김 기 홍

010-9486-2225
berserkppp1@gmail.com
github.com/utopiandreams
https://techforme.tistory.com

ABOUT ME.

과감하게 실행하고 극도로 몰입하여 폭발적으로 성장하는 개발자 김기홍 입니다.

'23년 7월 **사업기획 직무에서 개발자로의 전향**을 결심한 이래로 **3개의 팀 프로젝트를 쉴 새 없이 진행**하면서, 파죽지세의 성장을 이어나가고 있습니다. 퇴사 직후 개발 지식이 전무한 상태로 Hola라는 플랫폼에서 **직접 프로젝트 팀원을 모집하여, 프로젝트 리더로서 프로젝트 기획과 백엔드 개발을 주도**하였습니다. 올해 1월 현 직장에 입사한 이후에도 지속적인 프로젝트 수행으로 **인프라 자동화, 배포 관리, 컨테이너 오케스트레이션** 등을 공부하며 기술적 영역을 넓히고 있으며, 동료 개발자들과 **사내 세미나 주최, 코드 리뷰** 등을 진행하며 지식의 깊이도 더하고 있습니다.

PROJECT EXPERIENCE.

■ 꼼삐 | '24. 5 ~ 현재 | 팀원 6명 (FE 2, BE 2, 기획 1, 디자인 1)

- 반려식물을 키우기 위한 서비스 플랫폼
- 내 식물 등록 및 관리, 캐릭터 키우기, 식물 도감 등
- Service : https://comppi.site
- Github : github.com/com-pi

■ 배달서비공제조합 플랫폼 구축 | '24. 1 ~ '24. 4 | 티맥스핀테크 연구본부

- 라이더의 유상운송보험 가입/사고접수/보상/기간계 등 운영 플랫폼 구축 사업
- 기간계 공통 모듈 (기능별 권한 관리, 이미지/파일 모듈, 공지, 배너 등) 개발 담당

■ 내 잔고를 부탁해 | '23. 7 ~ '23. 11 | 팀원 3명 (FE 1, BE 2)

- 사용자 위치 기반의 중고거래 및 공동구매 플랫폼
- 물품 검색, 채팅, 매칭 등 기능 제공
- Github : github.com/Team-Naejango

TECH SUMMARY.

1) Development

□ **지리데이터 ORM 기술인 Hibernate Spatial 을 이용, 지리데이터를 객체 수준으로 핸들링 하였습니다.** 별도의 커스텀 쿼리없이 PostGis 내장함수로 지리데이터에 대한 쿼리를 수행하였습니다. R-tree 기반 인덱싱 으로 지리데이터 조회 쿼리 퍼포먼스를 향상 하였습니다. 다양한 검색 조건에 따른 동적쿼리 생성을 위해 Querydsl 을 도입하였습니다.

□ **Redis 를 데이터 캐싱 처리 및 pub/sub 구조의 메세지 브로커로 활용 하였습니다.** 빈번한 읽기가 예상되는 데이터에 대해 Redis를 이용하여 캐싱 처리하였습니다. 리소스가 변경되는 경우 엔티티 리스너를 통해 캐시를 업데이트하는 방식으로 항상 최신의 데이터를 유지하도록 하였습니다. 또한 채팅 기능 구현시 서버간 통신에서 메세지 브로커로 활용하였습니다.

□ **AOP 기술을 적극 활용 하였습니다.** 인증처리가 필요한 컨트롤러에 대해 커스텀 어노테이션을 정의하여 인증객체를 생성하는 로직을 구현하였습니다. ORM 의 취약점인 N+1 Problem 을 원천적으로 해결하고자, Spring AOP 와 AspectJ 를 이용하여 api 요청시 N+1 발생을 감지하는 기능을 구현하였습니다.

2) Infrastructure

□ **Type1 하이퍼바이저(Proxmox) 가상환경을 구축하고 Vagrant 를 통해 가상머신을 프로비저닝 하여 서비스 인프라를 자동화 관리 하였습니다.** Hypervisor를 활용해 가상 머신을 호스트 OS 없이 하드웨어에서 직접 실행하여 오버헤드를 줄이고, 하드웨어 리소스를 최대한 활용할 수 있도록 최적화했습니다. 또한, 워커 노드를 쉽게 추가할 수 있도록 Vagrant 스크립트를 작성하여 Kubernetes 클러스터를 간편하게 확장할 수 있게 구성했습니다.

