융복합 프로젝트 **4**조

<같이 걷개**> Cloud** 보고서

구정민, 송강현, 안민규, 조슬기

[목차**]**

개요

1. 주제 설명

Serverless 환경에서 동작하는 반려견 중심의 산책 데이터 수집을 통한 체계적인 산책 데이터 관리 및 반려견의 건강 관리 어플 개발

- IoT 웨어러블 기기를 통한 산책 데이터 수집 및 활용
- 실시간 날씨, 장소 정보 등 산책 시 견주가 필요로 하는 데이터 제공
- 견종별 적절한 산책 가이드라인 제공을 통해 체계적인 산책을 돕는 어플

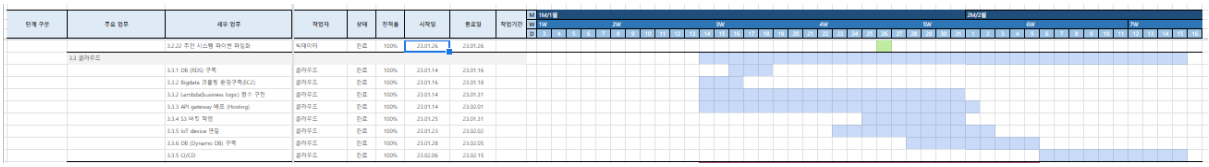
---

2. 구현 목표

AWS에서 제공하는 RDS를 이용해 빅데이터와의 협업점을 이루고, IoT Core를 이용하여 IoT와의 협업점을 이룬다. AWS에서 제공하는 FaaS 플랫폼인 Lambda를 이용해 Serverless 인프라를 구성하고 앱 백엔드를 구현한다. 인프라 구성 후 API Gateway를 이용해 앱 호스팅을 진행하고, Github와 AWS Codepipeline을 이용해 CI/CD 파이프라인을 구축한다.

---

3. 개발 일정

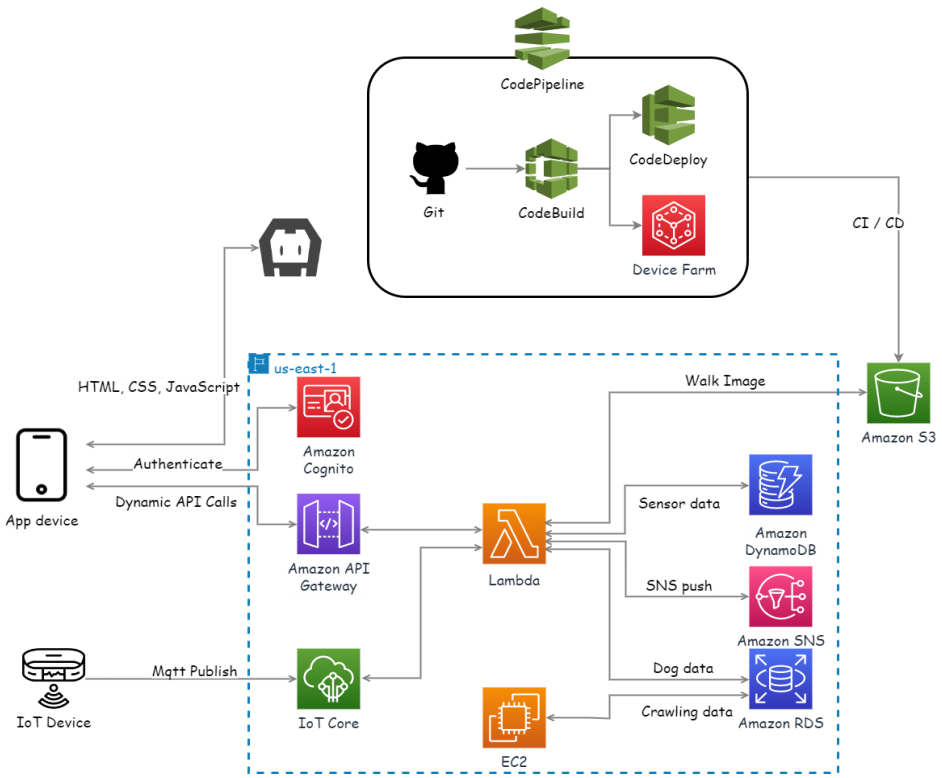


(WBS 일부)

https://docs.google.com/spreadsheets/d/16PRsAfC7GaPmUF5p4Qa8G7brk0si4IjC/edit#gid=1021512975

