

GCP INFRASTRUCTURE BUILD

Table of Contents

hybridbts.net	2
Virtual Private Cloud (VPC).....	3
Firewall	4
Compute Engine.....	5
Google Cloud Storage(GCS)	6
Cloud SQL	7
IAM- Service Account	9
Cloud DNS.....	10
Cloud Source Repository(CSR).....	11
Cloud Build.....	12
Google Kubernetes Engine(GKE).....	14
CI/CD pipeline(Jenkins).....	25
SQL, Google Cloud Storage, Dataproc, PySpark, Python	33
Dataflow, Apache Beam, Google Cloud Storage, Python	40
Cloud Bigtable, Cloud Function.....	43

hybridbts.net

BTS Category ▾ Review Event Media

Login Sign-up

MAP OF THE SOUL: ONE

팬을 위한 모든 순간,
위버스 카드

Virtual Private Cloud (VPC)

Custom VPC 를 생성하고 public subnet 과 private subnet 을 생성했습니다. NAT router 와 NAT gateway 를 생성하여 VPC 내의 프라이빗 인스턴스들이 인터넷에 접근 가능하도록 했습니다.

Create a custom VPC

```
gcloud projects create hybrid-bts  
gcloud config set project hybrid-bts
```

```
gcloud compute networks create bts-vpc --subnet-mode=custom
```

```
gcloud compute networks subnets create bts-priv-sub-1 --network=bts-vpc \  
--region=asia-northeast3 --range=10.0.0.0/24
```

```
dntwkzz79@cloudshell:~ (hybrid-bts)$ gcloud compute networks list  
NAME: bts-vpc  
SUBNET_MODE: CUSTOM  
BGP_ROUTING_MODE: REGIONAL  
IPV4_RANGE:  
GATEWAY_IPV4:
```

Update

```
gcloud compute networks subnets update bts-priv-sub-1 \  
--region=asia-northeast3 --enable-private-ip-google-access
```

```
gcloud compute networks subnets delete private1 --region=asia-northeast3
```

List

```
gcloud compute regions list
```

```
gcloud compute networks list  
gcloud compute networks subnets list --network=bts-vpc  
gcloud compute firewall-rules list --network=bts-vpc  
gcloud compute networks subnets describe bts-priv-sub-1 --region=asia-northeast3 \  
--format="get(privateIpGoogleAccess)"
```

NAT gateway

```
gcloud compute routers create bts-nat-router \  
--network bts-vpc \  
--region asia-northeast3  
  
gcloud compute routers nats create bts-nat-config \  
--router-region asia-northeast3 \  
--router bts-nat-router \  
--auto-allocate-nat-external-ips \  
--nat-all-subnet-ip-ranges \  
--enable-logging
```

```
dntwkzz79@cloudshell:~ (hybrid-bts)$ gcloud compute routers list  
NAME: bts-nat-router  
REGION: asia-northeast3  
NETWORK: bts-vpc
```

Firewall

Custom VPC 에 배치된 인스턴스들에 HTTP, HTTPS, ICMP, SSH 접근을 개방하였습니다.

Create

```
gcloud compute firewall-rules create btsvpc-allow-http-https-icmp-ssh  
--direction=INGRESS \  
--priority=1000 --network=bts-vpc --action=ALLOW --rules=icmp,tcp:80,443,22 \  
--source-ranges=0.0.0.0/0
```

```
NAME: btsvpc-allow-icmp-ssh  
NETWORK: bts-vpc  
DIRECTION: INGRESS  
PRIORITY: 1000  
ALLOW: tcp:22,tcp:80,tcp:443,icmp  
DENY:  
DISABLED: False
```

Compute Engine

관리자가 프라이빗 리소스들을 관리할 수 있도록 bastion host 를 public subnet 에 생성하였습니다.

Create

```
gcloud compute instances create bts-bastion-vm --zone=asia-northeast3-a \  
--machine-type=e2-micro --subnet=bts-pub-sub-1 --image-family=debian-10 \  
--image-project=debian-cloud --boot-disk-size=10GB --boot-disk-type=pd-standard \  
--boot-disk-device-name=bts-bastion-vm
```

```
NAME: bts-bastion-vm  
ZONE: asia-northeast3-a  
MACHINE_TYPE: e2-micro  
PREEMPTIBLE:  
INTERNAL_IP: 10.0.0.2  
EXTERNAL_IP:  
STATUS: TERMINATED
```

Manage

```
gcloud compute instances start bts-bastion-vm --zone asia-northeast3-a  
gcloud compute instances stop bts-bastion-vm --zone asia-northeast3-a  
gcloud compute instances set-machine-type bts-bastion-vm --zone asia-northeast3-a \  
    --machine-type f1-micro  
gcloud compute instances list --sort-by=ZONE  
gcloud compute ssh bts-bastion-vm --zone asia-northeast3-a --tunnel-through-iap
```

Google Cloud Storage(GCS)

웹사이트의 동적 컨텐츠들을 저장할 버킷을 생성하고, 오브젝트에 ACL 을 설정하였습니다.

Create

```
gsutil mb -l asia-northeast3 gs://bts-static  
gsutil cp album.svg gs://bts-static/images  
gsutil cp gs://bts-static/images/album.svg .
```

```
NAME: gs://bts-static  
LOCATION: US  
STORAGE_CLASS: STANDARD
```

ACL

```
export BUCKET_NAME=bts-static

gsutil cp album.svg gs://$BUCKET_NAME/
gsutil acl get gs://$BUCKET_NAME/album.svg > acl.txt
cat acl.txt

gsutil acl set private gs://$BUCKET_NAME/album.svg
gsutil acl get gs://$BUCKET_NAME/album.svg > acl2.txt
cat acl2.txt

gsutil acl ch -u AllUsers:R gs://$BUCKET_NAME/album.svg
gsutil acl get gs://$BUCKET_NAME/album.svg > acl3.txt
cat acl3.txt
```

Cloud SQL

웹사이트의 회원정보를 저장하기 위해 Cloud SQL 인스턴스와 데이터베이스, 테이블을 생성했습니다.

