GCP INFRASTRUCTURE BUILD

Table of Contents

[VPC 1](#_Toc91606685)

[Firewall 2](#_Toc91606686)

[vm(bastion) 2](#_Toc91606687)

[GCS 3](#_Toc91606688)

[Cloud SQL 4](#_Toc91606689)

[IAM- Service Account 5](#_Toc91606690)

[Cloud DNS 6](#_Toc91606691)

[CSR 6](#_Toc91606692)

[Cloud Build 7](#_Toc91606693)

[GKE 8](#_Toc91606694)

[CI/CD 20](#_Toc91606695)

Virtual Private Cloud (VPC)

Custom VPC를 생성하여 public subnet과 private subnet을 생성했습니다. NAT 라우터와 NAT 게이트웨이를 생성하여 VPC의 프라이빗 인스턴스들이 인터넷에 접근 가능하도록 했습니다.

|  |
| --- |
| Create a private cluster |
| 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 --enable-private-ip-google-access |
| 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 |
| 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 |

Firewall

Custom VPC에 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 |

VM instance

관리자가 프라이빗 GKE 클러스터 내의 인스턴스들을 관리할 수 있도록 bastion host를 public subnet에 생성하였습니다. Bastion host에 kubectl 패키지를 설치하였습니다.

|  |
| --- |
| Create |
| gcloud compute instances create bts-bastion-vm --zone=asia-northeast3-a \  --machine-type=f1-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 |
| 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 e2-medium  gcloud compute instances list --sort-by=ZONE  gcloud compute ssh bts-bastion-vm --zone asia-northeast3-a --tunnel-through-iap |

|  |
| --- |
| Install packages |
| sudo apt-get update sudo apt-get install -y apt-transport-https ca-certificates curl    sudo curl -fsSLo /usr/share/keyrings/kubernetes-archive-keyring.gpg <https://packages.cloud.google.com/apt/doc/apt-key.gpg>  echo"deb [signed-by=/usr/share/keyrings/kubernetes-archive-keyring.gpg] <https://apt.kubernetes.io/> kubernetes-xenial main"| sudo tee /etc/apt/sources.list.d/kubernetes.list    sudo apt-get update sudo apt-get install -y kubectl |

Google Cloud Storage(GCS)

웹사이트의 static contents들을 저장할 버킷을 생성하고 오브젝트에 ACL을 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 . |

|  |
| --- |
| ACL |
| export BUCKET\_NAME\_1=bts-static    gsutil cp album.svg gs://$BUCKET\_NAME\_1/  gsutil acl get gs://$BUCKET\_NAME\_1/ album.svg > acl.txt  cat acl.txt    gsutil acl set private gs://$BUCKET\_NAME\_1/album.svg  gsutil acl get gs://$BUCKET\_NAME\_1/album.svg > acl2.txt  cat acl2.txt    gsutil acl ch -u AllUsers:R gs://$BUCKET\_NAME\_1/album.svg  gsutil acl get gs://$BUCKET\_NAME\_1/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-2 --database-version=MYSQL\_5\_7 \  --cpu=1 --memory=3840MB \  --network=projects/hybrid-bts/global/networks/bts-vpc \  --region=asia-northeast3 --root-password=admin123 --no-assign-ip    gcloud sql instances list    gcloud sql instances delete bts-sql |

|  |
| --- |
| Connect |
| gcloud auth activate-service-account vm-ser-acc@hybrid-bts.iam.gserviceaccount.com \  --key-file=/home/dntwkzz79/hybrid-bts-97203600b216.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 제어 권한을 가질 수 있도록 service account를 생성했습니다.

|  |
| --- |
|  |
| gcloud auth activate-service-account vm-ser-acc@hybrid-bts.iam.gserviceaccount.com \  --key-file='./credentials.json' --project=hybrid-bts    kubectl create clusterrolebinding clu-admin \      --clusterrole=cluster-admin --serviceaccount=default:vm-ser-acc    gcloud iam service-accounts list    gcloud iam service-accounts create jenkins-admin \      --display-name="jenkins-admin"    gcloud iam service-accounts keys create key.json --iam-account= jenkins-admin @hybrid-bts.iam.gserviceaccount.com    gcloud projects add-iam-policy-binding $PROJECT\_ID \      --member="serviceAccount: jenkins-admin@hybrid-bts.iam.gserviceaccount.com" \      --role="roles/container.admin" |

Cloud DNS

웹서버에 hybridbts.tech 도메인을 연결하였습니다.

|  |
| --- |
| Create DNS records |
| gcloud beta dns --project=hybrid-bts managed-zones create hybridbts \  --dns-name="hybridbts.tech." --visibility="public" --dnssec-state="off"    gcloud dns record-sets create hybridbts.tech. --rrdatas="34.102.162.191" --type=A --ttl=60 \  --zone="hybridbts"    gcloud dns record-sets create [www.hybridbts.tech](http://www.hybridbts.tech). --rrdatas="34.102.162.191" \  --type=CNAME --ttl=60 \  --zone="hybridbts" |

Cloud Source Repository(CSR)

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

|  |
| --- |
| Repository |
| gcloud source repos create 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 |

Cloud Build

Cloud Build로 애플리케이션 배포의 CI/CD를 했습니다.

