About 한승엽

okayhan89@gmail.com

Intro

안녕하세요? 한승엽 입니다.
IT계에 취업을 하고 난 후,
취미와 일이 겹치는 것에 대해서 고민을 할 정도로 IT업무에 대해서 **순수한 열정**을 가지며 지내왔습니다.

회사에 들어와 여러 요구사항들과 그로 인한 불편함을 해결하고자 하는 노력도 중요하지만
주어진 일을 단순히 처리하는 사람이 아닌 일의 중요도와 업무강도 또한 고려하고, 함께 논의하여
능동적으로 한발 더 앞서나가는 업무 처리를 지향하고 있습니다.

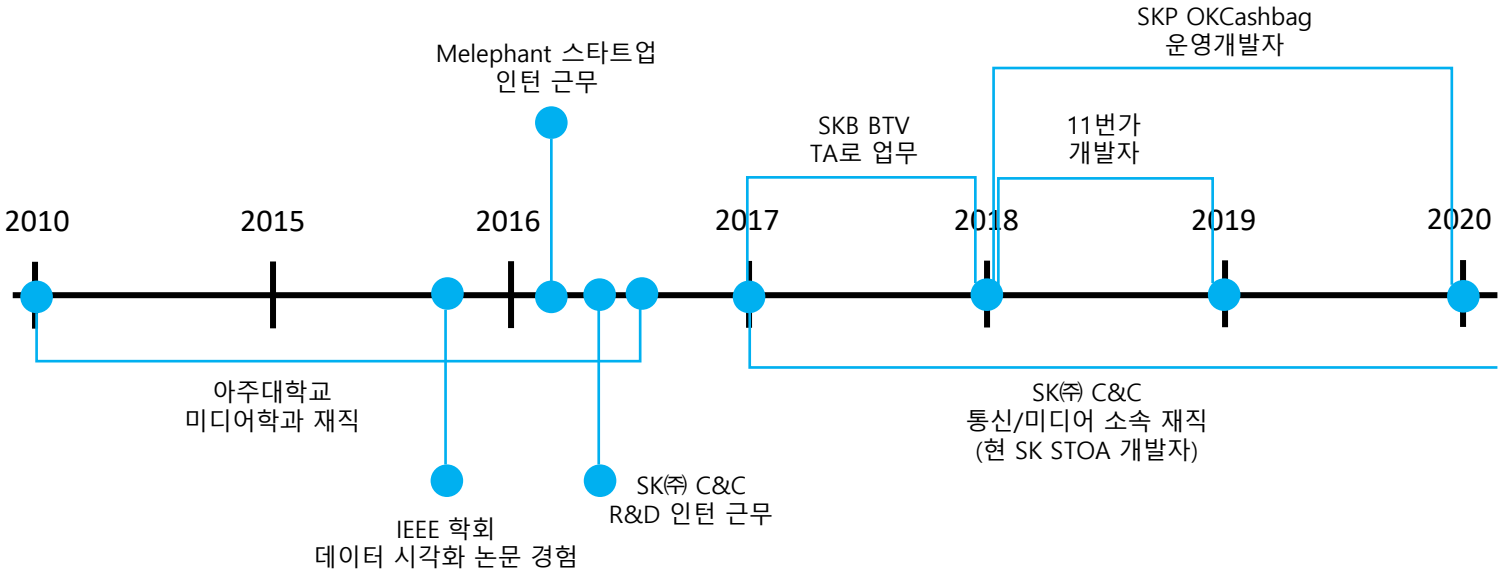
현재는 사회초년생의 순수한 열정에 연차와 책임감이 더해지면서 **저만의 자기완결성**을 찾아가고 있습니다.
지원부서의 Full-Stack 개발자가 되기 위해 다양한 boundry에서의 경험이 쌓였는데,
한 분야에 대해서 깊게 파고 들어 개발하는것도 중요하게 생각하지만 큰 그림을 통해
"어떻게 개발하는 것이 시스템에 어울리는가?" 에 대해서 많이 생각하고 발전해 나가는 중입니다.

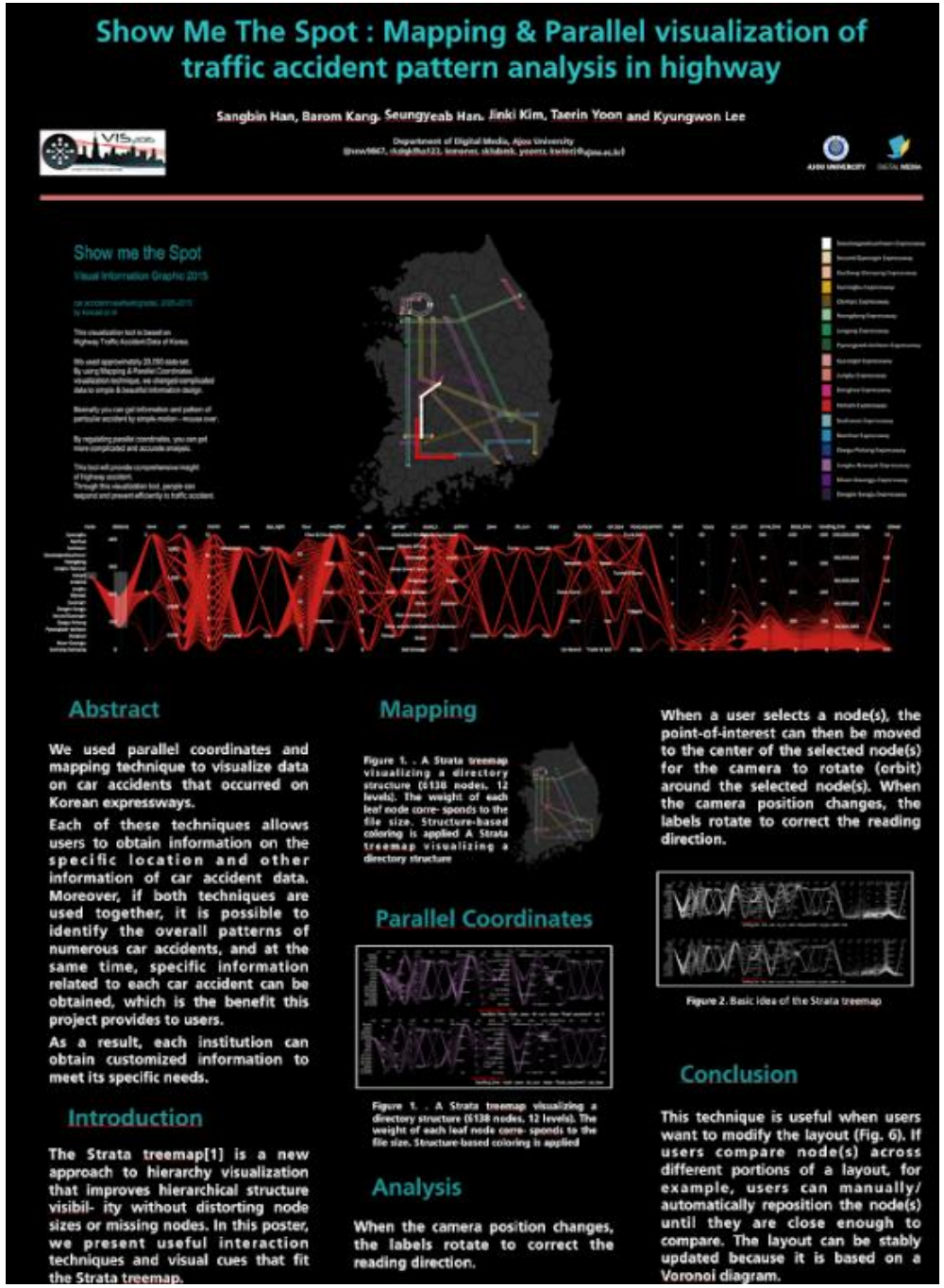
저는 여전히 현재진형형으로, 새로운 테크닉과 많은 경험을 가지신 분들의 이해도 앞에서는 배움이 필요하다고 느낍니다.
full-stack 개발자 시각에서 조금 더 안정적이며 **완벽한 서비스**를 위해서 더 나아가고 싶어 지원하게 되었습니다.

