Deploying Fullstack Web App to AWS & integrating Dynatrace 💻

We will not be doing a local deployment of this App (on your machine). Instead we'll go straight to Cloud (aws) deployment. If you'd like to spin it up locally first, please make sure that you have a local instance of Postgresql server running. You can use DBeaver/PgAdmin to interact with the underlying database.

AWS Deployment guide 🛠️

Key components that our Web App deployment will rely on:

- **AWS EC2** – hosting the backend
- **AWS RDS** – for managing the PostgreSQL database
- **AWS API Gateway** – routing & managing API requests
- **AWS Amplify** – deploying (CICD) & hosting the frontend
- **AWS S3** – graphical assets storage storage (product images)

Architecture

1. Pre-requisites ✅

1. Download & unzip the materials to this practice case. Open in IDE
2. Study **README-1_App-overview**
3. Set up an AWS account (with Billing enabled)
4. Install git client locally

2. AWS Cloud Networking 🛜

1. AWS Services -> VPC -> Create VPC
2. Subnets -> Create subnet -> Create 1 public & 2 private subnets
3. Internet Gateway -> Create
4. Set up Public & Private Route tables

3. AWS EC2 (backend hosting) ⚙️

1. Create EC2 instance -> Connect using AWS UI
2. Set up a git repo locally, set a remote origin master connection &

```
git add .
git commit –m"initial commit"
git push origin master
```

3. Install & set up all the required dependencies on your EC2 instance. Make sure that you install Node version that is currently supported by Dynatrace's OneAgent. Otherwise, you will lack "Deep

Monitoring" capabilities Node supported versions I suggest you pick the "Active LTS" Node version

```
        sudo su -
        curl -o- https://raw.githubusercontent.com/nvm-
sh/nvm/v0.40.3/install.sh | bash
        . ~/.nvm/nvm.sh
        nvm install 22.20.0
        nvm use 22.20.0
        node -v
        npm -v

        sudo yum update -y
        sudo yum install git -y
        git --version

        git clone https://github.com/migumax/inventory-management.git
        cd inventory-management
        npm i
        cd server/
        npm i

        echo "PORT=80" > .env
        npm run dev
```

Check that the endpoint is talking to us & then Ctrl+C to stop the backend

4. Install pm2 tool & typescript

```
npm i pm2 -g
npm i typescript -g

sudo env PATH=$PATH:$(which node) $(which pm2) startup systemd -u $USER --
hp $(eval echo ~$USER)

pm2 start ecosystem.config.js
```

Do the checks & explore:

```
pm2 status
pm2 monit
pm2 delete all
pm2 start ecosystem.config.js
```

4. AWS RDS (psql hosting)

1. Services -> RDS -> Subnet groups -> Create DB Subnet group -> Associate with 2 private subnets we created earlier

2. RDS -> Create database -> PostgreSQL
3. Click DB instance -> Connectivity & Security -> VPC Security groups -> Edit Inbound rules
4. Copy RDS's endpoint, db name, username, password
5. Connect to EC2 instance and:
    - ```
      sudo su -
      cd inventory-management/server
      ```
    - Edit .env "DATABASE_URL="postgresql://postgres:yourpasswordhere@rds-inventorymanagement.yourawsendpoint.com:5432/inventorymanagement?schema=public"
    - ```
      pm2 delete all

      npx prisma generate
      npx prisma migrate dev --name init
      npm run seed

      pm2 start ecosystem.config.js
      ```
6. Check DB's data availability in your browser Visit EC2's IPv4/dashboard. For example, *9.99.99.99/dashboard*

5. AWS Amplify (frontend CICD & hosting) 📊

1. Services -> API Gateway -> HTTP API -> Build -> Add Integrations & Add "prod" stage HTTP + GET + http://yourEC2IPv4/dashboard HTTP + GET + http://yourEC2IPv4/users HTTP + GET + http://yourEC2IPv4/expenses HTTP + ANY + http://yourEC2IPv4/products
2. Services -> Amplify -> Create new App -> GitHub (source) -> Select repository + "My app is a monorepo" + "client"
3. Advanced settings -> Add environment variable NEXT_PUBLIC_API_BASE_URL http://yourInvokeURL from API Gateway
4. Check the Web App in browser

6. AWS S3 (graphical assets storing) 📦

1. Services -> S3 -> Create bucket -> Allow "All public access" -> Create
2. Upload server/assets/* to S3
3. Bucket -> Permissions -> Add bucket policy: { "Version": "2012-10-17", "Statement": [{ "Sid": "PublicReadGetObject", "Effect": "Allow", "Principal": "*", "Action": "s3:GetObject", "Resource": "arn:aws:s3:::s3-inventorymanagement/*" }] }
4. Edit newly set up S3 endpoint in 4 locations:

- client/src/next.config.mjs
- client/src/app/(components)/Navbar/index.tsx
- client/src/app/(components)/Sidebar/index.tsx
- client/src/app/Dashboard/CardPopularProducts.tsx
- client/src/app/products/page.tsx

5. Push changes

```
git add .
git commit -m"updated frontend variables"
git push origin master
```

6. Make sure the new AWS Amplify deployment succeeded
7. Explore your fully functioning Web App & share with the world 🌍
8. Check logs with `pm2 logs inventory-management` while navigating across Web App's pages

7. AWS Infrastructure Monitoring 📊

1. AWS -> CloudWatch -> Dashboards -> Automatic Dashboards ->
   - EC2
   - RDS