개발 포트폴리오

황세현

🔗hwanghyun3@gmail.com

Built at 9 Mar 2025 from https://man.hwangsehyun.com/

1. 🔗개발 포트폴리오 ..... p. 1
1. 1. 🔗 발자취 ..... p. 3
1. 2. 🔗 MAN Sehyun Hwang - User's Manual (1) ..... p. 6

2. 🔗 Posts ..... p. 13

3. 🔗 수행 프로젝트 ..... p. 15
3. 1. 🔗 LLM을 이용한 영양 상담 서비스 ..... p. 21
3. 2. 🔗 LLM을 이용한 기업용 챗봇 서비스 ..... p. 27
3. 3. 🔗 LLM 기반 작명 서비스 ..... p. 31
3. 4. 🔗 보행자 & 번호판 인식 솔루션 ..... p. 35
3. 5. 🔗 X-Ray 식품 검사 시스템 ..... p. 44
3. 6. 🔗 Serverless 기반 신문 PDF 정형화 ..... p. 48
3. 7. 🔗 웹 기반 강화 학습 시뮬레이션 서비스화 ..... p. 54
3. 8. 🔗 플랜트 육상 운송성 평가 시스템 ..... p. 57

4. 🔗 Tech ..... p. 60
4. 1. 🔗 Experiences with Networking ..... p. 61
4. 2. 🔗 Monitoring Experiences ..... p. 62
4. 3. 🔗 Operations: Dev, Building, Testing, CI/CD, and more... ..... p. 65
4. 4. 🔗 Security Routines ..... p. 67
4. 5. 🔗 Skills & Experiences with AWS ..... p. 70

🔗Home

발자취

2 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

Career

소속	재직 기간	부서	담당 업무
연세대학교 기계공학과	2019/03 ~ 2021/02	🔗지식 기반 설계 연구실	AI serving full-stack 개발
🔗넥스트랩	2021/01 ~ 2024/02	지능형 자동화 사업부	백엔드 및 배포 리드
🔗크라우드웍스	2024/03 ~ 2024/11	AI 개발실	백엔드 및 인프라 / R&D

[2024] 크라우드웍스

- *AI 개발실*, 9개월 근무
- 🔗대표 프로젝트: LLM을 이용한 영양 상담 서비스

*크라우드웍스의 AI 개발실*은 고객사의 사내 문서부터 서비스 데이터베이스까지 정형/비정형 데이터에 LLM을 접목하는 사업을 위한 개발을 진행합니다. 더 큰 기업과 협업하기 위해 필요한 기초적인 보안과 인프라 관련 업무를 통해서 다양한 프로젝트에 참여했으며, HTML로 변환된 사내 문서를 LLM에 적합한 형태로 정형화 및 검수하는 실험적인

내부용 프론트엔드 개발을 진행했습니다. 대표 프로젝트에서는 메인 Python 개발자로서 백엔드 개발에 참여하면서 AWS에 서비스를 배포하기 위해 IaaS 툴을 취급했습니다.

[2021~2023] 넥스트랩

- *지능형 자동화 사업부*, 선임 연구원, 3년 근무
- 🔗대표 프로젝트: 번호판 인식 솔루션

*넥스트랩의 지능형 자동화 사업부*는 주로 보행자와 자동차 등에 대한 컴퓨터 비전 솔루션을 만드는 팀입니다. 팀이 막 만들어졌을 때 입사해 3년간 전문 연구 요원으로 복무했습니다. 입사 초기에는 알고리즘 POC부터 서비스 운영까지 모두 직접 했었지만, 팀이 성숙하면서 서비스 설계, 개발 방향 제안, 인프라 개발을 맡았었습니다. 가장 오래 재직한 회사인 만큼, 다양한 프로젝트를 수행하면서 모든 개발 분야에 걸쳐서 문제를 컨테이너 단위로 쪼개고 개발, 배포, 테스트하는 경험치를 많이 축적한 시간이었습니다. 7~8명 규모의 팀에서 따라서 체계적으로 협업하는 방법을 배웠습니다.

[2019~2021] 지식 기반 설계 연구실

- 석사 학위 취득
- 🔗졸업 연구: 웹 기반 강화 학습 시뮬레이션 서비스화

*지식 기반 설계 연구실*은 제조업과 기계 공학 전반에 걸쳐 현실 속의 문제를 풀어온 연구실입니다. 더 많은 현실 데이터를 다루고 연구 도메인을 확대하기 위해서 CAD와 설계 자동화에서 컴퓨터 비전으로 연구 트렌드를 변경하던 시기에 석사 학위를 취득했습니다. 저는 AI를 연구하던 동료들 사이에서 유일한 풀스택 웹 개발자로서 클라우드와 최신 Web API를 이용한 AI serving에 관심을 가졌습니다. 개발 여정의 시작 단계부터 연구실에 AWS를 도입하면서 클라우드 네이티브를 추구했으며, 네트워크를 통해서 분리된 시스템을 구성해 보통의 신입 개발자보다는 크고 복잡한 문제를 풀어보는 경험을 했습니다.

Supplements

- 소프트웨어 아키텍트로서의 전반적인 경험은 🔗프로젝트 목록 을 참조해주세요.
- 회사 프로젝트를 통해서는 소개하기 힘들었지만, 개인 프로젝트와 회사 프로젝트에서 빼놓지 않고 적용했던 기술들은 별도의 🔗기술 카테고리 에 별도로 정리되어 있습니다.

🔗Home

</> MAN Sehyun Hwang - User’s Manual (1)

6 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

🪪 NAME

- 황세현, *Sehyun Hwang*
- 🔗hwanghyun3@gmail.com
- Lives in Seoul

🙋 SYNOPSIS

개발팀이 조화롭게 일할 수 있도록 노력하는 백엔드 개발자 황세현입니다. 저는 대학원에서 웹 개발을 파고든 후로 현재 현업에서는 4년 차에 접어들었습니다. 제가 넥스트랩에서 일했던 지능형 자동화 팀은, 제가 입사하기 직전 신규로 꾸려져 현재는 차량, 보행자 영상 데이터를 다루는 서비스를 다양한 고객사에 납품할 수 있을 정도로 성장했습니다. 알고리즘 POC부터 서비스 운영까지 직접 발로 뛰었던 시기도 있었지만, 현재는 서비스 설계, 개발 방향 제안, 인프라 방면에서 역할을 다하고 있습니다.

- Long-running task management를 효과적으로 수행하기 위한 MSA 설계, 배포에 대해서 풍부한 경험이 있습니다.
- 이를 위해 클라우드 환경에서는 AWS의 Serverless service, 로컬 환경에서는 Redis와 Node.js를 이용합니다.

- Linux 인프라부터 클라우드 기술 트렌드까지 폭넓은 이해하고 있으며, 배포와 서비스의 취지를 고려해 front-, back-end의 개발 방향을 설정하고 개발하고 서비스화할 수 있습니다.
- 개발에서 지향하는 바가 뚜렷한 편이며, 이를 깊이 신뢰해주는 동료들과 함께 흉악하지만 빠르고 안정적인 소프트웨어를 만들어왔습니다.
- 컴퓨터를 다루는 기술인으로서 내가 만든 소프트웨어가 미치는 영향에 대해서 책임감을 느끼고, 이를 위해 사용자/비개발자와의 의사소통에 관심이 많습니다.

🌮 Culture & Beliefs

⚖️ Use of Design Principles

개발자들은 기능적 요구와 기술적 제약이 충돌하는 상황을 항상 맞닥뜨립니다. 저는 이러한 상황에서 현재 서비스 구조에서 꼭 취해야 할 장점을 선별하고 이를 토대로 설계 원칙을 지정해 문제를 풀어왔습니다. 종합적으로 각 Microservice가 수행하는 역할을 구분, 정의하고 이를 침범하지 않도록 의사 결정을 내리는 업무를 맡아왔습니다. 설계 의도/원칙의 초안을 문서화해 팀원들과 논의하여 뼈대를 구성하고, 업무를 분담하고 살을 붙여나가 서비스를 완성합니다.

🔗X-ray 식품 검사 시스템 을 전면 재개발하는 프로젝트의 초기 단계에서는 고유 규격을 가진 3종류의 하드웨어와 복잡한 비즈니스 로직을 적은 개발 인력으로 구현해야 하는 과제가 있었습니다. 저는 전체 시스템을 1) 각 장비에 명령을 전달하는 프로토콜을 구현한 Microservice와, 이들 Microservice를 동작 프로파일 단위로 추상화한 하드웨어 스택, 2) 검사 수행에 필요한 비즈니스 로직과 검사 이력을 관리하는 애플리케이션 스택으로 크게 나누고 기술 스택을 할당했습니다. 하드웨어 스택에서는 Stream interface와 이벤트 기반 처리에 강점이 있는 Node.js를 이용했으며, 객체 지향적인 TypeScript type을 Event Emitter에 적용해 하드웨어 프로토콜을 정교하고 편리하게 개발할 수 있도록 구성했으며, C++ SDK를 이용해 이미지 획득 모듈을 구현했습니다. 애플리케이션 스택에서는 제가 개발한 이미지 획득과 이물 검사 과정을 Postgre SQL에 적재할 수 있도록 하는 Schema 설계에 참여했으며, 이미지 데이터 S3 compatible storage를 채택

했습니다. 명확한 시스템 구성 덕에 개발 양에 비해 인력과 시간이 매우 부족한 상황에서 실물 장비를 이용한 개발과 테스트를 최소화할 수 있었습니다.

넥스트랩에서 소속 팀의 가장 핵심 서비스라고 할 수 있는 🔗번호판 인식 모듈은 1) Stateless하고 수평 확장이 용이해야 하며, 2) 이미지 처리 컨테이너는 Worker 역할을 수행하며, Worker 내에서는 검증되지 않은 비동기 코드를 사용하지 않는다는 원칙에 기반해 개발했습니다. 해당 모듈이 기업용 솔루션으로 납품될 때, 고객사마다 임베디드 보드, PC 환경, 고성능 서버 환경 등 다양한 인프라 위에서 구동될 것을 요구받은 상황에서 Horizontal scaling은 필수적인 고려 사항이었으며, 더 나아가 클라우드에서 구동되는 SaaS 서비스로 발전할 가능성이 논의된 사실도 반영되었습니다.

Worker 내부에서 비동기 처리를 지양했던 이유는 추론 가속을 위해 사용했던 Tensor RT framework의 비동기 처리와 충돌 가능성을 방지하고 안정성을 높이기 위함이었습니다. Main thread에서는 🔗Redis stream 또는 🔗mmap 에 적재된 메시지를 polling 하는 작업만 수행하는 단순한 구조를 선택했으며, 투입한 컴퓨팅 리소스에 비례해 Worker 개수만큼 선형적으로 성능이 향상함을 확인했습니다. Stateless를 지향했기 때문에 같은 코드베이스로부터 아래와 같은 다양한 형태의 서비스를 구현할 수 있었습니다:

- 카메라와 직접 연결되는 실시간과 request/response 서비스 모드가 존재합니다.
- 임베디드 환경에서는 산업용 카메라를 통한 이미지 획득 서비스는 60 FPS 이상, AI 모듈은 약 5 FPS으로 구동되는 상태에서, 고객사가 원할 때 추론 결과를 약 1 FPS로 가져갈 수 있었습니다.
- 서버 환경에서는 40개의 GPU worker를 이용해 300 RPS를 달성했습니다.
- 또한 정확도 확보를 최우선 과제로 하는 딥러닝 담당 팀원과의 협력에도 강점을 보였습니다.

🎨 Creativity

앞서 설정한 설계 원칙을 준수하기 위해서는 독창적인 구현이 필요할 때가 있었는데, 제가 일했던 팀은 실력 있는 개발자들로 이루어져 있으나 인원이 소수인 탓에, 코드 양이

많고 테스트가 복잡하다면 표준적인 개발 방식을 채택하기 힘든 경우가 많았습니다. 우선 전체 시스템이 예상 밖의 동작을 하지 않도록 인프라의 제약 사항, 사용하고자 하는 브라우저의 Web API와 Linux system call까지 고려해 각 모듈의 역할을 이상적으로 지정해 봅니다. ~~(추후 타협될 가능성이 물론 있습니다.)~~ 이때 다양한 이유에서 검토, 실험, 증명이 필요하다면, 풀어야 할 문제를 단순화한 개발 환경 또는 테스트 케이스를 별도로 구축해 커뮤니케이션했습니다. 이때 🔗repl.it , 🔗CodePen , 🔗StackBlitz , 🔗TypeScript Playground , Google CoLab과 같은 Browser IDE 또는 컨테이너 환경을 이용해 실험에 대한 접근성을 높이고 실험 결과를 관리할 수 있도록 신경 썼습니다.

🔗번호판을 인식하는 HTTP API를 개발하는 프로젝트 에서는 Web application server 없이 Nginx + Redis만으로 기능을 구현한 경험이 있습니다. 설계는 교과서적인 Fan-out 패턴이었지만, Python 번호판 인식 컨테이너에서 Fast API 등의 서버 프레임워크를 사용하지 않고 Nginx에서 담당하게 함으로써 속도와 안정성을 끌어올렸습니다. 표준적인 HTTP 상태 코드를 구현하기 위해서는 Nginx에서 HTTP 관련 로직이 작동해야 했는데, 이를 위해서 🔗OpenResty 의 Lua scripting을 이용했습니다. 이때 코드의 독해가 어려운 문제를 해결하기 위해서 Node.js를 이용해 커버리지가 100%에 가까운 테스트 코드를 작성했습니다. 이와 같이 레퍼런스가 없는 설계를 구현할 때는 팀원들에 비해 상대적으로 과다하게 창의적이지 않도록 테스트 코드를 작성하고 배경을 문서로 작성해 팀원들과 합의를 이룰 수 있도록 노력했습니다.