Profile

- 한승엽 (Han SeungYeob)
- okayhan89@gmail.com
- https://github.com/okayhan89
- 아주대학교 미디어학과 학사
  (2010.03 ~ 2016.08)
- Melephant
  (2016.01 ~ 2016.03)
- SK 주식회사C&C 통신/미디어Digital 추진그룹 (2017.01 ~)
- MSA / Devops / Spring / WEB-Frontend / REDIS / ELK / Oracle

Yeob's Career Path





Experiments

- 주요 활동
  - **IEEE VIS 2015** 학회 포스터논문 기재
- 논문 제목
  - Mapping & Parallel visualization of traffic accident pattern analysis in highway
- 논문 요약
  - parallel coordinates 와 map 의 연동처 리를 통한 데이터 시각화
  - **교통 사고 데이터를 통한 사고지역 예측**
- 수행환경
  - javascript (D3.js) / Google App engine / Spring MVC
- 동영상 URL
  - https://www.youtube.com/watch?v=_eoa0E0iyOw

SKB G2 프로젝트

- **프로젝트 설명**
  - SKB 시스템 MSA, Cloud (**Paas**) 구축 및 전환
- **프로젝트 수행 기간**
  - 2017. 02 ~2018.03
- **프로젝트 상세 업무 설명**
  - SKB TV 메뉴 / 인증 / 즐겨찾기 / 시청내역 / 구매내역 MSA 전환
  - SKB Private Cloud 환경의 PaaS Infra 구축
  - SKB DevOps 적용을 통한 통합운영
- **담당 업무**
  - 통합 Arch 파트에서의 **Technical Arch**
  - 타 파트에서의 이슈 트러블 슈팅 및 가이드라인 제시
  - **Backing Service** 구축 및 가이드

- **얻은 바 & 아쉬운점**

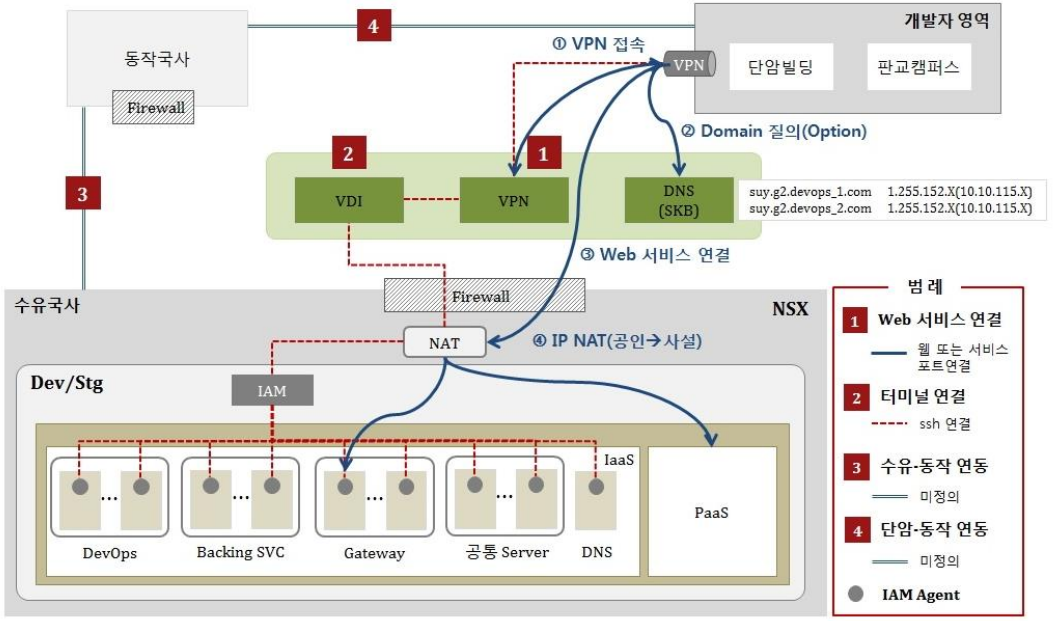
현재에는 많은 사용을 하고 있지만, 당시 접하기 어려웠던 **Cloud Foundry** 를 구축하며 그에 필요한 Backing service 를 구축하였습니다.

개발자를 위한 개발을 하면서 시스템을 조금 더 이해할 수 있었습니다.

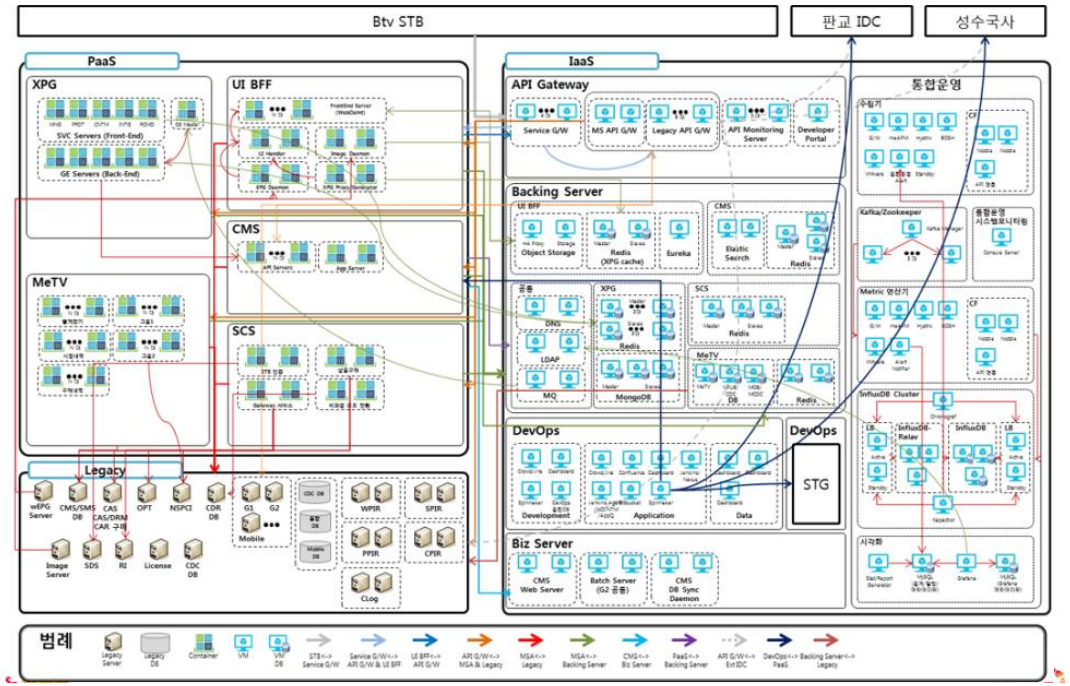
예를 들어, 기업의 개발환경구성을 구축, H/A 구성을 통한 **Fail-over Test**을 통해 안정적인 infra 를 구성, **Blue/Green 배포**를 통해 TR유실을 적은 배포환경을 구성하는 것은 개발자에만 머물러 있었다면 이해하기 힘든 구조였었습니다.

다만, 현재에는 CaaS (k&8) 을 사용하는 구성과 비교하였을 경우 부족한 점이 다소 있고, VCPU 와 Auto-Scaling을 서로 상충되는 개념으로 프로젝트를 구성했기에 안정성은 충분하나, 서버자원을 비효율적으로 활용되는 점은 아쉬운 부분입니다.