□ **Jenkins 와 Argo CD를 이용하여 Kubernetes 메니페스토 및 어플리케이션 컨테이너에 대한 CICD 파이프라인을 구축하였습니다.** Jenkin 파이프라인을 통해 배포의 전 과정을 자동화 하였습니다. Argo CD 를 도입하여 쿠버네티스 리소스 통합/배포 주기를 최대한 줄여 개발 속도를 극대화하고, Image Updater 를 통해 컨테이너 이미지 변경시에도 자동적으로 CICD 가 이루어지도록 하였습니다.

□ **kubernetes Gateway Api 를 도입하여 TLS 복호화, namespace 간 라우팅 등 트래픽 인입을 통합 관리하였습니다.** 인그레스, 서비스 메쉬를 통합한 차세대 쿠버네티스 리소스인 Gateway API(Nginx Gateway Fabric) 를 도입하여 dev 환경과 prod 환경의 리소스를 namespace 로 격리하고, namespace 간 라우팅을 수행하였습니다.

TECHNICAL CHALLENGES & PROBLEM SOLVING.

□ 위치 기반 서비스 구현

- **지리데이터 타입을 ORM 으로 핸들링** [techforme.tistory.com/39]

  〈내잔고를 부탁해〉 프로젝트에서 Storage 엔티티에 위경도 좌표 정보를 저장하도록 하여, 위치 기반 서비스를 구현 하였습니다. 위경도 데이터는 위도, 경도 데이터를 각각 Double 타입 컬럼에 저장하여 활용할 수 있으나, 지리데이터 관련 ORM 기술인 Hibernate Spatial 을 이용하여 지리데이터 타입의 컬럼을 객체(jts 라이브러리)에 직접 맵핑 하였습니다.

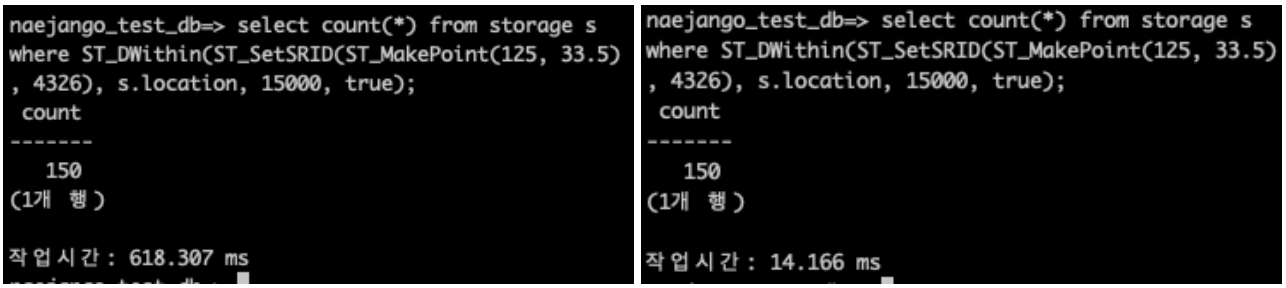
- **Postgresql extension 인 PostGis 내장 함수 사용**

  지리데이터를 특정 좌표와 반경을 기준으로 조회하거나, 거리를 계산하는 쿼리가 필요했습니다. 이는 Haversine 공식(지표면 상의 두 좌표 간 거리)을 이용한 커스텀 함수를 만들어 해결할 수 있었습니다. 하지만 DB의 자체적인 내장함수를 이용한다면 쿼리의 퍼포먼스와 개발 생산성을 높일 수 있겠다는 판단을 하였습니다.