🫂 Relational

> Illustration by Scott Garrett 🔗Source

저는 개성 있는 개발자들이 모인 개발 조직에서 일해왔고 앞으로도 그러한 조직에서 일하고 싶습니다. 모호한 문제를 맞닥뜨렸을 때, 다양한 희망과 우려가 자유롭게 논의되고, 이것이 설계와 개발에 반영되는 과정에서 즐거움을 느꼈습니다. 저는 세상 전반에 관심이 많고 다양한 분야의 경계를 넘나드는 직접을 가지고 싶다는 생각을 오래전부터 했었는데, 개발 업무도 이러한 이유로 제가 좋아하고 잘할 수 있는 일이라고 확신을 가지고 있습니다. 저는 잠재적으로 발생할 수 있는 문제들을 선제적으로 팀에 제기하고, 기술 스택과 AWS API들을 추천해 주면서 팀원들의 개발을 지원하는 업무를 해왔습니다. 또한 Linux OS와 C++에 대한 지식을 기반으로 컨테이너 내부, AWS Lambda, 임베디드 환경 등 특수한 환경에서 발생한 문제에 대해서 Troubleshooter 역할을 했으며, 이러한 역할을 할 수 있어 기쁘게 생각하고 있습니다.

🍱 Tech Lunchbox

🏗️ Architecture: Docker Compose, Nginx, Redis

- OCI Specification, Docker API와 1:1 대응이 이루어진다는 면에서 ***Docker Compose***를 개발, 배포에 모두 활용하고 있습니다.

    - MSA에서 각 서비스의 의존성을 통제하기 위해 서비스별 env/envfile과 volume, network 섹션을 주로 개발하고 유심히 코드 리뷰합니다.

- ***Nginx***는 프로젝트 외부와의 Gateway로서, ***Redis***는 프로젝트 내부의 Message broker로서 활용해왔습니다.
    - *Nginx* reverse proxy rule을 작성하고 🔗Nginx push stream module 과 같은 추가 Nginx module을 활용할 수 있습니다.
    - *Redis*의 거의 모든 내장 자료형을 적재적소에 활용할 수 있고, 컨테이너별 연동 방식을 제안할 수 있습니다.

🌩️ Serverless AWS

- *Python, Node.js* 언어로 ***Lambda*** function을 Production에서 꾸준히 사용했습니다.
    - 🔗Container 형태의 Lambda 로 AI inference를 구동할 수 있습니다.
    - *SNS, SQS, EventsBridge* 등의 AWS 서비스들로 Lambda의 Trigger/destination을 구성할 수 있습니다.

- ***Step Functions***로 AWS 서비스들을 비즈니스 로직에 따라서 연동할 수 있습니다.
    - *Step Functions*와 *API Gateway*를 연동해 코딩 없이 Microservice를 구성할 수 있습니다.

- TypeScript ***AWS CDK***로 브랜치별로 Serverless CI를 구성할 수 있습니다.

🤟 Languages: JavaScript, Python, Shell, C++...

- ***Node.js*** 를 이용해 복잡한 비즈니스 로직을 총괄하는 소프트웨어를 개발합니다.
    - *JavaScript*의 뛰어난 유연성과 비동기 처리 성능 때문에 해당 역할을 주로 할당했습니다.
    - 특정 프레임워크에 종속되기보다는 Frontend-, server-side에 관계 없이 JavaScript 기본기에 충실한 Full stack 개발을 지향합니다.

    - 애플리케이션 개발에 있어서 제가 **가장 전문적인 언어**입니다.

- ***Python*** 은 주로 AI와 관련된 작업에 대해서 명확한 역할을 배정해왔습니다.
    - MSA 구조 속에서 주어진 역할을 정확히 할 수 있는, 너무 크지 않은 프레임워크를 선택해서 사용했으며, Python 개발자와 가장 많이 협업해봤습니다.
    - 필요한 기능 구현은 충실히 할 수 있는 수준이나, Python 고유의 특성을 이해하고 코드베이스를 리드할 수 있는 수준은 아닙니다.

- ***Shell script*** 와 ***Makefile*** 을 이용해 Linux OS와 직접 상호작용이 필요한 스크립트를 작성하고 DevOps 업무를 수행합니다.
    - 애플리케이션 개발에 쓰이는 다른 스크립트 언어 수준으로 Shell script에 능숙합니다.
        - 프로그래밍 언어와 무관하게 Linux system call, Child process와 직접 상호작용할 수 있어서 ***Shell script*** 를 중시하고 있습니다.
    - Dependency가 복잡할 경우나, 다양한 언어가 쓰이는 MSA 구조에서 특정 언어에 의존하지 않기 위해서 ***Makefile*** 을 유용하게 사용했습니다.
        - 🔗개인 맞춤형 개발 환경 설치에 쓰이는 Makefile 예시

- 하드웨어 연동 관련해서 ***C++*** 를 이용해 재활용 가능한 코드를 작성할 수 있습니다.
    - 산업용 카메라, X-ray detector 등을 C++ SDK를 통해 제어하고 Linux IPC를 통해 다른 Microservice에 데이터를 제공하는 개발을 했습니다.
    - Modern C++ 문법과 객체를 이용해 안전하고 효율적인 개발을 지향했습니다.

🧪 Projects

- 🔗수행 프로젝트 목록
- 🔗사이드 프로젝트

🔗Home

Posts 🔗

기술 리서치 및 사이드 프로젝트 모음

🔗

Basic LLM Ops in AWS

Background Row of digital-based slot machines inside a casino in Las Vegas: Source LLMs Will Always Hallucinate, and We Need to Live With This . 프롬프트에 수정이 필요할 …

September 19, 2024 · 4 min · Sehyun Hwang

🔗

About this site: man.hwangsehyun.com

6년 간의 개발 경험과 개발자 황세현에 대해 공개적으로 소개할 수 있었으면 해서 품이 들어도 직접 사이트를 만들기로 결정했습니다. 사이트 구현에서는 기능에 대한 욕심 부리지 않고 HTML 표준에 충…

March 14, 2024 · 1 min · Sehyun Hwang

🔗

SSH Tunnel 기반 배포 & 운영 툴

Overview 인터넷은 가능하나 Inbound가 제한된 환경에 놓인 다수의 임베디드 또는 서버 형태의 Linux 머신들을 관리하기 위한 위한 OpenSSH server container 와 Node.js CLI 회사 특성 상 일…

March 1, 2022 · 1 min · Sehyun Hwang

수행 프로젝트

> **대학원 재학 시절과 재직했던 회사들에서 수행했던 프로젝트 목록입니다.**

🔗 🍚 LLM을 이용한 영양 상담 서비스	🔗@CrowdWorks
6/1/2024 ~ 11/8/2024	5.3 Months
다이어트에 필요한 고객 정보를 기반으로 고객이 섭취한 식단을 상담하고 식품을 추천하는 서비스. CloudFormation 기반의 IaaS 툴을 이용해 AWS 리소스를 구성하고 대기열 서비스를 Python으로 구현했습니다. 비개발자들도 프롬프트와 모니터링에 기여할 수 있는 프로세스를 만들고 개발, 테스트, 배포 프로세스를 체계화하려고 노력했습니다. • 🔗AWS Systems Manager Parameter Store , OpenTelemetry, CloudWatch를 이용한 🔗프롬프트 관리 및 LLM 운영 체계 개발	CDK Python AWS X-Ray

🔗 🗃️ LLM을 이용한 기업용 챗봇 서비스	🔗@CrowdWorks
2/29/2024 ~ 4/29/2024	2.0 Months
• 네이버 클라우드에서 고객사의 문서 데이터를 취급하기 위한 인프라 및 보안 설계 • 무정형의 HWP를 HTML 형태로 변환해 LLM에 적합한 형태로 정형화하고 검수하는 Frontend 개발	Lit Postgres Nginx Python

🔗 🚓 보행자 & 번호판 인식 솔루션	

🔗@NextLab	3/1/2022 ~ 2/29/2024
24.3 Months	Nginx Docker Compose S3 Lit TypeScript Redis vanilajs CDK
넥스트랩의 소속 팀에서 가장 주요하게 개발했던 Yolo 기반 번호판 인식 솔루션으로, 근래에는 서비스 아키텍처 설계와 트러블 슈팅 역할을 했습니다. • 여러 단계에 걸쳐 이루어지는 번호판 인식 성격에 맞는 Docker Compose service 구성, 비동기 처리 개발 • AWS CodeBuild를 이용한 Serverless CI, Vitest와 Pytest를 이용한 TDD 도입 • 납품처별 공통 기능, 외부 연동 프로토콜 정의 및 개발	

🔗 🪄 LLM 기반 브랜드 이름 작명 서비스	🔗@NextLab
10/31/2023 ~ 2/28/2024	4.0 Months
	Lambda Python React API Gateway

한글 브랜드 이름에 특화된 브랜드 이름 작명 서비스 🔗naimy.ai . 작명에 이어 작명된 이름을 도메인 API, 특허청 API, 소셜 네트워킹 서비스 크롤링 등의 외부 서비스 연동을 통해서 적절성을 분석할 수 있습니다. • 인증, 외부 API 호출, DB 조회, Frontend의 영역으로 구분된 AWS Serverless 기반 아키텍처 설계 • Next.js Frontend 배포 • 특허청을 제외한 외부 서비스 연동 개발 • DB connection pool 관리, 애셋 관리 등 운영 이슈 해결	EC2 RDS
🔗 🫁 X-Ray 식품 검사 시스템	**🔗@NextLab**
`6/1/2023 ~ 12/31/2023`	7.1 Months
X-Ray 식품 검사 장비에 AI 기반 이물 탐지 기능을 추가하는 프로젝트. Redis를 중심에 두고 C++과 Node.js로 하드웨어 연동 기능을 개발하고 서비스에 필요한 기능들을 분리해서 구현하는 아키텍처를 설계했습니다. > Production에는 도달하지 못했습니다.	Redis Docker Compose C++ Node.js TypeScript WebAssembly
🔗 📰 Serverless 기반 신문 PDF 정형화	

🔗@NextLab	7/1/2022 ~ 12/31/2022
6.1 Months	Step Functions Lambda Express RDS Postgres S3 Python
이미지만을 고려하는 기존 OCR 서비스와는 달리, PDF에 담긴 정보까지 이용해 모바일 기사 형태로 PDF를 정형화하는 Serverless 기반 백엔드 구현. AWS Step Functions를 이용해 복잡한 딥러닝과 알고리즘 파이프라인을 구축하고, 사용자가 모바일 기사 형태의 출력을 검수 및 수정할 수 있는 Frontend와 연동했습니다.	

🔗🤖 웹 기술을 이용한 강화 학습 시뮬레이션 환경 구성	🔗@KBD Lab
4/1/2020 ~ 12/31/2020	9.1 Months
접근성이 우수하고 표준화된 브라우저와 컨테이너 기술을 이용해 관찰과 실험이 용이한 딥러닝 환경을 구축한 석사 학위 졸업 연구. 대학원 시절 딥러닝을 응용하는 분야에서 실제 연구보다 소프트웨어 환경 셋업에 노력을 허비해야만 했던 동료들의 모습을 보고 졸업 연구로 진행하게 되었으며, 브라우저에 Three.js와 Cannon.js로	three.js Docker API Express Socket.io vanilajs

강화 학습 환경을 구축해 Python 강화 학습 환경과 Socket.io로 연동하는 방식으로 구현했습니다.	

🔗 🗺️ 플랜트 육상 운송성 평가 시스템	🔗@KBD Lab
4/1/2019 ~ 12/1/2019	8.1 Months
대학원에 막 입학해서 Google Maps API를 중심으로 Full stack 개발을 시작하며 JavaScript의 첫 걸음을 뗀 프로젝트. PHP & MySQL legacy를 탈피해서 Frontend 중심으로 설계를 변경하고 기능을 개선하는 방향으로 기획 및 구현했습니다.	Google Maps jQuery PHP EC2