SKB G2 프로젝트



[개 발 환 경 구 성 도]



[시 스 템 구 성 도]

- **Infra 개발 환경 구축**
  - Backing Service 설치 및 관리
    : Redis / Swift / Rabbit MQ 설치 및 관리
  - IaaS 관리
    : application 은 PaaS / 그 외의 영역은 IaaS로 관리
  - NAT 관리
    : 개발자들이 VPN으로만 통신하여 보안 안정성 관리
  - DNS 구축 및 관리
    : 내부/외부 DNS 설정을 통한 Domain 호출 가능
  - LDAP 구축 및 관리
    : 통합 로그인이 되도록 LDAP 활용 및 가이드

- **아키텍처**
  - 개발언어
    : JAVA Spring Boot (Cloud Foundry 향)
  - 시스템 구성 설계
    : **셋톱 - REACT – API G/W(CA 솔루션) –**
    BFF(Spring Boot) – 영역별 MSA – Backing Service

11번가 SNAPSHOT

- **프로젝트 설명**
  - 11번가 시스템 Cloud (**SaaS**) 구축 및 전환
- **프로젝트 수행 기간**
  - 2018. 02 ~2019.03
- **프로젝트 상세 업무 설명**
  - 자유 형식으로 업로드한 이미지 및 html 페이지를 규격에 맞게 가상브라우저에서 html로 load하여 **스크린샷을 저장 (ex : 웹툰)**
- **담당 업무**
  - API G/W 개발자
  - 모니터링 시스템 개발자

- **챌린지**

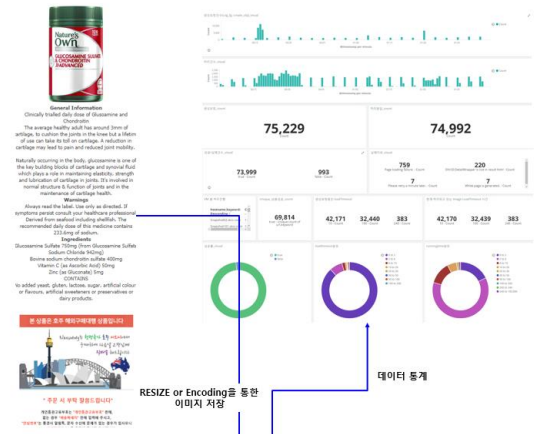
상품 이미지 Mig하는 방법 대신 SaaS 서비스를 통해 새롭게 데이터를 생성하고자 하는 Needs가 있었습니다.

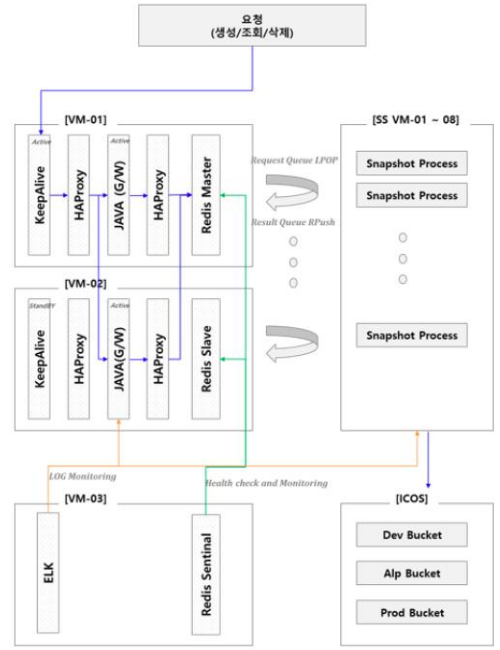
따라서 호출하는 API G/W 의 서버 구성이 중요하게 되었습니다. 왜냐면 Cloud 이기 때문에 더 많은 **비용**이 발생하기 때문입니다.

해결 방법으로는 일반적인 API G/W 의 SYNC 구조를 벗어나, **Async 구조**를 착안하여 Queue 로 진행하여 CUP 의 과부하를 줄였습니다.

또한 기본 1TR시, One Request - One Response 인 구성을 벗어나, ArrayList 로 I/F함으로 인해 요청에 의한 Memory 부하도 줄였습니다. **기존의 1000 TPS 로 요청하였던 것을 1TPS 로 줄이는 효과**로 서버사용을 줄일 수 있었습니다.

Async의 단점을 최소화 하기위해 **모니터링 시스템**에 중점을 두고 추후 진행하였습니다.





11번가 SNAPSHOT

- 아키텍처
  - **Gateway**
    - KeepAlive
      - Active – Standby 구성을 통해 VM Shutdown 시 이중화
    - HAProxy
      - Active – Active 구성을 통한 L7 이중화
    - JAVA (Spring Boot)
      - 서비스 인증 및 요청값을 Redis 에 적재
    - REDIS
      - Master/Slave 구성을 통한 HA안정성 확보
  - **Snapshot**
    - python
      - Queue Polling → Snapshot 처리 → Object Stroage 적재 및 응답값 REDIS 적재
      - VM 1개당 프로세스 여러 개 수행
  - **모니터링**
    - ELK 스택 / NGINX / Spring Boot

OK Cashbag 시스템 운영 프로젝트

- **프로젝트 설명**
  - OK Cashbag 시스템 운영
- **프로젝트 수행 기간**
  - 2018. 02 ~2020.02
- **프로젝트 상세 업무 설명**
  - 오픈형 마일리지 시스템 OK Cashbag 운영 및 유지/보수
- **담당 업무**
  - 정 가맹 / 부 회원 업무 담당자
- **규모**
  - 약 3,000 만명 회원
  - 최대 TPS 1,1000 TR

- **챌린지**

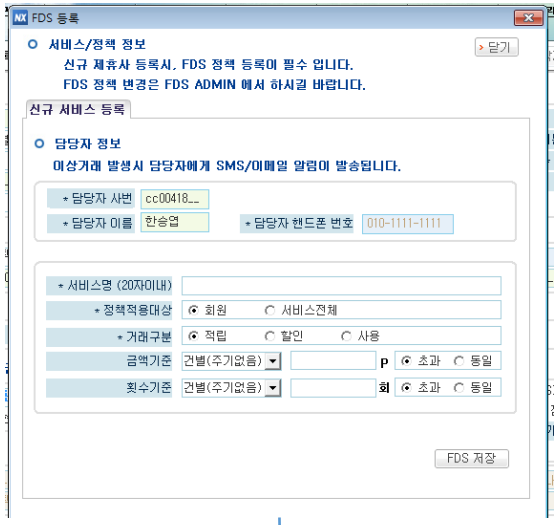
OK Cashbag 의 back단에는 운영의 안정성과 고객의 Needs 가 없어 현재까지도 **Java 1.4 version 으로 운영**되며 proC등 오래전 운영개발이 된 언어로 진행 되고 있습니다.

해당 환경에서의 JSON 통신은 어려움이 존재하였습니다. 따라서 **Service G/W** 라는 개념을 통해 해결하고자 하였습니다.

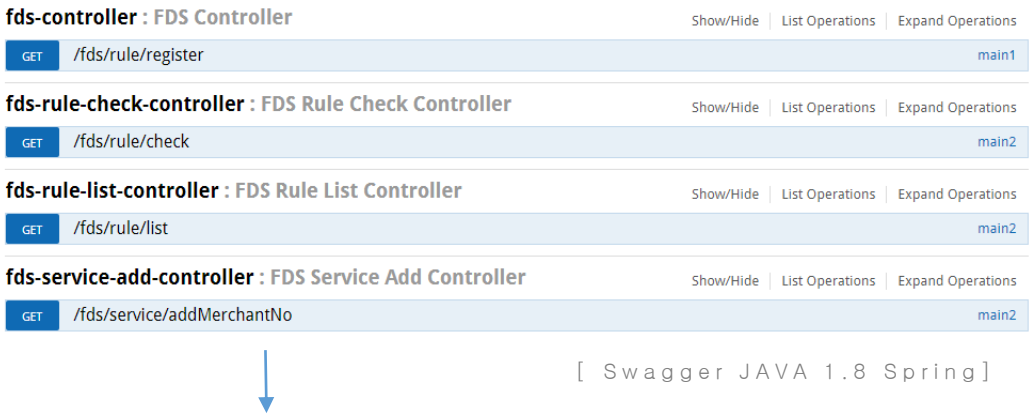
영향도가 적은 서버에 java 1.8version 으로 설치하여 Spring Boot을 구성하였습니다. 해당 문제해결을 통해 타사와의 I/F 의 안정성 뿐만 아니라, 추가적인 변경에도 빠른 대처할 수 있었습니다.

제가 해결방법을 위와 같이 하려고 했던 이유는 **오래된 시스템에서의 안정성을 도모하고 신규서비스에 대해서 확장성** 까지 확보하고자 진행하였습니다.

