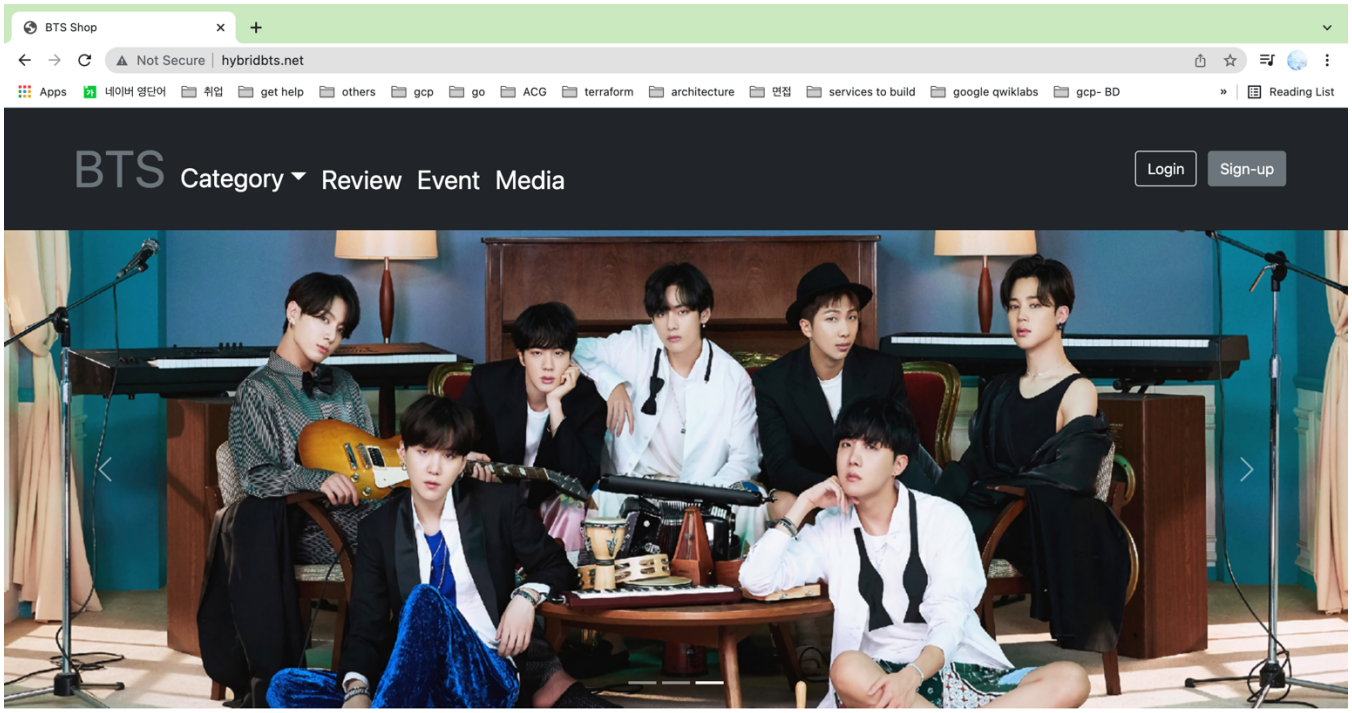
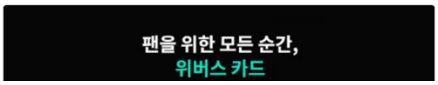
GCP Personal Project Overview