Create

```
gcloud beta sql instances create bts-sql-1 --database-version=MYSQL_5_7 \
--tier=db-f1-micro \
--network=projects/hybrid-bts/global/networks/bts-vpc \
--zone=asia-northeast3-a --root-password=admin123 --no-assign-ip

gcloud sql instances list
```

```
gcloud sql instances delete bts-sql-1
```

```
NAME: bts-sql-1
DATABASE_VERSION: MYSQL_5_7
LOCATION: asia-northeast3-a
TIER: db-f1-micro
PRIMARY_ADDRESS: -
PRIVATE_ADDRESS: 10.58.112.4
STATUS: RUNNABLE
```

Connect

```
gcloud auth activate-service-account bastion-sa@hybrid-bts.iam.gserviceaccount.com \
--key-file=/home/dntwkzz79/credentials.json --project=hybrid-bts
```

```
gcloud sql connect bts-sql-1 --user=root
mysql -u root -h 10.58.112.4 -p
```

Create Database & Tables

```
create database bts;

CREATE TABLE board (
    num int primary key auto_increment,
    id char(15),
    name char(10),
    subject char(200),
    content text,
    regist_day char(20),
```

```
hit int,  
file_name char(40),  
file_type char(40),  
file_copied char(40)  
);  
  
CREATE TABLE members (  
num int primary key auto_increment,  
id char(15),  
pass char(15),  
name char(10),  
email char(80),  
regist_day char(20),  
level int,  
point int  
);
```

IAM- Service Account

리소스들이 다른 리소스에 대한 viewer 혹은 admin 권한을 가질 수 있도록 service account 를 생성했습니다.

```
gcloud iam service-accounts create bastion-sa \  
--display-name="bastion-sa"
```

```
gcloud iam service-accounts keys create credentials.json --iam-account=bastion-sa  
@hybrid-bts.iam.gserviceaccount.com
```

```
gcloud projects add-iam-policy-binding $PROJECT_ID \
--member="serviceAccount: bastion-sa@hybrid-bts.iam.gserviceaccount.com" \
--role="roles/container.admin"

gcloud auth activate-service-account bastion-sa@hybrid-bts.iam.gserviceaccount.com \
--key-file='./credentials.json' --project=hybrid-bts

kubectl create clusterrolebinding cluster-admin \
--clusterrole=cluster-admin --serviceaccount=default:bastion-sa

gcloud iam service-accounts list

displayName: vm-ser-acc
email: vm-ser-acc@hybrid-bts.iam.gserviceaccount.com
etag: MDEwMjE5MjA=
name: projects/hybrid-bts/serviceAccounts/vm-ser-acc@hybrid-bts.iam.gserviceaccount.com
oauth2ClientId: '101819087859158478884'
projectId: hybrid-bts
```

Cloud DNS

웹 서버에 hybridbts.net. 도메인을 연결하고, www.hybridbts.net.과 review.hybridbts.net. CNAME 레코드를 생성하였습니다.

Create DNS records

```
gcloud dns --project=hybrid-bts managed-zones create hybridbts \
--dns-name=hybridbts.net --description="For web server"
```

```
gcloud dns record-sets create hybridbts.net. \
--rrdatas="34.102.162.191" --type=A --ttl=60 \
--zone="hybridbts"

gcloud dns record-sets create www.hybridbts.net. --rrdatas="hybridbts.net." \
--type=CNAME --ttl=60 \
--zone="hybridbts"
```

```
dntwkzz79@cloudshell:~ (hybrid-bts)$ gcloud dns managed-zones list
NAME: hybridbts
DNS_NAME: hybridbts.tech.
DESCRIPTION:
VISIBILITY: public
```

Cloud Source Repository(CSR)

애플리케이션 소스 코드를 CSR에 업로드하여 GKE, Cloud Build, Cloud Run 등 GCP 내 서비스와 연동이 쉽게 됩니다.

Repository

```
gcloud source repos create bts-web

cd bts-web
git init
git config credential.helper gcloud.sh
export PROJECT_ID=$(gcloud config get-value project)
git remote add origin https://source.developers.google.com/p/$PROJECT_ID/r/bts-web
```

```
git config --global user.email "dntwkzz79@gmail.com"  
git config --global user.name "yeseul park"
```

```
git add .  
git commit -m "Initial commit"  
git push origin master
```

```
dntwkzz79@cloudshell:~ (hybrid-bts)$ gcloud source repos list  
REPO_NAME: bts-web  
PROJECT_ID: hybrid-bts  
URL: https://source.developers.google.com/p/hybrid-bts/r/bts-web
```

Cloud Build

Cloud Build로 review-app 배포의 CI/CD를 했습니다.

Build submit

```
cd bts-review  
docker pull tiangolo/uwsgi-nginx-flask  
export PROJECT_ID=$(gcloud config list --format 'value(core.project)')  
gcloud builds submit --tag gcr.io/$PROJECT_ID/bts-review
```

Build trigger

```
gcloud beta builds triggers create cloud-source-repositories \
--repo="bts_review_app" \
--branch-pattern="^master$" \
--build-config="trigger_build.yaml" \
--name="bts-review"
```

* 코드 수정

```
git status
git add .
git commit -m "updated"
git push origin master
```

```
dntwkzz79@cloudshell:~ (hybrid-bts)$ gcloud alpha builds triggers list
---
createTime: '2021-12-26T18:02:00.233566912Z'
filename: trigger_build.yaml
id: f64e9358-c250-460d-bc69-a0eb875aa03c
name: bts-review
triggerTemplate:
  branchName: ^master$
  projectId: hybrid-bts
  repoName: bts_review_app
```

Trigger_build.yaml

```
steps:
- name: 'gcr.io/cloud-builders/docker'
  args: ['build', '-t', 'gcr.io/hybrid-bts/bts-review', '.']
- name: 'gcr.io/cloud-builders/docker'
  args: ['push', 'gcr.io/hybrid-bts/bts-review']
```

Google Kubernetes Engine(GKE)

웹 서버 애플리케이션을 컨테이너로 GKE 에 배포하고 운영했습니다. 프라이빗 클러스터로 생성하여 ingress 를 통해서만 외부에서 접근이 가능하도록 했습니다.

