이다은 (Lee Daeun)

- 현재에 최선을 다하는, 목표 지향적인 사람입니다.
- 읽기 좋은 코드, 깔끔한 코드를 작성하기 위해 노력합니다.
- 혼자 가면 빨리 가고, 함께 가면 멀리 간다. 라는 문장을 좋아합니다.
- 자동화 테스트가 무엇보다 중요하다고 생각합니다.
  현재 참여 중인 팀 프로젝트에서는 총 553개의 테스트 코드를 작성했습니다.

Contacts

Email. leede0418@gmail.com

Phone. 010-5633-2902

Channels

Github. github.com/da-nyee

Blog. da-nyee.github.io

Archive. woowacourse-projects

Project

깃-들다

2021-07 ~ 2021-10

Github / Website

Github Open API를 기반으로 구현한 개발자를 위한 SNS입니다.

참여 인력. 프론트엔드 2 / 백엔드 5

기술. **Java 11**, **Spring Boot 2.5.2**, **Spring Data JPA 2.5.2**, JUnit 5,
MariaDB 10.4.21, H2, Flyway 6.4.2,
AWS(EC2, S3), Docker, Nginx, Jenkins, k6,
Git, Notion, Slack

담당 내용

- 이슈 및 PR
- 외부 API 연동 및 CRUD API 구현
- 테스트 코드 작성 (팀 전체 총 553개)
- 조회 성능 개선을 위한 DB Replication 구성 (평균 TPS 11.8 → 40.1로 증가)
- Git Flow를 바탕으로 Git Branch 전략 수립
- Git Milestones와 Projects를 활용한 일정 관리
- 팀원 간 지식 격차를 줄이기 위한 문서화

Article & Presentation

2021-04 ~ 2021-11

Github / Website

테코블

우아한테크코스에서 코드 리뷰와 프로젝트를 통해 학습한 내용을 정리했습니다.

작성 글

- 웹 MVC 각 컴포넌트 역할
- DTO vs VO vs Entity
- 단위 테스트 vs 통합 테스트 vs 인수 테스트
- DispatcherServlet - Part 1
- DispatcherServlet - Part 2
- git submodule로 중요한 정보 관리하기
- JPA CascadeType.REMOVE vs orphanRemoval = true
- 커버링 인덱스

2021-10-07

Youtube

HTTPS

- HTTP, HTTPS, SSL, SSL 통신 과정에 대해 발표했습니다.
- 우아한 Tech 유튜브 채널에 업로드 됐습니다.
- 3.5K 조회수를 기록했습니다. (2021-11-08 기준)

Award

Girls in ICT 2019 Hackathon

2019-05

Best Award

- Ericsson-LG에서 주최한 해커톤입니다.
- ‘5G와 IoT를 통한 산업의 디지털화’를 주제로 IoT 허수아비를 기획, 제작했습니다.
- 상위 3팀에 선정되어 우승했습니다.

Experience

Ericsson-LG

2019-07 ~ 2019-08

R&D Intern

- Traffic Simulator 팀에 소속되어 인턴으로 근무했습니다.
- 네트워크에 대해 공부하며 회사 생활을 경험했습니다.
- 애자일 개발 방법론을 처음 접했습니다.

Education

우아한테크코스	2021-02 ~ 2021-11	웹 백엔드 3기 수료예정
가천대학교	2016-03 ~ 2022-02	컴퓨터공학과 졸업예정 (GPA 4.01/4.5)
일산대진고등학교	2013-03 ~ 2016-02	졸업

