Complete AWS EC2 Deployment Guide

Adaptive Assessment System (FastAPI + React + Docker)

Table of Contents

Prerequisites

What You Need

- AWS Account with EC2 access
- GitHub repository with your code
- Local computer (Windows/Mac/Linux)
- Internet connection

Tools Required (Local Machine)

- AWS CLI
- Session Manager Plugin
- SSH client
- Web browser

Part 1: AWS EC2 Setup

Step 1.1: Launch EC2 Instance

Go to AWS Console:

1. Navigate to https://console.aws.amazon.com
2. Go to EC2 service
3. Click "Launch Instance"

Instance Configuration:

- **Name:** `adaptive-assessment-app`
- **AMI:** Amazon Linux 2023 (Free tier eligible)
- **Instance Type:**
  - Testing: t2.micro (1 vCPU, 1 GB RAM)
  - Production: t3.medium (2 vCPU, 4 GB RAM)

Key Pair:

1. Click "Create new key pair"
2. Name: `adaptive-assessment-key`
3. Type: RSA
4. Format: .pem
5. **Download and save the .pem file**

Network Settings:

- Auto-assign public IP: **Disable** (per your organization policy)

- Create new security group: `adaptive-assessment-sg`

Security Group Rules:

Type	Port	Source	Description
SSH	22	My IP	SSH access
HTTP	80	0.0.0.0/0	Frontend
Custom TCP	8000	0.0.0.0/0	Backend API

Storage:

- Size: 30 GB
- Type: gp3

Launch the instance!

Step 1.2: Note Down Instance Details

After launch, copy:

- **Instance ID:** (e.g., i-02a54d27baf9fbaa3)
- **Private IPv4 address:** (e.g., 108.0.7.193)
- **Availability Zone:** (e.g., ap-south-1a)

Part 2: EC2 Configuration

Step 2.1: Connect to EC2

Using AWS Systems Manager (No SSH key needed):

1. Go to EC2 Console → Instances
2. Select your instance
3. Click "Connect" → "Session Manager"
4. Click "Connect"

Using SSH (if you have public IP):

bash

```
chmod 400 ~/.ssh/adaptive-assessment-key.pem
ssh -i ~/.ssh/adaptive-assessment-key.pem ec2-user@YOUR_EC2_IP
```

Step 2.2: Install Docker

bash

```
# Update system
sudo yum update -y

# Install Docker
sudo yum install docker -y

# Start Docker
sudo systemctl start docker
sudo systemctl enable docker

# Add user to docker group
sudo usermod -a -G docker ec2-user

# Logout and login again
exit
```

Reconnect to EC2, then verify:


bash

```
docker --version
```

Step 2.3: Install Docker Compose


bash

```
# Install Docker Compose
sudo curl -L "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docl

# Make executable
sudo chmod +x /usr/local/bin/docker-compose

# Verify
docker-compose --version
```

Step 2.4: Install Git


bash

```
sudo yum install git -y
git --version
```

Part 3: Code Deployment

Step 3.1: Clone Repository

Using Git:

bash

```
cd ~
git clone https://github.com/YOUR_USERNAME/adaptive-assessment-system.git
cd adaptive-assessment-system
```

Or upload code via other methods if needed

Step 3.2: Project Structure

```
adaptive-assessment-system/
├── docker-compose.yml          # Production config
├── docker-compose.local.yml    # Local testing config
├── backend/
│   ├── Dockerfile_BE
│   ├── requirements.txt
│   ├── main.py
│   └── data/                   # SQLite database location
└── frontend/
    ├── Dockerfile_FE
    ├── nginx.conf
    ├── package.json
    └── src/
```

Step 3.3: Create nginx.conf (If Missing)

bash

```
cd ~/adaptive-assessment-system/frontend

cat > nginx.conf << 'EOF'
server {
    listen 80;
    server_name localhost;

    root /usr/share/nginx/html;
    index index.html;

    location / {
        try_files $uri $uri/ /index.html;
    }

    gzip on;
    gzip_types text/plain text/css application/json application/javascript text/xml application/xml application/xml+rss text/javascript;

    location /static/ {
        expires 1y;
        add_header Cache-Control "public, immutable";
    }
}
EOF
```