Create clusters

```
export bts_network='bts-vpc'  
export bts_region='asia-northeast3'  
export bts_cluster='bts-cluster'  
  
gcloud container clusters create $bts_cluster \  
--network $bts_network --subnetwork bts-priv-sub-1 --region $bts_region \  
--machine-type e2-medium \  
--enable-autoscaling \  
--num-nodes 1 \  
--min-nodes 0 \  
--max-nodes 5 \  
--service-account cluster-admin@hybrid-bts.iam.gserviceaccount.com \  
--enable-ip-alias \  
--enable-private-nodes \  
--enable-private-endpoint \  
--master-ipv4-cidr 10.2.0.0/28 \  
--enable-master-global-access \  
--enable-master-authorized-networks  
  
gcloud container clusters update $bts_cluster --region $bts_region \  
--master-authorized-networks 10.0.0.2/32
```

```
gcloud container clusters resize $bts_cluster --region=$bts_region --num-nodes=0
```

```
NAME: bts-cluster
LOCATION: asia-northeast3
MASTER_VERSION: 1.21.5-gke.1302
MASTER_IP: 10.2.0.2
MACHINE_TYPE: e2-medium
NODE_VERSION: 1.21.5-gke.1302
NUM_NODES: 3
STATUS: RUNNING
```

Connect to clusters & configs

```
gcloud container clusters get-credentials $bts_cluster --region $bts_region
```

```
kubectl config view  
kubectl cluster-info  
kubectl config current-context  
kubectl config get-contexts  
kubectl top nodes
```

Create a sample job

```
Kubectl apply -f sample-job.yaml  
kubectl delete job sample-job  
kubectl describe jobs
```

sample-job.yaml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: sample-job
spec:
  template:
    metadata:
      name: sample-job
    spec:
      containers:
        - name: sample-container
          image: gcr.io/hybrid-bts/bts-web
      restartPolicy: OnFailure
```

Create secrets

```
kubectl create secret generic credentials-key \
  --from-file=$HOME/credentials.json --namespace=production
kubectl create secret generic credentials-key \
  --from-file=$HOME/credentials.json --namespace=default

kubectl delete secret generic credentials-key
kubectl get secrets
kubectl describe credentials-key
rm -rf ~/credentials.json
```

Create deployments

```
kubectl create ns production
kubectl --namespace=production apply -f k8s/production
kubectl --namespace=production apply -f k8s/canary
kubectl --namespace=production apply -f k8s/services

kubectl --namespace=production scale deployment bts-web-production --replicas=4

kubectl get deployments --namespace=production
kubectl get pods --namespace=production
kubectl get service bts-web-service --namespace=production

kubectl describe pod bts-web-canary --namespace=production
kubectl describe deployments --namespace=production
```

```
dntwkzz79@cloudshell:~ (hybrid-bts)$ kubectl get deployments --namespace=production
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
bts-web-canary   1/1     1            1           156m
bts-web-production   1/1     1            1           156m
```

k8s/bts-web-production.yaml

```
kind: Deployment
apiVersion: apps/v1
metadata:
  name: bts-web-production
  labels:
    app: bts-web
```

```
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bts-web
      role: web
      env: production
  template:
    metadata:
      name: web
      labels:
        app: bts-web
        role: web
        env: production
  spec:
    volumes:
      - name: google-cloud-key
        secret:
          secretName: credentials-key
    containers:
      - name: bts-web
        image: gcr.io/hybrid-bts/bts-web
    resources:
      limits:
        memory: "500Mi"
        cpu: "100m"
    imagePullPolicy: ifNotPresent
    volumeMounts:
      - name: google-cloud-key
        mountPath: /var/secrets/google
    env:
```

```
- name: GOOGLE_APPLICATION_CREDENTIALS  
  value: /var/secrets/google/key.json  
ports:  
- name: web  
  containerPort: 80
```

k8s/bts-web-canary.yaml

```
kind: Deployment  
apiVersion: apps/v1  
metadata:  
  name: bts-web-canary  
labels:  
  app: bts-web  
spec:  
  replicas: 1  
  selector:  
    matchLabels:  
      app: bts-web  
      role: web  
      env: canary  
  template:  
    metadata:  
      name: web  
      labels:  
        app: bts-web  
        role: web
```

```
env: canary
spec:
  volumes:
    - name: google-cloud-key
      secret:
        secretName: credentials-key
  containers:
    - name: bts-web
      image: gcr.io/hybrid-bts/bts-web
  resources:
    limits:
      memory: "500Mi"
      cpu: "100m"
    imagePullPolicy: ifNotPresent
  volumeMounts:
    - name: google-cloud-key
      mountPath: /var/secrets/google
  env:
    - name: GOOGLE_APPLICATION_CREDENTIALS
      value: /var/secrets/google/key.json
  ports:
    - name: web
      containerPort: 80
```

k8s/bts-web-dev.yaml