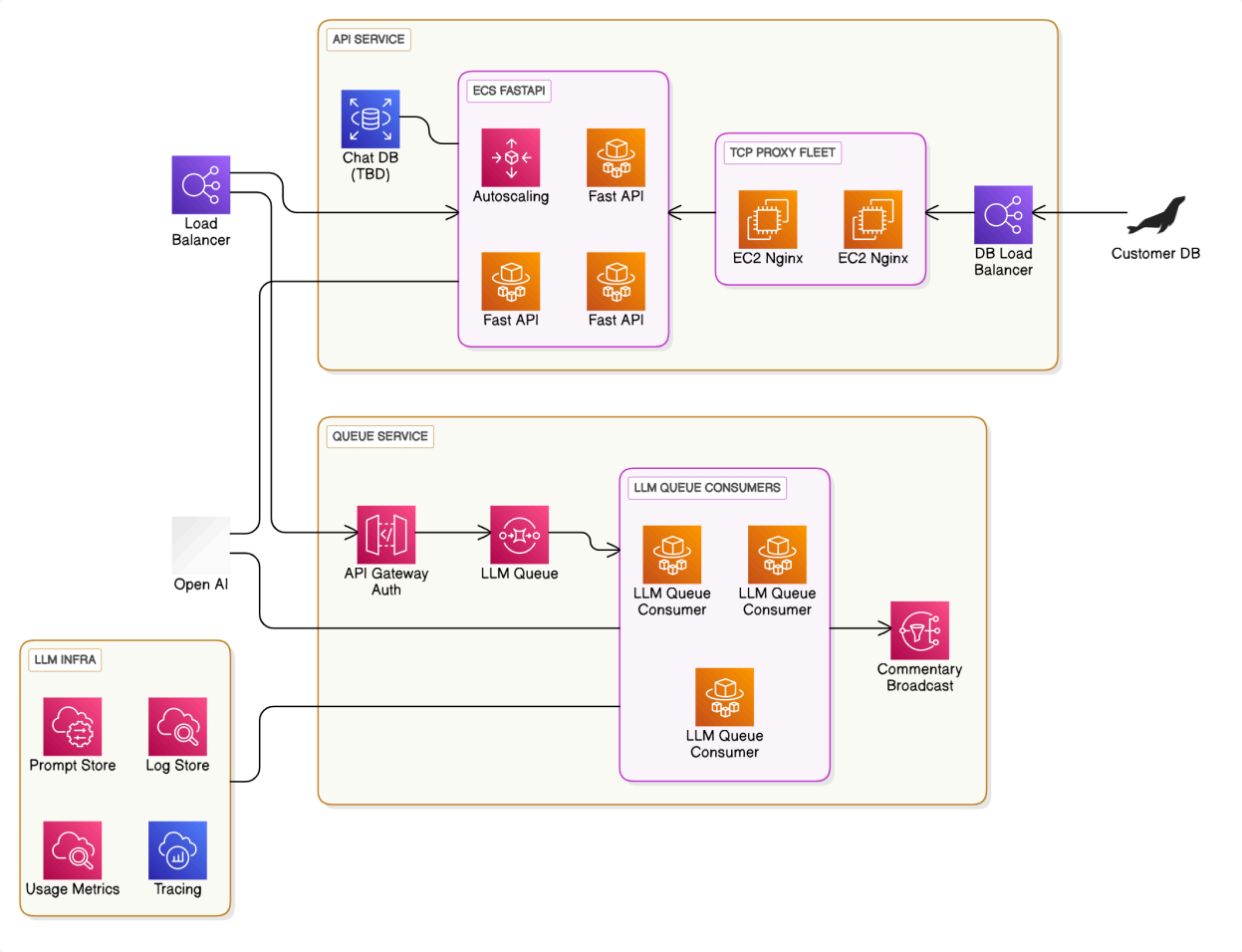
LLM을 이용한 영양 상담 서비스

November 8, 2024 · 3 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

Overview

- 크라우드웍스에서 🔗초기 6개월 동안 프로젝트 를 수행하면서 개발자와 비개발자들이 프롬프트 엔지니어링과 서비스 개발에 걸쳐 가지고 있던 요구 사항을 종합하고 AWS 생태계 내에서 지속 가능한 해결책을 도출하려고 노력했습니다.
- 3단계로 나누어진 프로젝트에서 첫 번째 단계로 Queue consumer service와 프롬프트 운영 프로세스를 Python으로 개발했습니다.
- 3단계 전체에서 CloudFormation을 기반으로 하는 🔗AWS CDK 와 🔗Copilot CLI 를 조합한 ECS 서비스 2개를 중심으로 다양한 AWS 리소스를 생성했습니다.



Technology

CloudFormation Stacks

> Copilot CLI + AWS CDK

- VPC 구성과 같은 정형화된 초기 설정과, 컨테이너 이미지만을 변경하는 빈번한 코드 배포는 Copilot CLI를 이용했습니다.
  - Copilot CLI는 AWS가 직접 개발에서 권장하는 ECS 서비스 배포 방식입니다.)

- ECS 서비스가 의존하는 추가적인 AWS 리소스들은 CDK로 정의해서 별도로 배포했습니다.
- 이때 Copilot CLI로 배포한 CloudFormation stack에 🔗YAML patch override 를 이용해 export를 추가하고, 이를 🔗 `Fn.importValue()` 함수로 import해서 CDK가 의존성을 가질 수 있도록 구성했습니다.
  - 🔗Reference: using-cfn-stack-exports

Monorepo & Build settings

▶ **Monorepo Layout**

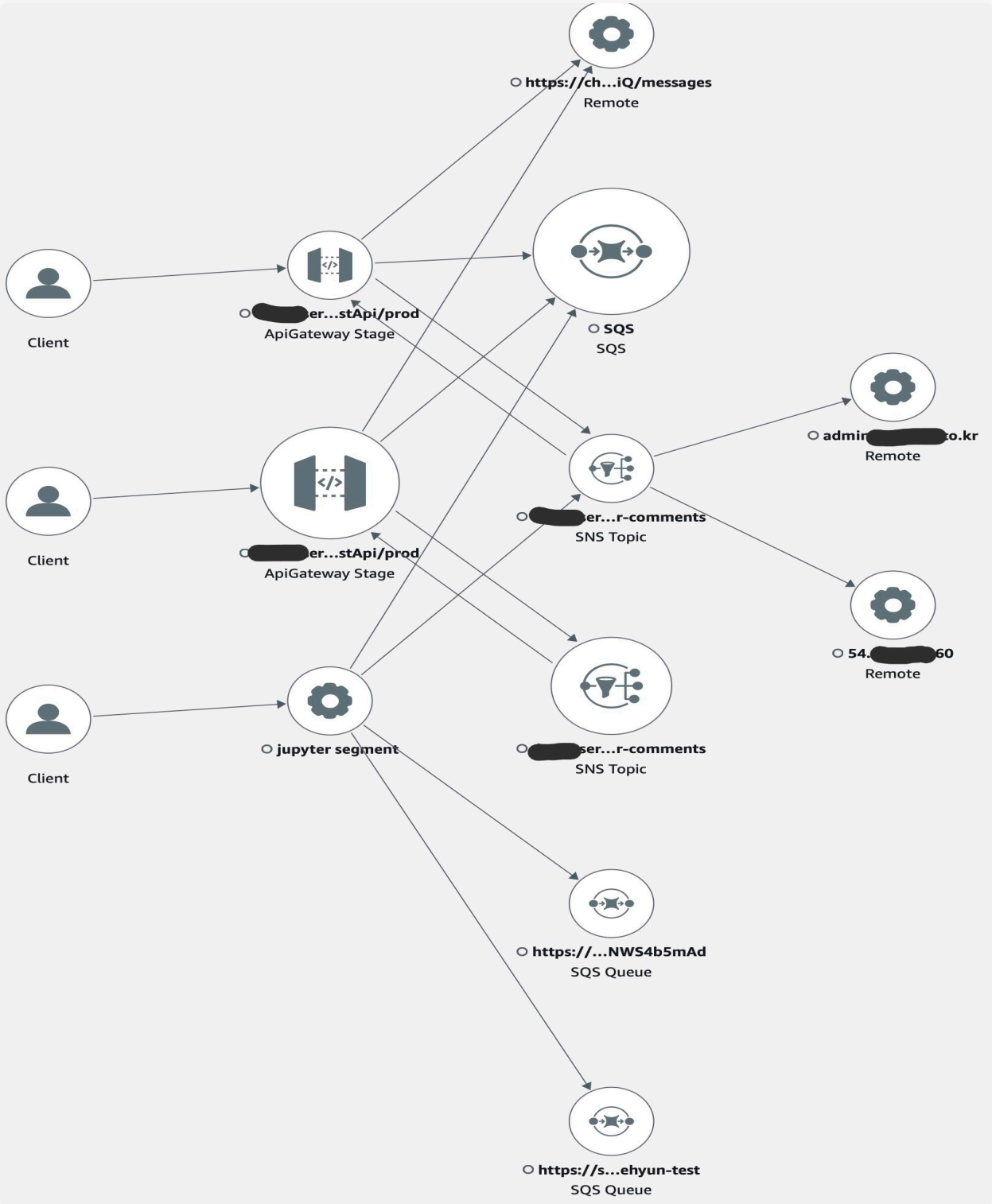
- cdk와 copilot 폴더에는 IaaS 구성에 필요한 TypeScript 파일과 YAML 파일이 있음
- 2개의 🔗Copilot service 의 Python 소스 코드가 worker와 chat-api 폴더에 존재
  - 각각 테스트 폴더를 포함함
  - 각각 컨테이너 이미지를 생성할 수 있음
- `docker-bake.hcl` 파일을 이용한 🔗High-level build 기능으로 Monorepo 전체를 빌드하고 3개의 컨테이너 이미지 Artifact를 생성할 수 있음
- 테스트, 빌드, 배포는 Makefile에 적힌 명령어로 모두 수행할 수 있음

Python Development

- Strong typing을 강제할 수 있는 🔗pyright language server 설정을 권장하고 TDD를 도입해 프로젝트 전체를 구동하거나 배포하기 전에 기능 단위로 개발할 수 있도록 했습니다. 그 결과 Edge case를 폭넓게 고려하는 개발을 효율적으로 진행할 수 있었으며, 크라우드웍스 최초의 B2C 프로젝트에 안정적인 서비스를 가능케 했습니다: 🔗기사 링크
- 사용자에게 추천하는 식품들의 후보는 DB화해 고객사가 정확하게 제어할 수 있도록 하는 기능을 개발했습니다.
- 복수의 LLM API key를 사용해야 할 경우, 키의 Tier를 고려해 사용량을 제거하고 추가/삭제할 수 있는 관리 체계를 개발했습니다. 기존에는 요청마다 API key를 임의로

선택하는 로직을 사용해 애플리케이션 전체에 코드나 설정 변경이 필요했으나, 이 프로젝트에서는 컨테이너 1개 당 API 키를 1개씩만을 할당해 모니터링과 운영을 ���이하게 했습니다. ECS 컨테이너는 초기화 시 🔗ECS task metadata API 와 ECS API를 통해 API key의 사용 현황을 확인하고 가장 적게 사용된 것을 선택합니다.

- AWS X-Ray tracing과 OpenTelemetry를 도입해 dev stage에서의 프롬프트 실험을 용이하게 하고 prod stage를 모니터링했습니다. X-Ray console에서 API Gateway, SQS, ECS task, SNS로 이어지는 구간 별 성능을 파악할 수 있었고, 🔗OpenTelemetry instrumentation OpenAI 을 적용해 생성 결과와 토큰 사용량을 추적했습니다: 🔗자세한 사용 방식과 코드


https://ch...iQ/messages
Remote
Client
er...stApi/prod
ApiGateway Stage
SQS
SQS
admin[redacted]co.kr
Remote
Client
er...stApi/prod
ApiGateway Stage
[redacted]er...r-comments
SNS Topic
54.[redacted]60
Remote
Client
jupyter segment
[redacted]ser...r-comments
SNS Topic
https://...NWS4b5mAd
SQS Queue
https://s...ehyun-test
SQS Queue

Lessons

- 🔗신문 PDF 정형화 프로젝트 에서 수백 개 단위의 AWS 리소스를 IaaC 없이 진행하며 많은 어려움을 겪었는데, 이번 프로젝트에서는 dev, prod 환경을 git에 커밋된 그대로 재현할 수 있었습니다.
- 크라우드웍스에서는 데이터 담당 개발자뿐만 아니라 비개발자들이 기획과 QA를 담당했습니다. 이 팀원들이 개발에 기여할 수 있도록 프로세스를 만들고 고객과 커뮤니케이션하는 데 참여할 수 있었습니다.
- Python typing을 본격적으로 적용한 첫 프로젝트였는데, TypeScript와 비교해서 장단점이 눈에 보였습니다. Python typing은 TypeScript와 달리 언어 표준에 의한 구현이다 보니 🔗 `satisfies` , const 등의 키워드를 통한 정밀한 제어가 부족하다고 느꼈으나, 🔗pydantic 과 같이 사용했을 때의 강력한 기능은 매우 유용했습니다.

🔗Aws 🔗Llm

LLM을 이용한 기업용 챗봇 서비스

April 30, 2024 · 2 min · Sehyun Hwang | Translations: 🔗문화어

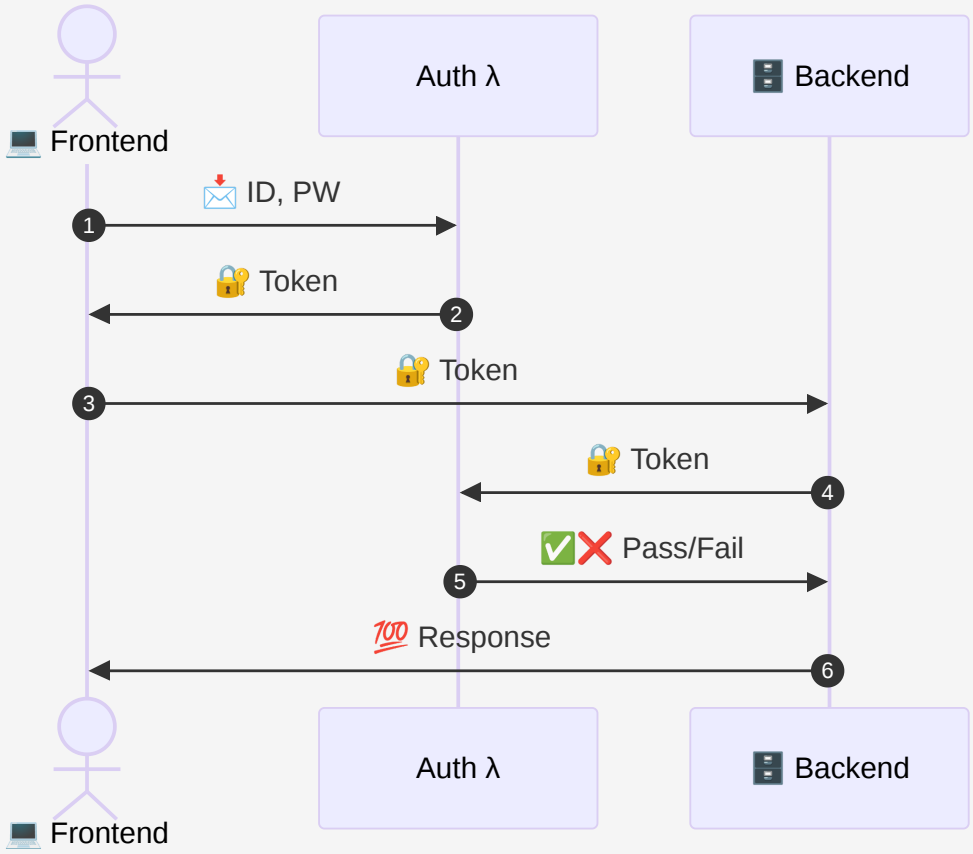
▶ Table of Contents

Overview

💬 사내 문서에 대한 Q&A 챗봇 개발

- 네이버 클라우드에서 고객사의 문서 데이터를 취급하기 위한 인프라 및 보안 설계
- 챗봇 형태의 PoC 서비스 아키텍처 설계
- 무정형의 HWP를 HTML 형태로 변환해 LLM에 적합한 형태로 정형화하고 검수하는 Frontend 개발