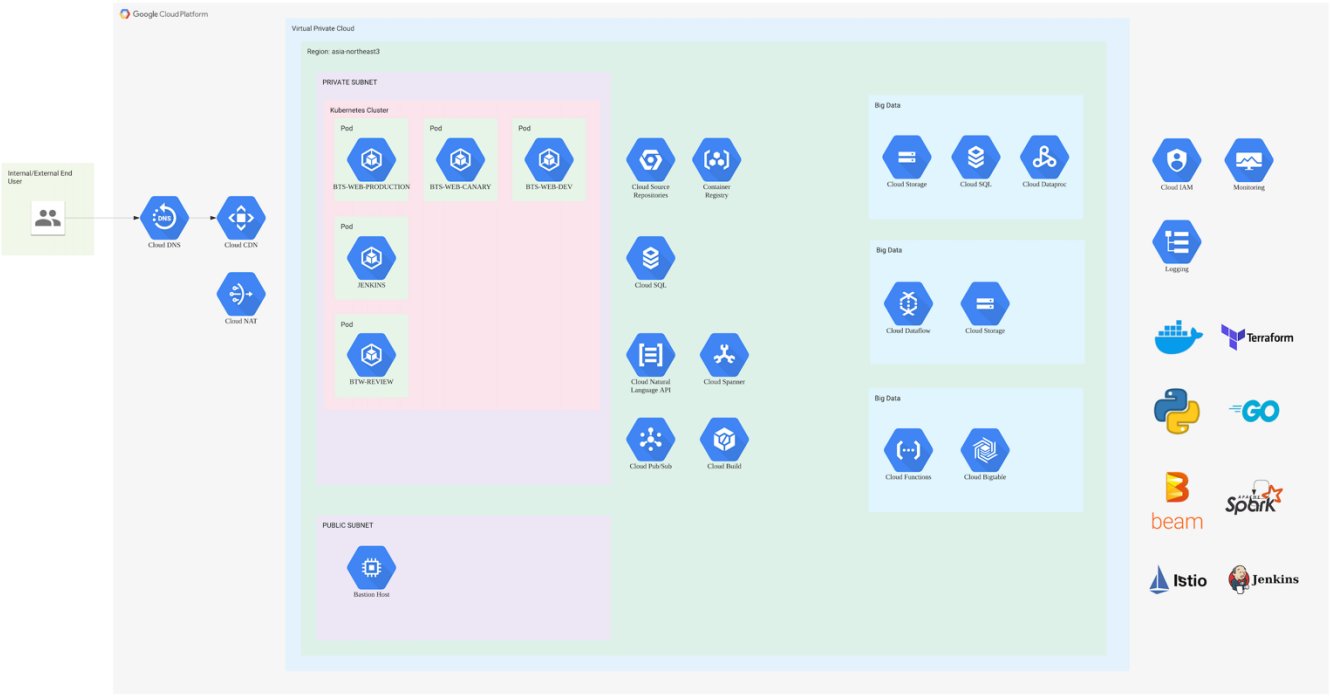
Scenario





쇼핑몰 HYBRID-BTS 의 인프라를 GCP 클라우드 환경 내에 구축했습니다. HYBRID-BTS 는 BTS 굿즈 상품을 판매하는 쇼핑몰입니다. 음반을 비롯해서 사진, 의류, 생활용품, 동영상 등 방탄소년단 관련 상품을 판매합니다. 한국을 비롯한 전세계 곳곳의 방탄소년단 팬들이 주요 고객층입니다. AWS 팀 프로젝트의 시나리오를 차용했습니다.

Architecture



HYBRID-BTS의 모든 인프라를 GCP 클라우드 내에 배치했습니다. Asia-northeast3 리전에 bts-vpc를 생성하고, public subnet과 private subnet을 생성했습니다. 메인 웹 서버인 bts-web 컨테이너를 private Kubernetes cluster에 배포하고, 상품 리뷰 서버인 bts-review와 CI/CD를 위한 Jenkins를 클러스터 내에 배포했습니다.

웹 서버에서 발생되는 데이터를 Cloud Storage, Cloud SQL, Cloud Spanner, Cloud Bigtable, Cloud Dataflow, Cloud Pub/Sub, Cloud Function 등을 사용해 변환하고 저장과 불러오기 하였습니다.

Infrastructure build using CLI/Google Cloud SDK

Networking	Custom VPC를 생성하여 public subnet과 private subnet을 생성했습니다. HTTP, HTTPS, SSH, ICMP를 허용하는 방화벽을 생성했습니다. NAT 라우터와 NAT 게이트웨이를 생성하여 VPC의 프라이빗 인스턴스들이 인터넷에 접근 가능하도록 했습니다.
Compute Engine	관리자가 프라이빗 GKE 클러스터 내의 인스턴스들을 관리할 수 있도록 bastion host를 public subnet에 생성하였습니다.
Cloud SQL	웹사이트의 회원정보를 저장하기 위해 Cloud SQL 인스턴스와 데이터베이스, 테이블을 생성했습니다.
Cloud Storage	웹사이트의 static contents들을 저장할 버킷을 생성하고 오브젝트에 ACL을 설정하였습니다.
Cloud DNS	웹 서버에 hybridbts.tech 도메인을 연결하였습니다.
Cloud CDN	웹 서버에 CDN을 적용하여 글로벌 엔드 유저에게 빠른 컨텐츠 딜리버리가 가능하도록 했습니다.
Google Kubernetes Engine	Hybridbts.tech 웹 서버 애플리케이션을 컨테이너로 GKE에 배포하고 운영했습니다. 프라이빗 클러스터로 생성하여 ingress를 통해서만 외부에서 접근이 가능하도록 했습니다. 클러스터 내에 Jenkins 앱을 배포해서 CI/CD pipeline을 생성하고, review app을 배포하여 사용자들로부터 리뷰를 받고 머신러닝을 통해 새로운 데이터를 추출했습니다.
CI/CD pipeline	Jenkins를 사용하여 CSR의 소스코드가 업데이트 되면 자동으로 클러스터에 배포가 이루어지도록 CI/CD pipeline을 생성했습니다. Dev environment를 사용하여 소스코드 수정을 하고, Canary