```
kind: Deployment
apiVersion: apps/v1
metadata:
```

```
name: bts-web-dev
labels:
  app: bts-web
spec:
  replicas: 1
  selector:
    matchLabels:
      app: bts-web
      role: web
      env: dev
  template:
    metadata:
      name: web
    labels:
      app: bts-web
      role: web
      env: dev
    spec:
      volumes:
        - name: google-cloud-key
          secret:
            secretName: credentials-key
      containers:
        - name: bts-web
          image: gcr.io/hybrid-bts/bts-web
      resources:
        limits:
          memory: "500Mi"
          cpu: "100m"
      imagePullPolicy: ifNotPresent
      volumeMounts:
```

```
- name: google-cloud-key
  mountPath: /var/secrets/google
env:
- name: GOOGLE_APPLICATION_CREDENTIALS
  value: /var/secrets/google/key.json
ports:
- name: web
  containerPort: 80
```

k8s/services/bts-web-service.yaml

```
apiVersion: v1
kind: Service
metadata:
  name: bts-web-service
annotations:
  networking.gke.io/load-balancer-type: "Internal"
labels:
  app: bts-web
spec:
  type: LoadBalancer
  selector:
    app: bts-web
  ports:
  - name: http
    protocol: TCP
    port: 80
    targetPort: 80
```

k8s/services/bts-ingress.yaml

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: bts-ingress
  annotations:
    kubernetes.io/ingress.global-static-ip-name: "bts-ingress"
spec:
  rules:
  - host: "review.hybridbts.net"
    http:
      paths:
      - path: /
        pathType: Prefix
      backend:
        service:
          name: bts-review-service
          port:
            number: 80
  - host: "*.hybridbts.net"
    http:
      paths:
      - path: /
        pathType: Prefix
      backend:
        service:
          name: bts-web-service
          port:
            number: 80
```

Deployment rollout & rollback

```
kubectl set image deployment bts-web-production \
bts-web=gcr.io/hybrid-bts/bts-web:latest --namespace=production

kubectl rollout status deployment/bts-web-production
kubectl rollout history deployment/bts-web-production

kubectl rollout undo deployments bts-web-production
kubectl rollout history deployment bts-web-production
```

Inject Istio sidecar

```
curl -L https://istio.io/downloadIstio | sh - && cd istio-1.12.1
export PATH=$PWD/bin:$PATH

istioctl install --set profile=demo -y
kubectl label namespace default istio-injection=enabled
kubectl label namespace production istio-injection=enabled
kubectl apply -f k8s/production --namespace=production
kubectl apply -f k8s/review-deployment.yaml
```

```
dntwkzz79@cloudshell:~/istio-1.12.1 (hybrid-bts)$ istioctl install --set profile=demo -y
✓ Istio core installed
✓ Istiod installed
✓ Ingress gateways installed
✓ Egress gateways installed
✓ Installation complete      Making this installation the default for injection and validation
```

CI/CD pipeline(Jenkins)

Jenkins 를 사용하여 CSR 의 소스코드가 업데이트 되면 자동으로 클러스터에 배포가 이루어지도록 CI/CD pipeline 을 생성했습니다. Dev environment 를 사용하여 개발을 하고 Canary environment 를 사용하여 카나리 배포로 테스트 후에 Production environment 에 최종적으로 업데이트가 됩니다.

Install Jenkins

```
kubectl create clusterrolebinding cluster-admin-binding \
--clusterrole=cluster-admin --user=$(gcloud config get-value account)

helm repo add jenkinsci https://charts.jenkins.io
helm repo update
helm install cd-jenkins -f jenkins/values.yaml jenkinsci/jenkins --wait
```

Connect to Jenkins

```
export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/component=jenkins-master" -l "app.kubernetes.io/instance=cd-
jenkins" -o jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:8080 >> /dev/null 2>&1 &
```

Create credentials

Kind

Google Service Account from private key

Project Name

hybrid-bts

JSON key

JSON key File

hybrid-bts-97...00b216.json

P12 key

Configure clouds

 Kubernetes

Name [?](#)

kubernetes

Kubernetes URL [?](#)

<https://kubernetes.default>

Jenkins URL [?](#)

Jenkins tunnel [?](#)

Create a multibranch pipeline job

Branch Sources

 **Git**

Project Repository [?](#)

Credentials [?](#)

Jenkinsfile

```
pipeline {  
  
    environment {
```

```

PROJECT = "hybrid-bts"
APP_NAME = "bts-web"
SVC_NAME = "${APP_NAME}-service"
CLUSTER = "bts-cluster"
CLUSTER_LOCATION = "asia-northeast3"
IMAGE_TAG =
"gcr.io/${PROJECT}/${APP_NAME}:${env.BRANCH_NAME}.${env.BUILD_NUMBER}"
JENKINS_CRED = "${PROJECT}"
}

agent {
  kubernetes {
    defaultContainer 'jnlp'
    yaml """
apiVersion: v1
kind: Pod
metadata:
  labels:
    component: ci
spec:
  serviceAccountName: 'kuber-sa'
  containers:
    - name: gcloud
      image: gcr.io/cloud-builders/gcloud
      command:
        - cat
      tty: true
    - name: kubectl
      image: gcr.io/cloud-builders/kubectl
      command:
        - cat
  """
}

```

```

tty: true
#####
}
}
stages {
stage('Build and push image with Container Builder') {
steps {
container('gcloud') {
sh "PYTHONUNBUFFERED=1 gcloud builds submit -t ${IMAGE_TAG} ."
}
}
}
stage('Deploy Canary') {
when { branch 'canary' }
steps {
container('kubectl') {
sh("sed -i.bak 's#gcr.io/hybrid-bts/bts-web#${IMAGE_TAG}#' ./k8s/canary/*.yaml")
step([$class: 'KubernetesEngineBuilder', namespace:'production', projectId:
env.PROJECT, clusterName: env.CLUSTER, location: env.CLUSTER_LOCATION,
manifestPattern: 'k8s/services', credentialsId: env.JENKINS_CRED, verifyDeployments:
false])
step([$class: 'KubernetesEngineBuilder', namespace:'production', projectId:
env.PROJECT, clusterName: env.CLUSTER, location: env.CLUSTER_LOCATION,
manifestPattern: 'k8s/canary', credentialsId: env.JENKINS_CRED, verifyDeployments:
true])
sh("echo http://`kubectl --namespace=production get service/${SVC_NAME} -o
jsonpath='{.status.loadBalancer.ingress[0].ip}` > ${SVC_NAME}'")
}
}
}
stage('Deploy Production') {