다만, 시스템 안에 **새끼 프로세스**가 생겨 관리 Point 가 생겼으나, 운영업무의 누가 되지 않게 팀원들에게 프로세스체계를 전파하였습니다.



[OCB 화면 JAVA 1.4]



[Swagger JAVA 1.8 Spring]

OCBFDS API

도메인

구분	개발도메인	IP	PORT	운영도메인	IP	PORT
fds api	fdsdev.okcashbag.com	172.22.245.86	80,443	fds.okcashbag.com	172.22.198.168	80,443

[기존 FDS 시스템의 JSON I/F]

OK Cashbag

- FDS 시스템 개발
  - 개발 언어 : Spring (JAVA 1.8) , JEUS (JAVA 1.4)
  - 개발 내용
    가맹점등록시, FDS정책을 활용할 수 있도록
    기존 SKP FDS 시스템과의 연동
  - 이슈 상황
    OK Cashbag 의 기존 시스템의 JAVA 버전이
    낮은 이유로 외부와의 JSON I/F 시 많은 예외처리 필요하였음.
    따라서 해당 이슈해결하기 위해 GW형식의 Spring 을 통해
    I/F 함.
  - 해결 방안
    JAVA 1.4 에서의 I/F 하여야 될 것들에 대해
    JAVA 1.8 FDS G/W 를 통해 I/F 되도록 변경

TOY 프로젝트

- **프로젝트 설명**
  - Don't Sleep
- **프로젝트 수행 기간**
  - 2017.06 ~ 2017.08
- **프로젝트 상세 업무 설명**
  - 차량 생활데이터/운전자 눈동자의 데이터를 통해
    운전자 수면 방지
- **담당 업무**
  - 실시간 웹 서비스구축
    (Spring Boot, AWS – RDS , AWS – BEANSTALK, AngularJS)
  - 프로젝트 PM 및 아키텍처
- **동영상**
  - http://youtu.be/9_OwcG6B-w8

- **배운 점**

IOT 머신 데이터를 실시간 으로 저장하고 그에 따른 분석을 동시에 진행하여야 하는 것이 목표였습니다.

IOT 실시간 데이터를 수집하기 위해서 Application에서의 DML 이 많이 일어나야 되었습니다. 당시, **AWS에서의 RDS** 의 DB의 Auto-scale을 습득하고 배워 지연을 줄이며 서버에 대해 재구성을 할 수 있었습니다.

아쉬운 점으로는 당시, **IMDB 를 활용한 구성**을 조금 했었다면 속도적인 면이나, RDS의 비용을 줄일 수 있었을 거라 생각됩니다.

TOY 프로젝트

- **프로젝트 설명**

  - Show Me The My HotSpot
- **프로젝트 수행 기간**

  - 2019.10
- **프로젝트 상세 업무 설명**

  - 맛집 뿐만 아닌 매력적인 상점에 대한 정보를 주는 프로젝트
- **담당 업무**

  - 회원 서비스 담당자 (JPA, K&8, Devops, Spring Securtity, JWT)
- **동영상**

  - https://github.com/okayhan89/hotspot-mbr

- **배운 점**

서비스 기획 담당과 개발을 같이 진행하였습니다. 대학생 때의 저의 모습은 불편한 것, 필요한 것에 대해서 제가 직접 개발하고자 하였습니다.

회사에 오고 난 뒤, 회사가 필요한 서비스를 만들다 보니, 흥미가 떨어진 저의 모습을 볼 수 있었습니다. 하고 싶어하는 일과 잘하는 일은 다소 다르겠지만, 좋은 동료들과 하고 싶은 업무를 같이 할수 있는 기회였기 때문에 꺼져갔던 **순수한 열망**을 갖게 한 프로젝트였습니다.

Cloud Foundry만 경험하였었는데, K&8구조를 통해 프로젝트에 활용할 수 있게 되어 즐거웠었습니다. **또한 협업에서의 MSA 에서의 Aggregation역할이 필요한 이유를 해당 프로젝트에서 배울 수 있었습니다.**

기타 Stack 및 부수 업무

- **프로젝트 설명**
  - 안정적인 DNS 구축
  - 안정적인 Redis 구축
  - 안정적인 Object Storage 구축
  - TR 속도 개선
  - 안정적인 서비스 구축
  - 운영 효율 및 모니터링
  - 운영효율화

- **TMI**

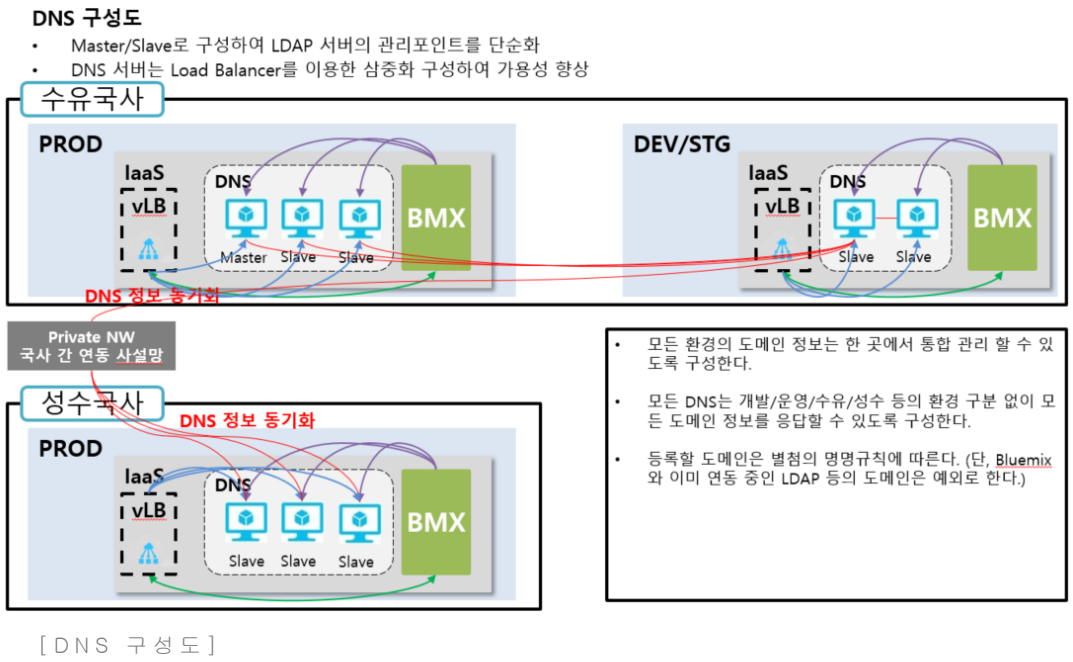
업무가 아닌 것에 대해서도 영역에 대해서도 관심을 가지고 함께 해결하려는 성격이 있습니다. Infra/MiddleWare/Front 두루두루 여러 시야에서 다양하게 생각하는 것을 좋아합니다.

예를 들어 한프로젝트에서 Spring boot 의 서킷브레이커는 REACT에서 try/catch예외처리와 같은 처리를 하게 될 것입니다.