Architecture



Technology

- 인프라
  - 🔗Minio 라는 S3-compatable object storage에 원본 문서와 Chunking document를 적재
  - 네이버 클라우드 VM 초기 구성, 목적 별로 할당 및 관리
  - VPN을 통해서만 VPC 내부에 접근할 수 있도록 구성
- 챗봇 형태의 PoC 서비스 구성
  - React frontend가 Fast API chat server, Self-hosted 🔗Sentry 등에 접근할 수 있는 Nginx 개발

    - 업데이트가 잦은 Fast API 서버에 의존하지 않도록 JWT 서비스를 🔗Naver Cloud Function 를 이용해 별도로 분리
    - 채팅 API 사용 시나리오 관리

HTML 정형화 Frontend 개발

- 기술 스택
    - 🔗Lit : 대량의 검수 대상 HTML element에 Virtual DOM 없이 직접 접근하고 Boiler plating 없이 빠르게 개발하기 위해서 선택
    - 🔗pREST : 백엔드의 지원 없이 HTML 데이터를 DB에 적재하기 위해 선택
    - DB
        - 문서 파싱 규칙은 Indexed DB에 적재해 개인이 관리
        - 문서 파싱 결과는 Postgres에 적재해 RAG 시스템과 연동
- html 트리를 탐색하며 html element마다 세상에 id를 부여하는 해싱 알고리즘 개발
- 문서 파싱 규칙을 UI를 통해 작성하고 규칙의 적용 결과를 바로 확인할 수 있음
    - 🔗 `textContent` 에 대한 Regex matching, CSS selector, DOM tree 내의 위치 등을 고려
- Minio와 연동된 Batch 기능 개발

Lessons

- 새로운 회사에서 일을 시작하며 직접 고객사와 소통하지 않고 PM과 협업했습니다.
- Lit framework를 이용해서 복잡한 상태 관리를 해보았다.
- AWS 이외의 클라우드로 네이버 클라우드를 처음으로 경험해보았으나, 불안정하고 완성도가 떨어지는 모습에 더는 경험해보고 싶지 않았습니다.

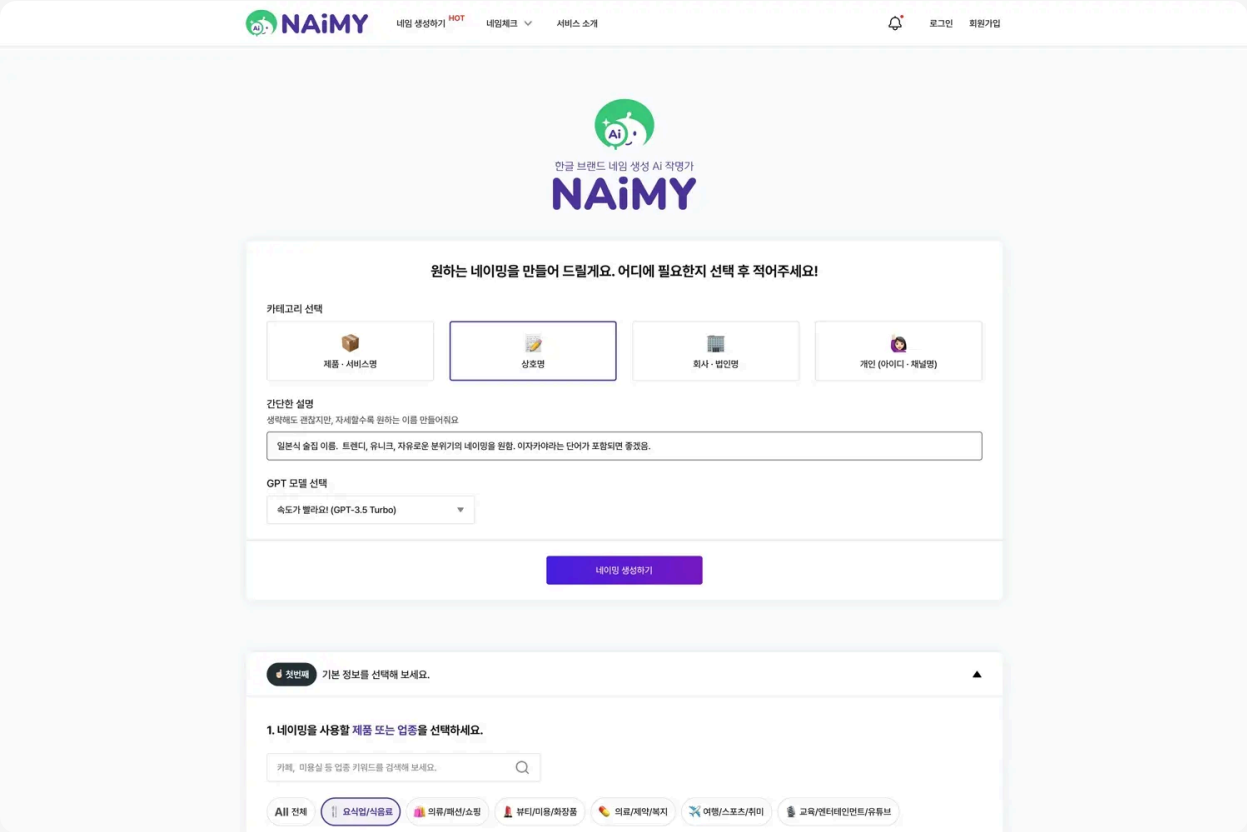
- 언어 모델과 RAG 시스템의 특성을 서서히 이해해가고 있습니다. 성능에 대한 사람들의 높은 기대치에는 절대 대중들이 예상하는 만큼 쉽게 도달할 수 없어 보이고, 언어 모델 이외의 전문 지식과 소프트웨어 기술에 대한 중요성을 매일 느끼며 겸손을 배우고 있습니다.

🔗Security 🔗Llm

🔗Home » 🔗수행 프로젝트

LLM 기반 작명 서비스

February 29, 2024 · 2 min · Sehyun Hwang | Translations: 🔗문화어



▶ Table of Contents

> 서비스 링크: 🔗https://naimy.ai/

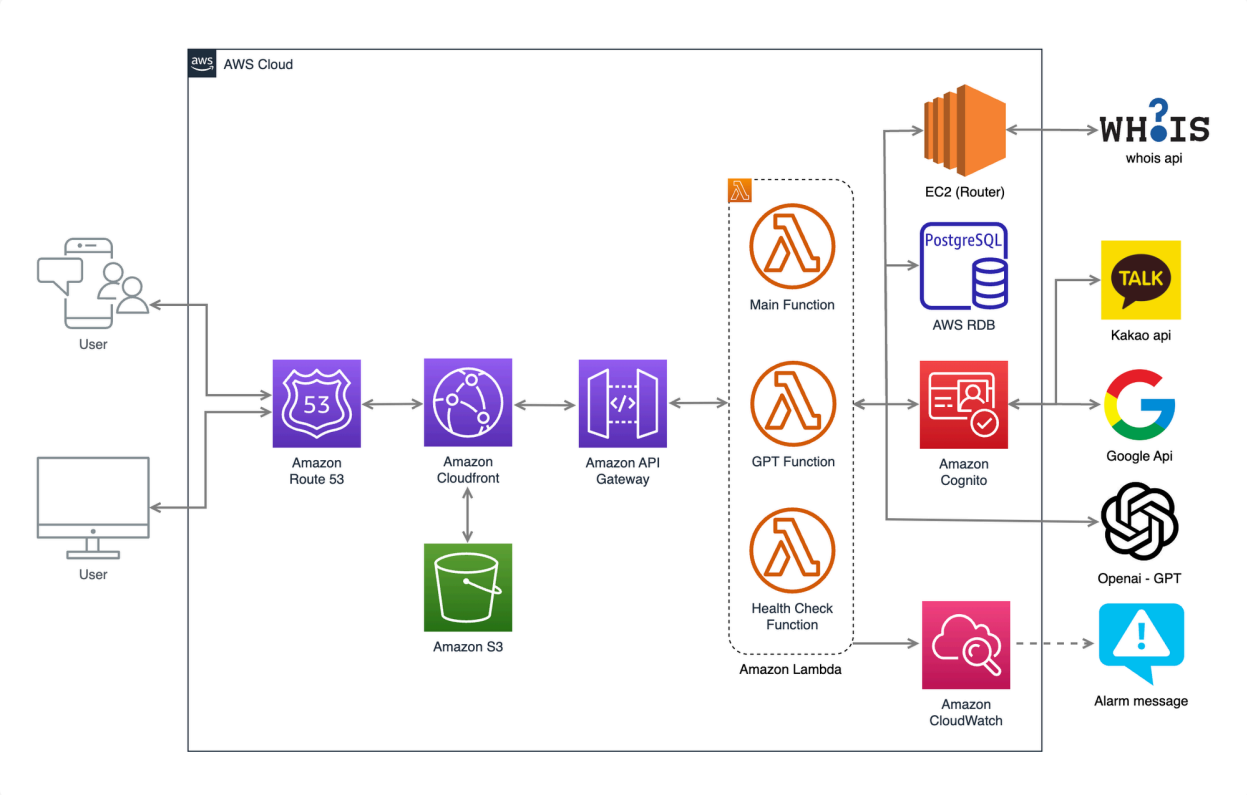
Overview

- 한글 브랜드 이름에 특화된 브랜드 이름 작명 서비스 🔗naimy.ai . 작명에 이어 작명된 이름을 도메인 API, 특허청 API, 소셜 네트워킹 서비스 크롤링 등의 외부 서비스 연

동을 통해서 적절성을 분석할 수 있습니다.

- 크게 Postgres RDS를 사용하는 Lambda backend와 React frontend로 이루어진 애플리케이션으로, 외부 서비스 연동을 고려한 AWS 아키텍처를 설계하고 특허청을 제외한 외부 서비스 연동 개발을 진행했습니다.
- 작명 옵션 선택지 변경, Database connection pool 디버깅, AWS 계정 마이그레이션 등 사이트 운영 이슈 대응

Architecture



▶ A screenshot of naimy options

Lambda Break-down

> 🔗Chalice micro framework 로 테스트, 배포되었습니다.

- `naimy`

    - Main API server
    - 외부 API 호출
    - 외부 API 테스트 코드를 주기적으로 구동

- `naimy-gpt`
    - Open AI SDK로 Chat GPT lambda 호출
    - 자연어로부터 네이밍 옵션 자동 생성
    - 기본적인 브랜드명 생성

- `naimy-trigger`
    - Raw query로 로그인을 빠르게 수행하기 위해서 분리한 Lambda
    - Cognitio user pool의 `pre-signup` , `post-signup` event에 의해서 Trigger 됨

AWS Services

- Postgres Database 관리
    - API Lambda마다 데이터베이스 연결을 수립하므로 connection 개수가 전통적인 서버보다 많았었음
    - 오래 걸리는 Open AI API 호출을 담당하는 Lambda는 Lazy하게 데이터베이스 연결을 수립하도록 변경
    - Database backup 정책 수립, Stage를 마이그레이션
    - Lambda에 적합한 🔗SQL Alchemy option 조사

- 고정 IP 보안이 필요한 🔗(주) 후이즈 API 연동
    - Elastic IP를 등록한 EC2에 후이즈 API로 Reverse proxy를 하는 Nginx 구성
    - 해당 EC2의 Private IP를 🔗CloudMap service에 등록
    - 후이즈 CloudMap service를 호출하는 API Gateway route를 생성하고 IAM authentication으로 보호

    - Lambda의 Service role로 호출

- CloudFront로 Next.js static site 서비스
    - S3 CloudFront origin 설정이 Next.js router 설정과 합치하도록 Next.js를 설정하고 배포 스크립트를 구성
    - 점검 중에 사용할 수 있는 별도 페이지 구성

Lessons

- Cognito 인증이 들어간 B2C 프로젝트 운영 경험. 홍보가 이루어지며 회원과 트래픽이 서서히 증가하는 모습을 보았습니다.
- Lambda로 API 서비스를 만드는 것의 장단점을 명확하게 느꼈습니다. 서버를 사용했을 때보다 비동기 처리와 인프라 방면에서 이슈 발생이 적었지만, Fast API 등의 서버 프레임워크보다 Chalice의 기능이 적어 고민하고 기능을 구현해야 하는 상황이 많았습니다.
- 🔗Recoil 을 기반의 상태 관리 코드를 리팩토링하며 React 상태 관리의 기본적인 원리에 대해서 고민하기 시작한 좋은 기회였습니다.