```

```

when { branch 'master' }

steps{
    container('kubectl') {
        sh("sed -i.bak 's#gcr.io/hybrid-bts/bts-
web#${IMAGE_TAG}#' ./k8s/production/*.yaml")

        step([$class: 'KubernetesEngineBuilder', namespace:'production', projectId:
env.PROJECT, clusterName: env.CLUSTER, location: env.CLUSTER_LOCATION,
manifestPattern: 'k8s/services', credentialsId: env.JENKINS_CRED, verifyDeployments:
false])

        step([$class: 'KubernetesEngineBuilder', namespace:'production', projectId:
env.PROJECT, clusterName: env.CLUSTER, location: env.CLUSTER_LOCATION,
manifestPattern: 'k8s/production', credentialsId: env.JENKINS_CRED, verifyDeployments:
true])

        sh("echo http://`kubectl --namespace=production get service/${SVC_NAME} -o
jsonpath='{{.status.loadBalancer.ingress[0].ip}}` > ${SVC_NAME}")
    }
}
}

stage('Deploy Dev') {
    when {
        not { branch 'master' }
        not { branch 'canary' }
    }
    steps {
        container('kubectl') {
            sh("kubectl get ns ${env.BRANCH_NAME} || kubectl create ns
${env.BRANCH_NAME}")

            sh("sed -i.bak 's#LoadBalancer#ClusterIP#' ./k8s/services/${APP_NAME}-
service.yaml")

            sh("sed -i.bak 's#gcr.io/hybrid-bts/bts-web#${IMAGE_TAG}#' ./k8s/dev/*.yaml")
        }
    }
}

```

```
        step([$class: 'KubernetesEngineBuilder', namespace: "${env.BRANCH_NAME}",
projectId: env.PROJECT, clusterName: env.CLUSTER, location:
env.CLUSTER_LOCATION, manifestPattern: 'k8s/services', credentialsId:
env.JENKINS_CRED, verifyDeployments: false])

        step([$class: 'KubernetesEngineBuilder', namespace: "${env.BRANCH_NAME}",
projectId: env.PROJECT, clusterName: env.CLUSTER, location:
env.CLUSTER_LOCATION, manifestPattern: 'k8s/dev', credentialsId: env.JENKINS_CRED,
verifyDeployments: true])

    }

}

}

}

}
```

CI/CD process – Dev environment

```
git checkout -b new-feature
*소스코드 수정
git push origin new-feature
```

CI/CD process – Canary environment

```
git checkout canary
git merge new-feature
git push origin canary
```

CI/CD process – Production environment

```
git checkout master  
git merge canary  
git push origin master  
  
git push origin :new-feature  
kubectl delete ns new-feature
```

CI/CD Results

```
DONE  
-----  
ID           CREATE_TIME      DURATION   SOURCE  
IMAGES       STATUS  
8c12d3ce-95df-4bd0-a06f-ed90cebdiale 2021-12-26T16:02:35+00:00 58S      gs://hybrid-bts_cloudbuild/source/1640534547.842045-  
5e632d10b8f044ffa2b9a7bf3f9328e1.tgz gcr.io/hybrid-bts/bts-web:master.2 SUCCESS  
  
[Pipeline] stage  
[Pipeline] { (Deploy Production)  
[Pipeline] container  
[Pipeline] {  
[Pipeline] sh  
+ sed -i.bak s#gcr.io/hybrid-bts/bts-web#gcr.io/hybrid-bts/bts-web:master.2# ./k8s/production/bts-web-production.yaml  
[Pipeline] step  
[Pipeline] step  
Verifying manifests: /home/jenkins/agent/workspace/bts-web_master/k8s/production  
Verifying 2 objects:  
Verifying: apps/v1/Deployment: bts-web-production  
Successfully verified apps/v1/Deployment: bts-web-production  
AvailableReplicas = 1, MinimumReplicas = 1  
  
Verifying: apps/v1/Deployment: bts-web-production  
Successfully verified apps/v1/Deployment: bts-web-production  
AvailableReplicas = 1, MinimumReplicas = 1
```

```
[Pipeline] sh
+ kubectl --namespace=production get service/bts-web-service -o jsonpath={.status.loadBalancer.ingress[0].ip}
+ echo http://34.64.184.156
[Pipeline] }
[Pipeline] // container
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy Dev)
Stage "Deploy Dev" skipped due to when conditional
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] }
[Pipeline] // podTemplate
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Big Data

Dataproc, SQL, Google Cloud Storage, PySpark, Python

Products 에 대한 고객의 ratings 데이터셋에 머신러닝 모델을 적용하여, 각 사용자에게 적합한 Recommendation 데이터셋을 생성하였습니다. GCS 에 있는 csv 파일을 SQL 로 옮기고, Dataproc 을 사용하여 Data transformation 을 하였습니다. Python 스크립트로 SQL 데이터를 불러온 뒤, PySpark 머신러닝 모델을 train 하고 적용한 데이터셋을 생성하여 SQL 에 load 합니다.

Cloud SQL

```
gcloud beta sql instances create bts-products \
--database-version=MYSQL_5_7 --cpu=1 --memory=3840MB \
--network=projects/hybrid-bts/global/networks/bts-vpc \
```