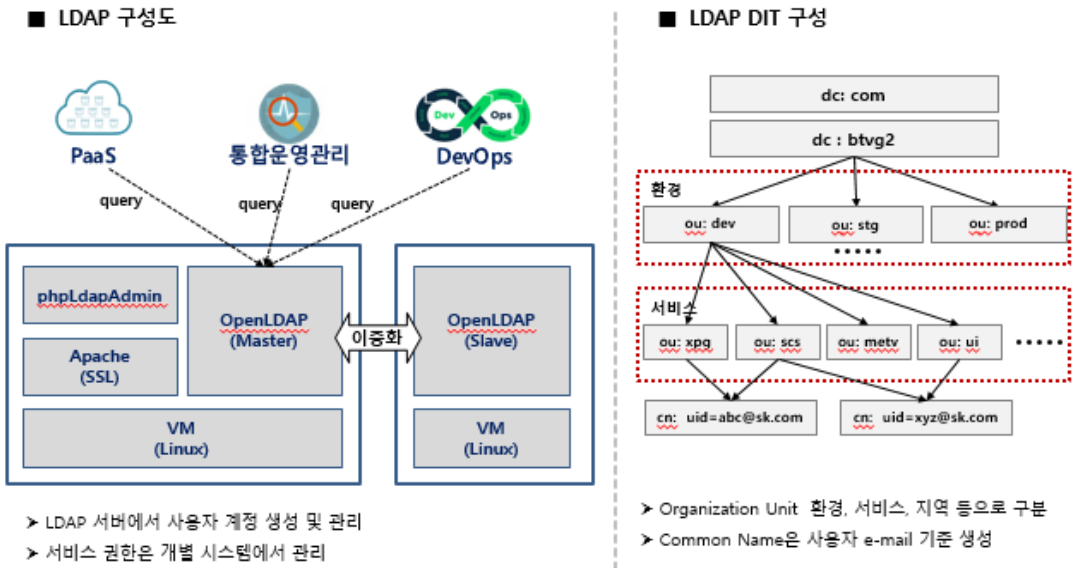
막상 이렇게 업무를 하다 보니 하나에만 집중하는 친구들에 비해서 커리어 패스에 두려움은 갖게 되었지만, **해결방법을 다양하게 가지고 갈 수 있어 좋았습니다.**

현재에는 FRONT에 대해서 소홀하다고 느껴, REACT를 회사 외적으로 공부하고 개인적으로 프로젝트를 하고 있습니다. (배움에는 끝이 없는 없는 영역이라 생각합니다.)

앞으로도 꾸준히 발전할 거라 자부하며 이만 TMI를 마치겠습니다 ^^;



[DNS 구성도]



[LDAP 구성도]

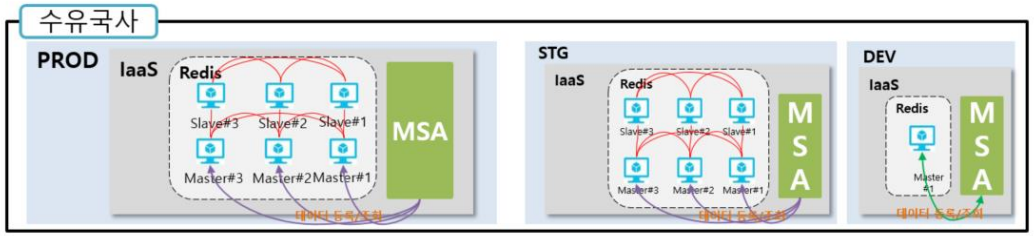
기타 Stack

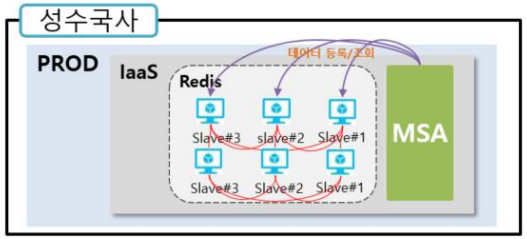
- DNS 구축 및 관리
  - DEV/STG/PRD DNS 가 하나의 Master 로 관리
  - vLB/GSLB 를 통해 Domain 분기처리
  - 외부 DNS / 내부 DNS 따로 관리
  - HA 구성을 통해 서버 8중화로 관리
  - 환경 : Ubuntu → CentOS 로 변경관리

- LDAP 구축 및 관리
  - 시스템 통합 로그인 체계 구축
  - openLDAP 을 통해 도메인 (환경/서비스) 구분값 관리
  - DevOps (JIRA/Confluence/GIT...) 을 통한 PW 수정 관리
  - DEV/STG/PRD LDAP 12중화로 관리
  - 환경 : Ubuntu → CentOS 로 변경관리

기타 Stack

- Backing Servie 구축 [REDIS]
  - DEV/STG/PRD 전부 다른 구성으로 진행 (자원)
  - PRD 국사간 동기화를 통해 관리
  - 12중화로 서버의 부하시 IMDB단점을 최소화
  - 서버 구축가이드를 통해 기본 IMAGE 제작
  - REDIS Fail-over 테스트를 통해 서버스펙산정
  - REDIS-STAT 페이지를 통해 모니터링 진행





1. REDIS서버구축가이드

REDIS 서버구축에 필요한 표준가이드를 설명하는 문서입니다.

본 가이드 는 CentOS 7.3을 기준으로 작성되어 있습니다.

1.1 REDIS 서버구축

1.1.1 전제 조건

☒ 클러스터를 위해서 3대의 Master와 3대의 Slave를 구성합니다.

1.1.2 시나리오

REDIS를 Master / Slave 형태의 클러스터를 구성 합니다.

예제 REDIS 서버

Host	Role	Private IP Address	PORT
Redis01	Master	192.168.56.101	6379
Redis02	Master	192.168.56.102	6379
Redis03	Master	192.168.56.103	6379
Redis04	Slave	192.168.56.104	6379
Redis05	Slave	192.168.56.105	6379
Redis06	Slave	192.168.56.106	6379

XPG Redis 테스트 이벤트 및 시작 시각 (2017.10.11 PRD 환경)

테스트 구분	테스트 이벤트		시작 시각	비고
정상 부하 테스트	테스트 준비 (부하 발생 전)		13:50:00	
	수유 Write 부하 발생		13:54:50	- 쓰기 부하 시작
	수유 Read 부하 발생		14:00:50	- 쓰기/읽기 동시 부하 시작
	성수 Read 부하 발생		14:08:00	
	장애 대응 테스트 진행 구간			- 14:22:00 ~ 14:49:00
	수유 Write 부하 제거		14:49:00	
장애 대응 테스트	Master 서버 1식 Down	수유 M3 Redis Service 수동 Stop	14:22:00	
		수유 Slave 자동 승격		- 자동 Failover 완료 시점
		Service 수동 Start	14:23:10	- Master 서버가 Slaver 로 시작됨
		M-S 수동 Change	14:26:20	
		M-S 수동 Change (수유->성수)	14:29:10	- 국사간 M-S 전환시 특이사항 확인
		M-S 수동 Change (성수->수유)	14:31:30	
	Slave 서버 1식 Down	수유 S3 Redis Service 수동 Stop	14:33:50	
		Service 수동 Start	14:35:01	
	Master 서버 2식 순차적 Down	수유 M3 Redis Service 수동 Stop	14:41:10	
		수유 Slave 자동 승격 (수유S3)		
		수유 M2 Redis Service 수동 Stop	14:43:10	
		수유 Slave 자동 승격 (수유S2)		
		Service 수동 Start	14:43:54	
		Service 수동 Start	14:44:10	
		M-S 수동 Change	14:45:51	
		M-S 수동 Change	14:46:30	

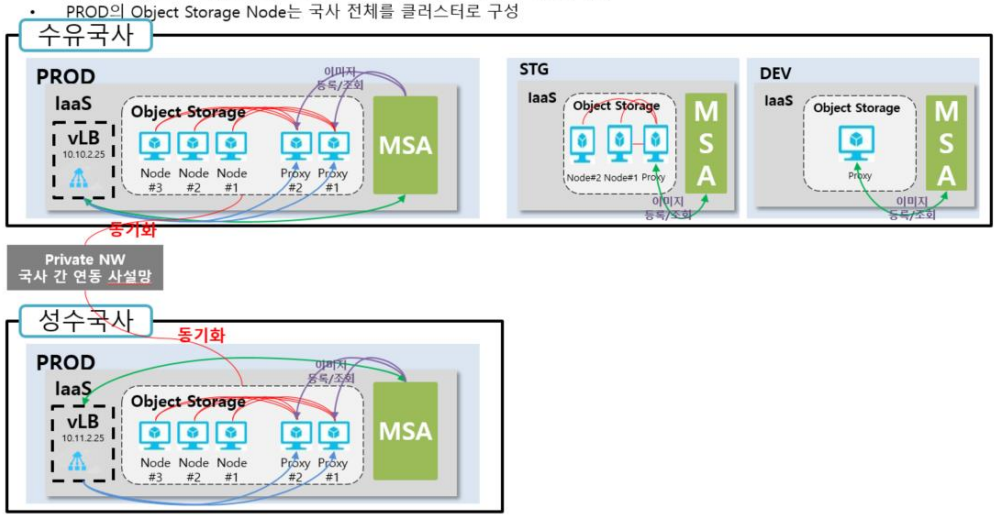
기타 Stack

- Backing Servie 구축 [Obejct Stroage]
  - Openstack SWIFT 사용
  - KEYSTONE 을 통해 MSA 별 파트 계정 인증 시스템 활용
  - VLB → HAProxy → NODE 구성을 통한 관리 체계 도입
  - STORAGE 당 동기화는 진행하지 않게 관리
  - API G/W 세션 관리를 Object Storage 에서 관리
  - Fail-over 테스트를 통해 서버스펙 산정

5) Object Storage 구성도

1. 구성도

- PROD의 Object Storage Proxy 서버는 각 국사 별 HA 구성하여 가용성 향상
- PROD의 Object Storage Node는 국사 전체를 클러스터로 구성



1. OBJECT STORAGE서버구축가이드

OBJECT STORAGE 서버구축에 필요한 표준가이드를 설명하는 문서입니다.