🔗Aws 🔗Llm

보행자 & 번호판 인식 솔루션

February 29, 2024 · 4 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

Overview

- 🔗넥스트랩 의 소속 팀에서 가장 장기적으로 진행한 중요 프로젝트로, CCTV 등의 카메라에서 획득한 보행자와 번호판 이미지를 대상으로 추적 and/or 번호판 인식 모듈을 서비스화하는 프로젝트
- 납품 사례마다 다양한 요구 사항이 있었는데, 주로 핵심 AI 모듈을 둘러싼 서비스 구조를 비즈니스 로직에 맞춰 설계하고 구현하는 업무를 맡았습니다. 크게 요청에 따라서 task를 수행하는 *pull* mode와 최대한 빠른 속도로 task를 수행하는 *push* mode으로 나눌 수 있습니다.
- 🔗*다련장 발사대* 라는 별명의 SSH tunnel을 통한 배포 및 운영 툴을 만들어서 이 프로젝트에서 2년간 안정적으로 사용했습니다.

Technology

- 컨테이너 단위로 역할을 지정하고 Stage와 GPU 환경을 고려한 docker-compose 개발
  - 배포와 관련해서 Nvidia GPU를 컨테이너 내부에서 사용하기 위한 고려 사항들을 알고 있고 실무에 활용해왔습니다. (CUDA version, Container runtime, Container base image, Runtime flags…)
  - 컨테이너 별 주요 Failure point를 고려해 Health check 구현
- AI task 관리 기능 개발

    - 🔗*pull* mode 에서는 tmpfs 또는 S3-compatable 임시 저장소에 최근 사진만을 적재
    - 🔗*push* mode 에서는 대기열에 요청받은 Task들을 적재
- 시스템 일부분에 대한 테스트 또는 운영 지원을 위한 내부용 프론트엔드 구현
- 🔗AWS CDK 를 이용해 브랜치 별로 docker compose build를 수행하는 Serverless CI 개발

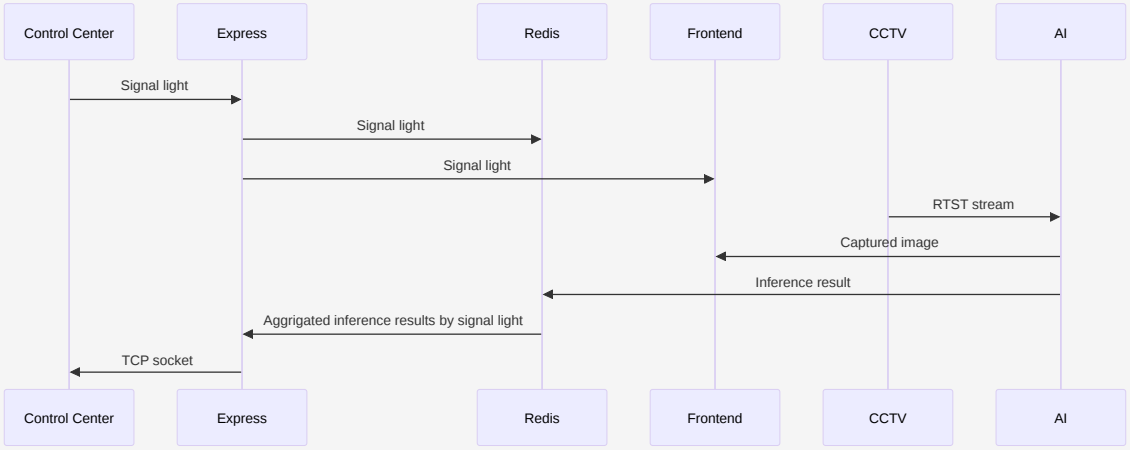
Progress

> 납품 케이스 별로 개발 사항들을 정리했습니다.

스마트 횡단보도용 보행자 인식 솔루션

> 수원시 스마트 횡단보도 사업에 40여대의 Jetson 보드 납품
> SSH tunnel 사용 1회차

- 보행자 추적 백엔드 개발 및 배포 총괄
- 신호등 값에 따라서 보행자 추적 결과를 Aggregate해 TCP socket으로 발신하는 컨테이너 개발
- 다련장 발사대 초기 버전으로 40여대의 Jetson 보드에 docker-compose 프로젝트를 배포하고 불안정한 카메라로 인한 운영 이슈에 대응



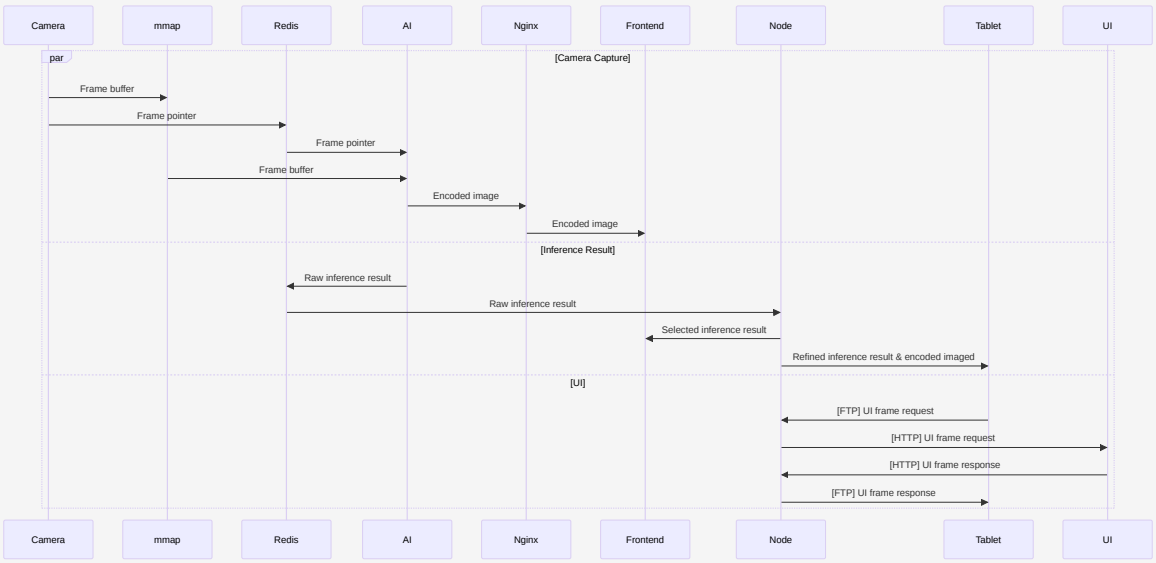
경찰차용 범죄 차량 단속 시스템

> 경찰차용 번호판 인식 시스템에 100여대의 Jetson 보드 납품 번호판 인식 솔루션 배포 1회차, 🔗 *push* mode SSH tunnel 사용 2회차

- C++ 언어로 산업용 카메라 영상 캡쳐 컨테이너 개발
  - ffmpeg으로 실 카메라를 모사하는 Mock 카메라 기능 개발
  - 🔗mmap 과 Redis hash map을 통해서 Frame buffer와 Frame pointer를 Overhead 없이 다른 컨테이너들에 공유
- 고객사 앱에 임베딩 가능한 UI frame을 렌더링하는 서버 개발
- Node.js를 이용한 TCP Socket & FTP 레거시 프로토콜 개발
  - UI frame을 FTP로 제공하는 기능 기능 개발
  - TCP socket을 통한 인식 결과 및 현장 사진 전송 기능 개발
  - 수배 차량 DB 업로드 기능 개발
- SSH tunnel을 통한 배포 시스템에 🔗Cloud9 IDE 를 연동해 고객사에게 제공

- 번호판 오탐을 보정하는 후처리 기능 개발
- 인증 시험용 로그를 Indexed DB에 적재하는 기능을 탑재한 간이 Frontend 개발

🔗 *push* mode 번호판 인식 처리 과정

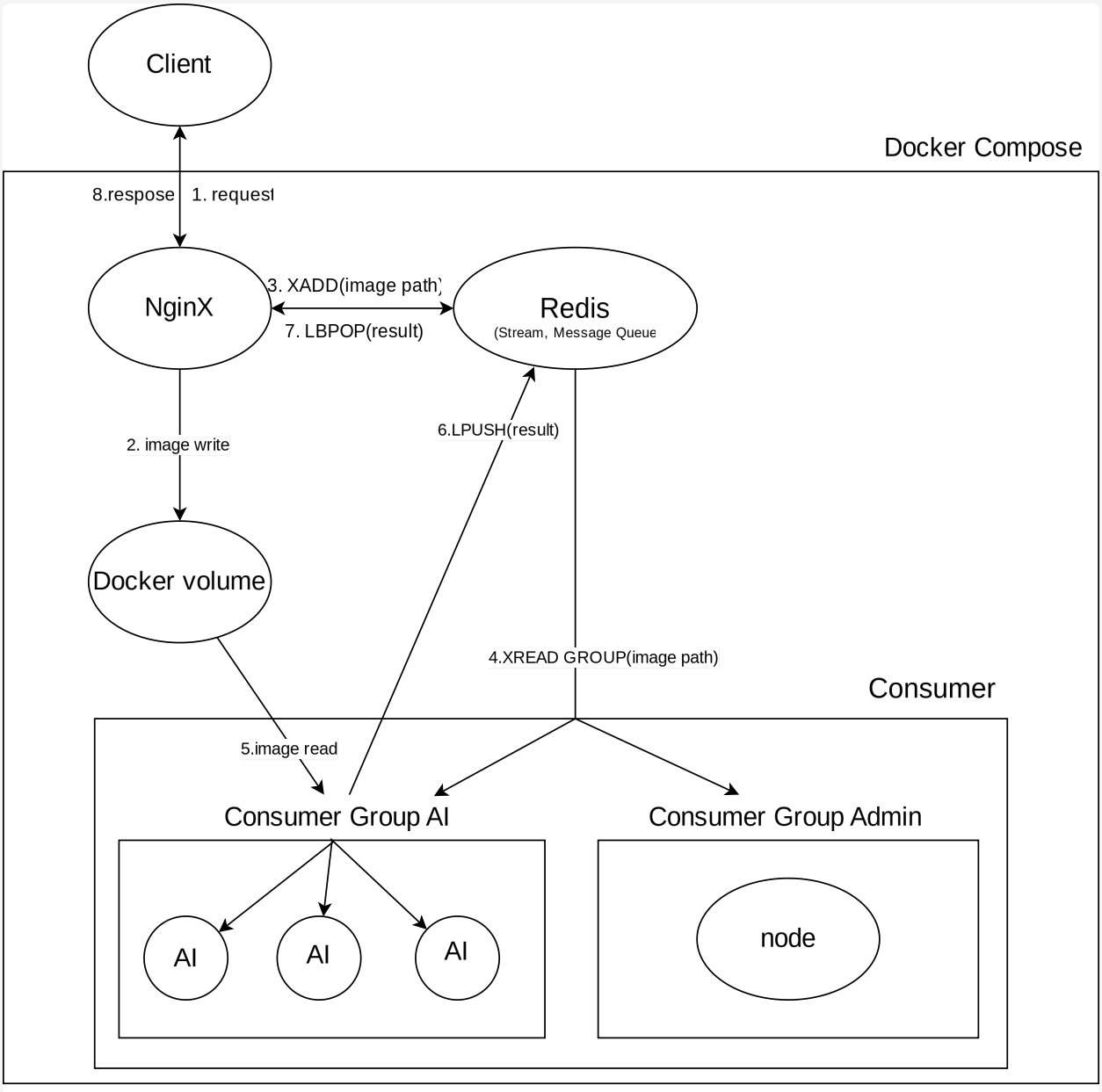


▶ **Docker Compose Containers**

▶ Details

서버형 번호판 인식 시스템

> 번호판 인식 솔루션 배포 2회차, 🔗 *pull* mode

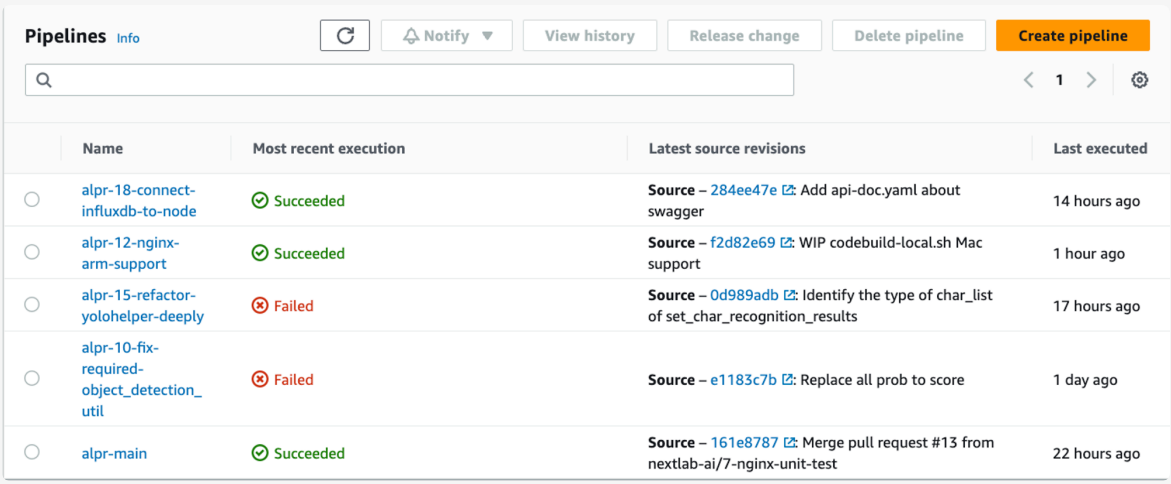


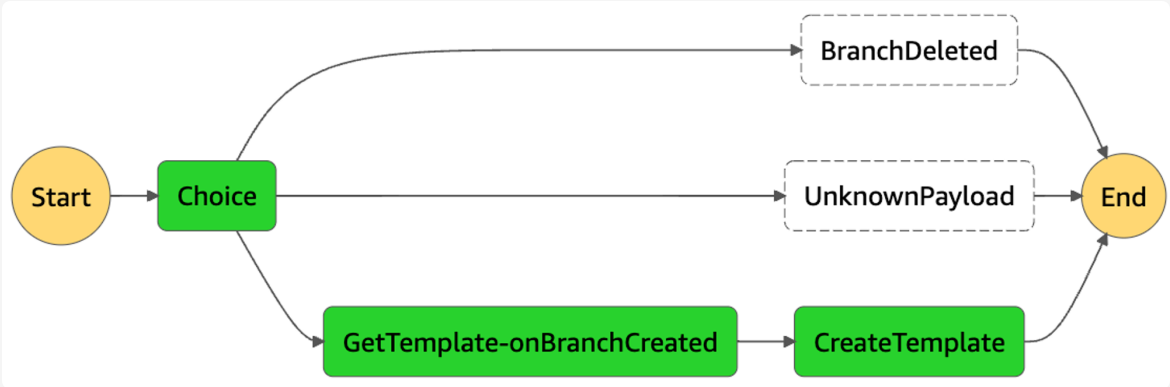
- Nginx 추가 모듈을 이용한 HTTP 애플리케이션 로직 개발

    - 🔗redis2-nginx-module , 🔗redis-nginx-module 등을 Lua scripting으로 연동해 fan-out 패턴 구현
    - 🔗Vitest 를 이용해 Nginx 설정에 대한 테스트 코드 작성

