0.0.0:3000->3000/tcp
```

Access frontend in a browser:

```
http://<FRONTEND_PUBLIC_IP>:3000
```

✅ Expected Output:

```
🚀 Express Frontend
✅ Backend says: Hello from Flask backend! 🚀
```

📜 Step 3: Health Check

```
curl -s -o /dev/null -w "Backend: %{http_code}\n" http://<BACKEND_PUBLIC_IP>:5000
curl -s -o /dev/null -w "Frontend: %{http_code}\n" http://<FRONTEND_PUBLIC_IP>:3000
```

✅ Expected:

```
Backend: 200
Frontend: 200
```