Step 3.4: Create Data Directory

bash

```
mkdir -p ~/adaptive-assessment-system/backend/data
chmod -R 755 ~/adaptive-assessment-system/backend/data
```

Part 4: Docker Configuration Files

Step 4.1: docker-compose.yml (Production - VPN Access)

yaml

```yaml
version: '3.8'

services:
  backend:
    build:
      context: ./backend
      dockerfile: Dockerfile_BE
    container_name: adaptive-backend
    ports:
      - "8000:8000"
    volumes:
      - ./backend:/app
      - ./backend/data:/app/data
    environment:
      - DATABASE_URL=sqlite:///./data/adaptive_assessment.db
      - FRONTEND_URL=http://YOUR_PRIVATE_IP
      - ENV=production
      - CORS_ORIGINS=http://YOUR_PRIVATE_IP,http://localhost:8080,http://localhost:3000,http://localhost
    restart: unless-stopped
    command: uvicorn main:app --host 0.0.0.0 --port 8000

  frontend:
    build:
      context: ./frontend
      dockerfile: Dockerfile_FE
    container_name: adaptive-frontend
    ports:
      - "80:80"
    environment:
      - REACT_APP_API_URL=http://YOUR_PRIVATE_IP:8000
    depends_on:
      - backend
    restart: unless-stopped
```

Replace `YOUR_PRIVATE_IP` with your EC2 private IP!

Step 4.2: docker-compose.local.yml (Local Testing - Port Forwarding)

bash

```
cd ~/adaptive-assessment-system

cat > docker-compose.local.yml << 'EOF'
version: '3.8'

services:
  backend:
    environment:
      - CORS_ORIGINS=http://localhost:8080,http://localhost:3000,http://localhost,http://YOUR_PRIVATE_IP

  frontend:
    environment:
      - REACT_APP_API_URL=http://localhost:8000
EOF
```

Part 5: Initial Deployment (Production)

Step 5.1: Build and Start Containers

bash

```
cd ~/adaptive-assessment-system

# Build all containers
docker-compose build

# Start all containers
docker-compose up -d

# Check status
docker-compose ps
```

Expected output:

```
NAME                STATUS    PORTS
adaptive-backend    Up        0.0.0.0:8000->8000/tcp
adaptive-frontend   Up        0.0.0.0:80->80/tcp
```

Step 5.2: View Logs

bash

```
# All logs
docker-compose logs

# Specific service
docker-compose logs backend
docker-compose logs frontend

# Follow logs in real-time
docker-compose logs -f
```

Step 5.3: Test on EC2


bash

```
# Test backend
curl http://localhost:8000/docs

# Test frontend
curl http://localhost:80

# Check database
ls -lh ~/adaptive-assessment-system/backend/data/
```

Part 6: Local Access Setup (Port Forwarding)

Step 6.1: Install AWS CLI (Local Machine)

Mac:


bash

```
brew install awscli
```

Windows: Download from: https://aws.amazon.com/cli/

Linux:


bash

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install
```

Verify:

bash

```
aws --version
```

Step 6.2: Configure AWS CLI

bash

```
aws configure
```

Enter:

- AWS Access Key ID: [Your access key]
- AWS Secret Access Key: [Your secret key]
- Default region: ap-south-1
- Default output format: json

Test:

bash

```
aws sts get-caller-identity
```

Step 6.3: Install Session Manager Plugin

Mac:

bash

```
brew install --cask session-manager-plugin
```

Windows: Download from: https://s3.amazonaws.com/session-manager-downloads/plugin/latest/windows/SessionManagerPluginSetup.exe

Linux:

bash