구현

1. 아키텍처 설계도



---

2. Infra

1. Computing Options

- Computing options이란 관리해야하는 on-premis 인프라에 따라 순서대로 on-site, IaaS, PaaS, SaaS로 나뉜다. IaaS는 클라우드나 인터넷을 통해 필요에 따라 제 3자가 Virtualization, Storage와 같은 서비스를 제공하는 종량제 서비스로 클라우드 컴퓨팅 제공업체인 AWS, Azure 등이 이에 해당한다. PaaS는 공급자가 자체 인프라에서 하드웨어와 소프트웨어를 호스팅하고 인터넷 연결을 통해 통합 솔루션, 솔루션 스택, 솔루션 서비스 등을 사용자에게 제공한다. FaaS는 PaaS에서 변형된 형태로 수요에 따라 동적으로 작동한다는 차이점이 있다. PaaS와 FaaS는 인프라 구축 및 관리에 대해 신경쓰지 않아도 된다는 장점이 있으나, 커스터마이징 면에서 단점이 존재한다. SaaS는 클라우드 어플리케이션 서비스라고도 불리며 PaaS보다도 자유도가 떨어진다는 단점이 존재한다. 대표적으로 Gmail, office365 등이 해당한다.

2. **Serverless**

- Serverless란, 개발자가 따로 서버 환경을 만들거나 관리할 필요 없이 애플리케이션을 빌드하고 실행할 수 있도록 하는 클라우드 네이티브 개발 모델이다. 일반적으로 BaaS와 FaaS로 나뉘고, 해당 프로젝트에서 주로 사용된 AWS Lambda는 FaaS에 해당한다. Lambda에서 작성된 코드는 AWS의 컨테이너에 배포되어 작동하고, Stateless, One-off, Triggered on events, Fully managed의 4가지 특징을 갖는다. 호스팅을 진행할 때는 API 호출을 이용하고, AWS에서는 API Gateway를 이용해 진행한다.

---

3. 기능

1. **Cognito** (송강현, 구정민)

- 기능 : 사용자 인증(회원가입,로그인)

- 채택이유 : 높은 보안성 , 간편한 로그인 구현 가능 , 서버리스환경과 연동하기 용이

2. **IoT Core** (안민규, 송강현)

- 기능 : IOT 데이터들을 MQTT 통신을 통하여 센서값 수신

- 채택이유 : 데이터가 들어올 때만 MQTT통신이 연결되어 상시환경 구성이 가능 , 라우팅 툴을 이용하여 센서 데이터 간편한 처리 가능

3. **RDS (MySQL)** (안민규, 구정민)

- 기능 : 강아지 정보데이터, 산책데이터, 날씨 크롤링 데이터, 장소 데이터등을 저장

- 채택이유 : 정형화된 스키마가 있는 데이터들을 효과적으로 저장하기 위해서 SQL 기반의 RDS를 사용

4. **DynamoDB** (안민규, 구정민)

- 기능 : IoT 센서 데이터 저장

- 채택이유 : 데이터와 트래픽이 큰 (빅데이터 형태) 센서 데이터 특성 상 수평적 확장이 쉬운 NoSQL 기반의 DynamoDB가 적합. 또한 실시간으로 바뀌는 거리와 좌표를 보여주기에 빠른 리드속도가 필요하므로 DynamoDB 채택

5. **Lambda** (안민규, 송강현, 구정민, 조슬기)

