헬라클레스 클라우드 수행 보고서

헬라클레스 프로젝트의 클라우드 파트에서는 전체적인 서버의 흐름을 설계하고 구성하였다.

1. 설계

설계 단계에서는 그 동안 멀티캠퍼스를 수강하며 배운 내용을 적극 활용하고자 AWS 아키텍쳐를 중점으로 설계 하였다.

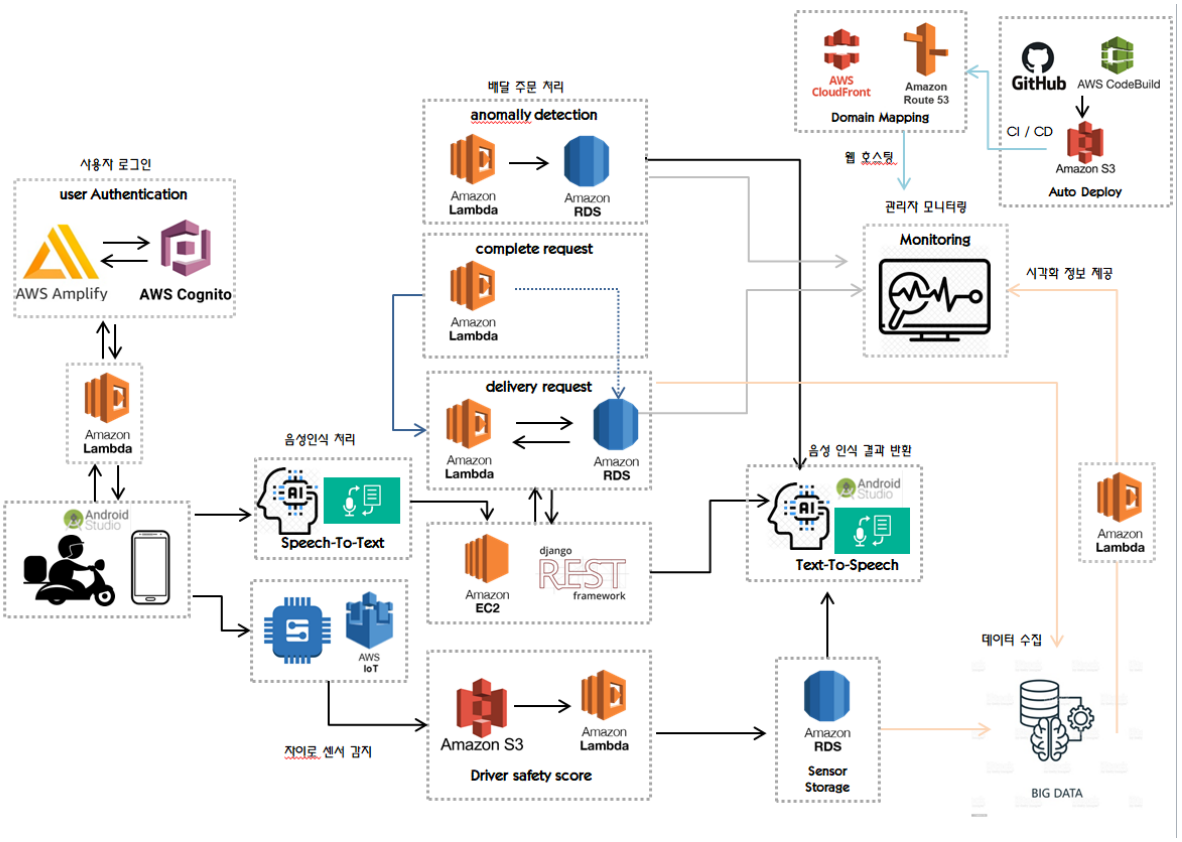


그림1. 클라우드 아키텍쳐 구성도

2. 설계상의 문제와 해결법

2-1. AI 학습 모델 운용

AI의 음성인식 학습 모델을 실행하기 위해서는 tensorflow를 활용하여야 했는데, tensorflow가 요구하는 gpu의 특성상 AWS Lambda에서 서버리스로 실행하기에는 메모리 문제가 있었다.