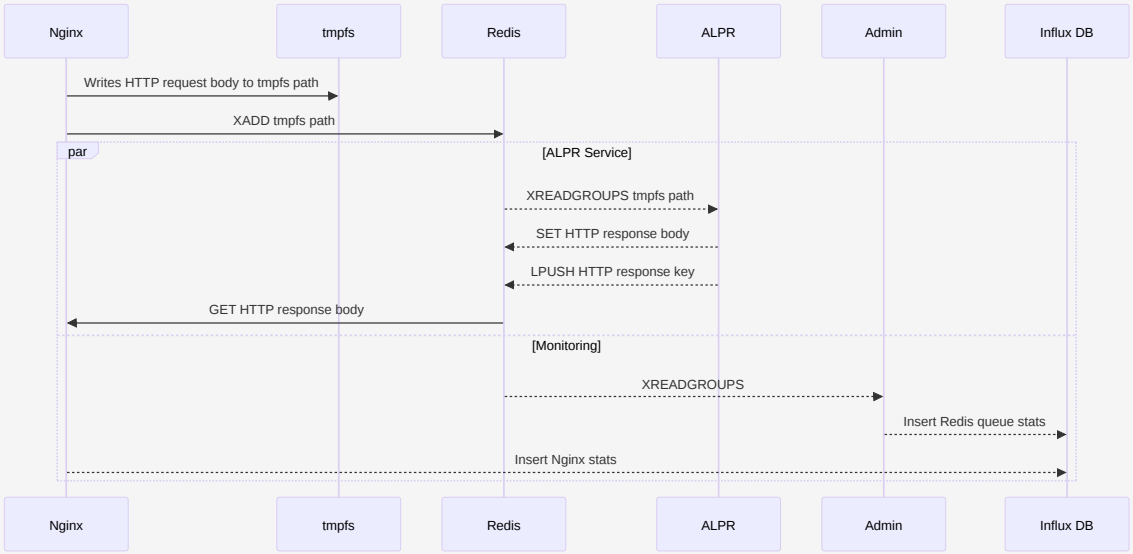
- 🔗redis-streams 라이브러리를 이용해 Python AI worker가 작업을 분배받아 결과 또는 오류를 전파하는 기능 개발
- 서버 모니터링 기능 개발
    - 모니터링용 Redis consumer group를 신설해 서버 metrics를 Influx DB에 적재하는 기능 개발 참여
    - 요구사항 파악, 모니터링 툴 선정: 🔗Kibana , 🔗Influx DB , 🔗Portainer , 🔗RedisInsight
    - 하나의 포트에서 모니터링 툴들을 사용할 수 있도록 하는 별도 Nginx를 포함한 Docker Compose 개발

- Redis를 이용한 동적 GPU 할당 기능 개발
- 배포 관련
    - 모니터링 스택과 연동이 가능한 서비스용 docker-compose 개발
    - 다양한 GPU architecture를 지원하는 AI 컨테이너 Dockerfile 개발
    - GitHub repo 브랜치 생성/삭제와 연동되는 AWS CodeBuild pipeline 구성





번호판 인식 요청 처리 과정



주차장 모니터링 솔루션

> 번호판 인식 솔루션 배포 3회차, *push* & *pull* mode
> SSH tunnel 사용 3회차

- 300개 이상의 촬영 각도를 이용해서 전주시 야외 주차장의 주차 현황 관리 프로젝트
- 고객사 연동용 REST API 설계
- 🔗Ray cluster를 5대의 물리 서버에 구성해 차량 추적 서비스 구축, 메모리 누수 해결

참조: 🔗아키텍처 PoC repo

- 🔗Lit element frontend에 S3-compatiable storage를 연동해 운영 상황에서 필요한 데이터 관리 툴 개발
- RTSP 프로토콜 CCTV로부터 캡쳐한 이미지를 S3-compatable storage에 적재하는 ffmpeg, nginx 개발

🔗Home » 🔗수행 프로젝트

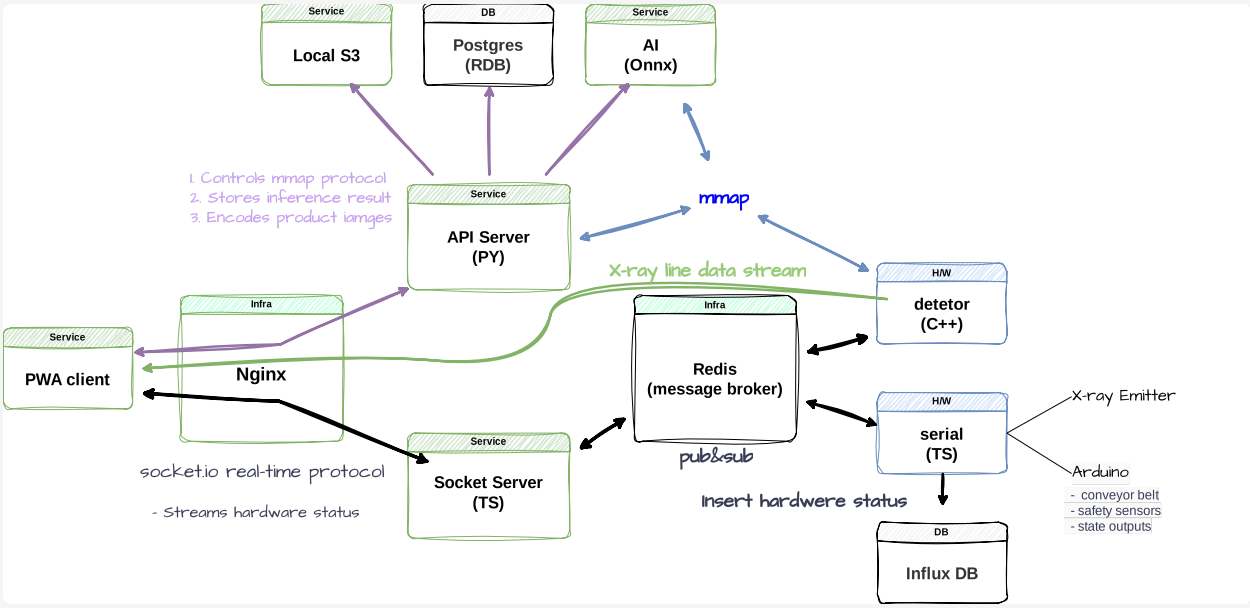
X-Ray 식품 검사 시스템

December 31, 2023 · 2 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

Overview

- 하드웨어 연동과 AI 이물 검사 기능을 핵심으로 하는 **식품 X-Ray 이물 검사 시스템 전면 재개발** 프로젝트. 🔗LabView 윈도우 응용 프로그램을 MSA로 변경.
- 크게 장비를 직접 통제하는 하드웨어 스택과 검사 수행에 필요한 비즈니스 로직과 검사 이력을 관리하는 애플리케이션 스택으로 나누어 서비스를 설계했습니다.



Technology

- 하드웨어 스택에서는 Redis message broker를 중심으로 하드웨어 연동 전용 컨테이너들을 배치하고 해당 컨테이너들에 대해 TDD를 도입해 하드웨어에 의한 Pain

point를 줄였습니다.

- X-Ray의 눈 `detector` 컨테이너를 C++로 개발했습니다. bmp 표준을 준수하되 깊은 비트 심도를 나타낼 수 있는 형식을 고안하고, 다른 컨테이너가 🔗mmap 프로토콜을 통해서 필요한 이미지 영역을overhead 없이 획득할 수 있도록 했습니다.
- 나머지 하드웨어 연동 컨테이너의 Redis 인터페이스 설계와 Node.js TypeScript 객체 지향 설계를 주도했습니다.

- 애플리케이션 스택에서는 에러 핸들링과 Canvas로 획득 이미지를 스트리밍하는 기능을 개발했습니다.

`docker-compose` services

Name	Language	Category	Features
`redis`	-	Infra	Event broker
`nginx`	Nginx	Infra	Static server, Reverse proxy, 이미지 스트리밍
`api`	Python	Service	CRUD, Auth, 이미지 처리 Microservice
`orchestrator`	Node	Service	하드웨어 상태 모니터링 및 통합 조작
`detector`	C++	H/W	Detector H/W 연동, 기초적인 이미지 처리
`serial`	Node	H/W	개별 시리얼 H/W 상태 관리자
`postgres`	SQL	DB	유저, 알고리즘 파라미터, 이미지 목록 관리
`influxdb`	Flux	DB	하드웨어 사용 이력 관리
`s3`	Node	Infra	mmap, 이미지 데이터 관리
`ai`	Python	Service	이물 검출 Yolo
`firmware`	C++, Python	Deployment	IO controller firmware uploader

Lessons

- 수많은 야근에도 불구하고 프로젝트의 목표가 달성 불가능한 목표로 설정된 탓에 Production에는 도달하지 못했습니다. 이 글은 프로젝트의 완성보다 제 커리어에서 가장 복잡한 문제를 MSA로 어떻게 풀려고 시도했는지를 중점으로 읽어주시길 부탁드립니다.

- 객체를 통해 메모리를 간접적으로 관리하는 현대적인 C++ 개발을 경험했습니다. WebAssembly, 🔗pybind 등을 통해서 일부 기능을 Native하게 구동하는 데 관심이 있었는데, 이번 프로젝트에서 쌓은 C++ 경험을 기반으로 미래에 이러한 개발을 맡을 수 있으면 좋겠습니다.
- Redis를 통한 하드웨어 추상화와 하드웨어에 대한 TDD가 효과적으로 작동했습니다.

Serverless 기반 신문 PDF 정형화

December 31, 2022 · 3 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

Overview

- 이미지만을 고려하는 기존 OCR 서비스와는 달리, PDF에 담긴 정보까지 이용해 모바일 기사로 나타낼 수 있는 정보를 만드는 Serverless 기반 백엔드와 검수용 프론트엔드 개발 프로젝트
- 요구 사항을 분석해 AWS Serverless 서비스들을 이용한 구조로 문제를 쪼개고 각 Lambda의 초중반부 개발을 진행했습니다.
- 수백 건 단위의 Lambda concurrency를 동원해 수십 페이지 단위의 PDF 정형화를 동시에 진행할 수 있었습니다.

Technology

- S3 presigned url을 이용해 PDF를 업로드하면 PDF 정형화 Step Function이 호출되도록 구성
- PDF를 HTML로 변환해서 좌표 정보를 추출하는 Lambda 개발 및 유지 보수
- 🔗Step Functions Activity Worker 를 이용한 이미지 처리 Worker 개발
- 모든 Lambda와 Step Function의 event payload 문서화 및 관리

Milestones

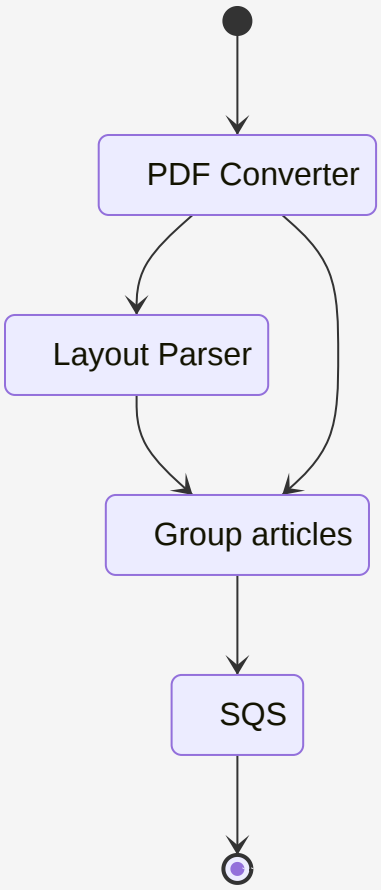
> 마일스톤 별로 집중했던 개발 내용에 따라서 정리했습니다.