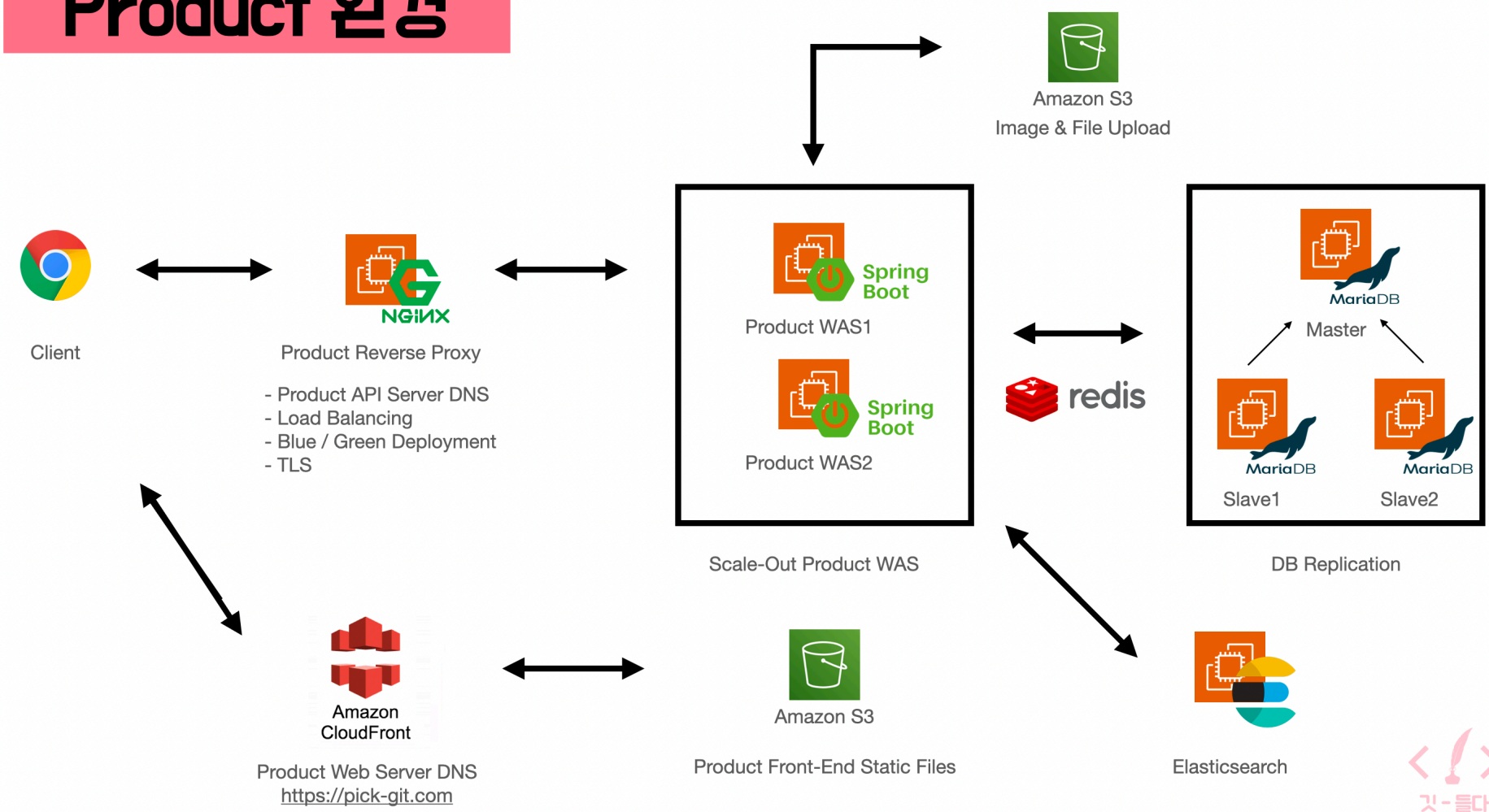
Certificates

정보처리기사	2020-12	취득
OPIc	2020-09	AL

깃-들다 (PickGit)