```
--region=asia-northeast3 --root-password=admin123

gcloud sql connect bts-products --user=root --quiet;

CREATE DATABASE IF NOT EXISTS product_spark;
USE product_spark;
DROP TABLE IF EXISTS Recommendation;
DROP TABLE IF EXISTS Rating;
DROP TABLE IF EXISTS Product;

CREATE TABLE IF NOT EXISTS Product;
(
    id varchar(255),
    title varchar(255),
    location varchar(255),
    price int,
    options int,
    rating float,
    type varchar(255),
    PRIMARY KEY (ID)
);
CREATE TABLE IF NOT EXISTS Rating
(
    userId varchar(255),
    prodId varchar(255),
    rating int,
    PRIMARY KEY(prodId, userId),
    FOREIGN KEY (prodId)
        REFERENCES Products(id)
);
CREATE TABLE IF NOT EXISTS Recommendation
```

```
(  
    userId varchar(255),  
    prodId varchar(255),  
    prediction float,  
    PRIMARY KEY(userId, prodId),  
    FOREIGN KEY (prodId)  
        REFERENCES Products(id)  
);
```

Google Cloud Storage

```
gsutil mb gs://bts-product-info  
gsutil cp product.csv gs://bts-product-info  
gsutil cp rating.csv gs://bts-product-info  
SA_SQL=$(gcloud sql instances describe bts-products | grep  
serviceAccountEmailAddress)  
  
gsutil acl ch -u ${SA_SQL}:R gs://bts-product-info/rating.csv  
gsutil acl ch -u ${SA_SQL}:R gs://bts-product-info/products.csv  
  
gcloud sql import csv bts-rating gs://bts-product-info/rating.csv \  
--database=rating_spark \  
--table=Rating;  
  
gcloud sql import csv bts-product gs://bts-product-info/products.csv \  
--database=product_spark \  
--table=Product;
```

```
dntwkzz79@cloudshell:~ (hybrid-bts)$ gcloud sql import csv bts-products gs://bts-product-info/rating.csv --database=products_spark --table=Rating;

Data from [gs://bts-product-info/rating.csv] will be imported to [bts-products]. 

Do you want to continue (Y/n)? y

Importing data into Cloud SQL instance...done.
Imported data from [gs://bts-product-info/rating.csv] into [https://sqladmin.googleapis.com/sql/v1beta4/projects/hybrid-bts/instances/bts-products].
```

Dataproc

```
gcloud dataproc clusters create bts-product \
--region=asia-northeast3

gcloud sql instances patch $CLOUDSQL --authorized-networks $ips
```

Bash script for authorizing Dataproc to connect to Cloud SQL

```
CLUSTER=bts-product
CLOUDSQL=bts-product
ZONE=asia-northeast3-a
NWORKERS=2
machines="$CLUSTER-m"
for w in `seq 0 $($NWORKERS - 1)`; do
    machines="$machines $CLUSTER-w-$w"
done

ips=""
for machine in $machines; do
    IP_ADDRESS=$(gcloud compute instances describe $machine --zone=$ZONE \
```

```
--format='value(networkInterfaces.accessConfigs[].natIP)' \  
| sed "s/['`]/\" | sed "s/'`]/\" )/32  
  
if [ -z $ips ]; then  
    ips=$IP_ADDRESS  
else  
    ips="$ips,$IP_ADDRESS"  
fi  
done
```

Machine Learning Model

```
gsutil cp train_and_apply.py gs://bts-product-info
```

train_and_apply.py

```
#!/usr/bin/env python  
  
import os  
import sys  
import pickle  
import itertools  
from math import sqrt  
from operator import add  
from os.path import join, isfile, dirname  
from pyspark import SparkContext, SparkConf, SQLContext  
from pyspark.mllib.recommendation import ALS, MatrixFactorizationModel, Rating  
from pyspark.sql.types import StructType, StructField, StringType, FloatType
```

```

CLOUDSQL_INSTANCE_IP = '104.155.188.32'
CLOUDSQL_DB_NAME = 'product_spark'
CLOUDSQL_USER = 'root'
CLOUDSQL_PWD = 'admin123'

conf = SparkConf().setAppName("train_model")
sc = SparkContext(conf=conf)
sqlContext = SQLContext(sc)
jdbcDriver = 'com.mysql.jdbc.Driver'
jdbcUrl = 'jdbc:mysql://%' + CLOUDSQL_INSTANCE_IP + '/%' + CLOUDSQL_DB_NAME + '?user=' + CLOUDSQL_USER + '&password=' + CLOUDSQL_PWD

sc.setCheckpointDir('checkpoint/')

dfRates = sqlContext.read.format('jdbc').options(driver=jdbcDriver, url=jdbcUrl,
dbtable='Rating', useSSL='false').load()
dfProd = sqlContext.read.format('jdbc').options(driver=jdbcDriver, url=jdbcUrl,
dbtable='Product', useSSL='false').load()
print("read ...")

model = ALS.train(dfRates.rdd, 20, 20)
print("trained ...")

allPredictions = None
for USER_ID in range(0, 100):
    dfUserRatings = dfRates.filter(dfRates.userId == USER_ID).rdd.map(lambda r:
r.accold).collect()
    rddPotential = dfAccos.rdd.filter(lambda x: x[0] not in dfUserRatings)
    pairsPotential = rddPotential.map(lambda x: (USER_ID, x[0]))

```

```

predictions = model.predictAll(pairsPotential).map(lambda p: (str(p[0]), str(p[1]),
float(p[2])))

predictions = predictions.takeOrdered(5, key=lambda x: -x[2]) # top 5

print("predicted for user={0}".format(USER_ID))

if (allPredictions == None):
    allPredictions = predictions
else:
    allPredictions.extend(predictions)

schema = StructType([StructField("userId", StringType(), True), StructField("prodId",
StringType(), True), StructField("prediction", FloatType(), True)])
dfToSave = sqlContext.createDataFrame(allPredictions, schema)
dfToSave.write.jdbc(url=jdbcUrl, table='Recommendation', mode='overwrite')

```

Submit Dataproc job