```
curl "https://s3.amazonaws.com/session-manager-downloads/plugin/latest/ubuntu_64bit/session-manager-plugin.deb" -o "session-manager-p
sudo dpkg -i session-manager-plugin.deb
```

Verify:

bash

```
session-manager-plugin
```

Step 6.4: Start Port Forwarding

Terminal 1 - Frontend Port Forwarding:


bash

```
aws ssm start-session \
  --target i-YOUR_INSTANCE_ID \
  --document-name AWS-StartPortForwardingSession \
  --parameters '{"portNumber":["80"],"localPortNumber":["8080"]}'
```

Keep this terminal open!

Terminal 2 - Backend Port Forwarding:


bash

```
aws ssm start-session \
  --target i-YOUR_INSTANCE_ID \
  --document-name AWS-StartPortForwardingSession \
  --parameters '{"portNumber":["8000"],"localPortNumber":["8000"]}'
```

Keep this terminal open too!

Step 6.5: Access Application Locally

Open browser:

- Frontend: http://localhost:8080
- Backend API Docs: http://localhost:8000/docs

Part 7: Build for Local Testing

When to Use Which docker-compose File?

Scenario	Command	When to Use
Production/VPN	`docker-compose up -d`	When deploying for VPN access
Local Testing	`docker-compose -f docker-compose.yml -f docker-compose.local.yml up -d`	When testing via port forwarding

Step 7.1: Build for Local Testing


bash

```
cd ~/adaptive-assessment-system

# Stop current containers
docker-compose down

# Build with local config
docker-compose -f docker-compose.yml -f docker-compose.local.yml build --no-cache frontend

# Start with local config
docker-compose -f docker-compose.yml -f docker-compose.local.yml up -d

# Check status
docker-compose ps
```

Step 7.2: Switch Back to Production


bash

```
# Stop local config
docker-compose down

# Rebuild for production
docker-compose build --no-cache frontend

# Start production
docker-compose up -d

# Check status
docker-compose ps
```

Part 8: Database Setup

Step 8.1: Create Item Banks

First, create test item banks:


bash

```
curl -X POST http://localhost:8000/api/item-banks/create \
  -H "Content-Type: application/json" \
  -d '{
    "name": "math_basic",
    "description": "Basic Math Questions",
    "items": [
      {
        "id": "q1",
        "content": "What is 2 + 2?",
        "options": ["2", "3", "4", "5"],
        "correct_answer": 2,
        "difficulty": 0.3,
        "discrimination": 1.0
      }
    ]
  }'
```

Verify:


bash

```
curl http://localhost:8000/api/item-banks
```

Step 8.2: Set Up Automated Backups


bash

```
# Create backup script
cat > ~/backup-db.sh << 'EOF'
#!/bin/bash
BACKUP_DIR="$HOME/backups"
DB_PATH="$HOME/adaptive-assessment-system/backend/data/adaptive_assessment.db"
DATE=$(date +%Y%m%d_%H%M%S)

mkdir -p "$BACKUP_DIR"

if [ -f "$DB_PATH" ]; then
    cp "$DB_PATH" "$BACKUP_DIR/db_backup_$DATE.db"
    echo "✅ Backup created: db_backup_$DATE.db"
    find "$BACKUP_DIR" -name "db_backup_*.db" -mtime +7 -delete
else
    echo "❌ Database not found"
fi
EOF

# Make executable
chmod +x ~/backup-db.sh

# Test
~/backup-db.sh

# Schedule daily backups (2 AM)
crontab -e
# Add: 0 2 * * * /home/ec2-user/backup-db.sh >> /home/ec2-user/backup.log 2>&1
```

Part 9: Management Commands

Daily Operations

Check Application Status:

bash

```
cd ~/adaptive-assessment-system
docker-compose ps
```

View Logs:

bash

```
# All services
docker-compose logs --tail=100

# Specific service
docker-compose logs backend --tail=50
docker-compose logs frontend --tail=50

# Follow in real-time
docker-compose logs -f
```

Restart Services:


bash

```
# Restart all
docker-compose restart

# Restart specific service
docker-compose restart backend
docker-compose restart frontend
```

Update Application:


bash