따라서 AWS EC2 Instance를 구성하고 Django를 설치하여 백엔드 서버를 운용하는 것으로 설계하였다.

2-2 IOT Core의 적용 여부

자이로 센서를 통한 실시간 위험감지 알람을 구현하고자 설계할 당시 라즈베리파이 - 서버 간 전송 방식으로 IOT Core를 사용할 계획이었다.

그러나 실시간 배달의 특성상 네트워크 환경이 단절된 환경에서도 위험감지 알람을 보내주어야 했기 때문에 자이로 센서와 안드로이드 간 블루투스 연결을 통해 알림을 보내주는 것으로 대체 하였다.
이후 안드로이드 애플리케이션 내부에서 위험 감지 횟수를 카운팅하여 Amazon RDS로 전송하였다.

3. 구현

3-1 회원가입, 로그인 (인증)

최초 안드로이드 애플리케이션을 구동할 시 회원가입을 통해 로그인한 유저만 서비스를 사용할 수 있도록 구현하였다.
이 과정에서 회원가입, 로그인의 프로세스는 Amazon에서 제공하는 Amplify와 Cognito를 활용하였다.
Cognito가 멀티 팩터 인증과 저장 데이터의 암호화를 자동으로 지원해주기 때문에 보다 안전한 인증 기능을 구현할 수 있었고, 자격 증명 공급자 연동을 통해 쉽게 앱의 액세스 제어 기능을 추가할 수 있었다.

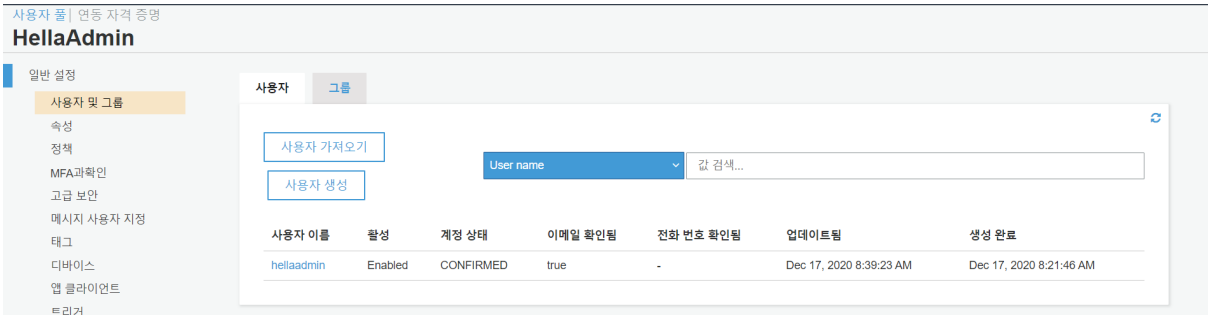


그림2. Cognito User Pool

3-2 클라이언트 - 서버 간 통신 (API)

안드로이드 애플리케이션 (클라이언트) 와 서버 간의 통신은 두 가지 방식을 활용하였다.

1. Amazon EC2 Instance를 구성하여 Django 서버를 설치하고 배포 후, DRF (Django Rest Framework) 를 통한 API 통신
2. AWS Lambda와 API-Gateway를 활용한 서버리스 API 통신

1의 경우 먼저 애플리케이션에서 음성 녹음을 처리할 시에 S3 버킷에 녹음된 wav 파일을 저장한다.
이후 안드로이드의 Retrofit 라이브러리를 통해 DRF API를 호출한다.
호출된 API 함수에서는 python의 라이브러리인 boto3를 활용하여 저장된 S3 버킷을 참조하여 wav 파일을 가져와 음성 분석을 시작한다.