	environment를 사용하여 카나리 배포로 테스트 후에 Production environment에 최종적으로 업데이트가 됩니다.
Cloud Source Repository	애플리케이션 소스 코드를 CRS에 업로드하여 GKE, Cloud Build, Cloud Run 등 GCP 내 서비스와 연동이 쉽게 했습니다.
Google Container Registry	Cloud Build나 Jenkins CI/CD로 생성한 컨테이너 이미지를 저장하고 Kubernetes에 애플리케이션이 배포 되도록 했습니다.
Cloud Build	Cloud Build로 애플리케이션 배포의 CI/CD를 했습니다.
IAM	리소스들이 다른 리소스에 대한 제어 권한을 가질 수 있도록 service account를 생성했습니다.
Big data-SQL, Google Cloud Storage, Dataproc, PySpark, Python	Products에 대한 고객의 ratings 데이터셋을 기반으로 머신러닝 모델을 적용하여 사용자에게 적합한 Recommendation 데이터셋을 생성합니다. GCS에 있는 csv 파일을 SQL에 옮기고, Dataproc을 이용하여 Data processing을 합니다. Python 스크립트로 SQL 데이터를 불러와 PySpark 머신러닝 모델을 train하고 적용한 데이터셋을 생성하고 SQL에 저장합니다.
Big data- Dataflow, Goolgle Cloud Storage, Apache Beam, Python	고객의 주문 정보 데이터셋에서 거주 도시를 필터로 데이터를 추출하였습니다. Google Cloud Storage에 저장된 데이터셋을 불러와 Apache Beam으로 추출한 데이터를 Google Cloud Storage에 저장하는 Python 스크립트를 Dataflow job으로 실행했습니다.
Big data- Cloud Bigtable, Cloud Function	고객의 거주 지역에 대한 데이터셋에서 거주 도시가 Paris인 데이터를 반환합니다. Python script로 Bigtable에 데이터를 삽입한 뒤, Cloud Function의 HTTP trigger를 사용해서 추출된 데이터를 얻습니다.

Infrastructure build using Terraform

Networking	동일
Compute Engine	동일
Cloud SQL	동일
Cloud Storage	동일
Cloud DNS	동일
IAM	동일
Google Kubernetes Engine	동일

Web application: Review App

상품에 대한 고객의 리뷰 데이터에 머신러닝 모델을 적용해 sentiment score 데이터를 생성하여 저장하는 웹 애플리케이션입니다. 고객이 리뷰를 남기면 Pub/Sub 을 사용해 메인 애플리케이션이 Publish한 메세지를 워커 애플리케이션이 Subscribe합니다. 구독한 메세지에 Natural Language API를 적용하여 sentiment score를 추출하고, 생성된 데이터는 Cloud Spanner 에 저장됩니다.

Python	애플리케이션의 back-end를 생성하고 GCP 서비스에 연결합니다.
Flask	애플리케이션의 routes를 생성하고, HTTP requests와 responses를 받습니다.
JavaScript, AngularJS	Angular module을 이용하여 동적 페이지를 생성합니다. 사용자가 /review에 접근하면 HTTP POST request에 따라 화면을 띄우고, 받은 데이터를 반환합니다.
HTML	애플리케이션의 front-end를 생성합니다.
Container	tiangolo/uwsgi-nginx-flask:python3.7 베이스로 이미지를 생성하여 Nginx 웹 서버가 UWSGI를 통해 Flask를 해석할 수 있습니다.