```
# Pull latest code
git pull origin main

# Rebuild and restart
docker-compose up -d --build

# Or for local testing
docker-compose -f docker-compose.yml -f docker-compose.local.yml up -d --build
```

Stop Application:


bash

```
docker-compose down
```

Full Cleanup (if needed):


bash

```
docker-compose down
docker-compose rm -f frontend backend
docker rmi adaptive-assessment-system-frontend adaptive-assessment-system-backend
docker builder prune -f
```

Part 10: Troubleshooting

Frontend Not Loading

Check containers:


bash

```
docker-compose ps
```

Check frontend logs:


bash

```
docker-compose logs frontend
```

Rebuild frontend:


bash

```
docker-compose up -d --build frontend
```

Backend API Errors

Check backend logs:


bash

```
docker-compose logs backend | tail -50
```

Restart backend:


bash

```
docker-compose restart backend
```

Test API directly:


bash

```
curl http://localhost:8000/docs
```

CORS Errors

Ensure CORS_ORIGINS includes all necessary origins:


yaml

```
environment:
  - CORS_ORIGINS=http://localhost:8080,http://localhost:3000,http://YOUR_PRIVATE_IP
```

Restart backend:


bash

```
docker-compose restart backend
```

Port Forwarding Issues

Check if ports are available:


bash

```
# Mac/Linux
lsof -i :8080
lsof -i :8000

# Windows
netstat -ano | findstr :8080
netstat -ano | findstr :8000
```

Restart port forwarding:

- Press Ctrl+C in terminal
- Run start command again

Database Issues

Check database file:


bash

```
ls -lh ~/adaptive-assessment-system/backend/data/adaptive_assessment.db
```

Check permissions:

bash

```
chmod -R 755 ~/adaptive-assessment-system/backend/data/
```

Restore from backup:

bash