본 가이드는 Ubuntu 16.04 및 오픈스택 Swift Ocata를 기준으로 작성되어 있습니다.

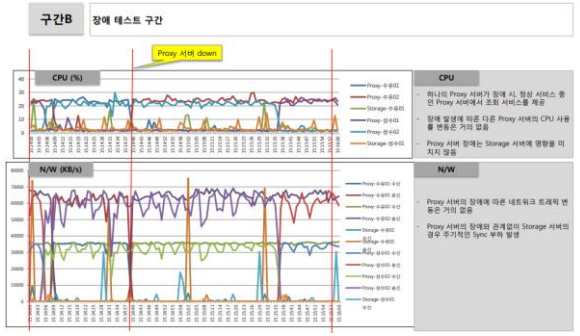
1.1 OBJECT STORAGE 서버구축

1.1.1 전제 조건

☒ root 계정으로 구성을 진행합니다.

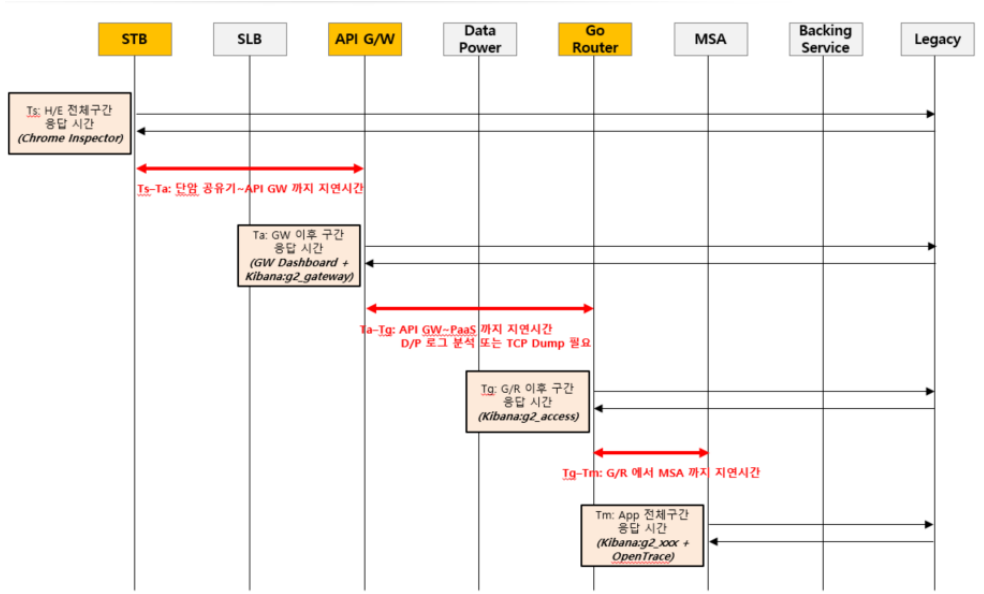
1.1.2 시나리오

OBJECT STORAGE는 Swift Proxy와 2개의 Storage Node 형태로 클러스터 구성을 합니다. 본 시나리오에서는 링 구성에 있어서 최소 3개의 스토리지 디바이스가 필요합니다. 따라서 각 Storage Node는 2개의 스토리지 디바이스를 가지고 있습니다.



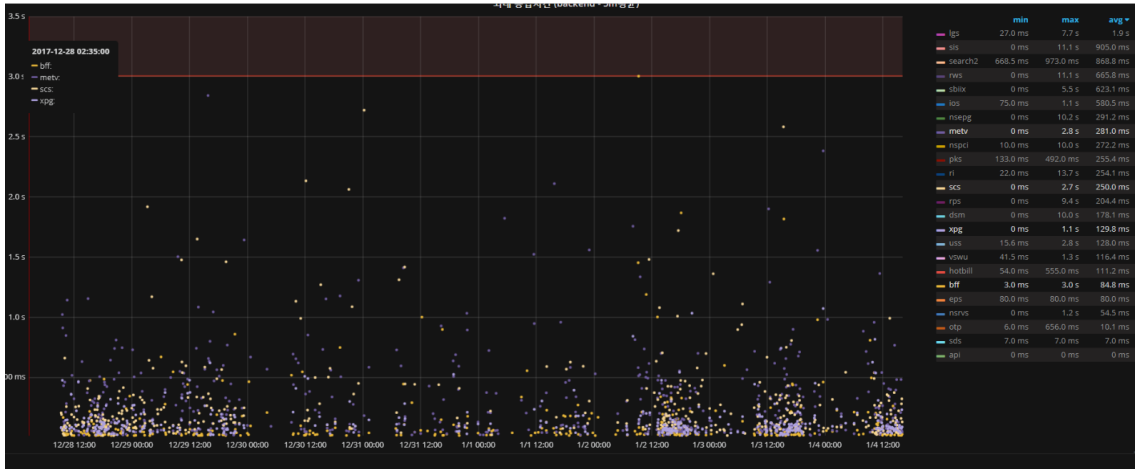
시스템 구성

Proxy Node	Storage Node#1	Storage Node#2
192.168.56.101	192.168.56.102	192.168.56.103
• Swift Proxy • Keystone • MariaDB • Memcached • httpd	• Swift-Account • Swift-Container • Swift-Object	• Swift-Account • Swift-Container • Swift-Object



❖ 플랫폼 전체 자원 현황은 통합 운영관리의 **Grafana** 대시보드를 통해, CPU, Memory, NW 트래픽에 대한 이상 상황 병행 점검

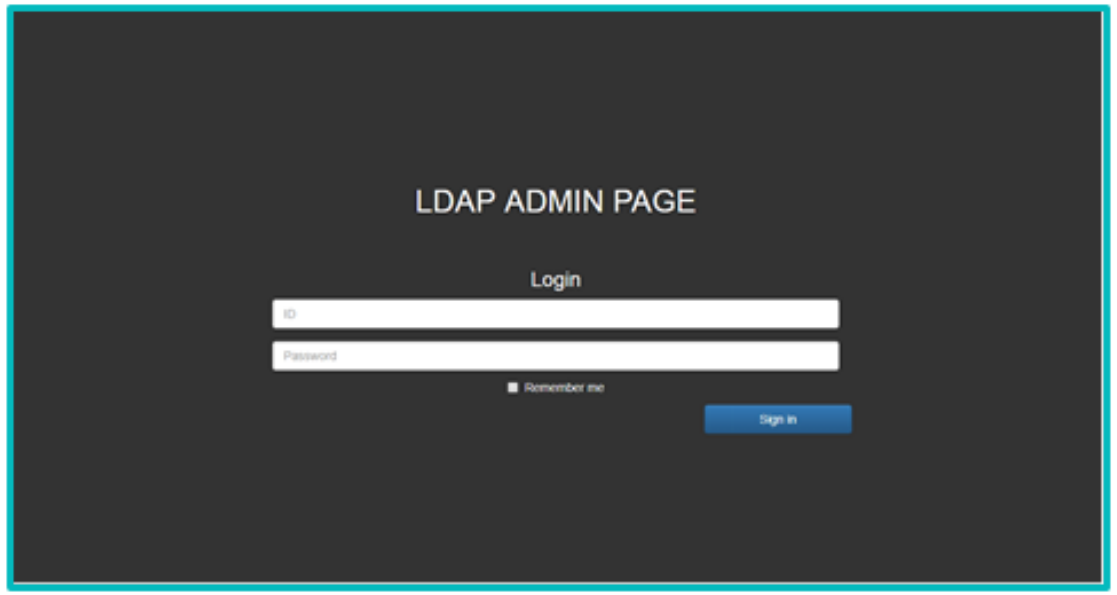
[구 간 분 석]