```

gcloud dataproc jobs submit pyspark \
gs://bts-product-info/train_and_apply.py \
--cluster=bts-product \
--region=asia-northeast3

```

```

mysql> select * from Products limit 10;
+----+-----+-----+-----+-----+-----+-----+
| id | title | category | price | options | rating | type |
+----+-----+-----+-----+-----+-----+-----+
| 1 | Comfy Quiet Chalet | Vancouver | 50 | 3 | 3.1 | cottage |
| 10 | Sizable Calm Country House | Auckland | 650 | 9 | 4.9 | mansion |
| 11 | Homy Quiet Shanty | Melbourne | 50 | 1 | 2.8 | cottage |
| 12 | Beautiful Peaceful Villa | Seattle | 90 | 2 | 2.1 | house |
| 13 | Enormous Peaceful Fortress | Melbourne | 3300 | 12 | 2.3 | castle |
| 14 | Colossal Peaceful Palace | Melbourne | 1200 | 21 | 1.5 | castle |
| 15 | Vast Private Fort | London | 1300 | 18 | 2.6 | castle |
| 16 | Large Calm House | Melbourne | 45 | 3 | 4.1 | house |
| 17 | Large Calm Sately House | NYC | 850 | 9 | 1.2 | mansion |
| 18 | Big Peaceful Hut | Melbourne | 60 | 2 | 2.4 | cottage |
+----+-----+-----+-----+-----+-----+-----+

```

```
mysql> select * from Rating limit 10;
+-----+-----+-----+
| userId | prodId | rating |
+-----+-----+-----+
| 10     | 1      | 1      |
| 13     | 1      | 1      |
| 18     | 1      | 2      |
| 12     | 10     | 3      |
| 18     | 10     | 1      |
| 21     | 10     | 2      |
| 4      | 10     | 1      |
| 1      | 11     | 1      |
| 10    | 11     | 1      |
| 11    | 11     | 1      |
+-----+-----+-----+
```

```
mysql> select * from Recommendation;
+-----+-----+-----+
| userId | prodId | prediction |
+-----+-----+-----+
| 12    | 99    | 2.8648407 |
| 0     | 76    | 3.2708821 |
| 0     | 75    | 3.2450237 |
| 12    | 49    | 2.7902086 |
| 12    | 72    | 2.7415967 |
| 0     | 66    | 3.2029953 |
| 0     | 49    | 3.167969  |
| 13    | 3     | 3.0129576 |
| 13    | 76    | 2.9712481 |
| 0     | 39    | 3.1403885 |
| 13    | 75    | 2.9348733 |
```

Big Data

Dataflow, Google Cloud Storage, Apache Beam, Python

고객의 주문 정보 데이터셋에서 거주 도시를 필터로 하여 데이터를 추출하였습니다. Google Cloud Storage에 저장된 데이터셋을 불러와, Apache Beam으로 추출한 데이터를 Google Cloud Storage에 저장하는 Python 스크립트를 Dataflow job으로 실행했습니다.

Install requirements

```
sudo apt-get install python3-pip  
sudo pip3 install apache-beam[gcp]==2.27.0  
sudo pip3 install oauth2client==3.0.0  
sudo pip3 install -U pip
```

Cloud Storage

```
export BUCKET=beam-df  
gsutil cp member_info.txt gs://beam-df  
  
SA_FLOW=service-786846853700@dataflow-accounts.iam.gserviceaccount.com  
gsutil acl ch -u ${SA_FLOW}:R gs://beam-df/member_info.txt  
gsutil acl ch -u ${SA_FLOW}:W gs://bts-product-info
```

nyc_grep.py

```
import apache_beam as beam  
  
def nyc_grep(line, term):  
    if line.startswith(term):  
        yield line  
  
PROJECT='hybrid-bts'
```

```

BUCKET='beam-df'

def run():
    argv = [
        '--project={0}'.format(PROJECT),
        '--job_name=nycgrep',
        '--save_main_session',
        '--staging_location=gs://{0}/staging/'.format(BUCKET),
        '--temp_location=gs://{0}/staging/'.format(BUCKET),
        '--region=asia-northeast3',
        '--runner=DataflowRunner'
    ]
    p = beam.Pipeline(argv=argv)
    input = 'gs://{0}/member_info.txt'.format(BUCKET)
    output_prefix = 'gs://{0}/output'.format(BUCKET)
    searchTerm = 'NYC'

    (p
        | 'GetCSV' >> beam.io.ReadFromText(input)
        | 'Grep' >> beam.FlatMap(lambda line: nyc_grep(line, searchTerm))
        | 'write' >> beam.io.WriteToText(output_prefix)
    )
    p.run()

if __name__ == '__main__':
    run()

```

Dataflow

```
python3 nyc_grep.py  
gsutil cp gs://beam_df/output*.  
cat output*
```

```
NYC NY 10022 USA NA Yu Kwai  
NYC NY 10022 USA NA Frick Michael  
NYC NY 10022 USA NA Yu Kwai  
NYC NY 10022 USA NA Hernandez Maria  
NYC NY 10022 USA NA Yu Kwai  
NYC NY 10022 USA NA Yu Kwai  
NYC NY 10022 USA NA Yu Kwai  
NYC NY 10022 USA NA Frick Michael  
NYC NY 10022 USA NA Yu Kwai  
NYC NY 10022 USA NA Yu Kwai
```

Big Data

Cloud Bigtable, Cloud Function

고객의 거주 지역에 대한 데이터셋에서 도시가 Paris 인 데이터를 반환합니다. Python script 를 사용하여 Bigtable 에 데이터를 삽입한 뒤, Cloud Function 의 HTTP trigger 를 사용해서 추출된 데이터를 얻습니다.

Cloud Bigtable