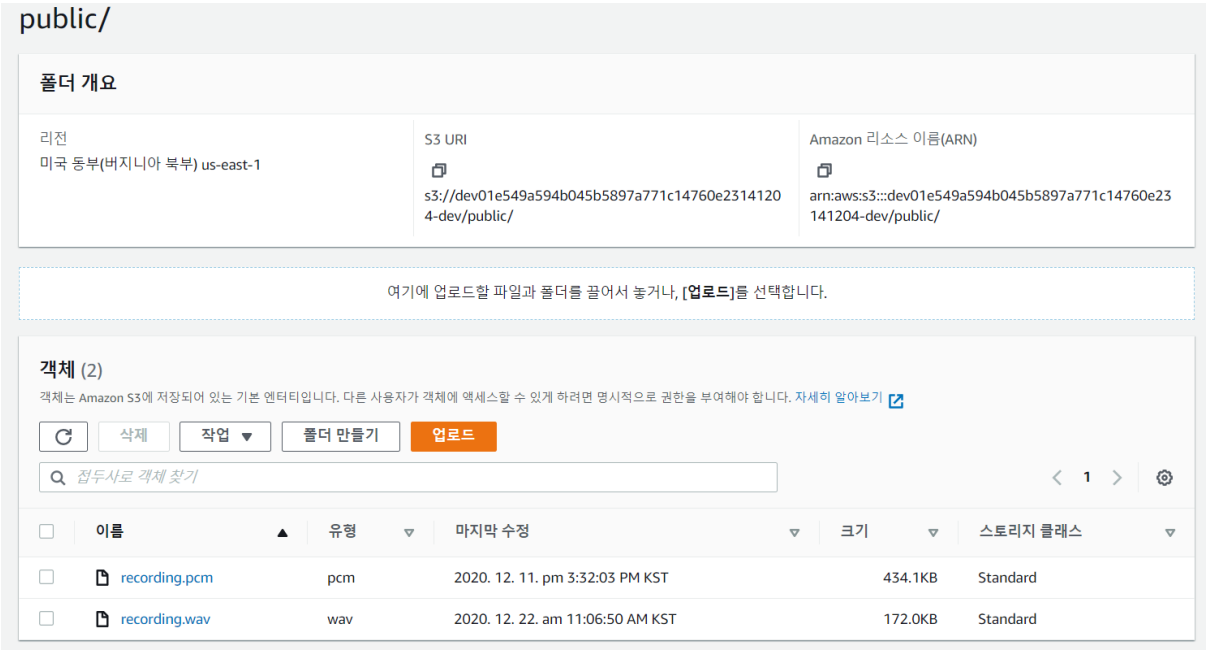


그림3. wav파일이 저장되는S3 버킷

음성 분석이 완료되면 매칭된 intent 값과 처리 결과를 반환하고, Django ORM (Object-Relational Mapping) 을 통해 Amazon RDS를 참조하여 배달 현황 DB의 상태 값을 변경한다.

2의 경우는 안드로이드 로그인 시에 Amazon RDS를 참조하여 사용자의 위,경도를 기반으로 1km 이내의 신규 주문건을 반환하기 위해 AWS Lambda를 통한 서버리스 통신을 구현하였다.

