Best practices and suggestions of project

Frontend: Next.js build deployed on EC2, using reverse proxy to host

Backend: Python (Fast API) app deployed on EC2 + /blogs route for wordpress blog

Route and records: Cloudflare

AI LLM: AWS Bedrock

Monitoring: No

Scaling: No

Load balancer: No

Want to Dockerize application

ECS, EKS, Beanstalk

Preferred to use Amplify for Next.js app but we then cannot add our custom proxy for going to /blogs route for a wordpress blog application
What other things will we need to migrate for doing this change

---

1. Create Dockerfile for both frontend and backend applications
2. Use ECS Fargate or ElasticBeanstalk (Scaling and Load balanced)
3. For CI/CD, we can use AWS Codepipeline which can integrate well with both Fargate and Beanstalk
4. In Codepipeline, we can create 2 build projects, frontend/buildspec.yml and backend/buildspec.yml

5. Monitoring Enable Detailed monitoring in ECS or Beanstalk to get logs at 1-5 sec rate (Or use Datadog or similar monitoring based managed tools)

1. Dockerize Both Apps

```
FROM python:3.9
WORKDIR /code
COPY ./requirements.txt /code/requirements.txt
RUN pip install --no-cache-dir --upgrade -r /code/requirements.txt
COPY ./app /code/app
CMD ["fastapi", "run", "app/main.py", "--port", "80"]
```

```
# Dockerfile for Next.js
FROM node:18-alpine
WORKDIR /app
COPY . .
RUN npm install && npm run build
EXPOSE 3000
CMD ["npm", "start"]

// can also do a multi-phase build with base node image and nginx image to s
erve ui through reverse-proxy inside container at port 80
```

2. AWS CodePipeline

There will be 2 build steps in a pipeline, frontend and backend, connect your github repo with github connector

```
# frontend/buildspec.yml
version: 0.2

env:
  variables:
    IMAGE_NAME: fastapi-app-frontend
    ECR_REPO: <your_account_id>.dkr.ecr.<region>.amazonaws.com/fastapi-app

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region $AWS_DEFAULT_REGION | docker log
      - TAG=$(date +%Y%m%d%H%M%S)
      - echo "IMAGE_TAG=$TAG" >> build.properties

  build:
    commands:
      - echo Building the Docker image...
      - docker build -t $ECR_REPO:latest -t $ECR_REPO:$TAG .
      - docker push $ECR_REPO:latest
      - docker push $ECR_REPO:$TAG

  post_build:
    commands:
      - printf '[{"name":"frontend","imageUri":"%s"}]' "$ECR_REPO:$TAG" > image

artifacts:
  files: imagedefinitions.json
```

```
# backend/buildspec.yml
```

This will create docker images of application in ECR, ready to be deployed in ECS fargate

- The imagedefinitions.json file built in artifacts can be directly used to deploy these builds in ECS fargate

- IAM Roles access will need to be added in ECS to access other services (like bedrock etc)
- Better to use https://www.npmjs.com/package/next-http-proxy-middleware for frontend application proxy of wordpress application and deploy this with Amplify
  - Add WAF rules in Amplify for web application security
- All endpoints created with these services should be routed through a CNAME route so the original DNS and IPs are not exposed.