  기존에 MySQL 을 사용하였으나, Postgresql 이 Hibernate Spatial 에서 더 많은 함수를 지원하고 있었고 Postgresql extension 인 PostGIS 의 성능이 뛰어나다는 것을 알게되었고, Postgresql 로 DB 를 교체하여 별도 커스텀 함수 없이 쿼리를 작성할 수 있었습니다. [github.com/Team-Naejango/back-end/issues/7]

- **지리데이터 쿼리 최적화를 위한 지리데이터 인덱스 생성** [techforme.tistory.com/49]

  프로젝트에서 가장 많이 쓰이는 지리 데이터 쿼리는 위경도 좌표간의 거리를 비교하는 연산을 자체적으로 수행해야 하기 때문에 데이터 증가시 DB 에 많은 부담이 될 것이라고 판단하였습니다. Stroage 의 좌표 데이터에 R-tree 를 기반으로 하는 인덱스를 생성하여 퍼포먼스를 향상 하였습니다. 20만 건의 데이터 기준 약 600 ms → 15 ms 로 약 40 배의 성능이 개선될 수 있었습니다.



- **다양한 검색조건에 따른 동적쿼리 생성을 위해 Querydsl 도입** [techforme.tistory.com/39]

  물품 검색 기능 구현 시 아이템의 활성화 여부, 판매/구매 타입, 카테고리, 키워드, 반경 등 여러가지 검색조건에 따라 검색 결과를 보여주고 싶었습니다. 이런 경우, 검색 쿼리는 해당 조건을 동적으로 생성해 낼 수 있도록 많은 조건문을 통해 구현하게 되는데, 구현에서의 사소한 실수가 런타임 시에 오류로 이어질 수 있다는 우려가 있었습니다.

  이에 대응하여 Querydsl 을 도입하여, 더 쉽게 동적 쿼리를 생성할 수 있었으며 개발자의 실수 또한 컴파일 타임에서 체크할 수 있어 개발 생산성과 정확도를 높일 수 있었습니다.

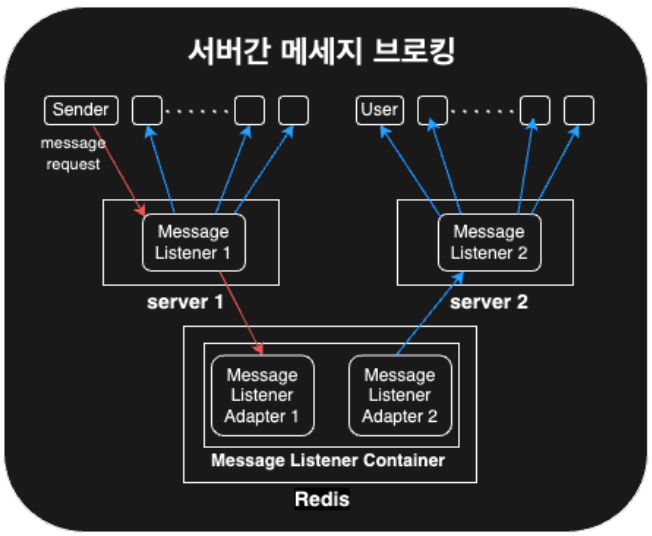
□ Redis 를 활용한 캐싱 처리 및 Message Broking

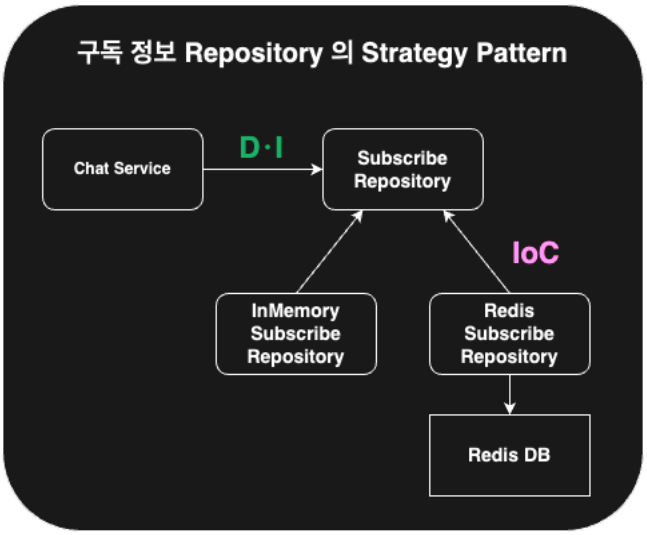
- **Redis 를 활용한 API 응답 속도 향상** [https://techforme.tistory.com/71]

  인메모리 데이터베이스인 레디스의 특성을 이용하여, 빈번하게 요청되거나 IO 에 오랜시간이 걸리는 데이터를 캐싱 처리하였습니다. 이를 통해 데이터베이스 IO 를 줄여 응답 속도를 개선하고, 데이터베이스의 부하를 줄였습니다. 캐시 데이터가 정합성을 유지하도록 엔티티 리스너를 통해 데이터가 변경된 경우, 레디스의 캐시를 갱신하도록 하였습니다.