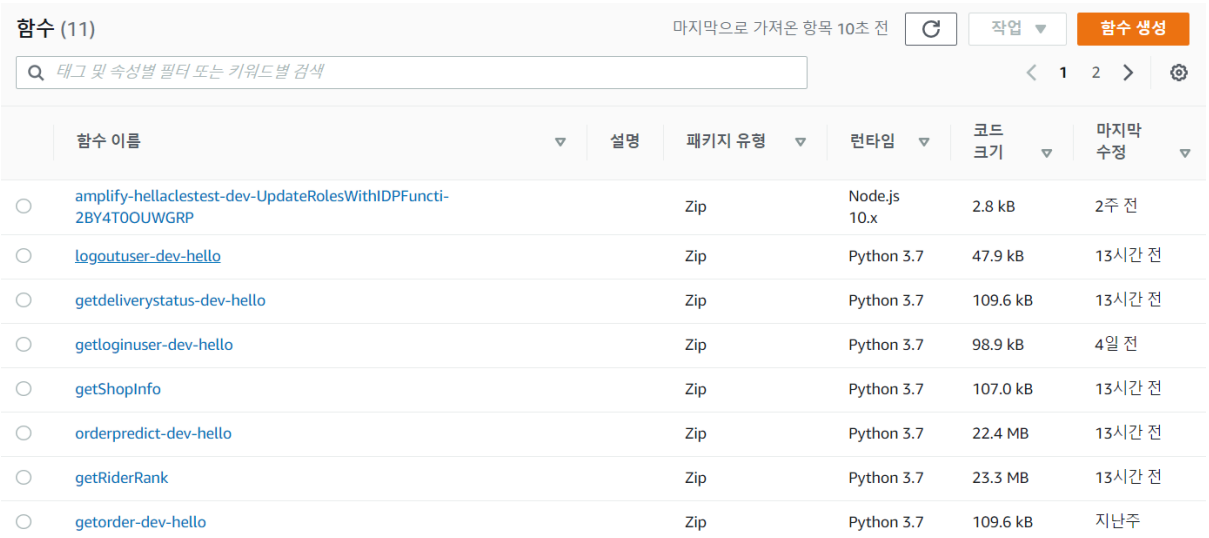
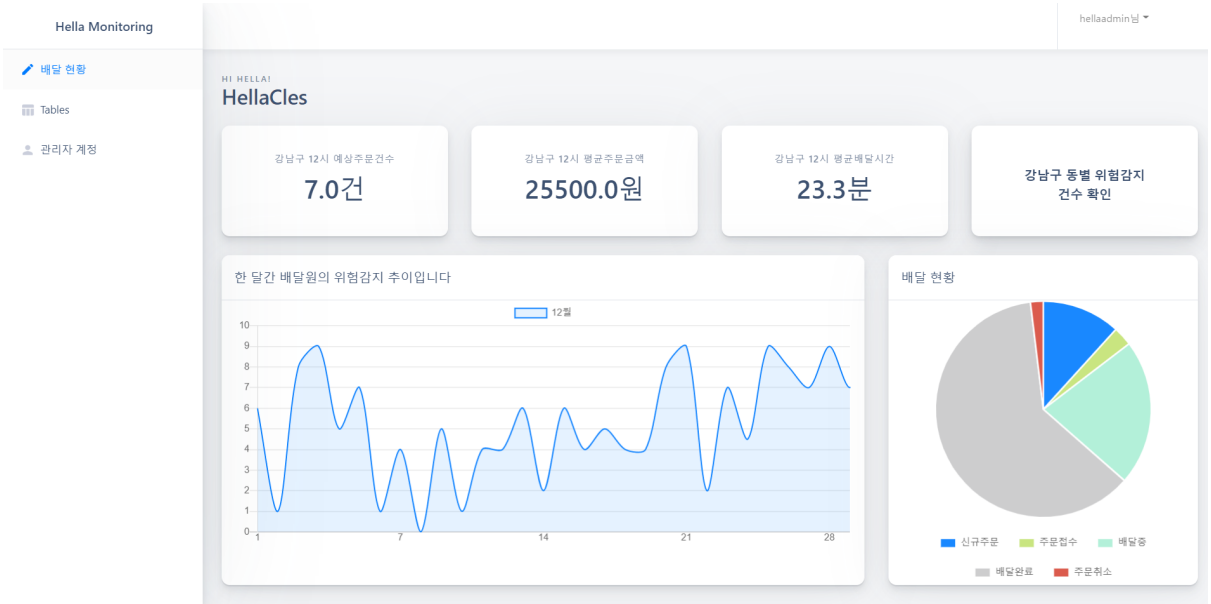