1st. 신문 레이아웃 이미지 처리 돌려보기

- pdf를 문서보다는 이미지 데이터로 간주해 레이아웃에 대한 object detection 모델을 적용해 기반 기술로 활용
- 제목, 문단, 이미지 등 기사 요소를 *레이아웃*이라고 정의하고 🔗layout-parser Python library 적용

2nd. 통합 Step Function 구성

- pdf를 변환하고 데이터를 추출하는 `PDF Converter` Lambda를 🔗pdf2htmlEX 기반으로 구축
- 🔗stefuna 라이브러리로 Step Functions Activity Worker 형태의 `Layout Parser` 라이브러리 서비스화
- `Group article` : 모든 결과를 종합해 기사 별로 묶는 알고리즘을 마지막에 배치
- S3 presigned url을 통해 신문 pdf를 S3에 업로드하면 아래와 같은 Step Function이 구동되고 SQS를 통해서 처리 결과를 획득할 수 있는 아키텍처; 🔗AWS docs

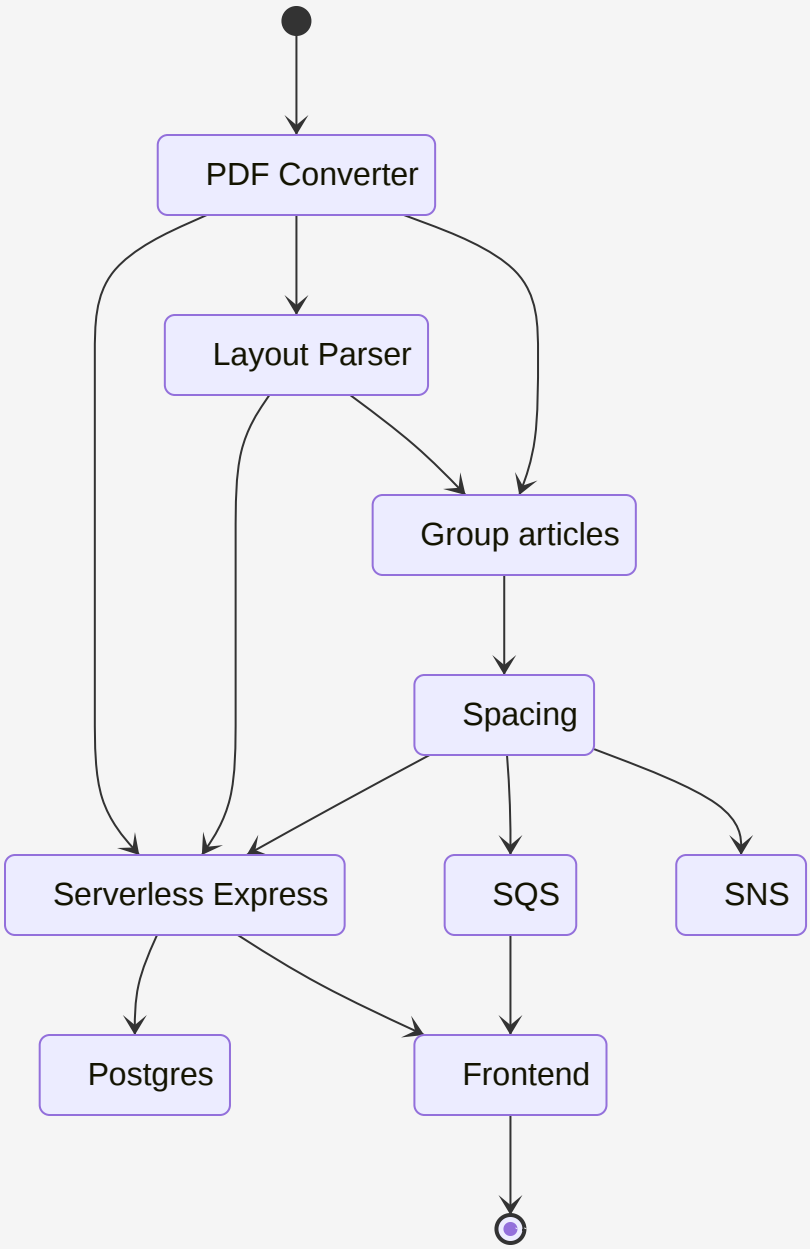


3rd. 결과 품질 개선, 결과 저장, 검수용 FE

- 워드 프로세서를 이용해 위에서부터 아래로 pdf를 작성한 경우와 그렇지 않은 경우로 나눠 알고리즘 개선
- 띄어쓰기 정제 모듈 도입
- 아래와 같이 용어를 정의하고 DB 스키마 정의

  > 신문 권호 - 페이지 - 기사 - 레이아웃

- 🔗serverless-express 를 이용해 Step Function 내부와 검수용 Frontend에서 Postgres DB에 접근할 수 있도록 함

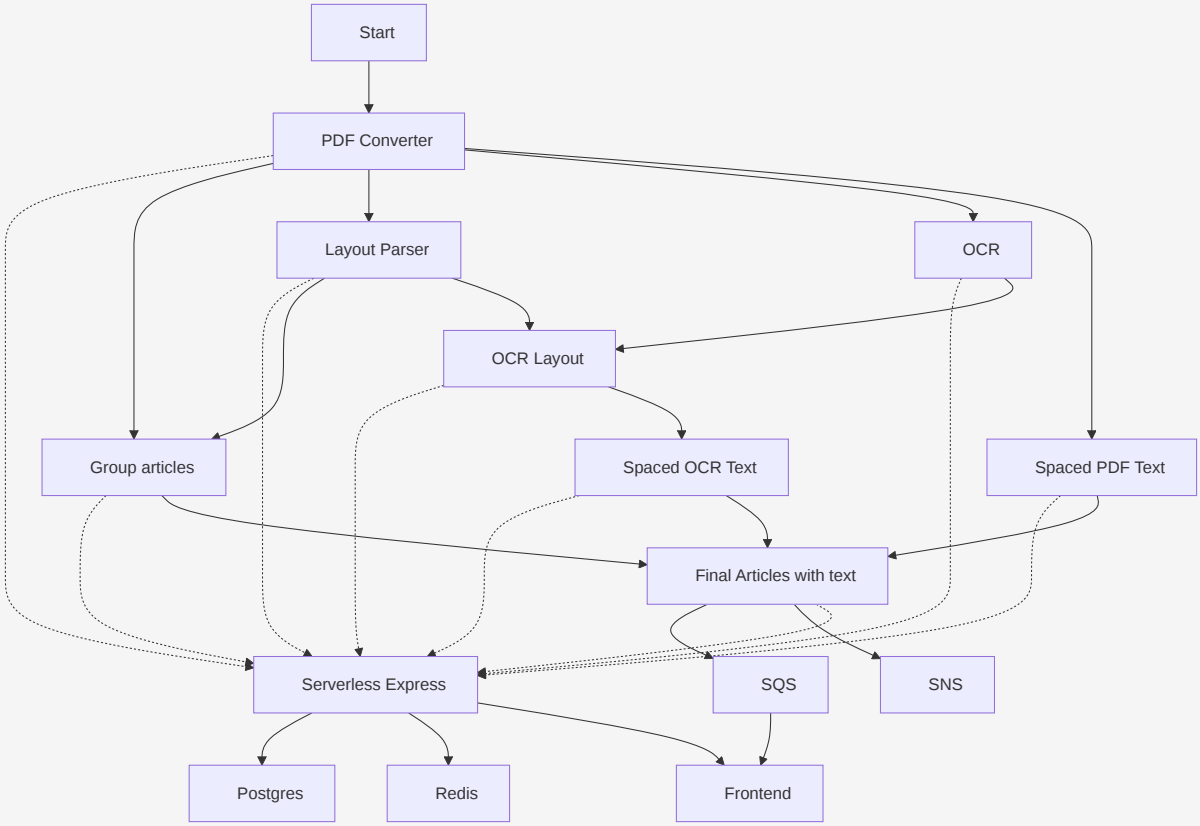


4th. 구조 개선 및 Naver OCR로 텍스트 소실 PDF 처리

- pdf 업로드 로직에 pdf를 장별로 분할하는 기능 추가
- Naver OCR을 적용해 텍스트 추출이 불가능한 pdf에 대응
  - 텍스트의 출처를 원본 pdf와 Naver OCR로 나누어 별도의 파이프라인을 만들어 레이아웃 별로 텍스트를 할당하고 띄어쓰기를 보정함

    - 레이아웃을 기사 별로 묶는 알고리즘을 텍스트 처리와 분리해 개발과 테스트를 독립적으로 수행함

- 마지막에 Step Function 바깥에서 DB 쿼리를 통해서 텍스트의 출처를 OCR과 PDF 중 선택하고 변경할 수 있음



Lessons

- pdf 자료 구조에 대한 이해를 얻었습니다.
- **AWS Serverless 서비스들을 조합한 아키텍처로 비즈니스 로직이 많이 들어간 대용량 데이터 처리를 효율적으로 할 수 있음**
- 프로젝트 기간 중 🔗AWS SDK integration for Step Functions 기능이 출시되며 Step Functions 내부에서 연동이 가능한 AWS 서비스들이 갑자기 많아졌는데, 이것를 보고 AWS가 Serverless를 정말 강력하게 추진하고 있음을 느꼈습니다.

- 클라우드 아키텍처의 복잡성에도 불구하고 수동 배포를 한 탓에 종합적인 진행 상황이 잘 관리되지 않아 연동과 테스트가 쉽지 않았었습니다. 복잡성이 있는 Serverless 프로젝트에는 CDK를 반드시 도입할 계획으로 개인적으로 미리 기술을 익혀두고 있습니다.

🔗Aws

웹 기반 강화 학습 시뮬레이션 서비스화

December 31, 2020 · 2 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

Demo

Technology

- Docker API를 통해 Worker 컴퓨터를 제어해 강화 학습 Training 컨테이너를 구동
- 🔗three.js 와 🔗cannon.js 를 연동해 브라우저 내에 강화 학습 시뮬레이션 환경 구성

- 강화 학습 컨테이너와 시뮬레이션 환경을 socket.io 프로토콜을 통해서 환경 정보와 지시 사항을 주고받으며 강화 학습 진행

Abstract

> 웹 기술을 이용한 3D 로봇 팔 대상의 유연한 강화 학습 환경 구성

본 연구는 웹 기술을 이용해 사용자 친화적이고 유연한 강화 학습 시뮬레이션 환경 아키텍쳐를 구성하는 목적을 가지고 있으며, 실증을 위해 로봇 팔을 강화 학습의 대상으로 설정했다. 이를 위한 시스템은, 정역학적 계산과 시각화를 일관성 있는 코드로 구현할 수 있는 브라우저 상의 3D 시뮬레이션 모듈, 컨테이너 형태로 강화 학습 모델을 구동하는 강화 학습 서버 모듈, 강화 학습 서버 모듈 내의 컨테이너들을 제어하는 이벤트 기반 서버 모듈, 강화 학습 구동을 포함해서 앞선 모듈들과의 실시간 이벤트 기반 네트워킹 기능을 수행하고 유저와 상호 작용하는 이벤트 기반 클라이언트 모듈로 구성됐다.

본 시스템을 실증하기 위한 목적으로 로봇 팔을 이용한 Task 2가지를 구현했으며, 컨테이너 환경에서 구동되는 DDPG 알고리즘이 성공적으로 수렴하는 것을 확인했다. 첫 번째 Task는 로봇 팔이 임의의 Target을 추종하는 Reacher다. Open CV와 NumPy를 이용해 구현된 Baseline 환경과 비교했을 때, 본 시스템을 통해서 향상된 유연성, 빠른 수렴 시간을 얻을 수 있었다. 본 시스템의 강화 학습 진행 속도는 Unity, MuJoCo로 구현된 Reacher 환경에 상응하는 속도로 나타났다.

두 번째 Task는 회전하는 물체를 로봇 팔로 한 방향으로 돌리는 Task다. 이 Task는 현실 로봇에서 시도된 Task인데, 브라우저에서 구동되는 물리 엔진이 강화 학습 모델의 수렴에 필요한 일관된 데이터를 제공하는지 검증하는 실험적인 Task로서 구현했다. 여기서는 첫 번째 Task에서 사용했던 동일한 DDPG 알고리즘을 이용해서 로봇이 다양한 전략으로 Task를 수행하는 것을 확인했다.

본 시스템은 현재 서비스되고 있는 본 강화 학습 시스템은, 환경 구축에 있어서 기존의 강화 학습 환경보다 뛰어난 유연성을 확보하고, 기존의 강화 학습 시스템과 비교할만한 수준의 강화 학습 수행 속도를 보여준다. 구현 방식에 있어서 이벤트 기반 프로그래밍, 컨

테이너 기술, 첨단 Web API 등의 웹 기술 등을 이용했으므로 본 시스템은 클라우드를 통한 서비스가 용이하며, 기존의 강화 학습 환경들에 비해서 개선된 사용자 친화성 및 접근성을 고려했을 때 본 시스템은 강화 학습 교육 소프트웨어로서 활용될 수 있다.

Lessons

- 이 프로젝트에서 Docker API를 직접 호출해보면서 컨테이너에 대해서 이해한 내용은 개발 커리어 내내 좋은 배경 지식으로 사용되고 있습니다.

플랜트 육상 운송성 평가 시스템

December 31, 2019 · 1 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

Demo

Overview

- 플랜트 산업 분야에서 대형 화물을 육상으로 운송할 때, 도로의 폭과 장애물의 높이를 고려해서 운송 경로를 탐색한다는 시나리오에 기반해 웹사이트를 개발한 연구
- 레거시 PHP 애플리케이션을 PHP JSON API + jQuery AJAX 형태로 재개발하고 Single page application으로 전환, Google Maps SDK를 통한 interactive한 경로와 지도 기능의 비중을 크게 확대했습니다.
- 시작지, 경유지, 도착지를 CRUD하고 비용 분석 Report를 생성하는 핵심 기능이 있습니다.

Lessons

- 자바스크립트 3개월 차에 프로젝트 리드를 맡았지만 무사히 프로젝트를 완료할 수 있어 다행이었습니다.
- 로컬에서 개발을 전혀 하지 않고, AWS EC2에 개발 환경을 구축하고 처음부터 Nginx를 통한 HTTPS를 도입해 클라우드 네이티브를 추구했습니다.