- **Redis MessageBroker 를 통한 서버간 Message Broking** [techforme.tistory.com/16]

  Spring Websocket 에서 지원하는 내장 Message Broker 는 각 채널과 사용자의 구독 정보를 서버 내부에서 관리하는데, 이 경우 단일 서버에서는 유효하게 동작할 수 있으나, 서버 스케일 아웃시 서버 간 Message Broking 을 할 수 없게 됩니다. 이에 Pub/Sub 형태의 Redis Message Broker를 추가적으로 구성하여 서버간의 Message Broking 을 수행하도록 하였습니다.





□ 온프레미스 환경에서 Kubernetes 클러스터 및 CICD 파이프라인 구축 [꼼삐]

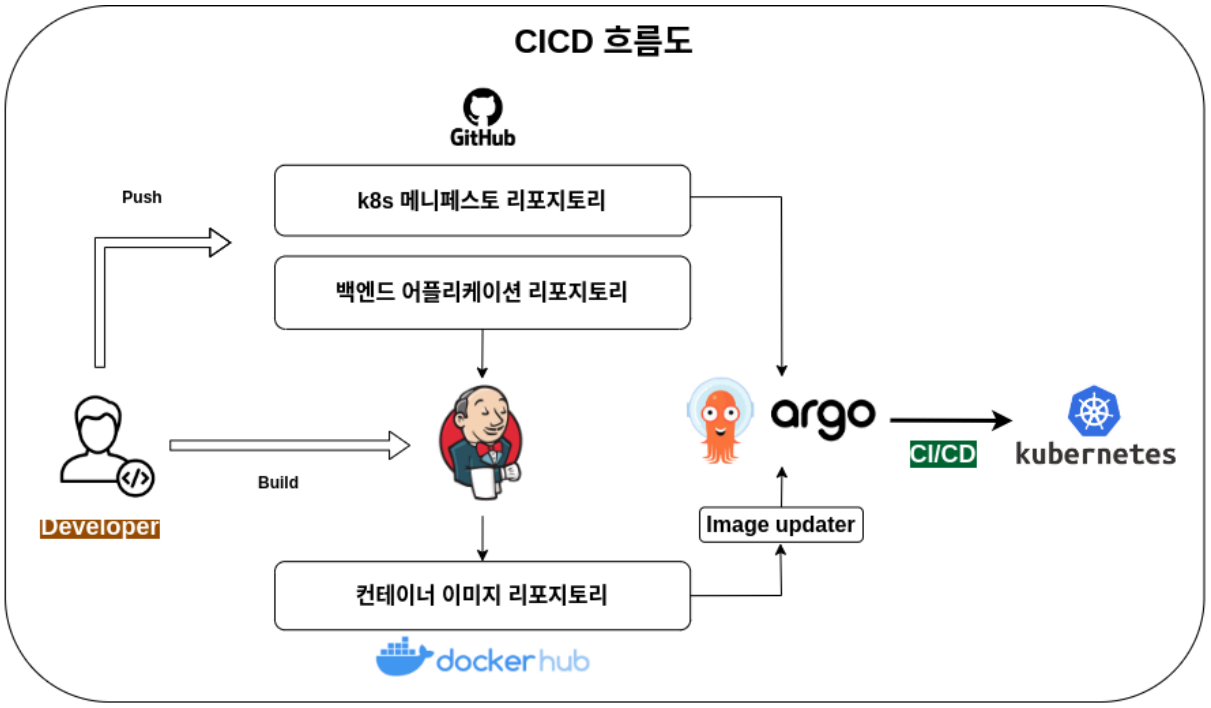
- **Hypervisor 를 통한 VM 운용 및 k8s 클러스터 구축** [https://techforme.tistory.com/66]

  프로젝트 배포를 위해 홈 서버에 VM을 생성하고 Kubernetes 클러스터를 구축했습니다. 처음에는 Ubuntu 환경에서 VirtualBox 로 VM 을 운용하였지만, 이후 Type1 하이퍼바이저인 Proxmox 로 전환하여 Host OS 를 거치는 오버헤드를 없애 하드웨어 리소스를 최대한 활용했습니다. 인프라 확장시 Vagrant 를 이용하여 인프라 구성의 자동화를 시도하였습니다.