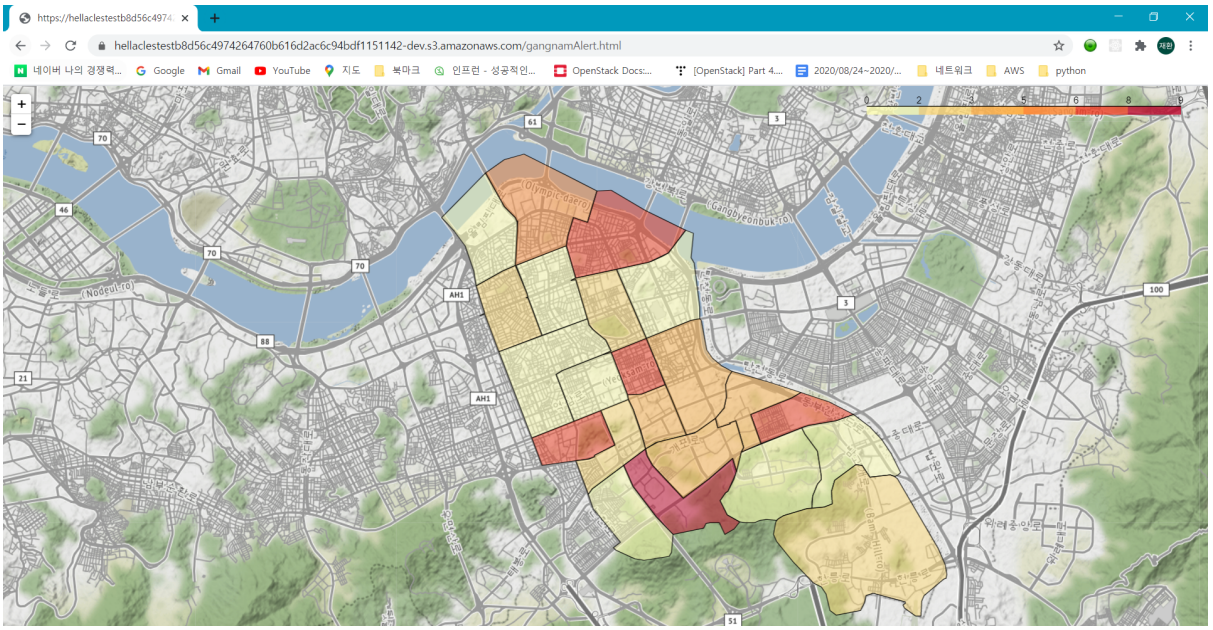