   - 기능 : Serverless 로 백엔드 환경 구성

   - 채택이유 : 비용 최적화, 트래픽이나 보안 관리에 대한 리소스 절감 가능

6. **API Gateway** (구정민, 송강현)

   - 기능 : Lambda - frontend 간 request, response 전달

7. **EC2** (안민규, 송강현)

   - 기능 : 산책 데이터를 통해 산책 결과 이미지화 후 S3에 저장

   - 채택 이유 : Lambda에 등록해 사용할 수 있는 라이브러리 한계치보다 많은 라이브러리가 필요하여 Lambda 내에서 사용 불가

8. **S3** (조슬기)

   - 기능 : Local storage 대신 이미지 데이터를 저장하는 Cloud Storage로써 작동. 어플에서 업로드하거나 다운로드하는 이미지 및 CI/CD의 Deploy point로써 APK 파일 저장. 버킷의 엔드포인트를 이용해 앱 및 람다 통신

9. **CI / CD** (안민규, 송강현)

   - 기능 : AWS의 code pipeline을 이용해 CI/CD 구현. 깃허브에 액션 발생 시 자동으로 실행. 업로드 되어 있는 코드와 yaml 파일에 따라 build를 진행하여 apk 파일 생성 후 S3에 저장

   - 채택 이유 : Jenkins등의 다른 pipeline 툴보다 사용이 간편하고, 다른 AWS 서비스와 함께 사용하기 용이함

---

3. IoT, Bigdata - Cloud 협업점

IoT - Cloud : AWS IoT Core에서 생성한 사물 device의 CA, Private key, Certificate를 IoT 웨어러블 기기 코드에 작성하여 연결. gps 센서와 진동 센서로 구성되어 gps 센서는 위도, 경도, 시간 데이터를, 진동 센서는 진동 횟수 데이터를 수집함. 실시간으로 데이터가 센서에 입력될 때마다 데이터를 IoT Core에서 지정한 라우팅 룰에 따라 Lambda로 데이터 전송.

Bigdata - Cloud : 빅데이터 전공에서 작성해준 크롤링 코드를 람다로 재구현 후 주기적으로 크롤링이 진행되게끔 설정. 크롤링 진행결과는 RDS에 Insert. 추천을 위한 가중치 계산 모델링 코드 또한 람다로 구현하여 프로그램에 적용.

-> IoT 파트에서 제작한 웨어러블 기기를 통해 들어온 센서 데이터를 AWS 에서 처리, Bigdata 파트에서 수집한 날씨, 장소 정보 등을 AWS에서 처리 (다른 분야에서 수집된 정보들을 처리하거나, 가공할 수 있는 인프라를 구축)

---

4. Trial&Error&Sol

1. 산책 결과 이미지화

   1st Trial : Lambda 내에서 산책 경로 이미지 생성

   Error : 이미지 생성을 위해 필요한 라이브러리가 AWS Lambda에 등록 가능한 계층 수 초과 (*Lambda 당 최대 5개)

   2nd Trial : 람다에서 boto3 라이브러리를 통해 EC2를 부팅해서 EC2에서 산책 경로 지도 이미지를 생성하는 파이썬 파일을 실행하게 설정

   Error : 동작시간이 120초 이상 소요로 실사용에 있어 불편함을 초래

   Sol : SSM(Amazon System Manager)채택, '산책 종료' 이벤트가 발생하면, EC2에 command로 (이미지를 생성하는) 파이썬 파일 실행 명령을 내려 약 2초로 실행시간을 단축. (EC2 서버를 계속 켜두어서 부팅에 소요되는 시간 단축)