  운영 서버에서는 VM 간 통신을 위해 NAT 기반 네트워크를 사용했으나, 운영 서버와 개발 PC를 하나의 Kubernetes 클러스터로 통합하기 위해 공유기를 게이트웨이로 설정한 브리지 네트워크로 변경했습니다. 또한, 운영 서버와 개발 서버를 물리적으로 구분하고 관리하기 위해 네임스페이스로 격리하여 별도로 운영했습니다

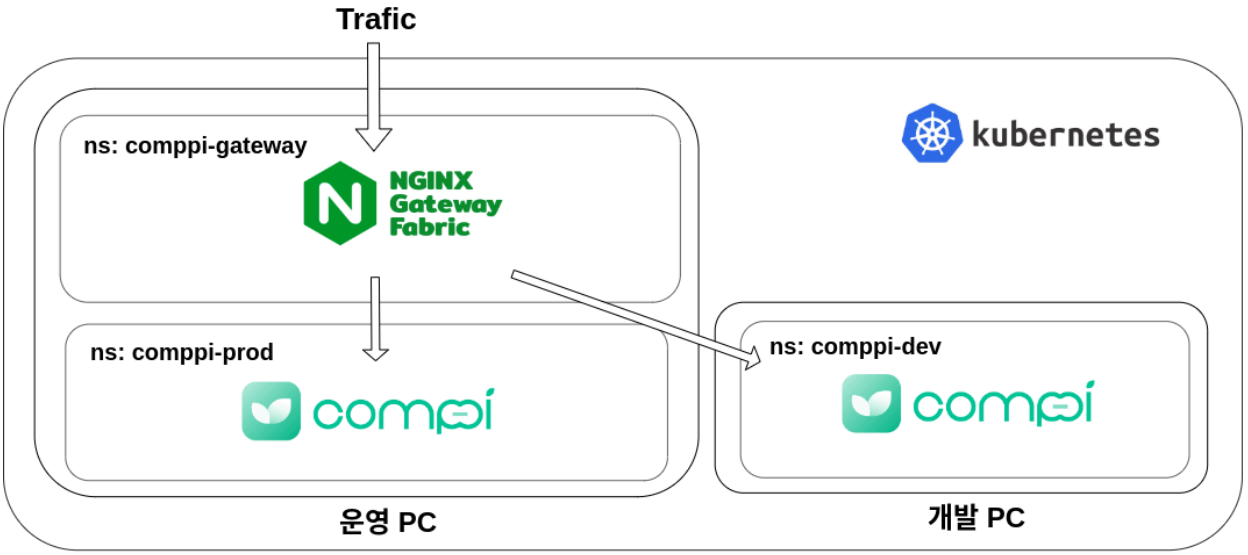
- **Jenkins 와 ArgoCD 를 이용한 CICD 파이프라인 구축** [https://techforme.tistory.com/68]

  ArgoCD 를 이용하여 Git 으로 관리되는 k8s 매니페스토와 배포된 k8s 리소스를 지속적으로 동기화하도록 하였고, 임시로 메니페스토를 수정적용하여 테스트를 해볼 수 있도록 하였습니다. 애플리케이션 컨테이너 이미지의 경우, 컨테이너 이미지 리포지토리를 바라보는 Argo CD Image updater 를 도입하여 컨테이너 이미지의 태그 변경사항을 추적, 실시간으로 k8s 리소스에 적용하여 애플리케이션 배포 또한 자동화 하였습니다.