그림4. Aws Lambda 함수 리스트

3-3 모니터링 페이지 구현
헬라클레스의 관리자 모니터링 페이지는 실시간 배달 현황, 한 달간 배달원의 위험감지 추이,
구역 별 예상 주문 건수, 평균 주문 금액, 평균 배달 시간, 위험감지 건수에 대한 지도시각화, 그리고 라이더 평가 정보를 확인할 수 있다.


Hella Monitoring
hellaadmin님
배달 현황
Tables
관리자 계정
HI HELLA!
HellaCles
강남구 12시 예상주문건수
7.0건
강남구 12시 평균주문금액
25500.0원
강남구 12시 평균배달시간
23.3분
강남구 동별 위험감지
건수 확인
한 달간 배달원의 위험감지 추이입니다
12월
배달 현황
신규주문
주문접수
배달중
배달완료
주문취소


https://hellaclestestb8d56c4974
hellaclestestb8d56c4974264760b616d2ac6c94bdf1151142-dev.s3.amazonaws.com/gangnamAlert.html
네이버 나의 경쟁력...
Google
Gmail
YouTube
지도
북마크
인프런 - 성공적인...
OpenStack Docs:...
[OpenStack] Part 4...
2020/08/24~2020/...
네트워크
AWS
python

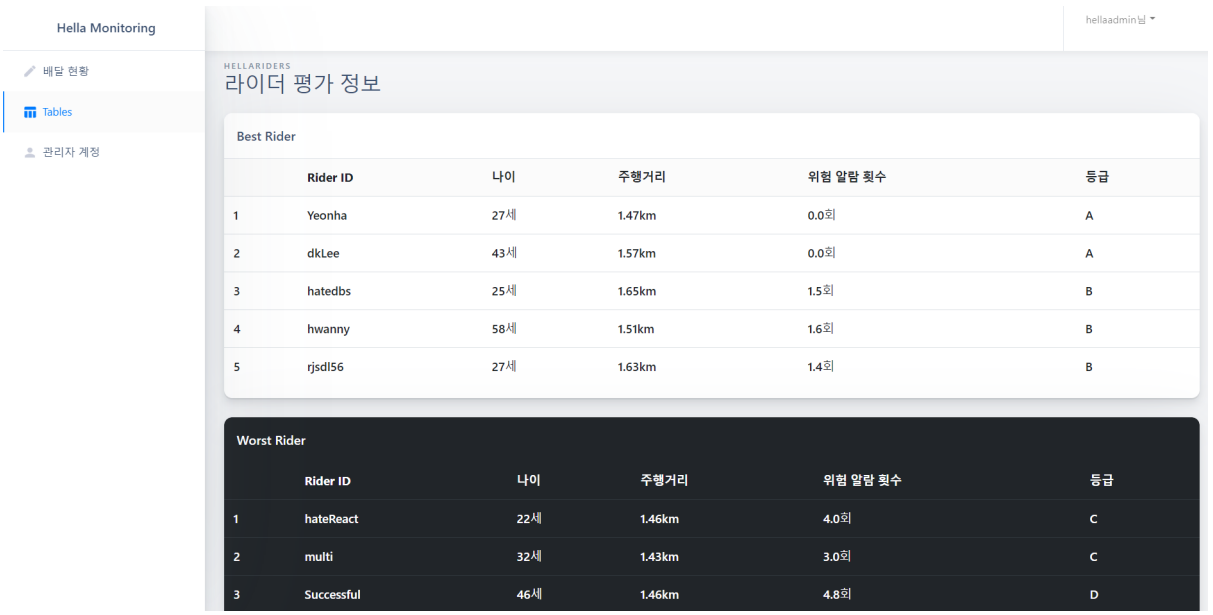


그림 5,6,7 헬라클레스 관리자 모니터링 페이지

페이지 스크립트 언어는 React.js를 활용하여 구현하였다.
각 기능에서 DB를 통해 조회하는 부분은 전부 AWS Lambda를 활용한 서버리스 API 통신을 이용하였으며, React의 axios 호출을 통해 클라이언트 - 백엔드 간 비동기 API 호출 방식을 구현할 수 있었다.

빅데이터 분석팀에서 Pandas를 활용해 배달 주문 현황 데이터베이스 시트를 참조해 분석한 코드를 AWS Lambda에 업로드 하였으며, 이를 통해 페이지 별 실시간 시각화가 가능했다.
모니터링 페이지는 Amazon CodePipeLine을 이용해 Github에 push할 때 마다 자동으로 S3 버킷에 업로드 되도록 구성하였으며, S3 버킷에서는 정적 웹사이트 호스팅을 통해 페이지를 호스팅 하였다.