```
gcloud bigtable instances create members \  
--display-name=members \  
--cluster-config=id=members-cluster,zone=asia-northeast3-a,nodes=1
```

```
echo project = hybrid-bts > ~/.cbtrc
echo instance = members >> ~/.cbtrc
```

```
cbt createtable member-info
cbt ls member-info
```

Write data to Bigtable

```
go run write_members.go
```

```
-----
10107
  city:NYC
    "\x00\x00\x00\x00\x00\x00\x00\x01"
  country:USA
    "\x00\x00\x00\x00\x00\x00\x00\x01"
  last_name:John
    "\x00\x00\x00\x00\x00\x00\x00\x01"
-----
10121
  city:Paris
    "\x00\x00\x00\x00\x00\x00\x00\x01"
  country:France
    "\x00\x00\x00\x00\x00\x00\x00\x01"
  last_name:Young
    "\x00\x00\x00\x00\x00\x00\x00\x01"
```

write_members.go

```
package main

import (
  "bytes"
  "context"
  "encoding/binary"
```

```

"fmt"
"cloud.google.com/go/bigtable"
)

func writeBatch(projectID, instanceID string, tableName string) error {
    projectID = "hybrid-bts"
    instanceID = "members"
    tableName = "member-info"

    ctx := context.Background()
    client, err := bigtable.NewClient(ctx, projectID, instanceID)
    if err != nil {
        return fmt.Errorf("bigtable.NewAdminClient: %v", err)
    }
    defer client.Close()
    tbl := client.Open(tableName)

    timestamp := bigtable.Now()

    var muts []*bigtable.Mutation

    binary1 := new(bytes.Buffer)
    binary.Write(binary1, binary.BigEndian, int64(1))

    mut := bigtable.NewMutation()
    mut.Set("city", "NYC", timestamp, binary1.Bytes())
    mut.Set("country", "USA", timestamp, binary1.Bytes())
    mut.Set("last_name", "John", timestamp, binary1.Bytes())
    muts = append(muts, mut)
}

```

```

mut = bigtable.NewMutation()
mut.Set("city", "Paris", timestamp, binary1.Bytes())
mut.Set("country", "France", timestamp, binary1.Bytes())
mut.Set("last_name", "Young", timestamp, binary1.Bytes())
muts = append(muts, mut)

mut = bigtable.NewMutation()
mut.Set("city", "San Francisco", timestamp, binary1.Bytes())
mut.Set("country", "USA", timestamp, binary1.Bytes())
mut.Set("last_name", "Mayer", timestamp, binary1.Bytes())
muts = append(muts, mut)

mut = bigtable.NewMutation()
mut.Set("city", "Seoul", timestamp, binary1.Bytes())
mut.Set("country", "Korea", timestamp, binary1.Bytes())
mut.Set("last_name", "Park", timestamp, binary1.Bytes())
muts = append(muts, mut)

mut = bigtable.NewMutation()
mut.Set("city", "Liverpool", timestamp, binary1.Bytes())
mut.Set("country", "UK", timestamp, binary1.Bytes())
mut.Set("last_name", "Devon", timestamp, binary1.Bytes())
muts = append(muts, mut)

rowKeys := []string{"10107", "10121", "10134", "10145", "10159"}
if _, err := tbl.ApplyBulk(ctx, rowKeys, muts); err != nil {
    return fmt.Errorf("ApplyBulk: %v", err)
}
return nil
}

```

```
func main() {
    writeBatch("hybrid-bts", "members", "member-info")
}
```

Cloud Function

```
gcloud functions deploy Paris \
--runtime go116 --trigger-http
```

```
dntwkzz79@cloudshell:~ (hybrid-bts)$ curl "https://us-central1-hybrid-bts.cloudfunctions.net/Paris" -H "project_id:hybrid-bts" -H "instance_id:members" -H "table_id:member-info" --output ./test
% Total    % Received % Xferd  Average Speed   Time     Time      Current
          Dload  Upload   Total Spent  Left  Speed
100  170  100  170    0     0      80      0  0:00:02  0:00:02 --::--   80
dntwkzz79@cloudshell:~ (hybrid-bts)$ cat test
order number: 10107, for city:
order number: 10121, for city:  Paris
order number: 10134, for city:
order number: 10145, for city:
order number: 10159, for city:
```

Paris.go

```
package bigtable

import (
    "context"
    "fmt"
    "log"
    "net/http"
    "sync"
```

```

    "cloud.google.com/go/bigtable"
)

var client *bigtable.Client
var clientOnce sync.Once

func BigtableRead(w http.ResponseWriter, r *http.Request) {
    clientOnce.Do(func() {
        var err error
        client, err = bigtable.NewClient(context.Background(), r.Header.Get("project_id"),
r.Header.Get("instance_id"))

        if err != nil {
            http.Error(w, "Error initializing client", http.StatusInternalServerError)
            log.Printf("bigtable.NewClient: %v", err)
            return
        }
    })
}

tbl := client.Open(r.Header.Get("table_id"))
err := tbl.ReadRows(r.Context(), bigtable.PrefixRange("10"),
func(row bigtable.Row) bool {
    city := ""
    for _, col := range row["city"] {
        if col.Column == "city:Paris" {
            city = "Paris"
        }
    }
    fmt.Fprintf(w, "order number: %s, for city: %s\n", row.Key(), city)
    return true
})

```

```
if err != nil {  
    http.Error(w, "Error reading rows", http.StatusInternalServerError)  
    log.Printf("tbl.ReadRows(): %v", err)  
}  
}  
  
func main() {  
}
```