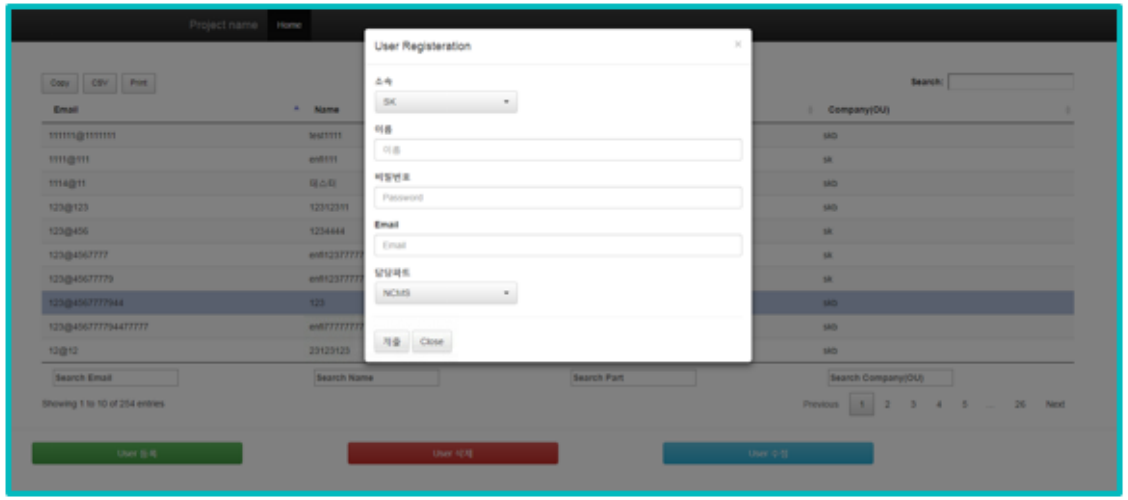


부수 업무

- 트러블 슈팅 (일정하지 않은 TR)
  - 환경 구성도 분석을 통해 현 TR 의 Flow 확인
  - 모니터링 시스템을 통한 TR Timeout 체계 구축
    : Grafana / ELK 를 통해 오류 TR 분석 강화
  - 패킷 분석을 통한 에러 원인 확인

- 원인
  - LEGACY 서버와의 연동 시, 일정하지 않는 구간 발생





부수 업무

- LDAP ADMIN PAGE
  - 개발개요
    : LDAP 서버를 통하여 앱과 유저 그룹을
    등록/관리할 수 있는 웹 애플리케이션 개발
  - 개발언어 : NodeJS, HTML5, BootStrap, Javascript, LDAP
  - 개발내용
    - ➔ 로그인 정보 관리를 위해 ADMIN 계정으로만 로그인 가능
    - ➔ 선택한 APPLICATION 그룹으로 USER 등록
    - ➔ USER 기본 정보 수정 기능
    - ➔ USER 패스워드 초기화
    - ➔ USER 등록 정보 속성 관리
    - ➔ APP 및 등록사용자 리스트 조회
    - ➔ USER 일괄 IMPORT / EXPORT 기능
    - ➔ 도움말 페이지 제공

부수 업무

- 담당업무
  - G/W 개발

    암호화 KEY I/F 개발

    ➔ SaaS 요청 시, KEY TOKEN 방식을 통해 I/F

    생성 요청 I/F

    ➔ 생성요청 시, 암복호화/유효성검증 이후 Redis 에 요청값 적재

    조회 요청 I/F

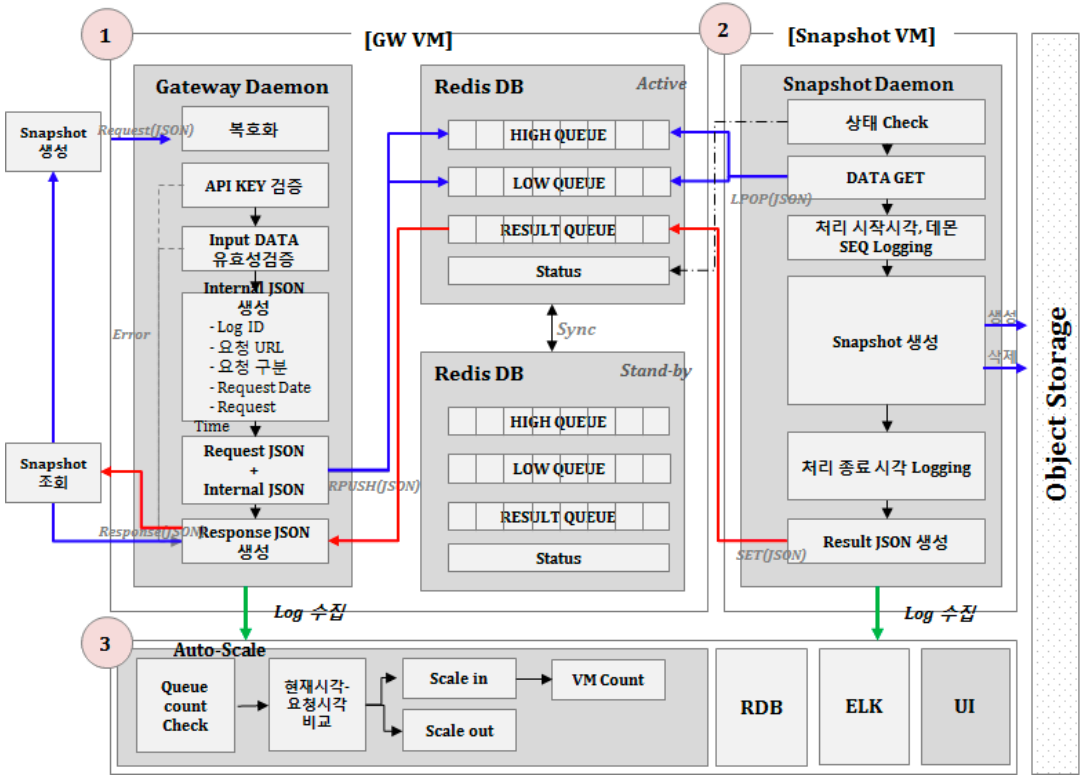
    ➔ 조회요청 시, 암복호화/유효성검증 이후 Redis 에 응답값 조회