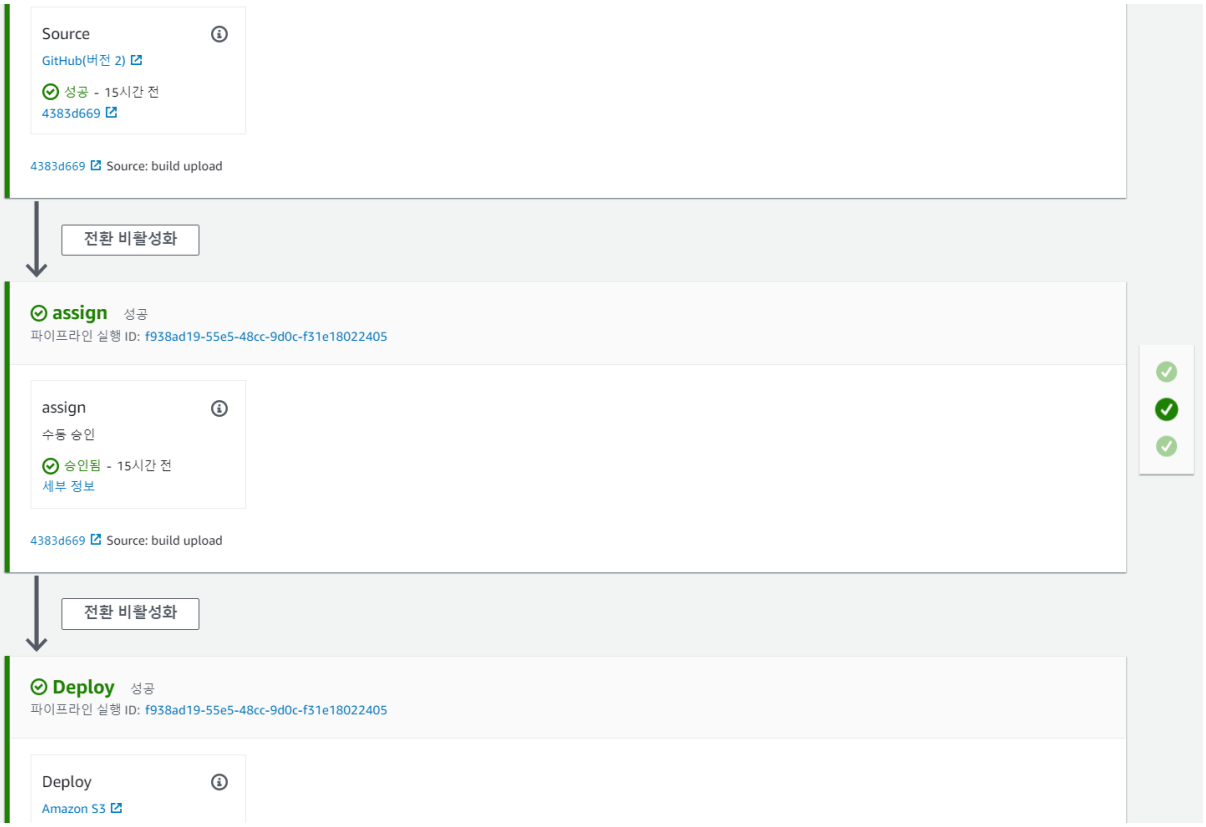


그림 8. Code PipeLine을 통한 자동 배포

보다 안전한 보안을 위해 Amazon에서 제공하는 CloudFront사용하였고, HTTP에서 HTTPS로 리다이렉팅 작업을 해주었다. 이 과정에서 필요한 SSL 인증서는 ACM (Amazon Certificate Manager)를 활용해 발급받을 수 있었다.
다음으로 가비아에서 실제 도메인을 구입하여 Amazon Route53을 이용해 도메인을 매핑하고 실제 서비스로 배포하였다.

