Seungwan Hong www.hosugator.com

Hosugator Web

Personal Portfolio Web Site with Next.js, FastAPI, AWS Solution

Summary

This system is a serverless architecture consisting of a Next.js-based frontend and a FastAPI-based backend. It ensures high availability and scalability by combining static hosting via AWS S3 and CloudFront with containerized API services using ECS Fargate.

Hierarchy

Client & Network

- User: Service visitor.
- Route 53: Handles DNS routing and branches frontend traffic and API requests.
- AWS Certificate Manager (ACM): Manages SSL/TLS certificates and provides HTTPS security for CloudFront and ALB.

Frontend: Static Hosting

- Amazon S3: Stores static asset files built with Next.js.
- CloudFront: Supports low-latency global access through edge locations and performs data caching.
- CloudFront Function: Handles 301 redirects from the www domain to the root domain while maintaining query parameters.

Backend: API & Logic

- Application Load Balancer (ALB): Receives incoming API requests and forwards them to target groups.
- Target Group: Specifies the actual ECS service destination for traffic forwarded by the ALB.
- ECS Cluster (Fargate Service): A serverless environment where FastAPI containers run to process business logic.
- ECS Environment Variables: Injects sensitive information, such as the OpenAI API Key, into containers at runtime.

Security & Auth

- Security Group (ALB): Controls external access via ports 80 and 443.
- Security Group (ECS): Enhances backend security by only allowing traffic originating from the ALB.
- IAM User / Role: Controls permissions between GitHub Actions and the ECS cluster while adhering to the principle of least privilege.

CI/CD and Operations

- GitHub Actions: The pipeline is triggered upon source code changes.
- Deploy Static: Updates static files in S3 to deploy the frontend.
- Push Image & Update Service: Builds and pushes new container images and updates the ECS service.

Retrospective

While I became familiar with theoretical architecture design while preparing for the AWS SAA certification, I had a strong desire to apply it to a real service.
It was rewarding to see UX improve and availability and scalability secured as theory was implemented into actual code.
Although current commitments such as certifications, resumes, and side projects limit my available time, I intend to record better architecture updates whenever possible.
As stated in my introduction, I believe technology only gains life when it creates external value.
I plan to continue efficient architecture updates for the purpose of value creation.
Thank you.

Client & Network
User
API Request /recommendati
Access: www or root
AWS Certificate Manager SSL
Route 53 DNS & Routing
Security & Auth
Security Group: ECS
IAM User / Role
Security Group: ALB
GitHub Actions
Allow Traffic from ALB
Least Privilege
Allow 80/443
HTTPS Certificate
Route to API
Push Image & Update Servi
HTTPS Certificate
Frontend Traffic
Deploy Static
301 Redirect with QueryPar
Backend: API & Logic
Application Load Balancer
Forward via SG_ALB
Target Group
ECS Environment Variables OpenAI API Key
Inject at Runtime
Route via SG_ECS
ECS Cluster
ECS Fargate Service FastAPI/Container
Frontend: Static Hosting
CloudFront CDN/Edge
CloudFront Function 301 Redirect: www to root
Fetch Content
Amazon S3 Static Files