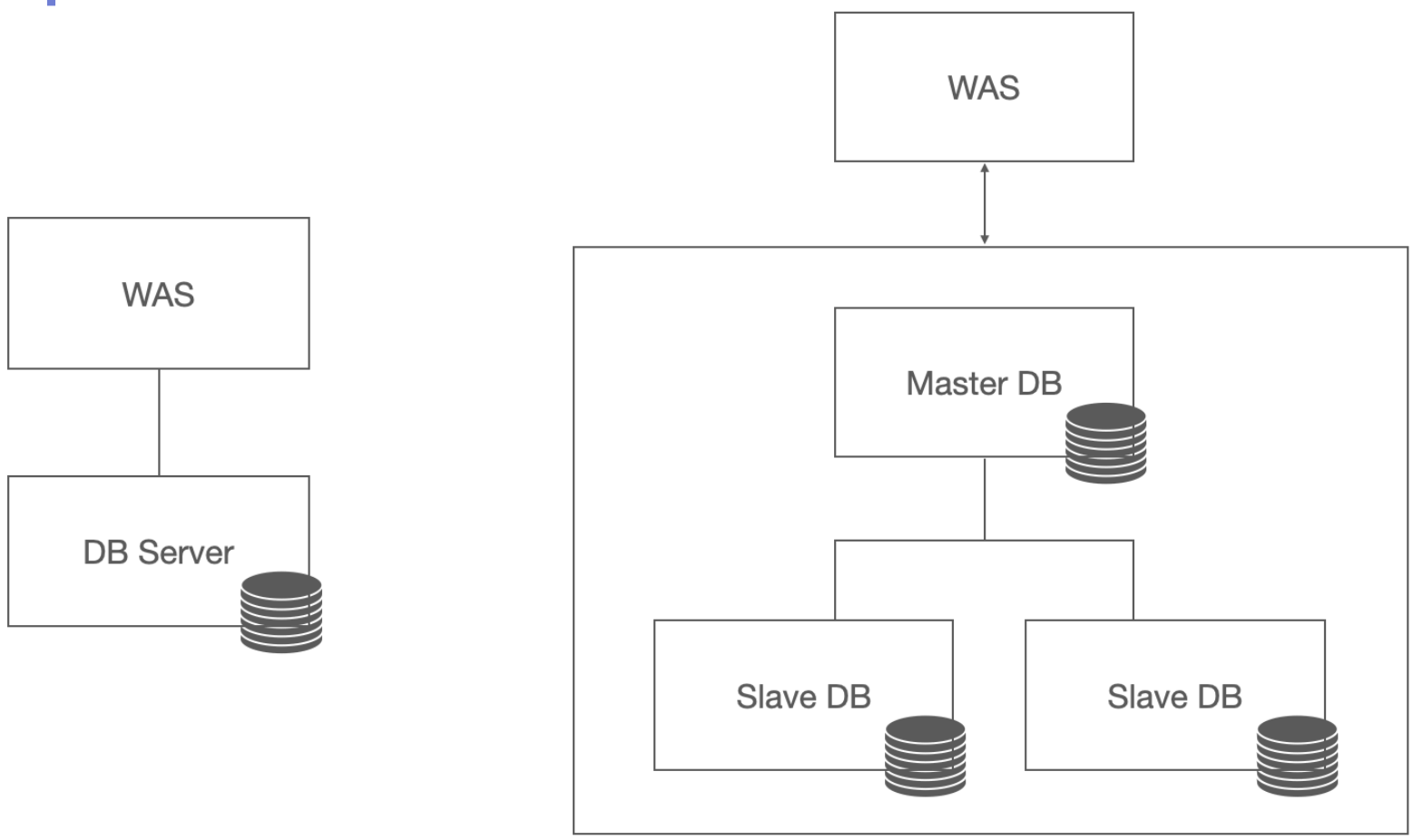
Infrastructure

Product 환경



- Reverse Proxy는 Nginx로 구성했습니다.
  - 로드 밸런싱을 적용했습니다. 이때 헬스 체크를 통해 WAS의 상태를 주기적으로 확인합니다.
  - 블루/그린 무중단 배포를 적용했습니다.
  - TLS와 HTTP 2.0을 설정했습니다.
- 운영 WAS는 총 2대가 운용되어 단일 장애점을 극복했습니다.
- 운영 DB는 Replication을 도입하여 R과 CUD를 분리했습니다.
  - Master DB 1대와 Slave DB 2대로 구성했습니다.
  - 조회 성능 개선을 이뤄냈습니다.
- Redis로 캐싱 기능을 구현했습니다.
  - 비 로그인 상태에서 메인 페이지를 캐싱합니다.
  - 조회 성능 개선을 이뤄냈습니다.
- Elastic Search로 검색 기능을 구현했습니다.
  - 조회 성능 개선을 이뤄냈습니다.

DB Replication



- 기존에는 WAS 1대와 DB 1대로 구성했습니다. (왼쪽 이미지)
- SNS 특성상 읽기 연산이 쓰기 연산보다 빈번하게 발생하고,
  DB를 1대만 두면 단일 장애점이 될 수 있다는 한계가 존재합니다.
- DB를 Scale-out해야 하나 고민하던 찰나,
  팀의 기대치만큼 서버가 트래픽을 처리할 수 있는지 성능 테스트를 진행하고 결정하기로 했습니다.
- 홈 피드 조회를 기준으로 부하 테스트를 했습니다.
  - 읽기 연산의 평균 TPS는 11.8, 쓰기 연산의 평균 TPS는 11.6이 나왔습니다.
- 목표 TPS를 17~50 정도로 잡아두었기 때문에 목표치보다 낮다고 판단했습니다.
  따라서, DB Scale-out을 위해 Replication을 도입하기로 했습니다.
- 이후에는 WAS 1대와 DB 3대로 변경했습니다. (오른쪽 이미지)
  - Master DB 1대와 Slave DB 2대로 구성했습니다.
  - Master DB에서는 CUD만, Slave DB에서는 R만 수행하도록 지정했습니다.
  - Slave DB는 라운드 로빈 방식으로 구현하여 트래픽이 균등하게 분산되게 했습니다.
- DB Replication을 적용하고 부하 테스트를 다시 진행했습니다.
  - 읽기 연산의 평균 TPS는 40.1, 쓰기 연산의 평균 TPS는 85.9로 늘어났습니다.
- 조회 성능 개선에 성공했으나, 여전히 아쉬운 점이 존재합니다.
  - DB Replication을 고려하던 당시 인덱스에 대한 이해도가 낮아, 먼저 인덱스 적용을 하지 못했습니다.
  - 홈 피드 조회 API 이외의 다른 조회 API에 대한 테스트를 하지 않았습니다.

작성 글

- [Database] DB Replication을 구성한 이유