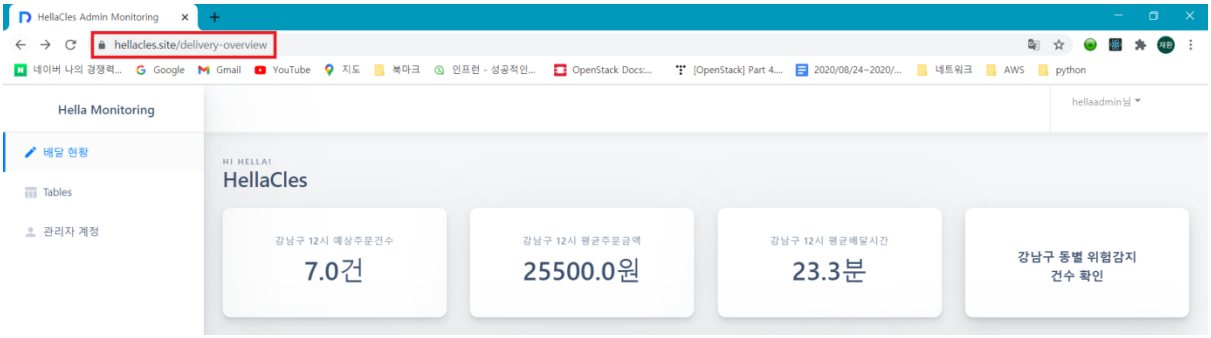


그림 9. Https 도메인 매핑

실제 작성한 API 리스트는 다음과 같다.

idx	Method	API명칭	EndPoint	Request	Response_Success	Define
1	POST	GetLoginUser	https://64mg85mq2c.execute-api.us-east-1.amazonaws.com/hella/login	userid: String	{ "statusCode": "200", "body": "login Success" } { "statusCode": "404", "body": "error" }	Hellacles 사용자 로그인
2	POST	GetLoutoutUser	https://64mg85mq2c.execute-api.us-east-1.amazonaws.com/hella/logout	userid String	{ "statusCode": "200", "body": "logout Success" } { "statusCode": "404", "body": "error" }	Hellacles 사용자 로그아웃
3	GET	GetDeliveryStatus	https://64mg85mq2c.execute-api.us-east-1.amazonaws.com/hella/status	-	{ "statusCode": "200", "status": "status_code" } { "statusCode": "404", "body": "error" }	배달 현황 조회
4	GET	OrderPredict	https://64mg85mq2c.execute-api.us-east-1.amazonaws.com/hella/order	-	{ "statusCode": 200, "predict_order": "predict_order", "pay_avg": "pay_avg", "time_expect": "time_expect" } { "statusCode": "404", "body": "error" }	시간대별 주문 예측 조회
5	GET	getRiderRank	https://64mg85mq2c.execute-api.us-east-1.amazonaws.com/hella/rank	-	{ "statusCode": "200", "body": "rider_rank_list" } { "statusCode": "404", "body": "error" }	배달원 평가 리스트 조회
6	POST	SpeechToText	http://ec2-15-165-62-176.ap-northeast-2.compute.amazonaws.com/SpeechToText	user	{ "statusCode": 200, "body": "intent", "intent_text": "intent_text", "shopName": "shop_name", "destination": "destination" } { "statusCode": "404", "body": "error" }	STT 분석 결과 반환
7	POST	getShopInfo	https://64mg85mq2c.execute-api.us-east-1.amazonaws.com/hella/shopinfo	latitude: Float, longitude: Float	{ "statusCode": "200", "body": "ShopInfoList" } { "statusCode": "404", "body": "error" }	신규 배달 조회

그림 10. 작성 API 명세서

4. 개선점

4-1 도커, 쿠버네티스를 활용한 서버의 컨테이너 운용

이번 멀티캠퍼스 과정에서 배운 강의 내용 중 유일하게 융복합 프로젝트에 적용하지 못했던 부분이 도커, 쿠버네티스를 활용한 컨테이너 배포였다.
비록 짧은 프로젝트의 시간 관계상 구현하지 못했지만, 프로젝트 발표 종료 이후에도 지속적인 관리를 통해 서버를 컨테이너로 배포하고 MSA 형식으로 관리할 계획이다.

4-2 소켓통신을 활용한 음성 파일 전송 속도 최적화

안드로이드 애플리케이션에서 서버로 음성 파일을 전송할 때 현재 구현한 프로세스는 S3 버킷을 활용해 wav 파일을 업로드 하고 서버에서 참조하는 형식이었다.
그러나 중간에 S3 버킷을 경유하지 않고 안드로이드 - 서버 간 소켓 통신을 활용한다면 보다 빠른 속도로 음성 파일의 전달이 가능할 것이다.