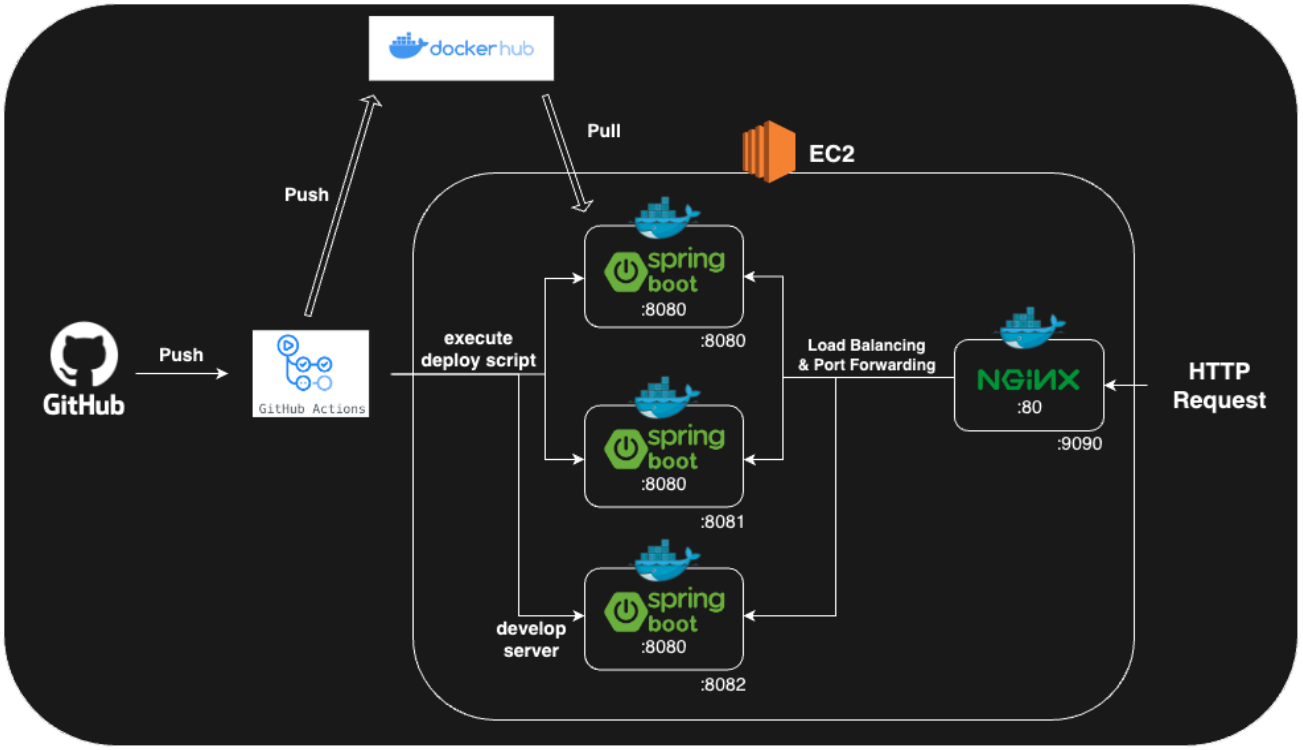


- **Gateway API 를 이용하여 트래픽 인입 관리** [https://techforme.tistory.com/67]

  K8s 클러스터에서 Ingress Controller 와 Service Mesh 를 통합한 차세대 리소스인 Gateway API 를 Nginx Gateway Fabric 을 통해 적용하였습니다. 기존에는 Nginx 를 파드 형태로 배포하였는데, 이를 Gateway API 로 변경하면서, Gateway, HTTPRoute 리소스를 이용해 네임스페이스 간 라우팅 및 TLS 를 수행하였습니다.



□ 무중단 배포 & 이중화 서버 아키텍쳐 구현 [내잔고를 부탁해]



- **서버 이중화**

  각각의 기능 구현 시 단일 서버가 아닌 다중 서버로의 스케일 아웃 상황을 고려하였습니다. 로그인 인증에서는 세션이 아닌 Jwt 토큰을 사용하였고 채팅 서비스의 경우 구독 정보를 외부 DB(Redis)에서 관리하였으며, 서버 간의 Message Broking 을 위해 Redis Message Broker 를 도입하는 등 여러 설계적 고민을 하였습니다.