|  |
| --- |
| Build submit |
| docker pull python    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 |

|  |
| --- |
| 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 GKE에 배포하고 운영했습니다. 프라이빗 클러스터로 생성하여 ingress를 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 vm[-ser-acc@hybrid-bts.iam.gserviceaccount.com](https://console.cloud.google.com/iam-admin/serviceaccounts/details/114048810555998278566?orgonly=true&project=hybrid-bts&supportedpurview=project,organizationId,folder) \      --enable-master-global-access \      --enable-master-authorized-networks \      --enable-ip-alias \      --enable-private-nodes \      --enable-private-endpoint \      --master-ipv4-cidr 10.2.0.0/28  gcloud container clusters update $bts\_cluster --region $bts\_region \      --enable-master-authorized-networks \      --master-authorized-networks 10.0.0.2/32  gcloud container clusters resize $bts\_cluster --region=$bts\_region --num-nodes=0 |

|  |
| --- |
| 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-jobs  spec:  containers:  - name: sample-container  image: gcr.io/hybrid-bts/bts-web  restartPolicy: OnFailure |

|  |
| --- |
| 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/dev  kubectl --namespace=production apply -f k8s/services    kubectl --namespace=production scale deployment bts-web-production --replicas=4  kubectl get deployments    kubectl describe pod bts-web-canary --namespace=production  kubectl get pods --namespace=production  kubectl --namespace=production get service bts-web-service  kubectl describe deployments --namespace=production |

|  |
| --- |
| 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: Always  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: Always  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: Always  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-global-ingress"  nginx.ingress.kubernetes.io/rewrite-target: /  spec:  rules:  - http:  paths:  - path: /  pathType: Prefix  backend:  service:  name: bts-web-service  port:  number: 80 |

|  |
| --- |
| Deployment rollout & rollback |
| kubectl rollout status bts-web-production  kubectl rollout history deployment bts-web-production    kubectl rollout undo deployments bts-web-production  kubectl rollout history deployment bts-web-production |

|  |
| --- |
| 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 |

CI/CD pipeline

Jenkins를 사용하여 GSR의 소스코드가 업데이트 되면 자동으로 클러스터에 배포가 이루어지도록 CI/CD pipeline을 생성했습니다. Dev environment를 사용하여 CD를 하고 Canary environment를 사용하여 카나리 배포로 테스트 후에 Production environment에 최종적으로 배포가 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 |

|  |
| --- |
| Jenkins/values.yaml |
| controller:  installPlugins:  - kubernetes:latest  - workflow-job:latest  - workflow-aggregator:latest  - credentials-binding:latest  - git:latest  - google-oauth-plugin:latest  - google-source-plugin:latest  - google-kubernetes-engine:latest  - google-storage-plugin:latest  resources:  requests:  cpu: "50m"  memory: "1024Mi"  limits:  cpu: "1"  memory: "3500Mi"  javaOpts: "-Xms3500m -Xmx3500m"  serviceType: ClusterIP  agent:  resources:  requests:  cpu: "500m"  memory: "256Mi"  limits:  cpu: "1"  memory: "512Mi"  persistence:  size: 100Gi  serviceAccount:  name: 'vm-ser-acc' |

|  |
| --- |
| Connect to Jenkins |
| gcloud container clusters get-credentials jenkins-cd-priv1 --region asia-northeast3    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 |
| Graphical user interface, text, application, email  Description automatically generated |

|  |
| --- |
| Configure clouds |
| Chart  Description automatically generated with medium confidence  Chart  Description automatically generated with medium confidence |

|  |
| --- |
| Create a multibranch pipeline job |
| Graphical user interface, text, application  Description automatically generated |

|  |
| --- |
| Jenkinsfile |
| pipeline {  environment {  PROJECT = "hybrid-bts"  APP\_NAME = "bts-web"  SVC\_NAME = "${APP\_NAME}-service"  CLUSTER = "bts-cluster-10"  CLUSTER\_LOCATION = "asia-northeast3"  IMAGE\_TAG = "gcr.io/${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: 'vm-ser-acc'  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 ${IMAGE\_TAG} ."  }  }  }  stage('Deploy Canary') {  // Canary branch  when { branch 'canary' }  steps {  container('kubectl') {  // Change deployed image in canary to the one just built  sh("sed -i.bak 's#gcr.io/hybrid-bts/bts-web#${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}'` > ${SVC\_NAME}")  }  }  }  stage('Deploy Production') {  // Production branch  when { branch 'master' }  steps{  container('kubectl') {  // Change deployed image in canary to the one just built  sh("sed -i.bak 's#gcr.io/hybrid-bts/bts-web#${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') {  // Developer Branches  when {  not { branch 'master' }  not { branch 'canary' }  }  steps {  container('kubectl') {  // Create namespace if it doesn't exist  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])  }  }  }  }  } |

|  |
| --- |
| cluster-role-binding.yaml |
| apiVersion: rbac.authorization.k8s.io/v1  kind: ClusterRoleBinding  metadata:  name: sa-role-binding  subjects:  - kind: ServiceAccount  name: vm-ser-acc  namespace: default  roleRef:  kind: ClusterRole  name: cluster-admin  apiGroup: rbac.authorization.k8s.io  kubectl apply -f cluster-role-binding.yaml |

|  |
| --- |
| Jenkins-ingress.yaml |
| apiVersion: networking.k8s.io/v1  kind: Ingress  metadata:  name: jenkins-ingress  annotations:  kubernetes.io/ingress.global-static-ip-name: "jenkins-ingress"  nginx.ingress.kubernetes.io/rewrite-target: /  spec:  rules:  - http:  paths:  - path: /  pathType: Prefix  backend:  service:  name: cd-jenkins  port:  number: 8080 |

|  |
| --- |
| CI/CD process – Dev environment |
| git checkout -b new-feature  git push origin new-feature    kubectl proxy &    curl <http://localhost:8001/api/v1/namespaces/new-feature/services/bts-web-service:80/proxy/> |

|  |
| --- |
| 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 |
| Graphical user interface, text, application, email  Description automatically generated  Graphical user interface, text, application, email  Description automatically generated |