    HEALTH-CHECK API

    ➔ Health-check 을 통해 호출 여부 판단 로직

    REDIS Queue Count API

    ➔ Redis Queue 에 대한 정보 API 개발



2.1.1 스냅샷 토큰 요청 HTTP interface

스냅샷 토큰은 하루 단위로 생성

Get 호출

Request URL : http://169.56.69.18:8443/api/v1/snapshot/token

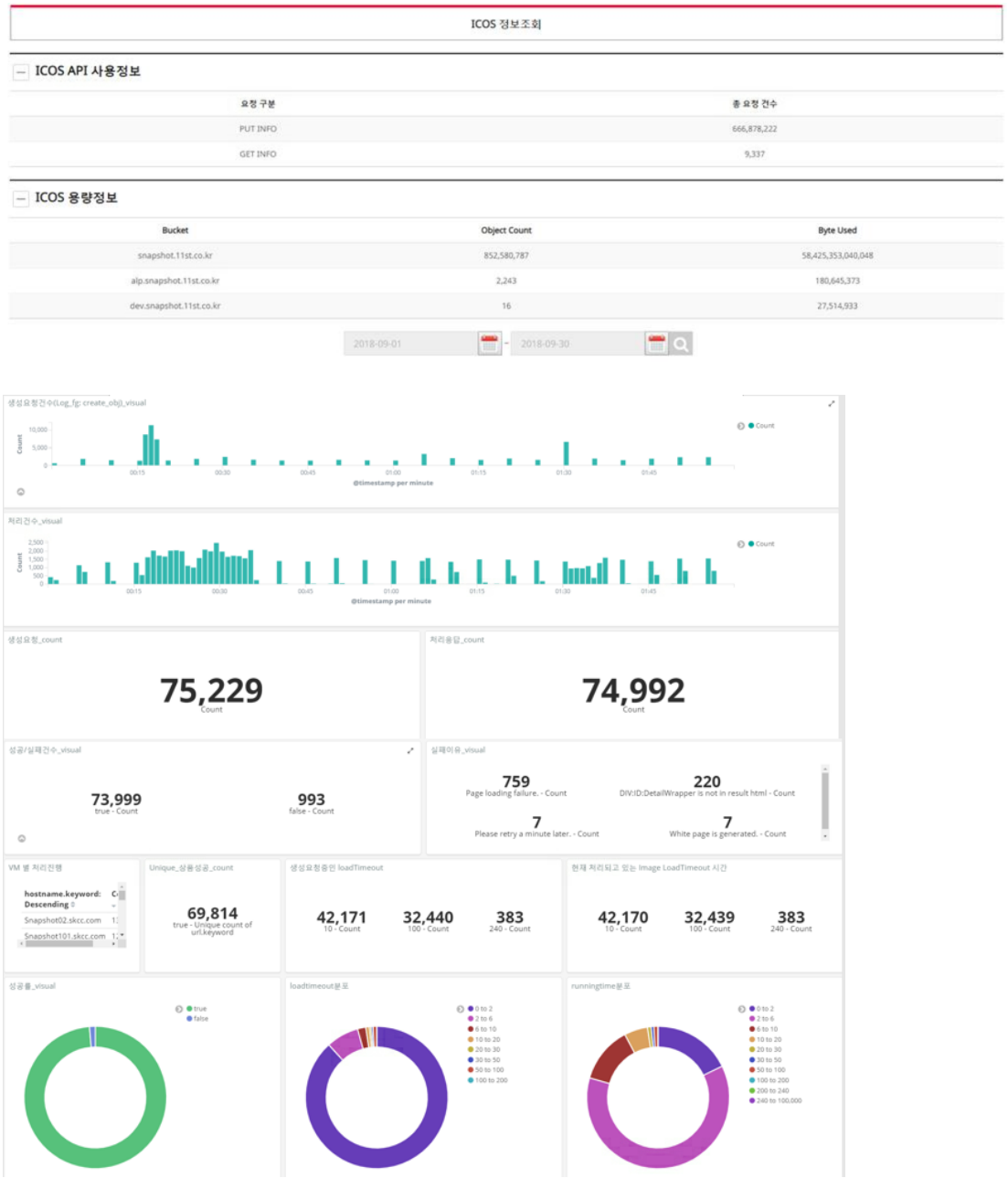
Response ex) :

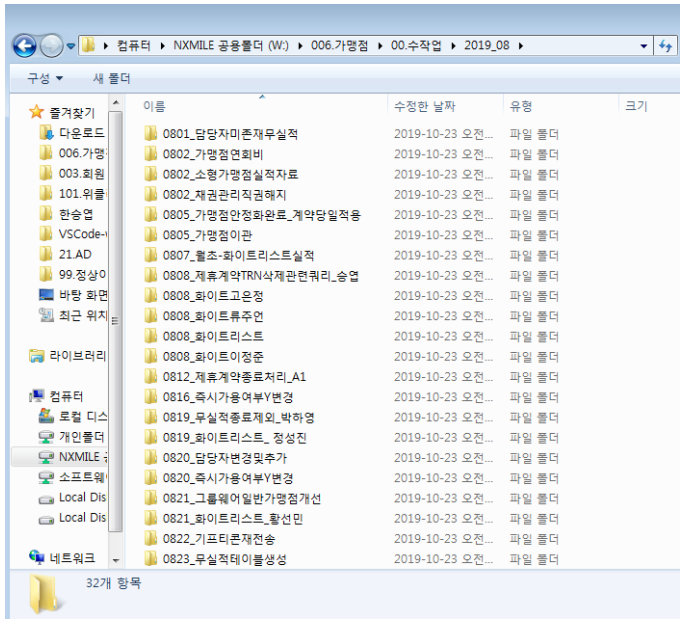
```
{
    "auth": "TWpBeE9DMHdOUzB6TVE9PQ=="
}
```

TWpBeE9DMHdOUzB6TVE9PQ== 값을 이용 가능

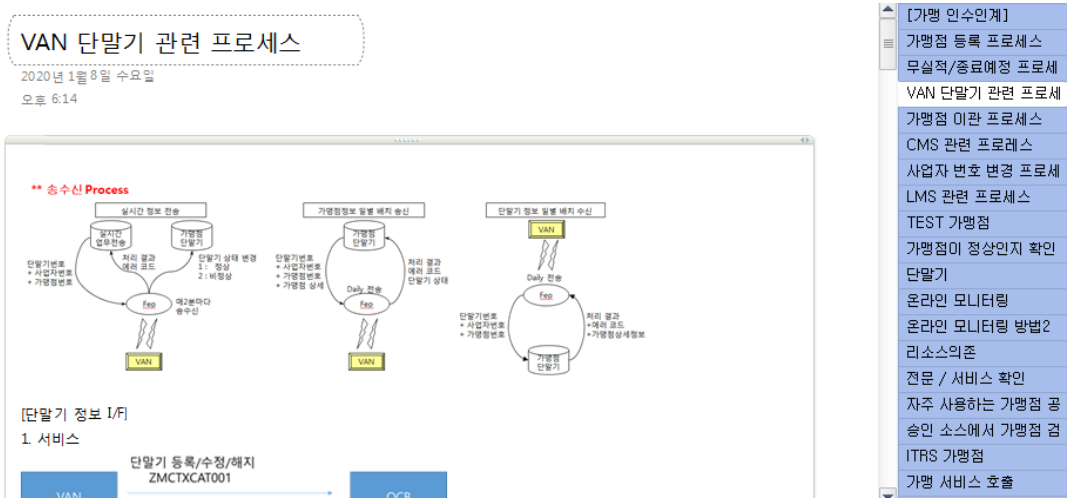
부수 업무

- 담당업무
  - 모니터링 개발
    - Object Stroage 사용량 화면 개발
      - ➔ IBM API 시, Class A/B/C 등의 HTTP 콜에 따라 사용량 계산
    - KIBANA 화면 조정
      - ➔ 각 이슈정리를 위해 KIBANA 화면 개선
  - 기타 운영개선
    - 오류코드 추가 개선
      - ➔ 네트워크의 에러로 이미지 로딩시,
        엑박이 도래될 가능성을 예외처리 되도록 변경
    - REDIS Queue NAME 개선
      - ➔ 기존 스태틱하게 관리한 Queue name 유연하게 DB로 관리
    - WEBP 개선
      - ➔ Cloud 처리시간을 단축 위해 jpg 이미지를 ,
        google image format인 webP 사용하도록 변경





[DML 수작업 처리 내용 정리]



[운영 매뉴얼/프로세스 정리]

부수 업무

- 운영 개발/개선 건
  - 개발 언어 : Pro*C (Proframe – Tmaxsoft)
  - 개발 내용

    SKT 가맹점 등록배치 개발 (2019.01 ~ 2019.03)

    가맹점 계약등록 1년제어 (2019.03 ~ 2019.05)

    무실적 가맹점 계약 종료 처리 (2019.05 ~ 2019.06)

    제휴사 FDS 시스템 연동 (2019.07 ~ 2019.08)

    유종별 고객 등급관리 개발 (2019.08 ~ 2019.09)

    이메일 전자결재 변경 (2019.10 ~ 2019.11)

    운영성 모니터링 메일 정리 (2019.11 ~ 2019.12)

- 운영 체계화
  - 운영 업무 프로세스 내용 정리

    10년이상 동안 운영되는 시스템이나,

    구전으로 운영되는 인수인계 내용 정리 및 체계화

Thank You