```
cp ~/backups/db_backup_TIMESTAMP.db ~/adaptive-assessment-system/backend/data/adaptive_assessment.db
docker-compose restart backend
```

Part 11: Network Team Requirements

Information to Provide to IT/Network Team

Subject: AWS EC2 Application - Network Access Request

Application: Adaptive Assessment System
Deployment Date: [DATE]
Status: Deployed and operational, awaiting network configuration

---

INSTANCE DETAILS:
- Instance ID: i-02a54d27baf9fbaa3
- Private IPv4: 108.0.7.193
- Region: ap-south-1
- Availability Zone: ap-south-1a
- VPC ID: [Get from AWS Console]
- Subnet ID: [Get from AWS Console]
- Security Group: adaptive-assessment-sg

APPLICATION DETAILS:
Technology Stack:
  - Backend: FastAPI (Python) on port 8000
  - Frontend: React (JavaScript) on port 80
  - Database: SQLite (local file storage)
  - Containerization: Docker

Application Status:
  ✅ Backend: Running and healthy
  ✅ Frontend: Running and healthy
  ✅ Database: Initialized and functional
  ✅ All services operational

REQUIRED NETWORK ACCESS:

Inbound Access Required:
  Port 80 (HTTP)   - Frontend application
  Port 8000 (HTTP) - Backend API
  Port 22 (SSH)   - Admin access (optional)

Users Need Access To:
  - Frontend: http://108.0.7.193
  - Backend API: http://108.0.7.193:8000
  - API Documentation: http://108.0.7.193:8000/docs

ACCESS METHOD OPTIONS:

Please configure ONE of the following:

Option A: VPN Access (Recommended)

- Users connect to company VPN
- Direct access via private IP: http://108.0.7.193
- Simple and secure

Option B: Internal Application Load Balancer
- Create internal ALB pointing to EC2
- Provide internal DNS name
- Example: http://assessment.internal.company.com
- Scalable solution

Option C: Route53 Private Hosted Zone
- Create private DNS record
- Map domain to private IP
- Example: http://adaptive-assessment.company.internal
- User-friendly URLs

Option D: AWS PrivateLink / VPC Peering
- Configure VPC connectivity
- Access via private network
- Enterprise-grade solution

SECURITY GROUP CONFIGURATION:

Current Inbound Rules:
Type: SSH, Port: 22, Source: [Your IP]
Type: HTTP, Port: 80, Source: 0.0.0.0/0
Type: Custom TCP, Port: 8000, Source: 0.0.0.0/0

Recommended Changes:
Please update Source from "0.0.0.0/0" to appropriate internal CIDR:
- VPN CIDR range
- Internal network range
- Specific user group IPs

TESTING CONFIRMATION:

Application has been tested and verified:
✅ Backend API responds correctly
✅ Frontend serves application
✅ Database operations functional
✅ All 7 API endpoints operational
✅ CORS configured for internal access
✅ Error handling working
✅ Logging operational

Test URLs (from within VPC/network):
- Frontend: http://108.0.7.193
- Backend Health: http://108.0.7.193:8000/docs

- API Test: http://108.0.7.193:8000/api/item-banks

DEPLOYMENT TIMELINE:

Completed:
- ✅ Infrastructure setup
- ✅ Application deployment
- ✅ Database configuration
- ✅ Testing and verification
- ✅ Documentation

Pending:
- ⏳ Network access configuration (your team)
- ⏳ User access testing
- ⏳ Production rollout

CONTACT INFORMATION:
Deployed by: [Your Name]
Email: [Your Email]
Team: [Your Team]
Date: [Deployment Date]

Please let me know your preferred access method and timeline.

Questions to Ask Network Team

1. **What is the preferred method for internal application access?**
   - VPN
   - Internal Load Balancer
   - Private DNS
   - Other
2. **What is the CIDR range for internal users?**
   - Needed for Security Group configuration
3. **Do you need a custom domain name?**
   - If yes, what format? (e.g., assessment.internal.company.com)
4. **What is the expected timeline for access configuration?**
5. **Are there any compliance or security requirements?**
   - SSL/TLS requirements
   - Authentication requirements
   - Logging requirements
6. **Who will manage DNS/Load Balancer configuration?**
7. **What is the process for user access provisioning?**

Part 12: Access URLs Reference

Production Access (After VPN Setup)

For End Users:

Frontend Application: http://108.0.7.193
Backend API Documentation: http://108.0.7.193:8000/docs

Local Testing (Port Forwarding)

For Developers:

Frontend: http://localhost:8080
Backend API: http://localhost:8000/docs

API Endpoints

POST /api/assessments/start
POST /api/assessments/{session_id}/answer
GET /api/assessments/{session_id}/results
GET /api/item-banks
POST /api/item-banks/create
GET /api/item-banks/{item_bank_name}
GET /api/debug/sessions

Part 13: Quick Reference Commands

Start Application (Production)


bash

```
cd ~/adaptive-assessment-system
docker-compose up -d
```

Start Application (Local Testing)


bash

```
cd ~/adaptive-assessment-system
docker-compose -f docker-compose.yml -f docker-compose.local.yml up -d
```

Stop Application


bash

```
docker-compose down
```

Update and Restart

bash

```
git pull
docker-compose up -d --build
```

View Logs

bash

```
docker-compose logs -f
```

Backup Database

bash

```
~/backup-db.sh
```

Check Status

bash

```
docker-compose ps
curl http://localhost:8000/docs
curl http://localhost:80
```

Deployment Complete! ✅

Your Adaptive Assessment System is now deployed on AWS EC2 and ready for use once network access is configured.

Next Steps:

1. Share network requirements with IT team
2. Wait for VPN/access configuration
3. Test with end users
4. Monitor and maintain

Support:

- Check logs for issues: `docker-compose logs`
- Restart services: `docker-compose restart`
- Full rebuild: `docker-compose up -d --build`

Document Version: 1.0 **Last Updated:** 2025-10-10 **Maintained By:** [Your Name]