2. 날씨 크롤링

   1st Trial : 앱 상에서의 이벤트와 상관없이 진행되어야하므로 처음엔 EC2에 구현

   Error : EC2로 인한 과도 비용 발생

Sol : 주기적으로 이벤트를 발생시키는 기능인 aws event bridge를 이용해 시간마다 크롤링 진행되게 Lambda로 재구현

3. DynamoDB

   1st Trial : 센서 데이터를 저장하기 위한 목적으로 한 개의 table 사용

   Error : 데이터의 양이 많아지면서 리드 속도가 빠른 DynamoDB의 장점이 퇴색, 과도한 리소스 비용 발생

   Sol : 업데이트용과 빅데이터 저장용 테이블을 나누어 구현

4. CI/CD

   1st Trial : Android CI/CD를 AWS CodePipeline을 구현, GIT저장소에 코드를 push하면 AWS CodeBuild에서 코드를 빌드하여 APK 파일을 S3에 생성, AWS CodeDeploy를 통하여 S3에 APK파일을 배포

   Error : AWS CodeBuild에서 제공하는 기본도커환경이 진행한 프로젝트 구성한 환경과 달라 에러 발생(Android Sdk version)

   Sol : Build할 때 Sdk버전을 업그레이드 하도록 yml파일을 수정

고찰

1. **<u>Strong Point</u>**

- 클라우드 관점

1. 비용 save (호출당 비용을 부과한다는 점에서 리소스와 비용을 최적화) : 45일의 프로젝트 기간동안 120달러 발생 <-> 타팀(3팀) 3500달러 발생
2. 기능 단위로 신속하게 개발 후 배포 가능
3. 서버리스를 활용하면 운영 체제 및 파일 시스템 관리, 보안 패치, 부하 분산, 용량 관리, 스케일링, 로깅, 모니터링과 같은 일상적인 태스크를 모두 클라우드 서비스 제공업체에 이관할 수 있음.

- 어플 전체에서의 관점

1. 산책에 도움되는 실시간 날씨 정보, 쓰레기통&공중화장실&애견동반카페 등 장소정보 마커 형태로 제공 - 기존 어플을 자연어 분석한 결과를 바탕으로 기존 어플에는 부족한 기능을 해당 어플에서 추가, 견주가 더 편리하고, 쉽게 산책 할 수 있도록 함.
2. 애견카페&애견동반카페 추천 시스템
3. 견종별 산책 가이드라인 제공(하루 동안 강아지가 산책한 횟수, 거리, 시간과 견종별 적절한 산책 횟수, 거리, 시간을 비교해 산책이 얼마나 적절한지 표시) 을 통해 체계적으로 반려견 건강 관리가 가능하도록 함. + 주별, 월별, 연별 데이터도 제공

---

2. Weak Point

1. 현재 람다 서비스에서의 계층 수 제한, 용량 제한 등으로 인하여 일부 기능에서 EC2 채택 (서버를 직접 구성하는 것이 아니므로, AWS에서 지원하는 서비스의 제한을 극복할 수는 X, 향후 FaaS 기능의 지원범위가 커지면 업그레이드 가능)
2. 현재는 센서 데이터 저장을 위해 업데이트용과 전체 저장용 테이블을 따로 구성.

---

3. Develop Point

1. Apache kafka 스터디 진행(하나의 테이블에서 필요한 데이터만 따로 레코드로 지정하여 사용 가능, 데이터의 수명주기를 따로 설정해 필요없는 데이터를 보관하지 않도록 - 업데이트용 테이블을 따로 둘 필요 X)
2. AWS SNS - 현재는 사용자가 직접 어플에서 산책 가이드라인이나 실시간 날씨를 확인해야 함. 푸시 알림 기능을 구현해 특정 시간이나 상황에서 산책 가이드라인을 제공받을 수 있도록 구현.

참고 자료

"서버리스란?", https://www.redhat.com/ko/topics/cloud-native-apps/what-is-serverless

"클라우드 패러다임의 전환: 서버리스 컴퓨팅",
https://www.samsungsds.com/kr/insights/1232763_4627.html

"AWS IoT Core", https://docs.aws.amazon.com/ko_kr/iot/latest/developerguide/what-is-aws-iot.html

"CI/CD란?", https://www.redhat.com/ko/topics/devops/what-is-ci-cd