- 실제로 구현은 못했지만, 위성 사진으로부터 도로 폭 추출, 장애물 높이 추출 등 동료들이 진행한 이미지 처리에 관련한 연구들을 어떻게 AWS Lambda 등을 이용해 어떻게 서비스할 수 있을지 고민했었습니다.

References

- 🔗Swept Path Analysis 활용 모듈화 플랜트 운송성 평가 알고리즘 및 자동화 시스템에 대한 연구
- 🔗웹 기반 육상 모듈화 플랜트 운송 프로젝트 통합 관리 및 분석 시스템 개발
- 🔗Pix2pix를 이용한 클라우드 기반 최소 도로 폭 추출 시스템 개발

🔗Home

Tech 🔗

🔗업무 와 🔗사이드 프로젝트 에서 공통적으로 즐겨 사용했던 기술들을 분야 별로 기록했습니다.

Experiences with Networking

1 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

AWS VPC

- HTTP API Gateway와 Elastic Load Balancer를 통해서 [Cloud Map Service]에 등록된 VPC 내부 리소스를 안전하게 공개합니다.
- Public IP가 없는 VPC 내부의 리소스에 대해서 🔗Gateway endpoints 를 통해서 AWS 내부의 서비스에 접근하게 할 수 있습니다.
- 적절한 기준을 가지고 ACL, Security group을 관리할 수 있습니다.
- VPC peering을 통해서 계정/리전이 다른 다수의 VPC를 운영할 수 있습니다.

Edge Networking

- 🔗ECS Anywhere 서비스로 On-premise 인프라를 ECS cluster에 등록해 서비스할 수 있습니다.
- Frontend 개발자와 협력해 🔗CloudFront routing rule 을 작성하고 배포와 테스트를 지원합니다.

Monitoring Experiences

2 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

CloudWatch

Log & Metrics Collection

- 🔗CloudWatch agent container 로 Systemd log와 Custom log를 CloudWatch로 수집합니다.
- Request id, Execution id 등 AWS에서 부여한 UUID들을 기준으로 DB를 설계하고 로그에 포함시켜 Tracing을 용이하게 합니다.
- 🔗Subscription filter 기능으로 서비스에서 발생한 중요 로그를 전파합니다.
- EC2, RDS 등의 AWS 서비스에서 기본으로 제공하는 🔗CloudWatch metrics 외에도 Custom metrics를 정의해 관심 대상으로 설정합니다.

Monitoring

- 🔗SNS topic에 email subscription 을 추가해 서비스에서 직접 Topic에 메시지를 발송합니다.
- Metric에 Anomaly condition을 정의하고 🔗CloudWatch alarm 을 설정합니다.
- 🔗AWS Chatbot Slack App 을 도입해 사내 메신저로 알림을 수신했습니다.
  - CloudWatch alarm state 변경 시
  - 🔗EC2 Instance state change event

    > EC2 state change event -> EventBridge -> SNS -> AWS ChatBot

- HTTP를 사용하는 서비스의 Status code 모니터링
  - 🔗CloudFront metrics 의 Error rates
  - 🔗Lambda function URL metrics 의 4xx, 5xx count
- 🔗Naimy 프로젝트 에서는 배포와 운영 이슈 해결을 맡았었습니다. 🔗크롤링 작업에 대한 모니터링 을 구축한 경험을 참조하실 수 있습니다.

Action Items

- 이상 상태가 나타난 시점을 특정하고 가능하다면 이상 상태를 추적할 수 있는 id를 획득합니다.
- CloudWatch log stream을 특정하고 S3, DB, Redis 등의 Storage를 조사해 원인을 파악하고 적절한 조치를 수행합니다.
- 필요한 경우 🔗Lambda versions , Container image ☒digest 등을 이용해 과거 버전으로 서비스를 롤백합니다.

On-Premise

Log & Metrics Collection

- 컨테이너들의 로그를 별도의 설정 없이 통일된 형식으로 수집할 수 있도록 🔗 `docker compose log` 를 기본으로 선정했습니다.
- 들어온 데이터에 따라서 처리 결과를 개발자와 비개발자가 직접 검증해야 하는 경우에는 Tracing이 용이하도록 입력을 기준으로 UUID를 부여했습니다.
- 🔗Sentry SDK 를 주요 서비스에 적용해 오류 원인 파악에 필요한 배경 정보를 수집했습니다.

Monitoring

- 클라우드에서와는 달리, 인프라가 이상 상태를 보일 수 있다는 가정 하에 컨테이너 별 Health check를 보다 꼼꼼하게 작성했습니다.

    - Redis 등 핵심 서비스에 대한 Reachability를 확인합니다.
    - 작업을 처리하지 않고 대기 상태에 있는 경우 카메라, GPU 등 특수한 하드웨어가 정상 작동하는지 확인합니다.

- 정형화된 형태의 서비스라고 할 수 있는 서버 형태의 번호판 인식 시스템의 모니터링에는 🔗Kibana , 🔗Influx DB 를 적용해 각각 로그와 지표를 관리할 수 있도록 했습니다.
- 특수한 환경에서의 모니터링을 위해서 🔗SSH 터널링 기반의 자체 인프라와 툴 을 개발해 2년 이상 꾸준히 유지보수하며 사용했습니다.

Action Items

- 🔗autoheal 이라는 컨테이너는 Docker socket을 모니터링해 Unhealthy한 컨테이너를 자동으로 재시작합니다.
- 🔗 `docker inspect` 를 통해 최근 Health check log들을 조회해 원인을 파악합니다.
    - 🔗 `HostConfig.State.Health[*].Log.Output` 참조

TODOs

> 최근 관심을 두고 있는 기술들입니다.

- 🔗OpenTelemetry
- 🔗AWS X-Ray

🔗Monitoring

Operations: Dev, Building, Testing, CI/CD, and more...

1 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

> 개발 팀이 편안한 환경에서 장기적인 성과를 낼 수 있도록 노력해왔습니다.

Development

- 가장 코드 양이 많은 JavaScript, Python을 중심으로 Convention 확립에 노력했습니다.
- JavaScript, TypeScript Node.js 프로젝트에 전역적으로 적용할 수 있는 🔗ESLint Sharable Config 를 🔗https://github.com/sehyun-hwang/eslint-config-nextlab 에서 오픈소스로 관리하고 있습니다.
- Python에서는 Type checking과 Linting을 동시에 구동하기 위해서 mypy를 Child process로 실행하는 pylint plugin을 제작한 적이 있으며, 최근에는 ruff linter와 black formatter를 사용하고 있습니다.
- 다양한 언어로 이루어진 Monorepo에 대한 Convention을 관리하기 위해서 🔗MegaLinter 와 그것의 🔗Github Action 을 도입했습니다.

Building

- 이미지 처리 등의 특수한 기능이 있는 컨테이너들에 대해서는 불필요한 컴파일러와 인터프리터를 제거하는 🔗Multi-stage build 를 수행했습니다.

- alpine 기반의 컨테이너로 컨테이너의 이미지 크기를 줄여 빌드 시간 단축, 배포 속도 개선, 취약점 개선 등의 효과를 얻었습니다.
- 복수의 Dockerfile을 이용한 Container build, 빌드 캐시 관리 등 High-level build 기능을 구현해주는 🔗 `buildx bake` 를 도입했습니다.
- Makefile을 이용해 직접 별도로 컴파일해야 하는 C++ 프로젝트나 표준적이지 않은 프로젝트의 개발-빌드-테스트-배포 과정을 관리합니다.
- 빌드를 위한 인프라를 관리할 필요가 없고 동시에 작업을 병렬적으로 진행할 수 있는 Serverless CI를 추구합니다.
  - AWS CodeBuild
  - GitHub Action

Testing

- Test runner가 테스트 케이스들을 병렬적으로 수행하는 데 문제가 없도록 하는 정석적인 객체 지향 코드를 작성합니다. 서비스 코드에 Side effect가 없고 부분 별로 역할 구분이 명확해야 테스트 코드에서 class를 상속받아서 테스트 케이스를 작성할 수 있었습니다.
- 🔗vitest , 🔗pytest 로 테스트 케이스를 작성할 수 있습니다.

🔗Dev-Ops

Security Routines

2 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

> 어떻게 보면 당연한 관습들을 풀어서 글로 남겼습니다.

Routines

- **🔗IAM**
  - Production 환경에서는 🔗Lambda execution role 과 🔗EC2 IAM role 등의 🔗IAM temporary credential 기능을 활용합니다.
  - 리소스에게 부여된 권한들은 해당 리소스의 동작에 대한 명세이기도 하다고 생각하고 있으며, Service role에 Least privilege rule을 최대한 준수했습니다.
  - 프로젝트 별로 User group을 만들고 개발자들에게 개발에 필요한 권한을 부여합니다.
  - 🔗GitHub CLI SSO , 🔗AWS CLI SSO 등의 SSO 서비스를 사용해 개발자가 개발에 필요한 권한을 획득할 수 있도록 도입 중입니다.
- **Secrets Management**
  - **영구적인 Secret보다는 직접 관리할 필요가 없는 Credential을 우선적으로 사용해야한다는 원칙을 지키고 있습니다.**
  - Secret이 불가피하게 필요하다면 🔗Docker secret , 🔗AWS Secrets manager , env file 등을 적절히 조합해 Stage를 안전하게 구분해 Secret을 관리합니다.
  - 필요할 경우 🔗gitleaks 등을 도입합니다.

    - 프로젝트 별로 SSH key를 다르게 사용하고 🔗EC2 Instance Connect 를 통해 인프라에 안전하게 접근합니다. Password 사용을 자제합니다.

- **Infra Security**
    - 🔗AWS Certificate Manager 의 관리형 인증서를 CloudFront, API Gateway, Load Balancer 등에 연동해 HTTPS 연결을 제공합니다.
    - 인증서를 직접 관리할 때는 🔗Certbot Route 53 plugin Docker image 또는 🔗acme.sh script 로 인증서를 발급받아 Nginx에 등록합니다.

- **Application Authentication**
    - API Gateway endpoint에 🔗IAM authorizer , 🔗Cognito user pool authorizer 을 부착해 인증 기능을 추가합니다.
    - 3rd party identity provider가 추가된 Cognito user pool을 운영할 수 있고, 🔗amazon-cognito-identity-js 라이브러리를 복수의 프로젝트에서 사용했습니다.
    - Nginx가 요청을 받았을 때, 지정된 Endpoint로 Subrequest를 보내 상태 코드에 따라서 요청을 Reject하는 🔗ngx_http_auth_request_module 의 기능으로 본 서비스와 인증 서비스를 분리하는 아키텍처를 사용할 수 있습니다.

Experiences

- VPC 외부에 있는 Lambda에서 VPC 내부의 ElasticCache Redis에 접근이 필요한 경우가 있었습니다. VPC 내부에서는 CloudMap에 등록된 주소를 통해 무인증으로 접근하도록 기존 로직을 유지했고, VPC 외부에서는 Nginx를 두어 Nginx가 트래픽을 중계할 때 mTLS authentication을 요하도록 구성했습니다.
    - 관련 Directive: 🔗 `ssl_verify_client`

- 🔗RDS에 IAM authentication 을 구성해 사용할 수 있습니다.

TODOs

> 회사에서 꼭 필요하다고 생각해 익히고자 하는 기술입니다.

- CDK를 이용한 IAM 역할 관리, 테스트 코드 작성
- AWS Secret Manager secret rotation

🔗Dev-Ops 🔗Security

Skills & Experiences with AWS

1 min · Sehyun Hwang | Translations: 🔗문화어

▶ Table of Contents

> *AWS에서 어떤 서비스를 써보셨나요?*라는 질문에 답하기 힘들어서 적은 페이지

Skills

Infra

- Python, Node.js 등의 언어로 작성된 코드가 구동되는 EC2 instance의 type과 FarGate, Lambda 등 Serverless 서비스의 spec을 선정할 수 있습니다.
- 비용 절감
  - Lambda와 Container에 **ARM architecture**를 우선적으로 검토합니다.
  - 🔗EC2 Spot instance 와 🔗Fargate Spot capacity provider 등 짧은 생명 주기를 가진 인프라를 이용합니다.
- 🔗EBS, EFS, S3 등 적절한 종류의 Storage를 선택하고 EC2 등의 인프라에서 사용할 수 있습니다.
- 🔗AWS Backup plan & lifecycle 을 지정해 🔗EBS , 🔗EFS , 🔗RDS 등에 대해서 주기적으로 Backup을 수행하고 삭제할 수 있습니다.

Monitoring

> 🔗Monitoring experience 페이지의 CloudWatch 부분 을 참조해주세요.

AWS 환경과 On-premise 환경에서 데이터 수집, 모니터링, 대처 방안을 정리했습니다. AWS에서는 큰 인프라를 직접 관리할 필요 없이 개발자가 문제 상황을 빠르게 파악하고 쉽게 트러블슈팅을 할 수 있는 환경을 구성하는 데 주안점을 두었습니다.

Security

> 🔗Security Routines 페이지 를 참조해주세요.

인터넷에 서비스를 책임감 있게 공개하기 위해서 노력했던 경험을 정리했습니다.

Networking

> 🔗Experiences with Networking 페이지를 참조해주세요.

VPC를 구성하고 내부 리소스들을 인터넷에 노출하는 방식에 관해서 정리했습니다. CloudFront, Route 53 등의 글로벌 서비스를 구성하는 AWS 서비스들도 기초적인 수준에서 사용 경험이 있습니다.

Projects

- 🔗Serverless 기반 신문 PDF 정형화
- 🔗LLM 기반 작명 서비스