  처음 서버를 이중화 하였을 때 OAuth Authorization 관련 에러가 있었습니다. 인증 요청 정보를 일시적으로 서버 세션에 저장하여, 인증 응답이 Round Robin 알고리즘에 따라 다른 서버로 반환되는 경우 해당 요청 정보를 찾지 못해 Authorization 이 실패하는 에러였습니다. 요청 정보를 서버 내부가 아닌 외부 DB(Redis) 에 저장하는 방식으로 해결하였습니다. [techforme.tistory.com/50]

- **Nginx, Docker 를 이용한 무중단 배포(rolling, blue-green)** [techforme.tistory.com/45]

  새로운 버전을 서비스의 중단 없이 연속적으로 배포하는 무중단 배포를 구현하였습니다.

  단일 서버 구동시에는 비어있는 포트에 신규 버전 배포 후 라우팅 설정을 변경하는 blue-green 배포를 적용하고, 2개 서버 구동시에는 기존 서버에 신규 버전을 차례로 배포 하는 rolling 방식의 무중단 배포를 채택 하였습니다. 또한 개발 중인 서버를 배포 환경에서 테스트 해볼 수 있도록, 개발용 브랜치에 푸시하는 경우 기존의 서비스 서버가 아닌 별도의 개발 서버에서 배포하도록 하였습니다. 해당 서버는 요청 호스트에 따라 라우팅 되도록 설정하여 일반적인 접근을 차단하였습니다.

  배포는 EC2 내부에 쉘 스크립트를 동작시키는 방식으로 진행되며, branch push 시 배포 branch와 배포 방식(blue-green, rolling, develop 방식)을 설정할 수 있도록 하였습니다.

□ AOP 기술(Spring AOP, AspectJ) 을 활용한 부가기능 구현

- **인증을 위해 어노테이션 기반 Passport 생성 로직 구현**
  Transaction 동작에 대해 테스트 코드 작성시 Transaction 중간에 예외를 던지는 Test stub 을 별도로 만들어야 했습니다. 이는 프록시 패턴이나 상속을 이용하여 특정 메서드에서 예를 던지도록 하는 식으로 Test stub 을 구현할 수 있으나, 여러 Transaction 테스트를 작성 해야함에 따라 코드의 중복이 생기게 되었습니다.

- **Transaction test stub 자동 생성 기능 구현** [techforme.tistory.com/37]
  Transaction 동작에 대해 테스트 코드 작성시 Transaction 중간에 예외를 던지는 Test stub 을 별도로 만들어야 했습니다. 이는 프록시 패턴이나 상속을 이용하여 특정 메서드에서 예를 던지도록 하는 식으로 Test stub 을 구현할 수 있으나, 여러 Transaction 테스트를 작성 해야함에 따라 코드의 중복이 생기게 되었습니다.

  이를 피하기 위하여 Spring AOP 와 AspectJ 를 이용, 커스텀한 어노테이션을 붙임으로서 테스트 시 자동으로 테스트 스텁 생성하는 기능을 만들어 생산성을 증대하였습니다.

- **N+1 발생시 이를 감지하는 N+1 Detector 기능 구현** [techforme.tistory.com/38]
  프로젝트에 ORM(Hibernate)을 적용하여 생산성을 높일 수 있었으나, 이는 동시에 Lazy Loading 으로 인한 N+1 문제가 발생 할 수 있다는 취약점으로 작용 하였습니다. Entity graph, Fetch join, Projection 등을 이용하여 최대한 예방하고자 하였으나, 수많은 로직을 구현함에 있어 예기치 못한 N+1 문제가 있을 가능성이 있었습니다.

  이에 AOP 기술을 적용하여 Lazy Loading 이 일어날 수 있는 조건을 정의하고, 쿼리를 execute 하는 spring bean 객체를 프록시하는 방식으로 N+1 Detector 를 구현할 수 있었습니다. 이를 통해 인지하지 못한 N+1 문제(생성자 내에 참조객체 프로젝션 시 발생) 를 발견해 내어 개선할 수 있었습니다.

EDUCATION.

한양대학교(서울) 신소재공학부 (‘11.3 ~ ‘16.8)

WORK EXPERIENCE.

티맥스핀테크 연구본부 연구원 - 연구/개발 (‘24.1 ~ 현재, 총 8개월)

다원시스 신사업실 매니저 - 신규사업개발, (‘20.12 ~ ‘23.6, 총 2년 6개월)

대한민국 공군 중위 - 항공무기정비 (’16.08 ~’19.12, 총 3년 4개월)